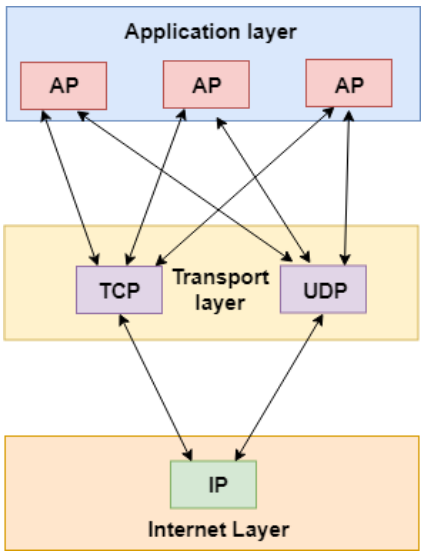
UNIT – 4 : TRANSPORT LAYER

Introduction:

- The transport layer is a 4th layer from the top.
- The main role of the transport layer is to provide the communication services directly to the application processes running on different hosts.
- The transport layer provides a logical communication between application processes running on different hosts. Although the application processes on different hosts are not physically connected, application processes use the logical communication provided by the transport layer to send the messages to each other.
- The transport layer protocols are implemented in the end systems but not in the network routers.
- A computer network provides more than one protocol to the network applications. For example, TCP and UDP are two transport layer protocols that provide a different set of services to the network layer.
- All transport layer protocols provide multiplexing/demultiplexing service. It also provides other services such as reliable data transfer, bandwidth guarantees, and delay guarantees.
- Each of the applications in the application layer has the ability to send a message by using TCP or UDP. The application communicates by using either of these two protocols. Both TCP and UDP will then communicate with the internet protocol in the internet layer. The applications can read and write to the transport layer. Therefore, we can say that communication is a two-way process.



Transport Services:

The services that can be provided by the transport layer are -

1. Process-to-Process Communication
2. Addressing: Port Numbers
3. Encapsulation and Decapsulation
4. Multiplexing and Demultiplexing
5. Flow Control
6. Error Control
7. Congestion Control

1. Process-to-Process Communication:

The Transport Layer is responsible for delivering data to the appropriate application process on the host computers.

This involves multiplexing of data from different application processes, i.e. forming data packets, and adding source and destination port numbers in the header of each Transport Layer data packet.

Together with the source and destination IP address, the port numbers constitutes a network socket, i.e. an identification address of the process-to-process communication.

2. Addressing: Port Numbers:

Ports are the essential ways to address multiple entities in the same location.

Using port addressing it is possible to use more than one network-based application at the same time.

Three types of Port numbers are used:

- **Well-known ports:** These are permanent port numbers. They range between - 0 to 1023.These port numbers are used by Server Process.
- **Registered ports:** The ports ranging from 1024 to 49,151 are not assigned or controlled.
- **Ephemeral ports (Dynamic Ports):** These are temporary port numbers. They range between 49152-65535.These port numbers are used by Client Process.

3. Encapsulation and Decapsulation:

To send a message from one process to another, the transport-layer protocol encapsulates and decapsulates messages.

Encapsulation happens at the sender site. The transport layer receives the data and adds the transport-layer header.

Decapsulation happens at the receiver site. When the message arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer.

4. Multiplexing and Demultiplexing:

The transport layer provides the multiplexing service to improve transmission efficiency in data communication. At the receiver side, demultiplexing is required to collect the data coming from different processes. Transport Layer provides Upward and Downward Multiplexing:

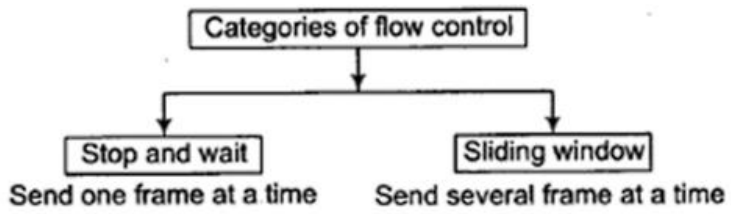
Upward multiplexing means multiple transport layer connections utilize the connection of the same network. Transport layer transmits several transmissions bound for the same destination along the same path in network.

Downward multiplexing means a transport layer connection utilizes the multiple connections. This multiplexing allows the transport layer to split a network connection among several paths to improve the throughput in the network.

5. Flow Control:

Flow Control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver.

It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting node.



6. Error Control:

Error control at the transport layer is responsible for

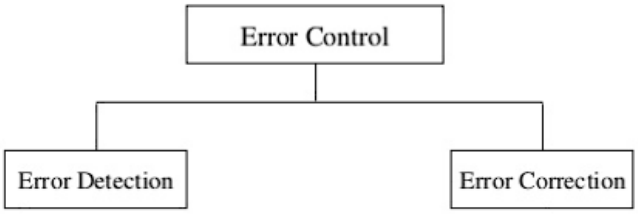
1. Detecting and discarding corrupted packets.

2. Keeping track of lost and discarded packets and resending them.

3. Recognizing duplicate packets and discarding them.

4. Buffering out-of-order packets until the missing packets arrive.

Error Control involves Error Detection and Error Correction



7. Congestion Control:

Congestion in a network may occur if the load on the network (the number of packets sent to the network) is greater than the capacity of the network (the number of packets a network can handle).

Congestion control refers to the mechanisms and techniques that control the congestion and keep the load below the capacity.

Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened

Congestion control mechanisms are divided into two categories,

1. Open loop - prevent the congestion before it happens.

2. Closed loop - remove the congestion after it happens.

Elements of Transport Protocols:

To establish a reliable service between two machines on a network, transport protocols are implemented, which somehow resembles the data link protocols implemented at layer 2. The major difference lies in the fact that the data link layer uses a physical channel between two routers while the transport layer uses a subnet.

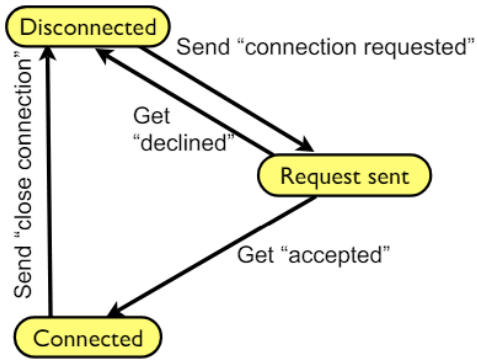
Here are the key elements of transport protocols:

1. **Addressing**: Transport protocols use port numbers to distinguish different communication channels on a device. Port numbers, along with IP addresses, direct incoming data to the appropriate application or service running on the device.
2. **Connection Establishment**: TCP is a connection-oriented protocol, meaning it establishes a connection between the sender and receiver before data exchange begins. This process involves a three-way handshake: SYN, SYN-ACK, and ACK. It ensures both parties are ready for data transmission.
3. **Connection Release**: TCP also manages the orderly release of connections through a process known as connection termination or graceful shutdown. It involves a four-way handshake: FIN, ACK, FIN, ACK. This ensures that both sides agree to close the connection and no data is lost during closure.
4. **Flow Control and Buffering**: TCP implements flow control mechanisms to regulate the rate of data transmission between sender and receiver. It uses sliding window flow control, where the receiver advertises its buffer space to the sender. TCP also utilizes buffering to temporarily store incoming data until it can be processed.
5. **Multiplexing**: TCP supports multiplexing, allowing multiple applications to share a single network connection. Each application is assigned a unique port number, and TCP demultiplexes incoming data packets to the appropriate application based on these port numbers.
6. **Crash Recovery**: TCP includes mechanisms for crash recovery or error recovery. This includes retransmission of lost or corrupted segments, acknowledgment (ACK) mechanisms to confirm successful receipt of data, and sequence numbers to ensure data is delivered in the correct order.

Connection Management:

Modelling Protocol Behaviour:

- Can model protocols using a finite state machine -
  - A set of states with transitions between them
  - The current state indicates what the system is doing at any time
  - Transitions show occurrence of events and the response of the system
- Can be used to define the behaviour of a protocol



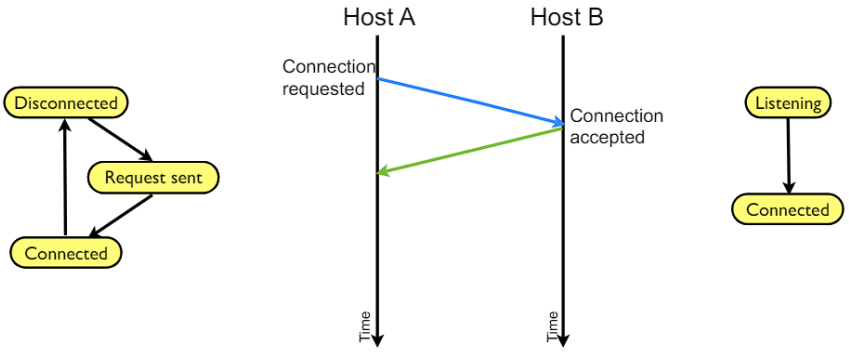
Example: partial state machine for opening and closing a connection

Managing Connections:

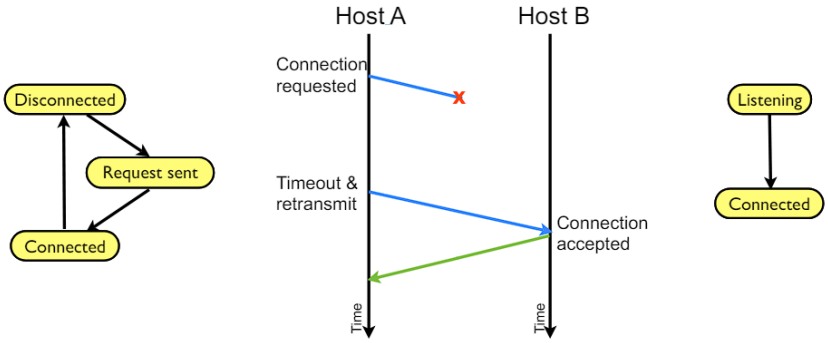
- One role of transport layer: provide reliability
- How to reliably manage transport connections?
  - Setup a reliable connection over an unreliable connectionless network
  - Transport data without loss over an unreliable network
  - Agree to tear down a connection

Connection Establishment:

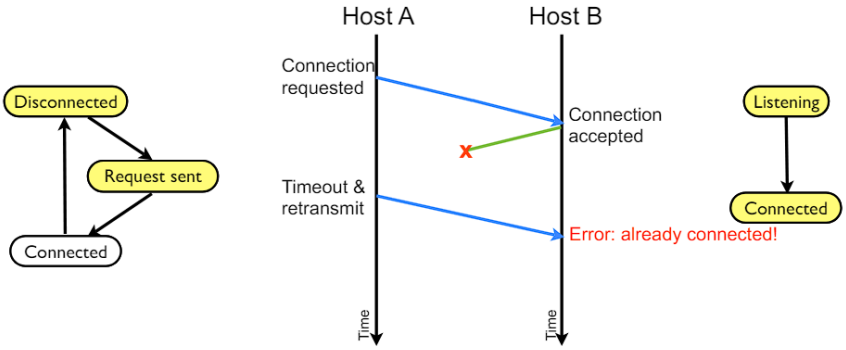
How do two hosts agree to communicate?



What if the initial request is lost?

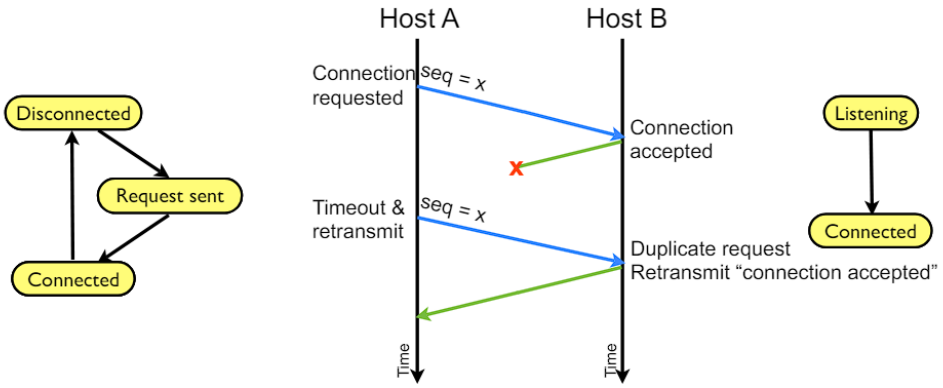


What if the "connection accepted" reply is lost?

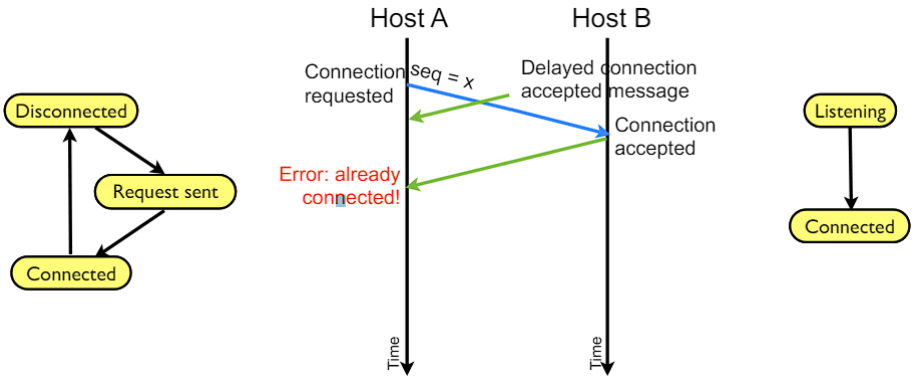


What if the "connection accepted" reply is lost?

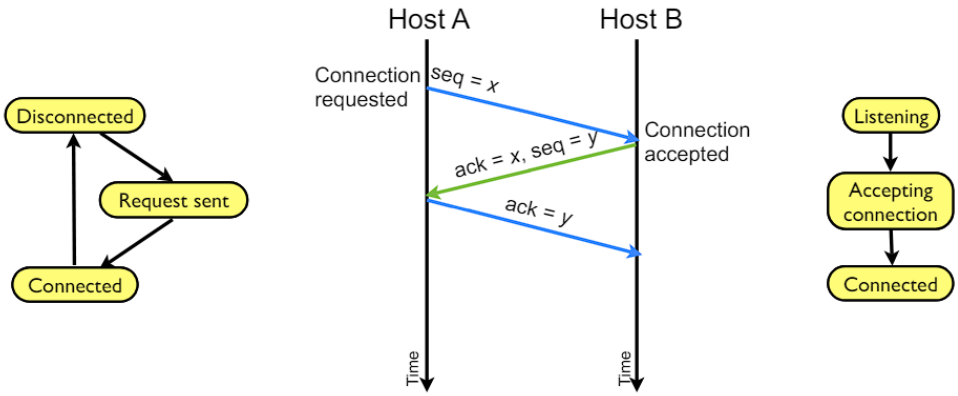
• Sequence number in messages; random initial value



What if data from an old connection is still in the network?



Solution for robust connection establishment: use a three-way handshake



Three Way Handshake:

- Three-way handshake ensures robustness
- Delayed control messages generate an acknowledgement with incorrect sequence number
  - ➢ This is detected, and stops the connection establishment

- Hosts cannot reuse initial sequence number until the maximum packet lifetime passed
  - Requires hosts to keep state regarding previous connections, to avoid reuse
  - Randomly chosen initial sequence number makes collisions unlikely if a host crash causes state to be lost

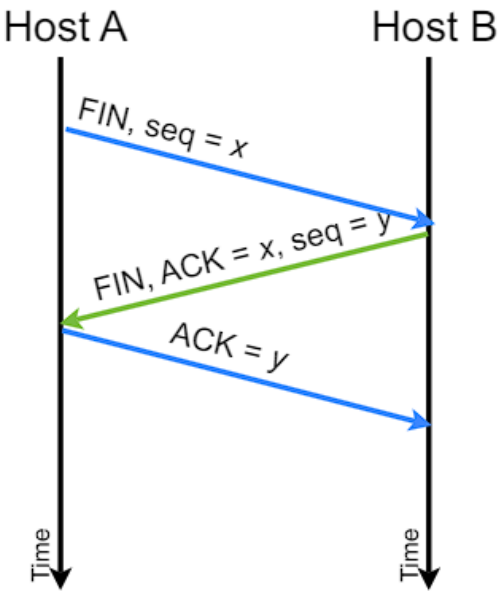
Reliable Data Transfer:

Two approaches to reliable data transfer at the transport layer

- End-to-end ARQ
  - Positive or negative acknowledgements
- End-to-end FEC
  - Within each network layer packet
  - Across several network layer packets
- Conceptually identical to operation at data link layer

Connection Tear Down:

- Three way handshake to tear down a connection
- What happens if the last ACK is lost?
  - A has closed the connection, so cannot resend the ACK; B is still waiting
  - Unavoidable problem → B must eventually give up, without knowing if the last packet arrived
  - Data sent on last packet is potentially lost



Error and Flow Control:

Error control and flow control are essential mechanisms employed by transport layer protocols, such as TCP (Transmission Control Protocol), to ensure reliable and efficient data transfer between communicating hosts. These mechanisms address two critical aspects of data communication: error detection and recovery, and regulating the flow of data to prevent overwhelming the receiver.

1. Error Control:

Error control mechanisms are responsible for detecting and recovering from errors that may occur during data transmission over the network. Transport layer protocols employ various techniques for error control, including:

a. Checksums: A checksum is a small value computed from the data being transmitted. The receiver recalculates the checksum and compares it with the transmitted checksum to detect any errors that may have occurred during transmission.

b. Sequence Numbers: Each data segment is assigned a sequence number by the sender. The receiver uses these sequence numbers to identify missing or out-of-order segments and request retransmissions.

c. Acknowledgments (ACKs): The receiver sends acknowledgments (ACKs) to the sender, confirming the successful receipt of data segments. If the sender does not receive an ACK for a particular segment within a specified time (timeout), it assumes that the segment was lost or corrupted and retransmits it.

d. Retransmissions: If a segment is detected as lost or corrupted, the sender retransmits that segment after a specified timeout period. This mechanism ensures reliable data delivery by recovering from transmission errors.

2. Flow Control:

Flow control mechanisms are used to regulate the rate at which data is transmitted between the communicating hosts. This is necessary to prevent the sender from overwhelming the receiver with data faster than it can process or buffer. Transport layer protocols implement flow control using the following techniques:

a. Receive Window: The receiver advertises a receive window to the sender, indicating the amount of buffer space available for receiving data. The sender can transmit data up to the advertised window size without causing buffer overflows at the receiver.

b. Sliding Window: The sliding window mechanism allows the sender to transmit multiple segments before receiving an acknowledgment. The window size determines the number of unacknowledged segments that can be outstanding at any given time.

c. Credit-based Flow Control: In this approach, the receiver grants credits to the sender, representing the number of segments or bytes the sender is allowed to transmit. The sender must stop transmitting when it runs out of credits and wait for the receiver to grant more credits.

d. Rate-based Flow Control: The sender adjusts its transmission rate based on feedback from the receiver, such as the advertised receive window size or explicit rate control messages.

Error control and flow control mechanisms work together to ensure reliable, ordered, and efficient data transfer between communicating hosts. Error control techniques detect and recover from transmission errors, while flow control mechanisms prevent the sender from overwhelming the receiver with data. These mechanisms are essential for maintaining data integrity and preventing buffer overflows or underflows during data communication over networks.

Congestion Control:

Congestion control is a critical mechanism employed in the transport layer, particularly by the Transmission Control Protocol (TCP), to manage network congestion and prevent the network from becoming overloaded with data. Network congestion occurs when the demand for network resources exceeds the available capacity, leading to packet loss, excessive delays, and reduced overall network performance.

The transport layer implements congestion control mechanisms to regulate the transmission rate of data and adapt to changing network conditions.

The primary objectives of congestion control are:

1. Prevent or mitigate network congestion: By controlling the sending rate, congestion control aims to prevent the network from becoming overwhelmed with more data than it can handle, thus avoiding excessive packet loss and delays.

2. Efficient utilization of network resources: Congestion control mechanisms attempt to maximize the utilization of available network resources while avoiding congestion and ensuring fair sharing among multiple data flows.

3. Maintain end-to-end performance: By adapting to network conditions, congestion control helps maintain acceptable end-to-end performance for applications, such as throughput, delay, and fairness.

TCP employs several congestion control mechanisms, including:

1. Slow Start: Initially, TCP starts with a small congestion window size (cwnd) and gradually increases the transmission rate by increasing the cwnd as acknowledgments (ACKs) are received from the receiver. This allows TCP to probe the network capacity gently.

2. Congestion Avoidance: Once the cwnd reaches a certain threshold, TCP enters the congestion avoidance phase, where it increases the cwnd more conservatively to avoid overwhelming the network.

3. Fast Retransmit and Fast Recovery: If TCP detects packet loss, it assumes network congestion and triggers fast retransmit and fast recovery mechanisms to reduce the cwnd and retransmit lost packets more efficiently.

4. Additive Increase, Multiplicative Decrease (AIMD): TCP employs an AIMD algorithm to adjust the cwnd. It increases the cwnd linearly (additive increase) when the network is not congested and decreases the cwnd exponentially (multiplicative decrease) when congestion is detected.

5. Explicit Congestion Notification (ECN): ECN is an extension to TCP that allows routers to mark packets instead of dropping them when congestion is detected. This explicit notification allows TCP to react to congestion before packet loss occurs, improving performance.

6. Delayed ACK and Nagle's Algorithm: These mechanisms help reduce the number of packets sent by delaying and combining small segments, which can improve efficiency and reduce network load.

Congestion control algorithms continuously monitor the network conditions and adjust the sending rate accordingly. This adaptive behavior helps maintain a balance between maximizing throughput and avoiding network congestion, ultimately improving overall network performance and fairness among competing data flows.

Transmission Control Protocol (TCP):

The transmission control protocol is a transport layer **connection-oriented protocol** that defines the standard of establishing and maintaining the conversation (or connection) that will be used by the applications to exchange the data. The transmission control protocol is one of the most important and widely used protocols of the **IP suite**.

Note:

- There are **two types** of connection namely connection-oriented and connection-less protocol. In the connection-oriented protocol, we first need to connect to the receiver before sending our data.
- The Internet Protocol Suite is the standard network model and stack of communication protocols that are used on the Internet.

One of the prime reasons for using the transmission control protocol over the other **protocol(s) like UDP** is that the TCP ensures the reliable transmission and delivery of our data packets. The transmission control protocol can deal with the various issues that can occur in the data transmission such as packet duplication, packet corruption, packet disordering, packet loss, etc.

The transmission control protocol is used with the internet protocols such as **IPV4, IPV6, ICMP**, etc. Let us now learn the working of the transmission control protocol.

Features of Transmission Control Protocol:

Given below are the features of TCP let us take a look at them:

1. Numbering System:

There are two fields in TCP mainly sequence number and acknowledgment number. These two fields in the TCP mainly refers to Byte Number.

- **Byte Number:** The bytes of data that are being transferred in each connection are numbered by TCP. The numbering mainly starts with a randomly generated number.
    - TCP mainly numbers all the data bytes that are transmitted in a connection.
    - It generates a random number between 0 and 2 raised to the power of 32 -1 for the number of the first byte.
    - Example: If the random no. is 1056 and there are a total of 6000 bytes to be sent then the bytes are numbered from 1056 to 7055.
- **Sequence Number**: After the numbering of bytes, the TCP makes the grouping of bytes in the form of "segments".
    - A sequence is assigned to each segment that is being sent.
    - The Sequence number of each segment is the number of the first byte that is carried in that segment.
    - Thus, the value in the sequence number field of the segment mainly defines the number of the first data byte that is contained in that segment.
- **Acknowledgment Number**: The value of the acknowledgment field in the segment mainly defines the number of the next byte that a party mainly expects to receive.
    - It is cumulative in nature.

2. Flow Control:

The TCP provides the facility of Flow control. With the help of TCP, the receiver of the data control the amount of the data that are to be sent by the sender.

- The flow control is mainly done in order to prevent the receiver from being overwhelmed with the data
- The numbering system also allows the TCP to use byte-oriented flow control.

3. Error Control: As TCP provides reliable services, thus it implements an error control mechanism for this purpose. The Error control though considers the segment as the unit of data for error detection. Error control is byte-oriented in nature.

4. Congestion Control: Another main feature of TCP is that it facilitates Congestion Control in the network. The Amount of the data that the sender sends is not only controlled by the receiver, but congestion in the network also determines it.

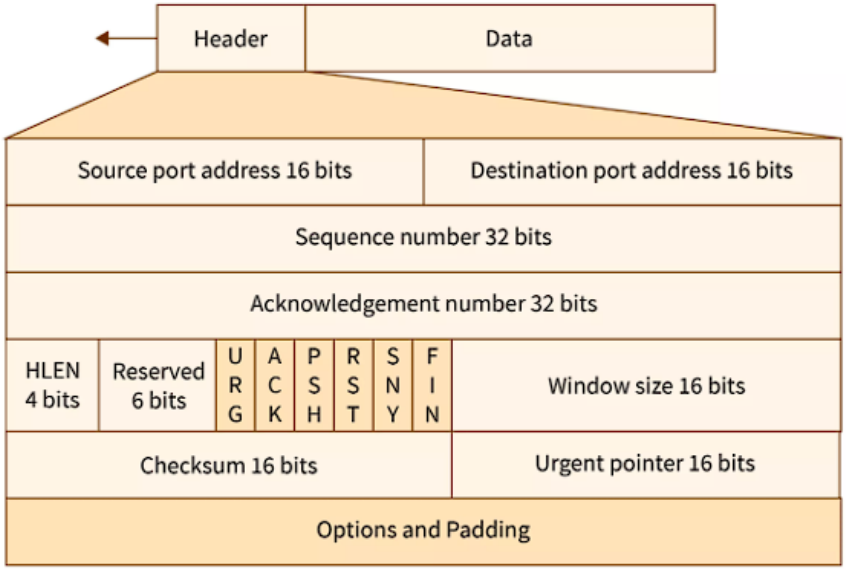
5. Full Duplex: TCP provides another feature and that is Full Duplex which means by using TCP the data can be transmitted in both directions.

6. TCP is a transport layer protocol because it is mainly used to transmit the data from the sender to the receiver.

Segment: The packet in the TCP is mainly known as a segment

TCP Header:

The header of the transmission control protocol is of minimum 20 bytes and a maximum of 60 bytes. Refer to the diagram below to visualize the header and the various header fields of the transmission control protocol.



Let us briefly discuss the various fields of the transmission control protocol:

- **Source Port:** As the name suggests, the source port number defines the port address of the source. The source port of 16 bits.
- **Destination Port:** As the name suggests, the destination port number defines the port address of the destination. The destination port is also of 16 bits.
- **Sequence Number:** It contains the sequence number of a segment or data bytes in a session.
- **Acknowledgment Number:** The ACK flags contain the next sequence number of the segment of data and it works as an acknowledgment for the previously received data.

  Let us take an example to understand the sequence number and acknowledgment number better. Let us suppose the receiver receives the segment number x, then it responds to the acknowledgment number x+1 to the sender.
- **HLEN:** HLEN stands for Header Length. It has 4-bit value. Using 4 bits, it specifies the length of the header. The minimum value for HLEN is 5 and maximum is 15. The length of the header is calculated by multiplying HLEN value by 4. Since, the minimum value of HLEN is 5 and maximum 15, so, minimum header length is 20 bytes and maximum header length is **60 bytes**.
- **Reserved:** It is reserved for any future use and it is of 6-bits.
- **Flags:** The flag is a 1-bit value. Flags are of mainly 6 types and are also known as control bits. Let us discuss the 6 types of flags:
  - **URG**: URG represents an urgent pointer, so, if the URG value is set to 1 then the data is processed urgently.
  - **ACK**: ACK denotes acknowledgment, so, if its bit value is set to 0 then the data packet does not contain an acknowledgment.
  - **PSH**: PSH represents PUSH, so, if the PSH bit is set to 1 then the receiver is requested to push the data without buffering it.
  - **RST**: RST stands for restart, so if its bit value is set to 1 then the connection needs to be restarted.
  - **SYN**: SYN bit as discussed earlier is used to establish a connection.
  - **FIN**: FIN represents finish, so, if its value is set to 1 then the connection is to be closed.
- **Window Size:** The window size is a 16-bit field that denotes the data size that the receiver can accept. As it contains the data size of the receiver, it helps in the flow control mechanism. The window size field is determined by the receiver only.

- **Checksum:** The checksum is also a 16-bit field that contains the checksum of the header, and the data.
- **Urgent Pointer:** If the value of URG is set to 1 then the urgent Pointer field points to the urgent data byte.
- **Options and Padding:** As the name suggests, the option field contains options that are not in the regular header. The options field is described as a 40-bit word. Padding is used in the case when the data in the option field is less than 40-bit. So, padding **(adding extra 0 bits)** helps to make the option bit reach the 40-bit word boundary.

TCP Segment Structure:

As discussed above, the TCP segment or data packet contains two things Header field and Data Field.

The data field contains the actual data that has to be transmitted to the destination. The range of the header is between 20 bytes to 60 bytes and it contains 10 fields denoting having different work. Refer to the above section to learn more about the TCP header segment structure.

TCP Connection Management:

As we know that TCP is a connection-oriented protocol and it means this protocol establishes a virtual path between the source and the destination. All the segments that belong to the message are then sent over this virtual path.

In TCP, the connection-oriented transmission mainly requires three phases and these phases are:

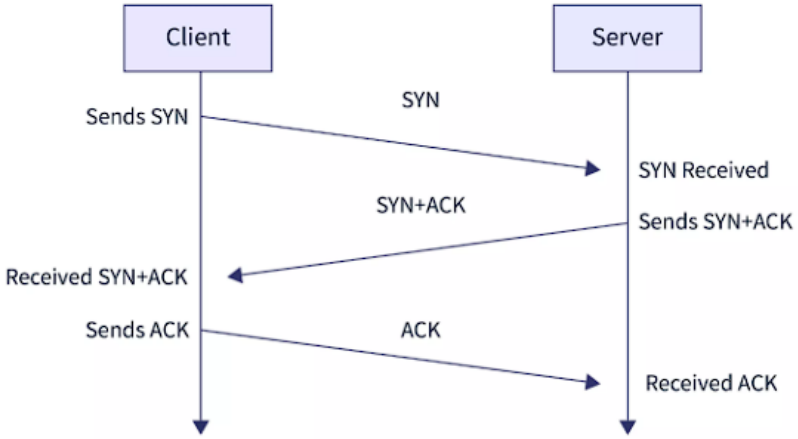
- Connection Establishment Phase
- Data Transfer Phase
- Connection Termination.

1. Connection Establishment Phase:

Transmission of data is done in full-duplex mode. The connection establishment in TCP is mainly termed as **three-way handshaking**.

Whenever two computer systems want to exchange data using the transmission control protocol, they first establish a three-way handshake connection. The three-way handshake connection is used to create a connection between the host or client and the server. As the name suggests, it is a three-step process in which first the client (wants to establish a connection) sends an **SYN** segment (Synchronize Sequence Number segment) which tells the server that the client wants to start the communication. In the second step, the server responds with an **SYN-ACK** signal (SYN Acknowledgement). The SYN-ACK signifies the server has received the client's request to establish the connection. In the third and the last step, the client again sends the ACK signal to the server and they both establish a reliable connection that will be used to transfer the data packets.

The three-way handshake is also known as **SYN-SYN-ACK**.



2. Data Transfer Phase:

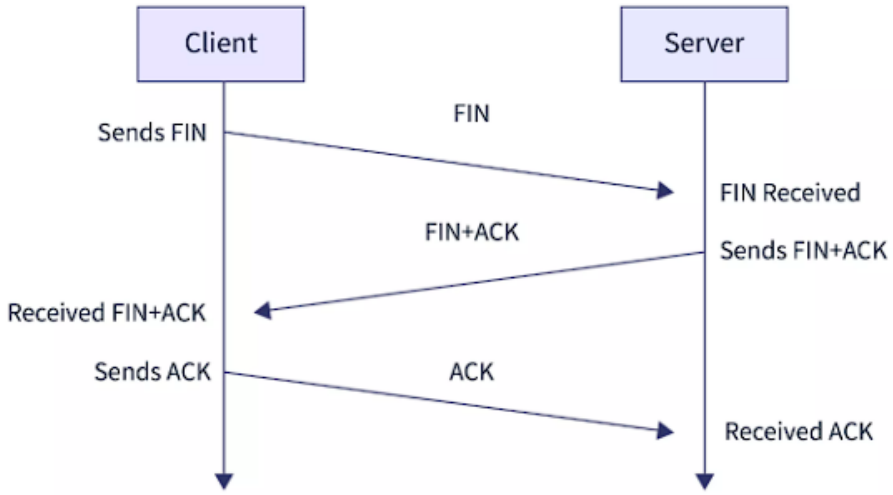
In the second step, the data packets along with the sequence number are sent from the first computer system (client). The second computer (server) responds to these sent packets by sending an acknowledgment or ACK. This acknowledgment bit keeps on increasing with the number of packets sent. This **ACK bit** helps to keep track of three things:

- the successfully received packets,
- the lost packets, and
- the packets that were accidentally sent twice.

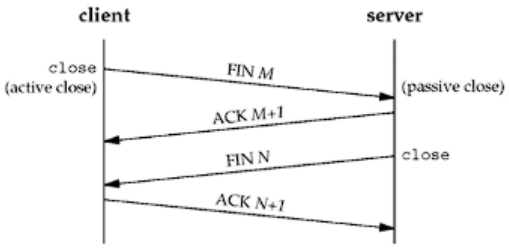
3. Connection Termination Phase:

As we have discussed the client initiates the connection with the server by sending a SYN. But in case of closing the connection, either the server or the client can close the connection. The first computer system (either the server or the client) initiates the closing of the connection by sending a packet with a **FIN** bit or finish bit attached to it.

The other computer sends back or responds with an ACK bit. Finally, the first computer sends an ACK bit back to the second computer, and the connection gets closed.



Half-Close—Client stops sending but receives data.



- Client half-closes the connection by sending a FIN segment.
- Server sends an ACK segment.
- Data transfer from client to the server *stops*.
- After sending all data, server sends FIN segment to client, which is acknowledged by the client.

Note: The TCP breaks the data in the form of packets so that the entire message can reach the target location intact, the TCP at the destination end reassembles the packets into the original message or data.

How Does TCP Ensure Reliable Data Transfer?

The TCP uses **SYN**, **SYN-ACK**, and **ACK** bits to ensure reliable data transfer. The transmission control protocol also offers an error control and flow control mechanism to ensure that the data is transferred at an adequate rate with **less or no loss**.

Let us discuss two major aspects of reliable data transfer i.e.

- how does the TCP handle and detect the lost packets? and
- how does the TCP handle the order of the packets?

Detecting Lost Packets:

If the sender receives three duplicate acknowledgments or the time period of retransmission is expired then the sender knows that the packet has been lost. On the loss of every data packet, the sender treats it as an indication of **network congestion**.

Handling Out-Of-Order Packets:

For handling the order of the packets at the receiver and transmission end, the sequence number and the **ACK number** are used. The sequence number helps to combine the data correctly at the receiver's end.

Advantages of TCP:

- It is a reliable protocol.
- It provides an error-checking mechanism as well as one for recovery.
- It gives flow control.
- It makes sure that the data reaches the proper destination in the exact order that it was sent.
- Open Protocol, not owned by any organization or individual.
- It assigns an IP address to each computer on the network and a domain name to each site thus making each device site to be distinguishable over the network.

Disadvantages of TCP:

- TCP is made for Wide Area Networks; thus, its size can become an issue for small networks with low resources.
- TCP runs several layers so it can slow down the speed of the network.
- It is not generic in nature. Meaning, it cannot represent any protocol stack other than the TCP/IP suite. E.g., it cannot work with a Bluetooth connection.
- No modifications since their development around 30 years ago.

User Datagram Protocol (UDP):

- User Datagram Protocol was developed by David P. Reed in 1980.
- UDP protocol is the connectionless and unreliable protocol.
- Since UDP is a connectionless protocol there is no need to establish a connection before transmitting data.
- User Datagram Protocol gives us a set of rules for transmitting data over the internet.
- UDP packets are called User Datagram.
- User Datagram has 8 bytes fixed-size header.
- UDP protocol will work just like an alternative to TCP (Transmission Control Protocol).
- Process can use UDP protocol if they don't care much about the reliability of transmission and want to send a small message.

Features of UDP protocol:

1. **Transport layer protocol**: UDP is the simplest transport layer communication protocol. It contains a minimum amount of communication mechanisms. It is considered an unreliable protocol, and it is based on best-effort delivery services. UDP provides no acknowledgment mechanism, which means that the receiver does not send the acknowledgment for the received packet, and the sender also does not wait for the acknowledgment for the packet that it has sent.

2. **Connectionless**: The UDP is a connectionless protocol as it does not create a virtual path to transfer the data. It does not use the virtual path, so packets are sent in different paths between the sender and the receiver, which leads to the loss of packets or received out of order.

3. **Ordered delivery of data is not guaranteed.** In the case of UDP, the datagrams are sent in some order will be received in the same order is not guaranteed as the datagrams are not numbered.

4. **Ports:** The UDP protocol uses different port numbers so that the data can be sent to the correct destination. The port numbers are defined between 0 and 1023.

5. **Faster transmission**: UDP enables faster transmission as it is a connectionless protocol, i.e., no virtual path is required to transfer the data. But there is a chance that the individual packet is lost, which affects the transmission quality. On the other hand, if the packet is lost in TCP connection, that packet will be resent, so it guarantees the delivery of the data packets.

6. **Acknowledgment mechanism**: The UDP does have any acknowledgment mechanism, i.e., there is no handshaking between the UDP sender and UDP receiver. If the message is sent in TCP, then the receiver acknowledges that I am ready, then the sender sends the data. In the case of TCP, the handshaking occurs between the sender and the receiver, whereas in UDP, there is no handshaking between the sender and the receiver.

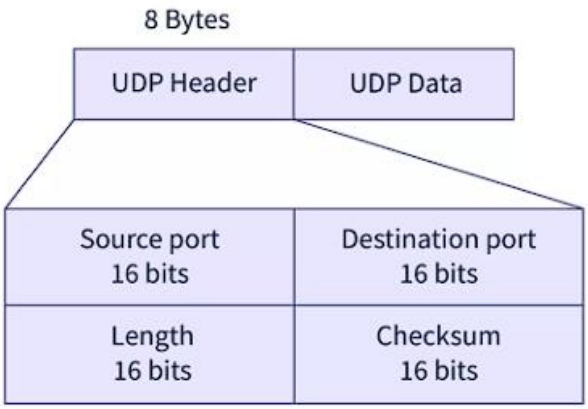
7. **Segments are handled independently.** Each UDP segment is handled individually of others as each segment takes different path to reach the destination. The UDP segments can be lost or delivered out of order to reach the destination as there is no connection setup between the sender and the receiver.

8. **Stateless**: It is a stateless protocol that means that the sender does not get the acknowledgement for the packet which has been sent.

Why do we require the UDP protocol?

As we know that the UDP is an unreliable protocol, but we still require a UDP protocol in some cases. The UDP is deployed where the packets require a large amount of bandwidth along with the actual data. For example, in video streaming, acknowledging thousands of packets is troublesome and wastes a lot of bandwidth. In the case of video streaming, the loss of some packets couldn't create a problem, and it can also be ignored.

UDP Header Format:



In UDP, the header size is 8 bytes, and the packet size is upto 65,535 bytes. But this packet size is not possible as the data needs to be encapsulated in the IP datagram, and an IP packet, the header size can be 20 bytes; therefore, the maximum of UDP would be 65,535 minus 20. The size of the data that the UDP packet can carry would be 65,535 minus 28 as 8 bytes for the header of the UDP packet and 20 bytes for IP header.

The UDP header contains four fields:

- **Source port number:** It is 16-bit information that identifies which port is going t send the packet.
- **Destination port number:** It identifies which port is going to accept the information. It is 16-bit information which is used to identify application-level service on the destination machine.
- **Length:** It is 16-bit field that specifies the entire length of the UDP packet that includes the header also. The minimum value would be 8-byte as the size of the header is 8 bytes.
- **Checksum:** It is a 16-bits field, and it is an optional field. This checksum field checks whether the information is accurate or not as there is the possibility that the information can be corrupted while transmission. It is an optional field, which means that it depends upon the application, whether it wants to write the checksum or not. If it does not want to write the checksum, then all the 16 bits are zero; otherwise, it writes the checksum. In UDP, the checksum field is applied to the entire packet, i.e., header as well as data part whereas, in IP, the checksum field is applied to only the header field.

Working of User Datagram Protocol (UDP):

UDP sends individual packets of data, called **datagrams**, from a sender to a receiver over an IP (Internet Protocol) network.

So, there are some steps to show you the working of the UDP:

1. Firstly, the sender creates a datagram. It contains the data needed to be sent. It also contains the IP address and port number of the receiver.
2. Then, the sender sends the datagram to the network without establishing a connection first.
3. Now, the network forwards the datagram to the receiver. It forwards using the IP address and port number specified in the datagram.
4. Then, the receiver receives the datagram. It extracts the data from it.

Applications of User Datagram Protocol (UDP):

- UDP protocol can be utilized for simple request-response communication when there is a smaller size data since there is very little concern about the error and flow control.

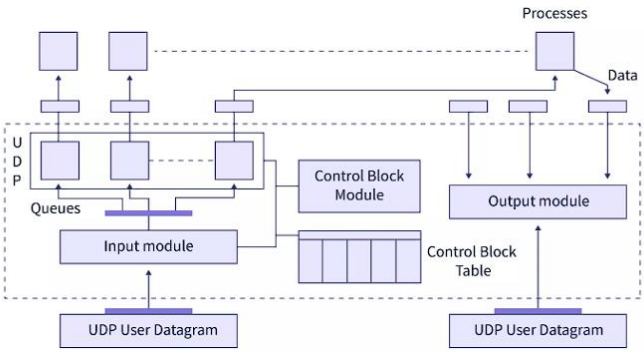
- UDP carries packet switching, so UDP is considered a suitable protocol for multicasting.
- UDP is also used in some routing update protocols such as Routing Information Protocol (RIP).
- UDP protocol is generally used for real-time applications that do not allow uneven delays between received message sections.
- Some of the implementations that use UDP as its transport layer protocol are given below:

    1. NTP (Network Time Protocol)
    2. BOOTP, DHCP
    3. DNS (Domain Name Service)
    4. Quote of the day protocol
    5. NNP (Network News Protocol)
    6. TFTP, RTSP, RIP

Concept of Queuing in UDP:

In User Datagram protocol (UDP), different processes on the network are distinguished by using numbers. We are already aware that UDP provides the process-to-process communication. The processes that require services are generated by the client. On the other hand, the processes that provide services are generated by the server. For both processes, queues are available. The first queue is the incoming queue that is used for receiving the messages, and the second queue is the outgoing queue that is used for sending the messages. When the process is in a running state, then only the queue functions and the queue will get destroyed with the termination of the process.

Refer to the image below to see the concept of queuing in UDP protocol and components used by UDP for sending and receiving packets



UDP uses the following components for handling the sending and receiving of the UDP packets:

- **Input queue**: For each process, UDP packets use a set of queues.
- **Input module**: The input module from the IP takes the user datagram, and then it identifies the information from the control block table of the same port. If it successfully finds any entry in the control block table with the same port as the user datagram, it enqueues the data.
- **Control Block Module**: Control block table is managed by this.
- **Control Block Table**: It contains the entry of open ports.
- **Output module**: Used for creating and sending the user datagram.

Advantages of User Datagram Protocol (UDP):

- UDP produces a minimal number of overheads for data transmission.
- UDP is the simplest transport layer protocol.
- UDP protocol uses packets of smaller size.
- User Datagram provides faster delivery of data as there is no acknowledgment mechanism in UDP.
- For error detection, the UDP protocol uses checksum.

Disadvantages of User Datagram Protocol (UDP):

- UDP is a **unreliable** protocol.
- UDP protocol does not provide **congestion control service**.
- It does not guarantee the order of data received as there is no concept of windowing in UDP.
- Flow control is also not provided by UDP protocol.
- There is **no acknowledgment** mechanism in UDP, so the receiver will not acknowledge the sender for the received packet.

Differences Between TCP and UDP:

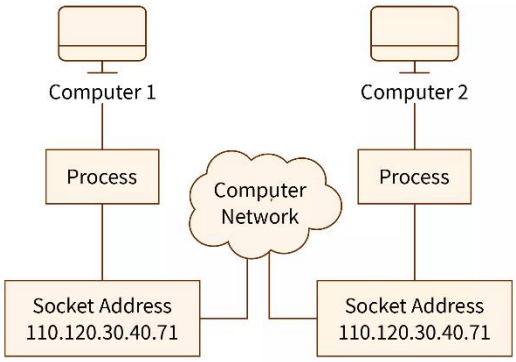
Parameter	TCP	UDP
Type of Service	Connection-oriented.	Connectionless.
Reliability	It is reliable as it guarantees that the data reaches the destination address.	Unreliable.
Error-Checking	TCP uses robust error-checking methods and ensures error-free data is transmitted.	UDP used basic error-checking mechanisms using checksums.
Sequence Control	Sequencing of data is done in TCP, i.e., packets arrive in order at the receiver host.	There is no sequencing in UDP.
Retransmission of lost data	Retransmission of lost or incorrect packets is possible in TCP.	No retransmission of lost packets in UDP.
Speed	Due to error-control and flow-control, there is processing overhead in TCP, and it is slow.	UDP is faster and more efficient than TCP.
Header Size	Varies between 20-60 bytes.	Has header of fixed size(8 bytes).
Broadcasting	Not supported	Supports broadcasting.
Protocols	Used by HTTP, HTTPs, FTP, SMTP, Telnet etc.	Used by DNS, DHCP, TFTP, SNMP, etc.
Overhead	Higher than UDP.	Very low.

Sockets:

Sockets allow communication of two processes that are running on the same or different machines. Sockets are the end of two-way communication between two programs that are running on the networks.

- Sockets are mostly used in client-server architecture for communication between multiple applications.
- Socket programming tells us how we can use socket API for creating communication between local and remote processes.
- The socket is created by the combination of the IP address and port number of the software. With this combination, the process knows the system address and address of the application where data is to be sent.
- **:** is used to separate IP address and port number. For eg: 192.168.1.67:80, 155.2.12.23:77, etc.

Below image to show the socket address example



Function Call	Description
Socket()	To create a socket
Bind()	It's a socket identification like a telephone number to contact
Listen()	Ready to receive a connection
Connect()	Ready to act as a sender
Accept()	Confirmation, it is like accepting to receive a call from a sender
Write()	To send data
Read()	To receive data
Close()	To close a connection

Which Classes are Used for Connection-Less Socket Programming?

Connection-oriented service involves connection establishment before transmitting the data and connection termination after data transmission. Connection-less service does not require any connection establishment and connection termination for transmitting the data over the network.

- For connection-less socket programming, DatagramSocket and DatagramPacket classes are used.
- For connection-oriented socket programming, Socket and ServerSocket classes are used.

DatagramSocket class represents a connectionless socket for transmitting datagram packets.

DatagramPacket is a message transmitted between two communicating parties. DatagramPacket is just like a data container that carries data between two communicating parties. When multiple datagram packets are sent over the network they may arrive in any order irrespective of their sending order.

Socket Programming in TCP:

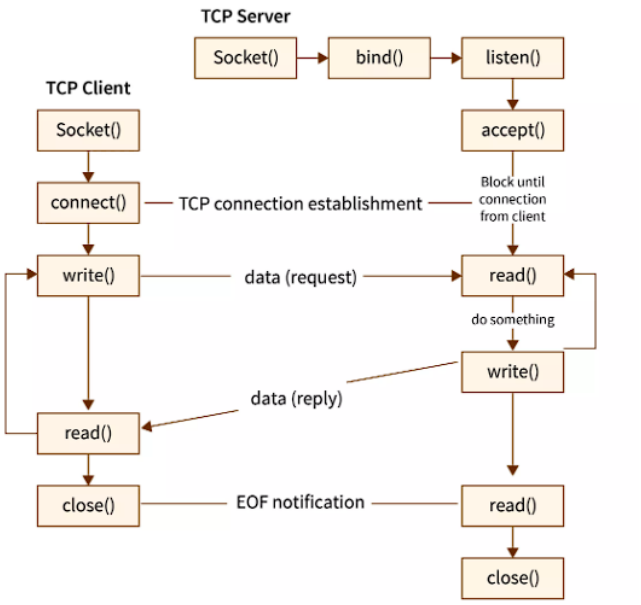
TCP stands for Transmission Control Protocol. TCP is a reliable connection-oriented protocol of the transport layer. TCP establishes the connection before data transmission. Steps for TCP socket programming for establishing TCP socket at the client-side:

- The first step is to create a socket and use the socket() function to create a socket.
- Use the connect() function for connecting the socket to the server address.
- Transmit data between two communicating parties using read() and write() functions.
- After data transmission completion close the connection using close() function.

Following are steps to be followed for establishing a TCP socket on the server-side:

- Use socket() for establishing a socket.
- Use the bind() function for binding the socket to an address.
- Then for listening client connections use listen() function.
- The accept() function is used for accepting the connection of the client.
- Transmit data with the help of the read() and write() function.

Below image to show TCP Socket connection



Socket Programming in UDP:

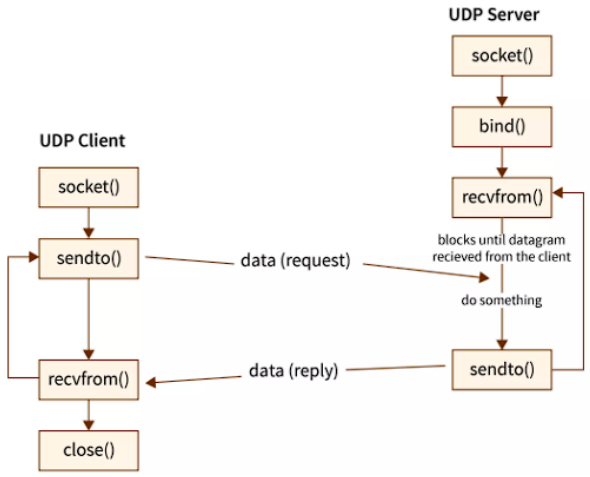
UDP stands for User Datagram Protocol. UDP is a connection-less and unreliable protocol of transport layer. UDP does not establish a connection between two communicating parties before transmitting the data. Following are the steps given that are to be followed for establishing UDP socket connection on the client-side

- Use socket() function for creating socket;
- recvfrom() and sendto() functions are used for transmitting data between two communicating parties.

Steps to be followed for establishing UDP socket connection at the server-side.

- Create a socket using the socket() function.
- Use the bind() function for the binding socket to an address.
- Transmit data with the help of the recvfrom() function and sendto().

Below image to show UDP socket connection



Socket Types:

1. Datagram Sockets: Datagram sockets allow processes to use the User Datagram Protocol (UDP). It is a two-way flow of communication or messages. It can receive messages in a different order from the sending way and also can receive duplicate messages. These sockets are preserved with their boundaries. The socket type of datagram socket is SOCK_DGRAM.

2. Stream Sockets: Stream socket allows processes to use the Transfer Control Protocol (TCP) for communication. A stream socket provides a sequenced, constant or reliable, and two-way (bidirectional) flow of data. After the establishment of connection, data can be read and written to these sockets in a byte stream. The socket type of stream socket is SOCK_STREAM.

3. Raw Sockets: Raw Socket provide user access to the Internet Control Message Protocol (ICMP). Raw sockets are not used for most applications. These sockets are the same as the datagram oriented, their characteristics are dependent on the interfaces. They provided support in developing new communication protocols or for access to more facilities of an existing protocol. Only the superusers can access the Raw Sockets. The socket type of Raw Socket is SOCK_RAW.

4. Sequenced Packet Sockets: Sequenced Packet Sockets are similar to the stream socket, with the exception that record boundaries are preserved in-stream sockets. The given interface in this section is of Network System (NS) that has an abstraction of Sockets and is ordered in all the applications. The Sequenced Packet Sockets enable the user to multiply the sequence packet protocol or some IDP (Internet Datagram Protocol) which heads on the packet or a packet group by writing in the header of the prototype along with the data that has been sent. The socket type of Sequenced Packet Socket is SOCK_SEQPACKET.

Benefits of Socket

- Data storing and sharing
- User access control
- Connections
- Services
- Resource sharing
- Flexible access

Drawbacks of Socket

- Need administrative time
- Failure of server
- Cable braking issue
- Security
- Costly to configure