WhatsApp System Architecture

Let’s design an instant messaging service like WhatsApp.

Photo by Alexander Shatov on Unsplash

WhatsApp is the most common application that almost all of us are using every day. WhatsApp helps us connect people across the world in a friendly and convenient manner. There are two types of the chat systems, one is Facebook Messenger which stores all messages permanently, another like WhatsApp which stores messages only until messages are undelivered. Once the receive confirmation is obtained, those messages will be deleted from the system.

Functional Requirements

1. Support one-to-one chats
2. Support offline sending message
3. Support to send messages to other users even the user is offline
4. support group chats
5. WhatsApp group with up to **256 participants**
6. Support video chat
7. Support group video chat
8. Allow voice messages
9. Support image, video, and file-sharing capabilities
10. Support encrypted message
11. Video cannot exceed 16 MB or 90 seconds to 3 minutes
12. Indicate read/receipt of messages
13. Last seen time of users (depends on a few scenarios)
14. Sent + Delivered + Read receipts ticks
15. File size can be shared is limited to 100MB
16. Support to play youtube, audio, and video on the display screen
17. Supports video formats — MP4, 3GP, MKV, AVI,and MOV
18. Notification will be shown once your file, video, audio, and image exceed the size limitation
19. Support to share the file, video, audio, and image to other applications which includes
20. Support reply to a particular message in a group chat

21. Allow us to forward any messages
22. Allow us to copy the message
23. Allow us to save the message
24. Allow us to archive the message
25. Allow us to reply privately to the particular message
26. Allow us to send and view videos and images one time and delete them once opened
27. Allow us to delete the message
28. Support to report to the WhatsApp team for a particular message (you can either block or report to the team)
29. Support to take photos via WhatsApp
30. Support to record the conversation once the voice recording function set to fully open
31. Support to access the contact list from the phone
32. Support to share location (either share real-time or share the current location)
33. Allow sharing the video and documents to Messages, Facebook Messenger, Mail, WeChat, Telegram and etc
34. Can edit/upload, delete the user profile
35. Can add your biography
36. Can link to other devices
37. Can update your status
38. Can have your customized settings (different scenarios)

Non-Functional Requirements

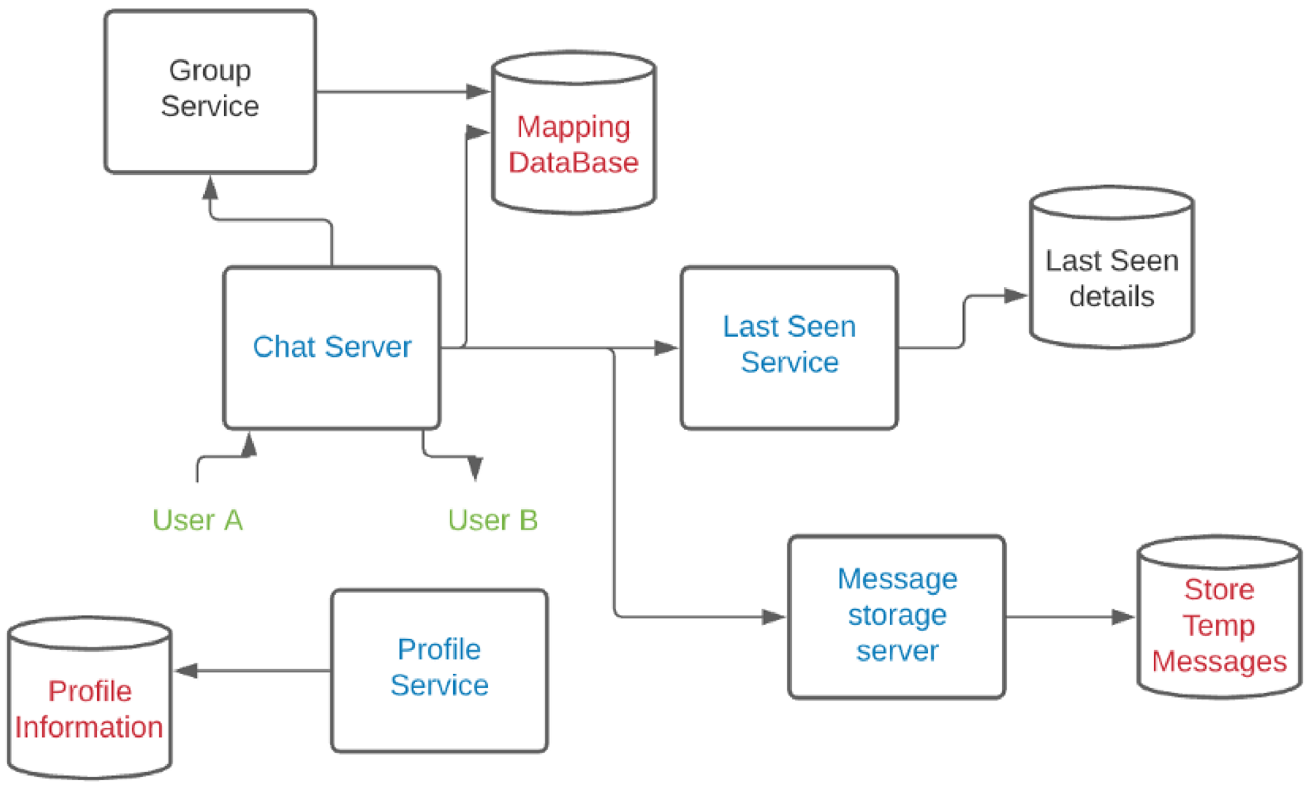
1. Very low latency
2. Always available

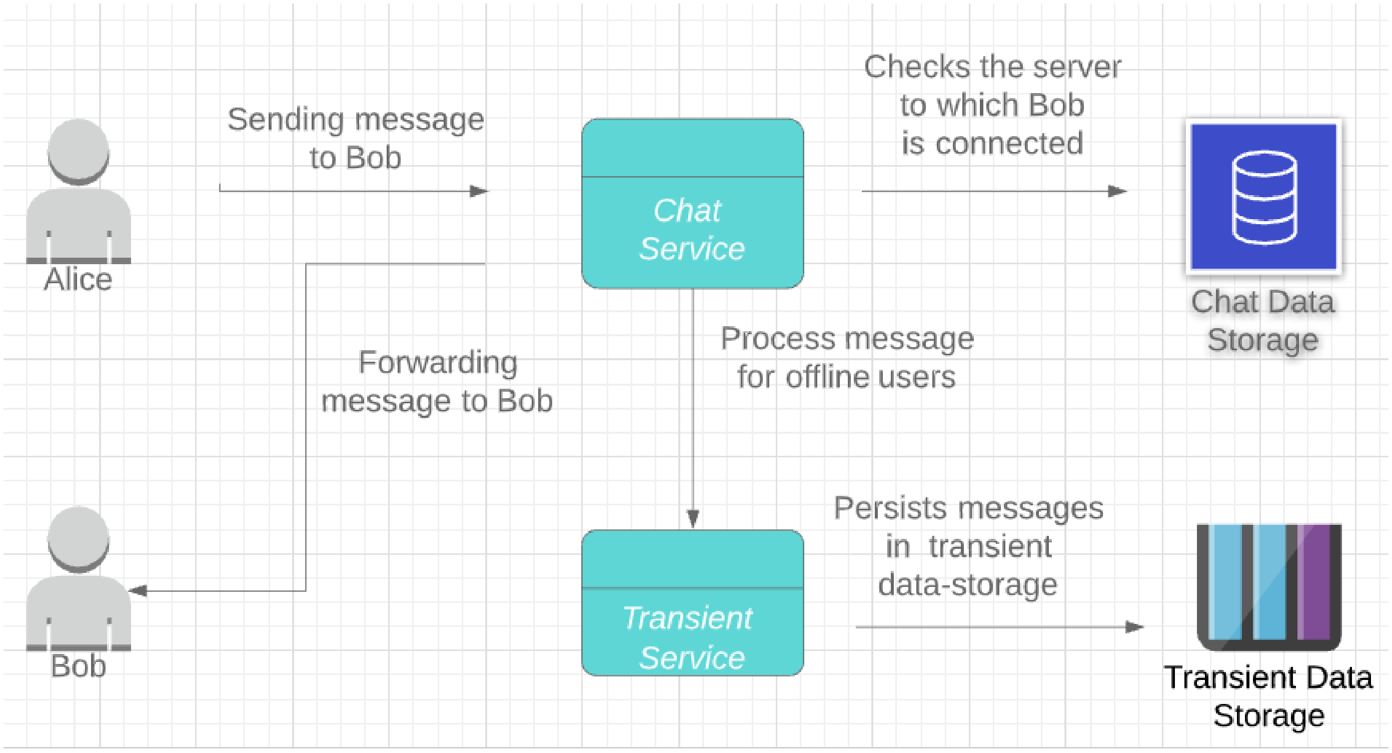
3. without any lags

4. Highly scalable

5. Consistency (what is shown on other devices, will be synchronized with other devices)

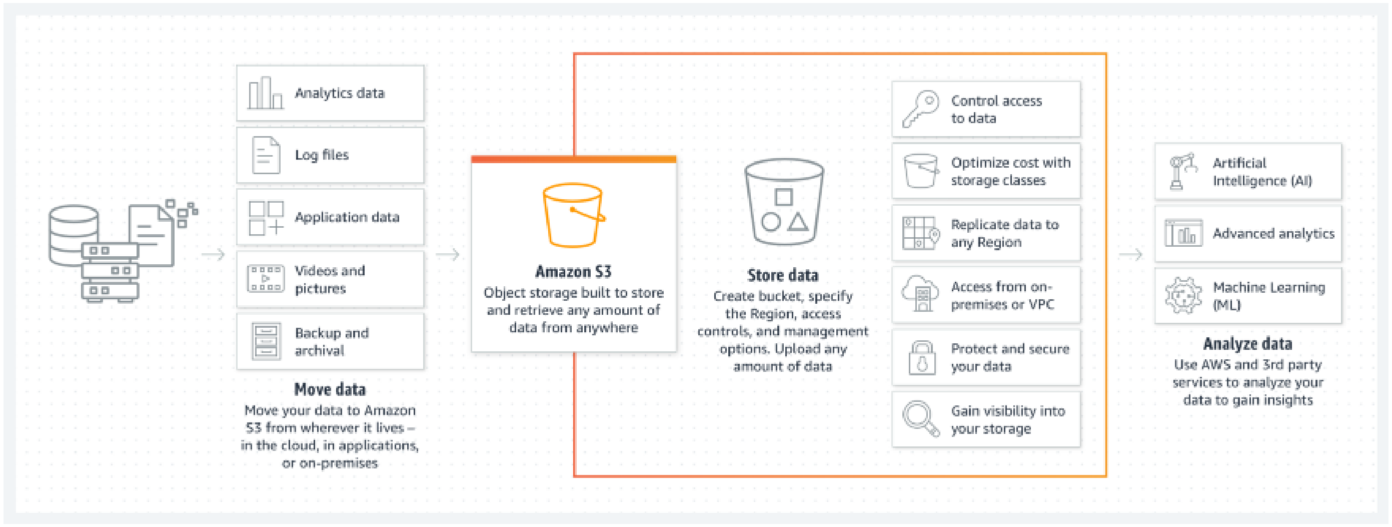
System Architecture Components




Alice
Sending message to Bob
Chat Service
Checks the server to which Bob is connected
Chat Data Storage
Forwarding message to Bob
Process message for offline users
Bob
Transient Service
Persists messages in transient data-storage
Transient Data Storage

1. **Profile Database**

- store profile information about users like status, profile picture, profile ID, and contact details
- Store profile picture via s3 bucket technology https://aws.amazon.com/s3/ so that every profile picture have their respective links


Analytics data
Log files
Application data
Videos and pictures
Backup and archival
Move data
Move your data to Amazon S3 from wherever it lives – in the cloud, in applications, or on-premises
Amazon S3
Object storage built to store and retrieve any amount of data from anywhere
Store data
Create bucket, specify the Region, access controls, and management options. Upload any amount of data
Control access to data
Optimize cost with storage classes
Replicate data to any Region
Access from on-premises or VPC
Protect and secure your data
Gain visibility into your storage
Artificial Intelligence (AI)
Advanced analytics
Machine Learning (ML)
Analyze data
Use AWS and 3rd party services to analyze your data to gain insights

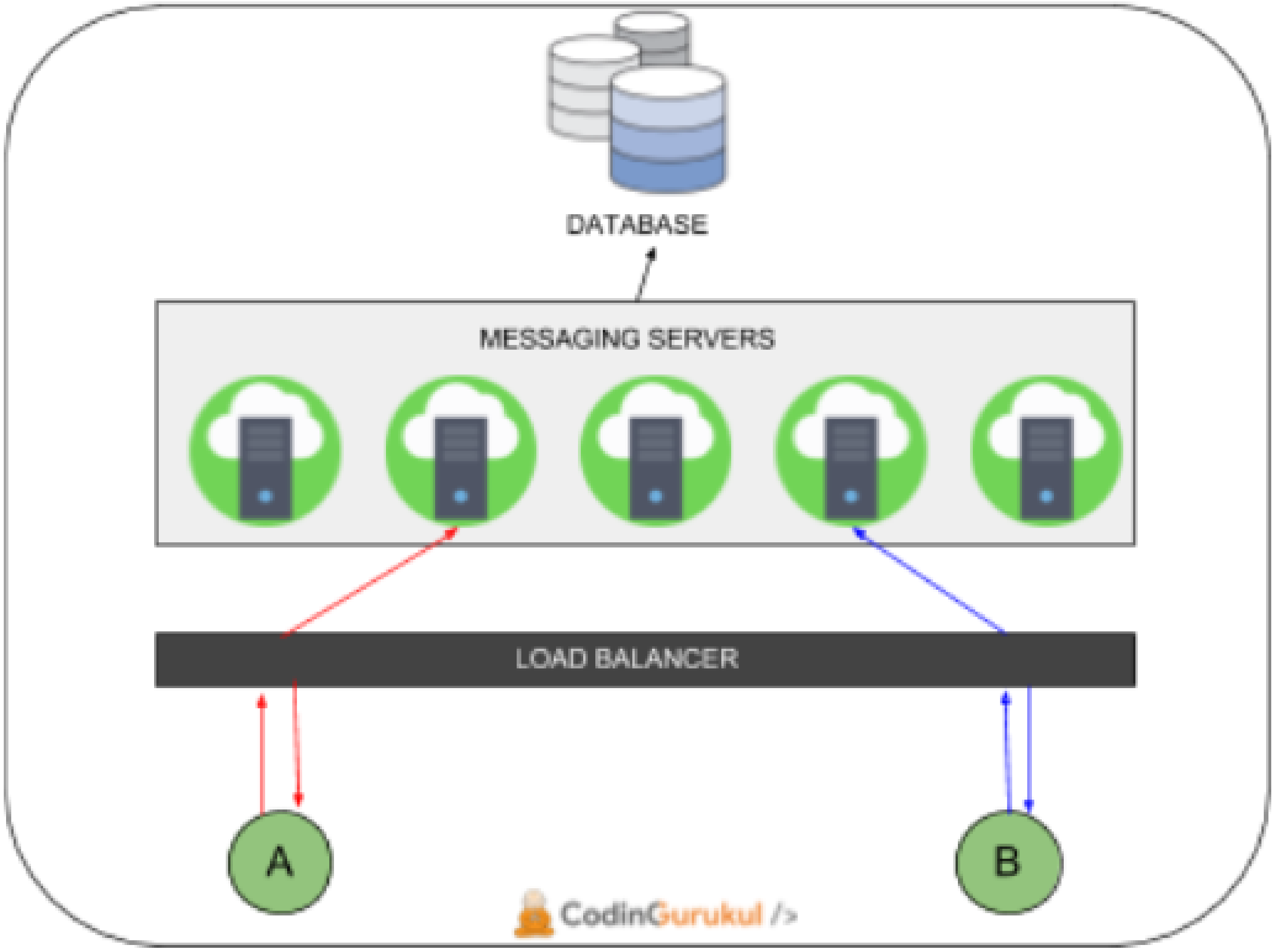
2. Profile Service (API)

- Has multiple endpoints which help retrieve details about the user. Get — retrieve profile information, post/put — add/update profile information

Chat Server

3. Mapping Database

- Before we go to understand deeper, do you have the concept of the general communication architecture? When 2 clients (A and B) want to communicate and send messages to each other, they have to know the address of each other (IP, MAC, or any customized unique identity) and exchange messages over a network, in this case, it is the internet. And they will use the bi-directional connection. Then it will go for many evolution processes, finalize it like the picture below. Server, client, load balancer, and databases.



- The HTTP protocol is not used because every user needs to send a request to the server, once a response is received from the client-server, then the connection is closed, it mean every message needs to wait for the full transaction between server-client. But, we don’t want to waste time and resources creating connection, so the WebSocket protocol is used because the connection is not closed immediately. Web Socket Handler WSH1 will be connected by the user. Web socket handler is a lightweight server which keeps an open connection with all the active users.

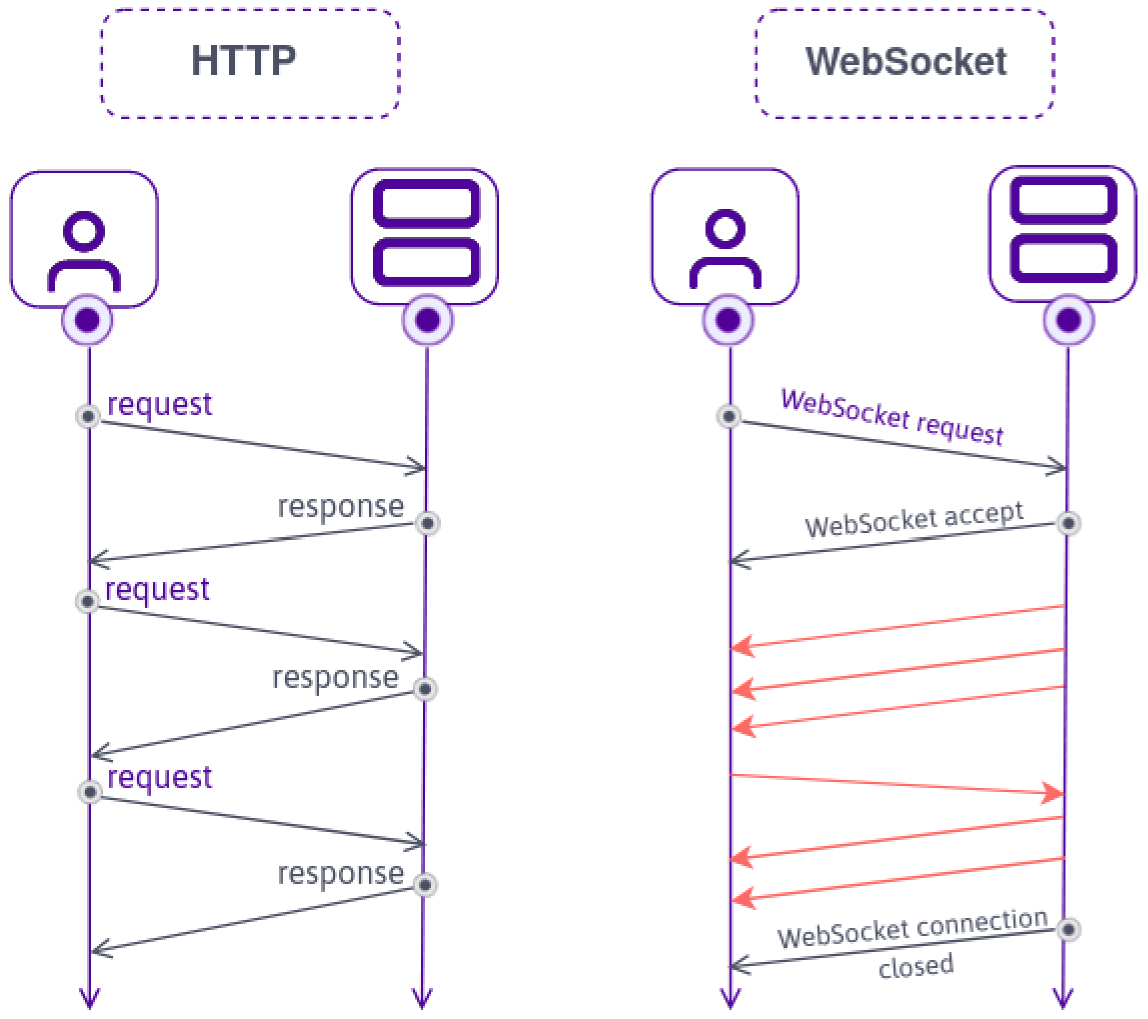
- Each machine will have around 65K open ports. Even if we end up using up to 5K ports for our internal use and communicating with other services in our system, it still has up to 60K ports that we can use to connect with users. A web socket handler will be connected to a web socket manager which is a repository of information about which web socket handlers are connected to which users. **Redis** is used because it can store 2 types of information: which user is connected to which web socket handler, what all users are connected to a web socket handler. When a connection between a user and web socket handler breaks and they connect to another handler, this information will be updated into Redis.

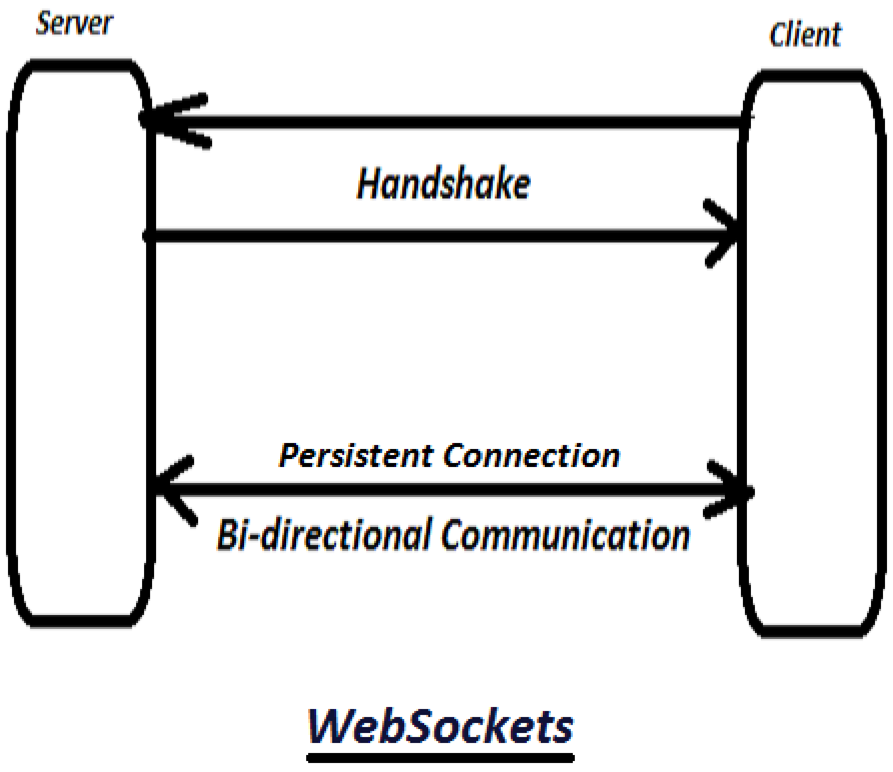
- Message service: a repository of all messages in the system. APIs will get messages by various filters like user id, message-id, delivery status and etc. Cassandra is used for message services because it can handle ever-increasing data because old and new users keep chatting every day. When a web socket handler talks to a web socket manager and message service in parallel.

rfc6455

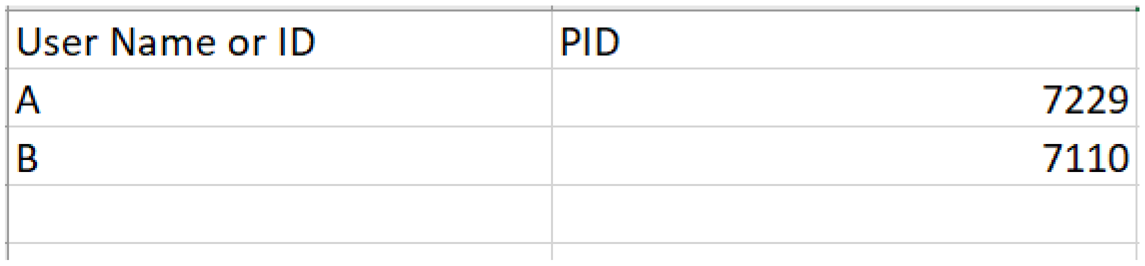
Internet Engineering Task Force (IETF) I. Fette Request for Comments: 6455 Google, Inc. Category: Standards Track A…

datatracker.ietf.org


HTTP
WebSocket
request
response
request
response
request
response
WebSocket request
WebSocket accept
WebSocket connection closed

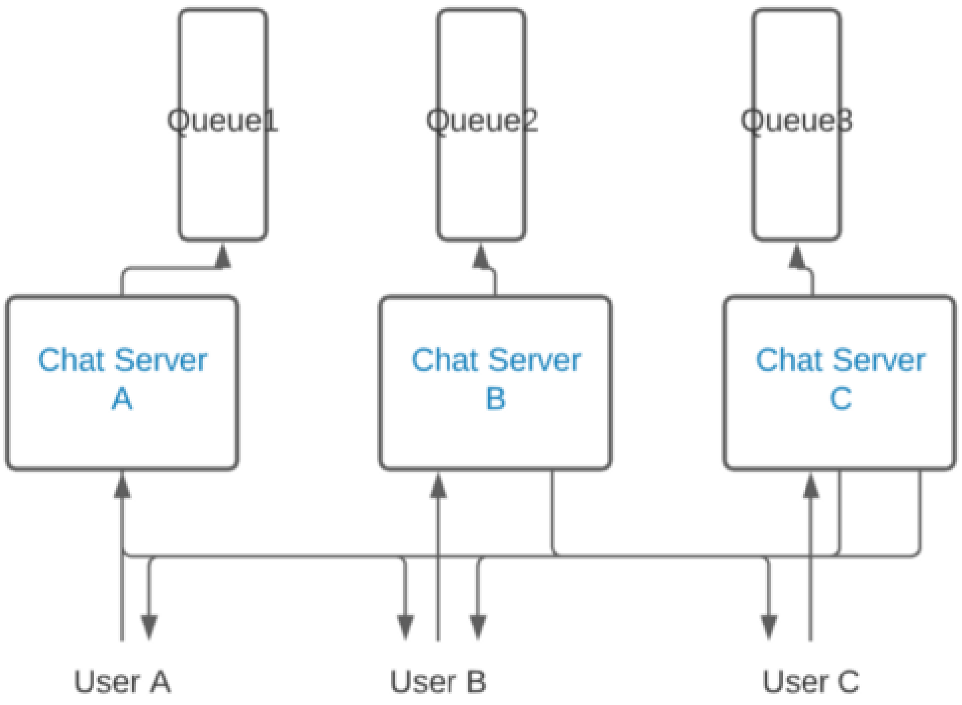


- Then we can derive a few scenarios out. In this case, the sender is connected to the server but the receiver is not connected, the message is stored in the database and when the receiver connects to the server, the message is fetched from the database and forwarded to the receiver. When the sender is not connected to the server, the message is saved in device storage (it may be SQLite or anything else based on the platform) and when the sender goes online, the message is fetched from the local storage and sent to the server. When both clients are connected to the server, the sender sends the message, and the server forwards that message to the receiver without storing the message to the database or device local storage.
- **Mapping Database — The chat** server will create a new process (a thread) for the sender, user A. Same goes for the receiver, user B if user B is online. The server finds out the name of the receiver, then fetches the data out from the database and finds out the process id (pid) for user B so that the message can be sent to user B. If user B wants to send the message to user A, the server finds out the name of user A in the mapping database, once the pid is found, the message is sent to that pid for user A. Whenever a user talks to the system, a new process is created and details are stored in the mapping database.



User Name or ID	PID
A	7229
B	7110

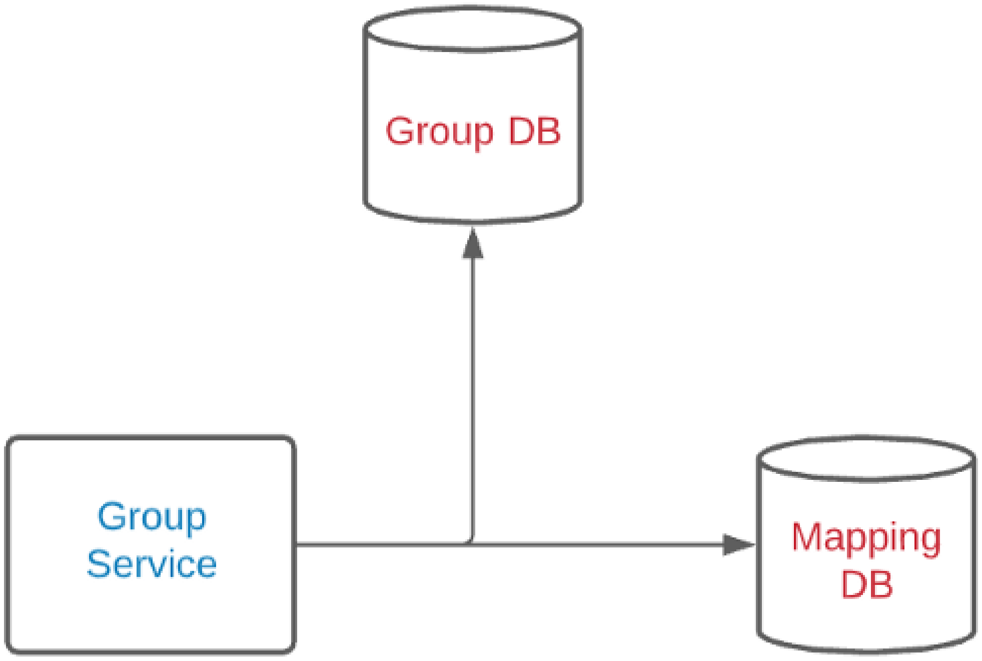
In the real world application, the structure should be similar to this picture below:



- A queue with each chat server is included because the queue can handle the excessive load of messages and there's no failure in sending any messages to different users.

4. Group service

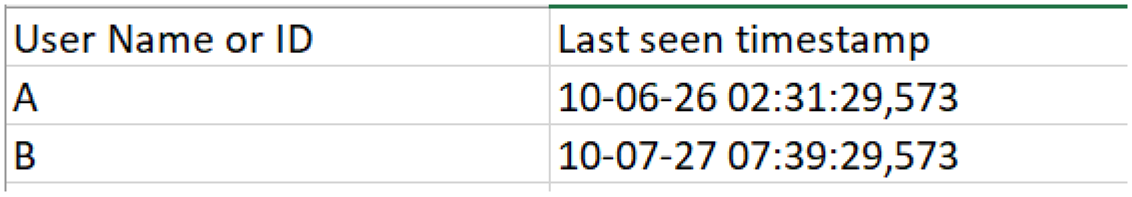
- For every group is created, a new groupId is created too. This groupId will then have a mapping to all the users who are in the group. Whenever a message is sent, the chat service will find out the kind of receiver, either a group or a single user, if it is a group, chat service will pass to the group service to make a call to group db to search the same groupId, and sent messages to all the userIds in the list.



- Web socket handlers cannot keep track of groups, it only tracks active users. So, when a sender wants to send a message to a group, a web socket handler from a sender gets in touch with the message service, and the messages will be stored in Cassandra. The message service then communicates with Kafka. Those messages are saved in Kafka with an instruction to send to the group. Kafka will interact with the Group message handler and send out all group messages, this is the mechanism of group service. The group message handler will talk to the group service and find out all the users in the group list, and back to the process through a web socket handler and deliver the message to all users individually.
- Group service maintains all group-related information like which user belongs to which group, user ids, group ids, a time when the group was created, a time when every user was added, status, group icon, etc. This data also will be stored in a MySQL database which has multiple slaves in different geographical locations to reduce latency. Caching method is applied for the frequency group.

5. Last Seen Service

- The last seen service uses a last seen database so that the last seen information can be stored. It just maintains the last seen time details, the latest always replace the older one. Whenever a user opens WhatsApp and sends any users related request to the chat server, the chat server can call this service to update the time details.



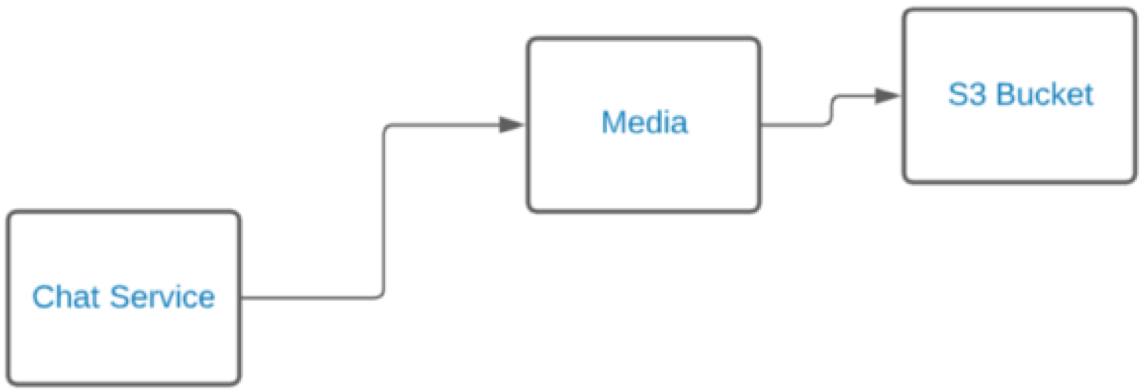
User Name or ID	Last seen timestamp
A	10-06-26 02:31:29,573
B	10-07-27 07:39:29,573

6. Message storage server & Store temp message DB

- when the receiver is offline, the message must be encrypted and stored in the server DB. once the receiver goes online, all the messages can be delivered

7. Media type messages

- chat service takes the messages and finds out the type of message. Once the message type or chat service detects the format of the message as media, it is stored in an S3 bucket which is an object storage service. The links to these media are then stored in a SQL or NoSQL db with mapping to the user details. We can use an HTTP protocol to transfer these messages.



- The image, files, and video that were sent over WhatsApp will be compressed and encrypted at the device end and the encrypted content will be sent to the receiver and the content will be decrypted on the device end on the receiver side.
- If a user sends an image to another user, the user will upload the image to a server and get the image id. Then it will send the image ID to another user. Another user can search and download the image from the server. Or another method is the image will be compressed on the device side and sent to an Asset service through the load balancer. The image will be stored in the s3 bucket. The image might not be loaded onto a CDN. Once the image is loaded in S3, the asset service will send the image id, the user will communicate with another user through web socket handlers about this information.

WhatsApp Front-End

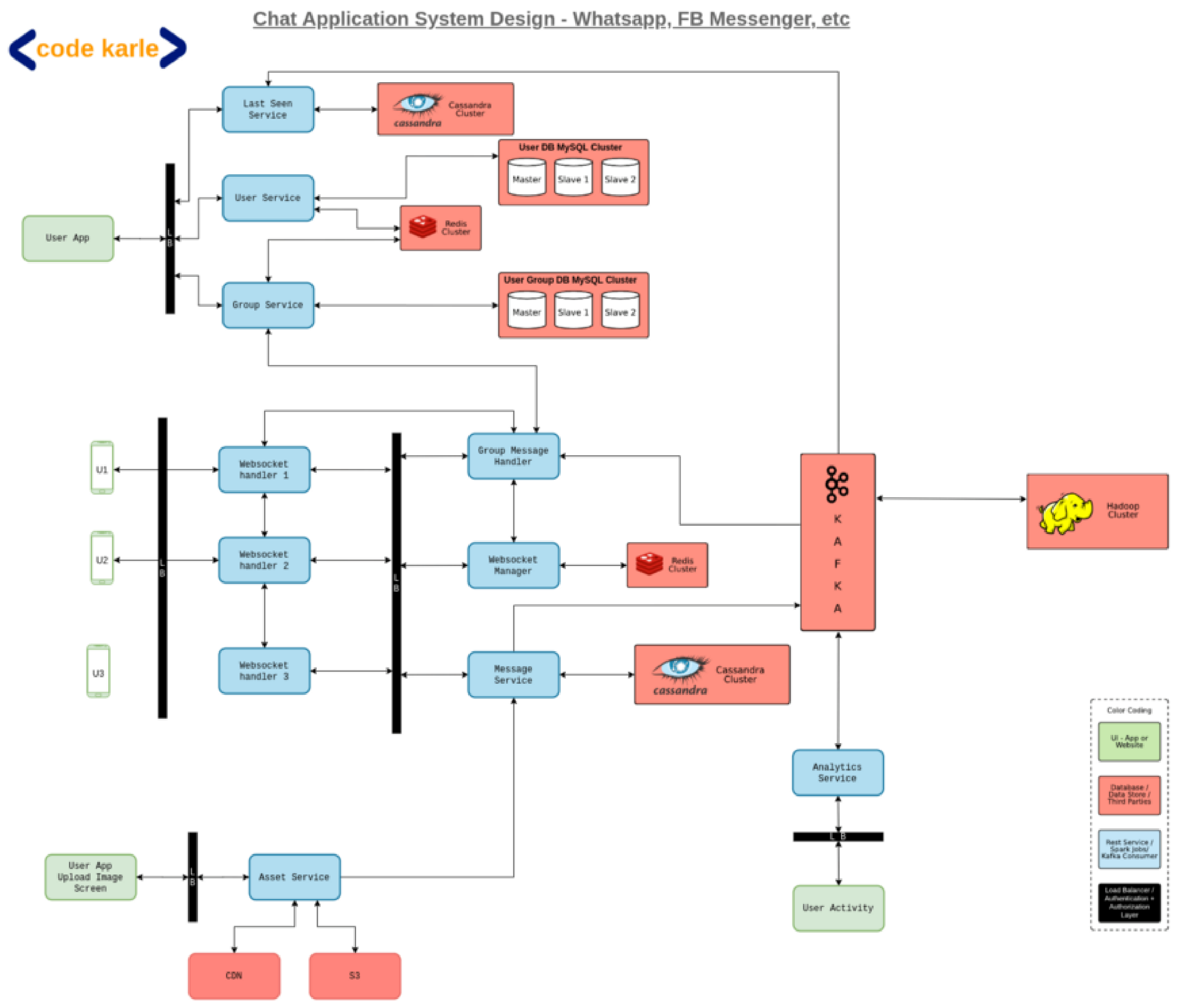
- **Android:** Java
- **iOS:** Swift
- **Windows Phone:** C#
- **Web app:** JavaScript/HTML/CSS/
- **Mac Desktop app**: Swift/Objective-C
- **PC Desktop app:** C/C#/Java

WhatsApp Back-end

- **Erlang** is the main programming language
- **FreeBSD** is the operating system
- **Ejabberd** is the XMPP application server
- **BEAM** is the Erlang-based virtual machine
- **Mnesia** is their Erlang-based database

- **YAWS is their multimedia web server**

The high-level system architecture



Further resources