UNIT-IV

Transport Layer Design Issues

The transport layer delivers the message from one process to another process running on two different hosts. So, it has to perform number of functions to ensure the accurate delivery of message. Thus, the various design issues or functions of transport layer are:

1. Connection Management: Establishing, Maintaining & Releasing Connection
2. Addressing
3. Data Transfer
4. Multiplexing and Demultiplexing
5. Flow Control
6. Error Control
7. Congestion Control

1. **Connection Management: Establishing, Maintaining & Releasing Connection:** The transport layer establishes, maintains & releases end-to-end transport connection on the request of upper layers. Establishing a connection involves allocation of buffers for storing user data, synchronizing the sequence numbers of packets etc. A connection is released at the request of upper layer.

2. **Addressing:** In order to deliver the message from one process to another, an addressing scheme is required. Several process may be running on a system at a time. In order to identify the correct process out of the various running processes, transport layer uses an addressing scheme involving the concept of port numbers. Each process has a specific port number.
   In order to provide communication between two different processes on different networks, both IP address and port number, i.e. socket address is required. Socket address is a combination of IP address and port number.

3. **Data Transfer:** Transport layer breaks user data into smaller units and attaches a transport layer header to each unit forming a TPDU (Transport Layer Data Unit). The TPDU is handed over to the network layer for its delivery to destination. The TPDU header contains port number, sequence number, acknowledgement number, checksum and other fields.

4. **Multiplexing and Demultiplexing:** The addressing mechanism allows multiplexing and demultiplexing by the transport layer, as shown in Figure 2

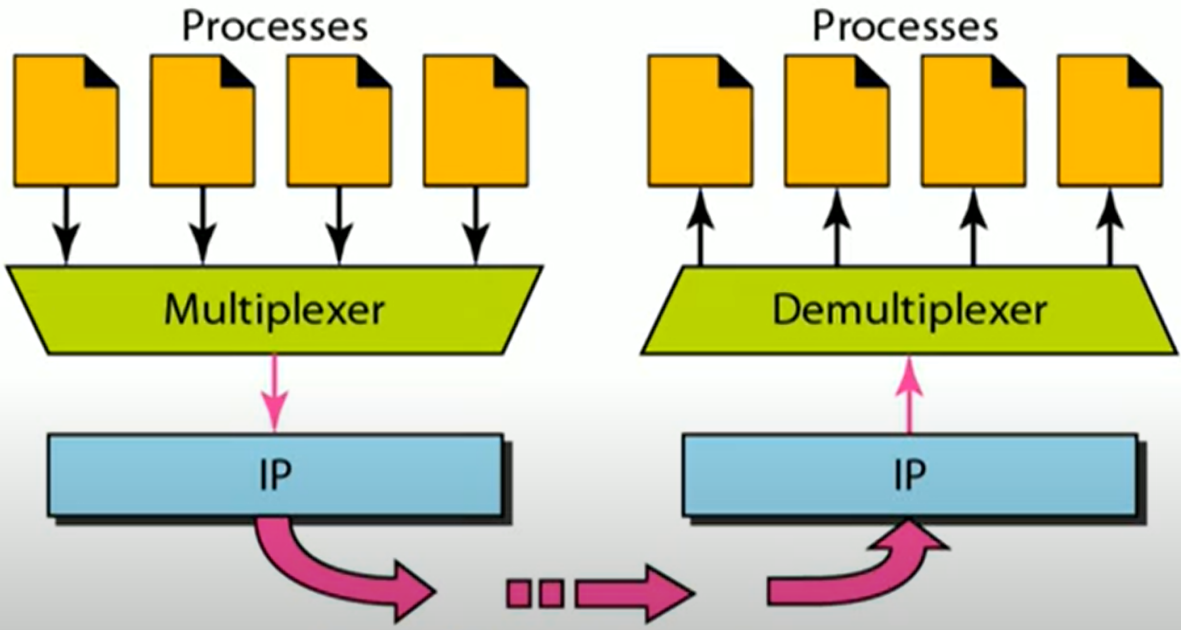


Figure 2: Multiplexing and demultiplexing

Multiplexing

At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing. The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer.

Demultiplexing

At the receiver site, the relationship is one-to-many and requires demultiplexing. The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

5. **Flow Control:** Like data link layer, transport layer also performs flow control. However, flow control at transport layer is performed end-to-end rather than node-to-node. Transport Layer uses a sliding window protocol to perform flow control.

6. **Error Control:** Transport layer also provides end-to-end error control facility. Transport layer deals with several different types of errors:
   - Error due to damaged bits.
   - Error due to non delivery of TPDUs.
   - Error due to duplicate delivery of TPDUs.
   - Error due to delivery of TPDU to a wrong destination.

7. **Congestion Control:** Transport layer also handles congestion in the networks. Several different congestion control algorithms are used to avoid congestion.

Connection Management

This process is completed into two phases:-

- Connection establishment(3-way Handshake Process)
- Connection Termination

TCP 3-Way Handshake Process

TCP is a connection-oriented protocol, which works in terms of full-duplex mode. So, the very first thing is that it requires a well-established logical connection between the devices so that data can be exchanged. Here the noticeable thing is that it can send data to both sides.

So, here are the main 3 steps, also known as TCP 3-way Handshake, which required to fully establish the TCP connection is given below:

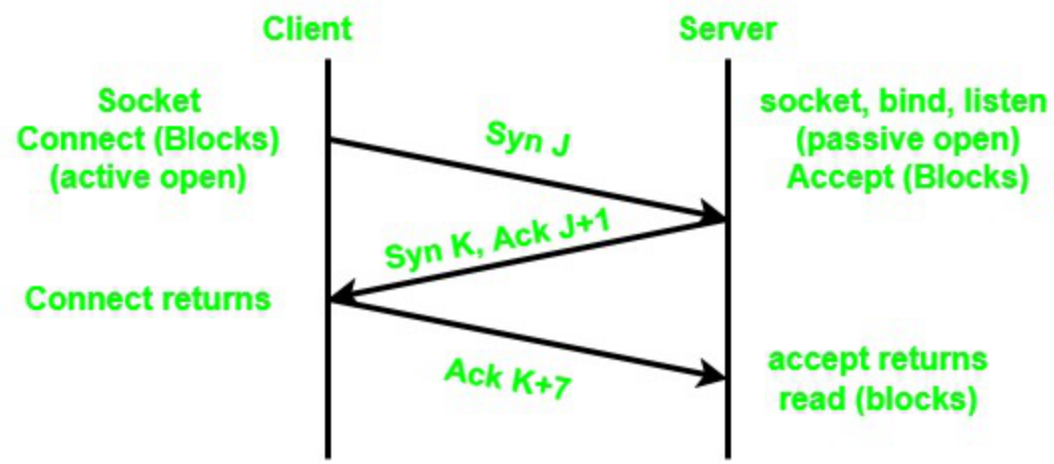
1. In the first step, on the one side, either client or the server initiates the connection establishment by sending **"synchronize"** or an **SYN flag** to the other side. In this step, the client's initial **sequence number** will be sent to the server to create a connection.

2. In return for the SYN flag, the TCP server will send an **"Acknowledgement"** or **ACK flag** along with its initial sequence number for the connection establishment.
3. And finally, with the client's confirmation through the ACK flag, in return to the server's SYN flag, the connection establishment will be activated.

Now, to understand the process clearly, here we will be discussing the further steps to establish the TCP connection down below:

In the first step, the client sends an SYN request to establish a connection, as the Packet SYN: 1 to the server. After successfully receiving the SYN flag or the packet, now the server will send an ACK flag as the packet ACK: 1 for the confirmation purpose to the client. In the process, the server can also send the data to the other side. For this, the server will be sending the data as the packet SYN: 1 along to the client. Finally, after receiving the data successfully, the client will then send an ACK flag as packet ACK: 1 to the server as a confirmation.

To understand it you can also take a look at the given diagram below:

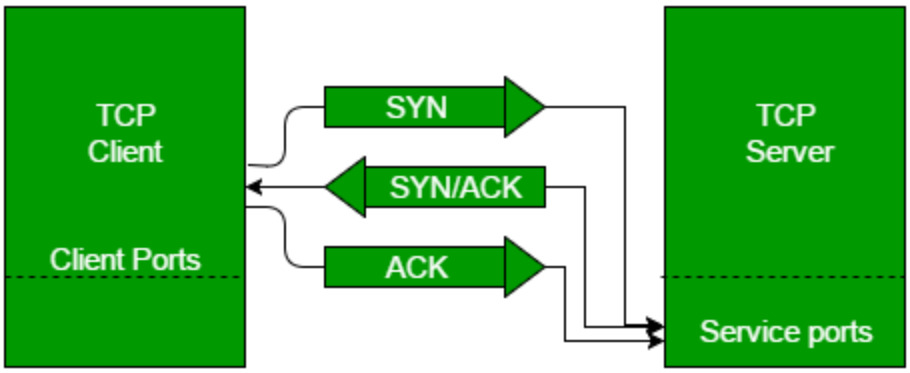


The Three-Way Connection Establishment

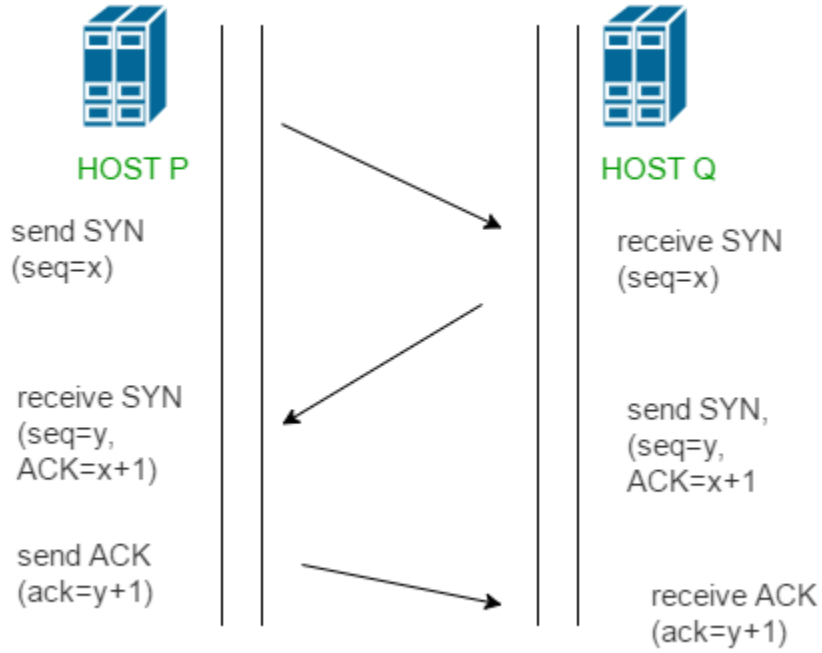
This could also be seen as a way of how TCP connection is established. Before getting into the details, let us look at some basics. TCP stands for **Transmission Control Protocol** which indicates that it does something to control the transmission of the data in a reliable way.

The process of communication between devices over the internet happens according to the current **TCP/IP** suite model(stripped out version of OSI reference model). The Application layer is a top pile of a stack of TCP/IP models from where network referenced applications like web browsers on the client-side establish a connection with the server. From the application layer, the information is transferred to the transport layer where our topic comes into the picture. The two important protocols of this layer are – TCP, **UDP(User Datagram Protocol)** out of which TCP is prevalent(since it provides reliability for the connection established). However, you can find an application of UDP in querying the DNS

server to get the binary equivalent of the Domain Name used for the website.



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. Now 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. You can realize from the above mechanism that three segments are exchanged between sender(client) and receiver(server) for a reliable TCP connection to get established. Let us delve into how this mechanism works :



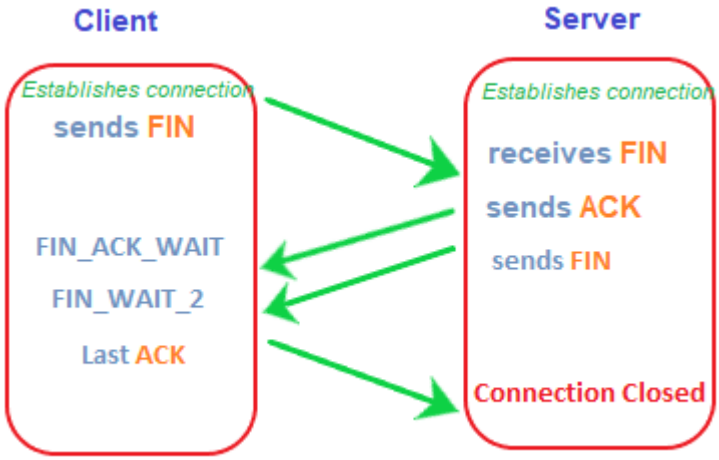
- **Step 1 (SYN):** In the first step, the client wants to establish a connection with a server, so it sends a segment with SYN(Synchronize Sequence Number) which informs the server that the client is likely to start communication and with what sequence number it starts segments with
- **Step 2 (SYN + 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

Connection Termination

Any device establishes a connection before proceeding with the termination. TCP requires 3-way handshake to establish a connection between the client and server before sending the data. Similarly, to terminate or stop the data transmission, it requires a 4-way handshake. The segments required for TCP termination are similar to the segments to build a TCP connection (ACK and SYN) except the FIN segment. The FIN segment specifies a termination request sent by one device to the other.

The client is the data transmitter and the server is a receiver in a data transmission process between the sender and receiver. Consider the below TCP termination diagram that shows the exchange of segments between the client and server.

The diagram of a successful TCP termination showing the four handshakes is shown below:



Let's discuss the TCP termination process with the help of six steps that includes the sent requests and the waiting states. The steps are as follows:

Step 1: FIN

FIN refers to the **termination request** sent by the client to the server. The first FIN termination request is sent by the client to the server. It depicts the start of the termination process between the client and server.

Step 2: FIN_ACK_WAIT

The client waits for the ACK of the FIN termination request from the server. It is a **waiting state** for the client.

Step 3: ACK

The server sends the ACK (Acknowledgement) segment when it receives the FIN termination request. It depicts that the server is ready to close and terminate the connection.

Step 4: FIN _WAIT_2

The client waits for the FIN segment from the server. It is a type of approved signal sent by the server that shows that the server is ready to terminate the connection.

Step 5: FIN

The FIN segment is now sent by the server to the client. It is a confirmation signal that the server sends to the client. It depicts the successful approval for the

termination.

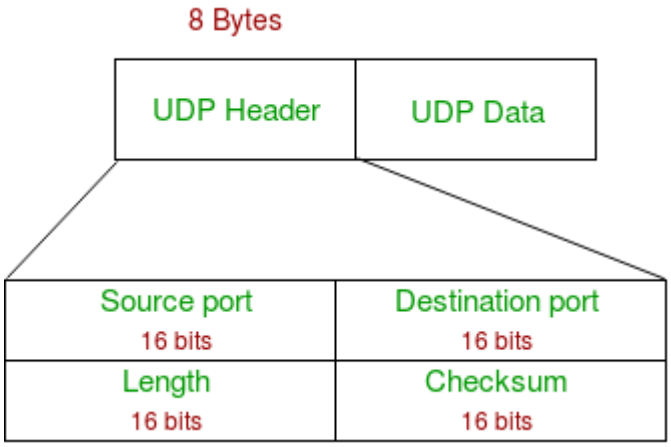
Step 6: ACK

The client now sends the ACK (Acknowledgement) segment to the server that it has received the FIN signal, which is a signal from the server to terminate the connection. As soon as the server receives the ACK segment, it terminates the connection.

User Datagram Protocol (UDP)

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an **unreliable and connectionless protocol.** So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network. The UDP enables process to process communication.
Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth.
User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

UDP Header –
UDP header is an **8-bytes** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consist of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.



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. It is the 16-bit one's complement of the one's complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Notes – Unlike TCP, 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. Also UDP provides port numbers so that is can differentiate between users requests.

Applications of UDP:

- Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.
- It is a suitable protocol for multicasting as UDP supports packet switching.
- UDP is used for some routing update protocols like RIP(Routing Information Protocol).
- Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.
- UDP is widely used in online gaming, where low latency and high-speed communication is essential for a good gaming experience. Game servers often send small, frequent packets of data to clients, and UDP is well suited for this type of communication as it is fast and lightweight.
- Streaming media applications, such as IPTV, online radio, and video conferencing, use UDP to transmit real-time audio and video data. The loss of some packets can be tolerated in these applications, as the data is continuously flowing and does not require retransmission.
- VoIP (Voice over Internet Protocol) services, such as Skype and WhatsApp, use UDP for real-time voice communication. The delay in voice communication can be noticeable if packets are delayed due to congestion control, so UDP is used to ensure fast and efficient data transmission.
- DNS (Domain Name System) also uses UDP for its query/response messages. DNS queries are typically small and require a quick response time, making UDP a suitable protocol for this application.
- DHCP (Dynamic Host Configuration Protocol) uses UDP to dynamically assign IP addresses to devices on a network. DHCP messages are typically small, and the delay caused by packet loss or retransmission is generally not critical for this application.
- 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.
    - The application layer can do some of the tasks through UDP-
        - Trace Route
        - Record Route
        - Timestamp
    - UDP takes a datagram from Network Layer, attaches its header, and sends it to the user. So, it works fast.
    - Actually, UDP is a null protocol if you remove the checksum field.
        1. Reduce the requirement of computer resources.
        2. When using the Multicast or Broadcast to transfer.
        3. The transmission of Real-time packets, mainly in multimedia applications.

Advantages of UDP:

1. Speed: UDP is faster than TCP because it does not have the overhead of establishing a connection and ensuring reliable data delivery.

2. Lower latency: Since there is no connection establishment, there is lower latency and faster response time.

3. Simplicity: UDP has a simpler protocol design than TCP, making it easier to implement and manage.

4. Broadcast support: UDP supports broadcasting to multiple recipients, making it useful for applications such as video streaming and online gaming.

5. Smaller packet size: UDP uses smaller packet sizes than TCP, which can reduce network congestion and improve overall network performance.

Disadvantages of UDP:

1. No reliability: UDP does not guarantee delivery of packets or order of delivery, which can lead to missing or duplicate data.

2. No congestion control: UDP does not have congestion control, which means that it can send packets at a rate that can cause network congestion.

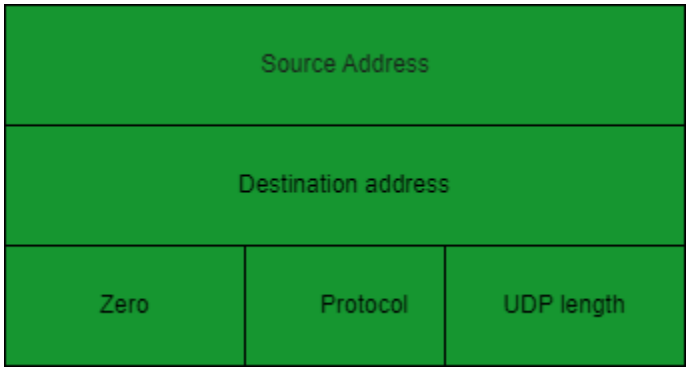
3. No flow control: UDP does not have flow control, which means that it can overwhelm the receiver with packets that it cannot handle.

4. Vulnerable to attacks: UDP is vulnerable to denial-of-service attacks, where an attacker can flood a network with UDP packets, overwhelming the network and causing it to crash.

5. Limited use cases: UDP is not suitable for applications that require reliable data delivery, such as email or file transfers, and is better suited for applications that can tolerate some data loss, such as video streaming or online gaming.

UDP PSEUDO HEADER:

- the purpose of using a pseudo-header is to verify that the UDP packet has reached its correct destination
- the correct destination consist of a specific machine and a specific protocol port number within that machine



UDP pseudo header

UDP pseudo header details:

- the UDP header itself specify only protocol port number.thus , to verify the destination UDP on the sending machine computes a checksum that covers the destination IP address as well as the UDP packet.
- at the ultimate destination, UDP software verifies the checksum using the destination IP address obtained from the header of the IP packet that carried the UDP message.
- if the checksum agrees, then it must be true that the packet has reached the intended destination host as well as the correct protocol port within that host.

User Interface:

A user interface should allow the creation of new receive ports, receive operations on the receive ports that returns the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and address to be sent.

IP Interface:

- the UDP module must be able to determine the source and destination internet address and the protocol field from internet header
- one possible UDP/IP interface would return the whole internet datagram including the entire internet header in response to a receive operation
- such an interface would also allow the UDP to pass a full internet datagram complete with header to the IP to send. the IP would verify certain fields for consistency and compute the internet header checksum.
- The IP interface allows the UDP module to interact with the network layer of the protocol stack, which is responsible for routing and delivering data across the network.
- The IP interface provides a mechanism for the UDP module to communicate with other hosts on the network by providing access to the underlying IP protocol.
- The IP interface can be used by the UDP module to send and receive data packets over the network, with the help of IP routing and addressing mechanisms.
- The IP interface provides a level of abstraction that allows the UDP

module to interact with the network layer without having to deal with the complexities of IP routing and addressing directly.
- The IP interface also handles fragmentation and reassembly of IP packets, which is important for large data transmissions that may exceed the maximum packet size allowed by the network.
- The IP interface may also provide additional services, such as support for Quality of Service (QoS) parameters and security mechanisms such as IPsec.
- The IP interface is a critical component of the Internet Protocol Suite, as it enables communication between hosts on the internet and allows for the seamless transmission of data packets across the network.

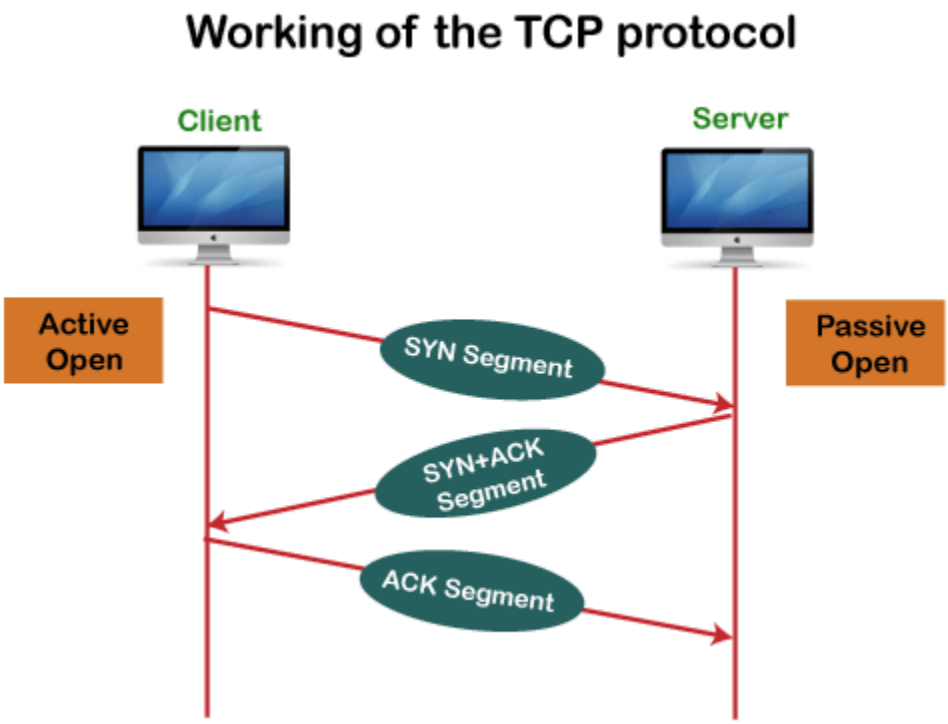
Transmission Control Protocol(TCP)

TCP stands for **Transmission Control Protocol**. It is a transport layer protocol that facilitates the transmission of packets from source to destination. It is a connection-oriented protocol that means it establishes the connection prior to the communication that occurs between the computing devices in a network. This protocol is used with an IP protocol, so together, they are referred to as a TCP/IP.

The main functionality of the TCP is to take the data from the application layer. Then it divides the data into a several packets, provides numbering to these packets, and finally transmits these packets to the destination. The TCP, on the other side, will reassemble the packets and transmits them to the application layer. As we know that TCP is a connection-oriented protocol, so the connection will remain established until the communication is not completed between the sender and the receiver.

Working of TCP

In TCP, the connection is established by using three-way handshaking. The client sends the segment with its sequence number. The server, in return, sends its segment with its own sequence number as well as the acknowledgement sequence, which is one more than the client sequence number. When the client receives the acknowledgment of its segment, then it sends the acknowledgment to the server. In this way, the connection is established between the client and the server.



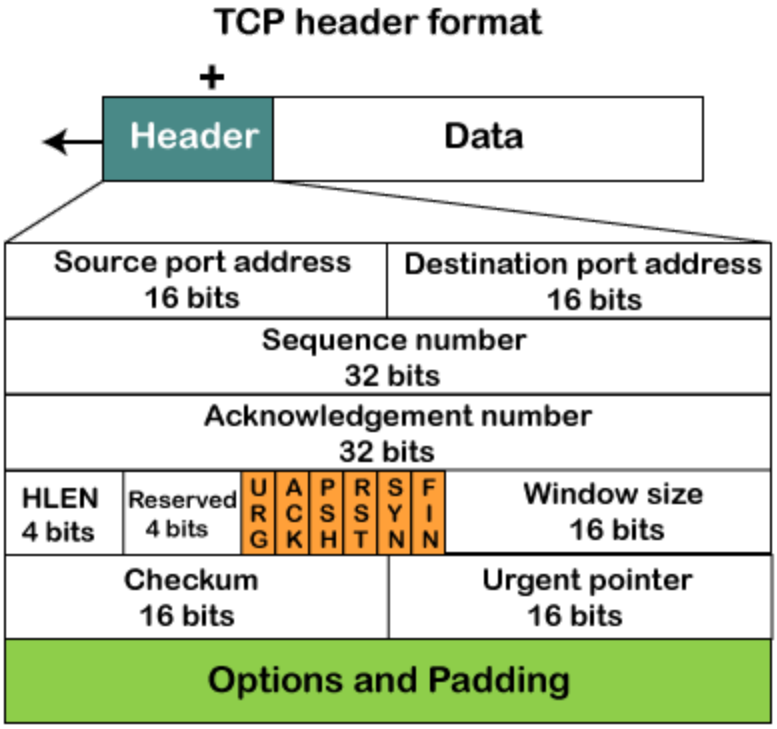
Advantages of TCP

- It provides a connection-oriented reliable service, which means that it guarantees the delivery of data packets. If the data packet is lost across the network, then the TCP will resend the lost packets.
- It provides a flow control mechanism using a sliding window protocol.
- It provides error detection by using checksum and error control by using Go Back or ARP protocol.
- It eliminates the congestion by using a network congestion avoidance algorithm that includes various schemes such as additive increase/multiplicative decrease (AIMD), slow start, and congestion window.

Disadvantage of TCP

It increases a large amount of overhead as each segment gets its own TCP header, so fragmentation by the router increases the overhead.

TCP Header Format



- **Source port:** It defines the port of the application, which is sending the data. So, this field contains the source port address, which is 16 bits.
- **Destination port:** It defines the port of the application on the receiving side. So, this field contains the destination port address, which is 16 bits.
- **Sequence number:** This field contains the sequence number of data bytes in a particular session.
- **Acknowledgment number:** When the ACK flag is set, then this contains the next sequence number of the data byte and works as an acknowledgment for the previous data received. For example, if the receiver receives the segment number 'x', then it responds 'x+1' as an acknowledgment number.
- **HLEN:** It specifies the length of the header indicated by the 4-byte words in the header. The size of the header lies between 20 and 60 bytes. Therefore, the value of this field would lie between 5 and 15.
- **Reserved:** It is a 4-bit field reserved for future use, and by default, all are set to zero.
- **Flags**

There are six control bits or flags:

1. **URG:** It represents an urgent pointer. If it is set, then the data is processed urgently.
2. **ACK:** If the ACK is set to 0, then it means that the data packet does not contain an acknowledgment.
3. **PSH:** If this field is set, then it requests the receiving device to push the data to the receiving application without buffering it.
4. **RST:** If it is set, then it requests to restart a connection.
5. **SYN:** It is used to establish a connection between the hosts.
6. **FIN:** It is used to release a connection, and no further data exchange will happen.

- **Window size**
  It is a 16-bit field. It contains the size of data that the receiver can accept. This field is used for the flow control between the sender and receiver and also determines the amount of buffer allocated by the receiver for a segment. The value of this field is determined by the receiver.
- **Checksum**
  It is a 16-bit field. This field is optional in UDP, but in the case of TCP/IP, this field is mandatory.
- **Urgent pointer**
  It is a pointer that points to the urgent data byte if the URG flag is set to 1. It defines a value that will be added to the sequence number to get the sequence number of the last urgent byte.
- **Options**
  It provides additional options. The optional field is represented in 32-bits. If this field contains the data less than 32-bit, then padding is required to obtain the remaining bits.