Lecture 4

Data Link Control

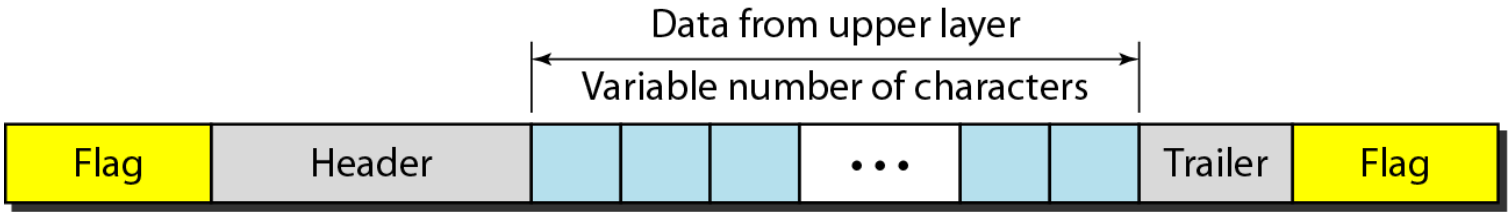
(Chapter 11 of “Data Communications and Networking” [B. A. Forouzan])

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.

Fixed-size framing: no need for defining the boundary. The size itself can serve as a delimiter. For example, ATM network.

Variable-size framing: need to define the beginning and end of a frame.

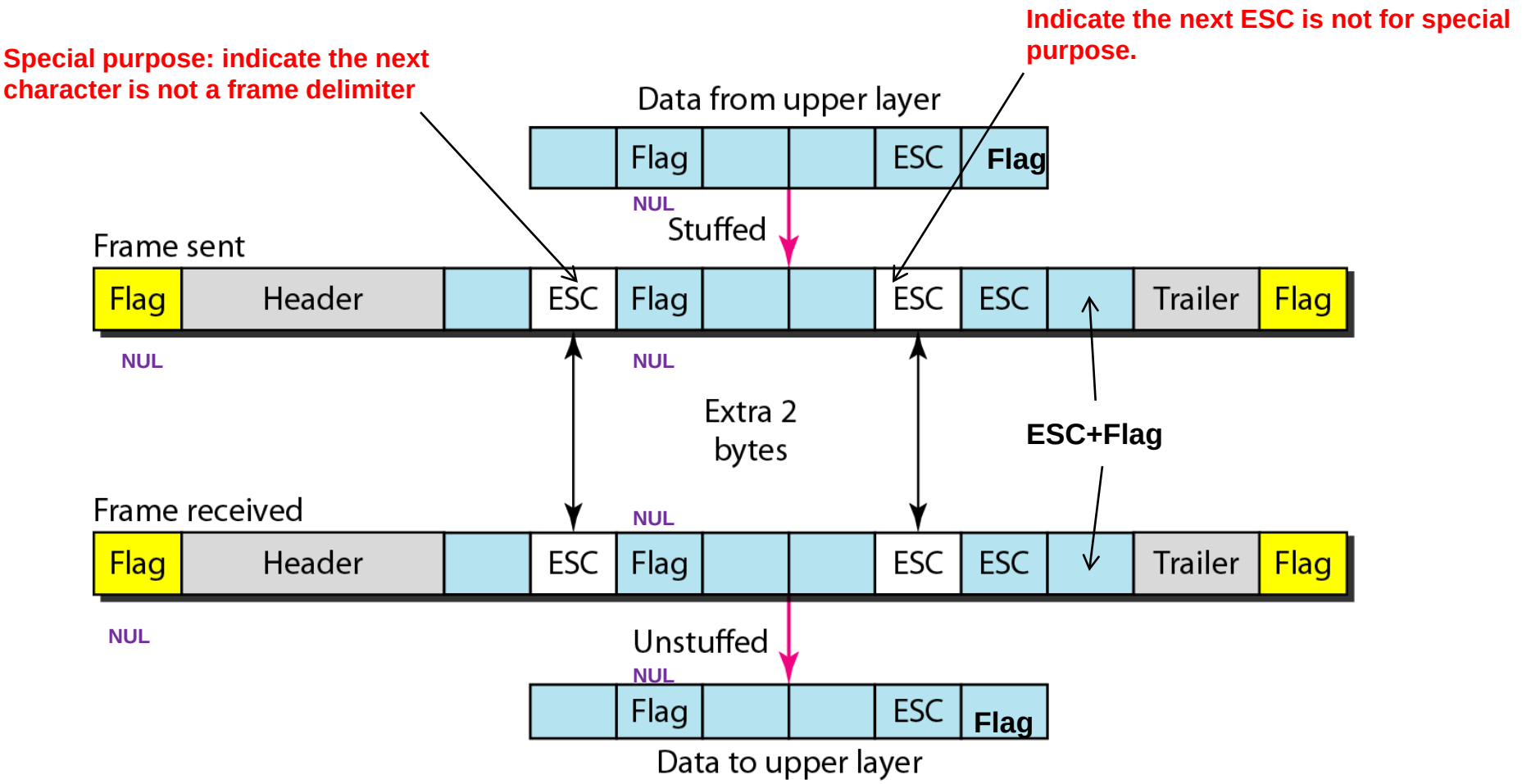


Character-oriented protocol: data are 8-bit characters (ASCII code). Header and trailer are also multiples of 8 bits.

ASCII Table

MSB LSB	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P		p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	“	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EQT	DC4	$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	‘	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	^	n	~
F 1111	SI	US	/	?	O	_	o	DEL

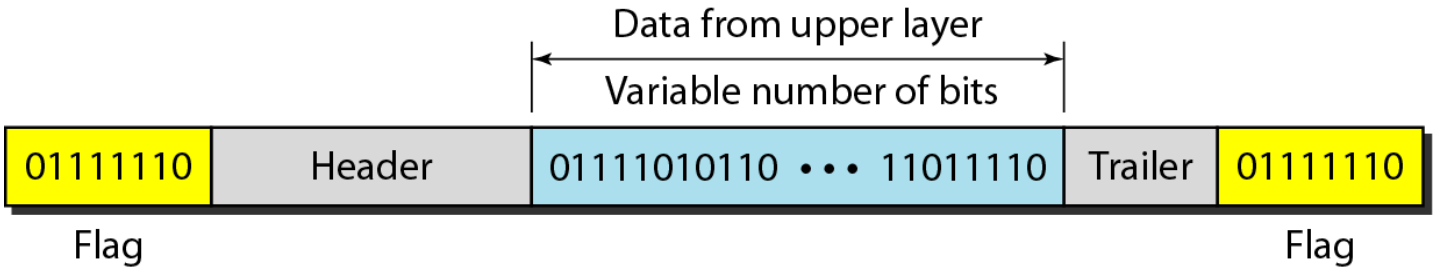
Figure 11.2 *Byte stuffing and unstuffing*



Note

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

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



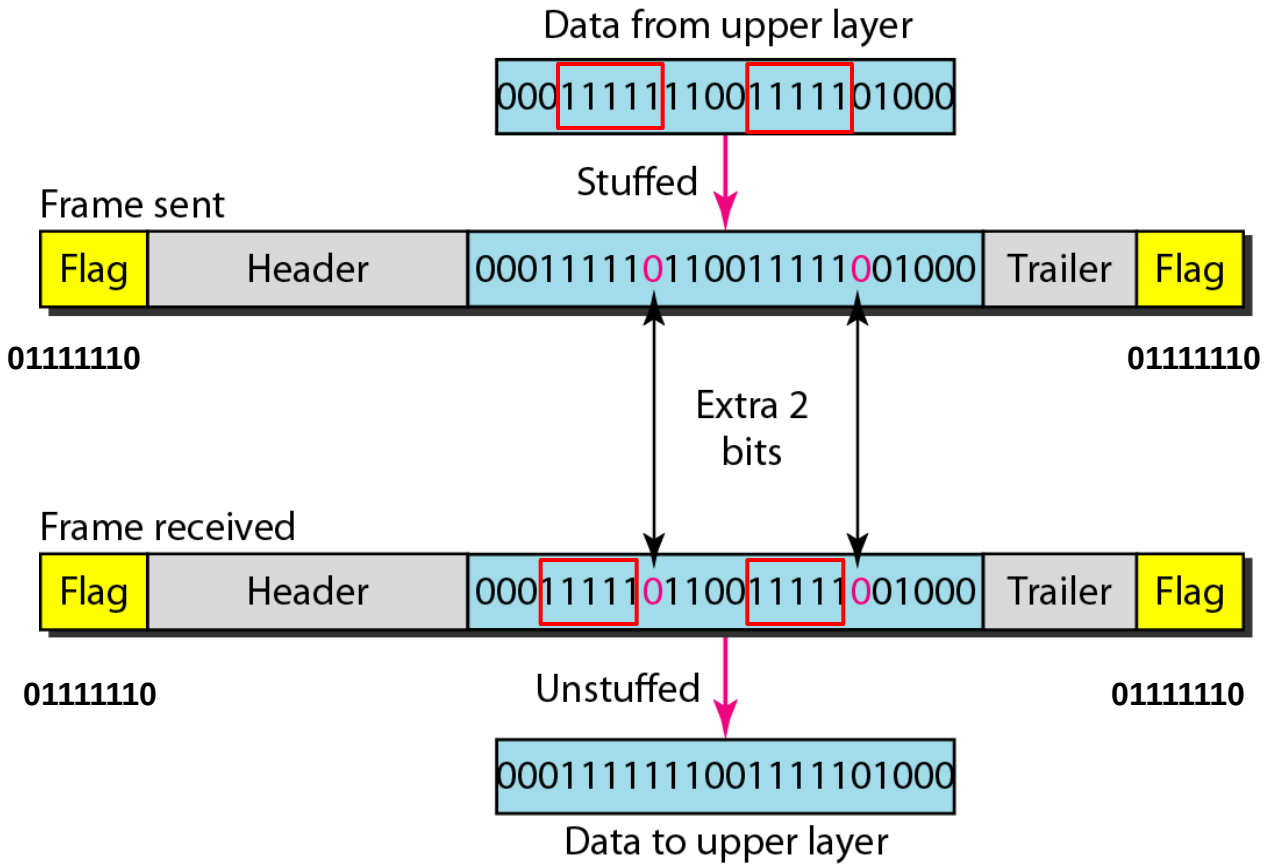
Bit-oriented protocol: data are a sequence of bits (which can be interpreted as text, graphic, audio, video, etc.).

Note

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s in the data.

At the receiver: if five consecutive 1s followed by a 0 bit, then unstuff the 0 bit.

Figure 11.4 *Bit stuffing and unstuffing*



Any other solution?

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.

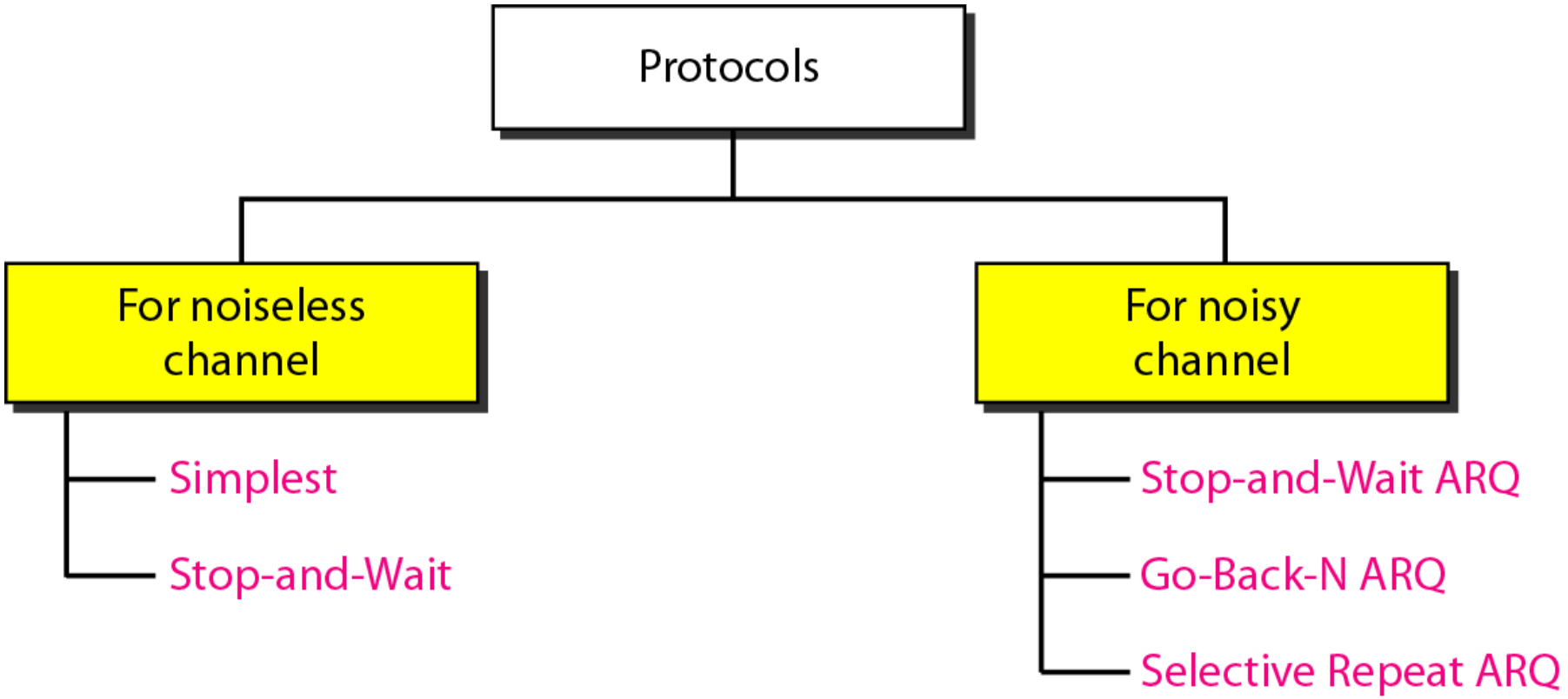
Note

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.

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



11-4 NOISELESS CHANNELS

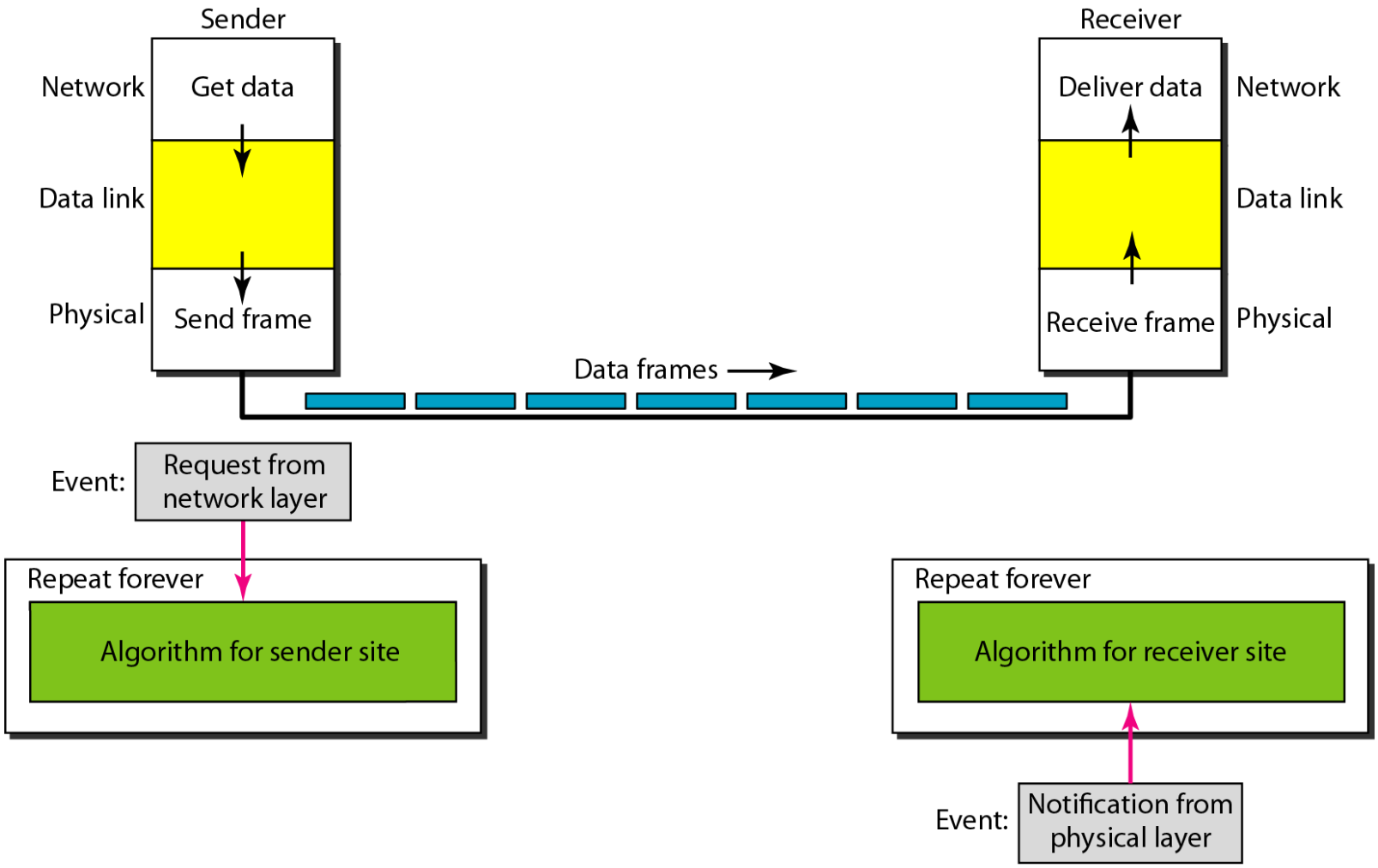
Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel.

Topics discussed in this section:

Simplest Protocol

Stop-and-Wait Protocol

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



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

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

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

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

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*

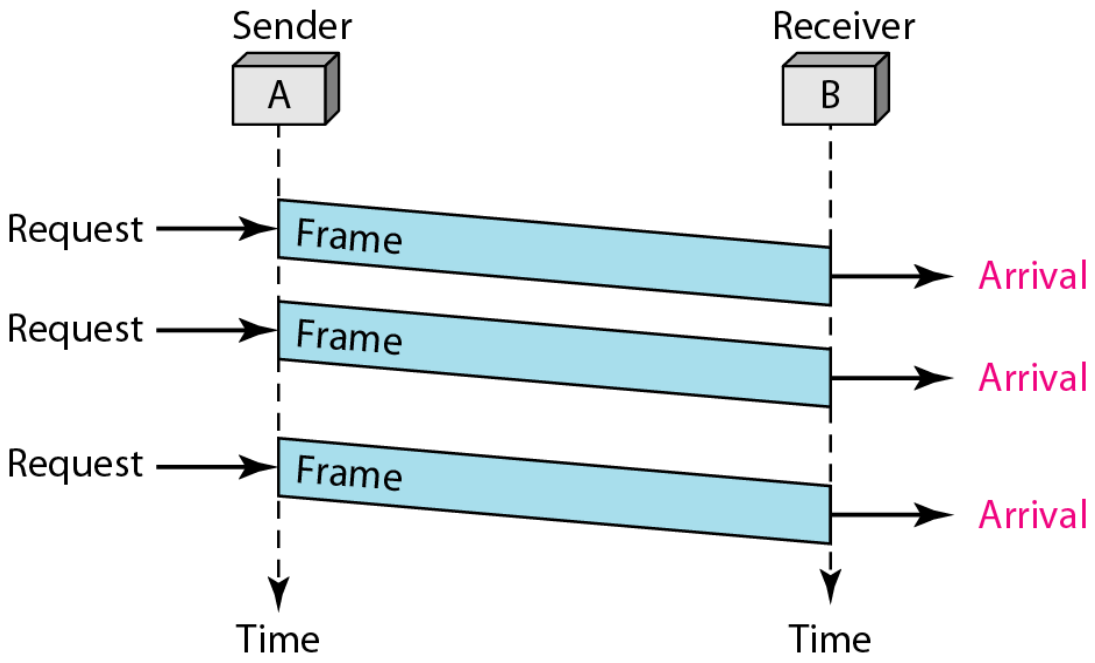
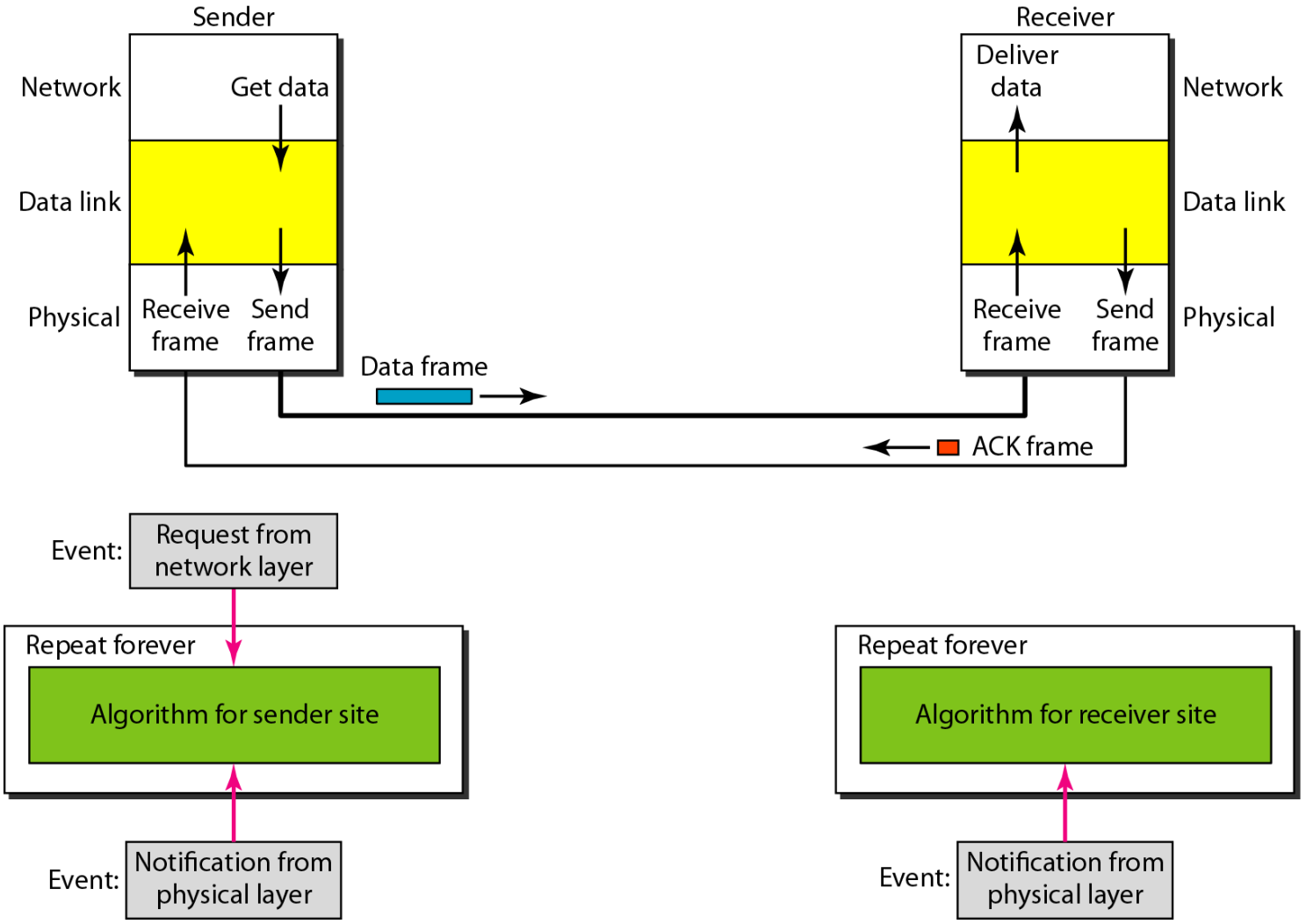


Figure 11.8 *Design of Stop-and-Wait Protocol*



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

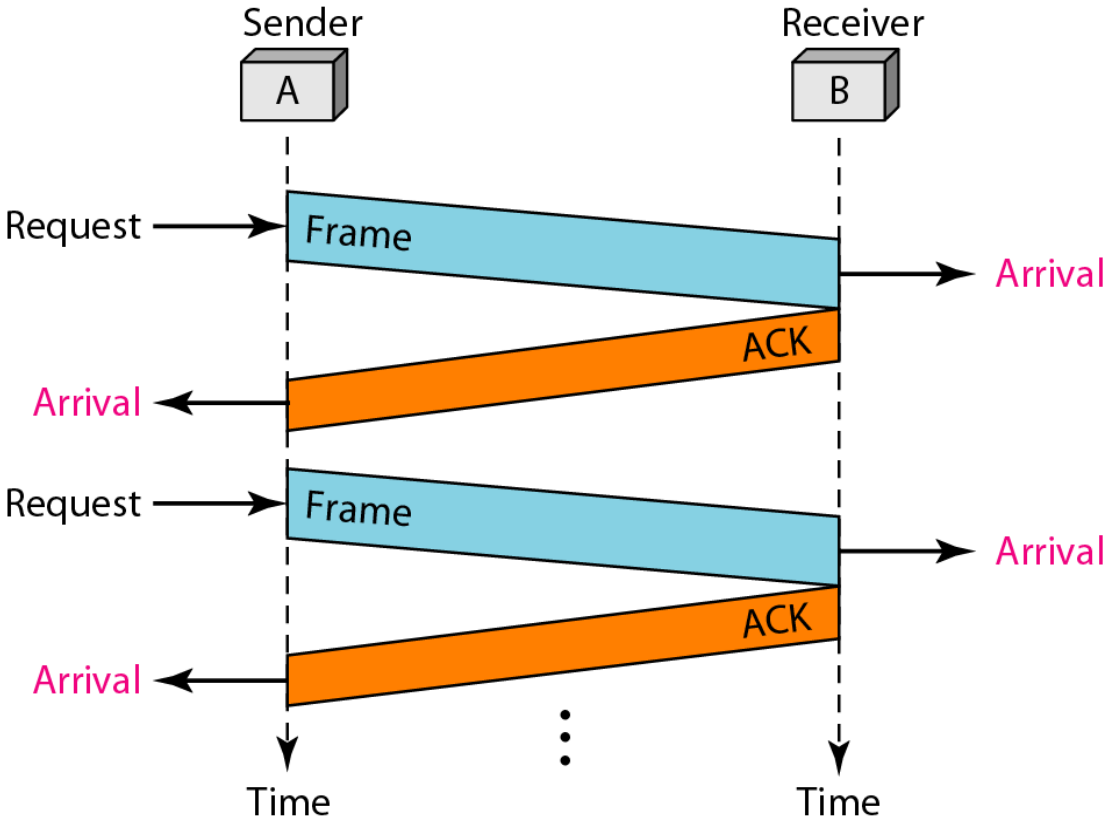
```
1  while(true)                          //Repeat forever
2  canSend = true                       //Allow the first frame to go
3  {
4    WaitForEvent();                    // Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend)
6    {
7       GetData();
8       MakeFrame();
9   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

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

Example 11.2

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

Figure 11.9 *Flow diagram for Example 11.2*



What's the maximum *throughput* (on average how many bits of information per time unit can be sent from the sender to the receiver)?

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

Note

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

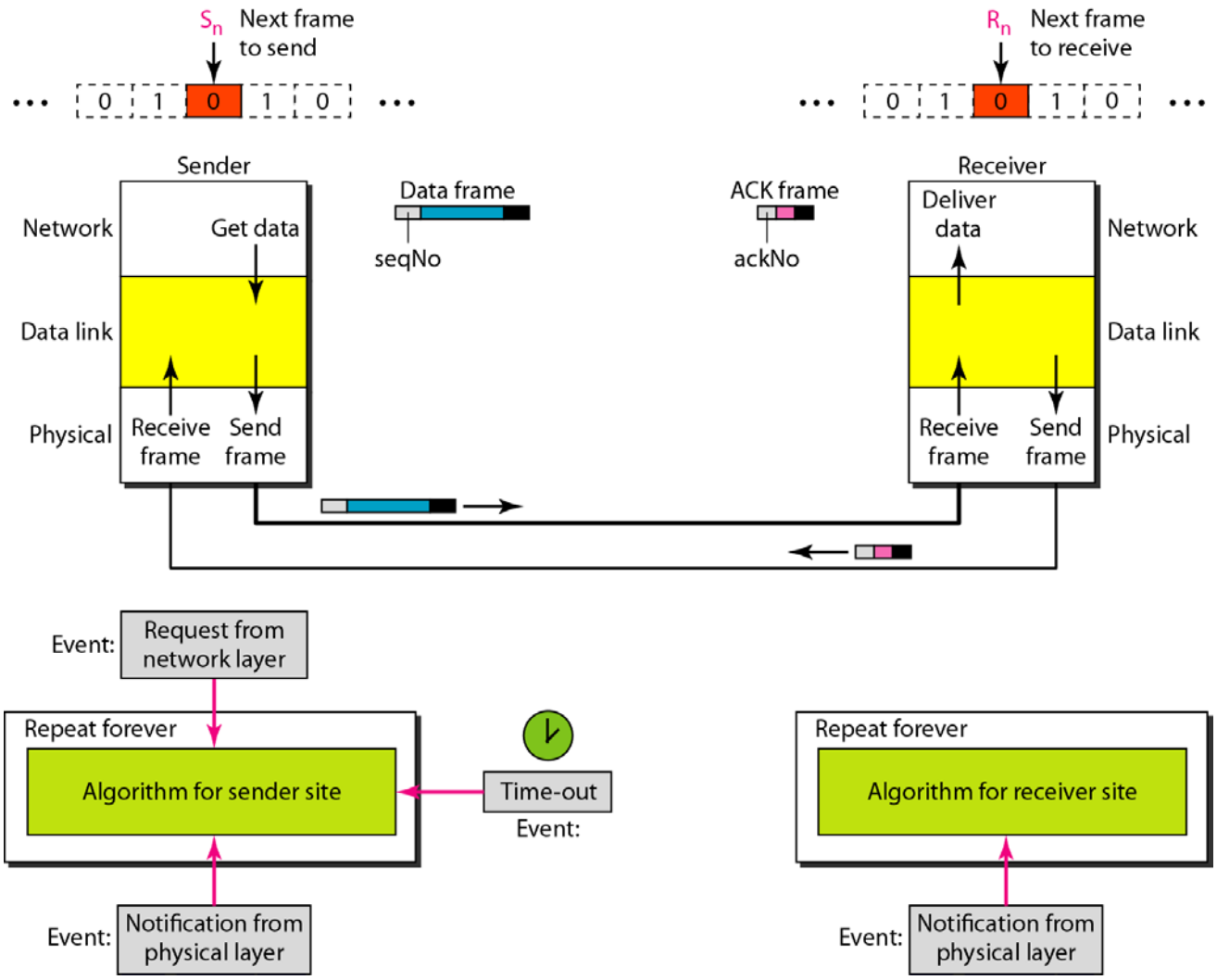
Note

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.

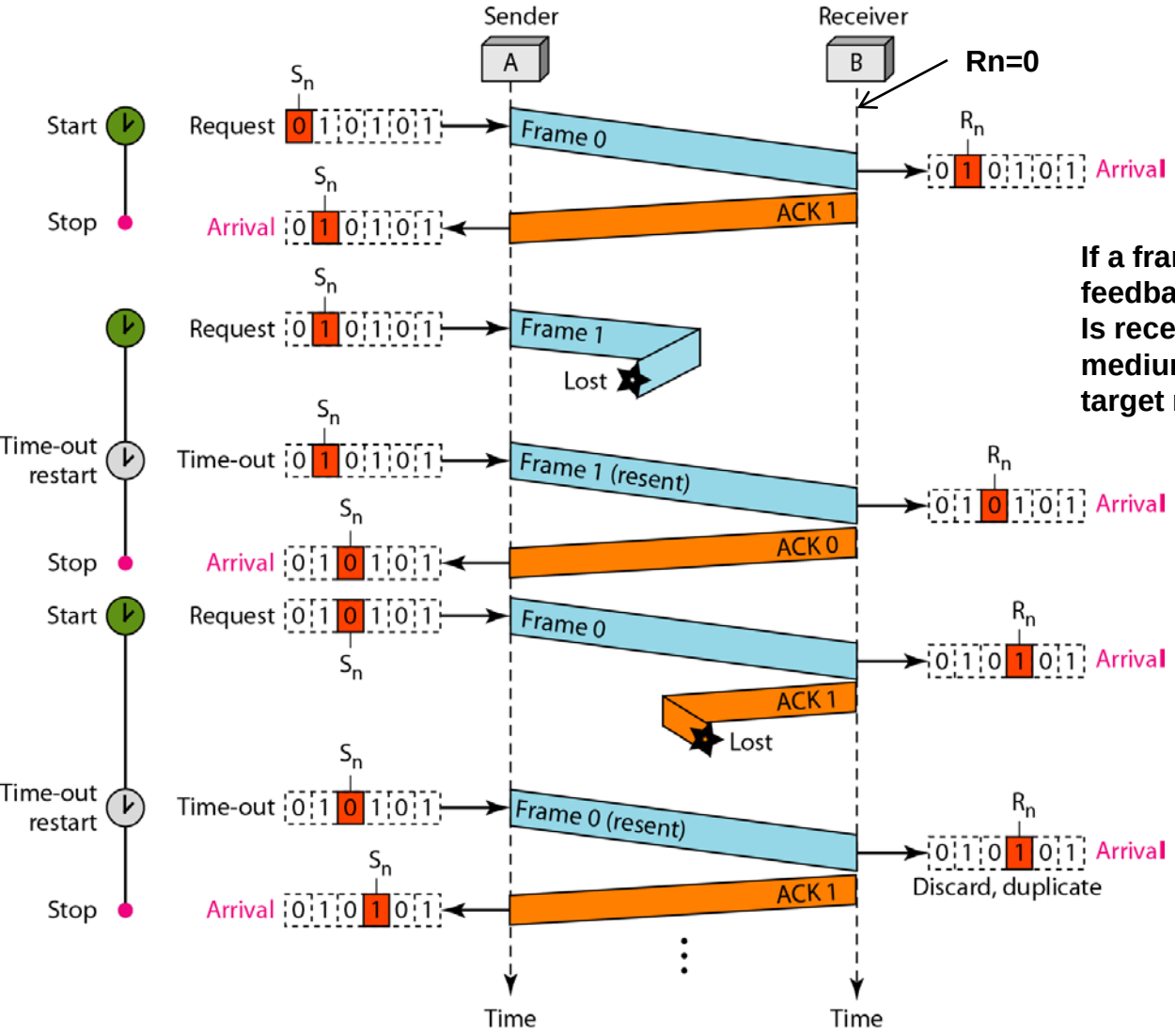
Figure 11.10 *Design of the Stop-and-Wait ARQ Protocol*



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*



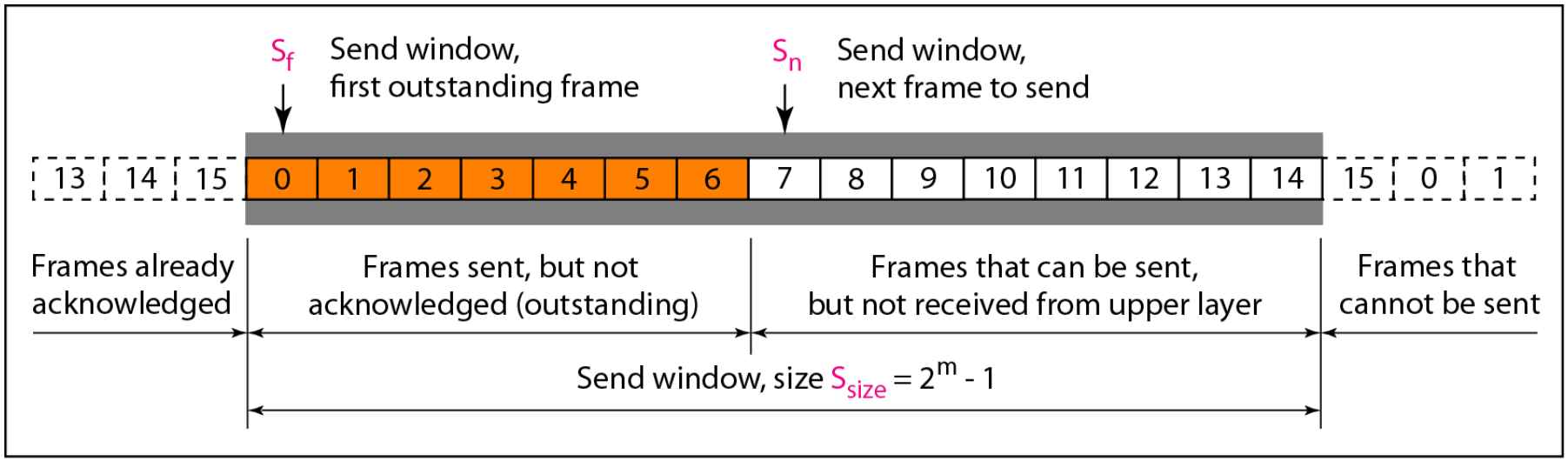
Recall that physical addresses of sender and receiver are included in header

If a frame is corrupted (lost), the receiver cannot feedback to sender. This is because the frame Is received by all stations over the same medium. Since the frame is corrupted, the target receiver does not know it is the receiver.

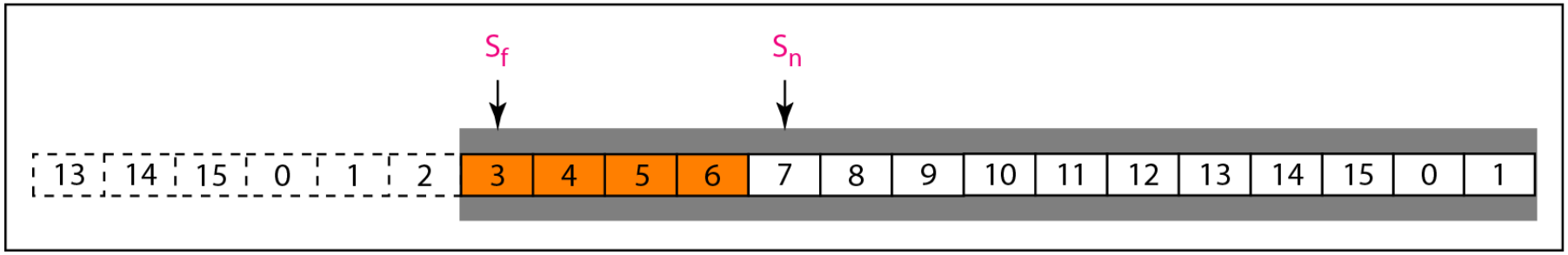
Note

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.

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



a. Send window before sliding



b. Send window after sliding

m=4

Note

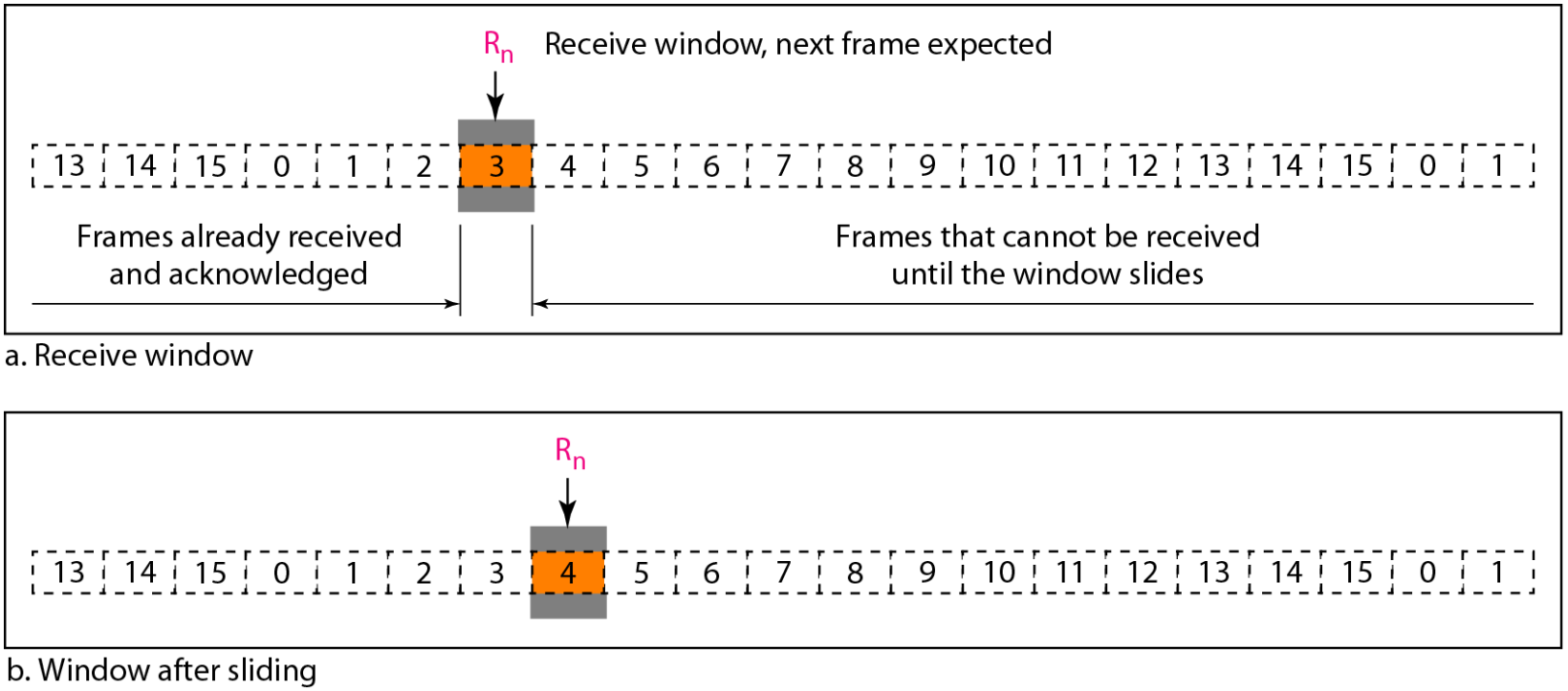
The send window is an abstract concept defining an imaginary box of size $2^m - 1$ with three variables: S_f, S_n, and S_{size}.

The send window size is the maximal number of outstanding frames allowed.

Note

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

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



Only frame(s) inside the receive window can be accepted by the receiver.

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*

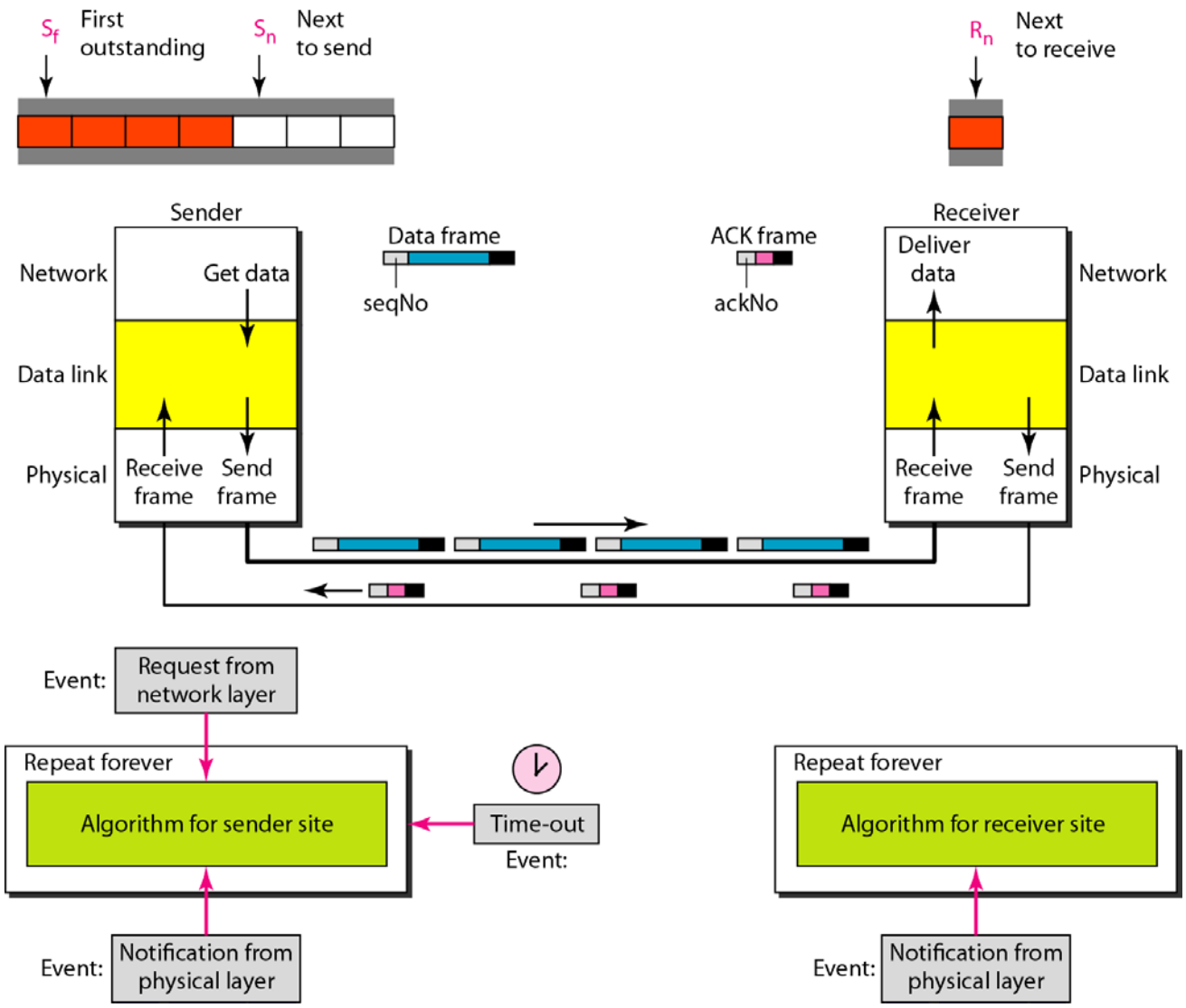
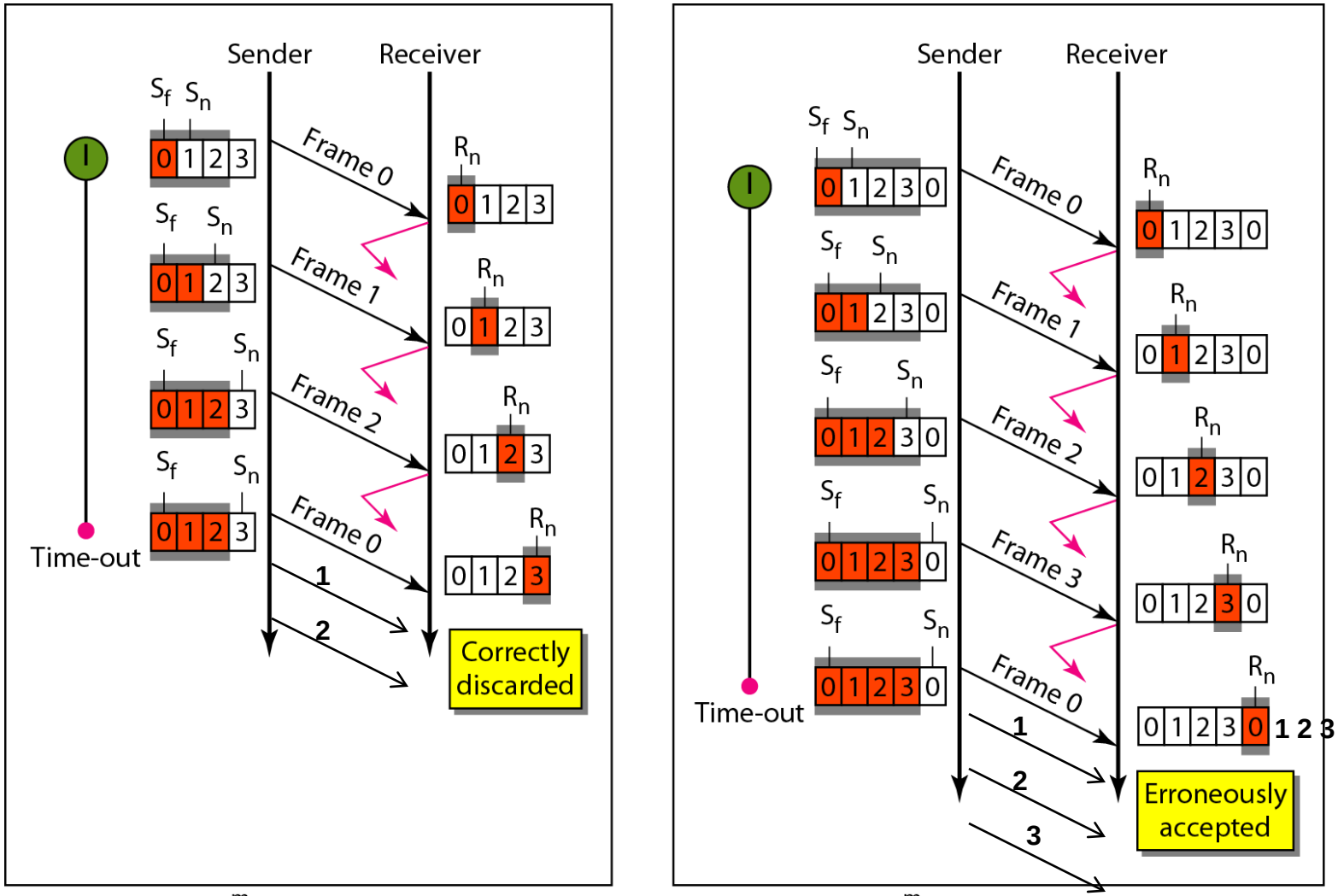


Figure 11.15 *Window size for Go-Back-N ARQ*



m=2

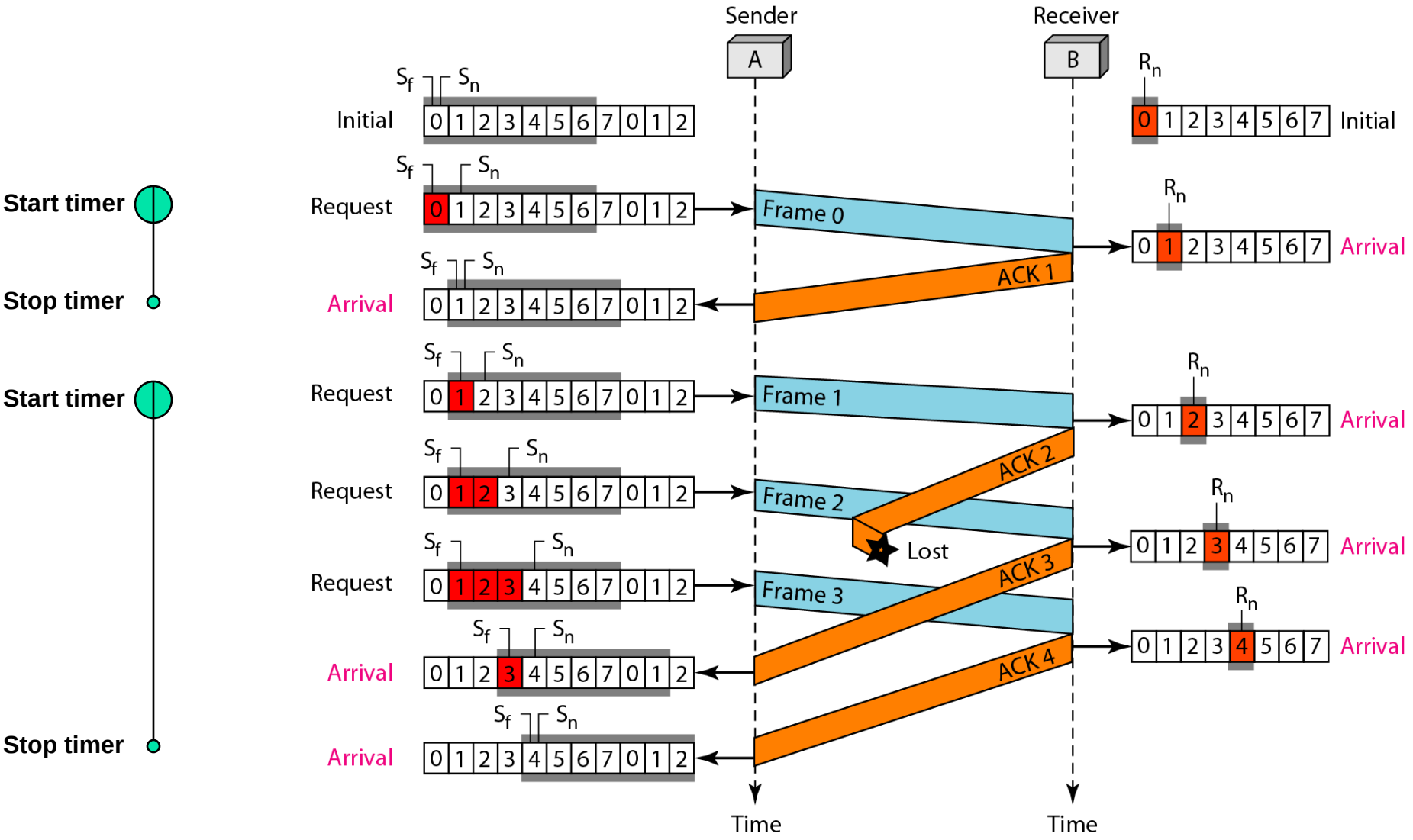
Note

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.

Example 11.6

Figure 11.16 shows an example of Go-Back-N. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost. After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

Figure 11.16 *Flow diagram for Example 11.6*



The timer stops upon ACK1 arrival, and restarts when Frame 1 is sent.

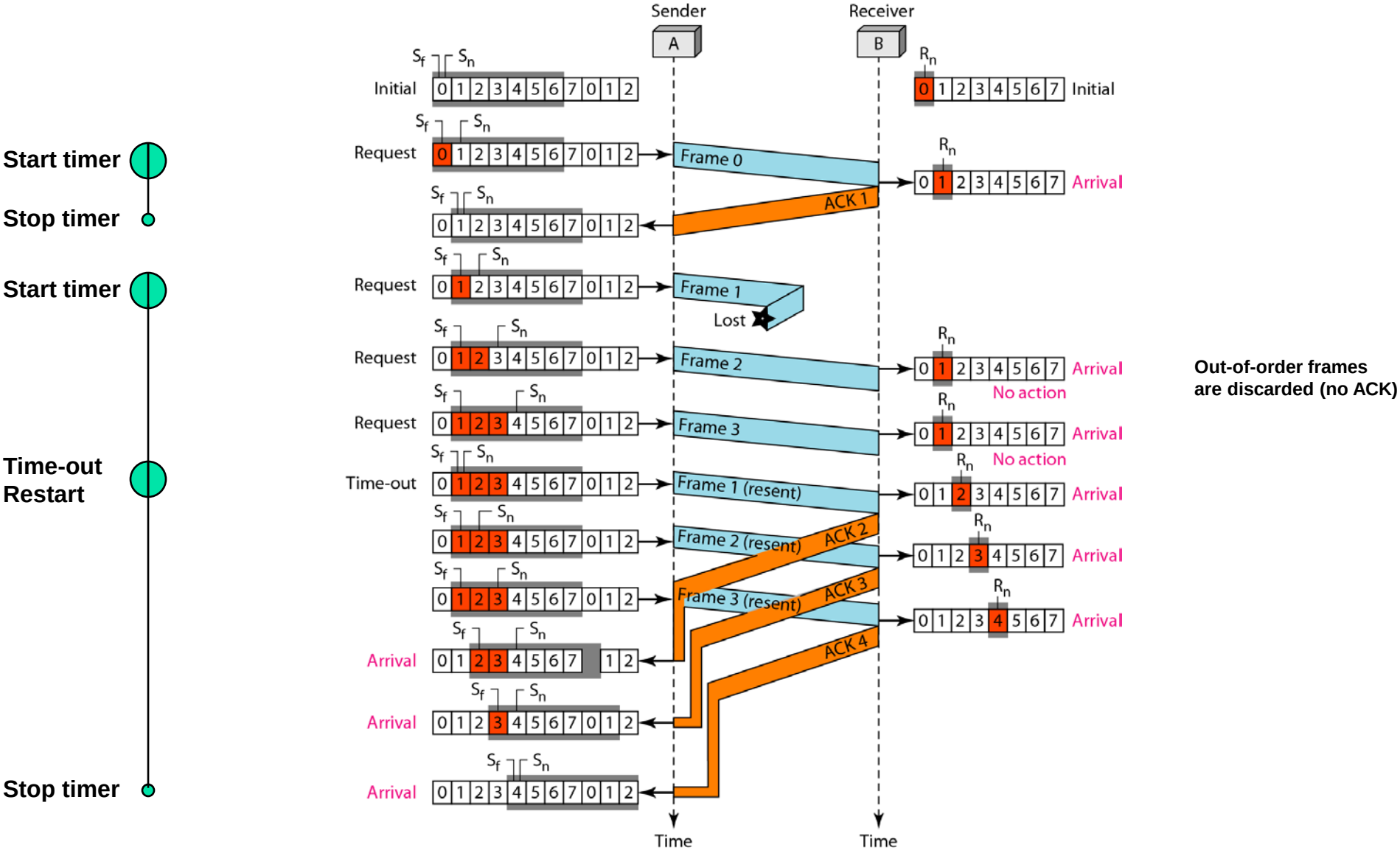
Example 11.7

Figure 11.17 shows what happens when a frame is lost. Frames 0, 1, 2, and 3 are sent. However, frame 1 is lost. The receiver receives frames 2 and 3, but they are discarded because they are received out of order. The sender receives no acknowledgment about frames 1, 2, or 3. Its timer finally expires. The sender sends all outstanding frames (1, 2, and 3) because it does not know what is wrong. Note that the resending of frames 1, 2, and 3 is the response to one single event. When the sender is responding to this event, it cannot accept the triggering of other events. This means that when ACK 2 arrives, the sender is still busy with sending frame 3.

Example 11.7 (continued)

The physical layer must wait until this event is completed and the data link layer goes back to its sleeping state. We have shown a vertical line to indicate the delay. It is the same story with ACK 3; but when ACK 3 arrives, the sender is busy responding to ACK 2. It happens again when ACK 4 arrives. Note that before the second timer expires, all outstanding frames have been sent and the timer is stopped.

Figure 11.17 *Flow diagram for Example 11.7*



An ACK is issued only when the expected frame arrives

Upon a timeout, retransmit all outstanding frames

Note

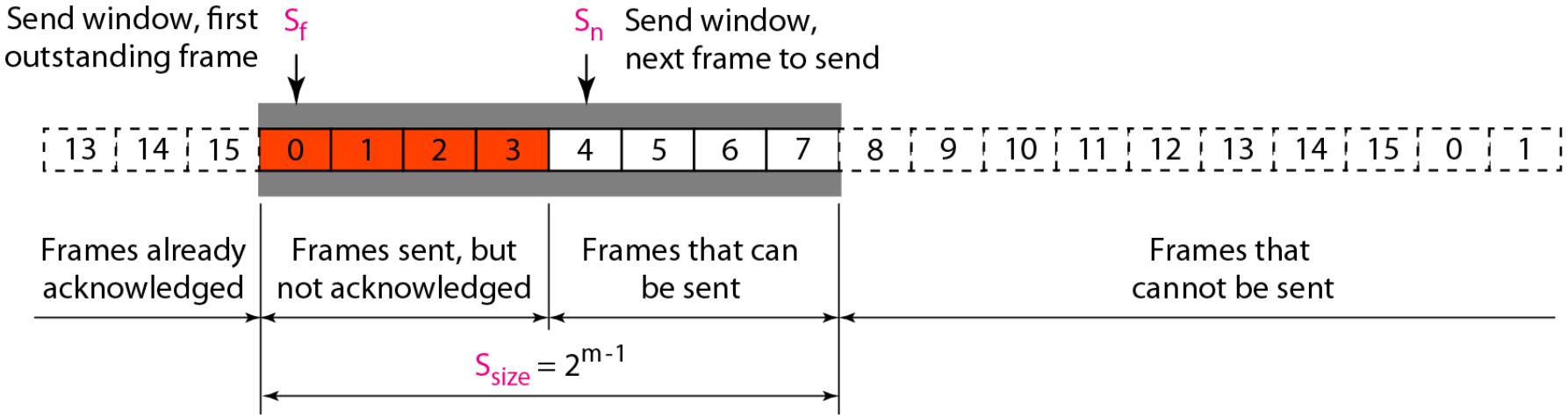
Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

Figure 11.18 *Send window for Selective Repeat ARQ*

Selective Repeat attempts to retransmit only those frames that are actually lost (due to errors)

– Receiver must be able to accept frames out of order → receive window size is more than one.

– Since receiver must release frames to higher layer in order, the receiver must be able to buffer some frames



m=4

Figure 11.19 *Receive window for Selective Repeat ARQ*

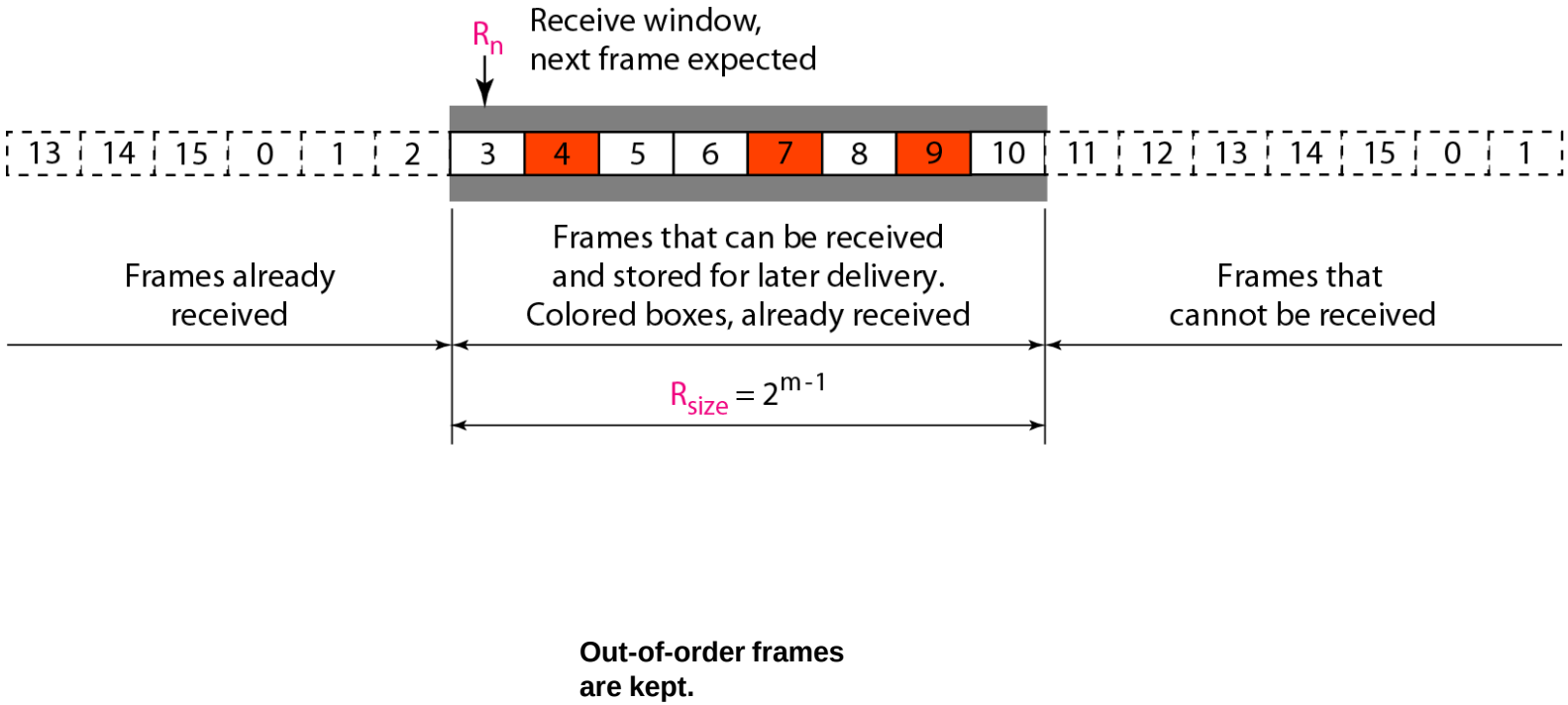


Figure 11.20 *Design of Selective Repeat ARQ*

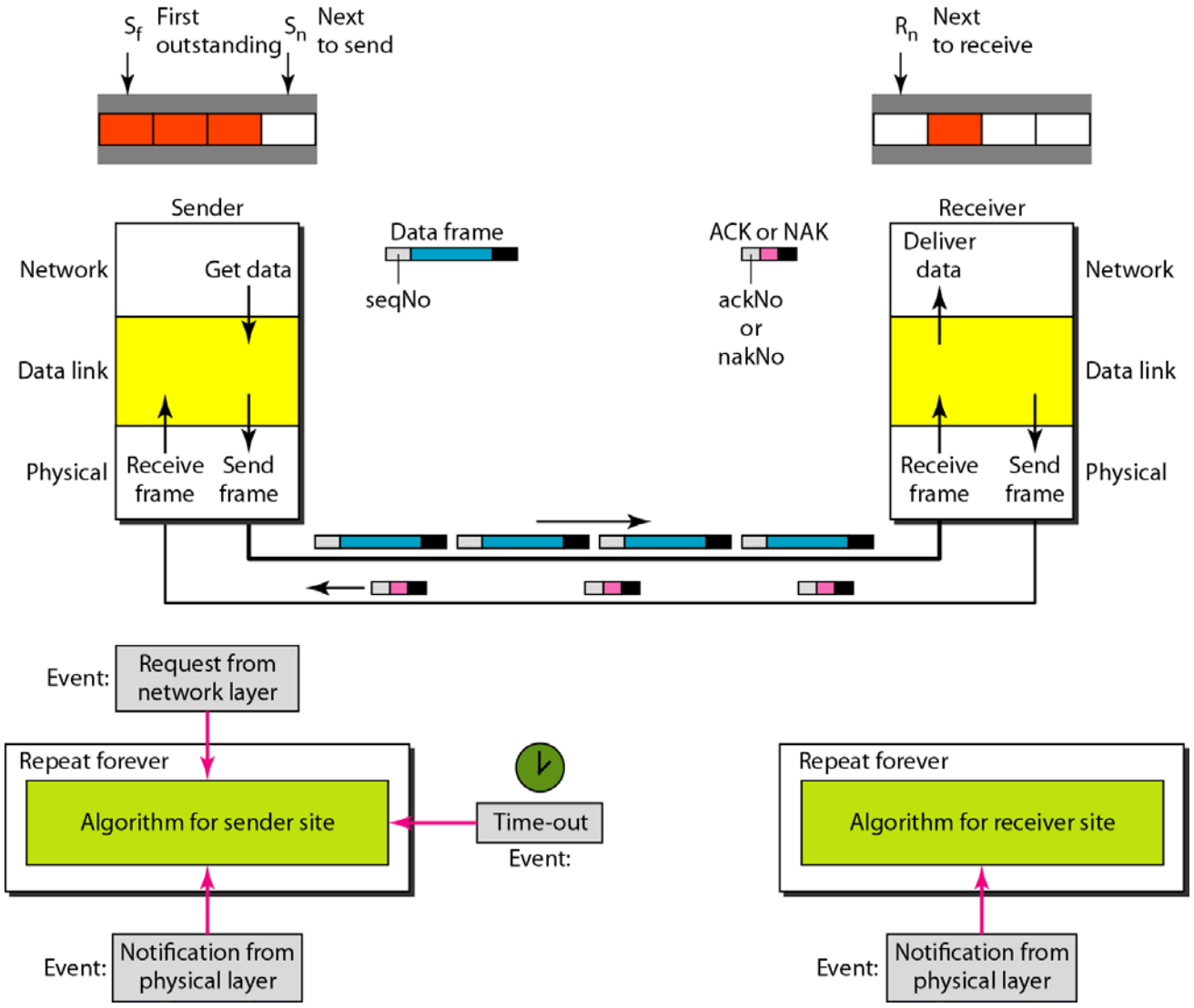
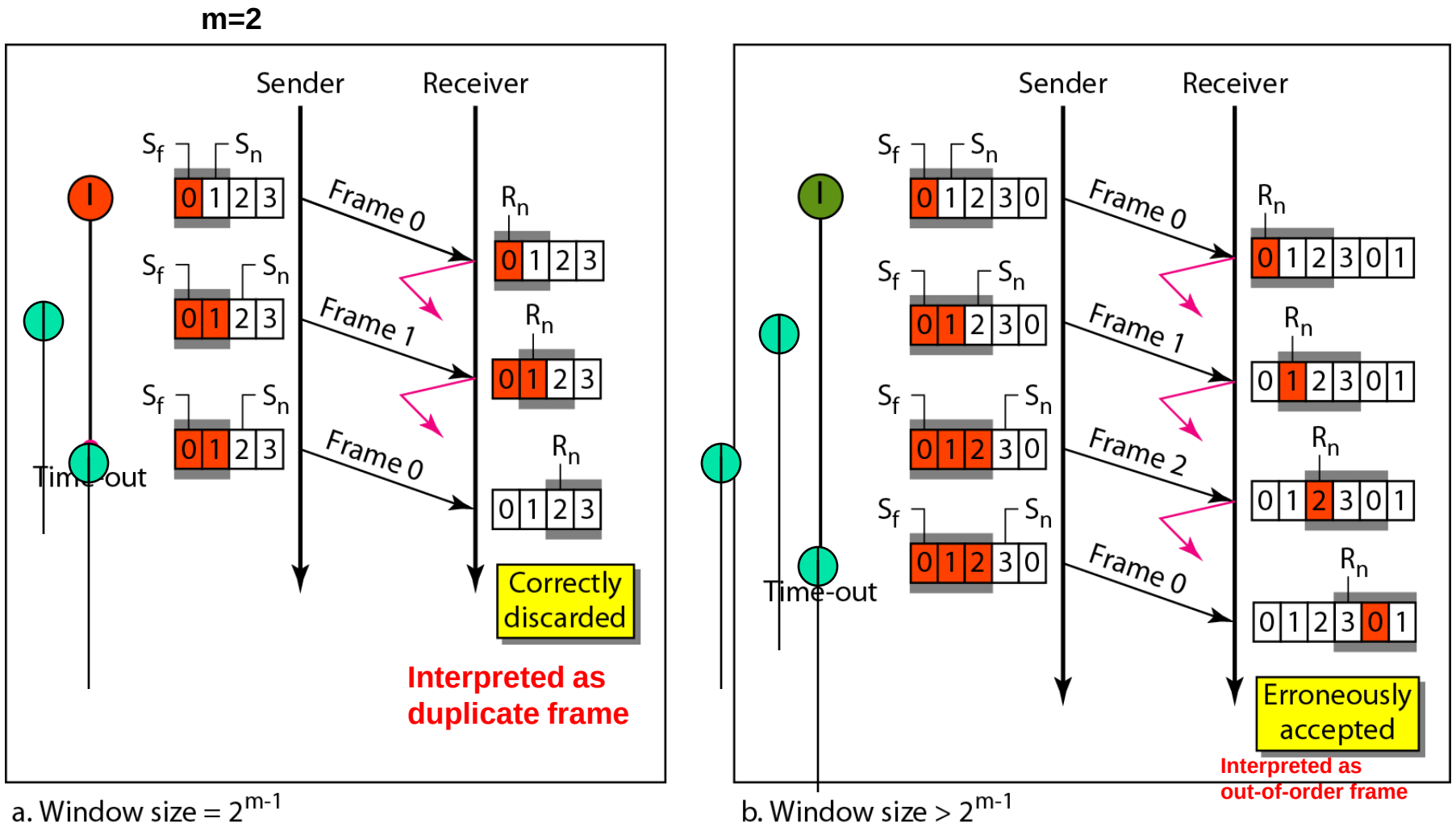


Figure 11.21 *Selective Repeat ARQ, window size*



When a frame not R_n is received, if the sequence number is within the receiver window, then it is considered an out-of-order frame; otherwise, it is considered a duplicate frame.

Note

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

Example 11.8

This example is similar to Example 11.3 and Example 11.7 in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation. One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.

Example 11.8 (continued)

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

Example 11.8 (continued)

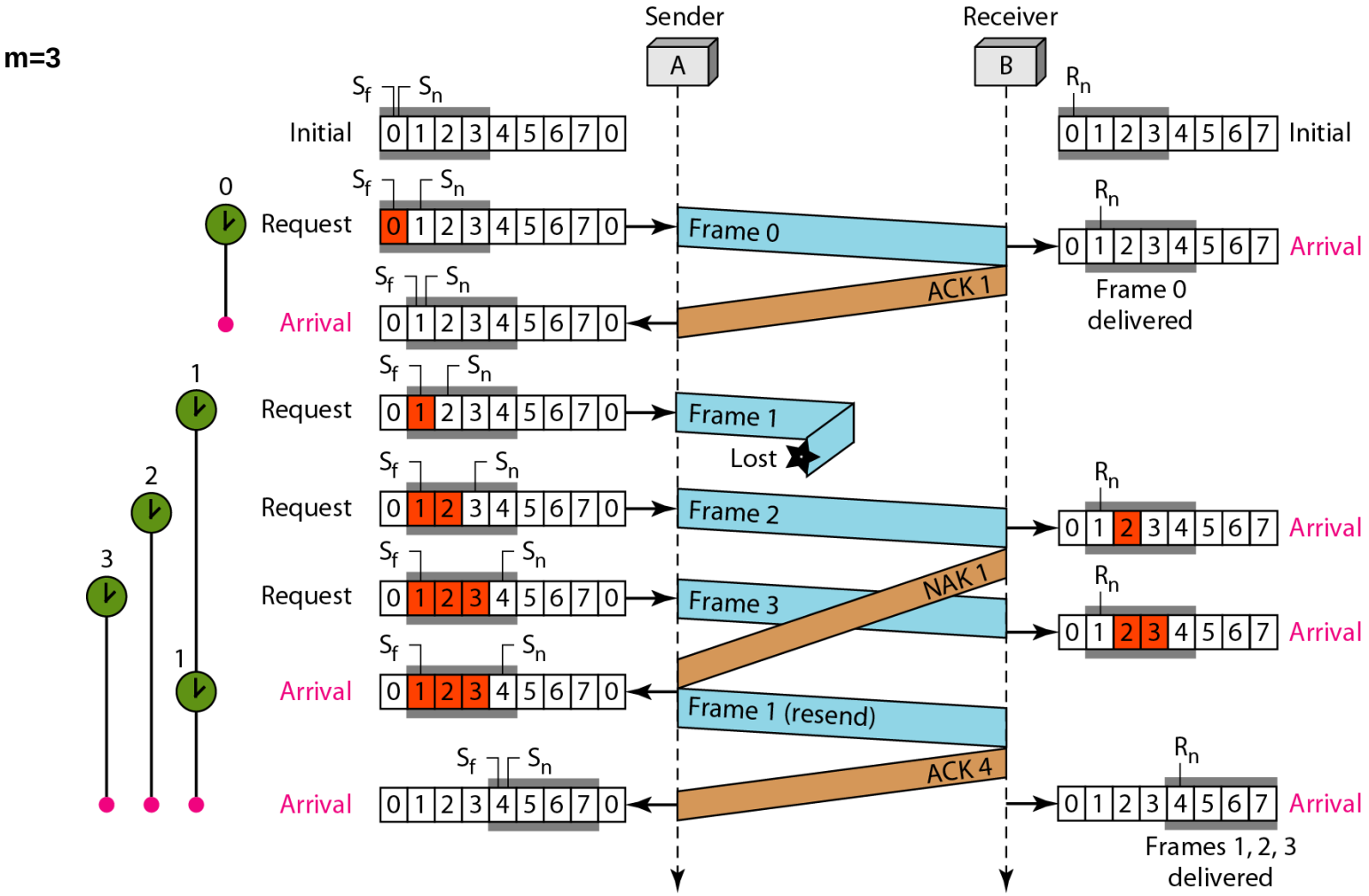
Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform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

NAK is to explicitly tell the sender that a particular frame is lost.

Example 11.8 (continued)

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

Figure 11.23 *Flow diagram for Example 11.8*



Compare Go-Back-N and selective repeat ARQ

- When all transmissions are successful, Go-Back-N is more efficient, as its send window is almost twice that of selective repeat ARQ
- When we have a large probability of errors, selective repeat ARQ is more efficient, since it has fewer retransmissions.

Retransmission limit:

- There is a limit for the number of retransmissions. For example, assume the limit is 7. If the sender has retransmitted a frame for 7 times but did not get ACK back, the sender will discard this frame and rely on the transport layer for recovery, as follows.
- At the transport layer, there will be a missing segment in the received segments. Once the transport layer notices there is a missing segment, the sender will retransmit the missing segment.

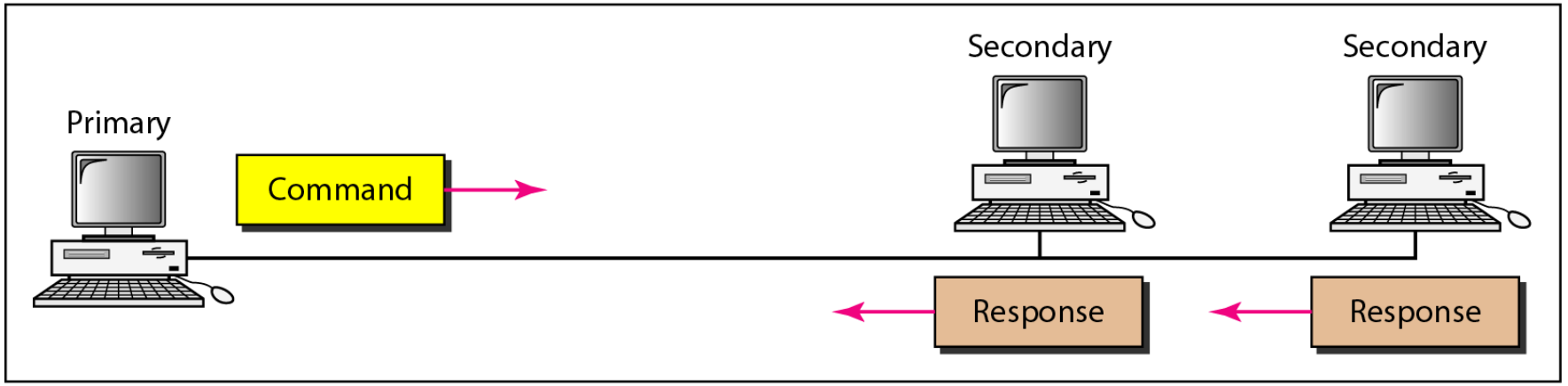
11-6 HDLC

High-level Data Link Control (HDLC) ***is a protocol for communication over point-to-point and multipoint links (but is now used almost exclusively to connect one device to another). It implements the ARQ mechanisms we discussed in this chapter.***

Bit oriented

Figure 11.25 *Normal response mode*

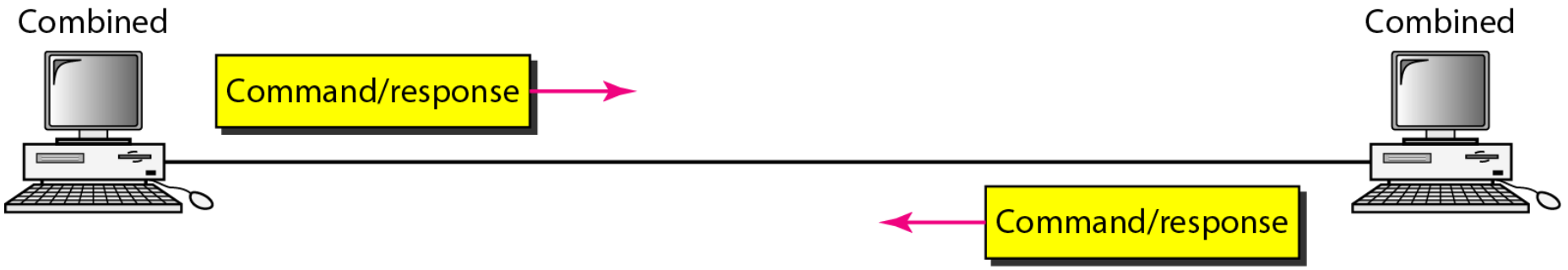
a. Point-to-point



b. Multipoint

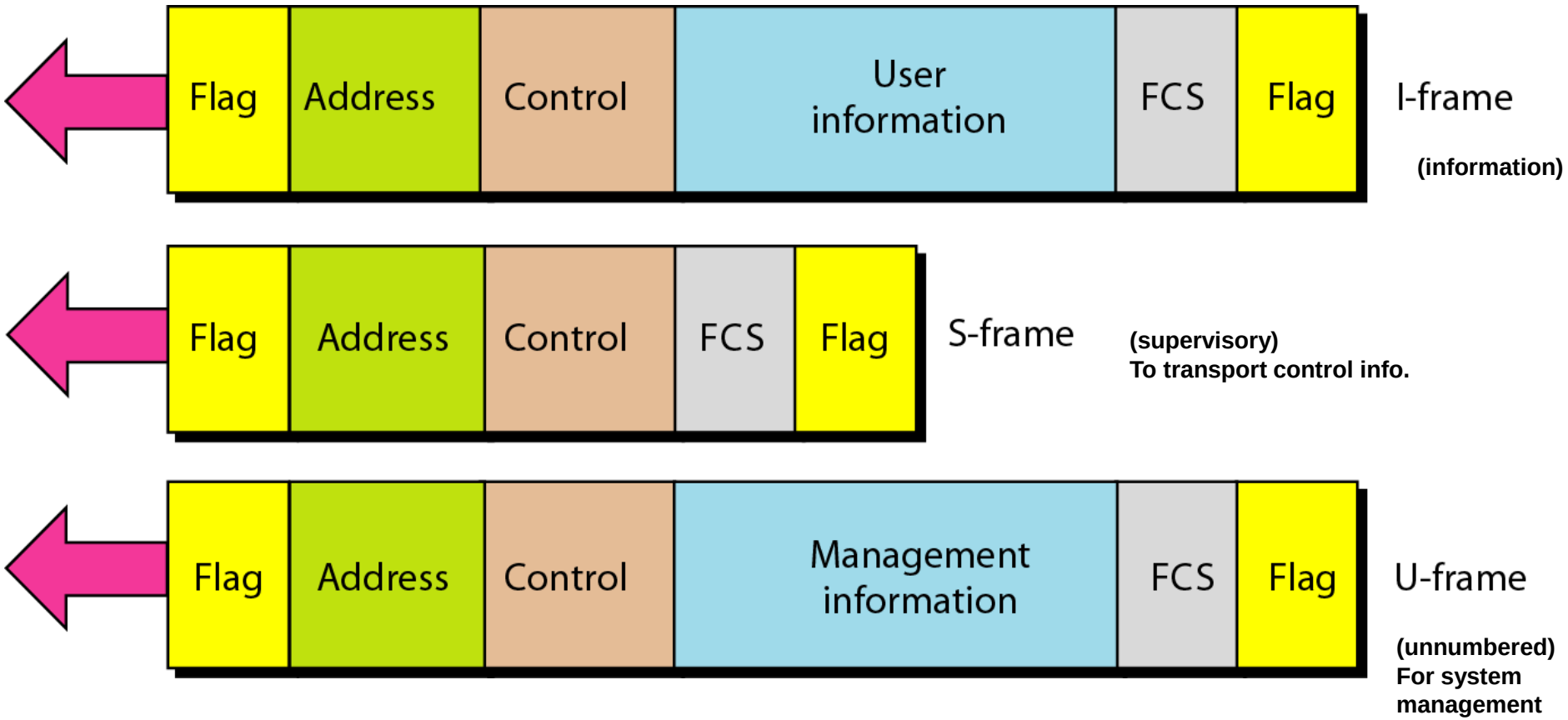
Unbalanced. Secondary stations can only respond.

Figure 11.26 *Asynchronous balanced mode*



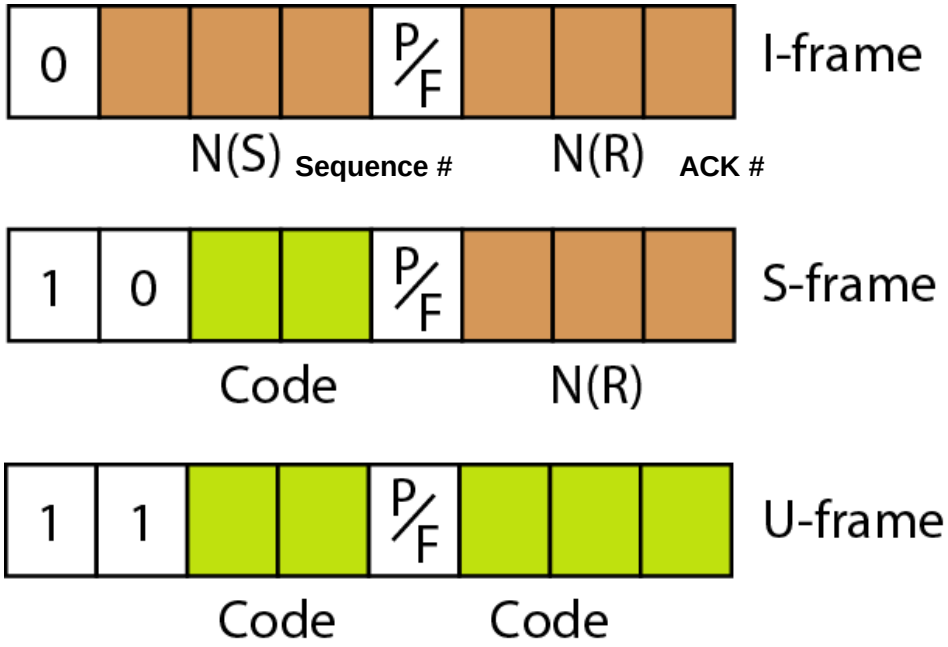
Each station can function as primary and secondary (acting as peers). Common mode today.

Figure 11.27 *HDLC frames*



FCS: frame check sequence, 16 or 32-bit CRC

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



P/F is used in normal response mode

N(S): sequence number of a frame (3 bits for regular, 7 bits for Extended)
N(R): acknowledge number (piggybacking is used)

Piggybacking: used to improve efficiency in bidirectional protocols. When station A sends a frame to station B, the frame also carries control information about arrived (or lost) frames from station B.

S-frame is used for flow and error control whenever piggybacking is either impossible or inappropriate. For example, if Code is 10, it means Receiver not ready (RNR). This is to acknowledge the receipt of a frame or groups of frames, and announce that the receiver is busy and cannot receive more frames.
If the Code is 00, it means Receive Ready (RR).

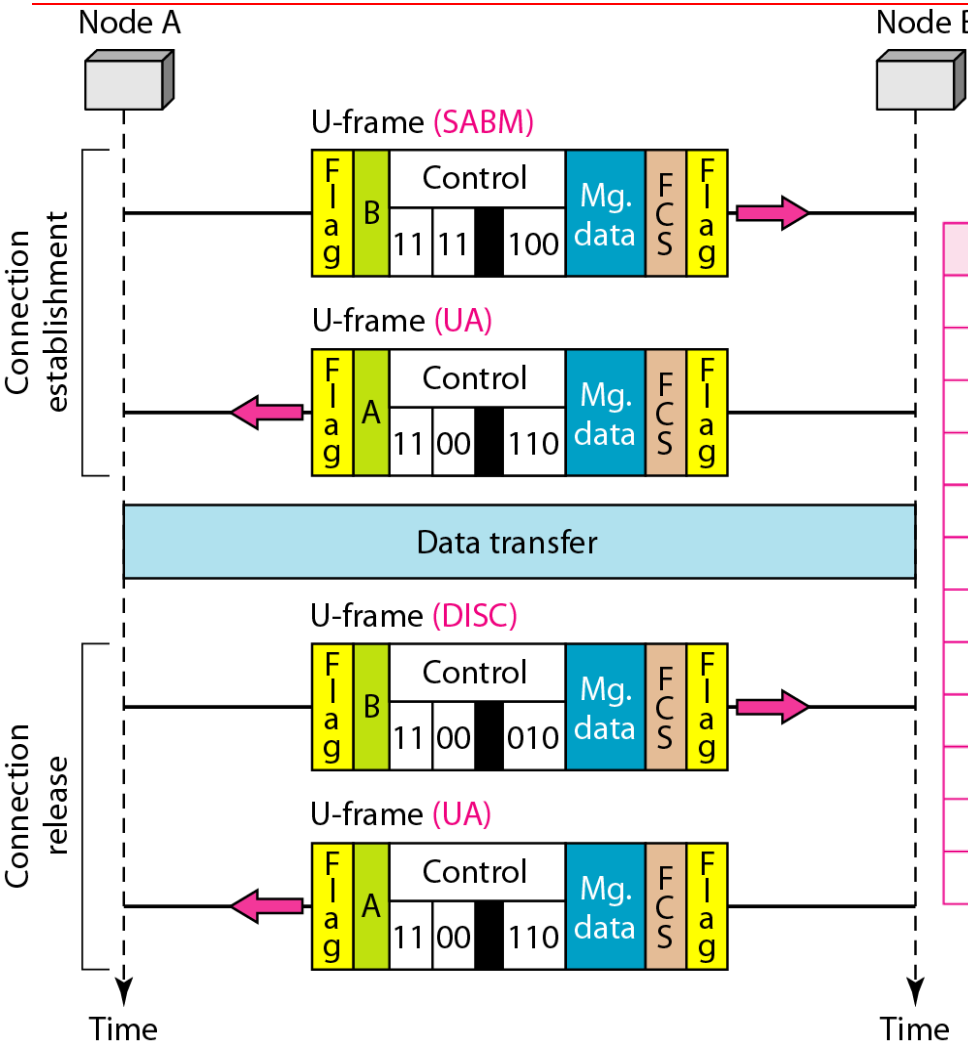
Table 11.1 *U-frame control command and response*

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

Example 11.9

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

Figure 11.29 *Example of connection and disconnection*



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

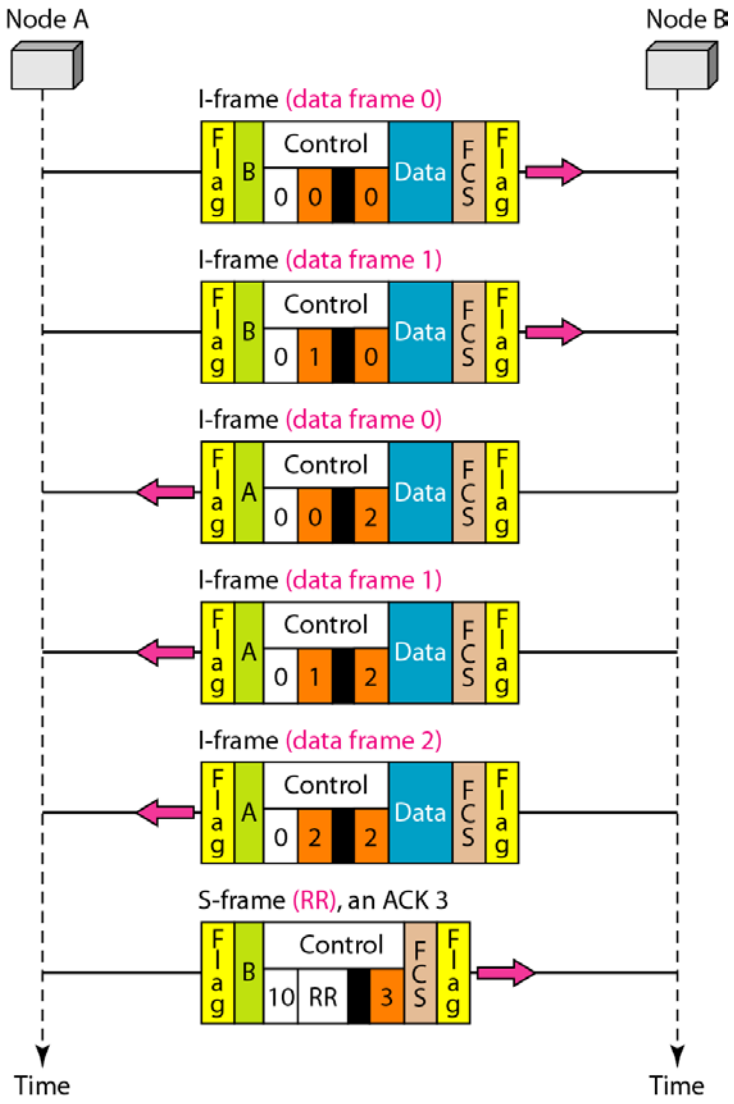
Example 11.10

Figure 11.30 shows an exchange using piggybacking. Node A begins the exchange of information with an I-frame numbered 0 followed by another I-frame numbered 1. Node B piggybacks its acknowledgment of both frames onto an I-frame of its own. Node B's first I-frame is also numbered 0 [N(S) field] and contains a 2 in its N(R) field, acknowledging the receipt of A's frames 1 and 0 and indicating that it expects frame 2 to arrive next. Node B transmits its second and third I-frames (numbered 1 and 2) before accepting further frames from node A.

Example 11.10 (continued)

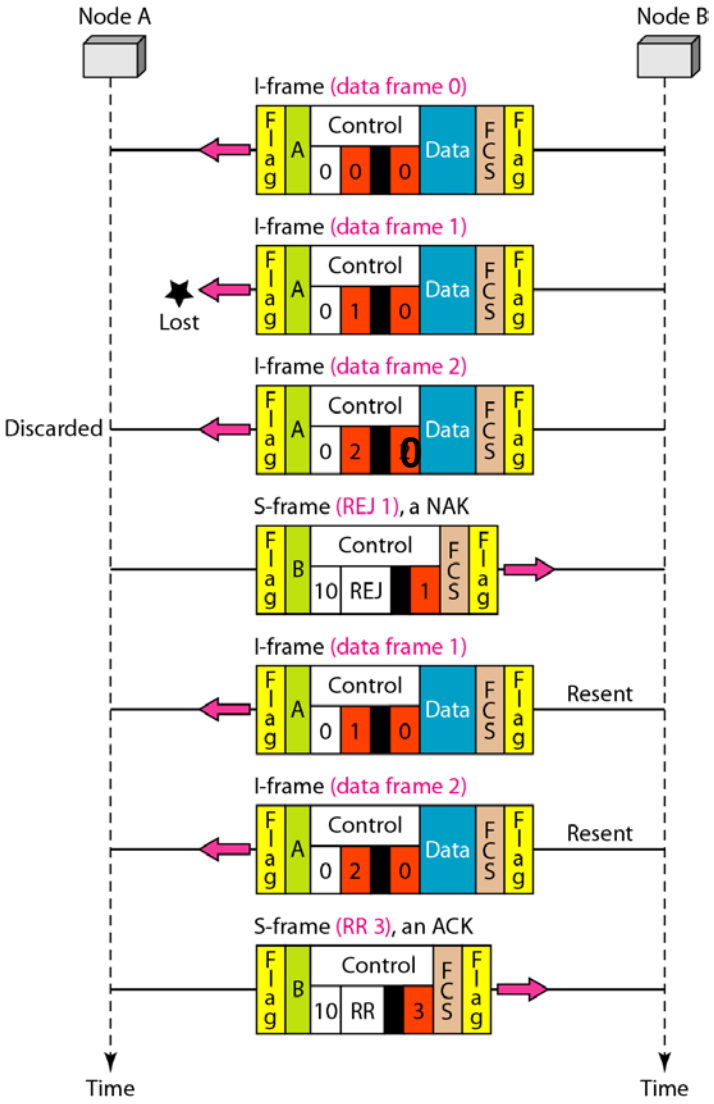
Its N(R) information, therefore, has not changed: B frames 1 and 2 indicate that node B is still expecting A's frame 2 to arrive next. Node A has sent all its data. Therefore, it cannot piggyback an acknowledgment onto an I-frame and sends an S-frame instead. The RR code indicates that A is still ready to receive. The number 3 in the N(R) field tells B that frames 0, 1, and 2 have all been accepted and that A is now expecting frame number 3.

Figure 11.30 *Example of piggybacking without error*



If errors, Go-back-N
Or selective repeat ARQ

Figure 11.31 *Example of piggybacking with error*

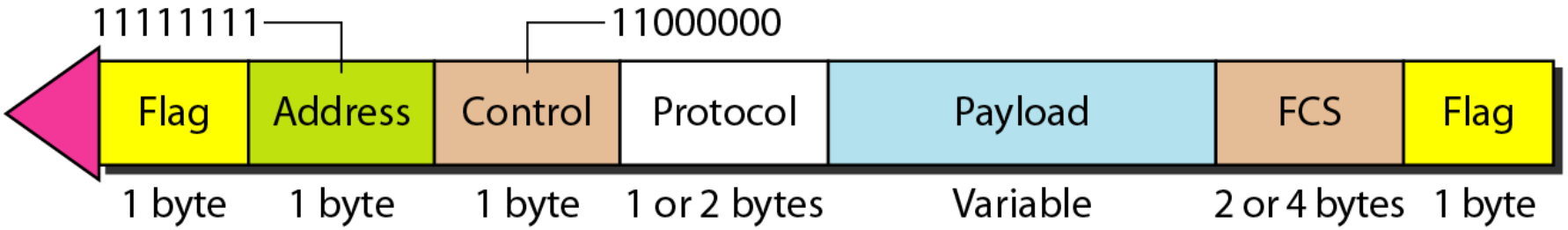


11-7 POINT-TO-POINT PROTOCOL

Although HDLC is a general protocol that can be used for both point-to-point and multipoint configurations, one of the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP).

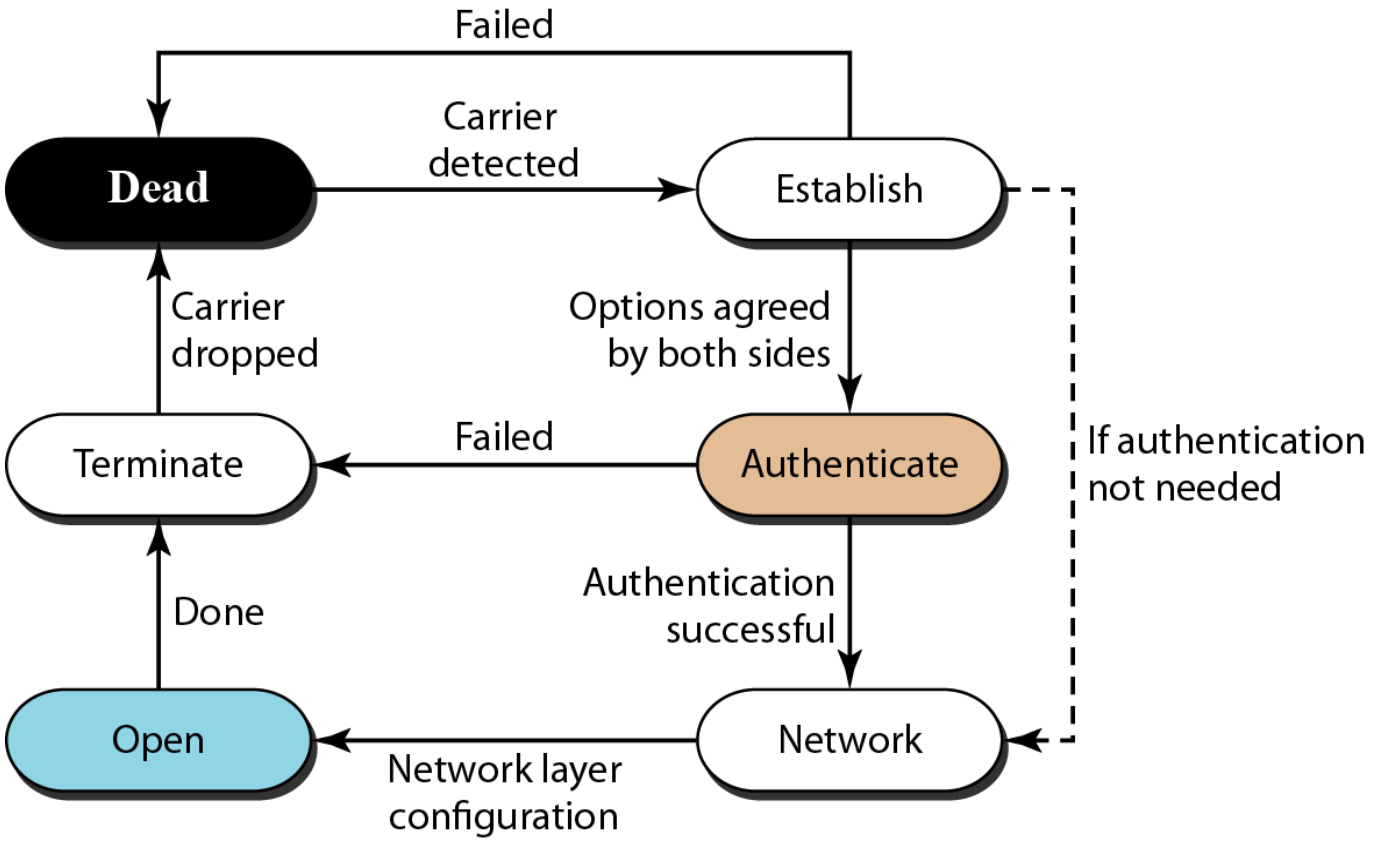
For example: to connect home computers to the server of an ISP through telephone lines and modems.

Figure 11.32 *PPP frame format*



Byte oriented
Both Address and Control fields have constant values
No flow control
Only error detection.
Protocol: defines what is carried in the data field; either user data or other info.

Figure 11.33 *Transition phases*



Although PPP is a data link protocol, PPP uses another set of other protocols to establish the link, authenticate the parties involved, and carry the network layer data.

Figure 11.35 *LCP packet encapsulated in a frame*

Link Control Protocol (LCP): establish, maintain, configure, and terminate links

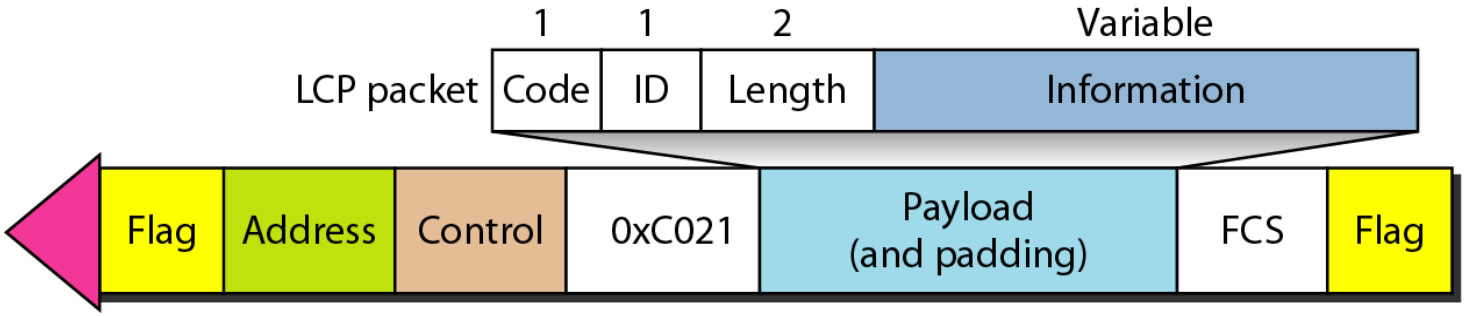


Figure 11.36 *PAP packets encapsulated in a PPP frame*

Password Authentication Protocol (PAP): the user who wants to access a system sends an authentication identification (usually the user name) and a password; the system checks the validity and either accepts or denies connection.

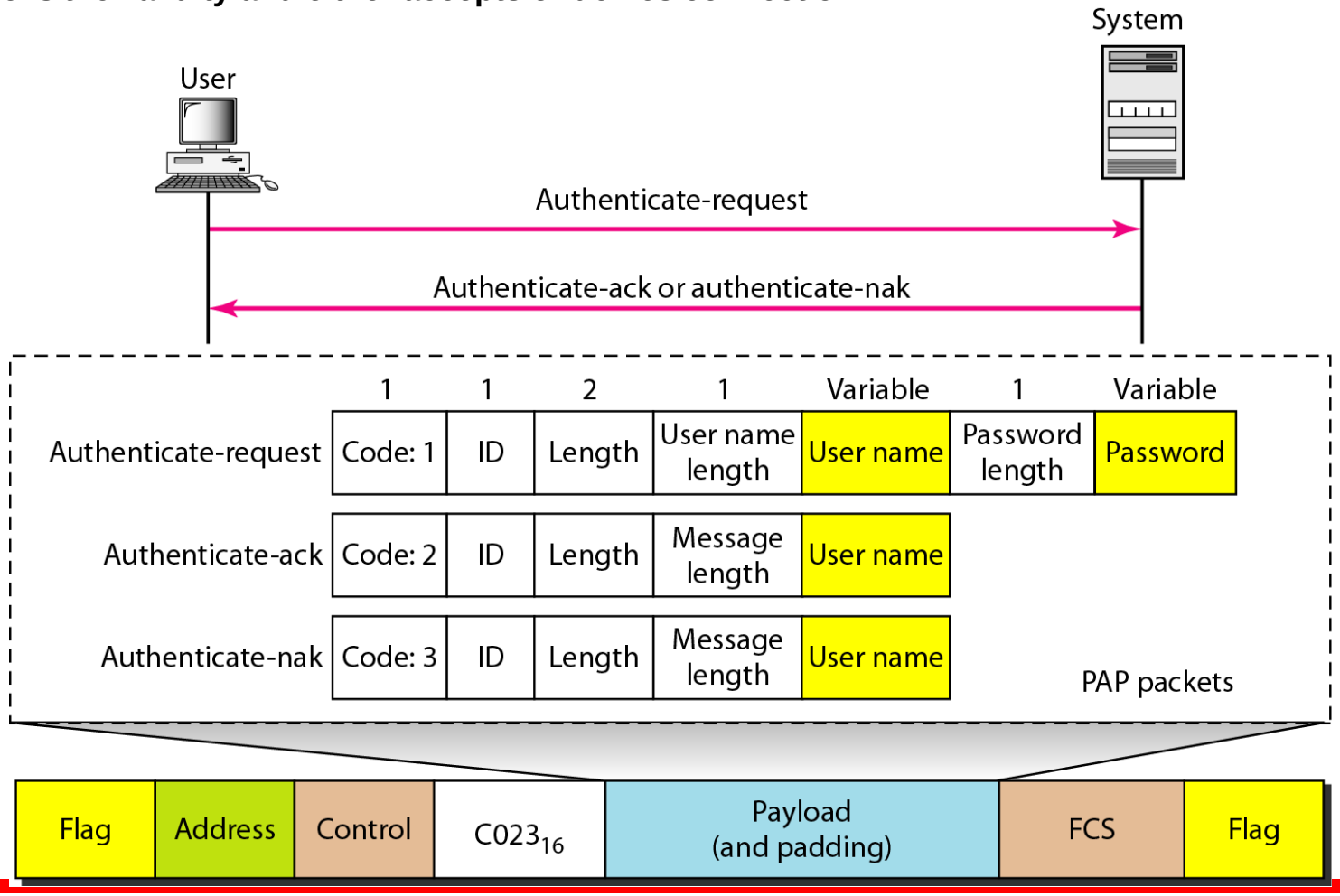
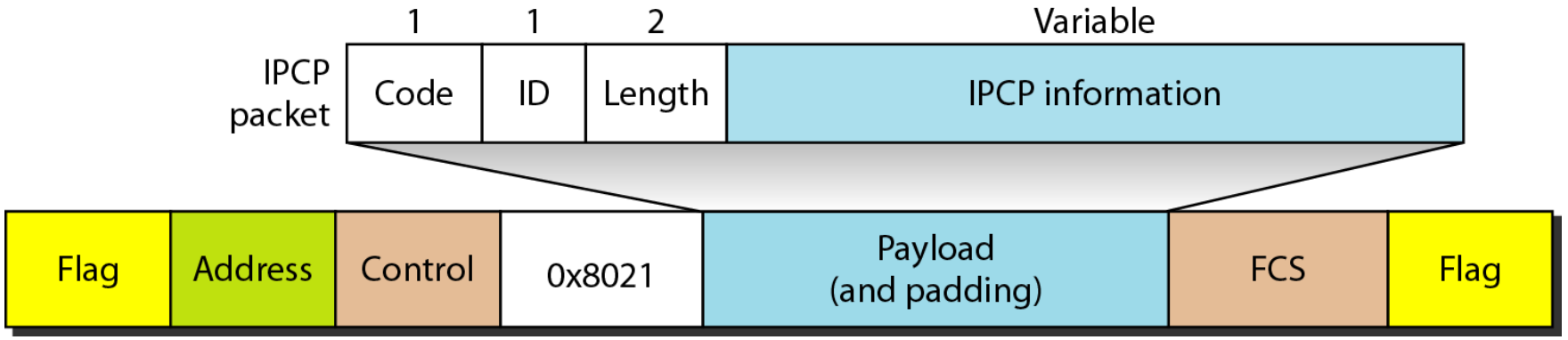


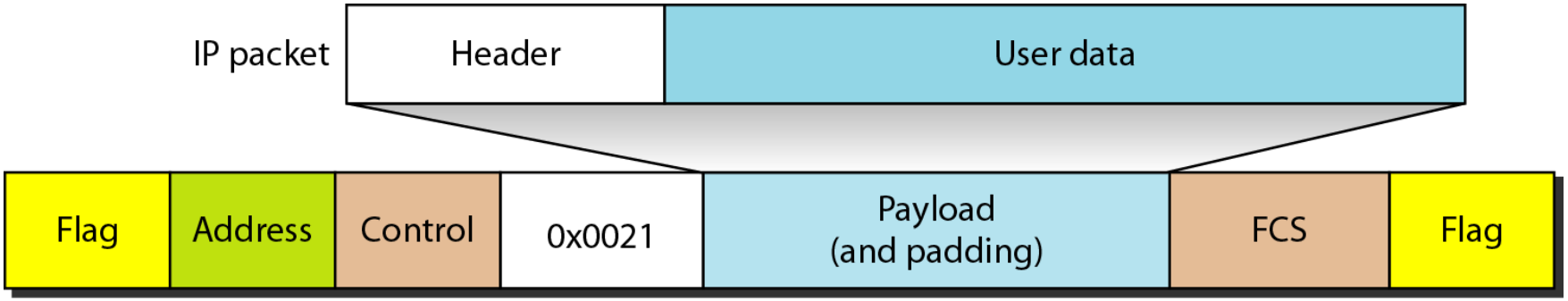
Figure 11.38 *IPCP packet encapsulated in PPP frame*

Internet Procotol Control Protocol (IPCP)



Configure the link used to carry IP packets in the Internet▪

Figure 11.39 *IP datagram encapsulated in a PPP frame*



Example 11.12

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

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

Example 11.12 (continued)

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

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

Figure 11.41 *An example*

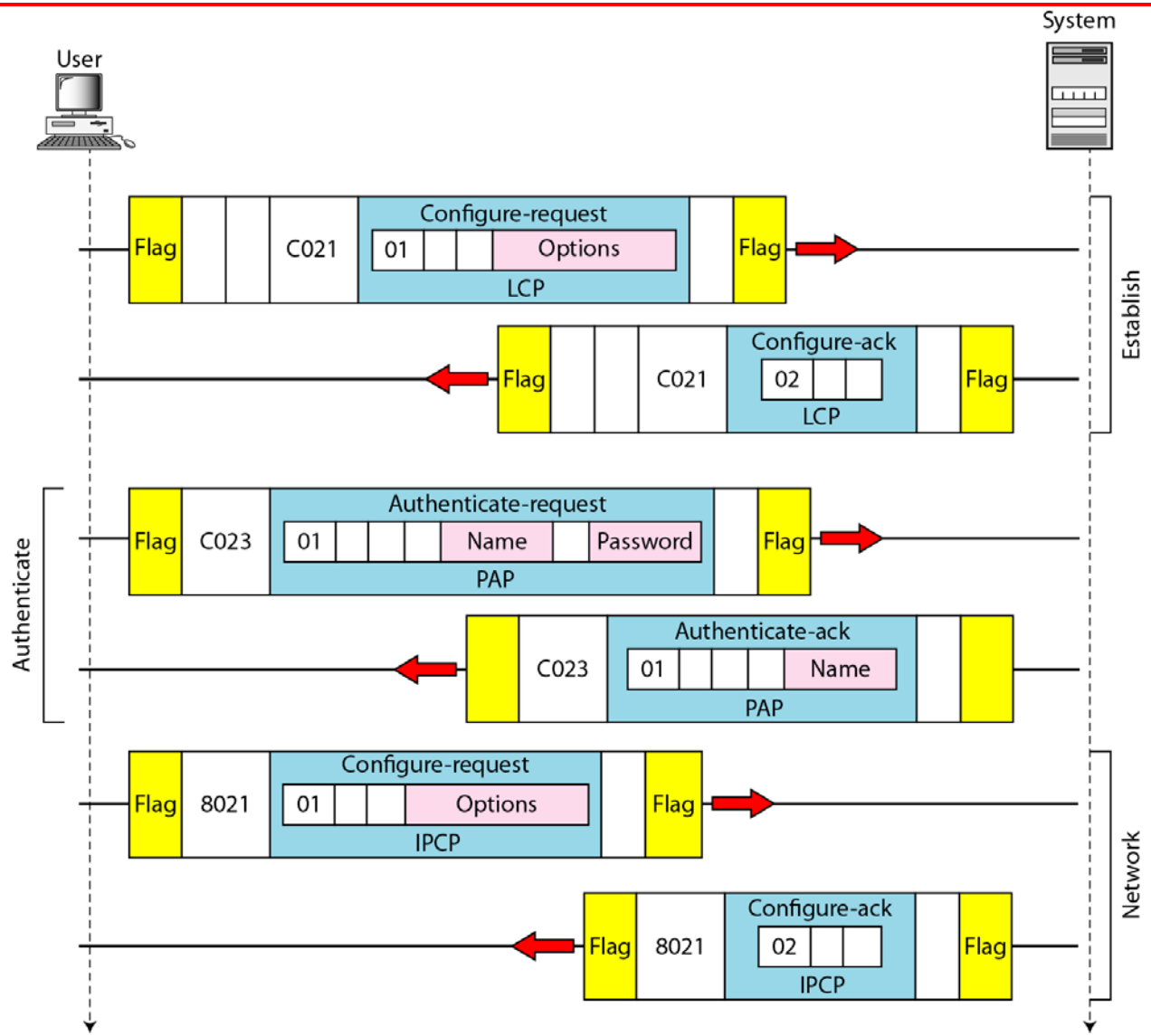


Figure 11.41 *An example (continued)*