

# COMPUTER NETWORKS - II

① Application layer arch.  
- CS }  
- P2P }

② Application layer

- HTTP

- Structure - Req.

- Response

③ Socket vs Ports

---

$$2^{16} - 1$$

→ [65, 535]\*

Socket → Python

## Architectures

① Monolith

# Client - server architecture

Two parts

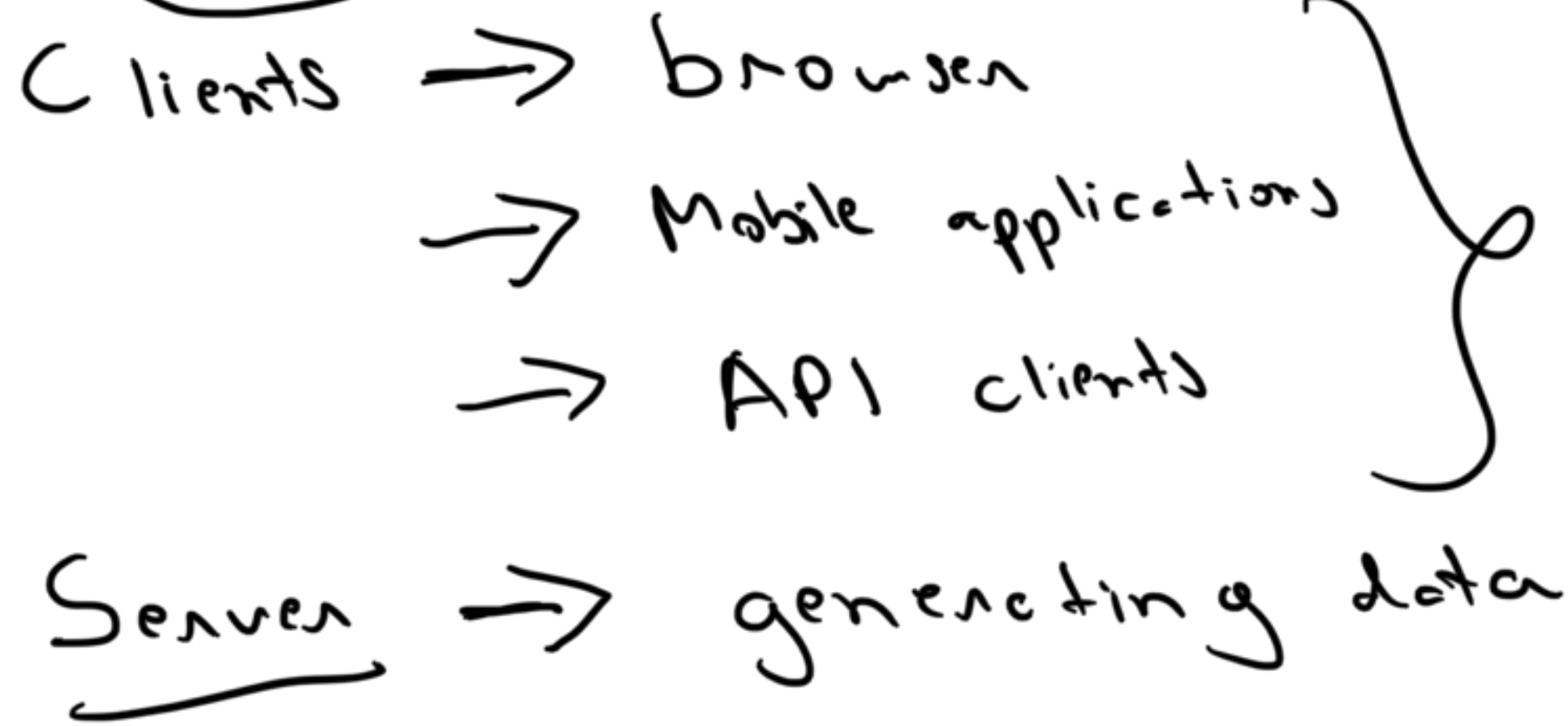
① Client

— consume data

— take input from the user

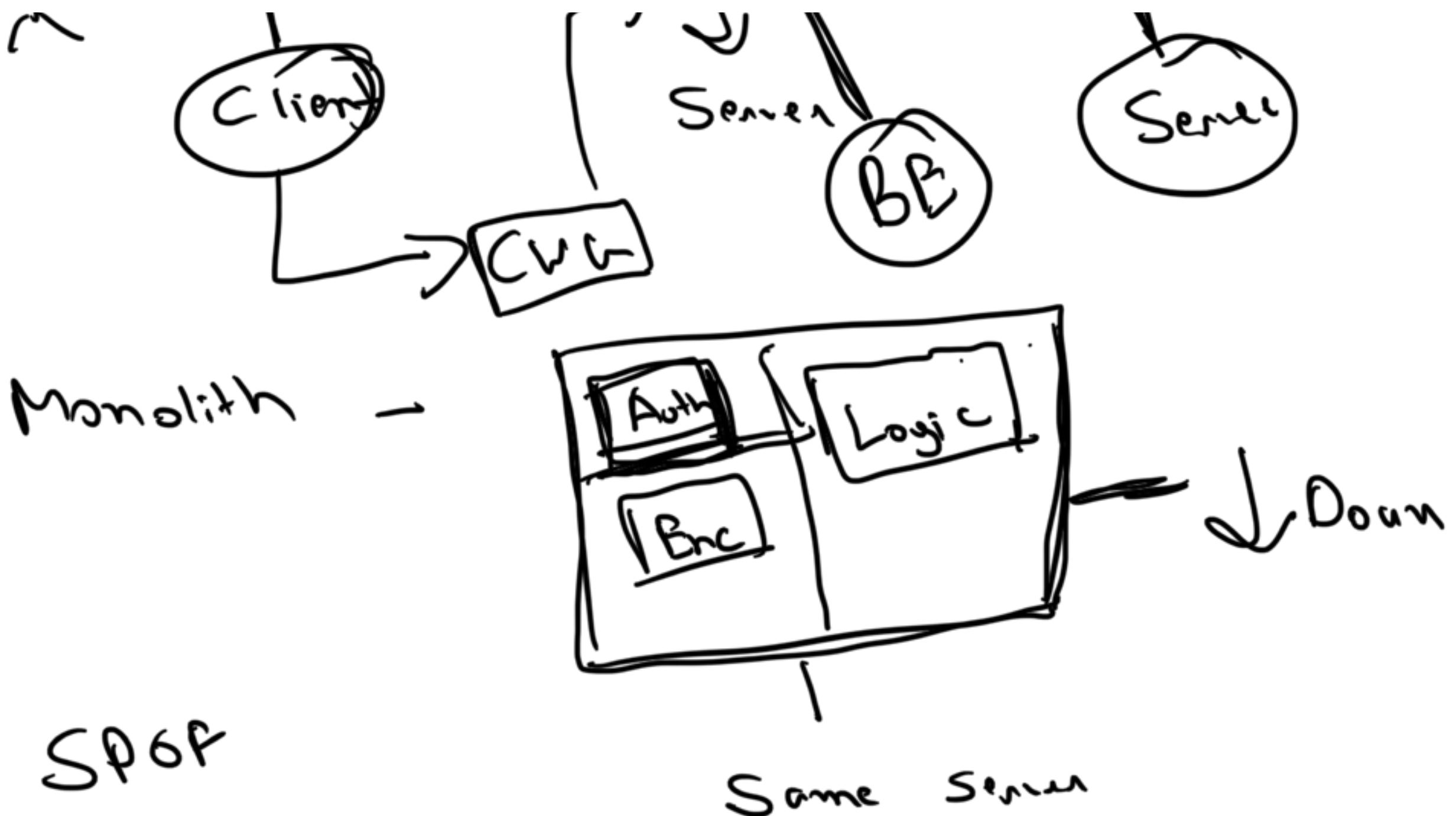
— gets data from the server





### Client - Server model

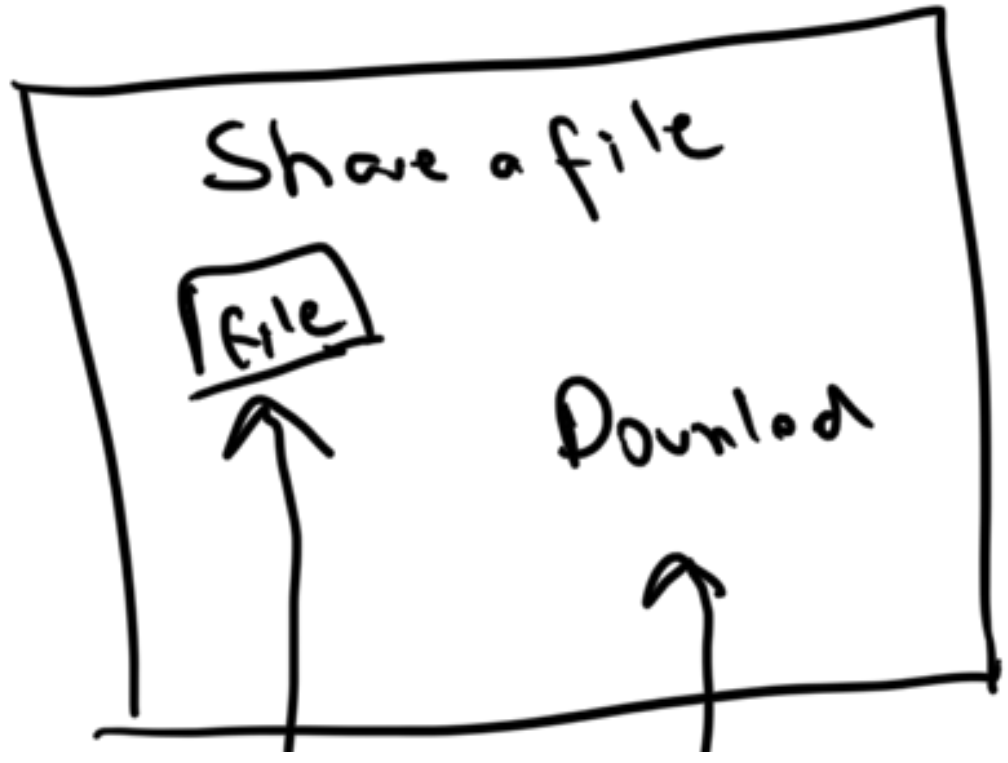


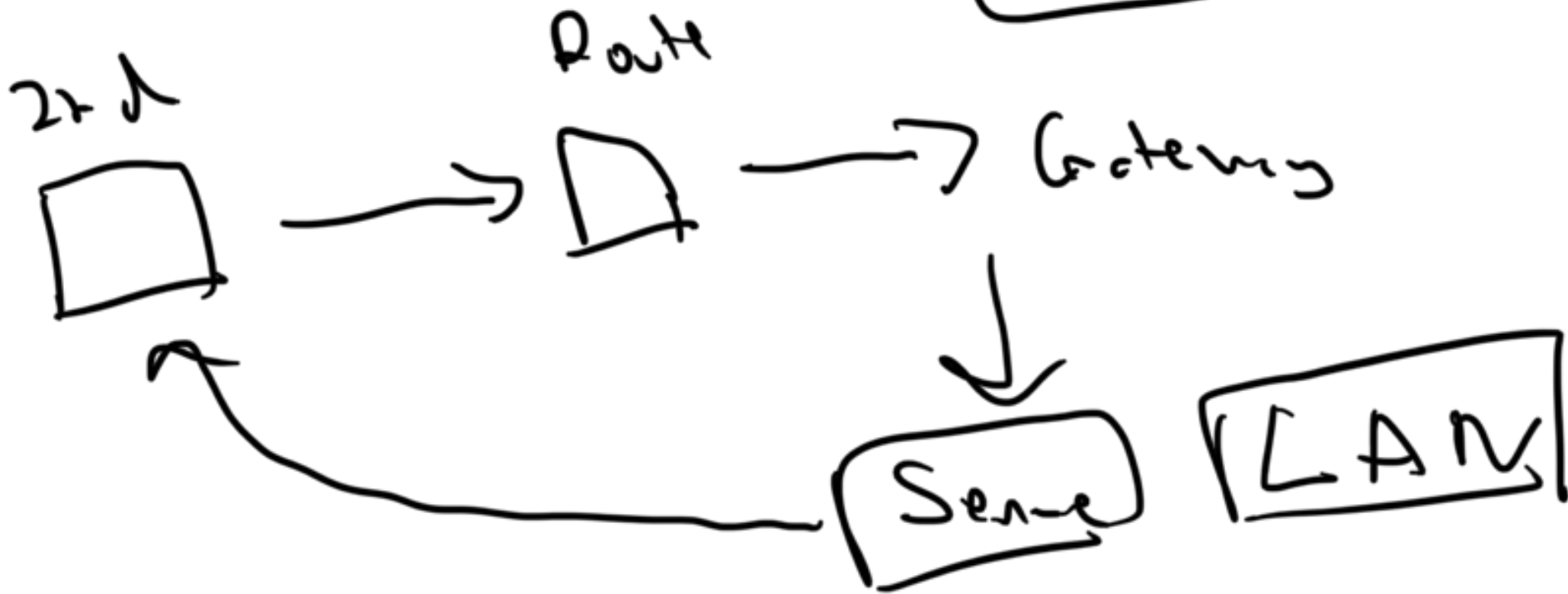




Microservices

---





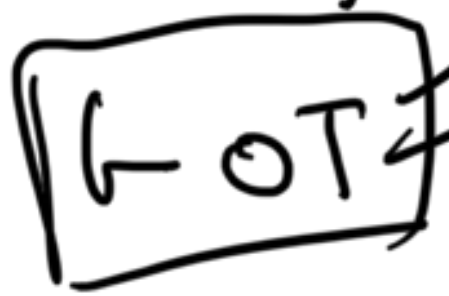
P2P → peer to peer networks

From ...

Linux amp...

① Torrents

②



③ Bitcoin, Eth

Torrents

• → I want to download a file

Server



CS



① Latency

② Concelling download

③ There are 1M people downloading

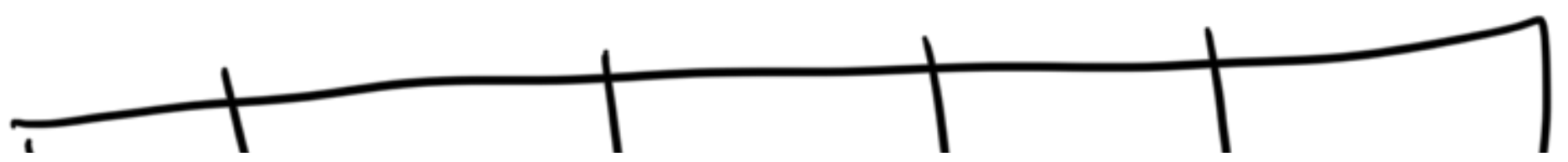
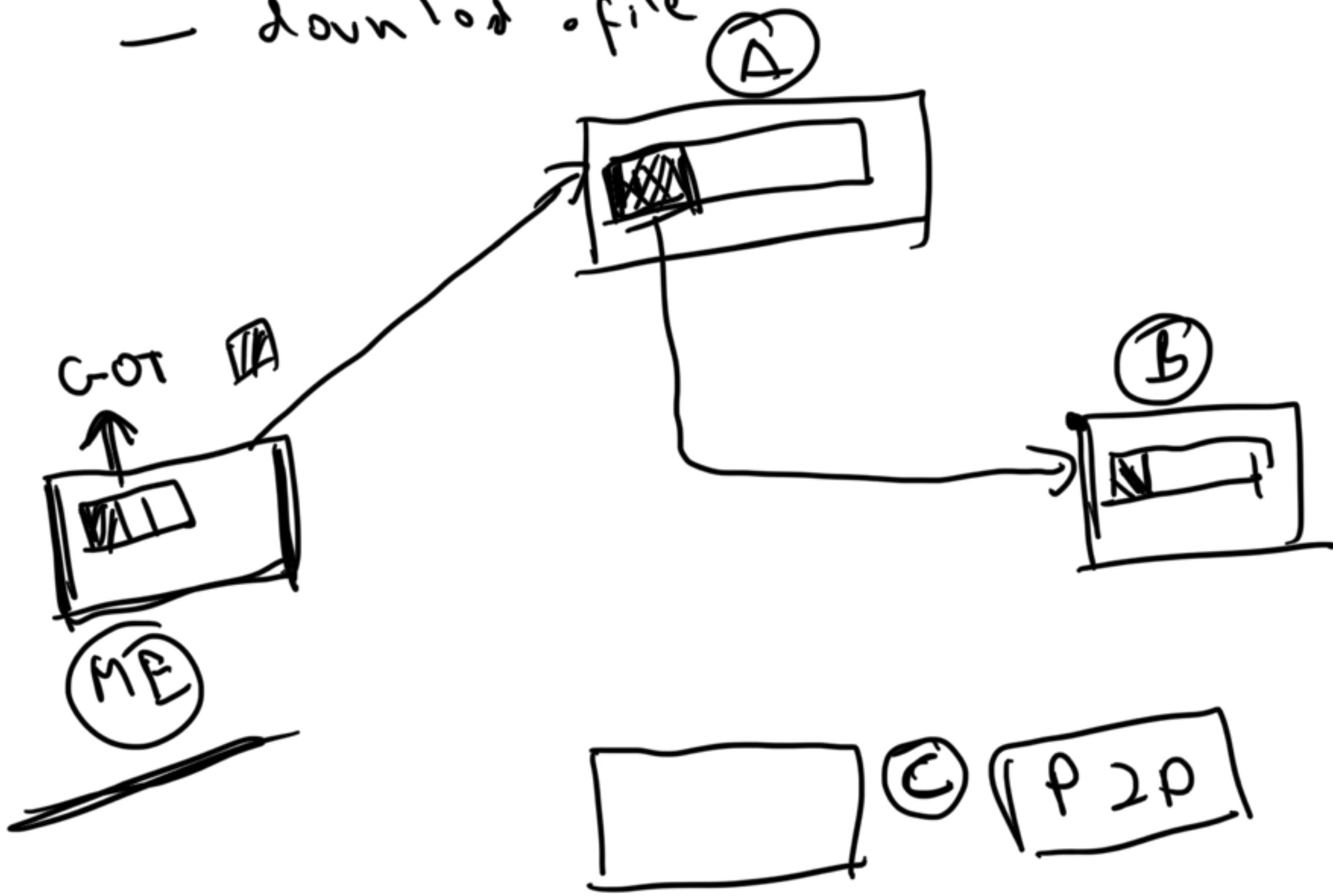
→ Expensive

P2P

- peer

- upload a file

— download of file

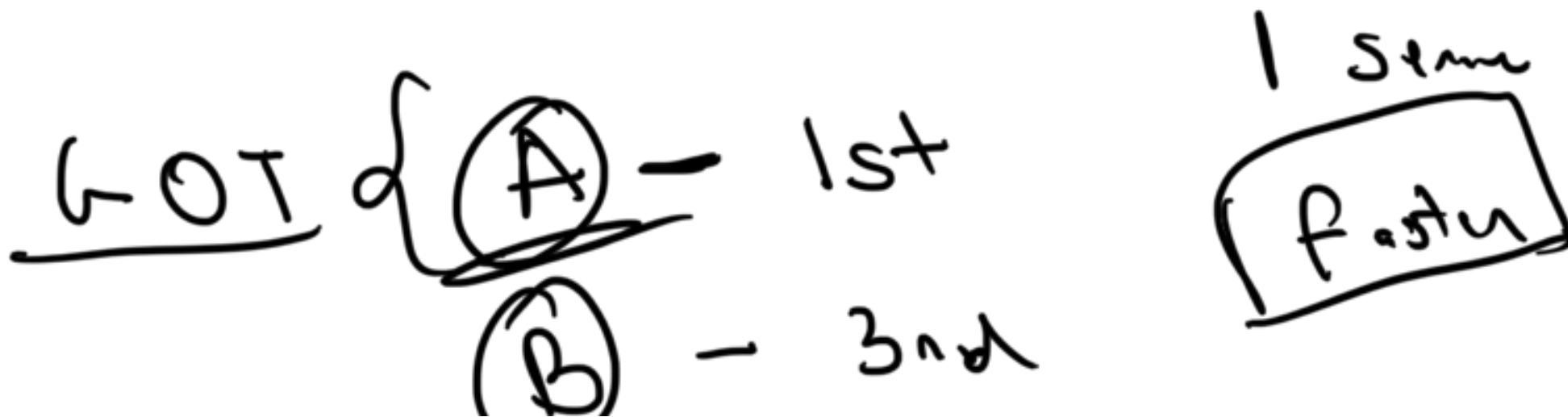


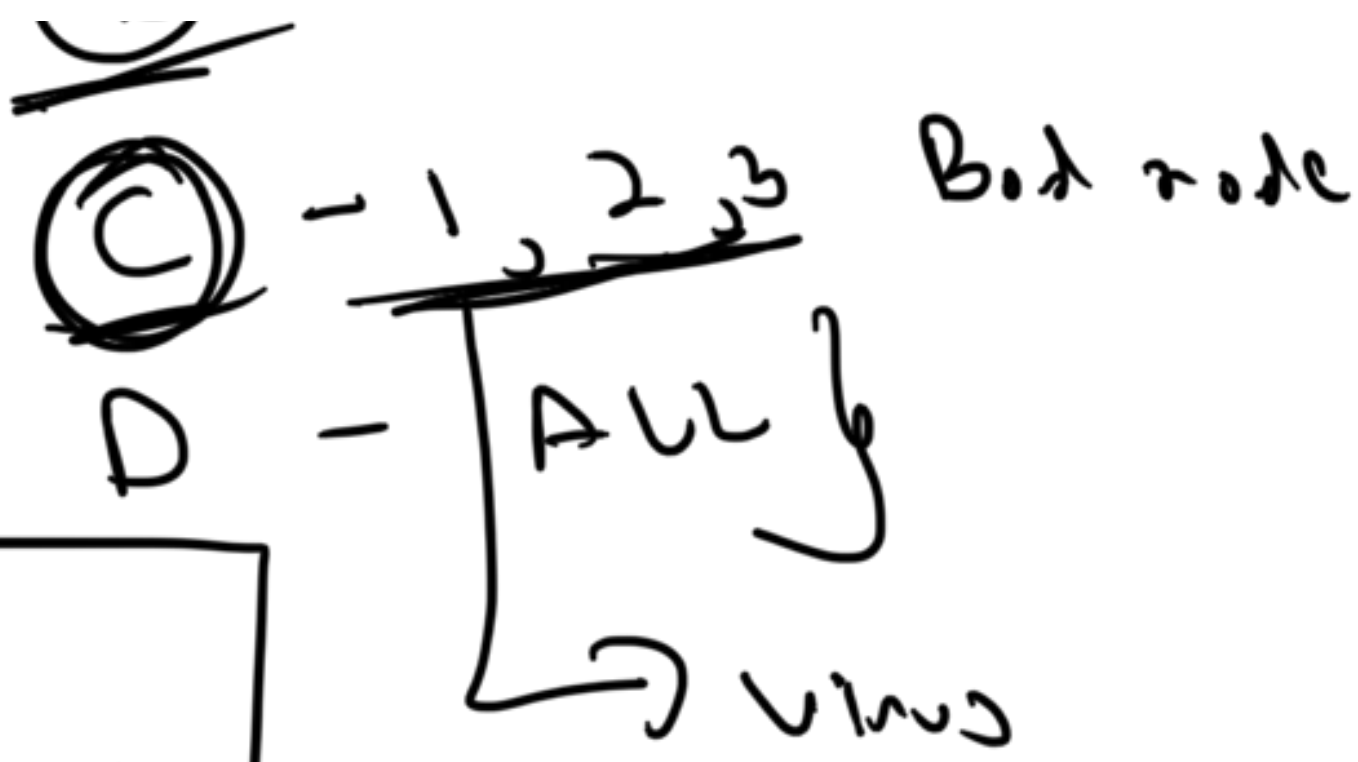
↓

BitTorrent - OTT

Two types of nodes

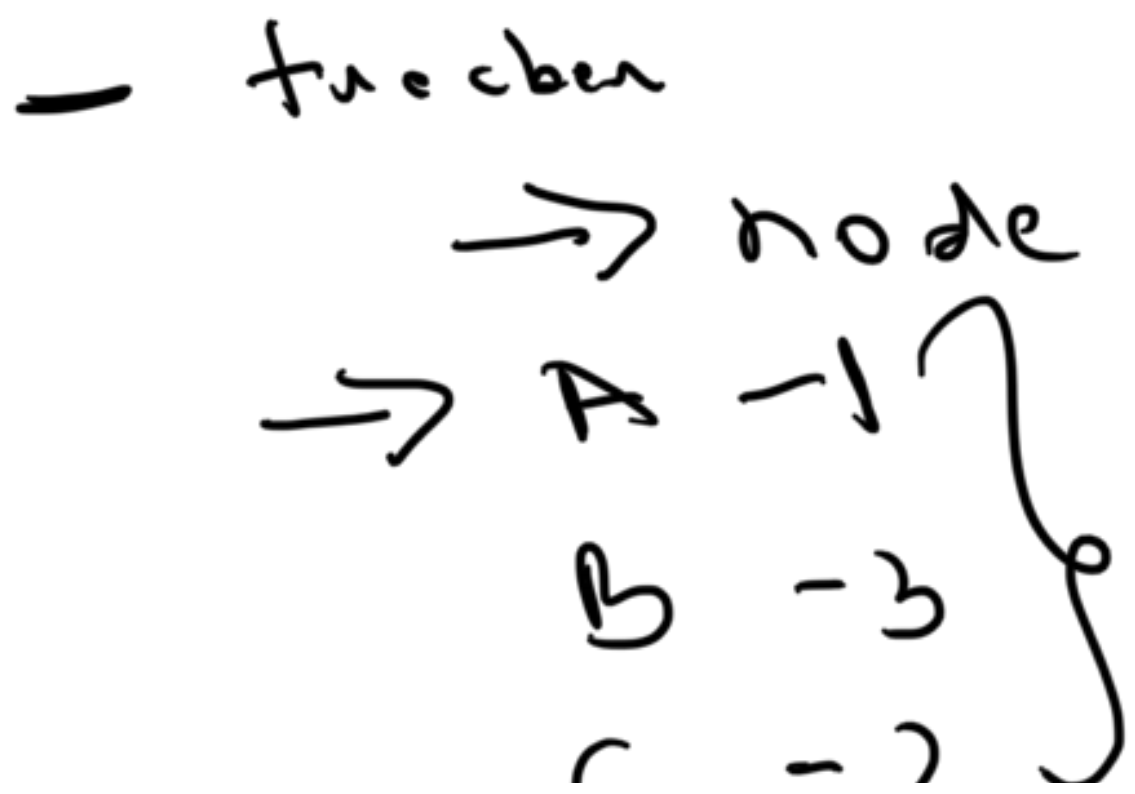
- ① Tracker - does not store data  
- which person has what part of a file
- ② Peers





• tournament  
 → tree based

Bit tournament client





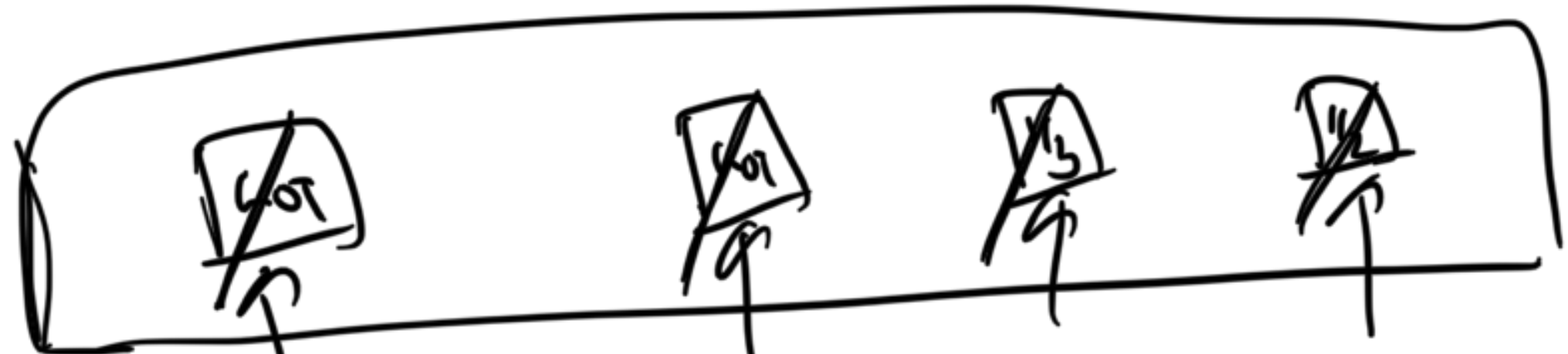
Got

. format

!!

- ~~checksums~~
- Hash

Security



Seeders

only do unseed - leechers

Private

Private

= 6 Seeders / leechers ratio

- 1 GB / 100 MB

- 10

1


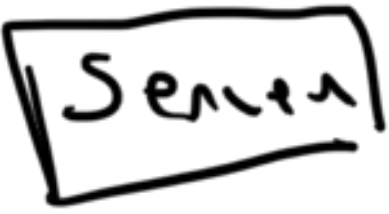
- slow speeds

- less seeders


- deactivated

CS

P2P

Security  Server 

No central control

 - BitTorrent

- Torrent insecure

Performance

can be super performance but it is expensive

Seeders

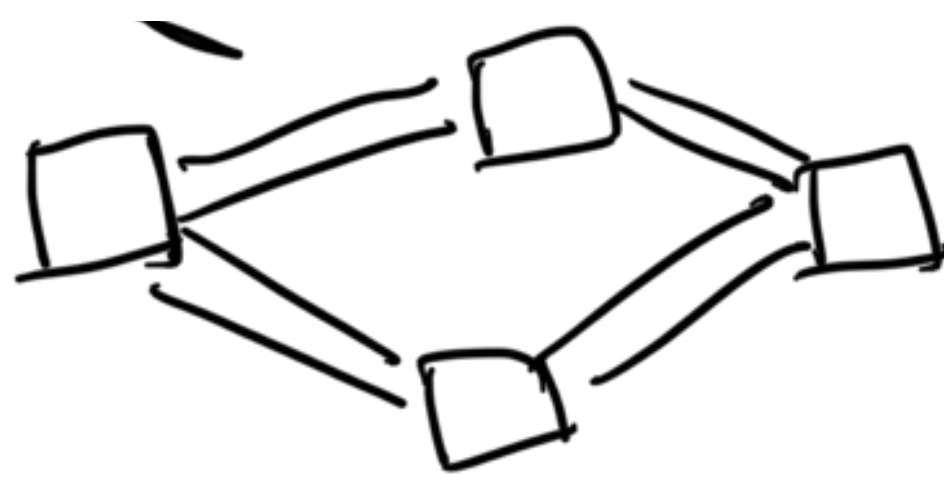
unpredictable

depends on nodes

① CS



② P2P -



in reducing load on servers

---

\* Microsoft Windows 10 - P2P update delivery

\* Spotify - mobile

CS + P2P

---

HTTP



5:36 : 5:42

10:06 : 10:12

---

Starting any dev

- Monolith



Auth Email



- microservice

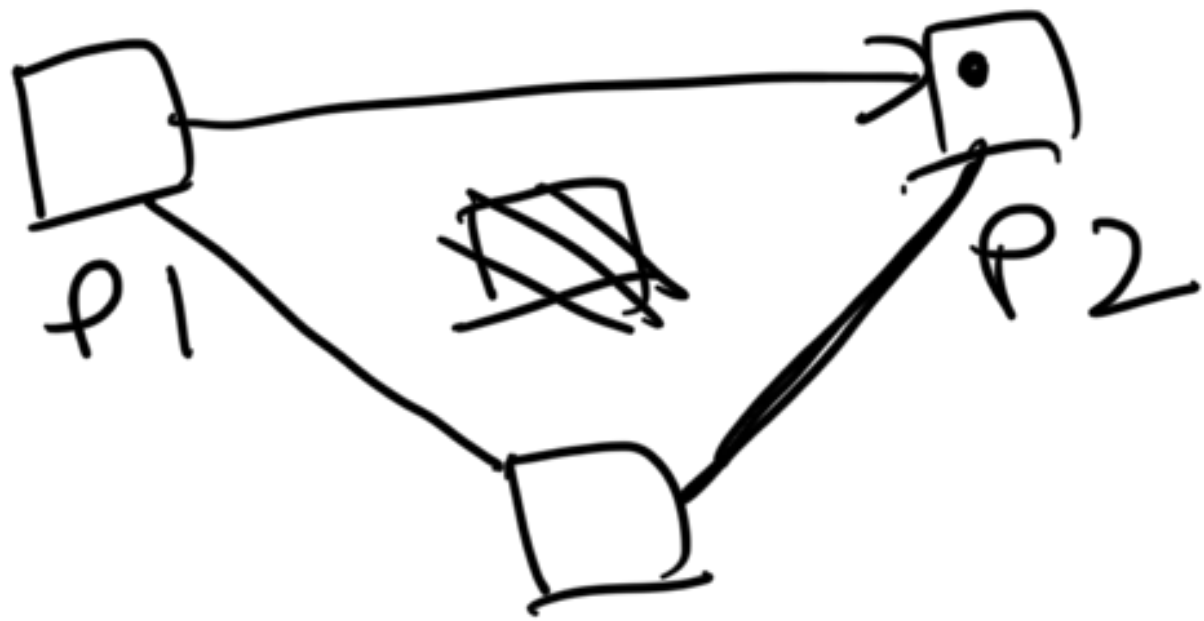
need to scale only m or y

 Tower





- google..
- fb.com



Root

JS = browser

---

HTTP

↳ Hyper text trans. proto.

HTML

Hyper text

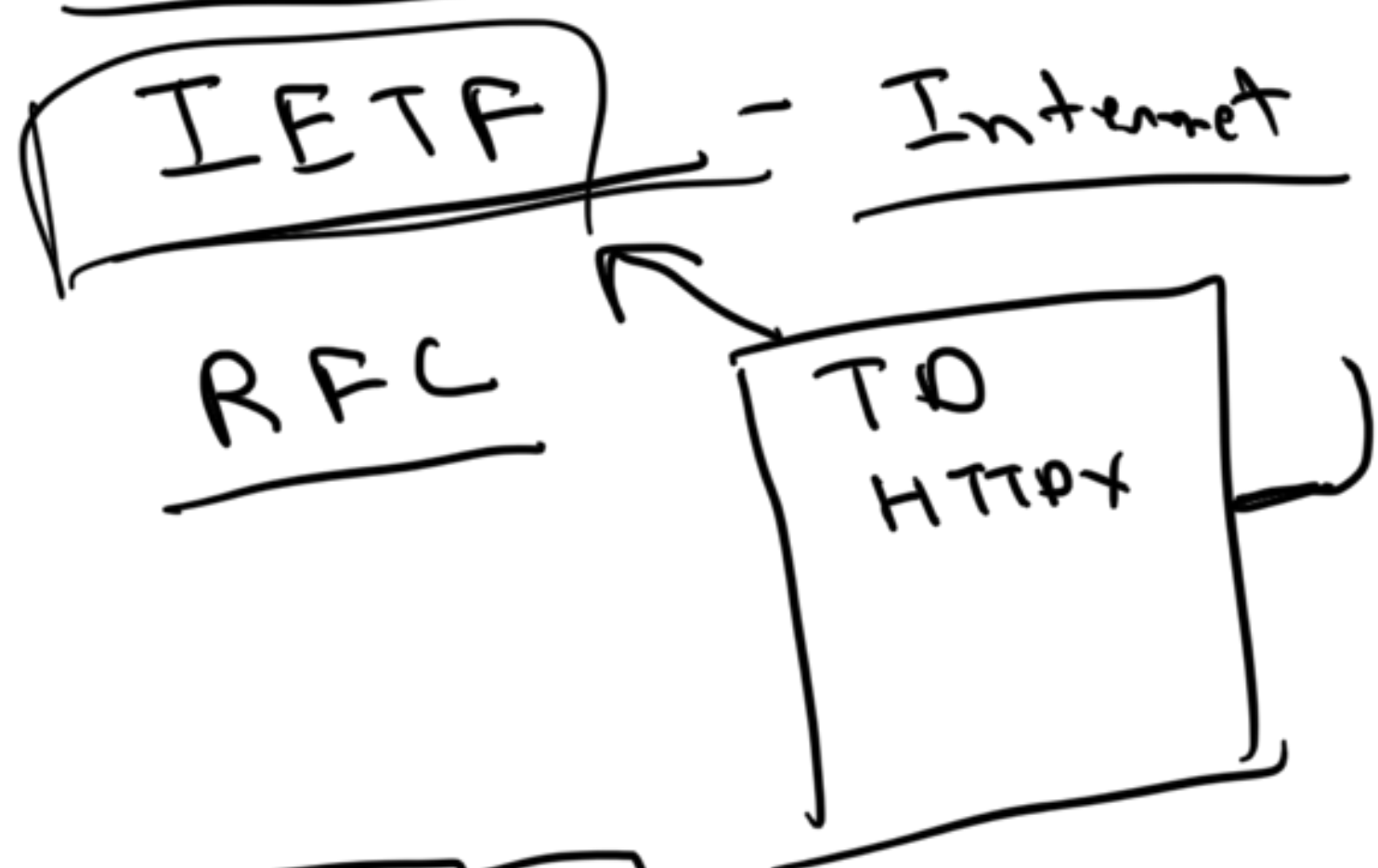
hyperlink

---



HTTP version

---



121

HTTP/1.5

HTTP/3

25'

HTTP/2

Tim Berners-Lee

HTTP/1.1

---

① Client-server protocol  
X P2P

② HTTP uses TCP  
as its Transport layer protocol

H1

H2



P1 - P2 → HTTP

Transport level - TCP

→ DNS solution

→ Cache

→ /etc/hosts

→ DNS server



→ IP address

Create a connection

TCP

host to host comm.

→ HTTP uses TCP

HTTP 1.1 } TCP  
2 } TCP

HTTP 3

QUIC

UDP

HTTP 3 - ~~3x~~

→ QUIC - google

→ SPDY

### ③ Stateless protocol

cd  
is





① cd directory → { cd: "" }

② ls

{ pwd: }

Stateful application

Stateless

① ls

{

command: 'ls'



Stateless memory

---

Stateful



① Set x = 10

Stateless

① Command: incr

② INE { x = 10 }

Stateful



FTP

| job x: 10

Stateless



HTTP

Disadvantage of stateless

① Waste bandwidth

Advantage

① Scalable

② Reliable

---

① CS ✓

② TCP ✓

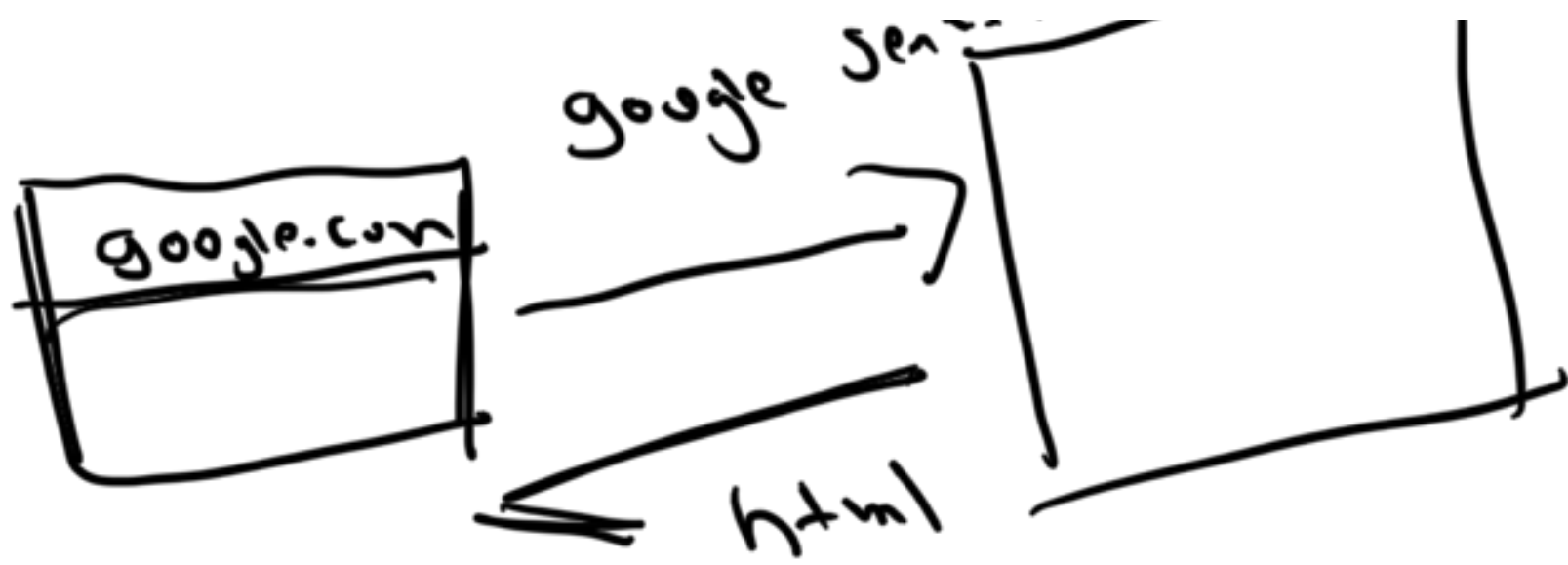
③ Stateless

④ Each response is stored as an

---

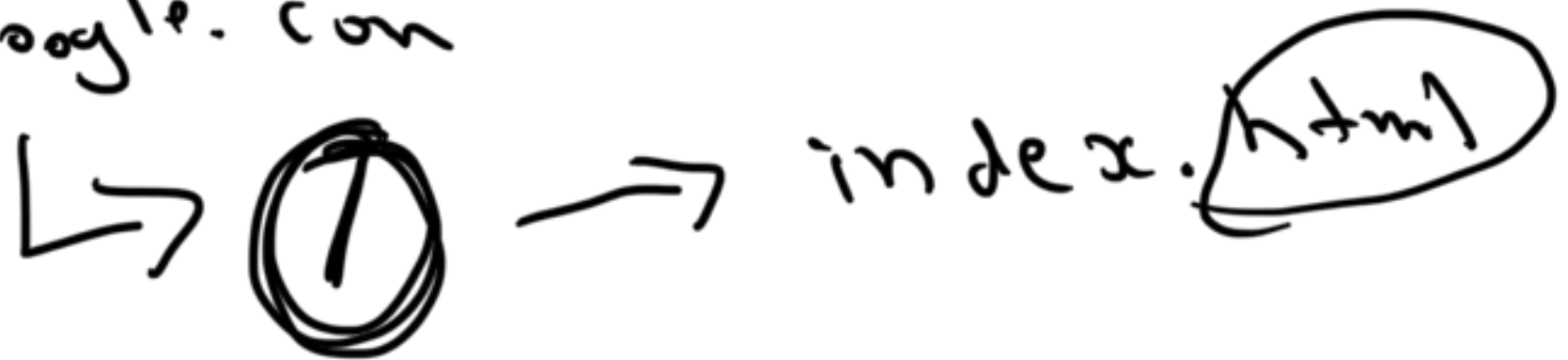
identifiable object

---



cd / → root

www.google.com



www.google.com/news



1 maps



→ html

→ JS on

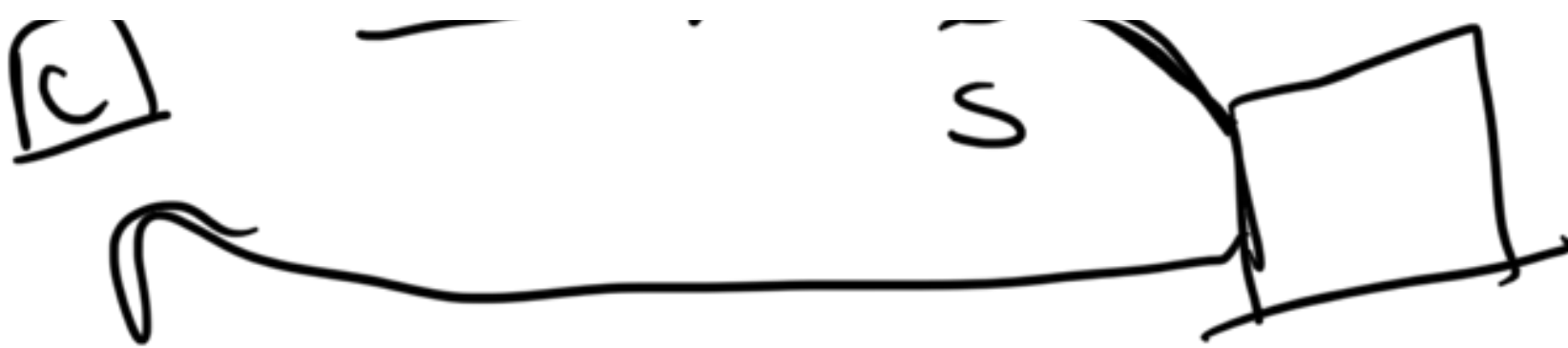
→ XML

---

1 news

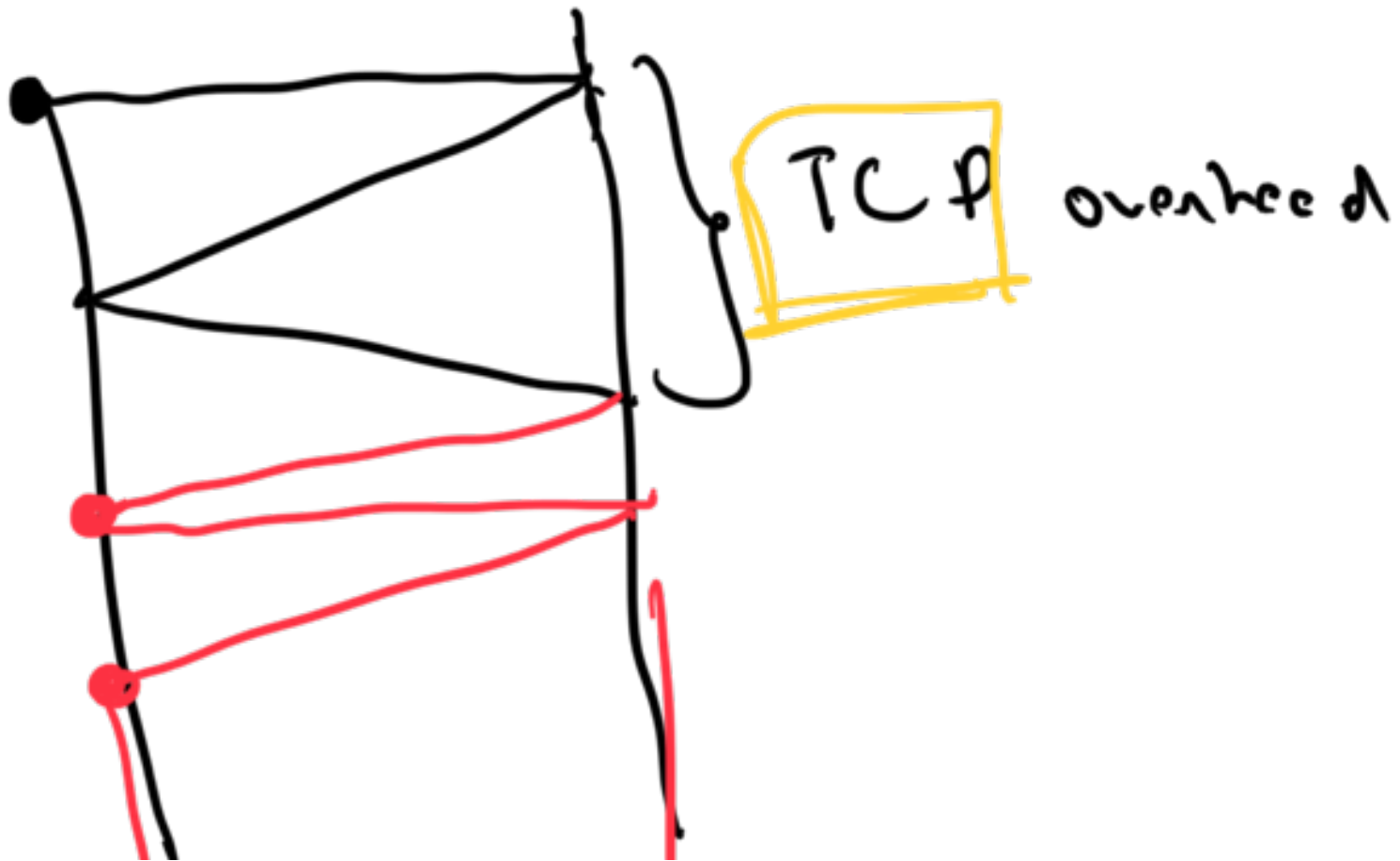
index.html



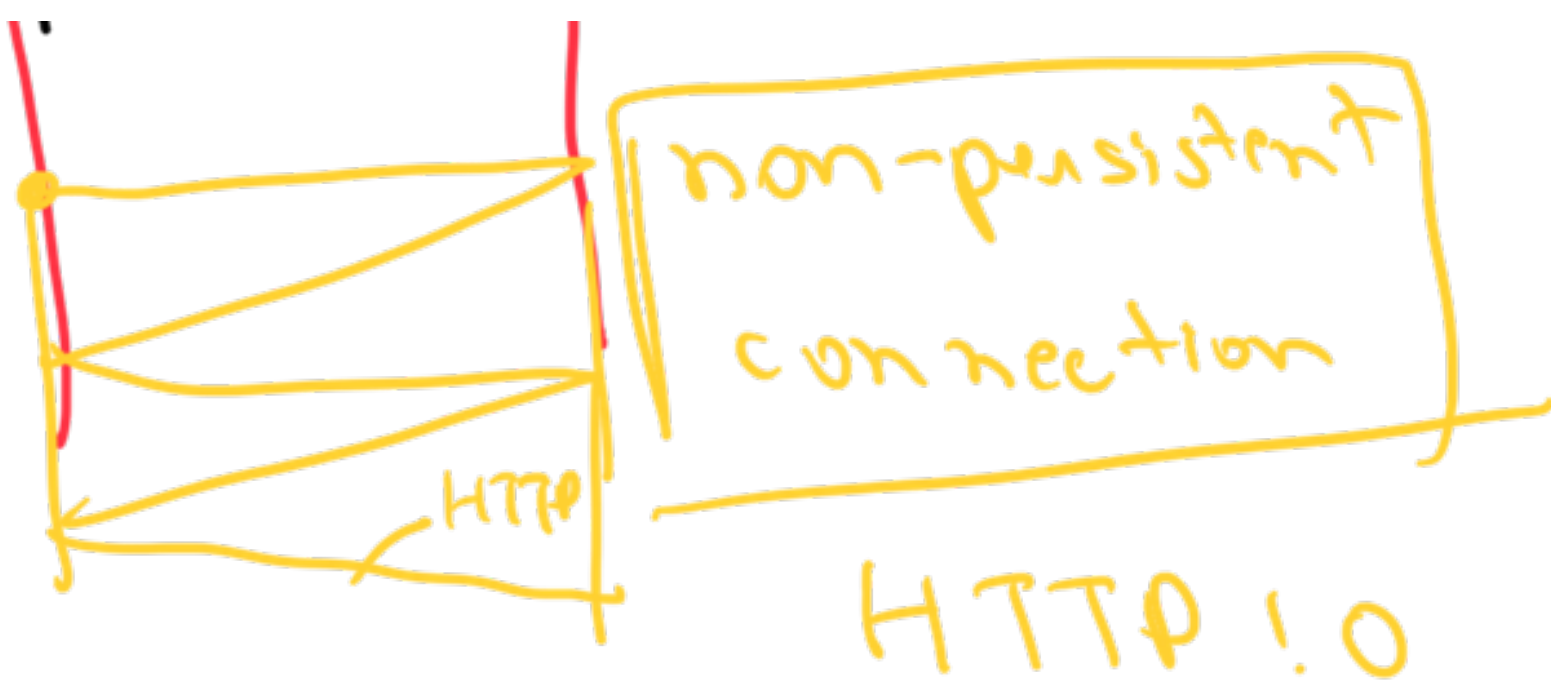


HTTP

TCP



++lo



192.1 TCP

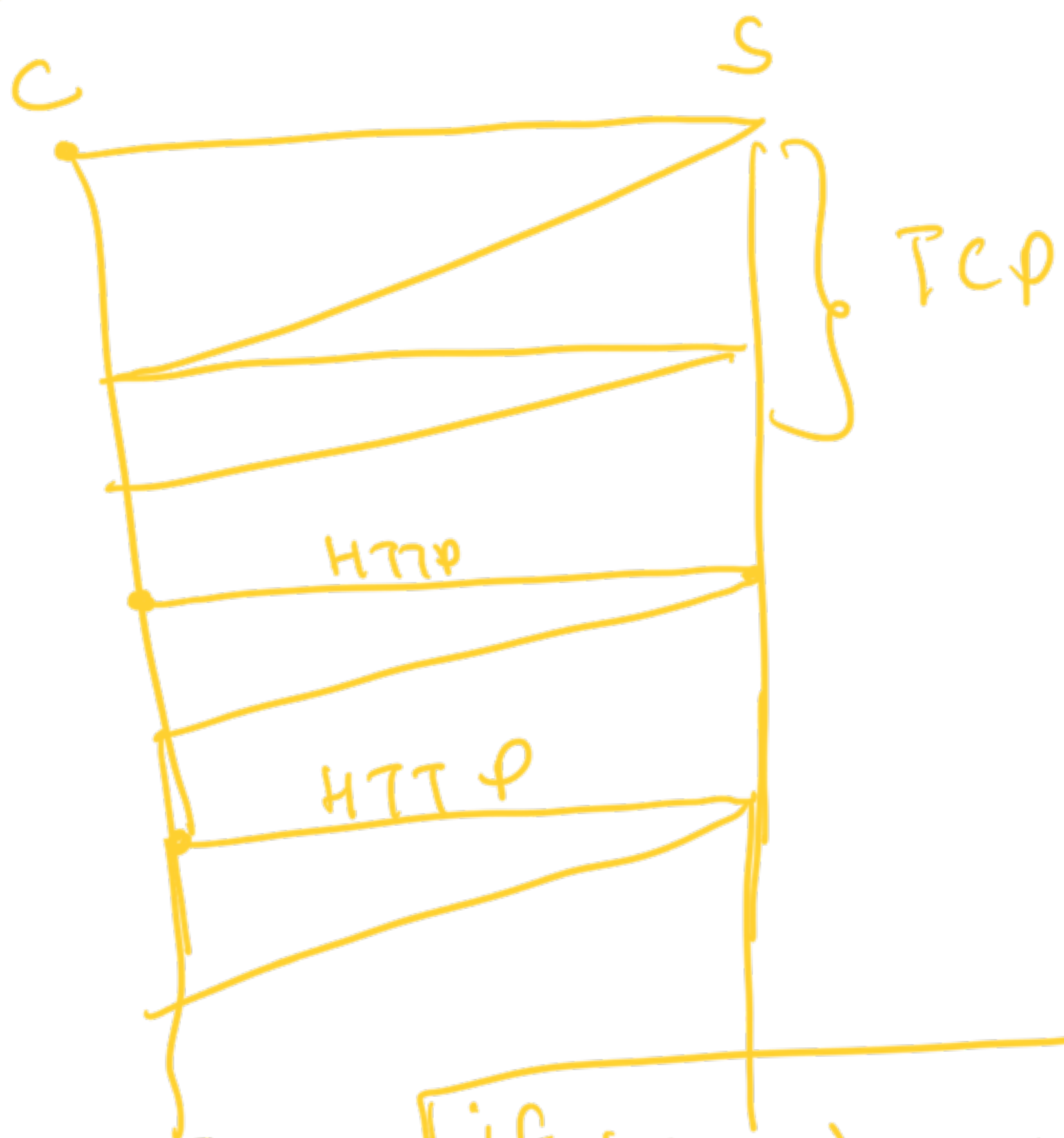
192.2

Stackful

HTTP 1.1

Persistent connections





if connection is not

HTTP!!!

closed

**Connection** : keep-alive  
: close

HTTP - Req - **HTTP method**

Resp.



# Status codes

## HTTP methods

request body

CRUD

HTTP

POST

PUT

GET

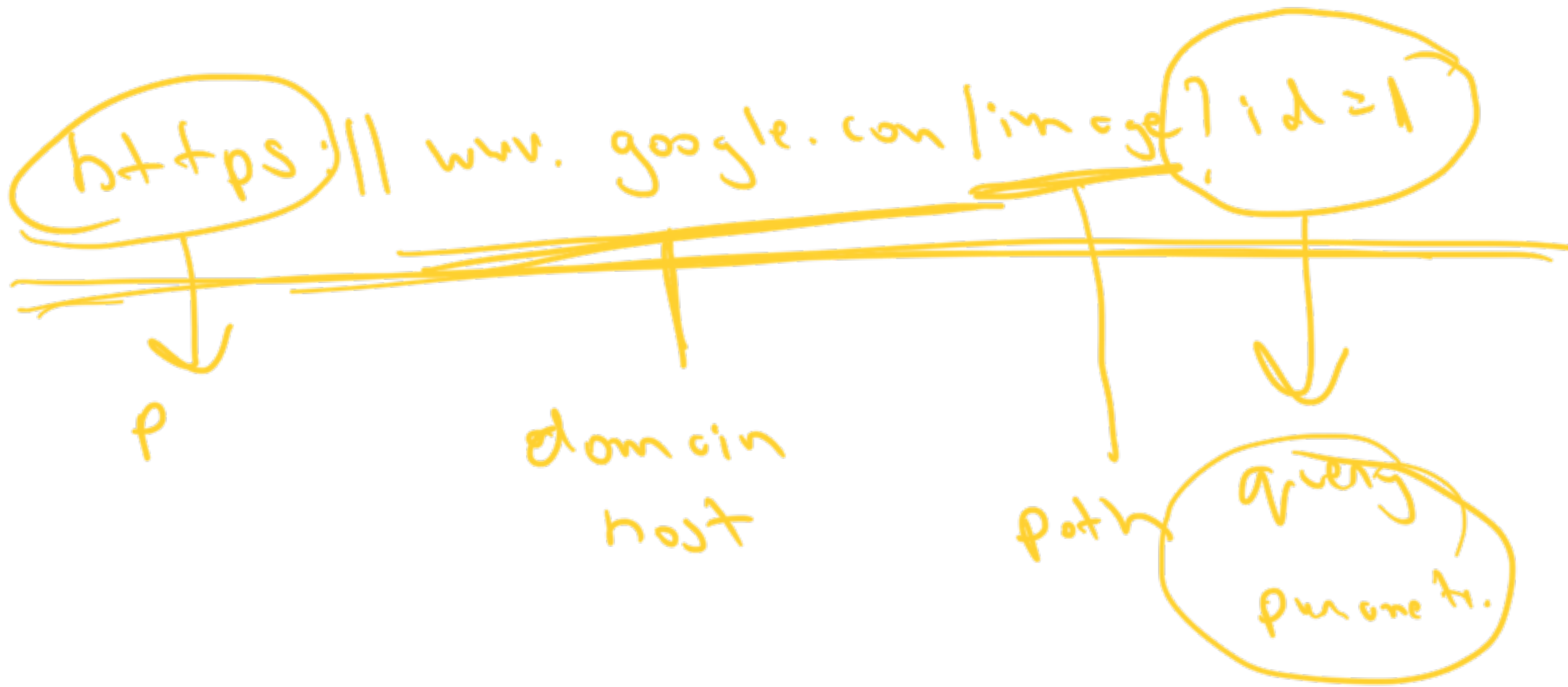
not have a body

PUT

PATCH

DELETE

- CREATE →
- READ —
- UPDATE →
- DELETE —



Request body

$\{$        $\}$  encrypted





e

↗

a: 1 put  
b: 2

↘

↗

set a=1  
set b=2

↘



# HTTP status codes

- 1xx = INFO

- 2xx = Success

- 3xx = Redirection

- 4xx = User error

- 5xx = Server error

ERROR

404 -

400 - Bad request { 10: }

200	OK
201	
204	No DATA

{ c: }

500 - ISE

502 - Bed gatenes



$\hat{1a} = 404$

So cber



HTTP

→ ~~identifiable~~

objects

↳ HTML

↳ JSON

↳ XML

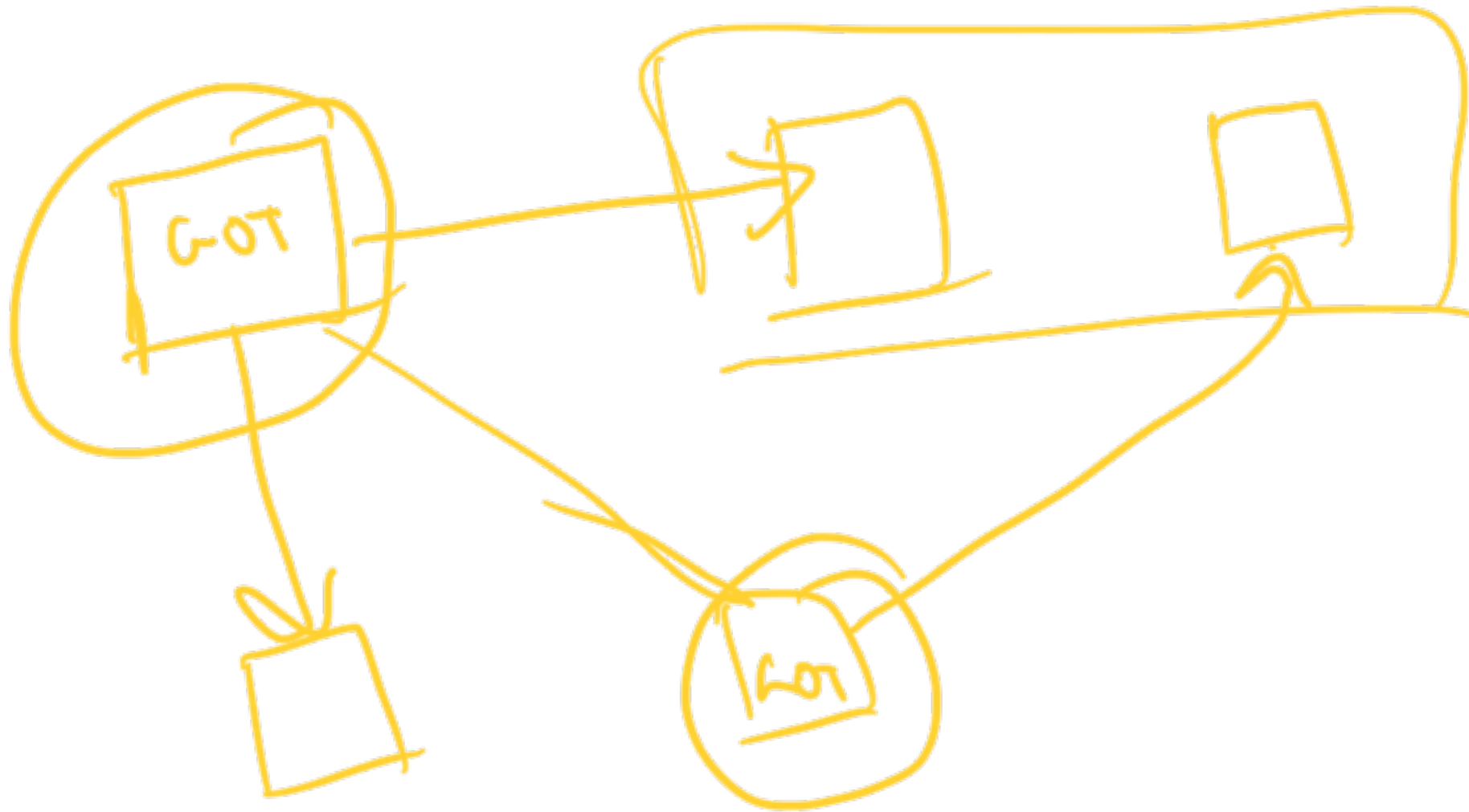
• json vs.

• doc vs .md .pdf

\*LS



XLSX



uploading

vs

downloading

seeders

leechers

REST

JSON

-

URI

SOAP

=

Envelope XML



RPC

1.1

Servers

⇒

conn:

timeout



Tomcat → Connection Timeout  
→ 

---