Data Link Layer

Data Link Layer is second layer of OSI Layered Model. This layer is one of the most complicated layers and has complex functionalities and liabilities. Data link layer hides the details of underlying hardware and represents itself to upper layer as the medium to communicate.

Data link layer works between two hosts which are directly connected in some sense. This direct connection could be point to point or broadcast. Systems on broadcast network are said to be on same link. The work of data link layer tends to get more complex when it is dealing with multiple hosts on single collision domain.

Data link layer is responsible 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 recognizable frame format, and hands over to upper layer.

Data link layer has two sub-layers:

- **Logical Link Control:** It deals with protocols, flow-control, and error control
- **Media Access Control:** It deals with actual control of media

Functionality of Data-link Layer

Data link layer does many tasks on behalf of upper layer. These are:

- **Framing**
  Data-link layer takes packets from Network Layer and encapsulates them into Frames. Then, it sends each frame bit-by-bit on the hardware. At receiver' end, data link layer picks up signals from hardware and assembles them into frames.
- **Addressing**
  Data-link layer provides layer-2 hardware addressing mechanism. Hardware address is assumed to be unique on the link. It is encoded into hardware at the time of manufacturing.
- **Synchronization**
  When data frames are sent on the link, both machines must be synchronized in order to transfer to take place.
- **Error Control**
  Sometimes signals may have encountered problem in transition and the bits are flipped. These errors are detected and attempted to recover actual data bits. It also provides error reporting mechanism to the sender.
- **Flow Control**
  Stations on same link may have different speed or capacity. Data-link layer ensures flow control that enables both machine to exchange data on same speed.
- **Multi-Access**

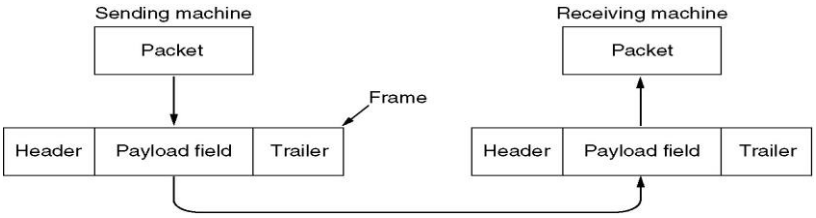
When host on the shared link tries to transfer the data, it has a high probability of collision. Data-link layer provides mechanism such as CSMA/CD to equip capability of accessing a shared media among multiple Systems.

Data link layer design issues:

Data link layer specifies number of specific functions which should be considered during its designing.

1) Providing a well-defined service interface to the network layer
2) Framing
3) Error detection
4) Flow control in the network

To accomplish all these goals the data link layer takes the packets and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet and a frame trailer



1) Service provided to network layer:

The function of data link layer is to provide service to the network layer. The principle service is transferring data from the network layer of the source to the network layer of the destination machine. The data link layer can be designed to offer various services and the services offered can vary from system to system. The three commonly used services are:

- **Unacknowledged connectionless service**

  Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. No logical connection is established before transmission. No attempt is made to detect the lost frame.

- **Acknowledged connectionless service**

  When this service is offered, there are still no logical connections used, each frame is sent is individually acknowledged. Here the sender knows whether a frame has arrived correctly. If it has not arrived within the specified time interval, it can be sent again.

- **Acknowledged connection oriented service**

  With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, the data link layer guarantees that each frame sent is indeed received exactly once in a right order.

2) Framing:

The usual approach is for the data link layer to break the bit stream into discrete frames and compute checksum for each frame. When the frames arrives at the destination, the checksum is re-computed. Breaking the bit stream into frames is more difficult and different techniques are implemented to achieve framing. Some of the methods are:

- Character count : specify the number of bytes in the frame
- Flag bytes with byte stuffing : Flag byte is used as both the starting and ending delimiter
- Starting and ending with the flag, with bit stuffing :Each frame begins and ends with a special bit pattern (ex:01111110)
- Physical layer coding violations :use some reserved signals to indicate the start and end.

3) Error detection or error control:

The data link layer ensures error free link for data transmission. The issues it caters to with respect to error control are −

- Dealing with transmission errors
- Sending acknowledgement frames in reliable connections
- Retransmitting lost frames
- Identifying duplicate frames and deleting them
- Controlling access to shared channels in case of broadcasting

4) Flow control:

Another important design issue that occurs in data link layer is, what should be done with a sender that systematically wants to transfer frames faster than the receiver can accept them. i.e the sender is very fast in transmission and the receiver receives the packets or frames very slowly. In such situations there are many possibilities of errors and loss of frames. Two approaches commonly used for flow control are:

- **Feedback based flow control:** Here the receiver sends back the information to the sender giving it permission to send more data or telling the sender how the receiver is doing its work.
- **Rate based flow control:** Here the protocol has a built in mechanism that limits the rate at which the sender may send the data without receiving feedback from receiver.

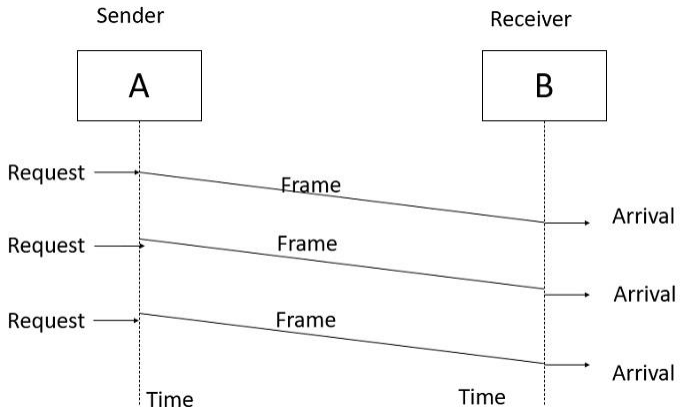
Elementary data link layer protocols

Elementary Data Link protocols are classified into three categories, as given below −

- Protocol 1 − Unrestricted simplex protocol
- Protocol 2 − Simplex stop and wait protocol
- Protocol 3 − Simplex protocol for noisy channels.

Unrestricted Simplex Protocol

Data transmitting is carried out in one direction only. The sender and receiver are always ready and the processing time can be ignored. In this protocol, infinite buffer space is available, and no errors are occurring that is no damage frames and no lost frames.



Simplex Stop and Wait protocol

In this protocol we assume that data is transmitted in one direction only. No error occurs, the receiver can only process the received information at finite rate. These assumptions imply that the transmitter cannot send frames at rate faster than the receiver can process them.

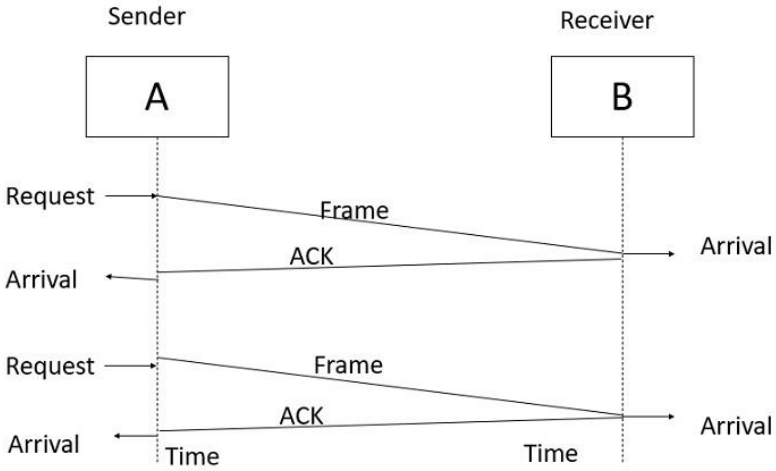
The main problem here is how to prevent the sender from flooding the receiver. The general solution for this problem is to have the receiver send some sort of feedback to sender, the process is as follows −

Step1 – Sender sends the data frame

Step 2 - The receiver send the acknowledgement frame back to the sender telling the sender that the last received frame has been processed and passed to the host.

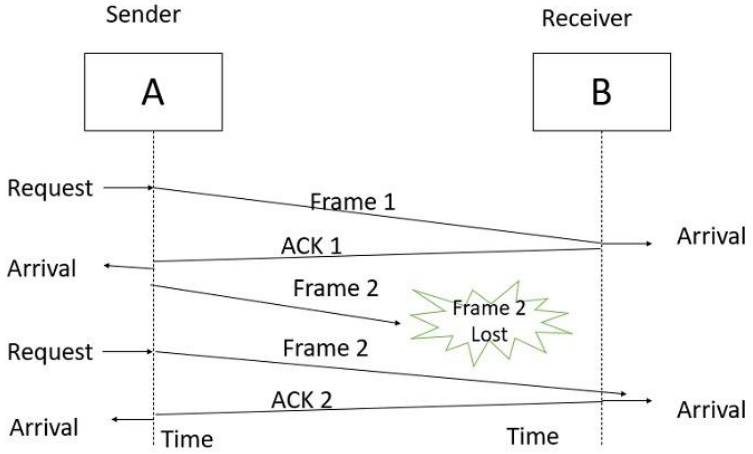
Step 3 – Permission to send the next frame is granted.

Step 4 − The sender receives the acknowledgement and then transmits the next frame.



Simplex Protocol for Noisy Channel

Data transfer is only in one direction, consider separate sender and receiver, finite processing capacity and speed at the receiver, since it is a noisy channel, errors in data frames or acknowledgement frames are expected. Every frame has a unique sequence number. It is a stop and wait protocol where after a frame has been transmitted, the timer is started for a finite time. Before the timer expires, if the acknowledgement is not received , the frame gets **retransmitted**.



Piggybacking:

In data transmission data packets acquire frames and these frames travel from sender to receiver. At the receivers end, the receiver will acquire another frame to transmit acknowledgment. Because of this process the efficiency of the channel is decreased as there is lot of bandwidth wastage of the channel to avoid this, a technique called piggybacking is introduced. " The acknowledgement here is attached to the outgoing data frame. In effect the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgement so that they can be hooked on to the next outgoing data frame is known as piggybacking".

The principal advantage of using piggybacking is a better use of available channel bandwidth. The acknowledge feed in the frame header cost only a few bits where as a separate frame would need a header, the acknowledgement and the checksum.

Sliding WindowProtocols :

Here data and control frames flow from sender to receiver in a more continuous manner and several frames can be transmitted at one time. The transmitting station maintains a sender's window that maintains the number of frames. It is permitted to send the frames to the receiving station, the receiving station also maintains a receiver's window that performs complimentary function. The two sides uses these windows to co-ordinate the flow of frames between each other. The same window is reused to transmit set of different frames. This process is also called as **pipelining**.

There are two sliding window techniques:

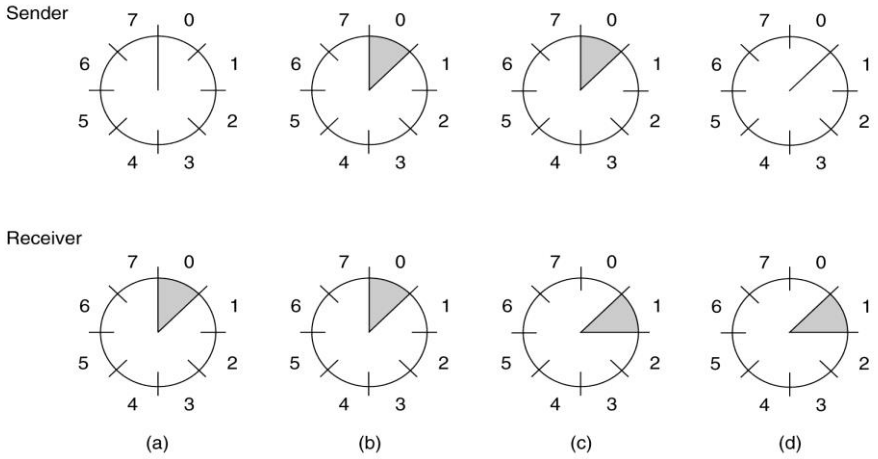
1) A one –bit Sliding Window Protocol
2) Go Back N.
3) Selective repeat.

1) A one – bit Sliding Window Protocol :

A window of size 1 is maintained by both sender and receiver. For the sender, it is 1 or for the receiver it is or 1. If the sender timed out before receiving ACK, which could be due to excessive delay, lost ACK or lost frame in the forward direction, the same frame is resent. The protocol can be implemented on a simplex channel since at any time transmission in only one direction is required.

One bit sliding window protocol is also called Stop-And-Wait protocol. In this protocol, the sender sends out one frame, waits for acknowledgment before sending next frame, thus the name Stop-And-Wait.

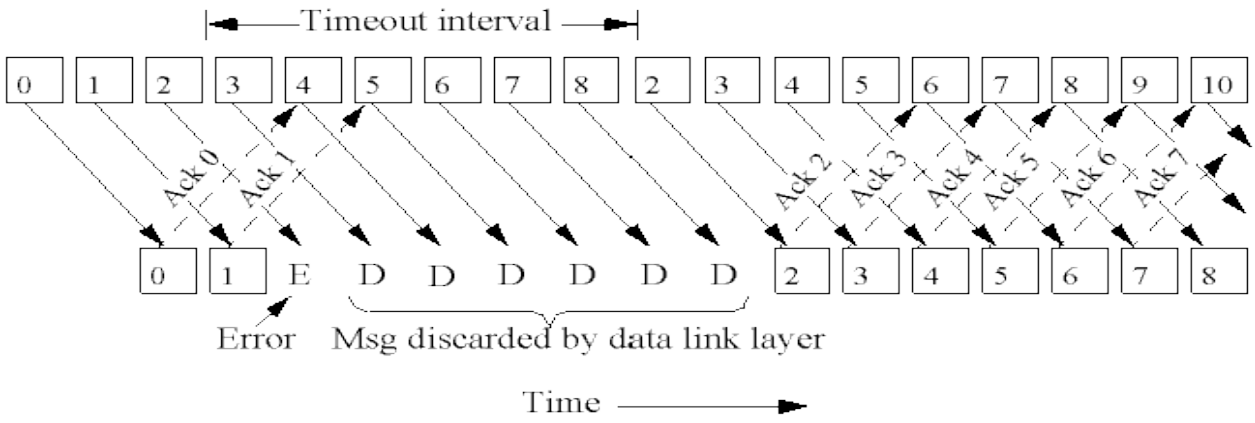
Problem with Stop-And-Wait protocol is that it is very inefficient. At any one moment, only in frame is in transition. The sender will have to wait at least one round trip time before sending next. The waiting can be long for a slow network such as satellite link



2) Go Back N sliding window protocol:

This is a sliding window technique which allows data and control messages to be transmitted continuously without waiting for its acknowledgement from the receiver. In event of error detection at the receivers side the message with error is retransmitted as well as all other frames are also retransmitted which are followed after that error message.

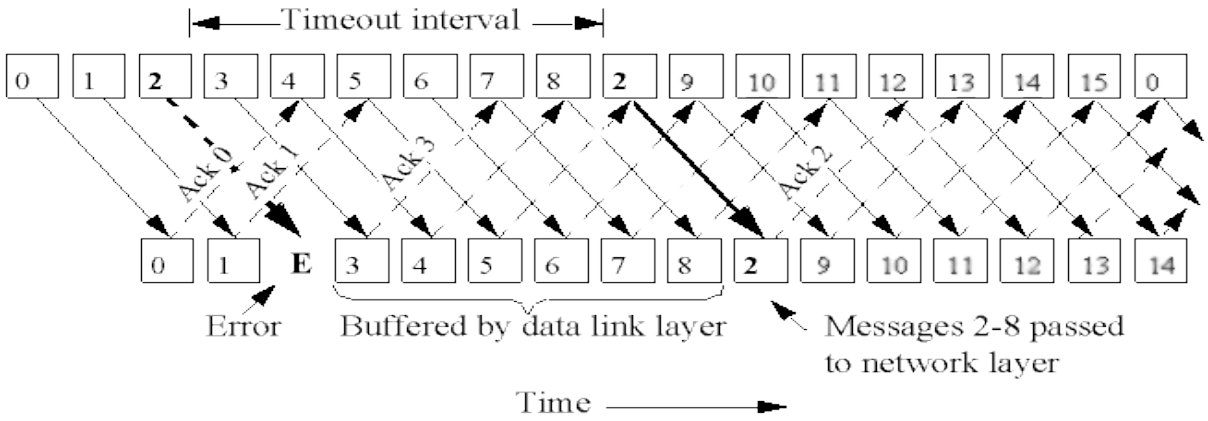
If there is one frame k missing, the receiver simply discard all subsequent frames k+1, k+2,…,sending no acknowledgments. So the sender will retransmit frames from k onwards. This can be a waste of bandwidth.



3) Selective repeat sliding window protocol:

This method provides for more refined approach. In contrast to the GoBack N, this method only retransmits the messages whose –ve acknowledgement is received. The selective repeat mechanism produces greater throughput than GoBack N. This mechanism requires additional logic to maintain the sequence of the messages and merge the retransmitted frames into proper place at the receivers end.

This method is to re-send only the ones that are actually lost or damaged. The receiver buffers all the frames after the lost one. When the sender finally noticed the problem (e.g. no ack for the lost frame is received within time-out limit), the sender retransmits the frame in question.

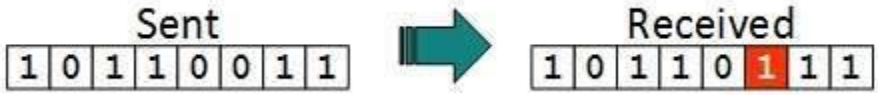


Error detection and correction

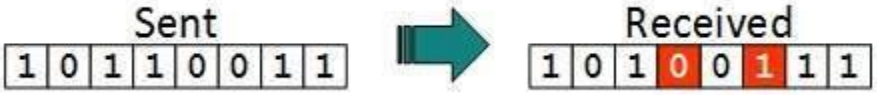
Error is a condition when the receiver's information does not match the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0. The network should ensure complete and accurate transmission of data from source node to destination node. But many times the data gets corrupted during transmission. A reliable system should have different methods to detect and correct the errors. There are several types of errors which can occur during transmission such as:

- One bit error
- Multiple bits error
- Brust error
- Lost frames or messages.

- **Single bit error :** One bit errors or single bit errors means only one bit changed in the data during transmission from the source to destination node i.e, either zero is changed to 1 or 1 is changed to zero



- **Multiple bits error :** Frame is received with more than one bits in corrupted state.



- **Burst error** :It means two or more bits of data are changed during transmission from source to destination node . Frame contains more than1 consecutive bits corrupted.

- **Lost message or lost frames :** Here the sender has sent the frame but that is not received to the destination i.e, it is lost in the n/w during transmission.

Error control mechanism may involve two possible ways :

- Error detection
- Error correction

Error detection:

During transmission the receiver should get error free data but due to some factors the data gets corrupted we need to correct these error using various techniques, for that we first require to detect these errors. For error detection there are various techniques used such as

- Single Parity Check
- Checksum
- CRC (Cyclic Redundancy Check) or Polynomial Code Checksum

Single Parity Check

In this technique, one extra bit also known as a parity bit is appended at the end of the data unit(original bits).

There are two types of parity check -

1. Even parity check

- In this technique, an extra bit is appended at the end of the data unit so that the number of 1s becomes even.
- If the number of 1s bits is odd, then parity bit 1 is appended and if the number of 1s bits is even, then parity bit 0 is appended at the end of the data unit.
- At the receiving end, if the total number of 1's are even the frame is accepted else the frame is rejected , reporting the error.

For example, If the number of 1s is odd, to make it even a bit with value 1 is added.

2. Odd parity check

- In this technique, an extra bit is appended at the end of the data unit so that the number of 1s becomes odd.
- If the number of 1s bits is even, then parity bit 1 is appended and if the number of 1s bits is odd, then parity bit 0 is appended at the end of the data unit.
- At the receiving end, if the total number of 1's are odd the frame is accepted else the frame is rejected , reporting the error.

Disadvantages

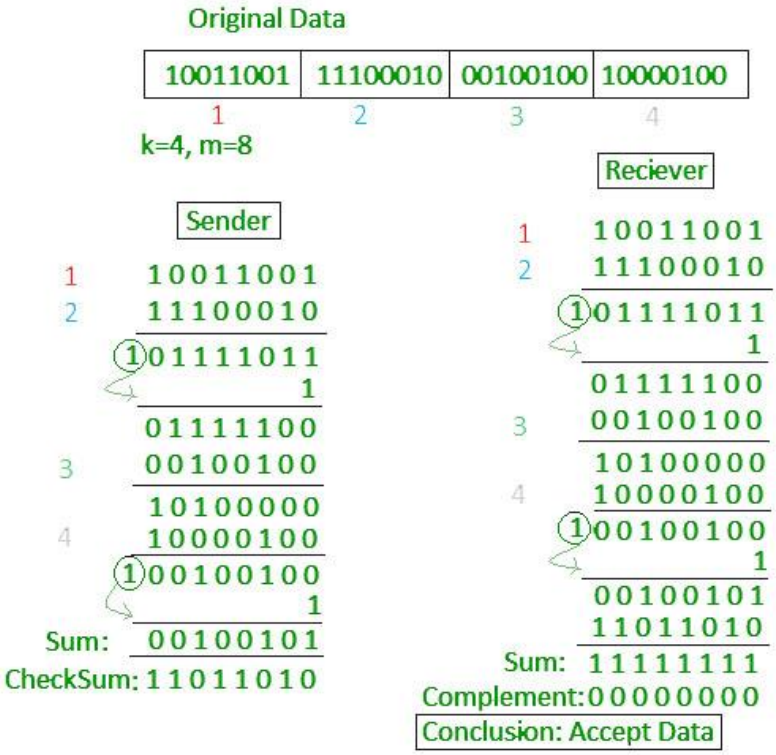
It can only detect single-bit errors. Single Parity check is not able to detect multiple bit error or burst error.

Checksum

Checksum error detection is a method used to identify errors in transmitted data.

- The process involves dividing the data into equally sized segments with n bits in each segment.
- Perform binary addition on all the segments.
- Add the carry to the sum, if any.
- Do 1's complement to the sum. Hence the result is the required checksum.
- The calculated sum is then sent along with the data to the receiver.
- At the receiver's end, the same process is repeated and if all 0's are obtained in the sum, it means that the data is correct.

Example 1 :



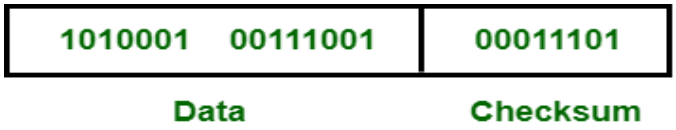
Example 2 :

If the data unit to be transmitted is 10101001 00111001, the following procedure is used at Sender site and Receiver site.

Sender Site :

10101001 subunit 1
00111001 subunit 2
11100010 sum (using 1s complement)
00011101 checksum (complement of sum)

Data transmitted to Receiver is –



Receiver Site :

10101001 subunit 1
00111001 subunit 2
00011101 checksum
11111111 sum
00000000 sum's complement

Result is zero, it means no error.

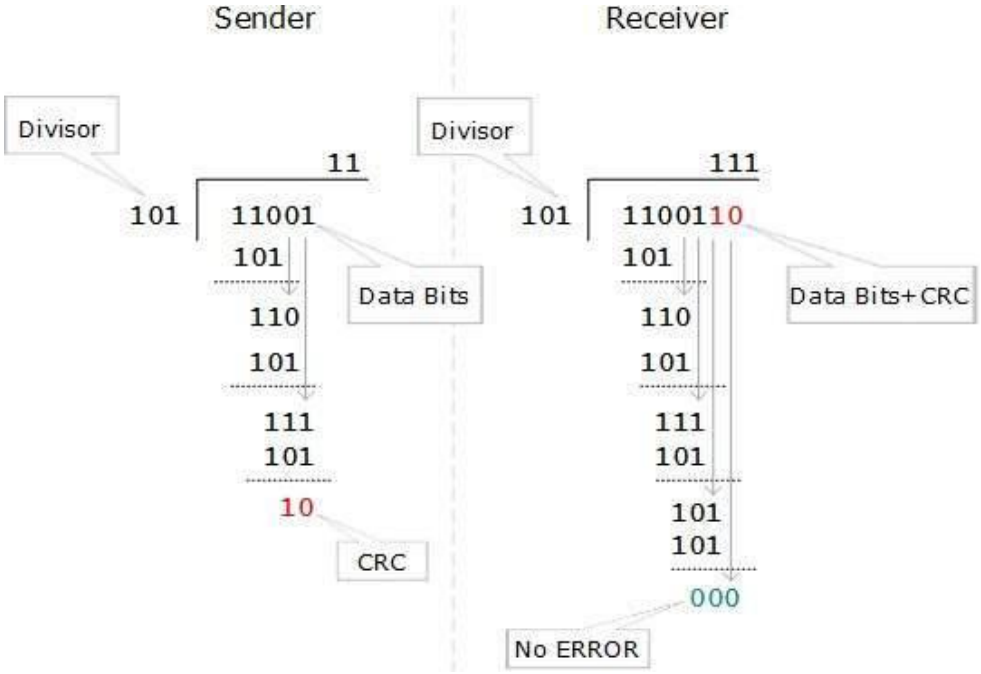
Disadvantages :

If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged.

Cyclic Redundancy Check (CRC) or Polynomial Code Checksum :

- CRC is based on binary division.
- The divisor is generated using polynomials.
- The sender performs a division operation on the bits being sent and calculates the remainder, called cyclic redundancy check bits.
- The sender adds the remainder at the end of the actual bits. Actual data bits plus the remainder is called a codeword. The sender transmits data bits as codewords.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

Example :



Example : Assume that g(x) is CRC = $X^4 + X + 1$, and the source data M is 10110011.

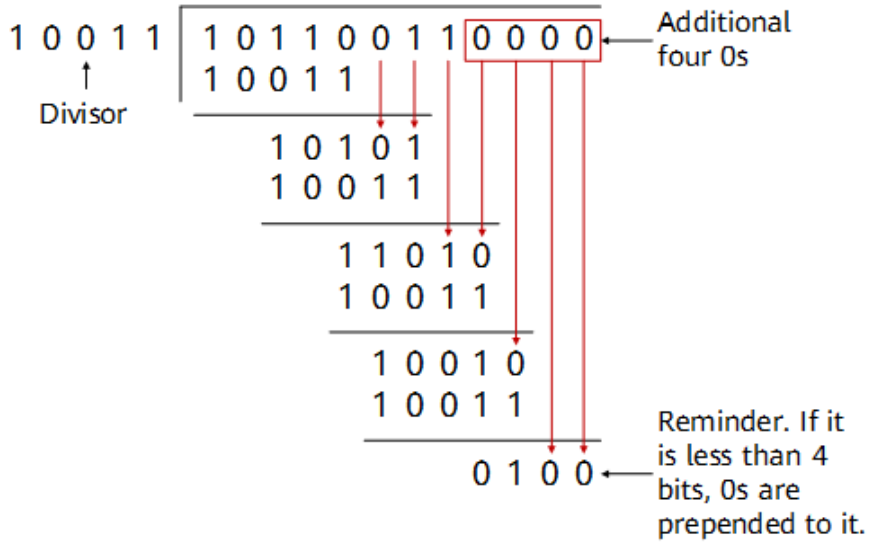
Solution :

CRC = $x^4 + x + 1$

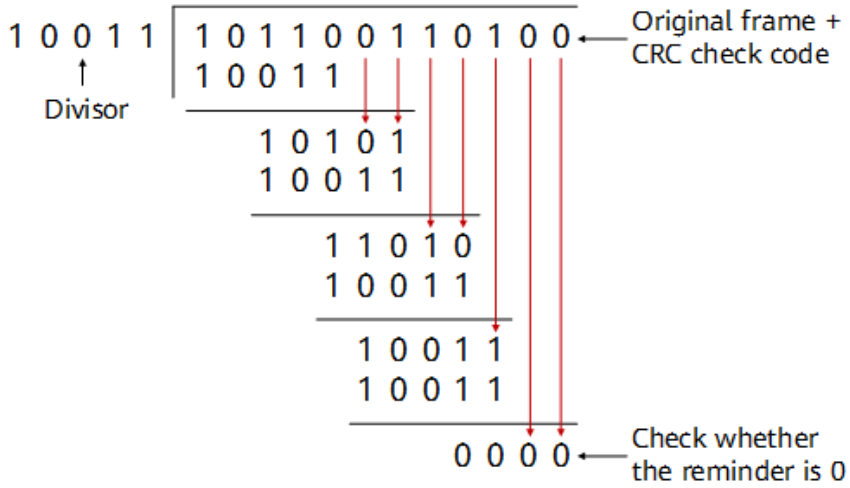
$=1*x^4 + 0*x^3 + 0*x^2 + 1*x^1 + 1$

$= 10011$

In this case, the divisor P is 10011.
The sender shifts M leftward by four bits, and divides the resulting number by P.



The remainder is the CRC check code, which is 0100 in this example. The transmit end appends 0100 to the original data frame 10110011 to generate a new frame 101100110100, and sends the new frame to the receive end. When receiving this frame, the receive end divides the frame by the divisor P, and considers the frame correct if the division leaves no remainder.



Error Correction

In the digital world, error correction can be done in two ways:

- **Backward Error Correction** When the receiver detects an error in the data received, it requests back the sender to retransmit the data unit.
- **Forward Error Correction** When the receiver detects some error in the data received, it executes error-correcting code, which helps it to auto-recover and to correct some kinds of errors.

Network designers have to develop different strategies for dealing with the errors. Once the errors are detected. These errors will be corrected at the receivers end by implementing various techniques such as

- Hamming code correction
- Parity bit correction

Hamming code correction:

Hamming code is a set of error-correction code s that can be used to detect and correct bit errors that can occur when computer data is moved or stored. Hamming code is named for R.W. Hamming of Bell Labs. Like other error-correction code, Hamming code makes use of the concept of parity and parity bit s, which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission. Using more than one parity bit, an error-correction code can not only identify a single bit error in the data unit, but also its location in the data unit.

In data transmission, the ability of a receiving station to correct errors in the received data is called forward error correction (FEC) and can increase throughput on a data link when there is a lot of noise present. To enable this, a transmitting station must add extra data (called error correction bits) to the transmission. However, the correction may not always represent a cost saving over that of simply resending the information. Hamming codes make FEC less expensive to implement through the use of a block parity mechanism.

Computing parity involves counting the number of ones in a unit of data, and adding either a zero or a one (called a parity bit) to make the count odd (for odd parity) or even (for even parity). For example, 1001 is a 4-bit data unit containing two one bits; since that is an even number, a zero would be added to maintain even parity, or, if odd parity was being maintained, another one would be added. To calculate even parity, the XOR operator is used; to calculate odd parity, the XNOR operator is used. Single bit errors are detected when the parity count indicates that the number of ones is incorrect, indicating that a data bit has been flipped by noise in the line. Hamming codes detect two bit errors by using more than one parity bit, each of which is computed on different combinations of bits in the data. The number of parity bits required depends on the number of bits in the data transmission, and is calculated by the Hamming rule.

Parity bit correction:

In communications, parity checking refers to the use of parity bits to check that data has been transmitted accurately. The parity bit is added to every data unit (typically seven or eight bits) that are transmitted. The parity bit for each unit is set so that all bytes have either an odd number or an even number of set bits.

Assume, for example, that two devices are communicating with even parity(the most common form of parity checking). As the transmitting device sends data, it counts the number of set bits in each group of seven bits. If the number of set bits is even, it sets the parity bit to 0; if

the number of set bits is odd, it sets the parity bit to 1. In this way, every byte has an even number of set bits. On the receiving side, the device checks each byte to make sure that it has an even number of set bits. If it finds an odd number of set bits, the receiver knows there was an error during transmission.