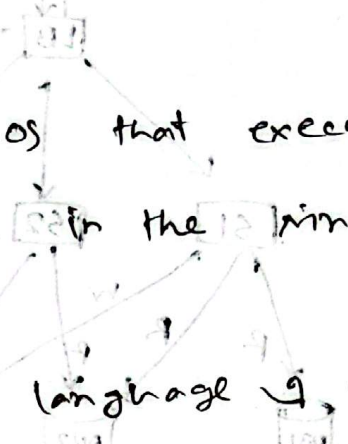
1. Monolithic Architecture
2. Microservice Architecture
3. Single thread, Multithread
4. synchronous, Asynchronous behavior
5. proxy, Reverse proxy



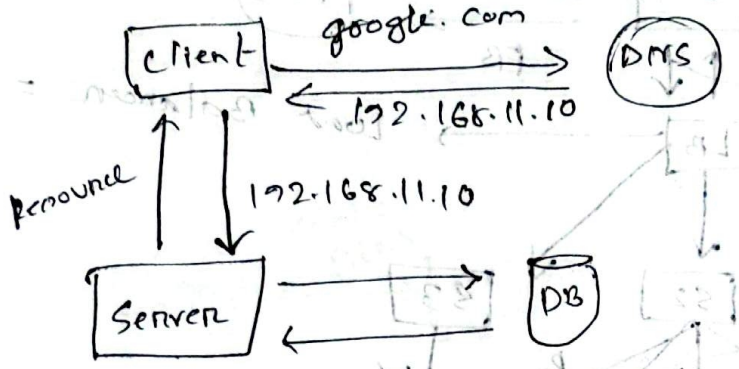
Thread is a component of os that executes the portion of code which are in the single control flow.

Single thread programming language এ, যদি Asyn behavior চাওয়া হয়ে Real time accurate output পেতে পারি, Ex: JavaScript.

proxy: It a software which hide client from server

Reverse proxy: Hide server from client

Single server setup:



There are two major problem with this setup.

→ ~~Server~~ Single point of failure

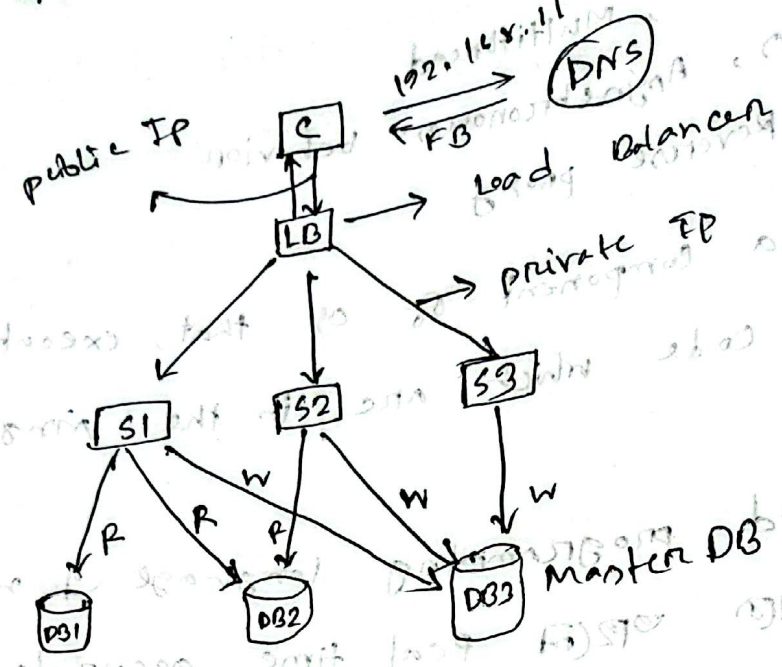
→ High latency

To solve this problem we need scaling:

Vertical scaling/Scale up: Means adding more power to our cpu. It is single point of failure

Horizontal scaling/Scale out: means adding more servers to our pull of resources. Both problems are solved.

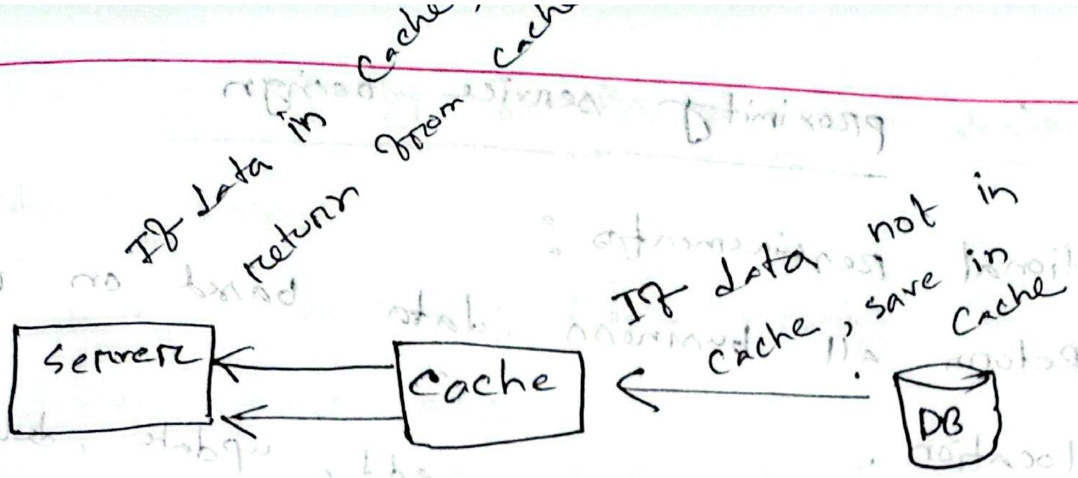
DB ঘাড় Horizontal scaling. এর DB replication

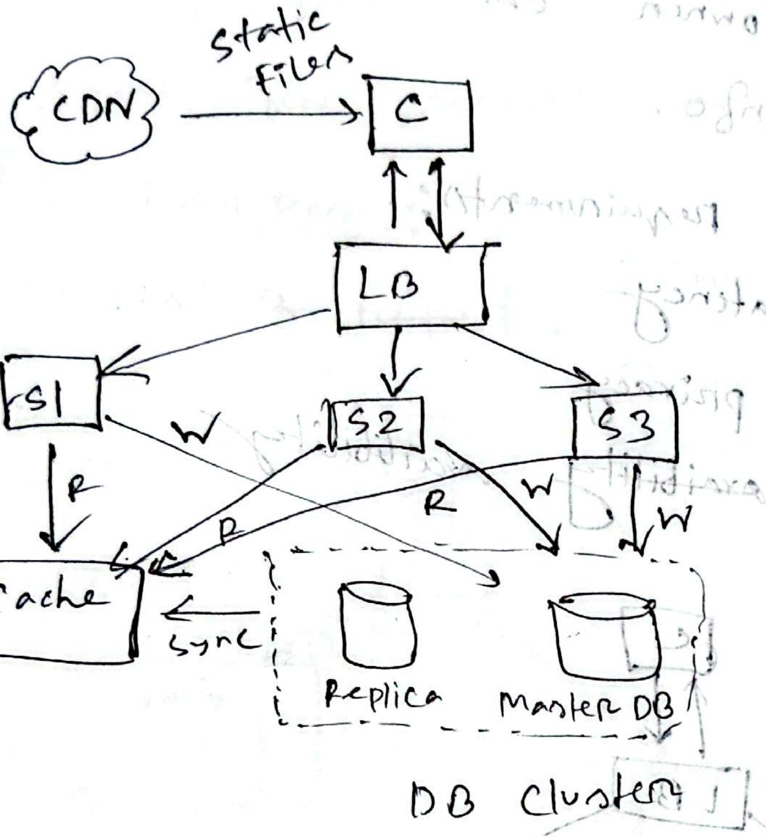


LB distributes incoming traffic among the web server

CDN → A content delivery network is a network of geographically dispersed servers that serve static content.

cache → It a temporary storage area that stor the result of exten responses on frequently access data.





single server single point failure, loading, latency. হচ্ছে সমস্যা [illegible]। Answer: reason এক্সেপশন থাকতে হবে।

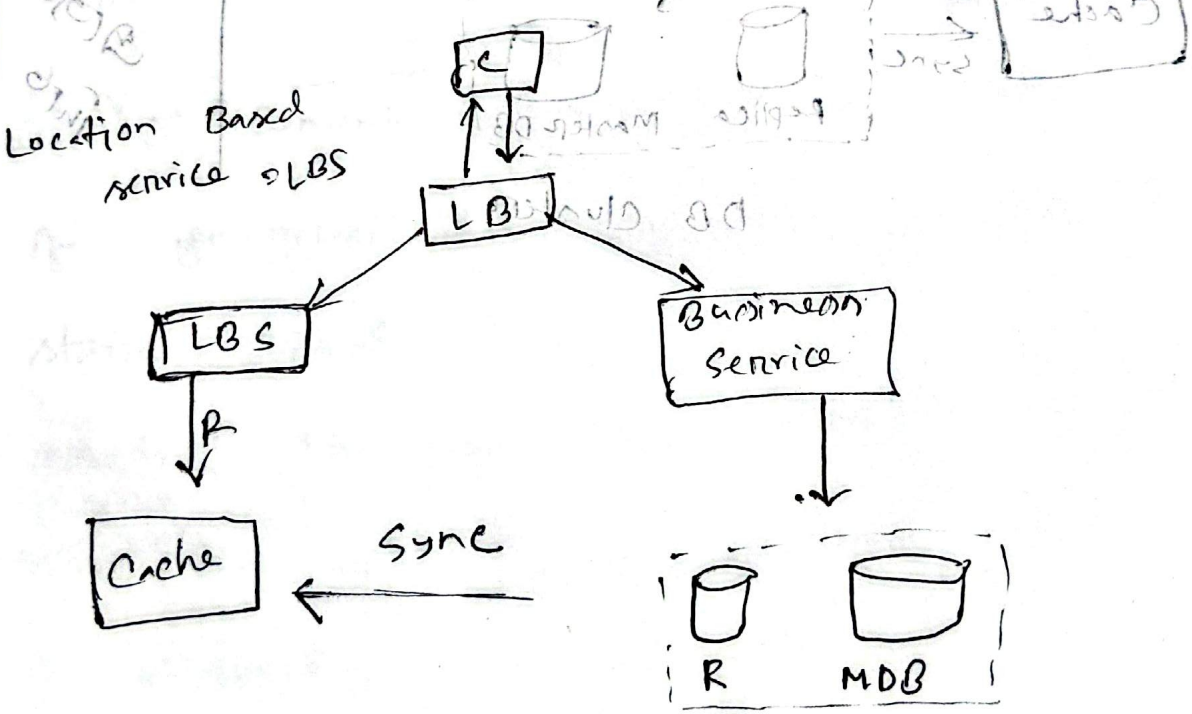
proximity service design

functional requirements :

1. Return all business data based on user location.
2. Business owner can add, update, delete their business info.

non-functional requirements :

1) low latency
2) data privacy
3) High ~~availibility~~ availibility



Nearby friends system design

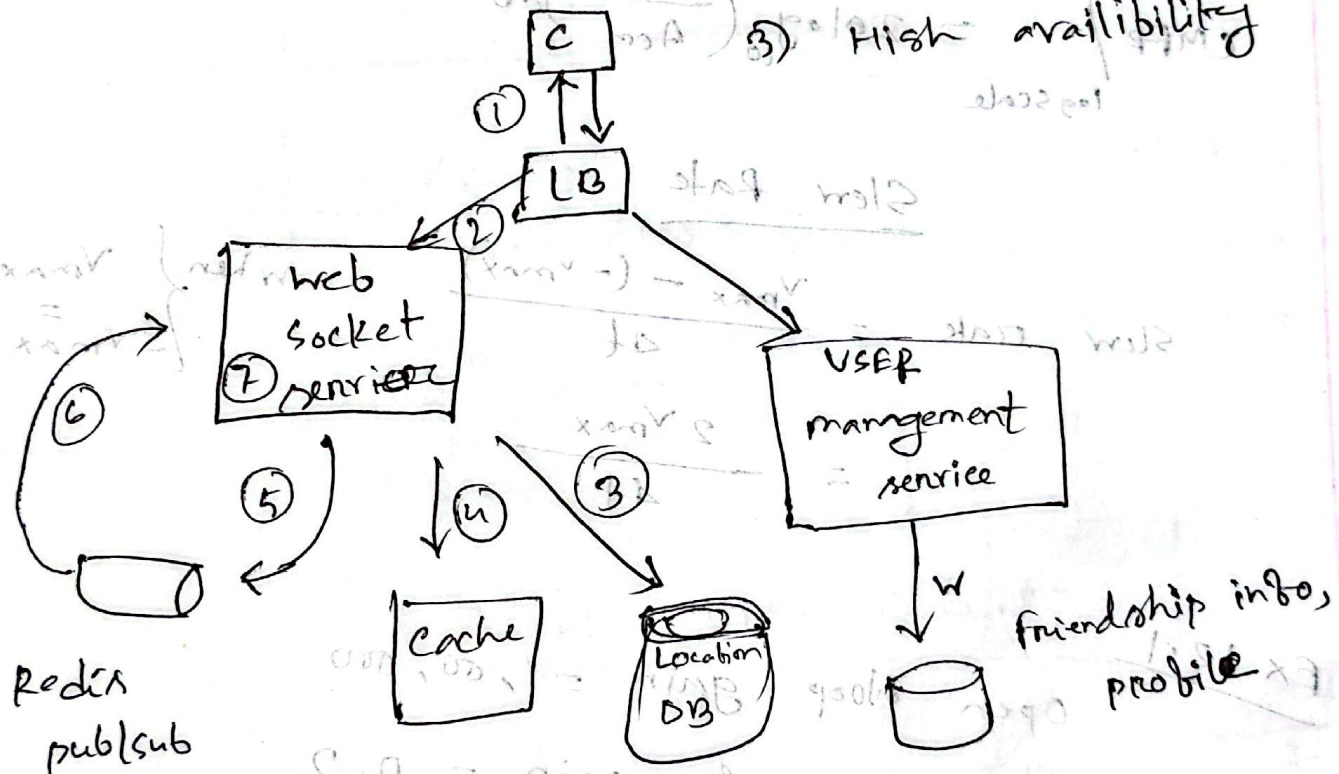
Functional req:

1. users can see their nearby friends on the mobile app
2. Nearby friends list should be updated every few seconds

Non - functional Req :

~~previous~~ ~~3~~ ~~repeat~~.

1) low latency
2) data privacy
3) High availibility



Software

1. The mobile client sends location update to the load balancer.
2. The load balancer forwards the location update to the persistent connection on the web socket server. for that client
3. web socket server saves the location data to the location DB.
4. Web socket server update the cache with new location.
5. The websocket server publishes the new location to the users channel. in the redis pub/sub server.

6. Redis pub-sub server broadcast the update to all the subscribers.

7. Web-socket server (Friends) computes the distance between the user and the subscribers. If the distance is ~~less~~ doesn't exceed the search radius the location is sent to the subscriber client.

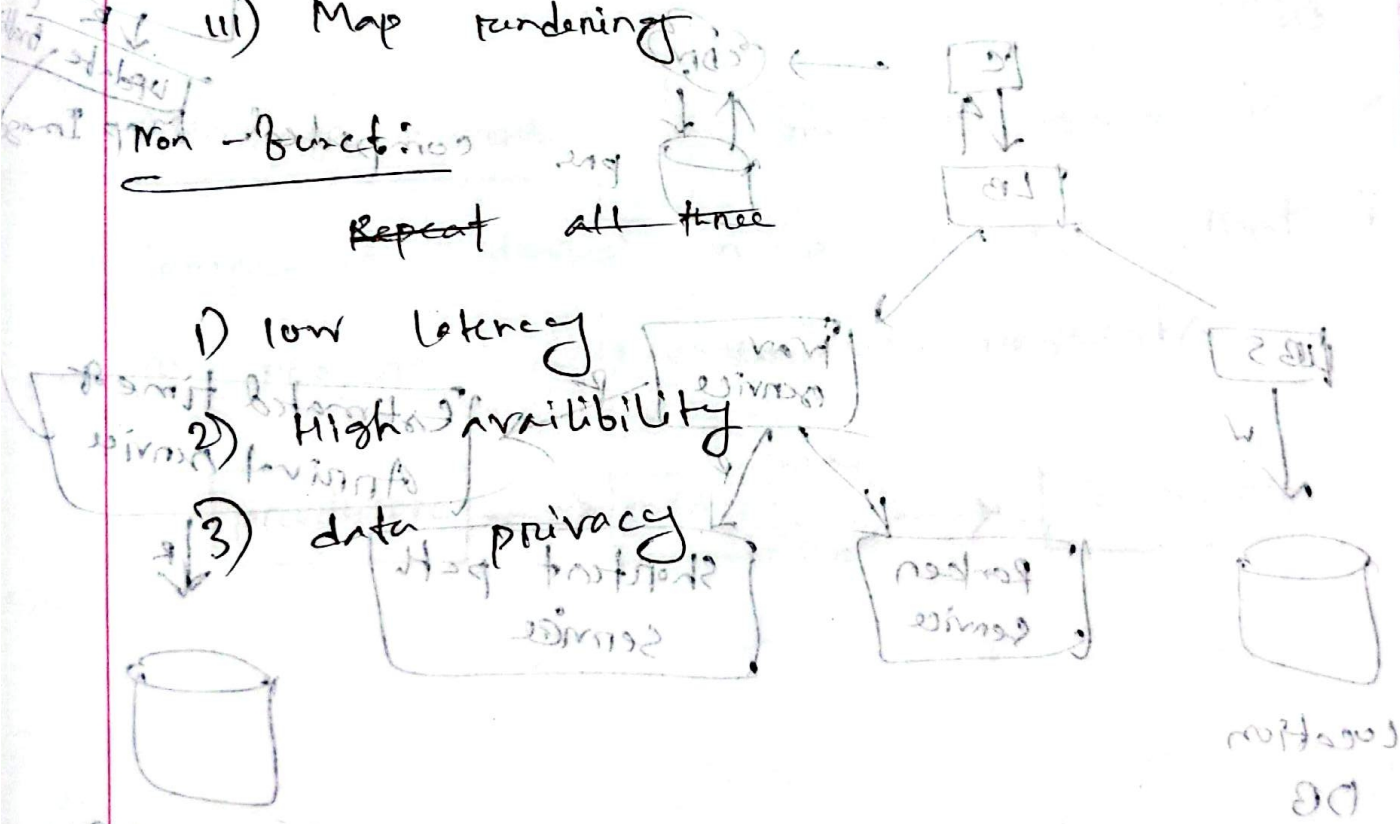
System design of google Maps

Functional Requirement : i) Users location update
ii) Navigation service including estimated time of arrival.
iii) Map rendering

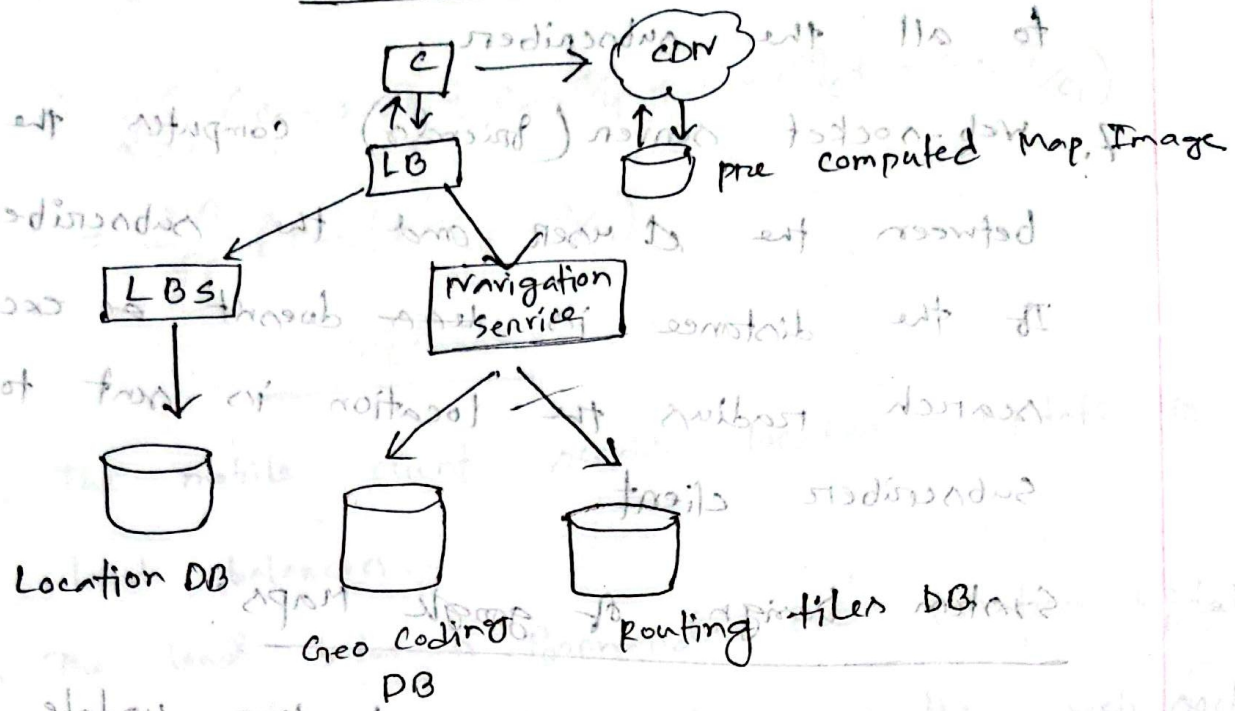
Non-functional

~~Repeat all three~~

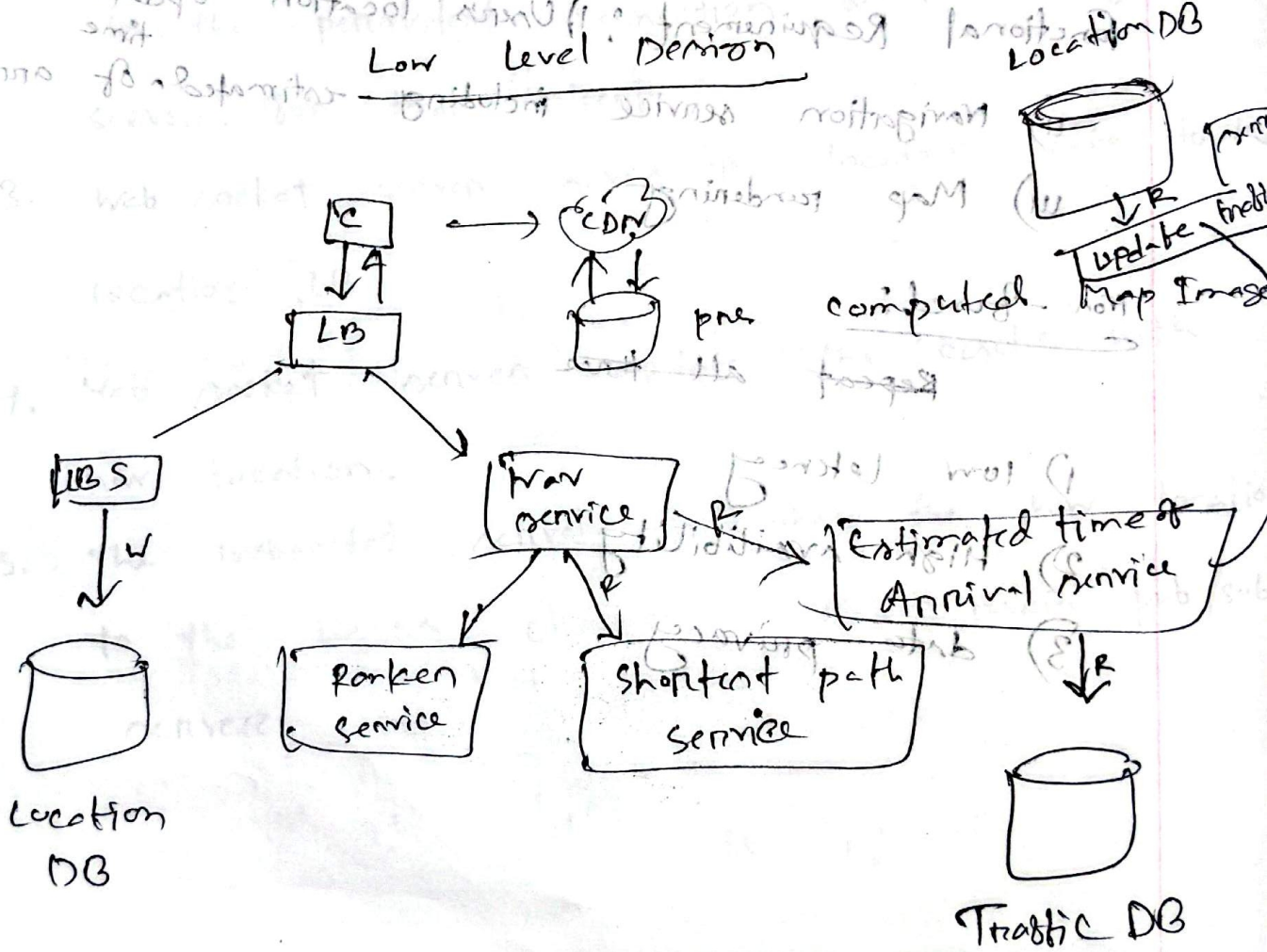
1) low latency
2) High availibility
3) data privacy



High level Design

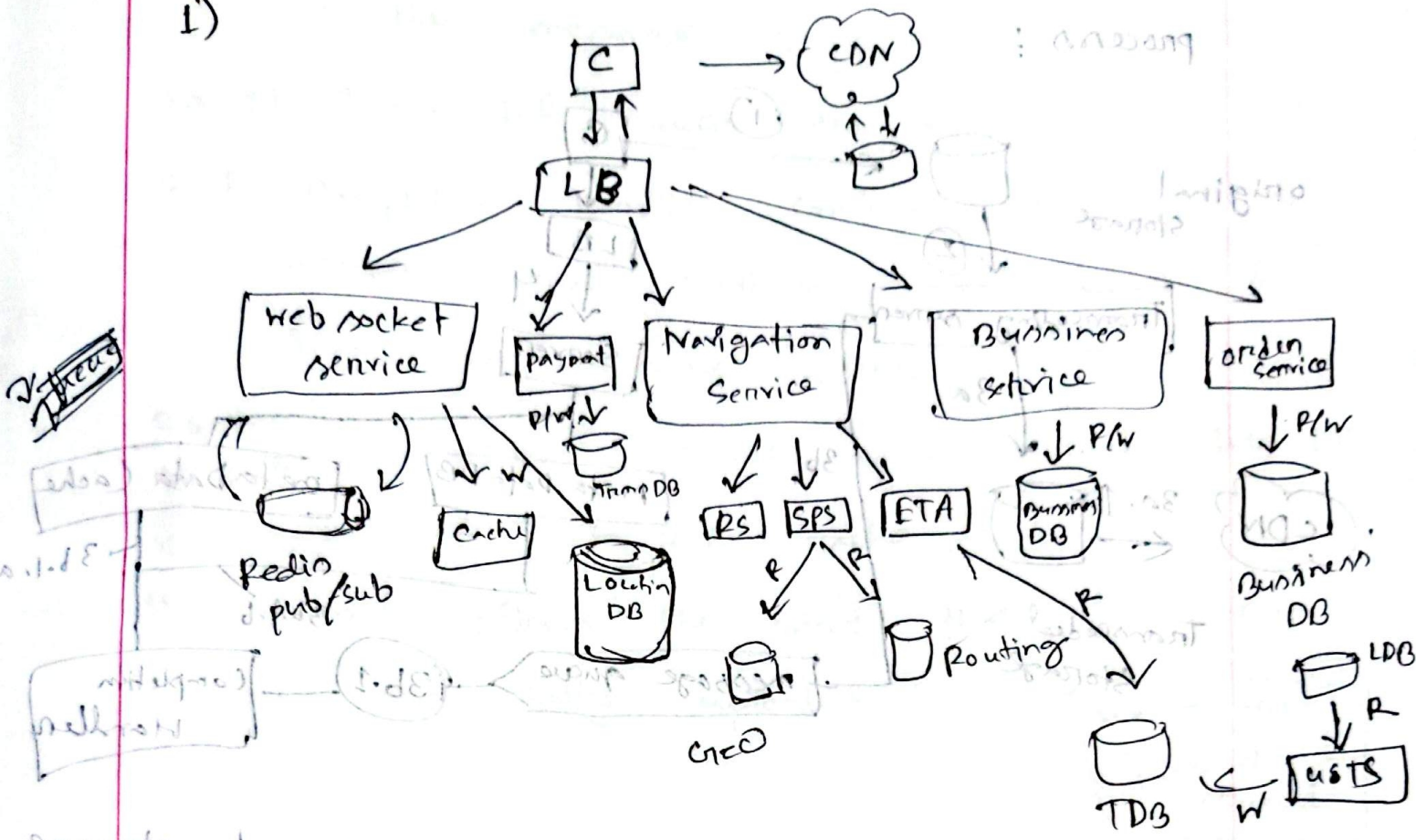


Low level Design

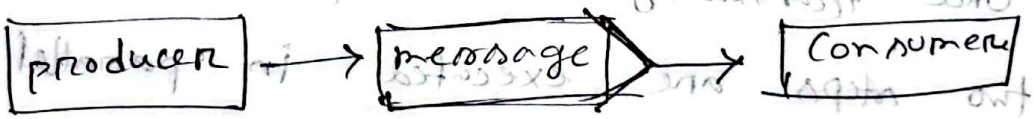


Food panda

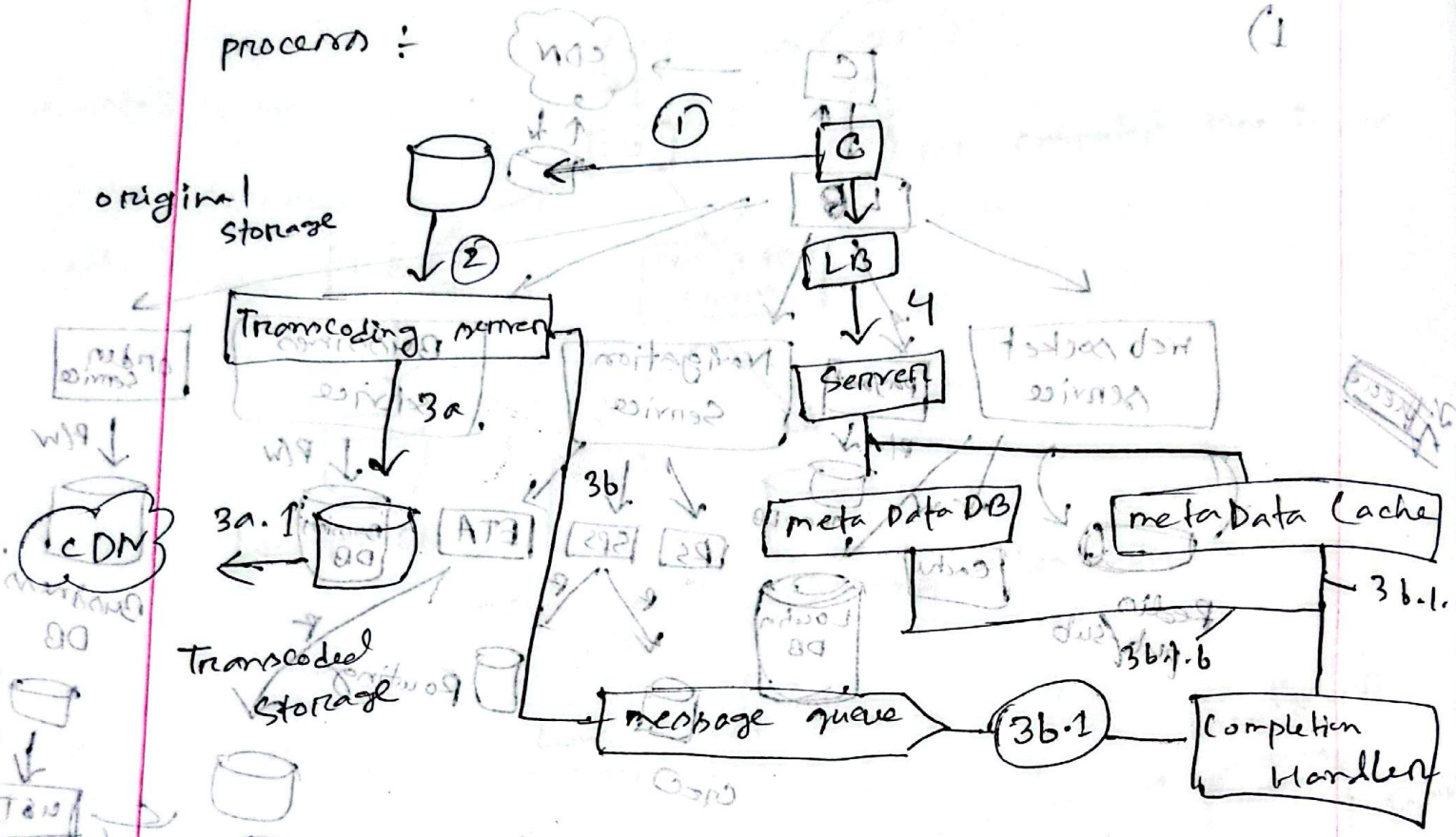
1)



Message queue: A message queue is a durable component stored in a memory that is used to process asynchronous requests.



System ~~video~~ Design of Youtube's video uploading process :



1. Videos are uploaded to the original storage
2. Transcoding servers fetch video from original storage and ~~starts~~ transcoding.
3. Once transcoding is complete, the following two steps are executed in parallel.

   3a. Transcoding videos are sent to transcod storage.

Transcoding → Device dependent to independent

3b. Transcoding completion events are queued in the message queue.

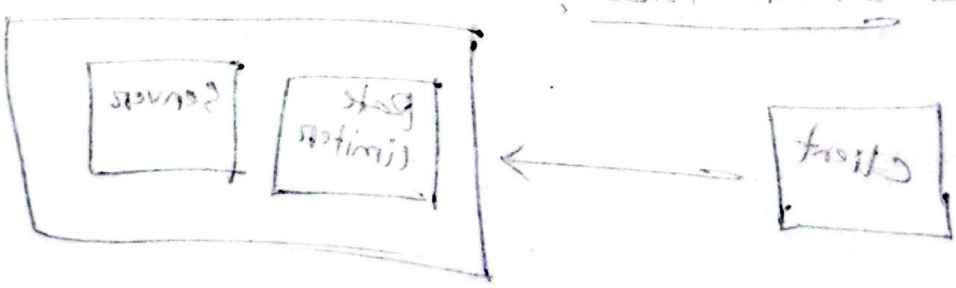
3a.1) Transcoded videos are distributed to CDN.

3b.1) Completion handler contains a bunch of workers that continuously pull event data from the message queue.

3b.1.a
3b.1.b } Completion handler updates meta data DB and meta data cache.

4. Servers inform the client that the video is successfully uploaded.

workers are kind of threads



System Design of a Rate Limiter:

In a network system, a rate limiter is used to control the rate of traffic sent by a client/server. In the HTTP world, a rate limiter limits the number of client requests.

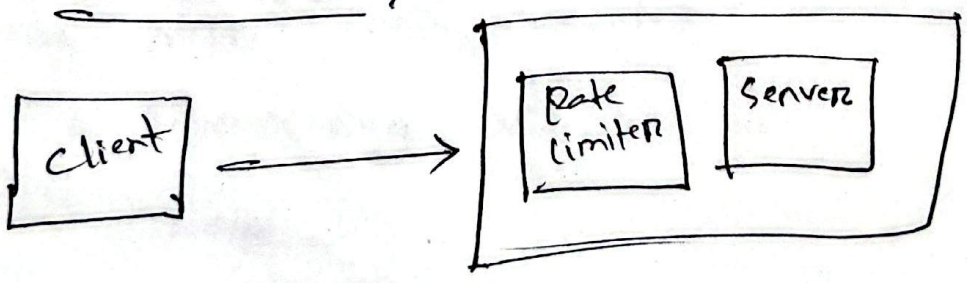
Benefits of using a rate limiter:

1) It prevents resource starvation caused by denial of service attack.

We can implement a rate limiter either a client side or server side

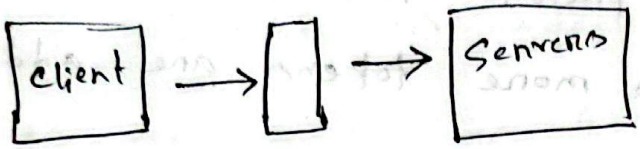
Client side: An unreliable place to enforce rate limiting.

Server side:



Software

Instead of putting a rate limiter at the server side we can create a rate limiter middleware. which throttles request to our server.



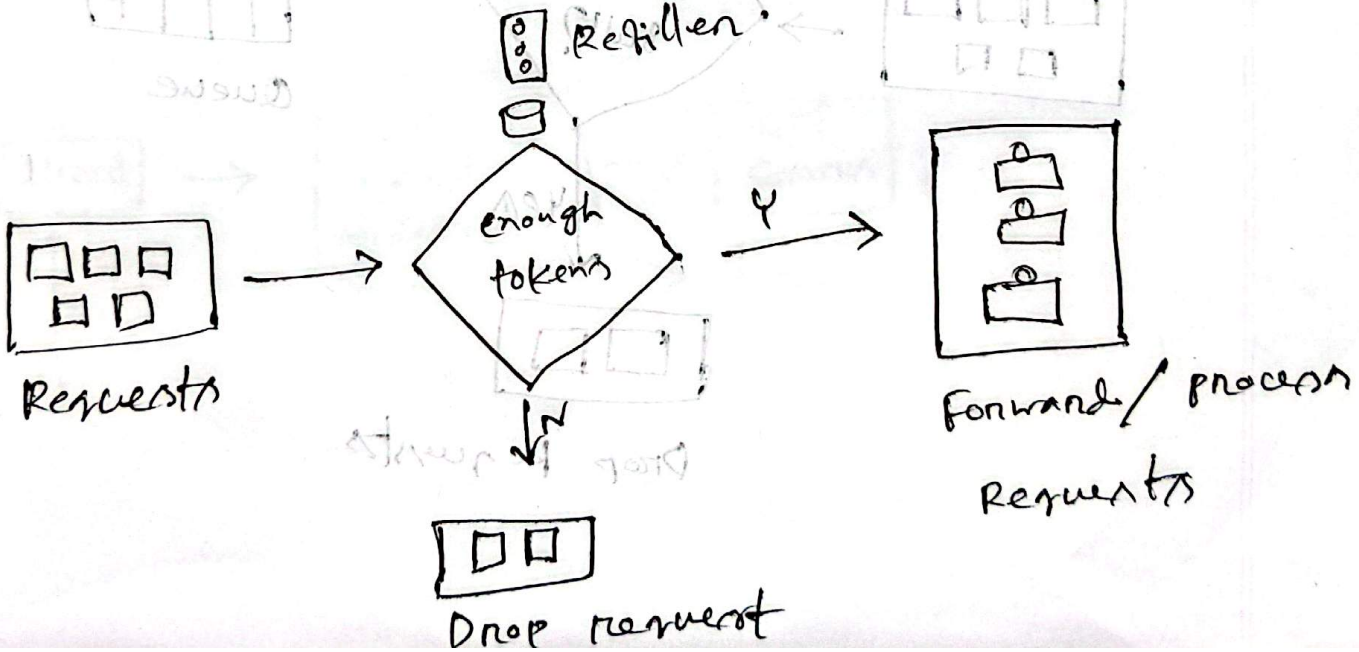
Rate limiter middleware

middleware → function

Let's assume our API allows two request per second. and a client sends 3 request within a seconds. The first two request are routed to servers. however the rate limiter throttles the third request and returns HTTP status code 429.

Algorithm for rate limiting

* Token bucket algorithm



Algo.

The token bucket works as follows:

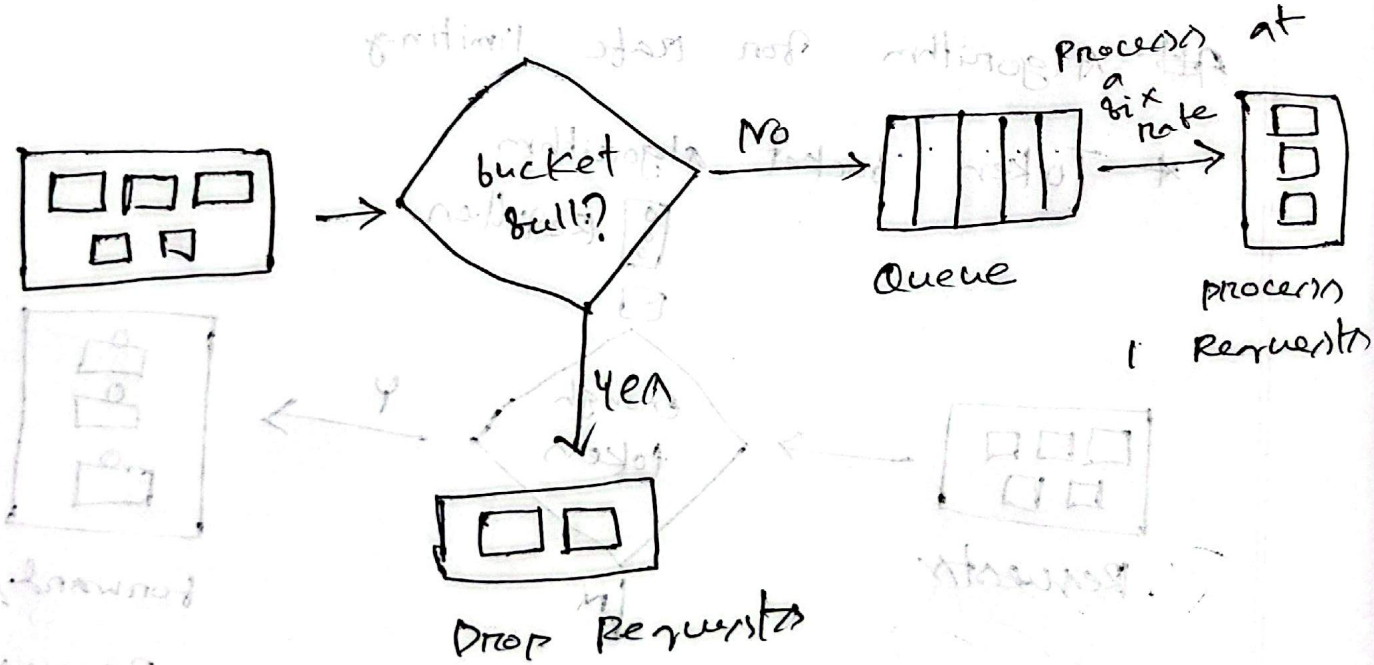
1) A token bucket is container that has predefined capacity. Tokens are put in the bucket at a preset rates. Once the bucket is full no more tokens are added.

2) Each request consumes one token.

2.1) If there are enough tokens we take one token out for each requecst.

2.2) No token request drop.

* Leaking bucket: Algorithm:



The Algo works as follows:

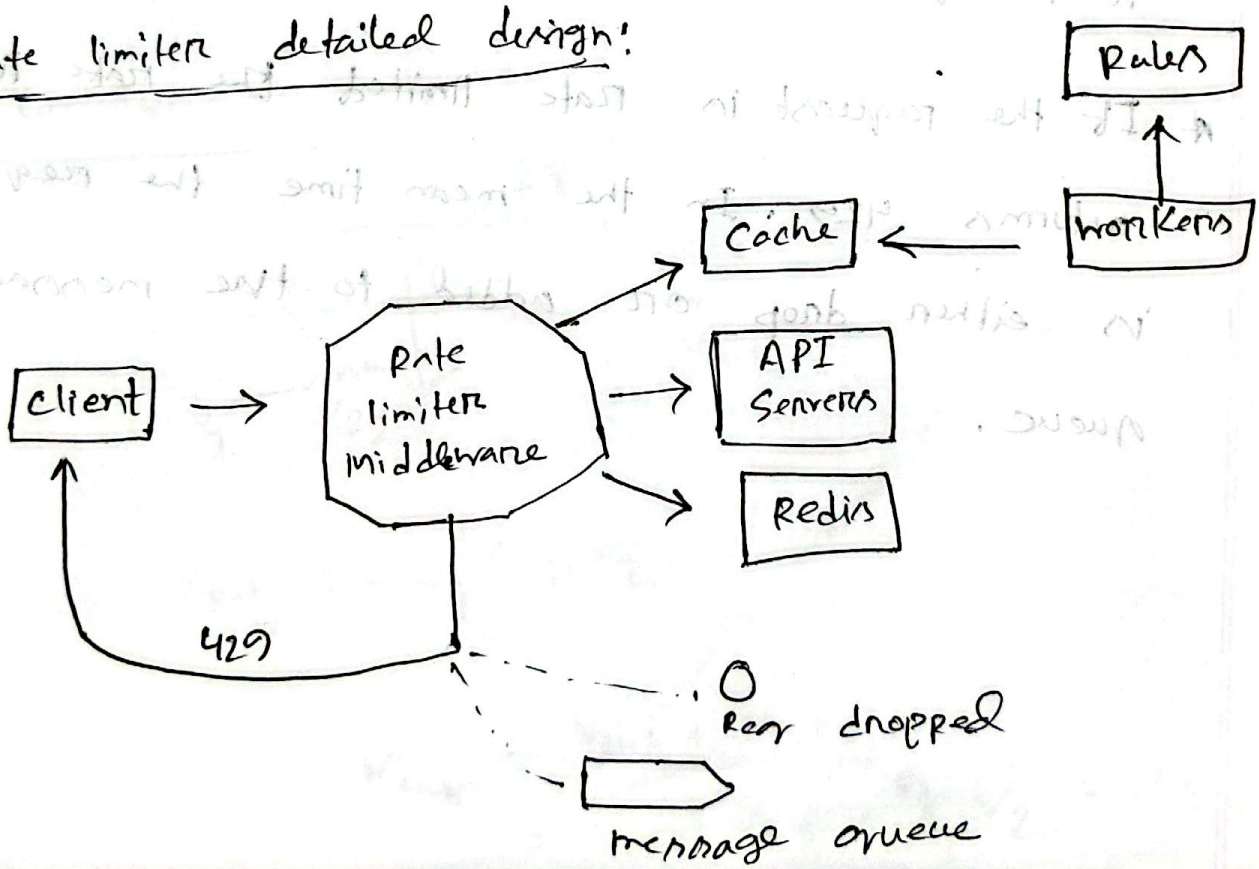
1) When a request arrives the system check if a the queue is full. If it is not full the request is added to queue.

2) If the queue is full the request is drop.

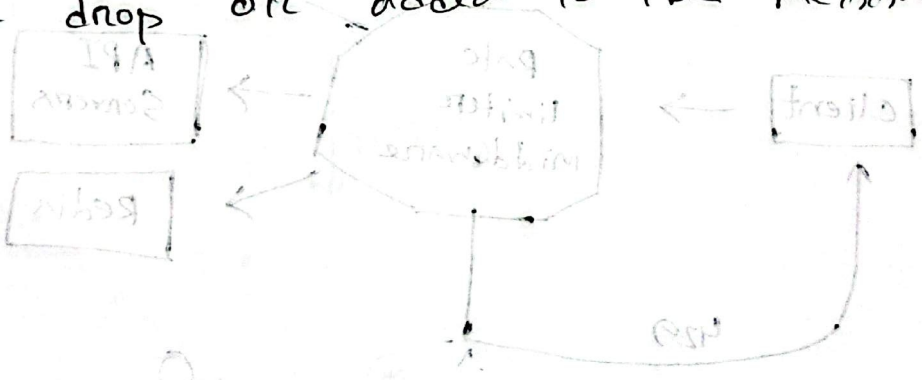
3) Requests are pulled from the queue and process at a regular interval.

[N.B If a user is allowed to make one facebook friends per second, add 150 per day, like 5 post per second. Three buckets are required for each user.

Rate limiter detailed design:

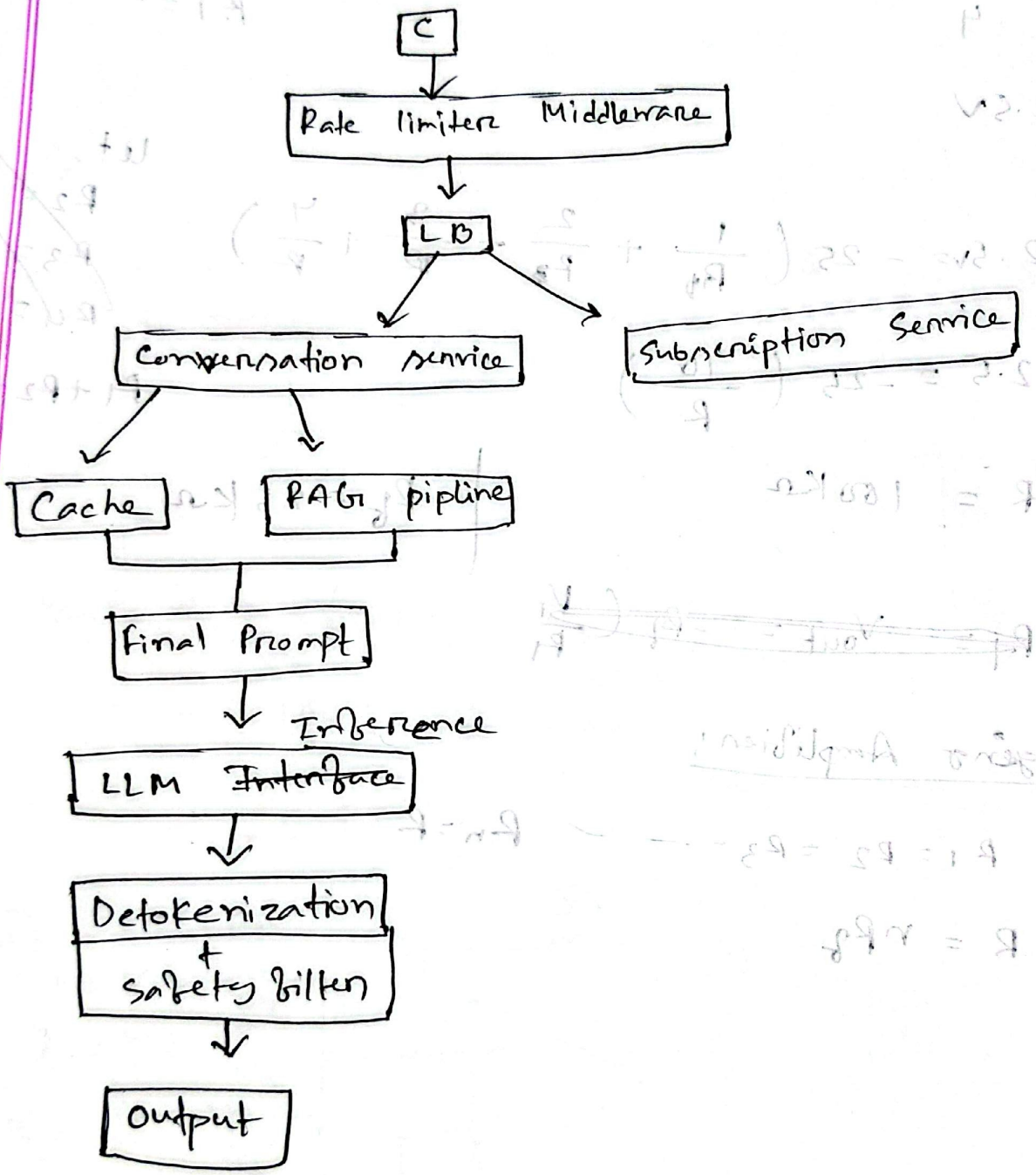


* Rules are stored on the disk. workers frequently pull rules from the disk and stored them in the cache.
* When a client ~~the~~ sends a request to the server, the request is sent to the rate limiter middleware first.
* Rate limiter middleware loads rules from the cache. It fetches the values of counters (variable and last request time-stamp. from redis.
* If the ~~user~~ request is not rate limited it is forwarded to API servers.
* If the request is rate limited the rate limiter returns 429. In the mean time the request is either drop or added to the message queue.



System Design of chatgpt

↳ Generative pretrained Transformer



How LLM works!