WELCOME TO OUR PRESENTATION

ERROR DETECTION AND CORRECTION

COURSE TEACHER: MR. SHAHADAT HOSSAIN

COURSE CODE: CSE-225

COURSE TITLE: DATA COMMUNICATION

SECTION: 61_X

DEPARTMENT OF CSE

MEET THE TEAM

MD. NEROB HAWLADER
221-15-5239

AMIT KUMAR GHOSH
221-15-4650

SADIQUL HAQUE SADIB
221-15-4863

MD. JAHIDUL ISLAM
221-15-4885

JARIN TASNIM SHOWRIN
221-15-4547

LAMIA AKTER LIZA
221-15-5100

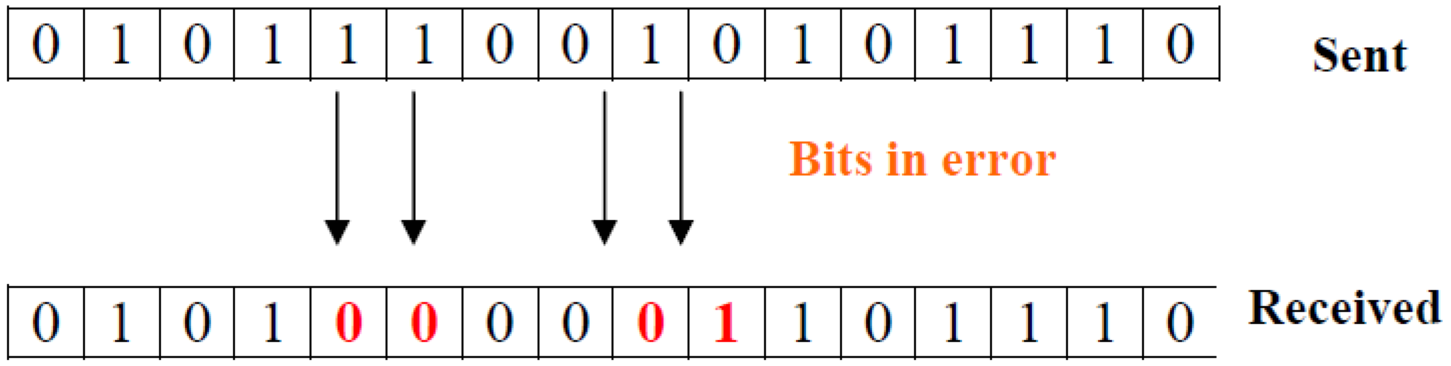
SIAM AKTER MIM
221-15-5924

TANJID HASAN TURKY
221-15-5916

TABLE OF CONTENTS

WHAT IS ERROR?

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a **0** bit may change to **1** or a **1** bit may change to **0**.

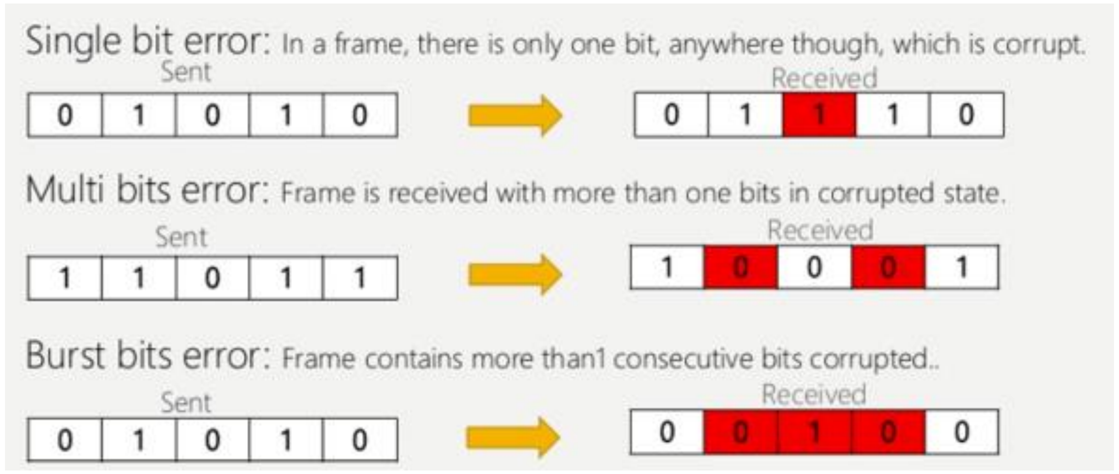


TYPES OF ERROR

1.**Single bit:** A single bit gets corrupted during transmission .

2.**Multi bit:** Bore than one bit gets corrupted during transmission.

3.**Brust bit:** A number of continued bits get corrupted during transmission.


Single bit error: In a frame, there is only one bit, anywhere though, which is corrupt.
Sent
0 1 0 1 0
Received
0 1 1 1 0
Multi bits error: Frame is received with more than one bits in corrupted state.
Sent
1 1 0 1 1
Received
1 0 0 0 1
Burst bits error: Frame contains more than1 consecutive bits corrupted..
Sent
0 1 0 1 0
Received
0 0 1 0 0

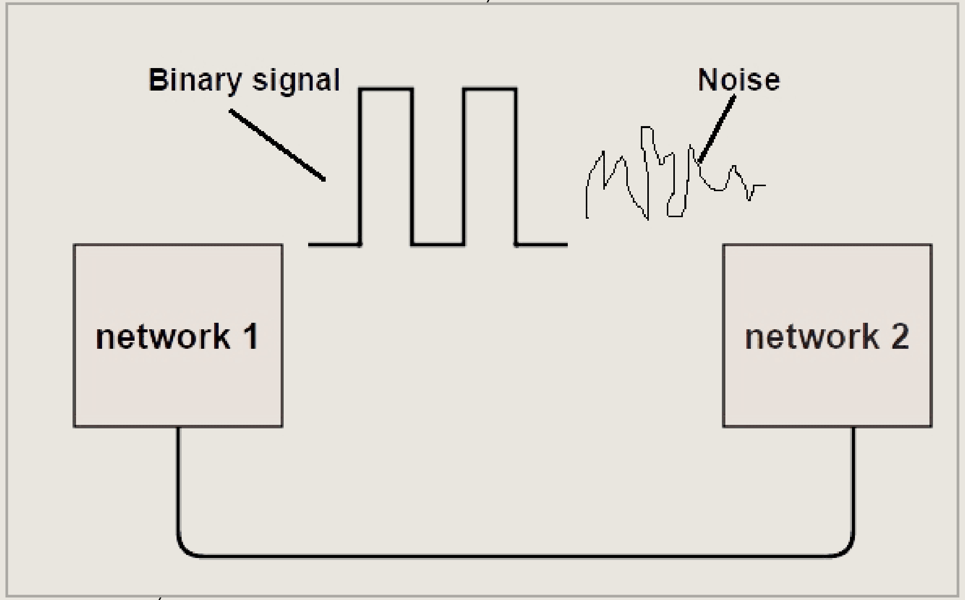
ERROR DETECTION & CORRECTION

Error Detection:

Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

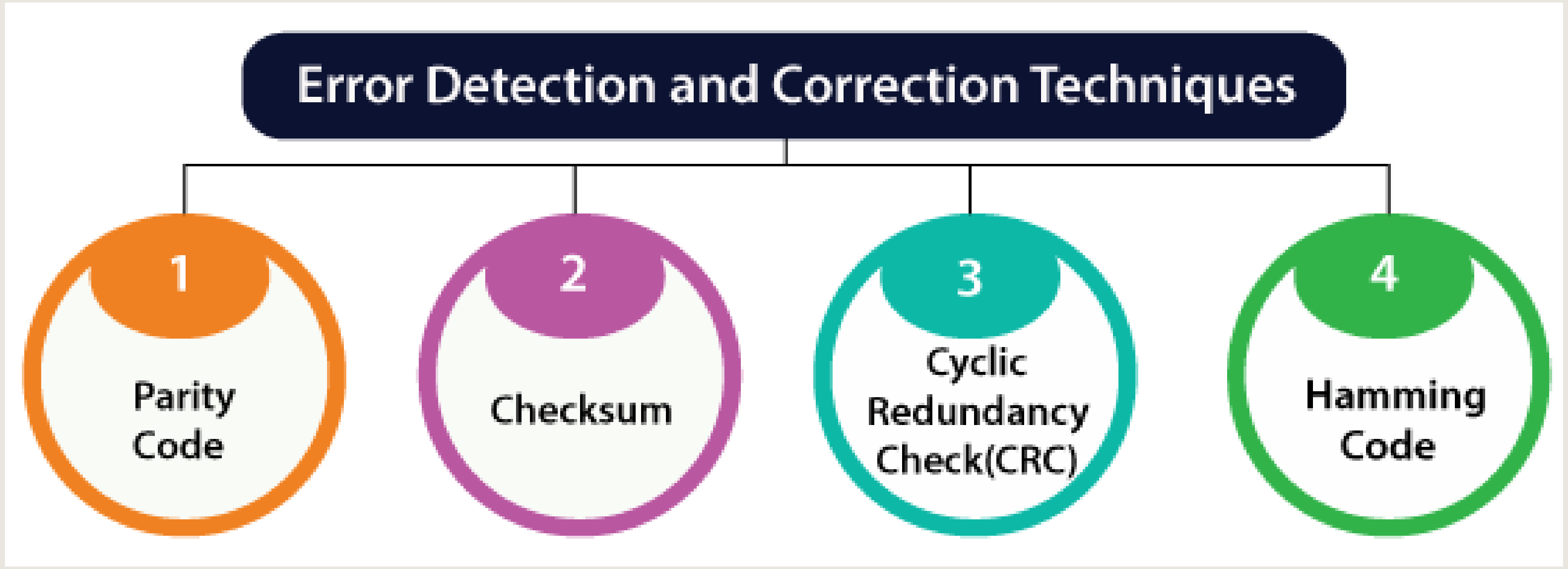
Error Correction:

Error correction is the detection of errors and reconstruction of the original, error-free data.



ERROR DETECTION TECHNIQUES

1. Hamming code
2. Checksum
3. Cyclic Redundancy check


Error Detection and Correction Techniques
1
Parity Code
2
Checksum
3
Cyclic Redundancy Check(CRC)
4
Hamming Code

HAMMING CODE

What is Hamming Code?

Hamming code is a liner code that is useful for error detection up to two immediate bit errors. It is capable of single-bit errors.

How to check if there’s any Error using Hamming Code?

To solve the data bit issue with the hamming code method, some steps need to be followed:

- ❖ **Step 1** - The position of the data bits and the number of redundant bits in the original data. The number of redundant bits is deduced from the expression [2^r >= d+r+1].
- ❖ **Step 2** - Fill in the data bits and redundant bit, and find the parity bit value using the expression [2^p, where, p - {0,1,2, …… n}].

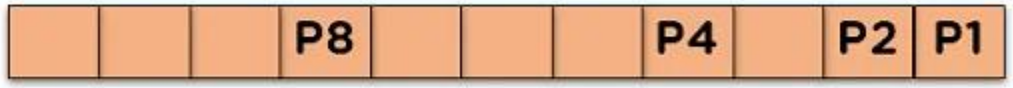
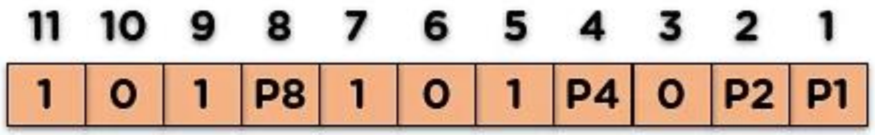


Figure 1

HAMMING CODE

❖ **Step 3** - Fill the parity bits and data bits obtained from the original data.



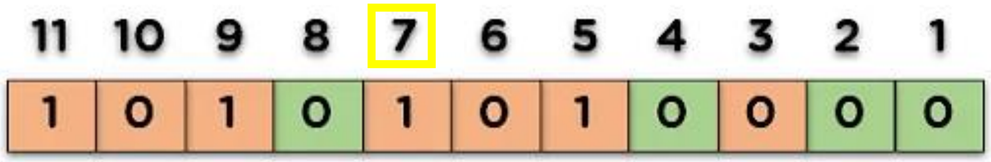
11	10	9	8	7	6	5	4	3	2	1
1	0	1	P8	1	0	1	P4	0	P2	P1

Figure 2

❖ **Step 4** - Check the received data using the parity bit and detect any error in the data, and in case damage is present, use the parity bit value to correct the error.

Next, we will solve an example using hamming code to clarify any doubts regarding the working steps.

Original Data:



11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	1	0	1	0	0	0	0

HAMMING CODE

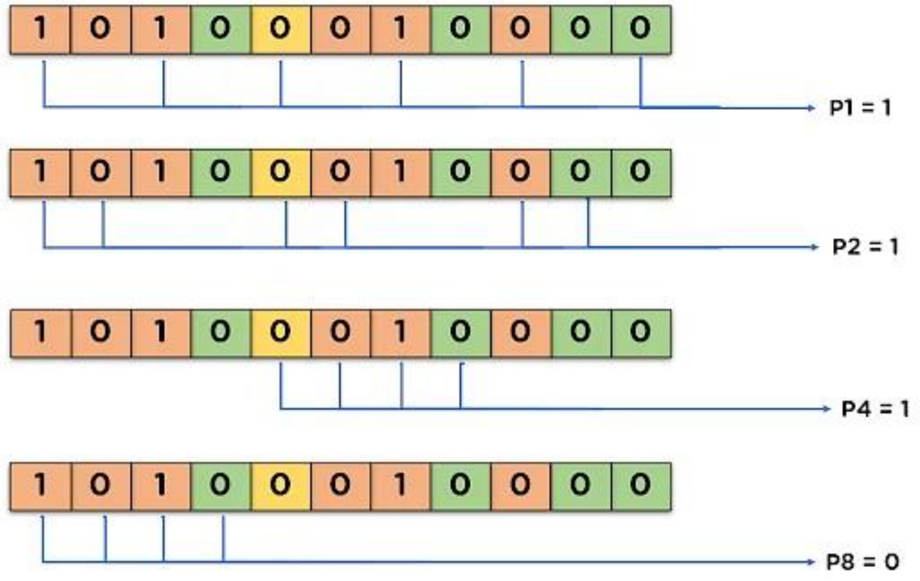
Error Detecting and Correction of the Data Received:

Assume that during transmission, the data bit at position 7 is changed from 1 to 0. Then by applying the parity bit technique, we can identify the error:

- ❖ **Odd Parity bits** - In this parity type, the total number of 1’s in the data bit should be odd in count, then the parity value is 0, and the value is 1.
- ❖ **Even Parity bits** - In this parity type, the total number of 1’s in the data bit should be even in count; then the parity value is 0, and the value is 1.

$_{p1}$ = parity(1, 3, 5, 7, 9, 11 and so on)
$_{p2}$ = parity(2, 3, 6, 7, 10, 11 and so on)
$_{p3}$ = parity(4-7, 12-15, 20-23 and so on)

Here:
P1=1, P2=1, P4=1, P8=0
0111 = 7 (binary to decimal conversion)
7 number bit has error.



CHECK-SUM

Checksums:

This is a block code method where a checksum is created based on the data values in the data blocks to be transmitted using some algorithm and appended to the data. When the receiver gets this data, a new checksum is calculated and compared with the existing checksum. A non-match indicates an error.

Error Detection by Checksums:

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

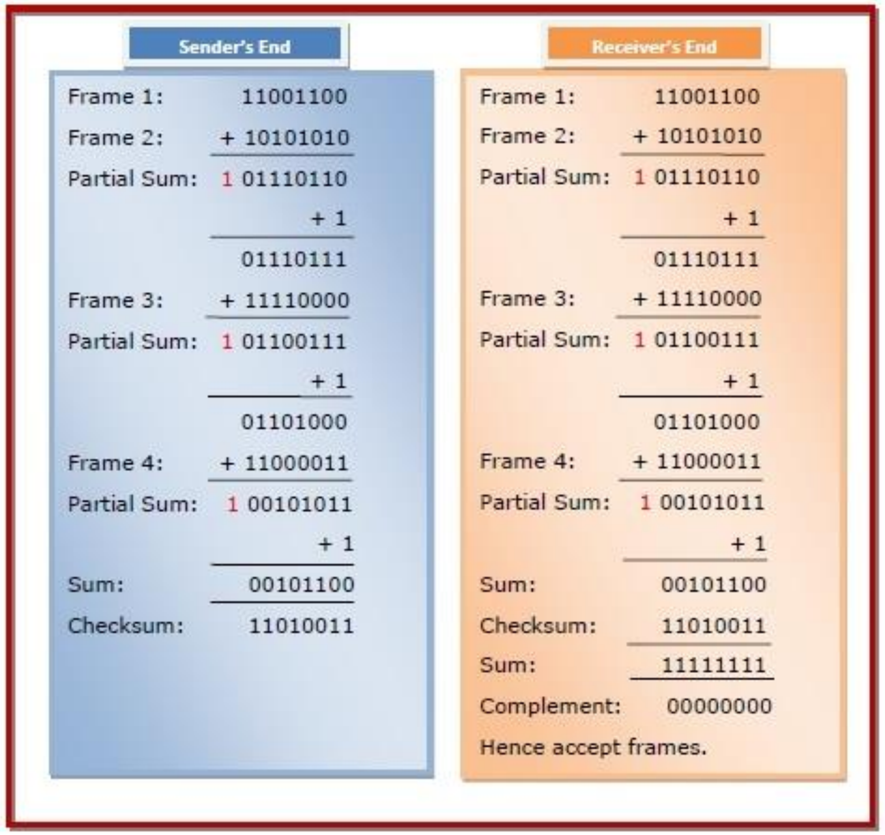
- ❖ **Sender’s End** – The sender adds the segments using 1’s complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
- ❖ **Receiver’s End** – The receiver adds the incoming segments along with the checksum using 1’s complement arithmetic to get the sum and then complements it.

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

CHECK-SUM

Example:

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



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.

CYCLIC REDUNDANCY CHECK

Cyclic Redundancy Check (CRC):

The CRC is a network method designed to detect errors in the data and information transmitted over the network. This is performed by performing a binary solution on the transmitted data at the sender's side and verifying the same at the receiver's side.

Sender Side:

- ❖ The first step is to add the no. of zeroes to the data to be sent, calculated using k-1 (k - is the bits obtained through the polynomial equation.)
- ❖ Applying the Modulo Binary Division to the data bit by applying the XOR and obtaining the remainder from the division
- ❖ The last step is to append the remainder to the end of the data bit and share it with the receiver.

CYCLIC REDUNDANCY CHECK

Receiver Side (Checking for errors in the received data):

To check the error, perform the Modulo division again and check whether the remainder is 0 or not,

- ❖ If the remainder is 0, the data bit received is correct, without any errors.
- ❖ If the remainder is not 0, the data received is corrupted during transmission.

Example -

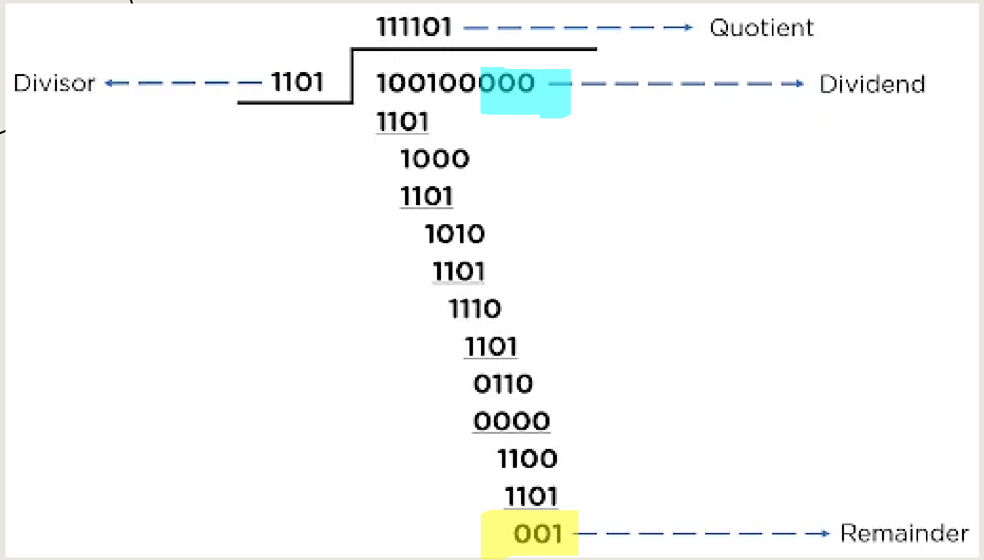
The data bit to be sent is [100100], and the divisor is 1101.

Data bit - 100100
Divisor (k) - 1101
Appending Zeros - (k-1) > (4-1) > 3
Dividend - 100100000

CYCLIC REDUNDANCY CHECK

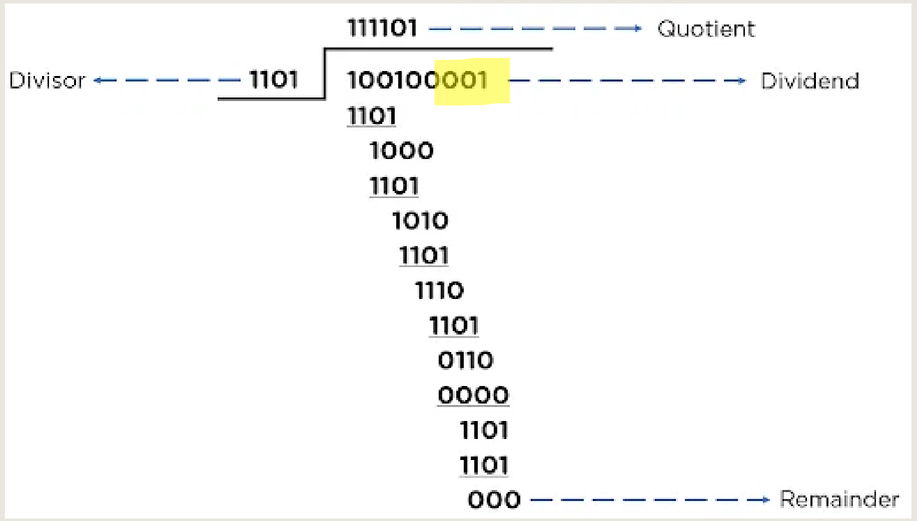
Original Data bit – 100100
Divisor (k) – 1101. Length=4
Appending Zeros - (k-1) = (4-1) > 3

Sender Side:



```
          111101
     1101 100100000
          1101
           1000
           1101
            1010
            1101
             1110
             1101
              0110
              0000
               1100
               1101
                001
```

Receiver Side:



```
          111101
     1101 100100001
          1101
           1000
           1101
            1010
            1101
             1110
             1101
              0110
              0000
               1101
               1101
                000
```

The Obtained remainder is [000], i.e., zero, which according to the CRC method, concludes that the data is error-free.

THANK YOU

Reference:

1. https://www.simplilearn.com/tutorials/networking-tutorial/what-is-cyclic-redundancy-check#:~:text=The%20CRC%20is%20a%20network,same%20at%20the%20receiver%27s%20side.
2. https://www.guru99.com/hamming-code-error-correction-example.html
3. https://www.slideshare.net/BadrulAlam16/error-detection-and-correction-presentation