VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC** with **A++** Grade, **ISO 9001:2015** Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India.

Department of Artificial Intelligence and Machine Learning
III B.Tech I Sem (R-22)
Course: CN (A8519)

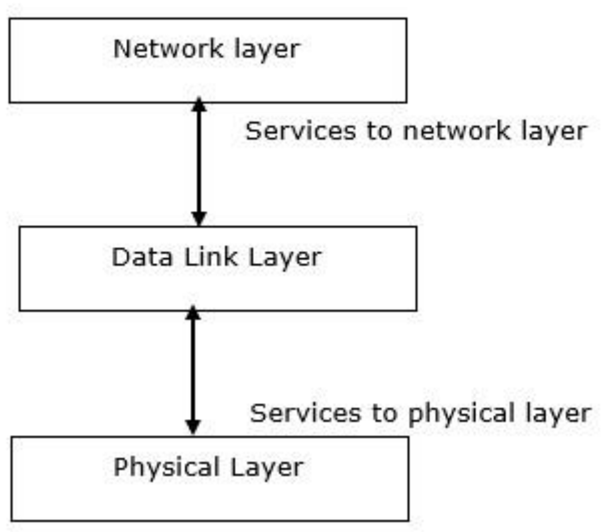
UNIT-II

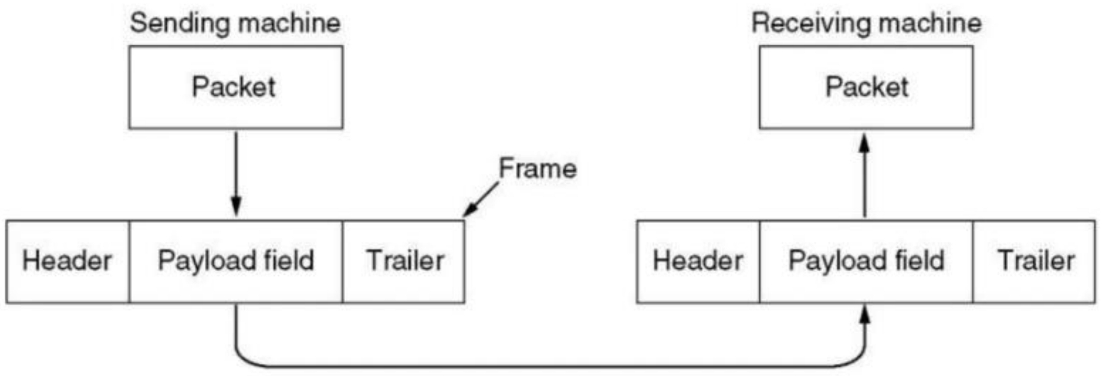
CHAPTER-I

DATA LINK LAYER DESIGN ISSUES

Data-link layer is the second layer after the physical layer. The data link layer is responsible for maintaining the data link between two hosts or nodes.

The **Data Link layer** is located between **physical** and **network** layers. It provides services to the Network layer and it receives services from the physical layer. The scope of the data link layer is node-to-node.





Relationship between packets and frames.

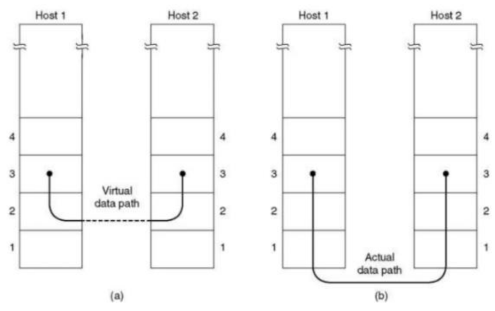
The following are the design issues in the Data Link Layer.

- The services that are provided to the Network layer.
- Framing
- Error control
- Flow control

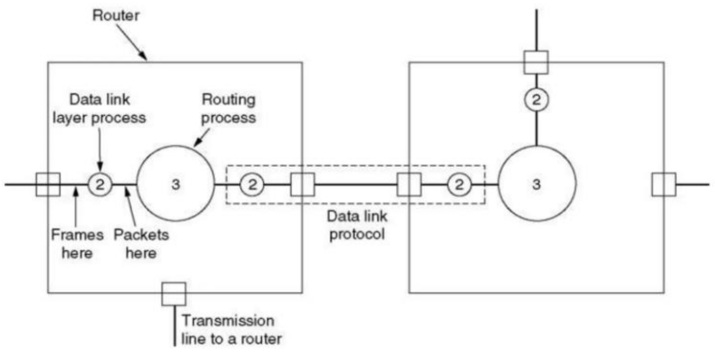
1. **Services provided to the network layer**

The data link layer act as a service interface to the network layer. The principle service is transferring data from network layer on sending machine to the network layer on destination machine. This transfer also takes place via DLL (Data link-layer).

Services Provided to Network Layer



(a) Virtual communication.
(b) Actual communication.



Placement of the data link protocol.

It provides three types of services:

1. Unacknowlwdged and connectionless services.
2. Acknowledged and connectionless services.
3. Acknowledged and connection-oriented services

Unacknowledged and connectionless services.

- Here the sender machine sends the independent frames without any acknowledgement from the sender.
- There is no logical connection established.

Acknowledged and connectionless services.

- There is no logical connection between sender and receiver established.
- Each frame is acknowledged by the receiver.
- If the frame didn't reach the receiver in a specific time interval it has to be sent again.
- It is very useful in wireless systems.

Acknowledged and connection-oriented services

- A logical connection is established between sender and receiver before data is trimester.
- Each frame is numbered so the receiver can ensure all frames have arrived and exactly once.

2. Framing :-

❖ Framing is the function of a data link layer that provides a way for a sender to transmit a set of bits that are meaningful to the receiver.

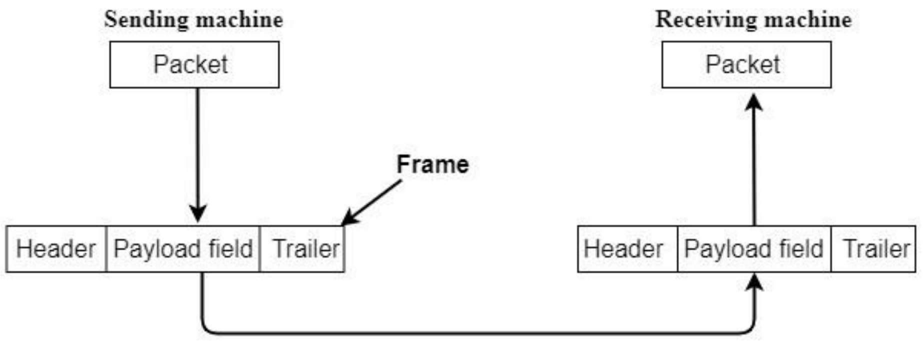
❖ To provide service to the network layer, the data link layer must use the service provided to it by the physical layer.

❖ The physical layer accepts a raw bit stream and attempts to deliver it to the destination.

❖ The usual approach is for the data link layer to break up the bit stream into discrete frames, compute the short token called a checksum for each frame, and include the checksum in the frame when it is transmitted.

The Frame contains the following –

- Frame Header
- Payload field for holding packet
- Frame Trailer



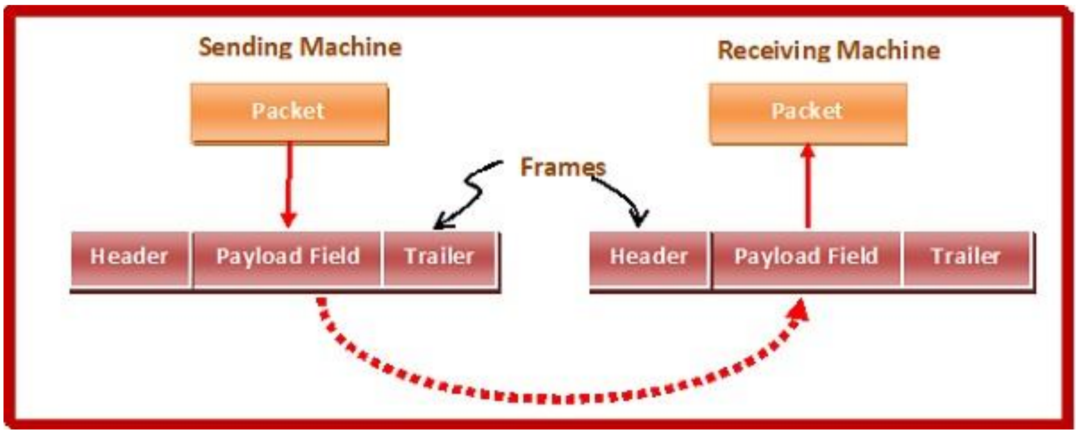
There are four framing methods used for framing.

1. Byte Count
2. Flag byte with byte stuffing
3. Flag bits with bit stuffing
4. Physical layer coding violations.

In the **physical layer**, data transmission involves synchronised transmission of bits from the source to the destination. The data link layer packs these bits into frames.

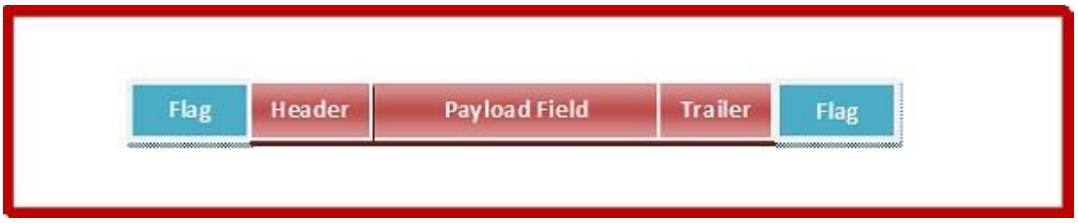
Data-link layer takes the packets from the **Network Layer** and encapsulates them into frames. If the frame size becomes too large, then the packet may be divided into small sized frames. Smaller sized frames makes flow control and error control more efficient.

Then, it sends each frame bit-by-bit on the hardware. At receiver's end, data link layer picks up signals from hardware and assembles them into frames.



A frame has the following parts −

- **Frame Header** − It contains the source and the destination addresses of the frame.
- **Payload field** − It contains the message to be delivered.
- **Trailer** − It contains the error detection and error correction bits.
- **Flag** − It marks the beginning and end of the frame.



Types of Framing

Framing can be of two types, fixed sized framing and variable sized framing.

❖ **Fixed-sized Framing**

Here the size of the frame is fixed and so the frame length acts as delimiter of the frame. Consequently, it does not require additional boundary bits to identify the start and end of the frame.

Example − ATM cells.

❖ **Variable – Sized Framing**

Here, the size of each frame to be transmitted may be different. So additional mechanisms are kept to mark the end of one frame and the beginning of the next frame.

It is used in local area networks.

Two ways to define frame delimiters in variable sized framing are −

- **Length Field** − Here, a length field is used that determines the size of the frame. It is used in Ethernet (IEEE 802.3).
- **End Delimiter** − Here, a pattern is used as a delimiter to determine the size of frame. It is used in Token Rings. If the pattern occurs in the message, then two approaches are used to avoid the situation −
  - **Byte – Stuffing** − A byte is stuffed in the message to differentiate from the delimiter. This is also called character-oriented framing.
  - **Bit – Stuffing** − A pattern of bits of arbitrary length is stuffed in the message to differentiate from the delimiter. This is also called bit – oriented framing.

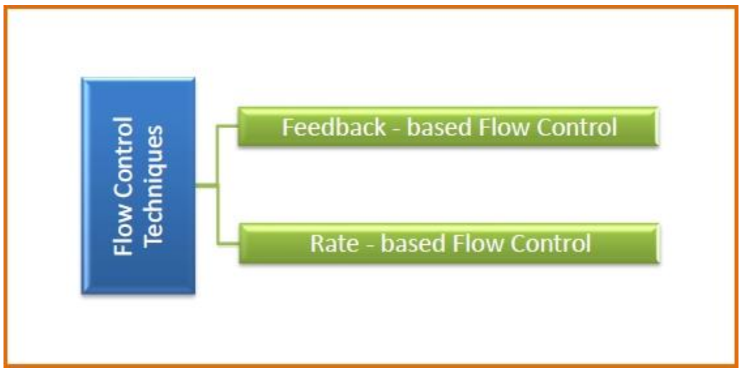
3. Flow control –

Flow control is done to prevent the flow of data frame at the receiver end. The source machine must not send data frames at a rate faster than the capacity of destination machine to accept them.

Flow control is a technique that allows two stations working at different speeds to communicate with each other. It is a set of measures taken to regulate the amount of data that a sender sends so that a fast sender does not overwhelm a slow receiver. In **data link layer**, flow control restricts the number of frames the sender can send before it waits for an acknowledgment from the receiver.

Approaches of Flow Control

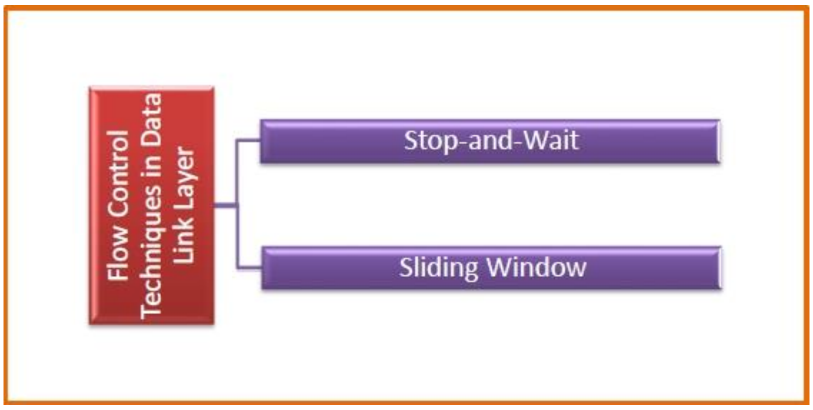
Flow control can be broadly classified into two categories −



- **Feedback based Flow Control** In these protocols, the sender sends frames after it has received acknowledgments from the user. This is used in the **data link layer.**
- **Rate based Flow Control** These protocols have built in mechanisms to restrict the rate of transmission of data without requiring acknowledgment from the receiver. This is used in the **network layer and the transport layer.**

Flow Control Techniques in Data Link Layer

Data link layer uses feedback based flow control mechanisms. There are two main techniques.



<u>Stop and Wait :-</u>

This protocol involves the following transitions −

- The sender sends a frame and waits for acknowledgment.
- Once the receiver receives the frame, it sends an acknowledgment frame back to the sender.
- On receiving the acknowledgment frame, the sender understands that the receiver is ready to accept the next frame. So it sender the next frame in queue.

<u>Sliding Window :-</u>

This protocol improves the efficiency of stop and wait protocol by allowing multiple frames to be transmitted before receiving an acknowledgment.

The working principle of this protocol can be described as follows −

- Both the sender and the receiver has finite sized buffers called windows. The sender and the receiver agrees upon the number of frames to be sent based upon the buffer size.
- The sender sends multiple frames in a sequence, without waiting for acknowledgment. When its sending window is filled, it waits for acknowledgment. On receiving acknowledgment, it advances the window and transmits the next frames, according to the number of acknowledgments received.

4. <u>Error control</u> –

Error control is done to prevent duplication of frames. The errors introduced during transmission from source to destination machines must be detected and corrected at the destination machine.

- At the sending node, a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame.
- Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The error needs first to be detected, after detection it needs to be either corrected by the receiver node or discarded and retransmitted by the sending node.

ERROR DETECTION AND CORRECTION

Data-link layer uses error control techniques to ensure that frames, i.e. bit streams of data, are transmitted from the source to the destination with a certain extent of accuracy.

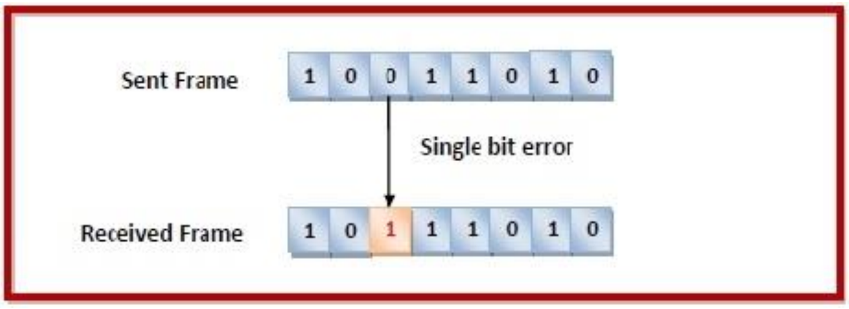
Errors

When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems. The corrupted bits leads to spurious data being received by the destination and are called errors.

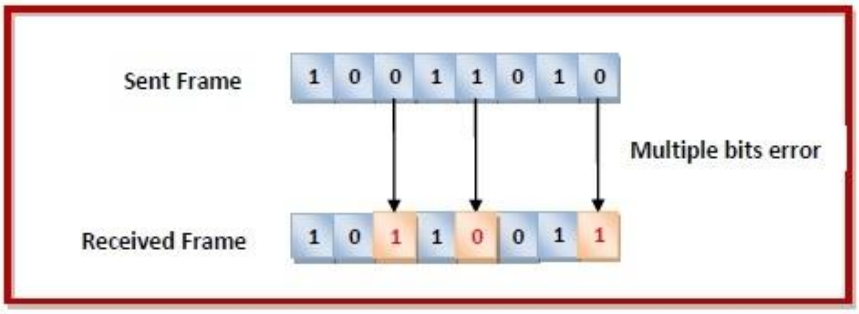
Types of Errors

Errors can be of three types, namely single bit errors, multiple bit errors, and burst errors.

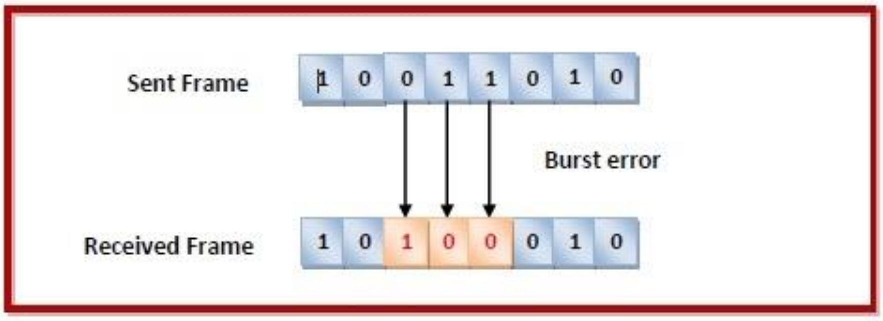
- **Single bit error** − In the received frame, only one bit has been corrupted, i.e. either changed from 0 to 1 or from 1 to 0.



- **Multiple bits error** − In the received frame, more than one bits are corrupted.



- **Burst error** − In the received frame, more than one consecutive bits are corrupted.



Error Control

Error control can be done in two ways

- **Error detection** − Error detection involves checking whether any error has occurred or not. The number of error bits and the type of error does not matter.
- **Error correction** − Error correction involves ascertaining the exact number of bits that has been corrupted and the location of the corrupted bits.

For both error detection and error correction, the sender needs to send some additional bits along with the data bits. The receiver performs necessary checks based upon the additional redundant bits. If it finds that the data is free from errors, it removes the redundant bits before passing the message to the upper layers.

ERROR DETECTION TECHNIQUES

There are three main techniques for detecting errors in frames: **Parity Check**, **Checksum**, and **Cyclic Redundancy Check** (CRC).

Parity Check

The parity check is done by adding an extra bit, called parity bit to the data to make a number of 1s either even in case of even parity or odd in case of odd parity.

While creating a frame, the sender counts the number of 1s in it and adds the parity bit in the following way

- In case of even parity: If a number of 1s is even then parity bit value is 0. If the number of 1s is odd then parity bit value is 1.
- In case of odd parity: If a number of 1s is odd then parity bit value is 0. If a number of 1s is even then parity bit value is 1.

On receiving a frame, the receiver counts the number of 1s in it. In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise, it is rejected. A similar rule is adopted for odd parity check.

NOTE:- The parity check is suitable for single bit error detection only.

If a single bit flips in transit, the receiver can detect it by counting the number of 1s. But when more than one bits are erroneous, then it is very hard for the receiver to detect the error.

Checksum

In this error detection scheme, the following procedure is applied

- Data is divided into fixed sized frames or segments.
- The sender adds the segments using 1’s complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
- The receiver adds the incoming segments along with the checksum using 1’s complement arithmetic to get the sum and then complements it.
- If the result is zero, the received frames are accepted; otherwise, they are discarded.

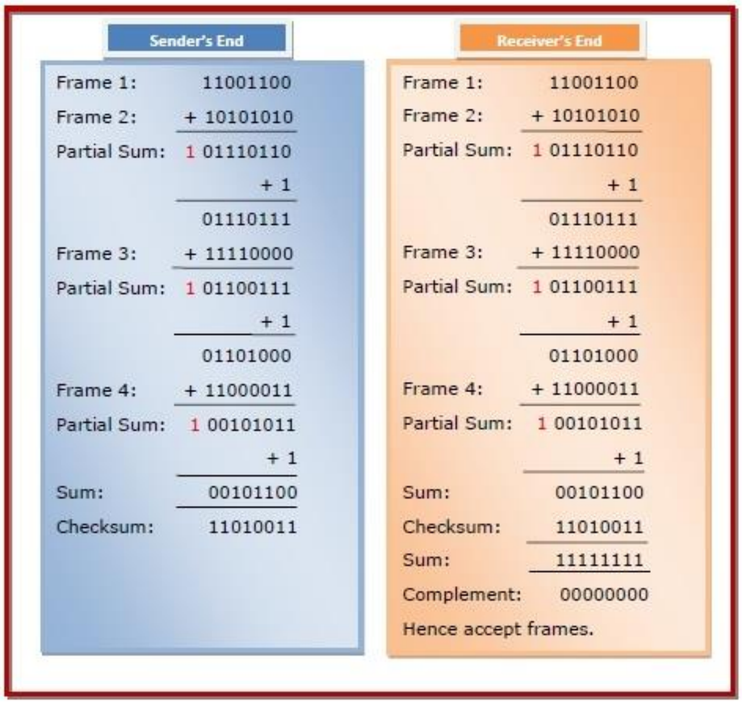
Example

Suppose that the sender wants to send 4 frames each of 8 bits, where the frames are 11001100, 10101010, 11110000 and 11000011.

The sender adds the bits using 1s complement arithmetic. While adding two numbers using 1s complement arithmetic, if there is a carry over, it is added to the sum.

After adding all the 4 frames, the sender complements the sum to get the checksum, 11010011, and sends it along with the data frames.

The receiver performs 1s complement arithmetic sum of all the frames including the checksum. The result is complemented and found to be 0. Hence, the receiver assumes that no error has occurred.

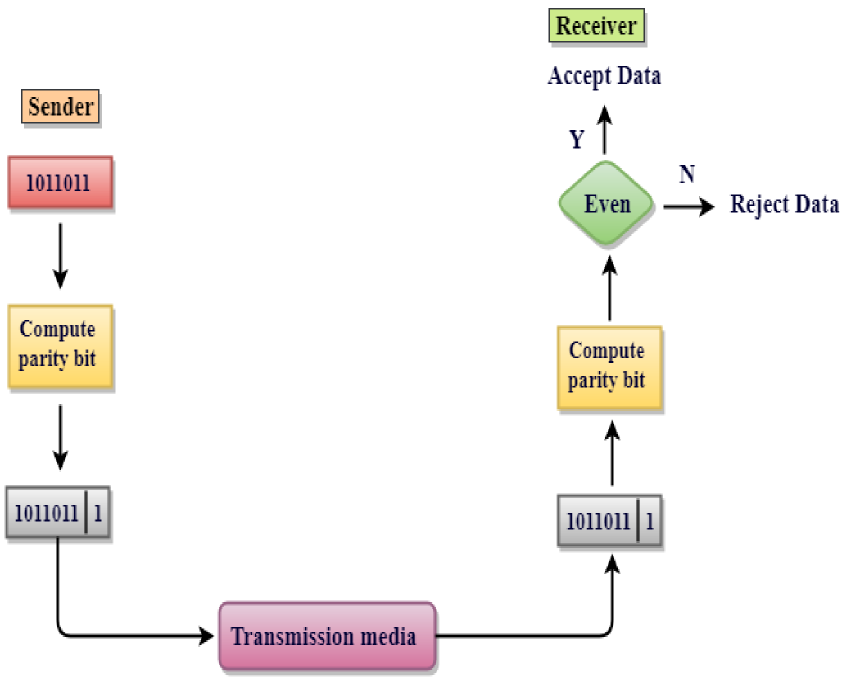


```
Sender's End                          Receiver's End

Frame 1:         11001100             Frame 1:         11001100
Frame 2:       + 10101010             Frame 2:       + 10101010
Partial Sum:   1 01110110             Partial Sum:   1 01110110
                      + 1                                   + 1
                 01110111                              01110111
Frame 3:       + 11110000             Frame 3:       + 11110000
Partial Sum:   1 01100111             Partial Sum:   1 01100111
                      + 1                                   + 1
                 01101000                              01101000
Frame 4:       + 11000011             Frame 4:       + 11000011
Partial Sum:   1 00101011             Partial Sum:   1 00101011
                      + 1                                   + 1
Sum:             00101100             Sum:             00101100
Checksum:        11010011             Checksum:        11010011
                                      Sum:             11111111
                                      Complement:      00000000
                                      Hence accept frames.
```

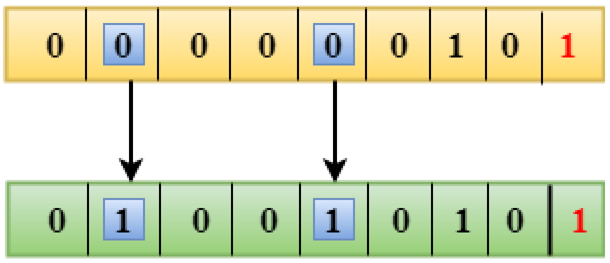
Single Parity Check

- Single Parity checking is the simple mechanism and inexpensive to detect the errors.
- In this technique, a redundant bit is also known as a parity bit which is appended at the end of the data unit so that the number of 1s becomes even. Therefore, the total number of transmitted bits would be 9 bits.
- 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, the parity bit is calculated from the received data bits and compared with the received parity bit.
- This technique generates the total number of 1s even, so it is known as even-parity checking.



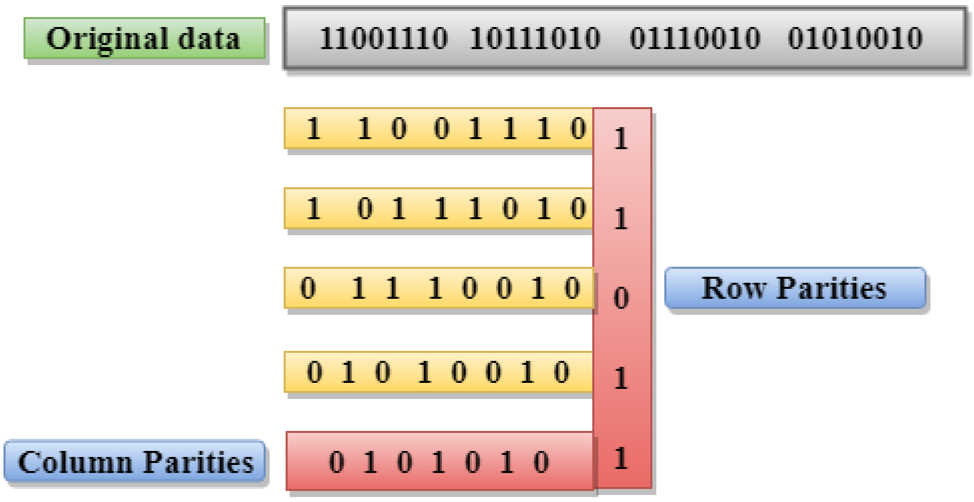
Drawbacks Of Single Parity Checking

- It can only detect single-bit errors which are very rare.
- If two bits are interchanged, then it cannot detect the errors.



Two-Dimensional Parity Check

- Performance can be improved by using **Two-Dimensional Parity Check** which organizes the data in the form of a table.
- Parity check bits are computed for each row, which is equivalent to the single-parity check.
- In Two-Dimensional Parity check, a block of bits is divided into rows, and the redundant row of bits is added to the whole block.
- At the receiving end, the parity bits are compared with the parity bits computed from the received data.



Drawbacks Of 2D Parity Check

- If two bits in one data unit are corrupted and two bits exactly the same position in another data unit are also corrupted, then 2D Parity checker will not be able to detect the error.
- This technique cannot be used to detect the 4-bit errors or more in some cases.

Checksum

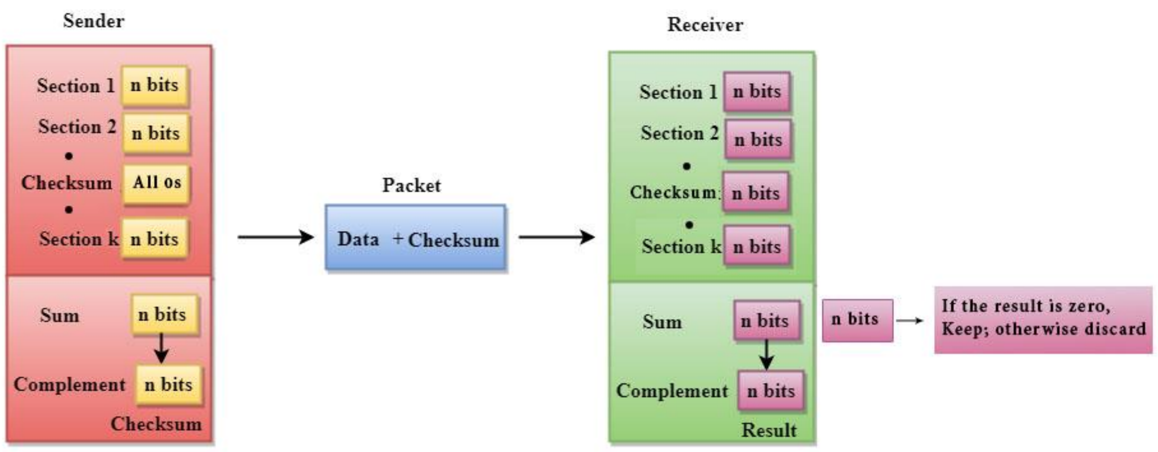
A Checksum is an error detection technique based on the concept of redundancy.

It is divided into two parts:

Checksum Generator

A Checksum is generated at the sending side. Checksum generator subdivides the data into equal segments of n bits each, and all these segments are added together by using one's complement arithmetic. The sum is complemented and appended to the original data, known as checksum field. The extended data is transmitted across the network.

Suppose L is the total sum of the data segments, then the checksum would be ?L



The Sender follows the given steps:

1. The block unit is divided into k sections, and each of n bits.
2. All the k sections are added together by using one's complement to get the sum.
3. The sum is complemented and it becomes the checksum field.
4. The original data and checksum field are sent across the network.

Checksum Checker

A Checksum is verified at the receiving side. The receiver subdivides the incoming data into equal segments of n bits each, and all these segments are added together, and then this sum is complemented. If the complement of the sum is zero, then the data is accepted otherwise data is rejected.

The Receiver follows the given steps:

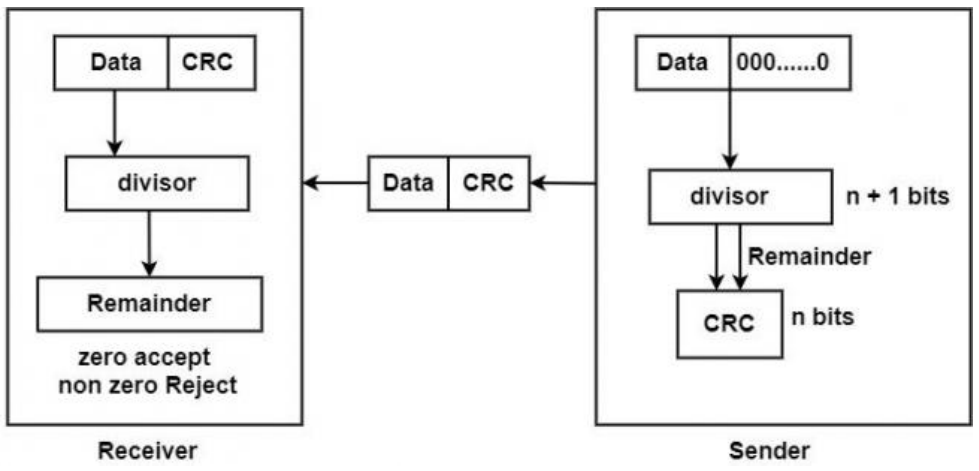
1. The block unit is divided into k sections and each of n bits.
2. All the k sections are added together by using one's complement algorithm to get the sum.
3. The sum is complemented.
4. If the result of the sum is zero, then the data is accepted otherwise the data is discard ed.

Cyclic Redundancy Check (CRC)

Cyclic Redundancy Check (CRC) involves binary division of the data bits being sent by a predetermined divisor agreed upon by the communicating system. The divisor is generated using polynomials.

- Here, the sender performs binary division of the data segment by the divisor. It then appends the remainder called CRC bits to the end of the data segment. This makes the resulting data unit exactly divisible by the divisor.

- The receiver divides the incoming data unit by the divisor. If there is no remainder, the data unit is assumed to be correct and is accepted. Otherwise, it is understood that the data is corrupted and is therefore rejected.



CRC is a different approach to detect if the received frame contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual 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.

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communicationchannel.
CRC uses **Generator Polynomial** which is available on both sender and receiver side. An example generator polynomial is of the form like $x^3 + x + 1$. This generator polynomial represents key 1011. Another example is $x^2 + 1$ that represents key 101.

n : Number of bits in data to be sent from sender side.

k : Number of bits in the key obtained from generator polynomial.

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

1. The binary data is first augmented by adding k-1 zeros in the end of the data
2. Use ***modulo-2 binary division*** to divide binary data by the key and store remainder of division.
3. Append the remainder at the end of the data to form the encoded data and send the same

Receiver Side (Check if there are errors introduced in transmission) Perform modulo-2 division again and if the remainder is 0, then there are no errors.

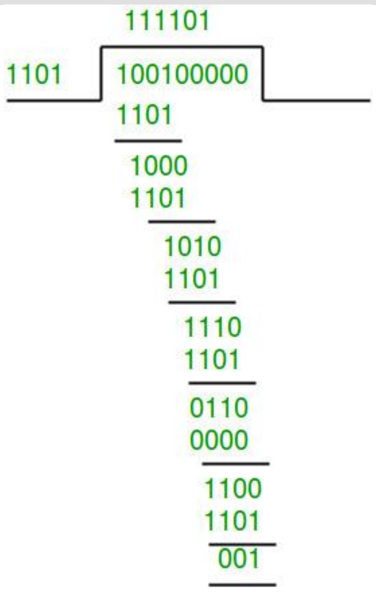
Modulo2Division:
The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. Just that instead of subtraction, we use XOR here.

- In each step, a copy of the divisor (or data) is XORed with the k bits of the dividend (or key).
- The result of the XOR operation (remainder) is (n-1) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-1)-bit remainder which is appended at the sender side.

Illustration:
Example 1 (No error in transmission):

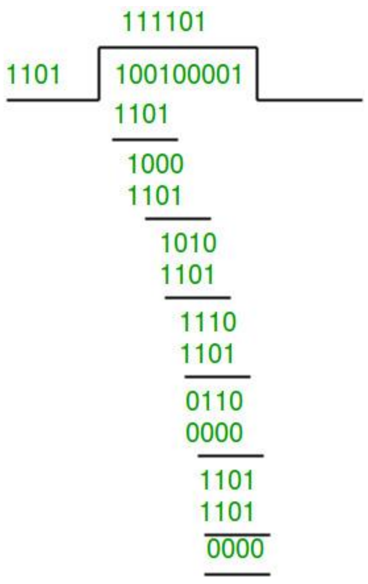
- Data word to be sent - 100100
- Key - 1101 [Or generator polynomial $x^3 + x^2 + 1$]

- Sender Side:



- Therefore, the remainder is 001 and hence the encoded
- data sent is 100100001.

- Receiver Side:
- Code word received at the receiver side 100100001

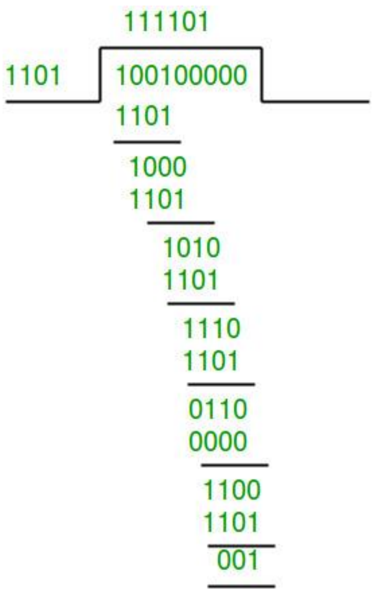


- Therefore, the remainder is all zeros. Hence, the
- data received has no error.

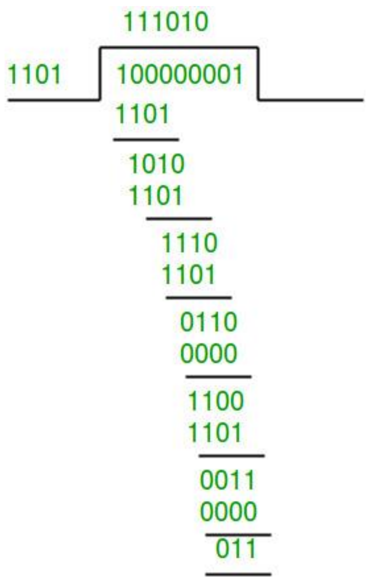
Example 2: (Error in transmission)

- Data word to be sent - 100100
- Key - 1101

- Sender Side:



- Therefore, the remainder is 001 and hence the
- code word sent is 100100001.
-
- Receiver Side
- Let there be an error in transmission media
- Code word received at the receiver side - 100000001



- Since the remainder is not all zeroes, the error is detected at the receiver side.

HAMMING DISTANCE

Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different.

The Hamming distance between two strings, a and b is denoted as d(a,b).

It is used for error detection or error correction when data is transmitted over computer networks. It is also using in coding theory for comparing equal length data words.

Calculation of Hamming Distance

In order to calculate the Hamming distance between two strings, and , we perform their XOR operation, (a⊕ b), and then count the total number of 1s in the resultant string.

Example

Suppose there are two strings 1101 1001 and 1001 1101.
11011001 ⊕ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d(11011001, 10011101) = 2.

Minimum Hamming Distance

In a set of strings of equal lengths, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of strings in that set.

Example

Suppose there are four strings 010, 011, 101 and 111.

$010 \oplus 011 = 001$, d(010, 011) = 1.
$010 \oplus 101 = 111$, d(010, 101) = 3.
$010 \oplus 111 = 101$, d(010, 111) = 2.
$011 \oplus 101 = 110$, d(011, 101) = 2.
$011 \oplus 111 = 100$, d(011, 111) = 1.
$101 \oplus 111 = 010$, d(011, 111) = 1.

Hence, the Minimum Hamming Distance, $d_{min} = 1$.

ERROR CORRECTION TECHNIQUES

Error correction techniques find out the exact number of bits that have been corrupted and as well as their locations. There are two principle ways.

- **Backward Error Correction (Retransmission)** − If the receiver detects an error in the incoming frame, it requests the sender to retransmit the frame. It is a relatively simple technique. But it can be efficiently used only where retransmitting is not expensive as in fiber optics and the time for retransmission is low relative to the requirements of the application.

- **Forward Error Correction** − If the receiver detects some error in the incoming frame, it executes error-correcting code that generates the actual frame. This saves bandwidth required for retransmission. It is inevitable in real-time systems. However, if there are too many errors, the frames need to be retransmitted.

Types of Error Correcting Codes

ECCs can be broadly categorized into two types, block codes and convolution codes.

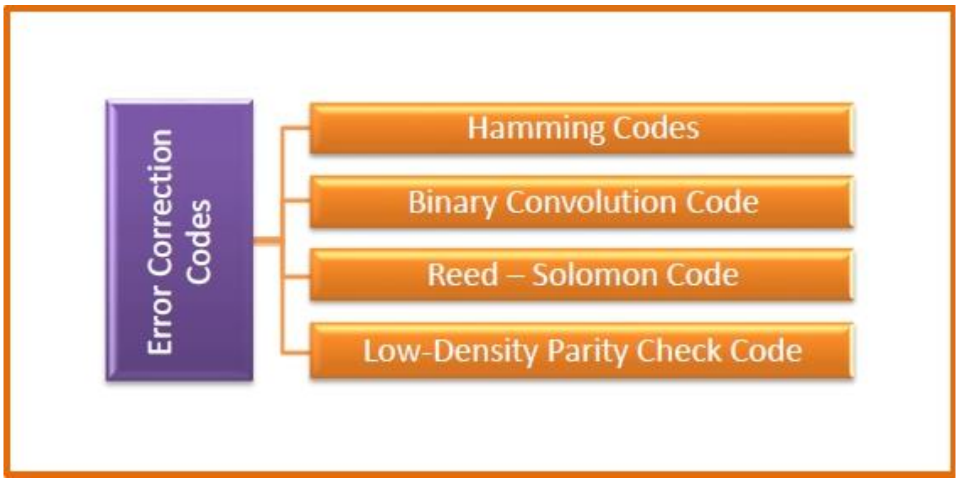
- **Block codes** − The message is divided into fixed-sized blocks of bits, to which redundant bits are added for error detection or correction.
- **Convolutional codes** − The message comprises of data streams of arbitrary length and parity symbols are generated by the sliding application of a Boolean function to the data stream.

Common Error Correcting Codes

There are four popularly used error correction codes.

- **Hamming Codes** − It is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors.
- **Binary Convolution Code** − Here, an encoder processes an input sequence of bits of arbitrary length and generates a sequence of output bits.
- **Reed - Solomon Code** − They are block codes that are capable of correcting burst errors in the received data block.

- **Low-Density Parity Check Code** − It is a block code specified by a parity-check matrix containing a low density of 1s. They are suitable for large block sizes in very noisy channels.



Hamming Code

Hamming code is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. It was developed by R.W. Hamming for error correction. In this coding method, the source encodes the message by inserting redundant bits within the message. These redundant bits are extra bits that are generated and inserted at specific positions in the message itself to enable error detection and correction. When the destination receives this message, it performs recalculations to detect errors and find the bit position that has error.

Encoding a message by Hamming Code

The procedure used by the sender to encode the message encompasses the following steps −

- **Step 1** − Calculation of the number of redundant bits.
- **Step 2** − Positioning the redundant bits.
- **Step 3** − Calculating the values of each redundant bit.

Once the redundant bits are embedded within the message, this is sent to the user.

Step 1 − Calculation of the number of redundant bits.

If the message contains mmnumber of data bits, rrnumber of redundant bits are added to it so that mr is able to indicate at least $(m + r+ 1)$ different states. Here, $(m + r)$ indicates location of an error in each of $(m + r)$ bit positions and one additional state indicates no error. Since, rr bits can indicate $2^r r$ states, $2^r r$ must be at least equal to $(m + r + 1)$. Thus the following equation should hold $2^r \geq m+r+1$

Step 2 − Positioning the redundant bits.

The *r* redundant bits placed at bit positions of powers of 2, i.e. 1, 2, 4, 8, 16 etc. They are referred in the rest of this text as r_1 (at position 1), r_2 (at position 2), r_3 (at position 4), r_4 (at position 8) and so on.

Step 3 − Calculating the values of each redundant bit.

The redundant bits are parity bits. A parity bit is an extra bit that makes the number of 1s either even or odd. The two types of parity are −

- **Even Parity** − Here the total number of bits in the message is made even.
- **Odd Parity** − Here the total number of bits in the message is made odd.
  Each redundant bit, r_i, is calculated as the parity, generally even parity, based upon its bit position. It covers all bit positions whose binary representation includes a 1 in the ith position except the position of r_i.

Thus −

- r_1 is the parity bit for all data bits in positions whose binary representation includes a 1 in the least significant position excluding 1 (3, 5, 7, 9, 11 and so on)
- r_2 is the parity bit for all data bits in positions whose binary representation includes a 1 in the position 2 from right except 2 (3, 6, 7, 10, 11 and so on)
- r_3 is the parity bit for all data bits in positions whose binary representation includes a 1 in the position 3 from right except 4 (5-7, 12-15, 20-23 and so on)

Decoding a message in Hamming Code

Once the receiver gets an incoming message, it performs recalculations to detect errors and correct them. The steps for recalculation are −

- **Step 1** − Calculation of the number of redundant bits.
- **Step 2** − Positioning the redundant bits.
- **Step 3** − Parity checking.
- **Step 4** − Error detection and correction

Step 1 − Calculation of the number of redundant bits

Using the same formula as in encoding, the number of redundant bits are ascertained.

$2^r \geq m + r + 1$ where *m* is the number of data bits and *r* is the number of redundant bits.

Step 2 − Positioning the redundant bits

The *r* redundant bits placed at bit positions of powers of 2, i.e. 1, 2, 4, 8, 16 etc.

Step 3 − Parity checking

Parity bits are calculated based upon the data bits and the redundant bits using the same rule as during generation of c_1,c_2 ,c_3 ,c_4 etc. Thus

c_1 = parity(1, 3, 5, 7, 9, 11 and so on)
c_2 = parity(2, 3, 6, 7, 10, 11 and so on)

c_3 = parity(4-7, 12-15, 20-23 and so on)

Step 4 − Error detection and correction

The decimal equivalent of the parity bits binary values is calculated. If it is 0, there is no error. Otherwise, the decimal value gives the bit position which has error. For example, if $c_1c_2c_3c_4$ = 1001, it implies that the data bit at position 9, decimal equivalent of 1001, has error. The bit is flipped to get the correct message.

Binary Convolutional Codes

In convolutional codes, the message comprises of data streams of arbitrary length and a sequence of output bits are generated by the sliding application of Boolean functions to the data stream.

In block codes, the data comprises of a block of data of a definite length. However, in convolutional codes, the input data bits are not divided into block but are instead fed as streams of data bits, which convolve to output bits based upon the logic function of the encoder. Also, unlike block codes, where the output codeword is dependent only on the present inputs, in convolutional codes, output stream depends not only the present input bits but also only previous input bits stored in memory.

Convolutional codes were first introduced in 1955, by Elias. After that, there were many interim researches by many mathematicians. In 1973, Viterbi developed an algorithm for maximum likelihood decoding scheme, called Viterbi scheme that lead to modern convolutional codes.

Encoding by Convolutional Codes

For generating a convolutional code, the information is passed sequentially through a linear finite-state shift register. The shift register comprises of (-bit) stages and Boolean function generators.

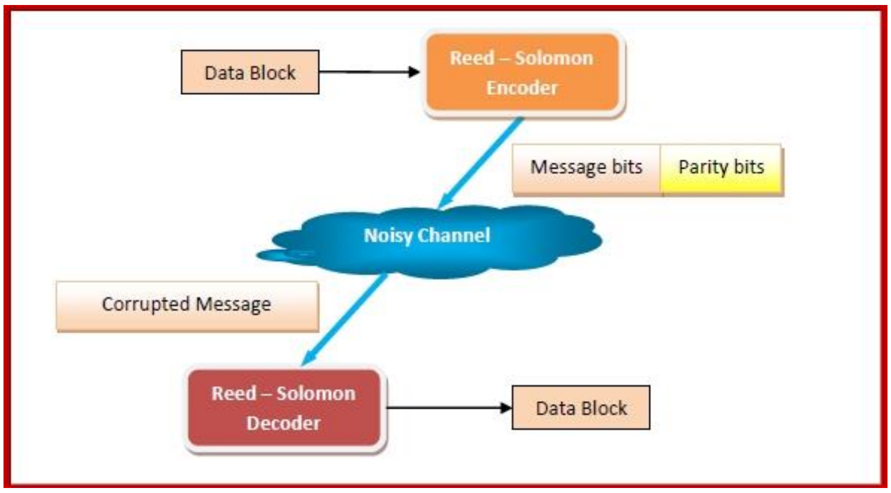
A convolutional code can be represented as (n,k, K) where

- *k* is the number of bits shifted into the encoder at one time. Generally, *k = 1*.
- n is the number of encoder output bits corresponding to k information bits.
- The code-rate, $R_c = k/n$.
- The encoder memory, a shift register of size k, is the constraint length.
- *n* is a function of the present input bits and the contents of *K*.
- The state of the encoder is given by the value of *(K - 1)* bits.

Reed - Solomon Code

Reed - Solomon error correcting codes are one of the oldest codes that were introduced in 1960s by Irving S. Reed and Gustave Solomon. It is a subclass of non - binary BCH codes. BCH codes (Bose-Chaudhuri-Hocquenghem codes) are cyclic ECCs that are constructed using polynomials over data blocks.

A Reed - Solomon encoder accepts a block of data and adds redundant bits (parity bits) before transmitting it over noisy channels. On receiving the data, a decoder corrects the error depending upon the code characteristics.



Application Areas of Reed-Solomon Codes

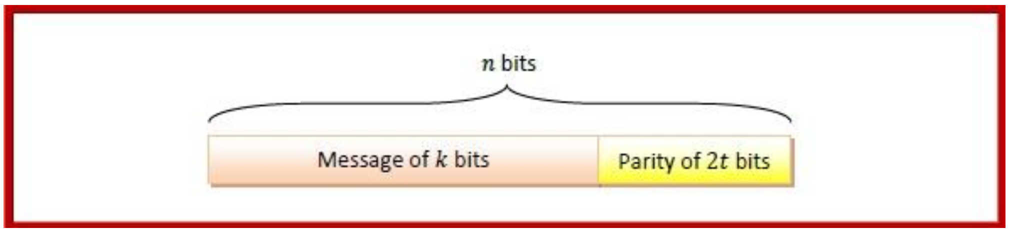
The prominent application areas are −

- Storage areas like CDs, DVDs, Blu-ray Discs
- High speed data transmission technologies such as DSL and WiMAX
- High speed modems
- QR Codes
- Broadcast systems such as satellite communications
- Storage systems such as RAID 6

Parameters of Reed - Solomon Codes

- A Reed-Solomon code is specified as RS(n,k).
- Here, n is the block length which is recognizable by symbols, holding the relation, $n = 2^m - 1$.
- The message size is of k bits.
- So the parity check size is ($n - k$) bits
- The code can correct up to (t) errors in a codeword, where ($2t = n - k$).

The following diagram shows a Reed-Solomon codeword −



Encoding using Reed Solomon Code

The method of encoding in Reed Solomon code has the following steps −

- The message is represented as a polynomial p(x), and then multiplied with the generator polynomial g(x).
- The message vector $[x_1,x_2,x_3.....x_k]$ is mapped to a polynomial of degree less than *k* such that $p_x(\alpha_i) = x_i$ for all i = 1,...k
- The polynomial is evaluated using interpolation methods like Lagrange Interpolation.
- Using this polynomial, the other points $\alpha_{k+1}....\alpha_n$, are evaluated.
- The encoded message is calculated as s(x) = p(x) * g(x). The sender sends this encoded message along with the generator polynomial g(x).

Decoding using Reed Solomon Code

At the receiving end, the following decoding procedure done −

- The receiver receives the message r(x) and divides it by the generator polynomial g(x).
- If r(x)/g(x)=0, then it implies no error.
- If r(x)/g(x)≠0, then the error polynomial is evaluated using the expression: r(x) = p(x) * g(x) + e(x)
- The error polynomial gives the error positions.

Low-Density Parity Check Codes

A low - density parity check (LFPC) code is specified by a parity-check matrix containing mostly 0s and a low density of 1s. The rows of the matrix represent the equations and the columns represent the bits in the codeword, i.e. code symbols.

A LDPC code is represented by , where is the block length, is the number of 1s in each column and is the number of 1s in each row, holding the following properties −

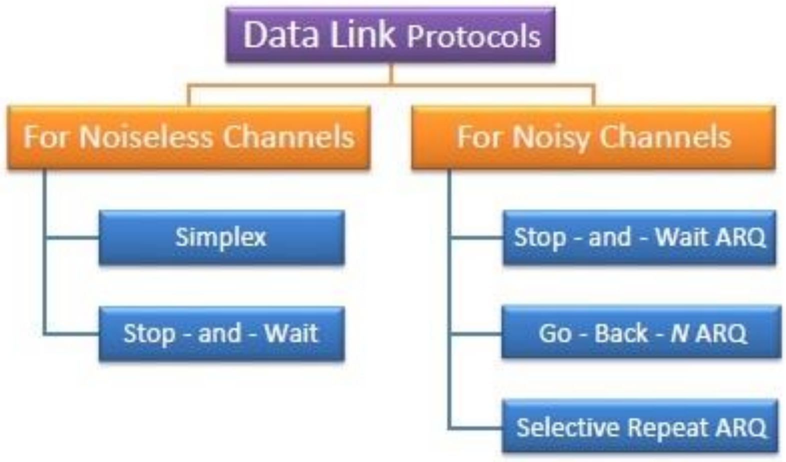
- *j* is the small fixed number of 1's in each column, where $j > 3$
- *k* is the small fixed number of 1's in each row, where $k > j$.

ELEMENTARY DATA LINK PROTOCOLS

Protocols in the **data link layer** are designed so that this layer can perform its basic functions: framing, error control and flow control. Framing is the process of dividing bit - streams from physical layer into data frames whose size ranges from a few hundred to a few thousand bytes. Error control mechanisms deals with transmission errors and retransmission of corrupted and lost frames. Flow control regulates speed of delivery and so that a fast sender does not drown a slow receiver.

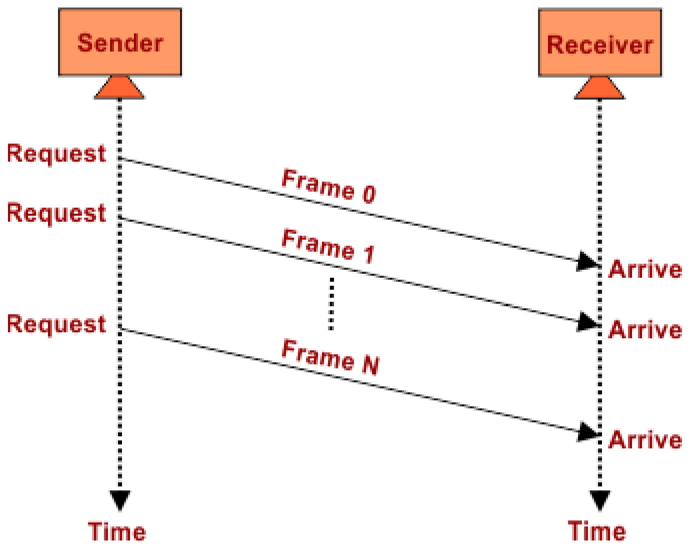
Types of Data Link Protocols

Data link protocols can be broadly divided into two categories, depending on whether the transmission channel is noiseless or noisy.



Simplex Protocol

A data flow diagram (DFD) is a graphical representation of the flow of data in a system. In the context of the simplest protocol, a DFD can illustrate the movement of data between the sender and the receiver. The DFD would show how the sender sends the data frames to the receiver, how the receiver processes the data, and what happens if any errors occur during the transfer. It could also show the absence of flow control and error control mechanisms, which are typically included in more complex protocols. The DFD can help to clarify the basic functioning of the simplest protocol, making it easier to understand and implement.



Simplest Protocol

Stop – and – Wait Protocol

Stop and wait is a protocol that is used for reliable data transmission in a noiseless channel. In this protocol, the sender sends a single packet at a time and waits for an acknowledgment (ACK) from the receiver before sending the next packet. This way, the sender can ensure that each packet is received by the receiver and has been successfully processed. If the sender does not receive an ACK within a certain time frame, the packet is considered lost and must be retransmitted.

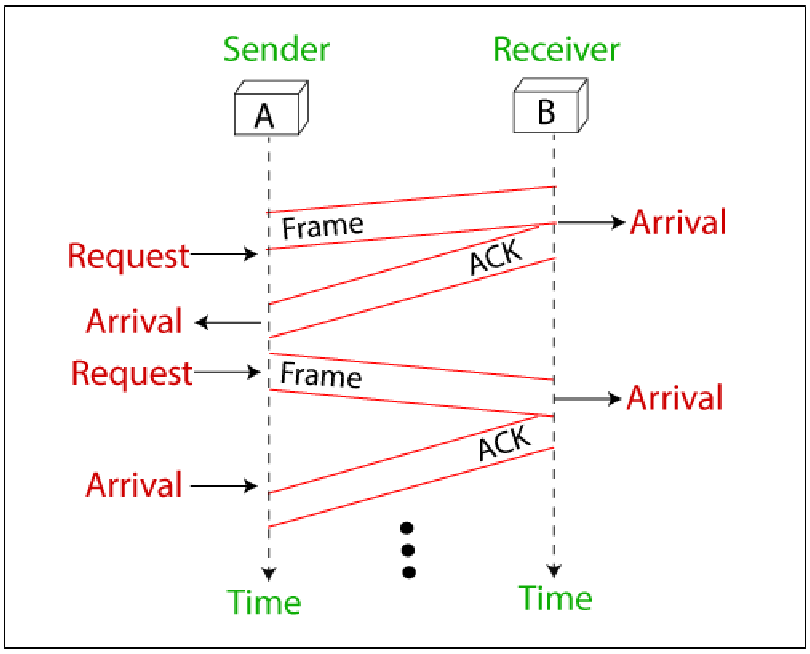
The stop and wait protocol is simple and efficient, but it has one major drawback. Because only one packet can be transmitted at a time, the overall data transmission rate is relatively slow. To overcome this limitation, the sliding window protocol was developed. In the sliding window protocol, multiple packets can be transmitted at the same time, allowing for faster data transmission. Despite this limitation, the stop and wait protocol is still widely used in many applications due to its simplicity and reliability.

- The flow of data frames at the receiver side may become too fast for it to be processed, causing the need for temporary storage.
- However, the limited storage space of the receiver may result in the loss or discarding of frames, or even denial of service.
- To prevent this, the sender must slow down their rate of transmission, which is achieved through the use of ACK messages from the receiver.
- The sender sends a single frame and waits for confirmation from the receiver before sending the next one, adding flow control to the previous protocol.
- The communication remains unidirectional for data frames, but ACK frames flow in the opposite direction.

Flow Diagram

The flow diagram of the Stop-and-wait protocol in a noiseless channel involves the following steps:

1. The sender transmits a data frame to the receiver.
2. The sender waits for an acknowledgment (ACK) from the receiver.
3. The receiver processes the received data frame.
4. The receiver sends an ACK to the sender to confirm receipt of the data frame.
5. The sender continues to transmit the next data frame, repeating the process from step i.



In this protocol, the sender sends one frame at a time and stops until it receives an ACK from the receiver. This prevents the receiver from becoming overwhelmed with incoming frames and ensures reliable data transmission. Additionally, the ACK frames are used for flow control, allowing the sender to adjust the transmission rate based on the receiver's processing capacity.

The main difference between the two protocols is that the Simplest Protocol has no flow control and error control mechanisms, while the Stop-and-Wait Protocol employs a flow control mechanism through the use of ACK frames.

In the Simplest Protocol, the recipient is always expected to be ready to receive any frames sent by the sender, so no acknowledgment is needed. In the Stop-and-Wait Protocol, the sender must wait for an acknowledgment from the receiver before sending the next frame.

Another difference between the two protocols is that the Simplest Protocol is unidirectional, while the Stop-and-Wait Protocol is bi-directional. In the Simplest Protocol, data frames can only move in one direction, from sender to receiver, while in the Stop-and-Wait Protocol, both data frames and ACK frames can travel in both directions.

Noisy Channel protocols

A Noisy Channel Protocol is a type of communication protocol that is used in communication systems where the transmission channel may introduce errors into the transmitted data. This type of protocol is designed to deal with errors in the communication channel and ensure that the data being transmitted is received accurately at the receiver end. The main objective of Noisy Channel Protocols is to minimize the error rate in the transmitted data by using techniques such as error detection and correction, flow control, and retransmission of lost or corrupted data frames. Some examples of Noisy Channel Protocols include the Stop-and-Wait Protocol, the Sliding Window Protocol, and the Automatic Repeat Request (ARQ) Protocol.

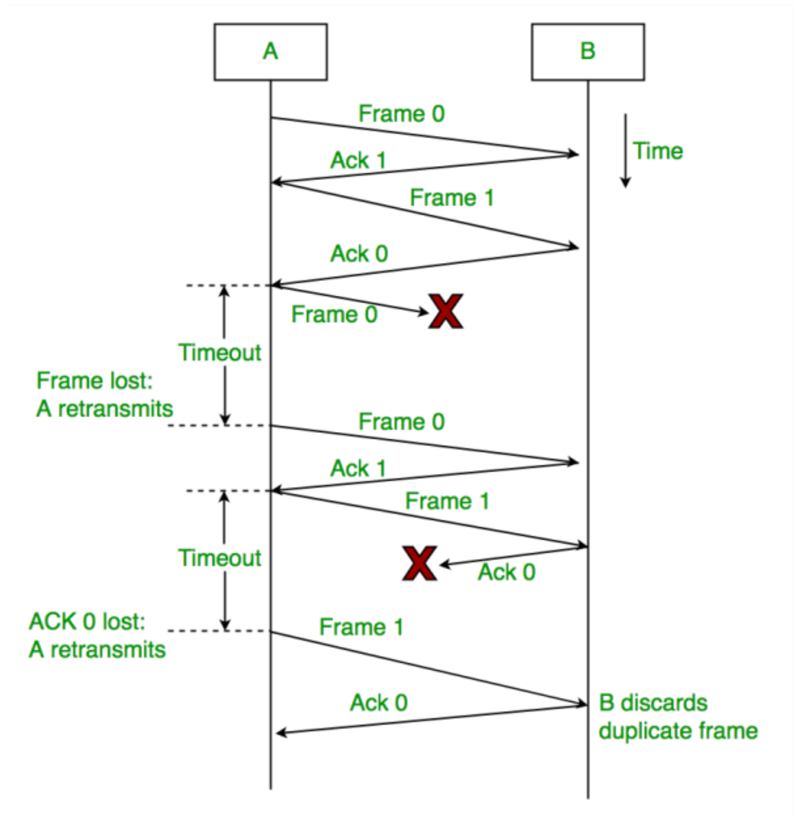
Stop – and – Wait ARQ

The Stop and Wait protocol is a protocol used for reliable data transmission over a noisy channel. In this protocol, the sender only sends one frame at a time and waits for an acknowledgment (ACK) from the receiver before sending the next frame. This helps to ensure that the receiver receives the data correctly and eliminates the need for retransmission in the case of errors caused by the noisy channel. The sender continuously monitors the channel for errors, and if an error is detected, it waits for the next ACK before resending the frame. This protocol adds error control to the basic unidirectional communication of data frames and ACK frames in the opposite direction.

Flow Diagram

A data flow diagram in the Stop-and-Wait protocol in a noisy channel can be used to describe the flow of data between the sender and the receiver. This diagram generally includes the following components:

1. ***Sender*:** The sender sends data frames one at a time, and waits for a response (ACK or NACK) from the receiver before sending the next data frame.
2. ***Receiver*:** The receiver receives the data frames and processes them. If the frame is received correctly, the receiver sends an ACK signal to the sender. If the frame is not received correctly, the receiver sends a NACK signal to the sender.
3. ***Noisy Channel*:** The noisy channel is the medium through which the data frames are transmitted from the sender to the receiver. The channel can add noise to the data frames, resulting in errors and corruption of the data.
4. ***Error Detection*:** The receiver uses error detection techniques such as checksums to detect errors in the received data frames.
5. ***Error Correction*:** If an error is detected, the receiver sends a NACK signal to the sender, requesting a retransmission of the frame.



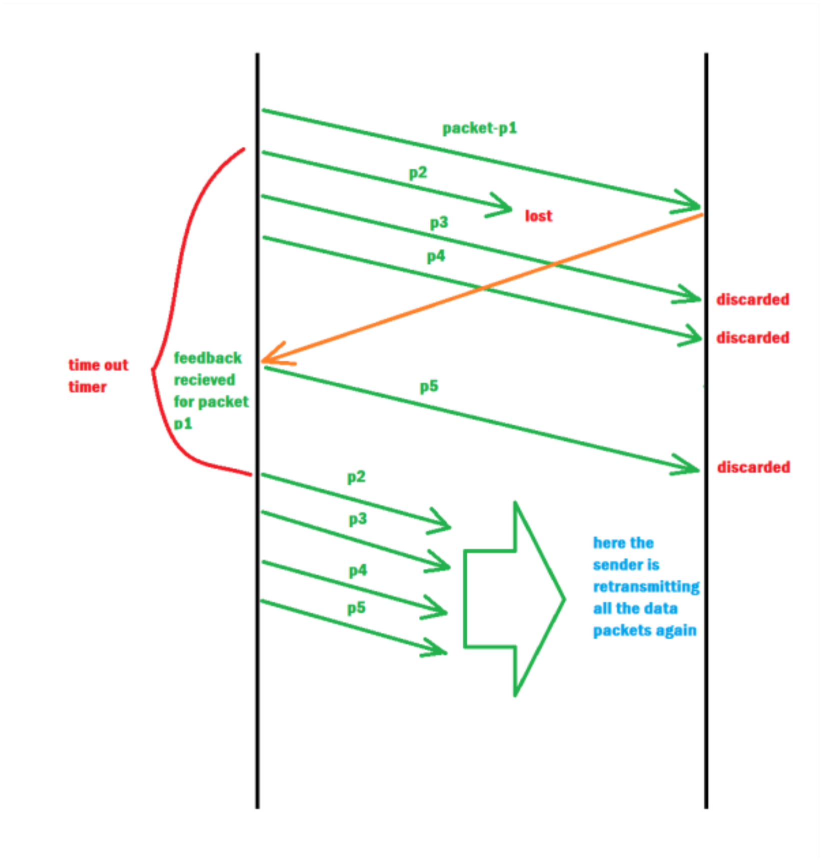
In this protocol, the sender only sends one data frame at a time and waits for a response from the receiver before sending the next frame. This ensures that the receiver has enough time to process each frame before receiving the next one. The Stop-and-Wait protocol is reliable, but has low throughput compared to other protocols.

Go – Back – N ARQ

It provides for sending multiple frames before receiving the acknowledgement for the first frame. It uses the concept of sliding window, and so is also called sliding window protocol. The frames are sequentially numbered and a finite number of frames are sent. If the acknowledgement of a frame is not received within the time period, all frames starting from that frame are retransmitted.

The Go-Back-N Automatic Repeat Request (ARQ) protocol is a type of error-control protocol used in data communication to ensure reliable delivery of data over a noisy channel. In a noisy channel, the probability of errors in the received packets is high, and hence, there is a need for a mechanism to detect and correct these errors.

The Go-Back-N ARQ protocol is a type of sliding window protocol where the sender transmits a window of packets to the receiver, and the receiver sends back an acknowledgment (ACK) to the sender indicating successful receipt of the packets. In case the sender does not receive an ACK within a specified timeout period, it retransmits the entire window of packets.


packet-p1
p2
lost
p3
p4
discarded
discarded
time out timer
feedback recieved for packet p1
p5
discarded
p2
p3
p4
p5
here the sender is retransmitting all the data packets again

Sender Side:

a. The sender transmits a window of packets to the receiver, starting with sequence number i and ending with sequence number i + N - 1, where N is the window size.

b. The sender sets a timer for each packet in the window.

c. The sender waits for an acknowledgment (ACK) from the receiver.

Receiver Side:

a. The receiver receives the packets and checks for errors.

b. If a packet is received correctly, the receiver sends an ACK back to the sender with the sequence number of the next expected packet.

c. If a packet is received with errors, the receiver discards the packet and sends a negative acknowledgment (NAK) to the sender with the sequence number of the next expected packet.

Sender Side (in case of no ACK received):

1. If the sender does not receive an ACK before the timer for a packet expires, the sender retransmits the entire window of packets starting with the packet whose timer expired.
2. The sender resets the timer for each packet in the window.
3. The sender waits for an ACK from the receiver.

Sender Side (in case of NAK received):

a. If the sender receives a NAK from the receiver, the sender retransmits only the packets that were not correctly received by the receiver.

b. The sender resets the timer for each packet that was retransmitted.

c. The sender waits for an ACK from the receiver.

The above steps are repeated until all packets have been successfully received by the receiver. The Go-Back-N ARQ protocol provides a reliable mechanism for transmitting data over a noisy channel while minimizing the number of retransmissions required.

Selective Repeat ARQ

This protocol also provides for sending multiple frames before receiving the acknowledgement for the first frame. However, here only the erroneous or lost frames are retransmitted, while the good frames are received and buffered.

The Selective Repeat ARQ protocol is a type of error-control protocol used in data communication to ensure reliable delivery of data over a noisy channel. Unlike the Go-Back-N ARQ protocol which retransmits the entire window of packets, the Selective Repeat ARQ protocol retransmits only the packets that were not correctly received.

In the Selective Repeat ARQ protocol, the sender transmits a window of packets to the receiver, and the receiver sends back an acknowledgment (ACK) to the sender indicating successful receipt of the packets. If the receiver detects an error in a packet, it sends a negative acknowledgment (NAK) to the sender requesting retransmission of that packet.

In the Selective Repeat ARQ protocol, the sender maintains a timer for each packet in the window. If the sender does not receive an ACK for a packet before its timer expires, the sender retransmits only that packet.

At the receiver side, if a packet is received correctly, the receiver sends back an ACK with the sequence number of the next expected packet. However, if a packet is received with errors, the receiver discards the packet and sends back a NAK with the sequence number of the packet that needs to be retransmitted.

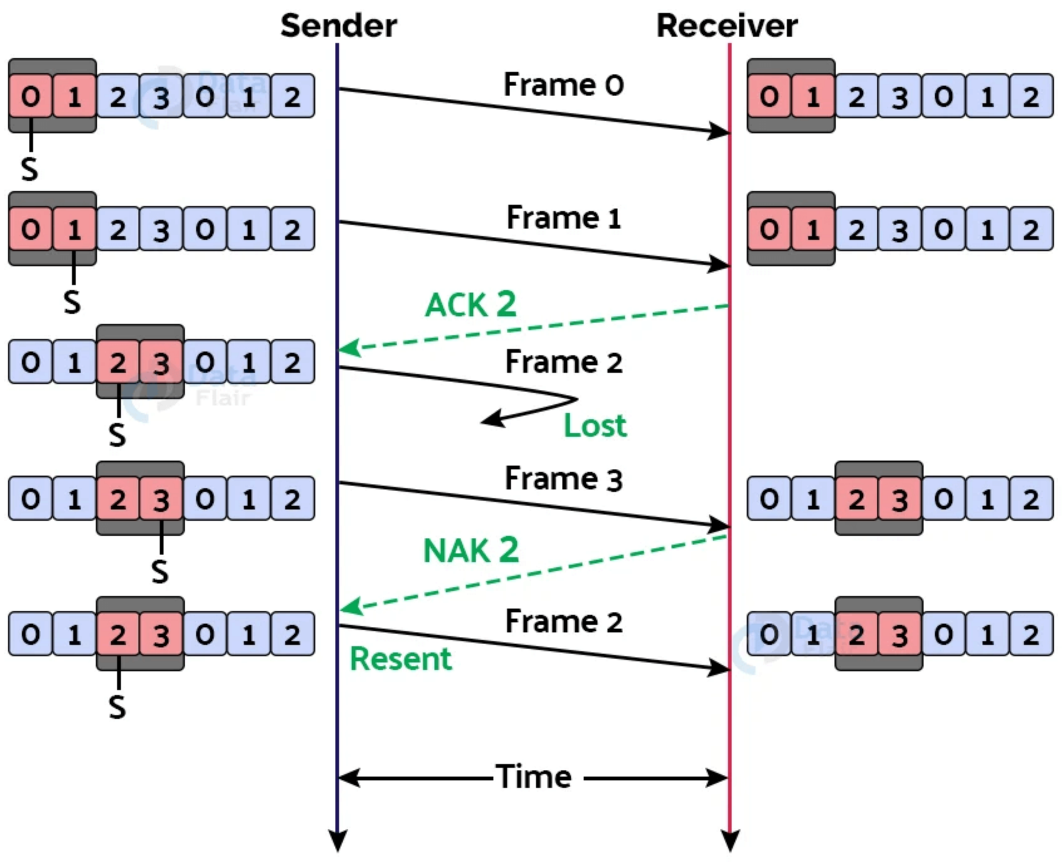
Unlike Go-Back-N ARQ, in Selective Repeat ARQ, the receiver buffer is maintained for all packets that are not in sequence. When a packet with a sequence number different from the expected sequence number arrives at the receiver, it is buffered, and the receiver sends an ACK for the last in-order packet it has received.

If a packet with a sequence number that the receiver has already buffered arrives, it is discarded, and the receiver sends an ACK for the last in-order packet it has received.

In summary, the Selective Repeat ARQ protocol provides a reliable mechanism for transmitting data over a noisy channel while minimizing the number of retransmissions required. It retransmits only the packets that were not correctly received and buffers packets that arrive out of order to reduce the number of retransmissions required.

Flow Diagram

The flow diagram that illustrates the operation of the Selective Repeat ARQ protocol in a noisy channel:



Sender Side:

a. The sender transmits a window of packets to the receiver, starting with sequence number i and ending with sequence number i + N - 1, where N is the window size.

b. The sender sets a timer for each packet in the window.

c. The sender waits for an acknowledgment (ACK) from the receiver.

Receiver Side:

a. The receiver receives the packets and checks for errors.

b. If a packet is received correctly and is in order, the receiver sends an ACK back to the sender with the sequence number of the next expected packet.

c. If a packet is received with errors or is out of order, the receiver discards the packet and sends a negative acknowledgment (NAK) to the sender with the sequence number of the packet that needs to be retransmitted.

d. The receiver buffers out-of-order packets and sends an ACK for the last in-order packet it has received.

Sender Side (in case of no ACK received):

1. If the sender does not receive an ACK before the timer for a packet expires, the sender retransmits only that packet.
2. The sender resets the timer for the retransmitted packet.
3. The sender waits for an ACK from the receiver.

Sender Side (in case of NAK received):

1. If the sender receives a NAK from the receiver, the sender retransmits only the packets that were not correctly received.
2. The sender resets the timer for each packet that was retransmitted.
3. The sender waits for an ACK from the receiver.

The above steps are repeated until all packets have been successfully received by the receiver. The Selective Repeat ARQ protocol provides a reliable mechanism for transmitting data over a noisy channel while minimizing the number of retransmissions required. It retransmits only the packets that were not correctly received, and buffers out-of-order packets to reduce the number of retransmissions required.

CONCLUSION

A. Stop-and-Wait ARQ:

- The simplest of the three protocols.
- The sender transmits one packet at a time and waits for an acknowledgment (ACK) from the receiver before sending the next packet.
- If the ACK is not received within a certain time, the sender retransmits the packet.
- Suitable for channels with low error rates, low data rates, and short transmission distances.

B. Go-Back-N ARQ:

- The sender transmits a window of packets to the receiver.
- If the receiver detects an error in a packet, it sends a negative acknowledgment (NAK) to the sender requesting retransmission of that packet, as well as all subsequent packets in the window.
- The sender retransmits the entire window of packets that were not correctly received.
- Suitable for channels with moderate to high error rates and moderate data rates.

C. Selective Repeat ARQ:

- The sender transmits a window of packets to the receiver.
- If the receiver detects an error in a packet, it sends a NAK to the sender requesting retransmission of that packet only.

- The sender retransmits only the packets that were not correctly received.
- The receiver buffers out-of-order packets to reduce the number of retransmissions required.
- Suitable for channels with moderate to high error rates, high data rates, and long transmission distances.

In summary, Stop-and-Wait ARQ is the simplest and most reliable protocol but not suitable for high data rates. Go-Back-N ARQ is suitable for channels with moderate to high error rates, and Selective Repeat ARQ is suitable for high data rates and long transmission distances. The choice of protocol depends on the characteristics of the communication channel and the requirements of the application.