Module-2

Error Detection and Correction, Data link control, Channelization

Data Link Layer: Error Detection and Correction: Introduction, Block Coding, Cyclic Codes, Check Sum

Data link control: DLC Services: Framing, Flow Control, Error Control

Channelization: FDMA, TDMA, CDMA

Textbook: Ch. 10.1-10.4, 11.1 -11.2, 12.3

Data Link Layer

Data can be corrupted during transmission.

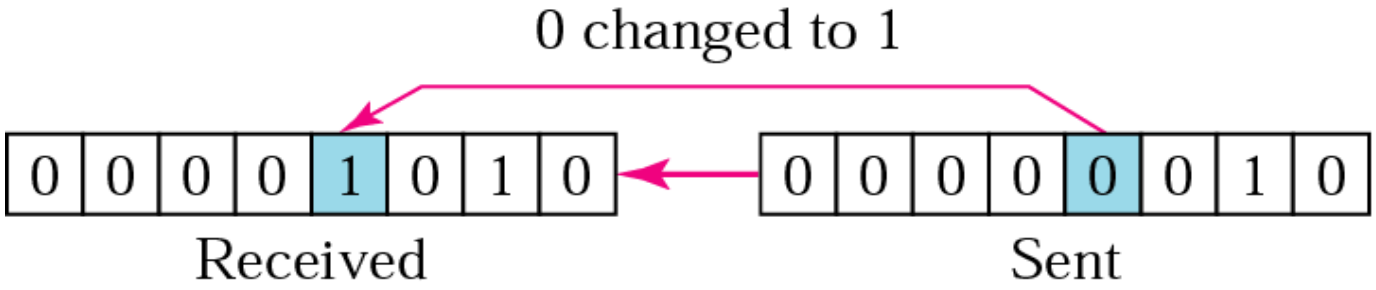
For reliable communication, errors must be detected and corrected.

Types of Error

Single-Bit Error

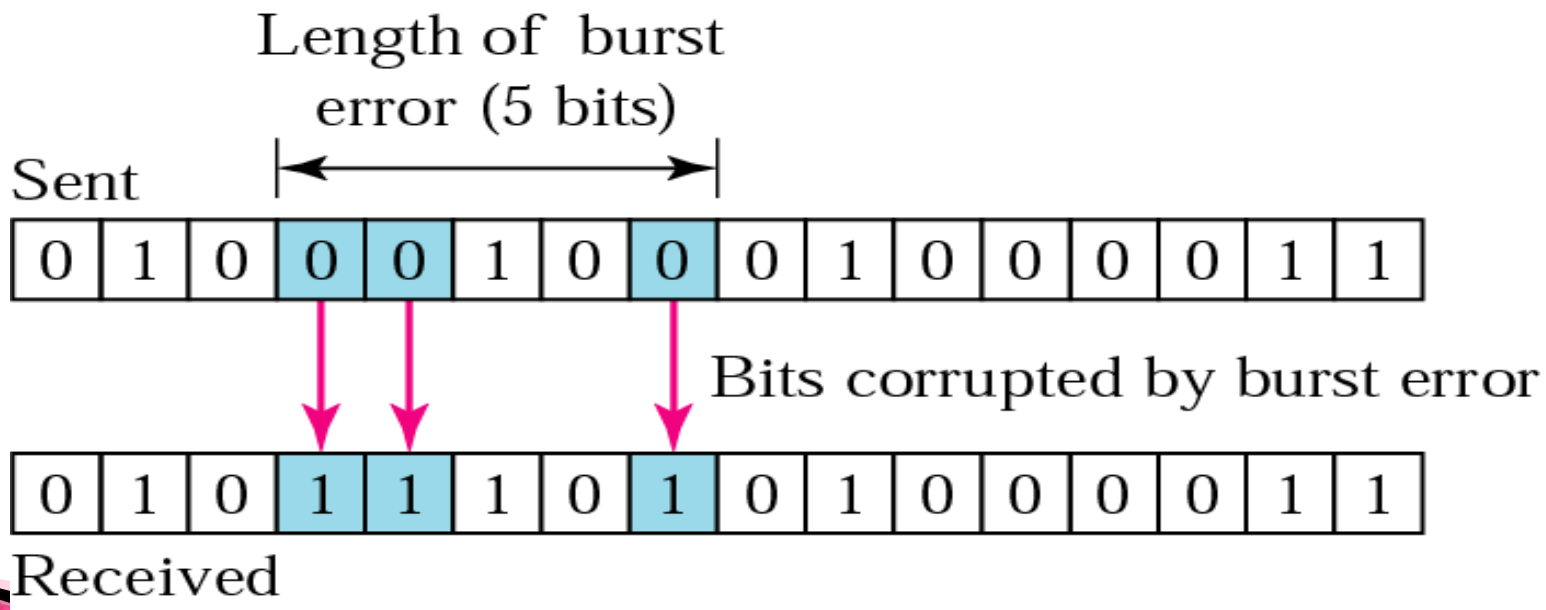
Burst Error

In a **single-bit error**, only one bit of a given data unit is changed from 1 to 0 or from 0 to 1.



A **burst error** means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

The length of the burst is measured from the first corrupted bit to the last corrupted bit



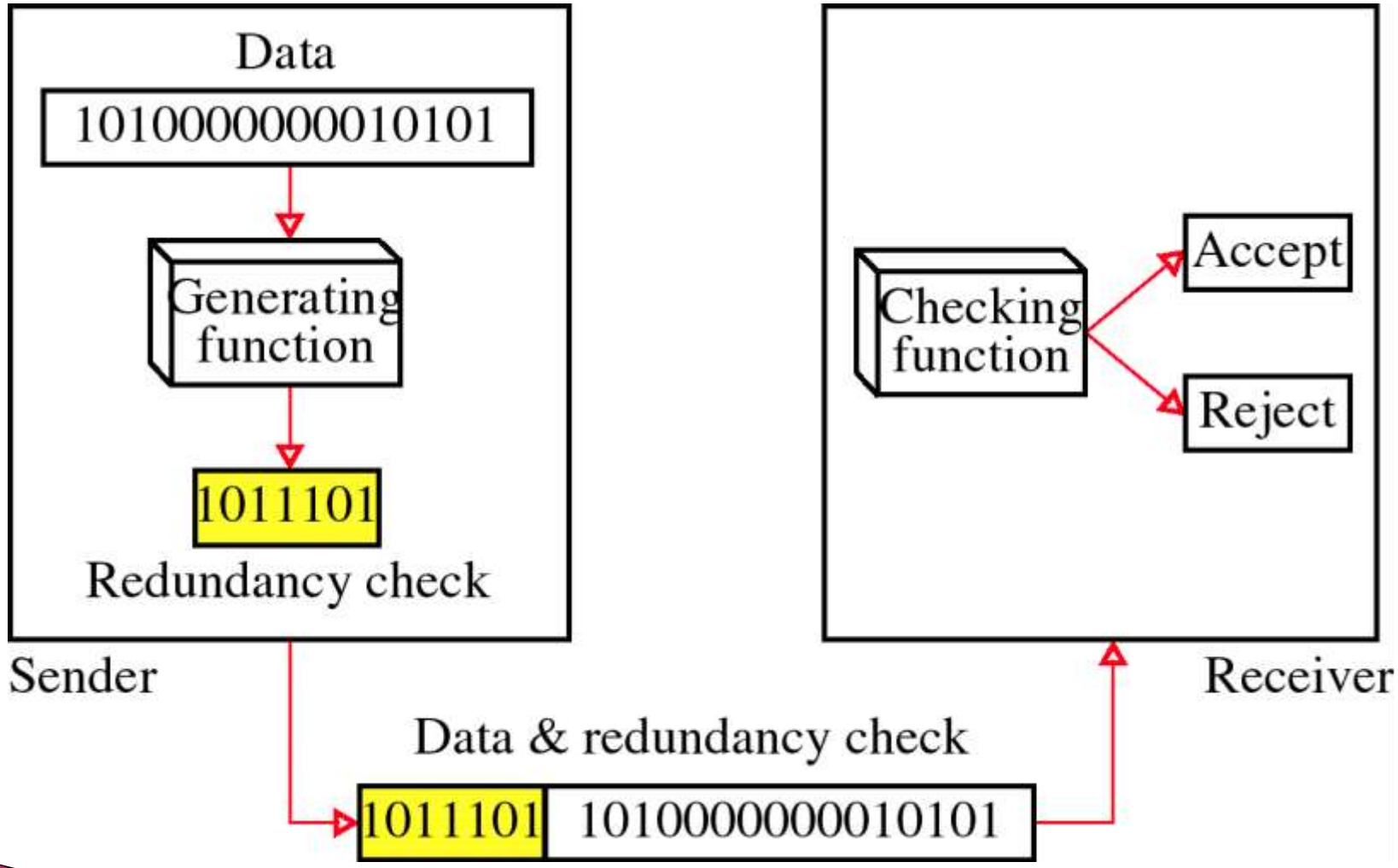
Redundancy

To detect or correct errors we need to send **redundant bits** with data.

These redundant bits are added by sender and removed by receiver.

Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.

Redundancy


Data
1010000000010101
Generating function
1011101
Redundancy check
Sender
Checking function
Accept
Reject
Receiver
Data & redundancy check
1011101
1010000000010101

Detection Vs Correction

Error Detection: Process of observing whether error has occurred or not. We are not even interested in the number of errors

Correction: The correction of errors is more difficult than the detection. Finding exact number of bits that are corrupted and their location. The number of errors and size of the message are important

Forward Error correction And Retransmission

There are two methods of error correction

Forward error correction: In this method, the sender adds redundant bits to the original data before transmission. The receiver uses these redundant bits to detect and correct errors without needing retransmission. This ensures continuous communication but increases overhead due to additional bits.

Correction by retransmission: In this method, the receiver checks the received data for errors. If an error is detected, the receiver requests the sender to retransmit the message. This reduces redundancy but may introduce delays due to retransmission.

Coding

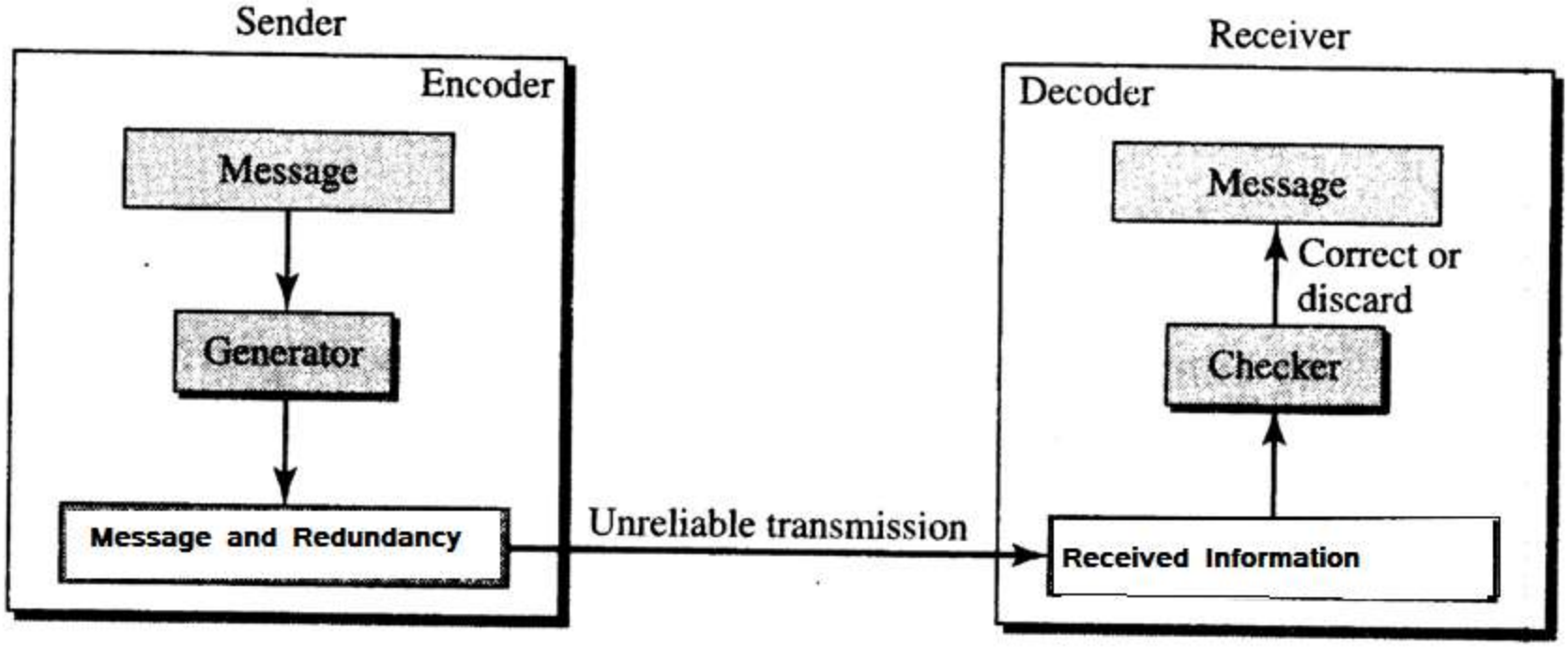
Redundancy is achieved through various coding scheme.

The sender adds redundant bits through a process that creates a relationship b/w the redundant bits and the actual data bits

The receiver checks the relationship b/w two sets of bits to detect or correct the errors

2 broad categories of coding are :Block coding and **Convolution coding**

Structure of encoder and decoder



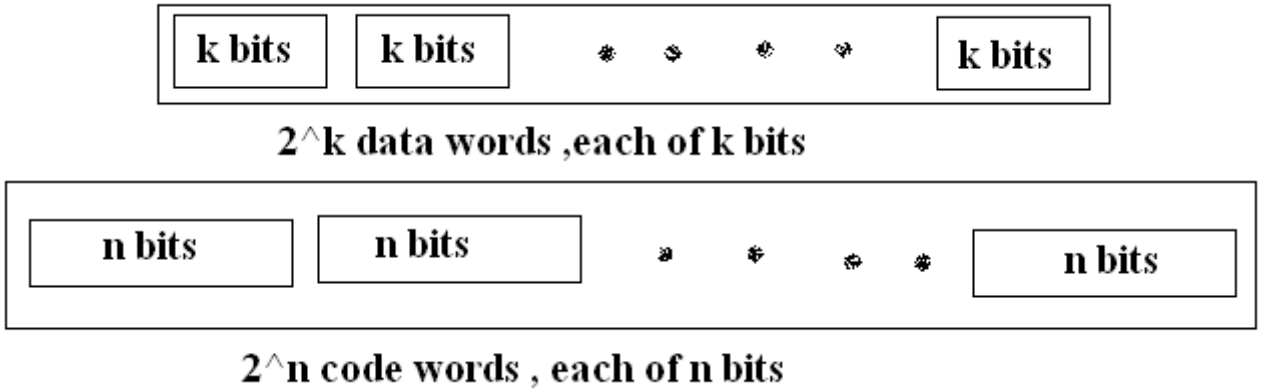
Modular Arithmetic

- Only limited range of integers are used.
- Upper limit N is used-Modulus.
- Allowed integers are 0 to N-1
- This operation is similar to XOR operation.

	1	0	1	0
	1	1	0	0
XOR	**0**	**1**	**1**	**0**

Block Coding

- Message is divided into blocks, each of **k** bits called **data words.**
- **r** –redundant bits are added to each block to make length **n=k+r**
- The resulting **n**-bit blocks are called **code words.**

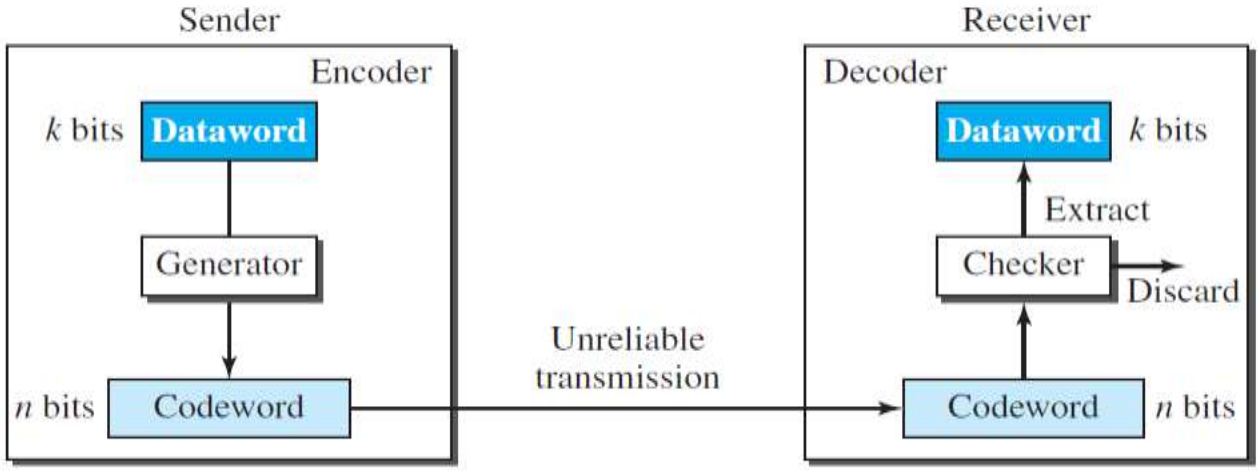


Error Detection

Conditions:

- The receiver has list of valid code words
- The original code word has changed to an invalid code words.
- In detection , the receiver needs to know only that the received code word is invalid

Figure 10.2 *Process of error detection in block coding*



Example

Data words	Code words
00	000
01	011
10	101
11	110

Assume sender encodes the data words 01 as 011 and sends it to receiver.

- Consider the following cases

1. The receiver receives 011. it is valid codeword. The receiver extracts the data word 01 from it

2. The code word is corrupted during transmission and 111 is received. This is not a valid code word and it is discarded

3. The code word is corrupted during transmission and 000 is received. This is a valid code word. The receiver incorrectly extracts the data word 00 (2 corrupted bits have made the error undetectable)

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

Error Correction

- The receiver needs to find the original code word sent.
- We need more redundant bits for error correction than for error detection

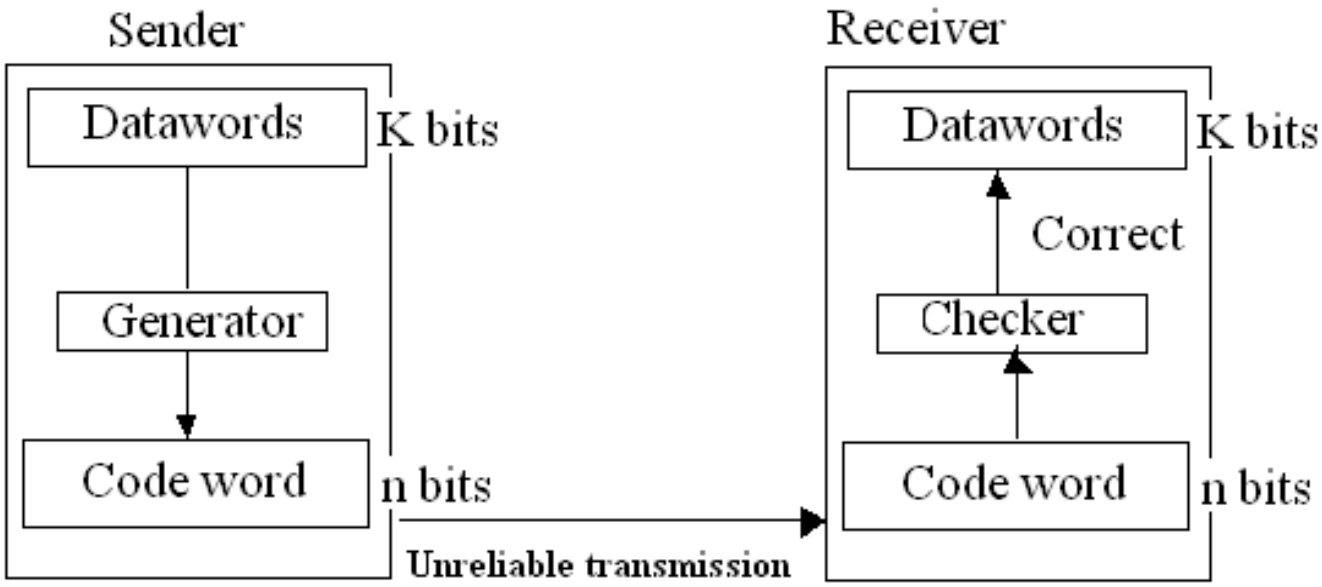


Table for error correction

Data words	Codeword
00	00000
01	01011
10	10101
11	11110

The data word is 01. the sender create the code word 01011. the code word is corrupted during transmission and 01001 is received.

First, the receiver finds that the received code word is not in the table. This means error has occurred.

The receiver uses the following strategy to guess the correct data word

1. Compare the received code word with the first code word in the table , the receiver decides that the first code word is not the one that was sent because there are two different bits

2. By the same reasoning, the original code word can not be the third or fourth in the table

3. The original code word must be the second one in the table because this is the only one that differs from the received code word by 1 bit. The receiver replaces 01001 with 01011 and consult the table to find the data word 01

Hamming Distance

- Hamming distance between two words is the number of difference between corresponding bits.
- It can easily be found by applying XOR operation on the two words and count the number of 1s in the result.
- Example: d(000,011) is 2

Minimum Hamming Distance

- It is the smallest hamming distance between all possible pairs in a set of words.

  Find minimum hamming distance between code words:

  00 **000**

  01 **011**

  10 **101**

  11 **110**

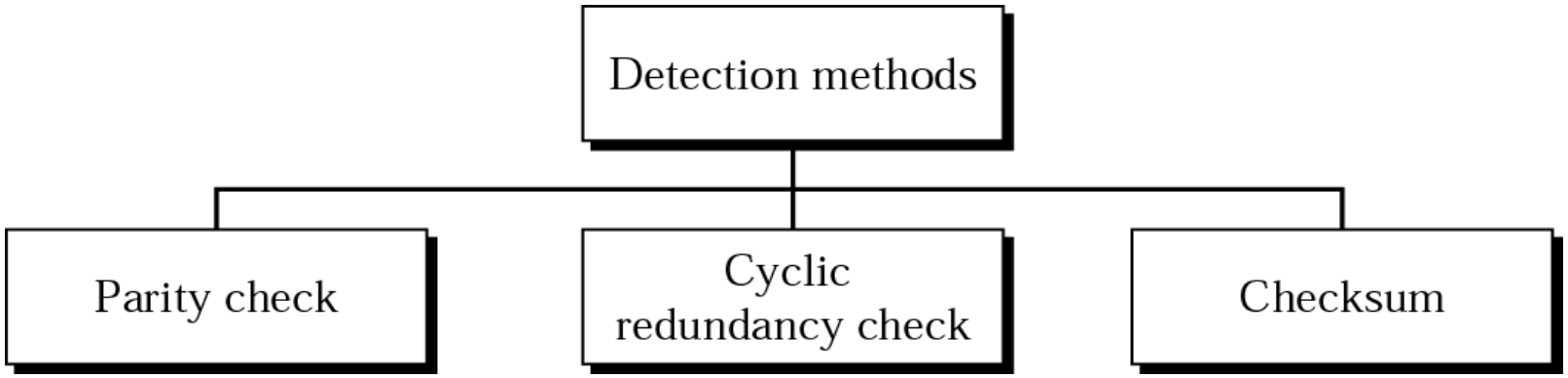
d(000,011)=2; d(000,011)=2;

d(000,101)=2; d(000,110)=2

d(011,110)=2; d(101,110)=2

d_{min} is **2**

10.4 *Detection methods*



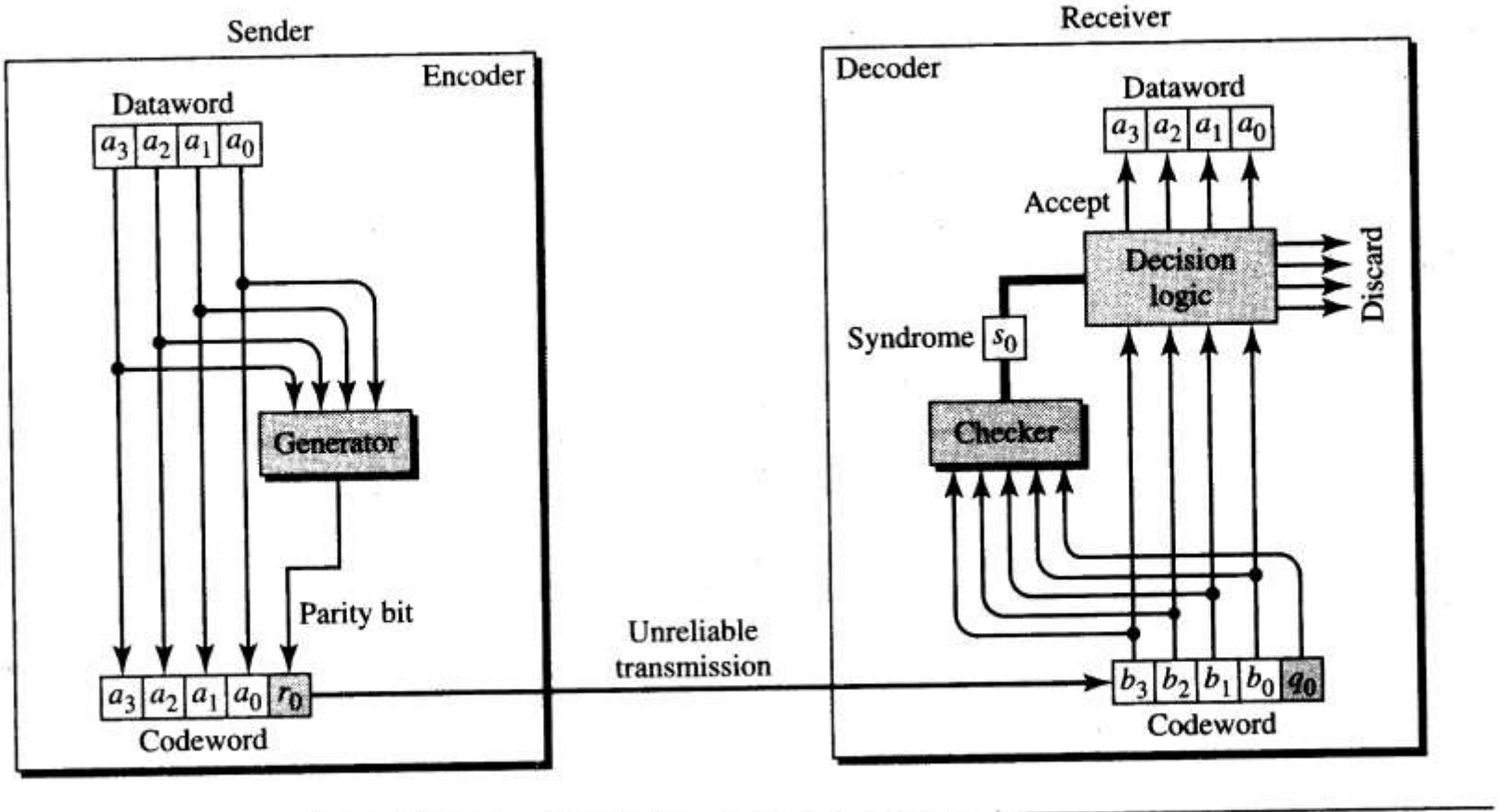
Simple Parity Check Code

- In this scheme , a k-bit data word is changed to an n-bit code word where n=k+1.
- This extra bit is called **parity** bit.
- The parity bit selected to make the total number of 1‘s in the codeword even.
- A simple parity check code is a single bit error detecting code in which n=k+1 with $d_{min}=2$

Simple parity check code –Encoder

- The encoder uses a generator that takes a copy of a 4-bit data word(a0,a1,a2,a3). It generates a parity bit r0
- Data word bits and the parity bit creates 5-bit codeword
- The parity bit that is added makes the number of 1s in the codeword even
- r0=a3+a2+a1+a1 **Modulo** 2

Encoder and Decoder for simple parity check code



Simple parity check code –decoder

- The receiver receives a 5-bit word
- The checker performs addition on all 5 bits received. The result is called **syndrome-**just 1 bit.
- The syndrome is 0 when the number of 1s in the received codeword is even ;otherwise it is 1.
- s0=b3+b2+b1+b0+q0 **modulo** 2

Assume the sender sends the dataword 1011. the code word created from this data word is 10111, which is sent to the receiver.

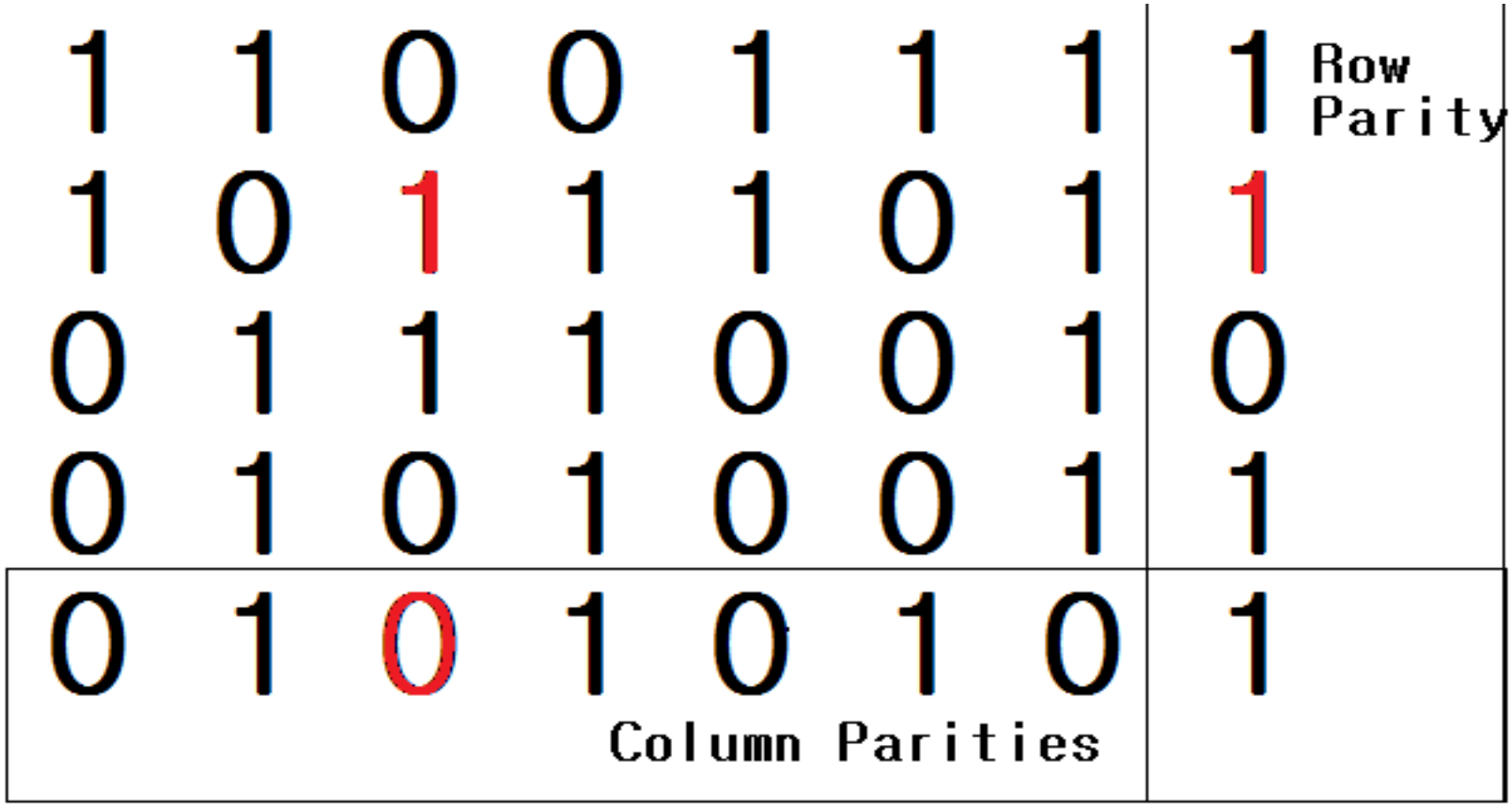
1. No error occurs, the received code word is 10111. the syndrome is 0. the data word is created
2. One single bit error changes a1. the received code word is 10011. the syndrome is 1. no data word is created
3. One single bit error changes r0. the received code word is 10110. the syndrome is 1. no data word is created
4. An error changes **r0** and a second error changes **a3**. the received code word is 00110. the syndrome is 0. the data word 0011 is created at the receiver
5. Three bits a3, a2, a1 are changed by errors. The received code word is 01011. the syndrome is 1. the data word not created.

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

Calculation of row and column parities

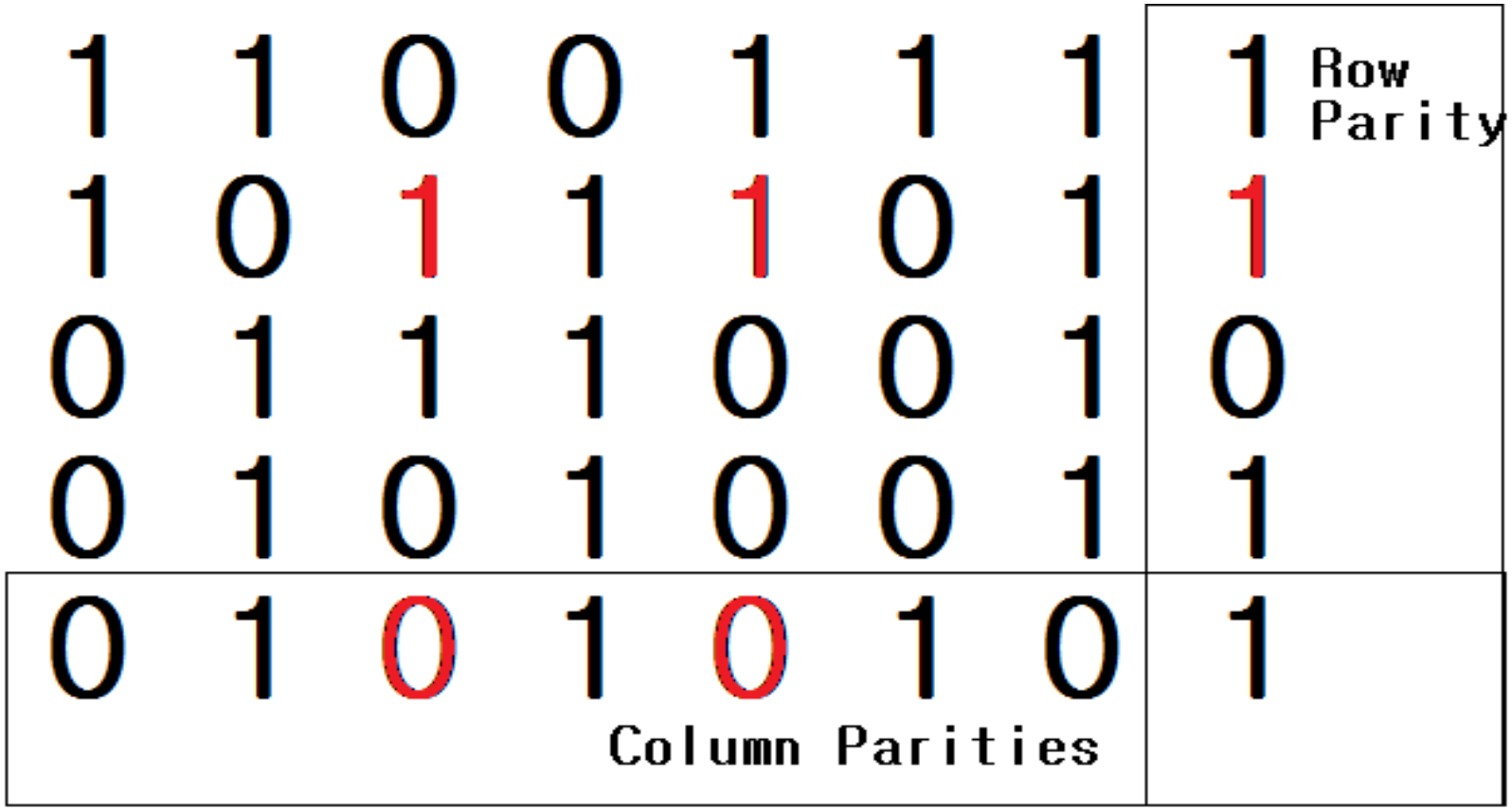
1	1	0	0	1	1	1	1 Row Parity
1	0	1	1	1	0	1	1
0	1	1	1	0	0	1	0
0	1	0	1	0	0	1	1
0	1	0	1	0	1	0	1
				Column Parityies			

Single bit error



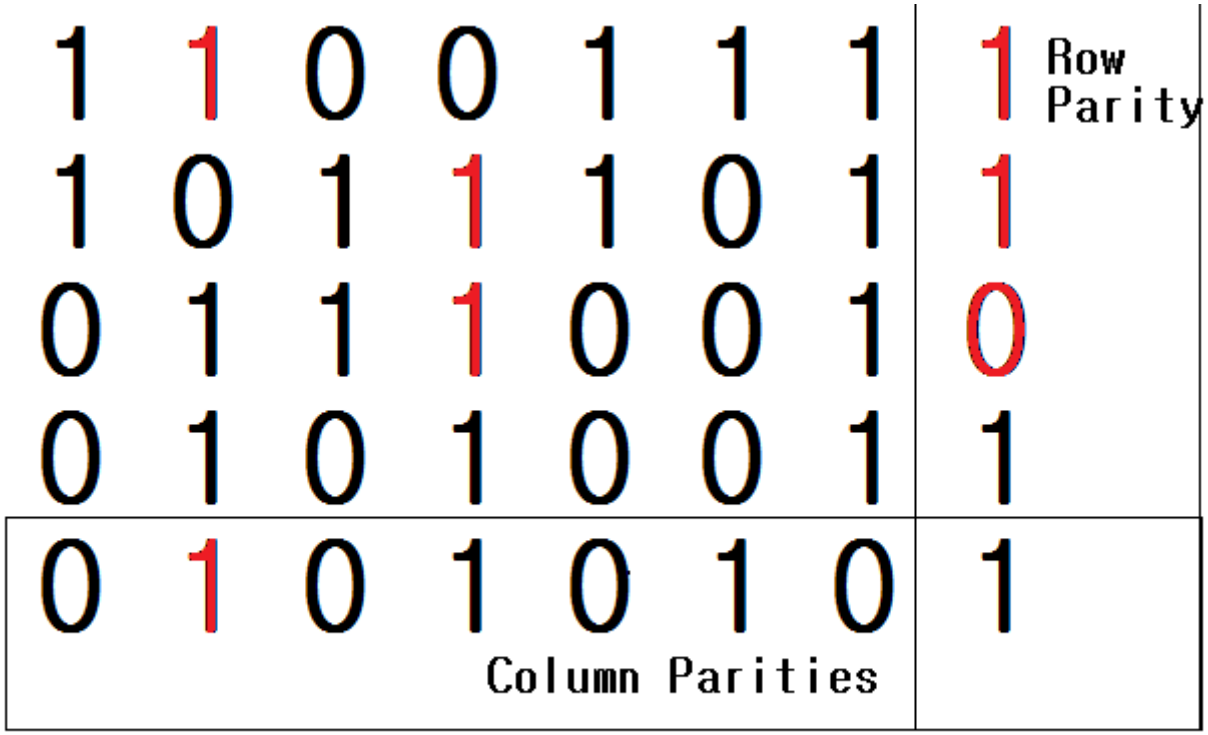
One error affects two parities

Two errors affects two parities



Two error affects two parities

Three errors



Three error affects Four parities

Four Errors

1	1	0	0	1	1	1	1 Row Parity
1	0	1	1	1	0	1	1
0	1	1	1	0	0	1	0
0	1	0	1	0	0	1	1
0	1	0	1	0	1	0	1
			Column Parities				

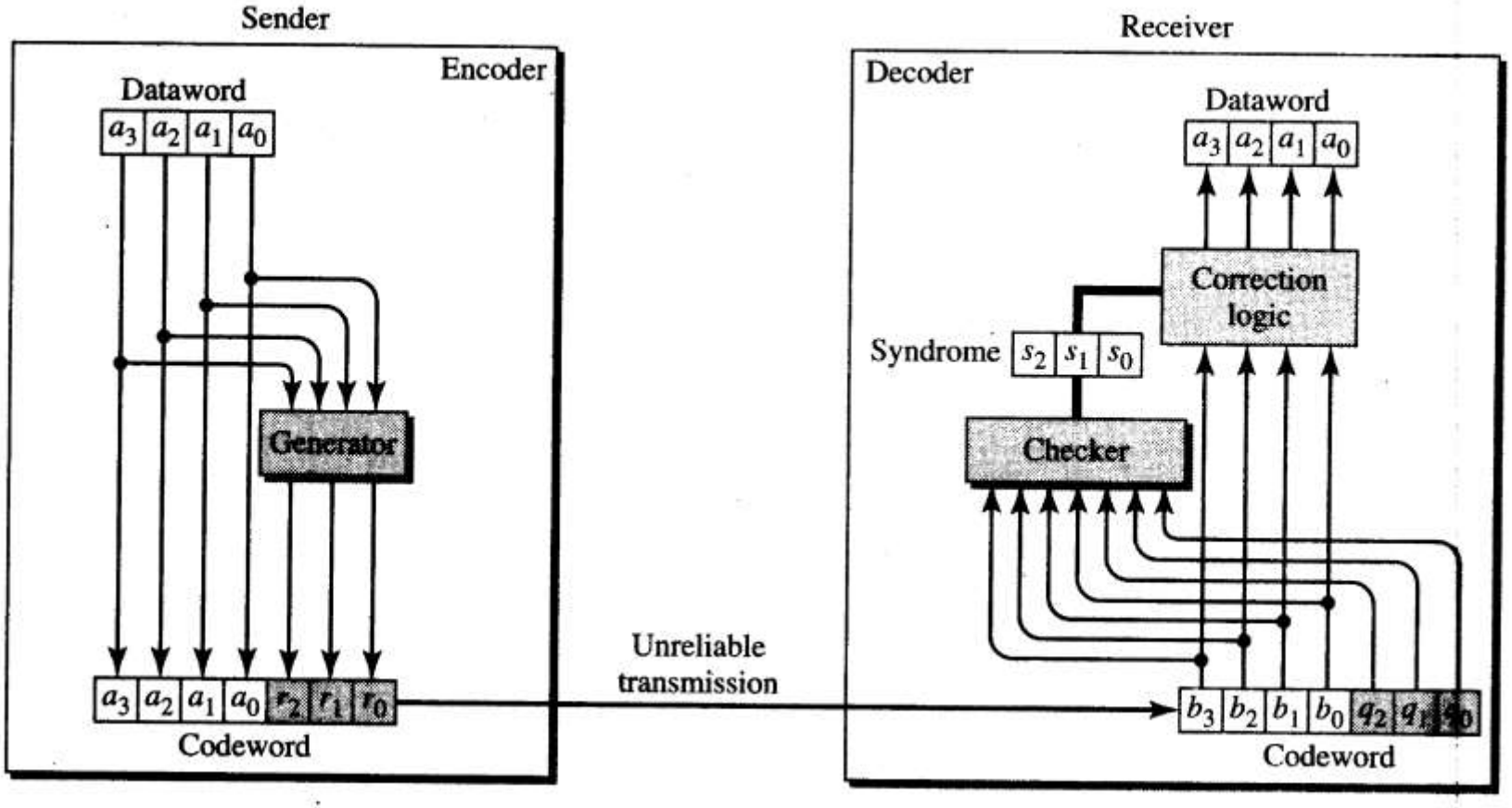
Four errors cannot be detected

Hamming Codes

- Minimum distance is 3. Expressed as C(n,k); where $n=2^m-1$
- Data words : a3,a2,a1,a0 and Code words : a3,a2,a1,a0,r2,r1,ro
- r0=a2+a1+a0; r1=a3+a2+a1; r3=a1+a0+a3
- Checker in the decoder creates a 3-bit syndrome (s2s1s0); where s0=b2+b1+b0+q0;s1=b3+b2+b1+q1;s3=b1+b0+b3+q2

Syndrome	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Error	**None**	**q0**	**q1**	**b2**	**q2**	**b0**	**b3**	**b1**

Structure of Encoder and Decoder Hamming code



- r0=a2+a1+a0 modulo-2
- r1=a3+a2+a1 modulo-2
- r2=a1+a0+a3 modulo-2

- s0=b2+b1+b0+q0 modulo-2
- s1=b3+b2+b1+q1 modulo-2
- s2=b1+b0+b3+q2 modulo-2

Let us trace the path of three data word from the sender to the destination

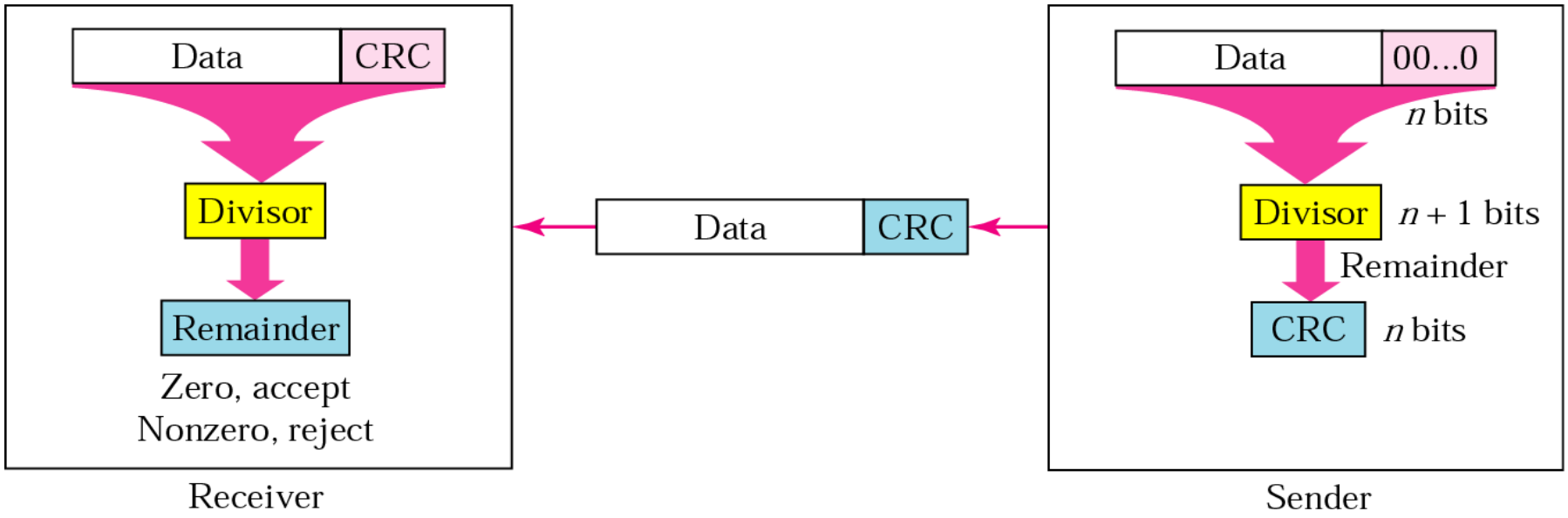
1. The data word 0100 becomes the code word 0100011. the code word 0100011 is received. The syndrome is 000, the final dataword is 0100
2. The data word 0111becomes the codeword 0011001 is received. The syndrome is 011. according to table b2 is in error. After flipping b2, the final data word is 0111
3. The dataword 1101 becomes the code word 1101000. the code word 0001000 is received. The syndrome is 101, which means that b0 is in error. After flipping b0, we get 0000, the wrong data word. This shows that our code can not correct two errors

Cyclic Codes

- In this scheme, if a code word is cyclically shifted, the result is another codeword

- b1=a0;b2=a1;b3=a2;b4=a3;b5=a4;b6=a5;b0=a6.

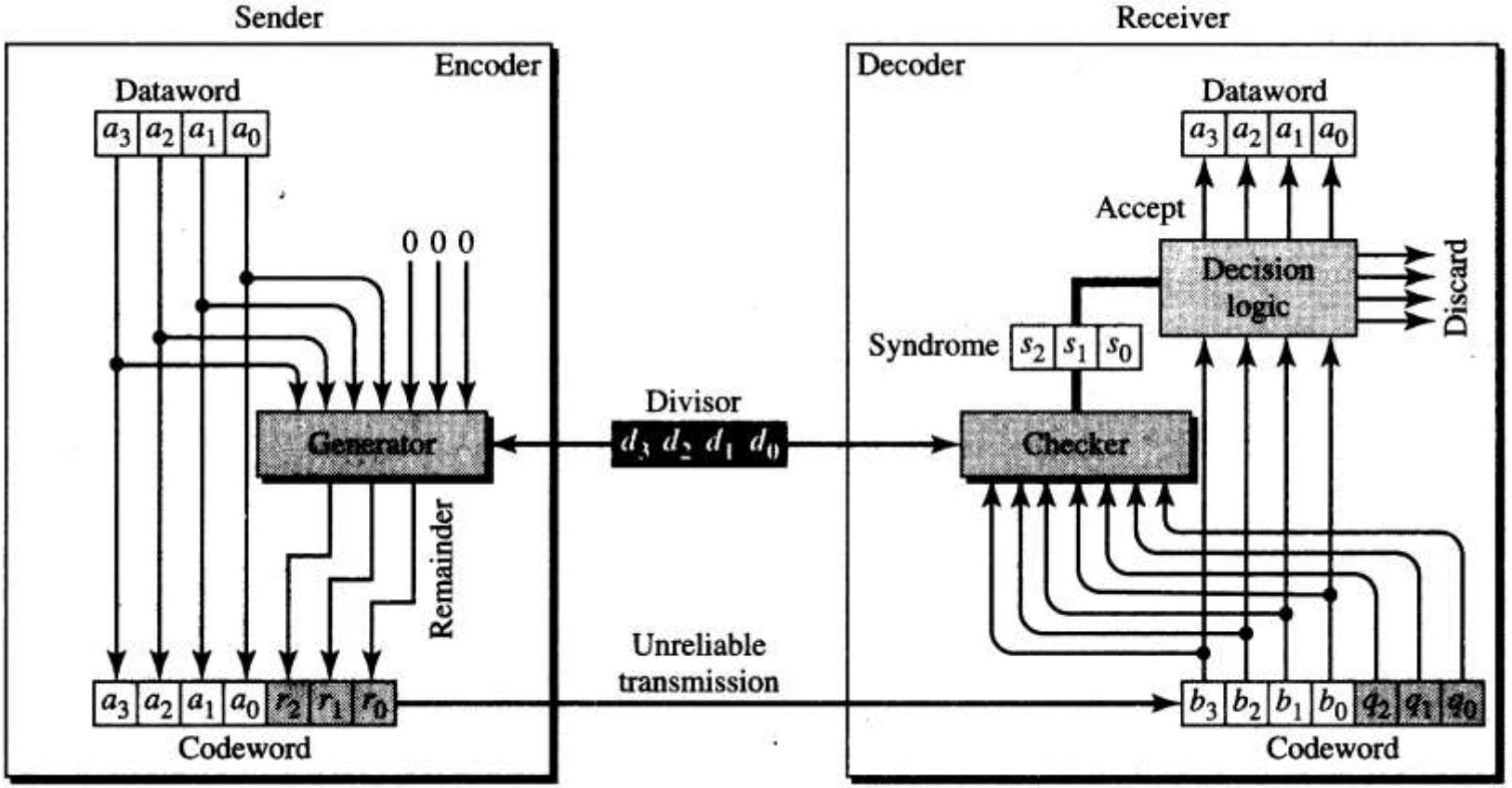
CRC generator and checker



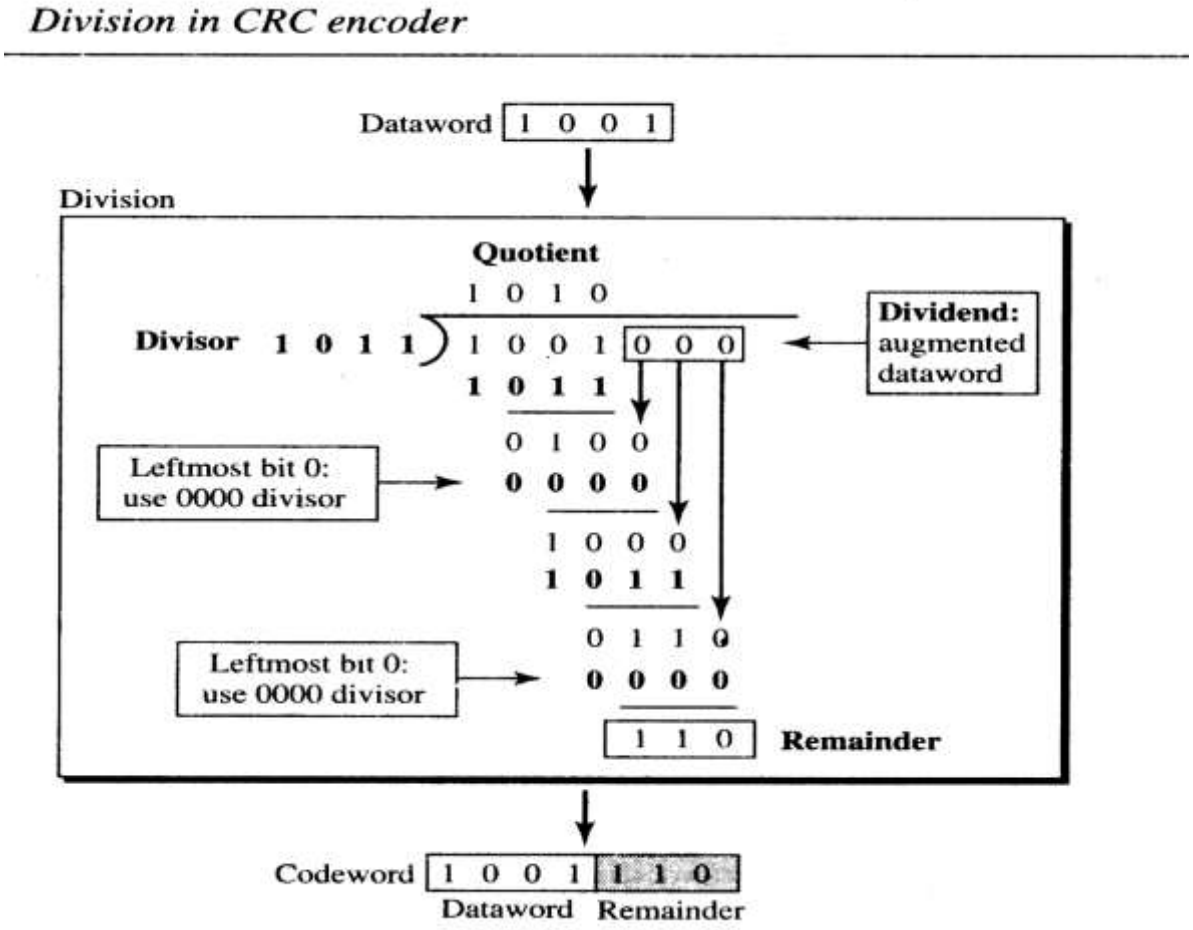
CRC

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

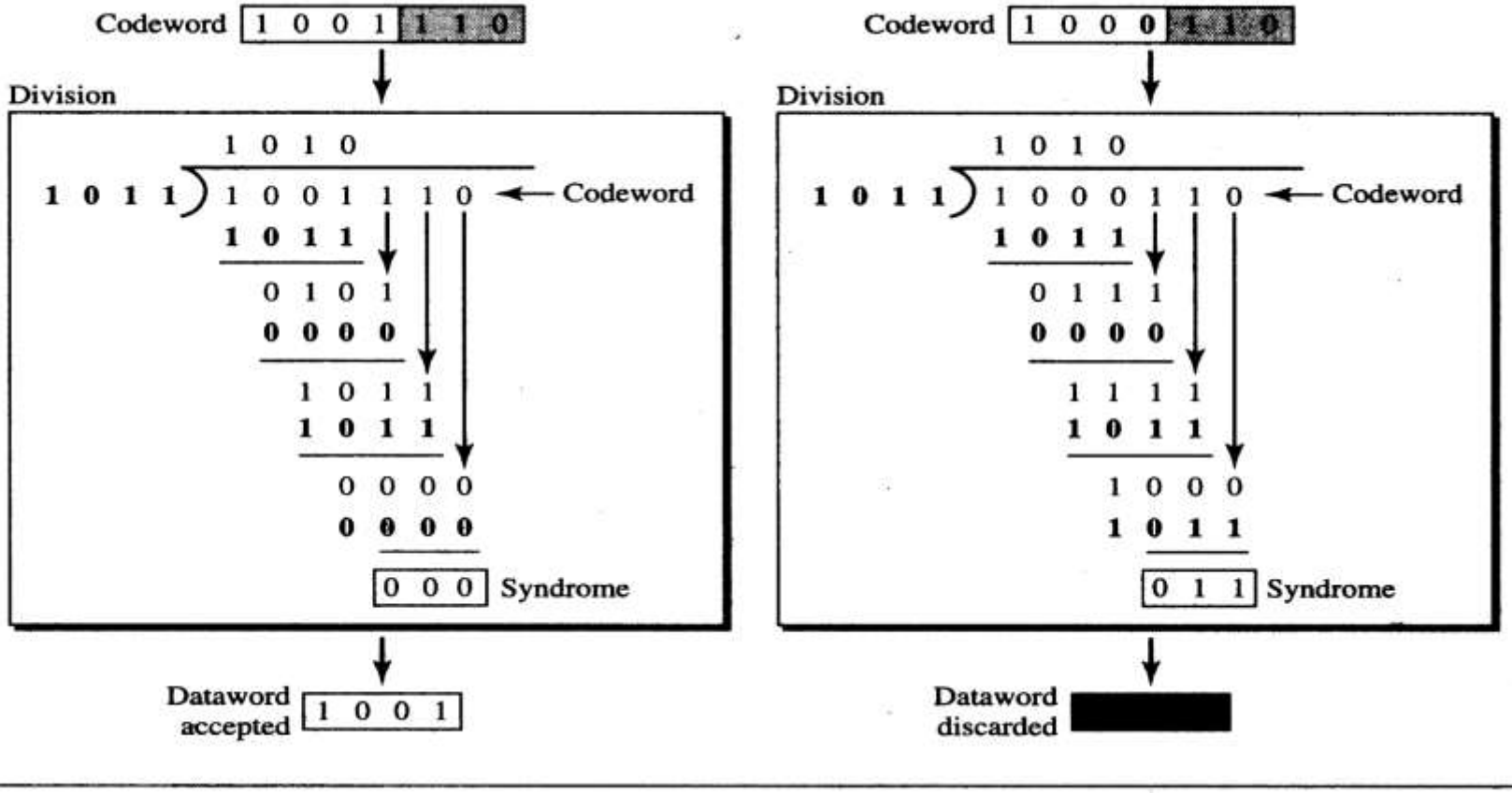
Design


Sender
Encoder
Dataword
a_3 a_2 a_1 a_0
0 0 0
Generator
Remainder
a_3 a_2 a_1 a_0 r_2 r_1 r_0
Codeword
Divisor
d_3 d_2 d_1 d_0
Unreliable transmission
Receiver
Decoder
Dataword
a_3 a_2 a_1 a_0
Accept
Decision logic
Discard
Syndrome
s_2 s_1 s_0
Checker
b_3 b_2 b_1 b_0 q_2 q_1 q_0
Codeword

Binary division in a CRC generator



Two cases – with /without Error



A polynomial and its representation

$$x^7 + x^5 + x^2 + x + 1$$

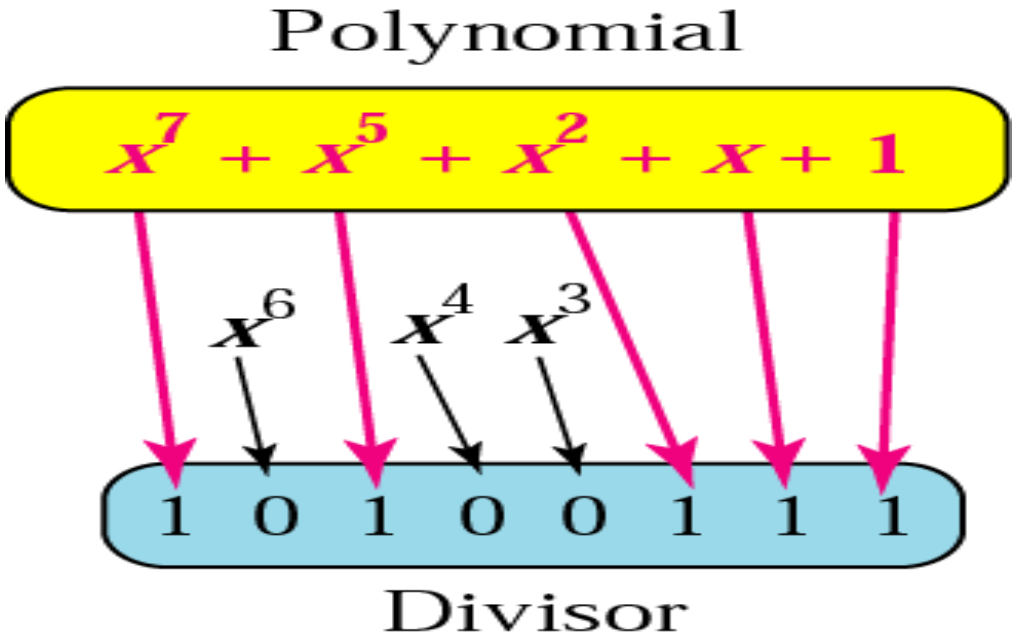
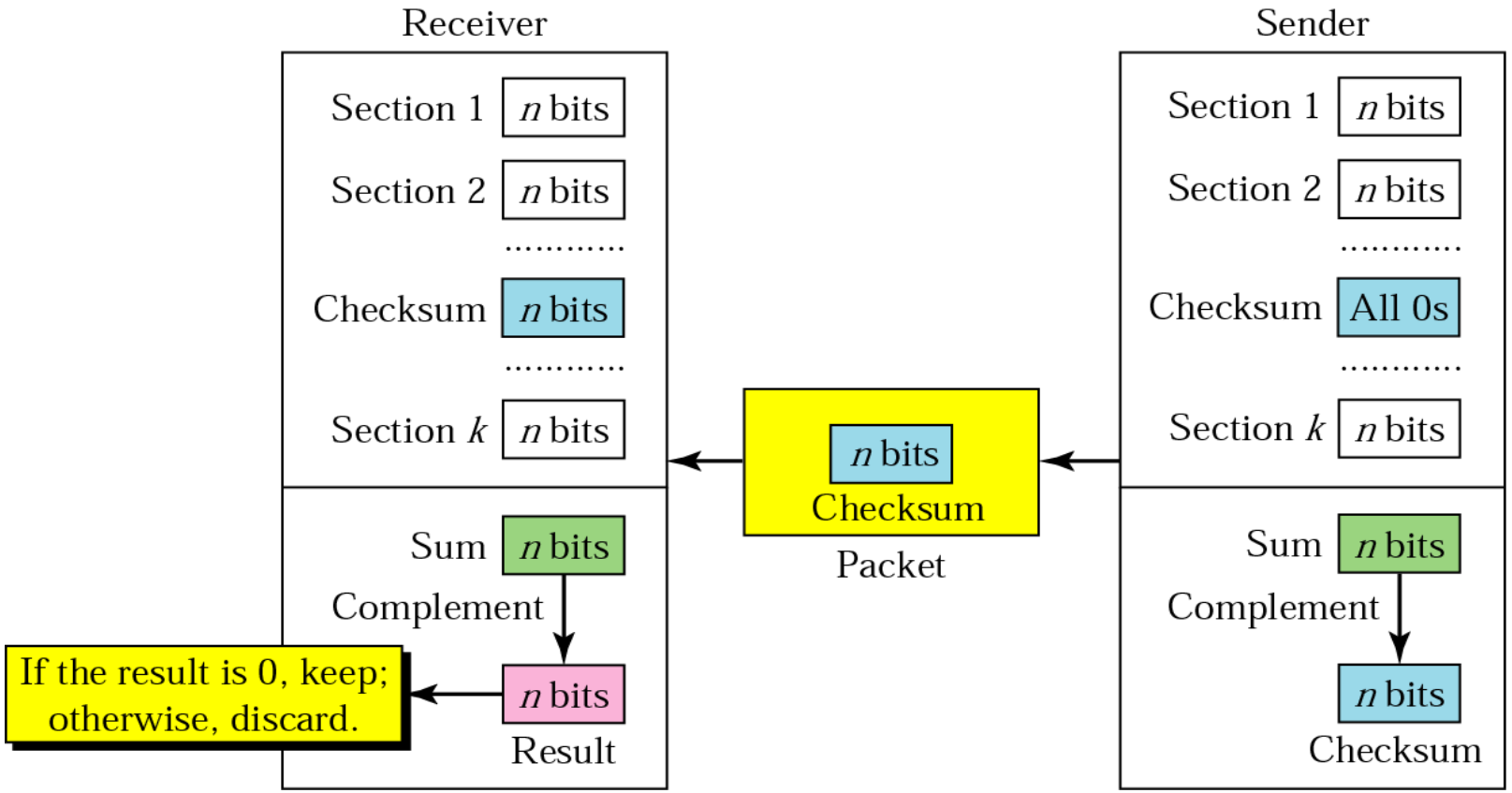


Table 10.1 Standard polynomials

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
ITU-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
ITU-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

Checksum


Receiver
Sender
Section 1
n bits
Section 2
n bits
Checksum
n bits
Section k
n bits
All 0s
Sum
Complement
Result
Checksum
Packet
If the result is 0, keep;
otherwise, discard.

Internet checksum

the internet has been using a 16-bit checksum. The sender calculates the checksum by following these steps.

The sender follows these steps:

- The message is divided into 16-bit words.
- The value of the checksum word is set to 0
- All words including the checksum are added using ones complement addition
- The sum is complemented and becomes the checksum.
- The checksum is sent with the data.

The receiver follows these steps:

- The message (including checksum) is divided into 16-bit words.
- All words are added using one's complement addition
- The sum is complemented and becomes the new checksum.
- If the value of checksum is zero, the data are accepted: otherwise, rejected.

Example 7

Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added using one's complement

10101001

00111001

Sum 11100010

Checksum 00011101

The pattern sent is 10101001 00111001 00011101

Example 8

Now suppose the receiver receives the pattern sent in Example 7 and there is no error.

10101001 00111001 00011101

When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

	10101001
	00111001
	00011101
Sum	**11111111**
Complement	**00000000 means that the pattern is OK.**

Example 9

Now suppose there is a burst error of length 5 that affects 4 bits.

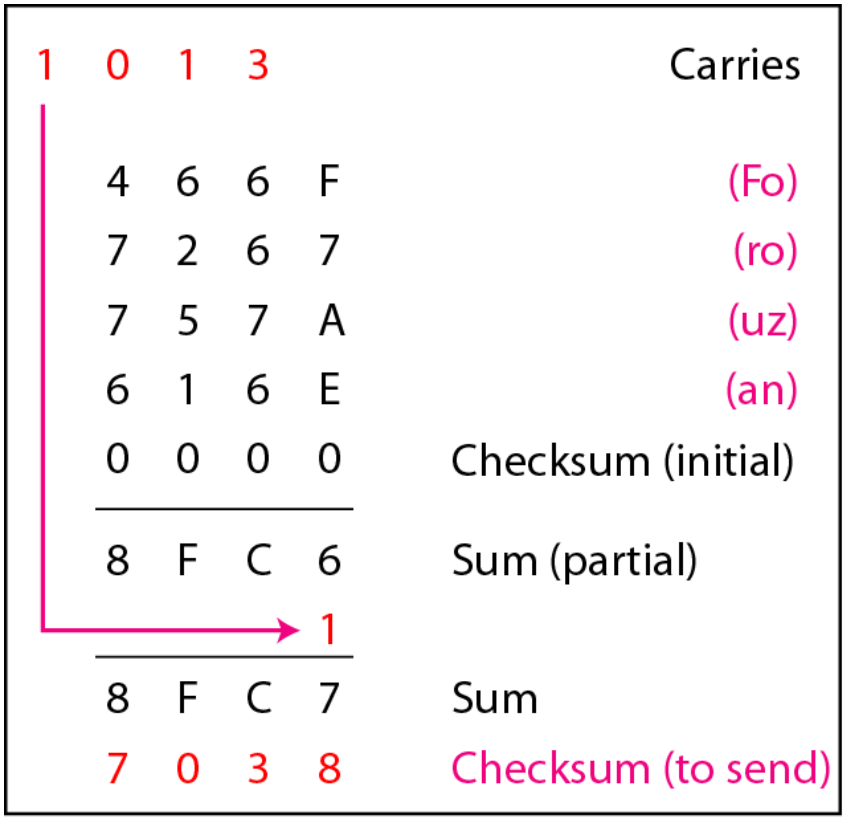
10101111 11111001 00011101

When the receiver adds the three sections, it gets

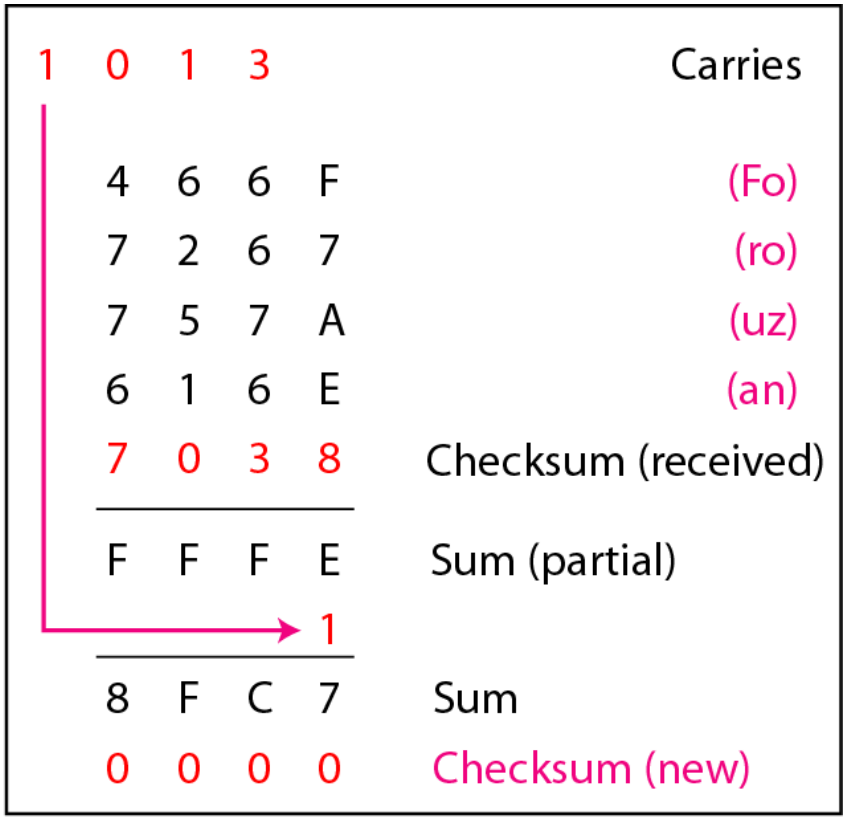
	1 0 1 0 1 1 1 1
	1 1 1 1 1 0 0 1
	0 0 0 1 1 1 0 1
Partial Sum	1 1 1 0 0 0 1 0 1
Carry	1
Sum	11000110
Complement	00111001 the pattern is corrupted

Let us calculate the checksum for a text of 8 characters (“Forouzan”). The text needs to be divided into 2-byte (16-bit) words. For example, F is represented as 0x46 and o is represented as 0x6F.

Figure 10.25 *Example 10.23*



a. Checksum at the sender site



a. Checksum at the receiver site

Data Link Control

Data link control

Functions:

- Framing
- Flow control
- Error control

Framing

- Data transmission in physical layer means moving bits in the form of a signal
- The data link layer needs to pack bits into frames.
- Each frames must be distinguishable
- **Framing ensures that a message is distinguished from others by attaching the sender's and receiver's addresses, thereby identifying the source and intended destination**.

Frame types: Fixed size, Variable size

Fixed size

- No need for defining boundaries
- Size itself is the delimiter:
- Ex: ATM network frame

Variable size

- Need to define the end of the frame and the beginning of the next.

Character Oriented / Byte Oriented / Byte stuffing protocols

- Data to be carried are 8-bit characters from a coding system-ASCII
- The header normally carries the source and destination addresses and other information
- The trailer carries error detection or error correction redundant bits, are also multiple of 8-bits.
- 1-byte Flag is added at the beginning and end of a frame.

Character-oriented protocol, stuffing, un-stuffing

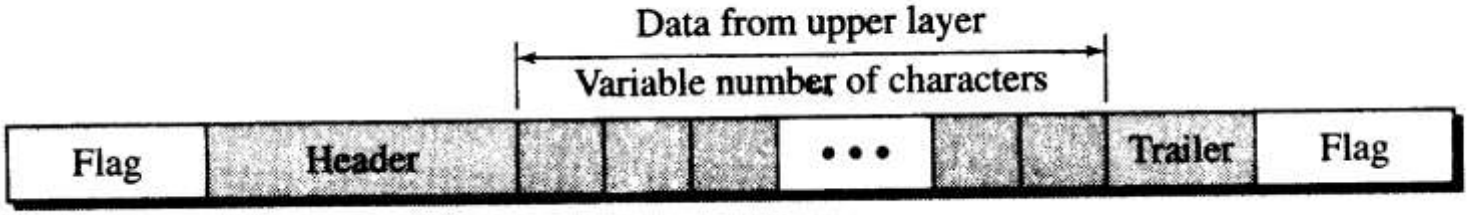
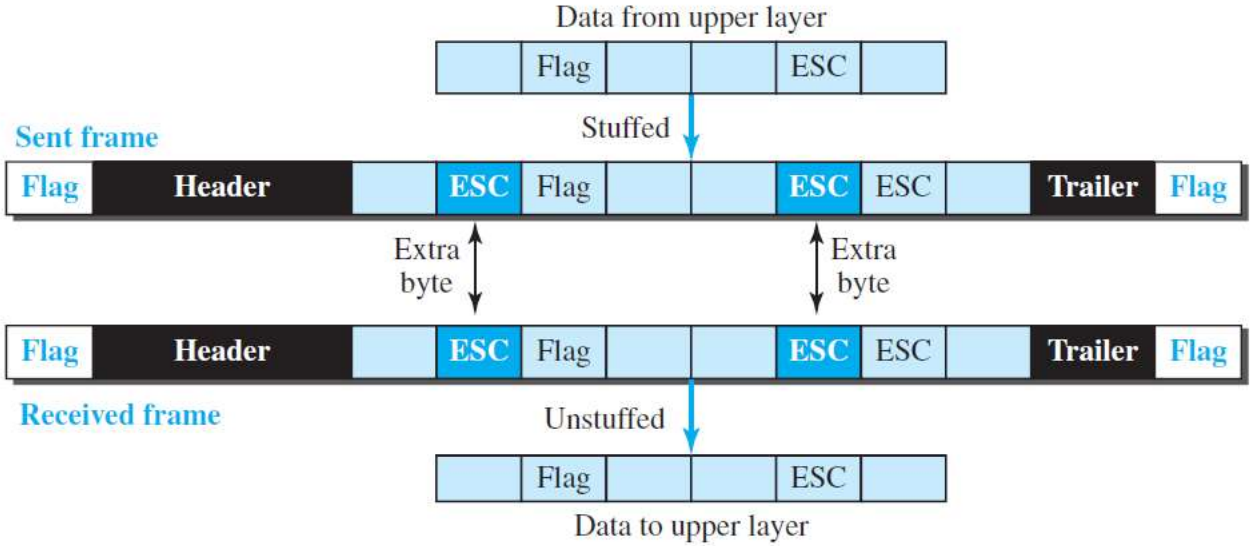


Figure 11.2 *Byte stuffing and unstuffing*



The flag type

- Byte stuffing strategy is used
  - A special byte is added to the data section of the frame when there is a character with the same pattern as flag.
  - This byte is called **escape character (ESC)**
  - Whenever the receiver encounters the ESC character; it removes it from data section and treats the next character as data

Bit-oriented protocols

- The data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, etc.
- In addition to header, delimiter is used to separate one frame from the other
- Usually the pattern flag 01111110 used as the delimiter.

- If the flag pattern appears in the data, we need to inform the receiver that this is not the end of the frame

- We do this by stuffing 1 single bit to prevent the pattern from looking like a flag. The strategy is called bit stuffing

- In bit stuffing, if five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is removed from the data by the receiver

Bit oriented

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

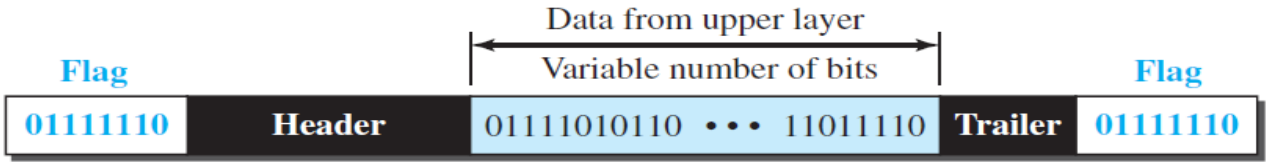
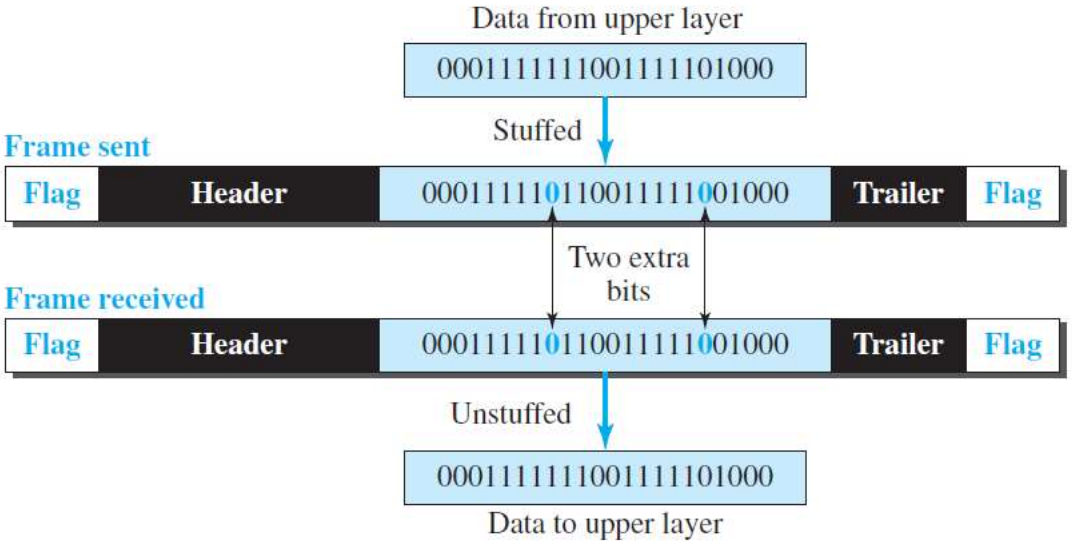


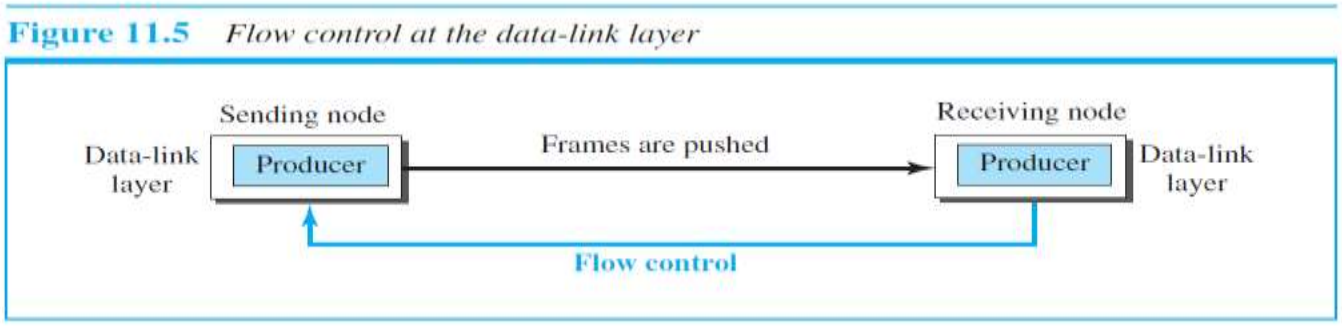
Figure 11.4 *Bit stuffing and unstuffing*



Flow and error control

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

Figure 11.5 *Flow control at the data-link layer*



- The figure shows that the data-link layer at the sending node tries to push frames toward the data-link layer at the receiving node. If the receiving node cannot process and deliver the packet to its network at the same rate that the frames arrive, it becomes overwhelmed with frames. Flow control in this case can be feedback from the receiving node to the sending node to stop or slow down pushing frames

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

Error control at the data-link layer is normally very simple and implemented using one of the following two methods. In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

❑ **Method 1 (Used in wired LANs like Ethernet)**

Sender action: Adds CRC (error check) in the frame header.

Receiver action:

If the frame is **corrupted** → discard silently.

If the frame is **not corrupted** → simply **deliver it to the network layer**.

No acknowledgment is sent back to the sender.

Key point: Assumes low error rate (like in Ethernet), so acknowledgments are unnecessary and would only add extra overhead.

❑ **Method 2**

Sender action: Same (adds CRC).

Receiver action:

If the frame is **corrupted** → discard silently.

If the frame is **not corrupted** → **send an acknowledgment** back to the sender.

Acknowledgment is used for both flow control and error control.

Key point: Useful in environments where errors are more frequent (e.g., wireless networks), since the sender needs confirmation.

- ***Connectionless Protocol***
- In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent.
- The frames are not numbered and there is no sense of ordering.
- Most of the data-link protocols for LANs are connectionless protocols.
- ***Connection-Oriented Protocol***
- In a connection-oriented protocol, a logical connection should first be established between the two nodes (setup phase).
- the frames are numbered and sent in order.
- Ex: wired in LANs, Point to point protocols , Wireless LANs and wired WANs

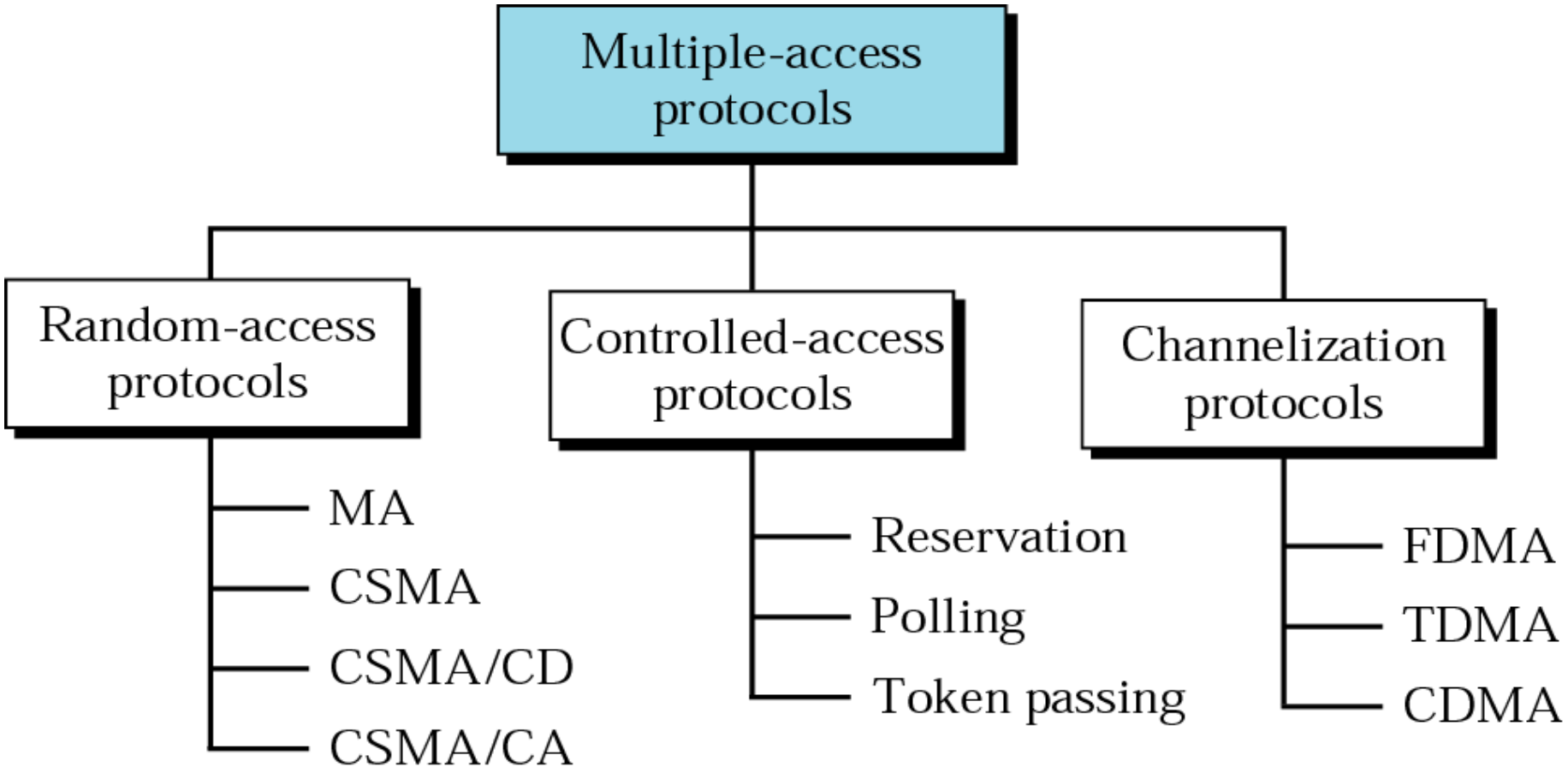
MEDIA ACCESS CONTROL-

Channelization

Multiple Access

- Sub layers of Data link:
  - **Data link control sublayer**
  - **Multiple access resolution sublayer**
- **IEE has actually made this division for LANs:**
  - **Logical Link Control(LLC layer)**: The upper sublayer – responsible for **flow and error control.**
  - **Medium Access Control(MAC layer)**: The lower sublayer– responsible for **multiple access resolution.**
- **Need**
  - A common link is shared by more than two nodes or stations
  - Multiple Access Protocol coordinate access to the link

Figure 13.1 Multiple-access protocols



Channelization

Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency and through code.

Channelization protocol:

- TDMA
- FDMA
- CDMA

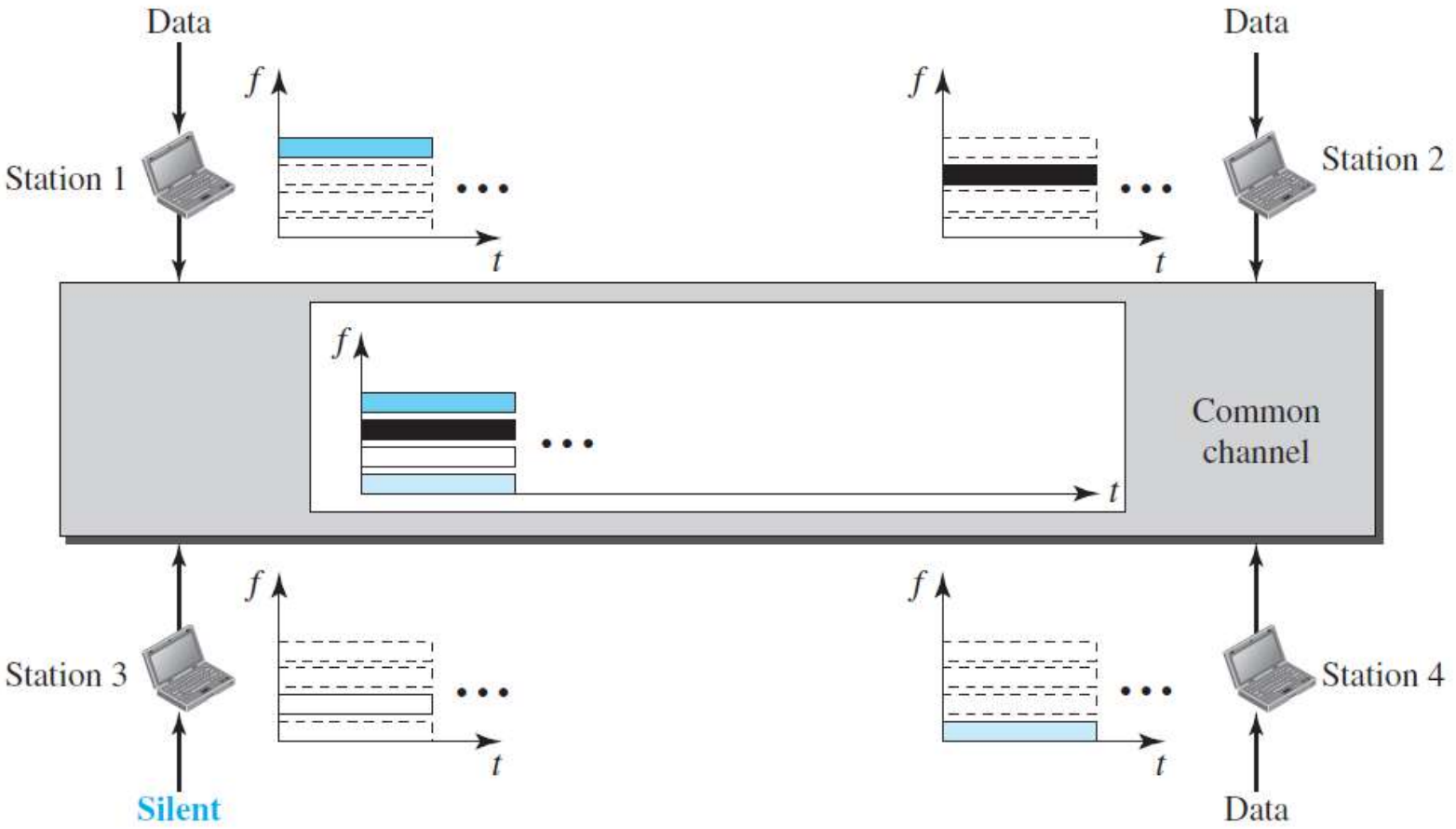
Frequency Division Multiple Access

- Available bandwidth is divided into frequency bands
- Each station is allocated a band to send its data
- Each band is reserved for a specific station and belongs to the station all the time.
- Each station also uses a bandpass filter to confine the transmitter frequencies.
- Small guard bands are used to separate allocated bands.
- To prevent station interferences, the allocated bands are separated from one another by small *guard bands.*

FDM and FDMA

- FDM is physical layer technique , it combines the loads from low-bandwidth channels and transmits them by using a high bandwidth channel.
- Multiplexers are used to modulate signals, to combine and to create a band pass signal
- FDMA is access method in the data link layer.
- The data link layer in each station tells its physical layer to make a band pass signal from the data passed to it.

Figure 12.21 *Frequency-division multiple access (FDMA)*



TDMA–Time Division Multiple Access

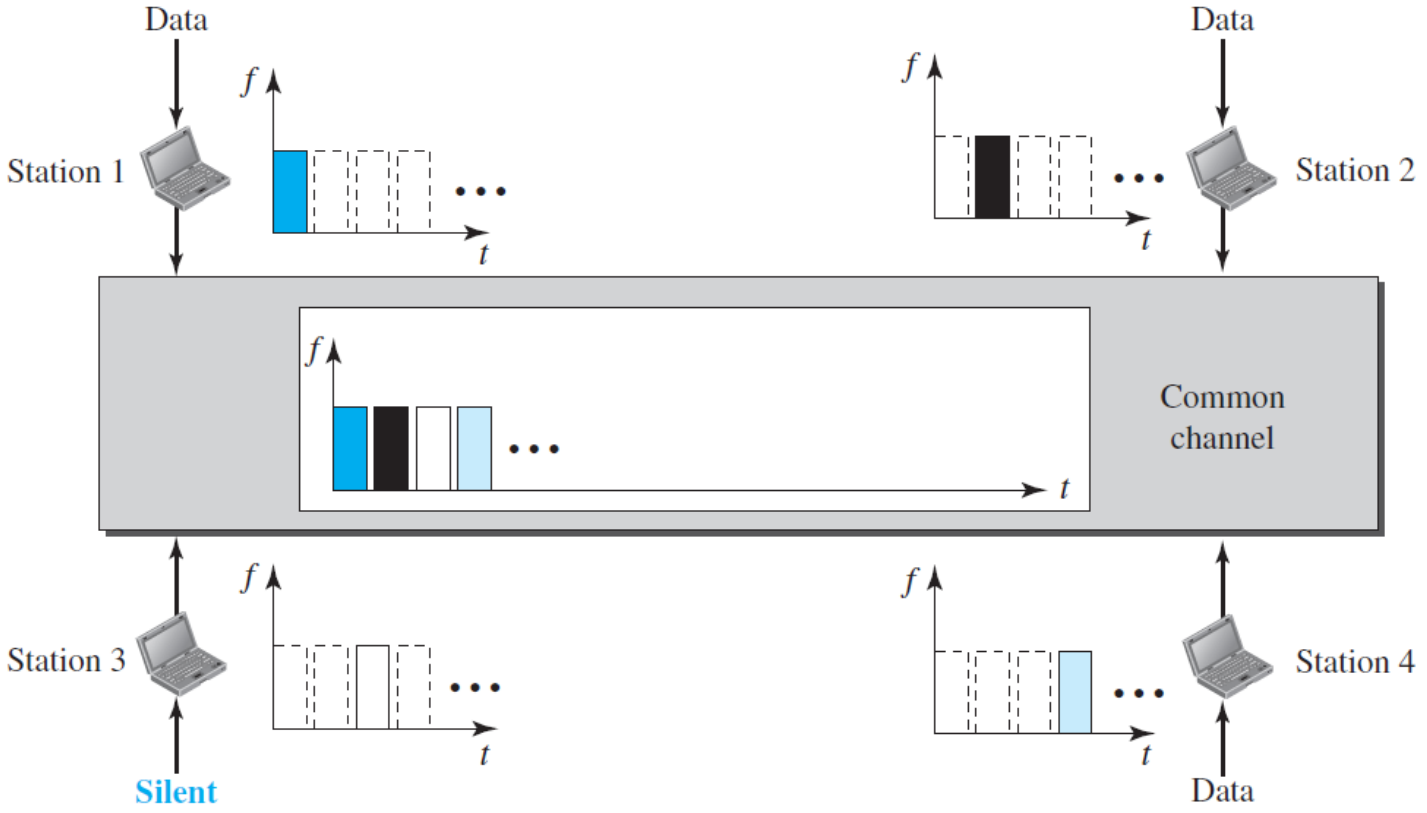
- The stations share bandwidth of the channel in time
- Each station is allocated a time slot during which it can send data.
- Each station transmits its data in its assigned time slot.

Main problem: **synchronization between different stations**

- Each station needs to know the beginning of its slot and location of its slot
- This difficult because of propagation delay introduced in the system if the stations are spread over large area
- To compensate for the delays, guard times is inserted to introduced for synchronization

TDMA

Figure 12.22 *Time-division multiple access (TDMA)*



TDM and TDMA

- **TDM** is **physical layer technique**
- TDM **combines** the data from **slower channels** and **transmits** them by using a **faster channel.**
- Multiplexers interleaves data units from each channel
- **TDMA** is an access method in the **data link layer**
- **Data link layer** in each station tells its **physical layer** to use the **allocated time slot.**

Code Division Multiple Access

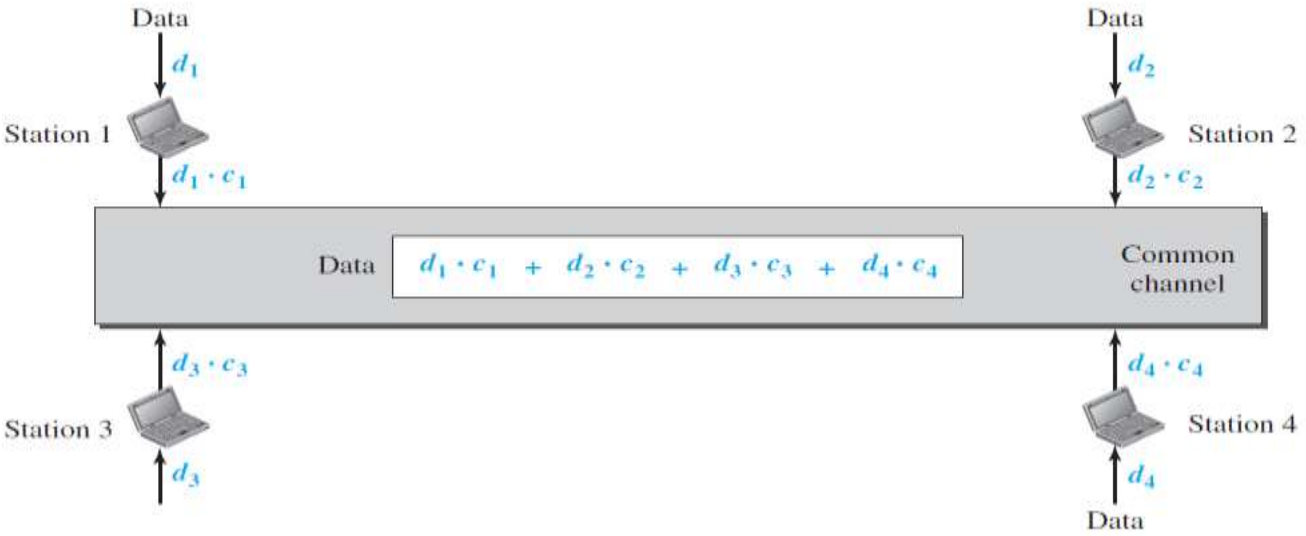
- It differs from FDMA: Only one channel occupies the entire bandwidth of the link.
- It differs from TDMA: all station can send data simultaneously ; there is no timesharing.

Idea

For stations 1,2,3 and 4 connected to the same channel
Data from 1 is d1 ,from 2 is d2 and so on
Station 1 multiplies its code by its data d1.c1 and so on for other stations
Codes have following properties:

1. If we multiply each code by another, we get 0
2. If we multiply each code by itself , we get 4

Figure 12.23 *Simple idea of communication with code*



For example, suppose stations 1 and 2 are talking to each other.
Station 2 wants to hear what station I is saying.
It multiplies the data on the channel by cl' the code of station 1.
Because (cl . cl) is 4, but *(c2 . cl), (c3 . cl), and (c4 . cl) are all Os, station 2 divides* the result by 4 to get the data from station 1.

Chip sequences

C_1	C_2	C_3	C_4
[+1 +1 +1 +1]	[+1 −1 +1 −1]	[+1 +1 −1 −1]	[+1 −1 −1 +1]

Properties:

1. Each sequence is made of N elements
2. If we multiply a sequences by a number, every element in the sequences is multiplied by that element
   2 • [+1 +1 −1 −1] = [+2 +2 −2 −2]
3. If we multiply two equal sequences, element by element and add the results , we get N- number of elements
   [+1 +1 −1 −1] • [+1 +1 −1 −1] = 1 + 1 + 1 + 1 = 4
4. If multiply two different sequences, element by element and add results , we get 0. **[+1 +1 −1 −1] • [+1 +1 +1 +1] = 1 + 1 − 1 − 1 = 0**
5. Adding two sequences, means adding the corresponding elements. The result is another sequences
   [+1 +1 −1 −1] + [+1 +1 +1 +1] = [+2 +2 0 0]

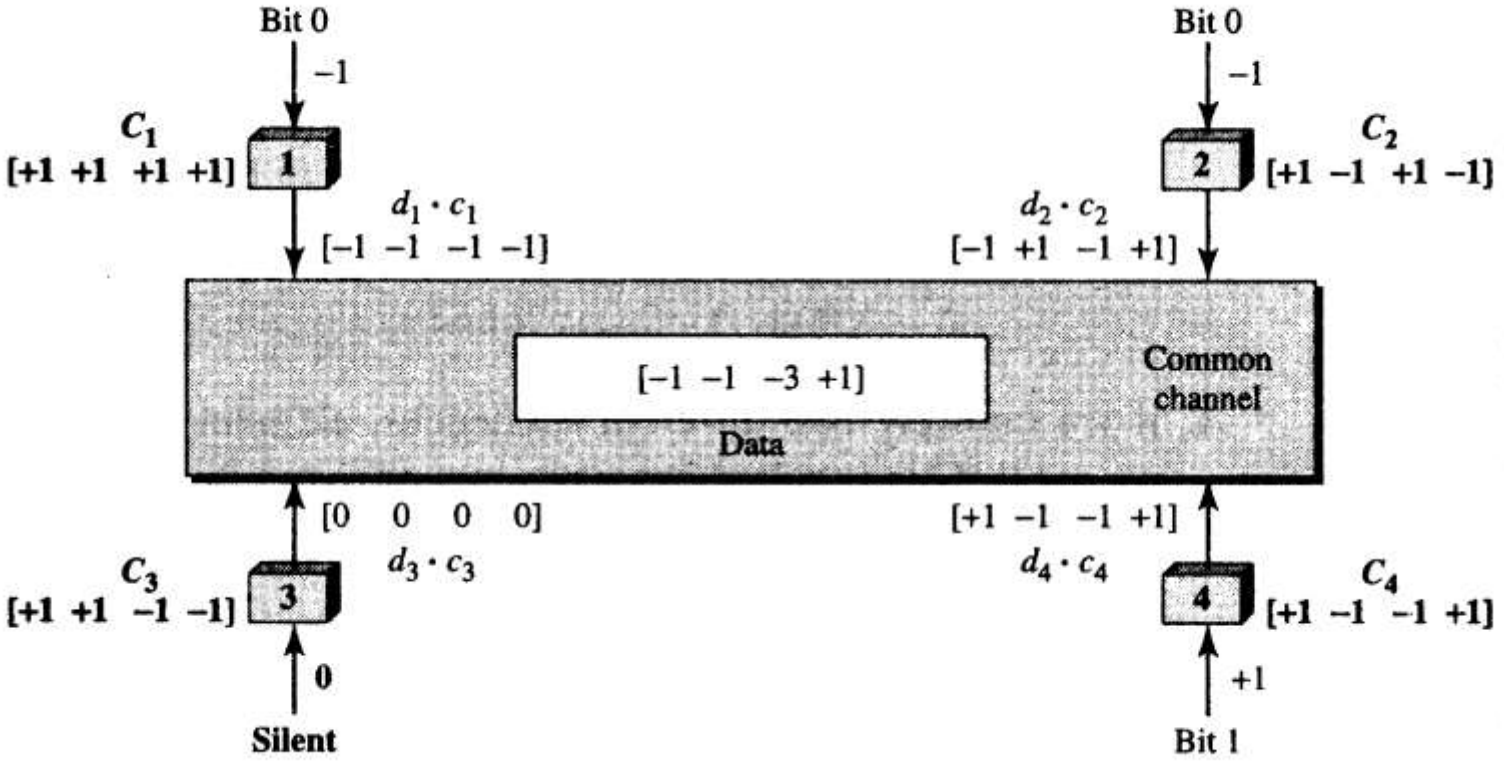
Sequence generation

Data bit 0 → −1 Data bit 1 → +1 Silence → 0

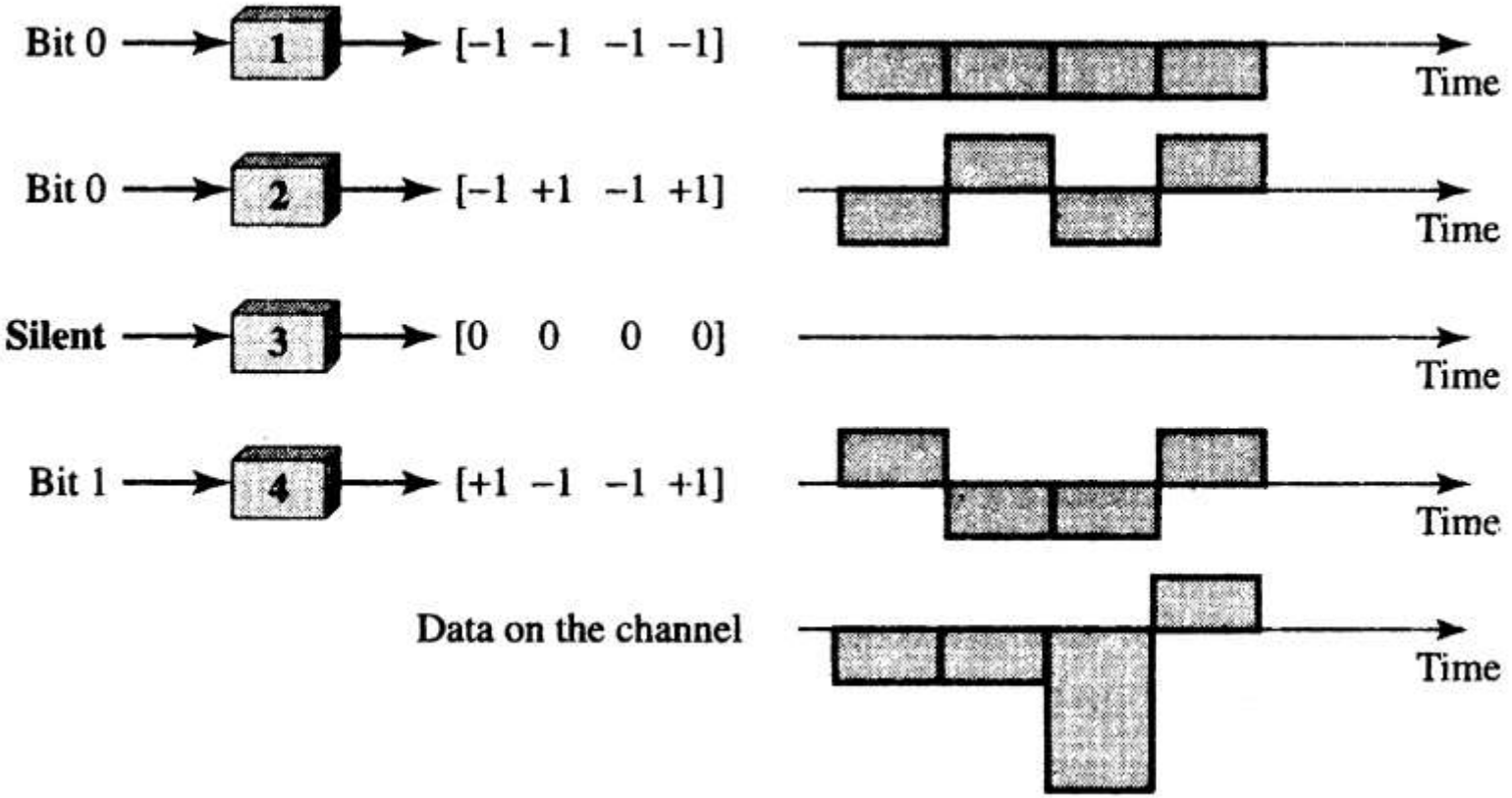
- Assume that 1 and 2 sending a 0 bit and channel 4 is sending a 1 bit, and station 3 is silent.
- Data at sender site are translated to -1 -1 0 +1
- Each station multiplies the chip sequences
- The result is a new sequences which is sent to the channel
- Each station multiplies code of itself and total data on the channel

[-1 -1 -3 +1].[+1 -1 +1 -1]=-4/4=-1

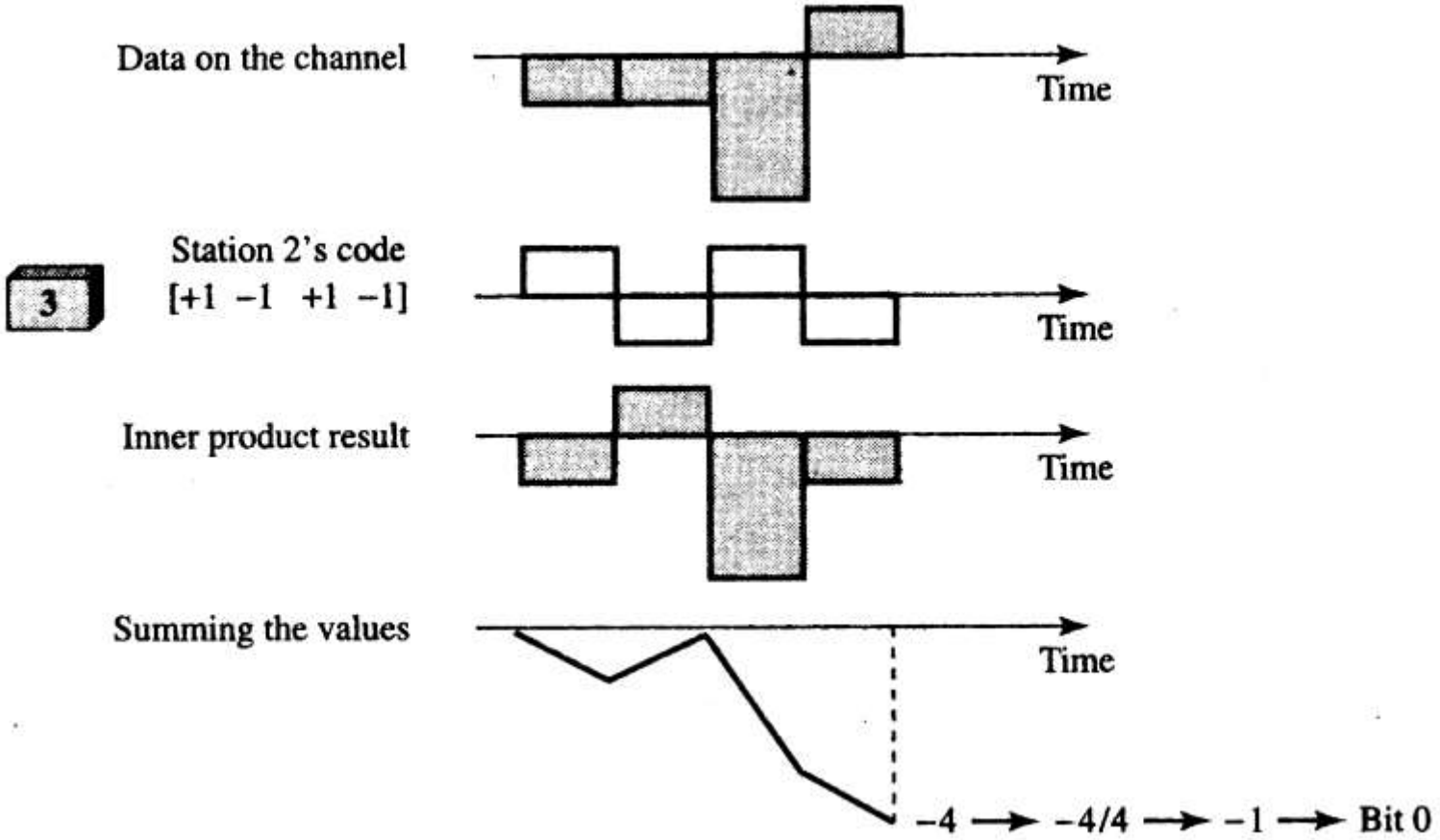
Sharing channel in CDMA



Signal Level



Decoding of composite signal



General rule and examples of creating Walsh tables

Figure 12.29 *General rule and examples of creating Walsh tables*

$$W_1 = [+1] \qquad W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

a. Two basic rules

$$W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \qquad W_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

b. Generation of W_2 and W_4

EXAMPLE: Find the chips for a network with

a. Two stations

b. Four stations

We can use the rows of *W2 and W4*

a. For a two-station network, we have [+1 +1] and [+1 -1].

b. For a four-station network we have [+1 +1 +1 +1], [+1 -1 +1 -1], [+1 +1 -1 -1], and [+1-1-1 +1].

Prove that a receiving station can get the data sent by a specific sender if it multiplies the entire data on the channel by the sender's chip code and then divides it by the number of stations.

Solution

Let us prove this for the first station, using our previous four-station example. We can say that the data on the channel

D = (d1 · c1 + d2 · c2 + d3 · c3 + d4 · c4).

The receiver which wants to get the data sent by station 1 multiplies these data by c1.

$$
\begin{aligned}
D \cdot c_1 &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1 \\
&= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 \\
&= d_1 \times N + d_2 \times 0 + d_3 \times 0 + d_4 \times 0 \\
&= d_1 \times N
\end{aligned}
$$

When we divide the result by N, we get d1 .

What is the number of sequences if we have 90 stations in our network?

Solution

The number of sequences needs to be 2^m. We need to choose $m = 7$ and $N = 2^7$ or 128. We can then use 90 of the sequences as the chips.

Model Questions

1. How does datawords and codewords is represented in block coding and also explain how can errors be detected and corrected by using block coding .
2. Find codeword, using cyclic redundancy code given generator 1011, data word 1001 and show how it is used to check for error detection in the receiver side
3. find codeword, using cyclic redundancy code given generator 1011, data word 1001 and show how it is used to check for error detection in the receiver side
4. Define hamming distance. Explain simple parity check code C(5,4) with dmin=2. How many bits can be corrected
5. Find the code word using CRC given data “1001” and generator “1011”
6. Why bit stuffing and byte stuffing are needed? Explain them, with examples