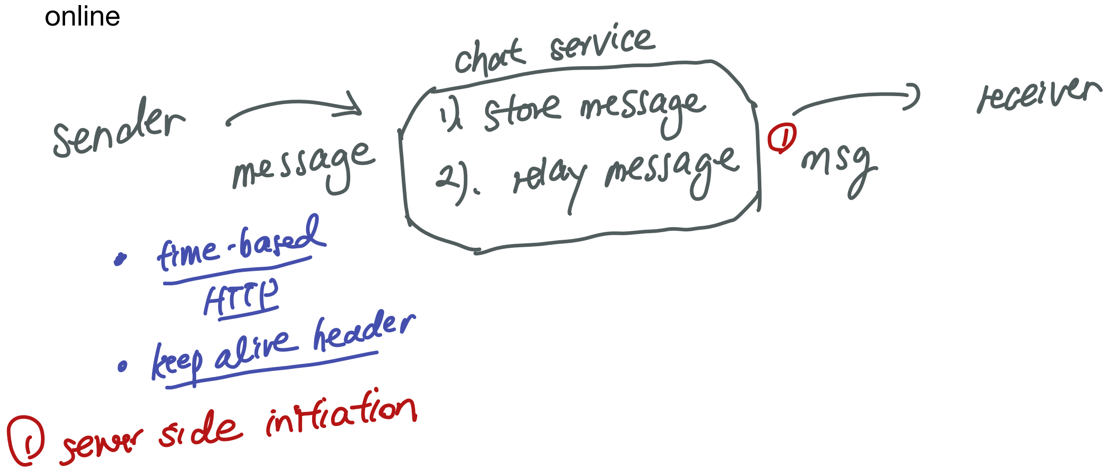
Focus on designing a chat app like Facebook messenger, it should be able to do

- 1-1 chat
- Group chat
- Online presence
- Multiple device support (can be logged into multiple accounts at the same time)
- Push notification

Backgrounds:

1. Client does not communicate directly with each other. One thing to keep in mind is that each client connects to a chat service, which supports all the features mentioned above.
   A. Receive messages from other clients
   B. Find the right recipients, relay the message to the recipients
   C. If a recipients is not online, hole the message for that recipient on the server until the person is online

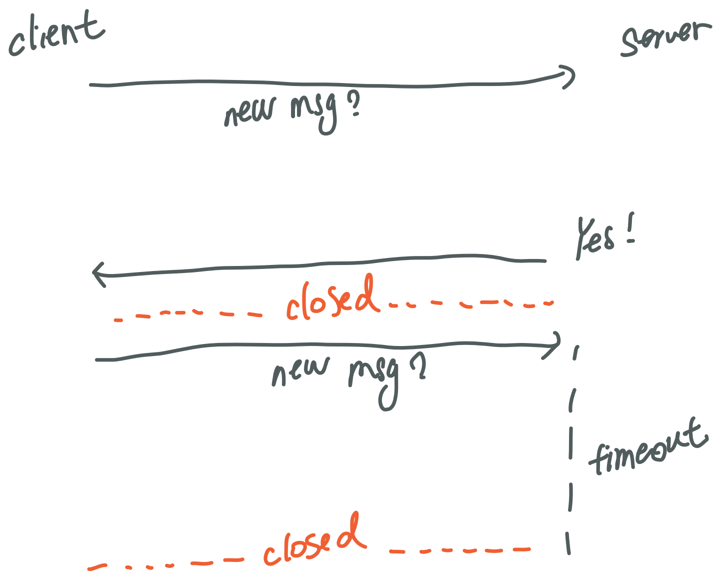


Polling

- The client periodically asks the server if there a message available. Depending on polling frequency, this can be very, very costly.
- Think about it, you don't wanna ask a question too many times. It would be very costly to the server answer time

Long Polling

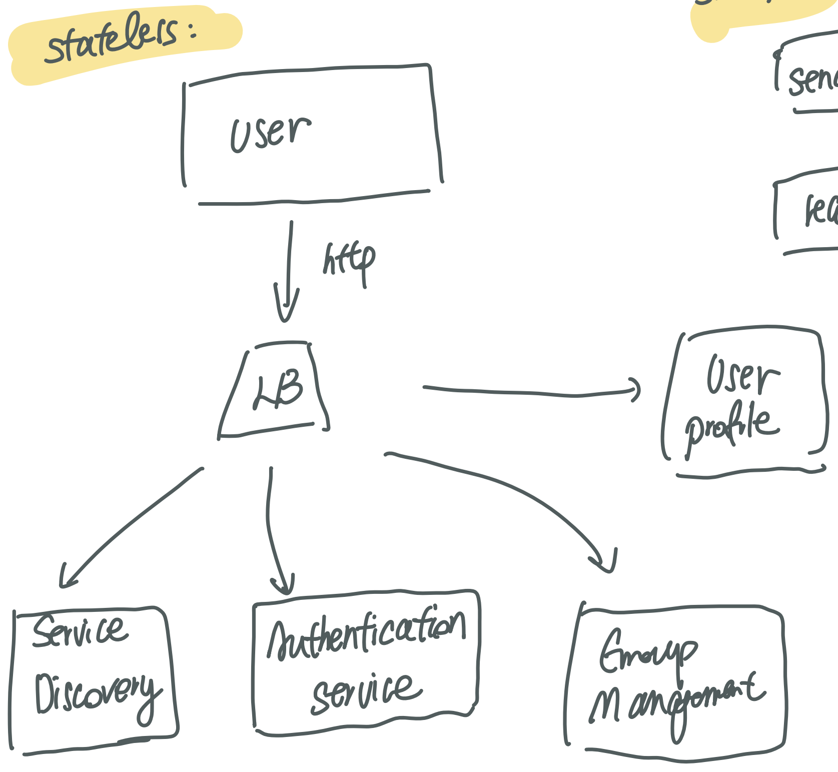
- Clien asks once, server returns only if there's a message; and then client asks again

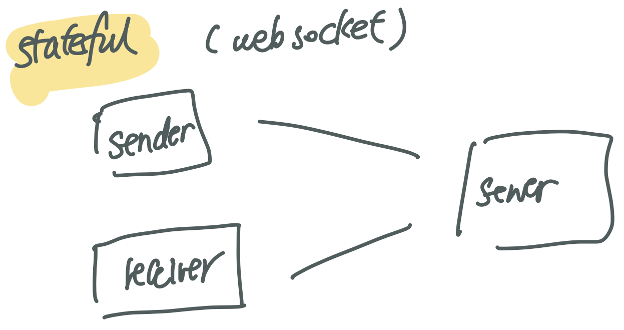


Lon Polling has some issues:
- Sender and receiver may not connect to the same chat server. HTTP based servers are usually stateless. If you use round robin for load balancing, the server that receives the messsage might not have a long-polling connection with the client who receives the message. (Basically you might just hit a different chat server at the end)
- (New font!)
- It's still very inefficient, because the user is inactive, it still wastes a lot of resource.

WebSocket
- The most common solution; The connection is initiated by the client, it's bi-directional and persistent. It starts its life as a HTTP connection and could be up graded via some well-defined handshake to a WebSocket connection. Though this persistent connection, a server could send updates to a client.
- The primary interface for connecting to a WebSocket server and then receiving data on the connection
- Allows web browser and web server with lower overhead than half-duplex alternatives such as HTTP polling, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to sending content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open

Now on the high level:


stateless:
User
http
LB
User profile
Service Discovery
Authentication service
Group Management


stateful
(websocket)
sender
receiver
server