CHAPTER 7

DATA LINK CONTROL PROTOCOLS

"Great and enlightened one," said Ten-teh, as soon as his stupor was lifted, "has this person delivered his message competently, for his mind was still a seared vision of snow and sand and perchance his tongue has stumbled?"

"Bend your ears to the wall," replied the Emperor, "and be assured."

—Kai Lung's Golden Hours, Earnest Bramah

KEY POINTS

- Because of the possibility of transmission errors, and because the receiver of data may need to regulate the rate at which data arrive, synchronization and interfacing techniques are insufficient by themselves. It is necessary to impose a layer of control in each communicating device that provides functions such as flow control, error detection, and error control. This layer of control is known as a **data link control protocol**.
- **Flow control** enables a receiver to regulate the flow of data from a sender so that the receiver's buffers do not overflow.
- In a data link control protocol, **error control** is achieved by retransmission of damaged frames that have not been acknowledged or for which the other side requests a retransmission.
- High-level data link control (HDLC) is a widely used data link control protocol. It contains virtually all of the features found in other data link control protocols.

Our discussion so far has concerned *sending signals over a transmission link*. For effective digital data communications, much more is needed to control and manage the exchange. In this chapter, we shift our emphasis to that of *sending data over a data communications link*. To achieve the necessary control, a layer of logic is added above the physical layer discussed in Chapter 6; this logic is referred to as **data link control** or a **data link control protocol**. When a data link control protocol is used, the transmission medium between systems is referred to as a **data link**.

To see the need for data link control, we list some of the requirements and objectives for effective data communication between two directly connected transmitting-receiving stations:

- **Frame synchronization:** Data are sent in blocks called frames. The beginning and end of each frame must be recognizable. We briefly introduced this topic with the discussion of synchronous frames (Figure 6.2).
- **Flow control:** The sending station must not send frames at a rate faster than the receiving station can absorb them.
- **Error control:** Bit errors introduced by the transmission system should be corrected.
- **Addressing:** On a shared link, such as a local area network (LAN), the identity of the two stations involved in a transmission must be specified.

- **Control and data on same link:** It is usually not desirable to have a physically separate communications path for control information. Accordingly, the receiver must be able to distinguish control information from the data being transmitted.
- **Link management:** The initiation, maintenance, and termination of a sustained data exchange require a fair amount of coordination and cooperation among stations. Procedures for the management of this exchange are required.

None of these requirements is satisfied by the techniques described in Chapter 6. We shall see in this chapter that a data link protocol that satisfies these requirements is a rather complex affair. We begin by looking at two key mechanisms that are part of data link control: flow control and error control. Following this background we look at the most important example of a data link control protocol: HDLC (high-level data link control). This protocol is important for two reasons: First, it is a widely used standardized data link control protocol. Second, HDLC serves as a baseline from which virtually all other important data link control protocols are derived. Finally, an appendix to this chapter addresses some performance issues relating to data link control.

7.1 FLOW CONTROL

Flow control is a technique for assuring that a transmitting entity does not overwhelm a receiving entity with data. The receiving entity typically allocates a data buffer of some maximum length for a transfer. When data are received, the receiver must do a certain amount of processing before passing the data to the higher-level software. In the absence of flow control, the receiver's buffer may fill up and overflow while it is processing old data.

To begin, we examine mechanisms for flow control in the absence of errors. The model we will use is depicted in Figure 7.1a, which is a vertical-time sequence diagram. It has the advantages of showing time dependencies and illustrating the correct send-receive relationship. Each arrow represents a single frame transiting a data link between two stations. The data are sent in a sequence of frames, with each frame containing a portion of the data and some control information. The time it takes for a station to emit all of the bits of a frame onto the medium is the transmission time; this is proportional to the length of the frame. The propagation time is the time it takes for a bit to traverse the link between source and destination. For this section, we assume that all frames that are transmitted are successfully received; no frames are lost and none arrive with errors. Furthermore, frames arrive in the same order in which they are sent. However, each transmitted frame suffers an arbitrary and variable amount of delay before reception.[1]

[1]On a direct point-to-point link, the amount of delay is fixed rather than variable. However, a data link control protocol can be used over a network connection, such as a circuit-switched or ATM network, in which case the delay may be variable.

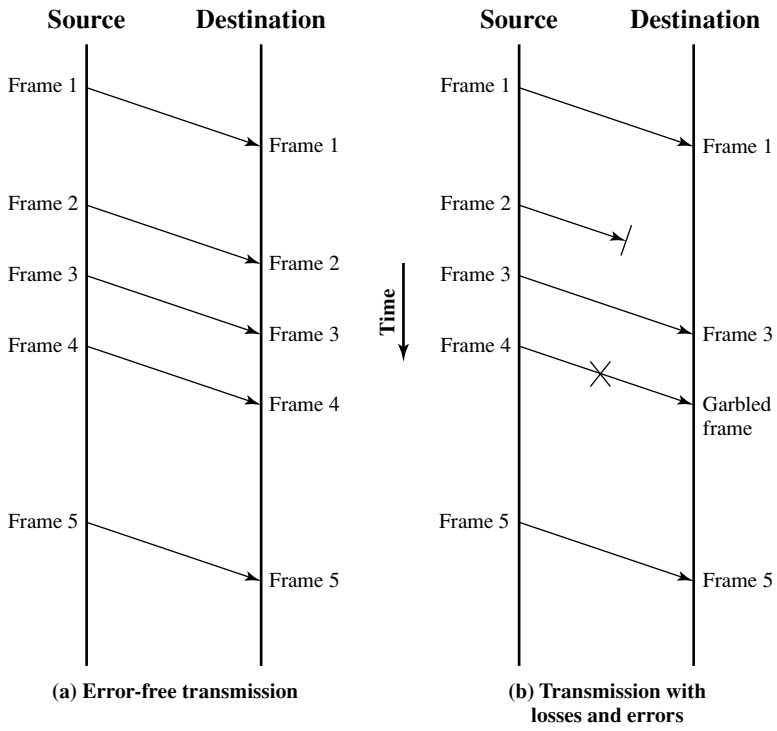

Figure 7.1 Model of Frame Transmission

Stop-and-Wait Flow Control

The simplest form of flow control, known as stop-and-wait flow control, works as follows. A source entity transmits a frame. After the destination entity receives the frame, it indicates its willingness to accept another frame by sending back an acknowledgment to the frame just received. The source must wait until it receives the acknowledgment before sending the next frame. The destination can thus stop the flow of data simply by withholding acknowledgment. This procedure works fine and, indeed, can hardly be improved upon when a message is sent in a few large frames. However, it is often the case that a source will break up a large block of data into smaller blocks and transmit the data in many frames. This is done for the following reasons:

- The buffer size of the receiver may be limited.
- The longer the transmission, the more likely that there will be an error, necessitating retransmission of the entire frame. With smaller frames, errors are detected sooner, and a smaller amount of data needs to be retransmitted.
- On a shared medium, such as a LAN, it is usually desirable not to permit one station to occupy the medium for an extended period, thus causing long delays at the other sending stations.

With the use of multiple frames for a single message, the stop-and-wait procedure may be inadequate. The essence of the problem is that only one frame at a time can be in transit. To explain we first define the **bit length of a link** as follows:

$$B = R \times \frac{d}{V} \tag{7.1}$$

where

B = length of the link in bits; this is the number of bits present on the link at an instance in time when a stream of bits fully occupies the link
R = data rate of the link, in bps
d = length, or distance, of the link in meters
V = velocity of propagation, in m/s

In situations where the bit length of the link is greater than the frame length, serious inefficiencies result. This is illustrated in Figure 7.2. In the figure, the transmission time (the time it takes for a station to transmit a frame) is normalized to one, and the propagation delay (the time it takes for a bit to travel from sender to receiver) is expressed as the variable a. Thus, we can express a as

$$a = \frac{B}{L} \tag{7.2}$$

where L is the number of bits in the frame (length of the frame in bits).

When a is less than 1, the propagation time is less than the transmission time. In this case, the frame is sufficiently long that the first bits of the frame have arrived at the destination before the source has completed the transmission of the frame.

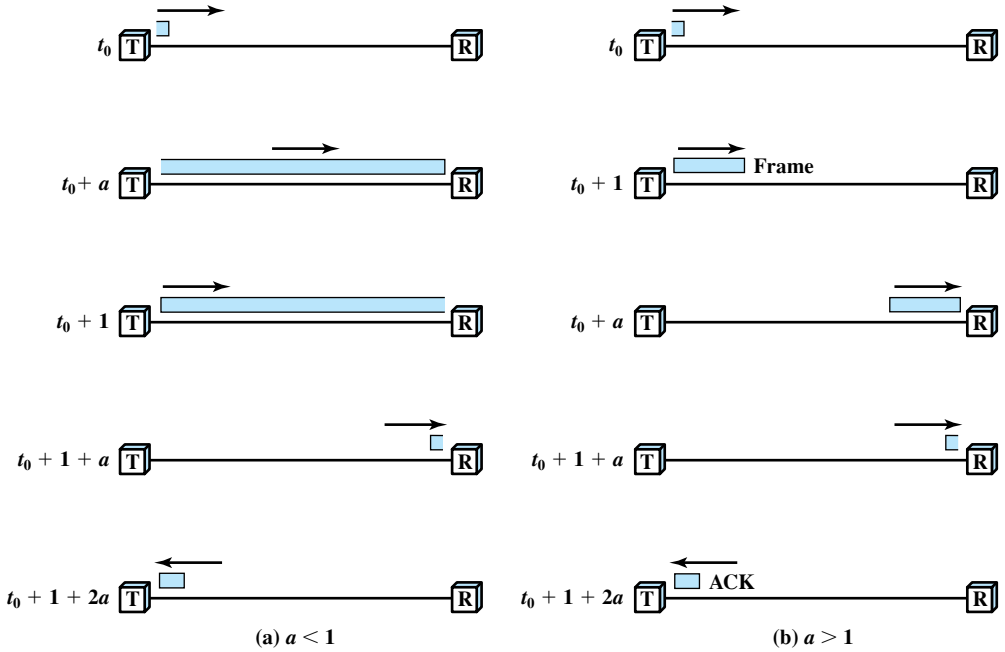

Figure 7.2 Stop-and-Wait Link Utilization (transmission time = 1; propagation time = a)

When a is greater than 1, the propagation time is greater than the transmission time. In this case, the sender completes transmission of the entire frame before the leading bits of that frame arrive at the receiver. Put another way, larger values of a are consistent with higher data rates and/or longer distances between stations. Appendix 7A discusses a and data link performance.

Both parts of Figure 7.2 (a and b) consist of a sequence of snapshots of the transmission process over time. In both cases, the first four snapshots show the process of transmitting a frame containing data, and the last snapshot shows the return of a small acknowledgment frame. Note that for $a > 1$, the line is always underutilized and even for $a < 1$, the line is inefficiently utilized. In essence, for very high data rates, for very long distances between sender and receiver, stop-and-wait flow control provides inefficient line utilization.

EXAMPLE 7.1 Consider a 200-m optical fiber link operating at 1 Gbps. The velocity of propagation of optical fiber is typically about 2×10^8 m/s. Using Equation (7.1), $B = (10^9 \times 200)/(2 \times 10^8) = 1000$ bits. Assume a frame of 1000 octets, or 8000 bits, is transmitted. Using Equation (7.2), $a = (1000/8000) = 0.125$. Using Figure 7.2a as a guide, assume transmission starts at time $t = 0$. After 1 μs (a normalized time of 0.125 frame times), the leading edge (first bit) of the frame has reached R, and the first 1000 bits of the frame are spread out across the link. At time $t = 8$ μs, the trailing edge (final bit) of the frame has just been emitted by T, and the final 1000 bits of the frame are spread out across the link. At $t = 9$ μs, the final bit of the frame arrives at R. R now sends back an ACK frame. If we assume the frame transmission time is negligible (very small ACK frame) and that the ACK is sent immediately, the ACK arrives at T at $t = 10$ μs. At this point, T can begin transmitting a new frame. The actual transmission time for the frame was 8 μs, but the total time to transmit the first frame and receive and ACK is 10 μs.

Now consider a 1-Mbps link between two ground stations that communicate via a satellite relay. A geosynchronous satellite has an altitude of roughly 36,000 km. Then $B = (10^6 \times 2 \times 36{,}000{,}000)/(3 \times 10^8) = 240{,}000$ bits. For a frame length of 8000 bits, $a = (240000/8000) = 30$. Using Figure 7.2b as a guide, we can work through the same steps as before. In this case, it takes 240 ms for the leading edge of the frame to arrive and an additional 8 ms for the entire frame to arrive. The ACK arrives back at T at $t = 488$ ms. The actual transmission time for the first frame was 8 ms, but the total time to transmit the first frame and receive an ACK is 488 ms.

Sliding-Window Flow Control

The essence of the problem described so far is that only one frame at a time can be in transit. In situations where the bit length of the link is greater than the frame length ($a > 1$), serious inefficiencies result. Efficiency can be greatly improved by allowing multiple frames to be in transit at the same time.

Let us examine how this might work for two stations, A and B, connected via a full-duplex link. Station B allocates buffer space for W frames. Thus, B can accept W frames, and A is allowed to send W frames without waiting for any acknowledgments. To keep track of which frames have been acknowledged, each is labeled with a sequence number. B acknowledges a frame by sending an acknowledgment that includes the sequence number of the next frame expected. This acknowledgment also implicitly announces that B is prepared to receive the next W frames, beginning with the number specified. This scheme can also be used to acknowledge multiple frames. For example, B could receive frames 2, 3, and 4 but withhold acknowledgment until frame 4 has arrived. By then returning an acknowledgment with sequence number 5, B acknowledges frames 2, 3, and 4 at one time. A maintains a list of sequence numbers that it is allowed to send, and B maintains a list of sequence numbers that it is prepared to receive. Each of these lists can be thought of as a *window* of frames. The operation is referred to as **sliding-window flow control**.

Several additional comments need to be made. Because the sequence number to be used occupies a field in the frame, it is limited to a range of values. For example, for a 3-bit field, the sequence number can range from 0 to 7. Accordingly, frames are numbered modulo 8; that is, after sequence number 7, the next number is 0. In general, for a k-bit field the range of sequence numbers is 0 through $2^k - 1$, and frames are numbered modulo 2^k. As will be shown subsequently, the maximum window size is $2^k - 1$.

Figure 7.3 is a useful way of depicting the sliding-window process. It assumes the use of a 3-bit sequence number, so that frames are numbered sequentially from

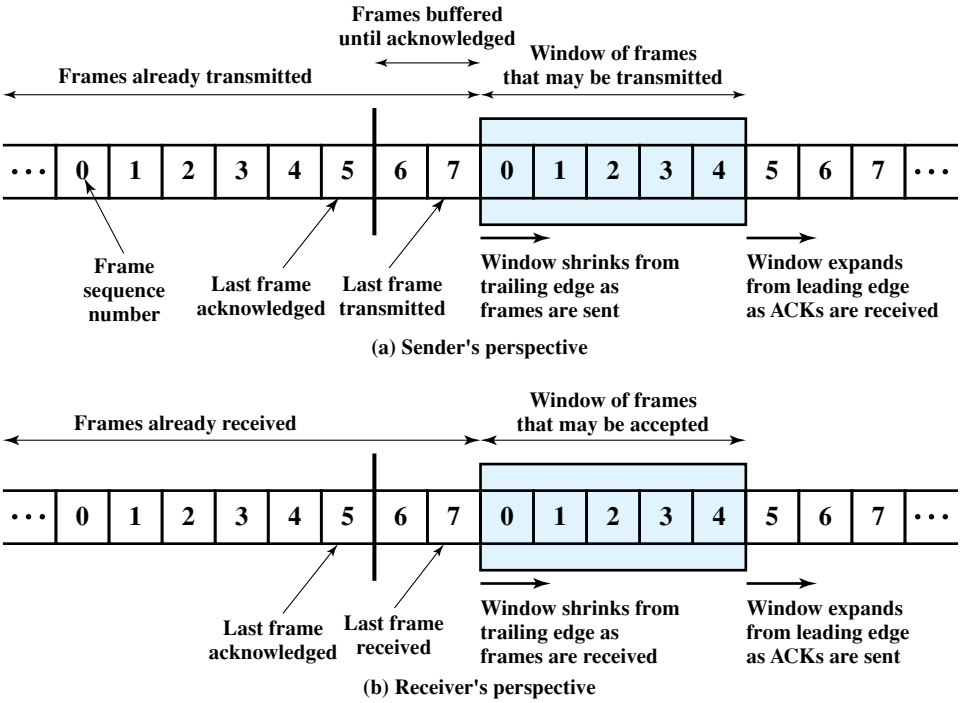

Figure 7.3 Sliding-Window Depiction

0 through 7, and then the same numbers are reused for subsequent frames. The shaded rectangle indicates the frames that may be sent; in this figure, the sender may transmit five frames, beginning with frame 0. Each time a frame is sent, the shaded window shrinks; each time an acknowledgment is received, the shaded window grows. Frames between the vertical bar and the shaded window have been sent but not yet acknowledged. As we shall see, the sender must buffer these frames in case they need to be retransmitted.

The window size need not be the maximum possible size for a given sequence number length. For example, using a 3-bit sequence number, a window size of 5 could be configured for the stations using the sliding-window flow control protocol.

EXAMPLE 7.2 An example is shown in Figure 7.4. The example assumes a 3-bit sequence number field and a maximum window size of seven frames. Initially, A and B have windows indicating that A may transmit seven frames, beginning with frame 0 (F0). After transmitting three frames (F0, F1, F2) without acknowledgment, A has shrunk its window to four frames and maintains a copy of the three transmitted frames. The window indicates that A may transmit four frames, beginning with frame number 3. B then transmits an RR (receive ready) 3, which means "I have received all frames up through frame number 2 and am ready to receive frame number 3; in fact, I am prepared to receive seven frames, beginning with frame number 3." With this acknowledgment, A is back up to permission to transmit seven frames, still beginning with frame 3; also A may discard the buffered frames that have now been acknowledged. A proceeds to transmit frames 3, 4, 5, and 6. B returns RR 4, which acknowledges F3, and allows transmission of F4 through the next instance of F2. By the time this RR reaches A, it has already transmitted F4, F5, and F6, and therefore A may only open its window to permit sending four frames beginning with F7.

The mechanism so far described provides a form of flow control: The receiver must only be able to accommodate seven frames beyond the one it has last acknowledged. Most data link control protocols also allow a station to cut off the flow of frames from the other side by sending a Receive Not Ready (RNR) message, which acknowledges former frames but forbids transfer of future frames. Thus, RNR 5 means "I have received all frames up through number 4 but am unable to accept any more at this time." At some subsequent point, the station must send a normal acknowledgment to reopen the window.

So far, we have discussed transmission in one direction only. If two stations exchange data, each needs to maintain two windows, one for transmit and one for receive, and each side needs to send the data and acknowledgments to the other. To provide efficient support for this requirement, a feature known as **piggybacking** is typically provided. Each **data frame** includes a field that holds the sequence number of that frame plus a field that holds the sequence number used for acknowledgment.

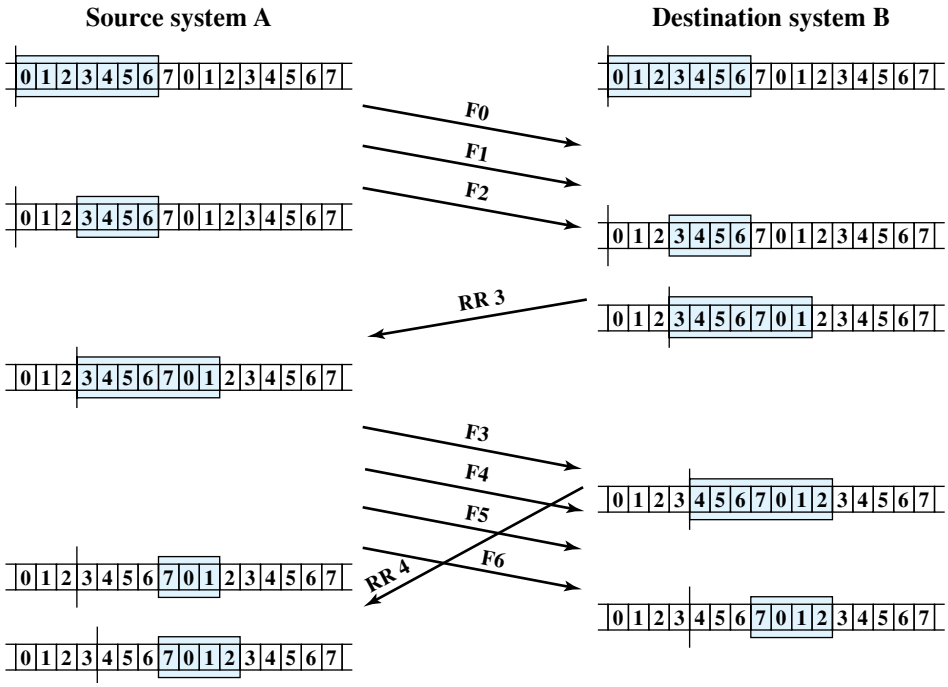

Figure 7.4 Example of a Sliding-Window Protocol

Thus, if a station has data to send and an acknowledgment to send, it sends both together in one frame, saving communication capacity. Of course, if a station has an acknowledgment but no data to send, it sends a separate **acknowledgment frame**, such as RR or RNR. If a station has data to send but no new acknowledgment to send, it must repeat the last acknowledgment sequence number that it sent. This is because the data frame includes a field for the acknowledgment number, and some value must be put into that field. When a station receives a duplicate acknowledgment, it simply ignores it.

Sliding-window flow control is potentially much more efficient than stop-and-wait flow control. The reason is that, with sliding-window flow control, the transmission link is treated as a pipeline that may be filled with frames in transit. In contrast, with stop-and-wait flow control, only one frame may be in the pipe at a time. Appendix 7A quantifies the improvement in efficiency.

EXAMPLE 7.3 Let us consider the use of sliding-window flow control for the two configurations of Example 7.1. As was calculated in Example 7.1, it takes 10 μs for an ACK to the first frame to be received. It takes 8 μs to transmit one frame, so the sender can transmit one frame and part of a second frame by the time the ACK to the first frame is received. Thus, a window size of 2 is adequate to

sender to transmit frames continuously, or a rate of one frame every 8 μs. With stop-and-wait, a rate of only one frame per 10 μs is possible.

For the satellite configuration, it takes 488 ms for an ACK to the first frame to be received. It takes 8 ms to transmit one frame, so the sender can transmit 61 frames by the time the ACK to the first frame is received. With a window field of 6 bits or more, the sender can transmit continuously, or a rate of one frame every 8 ms. If the window size is 7, using a 3-bit window field, then the sender can only send 7 frames and then must wait for an ACK before sending more. In this case, the sender can transmit at a rate of 7 frames per 488 ms, or about one frame every 70 ms. With stop-and-wait, a rate of only one frame per 488 ms is possible.

7.2 ERROR CONTROL

Error control refers to mechanisms to detect and correct errors that occur in the transmission of frames. The model that we will use, which covers the typical case, is illustrated in Figure 7.1b. As before, data are sent as a sequence of frames; frames arrive in the same order in which they are sent; and each transmitted frame suffers an arbitrary and potentially variable amount of delay before reception. In addition, we admit the possibility of two types of errors:

- **Lost frame:** A frame fails to arrive at the other side. For example, a noise burst may damage a frame to the extent that the receiver is not aware that a frame has been transmitted.
- **Damaged frame:** A recognizable frame does arrive, but some of the bits are in error (have been altered during transmission).

The most common techniques for error control are based on some or all of the following ingredients:

- **Error detection:** As discussed in the Chapter 6.
- **Positive acknowledgment:** The destination returns a positive acknowledgment to successfully received, error-free frames.
- **Retransmission after timeout:** The source retransmits a frame that has not been acknowledged after a predetermined amount of time.
- **Negative acknowledgment and retransmission:** The destination returns a negative acknowledgment to frames in which an error is detected. The source retransmits such frames.

Collectively, these mechanisms are all referred to as **automatic repeat request** (ARQ); the effect of ARQ is to turn an unreliable data link into a reliable one. Three versions of ARQ have been standardized:

- Stop-and-wait ARQ
- Go-back-N ARQ
- Selective-reject ARQ

All of these forms are based on the use of the flow control techniques discussed in Section 7.1. We examine each in turn.

Stop-and-Wait ARQ

Stop-and-wait ARQ is based on the stop-and-wait flow control technique outlined previously. The source station transmits a single frame and then must await an acknowledgment (ACK). No other data frames can be sent until the destination station's reply arrives at the source station.

Two sorts of errors could occur. First, the frame that arrives at the destination could be damaged. The receiver detects this by using the error-detection technique referred to earlier and simply discards the frame. To account for this possibility, the source station is equipped with a timer. After a frame is transmitted, the source station waits for an acknowledgment. If no acknowledgment is received by the time that the timer expires, then the same frame is sent again. Note that this method requires that the transmitter maintain a copy of a transmitted frame until an acknowledgment is received for that frame.

The second sort of error is a damaged acknowledgment. Consider the following situation. Station A sends a frame. The frame is received correctly by station B, which responds with an acknowledgment (ACK). The ACK is damaged in transit and is not recognizable by A, which will therefore time out and resend the same frame. This duplicate frame arrives and is accepted by B. B has therefore accepted two copies of the same frame as if they were separate. To avoid this problem, frames are alternately labeled with 0 or 1, and positive acknowledgments are of the form ACK0 and ACK1. In keeping with the sliding-window convention, an ACK0 acknowledges receipt of a frame numbered 1 and indicates that the receiver is ready for a frame numbered 0.

Figure 7.5 gives an example of the use of stop-and-wait ARQ, showing the transmission of a sequence of frames from source A to destination B.[2] The figure shows the two types of errors just described. The third frame transmitted by A is lost or damaged and therefore B does not return an ACK. A times out and retransmits the frame. Later, A transmits a frame labeled 1 but the ACK0 for that frame is lost. A times out and retransmits the same frame. When B receives two frames in a row with the same label, it discards the second frame but sends back an ACK0 to each.

The principal advantage of stop-and-wait ARQ is its simplicity. Its principal disadvantage, as discussed in Section 7.1, is that stop-and-wait is an inefficient mechanism. The sliding-window flow control technique can be adapted to provide more efficient line use; in this context, it is sometimes referred to as *continuous ARQ*.

Go-Back-N ARQ

The form of error control based on sliding-window flow control that is most commonly used is called go-back-N ARQ. In this method, a station may send a series of frames sequentially numbered modulo some maximum value. The number of unacknowledged frames outstanding is determined by window size, using the

[2]This figure indicates the time required to transmit a frame. For simplicity, other figures in this chapter do not show this time.

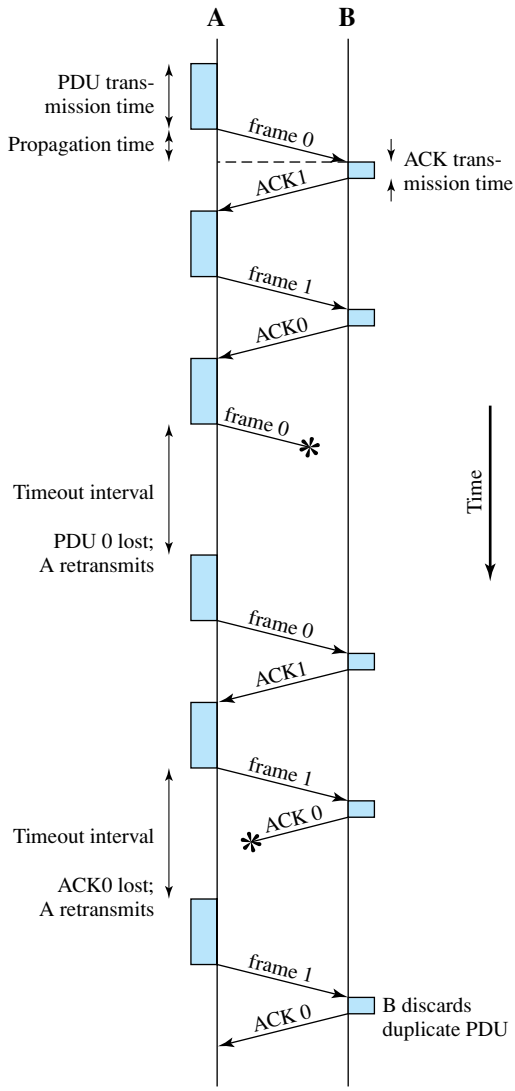

Figure 7.5 Stop-and-Wait ARQ

sliding-window flow control technique. While no errors occur, the destination will acknowledge incoming frames as usual (RR = receive ready, or piggybacked acknowledgment). If the destination station detects an error in a frame, it may send a negative acknowledgment (REJ = reject) for that frame, as explained in the following rules. The destination station will discard that frame and all future incoming frames until the frame in error is correctly received. Thus, the source station, when it receives a REJ, must retransmit the frame in error plus all succeeding frames that were transmitted in the interim.

Suppose that station A is sending frames to station B. After each transmission, A sets an acknowledgment timer for the frame just transmitted. Suppose that B has

previously successfully received frame $(i - 1)$ and A has just transmitted frame i. The go-back-N technique takes into account the following contingencies:

1. **Damaged frame.** If the received frame is invalid (i.e., B detects an error, or the frame is so damaged that B does not even perceive that it has received a frame), B discards the frame and takes no further action as the result of that frame. There are two subcases:
   - **(a)** Within a reasonable period of time, A subsequently sends frame $(i + 1)$. B receives frame $(i + 1)$ out of order and sends a REJ i. A must retransmit frame i and all subsequent frames.
   - **(b)** A does not soon send additional frames. B receives nothing and returns neither an RR nor a REJ. When A's timer expires, it transmits an RR frame that includes a bit known as the P bit, which is set to 1. B interprets the RR frame with a P bit of 1 as a command that must be acknowledged by sending an RR indicating the next frame that it expects, which is frame i. When A receives the RR, it retransmits frame i. Alternatively, A could just retransmit frame i when its timer expires.
2. **Damaged RR.** There are two subcases:
   - **(a)** B receives frame i and sends RR $(i + 1)$, which suffers an error in transit. Because acknowledgments are cumulative (e.g., RR 6 means that all frames through 5 are acknowledged), it may be that A will receive a subsequent RR to a subsequent frame and that it will arrive before the timer associated with frame i expires.
   - **(b)** If A's timer expires, it transmits an RR command as in Case 1b. It sets another timer, called the P-bit timer. If B fails to respond to the RR command, or if its response suffers an error in transit, then A's P-bit timer will expire. At this point, A will try again by issuing a new RR command and restarting the P-bit timer. This procedure is tried for a number of iterations. If A fails to obtain an acknowledgment after some maximum number of attempts, it initiates a reset procedure.
3. **Damaged REJ.** If a REJ is lost, this is equivalent to Case 1b.

EXAMPLE 7.4 Figure 7.6a is an example of the frame flow for go-back-N ARQ. Because of the propagation delay on the line, by the time that an acknowledgment (positive or negative) arrives back at the sending station, it has already sent at least one additional frame beyond the one being acknowledged. In this example, frame 4 is damaged. Frames 5 and 6 are received out of order and are discarded by B. When frame 5 arrives, B immediately sends a REJ 4. When the REJ to frame 4 is received, not only frame 4 but frames 5 and 6 must be retransmitted. Note that the transmitter must keep a copy of all unacknowledged frames. Figure 7.6a also shows an example of retransmission after timeout. No acknowledgment is received for frame 5 within the timeout period, so A issues an RR to determine the status of B.

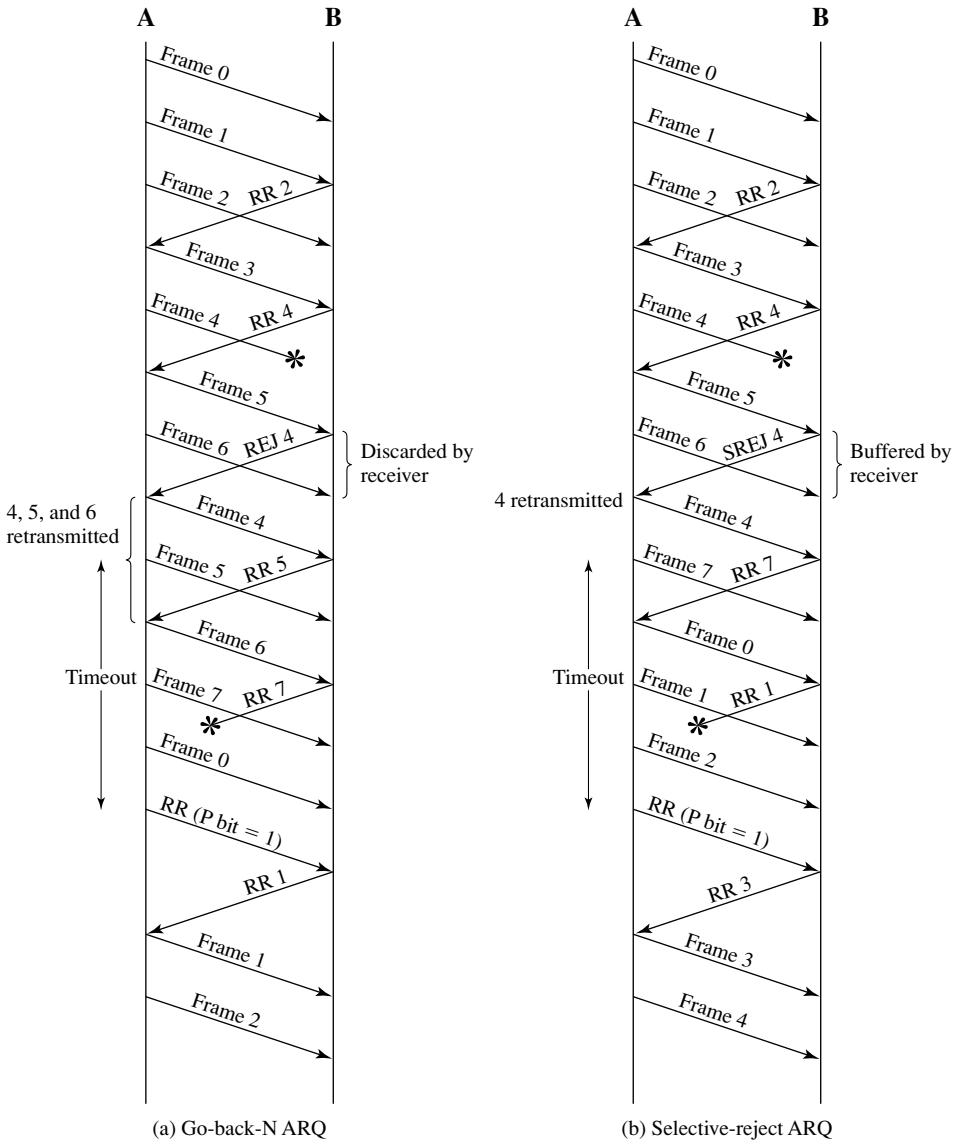

Figure 7.6 Sliding-Window ARQ Protocols

In Section 7.1, we mentioned that for a k-bit sequence number field, which provides a sequence number range of 2^k, the maximum window size is limited to $2^k - 1$. This has to do with the interaction between error control and acknowledgment. Consider that if data are being exchanged in both directions, station B must send piggybacked acknowledgments to station A's frames in the data frames being transmitted by B, even if the acknowledgment has already been sent. As we have mentioned, this is because B must put some number in the acknowledgment field of its data frame. As

an example, assume a 3-bit sequence number (sequence number space = 8). Suppose a station sends frame 0 and gets back an RR 1 and then sends frames 1, 2, 3, 4, 5, 6, 7, 0 and gets another RR 1. This could mean that all eight frames were received correctly and the RR 1 is a cumulative acknowledgment. It could also mean that all eight frames were damaged or lost in transit, and the receiving station is repeating its previous RR 1. The problem is avoided if the maximum window size is limited to 7 ($2^3 - 1$).

Selective-Reject ARQ

With selective-reject ARQ, the only frames retransmitted are those that receive a negative acknowledgment, in this case called SREJ, or those that time out.

EXAMPLE 7.5 Figure 7.6b illustrates this scheme. When frame 5 is received out of order, B sends a SREJ 4, indicating that frame 4 has not been received. However, B continues to accept incoming frames and buffers them until a valid frame 4 is received. At that point, B can place the frames in the proper order for delivery to higher-layer software.

Selective reject would appear to be more efficient than go-back-N, because it minimizes the amount of retransmission. On the other hand, the receiver must maintain a buffer large enough to save post-SREJ frames until the frame in error is retransmitted and must contain logic for reinserting that frame in the proper sequence. The transmitter, too, requires more complex logic to be able to send a frame out of sequence. Because of such complications, select-reject ARQ is much less widely used than go-back-N ARQ. Selective reject is a useful choice for a satellite link because of the long propagation delay involved.

The window size limitation is more restrictive for selective-reject than for go-back-N. Consider the case of a 3-bit sequence number size for selective-reject. Allow a window size of seven, and consider the following scenario [TANE03]:

1. Station A sends frames 0 through 6 to station B.
2. Station B receives all seven frames and cumulatively acknowledges with RR 7.
3. Because of a noise burst, the RR 7 is lost.
4. A times out and retransmits frame 0.
5. B has already advanced its receive window to accept frames 7, 0, 1, 2, 3, 4, and 5. Thus it assumes that frame 7 has been lost and that this is a new frame 0, which it accepts.

The problem with the foregoing scenario is that there is an overlap between the sending and receiving windows. To overcome the problem, the maximum window size should be no more than half the range of sequence numbers. In the preceding scenario, if only four unacknowledged frames may be outstanding, no confusion can result. In general, for a k-bit sequence number field, which provides a sequence number range of 2^k, the maximum window size is limited to 2^{k-1}.

7.3 HIGH-LEVEL DATA LINK CONTROL (HDLC)

The most important data link control protocol is HDLC (ISO 3009, ISO 4335). Not only is HDLC widely used, but it is the basis for many other important data link control protocols, which use the same or similar formats and the same mechanisms as employed in HDLC.

Basic Characteristics

To satisfy a variety of applications, HDLC defines three types of stations, two link configurations, and three data transfer modes of operation. The three station types are

- **Primary station:** Responsible for controlling the operation of the link. Frames issued by the primary are called commands.
- **Secondary station:** Operates under the control of the primary station. Frames issued by a secondary are called responses. The primary maintains a separate logical link with each secondary station on the line.
- **Combined station:** Combines the features of primary and secondary. A combined station may issue both commands and responses.

The two link configurations are

- **Unbalanced configuration:** Consists of one primary and one or more secondary stations and supports both full-duplex and half-duplex transmission.
- **Balanced configuration:** Consists of two combined stations and supports both full-duplex and half-duplex transmission.

The three data transfer modes are

- **Normal response mode (NRM):** Used with an unbalanced configuration. The primary may initiate data transfer to a secondary, but a secondary may only transmit data in response to a command from the primary.
- **Asynchronous balanced mode (ABM):** Used with a balanced configuration. Either combined station may initiate transmission without receiving permission from the other combined station.
- **Asynchronous response mode (ARM):** Used with an unbalanced configuration. The secondary may initiate transmission without explicit permission of the primary. The primary still retains responsibility for the line, including initialization, error recovery, and logical disconnection.

NRM is used on multidrop lines, in which a number of terminals are connected to a host computer. The computer polls each terminal for input. NRM is also sometimes used on point-to-point links, particularly if the link connects a terminal or other peripheral to a computer. ABM is the most widely used of the three modes; it makes more efficient use of a full-duplex point-to-point link because there is no polling overhead. ARM is rarely used; it is applicable to some special situations in which a secondary may need to initiate transmission.

Frame Structure

HDLC uses synchronous transmission. All transmissions are in the form of frames, and a single frame format suffices for all types of data and control exchanges.

Figure 7.7 depicts the structure of the HDLC frame. The flag, address, and control fields that precede the information field are known as a **header**. The FCS and flag fields following the data field are referred to as a **trailer**.

Flag Fields Flag fields delimit the frame at both ends with the unique pattern 01111110. A single flag may be used as the closing flag for one frame and the opening flag for the next. On both sides of the user-network interface, receivers are continuously hunting for the flag sequence to synchronize on the start of a frame. While receiving a frame, a station continues to hunt for that sequence to determine the end of the frame. Because the protocol allows the presence of arbitrary bit patterns (i.e., there are no restrictions on the content of the various fields imposed by the link protocol), there is no assurance that the pattern 01111110 will not appear somewhere inside the frame, thus destroying synchronization. To avoid this problem, a procedure known as *bit stuffing* is used. For all bits between the starting and ending flags, the transmitter inserts an extra 0 bit after each occurrence of five 1s in the frame. After

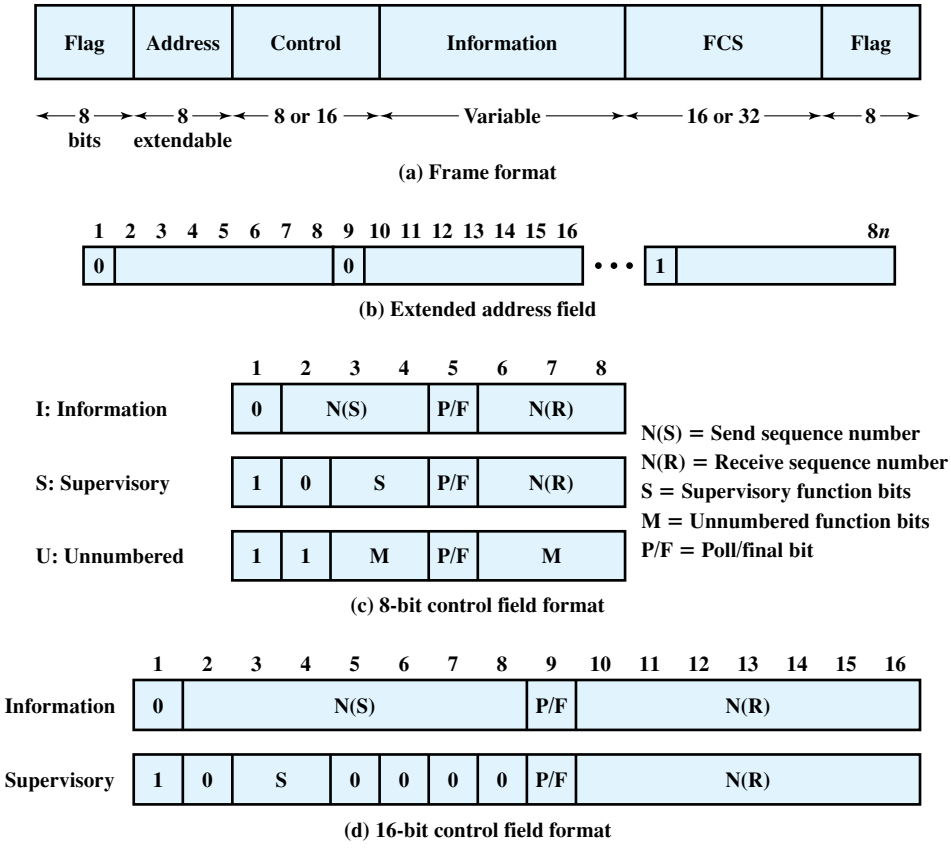

Figure 7.7

Original pattern:

111111111111011111101111110

After bit-stuffing:

111110111110110111110101111010

Figure 7.8 Bit Stuffing

detecting a starting flag, the receiver monitors the bit stream. When a pattern of five 1s appears, the sixth bit is examined. If this bit is 0, it is deleted. If the sixth bit is a 1 and the seventh bit is a 0, the combination is accepted as a flag. If the sixth and seventh bits are both 1, the sender is indicating an abort condition.

With the use of bit stuffing, arbitrary bit patterns can be inserted into the data field of the frame. This property is known as **data transparency**.

Figure 7.8 shows an example of bit stuffing. Note that in the first two cases, the extra 0 is not strictly necessary for avoiding a flag pattern but is necessary for the operation of the algorithm.

Address Field The address field identifies the secondary station that transmitted or is to receive the frame. This field is not needed for point-to-point links but is always included for the sake of uniformity. The address field is usually 8 bits long but, by prior agreement, an extended format may be used in which the actual address length is a multiple of 7 bits. The leftmost bit of each octet is 1 or 0 according as it is or is not the last octet of the address field. The remaining 7 bits of each octet form part of the address. The single-octet address of 11111111 is interpreted as the all-stations address in both basic and extended formats. It is used to allow the primary to broadcast a frame for reception by all secondaries.

Control Field HDLC defines three types of frames, each with a different control field format. **Information frames** (I-frames) carry the data to be transmitted for the user (the logic above HDLC that is using HDLC). Additionally, flow and error control data, using the ARQ mechanism, are piggybacked on an information frame. **Supervisory frames** (S-frames) provide the ARQ mechanism when piggybacking is not used. **Unnumbered frames** (U-frames) provide supplemental link control functions. The first one or two bits of the control field serves to identify the frame type. The remaining bit positions are organized into subfields as indicated in Figures 7.7c and d. Their use is explained in the discussion of HDLC operation later in this chapter.

All of the control field formats contain the poll/final (P/F) bit. Its use depends on context. Typically, in command frames, it is referred to as the P bit and is set to 1 to solicit (poll) a response frame from the peer HDLC entity. In response frames, it is referred to as the F bit and is set to 1 to indicate the response frame transmitted as a result of a soliciting command.

Note that the basic control field for S- and I-frames uses 3-bit sequence numbers. With the appropriate set-mode command, an extended control field can be used for S- and I-frames that employs 7-bit sequence numbers. U-frames always contain an 8-bit control field.

Information Field The information field is present only in I-frames and some U-frames. The field can contain any sequence of bits but must consist of an integral

number of octets. The length of the information field is variable up to some system-defined maximum.

Frame Check Sequence Field The frame check sequence (FCS) is an error-detecting code calculated from the remaining bits of the frame, exclusive of flags. The normal code is the 16-bit CRC-CCITT defined in Section 6.3. An optional 32-bit FCS, using CRC-32, may be employed if the frame length or the line reliability dictates this choice.

Operation

HDLC operation consists of the exchange of I-frames, S-frames, and U-frames between two stations. The various commands and responses defined for these frame types are listed in Table 7.1. In describing HDLC operation, we will discuss these three types of frames.

Table 7.1 HDLC Commands and Responses

Name	Command/ Response	Description
Information (I)	C/R	Exchange user data
Supervisory (S)		
Receive ready (RR)	C/R	Positive acknowledgment; ready to receive I-frame
Receive not ready (RNR)	C/R	Positive acknowledgment; not ready to receive
Reject (REJ)	C/R	Negative acknowledgment; go back N
Selective reject (SREJ)	C/R	Negative acknowledgment; selective reject
Unnumbered (U)		
Set normal response/extended mode (SNRM/SNRME)	C	Set mode; extended = 7-bit sequence numbers
Set asynchronous response/extended mode (SARM/SARME)	C	Set mode; extended = 7-bit sequence numbers
Set asynchronous balanced/extended mode (SABM, SABME)	C	Set mode; extended = 7-bit sequence numbers
Set initialization mode (SIM)	C	Initialize link control functions in addressed station
Disconnect (DISC)	C	Terminate logical link connection
Unnumbered Acknowledgment (UA)	R	Acknowledge acceptance of one of the set-mode commands
Disconnected mode (DM)	R	Responder is in disconnected mode
Request disconnect (RD)	R	Request for DISC command
Request initialization mode (RIM)	R	Initialization needed; request for SIM command
Unnumbered information (UI)	C/R	Used to exchange control information
Unnumbered poll (UP)	C	Used to solicit control information
Reset (RSET)	C	Used for recovery; resets N(R), N(S)
Exchange identification (XID)	C/R	Used to request/report status
Test (TEST)	C/R	Exchange identical information fields for testing
Frame reject (FRMR)	R	Report receipt of unacceptable frame

The operation of HDLC involves three phases. First, one side or another initializes the data link so that frames may be exchanged in an orderly fashion. During this phase, the options that are to be used are agreed upon. After initialization, the two sides exchange user data and the control information to exercise flow and error control. Finally, one of the two sides signals the termination of the operation.

Initialization Either side may request initialization by issuing one of the six set-mode commands. This command serves three purposes:

1. It signals the other side that initialization is requested.
2. It specifies which of the three modes (NRM, ABM, ARM) is requested.
3. It specifies whether 3- or 7-bit sequence numbers are to be used.

If the other side accepts this request, then the HDLC module on that end transmits an unnumbered acknowledged (UA) frame back to the initiating side. If the request is rejected, then a disconnected mode (DM) frame is sent.

Data Transfer When the initialization has been requested and accepted, then a logical connection is established. Both sides may begin to send user data in I-frames, starting with sequence number 0. The N(S) and N(R) fields of the I-frame are sequence numbers that support flow control and error control. An HDLC module sending a sequence of I-frames will number them sequentially, modulo 8 or 128, depending on whether 3- or 7-bit sequence numbers are used, and place the sequence number in N(S). N(R) is the acknowledgment for I-frames received; it enables the HDLC module to indicate which number I-frame it expects to receive next.

S-frames are also used for flow control and error control. The receive ready (RR) frame acknowledges the last I-frame received by indicating the next I-frame expected. The RR is used when there is no reverse user data traffic (I-frames) to carry an acknowledgment. Receive not ready (RNR) acknowledges an I-frame, as with RR, but also asks the peer entity to suspend transmission of I-frames. When the entity that issued RNR is again ready, it sends an RR. REJ initiates the go-back-N ARQ. It indicates that the last I-frame received has been rejected and that retransmission of all I-frames beginning with number N(R) is required. Selective reject (SREJ) is used to request retransmission of just a single frame.

Disconnect Either HDLC module can initiate a disconnect, either on its own initiative if there is some sort of fault, or at the request of its higher-layer user. HDLC issues a disconnect by sending a disconnect (DISC) frame. The remote entity must accept the disconnect by replying with a UA and informing its layer 3 user that the connection has been terminated. Any outstanding unacknowledged I-frames may be lost, and their recovery is the responsibility of higher layers.

Examples of Operation To better understand HDLC operation, several examples are presented in Figure 7.9. In the example diagrams, each arrow includes a legend that specifies the frame name, the setting of the P/F bit, and, where appropriate, the values of N(R) and N(S). The setting of the P or F bit is 1 if the designation is present and 0 if absent.

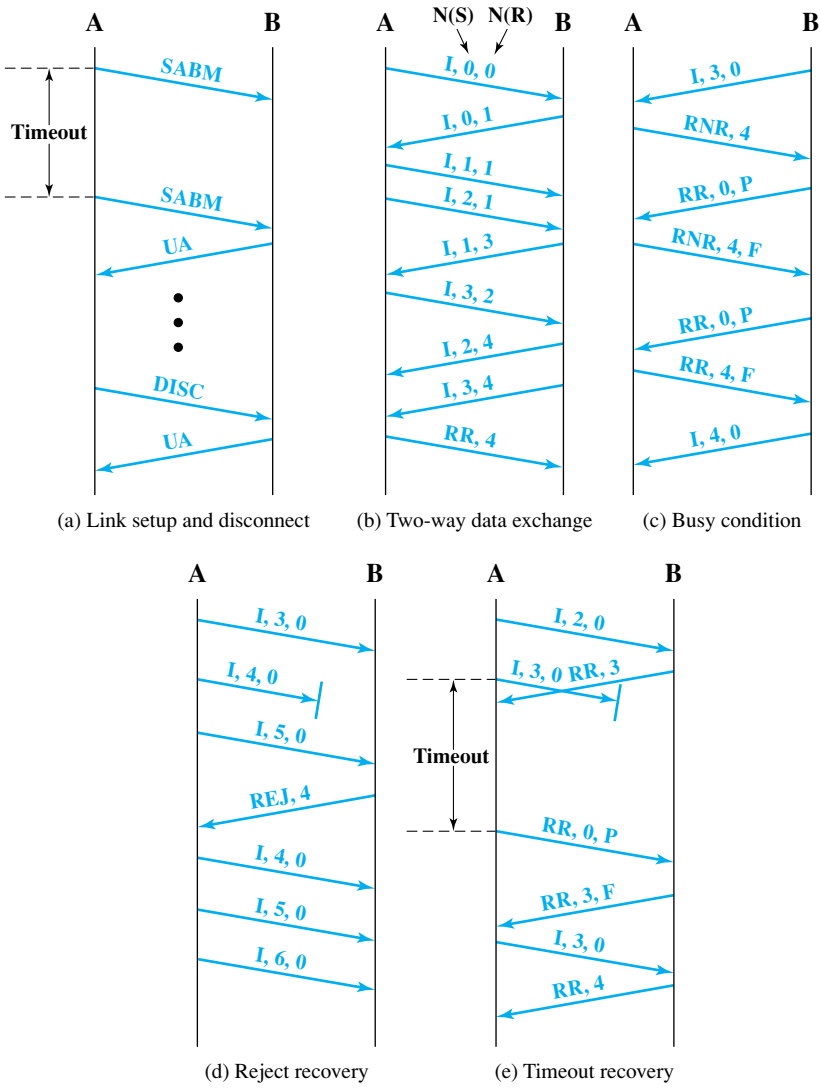

Figure 7.9 Examples of HDLC Operation

Figure 7.9a shows the frames involved in link setup and disconnect. The HDLC protocol entity for one side issues an SABM command to the other side and starts a timer. The other side, upon receiving the SABM, returns a UA response and sets local variables and counters to their initial values. The initiating entity receives the UA response, sets its variables and counters, and stops the timer. The logical connection is now active, and both sides may begin transmitting frames. Should the timer expire without a response to an SABM, the originator will repeat the SABM, as illustrated. This would be repeated until a UA or DM is received or until, after a given number of tries, the entity attempting initiation gives up and reports failure to a management entity. In such a case, higher-layer intervention is necessary. The same

figure (Figure 7.9a) shows the disconnect procedure. One side issues a DISC command, and the other responds with a UA response.

Figure 7.9b illustrates the full-duplex exchange of I-frames. When an entity sends a number of I-frames in a row with no incoming data, then the receive sequence number is simply repeated (e.g., I,1,1; I,2.1 in the A-to-B direction). When an entity receives a number of I-frames in a row with no outgoing frames, then the receive sequence number in the next outgoing frame must reflect the cumulative activity (e.g., I,1,3 in the B-to-A direction). Note that, in addition to I-frames, data exchange may involve supervisory frames.

Figure 7.9c shows an operation involving a busy condition. Such a condition may arise because an HDLC entity is not able to process I-frames as fast as they are arriving, or the intended user is not able to accept data as fast as they arrive in I-frames. In either case, the entity's receive buffer fills up and it must halt the incoming flow of I-frames, using an RNR command. In this example, A issues an RNR, which requires B to halt transmission of I-frames. The station receiving the RNR will usually poll the busy station at some periodic interval by sending an RR with the P bit set. This requires the other side to respond with either an RR or an RNR. When the busy condition has cleared, A returns an RR, and I-frame transmission from B can resume.

An example of error recovery using the REJ command is shown in Figure 7.9d. In this example, A transmits I-frames numbered 3, 4, and 5. Number 4 suffers an error and is lost. When B receives I-frame number 5, it discards this frame because it is out of order and sends an REJ with an N(R) of 4. This causes A to initiate retransmission of I-frames previously sent, beginning with frame 4. A may continue to send additional frames after the retransmitted frames.

An example of error recovery using a timeout is shown in Figure 7.9e. In this example, A transmits I-frame number 3 as the last in a sequence of I-frames. The frame suffers an error. B detects the error and discards it. However, B cannot send an REJ, because there is no way to know if this was an I-frame. If an error is detected in a frame, all of the bits of that frame are suspect, and the receiver has no way to act upon it. A, however, would have started a timer as the frame was transmitted. This timer has a duration long enough to span the expected response time. When the timer expires, A initiates recovery action. This is usually done by polling the other side with an RR command with the P bit set, to determine the status of the other side. Because the poll demands a response, the entity will receive a frame containing an N(R) field and be able to proceed. In this case, the response indicates that frame 3 was lost, which A retransmits.

These examples are not exhaustive. However, they should give the reader a good feel for the behavior of HDLC.

7.4 RECOMMENDED READING

An excellent and very detailed treatment of flow control and error control is to be found in [BERT92]. [FIOR95] points out some of the real-world reliability problems with HDLC.

There is a large body of literature on the performance of ARQ link control protocols. Three classic papers, well worth reading, are [BENE64], [KONH80], and [BUX80]. A readable survey with simplified performance results is [LIN84]. A more recent analysis is [ZORZ96]. Two books with good coverage of link-level performance are [SPRA91] and [WALR98].

[KLEI92] and [KLEI93] are two key papers that look at the implications of gigabit data rates on performance.

BENE64 Benice, R. "An Analysis of Retransmission Systems." *IEEE Transactions on Communication Technology*, December 1964.

BERT92 Bertsekas, D., and Gallager, R. *Data Networks.* Englewood Cliffs, NJ: Prentice Hall, 1992.

BUX80 Bux, W.; Kummerle, K.; and Truong, H. "Balanced HDLC Procedures: A Performance Analysis." *IEEE Transactions on Communications*, November 1980.

FIOR95 Fiorini, D.; Chiani, M.; Tralli, V.; and Salati, C. "Can We Trust HDLC?" *ACM Computer Communications Review*, October 1995.

KLEI92 Kleinrock, L. "The Latency/Bandwidth Tradeoff in Gigabit Networks." *IEEE Communications Magazine*, April 1992.

KLEI93 Kleinrock, L. "On the Modeling and Analysis of Computer Networks." *Proceedings of the IEEE*, August 1993.

KONH80 Konheim, A. "A Queuing Analysis of Two ARQ Protocols." *IEEE Transactions on Communications*, July 1980.

LIN84 Lin, S.; Costello, D.; and Miller, M. "Automatic-Repeat-Request Error-Control Schemes." *IEEE Communications Magazine*, December 1984.

SPRA91 Spragins, J.; Hammond, J.; and Pawlikowski, K. *Telecommunications: Protocols and Design.* Reading, MA: Addison-Wesley, 1991.

WALR98 Walrand, J. *Communication Networks: A First Course.* New York: McGraw-Hill, 1998.

ZORZ96 Zorzi, M., and Rao, R. "On the Use of Renewal Theory in the Analysis of ARQ Protocols." *IEEE Transactions on Communications*, September 1996.

7.5 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

automatic repeat request (ARQ)	error control	high-level data link control (HDLC)
acknowledgment frame	flag field	piggybacking
data frame	flow control	selective-reject ARQ
data link	frame	sliding-window flow control
data link control protocol	frame synchronization	stop-and-wait ARQ
data transparency	go-back-N ARQ	stop-and-wait flow control
	header	trailer

Review Questions

7.1. List and briefly define some of the requirements for effective communications over a data link.

7.2. Define *flow control.*

7.3. Describe stop-and-wait flow control.

7.4. What are reasons for breaking a long data transmission up into a number of frames?

7.5. Describe sliding-window flow control.

7.6. What is the advantage of sliding-window flow control compared to stop-and-wait flow control?

7.7. What is piggybacking?

7.8. Define *error control.*
7.9. List common ingredients for error control for a link control protocol.
7.10. Describe automatic repeat request (ARQ).
7.11. List and briefly define three versions of ARQ.
7.12. What are the station types supported by HDLC? Describe each.
7.13. What are the transfer modes supported by HDLC? Describe each.
7.14. What is the purpose of the flag field?
7.15. Define *data transparency.*
7.16. What are the three frame types supported by HDLC? Describe each.

Problems

7.1 Consider a half-duplex point-to-point link using a stop-and-wait scheme, in which a series of messages is sent, with each message segmented into a number of frames. Ignore errors and frame overhead.

a. What is the effect on line utilization of increasing the message size so that fewer messages will be required? Other factors remain constant.
b. What is the effect on line utilization of increasing the number of frames for a constant message size?
c. What is the effect on line utilization of increasing frame size?

7.2 The number of bits on a transmission line that are in the process of actively being transmitted (i.e., the number of bits that have been transmitted but have not yet been received) is referred to as the *bit length* of the line. Plot the line distance versus the transmission speed for a bit length of 1000 bits. Assume a propagation velocity of 2×10^8 m/s.

7.3 A channel has a data rate of 4 kbps and a propagation delay of 20 ms. For what range of frame sizes does stop-and-wait give an efficiency of at least 50%?

7.4 Consider the use of 1000-bit frames on a 1-Mbps satellite channel with a 270-ms delay. What is the maximum link utilization for

a. Stop-and-wait flow control?
b. Continuous flow control with a window size of 7?
c. Continuous flow control with a window size of 127?
d. Continuous flow control with a window size of 255?

7.5 In Figure 7.10 frames are generated at node A and sent to node C through node B. Determine the minimum data rate required between nodes B and C so that the buffers of node B are not flooded, based on the following:

- The data rate between A and B is 100 kbps.
- The propagation delay is 5 μs/km for both lines.
- There are full duplex lines between the nodes.
- All data frames are 1000 bits long; ACK frames are separate frames of negligible length.
- Between A and B, a sliding-window protocol with a window size of 3 is used.
- Between B and C, stop-and-wait is used.
- There are no errors.

Hint: In order not to flood the buffers of B, the average number of frames entering and leaving B must be the same over a long interval.

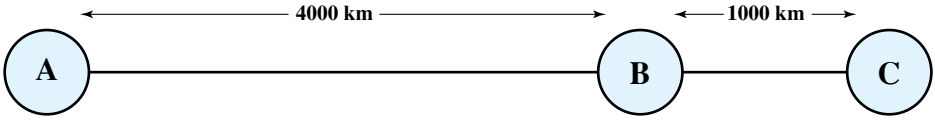

Figure 7.10 Configuration for Problem 7.4

7.6 A channel has a data rate of R bps and a propagation delay of t s/km. The distance between the sending and receiving nodes is L kilometers. Nodes exchange fixed-size frames of B bits. Find a formula that gives the minimum sequence field size of the frame as a function of R, t, B, and L. (considering maximum utilization). Assume that ACK frames are negligible in size and the processing at the nodes is instantaneous.

7.7 No mention was made of reject (REJ) frames in the stop-and-wait ARQ discussion. Why is it not necessary to have REJ0 and REJ1 for stop-and-wait ARQ?

7.8 Suppose that a selective-reject ARQ is used where $W = 4$. Show, by example, that a 3-bit sequence number is needed.

7.9 Using the same assumptions that are used for Figure 7.13 in Appendix 7A, plot line utilization as a function of P, the probability that a single frame is in error for the following error-control techniques:

a. Stop-and-wait
b. Go-back-N with $W = 7$
c. Go-back-N with $W = 127$
d. Selective reject with $W = 7$
e. Selective reject with $W = 127$

Do all of the preceding for the following values of a: 0.1, 1, 10, 100. Draw conclusions about which technique is appropriate for various ranges of a.

7.10 Two neighboring nodes (A and B) use a sliding-window protocol with a 3-bit sequence number. As the ARQ mechanism, go-back-N is used with a window size of 4. Assuming A is transmitting and B is receiving, show the window positions for the following succession of events:

a. Before A sends any frames
b. After A sends frames 0, 1, 2 and receives acknowledgment from B for 0 and 1
c. After A sends frames 3, 4, and 5 and B acknowledges 4 and the ACK is received by A

7.11 Out-of-sequence acknowledgment cannot be used for selective-reject ARQ. That is, if frame i is rejected by station X, all subsequent I-frames and RR frames sent by X must have N(R) $= i$ until frame i is successfully received, even if other frames with N(S) $> i$ are successfully received in the meantime. One possible refinement is the following: N(R) $= j$ in an I-frame or an RR frame is interpreted to mean that frame $j - 1$ and all preceding frames are accepted except for those that have been explicitly rejected using an SREJ frame. Comment on any possible drawback to this scheme.

7.12 The ISO standard for HDLC procedures (ISO 4335) includes the following definitions: (1) an REJ condition is considered cleared upon the receipt of an incoming I-frame with an N(S) equal to the N(R) of the outgoing REJ frame; and (2) a SREJ condition is considered cleared upon the receipt of an I-frame with an N(S) equal to the N(R) of the SREJ frame. The standard includes rules concerning the relationship between REJ and SREJ frames. These rules indicate what is allowable (in terms of transmitting REJ and SREJ frames) if an REJ condition has not yet been cleared and what is allowable if an SREJ condition has not yet been cleared. Deduce the rules and justify your answer.

7.13 Two stations communicate via a 1-Mbps satellite link with a propagation delay of 270 ms. The satellite serves merely to retransmit data received from one station to another, with negligible switching delay. Using HDLC frames of 1024 bits with 3-bit sequence numbers, what is the maximum possible data throughput; that is, what is the throughput of data bits carried in HDLC frames?

7.14 It is clear that bit stuffing is needed for the address, data, and FCS fields of an HDLC frame. Is it needed for the control field?

7.15 Because of the provision that a single flag can be used as both an ending and a starting flag, a single bit error can cause problems.

a. Explain how a single bit error can merge two frames into one.
b. Explain how a single bit error can split a single frame into two frames.

7.16 Suggest improvements to the bit stuffing-algorithm to overcome the problems of single-bit errors described in the preceding problem.

7.17 Using the example bit string of Figure 7.8, show the signal pattern on the line using NRZ-L coding. Does this suggest a side benefit of bit stuffing?

7.18 Assume that the primary HDLC station in NRM has sent six I-frames to a secondary. The primary's N(S) count was three (011 binary) prior to sending the six frames. If the poll bit is on in the sixth frame, what will be the N(R) count back from the secondary after the last frame? Assume error-free operation.

7.19 Consider that several physical links connect two stations. We would like to use a "multilink HDLC" that makes efficient use of these links by sending frames on a FIFO basis on the next available link. What enhancements to HDLC are needed?

7.20 A World Wide Web server is usually set up to receive relatively small messages from its clients but to transmit potentially very large messages to them. Explain, then, which type of ARQ protocol (selective reject, go-back-N) would provide less of a burden to a particularly popular WWW server.

APPENDIX 7A PERFORMANCE ISSUES

In this appendix, we examine some of the performance issues related to the use of sliding-window flow control.

Stop-and-Wait Flow Control

Let us determine the maximum potential efficiency of a half-duplex point-to-point line using the stop-and-wait scheme described in Section 7.1. Suppose that a long message is to be sent as a sequence of frames $F_1, F_2, \ldots, F_n$, in the following fashion:

- Station S_1 sends F_1.
- Station S_2 sends an acknowledgment.
- Station S_1 sends F_2.
- Station S_2 sends an acknowledgment.
- Station S_1 sends F_n.
- Station S_2 sends an acknowledgment.

The total time to send the data, T, can be expressed as $T = nT_F$, where T_F is the time to send one frame and receive an acknowledgment. We can express T_F as follows:

$$T_F = t_{\text{prop}} + t_{\text{frame}} + t_{\text{proc}} + t_{\text{prop}} + t_{\text{ack}} + t_{\text{proc}}$$

where

t_{prop} = propagation time from S_1 to S_2

t_{frame} = time to transmit a frame (time for the transmitter to send out all of the bits of the frame)

t_{proc} = processing time at each station to react to an incoming event

t_{ack} = time to transmit an acknowledgment

Let us assume that the processing time is relatively negligible, and that the acknowledgment frame is very small compared to a data frame, both of which are reasonable assumptions. Then we can express the total time to send the data as

$$T = n(2t_{prop} + t_{frame})$$

Of that time, only $n \times t_{frame}$ is actually spent transmitting data and the rest is overhead. The utilization, or efficiency, of the line is

$$U = \frac{n \times t_{frame}}{n(2t_{prop} + t_{frame})} = \frac{t_{frame}}{2t_{prop} + t_{frame}} \quad \textbf{(7.3)}$$

It is useful to define the parameter $a = t_{prop}/t_{frame}$ (see Figure 7.2). Then

$$U = \frac{1}{1 + 2a} \quad \textbf{(7.4)}$$

This is the maximum possible utilization of the link. Because the frame contains overhead bits, actual utilization is lower. The parameter a is constant if both t_{prop} and t_{frame} are constants, which is typically the case: Fixed-length frames are often used for all except the last frame in a sequence, and the propagation delay is constant for point-to-point links.

To get some insight into Equation (7.4), let us derive a different expression for a. We have

$$a = \frac{\text{Propagation Time}}{\text{Transmission Time}} \quad \textbf{(7.5)}$$

The propagation time is equal to the distance d of the link divided by the velocity of propagation V. For unguided transmission through air or space, V is the speed of light, approximately 3×10^8 m/s. For guided transmission, V is approximately 0.67 times the speed of light for optical fiber and copper media. The transmission time is equal to the length of the frame in bits, L, divided by the data rate R. Therefore,

$$a = \frac{d/V}{L/R} = \frac{Rd}{VL}$$

Thus, for fixed-length frames, a is proportional to the data rate times the length of the medium. A useful way of looking at a is that it represents the length of the medium in bits $[R \times (d/v)]$ compared to the frame length (L).

With this interpretation in mind, Figure 7.2 illustrates Equation (7.4). In this figure, transmission time is normalized to 1 and hence the propagation time, by Equation (7.5), is a. For the case of $a < 1$, the link's bit length is less than that of the frame. The station T begins transmitting a frame at time t_0. At $t_0 + a$, the leading edge of the frame reaches the receiving station R, while T is still in the process of transmitting the frame. At $t_0 + 1$, T completes transmission. At $t_0 + 1 + a$, R has received the entire frame and immediately transmits a small acknowledgment frame. This acknowledgment arrives back at T at $t_0 + 1 + 2a$. Total elapsed time: $1 + 2a$. Total transmission time: 1. Hence utilization is $1/(1 + 2a)$. The same result is achieved with $a > 1$, as illustrated in Figure 7.2.

EXAMPLE 7.6 First, consider a wide area network (WAN) using ATM (asynchronous transfer mode, described in Part Three), with the two stations a thousand kilometers apart. The standard ATM frame size (called a cell) is 424 bits and one of the standardized data rates is 155.52 Mbps. Thus, transmission time equals $424/(155.52 \times 10^6) = 2.7 \times 10^{-6}$ seconds. If we assume an optical fiber

link, then the propagation time is $(10^6$ meters$)/(2 \times 10^8$ m/s$) = 0.5 \times 10^{-2}$ seconds. Thus, $a = (0.5 \times 10^{-2})/(2.7 \times 10^{-6}) \approx 1850$, and efficiency is only $1/3701 = 0.00027$.

At the other extreme, in terms of distance, is the local area network (LAN). Distances range from 0.1 to 10 km, with data rates of 10 Mbps to 1 Gbps; higher data rates tend to be associated with shorter distances. Using a value of $V = 2 \times 10^8$ m/s, a frame size of 1000 bits, and a data rate of 10 Mbps, the value of a is in the range of 0.005 to 0.5. This yields a utilization in the range of 0.5 to 0.99. For a 100-Mbps LAN, given the shorter distances, comparable utilizations are possible.

We can see that LANs are typically quite efficient, whereas high-speed WANs are not. As a final example, let us consider digital data transmission via modem over a voice-grade line. A typical data rate is 56 kbps. Again, let us consider a 1000-bit frame. The link distance can be anywhere from a few tens of meters to thousands of kilometers. If we pick, say, as a short distance $d = 1000$ m, then $a = (56{,}000 \text{ bps} \times 1000 \text{ m}) / (2 \times 10^8 \text{ m/s} \times 1000 \text{ bits}) = 2.8 \times 10^{-4}$, and utilization is effectively 1.0. Even in a long-distance case, such as $d = 5000$ km, we have $a = (56{,}000 \times 5 \times 10^6)/(2 \times 10^8 \times 1000 \text{ bits}) = 1.4$ and efficiency equals 0.26.

Error-Free Sliding-Window Flow Control

For sliding-window flow control, the throughput on the line depends on both the window size W and the value of a. For convenience, let us again normalize frame transmission time to a value of 1; thus, the propagation time is a. Figure 7.11 illustrates the efficiency of a full duplex point-to-point line.[3] Station A begins to emit a sequence of frames at time $t = 0$. The leading edge of the first frame reaches station B at $t = a$. The first frame is entirely absorbed by $t = a + 1$. Assuming negligible processing time, B can immediately acknowledge the first frame (ACK). Let us also assume that the acknowledgment frame is so small that transmission time is negligible. Then the ACK reaches A at $t = 2a + 1$. To evaluate performance, we need to consider two cases:

- **Case 1:** $W \geq 2a + 1$. The acknowledgment for frame 1 reaches A before A has exhausted its window. Thus, A can transmit continuously with no pause and normalized throughput is 1.0.
- **Case 2:** $W < 2a + 1$. A exhausts its window at $t = W$ and cannot send additional frames until $t = 2a + 1$. Thus, normalized throughput is W time units out of a period of $(2a + 1)$ time units.

Therefore, we can express the utilization as

$$U = \begin{cases} 1 & W \geq 2a + 1 \\ \dfrac{W}{2a + 1} & W < 2a + 1 \end{cases} \tag{7.6}$$

[3]For simplicity, we assume that a is an integer, so that an integer number of frames exactly fills the line. The argument does not change for noninteger values of a.

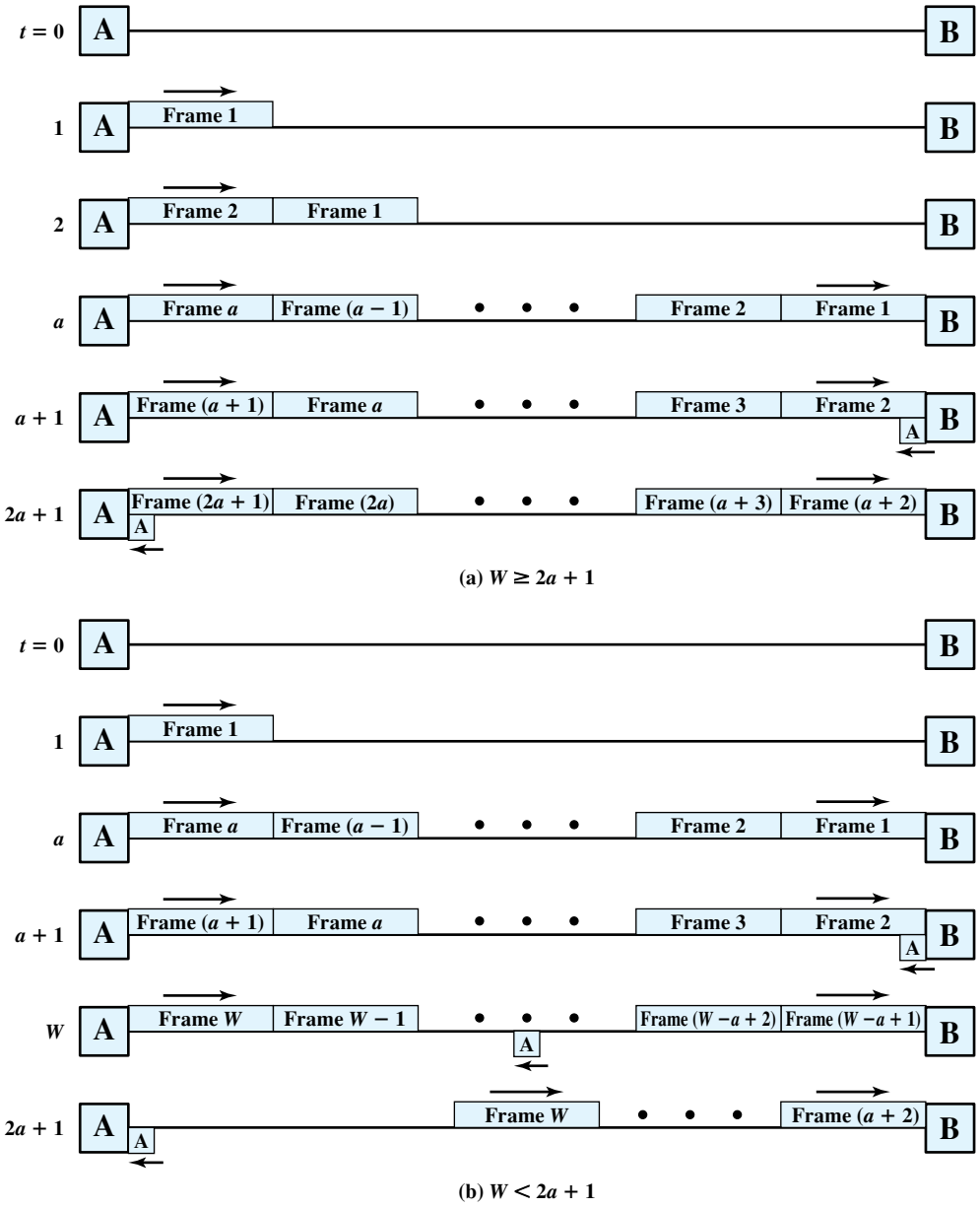

Figure 7.11 Timing of Sliding-Window Protocol

Typically, the sequence number is provided for in an n-bit field and the maximum window size is $W = 2^n - 1$ (not 2^n; this is explained in Section 7.2). Figure 7.12 shows the maximum utilization achievable for window sizes of 1, 7, and 127 as a function of a. A window size of 1 corresponds to stop and wait. A window size of 7 (3 bits) is adequate for many applications. A window size of 127 (7 bits) is adequate for larger values of a, such as may be found in high-speed WANs.

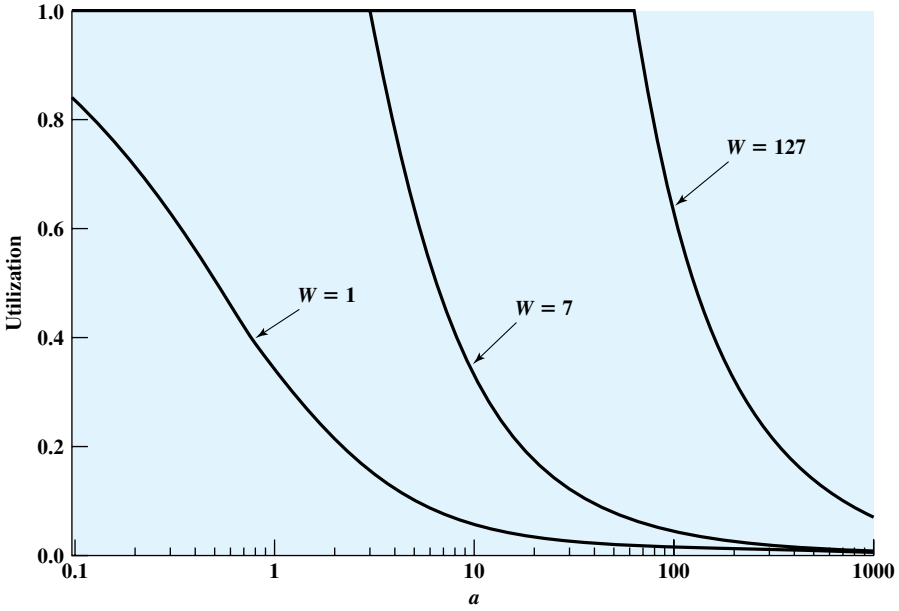

Figure 7.12 Sliding-Window Utilization as a Function of *a*

ARQ

We have seen that sliding-window flow control is more efficient than stop-and-wait flow control. We would expect that when error control functions are added that this would still be true: that is, that go-back-N and selective-reject ARQ are more efficient than stop-and-wait ARQ. Let us develop some approximations to determine the degree of improvement to be expected.

First, consider stop-and-wait ARQ. With no errors, the maximum utilization is $1/(1 + 2a)$ as shown in Equation (7.4). We want to account for the possibility that some frames are repeated because of bit errors. To start, note that the utilization U can be defined as

$$U = \frac{T_f}{T_t} \tag{7.7}$$

where

T_f = time for transmitter to emit a single frame

T_t = total time that line is engaged in the transmission of a single frame

For error-free operation using stop-and-wait ARQ,

$$U = \frac{T_f}{T_f + 2T_p}$$

where T_p is the propagation time. Dividing by T_f and remembering that $a = T_p/T_f$, we again have Equation (7.4). If errors occur, we must modify Equation (7.7) to

$$U = \frac{T_f}{N_r T_t}$$

where N_r is the expected number of transmissions of a frame. Thus, for stop-and-wait ARQ, we have

$$U = \frac{1}{N_r(1 + 2a)}$$

A simple expression for N_r can be derived by considering the probability P that a single frame is in error. If we assume that ACKs and NAKs are never in error, the probability that it will take exactly k attempts to transmit a frame successfully is $P^{k-1}(1 - P)$. That is, we have $(k - 1)$ unsuccessful attempts followed by one successful attempt; the probability of this occurring is just the product of the probability of the individual events occurring. Then[4]

$$N_r = \text{E[transmissions]} = \sum_{i=1}^{\infty} (i \times \Pr[i \text{ transmissions}]) = \sum_{i=1}^{\infty} (iP^{i-1}(1 - P)) = \frac{1}{1 - P}$$

So we have

$$\textbf{Stop-and Wait:} \qquad U = \frac{1 - P}{1 + 2a}$$

For the sliding-window protocol, Equation (7.6) applies for error-free operation. For selective-reject ARQ, we can use the same reasoning as applied to stop-and-wait ARQ. That is, the error-free equations must be divided by N_r. Again, $N_r = 1/(1 - P)$. So

$$\textbf{Selective Reject:} \qquad U = \begin{cases} 1 - P & W \geq 2a + 1 \\ \dfrac{W(1 - P)}{2a + 1} & W < 2a + 1 \end{cases}$$

The same reasoning applies for go-back-N ARQ, but we must be more careful in approximating N_r. Each error generates a requirement to retransmit K frames rather than just one frame. Thus

$$\begin{aligned} N_r &= \text{E[number of transmitted frames to successfully transmit one frame]} \\ &= \sum_{i=1}^{\infty} f(i)P^{i-1}(1 - P) \end{aligned}$$

where $f(i)$ is the total number of frames transmitted if the original frame must be transmitted i times. This can be expressed as

$$\begin{aligned} f(i) &= 1 + (i - 1)K \\ &= (1 - K) + Ki \end{aligned}$$

Substituting yields[5]

$$\begin{aligned} N_r &= (1 - K)\sum_{i=1}^{\infty} P^{i-1}(1 - P) + K\sum_{i=1}^{\infty} iP^{i-1}(1 - P) \\ &= 1 - K + \frac{K}{1 - P} \\ &= \frac{1 - P + KP}{1 - P} \end{aligned}$$

By studying Figure 7.11, the reader should conclude that K is approximately equal to $(2a + 1)$ for $W \geq (2a + 1)$, and $K = W$ for $W < (2a + 1)$. Thus

---

[4] This derivation uses the equality $\sum_{i=1}^{\infty} (iX^{i-1}) = \frac{1}{(1 - X)^2}$ for $(-1 < X < 1)$.

[5] This derivation uses the equality $\sum_{i=1}^{\infty} X^{i-1} = \frac{1}{1 - X}$ for $(-1 < X < 1)$.

$$\textbf{Go-back-N:} \quad U = \begin{cases} \dfrac{1 - P}{1 + 2aP} & W \geq 2a + 1 \\ \dfrac{W(1 - P)}{(2a + 1)(1 - P + WP)} & W < 2a + 1 \end{cases}$$

Note that for $W = 1$, both selective-reject and go-back-N ARQ reduce to stop and wait. Figure 7.13[6] compares these three error control techniques for a value of $P = 10^{-3}$. This figure and the equations are only approximations. For example, we have ignored errors in acknowledgment frames and, in the case of go-back-N, errors in retransmitted frames other than the frame initially in error. However, the results do give an indication of the relative performance of the three techniques.

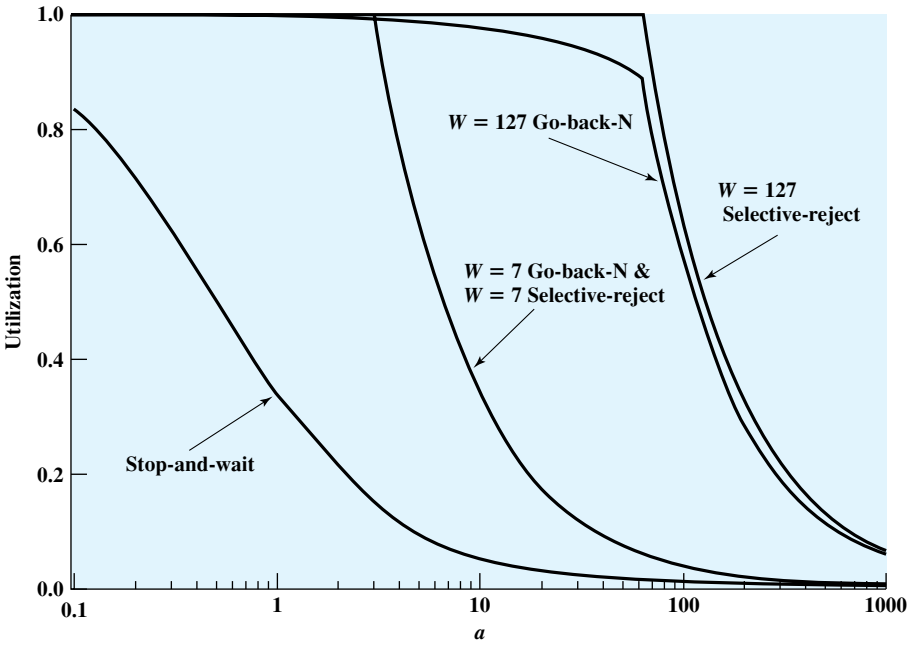

Figure 7.13 ARQ Utilization as a Function of a ($P = 10^{-3}$)

[6] For $W = 7$, the curves for go-back-N and selective-reject are so close that they appear to be identical in the figure.