Unit-IV :

→ Transport layer
- Transport services
- Elements of Transport protocols
- Connection management
- TCP and UDP protocols.

Unit-V :

Unit-4:

* Transport layer

→ Transport layer is fourth layer in OSI model & second layer in TCP/IP model.

→ It provides with end to end connection b/w source & destination,

→ It manage error & flow control b/w devices.

→ It provides reliable delivery of the services

→ the data is represented in SEGMENTS

→ It provides logical communication b/w the applications.

→ the segments are transmitted from source to destination.

→ If segment successfully delivered to destination, this layer provides acknowledgement.

→ If there is any error during transmission, then this layer is responsible to retransmit the data.

→ Transport layer protocols are implemented at end systems but not at routers.

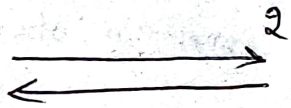
→ Transport layer Protocols are: TCP and UDP.

* Services / Transport Services (or) TCP Services

1. Full duplex communication
2. Connection oriented service
3. Reliable service
4. Process to process communication
5. Stream Delivery Service
6. Segments
7. Flow control
8. Error control.
9. Gurantee delivery.

1. Full duplex Communication:

→ the data will be transferred to Host 1 to Host 2 & Host 2 to Host 1 at a time (or) same time.



→ Sender & receiver will communicate with each other simultaneously.

2). Connection oriented Service

→ At first, the connection will be established. b/w Sender & receiver
→ the data will be transferred.
→ Connection is terminated (After data transferred)

3). Reliable service

→ It provides reliable service
→ It means, it provides guarantee to the Sender that the data transmitted by the Sender arrives correctly at the receiver.

4). Process to Process Communication:

→ For communication of Sender to receiver, Port numbers are used.
→ Totally 1024 predefined Port numbers are available.

Ex: 25 – SMTP
80 – HTTP

→ A Process in Sender can communicate with the Process in receiver with the help of Port number.

Sender Process ⟷ receiver Process

5) Stream Delivery Service

⇒ Stream means collection of data.

Process S ⟶ Process R

→ Sender process can send the data as a stream of bytes & stored on buffer.
→ Receiver process can also receives the data as a stream of bytes & receiver will consume data from buffer.

6) Segments.

→ On top of Transport layer we have Application layer
→ On bottom — Network layer.
→ Application layer represents the data in the form of messages
→ Application layer sends message to Transport layer.
→ If the message is too big, then it will be splitted into various segments.

→ And Transport layer add a header to each of segment then send to n/w layer.

7) Flow Control

→ Control the flow.
→ If the receiver is overloaded, then it control the flow.

8) Error control

→ If there is any error, then it is reported to the Sender.
→ So, there is no problem.

9) Guarantee delivery

→ It ensures guarantee delivery.

* Elements of Transport protocols

1. Segmentation & Reassembly.
2. Error detection & correction
3. Flow control
4. Congestion Control
5. Addressing

1. Segmentation and Reassembly

Segmentation: Breaking down large data into smaller segments for transmission.

Reassembly: Reassembling (means combining) the segments into the original data at the receiving end.

2. Error detection & Correction

→ Identifying error & correcting it.

- Checksum
- Acknowledgment
- Negative Acknowledgement.

3. Multiplexing & Demultiplexing

Multiplexing: combining multiple data streams into a single stream.

Demultiplexing: Separating the combined data stream into individual streams.

4. Addressing → assigning addresses, Port numbers to devices

Types of addresses

- Source Address: Address of sender data.
- Destination Address: Address of receiver data.

Port numbers: used to identify application.

* Connection Management

→ Connection management involves 3 phases or stages:

1. Connection Establishment
2. Data Transfer
3. Connection Termination

1. Connection establishment

→ A connection has to be established before transmission.

→ Full duplex is used

→ For connection establishment, three-way handshaking is used.

1. Client sends SYN segment to server (to establish connection)

SYN - Synchronization

SYN segment: It specifies that; I want to establish a connection with you.

2. Server acknowledges the client segment & sends SYN+ACK segment to client

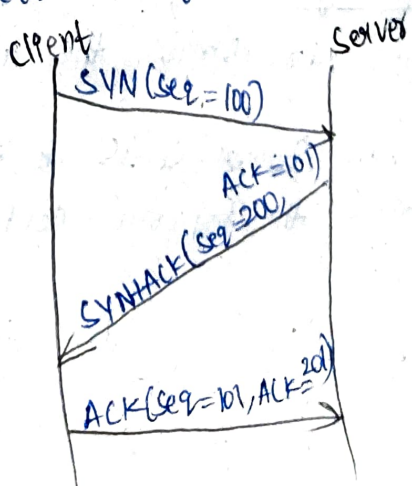
→ Server accepts the SYN segment

- then server sends SYN+ACK segment to client.

- SYN specifies that server wants to establish a connection with client.

3. Client acknowledges the server

I am ready to have a connection with you



Data Transfer

→ Once the connection is established, then b/w the client and server.
- then both can transfer the data.

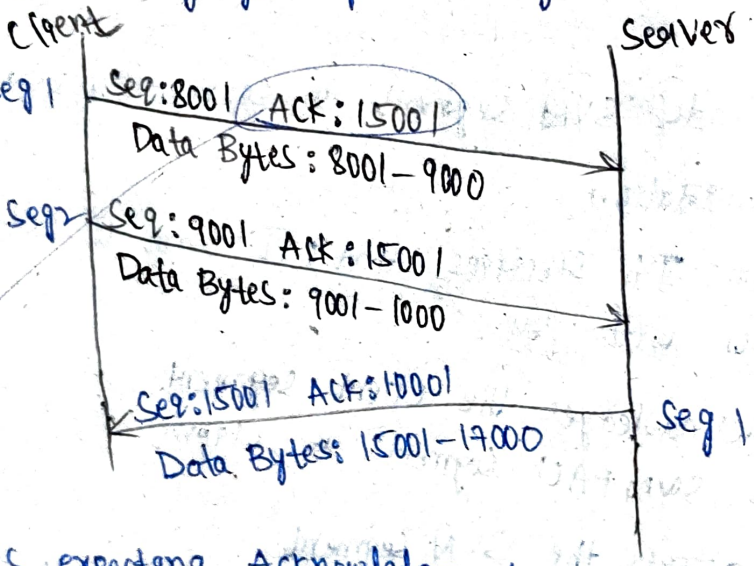
→ For data transferring, Full duplex is used.

→ Initially client sends a segment to the server.

→ client sends 2 segments.

→ Sequence number means the first byte

→ ACK specifies client is expecting a segment from server
- where the starting byte of that segment is 15001.



* client is expecting Acknowledgement 15001 from server.

→ , client is sends a segment to server
- where the segment contains the data of 9001-10000 bytes

→ Here, client & server uses Piggybacking concept.

Piggybacking : Along with the data, the corresponding machine also transmits the acknowledgement

→ then server sends segment to the client
- where the segment contains data from 15001-17000

Connection Termination

→ Once the data is transferred, then we need to terminate the connection.

→ We can terminate connection in 2 ways:
1. Three-way handshake.
2. Four-way handshake.

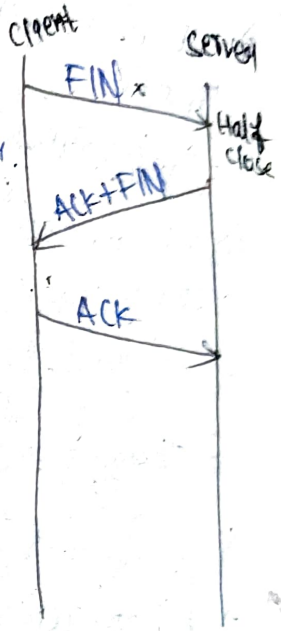
1. 3-way

→ Let assume client has no data to send

→ then client send FIN segment to the server.

FIN - Finish
↓
It specifies that, client wants to terminates it's connection.

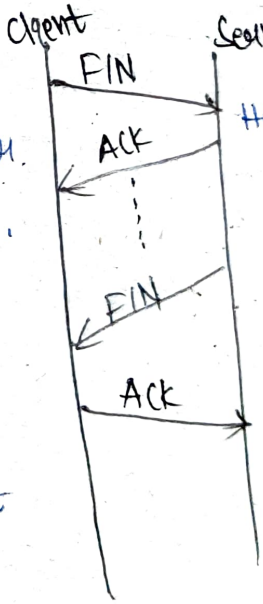
⇒ whenever client sends FIN to server, then the connection b/w client & server is terminated. It is called as "Half close"



→ Next assume, server has no data to send
- then server also send FIN segment & Acknowledgement
So
- client receives ACK + FIN segment.
- And client sends ACK to server (responds)

→ Connection b/w server & client is terminated.

2. Four-way handshake.



→ client has no data to send.
- client send FIN segment to server.

→ Let assume, server has some data.
- server sending acknowledgment to client.
- server performs some operations bcz it has data.
- Once server has no data, then it will send FIN to client.
- So connection closed.

→ client receives & then responds with ACK segment then connection is closed.

TCP VS UDP

TCP	UDP
1). TCP stands for "Transmission Control Protocol".	1). UDP stands for "User Datagram Protocol".
2). It is a connection oriented protocol.	2). It is a connection less protocol.
3). It is Reliable protocol.	3). It is Unreliable protocol
4). It provides Gurantee delivery.	4). It doesn't provides gurantee delavery.
5). It uses 3-way handshake	5). No need of handshake.
6). There is more overhead.	6). Less overhead.
7). Provides flow control.	7). No flow control.
8). TCP is best suitable for E-mail file sharing.	8). It is suitable for Audio/video.
9). TCP is slower.	9). UDP is faster.

UDP Services

1. Process to Process communication
2. Connection less service
3. No flow control
4. No Error control
5. Congestion control
6. & Checksum
7. Unreliable
8. Encapsulation & Decapsulation
9. Multiplexing & Demultiplexing

UDP- User Datagram Protocol

Features of UDP

→ Connection less → No connection b/w source & destination
→ Unreliable → source process send data, but doesn't bother about it
→ No guarantee delivery
→ No flow control
→ less overhead → No need of retransmission, connection establishment.

UDP header

→ The size of UDP header is 8 bytes

1 byte = 8 bits

→ Source process can send data to destination process with the help of port numbers

Header format of UDP

Source Port number (16 bits)	Destination Port number (16 bits)
total length (16 bits)	Check sum (16 bits)

→ The total length UDP is 16 bits, so we have a 2^{16} bits i.e. 65,535 bits (0 to 65,535) → we can send data.
→ Check sum capacity is 16 bits,
→ Check sum is used in Error detection