5.2 ARQ PROTOCOLS

Automatic Repeat Request (ARQ) is a technique used to ensure that a data stream is delivered accurately to the user despite errors that occur during transmission. ARQ forms the basis for peer-to-peer protocols that provide for the reliable transfer of information. In this section we develop the three basic types of ARQ protocols, starting with the simplest and building up to the most complex. We also discuss the settings in which the three ARQ protocol types are applied.

This discussion assumes that the user generates a sequence of information blocks for transmission. The ARQ mechanism requires the block to contain a header with control information that is essential to proper operation, as shown in Figure 5.8. The transmitter will also append CRC check bits that cover the header and the information bits to enable the receiver to determine whether errors have occurred during transmission. We assume that the design of the CRC ensures that transmission errors can be detected with very high probability, as discussed in Chapter 3. Recall from Chapter 2 that we use the term **frame** to refer to the binary block that results from the combination of the header, user information, and CRC at the data link layer. We refer to the set of rules that govern the operation of the transmitter and receiver as the **ARQ protocol**.

The ARQ protocol can be applied in a number of scenarios. Traditionally, the most common application involves transmission over a single noisy communication channel. ARQ is introduced here to ensure a high level of reliability across a single transmission hop. As communication lines have become less noisy, the ARQ protocol has been implemented more often at the edges of the network to provide end-to-end reliability in the transmission of packets over multiple hops in a network, that is, over multiple communication channels and other network equipment. In this section we assume that the channel or

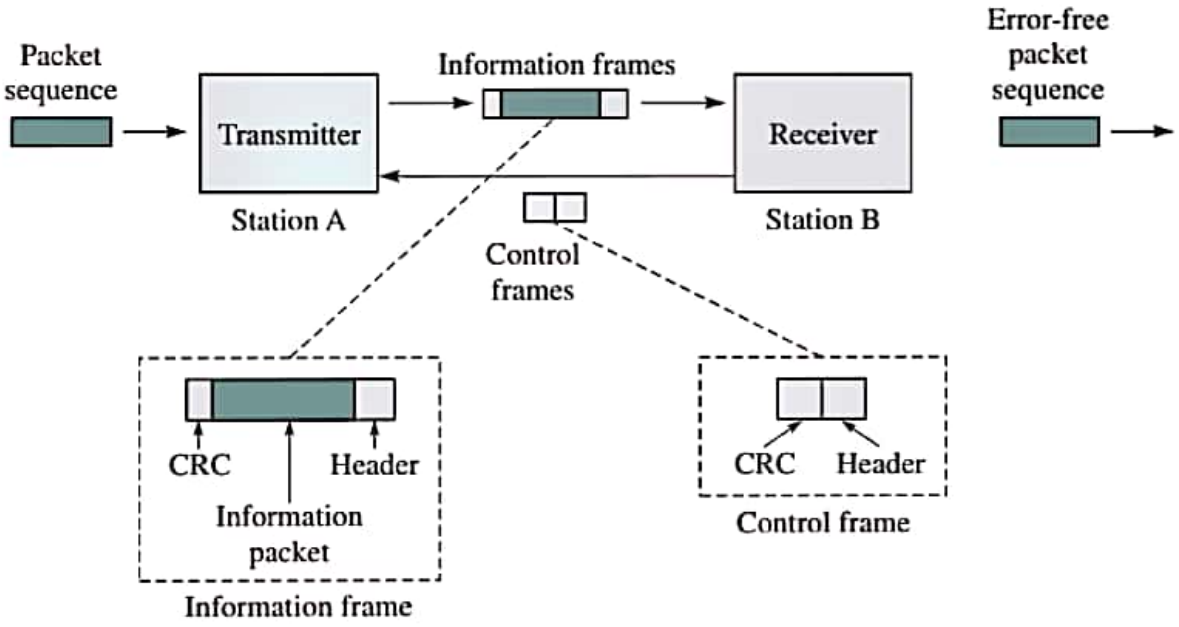


FIGURE 5.8 Basic elements of ARQ

sequence of channels is "wirelike" in the sense that the frames arrive at the receiver, if they arrive at all, in the same order in which they were sent. In the case of multiple hops over a network, this assumption would hold when a connection is set up and where all frames follow the same path, as in frame relay or ATM networks. In particular, we assume that frames, while in transit, cannot pass previously transmitted frames.[3] In situations where these assumptions hold, the objective of the ARQ protocol is to ensure that packets are delivered error free to the destination, exactly once without duplicates, in the same order in which they were transmitted.

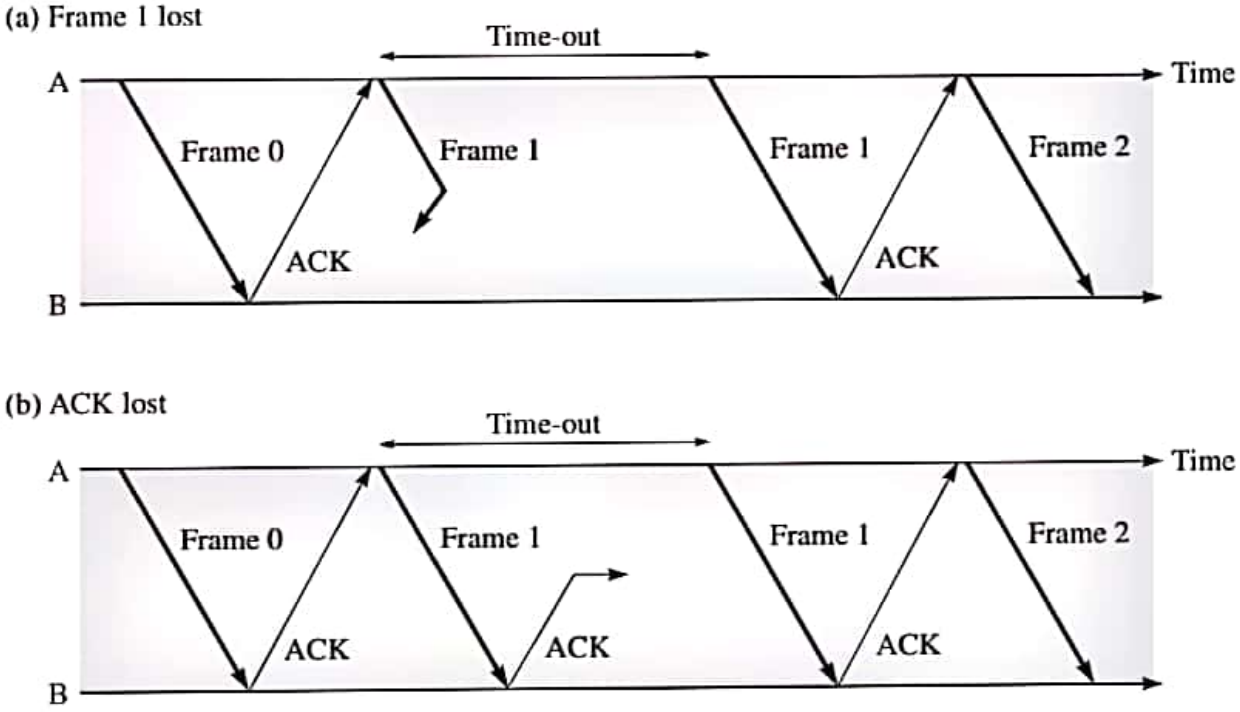
In addition to the error-detection code, the other basic elements of ARQ protocols consist of **information frames (I-frames)** that transfer the user packets, control frames, and time-out mechanisms, as shown in Figure 5.8. **Control frames** are short binary blocks that consist of a header that provides the control information followed by the CRC. The control frames include **ACKs**, which acknowledge the correct receipt of a given frame or group of frames; **NAKs**, which indicate that a frame has been received in error and that the receiver is taking certain action; and an **enquiry frame ENQ**, which commands the receiver to report its status. The time-out mechanisms are required to prompt certain actions to maintain the flow of frames. We can visualize the transmitter and receiver as working jointly on ensuring the correct and orderly delivery of the sequence of packets provided by the sender.

We begin with the simple case where information flows only in one direction, from the transmitter to the receiver. The reverse communication channel is used only for the transmission of control information. Later in the section we consider the case of bidirectional transfer of information.

5.2.1 Stop-and-Wait ARQ

The first protocol we consider is **Stop-and-Wait ARQ** where the transmitter and receiver work on the delivery of one frame at a time through an alternation of actions. In Figure 5.9a we show how ACKs and time-outs can be used to provide recovery from transmission errors, in this case a lost frame. At the initial point in the figure, stations A and B are working on the transmission of frame 0. Note that each time station A sends an I-frame, it starts an **I-frame timer** that will expire after some time-out period. The time-out period is selected so that it is greater than the time required to receiver the corresponding ACK frame. Figure 5.9a shows the following sequence of events:

1. Station A transmits frame 0 and then waits for an ACK frame from the receiver.
2. Frame 0 is transmitted without error, so station B transmits an ACK frame.
3. The ACK from station B is also received without error, so station A knows the frame 0 has been received correctly.
4. Station A now proceeds to transmit frame 1 and then resets the timer.
5. Frame 1 undergoes errors in transmission. It is possible that station B receives frame 1 and detects the errors through the CRC check; it is also possible that frame 1 was so badly garbled that station B is unaware of the transmission.[4] In either case station B does not take any action.
6. The time-out period expires, and frame 1 is retransmitted.



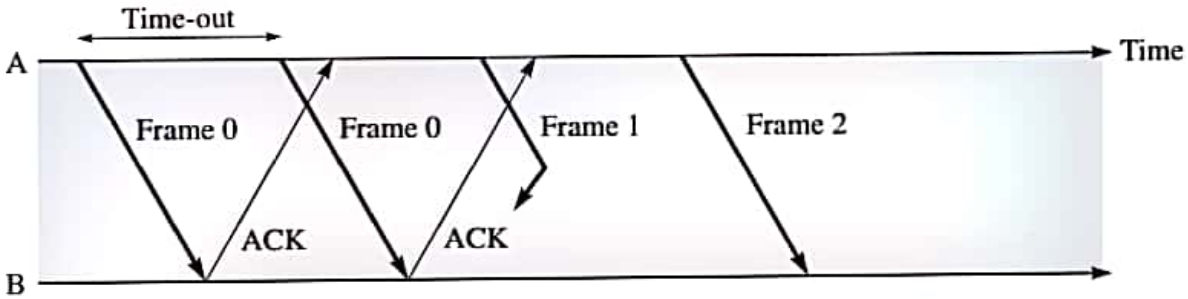
In parts (a) and (b) transmitting station A acts the same way, but part (b) receiving station B accepts frame 1 twice.

The protocol continues in this manner until frame 1 is received and acknowledged. The protocol then proceeds to frame 2, and so on.

Transmission errors in the reverse channel lead to ambiguities in the Stop-and-Wait protocol that need to be corrected. Figure 5.9b shows the situation that begins as in Figure 5.9a, but where frame 1 is received correctly, and its acknowledgment undergoes errors. After receiving frame 1 station B delivers its contents to the destination. Station A does not receive the acknowledgment for frame 1, so the time-out period expires. Note that at this point station A cannot distinguish between the sequence of events in parts (a) and (b) of Figure 5.9. Station A proceeds to retransmit the frame. If the frame is received correctly by station B, as shown in the figure, then station B will accept frame 1 as a new frame and redeliver it to the user. Thus we see that the loss of an ACK can result in the delivery of a duplicate packet. The ambiguity can be eliminated by including a sequence number in the header of each I-frame. Station B would then recognize that the second transmission of frame 1 was a duplicate, discard the frame, and resend the ACK for frame 1.

A second type of ambiguity arises if the ACKs do not contain a sequence number. In Figure 5.10 frame 0 is transmitted, but the time-out expires prematurely. Frame 0 is received correctly, and the (unnumbered) ACK is returned. In the meantime station A has resent frame 0. Shortly thereafter, station A receives an ACK and assumes it is for the last frame. Station A then proceeds to send frame 1, which incurs transmission errors. In the meantime the second transmission of frame 0 has been received and acknowledged by station B. When station A receives the second ACK, the station assumes the ACK is for frame 1 and proceeds to transmit frame 2. The mechanism fails because frame 1 is not delivered. This example shows that *premature time-outs (or delayed ACKs) combined with loss of I-frames can result in gaps in the delivered packet sequence.* This ambiguity is resolved by providing a sequence number in the acknowledgment frames that enables the transmitter to determine which frames have been received.

The sequence numbers cannot be allowed to become arbitrarily large because only a finite number of bits are available in the frame headers. We now show that a one-bit sequence number suffices to remove the above ambiguities in the Stop-and-Wait protocol. Figure 5.11 shows the information or "state" that is main-



Transmitting station A misinterprets duplicate ACKs

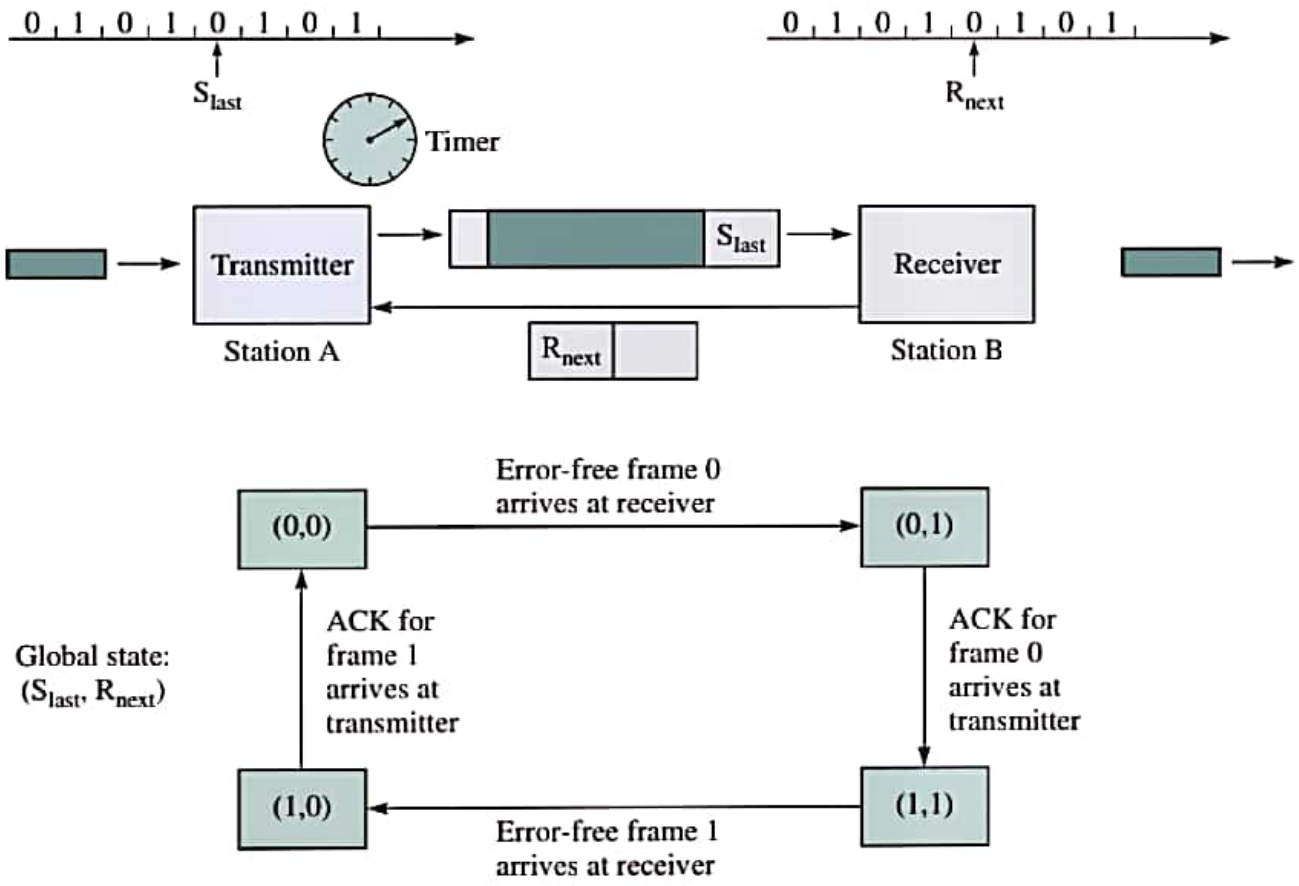


FIGURE 5.11 System state information in Stop-and-Wait ARQ

tained by the transmitter and receiver. The transmitter must keep track of the sequence number S_{last} of the frame being sent, its associated timer, and the frame itself in case retransmission is required. The receiver keeps track only of the sequence number R_{next} of the next frame it is expecting to receive.

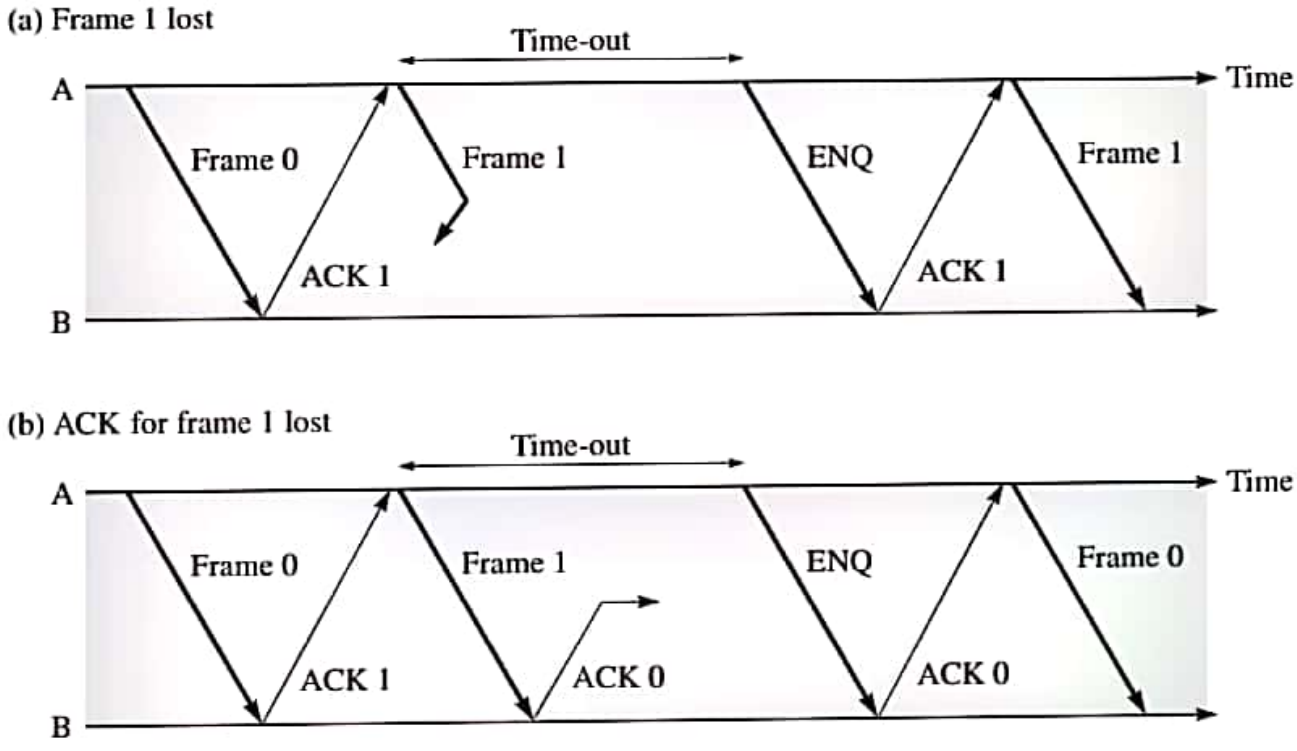
Suppose that initially the transmitter and receiver are synchronized in the sense that station A is about to send a frame with $S_{last} = 0$ and station B is expecting $R_{next} = 0$. In Figure 5.11 the global state of the system is defined by the pair (S_{last}, R_{next}), so initially the system is in state (0,0).

The system state will not change until station B receives an error-free version of frame 0. That is, station A will continue resending frame 0 as dictated by the time-out mechanism. Eventually station B receives frame 0, station B changes R_{next} to 1 and sends an acknowledgment to station A with $R_{next} = 1$ implicitly acknowledging the receipt of frame 0. At this point the state of the system is (0,1). Any subsequent received frames that have sequence number 0 are recognized as duplicates and discarded by station B, and an acknowledgment with $R_{next} = 1$ is resent. Eventually station A receives an acknowledgment with $R_{next} = 1$ and then begins transmitting the next frame, using sequence number $S_{last} = 1$. The system is now in state (1,1). The transmitter and receiver are again synchronized, and they now proceed to work together on the transfer of frame 1. Therefore, a protocol that implements this mechanism that follows the well-defined sequence of states shown in Figure 5.11 can ensure the correct and orderly delivery of frames to the destination.

A number of modifications can be made to the above ARQ mechanism by the use of additional control frames. For example, in **checkpointing** the error recovery can be expedited through the use of a short control frame called the enquiry frame (ENQ). When a time-out expires, if the frame that needs to be retransmitted is very long, the transmitter can send an ENQ. When a station receives such a frame that station is compelled to retransmit its previous frame. As shown in Figure 5.12, station B then proceeds to retransmit its last ACK frame, which when received by station A resolves the ambiguity. Station A can then proceed with the transmission of the appropriate frame, as shown in the figure. By convention ACK and NAK frames contain R_{next}, indicating the next frame that is expected by the receiver and implicitly acknowledging delivery of all prior frames.

Stop-and-Wait ARQ becomes inefficient when the propagation delay is much greater than the time to transmit a frame. For example, suppose that we are transmitting frames that are 1000 bits long over a channel that has a speed of 1.5 megabits/second and suppose that the time that elapses from the beginning of the frame transmission to the receipt of its acknowledgment is 40 ms. The number of bits that can be transmitted over this channel in 40 ms is $40 \times 10^{-3} \times 1.5 \times 10^{6} = 60{,}000$ bits. However, Stop-and-Wait ARQ can transmit only 1000 bits in this period time. This severe inefficiency is due to the requirement that the transmitter wait for the acknowledgment of a frame before proceeding with other transmissions. Later in the section we show that the situation becomes much worse in the presence of transmission errors that trigger retransmissions.

The **delay-bandwidth product** is the product of the bit rate and the delay that elapses before an action can take place. In the preceding example the delay-



5.2.2 Go-Back-N ARQ

In this section we show that the inefficiency of Stop-and-Wait ARQ can be overcome by allowing the transmitter to continue sending enough frames so that the channel is kept busy while the transmitter waits for acknowledgments. We develop the Go-Back-N protocol that forms the basis for the HDLC data link protocol, which is discussed at the end of this chapter. Suppose for now that frames are numbered 0, 1, 2, 3, ... The transmitter has a limit on the number of frames W_S that can be outstanding. W_S is chosen larger than the delay-bandwidth product to ensure that the channel can be kept busy.

The idea of the **Basic Go-Back-N ARQ** is as follows: Consider the transfer of a reference frame, say, frame 0. After frame 0 is sent, the transmitter sends $W_S - 1$ additional frames into the channel, optimistic that frame 0 will be received correctly and not require retransmission. If things turn out as expected, an ACK for frame 0 will arrive in due course while the transmitter is still busy sending frames into the channel, as shown in Figure 5.13. The system is now done with frame 0. Note, however, that the handling of frame 1 and subsequent frames is already well underway. A procedure where the processing of a new task is begun before the completion of the previous task is said to be **pipelined**. In

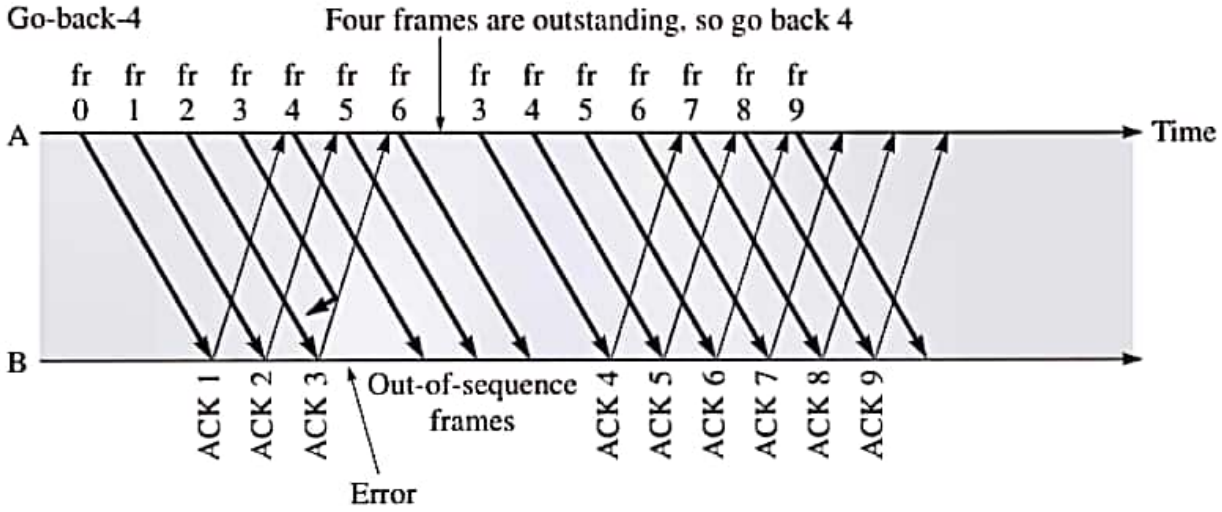


FIGURE 5.13 Basic Go-Back-N ARQ

effect Go-Back-N ARQ pipelines the processing of frames to keep the channel busy.

Go-Back-N ARQ gets its name from the action that is taken when an error occurs. As shown in Figure 5.13, after frame 3 undergoes transmission errors, the receiver ignores frame 3 and all subsequent frames. Eventually the transmitter reaches the maximum number of outstanding frames. It is then forced to "go back N" frames, where $N = W_S$, and begin retransmitting all packets from 3 onwards.

In the previous discussion of Stop-and-Wait, we used the notion of a global state (S_{lasts}, R_{next}) to demonstrate the correct operation of the protocol in the sense of delivering the packets error free and in order. Figure 5.14 shows the similarities between Go-Back-N ARQ and Stop-and-Wait ARQ in terms of error recovery. In Stop-and-Wait ARQ, the occurrence of a frame-transmission error results in the loss of transmission time equal to the duration of the time-out period. In Go-Back-N ARQ, the occurrence of a frame-transmission error results in the loss of transmission time corresponding to W_S frames. In Stop-and-Wait the receiver is looking for the frame with sequence number R_{next}; in Go-Back-N the receiver is looking for a frame with a specific sequence number, say, R_{next}. If we identify the oldest outstanding (transmitted but unacknowledged) frame in Go-Back-N with the S_{last} frame in Stop-and-Wait, we see that the correct operation of the Go-Back-N protocol depends on ensuring that the oldest frame is eventually delivered successfully. The protocol will trigger a retransmission of S_{last} and the subsequent $W_S - 1$ frames each time the send window is exhausted. Therefore, as long as there is a nonzero probability of error-free frame transmission, the eventual error-free transmission of S_{last} is assured and the protocol will operate correctly. We next modify the basic protocol we have developed to this point.

The Go-Back-N ARQ as stated above depends on the transmitter exhausting its maximum number of outstanding frames to trigger the retransmission of a frame. Thus this protocol works correctly as long as the transmitter has an unlimited supply of packets that need to be transmitted. In situations where

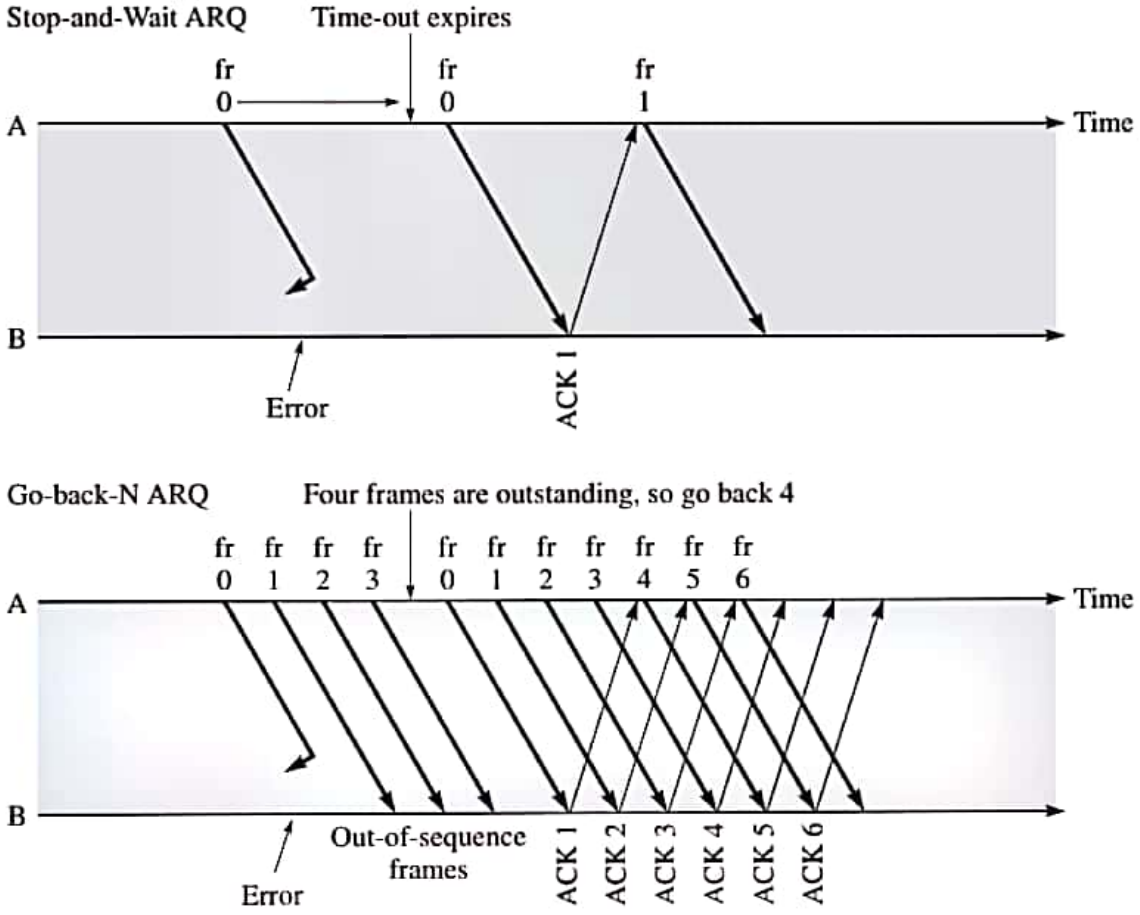


FIGURE 5.14 Relationship of Stop-and-Wait ARQ and Go-Back-N ARQ

packets arrive sporadically, there may not be $W_S - 1$ subsequent transmissions. In this case retransmissions are not triggered, since the window is not exhausted. This problem is easily resolved by modifying Go-Back-N ARQ such that a timer is associated with each transmitted frame.

Figure 5.15 shows how the resulting Go-Back-N ARQ protocol operates. The transmitter must now maintain a list of the frames it is processing, where S_{last} is the number of the last transmitted frame that remains unacknowledged and S_{recent} is the number of the most recently transmitted frame. The transmitter must also maintain a timer for each transmitted frame and must also buffer all frames that have been transmitted but have not yet been acknowledged. At any point in time the transmitter has a **send window** of available sequence numbers. The lower end of the window is given by S_{last}, and the upper limit of the transmitter window is $S_{last} + W_S - 1$. If S_{recent} reaches the upper limit of the window, the transmitter is not allowed to transmit further new frames until the send window slides forward with the receipt of a new acknowledgment.

The Go-Back-N protocol is an example of a **sliding-window protocol**. The receiver maintains a **receive window** of size 1 that consists of the next frame R_{next} it expects to receive. If an arriving frame passes the CRC check and has the correct sequence number, that is, R_{next}, then it is accepted and R_{next} is incremented. We say that the receive window slides forward. The receiver then sends an acknowledgment containing the incremented sequence number R_{next}, which

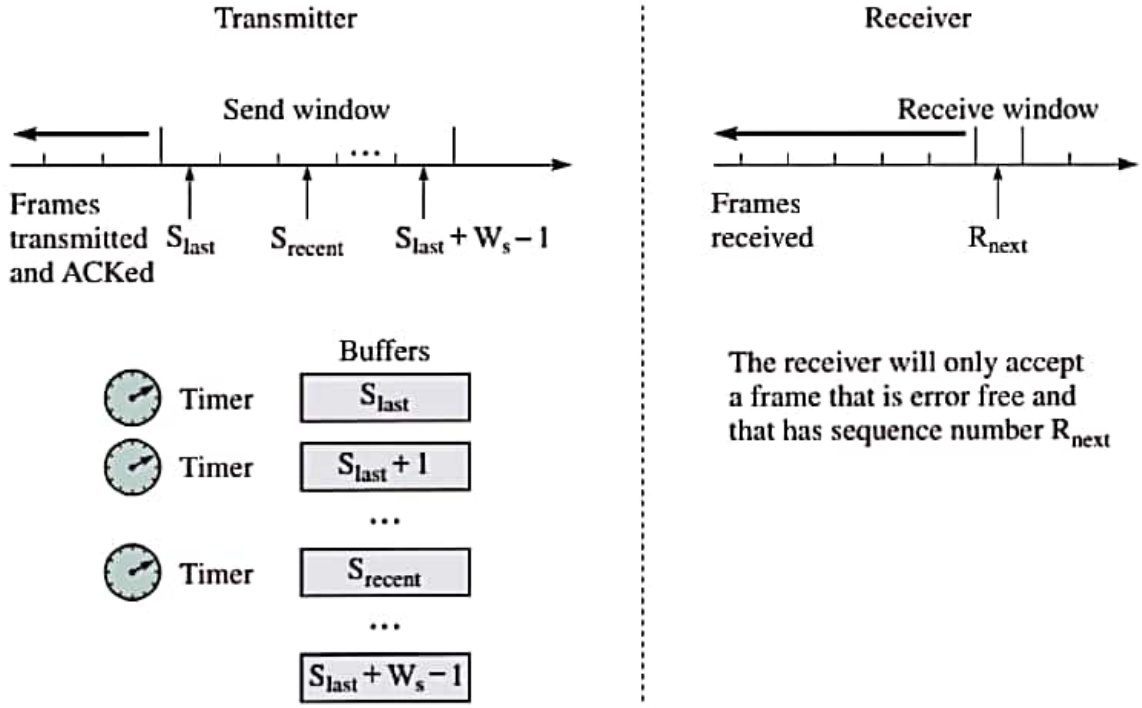


FIGURE 5.15 Go-Back-N ARQ

implicitly acknowledges receipt of all frames prior to R_{next}. Note how we are making use of the assumption that the channel is wirelike: When the transmitter receives an ACK with a given value R_{next}, it can assume that all prior frames have been received correctly, even if it has not received ACKs for those frames, either because they were lost or because the receiver chose not to send them. Upon receiving an ACK with a given value R_{next}, the transmitter updates its value of S_{last} to R_{next} and in so doing the send window slides forward. (Note that this action implies that $S_{last} \leq R_{next}$. Note as well that $R_{next} \leq S_{recent}$, since S_{recent} is the last in the transmission frames.)

The number of bits that can be allotted within a header per sequence number is limited to some number, say, m, which then allows us to represent at most 2^m possible sequence numbers, so the sequence numbers must be counted using modulo 2^m. Thus if $m = 3$, then the sequence of frames would carry the sequence numbers 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, ... When an error-free frame arrives, the receiver must be able to unambiguously determine which frame has been received, taking into account that the sequence numbers wrap around when the count reaches 2^m. We will next show that the receiver can determine the correct frame if the window size is smaller than 2^m.

The example in Figure 5.16 shows the ambiguities that arise when the above inequality is not satisfied. The example uses $2^m = 2^2 = 4$ sequence numbers. The transmitter initially sends four frames in a row. The receiver sends four corresponding acknowledgments, which are all obliterated in the return channel. When the transmitter exhausts its available frame numbers, it goes back four and begins retransmitting from frame 0. When frame 0 arrives at the receiver, the receiver has $R_{next} = 0$, so it accepts the frame. However, the receiver does not know whether the ACK for the previous frame 0 was received. Consequently, the

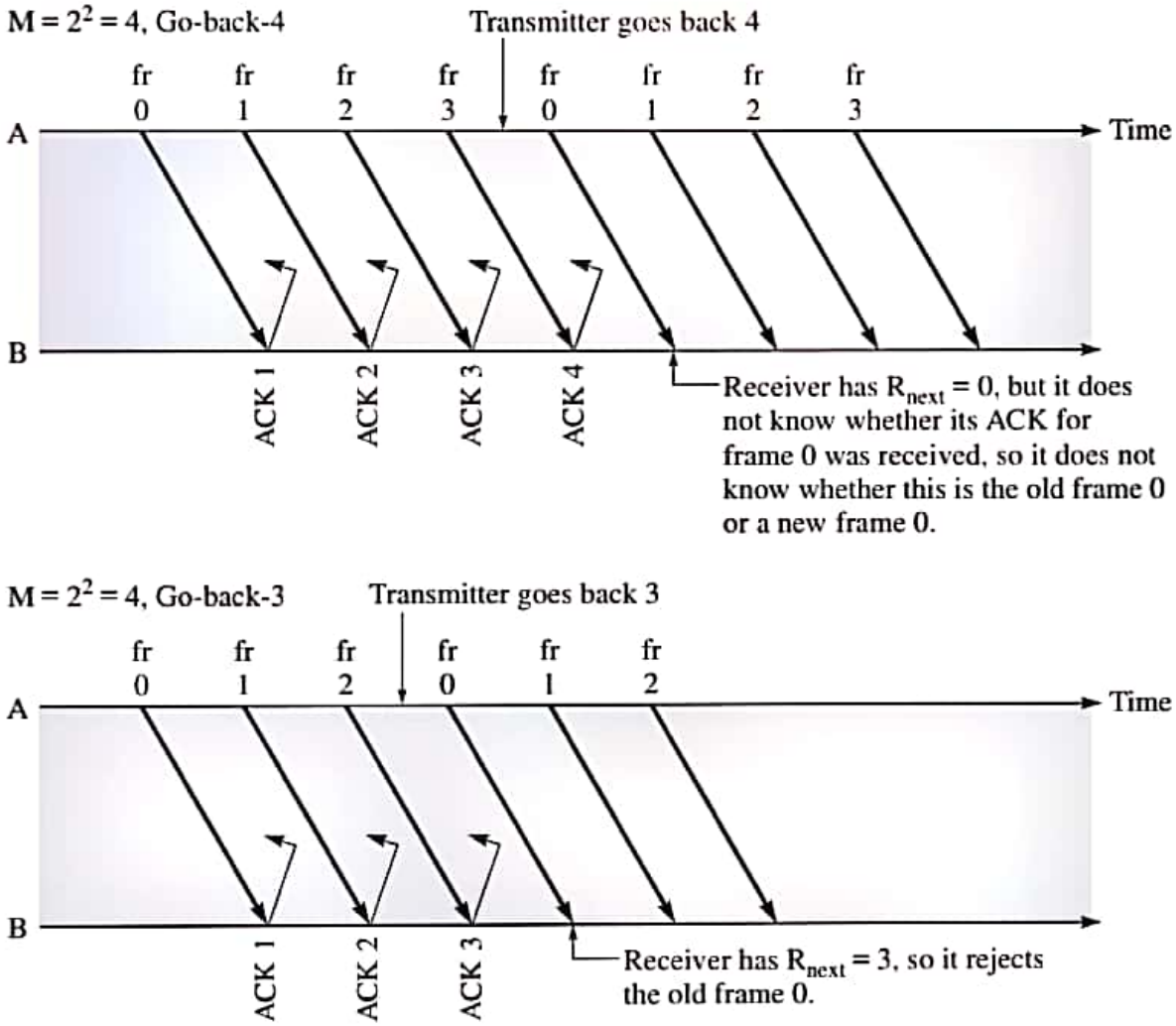


FIGURE 5.16 The window size should be less than 2^m

receiver cannot determine whether this is a new frame 0 or an old frame 0. The second example in Figure 5.16 uses $2^m = 4$ sequence numbering, but a window size of 3. In this case we again suppose that the transmitter sends three consecutive frames and that acknowledgments are lost in the return channel. When the retransmitted frame 0 arrives at the receiver, $R_{next} = 3$, so the frame is recognized to be an old one.

We can generalize Figure 5.16 as follows. In general, suppose the window size W_S is $2^m - 1$ or less and assume that the current send window is 0 up to $W_S - 1$. Suppose that frame 0 is received, but the acknowledgment for frame 0 is lost. The transmitter can only transmit new frames up to frame $W_S - 1$. Depending on which transmissions arrive without error, R_{next} will be in the range of 1 to W_S. Crucially, the receiver will not receive frame W_S until the acknowledgment for frame 0 has been received at the transmitter. Therefore, any receipt of frame 0 prior to frame W_S indicates a duplicate transmission of frame 0.

Figure 5.17 shows that the performance of Go-Back-N ARQ can be improved by having the receiver send a NAK message immediately after the first out-of-sequence frame is received. The NAK with sequence number R_{next} acknowledges all frames up to $R_{next} - 1$ and informs the transmitter that an error

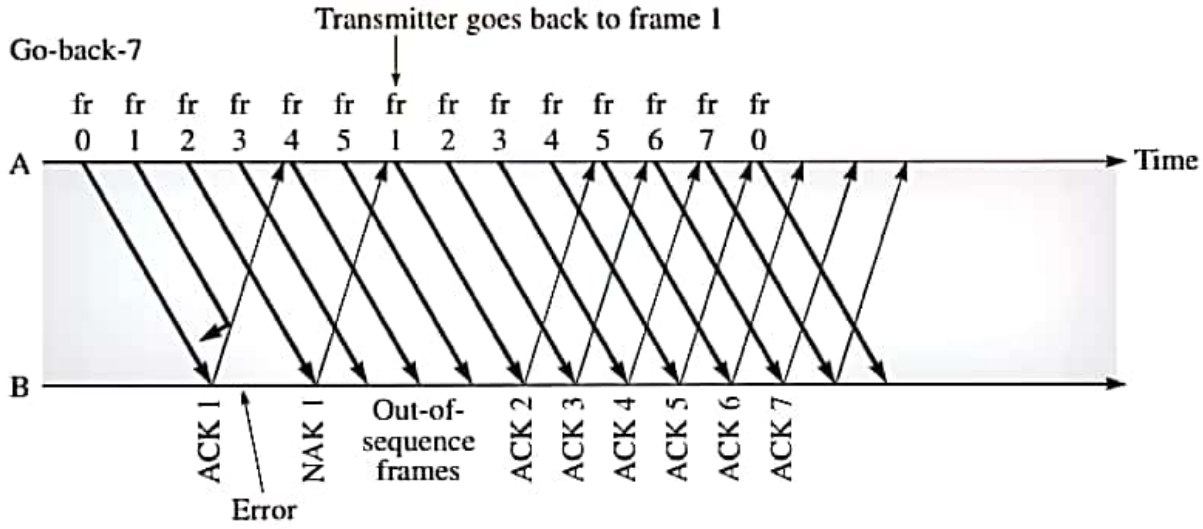


FIGURE 5.17 NAK error recovery

condition has been detected in frame R_{next}. The NAK informs the transmitter that the receiver is now discarding frames that followed R_{next} and instructs the transmitter to go back and retransmit R_{next} and all subsequent packets. By sending the NAK, the receiver sets a condition that cannot be cleared until frame R_{next} is received. For this reason, the receiver is allowed to send only one NAK for any given frame. If the NAK request fails, that is, the NAK is lost or the response transmission fails, then the receiver will depend on the time-out mechanism to trigger additional retransmissions. Note that in general the NAK procedure results in having the transmitter go back less than W_S frames.

Finally we consider the case where the information flow is bidirectional. The transmitter and receiver functions of the modified Go-Back-N protocol are now implemented by both stations A and B. In the direct implementation the flow in each direction consists of information frames and control frames. Many of the control frames can be eliminated by **piggybacking** the acknowledgments in the headers of the information frames as shown in Figure 5.18.

This use of piggybacking can result in significant improvements in the use of bandwidth. When a receiver accepts an error-free frame, the receiver can insert the acknowledgment into the next departing information frame. If no information frames are scheduled for transmission, the receiver can set an **ACK timer** that defines the maximum time it will wait for the availability of an information frame. When the time expires a control frame will be used to convey the acknowledgment.

In the bidirectional case the receiver handles out-of-sequence packets a little differently. A frame that arrives in error is ignored. Subsequent frames that are out of sequence but error free are discarded *after* the ACK (i.e., R_{next}) has been examined. Thus R_{next} is used to update the local S_{last}.

The I-frame time-out value should be selected so that it exceeds the time normally required to receive a frame acknowledgment. As shown in Figure 5.19, this time period includes the round-trip propagation delay $2T_{prop}$, two maximum-length frame transmission times on the reverse channel $2T_f^{max}$, and the frame

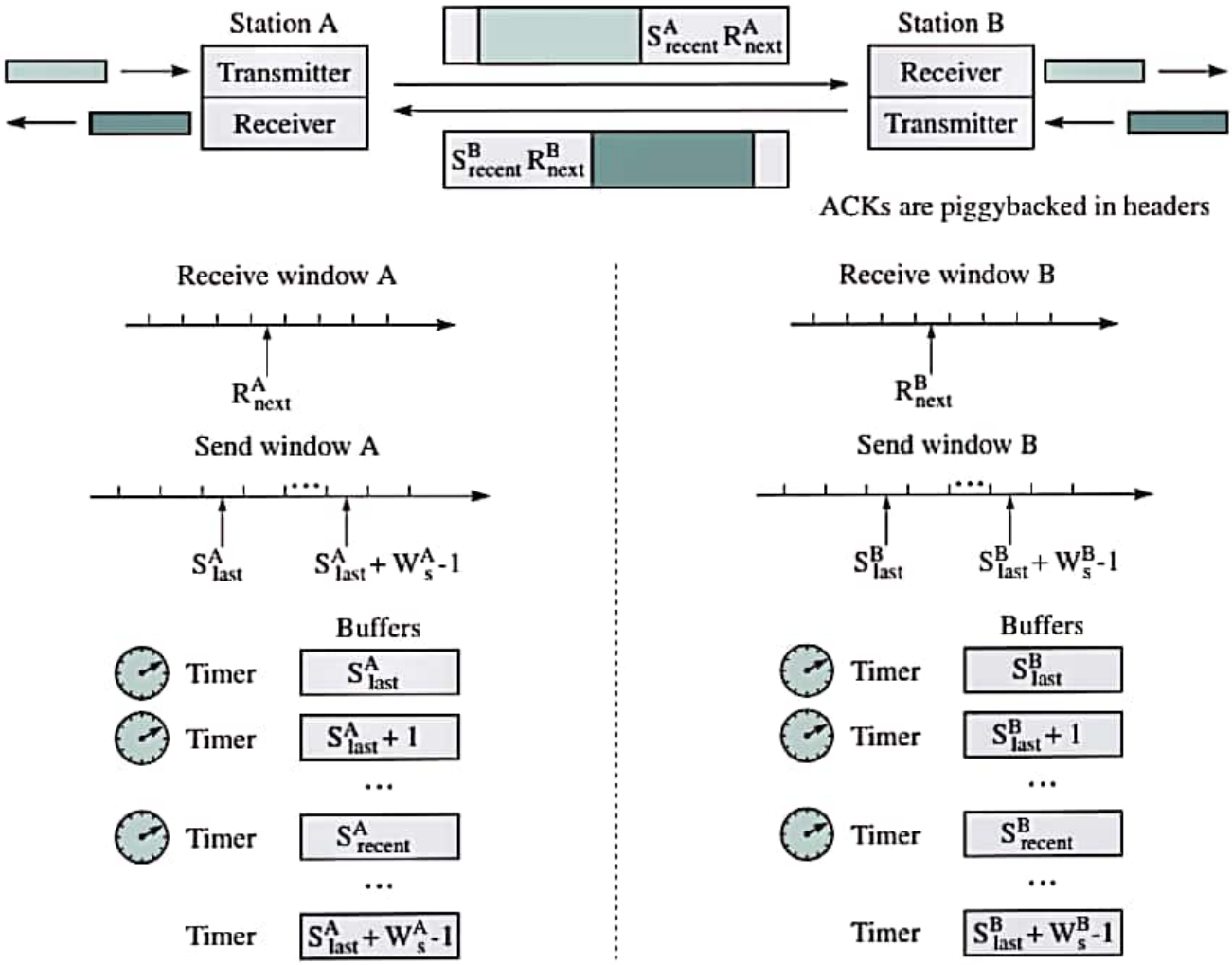


FIGURE 5.18 System parameters in bidirectional Go-Back-N ARQ

processing time T_{proc}.[5] The transmission time of the frame in the forward direction and the ACK time-out are absorbed by the aforementioned frame transmission times.

$$T_{out} \simeq 2T_{prop} + 2T_f^{max} + T_{proc}$$

As in the case of Stop-and-Wait ARQ, the performance of the modified Go-Back-N ARQ protocol is affected by errors in the information frames and errors in the acknowledgments. The transmission of long information frames at the receiver end can delay the transmission of acknowledgments, which can result in the expiry of time-outs at the transmitter end and in turn trigger unnecessary retransmissions. These long delays can be avoided by not using piggybacking and instead sending the short control frames ahead of the long information frames.

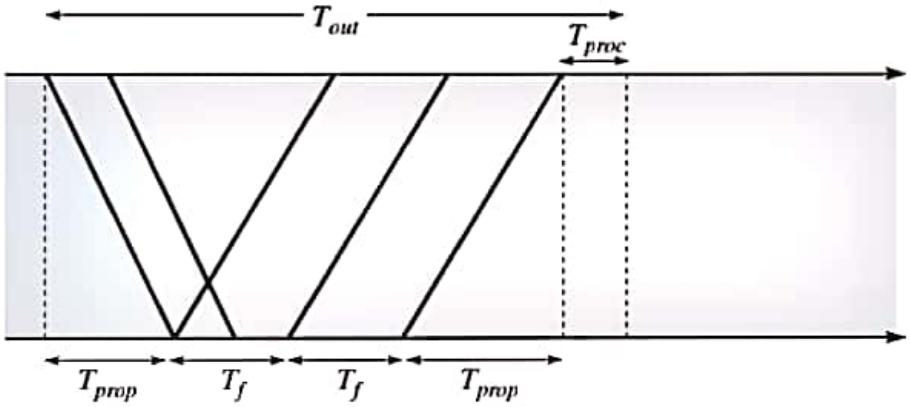


FIGURE 5.19 Calculation of time-out values

Example—HDLC

The High-level Data Link Control (HDLC) is a data link control standard developed by the International Standards Organization. HDLC is a bit-oriented protocol that uses the CRC-16 and CRC-32 error-detection codes discussed in Chapter 3. HDLC can be operated so that it uses Go-Back-N ARQ, as discussed in this section, or Selective Repeat ARQ, which is discussed in section 5.2.3. HDLC is discussed in detail later in this chapter.

The most common use of HDLC is the Link Access Procedure—Balanced (LAPB) discussed later in this chapter. Several variations of the LAPB are found in different applications. For example, Link Access Procedure—Data (LAPD) is the data link standard for the data channel in ISDN. Another example is Mobile Data Link Protocol (MDLP), which is a variation of LAPD that was developed for Cellular Digital Packet Data systems that transmit digital information over AMPS and digital cellular telephone networks.

Example—V.42 Modem Standard

Error control is essential in telephone modem communications to provide protection against disturbances that corrupt the transmitted signals. The ITU-T V.42 standard was developed to provide error control for transmission over modem links. V.42 specifies the Link Access Procedure for Modems (LAPM) to provide the error control. LAPM is derived from HDLC and contains extensions for modem use. V.42 also supports the Microcom Network Protocol (MNP) error-correction protocols as an option. Prior to the development of V.42, these early protocols were de facto standards for error control in modems.

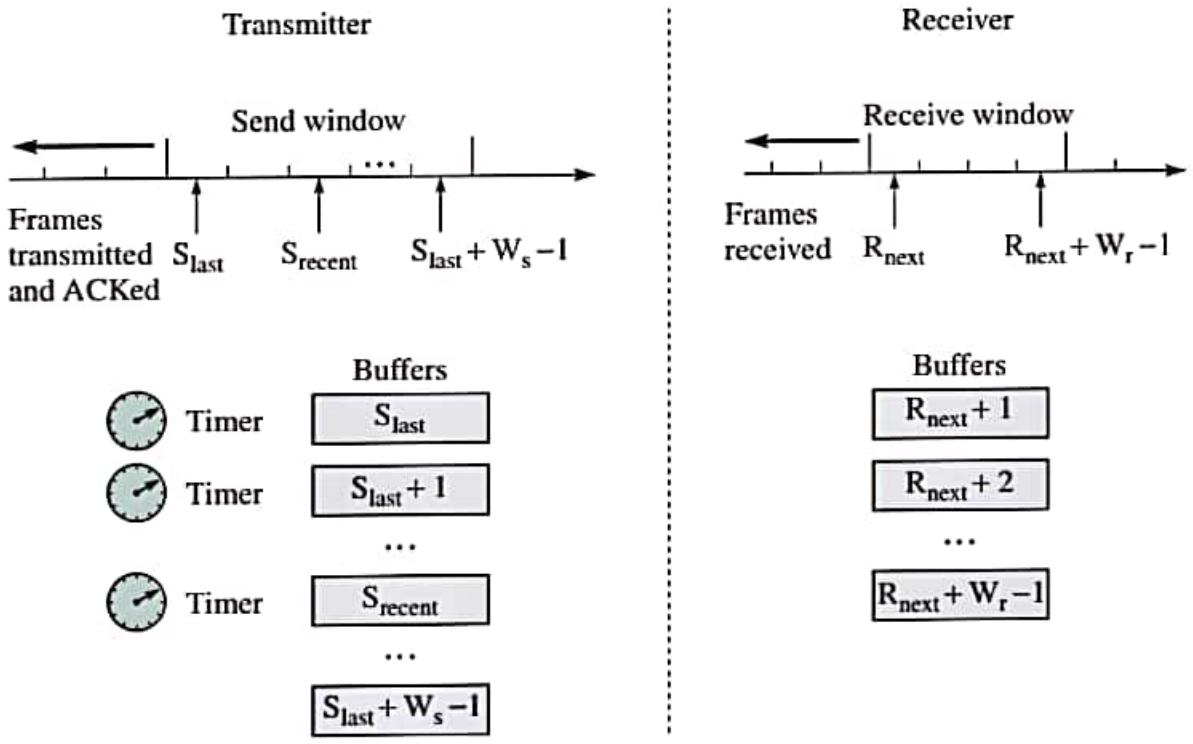
5.2.3 Selective Repeat ARQ

In channels that have high error rates, the Go-Back-N ARQ protocol is inefficient because of the need to retransmit the frame in error and all the subsequent

frames. A more efficient ARQ protocol can be obtained by adding two new features: first, the receive window is made larger than one frame so that the receiver can accept frames that are out of order but error free; second, the retransmission mechanism is modified so that only individual frames are retransmitted. We refer to this protocol as **Selective Repeat ARQ**. We continue to work under the constraint that the ARQ protocol must deliver an error-free and ordered sequence of packets to the destination.

Figure 5.20 shows that the send window at the transmitter is unchanged but that the receive window now consists of a range of frame numbers spanning from R_{next} to $R_{next} + W_R - 1$, where W_R is the maximum number of frames that the receiver is willing to accept at a given time. As before, the basic objective of the protocol is to advance the values of R_{next} and S_{last} through the delivery of the oldest outstanding frame. Thus ACK frames carry R_{next}, the oldest frame that has not yet been received. The receive window is advanced with the receipt of an error-free frame with sequence number R_{next}. Unlike the case of Go-Back-N ARQ, the receive window may be advanced by several frames. This step occurs when one or more frames that follow R_{next} have already been received correctly and are buffered in the receiver. R_{next} and the following consecutive packets are delivered to the destination at this point.

Now consider the retransmission mechanism in Selective Repeat ARQ. The handling of timers at the transmitter is done as follows. When the timer expires, only the corresponding frame is retransmitted. There is no longer a clear correspondence between the age of the timers and the sequence numbers. This situation results in a considerable increase in complexity. Whenever an out-of-sequence frame is observed at the receiver, a NAK frame is sent with sequence number R_{next}. When the transmitter receives such a NAK frame, it retransmits



the specific frame, namely, R_{next}. Finally, we note that the piggybacked acknowledgment in the information frames continues to carry R_{next}.

For example, in Figure 5.21 when the frame with sequence number 2 finally arrives correctly at the receiver, frames 3, 4, 5, and 6 have already been received correctly. Consequently, the receipt of frame 2 results in a sliding of the window forward by five frames.

Consider now the question of the maximum send window size that is allowed for a given sequence numbering 2^m. In Figure 5.22 we show an example in which the sequence numbering is $2^2 = 4$ and in which the send windows and receive windows are of size 3. Initially station A transmits frames 0, 1, and 2.

All three frames arrive correctly at station B, and so the receive window is advanced to {3, 0, 1}. Unfortunately all three acknowledgments are lost, so when the timer for frame 0 expires, frame 0 is retransmitted. The inadequacy of the window size now becomes evident. Upon receiving frame 0, station B cannot determine whether it is the old frame 0 or the new frame 0. So clearly, send and receive windows of size $2^m - 1$ are too large.

It turns out that the maximum allowable window size is $W_S = W_R = 2^{m-1}$, that is, half the sequence number space. To see this we retrace the arguments used in the case of Go-Back-N ARQ. Suppose the window size W_S is 2^{m-1} or less and assume that the current send window is 0 to $W_S - 1$. Suppose also that the initial receive window is 0 to $W_S - 1$. Now suppose that frame 0 is received correctly but that the acknowledgment for frame 0 is lost. The transmitter can transmit new frames only up to frame $W_S - 1$. Depending on which transmissions arrive without error, R_{next} will be in the range between 1 and W_S while $R_{next} + W_R - 1$ will be in the range of 1 to $2W_S - 1$. The maximum value of R_{next} occurs when frames 0 through $W_S - 1$ are received correctly, so the value of R_{next} is W_S and the value of $R_{next} + W_R - 1$ increases to $2W_S - 1$. Crucially, the receiver will not receive frame $2W_S$ until the acknowledgment for frame 0 has been received at the transmitter. Any receipt of frame 0 prior to frame $2W_S$ indicates a duplicate transmission of frame 0. Therefore, the maximum size windows when $W_S = W_R$ is 2^{m-1}.

In the next section we develop performance models for the ARQ protocols. We show that Selective Repeat ARQ outperforms Go-Back-N and Stop-and-