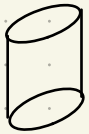
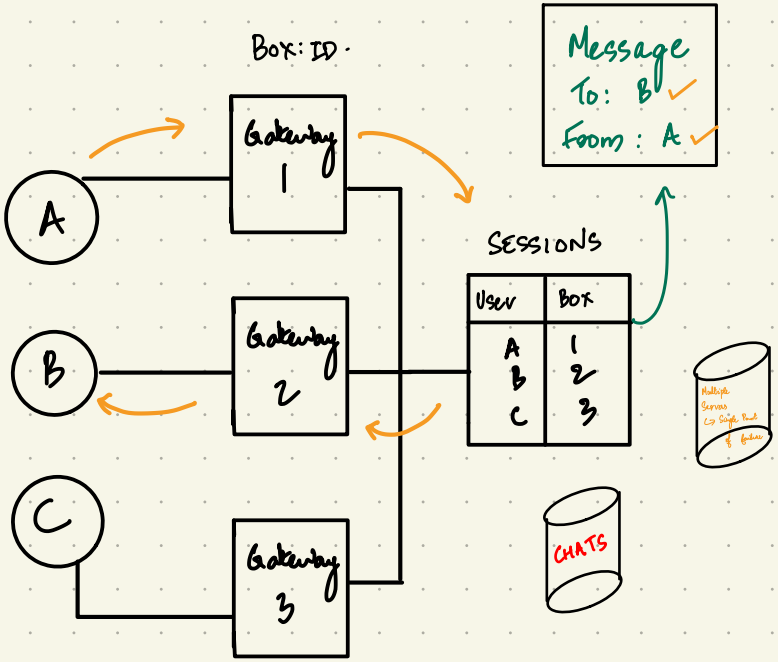
WhatsApp



HTTP : This is an option, but this is a one sided communication.
Client ->Server
We can try **“Long Polling”** However, this is not **Real Time**.

1. Group Messaging
2. Sent/Delivered/Read reciepts
3. Online/Last seen
4. Image Sharing
5. Chats- Temporary/Permanent



A sends a message to gateway 1 which forwards it to sessions.
When session receives the message it parallelly sends a message back to A saying the message has been **“SENT”**
Session stores where each user is connected to
It checks where B is and forwards it to B
The message is the sessions is stored with a field “To” and “From”, so when the “To” says it’s received the message, A is notified that the message has been delivered to B
When B receives it, it parallelly sends a message back to sessions saying it has been **“DELIVERED”**
When user **B** opens the message and reads it another message is sent to the session saying the user has read it and A is notified that B has read it

Last Seen - HOW TO IMPLEMENT

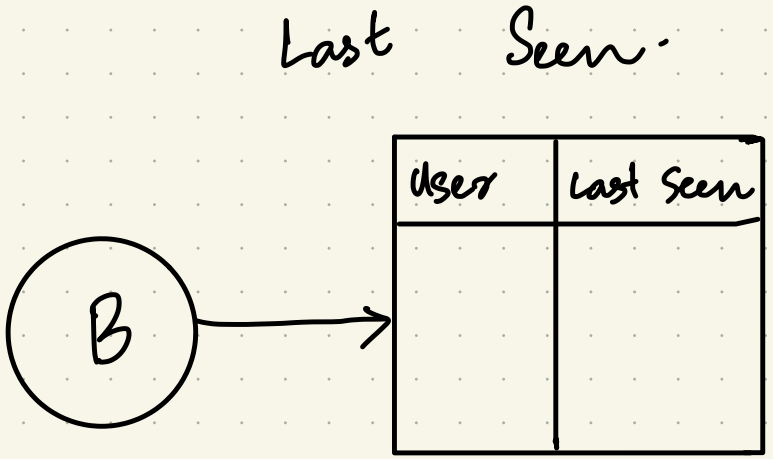
1. Whenever A requests a particular service from the server, we know that A was online at that point.

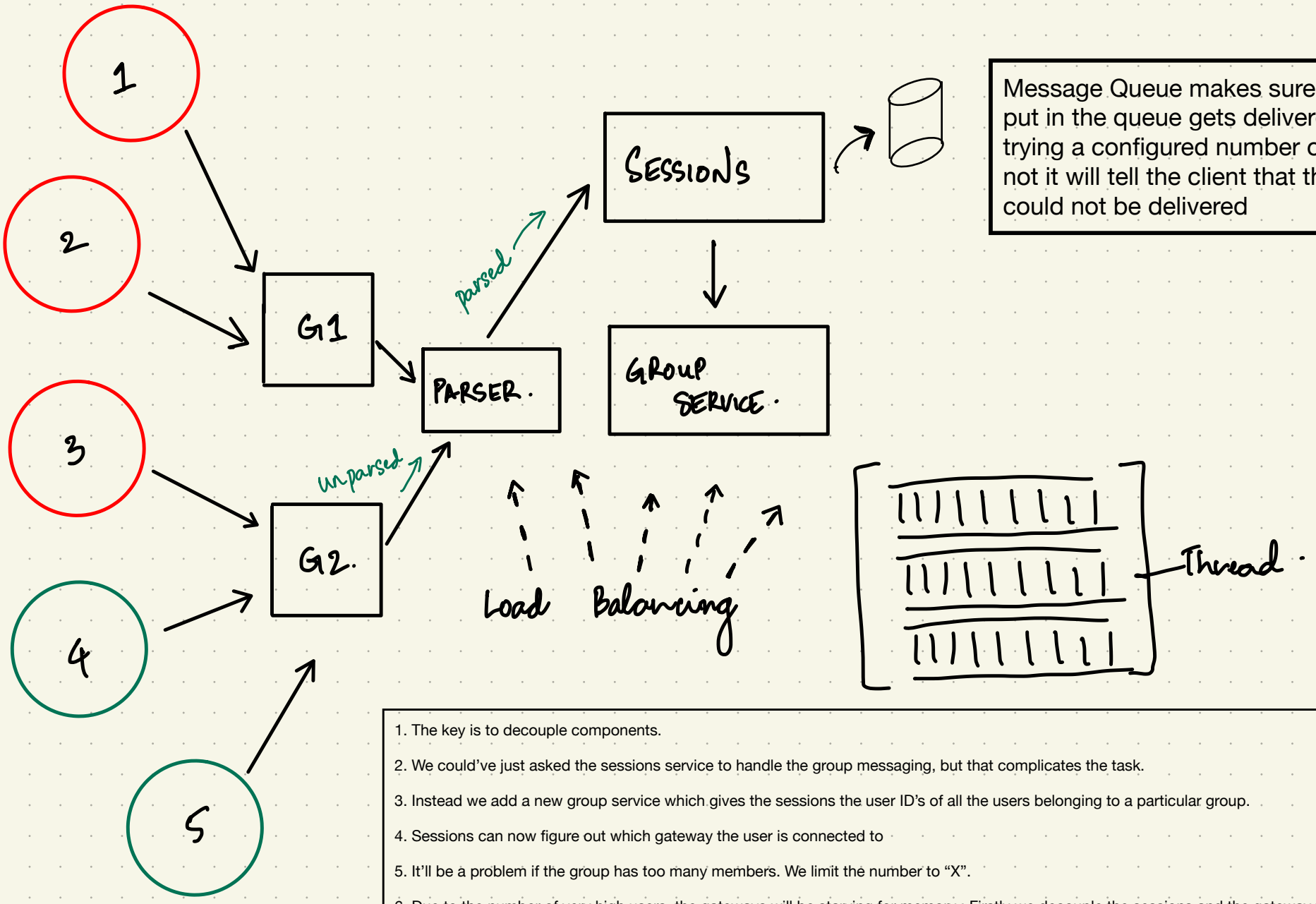
2. There should be a threshold, eg: it should not say the user was last seen 3 seconds ago.

3. Updating Last Seen: Whenever a user sends a request to the gateway, a micro-service called the “Last Seen” is updated
   # There are two kinds of request:
      1. User requesting
      2. Application Requesting (Delivery reciepts)

The client should only send user requests and not system generated requests.

4. B can query the Last Seen micro-service and find out when A was last seen





Message Queue makes sure a message put in the queue gets delivered after trying a configured number of times. If not it will tell the client that the message could not be delivered

1. The key is to decouple components.

2. We could've just asked the sessions service to handle the group messaging, but that complicates the task.

3. Instead we add a new group service which gives the sessions the user ID's of all the users belonging to a particular group.

4. Sessions can now figure out which gateway the user is connected to

5. It'll be a problem if the group has too many members. We limit the number to "X".

6. Due to the number of very high users, the gateways will be starving for memory : Firstly we decouple the sessions and the gateway then we sedn unparsed messaged from the gateway to a new micro-service called the Parser which takes care of authenticating, converting the message to programming language object and what not.

7. The Group service is carried out using consistent hashing since there are a lot of duplicates in here. If this service fails, we have to try again, therefore a "Message Queue" is used.