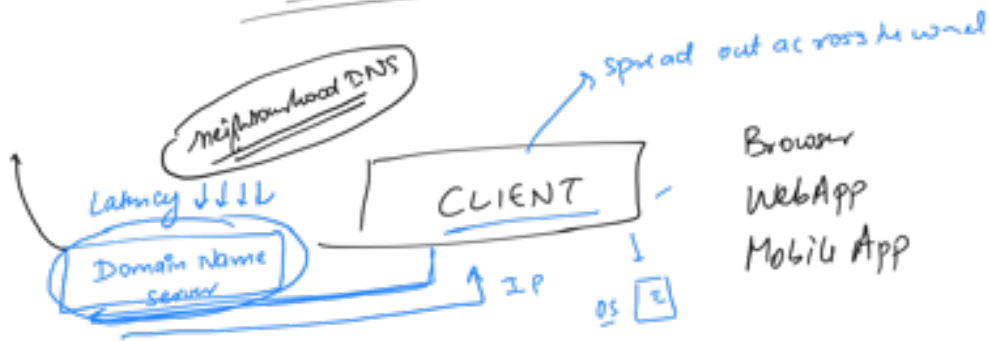
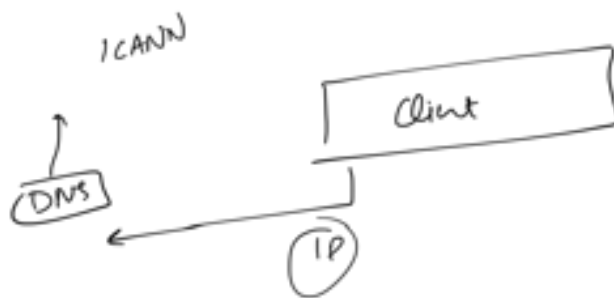
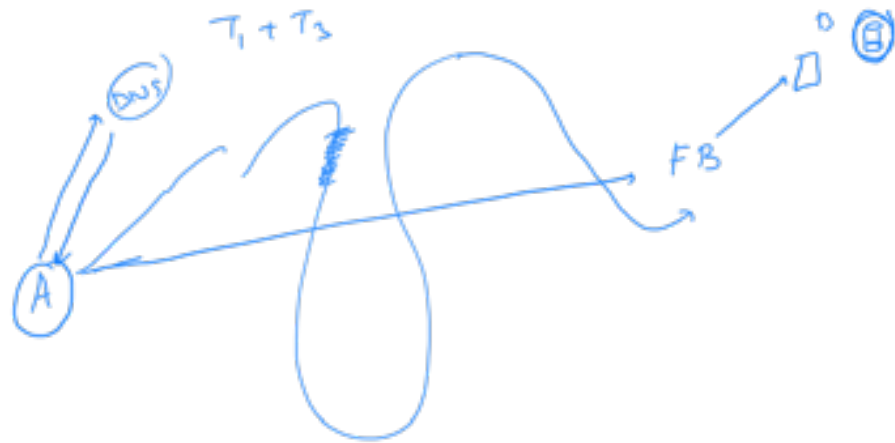
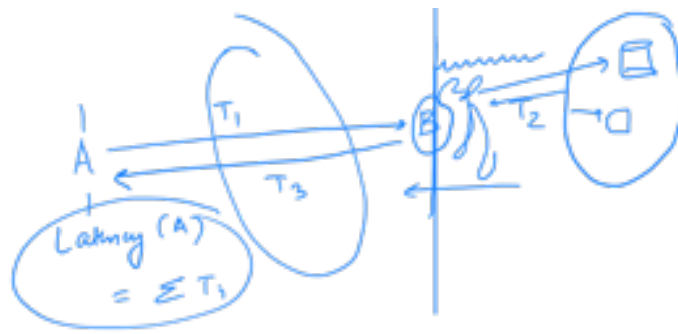


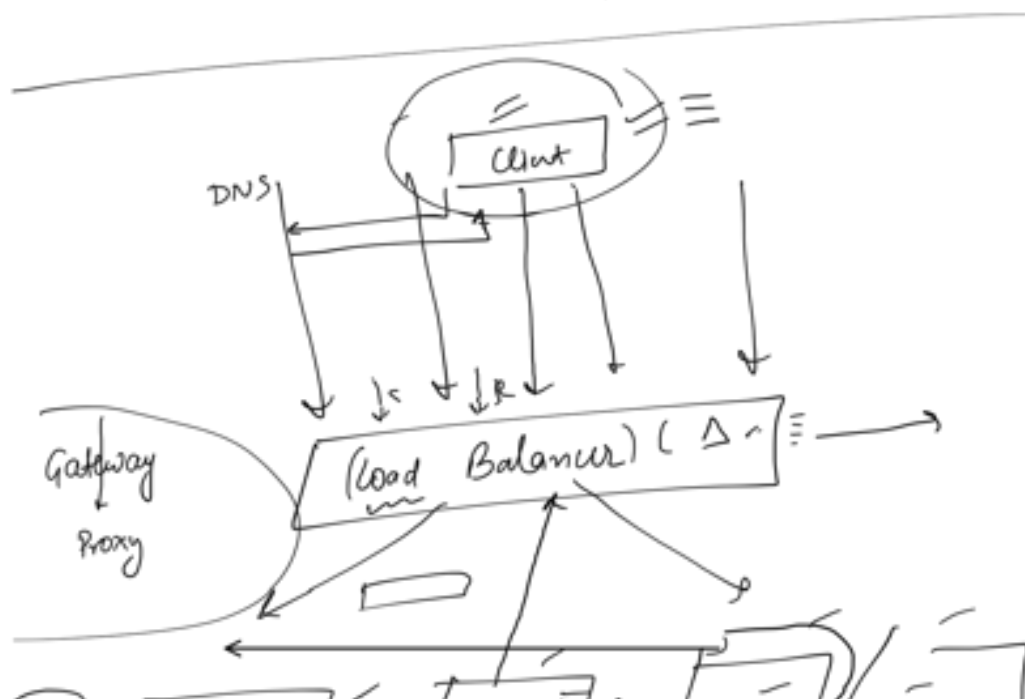
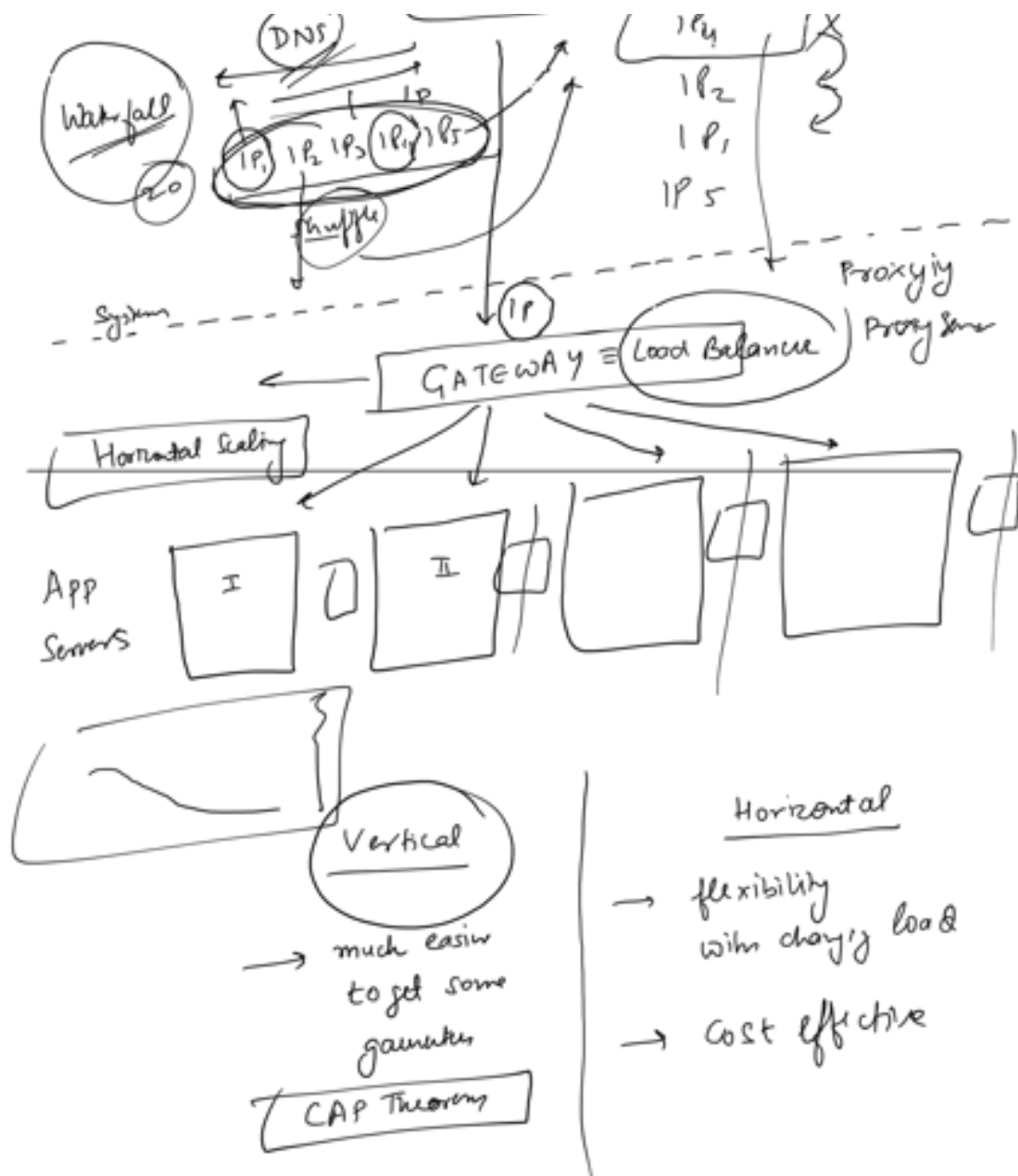
High Level Design

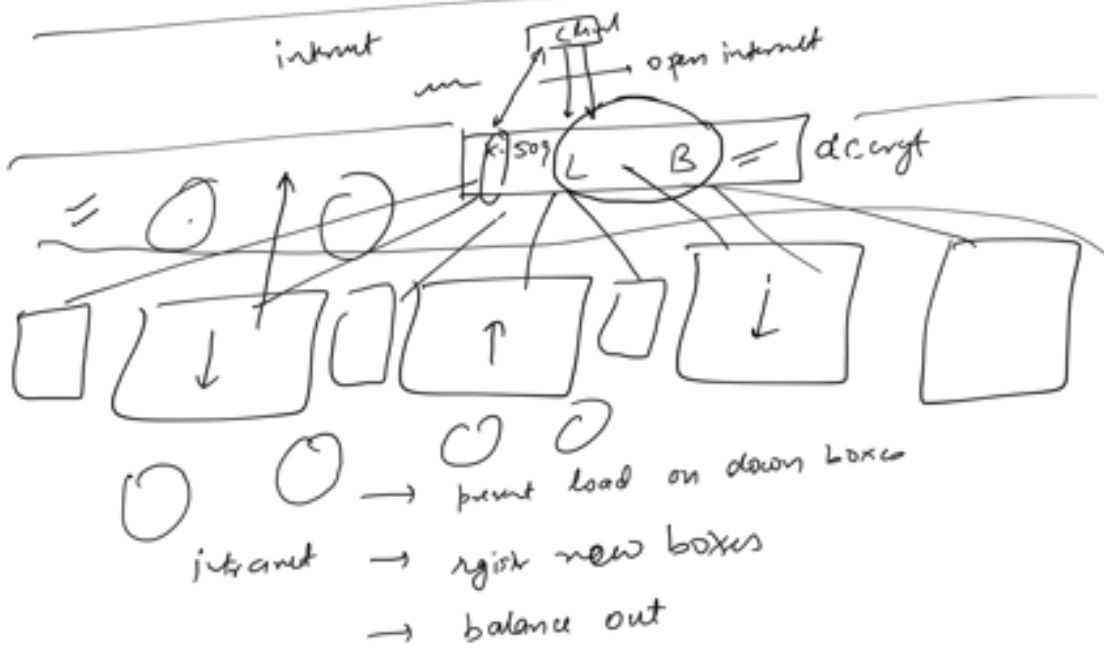


$T_1 + T_3$





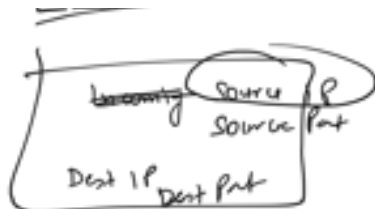




- X.509 -

Layer 4 ≡ Transport Layer

additional security



fast
simple

✓ Layer 7 LB \equiv Application

user id \equiv 707

uid = 107

✓ much better flexibility

Slower



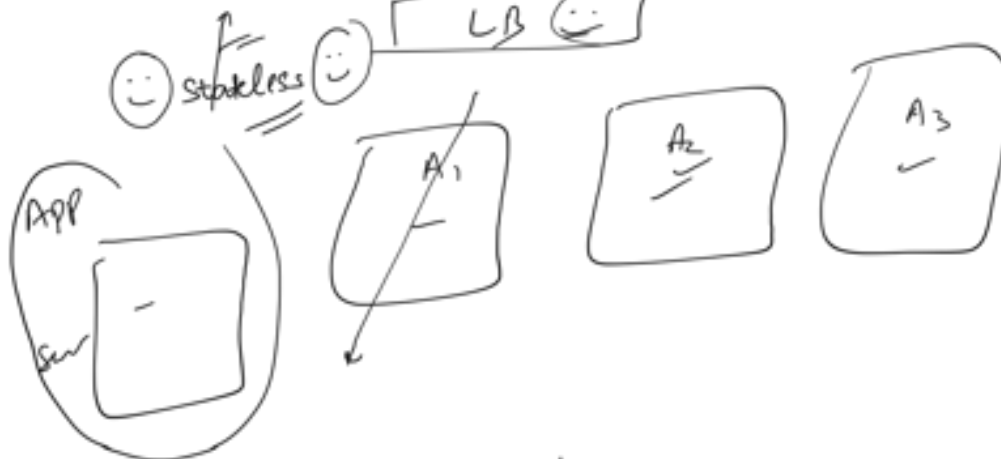
Stateful

Stateless

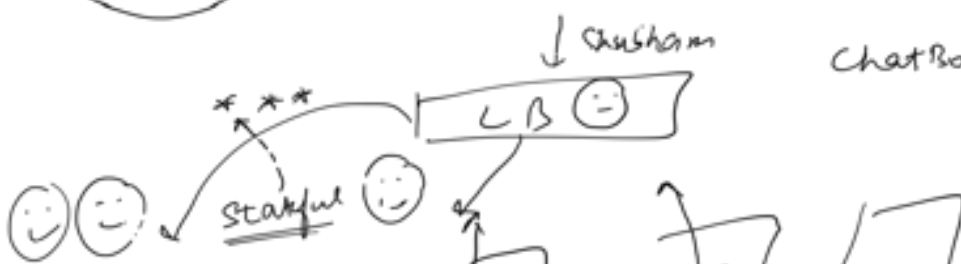
177/15
7+5
a+b

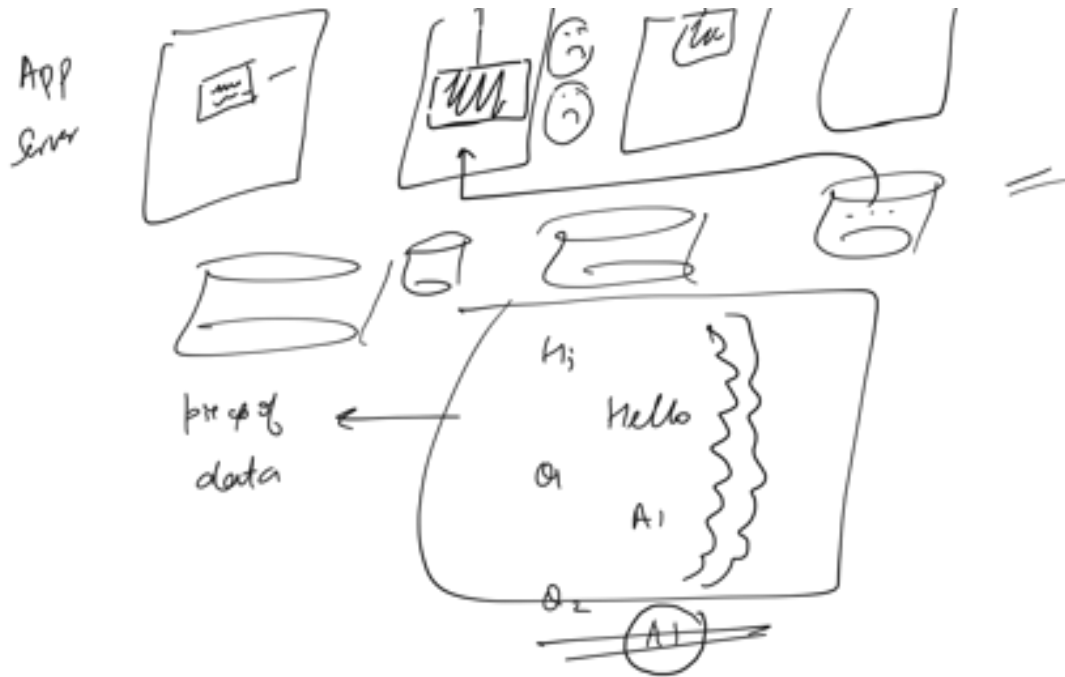
LB

Calculator



ChatBot

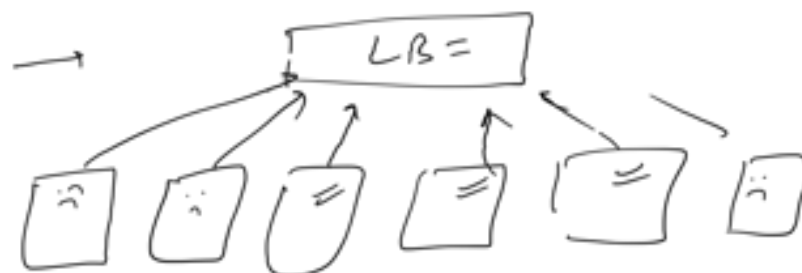




Load Balancing

Startlen

→ Round Robin



WRR
→ (Least Connection first)

→ WRR ARR



Load Balancing in Statfull





Sol ① Maindary Map **BAD**

map-size =
order of key-size

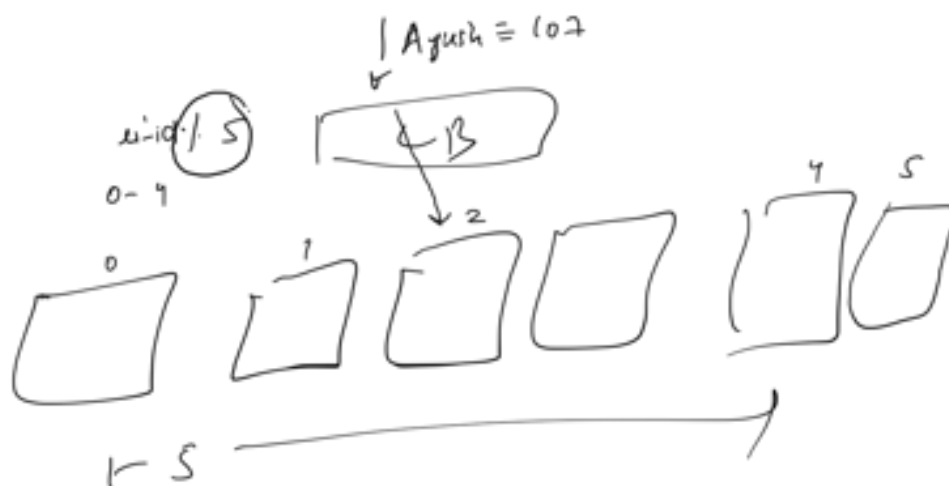
Load Balancing func
will slow down

Sol ② Hash

BAD!!!

10 boxes \equiv 10 App Sene

$$uid \cdot / . 10 = \overset{0-9}{\square}$$



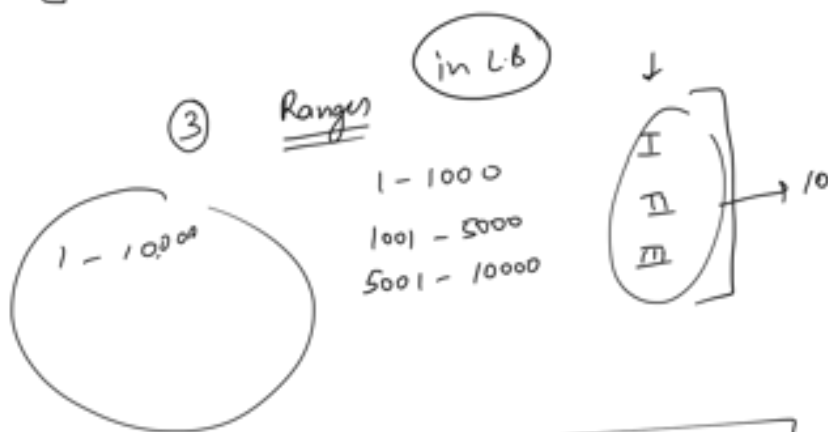
\rightarrow gaurat of stickiness
vsy hashing

\rightarrow LB 😊

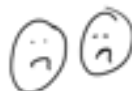
no unmdr

On server inc/dec
you will have
to move state of all boxes
internally

1	1.5	→	1	1.6	1
2		→	2		2
3		→	3		3
4		→	4		4
5		→	5		5
6		→	6		6
7		→	7		7
8		→	8		8
9		→	9		9



Bottleneck: Add/Removal
of items is
inefficient



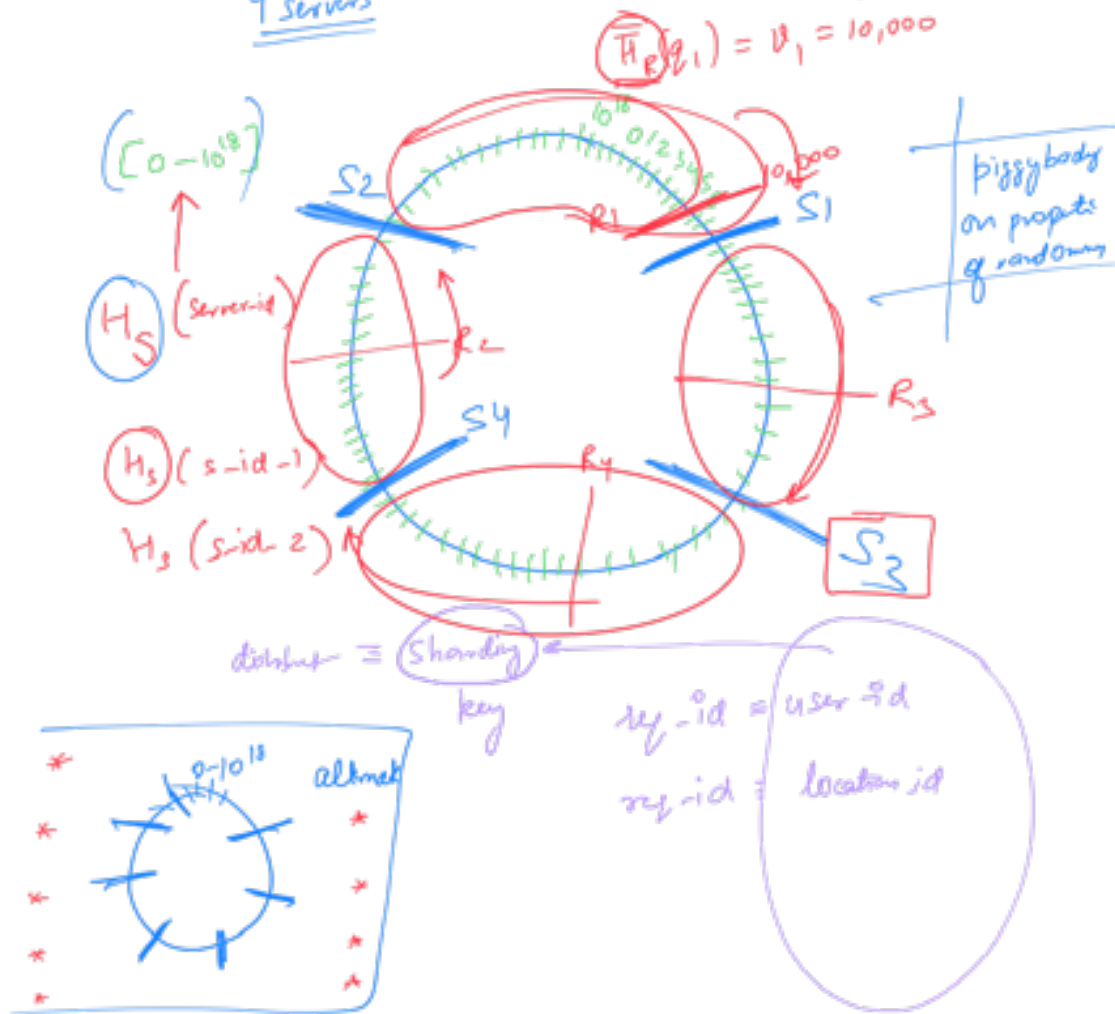
④

Consistent Hashing

LB for Stateful Application

const addition/removal
of servers

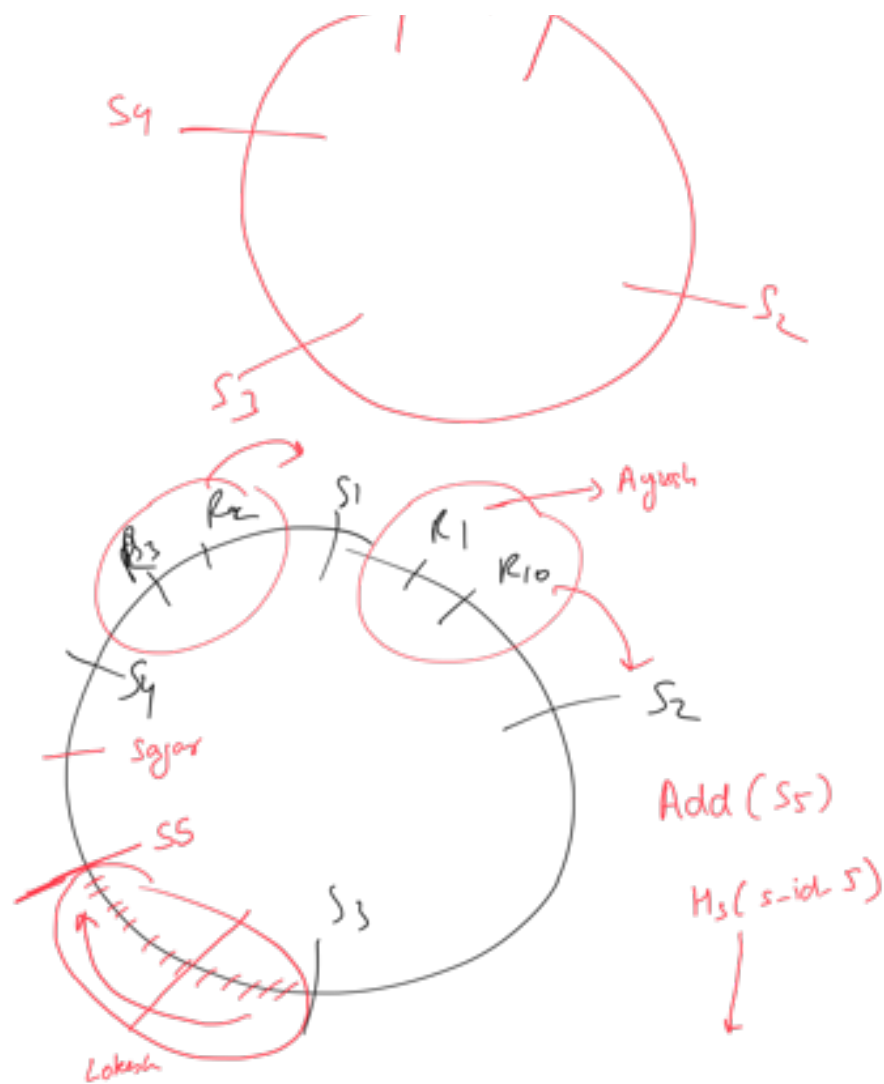
4 servers



$$\left(\begin{matrix} H_R \\ = \end{matrix} \right) (rq-id)$$

$[0 - 10^{18}]$





✓ (I)

Amount of state transfer that happens
has been reduced a lot

😊😊

😊😊 by a factor of # servers 😊

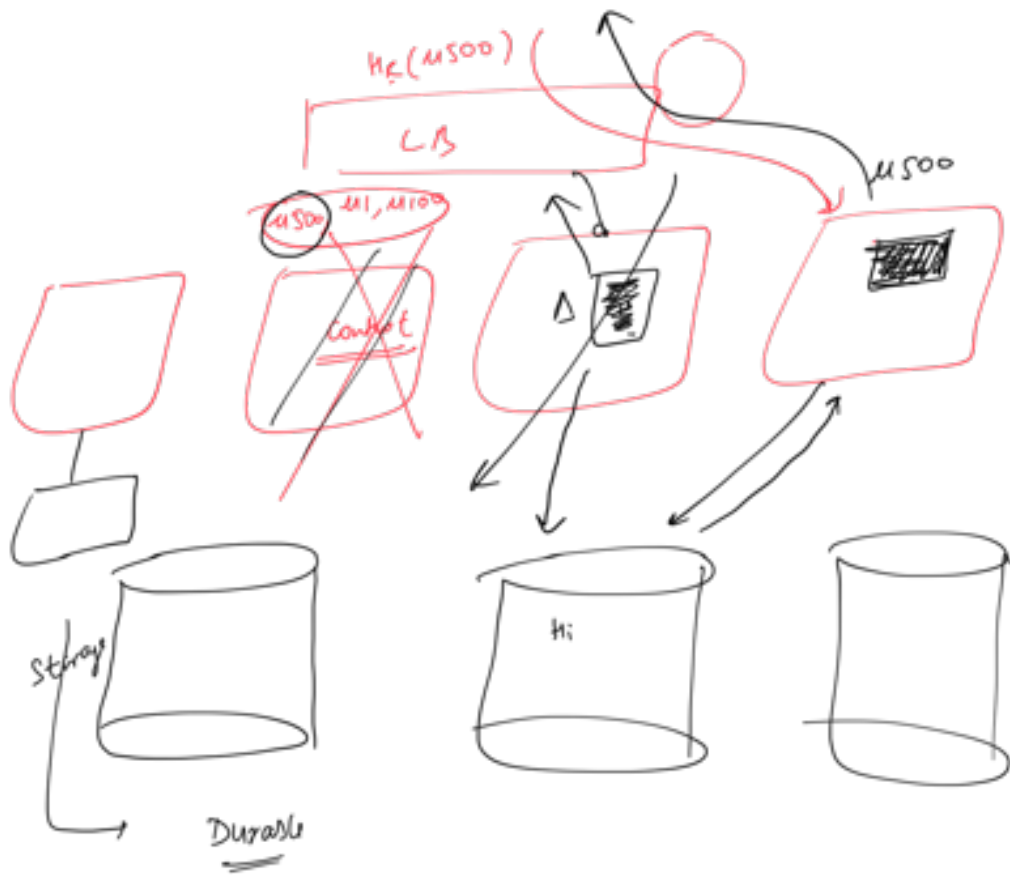
✓ (II)

downtime will also be low.....

✓ (III)

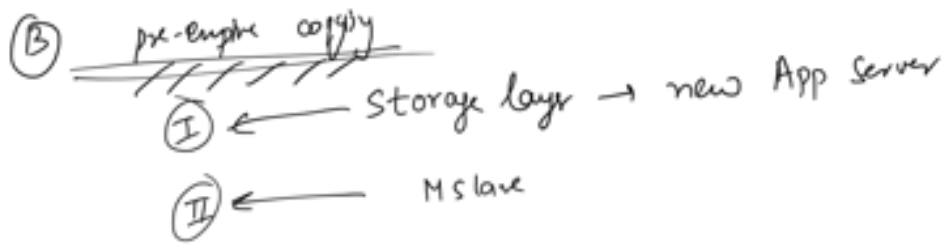
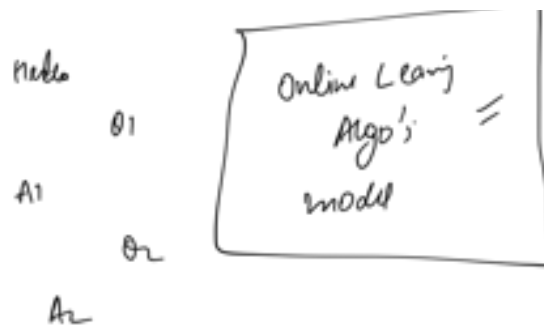
you don't have downtime for every server. Small set of users get impacted





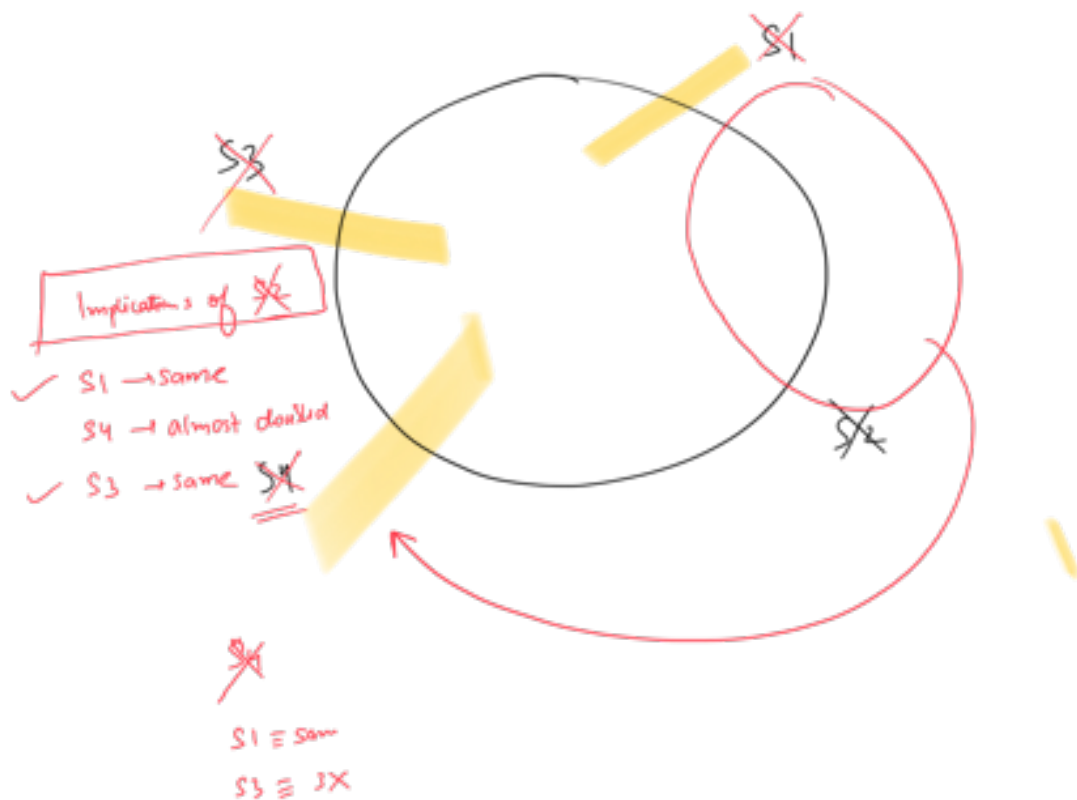
→ we are not losing the ultimate source of truth

(A) Cold start \equiv requests latency inc for some time
Hi

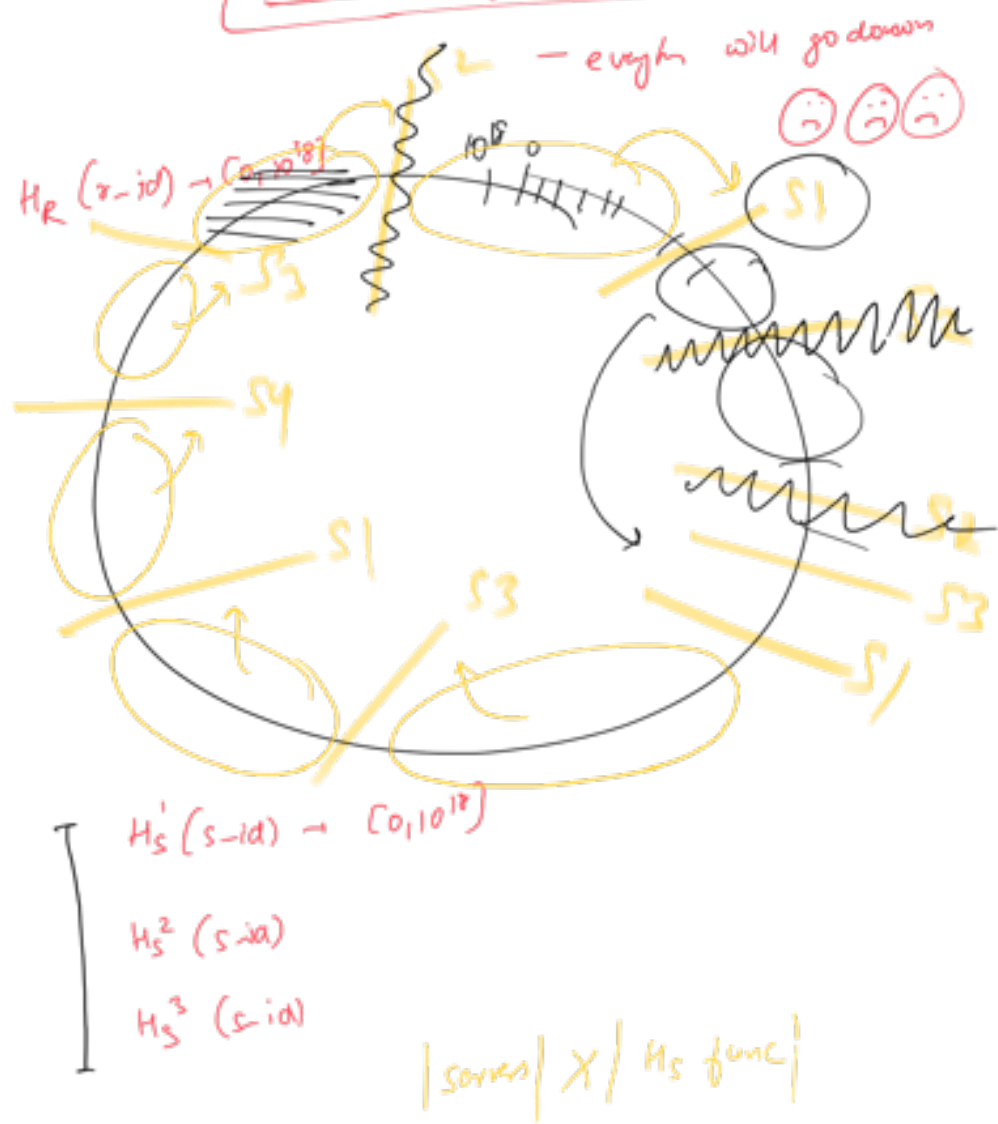


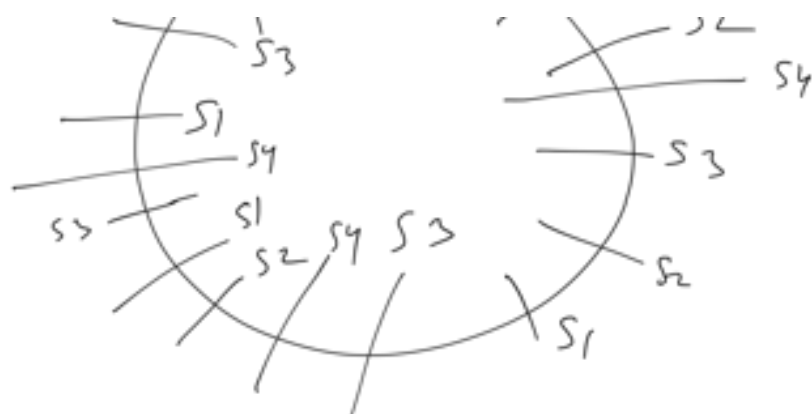
Solved problem

impact of server addition or removal
 is minimized

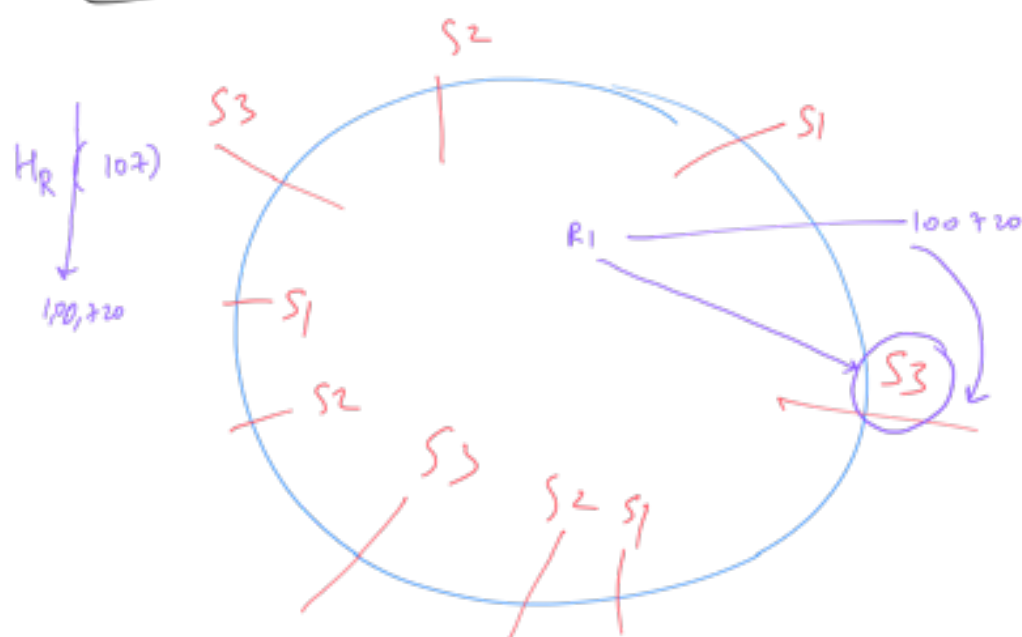


"Cascading Failures"





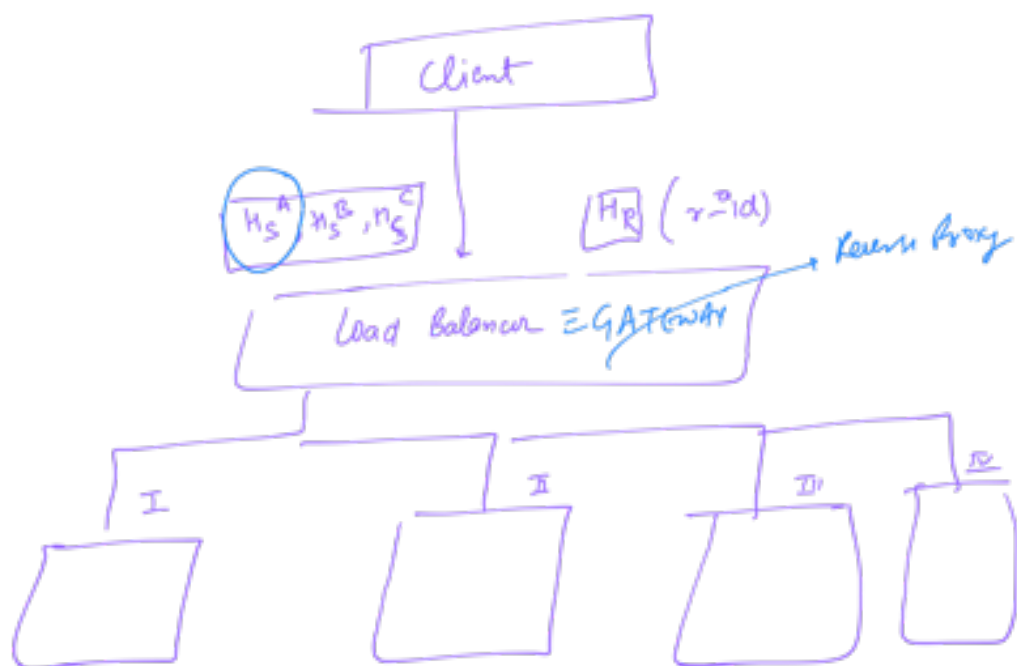
Solved Cascading Failures
 [Now, the load will get divided almost equally.]



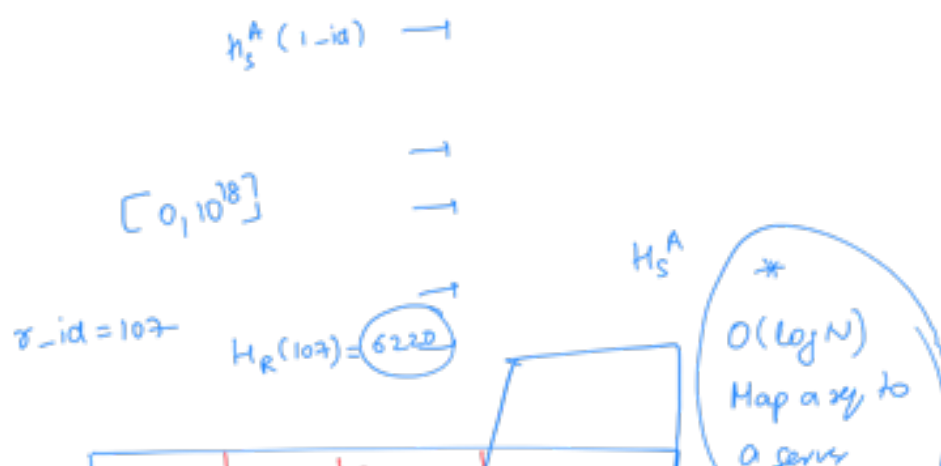
$|H_S| \uparrow \uparrow \equiv \text{Randomness} \uparrow \uparrow$
 \downarrow
 Equitable Distribution $\uparrow \uparrow$

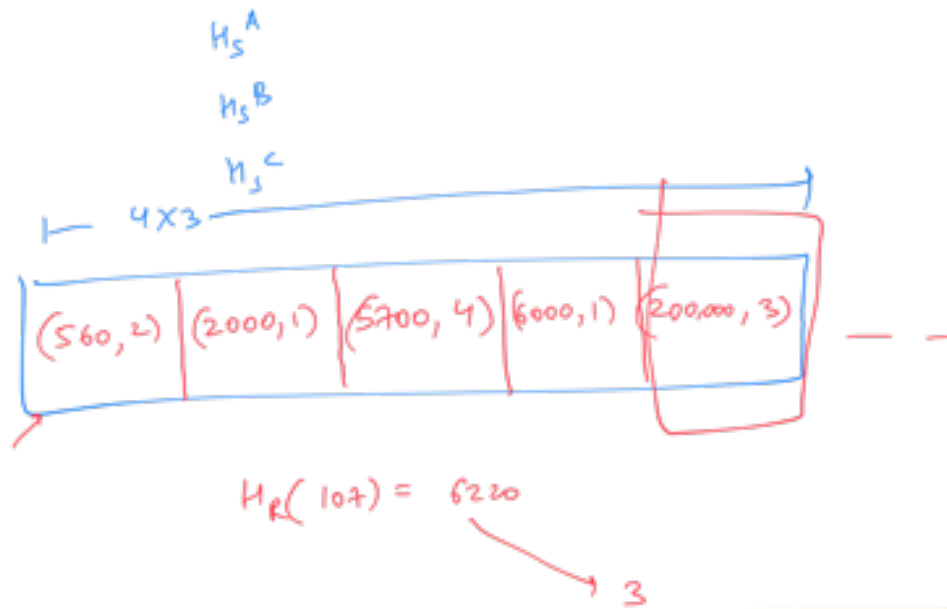
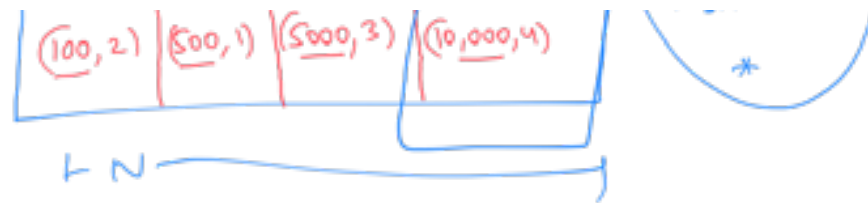
$|H_S| \uparrow \uparrow \uparrow \equiv \text{Time for Resolution} \uparrow \uparrow \uparrow$

ASG



10 servers

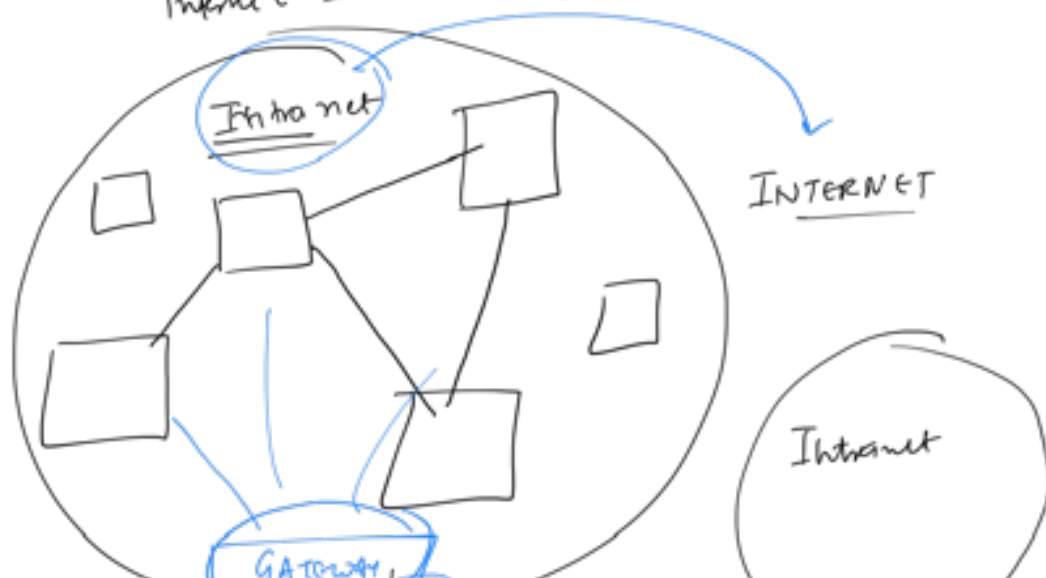


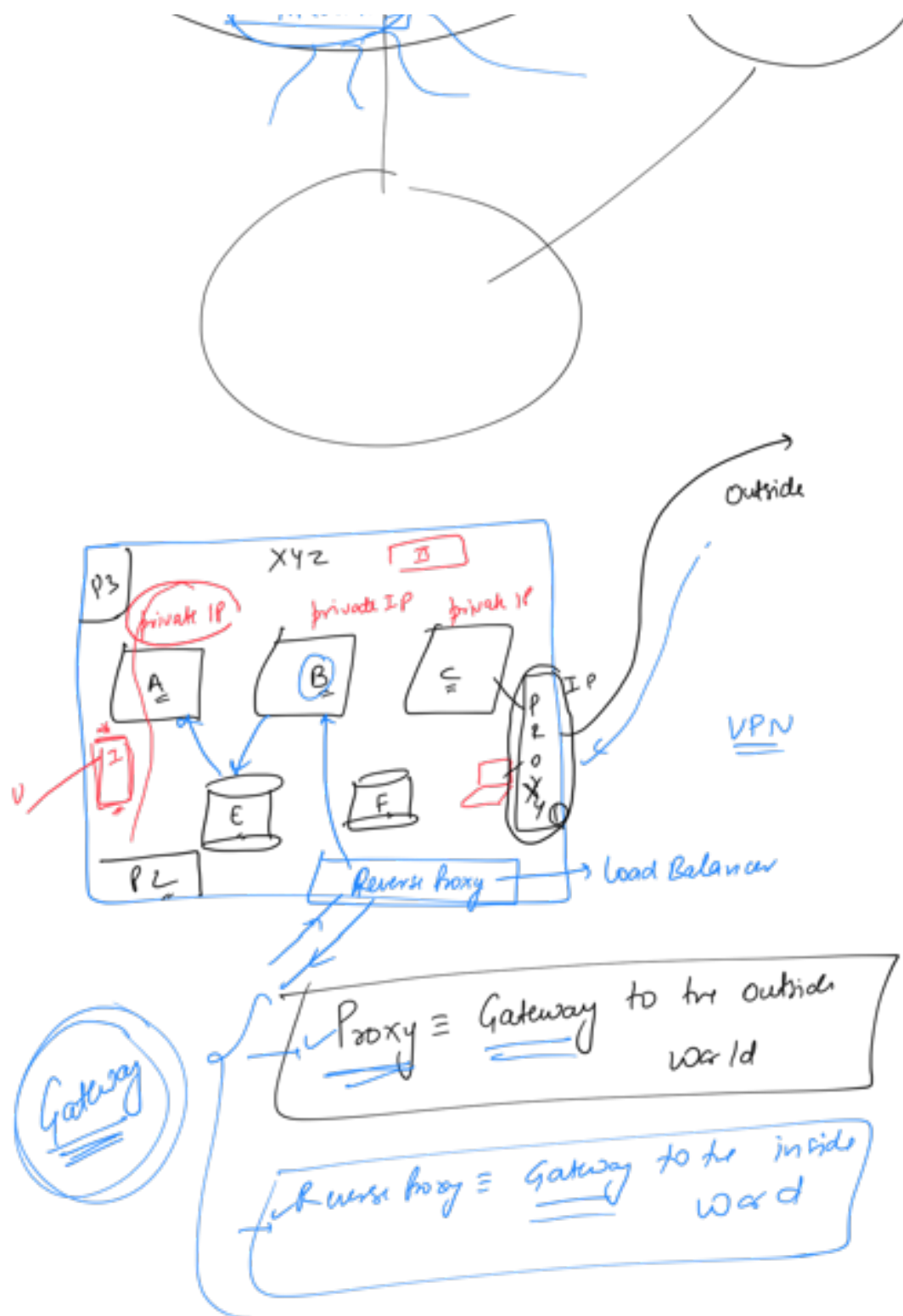


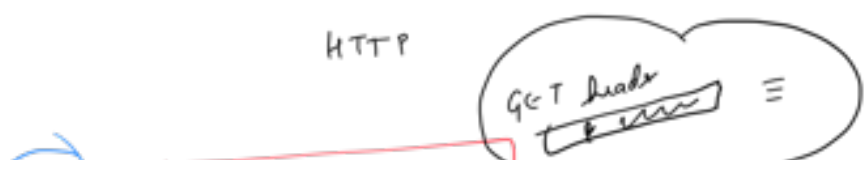
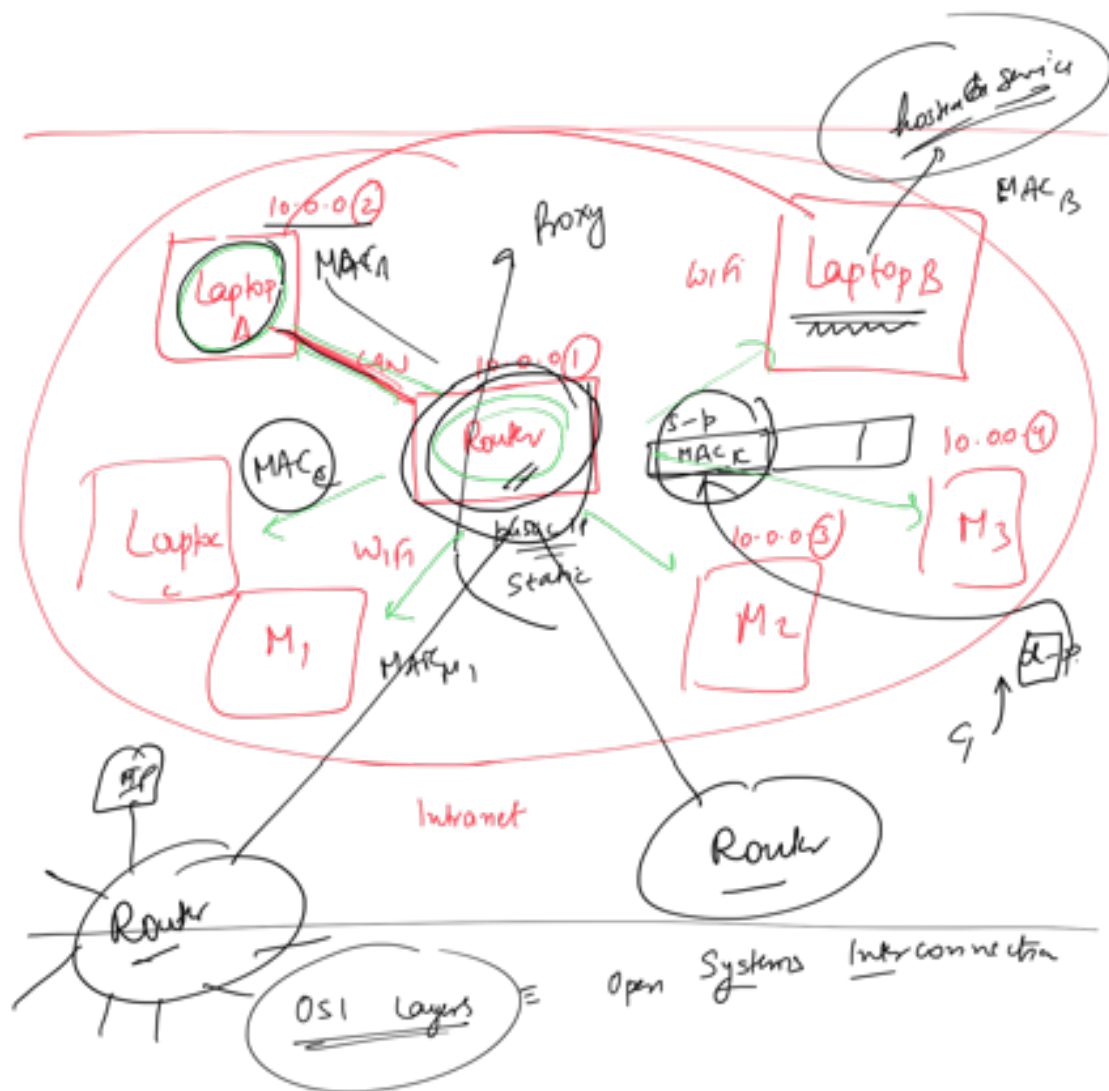
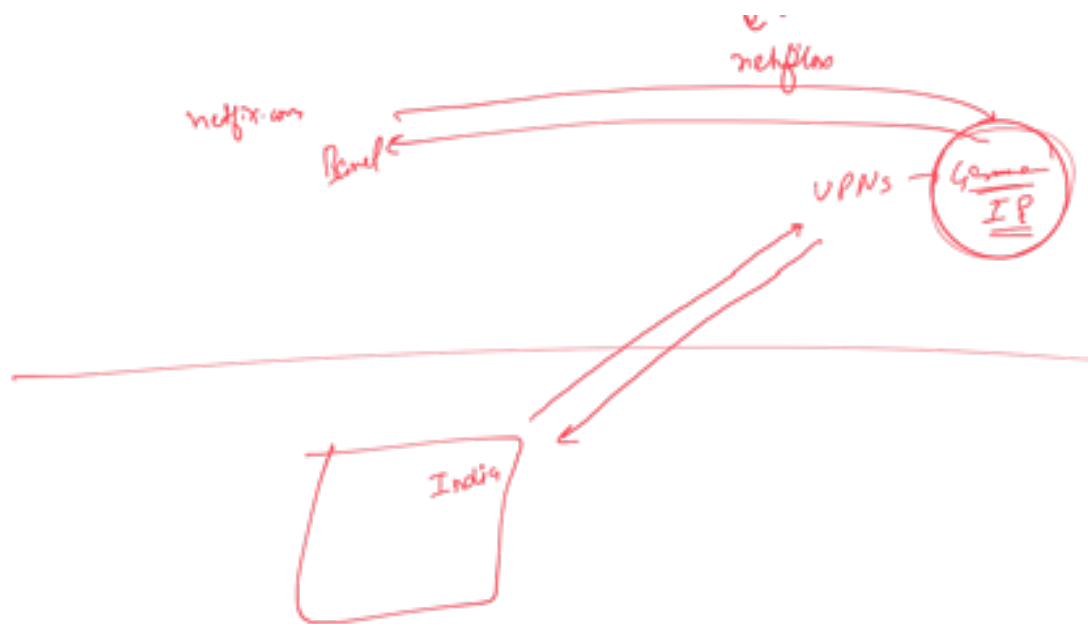
High Level Design

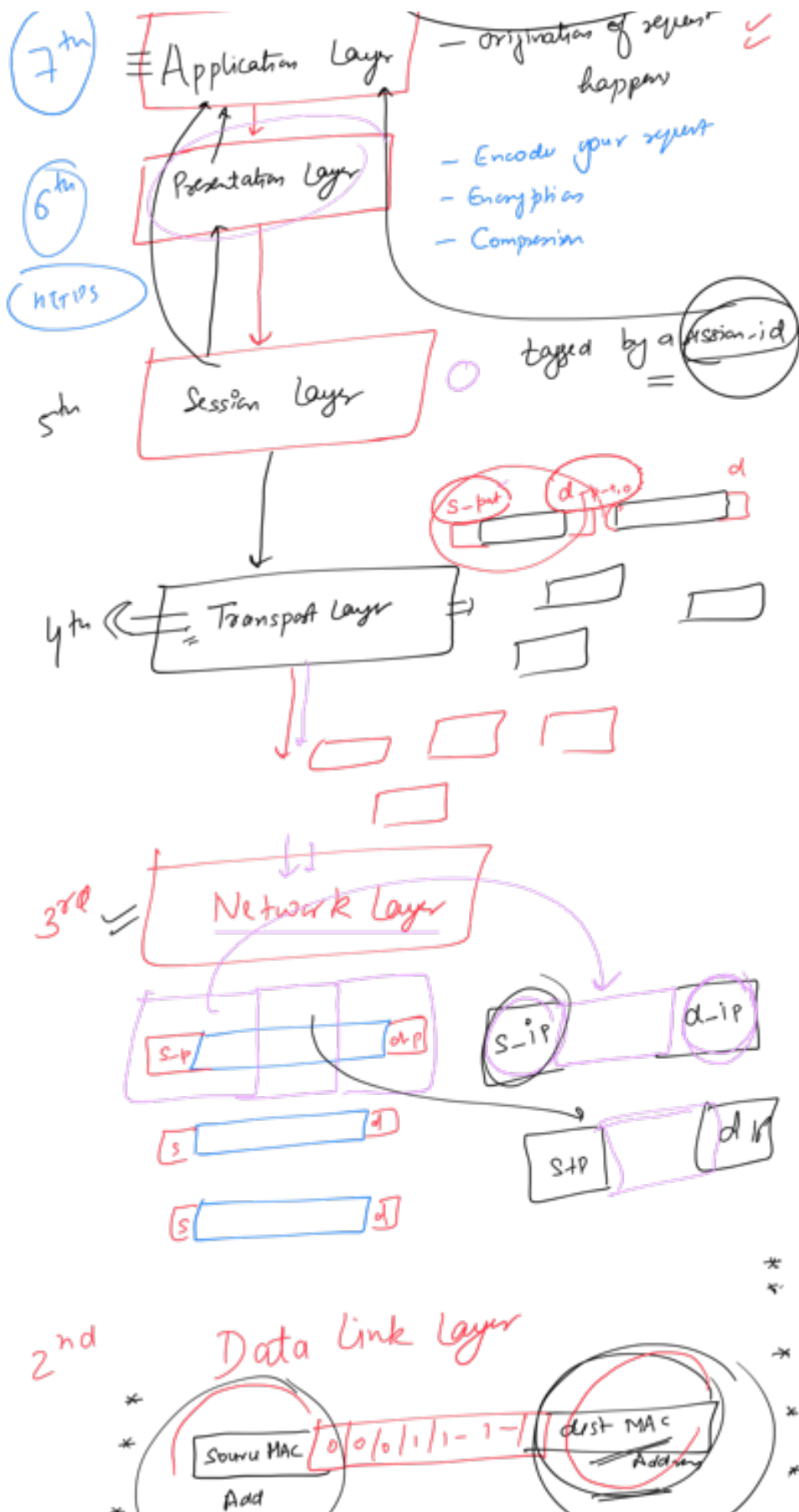
OSI Layers

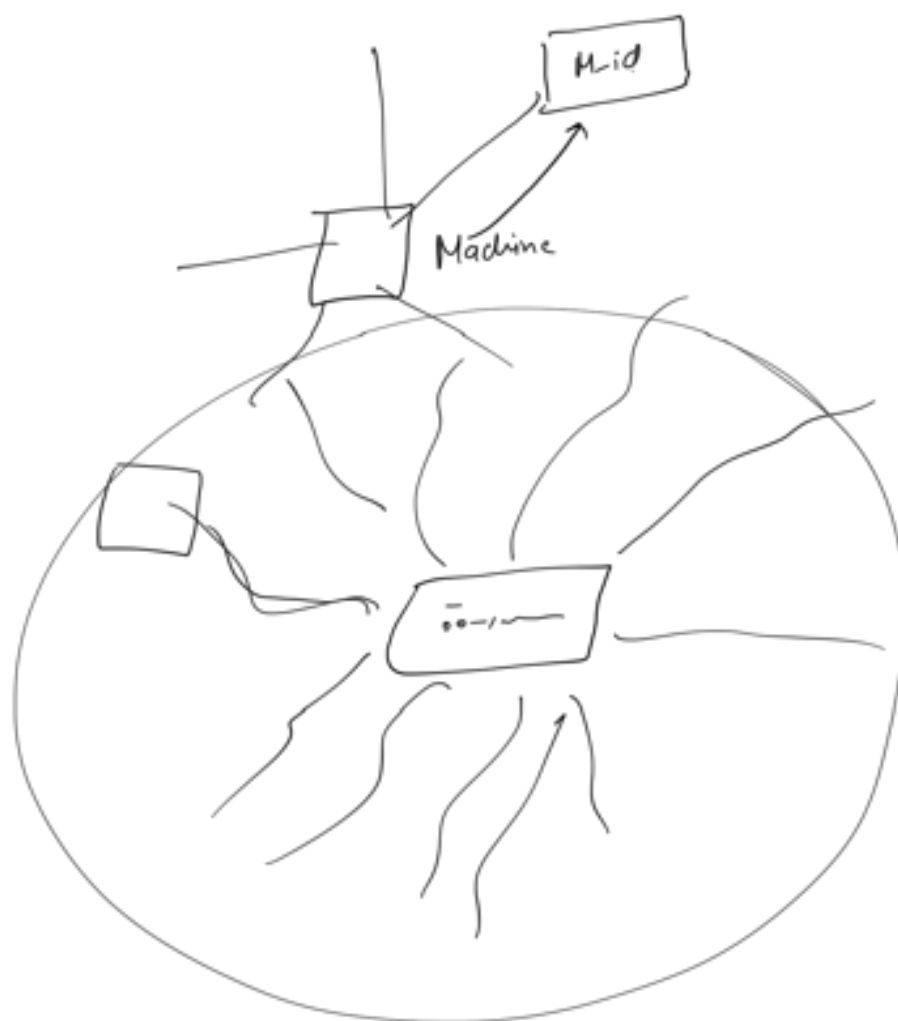
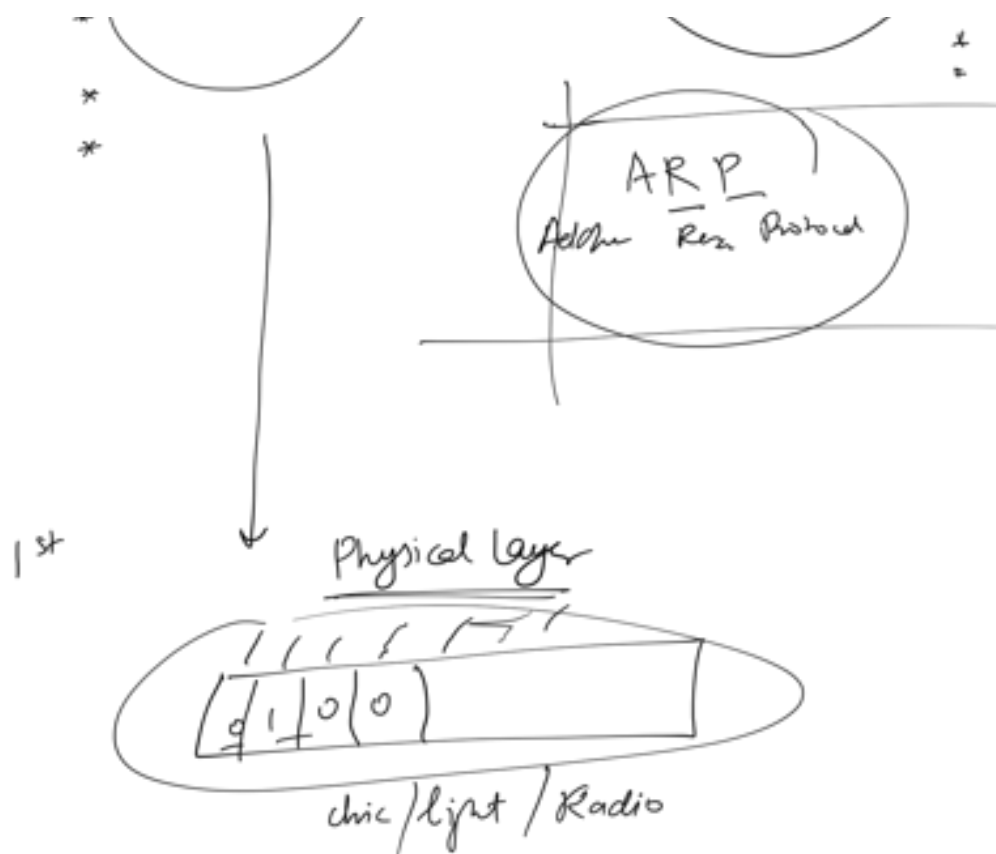
Internet \equiv network of networks

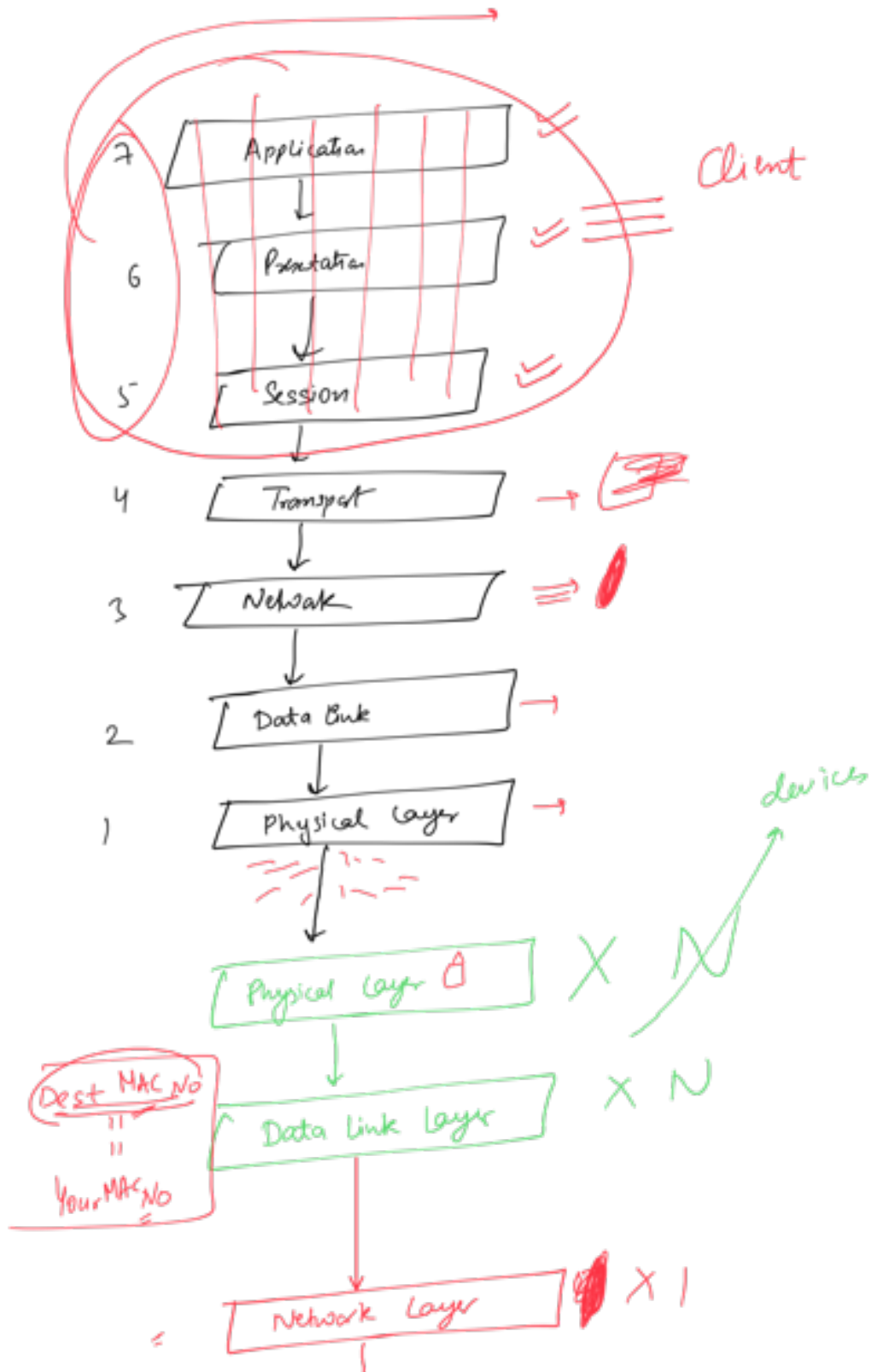
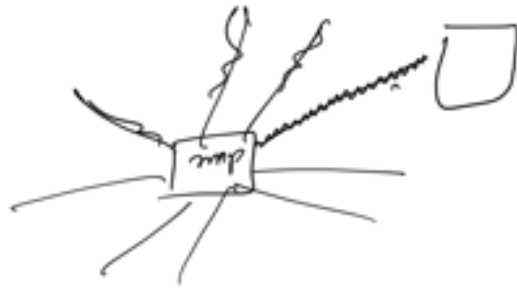


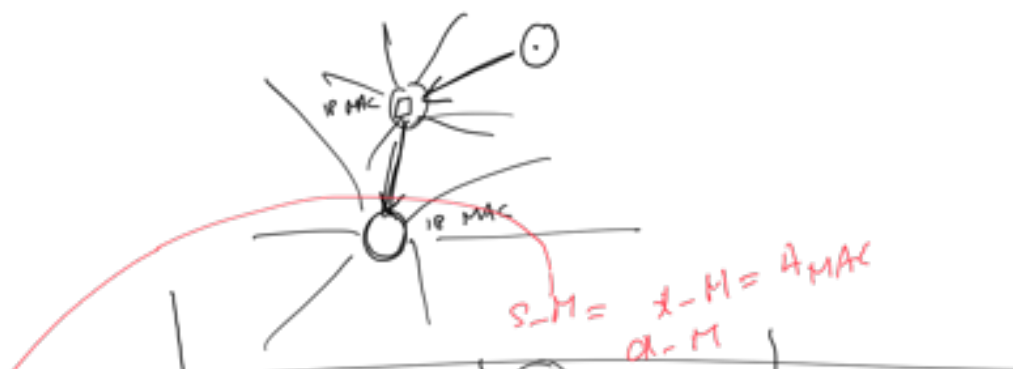
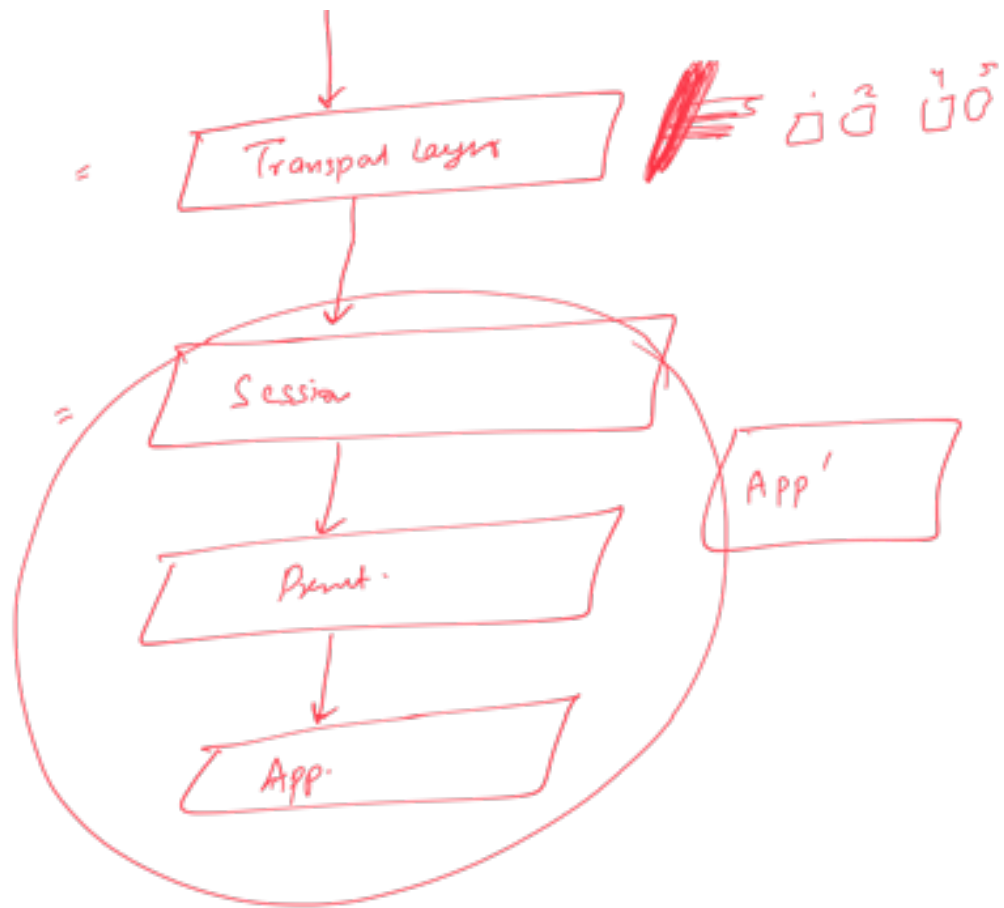


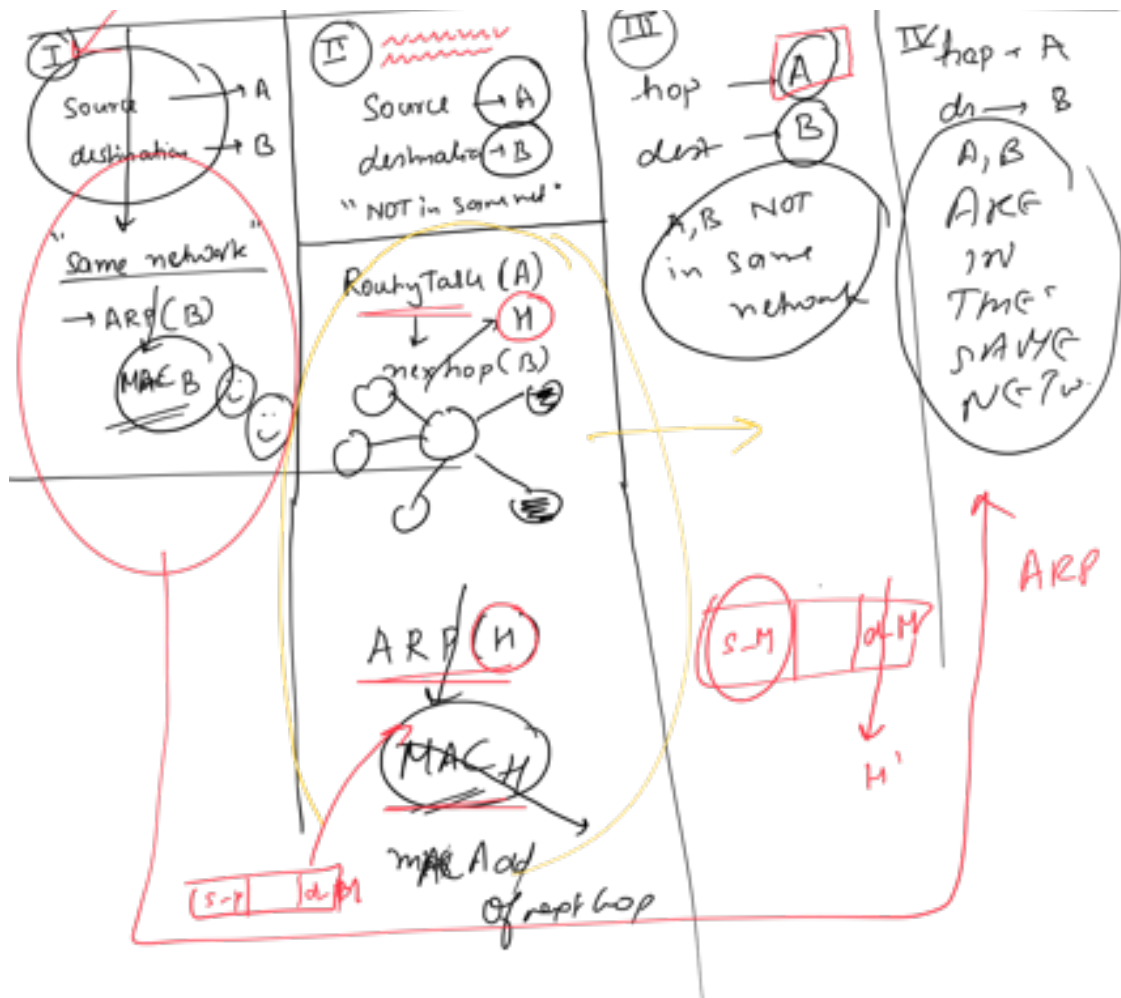












* CAP THEOREM *

(only guarantee $\frac{2}{3}$ properties simultaneously)

*	<u>$C \equiv$ Consistency</u>
*	<u>$A \equiv$ Availability</u>
*	<u>$P \equiv$ Partition Tolerance</u>

(I) Consistency → Immediate Consistency

Any read that you do, must return the value
after the latest write

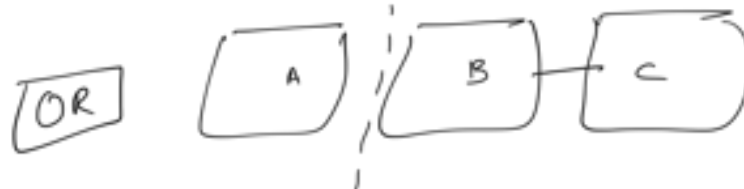
(II) Availability



Every and any request that goes to a
non-dead server, must be responded to

(III) Partition Tolerance

The system as a whole is allowed to
have partitions.

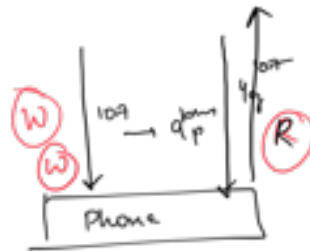


You have accepted the fact
that network partitions can happen at any
time

... the system as a whole will NOT

and in the
stop functioning

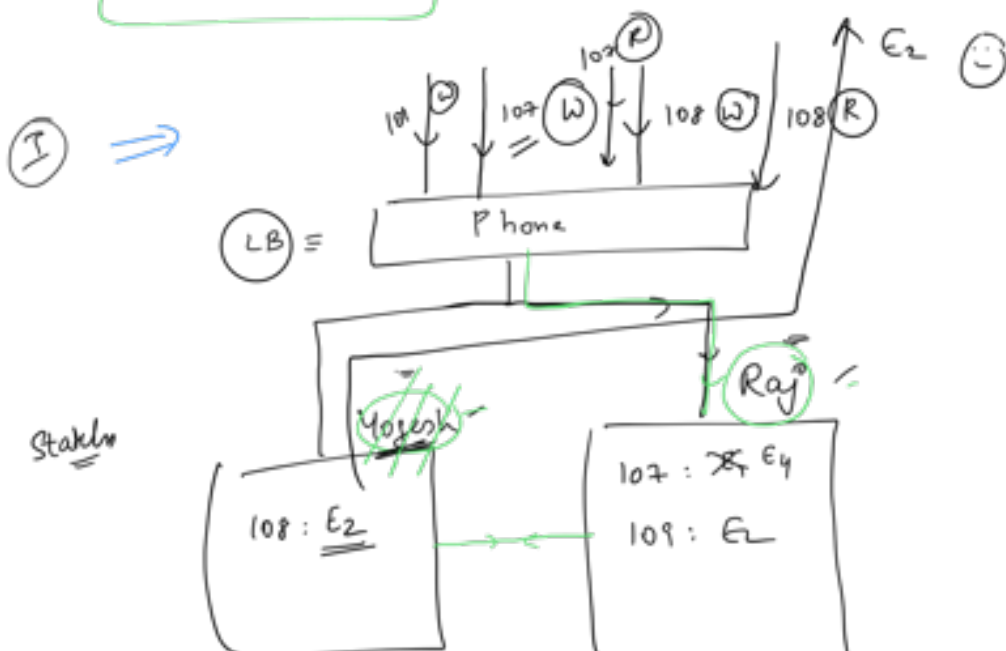
Event Reminder Service



DOES NOT Require
 Partition Tolerance

Consistent

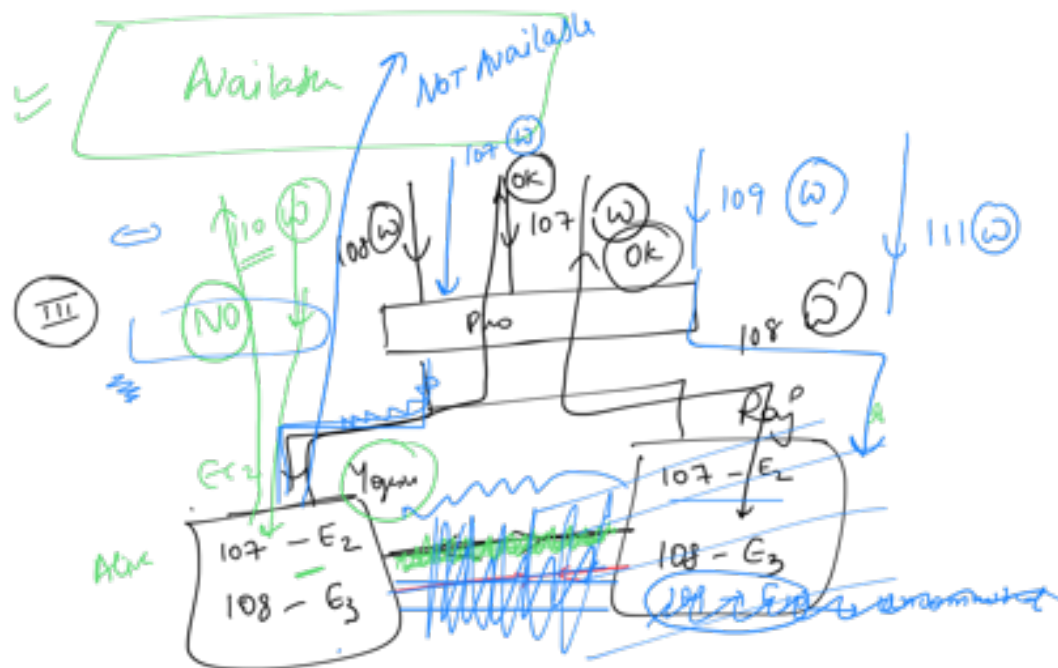
Available



Position Tolerance

NOT CONSISTENT

$\frac{2}{3}$



CONSISTENT

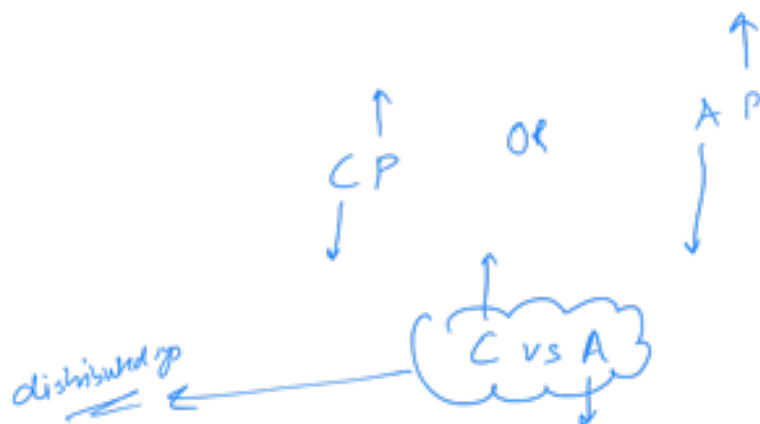
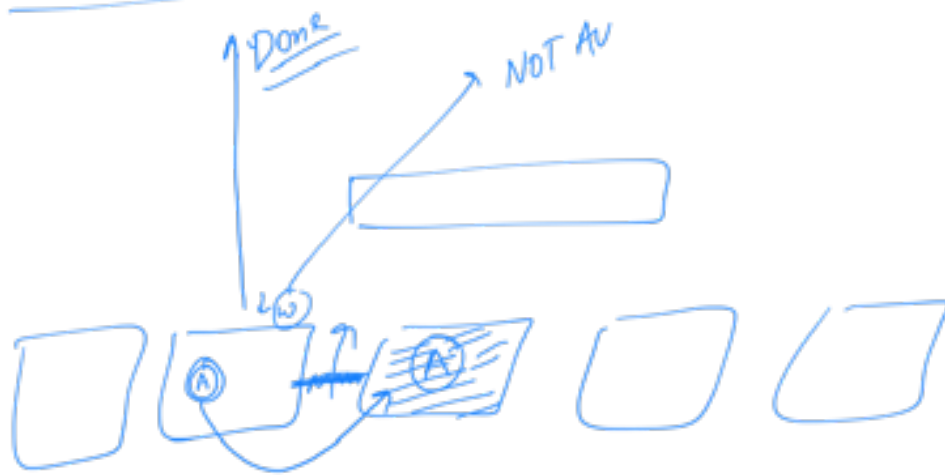
PARTITION TOLERANT

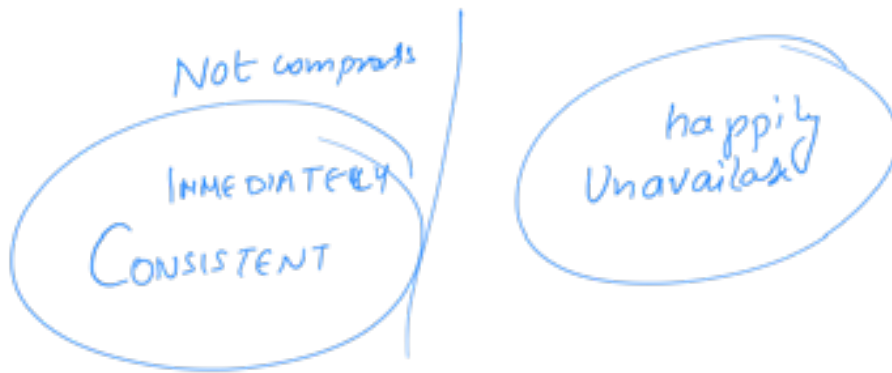
NOT AVAILABLE

CA or AP

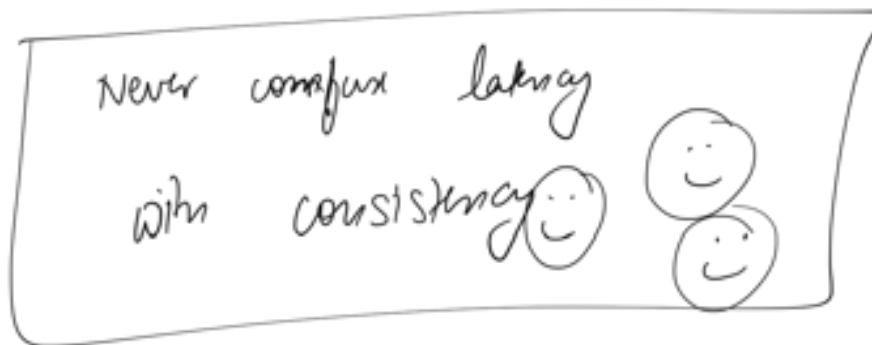
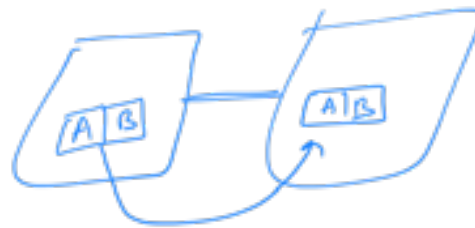
→ (CP) vs ()

— CA — if you are doing vertical scaling



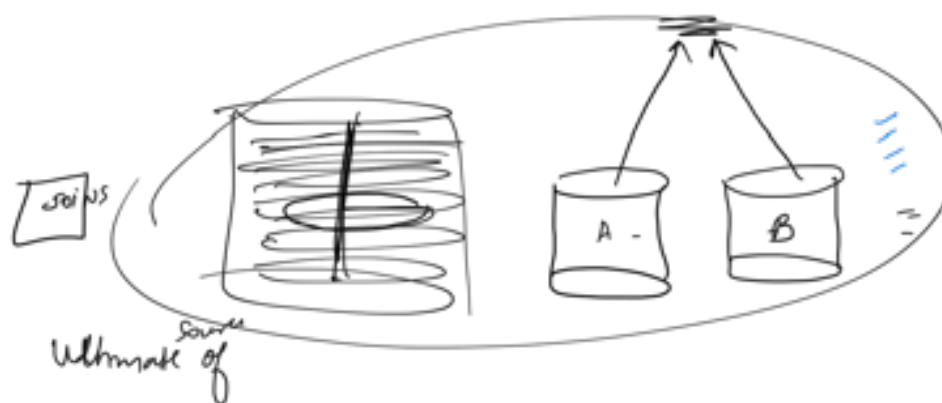
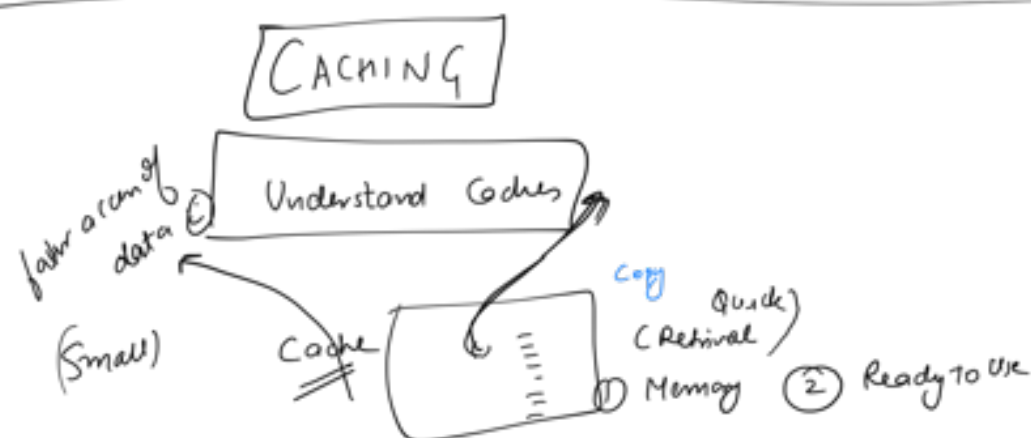
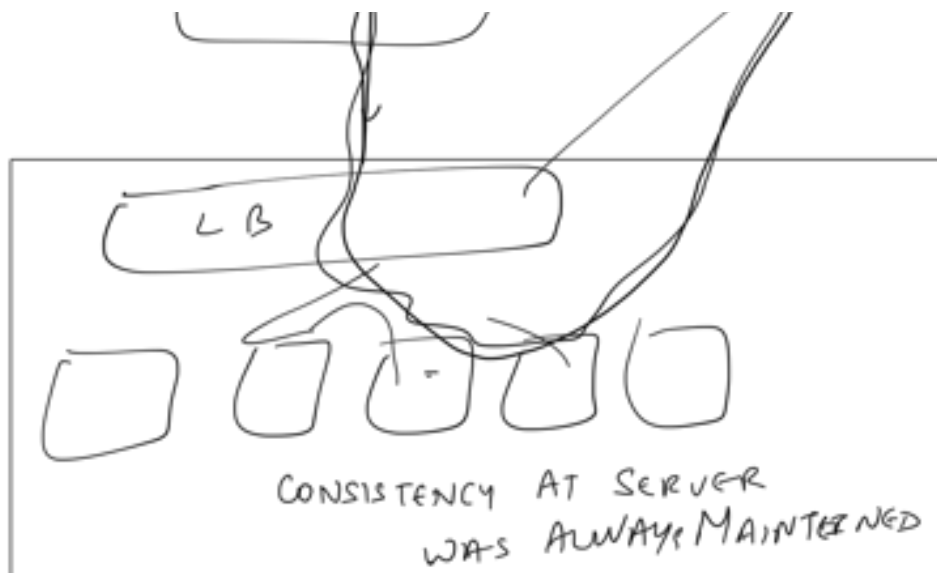


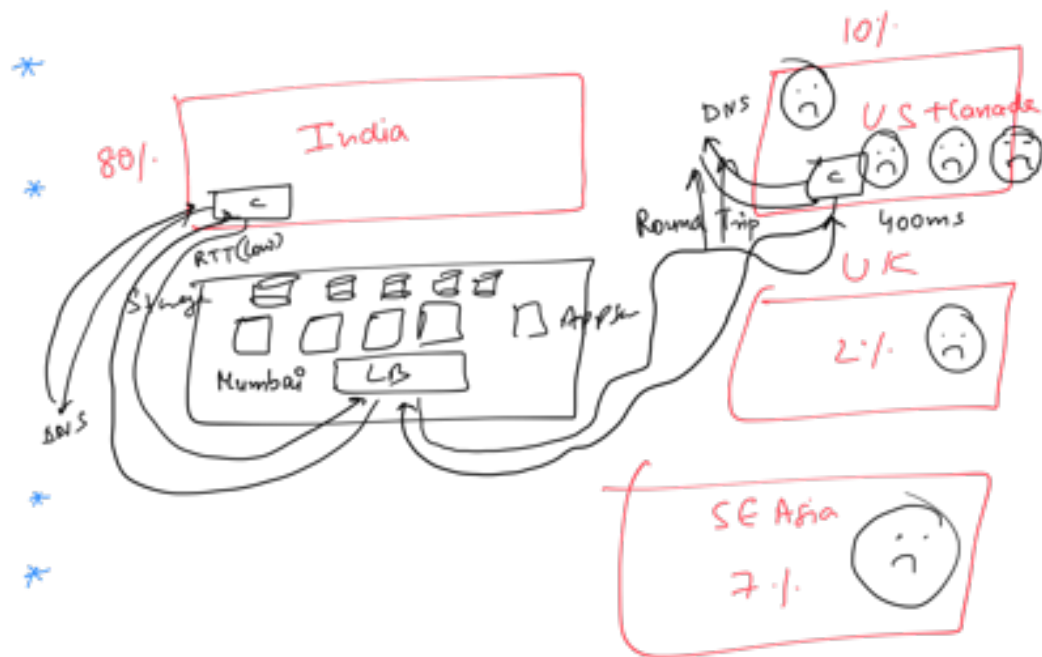
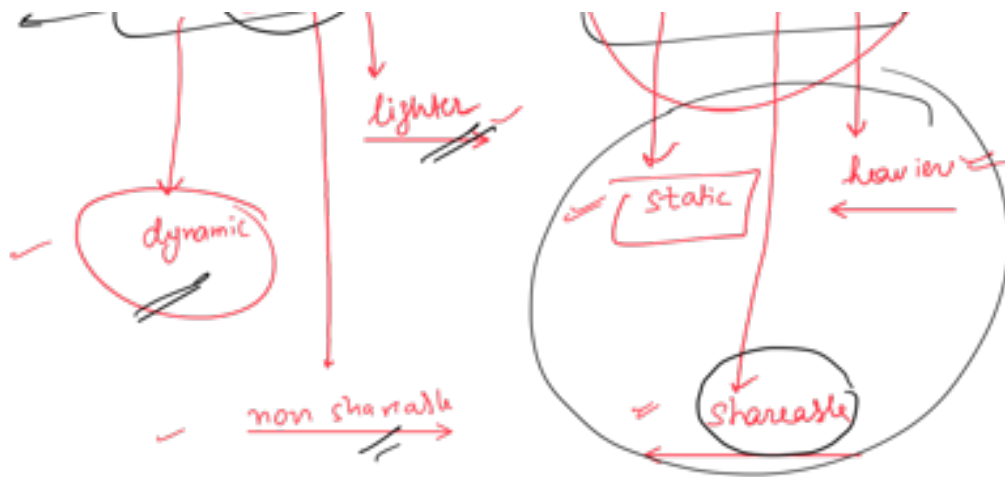
120mm.
60mm



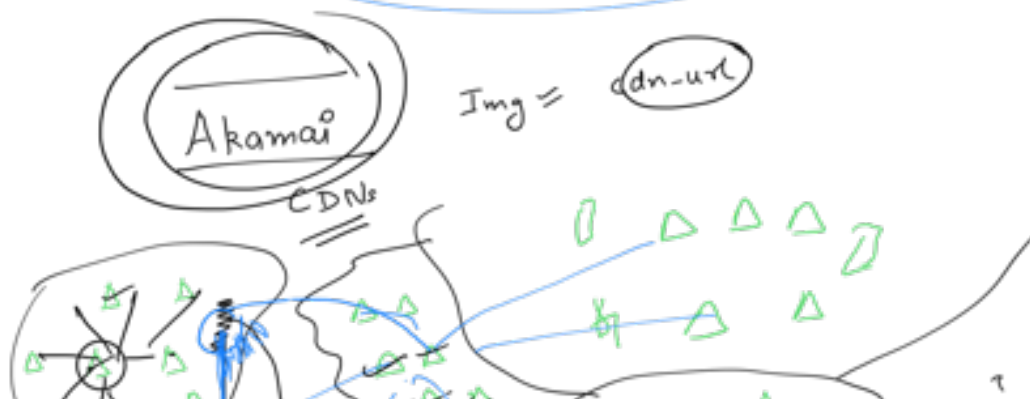
Mobile

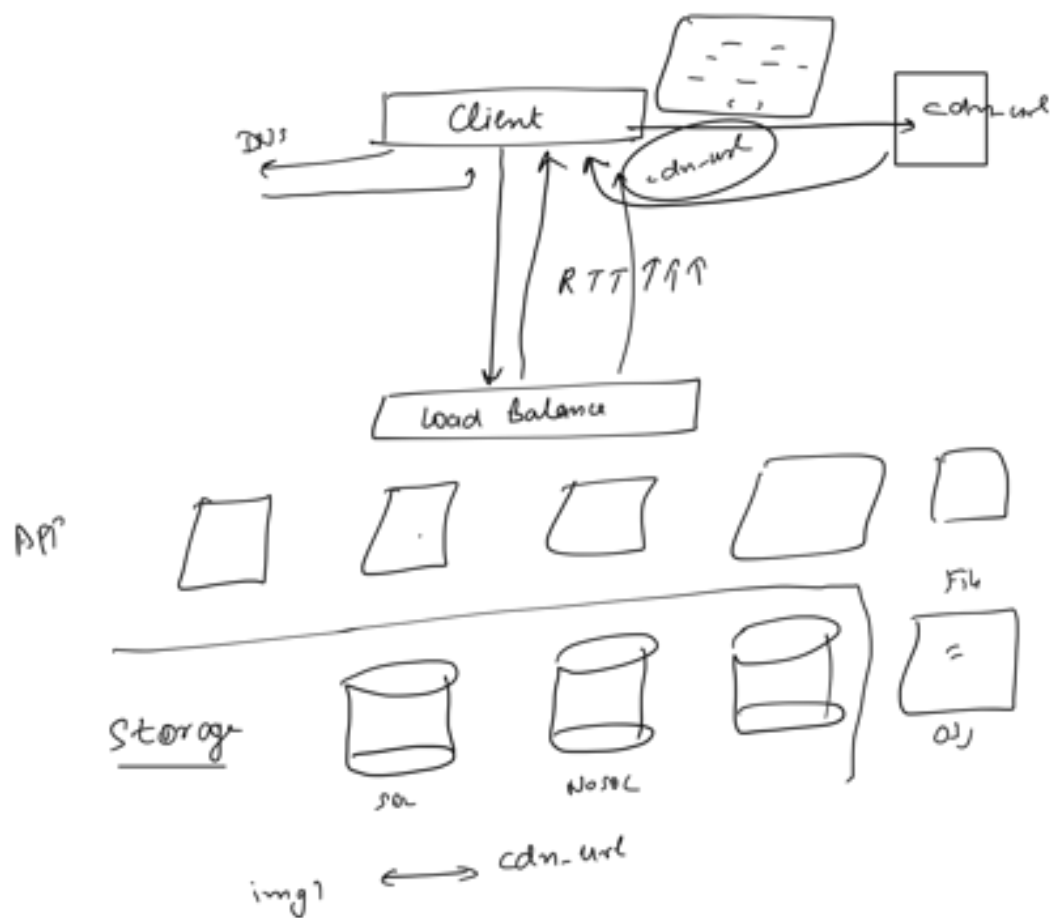
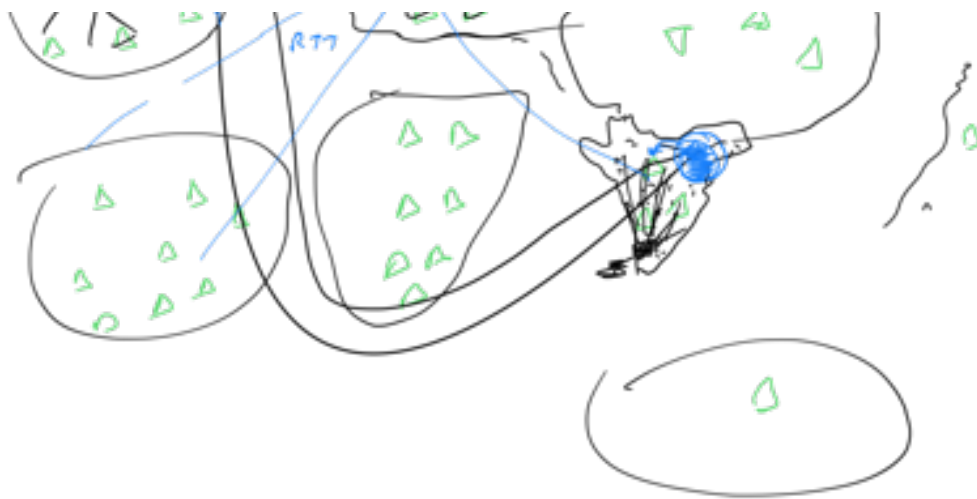






Content Delivery Network
CDN





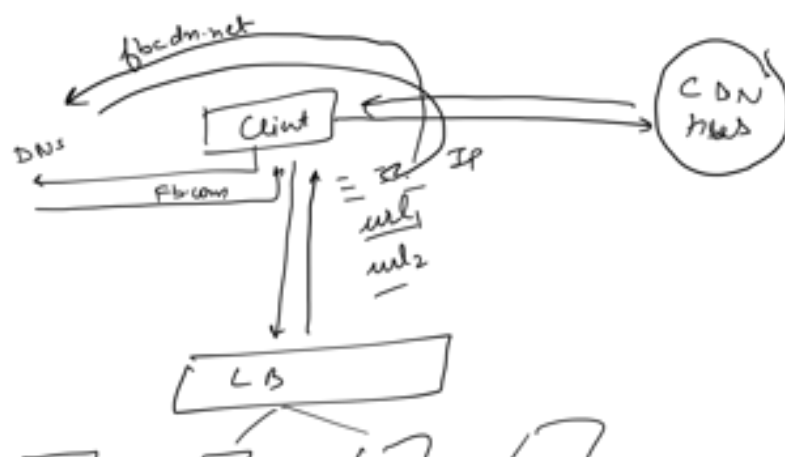
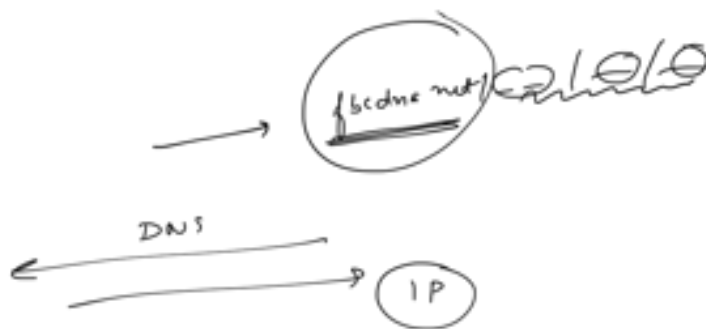
→ network is NOT getting checked
→ latency solver

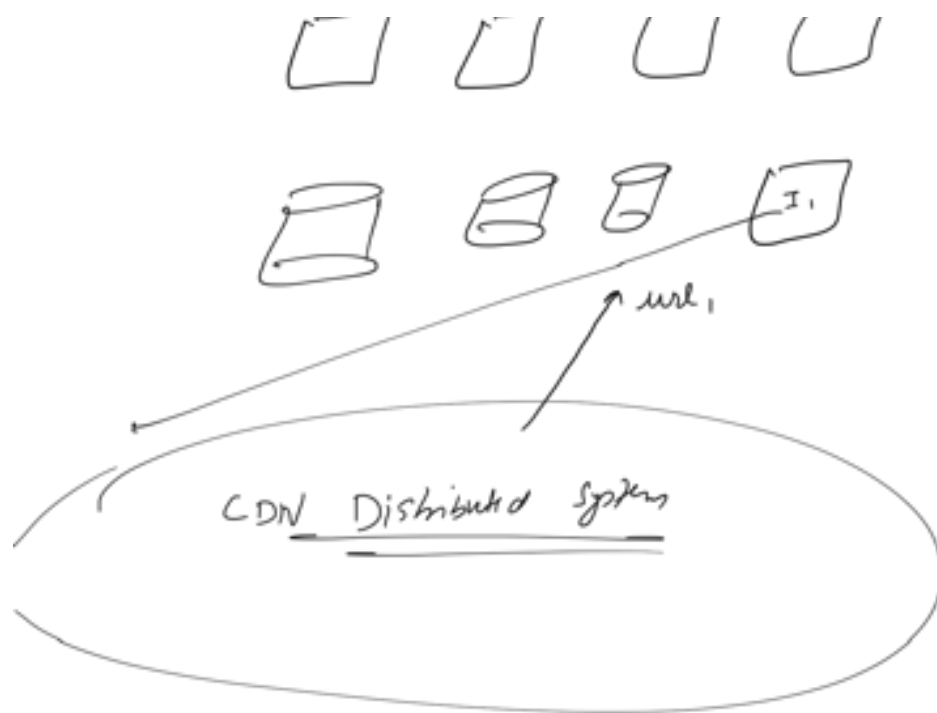
(Deals)



Company → CDN
 CDN hubs → CDN subscribers
 User → CDN

cdn





Cache Invalidation

① → Time to live (TTL)
 = url should automatically become void after this time

② → Start sending a new URL
 ③ → Versioning

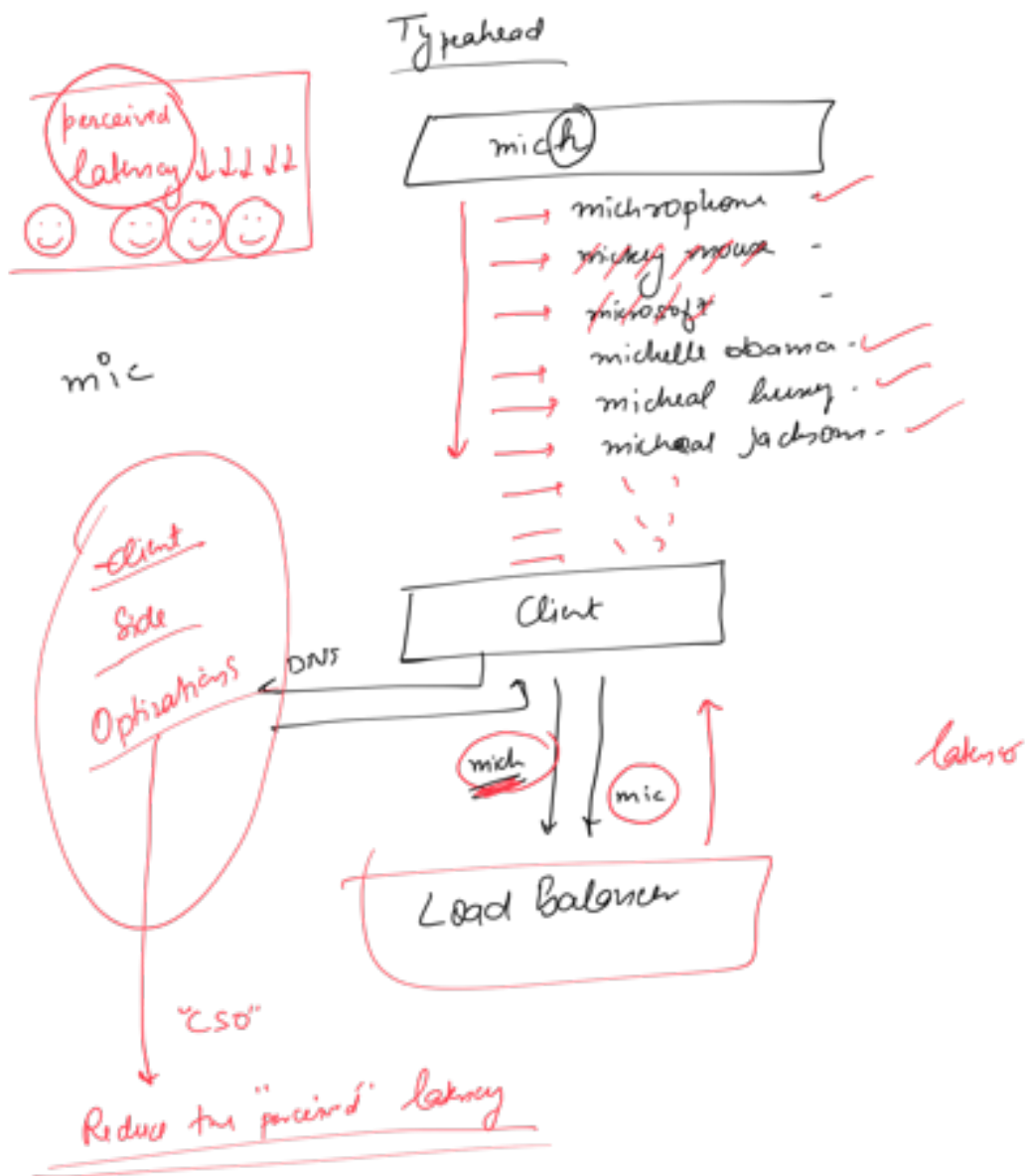
Privacy in CDNs

- ① CDN urls are by design not guessable
- ② encrypted format
- ③ Authentication

② Client Side Caching

You can choose to store some amount of "important" data on the client's side

Browser
Mobile App
Desktop App



Browser Optimization

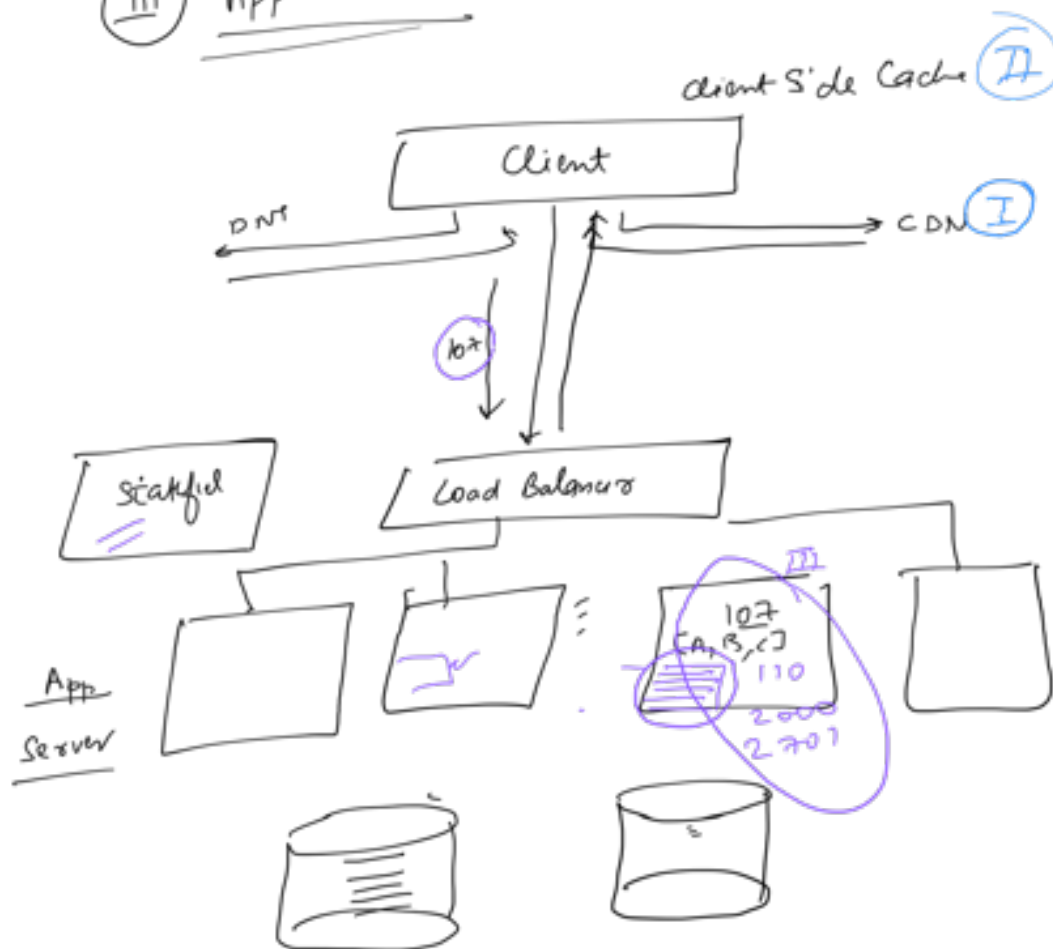
automatically identify a few
static resources for you and
cache them

Hard Reload / Forced Reloat

'ctrl + shift + R'



III App Server Cache

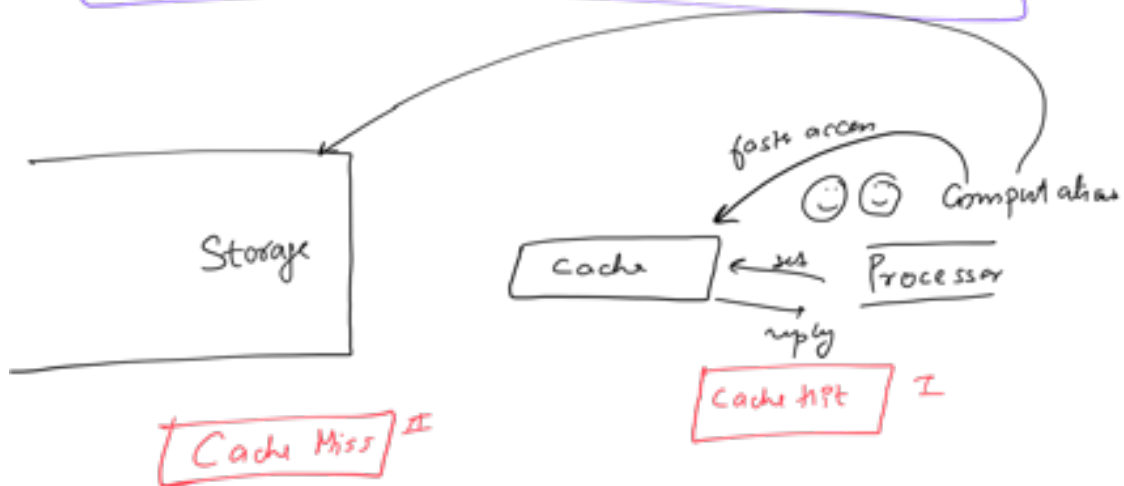


Cache

Replacement

→ Recently Used

- (A) LRU
- (B) MFU
- (C)



Access Time for Cache = X
 " " DB = $100X$

k calls

Without Cache

$$\text{Time} = k \times \text{Avg DB Access Time} + k \times \text{Avg Network Access Time}$$

80-20

With Cache

70% Hit Rate
 30% Cache Miss Rate

$$= [70 \times k] \times \text{Cache Access Time}$$

$$\text{Time} = \left\lfloor \frac{\dots}{100} \right\rfloor + \dots$$

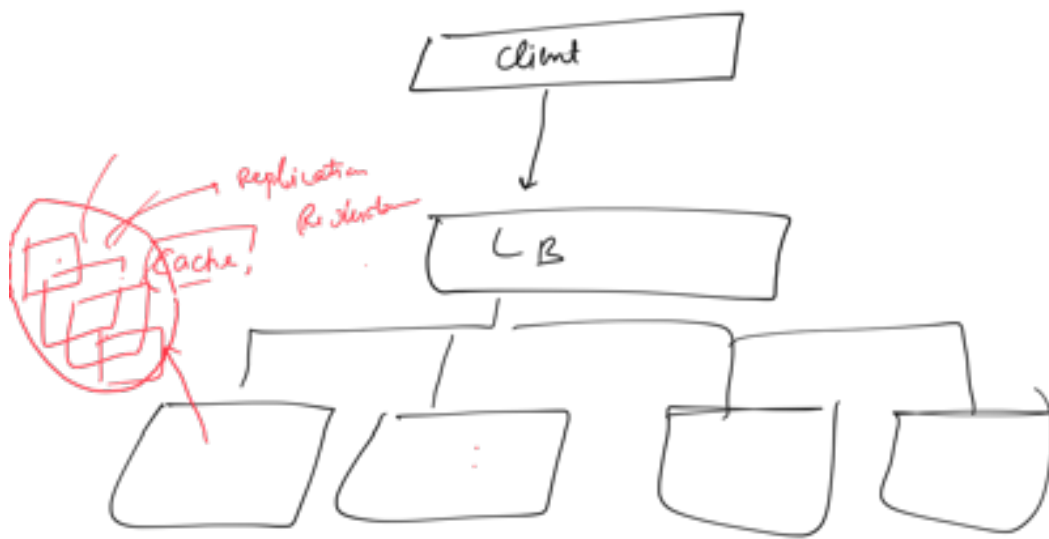
$$\left\lfloor \frac{30 \times k}{100} \right\rfloor \times \text{Cache Access Time} + 0.3k \times \text{DB Access Time} + 0.3k \times \text{Net A-Time}$$



- (A) How frequent are your incoming requests
- (B) Cache Eviction Policy
- (C) Amount of cache storage

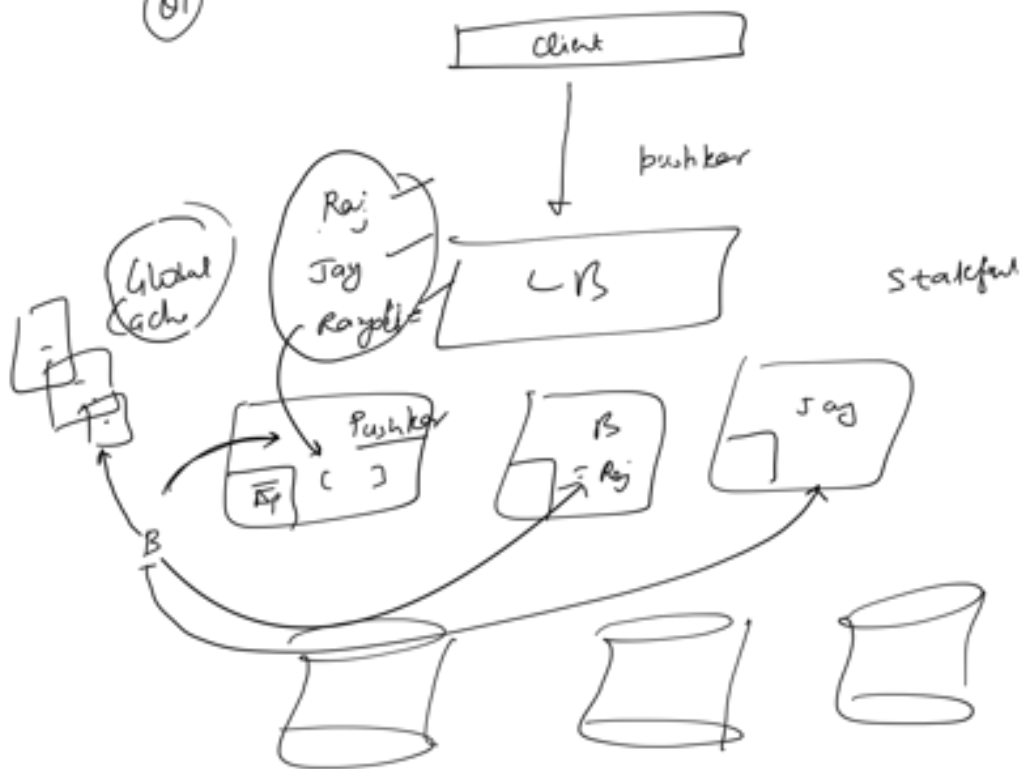
④

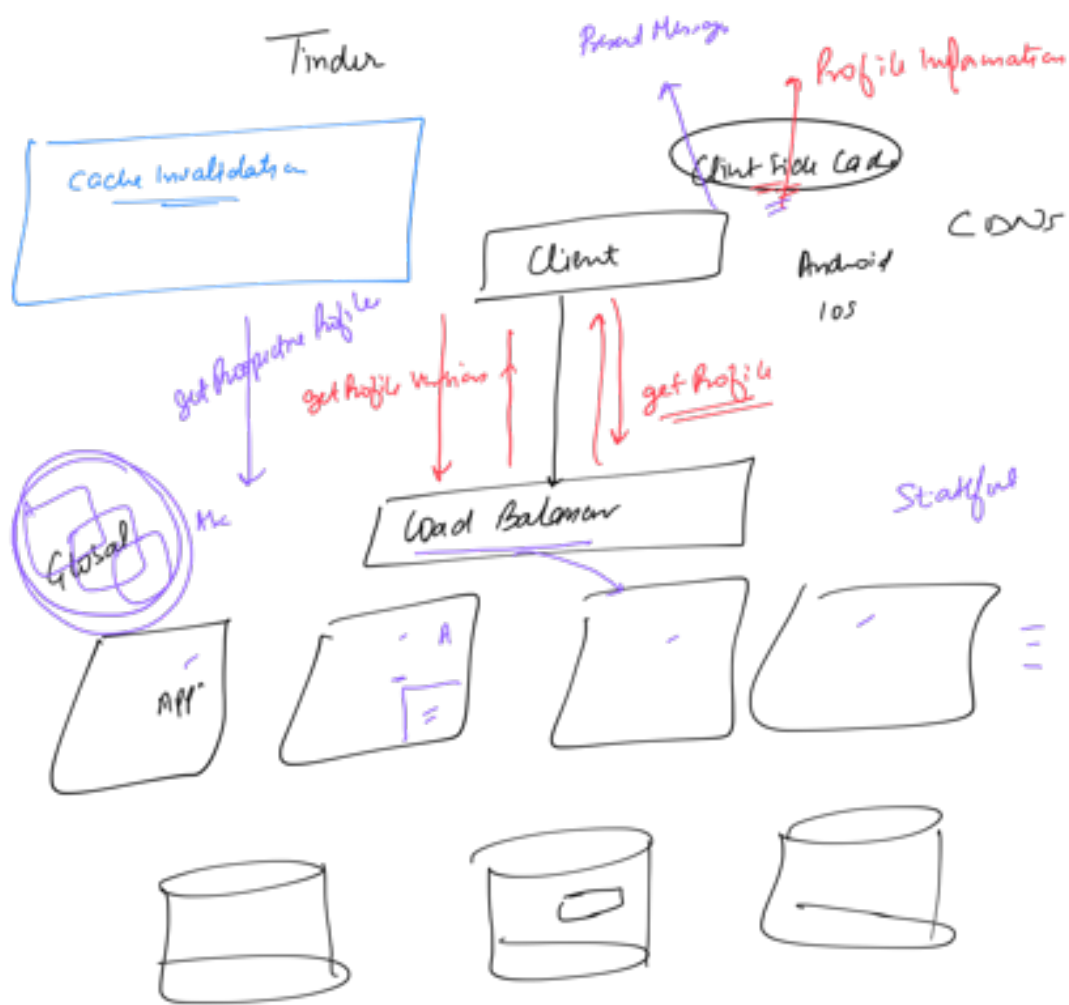
Global Cache





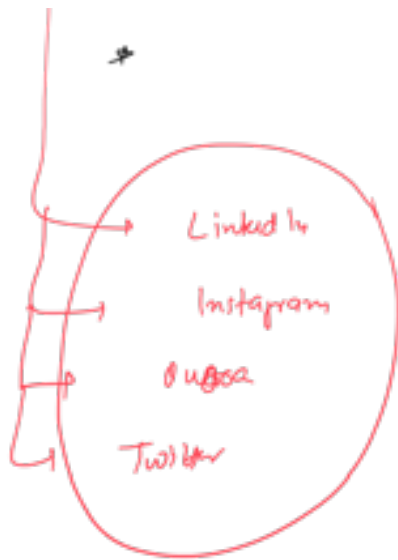
(01)





1. Client Side Cache = profile info





~~(I)~~ Ability to view posts of interest

~~(II)~~ Like / Comment / share a post

~~(III)~~ Multimedia Support → share Image / videos

(IV) Bookmark a post

~~(V)~~ Making a friend / follow → ancillary feature

(VI) Suggesting friends

(VII) Report a post

(VIII) Support hashtags

~~(IX)~~

Relevance

Time Decay

Connections Strength

(X)

Ads



~~Automatic refresh~~
~~Write a post~~
 Tagging yes

B Estimation of scale

① # users = 2 B →
 ② DAU = 400 M → new posts
 ③ # people who produce content or make a post = 40 M

④ # new posts / day = $2.5 \times 40 M$
 = 100 M

6 MPUT & 6 AD

① writes / day = 100 M

② Reads / day = $\square \rightarrow \times \text{mul fact}$
 = $400 M \times 25$

10,000 M → 1,15,000
 30 M →

= 10,000 M
 = 10 B reads / day

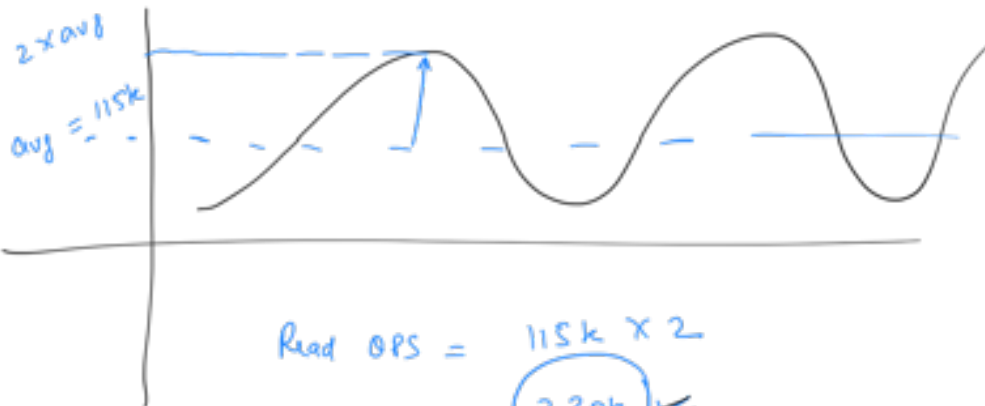
Read OPS = $\frac{10 B \times 10^9 \times 10^6 \times 7}{24 \times 60 \times 60}$

$$= 115000 \text{ OPS}$$

$$= \underline{\underline{115k \text{ OPS}}}$$



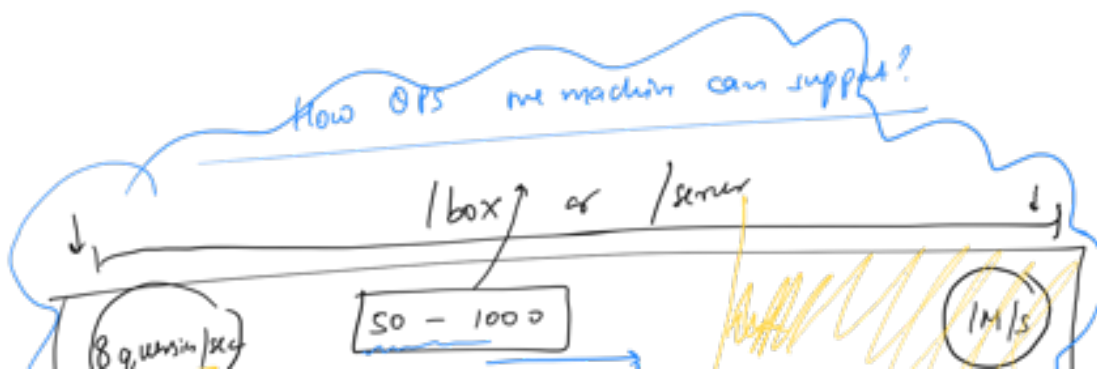
$$\frac{\text{Write OPS}}{100M} = \frac{1000 \text{ write/sec}}{86400}$$

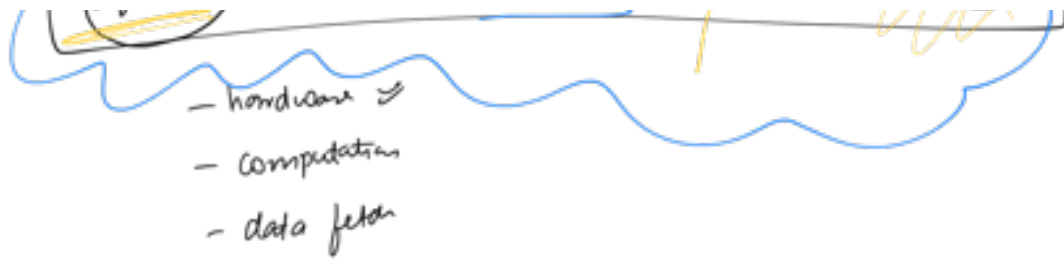


$$\text{Read OPS} = 115k \times 2$$

$$= \underline{\underline{230k}}$$

$$\text{Write OPS} = 2000$$



- 
- hardware
 - computation
 - data fetch

$$\rightarrow \boxed{100 \text{ ops/box}}$$

1000

$$\boxed{\frac{230k}{100} = \# \text{ boxes}}$$

2300 boxes

Load Factor \Rightarrow 65% - 75%

$$\frac{2300}{0.7} = \sim 4000$$

EC2 \equiv App Server