Transport Layer protocols: TCP and UDP

By
Ashish Kumar

Transport Layer protocols

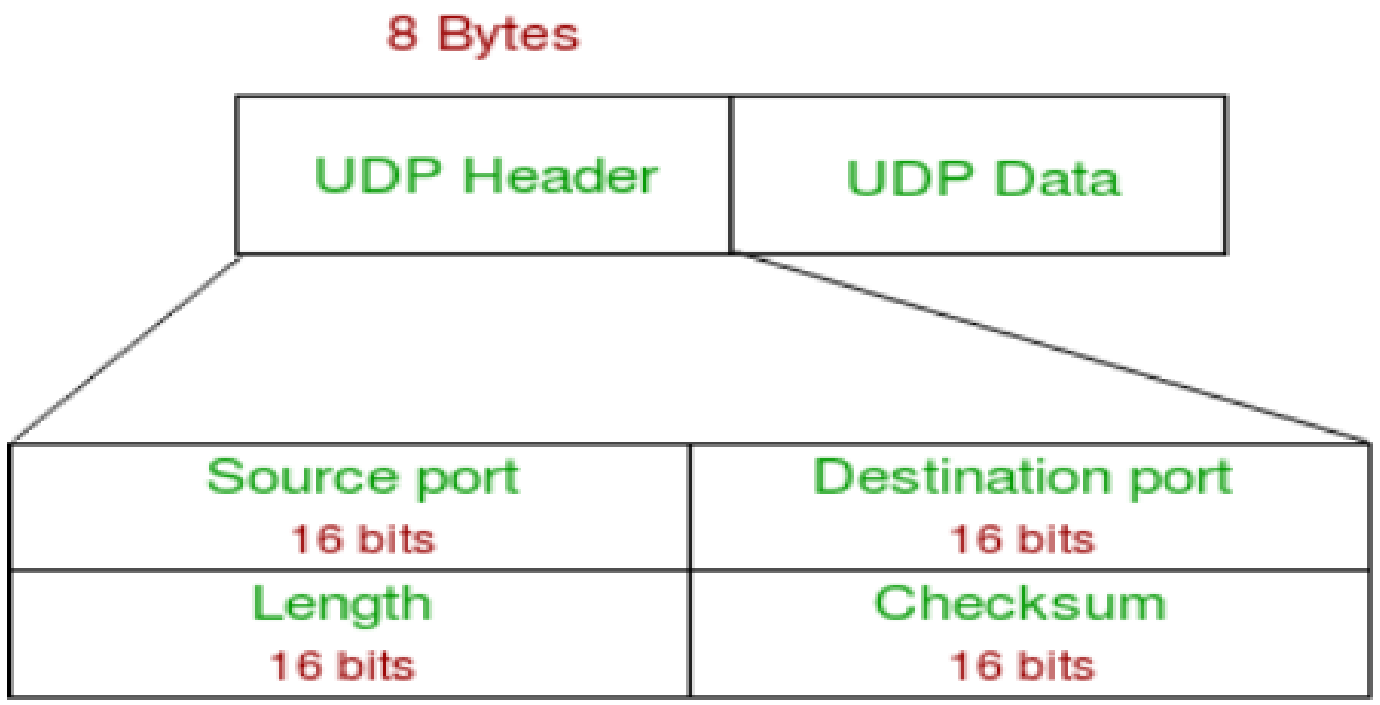
- Transport layer uses two protocols for sending messages from sender to receiver. They are – TCP (Transport Control Protocol) and UDP (User Datagram Protocol).

SL. No.	TCP	UDP
1	TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
2	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
3	An acknowledgment segment is present.	No acknowledgment segment.

Transport Layer protocols

SL. No.	TCP	UDP
4	Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
5	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.
6	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).
7	TCP has a (20-60) bytes variable length header.	UDP has an 8 bytes fixed-length header.
8	Uses handshakes such as SYN, ACK, SYN-ACK	It’s a connectionless protocol i.e. No handshake

UDP Header


8 Bytes
UDP Header
UDP Data
Source port
16 bits
Destination port
16 bits
Length
16 bits
Checksum
16 bits

UDP Header

- **Fields in a UDP header**

1. **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
2. **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
3. **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
4. **Checksum:** Checksum is 2 Bytes long field.

Note - The Checksum calculation is not mandatory in UDP. No Error control or flow control is provided by UDP. Hence UDP depends on IP and ICMP for error reporting.

UDP

- **Application of UDP**

1. Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.

2. It is a suitable protocol for multicasting as UDP supports packet switching.

3. UDP is normally used for real-time applications which can not tolerate uneven delays between sections of a received message.

4. Following implementations uses UDP as a transport layer protocol:
   - NTP (Network Time Protocol)
   - DNS (Domain Name Service)
   - BOOTP, DHCP.
   - NNP (Network News Protocol)
   - Quote of the day protocol
     - TFTP, RTSP, RIP.

TCP

- TCP stands for **Transmission Control Protocol** which indicates that it does something to control the transmission of the data in a reliable way.

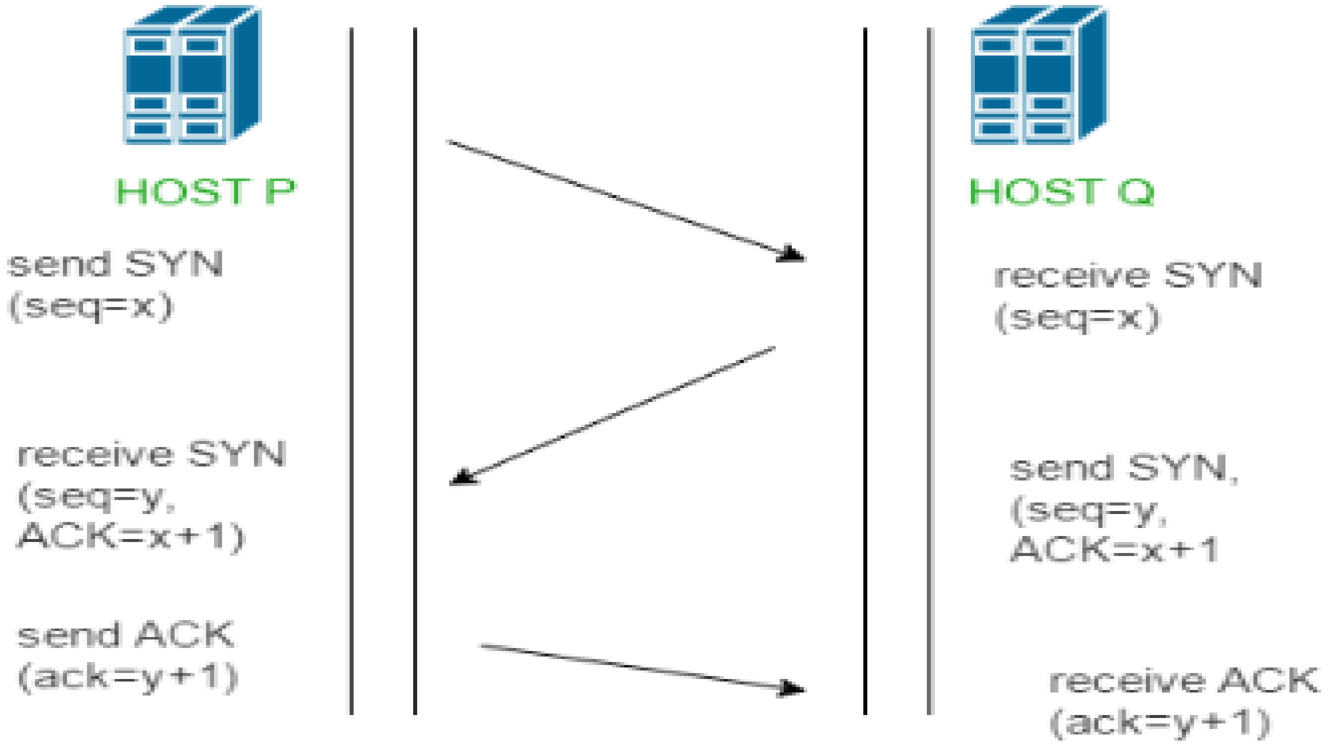
- TCP provides reliable communication with something called **Positive Acknowledgement with Re-transmission(PAR)**.

- The Protocol Data Unit(PDU) of the transport layer is called a segment.

- A device using PAR resend the data unit until it receives an acknowledgement.

- If the data unit received at the receiver's end is damaged (It checks the data with checksum functionality of the transport layer that is used for Error Detection), the receiver discards the segment. So the sender has to resend the data unit for which positive acknowledgement is not received.

TCP



TCP Three-way Handshake

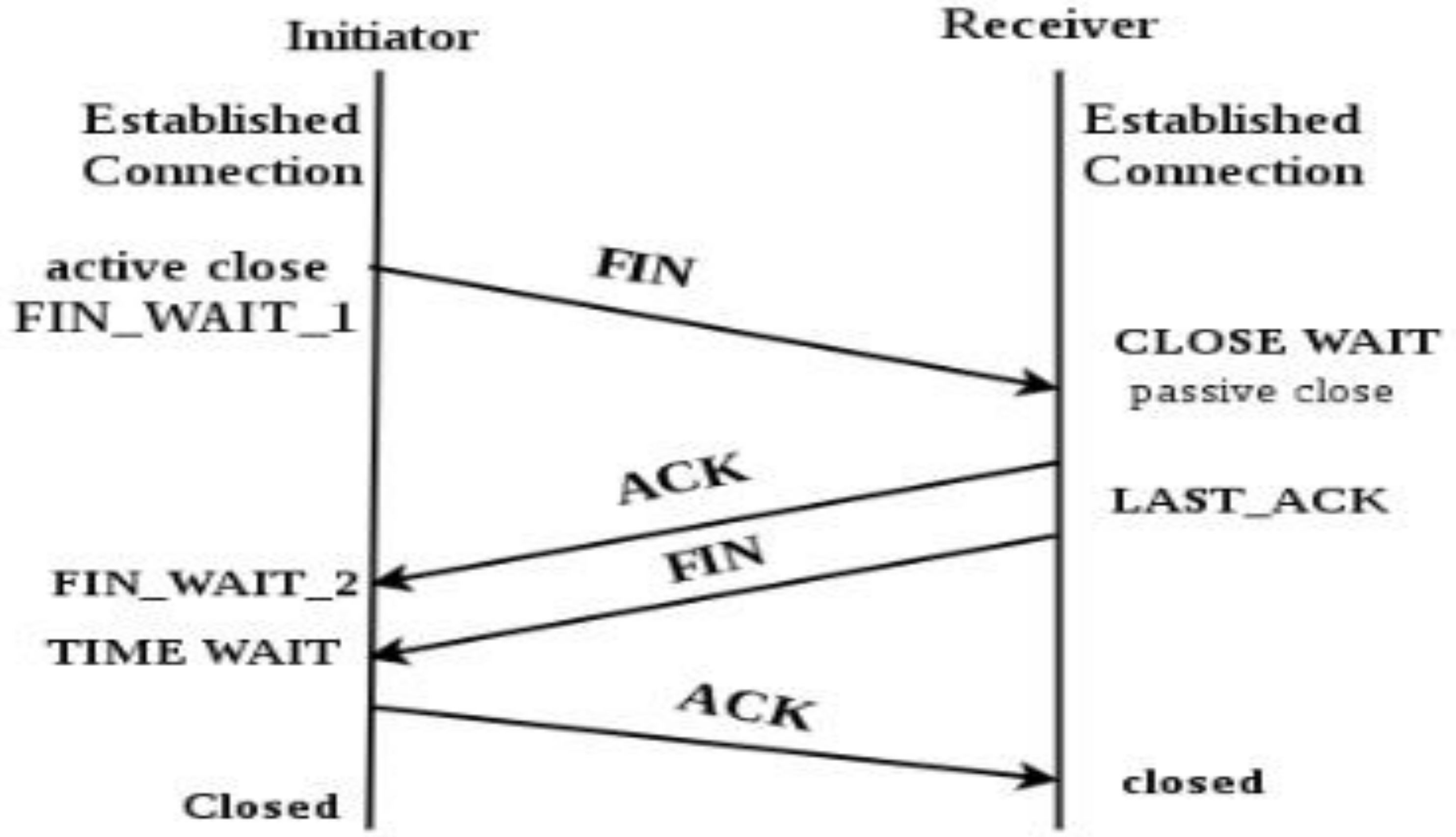
TCP

- **TCP Three-way Handshake**
- **Step 1 (SYN):** In the first step, the client wants to establish a connection with a server, so it sends a segment with SYN(Synchronize Sequence Number) which informs the server that the client is likely to start communication and with what sequence number it starts segments with.

- **Step 2 (SYN + ACK):** Server responds to the client request with SYN-ACK signal bits set. Acknowledgement(ACK) signifies the response of the segment it received and SYN signifies with what sequence number it is likely to start the segments with.

- **Step 3 (ACK):** In the final part client acknowledges the response of the server and they both establish a reliable connection with which they will start the actual data transfer.

TCP

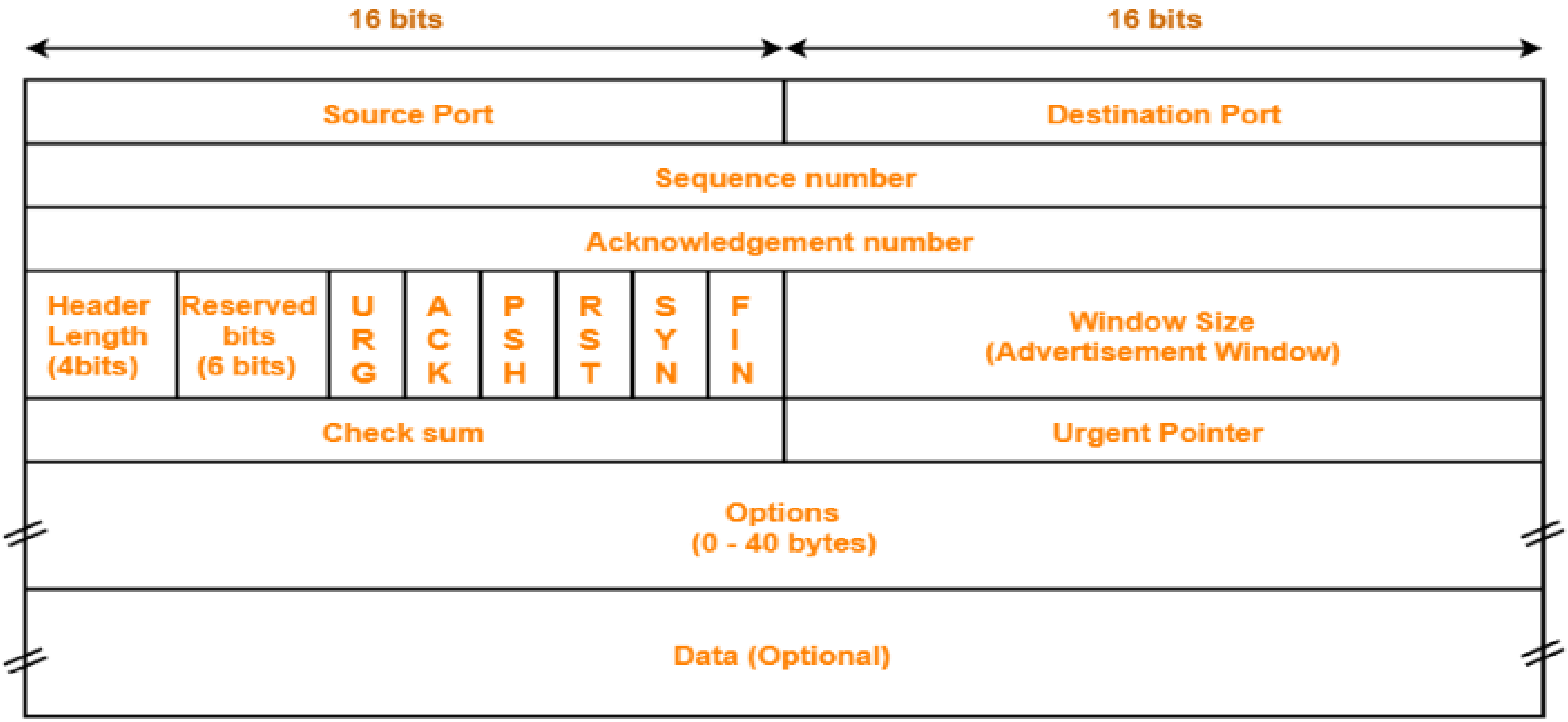


TCP Connection Terminate

TCP

- **TCP Connection Termination**
- **Step 1 (FIN From Client) –** Suppose that the client application decides it wants to close the connection. This causes the client to send a TCP segment with the **FIN** bit set to **1** to the server and to enter the **FIN_WAIT_1** state. While in the **FIN_WAIT_1** state, the client waits for a TCP segment from the server with an acknowledgment (ACK).
- **Step 2 (ACK From Server) –** When the Server received the FIN bit segment from Sender (Client), Server Immediately sends acknowledgement (ACK) segment to the Sender (Client).
- **Step 3 (Client waiting) –** When it receives this segment, the client enters the **FIN_WAIT_2** state. While in the **FIN_WAIT_2** state, the client waits for another segment from the server with the FIN bit set to 1.
- **Step 4 (FIN from Server) –** The server sends the FIN bit segment to the Sender(Client) after some time when the Server sends the ACK segment (because of some closing process in the Server).
- **Step 5 (ACK from Client) –** When the Client receives the FIN bit segment from the Server, the client acknowledges the server's segment and enters the **TIME_WAIT** state. The **TIME_WAIT** state lets the client resend the final acknowledgment in case the **ACK** is lost

TCP Header



16 bits
16 bits
Source Port
Destination Port
Sequence number
Acknowledgement number
Header Length (4bits)
Reserved bits (6 bits)
URG
ACK
PSH
RST
SYN
FIN
Window Size (Advertisement Window)
Check sum
Urgent Pointer
Options (0 - 40 bytes)
Data (Optional)

TCP

1. **Source Port-** Source Port is a 16 bit field. It identifies the port of the sending application.
2. **Destination Port-** Destination Port is a 16 bit field. It identifies the port of the receiving application.
3. **Sequence Number-** Sequence number is a 32 bit field. TCP assigns a unique sequence number to each byte of data contained in the TCP segment. This field contains the sequence number of the first data byte.
4. **Acknowledgement Number-** Acknowledgment number is a 32 bit field. It contains sequence number of the data byte that receiver expects to receive next from the sender. It is always sequence number of the last received data byte incremented by 1.
5. **Header Length-** Header length is a 4 bit field. It contains the length of TCP header. It helps in knowing from where the actual data begins. To represent the header length, we use a scaling factor of 4.

TCP

6. **Reserved Bits-** The 6 bits are reserved. These bits are not used.
7. **URG Bit-** URG bit is used to treat certain data on an urgent basis. When URG bit is set to 1, it indicates the receiver that certain amount of data within the current segment is urgent. The urgent data has be prioritized. Receiver forwards urgent data to the receiving application on a separate channel.
8. **ACK Bit-** ACK bit indicates whether acknowledgement number field is valid or not. When ACK bit is set to 1, it indicates that acknowledgement number contained in the TCP header is valid. For all TCP segments except request segment, ACK bit is set to 1. Request segment is sent for connection establishment during **Three Way Handshake**.
9. **PSH Bit-** PSH bit is used to push the entire buffer immediately to the receiving application. When PSH bit is set to 1, all the segments in the buffer are immediately pushed to the receiving application. No wait is done for filling the entire buffer. This makes the entire buffer to free up immediately.

TCP

10. **RST Bit-** RST bit is used to reset the TCP connection. When RST bit is set to 1, it indicates the receiver to terminate the connection immediately. It causes both the sides to release the connection and all its resources abnormally. The transfer of data ceases in both the directions. It may result in the loss of data that is in transit. This is used only when there are unrecoverable errors and also there is no chance of terminating the TCP connection normally.
11. **SYN Bit-** SYN bit is used to synchronize the sequence numbers. When SYN bit is set to 1, It indicates the receiver that the sequence number contained in the TCP header is the initial sequence number. Request segment sent for connection establishment during 3-way handshake contains SYN bit set to 1.
12. **FIN Bit-** FIN bit is used to terminate the TCP connection. When FIN bit is set to 1, it indicates the receiver that the sender wants to terminate the connection. FIN segment sent for TCP Connection Termination contains FIN bit set to 1.

TCP

13. **Window Size-** Window size is a 16 bit field. It contains the size of the receiving window of the sender. It advertises how much data (in bytes) the sender can receive without acknowledgement. Thus, window size is used for Flow Control.

Note - The window size changes dynamically during data transmission. It usually increases during TCP transmission up to a point where congestion is detected. After congestion is detected, the window size is reduced to avoid having to drop packets.

13. **Checksum-** Checksum is a 16 bit field used for error control. It verifies the integrity of data in the TCP payload. Sender adds CRC checksum to the checksum field before sending the data. Receiver rejects the data that fails the CRC check.

14. **Urgent Pointer-** Urgent pointer is a 16 bit field. It indicates how much data in the current segment counting from the first data byte is urgent. Urgent pointer added to the sequence number indicates the end of urgent data byte. This field is considered valid and evaluated only if the URG bit is set to 1.

Checksum

- Checksum is an error detection method.
- Error detection using checksum method involves the following steps:

1. At sender side

- If m bit checksum is used, the data unit to be transmitted is divided into segments of m bits.
- All the m bit segments are added.
- The result of the sum is then complemented using 1's complement arithmetic.
- The value so obtained is called as **checksum**.

Note: If while adding the m bit segments, the result obtained consists of more than m bits. Then, wrap around the extra bits and add to the result so that checksum value consists of m bits.

2. The data along with the checksum value is transmitted to the receiver.

Checksum

3. At receiver side

- If m bit checksum is being used, the received data unit is divided into segments of m bits.
- All the m bit segments are added along with the checksum value.
- The value so obtained is complemented and the result is checked.
- If the result = 0, then the receiver assumes that no error occurred in the data during the transmission and thus accepts the data.
- If the result is non-zero, then the receiver assumes that error occurred in the data during the transmission. Receiver discards the data and asks the sender for retransmission.

Checksum

Q. Calculate the 8 bit checksum value of the given data 10110001 01010000 10110100 01000001 11101010.

THANK
YOU