Design Whatsapp (WA)

FEATURES

0) One to one chat
1) Group Mesgin (around 256 ppl / grp)
2) Sent + Delivered + Read recipts
3) Img sharin (covered in Tinder design)
4) Online / last seen
5) Chats are temporary / permanent

WORKFLOW

(1) Connect to WA on the cloud via a gateway

Reason :- you might be usin a diff protocol (to talk to WA) than the protocol WA uses to talk to its sys. comps internally

HTTP over TCP

websockets (WS)

Reason → you don't need that much security or big headers that HTTP provides you when talkin internally

Gateway takes care of most of the security mech.

(2) Connect user's via gateway boxes (If A is connected to Box 1, B is connected to Box 1, A can talk to B

Problem: Currently our gateway is already stateful as we store mem. to maintain TCP connec"s :. we shudn't include data of who's connected to which box here
Plus, if we store this data, we gotta replicate it in other gateways and :. this data is transient
gotta do a lotta syncin → our sys. is too "coupled"

Date:
M T W T F S S

Soln : Use a separate 'session service' to store who's connected to which box. Now, sys has been "decoupled"

Gateway → Stores and maintains more TCP connec"s (dumb connec"s → dunno what to do with a req → just feed it to sessions service)

Session's Service → Stores which user is connected to which box
→ Makes this data accessible to all gateways.
→ has Multip-servers for redundancy

(3) A sends a msg to B —HTTP (client→server)→ recd by gateway A is connected to → feeds to session's service → figures out which gateway B is connected to → feeds msg to gateway B is connected to ←WS (client←server)— B receives msg

Coulda also used "Long Polling" in HTTP instead of WebSockets (WS) but WS is more real time

WS → allow P2P comm" i.e client can send to server and server can also send to client

Read, sent & Delivered Receipts

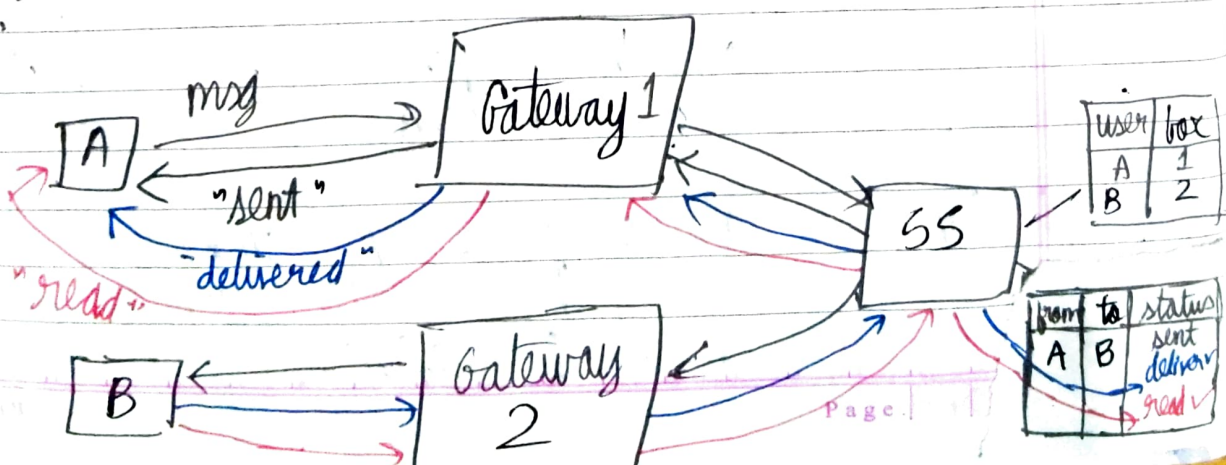
Date: M T W T F S S

Parallely, when session's service (SS) is fwdin msg from A → B to gateway of B, it also has another dB in which it stores the status of this msg

(1) When the msg from A is successfully recd by SS frm A, it sends back a msg to A (sent receipt → single tick on WhatsApp) sayin it has recd the msg and its now gonna fwd it to B. Msg status dB is updated

(2) When B receives the msg fwded to it by SS it sends back a res. sayin it recd the msg frm A
Our msg status is updated (msg is now at delivered → (double tick on WhatsApp) stage)

(3) When B opens and logs into WhatsApp automatically and marks a msg as read, then, automatically a res is sent frm B to SS sayin msg was read → double BLUE tick on WhatsApp)
Our msg status dB is updated & SS sends a msg to A that msg sent to B has been read.



Date:
M T W T F S S

Last Seen / Online

Basically a client sends to gateways 2 kinda reqs

1. User generated → sendin, rel ; texts
   generate "read" receipts
2. Sys. generated → receivin texts frm server
   everytime your ~~network~~ internet connee" is switched on
   → generatin "delivered" receipts and sendin to ss once msgs are received (not read yet)

i) ∴ Wes introduce a new service → Last Seen service which has access to a DB that logs user activity. It stores
(1) user (2) timestamp of latest activity
DB → a time-series DB

ii) Anytime a user generates a req to WhatsApp servers (to read, send, delete, star msgs, block ppl, set status etc.), an "activity is performed and thus the timestamp ~~in~~ in the DB gets updated

iii) Thus, when another user wants to know if a user is online or not, "last seen services" fetches this data

iv) IMP :- "activities" only entail user-generated reqs. & not sys-generated reqs

Last Seen / Online

Basically a client sends 2 kinda reqs to gateways

1. User generated → sendin, recvin texts generate "read" receipts
2. Sys. generated → receivin texts from server anytime your internet connec" is switched on
   → generatin "delivered" receipts and sendin to SS once msgs are received (not read yet)

i) ∴ when introduce a new service - Last Seen service which has access to a DB that logs user activity. It stores (1) user (2) timestamp of latest activity → a time-series DB

ii) Anytime a user generates a req to WhatsApp servers (to read, send, delete, store msgs, block ppl, set status etc.) & activity is performed and, thus, the timestamp in the DB gets updated

iii) Thus, when another user wants to know if a user is online or not, "last seen service" fetches this data

iv) IMP :- "activities" only entail user-generated reqs. & not sys-generated reqs

Grp Msgin

1) Now we introduce a separate service called a "Grp service" (GS) that our SS talks to for grp msgin f"ality. Group service has a dB that maps 1 grp id → many user IDs

2) ∴ When an SS gets a msg frm a user it asks the GS that → "who are the other members in this grp?"
Then, once it receives the members' userID SS uses its own dB to map which gateway/box it needs to frwd msg to.

3) Most chat apps tryna limit no. of members in a grp else you gonna be fannin out way too many reqs. to keep track of in an encrypted, authorized manner in real-time.

One soln → batch process these reqs.
→ wait for users to pull the msgs (not real-time tho)

4) Parser service → lies b/w a box and SS

Parsin → convert an electronic msg into a sensible prog ing lang. obj

→ unparsed msg is encrypted, auth'ed for sent msgs

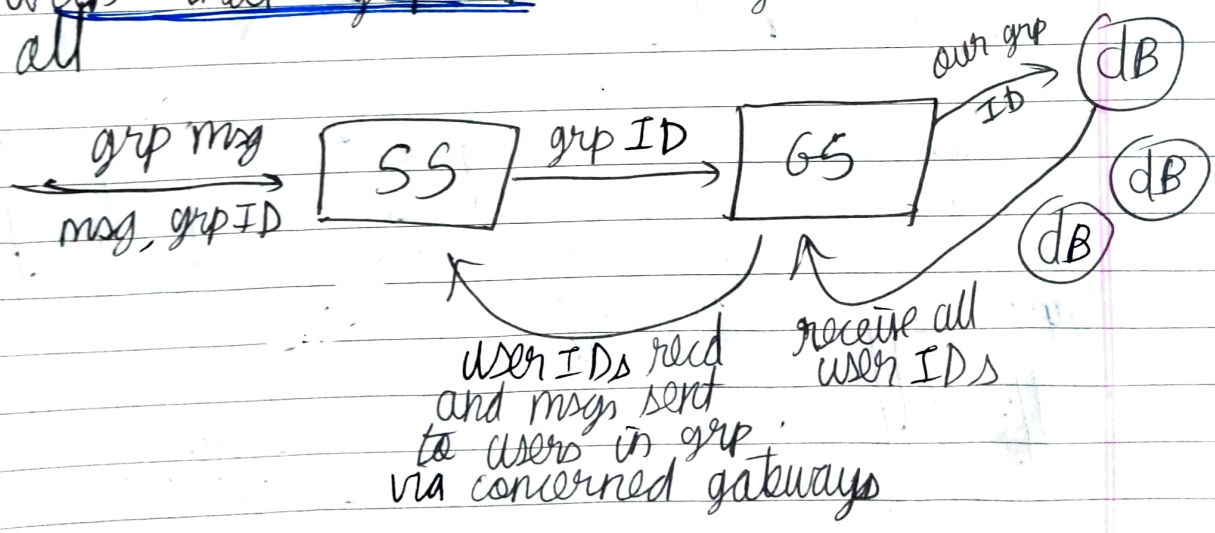
→ parsed msg is decrypted, authed. for delivered msgs to be delivered

5) Consistent Hashing → helps reduce memory footprint across servers by delegating only some info to some ...

→ helps route reqs. to concerned bores. Routing is based on grp ID to tell users belonging to this grp.

Basically, each grp msg has grp IDs attached and, on receiving a msg pertaining to a partic grp ID, SS routes its req. (to grp service) based on grp ID and when grp service receives this msg, it accesses its dB associated with that grp ID → gets user IDs of all

got via consistent hashing.



6) Message Queues → a part of our SS which ensures that msg will be sent.

→ Needed to retry sending a failed msg

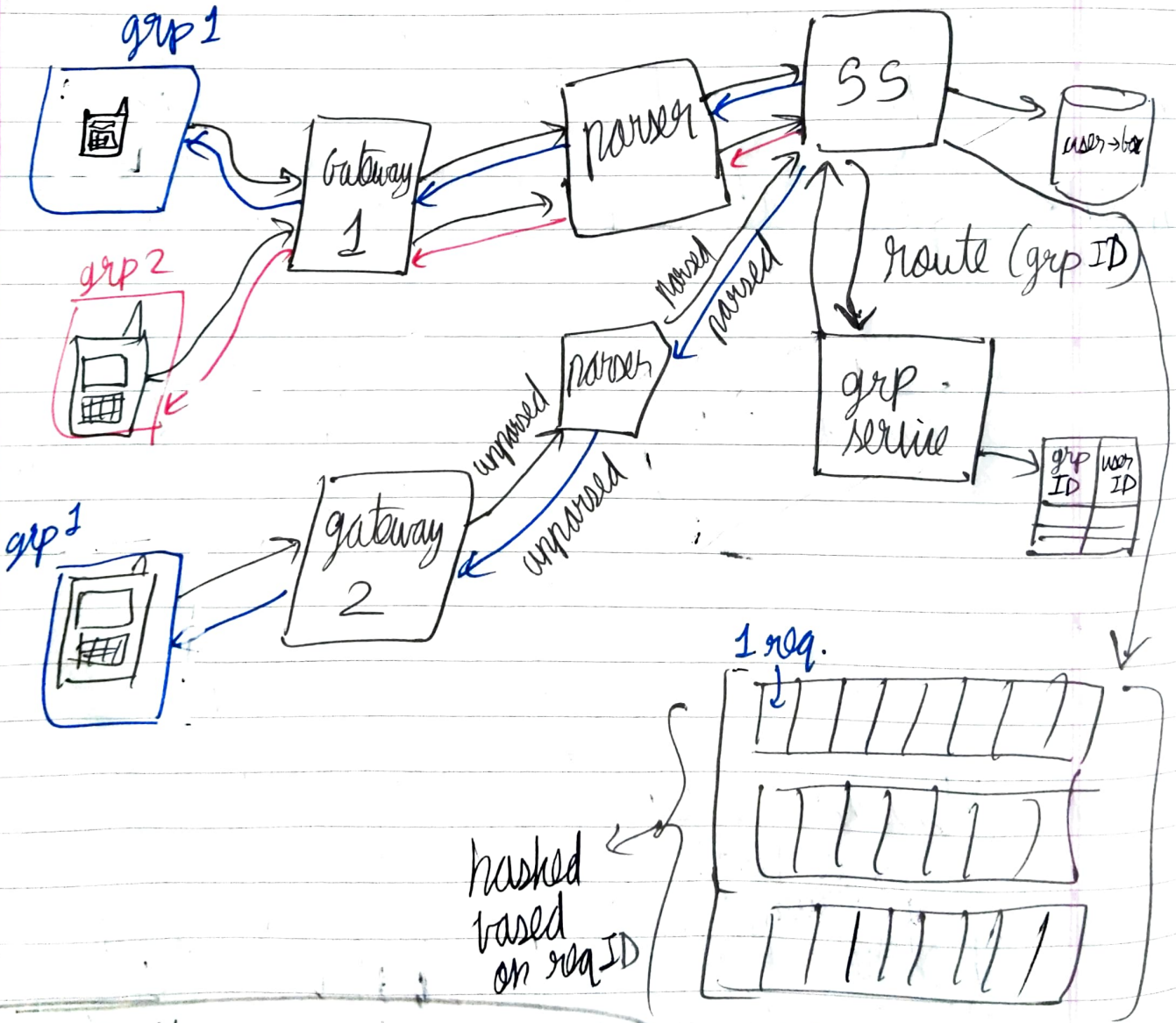
→ helps us keep track of upto what pt msgs were successfully sent

→ ∴ knows from what req. it needs to retry sending failed msgs

NOTE : Grp receipts (sent, read, delivered) are pretty expensive to manage since every connected grp member needs to +vely respond

NOTE : Deprioritizing reqs → During time periods of excessive load on servers, Facebook just serves reqs pertaining to sending, receiving msgs and all other op's such as read receipts are suspended. (rate-ltd.)

eg:- festivals



NOTE : Auth. services and LBs are also introduced in sys.