DATA LINK LAYER

DATA LINK LAYER DESIGN ISSUES

Services Provided to the Network Layer

Framing

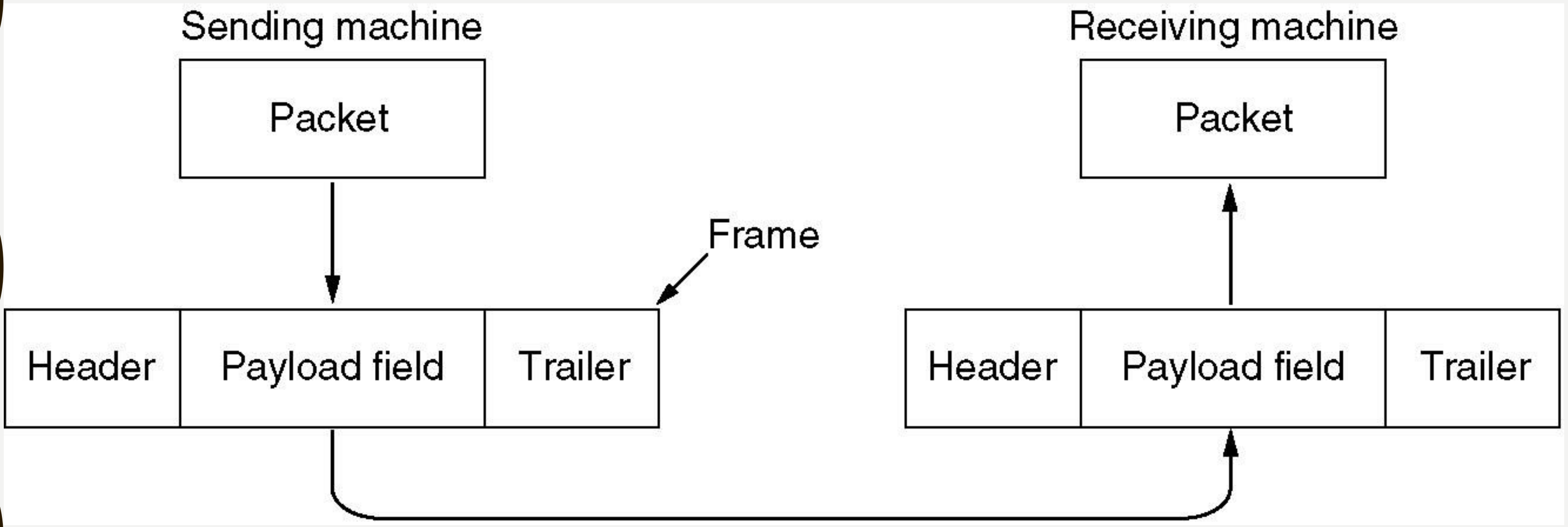
Error Control

Flow Control

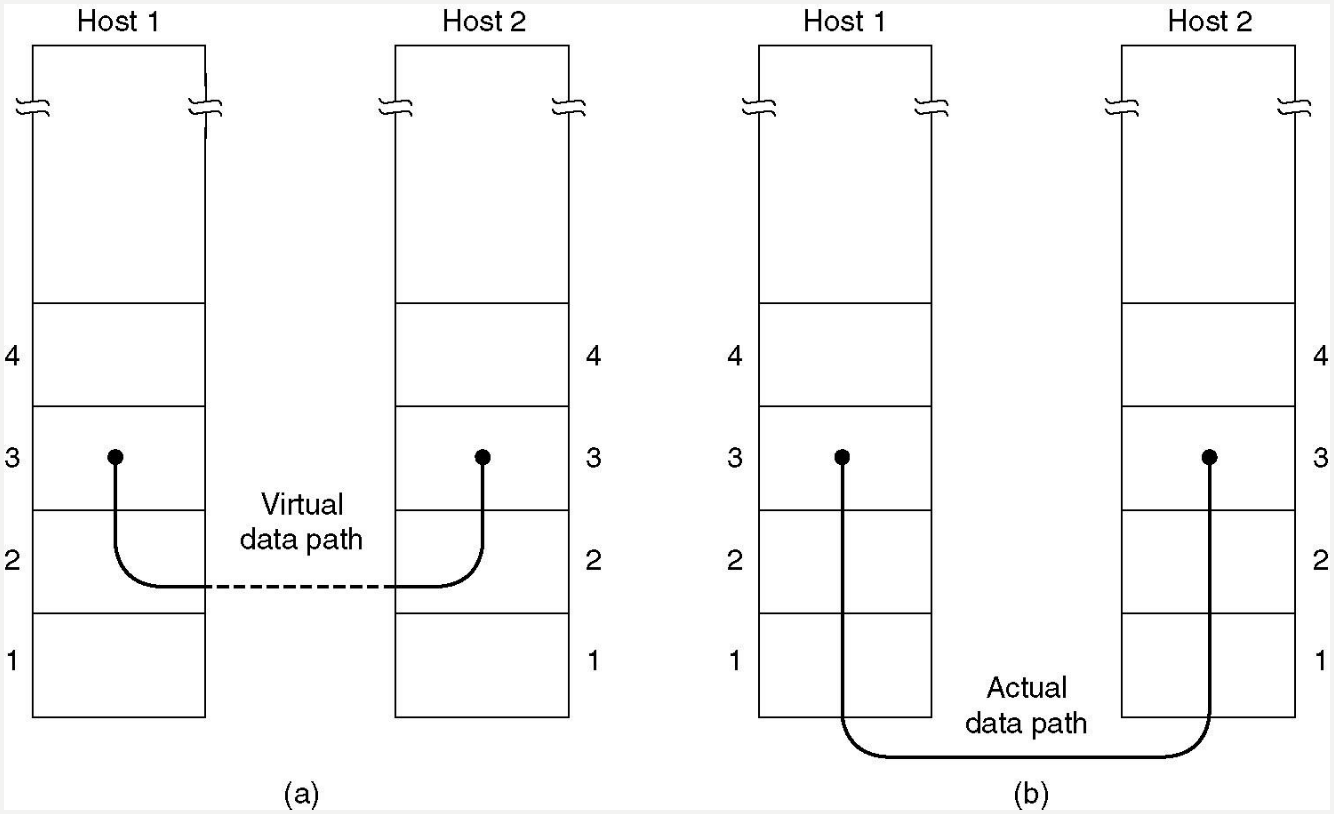
FUNCTIONS OF THE DATA LINK LAYER

- Provide service interface to the network layer
- Dealing with transmission errors
- Regulating data flow
    - Slow receivers not swamped by fast senders

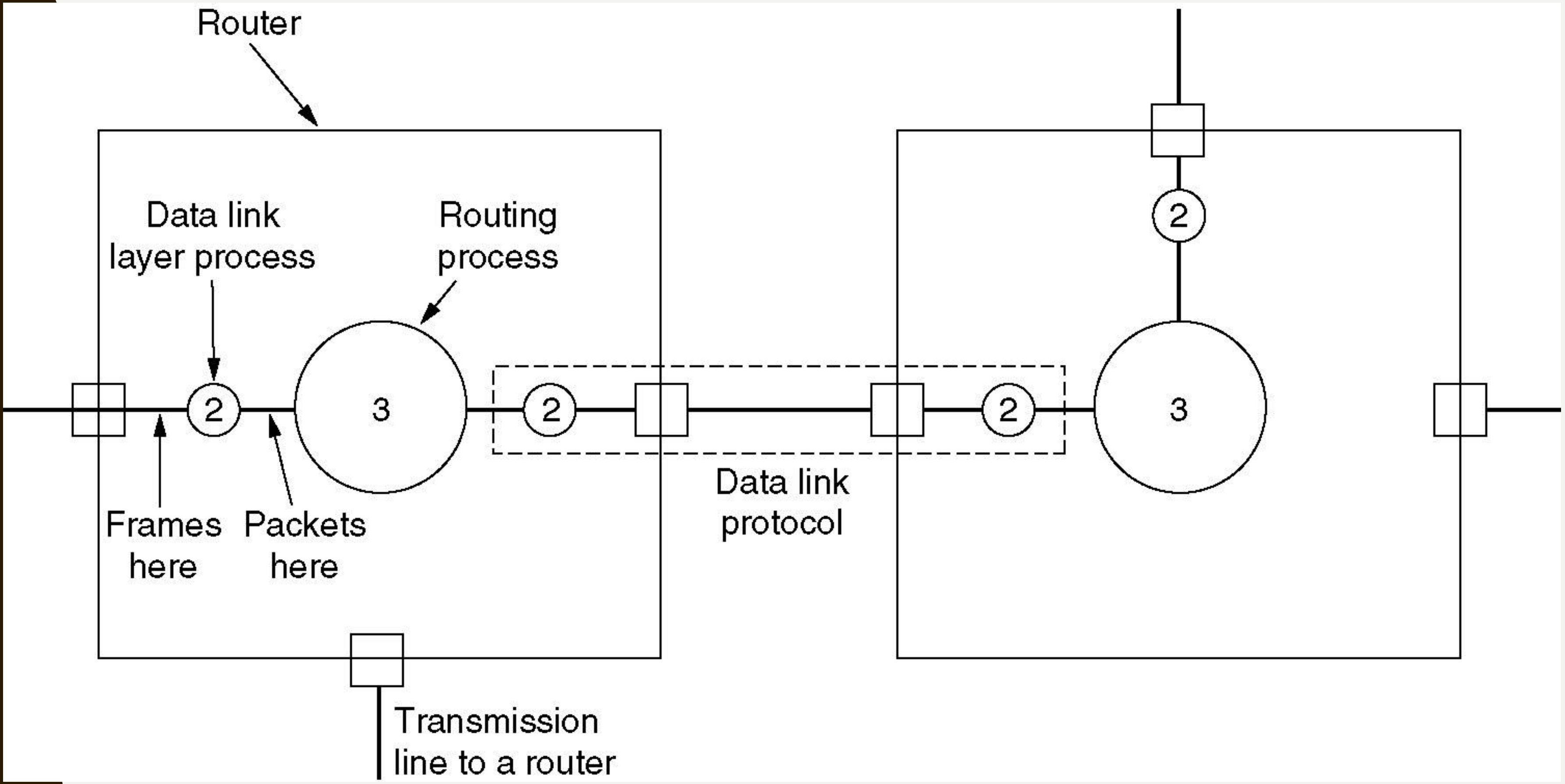
FUNCTIONS OF THE DATA LINK LAYER (2)



SERVICES PROVIDED TO NETWORK LAYER



PLACEMENT OF DLL



TYPES OF SERVICES PROVIDED TO THE NETWORK LAYER

Unacknowledged Connectionless service

Acknowledged Connectionless service

Acknowledged Connection-Oriented service

UNACKNOWLEDGED CONNECTIONLESS SERVICE

- Losses are taken care of at higher layers
- Used on reliable medium like coax cables or optical fiber, where the error rate is low.
- Appropriate for voice, where delay is worse than bad data.

ACKNOWLEDGED CONNECTIONLESS SERVICE

- Useful on unreliable medium like wireless.
- Acknowledgements add delays.
- Adding ack in the DLL rather than in the NL is just an optimization and not a requirement.
  - Leaving it for the NL is inefficient as a large message (packet) has to be resent in that case in contrast to small frames here.
- On reliable channels, like fiber, the overhead associated with the ack is not justified.

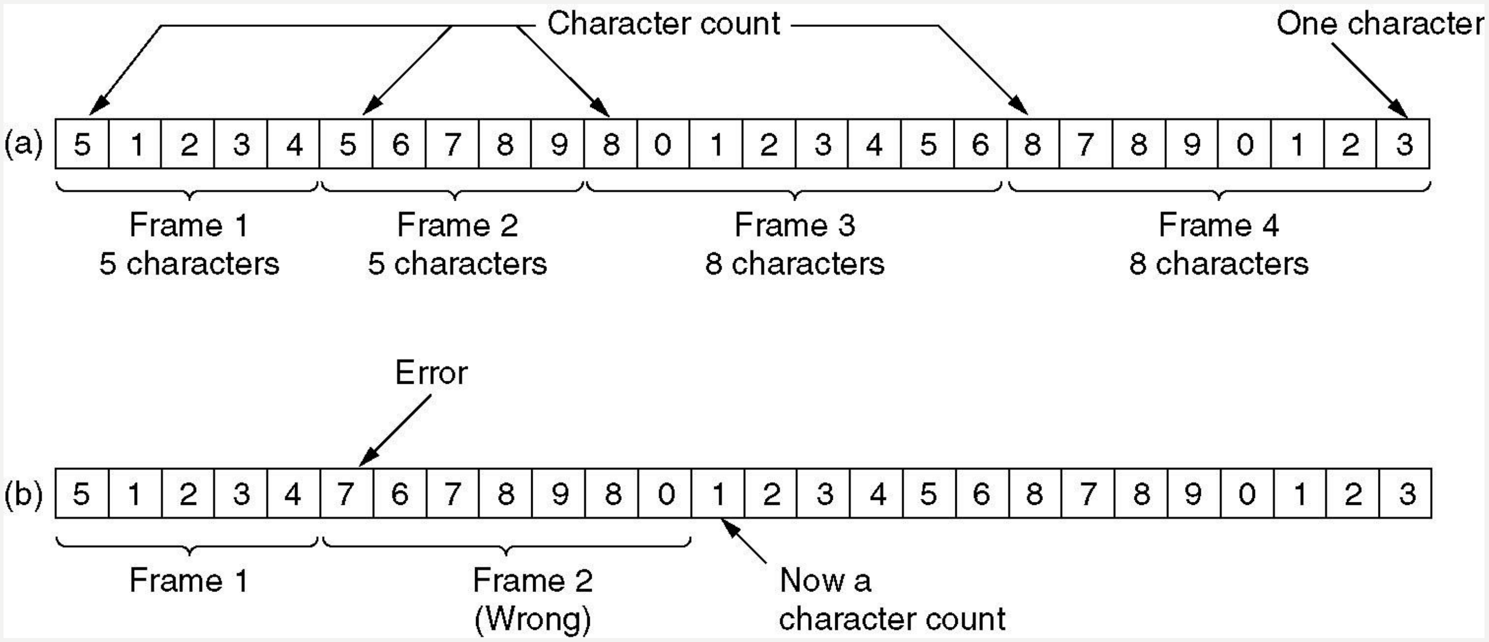
ACKNOWLEDGED CONNECTION-ORIENTED SERVICE

- Most reliable,
- Guaranteed service –
  - Each frame sent is indeed received
  - Each frame is received exactly once
  - Frames are received in order
- Special care has to be taken to ensure this in connectionless services

FRAMING

- Character Count
- Flag bytes with byte stuffing
- Flag bytes with bit stuffing

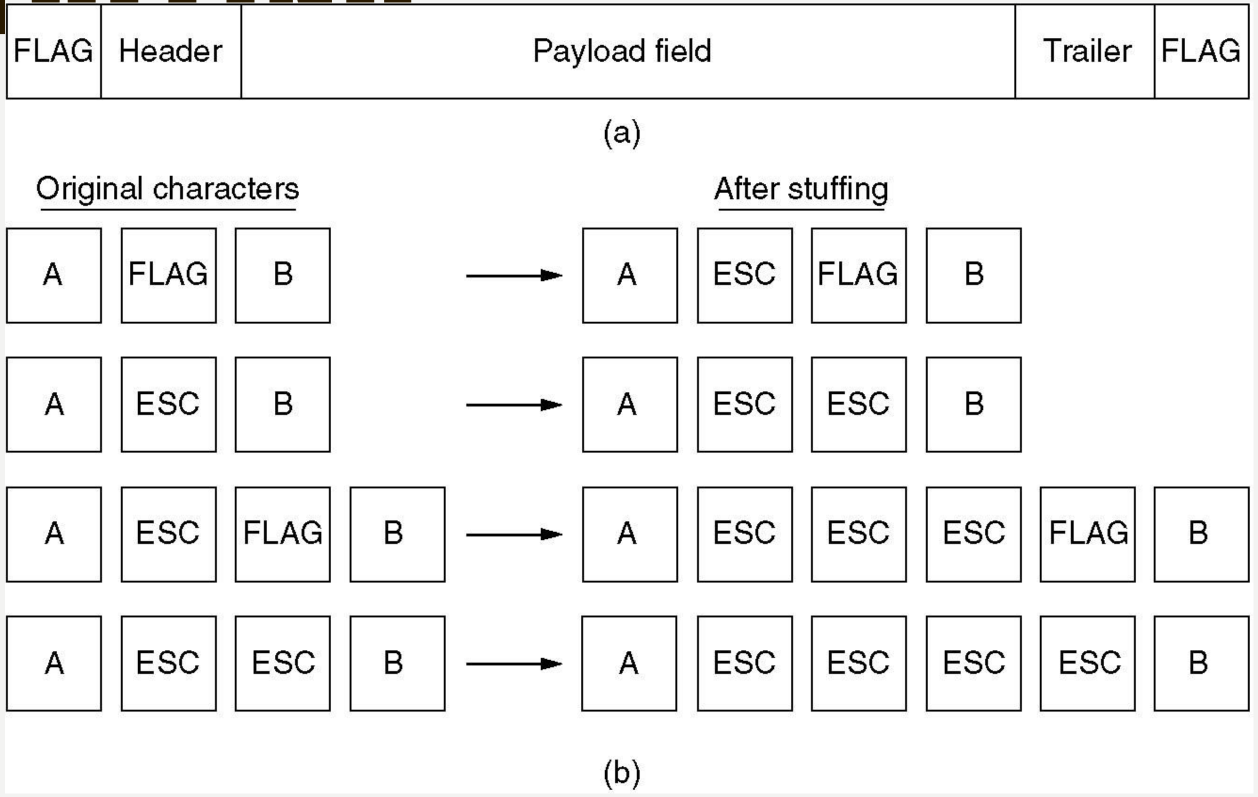
FRAMING WITH CHARACTER COUNT



PROBLEM WITH FRAMING WITH CC

- What if the count is garbled
- Even if with checksum, the receiver knows that the frame is bad there is no way to tell where the next frame starts.
- Asking for retransmission doesn’t help either because the start of the retransmitted frame is not known
- No longer used

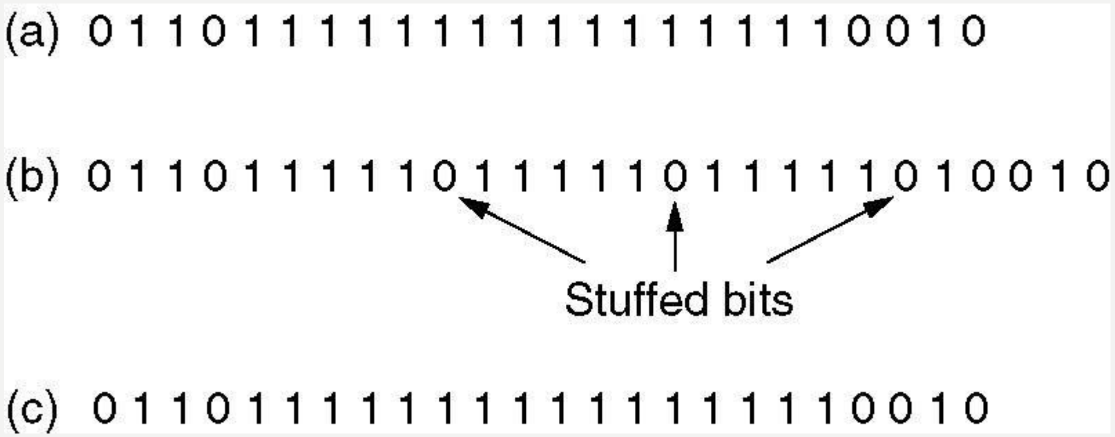
FRAMING WITH BYTE STUFFING



FRAMING WITH BYTE STUFFING

- Problem : fixed character size : assumes character size to be 8 bits : can’t handle heterogeneous environment.

FRAMING WITH BIT STUFFING


(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit stuffing
(a) The original data.
(b) The data as they appear on the line.
(c) The data as they are stored in receiver's memory after destuffing.

ERROR CONTROL

- Positive and Negative feedback
- Timers : what happens when a frame completely vanishes : receiver neither sends a +ack nor –ack … then timer comes to help.
  - It may result in a frame being sent more than once and received more than once :
  - solution : assign sequence numbers to frames

ERROR DETECTION AND CORRECTION

- In some cases it is sufficient to detect an error and in some, it requires the errors to be corrected also. For eg.
  - On a reliable medium : ED is sufficient where the error rate is low and asking for retransmission after ED would work efficiently
  - In contrast, on an unreliable medium : Retransmission after ED may result in another error and still another and so on. Hence EC is desirable.

HAMMING CODES : FOR ED N EC

- m data bits together with r error check bits form an n = (m + r) bit codeword

- The number of bits two codewords differ in is called the hamming distance between the two codewords

- Significance : If two codewords are at HD d then it requires d single bit errors to convert one into the other

HD OF A CODING SCHEME

- For m bit data ..All the 2^m possible combinations are legal
- But not all the 2^n codewords are used

-- in a coding scheme (algorithm to compute the check bits) some of these codewords are legal and others are illegal

For eq .. Consider parity : 1(r = 1) parity bit is appended with value so that the total number of 1's in the codeword is even ..

Then 11011 is a legal codeword in this scheme but 11010 is not

HD OF A LIST OF LEGAL CODEWORDS

- Minimum HD between any pair of legal codewords in the list

Remember : Each algorithm to compute the check bits create a different list of legal codewords

USE OF HD FOR ERROR DETECTION

- To detect d single bit errors , we need (an algorithm that creates) a code list with HD at least d + 1

For eg . For the parity scheme .. HD is 2 ..hence it can be used to detect single bit errors (d=1)

CONTINUED…

- If the recvd codeword is legal .. We accept it ,
- And if it is illegal we report (detect) an error
- Q1 : Can it happen that we recv a legal codeword when d single bit errors have ocurred …this is eqwt to saying can we get a legal code from another legal code by d single bit errors?
- A1 : No, since the HD of the code is at least d + 1. So a legal CW can be genearted from another LCW by inerting at least d + 1 bits and not by inverting d or less bits.

CONTINUED…

- Q2 : Can we get an illegal CW when no error has occurred ?
- A2 : Obviously not ..since the legal CW was sent by the sender and if no error has occurred then the recver must recv a legal CW

USE OF HD FOR ERROR CORRECTION

- To correct d single bit errors , we need (an algorithm that creates) a code list with HD at least 2d + 1.

- For eg. Consider the following legal CWs:

  0000000000, 0000011111, 1111100000, 1111111111

HD is 5 .. It can be used to correct 2 single bit errors

CONTINUED..

Claim : Suppose we recv an illegal code C .. Then there is a unique legal code which is at a distance d or less from C

Proof : Suppose there are 2 codes C1 and C2 at distance d (or less) from C .. Then C1 can be obtained from C2 by 2d (or less) inversions .. A contradiction to (code has HD at least 2d + 1)

CONTINUE…

- Obtain C1 from C2

Lets rearrange the bits of C so that all the bits(B1) that are inverted to obtain C1 are in the beginning followed by bits(B2) inverted to obtain C2 ..followed by the remaining bits (B3) ..In the worst case there is no overlapping between B1 and B2 .. In that case C1 is obtained from C2 by inverting exactly these B1 and B2 bits which together are no more than 2d .. (if there is some overlapping then those bits are not inverted, hence < 2d)

HAMMING CODE TO CORRECT ONE BIT ERRORS

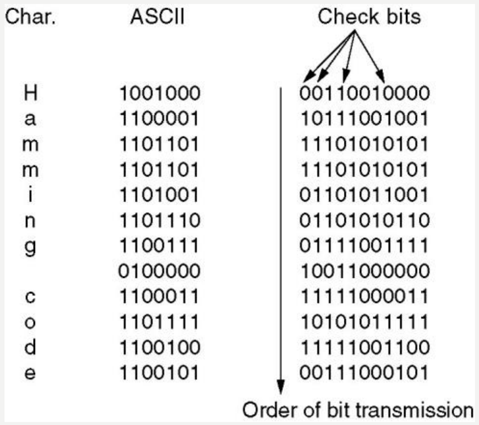
- The bits of the CW are numbered left to right , starting from 1 … the bits that are powers of 2 are check bits (1,2,4,8 …) and the remaining are data bits.
- Expand the position of each data bit in powers of 2 ..for eg. 11 = 1 + 2 + 8 .. So 11th bit contributes to the computation of value of these check bits I.e. 1,2, 8

CONTINUED…

- We do this for each data bit ..
- The value of a check bit is computed so that the parity of the all the data bits that contribute to it together with the check bit itself is even.
- For eg .

data bits 1001000 will be sent as the codeword 00110010000

HAMMING CODE TO CORRECT BURST ERRORS



Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

ERROR DETECTING CODE

- Polynomial code or CRC(Cyclic Redundancy Check)

CRC

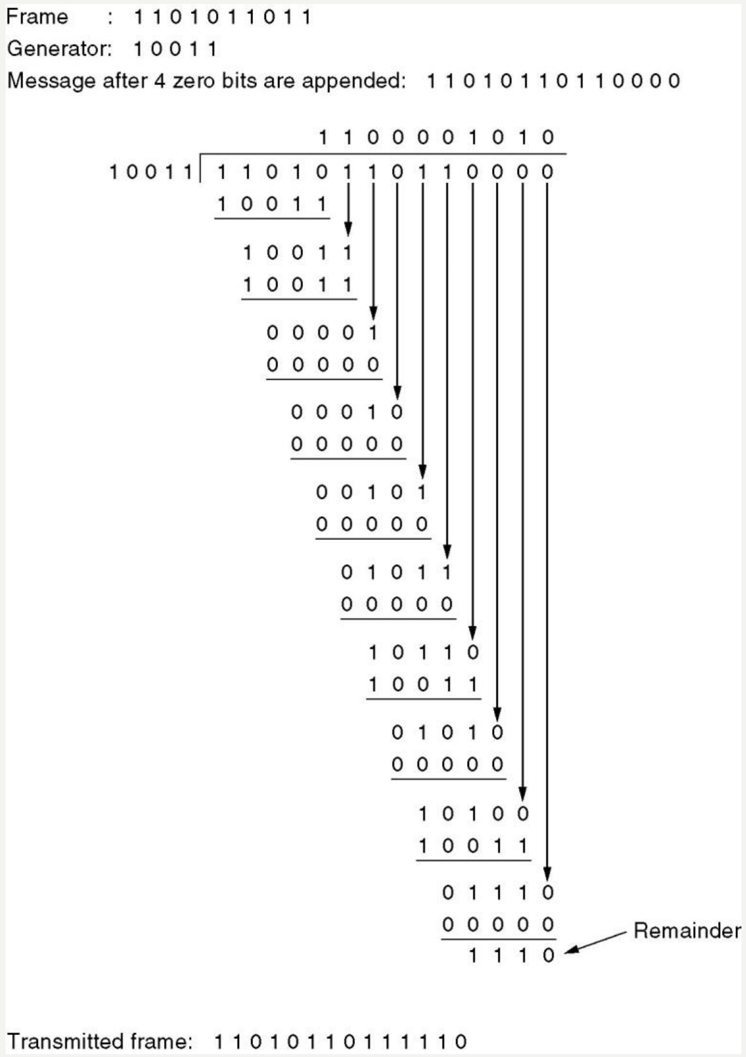
- A message m : a string of bits corresponds to a polynomial denote it by M(x).
- r check bits ….polynomial R(x).
- Transmitted bits m + r ….polynomial

T(x) = M(x) + R(x)

- Generator polynomial G(x)
- r checkbits are computed so that when G(x) divides T(x), the remainder is zero.

ERROR-DETECTING CODES

Calculation of the polynomial code checksum.



```
Frame     :  1 1 0 1 0 1 1 0 1 1
Generator:   1 0 0 1 1
Message after 4 zero bits are appended:  1 1 0 1 0 1 1 0 1 1 0 0 0 0

                      1 1 0 0 0 0 1 0 1 0
            1 0 0 1 1 ) 1 1 0 1 0 1 1 0 1 1 0 0 0 0
                        1 0 0 1 1
                        ---------
                          1 0 0 1 1
                          1 0 0 1 1
                          ---------
                            0 0 0 0 1
                            0 0 0 0 0
                            ---------
                              0 0 0 1 0
                              0 0 0 0 0
                              ---------
                                0 0 1 0 1
                                0 0 0 0 0
                                ---------
                                  0 1 0 1 1
                                  0 0 0 0 0
                                  ---------
                                    1 0 1 1 0
                                    1 0 0 1 1
                                    ---------
                                      0 1 0 1 0
                                      0 0 0 0 0
                                      ---------
                                        1 0 1 0 0
                                        1 0 0 1 1
                                        ---------
                                          0 1 1 1 0
                                          0 0 0 0 0
                                          ---------
                                            1 1 1 0   <- Remainder

Transmitted frame:  1 1 0 1 0 1 1 0 1 1 1 1 1 0
```

CRC CONTD..

- At the receiving end, receiver again divides the polynomial corresponding to the received bits by G(x) and accepts it iff the remainder is zero.
- Now let E(x) denote the polynomial corresponding to the errored bits. Then receiver receives

  T'(x) = T(x) + E(x)
- G(x) divides T'(x) iff it divides E(x)

CRC

- Detecting single bit errors

  $E(x) = x^i$

Choose $G(x)$ = any polynomial with at least two terms

- Detecting 2 single bit errors

  $E(x) = x^i + x^j = x^i (x^{(j-i)} + 1)$

Choose $G(x)$ s.t it neither divides x nor divides $x^k + 1$ for any $k \le$ frame length

- Detecting odd number of single bit errors

$E(x)$ can’t be of the form $(x + 1)\ Q(x)$

Choose $G(x)$ of the type $(x + 1)\ Q(x)$

G(X) = A GENERAL POLYN OF DEGREE R

- Will detect single burst of length <= r
- Will accept (without detecting) bursts of length r+1 with probably only ½^(r-1)

- Will accept longer bursts (without detecting) with probability only ½^r

Note : Certain Polynomials have become international standards

DETECTING SINGLE BURST OF LENGTH K <=R WITH A GEN POLYN OF DEGREE R

- $E(x) = x^i (x^{(k-1)} + \dots + 1)$

- Choose $G(x) = Q(x) + 1$

If $k-1 <$ degree of $G(x)$ then $G(x)$ can never divide $E(x)$ … I.e. if $k-1 < r$ ..I.e. if $k <= r$

IEEE 802 LANS USE

- For eg.

X^32 + x^26 + x^23 + x^22 …...x^2 + x + 1

Detects single burst of length <= 32

Note :A simple shift register circuit can be constructed to compute and verify the checksums in hardware.