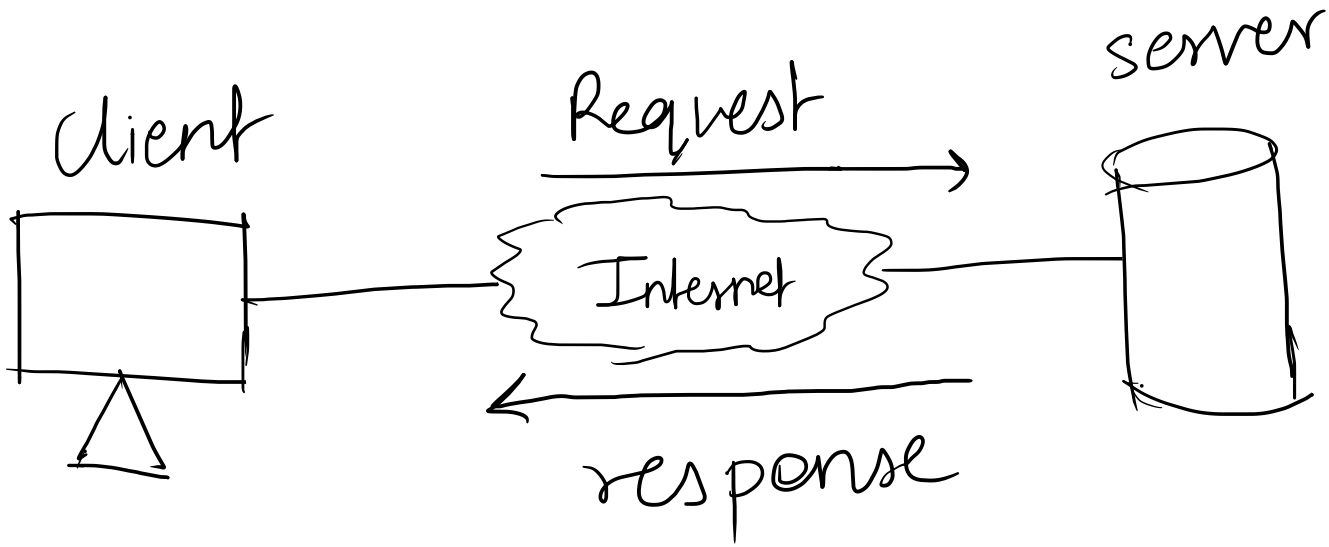
Client - Server Model

Client : Service requester

Server : Service provider



Client - Server model

DNS (Domain name system)

→ User enters URL

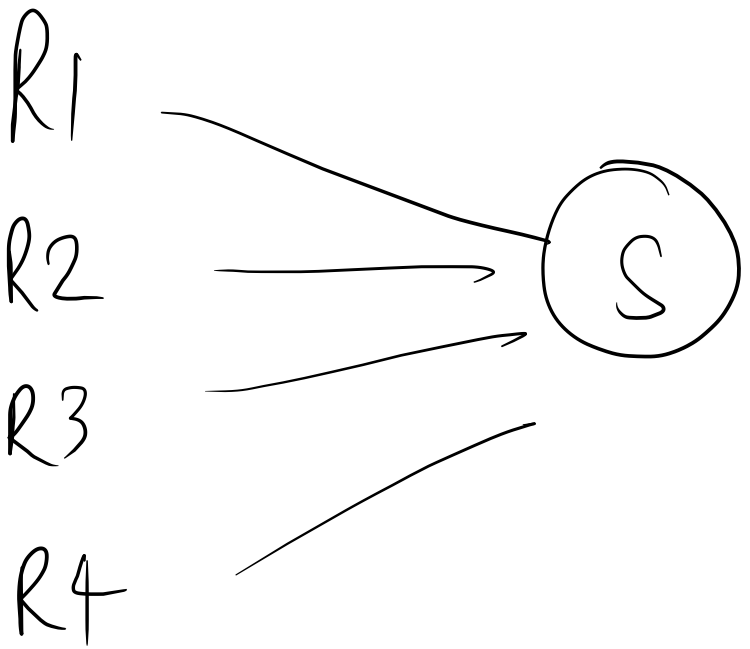
→ DNS server lookup for the address

→ HTTP / HTTPS request

→ Server sends the files of website

Requests

Server



Popular Website

Requests ↑

(heavy traffic)

Server

(load ↑)

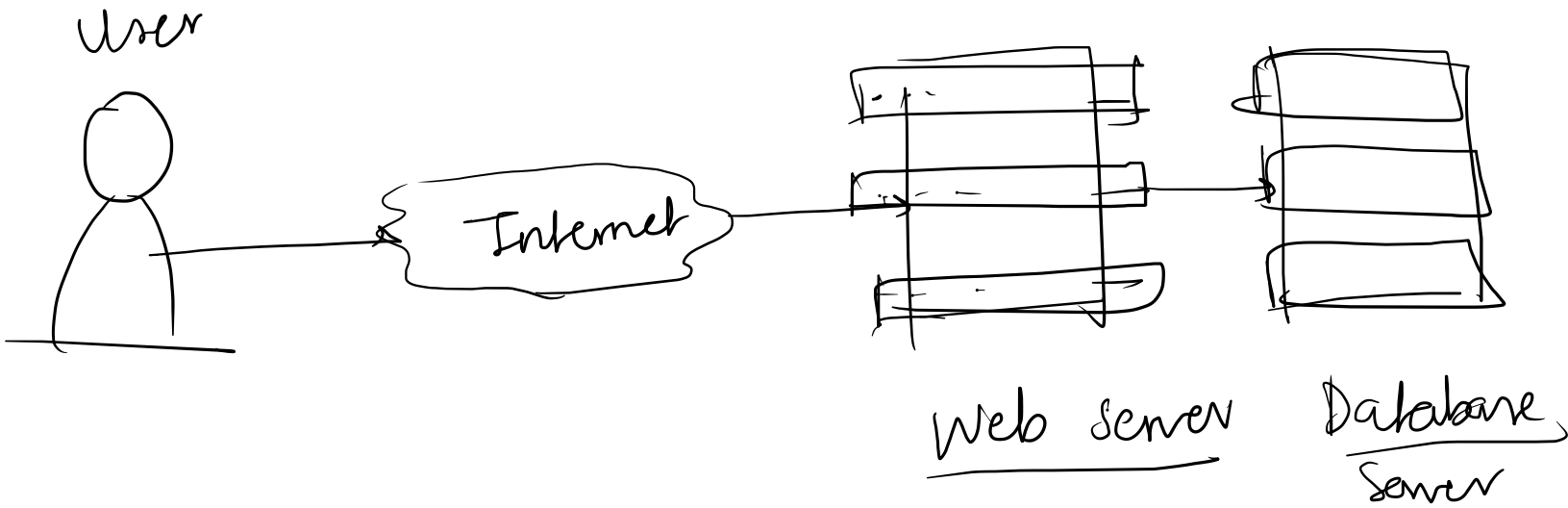
Application Scaling

Horizontal Scaling

↑ no. of servers

vertical Scaling

↑ size of server


User
Internet
Web server
Database
Server

Problems

① Single Point of failure

If server goes down, service unavailable

→ Bad experience for users

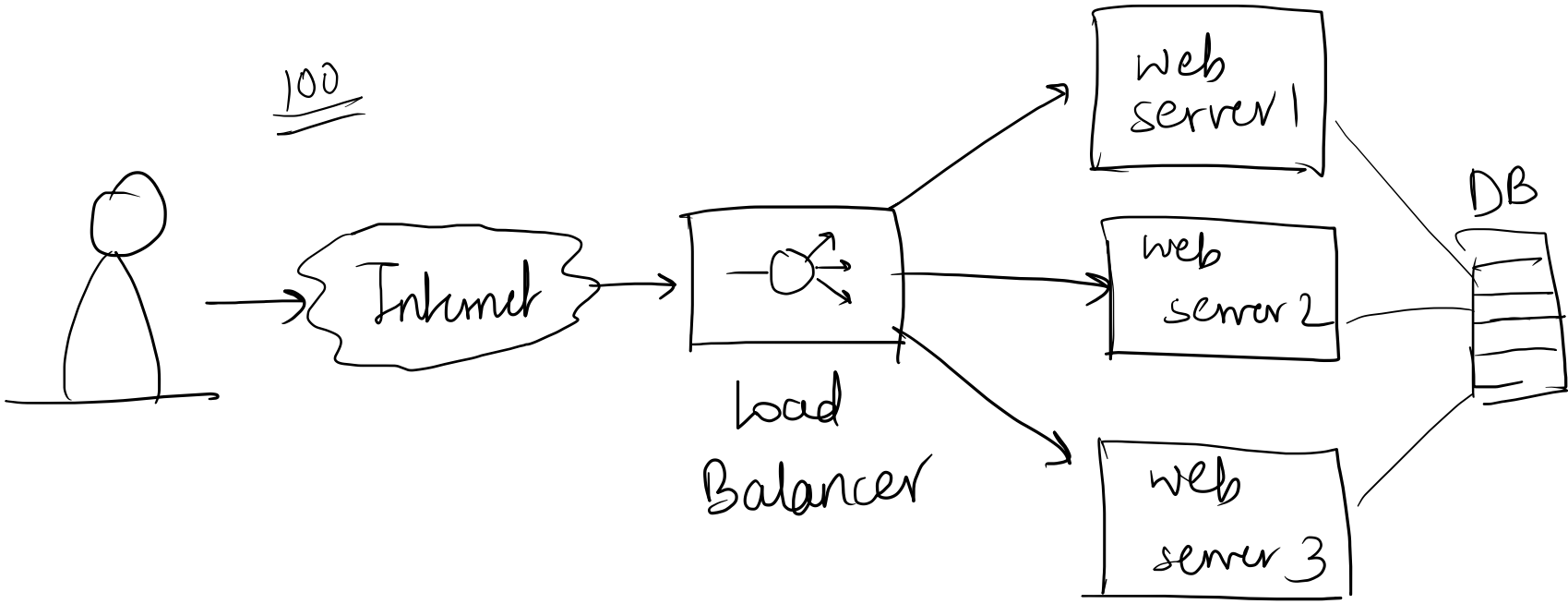
→ Unacceptable for service providers

(2) Overloaded servers

→ Limitation of no. of requests that a server can handle

→ Requests ↑ server overloaded

LOAD BALANCER



Solution :

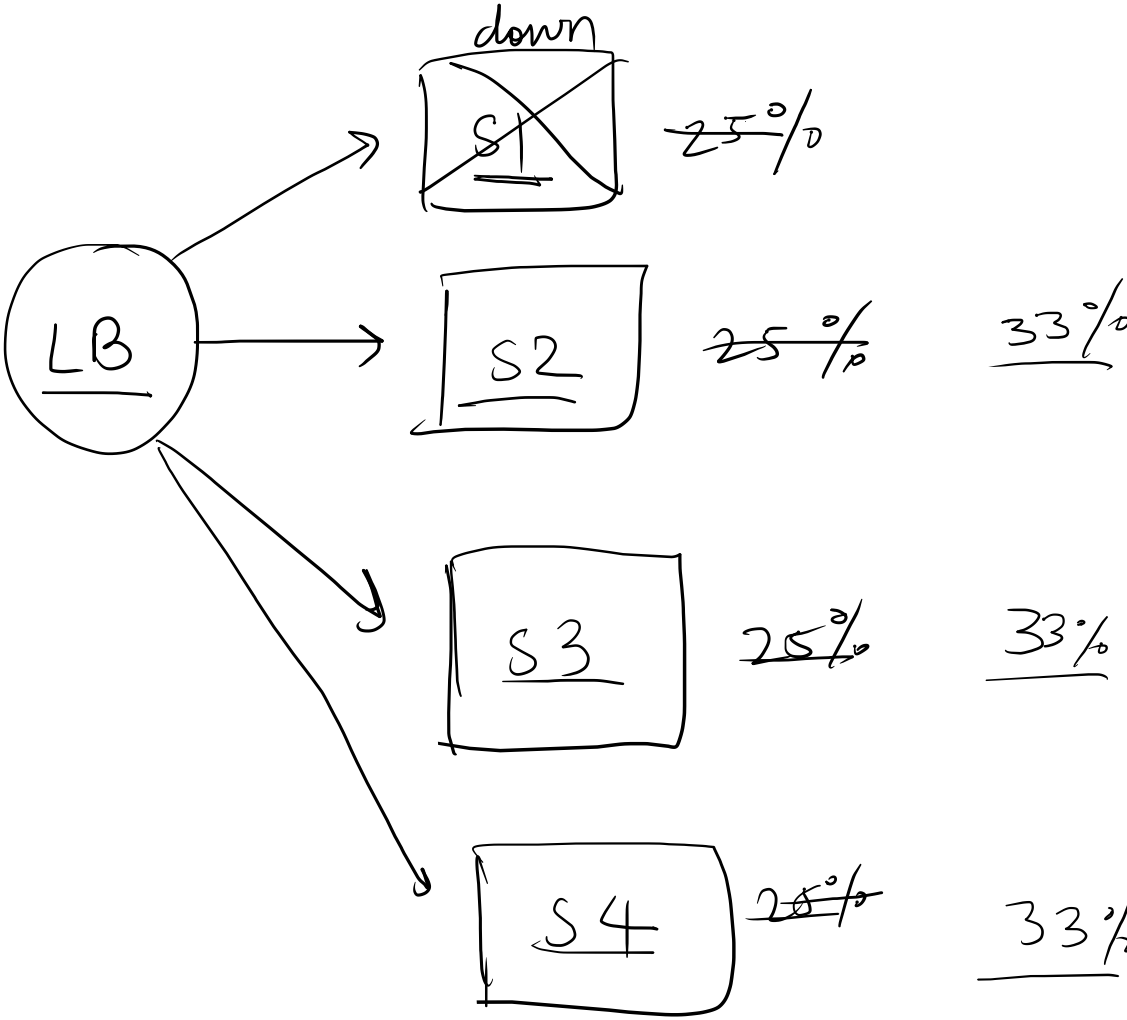
1.) Add a L.B. in front of servers
2.) Handle multiple requests (heavy volume)
3.) Share the requests across multiple servers
4.) Even if one of the servers go down, service will continue.
5.) faster response time, low latency

→ L.B. minimize response time

→ maximise throughput

→ High availability and reliability

→ continuous health checks to monitor servers capability of handling requests


down
S1
25%
LB
S2
25%
33%
S3
25%
33%
S4
25%
33%

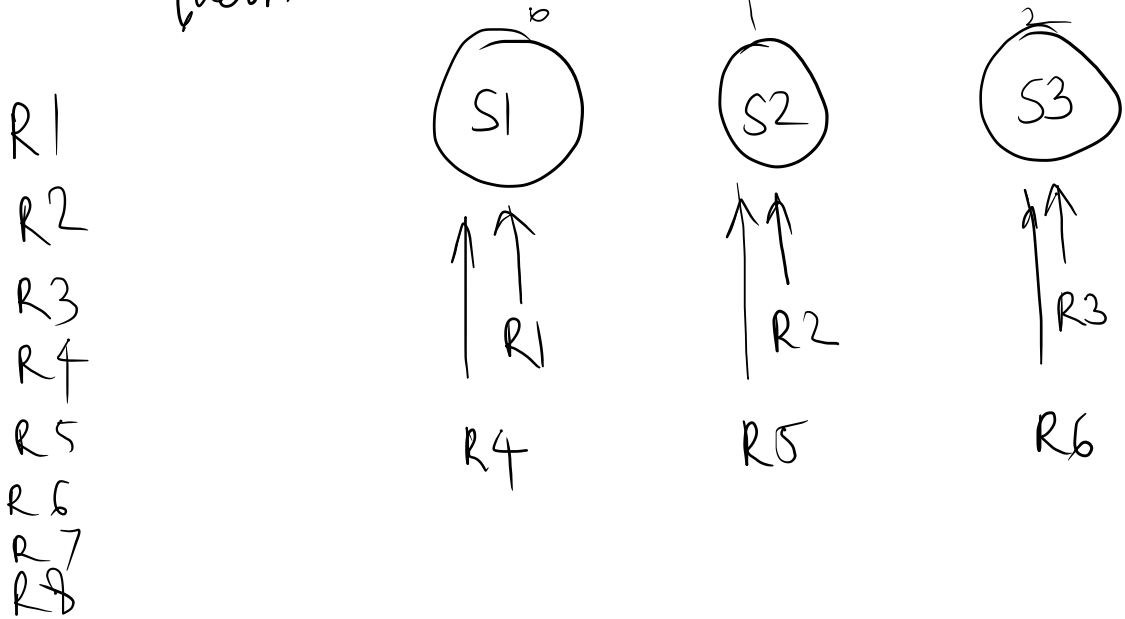
Horizontal Scaling

Load Balancer ✓

Load Balancing Algorithms

(1) Round Robin Algorithm

Requests are distributed in a sequential/rotational manner



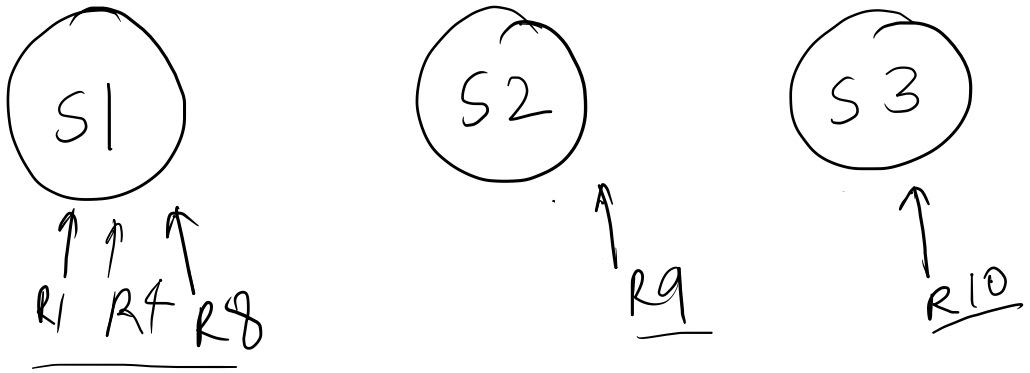
$R_i \% N$

Round Robin limitation :

→ doesn't consider the load already on the server

→ risk of single server overloading

(2) Least Connection Method



* Requests directed to the server with fewest no. of requests / active connections
* expensive, as L.B. needs to compute, to identify the server with least no. of requests