What’s the best data store key value to use for a chat service as mentioned before?

How many websocket connectison can each service have?

**65,536 socket connections**

For better understanding of WS load balancing let's dive a bit deeper into TCP sockets background. By default, a single server can handle 65,536 socket connections just because it's the max number of TCP ports

**High level design here**

**Building the chat system design here**

Desigining the database table for 1-1 chat, for group chat, with contacts, adding attachments for messages. And then also some other parts of this

**To handle millinos of ysers, 1 websocket will not be enough here, needs more for sure her.**

A diagram of a software system

Description automatically generated

**What does message service do?**

It acts as an interface to the database for other services interacting with the databases. It stores and retrieves messages from the database and deletes them after a specific amount of time

**What happens if a user A wants to send a mesasge to a user B?**

Now, if user A wants to send a message to user B. Since we have a number of WebSocket servers these users can be connected to different ones. So how does this work:

1. User A communicates with the corresponding WebSocket server to which it is connected.
2. The WebSocket server associated with user A identifies the WebSocket to which user B is connected via the WebSocket manager. If user B is online, the WebSocket manager responds back to user A’s WebSocket server that user B is connected with its WebSocket server.
3. At the same time, the WebSocket server sends the message to the message service and it is stored in the database (in case user B is offline). So the eviction policy for the message to be processed will be first-in-first-out (which makes perfect sense in this case). And when the messages are delivered to the receiver, they are deleted from the database.
4. Now, user A’s WebSocket server has the information that user B is connected with its own WebSocket server. Both users communicate with the WebSocket manager to find each other’s WebSocket server.
5. If user B is offline, messages are kept in the database. Whenever they become online, all the messages intended for user B are delivered via push notification. Otherwise, these messages are deleted permanently after 30 days.

**How does user A send data to user B here?**

A diagram of a service

Description automatically generated

when a user say user a wants to send a message to user B he sends a request to the messaging service with the ID of user B now before this user a establishes a persistent connection to the messaging service via websocket protocol because like we discussed earlier it's a bi-directional connection now in a traditional HTTP via session service and sends the message accordingly

**How would a session service work?**

so how does session service works? whenever a user connects to the messaging service the messaging service tells the session service in which websocket handler the user has established the connection which is stored in a database more

**What happens if user is offline here?**

A diagram of a service

Description automatically generated

likely a nosql database since you don't have any scope for relations between the data later this information is used to send messages to the other end now what if the user B is offline in such cases we need to temporarily store the message to deliver it

**Using the relay service:**

lately message service forwards this message to the relay service in the occasion when the user is offline, he relay service will store the unsigned messages the from and the two user ID in a database like Cassandra.

**Last seen service:**

last scene service is used to store the timestamp of each user this information is based on logging of each user activity the client-side application should be intelligent enough to identify the difference between the user activity and the application activity itself and sends a signal to the app server this information can also be used to show the online status of the users. Used for online services

**Using an asset service**

Asset Services used to store and retrieve multimedia files in an object-based storage or a blob store like AWS S3 bucket and finally we have a group messaging service which is more like the messaging service except we need to publish the message to all the users associated with the same group ID this service will rely on the session service to identify the server to which each user is connected to now compared to other services group messaging service is fairly complicated and

**Maintaining a hash for each file**

When sending a file here, the asset service assigns an ID that’s communicated with the sender. The asset service also maintains a hash for each file to avoid duplication of content on the blob storage.

* + For example, if a user wants to upload an image that’s already there in the blob storage, the image won’t be uploaded. Instead, the same ID is forwarded to the receiver.

File can be downloaded using the same hash id here,

**How would user b know what image or asset to download?**

when a user a sends an image to another user B user a will upload the image to a server and get the image ID and then it will send the image ID to user B user B can now search the image and download the image from the server or another method is the image itself will be compressed on the device side

and sent to the asset service the asset service can take the message and find out the type of message once the asset service detects the format of the message as media or image it is stored in our blob store which in this case is an S3 bucket the links to the location of the media files can be stored in a SQL or nosql database with mapping to the user details now guys there is no right and wrong design as long you can clarify the requirements for the interviewer defined your design and make changes if any you are all set [Music

Database schema here

A diagram of a database schema

Description automatically generated

**Database section here**

apical contact number and then we'll have groups and each group will have its own group ID and the user ID of the user part of that group

then we have unsent messages with Fields message ID sent from ID sent to ID the content or the media URL say if you are sending messages such as video messages or PNG

files you send the location of that PNG file in the form of media URL and the timestamp you can also have a last scene table with a user ID and this timestamp and finally you have the sessions table which basically maintains mapping of user ID with a server ID what we discuss until now was how a personal chat works on a system like

WhatsApp but what about group chats for every group Creator will have a new group ID and this group ID will have a mapping to all the users who are in the group that is in the group mapping database

**Desigining for group chat here**

A diagram of a chat

Description automatically generated

now groups will behave a little different from users because websocket handlers won't keep track of groups it just tracks active users so when user a wants to send a message to Group 1 websocket

**Why add kafka here?**

1. Using producer here: Messaging service
2. Using consumer: the

Handler 1 gets in touch with message service message service will store the information to a queue or Kafka topic automation

**How does it work exactly here? What’s the total flow here?**

such as which user is sending word message to which group basically the message service will act as a Kafka producer and then we'll have group message Handler as our Kafka consumer which will be listening to the Kafka topics so whenever the message service posts a new message to Kafka topic that is user a is sending this message hello to group 1.

**group message Handler will query group service** to get all the users which are in group id1 after it gets the list of users who are supposed to get the message it now needs the respective list of machines those users are connected to which it will get from the websocket manager once it gets the list of all the machines the group message Handler will send messages to individual machines by talking to the respective websocket Handler again websocket Handler is a lightweight server which keeps an open bi-directional connection with all the active users and for whatever reason if the receiver is offline the message must be encrypted and stored in the servers database o

**What exactly happens?**

Since user A is connected to a WebSocket server, it sends a message to the message service intended for Group A.The message service sends the message to Kafka with other specific information about the group. The message is saved there for further processing. In Kafka terminology, a group can be a topic, and the senders and receivers can be producers and consumers, respectively.

Now, here comes the responsibility of the group service. The group service keeps all information about users in each group in the system. It has all the information about each group, including user IDs, group ID, status, group icon, number of users, and so on. This service resides on top of the MySQL database cluster, with multiple secondary replicas distributed geographically. A Redis cache server also exists to cache data from the MySQL servers. Both geographically distributed replicas and Redis cache aid in reducing latency.

The group message handler communicates with the group service to retrieve data of Group/A users. In the last step, the group message handler follows the same process as a WebSocket server and delivers the message to each user.