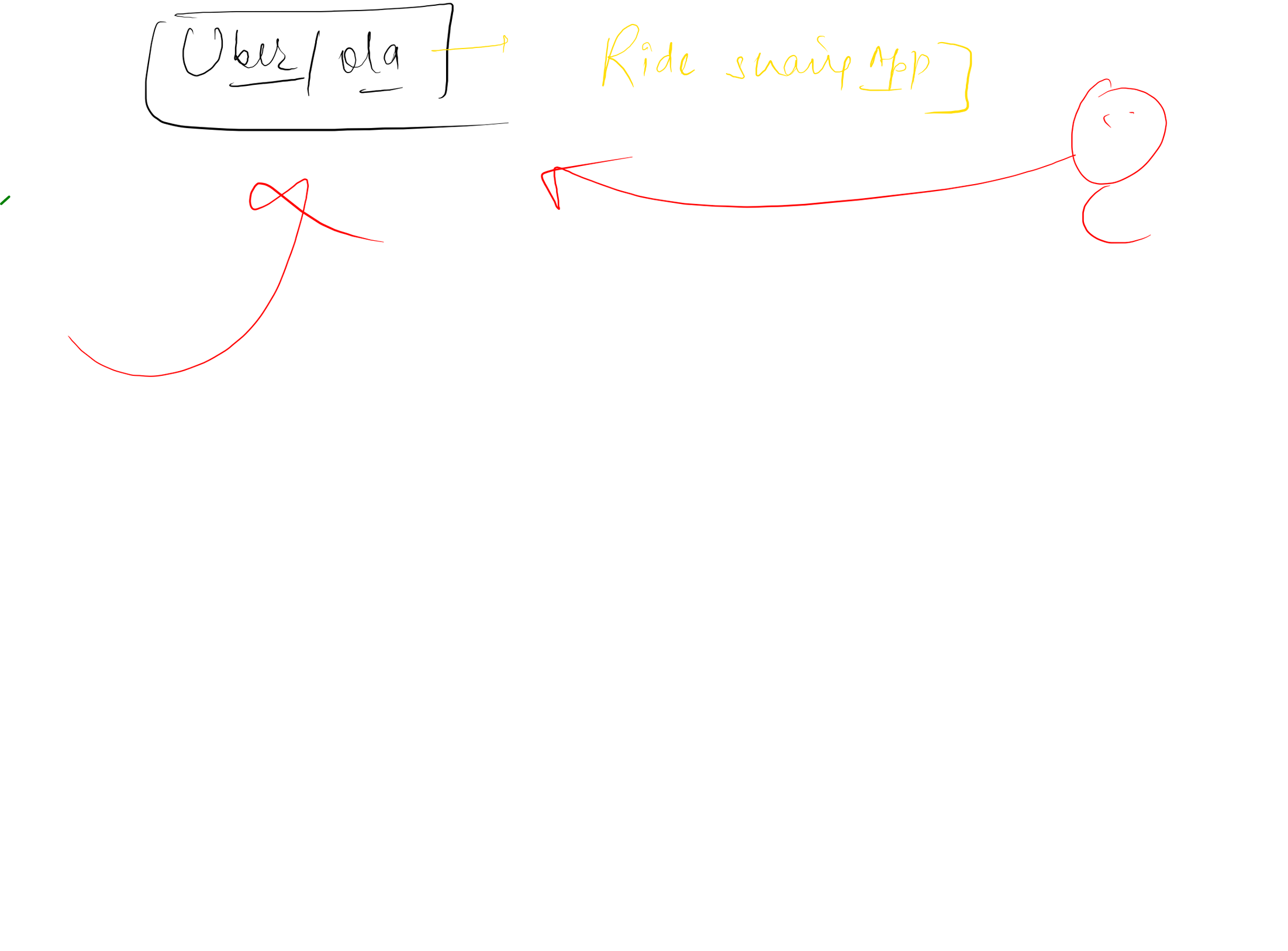
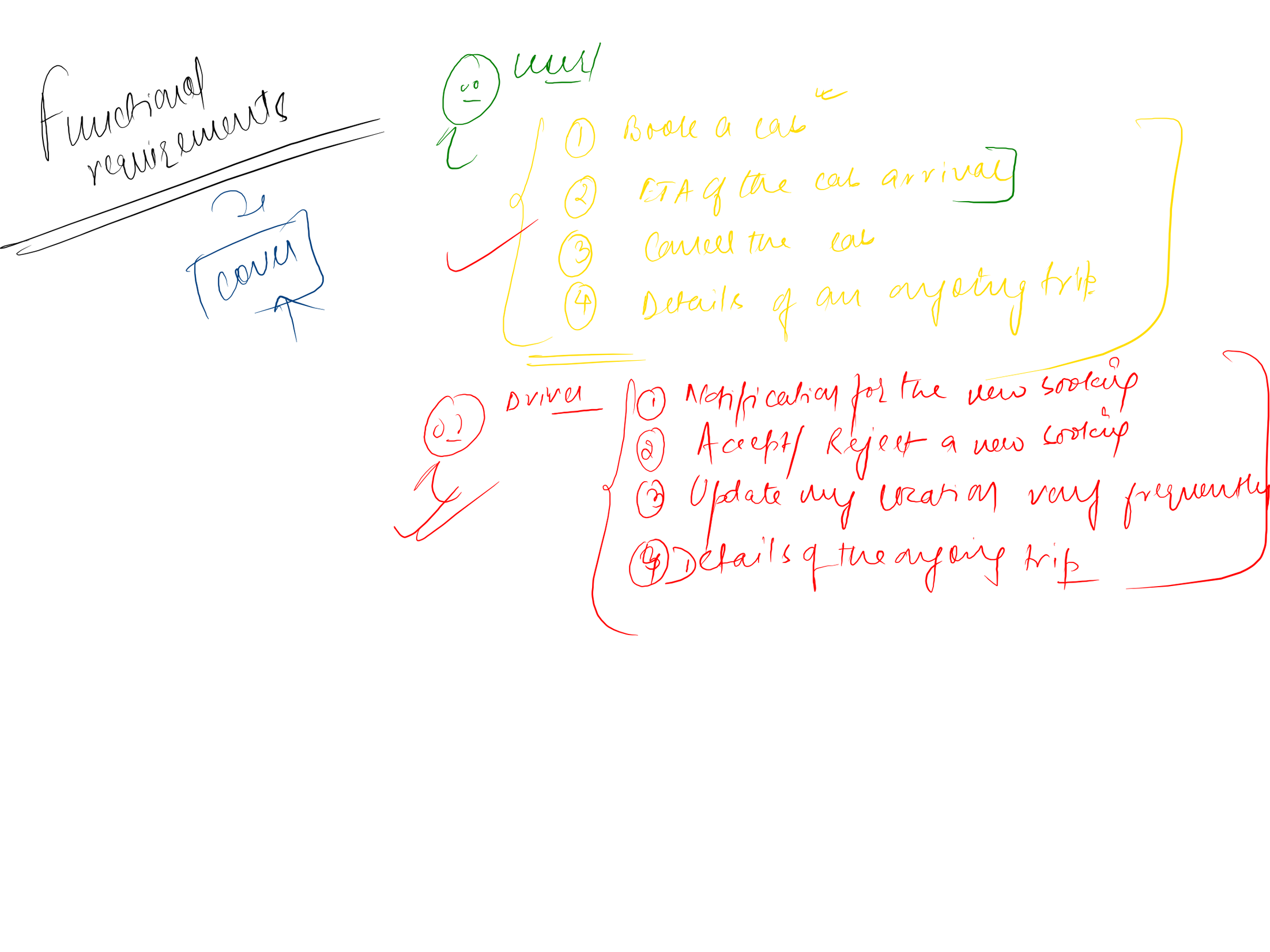

Uber / ola
Ride sharing App

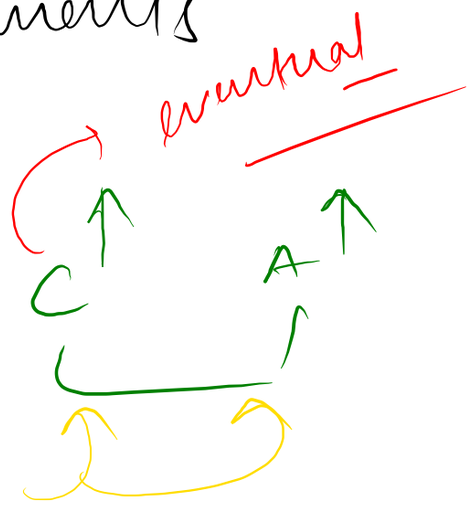
Functional requirements

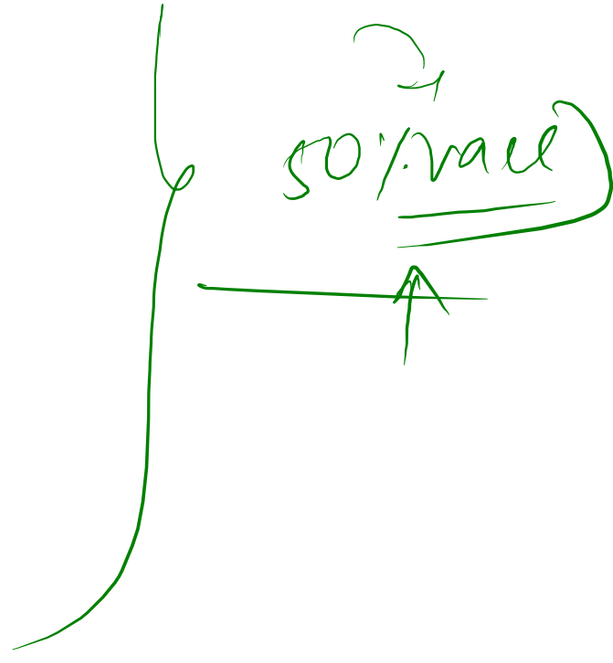


Non-functional requirements

① Real time

② CAP





③ Scalable → parallelism

④ Reliable → probability of failure

Next steps :-

Estimations

- compute
- Storage
- N/w bandwidth

Make the infra ready

Total customers/riders = 500 M

Total No of drivers = 5 Million

Daily Active

Riders = 20 Million → Trips

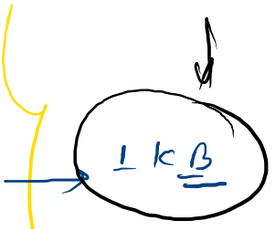
Drivers = 3 Million

Storage

Rider

Rider's data

id
name
age



↳ total storage needed for all the riders = 500 M × 1 KB

= 500 × 10^6 × 1 KB

= 500 GB

Daily, new riders are increasing at the rate of 50,000/days

50,000 × 1 KB → 50 MB

Storage requirement 50 MB/day

Next 1 year

current → 500 GB

365 days = 365 × 100 MB

= 182.5 × 100 MB / 2

= 18.2 GB

500 GB + 18.2 GB

= 518.2 GB

total storage needed now = $3 \times 10^6 \times 1KB$

= 3GB

total storage @ in next 365 days = 5MB x 365 + 3GB

= 1825 MB + 3GB

= 1.8 GB + 3GB

= 4.8 GB

5000 new drives

= 5000 x 1KB

= 5000KB

= 5MB

Trips data storage

in 1 day = 20 million trips

1 trip data = 100 Bytes

1 day storage = $20 \times 10^6 \times 100$ Bytes

$= 2 \times 10^9 \times$ Bytes

$=$ 2 GB

365 days → 365×2 GB

$=$ 730 GB

Trip → Resulting in queries / API calls

24 hrs = 20 Million

$= 20 \times 10^6$

1 seconds $= \frac{20 \times 10^6}{24 \times 60 \times 60} = 230$ trips/seconds

100 Bytes

230×100 bytes/seconds

$= 23$ KB/seconds

Driver → Needs to update it's location very frequently

36 Bytes

Updates the location every 4 seconds

3 million drivers

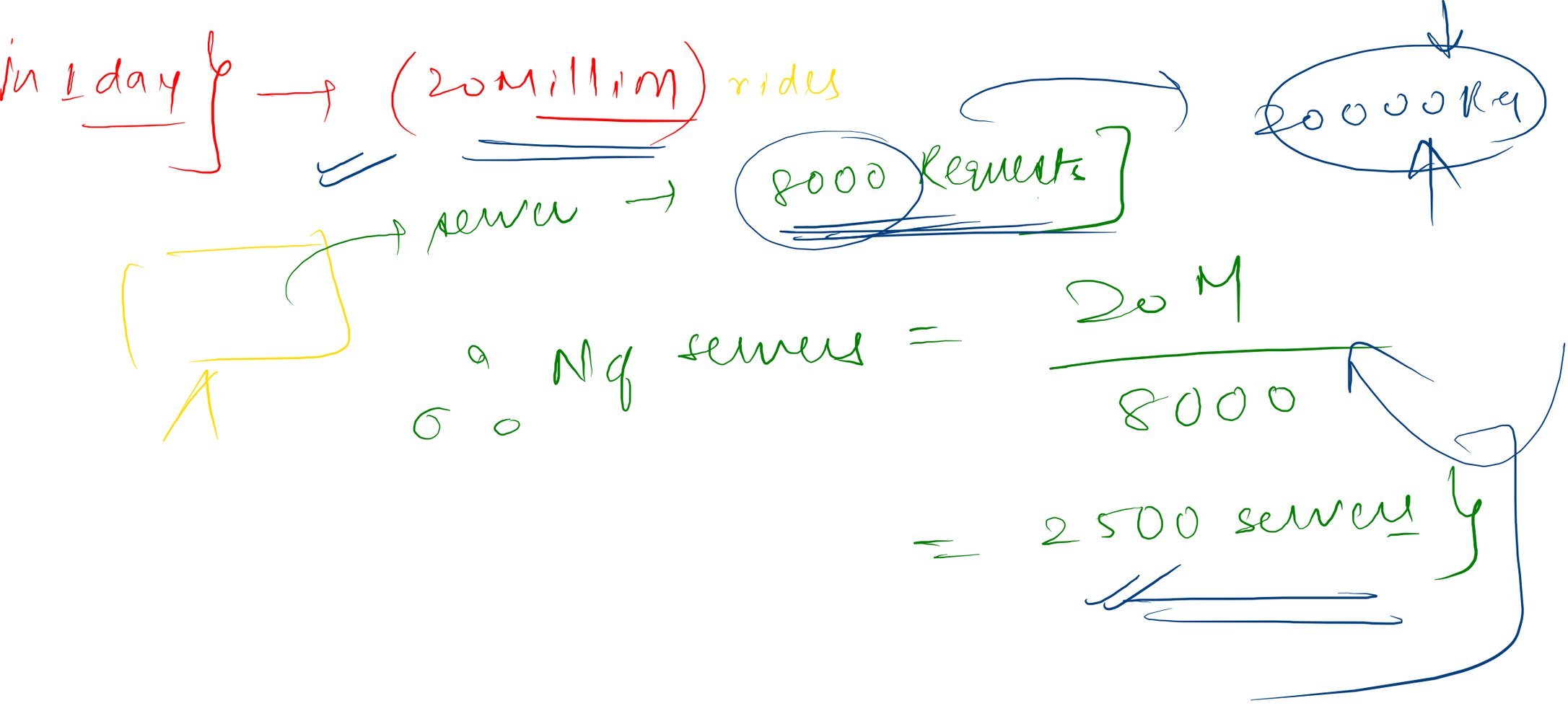
$= 36 \times 3 \times 10^6$ Bytes

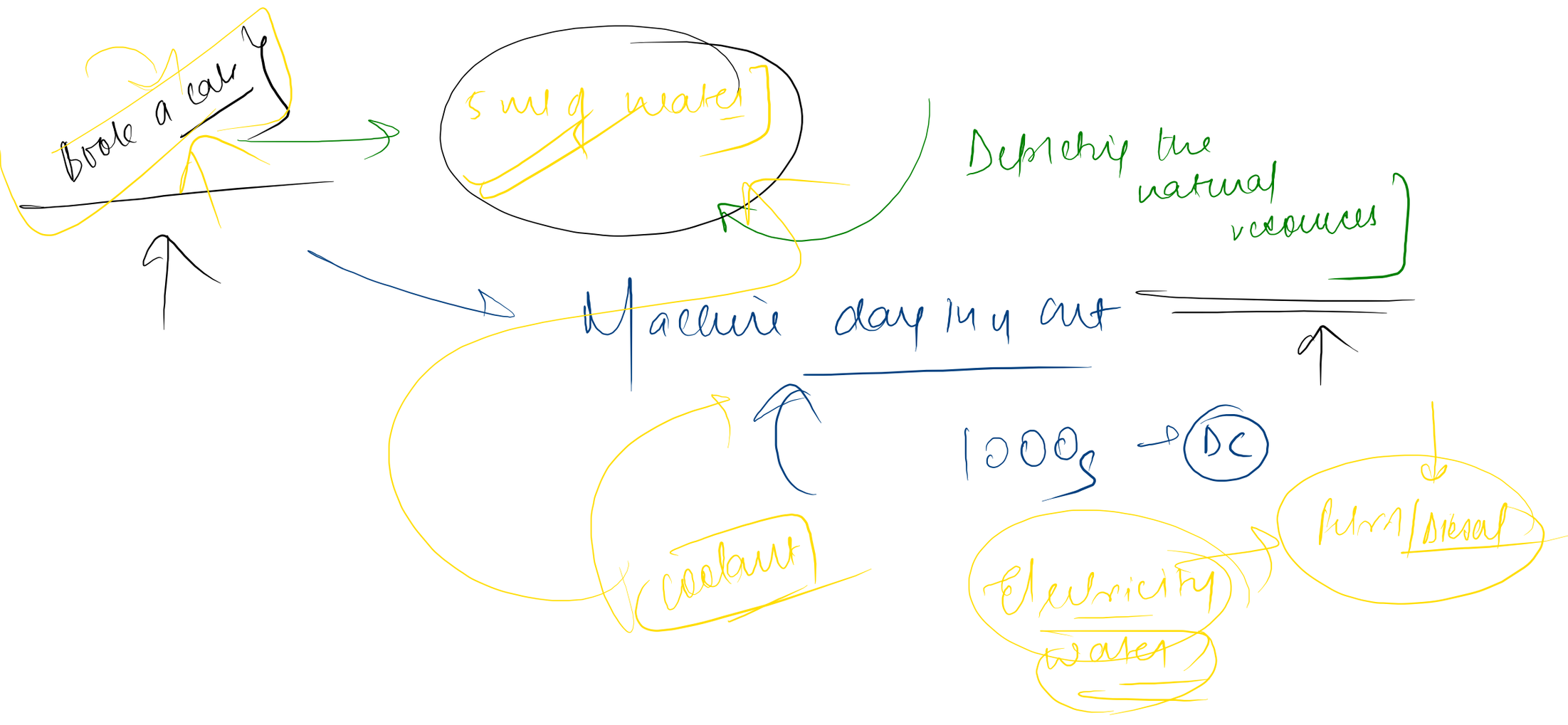
$= 108 \times 10^6$ Bytes

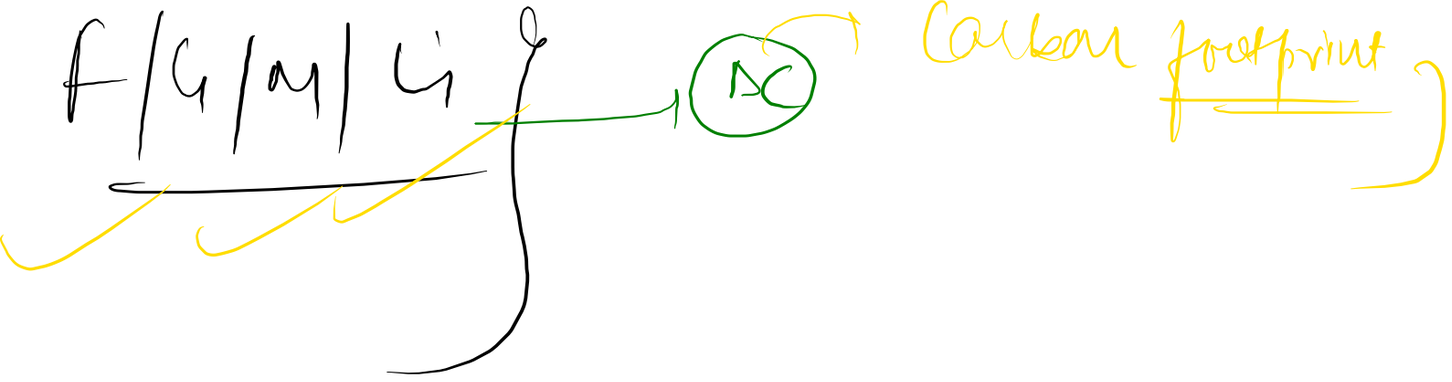
$= 108$ MB

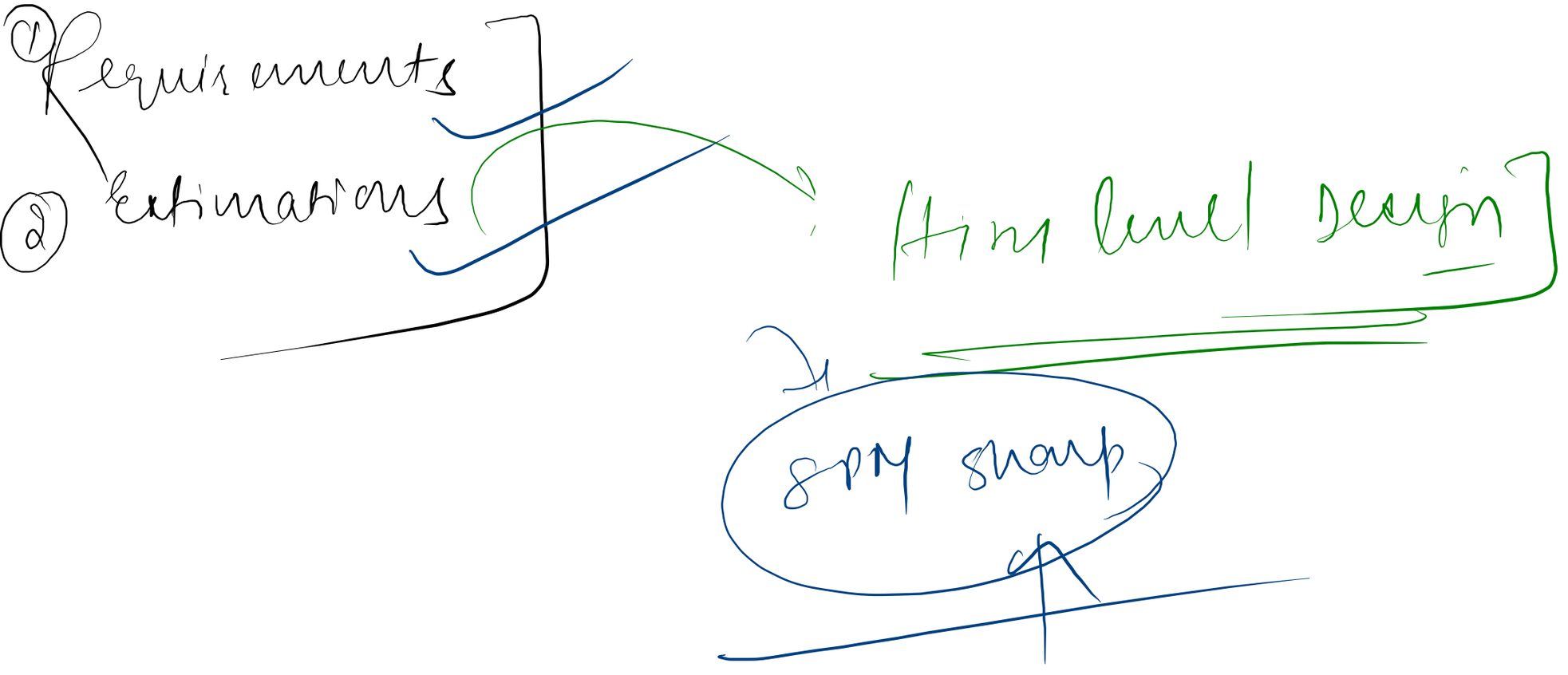
Throughput = 108 MB / 4 second

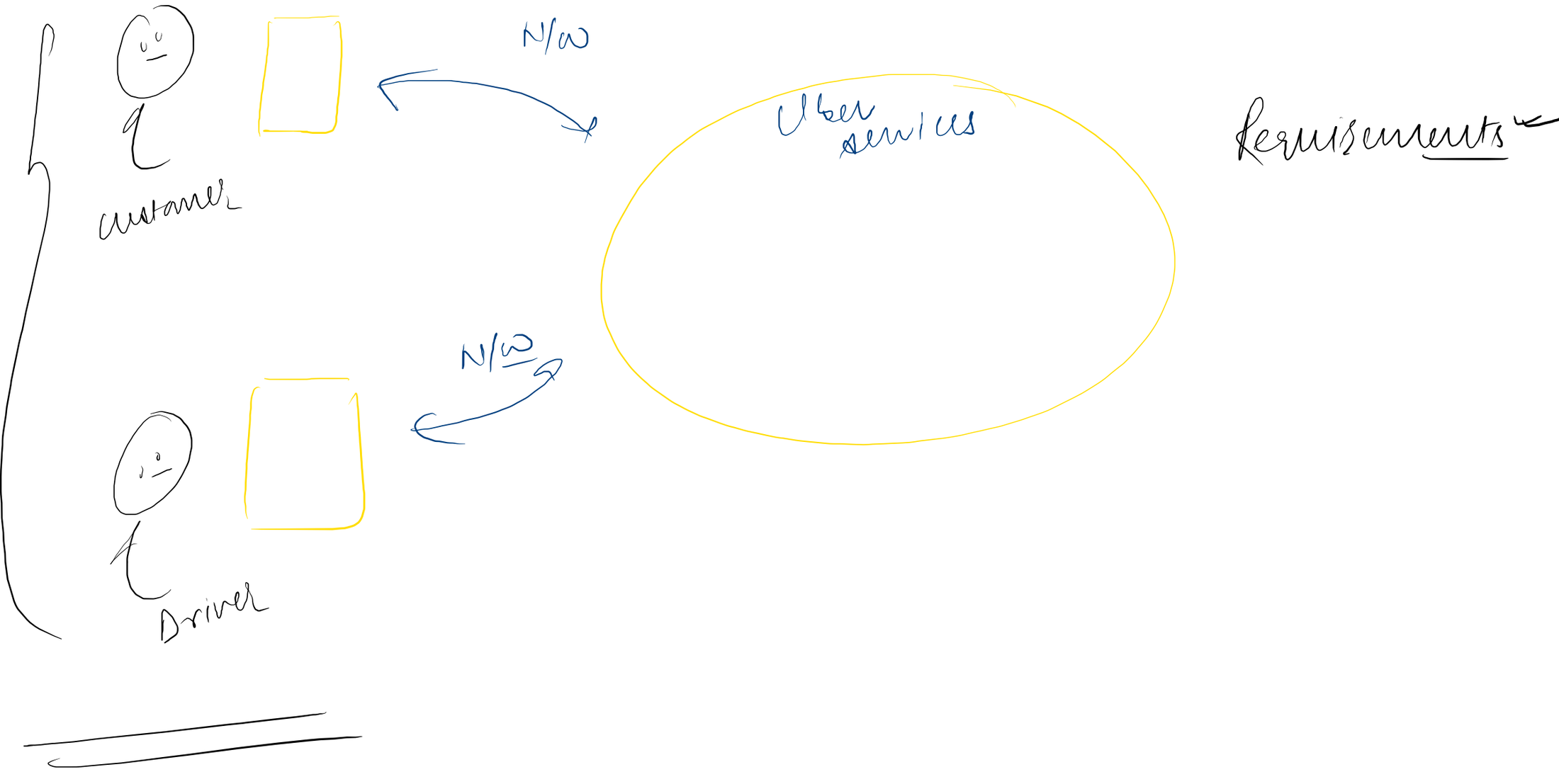
= 24.5 MB/second


in 1 day
20 Million rides
8000 Requests
server
20000 Req
No. of servers = 20 M / 8000
= 2500 servers

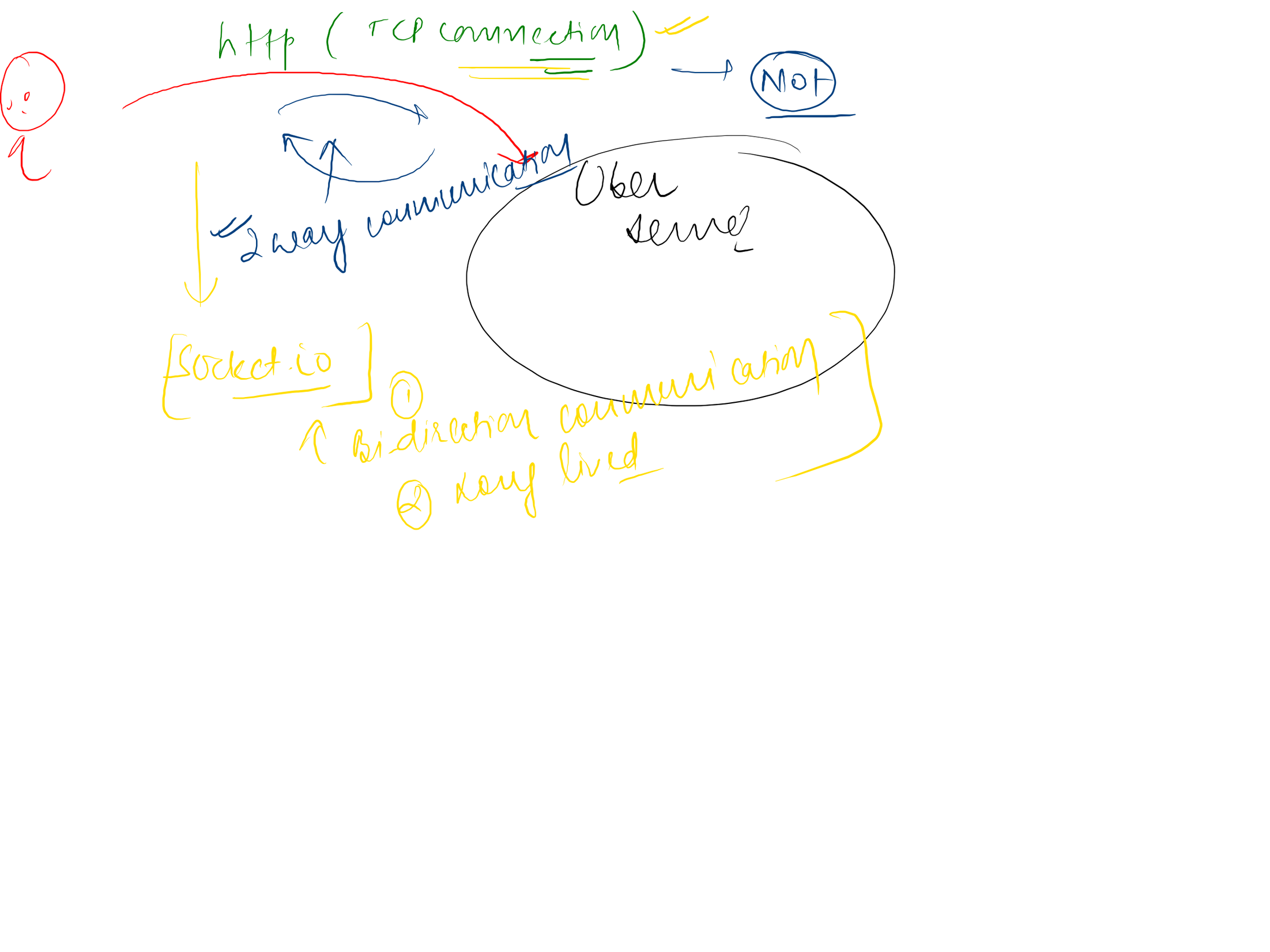

Book a cab
5 ml of water
Depleting the natural resources
Machine doing its cut
1000s
DC
coolant
Electricity
water
Petrol/Diesel

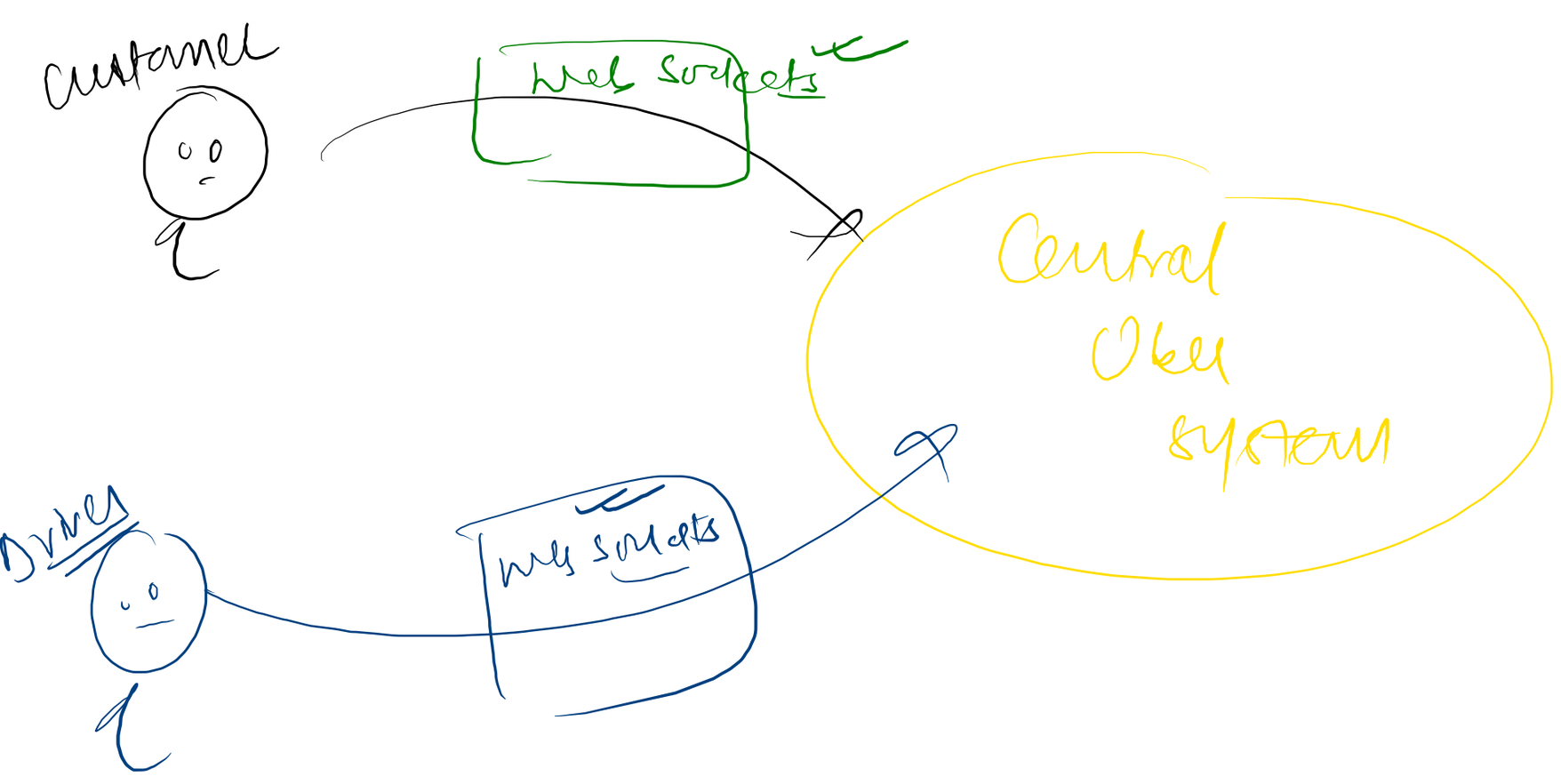

F
G
M
Li
DC
Carbon footprint

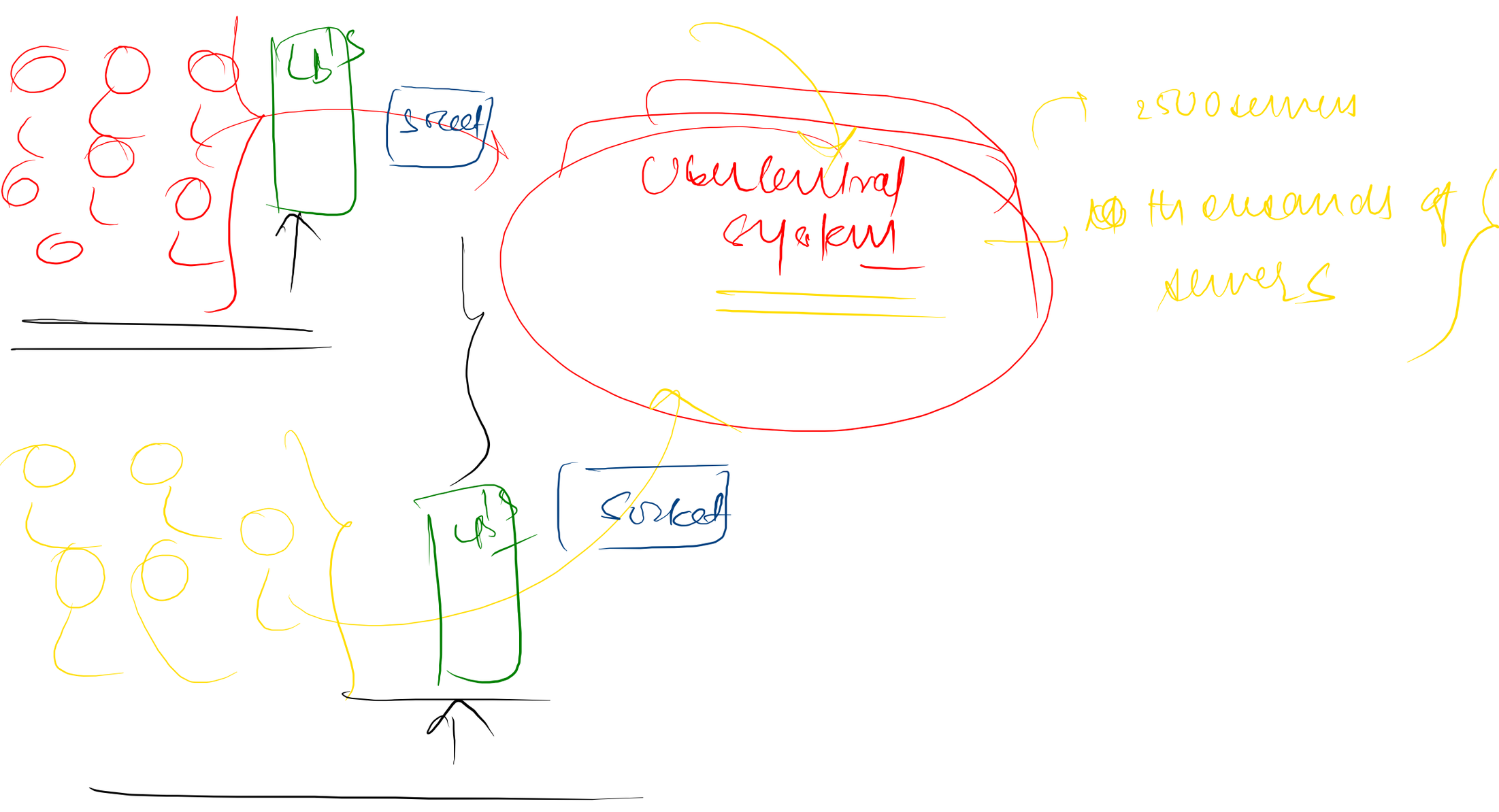

1 Requirements
2 Estimations
High level Design
SPM snap

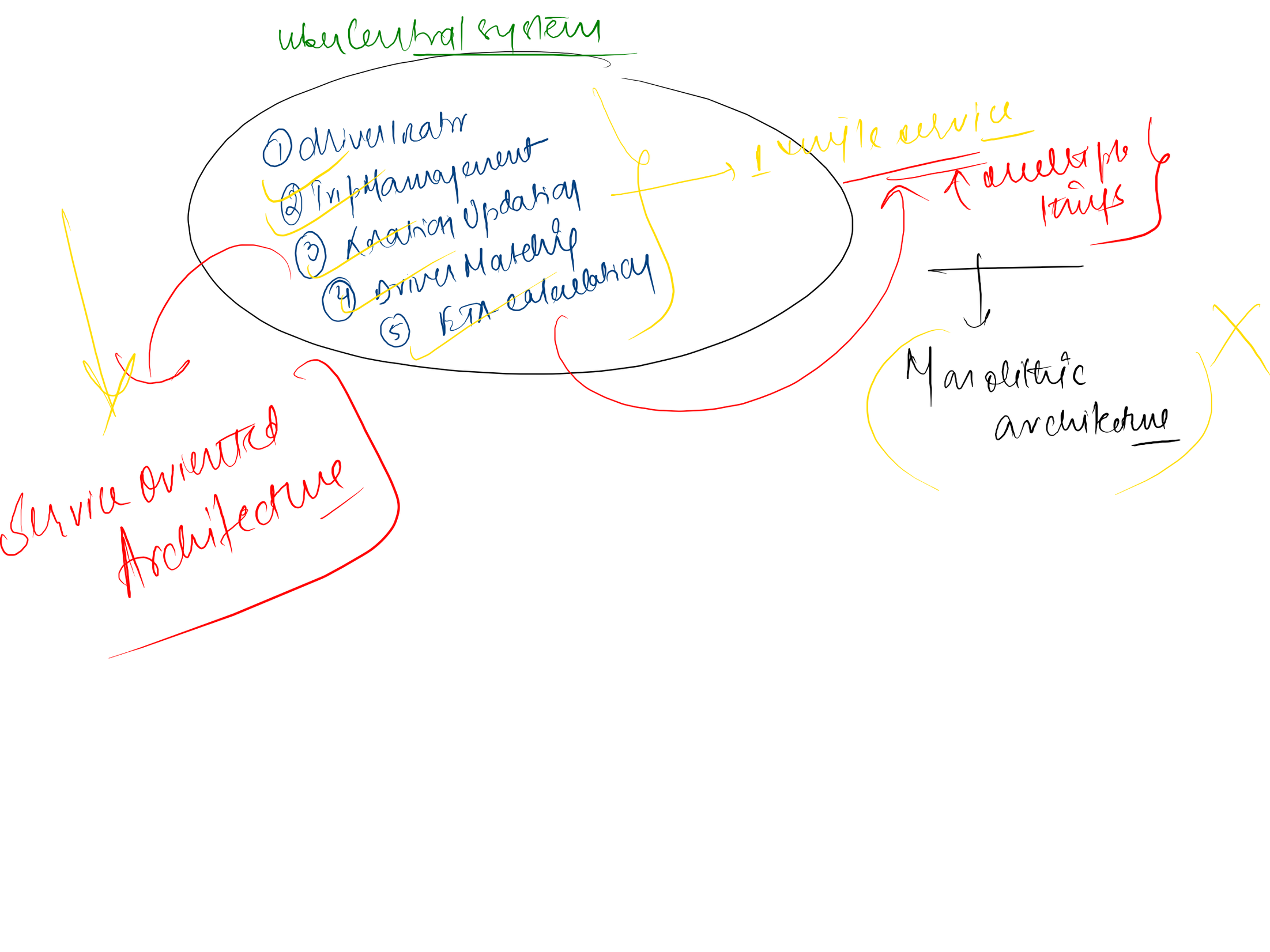


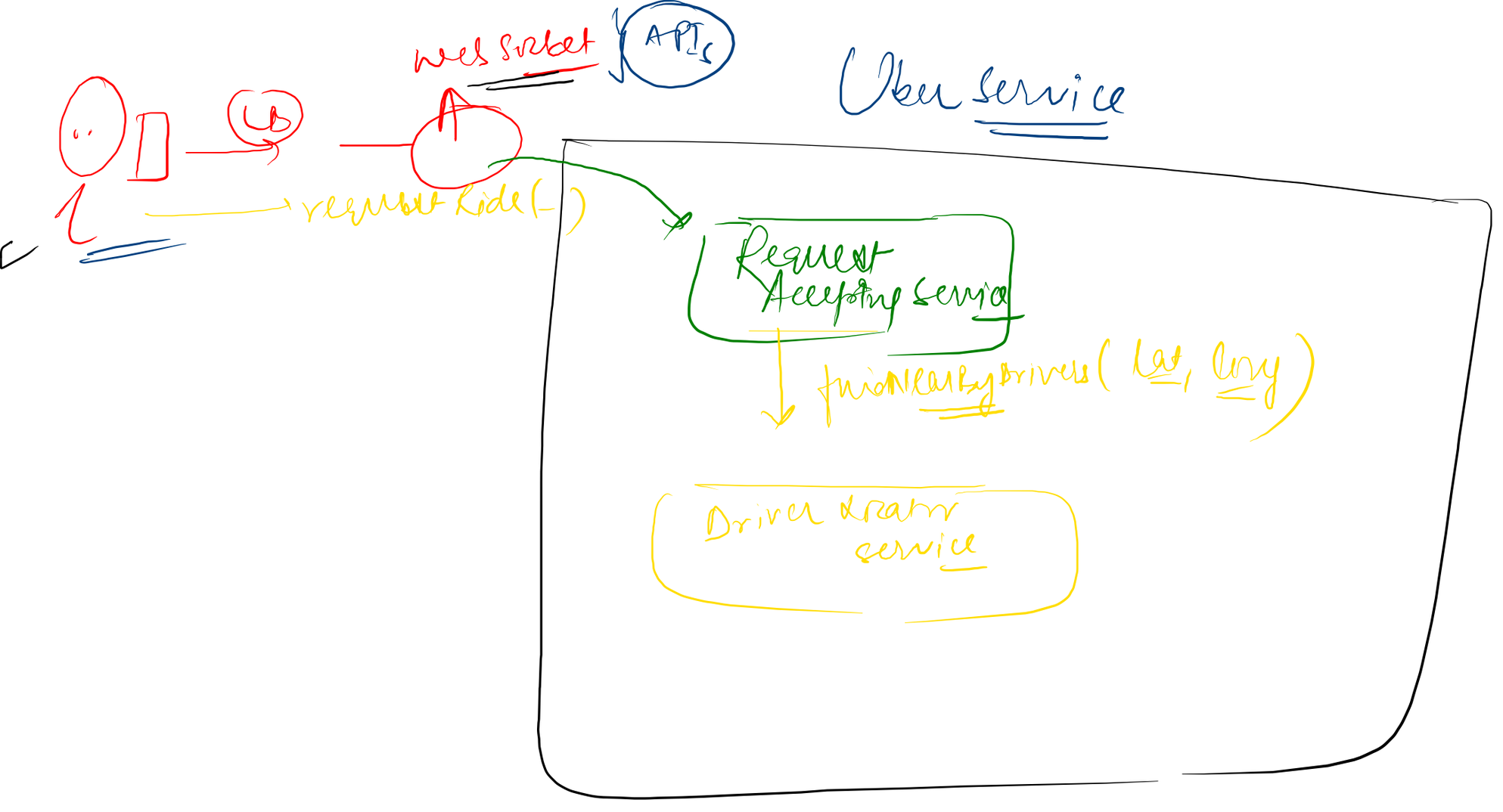
Requirements

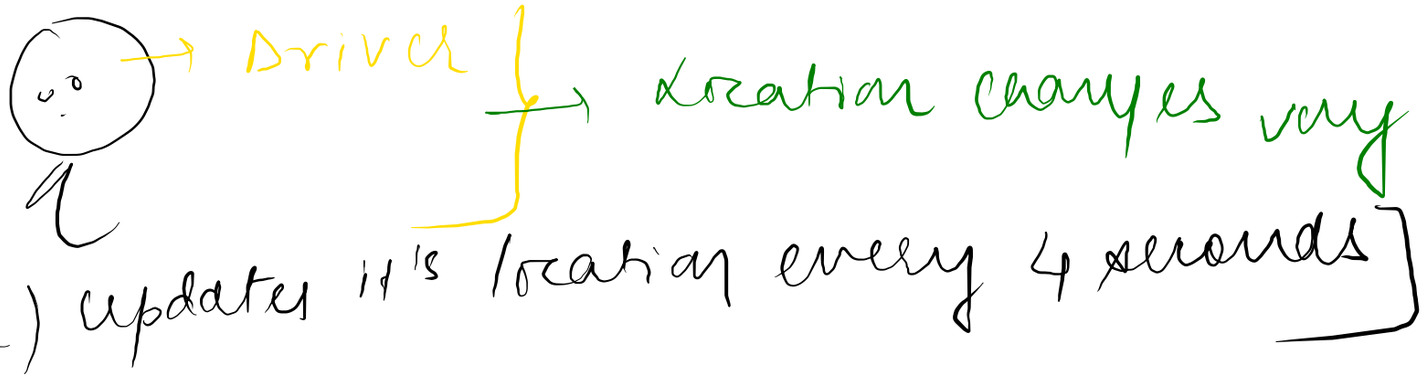

http (TCP connection)
NOT
2 way communication
Uber servers
Socket.io
① Bi direction communication
② long lived


customer
Web Sockets
Central
Uber
System
Drivers
Web Sockets

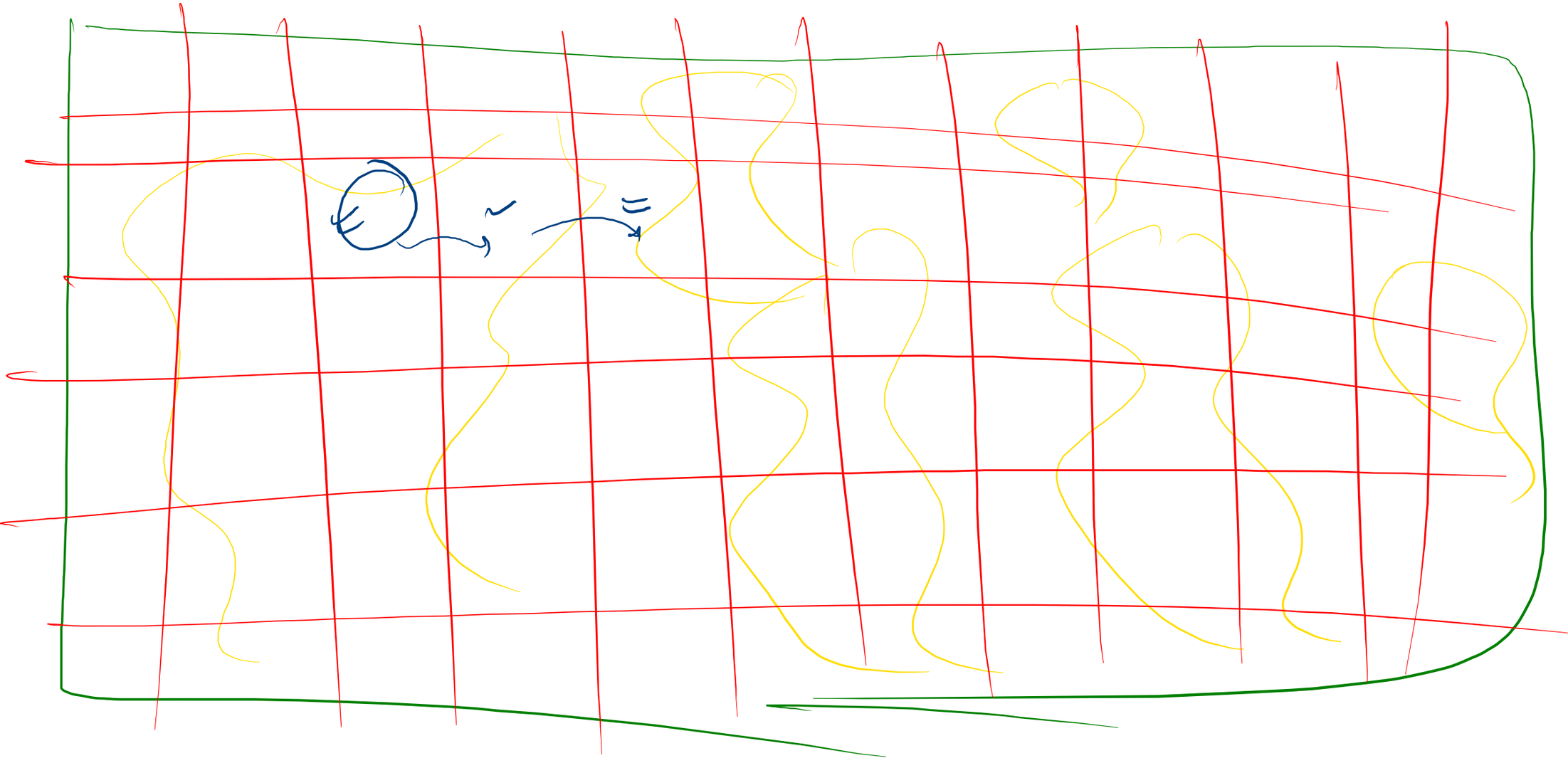

Distributed system
2500 servers
thousands of servers

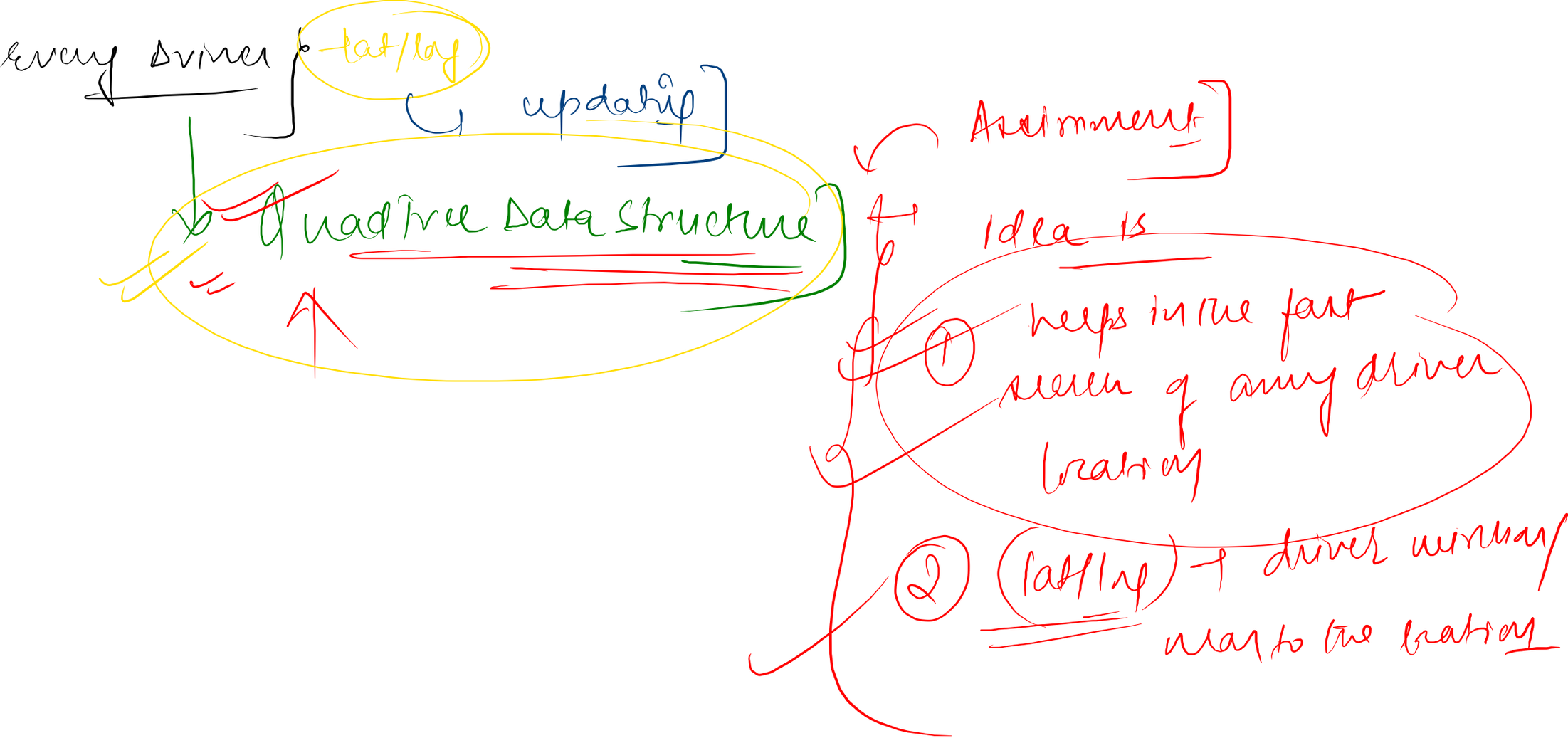

Uber Central system
1. Driver locator
2. Trip Management
3. Location Updating
4. Driver Matching
5. ETA calculating
1 single service
Monolithic architecture
Service Oriented Architecture


web socket
APIs
Uber service
request Ride(..)
Request Accepting service
findNearbyDrivers(lat, long)
Driver Location service

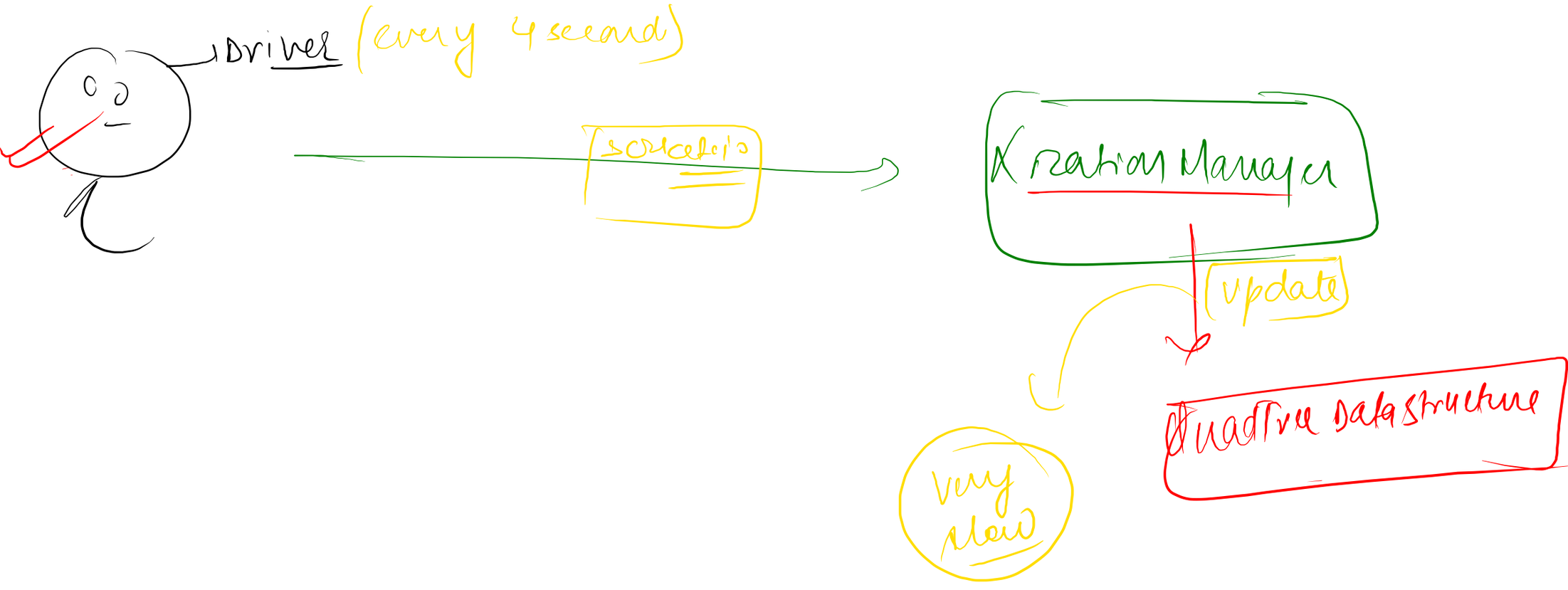


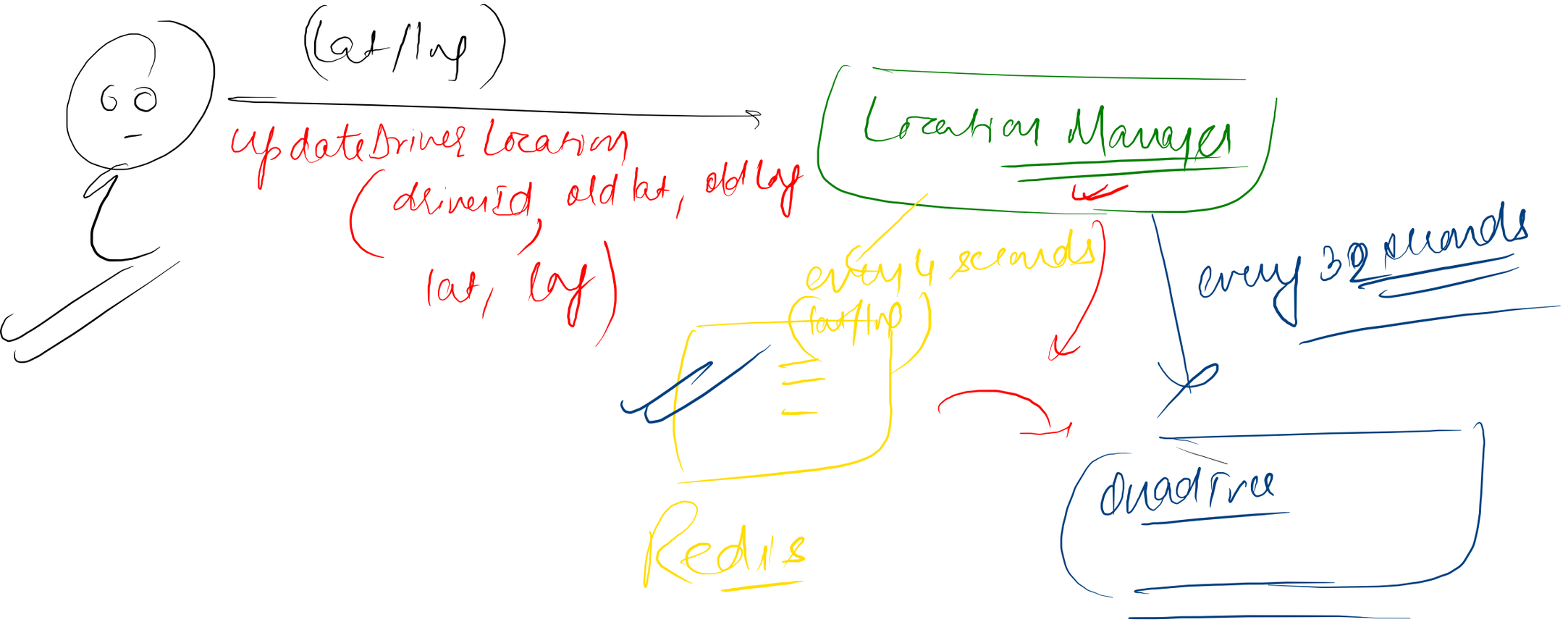
Entire globe

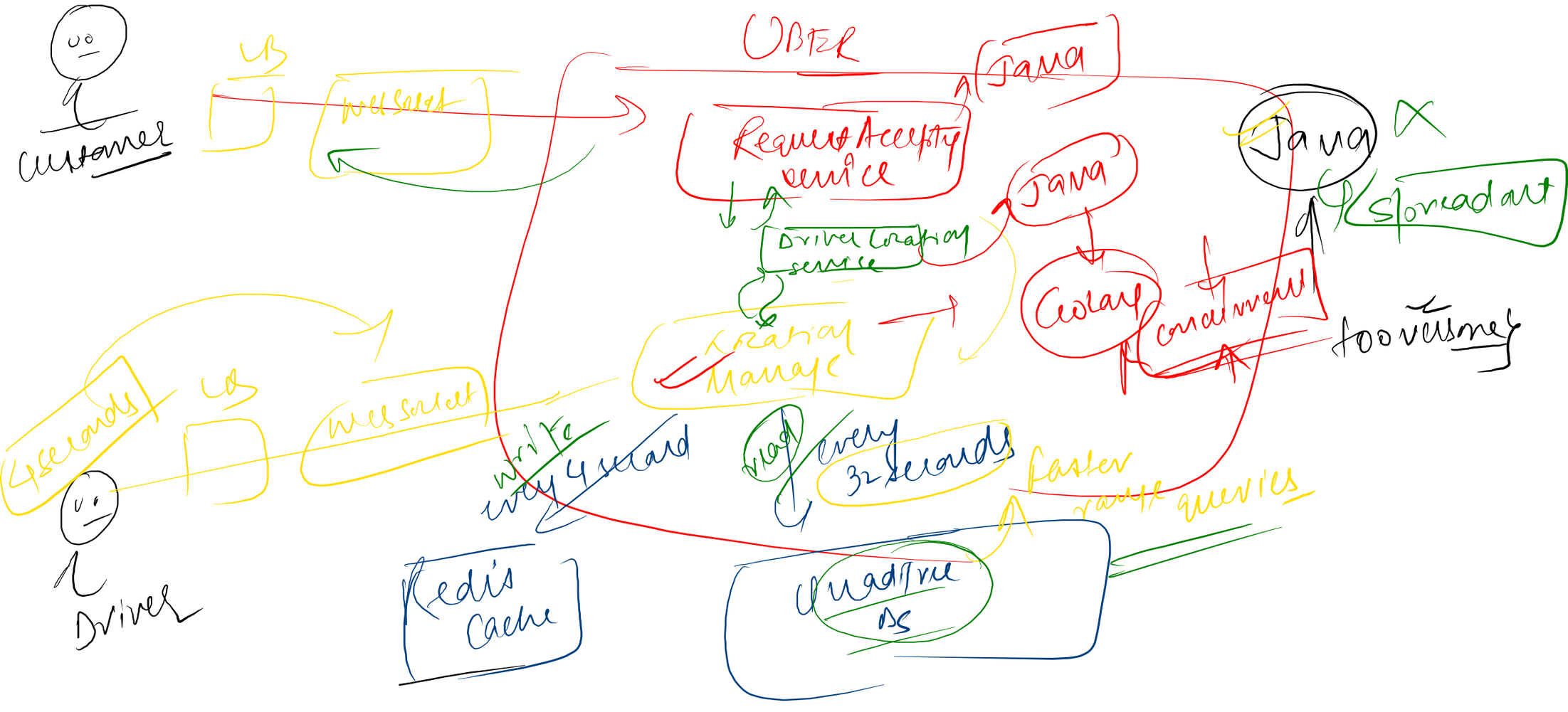


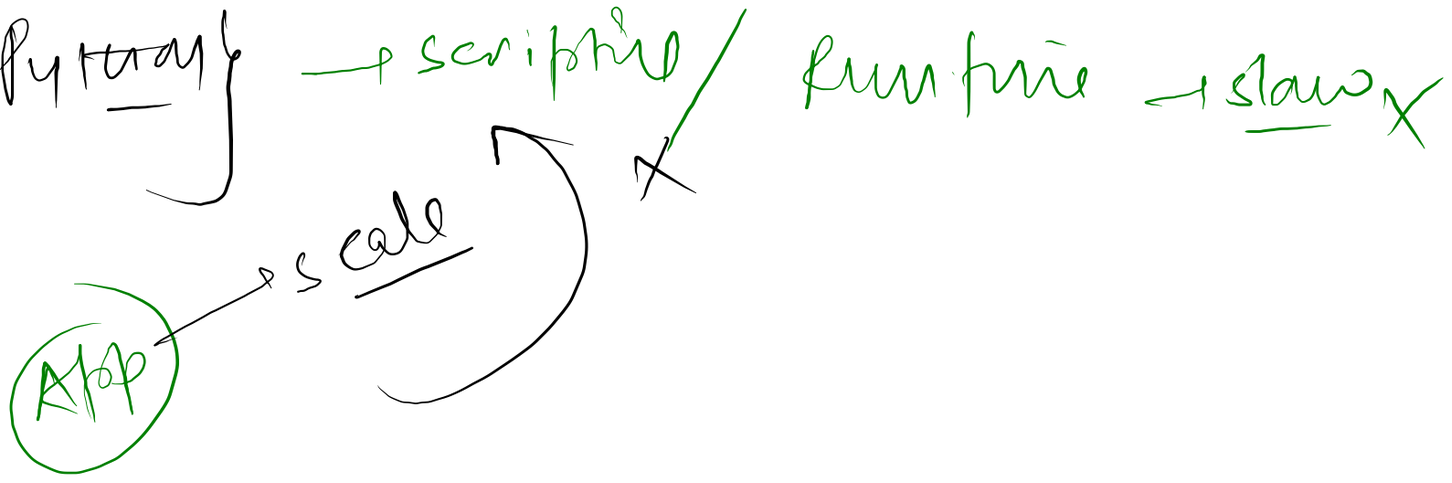

every driver
lat/lng
updating
Quadtree Data Structure
Assignment
Idea is
① helps in the fast search of any driver location
② (lat/lng) of driver coming/ near to the location


Zomato
Restaurants
lat/lng


Driver
(every 4 second)
socket's
Location Manager
update
Updating Data structure
very slow

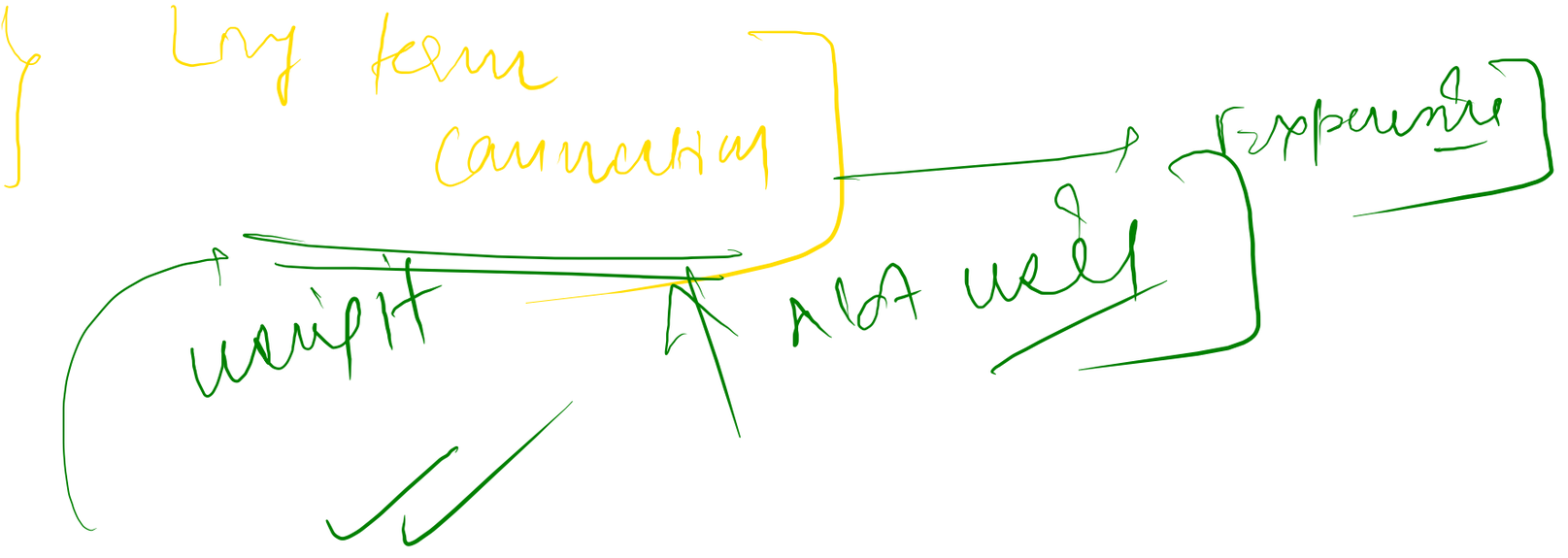

(lat/lng)
updateDriverLocation
(driverId, old lat, old lng
lat, lng)
Location Manager
every 4 seconds
(lat/lng)
every 30 seconds
Redis
QuadTree

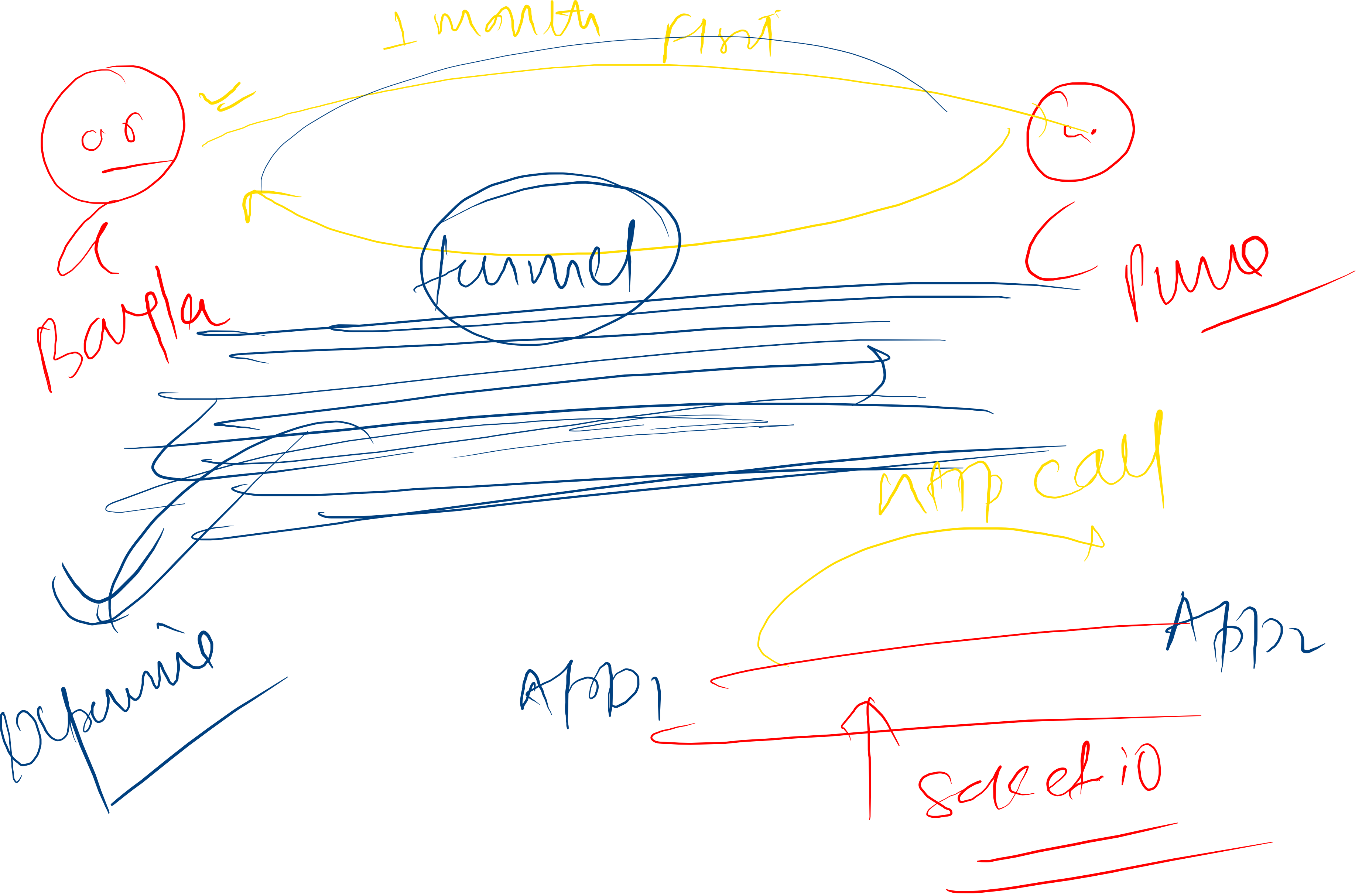

Customer
UBER
Redis Cache
Driver
every 4 seconds
every 4 second
every 30 seconds
faster range queries


Python
scripting
Runtime
slow
scale
App


Google Maps
1 Satellite
2 Map of other users
3 installed the sensing units
ML
Graph database
Spanner

Web Sockets



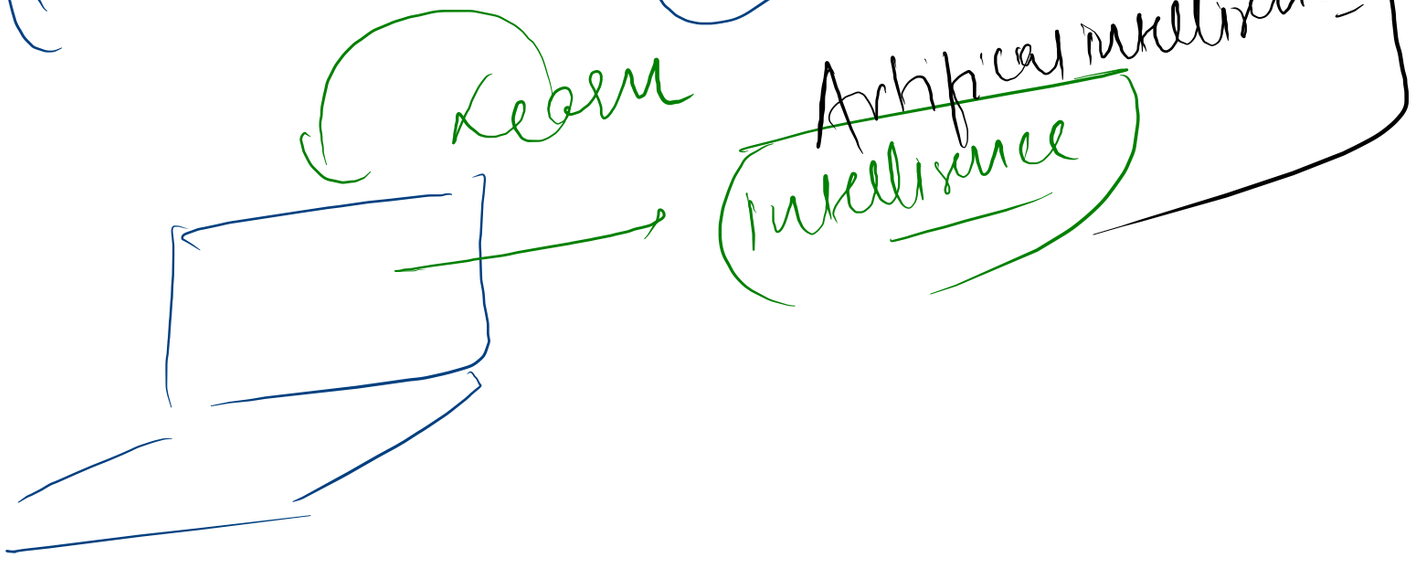

Pune
funnel
http call
App1
App2
socket io

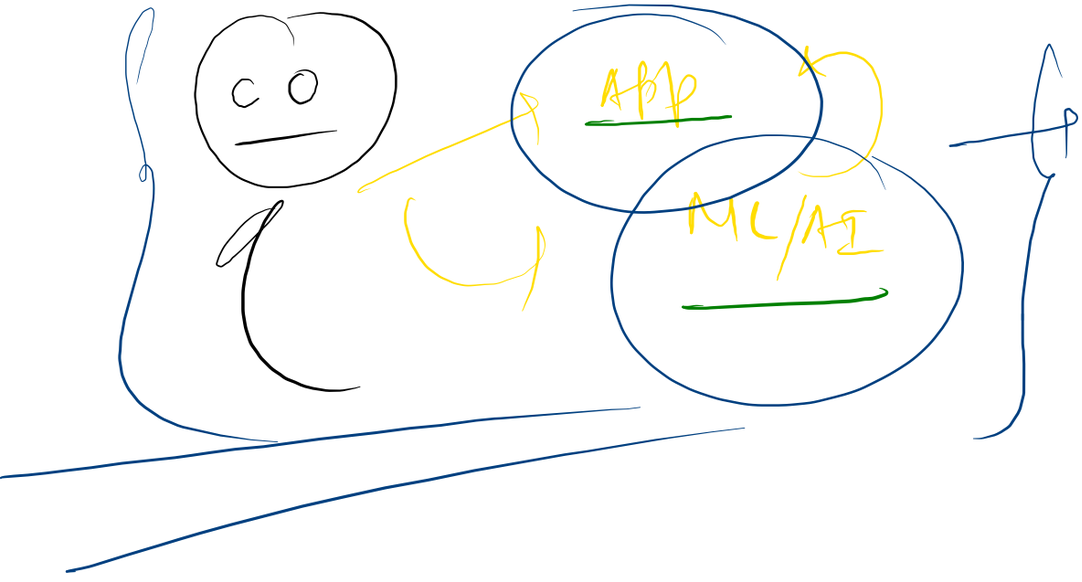
Socket → 1 way two connection → expensive

AI/ML

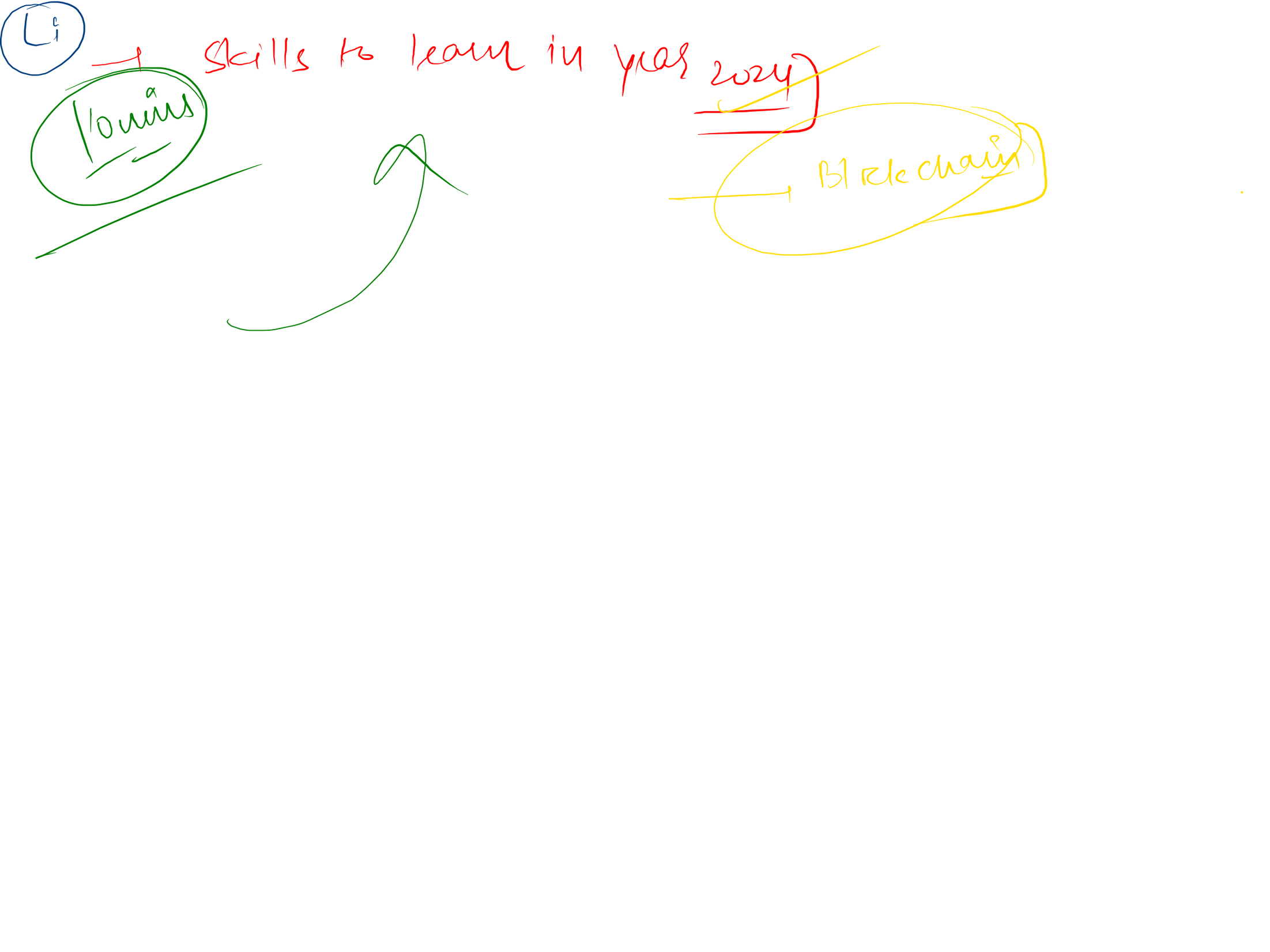
are and same

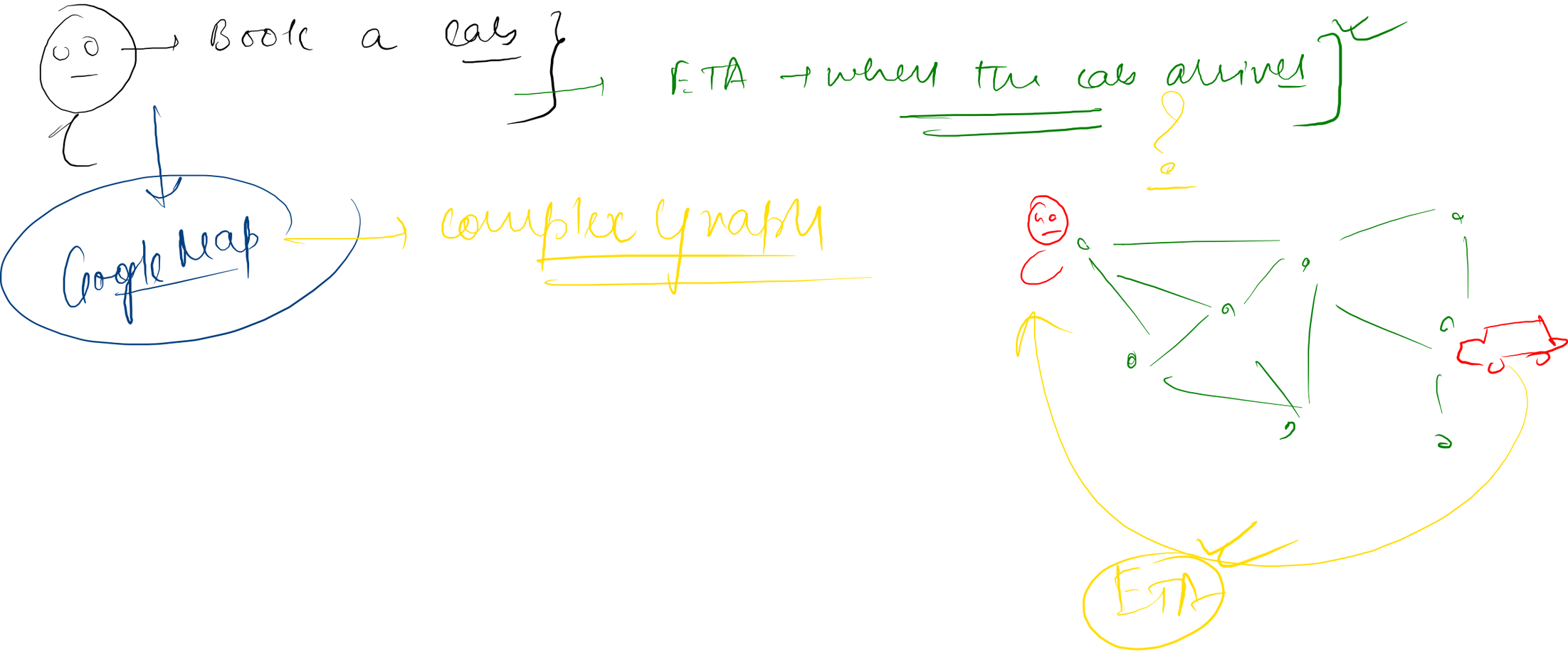
ML

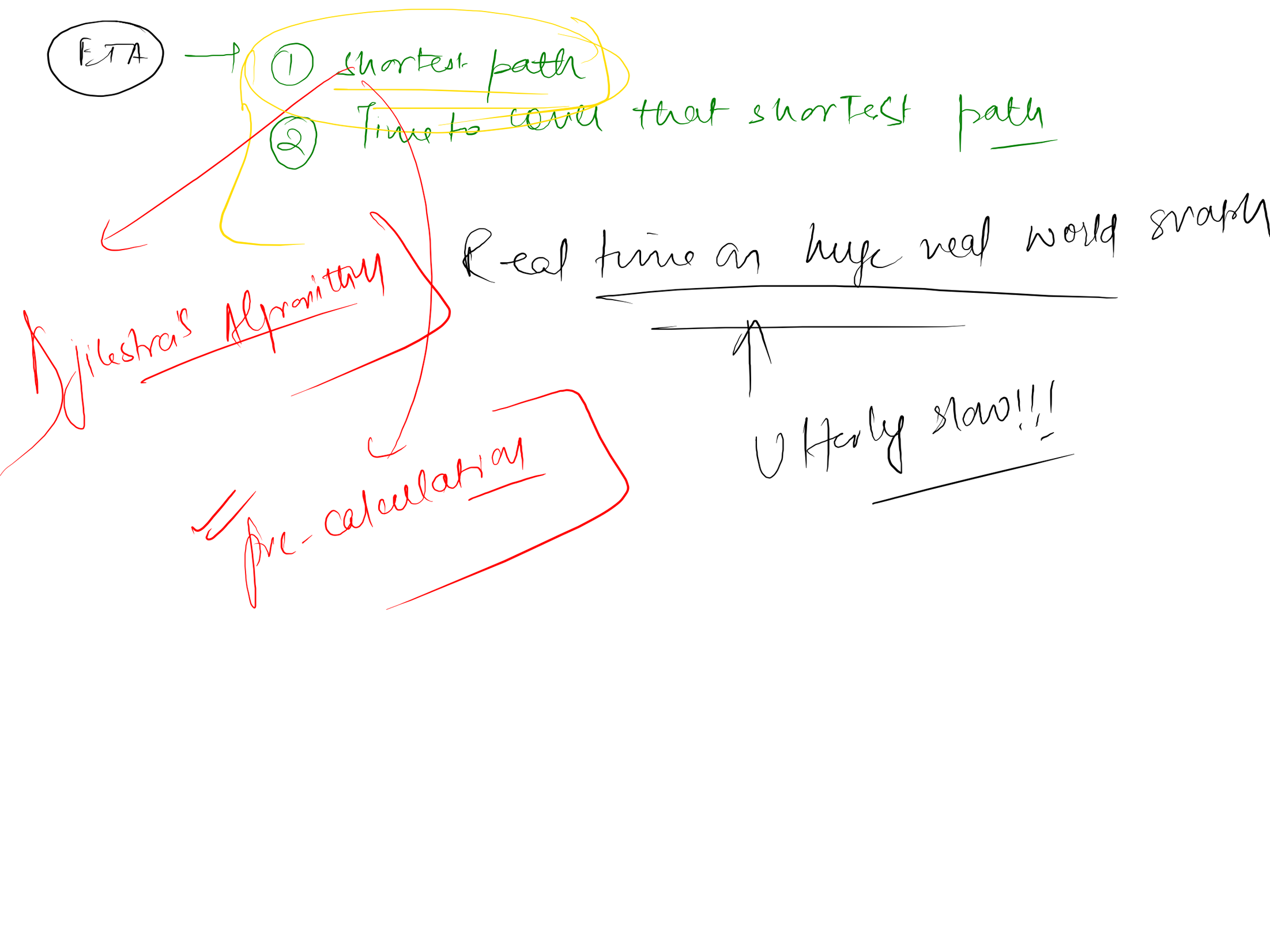


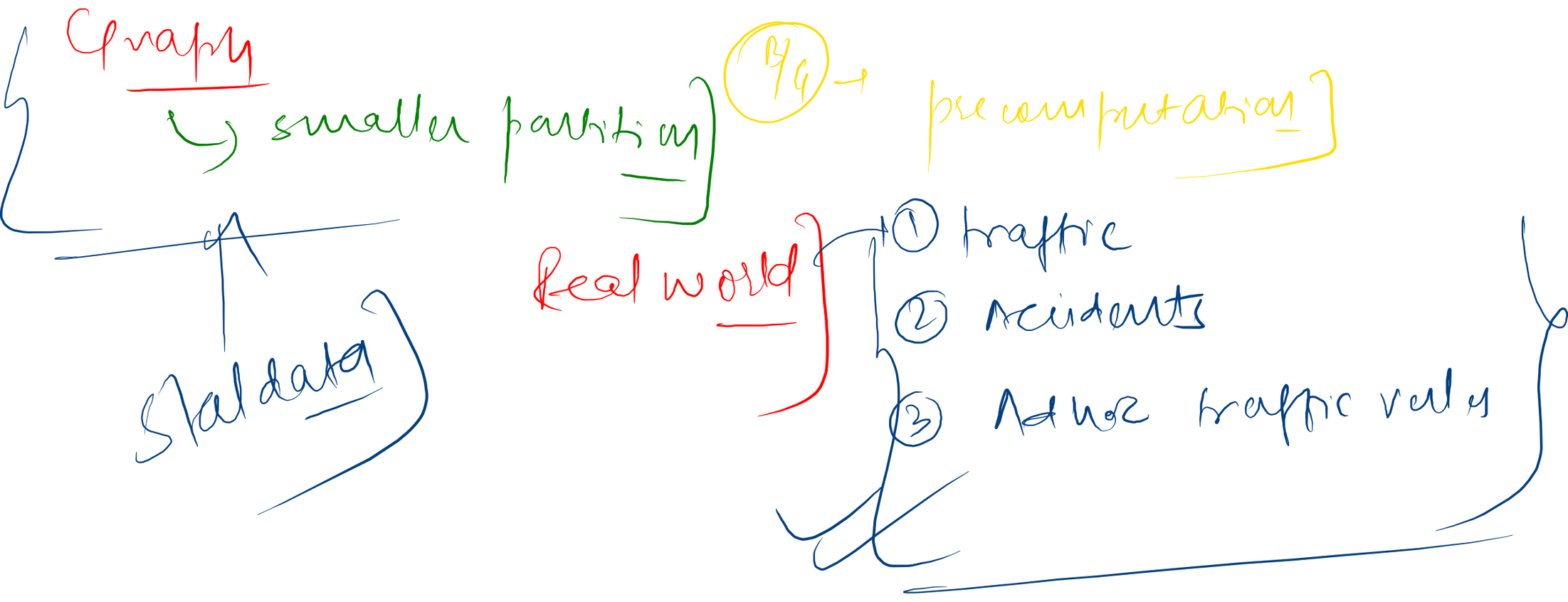


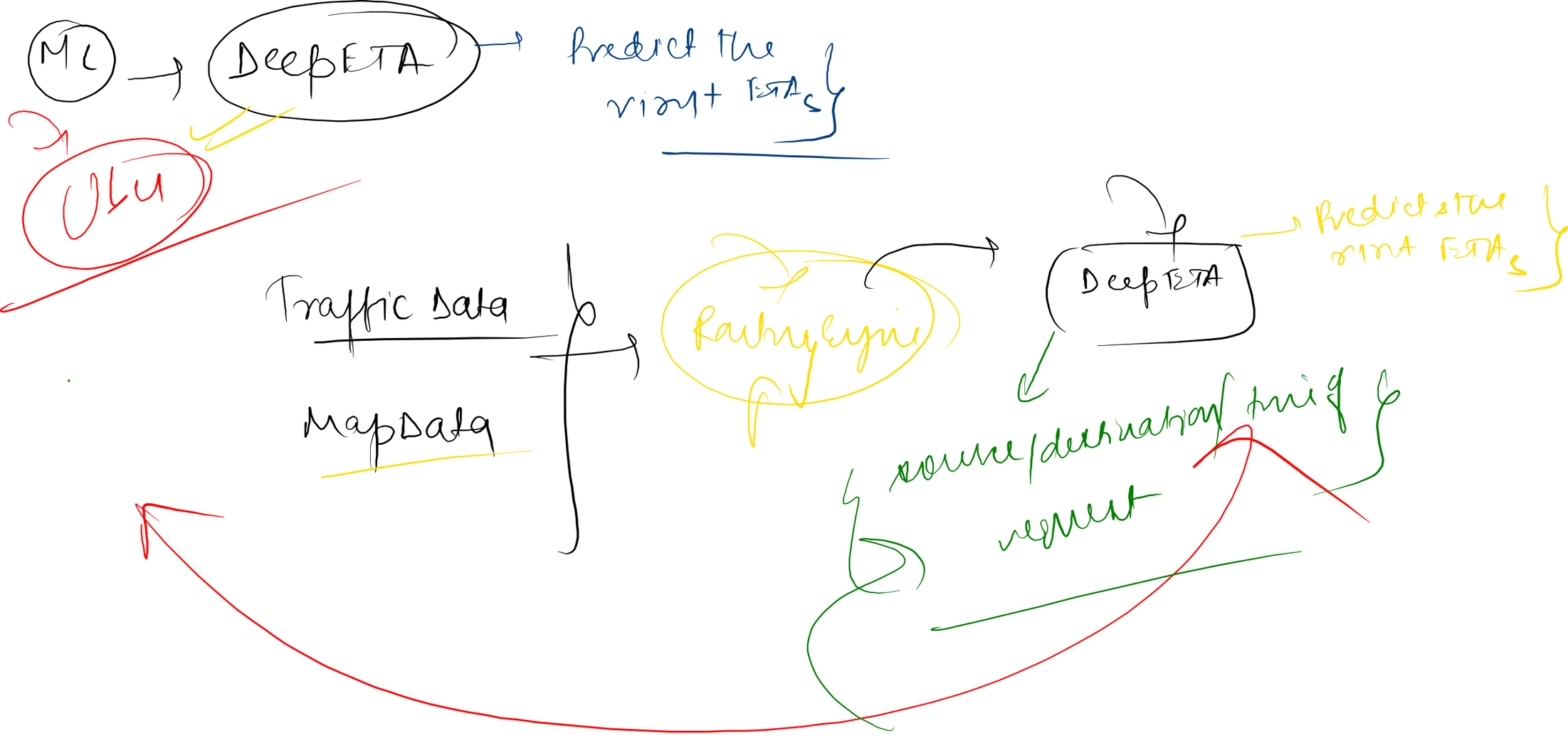
Best profile for 2024

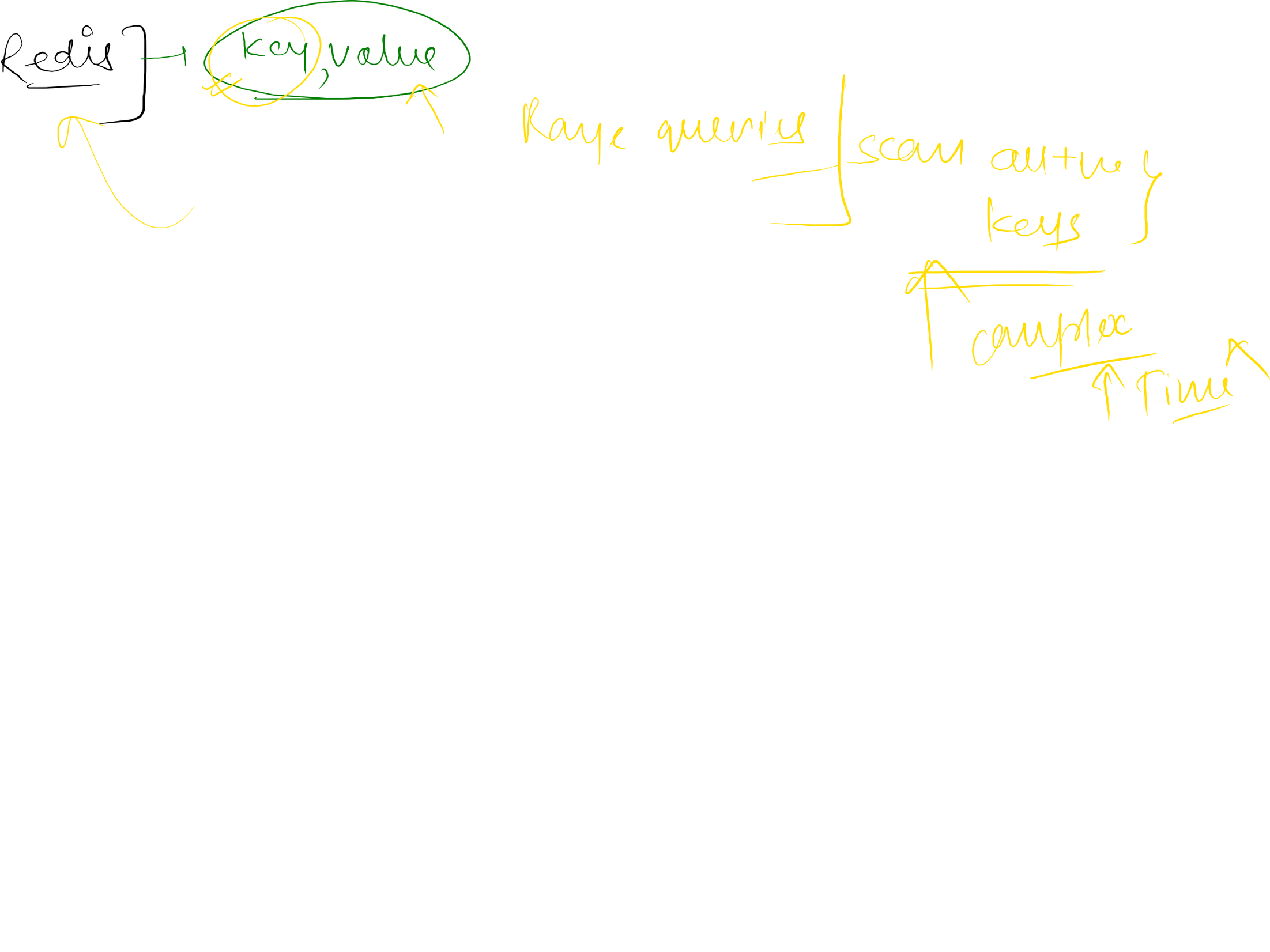

Skills to learn in year 2024
10 min
Blockchain

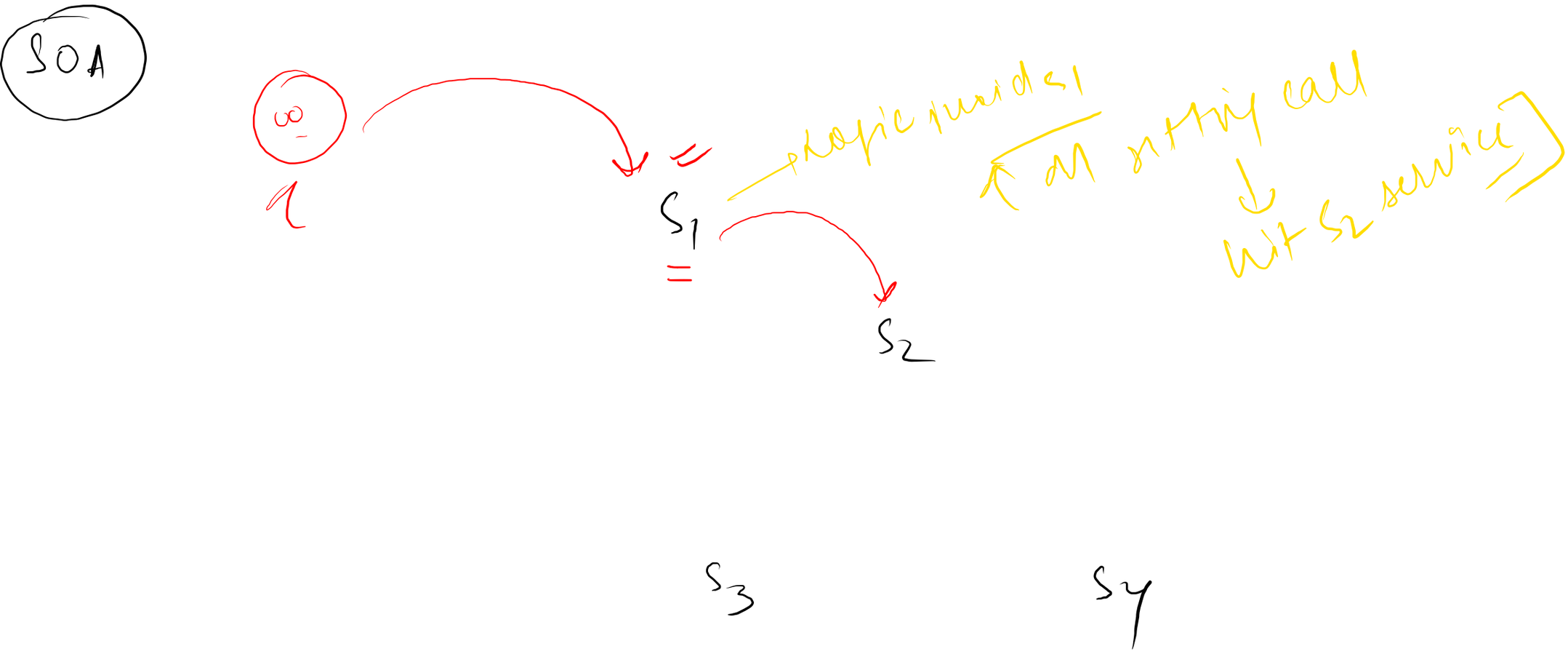

Book a cab
ETA → when the cab arrives
Google Map
complex graph
ETA

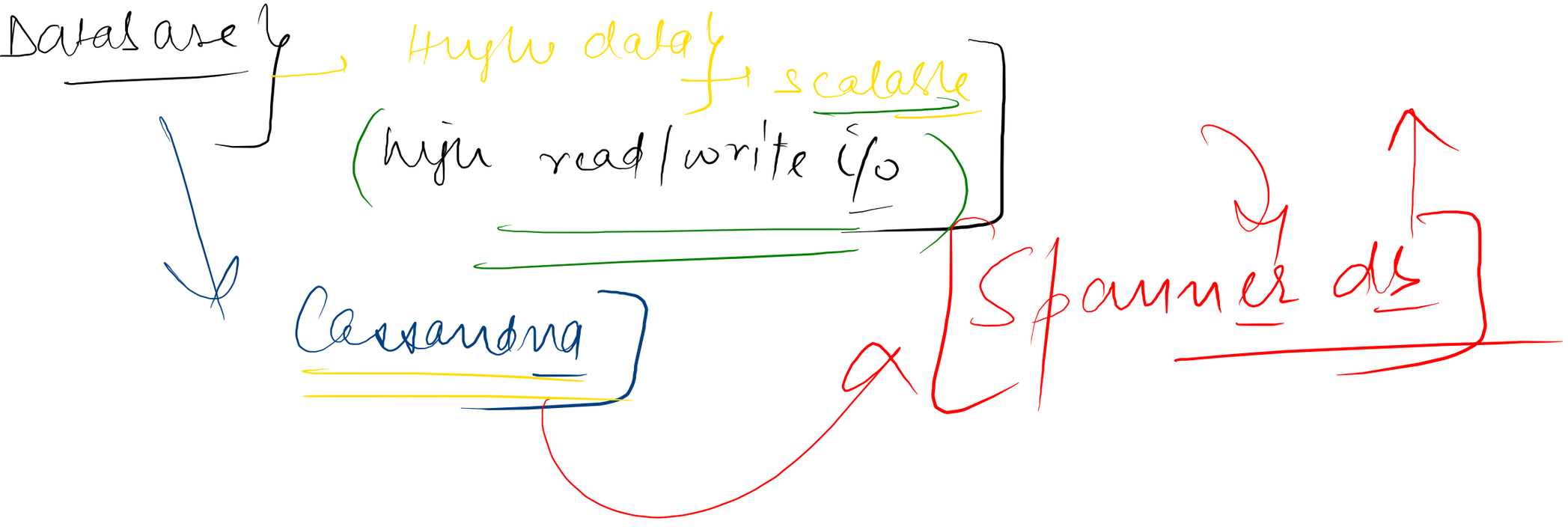

ETA
1 shortest path
2 Time to cover that shortest path
Dijkstra's Algorithm
Real time on huge real world graph
Utterly slow!!!
Pre-calculation

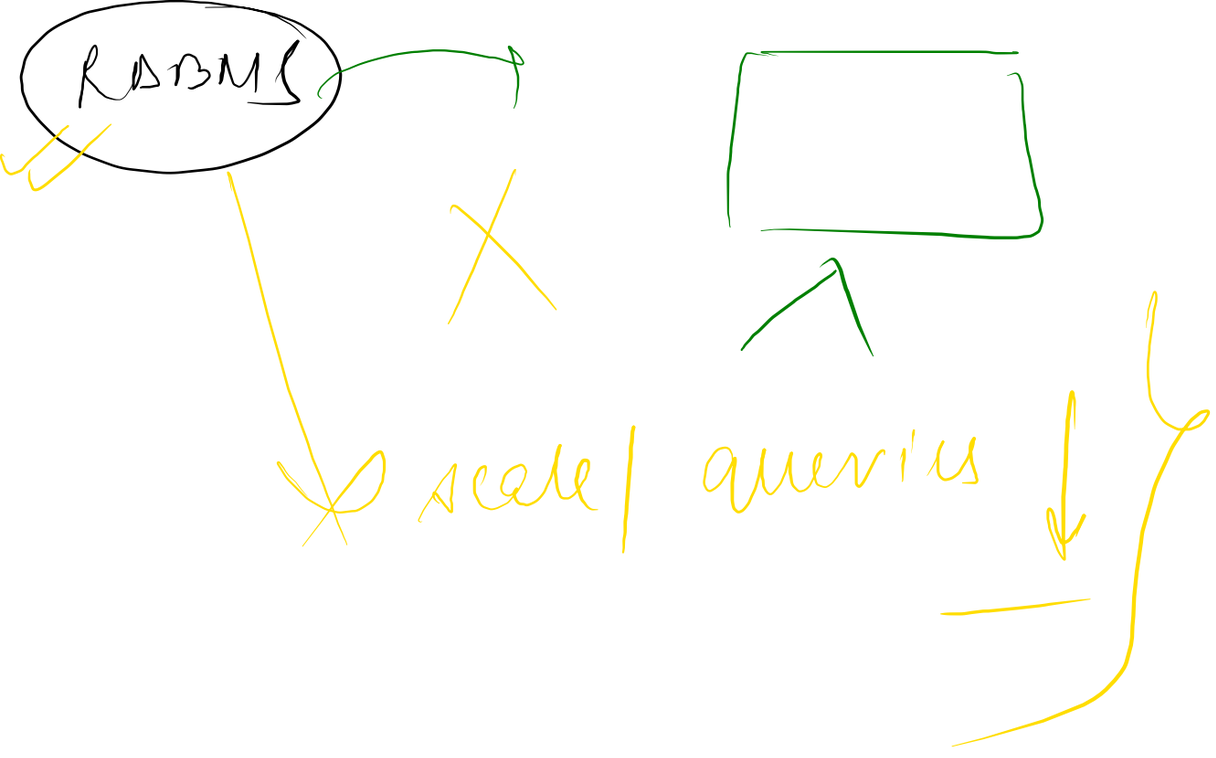

Graph
smaller partition
Static data
Real world
1 traffic
2 accidents
3 Adhoc traffic rules
pre computation

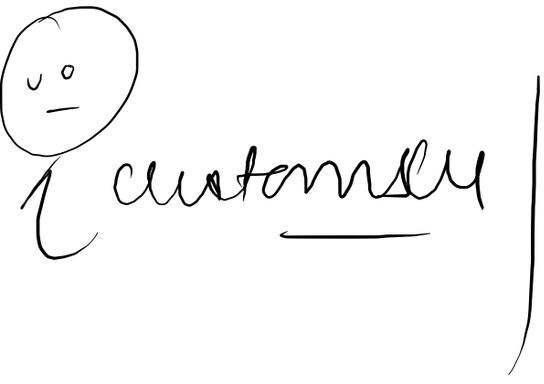

ML
DeepETA
Predict the visit ETAs
ULU
Traffic Data
MapData
Routing Engine
DeepETA
Predicts the visit ETAs
route/driving trajectory
result


Redis
key, value
Range queries
scan all the keys
complex
Time

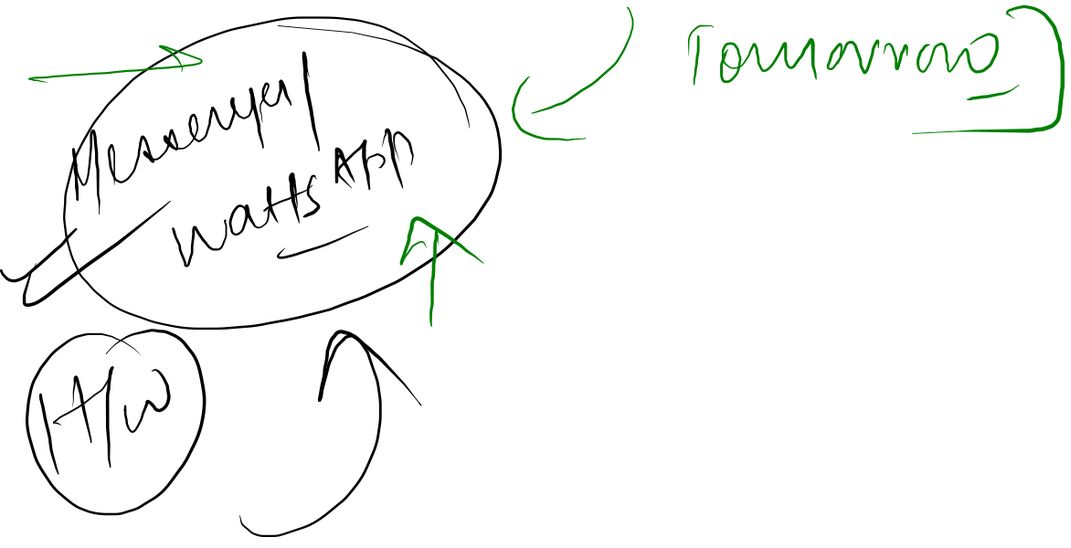

SOA
S_1
S_2
S_3
S_4

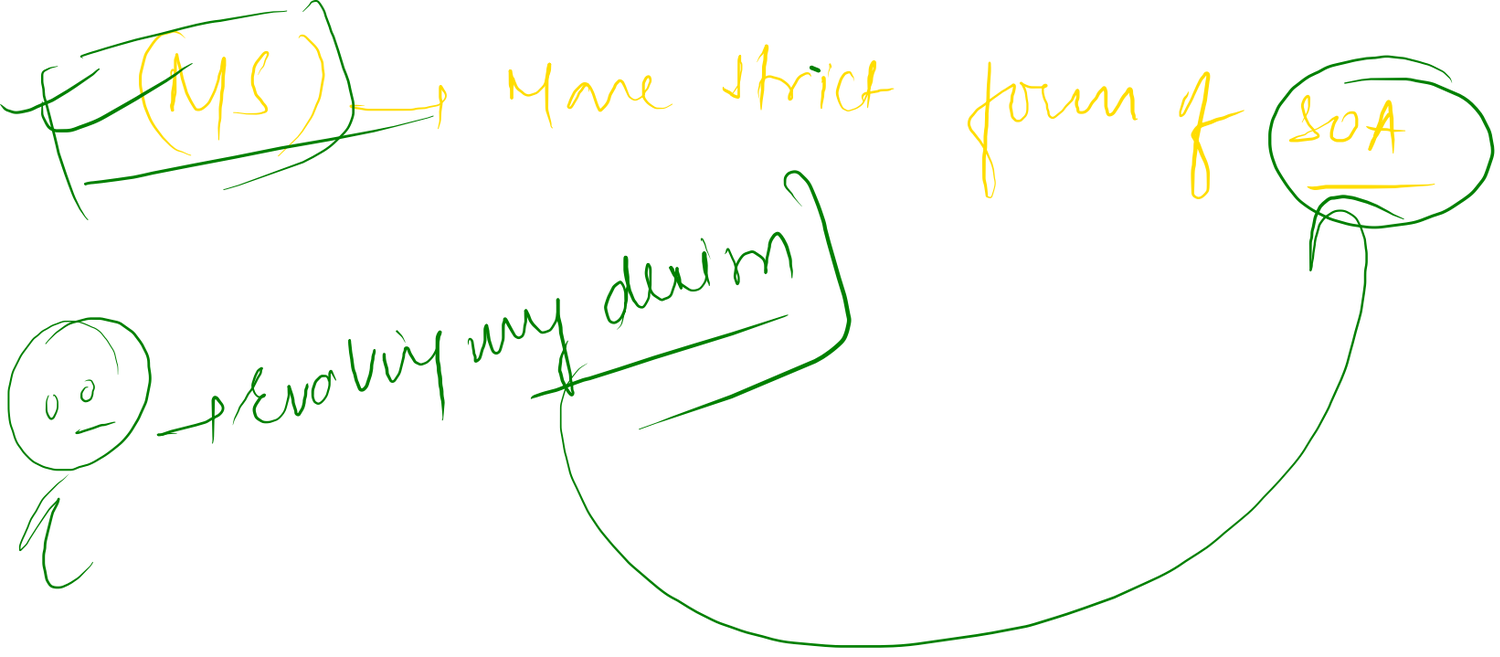

Database
Huge data
scalable
high read/write I/O
Cassandra
Spanner db

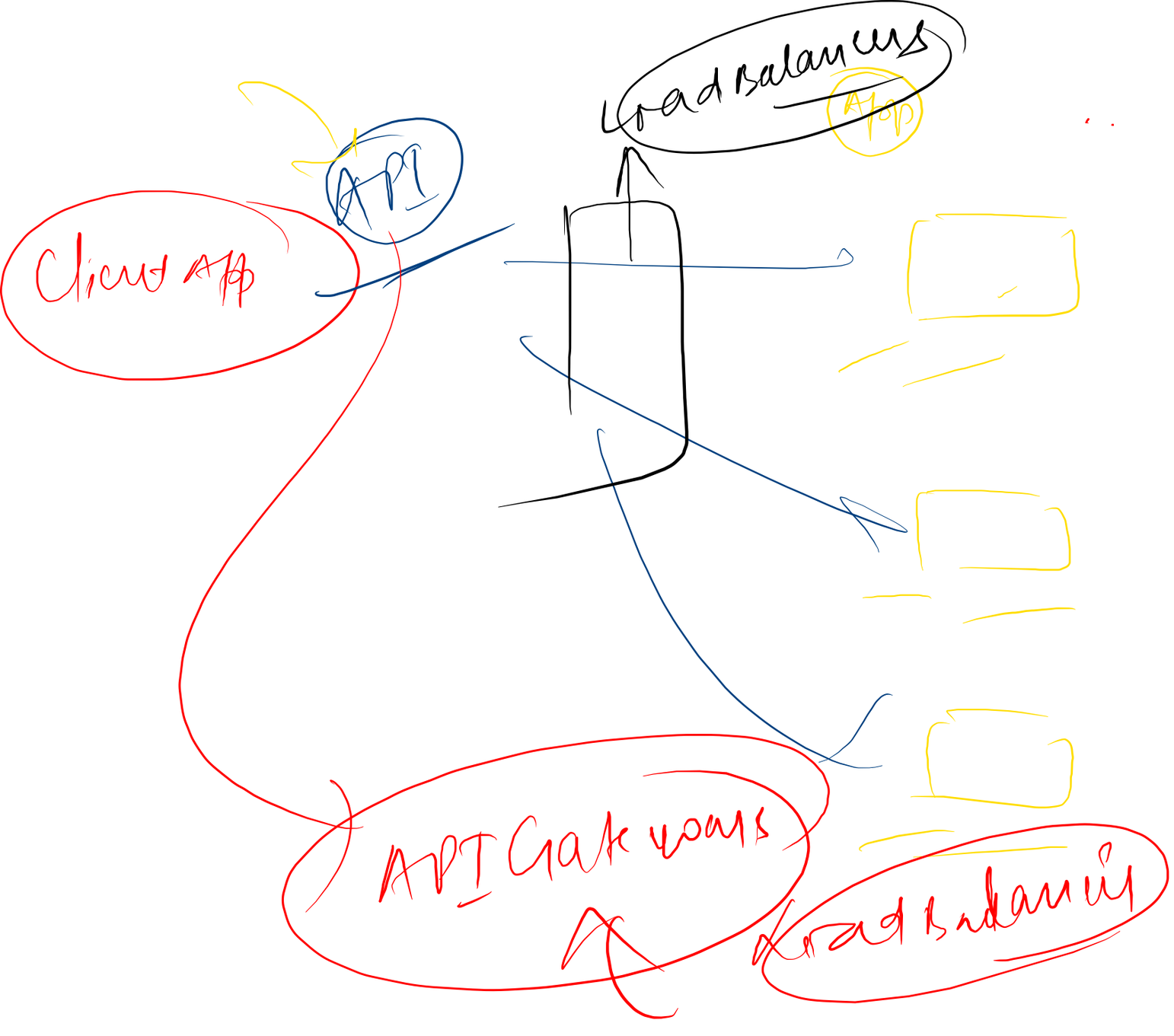

RDBMS
scale queries



① requestRide (riderId, lat, long, destLat, destLong, typeOfVehicle)


Messenger/
Watts APP
H/W


MS
More strict form of SOA
evolving my domain


Load balancers
App
API
Client App
API Gateways
Load balancer