Logical Link Control

Outline

Design Issues: Services to Network Layer,Framing,Error Control and Flow Control

Flow Control Protocols: Unrestricted Simplex, Stop and Wait, Sliding Window Protocol

Error Control: Parity Bits, Hamming Codes (11/12-bits) and CRC.

WAN Connectivity : PPP and HDLC

Data Link Layer Design Issues

- Network layer services
- Framing
- Error control
- Flow control

Data Link Layer Design Issues

Physical layer delivers bits of information to and from data link layer.The functions of Data Link Layer are:

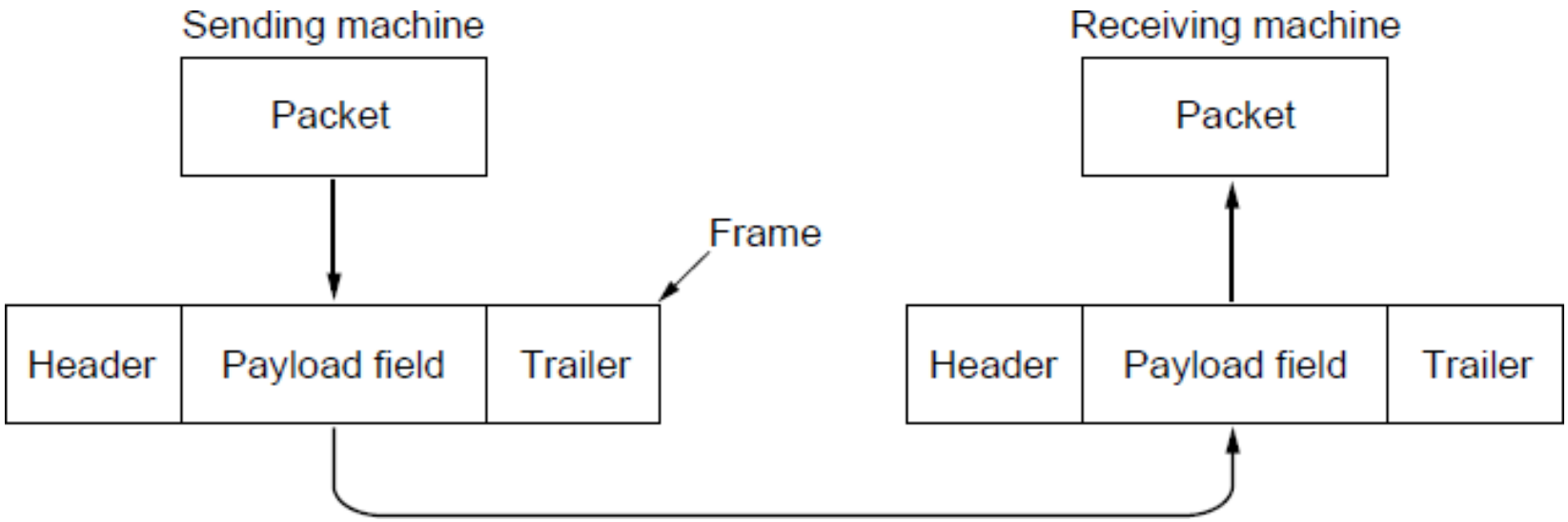
- Providing a well-defined service interface to the network layer.
- Dealing with transmission errors.
- Regulating the flow of data so that slow receivers are not swamped by fast senders.

Data Link layer

- Takes the packets from Network layer, and
- Encapsulates them into frames

Packets and Frames

Relationship between packets and frames.



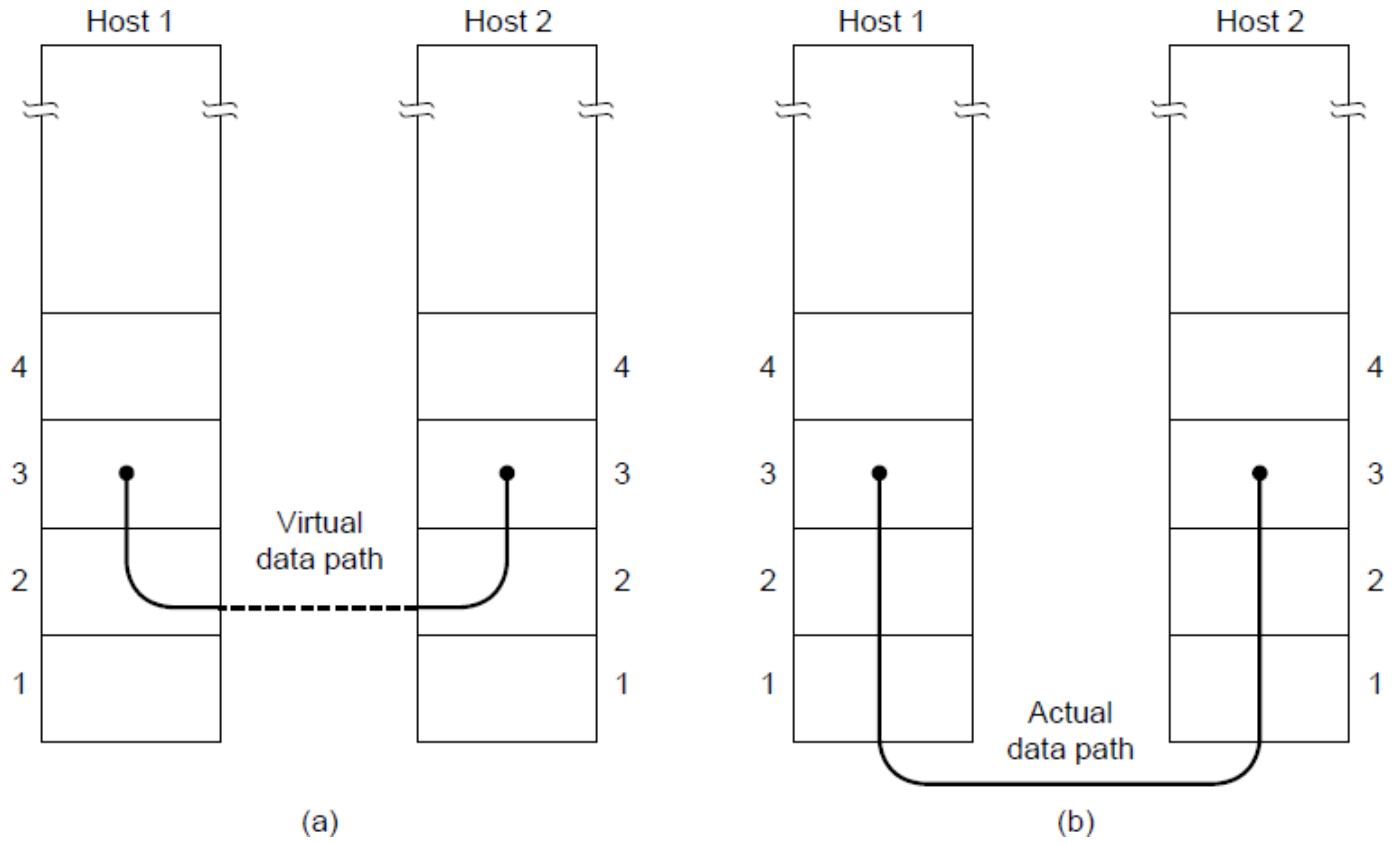
Services Provided to the Network Layer

Principal Service Function of the data link layer is to transfer the data from the network layer on the source machine to the network layer on the destination machine.

- Process in the network layer that hands some bits to the data link layer for transmission.
- Job of data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there (see figure in the next slide).

Network Layer Services

(a) Virtual communication. (b) Actual communication.



Possible Services Offered

- Unacknowledged connectionless service.
- Acknowledged connectionless service.
- Acknowledged connection-oriented service.

Unacknowledged Connectionless Service

It consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them.

Example: Ethernet, Voice over IP, etc. in all the communication channel were real time operation is more important that quality of transmission.

Acknowledged Connectionless Service

Each frame send by the Data Link layer is acknowledged and the sender knows if a specific frame has been received or lost.

Typically the protocol uses a specific time period that if has passed without getting acknowledgment it will re-send the frame.

This service is useful for commutation when an unreliable channel is being utilized (e.g., 802.11 WiFi).

Acknowledged Connection Oriented Service

Source and Destination establish a connection first.

Each frame sent is numbered

- Data link layer guarantees that each frame sent is indeed received.
- It guarantees that each frame is received only once and that all frames are received in the correct order.

Examples:

- Satellite channel communication,
- Long-distance telephone communication, etc.

Acknowledged Connection Oriented Service

Three distinct phases:

- Connection is established by having both side initialize variables and counters needed to keep track of which frames have been received and which ones have not.
- One or more frames are transmitted.
- Finally, the connection is released – freeing up the variables, buffers, and other resources used to maintain the connection.

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:

Types of Frames:-

Fixed-Size Framing

Variable-Size Framing

Framing

- To provide service to the network layer the data link layer must use the service provided to it by physical layer.
- Stream of data bits provided to data link layer is not guaranteed to be without errors.
- Errors could be:
  - Number of received bits does not match number of transmitted bits (deletion or insertion)
  - Bit Value
- It is up to data link layer to correct the errors if necessary.

Framing

- Transmission of the data link layer starts with breaking up the bit stream
  - into discrete frames
  - Computation of a checksum for each frame, and
  - Include the checksum into the frame before it is transmitted.
- Receiver computes its checksum error for a receiving frame and if it is different from the checksum that is being transmitted will have to deal with the error.
- Note : Checksum is a small-sized block of data derived from another block of digital data for the purpose of detecting errors that may have been introduced during transmission or storage

Framing Methods

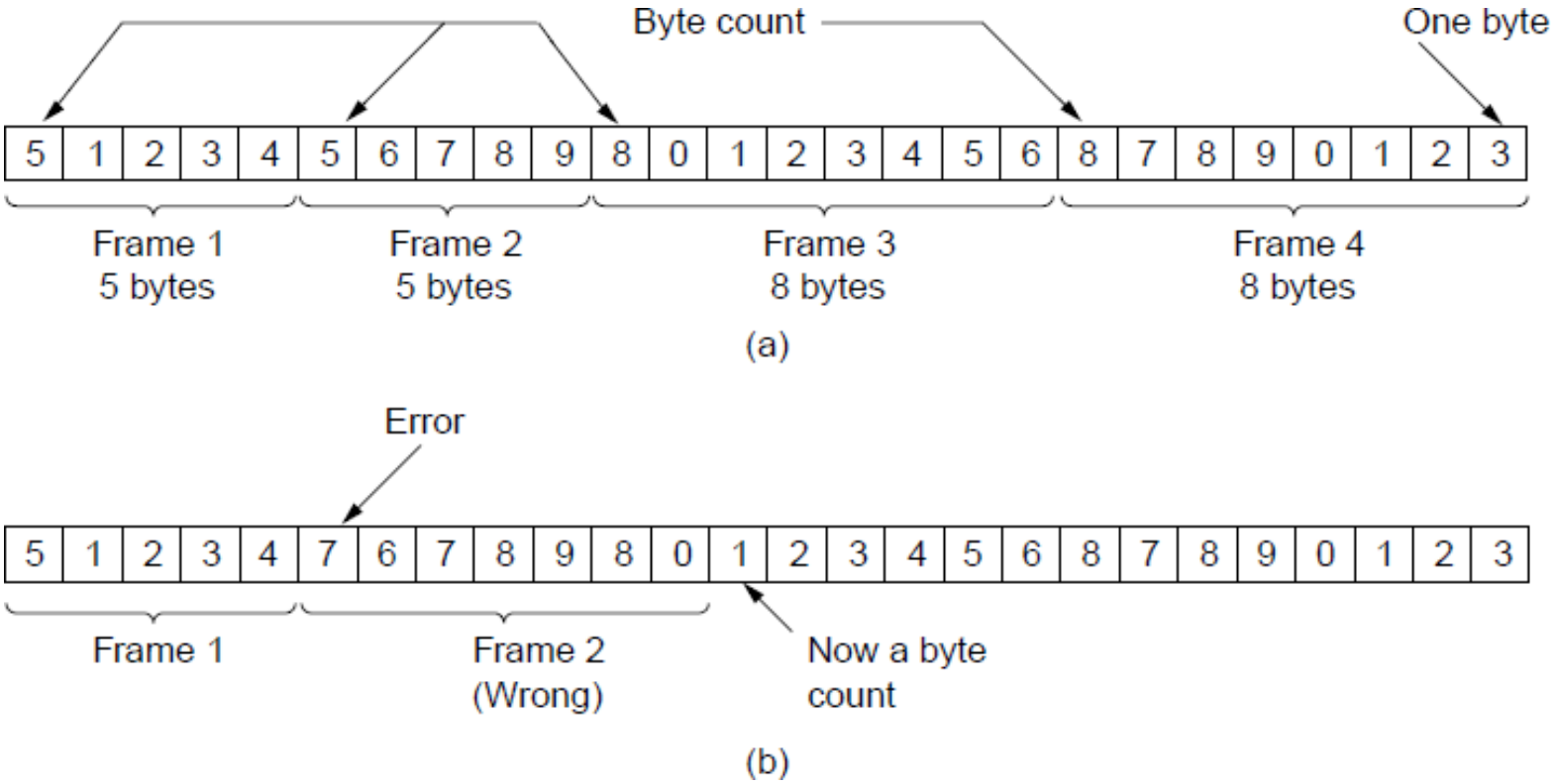
1. Byte count.
2. Flag bytes with byte stuffing
3. Flag bits with bit stuffing
4. Physical layer coding violations.

Byte Count Framing Method

- It uses a field in the header to specify the number of bytes in the frame.
- Once the header information is being received it will be used to determine end of the frame.
- See figure in the next slide:
- Trouble with this algorithm is that when the count is incorrectly received the destination will get out of synch with transmission.
  - Destination may be able to detect that the frame is in error but it does not have a means (in this algorithm) how to correct it.

Framing (1)

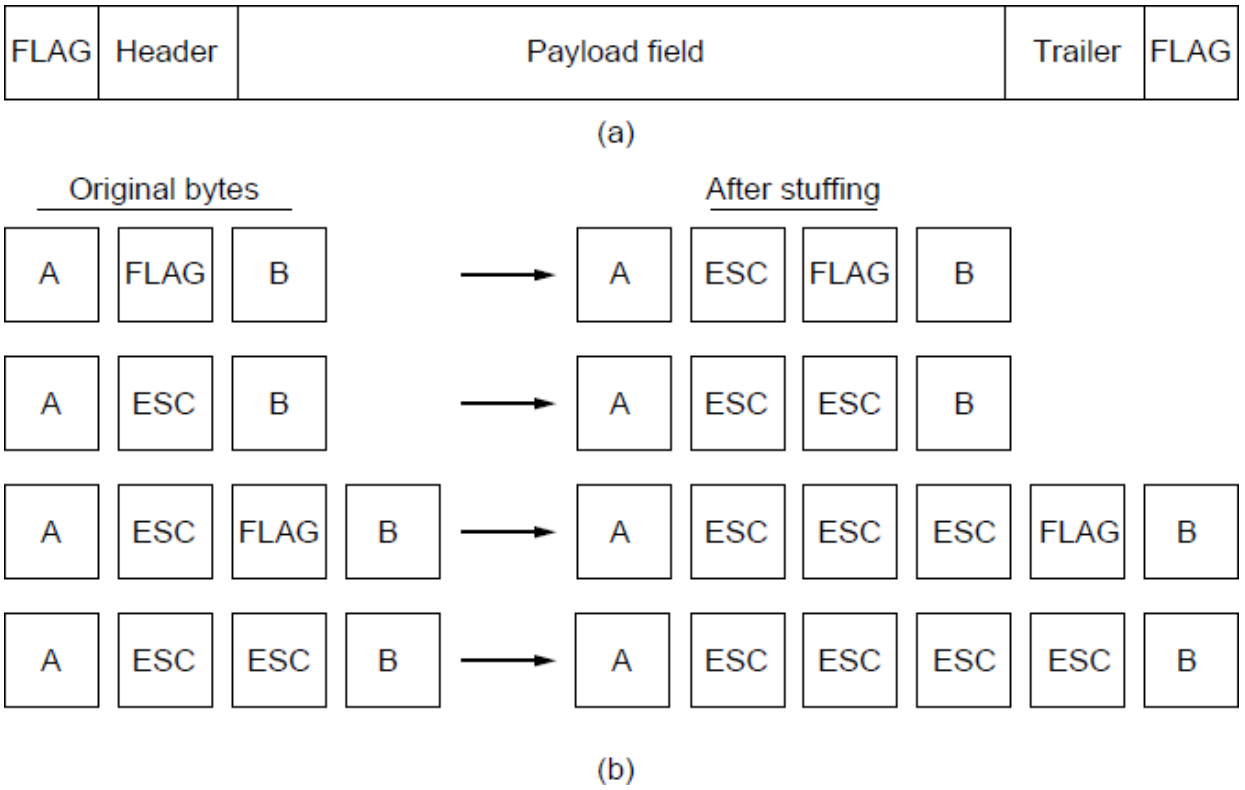
A byte stream. (a) Without errors. (b) With one error.



Flag Bytes with Byte Staffing Framing Method

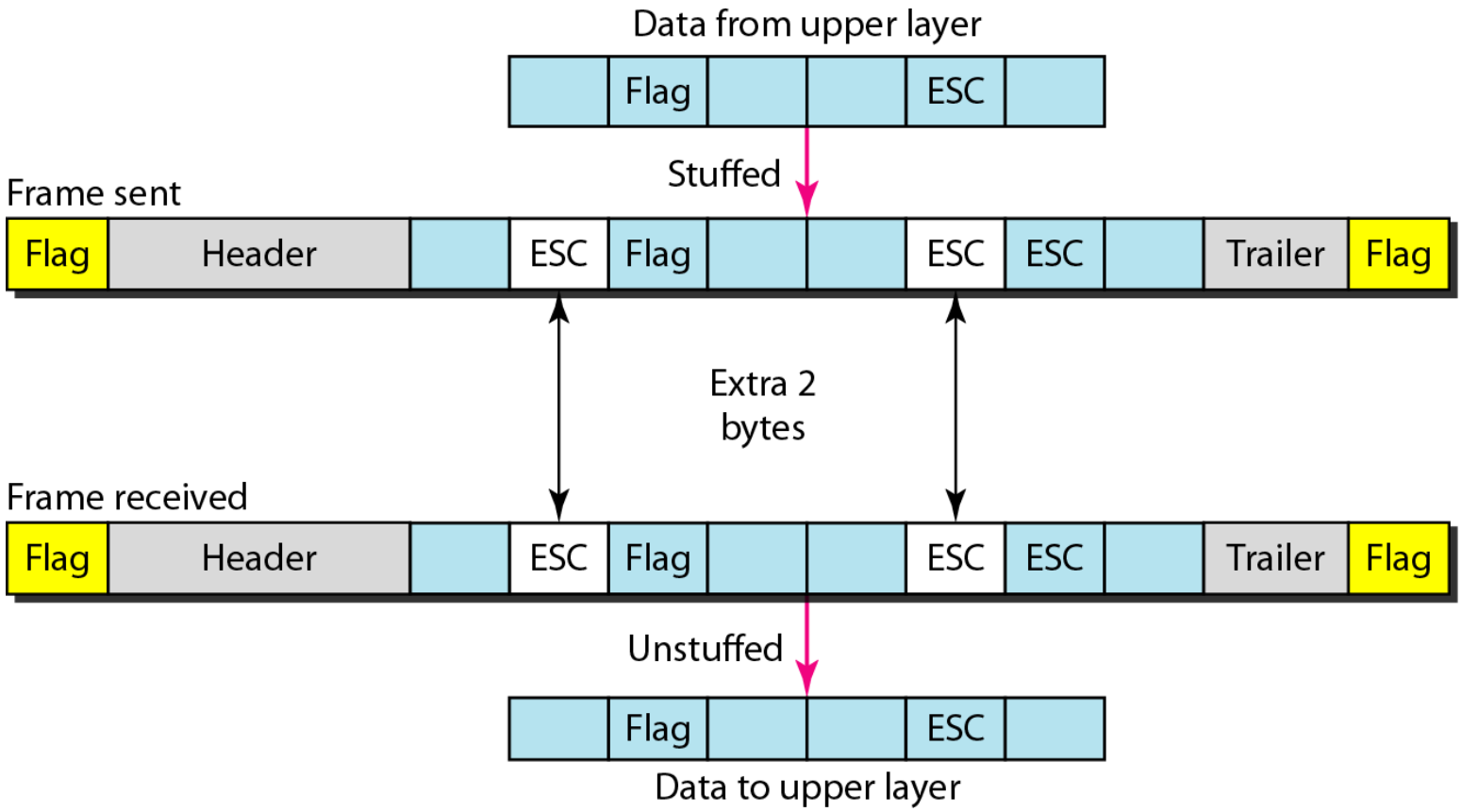
- This methods gets around the boundary detection of the frame by having each appended by the frame start and frame end special bytes.
- If they are the same (beginning and ending byte in the frame) they are called **flag byte**.
- In the next slide figure this byte is shown as FLAG.
- If the actual data contains a byte that is identical to the FLAG byte (e.g., picture, data stream, etc.) the convention that can be used is to have escape character inserted just before the "FLAG" character.

Framing (2)



- A frame delimited by flag bytes.
- Four examples of byte sequences before and after byte stuffing.

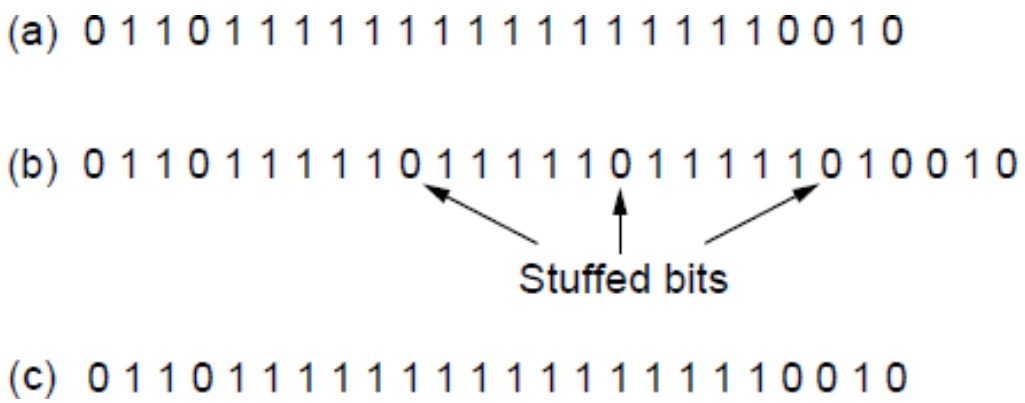
Figure *Byte stuffing and unstuffing*



Flag Bits with Bit Stuffing Framing Method

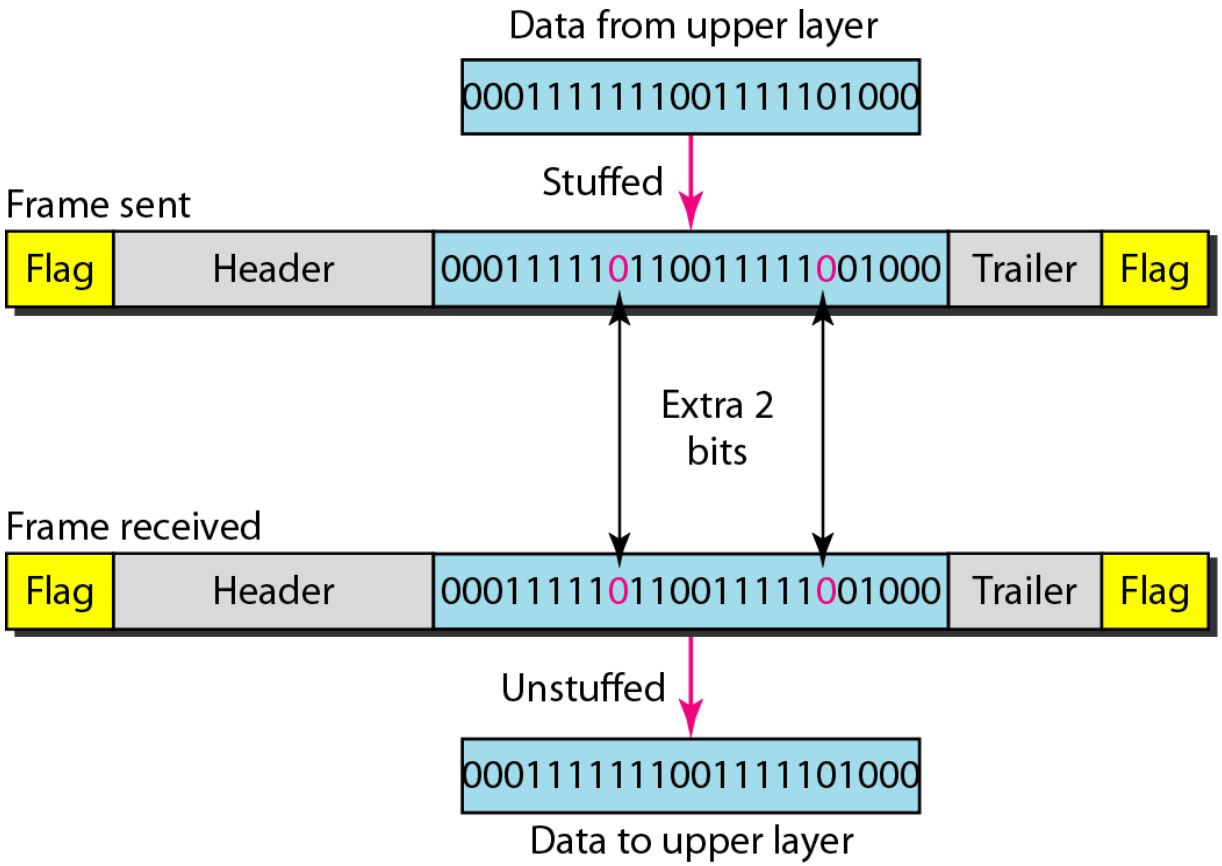
- This methods achieves the same thing as Byte Stuffing method by using Bits (1) instead of Bytes (8 Bits).
- It was developed for High-level Data Link Control (HDLC) protocol.
- Each frames begins and ends with a special bit patter:
  - 01111110 or 0x7E <- Flag Byte
  - Whenever the sender's data link layer encounters five consecutive 1s in the data it automatically stuffs a 0 bit into the outgoing bit stream.
  - USB uses bit stuffing

Framing (3)



Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

Figure *Bit stuffing and unstuffing*



Framing

- Many data link protocols use a combination of presented methods for safety. For example in Ethernet and 802.11 each frame begin with a well-defined pattern called a preamble.
- Preamble is typically 72 bits long
- It is then followed by a length fileld.

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.

Error Control

- Error control in the data link layer is a critical design issue ensuring reliable data transmission over potentially unreliable physical links.
- It involves mechanisms to detect and, in some cases, correct errors that may occur during the transmission of data frames.

Flow Control

- Important Design issue for the cases when the sender is running on a fast powerful computer and receiver is running on a slow low-end machine.
- Two approaches:
  1. Feedback-based flow control
  2. Rate-based flow control

Feedback-based Flow Control

- Receiver sends back information to the sender giving it permission to send more data, or
- Telling sender how receiver is doing

Rate-based Flow Control

- Built in mechanism that limits the rate at which sender may transmit data, without the need for feedback from the receiver.

Flow control 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.

Outline

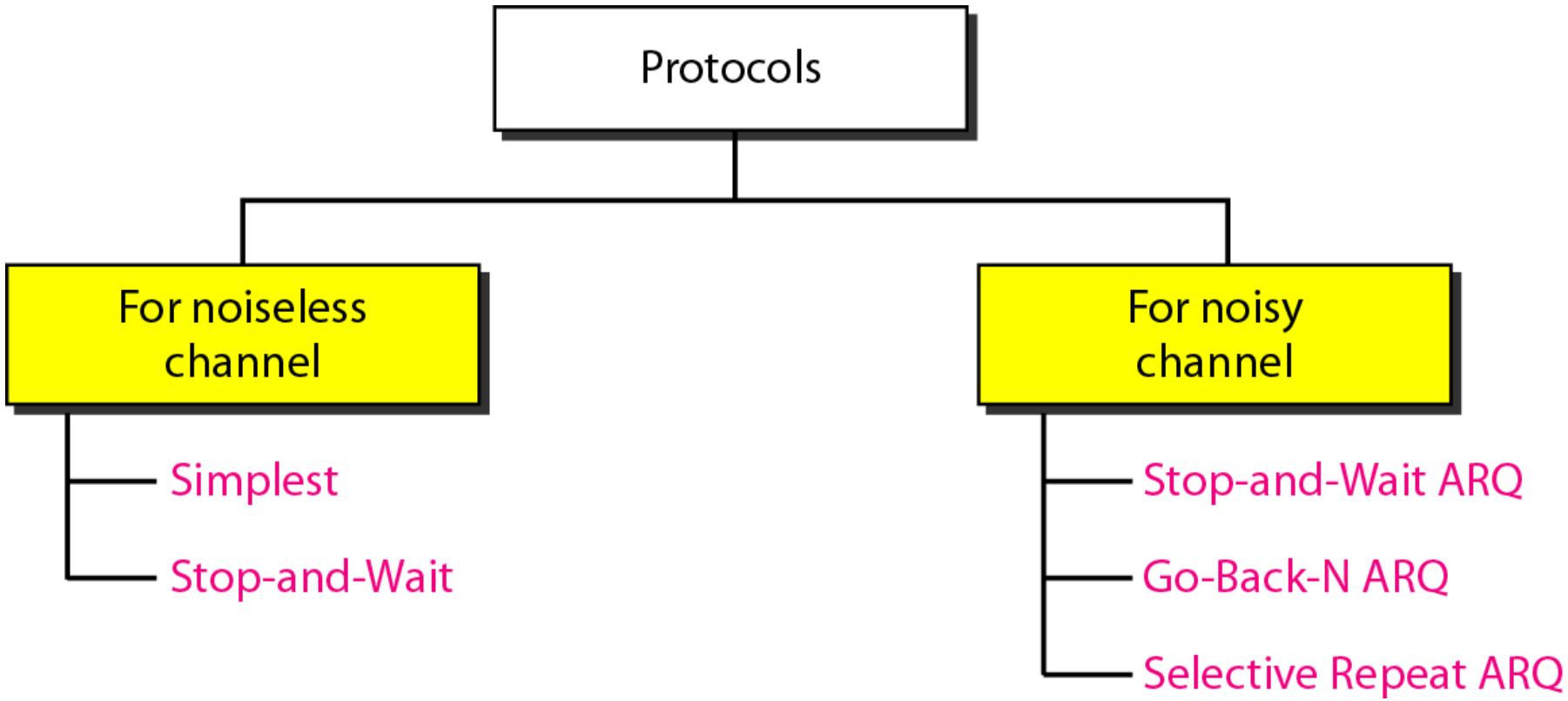
Design Issues: Services to Network Layer,Framing,Error Control and Flow Control

Flow Control Protocols: Unrestricted Simplex, Stop and Wait, Sliding Window Protocol

Error Control: Parity Bits, Hamming Codes (11/12-bits) and CRC.

WAN Connectivity : PPP and HDLC

Figure *Taxonomy of Flow Control protocols*



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

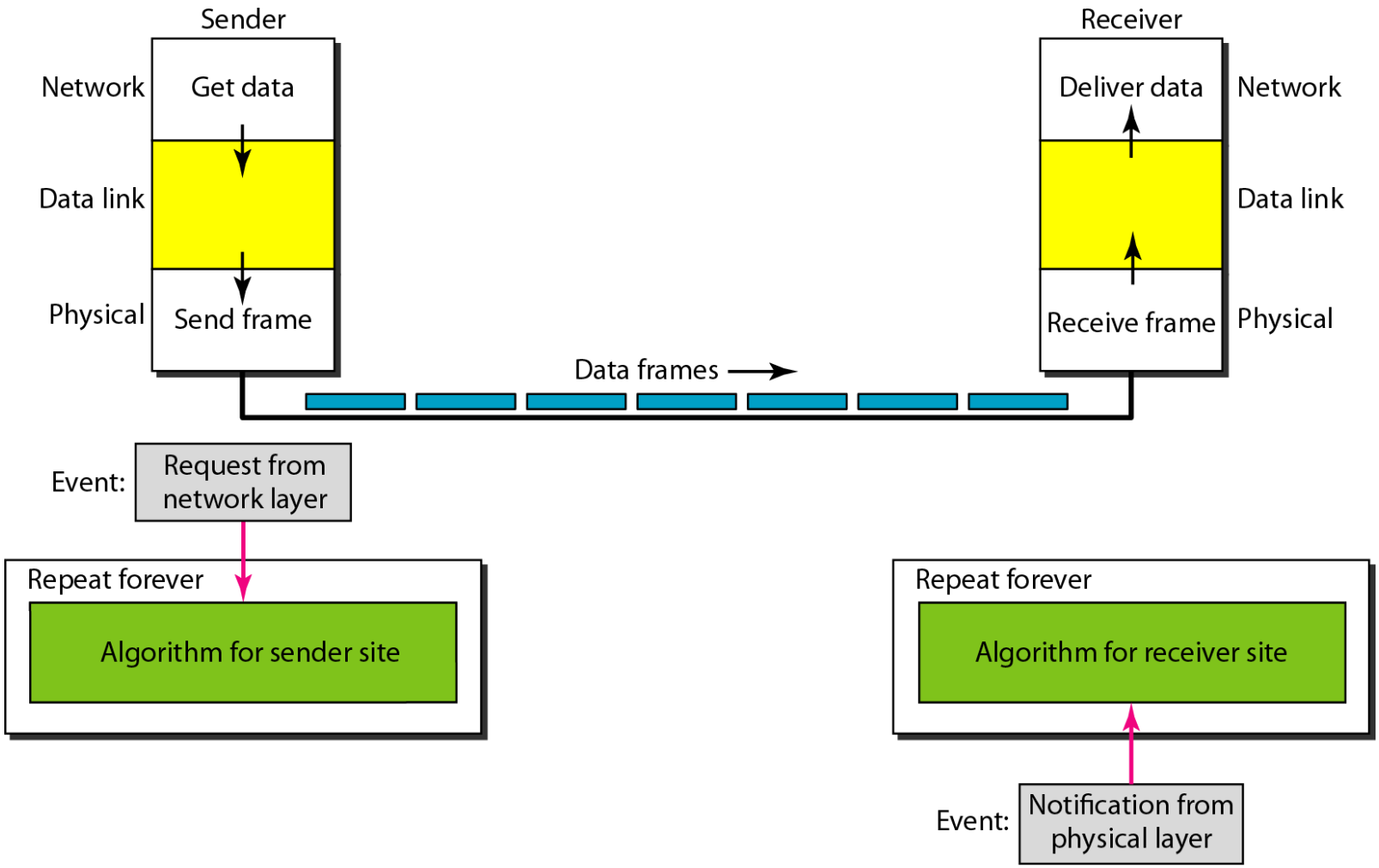


Figure *Flow diagram for Simplex Protocol*

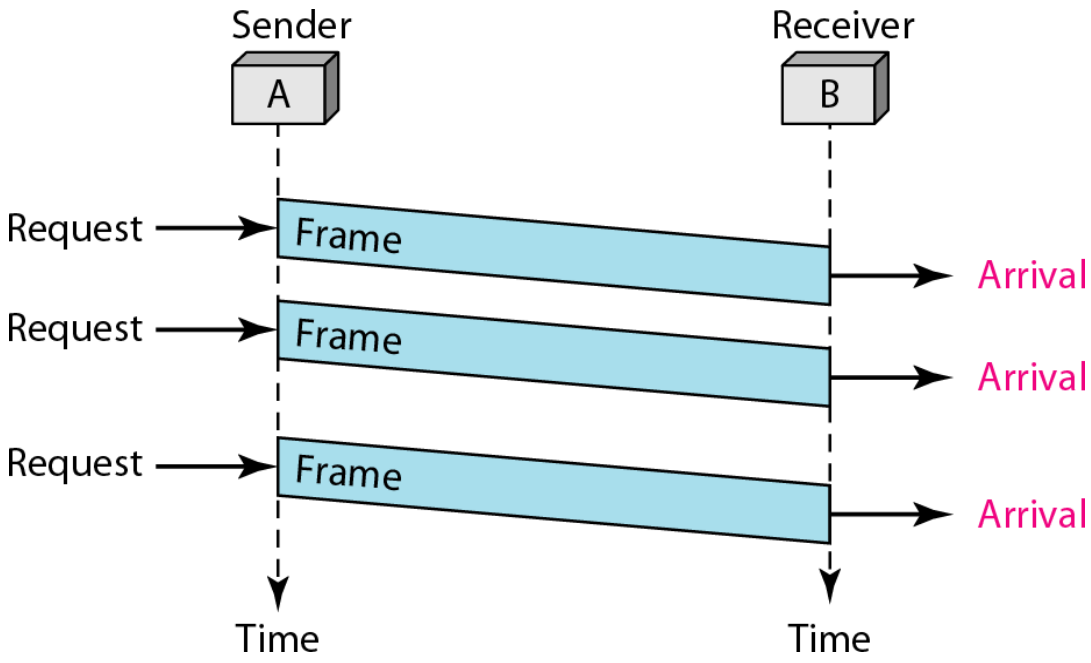


Figure *Design of Stop-and-Wait Protocol*

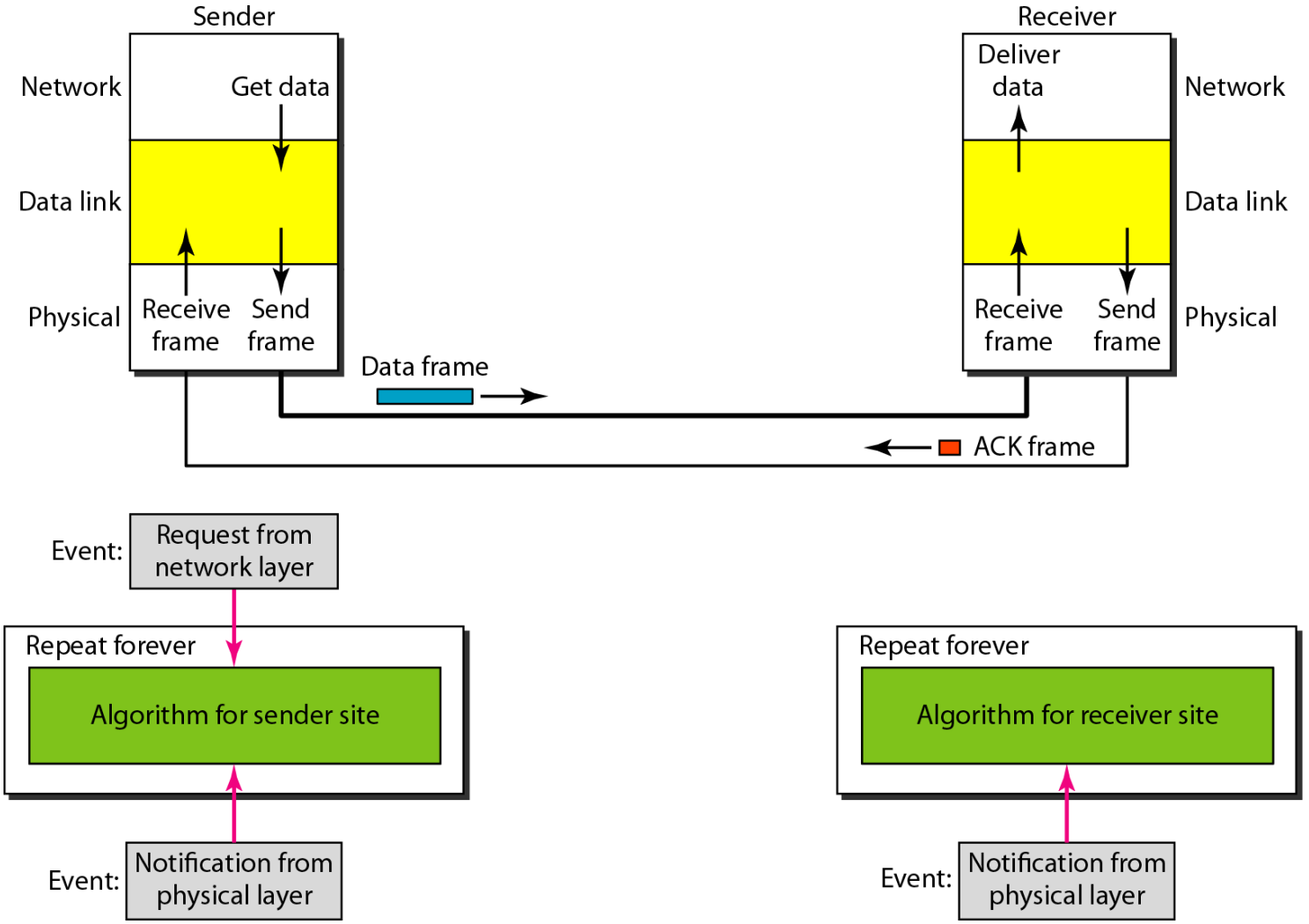
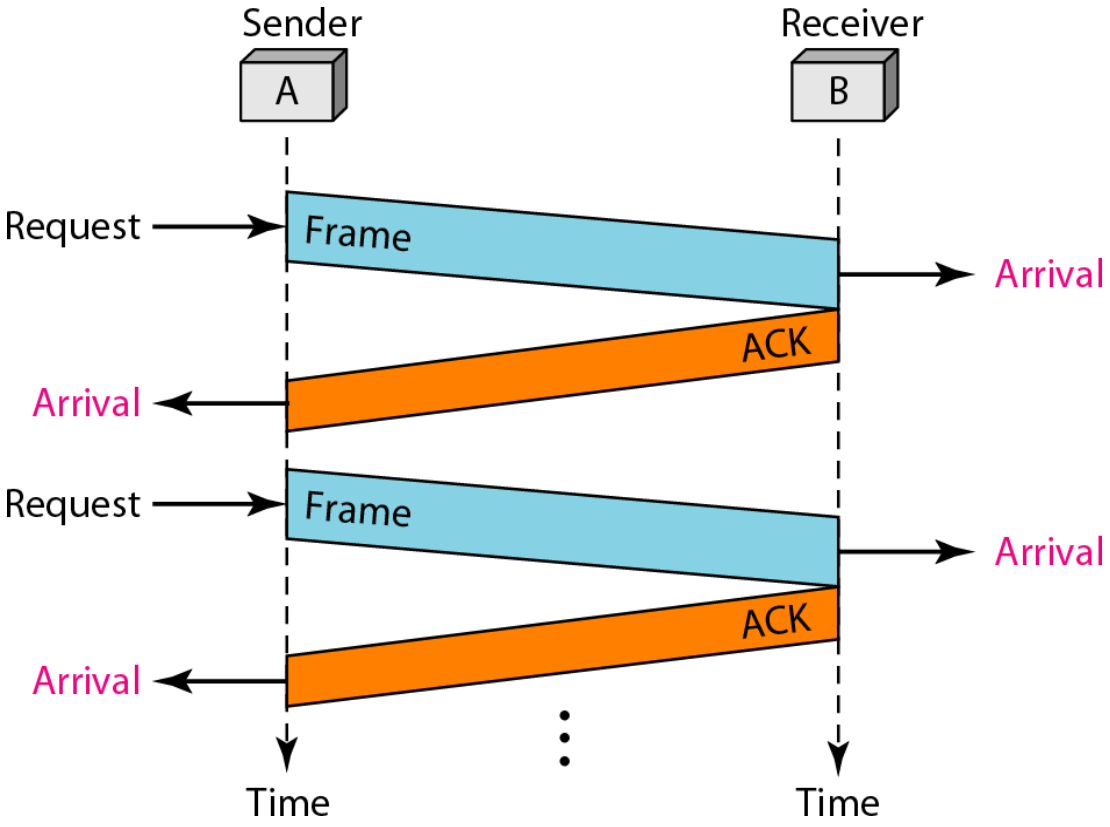


Figure *Flow diagram for Stop-and-Wait Protocol*



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 flow control.

Topics discussed in this section:

Stop-and-Wait Automatic Repeat Request

Go-Back-N Automatic Repeat Request → **Sliding window protocol**

Selective Repeat Automatic Repeat Request → **Sliding window protocol**

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.

Sliding Window Protocol

Assumes two-way communication (full duplex). It uses two types of frames: Data and Ack

The basic idea of sliding window protocol is that both sender and receiver keep a ``window" of acknowledgment.

The sender keeps the value of expected acknowledgment; while the receiver keeps the value of expected receiving frame.

When it sender an acknowledgment from the receiver, the sender advances the window.

When it receives the expected frame, the receiver advances the window.

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

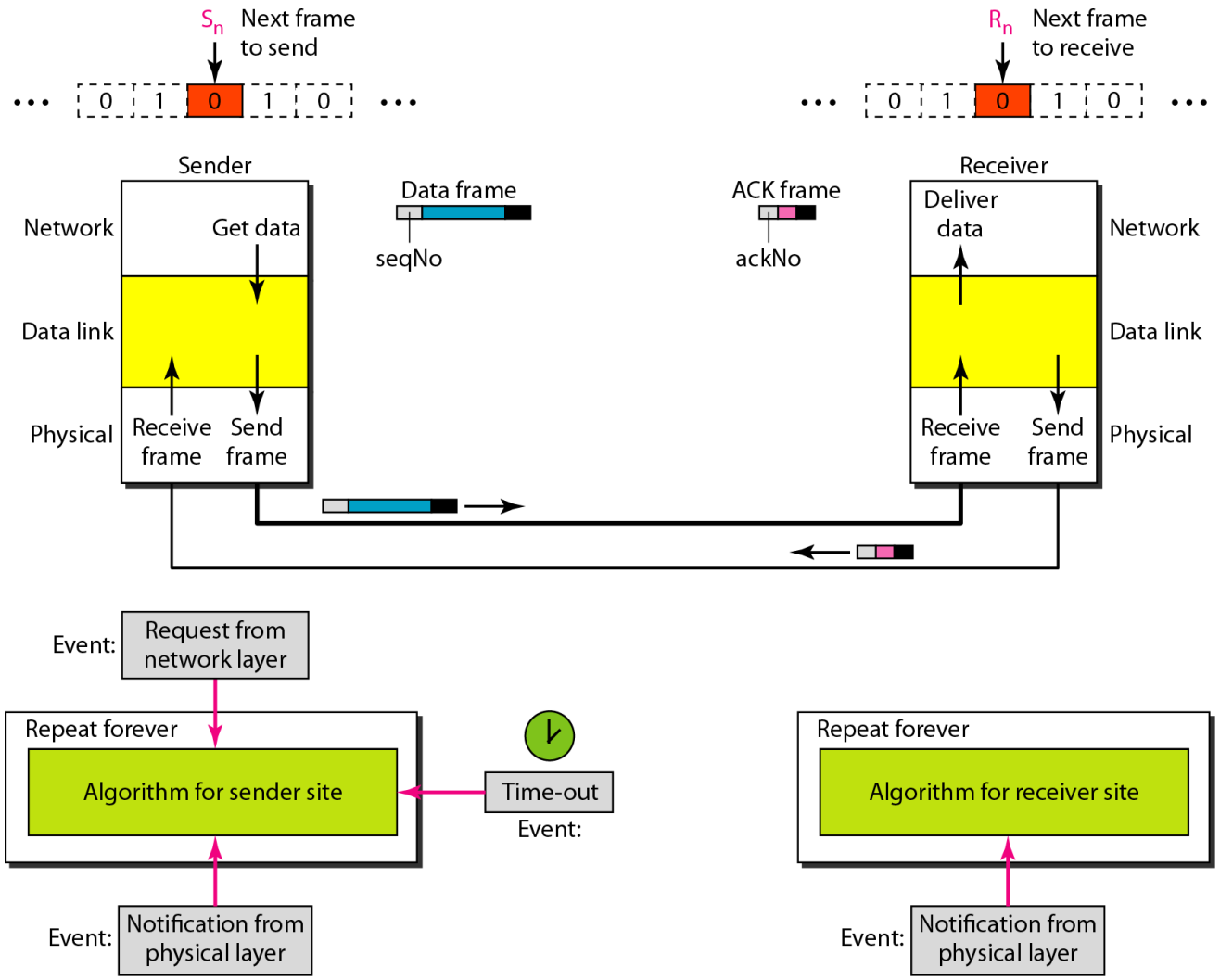
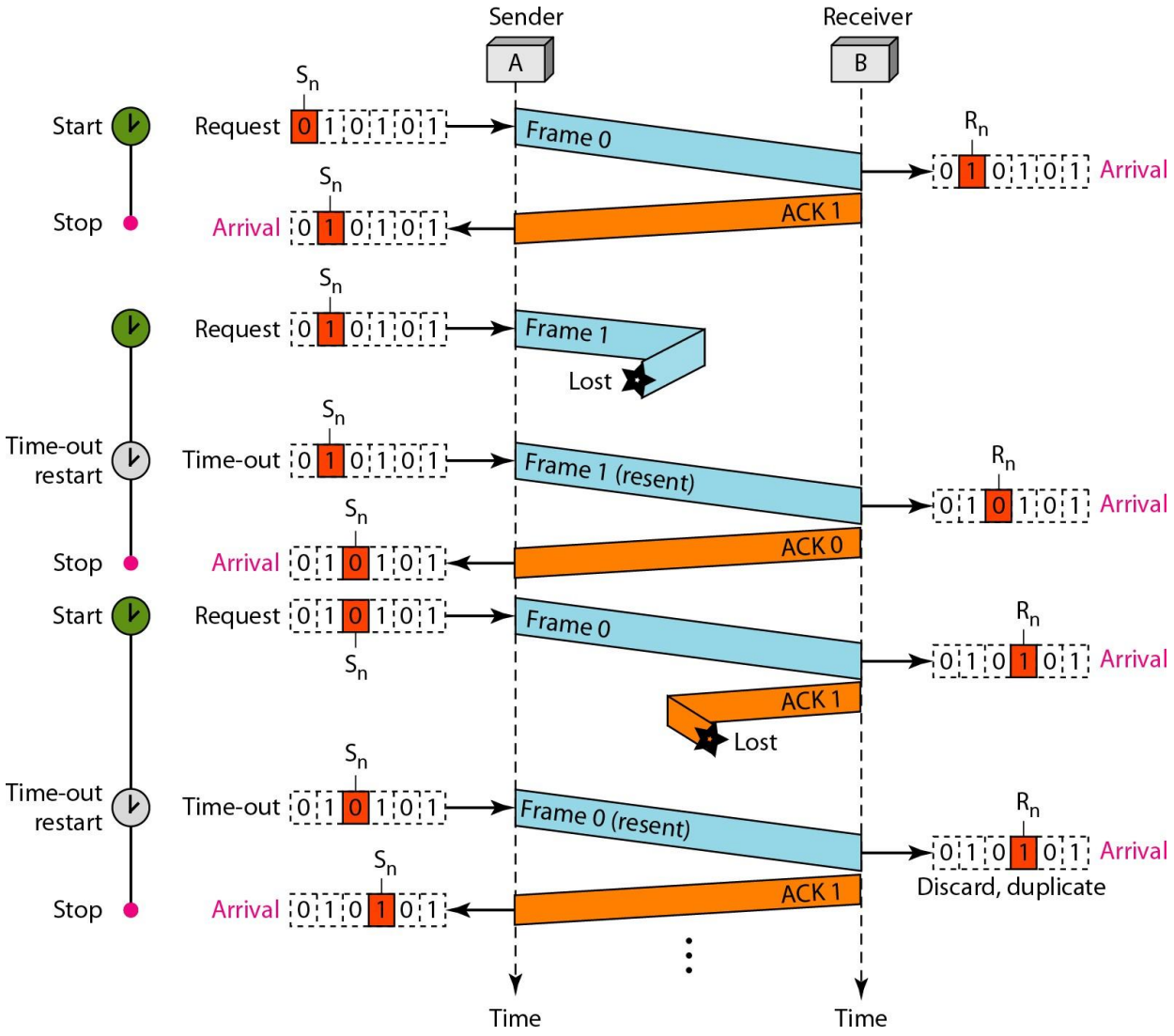


Figure *Flow diagram for Example-Stop-and-Wait ARQ Protocol*



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}.

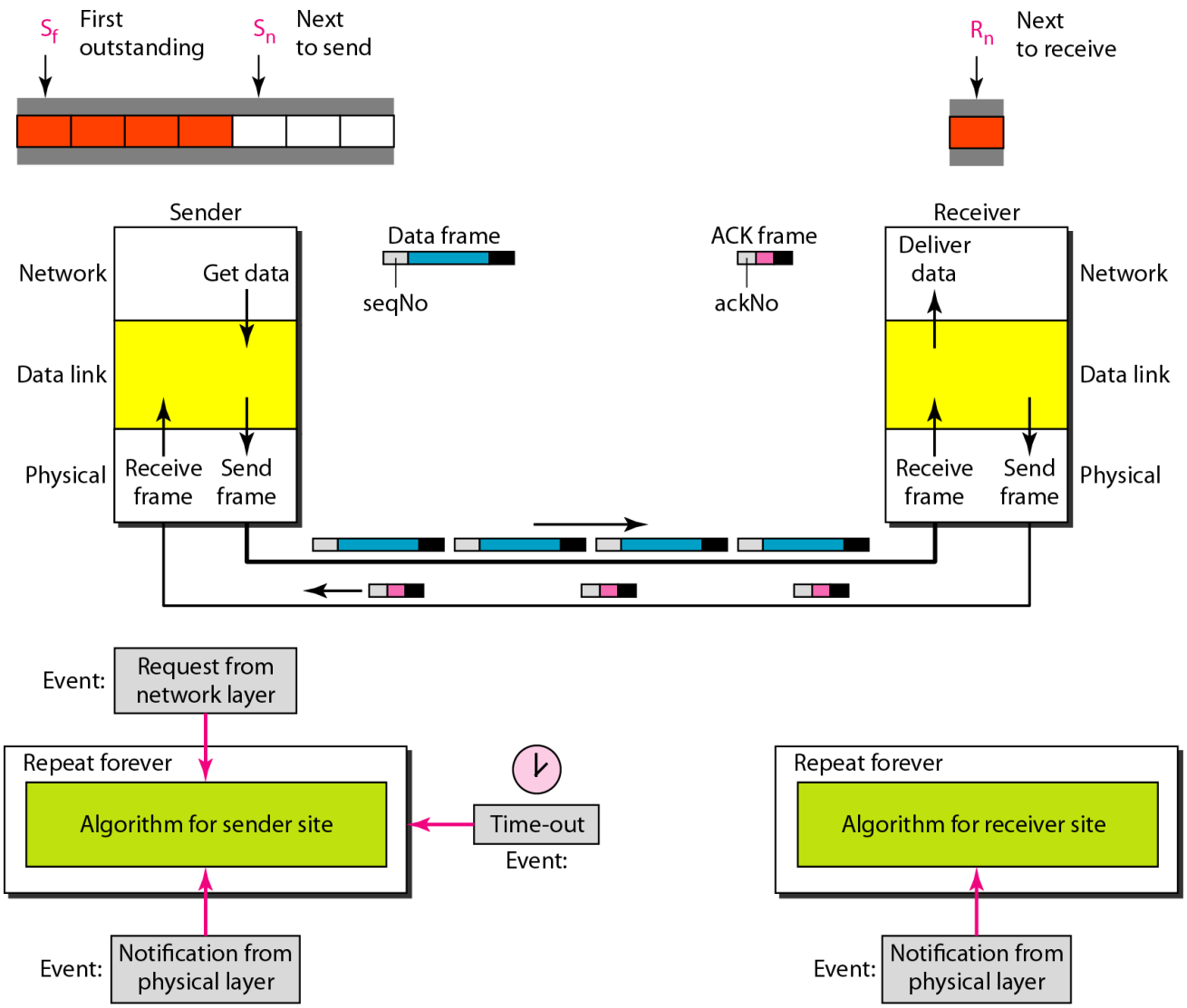
Note

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 *Design of Go-Back-N ARQ*



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.

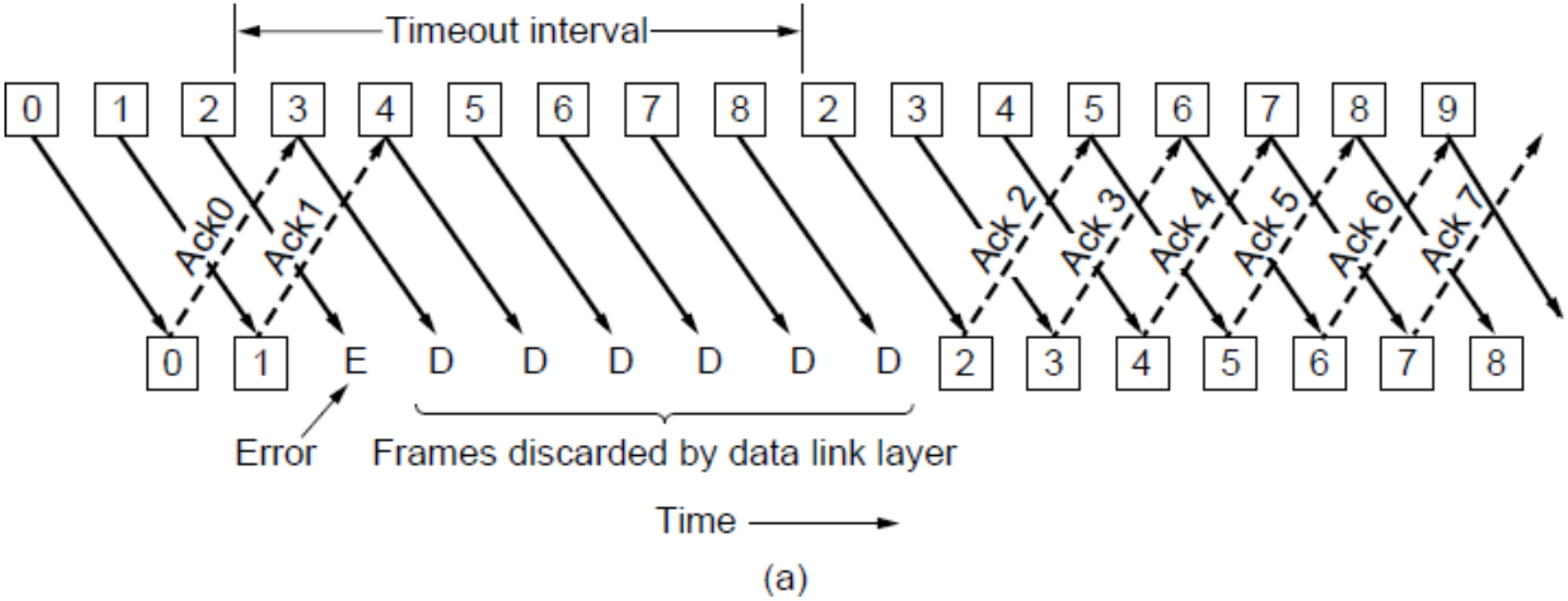
Note

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

Go-back-n

- If there is one frame k missing, the receiver simply discard all subsequent frames k+1, k+2, …, sending no acknowledgments. So the sender will retransmit frames from k onwards.
- This effectively sets the receiver window size to be 1.
- This can be a waste of bandwidth.

Go-back-n

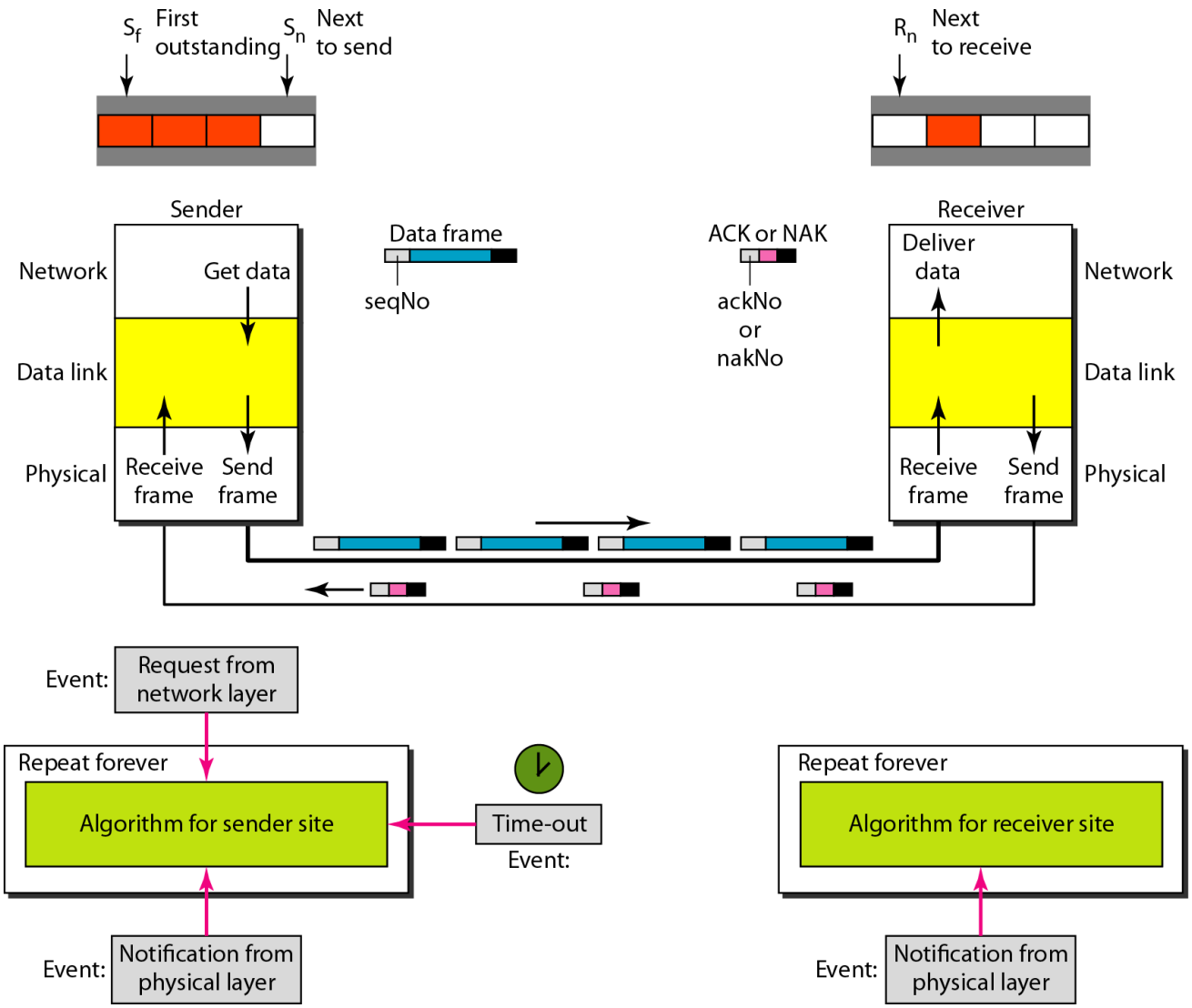

Timeout interval
0 1 2 3 4 5 6 7 8 2 3 4 5 6 7 8 9
Ack0
Ack1
Ack 2
Ack 3
Ack 4
Ack 5
Ack 6
Ack 7
0 1 E D D D D D D 2 3 4 5 6 7 8
Error
Frames discarded by data link layer
Time

(a)

Selective repeat

- Another strategy is to re-send only the ones that are actually lost or damaged. The receiver buffers all the frames after the lost one. When the sender finally noticed the problem (e.g. no ack for the lost frame is received within time-out limit), the sender retransmits the frame in question

Figure 11.20 *Design of Selective Repeat ARQ*



Note

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

Selective repeat

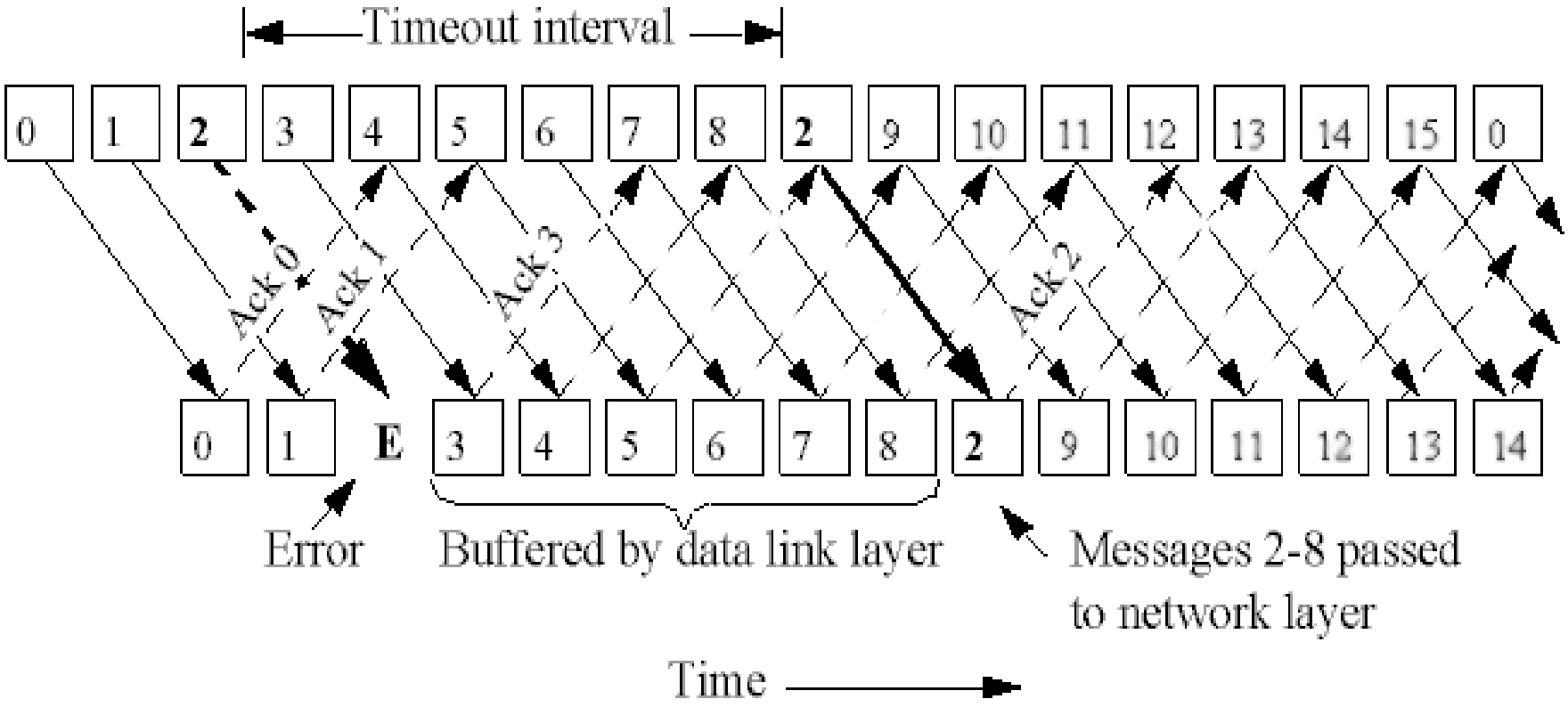
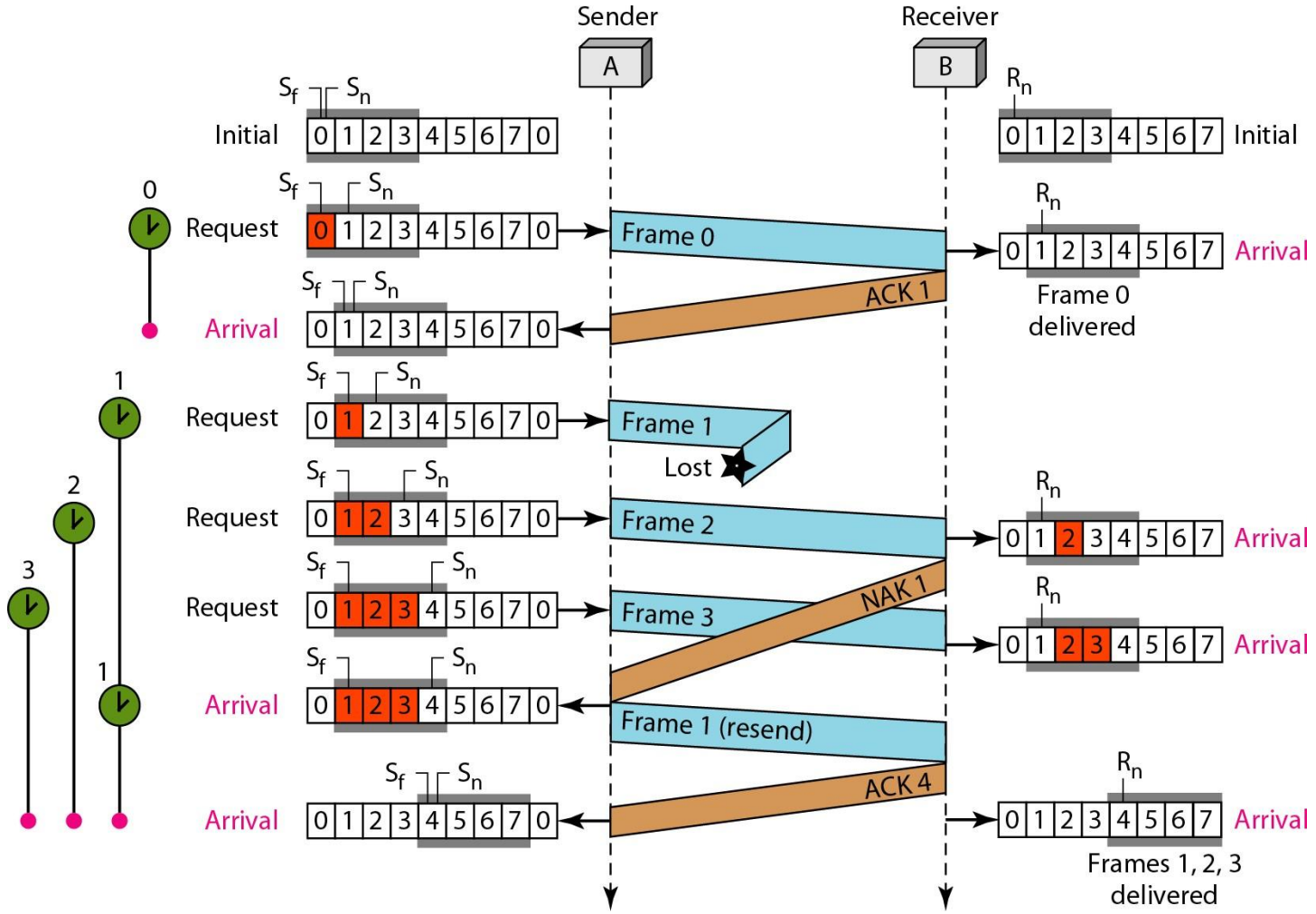


Figure *Flow diagram for Selective Repeat*



Outline

Design Issues: Services to Network Layer,Framing,Error Control and Flow Control

Flow Control Protocols: Unrestricted Simplex, Stop and Wait, Sliding Window Protocol

Error Control: Parity Bits, Hamming Codes (11/12-bits) and CRC.

WAN Connectivity : PPP and HDLC

Note

Data can be corrupted during transmission.

Some applications require that errors be detected and corrected.

INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correction.

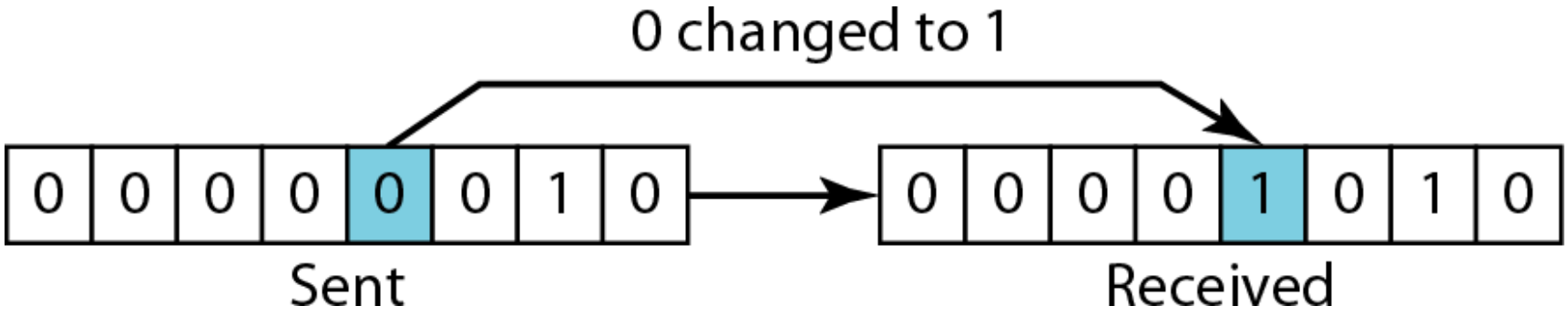
Topics discussed in this section:

Types of Errors
Redundancy
Detection Versus Correction
Forward Error Correction Versus Retransmission
Coding
Modular Arithmetic

Introduction

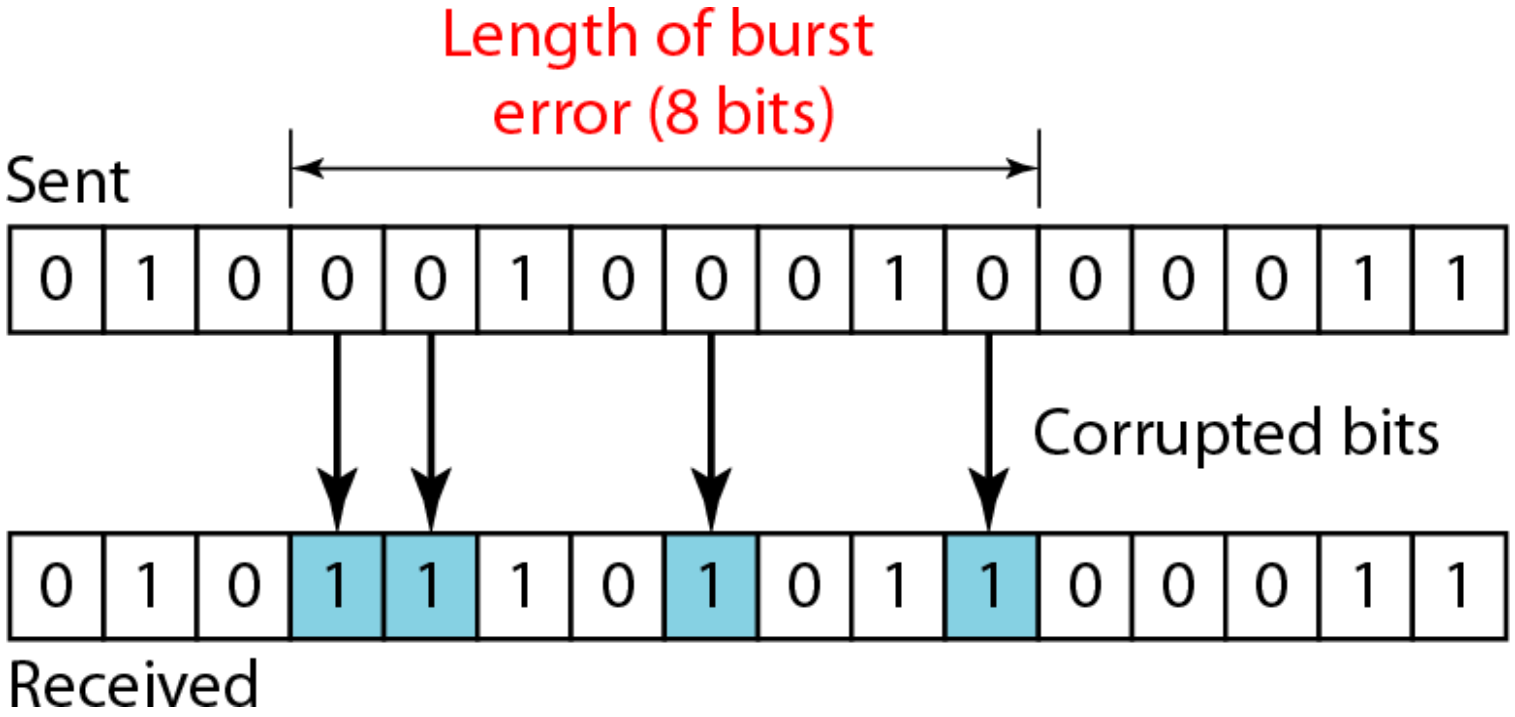
- Error transmissions are caused by Interference
- Types of errors
  - Single-bit error
    - Only 1 bit of a given data unit is changed from 1 to 0 or from 0 to 1.
  - Burst error
    - 2 or more bits in the data unit changed from 1 to 0 or from 0 to 1.
- Redundancy
  - The central concept in detecting or correcting error is redundancy.To be able to detect or correct errors, we need to send **some extra bits** with our data.

Figure *Single-bit error*



In a single-bit error, only 1 bit in the data unit has changed.

Figure *Burst error of length 8*

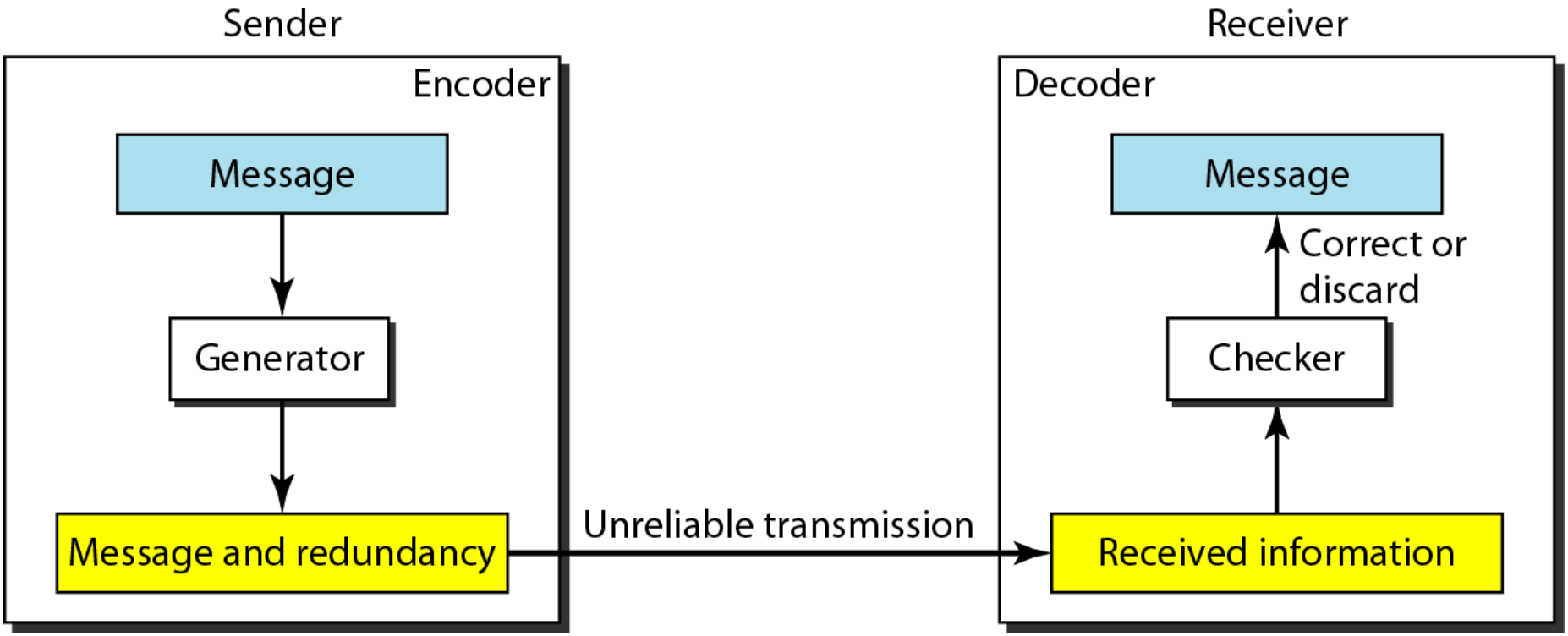


A burst error means that 2 or more bits in the data unit have changed.

Detection versus Correction

- The correction of errors is more difficult than the detection.
- Error detection
  - Looking only to see if any error has occurred
  - The answer is a simple yes or no.
- Error correction
  - Forward error correction
    - The process in which the receiver tries to guess the message by using redundant bits.
  - Retransmission
    - The receiver detects the occurrence of an error and asks the sender to resend the message.

Figure *The structure of encoder and decoder*



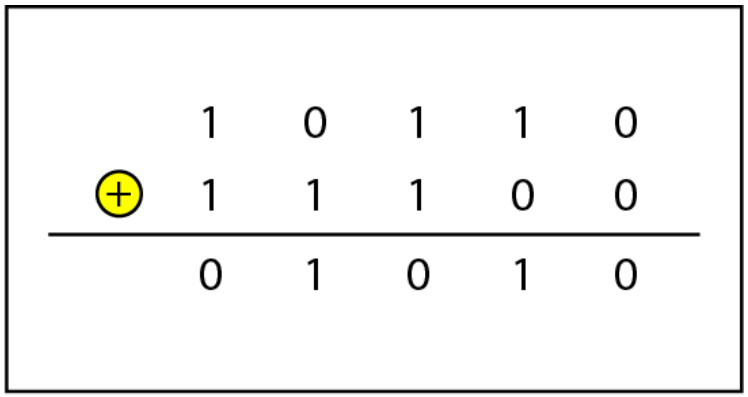
Note

In modulo-N arithmetic, we use only the integers in the range 0 to N −1, inclusive.

Figure *XORing of two single bits or two words*

a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.



c. Result of XORing two patterns

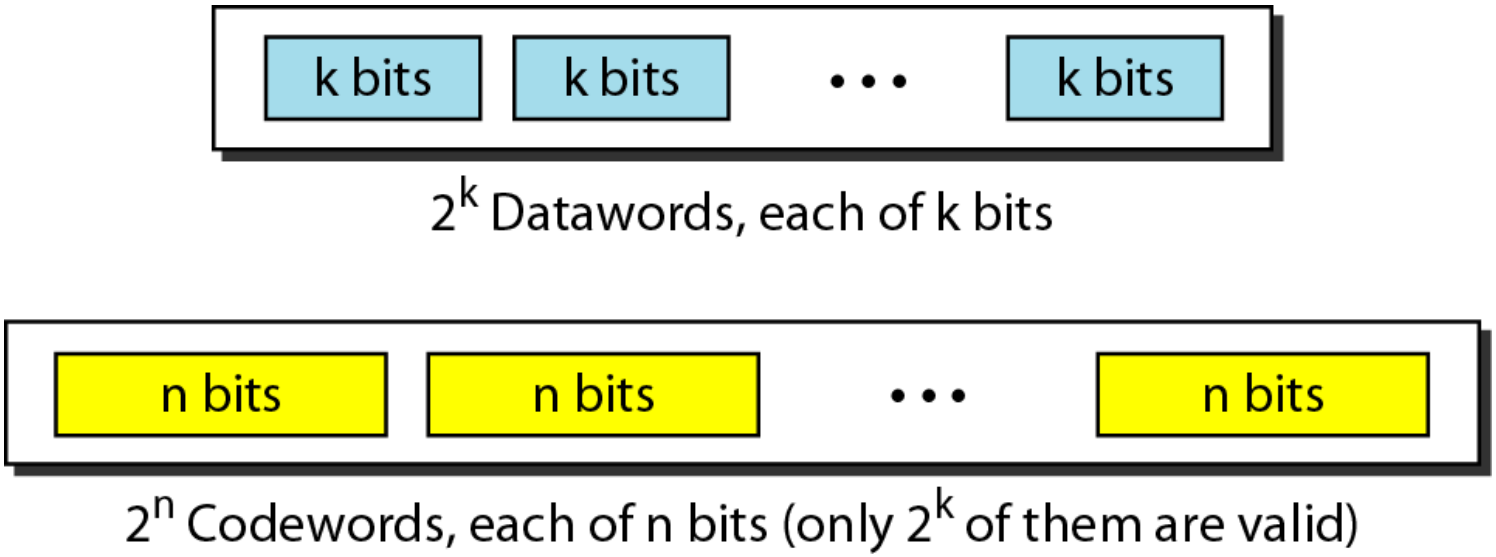
BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords.

Topics discussed in this section:

Error Detection
Error Correction
Hamming Distance
Minimum Hamming Distance

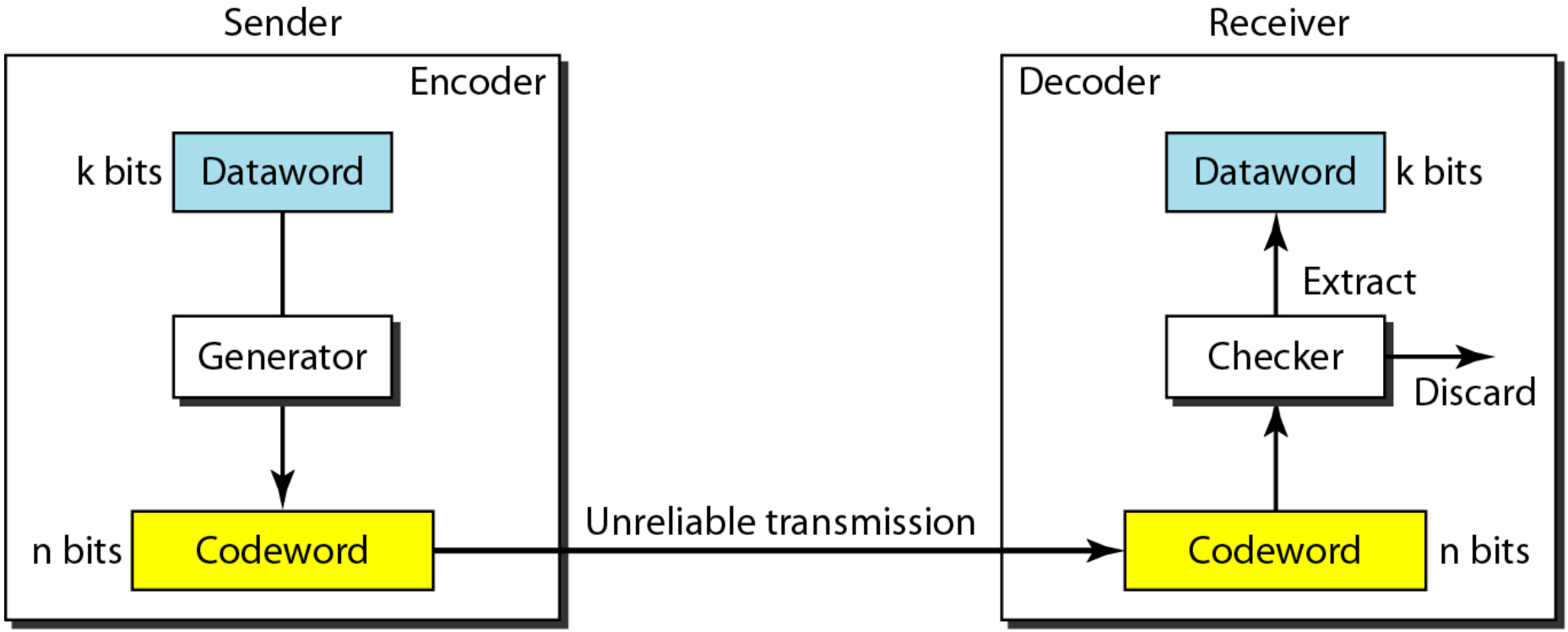
Figure ***Datawords and codewords in block coding***



Example 1

The 4B/5B block coding discussed in Chapter 4 is a good example of this type of coding. In this coding scheme, $k = 4$ and $n = 5$. As we saw, we have $2^k = 16$ datawords and $2^n = 32$ codewords. We saw that 16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

Figure ***Process of error detection in block coding***



Example 2

Let us assume that k = 2 and n = 3. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. *The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.*

Example2 (continued)

2. ***The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.***

3. ***The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.***

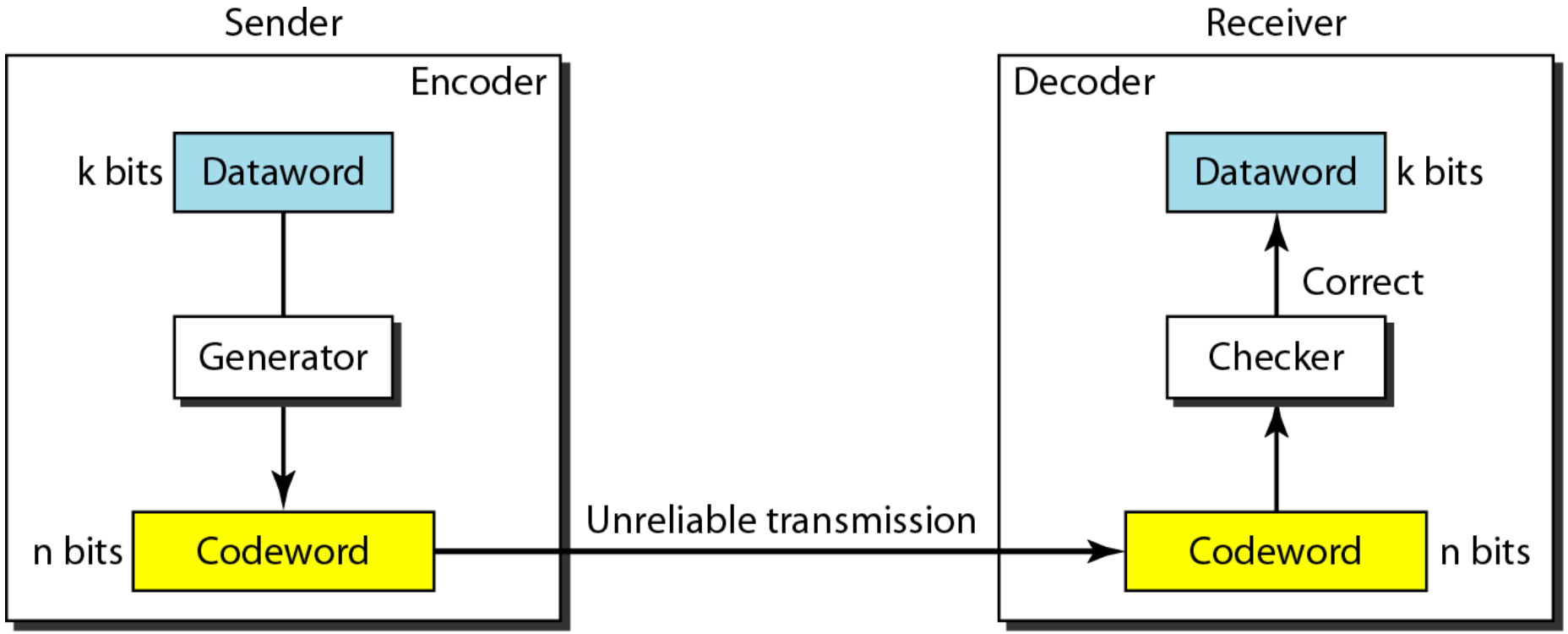
Table *A code for error detection*

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Note

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Figure *Structure of encoder and decoder in error correction*



Example 3

Let us add more redundant bits to Example 10.2 to see if the receiver can correct an error without knowing what was actually sent. We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords. Table 10.2 shows the datawords and codewords. Assume the dataword is 01. The sender creates the codeword 01011. The codeword is corrupted during transmission, and 01001 is received. First, the receiver finds that the received codeword is not in the table. This means an error has occurred. The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

Example 3 (continued)

1. ***Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.***

2. ***By the same reasoning, the original codeword cannot be the third or fourth one in the table.***

3. ***The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.***

Table *A code for error correction*

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Note

The Hamming distance between two words is the number of differences between corresponding bits.

Example 4

Let us find the Hamming distance between two pairs of words.

1. ***The Hamming distance d(000, 011) is 2 because***

$000 \oplus 011$ is 011 (two 1s)

2. ***The Hamming distance d(10101, 11110) is 3 because***

$10101 \oplus 11110$ is 01011 (three 1s)

Note

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Example 5

Find the minimum Hamming distance of the coding scheme in Table 10.1.

Solution

We first find all Hamming distances.

$d(000, 011) = 2$	$d(000, 101) = 2$	$d(000, 110) = 2$	$d(011, 101) = 2$
$d(011, 110) = 2$	$d(101, 110) = 2$		

The d_{min} in this case is 2.

Example 6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

Solution

We first find all the Hamming distances.

$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

The d_{min} in this case is 3.

Note

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Example 7

The minimum Hamming distance for our first code scheme (Table 10.1) is 2. This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Example 8

Our second block code scheme (Table 10.2) has $d_{min} = 3$. This code can detect up to two errors. Again, we see that when any of the valid codewords is sent, two errors create a codeword which is not in the table of valid codewords. The receiver cannot be fooled.

However, some combinations of three errors change a valid codeword to another valid codeword. The receiver accepts the received codeword and the errors are undetected.

Figure *Geometric concept for finding d_{min} in error detection*

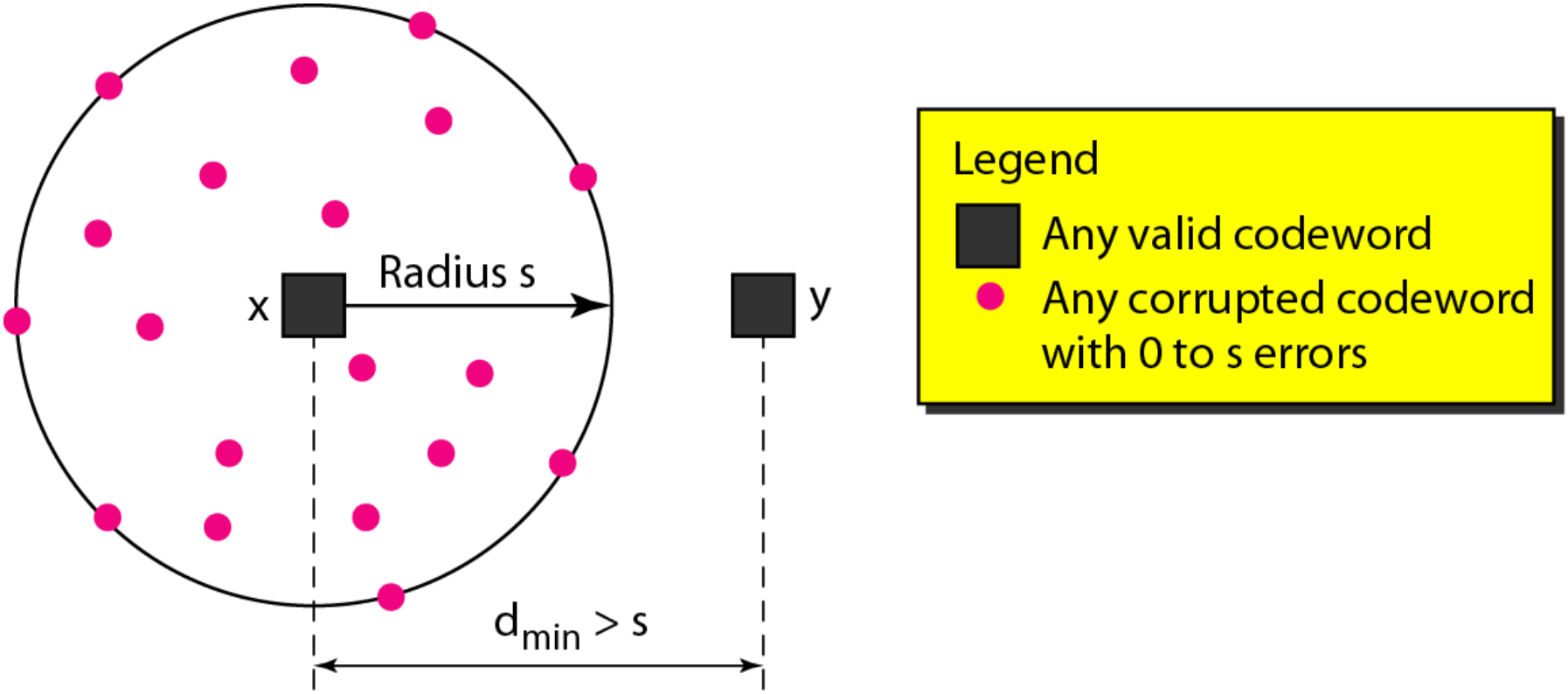
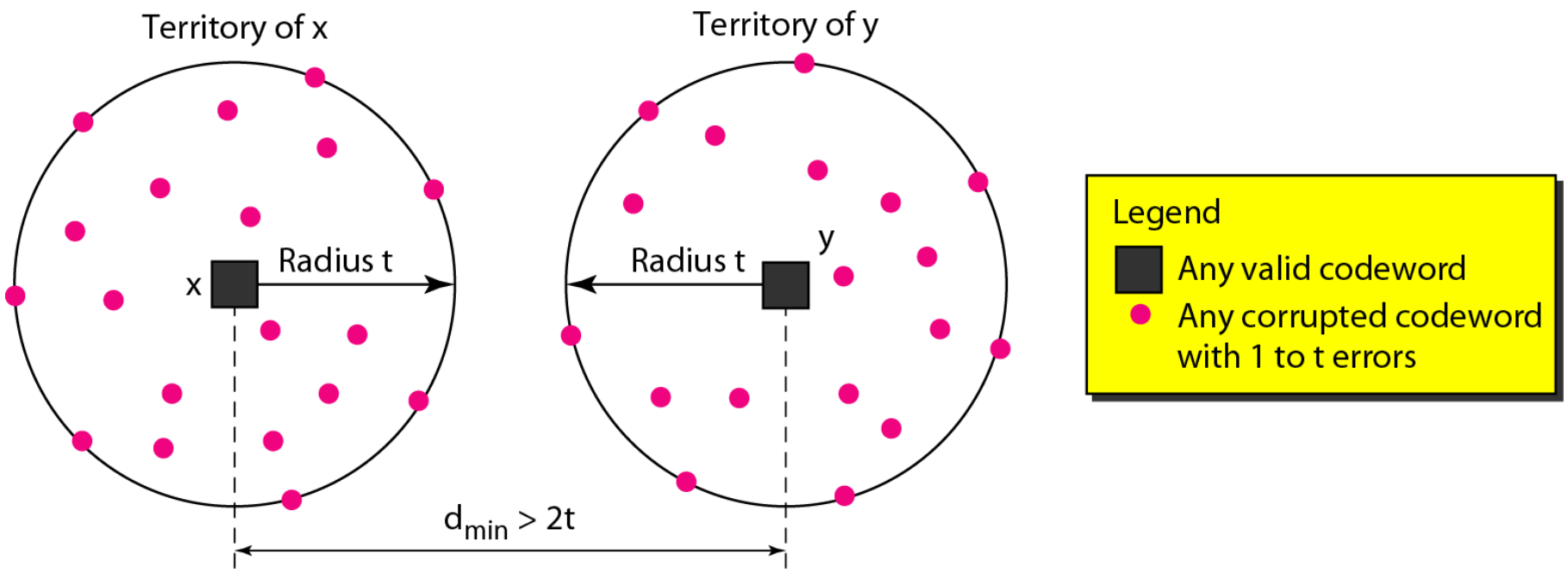


Figure *Geometric concept for finding d_{min} in error correction*



Note

To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

Example 9

A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?

Solution

This code guarantees the detection of up to three errors ($s = 3$), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. Error correction codes need to have an odd minimum distance (3, 5, 7, . . .).

LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called linear block codes. A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

Topics discussed in this section:

Minimum Distance for Linear Block Codes

Some Linear Block Codes

Note

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Example 10

Let us see if the two codes we defined in Table 10.1 and Table 10.2 belong to the class of linear block codes.

1. ***The scheme in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.***

2. ***The scheme in Table 10.2 is also a linear block code. We can create all four codewords by XORing two other codewords.***

Example 11

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$. In our second code (Table 10.2), the numbers of 1s in the nonzero codewords are 3, 3, and 4. So in this code we have $d_{min} = 3$.

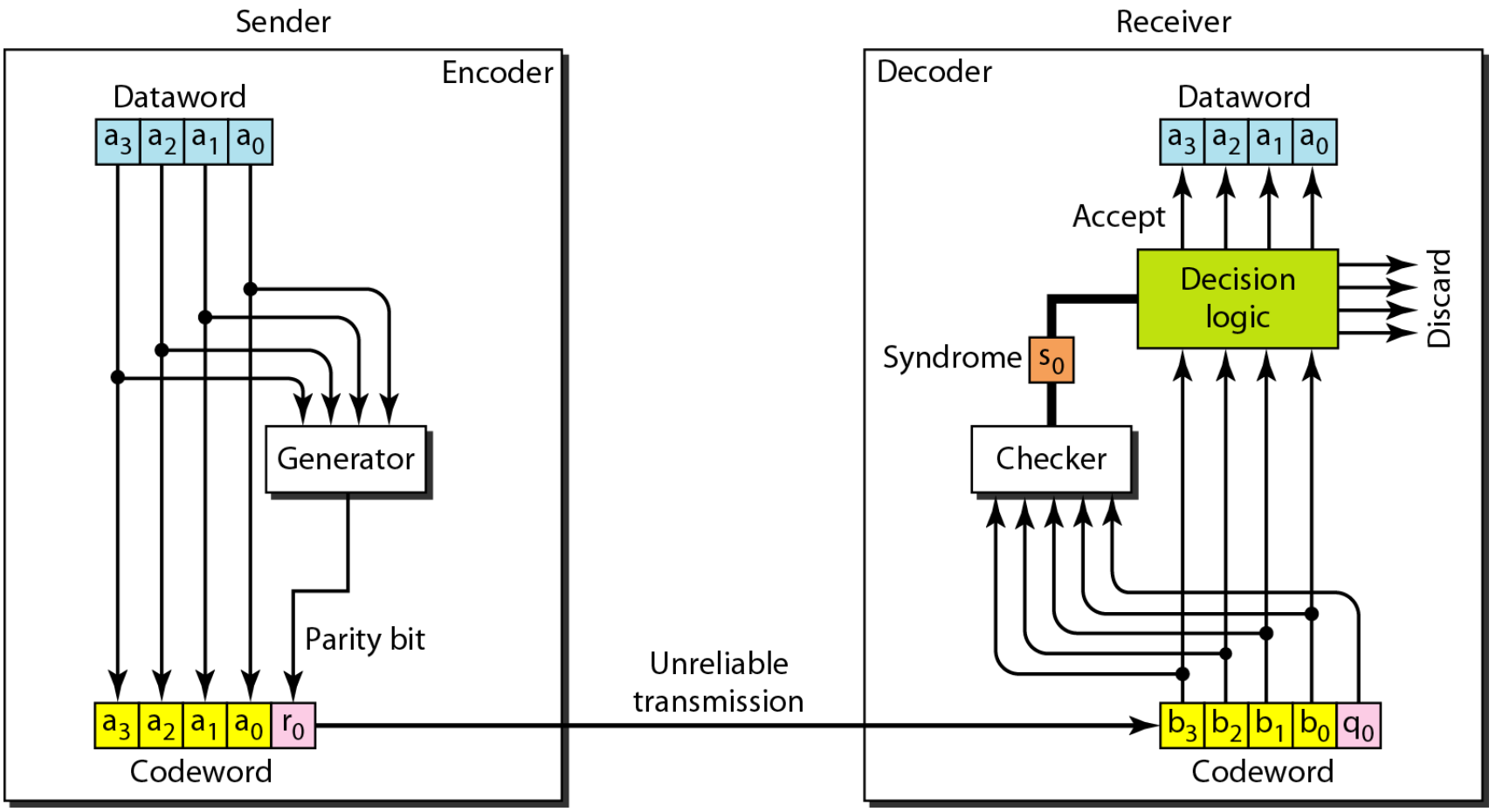
Note

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$.

Table ***Simple parity-check code C(5, 4)***

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure ***Encoder and decoder for simple parity-check code***



Example 12

Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1. ***No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.***
2. ***One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.***
3. ***One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created.***

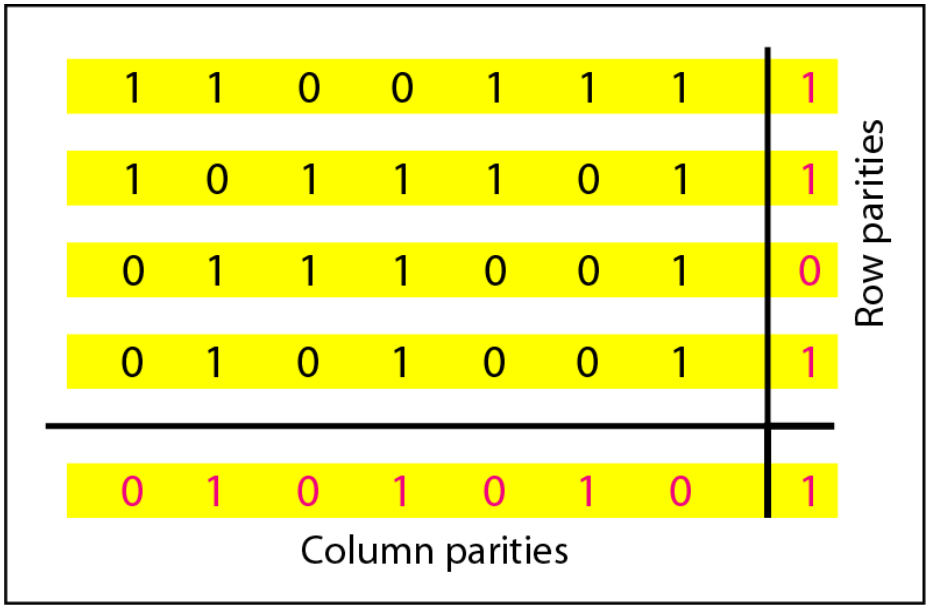
Example 12 (continued)

4. *An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value.*
5. *Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.*

Note

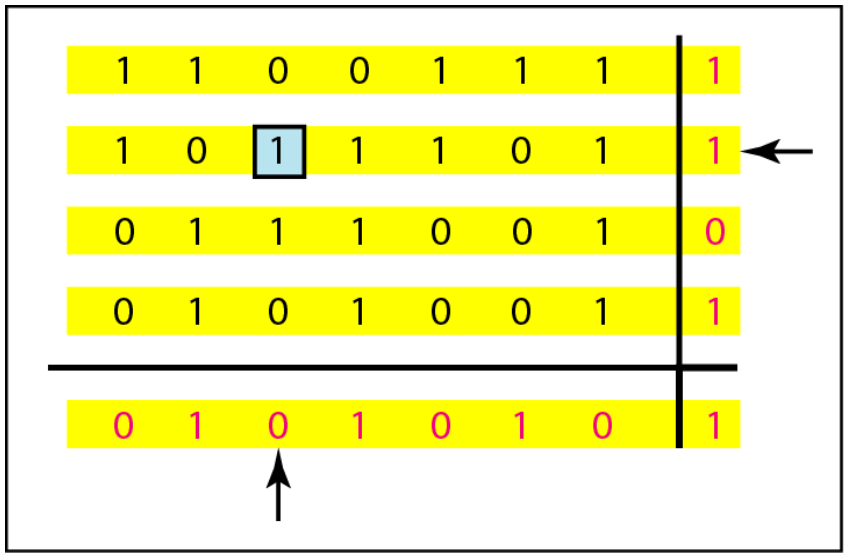
A simple parity-check code can detect an odd number of errors.

Figure *Two-dimensional parity-check code*

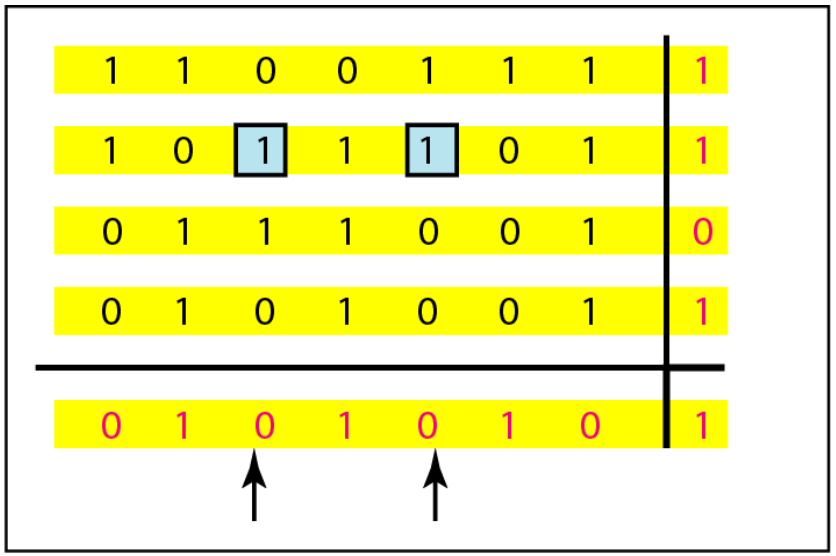


a. Design of row and column parities

Figure *Two-dimensional parity-check code*

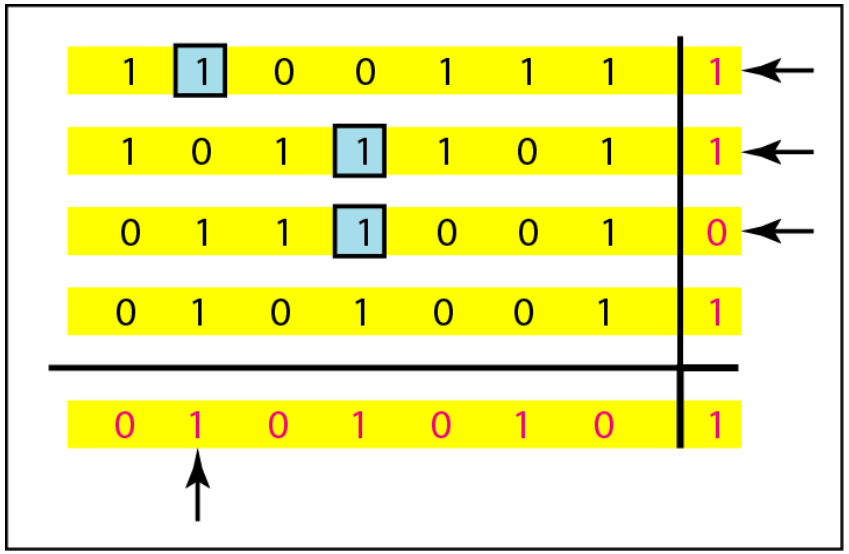


b. One error affects two parities

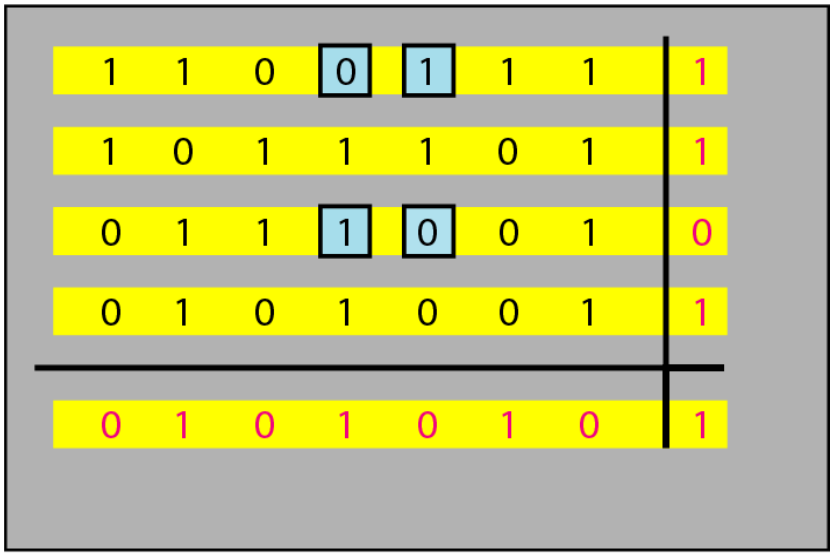


c. Two errors affect two parities

Figure *Two-dimensional parity-check code*



d. Three errors affect four parities



e. Four errors cannot be detected

Note

All Hamming codes discussed in this book have $d_{min} = 3$.

The relationship between m and n in these codes is $n = 2m - 1$.

Table ***Hamming code C(7, 4)***

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Figure *The structure of the encoder and decoder for a Hamming code*

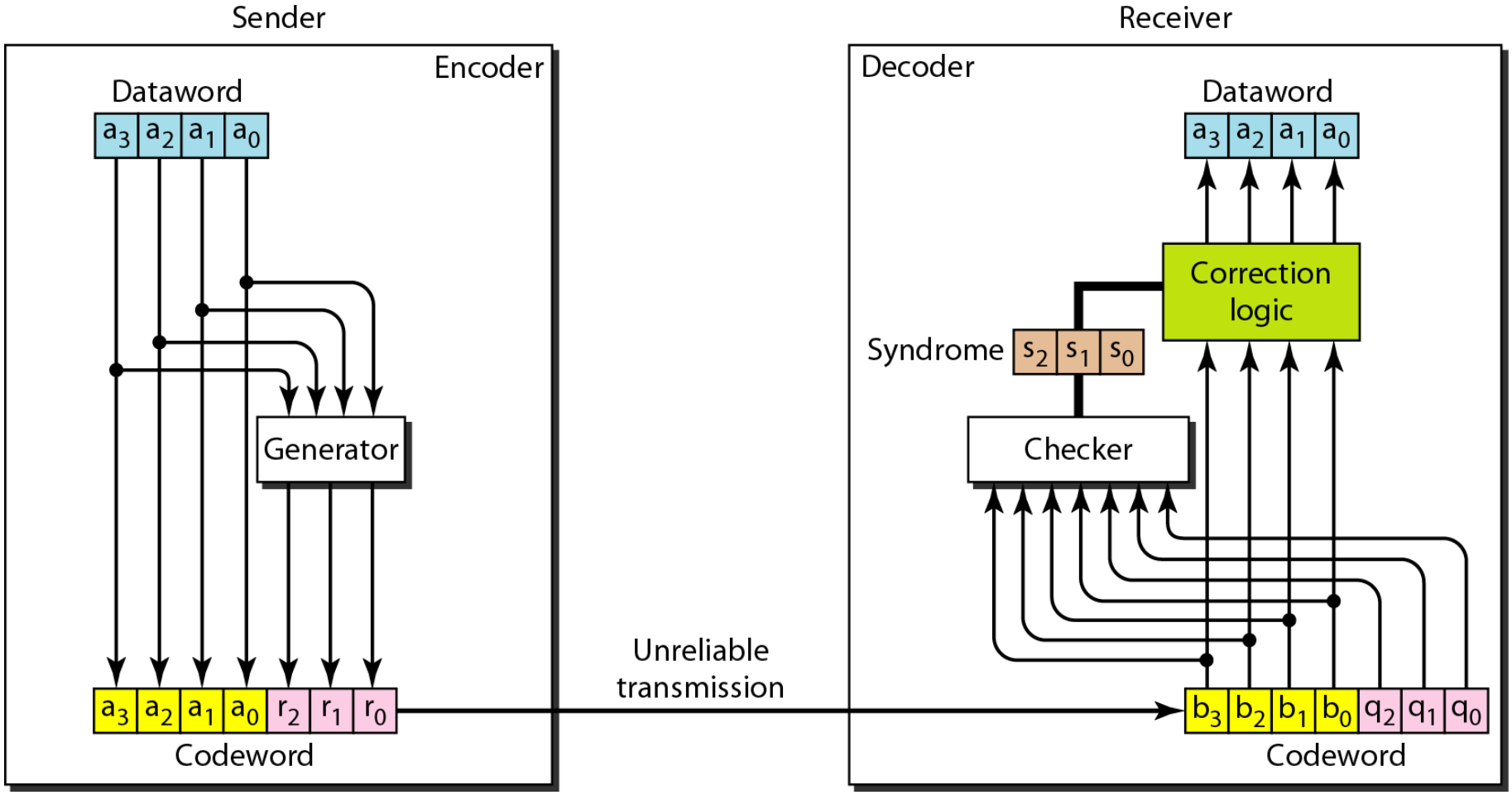


Table ***Logical decision made by the correction logic analyzer***

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

$r_0 = a_2 + a_3 + a_0$ modulo 2

$r_1 = a_3 + a_2 + a_1$ modulo 2

$r_2 = a_1 + a_0 + a_3$ modulo 2

$s_0 = b_2 + b_1 + b_0 + q_0$ modulo 2

$s_1 = b_3 + b_2 + b_1 + q_1$ modulo 2

$s_2 = b_1 + b_0 + b_3 + q_2$ modulo 2

Example 13

Let us trace the path of three datawords from the sender to the destination:

1. ***The dataword 0100 becomes the codeword 0100011. The codeword 0100011 is received. The syndrome is 000, the final dataword is 0100.***
2. ***The dataword 0111 becomes the codeword 0111001. The syndrome is 011. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.***
3. ***The dataword 1101 becomes the codeword 1101000. The syndrome is 101. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.***

Example 14

We need a dataword of at least 7 bits. Calculate values of k and n that satisfy this requirement.

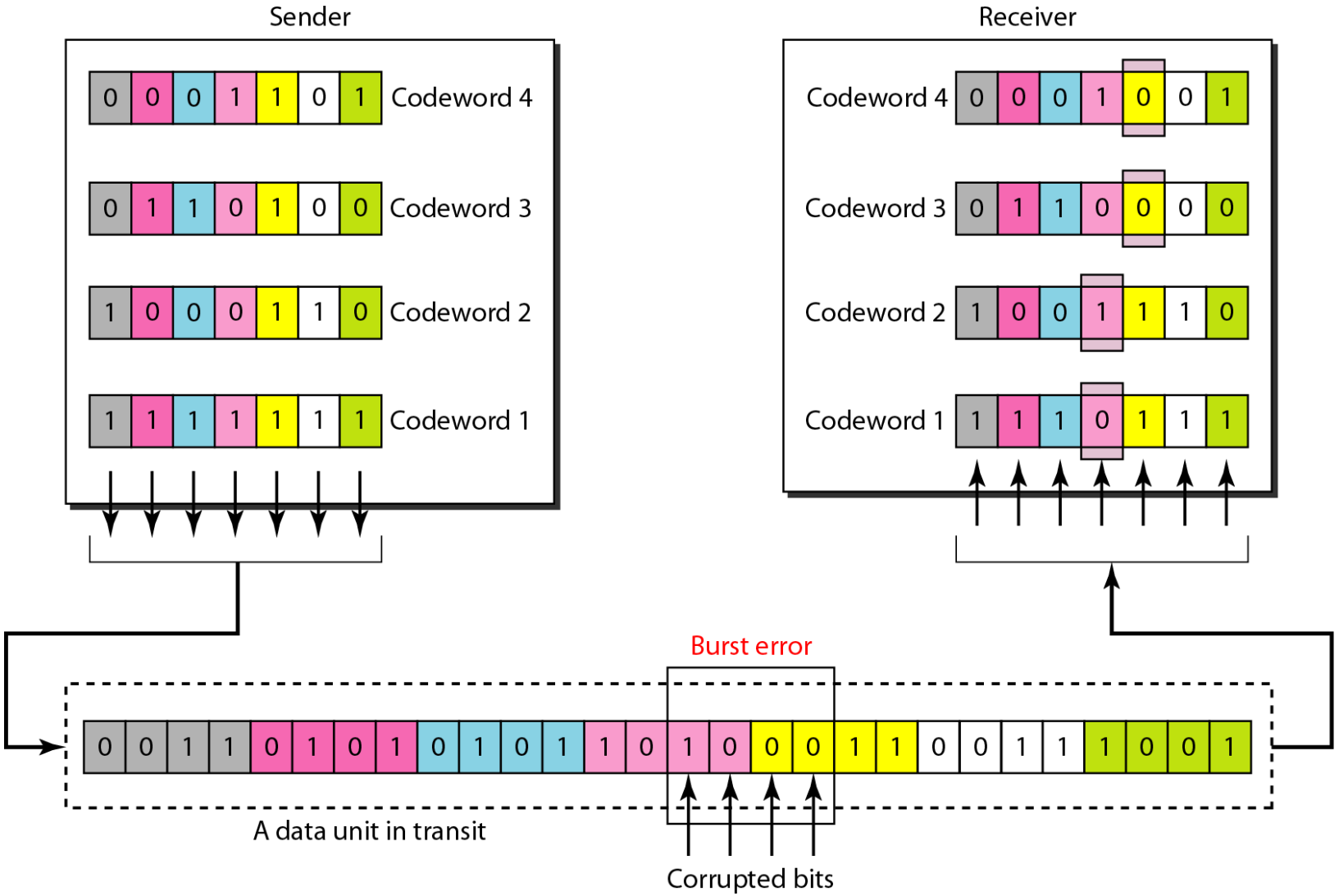
Solution

We need to make $k = n - m$ greater than or equal to 7, or $2m - 1 - m \geq 7$.

1. ***If we set $m = 3$, the result is $n = 23 - 1$ and $k = 7 - 3$, or 4, which is not acceptable.***
2. ***If we set $m = 4$, then $n = 24 - 1 = 15$ and $k = 15 - 4 = 11$, which satisfies the condition. So the code is***

C(15, 11)

Figure *Burst error correction using Hamming code*



CYCLIC CODES

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

Topics discussed in this section:

Cyclic Redundancy Check
Hardware Implementation
Polynomials
Cyclic Code Analysis
Advantages of Cyclic Codes
Other Cyclic Codes

Table *A CRC code with C(7, 4)*

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure *CRC encoder and decoder*

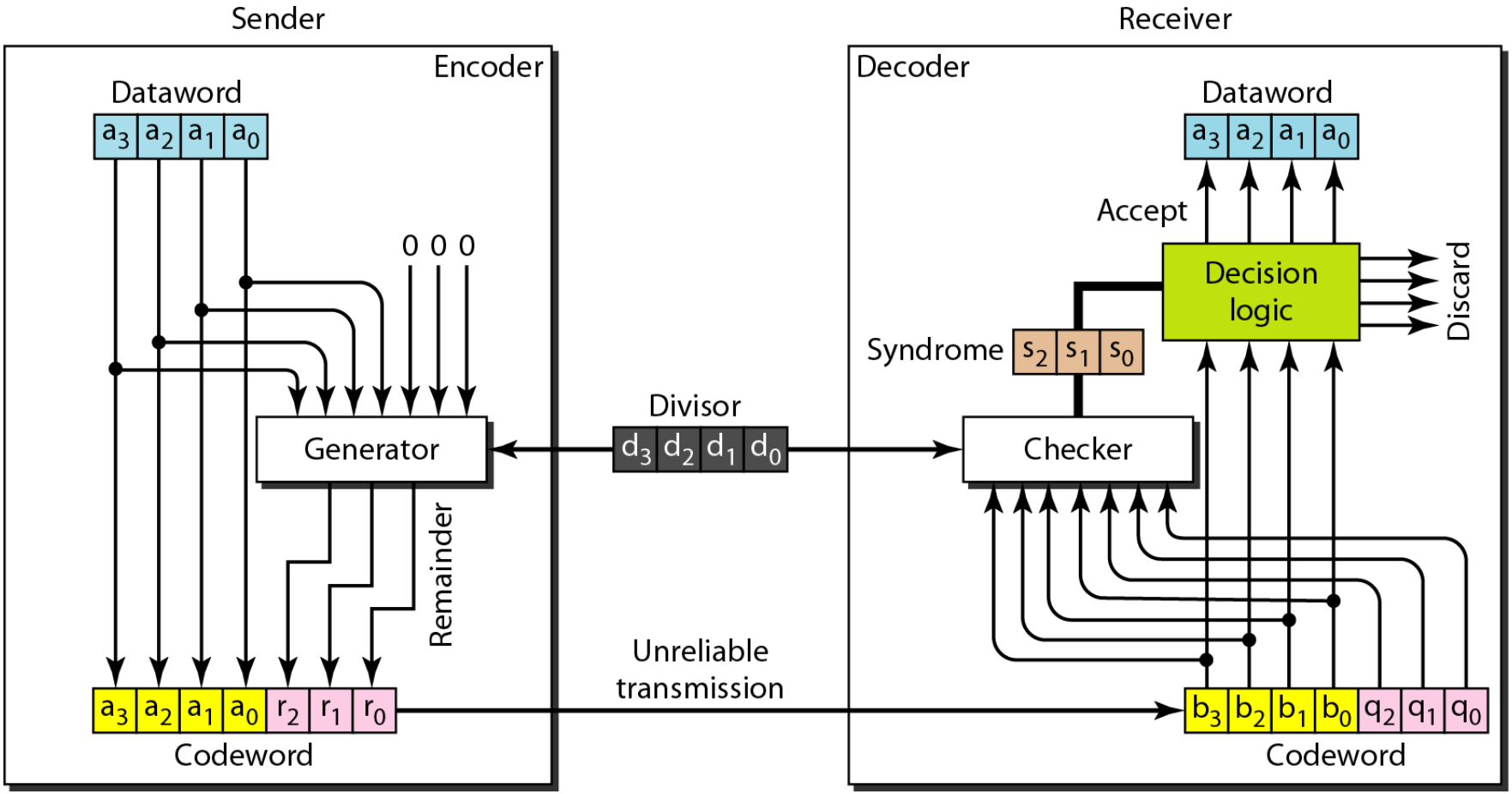


Figure *Division in CRC encoder*

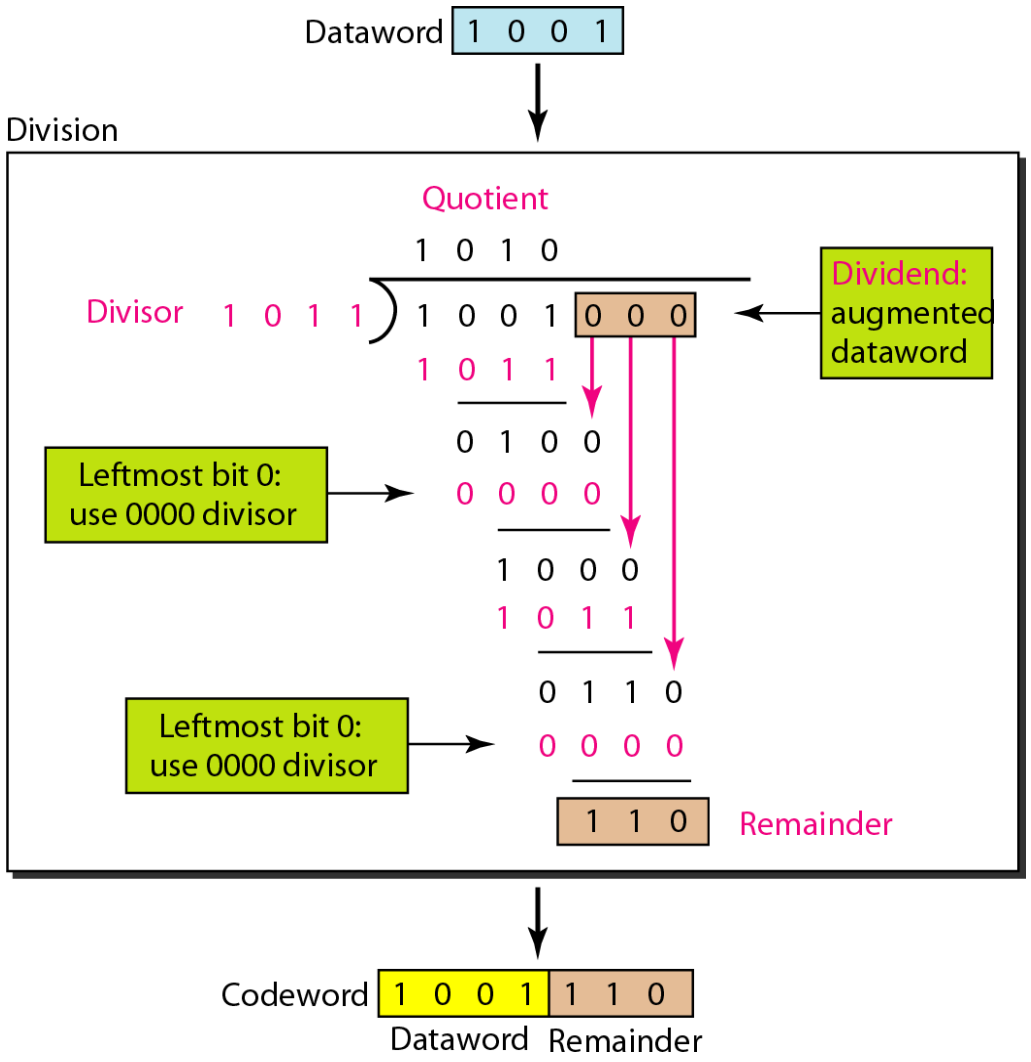


Figure *Division in the CRC decoder for two cases*

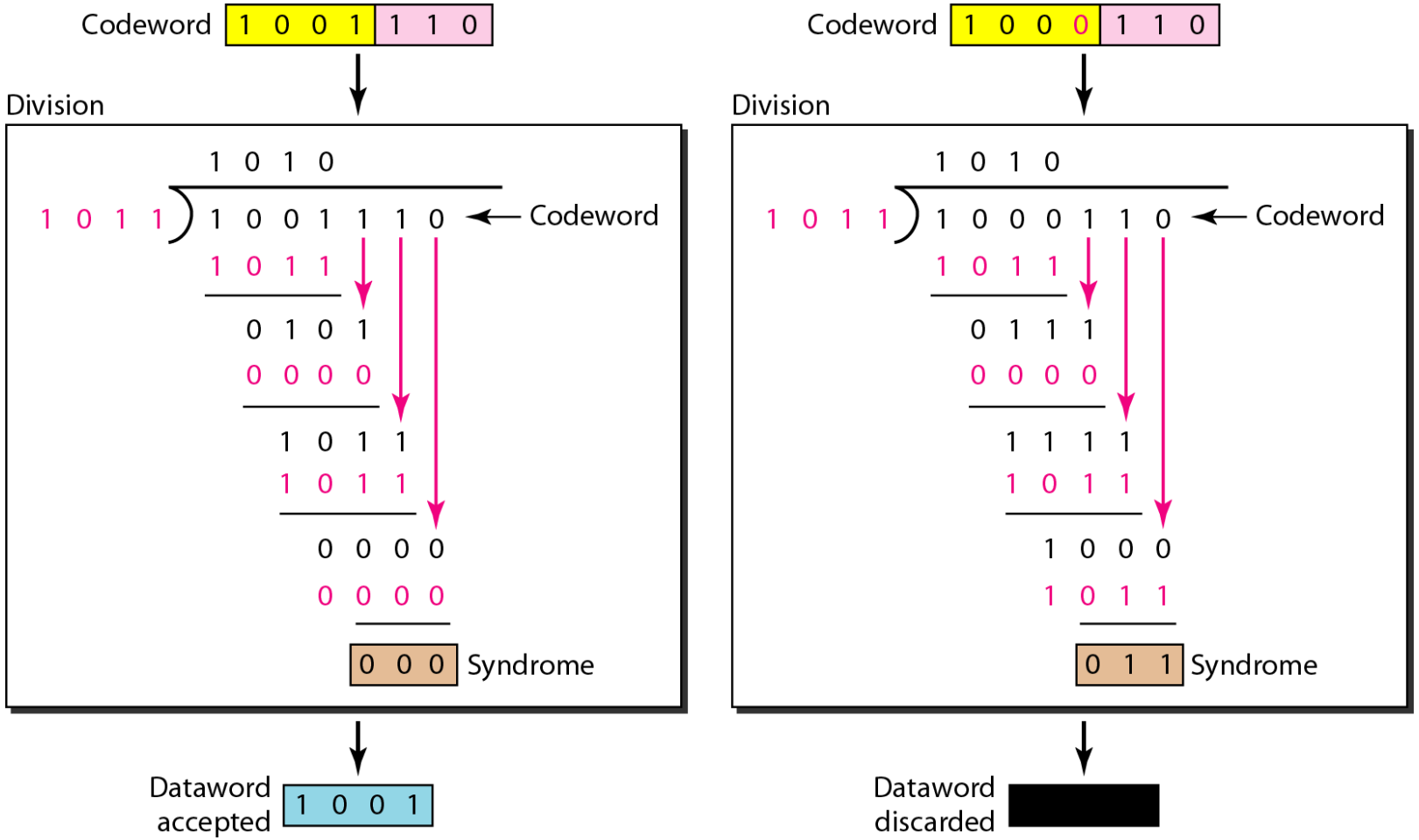


Figure ***Hardwired design of the divisor in CRC***

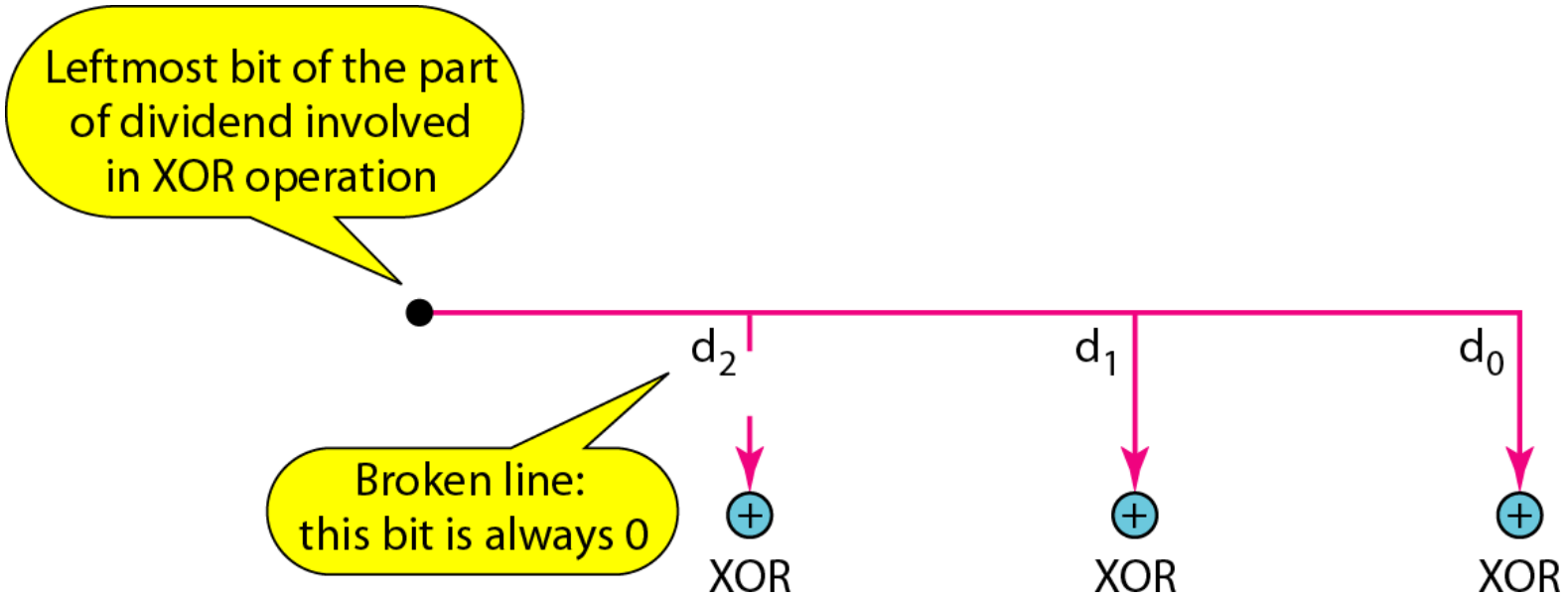


Figure *Simulation of division in CRC encoder*

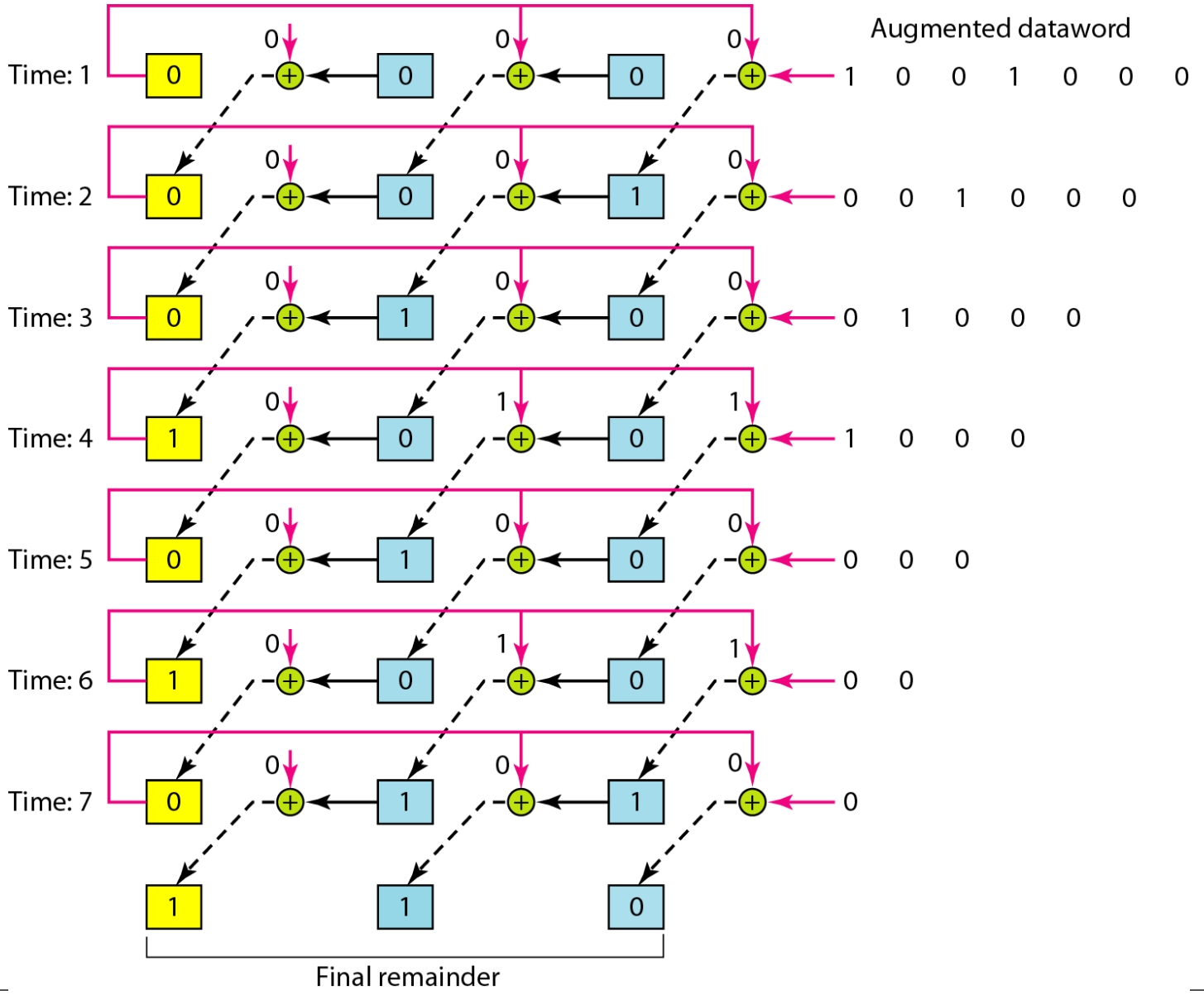


Figure ***The CRC encoder design using shift registers***

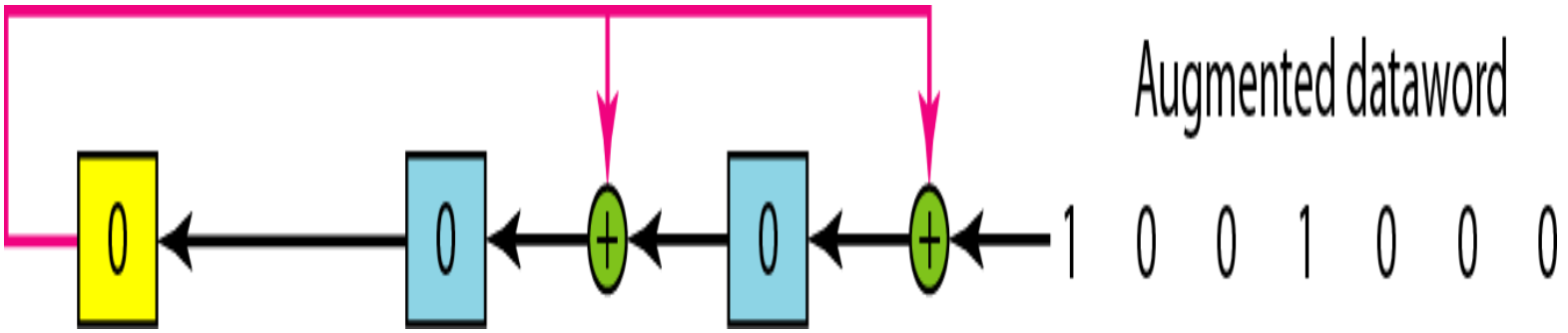


Figure ***General design of encoder and decoder of a CRC code***

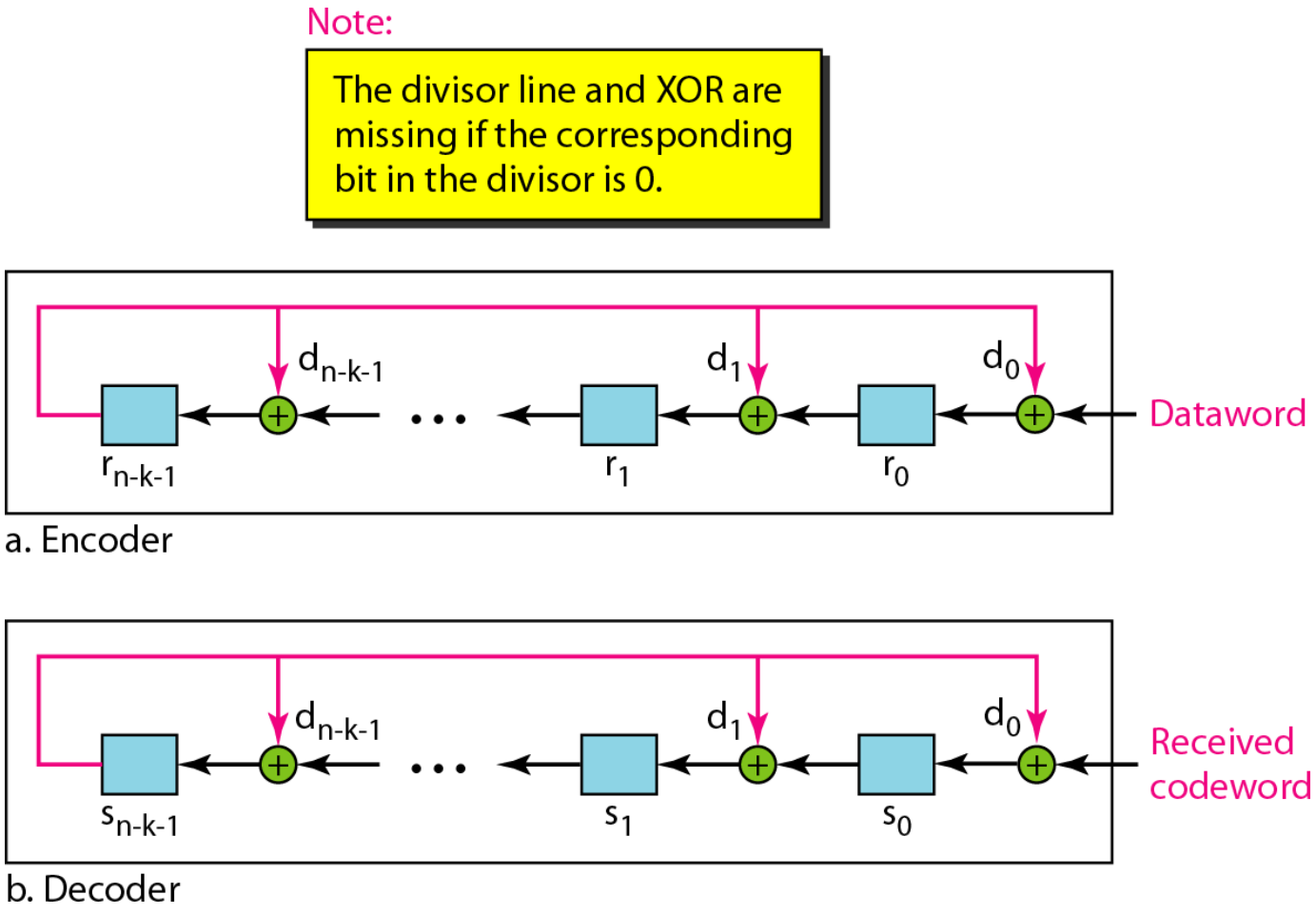
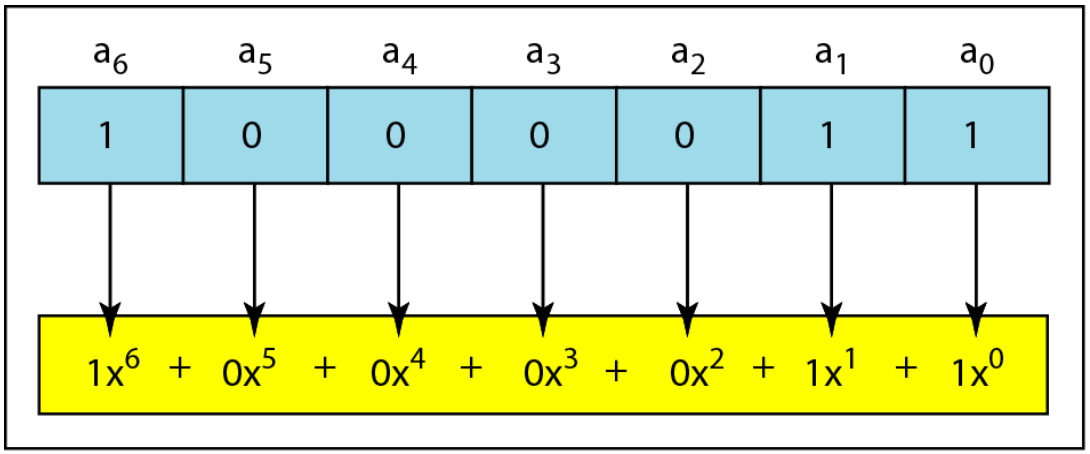
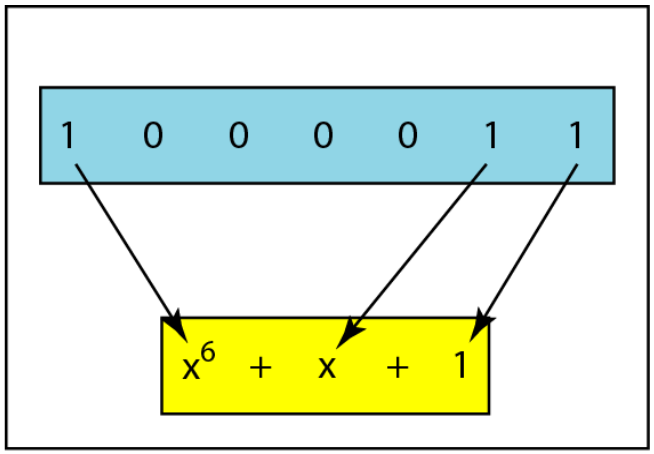


Figure *A polynomial to represent a binary word*

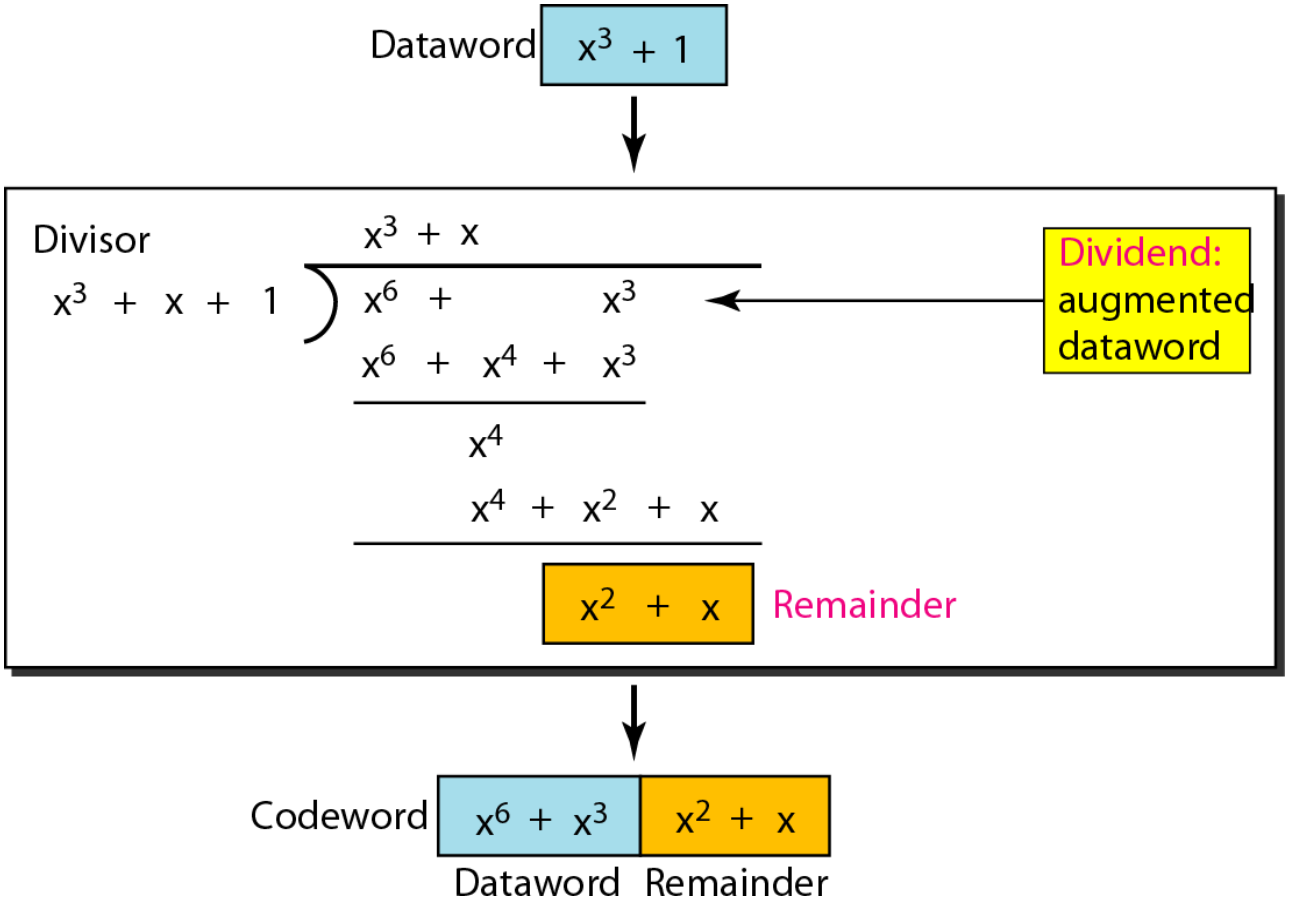


a. Binary pattern and polynomial



b. Short form

Figure *CRC division using polynomials*



Note

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

Note

In a cyclic code,

If $s(x) \neq 0$, one or more bits is corrupted.

If $s(x) = 0$, either

a. **No bit is corrupted. or**
b. **Some bits are corrupted, but the decoder failed to detect them.**

Note

In a cyclic code, those *e*(x) errors that are divisible by *g*(x) are not caught.

Note

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

Example 15

Which of the following g(x) values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?

a. $x + 1$ ***b.*** x^3 ***c.*** 1

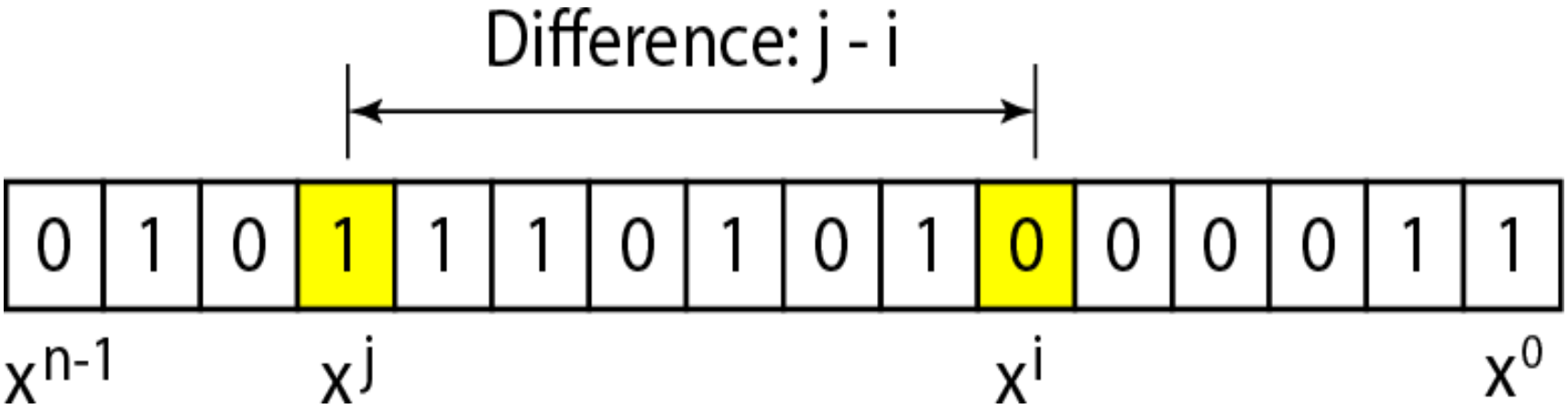
Solution

a. *No x^i can be divisible by $x + 1$. Any single-bit error can be caught.*

b. *If i is equal to or greater than 3, x^i is divisible by g(x). All single-bit errors in positions 1 to 3 are caught.*

c. *All values of i make x^i divisible by g(x). No single-bit error can be caught. This g(x) is useless.*

Figure ***Representation of two isolated single-bit errors using polynomials***



Note

If a generator cannot divide $x^t + 1$ (t between 0 and n – 1), then all isolated double errors can be detected.

Example 16

Find the status of the following generators related to two isolated, single-bit errors.

a. $x + 1$ *b.* $x^4 + 1$ *c.* $x^7 + x^6 + 1$ *d.* $x^{15} + x^{14} + 1$

Solution

a. This is a very poor choice for a generator. Any two errors next to each other cannot be detected.

b. This generator cannot detect two errors that are four positions apart.

c. This is a good choice for this purpose.

d. This polynomial cannot divide $x^t + 1$ if t is less than 32,768. A codeword with two isolated errors up to 32,768 bits apart can be detected by this generator.

Note

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

Note

- All burst errors with $L \leq r$ will be detected.
- All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.
- All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^{r}$.

Example 17

Find the suitability of the following generators in relation to burst errors of different lengths.

a. $x^6 + 1$ ***b.*** $x^{18} + x^7 + x + 1$ ***c.*** $x^{32} + x^{23} + x^7 + 1$

Solution

a. ***This generator can detect all burst errors with a length less than or equal to 6 bits; 3 out of 100 burst errors with length 7 will slip by; 16 out of 1000 burst errors of length 8 or more will slip by.***

Example 17 (continued)

b. ***This generator can detect all burst errors with a length less than or equal to 18 bits; 8 out of 1 million burst errors with length 19 will slip by; 4 out of 1 million burst errors of length 20 or more will slip by.***

c. ***This generator can detect all burst errors with a length less than or equal to 32 bits; 5 out of 10 billion burst errors with length 33 will slip by; 3 out of 10 billion burst errors of length 34 or more will slip by.***

Note

A good polynomial generator needs to have the following characteristics:

1. **It should have at least two terms.**
2. **The coefficient of the term x^0 should be 1.**
3. **It should not divide $x^t + 1$, for t between 2 and $n - 1$.**
4. **It should have the factor $x + 1$.**

Table ***Standard polynomials***

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Outline

Design Issues: Services to Network Layer,Framing,Error Control and Flow Control

Flow Control Protocols: Unrestricted Simplex, Stop and Wait, Sliding Window Protocol

Error Control: Parity Bits, Hamming Codes (11/12-bits) and CRC.

WAN Connectivity : PPP and HDLC

HDLC

High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms we discussed in this chapter.

Topics discussed in this section:

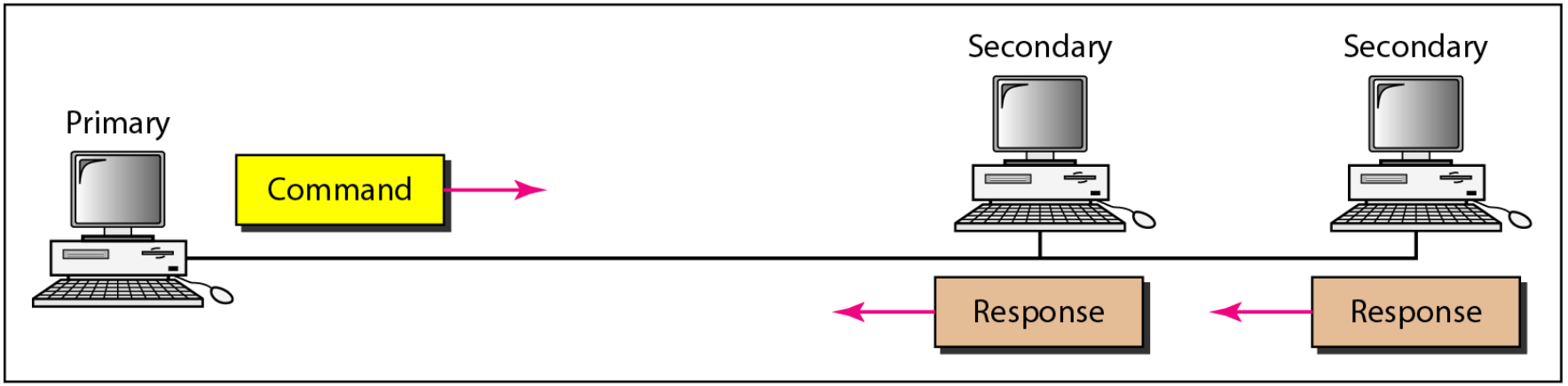
Configurations and Transfer Modes

Frames

Control Field

Figure *Normal response mode*

a. Point-to-point



b. Multipoint

Figure *Asynchronous balanced mode*

Figure *HDLC frames*

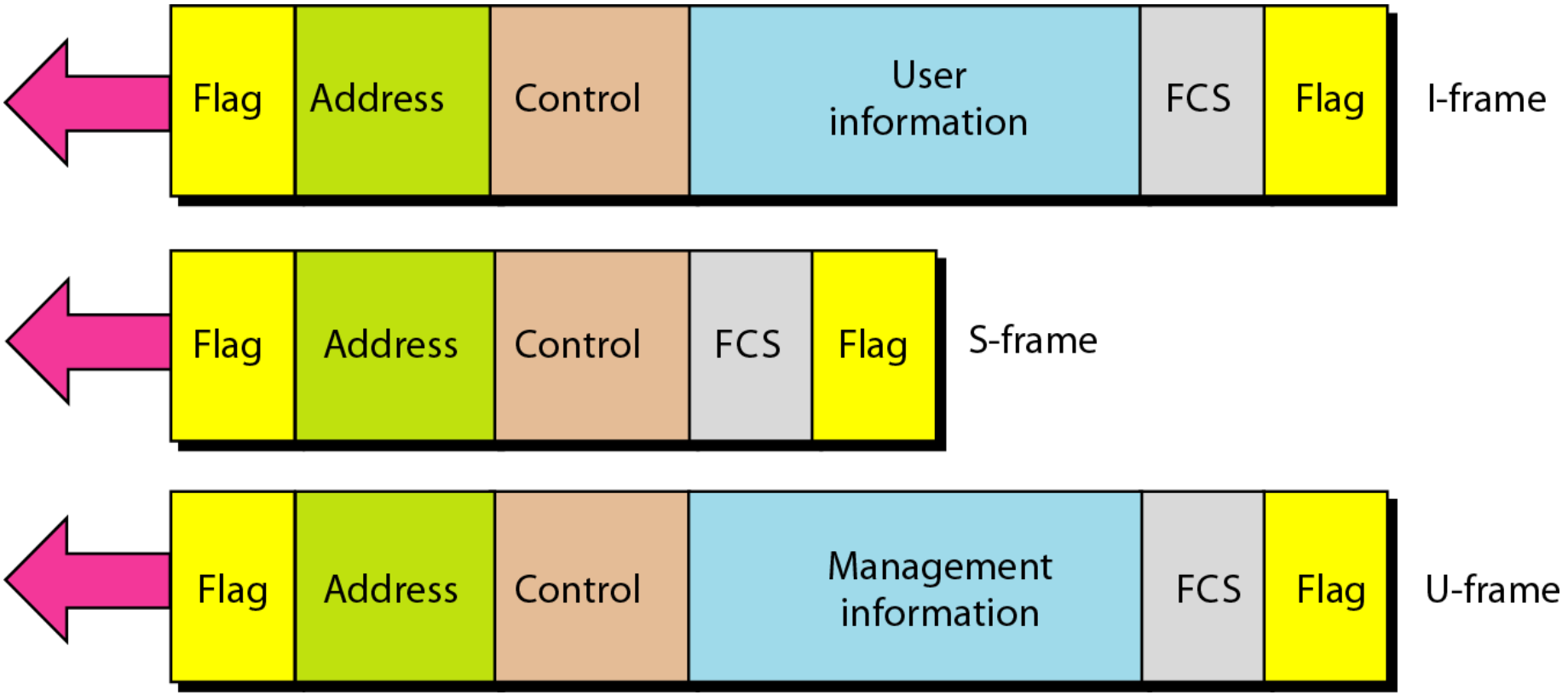


Figure ***Control field format for the different frame types***

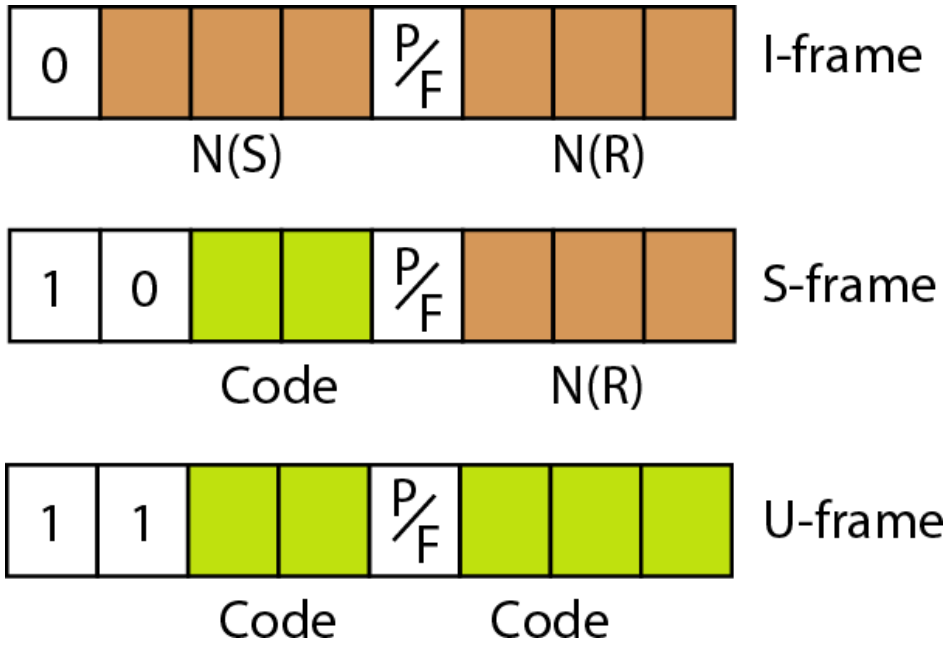


Table *U-frame control command and response*

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

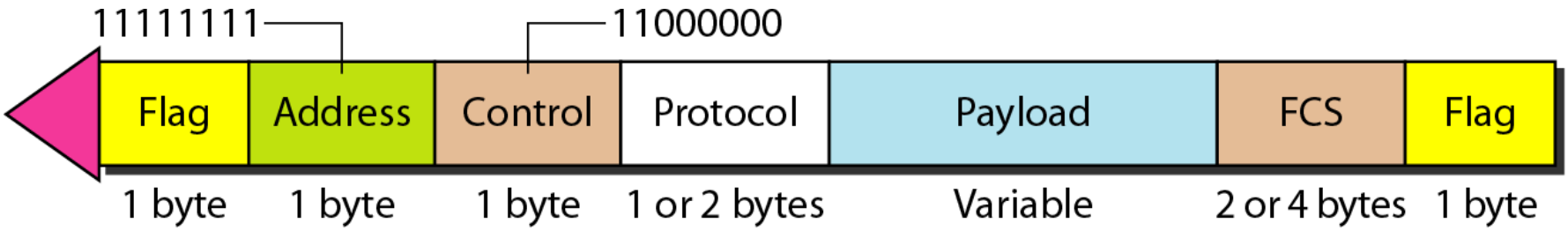
POINT-TO-POINT PROTOCOL

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). PPP is a byte-oriented protocol.

Topics discussed in this section:

Framing
Transition Phases
Multiplexing
Multilink PPP

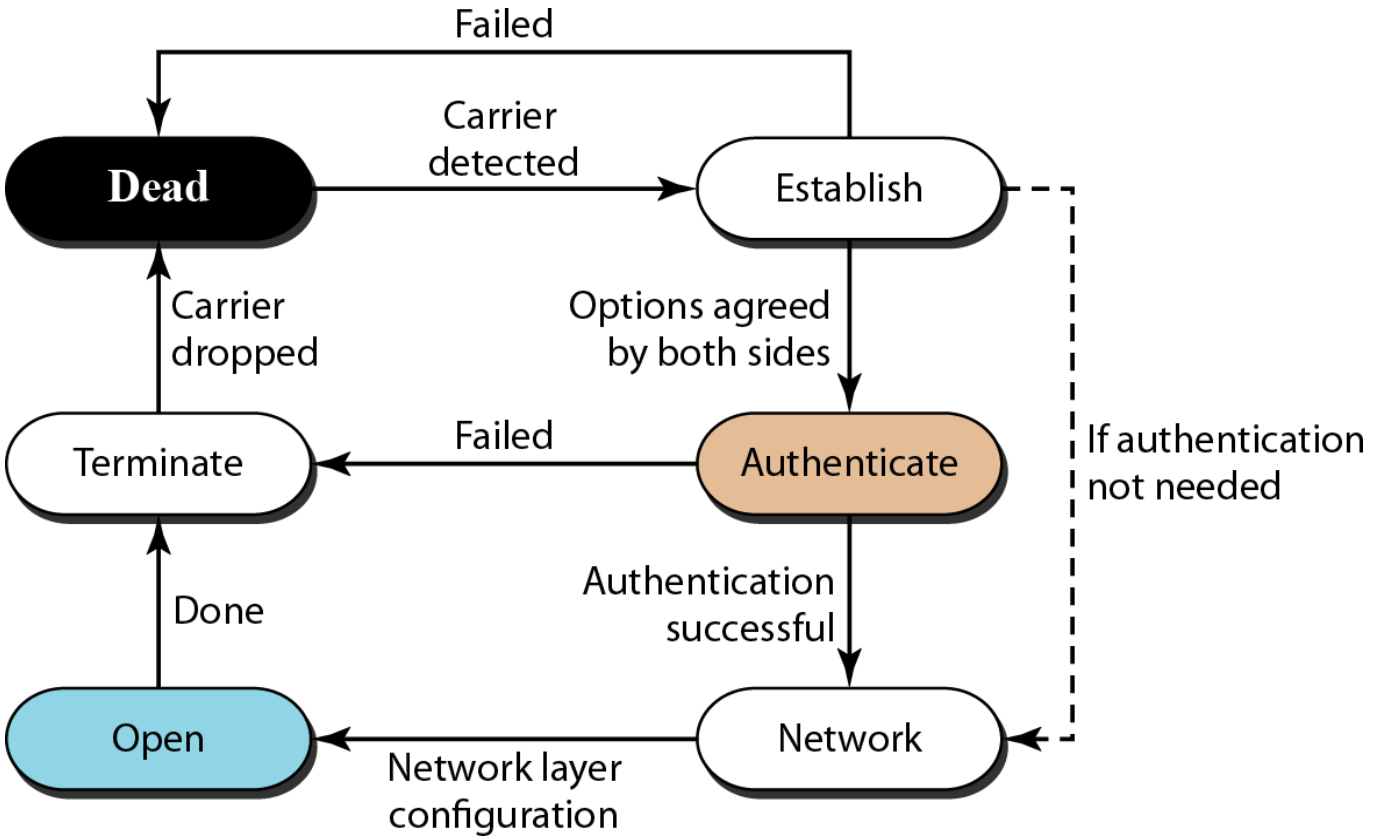
Figure *PPP frame format*



Note

PPP is a byte-oriented protocol using byte stuffing with the escape byte 01111101.

Figure *Transition phases*



References

Websites:

- *www.csie.cgu.edu.tw/~jhchen/course/CommSys/ch10.ppt*
- *www.csie.cgu.edu.tw/~jhchen/course/CommSys/ch11.ppt*

Videos(Gate Lectures by Ravindrababu Ravula)

- https://www.youtube.com/watch?v=1DvXxrzq0Wg

Text Books:

- Andrew S.Tenenbaum, "Computer Networks",5th Edition, PHI, ISBN 81-203-2175-8.
- Fourauzan B., "Data Communications and Networking", 5th Edition, Tata McGraw- Hill, Publications, 2006