Unit 3: The Data Link Layer

Functions of Data Link Layer:

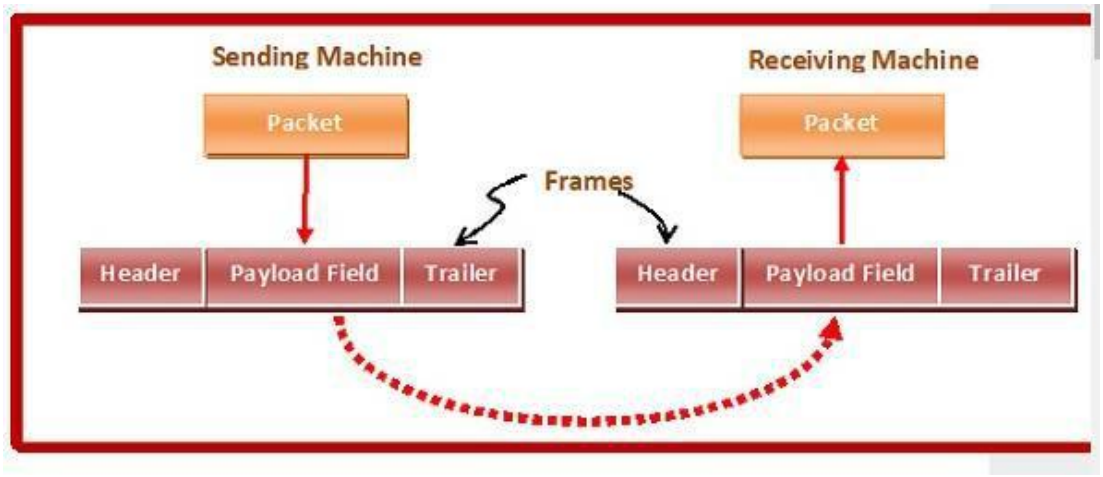
- The data link layer transforms the physical layer, a raw transmission facility, to a link responsible for node-to-node (hop-to-hop) communication.
- Specific responsibilities of the data link layer include framing, addressing, flow control, error control, and media access control.
- The data link layer also adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged, duplicate, or lost frames.

Services Provided to Network Layer:

- Data link layer provides several services to the network layer. The one of the major services provided is the transferring the data from network layer on the source machine to the network layer on destination machine.
- The path is Network layer -> Data link layer -> Physical layer on source machine, then to physical media and thereafter physical layer -> Data link layer -> Network layer on destination machine.

Framing:

- In the physical layer, data transmission involves synchronized 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 make 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.



Parts of a Frame

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 is an 8-bit sequence with bit pattern** Address: It contains the address of the receiver.

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. Consequently, it does not require additional boundary bits to identify the start and end of the frame.

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.

Flow Control:

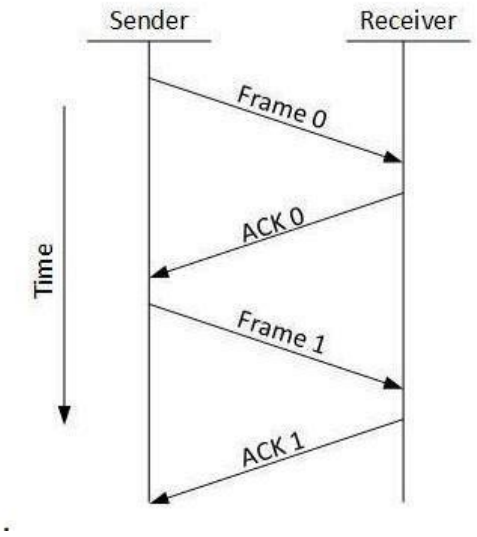
- Data link layer protocols are mainly responsible for flow control. When a data frame is sent from one host to another over a single medium, it is required that the sender and receiver should work at the same speed.
- That is, sender sends at a speed on which the receiver can process and accept the data. If sender is sending too fast, the receiver may be overloaded and data may be lost.
- Flow control is basically technique that gives permission to work and process at different speeds to communicate with one another.
- Flow control is actually set of procedures that explains sender about how much data or frames it can transfer or transmit before data over receiver.

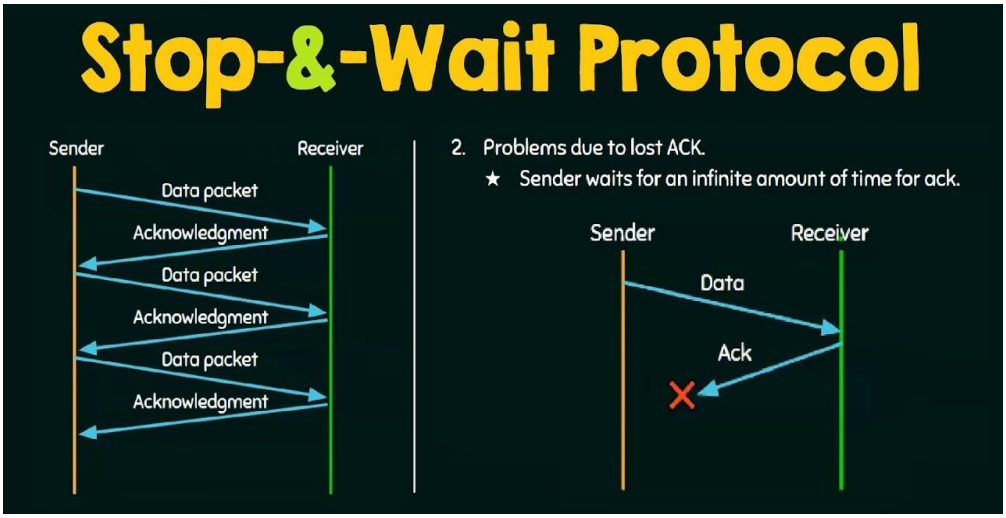
Two types of mechanisms can be deployed to control the flow based on the feedback:

- A simple → stop and wait Protocol
- Sliding Window Protocol

Simplex Stop and Wait

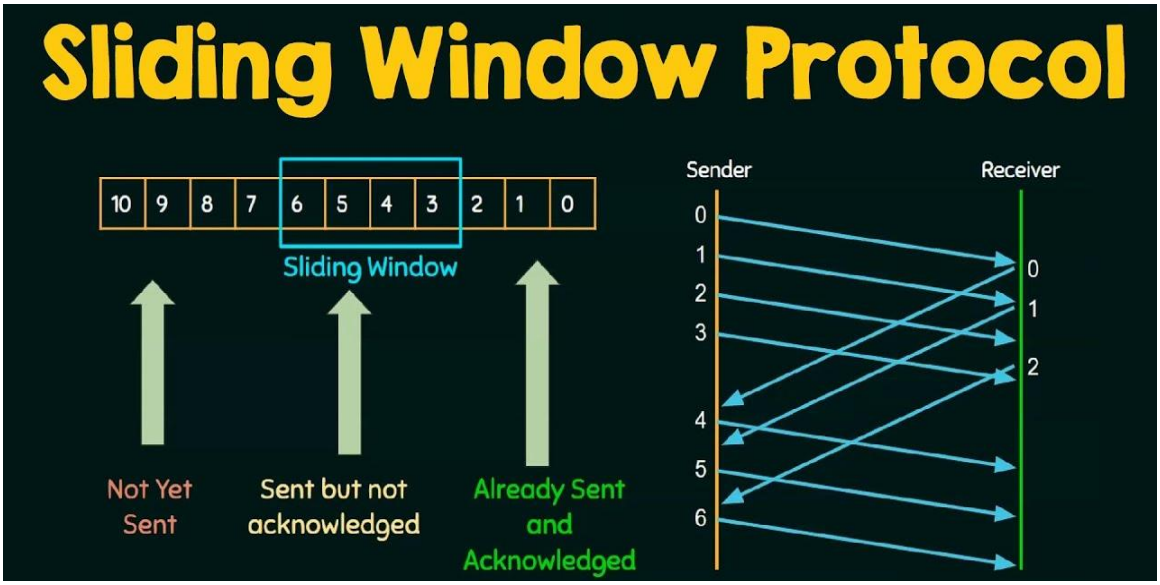
- This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data-frame sent is received.
- The sender sends the next frame only when it has received a positive acknowledgement from the receiver that it is available for further data processing.





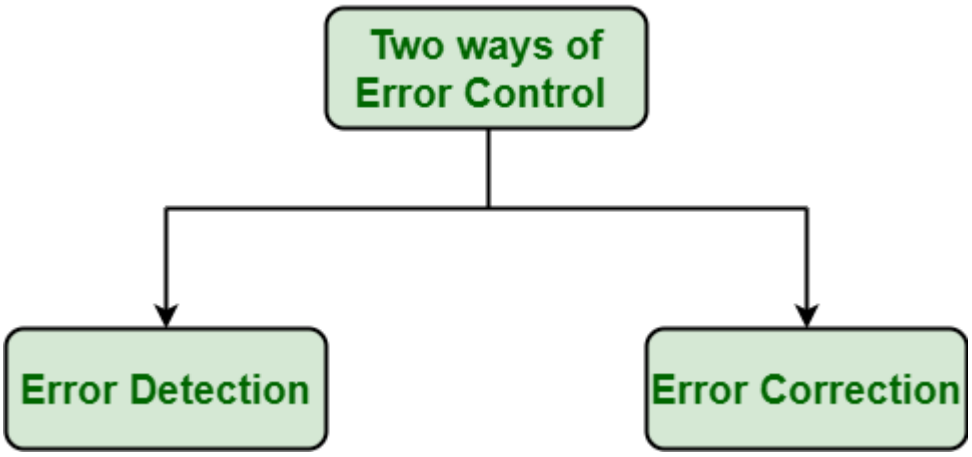
Sliding Window

- 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 have finite sized buffers called windows. The sender and the receiver agree 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.



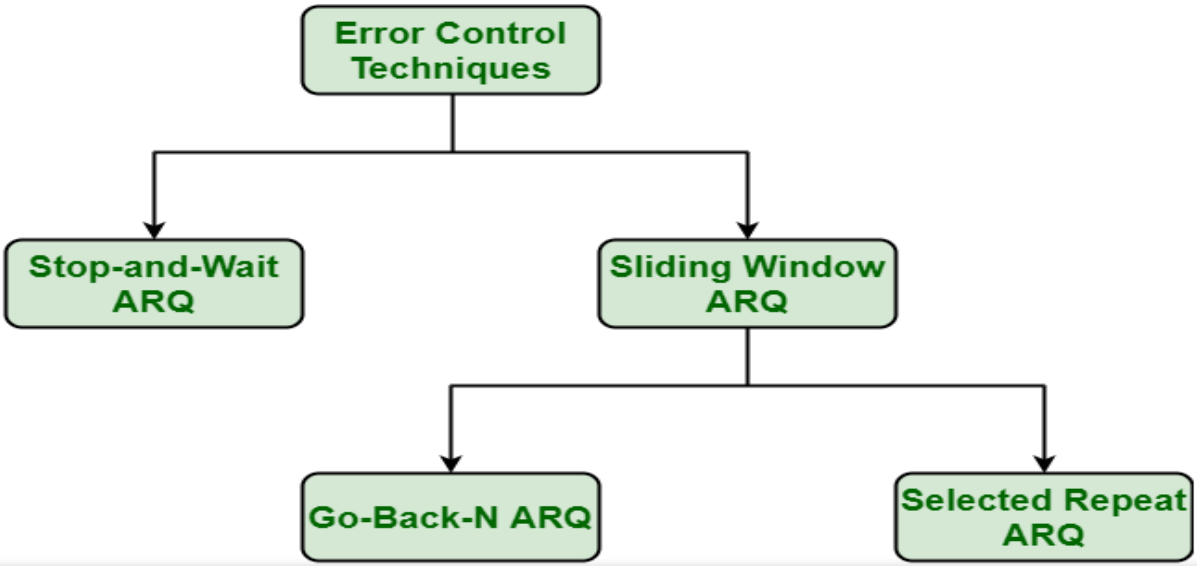
Error Control:

- Error control in data link layer is the process of detecting and correcting data frames that have been corrupted or lost during transmission.
- In case of lost or corrupted frames, the receiver does not receive the correct data-frame and sender is ignorant about the loss.

- 

Error Detection: Error detection, as name suggests, simply means detection or identification of errors.

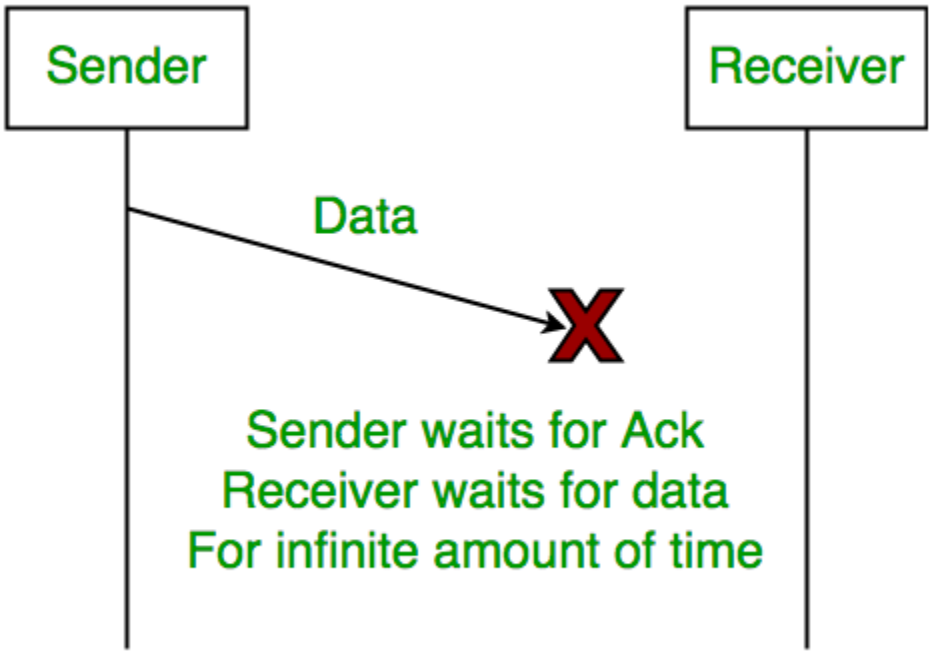
Error Correction: Error correction, as name suggests, simply means correction or solving or fixing of errors. It simply means reconstruction and of original data that is error-free. But error correction method is very costly and is very hard.



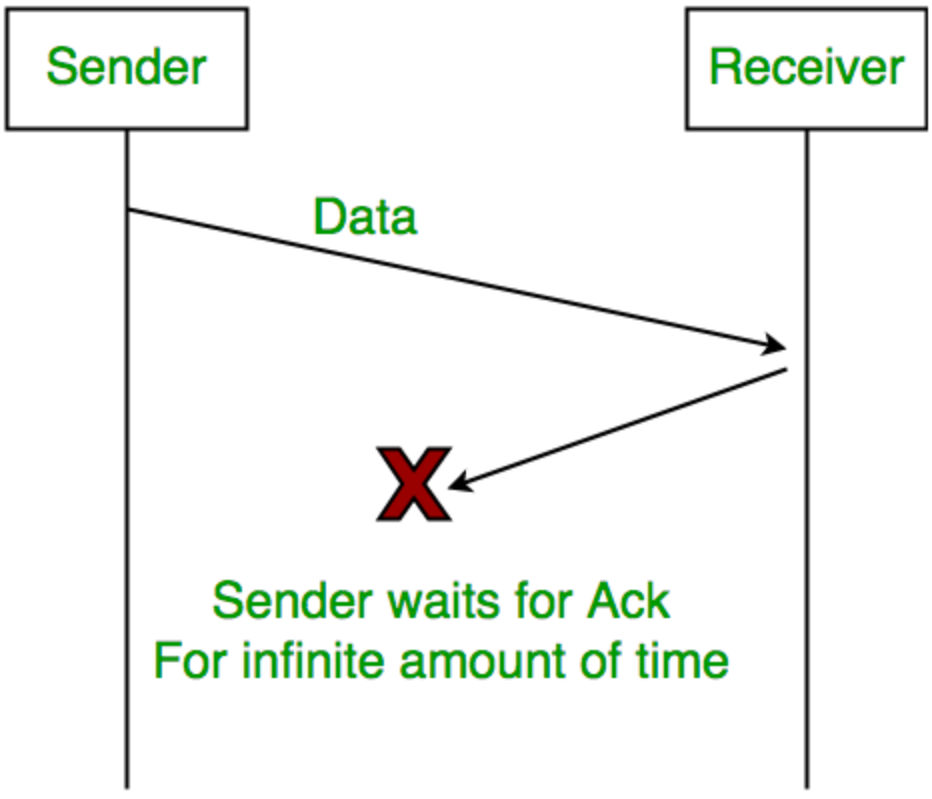
Stop and wait ARQ

Problems :

1. Lost Data



2. Lost Acknowledgement:



3. Delayed Acknowledgement/Data: After a timeout on the sender side, a long-delayed acknowledgement might be wrongly considered as acknowledgement

Stop and Wait for ARQ (Automatic Repeat Request)

The above 3 problems are resolved by Stop and Wait for ARQ (Automatic Repeat Request) that does both error control and flow control.

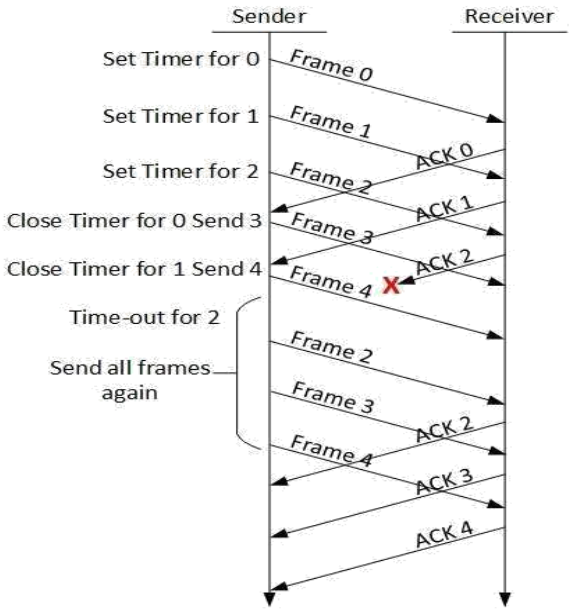
Sliding Window ARQ:

This technique is generally used for continuous transmission error control. It is further categorized into two categories as given below:

Go-Back-N ARQ:

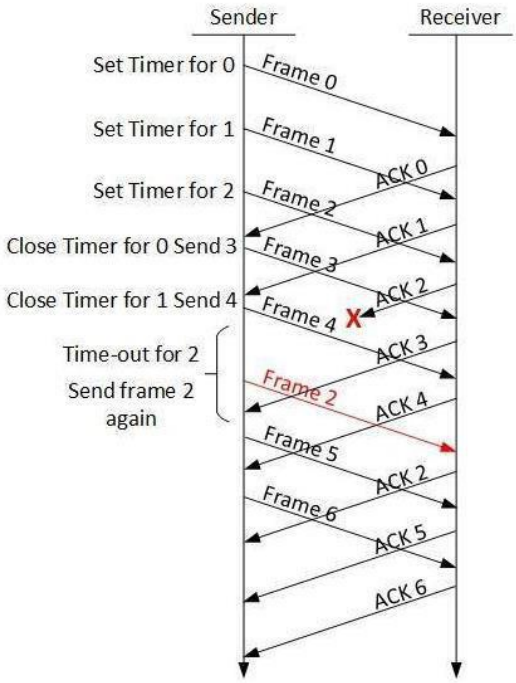
In this protocol, we can send several frames before receiving acknowledgements; we keep a copy of these frames until the acknowledgements arrive. When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N method, both sender and receiver maintain a window.

When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames.



Selective Repeat ARQ:

- In Selective-Repeat, the receiver while keeping track of sequence numbers, buffers the frames in memory and sends NACK for only frame which is missing or damaged.
- The sender in this case, sends only data for which NACK is received.



Error Detection and Correction Techniques:

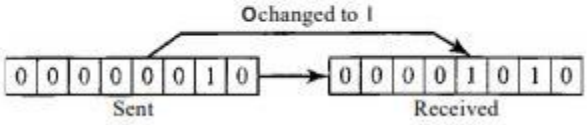
Types of Errors:

- In a single-bit error, a 0 is changed to a 1 or a 1 to a 0.
- In a burst error, multiple bits are changed.

Single-Bit Error:

- The term single-bit error means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

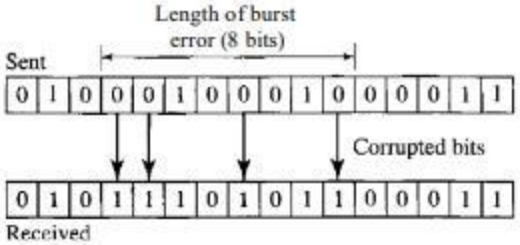
Single-bit error



Burst Error:

- The term burst error means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Burst error of length 8



- A burst error is more likely to occur than a single-bit error.

Detection Versus Correction

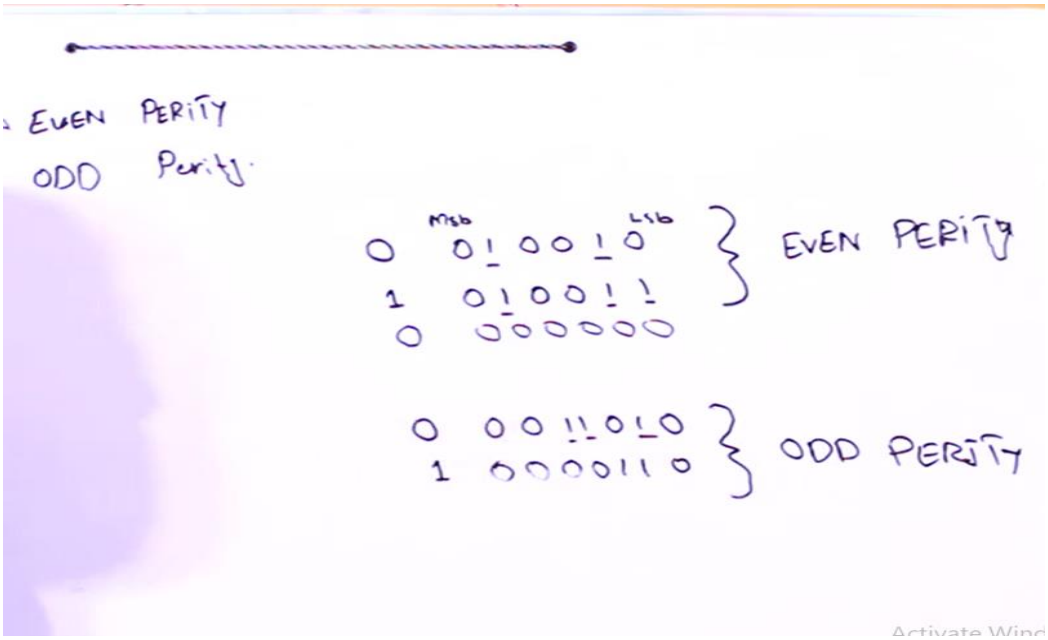
- The correction of errors is more difficult than the detection.
- In error detection, we are looking only to see if any error has occurred.
- The answer is a simple yes or no. We are not even interested in the number of errors.
- In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message.

Some popular techniques for error detection are:

- Parity
- Checksum
- Cyclic redundancy check

Parity check

- The most common and least expensive mechanism for error-detection is the 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:

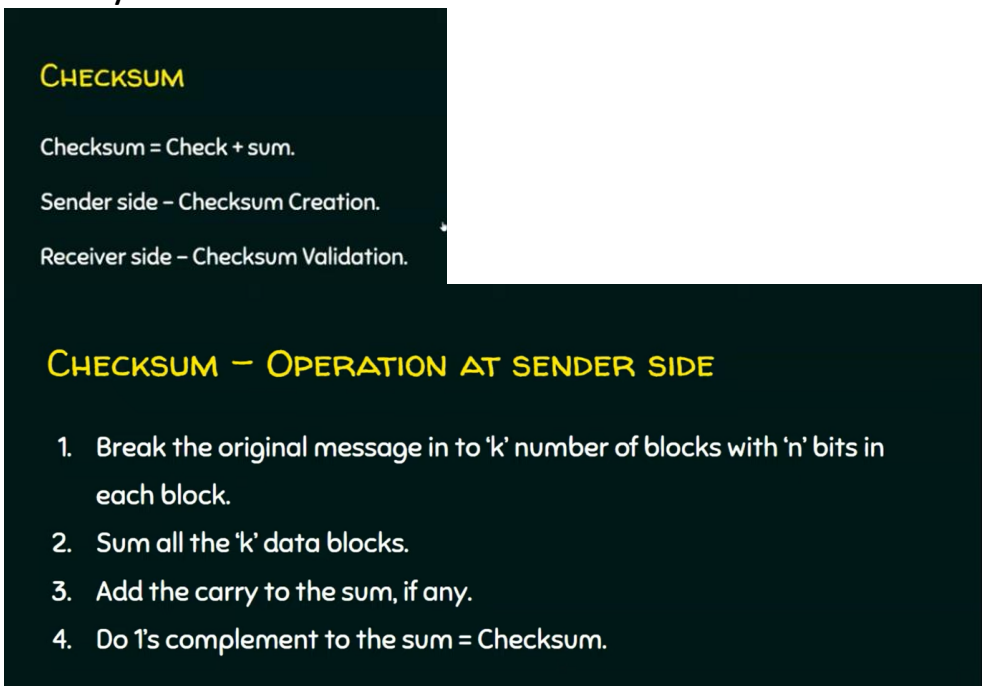


- In case of even parity: If a number of 1s is even then parity bit value is 0. If the number of 1s is odd then parity bit value is 1.
- In case of odd parity: If a number of 1s is odd then parity bit value is 0. If a number of 1s is even then parity bit value is 1.
- On receiving a frame, the receiver 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.
- The parity check is suitable for single bit error detection only.

Checksum

In this error detection scheme, the following procedure is applied:

- Data is divided into fixed sized frames. (k segments each of m bits)
- 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.



Checksum – Example

Consider the data unit to be transmitted is:

1001100111100010001001001000100

10011001	11100010	00100100	10000100

Checksum – Example

10011001		11100010		00100100		10000100		
Carry		1 1 1 1 1						
		1 0 0 0 0		1	0	0		
		0 0 1 0 0		1	0	0		
		1 1 1 0 0		0	1	0		
		1 0 0 1 1		0	0	1		
1	0	0 0 1 0 0		0	1	1		

Checksum – Example

10011001		11100010		00100100		10000100		
Carry		1 1 1 1 1						
		1 0 0 0 0		1	0	0		
		0 0 1 0 0		1	0	0		
		1 1 1 0 0		0	1	0		
		1 0 0 1 1		0	0	1		
		0 0 1 0 0		0	1	1		
						1	0	
		0 0 1 0 0		1	0	1		

Checksum – Example

10011001	11100010	00100100	10000100

Sender

Carry	1	1	1	1	1			
	1	0	0	0	0	1	0	0
	0	0	1	0	0	1	0	0
	1	1	1	0	0	0	1	0
	1	0	0	1	1	0	0	1
	0	0	1	0	0	0	1	1
							1	0
	0	0	1	0	0	1	0	1
CHECKSUM	1	1	0	1	1	0	1	0

Checksum – Operation at receiver side

- ★ Collect all the data blocks including the checksum.
- ★ Sum all the data blocks and checksum
- ★ If the result is all 1's, ACCEPT; Else, REJECT.

Checksum – Example

11011010	10011001	11100010	00100100	10000100

Receiver

Carry	1	1	1	1	1	1		
	1	0	0	0	0	1	0	0
	0	0	1	0	0	1	0	0
	1	1	1	0	0	0	1	0
	1	0	0	1	1	0	0	1
	1	1	0	1	1	0	1	0
1 0	1	1	1	1	1	1	0	1


CHECKSUM – EXAMPLE
11011010
10011001
11100010
00100100
10000100
Carry
1 1 1 1 1 1
1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 0
1 1 1 0 0 0 1 0
1 0 0 1 1 0 0 1
1 1 0 1 1 0 1 0
1 1 1 1 1 1 0 1
1 0
Receiver
Activate Windows
CHECKSUM – EXAMPLE
11011010
10011001
11100010
00100100
10000100
Carry
1 1 1 1 1 1
1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 0
1 1 1 0 0 0 1 0
1 0 0 1 1 0 0 1
1 1 0 1 1 0 1 0
1 1 1 1 1 1 0 1
1 0
Receiver
Activate Windows