EXAMPLE **Service Specific Connection Oriented Protocol**

The Service Specific Connection Oriented Protocol (SSCOP) provides error control for signaling messages in ATM networks. SSCOP uses a variation of Selective Repeat ARQ that detects the loss of information by monitoring the sequence number of blocks of information that have been received and requesting selective retransmission. The ARQ protocol in SSCOP was originally designed for use in high-speed satellite links. These links have a large delay-bandwidth product, so the protocol was also found useful in ATM networks, which also exhibit large delay-bandwidth product. SSCOP is discussed in Chapter 9.

PERFORMANCE ISSUES

Selective Repeat ARQ is the most efficient of the ARQ protocols as it uses the minimum number of retransmissions. To assess the performance of Selective Repeat ARQ, we note that any given frame transmission is received correctly with probability $1 - P_f$, independently of all other frame transmissions. Thus the average number of times a frame needs to be sent before it is received correctly is $1/(1 - P_f)$, and so the average transmission time is:

$$t_{SR} = \frac{t_f}{1 - P_f} \tag{5.10}$$

Equation (5.3) then gives the following simple expression for the efficiency of Selective Repeat ARQ:

$$\eta_{SR} = \frac{\frac{n_f - n_o}{t_{SR}}}{R} = \left(1 - \frac{n_o}{n_f}\right)(1 - P_f). \tag{5.11}$$

If the frame error rate is zero, then we get the best possible efficiency, $1 - n_o/n_f$. Note that the expression is valid only if the send window size is larger than the delay-bandwidth product. If the window size is too small, then the transmitter may run out of sequence numbers from time to time, and the efficiency will then be reduced.

EXAMPLE **Effect of Bit Error Rate and Delay-Bandwidth Product**

Suppose that frames are 1250 bytes long including 25 bytes of overhead, and that ACK frames are 25 bytes long. Compare the efficiency of Go-Back-N and Selective Repeat ARQ in a system that transmits at $R = 1$ Mbps with reaction time of 100 ms, and for bit error rate of 10^{-6}, 10^{-5}, and 10^{-4}.

This example is a continuation of the previous bit rate examples. The delay-bandwidth product of this channel is 10^5 bits, and the probability of successful frame transmissions are $1 - P_f = 0.99$, 0.905, and 0.368 for $p = 10^{-6}$, $p = 10^{-5}$, and $p = 10^{-4}$, respectively. The corresponding efficiencies for Selective Repeat ARQ are then 97%, 89%, and 36%. These are significant improvements over Go-Back-N, which has corresponding efficiencies of 88.2%, 45.4%, and 4.9%.

COMPARISON OF STOP-AND-WAIT, GO-BACK-N, AND SELECTIVE REPEAT ARQ

Stop-and-Wait, Go-Back-N, and Selective Repeat ARQ provide a range of choices in terms of implementation complexity and transmission efficiency performance. Consider the parameters that affect efficiency: header and CRC overhead, delay-bandwidth product, frame size, and frame error rate. The header and CRC overhead are negligible as long as the frames are not too short. If we neglect the header and CRC overhead, then the efficiency expressions for the three protocols simplify to:

$$\eta_{SR} = (1 - P_f) \qquad \eta_{GBN} = \frac{(1 - P_f)}{1 + LP_f} \qquad \eta_{SW} = \frac{1 - P_f}{1 + L} \tag{5.12}$$

where $L = 2(t_{prop} + t_{proc})R/n_f$ is the size of the "pipe" in multiples of frames, and where we have assumed that the send window size is $W_S = L + 1$. These simple equations allow us to examine the main factors in play.

The above expression for Selective Repeat ARQ shows that the efficiency performance is ultimately limited by the frame error rate P_f. The only way to improve performance is by improving P_f. This may be done, for example, by including some error correction capability into the coding of the frame. The expressions show that Selective Repeat ARQ can act as a benchmark for Go-Back-N and Stop-and-Wait. Go-Back-N is worse than Selective Repeat by a factor of $1 + LP_f$, and Stop-and-Wait is worse than Selective Repeat by a factor of $1 + L$. When LP_f is very small, Go-Back-N has almost the same performance as Selective Repeat. On the other hand when P_f increases and approaches 1, Go-Back-N has the performance of Stop-and-Wait. The above observations are confirmed in Figure 5.24, which shows the efficiencies of the three protocols for a channel with random bit errors, a frame size of 1250 bytes, and delay-bandwidth product of 10 and 100.

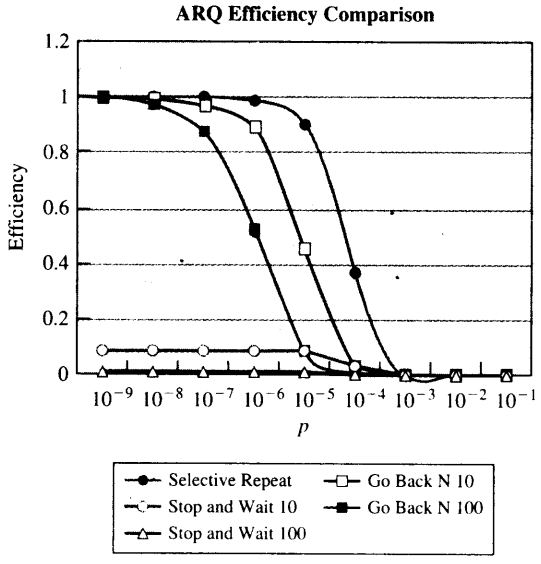


FIGURE 5.24 Efficiency performance of ARQ protocols as a function of bit error rate for a frame size of 1250 bytes and for channels with delay-bandwidth product of 10 and 100 frames.

ARQ: ROBUSTNESS AND ADAPTIVITY

Paul Green, in providing his list of remarkable milestones in computer communications in 1984, made the following remark about ARQ: "(ARQ) is one of those disarmingly simple ideas that seems so trivial (particularly in hindsight) that it really shouldn't be on anyone's list of exciting ideas." ARQ was invented by H. C. A. van Duuren during World War II to provide reliable transmission of characters over radio. In his system each seven-bit character consisted of a combination of four marks (1s) and three spaces (0s). The reception of any other combination of bits led to a request for retransmission. This simple system led to the development of the many modern protocols that have ARQ as their basis.

Simplicity is not the only reason for the success of ARQ. Adaptivity and robustness are equally important attributes of ARQ. The retransmission mechanism gives ARQ the ability to adapt to variable and time-varying channel conditions. If a channel is clean, then ARQ operates in an efficient manner. If a channel becomes noisy, then ARQ adapts the transmission rate to the capability of the channel. This adaptivity makes ARQ relatively robust with respect to channel conditions and hence safe to deploy without detailed knowledge of the channel characteristics.

5.3 OTHER PEER-TO-PEER PROTOCOLS

In this section we show how the various elements of the ARQ protocols can be used in other protocols to provide other types of services. In particular, we introduce protocols that can provide flow control, reliable stream service, and synchronization and timing information. We also introduce the TCP protocol for providing reliable stream service end-to-end across a network.

5.3.1 Sliding-Window Flow Control

In situations where a transmitter can send information faster than the receiver can process it, messages can arrive at the receiver and be lost because buffers are unavailable. **Flow control** refers to the procedures that prevent a transmitter from overrunning a receiver's buffer.

The simplest procedure for exercising flow control is to use signals that direct the sender to stop transmitting information. Suppose process A is transmitting to process B at a rate of R bps. If process B detects that its buffers are filling up, it issues a stop signal to process A. After approximately one propagation delay T_{prop}, process A stops transmitting as shown in Figure 5.25. From the instant that B sent its signal, it receives an additional $2T_{prop}R$ bits, which is equal to the delay-bandwidth product of the link. Thus process B must send the off signal when its buffer contents exceed a threshold value. For example, this type of flow control is used in the X-ON/X-OFF protocol that

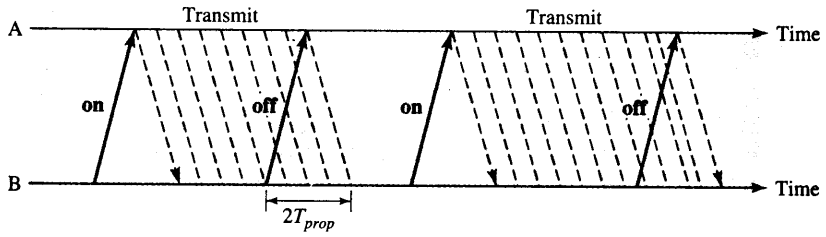


FIGURE 5.25 ON-OFF flow control.

is used between a terminal and a computer. This type of control is also used in various data link controls.

The sliding-window protocols that were introduced in Section 5.2 for ARQ can also be used to provide flow control. In the simplest case the size of the send window W_S is made equal to the number of buffers that are available at the receiver for messages from the given transmitter. Because W_S is the maximum number of outstanding frames from the transmitter, buffer overflow cannot occur at the receiver. When the sliding window protocol is used for flow control, each acknowledgment of a frame can be viewed as an issuing of a credit by the receiver that authorizes the transmitter to send another frame.

Figure 5.26 shows an example where the receiver sends an acknowledgment after the last frame in a window has been received. In this figure t_{cycle} is the basic delay that elapses from the time the first frame is transmitted to the receipt of its acknowledgment. For this example we have $W_S t_f < t_{cycle}$ where $W_S = 2$ and t_f is the time to transmit a frame. The delay in sending acknowledgments has the effect of pacing or controlling the rate at which the transmitter sends messages to the receiver. The average rate at which the sender sends frames is then W_S / t_{cycle} frames/second.

The sliding window mechanism can be used to simultaneously provide *both* error control and flow control. However the pacing shown in Figure 5.26 can be disrupted by retransmissions from the error-control function. In addition, the choice of window size must take into account the delay-bandwidth product of the channel as well as the buffer size of the receiver. This situation shows how limitations can arise when the same mechanism (sliding-window control) is used for more than one purpose, namely, error control and flow control. For this reason, special control messages are also used to direct a sender to stop sending frames and later to resume sending frames.

It is also possible to decouple acknowledgments of received frames from the issuing of transmitter credits. In this approach separate fields in the header are used to acknowledge received information and to issue transmission credits. The transmission

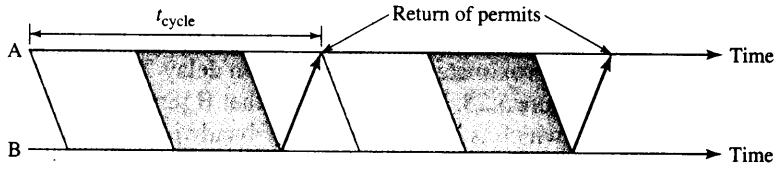


FIGURE 5.26 Sliding-window flow control.

credits authorize the sender to send a certain amount of information in addition to the information that has already been acknowledged. In effect, the receiver is advertising a window of information that it is ready to accept. This approach is used by TCP.

5.3.2 Timing Recovery for Synchronous Services

Many applications involve information that is generated in a synchronous and periodic fashion. For example, digital speech and audio signals generate samples at the corresponding Nyquist sampling rate of 8 kHz and 44 kHz, respectively. Video signals generate blocks of compressed information corresponding to an image 30 times a second. To recover the original signal at the receiver end, the information must be input into the decoder at the same rate at which it was produced at the encoder. Figure 5.27 shows the effect of a transmission over a network on the temporal relationship of the original information blocks. In general networks will introduce a variable amount of delay, and so certain blocks will arrive relatively earlier or later than others, resulting in break up of the periodicity in the sequence, as indicated in the figure. This effect is referred to as **timing jitter**. In this section we consider **timing recovery** protocols that can be used to restore the original timing relationship of the sequence of information to allow the decoder to operate correctly.

The first thing that can be done is to provide relative timing information in the sequence of transmitted blocks by inserting a timestamp in each block that enters the network. These timestamps can be used at the receiver to determine both the sequencing of the blocks as well as their relative timing. The typical approach in the "playout" procedure is to select some fixed delay target $T_{playout}$ that exceeds the total delay that can be experienced by blocks traversing the network. The objective is to play out all the blocks of information so that the total delay, including the network delay, is constant and equal to $T_{playout}$. Thus each time a block of information arrives, an additional delay is calculated, using the difference in the timestamps of the current block and the preceding block.

Most applications have a nominal clock frequency that is known to the transmitter and the receiver, for example, 8 kHz for telephone speech, 44 kHz for audio, or 27 MHz for television. This clock frequency dictates the rate at which samples are to be played out. However, it is extremely unlikely that the clock at the receiver will be exactly synchronized to the clock at the transmitter. Figure 5.28 shows the effect of differences in clock rate. The figure shows the times at which the information blocks were produced, and the arrows indicate the total delay that was encountered in the network. When the receiver clock is slow relative to the transmitter clock, the receiver will play its samples at a rate slower than they were produced. Consequently, over a period of time the receiver

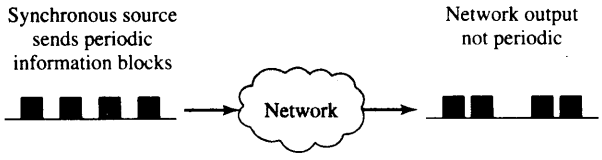


FIGURE 5.27 Timing recovery.

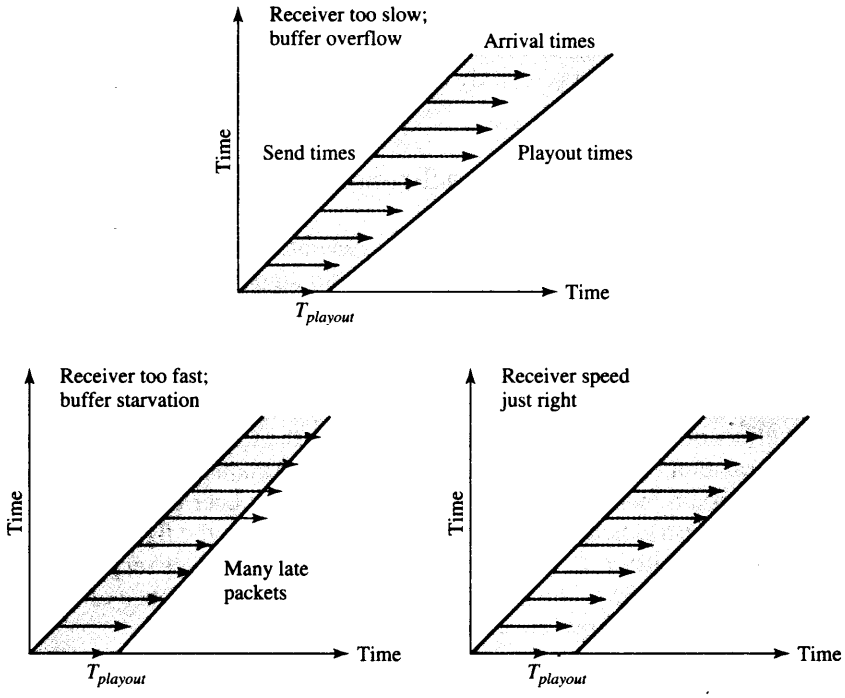


FIGURE 5.28 Clock rate differences and their effect on transmission.

buffer will grow, eventually leading to loss of information due to buffer overflow. On the other hand, when the receiver is too fast, the receiver buffer will gradually empty and the playout procedure will be interrupted due to lack of available samples. These examples show why the timing recovery protocol must also include a clock recovery procedure that attempts to also synchronize the receiver clock to the transmitter clock.

Many techniques have been proposed for carrying out clock recovery. We will discuss two representative approaches. Figure 5.29 shows a system in which the sequence

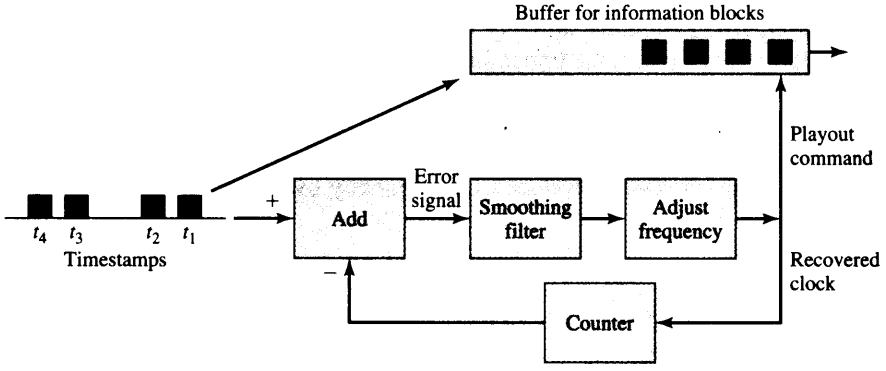


FIGURE 5.29 Adaptive clock recovery.

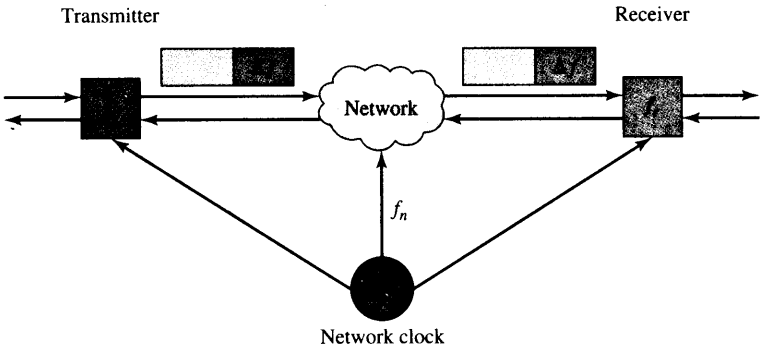


FIGURE 5.30 Clock recovery with synchronous network.

of timestamp values is used to perform clock recovery. The timestamps in the arriving blocks of information were generated by sampling a counter that is driven by the transmitter clock. The receiver system has a counter that attempts to synchronize to the transmitter clock. The sequence of timestamp values is compared to the local counter values to generate an error signal that is indicative of the difference between the transmitter and receiver clocks. This error signal is filtered and used to adjust the frequency of the local clock. If the difference signal indicates that the local clock is slow, then the frequency of the local clock is increased. If the difference indicates that the local clock is fast, then the frequency of the local clock is reduced. The recovered clock is then used to control the playout from the buffer.

When the local clock is synchronized to the transmitter clock the buffer contents will vary about some constant value. Variations in the buffer contents occur according to the delay jitter experienced by the blocks in traversing the network. Therefore, the size of the buffer must be designed to accommodate the jitter that is experienced in the given network.

Another very effective clock recovery procedure can be used when the transmitter and receiver are connected to a network in which all the elements are synchronized to a common clock, as shown in Figure 5.30. For example, many networks now use the Global Positioning System (GPS) to provide this capability.[7] The transmitter can then compare its frequency f_s to the network frequency f_n to obtain difference Δf. The difference Δf is calculated as follows. Every time the transmitter clock completes N cycles, the number of cycles M completed by the network clock is found. The measured value M is transmitted to the receiver. The receiver deduces Δf by using the fact that the frequencies are related by the following equation:

$$\frac{f_n}{f_s} = \frac{1/M}{1/N} \tag{5.13}$$

[7]GPS was developed by the U.S. military to provide satellite signals that can be processed to enable a receiver to compute position, velocity, and time.

The receiver then controls the playout using the frequency f_r, which is given by

$$f_r = f_n - \Delta f \tag{5.14}$$

The advantage of this method is that the jitter that takes place within the network does not at all affect the timing recovery procedure, as was the case in Figure 5.29. The basic approach presented here has been made more efficient through a technique called the *synchronous residual timestamp (SRTS)* method. This method has been incorporated in standards for timing recovery in ATM networks that operate over SONET.

EXAMPLE **Real-Time Transport Protocol**

The Real-Time Transport Protocol (RTP) is an application layer protocol designed to support real-time applications such as videoconferencing, audio broadcasting, and Internet telephony. RTP usually operates over UDP, but it can work over connection-oriented or connectionless lower-layer protocols. RTP is concerned only with providing a mechanism for the transfer of information regarding source type, sequence numbering, and timestamps. RTP itself does not implement the procedures for performing timing recovery. This function must be done by applications that operate on top of RTP. RTP is discussed in Chapter 10.

5.3.3 TCP Reliable Stream Service and Flow Control

We now introduce the TCP protocol that provides connection-oriented reliable stream service.[8] As in the case of ARQ protocols discussed in previous sections, we are interested in delivering the user information so that it is error free, without duplication, and in the order produced by the sender. However, the user information does not necessarily consist of a sequence of information blocks, but instead consists of a stream of bytes,[9] that is, groups of eight bits, as shown in Figure 5.31. For example, in the transfer of a long file the sender is viewed as inserting a byte stream into the transmitter's send buffer. The task of the TCP protocol is to ensure the transfer of the byte stream to the receiver and the orderly delivery of the stream to the destination application.

TCP was designed to deliver a connection-oriented service over IP that itself offers connectionless packet transfer service. In this environment each packet that is transmitted between a sender and receiver machine can traverse a different path and packets can therefore arrive out of order. Unlike "wirelike" links considered in previous sections, in the Internet it is possible for old segments from previous connections to arrive at a receiver, thus potentially complicating the task of eliminating duplicate messages. TCP deals with this problem by using long (32 bit) sequence numbers and by establishing randomly selected initial sequence numbers during connection setup. At any given time the receiver is accepting sequence numbers from a much smaller window, so the

[8]TCP is discussed in detail in Chapter 8.

[9]The term *octet* is used for an eight-bit byte in RFC 793 in which TCP is defined.

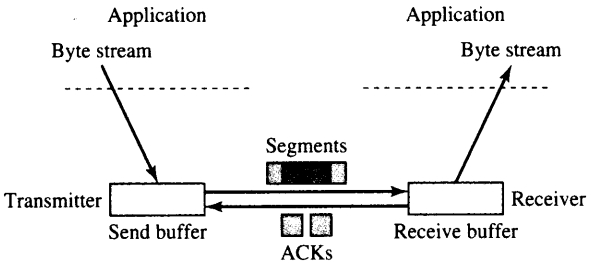


FIGURE 5.31 TCP preview.

likelihood of accepting a very old message is very low. In addition, TCP enforces a time-out period at the end of each connection to allow the network to clear old segments from the network.

Figure 5.32 shows the segments that are exchanged in the "three-way handshake" used to set up a TCP connection and establish the initial sequence numbers. The first segment has the SYN bit in the TCP header set indicating that this is a connection setup request from the ephemeral port 2743 in the machine with IP address 65.95.113.77 to the well-known port 23 (Telnet) in the machine with IP address 128.113.25.22. The segment indicates an initial sequence number 1839733355. The second frame contains the response segment with the ACK bit in its header set to 1, and the acknowledgment sequence number set to $1839733355 + 1 = 1839733356$ to acknowledge the first segment. The pair of SYN segment and ACK response complete the setup of the TCP connection in one direction. The second segment also has the SYN bit set to 1 to request the connection in the opposite direction and to propose initial sequence number 1877388864. The third segment completes the handshake by acknowledging the second

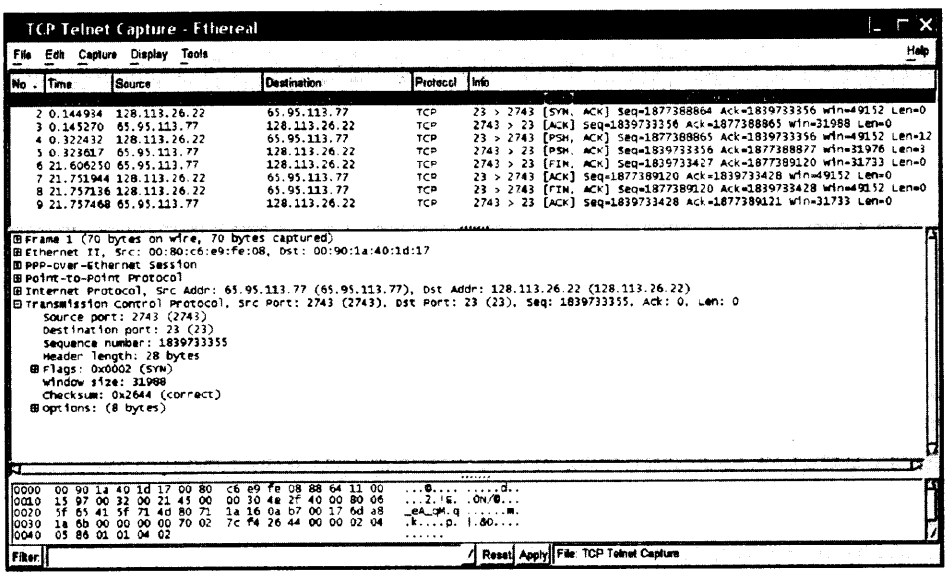


FIGURE 5.32 Example of TCP segment exchange: Three-way handshake for connection setup; data transfer; and graceful connection close.

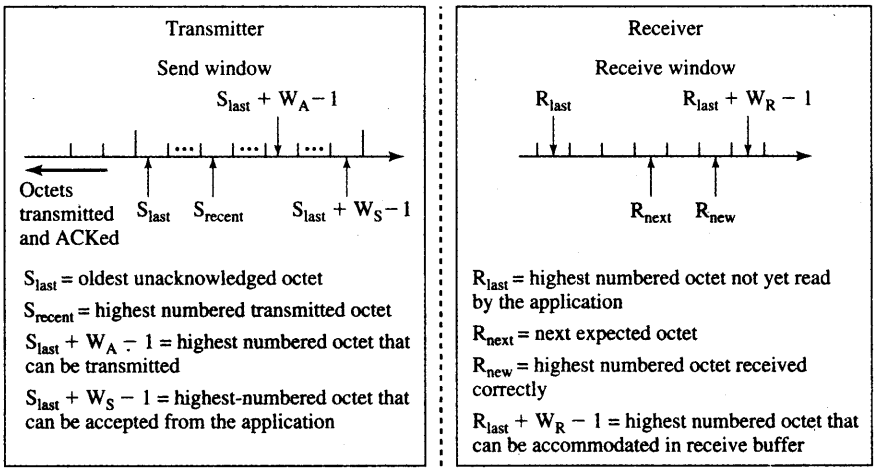


FIGURE 5.33 TCP end-to-end flow control.

segment and confirming the sequence numbers. The bidirectional connection is now set up and the applications can begin exchanging information.[10]

TCP uses a sliding window mechanism that implements a form of Selective Repeat ARQ using ACK messages and a timer mechanism, as shown in Figure 5.33. The send window contains three pointers:

- S_{last} points to the oldest byte that has not yet been acknowledged.
- S_{recent} points to the last byte that has been transmitted but not yet acknowledged.
- $S_{last} + W_S - 1$ indicates the highest numbered byte that the transmitter is willing to accept from its application.

Note that W_S no longer specifies the maximum allowable number of outstanding transmitted bytes. A different parameter, discussed below, specifies this value.

The receiver maintains a receive window which contains three pointers:

- R_{last} points to the oldest byte that has been read by the destination application.
- R_{next} points to the location of the highest numbered byte that has not yet been received correctly, that is, the next byte it expects to receive.
- R_{new} points to the location of the highest numbered byte that has been received correctly.

Note that R_{new} can be greater than R_{next} because the receiver will accept out-of-sequence, error-free segments. The receiver can buffer at most W_R bytes at any given time, so $R_{last} + W_R - 1$ is the maximum numbered byte that the receiver is prepared to accept. Note that we are assuming that the user application does not necessarily read the received bytes as soon as they are available.

The transmitter arranges a consecutive string of bytes into a segment. The segment contains a header with address information that enables the network to direct the segment to its destination application process. The segment also contains a *sequence*

[10]The rationale for the design of the three-way handshake is given in Chapter 8.

number that corresponds to the number of the first byte in the string that is being transmitted. Note that this differs significantly from conventional ARQ. The transmitter decides to transmit a segment when the number of bytes in the send buffer exceeds some specified threshold or when a timer that is set periodically expires. The sending application can also use a **push command** that forces the transmitter to send a segment.

When a segment arrives, the receiver performs an error check to detect transmission errors. If the segment is error free and is not a duplicate segment, then the bytes are inserted into the appropriate locations in the receive buffer if the bytes fall within the receive window. As indicated earlier, the receiver will accept out-of-order but error-free segments. If the received segment contains the byte corresponding to R_{next}, then the R_{next} pointer is moved forward to the location of the next byte that has not yet been received. An acknowledgment with the sequence number R_{next} is sent in a segment that is transmitted in the reverse direction. R_{next} acknowledges the correct receipt of all bytes up to $R_{next} - 1$. This acknowledgment, when received by the transmitter, enables the transmitter to update its parameter S_{last} to R_{next}, thus moving the send window forward.

As an example consider the sequence of TCP segments in Figure 5.32. Frame 4 in the top pane contains a segment from the Telnet server to the Telnet client. The first byte in the segment has sequence number 1877388865 and carries a payload of 12 bytes. Frame 5 contains a segment in the reverse direction which carries an ACK number of 1877388877 which acknowledges these 12 bytes.

TCP separates the flow control function from the acknowledgment function. The flow control function is implemented through an **advertised window** field in the segment header. Segments that travel in the reverse direction contain the advertised window size that informs the transmitter of the number of buffers currently available at the receiver. The advertised window size is given by

$$W_A = W_R - (R_{new} - R_{last}) \tag{5.15}$$

The transmitter is obliged to keep the number of outstanding bytes below the advertised window size, that is

$$S_{recent} - S_{last} \leq W_A \tag{5.16}$$

To see how the flow control takes effect, suppose that the application at the receiver's side stops reading the bytes from the buffer. Then R_{new} will increase while R_{last} remains fixed, thus leading to a smaller advertised window size. Eventually the receive window will be exhausted when $R_{new} - R_{last} = W_R$, and the advertised window size will be zero. This condition will cause the transmitter to stop sending. The transmitter can continue accepting bytes from its application until its buffer contains W_S bytes. At this point the transmitter blocks its application from inserting any more bytes into the buffer.

As an example consider the sequence of segments in Figure 5.32 once more. From frame 1 we can see that the initial window size for the client is 31988 bytes. The initial window size for the server is 49152 bytes. The variation in the client's window size can be seen from the sequence of segments in the figure.

Finally, we consider the retransmission procedure that is used in TCP. The transmitter sets a timer each time a segment is transmitted. If the timer expires before any of the bytes in the segment are acknowledged, then the segment is retransmitted. The Internet environment in which TCP must operate implies that the delay incurred by segments in traversing the network can vary widely from connection to connection and even during

the course of a single connection. For this reason TCP uses an adaptive technique for setting the retransmission time-out value. The **round-trip time (RTT)** t_{RTT} is estimated continuously, using measurements of the time τ_n that elapses from the instant a segment is transmitted until when the corresponding acknowledgment is received

$$t_{RTT}\ (new) = \alpha t_{RTT}\ (old) + (1 - \alpha)\tau_n \tag{5.17}$$

where α is a number between 0 and 1. A typical value of α is 7/8. The time-out value t_{out} is chosen to take into account not only t_{RTT} but also the variability in the estimates for the round-trip time. This variability is given by the standard deviation σ_{RTT} of the round-trip time. The time-out value is then

$$t_{out} = t_{RTT} + k\sigma_{RTT} \tag{5.18}$$

where k is some suitable constant. Thus if the estimates for round-trip times are highly variable, then the standard deviation will be large and a large time-out value will be used. On the other hand, if the round-trip times are nearly constant, the standard deviation will be close to zero, and the time-out value will be slightly larger than the mean round-trip time. The standard deviation involves estimating the average of the squared deviation, $(\tau_n - t_{RTT})^2$, and taking the square root. In practice, the average of the absolute deviation $|\tau_n - t_{RTT}|$ is simpler to estimate and is used instead:

$$d_{RTT}\ (new) = \beta d_{RTT}\ (old) + (1 - \beta)|\tau_n - t_{RTT}| \tag{5.19}$$

A typical value is $\beta = 3/4$. The time-out value that has been found to work well is then

$$t_{out} = t_{RTT} + 4d_{RTT} \tag{5.20}$$

PART II: Data Link Controls

The main purpose of the data link layer is to enable the transfer of frames of information over the digital bit or octet stream provided by the physical layer. To provide this service the data link layer may be called upon to:

1. Insert framing information into the transmitted stream to indicate the boundaries that define frames.
2. Provide error control to ensure reliable transmission.
3. Provide flow control to prevent the transmitter from overrunning the receiver buffers.
4. Insert address or protocol type information to enable the data link layer to carry information from multiple users.

In addition, the data link protocol may provide maintenance and security functions that are required to operate the data link. The main assumption here is that the data link is wirelike in the sense that frames arrive, if they arrive at all, in the same order that they were transmitted. Thus the data link controls are applicable to point-to-point communication lines as well as connections over a network where frames follow the same path.

We begin this part by discussing various approaches to perform the framing function. We then show how framing, error control, and flow control are incorporated into

standard data link controls. We will focus on the Point-to-Point Protocol (PPP) and the High-level Data Link Control (HDLC), which were developed by the IETF and ISO, respectively.

5.4 FRAMING

Framing involves identifying the beginning and end of a block of information within a digital stream. Framing presupposes that there is enough synchronization at the physical layer to at least identify an individual bit or byte. There is a hierarchy of degrees of bit synchronization accuracy and associated framing methods. At the coarsest level of accuracy we have *asynchronous data transmission*, which was discussed for the RS-232 standard for serial line interfaces in Appendix 3A. In this case, transmissions do not occur at regular intervals and the receiver resynchronizes at the start of each eight-bit character by the use of a start bit that precedes and a stop bit that ends each character as shown in Figure 3.79. In *synchronous data transmission* bits are transmitted at regular intervals and the receiver has circuitry (typically, a phase locked loop) that recovers and tracks the frequency and bit transitions of the received data. We saw in Chapter 3 that line coding is used to facilitate bit synchronization at the receiver by providing enough transitions in the transmitted bit stream regardless of the information contents.

The requirements on a framing method can vary in several ways. Framing may involve delineating the boundaries between frames that are of *fixed length* or it may involve delineating between frames that are of *variable length*. In some cases, the information contained inside a frame can be any number of bits, and in others, the information is constrained to be an integer number of characters of a certain length (e.g., octets or 32-bit words).

Consider the case of frames that are fixed in length. The first example we encountered of this case was in the physical layer for the T-1 carrier system in which a frame consists of a single framing bit followed by 24 octets as shown in Figure 4.4. The single framing bit follows the pattern 101010.... When the receiver needs to synchronize to a bit streams it is initially in a "hunt" state where it tests a given bit position to see if it corresponds to the framing bit. It is statistically very improbable that an arbitrary position will carry the sequence 1010... for a long time, so eventually the receiver will lock onto the correct framing bit position. Thereafter, only bit errors, loss of bits, or loss of signal will cause a loss of frame synchronization. A second example of fixed length framing in the physical layer is provided by SONET frames where the first two octets of each frame consist of the sequence 11110110 00101000. A third example of a fixed-length framing is provided by ATM cell delineation. An ATM cell is a fixed length packet that consists of 53 bytes. ATM framing can be viewed as a data link layer function and is based on the use of header CRC checking, which is discussed later in this section. The essence of framing in these three examples is character counting. After the framing bit or character is found, the end of a frame can be found by counting the specified number of characters. The purpose of the framing bit or character is merely to confirm that the frame position has not changed or been lost.

Variable-length frames need more information to delineate. The methods available include: special characters to identify beginning and end of frame, special bit patterns

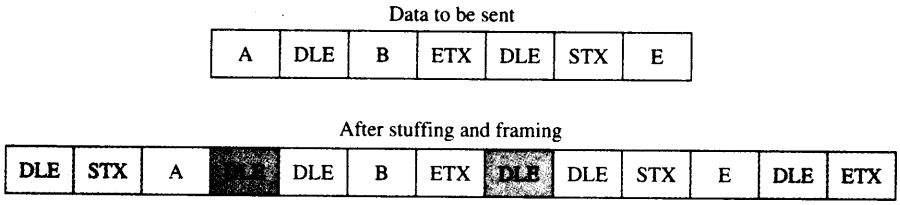


FIGURE 5.34 Example of byte stuffing using data link escape (DLE) characters.

"flags" to identify the beginning and end of frames, and character counts and CRC checking methods.

BEGINNING AND END OF FRAME CHARACTERS AND BYTE STUFFING

Character-based frame synchronization methods are used when the information in a frame consists of an integer number of characters. For example, asynchronous transmission systems are used extensively to transmit sequences of printable characters using eight-bit ASCII code. To delineate a frame of characters, special eight-bit codes that do not correspond to printable characters are used as control characters. In ASCII code all characters with hexadecimal values less than 20 correspond to nonprintable characters. In particular an STX (start of text) control character has HEX value 02 and indicates the beginning of a frame and an ETX (end of text) character has HEX value 03 and denotes the end of a frame. This method works if the frame contains only printable characters. If a frame carries computer data, then it is possible that an ETX character will appear inside the frame and cause the receiver to prematurely truncate the frame. We say that the method is not *transparent* because the frame cannot carry all possible bit sequences.

The use of **byte stuffing** enables transparent operation. Byte stuffing operates as follows. A special DLE (data link escape) control with HEX value 10 is introduced. The two-character sequence DLE STX is used to indicate the beginning of a frame and DLE ETX denotes the end of a frame. The receiver looks for these character pairs to identify the beginning and end of frames. In order to deal with the occurrence of DLE STX or DLE ETX in the data contained in the frame, an extra DLE is inserted or "stuffed" before the occurrence of a DLE inside the frame. Consequently every legitimate DLE in the data is replaced by two DLEs. The *only* incidence of an individual DLE occurs when DLE precedes the STX or the ETX that identify the beginning and end of frame. Figure 5.34 shows an example of byte stuffing.

FLAGS, BIT STUFFING, AND BYTE STUFFING

Flag-based frame synchronization was developed to transfer an *arbitrary number of bits* within a frame. Figure 5.35 shows the structure of an HDLC frame. The beginning

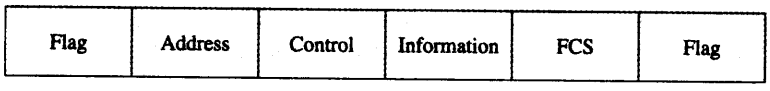


FIGURE 5.35 Frame structure of HDLC: The information field can contain an arbitrary number of bits.

(a) Data to be sent

0110111111111100

After stuffing and framing

*01111110*0110111110111110*0000*1111110*

(b) Data received

01111110000111011111011111011001111110

After destuffing and deframing

000111011111-11111-110

FIGURE 5.36 Example of (a) bit stuffing and (b) destuffing in HDLC.

and end of an HDLC frame is indicated by the presence of an eight-bit flag. The flag in HDLC consists of the byte 01111110, that is, HEX 7E. **Bit stuffing** prevents the occurrence of the flag inside the frame. The transmitter examines the contents of the frame and inserts an extra 0 after each instance of five consecutive 1s. The transmitter then attaches the flag at the beginning and end of the resulting bit-stuffed frame. The receiver looks for five consecutive 1s in the received sequence. Five 1s followed by a 0 indicate that the 0 is a stuffing bit, and so the bit is removed. Five consecutive 1s followed by 10 indicate a flag. Five 1s followed by 11 indicate an error.

Figure 5.36a gives an example of bit stuffing in HDLC. To send the sequence, the transmitter needs to stuff two bits as shown. Part (b) shows how a receiver recovers the original sequence by removing stuff bits and flags. The "-" in the above sequence indicates locations where a stuffing 0 has been removed and the "*" denotes locations where the flag has been removed.

The HDLC flag is used in PPP data link control to provide framing as shown in Figure 5.37.[11] In PPP the data in a frame is constrained to be an integer number of octets. As before the flag 0x7E is used to indicate the beginning and end of a frame. Byte stuffing is used to deal with the occurrence of the flag inside the frame. A Control Escape octet is defined as binary 01111101, 0x7D. Any occurrence of the flag or the Control Escape character inside the frame is replaced by a two-character sequence consisting of Control Escape followed by the original octet exclusive-ORed with 0x20, that is, 00100000. After the exclusive-OR operation the characters 0x7E and 0x7D are replaced by 0x7D 0x5E and 0x7D 0x5D, respectively.

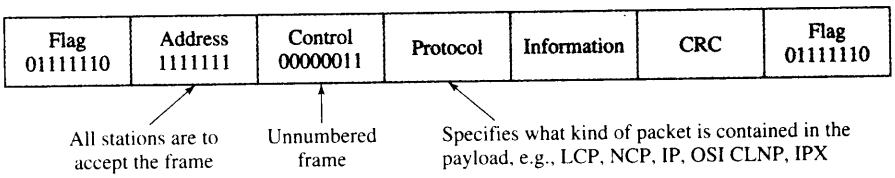


FIGURE 5.37 Frame structure of PPP frame: PPP uses the same flag as HDLC and inserts a protocol field that specifies the type of packet contained in the payload.

[11] PPP framing is defined in RFC 1662.

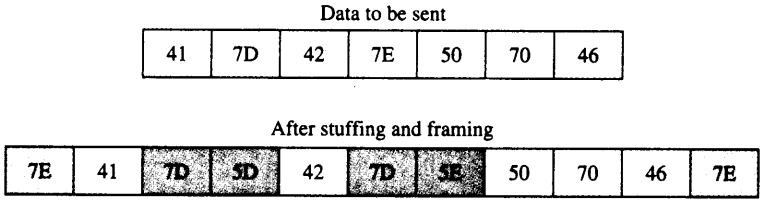


FIGURE 5.38 Example of byte stuffing in PPP.

The receiver must remove the inserted Control Escape characters prior to computing the CRC checksum. Each Control Escape octet is removed and the following octet is exclusive-ORed with 0x20, unless the octet is the Flag, which indicates the end of frame. Figure 5.38 gives an example of byte stuffing in PPP.

PPP framing can be used over asynchronous, bit-synchronous, or octet-synchronous transmission systems, as discussed in RFC 1549. PPP is used extensively in dialup modems. PPP also provides the framing in **Packet-over-SONET (POS)** to carry packet streams over high-speed SONET digital transmission systems.[12]

CRC-BASED FRAMING

Generic Framing Procedure (GFP) is a new standard for framing that is intended to address some shortcomings of PPP framing. A problem with byte stuffing as applied in PPP is that each appearance of 0x7E or 0x7D inside a frame adds an extra byte to the signal that needs to be transmitted. This creates a situation where the size of the transmitted frame that contains a given number of characters cannot be predicted ahead of time. Byte stuffing also provides an opportunity for malicious users to inflate the bandwidth consumed by inserting flag patterns within a frame.

GFP combines a frame length indication field with the Header Error Control (HEC) method used to delineate ATM cells. The use of a frame length indication field is simple. If we know the beginning of a frame, then we can find the length of the frame by looking at the length indication field. By counting the number of bytes indicated in the field, we then find the beginning of the next frame. The procedure can be continued indefinitely in the absence of errors. Unfortunately the occurrence of an error in the count field leads to a situation where the beginning of the next frame cannot be found. The use of HEC solves this problem.

Figure 5.39 shows the structure of the GFP frame. The first two fields, PLI and cHEC, are used to delineate a frame. The two-byte Payload Length Indicator (PLI) gives the size in bytes of the GFP payload area and so indicates the beginning of the next GFP frame. The two-byte cHEC field contains the CRC-16 redundancy check bits for the PLI and cHEC fields. The cHEC can be used to correct single errors and to detect multiple errors.

The GFP receiver synchronizes to the GFP frame boundary through a three-state process. The receiver is initially in the *hunt state* where it examines four bytes at a time

[12]Packet over SONET is defined in RFC 2615.

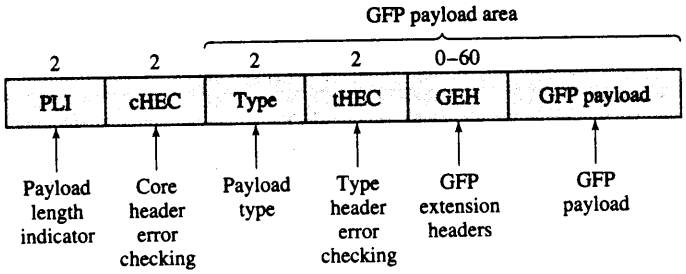


FIGURE 5.39 Frame structure of generic framing procedure.

to see if the CRC computed over the first two bytes equals the contents of the next two bytes. If there is no match, the receiver moves forward by one byte as GFP assumes octet synchronous transmission given by the physical layer. When the receiver finds a match it moves to the *pre-sync state*. While in this intermediate state the receiver uses the tentative PLI field to determine the location of the next frame boundary. If a target number N of successful frame detections has been achieved, then the receiver moves to the *sync state*. The sync state is the normal state where the receiver examines each PLI, validates it using the cHEC, extracts the payload, and proceeds to the next frame. In the sync state the single-error correcting capability of the CRC-16 is activated so that the occurrence of an isolated error does not disrupt frame synchronization.

The payload area in the GFP frame is scrambled using a $1 + x^{43}$ scrambler developed for ATM. The scrambling prevents malicious users from inserting sequences in the payload that consist of long strings of zeros and can cause SONET equipment to lose bit synchronization. This issue was discussed in the section on line coding in Chapter 3.

GFP is designed to operate over octet-synchronous physical layers. GFP carries payloads that are a multiple number of octets. The frames in GFP can be of variable length or of fixed length. In the frame-mapped mode, GFP can be used to carry variable-length payloads such as Ethernet frames, PPP/IP packets, or any HDLC-framed PDU. In the transparent-mapped mode, GFP carries synchronous information streams with low delay in fixed-length frames. Transparent mode GFP is suitable for handling traffic generated by computer storage devices using standards such as Fiber Channel, ESCON, FICON, and Gigabit Ethernet. GFP is an important emerging standard because it enables the most popular access standards (e.g., Ethernet, IP) to be carried over the most common transport equipment, namely, SONET/SDH.

5.5 POINT-TO-POINT PROTOCOL

The **Point-to-Point Protocol (PPP)** provides a method for encapsulating IP packets over point-to-point links. PPP can be used as a data link control to connect two routers or can be used to connect a personal computer to an Internet service provider (ISP) using a telephone line and a modem. The PPP protocol can operate over almost any

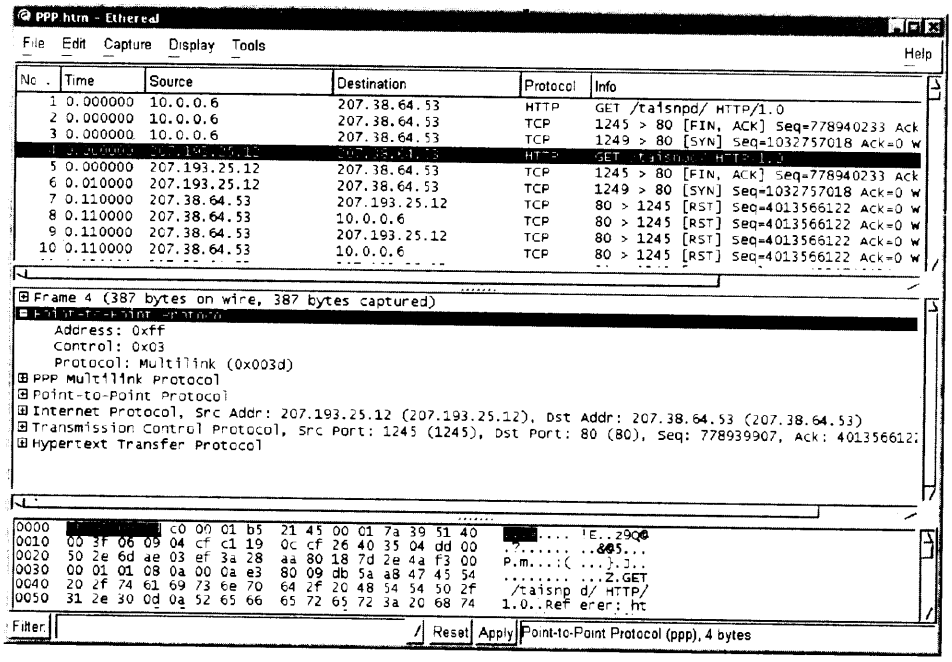


FIGURE 5.40 Example of a PPP frame.

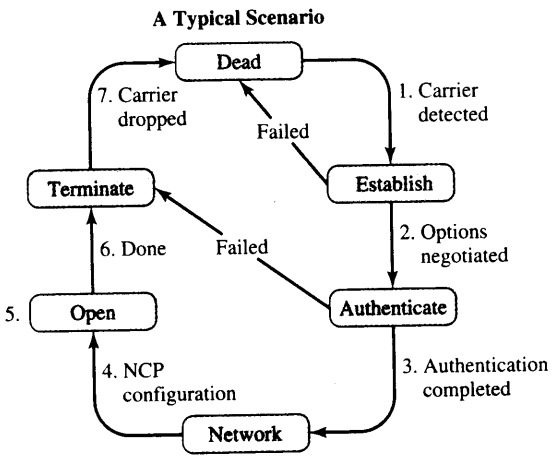
type of full-duplex point-to-point transmission link. It can also operate over traditional asynchronous links,[13] bit synchronous links, and new transmission systems such as ADSL and SONET.

The PPP protocol uses an HDLC-like frame format to encapsulate datagrams over point-to-point links, as shown in Figure 5.36. The PPP frame always begins and ends with the standard HDLC flag. Unlike HDLC, PPP frames consist of an integer number of bytes. For this reason, the bit stuffing technique of HDLC is not used. Instead the byte insertion method discussed in the previous section is used.

The second field in the PPP frame normally contains the "all 1s" address field (0xFF) that indicates that all stations are to accept the frame.[14] The control field is usually set to 00000011 (0x03) because PPP is normally run in connectionless mode. The 00000011 indicates an unnumbered HDLC frame, and so sequence numbers are not used. Figure 5.40 shows an example of a PPP frame. The middle pane shows the address and control fields of packet 4, which can be seen to conform to the preceding discussion. The protocol field indicates that the given frame carries a multilink PPP protocol PDU in its payload, which ultimately carries an HTTP message.

PPP was designed to support multiple network protocols simultaneously; that is, PPP can transfer packets that are produced by different network layer protocols. This

[13] In asynchronous links the transmission of each character is preceded by a "start" bit and followed by a "stop" bit. Synchronous links provide long-term bit synchronization, making start and stop bits unnecessary.
[14] HDLC is discussed in the next section.



Home PC to Internet Service Provider

1. PC calls router via modem.
2. PC and router exchange LCP packets to negotiate PPP parameters.
3. Check on identities.
4. NCP packets exchanged to configure the network layer, for example, TCP/IP (requires IP address assignment).
5. Data transport, for example, send/receive IP packets.
6. NCP used to tear down the network layer connection (free up IP address); LCP used to shut down data link connection.
7. Modem hangs up.

FIGURE 5.41 PPP phase diagram.

situation arises in *multiprotocol* routers that can simultaneously support several network layer protocols. The protocol field is 1 or 2 bytes long and is used to identify the network layer protocol of the packet contained in the information field. Note that the protocol and PPP information field together correspond to the information field in the normal HDLC frame (see Figure 5.35).[15] Finally, the CRC field can use the CCITT 16 or CCITT 32 generator polynomials presented in Chapter 3.

The PPP protocol provides many useful capabilities through a link control protocol and a family of network control protocols. The **Link Control Protocol (LCP)** is used to set up, configure, test, maintain, and terminate a link connection. As shown in Figure 5.41, the LCP begins by establishing the physical connection. The LCP involves an exchange of messages between peers to negotiate the link configuration. During the negotiation, a PPP endpoint may also indicate to its peer that it wants to multilink (i.e., combine multiple physical links into one logical link). Multilink PPP allows a high-speed data link to be built from multiple low-speed physical links. Once the peers have agreed on a configuration, an authentication process, if selected in the configuration, is initiated. We discuss the authentication option below.

After authentication has been completed, a **Network Control Protocol (NCP)** is used to configure each network layer protocol that is to operate over the link. PPP can subsequently transfer packets from these different network layer protocols (such as IP, IPX, Decnet, AppleTalk) over the same data link. The destination peer can then direct the encapsulated packet to the appropriate network layer protocol by reading the protocol field in the frame. The reason this capability is important is that routers have evolved to simultaneously support packets from different network protocols.

When a PC is connecting to an IP network, as in Figure 5.41, the NCP for IP negotiates a dynamically assigned IP address for the PC. In low-speed lines it may also

[15]Figure 5.46 shows the bits in the PPP control field appear in reverse order relative to the HDLC control field.

negotiate TCP and IP header compression schemes that reduce the number of bits that need to be transmitted.

A particular strength of PPP is that it includes authentication protocols, which is a major issue when the computer connects to a remote network. After the LCP has set up the link, these protocols can be used to authenticate the user. The **Password Authentication Protocol (PAP)** requires the initiator to send an ID and a password. The peer process then responds with a message indicating that the authentication has been successful or has failed. Depending on the type of situation, a system may allow a few retries. When PAP decides that a request has failed, it instructs the LCP to terminate the link. PAP is susceptible to eavesdropping because the ID and password are sent in plain text; PAP is therefore vulnerable to many different types of security attacks.

The **Challenge-Handshake Authentication Protocol (CHAP)** provides greater security by having the initiator and the responder go through a challenge-response sequence. CHAP assumes that the peer processes have somehow established a shared secret key. After LCP has established the link, the authenticator sends a challenge to its peer. The challenge consists of a random number and an ID. The peer process responds with a cryptographic checksum of the challenge value that makes use of the shared secret. The authenticator verifies the cryptographic checksum by using the shared secret key. If the two checksums agree, then the authenticator sends an authentication message. The CHAP protocol allows an authenticator to reissue periodically the challenge to reauthenticate the process. Security protocols are discussed in detail in Chapter 11.

EXAMPLE **PPP Connection Setup and Release**

Figure 5.42 shows the frames exchanged during the setup of a PPP connection on a dialup modem to an Internet Service Provider. The first nine frames are associated with the Link Control Protocol. Each end of the link sends LCP Configuration Request frames proposing a set of configuration options. The other end responds with an LCP Configuration-Ack if the options are acceptable, Configuration-Nak if all the options are recognizable but not acceptable, or Configuration-Reject if some of the options are not recognizable or not acceptable for negotiation. It can be seen from the figure that the options negotiation may require a number of frame exchanges. The content of the final LCP options negotiation frame is shown in the middle pane of the figure, which confirms that Password Authentication Protocol (PAP) is to be used for authentication.

Frames 10 and 11 carry out the PAP exchange. Recall that PAP sends the password in plain text, and if one were to zoom into the contents of frame 10 one can see the password. The final phase of the connection setup involves the Network Control Protocol. The example shows the use of the Internet Protocol Control Protocol (IPCP) to negotiate IP addresses and the use of IP datagram compression. Frame 21 is the final IPCP configuration acknowledgment. The PPP connection is now ready for data transfer. The PPP connection release involves the exchange of LCP Termination Request frame and LCP Termination Ack frames. These are not shown in the figure.

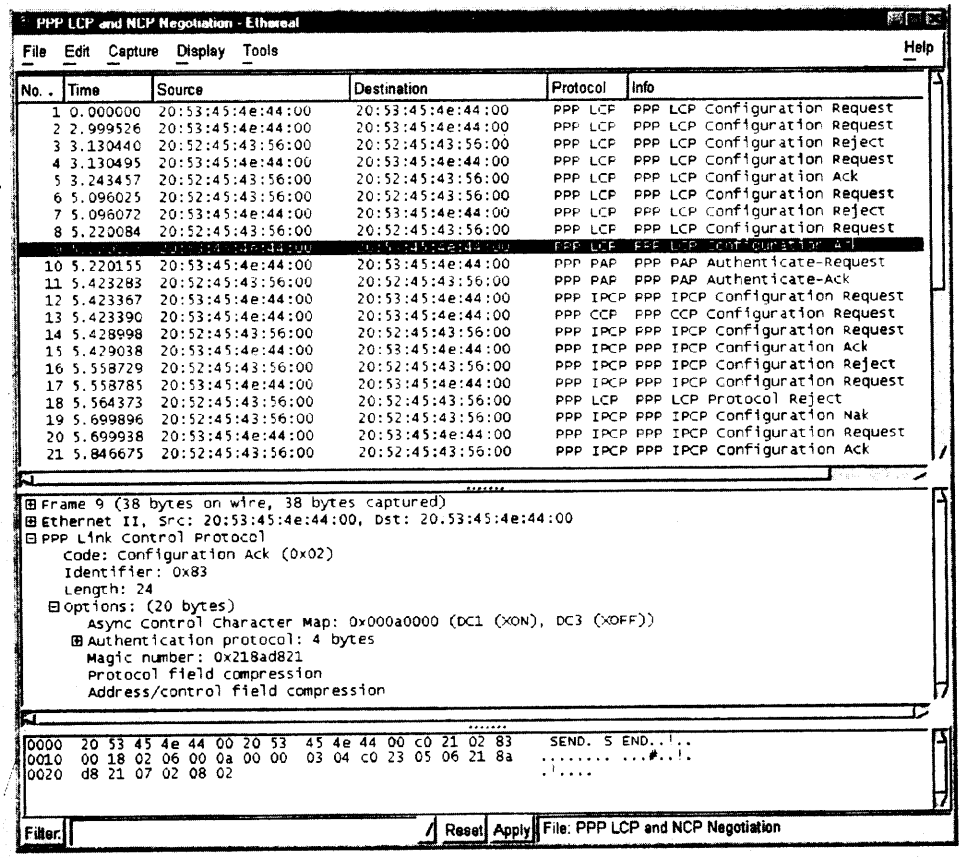


FIGURE 5.42 PPP packet capture during LCP and NCP negotiations.

5.6 HDLC DATA LINK CONTROL

High-level Data Link Control (HDLC) provides a rich set of standards for operating a data link over bit synchronous physical layers. HDLC is derived from the Synchronous Data Link Control (SDLC) developed by IBM. It is also related to the Link Access Procedure standards (LAPB and others) developed by CCITT-ITU.

5.6.1 Data Link Services

In Figure 5.43 we show the **data link control** as a set of functions whose role is to provide a communication service to the network layer. The network layer entity is

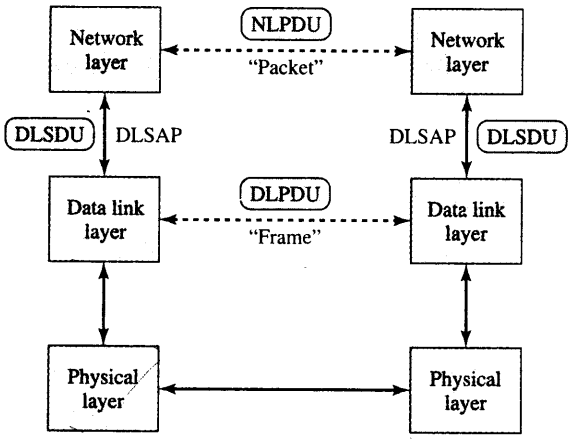


FIGURE 5.43 The data link layer.

involved in an exchange of packets with a peer network layer entity located at a neighbor packet-switching node. To exchange these packets the network layer must rely on the service that is provided by the data link layer. The **data link layer** itself transmits frames and makes use of the bit transport service that is provided by the physical layer, that is, the actual digital transmission system.

The network layer passes its packets (network layer PDU or NLPDU) to the data link layer in the form of a data link SDU (DLSDU). The data link layer adds a header and CRC check bits to the SDU (packet) to form a data link PDU (DLPDU) frame. The frame is transmitted using the physical layer. The data link layer at the other end recovers the frame from the physical layer, performs the error checking, and when appropriate delivers the SDU (packet) to its network layer.

Data link layers can be configured to provide several types of services to the network layer. For example, a *connection-oriented service* can provide error-free, ordered delivery of packets. Connection-oriented services involve three phases. The first phase involves setting up the connection. Each network layer has a **service access point (SAP)** through which it accesses its data link layer. These SAPs are identified by addresses, and so the connection setup involves setting up variables and allocating buffers in the data link layers so that packets can flow from one SAP to the other. The second phase of a connection-oriented service involves the actual transfer of packets encapsulated in data link frames. The third phase releases the connection and frees up the variables and buffers that have been allocated to the connection.

Data link layers can also be configured to provide *connectionless service*. In this case there is no connection setup, and the network layer is allowed to pass a packet across its local SAP together with the address of the destination SAP to which the packet is being sent. The data link layer transmits a frame that results in the delivery of a packet to the destination network layer. The connectionless service can be an *acknowledged* service in which case the destination network layer must return an acknowledgment that is delivered to the sending network layer. The connectionless service can also be *unacknowledged* in which case no acknowledgment is issued to the sending network layer. We show later in Chapter 6 that the local area networks generally provide unacknowledged connectionless service.

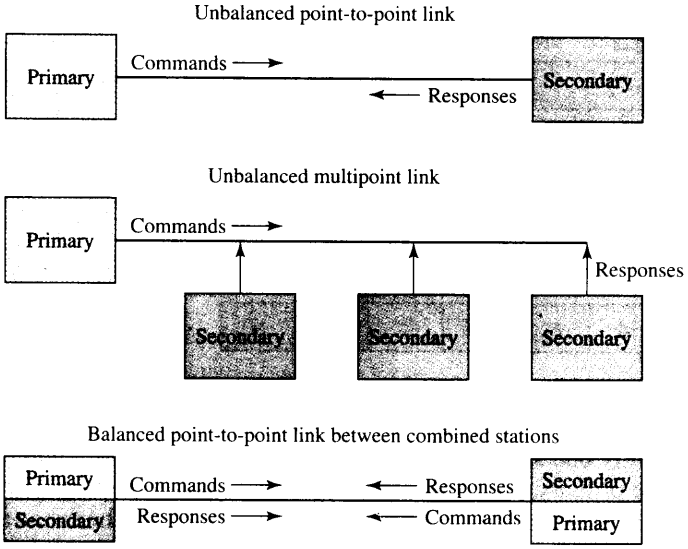


FIGURE 5.44 HDLC configurations.

5.6.2 HDLC Configurations and Transfer Modes

HDLC provides for a variety of data transfer modes that can be used in a number of different configurations. The **normal response mode (NRM)** of HDLC defines the set of procedures that are to be used with the *unbalanced* configurations shown in Figure 5.44. This mode uses a command/response interaction whereby the primary station sends command frames to the secondary stations and interrogates or polls the secondaries to provide them with transmission opportunities. The secondaries reply using response frames. In the *balanced* point-to-point link configuration, two stations implement the data link control, acting as peers. This configuration is currently in wide use. HDLC has defined the **asynchronous balanced mode (ABM)** for data transfer for this configuration. In this mode information frames can be transmitted in full-duplex manner, that is, simultaneously in both directions.

5.6.3 HDLC Frame Format

We saw in the discussion of ARQ that the functionality of a protocol depends on the control fields that are defined in the header. The format of the HDLC frame is defined so that it can accommodate the various data transfer modes. Figure 5.45 shows the

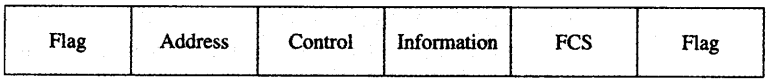


FIGURE 5.45 HDLC frame format.

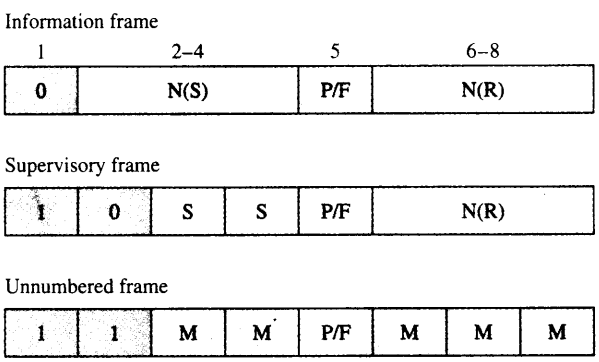


FIGURE 5.46 Control field format.

format of an HDLC frame. Each frame is delineated by two 8-bit flags. The frame has a field for only one address. Recall that in the unbalanced configuration, there is always only one primary, but there can be more than one secondary. For this reason, the address field always contains the address of the secondary. The frame also contains an 8- or 16-bit control field. We discuss the various types of controls fields below. The information field contains the user information, that is, the SDU. Finally, a 16- or 32-bit CRC calculated over the control, address, and information fields is used to provide error-detection capability. The ITU-CRC polynomials discussed in Chapter 3 are used with HDLC.

There are three types of control fields. Figure 5.46 shows the general format of the control field. A 0 in the first bit of the control field identifies an **information frame (I-frame)**. A 10 in the first two bits of the control field identifies a **supervisory frame**. Finally, a 11 in the first two bits of the control field identifies an **unnumbered frame**. The information frame and supervisory frames implement the main functions of the data link control, which is to provide error and flow control.

Each control field contains a Poll/Final bit indicated by P/F in Figure 5.46. In unbalanced mode this bit indicates a poll when being sent from a primary to a secondary. The bit indicates a final frame when being sent from a secondary to a primary. Thus to poll a given secondary, a host sends a frame to the secondary, indicated by the address field with the P/F bit set to 1. The secondary responds to such a frame by transmitting the frames it has available for transmission. Only the last frame transmitted from the secondary has the P/F bit set to 1 to indicate that it is the final frame.

The N(S) field in the I-frame provides the send sequence number of the I-frame. The N(R) field is used to piggyback acknowledgments and to indicate the next frame that is expected at the given station. N(R) acknowledges the correct receipt of all frames up to and including N(R) − 1.

There are four types of supervisory frames, corresponding to the four possible values of the S bits in the control field. A value of SS = 00 indicates a **receive ready (RR)** frame. RR frames are used to acknowledge frames when no I-frames are available to piggyback the acknowledgment. A value of SS = 01 corresponds to a **reject (REJ)** frame. REJ frames are used by the receiver to send a negative acknowledgment. As discussed in the section on ARQ, a REJ frame indicates that an error has been detected and that the transmitter should go back and retransmit frames from N(R) onwards (for

Go-Back-N). A value of SS = 10 indicates a **receive not ready (RNR)** frame. The RNR frame acknowledges all frames up to N(R) − 1 and informs the transmitter that the receiver has temporary problems, that is, no buffers, and will not accept any more frames. Thus RNR can be used for flow control. Finally, SS = 11 indicates a **selective reject (SREJ)** frame. SREJ indicates to the transmitter that it should retransmit the frame indicated in the N(R) subfield (for Selective Repeat ARQ). Note that this frame is defined only in HDLC, but not necessarily in other variations of HDLC.

The combination of the I-frames and supervisory frames allow HDLC to implement Stop-and-Wait, Go-Back-N, and Selective Repeat ARQ. The discussions earlier in the chapter regarding the operation of these ARQ protocols apply here. In particular, we note that HDLC has two options for sequence numbering. In the default case HDLC uses a three-bit sequence numbering. This scheme implies that the maximum send window size is $2^3 - 1 = 7$ for Stop-and-Wait and Go-Back-N ARQ. In the extended sequence numbering option, the control field is increased to 16 bits, and the sequence numbers are increased to 7 bits. The maximum send window size is $2^7 - 1 = 127$ for Stop-and-Wait and Go-Back-N ARQ. For Selective Repeat ARQ the maximum send and receive window sizes are 4 and 64, respectively.

The unnumbered frames implement a number of control functions. Each type of unnumbered frame is identified by a specific set of M bits. During call setup or release, specific unnumbered frames are used to set the data transfer mode. For example, the **set asynchronous balanced mode (SABM)** frame indicates that the sender wishes to set up an asynchronous balanced mode connection; similarly, a **set normal response mode (SNRM)** frame indicates a desire to set up a normal response mode connection. Unnumbered frames are also defined to set up connections with extended, that is, seven-bit sequence numbering. For example, a set asynchronous balanced mode extended (SABME) frame indicates a request to set up an asynchronous balanced mode connection with seven-bit sequence numbering. The **disconnect (DISC)** frame indicates that a station wishes to terminate a connection. An **unnumbered acknowledgment (UA)** frame acknowledges frames during call setup and call release. The **frame reject (FRMR)** unnumbered frame reports receipt of an unacceptable frame. Such a frame passes a CRC check but is not acceptable. For instance, it could have invalid values of the S bits in the case of supervisory frames, for example, 11 when SREJ is not defined, or invalid values of the M bits for an unnumbered frame. The FRMR frame contains a field where additional information about the error condition can be provided. Finally, we note that additional unnumbered frame types are defined for such tasks as initialization, status reporting, and resetting.

5.6.4 Typical Frame Exchanges

We now consider a number of simple examples to show how the frames that are defined for HDLC can be used to carry out various data link control procedures. We use the following convention to specify the frame types and contents in the following figures. The first entry indicates the contents of the address field; the second entry specifies the type of frame, that is, I for information, RR for receive ready, and so on; the third entry

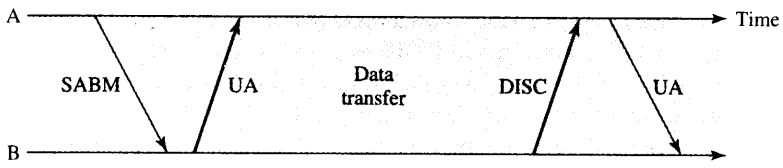


FIGURE 5.47 Exchange of frames for connection establishment and release.

is N(S) in the case of I-frames only, the send sequence number; and the following entry is N(R), the receive sequence number. A P or F at the end of an entry indicates that the Poll or Final bit is set.

Figure 5.47 shows the exchange of frames that occurs for connection establishment and release. Station A sends an SABM frame to indicate that it wishes to set up an ABM mode connection. Station B sends an unnumbered acknowledgment to indicate its readiness to proceed with the connection. A bidirectional flow of information and supervisory frames then takes place. When a station wishes to disconnect, it sends a DISC frame and the other station sends UA.

Figure 5.48 shows a typical exchange of frames using normal response mode. In this example the primary station A is communicating with the secondary stations B and C. Note that all frames contain the address of the secondary station. Station A begins by sending a frame to station B with the poll bit set and the sequence numbering N(R) = 0, indicating that the next frame it is expecting from B is frame 0. Station B receives the polling frame and proceeds to transmit three information frames with N(S) = 0, 1, 2. The last frame has the final bit set. Frame 1 from station B incurs transmission errors,

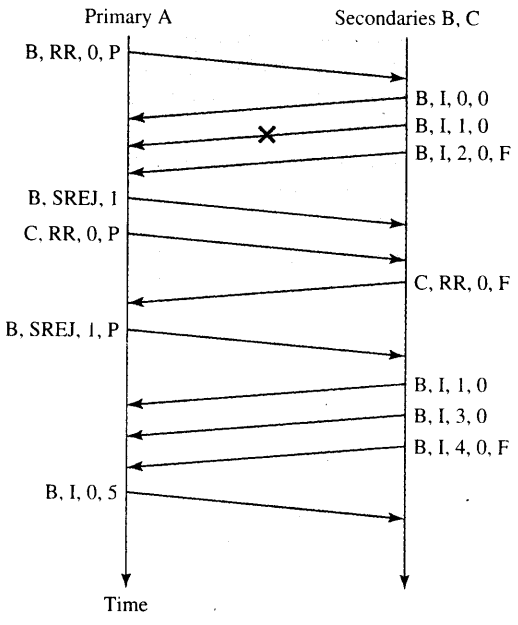


FIGURE 5.48 Exchange of frames using normal response mode.

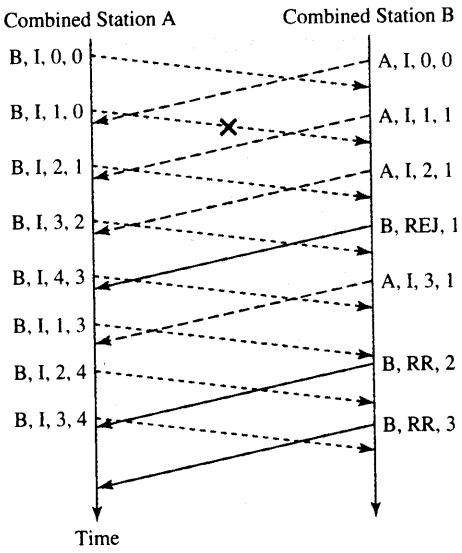


FIGURE 5.49 Exchange of frames using asynchronous balanced mode.

and subsequently station A receives an out-of-sequence frame with N(S) = 2. Station A now sends an SREJ frame with N(R) = 1 but without setting the poll bit. This frame indicates to station B that it should be prepared to retransmit frame 1. Station A proceeds to poll station C, which replies that it has no I-frames to transmit. Station A now sends an SREJ to B with N(R) = 1, and the poll bit set. Station B responds by resending frame 1 and then skipping to frames 3 and 4. In the last frame in the figure, station A sends to station B an information frame with N(S) = 0 and with N(R) = 5, acknowledging receipt of all frames from B up to 4.

Finally, in Figure 5.49 we consider the case of bidirectional flow of information using ABM. The following convention is used for the addressing: The address field always contains the address of the secondary station. Thus if a frame is a command, then the address field contains the address of the receiving station. If a frame is a response, then the address field contains the address of the sending station. Information frames are always commands. RR and RNR frames can be either command or response frame; REJ frames are always response frames.

In the example, station A begins by sending frames 0 and 1 in succession. Station B begins slightly later and transmits frame 0. Shortly thereafter station B receives frame 0 from A. When station B transmits its I-frame with $N_B(S) = 1$, an acknowledgment is piggybacked by setting $N_B(R) = 1$. In the meantime, station A has received frame 0 from station B so when it transmits its frame 2, it piggybacks an acknowledgment by using $N_A(R) = 1$. Now frame 1 from station A has undergone transmission errors, and so when station B receives a frame with $N_A(S) = 2$, it finds that the frame is out of sequence. Station B then sends a negative acknowledgment by transmitting an REJ frame with $N_B(R) = 1$. Meanwhile station A is happily proceeding with the transmission of frames 2, 3, and 4, which carry piggybacked acknowledgments. After receiving the REJ frame, station A goes back and begins retransmitting from frame 1 onward. Note that the value of N(R) is unaffected by retransmission. In the figure we use

solid lines to indicate response frames that are sent from B to A. Thus we see that when station B does not have additional I-frames for transmission, it sends acknowledgments by using RR frames in response form.

◆ 5.7 LINK SHARING USING PACKET MULTIPLEXERS

In Chapter 1 we discussed how applications in early terminal-oriented networks were found to generate data in a bursty fashion: Message transmissions would be separated by long idle times. Assigning a dedicated line to transmission from a single terminal resulted in highly inefficient use of the line. Such inefficiency became a serious issue when the transmission lines involved were expensive, as in the case of long-distance lines. Statistical multiplexers were developed to concentrate data traffic from multiple terminals onto a shared communication line, leading to improved efficiency. Framing, addressing, and link control procedures were developed to structure the communications in the shared transmission link, eventually leading to the development of data link control protocols such as HDLC and PPP.

Current networks support a very broad array of applications many of which generate traffic in highly bursty fashion, making statistical multiplexing an essential component in the operation of packet networks. Indeed in Figure 5.7 we see that in modern networks each packet switch takes packets that it receives from users and from other packet switches and multiplexes them onto shared data links. In this section we present an introduction to classical modeling techniques for analyzing the performance of packet multiplexers.[16] We begin by looking at the problem of multiplexing the packet flows from various data sources. We find that the approach leads to significant improvements in the utilization of the transmission line. We also find that the aggregation of traffic flows results in improved performance. We then consider the multiplexing of packetized voice traffic and again find benefits in the aggregation of multiple flows.

5.7.1 Statistical Multiplexing

Computer applications tend to generate data for transmission in a bursty manner. Bursts of information are separated by long idle periods, and so dedicating a transmission line to each computer is inefficient. This behavior led to the development of statistical multiplexing techniques for sharing a digital transmission line. The information generated by a computer is formatted into packets that contain headers that identify the source and destination, as shown in Figure 5.50. The packets are then transmitted in a shared communications line. In general the packets are variable in length.

[16]The characterization of the traffic generated by current applications is not well understood and is currently an active area of research. The reader is referred to recent issues of the *IEEE Journal on Selected Areas in Communications.*

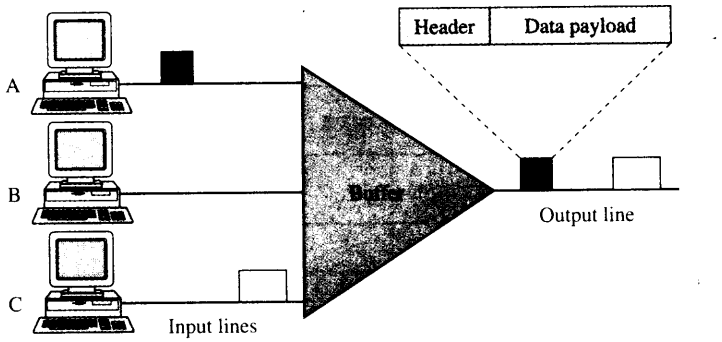


FIGURE 5.50 Statistical multiplexing of data.

To see the benefit of multiplexing, consider Figure 5.51 where the packets from three terminals are to be multiplexed. Part (a) shows the times when the packets would have been transmitted if each terminal had its own line at speed R bps. Part (b) shows the time when the packets are transmitted if the packets from the three flows are combined by a multiplexer. Because the terminals generate packets in bursty fashion, it is possible to combine the packet streams into a single line of speed R bps. Note that because the packet generation times overlap, the multiplexer must buffer and delay some of the packets. Nevertheless, all the packets get transmitted in the order in which they were generated. Thus by aggregating the packet flows into a single transmission line, the multiplexer reduces the system cost by reducing the number of lines.

In general, packet multiplexers are used in two situations. In the first situation packets arrive from multiple lines to a statistical multiplexer for transmission to a remote site. In the second situation packets arrive to a packet switch that then routes some of the packets for transmission over a specific data link. In effect, these packets are statistically multiplexed onto the given link. In both situations the multiplexer buffers the packets and arranges them in a queue. As the transmission line becomes available, packets are then transmitted according to their position in the queue. Typically packets are transmitted in first-in, first-out (FIFO) fashion, but increasingly multiplexers use priorities and scheduling of various types to determine the order of packet transmission.

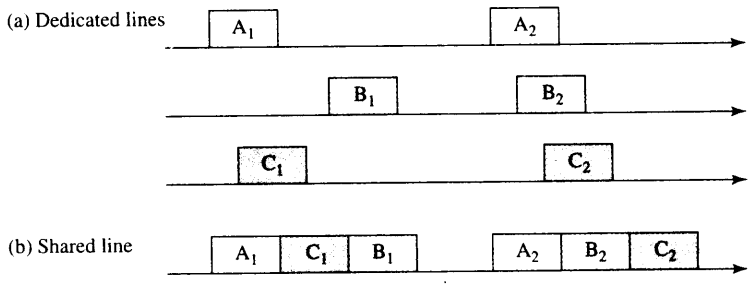


FIGURE 5.51 Lines with and without multiplexing.

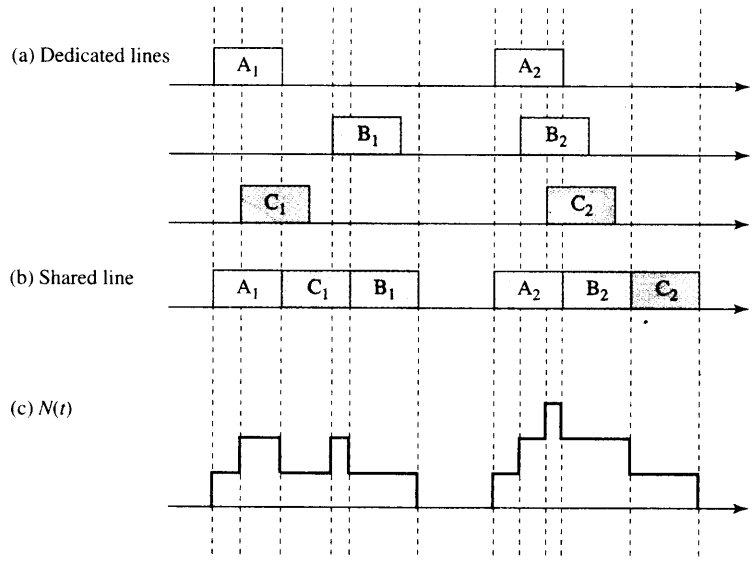


FIGURE 5.52 The number of packets in the statistical multiplexer $N(t)$.

In general, most computer data applications do not tolerate loss, but they can tolerate some delay. Consequently, the multiplexers are operated so that they trade off packet delay versus utilization of the transmission line. However, the amount of buffering available is limited, so packet losses can occur from time to time when a packet arrives to a full system. End-to-end protocols are responsible for recovering from such losses.

The behavior of a statistical multiplexer is characterized by $N(t)$, the number of packets in the statistical multiplexer at time t. Figure 5.52 shows $N(t)$ for the example in Figure 5.51. The variations in $N(t)$ are determined by the arrival times and the departure times of the packets. Suppose that the average length of a packet is $E[L]$ bits and that the transmission line has a speed of R bits/second. The average packet transmission time is then $E[L]/R$ seconds. Over the long run the transmission line can handle at most $\mu = R/E[L]$ packets/second, so μ is the **maximum departure rate** at which packets can be transmitted out of the system. For example, suppose that the average packet size is 1000 bytes and that the transmission line speed is 64,000 bps. The maximum packet transmission rate is then $\mu = 64{,}000$ bps/(1000 bytes $\times$ 8 bits per byte) = 8 packets/second.

Let λ be the average packet **arrival rate** to a multiplexer in packets/second. If λ is higher than μ, then the buffer will build up on the average and many packet losses will occur. However, when λ is less than μ, the number of packets in the multiplexer will fluctuate because packet arrivals can bunch up or build up during the transmission of particularly long packets. Consequently, packets can still overflow from time to time. We can reduce the incidence of this type of packet loss by increasing the number of the buffers. We define the **load** ρ to be given by $\rho = \lambda/\mu$. Clearly, we want the arrival rate λ to be less than the departure rate μ, and hence $\rho < 1$.

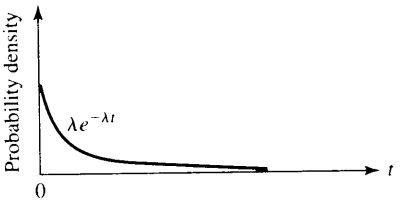


FIGURE 5.53 Exponential density function for interarrival times; the likelihood of an interarrival time near t is given by the value of the density function.

As an example we present the results for a statistical multiplexing system that is modeled by the so-called M/M/1/K queueing model.[17] In this model packets arrive at a rate of λ packets/second, and the times between packet arrivals are random, are statistically independent of each other, and have an exponential density function with mean $1/\lambda$ seconds as shown in Figure 5.53. This arrival process is usually called the **Poisson arrival process**.

Note from the figure that short interarrival times are more likely than are long interarrival times. The model also assumes that the packet transmission times are random and have an exponential density with mean $1/\mu = E[L]/R$. The model assumes that there is enough buffering to hold up to K packets in the statistical multiplexer. The packet delay T is defined as the total time a packet spends in the multiplexer and is given by the sum of the time spent waiting in queue and the packet transmission time. We show in Appendix A that the **packet loss probability** is given by

$$P_{loss} = \frac{(1-\rho)\rho^K}{1-\rho^{K+1}} \tag{5.21}$$

and the average packet delay is given by

$$E[T] = \frac{E[N]}{\lambda(1-P_{loss})} \tag{5.22}$$

where $E[N]$ is the average number of packets in the multiplexer

$$E[N] = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}} \tag{5.23}$$

Figure 5.54 and Figure 5.55 show the average delay and the packet loss probability as a function of the load ρ. Note that in Figure 5.54 the average delay $E[T]$ is normalized to multiples of the average packet transmission time $E[L]/R$. Figure 5.54 assumes a value of $K = 10$ and shows both the desired case where $\rho < 1$ and overload case where $\rho > 1$. We discuss these two cases below.

When the arrival rate is very small, that is, ρ is approximately zero, an arriving packet is unlikely to find any packets ahead of it in queue, and so the delay in the statistical multiplexer is simply one packet transmission time. Note that the packet

[17]The M/M/1/K queueing system is analyzed in Appendix A. The first M refers to the exponentially distributed interarrival times, the second M refers to the exponentially distributed transmission times, the 1 refers to the fact that there is a single server, that is, transmission line, and the K refers to the maximum number of packets allowed in the system. For this discussion we only need the formulas that result from the analysis.

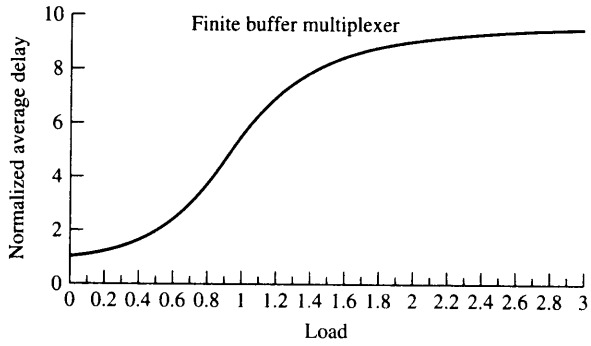


FIGURE 5.54 Average packet delay as a function of load in M/M/1/10 system; delay is normalized to multiples of a packet transmission time.

loss probability is very low at small loads. As the load approaches about 0.7, the average delay and the packet loss probability begin increasing. Note that if $\rho < 1$, then ρ^K decreases with K, so the loss probability P_{loss} can be made arbitrarily small by increasing K.

Now suppose that we have the overload condition $\rho > 1$. The packet loss probability increases steadily as the load increases beyond 1 because the system is usually full, and so the excess packet arrivals are lost. However, note from Figure 5.54 that the average delay approaches an asymptotic value of 10. Because the system is almost always full, most of the packets that actually enter the system experience close to the maximum delay of 10 transmission times.

The behavior of the average delay and the loss probability as a function of load shown in the two figures is typical of a statistical multiplexing system. The exact behavior will depend on the distribution of the packet interarrivals, on the distribution of the packet transmission times, and on the size of the buffer. For example, consider systems that are "more random" that the M/M/1/K system, that is, because arrivals are more bursty or because the packet length distribution is such that long packets are more probable. For such systems the average delay and the loss probabilities will increase more quickly as a function of load ρ than in the figures. On the other hand, systems that are "less random" will increase less quickly as a function of load.

Now suppose that the buffer size K is made arbitrarily large; that is, $K \to \infty$ when $\rho < 1$. The result is the so-called M/M/1 model. The loss probability P_{loss} goes to zero,

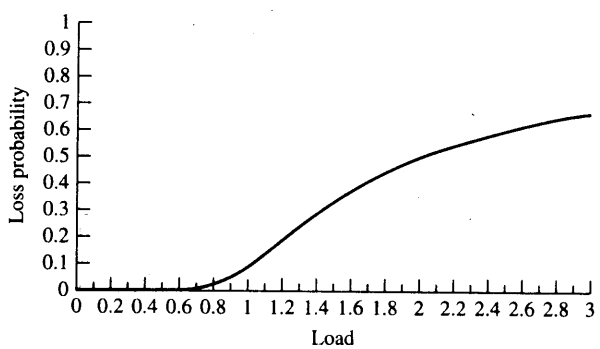


FIGURE 5.55 Packet loss probability as a function of load for M/M/1/10.

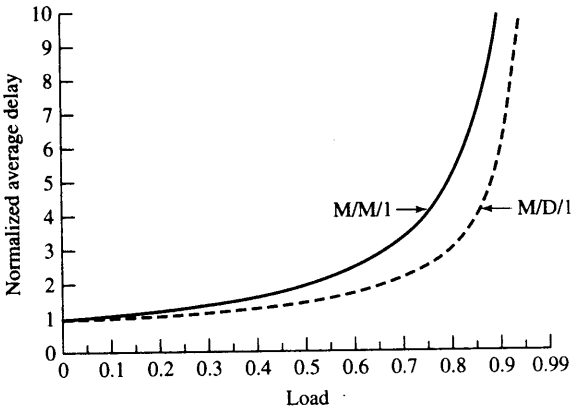


FIGURE 5.56 Average delay for systems with infinite buffers.

and the average delay becomes

$$E[T_M] = \frac{1}{\lambda}\left[\frac{\rho}{1-\rho}\right] = \left[\frac{1}{1-\rho}\right]\frac{1}{\mu} = \left[\frac{\rho}{1-\rho}\right]\frac{1}{\mu} + \frac{1}{\mu} \text{ for M/M/1 model} \quad (5.24)$$

Figure 5.56 shows $E[T]$ for the M/M/1 system. The average packet transmission time is $1/\mu$, so the average time spent waiting in queue prior to transmission is

$$E[W_M] = E[T_M] - \frac{1}{\mu} = \left[\frac{\rho}{1-\rho}\right]\frac{1}{\mu} \text{ for M/M/1 model} \quad (5.25)$$

Figure 5.56 also shows the average delay $E[T_D]$ for the M/D/1 system that has exponential interarrivals, *constant* service times L/R corresponding to fixed-length packets, and infinite buffer size. For the M/D/1 system, we have

$$E[T_D] = \left[1 + \frac{\rho}{2(1-\rho)}\right]\frac{1}{\mu} = \left[\frac{\rho}{2(1-\rho)}\right]\frac{1}{\mu} + \frac{1}{\mu} \text{ for M/D/1 system} \quad (5.26)$$

so the average time spent waiting in queue is

$$E[W_D] = \left[\frac{\rho}{2(1-\rho)}\right]\frac{1}{\mu} \text{ for M/D/1 system} \quad (5.27)$$

Note that the M/M/1 and M/D/1 systems have delays that become arbitrarily large as the load approaches 1, that is, as the arrival rate approaches the maximum packet transmission rate. However, the system with constant service times has half the average waiting time of the system with exponential service times. As expected, the packet delay increases because the system becomes more random, that is, more variable in terms of packet transmission times.

EXAMPLE **M/M/1 versus M/D/1**

Consider a statistical multiplexer that has a transmission line with a speed of R = 64 kbps. Suppose that the average packet length is $E[L]$ = 1000 bytes = 8000 bits and that the average arrival rate is 4 packets/second. Compare the average packet delay for constant-length packets to exponentially distributed packets.

The packet service rate is μ = 64,000 bps/8000 bits/packet = 8 packets/second. Since λ = 4 packets/second, the load is $\rho = \lambda/\mu = 4/8 = 1/2$. If packets have an exponential density function, then $E[T_M] = 2/8$ = 250 ms. If packets are constant, we then have $E[T_D] = 1.5/8$ = 187 ms.

EXAMPLE **Effect of Header Overhead on Goodput**

Consider a statistical multiplexer that has a transmission line with a speed of R = 64 kbps. The *goodput* is the amount of actual user information that is transmitted. Suppose that each packet has 40 bytes of IP and TCP header and that packets are constant in length. Find the useful throughput if the total packet length is 200 bytes, 400 bytes, 800 bytes, and 1200 bytes.

Let L be the packet length in bytes. The packet service rate is then $\mu = 64{,}000/8L$ packets/second. Let λ be the packet arrival rate; then the load is $\rho = \lambda/\mu = 8\lambda L/64{,}000$, and the goodput is $\gamma = 8\lambda(L - 40)$ bps. Figure 5.57 shows the average packet delay versus the goodput. Note that the delay is given in seconds. Thus when the multiplexer has longer packets, the delay at low goodput is higher, since it takes longer to transmit each packet. On the other hand, the longer packets also incur less header overhead in terms of percentage and hence have a higher maximum achievable goodput. Indeed, it is easy to show that the maximum goodput is given by $\gamma = (1 - 40/L)\ 64{,}000$ bps.

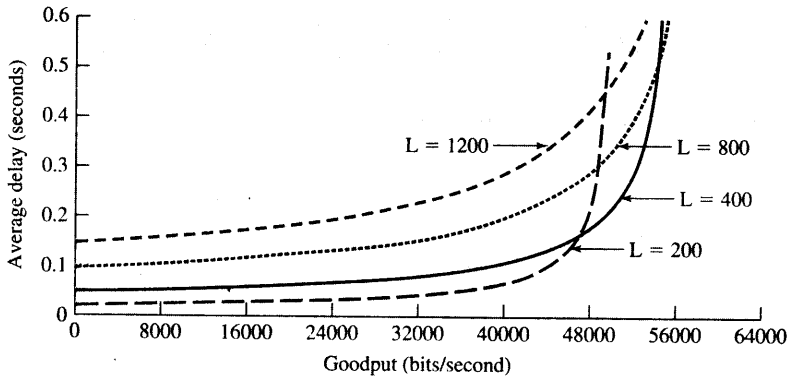


FIGURE 5.57 Effect of header overhead on packet delay and goodput.

MEASUREMENT, MODELS, AND REAL PERFORMANCE

The flow of traffic in modern networks is extremely complex. The various interacting layers of protocols, for example, HTTP, TCP/IP, Ethernet, and PPP, provide a framework for the interaction of the information flows generated by a multiplicity of applications. The times when applications make requests are not scheduled, and the pattern of packets transmitted is never quite the same. The challenge of the network operator is to have enough resources (bandwidth, buffering, processing) in the right configuration to handle the traffic at any given time. Measurement and performance modeling are essential tools in carrying out this task.

Measurement helps identify patterns in the apparent chaos that is network traffic. Time-of-day and day-of-week cycles in traffic levels and traffic patterns tend to persist, and their changes can be tracked by measurement. The traffic flows generated by specific types of applications can also be characterized and tracked over time. Sudden surges and changes in traffic patterns can also be recognized, given the template provided by what is "normal." Longer-term trends in traffic levels and patterns are also used to plan the deployment of network equipment.

Traffic models like the ones introduced in this section are useful in understanding the dynamics and interplay between the basic parameters that determine performance. Models are intended to simplify and capture only the essential features of a situation. In doing so, models can be used to predict the (approximate) performance in a given situation, and so they can form the basis for making decisions regarding traffic management. However models are merely our attempt to characterize what is going on "out there." In fact, models that do not capture all the relevant features of a situation can lead to incorrect conclusions. Measurement can be used to close the loop between models and real performance; by comparing predictions of the models with actual observations, we can modify and fine-tune the models themselves.

PERFORMANCE IMPROVEMENTS FROM FLOW AGGREGATION

Suppose that we initially have 24 individual statistical multiplexers, each with a 64 kbps line, and that we aggregate the individual packet arrivals into one stream and apply it to a statistical multiplexer with a 24×64 kbps line. Suppose that each multiplexer has an arrival rate $\lambda = 4$ packets/second, a service rate $\mu = 8$ packets/second, and hence a load $\rho = 1/2$. If we use an M/M/1 model the average packet delay in each individual multiplexer is $2/8 = 250$ ms. On the other hand, the combined system has arrival rate $\lambda' = 24 \times 4$ packets/second, a service rate $\mu' = 24 \times 8$ packets/second, and hence a load $\rho' = \frac{1}{2} = \rho$. A key property of exponential interarrivals is that the merged arrival streams also have exponential interarrivals. Thus the same expression for the average delay holds, and the average packet delay in the combined system is $2/(8 \times 24) = 250/24 \approx 10$ ms. The average packet delay has been reduced by a factor of 24! We leave as an exercise the task of showing that a factor of 24 improvement also results when packets are constant.

The improved performance that results when the arrival rate and the transmission rate are increased by the same factor k is simple to explain. In effect the time scale of the system is reduced by a factor k. The interarrivals and service times of packets with respect to each other remain unchanged. Packets in the system find the same number of packets ahead of them in queue as in the old system. The only difference is that the packets are moving k times faster.

Suppose instead that each individual system can only hold 10 packets, and so we model it with an M/M/1/K system with $K = 10$. The packet loss probability for each individual system is given by $P_{loss} = (1/2)(1/2)^{10}/(1 - (1/2)^{11}) = 4.88 \times 10^{-4}$. Suppose that the combined multiplexer has the same total buffers as the individual multiplexers; that is, $K' = 24 \times 10 = 240$. The combined multiplexer then has loss probability $P'_{loss} = (1/2)(1/2)^{240}/(1 - (1/2)^{241}) = 2.83 \times 10^{-73}$. This is a huge improvement in packet loss performance!

Thus we find that for Poisson arrivals (that is, exponential packet interarrivals) increasing the size or scale of the system by aggregating packet flows leads to improved performance in terms of delay and loss.

5.7.2 Speech Interpolation and the Multiplexing of Packetized Speech

In telephone networks a connection is set up so that speech samples traverse the network in a single uninterrupted stream. Normal conversational speech, however, is moderately bursty as it contains silence periods. In this section we consider the multiplexing of packetized speech.

First, consider n one-way established telephone connections. In a typical conversation a person is actively speaking less than half of the time. The rest of the time is taken up by the pauses inherent in speech and by listening to the other person. This characteristic of speech can be exploited to enable m telephone lines to carry the n conversations, where m is less than n. Because a connection produces active speech only about half of the time, we expect that the minimum number of trunks required is approximately $n/2$. This problem was first addressed in speech interpolation systems in transatlantic undersea cable telephone communications.

As shown in Figure 5.58, the n speakers generate bursts of active speech that are separated by periods of silence. The early systems monitored the speech activity in each line and multiplexed active bursts onto available lines. Associated signaling allowed the

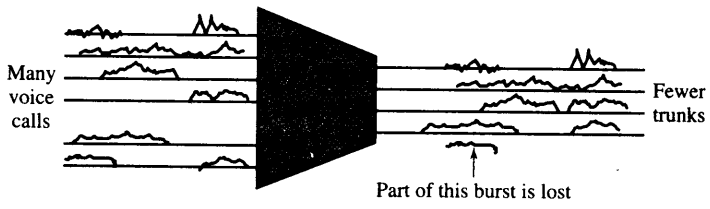


FIGURE 5.58 Multiplexing of bursty speech.

receiving end to reconstitute the original speech signal by reinserting the appropriate silence periods. Clearly, the technique involves juggling the n calls, using the m available trunks. It is inevitable that from time to time there will be more active bursts than the number of lines. The early systems simply discarded the excess bursts, which then resulted in "clipping" in the recovered speech signal. The resulting speech quality was acceptable as long as the amount of lost speech was kept below a certain level.

This concentration technique can also be implemented and enhanced by using digital signal processing. The digitized speech signal corresponding to some duration, say, 10 ms, is used to form a packet to which a header is attached. Packets are classified as active or silent, depending on the speech activity. Silence packets are discarded. Suppose that one line can transmit all the packets, active and silent, for a single conversation. The digital system now tries to use m lines to transmit the active packets from $n > m$ calls; that is, the digital transmission line can send m packets per 10 ms period. Whenever the number of active packets exceeds m, the excess number is discarded. Given n speakers, we need to find the number of trunks required so that the **speech loss**, that is, the fraction of active speech that is discarded, is kept below a certain level.

Let p be the probability that a packet is active. Then it can be shown that the speech loss is given by

$$\text{speech loss} = \frac{\sum_{k=m+1}^{n} (k-m)\binom{n}{k} p^k (1-p)^{n-k}}{np} \quad \text{where } \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{5.28}$$

The denominator in Equation (5.28) is the average number of active packets in a 10 ms period. The numerator is the average number of active packets in excess of m. Figure 5.59 shows the speech loss for various numbers of speakers, that is, $n = 24, 32, 40$, and 48. We assume that $p = 0.4$; that is, 40 percent of packets are active.

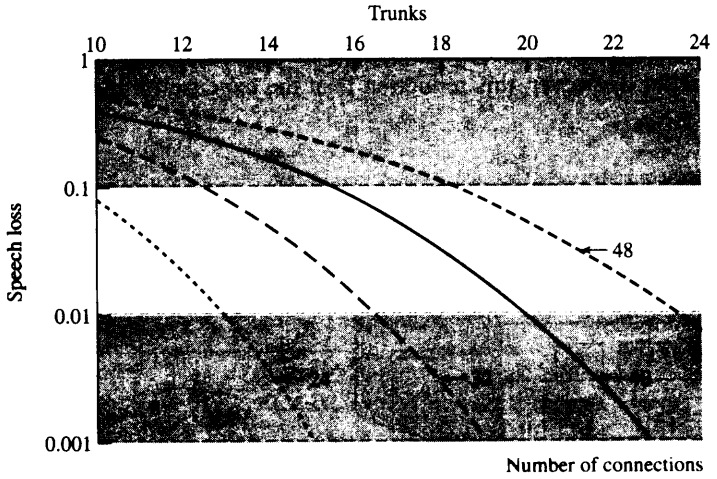


FIGURE 5.59 Speech loss—number of speakers versus number of trunks.

TABLE 5.1 Trunks required for 1% speech loss.

Speakers	Trunks	Multiplexing gain	Utilization
24	13	1.85	0.74
32	16	2.00	0.80
40	20	2.00	0.80
48	23	2.09	0.83

As expected, the speech loss decreases as the number of trunks increases. The acceptable level of speech loss is approximately 1%.

Table 5.1 shows the number of trunks needed to meet a 1% speech loss requirement. The **multiplexing gain** is defined as the ratio of the number of connections, that is, speakers, to the number of trunks actually provided. We note that because $p = 0.4$, the maximum multiplexing gain is approximately 2.5. Table 5.1 also shows the utilization, which is the percentage of time that the trunks are in use transmitting active packets. If the number of connections is n, then each 10 ms period produces np active packets on the average. Because the speech loss is 1%, the number of active packets transmitted per 10 ms period is .99 np. Thus the utilization is given by .99 np/m.

Let us now consider the statistical multiplexing of packetized speech using delay and loss. The speech signal is digitized, say, at a rate of 8000 samples/second and eight bits/sample. As each sample is obtained, it is inserted into a fixed-length packet that has appropriate header information and that holds the samples produced in some time interval, say, 10 ms. Note that the first sample inserted into the packet must wait 10 ms until the packet is filled. This initial delay is called the **packetization delay**.

Once the packet is full, and if it contains active speech, it is passed to a statistical multiplexer that operates in the same way as the data packet multiplexer discussed in Figure 5.50. As shown in Figure 5.60, the multiplexer accepts speech packets from various conversations and holds each packet in the queue until the transmission line becomes available. Note that the addition of buffering reduces the number of packets that need to be discarded in comparison to the digital speech interpolation system discussed earlier. However, this reduction is at the expense of additional delay while

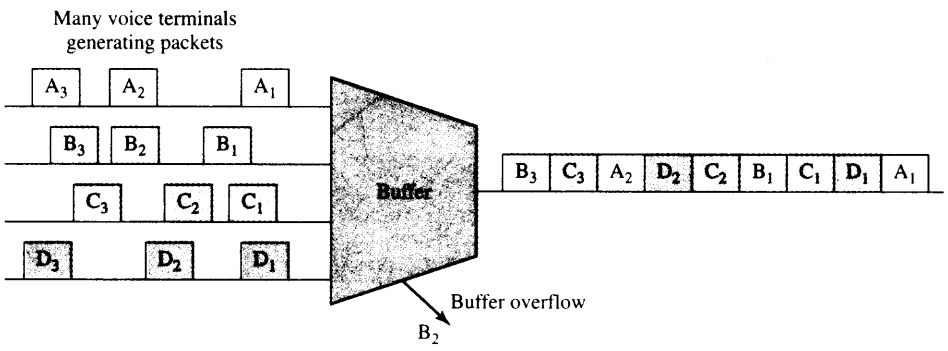


FIGURE 5.60 Statistical multiplexing of packetized speech.

ON-OFF MODELS, LOSS AND DELAY, AND MEMORY IN THE SYSTEM

The ON-OFF nature of the voice model is typical of many applications that alternate between periods of activity when data is generated and idle periods when no traffic is produced. Suppose that a system is operated on a *loss* basis—that is, any information in excess of what can be handled immediately is discarded—and suppose that the source *does not* retransmit or in any other way regenerate the information. For these systems the result given by Equation (5.28) applies for n identical users as long as they generate information independently of each other. The number of packets discarded at a particular time slot is characterized completely by the probabilities in the equation. In particular, the result *does not* depend on the length of the ON periods or the length of the OFF periods. Once the packets are discarded, the system can completely forget about the lost packets because they are irrelevant to future behavior.

If the system is operated on a *delay* basis, then any packets that cannot be transmitted at a given time are buffered for later transmission. In other words, delay systems must "remember" all excess packets. This memory causes past events (surges in arrivals) to influence the future (congested queues and longer delays). In this case the durations of ON and OFF periods *do* matter. The cooccurrence of many long ON periods implies a prolonged period of higher than average arrivals and correspondingly large queue buildups and delays.

The buffering in the multiplexer is not the only source of memory in the packet arrival process. Protocols that involve retransmission or adjustment of transmission rate to network conditions are another source of memory. Finally, the users themselves have memory and will reattempt a transmission at a later time when their requests are not met. This behavior can be another source of long-term dependencies in the traffic arrival process.

waiting in queue for transmission. Note also that in packet speech, packet arrivals are periodic when the conversation is generating active speech. This situation differs from the exponential interarrivals discussed in Section 5.7.1 for data traffic. Nevertheless the queueing delays experienced by speech packets are random in nature and depend on the arrivals from the other conversations. Consequently, the delay experienced in traversing the multiplexing link is not constant, and so the packets experience delay jitter, as shown in Figure 5.61. A playout procedure along the lines of the timing recovery methods discussed earlier in this chapter is required to compensate for the delay jitter.

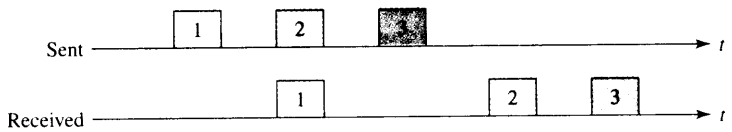


FIGURE 5.61 Packets experience delay jitter during transmission through multiplexer.

The real-time nature of speech requires that the packets not experience an end-to-end delay greater than some value, usually around 250 ms. Hence excessive buffering should be avoided for packet speech multiplexers. If the buffers become too large, packets will experience greater delays. Packets that arrive after the required end-to-end delay are useless and will be discarded. Therefore, no advantage accrues from increasing the buffer size beyond a certain point.

SUMMARY

This chapter had two primary objectives. The first objective was to discuss the *peer-to-peer protocols* that operate within a layer and the *services* that they provide to the layer above them. We considered the simplest case in which only two peer processes are involved in executing a protocol to provide a service, namely, the transfer of information according to a service model that provides features, such as reliability and sequencing, multiplexing, arbitrary message size, timing, and flow control and pacing.

We developed ARQ protocols as an example of peer-to-peer protocols whose function is to provide reliable data transfer service. We saw that the operation of a protocol involves the exchange of *PDUs* that consist of *headers* with protocol control information and of user SDUs. We showed how a protocol is specified in terms of a *state* machine that dictates what actions each protocol entity is to take when an event occurs. We saw how the peer processes *exchange control information* through specific control PDUs (to set up a connection, to release a connection, to provide acknowledgments) as well as through control information that is piggybacked onto data PDUs (sequence numbers, CRC check sums, ACKs). We also saw how ARQ protocols depend on *timers* to keep the protocol alive.

We introduced the *sliding-window* mechanism as a means of providing sequence numbering within a finite sequence space. For the ARQ protocols to operate correctly, only a portion of the sequence space can be in use at any given time. The sliding-window mechanism can also provide *flow control* to regulate the rate at which a sender transmits information to a receiver. The use of timestamps by peer-to-peer protocols to assist in the transfer of information with timing requirements was also discussed. We introduced the Transmission Control Protocol (TCP) which uses a form of Selective Repeat ARQ to provide *connection-oriented, reliable stream service* and flow control end-to-end across connectionless packet networks.

The second objective of this chapter was to introduce the data link layer and the two important examples of data link control: PPP and HDLC. We saw that the task of data link control protocols is to provide for the *connection-oriented or connectionless transfer of blocks of information across a data link.* We introduced the basic, but essential, functions of *bit and byte stuffing* as well as frame length indication to provide *framing* that demarcates the boundary of data link PDUs. We also examined the structure of data link frames, focusing on the control information in the header that directs the operation of the protocol. We considered *PPP*, which provides a versatile data link protocol with enhanced link monitoring, with authentication capability, and with the ability to simultaneously support several network layer protocols. We also discussed

HDLC, and found that it provides a toolkit of control frames and protocol mechanisms for a wide range of data link protocols, from simple unacknowledged connectionless transfer to full-fledged connection-oriented reliable, sequenced transfer.

An essential feature of networks is the use of resource sharing to achieve economies of scale. In the final section we introduced simple models that offer insight into the performance of statistical multiplexers that allow packets from different flows to share a common data link. Bandwidth sharing at this level is a key feature of packet networks. We identified the packet arrival pattern and the packet length distribution as key parameters that determine delay and loss performance. We also indicated the challenges posed to network designers by rapidly evolving network-based applications that can result in sudden changes in network traffic that can have dramatic impact on network performance.

CHECKLIST OF IMPORTANT TERMS

ACK timer
acknowledgment frame (ACK)
advertised window
ARQ protocol
◆ arrival rate λ
asynchronous balanced mode (ABM)
Automatic Repeat Request (ARQ)
best-effort service
bit stuffing
byte stuffing
Challenge-Handshake Authentication Protocol (CHAP)
connectionless service
connection-oriented service
control frame
data link control
data link layer
delay-bandwidth product
Disconnect (DISC)
end-to-end requirement
flow control
frame
Frame Reject (FRMR)
framing
Generic Framing Procedure (GFP)
Go-Back-N ARQ
header
High-level Data Link Control (HDLC)
I-frame timer
information frame (I-frame)
Link Control Protocol (LCP)
◆ load ρ
◆ maximum departure rate μ
◆ multiplexing gain
negative acknowledgment frame (NAK)
Network Control Protocol (NCP)
normal response mode (NRM)
◆ packet loss probability
◆ packetization delay
Packet over SONET (POS)
Password Authentication Protocol (PAP)
peer process
peer-to-peer protocol
piggybacking
pipeline
Point-to-Point Protocol (PPP)
◆ Poisson arrival process
protocol data unit (PDU)
push command
quality-of-service (QoS)
Receive Not Ready (RNR)
Receive Ready (RR)
receive window
Reject (REJ)
round-trip time (RTT)
Selective Reject (SREJ)
Selective Repeat (ARQ)
send window

sequence number
service access point (SAP)
service data unit (SDU)
service model
Set Asynchronous Balanced Mode (SABM)
Set Normal Response Mode (SNRM)
sliding-window protocol
◆ speech loss
stop-and-Wait ARQ
supervisory frame
time-out
timing jitter
timing recovery
Transmission Control Protocol (TCP)
transmission efficiency
unnumbered acknowledgment (UA)
unnumbered frame
window size

FURTHER READING

Bertsekas, D. and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, 1992.

Bonenfant, P. and A. Rodriguez-Moral, "The Generic Framing Procedure: An Overview," IEEE Communications Magazine, May, 2002.

Davies, D. W., D. L. A. Barber, W. L. Price, and C. M. Solomonides, *Computer Networks and Their Protocols*, John Wiley & Sons, New York, 1979.

Jain, B. N. and A. K. Agrawala, *Open Systems Interconnection: Its Architecture and Protocol*, McGraw-Hill, New York, 1993.

Schwartz, M., *Telecommunication Networks: Protocols, Modeling, and Analysis*, Addison-Wesley, Reading, Massachusetts, 1987.

RFC 793, J. Postel (ed.), "Transmission Control Protocol: DARPA Internet Program Protocol Specification," 1981.

RFC 1661, W. Simpson, "The Point-to-Point Protocol (PPP)," July 1994.

PROBLEMS

5.1. Explain the difference between connectionless unacknowledged service and connectionless acknowledged service. How do the protocols that provide these services differ?

5.2. Explain the difference between connection-oriented acknowledged service and connectionless acknowledged service. How do the protocols that provide these services differ?

5.3. Suppose that the two end systems α and β in Figure 5.6 communicate over a connection-oriented packet network. Suppose that station α sends a 10-kilobyte message to station β and that all packets are restricted to 1000 bytes (neglect headers); assume that each packet can be accommodated in a data link frame. For each of the links, let p be the probability that a frame incurs errors during transmission.

(a) Suppose that the data link control just transfers frames and does not implement error control. Find the probability that the *message* arrives without errors at station β.

(b) Suppose that error recovery is carried out end to end and that if there are any errors, the entire message is retransmitted. How many times does the message have to be retransmitted on average?

(c) Suppose that the error recovery is carried out end to end on a packet by packet basis. What is the total number of packet transmissions required to transfer the entire message?

5.4. Suppose that two peer-to-peer processes provide a service that involves the transfer of discrete messages. Suppose that the peer processes are allowed to exchange PDUs that have a maximum size of M bytes, including H bytes of header. Suppose that a PDU is not allowed to carry information from more than one message.
(a) Develop an approach that allows the peer processes to exchange messages of arbitrary size.
(b) What essential control information needs to be exchanged between the peer processes?
(c) Now suppose that the message transfer service provided by the peer processes is shared by several message source-destination pairs. Is additional control information required, and if so, where should it be placed?

5.5. Suppose that two peer-to-peer processes provide a service that involves the transfer of a stream of bytes. Suppose that the peer processes are allowed to exchange PDUs that have a maximum size of M bytes, including H bytes of header.
(a) Develop an approach that allows the peer processes to transfer the stream of bytes in a manner that uses the transmission line efficiently. What control information is required in each PDU?
(b) Suppose that the bytes in the stream arrive sporadically. What is a reasonable way to balance efficiency and delay at the transmitter? What control information is required in each PDU?
(c) Suppose that the bytes arrive at a constant rate and that no byte is to be delayed by more than T seconds. Does this requirement have an impact on the efficiency?
(d) Suppose that the bytes arrive at a variable rate and that no byte is to be delayed by more than T seconds. Is there a way to meet this requirement?

5.6. Suppose that two peer-to-peer processes provide a service that involves the transfer of a stream of bytes. Develop an approach that allows the stream transfer service to be shared by several pairs of users in the following cases:
(a) The bytes from each user pair arrive at the same constant rate.
(b) The bytes from the user pairs arrive sporadically and at different rates.

5.7. Consider the transfer of a single real-time telephone voice signal across a packet network. Suppose that each voice sample should not be delayed by more than 20 ms.
(a) Discuss which of the following are relevant to meeting the requirements of this transfer: handling of arbitrary message size; reliability and sequencing; pacing and flow control; timing; addressing; and privacy, integrity, and authentication.
(b) Compare a hop-by-hop approach to an end-to-end approach to meeting the requirements of the voice signal.

5.8. Suppose that a packet network is used to transfer *all* the voice signals that arrive at the base station of a cellular telephone network to a telephone office. Suppose that each voice sample should not be delayed by more than 20 ms.
(a) Discuss which of the following are relevant to meeting the requirements of this transfer: handling of arbitrary message size; reliability and sequencing; pacing and flow control; timing; addressing; and privacy, integrity, and authentication.
(b) Are the requirements the same in the opposite direction from the telephone office to the base station?

(c) Do the answers to parts (a) and (b) change if the signals arriving at the base station include e-mail and other short messages?

5.9. Suppose that streaming video information is transferred from a server to a user over a packet network.
(a) Discuss which of the following are relevant to meeting the requirements of this transfer: handling of arbitrary message size; reliability and sequencing; pacing and flow control; timing; addressing; and privacy, integrity, and authentication.
(b) Suppose that the user has basic VCR features through control messages that are transferred from the user to the server. What are the adaptation requirements for the control messages?

5.10. Discuss the merits of the end-to-end versus hop-by-hop approaches to providing a constant transfer delay for information transferred from a sending end system to a receiving end system.

5.11. Consider the Stop-and-Wait protocol as described in the chapter. Suppose that the protocol is modified so that each time a frame is found in error at either the sender or receiver, the last transmitted frame is immediately resent.
(a) Show that the protocol still operates correctly.
(b) Does the state transition diagram need to be modified to describe the new operation?
(c) What is the main effect of introducing the immediate-retransmission feature?

5.12. In Stop-and-Wait ARQ why should the receiver always send an acknowledgment message each time it receives a frame with the wrong sequence number?

5.13. Discuss the factors that should be considered in deciding whether an ARQ protocol should act on a frame in which errors are detected.

5.14. Suppose that a network layer entity requests its data link layer to set up a connection to another network layer entity. To set up a connection in a data link, the initiating data link entity sends a SETUP frame, (such as SABM in Figure 5.47). Upon receiving such a frame, the receiving data link entity sends an acknowledgment frame confirming receipt of the SETUP frame. Upon receiving this acknowledgment, the initiating entity can inform its network layer that the connection has been set up and is ready to transfer information. This situation provides an example of how *unnumbered acknowledgments* can arise for confirmed services.
(a) Reexamine Figure 5.10 and Figure 5.11 with respect to error events that can take place and explain how these events are handled so that connection setup can take place reliably.
(b) To terminate the connection, either data link layer can send a DISC frame that is then acknowledged by an unnumbered acknowledgment. Discuss the effect of the above error events and how they can be dealt with.
(c) Suppose that an initiating station sends a SETUP frame twice but that the corresponding ACK times are delayed a long time. Just as the ACK frames from the original transmissions are about to arrive, the initiating station gives up and sends a DISC frame followed by another SETUP frame. What goes wrong if the SETUP frame is lost?

5.15. A 1 Mbyte file is to be transmitted over a 1 Mbps communication line that has a bit error rate of $p = 10^{-6}$.

(a) What is the probability that the entire file is transmitted without errors? Note for n large and p very small, $(1 - p)^n \approx e^{-np}$.

(b) The file is broken up into N equal-sized blocks that are transmitted separately. What is the probability that all the blocks arrive without error? Is dividing the file into blocks useful?

(c) Suppose the propagation delay is negligible, explain how Stop-and-Wait ARQ can help deliver the file in error-free form. On the average how long does it take to deliver the file if the ARQ transmits the entire file each time?

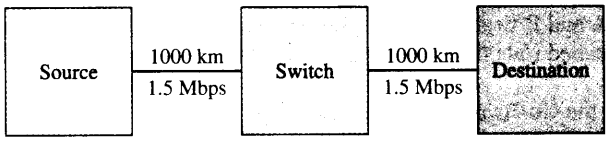
(d) Now consider breaking up the file into N blocks. (Neglect the overhead for the header and CRC bits.) On the average how long does it take to deliver the file if the ARQ transmits the blocks one at a time? Evaluate your answer for $N = 80$, 800, and 8000.

(e) Explain qualitatively what happens to the answer in part (d) when the overhead is taken into account.

5.16. Consider the state transition diagram for Stop-and-Wait ARQ in Figure 5.12. Let P_f be the probability of frame error in going from station A to station B and let P_a be the probability of ACK error in going from B to A. Suppose that information frames are two units long, ACK frames are one unit long, and propagation and processing delays are negligible. What is the average time that it takes to go from state (0,0) to state (0,1)? What is the average time that it then takes to go from state (0,1) to state (1,1)? What is the throughput of the system in information frames/second?

5.17. Write a program for the transmitter and the receiver implementing Stop-and-Wait ARQ over a data link that can introduce errors in transmission. Assume station A has an unlimited supply of frames to send to station B. Only ACK frames are sent from station B to station A. Hint: Identify each event that can take place at the transmitter and receiver and specify the required action.

5.18. A 64-kilobyte message is to be transmitted from the source to the destination, as shown below. The network limits packets to a maximum size of two kilobytes, and each packet has a 32-byte header. The transmission lines in the network have a bit error rate of 10^{-6}, and Stop-and-Wait ARQ is used in each transmission line. How long does it take on the average to get the message from the source to the destination? Assume that the signal propagates at a speed of 2×10^5 km/second.



5.19. Suppose that a Stop-and-Wait ARQ system has a time-out value that is less than the time required to receive an acknowledgment. Sketch the sequence of frame exchanges that transpire between two stations when station A sends five frames to station B and no errors occur during transmission.

5.20. The Trivial File Transfer Protocol (RFC 1350) is an application layer protocol that uses the Stop-and-Wait protocol. To transfer a file from a server to a client, the server breaks the file into blocks of 512 bytes and sends these blocks to the client using Stop-and-Wait ARQ. Find the efficiency in transmitting a 1 MB file over a 10 Mbps Ethernet LAN that has a diameter of 300 meters. Assume that the transmissions are error free and that each packet has 60 bytes of header attached.

5.21. Compare the operation of Stop-and-Wait ARQ and bidirectional Go-Back-N ARQ with a window size of 1. Sketch out a sequence of frame exchanges using each of these protocols and observe how the protocols react to the loss of an information frame and to the loss of an acknowledgment frame.

5.22. Consider the various combinations of communication channels with bit rates of 1 Mbps, 10 Mbps, 100 Mbps, and 1 Gbps over links that have round-trip times of 10 msec, 1 msec, and 100 msec.
(a) Find the delay-bandwidth product for each of the 12 combinations of speed and distance.
(b) Suppose that 32-bit sequence numbers are used to transmit blocks of 1000 bytes over the above channels. How long does it take for the sequence numbers to wrap around, that is, to go from 0 up to 2^m?
(c) Now suppose the 32-bit sequence numbers are used to count individual transmitted bytes. How long does it take for the sequence numbers to wrap around?

5.23. Consider a bidirectional link that uses Go-Back-N with $N = 7$. Suppose that all frames are one unit long and that they use a time-out value of 2. Assume the propagation is 0.5 unit and the processing time is negligible. Assume the ACK timer is one unit long. Assuming stations A and B begin with their sequence numbers set to zero, show the pattern of transmissions and associated state transitions for the following sequences of events:
(a) Station A sends six frames in a row, starting at $t = 0$. All frames are received correctly.
(b) Station A sends six frames in a row, starting at $t = 0$. All frames are received correctly, but frame 3 is lost.
(c) Station A sends six frames in a row, starting at $t = 0$. Station B sends six frames in a row starting at $t = 0.25$. All frames are received correctly.

5.24. Consider a bidirectional link that uses Go-Back-N with $N = 3$. Suppose that frames from station A to station B are one unit long and use a time-out value of 2. Frames in the opposite directions are 2.5 units long and use a time-out value of 4. Assume that propagation and processing times are negligible, that the stations have an unlimited number of frames ready for transmission, and that all ACKs are piggybacked onto information frames. Assuming stations A and B begin with their sequence numbers set to zero, show the transmissions and associated state transitions that result when there are no transmission errors.

5.25. Consider the Go-Back-N ARQ protocol.
(a) What can go wrong if the ACK timer is not used?
(b) Show how the frame timers can be maintained as an ordered list where the time-out instant of each frame is stated relative to the time-out value of the previous frame.
(c) What changes if each frame is acknowledged individually instead of by using a cumulative acknowledgment (R_{next} acknowledges all frames up to $R_{next} - 1$)?

5.26. Suppose that instead of Go-Back-N ARQ, N simultaneous Stop-and-Wait ARQ processes are run in parallel over the same transmission channel. Each SDU is assigned to one of

the N processes that is currently idle. The processes that have frames to send take turns transmitting in round-robin fashion. The frames carry the binary send sequence number as well as an ID identifying which ARQ process the frame belongs to. Acknowledgments for *all* ARQ processes are piggybacked onto *every* frame.

(a) Qualitatively, compare the relative performance of this protocol with Go-Back-N ARQ and with Stop-and-Wait ARQ.

(b) How does the service offered by this protocol differ from the service offered by Go-Back-N ARQ?

5.27. Write a program for the transmitter and the receiver implementing Go-Back-N ARQ over a data link that can introduce errors in transmission.

(a) Identify which variables need to be maintained.

(b) The program loops continuously waiting for an event to occur that requires some action to take place. Identify the main events that can occur in the transmitter. Identify the main events that can occur in the receiver.

5.28. Modify the program in Problem 5.27 to implement Selective Repeat ARQ.

5.29. Three possible strategies for sending ACK frames in a Go-Back-N setting are as follows: send an ACK frame immediately after each frame is received, send an ACK frame after every other frame is received, and send an ACK frame when the next piggyback opportunity arises. Which of these strategies are appropriate for the following situations?

(a) An interactive application produces a packet to send each keystroke from the client; the server echoes each keystroke that it receives from the client.

(b) A bulk data transfer application where a server sends a large file that is segmented in a number of full-size packets that are to be transferred to the client.

5.30. Consider a bidirectional link that uses Selective Repeat ARQ with a window size of $N = 4$. Suppose that all frames are one unit long and use a time-out value of 2. Assume that the one-way propagation delay is 0.5 time unit, the processing times are negligible, and the ACK timer is one unit long. Assuming stations A and B begin with their sequence numbers set to zero, show the pattern of transmissions and associated state transitions for the following sequences of events:

(a) Station A sends six frames in a row, starting at $t = 0$. All frames are received correctly.

(b) Station A sends six frames in a row, starting at $t = 0$. All frames are received correctly, but frame 3 is lost.

(c) Station A sends six frames in a row, starting at $t = 0$. Station B sends six frames in a row, starting at $t = 0.25$. All frames are received correctly.

5.31. In the chapter we showed that if the transmit and receive maximum window sizes are both equal to the available sequence number space, then Selective Repeat ARQ will work correctly. Rework the arguments presented in the chapter to show that if the sum of the transmit and receive maximum window sizes equals the available sequence number space, then Selective Repeat ARQ will work correctly.

5.32. Suppose that Selective Repeat ARQ is modified so that ACK messages contain a list of the next m frames that the transmitter expects to receive.

(a) How does the protocol need to be modified to accommodate this change?

(b) What is the effect of the change on protocol performance?

5.33. A telephone modem is used to connect a personal computer to a host computer. The speed of the modem is 56 kbps and the one-way propagation delay is 100 ms.
(a) Find the efficiency for Stop-and-Wait ARQ if the frame size is 256 bytes; 512 bytes. Assume a bit error rate of 10^{-4}.
(b) Find the efficiency of Go-Back-N if three-bit sequence numbering is used with frame sizes of 256 bytes; 512 bytes. Assume a bit error rate of 10^{-4}.

5.34. A communication link provides 1 Mbps for communications between the earth and the moon. The link sends color images from the moon. Each image consists of 10,000 × 10,000 pixels, and 16 bits are used for each of the three color components of each pixel.
(a) How many images/second can be transmitted over the link?
(b) If each image is transmitted as a single block, how long does it take to get an acknowledgment back from earth? The distance between earth and the moon is approximately 375,000 km.
(c) Suppose that the bit error rate is 10^{-5}, compare Go-Back-N and Selective Repeat ARQ in terms of their ability to provide reliable transfer of these images from the moon to earth. Optimize the frame size for each case using trial and error.

5.35. Two computers are connected by an intercontinental link with a one-way propagation delay of 100 ms. The computers exchange 1-Megabyte files that they need delivered in 250 ms or less. The transmission lines have a speed of R Mbps, and the bit error rate is 10^{-8}. Design a transmission system by selecting the bit rate R, the ARQ protocol, and the frame size.

5.36. Find the optimum frame length n_f that maximizes transmission efficiency for a channel with random bit errors by taking the derivative and setting it to zero for the following protocols:
(a) Stop-and-Wait ARQ.
(b) Go-Back-N ARQ.
(c) Selective Repeat ARQ.
(d) Find the optimum frame length for a 1 Mbps channel with 10 ms reaction time, 25-byte overhead, 25-byte ACK frame, and $p = 10^{-4}$, 10^{-5}, and 10^{-6}.

5.37. Suppose station A sends information to station B on a data link that operates at a speed of 10 Mbps and that station B has a 1-Megabit buffer to receive information from A. Suppose that the application at station B reads information from the receive buffer at a rate of 1 Mbps. Assuming that station A has an unlimited amount of information to send, sketch the sequence of transfers on the data link if Stop-and-Wait ARQ is used to prevent buffer overflow at station B. Consider the following cases:
(a) One-way propagation delay is 1 microsecond.
(b) One-way propagation delay is 1 ms.
(c) One-way propagation delay is 100 ms.

5.38. Redo Problem 5.37 using Xon/Xoff flow control.

5.39. Suppose station A sends information to station B over a two-hop path. The data link in the first hop operates at a speed of 10 Mbps, and the data link in the second hop operates at a speed of 100 kbps. Station B has a 1-Megabit buffer to receive information from A, and the application at station B reads information from the receive buffer at a rate of 1 Mbps. Assuming that station A has an unlimited amount of information to send, sketch

the sequence of transfers on the data link if Stop-and-Wait ARQ is used on an end-to-end basis to prevent buffer overflow at station B.
(a) One-way propagation delay in data link 1 and in data link 2 is 1 ms.
(b) One-way propagation delay in data link 1 and in data link 2 is 100 ms.

5.40. A sequence of fixed-length packets carrying digital audio signal is transmitted over a packet network. A packet is produced every 10 ms. The transfer delays incurred by the first 10 packets are 45 ms, 50 ms, 53 ms, 46 ms, 30 ms, 40 ms, 46 ms, 49 ms, 55 ms and 51 ms.
(a) Sketch the sequence of packet transmission times and packet arrival times.
(b) Find the delay that is inserted for each packet at the receiver to produce a fixed end-to-end delay of 75 ms.
(c) Sketch the contents of the buffer at the receiver as a function of time.

5.41. Consider an application in which information that is generated at a constant rate is transferred over a packet network so timing recovery is required at the receiver.
(a) What is the relationship between the maximum acceptable delay and the playout buffer?
(b) What is the impact of the bit rate of the application information stream on the buffer requirements?
(c) What is the effect of jitter on the buffer size design?

5.42. A speech signal is sampled at a rate of 8000 samples/second. Packets of speech are formed by packing 10 ms worth of samples into each payload. Timestamps are attached to the packets prior to transmission and used to perform error recovery at the receiver. Suppose that the timestamp is obtained by sampling a clock that advances every Δ seconds. Is there a minimum value that is required for Δ? If so, what is it?

5.43. Suppose that UDP is used to carry the speech signal in the previous problem. What is the total bit rate consumed by the sequence of UDP PDUs? What is the percent overhead?

5.44. Suppose that PDUs contain both timestamps and sequence numbers. Can the timestamps and sequence numbers be combined to provide a larger sequence number space? If so, should the timestamps or the sequence numbers occupy the most significant bit locations?

5.45. Consider the timestamp method for timing recovery discussed in the chapter.
(a) Find an expression that relates the difference frequency Δf to the number of cycles M and N.
(b) Explain why only M needs to be sent.
(c) Explain how the receiver uses this value of M to control the playout procedure.

5.46. In Figure 5.32, determine the number of bytes exchanged between client and server in each direction.

5.47. Suppose that the delays experienced by segments traversing the network are equally likely to be any value in the interval [50 ms, 75 ms].
(a) Find the mean and standard deviation of the delay.
(b) Most computer languages have a function for generating uniformly distributed random variables. Use this function in a short program to generate random times in the above interval. Also, calculate t_{RTT} and d_{RTT} and compare to part (a).

5.48. Suppose that the advertised window is 1 Mbyte long. If a sequence number is selected at random from the entire sequence number space, what is the probability that the sequence number falls inside the advertised window?

5.49. Explain the relationship between advertised window size, RTT, delay-bandwidth product, and the maximum achievable throughput in TCP.
(a) Plot the maximum achievable throughput versus delay-bandwidth product for an advertised window size of 65,535 bytes.
(b) In the preceding plot include the maximum achievable throughput when the above window size is scaled up by a factor of 2^K, where $K = 4, 8, 12$.
(c) Place the following scenarios in the plot obtained in part (b): Ethernet with 1 Gbps and distance 100 meters; 2.4 Gbps and distance of 6000 km; satellite link with speed of 45 Mbps and RTT of 500 ms; 40 Gbps link with distance of 6000 km.

5.50. Use a network analyzer to capture the sequence of packets in a TCP connection. Analyze the contents of the segments that open and close the TCP connection. Estimate the rate at which information is transferred by examining the frame times and the TCP sequence numbers. Do the advertised windows change during the course of the connection?

5.51. Explain why framing information is required even in the case where frames are of constant length.

5.52. Perform the bit stuffing procedure for the following binary sequence: 1101111111011111110101.

5.53. Perform bit destuffing for the following sequence: 1110111110111111001111110.

5.54. Consider the PPP byte-stuffing method. What are the contents of the following received sequence of bytes after byte destuffing:
0x7D 0x5E 0xFE 0x24 0x7D 0x5D 0x7D 0x5D 0x62 0x7D 0x5E

5.55. Suppose that GFP operates over an error-free octet-synchronous physical layer. Find the average time the GFP receiver spends in the hunt state.

5.56. For GFP framing find the probability that the PLI and cHEC are not consistent. Assume that bit errors occur at random with probability p. Find the probability that the PLI and cHEC are not consistent in two consecutive frames.

5.57. What is the maximum efficiency of GFP with respect to header overhead?

5.58. Can GFP be used to carry video signals? If so, how would you handle the case where each video image produces a variable length of compressed information? How would you handle the case where each video frame produces a fixed length of compressed information?

5.59. Suppose a server has a 1 Gbps Ethernet link to a GFP transmitter. The GFP has an STS-1 SONET connection to another site. The GFP transmitter has a limited amount of buffering to accommodate frames prior to transmission. Explain how the GFP transmitter may use flow control on the Ethernet link to prevent buffer overflow. What type of flow control is preferred if the server and GFP transmitter are co-located? If the server and transmitter are 5 km apart?

5.60. An inverse multiplexer combines n digital transmission lines of bit rate R bps and makes them appear as a single transmission line of $n \times R$ bps. Consider an inverse multiplexer that combine multiple STS-1 SONET lines. Find an appropriate value of n to carry a 1 Gbps Ethernet stream. What is the improvement in efficiency relative to carrying the Ethernet stream on an OC-48 link?

5.61. Suppose that a 1-Megabyte message is sent over a serial link using TCP over IP over PPP. If the speed of the line is 56 kbps and the maximum PPP payload is 500 bytes, how long does it take to send the message?

5.62. Compare PPP and GFP. Identify situations where one is preferable to the other.

5.63. Use the Ethereal protocol analyzer tool to analyze the exchange of packets during PPP setup using a dialup connection to a local ISP. Start the Ethereal packet capture and then initiate the connection to the ISP. The number of captured packets will stop increasing after some point.

(a) Analyze the options fields of the packets during the LCP negotiations. You may need to consult RFC 1662 to interpret some of the packet contents.

(b) Examine the packet contents during the PAP exchange. Repeat the packet capture using CHAP.

(c) Analyze the IPCP packet exchange. What IP addresses are configured during the exchange? What other options are negotiated? You may wish to consult RFC 1332 for this part.

5.64. Explain the differences between PPP and HDLC.

5.65. A 1.5 Mbps communications link uses HDLC to transmit information to the moon. What is the smallest possible frame size that allows continuous transmission? The distance between earth and the moon is approximately 375,000 km, and the speed of light is 3×10^8 meters/second.

5.66. Suppose HDLC is used over a 1.5 Mbps geostationary satellite link. Suppose that 250-byte frames are used in the data link control. What is the maximum rate at which information can be transmitted over the link?

5.67. In HDLC how does a station know whether a received frame with the fifth bit set to 1 is a P or an F bit?

5.68. Which of the following statements about HDLC are incorrect?

(a) A transmitting station puts its own address in command frames.

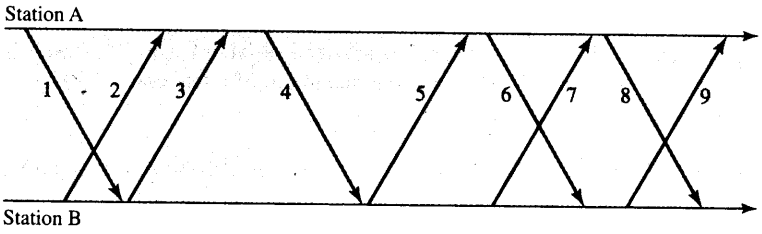
(b) A receiving station sees its own address in a response frame.

(c) A response frame contains the address of the sending station.

5.69. In HDLC suppose that a frame with P = 1 has been sent. Explain why the sender should not send another frame with P = 1. What should be done if the frame is lost during transmission?

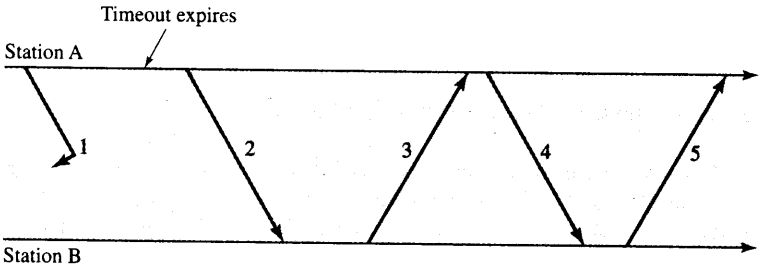
5.70. HDLC specifies that the N(R) in a SREJ frame requests the retransmission of frame N(R) and also acknowledges all frames up to N(R) − 1. Explain why only one SREJ frame can be outstanding at a given time.

5.71. The following corresponds to an HDLC ABM frame exchange with no errors.
(a) Complete the diagram by completing the labeling of the frame exchanges.
(b) Write the sequence of state variables at the two stations as each event takes place.



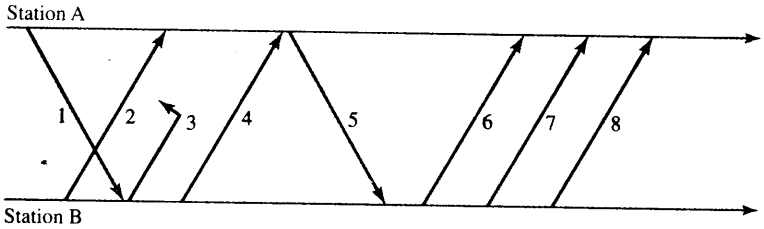
1. BI00 **2.** AI00 **3.** xIxx **4.** xIxx **5.** xRRxx **6.** xIxx **7.** xIxx **8.** xRRxx **9.** xRRxx

5.72. Assume station B is awaiting frame 2 from station A.
(a) Complete the diagram in HDLC ABM by completing the labeling of the frame exchanges.
(b) Write the sequence of state variables at the two stations as each event takes place.



1. BI23 **2.** xRRxP **3.** xRRxy **4.** xIxx **5.** xRRx

5.73. The following corresponds to an HDLC ABM frame exchange.
(a) Complete the diagram by completing the labeling of the frame exchanges.
(b) Write the sequence of state variables at the two stations as each event takes place.



1. BI00 **2.** AI00 **3.** xIxx **4.** xIxx **5.** xREJx **6.** xIyx **7.** xyxx **8.** xyxx

5.74. Suppose that packets arrive from various sources to a statistical multiplexer that transmits the packets over 64 kbps PPP link. Suppose that the PPP frames have lengths that follow

an exponential distribution with mean 1000 bytes and that the multiplexer can hold up to 100 packets at a time. Plot the average packet delay as a function of the packet arrival rate.

5.75. Suppose that the traffic that is directed to a statistical multiplexer is controlled so that ρ is always less than 80%. Suppose that packet arrivals are modeled by a Poisson process and that packet lengths are modeled by an exponential distribution. Find the minimum number of packet buffers required to attain a packet loss probability of 10^{-3} or less.

5.76. Suppose that packets arrive from various sources to a statistical multiplexer that transmits the packets over a 1 Mbps PPP link. Suppose that the PPP frames have a constant length of L bytes and that the multiplexer can hold a very large number of packets at a time. Assume that each PPP frame contains a PPP, IP, and TCP header in addition to the user data. Write a program or use a spreadsheet to plot the average packet delay as a function of the rate at which user information is transmitted for $L = 250$ bytes, 500 bytes, and 1000 bytes.

5.77. Suppose that a multiplexer receives constant-length packets from $N = 60$ data sources. Each data source has a probability $p = 0.1$ of having a packet in a given T-second period. Suppose that the multiplexer has one line in which it can transmit eight packets every T seconds. It also has a second line where it directs any packets that cannot be transmitted in the first line in a T-second period. Find the average number of packets that are transmitted in the first line and the average number of packets that are transmitted in the second line.

5.78. Discuss the relative importance of queueing delays in multiplexers that operate at bit rates of 1 Gbps or higher.

APPENDIX 5A: DERIVATION OF EFFICIENCY OF ARQ PROTOCOLS

In this appendix we derive the results that were presented in the chapter. To calculate the transmission efficiency for Stop-and-Wait ARQ, we need to calculate the average total time required to deliver a correct frame. Let n_t be the number of transmissions required to deliver a frame successfully; then $n_t = i$ transmissions are required if the first $i - 1$ transmissions are in error and the ith transmission is error free. Therefore,

$$P[n_t = i] = (1 - P_f)P_f^{i-1} \text{ for } i = 1, 2, 3 \ldots \tag{5A.1}$$

Each unsuccessful frame transmission will contribute a time-out period t_{out} to the total transmission time. The final successful transmission will contribute the basic delay t_0. The average total time to transmit a frame is therefore

$$\begin{aligned} E[t_{SW}] &= t_0 + \sum_{i=1}^{\infty} (i-1)t_{out}P[n_t = i] \\ &= t_0 + \sum_{i=1}^{\infty} (i-1)t_{out}(1 - P_f)P_f^{i-1} \\ &= t_0 + \frac{t_{out}P_f}{1 - P_f} \end{aligned} \tag{5A.2}$$

Equation (5A.2) takes into account the fact that in general the time-out period and the basic transmission time are different. To simplify the equation we will assume that $t_{out} = t_0$, then

$$E[t_{SW}] = \frac{t_0}{1 - P_f} \tag{5A.3}$$

Thus the effective transmission time for Stop-and-Wait ARQ is

$$R_{eff} = \frac{n_f - n_o}{E[t_{SW}]} = (1 - P_f)\frac{n_f - n_o}{t_0} = (1 - P_f)R^0_{eff}. \tag{5A.4}$$

and the associated transmission efficiency is

$$\eta_{SW} = \frac{\frac{n_f - n_o}{E[t_{SW}]}}{R} = (1 - P_f)\frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}} = (1 - P_f)\eta_0 \tag{5A.5}$$

In other words, the effect of errors in the transmission channel is to reduce the effective transmission rate and the efficiency by the factor $(1 - P_f)$.

Consider the basic Go-Back-N ARQ where the window size W_S has been selected so that the channel can be kept busy all the time. As before, let n_t be the number of transmissions required to deliver a frame successfully. If $n_t = i$, then as shown in Figure 5.16, $i - 1$ retransmissions of groups of W_S frames are followed by the single successful frame transmission. Therefore, the average total time required to deliver the frame is

$$\begin{aligned} E[t_{GBN}] &= t_f\left\{1 + W_S\sum_{i=1}^{\infty}(i-1)P[n_t = i]\right\} \\ &= t_f\left\{1 + W_S\sum_{i=1}^{\infty}(i-1)(1-P_f)P_f^{i-1}\right\} \\ &= t_f\left\{1 + W_S\frac{P_f}{1-P_f}\right\} = t_f\left\{\frac{1 + (W_S - 1)P_f}{1 - P_f}\right\} \end{aligned} \tag{5A.6}$$

Thus the effective transmission time for basic Go-Back-N ARQ is

$$R_{eff} = \frac{n_f - n_o}{E[t_{GBN}]} = (1 - P_f)\frac{n_f - n_o}{t_f\{1 + (W_S - 1)P_f\}} = (1 - P_f)\frac{1 - \frac{n_0}{n_f}}{1 + (W_S - 1)P_f}R \tag{5A.7}$$

and the associated transmission efficiency is

$$\eta_{GBN} = \frac{\frac{n_f - n_o}{E[t_{GBN}]}}{R} = (1 - P_f)\frac{1 - \frac{n_o}{n_f}}{1 + (W_S - 1)P_f} \tag{5A.8}$$

Finally, for Selective Repeat ARQ each transmission error involves only the retransmission of the specific frame. Therefore, the average time required to transmit a frame is

$$E[t_{SR}] = t_f\left\{1 + \sum_{i=1}^{\infty}(i-1)(1-P_f)P_f^{i-1}\right\}$$
$$= t_f\left\{1 + \frac{P_f}{1-P_f}\right\} = t_f\frac{1}{1-P_f} \tag{5A.9}$$

Thus the effective transmission time for Selective Repeat ARQ

$$R_{eff} = (1-P_f)\left(1 - \frac{n_0}{n_f}\right)R \tag{5A.10}$$

and the associated transmission efficiency is

$$\eta_{SR} = (1-P_f)\left(1 - \frac{n_o}{n_f}\right) \tag{5A.11}$$

CHAPTER 6

Medium Access Control Protocols and Local Area Networks

Switched networks provide interconnection between users by means of transmission lines, multiplexers, and switches. The transfer of information across such networks requires routing tables to direct the information from source to destination. To scale to a very large size, the addressing scheme in switched networks is typically hierarchical to help provide location information that assists the routing protocol in carrying out its task.[1] **Broadcast networks**, in contrast, are much simpler. Because all information is received by all users, routing is not necessary. A nonhierarchical addressing scheme is sufficient to indicate which user the information is destined to. However, broadcast networks require a medium access control protocol to orchestrate the transmissions from the various users. Local area networks (LANs), with their emphasis on low cost and simplicity, have been traditionally based on the broadcast approach. In this chapter we consider medium access control protocols and local area networks. We also consider the access methods used in cellular radio networks.

In broadcast networks a single transmission medium is shared by a community of users. For this reason, we also refer to these networks as **multiple access networks**. Typically, the information from a user is broadcast into the medium, and all the stations attached to the medium listen to all the transmissions. There is potential for user transmissions interfering or "colliding" with each other, and so a protocol has to be in place to prevent or minimize such interference. The role of the **medium access control (MAC)** protocols is to coordinate the access to the channel so that information gets through from a source to a destination in the same broadcast network.

It is instructive to compare MAC protocols with the peer-to-peer protocols discussed in Chapter 5. The basic role of both protocol classes is the same: to transfer blocks of user information despite transmission impairments. In the case of peer-to-peer protocols, the main concern is loss, delay, and resequencing of PDUs during transmission. In the case of MAC protocols, the main concern is interference from other users.

[1] Switched networks are discussed in Chapter 7.

The peer-to-peer protocols in Chapter 5 use sequence number information to detect and react to impairments that occur during transmission. We show that MAC protocols use a variety of mechanisms to prevent or to adapt to collisions in the medium. The peer-to-peer protocol entities exchange control information in the form of ACK PDUs to coordinate their actions. Some MAC entities also make use of mechanisms for explicitly exchanging information that can be used to coordinate access to the channel. For peer-to-peer protocols we found that the delay-bandwidth product of the channel was a key parameter that determines system performance. The delay-bandwidth product plays the same fundamental role in medium access control. Finally, we note one basic difference between peer-to-peer protocols and MAC protocols: The former are involved with the interaction of only two peer processes, whereas the latter require the coordinated action from *all* of the MAC protocol entities within the same broadcast network. The lack of cooperation from a single MAC entity can prevent any communication from taking place over the shared medium.

The chapter is organized as follows.

Part I:

1. *Introduction to multiple access communications.* We discuss the need for multiple access in the context of wireline as well as wireless networks and indicate the general approaches that are used to address the sharing of the medium.
2. *Random access.* First, we consider MAC protocols that involve the "random" transmission of user frames into a shared medium. We begin with the seminal ALOHA protocol and proceed to the Carrier-Sense Multiple Access with Collision Detection (CSMA-CD) protocol, which forms the basis for the Ethernet LAN standard. We show that the delay-bandwidth product has a dramatic impact on protocol performance.
3. *Scheduling.* We consider MAC protocols that use scheduling to coordinate the access to the shared medium. We present a detailed discussion of ring-topology networks that make use of these protocols. We again show the impact of delay-bandwidth product on performance.
4. *Channelization.* Many shared medium networks operate through the assignment of channels to users. We discuss three approaches to the creation of such channels: frequency-division multiple access, time-division multiple access, and code-division multiple access. We use various cellular telephone network standards to illustrate the application of these techniques.
5. *Delay performance.* In this optional section we present models for assessing the frame transfer delay performance of random access and scheduling MAC protocols as well as channelization schemes.

Part II:

6. *Overview of LANs.* We discuss the structure of the frames[2] that are used in LANs, and we discuss the placement of LAN protocols in the OSI reference model.

[2]Because LAN protocols reside in the data link layer of the OSI reference model, the PDU is usually referred to as frame rather than packet.

7. *LAN standards.* We discuss several important LAN standards, including the IEEE 802.3 Ethernet LAN, the IEEE 802.5 token-ring LAN, the FDDI LAN, and the IEEE 802.11 wireless LAN. The MAC protocols associated with each LAN standard are also described.
8. *LAN bridges.* A LAN is limited in the number of stations that it can handle, so multiple LANs are typically deployed to handle a large number of stations with each LAN serving a reasonable number of stations. Devices to interconnect LANs become necessary to enable communications between users in different LANs. Bridges provide one approach for the interconnection of LANs and form the basis for current large-scale Ethernet networks.

PART I: Medium Access Control Protocols

6.1 MULTIPLE ACCESS COMMUNICATIONS

Figure 6.1 shows a generic multiple access communications situation in which a number of user stations share a transmission medium. M denotes the number of stations. The transmission medium is broadcast in nature, and so all the other stations that are attached to the medium can hear the transmission from any given station. When two or more stations transmit simultaneously, their signals will collide and interfere with each other.

There are two broad categories of schemes for sharing a transmission medium. The first category involves a static and collision-free sharing of the medium. We refer to these as **channelization schemes** because they involve the partitioning of the medium into separate channels that are then dedicated to particular users. Channelization techniques are suitable when stations generate a steady stream of information that makes efficient use of the dedicated channel. The second category involves a dynamic sharing of the medium on a per frame basis that is better matched to situations in which the user traffic is bursty. We refer to this category as **MAC schemes**. The primary function of medium access control is to minimize or eliminate the incidence of collisions to achieve a reasonable utilization of the medium. The two basic approaches to medium access control are random access and scheduling. Figure 6.2 summarizes the various approaches to sharing a transmission medium.

Networks based on radio communication provide examples where a medium is shared. Typically, several stations share two frequency bands, one for transmitting and

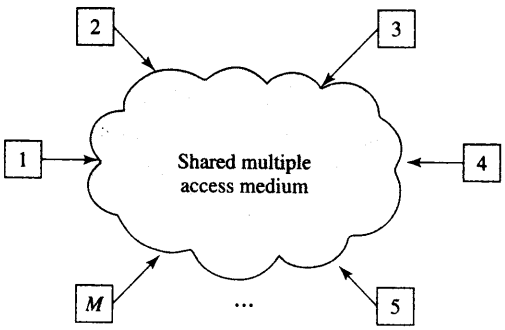


FIGURE 6.1 Multiple access communications.

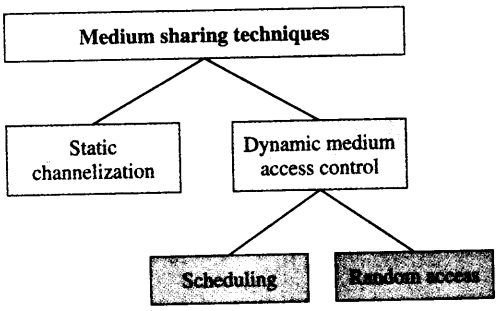


FIGURE 6.2 Approaches to sharing a transmission medium.

one for receiving. For example, in satellite communications each station is assigned a channel in an uplink frequency band that it uses to transmit to the satellite. As shown in Figure 6.3 the satellite is simply a repeater that takes the uplink signals that it receives from many earth stations and broadcasts them back on channels that occupy a different downlink frequency band. Cellular telephony is another example involving radio communications. Again we have two frequency bands shared by a set of mobile users. In Chapter 4 we showed that a cellular telephone connection involves the assignment of an inbound channel from the first band and an outbound channel from the second band to each mobile user. Later in this chapter we present different approaches to the creation of such channels.

Channel sharing techniques are also used in wired communications. For example, multidrop lines were used in early data networks to connect a number of terminals to a central host computer, as shown in Figure 6.4. The set of M stations shares an inbound and outbound transmission line. The host computer broadcasts information to the users on the outbound line. The stations transmit information to the host using the inbound line. Here there is a potential for interference on the inbound line. In the MAC protocol developed for this system, the host computer issues polling messages to each terminal, providing it with permission to transmit on the inbound line.

Ring networks also involve the sharing of a medium. Here a number of ring adapter interfaces are interconnected in a ring topology by point-to-point transmission lines, as shown in Figure 6.5a. Typically, information frames are transmitted in the ring in cut-through fashion, with only a few bits delay per ring adapter. All stations can monitor the passing signal and extract frames intended for them. A MAC protocol is required

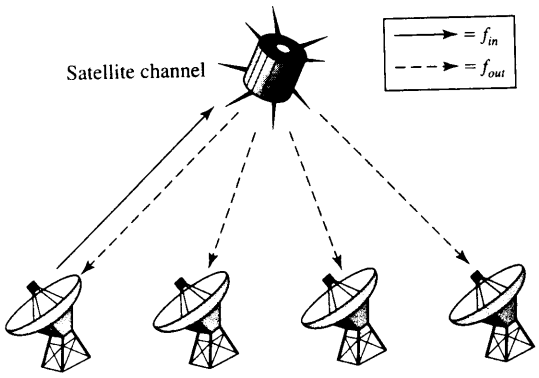


FIGURE 6.3 Satellite communication involves sharing of uplink and downlink frequency bands.

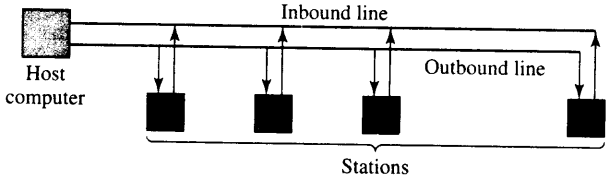


FIGURE 6.4 Multidrop line requires access control.

here to orchestrate the insertion and removal of frames from the shared ring, since obviously only one signal can occupy a particular part of the ring.

Shared buses are another example of shared broadcast media (Figure 6.5b). For example, in coaxial cable transmission systems users can inject a signal that propagates in both directions along the medium, eventually reaching all stations connected to the cable. All stations can listen to the medium and extract transmissions intended for them. If two or more stations transmit simultaneously, the signal in the medium will be garbled, so again a procedure is needed to coordinate access to the medium. In Section 6.7.3 we show that hub topology networks lead to a situation identical to that of shared buses.

Finally, we return to a more current example involving wireless communications, shown in Figure 6.6. Here a set of devices such as workstations, laptop computers, cordless telephones, and other communicating appliances share a wireless medium, for example, 2.4 or 5 GHz radio or infrared light. These devices could be transmitting short messages, real-time voice, or video information, or they could be accessing web pages. Again a MAC protocol is required to share the medium. In this example the diversity of the applications requires the medium access control to also provide some degree of quality-of-service (QoS) guarantees, for example, low delay for real-time voice traffic.

In the examples discussed so far, we can discern two basic approaches: centralized and distributed. In the first approach communications between the M stations makes use of *two* channels. A base station or controller communicates to all the other stations by

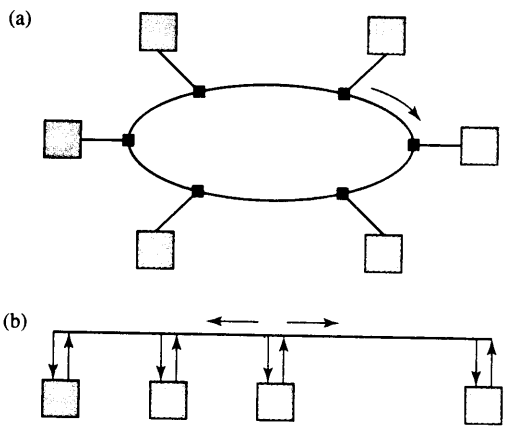


FIGURE 6.5 (a) Ring networks and (b) multitapped buses require MAC.

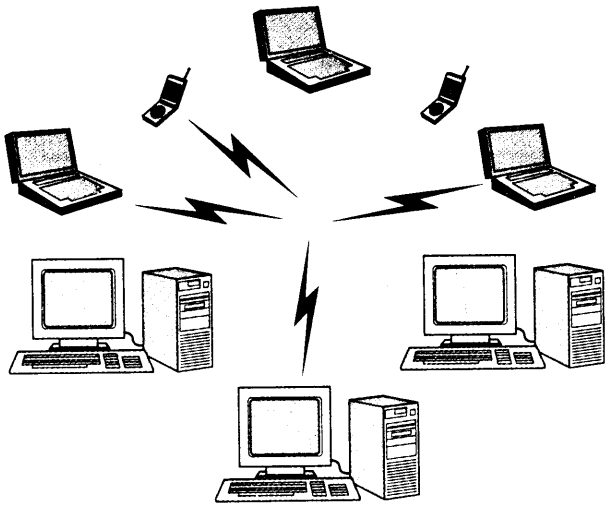


FIGURE 6.6 Wireless LAN.

broadcasting over the "outbound," "forward," or "downstream" channel. The stations in turn share an "inbound," "reverse," or "upstream" channel in the direction of the base station. In the preceding examples the satellite system, cellular telephone systems, and the multidrop line take this approach. It is also the approach in cable modem access systems and certain types of wireless data access systems. In this approach the base station or controller can send coordination information to the other stations to orchestrate the access to the shared reverse channel. The second approach to providing communications uses direct communications between all M stations. This is the case for ad hoc wireless networks as well as for the multitapped bus. Note that the ring network has features of both approaches. On the one hand, ring networks involve all stations communicating directly with other stations; on the other hand, each time a station has the token that entitles it to transmit, it can be viewed as acting as a temporary controller that directs how traffic enters the ring at that moment.

The preceding examples have the common feature that the shared medium is the *only* means available for the stations to communicate with each other. Thus any explicit or implicit coordination in accessing the channel must be done through the channel itself. This situation implies that some of the transmission resource will be utilized implicitly or explicitly to transfer coordination information. We saw in Chapter 5 that the *delay-bandwidth product* plays a key role in the performance of ARQ protocols. The following example indicates how the delay-bandwidth product affects performance in medium access control.

EXAMPLE **Delay-Bandwidth Product and MAC Performance**

Consider the situation shown in Figure 6.7 in which two stations are trying to share a common medium. Let's develop an access protocol for this system. Suppose that when a station has a frame to send, the station first listens to the channel to see whether it is

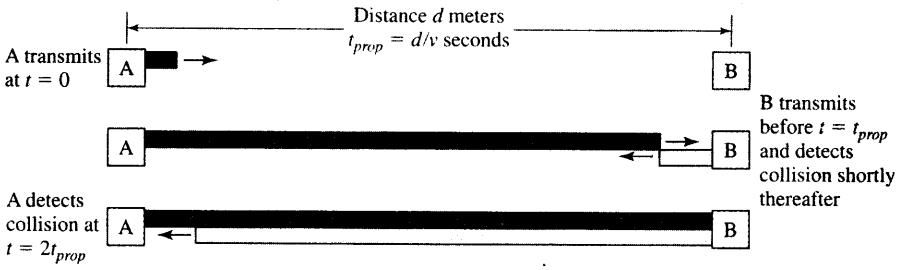


FIGURE 6.7 Channel capture and delay-bandwidth product where v is the speed of light in the medium, typically 2 to 3 $\times$ 10^8 meters/second.

busy with a transmission from the other station. If it is not busy, then the station begins to transmit, but it continues observing the signal in the channel to make sure that its signal is not corrupted by a signal from the other station. The signal from station A does not reach station B until time t_{prop}. If station B has not begun a transmission by that time, then station A is assured that station B will refrain from transmitting thereafter and so station A has captured the channel and its entire message will get through.

Figure 6.7 shows what happens when a collision of frame transmissions takes place. In this case station B must have begun its transmission sometime between times $t = 0$ and $t = t_{prop}$. By time $t = 2t_{prop}$, at the latest, station A will find out about the collision. At this point both stations are aware that they are competing for the channel. Some mechanism for resolving this contention is required. We will suppose for simplicity that both stations know the value of the propagation delay t_{prop} and that they measure the time from when they began transmitting to when a collision occurs. The station that began transmitting earlier (which measures the aforementioned time to be greater than $t_{prop}/2$) is declared to be the "winner" and proceeds to retransmit its frame as soon as the channel goes quiet. The "losing" station defers and remains quiet until the frame transmission from the other station is complete. For the sake of fairness, we suppose that the winning station is compelled to remain quiet for time $2t_{prop}$ after it has completed its frame transmission. This interval gives the losing station the opportunity to capture the channel and transmit its frame.

Thus we see for this example that a time approximately equal to $2t_{prop}$ is required to coordinate the access for each frame transmitted. Let L be the number of bits in a frame. The sending station then requires $X = L/R$ seconds to transmit the frame, where R is the transmission bit rate. Therefore a frame transmission requires $L/R + 2t_{prop}$ seconds. The **throughput** of a system is defined as the actual rate at which information is sent over the channel, and is measured in bits/second or frames/second. The *maximum throughput* in this example in bps is:

$$R_{eff} = \frac{L}{L/R + 2t_{prop}} = \frac{1}{1 + \frac{2t_{prop}R}{L}} R = \frac{1}{1 + 2a} R \qquad (6.1a)$$

and the normalized *maximum throughput* or **efficiency** is given by

$$\rho_{max} = R_{eff}/R = \frac{1}{1+2a} \tag{6.1b}$$

where the **normalized delay-bandwidth product** a is defined as the ratio of the one-way delay-bandwidth product to the average frame length

$$a = \frac{t_{prop}R}{L} = \frac{t_{prop}}{L/R} = \frac{t_{prop}}{X} \tag{6.2}$$

When a is much smaller than 1, the medium can be used very efficiently by using the above protocol. For example, if $a = 0.01$, then the efficiency is $1/1.02 = 0.98$. As a becomes larger, the channel becomes more inefficient. For example if $a = 0.5$, then the efficiency is $1/2 = 0.50$.

The efficiency result in the preceding example is typical of MAC protocols. Later in the chapter we show that the CSMA-CD protocol that originally formed the basis for the Ethernet LAN standard has an efficiency or maximum normalized throughput of approximately $1/(1+6.44a)$. We also show that for token-ring networks, such as in the IEEE 802.5 standard, the maximum efficiency is approximately given by $1/(1+a')$ where a' is the ratio of the latency of the ring in bits to the average frame length. The ring latency has two components: The first component is the sum of the bit delays introduced at every ring adapter; the second is the delay-bandwidth product where the delay is the time required for a bit to circulate around the ring. In both of these important LANs, we see that a, the relative value of the delay-bandwidth product to the average frame length is a key parameter in system performance.

Table 6.1 shows the number of bits in transit in a one-way propagation delay for various distances and transmission speeds. Possible network types range from a "desk area network" with a diameter of 1 meter to a global network with a diameter of 100,000 km (about two times around the earth). It can be seen that the reaction time as measured by the number of bits in transit can become quite large for high transmission speeds and for long distances. To estimate some values of the key parameter a, we note that Ethernet frames have a *maximum* length of approximately 1500 bytes = 12,000 bits. TCP segments have a maximum length of approximately 65,000 bytes = 520,000 bits. For desk area and local area networks, we see that these frame lengths can provide acceptable values of a. For higher speeds or longer distances, the frame sizes are not

TABLE 6.1 Number of bits in transit in one-way propagation delay assuming propagation speed of 3×10^8 meters/second.

Distance	10 Mbps	100 Mbps	1 Gbps	Network type
1 m	3.33×10^{-02}	3.33×10^{-01}	3.33×10^{0}	Desk area network
100 m	3.33×10^{0}	3.33×10^{01}	3.33×10^{02}	Local area network
10 km	3.33×10^{02}	3.33×10^{03}	3.33×10^{04}	Metropolitan area network
1000 km	3.33×10^{04}	3.33×10^{05}	3.33×10^{06}	Wide area network
100000 km	3.33×10^{06}	3.33×10^{07}	3.33×10^{08}	Global area network

long enough to overcome the large delay-bandwidth products. Consequently, we find that networks based on broadcast techniques are used primarily in LANs and other networks with small delay-bandwidth products.

The selection of a MAC protocol for a given situation depends on delay-bandwidth product and throughput efficiency, as well as other factors. The *transfer delay* experienced by frames is an important performance measure. The **frame transfer delay** T is defined as the time that elapses from when the first bit of the frame arrives at the source MAC to when the last bit of the frame is delivered to the destination MAC. *Fairness* in the sense of giving stations equitable treatment is also an issue. The concern here is that stations receive an appropriate proportion of bandwidth and that their frames experience an appropriate transfer delay. Note that fairness does not necessarily mean equal treatment; it means providing stations with treatment that is consistent with policies set by the network adminstrator. *Reliability* in terms of robustness with respect to failure and faults in equipment is important as it affects the availability of service. An issue that is becoming increasingly important is the capability to carry *different types of traffic,* that is, stream versus bursty traffic, as well as the capability to provide *quality-of-service* guarantees to different traffic types. *Scalability* with respect to number of stations and the user bandwidth requirements are also important in certain settings. And of course *cost* is always an issue.

Suppose that the M stations in the system generate frames at an aggregate rate of λ frames/second. If frames have a length of L bits, then the average bit rate generated by all the stations is λL bits/second. The *normalized throughput* or **load** is defined as $\rho = \lambda L/R$. Note that the bit rate generated by all the stations cannot be greater than R, so $\lambda L < R$, which implies that $\rho = \lambda L/R < 1$. In general, every MAC protocol has a maximum throughput less than R/L frames/second, since some channel time is wasted in collisions or in sending coordination information. Consequently, the load ρ cannot exceed some maximum normalized throughput value $\rho_{max} < 1$.

The frame transfer delay and the throughput of a MAC protocol are interrelated. Figure 6.8 shows a typical graph of how the average transfer delay increases with throughput. The abscissa (x-axis) ρ represents the load generated by all the stations.

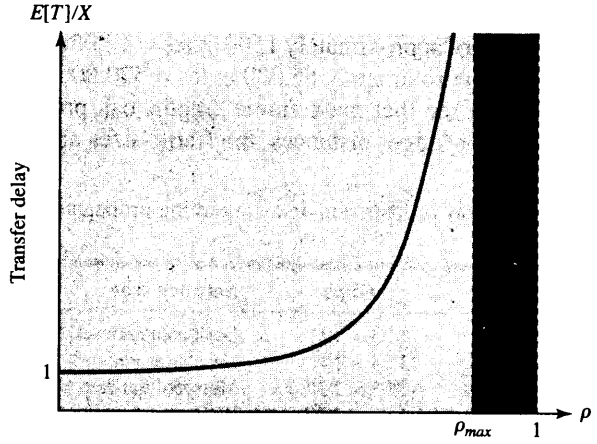


FIGURE 6.8 Typical frame transfer delay versus load (normalized throughput).

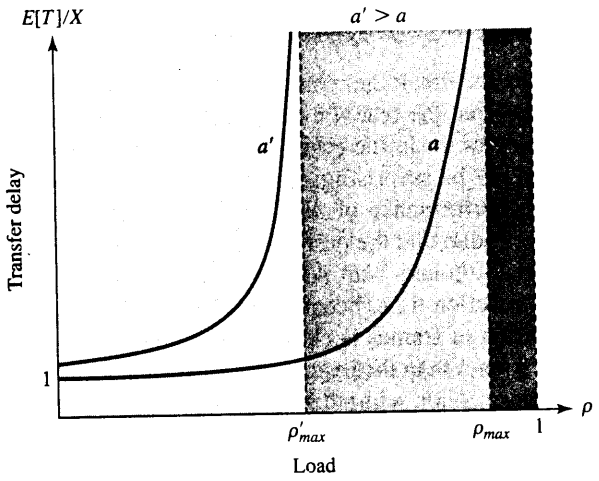


FIGURE 6.9 Dependence of delay-throughput performance on $a = Rt_{prop}/L$.

The ordinate (y-axis) gives the *average frame transfer delay* $E[T]$ in multiples of a single average *frame transmission time* $X = L/R$ seconds. When the load ρ is small, the stations transmit infrequently. Because there is likely to be little or no contention for the channel, the transfer delay is close to one frame transmission time, as shown in the figure. As the load increases, there is more contention for the channel, and so frames have to wait increasingly longer times before they are transferred. Finally, the rate of frame arrivals from all the stations reaches a point that exceeds the rate that the channel can sustain. At this point the buffers in the stations build up with a backlog of frames, and if the arrivals remain unchecked, the transfer delays grow without bound as shown in the figure.

Note from Figure 6.8 that the *maximum achievable throughput* ρ_{max} is usually less than 1. Figure 6.9 shows that the transfer delay versus load curve varies with parameter a, which is defined as the ratio of the one-way delay-bandwidth product to the average frame length. When a is small, the relative cost of coordination is low and ρ_{max} can be close to 1. However, when a increases, the relative cost of coordination becomes high and ρ_{max} can be significantly less than 1. Figure 6.9 is typical of the delay-throughput performance of medium access controls. The exact shape of the curve depends on the particular medium and MAC protocol, the number of stations, the arrival pattern of frames at the stations, and the distribution of lengths of the frames. Our discussion on medium access controls will focus on their maximum achievable throughput. The transfer delay performance is discussed in the optional Section 6.5.

6.2 RANDOM ACCESS

In this and the next section we consider the two main classes of MAC protocols: random access and scheduling. In particular we investigate the system parameters that affect MAC performance. An understanding of these issues is required in the design and

selection of MAC protocols, especially in situations that involve large delay-bandwidth products.

Random access methods constitute the first major class of MAC procedures. In the beginning of this chapter, we indicated that the reaction time in the form of propagation delay and network latency is a key factor in the effectiveness of various medium access techniques. This factor should not be surprising, as we have already observed the influence of reaction time on the performance of ARQ retransmission schemes. In the case of Stop-and-Wait ARQ, we found that the performance was good as long as the reaction time was small. However, Stop-and-Wait ARQ became ineffective when the propagation delay, and hence the reaction time, became very large. The weakness of Stop-and-Wait is that the transmission of frames is closely tied to the return of acknowledgments, which cannot occur sooner than the reaction time. The solution to this problem involved proceeding with transmission without waiting for acknowledgments and dealing with transmission errors after the fact.

A similar situation arises in the case of scheduling approaches to medium access control that are considered in Section 6.3. When the reaction time is small, the scheduling can be done quickly, based on information that is current. However, when the reaction time becomes very large, by the time the scheduling takes effect the state of the system might have changed considerably. The experience with ARQ systems suggests an approach that involves proceeding with transmissions without scheduling. It also suggests dealing with collisions after the fact. The class of random access MAC procedures discussed in this section takes this approach.

6.2.1 ALOHA

As the name suggests, the ALOHA random access scheme had its origins in the Hawaiian Islands. The University of Hawaii needed a means to interconnect terminals at campuses located on different islands to the host computer on the main campus. The solution is brilliant in its simplicity. A radio transmitter is attached to the terminals, and messages are transmitted as soon as they become available, thus producing the smallest possible delay. From time to time frame transmissions will collide, but these can be treated as transmission errors, and recovery can take place by retransmission. When traffic is very light, the probability of collision is very small, and so retransmissions need to be carried out infrequently. Consequently, under light traffic the frame transfer delay will be low.

There is a significant difference between normal transmission errors and those that are due to frame collisions. Transmission errors that are due to noise affect only a single station. On the other hand, in the frame collisions more than one station is involved, and hence more than one retransmission is necessary. This interaction by several stations produces a positive feedback that can trigger additional collisions. For example, if the stations use the same time-out values and schedule their retransmissions in the same way, then their future retransmissions will also collide. For this reason, the ALOHA scheme requires stations to use a **backoff algorithm**, which typically chooses a random number in a certain retransmission time interval. This randomization is intended to spread out the retransmissions and reduce the likelihood of additional

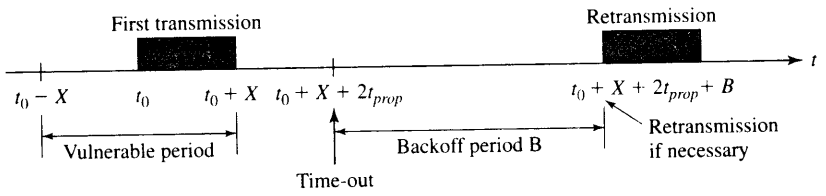


FIGURE 6.10 ALOHA random access scheme.

collisions between the stations. Nevertheless, the likelihood of collisions increases after each frame collision.

Figure 6.10 shows the basic operation of the ALOHA scheme. It can be seen that the first transmission is done without any scheduling. Information about the outcome of the transmission is obtained at the earliest after the reaction time $2t_{prop}$, where t_{prop} is the maximum one-way propagation time between two stations. If no acknowledgment is received after a time-out period, a backoff algorithm is used to select a random retransmission time.

It is clear that in the ALOHA scheme the network can swing between two modes. In the first mode frame transmissions from the station traverse the network successfully on the first try and collide only from time to time. The second mode is entered through a snowball effect that occurs when there is a surge of collisions. The increased number of backlogged stations, that is, stations waiting to retransmit a message, increases the likelihood of additional collisions. This situation leads to a further increase in the number of backlogged stations, and so forth.

Abramson provided an approximate analysis of the ALOHA system that gives an insight into its behavior. Assume that frames are a constant length L and constant transmission time $X = L/R$. Consider a reference frame that is transmitted starting at time t_0 and completed at time $t_0 + X$, as shown in Figure 6.10. This frame will be successfully transmitted if no other frame collides with it. Any frame that begins its transmission in the interval t_0 to $t_0 + X$ will collide with the reference frame. Furthermore, any frame that begins its transmission in the prior X seconds will also collide with the reference frame. Thus the probability of a successful transmission is the probability that there are no additional frame transmissions in the **vulnerable period** $t_0 - X$ to $t_0 + X$.

Abramson used the following approach to find the probability that there is no collision with the reference frame. Let S be the *arrival rate of new frames* to the system in units of frames/X seconds. We assume that all frames eventually make it through the system, so S also represents the throughput of the system. Now the actual arrival rate to the system consists of new arrivals and retransmissions. Let G be the *total arrival rate* in units of frames/X seconds. G is also called the total load. The probability of transmissions from new frames and retransmitted frames is not straightforward to calculate. Abramson made the key simplifying assumption that the backoff algorithm spreads the retransmissions so that frame transmissions, new and repeated, are equally likely to occur at any instant in time. This implies that the number of frames transmitted in a time interval has a Poisson distribution with average number of arrivals of $2G$ arrivals/$2X$ seconds, that is:

$$P[k \text{ transmissions in } 2X \text{ seconds}] = \frac{(2G)^k}{k} e^{-2G}, \quad k = 0, 1, 2, \ldots \tag{6.3}$$

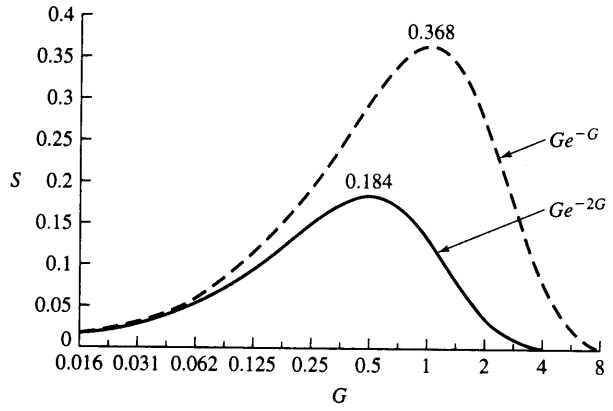


FIGURE 6.11 Throughput S versus load G.

The throughput S is equal to the total arrival rate G times the probability of a successful transmission, that is:

$$\begin{aligned} S &= GP[\text{no collision}] = GP[0 \text{ transmissions in } 2X \text{ seconds}] \\ &= G\frac{(2G)^0}{0!}e^{-2G} \\ &= Ge^{-2G} \end{aligned} \tag{6.4}$$

Figure 6.11 shows a graph of S versus G. Starting with small G, we see that S increases, reaches a peak value of $1/2e$ at $G = 0.5$, and then declines back toward 0. For a given value of S, say, $S = 0.05$, there are two associated values of G. This is in agreement with our intuition that the system has two modes: one associating a small value of G with S, that is, $S \approx G$, and another associating a large value of G with S, that is, $G \gg S$ when many stations are backlogged. The graph also shows that values of S beyond $1/2e$ are not attainable. This condition implies that the ALOHA system cannot achieve throughputs higher than $1/2e = 18.4$ percent. Note that the maximum value of S occurs at $G = 1/2$, which corresponds to a total arrival rate of exactly one frame per vulnerable period. This makes sense since two or more arrivals in a vulnerable period result in a collision.

6.2.2 Slotted ALOHA

The performance of the ALOHA scheme can be improved by reducing the probability of collisions. The slotted ALOHA scheme shown in Figure 6.12 reduces collisions by constraining the stations to transmit in synchronized fashion. All the stations keep track of transmission time slots and are allowed to initiate transmissions only at the beginning of a time slot. Frames are assumed to be constant and to occupy one time slot. The frames that can collide with our reference frame must now arrive in the period $t_0 - X$ to t_0. The vulnerable period in the system is now X seconds long.

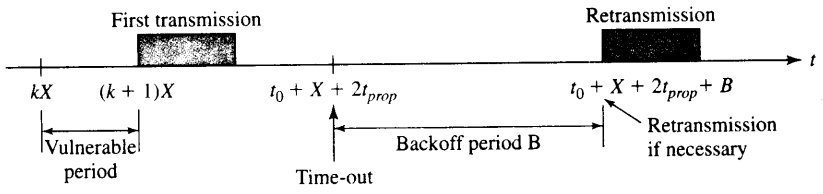


FIGURE 6.12 Slotted ALOHA random access scheme ($t_0 = (k + 1)X$).

Proceeding as before, we now have G arrivals in X seconds and we use the Poisson distribution for number of events in X seconds to obtain

$$\begin{aligned} S &= GP[\text{no collision}] = GP[0 \text{ transmissions in } X \text{ seconds}] \\ &= G\frac{(G)^0}{0!}e^{-G} \\ &= Ge^{-G} \end{aligned} \tag{6.5}$$

Figure 6.11 shows the behavior of the throughput S versus load G for the slotted ALOHA system. The system still exhibits its bimodal behavior and has a maximum throughput of $1/e = 36.8$ percent at $G = 1$.

The ALOHA and the slotted ALOHA systems show how low-delay frame transmission is possible using essentially uncoordinated access to a medium. However, this result is at the expense of significant wastage due to collisions, which limits the maximum achievable throughput to low values. However, unlike the other MAC schemes discussed in this chapter, the maximum throughput of the ALOHA schemes is not sensitive to the reaction time because stations act independently.

EXAMPLE **ALOHA and Slotted ALOHA**

Suppose that a radio system uses a 9600 bps channel for sending call setup request messages to a base station. Suppose that frames are 120 bits long. What is the maximum throughput possible with ALOHA and with slotted ALOHA?

The system transmits frames at a rate of 9600 bits/second × 1 frame/120 bits = 80 frames/second. The maximum throughput for ALOHA is then 80(0.184) ≈ 15 frames/second. The maximum throughput for slotted ALOHA is then ≈ 30 frames/second.

6.2.3 Carrier Sense Multiple Access

The low maximum throughput of the ALOHA schemes is due to the wastage of transmission bandwidth because of frame collisions. This wastage can be reduced by avoiding transmissions that are certain to cause collisions. By sensing the medium for the presence of a carrier signal from other stations, a station can determine whether there is an ongoing transmission. The class of **carrier sensing multiple access (CSMA)** MAC schemes uses this strategy.

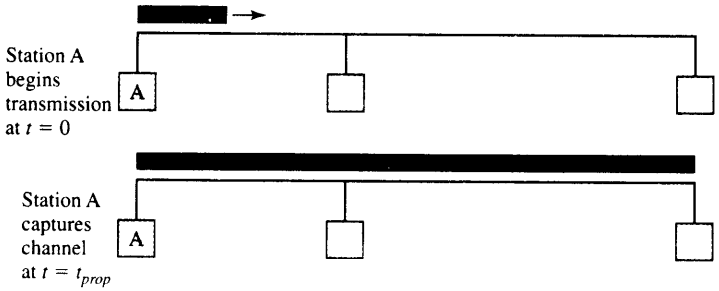


FIGURE 6.13 CSMA random access scheme.

In Figure 6.13 we show how the vulnerable period is determined in a CSMA system. At time $t = 0$, station A begins transmission at one extreme end of a broadcast medium. As the signal propagates through the medium, stations become aware of the transmission from station A. At time $t = t_{prop}$, the transmission from station A reaches the other end of the medium. By this time all stations are aware of the transmission from station A. Thus the vulnerable period consists of one propagation delay, and if no other station initiates a transmission during this period, station A will in effect capture the channel because no other stations will transmit thereafter.

CSMA schemes differ according to the behavior of stations that have a frame to transmit when the channel is busy. In **1-Persistent CSMA**, stations with a frame to transmit sense the channel. If the channel is busy, they sense the channel continuously, waiting until the channel becomes idle. As soon as the channel is sensed idle, they transmit their frames. Consequently if more than one station is waiting, a collision will occur. In addition, stations that have a frame arrive within t_{prop} of the end of the preceding transmission will also transmit and possibly be involved in a collision. Stations that are involved in a collision perform the backoff algorithm to schedule a future time for resensing the channel. In a sense, in 1-Persistent CSMA stations act in a "greedy" fashion, attempting to access the medium as soon as possible. As a result, 1-Persistent CSMA has a relatively high collision rate.

Non-Persistent CSMA attempts to reduce the incidence of collisions. Stations with a frame to transmit sense the channel. If the channel is busy, the stations immediately run the backoff algorithm and reschedule a future resensing time. If the channel is idle, the stations transmit. By immediately rescheduling a resensing time and not persisting, the incidence of collisions is reduced relative to 1-Persistent CSMA. This immediate rescheduling also results in longer delays than are found in 1-Persistent CSMA.

The class of **p-Persistent CSMA** schemes combines elements of the above two schemes. Stations with a frame to transmit sense the channel, and if the channel is busy, they persist with sensing until the channel becomes idle, as shown in Figure 6.14. If the channel is idle, the following occurs: with probability p, the station transmits its

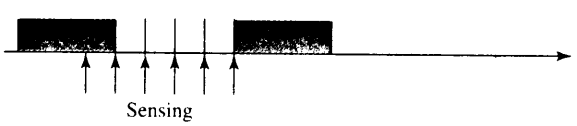


FIGURE 6.14 p-Persistent CSMA random access scheme.

frame; with probability $1 - p$ the station decides to wait an additional propagation delay t_{prop} before again sensing the channel. This behavior is intended to spread out the transmission attempts by the stations that have been waiting for a transmission to be completed and hence to increase the likelihood that a waiting station successfully seizes the medium.

All of the variations of CSMA are sensitive to the end-to-end propagation delay of the medium that constitutes the vulnerable period. An analysis of the throughput S versus load G for CSMA is beyond the scope of this text. Figure 6.15 shows the throughput S versus load G for 1-Persistent and Non-Persistent CSMA for three values

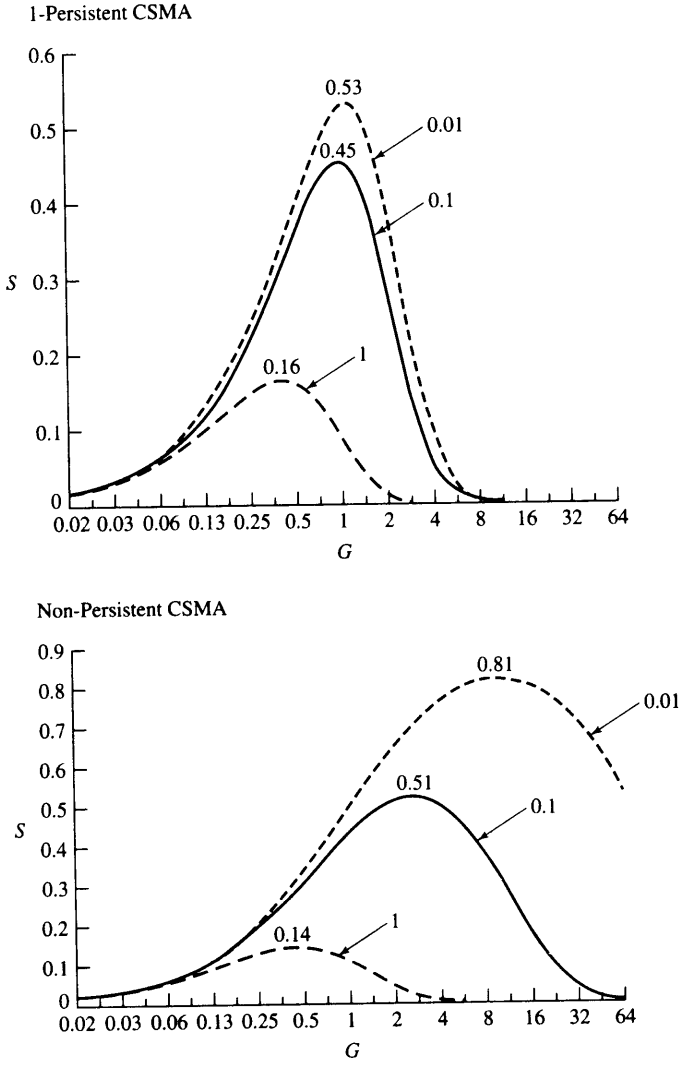


FIGURE 6.15 Throughput S versus load G for 1-Persistent and Non-Persistent CSMA. The curves are for different values of a.

of a. It can be seen that the throughput of 1-Persistent CSMA drops off much more sharply with increased G. It can also be seen that the normalized propagation delay $a = t_{prop}/X$ has a significant impact on the maximum achievable throughput. Non-Persistent CSMA achieves a higher throughput than 1-Persistent CSMA does over a broader range of load values G. For very small values of a, Non-Persistent CSMA has a relatively high maximum achievable throughput. However, as a approaches 1, both 1-Persistent and Non-Persistent CSMA have maximum achievable throughputs that are even lower than the ALOHA schemes.

6.2.4 Carrier Sense Multiple Access with Collision Detection

The CSMA schemes improve over the ALOHA schemes by reducing the vulnerable period from one- or two-frame transmission times to a single propagation delay t_{prop}. In both ALOHA and CSMA schemes, collisions involve entire frame transmissions. If a station can determine whether a collision is taking place, then the amount of wasted bandwidth can be reduced by aborting the transmission when a collision is detected. The **carrier sensing multiple access with collision detection (CSMA-CD)** schemes use this approach.

Figure 6.16 shows the basic operation of CSMA-CD. At time $t = 0$, station A at one extreme end of the network begins transmitting a frame. This frame transmission reaches station B at another extreme end of the network, at time t_{prop}. If no other station initiates a transmission in this time period, then station A will have captured the channel. However, suppose that station B initiates a transmission just before the transmission arrival from station A. Station A will not become aware of the collision until time $= 2t_{prop}$. Therefore, station A requires $2t_{prop}$ seconds to find out whether it has successfully captured the channel.

In CSMA-CD a station with a frame first senses the channel and transmits if the channel is idle. If the channel is busy, the station uses one of the possible strategies from CSMA; that is, the station can persist, backoff immediately, or persist and attempt transmission with probability p. If a collision is detected during transmission, then a short jamming signal is transmitted to ensure that other stations know that collision has occurred before aborting the transmission, and the backoff algorithm is used to schedule a future resensing time.

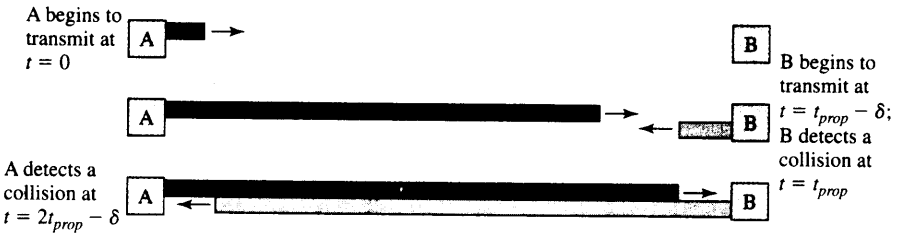


FIGURE 6.16 In CSMA-CD it takes $2t_{prop}$ (the reaction time) to find out whether the channel has been captured.

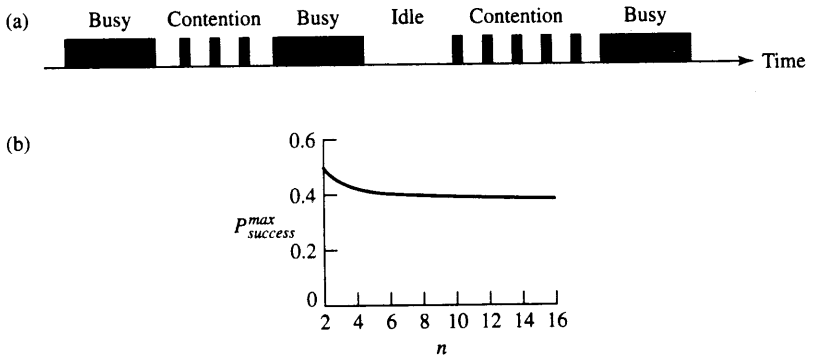


FIGURE 6.17 Frame transmission times and contention periods.

As shown in Figure 6.17a, the channel can be in three states: busy transmitting a frame, idle, or in a contention period where stations attempt to capture the channel. The throughput performance of 1-persistent CSMA-CD can be analyzed by assuming that time is divided into minislots of length $2t_{prop}$ seconds to ensure that stations can always detect a collision. Each time the channel becomes idle, stations contend for the channel by transmitting and listening to the channel to see if they have successfully captured the channel. Each such contention interval takes $2t_{prop}$ seconds. Next we calculate the mean time required for a station to successfully capture the channel.

Suppose that n stations are contending for the channel and suppose that each station transmits during a contention minislot with probability p. The probability of a successful transmission is given by the probability that only one station transmits:

$$P_{success} = np(1-p)^{n-1} \tag{6.6}$$

since the probability that a given station transmits and $n-1$ do not is given by $p(1-p)^{n-1}$ and there are n possible ways in which to select the one station that transmits. To find the maximum achievable throughput, assume that stations use the value of p that maximizes $P_{success}$. By taking a derivative of $P_{success}$ with respect to p and setting it to zero, we find that the probability is maximized when $p = 1/n$. The maximum probability of success is then

$$P^{max}_{success} = n\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} = \left(1-\frac{1}{n}\right)^{n-1} \rightarrow \frac{1}{e} \tag{6.7}$$

Figure 6.17b shows that the maximum probability of success rapidly approaches $1/e$ as n increases.

If the probability of success in one minislot is $P^{max}_{success}$, then the average number of minislots that elapse until a station successfully captures the channel is $1/P^{max}_{success}$. Assuming a large value of n, we have that $P^{max}_{success} = 1/e$, and so the average number of minislots until a station successfully captures the channel is:

$$\frac{1}{P^{max}_{success}} = e = 2.718 \text{ minislots} \tag{6.8}$$

The maximum throughput in the CSMA-CD system occurs when all of the channel time is spent in frame transmissions followed by contention intervals. Each frame transmission time X is followed by a period t_{prop} during which stations find out that the frame transmission is completed and then a contention interval of average duration $2e\ t_{prop}$ and, so the maximum throughput is then

$$\rho_{max} = \frac{X}{X + t_{prop} + 2et_{prop}} = \frac{1}{1 + (2e + 1)a} = \frac{1}{1 + (2e + 1)Rd/vL} \tag{6.9}$$

where $a = t_{prop}/X$ is the propagation delay normalized to the frame transmission time. The rightmost term in the above expression is in terms of the bit rate of the medium R, the diameter of the medium d, the propagation speed over the medium v, and the frame length L. The expression shows that CSMA-CD can achieve throughputs that are close to 1 when a is much smaller than 1. For example, if $a = 0.01$, then CSMA-CD has a maximum throughput of 94 percent. The CSMA-CD scheme provides the basis for the Ethernet LAN protocol.

As the load approaches the maximum throughput in Equation (6.9), the average transfer delay increases as collisions occur more frequently. This effect can be seen in Figure 6.52 (later in the chapter) which shows the frame transfer delay for various values of a. It should be emphasized that CSMA-CD does not provide an orderly transfer of frames. The random backoff mechanism and the random occurrence of collisions imply that frames need not be transmitted in the order that they arrived. Indeed once a frame is involved in a collision, it is quite possible for other later arrivals to be transferred ahead of it. This behavior implies that in CSMA-CD frame delays exhibit greater variability than in first-in-first-out statistical multiplexers. In the next section we will consider scheduling approaches to medium access control that provide a more orderly access to the shared medium.

Figure 6.18 shows a comparison of the maximum throughput of the main random access MAC techniques discussed in this section. For small values of a, CSMA-CD has the highest maximum throughput followed by CSMA. As a increases, the maximum throughput of CSMA-CD and CSMA decrease since the reaction times are approaching the frame transmission times. As a approaches 1, the throughput of these schemes becomes less than that of ALOHA and slotted ALOHA. As indicated before, ALOHA

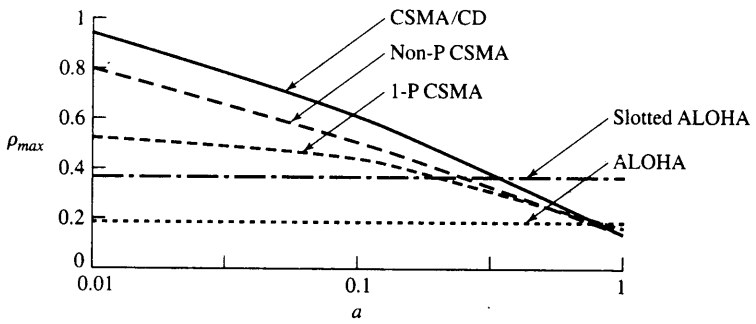


FIGURE 6.18 Maximum achievable throughputs of random access schemes.

and slotted ALOHA are not sensitive to a since their operation does not depend on the reaction time.

EXAMPLE **Cellular Digital Packet Data (CDPD) MAC protocol**

CDPD uses a 30 kHz channel of the AMPS analog cellular telephone system to provide packet data service to multiple mobile stations at speeds of up to 19,200 bps. The base stations are connected to interface nodes that connect to wired packet networks. The MAC protocol in CDPD is called *Digital Sense Multiple Access* and is a variation of CSMA-CD. The base station broadcasts frames in the forward channel, and mobile stations listen for packets addressed to them. The frames are in HDLC format and are segmented into blocks of 274 bits. Reed-Solomon error-correction coding increases the block to 378 bits. This block is transmitted in seven groups of 54 bits. A 6-bit synchronization and flag word is inserted in front of each 54-bit group to form a microblock. The flag word is used to provide coordination information in the reverse link.

To transmit on the reverse link, a station prepares a frame in HDLC format and prepares 378-bit blocks as in the forward channel. The station observes the flag words in the forward channel to determine whether the reverse link is idle or busy. This step is the "digital sensing." If the channel is busy, the station schedules a backoff time after which it will attempt again. If the station again senses the channel busy, it selects a random backoff time again but over an interval that is twice as long as before. Once it senses the channel idle, the station begins transmission. The base station provides feedback information, with a two-microblock delay, in another flag bit to indicate that a given 54-bit block has been received correctly. If a station finds out that its transmission was involved in a collision, the station aborts the transmission as in CSMA-CD.

INTRODUCING PRIORITY IN CARRIER SENSING

A simple mechanism to provide different levels of priority in carrier sensing access systems is to assign mandatory different interframe times to different priority classes. Thus a high-priority frame must wait τ_1 seconds after the end of a transmission before it can attempt to access the medium. A lower-priority frame would wait $\tau_2 > \tau_1$ seconds. This mechanism ensures that high-priority traffic gets first opportunity to capture the channel. The mechanism is implemented in 802.11 systems.

6.3 SCHEDULING APPROACHES TO MEDIUM ACCESS CONTROL

In Section 6.2 we considered random access approaches to sharing a transmission medium. These approaches are relatively simple to implement, and we found that under light traffic they can provide low-delay frame transfer in broadcast networks.

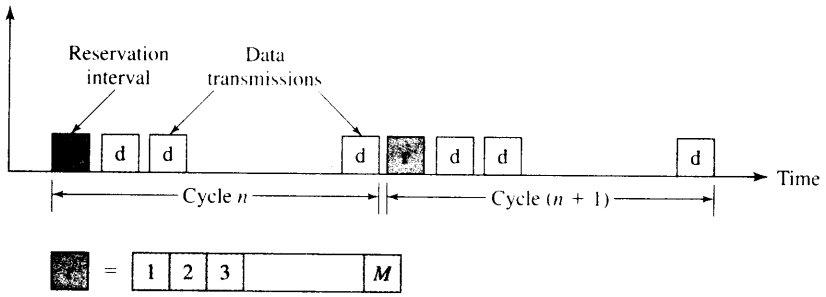


FIGURE 6.19 Basic reservation system: each station has own minislot for making reservations.

However, the randomness in the access can limit the maximum achievable throughput and can result in large variability in frame delays under heavier traffic loads. In this section we look at scheduling approaches to medium access control. These approaches attempt to produce an orderly access to the transmission medium. We first consider reservation systems and then discuss polling systems as a special form of reservation systems. Finally, we consider token-passing ring networks.

6.3.1 Reservation Systems

Figure 6.19 shows a basic reservation system. The stations take turns transmitting a single frame at the full rate R bps, and the transmissions from the stations are organized into cycles that can be variable in length. Each cycle begins with a reservation interval. In the simplest case the reservation interval consists of M minislots, one minislot per station. Stations use their corresponding minislot to indicate that they have a frame to transmit in a corresponding cycle. The stations announce their intention to transmit a frame by broadcasting their reservation bit during the appropriate minislot. By listening to the reservation interval, the stations can determine the order of frame transmissions in the corresponding cycle. The length of the cycle will then correspond to the number of stations that have a frame to transmit. Note that variable-length frames can be handled if the reservation message includes frame-length information

The basic reservation system described above generalizes and improves on a time-division multiplexing scheme by taking slots that would have gone idle and making them available to other stations. Figure 6.20a shows an example of the operation of the basic reservation system. In the initial portion only stations 3 and 5 have frames to transmit. In the middle portion of the example, station 8 becomes active, and the cycle is expanded from two slots to three slots.

Let us consider the maximum attainable throughput for this system. Assume that the propagation delay is negligible, that frame transmission times are $X = 1$ time unit, and that a reservation minislot requires v time units where $v < 1$.[3] Assume also that one minislot is required per frame reservation. Each frame transmission then requires

[3]Equivalently, we can assume that a slot takes X seconds and a minislot vX seconds.

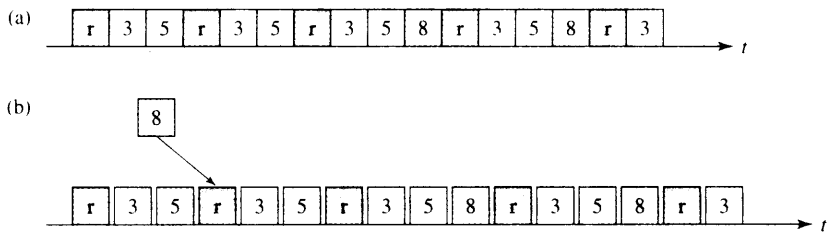


FIGURE 6.20 Operation of reservation system with (a) negligible and (b) nonnegligible delays.

$1 + v$ time units. The maximum throughput occurs when all stations are busy, and hence the maximum throughput is

$$\rho_{max} = \frac{1}{1+v} \qquad \text{for one frame reservation/minislot} \tag{6.10}$$

It can be seen that very high throughputs are achievable when v is very small in comparison to 1. Thus, for example, if $v = 5\%$, then $\rho_{max} = 95\%$.

Suppose that the propagation delay is not negligible. As shown in Figure 6.20b, the stations transmit their reservations in the same way as before, but the reservations do not take effect until some fixed number of cycles later. For example, if the cycle length is constrained to have some minimum duration that is greater than the round-trip propagation delay, then the reservations would take effect in the second following cycle.

The basic reservation system can be modified so that stations can reserve more than one slot per frame transmission per minislot. Suppose that a minislot can reserve up to k frames. The maximum cycle size occurs when all stations are busy and is given by $Mv + Mk$ time units. One such cycle transmits Mk frames, and so we see that the maximum achievable throughput is now

$$\rho_{max} = \frac{Mk}{Mv + Mk} = \frac{1}{1 + v/k} \qquad \text{for } k \text{ frame reservations/minislot} \tag{6.11}$$

Now let us consider the impact of the number of stations on the performance of the system. The effect of the reservation intervals is to introduce overhead that is proportional to M, that is, the reservation interval is Mv. If M becomes very large, this overhead can become significant. This situation becomes a serious problem when a very large number of stations transmit frames infrequently. The reservation minislots are incurred in every cycle, even though most stations do not transmit. The problem can be addressed by *not* allocating a minislot to each station and instead making stations contend for a reservation minislot by using a random access technique such as ALOHA or slotted ALOHA. If slotted ALOHA is used, then each successful reservation will require $1/0.368 = 2.71$ minislots on average. Therefore, the maximum achievable throughput for a reservation ALOHA system is

$$\rho_{max} = 1/(1 + 2.71v) \tag{6.12}$$

If again we assume that $v = 5\%$, then we have $\rho_{max} = 88\%$. The GPRS protocol, discussed in Section 6.4.4, uses a reservation protocol with slotted ALOHA to provide data service over GSM cellular telephone networks.

If the propagation delay is not negligible, then it is possible for slots to go unused because reservations cannot take effect quickly enough. This situation results in a reduction in the maximum achievable throughput. For this reason reservation systems are sometimes modified so that frames that arrive during a cycle can attempt to "cut ahead of the line" by being transmitted during periods that all stations know have not been reserved. If a frame is successfully transmitted this way, its reservation in a following cycle is canceled.

6.3.2 Polling

The reservation systems in the previous section required that stations make explicit reservations to gain access to the transmission medium. We now consider **polling systems** in which stations *take turns* accessing the medium. At any given time only one of the stations has the right to transmit into the medium. When a station is done transmitting, some mechanism is used to pass the right to transmit to another station.

There are different ways for passing the right to transmit from station to station. Figure 6.21a shows the situation in which M stations communicate with a host computer. The system consists of an outbound line in which information is transmitted from the host computer to the stations and an inbound line that must be shared with the M stations. The inbound line is a shared medium that requires a medium access control to coordinate the transmissions from the stations to the host computer. The technique developed for this system involves the host computer acting as a central

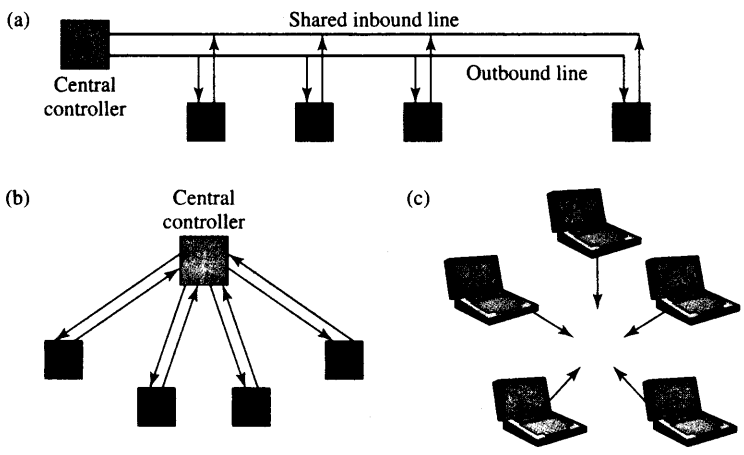


FIGURE 6.21 Examples of polling systems: (a) polling by central controller over lines; (b) polling by central controller over radio transmissions; and (c) polling without a central controller.

controller that issues control messages to coordinate the transmissions from the stations. The central controller sends a *polling message* to a particular station. When polled, the station sends its inbound frames and indicates the completion of its transmission through a *go-ahead message*. The central controller might poll the stations in round-robin fashion, or according to some other pre-determined order. The normal response mode of HDLC, which was discussed in Chapter 5, was developed for this type of system.

Figure 6.21b shows another situation where polling can be used. Here the central controller may use radio transmissions in a certain frequency band to transmit outbound frames, and stations may share a different frequency band to transmit inbound frames. This technique is called the **frequency-division duplex (FDD)** approach. Again the central controller can coordinate transmissions on the inbound channel by issuing polling messages. Another variation of Figure 6.21b involves having inbound and outbound transmissions share one frequency band. This is the **time-division duplex (TDD)** approach. In this case we would have an alternation between transmissions from the central controller and transmissions from polled stations.

Figure 6.21c shows a situation where polling is used without a central controller. In this particular example we assume that the stations have developed a polling order list, using some protocol. We also assume that all stations can receive the transmissions from all other stations. After a station is done transmitting, it is responsible for sending a polling message to the next station in the polling list.

Figure 6.22 shows the sequence of polling messages and transmissions in the inbound line that are typical for the preceding systems. In the example, station 1 is polled first. A certain time, called the **walk time**, elapses while the polling message propagates and is received and until station 1 begins transmission. The next period is occupied by the transmission from station 1. This period is followed by the walk time that elapses while station 2 is polled and then by the transmissions from station 2. This process continues until station M is polled and has completed its transmissions. At this point the polling cycle is begun again by polling station 1. In some systems a station is allowed to transmit as long as it has information in its buffers. In other systems the transmission time for each station is limited to some maximum duration.

The **total walk time** τ' is the sum of the walk times in one cycle and represents the minimum time for one round of polling of all the stations. The walk time between consecutive stations t' is determined by several factors. The first factor is the propagation time required for a signal to propagate from one station to another. This time is clearly a function of distance. Another factor is the time required for a station to

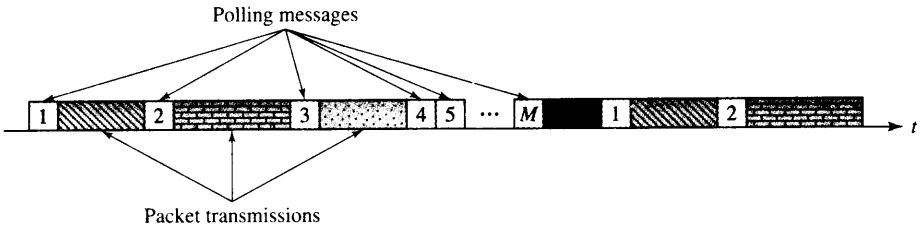


FIGURE 6.22 Interaction of polling messages and transmissions in a polling system.

begin transmitting after it has been polled. This time is an implementation issue. A third factor is the time required to transmit the polling message. These three factors combine to determine the total walk time of the system.

The **cycle time** T_c is the total time that elapses between the start of two consecutive polls of the same station. The cycle time is the sum of the M walk times and the M station transmission times. The average cycle time $E[T_c]$ can be found as follows. Let λ/M frames/second be the average arrival rate of frames for transmission from a station, and let $E[N_c]$ be the average number of message arrivals to a station in one cycle time. If we assume that all messages that arrive in a cycle time are transmitted the next time the station is polled, then $E[N_c] = (\lambda/M)E[T_c]$. Assume that all stations have the same frame transmission time X. Therefore, the time spent at each station is $E[N_c]X + t'$, where t' is the walk time. The average cycle time is then M times the average time spent at each station:

$$E[T_c] = M\{E[N_c]X + t'\} = M\left\{\frac{\lambda}{M}E[T_c]X + t'\right\}. \tag{6.13}$$

The preceding equation can be solved for $E[T_c]$:

$$E[T_c] = \frac{Mt'}{1 - \lambda X} = \frac{\tau'}{1 - \rho}. \tag{6.14}$$

Note the behavior of the mean cycle time as a function of load $\rho = \lambda X$. Under light load the cycle time is simply required to poll the full set of stations, and the mean cycle time is approximately τ', since most stations do not have messages to transmit. However, as the load approaches 1, the cycle time can increase without bound.

The walk times required to pass control of the access right to the medium can be viewed as a form of overhead. The normalized overhead per cycle is then given by the ratio of the total walk time to the cycle time. Note that as the load approaches 1, τ' remains constant while the cycle time increases without bound. In effect, the overhead due to polling, that is, the walk time, becomes negligible. This implies that polling can achieve a maximum normalized throughput of 100% when stations are allowed to send all of the frames in their buffers.

It is interesting to consider the cases where the walk time approaches zero. This yields a system in which frames from different queues are transmitted in turn according to a polling list. The zero walk time implies that the system can switch between queues very quickly, with negligible overhead. In this sense, the system operates much like a statistical multiplexer that serves user queues in some order.

In situations where stations carry delay-sensitive traffic it is desirable to have a cycle time that has a strict upper bound. The stations will then receive polling messages at some minimum frequency. The cycle time can be bounded by limiting the time or amount of information that a station is allowed to send per poll. For example, if a station is limited to one frame transmission per poll then the maximum cycle time is given by $MX + \tau'$. Note however that the maximum normalized throughput will now be given by $MX/(MX + \tau') = (1 + \tau'/MX)^{-1}$ which is less than 100 percent.

6.3.3 Token-Passing Rings

Polling can be implemented in a distributed fashion on networks with a ring topology. As shown in Figure 6.23, such ring networks consist of station interfaces that are connected by point-to-point digital transmission lines. Each interface acts like a repeater in a digital transmission line but has some additional functions. An interface in the listen mode reproduces each bit that is received from its input to its output after some constant delay, ideally in the order of one bit time. This delay allows the interface to monitor the passing bit stream for certain patterns. For example, the interface will be looking for the address of the attached station. When such an address is observed, the associated frame of information is copied bit by bit to the attached station. The interface also monitors the passing bit stream for the pattern corresponding to a "free token."

When a free token is received and the attached station has information to send, the interface changes the passing token to busy by changing a particular bit in the passing stream. In effect, receiving a free token corresponds to receiving a polling message. The station interface then changes to the transmit mode where it proceeds to transmit frames of information from the attached station. These frames circulate around the ring and are copied at the destination station interfaces.

While the station is transmitting its information, it is also receiving information at the input of the interface. If the time to circulate around the ring is less than the time to transmit a frame, then this arriving information corresponds to bits of the same frame that the station is transmitting. When the ring circulation time is greater than a frame transmission time, more than one frame may be present in the ring at any given time. In such cases the arriving information could correspond to bits of a frame from a different station, so the station must buffer these bits for later transmission.

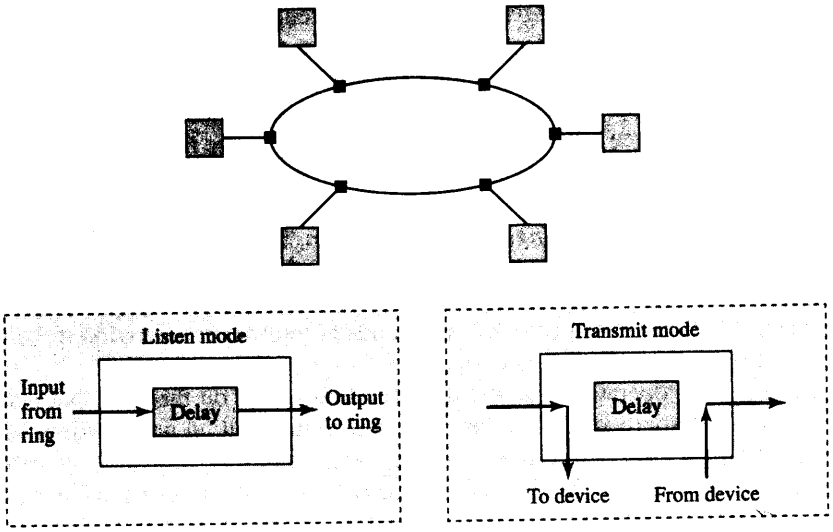


FIGURE 6.23 Token-passing rings: Interfaces monitor the ring (in listen or transmit mode) on behalf of their attached stations.

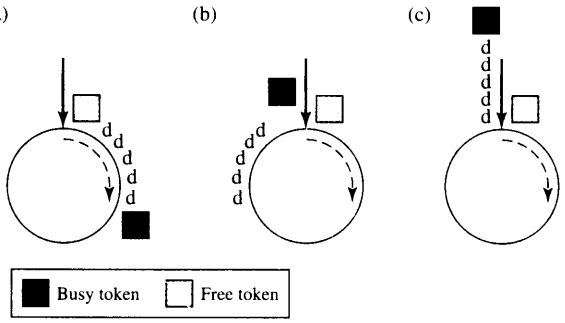


FIGURE 6.24 Approaches to token reinsertion: (a) multitoken, (b) single token, and (c) single frame.

A frame that is inserted into the ring must be removed. One approach to frame removal is to have the destination station remove the frame from the ring. Another approach is to allow the frame to travel back to the transmitting station. This approach is usually preferred because the transmitting station interface can then forward the arriving frame to its attached station, thus providing a form of acknowledgment.

Token rings can also differ according to the method used to reinsert the token after transmission has been completed. There are three approaches to token reinsertion, as shown in Figure 6.24. The main differences between the methods arise when the ring latency is larger than the frame length. The **ring latency** is defined as the number of bits that can be simultaneously in transit around the ring. In the *multitoken operation*, the free token is transmitted immediately after the last bit of the data frame. This approach minimizes the time required to pass a free token to the next station. It also allows several frames to be in transit in different parts of the ring.

The second approach, the *single-token operation*, involves inserting the free token after the last bit of the busy token is received back and the last bit of the frame is transmitted. If the frame is longer than the ring latency, then the free token will be inserted immediately after the last bit of the frame is transmitted, so the operation is equivalent to multitoken operation. However, if the ring latency is greater than the frame length, then a gap will occur between the time of the last bit transmission and the reinsertion of the free token as shown in Figure 6.24b. The recovery from errors in the token is simplified by allowing only one token to be present in the ring at any given time.

In the third approach, a *single-frame operation*, the free token is inserted after the transmitting station has received the last bit of its frame. This approach allows the transmitting station to check the return frame for errors before relinquishing control of the token. Note that this approach corresponds to multitoken operation if the frame length is augmented by the ring latency.

The token-ring operation usually also specifies a limit on the time that a station can transmit. One approach is to allow a station to transmit an unlimited number of frames each time a token is received. This approach minimizes the delay experienced by frames but allows the time that can elapse between consecutive arrivals of a free token to a station to be unbounded. For this reason, a limit is usually placed either on the number of frames that can be transmitted each time a token is received or on the total time that a station may transmit information into the ring. These limits have the

effect of placing a bound on the time that elapses between consecutive arrivals of a free token at a given station.

The introduction of limits on the number of frames that can be transmitted per token affects the maximum achievable throughput. Suppose that a maximum of one frame can be transmitted per token. Let τ' be the ring latency (in seconds) and a' be the ring latency normalized to the frame transmission time. We then have

$$\tau' = \tau + \frac{Mb}{R} \qquad a' = \frac{\tau'}{X} \tag{6.15}$$

where τ is the total propagation delay around the ring, b is the number of bit delays in an interface, Mb is the total delay introduced by the M station interfaces, and R is the speed of the transmission lines. The maximum throughput occurs when all stations transmit a frame. If the system uses multitoken operation, the total time taken to transmit the frames from the M stations is $MX + \tau'$. Because MX of this time is spent transmitting information, the maximum normalized throughput is then

$$\rho_{max} = \frac{MX}{MX + \tau'} = \frac{1}{1 + \tau'/MX} = \frac{1}{1 + a'/M} \quad \text{for multitoken.} \tag{6.16}$$

Now suppose that the ring uses single-token operation. From Figure 6.24b we can see that the effective frame duration is the maximum of X and τ'. Therefore, the maximum normalized throughput is then

$$\begin{aligned} \rho_{max} &= \frac{MX}{M \max\{X, \tau'\} + \tau'} = \frac{1}{\max\{1, a'\} + \tau'/MX} \\ &= \frac{1}{\max\{1, a'\} + a'/M} \quad \text{for single-token.} \end{aligned} \tag{6.17}$$

When the frame transmission time is greater than the ring latency, we see that the single-token operation has the same maximum throughput as multitoken operation. However, when the ring latency is larger than the frame transmission time, that is, $a' > 1$, then the maximum throughput is less than that of multitoken operation.

Finally, in the case of single-frame operation the effective frame transmission time is always $X + \tau'$. Therefore, the maximum throughput is given by

$$\rho_{max} = \frac{MX}{M(X + \tau') + \tau'} = \frac{1}{1 + a'\left(1 + \frac{1}{M}\right)} \quad \text{for single-frame.} \tag{6.18}$$

We see that the maximum throughput for single-frame operation is the lowest of the three approaches. Note that when the ring latency is much bigger than the frame transmission time, the maximum throughput of both the single-token and single-frame approaches is approximately $1/a'$. Recall from Figure 6.24 that this situation occurs when the distance of the ring becomes large or the transmission speed becomes very high. Figure 6.25 shows the maximum throughput for the three approaches for different values of a'. It is clear that single-frame operation has the lowest maximum throughput for all values of a'. Multitoken operation, on the other hand, has the highest maximum throughput for all values of a'. In fact, multitoken operation is sensitive to the per hop

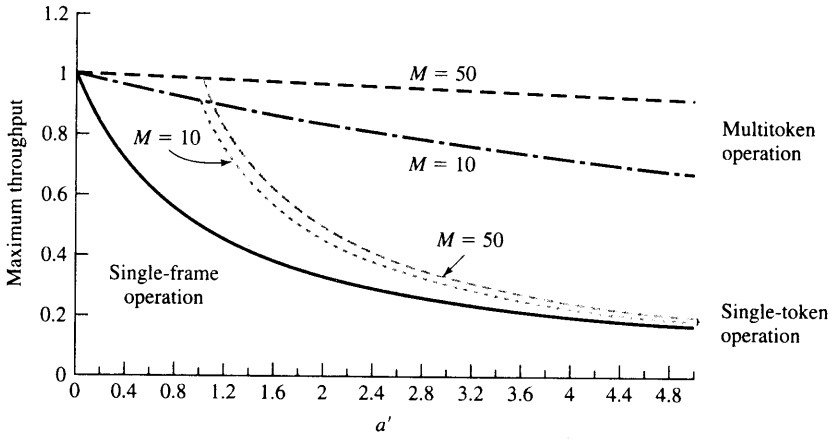


FIGURE 6.25 Throughput comparisons for single frame/token schemes.

latency a'/M, not the overall ring latency a'. The figure also shows how single-token operation approaches single-frame operation as a' becomes large.

6.3.4 Comparison of Scheduling Approaches in Medium Access Control

We have discussed two basic scheduling approaches to medium access control: reservations and polling. Token-passing rings are essentially an extension of the polling concepts to ring-topology networks. The principal strength of these approaches is that they provide a relatively fine degree of control in accessing the medium. These approaches can be viewed as an attempt to make time-division multiplexing more efficient by making idle slots available to other users.

Reservation systems are the most direct in obtaining the coordination in medium access that is inherent in a multiplexer. Reservation systems can be modified to implement the various scheduling techniques that have been developed to provide quality-of-service guarantees in conventional multiplexers. However, unlike centralized multiplexers, reservation systems must deal with the overheads inherent in multiple access communications, for example, time gaps between transmissions and reaction-time limitations. In addition, the decentralized nature of the system requires the reservation protocols to be robust with respect to errors and to be conducive to simple error-recovery procedures.

Polling systems and token-ring systems in their most basic form can be viewed as dynamic forms of time-division multiplexing where users transmit in round-robin fashion, but only when they have information to send. In polling systems the overhead is spread out in time in the form of walk times. The limitations on transmission time/token can lead to different variations. At one extreme, allowing unlimited transmission time/token minimizes delay but also makes it difficult to accommodate frames with stringent delay requirements. At the other extreme, a limit of one frame/token

leads to a more efficient form of time-division multiplexing. Polling systems however can be modified so that the polling order changes dynamically. When we reach the extreme where the polling order is determined by the instantaneous states of the different stations, we obtain what amounts to a reservation system From this viewpoint polling systems can be viewed as an important special case of reservation systems.

All the scheduling approaches were seen to be sensitive to the reaction time as measured by the propagation delay and the network latency normalized by the average frame transmission time. The reaction time of a scheme is an unavoidable limitation of scheduling approaches. Thus in reservation systems with long propagation delays, there is no way for the reservations to take effect until after a full propagation delay time. However, in some cases there is some flexibility in what constitutes the minimum reaction time. For example, in token-passing rings with single-frame operation the reaction time is a full ring latency, whereas in multitoken operation the reaction time is the latency of a single hop.

6.3.5 Comparison of Random Access and Scheduling Medium Access Controls

The two classes of medium access control schemes, random access and scheduling, differ in major ways, but they also share many common features. Their differences stem primarily from their very different points of departure. Scheduling techniques have their origins in reservation systems that attempt to emulate the performance of a centrally scheduled system such as a multiplexer. Random access techniques, on the other hand, have their origins in the ALOHA scheme that involves transmitting immediately, and subsequently at random times in response to collisions. The scheduling approach provides methodical orderly access to the medium, whereas random access provides a somewhat chaotic, uncoordinated, and unordered access. The scheduling approach has less variability in the delays encountered by frames and therefore has an edge in supporting applications with stringent delay requirements. On the other hand, when bandwidth is plentiful, random access systems can provide very small delays as long as the systems are operated with light loads.

Both random access and scheduling schemes have the common feature that channel bandwidth is used to provide information that controls the access to the channel. In the case of scheduling systems, the channel bandwidth carries explicit information that allows stations to schedule their transmissions. In the case of random access systems, channel bandwidth is used in collisions to alert stations of the presence of other transmissions and of the need to spread out their transmissions in time. Indeed, the contention process in CSMA-CD amounts to a distributed form of scheduling to determine which station should transmit next.

Any attempt to achieve throughputs approaching 100 percent involves using some form of coordination, either through polling, token-passing, or some form of contention resolution mechanism. All such systems can be very sensitive to the reaction time in the form of propagation delay and network latency. The comparison of the single-frame and multitoken approaches to operating a token ring shows that a judicious choice of

protocol can result in less demanding reaction times. Truly random access schemes such as ALOHA and slotted ALOHA do not attempt coordination and are not sensitive to the reaction time, but they also do not achieve high throughputs.

In keeping with the title of this book, this section has focused on fundamental aspects of medium access control. The current key standards for LANs that use these MACs are discussed in Part II of this chapter. As an aside we would like to point out to the student and designer of future networks that as technologies advance, the key standards will change, but the fundamental concepts will remain. In other words, there will be many new opportunities to apply the fundamental principles to develop the key standards of the future, especially in the area of wireless networks and optical networks.

HAVING IT BOTH WAYS: MULTIMODE MEDIUM ACCESS CONTROLS

Medium access controls need to be modified as networks evolve from providing a single service, such as data transfer only, to multiple services such as voice, image, video, and data transfer. Typically the MAC protocol is called upon to meet the transfer requirements of data while also meeting the relatively stringent access delay requirements from services such as voice. Multimode medium access controls address this situation by combining a polling/scheduling mechanism with a random access mechanism. Typically the medium access is organized in cycles that consist of several frame transmissions. Frames that contain traffic requiring predictable low access delay are scheduled or polled in the first part of a cycle, and other "data traffic" is provided access in the rest of a cycle possibly through a random access mechanism. To provide predictable access to the low-delay traffic, cycles are limited to some given maximum duration. Thus the data traffic gains access only to the residual portion of each cycle. To guarantee that data traffic receives some minimum amount of bandwidth, the number of low-delay connections needs to be kept below some level. We will see later in the chapter that the FDDI ring protocol and the 802.11 wireless LAN protocol use medium access controls of this type.

◆ 6.4 CHANNELIZATION

Consider a network in which the M stations that are sharing a medium produce the same steady flow of information, for example, digital voice or audio streams. It then makes sense to divide the transmission medium into M channels that can be allocated for the transmission of information from each station. In this section we first present two channelization schemes that are generalizations of frequency-division multiplexing and time-division multiplexing: frequency-division multiple access (FDMA) and time-division multiple access (TDMA). The third channelization technique, code-division multiple access (CDMA), involves coding the transmitted signal to produce a number of separate channels. We explain the basic features of CDMA and compare it to TDMA and FDMA. Finally we discuss how all three of these techniques have been applied in telephone cellular networks.

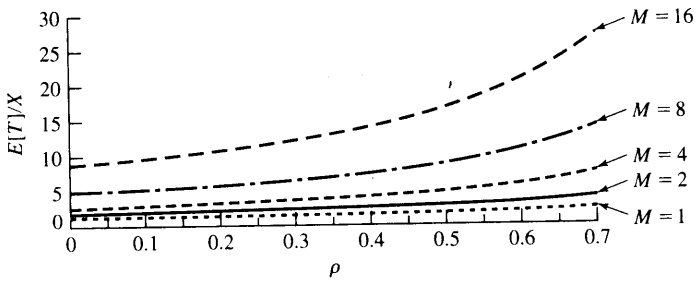


FIGURE 6.26 Average delay for TDMA.

Before proceeding into a description of channelization techniques, we explain why these techniques are *not suitable for the transmission of bursty data traffic*. In dividing the bandwidth of a channel into equal bands and *allocating* these to specific stations or users, we reduce the degree of sharing that can take place. Figure 6.27 shows the average frame transfer delay for TDMA (derived in Section 6.5.1) that is representative of the performance of channelization techniques. We see that the delay increases with M. We note that the root cause of this inferior frame delay performance is due to the fact that when the medium is divided among several stations, transmission slots can go unused when a station has no frames to transmit, whereas in a combined system these slots would be used by other stations. The problem, of course, is that we are dealing with a multiaccess system in which stations are geographically distributed. One way to view the dynamic MAC algorithms discussed in previous sections is as an attempt to achieve better sharing and hence better frame delay performance.

6.4.1 FDMA

In **frequency-division multiple access (FDMA)** the transmission medium is divided into M separate frequency bands. Each station transmits its information continuously on an assigned band. Bandpass filters are used to contain the transmitted energy within the assigned band. Because practical transmitters cannot completely eliminate the out-of-band energy, *guard bands* are introduced between the assigned bands to reduce the co-channel interference. We will suppose that the total bandwidth available to the station is W, which can support a total bit rate of R bits/second. For simplicity we neglect the effect of the guard bands and assume that each station can transmit at a rate of R/M bits/second on its assigned band. As shown in Figure 6.27, in FDMA a station uses a fixed portion of the frequency band *all the time*. For this reason, FDMA is suitable for stream traffic and finds use in connection-oriented systems such as cellular telephony where each call uses a forward and reverse channel to communicate to and from a base station.

If the traffic produced by a station is bursty, then FDMA will be inefficient in its use of the transmission resource. The efficiency can be improved by allocating a frequency band to a group of stations in which each station generates bursty traffic. However, because the stations are uncoordinated, they will also need to use a dynamic sharing technique, that is, a medium access control, to access the given frequency band.

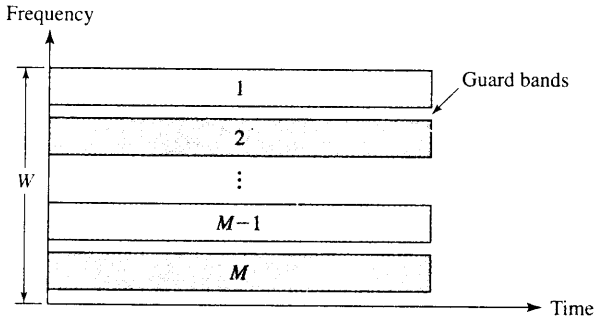


FIGURE 6.27 Frequency-division multiple access.

6.4.2 TDMA

In **time-division multiple access (TDMA)** stations take turns making use of the *entire* transmission channel. The stations periodically transmit in their appropriate time slots for each TDMA cycle, as shown in Figure 6.28. Each station transmits during its assigned time slot and uses the entire frequency band during its transmission. Thus each station transmits at R bits/second $1/M$ of the time for an average rate of R/M bits/second. Each station spends most of the time accumulating frames and preparing them for transmission in a burst during the assigned time slot. Unlike the TDM system where multiplexing occurs in a single location, different stations in different locations may experience different propagation delays in the TDMA system. To allow for inaccuracy in the propagation delay estimate, *guard times* are required to ensure that the transmissions from different stations do not overlap. Another source of overhead in TDMA is a *preamble* signal that is required at the beginning of each time slot to allow the receiver to synchronize to the transmitted bit stream.

In the basic form of TDMA, each station is assigned the same size time slot, so each station has a channel with the same average bit rate. However, TDMA can accommodate a wider range of bit rates by allowing a station to be allocated several slots or by allowing slots to be variable in duration. In this sense TDMA is more flexible

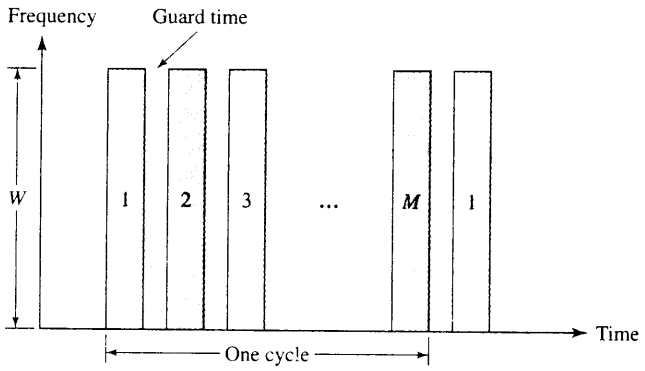


FIGURE 6.28 Time-division multiple access.

than FDMA. Nevertheless, in TDMA the bit rate allocated to a station is static, and this approach is not desirable for bursty traffic.

Similar to the system considered in Figure 6.21b (page 390), TDMA can be used to connect to a base station or controller in two ways. In the FDD approach, one band is used for communication in the "forward" direction from the base station to the other stations. The second band is used for communications in the "reverse" direction from the stations to the base station. Each station has an assigned time slot in the forward channel to receive from the base station, as well as an assigned time slot to transmit to the base station. Typically the slot assignments are staggered to allow a station to divide its time between transmit and receive processing functions. Observe that the forward direction simply involves TDM, since there is only one transmitter. In the TDD approach, the base station and the other stations take turns transmitting over the same shared channel. This approach requires the coordination of transmissions in the forward and reverse directions. FDD requires that the frequency bands be allocated ahead of time, and so it cannot be adapted to changes in the ratio between forward and reverse traffic. TDD can more readily adapt to these types of changes. We will see in Section 6.4.4 that TDMA is used with FDD in several digital cellular telephone systems. TDMA is also used with TDD in cordless telephone systems for in-building communications.

6.4.3 CDMA

Code-division multiple access (CDMA) provides another type of channelization technique. In TDMA and FDMA the transmissions from different stations are clearly separated in either time or in frequency. In CDMA the transmissions from different stations occupy the entire frequency band at the same time. The transmissions are separated by the fact that different codes are used to produce the signals that are transmitted by the different stations. The receivers use these codes to recover the signal from the desired station.

Suppose that the user information is generated at R_1 bits/second. As shown in Figure 6.29, each user bit is transformed into G bits by multiplying the user bit value (as represented by a $+1$ or a -1) by G "chip" values (again represented by $+1$s and -1s) according to a unique binary pseudorandom sequence that has been assigned to the station. This sequence is produced by a special code and appears to be random except that it repeats after a very long period. The resulting sequence of $+1$s and -1s is then digitally modulated and transmitted over the medium. The **spreading factor** G is selected so that the transmitted signal occupies the entire frequency band of the medium.[4] Thus we have a situation in which a user transmits over *all the frequency band all the time*. Other stations transmit in the same manner at the same time but use different binary random sequences to spread their binary information.

[4]We warn the reader that the throughput term G in the discussion of ALOHA is different from the spreading factor G in the discussion of spread spectrum systems. In situations where both terms arise, we suggest using the term G_{proc} for the spread spectrum term.

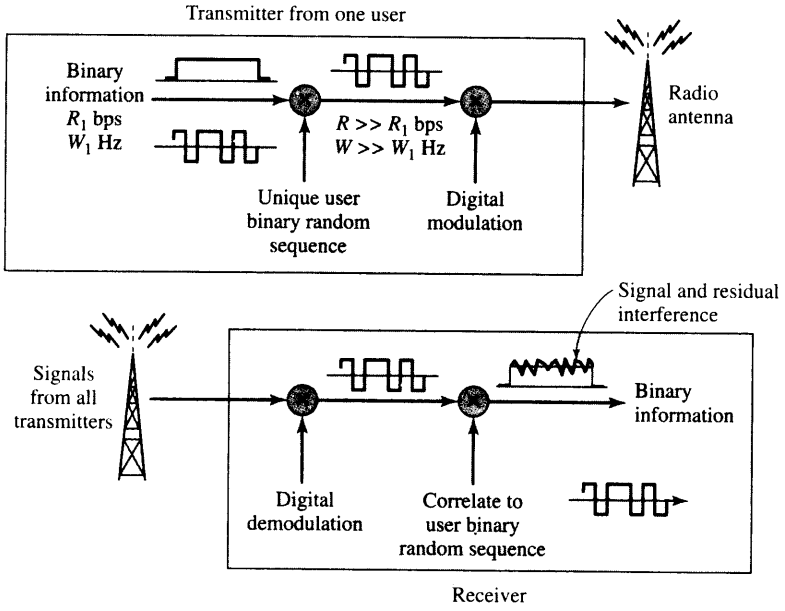
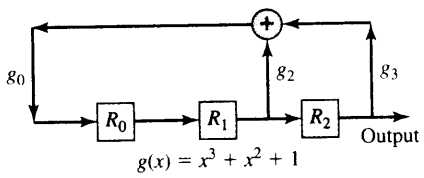


FIGURE 6.29 Code-division multiple access.

Let us see why this type of modulation works. Suppose that the spreading sequence is indeed random and selected ahead of time by the transmitter and receiver by flipping a fair coin G times. If the transmitter wants to send a $+1$ symbol, the modulator multiplies this $+1$ by a sequence of G chip values, say, $c_1, c_2, \ldots, c_G$, each of which takes on a $+1$ or -1 according to a random coin flip. When the signal arrives at the receiver, the received signal is correlated with the known chip sequence; that is, the arriving chips are multiplied by the known sequence $c_1, c_2, \ldots, c_G$, and the resulting products are added. The resulting output of the correlator is $c_1^2 + c_2^2 + \cdots + c_G^2 = G$, since $(-1)^2 = (+1)^2 = 1$. Now suppose that another receiver attempts to detect the sequence $c_1, c_2, \ldots, c_G$ using another random chip sequence, say, $d_1, d_2, \ldots, d_G$. The output of this correlator is $c_1d_1 + c_2d_2 + \cdots + c_Gd_G$. Because each c_j is equally likely to have value $+1$ or -1 and each d_j is also equally likely to have value $+1$ or -1, then each product term is equally likely to be $+1$ and -1. Consequently, the average value of the correlator output is 0. Indeed for large G, the vast majority of combinations of c_j and d_j will result in approximately half the terms in the sum being $+1$ and the other half -1, so the correlator output is almost always close to zero.[5] In conclusion, if the receiver uses the correct chip sequence then the correlator output is G; if it uses some other random chip sequence, then the correlator output is a random number that is usually close to zero. Thus as the value of G is increased, it becomes easier for the receiver to detect the signal. Consequently, it becomes possible to decrease the amplitude (and lower the power) of the transmitted signal as G is increased.

[5]The probability that there are k appearances of $+1$s in a sequence is the probability that there are k heads in G tosses of a fair coin. This probability is given by the binomial distribution.



The coefficients of a primitive generator polynomial determine the feedback taps

Time	R_0	R_1	R_2
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0

Sequence repeats from here onward

FIGURE 6.30 A maximum-length sequence generator.

The preceding discussion requires each transmitter and receiver pair to select very long random chip sequences ahead of time before they communicate. Clearly this approach is impractical, so an automated means of generating pseudorandom sequences is required. The shift-register circuits that we introduced when we discussed error-detection codes can be modified to provide these sequences. Figure 6.30 shows such a circuit. If the feedback taps in the feedback shift register are selected to correspond to the coefficients of a primitive polynomial, then the contents of the shift register will cycle over all possible $2^n - 1$ nonzero states before repeating. The resulting sequence is the maximum length possible and can be shown to approximate a random binary sequence in the sense that shifted versions of itself are approximately uncorrelated. Thus these sequences are suitable for spread spectrum communications.

In Chapter 3 we showed that transmitted signals can occupy the same frequency band and can still be separated by a receiver. In particular, in QAM modulation a sine and cosine can be used to produce signals that occupy the same frequency band but that can be separated at the receiver. In CDMA the spreading sequences that are used by the stations are approximately uncorrelated so that the correlators at the receiver can separate the signals from different transmitters. Thus to receive the information from a particular station, a receiver uses the same binary spreading sequence synchronized to the chip level to recover the original information, as shown in Figure 6.29. In the case of QAM modulation, the original signal could be recovered exactly. In the case of CDMA, the signals from the other stations appear as residual noise at the receiver. From time to time the correlator output will yield an incorrect value, so error-correction coding needs to be used. Nevertheless, low error rates can be attained as long as the residual noise is kept below a certain threshold. This situation in turn implies that the number of active transmitters needs to be kept below some value.

In our analysis of the correlator output, we assumed that the signal level for each received signal was the same at a given receiver. This is a crucial assumption, and in order to work properly, CDMA requires all the signals at the receiver to have approximately the same power. Otherwise, a powerful transmission from a nearby station could overwhelm the desired signal from a distant station, which is called the *near-far* problem. For this reason CDMA systems implement a power control mechanism that dynamically controls the power that is transmitted from each station.

Figure 6.31 shows how CDMA can be viewed conceptually as dividing the transmission medium into M channels in "codespace." Each channel is distinguished by its spreading sequence. Unlike FDMA and TDMA, CDMA provides a graceful trade-off

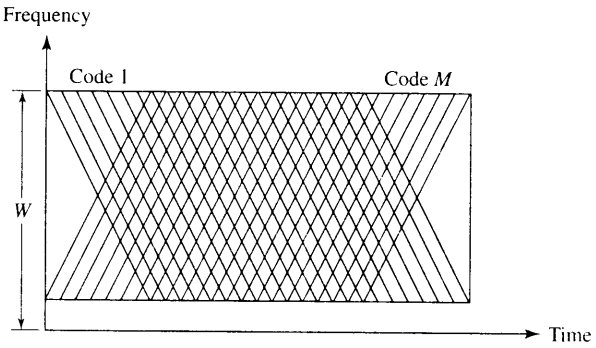


FIGURE 6.31 Conceptual view of CDMA (Code 1 is indicated by the back hatches and code M by the forward hatches).

between number of users and residual interference. As the number of users is increased, the residual interference that occurs at the receiver increases. Unlike FDMA and TDMA, CDMA degrades only gradually when the channels begin to overlap; that is, the interference between channels becomes evident through a gradual increase in the bit error rate. This behavior provides the system with flexibility in terms of how to service different types of traffic, for example, allow higher error rate for voice traffic but provide more error correction for data traffic.

The set of spreading codes that are to be used for different channels must be selected so that any pair of spreading sequences will have low cross-correlation. Otherwise, the correlator in the receiver in Figure 6.29 will not always be able to separate transmissions from stations using these codes.[6] In the discussion so far we have assumed that the transmissions from the stations are not synchronized. We will now show that when it is possible to synchronize transmissions for the different channels, then it is possible to eliminate the interference between channels by using **orthogonal sequences** in the spreading. This situation is possible, for example, in the transmission from a base station to the different mobile stations. It is also possible when a mobile station sends several subchannels in its transmissions back to the base. We will use a simple example to show how this is done.

EXAMPLE **Orthogonal Spreading by Using Walsh Functions**

Suppose that a four-station system uses the following four orthogonal spreading sequences to produce four channels: $\{(-1, -1, -1, -1), (-1, 1, -1, 1), (-1, -1, 1, 1),$ and $(-1, 1, 1, -1)\}$. These sequences are examples of the Walsh orthogonal functions. To transmit an information bit, a station converts a binary 0 to a -1 symbol and a binary 1 to a $+1$ symbol. The station then transmits the symbol times its spreading sequence. Thus station 2 with code $(-1, 1, -1, 1)$ transmits a binary 0 as $(1, -1, 1, -1)$ and a 1 as $(-1, 1, -1, 1)$.

[6]See [Stüber 1996, Chapter 8] or [Gibson 1999, Chapter 8] for a discussion on the selection of spreading sequences.

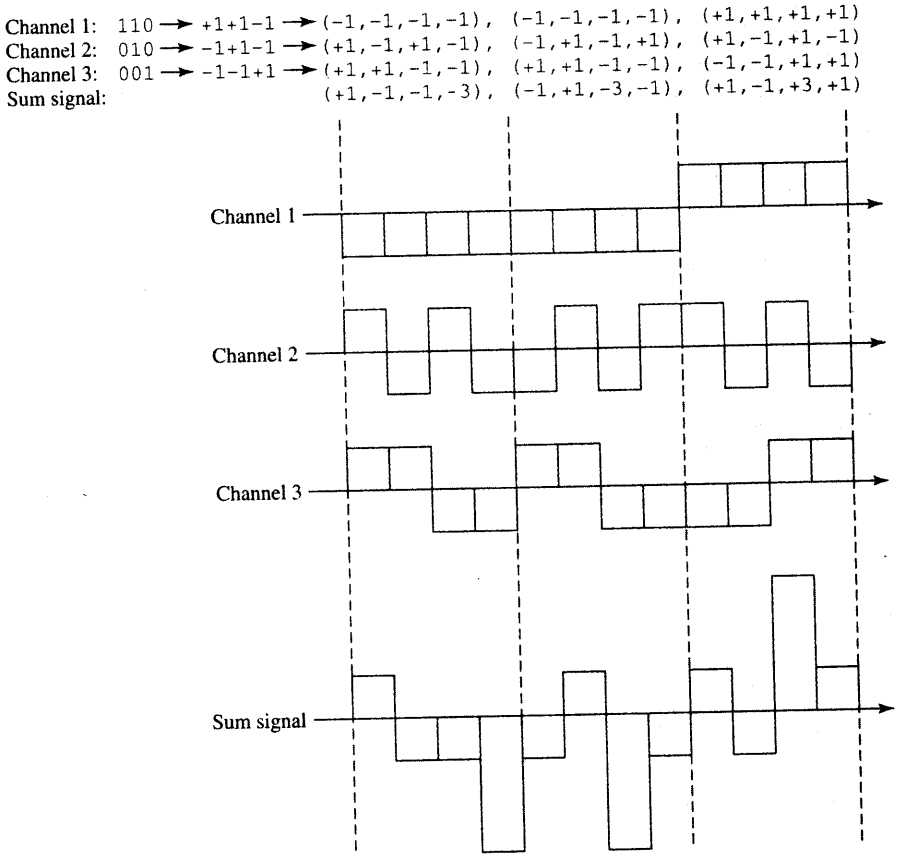


FIGURE 6.32 Example of orthogonal coding for channelization.

Now suppose that stations 1, 2, and 3 transmit the following respective binary information sequences: 110, 010, and 001. Figure 6.32 shows how these binary information bits are converted into symbols, how the orthogonal spreading is applied, and the resulting individual channel signals as well as the aggregate signal.

Figure 6.33 shows how the signal from channel 2 is recovered from the aggregate signal. The receiver must be synchronized to the time slot that corresponds to each symbol. The receiver then multiplies the aggregate signal by the four chip values of the spreading code for channel 2. The resulting signal is then integrated. Each integrated signal gives either $+4$ or -4, depending on whether the original symbol was $+1$ or -1. The same procedure can be used to recover the signals for channels 1 and 3. It should also be noted that if we try to recover the channel 2 signal with the spreading sequence for channel 4 $(-1, 1, -1, 1)$, we will obtain zero for each integrated period, indicating that no signal from channel 4 was present.

We also emphasize that the bit transmission times for the different channels must be aligned precisely. An error of a fraction of a single chip period can cause the receiver to fail.

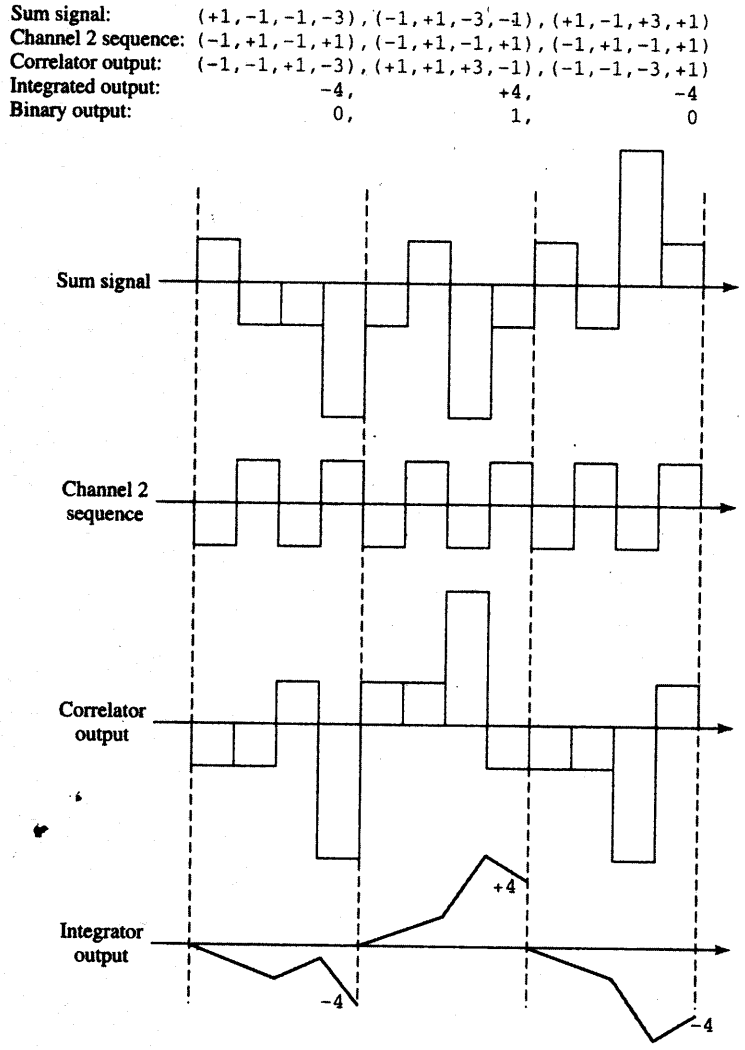


FIGURE 6.33 Example of channel signal recovery using orthogonal coding.

The preceding example shows that when orthogonal sequences are used for spreading then the signal for each channel can be recovered from the aggregate signal without any interference from the other channel signals. The reason for this behavior is in the orthogonality property itself. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ be two spreading sequences. We say that the two sequences are orthogonal if their inner product (also called the dot product) is zero:

$$\boldsymbol{a} * \boldsymbol{b} = \sum a_j b_j = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = 0 \tag{6.19}$$

Because the spreading sequences consist of +1s and −1s, we have

$$\boldsymbol{a} * \boldsymbol{a} = \sum a_j^2 = a_1^2 + a_2^2 + \cdots + a_n^2 = n \tag{6.20}$$

and

$$\boldsymbol{b} * \boldsymbol{b} = \sum b_j^2 = b_1^2 + b_2^2 + \cdots + b_n^2 = n \tag{6.21}$$

Now suppose that we transmit a binary 0 in the a channel and a binary 1 in the b channel, then the signal in the a channel is $-\boldsymbol{a} = -(a_1, a_2, \ldots, a_n)$, and the signal in the b channel is $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, so the aggregate channel signal is $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) = (-a_1 + b_1, -a_2 + b_2, \ldots, -a_n + b_n) = -\boldsymbol{a} + \boldsymbol{b}$. When a receiver attempts to recover the a channel, the receiver multiplies the jth received chip r_j by a_j and then integrates, or adds, the terms over all j:

$$\begin{aligned} \sum a_j r_j &= a_1 r_1 + a_2 r_2 + \cdots + a_n r_n = \boldsymbol{a} * \boldsymbol{r} = \boldsymbol{a} * (-\boldsymbol{a} + \boldsymbol{b}) \\ &= -\boldsymbol{a} * \boldsymbol{a} + \boldsymbol{a} * \boldsymbol{b} = -n + 0 = -n \end{aligned} \tag{6.22}$$

Therefore, the receiver will conclude that the symbol in channel a was −1 and the information bit was 0. Similarly, if a receiver tries to detect channel b, the receiver will obtain

$$\begin{aligned} \sum b_j r_j &= b_1 r_1 + b_2 r_2 + \cdots + b_n r_n = \boldsymbol{b} * \boldsymbol{r} = \boldsymbol{b} * (-\boldsymbol{a} + \boldsymbol{b}) \\ &= -\boldsymbol{b} * \boldsymbol{a} + \boldsymbol{b} * \boldsymbol{b} = 0 + n = n \end{aligned} \tag{6.23}$$

and so the receiver will conclude that the symbol and information bit in channel b were +1 and 1, respectively.

The **Walsh-Hadamard matrix** provides orthogonal spreading sequences of length $n = 2^m$. These matrices have binary coefficients and are defined recursively

$$\begin{aligned} \mathbf{W}_1 &= [0] \\ \mathbf{W}_{2n} &= \begin{bmatrix} \mathbf{W}_n & \mathbf{W}_n \\ \mathbf{W}_n & \mathbf{W}_n^c \end{bmatrix} \end{aligned} \tag{6.24}$$

where $\mathbf{W}_n^c$ is obtained by taking the complement of the elements of $\mathbf{W}_n$. Figure 6.34

$$\mathbf{W}_1 = \begin{pmatrix} 0 \end{pmatrix} \quad \mathbf{W}_2 = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & 1 \end{array}\right) \quad \mathbf{W}_4 = \left(\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right)$$

FIGURE 6.34 Construction of Walsh-Hadamard matrices.

$$\mathbf{W}_8 = \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right)$$

shows the construction of the Walsh-Hadamard matrices with $n = 2, 4, 8$. The n rows of a Walsh-Hadamard matrix provide a set of n orthogonal spreading sequences by replacing each 0 by a -1 and each 1 by a $+1$. Note from the figure that not all spreading sequences alternate quickly between $+1$ and -1. These sequences will not produce a transmitted signal that is spread over the available spectrum. For this reason the purpose of using Walsh sequences is primarily to provide channelization. In practice additional spreading using other sequences is combined with Walsh sequences. The resulting spread signal is robust with respect to multipath fading and interference in general.

6.4.4 Channelization in Telephone Cellular Networks

In this section we give an overview of how FDMA, TDMA, and CDMA channelization techniques have been implemented in various telephone cellular networks.

ADVANCED MOBILE PHONE SYSTEM AND IS-54/IS-136

The **Advanced Mobile Phone System (AMPS)**, which was developed in the United States, is an example of a first-generation telephone cellular system. The system had an initial allocation of 40 MHz that was divided between two service providers (A and B) as shown in Figure 6.35. The allocation was later increased to 50 MHz as shown in the figure. AMPS uses FDMA operation for transmission between a base station and mobile stations. One band is used for forward channels from the base station to the mobile stations and the other band is used for reverse channels from the mobile stations to the base station. Each channel pair is separated by 45 MHz. AMPS uses analog frequency modulation to send a single voice signal over a 30 kHz transmission channel. Thus the 50 MHz allocation provides for $(50 \times 10^6)/(2 \times 30 \times 10^3) = 832$ two-way channels. Of these channels 42 are set aside for control purposes, such as call setup, and the remainder are used to carry voice traffic. AMPS uses a seven-cell frequency reuse pattern so only one-seventh of the channels is available in a given cell. A measure of the **spectrum efficiency** in a cellular system is the number of calls/MHz/cell that can be supported. For AMPS each service provider has $416 - 21$ traffic channels that are divided over seven cells and 25 MHz, thus

$$\text{Spectrum efficiency for AMPS} = 395/(7 \times 25) = 2.26 \text{ calls/cell/MHz} \qquad (6.25)$$

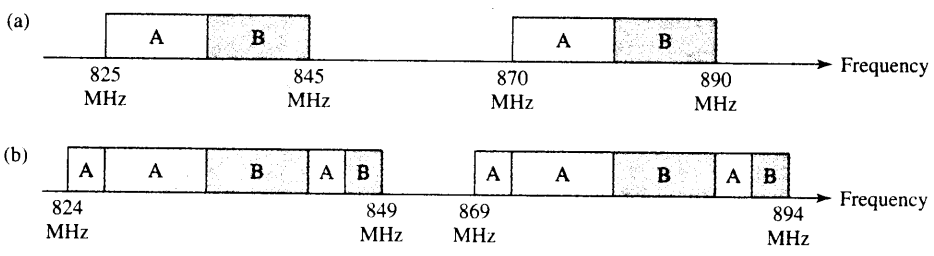


FIGURE 6.35 AMPS frequency allocation and channel structure: (a) initial allocation; (b) extended allocation.

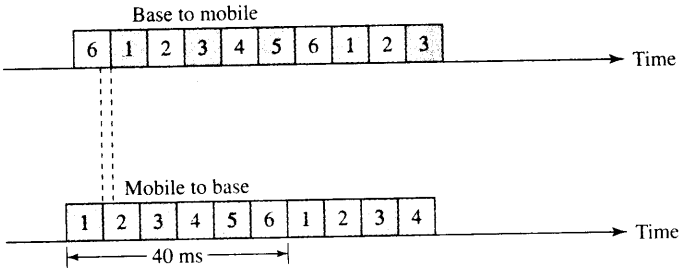


FIGURE 6.36 IS-54 TDMA structure.

The success of the cellular telephone service led to an urgent need to increase the capacity of the systems. This need was met through the introduction of digital transmission technologies. The **Interim Standard 54/136 (IS-54, IS-136)** was developed in North America to meet the demand for increased capacity. IS-54 uses a hybrid channelization technique that retains the 30 kHz structure of AMPS but divides each 30 kHz channel into several digital TDMA channels. This approach allows cellular systems to be operated in dual mode, AMPS and TDMA. Each 30 kHz channel carries a 48.6 kbps digital signal organized into six-slot cycles as shown in Figure 6.36. Each cycle has a duration of 40 ms, and each slot contains 324 bits. Thus each slot corresponds to a bit rate of 324 bits/40 ms = 8.1 kbps. Typically a *full-rate* channel consisting of two slots per cycle, and hence 16.2 kbps, is used to carry a voice call.[7] Thus IS-54 supports three digital voice channels in one analog AMPS channel. Half-rate channels (8.1 kbps), double full-rate channels (32.4 kbps), and triple full-rate channels (48.6 kbps) are also defined. Note that the time slots in the forward and reverse directions are offset with respect to each other to allow a mobile station to operate without having to transmit and receive at the same time. The 30 kHz spacing and hybrid TDMA/FDMA structure is also used in the 1.9 GHz PCS band. Interim Standard 136 is a revision of IS-54 that takes into account the availability of fully digital control channels.

To calculate the spectrum efficiency of IS-54, we note that the 416 analog channels available provide $3 \times 416 = 1248$ digital channels. If we suppose that 21 of these channels are used for control purposes and that the frequency reuse factor is 7, then we have

$$\text{Spectrum efficiency of IS-54} = 1227/(7 \times 25) = 7 \text{ calls/cell/MHz} \qquad (6.26)$$

GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS

The **Global System for Mobile Communications (GSM)** is a European standard for cellular telephony that has gained wide acceptance. GSM was designed to operate in the band 890 to 915 MHz for the reverse channels and the band 935 to 960 MHz for the forward channel (see Figure 6.37a). Initially the upper 10 MHz of this band was used for GSM, and the other portion of the band is used for existing analog services. The GSM technology can also be used in the PCS bands, 1800 MHz in Europe and 1900 MHz in North America.

[7]The actual bit rate for a voice signal is about 13 kbps, since only 260 of the 324 bits carry data.

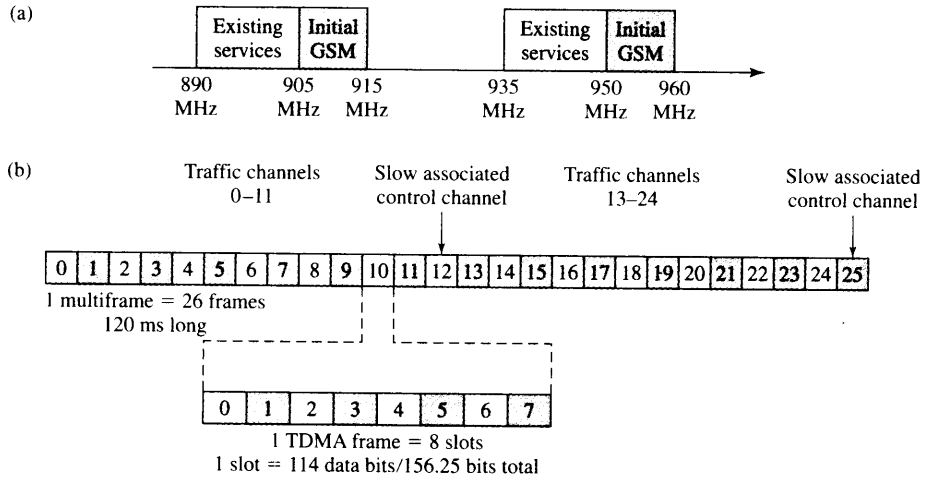


FIGURE 6.37 (a) GSM channel structure; (b) GSM TDMA structure.

GSM uses a hybrid TDMA/FDMA system. The available frequency band is divided into carrier signals that are spaced 200 kHz apart. Thus the 25 MHz bandwidth can support 124 one-way carriers. Each base station is assigned one or more carriers to use in its cell. Each carrier signal carries a digital signal that provides traffic and control channels. The carrier signal is divided into 120 ms multiframes, where each multiframe consists of 26 frames, and each frame has eight slots as shown in Figure 6.37b. Two frames in a multiframe are used for control purposes, and the remaining 24 frames carry traffic. In GSM the slots in the frames in the reverse direction lag the corresponding slots in the forward direction to allow the mobile station to alternate between receive and transmit processing.

A *full-rate traffic channel* uses one slot in every traffic frame in a multiframe. Therefore, the bit rate of a full-rate channel is

$$\begin{aligned}\text{Traffic channel bit rate} &= 24 \text{ slots/multiframe} \times 114 \text{ bits/slot} \\ &\quad \times (1 \text{ multiframe}/120 \text{ ms}) = 22{,}800 \text{ bps} \qquad (6.27)\end{aligned}$$

A substantial number of bits are used to provide error correction for voice calls. The full-rate traffic channel actually carries a digital voice signal that is 13 kbps, prior to the addition of error-correction bits. The hefty error-correcting capability allows GSM to operate with a frequency reuse factor of 3 or 4. If we assume 124 carriers in the 50 MHz band, then we obtain a total of $124 \times 8 = 992$ traffic channels. Assuming a frequency reuse factor of 3, we then have

$$\text{Spectrum efficiency of GSM} = 992/(3 \times 50) = 6.61 \text{ calls/cell/MHz} \qquad (6.28)$$

The higher frequency reuse factor allows GSM to achieve a spectrum efficiency close to that of IS-54.

EXAMPLE **General Packet Radio Service (GPRS)**

GPRS is an enhancement of the GSM network architecture to allow mobile stations to connect to IP or other packet networks. The medium access control in GPRS uses the following channels: a packet random access channel is used by mobile stations to initiate packet transfers; a packet access grant channel is used to send a grant to a mobile station prior to the packet transfer; and a packet data traffic channel is used to transfer the packet. When a mobile station has a packet to send, the station uses slotted ALOHA access to send a reservation request in the packet random access channel. The base station sends a notification to the successful mobile station indicating a channel allocation for the packet transmission. The mobile station then transmits the packet on the allocated packet data traffic channel. The transmission of packets from the base to the mobile is simpler. The base transmits a notification to the mobile indicating the channel for a pending packet transmission and the mobile monitors the indicated channel and receives the packet. Note that there is contention in the uplink from the mobile to the base, but there is no contention in the downlink from the base to the mobile.

IS-95

The **Interim Standard 95 (IS-95)** is a second standard developed in North America to meet the demand for increased capacity. IS-95 is based on spread spectrum communication, which represents a very different approach to partitioning the available bandwidth. Spread spectrum communication introduces into cellular communication new features that provide higher voice quality and capacity than IS-54 TDMA systems. IS-95 systems can operate in the original AMPS frequency bands as well as in the 1900 MHz PCS bands.

IS-95 supports dual-mode operation with AMPS. Each channel signal is spread into a 1.23 MHz band, so the conversion of spectrum of AMPS to IS-95 must be done in chunks of 41 AMPS channel $\times$ 30 kHz/channel = 1.23 MHz. Recall that each service provider has 12.5 MHz of bandwidth, so these chunks represent about 10 percent of the total band.

For reasons that will become apparent IS-95 requires that all base stations be synchronized to a common clock. This synchronization is achieved by using the Global Positioning System (GPS), which is a network of satellites that can provide accurate timing information to a precision of 1 microsecond. In addition, all base stations use a common *short code* pseudorandom sequence in the spreading of signals and in the production of pilot signals that are used in the forward channel as well as in the handoff between cells.[8] All the base stations transmit the same sequence that is produced by this short code, but each base station has a unique phase or timing offset of the signal. The use of the same sequence reduces the task of synchronizing to this pilot signal.

In IS-95 the forward and reverse direction use *different* transmission techniques. First we consider the transmission in the forward direction, from the base station to the

[8]The short code produces a pseudorandom sequence that repeats every $2^{15} - 1$ chip times, which translates into $80/3 = 26.667$ ms at the 1.2288 MHz spreading rate.

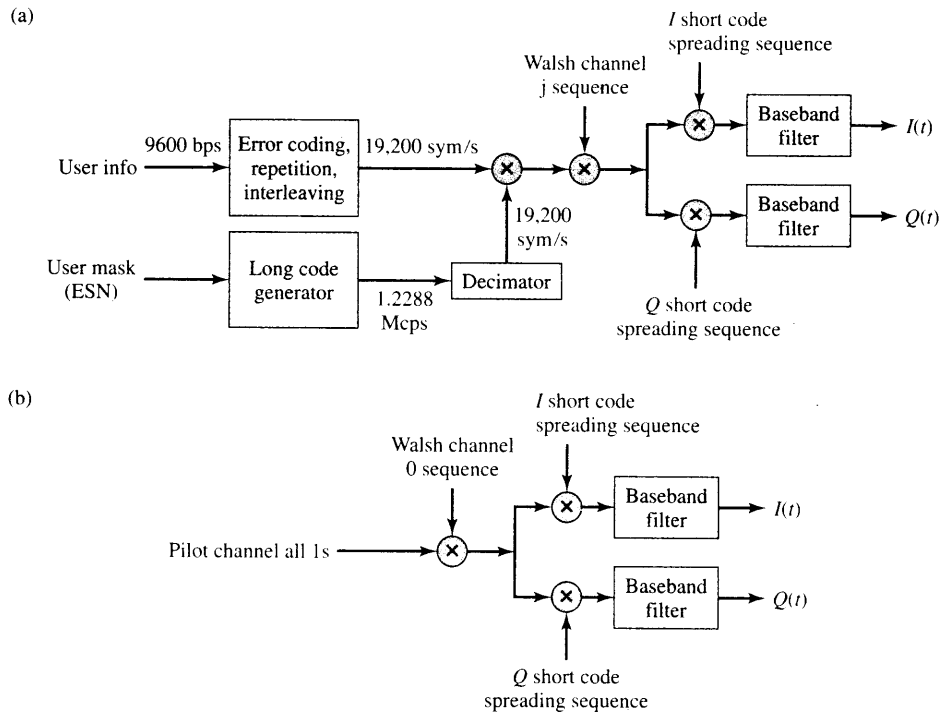


FIGURE 6.38 IS-95 modulator for forward channel.

mobile stations. The IS-95 supports a basic user information bit rate of 9600 bps. After error-correction coding and interleaving, a 19,200 bps binary sequence is produced that is converted into a symbol sequence of +1s and −1s. This 19,200 symbol/second stream is then multiplied by a 19,200 symbol/second stream that is derived by taking every 64th symbol from a *long code* pseudorandom sequence that depends on the user electronic serial number (ESN) and operates at 1.2288 Msymbol/second as shown in Figure 6.38a.[9] Each symbol in the resulting 19,200 symbol/second sequence is then multiplied by a 64-chip Walsh orthogonal sequence that corresponds to the given channel. Because the base station simultaneously handles the transmissions to all the base stations, it can synchronize its transmissions so that the signals to the different channels are orthogonal. This feature allows the receivers at the mobile stations to eliminate interference from the transmissions that are intended for other stations in the cell as discussed in Section 6.4.3. The chip rate of the resulting signal is 19,200 × 64 = 1.2288 Mchips/second. Note that in the forward channel the spreading is divided into two parts: the first part is provided by the channel-specific Walsh sequences; the second part is provided by the spreading sequence provided by the short code. The symbol sequence that results from the Walsh spreading is spread by using the short code and is then modulated using QPSK, a form of QAM with four constellation points.

[9]The long code repeats every $2^{42} - 1$ chips, which translates into every 41.4 days.