gradeup

Function of OSI and TCP/IP Layers Part-2

Prep Smart. Stay Safe. go Gradeup

gradeup.co

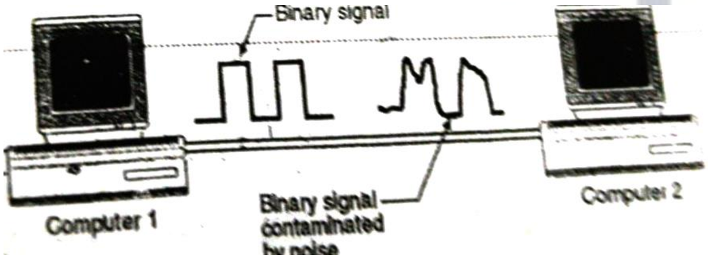
Function of OSI and TCP/IP Layer Part-2

Content :

1. **Types of error**
   a. **Single bit**
   b. **Burst Error**
2. **Definition of related to Codes**
3. **Meaning of error detection**
4. **Methods of Detection Method**

Introduction to Error Detection and Correction :

- When transmission of digital signals take place between two systems such as computer system shown below , the signal get contaminated due to the addition of "Noise" to it
- The noise can introduce an error in the binary bits travelling from one system to the other . That means a 0 may change to 1 or 1 may change to 0



- There error can become a serious threat to the accuracy of the digital system. Therefore it is necessary to detect and correct the errors

Types of Error :

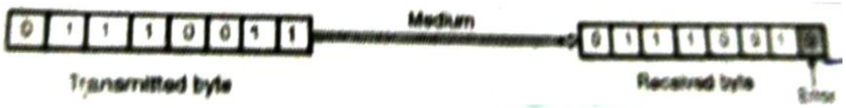
The error introduced in the data bits during their transmission can be categorized as :

- Content errors
- Flow integrity errors

- The content errors are nothing but errors in the contents of message e.g. a “0” may be received as “1” or “1” may be received as “0” . Such error are introduced due to noise added into the data signal during its transmission
- Flow integrity error means missing blocks of data . It is possible that a data block may be lost in the network as it has been delivered to the wrong destination
- Depending on the number of bits in error we can classify the errors into two types as
  1. Single bit error
  2. Burst errors

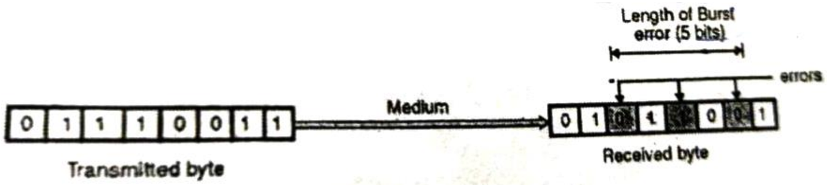
Single Bit Error :

- The term bit error suggests that only one bit in the given data unit such as byte is in error .
- That means only one bit will change from 0 to 1 or 1 to 0 as shown below :



Burst Error

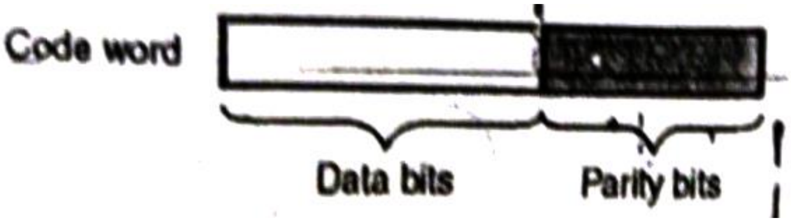
- If two or more bits from the data unit such as byte change from 0 to 1 or from 1 to 0 then burst errors are said to have occurred
- The length of burst is measured from the first corrupted bit to last corrupted bit . Some of the bit in between may not have been corrupted.
- Burst error illustrated below:



Disadvantage of coding : As increased transmission bandwidth is required in order to transmit the encoded signal This is due to the additional bits(redundancy) added by the encoder .

Important definition released to codes:

- Code word: It is n bit encoded block of bits . As already seen it contains message bits and parity or redundant bits , shown below :



Code word = data bits + parity bits

Code rate : It is define as the ratio of number of message bit(K) to the total number of bits (n) in code word.

Code rate(r) = k/n

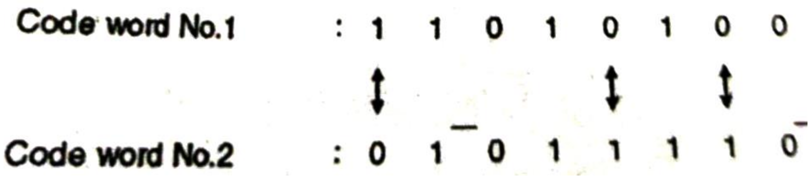
Hamming weight of a code word([w(x)] : The hamming weight of a code word x is defined as the number of zero elements in the code word . Hamming weight of a code vector (code word) is the distance between that code word and an all zero code vector.(a code having all elements equal zero).

Code efficiency : It is defined as the ratio of message bits to the number of transmission bits per block

Code efficiency = Code rate(r) = k/n

Hamming distance :

- Consider two code vectors(or code word) having the same number of element
- The "Hamming distance " or simply distance between the two code words is defined as the number of locations in which their respective element differ. Example , consider the two code words as shown below :



Code word No.1 : 1 1 0 1 0 1 0 0

Code word No.2 : 0 1 0 1 1 1 1 0

Note: The bit 2,4 and 8 are different from each other . Hence hamming distance is 3.

Minimum distance d_{min} :

- The minimum distance "d_{min} " of linear block code is defined as the smallest Hamming distance between any pair of code vector in the code
- Therefore the minimum distance is same as the smallest Hamming weight of distance between any pair of code vectors.
- It can be proved that the minimum distance of linear block code is the smallest hamming weight of the non-zero code vector in the code .

Role of the "d_{min} " in error detection and correction :

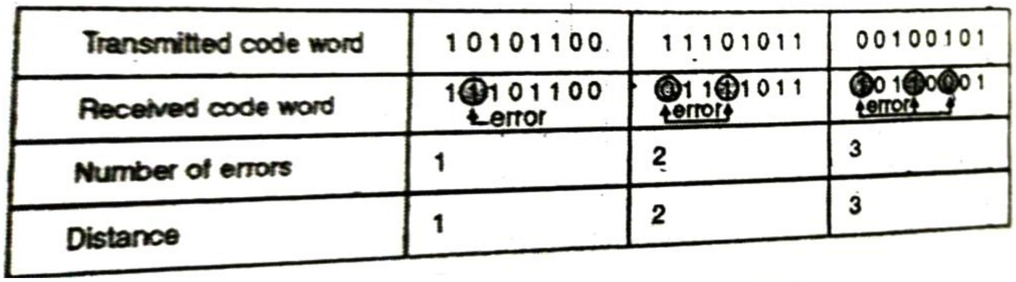
- The error detection is always possible when the number of transaction errors in a code word is less than the minimum distance "d_{min} " because then the erroneous code word many correspond to another valid code word and error cannot to detected .
- The error detection and correction capabilities of a coding technique depend on the minimum distance as show below :

Detect up to "s" errors per word	d_{min} >= (s+1)
Correct up to "t" error per word	d_{min} >=(2t+1)
Correct up to "t" errors and detect s>t errors per word	d_{min} >=(t+s+1)

Error Detection :

- When a code word is transmitted one or more number of transmitted bits will be reserved (0 to 1 or vice-versa) due to transmission impairments
- Thus error will be introduced

- It is possible for the receiver to detect these errors if the received code word (corrupted) is not one of the valid code word.
- When the error is introduced , the distance between the transmitted and received code words will be equal to the number of errors as illustrated below

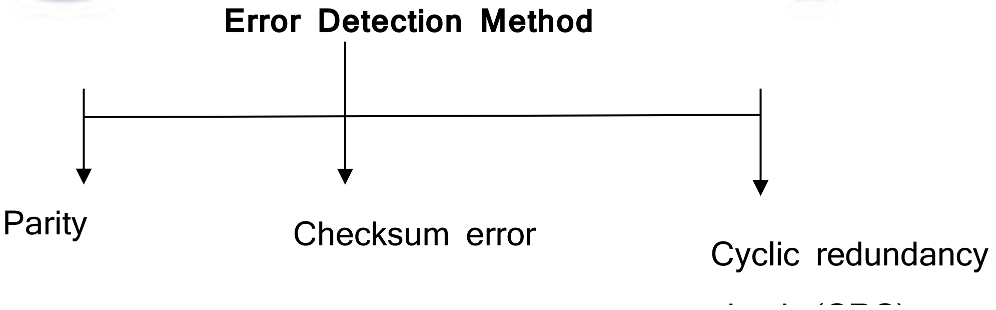


Transmitted code word	10101100	11101011	00100101
Received code word	11101100 error	01111011 error error	10110001 error
Number of errors	1	2	3
Distance	1	2	3

- Hence to detect the errors at the receiver , the valid code word should separated by a distance of more than 1
- Otherwise the incorrect received code word will also become some other valid code word and the error detection will be impossible
- The number of errors that can be detected depends on the distance between any two valid code words

Error detection methods :

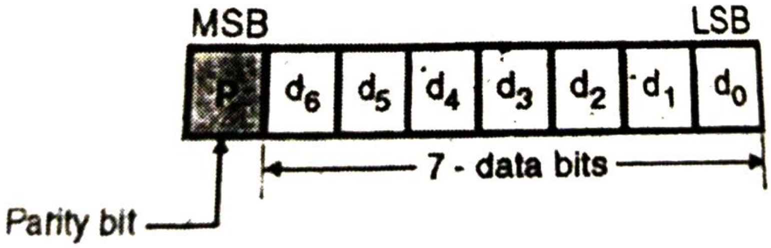
- Some of the most important error detection methods as follows



Parity :

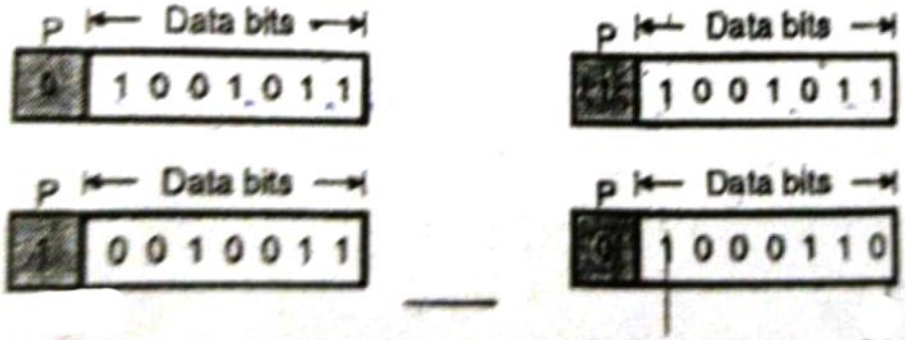
- The simplest technique for detecting the errors is to add an extra bit known as parity bit to each word being transmitted
- As shown below , generally the MSB of 8- bit word is used as the parity bit and the remaining 7 bits are used as data or message bits

- The parity of the 8 bit(byte) transmitted word can be either even parity or odd parity
- EVEN parity means that the number of 1’s in the given word including the parity bit should be even (2,4,6,8,10…)
- ODD parity means that the number of 1’s in the given word including the parity bit should be odd (1,3,5,7,9….)



Use of parity bit to decide parity :

- The parity bit can be set to 0 or 1 depending on the type of parity required
- For odd parity this bit is set to 1 or 0 at the transmitter such that the number of “1 bits” in the entire word is odd
- For even parity this bit is set to 1 or 0 such that the number of “1 bits” in the entire word is even . This is illustrated below



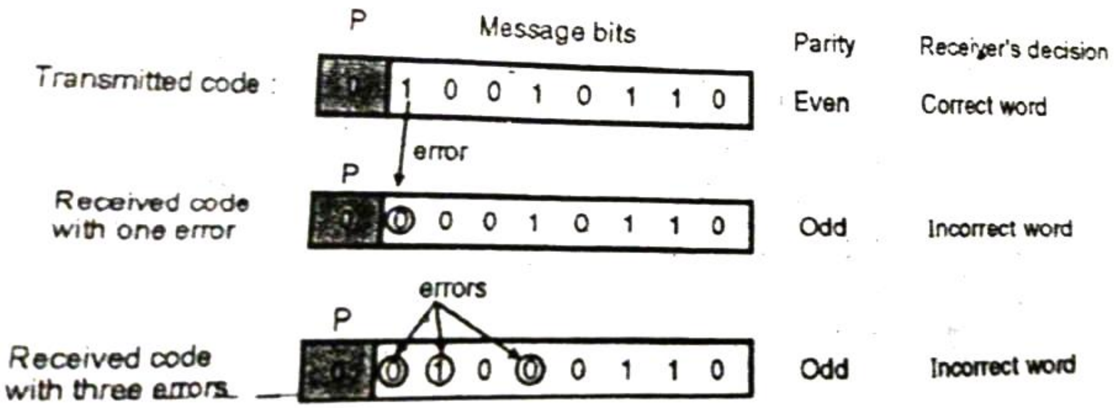
(a) Inclusion of a parity bit to obtain an even parity

(b) Inclusion of a parity bit to parity obtain the odd parity

How does error detection take place :

- The parity checking at the receiver can detect the presence of an error it the parity of the received signal is different from the expected parity

- That means if it is known that the parity of the transmitted signal is always going to be “even” and if the received has n odd parity then the receiver can conclude that the received signal is not correct as shown below
- When a single error or an odd number of errors occur during transmission the parity of the code word changes
- Parity of the received code word is checked at the receiver and change in parity indicate that error is present in the received word
- If presence of error is detected then the receiver will ignore the received byte and request for the retransmission of the same byte to the transmitter



Conclusion:

1. Double or any even number of error in the received word will not change the parity . Therefore even number of error will be unnoticed
2. If one or odd number of errors occur then the parity of the received word will be different from the parity of transmitted signal. Thus error is noticed . however this error cannot be corrected.

Limitation of parity checking :

1. Thus the simple parity checking method has its limitations . It is not suitable for detection of multiple errors(two, four, six etc)
2. The other limitation of parity checking method is that it cannot reveal the location of erroneous bit . It cannot correct the error either

Cycle Redundancy Check(CRC):

- This is a type of polynomial code in which a bit string is represented in the form of polynomials with coefficient of o and 1 only
- Polynomial arithmetic uses a mudlo-2 arithmetic i.e. addition and subtraction are identical to EXOR
- For CRC code the sender and the receiver should agree upon a generator polynomial G(x) . A code word can be generated for a given data word (message) polynomial M(x) with the help of long division
- This technique is more powerful than other two
- CRC is based on binary division . A sequence of the redundant bits called CRC or CRC remainder is appended at the end of a data unit such as byte.
- The resulting data unit after adding the CRC remainder becomes exactly divisible by another predetermined binary number.
- At receiving end , this data unit is divided by the same binary number
- There is no error if this division does not yield any remainder . But a non-zero remainder indicates the presence of errors in the received data unit
- Such the erroneous data unit is then rejected

Procedure to obtain CRC:

The redundancy bits used by CRC are derived by following procedure:

1. Divide data unit by a predetermined divisor
2. Obtain the remainder . It is CRC

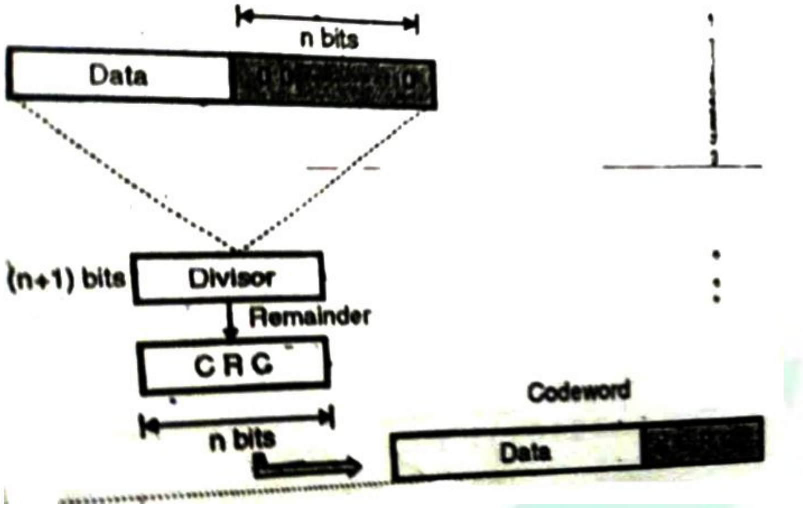
Requirements of CRC

A CRC will be valid if and only if it satisfies the following requirement :

- It should have exactly one less (n-1) bit than divisor
- Appending CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor

CRC generator :

The CRC generator is shown below :



The stepwise procedure in CRC generator is as follows :

Step1: Append a string on n 0s to the data unit where n is 1 less than the number of bits in the predecided divisor (n+1 bit long0

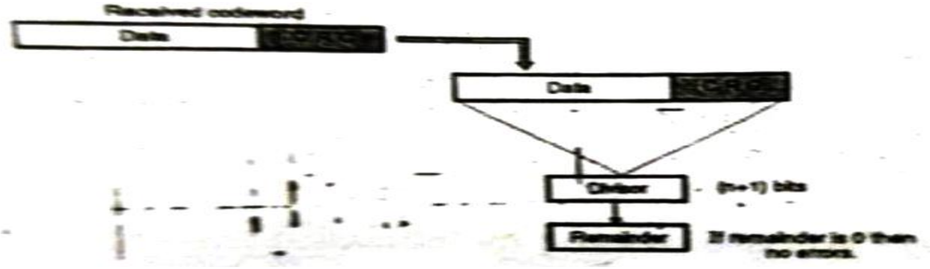
Step2: divide the newly generated data unit step 1 by the divisor . This is a binary division

Step 3: the remainder obtained after the division in step 2 is the n bit CRC

Step4: This CRC will replace the n 0s appended to the data unit in step 1 , to get the code word to be transmitted as shown below :

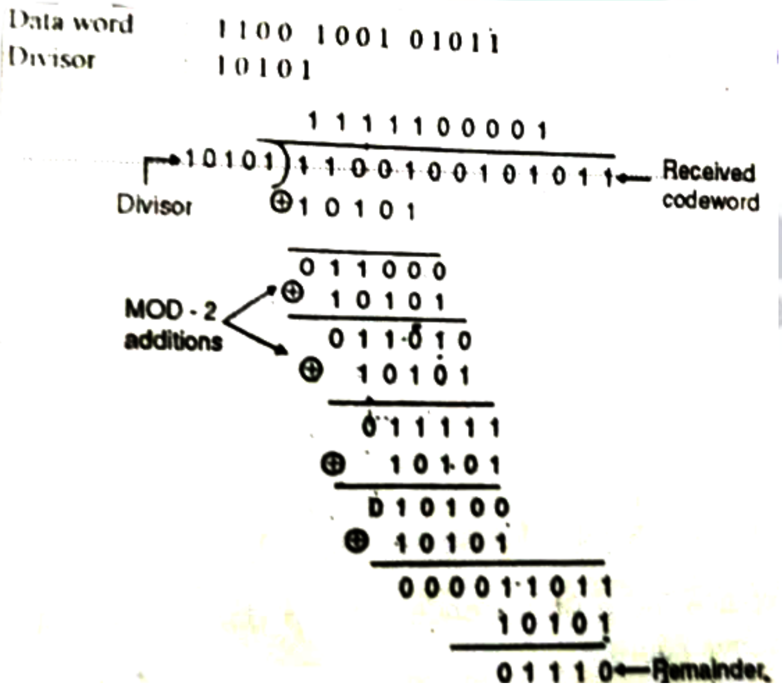
CRC checker

- Below figure shows the CRC checker



- He code word received at the receiver consists of data and CRC
- The receiver treats it as one unit and divides it by the same (n+1) bit divisor which was used at the transmitted
- The remainder of this divisor is then checked
- If the remainder of this divisor is then checked
- But a non-zero remainder indicates presence of errors of errors hence the corresponding code word should be rejected

Example: The code word is received as 1100100101011 . Check whether there are errors in the received code word , if the divisor is 10101 (The divisor corresponds to the generator polynomial)


Data word 1100 1001 01011
Divisor 10101
1 1 1 1 1 0 0 0 0 1
10101
1 1 0 0 1 0 0 1 0 1 0 1 1
Received codeword
Divisor
⊕ 1 0 1 0 1
0 1 1 0 0 0
⊕ 1 0 1 0 1
MOD - 2 additions
0 1 1 0 1 0
⊕ 1 0 1 0 1
0 1 1 1 1 1
⊕ 1 0 1 0 1
0 1 0 1 0 0
⊕ 1 0 1 0 1
0 0 0 0 1 1 0 1 1
1 0 1 0 1
0 1 1 1 0
Remainder