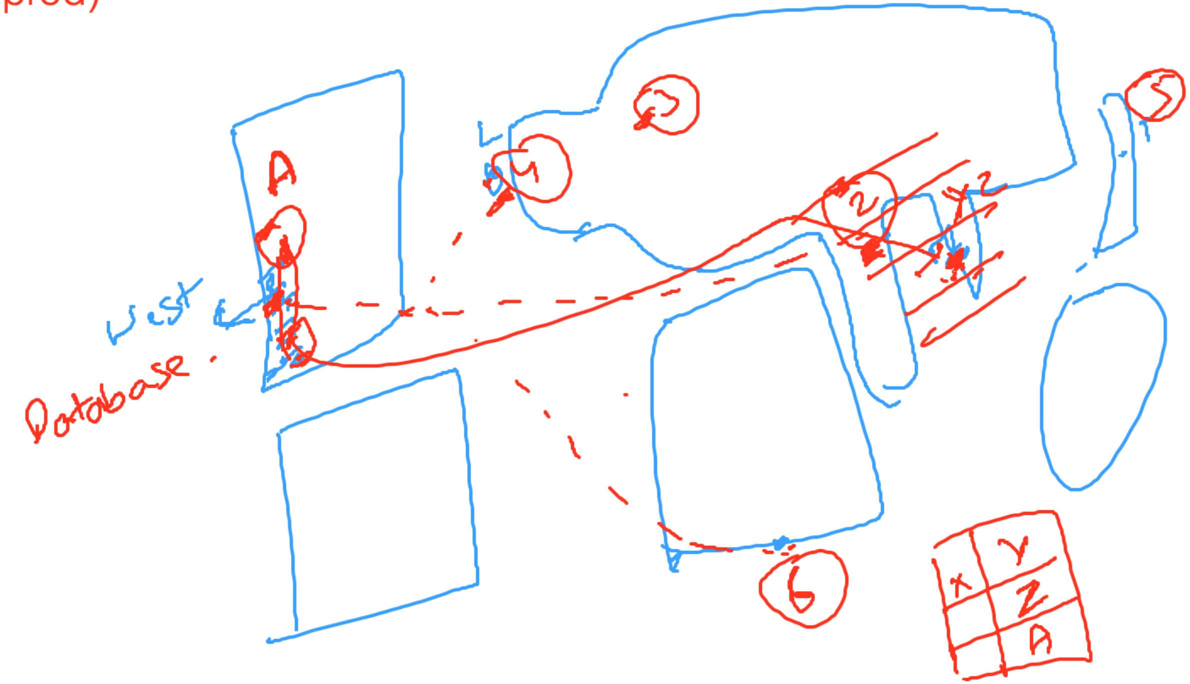
Goal of System Design : Scalable and Maintainable

LLD : OOPS, DP, System Designs

HLD
- LLD vs HLD
- LLD : Code, Servers, Databases (local, pre-prod, prod)
- HLD : Infrastructure(Buy/Rent), Resources
- Server
    - Basics : RR, PORT
    - Latency
    - Throughput
    - Scaling
    - Cloud Systems
    - Distributed Systems

ip+port = full address



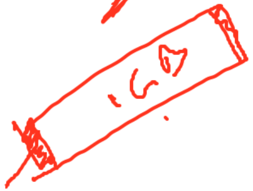
Latency

Higher or Lower Latency?

MB

KB ↑ ↓

~~1GB~~

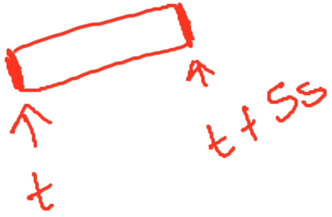


Filter:

P: L → H

B: Nike.

S: Relevance



10 M

$$1M \rightarrow 10\Sigma = \frac{10M\Sigma}{5M\Sigma}$$

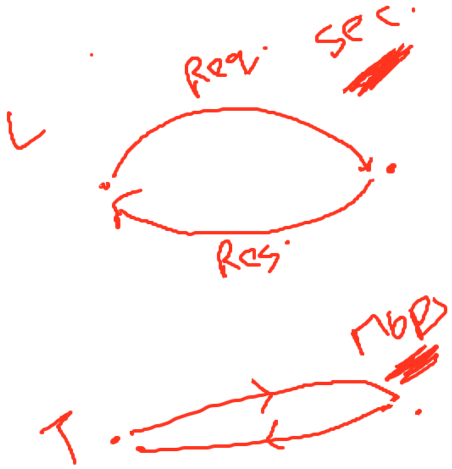
↳

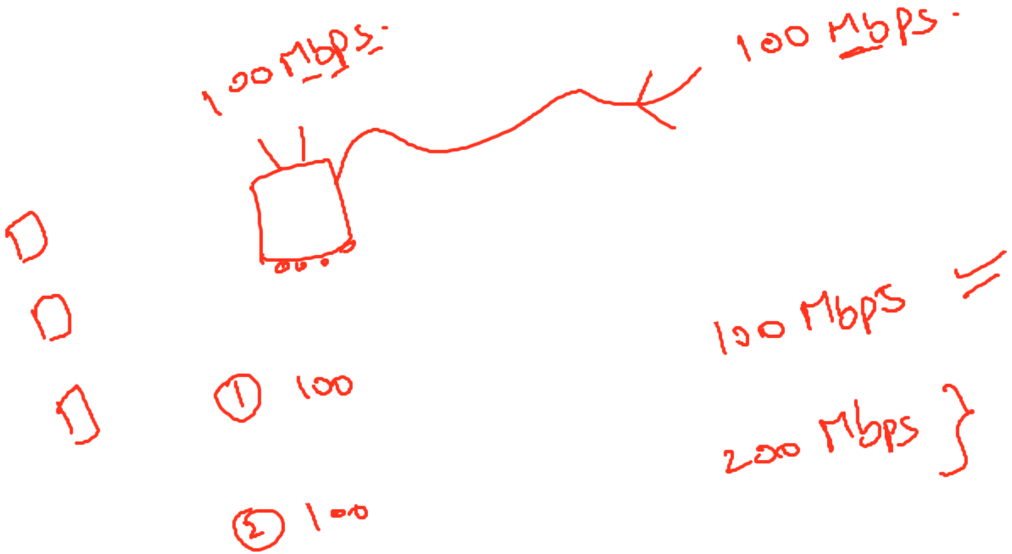
Link → JS

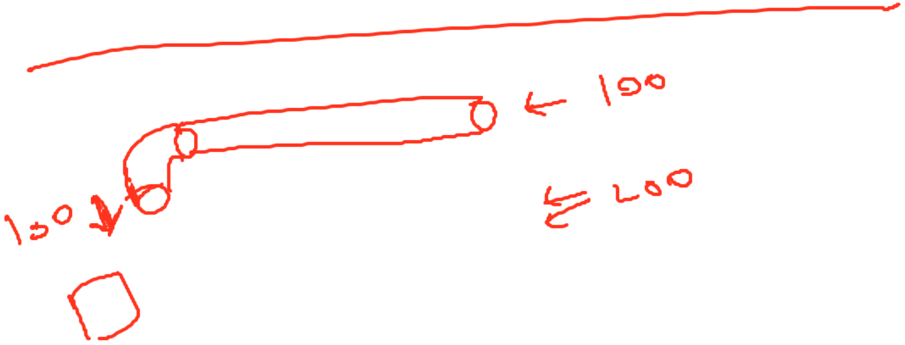
→ TS

Throughput

-


L
Req.
sec.
Res.
T
Mbps


100 Mbps.
100 Mbps.
100 Mbps
200 Mbps
① 100
② 100


100
200
100

10k → x6 → 20k → x6 → 40k → x6 → 80k → x6 → 1.6L

11 → Startup → Product 10k

↓

1k 100

1L

F B Business

↓ Own | Rent?

Custom

Privacy

Specials

SQL

mongo

Server

=

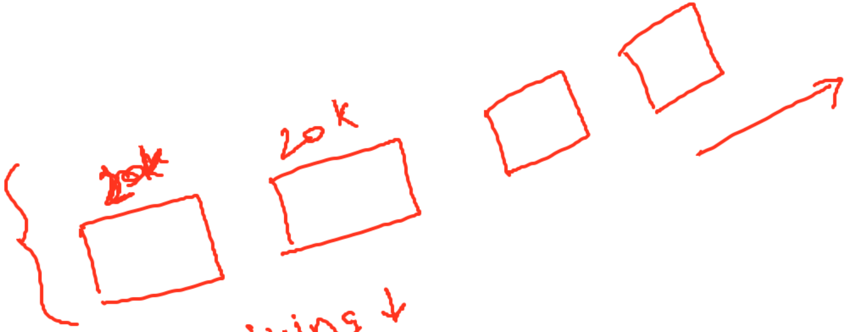
AWS → A

Azure → M

GCP → Google

Heroku

etc



issue: Scaling Billing ↓

{ Elect Load

6KW } Peak demand

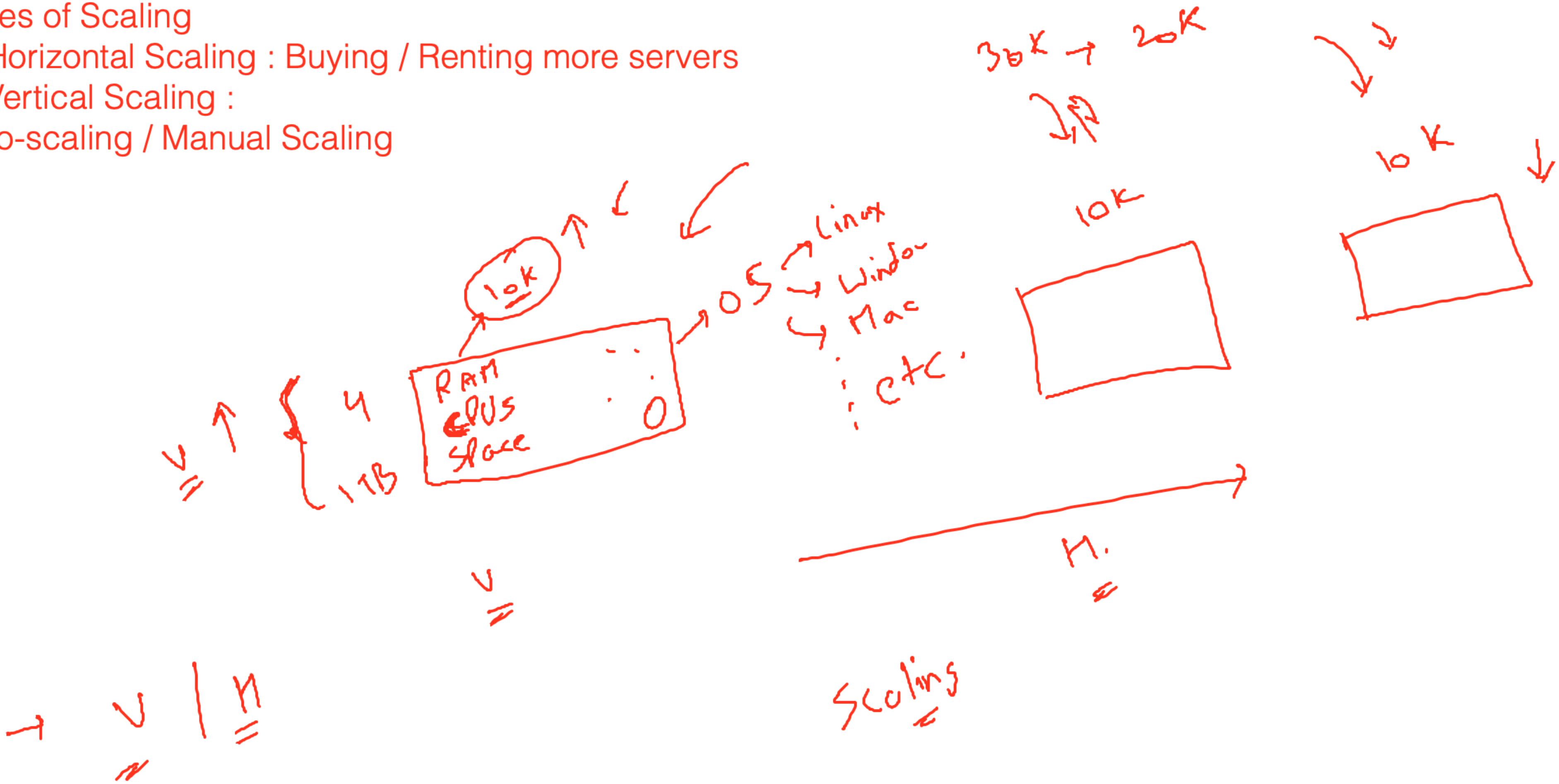
6KW

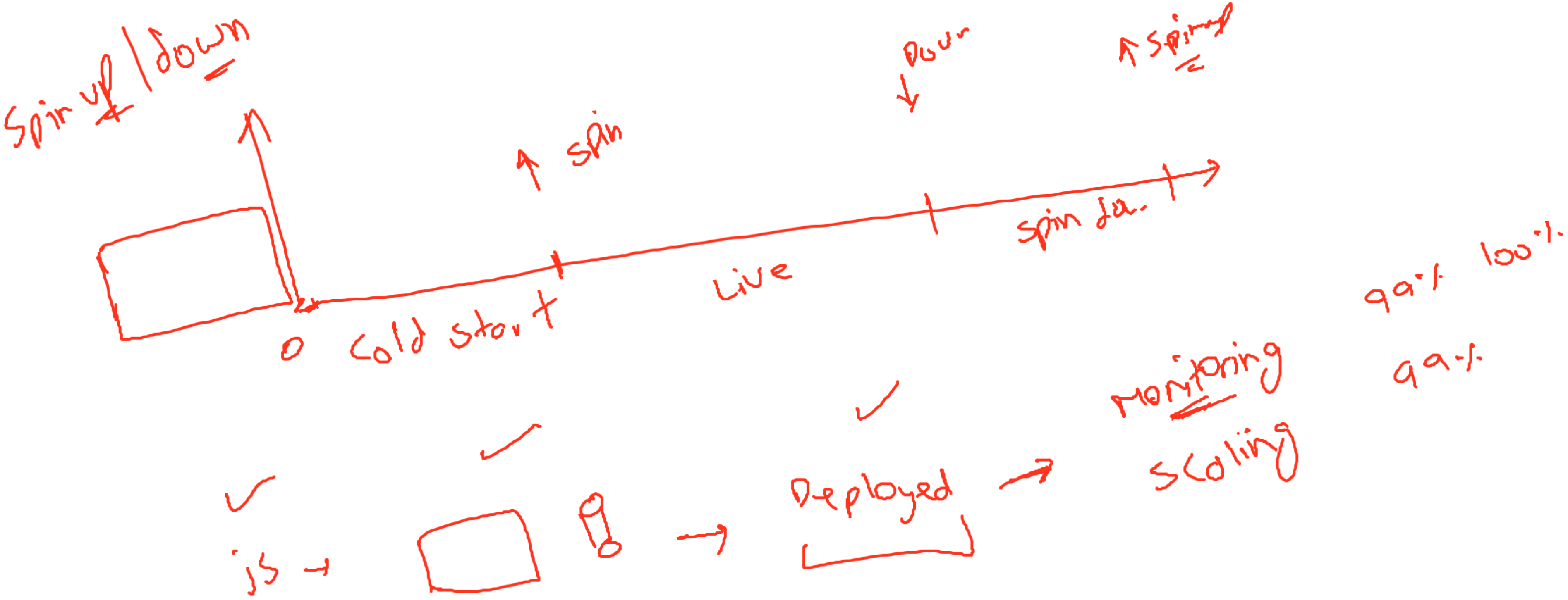
3KW

300

Scalling : Handling the traffic

- issue : unpredictable traffic
- solution : Scaling is the solution : Increase/ Decrease the capacity handling of server
- Types of Scaling
    - Horizontal Scaling : Buying / Renting more servers
    - Vertical Scaling :
- Auto-scaling / Manual Scaling





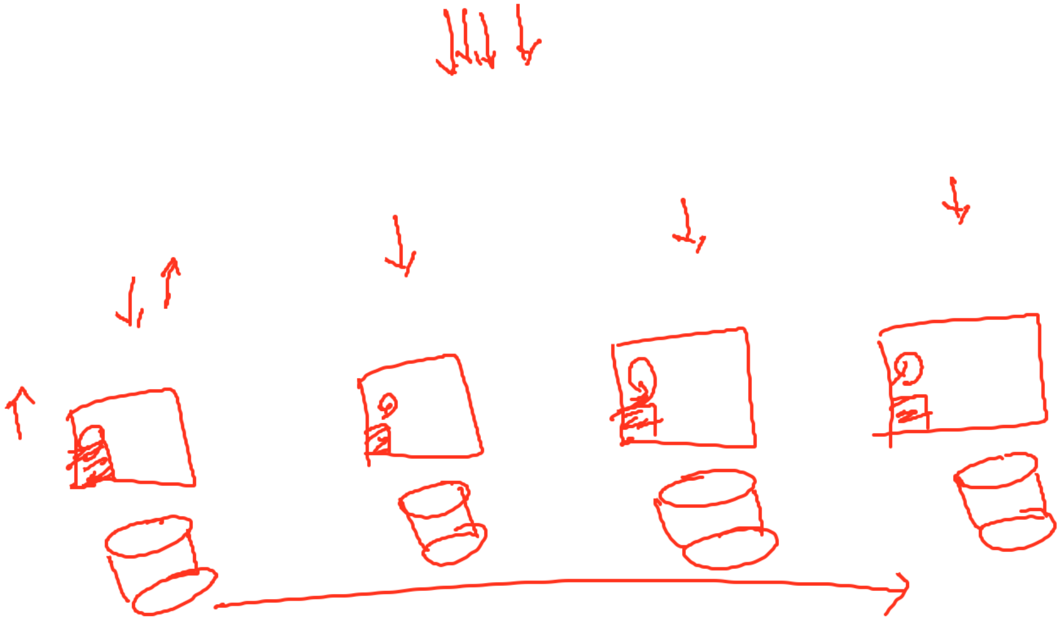
- Service is up and running : You are live : Ready to serve the req.
- Spin up
- Render : Spin down policy : instance traffic for 15 min : Spin down -> Spin up : solution?? cron job : 14 min
- free con job : URL, 14 min

synchron

Cloud Computing
- Cloud : remote
- Computing : server
- Cloud services : drives, computing

Distributed System
- Multiple servers (nodes)

Servers(node) => nodeJS



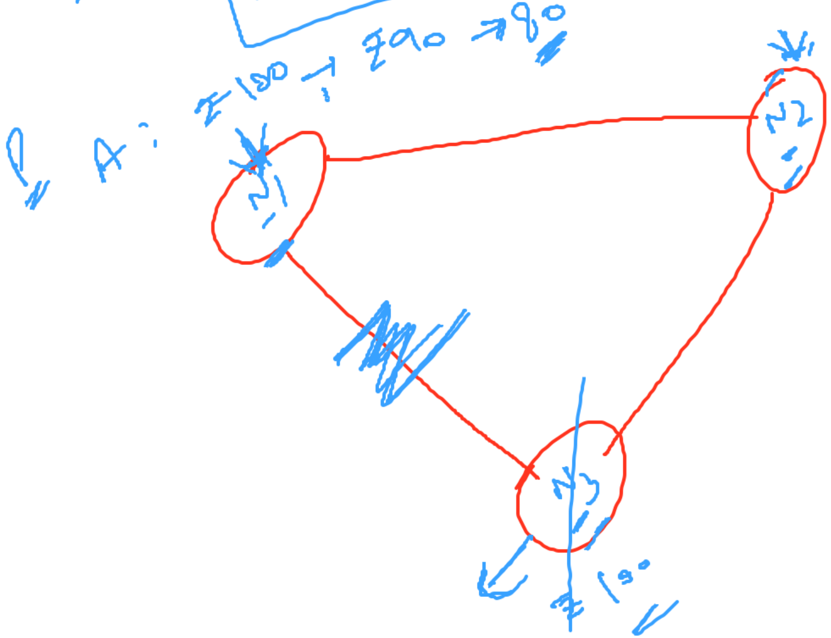
CAP Theorem (related to Servers/Nodes/ Distributed Systems)

Consistency : Servers, DB

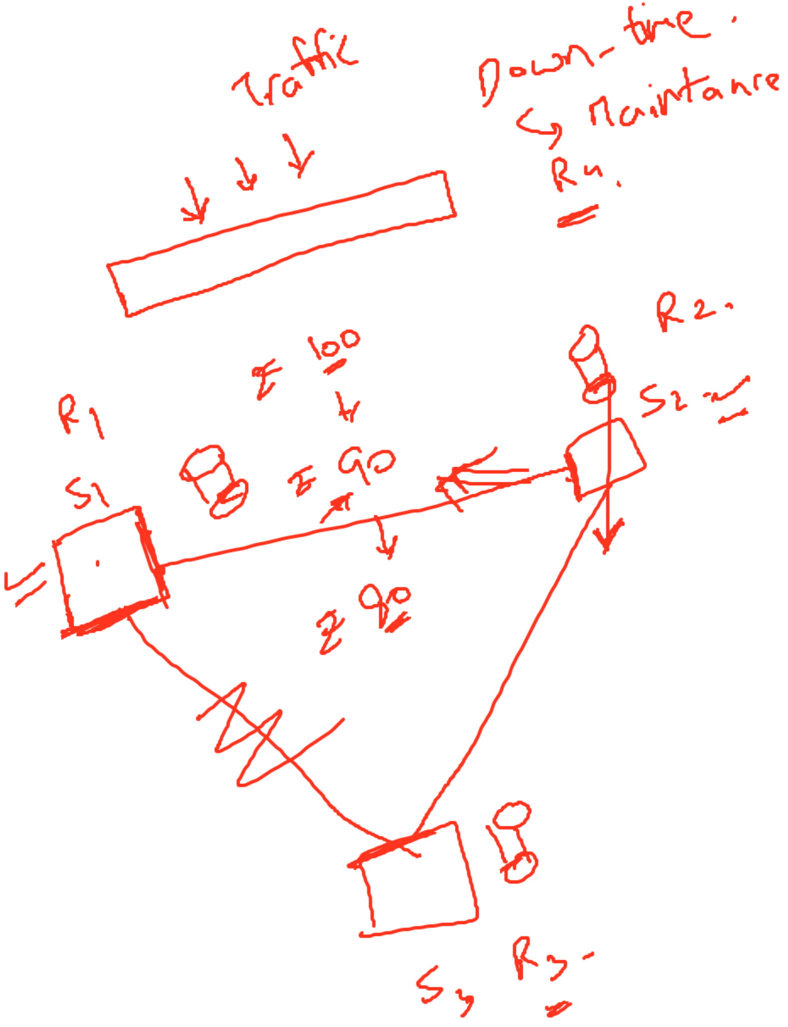
Availability :

Partition-Tolerance :

⇒ CA | CP | AP



Consistent X



System: tolerate delays | loss