NETWORK PROTOCOLS & SECURITY
23EC2210 R/A/E

Topic:

ERROR CONTROL

Session - 8 & 9

AIM OF THE

To familiarize students with the basic idea of Error Control in Data link layer.

INSTRUCTIONAL

This Session is designed to:

1. Describe different types of errors.
2. Understand different types of error detection methods.
3. Understand error correction mechanism.

LEARNING

At the end of this session, you should be able to:

1. Identify the types of errors.
2. Apply different error detection techniques to identify the error.
3. Apply hamming code to detect and correct the error in the data.

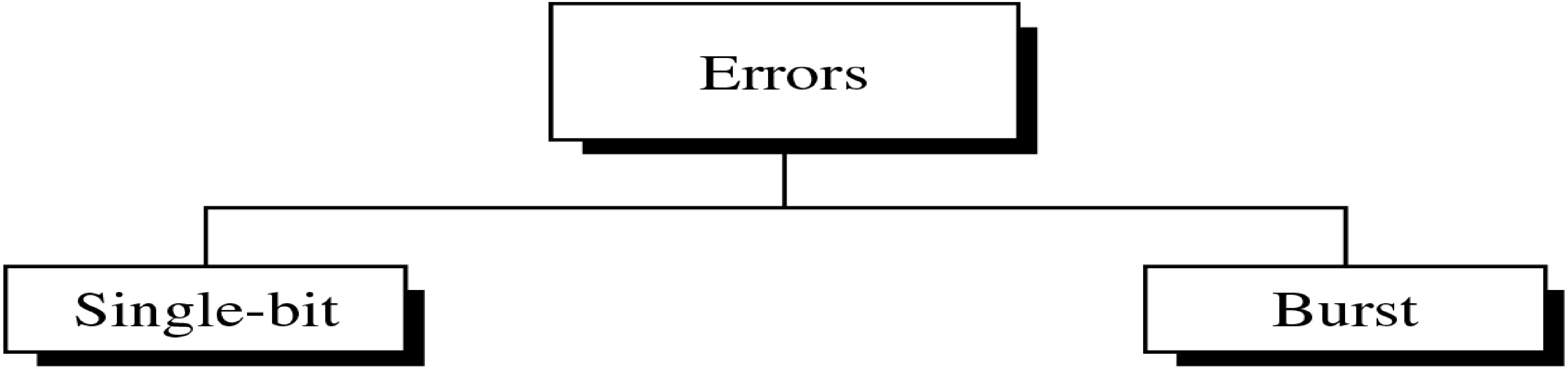
Error Detection and Correction

- Types of Errors
- Detection
- Error Correction

Introduction

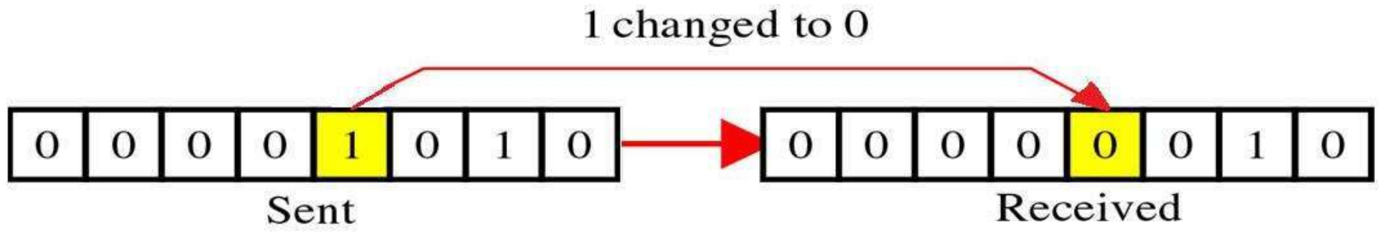
- Data can be corrupted during transmission. For reliable communication, error must be detected and corrected.
- Error Detection and Correction are implemented either at the data link layer or the transport layer of the OSI model.

Type of Errors

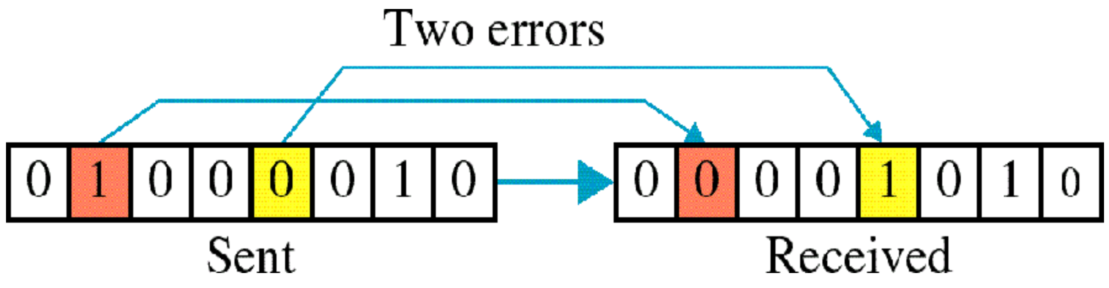


Type of Errors(cont'd)

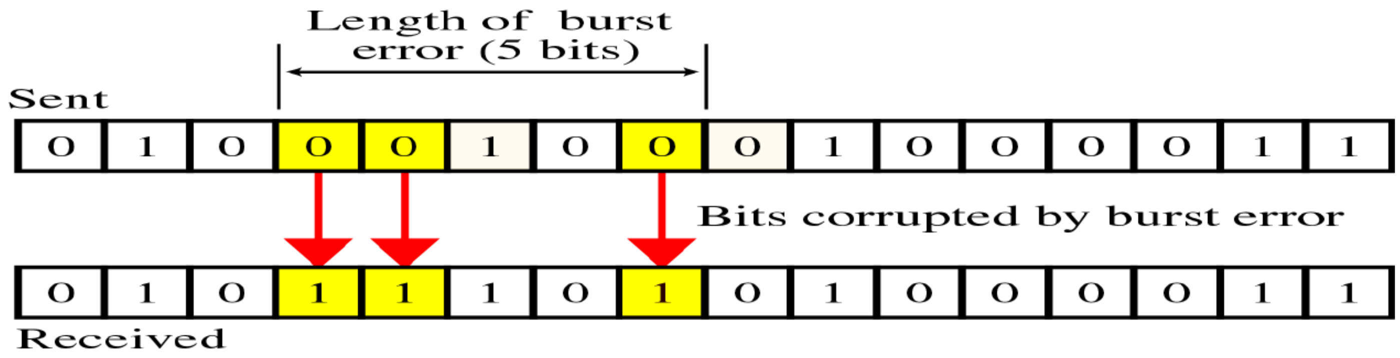
- **Single-Bit Error**: Only one bit in the data unit has changed



- **Multiple-Bit Error:** Two or more nonconsecutive bits in the data unit have changed.

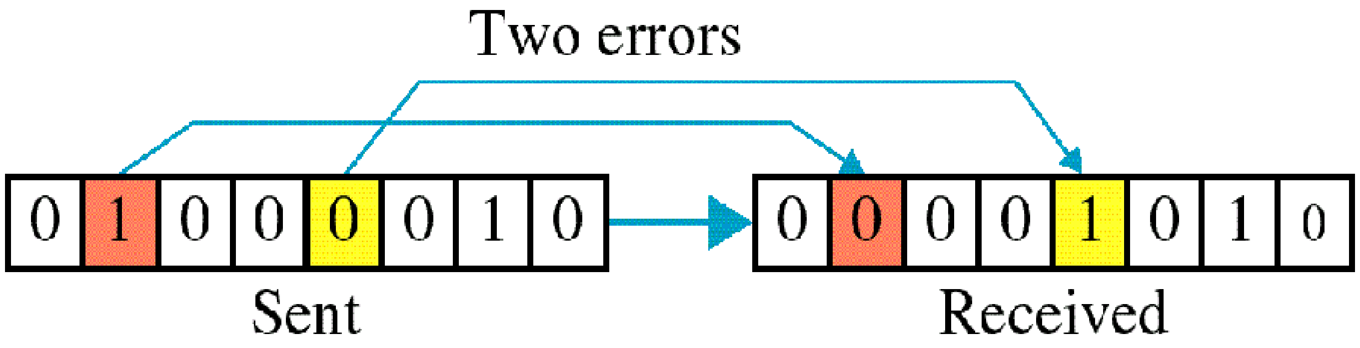


- **Burst Error:** Two or more consecutive bits in the data unit have changed



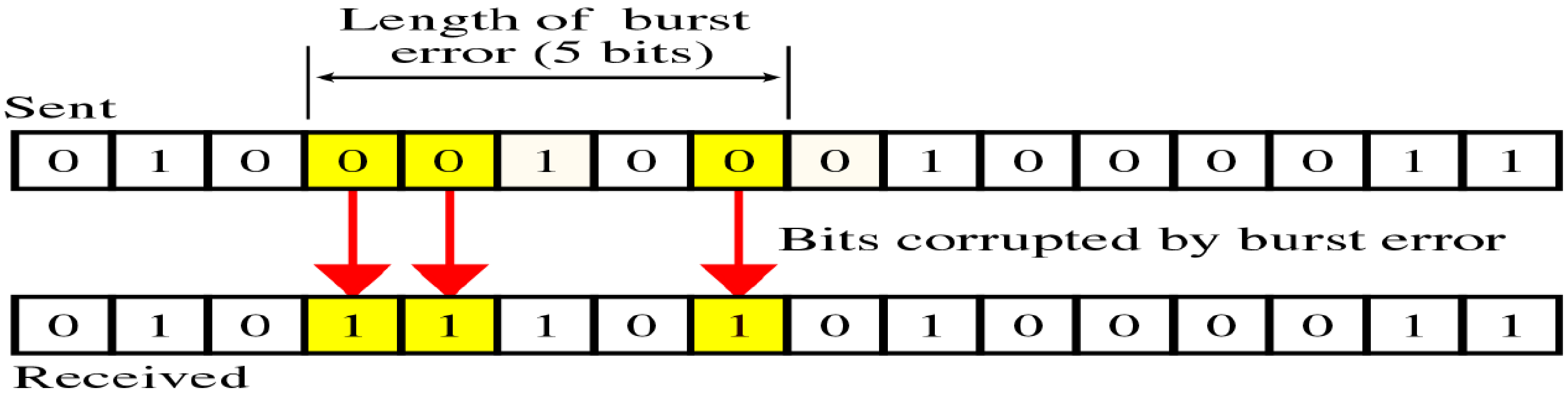
Type of Errors(cont'd)

- **Multiple-Bit Error:** is when two or more nonconsecutive bits in the data unit have changed.



Type of Errors(cont'd)

- **Burst Error** means that 2 or more consecutive bits in the data unit have changed



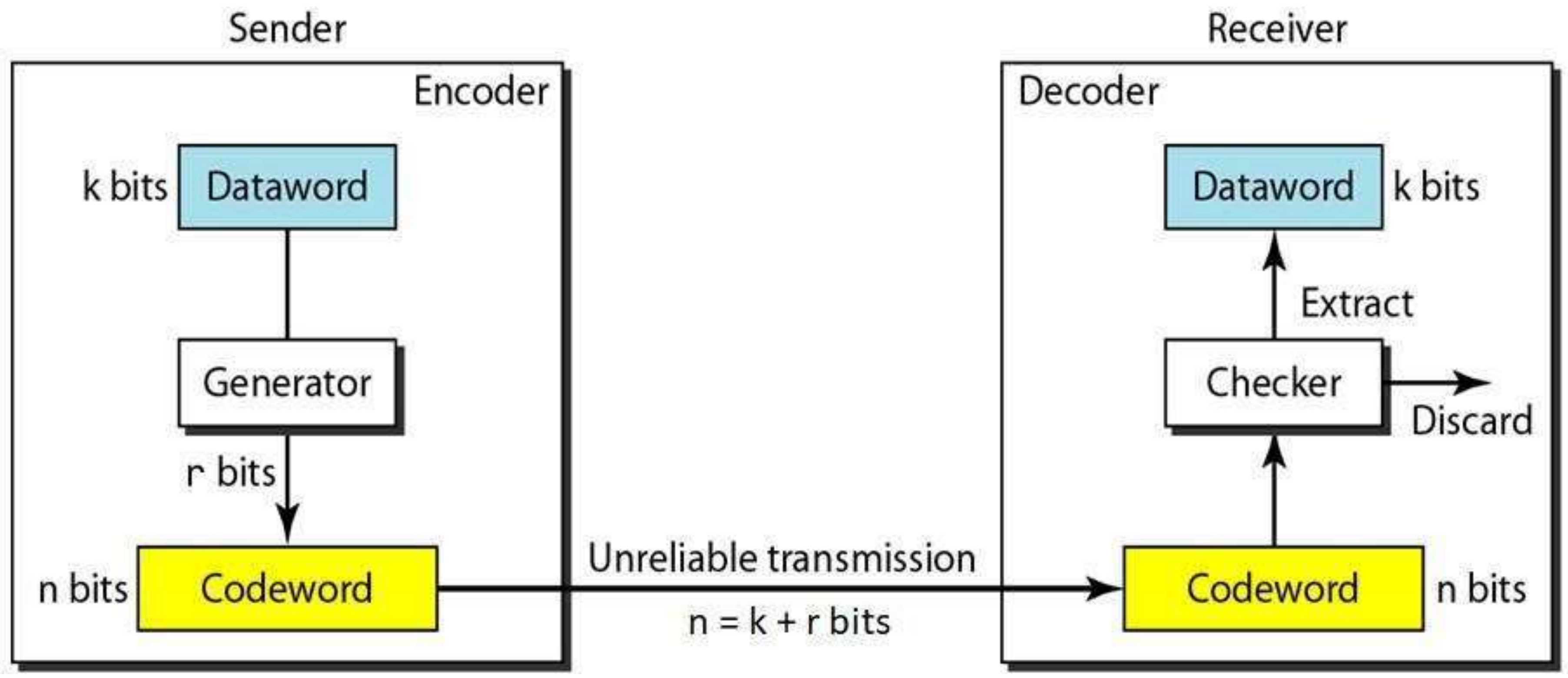
Error Detection

Error Detection

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

- **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.*

The structure of encoder and decoder

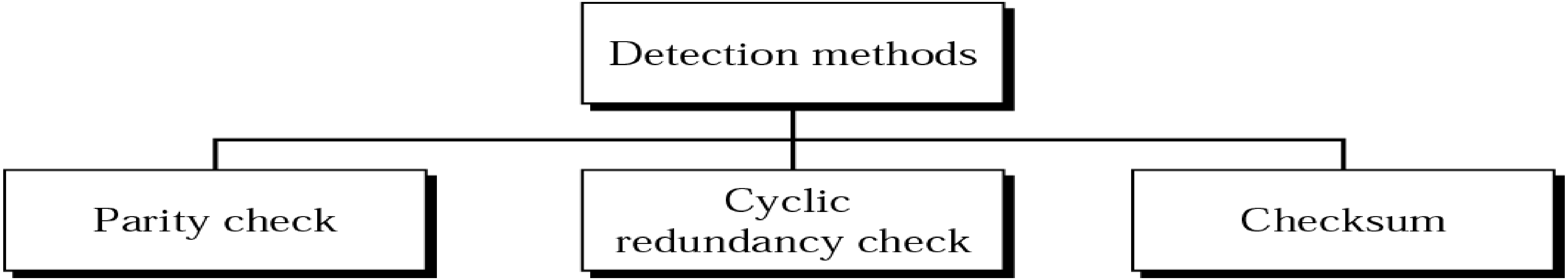


To detect or correct errors, we need to send redundant bits

Error Detection Methods

Error Detection Methods

- Detection methods



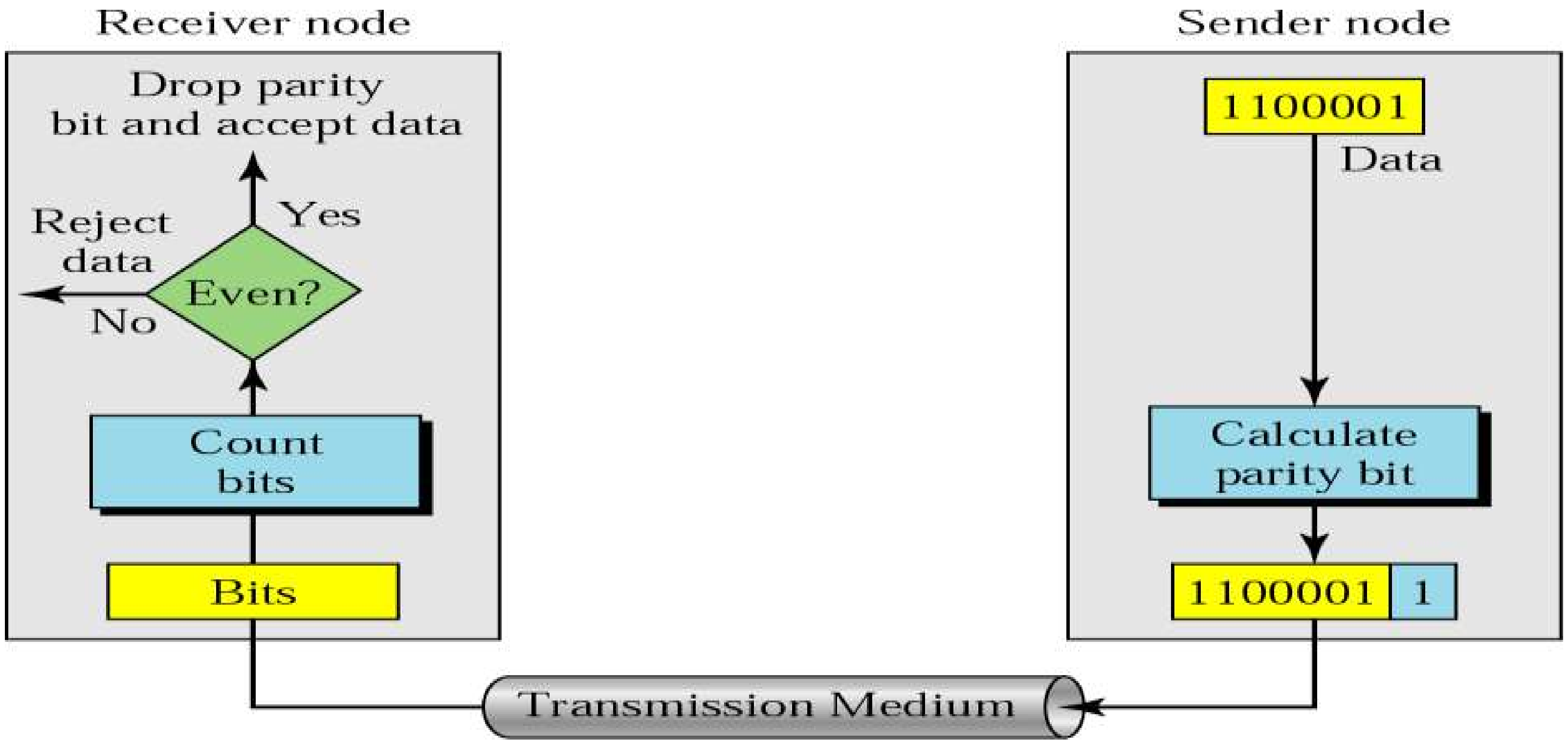
1. Parity Method

Simple Parity Method

1. Parity Check:

- A parity bit is added to every data unit so that the total number of 1s(including the parity bit) becomes even for even-parity check or odd for odd-parity check.

- **Simple parity check**



Parity method examples

Example 1

Suppose the sender wants to send the word "***world***".

In ASCII the five characters are coded as

1110111 1101111 1110010 1101100 1100100

Sent data:

1110111**0**	1101111**0**	1110010**0**	1101100**0**	1100100**1**

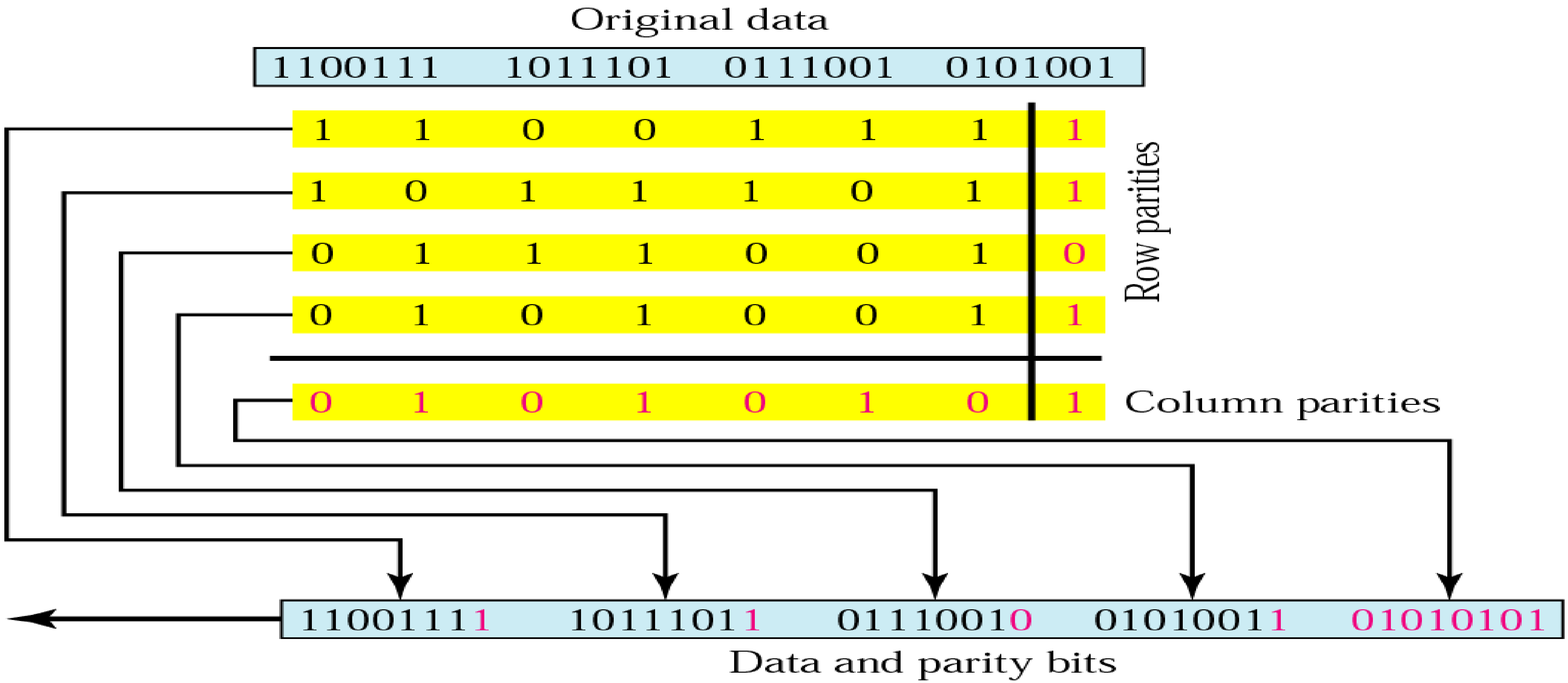
Drawback of Simple Parity

- M = 1100101
- Parity followed: Even

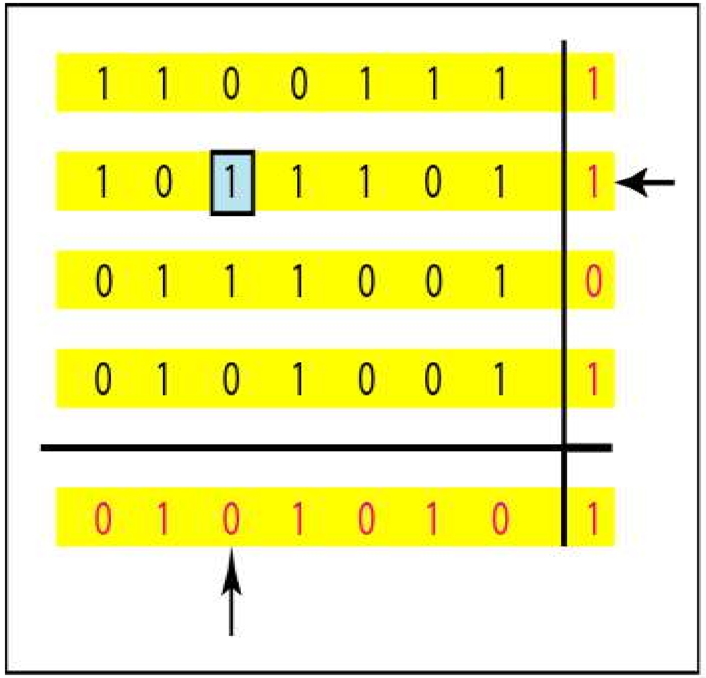
Sent Message	Received Message	Parity	Message Status
11001010	11001110	Odd	Discard
11001010	11101110	Even	Accept
11001010	11101100	Odd	Discard
11001010	10101100	Even	Accept

Disadvantage: With Simple parity method, only the odd number errors can be detected.

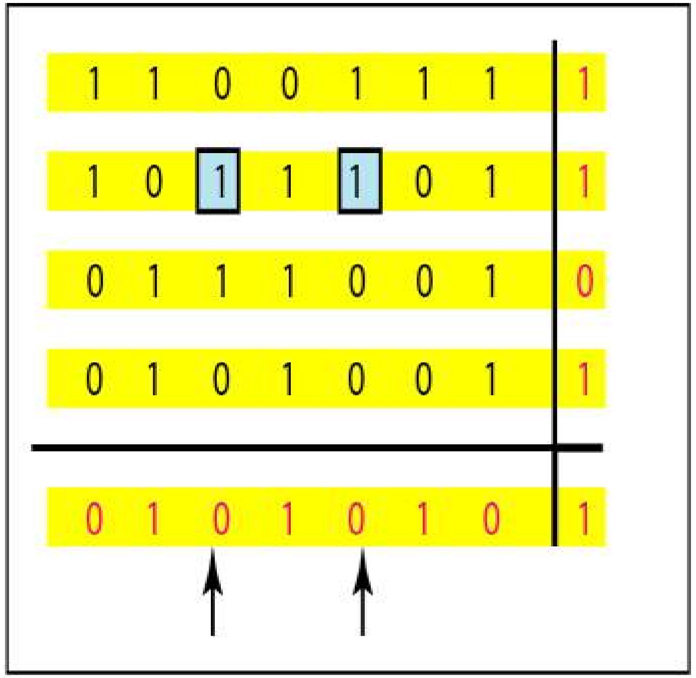
Two-Dimensional Parity Method



Two-dimensional parity-check code

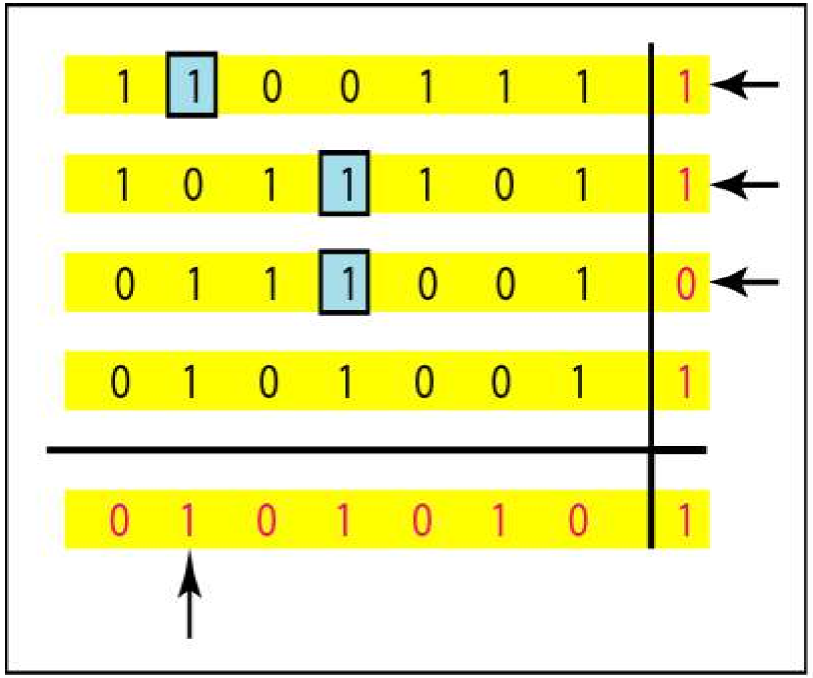


b. One error affects two parities

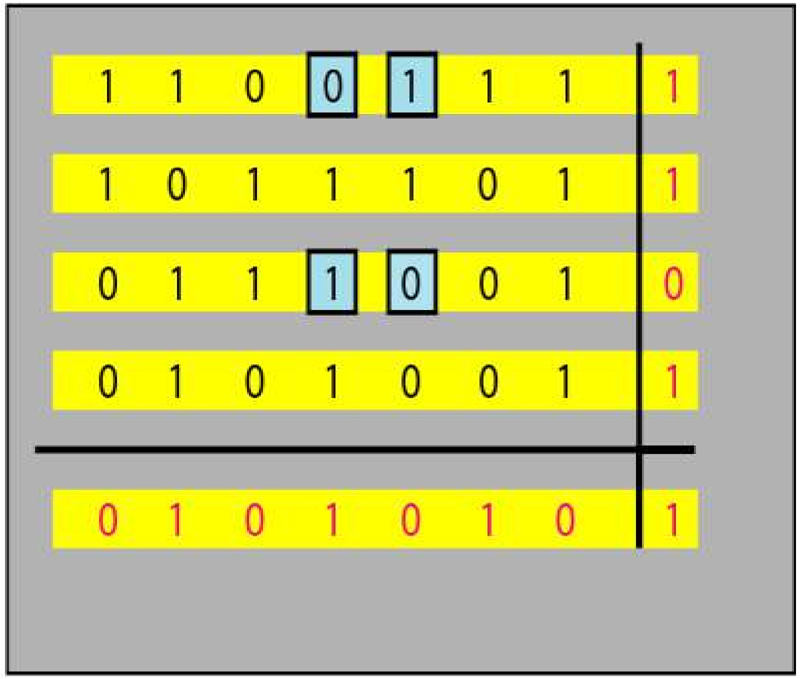


c. Two errors affect two parities

Two-dimensional parity-check code…



d. Three errors affect four parities



e. Four errors cannot be detected

2. Cyclic Redundancy Check (CRC)

2. Cyclic Redundancy Check (CRC):

- Cyclic Redundancy Check (CRC) is an error detection method.
- It is based on binary division.

CRC Generator-

- CRC generator is an algebraic polynomial represented as a bit pattern.
- Bit pattern is obtained from the CRC generator using the following rule-

> ***The power of each term gives the position of the bit and the coefficient gives the value of the bit.***

Example-

- Consider the CRC generator is $x^7 + x^6 + x^4 + x^3 + x + 1$. The corresponding binary pattern is obtained as-

$$1x^7 + 1x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1$$

Steps in Computing CRC:

Step-01: Calculation Of CRC At Sender Side-

At sender side,

- A string of n 0's is appended to the data unit to be transmitted.
- Here, n is one less than the number of bits in CRC generator.
- Binary division is performed of the resultant string with the CRC generator.
- After division, the remainder so obtained is called as **CRC**.
- It may be noted that CRC also consists of n bits.

Step-02: Appending CRC To Data Unit-

- At sender side,
- The CRC is obtained after the binary division.
- The string of n 0's appended to the data unit earlier is replaced by the CRC remainder.

Step-03: Transmission To Receiver-

- The newly formed code word (Original data + CRC) is transmitted to the receiver.

Steps in Computing CRC…

<u>Step-04: Checking at Receiver Side-</u>

At receiver side,

- The transmitted code word is received.
- The received code word is divided with the same CRC generator.
- On division, the remainder so obtained is checked.

The following two cases are possible-

Case-01: Remainder = 0

- If the remainder is zero,
- Receiver assumes that no error occurred in the data during the transmission.
- Receiver accepts the data.

Case-02: Remainder ≠ 0

- If the remainder is non-zero,
- Receiver assumes that some error occurred in the data during the transmission.
- Receiver rejects the data and asks the sender for retransmission.

CRC Example

A bit stream 11101101 is to be transmitted using the standard CRC method. The generator polynomial given is x^3+1.

1. What is the actual bit string transmitted?

2. Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?

CRC Example : Solution

Solution:

1) Data = 11101101

GP = x^3+1 = 1001

Append three 0's to Data and divide it with GP

```
          11110011
1001 | 11101101000
       1001
       01111
        1001
        01101
         1001
         01000
          1001
          0001100
             1001
             01010
              1001
              0011
```

CRC = 011

$\boxed{\text{Bit String transmitted} = \text{Data} + \text{CRC} = 11101101011}$

CRC Example : Solution

2) Third bit from the left is inverted.

Transmitted data : 1110110101 1

Received data : 1100110101 1

↑ 3rd bit

<u>Error Checking Process</u>:

Divide the received data with GP.

```
          11010111
1001 | 110011010111
       1001↓
       01011
        1001↓↓
        001010
          1001↓↓
          001110
            1001↓
            01111
             1001↓
             01101
              1001
              0100
```

Remainder = 100

Since the remainder is non-zero value, it indicates that the received data contains error.

3. Checksum

3. Checksum…

- **Checksum** is the error detection method used by upper layer protocols and is considered to be **more reliable** than Parity and CRC.

- **For error detection by checksums, data is divided into fixed sized frames or segments.**

➢ **Sender's End** – The sender adds the segments using 1's complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.

➢ **Receiver's End** – The receiver adds the incoming segments along with the checksum using 1's complement arithmetic to get the sum and then complements it.

If the result is zero, the received frames are accepted; otherwise they are discarded.

3. Checksum…

Example: Suppose that the sender wants to send 4 frames each of 8 bits, where the frames are 11001100, 10101010, 11110000 and 11000011.

- The sender adds the bits using 1s complement arithmetic. While adding two numbers using 1s complement arithmetic, if there is a carry over, it is added to the sum.

- After adding all the 4 frames, the sender complements the sum to get the checksum, 11010011, and sends it along with the data frames.

11001100	**10101010**	**11110000**	**11000011**	**11010011**

- The receiver performs 1s complement arithmetic sum of all the frames including the checksum. The result is complemented and found to be 0. Hence, the receiver assumes that no error has occurred.

Sender's End

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011

Receiver's End

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011
Sum:	11111111
Complement:	00000000

Hence accept frames.

Practice Questions

1. The message 1101011011 is to be transmitted using the generator polynomial x4+x+1 to protect it from errors. Find the final data to be transmitted after performing CRC.

2. Original data is 10110011 10101011 01011010 11010101. Perform 2D parity and compute the codeword.

3. Original data is 10110011 10101011 01011010 11010101. Find out the checksum to detect error.

Error Correction

Error Correction

- Error-correcting codes (ECC) are a sequence of numbers generated by specific algorithms for detecting and removing errors in data that has been transmitted over noisy channels.

- **Hamming Code** is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. It was developed by R.W. Hamming for error correction.

Hamming Code

Encoding a message by Hamming Code:

- The procedure used by the sender to encode the message encompasses the following steps –

  Step 1 – Calculation of the number of redundant bits.

  ($2^r \geq m+r+1$)

  Step 2 – Positioning the redundant bits.

  (*r* redundant bits placed at bit positions of powers of 2)

  Step 3 – Calculating the values of each redundant bit.

  (bit positions whose binary representation includes a 1 in the i^{th} position)

- Once the redundant bits are embedded within the message, this is sent to the user.

Hamming Code…

Decoding a message in Hamming Code:

- Once the receiver gets an incoming message, it performs parity checking.
- Parity bits are calculated based upon the data bits and the redundant bits using the same rule as during generation of c1,c2 ,c3 ,c4 etc. Thus

  c1 = parity(1, 3, 5, 7, 9, 11 and so on)

  c2 = parity(2, 3, 6, 7, 10, 11 and so on)

  c3 = parity(4-7, 12-15, 20-23 and so on)

- The decimal equivalent of the parity bits binary values is calculated. If it is 0, there is no error. Otherwise, the decimal value gives the bit position which has error.

Hamming Code Example

1. a) Shradha is transmitting data-1101 to her friend. For security reasons, Shradha wishes to send data by implementing the Hamming Code technique so as to make sure of error control. Help Shradha in performing the above (Assume Even parity).

   b) Also show the error detection process done by Shradha's friend for the following cases.

   Case1: Received data is correct.

   Case2: Second bit from left is changed.

Hamming Code : Solution

a) Given D = 1101

Parity = Even

$m = 4$ (no. of data bits)

Step 1:

$$2^r \geq m+r+1$$

$r=1 \Rightarrow 2^1 \geq 4+1+1 \Rightarrow 2 \geq 6$ ✗

$r=2 \Rightarrow 2^2 \geq 4+2+1 \Rightarrow 4 \geq 7$ ✗

$r=3 \Rightarrow 2^3 \geq 4+3+1 \Rightarrow 8 \geq 8$ ✓

$$\boxed{\therefore r = 3}$$

Total bits $= m+r = 4+3 = 7$ bits

Step 2:

7	6	5	4	3	2	1
D_7	D_6	D_5	r_4	D_3	r_2	r_1
1	1	0	r_4	1	r_2	r_1

r bits to be placed in positions of powers of 2 $(2^0, 2^1, 2^2)$

Step 3:

$$\left.\begin{aligned} r_1 &= (1,3,5,7) = (r_1,1,0,1) \Rightarrow r_1 = 0 \\ r_2 &= (2,3,6,7) = (r_2,1,1,1) \Rightarrow r_2 = 1 \\ r_4 &= (4,5,6,7) = (r_4,0,1,1) \Rightarrow r_4 = 0 \end{aligned}\right\} \text{Even Parity}$$

Hamming Code = | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

Hamming Code : Solution

b) Case 1 : Received data is correct

Received data : 1 1 0 0 1 1 0

7	6	5	4	3	2	1
1	1	0	0	1	1	0

$r_1 = (1,3,5,7) = (0,1,0,1) = \text{Even} \checkmark \Rightarrow C_1 = 0$

$r_2 = (2,3,6,7) = (1,1,1,1) = \text{Even} \checkmark \Rightarrow C_2 = 0$

$r_4 = (4,5,6,7) = (0,0,1,1) = \text{Even} \checkmark \Rightarrow C_4 = 0$

$C_4 C_2 C_1 = 000 = 0 \text{ (Decimal)}$

Since the checkbits $(C_4 C_2 C_1)$ value is 0, it indicates received data is correct

Case 2 : Second bit from left is changed

Received data : 1 0 0 0 1 1 0

↑ 2nd bit

7	6	5	4	3	2	1
1	0	0	0	1	1	0

$r_1 (1,3,5,7) = (0,1,0,1) = \text{Even} \checkmark \Rightarrow C_1 = 0$

$r_2 (2,3,6,7) = (1,1,0,1) = \text{odd} \times \Rightarrow C_2 = 1$

$r_4 (4,5,6,7) = (0,0,0,1) = \text{odd} \times \Rightarrow C_4 = 1$

$C_4 C_2 C_1 = 110 = 6 \text{ (decimal)}$

It indicates that 6th bit is an error bit.

$\boxed{\text{Correct data} = 1\,1\,0\,0\,1\,1\,0}$

SELF-ASSESSMENT QUESTIONS

1. What is the codeword for the dataword 1011 when even parity is used?

(a) 10110
(b) 10111
(c) 10011
(d) 10010

2. What is the drawback of simple parity method?

(a) Detects only odd no. of errors
(b) Detects only even no. of errors
(d) Neither a nor b

(a) Dataword
(b) Redundant bits
(c) Dataword + Redundant
(d) bits

SELF-ASSESSMENT QUESTIONS

1. What is the codeword for the dataword 1011 when even parity is used?

(a) 10110
(b) 10111
(c) 10011
(d) 10010

2. What is the drawback of simple parity method?

(a) Detects only odd no. of errors
(b) Detects only even no. of errors

(d) Neither a nor b

(a) Dataword
(b) Redundant bits
(c) Dataword + Redundant
(d) bits

SUMMAR

- Types of Errors
- Error Detection Methods
- Error Correction Methods

TERMINAL QUESTIONS

1. The message 1101011011 is to be transmitted using the generator polynomial x4+x+1 to protect it from errors. Find the final data to be transmitted after performing CRC.

2. Original data is 10110011 10101011 01011010 11010101. Perform 2D parity and compute the codeword.

3. Original data is 10110011 10101011 01011010 11010101. Find out the checksum to detect error.

4. Similarly, if Shradha is receiving a 7-bit hamming code 1011101, assume Even parity and state whether received data is correct or not. If not locate the error bit?

REFERENCES FOR FURTHER LEARNING OF TH

Reference Books:

1. A.S. Tanenbaum, David J. Wetheral “Computer Networks” Pearson, 5th Edition.
2. Kurose, J and Ross, K Computer Networking: A Top-Down Approach Addison-Wesley- 6th edition.

Sites and Web links:

1. https://www.tutorialspoint.com/data_communication_computer_network/error_detection_and_ correction.htm

THANK YOU

Team –Network Protocols & Security