Unit 5

Data Link Layer

11-1 FRAMING

The data link layer needs to pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.

Topics discussed in this section:

Fixed-Size Framing
Variable-Size Framing

Fixed Size Framing

- All the frames will be of same size
- Example: ATM wide area network

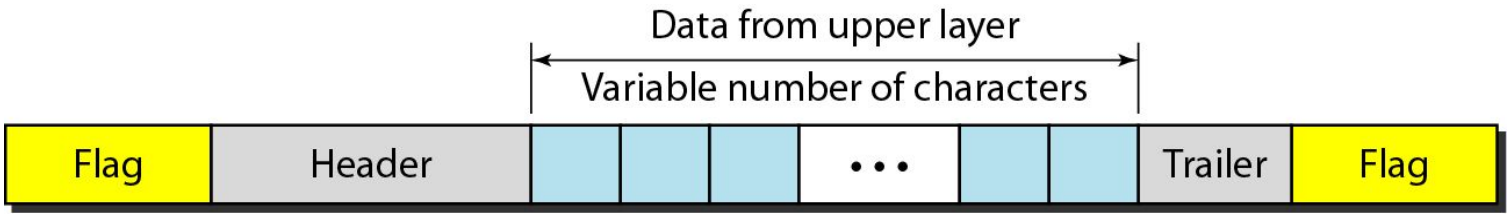
Variable Size Framing

- All the frames will have different size
- There should be a way to define frame boundary like,
  - Character-oriented approach
  - Bit-oriented approach

Character-oriented approach

- In this approach, character-oriented protocol is used
- As per this protocol, 8 bit of flag is used to separate the frame from other.
- Flag is added at the start and end of the frame

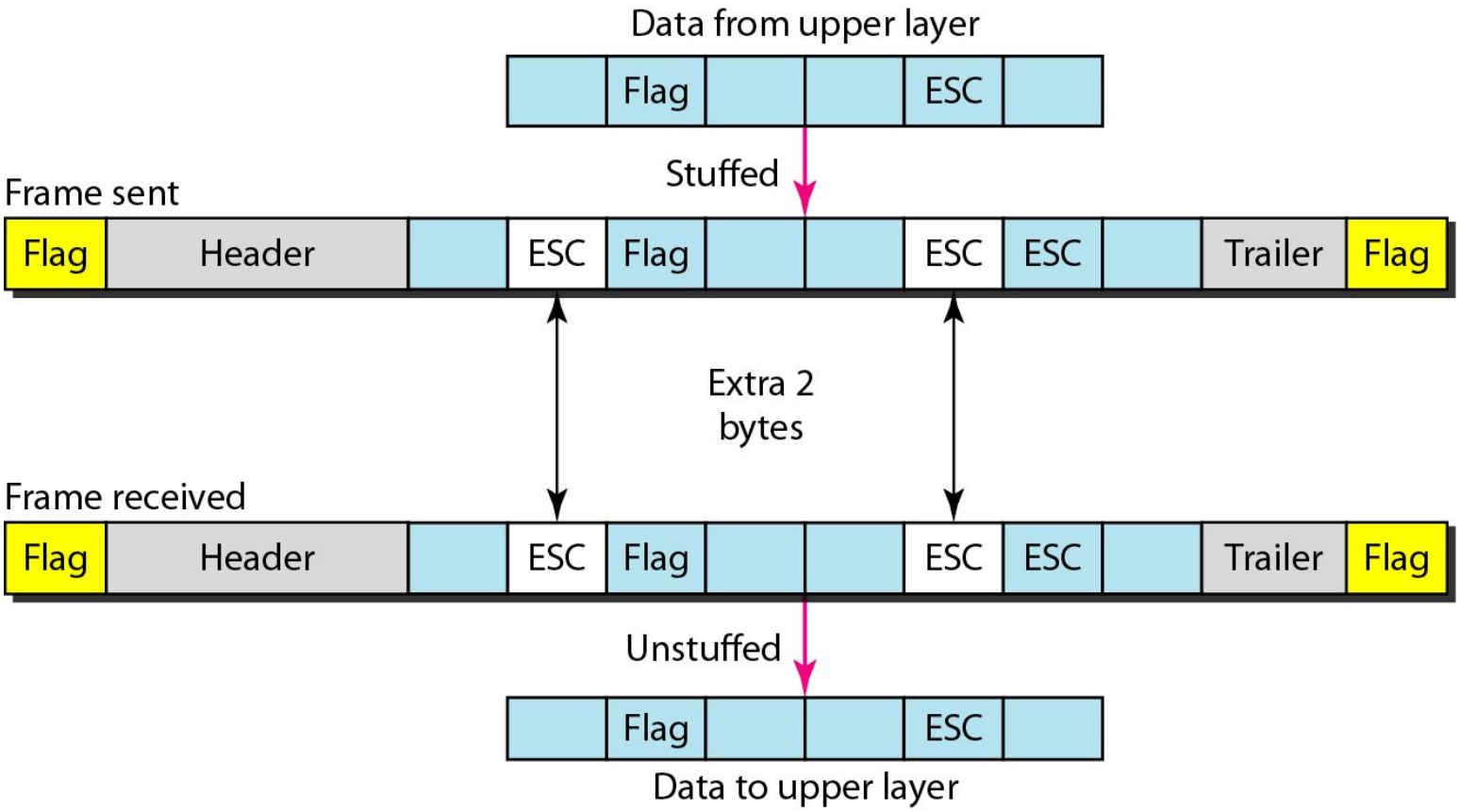
Figure 11.1 *A frame in a character-oriented protocol*



Cont…

- To differentiate the character of flag from the character of data, byte stuffing is used
- In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag.
- The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern
- Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data

Figure 11.2 *Byte stuffing and unstuffing*



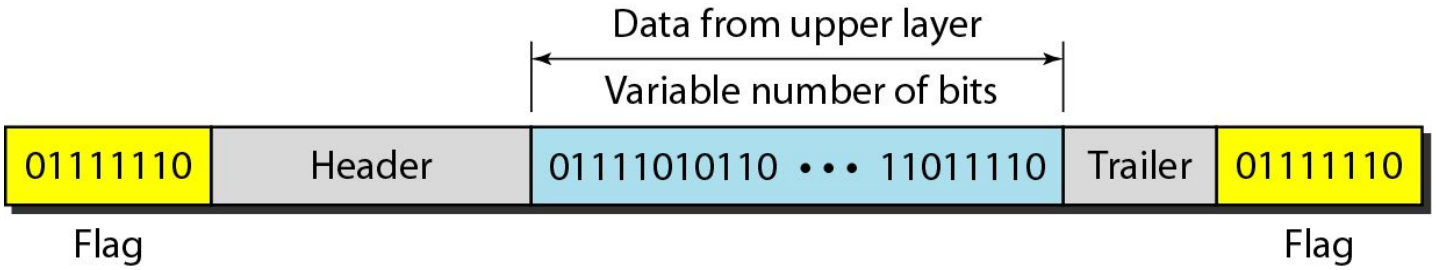
Note

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

Bit-Oriented Protocols

- In this protocol to differentiate the frame, pre-defined bit pattern is added
- Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame

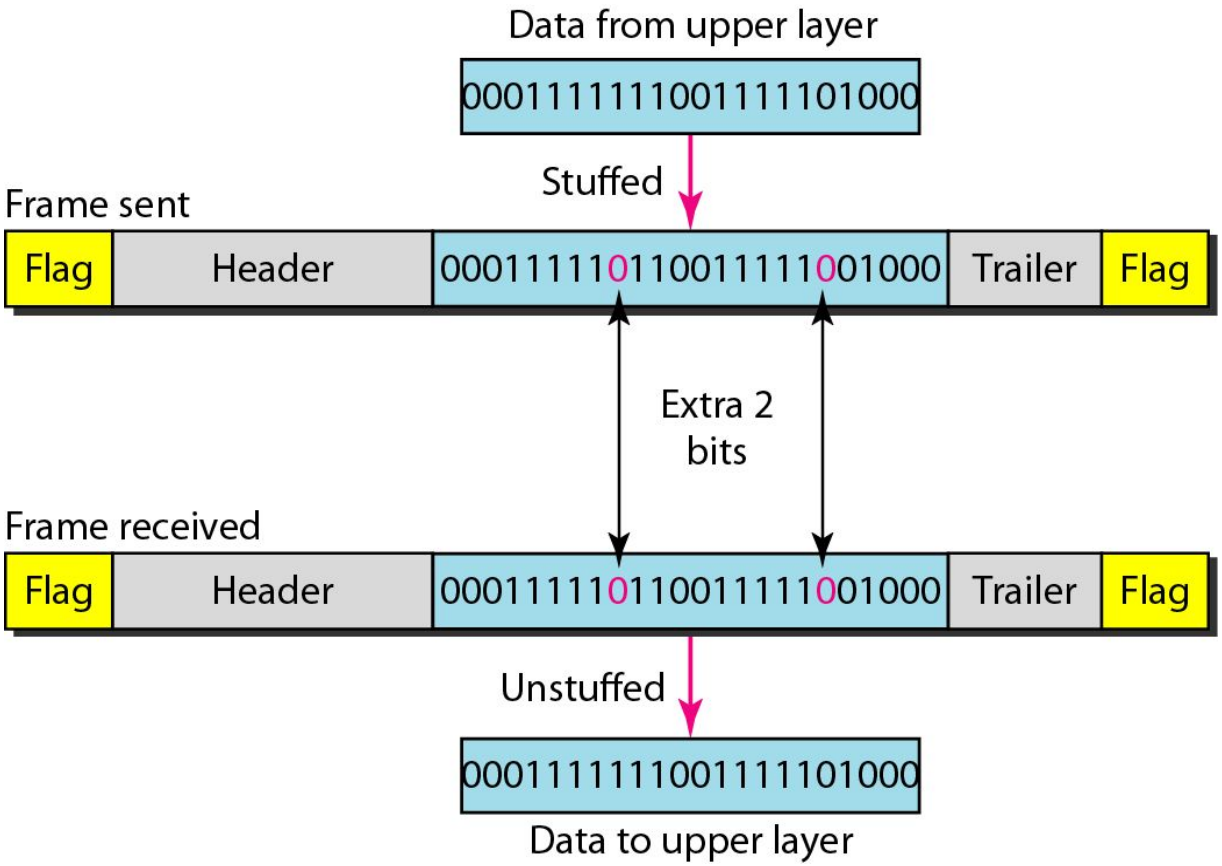
Figure 11.3 *A frame in a bit-oriented protocol*



Note

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 01111110 for a flag.

Figure 11.4 *Bit stuffing and unstuffing*

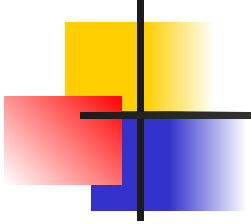


11-2 FLOW AND ERROR CONTROL

The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.

Topics discussed in this section:

Flow Control
Error Control



Note

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

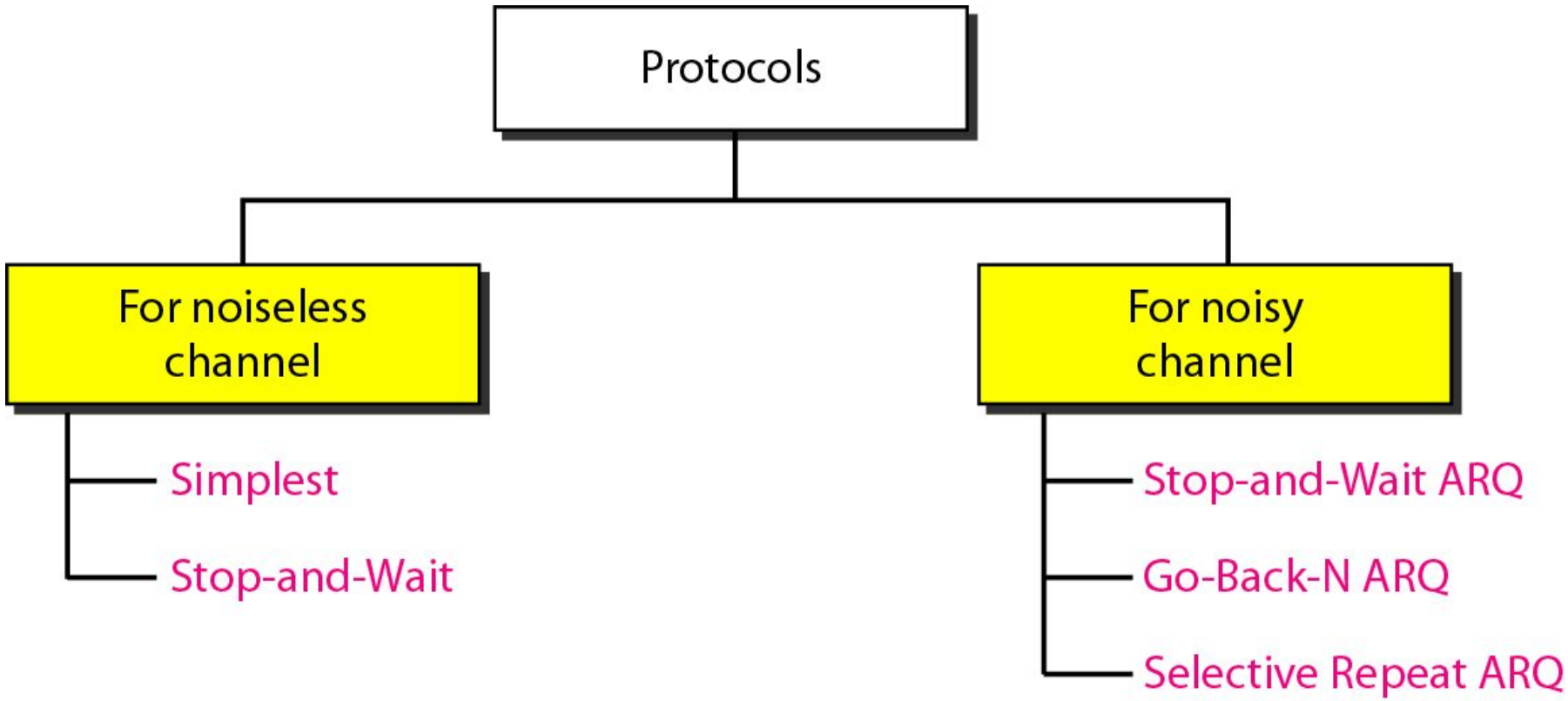
Note

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

11-3 PROTOCOLS

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages. To make our discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



11-4 NOISELESS CHANNELS

Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel.

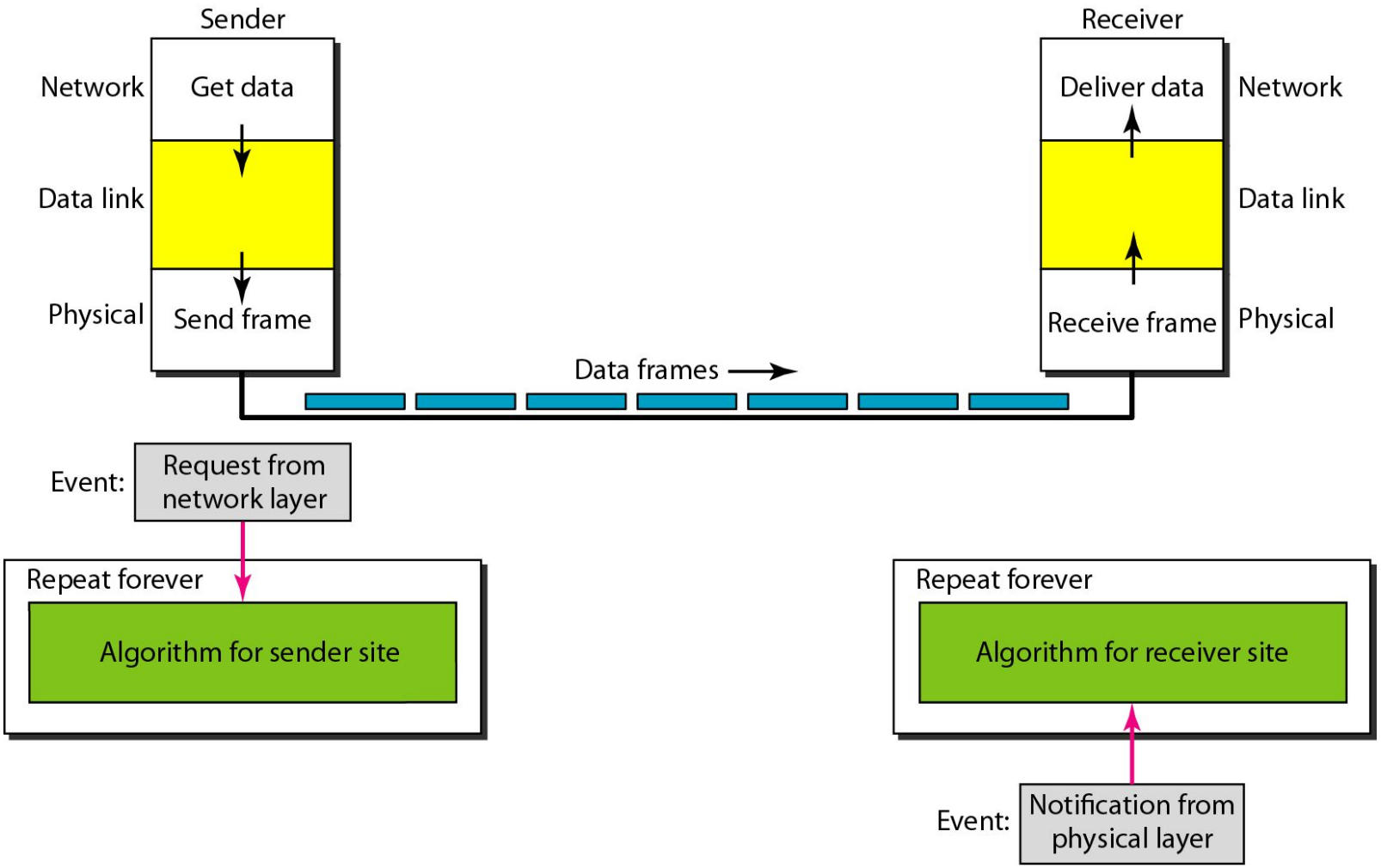
Topics discussed in this section:

Simplest Protocol
Stop-and-Wait Protocol

Simplest Protocol

- Unidirectional
- No error control
- No flow control
- Frame processing time is negligible
- There is no acknowledgement from receiver side

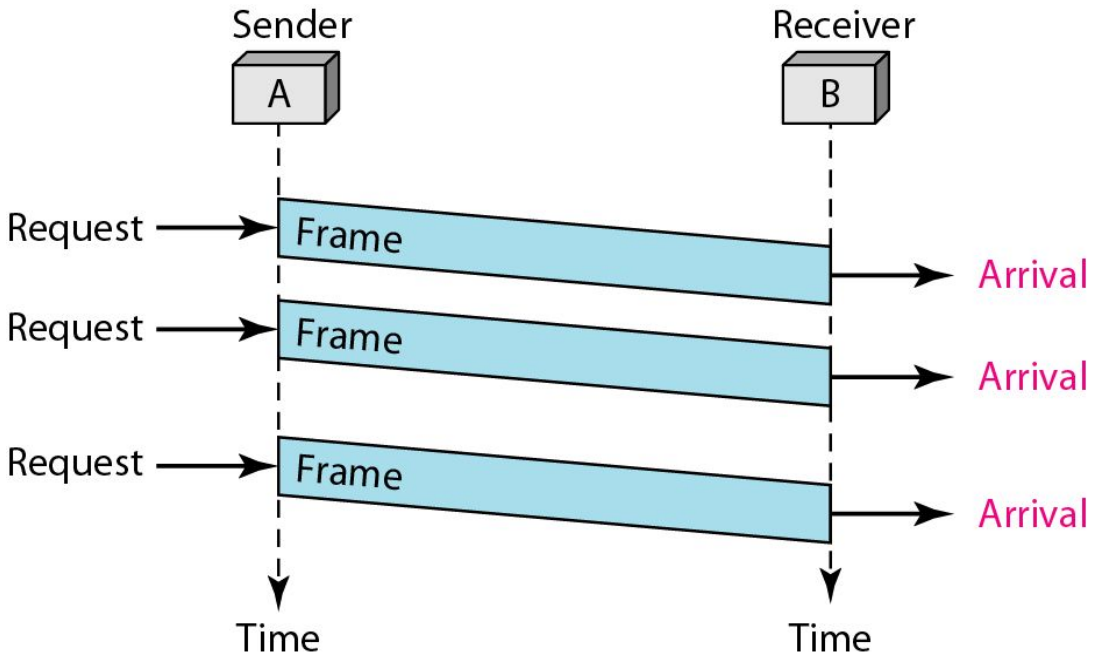
Figure 11.6 *The design of the simplest protocol with no flow or error control*



Example 11.1

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

Figure 11.7 *Flow diagram for Example 11.1*



Stop-and-Wait Protocol

- Unidirectional for data frames only
- Flow control
- No error control
- There is acknowledgment from receiver side

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

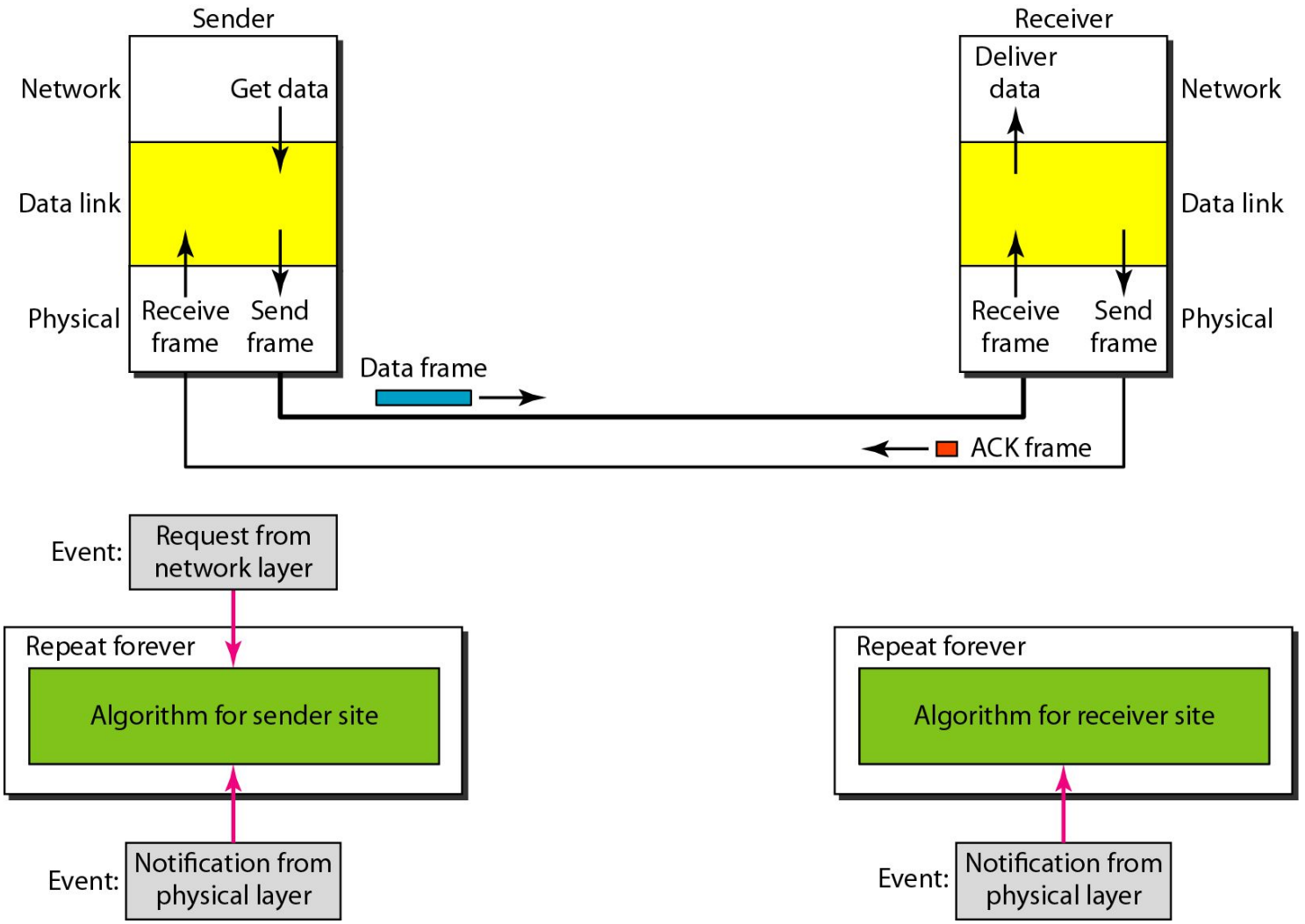
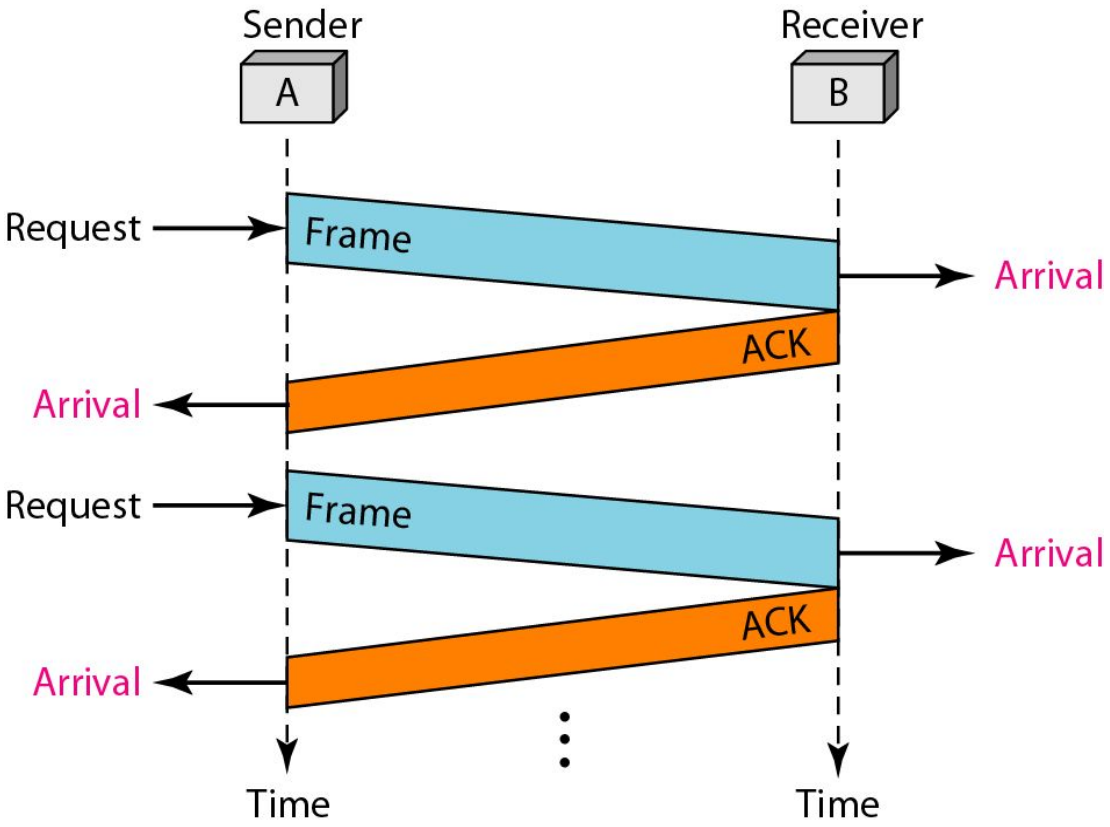


Figure 11.9 *Flow diagram for Example 11.2*



11-5 NOISY CHANNELS

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We discuss three protocols in this section that use error control.

Topics discussed in this section:

Stop-and-Wait Automatic Repeat Request
Go-Back-N Automatic Repeat Request
Selective Repeat Automatic Repeat Request

Stop-and-Wait Automatic Repeat Request

- Error Control Mechanism
- Sequence Number
- Acknowledgment Number

Note

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

Note

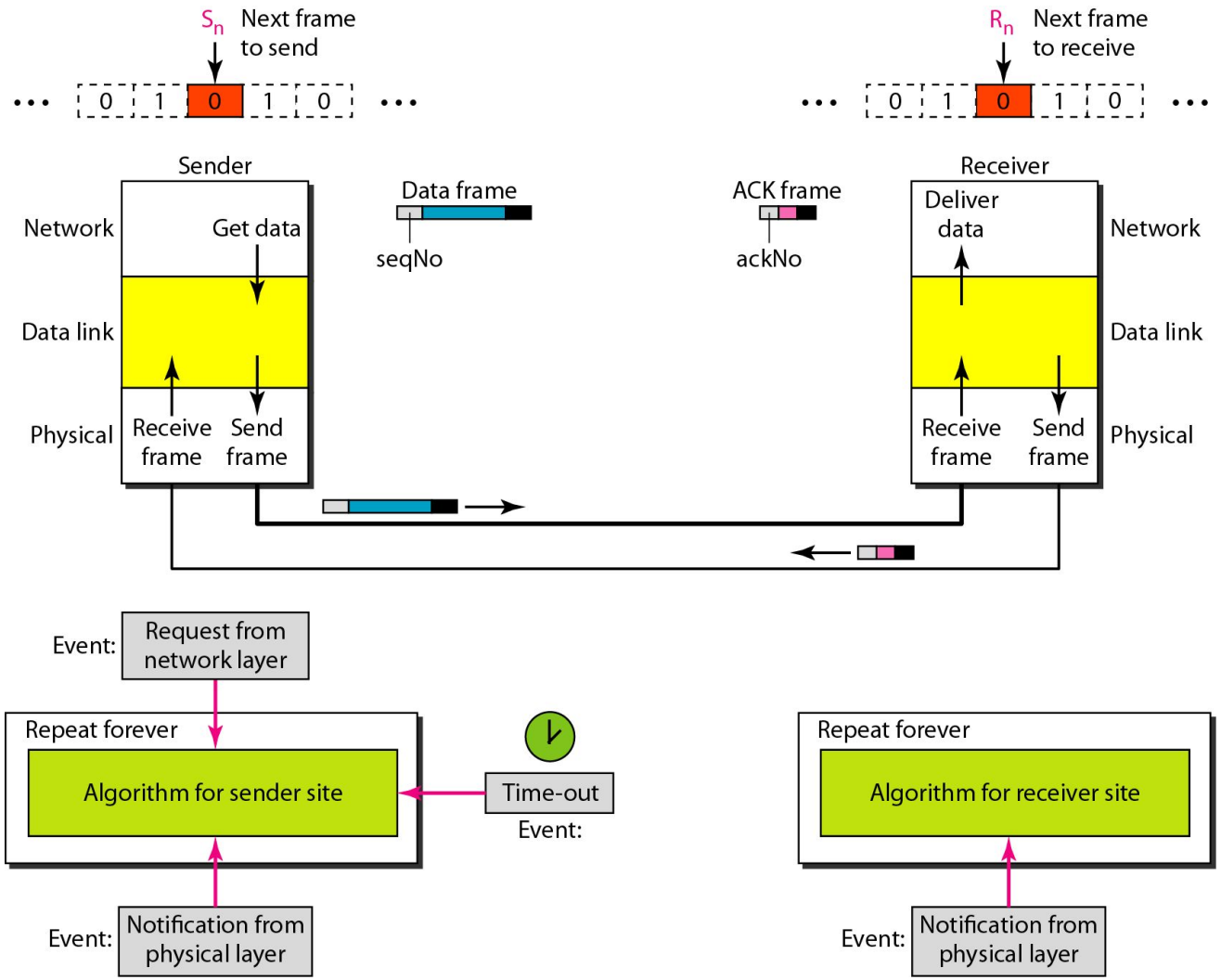
In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

Note

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


Source
Stop & Wait ARQ
Destination
SOURCE
DESTINATION

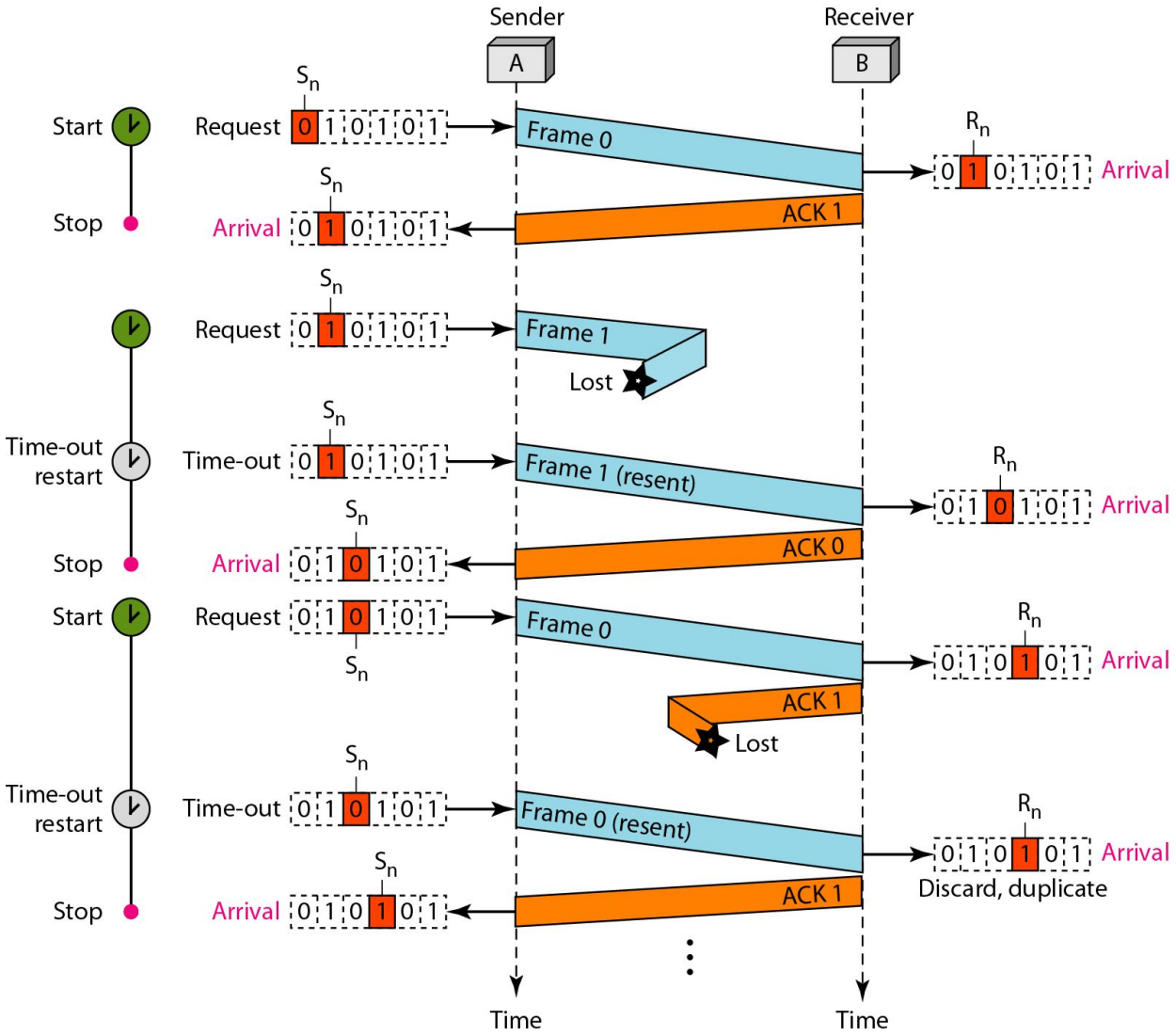
Figure 11.10 *Design of the Stop-and-Wait ARQ Protocol*



Example 11.3

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

Figure 11.11 *Flow diagram for Example 11.3*



Example 11.4

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

Solution

The bandwidth-delay product is

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

Example 11.4 (continued)

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

Example 11.5

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

Solution

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

Example 11.4

During Transmission if every 4^{th} frame is lost then total how many frames will be transmitted to transmit 10 frames using Stop and Wait with ARQ protocol

Solution

13

Go-Back-N Automatic Repeat Request

- Sender Window
- Increase the channel utilization
- Sequence Number
- Timer
- Acknowledgment

Note

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

Note

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

Note

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

Note

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

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

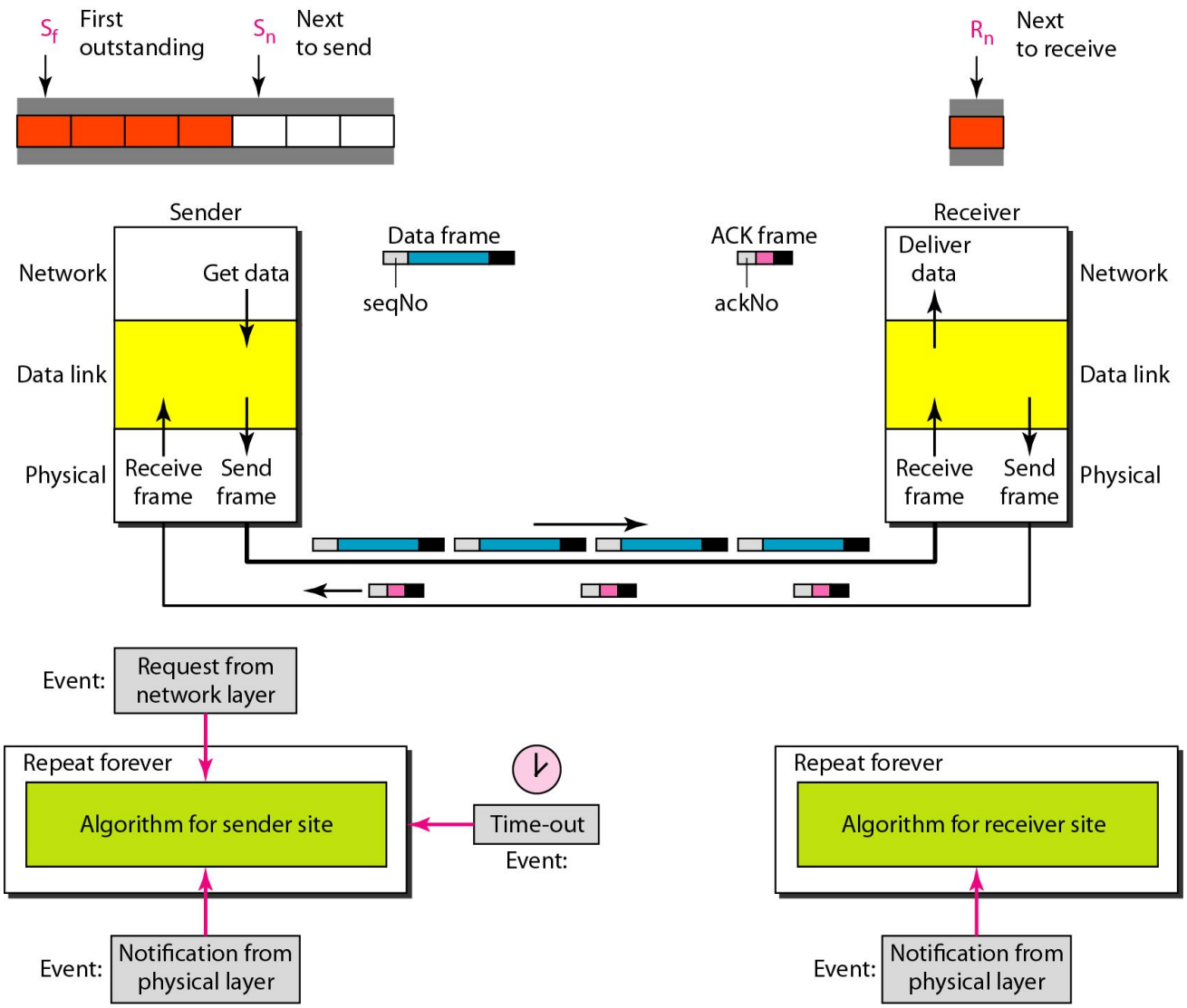
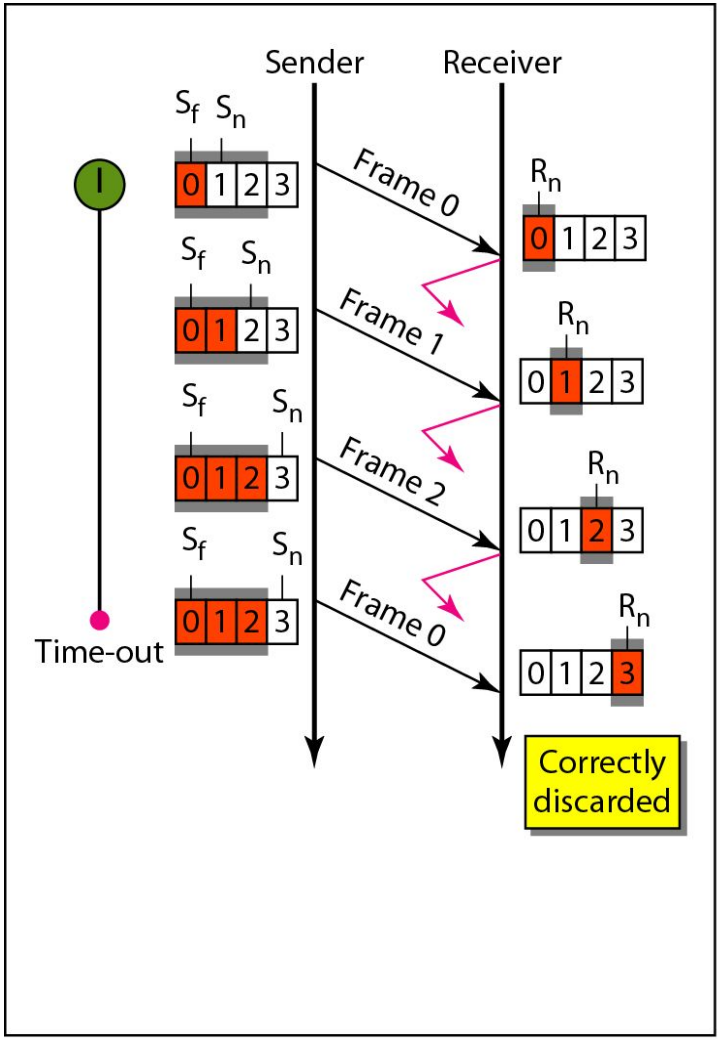
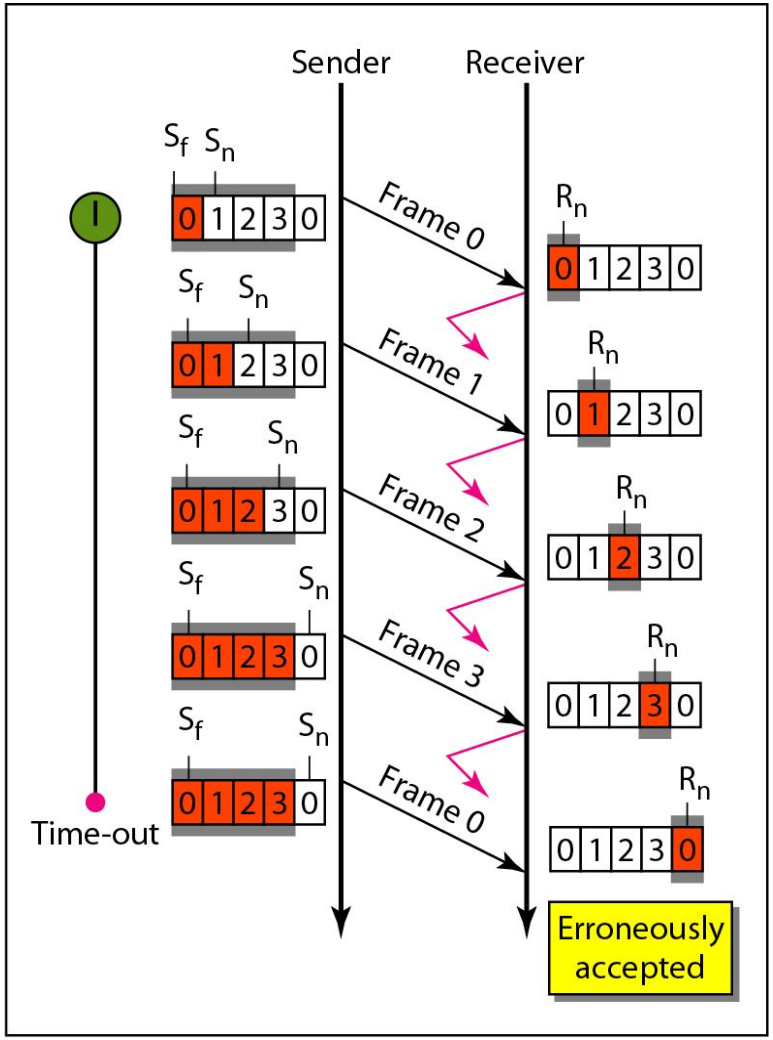


Figure 11.15 *Window size for Go-Back-N ARQ*



a. Window size < 2^m



b. Window size = 2^m

Note

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

Figure 11.17 *Flow diagram for frame lost*

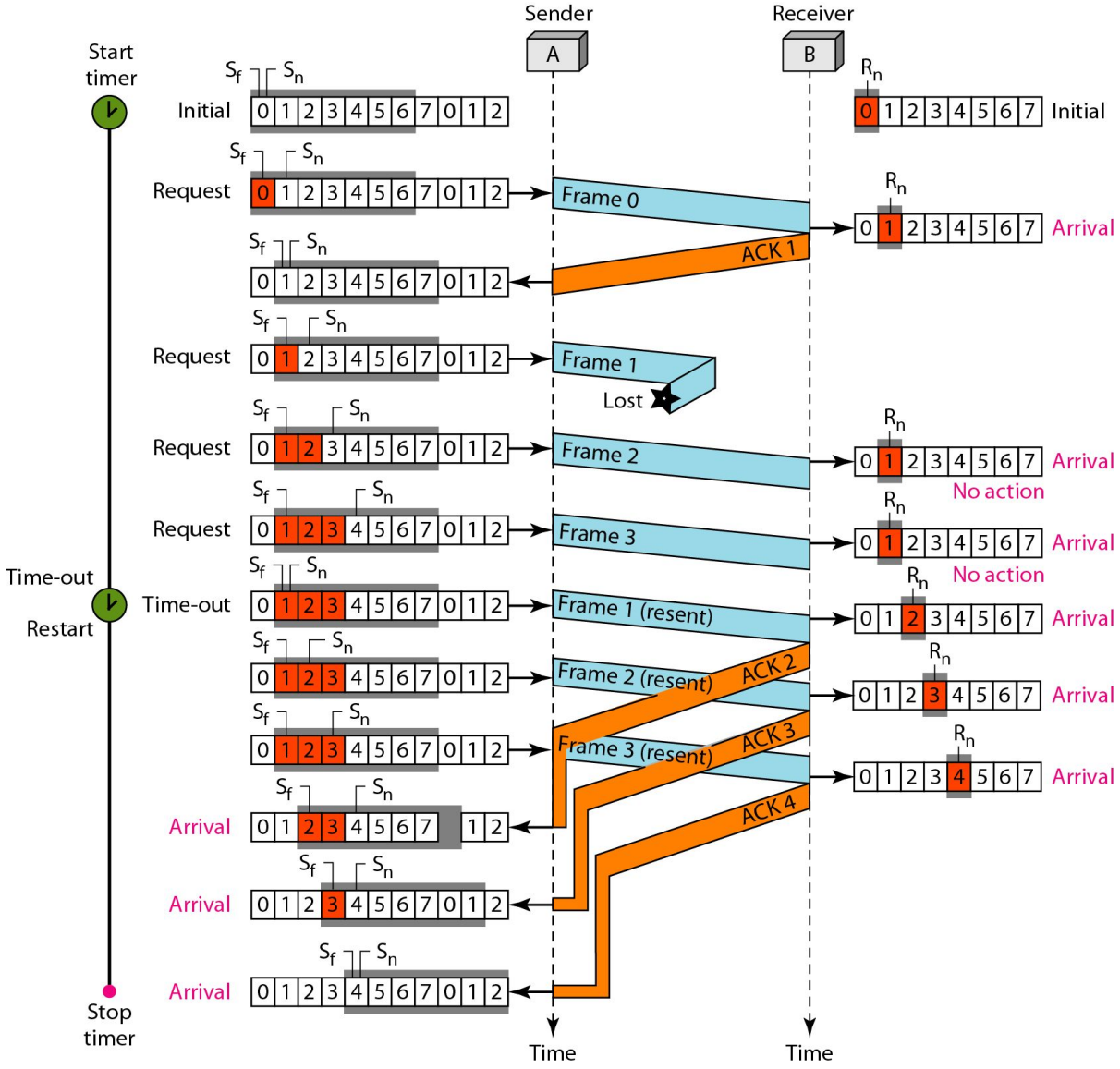
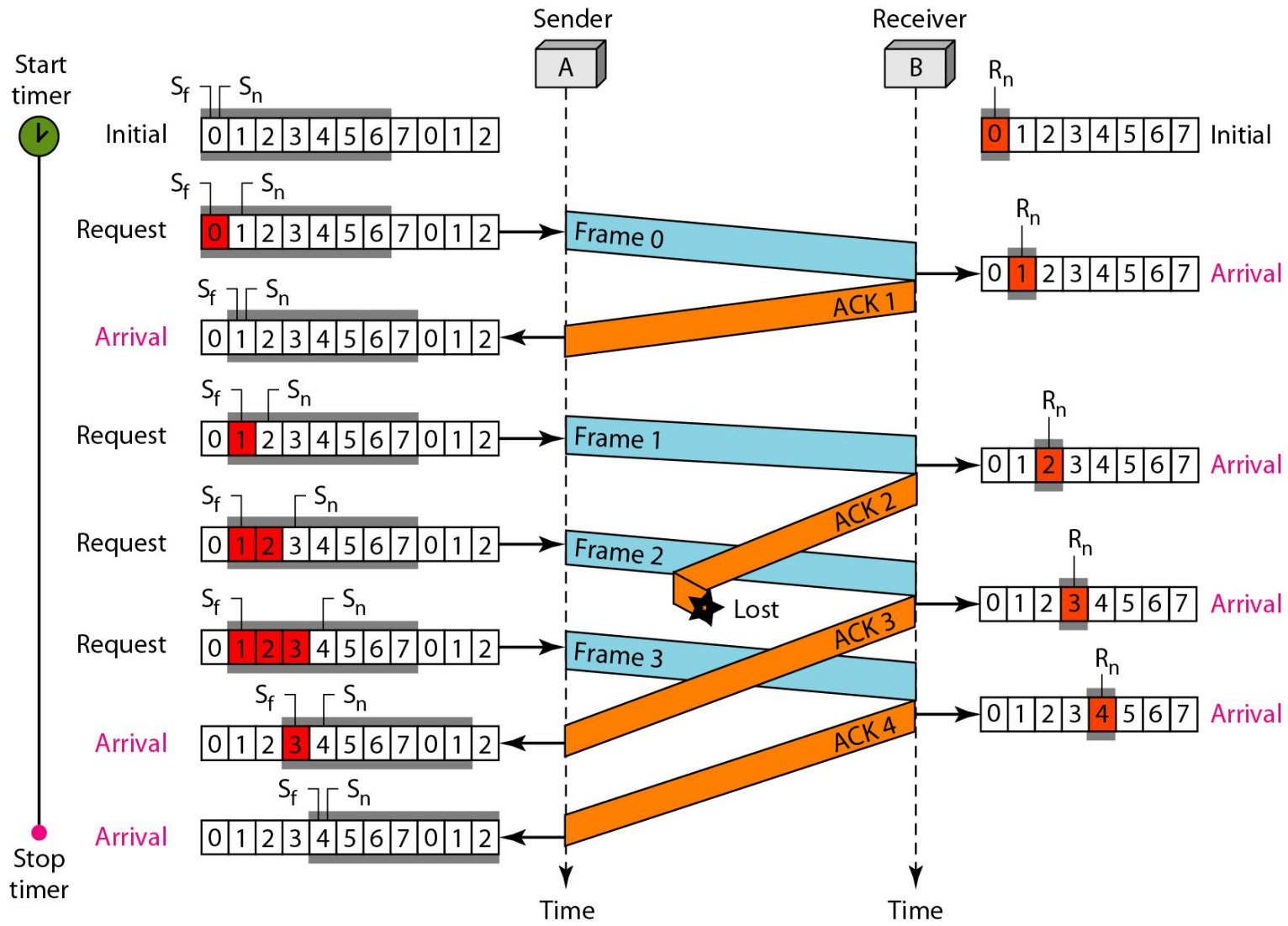


Figure 11.16 *Flow diagram for Acknowledgement lost*



Note

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

Selective Repeat Automatic Repeat Request

- Two window : Sender window, Receiver window
- Two Acknowledgment: ACK, NAK

Figure 11.18 *Send window for Selective Repeat ARQ*

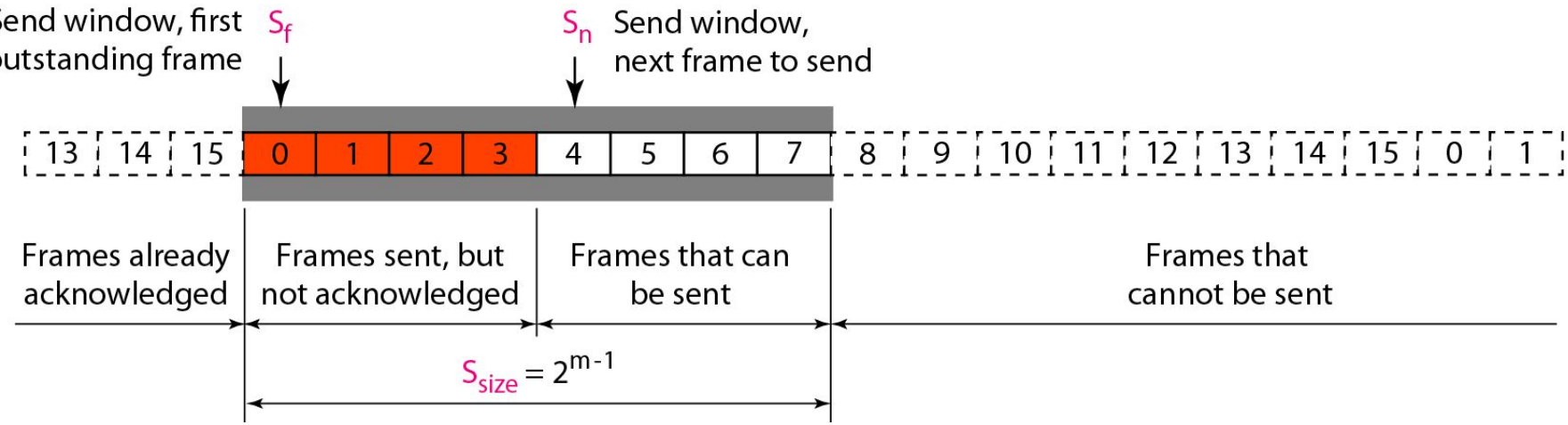


Figure 11.19 *Receive window for Selective Repeat ARQ*

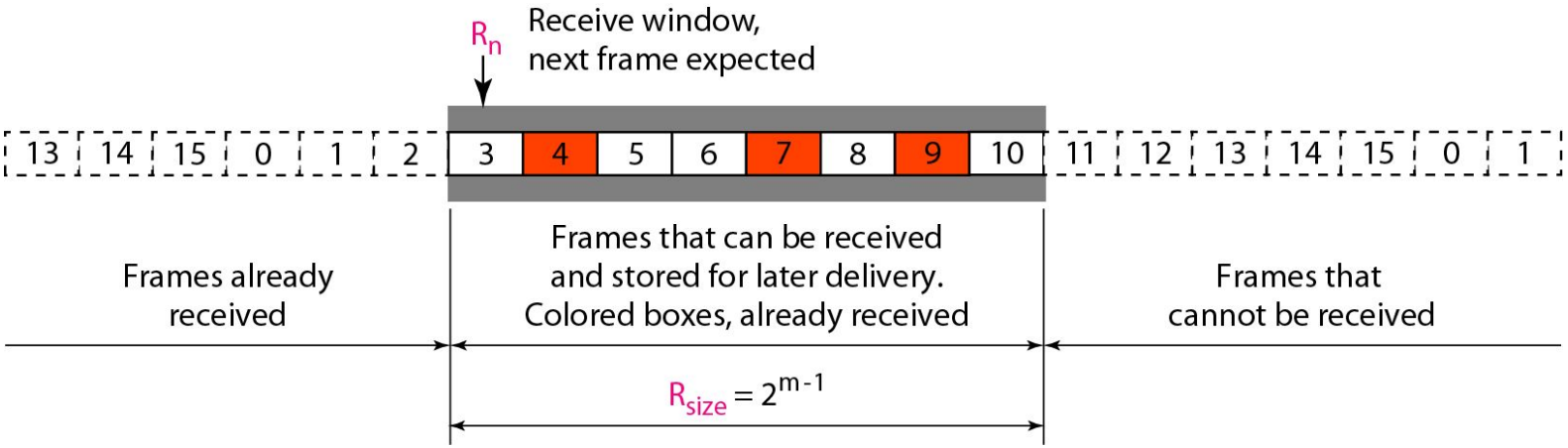


Figure 11.20 *Design of Selective Repeat ARQ*

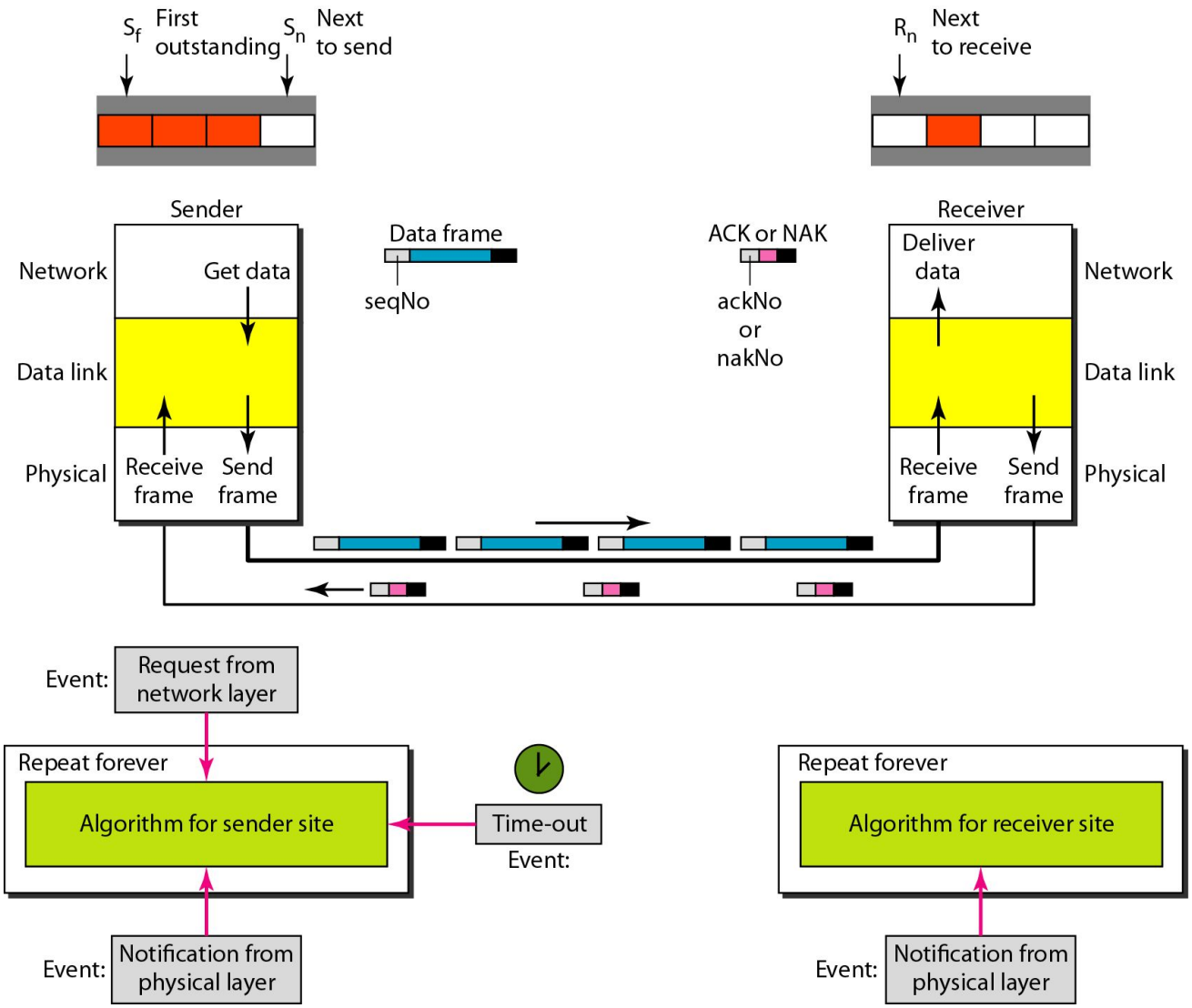
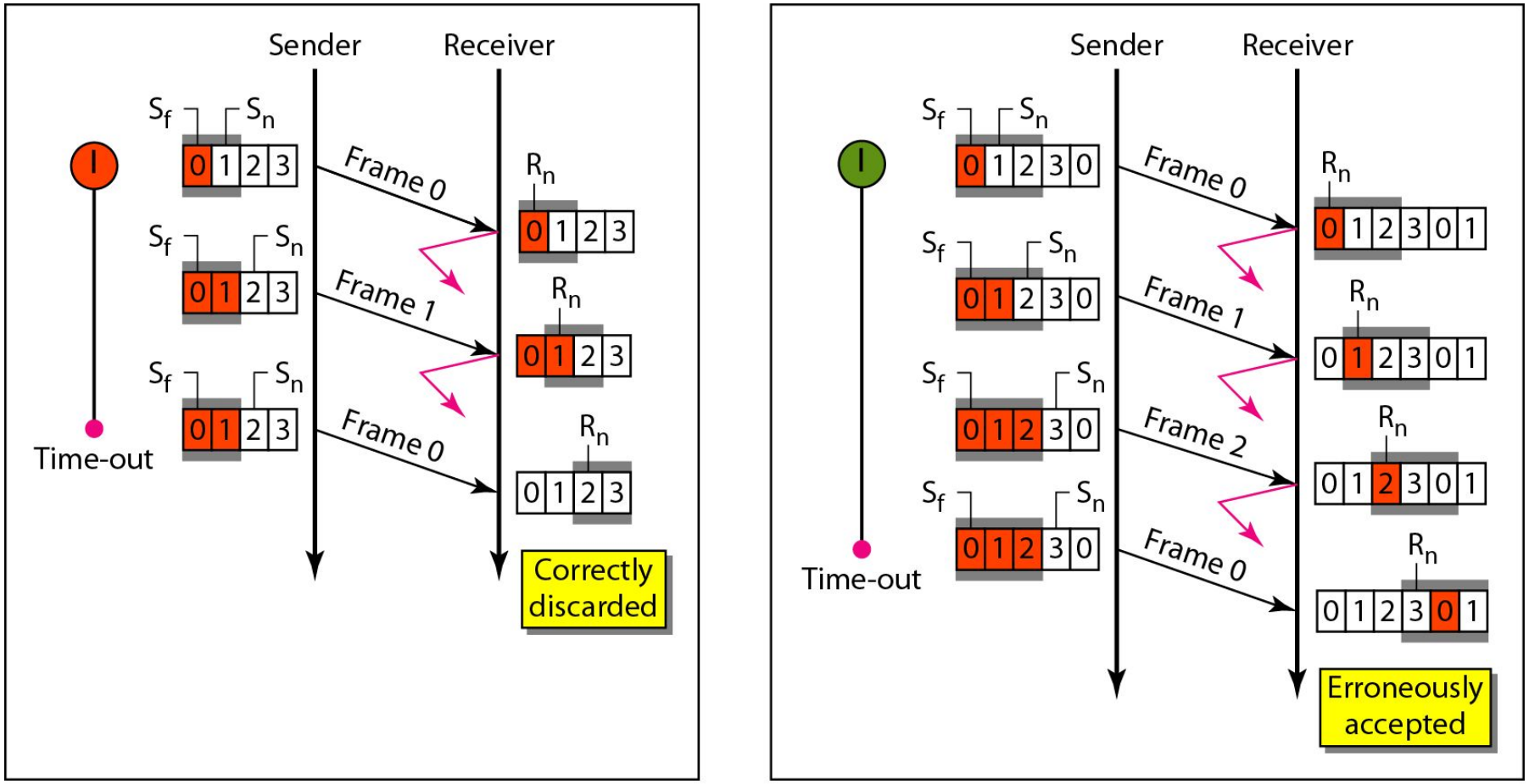


Figure 11.21 *Selective Repeat ARQ, window size*



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

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

Note

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

Figure 11.22 *Delivery of data in Selective Repeat ARQ*

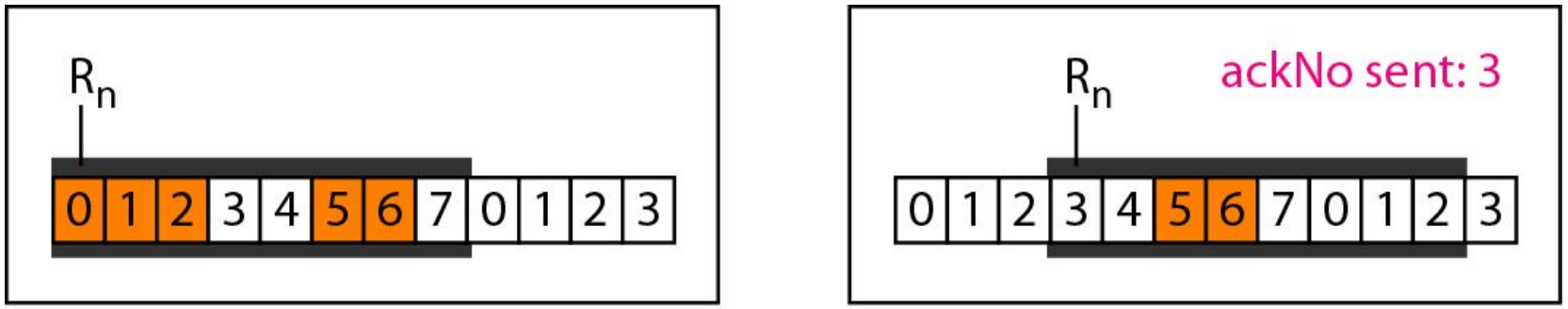
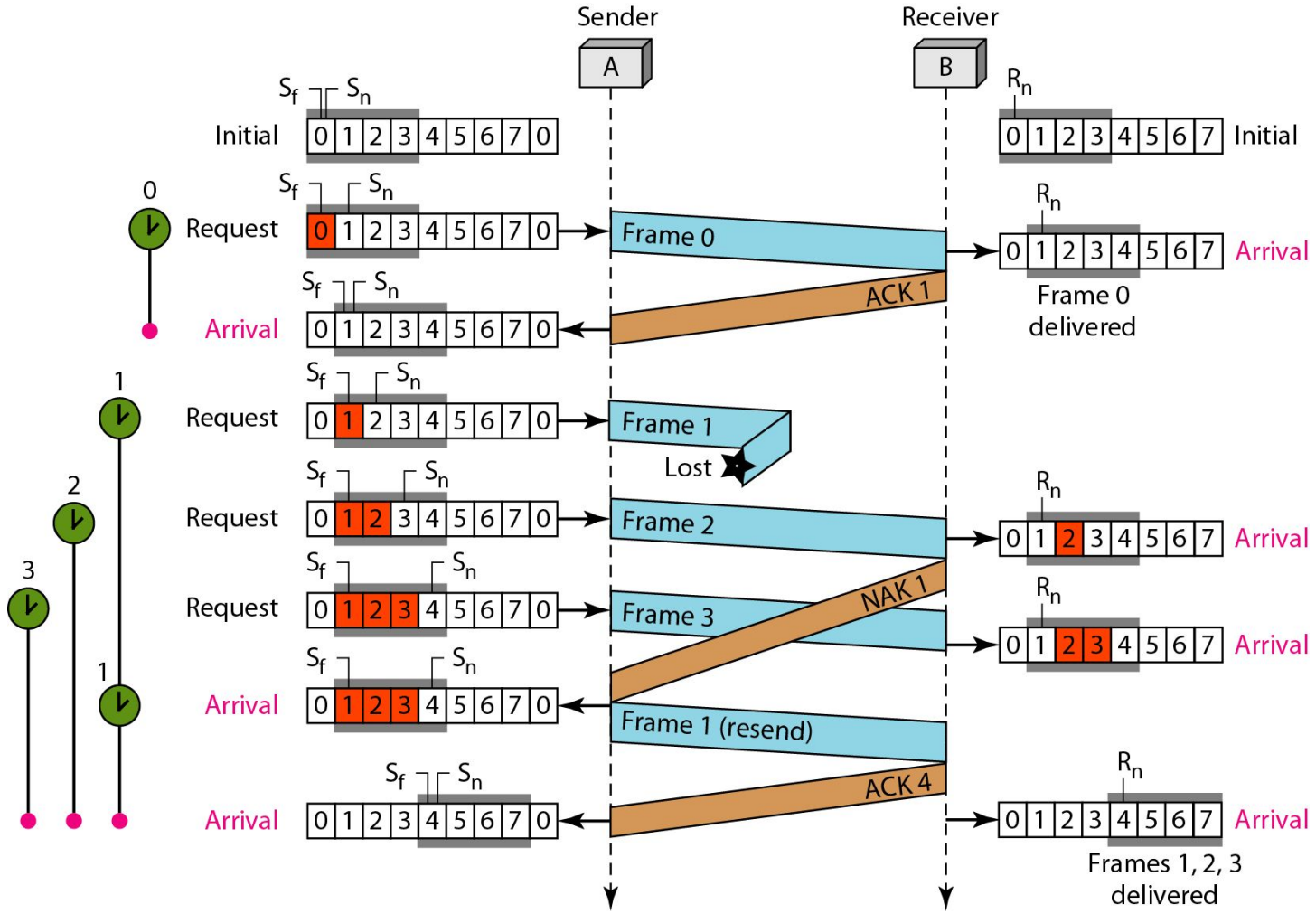


Figure 11.23 *Flow diagram for Example 11.8*



Piggybacking

- For bidirectional data flow, the control information (ACK, NAK) can be transmitted along with data frames to improve the efficiency. This technique is called as Piggybacking

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