Report **Chenyuan ZHANG**
Dictionary System Student ID:815901
Prof. Rajkumar Buyya Email Address: chenyuanz@student.unimelb.edu.au
Tutor: Xunyun Liu Tutorial Time: Monday 5:15-6:15

Problem Statement

Using a client-server architecture to implement a multi-threaded dictionary server which can support multiple clients perform operations concurrently. Sockets and threads should be explicitly used in the program.

Functions should be provided by the server includes adding a new word, removing an existing word and querying the meaning(s) of a given word.

For client, a GUI is required to give clients convenience to perform actions.

Design

Server

The server is a multiple thread program with thread-per-connection architecture. The advantage of this design is client can be identified by server and the network traffic will reduce. The disadvantages of thread-per-connection is that it cannot handle a large number of connections and it will spend additional time and resources to create and destroy threads. But this problem will not affect the performance when the amount of users is relatively small.

When the server is launched, a main thread will be used to monitor the port for incoming connection requests and a save thread will be used to save the dictionary to the file every second. After a connection is accepted, two threads are created: one is to listen to the client messages and place them in a command queue, another is to monitor the command queue, process the command and send replies to the client. In addition, a singleton object is used to hold the server state and perform operations on the dictionary. Every methods in the singleton object is synchronized.

Client

The client is a two thread program. One thread is to monitor the user input from GUI and send commands to server. Another thread is to receive the messages from the server and try to reconnect the server if connection failed.

When the client GUI is launched, a login window is displayed. User need to enter the IP address and port number of server. If fail to connect to the server, a error message will be displayed and user need to enter again. If connect to the server successfully, a operation GUI will be displayed and a reader thread will also be created to listen the messages from server. User can perform "add a word", "remove a word" or "query a word" three operations. The message received from server will be showed on the information window.

For all three operations, if the connection failed, the program will show a error message after submitting command and then return to the operation GUI. If the connection works, the command will be sent to the server.

The reader thread will always monitor the socket and display the messages received on the information window as soon as receive it.

Figure 1: Server UML Diagram

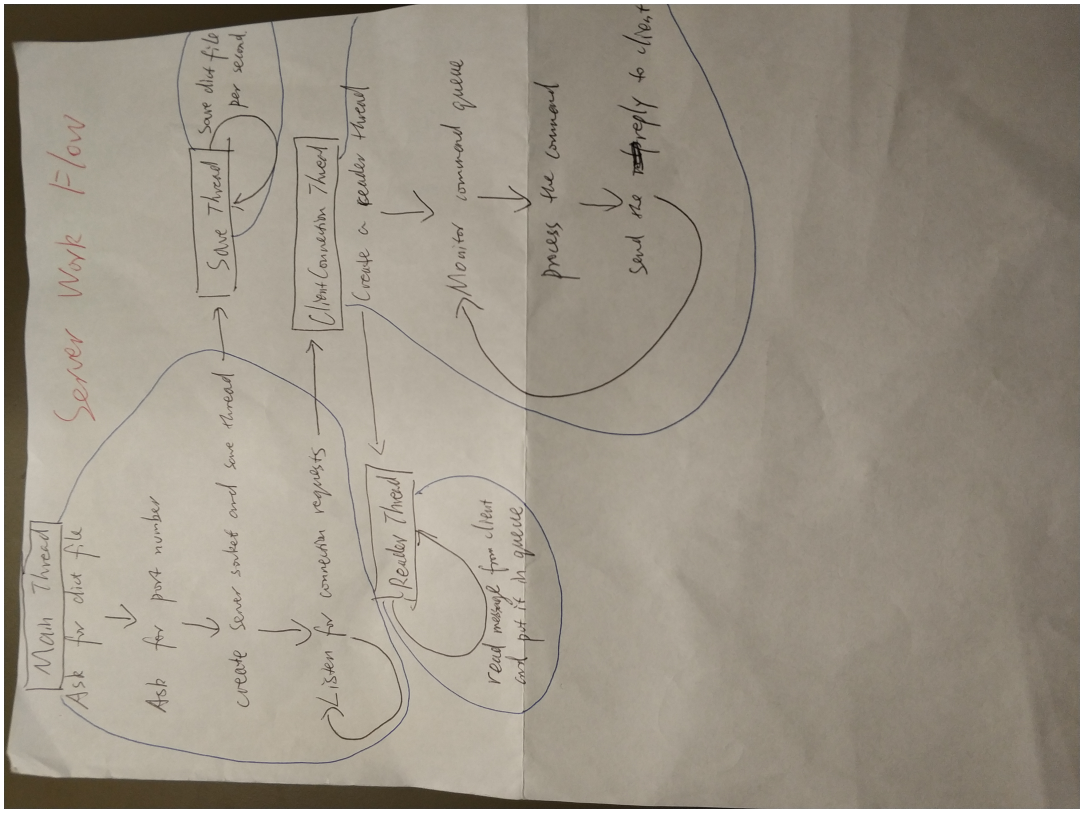

Figure 2: Server Work Flow

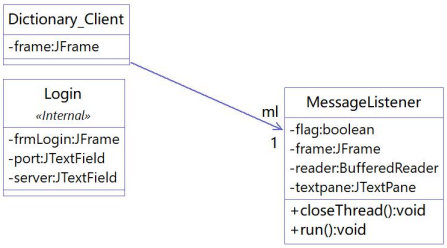

Figure 3: Client UML Diagram

Communication protocol

When a server accepts a connection request, a welcome message will be sent to the client. Then the user can send the commands to the server and server will respond each command.

A command is a JSON object. The first field is "Operation", it can be "addWord", "delWord" or "queryWord". The second field is "Item". For "addWord", the value of "Item" is also a JSON object which contains two fields, "Spelling" and "Meanings". "Spelling" is a string while "Meanings" is an ArrayList of String. For the other two operations, the "item" field is just a string which represents the spelling of the word. The command will be convert into a string before send it and the server will parse it back to a JSON object.

The messages from server to client are just strings.

Invalid commands will be ignored by the server and reply a warning message(It is not supposed to occur if use the client program).

Additional features

Server

- For a successful addition or deletion, the update will be broadcast to all clients connected to the server. Other messages will be private. Public messages will show the operation owner.
- If a user want to delete or query a non-existing word, server will provide spelling correction function with minimal edit distance algorithm.
- When a client's request is accepted, the server will show the number of clients connected to the server.
- A save thread will save the dictionary to the file per second.

Client

- Private messages from server will be highlighted.
- User input will be tested whether it is a "word"(consist of letters) before sending it to the server.
- User can disconnect actively by clicking the "back to login" button. It will lead user back to the login window then user can choose another server to connect
- If the connection is failed, the client will try to reconnect with same IP address and port number every 5 seconds.

- Title of the operation GUI will show the client number of the user.

Conclusion and Critical Analysis

Overall, the advantages of the project are two aspects: first, both the client and server use the separate threads to receive message, so local functions will not be blocked when waiting for incoming messages. It can greatly improve the performance if connection congestion occurs or the local operation is time-consuming. Secondly, thread-per-connection architecture and reconnect function improve the stability of the system.

Due to the skill and time limitations, there are some drawbacks of the project to be improved.

- The appearance of GUI is not well-designed.
- A worker-pool architecture may be used to provide scalability.
- A user database may be used to give different authority to different users.
- The programming style of the project is not perfect(for example, hard coding) so openness may be lost.
- If two servers run on the same dictionary file, the unintended results will occur.