System Design of FB Messenger

→ Idempotency

1. Req :
   - → fⁿ (MVP)
   - → Non fⁿ
2. Estimation of Scale
3. APIs
4. Design
5. Follow up (if any)

---

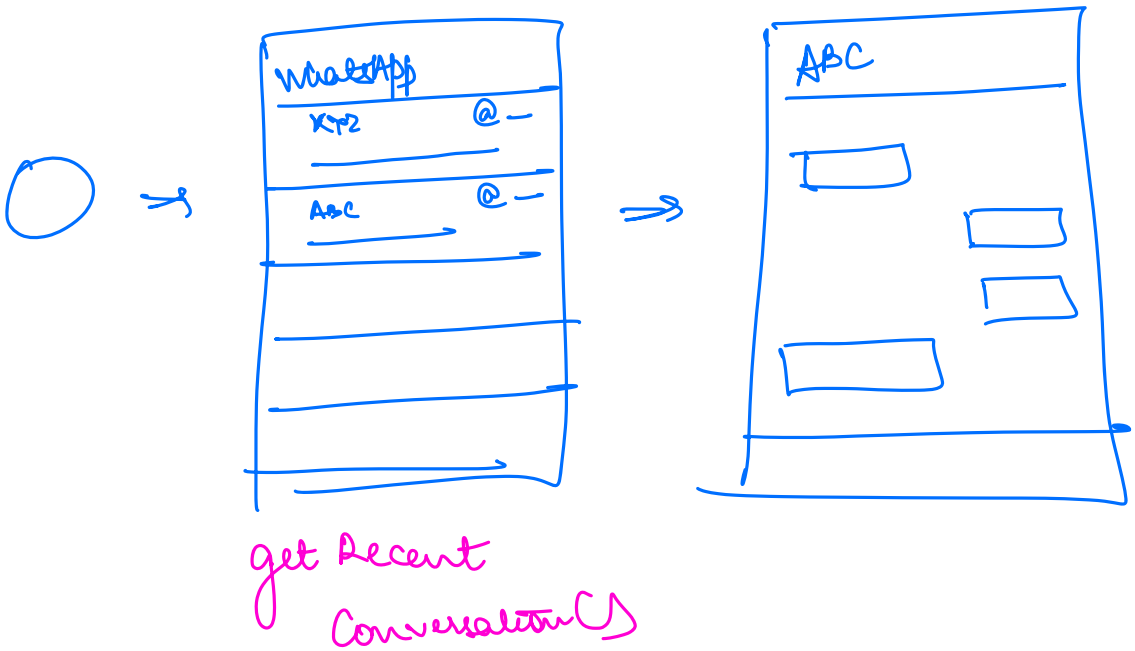
1. Requirement Gathering

fⁿ Req

- → Ability to send and receive message
  - → only 1:1 chats
  - → support message history.
  - → get recent conversation
  - → only text messages.

Other Features

1. Online Indicator
2. Multimedia message
3. Read/Delivered Indicator
4. Last online at
5. Read/Unread status



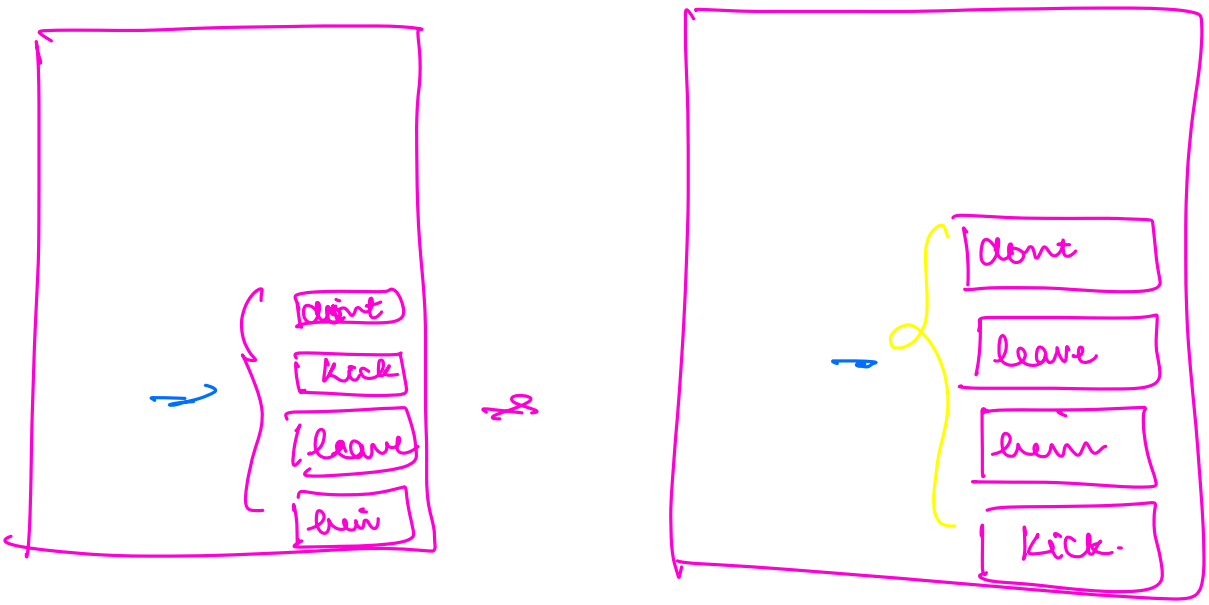
get Recent Conversations

Non f^n Req (Design Goals)

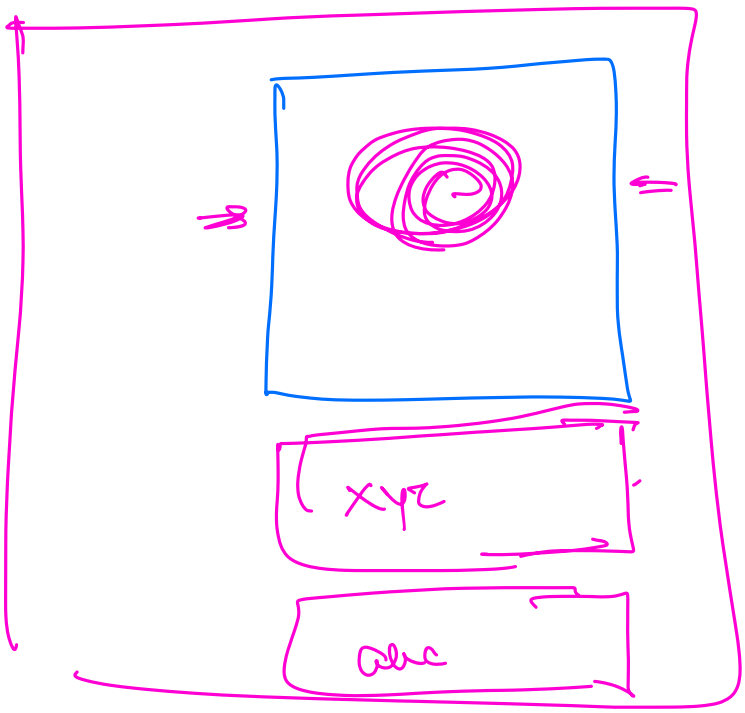
1. Consistency V/S Availability
2. Consistency V/S Latency

Consistency

1. Order of Delivery.



Keep as low latency as possible while keeping my system consistent



Estimation of Scale

- Size of Storage
- Req/ Sec
- Read v/s Write Heavy
- Both

Do I need a sharding

Yes

Obviously yes

50B

users ⇒ 1B

Avg message / user ⇒ 50

⇒ 50B messages / day

Size of message :

- id ⇒ 8B
- sender_id ⇒ 8B
- receiver_id ⇒ 8B
- time ⇒ 8B ⇒ epoch time # milliseconds since 1 jan 1970 GMT+0:0 (long)
- content ⇒ 200B

≈ 250B

Hi
Hello
How r u.

200 char

⇒ $50 \times 10^9 \times 250$ B / day

⇒ $50 \times 10^9 \times 250 \times 365$ B / year.

⇒ $50 \times 10^9 \times 250 \times 365 \times 5$ B

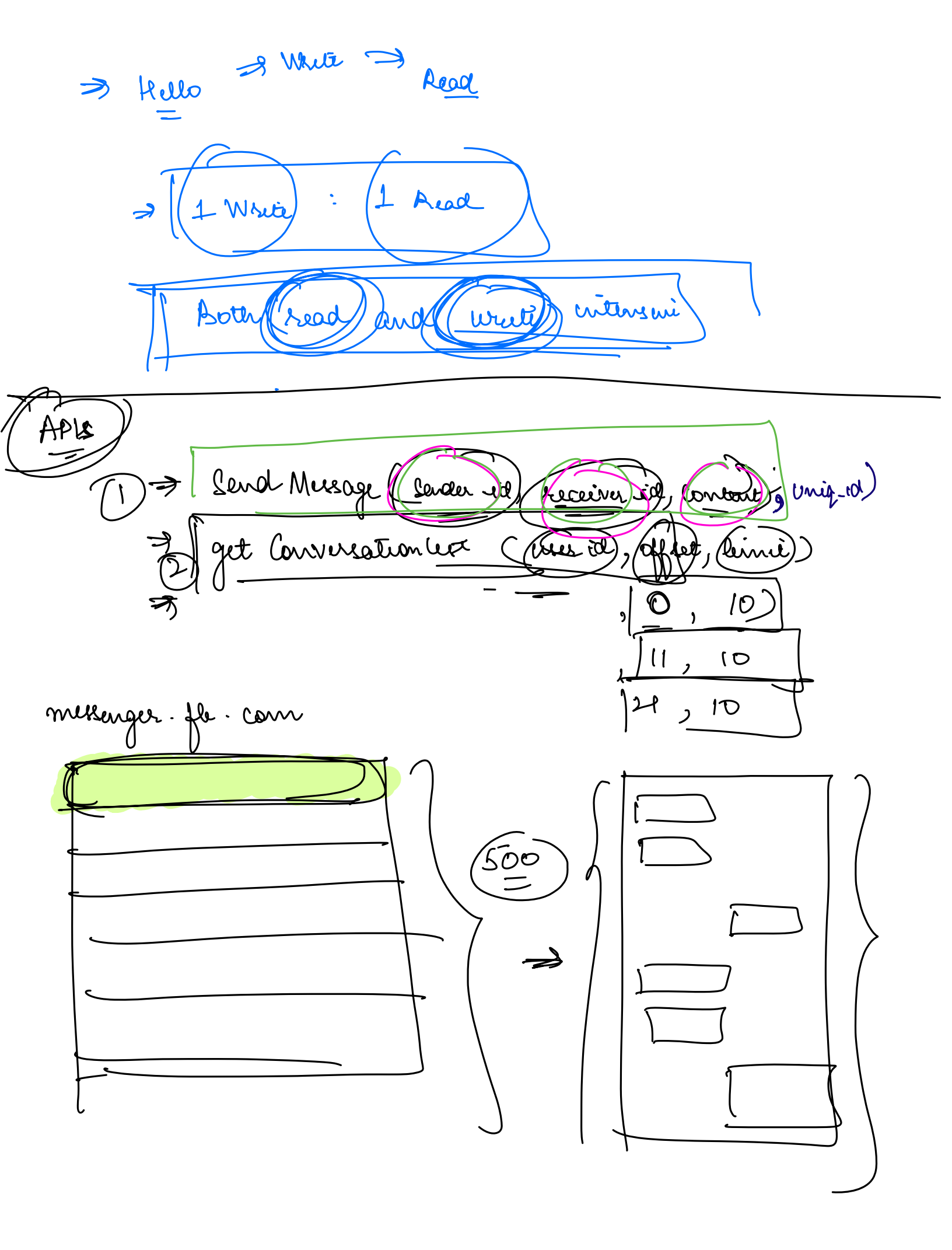
5 year

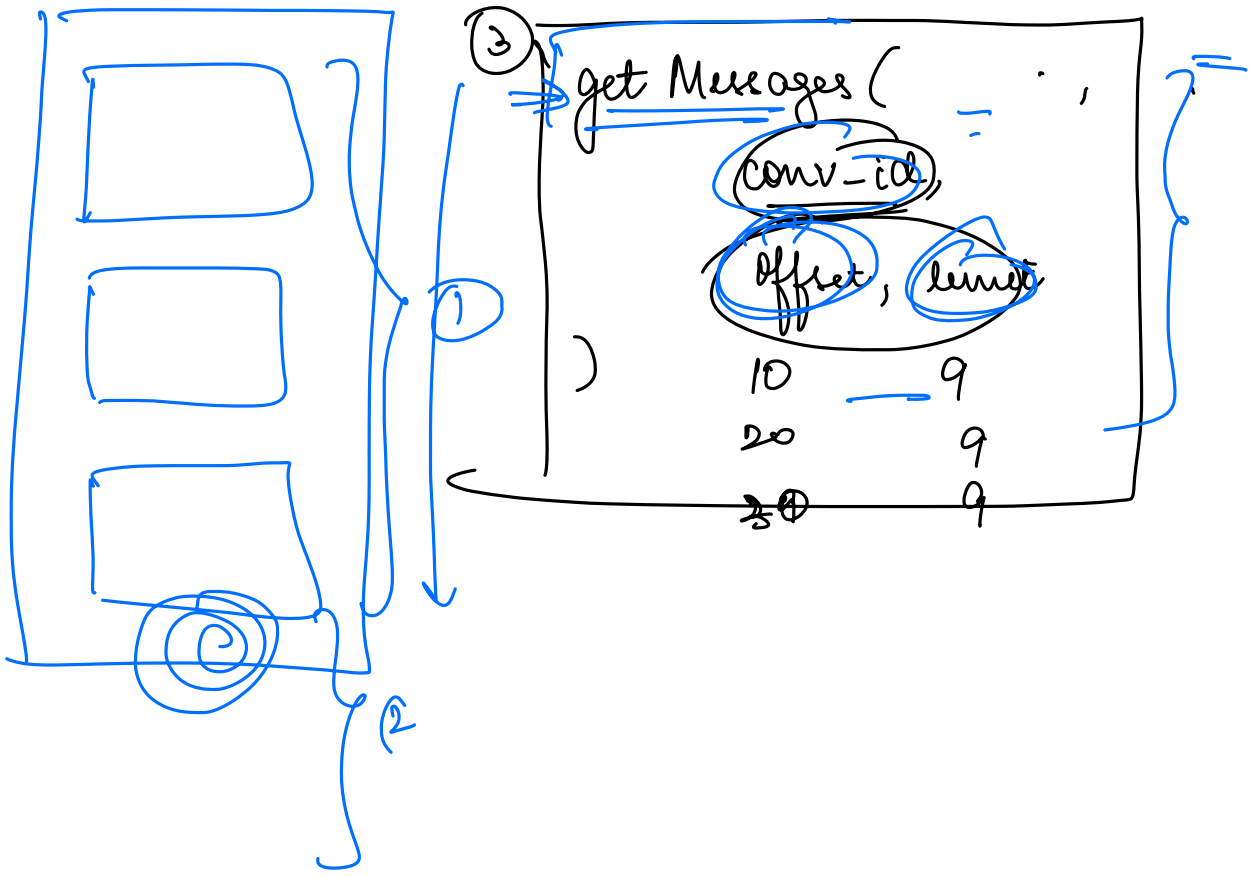
⇒ $2000 \times 50 \times 250 \times 10^9$ B

⇒ 100000×250 GB

⇒ 100×250 TB

⇒ 25 PB


Hello
Write
Read
1 Write : 1 Read
Both read and write intensive
APIs
1
Send Message (Sender id, receiver id, content, uniq_id)
2
get Conversation (user id, offset, limit)
0, 10
11, 10
21, 10
messenger.fb.com
500



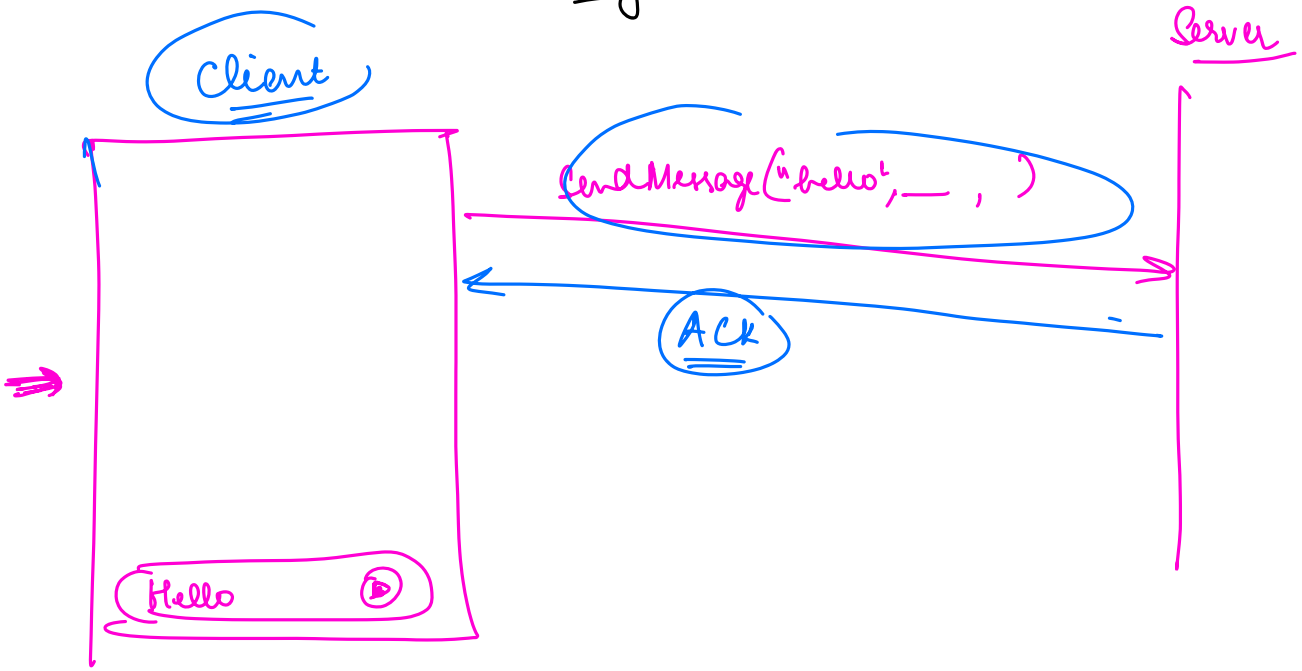
convs = [{name: "Asif", lastAt: "5minsago", convo ... }
{name: "naveen", lastAt: "8minsago" ... }

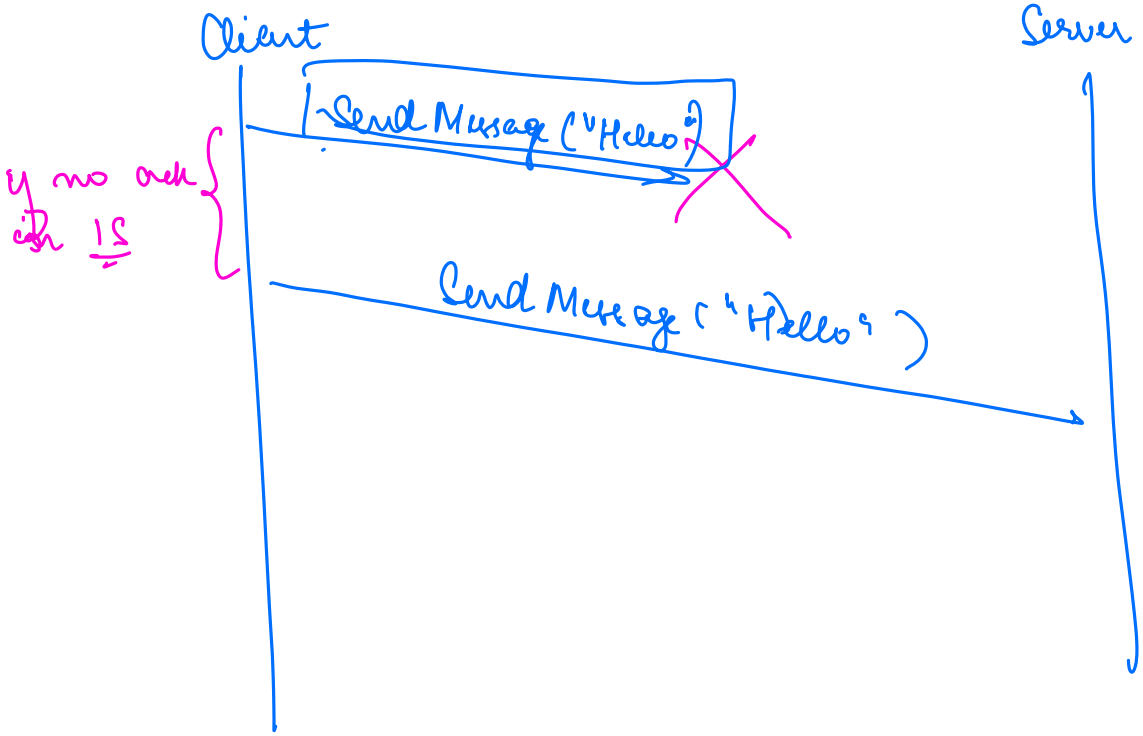
]

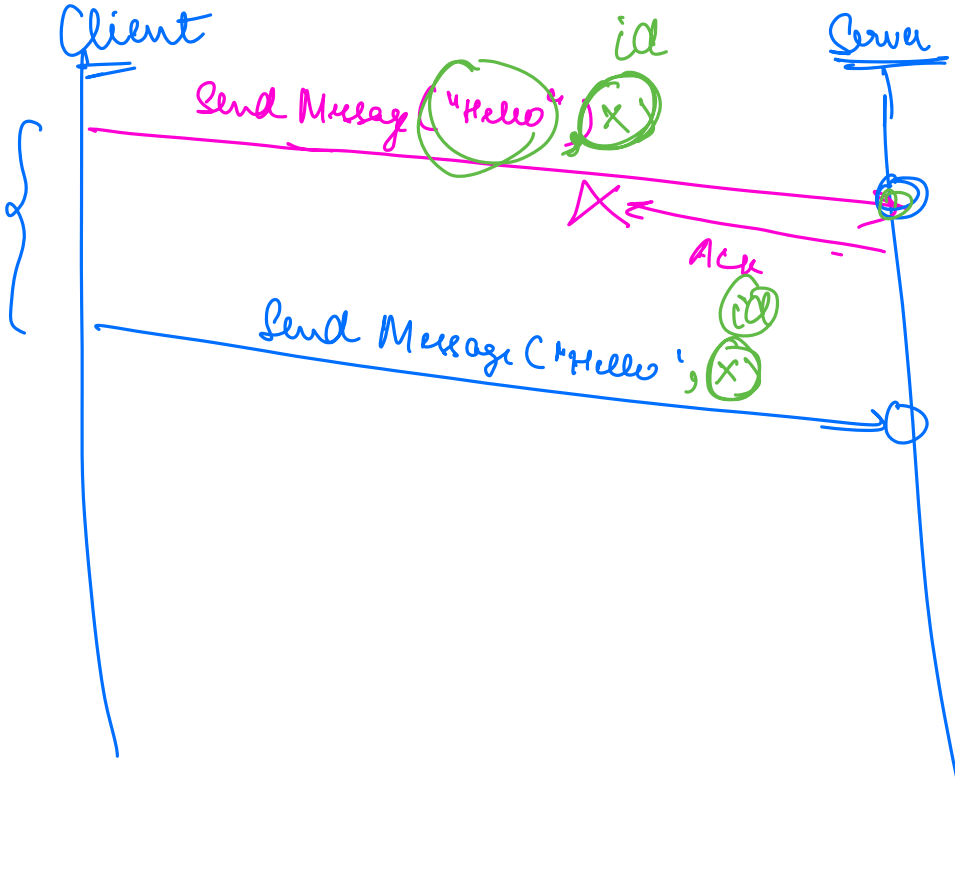
Design

How TCP works

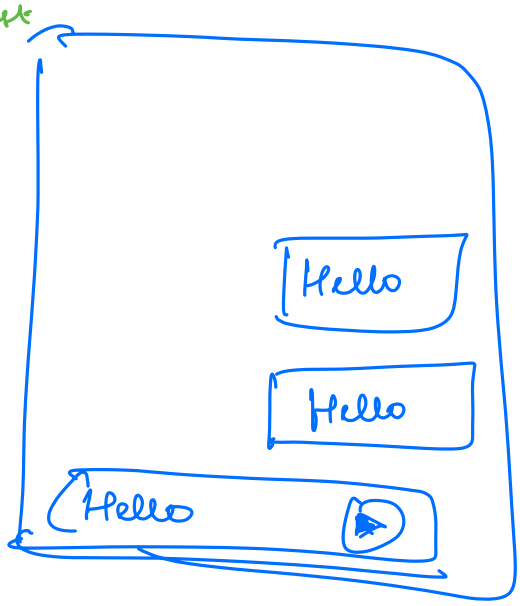
→ Idempotency



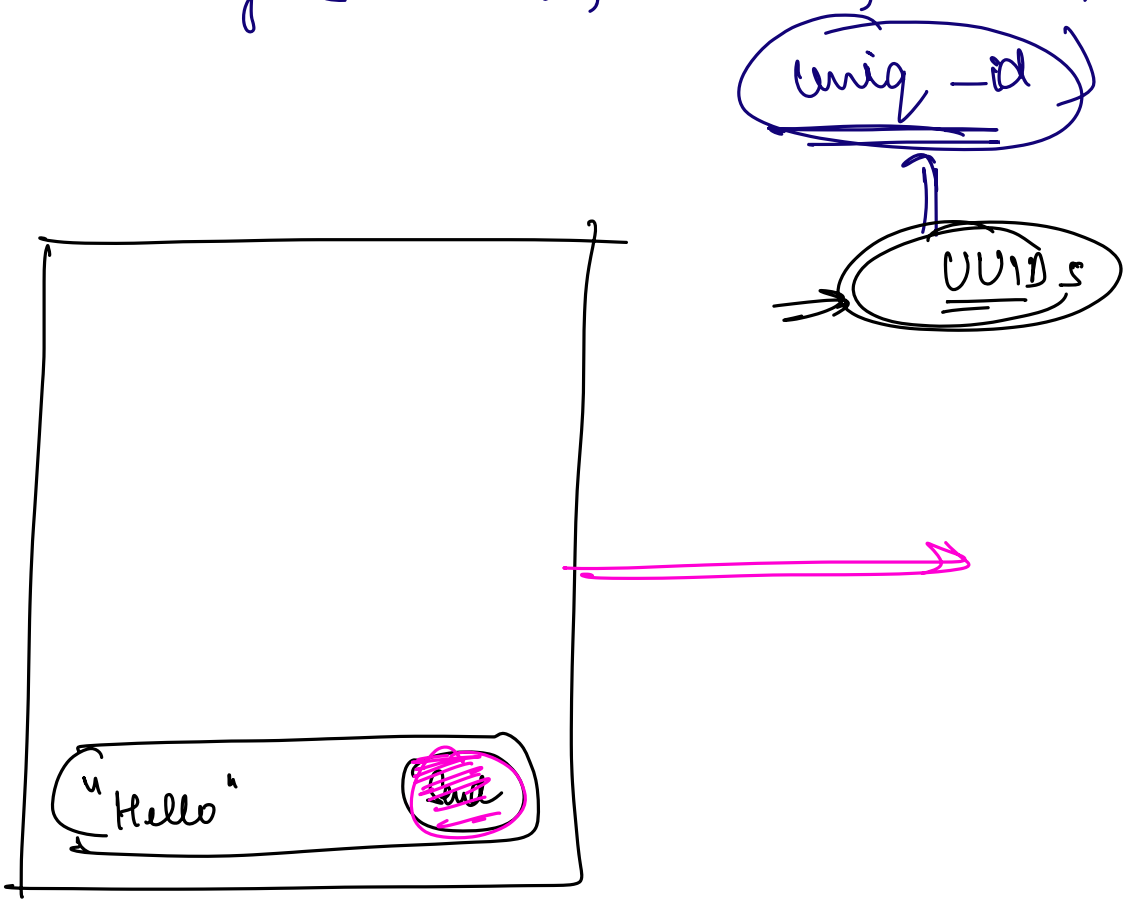


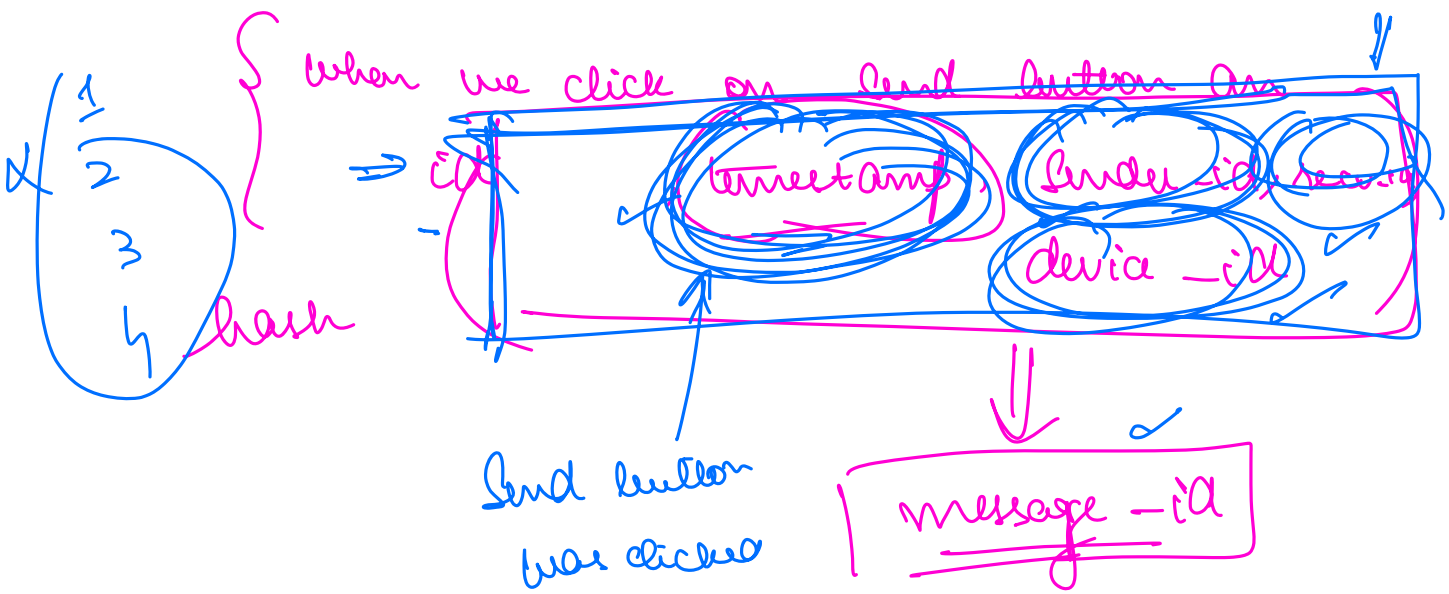


⇒ We need a way for server to be able to reject a duplicate request



Send Message (sender_id, recv_id, content,





idempotency ⇒ uniquely identify a req to ensure no duplicate processing

send ( ) {

ID = hash (timestamp, device_id, sender_id, recv_id)

sendMessage (content, sender_id, recv_id, id)

id → only to help server uniquely identify seq.

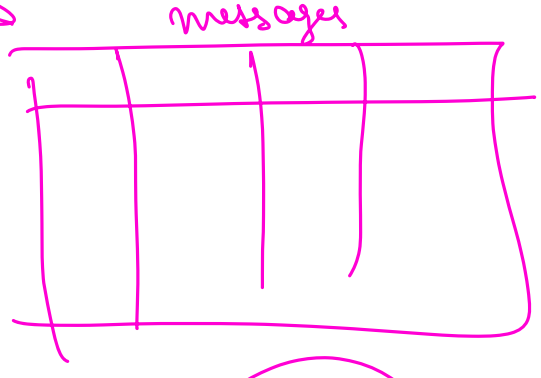
2

1

getMessage ( )

getConversationList ( )

sendMessage ( )

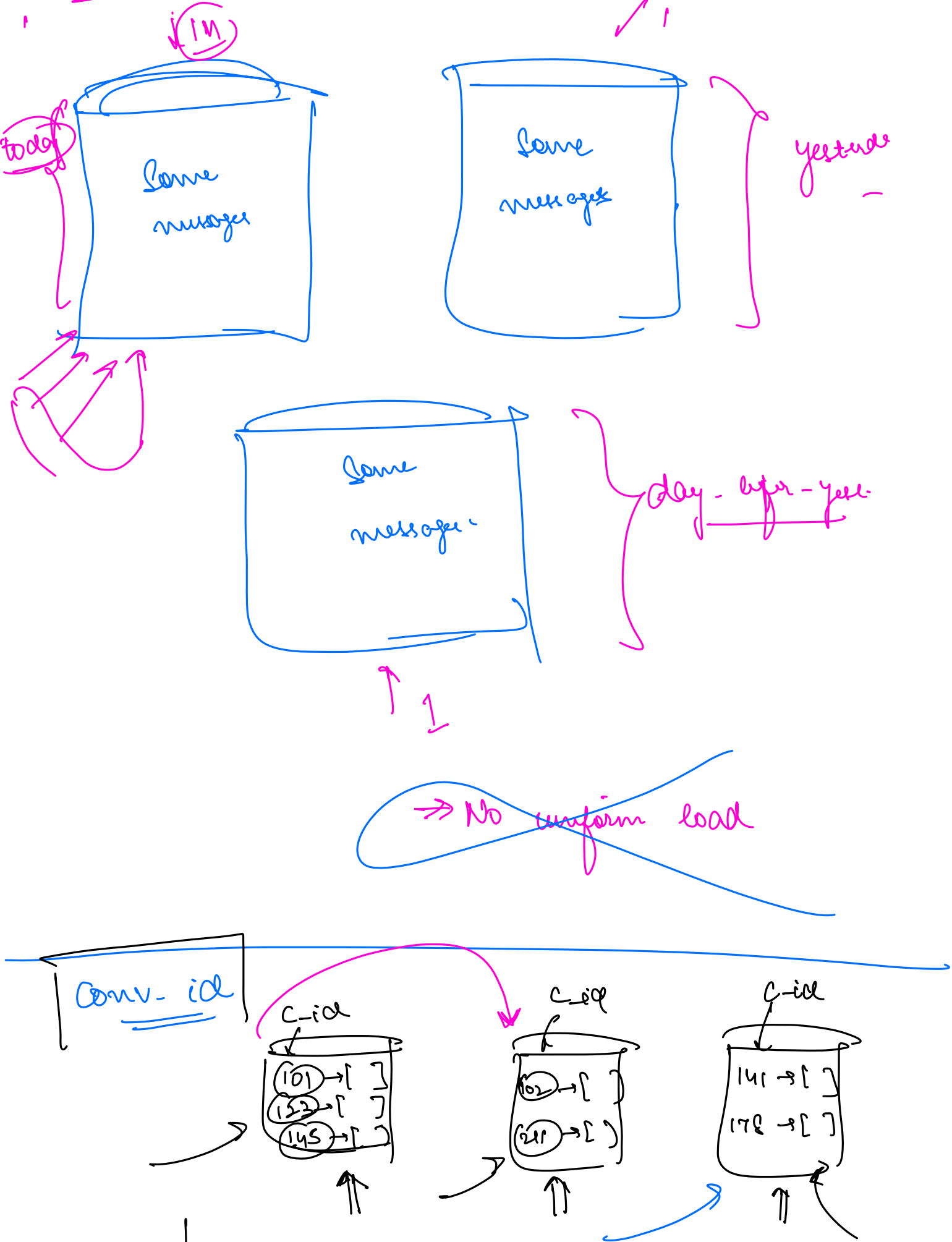


≈ 25 PB

⇒ Shard

↳ Sharding Key?

Timestamp

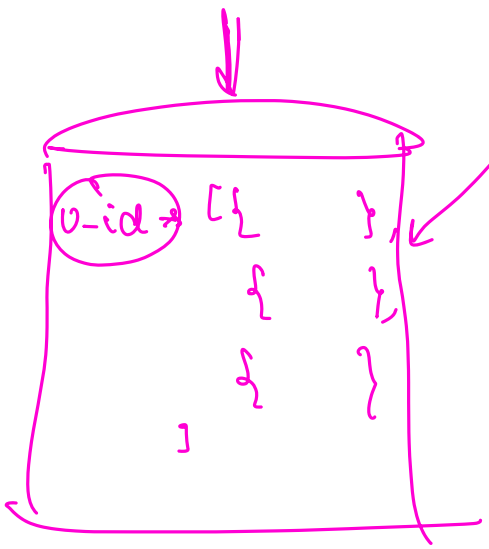

today
Some messages
yesterday
Some messages
Some messages
day - before - yesterday
⇒ No uniform load
Conv - id
C_id
101
132
145
C_id
102
211
C_id
141
178
① Send Message ( ) ⇒ write to machine that holds

messages of that conv.

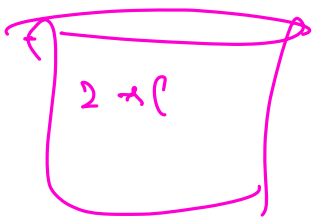
(2) getMessages( ) → read from 1 machine and you get all messages in that

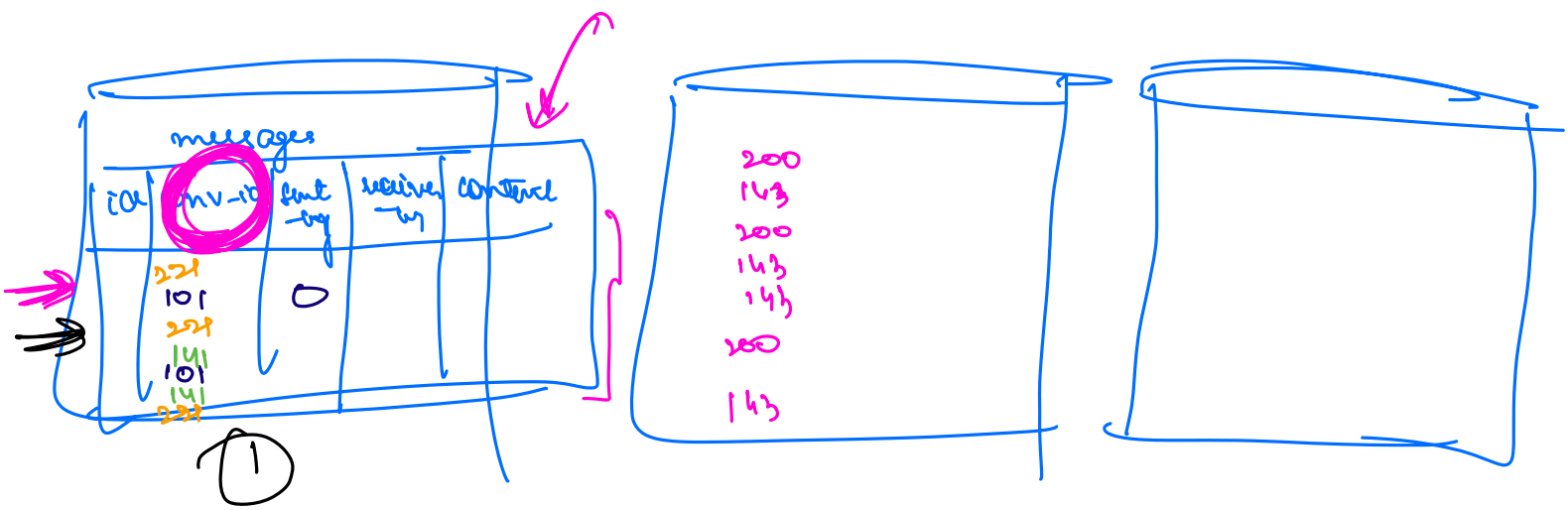
(3) getConvList( ) → query multiple machines → :(

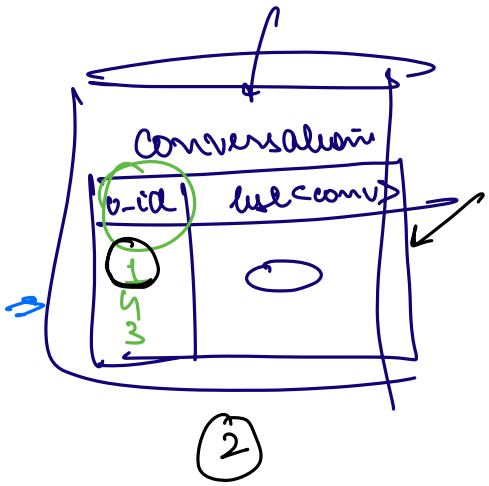
Soln: Create a cp table which stores for all user, all of their conv

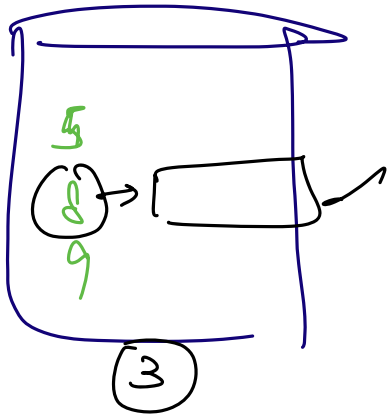


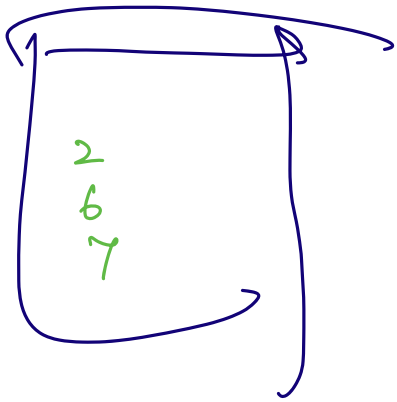
for a u-id, all their conv, and the latest message of that conv, in sorted order

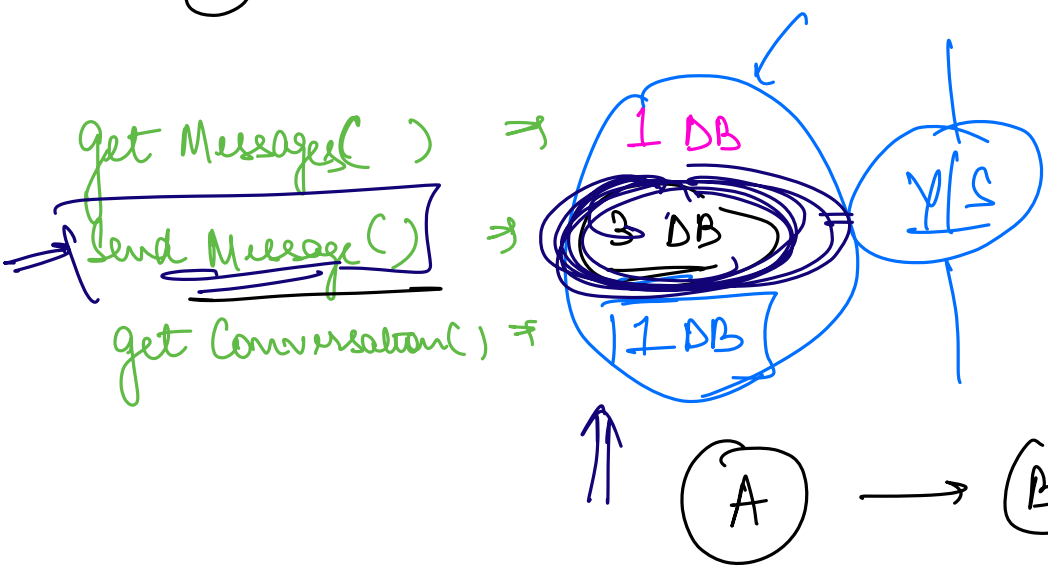


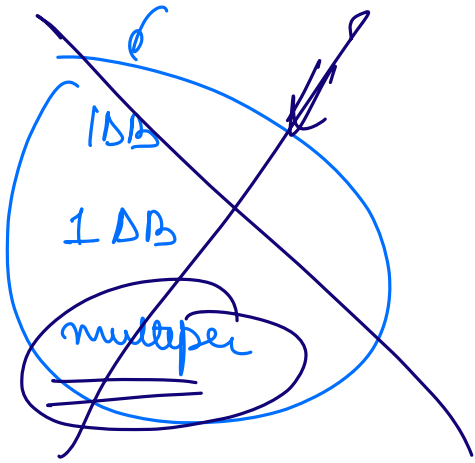



Conversation
b_id
list<conv>
1
4
3
2

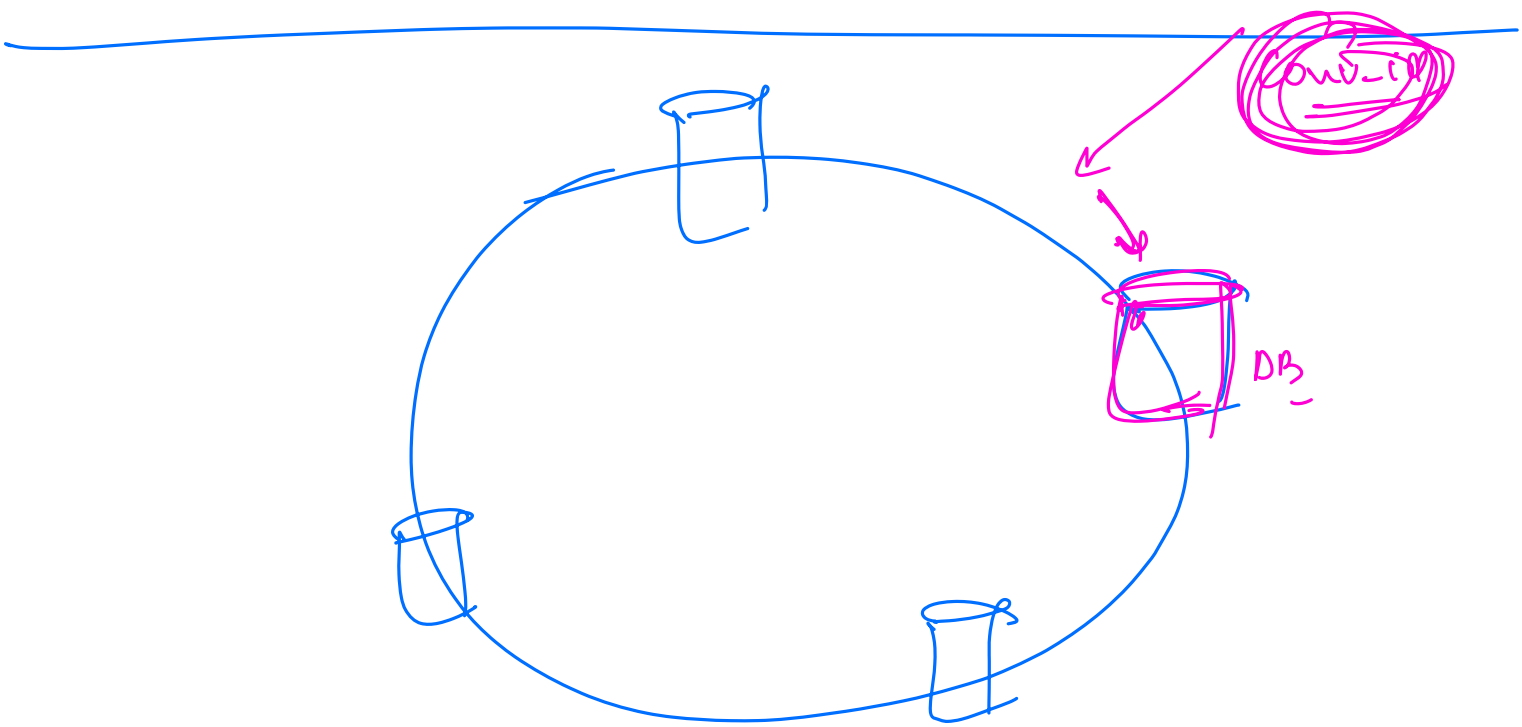

5
8
9
3


2
6
7


get Message( )
Send Message( )
get Conversation( )
1 DB
3 DB
1 DB
YES

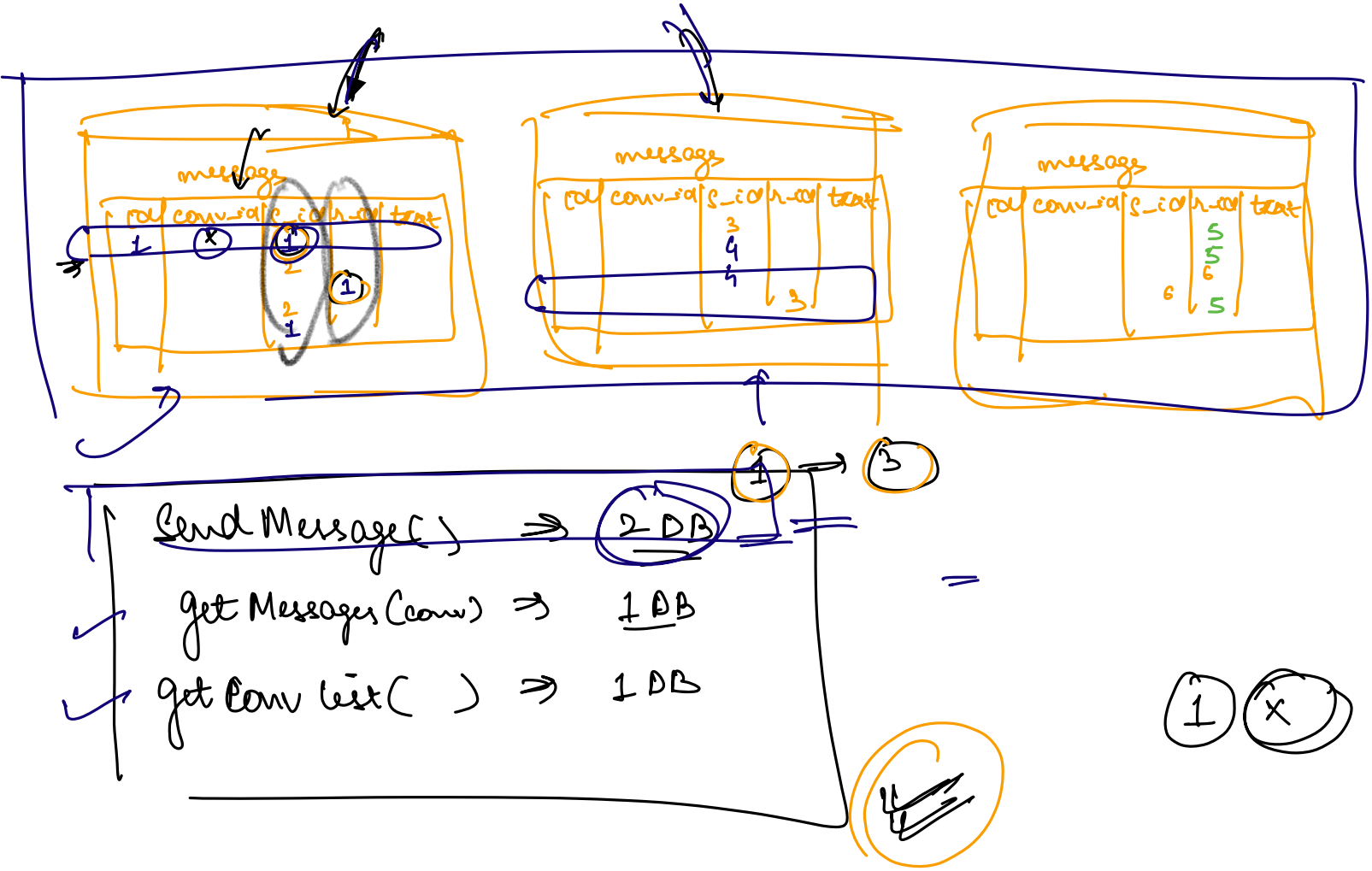

IDB
1 DB
multiple

A
B


DB

Y|S Shard messages by user_id

↳ all messages for that user (sent by them) as well as received by them are in one machine.



Ordering (A)(B) =

Sender_id + Conv_id

{ hash(1, 101) → X
  hash(2, 101) → Y }

1250 years

250B → 10TB

10^{12} B / 250B

→ $1000^4 \times 10^9$ B / 250B

→ 4×10^9

$\frac{4 \times 10^9}{60 \times 60 \times 24}$

→ For messages we will shard by user-id.

Send Message (sender_id, recv_id, content, idm_key) {

if (reqHistory.contains(idm_key) return;

→ db1 = conn(sender_id)

→ db2 = conn(recv_id)

⇒ db1.execute(——————)

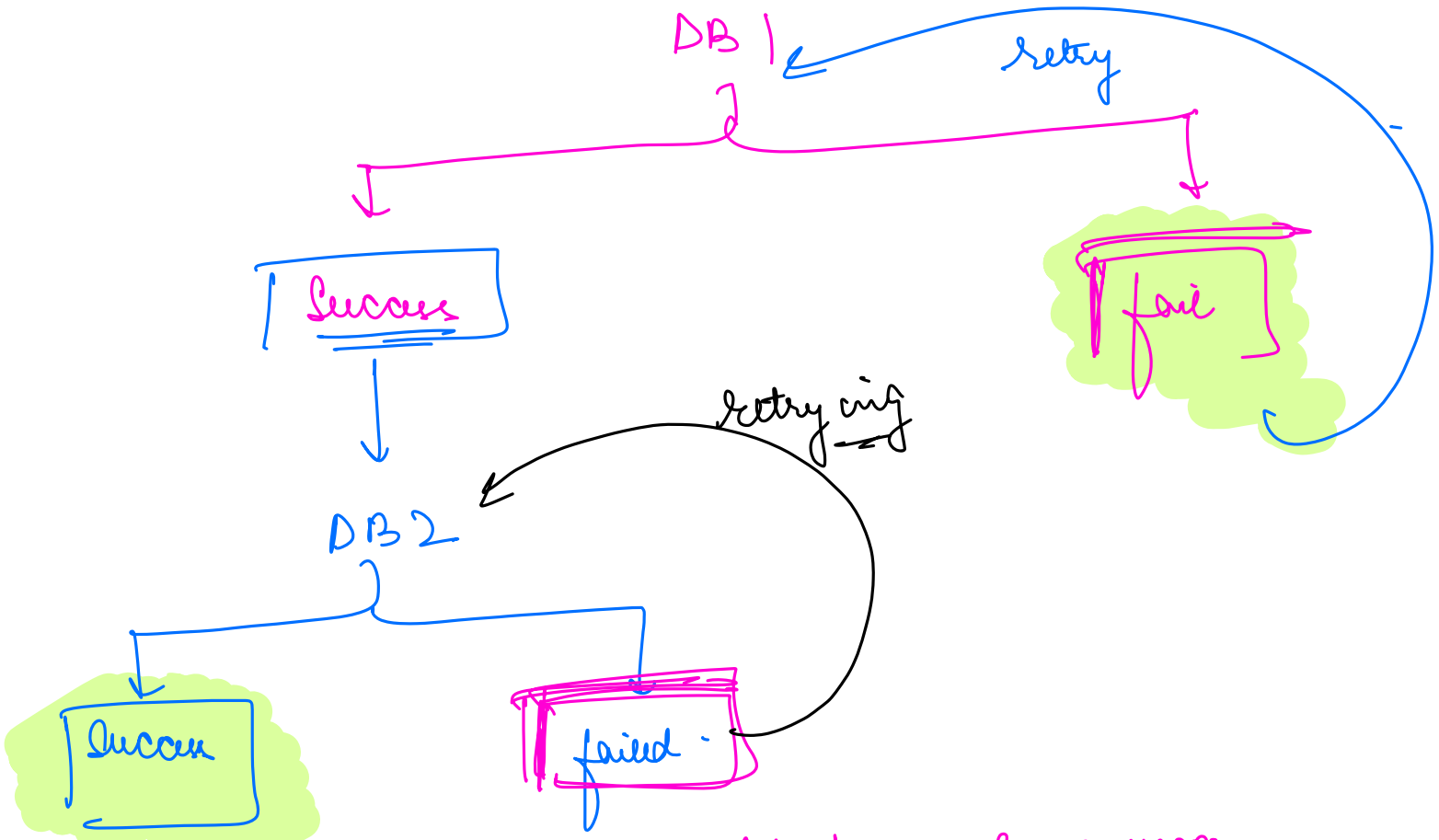
db2.execute(——————)

return;

}

How to write to 2 DB

(1) Write to DB1 first and then to DB2



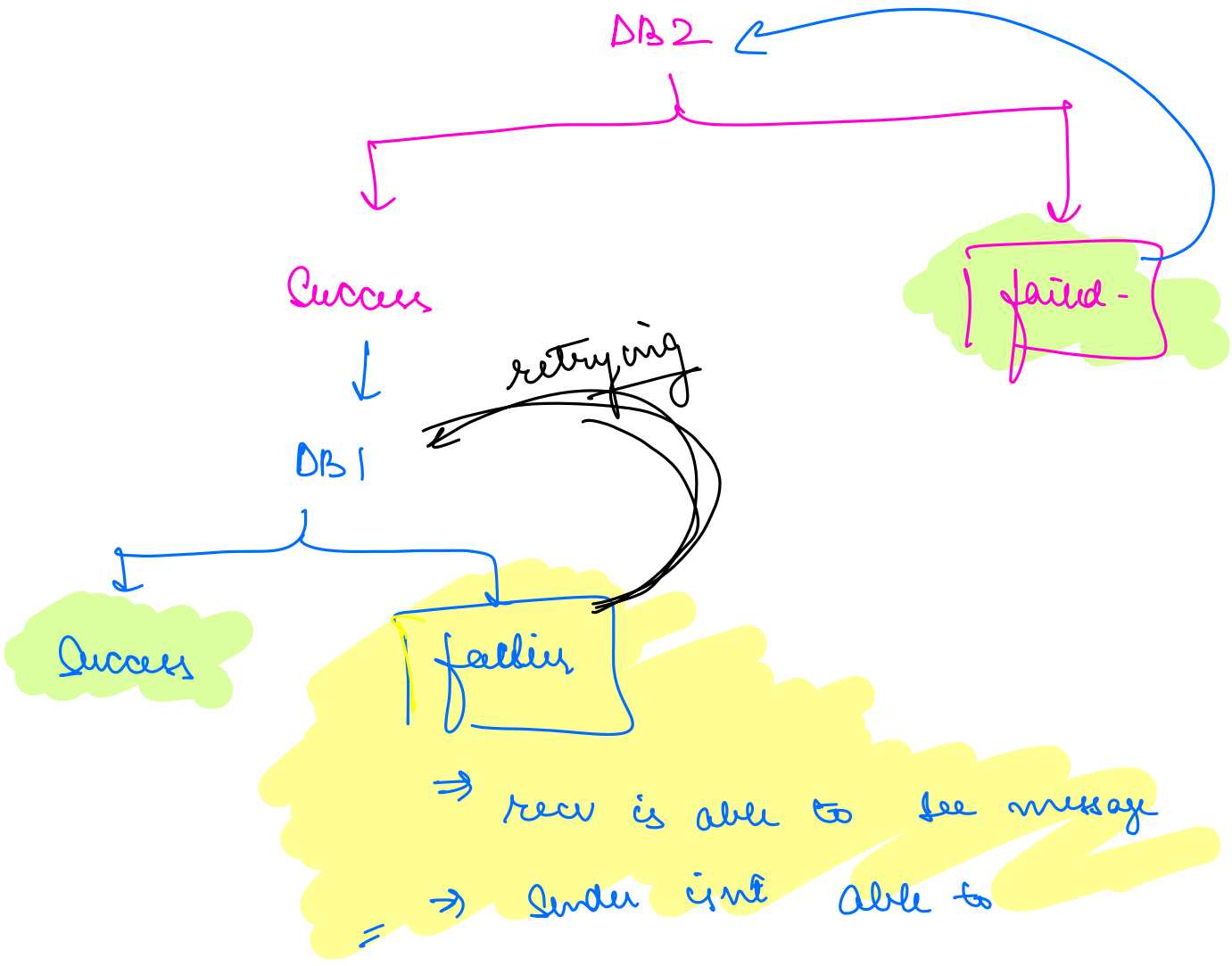
user1 can see message

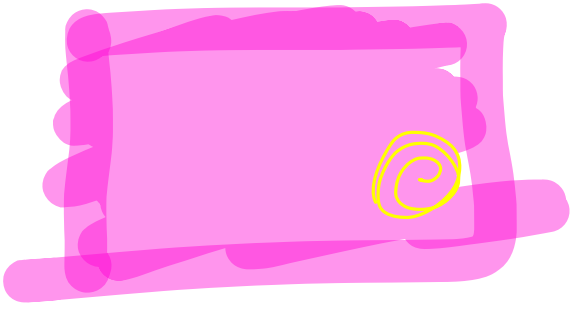
but user2 can't

BAD

(2) Write to DB2 and then to DB1

Better





1. Choosing DB for FB Messenger ⇒ How to handle read: write
2. Bloom filters
3. Doubt Resolution

```
getMessages(conv_id, user_id, limit, offset)
    db = conn(user_id)
    db.execute(
        Select distinct message_id, content
        where conv_id = conv_id
        limit x offset y
    )
}
```