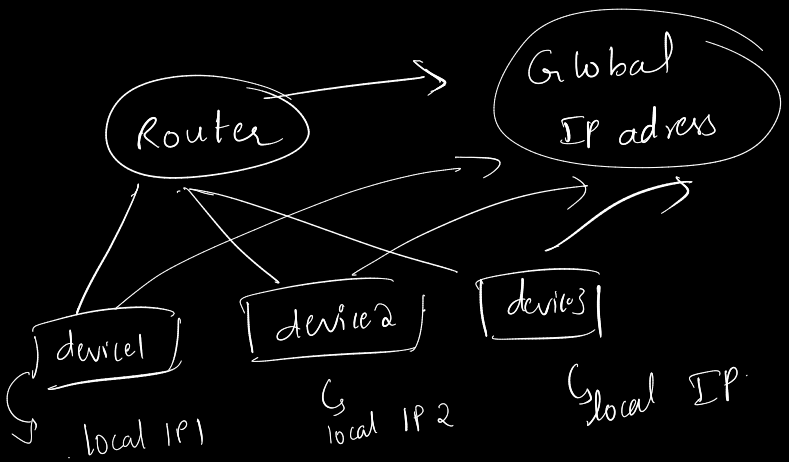
IP Address:



- Router assigns local IP using DHCP (Dynamic Host Configuration protocol)
- say its D1 made a request to google the router decides it by NAT (Network Access Translator)
- Which device is decided by IP adress and which application to send is decided by port number.
- PORT number is a 16-bit number. Total ports possible are 2^{16}
- Web pages use HTTP protocol.

Say some rules set by very cool people at Internet society.

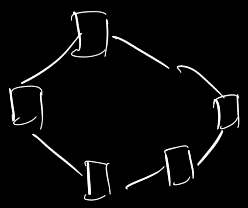
- 0-1023 ⇒ Reserved ports for HTTP stuff.
- 1024 - 49152 ⇒ Applications
-

Topology:

1) Bus Topology :

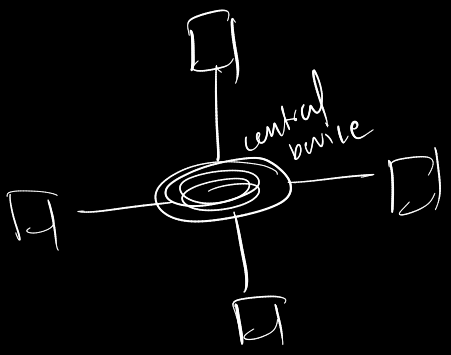
↳ if link break all are disconnected

2) Ring Topology



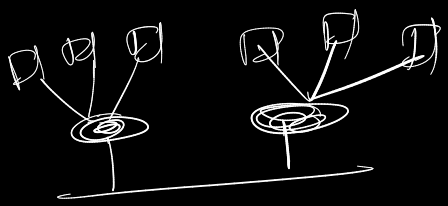
→ unecessary transfer
→ if one link break its useless

3) Star Topology



→ central device stop the connection is gone.

4) Tree (Star + Bus)



5) Mesh → every single comp connected to every other comp

⇒ Expensive
⇒ less scalability

Structure of the Network:

⇒OSI MODEL (Open System Inter Connection):

- There are 7 layers

- Example: A whatsapp msg.
  1) Whatsapp se msg sending (Application layer)
  2) Application layer → Presentation layer
  3) Presentation Layer converts ASCII to something else (Encryption + compression + translation)

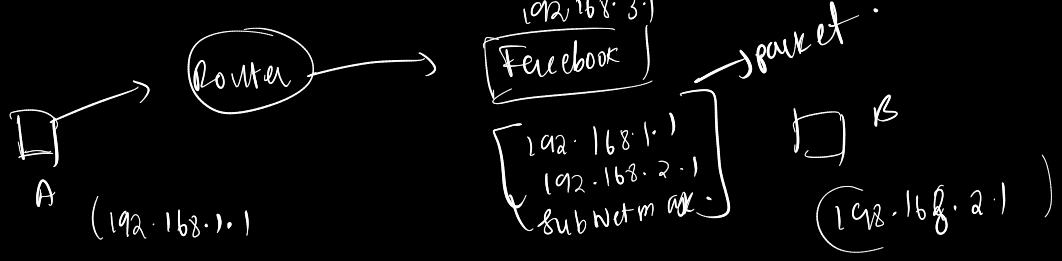
Application
↓
presentation
↓
Session
↓
Transport
↓
network
↓
Data - Link
↓
Physical

4) Session layer ⇒ Authentication, Authorisation stuff. It just establishes & terminates sessions

5) Transport Layer ⇒ i) Segments (divides into small parts)
   ii)

6) network Layer ⇒ sends to computer on other networks (say the router lives in the layer.
   IP adressing of senders & recievers and forms IP packet and also routing

7) Data Link ⇒ recieves data packet from network layer.



8) Handles Logical addressing & physical adressing

TCP/IP Model: (Pratically more used in real world)
It contains lesser steps compared to OSI Model.

1. Application
2. Transport
3. Network
4. Datalink
5. Physical.

1) Application Layer:

i) User interaction
ii) Where: devices
iii) Protocols
iv) Client- Server Architecture.

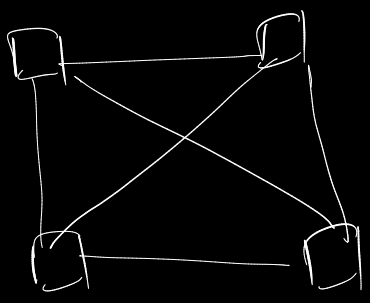
⇒ Collection of servers are called data centers

⇒ Client- Server Architecture

client ⇌ server

⇒ Peer to Peer (P to P) Architecture:

Eg: Bit-torrent



Key advantage
1) scalable
2) Decentralized
3) Every Comp acts as client & Server.

⇒ Protocols:

* **TCP/IP** :
  - ☆ HTTP
  - ☆ DHCP.
  - ☆ FTP (File transfer Protocol)
  - ☆ SMTP (Simple Mail Transfer Protocol) (for sending EMAIL)
  - ☆ POP3 & IMAP (Recieving EMAIL)
  - ☆ SSH (To login to someone's comp using terminal)
  - ☆ UDP (stateless ⇒ Data maybe lost)

* **Sockets** : Used to send msg's from one comp → other.

* **Ports** : Tells which application.

say we know to send data chrome but to which tab will be determined by "Ephermeral Ports". It basically assigns itself random ports.

* **HTTP**: It is a client server protocol. It tells how to request data and how to send data. This is basically a application Layer protocol. HTTP uses TCP (Transmission Control Protocol) inside it.

(stateless)

(at transport layer)

☆ Cookies:

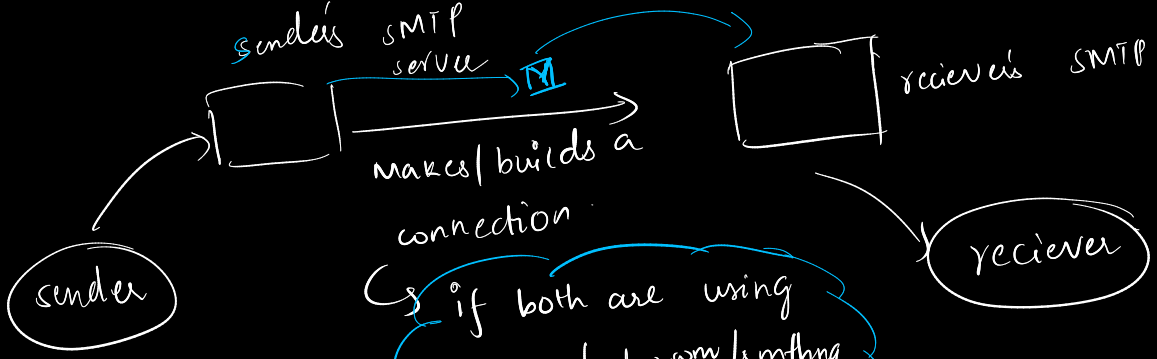
A unique string stored in browser. Login data is stored in cookies and then it is sent in headers everytime it makes a request to backend.

⇒ How Email Works:

Application Layer : SMTP, POP3 → how to send & receive

Transfer : TCP protocol.

Eg:



- Client Server Authorize → POP server
  ← Transaction (download, delete all other than sending)

so basically gmail has two seperate servers SMTP & POP

⇒ IMAP : Allows to use emails on multiple devices .. so the Syncing thing on diff devices.

⇒ DNS (Domain Name System):

Used to find the IP address associated with a URL.

mail.google.com → Top level domain
(sub-domain, second level domain)

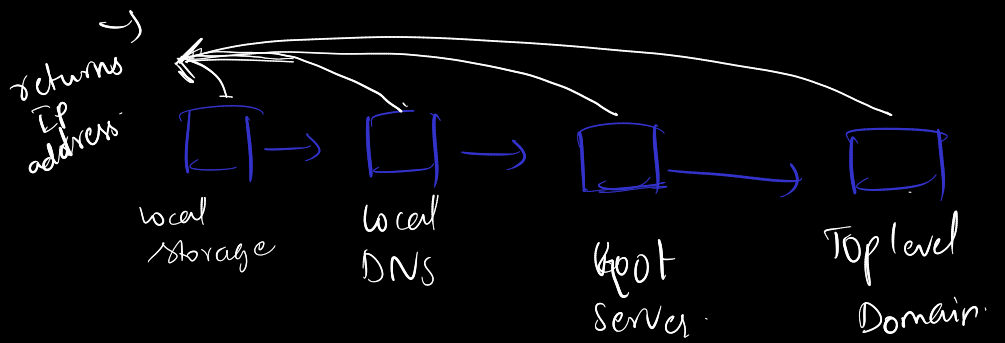
☆ Root DNS Servers: (Top level domains)

→ First point of contact
→ Stored / Maintained by ICANN.

IP Searching Process:

→ First checks in the devices storage
→ Look it up in the local DNS server (ISP, Router)

They know every website we visit even in incognito mode



2) Transport Layer:

this transportation is done by network layer.

⇒ Transport layer just transports data from network layer to application within the computer

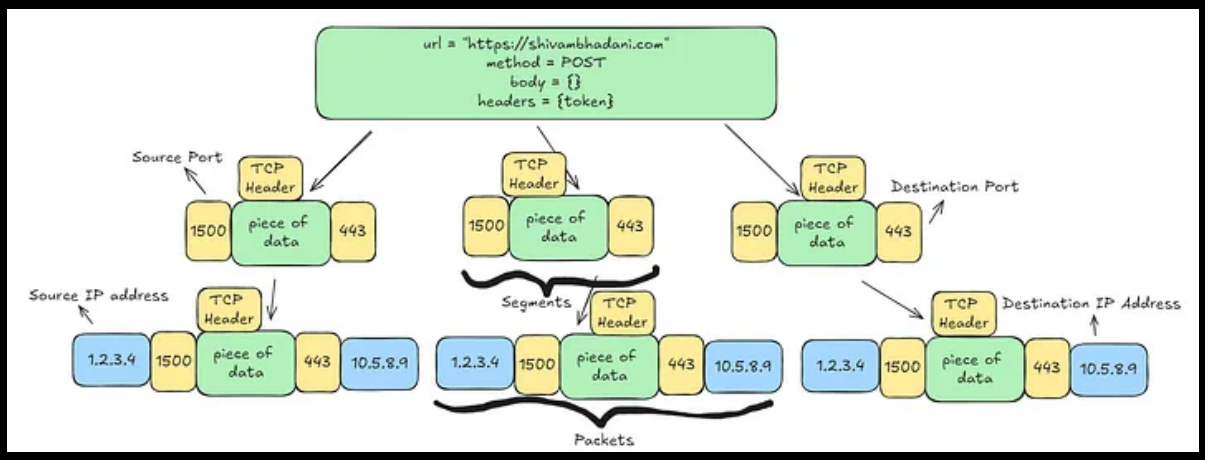
☆ How axios.post() works undahood.

→ when we send the data along with URL it is first broken into small units ⟶ Y? Cuz we can't transfer large data at one go through cables, wires or wifi

→ Each unit needs the recievers port and senders port data for the transmission of data to be successful.

→ Each unit is called a Segment.

segment + IP adress of source & destination ⟹ Packet



- Every packet sent by sender will be recieved. To make sure that happens we use acknowledgement. So kind of like replying I've recieved packet 1, packet 2, etc. and I have not recieved. too works

Note: TCP works on 3 way handshake

→ Sending data over TCP has 3 steps:

i) Connection Establishment (3 way handshake)

ii) Transfer data

iii) Connection Termination (4 way handshake)

☆ 3-way Handshake:

FLAG bits: SYN, ACK, SYN-ACK, FIN,

(kind of like switches)

SYN = 1 (set)
ACK = 0 (unset)

i) Client side sends SYN (synchronize) bit.

ii) Server side SYN-ACK (Acknowledge)

iii) Client side ACK

Analogy: Client: I want to establish a connection will initialise all my mugs to X

server: I acknowledge it. I will initialise all mug to Y.

Client: Accepted.

→ A random number (Seq) is generated at the start and from the messages are numbered by +1.

→ Y: random number → solves confusion on multiple connections.
→ Extra layer of security

☆ 4 way Hand Shake:

- client sends a FIN to indicate it wants to close connection
- server ⇒ ACK
- server ⇒ FIN
- Client ⇒ FIN

Congestion:

When a network is overloaded by large data being transferred by multiple devices, then there might be packets loss, time delay, or the breakdown of entire network sometimes.

✲ How does TCP handle it:

It uses some jargon (algo's) to figure out how much load can the network take and adjust the transmission speed accordingly

There are 4 phases in TCP congestion control:

1) Slow Phase
2) Congestion Avoidance
3) Fast retransmit
4) Fast Recovery.

1) Slow Phase:

In this the sender sends the data by gradually increasing the amount of data until, he encounters congestion.

Congestion Window: (CWD)

TCP maintains a congestion window that tells the sender how much data he can send before ACK from reciever

- Initially CWD is set to 1 MSS (Maximum Segment Size)
- It ↑es it for every ACK recieved.

Congestion Avoidance:

- Once the CWD reaches threshold value. It stops growing exponentially to avoid congestion. It increases linearly by a small amount.

2) Fast Retransmit:

- If the network ⇒ congested. then some packets will be lost. If the sender recieves 3 ACKs (instead of 2 ACKs) that indicates a packet is lost.
- Instead of raising an error which would take more time, TCP retransmits the missing packet. This process ⇒ fast retransmit

3) Fast Recovery:

- To retransmit the missing packets the TCP does not go back to the slow start phase.
- Instead it cuts down the MSS to MSS/2 ⇒ Then it linearly until senders start receiving normal ACKs
- After that it goes back normal congestion avoidance Mode.

Sockets :

- Socket is an endpoint for transferring (sending/recieving) data over the network. It is combined of

  1) IP adress
  2) Port
  3) Transfer Protocol (In web's its TCP generally)

- What it does ⇒ attaching port (TCP segments) + IP address (packets)
  + establishing 3 way handshake

Note: Socket ≠ Websocket