COMP9331 Assignment One

Student Name: Chunnan Sheng

Student Code: z5100764

ScnChat 1.0

### Brief Introduction

ScnChat is an online chat program developed by JAVA with socket interfaces from java.net and java.io. This program has a Server module and a Client module as two main components. The client can be launched by multiple users simultaneously while the Server could only run on one machine as a single instance.

This program has covered all the requirements documented in the specification of Assignment One including both mandatory features and extended features. In addition, extra P2P commands are introduced in **ScnChat 1.0** for user convenience: ***stopprivate <user>*** to terminate the P2P connection and ***privatewhoelse*** to check online P2P users connected with this client.

### Software Design Overview

**Structure of the Server Structure of the Client**

**Client**

Entrance of the Client program

**ClientP2PListener**

This listener waits for P2P connections from other clients.

**ClientP2PSession**

The client entity holds a hash table to manage all P2P sessions with other Clients

**ClientConsole**

Console command lines of the Client

**ClientP2PReceivingSession**

This session focuses on receiving messages from the P2P remote.

**ClientPeerReplySession**

This session focuses on receiving messages from other clients via the server.

**ClientEntity**

Main instance of the Client program

**ClientP2PSendingSession**

This session focuses on sending messages to the P2P remote.

**Server**

Entrance of the Server program

**ServerConnectionSession**

The server entity holds a hash table to manage all connection sessions with Client. Each connection session manages all requests from each client.

Messages sending to the destination client will be inserted into the reply session’s message queue.

**ServerUserProfile**

Profile of each user together with the password is stored in a hash table of the ServerEntity. Login failures, log out timestamps and blacklist of blocking will also be recorded here.

**ServerEntity**

Main instance of the Server program and listens to new connection requests from clients

**ServerPeerReplySession**

This session checks its message queue regularly and picks up messages if there are. And then sends them to the destination client.

### Instruction of the protocol

**Comparison of packets between application (ScnChat) level and transport (TCP) level**

JAVA socket stream allows data buffering and the “flush” function can be executed if it is time for the program to send out all the buffered data. In other words, multiple APP level requests may be included in a single TCP packet.

On the other hand, one request of APP level could be chopped into multiple TCP packets if the message sent by the client is too long.

Therefore, each APP level packet should be structurally designed with headers and properties, so that the software can determine the beginning and the ending of each request.

|  |  |  |
| --- | --- | --- |
| **TCP packet 1**  Request 3  Request 2  Request1 | **TCP packet 2**  Request 4 (chopped) | **TCP packet 3**  Request 4 continues … |

* 1. **Packets sent by the client (examples):**

Structure of a login request:

Properties

The header

**Client ScnChat 1.0\r\n**

**Login\r\n**

**User: Mike\r\n**

**Password: 12345\r\n**

Structure of a message sent from a client to another client via the server.

The message payload

Properties

The header

**Client ScnChat 1.0\r\n**

**Message\r\n**

**Host: 192.120.0.125\r\n**

**Sender: Mike\r\n**

**Receiver: Lily\r\n**

**Sent Time: 20170325 15:00:23\r\n**

**Content Length: 30\r\n**

**How are you? It is a nice day!**

Structure of a whoelsesince request

Properties

The header

**Client ScnChat 1.0\r\n**

**Whoelsesince\r\n**

**Time: 20170412 00:54:02\r\n**

**2.2 Packets sent by the server (examples):**

A request acknowledgement sent to the client

Properties

The message payload

The header

**Server ScnChat 1.0\r\n**

**Acknowledge\r\n**

**Content Length: 19\r\n**

**Error. Invalid user**

A notification that a new client (Yoda) is online

The header

Properties

**Server ScnChat 1.0\r\n**

**Presence\r\n**

**Type: Login\r\n**

**User: Yoda\r\n**

**Time: 20170325 15:00:23\r\n**

**2.3 Packets sent via P2P connections (examples):**

A application from one client (Yoda) to another client for a P2P sending connection

Properties

The header

**Client ScnChat 1.0\r\n**

**P2PSend\r\n**

**User: Yoda\r\n**

A message sent privately from a client to another client

The header

The message payload

Properties

**Client ScnChat 1.0\r\n**

**P2PMessage\r\n**

**Host: 192.120.0.125\r\n**

**Sender: Mike\r\n**

**Receiver: Lily\r\n**

**Sent Time: 20170325 15:20:25\r\n**

**Content Length: 34\r\n**

**We don’t let other people know us!**

### The multi-thread mechanism of the server

Each connection session of the server receives messages from the client and distribute these messages to other connection sessions’ message queues according to whom the message should be sent to. Sending thread of each session would check its message queue regularly, remove messages and send them to its client. Synchronizations between threads are needed for this mechanism.

Server

Connection2

Connection4

Connection3

Client 1

Client 2

Client 3

Client 4

Connection1

### The multi-thread mechanism of P2P connections

A client could establish P2P connections with multiple clients. Messages send by this client’s terminal would be distributed to the right P2P connections’ message queues according to whom these messages should be sent to. Each P2P connection session checks the queue regularly, removes messages from its queue and sends them to the corresponding receiver.

Client A

Connection2

Connection4

Connection3

Client B

Client C

Client D

Client E

Connection1

Terminal (Console)