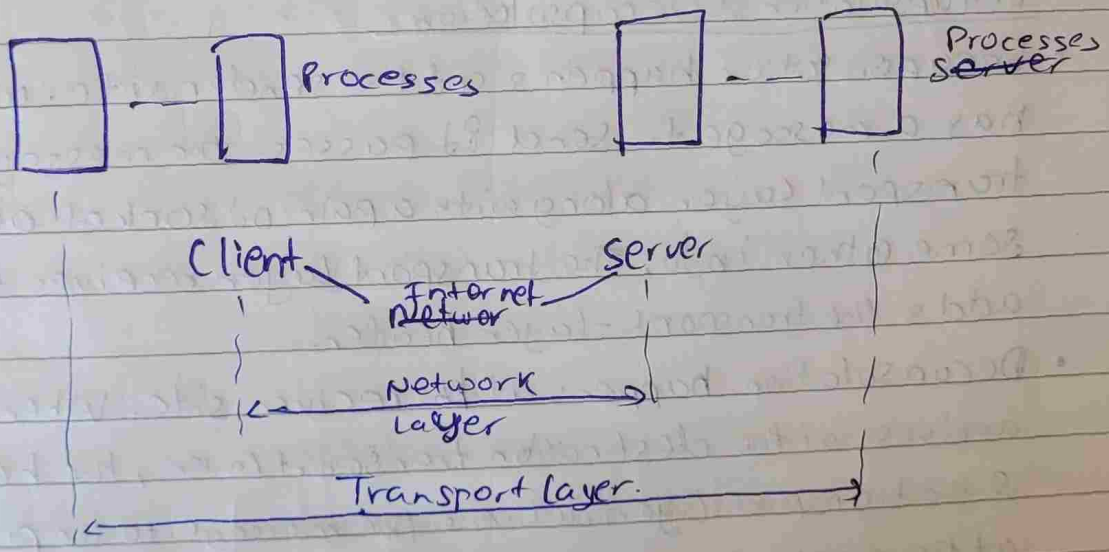
MODY

Transport Layer Services

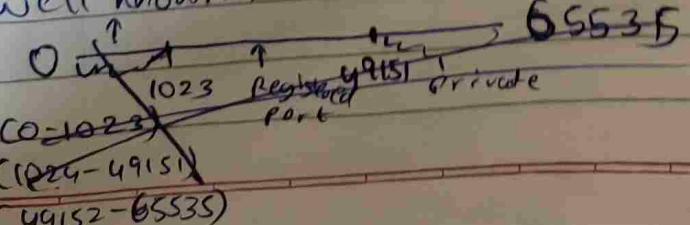
→ Process-to-process communication

- A process is an application layer entity that uses the services of the transport layer
- The network layer is responsible for communication at the computer level.
- The network layer protocol can deliver the message only to the desitination computer.
- The transport layer is responsible for delivery of the message to the appropriate process



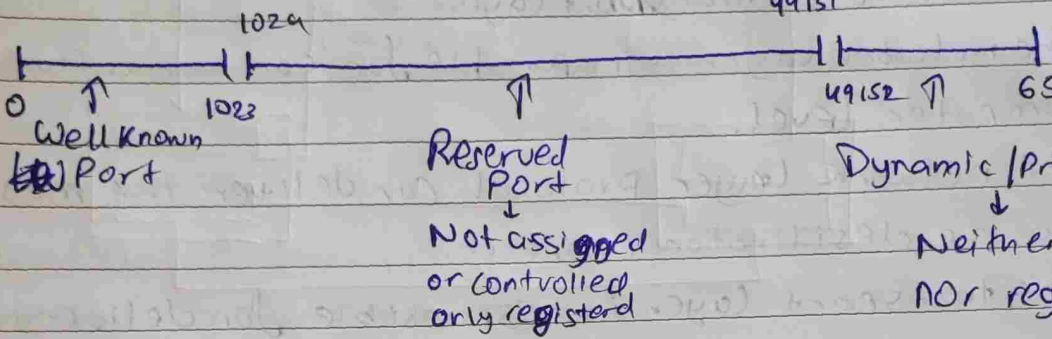
→ Addressing

- The local host & remote host are identified by IP address but for a process we need second identifiers called port address
- Port address → 0 to 65535
- The client program defines itself with a port no. called ephemeral port no.
- TCP/IP has decided to use universal port no. for server called well known port no. (0-1023).



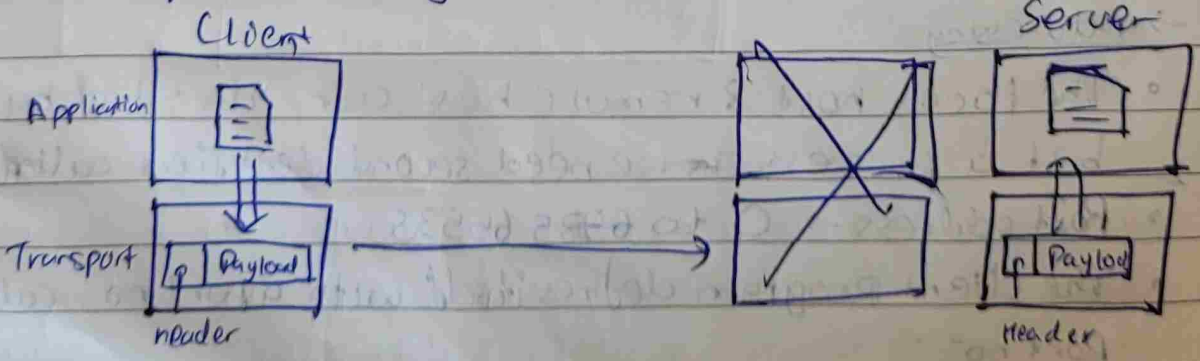
→ ICANN

- Internet corporation for Assigned names & no° Range is divided into three ranges:



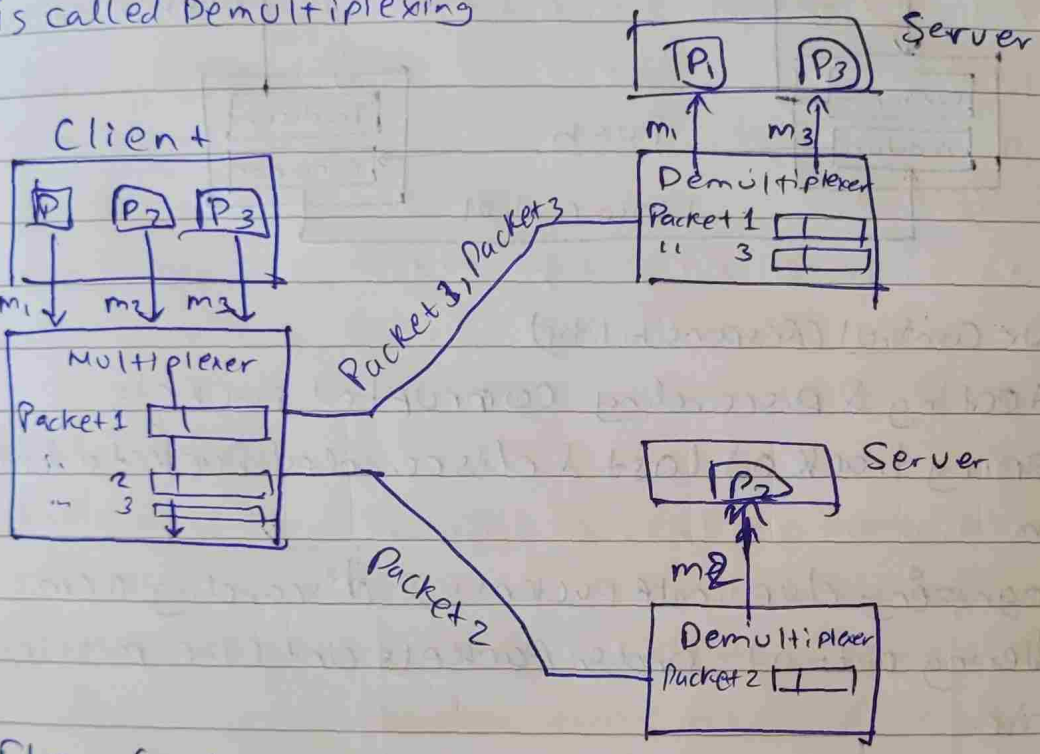
→ Encapsulation & Decapsulation

- Encapsulation happens at the sender site. when a process has a message to send it passes the message to the transport layer along with a pair of socket address and some other info. The transport layer receives the data and adds the transport-layer header.
- Decapsulation happens at the reciver site. When the message arrives at the destination transport layer, the header is dropped & the transport layer delivers the message to the process running at the application layer



→ Multiplexing & Demultiplexing

- Whenever an entity accepts items from more than one source is called Multiplexing
- Whenever an entity delivers items to more than one source is called Demultiplexing



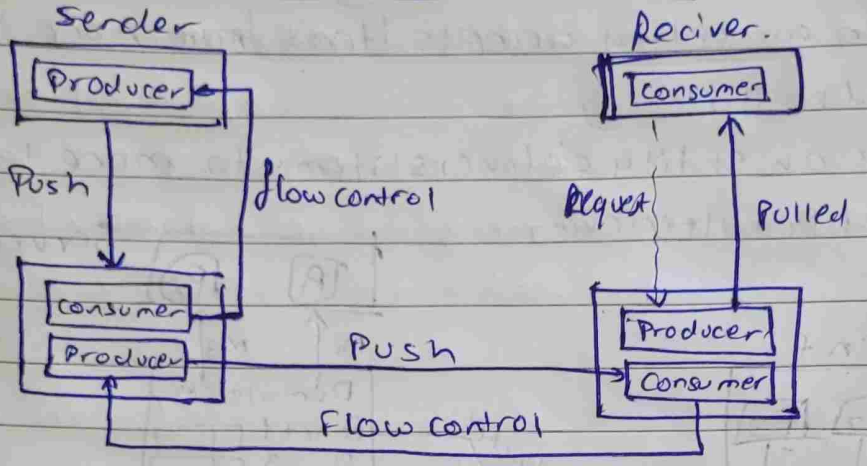
→ Flow Control

- It is balance between producer & consumer rates
- If production rate > consumption rate ⇒ congestion can occur leading to loss in packeds

→ Pushing or Pulling

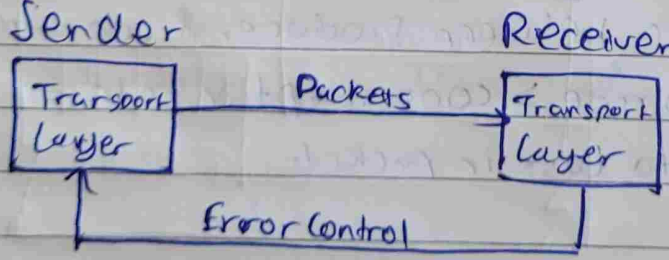
- If a sender delivers a items whenever they are produced without prior request from the consumer it is called pushing
- If a producer delivers a items after the consumer has requested them is called pulling.

→ Flow control at Transport Layer



→ Error Control (Responsibility)

- Detecting & Discarding corrupted packets
- Keeping track of lost & discarded packets & resending them
- Recognizing duplicate packets & discarding them.
- Buffering out-of-order packets until the missing packets arrive.

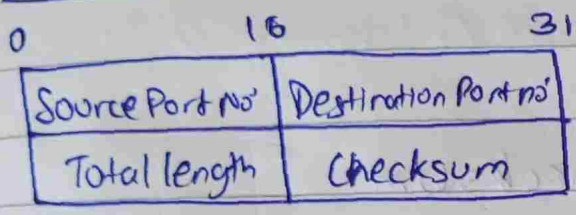


→ Congestion Control

- Congestion may occur if the load on the network is greater than the capacity of the network
- Congestion Control refers to the mechanisms & techniques to control congestion & keep the load below the capacity

UDP

→ Header Format



→ UDP is a connectionless, unreliable transport protocol.

→ Why use it then?

- UDP is a simple protocol using a minimum of overhead
- If process wants to send a small message and does not care much about reliability it can use UDP.
- It takes much lesser time to send small message between sender & Reciever using UDP than TCP

→ Services

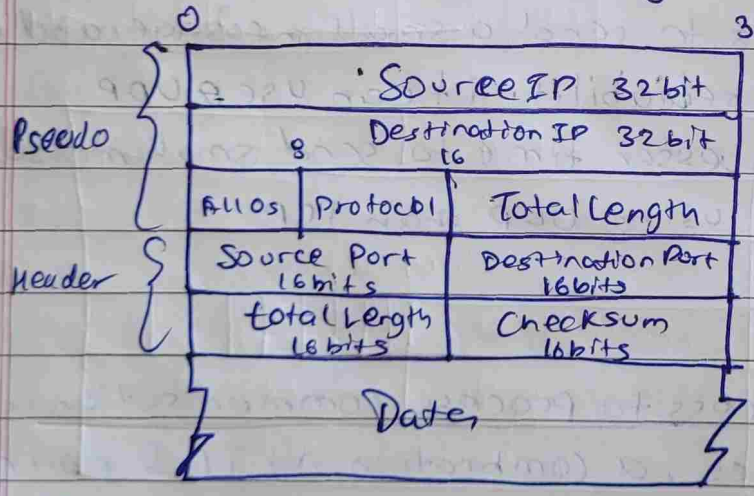
1) It provides process to process communication using socket addresses, a combination of IP & port no.
Some eg.
7 - Echo
9 - Discard
53 - Domain
69 - TFTP

2) Connectionless
- The user datagram sent by UDP is an independent datagram
- There is no relationship between different user datagram even if they are coming from the same source process & going to the same destination program.

→ They arent numbered.
→ Also no connection is established or terminated.

3) Flow control
- Does no provide flow control.
- Due to this ~~that~~ the process using UDP should provide for this service, if needed.

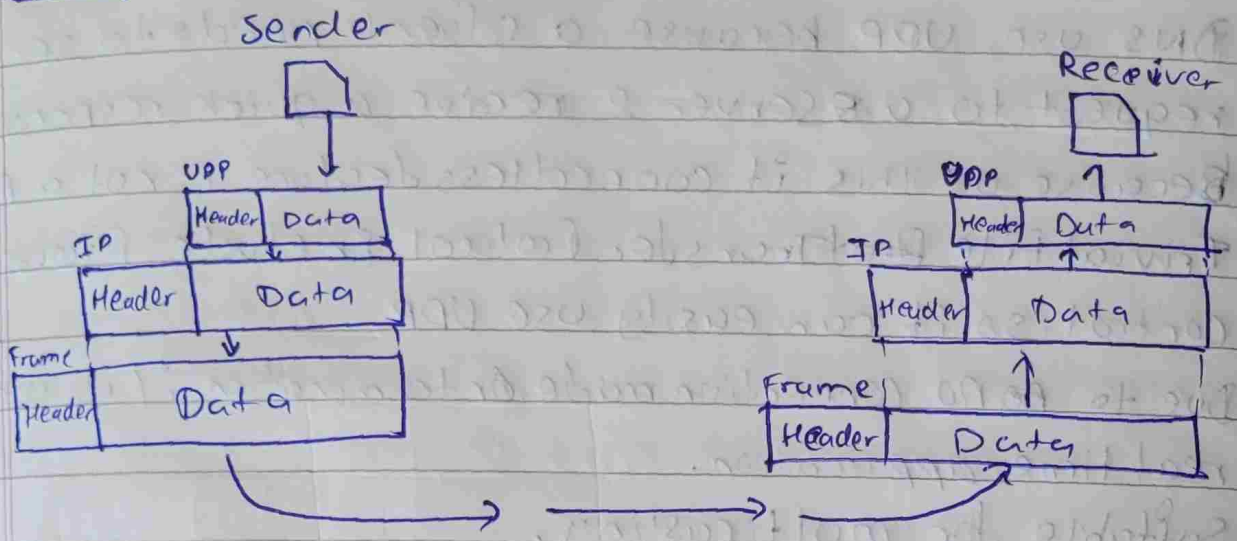
4) Error Control
- No Error Control except checksum.
- So the ~~user~~ sender does not known id a message has been lost or duplicated.
- The pseudoheader is part of the header of the IP packet in which user datagram is encapsulated



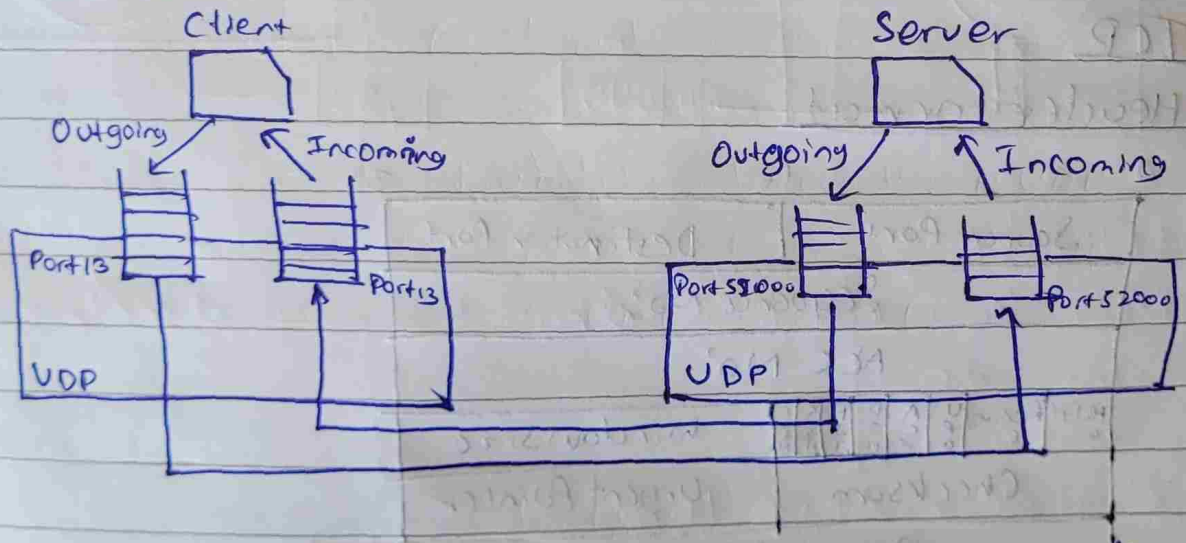
- If the checksum does not include a pseudoheader, a user datagram may arrive safe. However if the IP header is corrupted, it may be delivered to the wrong host.

4) Congestion Control
- As it is connection less service, it does not provide congestion control.

5) Encapsulation and decapsulation

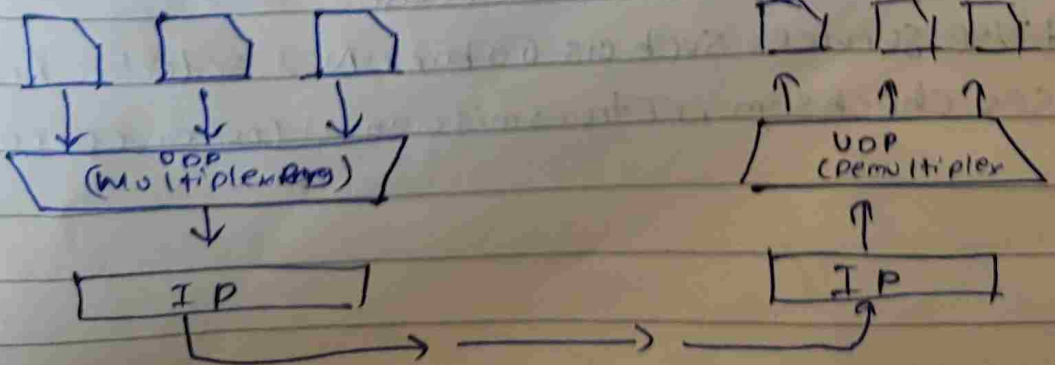


6) Queuing



→ It is the process of storing data packets in a buffer before they are sent over the network

7) Multiplexing & Demultiplexing

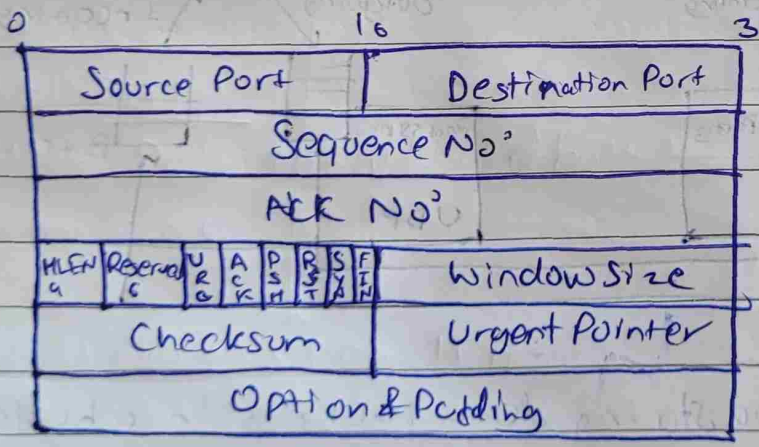


→ Application

- DNS use UDP because a client needs to send a short request to a server & receive a quick response from it. Because of this it connectless feature is not a problem
- Trivial File ~~Prot~~ Transfer Protocol include Flow & error control so it can easily use UDP
- Due to ~~no~~ no connection made or termination if is used in real time application.
- Suitable for multicasting.
- Used in Routing updating protocol such as Routing Information Protocol (RIP)

TCP

→ Header Format



→ TCP is a connection oriented, reliable protocol
→ It explicitly defines connection established, data transfer & connection termination Phase.
→ It use services such as Go back-N & selective Repeat ARQ protocol
→ Uses checksum, retransmission of lost or corrupted packets

-> Services

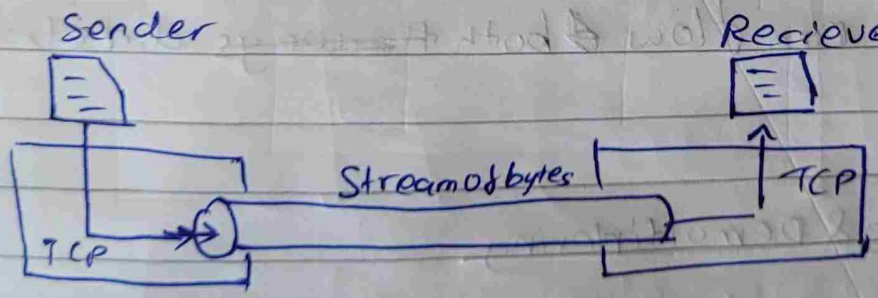
1) Process to Process Communication

- Provides using Port no°
- eg:

7 - Echo	67 - BOOTP
9 - Discard	80 - HTTP
20821 - FTP	
25 - SMTP	

2) Stream Delivery

- Allows sending process to deliver data as a stream of bytes & allows the receiving process to obtain as a stream of bytes.
- TCP creates an enviroment in which two processes seem to be connected by a imaginary 'tube' that carriers their bytes across the Internet.

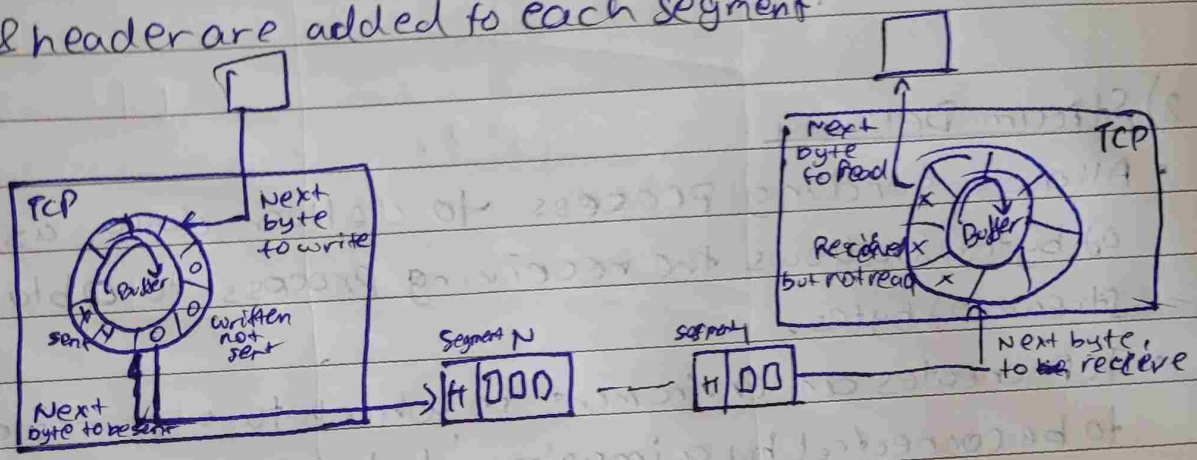


3) Sending & Recieving Buffers

- It is used to store data during transmission of a message
- Sending Buffers : Stores data that written but yet not transmitted
- Recieving Buffers : Stores data that arrives out of order or in correct order but not yet read.

4) Segments

- Although buffering handles the disparity between speed of producing & consuming, we need one more step before we sent data
- The network layer as a service provider for TCP, needs to send data in packets not as a stream of bytes
- TCP groups a no° of bytes into ~~segm~~ packets called segments & header are added to each segment



5) Full Duplex

- Mean Data can flow in both ~~the ways~~ direction at the same time

6) Multiplexing & Demultiplexing

7) Connection Oriented

- The two TCP's establish a logical connection between them
- Data are exchanged in both direction
- The connection is terminated

8) Reliable

→ Features

1) Numbering System

- Byte numbers: The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with an arbitrarily generated no°
- Sequence number: The value in the sequence no° field of a segment defines the no° assigned to the first data byte contained in that segment.
- Acknowledgment number: The value in the acknowledgment field in a segment defines the number of next byte a party expects to receive. It is cummulative

2) Flow Control

- The sending TCP controls how much data can be accepted from the sending process.
- The receiving TCP controls how much data can be sent by the sending TCP

3) Error Control

4) Congestion Control

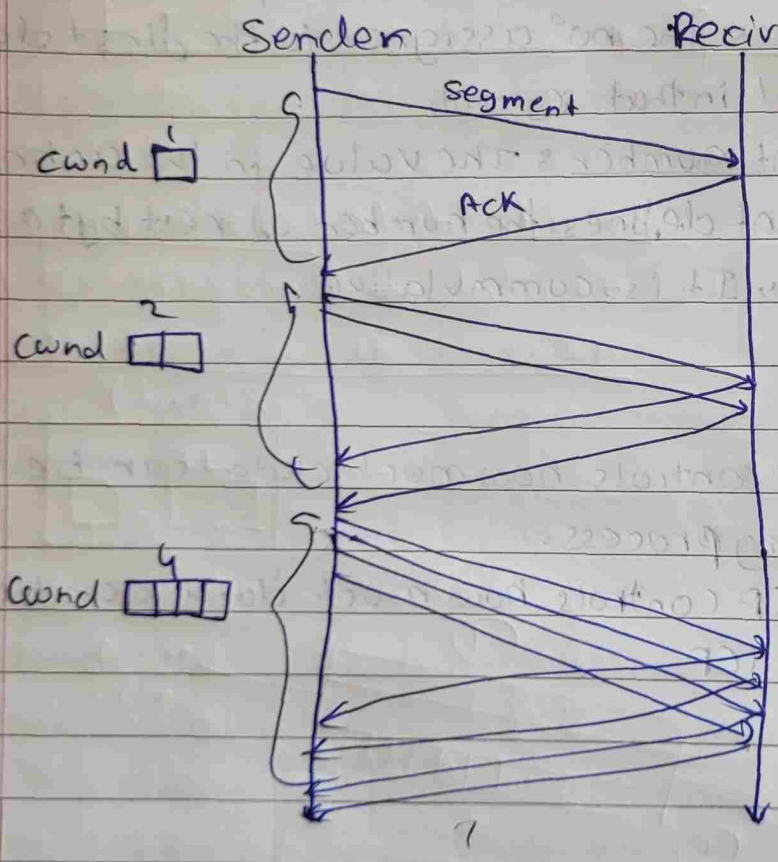
i) Congestion window

- We assume that it is only the receiver that can dictate to the sender the size of the sender's window.
- If the network cannot deliver the data as fast as it is created by the sender, it must tell the sender to slow down
- The actual window size = min (rwnd, cwnd)

ii) Policy

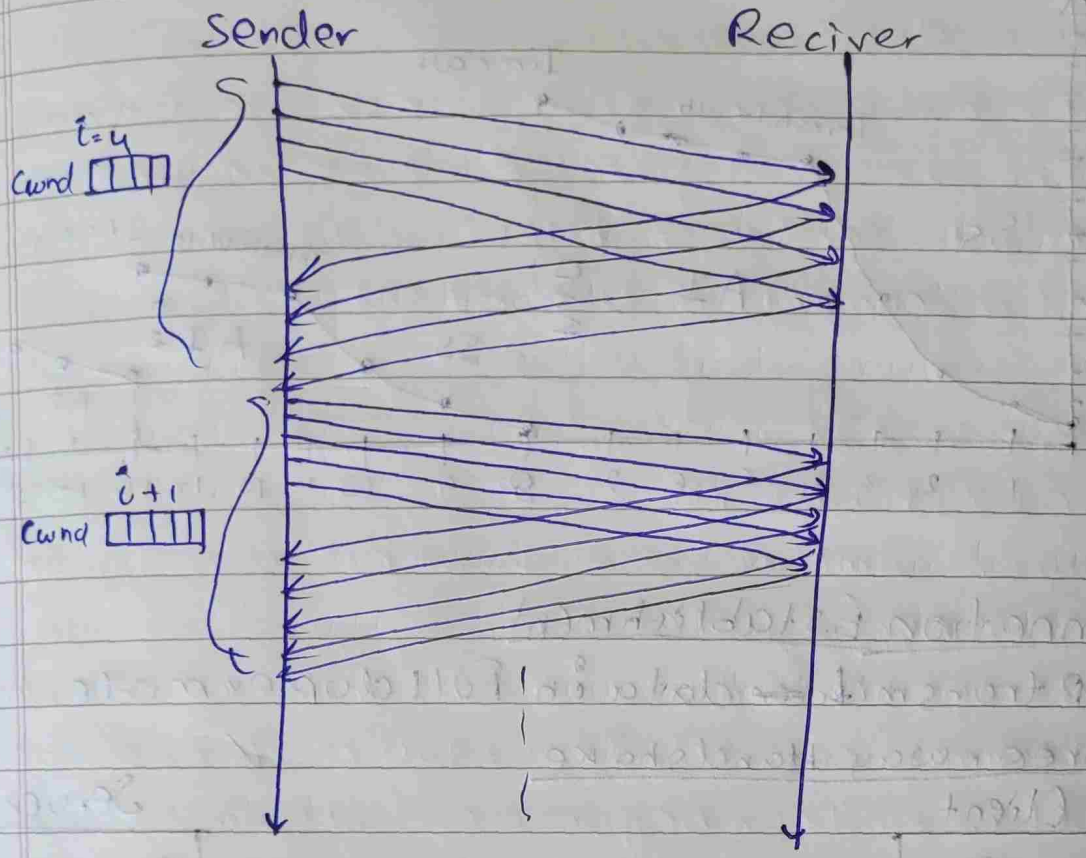
a) Slow Start: Exponential Increase

→ It is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size.
→ The size of the window increases one MSS each time one ack ~~is received~~ arrives.
→ As the name implies it start slow & increases exponentially.



→ In the slow start algo., the size of congestion window increase exponentially until it reaches a threshold.

b) Congestion Avoidance : Additive increase
→ After the threshold is reached if the exponential growth continues it will lead to congestion.
→ So to avoid it ~~we~~ after the threshold we continue with additive increase.
→ In the congestion avoidance algo; the size of the cwnd increases additively until congestion is detected.



c) Congestion Detection : Multiplicative Decrease

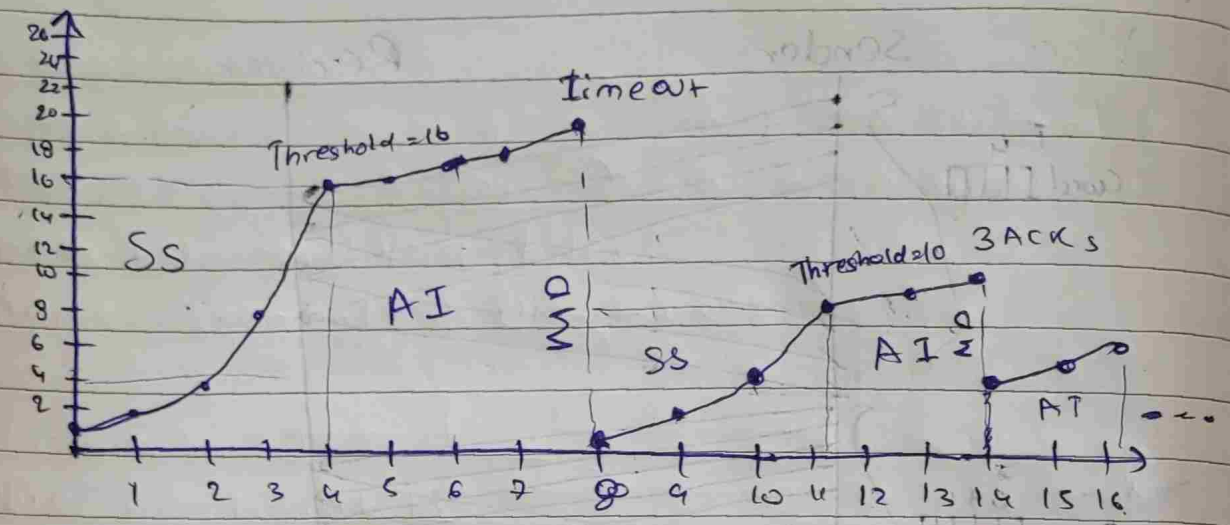
→ The way TCP detects a congestion :

1) If a time-out occurs, there is a strong possibility of congestion, a segment is probably been dropped in the network. In that case TCP reacts strongly :

- It sets the value of the threshold to half of the current window size
- It reduces the cwnd to one segment.
- It starts the slow-start phase again.

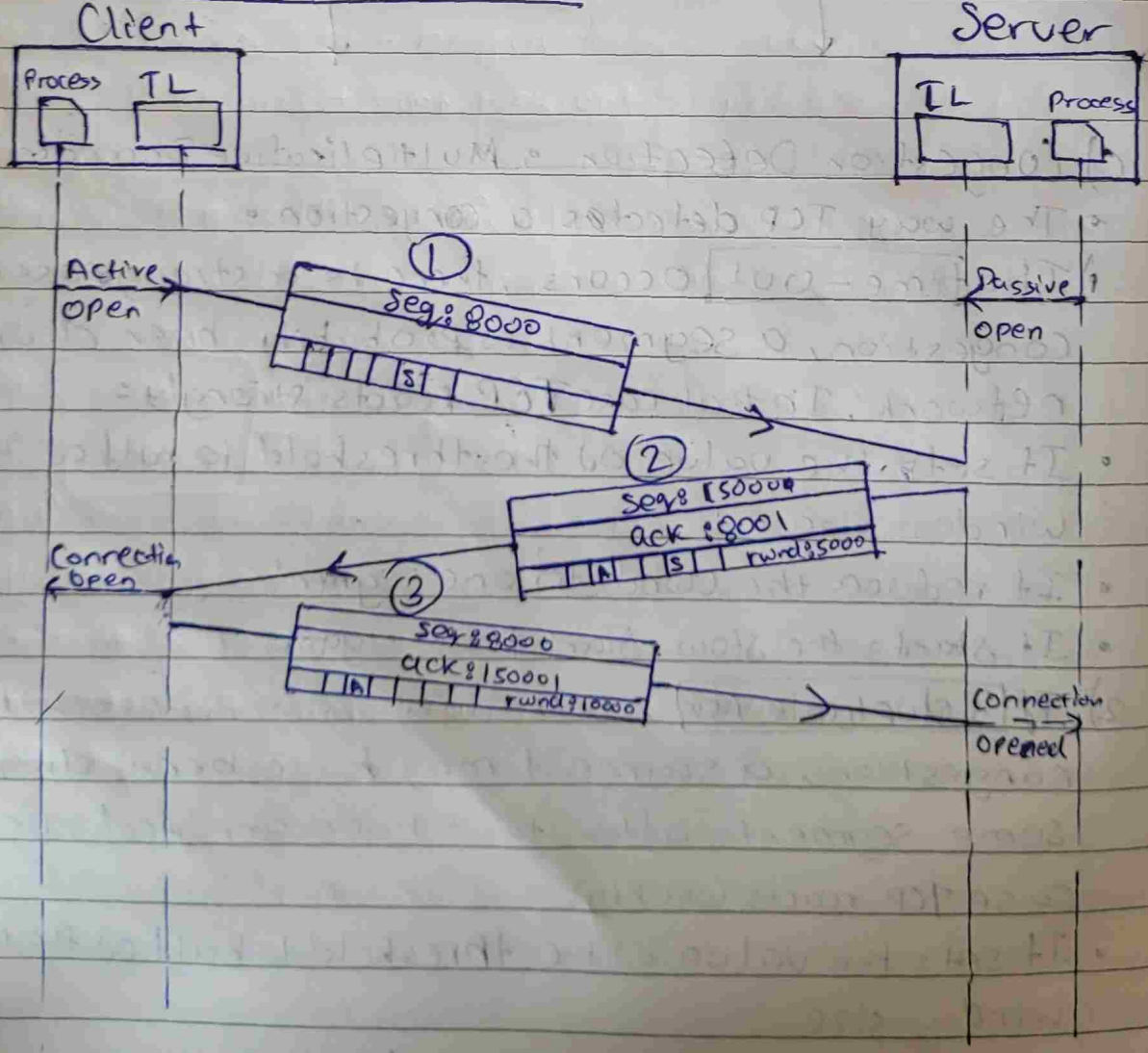
2) If 3 duplicate ACK, there is a weaker possibility of congestion, a segment may have been dropped but some segments after that have arrived safely. In that case TCP reacts weakly :

- It sets the value of the threshold to half of the current window size
- It sets cwnd to value of the threshold
- It starts the congestion avoidance phase



→ Connection Establishment

- TCP transmits ~~for~~ data in Full duplex mode
- Three-way Handshake



1) The client sends the first segment, a SYN segment. This segment is used for synchronization of sequence no°, and the segment does not contain an ack no° & does not define window size.
SYN segment can not carry data but it consumes one sequence no°

2) The server sends the second segment, a SYN+ACK. This segment has a dual purpose:
- SYN for communication in the other direction
- The server also ack the receipt of the SYN segment from the client by setting ACK flag
- Because segment contain a ACK, it also needs to define a window size.

A SYN+ACK cannot carry data, but it does consume one sequence no°

3) The Client sends the third segment - an ACK segment. and also defines a window size. An ACK segment if carrying no data consumes no sequence no°

→ Data Transfer
- After connection establishment bidirectional data transfer can tak place
- ACK can be piggybacked with the data
- Pushing data
  - Delayed transmission & delayed delivery of data may not be accepted by the application program.
  - application program at the sender can request a Push operation
  - TCP can choose whether or not to use this feature.

- **Urgent data**
  - Each byte has a position in the stream, however there are occasions in which an application program needs to send urgent bytes
  - Send a segment with URG bit set.

→ **Connection Termination**

i) **Three way handshake**

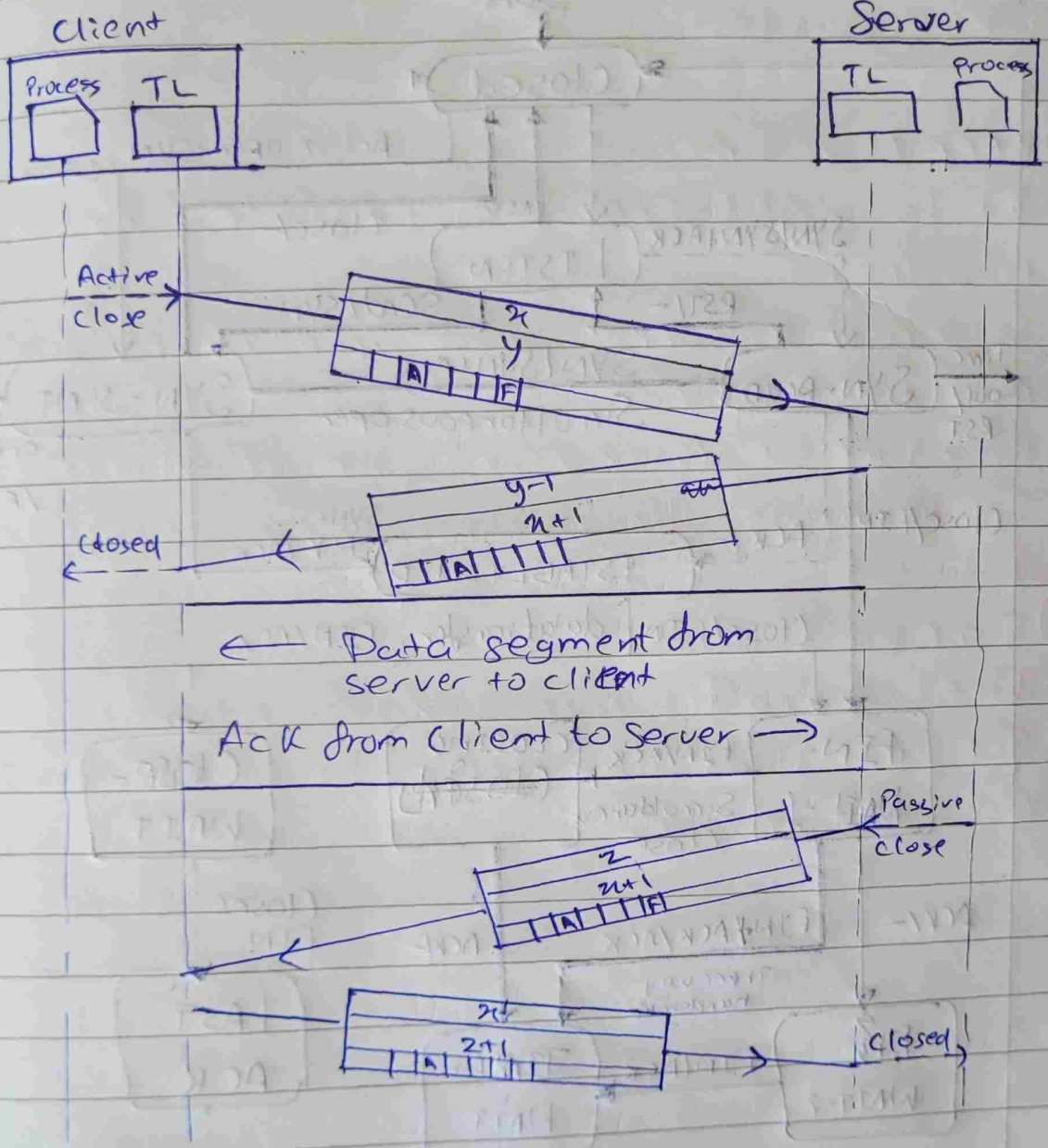
Note: Diagram same as Establishment but instead of open it does & SYN &, ACK it is ACK, FIN

a) Client TCP sents a FIN segment in which FIN flag is set
- FIN segment can include the last chunk of data sent by the client or it can be just a control segment
- The FIN segment consumes one sequence no° if it does not carry data

b) Server TCP, after receiving FIN segment sends a FIN+ACK segment to confirm receipt of FIN segment & also to announce closing of connection in other direction.
- This segment can also carry last chunk of data from server
- If does not carry data, it consumes only one sequence no° because it needs to be ACK

c) Client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server
- This segment cannot carry data and consumes no° sequence no°.

2) Half-close

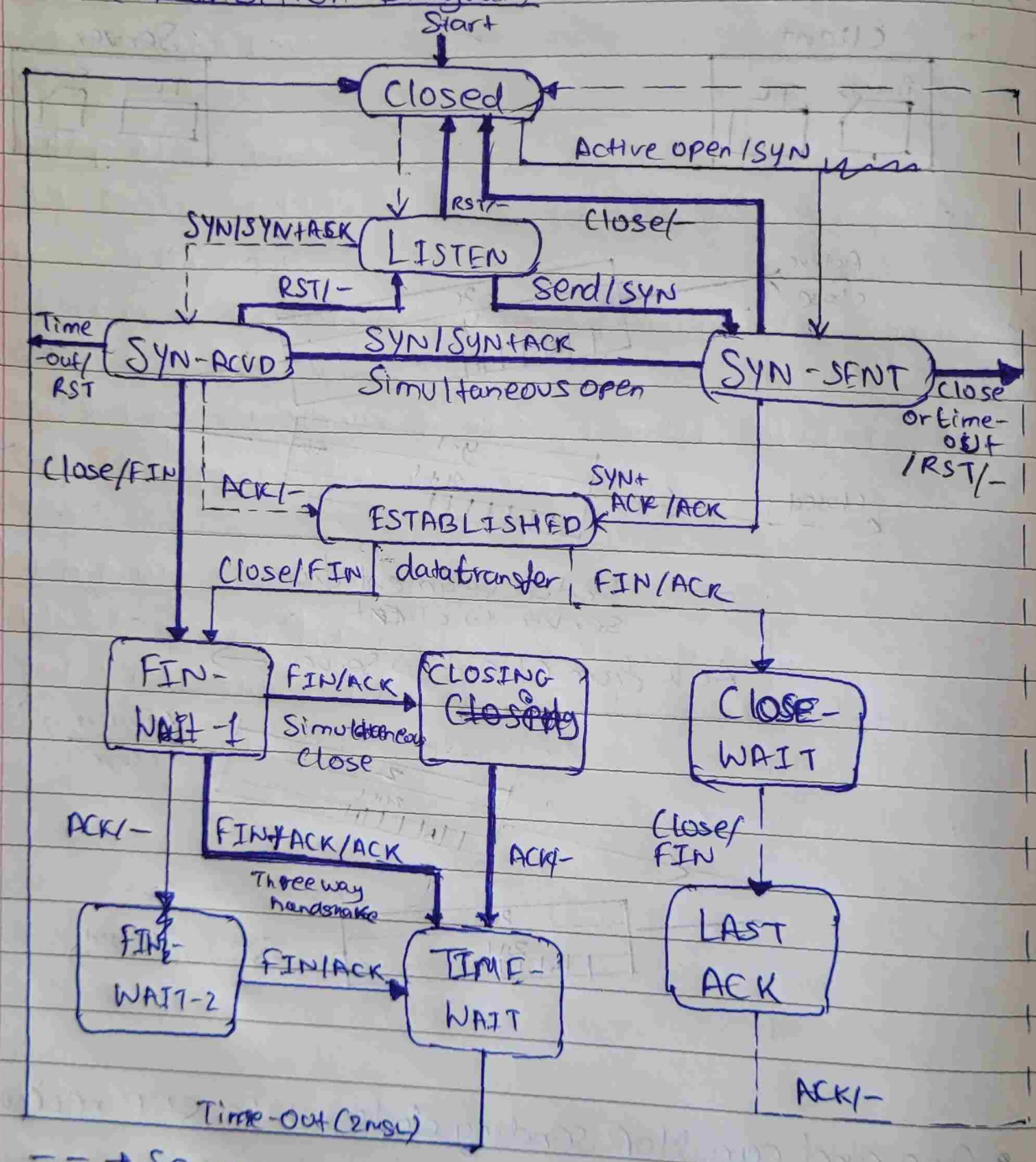


- One end can stop sending data while still receiving data

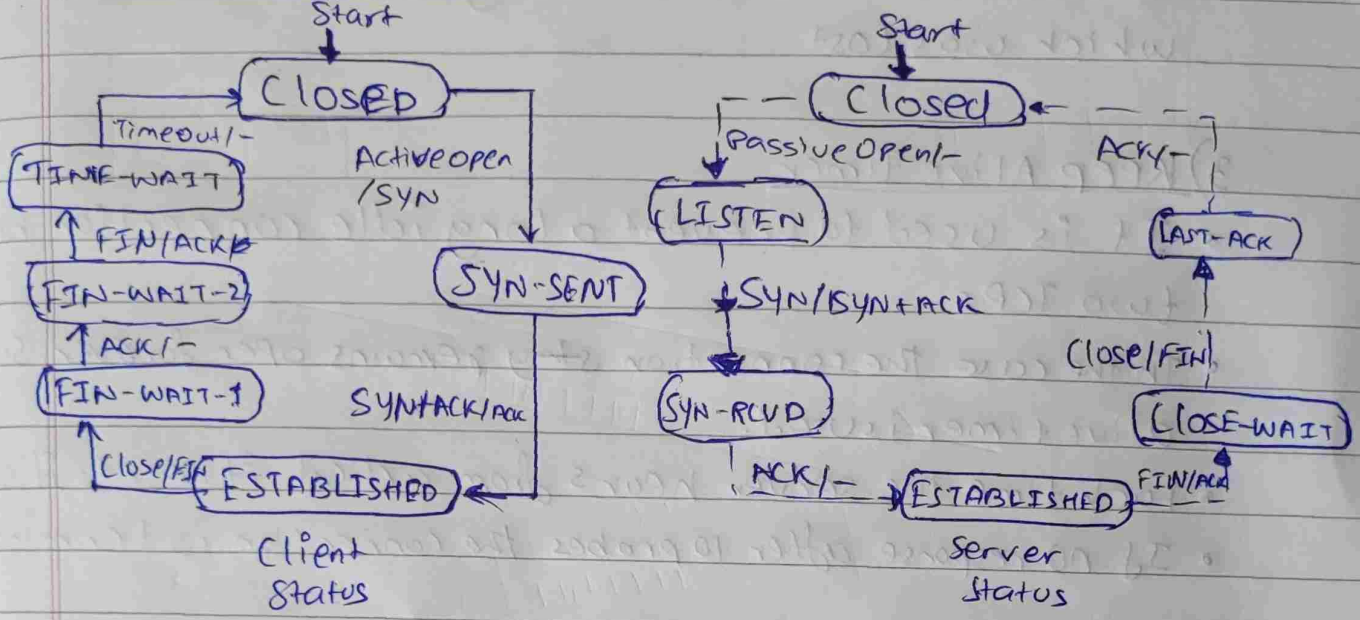
→ Connection Reset

- ICP at one end,
  - may deny a connection request
  - may abort an existing connection
  - may terminate an idle connection
- All done by RST flag.

State transition Diagram



Halfclose Scenario



→ TCP TIMERS

1) Retransmission Timer

- When TCP sends a segment the timer starts & stops when acknowledgment is received
- If the timer expires timeout occurs & the segment is retransmitted.
- To calculate RTO we need to first calc RTT:
  - Measured RTT: It is the time required for the segment reaching the destination & be acknowledged (RTTm)
  - Smoothed RTT: It is the weighted avg of RTTm
  - Deviated RTT

2) Persistent Timer

- To deal with zero window size deadlock situation
- When the sending TCP receives an ack with window size = 0 it starts persistence timer.
- When it goes odd, the sending TCP sends a special segment called Probe.
- This segment contains only 1 byte of new data.

- The probe causes the receiving TCP to resend the ack which was lost

3) Keep Alive Timer

- It is used to prevent a long idle connection between two TCPs
- In the case the connection ~~stay~~ remains open forever, so keep alive timer is used.
- Each time the server hears from a client, it resets it's Timer
- If no response after 10 probes the connection is terminated.

4) Time wait Timer

- Used during TCP connection termination
- The timer starts after sending the last ACK for 2nd FIN & closing connection.

Aspect	SCTP	TCP
Definition	Is designed for message-oriented applicati	Is designed for connection-oriented protoco applicati for reliable data tran
Multi-Streaming	Supports within a single connection, ↓ Packet loss	Does Not support
Multi-Homing	Supports, allowing connection to use multiple IP addresses reliability	Does Not support
Data Orientation	Message oriented, providing better control over data chunks	Byte stream-oriented, treatin data as a continous stream
Error Handling	Provides built-in mech- to handle SYN flood attac & other issues	Uses ack & retransmissio for error correction
Data transfer	Offers more reliable & secure data transfer	Less reliable compared to SCTP in certain cond
Security	More secure	Less Secure
Partial Data Transfer	Supports	Does not support
Unordered Data Delivery	Support	Does Not support
Overhead	Higher overhead due to additional features	Higher overhead compared to UDP, but lower than SCTP
Suitable for real-time Apps	Suitable due to multi-streaming & Partial delivery	Not ideal

MODS

Server

→ A server is a program running on the remote machine providing service to the clients.

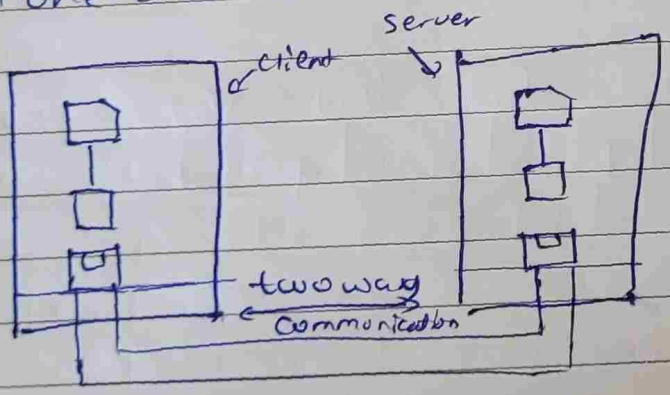
→ When it starts it opens the door for incoming request from clients, but it never initiates a service until it is requested to do so

→ It is an infinite program i.e. it runs till a program arises

Socket

→ It is a ~~program~~ software abstract simulating a hardware socket we see in a daily life.

→ For data communication to occur, a pair of sockets each at one end of communication is needed.



DNS

→ It is a client/server application program used to help other application program

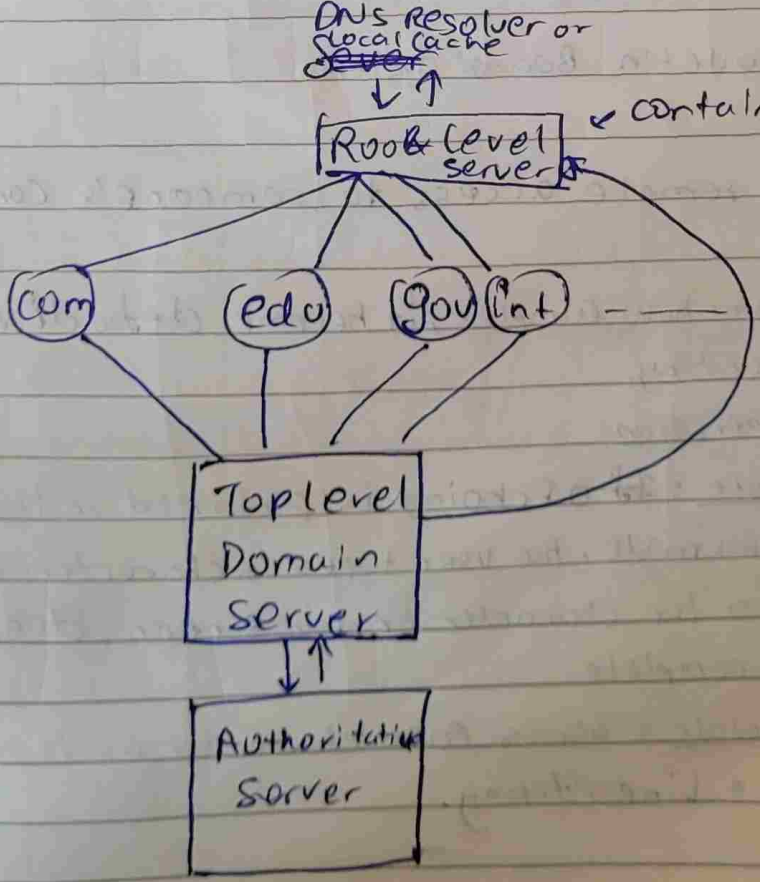
→ It is used to map a host name in application layer to an IP address in the network layer.

→ To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the internet.

→ However, people prefer to use name instead of numeric address

→ Working

- DNS client sends request to the server. While DNS server sends a response to the client
- If a client requests contains a name which is converted into an IP address known as the forward DNS lookups while requests containing an IP address which is converted into name is known as reverse DNS lookups
- If a client like a web browser sends a request containing a hostname is sent to DNS resolver which is then sent to the DNS server to obtain the IP address of a hostname.
- If not found it is forward to the server to get it & follows the hierchracy eg: google.com



Telnet

→ Terminal Network

→ It is a standard TCP/IP protocol for virtual terminal service as proposed by ISO

→ Enables the establishment of a connection to remote system in such a way that the local terminal appears to be a terminal at the remote system.

→ It uses NVT (network virtual Terminal) system to encode characters on the local system. On the server machine NVT decodes the characters to a form acceptable to the remote machine.

→ NVT uses a set of characters for data & a set of characters for control.

→ Application

- CLI
- Accessing Bulletin Board systems

→ Adv

- It provides remote access to someone's computer system

→ Disadv

- As it is somehow complex, it become difficult to beginners in understanding.

→ Modes of Operation

- Default mode : Echoing is performed in this mode by the client. In this mode, the user types a character & the client echos the character the character on the screen, does not send till the line is complete
- Character Mode : Sents each character as it is typed.
- Line Mode : Line editing.

FTP

→ File transfer Protocol is TCP/IP client-server application for copying files from one host to another.

→ It requires two connections for data transfer: a control & a data connection

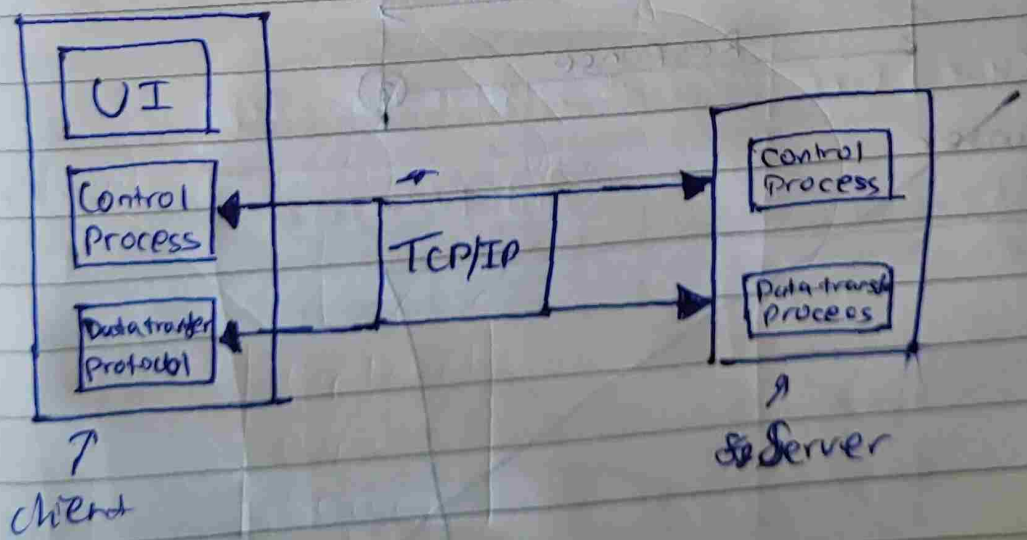
→ FTP employs NVT ASCII for communication between dissimilar systems.

→ Prior to the actual transfer of files, the file type, data structure, & transmission mode are defined by the client through the control connection.

→ There are 3 types of file transfer

- server to client
- client-to-server
- transfer of list of directories

→ It uses secure socket layer (SSL) for secured transfer of File



WWW

→ It is a repository of information linked together from points all over the world. Hypertext & hyper media documents are linked to one another through pointers.

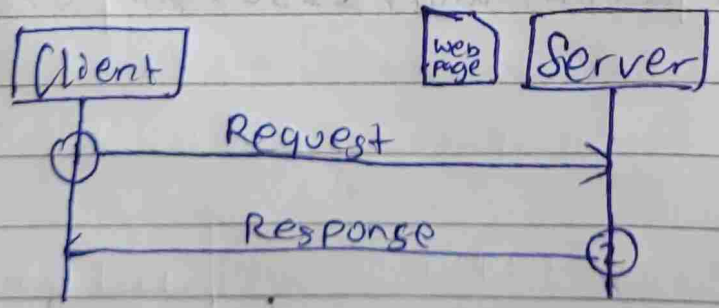
→ It is made up of clients & servers.

→ A client or a browser interprets & displays web doc.

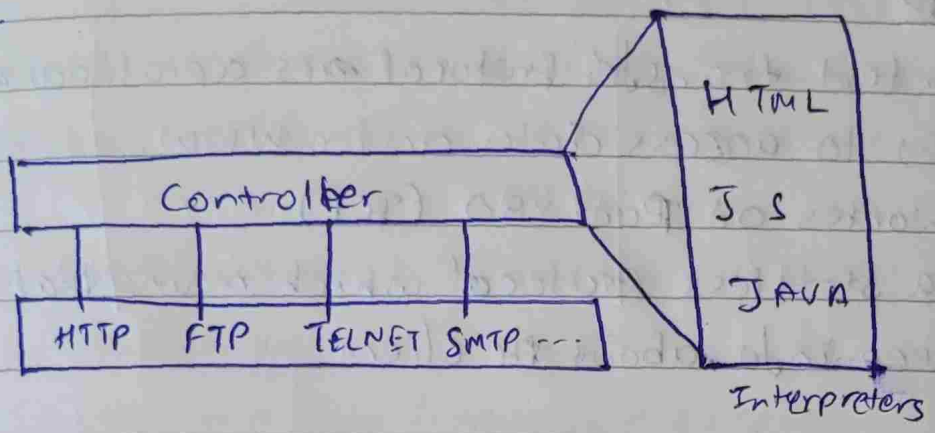
→ A browser consists of a controller, client program & interpreters. A server stores web pages

→ It can be classified as:

- Static: A static doc. is one in which the contents are fixed & stored in a server
- Dynamic: A dynamic web doc. is created by server on at a browser request.
- Active: An active doc. is a copy of a program retri by client & run at the client site.

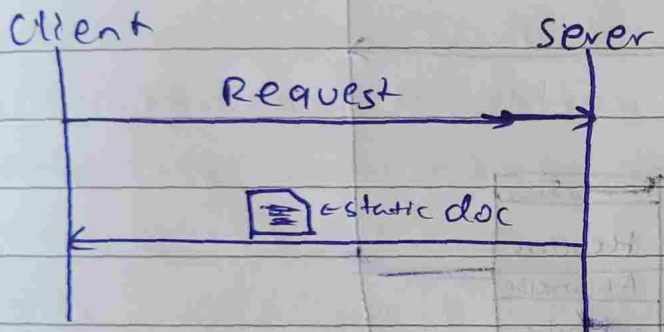


→ Brower

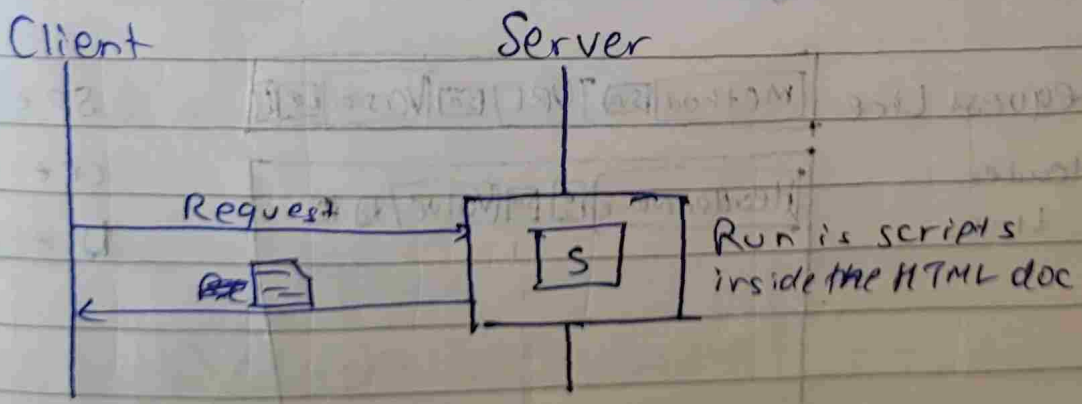


Static / Dynamic / Active

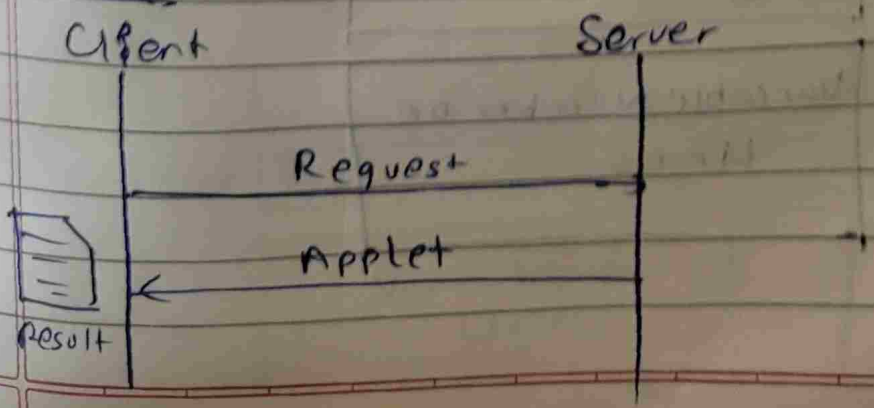
- Static



- Dynamic



- Active

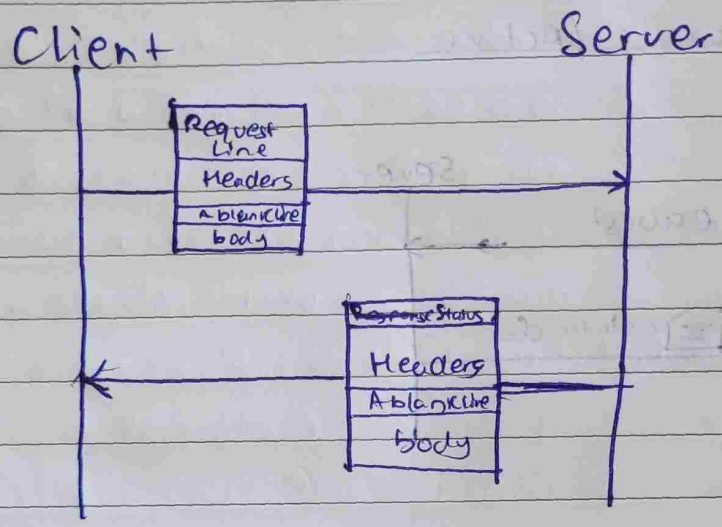


HTTP

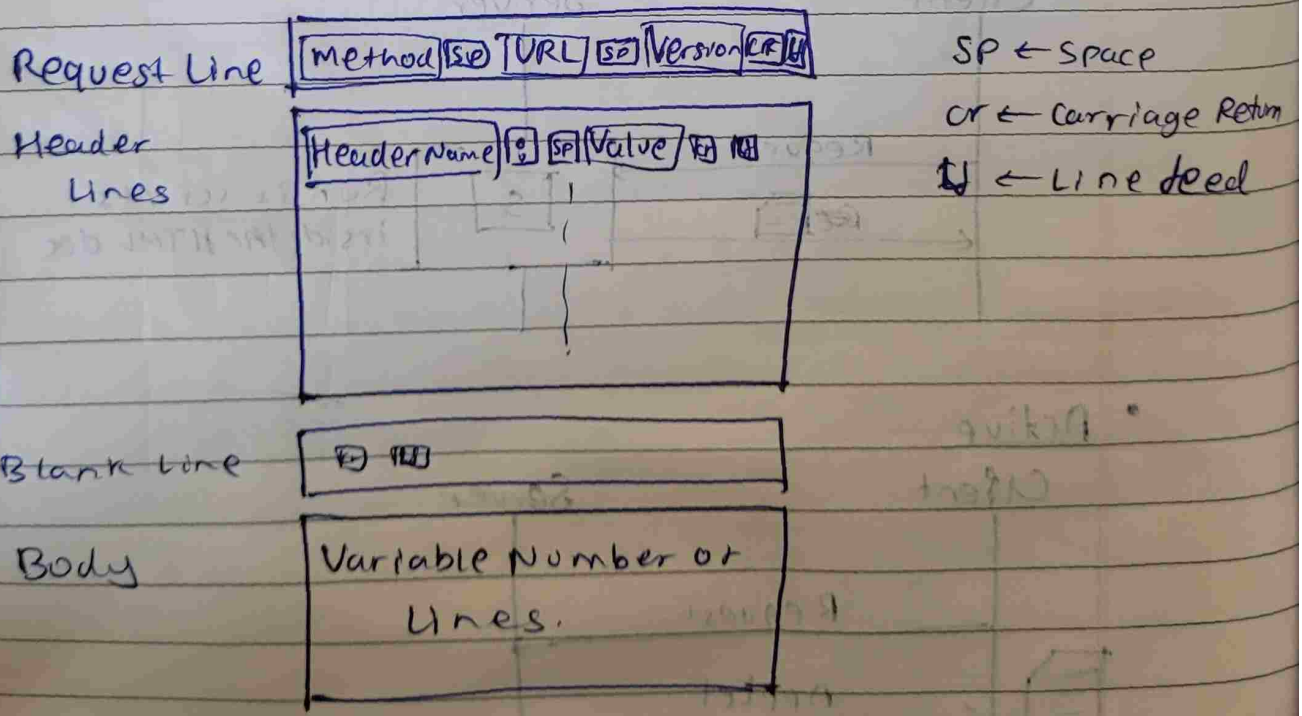
→ Hypertext transfer Protocol is a protocol ~~mainly~~ used mainly to access data on the WWW.

→ It works of Port 80 (TCP)

→ It's a stateless protocol which means that the server does not keep info about the client.



→ Request message



- Request line

Consists of three parts each seperated by a space

~~URL~~, METHOD, URL, version.

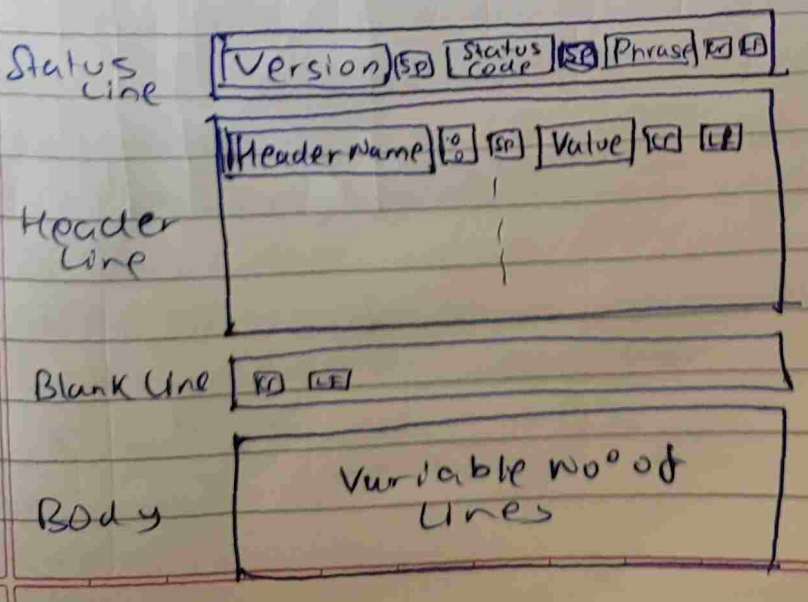
Methods

a) GET → Request DOC form server
b) HEAD → Request info about a doc but not the doc itself
c) POST → Sends some info from client to server
d) PUT → Sends a doc from server to client
e) TRACE → Echoes the incoming request
f) CONNECT → Reserved
g) DELETE → Remove the web page
h) OPTIONS → Enquires about available options

- Header Line

- Each header line sends additional info from client to server.
- eg, the client can request that doc. be sent in a special format,
- Consist of a header name, a colon, a space, and a value

→ Response Message



- **Status Code & Phase**

a) 100 → continue	e) 202 → Accepted	i) 404 → Not found
b) 101 → Switching	f) 204 → No content	j) 500 → Internal server error
c) 200 → OK	g) 400 → Bad request	k) 501 → Not Implemented
d) 201 → Created	h) 403 → Forbidden	l) 503 → Service unavailable

→ **Non Persistent**

- ~~It is~~ In a non persistent connection, one TCP connect is made for each request/Response
  1) Client opens a TCP connection & sends a request
  2) Server sends a response & closes the connection
  3) The client reads the data until it encounters an end-of-file marker & then closes the connection.

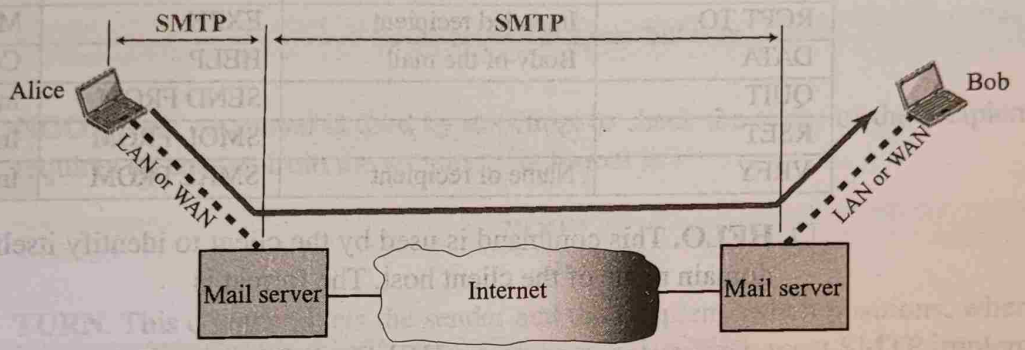
→ **Persistent.**

- If leaves the connection open for more request to respond to

23.3 MESSAGE TRANSFER AGENT: SMTP

The actual mail transfer is done through message transfer agents (MTAs). To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called **Simple Mail Transfer Protocol (SMTP).** As we said before, two pairs of MTA client-server programs are used in the most common situation (fourth scenario). Figure 23.8 shows the range of the SMTP protocol in this scenario.

Figure 23.8 *SMTP range*



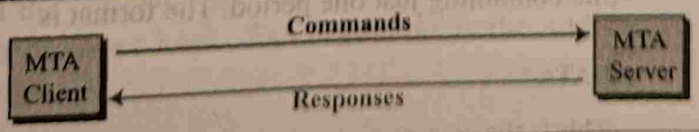
SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver.

SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We will discuss the mechanism of mail transfer by SMTP in the remainder of the section.

Commands and Responses

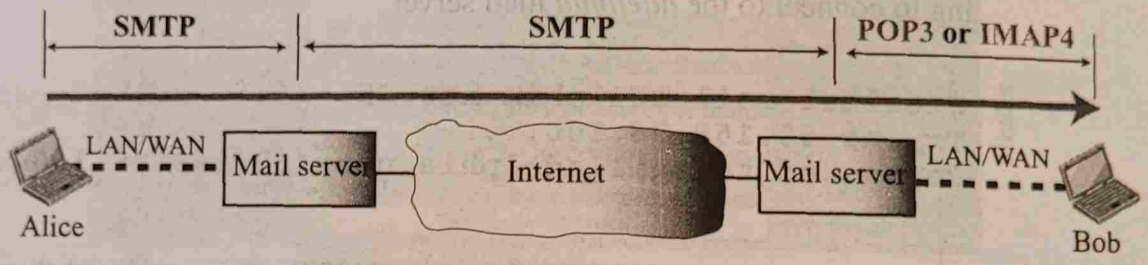
SMTP uses commands and responses to transfer messages between an MTA client and an MTA server (see Figure 23.9).

Figure 23.9 *Commands and responses*



Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.

Figure 23.13 *POP3 and IMAP4*

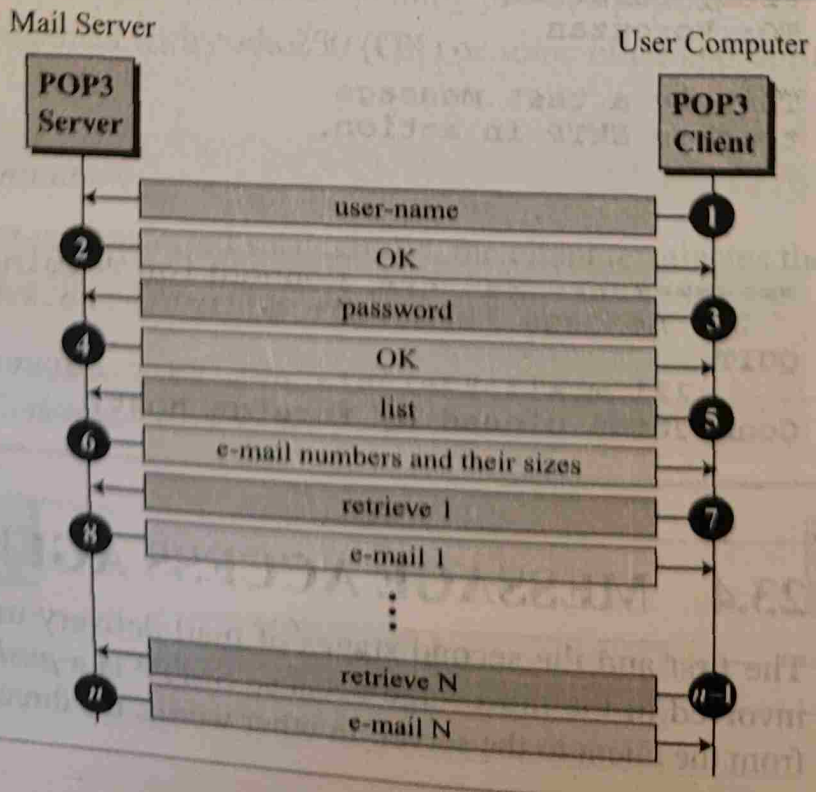


POP3

Post Office Protocol, version 3 (POP3) is simple and limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

Mail access starts with the client when the user needs to download its e-mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one. Figure 23.14 shows an example of downloading using POP3.

Figure 23.14 *POP3*



POP3 has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying. The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

IMAP4

Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4).** IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.

POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.) In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides the following extra functions:

- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.