COM90049 Project 2 Report

Name StudentID
Bing Xie 741012
Chi Zhang 775016
Da Huo 791094
Jigar Thakkar 800301

1 Introduction

This report presents the extend implementation of multi-server Chat System and how it address the distributed system challenges on four aspects, namely Security, failure handling, Scalability, A Chat client client program with GUI.

2 Security

To address security, SSL TCP connections and an authentication mechanism have been implemented.

2.1 SSL TCP communication

Security Socket Layer (SSL) TCP communication is one of security method for Network programming. Java Secure Socket Extension (JSSE) provides a set of packages which implement a version of SSL protocols. Two methods of constructing SSL communication are implemented and successfully worked separately, namely bidirectional and one-way authentication.

2.1.1 One-way authentication of SSL communication

One-way authentication SSL of SSL communication as described in tutorial, the basic implementations contains three parts. Firstly build a server certification and store the keys and certificates in the keystore. Secondly, define the key store location in the server and define trusted certificate for client. Finally, change all socket and serversocket to SSLsocket and SSLserversocket created by SSLsocket and SSLserversocket factory.
However, for this distributed system, server should communicate with other server. In this case, server will play a role of both client and server. So in the server part, key store location and trusted certificate should be both defined in server as shown in Figure 1

```
public static void main(String[] args) throws CmdLineException, Number
    /*------------------------------------------------------------------*
    //security ssl serversocket prepare
    System.setProperty("javax.net.ssl.keyStore","mykeystore");
    //Password to access the private key from the keystore file
    System.setProperty("javax.net.ssl.keyStorePassword","123456");

    // Enable debugging to view the handshake and communication which
    System.setProperty("javax.net.debug","all");
    System.setProperty("javax.net.ssl.trustStore", "mykeystore");
```

Figure 1. Parts of codes in server of One-way SSL TCP communication

2.1.2 Bidirectional authentication of SSL TCP communication

The mechanism for server is that server should have a digital certification. When a client connects to server, it will get the digital certification and make a judgment whether the certification deserves trust. If client trust the certification, a connection will be constructed, otherwise be failed. In terms of server, a digital certification is generated using keytool with RSA algorithm, given password in JAVA environment. And the certification is stored in a given certification warehouse server_ks as shown in Figure 2. Then modify the code in server of server, apart from setting system property as shown in Figure 3, all serverSocket should generated by SSLServerSocketFactory. The main function of codes of Figure 3 is to import and the client certification and to be used by server.

```
C:\Users\Administrator>keytool -genkey -v -alias server -keyalg RSA -keystore ./server_ks -dname "CN=loca
lhost,OU=cn,O=cn,L=cn,ST=cn,C=cn" -storepass server -keypass 123123
```

Figure 2. Command to generate certification warehouse "server_ks"

```
private static String SERVER_KEY_STORE = "server_ks";
private static String SERVER_KEY_STORE_PASSWORD = "123123";


public static void main(String[] args) throws CmdLineException, NumberFormatExceptio
    /*------------------------------------------------------------------*/
    //security ssl serversocket prepare
        System.setProperty("javax.net.ssl.trustStore", SERVER_KEY_STORE);
        System.setProperty("javax.net.debug", "ssl,handshake");
        SSLContext context = SSLContext.getInstance("TLS");  //protocal
        KeyStore ks = KeyStore.getInstance("jceks");
        ks.load(new FileInputStream(SERVER_KEY_STORE), null);
        KeyManagerFactory kf = KeyManagerFactory.getInstance("SunX509");//algorithm
        kf.init(ks, SERVER_KEY_STORE_PASSWORD.toCharArray());
        context.init(kf.getKeyManagers(), null, null);
        ServerSocketFactory factory = context.getServerSocketFactory();
```

Figure 3. Parts of code for SSL connection of Server

Since server has established with SSL TCP

connection configuration, the original client must have SSL TCP connection configuration. Similar to server, in terms of client, a digital certification is generated using keytool with RSA algorithm, given password in JAVA environment. And the certification is stored in a given certification warehouse client_ks as shown in Figure 3. The main function of code in Figure 4 is to import and the server certification and to be used by client.

```
c:\Users\Administrator>keytool -genkey -v -alias client -keyalg RSA -keystore ./client_ks -dname "CN=loca
lhost,OU=cn,O=cn,L=cn,ST=cn,C=cn" -storepass client -keypass 456456
```

Figure 4. Command to generate certification warehouse client_ks

```
private static String CLIENT_KEY_STORE = "client_ks";

private static String CLIENT_KEY_STORE_PASSWORD = "456456";

public static void main(String[] args) throws IOException, ParseException,
    /*------------------------------------------------------------------*/
    // Set the key store to use for validating the server cert.
    System.setProperty("javax.net.ssl.trustStore", CLIENT_KEY_STORE);
    System.setProperty("javax.net.debug", "ssl,handshake");
    SSLContext context = SSLContext.getInstance("TLS");
    KeyStore ks = KeyStore.getInstance("jceks");
    ks.load(new FileInputStream(CLIENT_KEY_STORE), null);
    KeyManagerFactory kf = KeyManagerFactory.getInstance("SunX509");
    kf.init(ks, CLIENT_KEY_STORE_PASSWORD.toCharArray());
    context.init(kf.getKeyManagers(), null, null);
    SSLSocket socket = null;
    String identity = null;
```

Figure 5 Parts of code for SSL connection of Client

As server_ks holds the server certification and clients_ks holds the client certification, server certification should be export from server_ks and import it to clients_ks as shown in Figure 6, and client certification should be export from clients_ks and import to clients_ks as shown in Figrue 7.

```
c:\Users\Administrator>keytool -export -alias server -keystore ./server_ks -file server_key.cer
```

Figure 6 Export client certification and import it to server_ks

```
c:\Users\Administrator>keytool -import -trustcacerts -alias server -file ./server_key.cer -keystore ./client_ks
```

Figure 7 export server certification and import it to client_ks

2.2 Authentication Mechanism

Rewrite the client part by adding an arguments password for command line and the original identity changes to username. Server stores the information of users in format Username Password Identity.

2.3 Failure Handling

Failure handling is an important part for a chatroom software. If there is no solution to handle the server crashes, the crashed servers relative information such as chatroom detail, port number, server address will still save inside other servers which could be lead to very serious problem during broadcast period. For solving this problem, our team decided to use heartbeat signal as a mark to detect which server has crashed down in the whole environment. The solution is like this: When the server opens, it starts to send new threads to other servers which have on status (we have a global config file, and the final column save the servers status. on means the server is opening, and off means server is closing) every several seconds. And if the thread is correctly created, the server will output a sentence XXX server runs normal. While if creating server process is unsuccessfully, then the exception part will detect that and output a sentence XXX server has crashes down! to tell the administrator about this problem, and after that the server who firstly detects this situation will change the crashed servers status to off and write it into config file. After this, the running servers will not detect that crashed server and crashed servers information also be deleted from running servers.

2.4 Scalability

To implement the function of scalability, the system maintains a file that records all the information of the existing servers in the whole system with an extra column representing the current status of each server. If the server is running, then the status is on. If the server is offline, then the status is off.

If a new server is added into the system, then all the information of this server will be written into the recording file and the status of this server is set to on. Then this server will read the file and broadcast its information to all the currently existing servers in this system to introduce itself to the others in the system.

There are 3 functions that use the scalability. To implement all the functions, the system has to read the file for a second time and broadcast to all the servers whose status is on, otherwise the system will send the message to an offline server and cause a connection reset exception.

newIdentity: After verifying the authorization of the connecting client, the server broadcasts the lockIdentity JSON messages to the other servers whose statuses are on recorded in the file, when the server receives all the replies from the other servers, the server will judge whether the new identity is authorized to be created.

Createroom : After verifying the authorization of creating new chatrooms compared with the local chatrooms, the server broadcasts the lockroomid JSON messages to the other servers whose statuses are on recorded in the file,

when the server receives all the replies from the other servers, the server will judge whether the new chatroom is authorized to be created. If this chatroom is created, the server will broadcast the information of this chatroom to the other existing servers in the system and the other servers will record the information of this new chatroom in a map called chatroomMap recorded the name of new chatroom and the information of the server.
List: The system will add the name of the mainhalls whose chatroom status is on from the file and add them into a list. After join this list with the chatroomMap which records all the names of the existing chatrooms except for the mainhall, the system will get the complete list of the chatrooms.

2.5 Graphical user interface(GUI) in Android and OAuth2 login with Facebook

In this busy world, no one has time to put username and password for every new application. Also, people have to remember username and password for each app and sometimes when it does not work then it creates little frustration. OAuth2 provides login/signup mechanism which simple as well as secure which overall helps to achieve better User Experience design.

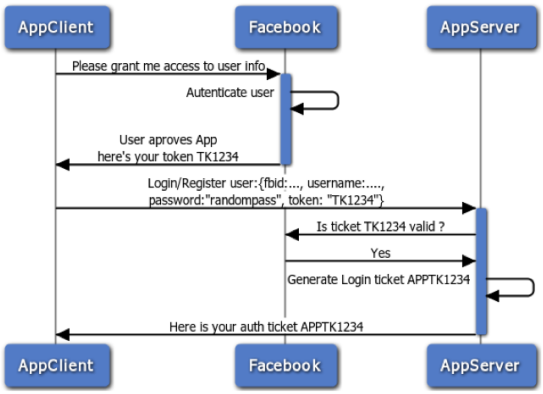
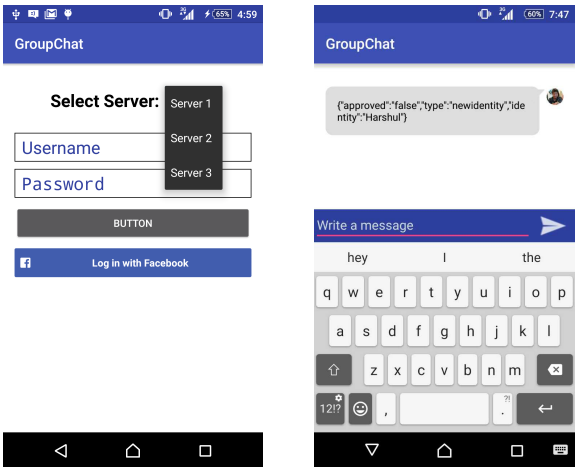


Figure 8 working of OAuth2 (Sequence diagram)
The mechanism for OAuth2 is simple, the first user taps the button of third party login (in this case Facebook) after the user has to put login details of Facebook into a pop-up web page, or if the user already has a Facebook application into their phone, then it will redirect to theirs. Facebook shows user to asked permission required by our application if user grant permission than Facebook returns that request with a token
Using that token we can ask to Facebook various details about the user. If this token is valid, then Facebook will return us asked information. This token is only valid for the limited amount of time. We can renew our token before expiry.



User Interface is an essential part of the success of any software system. Simple, bright and attractive UI design plus proper functionality, usability, accessibility extends user experience. I have made UI in Android which shares 84.60% market share of global smartphone users. [1] and in Android, this app can run more than 80% of all android os versions. There is a significant advantage for making smartphone application software than desktop because smartphone software can be used my many people than desktop software. I have used Google material design for designing of components. Android UI designs are made in XML (extensible markup language). And interaction and other programming are done in Java, which is primary language for making application in android.

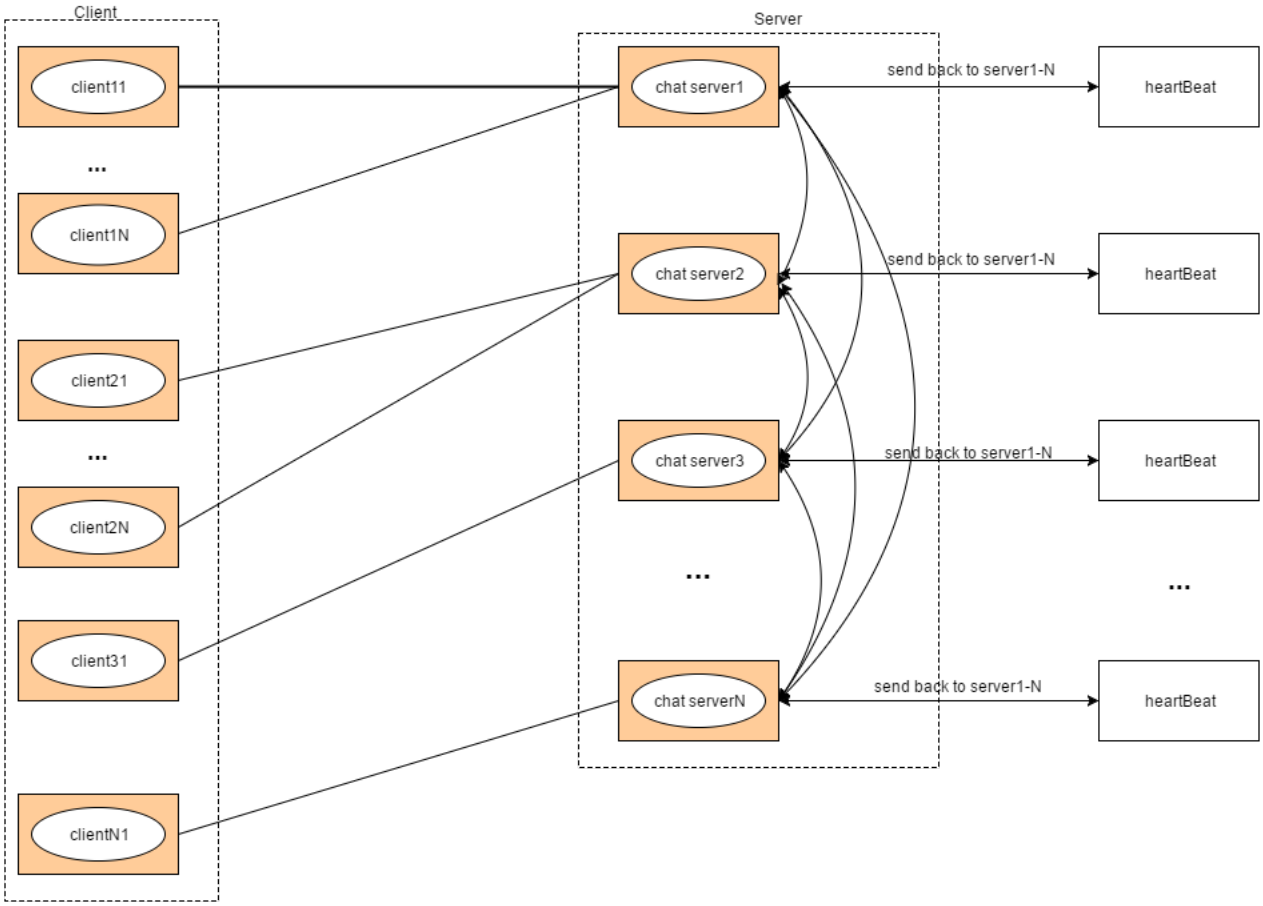
3 Contribution

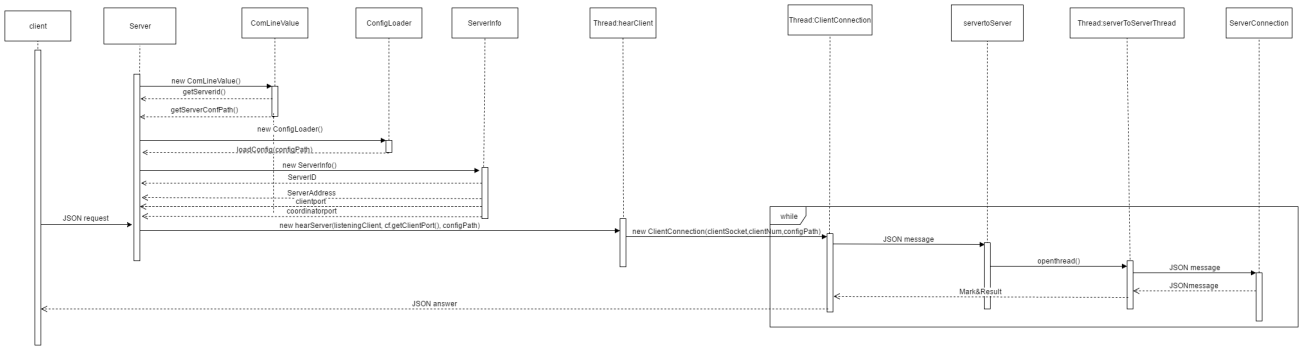
Bing Xie	Scalability,GUI	Code Merging
Chi Zhang	Failure Handling	Code Merging
Da Huo	Scalability	Code Merging
Jigar Thakkar	GUI,Cloud	OAuth2

4 Reference

[1] 1. http://stackoverflow.com/questions/23794501/how-to-use-oauth-with-forms-auth

5 Appendix


Client
Server
client11
client1N
client21
client2N
client31
clientN1
chat server1
chat server2
chat server3
chat serverN
send back to server1-N
send back to server1-N
send back to server1-N
send back to server1-N
heartBeat
heartBeat
heartBeat
heartBeat


client
Server
ComLineValue
ConfigLoader
ServerInfo
Thread:hearClient
Thread:ClientConnection
serverToServer
Thread:serverToServerThread
ServerConnection
new ComLineValue()
getServerId()
getServerConfPath()
new ConfigLoader()
loadConfig(configPath)
new ServerInfo()
ServerID
ServerAddress
clientport
coordinatorport
JSON request
new hearServer(listeningClient, cf.getClientPort(), configPath)
new ClientConnection(clientSocket,clientNum,configPath)
while
JSON message
openthread()
JSON message
JSONmessage
Mark&Result
JSON answer