5. Transport Layer

The transport layer is the layer in the open system interconnection (OSI) model responsible for end-to-end communication over a network. It provides logical communication between application processes running on different hosts within a layered architecture of protocols and other network components.

The transport layer is also responsible for the management of error correction, providing quality and reliability to the end user. This layer enables the host to send and receive error corrected data, packets or messages over a network and is the network component that allows multiplexing.

The following are the design issues for the layers:

1. **Reliability:** It is a design issue of making a network that operates correctly even when it is made up of unreliable components.
2. **Addressing:** There are multiple processes running on one machine. Every layer needs a mechanism to identify senders and receivers.
3. **Error Control**: It is an important issue because physical communication circuits are not perfect. Many error detecting and error correcting codes are available. Both sending and receiving ends must agree to use any one code.
4. **Flow Control:** If there is a fast sender at one end sending data to a slow receiver, then there must be flow control mechanism to control the loss of data by slow receivers. This property leads to mechanisms for disassembling, transmitting and the reassembling messages.
5. **Multiplexing and De-multiplexing:** If the data has to be transmitted on transmission media separately, it is inconvenient or expensive to setup separate connection for each pair of communicating processes. So, multiplexing is needed in the physical layer at sender end and de-multiplexing is need at the receiver end.
6. **Scalability**: When network gets large, new problem arises. Thus scalability is important so that network can continue to work well when it gets large.
7. **Routing:** When there are multiple paths between source and destination, only one route must be chosen. This decision is made on the basis of several routing algorithms, which chooses optimized route to the destination.
8. **Confidentiality and Integrity:** Network security is the most important factor. Mechanisms that provide confidentiality defend against threats like eavesdropping. Mechanisms for integrity prevent faulty changes to messages.
9. Accepting data from Session layer, split it into segments and send to the network layer.
10. Ensure correct delivery of data with efficiency.
11. Isolate upper layers from the technological changes.
12. Error control and flow control.

B) Service Primitive, QoS

What are Service Primitives?

A service is formally specified by a set of primitives (operations) available to a user process to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, as it often is, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets. The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connection-less service. There are five types of service primitives :

1. **LISTEN :** When a server is ready to accept an incoming connection it executes the LISTEN primitive. It blocks waiting for an incoming connection.

2. **CONNECT :** It connects the server by establishing a connection. Response is awaited.
3. **RECIEVE:** Then the RECIEVE call blocks the server.
4. **SEND :** Then the client executes SEND primitive to transmit its request followed by the execution of RECIEVE to get the reply. Send the message.
5. **DISCONNECT :** This primitive is used for terminating the connection. After this primitive one can't send any message. When the client sends DISCONNECT packet then the server also sends the DISCONNECT packet to acknowledge the client. When the server package is received by client then the process is terminated.

Connection Oriented Service Primitives

There are 4 types of primitives for Connection Oriented Service :

CONNECT	**This primitive makes a connection**
DATA, DATA-ACKNOWLEDGE, EXPEDITED-DATA	Data and information is sent using thus primitive
CONNECT	Primitive for closing the connection
RESET	Primitive for reseting the connection

Connectionless Oriented Service Primitives

There are 2 types of primitives for Connectionless Oriented Service:

UNIDATA	**This primitive sends a packet of data**
FACILITY, REPORT	Primitive for enquiring about the performance of the network, like delivery statistics.

Consider an application with a server and a number of remote clients.

a) To start with, the server executes a LISTEN primitive, typically by calling a library procedure that makes a system call to block the server until a client turns up.
b) For lack of a better term, we will reluctantly use the somwhat ungainly acronym TPDU(Transport Protocol Data Unit) for message sent from transport entity to transport entity.
c) Thus, TPDUs(exchanged by the transport layer) are contained in packets(exchanged by the network layer).
d) In turn, packets are contained in frames(exchanged by the data link layer).
e) When a frame arrives, the data link layer processes the frame header and passes the contents of the frame payload field up to the network entity.
f) When a client wants to talk to the server, it executes a CONNECT primitive.
g) The transport entity carries out this primitives by blocking the caller and sending a packet to the server.
h) Encapsulated in the payload of this packet is a transport layer message for the server's transport entity.
i) The client's CONNECT call causes a CONNECTION REQUEST TPDU to be sent to the server.
j) When it arrives, the transport entity checks to see that the server is blocked on a LISTEN.
k) It then unblocks the server and sends a CONNECTION ACCEPTED TPDU back to the client.
l) When this TPDU arrives, the client is unblocked and the connection is established. Data can now be exchanged using the SEND and RECEIVE primitives.
m) In the simplest form, either party can do a(blocking)RECEIVE to wait for the other party to do a SEND. When the TPDU arrives, the receiver is unblocked.
n) It can then process the TPDU and send a reply. As long as both sides can keep track of whose turn it is to send, this scheme works fine.
o) When a connection is no longer needed, it must be released to free up table space within the two transport entities.

Quality OF Service (QOS)

QoS is an overall performance measure of the computer network. Important flow characteristics of the QoS are given below:

i. Reliability

If a packet gets lost or acknowledgement is not received (at sender), the re-transmission of data will be needed. This decreases the reliability.

The importance of the reliability can differ according to the application.

For example:

E- mail and file transfer need to have a reliable transmission as compared to that of an audio conferencing.

ii. Delay

Delay of a message from source to destination is a very important characteristic. However, delay can be tolerated differently by the different applications.

For example:

The time delay cannot be tolerated in audio conferencing (needs a minimum time delay), while the time delay in the e-mail or file transfer has less importance.

iii. Jitter

The jitter is the variation in the packet delay.

If the difference between delays is large, then it is called as high jitter. On the contrary, if the difference between delays is small, it is known as low jitter.

Example:

Case1: If 3 packets are sent at times 0, 1, 2 and received at 10, 11, 12. Here, the delay is same for all packets and it is acceptable for the telephonic conversation.

Case2: If 3 packets 0, 1, 2 are sent and received at 31, 34, 39, so the delay is different for all packets. In this case, the time delay is not acceptable for the telephonic conversation.

iv. Bandwidth

Different applications need the different bandwidth.

For example:

Video conferencing needs more bandwidth in comparison to that of sending an e-mail.

Integrated Services and Differentiated Service.

These two models are designed to provide Quality of Service (QoS) in the network.

1. Integrated Services(IntServ)

Integrated service is flow-based QoS model and designed for IP.

In integrated services, user needs to create a flow in the network, from source to destination and needs to inform all routers (every router in the system implements IntServ) of the resource requirement.

Following are the steps to understand how integrated services works.

i. Resource Reservation Protocol (RSVP)

An IP is connectionless, datagram, packet-switching protocol. To implement a flow-based model, a signaling protocol is used to run over IP, which provides the signaling mechanism to make reservation (every applications need assurance to make reservation), this protocol is called as RSVP.

ii. Flow Specification

While making reservation, resource needs to define the flow specification. The flow specification has two parts:

- Resource specification
  It defines the resources that the flow needs to reserve. For example: Buffer, bandwidth, etc.
- Traffic specification
  It defines the traffic categorization of the flow.

iii. Admit or deny

After receiving the flow specification from an application, the router decides to admit or deny the service and the decision can be taken based on the previous commitments of the router and current availability of the resource.

Problems with Integrated Services.

The two problems with the Integrated services are:

i. Scalability

In Integrated Services, it is necessary for each router to keep information of each flow. But, this is not always possible due to growing network.

ii. Service- Type Limitation

The integrated services model provides only two types of services, guaranteed and control-load.

2. Differentiated Services (DS or Diffserv):

- DS is a computer networking model, which is designed to achieve the scalability by managing the network traffic.
- DS is a class based QoS model specially designed for IP.
- DS was designed by IETF (Internet Engineering Task Force) to handle the problems of Integrated Services.

The solutions to handle the problems of Integrated Services are explained below:

- Scalability

The main processing unit can be moved from central place to the edge of the network to achieve the scalability. The router does not need to store the information about the flows and the applications (or the hosts) define the type of services they want every time while sending the packets.

- Service Type Limitation

The routers, route the packets on the basis of class of services define in the packet and not by the flow. This method is applied by defining the classes based on the requirement of the applications.

C) Connection-oriented and Connectionless Networks:

1. Connection-oriented service

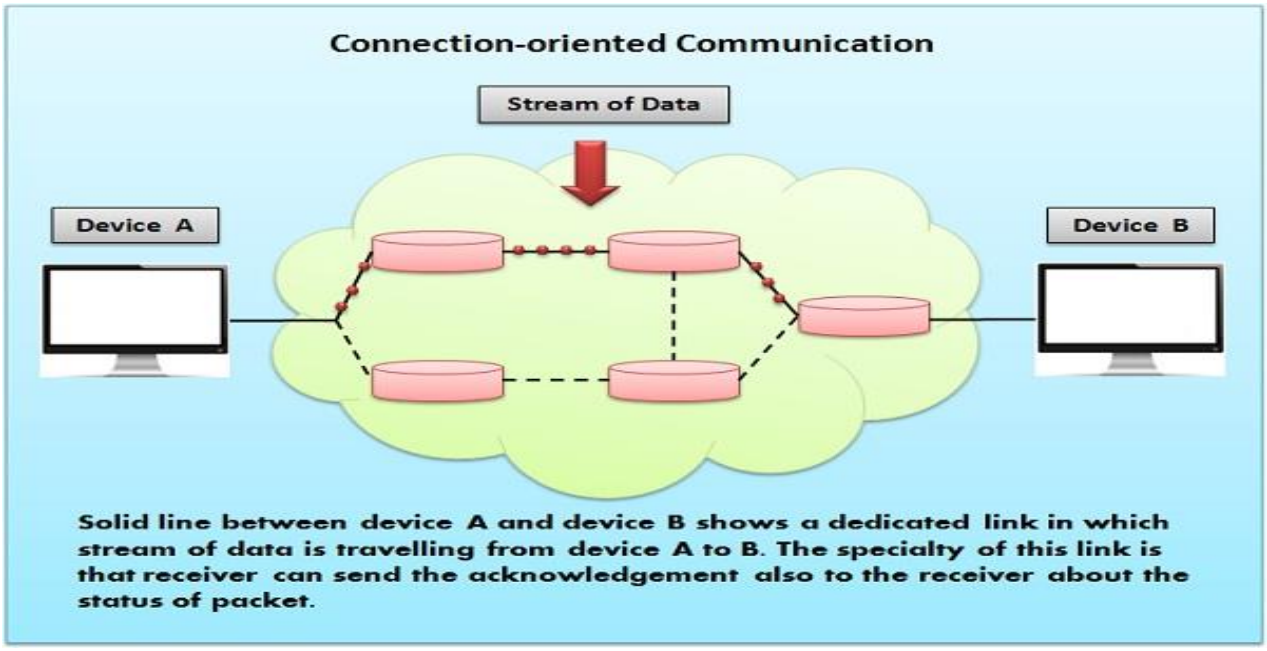
Connection-oriented service works more like a telephone system in where we pick up the phone, make a call, talk, and then hang up the phone. Similarly, there is a dedicated line between the sender and the receiver on a network when it comes to connection-oriented service.

Connection-oriented service works more like a virtual tube in which the sender sends the data and they arrive in the order they were pushed by the sender to the receiver. **Reliability** is achieved by having recipient acknowledge each message. There are **sequencing** and **flow control,** that's the reason packets received at the receiving end are always in **order**. It uses **circuit switching** for transmission of data.

TCP provides Connection-oriented services as does **ATM, Frame Relay** and **MPLS**hardware. It uses **handshake process** to establish the connection between the sender and receiver.

A handshake process includes some steps which are:

- Client requests server to set up a connection for transfer of data.
- Server program notifies its TCP that connection can be accepted.
- The client transmits a SYN segment to the server.
- The server sends SYN+ACK to the client.
- Client transmits 3rd segment i.e. just ACK segment.
- Then server terminates the connection.



There can also be different variants of connection-oriented services based on their functionalities at the layer two:

i. Store and forward switching

In store and forward switching, an intermediate node stores the entire message before forwarding it to the next node. Its advantage is that when a packet gets lost, instead of actual receiver asking for retransmission, the intermediate node does the retransmission work. Also, the retransmission time is reduced and saved by a great deal.

ii. Byte streaming

In byte streaming, packets are sent in any sequence. The order in which the packets are sent is not so important. For example – when we are downloading a movie from the internet, the byte sequence is not so important but the whole film.

iii. Acknowledged datagram

An acknowledgment is also sent for each packet transmitted to the receiver. If the acknowledgment is not received for a particular set of packets, the receiver waits for some time before asking for the retransmission.

iv. Request-reply service

Request-reply service is an improvement to the ACK datagram. In this service, an ACK is not sent by the receiver for every packet received rather the data is retransmitted on request by the receiver. That's why its called request-reply service.

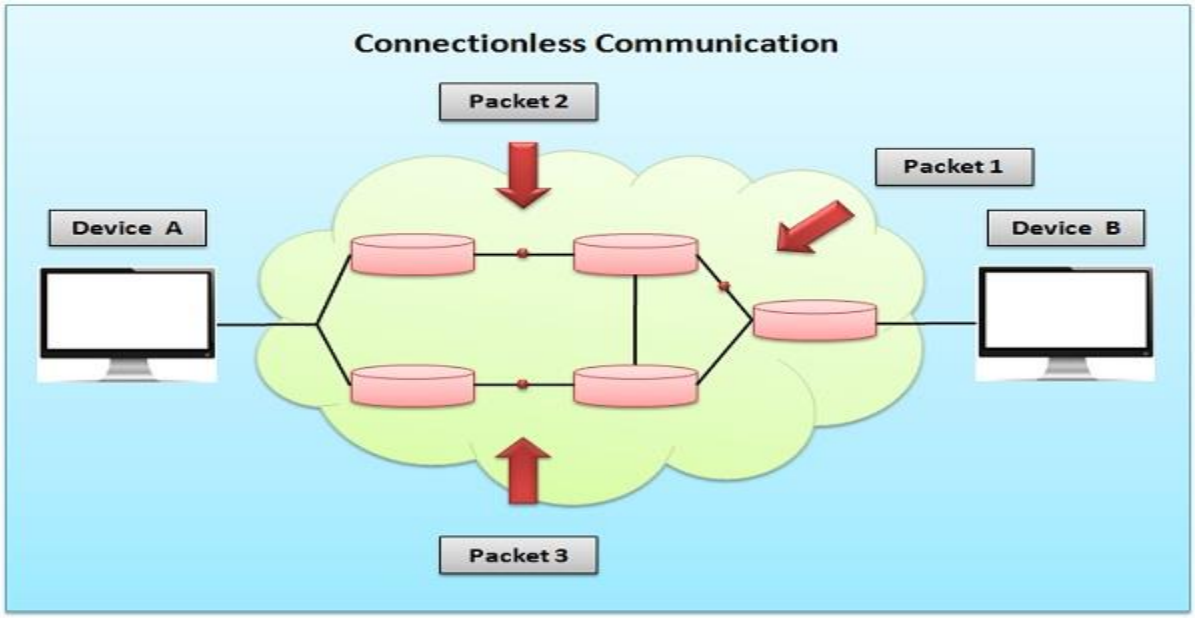
2. Connectionless service

In connectionless service, there is no initial setup of the connection between the sender and the receiver for the data transfer. That means it cannot guarantee the delivery and QoS of the packets. Thus, the connectionless service becomes synonymous with unreliable service.

Connection-less service is analogous to the **postal system**. In which packets of data (usually known as a **datagram**) is transmitted from source to destination directly. Each packet is treated as an individual entity, which allows communication entities to send data before establishing communication. Each packet carries a **destination address** to identify the intended recipient.

Packets don't follow a **fixed path** that is the reason the packets received at receiver end can be out of order. It uses **packet switching** for transmission of data.

Most network hardware, the **Internet Protocol (IP)**, and the **User Datagram Protocol (UDP)** provides connection-less service.



i. Cut through Switching

Unlike store and forward switching, in cut-through switching, the intermediate node does not store any packet rather it keeps on forwarding the packets in the order they are received.

ii. Contrast to the real world scenario

If we look at the OSI layer, initially, guaranteed data delivery used to be implemented at the lower layers such as data link layer and network layer. That is so because, in PSTN (Public Switched Telephone Network), the data reliability was very less because of the poor-quality telephone connections.

If we look at the current scenario, networks have become more reliable due to newer technologies. So, the first three underlying layers are now made to focus more on the data speed and delivery rate. With time, the reliability and acknowledgment were pushed up to the upper layers such as transport layer.

A common question is also often asked that **why would one need an unreliable service at first hand itself?** If we look at the OSI layer, we can realize that when the internet came into existence, the whole of the internet was built on the unreliable communication only. Later, when QoS and other things were introduced to the internet, connection-oriented services were needed.

Situations when UDP is preferred over TCP :-SNMP (Simple Network Management Protocol), DNS (Domain Name System), Internet Radio, LAN requests, Router updates exchanges etc.

BASIS OF COMPARISON	CONNECTION-ORIENTED SERVICE	CONNECTION-LESS SERVICE
Prior Connection Requirement	Necessary	Not required
Reliability	Ensures reliable transfer of data.	Not guaranteed.
Congestion	Unlikely	Occur likely.
Transferring mode	It can be implemented using circuit switching and virtual circuit.	It is implemented using packet switching.
Lost data retransmission	Feasible	Practically, not possible.
Suitability	Suitable for long and steady communication.	Suitable for bursty Transmission.
Signalling	Used for connection establishment.	There is no concept of signalling.
Packet forwarding	Packets sequentially travel to their destination node and follows the same route.	Packets reach the destination randomly without following the same route.
Delay	There is a delay in transfer of information, but once the connection is established faster delivery can be achieved.	Because to the absence of connection establishment phase, the transmission is faster.
Resource Allocation	Need to be allocated.	No prior allocation of the resource is required.

D) Transport Layer Protocols: TCP and UDP

1. Transmission Control Protocol (TCP)

The transmission Control Protocol (TCP) is one of the most important protocols of Internet Protocols suite. It is most widely used protocol for data transmission in communication network such as internet.

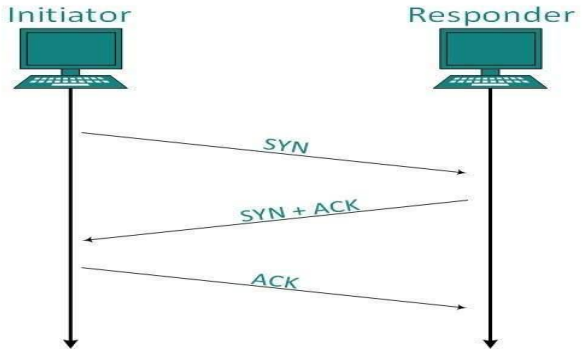
- TCP is reliable protocol. That is, the receiver always sends either positive or negative acknowledgement about the data packet to the sender, so that the sender always has bright clue about whether the data packet is reached the destination or it needs to resend it.
- TCP ensures that the data reaches intended destination in the same order it was sent.
- TCP provides error-checking and recovery mechanism.
- TCP provides end-to-end communication.
- TCP provides flow control and quality of service.
- TCP operates in Client/Server point-to-point mode.
- TCP provides full duplex server, i.e. it can perform roles of both receiver and sender.

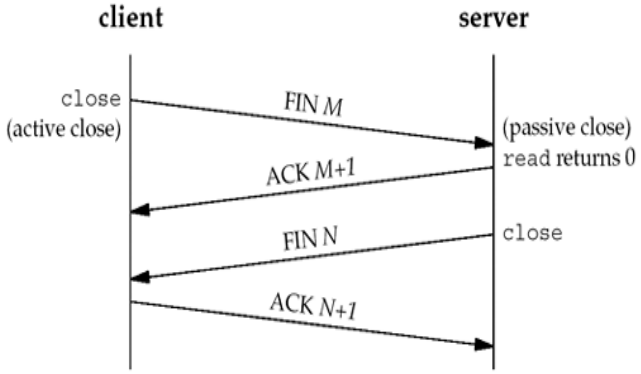
a) Connection Oriented Protocol/Connection Management:

TCP communication works in Server/Client model. The client initiates the connection and the server either accepts or rejects it. Three-way handshaking is used for connection management.

i. Establishment

Client initiates the connection and sends the segment with a Sequence number. Server acknowledges it back with its own Sequence number and ACK of client's segment which is one more than client's Sequence number. Client after receiving ACK of its segment sends an acknowledgement of Server's response. And the Connection is established.





ii. Release

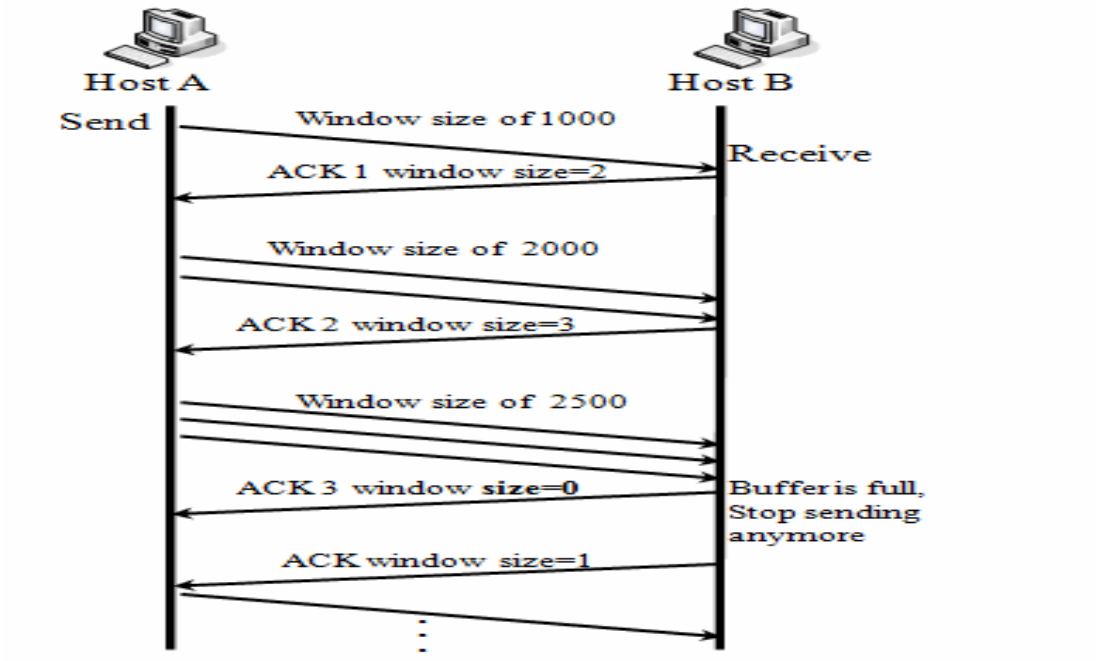
Either of server and client can send TCP segment with FIN flag set to 1. When the receiving end responds it back by ACKnowledging FIN, that direction of TCP communication is closed and connection is released. Otherwise if other side doesn't want to end connection it sends ACK signal to wait for some time.

b) Flow Control Through use of Window Size/Bandwidth Management and Congestion Control

TCP uses the concept of window size to accommodate the need of Bandwidth management. Window size tells the sender at the remote end, the number of data byte segments the receiver at this end can receive. TCP uses slow start phase by using window size 1 and increases the window size exponentially after each successful communication.

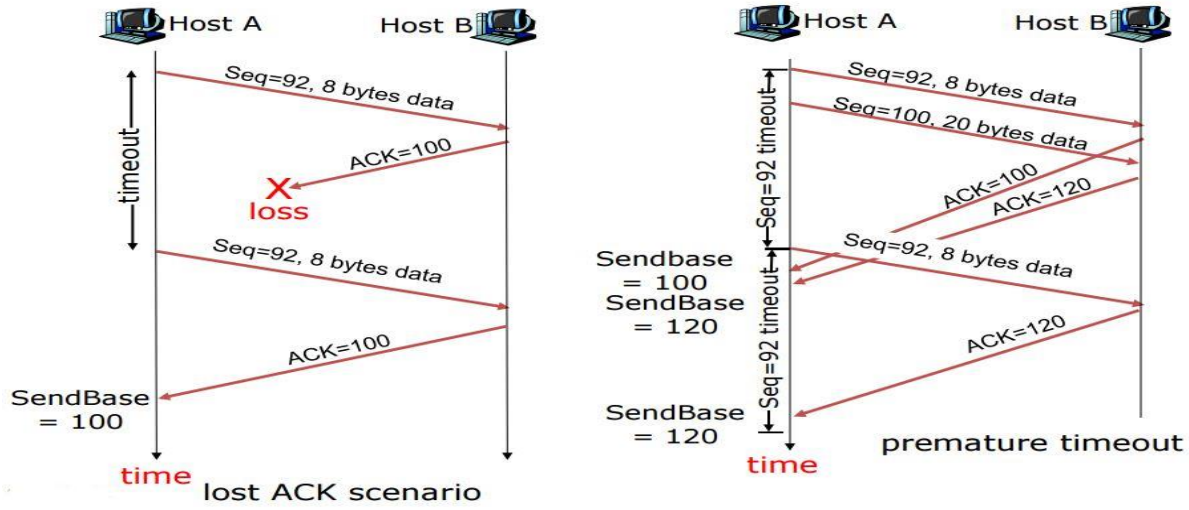
For example, the client uses windows size 2 and sends 2 bytes of data. When the acknowledgement of this segment received the windows size is doubled to 4 and next sent the segment sent will be 4 data bytes long. When the acknowledgement of 4-byte data segment is received, the client sets windows size to 8 and so on.

If an acknowledgement is missed, i.e. data lost in transit network or it received NACK, then the window size is reduced to half and slow start phase starts again.
When large amount of data is fed to system which is not capable of handling it, congestion occurs. TCP controls congestion by means of Window mechanism. TCP sets a window size telling the other end how much data segment to send.



c) Reliable Delivery or retransmission: Through use of Acknowledgement (Error Control &and Flow Control and Timer Management)

TCP uses port numbers to know what application process it needs to handover the data segment. Along with that, it uses sequence numbers to synchronize itself with the remote host. All data segments are sent and received with sequence numbers. The Sender knows which last data segment was received by the Receiver when it gets ACK. The Receiver knows about the last segment sent by the Sender by referring to the sequence number of recently received packet.
If the sequence number of a segment recently received does not match with the sequence number the receiver was expecting, then it is discarded and NACK is sent back. If two segments arrive with the same sequence number, the TCP timestamp value is compared to make a decision.
TCP uses different types of timer to control and management various tasks.



d) Application Multiplexing through use of port no

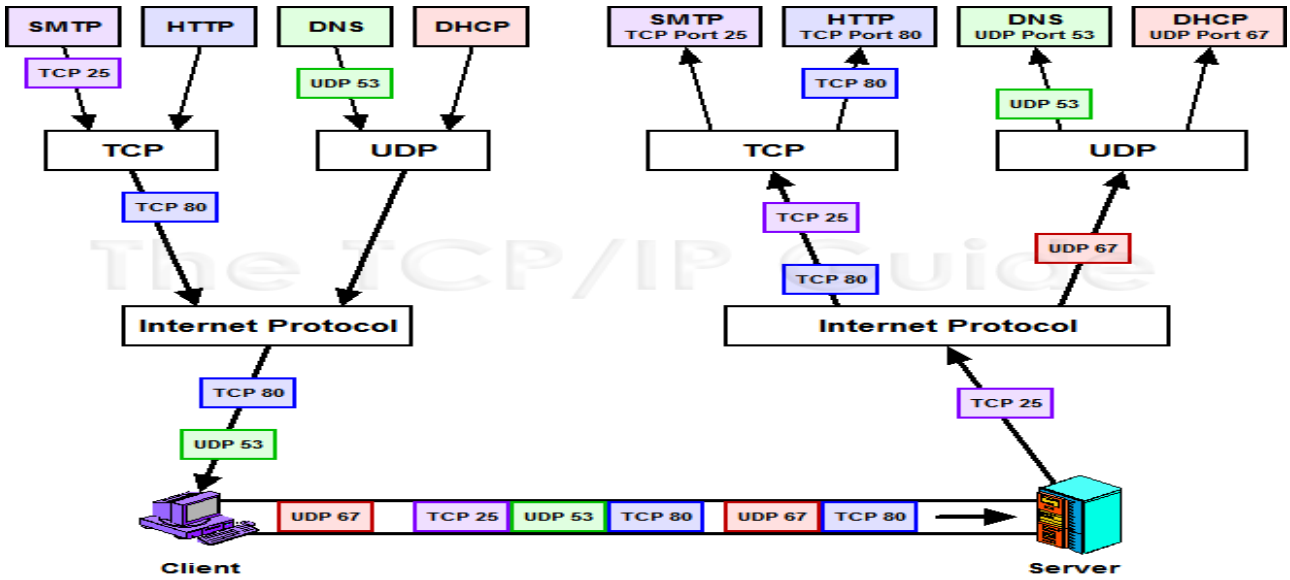
The technique to combine two or more data streams in one session is called Multiplexing. When a TCP client initializes a connection with Server, it always refers to a well-defined port number which indicates the application process. The client itself uses a randomly generated port number from private port number pools. Using TCP Multiplexing, a client can communicate with a number of different application process in a single session. For example, a client requests a web page which in turn contains different types of data (HTTP, SMTP, FTP etc.) the TCP session timeout is increased and the session is kept open for longer time so that the three-way handshake overhead can be avoided. This enables the client system to receive multiple connection over single virtual connection. These virtual connections are not good for Servers if the timeout is too long.

It creates multiplexing such that for every port there is no need of handshaking and every application can use the same ip but with different port to differentiate application.

The combination of ip address and port no is separated by colon that is used to uniquely identify an connection in socket: 19.16.1.5:5555

Addressing: TCP communication between two remote hosts is done by means of port numbers (TSAPs). Ports numbers can range from 0 – 65535 which are divided as:

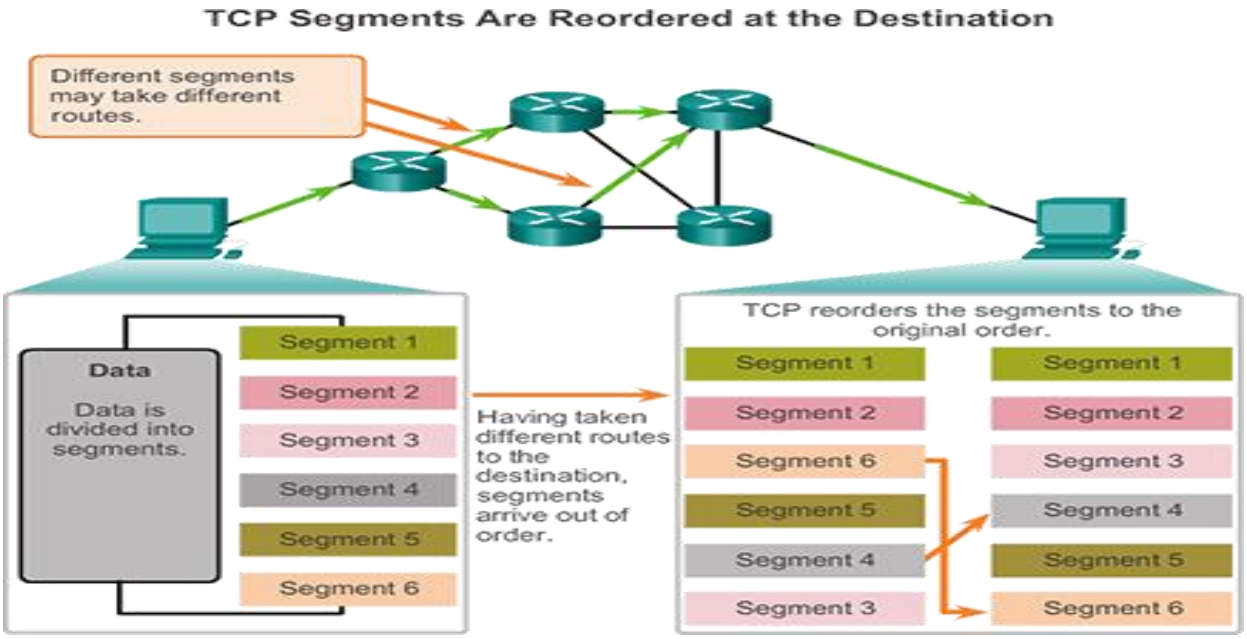
- Well known port (Reserved by common application) :- 0 – 1023
- Registered Port (Used by registered application, can be used by any application if free):- 1024 – 49151
- Dynamic Ports (Assigned dynamically to application) :- 49152 – 65535



e) Segmentation Ordered delivery and reassembly: Through use of sequence no/Crash Recovery

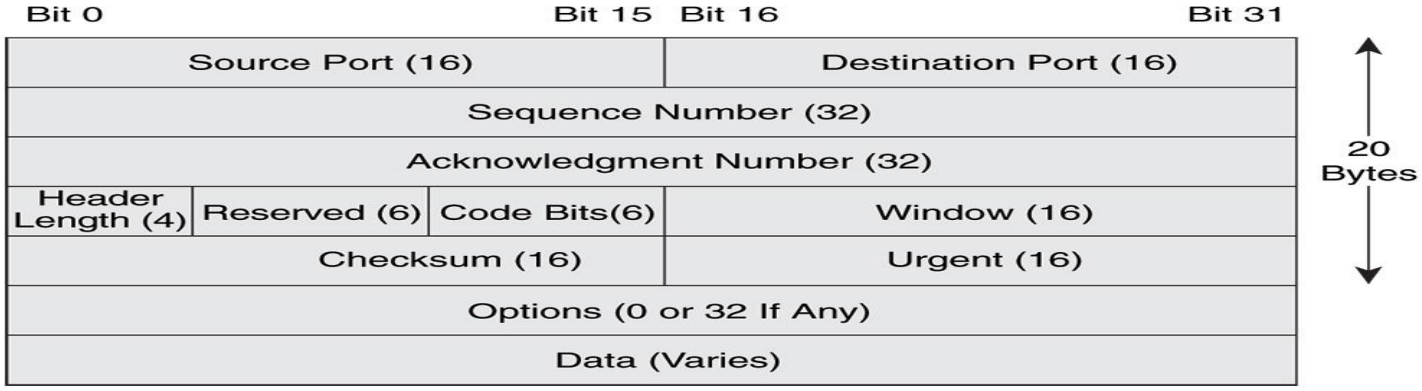
TCP is very reliable protocol. It provides sequence number to each of byte sent in segment. It provides the feedback mechanism i.e. when a host receives a packet, it is bound to ACK that packet having the next sequence number expected (if it is not the last segment).

When a TCP Server crashes mid-way communication and re-starts its process it sends TPDU broadcast to all its hosts. The hosts can then send the last data segment which was never unacknowledged and carry onwards.



TCP Header

The length of TCP header is minimum 20 bytes long and maximum 60 bytes.



- **Source Port (16-bits)** - It identifies source port of the application process on the sending device.
- **Destination Port (16-bits)** - It identifies destination port of the application process on the receiving device.
- **Sequence Number (32-bits)** - Sequence number of data bytes of a segment in a session.
- **Acknowledgement Number (32-bits)** - When ACK flag is set, this number contains the next sequence number of the data byte expected and works as acknowledgement of the previous data received.
- **Data Offset (4-bits)** - This field implies both, the size of TCP header (32-bit words) and the offset of data in current packet in the whole TCP segment.
- **Reserved (3-bits)** - Reserved for future use and all are set zero by default.
- **Flags (1-bit each)**
  - **SYN** - This flag is used to set up a connection between hosts.
  - **FIN** - This flag is used to release a connection and no more data is exchanged thereafter.
- **Windows Size** - This field is used for flow control between two stations and indicates the amount of buffer (in bytes) the receiver has allocated for a segment, i.e. how much data is the receiver expecting.
- **Checksum** - This field contains the checksum of Header, Data and Pseudo Headers.
- **Urgent Pointer** - It points to the urgent data byte if URG flag is set to 1.
- **Options** - It facilitates additional options which are not covered by the regular header. Option field is always described in 32-bit words. If this field contains data less than 32-bit, padding is used to cover the remaining bits to reach 32-bit boundary.

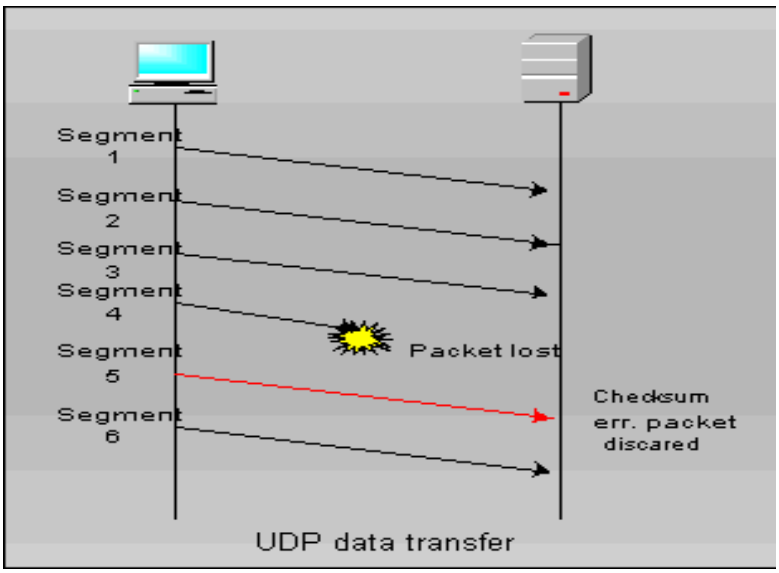
2. User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is simplest Transport Layer communication protocol available of the TCP/IP protocol suite. It involves minimum amount of communication mechanism. UDP is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism.

In UDP, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This shortcoming makes this protocol unreliable as well as easier on processing.

Requirement of UDP

A question may arise, why do we need an unreliable protocol to transport the data? We deploy UDP where the acknowledgement packets share significant amount of bandwidth along with the actual data. For example, in case of video streaming, thousands of packets are forwarded towards its users. Acknowledging all the packets is troublesome and may contain huge amount of bandwidth wastage. The best delivery mechanism of underlying IP protocol ensures best efforts to deliver its packets, but even if some packets in video streaming get lost, the impact is not calamitous and can be ignored easily. Loss of few packets in video and voice traffic sometimes goes unnoticed.



Time-sensitive applications often use UDP because if dropped packets means less delayed information then UDP it is. In connectionless protocols there is no effort made to set up a connection like TCP, UDP just sends the information in one direction from the source address to the destination address and hopes it arrives in complete package but if not, the protocol does not care.

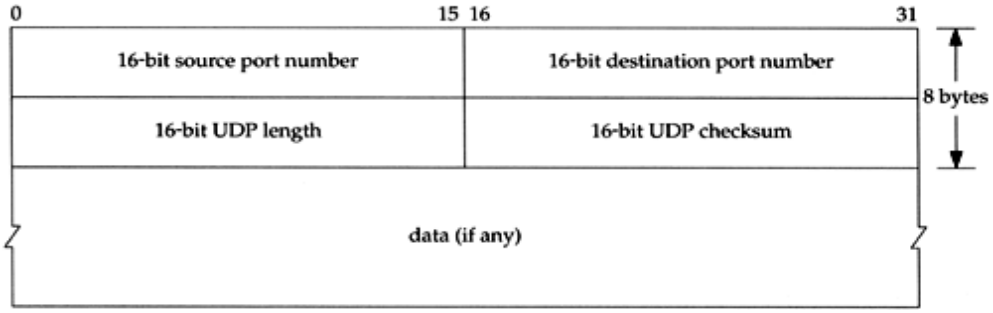
So to review:

- UDP is Unreliable – When a message is sent, it cannot or will not know if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, re-transmission and or timeout like the TCP protocol.
- UDP is not ordered – If two messages are sent to the same destination, the order in which they arrive cannot be predicted.
- UDP is lightweight – There is no ordering of messages, any tracking connections, etc. It is a small transport layer designed on top of IP.
- Inside UDP Datagrams – Packets that are sent individually and are guaranteed to be whole if they arrive. Packets have definite bounds and do not split or merge into data streams. (Meaning that UDP packets are not coherent)

Features

- UDP is used when acknowledgement of data does not hold any significance.
- UDP is good protocol for data flowing in one direction.
- UDP is simple and suitable for query based communications.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.

UDP Header



UDP header contains four main parameters:

- **Source Port** - This 16 bits information is used to identify the source port of the packet.
- **Destination Port** - This 16 bits information, is used identify application level service on destination machine.
- **Length** - Length field specifies the entire length of UDP packet (including header). It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header itself.
- **Checksum** - This field stores the checksum value generated by the sender before sending. IPv4 has this field as optional so when checksum field does not contain any value it is made 0 and all its bits are set to zero.

E) Elements of Transport Layer

The transport service is implemented by a transport protocol.

1. **Addressing in TCP and UDP**

A typical host has many application processes running on it. These processes generate data that is sent to either TCP or UDP, which in turn passes it to IP for transmission. This multiplexed stream of datagrams is sent out by the IP layer to various destinations. Similarly, a device's IP layer receives datagrams, these datagrams need to be de-multiplexed, so they end up at the correct process on the device that receives them.

2. **Multiplexing and De-multiplexing Using Ports:**

The question is: How do we de-multiplex a sequence of IP datagrams that need to go to different application processes on the same host? Let's consider a particular host bearing the IP address 24.156.79.20. How does the IP layer know which packet goes to which process on the same host?

The first part of the answer lies in the Protocol field included in the header of each IP datagram. This field carries a code that identifies the protocol that sent the data. Since most end-user applications use TCP or UDP at the transport layer, the Protocol field in a received datagram tells IP to pass data to either TCP or UDP as appropriate.

But both TCP and UDP are used by many applications at once. This means TCP or UDP must figure out which process to send the data to. To make this possible, an additional addressing element is necessary. This transport layer address is called a port. Each port number within a particular IP device identifies a particular application process.

3. **Source Port and Destination Port Numbers:**

In both UDP and TCP messages, two addressing fields appear, for a Source Port and a Destination Port. These are analogous to the fields for source address and destination address at the IP level, but at a higher level of detail. While the source address and destination address at the IP level identify the source machine and the destination machine; The Source Port and a Destination Port identify the originating process on the source machine, and the destination process on the destination machine. TCP and UDP port numbers are 16-bits in length, here's the basics of how transport-layer addressing (port addressing) works in TCP and UDP:

i. **Sending Datagrams:**

An application specifies the source and destination port it wishes to use for the communication. These are encoded into the TCP or UDP header, depending on which transport layer protocol the application is using. When TCP or UDP passes data to IP, IP indicates the protocol type (TCP or UDP) in the Protocol field of the IP datagram. The source and destination port numbers are encapsulated as part of the TCP or UDP message, Within the IP datagram's data area.

ii. **Receiving Datagrams:**

The IP software receives the datagram, inspects the Protocol field and decides to which protocol the datagram belongs (TCP or UDP). TCP or UDP receives the datagram and passes its contents to the appropriate process based on the destination port number.

FLOW CONTROL and BUFFERING

1. **Buffering**

a) Sender:

- In case of an unreliable network service the sending transport entity must buffer all TPDU's as it might need to retransmit them.
- In case of a reliable network service the sender may/may not buffer outgoing TPDU's.

b) Receiver:

- If all TPDU's are of same size, there will be a pool of identically sized buffers (one TPDU per buffer).
- If there is a wide variation in the TPDU sizes, variable sized buffer are used.
- Another option is to dedicate a single large buffer per connection.

2. **Dynamic Buffer Management**

In cases where traffic patterns change, the sender and receiver need to dynamically adjust their buffer allocations.

Working:

a) Sender requests receiver to allocate certain number of buffers to it.
b) The receiver depending on its load allocates as many buffers as possible.
c) When the sender sends a TPDU it decrements its allocation.
d) When the sender finishes its allocated quota it stops sending.
e) The receiver piggybacks acknowledgement and further buffer allocations on the reverse traffic. There might be a case where there are sufficient buffers at the sender and receiver but the subnet is congested. Therefore, along with buffering, flow control should also concentrate on the carrying capacity of the subnet.

3. **Multiplexing**

a) **Upward Multiplexing:**

If only one network address is available on a host all the transport connections on that machine use that single address. When a TPDU comes in, upward multiplexing is used to decide which process to give it to.

b) **Downward Multiplexing:**

Assume that a subnet uses virtual circuits internally and a maximum data rate is imposed on each one. If a user needs more bandwidth than what a single virtual circuit can provide, downward multiplexing can be used to open multiple network connection and distribute the traffic among them.

TRANSMISSION CONTROL PROTOCOL (TCP)	USER DATAGRAM PROTOCOL (UDP)
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 type of network transmission.
TCP is reliable as it guarantees delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
TCP provides extensive error checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error checking mechanism using checksums.
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 ordering is required, it has to be managed by the application layer.
TCP is comparatively slower than UDP.	UDP is faster, simpler and more efficient than TCP.
Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in User Datagram Protocol (UDP).
TCP header size is 20 bytes.	UDP Header size is 8 bytes.
TCP is heavy-weight.	UDP is lightweight.
TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.