CHAT APPLICATION DESIGN

(1) Whatsapp

(2) Telegram

(3) Facebook Messanger

etc...

CHAT APPLICATION DESIGN

Functional Requirements:

a) One to one chatting

b) Status (last Seen, Online)

c) Send audio, video, images

d) Group chatting

e) Read Receipts (Sent, delivered, Seen)

Non- Functional Requirements:

a) Low Latency

b) High Availability

c) High Consistency

CAP Theorem from the corner laughing ☺

Estimations:

a) Registered users : 2 Billion

b) Daily Active users: 1.5 Billion

c) Daily Active messages : 1.5 X 20

= 30 Billion

To handle 30 Billion messages we need to follow distributed approach

LIST OF SERVICES:

a) WebSocket Handler

b) WebSocket Manager

c) Message Service

d) Media Service

e) User Service

f) Group Service

g) Last seen service

h) analytic service

Databases used:

a) Cassandra : To read & write billions of messages.

b) Redis: To keep track of user_ids & corresponding websocket Handler ips

c) Redis: To store latest user & group information

d) Amazon S3: To store images, audio, videos

Possible combinations:

1) User online sent a message

2) User offline sent a message

3) User online received a message

4) User offline received a message

APIs:

POST /user/message-id

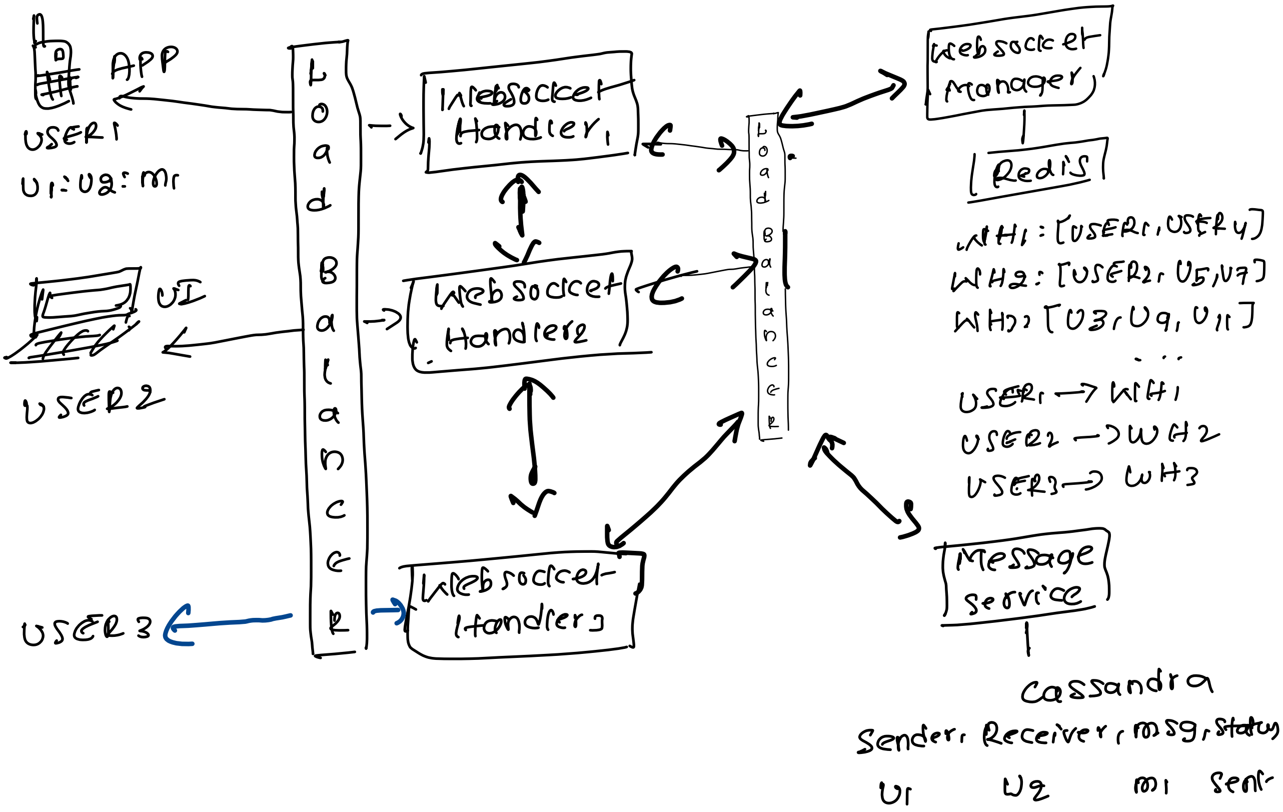
GET /USER/message-id

GET /USER-id/allMessages

POST /USER/group_message-id

GET /USER/group-message-id

HIGH LEVEL DIAGRAM:



Dry Run:

/Post /USER / msg-id

1) Online user1 wants to send message1 to user2

2) Websocket Handler checks with WebSocket Manager to identify to which WebSocket Handler user2 is connected to.

3) WebSocket Management service looks into Redis (Redis is used so that the communication happens faster (low latency).

In Redis -- We maintain key-value pairs

i.e. USER and Connected WebSocket Handler

WebSocket Handler and Connected USERs

4) Parallely, Websocket Handler communicates with Message service.

5) Message service updates the details such as sender, receiver, message, status

eg: USER1 USER2 m1 sent

6) Now, WebsocketHandler1 has information related to USER2 websocketHandler2 which is communicated from websocket management

7) Finally, WebsocketHander1 requests WH2 to get the message from cassandra for USER2

GET /USER1 / message-id

8) WH2 Fetches message from cassandra and sends it to the USER2

9) Since USER2 is connected to WH2 status in cassandra is updated to delivered.

10) upon updating status to delivered, We delete the record from cassandra

11) Message is stored in Local disk of the USERS

12) If USER is not connected to WHS, inthat case --- The message will be in cassandra until status is delivered or seen

13) If we want to keep track of sent, delivered seen timings then we can store this information in another Table of cassandra or mysql cluster

Sender	Receiver	status	Timestamp
U1	U2	sent	00:01
U1	U2	delivered	10:00
U1	U2	seen	11:01

CASE 3:

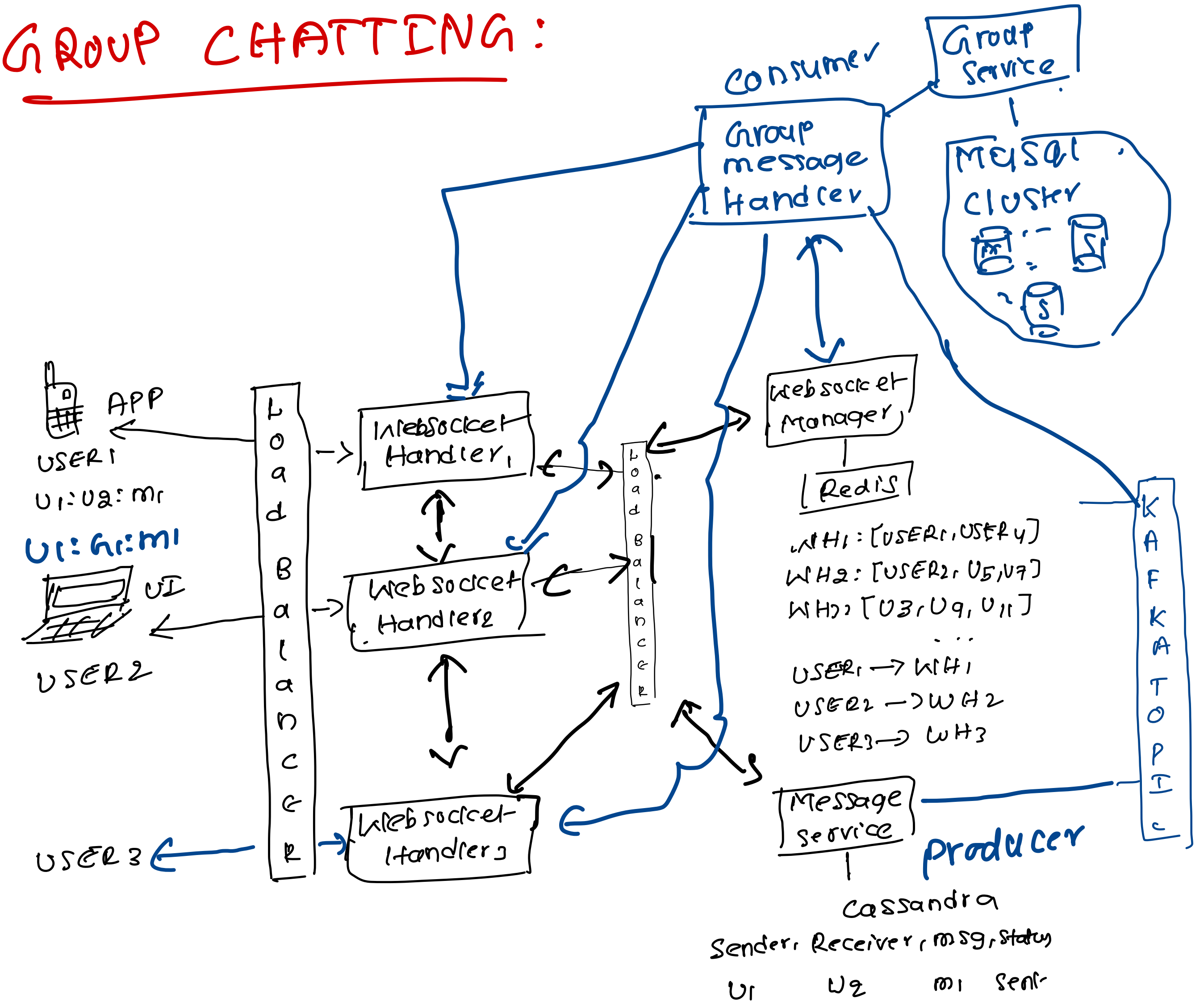
USER3 was offline or disconnected from websocket Handler.

=> once user3 reconnects -- He now communicates with message service

/GET/USER-id

(get all messages of USER3 in cassandra with status = sent

GROUP CHATTING:



Step 1: U1 wants to send m1 to Group id 1

2) Message is stored in cassandra

3) Message service pushes about group id to Kafka topic

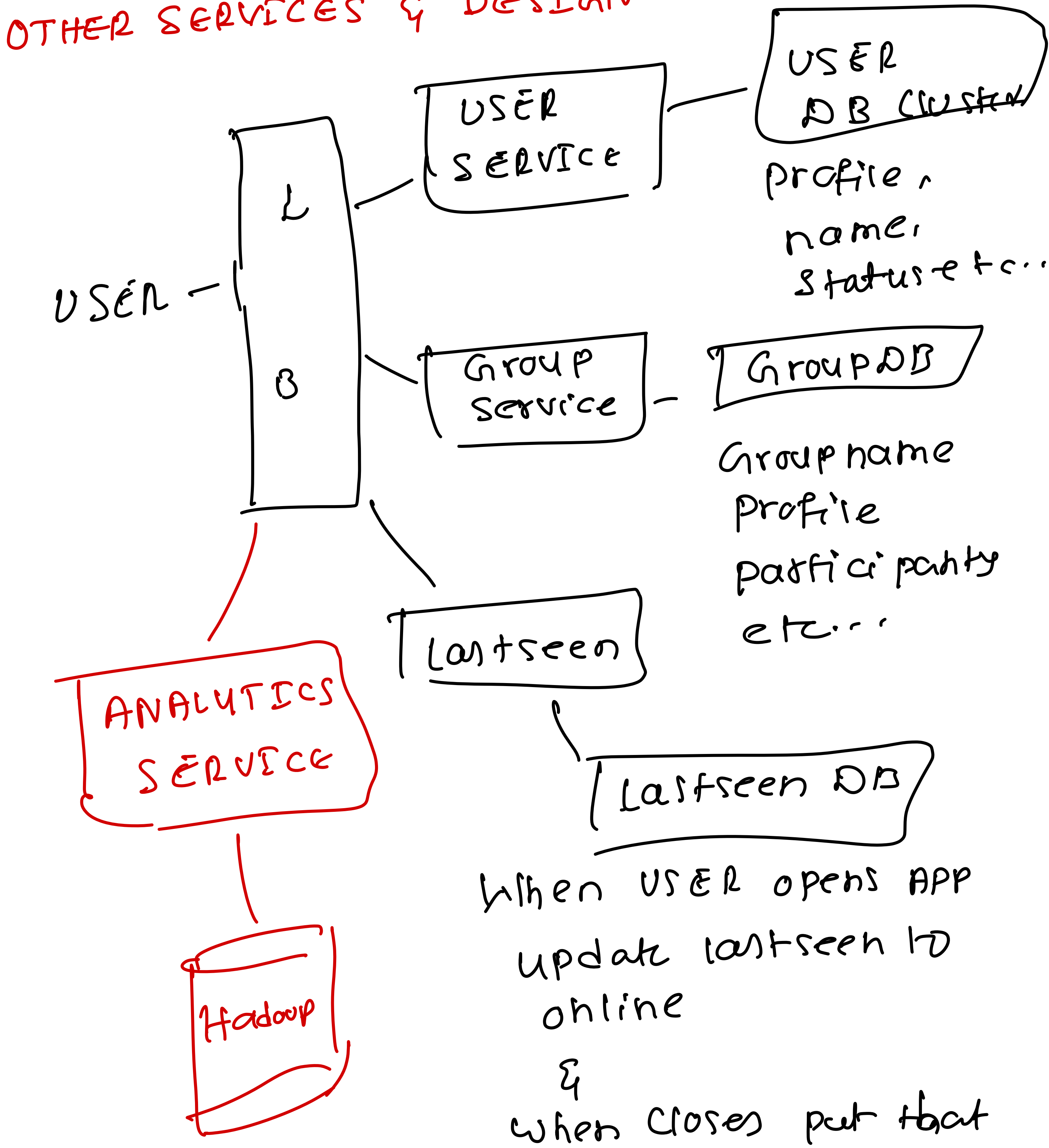
4) GroupMessage Handler now consumes group id and communicates with group service to group details like USERS etc...

5) Now, Group Message Handler gets the Websocket Handlers information of each USER from Websocket management service

6) sends the message individually to each USER

7) Deletes message from cassandra

OTHER SERVICES & DESIGN:



profile, name, status etc...

Group name profile participants etc...

When USER opens APP update lastseen to online

&

when closes put that as last seen time

ANALYTICS:

1) keep track of Users who talk on particular topics so that we can push relavant apps

2) keep track of USERs who opens a persons profile frequently ☺

METRICS:

1) keep track on no. of messages, type of messages that are communicated in a day for audit purpose.

For Audit purpose ??? Hahaha.

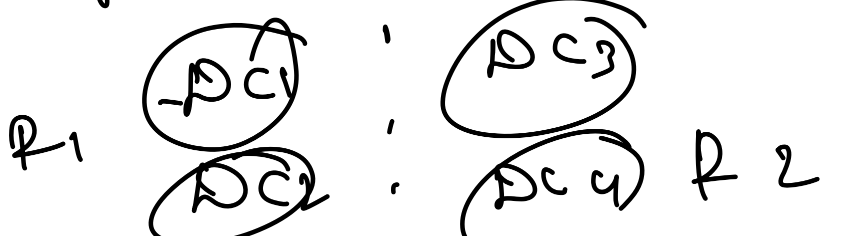
Logging & Monitoring:

Data centers:

More frequent messaging country will get primary Data center while others will get stand by DCs

or

Divide into R1, R2


DC1
DC3
R1
DC2
DC4
R2