Lecture 1: Introduction to HLD

System Design - divided in two parts

HLD (focuses on-)
- Architecture
- Tech stack → MERN, SPRINGBOOT, DJANGO
- Cost Optimization
- Database → SQL, NOSQL, GraphQL
- USER → million of users should handle concurrently
  - We also have to estimate no. of users that can come so that they are handled ⟹ How do we estimate? ⇒ using envelope estimate

→ main focus is on architecture
→ crucial phase as it decides what to choose for developing applications based on trade offs and requirements

LLD (focuses on)
- Coding Part
  - OOPS
  - design principle (SOLID)
- Clean code
- Loosely coupled code

HLD focuses on architecture of Application

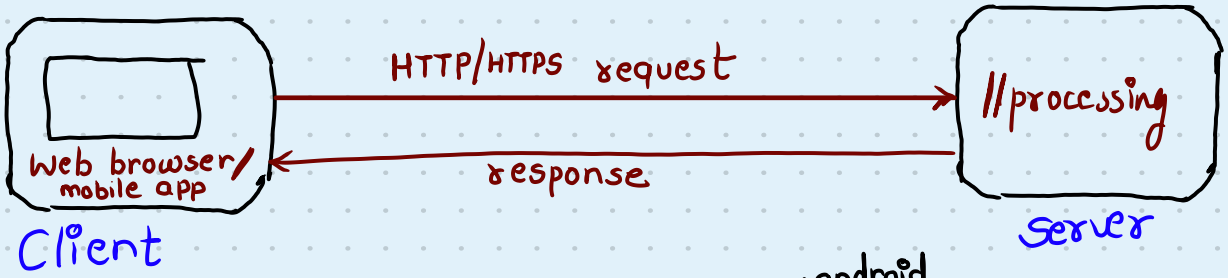
Software / Application includes-
- Website
- Mobile Application
- Software

→ focus of HLD (related to internet)

- HLD do not focus much on s/w as suppose Adobe Photoshop is software it uses your computer resource like ram & memory and it depends on user whether to update to its newer version or not & there is no multiple user problem
  - This kind of s/w has more focus on LLD rather than HLD

Just for Info: How you get Response?

Request Response Model / Client-Server Model



- Client could be any frontend as - Mobile App (android, ios) and web browser.

Jiske through aap request bhej skte ho using HTTP/HTTPS request ke through to server par, aur after processing server respond karta hai & that's called request - response model

- This will work fine till limited users access our application
  - Lekin ek time pe jab no. of users (millions of user approx) ek saath request karte hai server pe toh server crash ho jaata hai due to unable to serve all request as count of request are much more than it is able to serve.

- How to handle it ⇒ Distributed System

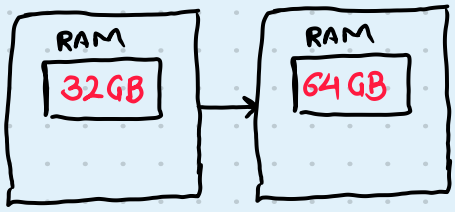
Distributed System?

- refers to a system where components (Server, DB or Application) are spread across multiple network

- Horizontal Scaling Vs Vertical Scaling
  - different methods to handle multiple user by adding or increasing capacity of servers

1] Vertical Scaling

→ increasing capacity of already existing server like previously RAM = 32GB ⟶ RAM = 64GB



disadvantage
- There is limitation to maximum increase of RAM as CPU should be able to handle it
- No increase in computational power
- Good for small application

2] Horizontal Scaling

→ increasing no. of servers to handle multiple user

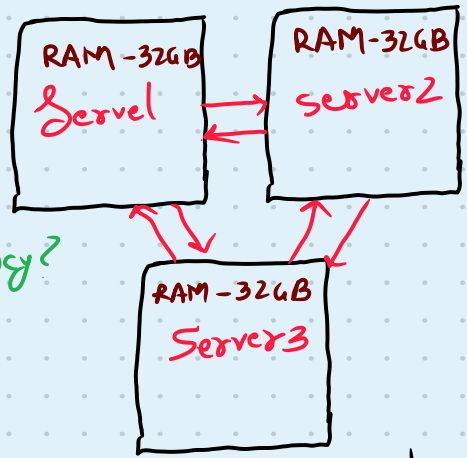
→ Good for large application

→ But How will server interact with each other?

→ How data will be stored in all servers with consistency?

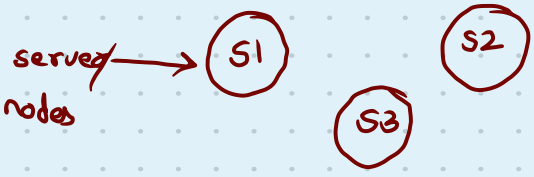
→ How we will decide who will handle which user

- we will answer this qs in coming lectures



- Vertical & Horizontal scaling is base to understand distributed systems

- In HLD ⇒ server are known as nodes



soln will be discussed later on

- But this creates complexity in our website as server will interact with each other to maintain data integrity
- Distributed system ⟶ complex architecture but solves huge problem

Network Communication

↳ Network communication are rules that both client and server follow to communicate over internet

- Network Protocols → (Rules)

Let's discuss of OSI model (7 layers)

- Application layer →
- Presentation layer
- Session layer
- Transport layer →
- Network layer
- Data link layer
- Physical layer

(we will discuss only these layer's protocols not others as they don't have role in HLD)

Application follows any of these 2 model

- ISO_OSI Model (7 layers) — Each layer has its own protocol
- TCP/IP (5 layer) — Each layer has its own protocol

- Application → HTTP, HTTPS, FTP, SFTP, SMTP, WebRTC, WebSocket

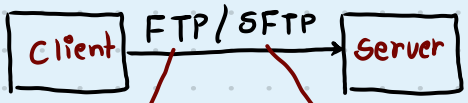
↳ client when communicate with server these rules will be followed depend on request of client

1] HTTP/HTTPS : request over internet Ex:
- load webpage
- open application

HyperText Transfer Protocol

HyperText Transfer Protocol Secured ↳ used when authentication / secure needed things are requested

2] FTP/SFTP → when we are sharing files/pdfs over internet



File transfer protocol

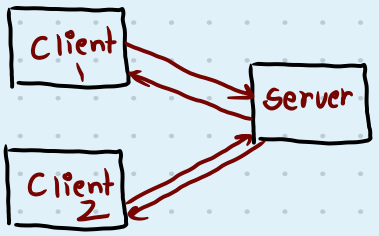
Secured File Transfer Protocol

3] SMTP
↳ Simple Mail Transfer Protocol
↳ used when we are sending mails over Internet

POP3/IMAP → used to receive mail over internet

4] WebRTC (used in application where server can share response without client requesting it

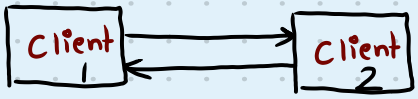
Ex: WhatsApp
- Client1 wants to send message to client2 (can be done throug HTTP)
- server should deliver that message to client2 without client2 requesting for it



∴ WebRTC is used

5] WebSocket - peer to peer communication

↳ no server involved to manage communication



Difference btw websocket and webRTC

websocket ⟶ client directly interacts ⇒ no server in btw

WebRTC ⟶ communication through server

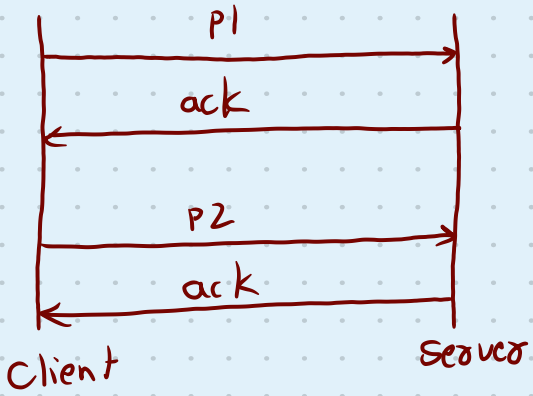
○ Transport Layer ⟶ TCP, UDP

TCP

→ Transmission Control Protocol

→ Reliable

→ slow transmission



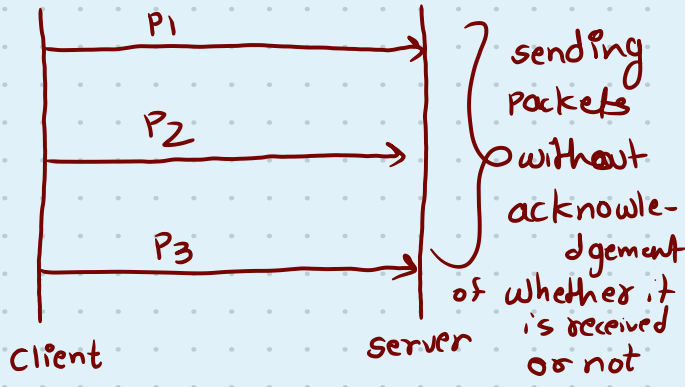
UDP

→ User DataGram Protocol

→ Non-Reliable

→ Fast transmission

Ex: Video streamin



• A message is divided into small packets

DNS

↳ client do not directly communicate with server

- ↳ client first communicates with DNS
- ↳ DNS returns server IP Address
- ↳ Then client requests to that IP Address

address of application where they are hosted

Why DNS ?

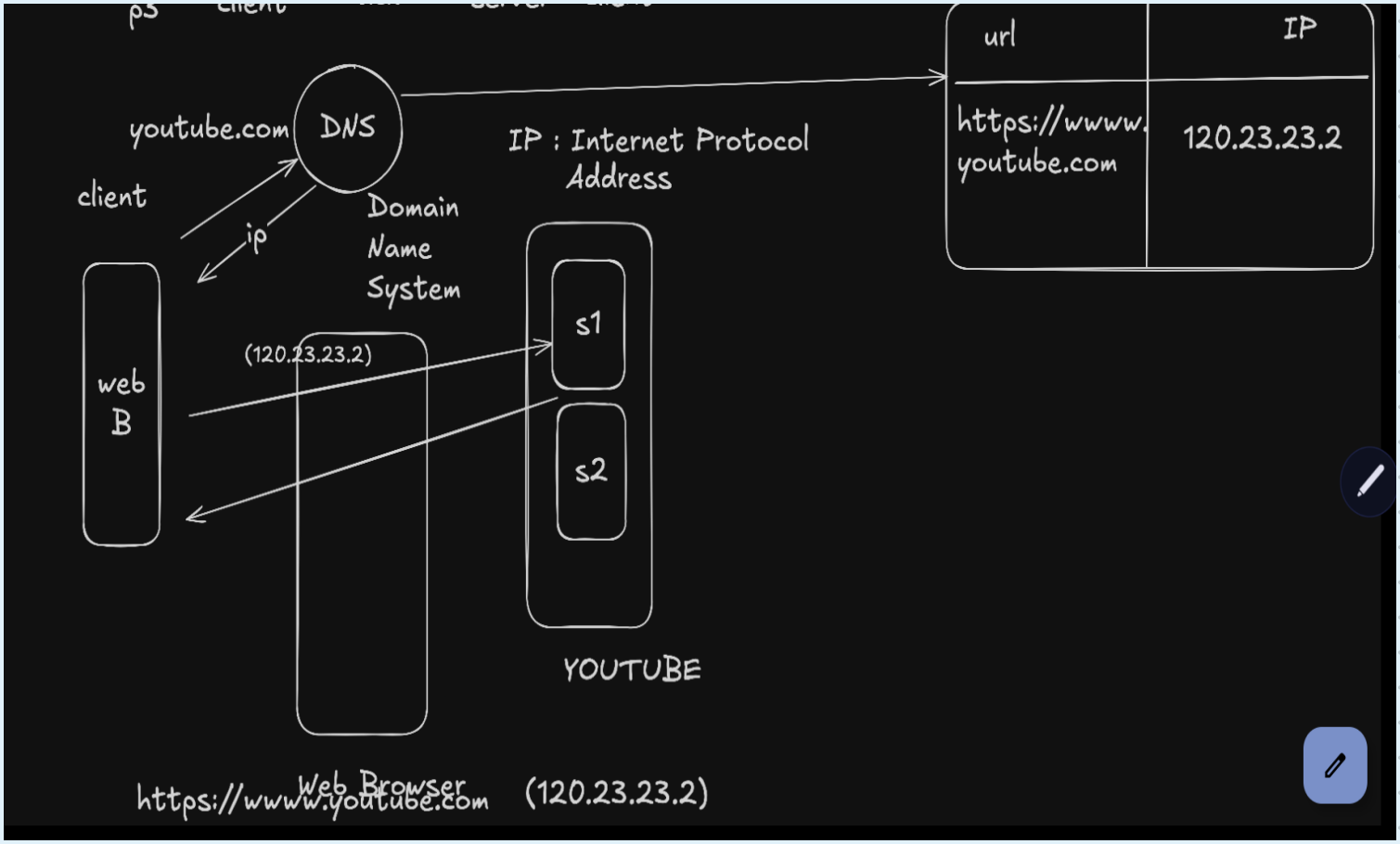
↳ client asks as domain name of that application but we need IP address

↳ DNS stores it domain_name : IP_address

↳ when client asks it returns ip address

◉ we will study it as HLD problem in deep later

DNS Example


youtube.com
DNS
client
ip
Domain
Name
System
IP : Internet Protocol
Address
url
IP
https://wwww.
youtube.com
120.23.23.2
web
B
(120.23.23.2)
s1
s2
YOUTUBE
Web Browser
https://wwww.youtube.com
(120.23.23.2)