**DESIGN A CHAT SYSTEM**

Before design the system clears the some of the questions from the interviewer.

Candidate: Is this a web app or mobile app

Interviewer: both

Candidate: What kind of app this is? 1-1 or group.

Interview: It should support both.

Candidate: What is the scale of this app?

Interviewer: It should support 50M DAU (Daily active user).

Candidate: For group chat what is the group limit.

Interviewer: 100 users.

Candidate: What are the important features.

Interviewer: 1-1 chat, group chat, indicator. The system only support text.

Candidate: How long should you store the message history.

Interviewer: forever

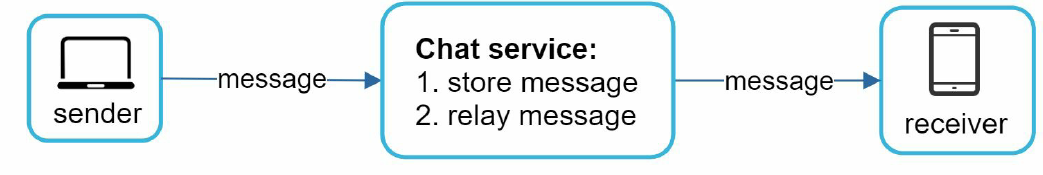
Candidate: What is the text message length.

Interviewer: 100,000 long.

Candidate: Is end to end encryption required.

Interviewer: Not required now, we can discuss if time permits.

**High level Design:**



Let’s consider the protocols used. Let’s use HTTP. time-tested HTTP used with a keep-alive header allows the client with a persistent connection with a chart service.

This is ok for the sender side. Http protocol can’t send message for the server.

Server-initiated connection: polling, long polling, and WebSocket.

Pooling: is not a good solution as it costly because checking status frequently.

Long Polling: If it is a distributed system then the sender and the receiver may not connect to the same server.

WebSocket: This is the best solution. It uses 80 and 443 ports and can be operated on firewall and also bidirectional. By using WebSocket for both sending and receiving, it simplifies the design and makes implementation on both client and server more straightforward.

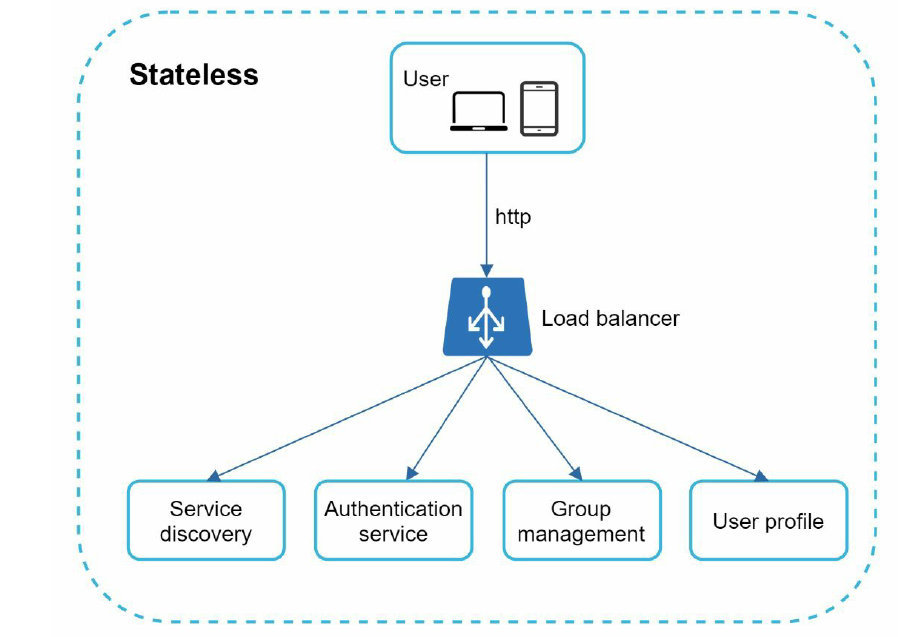
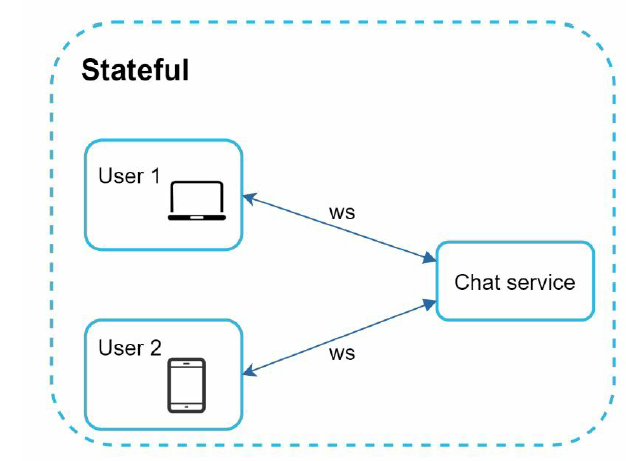
High Level Design:

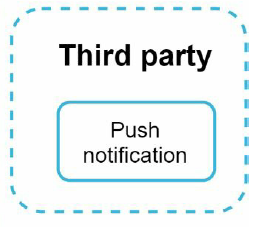
The chat server mostly divided into:

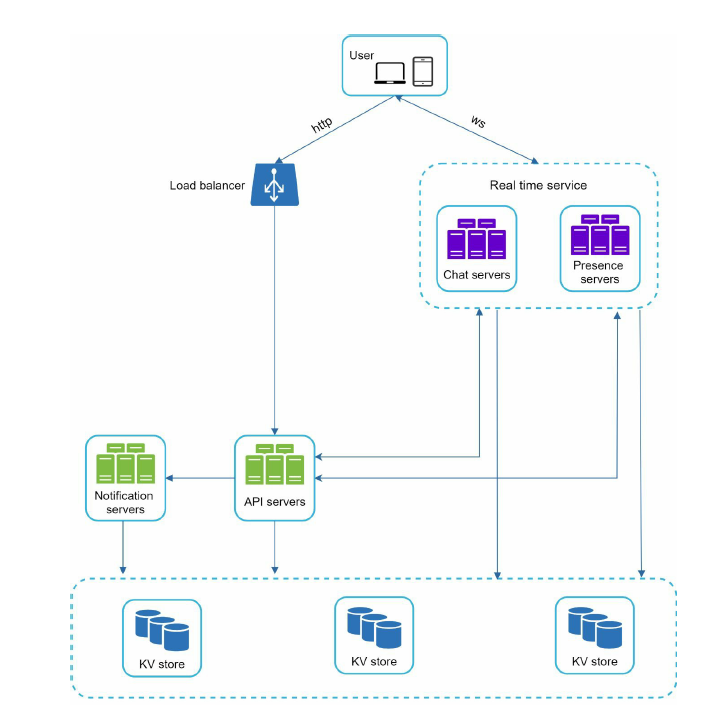
1) stateless service

2) stateful service

3) third party integration







Chat server facilitate message sending and receiving.

Presence server manages offline/online satus.

API server handels everything -user login,sign up, change profile etc..

Notification server push send notification.

KV store chat history.

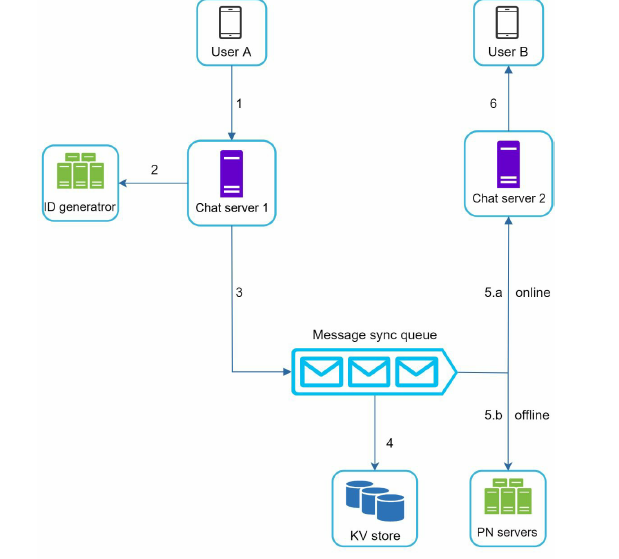
DB structure:

For Authentication/Authorization we can use a RDBMS DB. But for the text storege we can use a no-Sql DB.

|  |  |
| --- | --- |
| Message | |
| message\_id | bigint |
| message\_from | string |
| message\_to | string |
| content | text |
| Created\_date | timestamp |

|  |  |
| --- | --- |
| Group\_mesage | |
| Group\_id | bigint |
| User\_id | bigint |
| Message\_id | bigint |
| content | text |
| Created\_date | timestamp |

Low Level\_design:

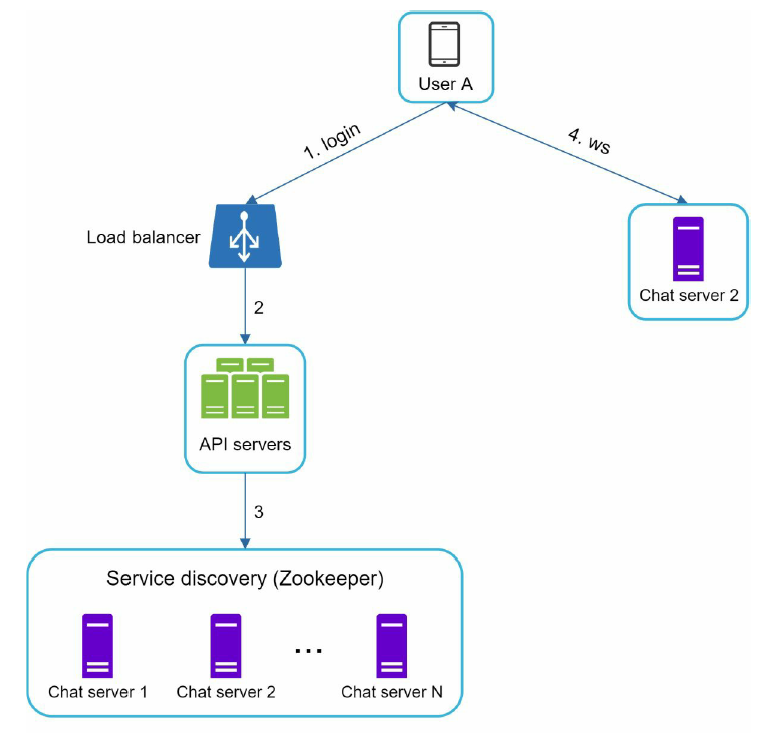


**Design Deep dive:**

For a chat system service discovery, messaging flows and online and offline requires a deep dive discussion.

Service Discovery:

Service discovery recommends the best serve to the client. By taking consideration into geographical area, capacity and availability. Mostly Zookeeper and Eureka.



User-A try to login to the app-

->the load balancer send login to the API server

-> After back-end verification done

->service discovery finds the best server to connect (here chat server-2)

->the user A connects to the chat server -2 with websocket.

Message flows:

1-1:

User-A: sends a message to chat server-1.

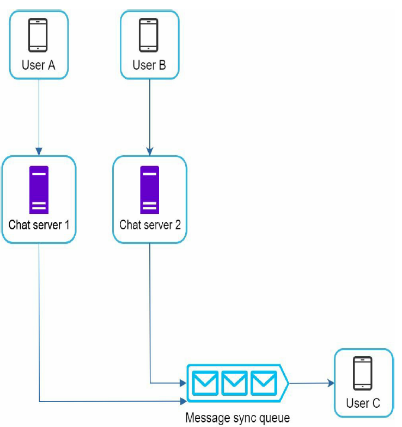
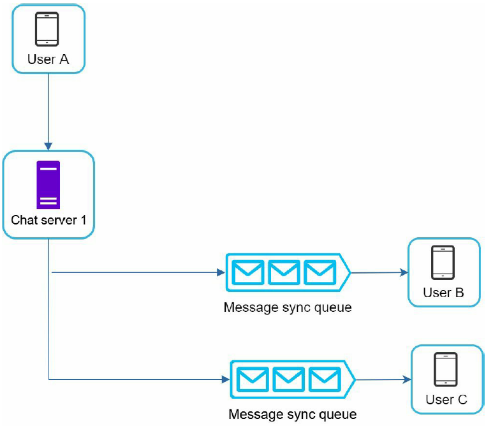
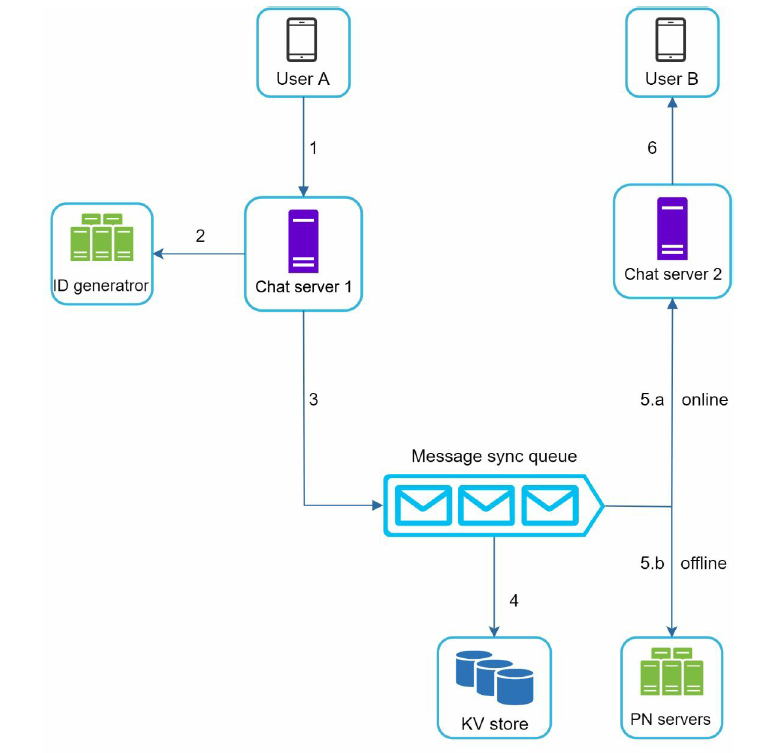
Chart server-1 generates a message\_id from id generator.

Chart server-1 sends message to the message sync queue.

The message is stored a key-value pair.

If user-B is online it sends message to User-B. if user-B is offline push notification send from push notification to user-B though chart server.

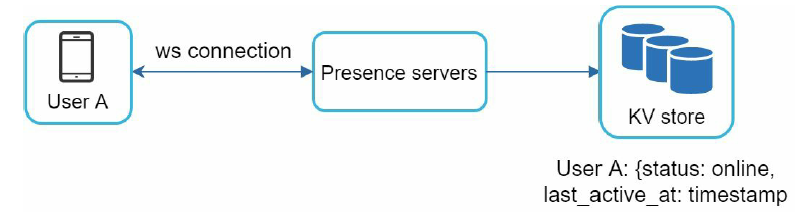
**Small Group chat flow:**

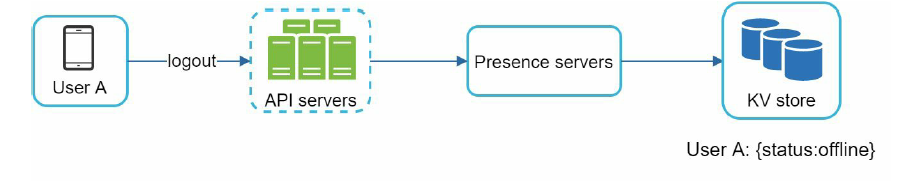
Online -Presence:

User login:

Once the user connected to the server through webSocket. A’s online status and last\_active timestamp stored to the KV store.

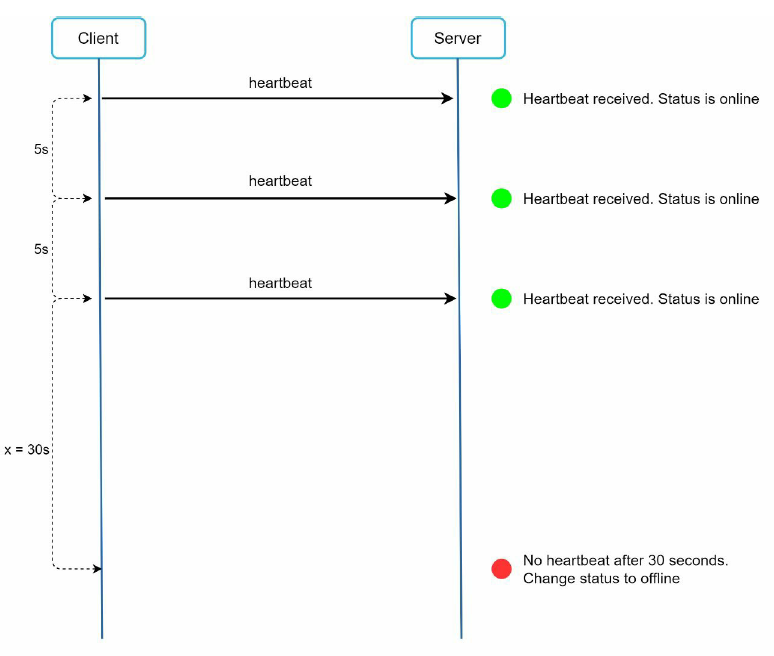


User Logout: -



User Disconnection: -

User disconnection can be taken care by heartbeat.



Online status FanOut in group:

For a group a publish-subscribe model is used for the online status change. Wher each friend pair maintain a channel A-B, A-D and A-D. Those are three channels subscribed by user B,C and D.

