Design document

1. System design

This project uses the Java programming language to implement a peer-to-peer (C/S architecture) instant messaging system, including server and client programs. As an instant messaging program, the client and server are two essential parts of this system. Socket and ServerSocket provided by Java were used to implement the low-level data transmission part of the client and server respectively.
The two programs on the network can exchange data through a Socket connection. The server uses ServerSocket to listen on a specific port. When the client program connects to the server's port, the establishing produces of a Socket is completed. Then I/O streams can be obtained from this Socket to do further communications.
In this situation, the server needs to have a connection with multiple clients, which means the server have to keep multiple socket simultaneously. Therefore, the communication with the multiple clients can be achieved by opening a separated thread after each connection from clients was accepted. In this way, multiple clients can be accepted at the same time, implementing the main functionality of server program.
The functionalities of server include five client commands listed in the requirements and an additional client login command. A simple event dispatcher was implemented in the processing of supported commands, by obtaining the event handler class in the corresponding function package through the reflection of Java. Then the message packet received from the client will be send to the unified implementation interface, collecting returned message from event handler. Finally, the processed result is sent back to the client.
An open sourced Tars (https://github.com/TarsCloud) protocol is used as a serialization and deserialization library for messages packaging, which is implemented by Interface Description Language (IDL), which is a binary, extensible, code-generated, multi-platform protocol that enables objects running on different platforms and written in different languages to communicate with each other in the form of RPC remote calls.

2. Network packet description

The structure of a network packet is shown in the following table.

1Byte	1Byte	1Byte	1Byte	1Byte	4Bytes	4Bytes	…	1Byte
Packet Head	Version	Packet Head	Packet Tail	Packet Type	Packet Sequence	Data Length	Data block	Packet Tail

The part of the network packet marked as cyan background is its head and tail, indicating the validity to program. Then we have 6 parts marked as light gray containing some metadata of a packet. The packet length is total length of one packet, while type indicate the way program determine the application of it. And the packet sequence keeps the order of communication between the server and the clients, for example the reply packet form server will be same sequence as the request packet sent by the client.
Afterward, the server will read the data with length descripted in its metadata, and then check the remaining packet tail to verify it, only the valid packet will be processed continually by deserialization library. After deserialization, the message class will be created and sent to the event dispatcher.

3. System advantages

- The system and business logic processing structure are clear, and it is convenient to continue to develop new functions and maintenance.
- The event dispatcher can be updated to more distributed mode, for dividing business logic to more servers for processing under concurrent situation.
- The use of a binary-encoded serialization and deserialization library guarantees efficiency as the amount of information transmitted is drastically reduced.

4. System disadvantages

- The encryption was not taking into account, the information transmission process is weak to attack.
- The network packet is not integrity checked, and the information may be changed during transmission.
- The processing of a network packet is too complicated and may put higher pressure on the server under concurrent situation.