Distributed SystemsCOMP90015 Project

Extended Multi-Server Chat System

Yixin Chen(alachen) 522819
Derui Wang(deruiw) 679552
Shuangshuang Wang(sswang1) 751689
Fangqing Zhang(fangqingz) 767712

1. Architecture model

The corresponding architecture design has been attached in appendix.

2. Communications between components

All components communicate with each other by JSON messages. The detailed communication procedure is included in the sequence diagram which has been attached in appendix.

(1) Initially, we need to start servers are predefined in the config.txt file. If there are new servers coming in, new server will write its own configuration into config.txt. Then new server will send a JSON message to other servers to remind them update their data storage.
(2) Then we need to start central server for user name and password validation. Central server will receive client input and check with the data stored in Userinfo.txt. If username and password match with the record, central server will send a JSON message back to client to approve his login.
(3) Heartbeat thread starts when the server is active. Heartbeat thread within each server will send heartbeat signal (a JSON message) to each existing servers to check their activation status. Server will receive heartbeat reply (a JSON message) from each server including their response.
(4) Client sends request via JSON message and get response from server by the same way.

3. Technologies

(1) JSON: JSON is a lightweight format that is commonly used for data interchanging. Its own predefined structure is very easy to handle. JSON can represent two structured types which are objects and arrays. These two structures are very convenient for implementing functions for our project.
(2) Java Swing - GUI
Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes which means less dependencies. As Java swing is being published very early, it has many elements can be used. However, it is hard to learn and very old-fashioned these days. For further development, we may choose another framework instead.
(3) Security
For handling security issue, we decide to use the central server idea to store predefined client information including their username and password. There is a "Userinfo.txt" file to store every predefined client information.

When a client wants to login to the chatroom application, he has to input the username and password. The program will check with the stored information and reply messages to client of whether to let him access to the application.

To encrypt communication, we use the SSL protocol which is intended to provide a flexible means for clients and servers to communicate using a secure channel. The protocol allows servers and clients communicate in a way that is designed to prevent eavesdropping, tempering and message forgery. It includes two sub-protocols: the record protocol and "handshake" protocol. Both of them allow a client to authenticate a server and establish an encrypted SSL connection. Once the server has been authenticated, the client and server establish cipher settings and a shared key to encrypt the information they

exchange during the remainder of the session. In our case, we use the two-way authentication mechanism, which means that not only the client need the authentication certificate from servers but also servers need the clients' certificate. We think that two-way authentication is more secure than one-way as this way makes sure clients are from a trust platform.

However, the choice between these two mechanisms depend on the real world cases. If this chatroom application is used by online normal users, we will choose one-way authentication because it is not necessary to check specific client's identity. If this application is used by a company privately, it is more secure to use two-way authentication as we need to make sure the confidentiality of communication within the company.

(4) Failure handling

To handle crashes for servers, we use heartbeat signals to detect server status.

The heartbeat thread will start after each server is established. Our system allows 10 seconds to ask admin to start all servers, while this can be changed manually. After the server is active for 10 seconds, server will send heartbeat signal (A JSON message) to other servers to check their status. After a server receives heartbeat signal message, it will reply with a heartbeat reply message (A JSON message) to report their status to other servers. At the same time, the server will cut down the connection with the previous server for short connection. When a server receives a heartbeat reply message, the server will update its own data structure and label the corresponding server to be running normally.

(5) Scalability

To handle scalability issue (adding new server), we decide to ask system admin to add new servers into system manually. To implement this, we add a judgement mechanism in main server class.

When admin staff try to type command in command line. The program can tell the difference from whether admin wants to add a new server or start the existing server in configuration list. If the admin wants to open a new server, the command line needs to include "-n" (specify server id), "-l" (specify config.txt path), "-h" (host address), "-p" (specify port number) and "-c" (specify coordination number). After typing in the right command, the program will write new configuration into config.txt file automatically. If admin wants to open an existing server in config.txt. Admin only needs to type in server id and config.txt path. Then the program will open a normal server as previous project requested.

After adding new server, new server will send a JSON message to other existing servers, to remind them to update their stored current server info and the room list. After updating the messages, new server could have fully functional components as normal server does.

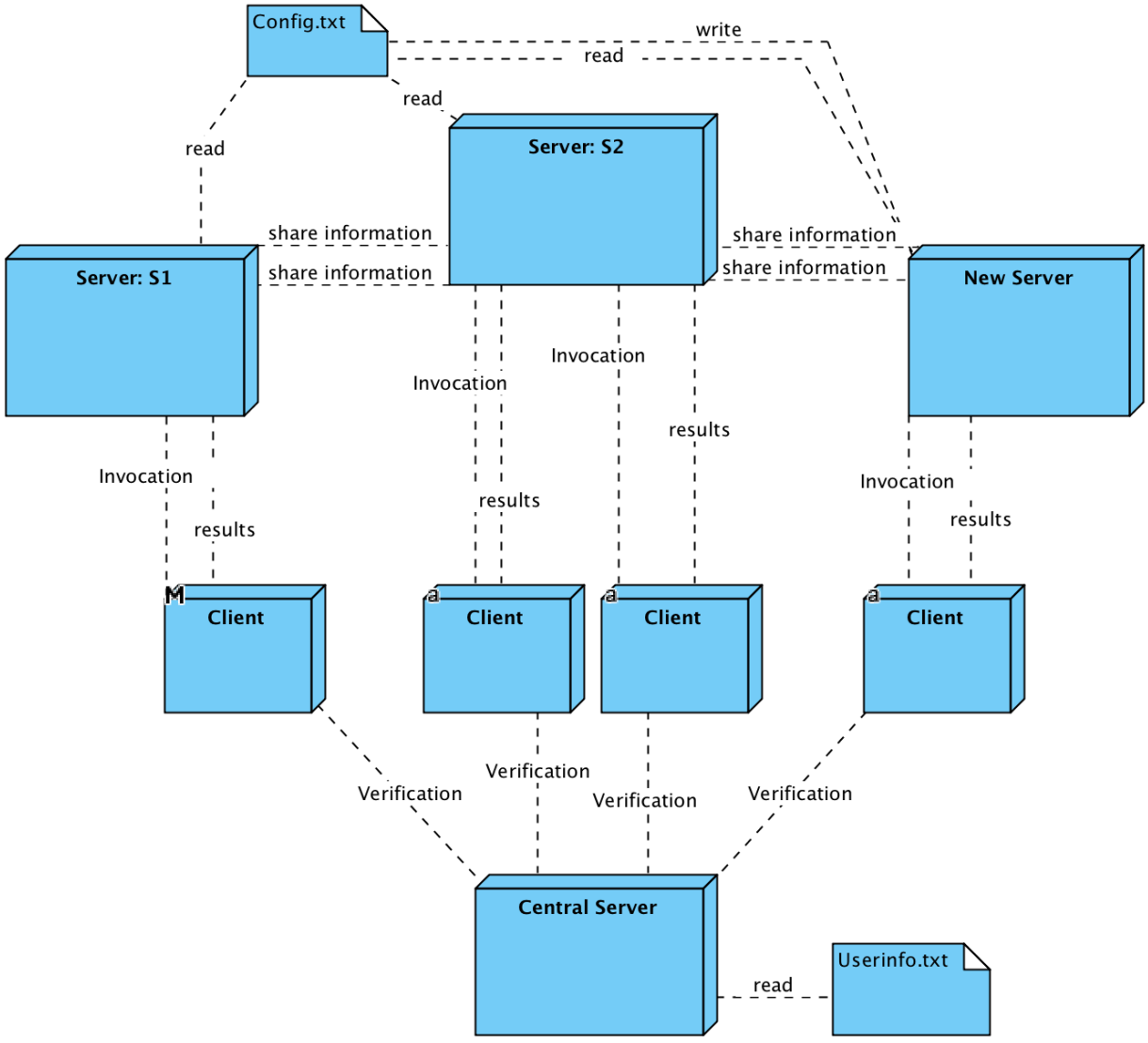
4. Work distribution

Jennifer is the team leader who distributes works to team members. She personally handles GUI design. Alan designs failure handling. Derek handles scalability issue. Fangqing Zhang handles security issue. After each parts finish, all team members contribute to integrate codes. Derek writes the report and all team members help with

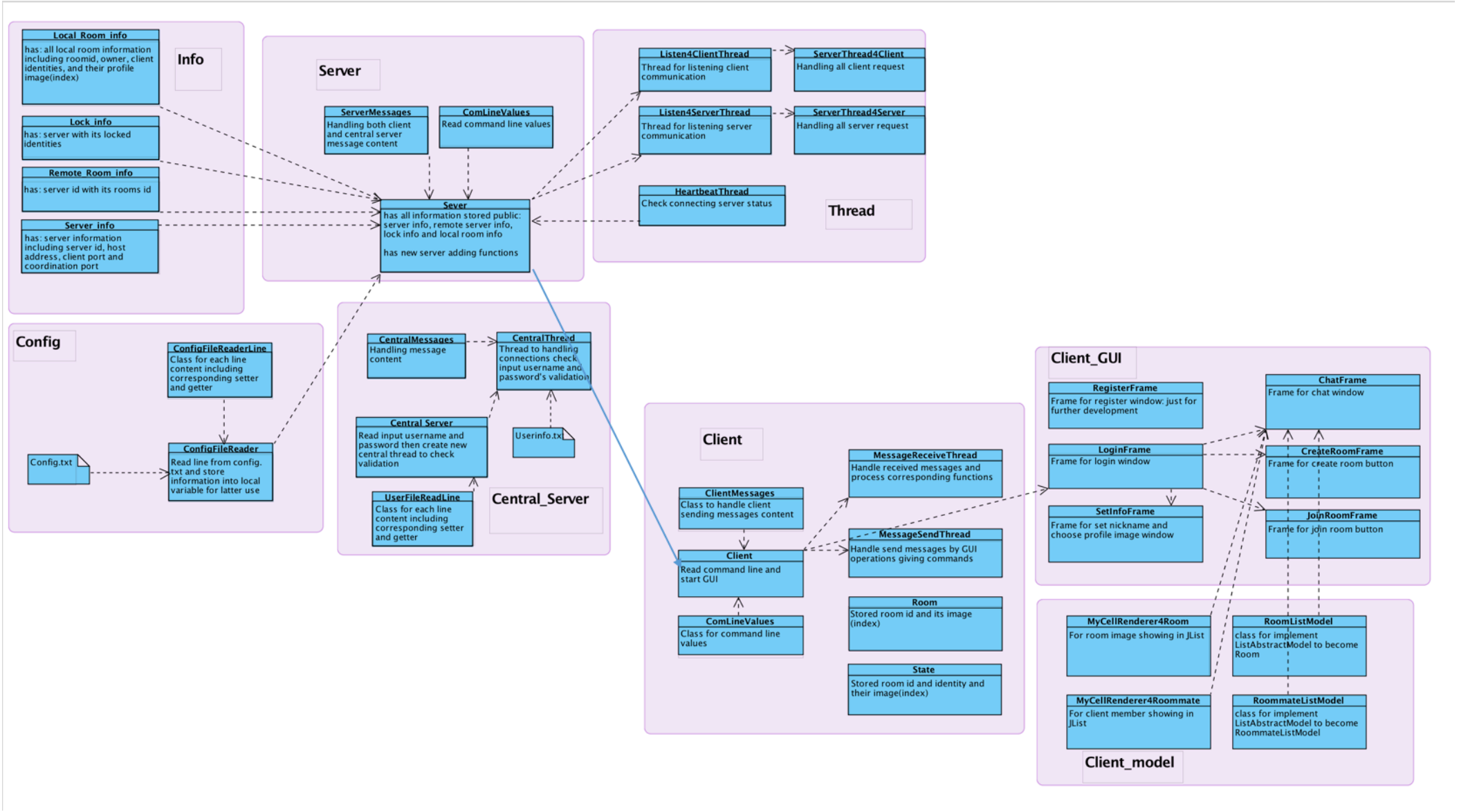
proof reading.

5. Appendix

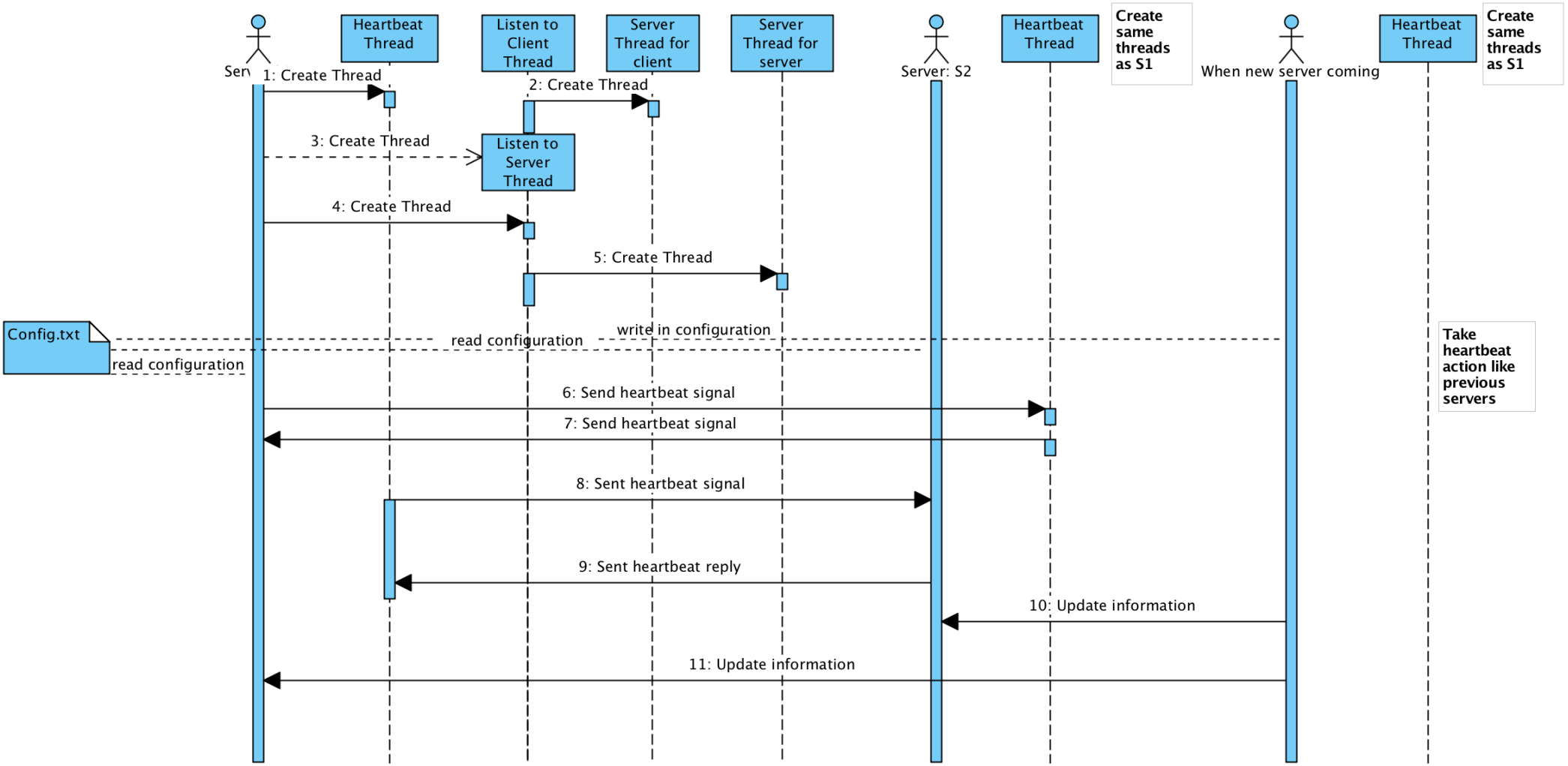
1. Architecture Design

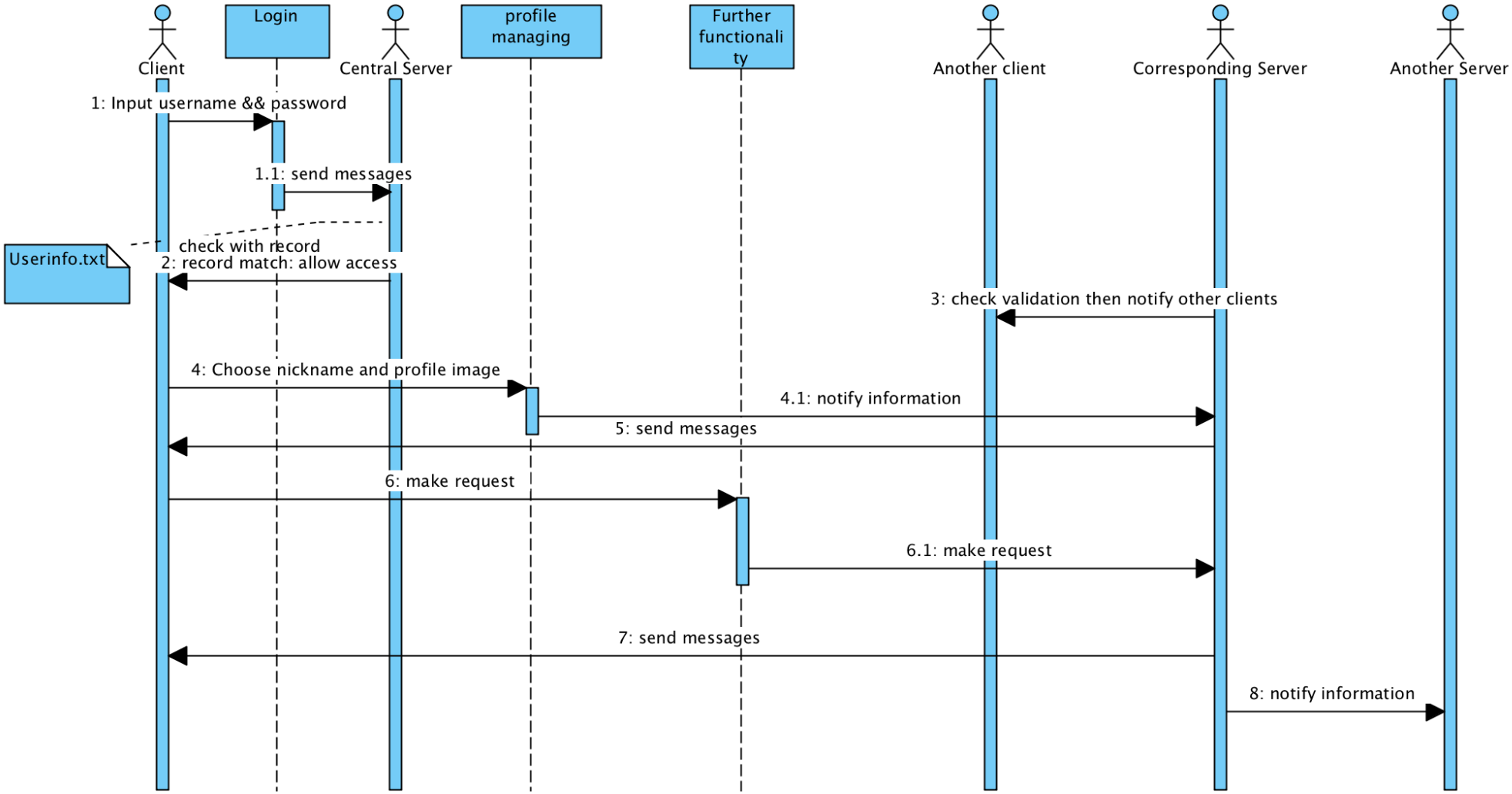


1. Architecture Design



2. Sequence diagrams




Client
Login
Central Server
profile managing
Further functionality
Another client
Corresponding Server
Another Server
1: Input username && password
1.1: send messages
check with record
Userinfo.txt
2: record match: allow access
3: check validation then notify other clients
4: Choose nickname and profile image
4.1: notify information
5: send messages
6: make request
6.1: make request
7: send messages
8: notify information