DATALINK LAYER

Introduction

The Data Link Layer break the bit stream into discrete frames and compute the checksum for each frame. When a Frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from one computed contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

FRAMING METHODS

1. CHARATER COUNT METHOD

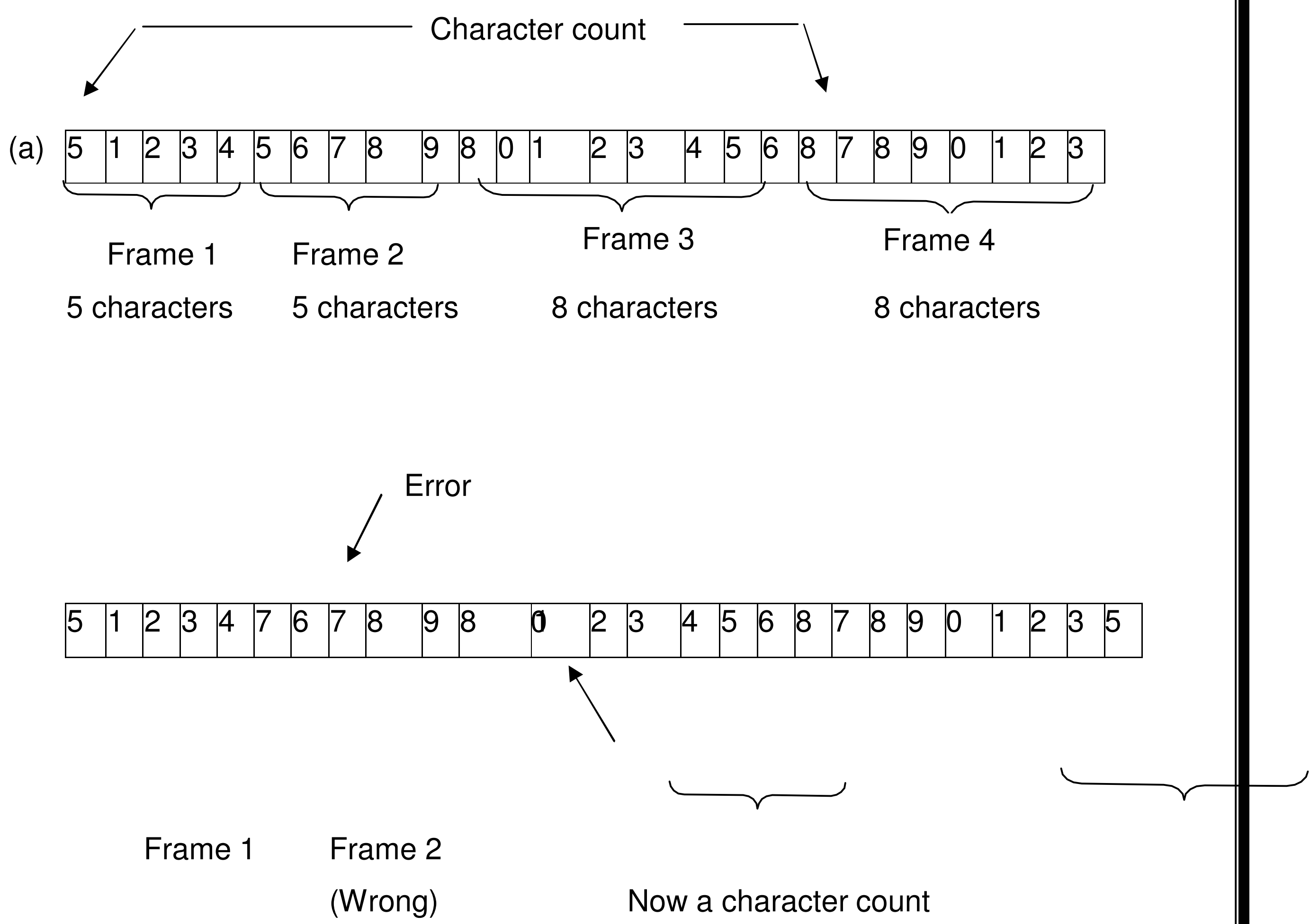
2. STARTING AND ENDING CHARACTERS, WITH CHARATER STUFFING

3. STARTING AND ENDING FLAGS, WITH BIT STUFFING

CHARATER COUNT METHOD:

In this method a field in the header will be used to specify the number of CHARACTERS in the frame. When data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is.

The trouble with this algorithm is that the count can be garbed by a transmission error resulting the destination will get out of synchronization and will be unable to locate the start of the next frame. There is no way of telling where the next frame starts. For this reason this method is rarely used.



A Character Stream (a) Without errors (b) With one error

CHARATER STUFFING METHOD:

In this method each frame will start with a FLAG and ends with a FLAG.

The starting flag is **DLE STX ---- Data Link Escape Start of Text**

The ending flag is **DLE ETX ------- Data link Escape End of Text.**

Ex 1. The given Data ABRFCXDGJHKK12435ASBGXRR

The Data will be sent

DLE STX ABRFCXDGJHKK12435ASBGXRR DLE STX

Ex 2. The given Data **ASHGTRDXZBNHG DLE STX %$#54378**

The data will be sent as

DLE STX ASHGTRDXZBNHG DLE DLE STX %$#54378 DLE ETX

Dis Adv:

1.24 bits are unnecessarily stuffed.

2. Transmission delay.

BIT STUFFING METHOD

In this method every frame will start with a **flag 01111110.**

In the data if there are **FIVE** consecutive ONE 's are there then a ZERO will be stuffed.

Ex. The given data is 011110000111111010100111110 01111101100

The data will be sent as

01111110 01111000011111**0**1101010011111**0**10 011111**0**01100

Stuffed bits

Advantages:

1. Only one bit is stuffed.
2. No transmission delay

ERROR – CORRECTING AND DETECTING CODES

Network designers have developed two basics strategies for dealing with errors. One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been .The other way is to include only enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and have it request a retransmission. The former strategy uses **Error – correcting codes** and the latter uses **Error- detecting codes**.

The **Error – correcting and Error- detecting methods are**

1. PARITY METHOD
2. LRC METHOD (Longitudinal redundancy check)
3. CRC METHOD (Cyclic redundancy check)
4. HAMMING CODE METHOD

PARITY METHOD

- appends a parity bit to the end of each word in the frame

- Even parity is used for asynchronous Transmission

- Odd parity is used for synchronous Transmission

Ex	Character code	even parity	odd parity
1.	1100100	1100100 1	1100100 0
2.	0011000	0011000 0	0011000 1

If one bit or any odd no bits is erroneously inverted during Transmission, the Receiver will detect an error. How ever if two or even no of bits are inverted an undetected error occurs.

Ex 3. The Transmitted data is 10011010. The received data is 11011010.

Let both the transmitter and receiver are agreed on EVEN parity.
Now an error will be detected, since the no of ones received are ODD

4. The Transmitted data is 10011010. The received data is 01011010

The received data is wrong even though the no of ones are EVEN.

Science two bits are inverted error can't be detected.

Longitudinal Redundancy Check(LRC)

The frame is viewed as a block of characters arranged in 2-dimensions. To each character is appended a parity bit. In addition a parity bit is generated for each bit position across all characters i.e., an additional character is generated in which the I^{th} bit of the character is parity bit for the I^{th} bit of all other characters in the block. This can be expressed mathematically using exclusive OR(+) operation. The parity bit at the end of each character of row parity

$R_j = b_{1j} + b_{2j} + \text{------} b_{nj}$

Where R_j=Parity bit of jth character

b_{ij}=ith bit in jth character

This equation generates even parity.

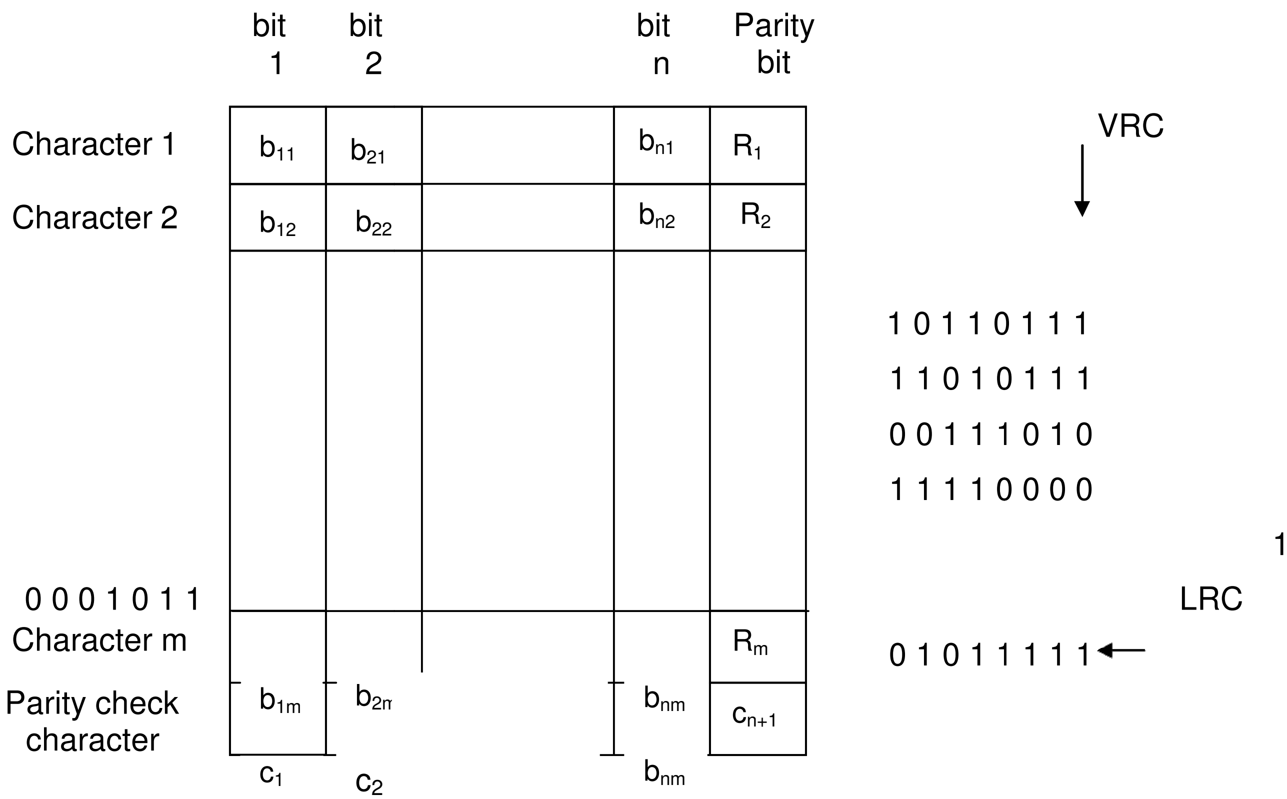
$C_i = b_{i1} + b_{i2} + \text{----} + b_{in}$

Where C_i=ith bit of parity check character

m=number of characters in a frame

In this format the parity bits at the end of each character are referred to as

The Vertical Redundancy Check (VRC) and the Parity check character is referred to as the Longitudinal Redundancy Check (LRC).



CRC Method

1. The frame is expressed in the form of a Polynomial F(x).0 1 1 1 1 1 1 0

2. Both the sender and receiver will agree upon a generator polynomial G(x) in advance.

3. Let 'r' be the degree of G(x).Append 'r' zero bits to the lower – order end of frame now it contains m+r bits.

4. Divide the bit string by G(x) using Mod 2 operation.

5. Transmitted frame [T(x)] = frame + remainder

6. Divide T(x) by G(x) at the receiver end. If the result is a zero, then the frame is transmitted correctly. Ex. Frame: 1101011011

Generator: 10011

Message after appending 4 zero bits: 11010110000

```
          1100001010
10011 | 11010110 110 0 0 0
        10011
        -----
          10011
          10011
          -----
            00001
            00000
            -----
              00010
              00000
              -----
                00101
                00000
                -----
                  01011
                  00000
                  -----
                    10110
                    10011
                    -----
                      01010
                      00000
                      -----
                        10100
                        10011
                        -----
                          01110          Remainder
                          00000
                           1110
```

Transmitted frame: 11010110111110

```
         1100001010
10011 | 11010110 111 1 1 0
        10011
        -----
          10011
          10011
          -----
            00001
            00000
            -----
              00010
              00000
              -----
                00101
                00000
                -----
                  01011
                  00000
                  -----
                    10111
                    10011
                    -----
                      01001
                      00000
                      -----
                        10011
                        10011
                        -----
                          00000        Remainder
                          00000
                           0000
```

Since the remainder is zero there is no error in the transmitted frame.