```
        11010011
1001 ) 11001001000
       1001
       01011
        1001
        001000
          1001
          0001100
             1001
             01010
              1001
              0011
```

Length of CRC ⇒ 3
$\boxed{CRC = 011}$

Reciever Side →

Data Transmitted data

$\boxed{11001001}\boxed{011}$

```
        1100011
1001 ) 11001001011
       1001
       01011
        1001
        001000
          1001
          0001101
             1001
             01001
              1001
              0000
```

* Burst Error length = polynomial degree tk ki level ki error ko ditect kr skti hai CRC.

* Protocol decides the divisor. Normally, the divisor is a polynomial expression.

* Whatever recieved as data (data + CRC) at Reciever side; reciever again perform binary devision on recieved data by the divisor. Then if remainder is all 0's then it accepts.

Question: Suppose we want to transmit the message 11001001 and protect it from errors using the CRC polynomial x^3+1. Use polynomial long division to determine the message that should be transmitted. Corrupt the left-most third bit of the transmitted message and show that the error is detected by the reciever using CRC technique.

$\Rightarrow x^3+1 \Rightarrow x^3 + \qquad 1x^0$

$\quad\ \ x^2 + x^1 + x^0$

$\quad 1 \quad 0 \quad 0 \quad 1$ Divisor.

& L = 4, bits to append = 3

$\Rightarrow$ 1001) 11001001

(18)

⇒ Find the CRC for 1110010101 with the divisor $x^3 + x^2 + 1$?

⇒ Divisor :

$x^3 + x^2 + 1$ OR $x^7 + x^5 + x^3 + x + 1$

$x^7\ x^6\ x^5\ x^4\ x^3\ x^2\ x\ 1$

$\boxed{1\ 1\ 1}$ → Divisor $\boxed{1\ 0\ 1\ 0\ 1\ 0\ 1\ 1}$ → Divisor

$\boxed{L=3}$

bits to append = 2

```
         100
Now; 111 ) 1110010101 00
           111
           0000
            000
           0000
            000
            0001
```

4. Cyclic Redundancy Check (CRC):

Note: Exclusive OR (XOR): a logical operation that is true if and only if its input differ.

A	B	Q
0	1	1
0	0	0
1	0	1
1	1	0

• CRC Generation at sender side:

1. Find the length of divisor 'L'.
2. Append 'L-1' bits to the original message.
3. Perform binary division operation.
4. Remainder of the division = CRC

Note: CRC must be of (L-1) bits.

⇒ Find the CRC for the data blocks 100100 with the divisor 1101?

⇒ L = 4, bits to append = (L-1) = 3 = 000

```
=)           111101
    1101 ) 1001|00|000
           1101         XOR
No effect <-- 0|1000
              1101      XOR
           <- 0|1010
              1101      XOR
           <- 0|1110
              1101
           <- 0|0|0110
                 0000
                           1100
                           1101
                           001   CRC
```

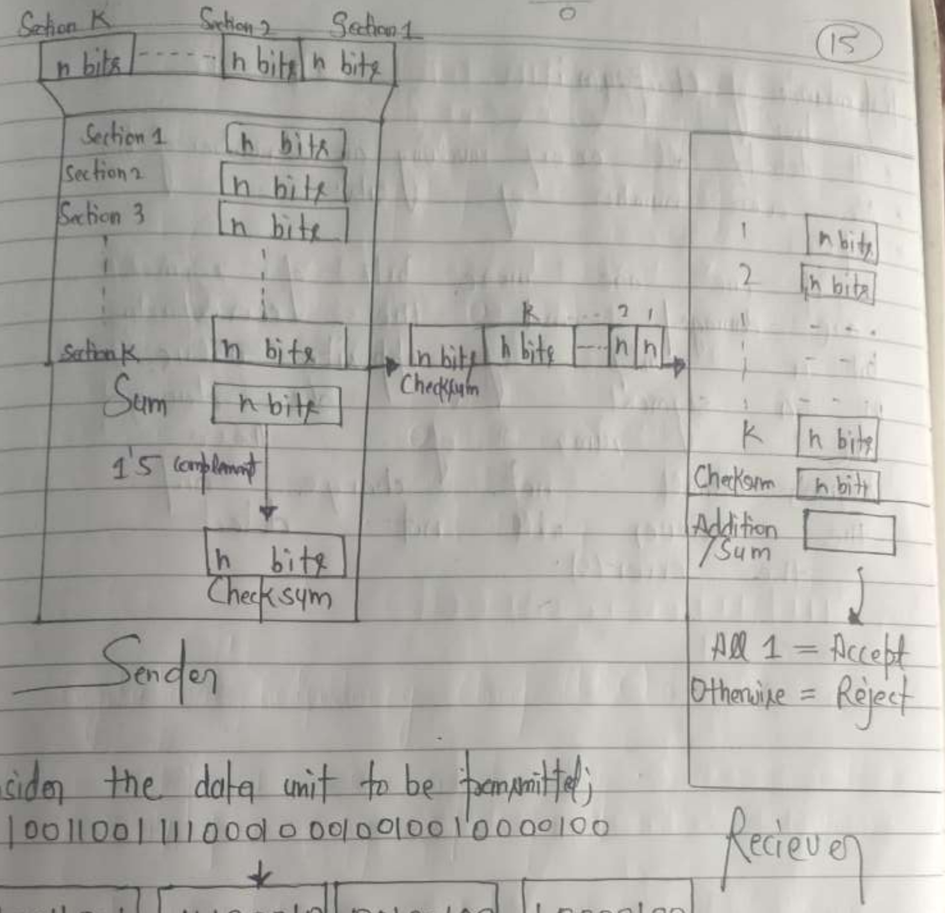
Data Recieved
100100|001
↓
CRC

Performance of Checksum :-

* It detects all errors involving an odd number of bits.
* Also even number of bits.
* But

if one ore more bits of segment are damaged and the corresponding bit or bits of opposite value in a segment are also damaged the sums of those columns will not change and the reciever will not detect the error(s).

15



Consider the data unit to be transmitted;

10011001 11100010 00100100 10000100

10011001	11100010	00100100	10000100
8 bit	8 bits	8 bits	8 bit

```
   10011001
   11100010
   00100100
   10000100
 ----------
 1000100011
 carry  10
 ----------
   00100101   Sum
   1's complement
   11011010
```

Checksum	8	8	8	8
11011010	10011001	11100010	00100100	10000100
	D	C	B	A

```
          10000100
          00100100
          11100010
          10011001
 Checksum 11011010
 ----------------
 Add    1011111101
                10
 ----------------
          11111111
```

↳ ACCEPTED

Direction of Movement → A

LRC	D	C	B	A
10101010	10101001	00111001	11011101	11100111

LRC | Data

Performance of LRC:

* LRC increases the likelihood of detecting burst errors.
* But if two bits in one data unit are damaged and two bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

3. Check Sum :→ the Check + Sum

Sender side = Checksum Creation
Reciever side = Checksum Validation

1. Operation at sender side:

↳ Break the original message into 'K' number of blocks with 'n' bits in each block.

↳ Sum all the 'K' data blocks.

↳ Add the carry to the sum; if any.

↳ Do 1's complement to the sum = Checksum

Performance of VRC : • It can detect single bit error mostly
• It can detect burst error only if the number of $\boxed{1's}$ errors is $\boxed{odd}$.

Sender: 11100001 → Transmision Error 10100001 → Reciever rejects this data.

Sender: 11100001 → Transmission Error 10100101 → Reciever accepts this data despite being currupt data

2. Longitudinal Redundancy Check (LRC) :

• Also called "Two Dimensional parity". Because, a block of bits is organized in rows & columns.

• The pairity bit is calculated for each column and sent along with data.

• The block of pairity act as redundant bits.

Example: Find the LRC for the data blocks 11100111 (A), 11011101 (B), 00111001 (C), 10101001 (D) and determine data that is transmitted;

→ We know;

odd no. of 1's	1
Even no. of 1's	0

1	1	1	0	0	1	1	1	A
1	1	0	1	1	1	0	1	B
0	0	1	1	1	0	0	1	C
1	0	1	0	1	0	0	1	D
1	0	1	0	1	0	1	0	LRC

• ~~Error Correction~~:
~~Error Detection~~

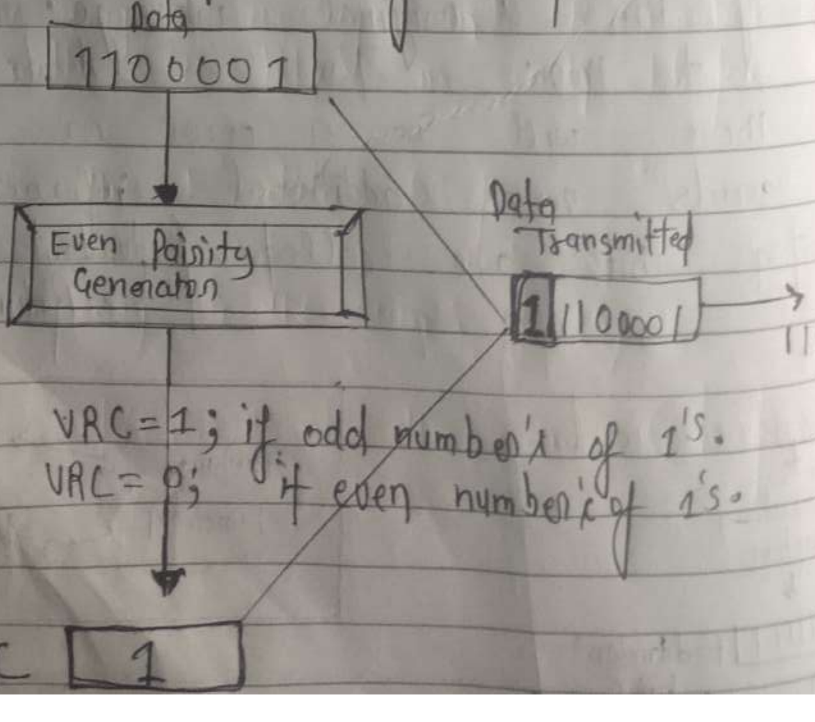
• Error Correction: When data arrives at Reciever side. It's Checker algorithm or checker function detect the data. It detect which part is corrupted; if possible then it correct it, otherwise it request sender to retransmit the data.

• Error detection Techniques:

1. Vertical Redundancy Check (VRC)
2. Longitudinal Redundancy Check (LRC)
3. Checksum
4. Cyclic Redundancy Check (CRC)

1. Vertical Redundancy Check (VRC) :-

It is also called "Pairity Check".



This counts number of 1's in data. And add VRC

VRC = 1; if odd number's of 1's.
VRC = 0; if even number's of 1's.

2. Error - Control

(Error control happens node to node.)

Error detection — Error - correction

- Error-control techniques are implemented either at the data link layer or the Transport layer of the OSI model.

- Types of Error: 1. Bit Error = Single bit Error
  2. Burst Error = more than one bits are corrupted

Sent
0 0 1 0 1 0 [1] 0 0 0
(corrupt)
0 0 1 0 1 0 [0] 0 0 0
Recieved

Length of burst error
8 bit
Sent
0 1 0 0 0 [0] 1 [0] 0 [0] 1 [0] 0 0 1 1
0 1 0 0 0 1 0 1 1 1 1 1 1 0 0 1 1
Recieved

- Error detection means to decide wheather recieved data is correct or not without having a copy of the original message. Generally, reciever do this. To decide this wheather there will be a error or not sender sends some additional information. These extra bits are called Redundant bits → Redundancy Check