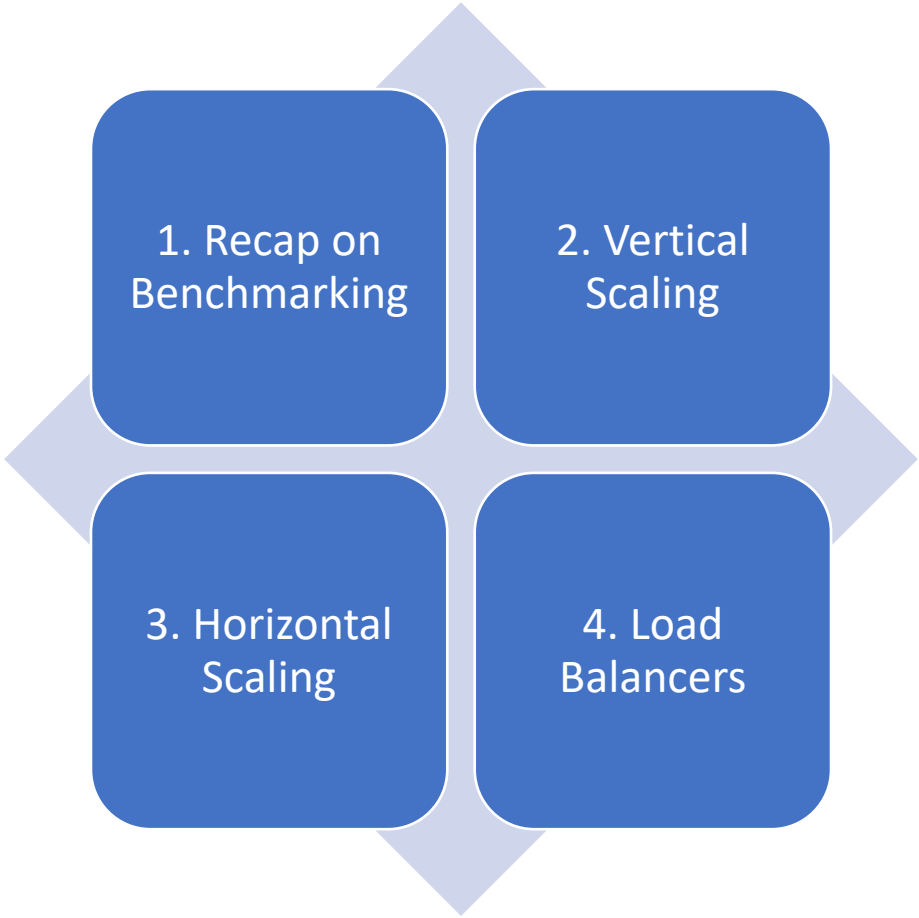
System Architecture Part 2

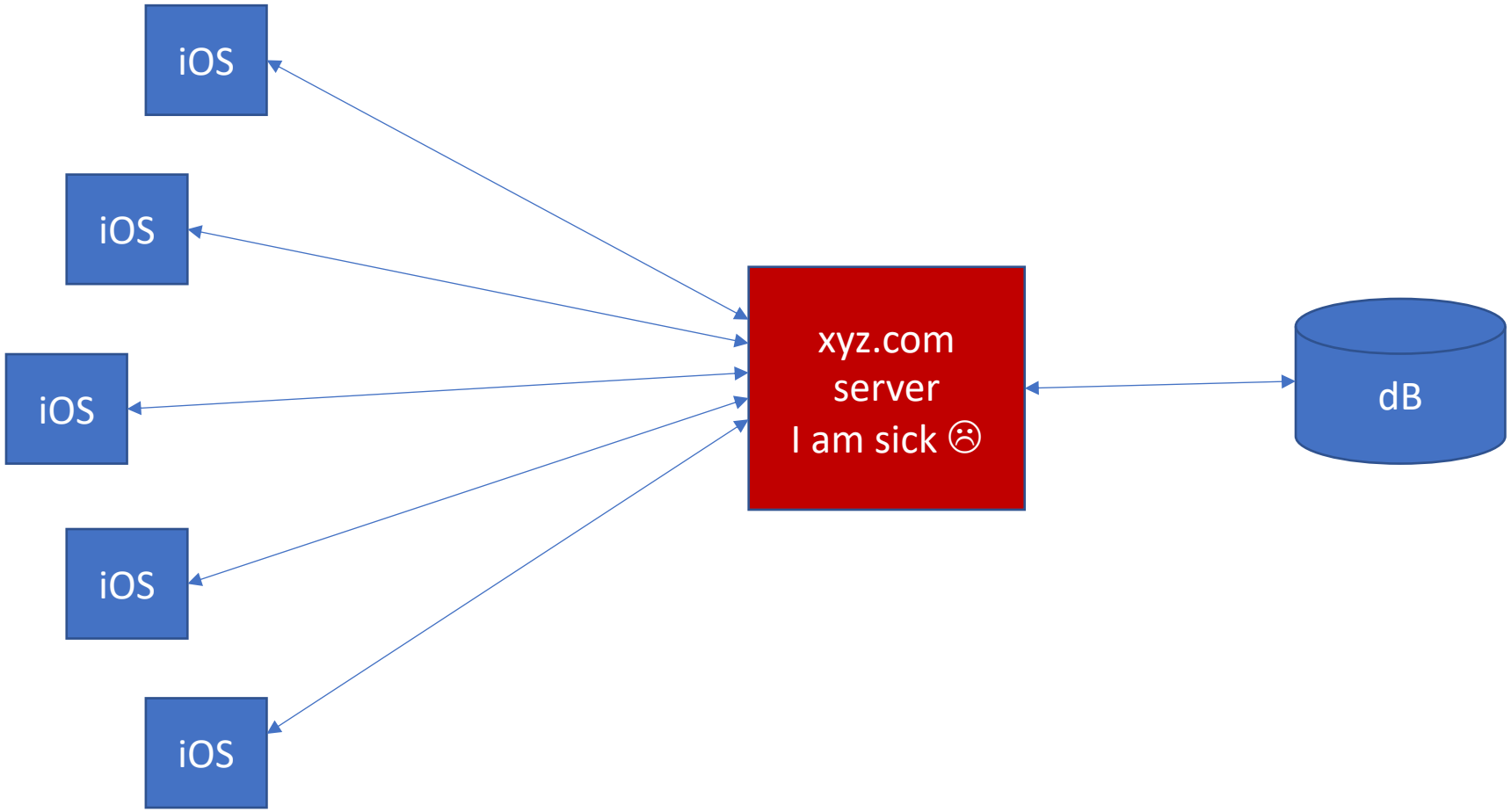
Lets Scale that system!!!

What We will be learning ?



1. Benchmarking

1. Benchmarking(Recap)



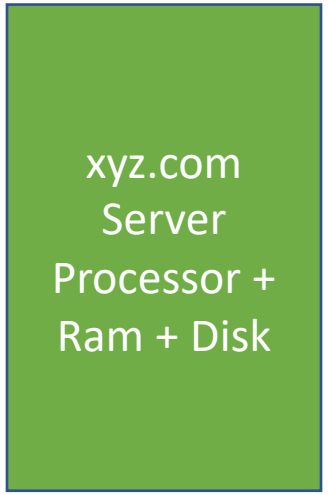
1. Benchmarking Report

1. Benchmarking Report: **requests/sec = 100(-c), latency in response Time = 10 ms**(Round Trip Time)
2. What is the latency of 101th, 201th response coming from the server ?
3. **Latency for 101th request** = 10 ms(for first 100 requests) + 10 ms = 20 ms
4. **Latency for 201th request** = 10 ms(for first 100 requests) + 10ms(for next 100 requests) + 10 ms = 30 ms
5. Eventually the **server will get slower** as the number of requests increase

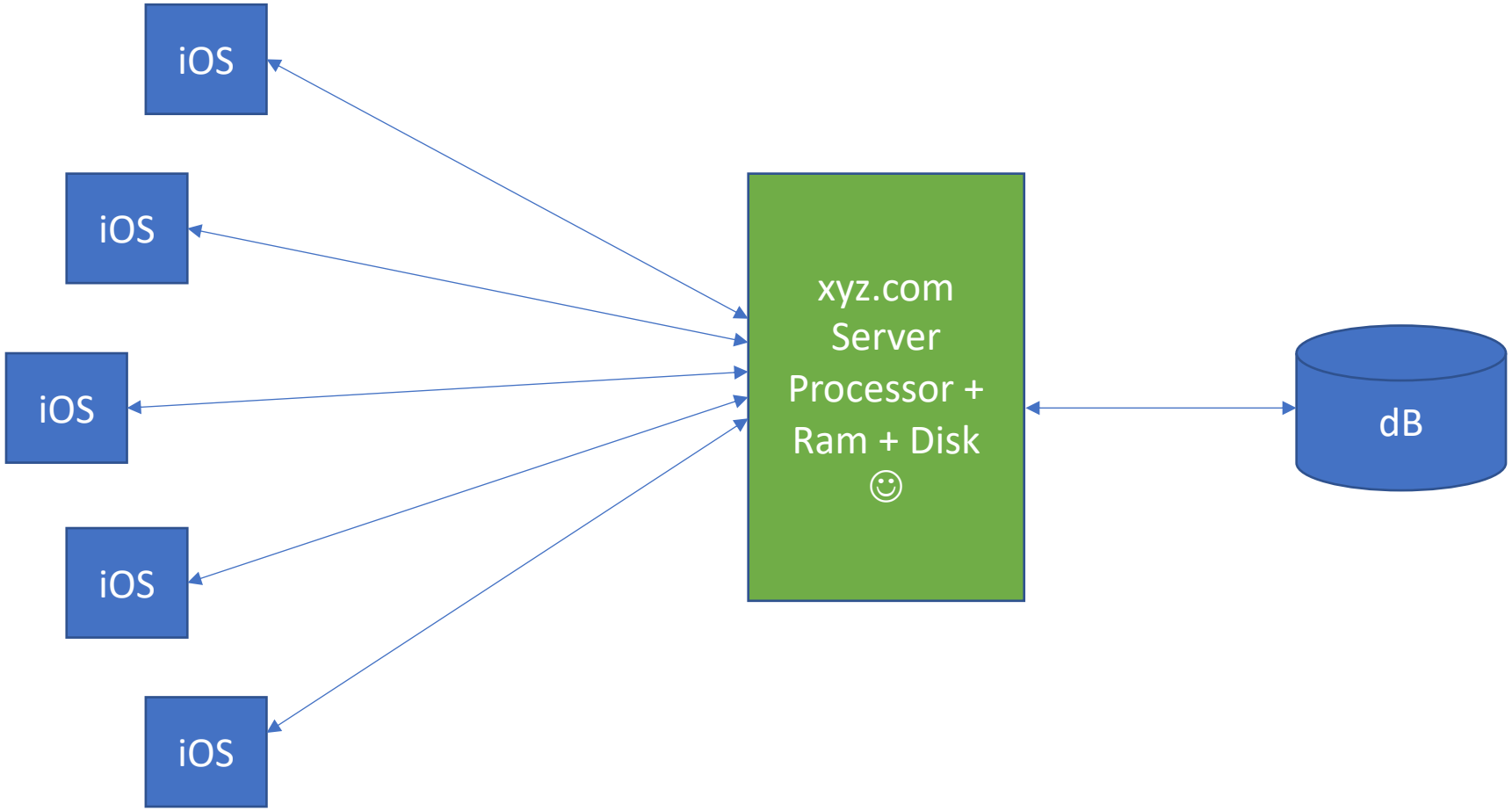
2. Vertical Scaling

2. Vertical Scaling

1. We increase the computation power(processer), memory and disk to scale the server. We **throw money to the problem** and buy a **bigger server machine**. This is called vertical scaling.
2. BM Report: 1000 req/sec, latency = 5 ms



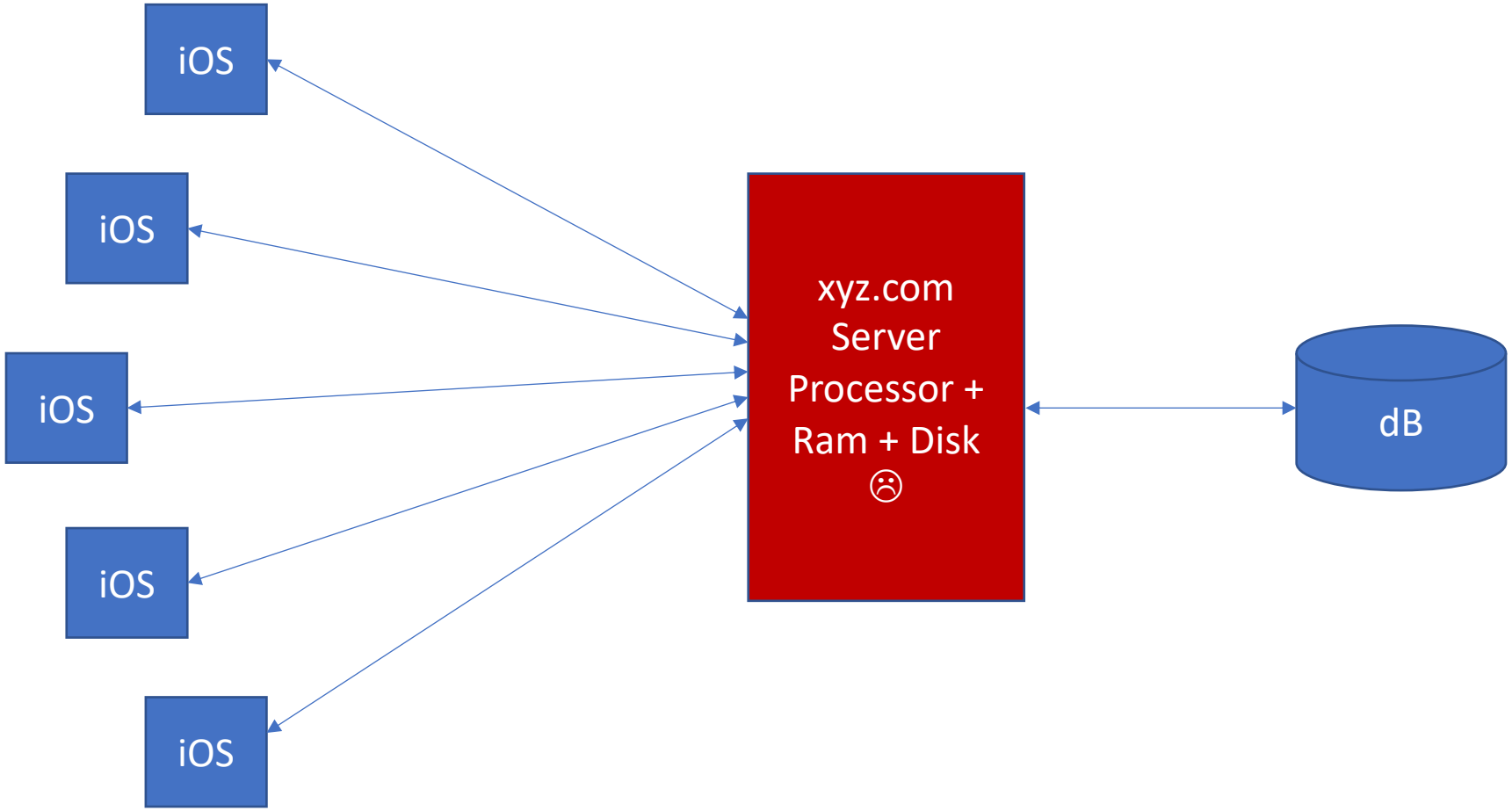
2. Vertical Scaling



2. Concerns with Vertical Scaling

1. What happens if the number of **requests/sec increases further** ?
2. There is a **ceiling** to it.!!
3. Eventually we are going to **exhaust our financial resources** or the **state of the art technology** we use.
4. This has a **single point of failure**.
5. What do we do when the number of **requests increases to 10K ?**

2. Concerns with Vertical Scaling

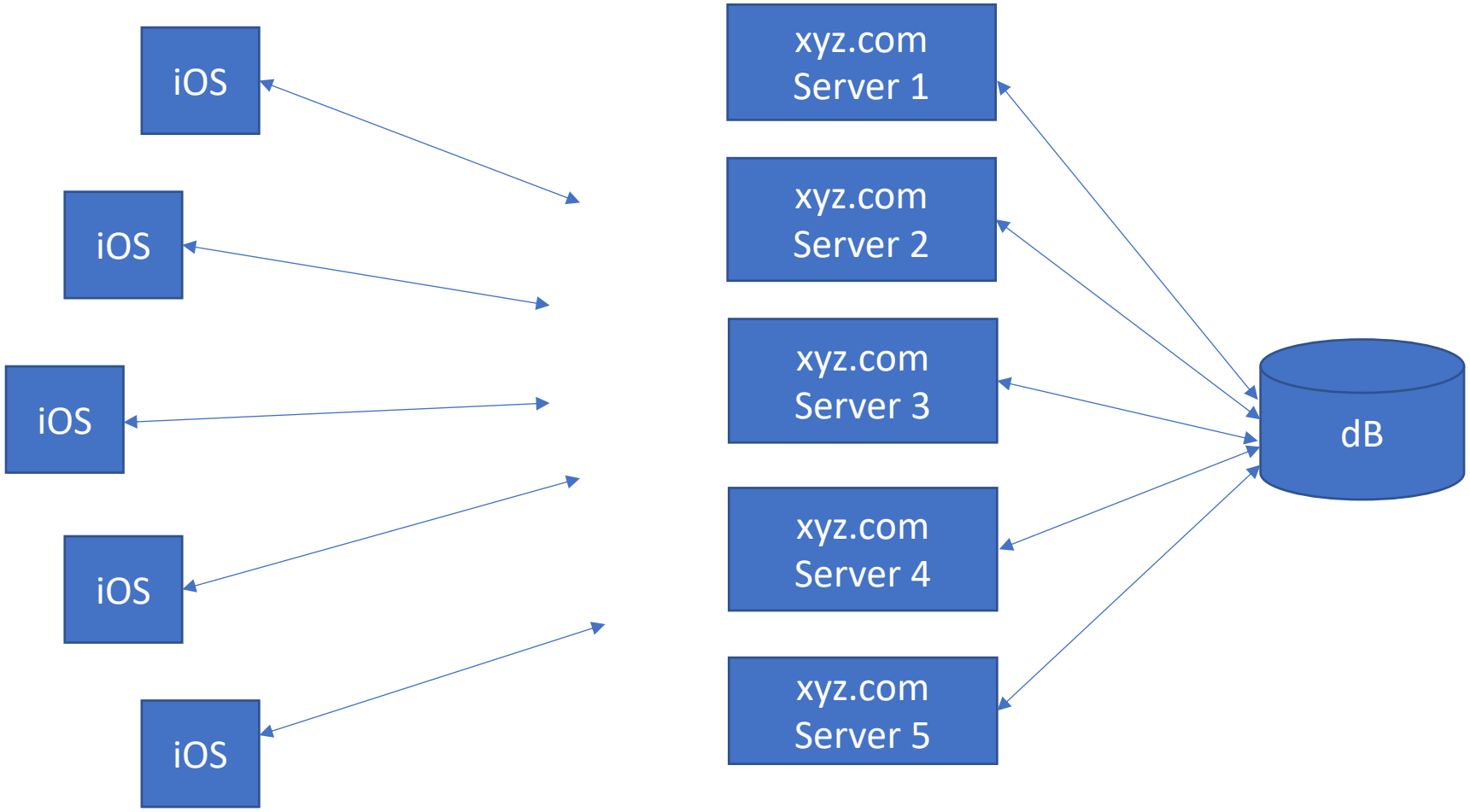


3. Horizontal Scaling

3. Horizontal Scaling

1. How should we architect our system to **not reach that ceiling** ?
2. Instead of one state of the art costlier server, we get multiple **cheaper servers to handle requests**. This is called **Horizontal Scaling**.
3. Number of servers varies almost **linearly** with #requests(10K req/s)

3. Horizontal Scaling



3. Horizontal Scaling

1. **Number of servers** varies almost **linearly** with **#requests**(1000 req/s). So we have **solved the scalability problem**
2. What **problems** arise now that we have 5 servers instead of 1?

3. Concerns with Horizontal Scaling

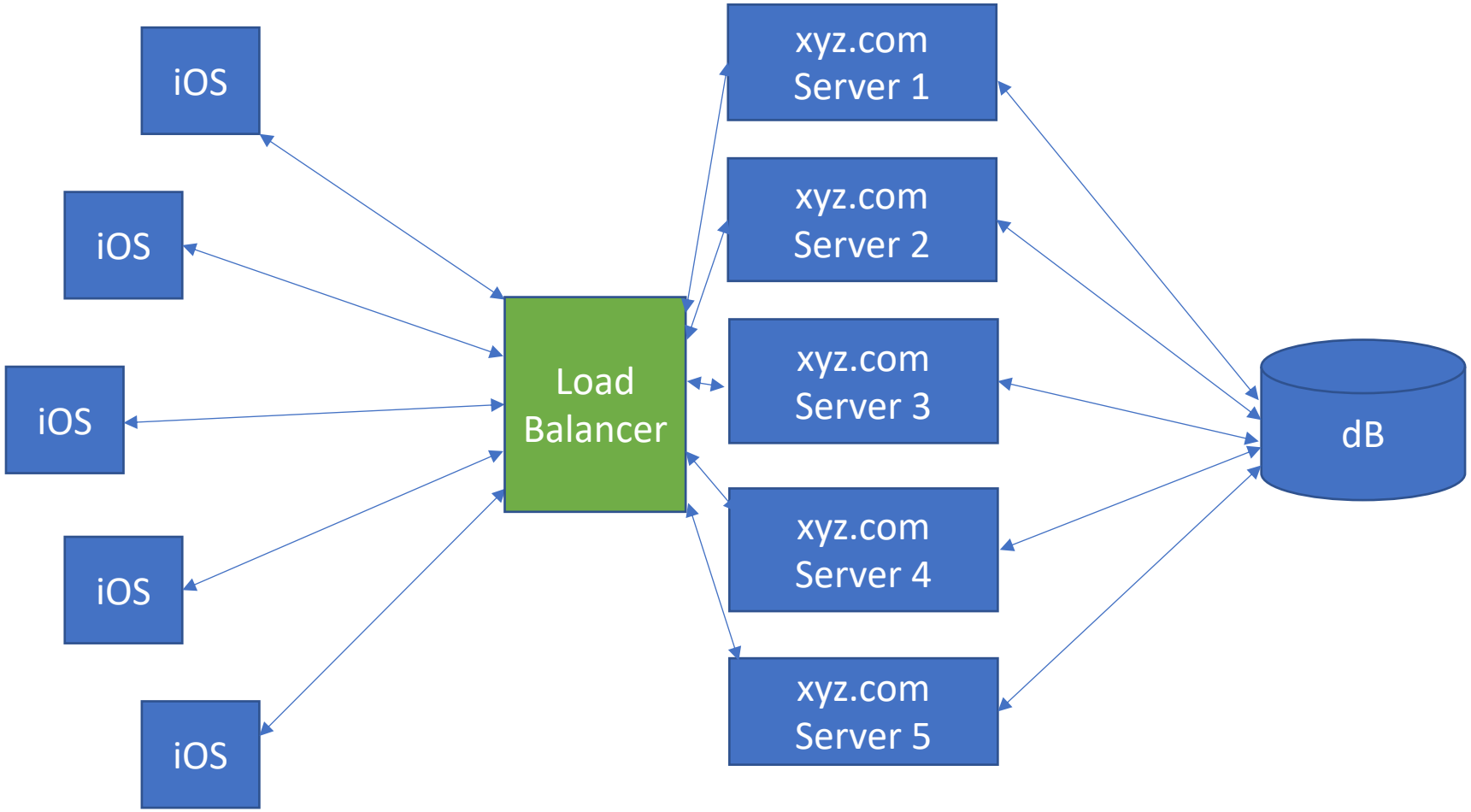
1. **One potential problem** is that we have a **single Postgre dB** which all 5 servers are trying to access at the same time and concerns might arise. We will understand dB scalability later.

2. Second, How to figure out **which server to send client requests to** ?

This is addressed by another piece of hardware that's sits between client and servers, **Load Balancer**

4. Load Balancer

4. Load Balancer



4. Load Balancer

How does a load balancer decide which server to send client requests to ?

4. Load Balancing Algorithms

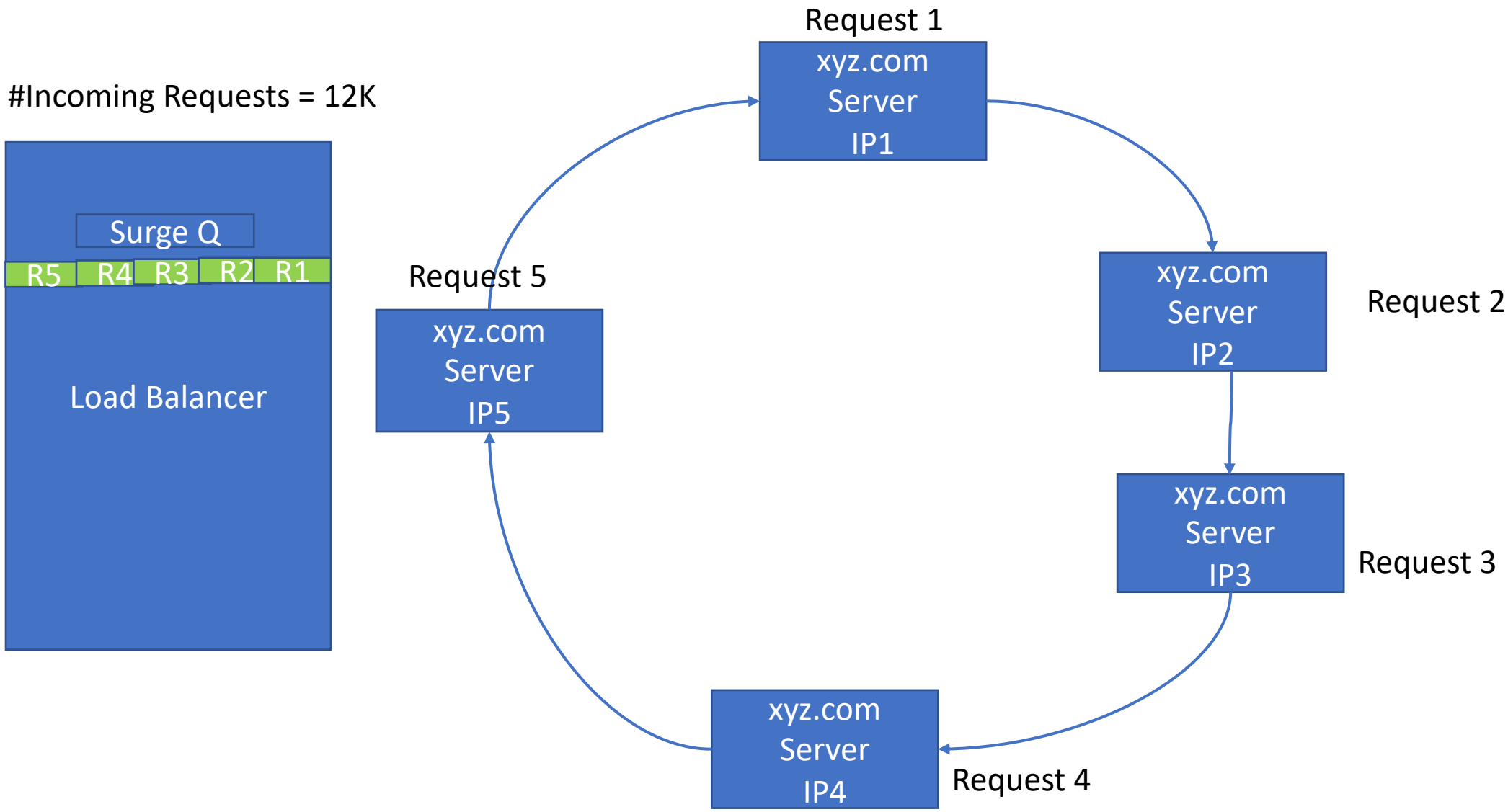
1. Random Choice
2. Round Robin/ Weighted Round Robin
3. Fewest Connections/ Least Loaded
4. Layer 4 load Balancing
5. Layer 7 load Balancing
6. IP Hashing Load Balancing

4.2 Round Robin Load Balancing

4.2 Round Robin Load Balancing

1. Each server has the domain **xyz.com** but **unique IP address**
2. **Load Balancer** has the list of all unique IP addresses for **xyz.com**
3. Incoming requests from Clients enq on a **surge Queue inside LB**.
4. Those requests are assigned in a **rotating sequential manner**
5. Request Cycle 1: Req 1→ IP1, Req2→ IP2, Req 3→ IP3, Req 4→ IP4, Req 5→ IP5
6. Request Cycle 2: Req6→ IP1, Req7→ IP2…

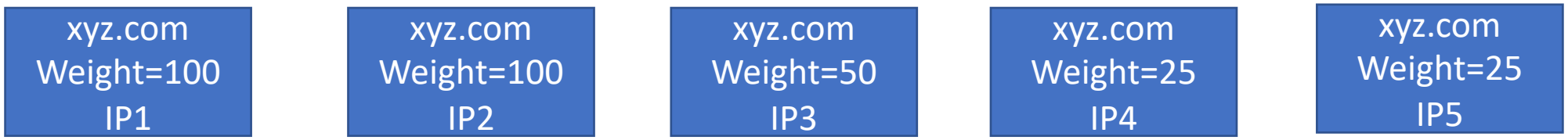
4.2 Round Robin Load Balancing



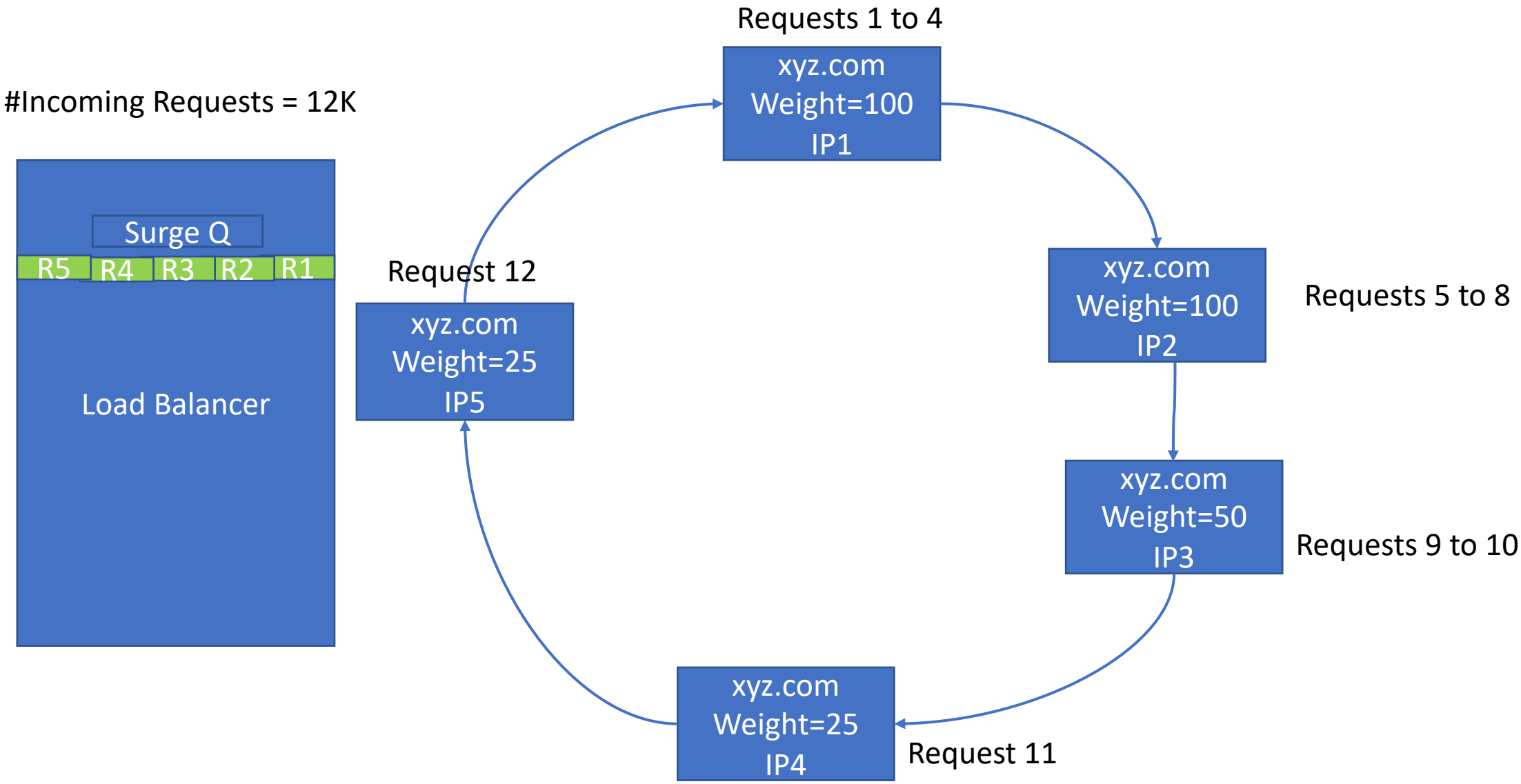
4.2 Weighted Round Robin Load Balancing

4.2 Weighted Round Robin Load Balancing

1. Let's say **Server 1** and **Server 2** are **most powerful** and highly efficient and remaining servers are less efficient.
2. So we **assign weights** to each server based on its performance capacity.
3. Incoming requests are assigned in a **rotating sequential manner based on server weights** on every request cycle.
4. Weights: S1: S2: S3: S4: S5 = 100: 100: 50: 25: 25 = 4: 4: 2: 1: 1



4.2 Weighted Round Robin Load Balancing



Drawbacks of Round Robin ?

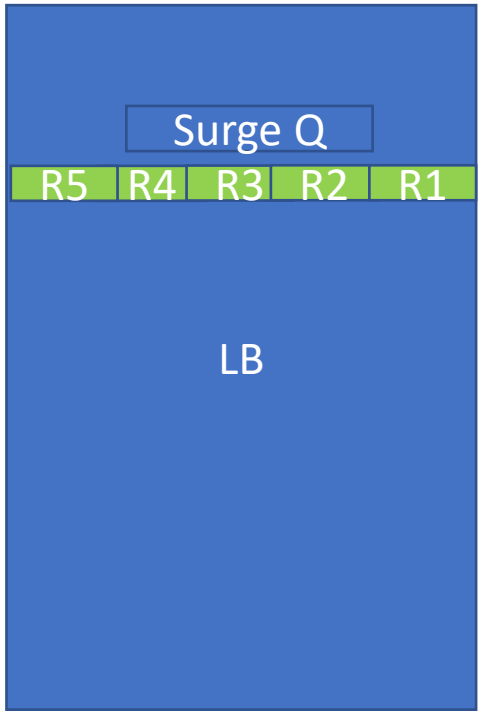
4.3 Fewest Connections

4.3 Fewest Connections

1. Neither Round Robin or Weighted Round Robin **take the current server load/connections into consideration** when distributing requests
2. **Active Connections**: Those HTTP or HTTPS requests that have not yet received a response.
3. Current Active Connections:

4.3 Fewest Connections

#Incoming Requests = 12K



xyz.com
AC=3
IP1

xyz.com
AC=5
IP2

xyz.com
AC=20
IP5

xyz.com
AC=0
IP3

xyz.com
AC=10
IP4

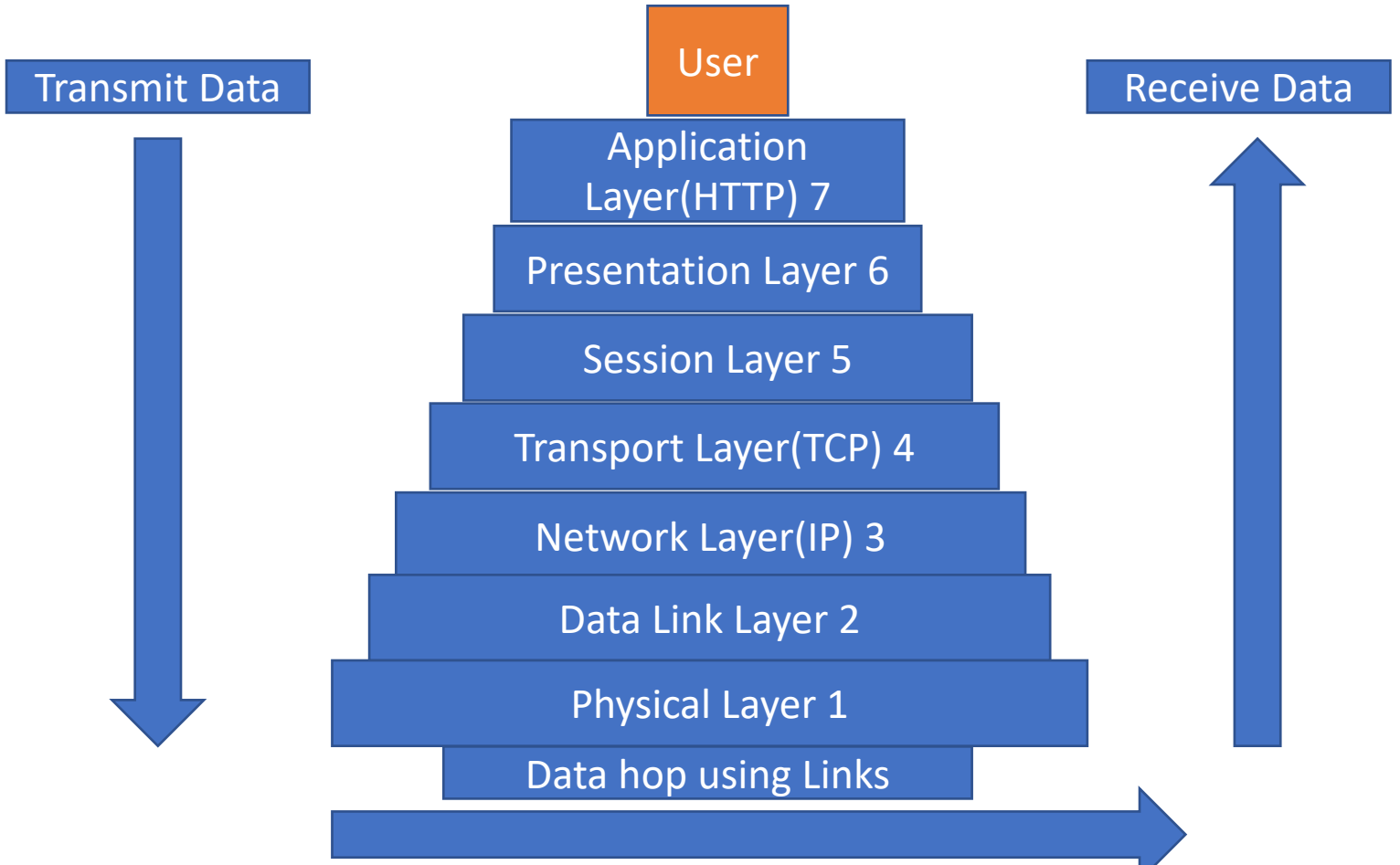
4.3 Fewest Connections

Request #	Server 1(AC= 3)	Server 2(AC=5)	Server 3(AC=0)	Server 4(AC=10)	Server 5(AC=20)
R1	3	5	1	10	20
R2	3	5	2	10	20
R3	3	5	3	10	20
R4	4	5	3	10	20
R5	4	5	4	10	20
R6	5	5	4	10	20
R7	5	5	5	10	20

4.4 Layer 4 Load Balancing

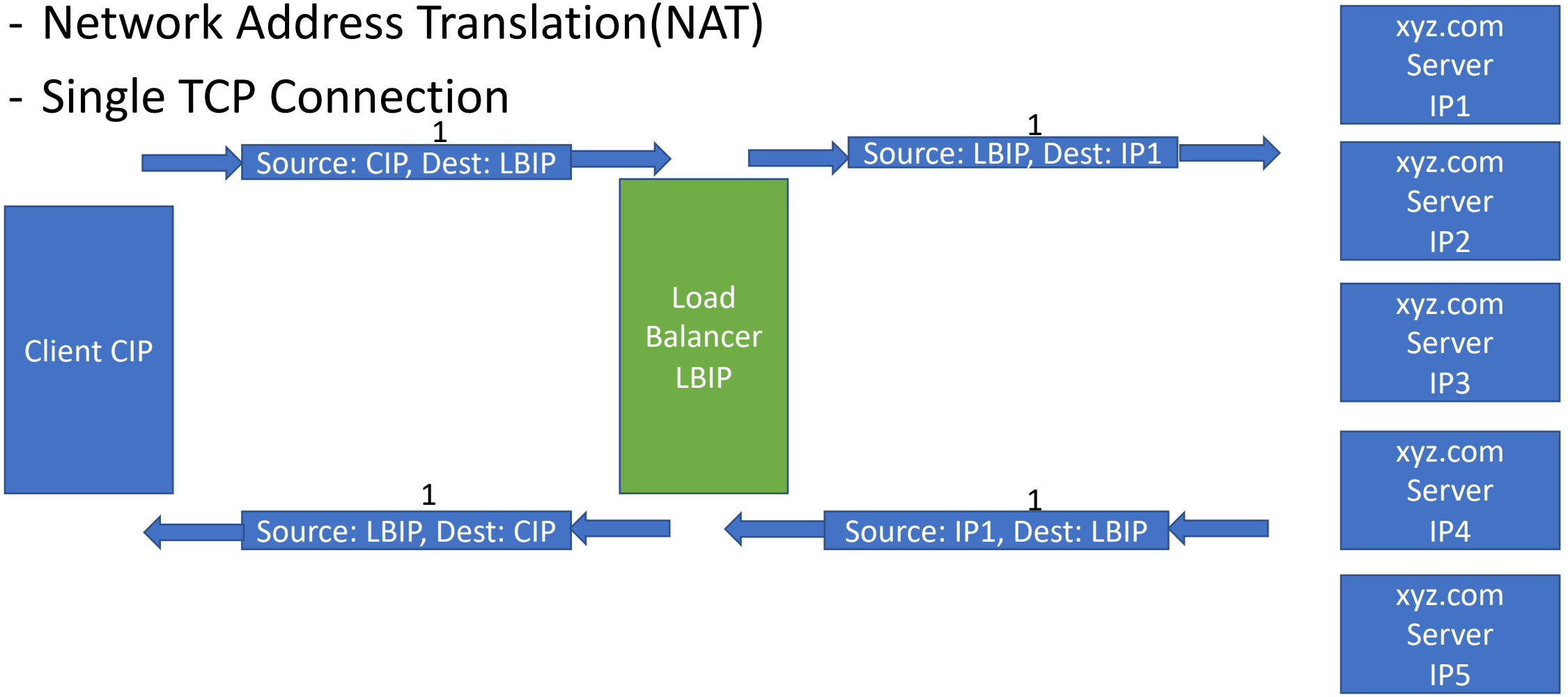
4.4 Layer 4 Load Balancing

7 Layer OSI Model:



4.4 Layer 4 Load Balancing

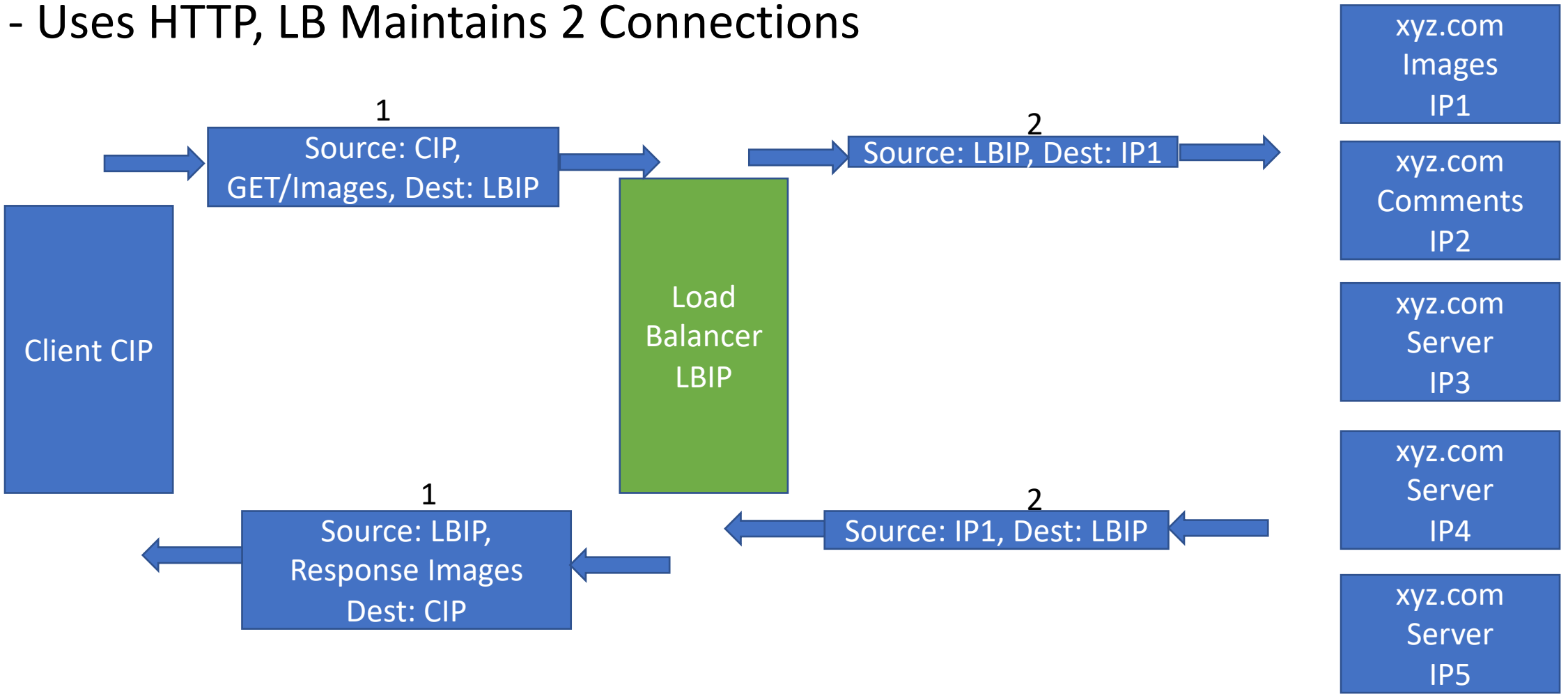
- Network Address Translation(NAT)
- Single TCP Connection



4.5 Layer 7 Load Balancing

4.5 Layer 7 Load Balancing

- Uses HTTP, LB Maintains 2 Connections



5. IP Hashing Load Balancing

5. IP Hashing Load Balancing

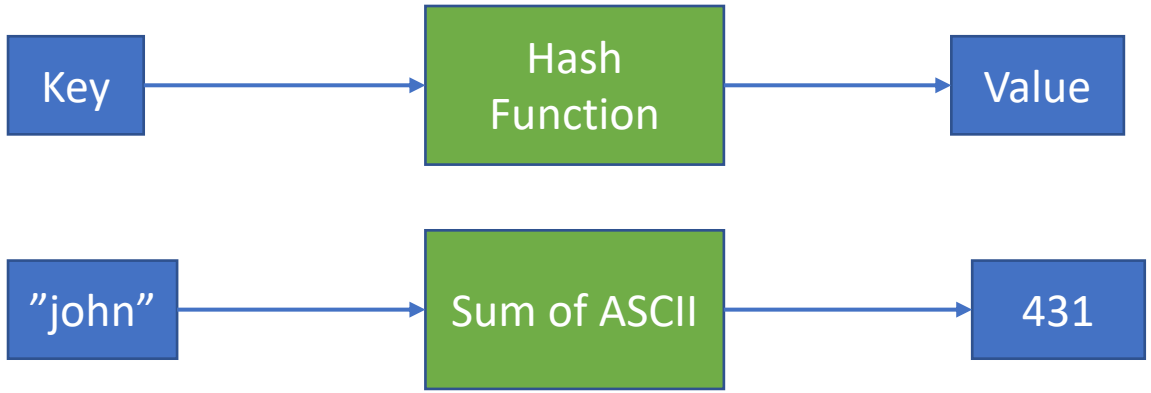
How do we load balance if the **number of servers change dynamically**?

What happens if we want to **add/remove servers** based on client load?

How do we **manage client sessions** during load balancing ?

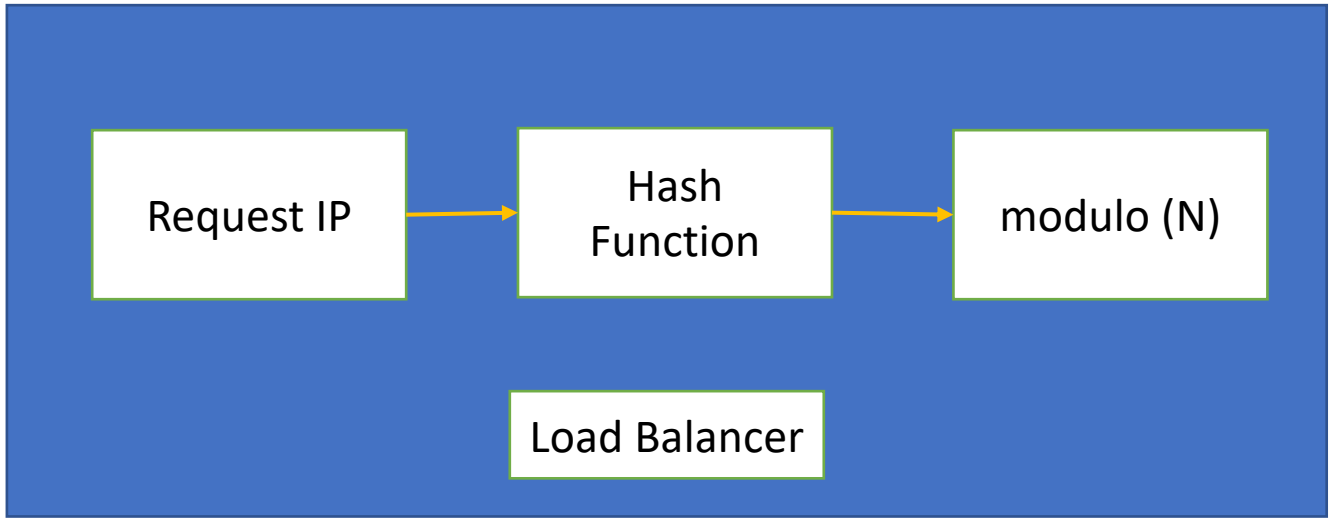
5. IP Hashing Load Balancing

1. What is Hashing ?
2. Choosing an **efficient Hash function** to avoid collision(one to one mapping)



5. IP Hashing Load Balancing

N = Number of Servers = 5



xyz.com
Server ID - 0

xyz.com
Server ID - 1

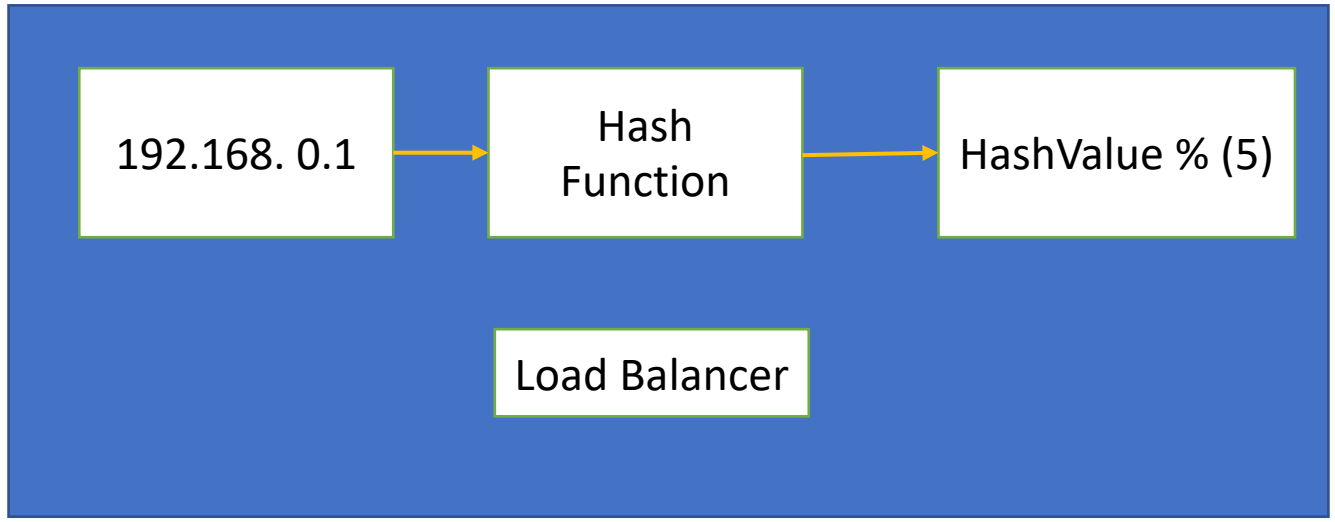
xyz.com
Server ID - 2

xyz.com
Server ID - 3

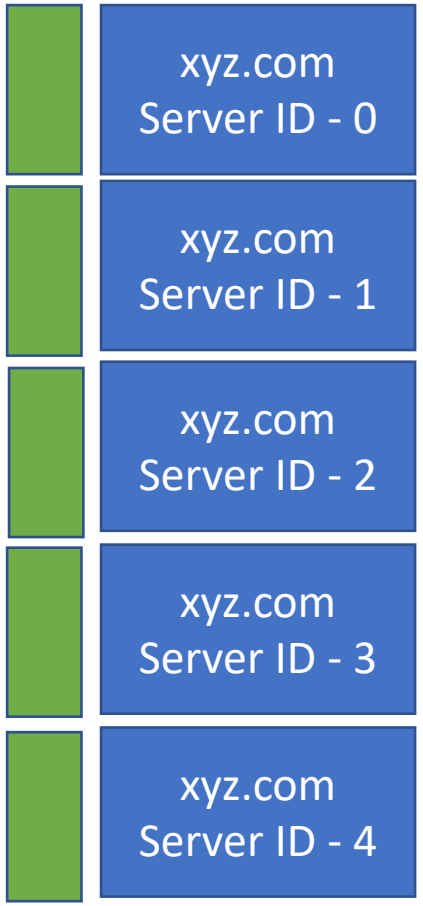
xyz.com
Server ID - 4

5. IP Hashing Load Balancing

Hash Value	85	99	62	38	71
Server ID	0	4	2	3	1



Session Cache



5. IP Hashing Load Balancing

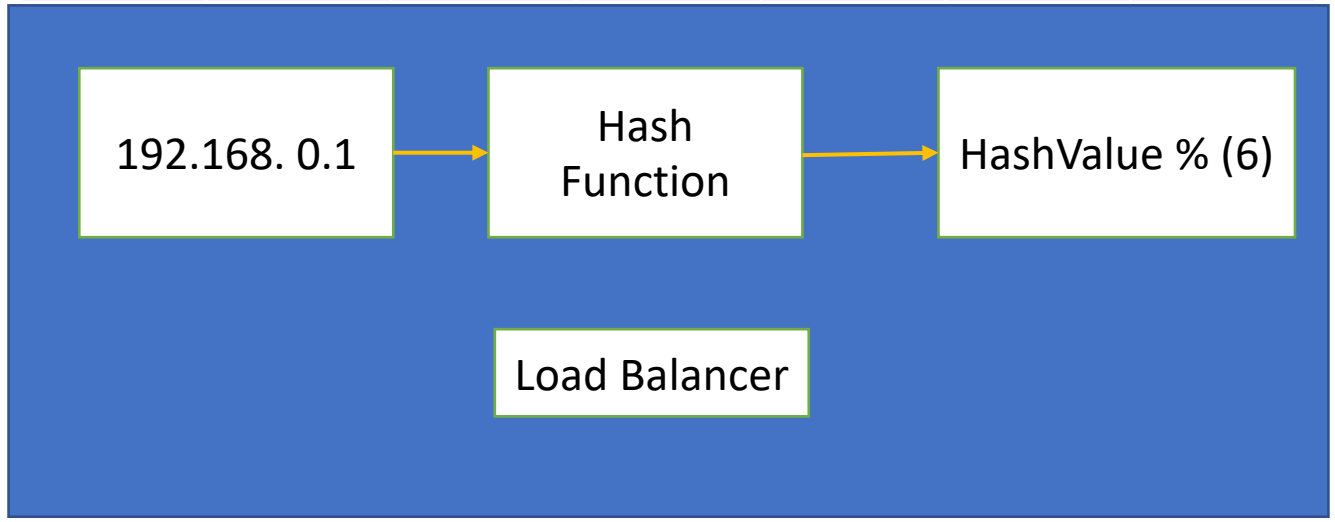
1. Incoming **Requests are mapped** to a server based on a hashing function **randomly**.
2. **Session Awareness**: The same server ID can be generated for next request in same session because the **request IP and hash function is same**.
3. **Server cache** can be reused.

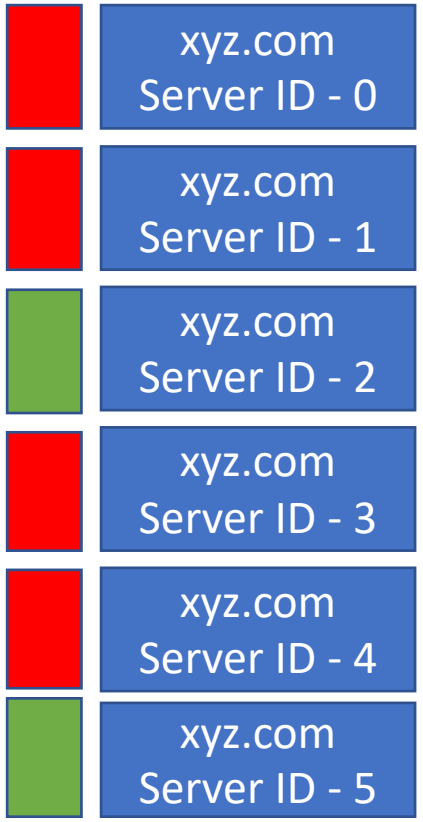
5. IP Hashing Load Balancing

What happens if we want to **add a new server to my cluster** ?

5. IP Hashing Load Balancing

Hash Value	85	99	62	38	71	42
Old SID	0	4	2	3	1	NA
New SID	1	3	2	2	5	0





Invalidated Session Cache

Session Cache

5. Concerns with IP Hashing Load Balancing

1. All server **cache gets invalidated** because of **redirection** caused by addition of new server. This is costly if we have >10K servers.

2. How can we **reduce the impact on other servers** while adding or removing servers?

5. Consistent Hashing Load Balancing

- We follow a three step process:

Step 1: **Map requests** to locations on a ring

Step 2: **Map servers** to locations on the ring

Step 3: **Move clockwise** and map requests to servers

5. Consistent Hashing Load Balancing

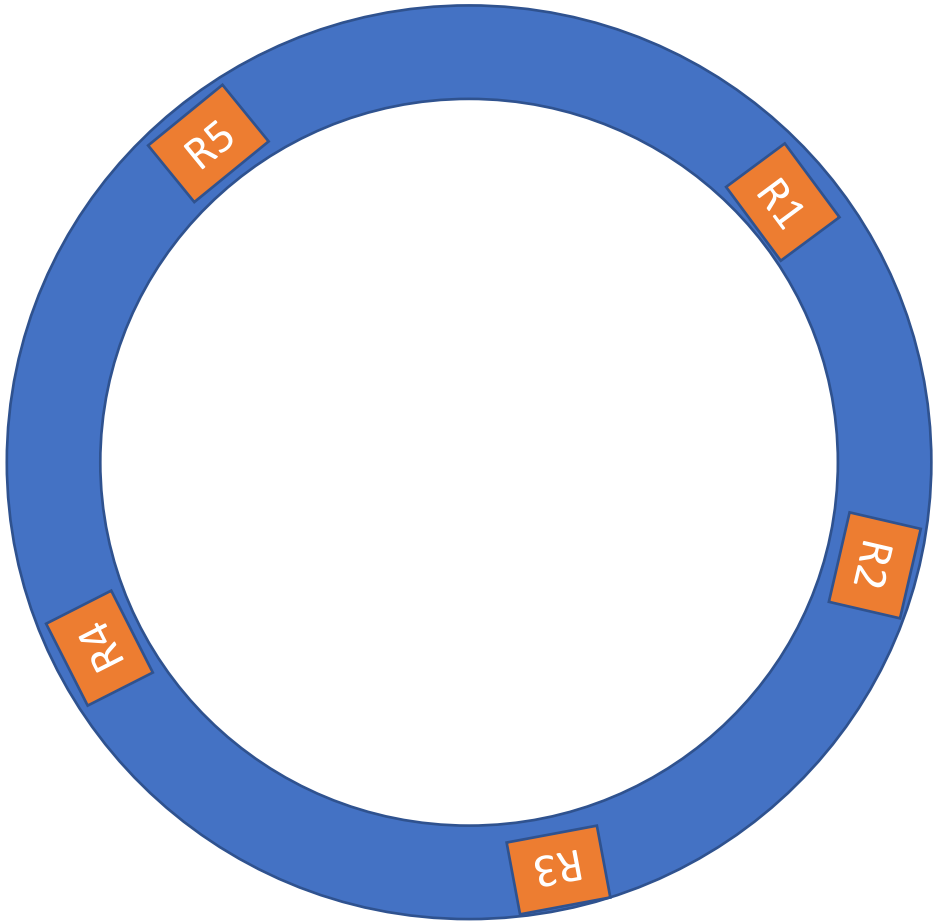
Step 1: **Map requests to locations** on a ring using hash function H

- Min Hash Value = 0 degrees, Max Hash Value = 360 degrees

Req ID/IP	1	2	3	4	5
Hash Value	345	269	434	865	1056
Angle	0	56.5	100.3	170.4	220.7

5. Consistent Hashing Load Balancing

Request Mapping:



5. Consistent Hashing Load Balancing

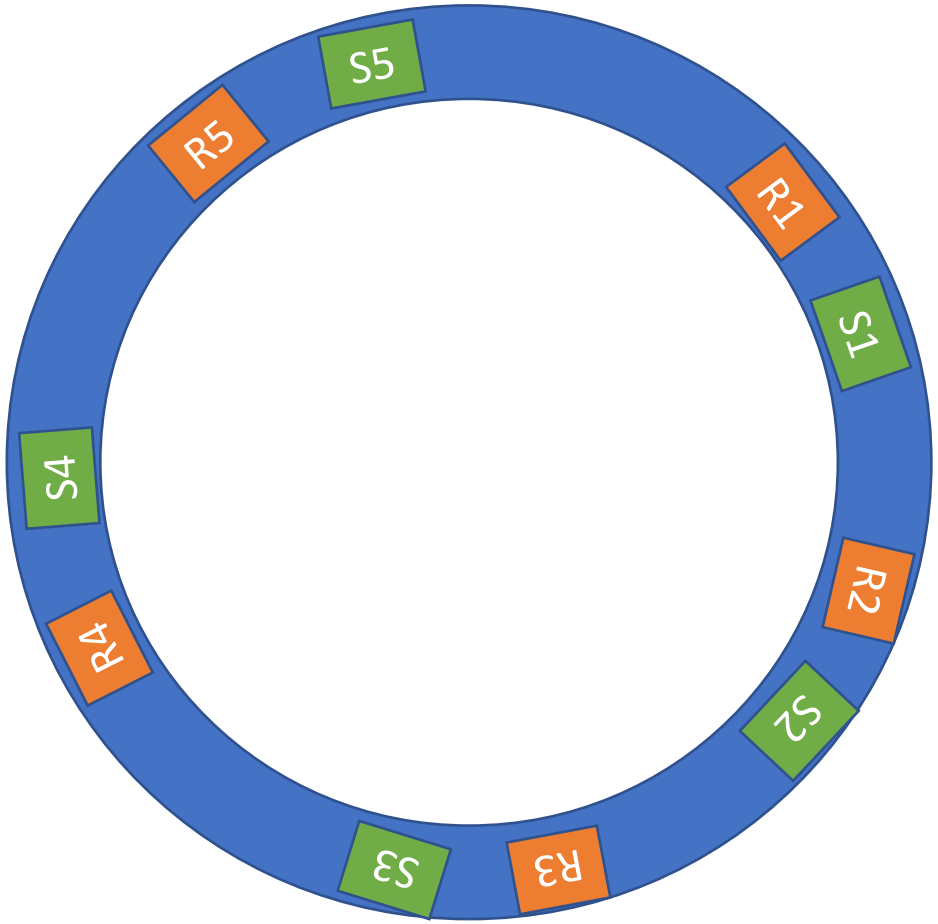
Step 2: **Map servers to locations** on a ring using same Hash function H

- Min Hash Value = 0 degrees, Max Hash Value = 360 degrees

Server ID/IP	10	12	14	16	18
Hash Value	20	300	543	810	1259
Angle	10.1	70.2	120.9	210.0	340.7

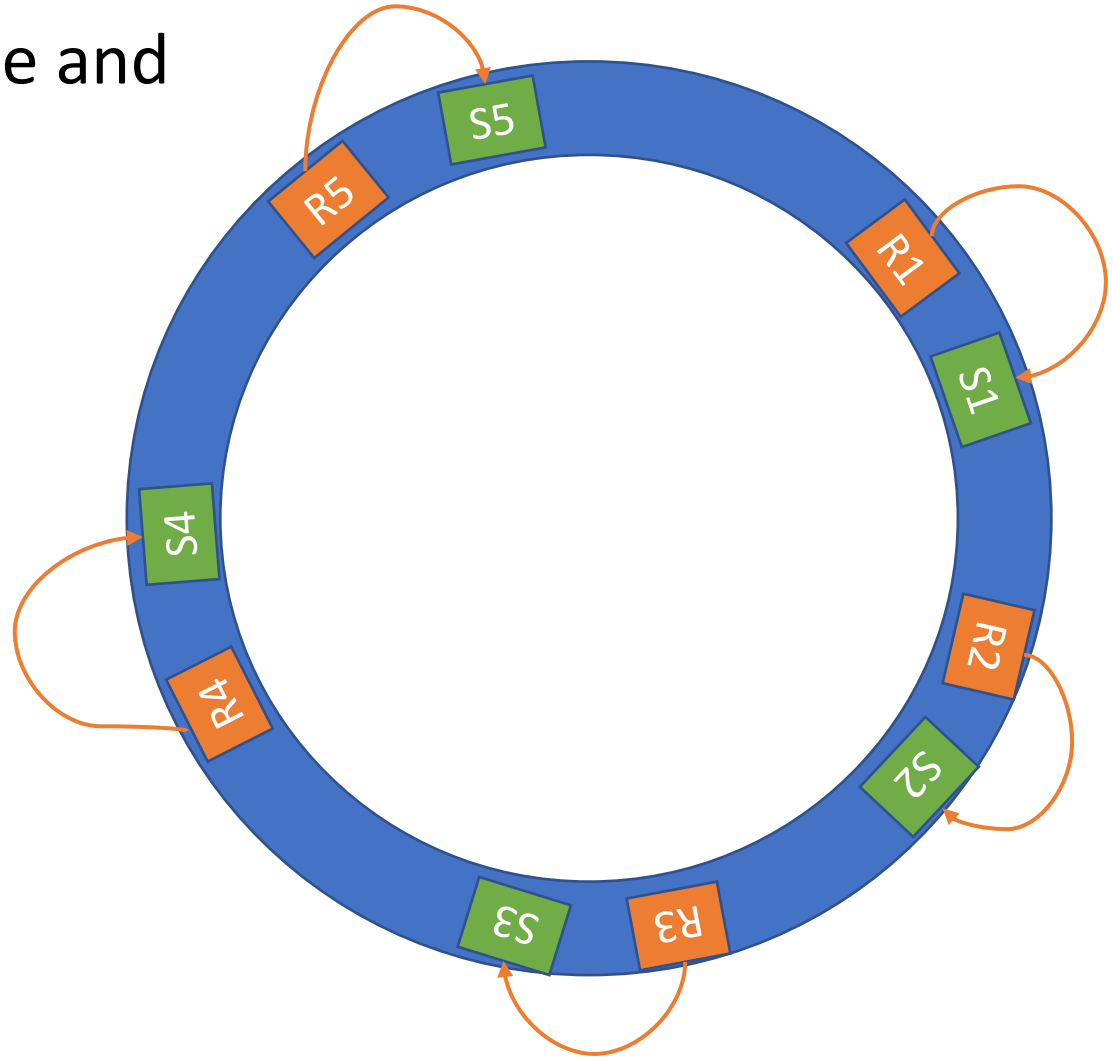
5. Consistent Hashing Load Balancing

Server Mapping:



5. Consistent Hashing Load Balancing

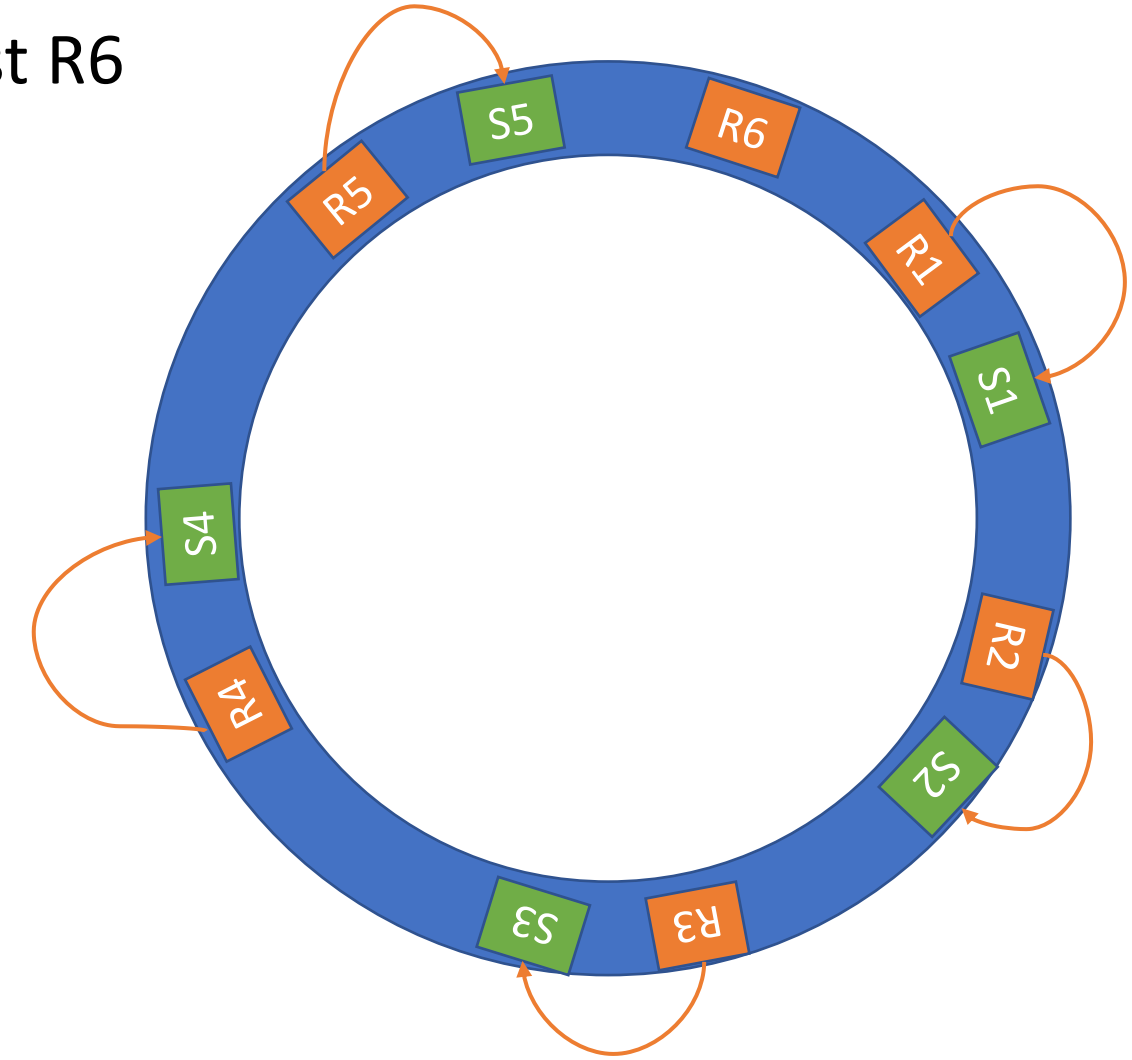
Move clockwise and map requests to servers:



5. Consistent Hashing Load Balancing

A new Request R6

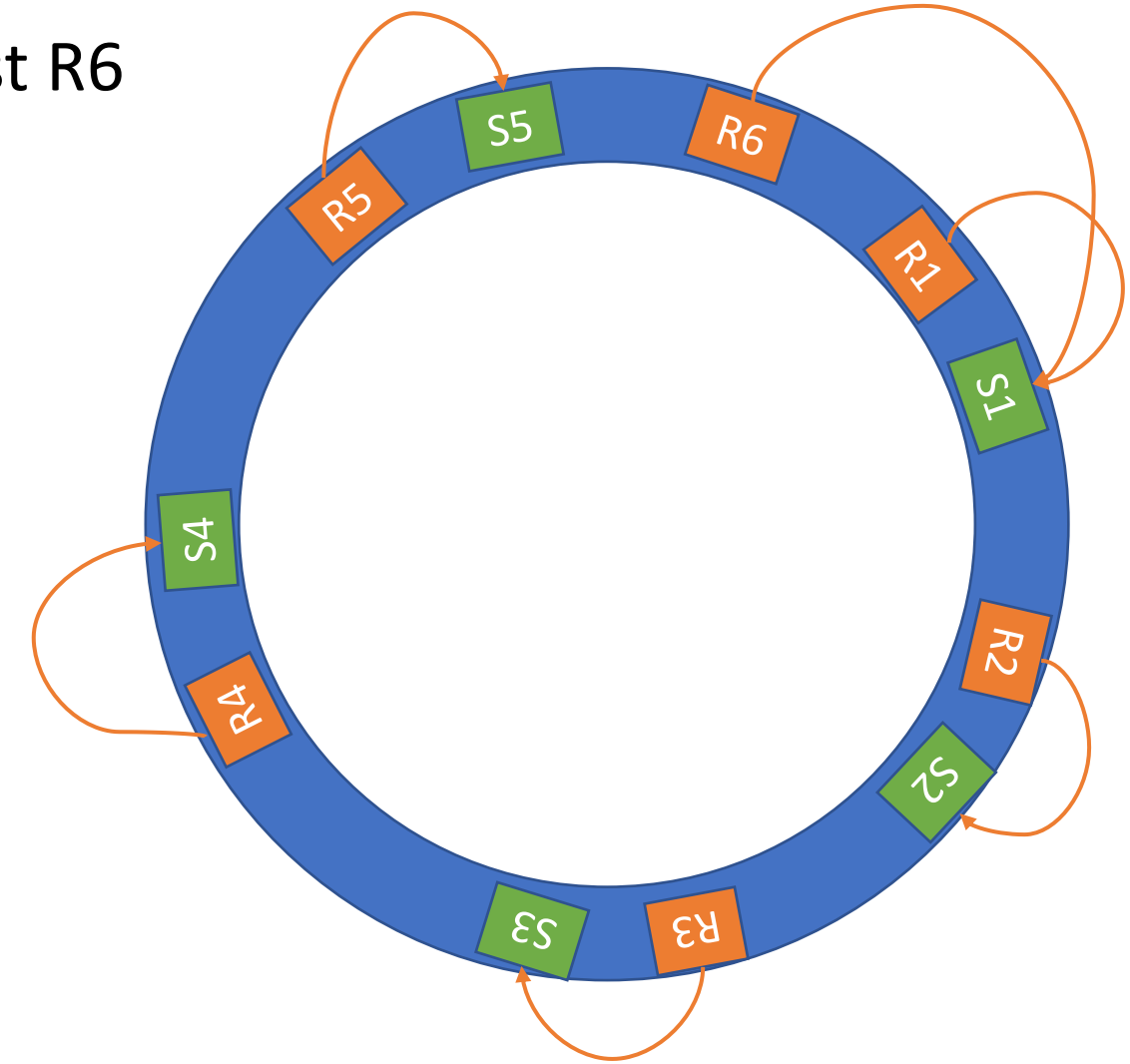
Comes in:



5. Consistent Hashing Load Balancing

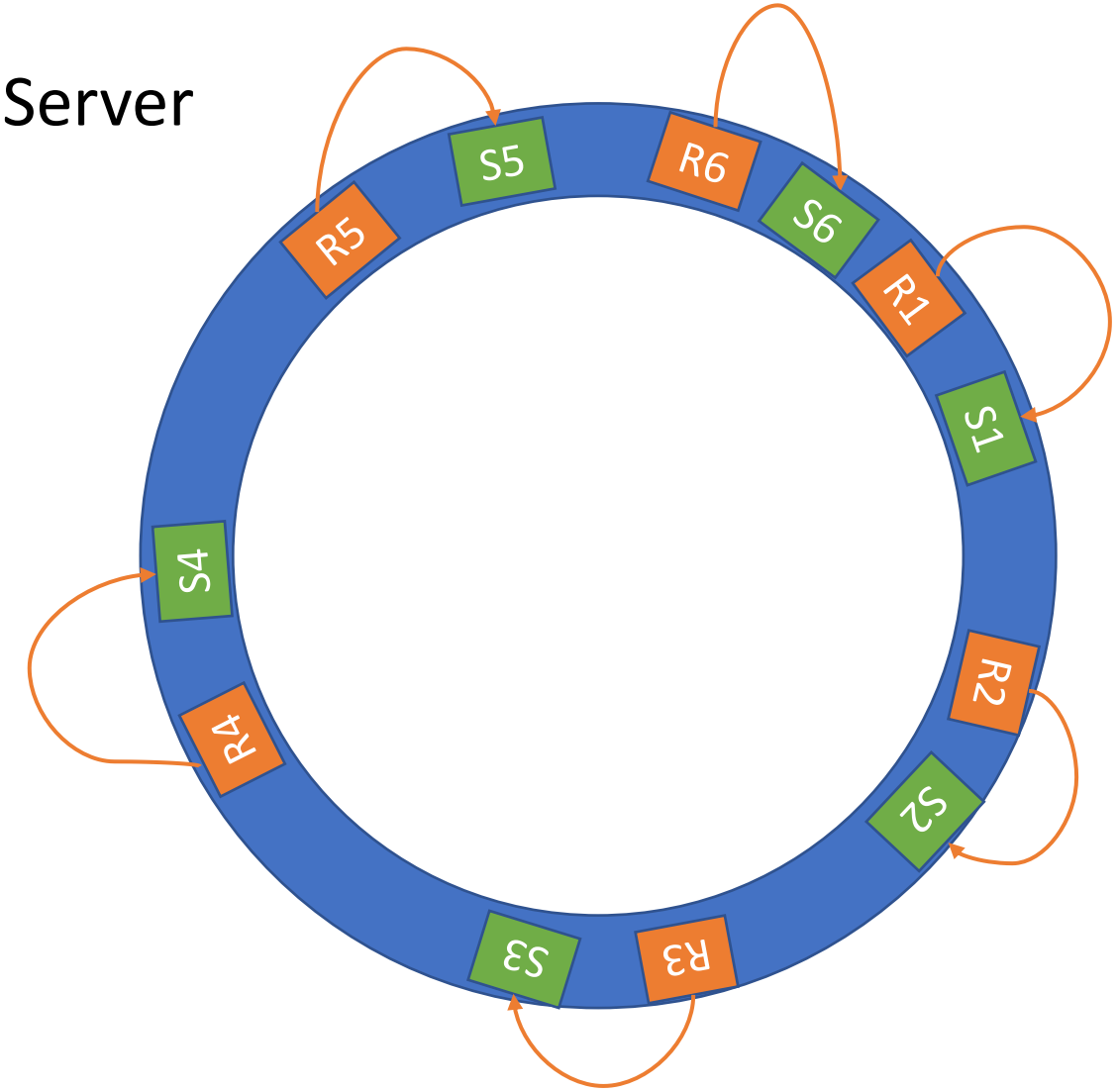
A new Request R6

Comes in:



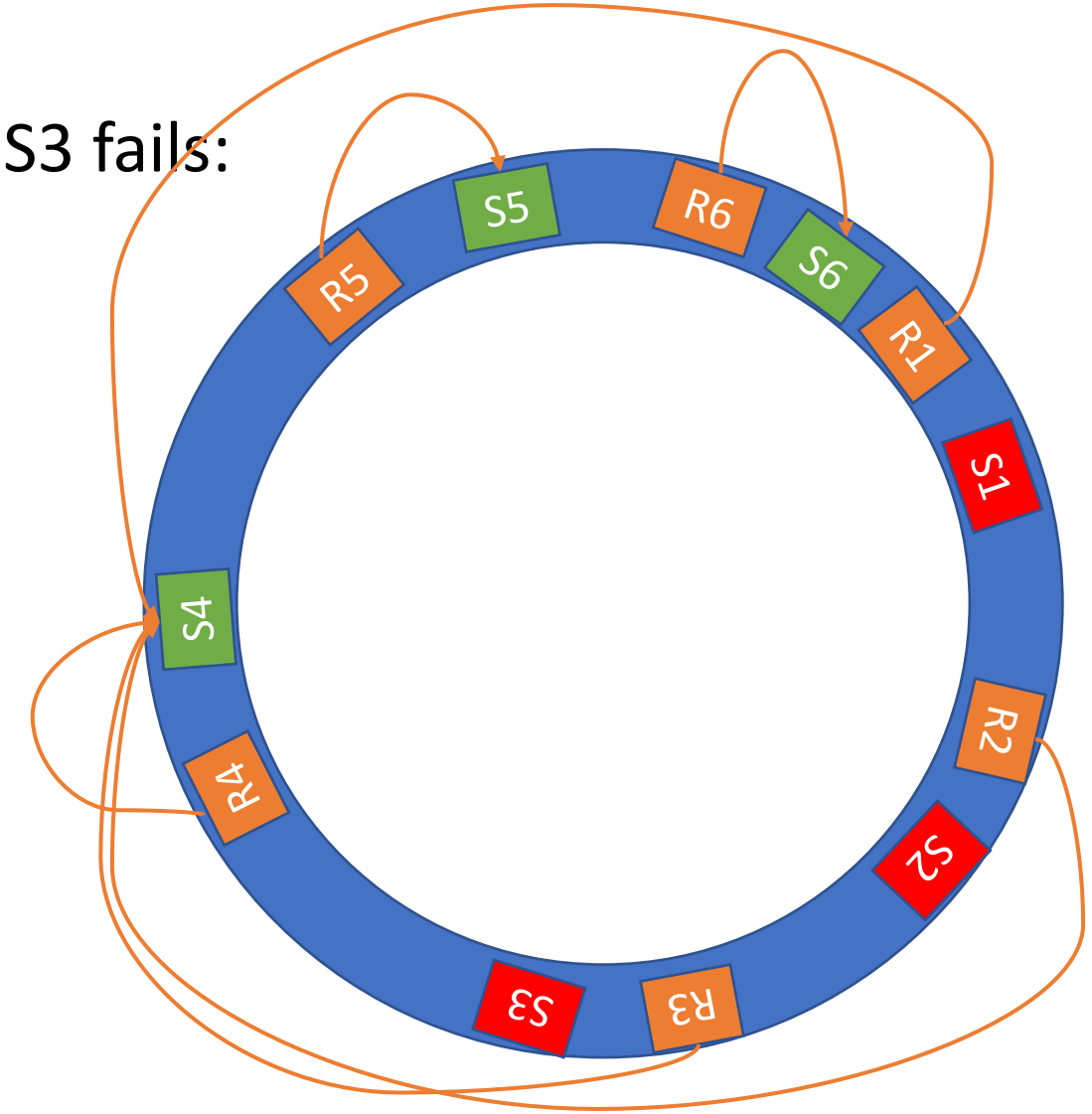
5. Consistent Hashing Load Balancing

Adding a new Server S6:



5. Consistent Hashing Load Balancing

Server S1, S2, S3 fails:



5. Consistent Hashing Load Balancing

1. **Virtual Nodes**: Use **another hash function** to map position of servers S4, S5, and S6

2. Now Server S4, S5 and S6 have **two virtual positions** each. This helps us in **uniform distribution of load** among remaining healthy servers.

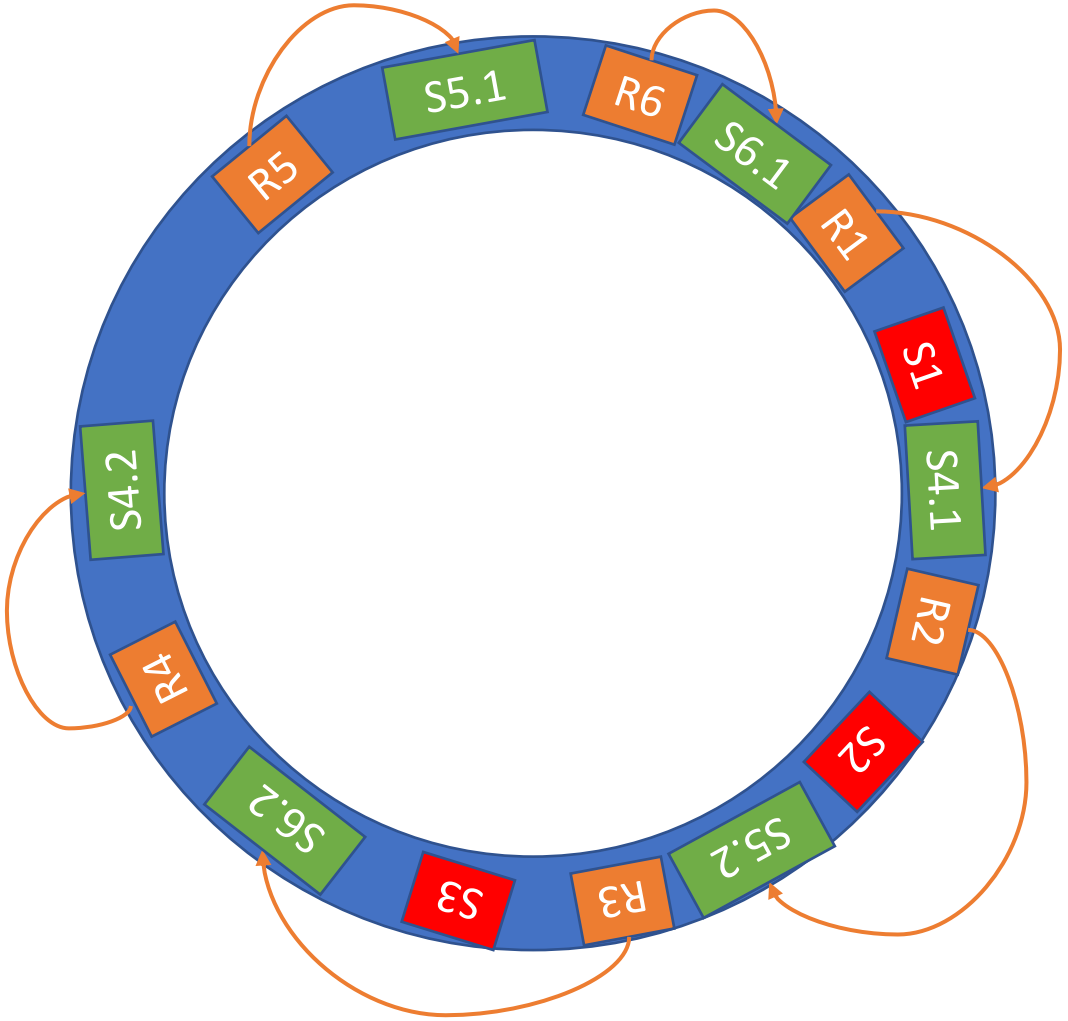
5. Consistent Hashing Load Balancing

Virtual Nodes:

S4: 1, 2

S5: 1, 2

S6: 1, 2



Stay Tuned for next Session On

Database Scaling, Caching

Questions