COMPUTER NETWORKS - 2

[Will start at 9:05 PM]

Agenda

1. Client Server v/s P2P
2. Sockets
3. Ports
   - Ephemeral ports

[Socket v/s Port]

4. HTTP
5. DNS

(TCP/IP Model)

Intro to Appⁿ layer

1. User Facing
2. Generate Data or consumes data
3. Diversity of use cases over internet
   - VoIP
   - SMTP
   - HTTP

Appⁿ layer Architectures

Client Server Architecture

→ 2 diff appⁿ running at diff places, complimenting each other

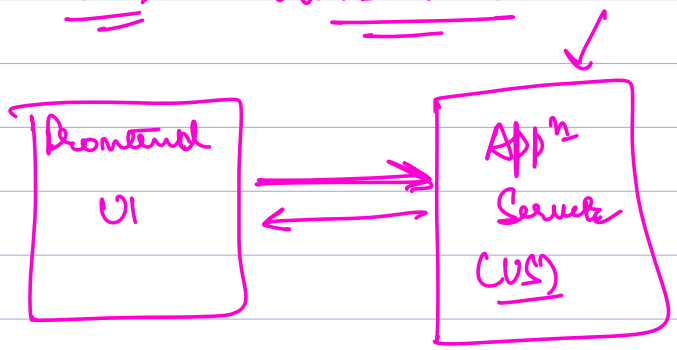
Peer to Peer Architecture

→ Same appⁿ is running across all machines.

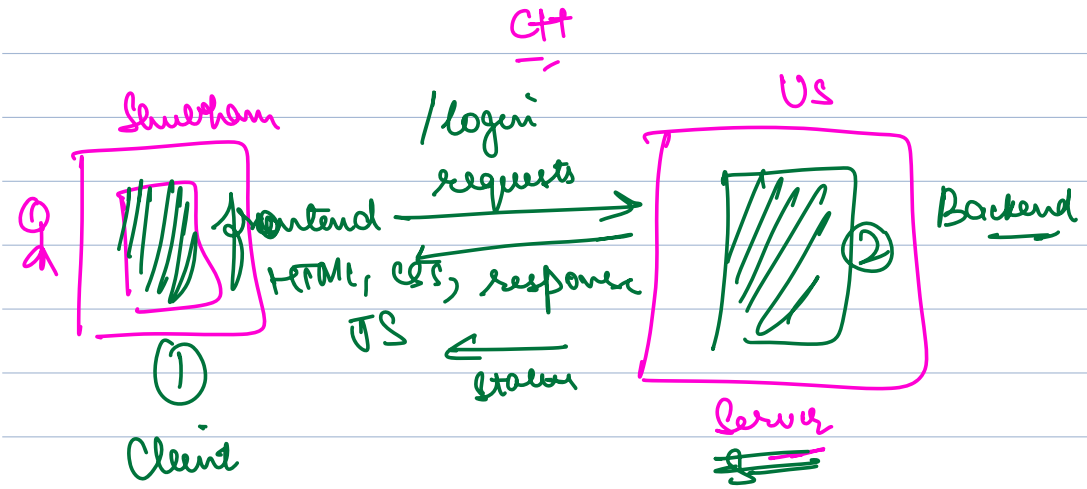
→ Client , Server

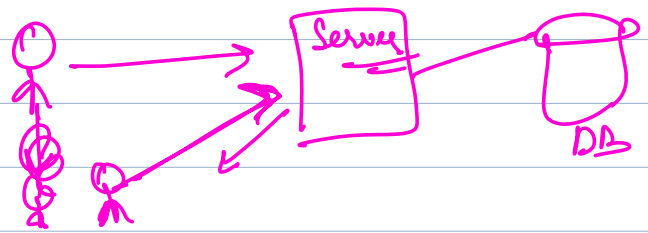
Client : requests data
: Consumes content

Server : Owns the data



Java ⇒ Spring Boot
Python ⇒ Django
JS ⇒ ExpressJS





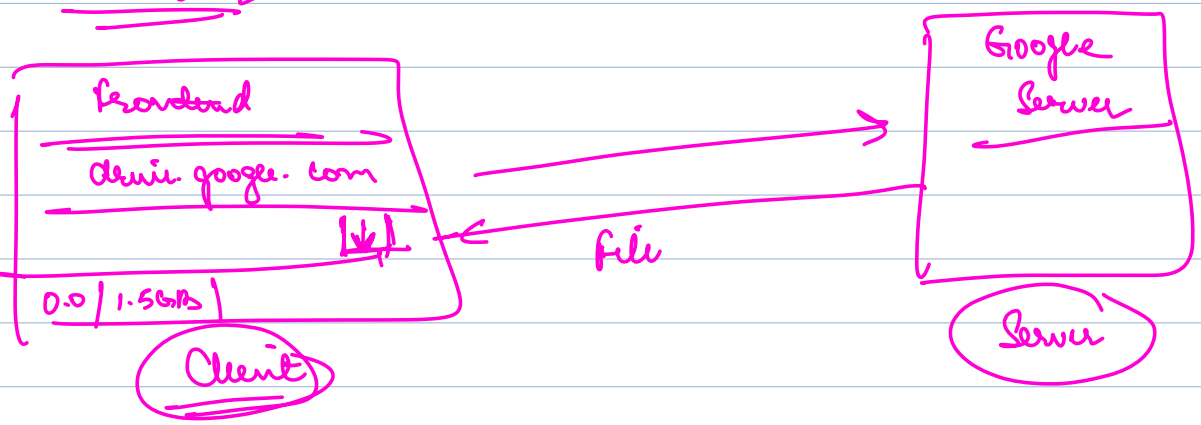
Peer to Peer

torrent

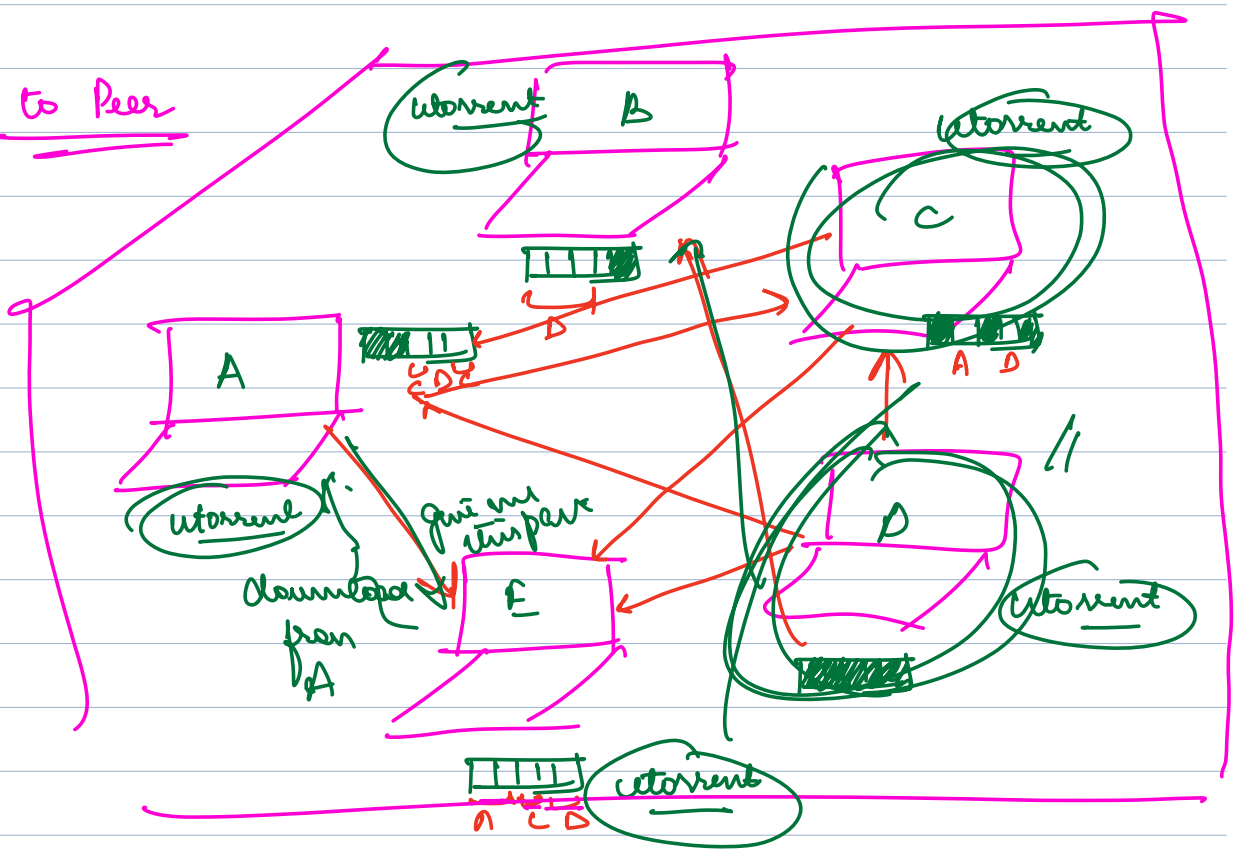
1.5GB Movie

Downloading a 1.5 GB Movie

Client Server



Peer to Peer

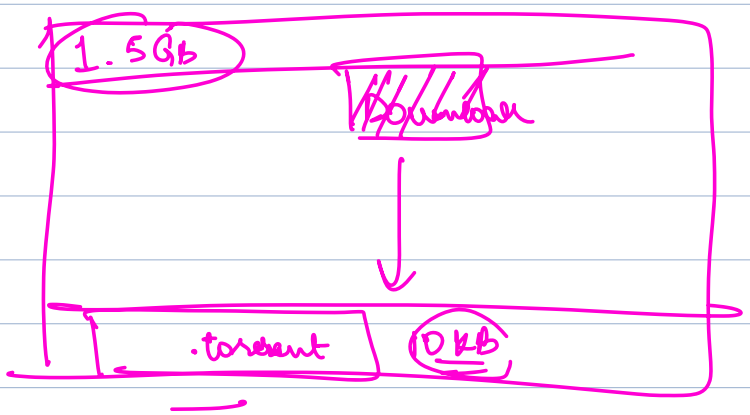


→ torrent knows across the world which person has what part of file

→ every peer at the same time is downloading as well as uploading (requesting as well as serving)

→ No division of resp. every machine is same.

Step 1



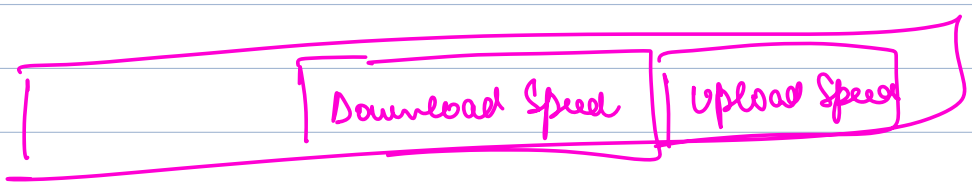
Govt banned

⇒ file contains list of tracker servers

↓

{ Store info about which peer has what part of the file

→ Each peer connects directly to peer and requests a part of data from them

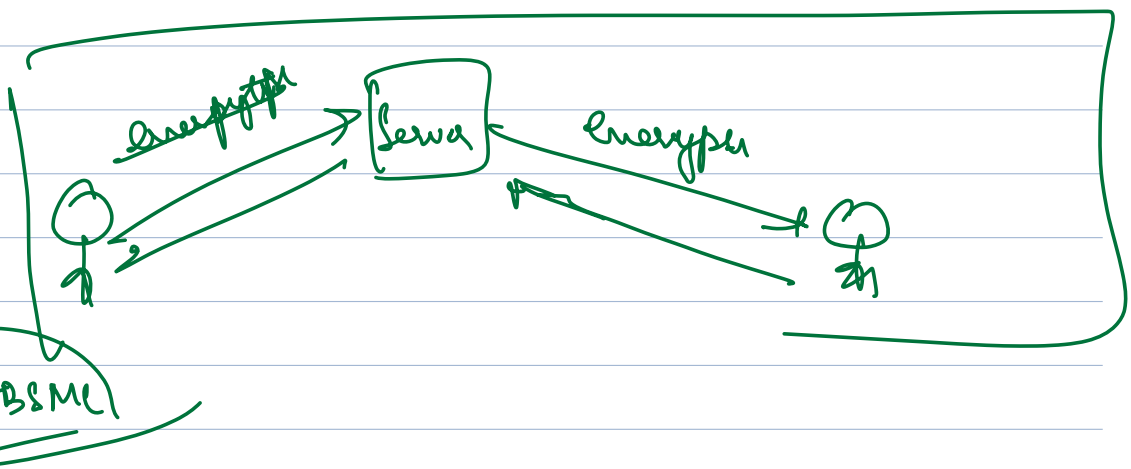


Leecher → Only Leeching

Seeder → only contributing

Eg: Bitcoin, Skype

Darknet . onion . tor



SOCKETS

→ **power socket**: allows an appliance to connect to electricity

→ **network socket**: allows an application to connect to internet (send data / receive data)

Application.java {

File → file interface allows an app^x to talk to disk

File f = new File (path to)

f.open()

}

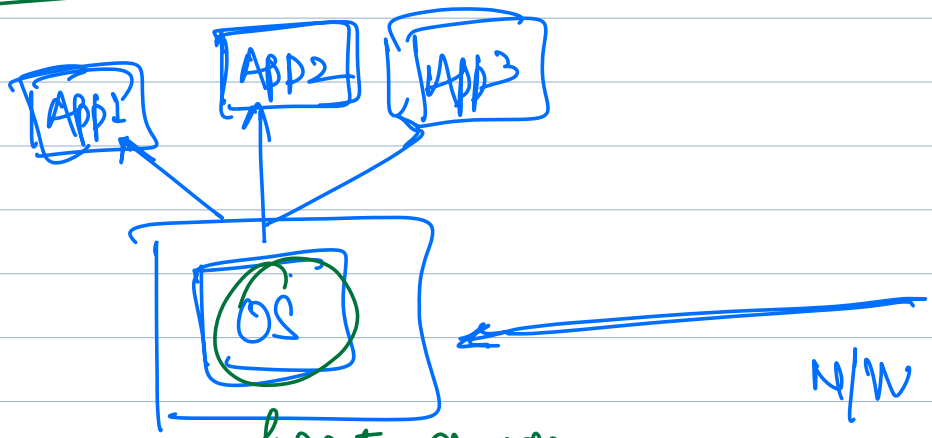
⇒ Socket interface allows an app^x to talk to a network

- → Send data ✓
- → Receive data ✓
- → Connect to a peer / server / client

fetch() | http.client | request | Axios.

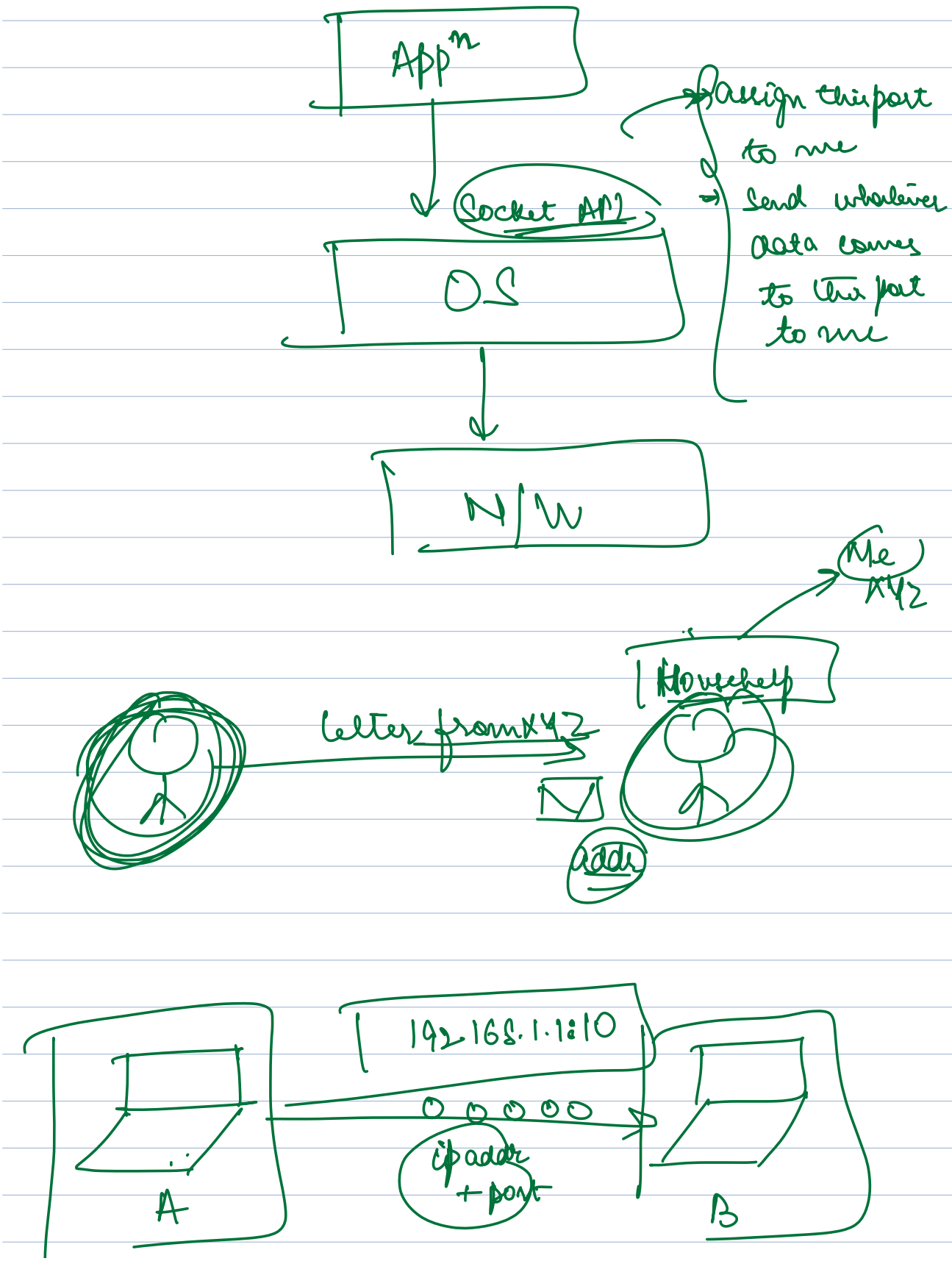
↓ | ↓ Socket | ↓ Socket | ↓ Socket

⇒ Socket is an OS level API

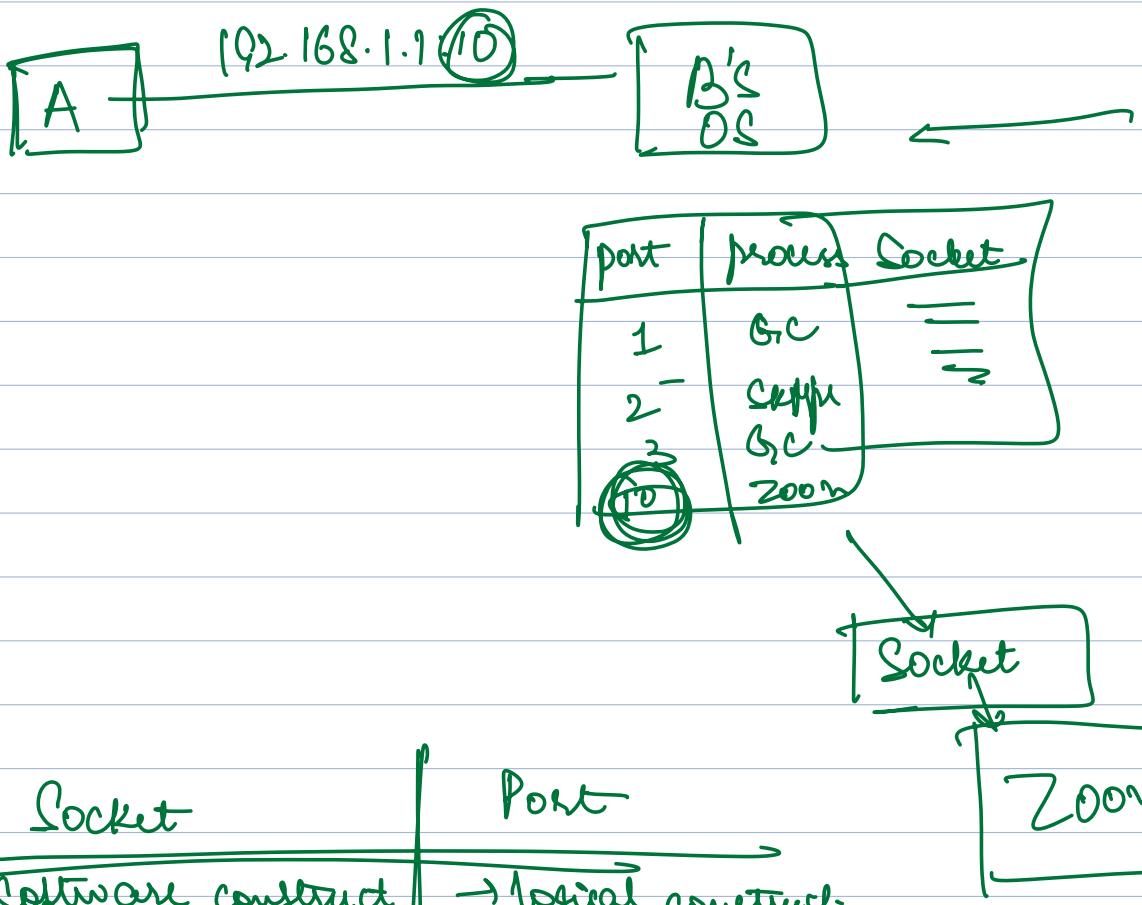


has to decide what app^n has to get what data

⇒ using Socket API, we tell OS
to assign me a specific port
⇒ tell OS to send request to another server

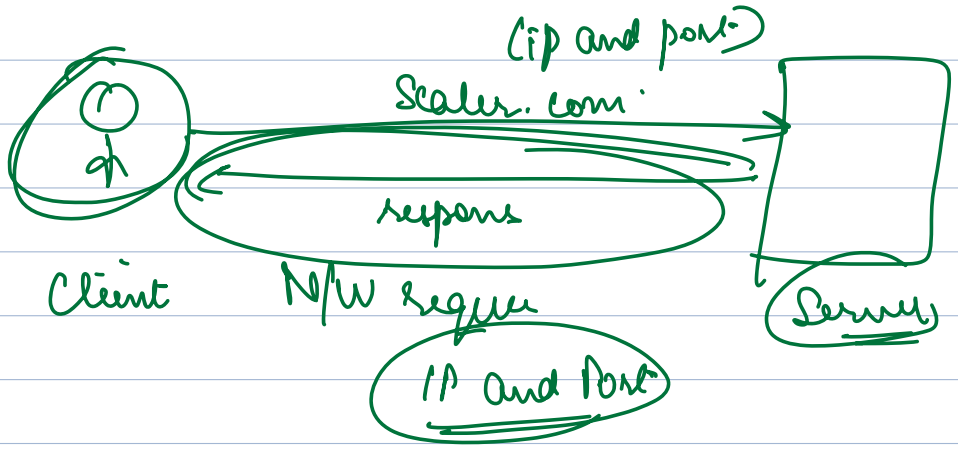

App^n
Socket API
OS
N/W
Assign this port to me
send whatever data comes to this port to me
Me XYZ
Household
letter from XYZ
addr
192.168.1.1:10
ip addr + port
A
B

v



Socket	Port
→ Software construct	→ logical construct.
→ No one outside knows about Socket	→ People outside know about your port number to send data to you.

⇒ Any app^n that connects to a n/w must have a socket



⇒ Port that is randomly assigned to a client so that it can talk to N/w and receive responses ⇒ Ephemeral Port

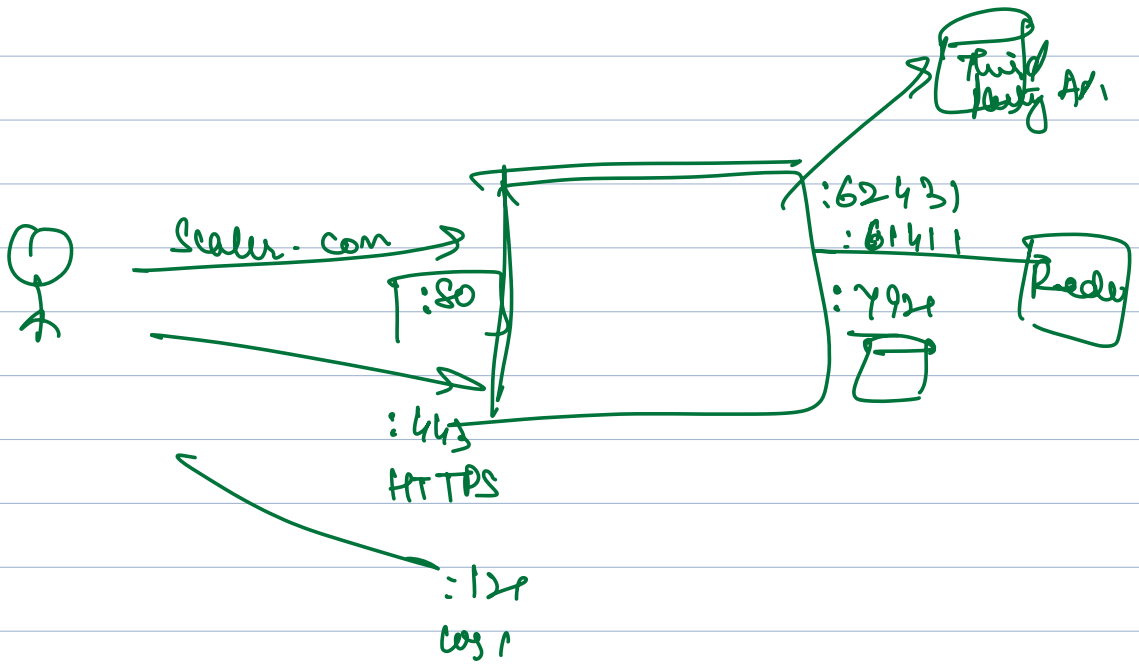
(1201)

A ⇒ Socket → OS ⇒ N/W

1:1

1 Socket == 1 port

CN4: Implement Client Server using Sockets



HTTP Protocol

HTTP → Hypertext Transfer Protocol

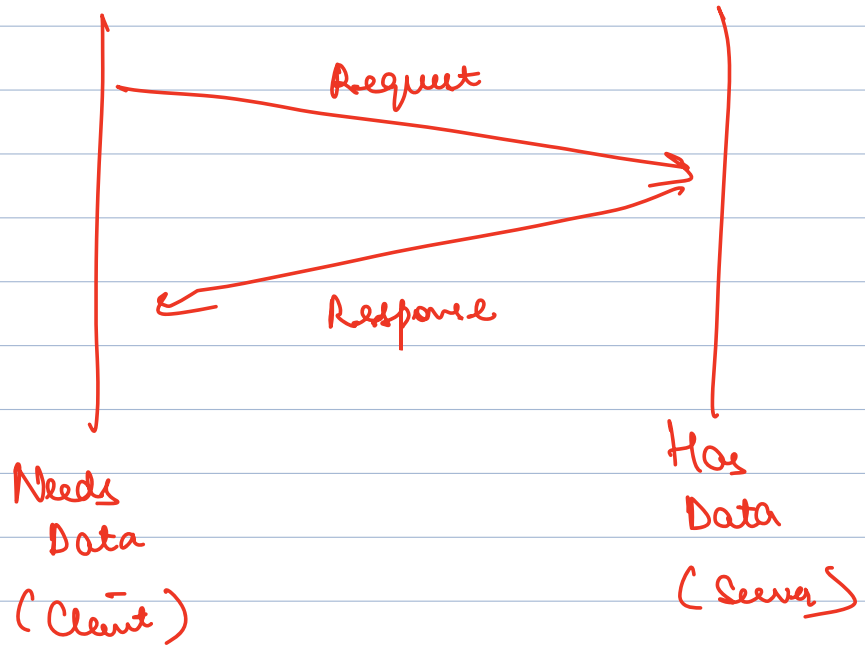
↳ Not just raw text
↳ "more than" text

HTML → Hypertext Markup Language

HTTP

→ Appⁿ layer protocol
→ Client Server Protocol
↳ Request response protocol

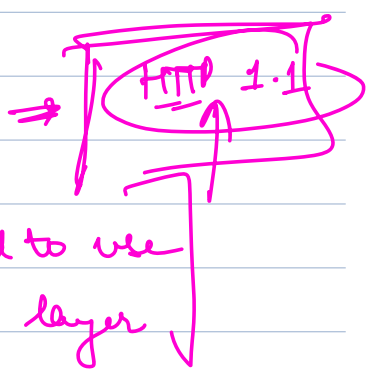
BE FE



→ defines how a client / server should communicate with each other

App^L
Transport
N/W
Data link
Physical

⇒ HTTP, FTP, SMTP, IMAP, POP
⇒ TCP, UDP
⇒ I/P = Internet
✗
✗

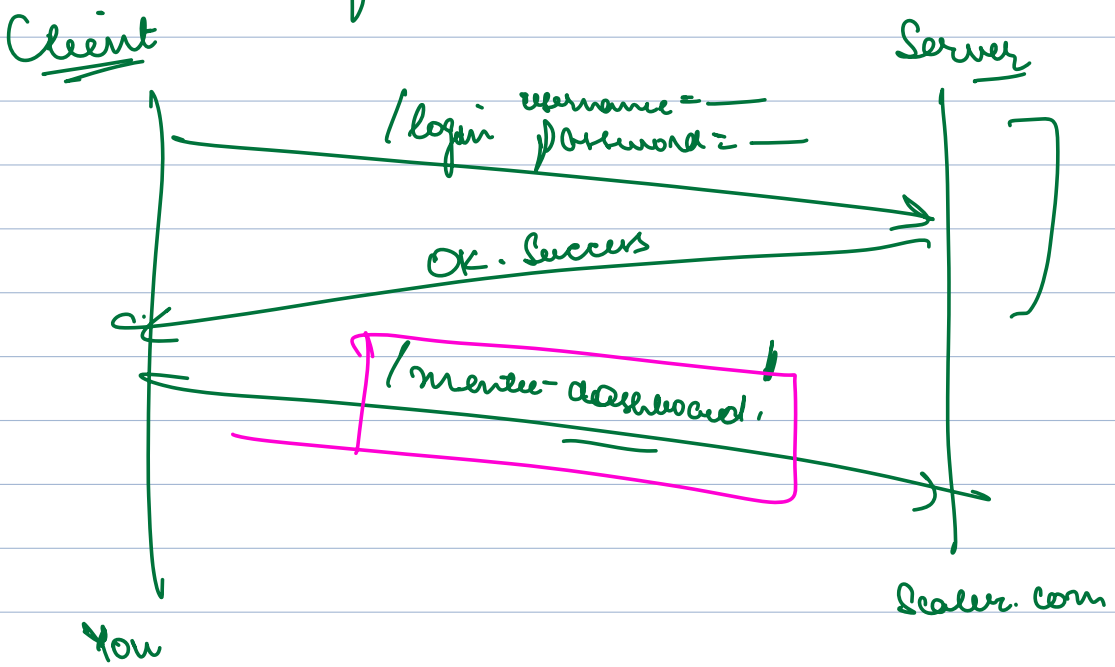


⇒ HTTP protocol before HTTP 3 used to use TCP protocol as its transport layer
⇒ HTTP3 ⇒ UDP

Stateless Protocol

→ By default every HTTP request is complementely independent of previous HTTP

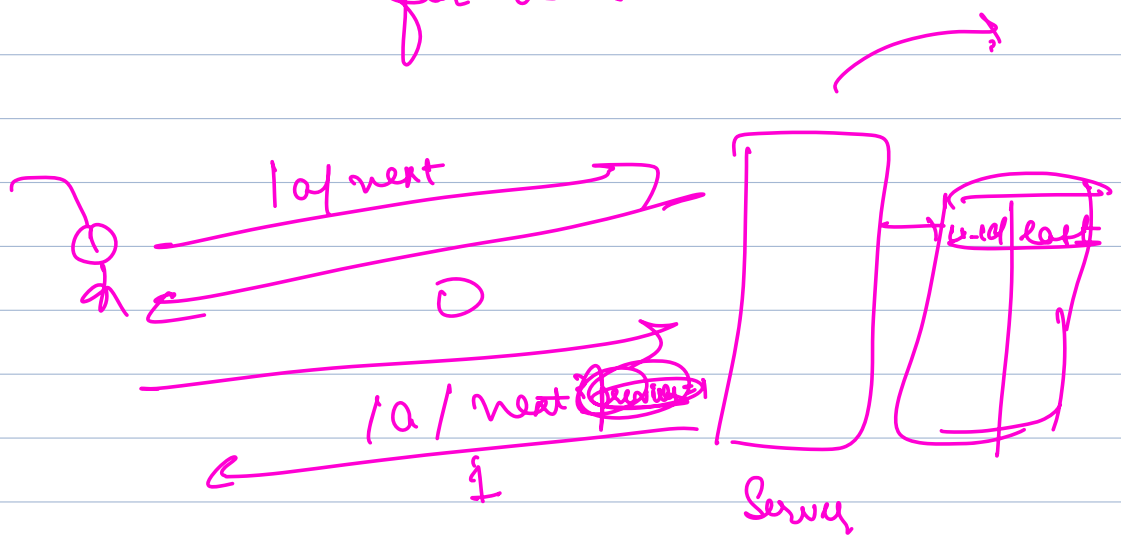
no data that was sent/ recieved earlier will be carry forwarded.



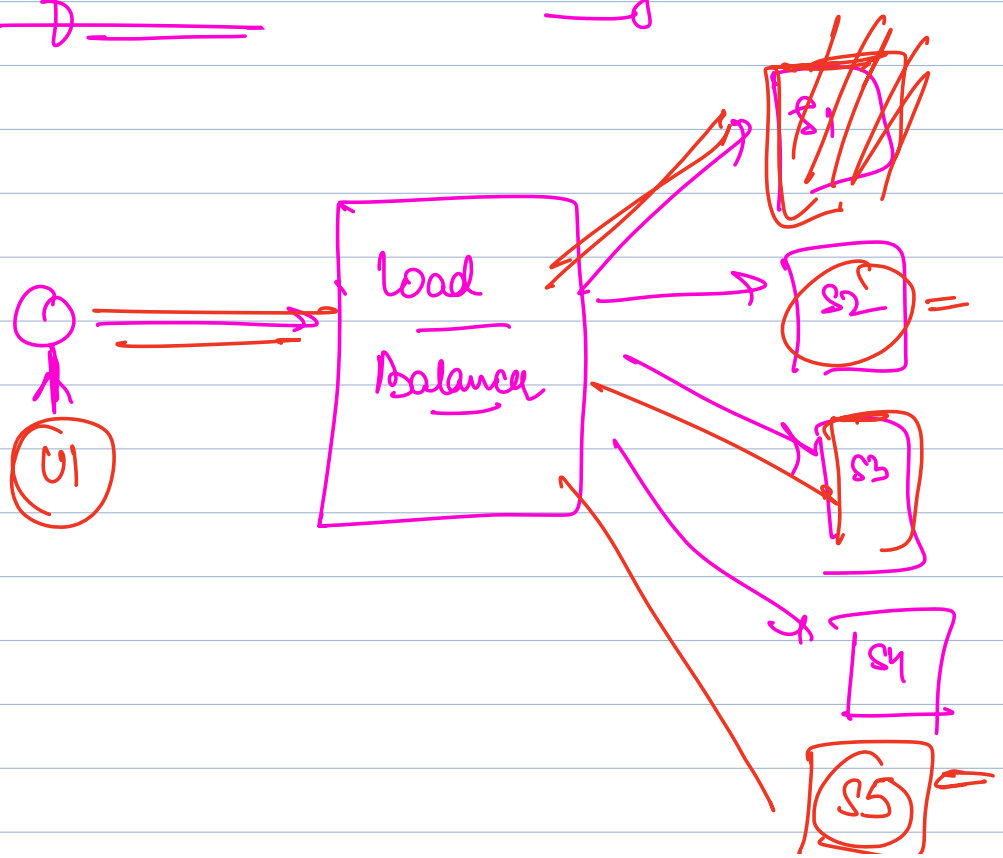
By default, no info about prev things is sent in future request

/articles/next

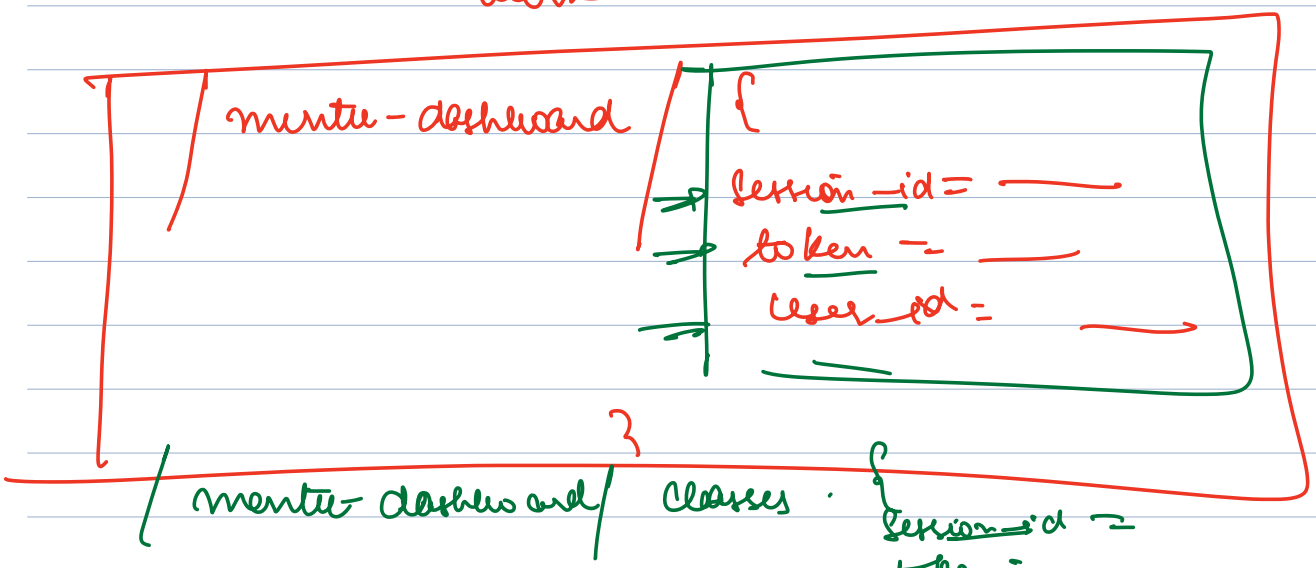
API returns the next article for a user



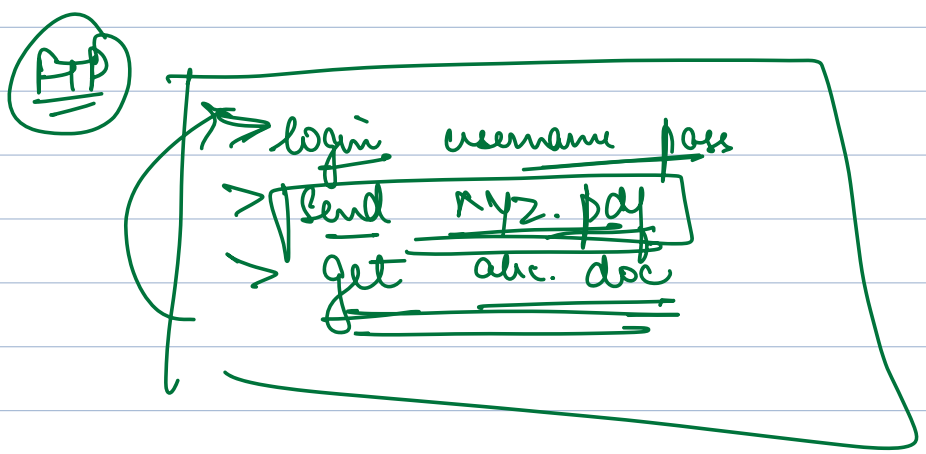
Benefit of stateless ⇒ Scalability



→ every request is self sufficient (has all info needed by it to do its work)



→ whatever is needed by a server to serve a request needs to be explicitly sent in the req



HTTP considers everything as an Object

→ Object → file

→ everything is a file at some loc^n
  → HTML
  → JS
  → PDF
  → IMG
  → JSON

→ every request is for a file that is there at a particular loc^n

loc^n ⇒ identified using URL

HTTP ⇒ 80
HTTPS ⇒ 443

URL ⇒ Uniform Resource Locator

if no port specified, default port of that protocol.

protocol :// hostname/ip : port / path of file ? params

https://scaler.com:443/mentic-dashboard/todos

?query="Book"

HTTP

- → Client Server / Request - Response
- → Stateless
- → URL (everything is a file)
- → Specifies how client should req and server should respon
- → uses TCP at transport layer
  (HTTP3 → UDP)