International Journal of Innovative Research in Engineering & Management (IJIREM)
ISSN: 2350-0557, Volume-8, Issue-3, May 2021
https://doi.org/10.21276/ijirem.2021.8.3.6
www.ijirem.org

Analysis of Error Detection and Correction in Data Link Layer

R. Raja Rajeswari
Department of Computer Science,
Bon Secours Arts & Science College for Women,
Mannargudi-614001, Thiruvarur, Tamil Nadu, India
rajpreethika81@gmail.com

ABSTRACT

In digital systems, the analog signals will change into a digital sequence. This sequence of bits is called Data stream. The objective of this paper represents that the change in position of a single bit also leads to a major error in the data output. In this paper, we present an overview of error control regarding error detection and error correction. It reveals that finds errors and uses error detection and correction techniques to get the exact or approximate output. Error control describes how the network handles and detects errors, especially in the data link layer. This article mainly discusses the type of error detection mechanisms that are used to detect the errors and how the errors can be corrected so the receiver can extract the real data.

Keywords

Types of Error, Error Detection, Error Correction Techniques

1. INTRODUCTION

The data link layer, or layer 2, is the second layer of the seven-layer OSI model of computer networking. Data link layer is accountable for converting data stream to signals bit by bit and to send that over the underlying hardware. At the receiving end, Data link layer picks up data from hardware which are in the form of electrical signals assembles them in a noticeable frame format, hands over to upper layer. The data link layer affords the purposeful and procedural means to transfer data between network entities and might provide the means to detect and probably correct errors that may occur in the physical layer. Data-link layer uses error control techniques to ensure that accuracy of frames. These errors can be single bit or multiple bits, which make the information unreadable at the receiver's end. These types of error are also known as single bit or multiple bits burst errors. To overcome these types of errors, two phases of error checking are involved, i.e. error detection and correction. Generally the data link layer offers some functions to detect and correct such errors. Different error detection such as Parity bit checking, Checksum method, error detection based on Hamming Distance, cyclic redundancy checking, etc., and correction such as Automatic repeat Request, Forward error correction code, Low Density Parity Checking, etc. [1] are also used in the data link layer to solve these types of problems [2].

2. METHODS

2.1 Error

The data can be corrupted during transmission from source to receiver. It may be affected by external noise or some other physical imperfections. In this case, the input data is not same as the received output data. This mismatched data is called "Error" in figure 1. The data errors cause loss of data. Even one bit is modified in data may affect the entire system's response. In this case, the data error is likely to be changed in positions of 0 and 1

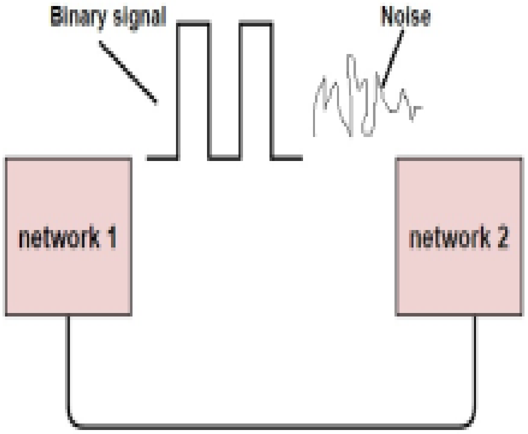


Figure 1: Error

2.2 Types of Errors

In a data sequence, if 1 is changed to zero or 0 is changed to 1, it is called "Bit error". There are generally 3 types of errors occur in data transmission from transmitter to receiver. They are Single Bit Data Errors, Multiple Bit Data Errors, and Burst Errors.

2.2.1 Single Bit Data Errors

The change in one bit in the whole data sequence is called "Single bit error" shown in figure 2. The Existence of single bit error is uncommon in serial communication method. This error type happens only in parallel communication method, as data is transmitted bit wise in a single line, there may be noise in the single line.

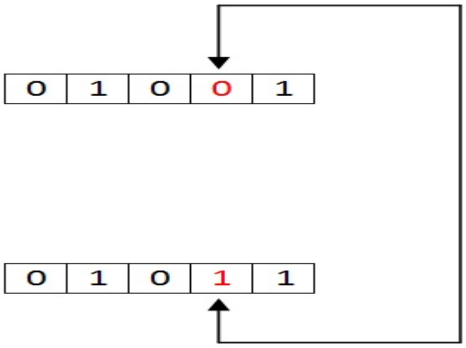


Figure 2: Single bit error

2.2.2 Multiple Bit Data Errors

If there is change in two or more bits of data sequence of transmitter to receiver, it is called “Multiple bit error” shown in figure 3. This type of error occurs in both serial type and parallel type data communication networks.

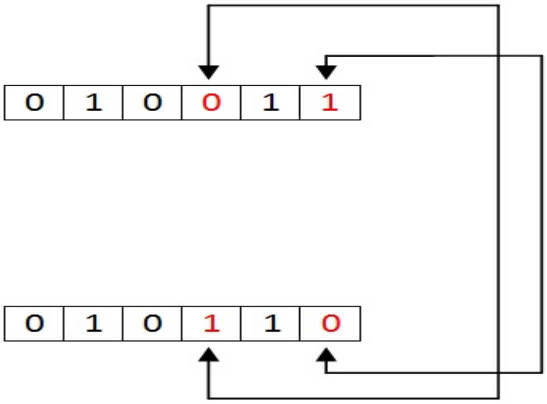


Figure 3: Multiple bit error

2.2.3 Burst Errors

The change of set of bits in data sequence is called “Burst error” as shown in figure 4. It is computed from the first-bit change to the last bit change.

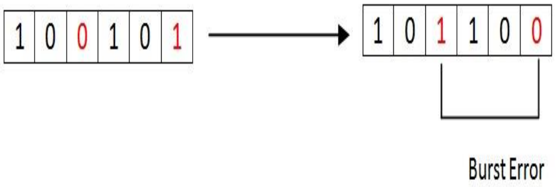


Figure 4: Burst Errors

Here identify the error from 3rd to 6th bit. The numbers between 3rd and 6th bits are also considered as error. These set of bits are called “Burst error”. These burst bits changes from transmitter to receiver [10], which may cause a major error in data sequence. This type of errors occurs in serial communication and they are difficult to solve.

3. ERROR DETECTION CODES

The transmission control circuit enables the existence of bits or transmission error in the receiving group to be detected when moving through a bit stream over a transmission line or channel a scheme incorporated. Generally, this is completed by the transmitter which processes a set of extra bits based on the contents of groups of bits to be transmitted. It is known as error detection which is constructed on a group of extra bits which is moved with the real bits in the block. The receiver uses the complete sets of received bits to determine whether the block contains any error to the high probability [3].The two factors that determine the type of error detection scheme used are the bit error rate (BER) probability of the line and the type of error, that whether the errors occur as random single-bit errors or as burst error. The three most widely used schemes are parity, cyclic redundancy checks (CRC) and checksum.

3.1 Parity Checking

The parity-check is the simplest error detecting method. In this, an extra parity is appended to the data bits to create an even or odd bit. There are two types of parity bits in error detection; they are even parity, odd parity shown in table 1.

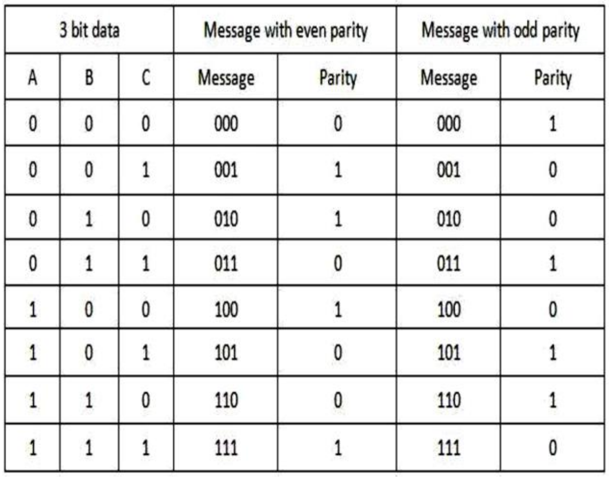
3.1.3 Even Parity

If the data has even number of 1’s, the parity bit is 0. Ex: Number is 10000010 the result of parity bit is 0, Odd number of 1’s, the parity bit is 1. Ex: Number is 11000001 the result of parity bit is 1

3.1.2 Odd Parity

If the data has odd number of 1’s, the parity bit is 0. Ex: Number is 10001100 the result of parity bit is 0, Even number of 1’s, the parity bit is 1. Ex: Number is 11010001 the result of parity bit 1.

Table 1: Message with Even and Odd Parity



3 bit data			Message with even parity		Message with odd parity	
A	B	C	Message	Parity	Message	Parity
0	0	0	000	0	000	1
0	0	1	001	1	001	0
0	1	0	010	1	010	0
0	1	1	011	0	011	1
1	0	0	100	1	100	0
1	0	1	101	0	101	1
1	1	0	110	0	110	1
1	1	1	111	1	111	0

4. ERROR DETECTING TECHNIQUES

This approach implemented either at Data link layer or Transport Layer of OSI Model. The sender and receiver, either both or any, must ascertain that there is some error transit. It can be used error-detecting codes which are extra data added to a given digital message to aid to identify if any error has ensued during communication of the message. A basic approach that can be used for error detection is redundancy bits, where extra bits are added to aid the detection of errors. Some popular techniques for error detection are: - Simple Parity check, Two-dimensional Parity check, Checksum, Cyclic redundancy check.

4.1 Simple Parity check

Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of 1 is added to the block if it contains odd number of 1's, and 0 is added if it contains even number of 1's .This scheme makes the total number of 1's even, that is why it is called even parity checking shown in figure 6.

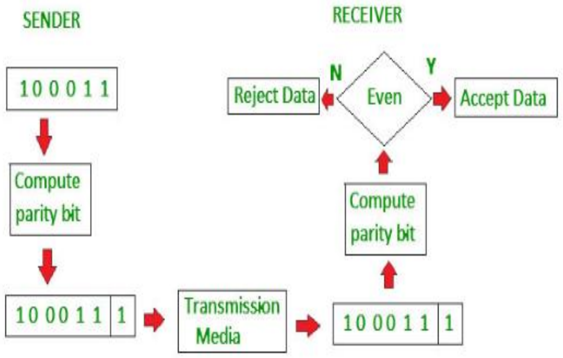
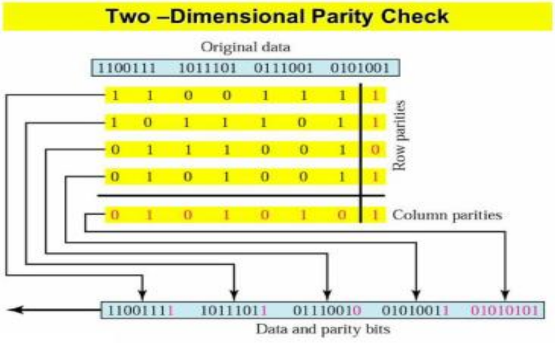


Figure 6: Simple Parity Check

4.2. Two-dimensional Parity check

Parity check bits are calculated for each row, which is equivalent to a simple parity check bit.



1100111 1011101 0111001 0101001

1	1	0	0	1	1	1	1
1	0	1	1	1	0	1	1
0	1	1	1	0	0	1	0
0	1	0	1	0	0	1	1
0	1	0	1	0	1	0	1

11001111 10111011 01110010 01010011 01010101

Figure 7: Two-dimensional parity check

These are also considered for each column, and then both are directed in addition to the data. At the receiving end these are related with the parity bits computed from the received data. Original data is shown in Figure 7.

4.3 Cyclic redundancy check (CRC)

Unlike checksum scheme, which is based on addition, CRC is based on binary division shown in figure 8. CRC is an order of redundant bits, called cyclic redundancy check bits, are added to the end of data unit thereby the resultant data unit becomes accurately divisible by a second, predetermined binary number.

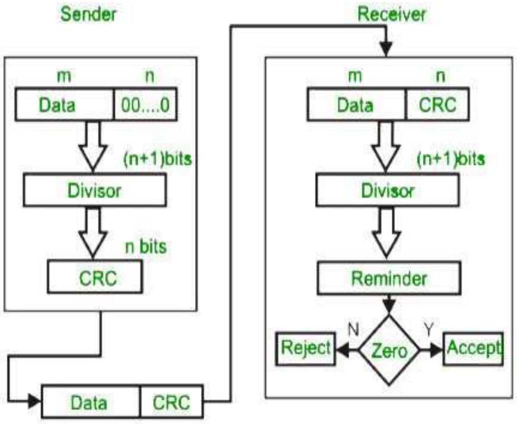


Figure 8: Cyclic redundancy check (CRC)

At the destination, the received data unit is separated by the similar number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.

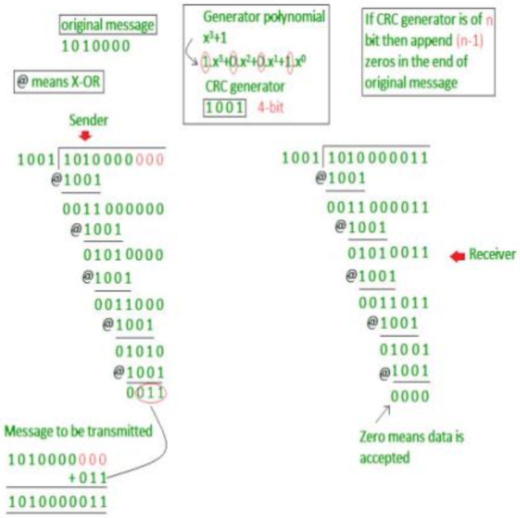


Figure 9: Polynomial Division

A remainder indicates that the data unit has been damaged in transit and therefore must be rejected shown in figure 9.

4.4 Checksum

In checksum error detection scheme, the data is divided into k segments each of m bits. The sections are added together using 1's complement to get the sum for the senders.

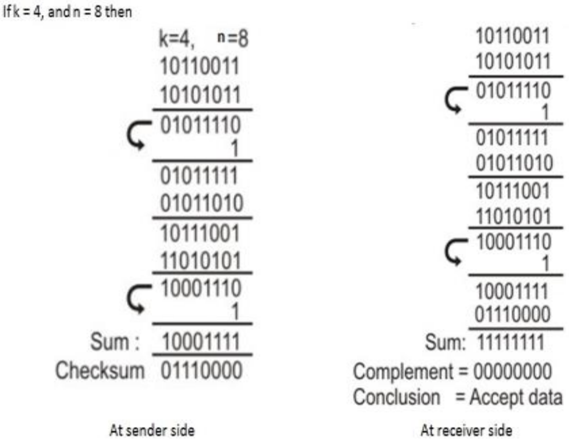


Figure 10: Check Sum

In the sum is complemented to get the checksum. The checksum segment is sent along with the data segments. All received sections are added using 1's complement to get the sum for the receiver's end. The sum is complemented. If the result is zero, the received data is accepted; otherwise discarded shown in figure 10.

5. ERROR CORRECTION TECHNIQUES

Error Detection and Correction types are shown in figure 11.

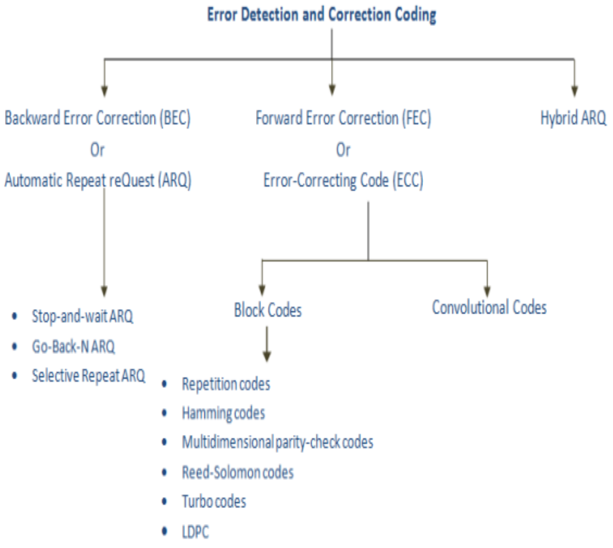

Error Detection and Correction Coding
Backward Error Correction (BEC)
Or
Automatic Repeat reQuest (ARQ)
Forward Error Correction (FEC)
Or
Error-Correcting Code (ECC)
Hybrid ARQ
Stop-and-wait ARQ
Go-Back-N ARQ
Selective Repeat ARQ
Block Codes
Convolutional Codes
Repetition codes
Hamming codes
Multidimensional parity-check codes
Reed-Solomon codes
Turbo codes
LDPC

Figure 11: Types of Error Detection and Correction

In error correction two basic approaches are used the first approach that is based on backward error correction. The backward error correction can be corrected, when the error is identified in the frame. Then it will send information to the sender and ask the sender to retransmit the frame once again. This method is known as ARQ and this system is known as backward error correction, for going back to communication once again and this certain system. Error Detection and Correction When the receiver receives a correct frame, it sends acknowledge to the sender is called positive ACK. When the receiver receives a damaged frame or error frame, it sends a NACK back to the sender and the sender again retransmit the correct frame is called Negative ACK. The sender sends the data and maintains a clock limit and sets a fix timeout period. If an acknowledgement of a data-frame previously transmitted does not arrive before the timeout at the transmitter time period. The sender understand data was loosed and not reach to transmitter, it again sends data frame and wait for acknowledgement is called Retransmission. There are three types of techniques available for control the errors by Automatic Repeat Requests (ARQ) such as Stop-and-wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ.

5.1 Forward error correction

In forward error correction, send a frame specially encoded such that in some cases, the receiver can correct the errors but, in some cases, they couldn't correct errors. There were certain cases, where it's not perfect. So, we still need some way to ask the sender to retransmit to send again for example, send a frame and because of errors it doesn't get to the destination, it says that's a lost frame. On this method, Sender sends a frame to the destination and the destination sends back an ACK that frame is a positive acknowledgment. If received the frames with errors on this approach receiver give a negative acknowledgement and send a frame to the destination. The destination sends back a message NAK that frame sent me has errors, that's an acknowledgment that something's gone wrong and if detected something's gone wrong [2].

5.2 Automatic Repeat Request or ARQ

When it combines that with a positive acknowledgement, it will need a timeout mechanism the approach. The sender sends a frame to the destination, and waits for a positive acknowledgment, if the destination receives that frame it sends back a positive acknowledgement. If sending the frame and there's an error and the destination, does not receive the frame and waiting for a positive acknowledgement. So here implements a timeout mechanism, after sending frame wait for some time, if it does not receive the ACK within that time, then it will assume, it's lost and resend or retransmit it. This process is called automatic repeat request or ARQ [9].

5.3 Stop-and-wait ARQ

Stop-and-wait ARQ protocol is based on the flow control, the origin sends a frame reminisce stop-and-wait flow control send a frame wait for the data ACK, direct the next frame waits for the ACK, and so on. Here the source sends a single frame, but we now introduce a timer. Send frame and then start the timer and start counting time and waiting for an ACK to come back. If an

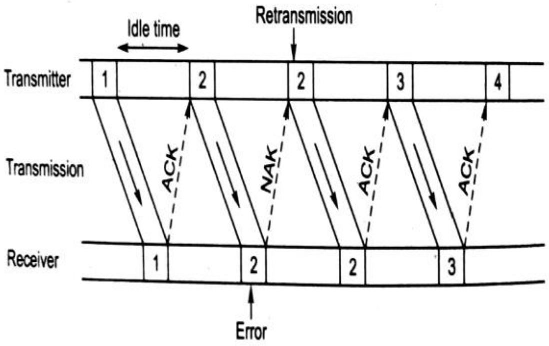


Figure 12: Stop and wait ARQ

When will the timer stop that time automatically an ACK return to the sender, and transmit on the next frame it's no errors, if the errors and frame damaged, then it eliminated that frame. Stop and wait ARQ data transmission system, the transmitter sends a code vector to the receiver and wait for an acknowledgement from the receiver as shown in figure 12. When positive acknowledgment received from receiver that means original code vector received, further the transmitter sends the next code vector. A negative acknowledgement (NAK) from the receiver indicates that the receiver vector has been detected in error; then transmitter recent the code vector. Retransmission continues until and ACK is received by the transmitter [4]. However, the stop and wait ARQ scheme is inherently inefficient because of the ideal time spend waiting for an acknowledgement is transmitted code vector. Unless the code vector length n is extremely long the fraction of ideal time can be large [5]

5.4 GO-Back-N ARQ

At the receiver, the N-1 received vectors. Therefore, the receiver needs to store only one receive vector at a time. Because of the continuous transmission and we transmission support vector, then go back N ARQ scheme is more efficient than the stop and wait ARQ and it can be implemented at moderate cost communication protocols [5].

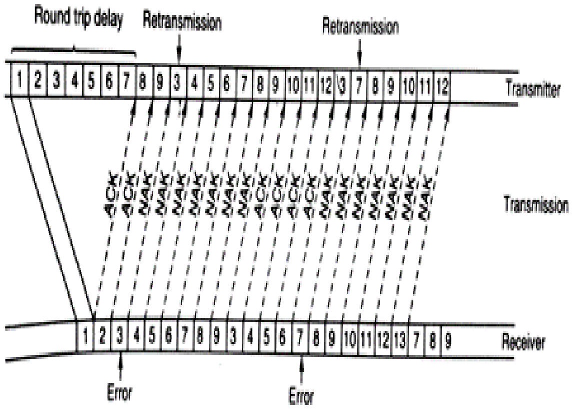


Figure 13: Go- back- N ARQ

Then go back N ARQ system becomes not efficient when the round-trip delay is huge and the rate of data transmission is high. As shown in figure 13. The transmitter transmits code vector 1 and does not wait for acknowledgement and its continuous transmit the next code vector 2 3 4 5 6 7 8 9 and receiver send the ACK and NAK. There for 3 respectively, then transmitter stops transmission of the code vector after 9 [6] [7]. And firstly, retransmit the all data from 3 to 9 which was in not acknowledge from the receiver after that transmit next code vector.

5.5 Selective repeat ARQ

In the selective repeat ARQ scheme, code better or also transmitted continuously. However, the transmitter only recent those code vectors that are negatively acknowledged as shown in figure 14. Since ordinary code vectors must be delivered to the user in the correct order a buffer must be provided. At the receiver to store the error free received vectors, following receipt vector detected in error. When the first negatively acknowledgement code vector is successfully received, the receiver then release the error-free received vectors in consecutive order until the next error received vector is encountered sufficient receiver buffer must be provided otherwise buffer overflow make your and data may be lost.

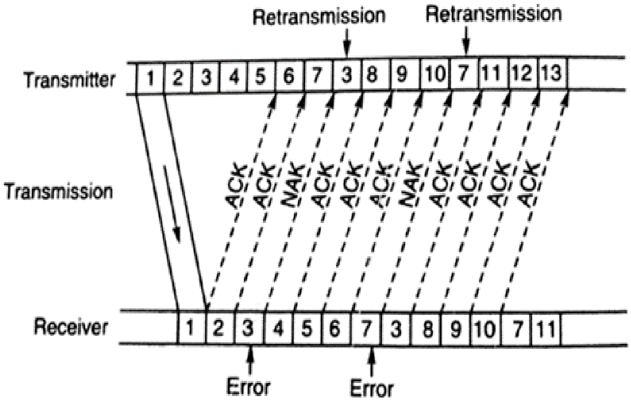


Figure 14: Selective repeat ARQ

The selective repeat ARQ is most efficient as per compared to other ARQ schemes [8]. The transmitter continuously transmits the code vector as in order. The receiver sends the acknowledgement to transmitter for all the code vectors and NAK, code vector re-transmits again, which was NAK through the receiver end after that continue to transmit until unless not acknowledge receiving.

6. CONCLUSIONS

The paper overviews that errors occur along with error-detecting code, it can also pass some data to figure out the original message from the corrupt message when it received. This type of code is called an error-correcting code. Error-correcting codes also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit. In error-correcting codes, parity check has a simple way to detect errors along with a suitable mechanism to refine the corrupt bit location. Once the corrupt bit is located, its value is reverted likely from 0 to 1 or 1 to 0 to get the original message.

REFERENCES

[1] Afiqah Azahari, Raed Alsaqour, Mohammed Al-Hubaishi and Mueen Uddin, Review of Error Detection of Data Link Layer in Computer Network, Middle-East Journal of Scientific Research 18 (8): 1105-1110, 2013.

[2] Shiladitya Bhattacharjee, Lukman Bin Ab. Rahim, Izzatdin B A Aziz, A Multibit Burst Error Detection and Correction Mechanism for Application Layer, 2014 International Conference on Computer and Information Sciences(ICCOINS),DOI.org/10.1109/ICCOINS.2014.6868380)

[3] Halsall, F., 2006. Computer Networking and the Internet,5/e: Pearson Education India.

[4] S B Wicker, (1994), Error Control Systems for Digital Communication and Storage, Prentice Hall Publishers.

[5] George Clark and J Cain," Error correcting code for digital communications" John Willey Publishers.

[6] Behrouz A. Forouzan, "Data Communication and Networking", 5 th Edition, Tata McGraw Hill, 2013.

[7] Tanenbaum, A. S,: Computer Networks, 4th edn.Prentice-Hall PTR,Englewood Cliffs(2003)

[8] William Stallings, "Data and Computer Communications", 8th Edition, Pearson Education, 2007.

[9] Leon-Garcia, Indra Widjaja, "Communication Networks", 2nd Edition, Tata McGraw-Hill, 2004.

[10] J.Fletcher, "An arithmetic checksum for serial transmissions, Communication". IEEE Transactions on 30:247-252,1982.