

Chat Server/Client made by C++

Mes 109201547

Department of Mathematics
National Central University

Abstract—This is a project of chat server and client made by C++. The project [2] used the asio library in boost. There are several object in server and client object helping to do some works such as connection, message passing and message analyze.

I. Introduction

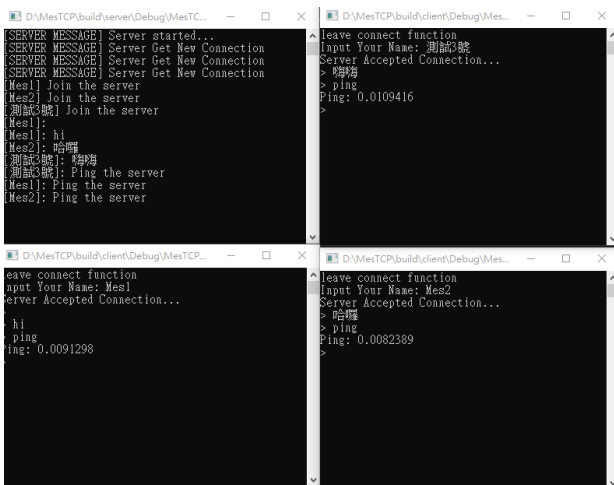
TCP/IP is the set of communication protocols in the Internet. And boost [1] is a set of libraries for C++, provides lot of support for tasks and structures such as linear algebra, multithreading, image processing, and internet. Many of Boost's founders are on the C++ standards committee, and several Boost libraries have been accepted for incorporation into standard.

In this project, I use the asio library in boost, which contains some implementations such as socket, data stream buffer and asynchronous reading and writing functions, to help me build the net. And I refer to Javidx9's software architecture on github [3] to build the whole project.

II. Usage

There are at least two process in this project, server and client. It can only launch one server, but can launch multiple clients.

In the begining of the clients process, it will ask the name of the user, then the server will display the join message. Also, the message sended by the clients will be displayed at the server.



```
SERVER MESSAGE] Server started...
SERVER MESSAGE] Server Get New Connection
SERVER MESSAGE] Server Get New Connection
SERVER MESSAGE] Server Get New Connection
Mes1] Join the server
Mes2] Join the server
Mes1] Join the server
Mes1]: hi
Mes2]: 哈哈
Mes1]: Ping the server
Mes2]: Ping the server
Mes1]: Ping the server
Mes2]: Ping the server

leave connect function
input Your Name: Mes1
Server Accepted Connection...
hi
ping
ping: 0.0091298

leave connect function
input Your Name: Mes2
Server Accepted Connection...
哈哈
ping
ping: 0.0082389
```

Fig. 1. Example of the server and client processes.

III. Program Procedure

The goal of this project is to make a chat server that clients can send message to the chat room. Thus, there are two programs in this project, server and client.

The server part needs to build a socket that listen to the client, and the client part needs to connect to the server. Once the socket be connected, the message can be passed.

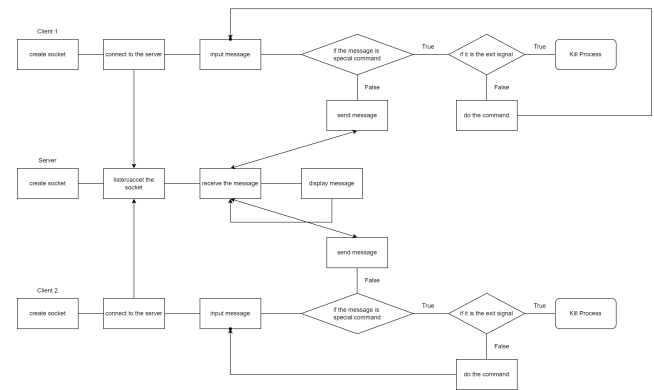


Fig. 2. Flow chart of the program

IV. Architecture

There are a base object named connection block in the architecture. It's contains a socket object, a output message buffer, an owner id that can mark who owned this object, and a reference of input message buffer. The input message buffer stores the message that the server/client want to send. And the output message buffer stores the message that the server/client received. The message controller is a abstract object that coordinates the buffer and the socket post, in my project, it actually just some functions, not an object.

The server contains an vector of connection block, an connection acceptor, an incoming message buffer, and a referenceclient list. The acceptor will help the server to link the socket between the client and the connection block in the vector, when the connection was built, the user will be denoted in the client list. When the connection block received message, it will push it into the incoming message buffer, so the buffer needs to be thread safe. And there is a abstract object called message parser in the server, it will analyze the message and decide what to do. It may resend a message to client, such as the command ping.

The client contains a connection block, an interface for user to type the message, and an incoming message buffer. The connection block will apply to the server, if the server accept it, the connection was built. Also, there is a message parser in client too, it will analyze the message and decide what to do, too. When the user enter the message, it will pass it to the message parser, and the parser will analyze the message, if it is a command, it will do the corresponding work, otherwise, it will treat it as a normal message and send it to server.

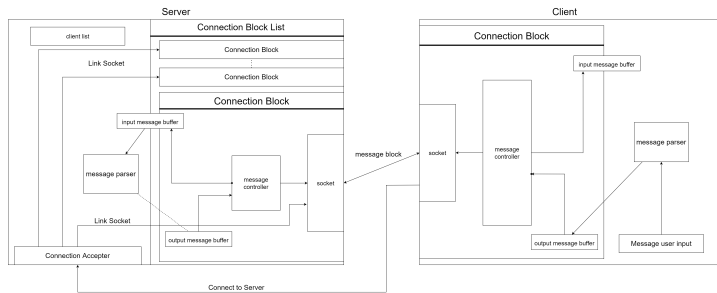


Fig. 3. The architecture of the program

The message block, which was sent by the server and clients, contains a type id, a name of the message sender, an array of message content, and a time data. The type id helps the message parser to analyze what type of this message is, such as ping command, error message or normal message.

V. Future Works

In this project, it didn't apply any application layer protocols such as HTTP and IRC. The IRC protocol is popular in open source chat application. And Currently, some chat applications use HTTP to communicate, such as Discord and Matrix. In Matrix, every message is a json file, passed via HTTP protocol.

Thus, the next target of this project is to apply one of the protocols above, make it more general useful.

References

- [1] boost libraries. <https://www.boost.org/>.
- [2] The github of this project. <https://github.com/Mes0903/MesTCP>.
- [3] Javid's network sample. <https://github.com/OneLoneCoder/olcPixelGameEngine/tree/master/Videos/Networking/Parts1%20>