Agenda.
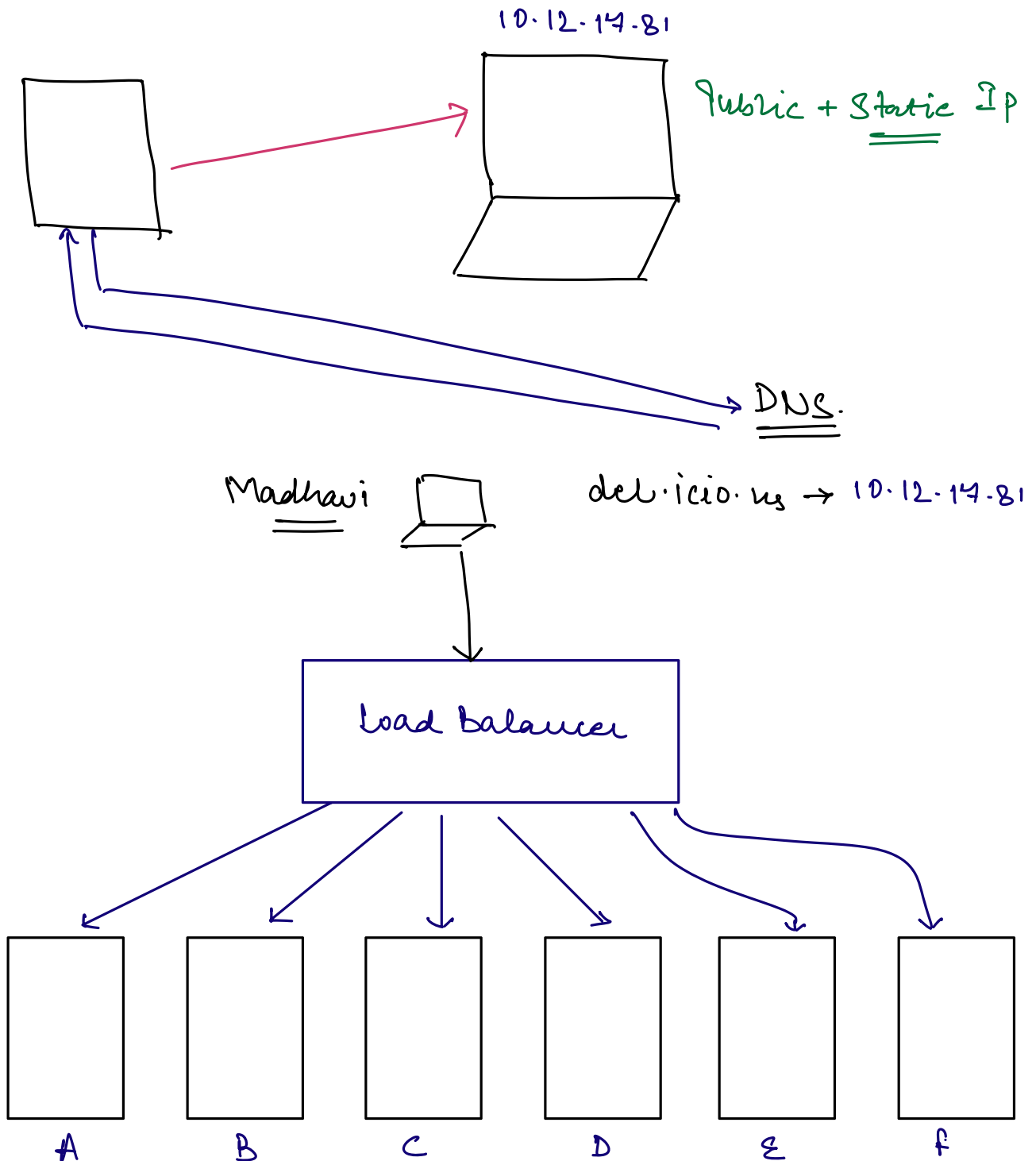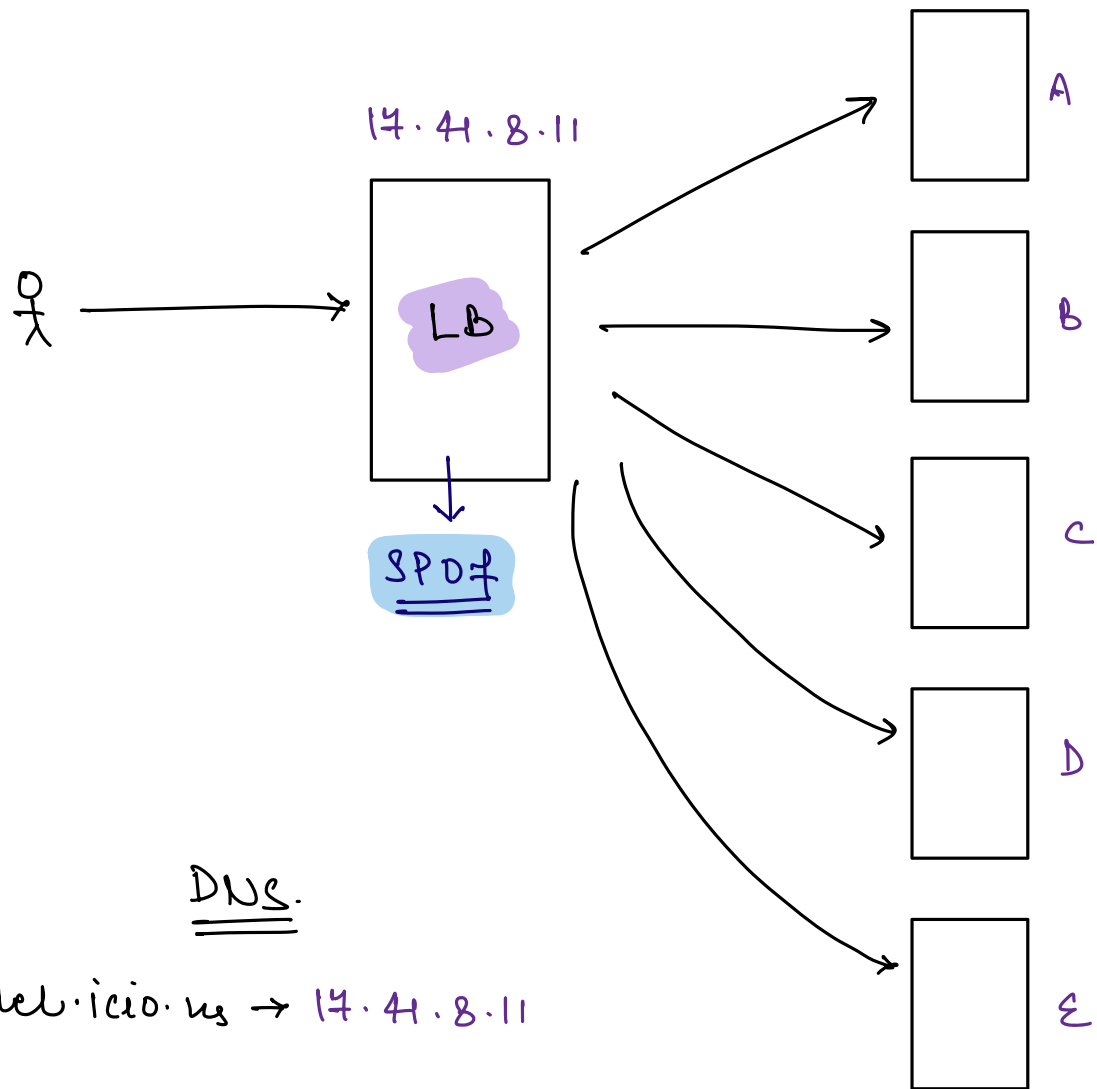
→ Load Balancing

→ Stateful (vs) Stateless Load Balancing

→ Algos for Load Balancing

   ↳ Consistent Hashing.
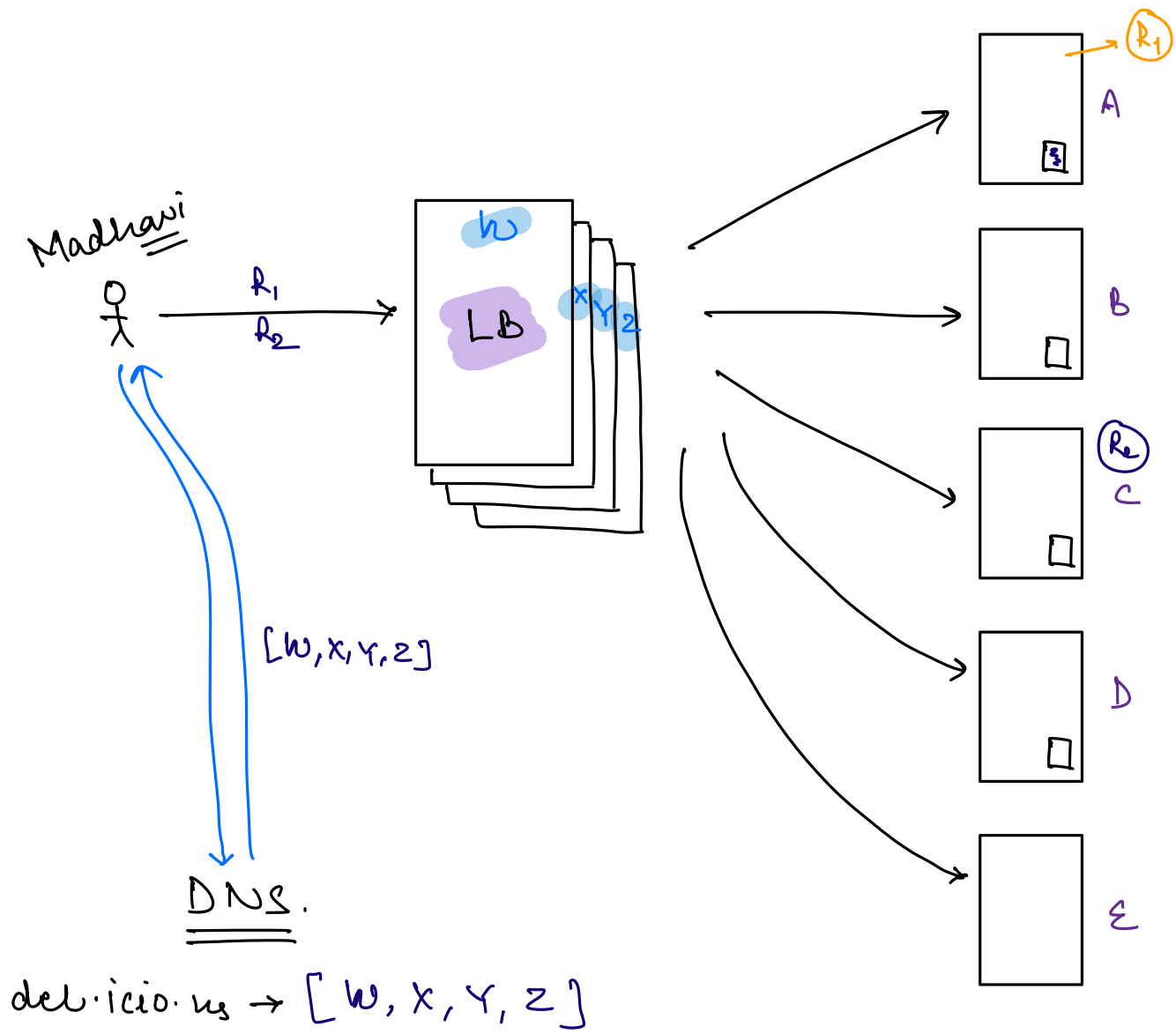
10.12.14.81

Public + Static IP

DNS.

del·icio·us → 10.12.14.81

Madhavi

Load Balancer

A   B   C   D   E   f

17.41.8.11

LB

SPOf

A

B

C

D

E

DNS.

del·icio·us → 17.41.8.11

Load balancer : Distribute the traffic among all the servers.

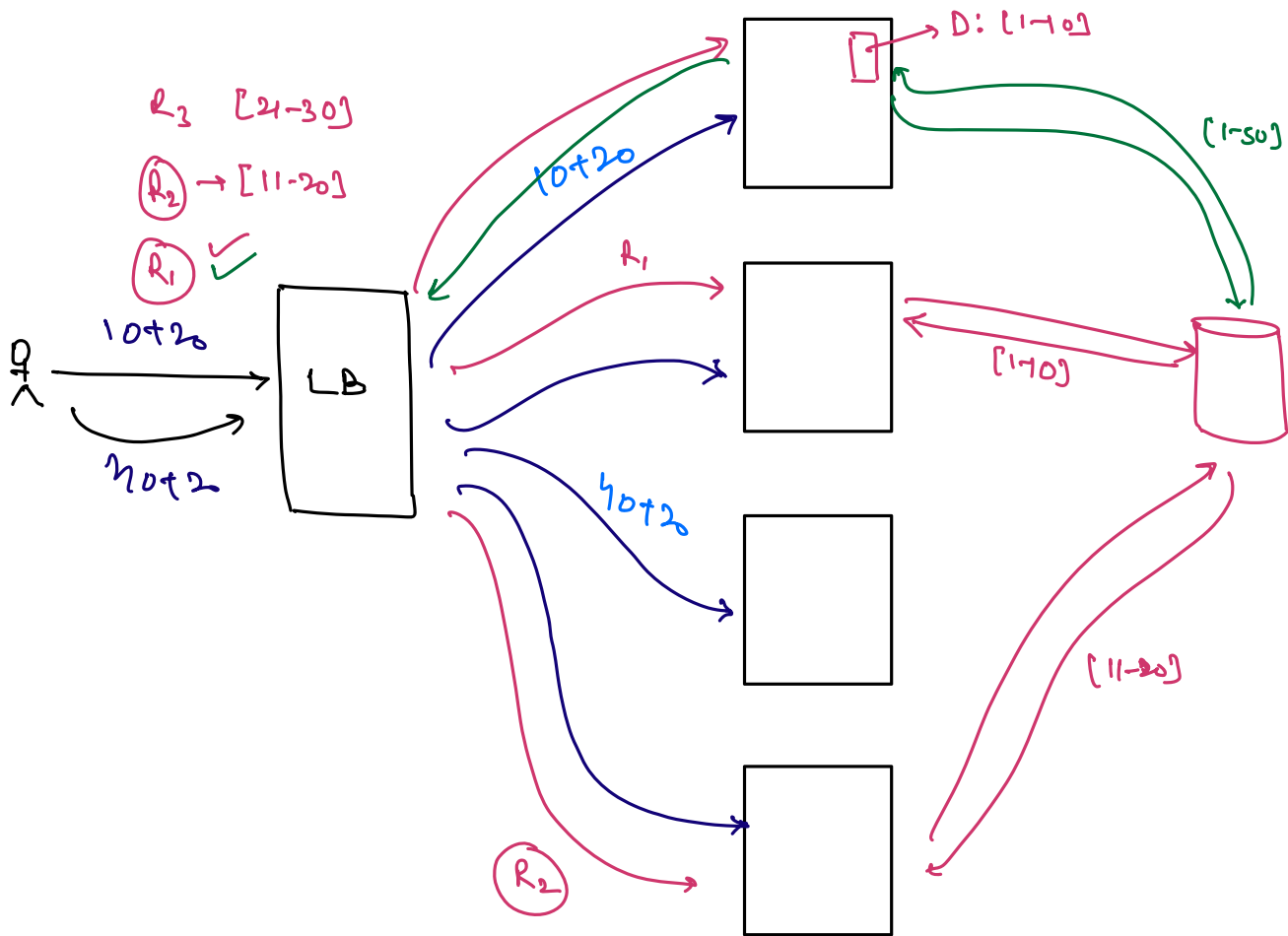⇒ Instead of having a single LB, we can have multiple LB.

Madhavi

R_1
R_2

LB

w

x Y z

[W, X, Y, Z]

DNS.

del·icio·us → [W, X, Y, Z]

GeoDNS.

R_1

A

B

R_e

C

D

E

8.8.8.8
↓
Google

Stateless (vs) Stateful LB.

1) **Calculator App**^{n}

$$Cal(X+Y) \rightarrow 2$$

Stateless
↓
When requests are completely independent
of each other.

2) Chat4pt

RAM

u1: ▭
→ ▭
u1: ▭
→ ▤

LB

R1

Q

R2

R1|R2
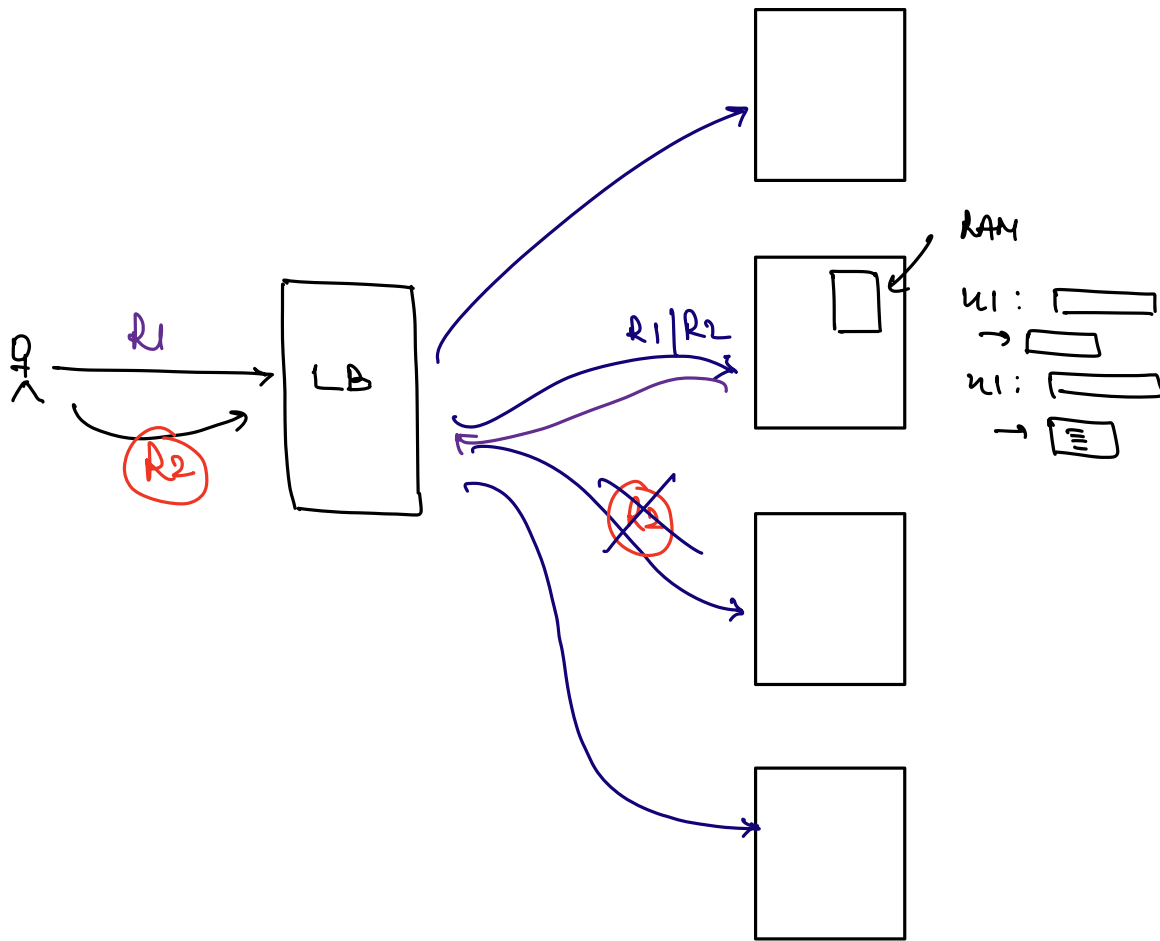
Ⓡ1

Q.1 Who was the captain of Indian Cricket team during 2011 WC ?

→ MS Dhoni.

R2 Tell me more about him.
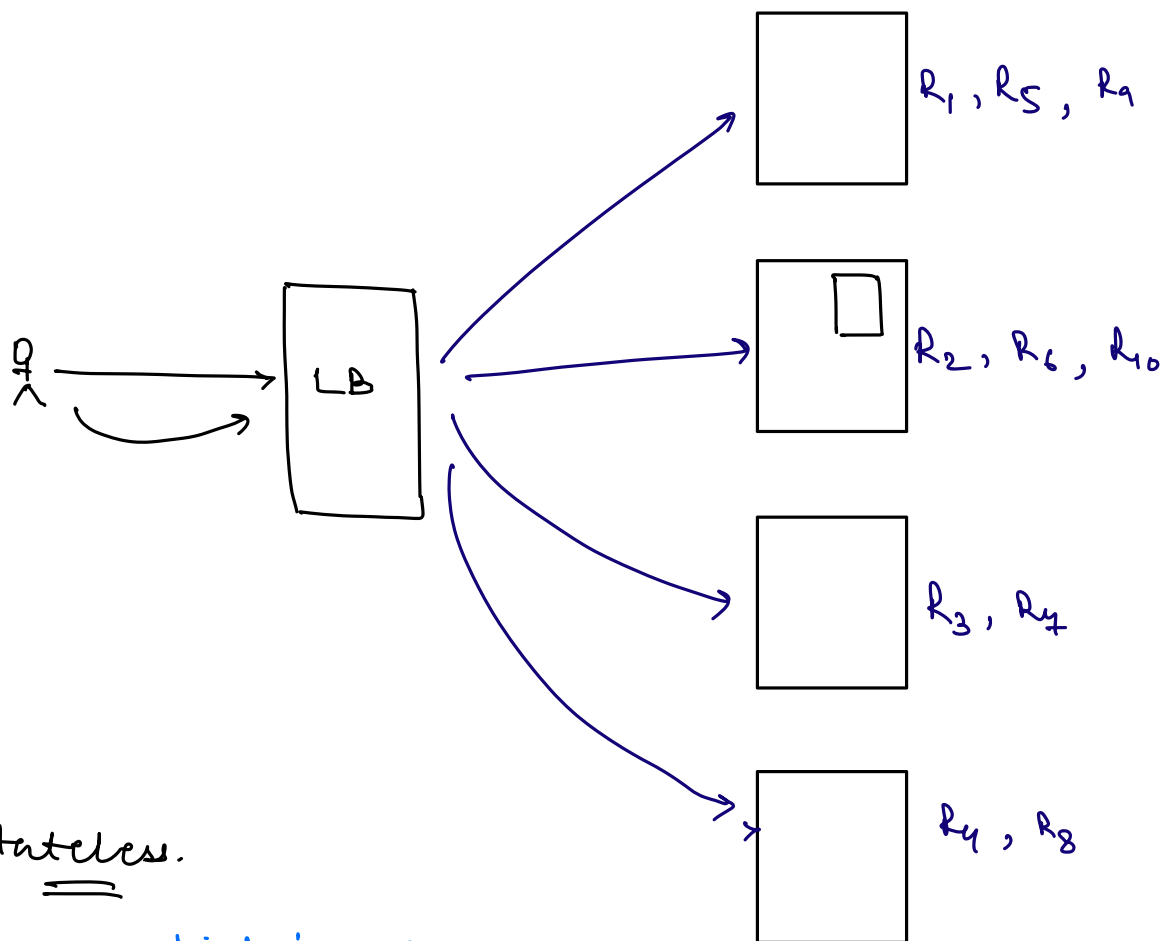
Stateful.

└─> When the current request is dependent on the state of the previous request.

1. Random LB.

LB : [A, B, C, D, - - - - -]

2. Round Robin

→ Stateless.

→ Equal distribution of load.

→ Easy to maintain.

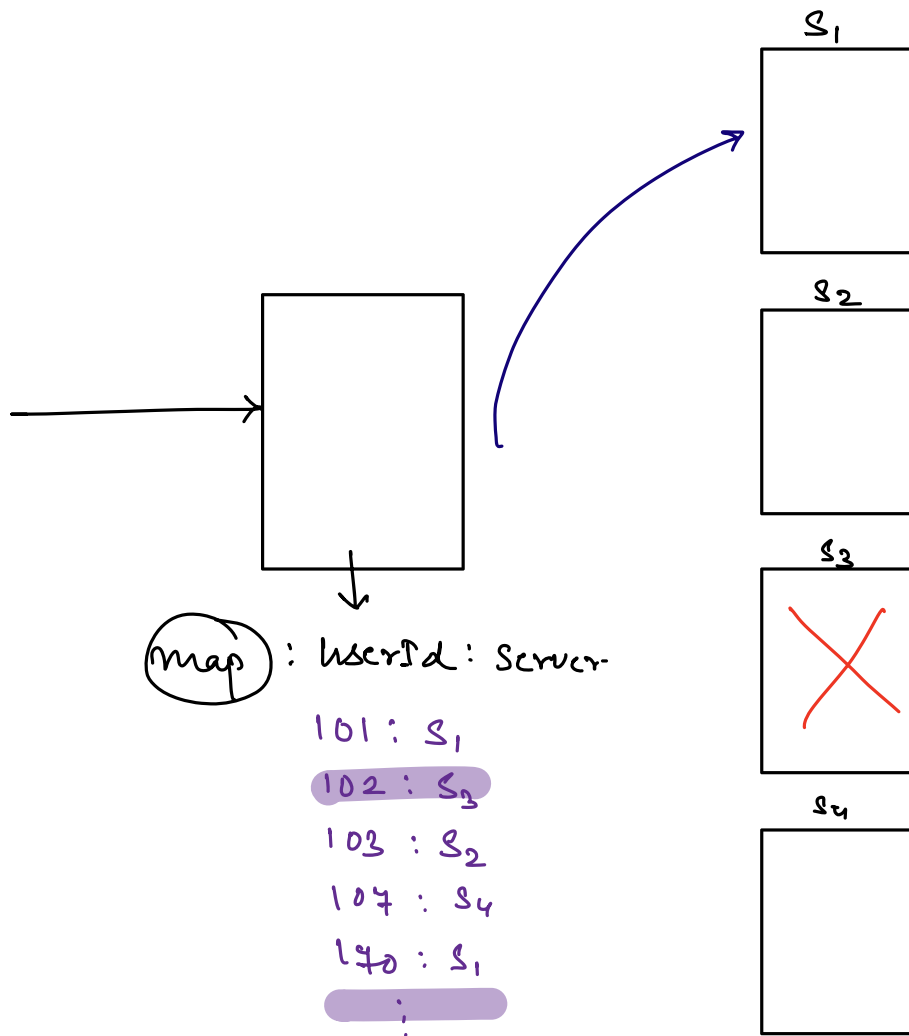⟹ Properties of a good LB Algorithm.

→ Equal load distribution

→ fast

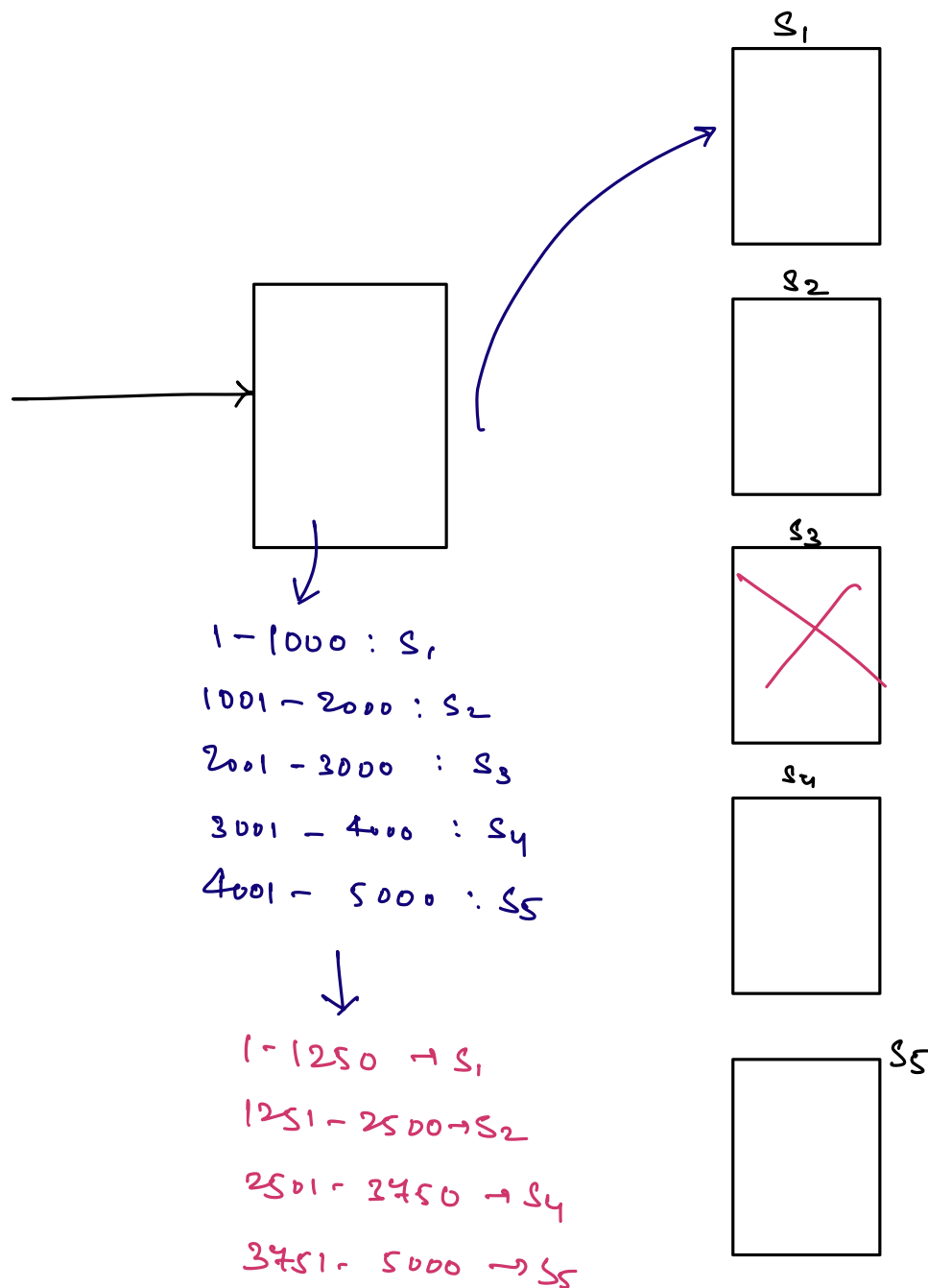→ Easy to maintain at LB.

→ Lesser movement incase a m/c goes down.

3) Least Connection first. → Stateless.

4) less response time. → Stateless.

Stateful LB algo.

map : UserId : Server
101 : S_1
102 : S_3
103 : S_2
107 : S_4
170 : S_1
:

Range based Sharding

S1

S2

S3

S4

S5

1 - 1000 : S1

1001 - 2000 : S2

2001 - 3000 : S3

3001 - 4000 : S4

4001 - 5000 : S5

1 - 1250 → S1

1251 - 2500 → S2

2501 - 3750 → S4

3751 - 5000 → S5

⇒ Lot of chaos is happening if a m/c goes down.

Modulo based Sharding.

N = 4

101

S1

S2

S3

S4

$101 \% 4 = 1$

$107 \% 4 = 3$

$200 \% 4 = 0$
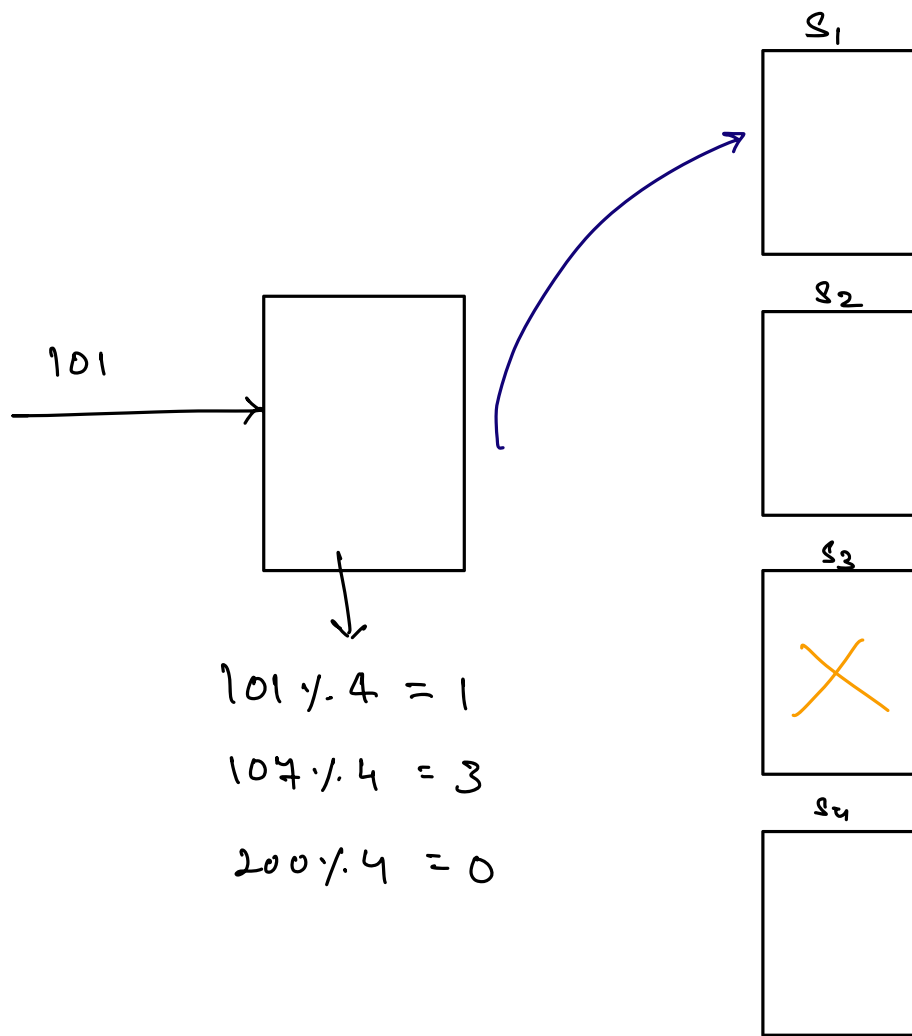
$X \% 4 = 0, 1, 2, 3$

$101 \% 3 = 2$

$102 \% 3 = 0$

$103 \% 3 = 1$

$200 \% 3 = \text{——}$

⇒ Lot of chaos is happening if a m/c goes down.

Consistent Hashing

$\Rightarrow$

H_S

$\downarrow$

Servers

S_1 S_2 S_3 S_4

H_u

$\downarrow$

users.

H_u
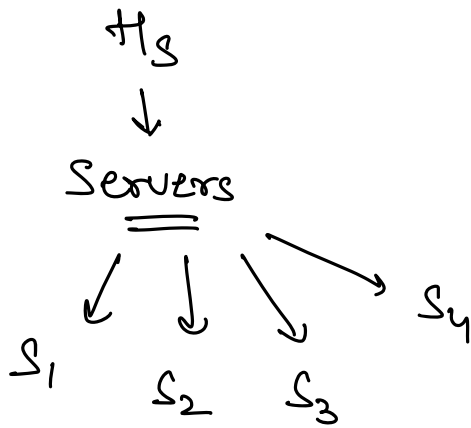
$\quad \hookrightarrow [0, 10^{18}]$

$$H_S(S_1) = S_1'$$

$$H_S(S_2) = S_2'$$

$$H_S(S_3) = S_3'$$

$$H_S(S_4) = S_4'$$

$\downarrow$

$[0, 10^{18}]$

$\Rightarrow$ Stateful.

$$H_u(u_1) = u'_1$$
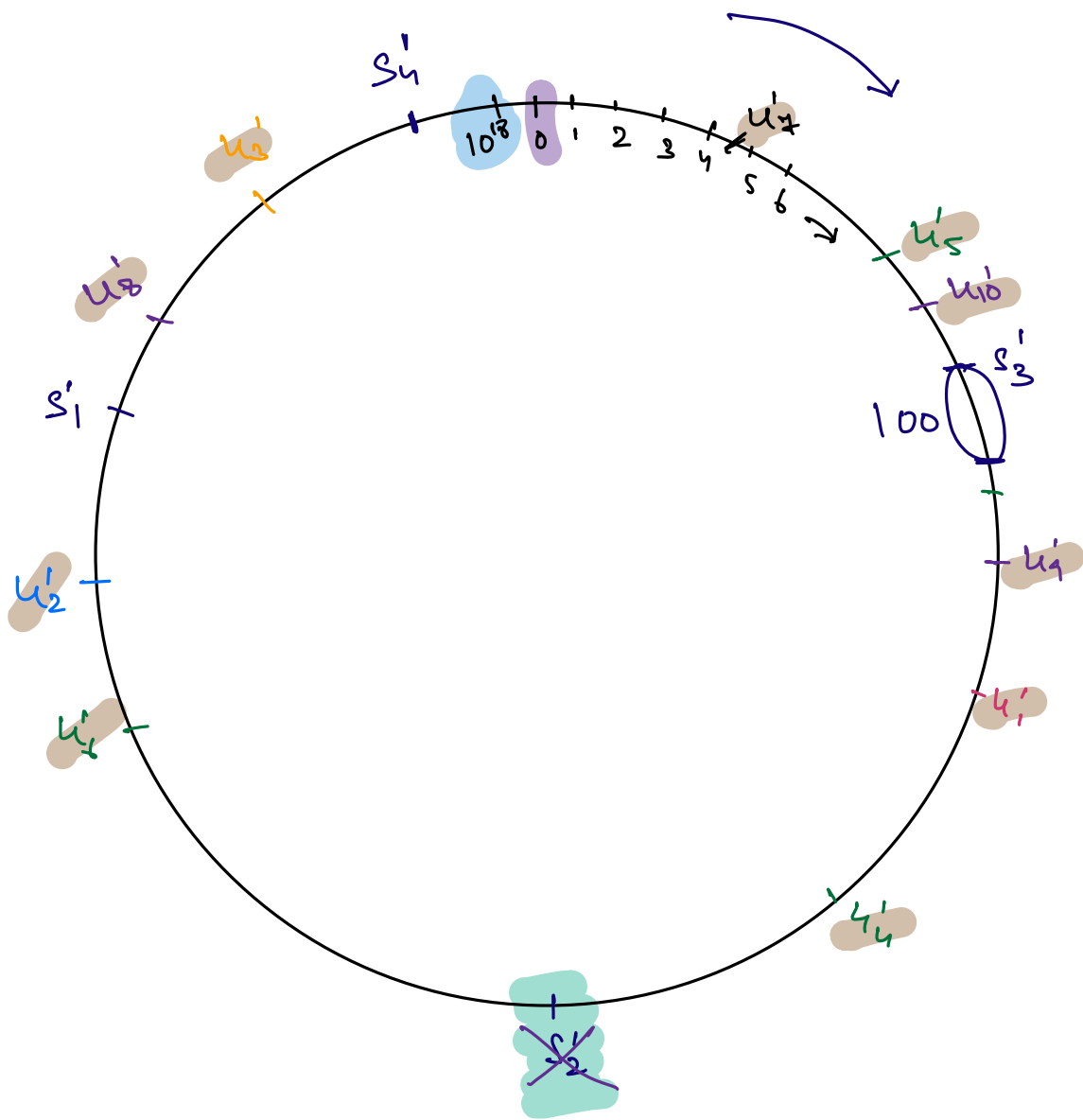
$$S_3 \rightarrow u_5 \quad u_7 \quad u_{10}$$

$$S_2 \rightarrow u_9 \quad u_1 \quad u_4$$

$$S_1 \rightarrow u_6 \quad u_2$$

$$S_4 \rightarrow u_3 \quad u_8$$

$\Rightarrow$

$$H_S(S_1) = S_1' = 1000$$
$$= S_2' = 10^6$$
$$= S_3' = 10^8$$
$$= S_4' = 10^{10}$$

Sorted Array.

1000	10^6	10^8	10^{10}

$\longrightarrow$ (u_1)

$$H_u(u_1) = (x) = 5 \times 10^6$$

from the Array, just find the next greater element than

$\underline{\underline{x.}}$

$\rightarrow$ Binary Search

If a server goes down then ==send the traffic of that server== to the ==next server== in the ring.

⇒ Lesser Chaos.

If S_2 goes down, then S_1 gets $2x$ of traffic.

⇒ S_1 may go down.

If S_1 goes down, then S_4 gets $3x$ of traffic.

⇒ S_4 may go down.

⇒ Cascading failure.
↓
SOLⁿ: Distribute of traffic of server that goes down to all the servers.