1. Good Evening
2. Lecture begins at 9:08 pm
3. Topic → HTTP

Agenda

1. Recap
2. Application Layer Architecture
   - Client Server
   - Peer to Peer
3. Sockets vs Ports [Lecture 4]
4. HTTP

Recap

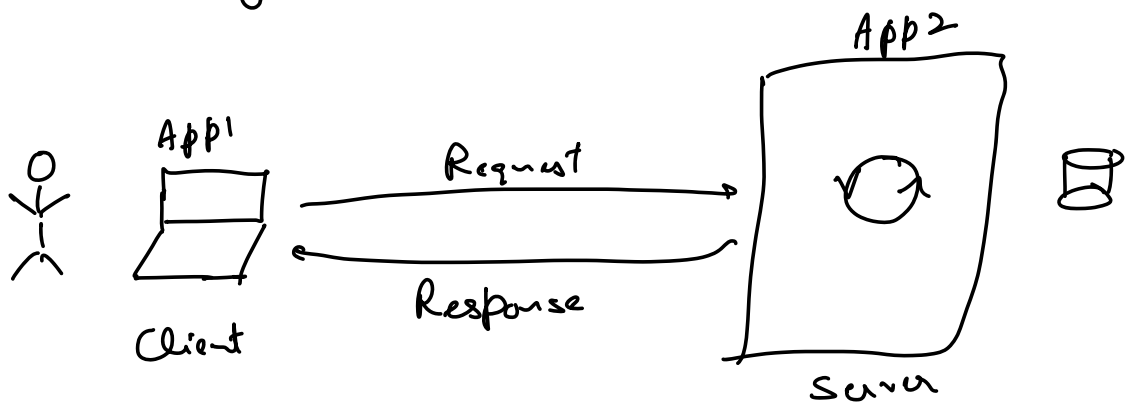
	OSI Model		TCP/IP Model
1.	Application	S/P	Application
2.	Presentation	Encrypt / compress	
3.	Session	Ocn	
4.	Transport	Data break & reassemble	Transport
5.	Network	Routing	Network
6.	Data Link	Hop to Hop	Data Link
7.	Physical		

Application Layer Architecture

1. Client - Server]
2. Peer to Peer]

Client Server Architecture.

* Different application on two sides of network.



Responsibility of appn. on client side.

1. Frontend : JS, React
   Android
   IOS
   - Presented to the user
   - Take input from the user.

2. Send user i/p to server side.

Responsibility of code on server side.

1. Receive data sent in client request
2. Process the request [Business logic]
   - Maybe store in a db
3. Send the response to client.

Client	vs	Server
1. Interacts with user		1. Interacts with db
2. Request		2. Response.
3. Frontend e.g. React		3. Backend e.g. Java, Spring, Python, Django, Node, Express
4. Consumer of the content		4. Owner of content.

Peer to Peer Architecture.

e.g. Bit Torrent.

* Same application on each device

* No division of responsibility.

* Everyone is a consumer as well as provider of data.

Torrents.

1. Indexer : Site
   ↳ Maintains list of files that can be downloaded.

abc.mp4 → abc.mp4.torrent

512 MB file → 30 kb file (Torrent file)

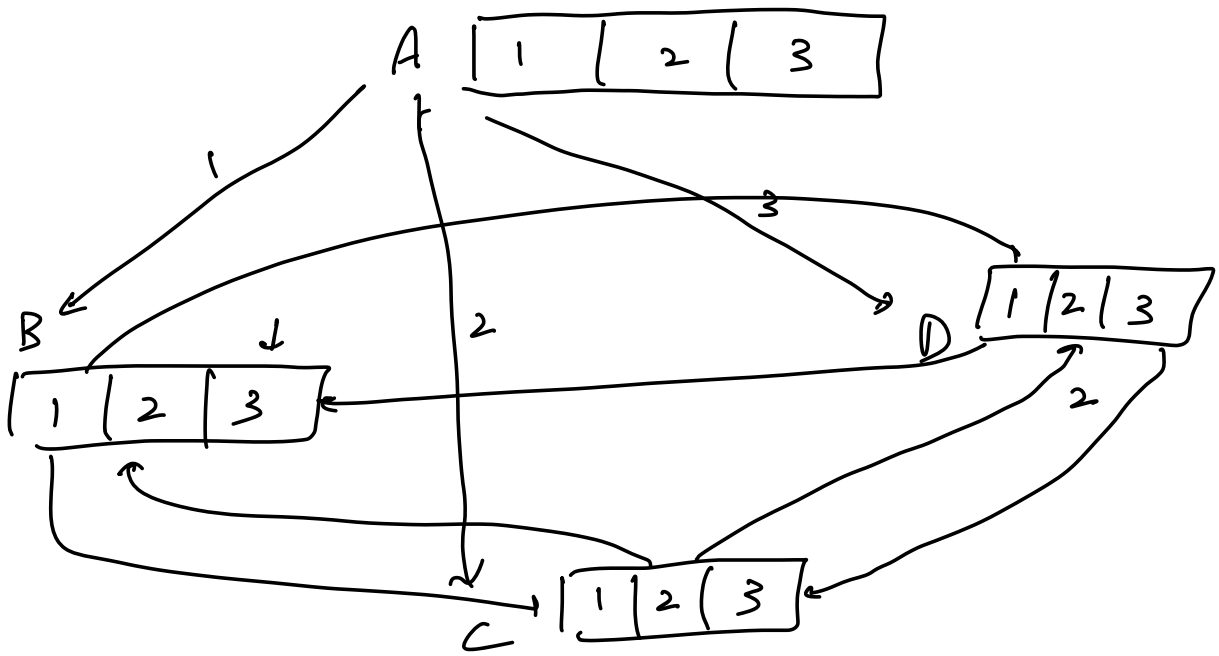
2. Torrent Client App : utorrent

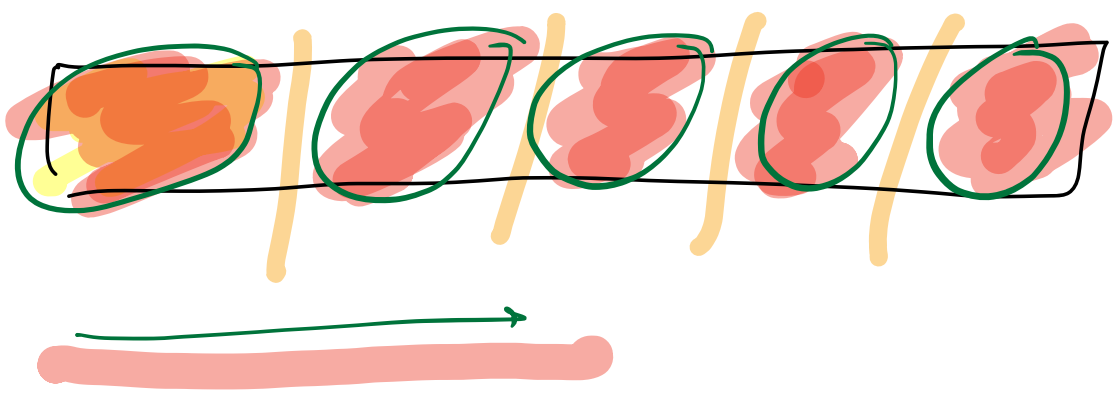
Torrent file → Original file

* Torrent file → Has details of a server (tracker) [who has dynamic info about which peer has which

part of file.]

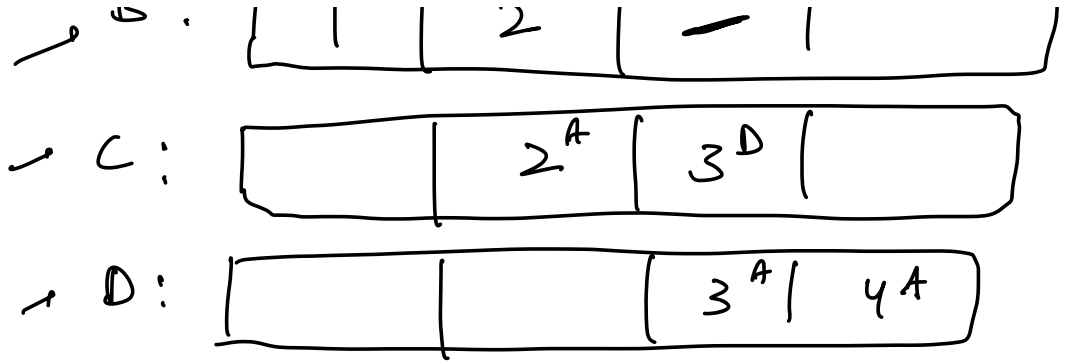
Tracker → A special node





A : | 1 | 2 | 3 | 4 |

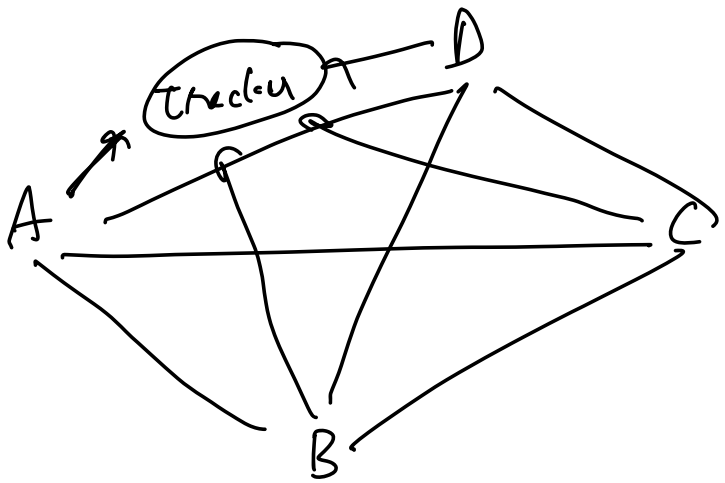
D : | A | C | | |



Tracker:

Dynamic Hash table.

1 → A, B
2 → A, C, B
3 → A, D, C
4 → A, D



Tracker → Server which maintains which peers have which parts of the file available.

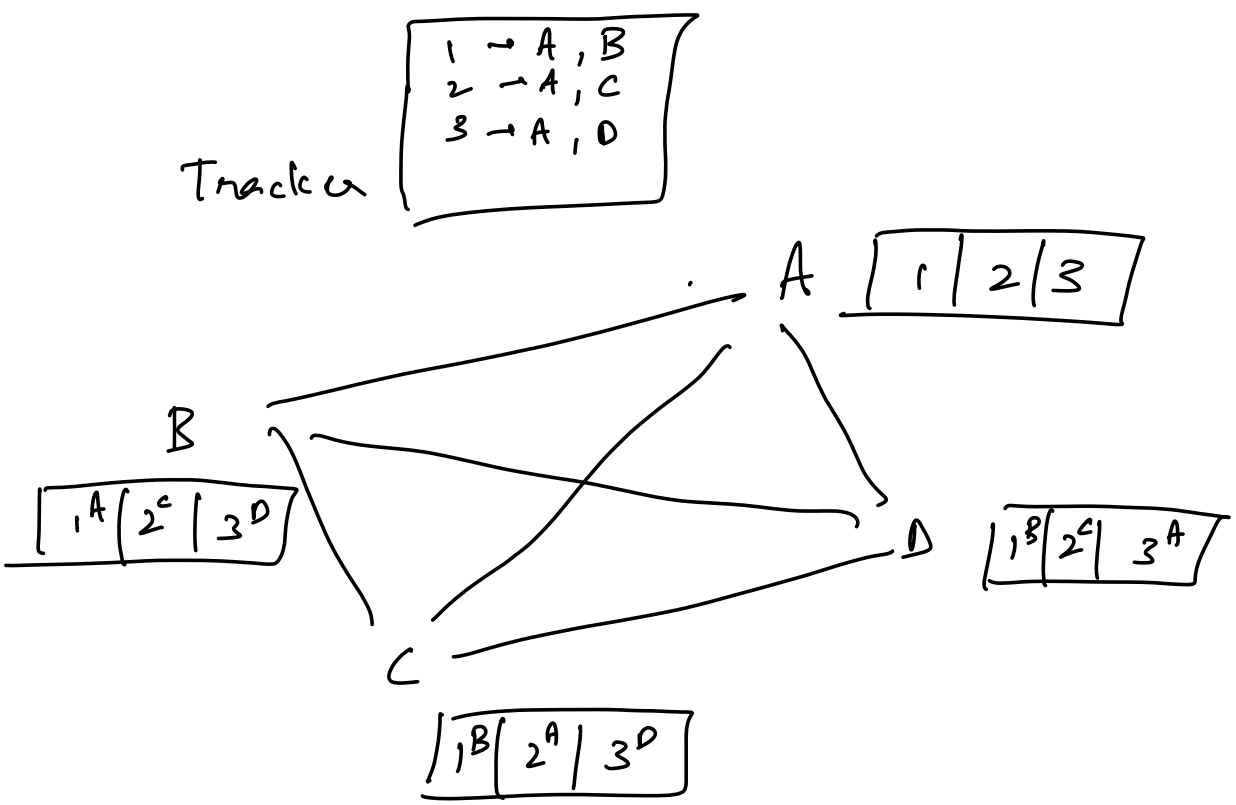
Peers: A, B, C, D want to

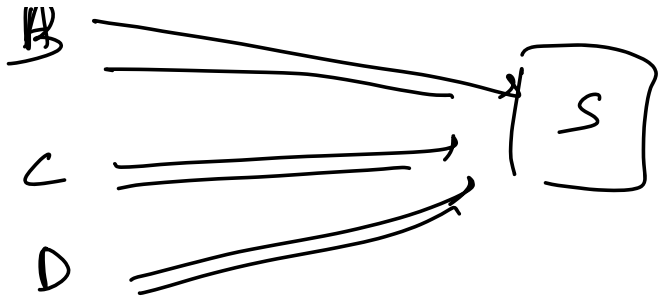
download a particular file.

→ Indexer : Website ⟶ A torrent file

Torrent file → Has address of Tracker

* Tracker : Server which maintains dynamic info about which peers have which parts of the file





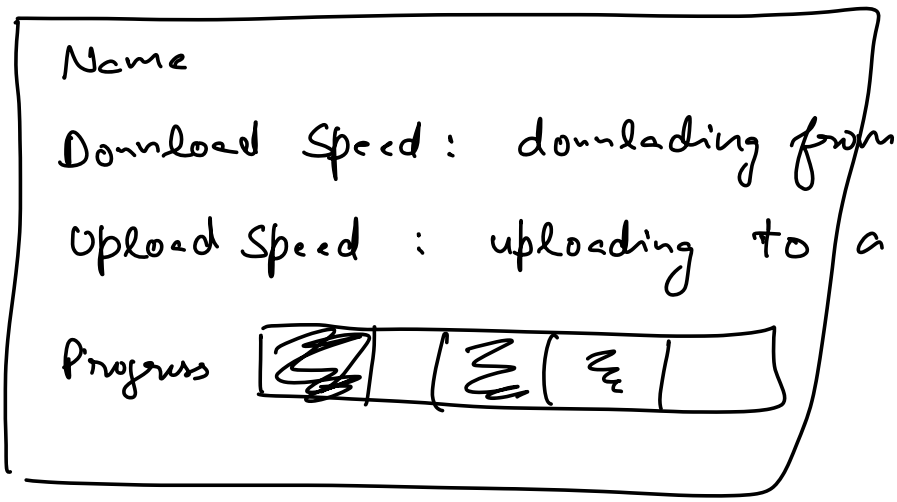
Peer to peer architecture is trying to decentralize the load on A?

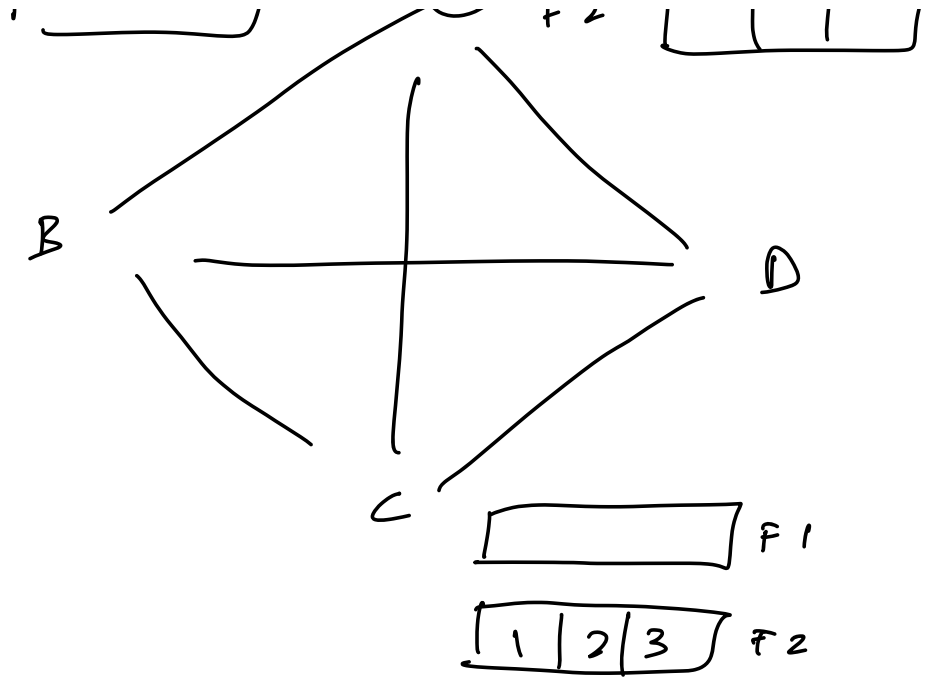
Peers

→ Seeders : Upload files

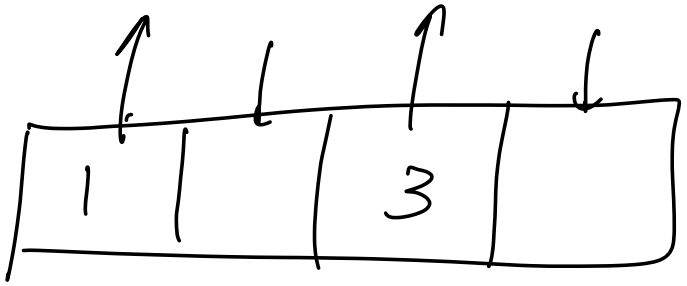
→ Leecher : Download

Peer → Seeder & Leacher.





☆ If there are more seeders download speed will increase.



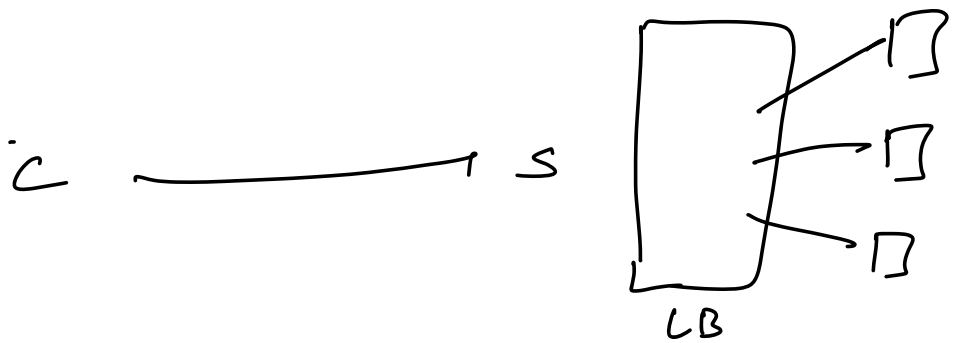
☆ Seeder :

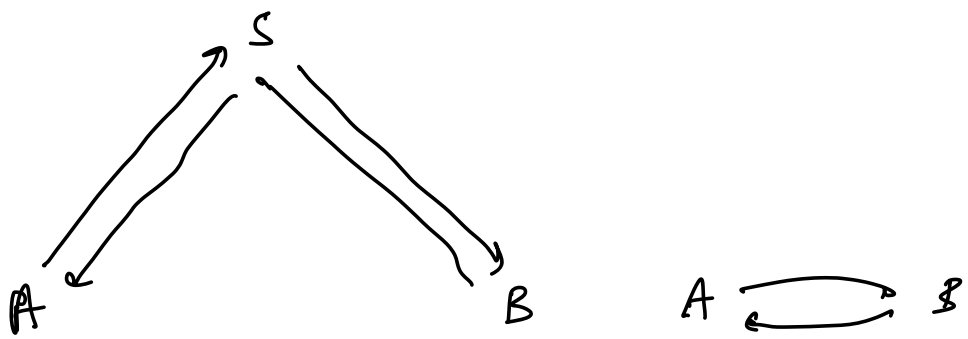
☆ Leechers :

☆ Peer → Both a client & a server

RabbitMQ → Confirm ?

CS	VS	P2P
✡ Division of Responsibility		✡ No division of responsibility
✡ Client consumer		✡ Peer consumer &
✡ Server provider		provider also





HTTP Protocol

HTTP → [Hyper Text Transfer Protocol]

HTML → Hyper Text Markup Language.

Origin of HTTP → A protocol to enable the exchange of Hyper Text documents.

Break → 10:10 → 10:18

Properties of HTTP

1. Client – Server communication
2. Uses TCP on transport layer

HTTP 1
HTTP2 → TCP* [Guarantee] Segments

HTTP3] → UDP* [No Guarantee]

3. Stateless protocol

What is state?

State of an object

Person ?

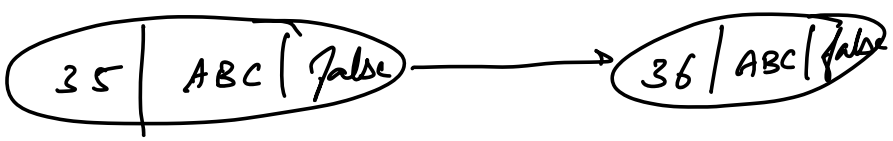
age
name
ms

}

Person p1

p1. age = 35
p1. name = "ABC"
p1. ms = false

p1. age = 36
p1. ms = true




```
int square (int x) ?
    return x * x;
}
```

```
int x = Math.square (3) ⟶ (1)
int y = ".  square (4) ⟶ (2)
```

```
class InterestCalculator {
        int roi
    void setROI (int roi) {
        this.roi = roi;
    }

    int getInterest (int p, int t) {
        return (p * t * roi) / 100;
    }
}
```

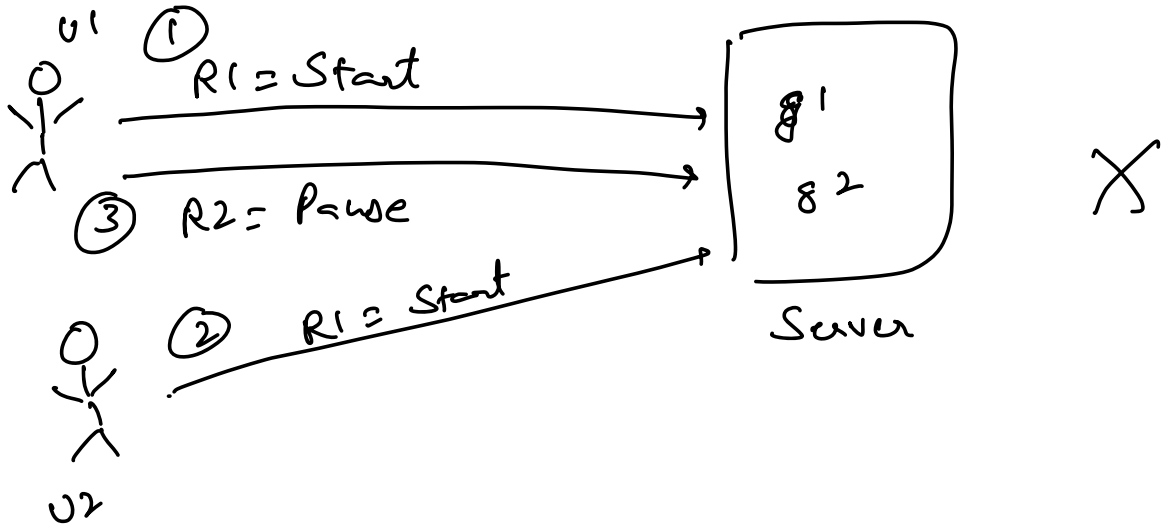
```
InterestCalculator ic
8 ← [ ic.roi(4);
      int y = ic.getInterest(100, 2); ]

ic.roi(5)
10 ⟶ int z = ic.getInterest(100, 2);
```

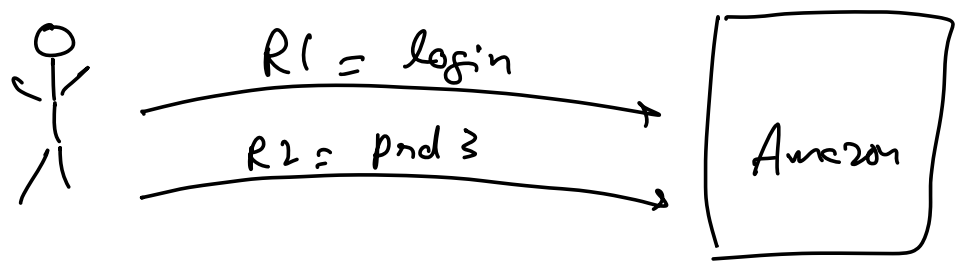
HTTP is a stateless protocol

* Every request is independent & self-sufficient.

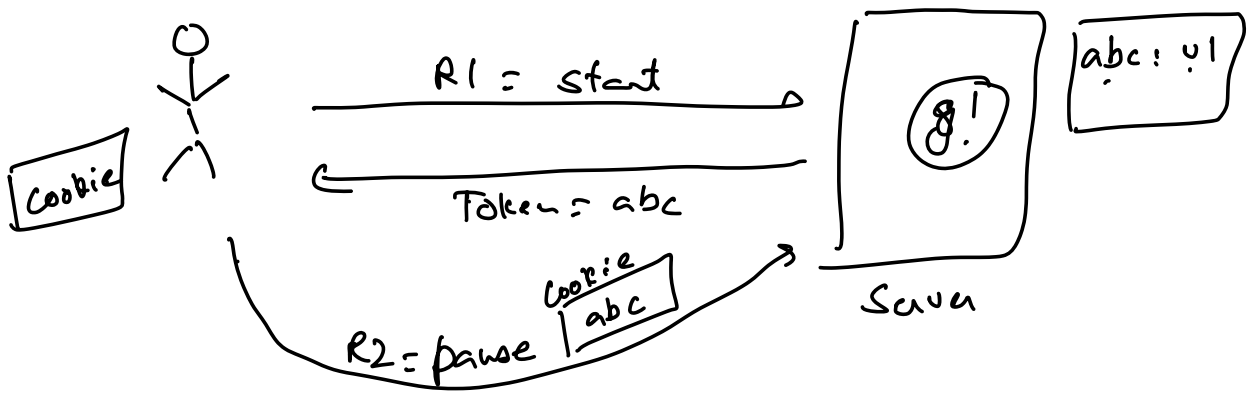
* No data is automatically stored on the server for future.



HTTP is stateless



Cookies



Servers use tokens & cookies to remember states btw requests.

is a stateful protocol → fff

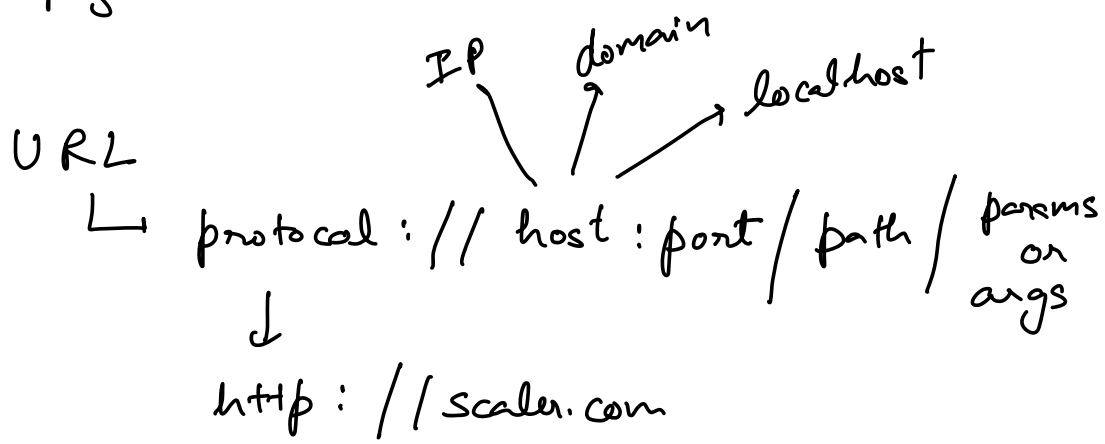
1. Client - Server
2. TCP
3. Stateless

4. Every resource is an object which is identified via URI / URL

html
pdf
css
js

URL : Uniform Resource Locator.

jpg
png



Is Port mandatory? → No

HTTP → 80

HTTPS → 443

✡ Path → [meetings / i / ——— / live]

✡ Query Params → ? → forced = 1

5. Types of HTTP Connection

1. Non persistent

✡ connection → close → New TCP connection for all requests

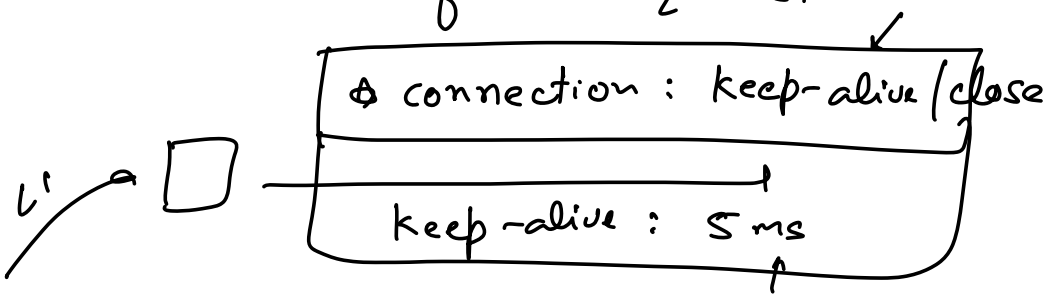
↓ keep-alive

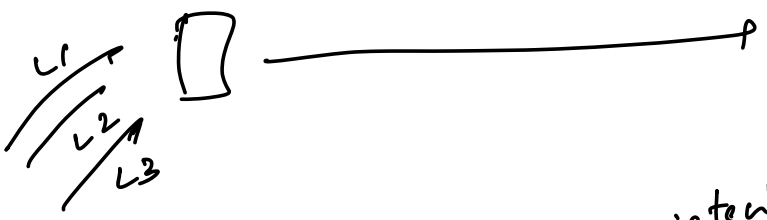
KA

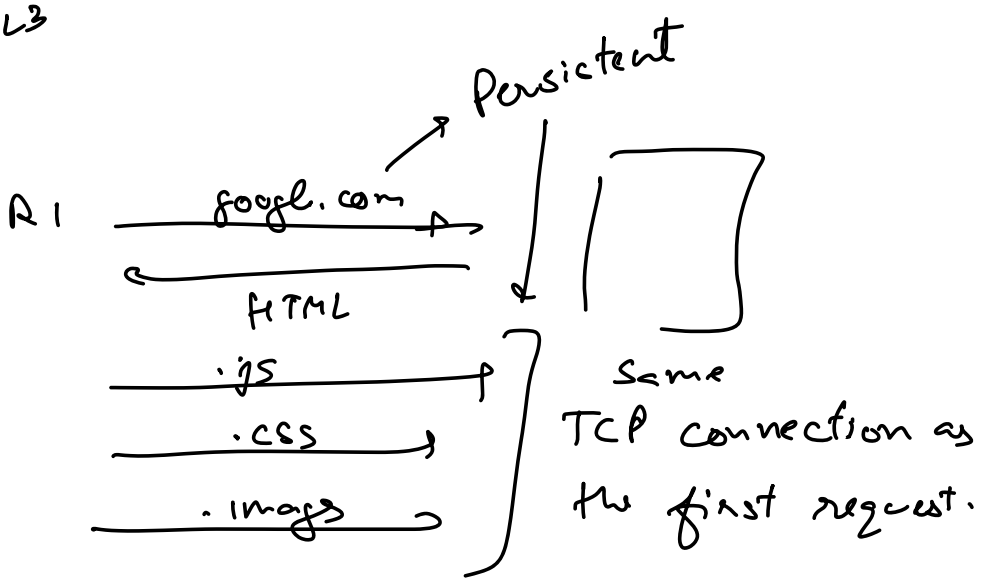
2. Persistent

keep-alive : 5ms

* Same TCP connection for future requests.







HTTP Request

1. Method in HTTP : Get, POST, PUT, Delete, Patch.

2. ✓ Path

3. ✓ HTTP Version

4. ✓ Host : domain name

Persistent TCP

5. Connection : keep-alive | closed

6. Keep-alive : 5 ms

7. User Agent

Methods

1. Get URL [Information passed in get becomes a part of url]

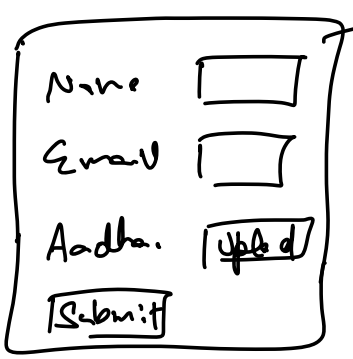
↳ Request data from server from a location.

2. POST → Send data to the server to create a new resource.

↓

Insert

e.g.

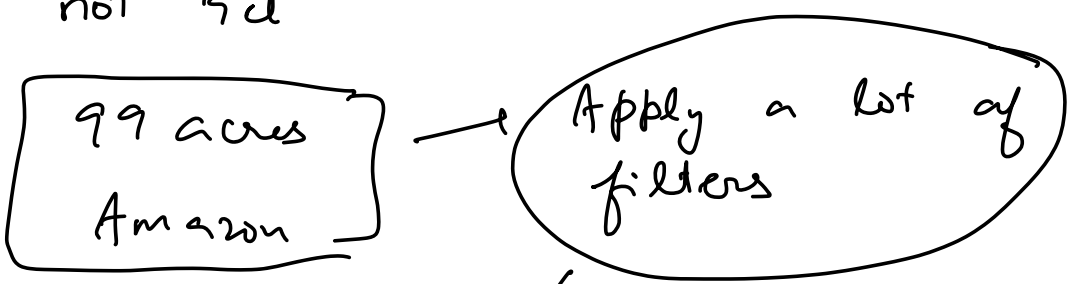


→ POST HTTP method

3. Put → Updating resources on the server.

Limitations of Get → Size of url
↓
256 [2048] characters

e.g. of a request to get information where we have to use POST & not Get

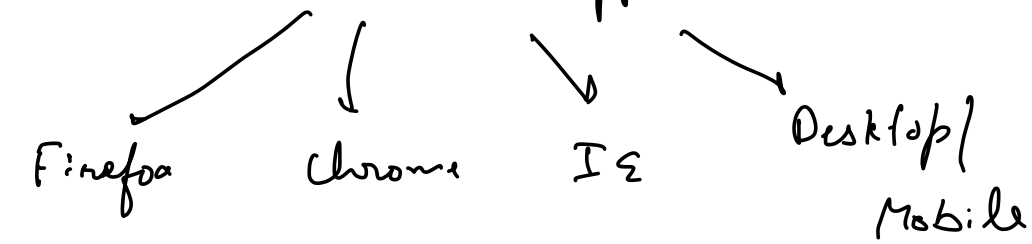


↓
might exceed the limit of Get Method.

Ideal

Get	POST	Put	Delete
↓	↓	↓	↓
Get	Insert	Update	Deleting.

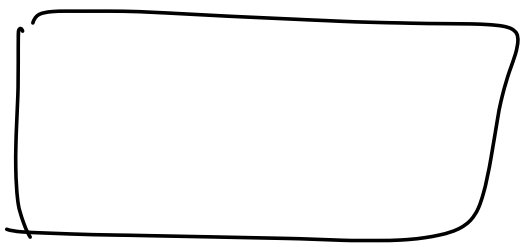
User-Agent : [Contains information about client application



Server can send customized pages

HTTP Response.

1. HTTP Version
2. ☆ Status Code
3. Date
4. Server.
5. Content-Length
6. Content-Type → HTML | Pdf | mp4 | json.
7. Real data



Status Code

1xx → Information

2xx ⟶ Success [200]

3xx ⟶ Redirection [301]

4xx ⟶ Client errors [404 → Not found, 403 → Access denied]

5xx ⟶ Server errors

500: Internal server error

501: Not implement

502: Bad gateway [Load]

Next class

- Curl [Cmd line]
- Cookies
- TCP
- UDP
- IP

L4 — Socket Programming