Two-page project proposal

Group 02: Ning Ding (3610727), Xin Pang (3111543), and Yang Fu(3595130)

1 Introduction

The goal is to build a distributed chat system. It allows users send messages to each other or in form of a group chat. The whole system are consists of servers and clients, it has to fulfill some properties such that the communication and organisation among the clients and servers is not ambiguous. e.g. the same message is only sent once. At beginning the components has to send broadcast messages to dynamically discover other components in the system.

2 Project requirements analysis

2.1 Architecture model

Client-Server model is used for the system. There are several servers and clients in the system. Each client is connected to one server and each server can have many clients. There is one leader among the servers, which takes care of the new participants and redistributes the servers to clients if some server fails. Clients are the users who participate in chat, servers are responsible to transfer the messages to their receivers i.e. other clients. Clients can be reconnected to the server with best performance by the leader at any time. The consistency of the whole system is granted by using the raft algorithm, which includes leader selection and communication among servers. The servers has one of the three roles: leader, candidate and follower. The leader is responsible for communicating with clients and sends messages to the other follower, followers are the servers which receive the heartbeat within a certain timeout. If a follower doesn't get the heartbeat message within the timeout, it will become a candidate and start asking votes from all other server.

2.2 Dynamic discovery

The system is not restricted to a fixed number of clients/servers. Participants addresses should not be hard-coded. New clients and servers should be able to join the system properly. There is one leader selected by other servers. The leader will broadcast heart beat messages to get information about its followers. When a new server comes online, it will wait for the leader's heartbeat message till timeout. The client goes online and connected to the first server responses it, after the connection, the client will regularly check for new messages for it by asking the connected server. Every server has a maximum number of clients to connect, if a sever has already the maximum number of clients, it will not answer to other clients.

2.3 Fault tolerance

Every follower has a heartbeat timeout, which is the maximum time without receiving a heartbeat message from the leader, if this time exceeds the follower becomes a candidate and try to ask for votes from other followers. If it gets the majority of the votes it is then elected as the new leader. If one message is missing, the sender has to discover that by using e.g. timeout mechanism, and resend that message to the corresponding receiver until it gets the corresponding acknowledgment. If one server is offline, the corresponding clients will look for new server to connect.

2.4 Voting

If a follower doesn't hear from the leader within the timeout, it becomes candidate and asks other servers for vote. If it gets more than half of the votes, it will be the new leader among all servers.

2.5 Ordered reliable multicast

The user sends message to the leader and then the leader will store the message and sends it to all followers, then the receiver check the server to gets its message. Total ordering is preserved since all updates are made by the leader in the order of receiving those messages.

3 Architecture diagram

For a client to join the system, it has to register namely login by one of the server.
In one to one chat, the client can sent to a other client message, via a server, which is assigned by the leader of the servers.
In group chat, a group of clients share the same messages and broadcast messages within the group.

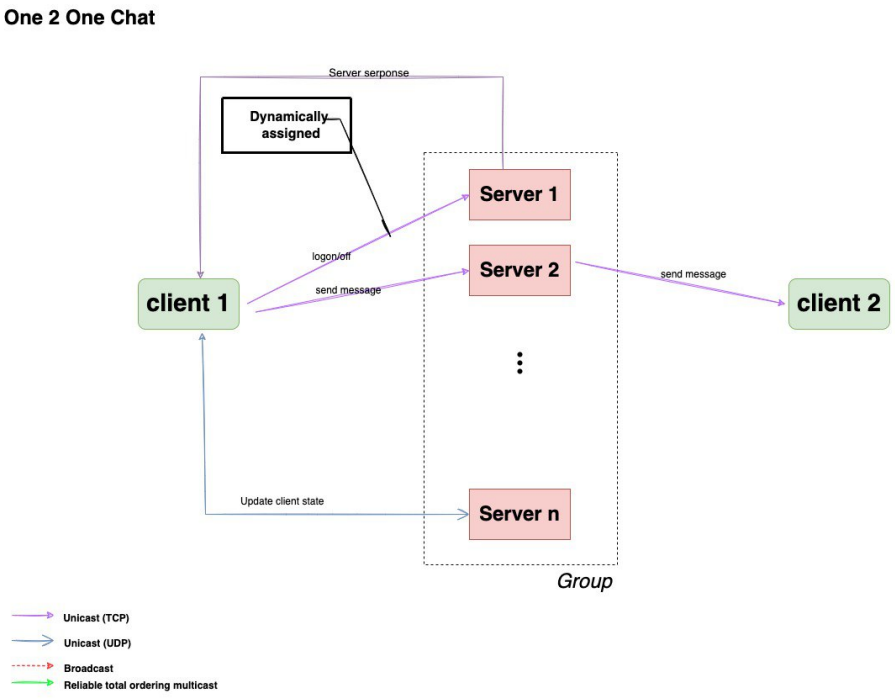

Fig. 1. Architecture diagram of one to one chat.

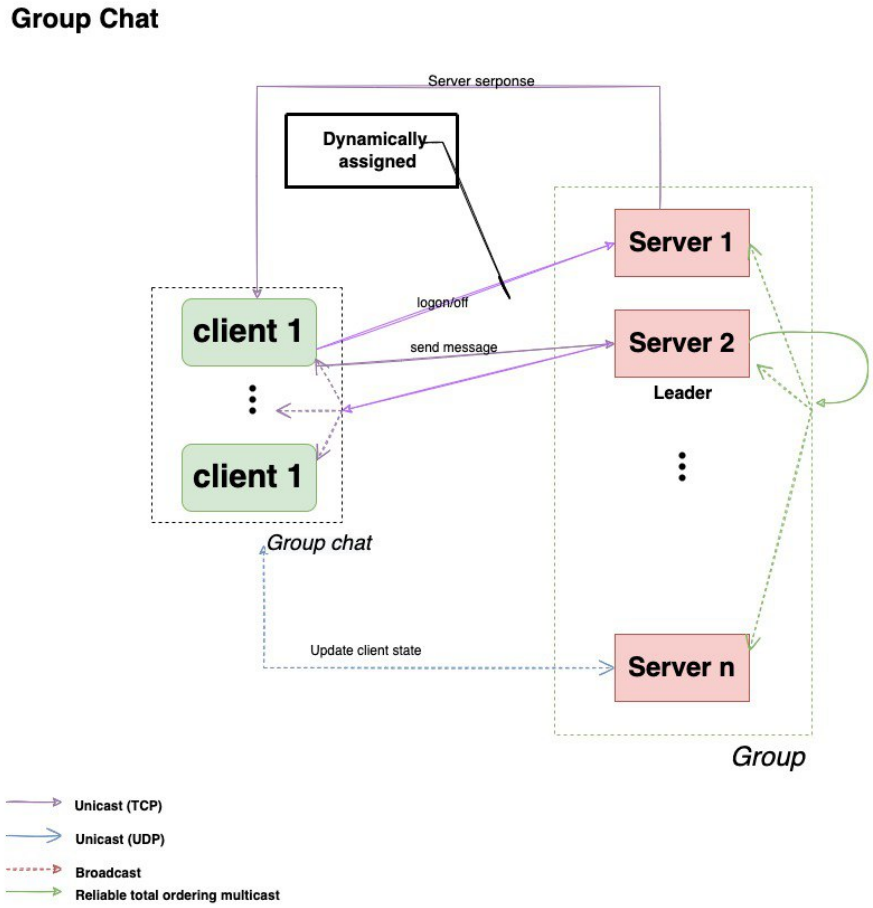

Fig. 2. Architecture diagram of group chat.