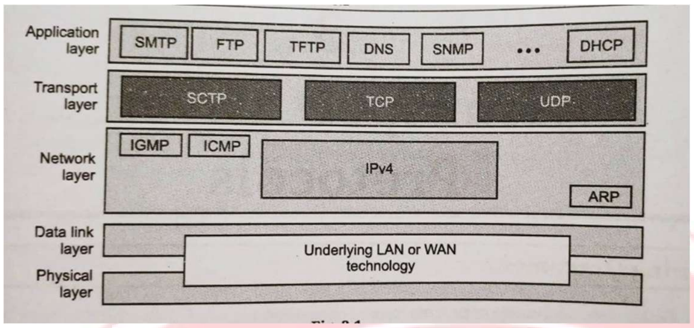
UNIT 3 TRANSPORT LAYER PROTOCOLS



- The Application player program interact with each other using services of transport layer
- Transport layer takes services from network layer and provides services to application layer.

Q: What are different transport layer services

1 Process to Process Delivery

- Data link layer performs delivery of frames between two neighbouring nodes over a link → node to node communication
- Network layer carries out datagram between two hosts → Host to host delivery.
- But the real communication takes between processes or application programs → process to process delivery
- Here packet from one process is delivered to other process.
- Relationship between communicating processes → client server relationship.

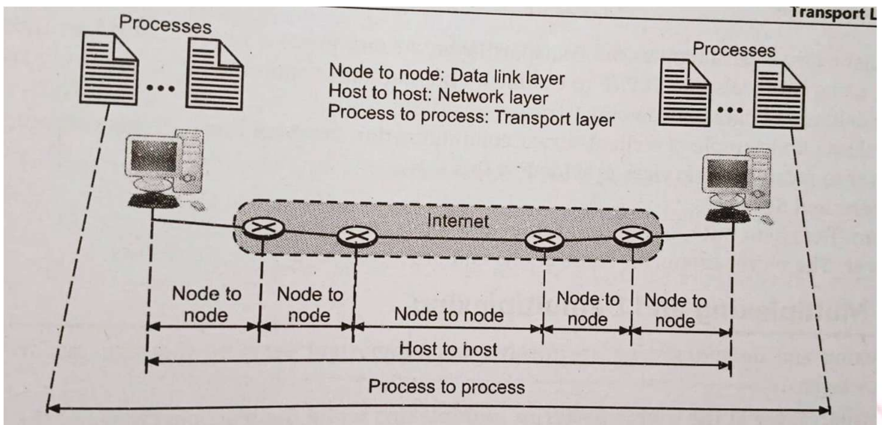


Fig. 3.2: Network Layer versus Transport Layer

- Host to host communication only ensures that message is delivered to the destination.
- This is not enough
- It is necessary to handover this message to the correct process.
- Transport layer will take care of this.

2. Client/Server paradigm (a typical example or pattern of something; a pattern or model)

- Process to process communication can be achieved through several ways
- Most common method is using client server paradigm
- **Client:** defined as process on local host.
- **Server:** Client needs services from another process → which is on another host → called server.
- Both clint and server process have same name
- Important terms related to Client/Server paradigm
  1. Local Host
  2. Local Process
  3. Remote host
  4. Remote Process

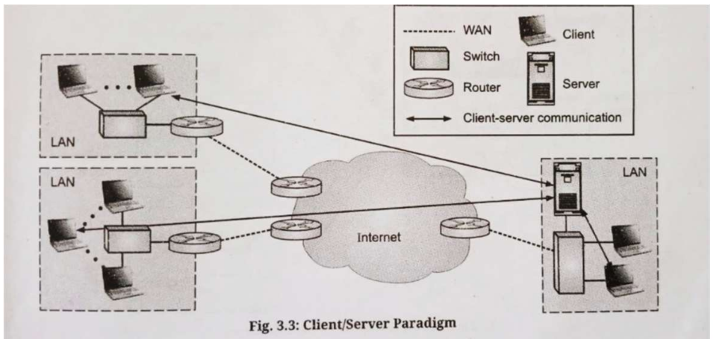


Fig. 3.3: Client/Server Paradigm

- Client and servers connect across a computer network
- Separate hardware or they may share a system
- Client does not share any of its resources
- Server shares resources/services

Roles of client and server

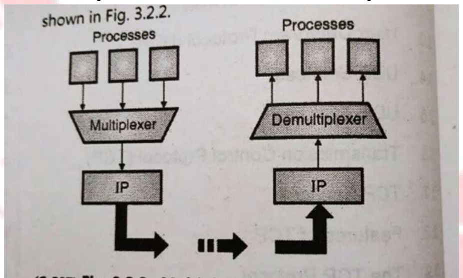
Client: Client initiates the connection and send request to server.

Server: Server listens for the connection and processes requests from client.

3. Multiplexing and Demultiplexing

- Addressing mechanism allows Multiplexing and Demultiplexing
- These are important services provided by transport layer

shown in Fig. 3.2.2.



(G-597) Fig. 3.2.2 : Multiplexing and demultiplexing

Multiplexing:

- ○ At the sending end – several processes are interested in sending packets
- ○ But there is only one transport layer protocol (UDP/TCP)
- ○ It's a many to one relationship

- Protocol first accepts messages from different processes
- These messages are separated from each other by their port number.
- Each process has unique port number
- Transport layer adds header and passes packet to the network layer.

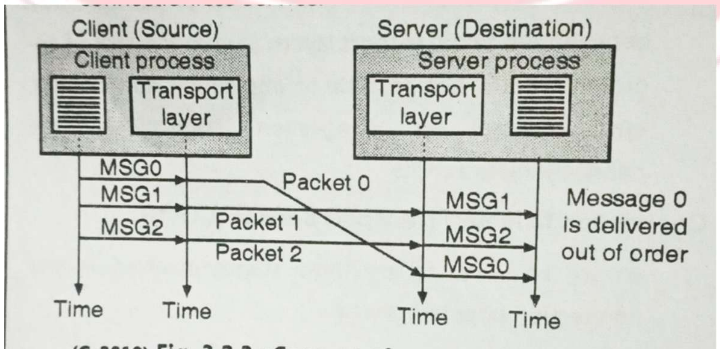
Demultiplexing:

- At receiving end, the relationship is one to many, so we need a demultiplexer.
- Transport layer receives datagram from network layer
- It checks for errors
- Drops the header to obtain the messages
- Delivers them to appropriate process based on the port number

4. Connectionless vs. Connection-Oriented Service

- Transport later protocol provides 2 types of services
  1. **Connectionless Services**
  2. **Connection Oriented Services**
- At network layer connectionless means → different datagrams of same message follow different path
- At transport layer connectionless means → independency between packets
- Connection oriented means → packets are interdependent

1. Connectionless Services



(G-2018) Fig. 3.2.3 : Concept of connectionless service

- The source process divides message into chunks of data
- The size of chunks is acceptable by transport layer
- Chunks are delivered to the transport layer one by one
- They are treated as independent units
- Every chunk is encapsulated in a packet by the transport layer
- Then it is sent to the transport layer of destination
  - **Out of order delivery**
    - Suppose there are three chunks of independent messages – 0,1,2
    - As they are independent of each other → they can arrive out of order at the destination
    - Packet 0 travels longer path and undergo extra delay
    - So, chunks are delivered out of order (1,2,0)
    - If these chunks are of same messages → due to their out of order delivery the server will receive a strange message.
  - **One packet is lost**
    - UDP packets are not numbered
    - If one of the packets is lost – receiving transport layer will not have any idea about lost packet
    - It will simply deliver received chunks to server process
    - The problem arises due to lack of coordination
    - It leads to difficulty in flow control, error control, or congestion control.

2. **Connection oriented transport services**

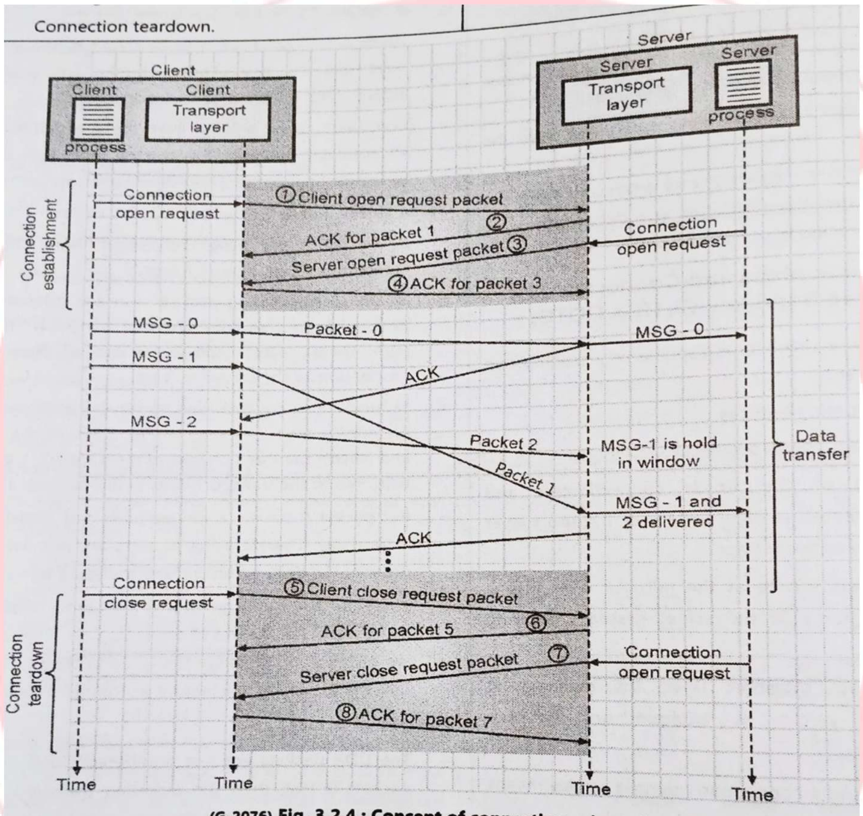
   There are 3 stages in Connection oriented services
   1. Connection establishment
   2. Exchange of data
   3. Connection tear down

Connection oriented services present at network layer as well but it is different from transport layer.

Here it coordinates between hosts on either side and all routers between them.

At transport layer it is end to end service that involve only 2 hosts

It is possible to implement flow control, error control, and congestion control here.



(G-2076) **Fig. 3.2.4 : Concept of connection oriented service**

Q: Compare CLTS and COTS

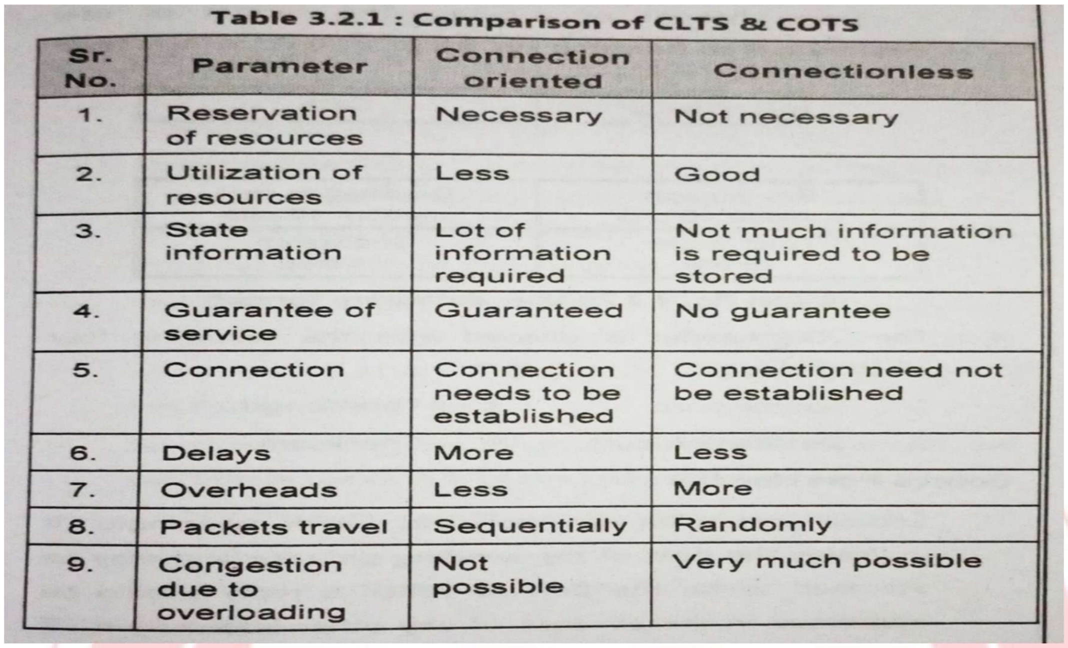


Table 3.2.1 : Comparison of CLTS & COTS

Sr. No.	Parameter	Connection oriented	Connectionless
1.	Reservation of resources	Necessary	Not necessary
2.	Utilization of resources	Less	Good
3.	State information	Lot of information required	Not much information is required to be stored
4.	Guarantee of service	Guaranteed	No guarantee
5.	Connection	Connection needs to be established	Connection need not be established
6.	Delays	More	Less
7.	Overheads	Less	More
8.	Packets travel	Sequentially	Randomly
9.	Congestion due to overloading	Not possible	Very much possible

Q: What is UDP? Enlist the services provided by UDP?

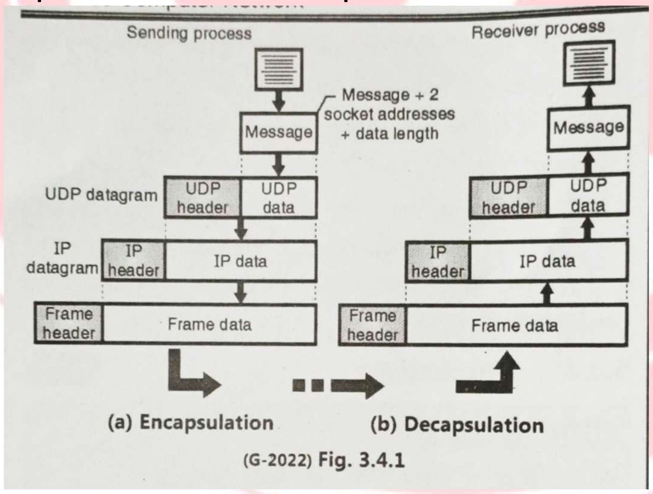
- **UDP- User Datagram Protocol**
- It is a transport layer protocol
- Belongs to TCP/IP protocol suite
- Serves as intermediary between application programs and network operations
- Designed by David P. Reed in 1980
- Defined in RFC 768
- UDP is connectionless protocol
- UDP is suitable for streaming applications
- **UDP Services:**
  - Process to process communication
  - Connectionless services
  - Flow control
  - Error control
  - Checksum
  - Congestion control
  - Encapsulation and decapsulation

- Queuing
- Multiplexing and demultiplexing

Q: Explain any Two services offered by UDP

- Process to process communication:
    - UDP provides process to process co by using sockets → combination of IP address and port numbers.
    - (List of Ports is given in further question)
- Connectionless service
    - UDP is connectionless, hence unreliable.
    - Each user datagram sent by UDP is an independent datagram
    - Different user datagrams sent by UDP have no relationship between them →even though they are origination from same process and sent to same destination
    - User datagram does not have any number
    - Ther is no connection establishment and no connection termination
    - Each datagram is free to travel in path
    - Processes sending short messages can successfully use UDP
    - Messages less than 65507 bytes can use UDP
- Flow and error control
    - It does not provide flow control hence receiver can overflow with incoming messages.
    - It also does not support error control mechanism except for the checksum
    - No acknowledge is sent from destination to the sender so sender does not know whether message has reached or lost or duplicated
    - If receiver detects any error using checksum → the datagram is discarded.

- Checksum
  - Checksum is provided for data integrity
  - It is different from Checksum calculation IP
  - Checksum is calculated by considering following 3 sections
    1. Pseudo header
    2. UDP header
    3. Data coming from application layer
  - Checksum is optional in UDP
  - Sender can make a decision of not calculating the checksum
  - If so → field is filled with zeros before sending packet
- Congestion control
  - UDP does not provide Congestion control
  - It assumes the packets being small will not crest any congestion
  - The assumption may not be true → in case of real time transfer of audio and video
- Encapsulation and decapsulation

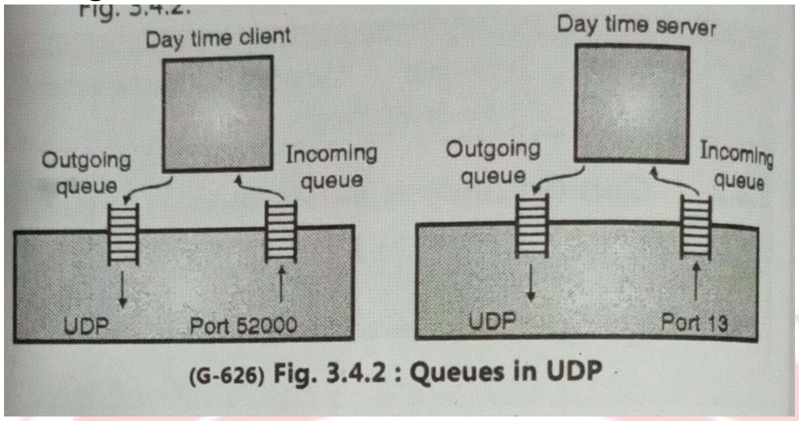


-
  - UDP encapsulates and decapsulates messages→ to exchange message between two communication processes
  - Encapsulation:
    - ➢ Message produced by a process is to be sent by UDP

- Process passes→ the message and 2 socket addresses with the length of data to UDP
- UDP receives the data and adds the UDP header to it
- "This is called UDP datagram which is passes to IP with the socker address"
- IP adds its own header to UDP datagram
- It enters value 17 into the protocol field→ indicating UDP is being used
- IP datagram is then passed to data link layer
- DLL adds its own header and trailer to create frame and sends it to physical later
- Finally physical layer converts these bits into electrical/optical signals and sends to the destination

- Decapsulation:
  - Encoded message arrives at destination physical layer
  - Optical/electrical signal is decoded into bits and passes them to DLL
  - DLL checks the data using header and trailer
  - Header and trailer are discarded if no errors
  - Datagram is passed to IP
  - IP carries out checking of errors
  - If no errors datagram is passed to UDP after dropping IP header
  - Datagram from IP to UDP also contains sender and receiver IP address
  - Entire datagram is checked using checksum
  - If no errors UDP header is dropped
  - Application data + senders' sockets are handed over to the process
  - Process uses socket address to respond that message is received

- Queuing



(G-626) Fig. 3.4.2 : Queues in UDP

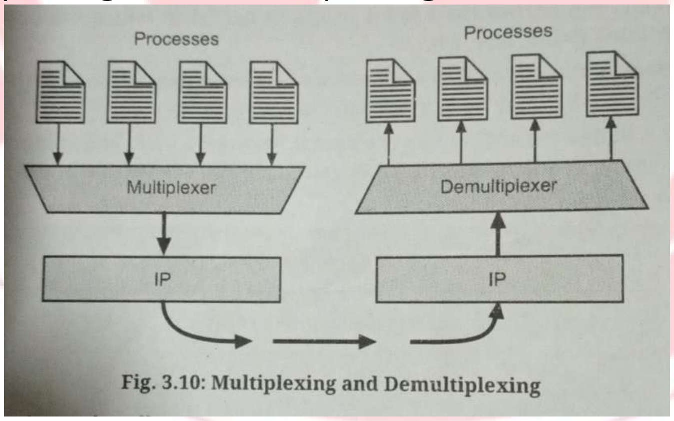
-
    - The process starts at the **client** site by requesting a port number from OS
    - Every process gets one port number→ hence can create one outgoing and incoming queue
    - Queues function only when process is running
    - They are destroyed as soon as process is terminated
    - **Server** creates incoming and outgoing queues using its well-known ports
    - Queues exist as long as server is running
- Multiplexing and demultiplexing
    - 

      Fig. 3.10: Multiplexing and Demultiplexing
    - Multiplexing: at the sender's side -several processes need to send user datagram
    - There is only one UDP
    - Many-to-one relationship
    - Hence requires multiplexing
    - Demultiplexing: at receiver's side there is only one UDP

- Many processes can receive user datagram
- One-to-many relationship
- Hence requires demultiplexing

Q: Enlist Well known ports with UDP.

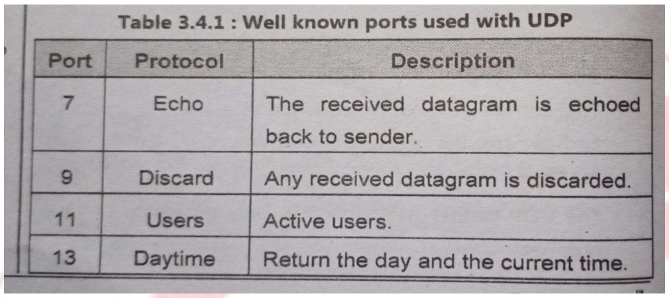
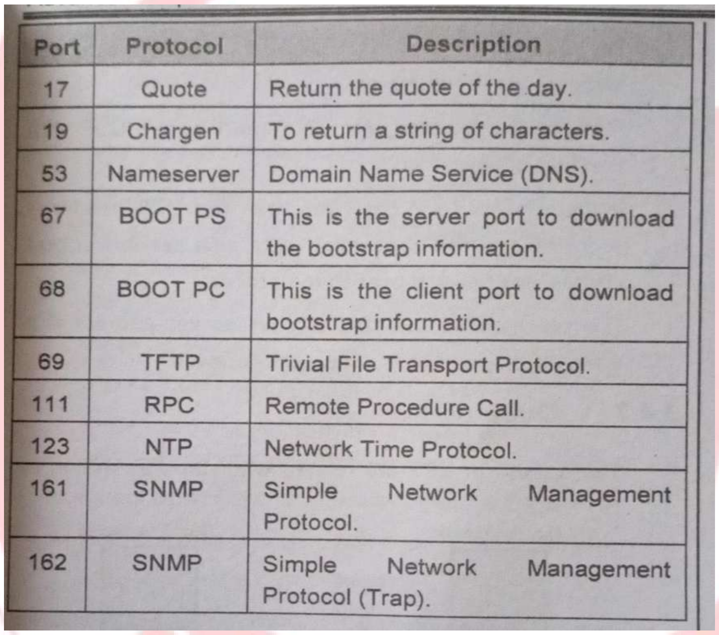


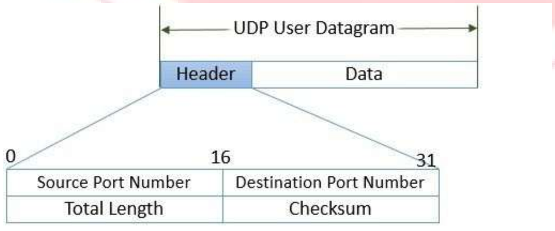
Table 3.4.1 : Well known ports used with UDP

Port	Protocol	Description
7	Echo	The received datagram is echoed back to sender.
9	Discard	Any received datagram is discarded.
11	Users	Active users.
13	Daytime	Return the day and the current time.



Port	Protocol	Description
17	Quote	Return the quote of the day.
19	Chargen	To return a string of characters.
53	Nameserver	Domain Name Service (DNS).
67	BOOT PS	This is the server port to download the bootstrap information.
68	BOOT PC	This is the client port to download bootstrap information.
69	TFTP	Trivial File Transport Protocol.
111	RPC	Remote Procedure Call.
123	NTP	Network Time Protocol.
161	SNMP	Simple Network Management Protocol.
162	SNMP	Simple Network Management Protocol (Trap).

Q: Explain UDP header format/ user datagram format with diagram



User Datagram Header Format

- **UDP** packets are called user datagram
- Fixed header size→8 bytes
- Header + Data = 65,535 bytes
- 4 fields in header format

1. **Source Port Number**

- Optional field
- Indicates port of sending process
- 16 bits field
- Port number can range from 0 to 65,535
- If source host is a client→it is ephemeral port number(temporary)→ requested by the process and chosen by UDP
- If the source host is a server→well known port number is used

2. **Destination Port Number**

- 16 bits filed
- Used by the process running on the destination
- If destination is a server→ well known port is used
- If destination is a client→ephemeral port is used

3. **Total Length**

- 16 bits field
- Defines total length of user datagram
- Header + Data
- 0 to 65,535
- This field is actually not necessary
- Because UDP datagram is encapsulated in IP datagram and IP datagram has its own length field
- UDP length = IP length – IP header length
- While delivering UDP datagram to UDP layer→ IP software drops the IP header

4. **Checksum**

- UDP provides checksum for data integrity

- Checksum includes 3 sections→ pseudo header, UDP header, data

Q: Explain UDP Pseudo Header Format

- Pseudo header is part of header of IP packet
- Even if pseudo header is not included user datagram may arrive safe and sound
- If IP header is corrupted – it may be delivered to wrong host

Q: What are advantages of UDP over TCP?

- UDP has minimum overheads
- It can be easily used if the sending process is not too bothered about reliability
- It reduces interaction between sender and receiver

Q: Enlist and explain features of UDP

- **Connectionless Service:** independent from other packet sent by same application
- **Lack of Congestion Control:** does not provide congestion control→ does not create additional traffic in error prone zone
- **Lack of Error Control:** does not provide error control hence unreliable
- **Transaction-Oriented:** so suitable for query response protocols
- **Datagram:** UDP datagram is suitable for modelling other protocols
- **Simple:**
- **Stateless:** suitable for very large number of clients
- **Lack of Retransmission Delays:** makes it suitable for real time applications- gaming, VOIP, streaming
- **Support Multicast:** supports multicasting, suitable for broadcasting
- **Faster in Data Transfer:** light weight protocol→faster and simple data transmission

- **Queuing:** simple and suitable for query-based communication→queues are associated with the ports
- **Low Overhead:** data is transferred with minimal overhead
- **Port:** UDP contains source port number and destination port number
- **No Acknowledgement/ Not Reliable:** so not reliable as TCP→does not guarantee ordered delivery of data

Q: Enlist different applications of UDP

- It is used for simple request response communication → when size of data is less and hence there is less concern of flow and error control
- It is suitable for multicasting
- It is used for routing update protocols like RIP
- Normally used for real time applications
- Uses of 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, OSPF
- Uses of UDP as an application layer protocol:
    - Trace Route
    - Record Route
    - Time Stamp

Q: What is Transmission Control Protocol (TCP)?

- It's a reliable connection-oriented protocol
- Connection is established between sender and receiver before data is transmitted
- It divides the data received from upper layer into segments and tags the sequence number to each segment→which are used at receiving end for reordering the data

- TCP relies on application layer and network layer
- It is a transport layer protocol

Q: Enlist the TCP services?

- Following are the services provided by TCP
    - Stream delivery service
    - Sending and receiving buffers
    - Bytes and segments
    - Full duplex service
    - Connection oriented service
    - Reliable service
    - Process to process communication

Q: Explain TCP services in details.

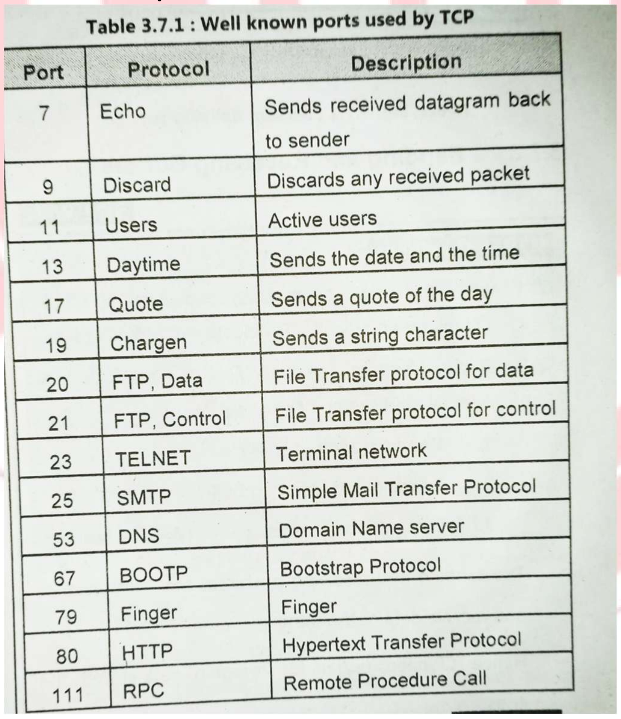
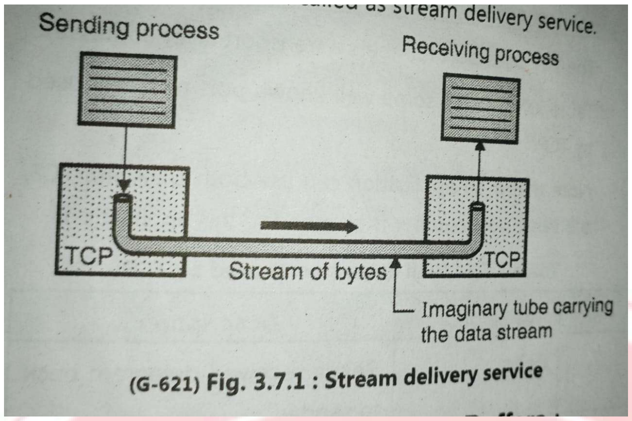
1. **Process to Process Communication**
    - TCP uses port numbers- similar to UDP
    - 

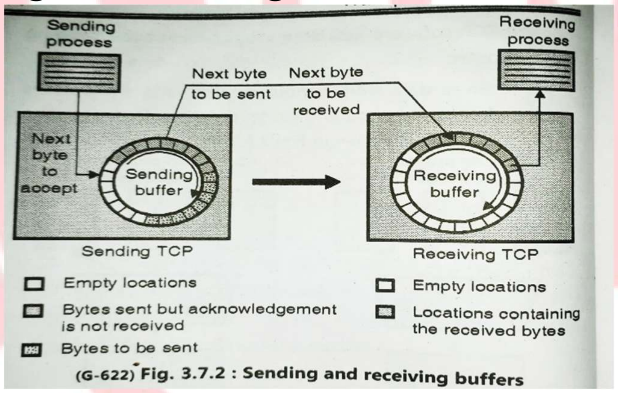
Table 3.7.1 : Well known ports used by TCP

Port	Protocol	Description
7	Echo	Sends received datagram back to sender
9	Discard	Discards any received packet
11	Users	Active users
13	Daytime	Sends the date and the time
17	Quote	Sends a quote of the day
19	Chargen	Sends a string character
20	FTP, Data	File Transfer protocol for data
21	FTP, Control	File Transfer protocol for control
23	TELNET	Terminal network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

2. **Stream Oriented/ Stream Delivery Service**

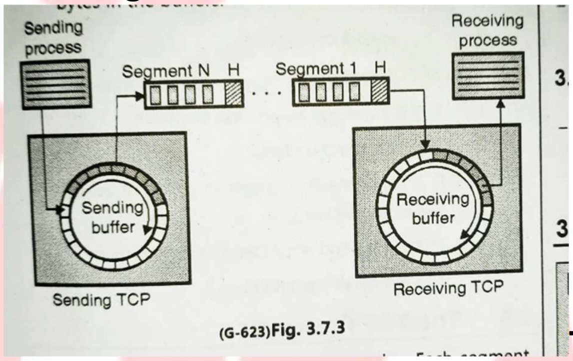
- 
...called as stream delivery service.
Sending process
Receiving process
TCP
TCP
Stream of bytes
Imaginary tube carrying the data stream
(G-621) Fig. 3.7.1 : Stream delivery service
- TCP is a stream-oriented protocol
- A process delivers/receives the data in the form of stream of bytes.
- Sending and Receiving processes seams to be connected by imaginary tubes
- This is called as stream delivery services.

3. Sending and Receiving Buffers

- 

(G-622) Fig. 3.7.2 : Sending and receiving buffers
- Sending and Receiving process may not produce or receive data at the same speed hence TCP needs buffer at both ends
- Two types of buffers used → sending buffer, & receiving buffer
- Figure shows direction of movement of data
- Sending buffer has three types of locations
- Receiving buffer has two types of locations

4. Bytes and Segments

- Only buffering is not enough → we need one more step before sending data
- TCP groups a number of bytes to form a packet called a segment
- Header is added to each segment
- Segment is inserted into IP datagram and transmitted.
- Segments may be received out of order or lost or corrupted
- All segments are not of same size
- Each segment carries hundreds of bytes
- 

(G-623)Fig. 3.7.3
T

5. Full Duplex Service

- TCP offers Full Duplex Service → data can travel in both directions simultaneously.
- Each TCP has sending and receiving buffer hence full duplex service possible.

6. Connection Oriented Service

- TCP is a connection-oriented protocol
- When process 1 wants to communicate with process 2
- Sequence of operations is as follows:
  1. TCP of process 1 informs TCP of process 2 and create a connection between them
  2. Both TCP exchange data in both the directions
  3. After completing the exchange both the buffers are empty – TCP destroys buffer to terminate the connection.
- The connection is virtual
- Datagram is encapsulated

- If segment get lost or corrupted – have to be resend
- Each segment may take different path to reach destination

7. **Reliable Service**
    - TCP is a reliable protocol
    - It uses checksum for error detection
    - Attempts to recover lost or corrupted packets by retransmission, acknowledge policy and timers
    - To ensure reliability it uses- byte number, sequence number, acknowledge number
    - It uses congestion control mechanism
8. **Multiplexing and Demultiplexing Service**
    - TCP does multiplexing and demultiplexing at sender and receiver ends respectively as a number of logical connections can be established between port numbers over a physical connection

Q: Enlist and explain TCP features.

1. **Numbering System**

- TCP keeps track of segments being transmitted or received
- In segment header there is no field for segment number value
- Sequence number and acknowledge number are used
- They correspond to byte number→ not the segment number

a. **Byte number assigned to data bytes to be transferred**
    - TCP gives numbers to the data bytes
    - Byte number is used for flow control and error control
    - Numbering starts with randomly generated numbers
    - Numbering is independent of direction
    - TCP stores the received data bytes in sending buffer and numbers them

b. **Sequence number to segment**
    - TCP assigns sequence number to each segment transmitted.

- Sequence number is the first byte number carrie in that segment.
- Segment number is assigned when:
  - Segment carries both data and control information (piggy backing)
  - Segment without data, no sequence number.

c. Acknowledgement number to received segments

- It indicates number of bytes – expected by the receiver
- Acknowledgement number is cumulative
- Confirmation of received data bytes

2. Flow Control

- TCP provides flow control
- The receiver of the data controls the amount of data that are to be sent by the sender.
- This is done to prevent the receiver from being overwhelmed with data.
- The numbering system allows TCP to use a byte-oriented flow control.

3. Error Control

- TCP implements error control mechanism for reliable data transfer
- Segments are checked for error detection
- Error control is byte oriented
- It includes detecting:
  - Corrupted segment and lost segments
  - Out-Of-Order segment
  - Duplicated Segments

4. Congestion Control

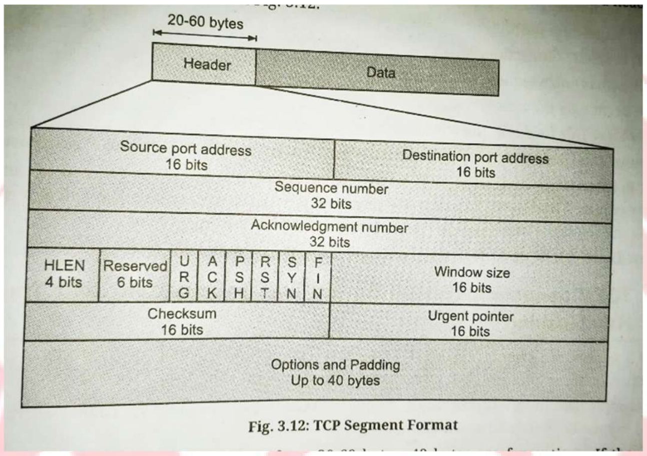
- It takes into account the congestion in the network
- Amount of data sent by sender depends upon following factors→ flow control, level of congestion

5. Interoperability

- TCP has become industry standard

- Supports interoperability across networks
- TCP framework is used to develop complete range of computer communication standard

Q: Draw and explain TCP Segment structure / TCP header format


20-60 bytes
Header
Data
Source port address 16 bits
Destination port address 16 bits
Sequence number 32 bits
Acknowledgment number 32 bits
HLEN 4 bits
Reserved 6 bits
URG
ACK
PSH
RST
SYN
FIN
Window size 16 bits
Checksum 16 bits
Urgent pointer 16 bits
Options and Padding Up to 40 bytes
Fig. 3.12: TCP Segment Format

1. **Source port address**
    - 16-bit field
    - Identifies the application that is sending the data segment
    - 3 Ranges of port
    - 0 to 1023 → well known ports
    - 1024 to 49151 → registered ports
    - 49152 to 65535 → private ports
    - Ports are used by TCP as an interface to application layer
2. **Destination port address**
    - 16 bit field
    - Identifies destination→ receiving hosts
    - Destination port uses the same port number
    - Used to reassemble the message at the receiving end
3. **Sequence Number**
    - 32 bit field
    - Holds sequence number
      i.e. byte number of first byte sent in that segment
4. **Acknowledgement number**
    - 32 bit field

- Identified next data byte that sender expects from receiver
- It is an acknowledgment for previous byte being received successfully.

5. **Header length (HLEN)**
   - 4 bit fiend
   - Specifies length of TCP header in 32 bit words(4 byte words)
   - Minimum length - If header is of 20 bytes - field will hold 5 (5 X 4 = 20 bytes)
   - Maximum length - If header is of 60 bytes - field will hold 15 (15 X 4 = 60 bytes)
   - This field value is always between 5 to 15
6. **Reserved**
   - Reserved for future use
7. **Control Flags**

   6 Flags that control connection establishment, termination flow control, mode of transfer etc. There are 6,1-bits
   - **URG (Urgent Pointer)**
   - Urgent pointer is valid
   - **ACK(Acknowledgement)**
   - Acknowledge number is valid
   - **PSH (Push function)**
   - Request for push (ex. Break request which can jump ahead the queued data
   - **RST (Re set the connection)**
   - Reset the connection
   - **SYN1(Synchronise)**
   - Synchronise sequence number
   - **FIN (NO more data from sender)**
   - Terminate the connection
8. **Window size**
   - 16-bit field

- Tells the sender how much data the receiver is willing to accept
- Maximum window size 65535

9. Checksum

- 16 bit value
- Sender calculates the value based on the content of TCP header and data field
- Receiver generates the same computation
- If values match- receiver is confident that segment is arrived intact

10. Urgent pointer

- It is 16 bit field
- It is used to point to a data that should be processed urgently
- It tells receiver when the last byte of urgent data in the segment ends

11. Options

- Provides additional functionality
- Variable length field
- Can not be larger than 40 bytes
- Most common option is MSS – maximum segment size

12. Padding

- As options may vary in size – it is necessary to pad TCP header with 0's to maintain standard size

13. Data

- Variable length field 'Carries application data from sender to receiver
- TCP header + data field – TCP segment

Q: Explain TCP connection establishment using Three-way handshaking mechanism

- 3 phases in connection-oriented TCP transmission are:
  - ➢ **Connection establishment**

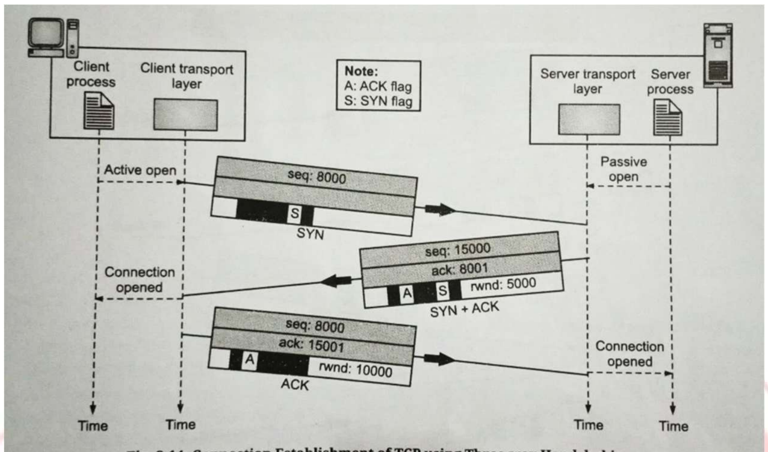


Fig. 3.14: Connection Establishment of TCP using Three-way Handshaking

- To establish a connection TCP uses 3-way handshaking
- Consider an application program known as client wants to make connection with another application program known as server using TCP.
- Process starts with server
- Server tells TCP that it is ready to accept connection.
- This request called as passive open
- Although TCP server is ready to accept connection from any machine in the world → it can not make the connection itself.
- Client program initiates a request for an active open
- Client that wishes to connect to an open server tells its TCP to connect to a particular server.
- TCP can now start the 3-way handshaking process.
- To establish a connection 3-way handshake occurs

  1. **SYN segment**
     - Only SYN flag is set
     - SYN Segment is for synchronization of sequence number
     - Client chooses random number as first sequence number and sends it to server
     - Sequence number is called as ISN-Initial Sequence Number
     - SYN is a controlled segment – carries no data
     - It consumes one sequence number

- When data transfer starts ISN is incremented by one
- It contains no real data but imaginary bytes

2. **SYN +ACK segment**
    - It sets 2 flag bits – SYN and ACK
    - This segment cannot carry data
    - But it consumes one sequence number
    - Segment has dual purpose
    - 1st – SYN segment for communication in other direction→ server uses this segment to initialise sequence number
    - Server also acknowledge receipt of SYN from the client by setting ACK flag
    - It also needs to define receive window size
3. **ACK segment**
    - ACK segment acknowledges receipt of second segment with ACK flag and acknowledges field
    - It carries no data
    - Consumes no sequence number

➢ **Data Transfer**

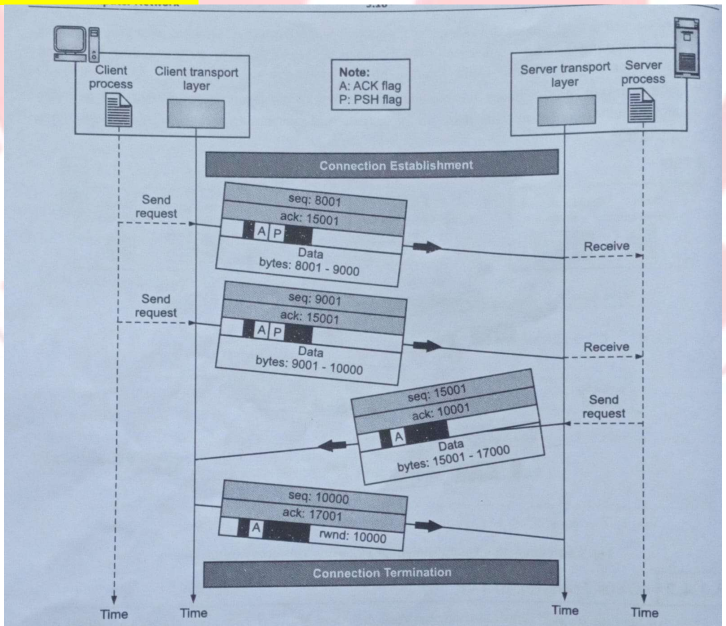


Fig. 3.15: Data Transfer in TCP

- After connection is established, bidirectional data transfer can take place.
- The client and server can both send data and acknowledgments.
- the client sends 2000 bytes of data in two segments.
- The server then sends 2000 bytes in one segment.
- The client sends one more segment.
- The first three segments carry both data and acknowledgment,
- But the last segment carries only an acknowledgment because there are no more data to be sent.
- The data segments sent by the client have the PSH (push) flag set
- so server TCP knows to deliver data to the server process as soon as they are received.
- The segment from the server→ does not set the push flag.
- Most TCP implementations have the option to set or not set this flag.

  - **Connection termination**

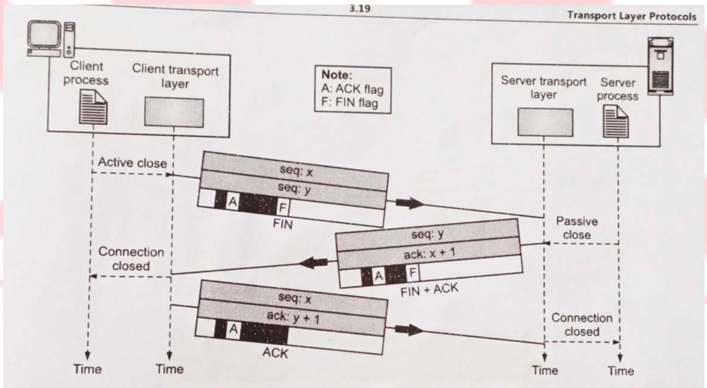


Fig. 3.16: three-way Handshaking for Connection Termination for TCP

- Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.

- Most implementations today allow two options for connection termination→ three-way handshaking and four-way handshaking with a half-close option.
    - **3 -way handshaking**
- **FIN segment:**
- The client TCP, after receiving a close command from the
- client process, sends the first segment
- A FIN segment in which the FIN flag is set.
- It can include the last chunk of data sent by the client, or it can be just a control segment
- if it is only a control segment or number ifit does not carry data - it consumes only one sequence number.
- **FIN + ACK segment:**
- Server TCP- after receiving the FIN segment- informs its process of the situation and sends the second segment→ a FIN +ACK segment, to confirm the receipt of the FIN segment from the client
- At the same time to announce the closing of the connection in the other direction.
- This segment can also contain the last chunk of data from the server.
- The FIN +ACK segment consumes one sequence
- number if it does not carry data.
- **ACK segment:**
- client TCP sends the last segment→an ACK segment→ to confirm the receipt of the FIN segment from the TCP server.
- This segment contains the acknowledgment number
- which is 1 plus the sequence number received in the FIN segment from the server.
- This segment cannot carry data and consumes no sequence numbers.
    - **4-way handshaking in brief.**
- Four-way handshaking in TCP uses half-close option
- one end can stop sending data while still receiving data.

- Either end can issue a half-close
- It is normally initiated by the client.
- It can occur when the server needs all the data before processing can begin.
- A good example is sorting→When the client sends data to the server to be sorted→ server needs to receive all the data before sorting can start
- after sending
- all the data, can close the connection in the client-to-server direction.
- However, server-to-client direction must remain open to receive the sorted data.
- server, after receiving the data- needs time for sorting
- Its outbound direction must remain open.

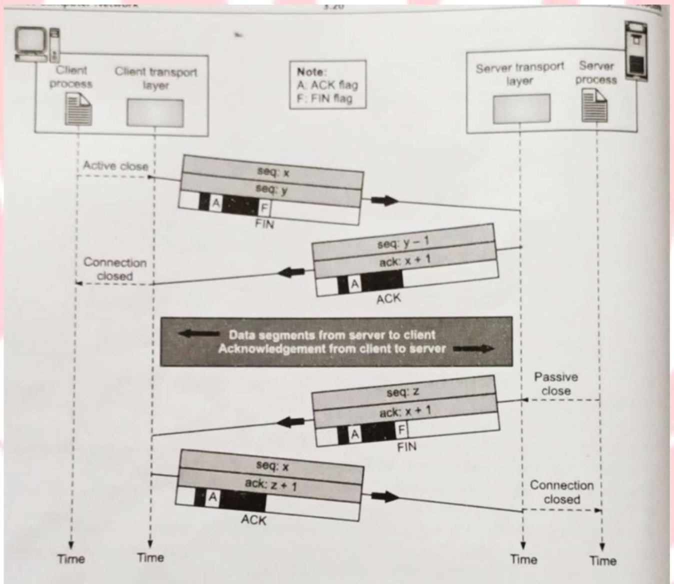


Fig. 3.17: Half Close in TCP

- Figure shows an example of a half-close.
- client half-closes the connection by sending a FIN segment.
- server accepts the half-close by sending ACK segment.
- The data transfer from the client to the server stops.
- Server can still send data.

- When the server has sent all the processed data, it sends a FIN segment→acknowledged by an ACK from the client.
- After half-closing of the connection→data can travel from the server to the client
- Acknowledgments can travel from the client to the server.
- Client cannot send any more data to the server.
- second segment (ACK) consumes no sequence number

Q: what is Flow control and Error control in TCP?

- **Flow control:**

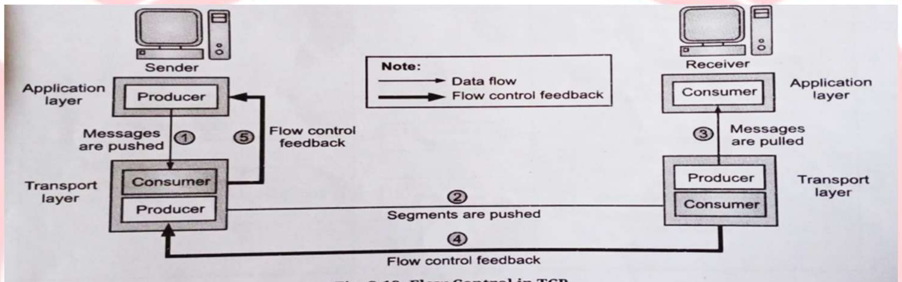


Fig. 3.19: Flow Control in TCP

- A technique used for controlling data rate of the sender – so that receiver is not overwhelmed
- TCP keeps flow control separate from error control
- Data flow→ sending process tp sending TCP→receiving TCP to receiving process
- Data flow→ sender to receiver
- Control feedback travels→from receiver to sender
- **Error Control**

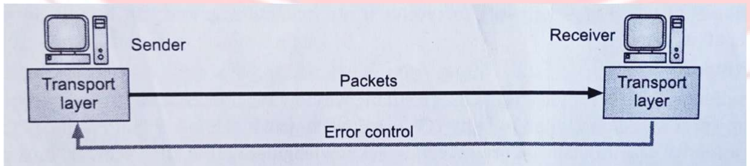


Fig. 3.21: Error Control at Transport Layer

- **Error control includes:**
- Mechanism of detecting and resending corrupted segments, missing segments, duplicated segments, out of order segments

- Achieved through 3 tools: **checksum, acknowledgement, time-out**
- Error control at transport level is responsible for→
  - Detect and discard corrupted packets
  - Keep track of lost and discarded packets and resend them

Q: Explain Congestion control in TCP.

- To prevent network load by adjusting sender's transmission rate base on n/w conditions – ensuring efficient and reliable data delivery
- Need of congestion control:

  - **End-to-End Responsibility** The transport layer (e.g., TCP) manages data flow between sender and receiver. It has the best visibility into how much data is being sent and received, and how the network is responding.
  - **Avoiding Congestion Collapse** Without control, multiple senders may overwhelm the network, causing packet loss, retransmissions, and delays—ultimately leading to *congestion collapse*, where throughput drops drastically.
  - **Fair Resource Allocation** Congestion control ensures that all users get a fair share of bandwidth, preventing aggressive senders from monopolizing the network.
  - **Dynamic Adaptation** The transport layer can adjust transmission rates based on feedback (like packet loss or delay), making it ideal for reacting to changing network conditions.
  - **Reliable Communication** Protocols like TCP use congestion control to maintain reliability. If congestion is detected, TCP slows down to avoid further packet loss and ensures data arrives intact.

Open loop

- Prevention-based: Tries to avoid congestion before it happens
- No feedback used
- Relies on fixed policies like limiting retransmissions or controlling admission

Closed loop

- Reaction-based: Responds to congestion after it occurs
- Uses feedback from the network (e.g., packet loss, delay)
- Adjusts transmission rate dynamically to reduce congestion

Benefits of Transport-Layer congestion control:

Data Reliability

Transport layer congestion control (like in TCP) ensures that data is delivered **accurately and in order**, even during network congestion. It detects lost packets and **retransmits them**, maintaining reliable communication between sender and receiver.

Improved Network Utilization

By adjusting the data transmission rate based on network feedback, congestion control **prevents overload** and keeps the network running smoothly. This leads to **efficient use of bandwidth**, reduced delays, and better overall performance.

Q: Compare UDP and TCP

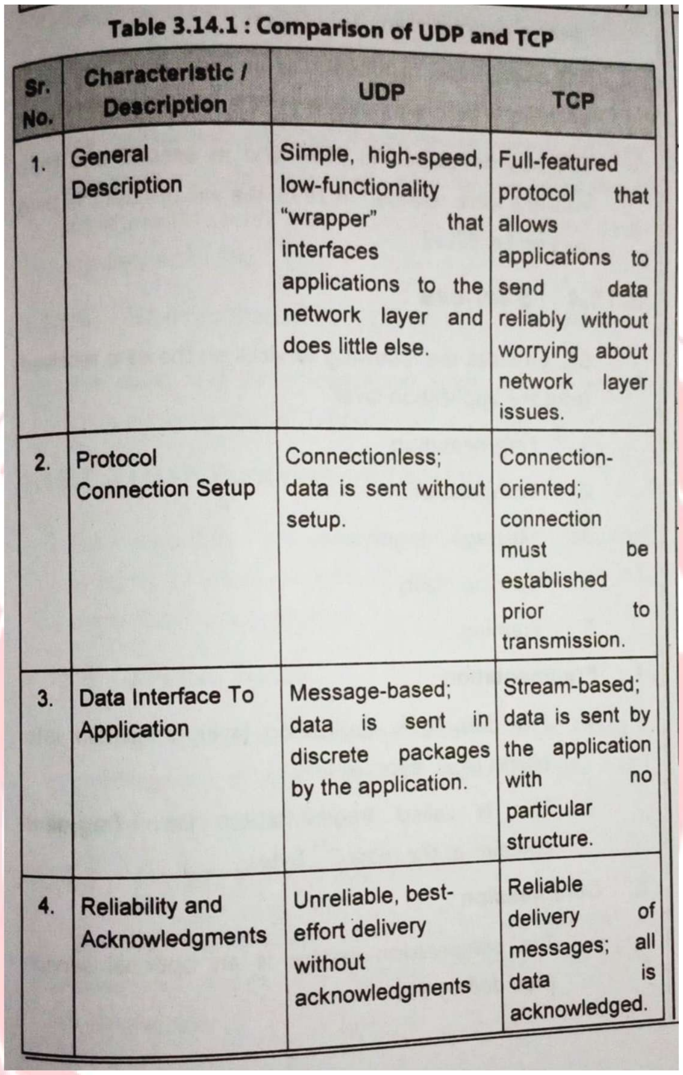
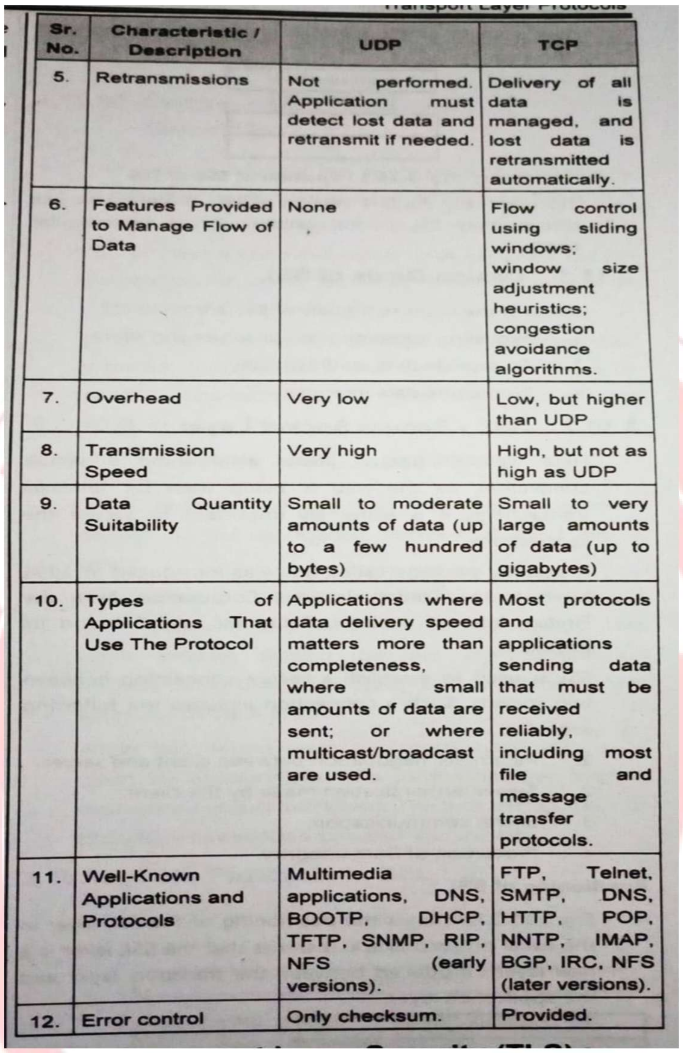
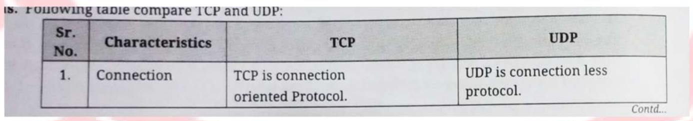


Table 3.14.1 : Comparison of UDP and TCP

Sr. No.	Characteristic / Description	UDP	TCP
1.	General Description	Simple, high-speed, low-functionality "wrapper" that interfaces applications to the network layer and does little else.	Full-featured protocol that allows applications to send data reliably without worrying about network layer issues.
2.	Protocol Connection Setup	Connectionless; data is sent without setup.	Connection-oriented; connection must be established prior to transmission.
3.	Data Interface To Application	Message-based; data is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure.
4.	Reliability and Acknowledgments	Unreliable, best-effort delivery without acknowledgments	Reliable delivery of messages; all data is acknowledged.



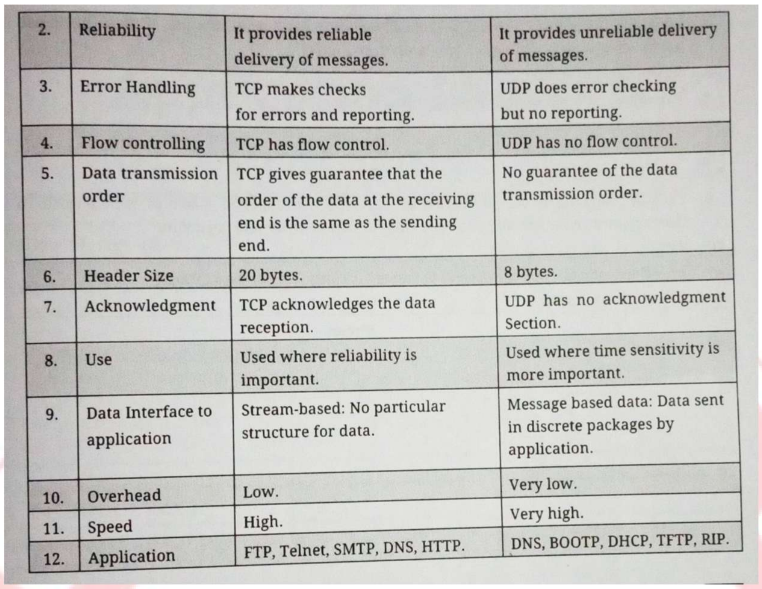
Sr. No.	Characteristic / Description	UDP	TCP
5.	Retransmissions	Not performed. Application must detect lost data and retransmit if needed.	Delivery of all data is managed, and lost data is retransmitted automatically.
6.	Features Provided to Manage Flow of Data	None	Flow control using sliding windows; window size adjustment heuristics; congestion avoidance algorithms.
7.	Overhead	Very low	Low, but higher than UDP
8.	Transmission Speed	Very high	High, but not as high as UDP
9.	Data Quantity Suitability	Small to moderate amounts of data (up to a few hundred bytes)	Small to very large amounts of data (up to gigabytes)
10.	Types of Applications That Use The Protocol	Applications where data delivery speed matters more than completeness, where small amounts of data are sent; or where multicast/broadcast are used.	Most protocols and applications sending data that must be received reliably, including most file and message transfer protocols.
11.	Well-Known Applications and Protocols	Multimedia applications, DNS, BOOTP, DHCP, TFTP, SNMP, RIP, NFS (early versions).	FTP, Telnet, SMTP, DNS, HTTP, POP, NNTP, IMAP, BGP, IRC, NFS (later versions).
12.	Error control	Only checksum.	Provided.



Following table compare TCP and UDP:

Sr. No.	Characteristics	TCP	UDP
1.	Connection	TCP is connection oriented Protocol.	UDP is connection less protocol.

Contd...



2.	Reliability	It provides reliable delivery of messages.	It provides unreliable delivery of messages.
3.	Error Handling	TCP makes checks for errors and reporting.	UDP does error checking but no reporting.
4.	Flow controlling	TCP has flow control.	UDP has no flow control.
5.	Data transmission order	TCP gives guarantee that the order of the data at the receiving end is the same as the sending end.	No guarantee of the data transmission order.
6.	Header Size	20 bytes.	8 bytes.
7.	Acknowledgment	TCP acknowledges the data reception.	UDP has no acknowledgment Section.
8.	Use	Used where reliability is important.	Used where time sensitivity is more important.
9.	Data Interface to application	Stream-based: No particular structure for data.	Message based data: Data sent in discrete packages by application.
10.	Overhead	Low.	Very low.
11.	Speed	High.	Very high.
12.	Application	FTP, Telnet, SMTP, DNS, HTTP.	DNS, BOOTP, DHCP, TFTP, RIP.

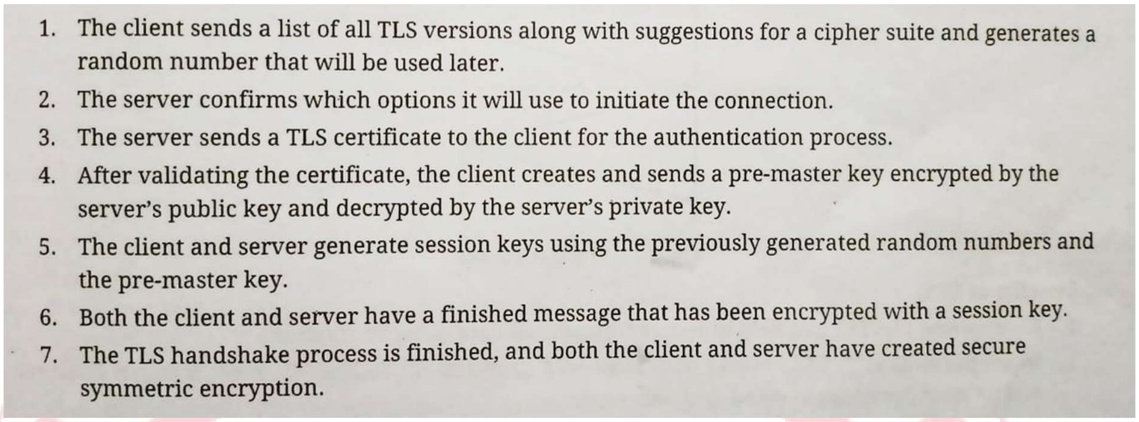
Q: What is TLS (Transport Layer Security)

- widely adopted security protocol
- designed to facilitate privacy and data security
- TLS can encrypt communications over network and various modes – email, messaging VOIP etc
- TLS is derived from SSL (Secure Socket Layer)
- TLS ensures that no third party eavesdrop or tamper with any message
- It is a cryptographic protocol

Q: Describe working of TLS.

- It is a cryptographic protocol provides end to end security
- Used in secure web browsing
- TLS—encrypts the data sent over internet—to ensure that eavesdroppers and hackers are unable to see what we transmit
- Private and sensitive information- passwords, credit card numbers, personal information
- It uses AES (Advanced Encryption Standard)
- It works with 2 security levels
    - TLS record protocol

  - TLS handshake protocol
- These protocols use symmetric and asymmetric cryptography
- **Process**:



1. The client sends a list of all TLS versions along with suggestions for a cipher suite and generates a random number that will be used later.
2. The server confirms which options it will use to initiate the connection.
3. The server sends a TLS certificate to the client for the authentication process.
4. After validating the certificate, the client creates and sends a pre-master key encrypted by the server's public key and decrypted by the server's private key.
5. The client and server generate session keys using the previously generated random numbers and the pre-master key.
6. Both the client and server have a finished message that has been encrypted with a session key.
7. The TLS handshake process is finished, and both the client and server have created secure symmetric encryption.

- TLS Handshake Protocol:

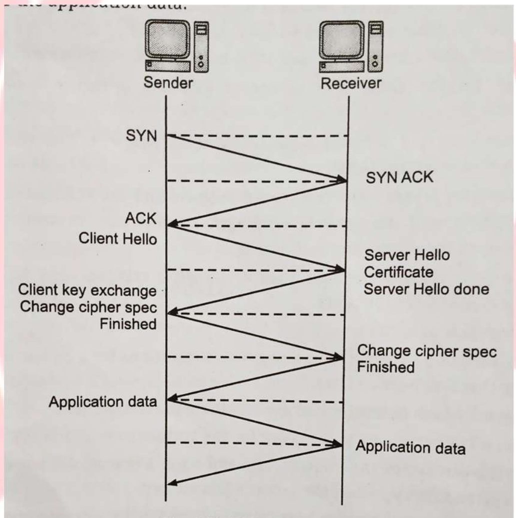


Fig. 3.24: TLS Handshake

- A client sends a synchronous message "client hello" requesting a connection and presents a list of supported cipher suites and a random string of bytes.
- The server responds with a "server hello" message containing a server certificate.
- The server is sending its SSL certificate to the client for the purpose of authentication. The client then authenticates the server by verifying the server's SSL certificate, and also sends a certificate for authentication if requested by the server.
- The client sends the client key exchange, change Cipher specification finished message to the server.

- The server decrypts the message sent by client secret with the private key.
- Both client and server generate session keys from the client random, the server random, and the secret message.
- The client sends a "finished" message that has been encrypted with a session key.
- The server responds with a finished message which was encrypted with a session key.
- The client and server have successfully achieved secure symmetric encryption, meaning the handshake is complete and communication can continue with the established session keys.
- Finally transfer the application data
-

Q: What are the benefits of TLS?

1. **Encryption:** TLS/SSL can help to secure transmitted data using encryption
2. **Interoperability:** TLS/SSL work with most of the web browsers and most of the OS and webservers
3. **Algorithm Flexibility:** provides operations for authentication mechanism, encryption algorithm and hashing algorithm that are used during the secure session
4. **Ease of Deployment:** many applications TLS/SSL temporarily on a windows server 2003 OS
5. **Ease of Use:** most of the operations are completely invisible to client
6. **Data Privacy:** TLS encrypts data, preventing eavesdropping and enduring that sensitive information remains the same
7. **Data Integrity:** ensures data is not tampered during transit ensuring the data received is same as data sent
8. **Authentication:** allows validation of identities of both client and server→preventing man in the middle attack
9. **Trust:** TLS helps building trust between users and websites

Q: Enlist Applications of TLS

1. **Secure web browsing**
   - It is the foundation of secure web browsing

- Websites use HTTPS-Hyper Text Transfer Protocol Secure to encrypt communication

2. **Email**
   - TLS is used to encrypt email communication- protecting the privacy of email
3. **VoIP**
   - TTL can secure voice and video communication over the internet
4. **File Transfer**
   - TLS can secure file transfer→ ensuring that data is not intercepted during transmission
5. **VPNs(Virtual Private Networks)**
   - TLS can secure VPN- protecting data from eavesdropping
6. **Other applications**
   - Secure communication over any network

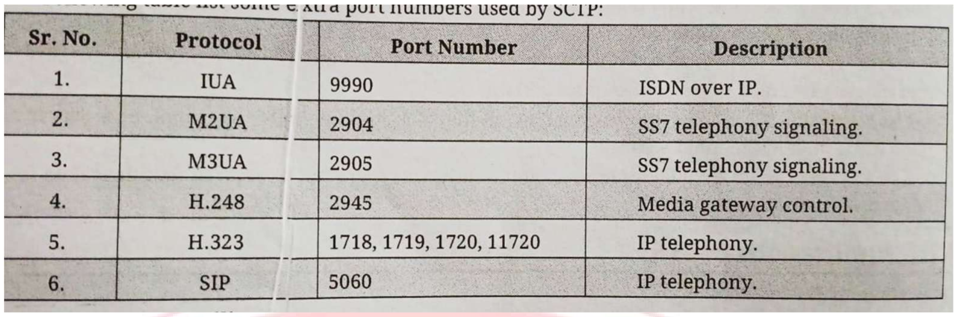
Q: What is SCTP and enlist SCTP services

Stream Control Transmission Protocol (SCTP)

- Transport layer protocol
- Combined features of UDP and TCP
- Can detect lost data
- Robust and flexible communication
- Can handle multimedia and stream traffic
- Lies between application layer and network layer

- services

1. **Process to process communication**
   - Done via following ports

- 

Sr. No.	Protocol	Port Number	Description
1.	IUA	9990	ISDN over IP.
2.	M2UA	2904	SS7 telephony signaling.
3.	M3UA	2905	SS7 telephony signaling.
4.	H.248	2945	Media gateway control.
5.	H.323	1718, 1719, 1720, 11720	IP telephony.
6.	SIP	5060	IP telephony.

2. **Multi stream facility**
    - Multi stream service provided to each connection is called association. If one stream is blocked other stream can deliver the data

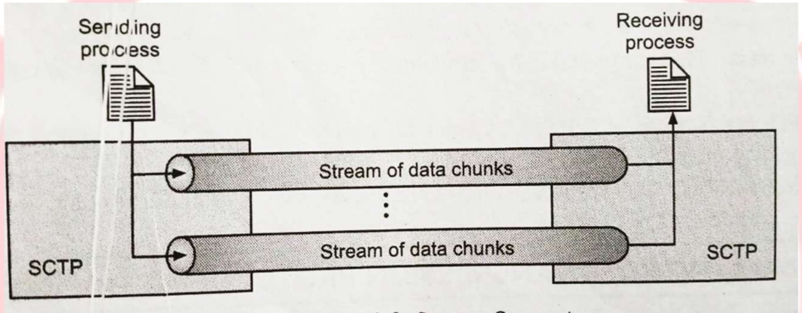


Fig. 3.25: Multiple Stream Concept

3. **Full duplex communication**
    - The data can flow in both direction at the same time
4. **Connection oriented services**
    - If user 1 wants to send and receive message from user 2:
    1. 2 SCTPs establish the connection with each other
    2. Once connection is established data gets exchanged in both directions.
    3. Finally, the association is terminated.
5. **Reliability**
    - SCTP uses acknowledgement mechanism to check the arrival of data
6. **Multihoming**
    - Sender and receiver both are multihomed → connected to more than 1 physical address with multiple IP addresses

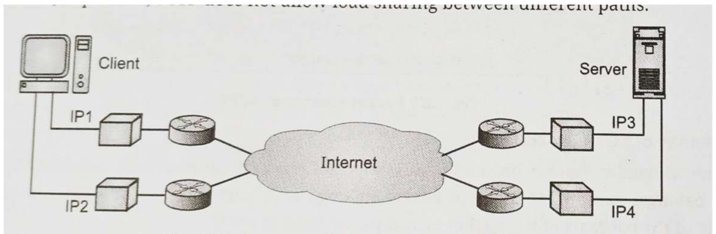


Fig. 3.26: Concept of Multihoming in SCTP

Only one pair of IP can be chosen for normal communication → alternative is used if main choice fails

Q: Explain features of SCTP

- **Transmission Sequence Number (TSN)**
  - Unit of SCTP is Data Chunk. Data transfer is controlled by numbering data chunks. TSN is used to assign numbers to data chunks
- **Stream Identifier (SI)**
  - 16-bit number starts with zero needed to identify streams.

    Each data chunks need to carry SI in the header so that it is properly placed in its stream on arrival.
- **Packets**
  - Data is carried out in the form of data chunks and control information is carried out in control chunks. Both are packed together in the packet.
- **Multihoming**
  - It allows both ends to define multiple IP addresses for communication. Only one is primary and rest are alternative addresses
- **Flow Control**
  - Like TCP, SCTP executes flow control to prevent overwhelming the receiver.

- **Error Control**
  - Like TCP, SCTP executes error control to support reliability. TSN numbers and acknowledgment numbers are used for error control.
- **Congestion Control**
  - Like TCP, SCTP executes congestion control to decide how many data blocks can be inserted into the network.

Q: Compare TCP, UDP, SCTP

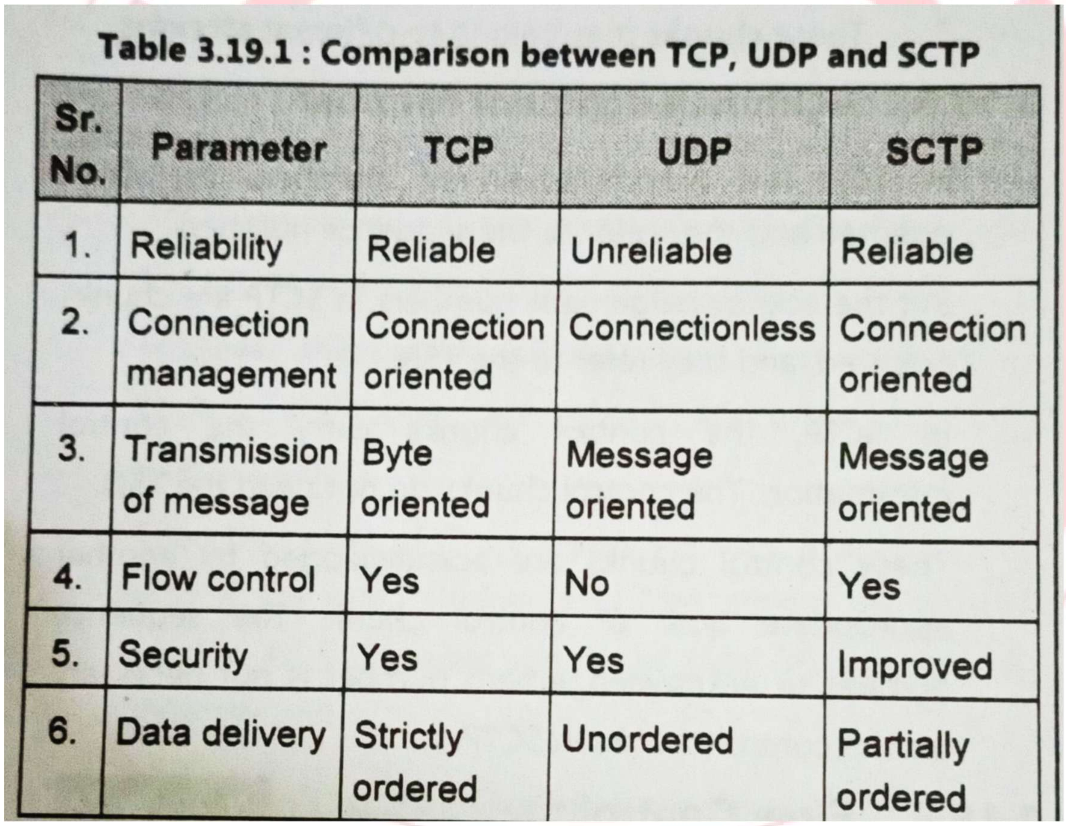


Table 3.19.1 : Comparison between TCP, UDP and SCTP

Sr. No.	Parameter	TCP	UDP	SCTP
1.	Reliability	Reliable	Unreliable	Reliable
2.	Connection management	Connection oriented	Connectionless	Connection oriented
3.	Transmission of message	Byte oriented	Message oriented	Message oriented
4.	Flow control	Yes	No	Yes
5.	Security	Yes	Yes	Improved
6.	Data delivery	Strictly ordered	Unordered	Partially ordered