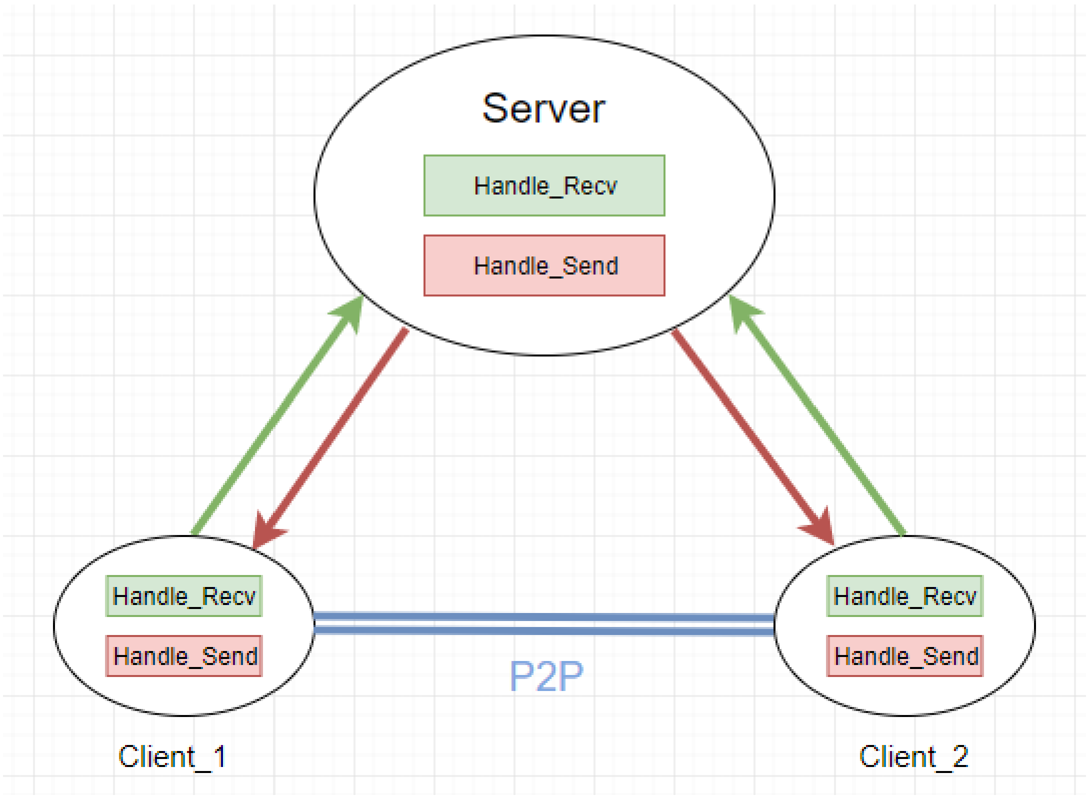
COMP 9331 Assignment

Student Name: YIJING MA

Student ID: z5190667

Language: Python 3.7

Concept map:



Picture 1: Concept Map of CS & P2P

According to the above picture. It is easy for us to find that in this project, we should use multithreading method to handle different transactions (receiving & sending).

clinet.py

There are totally 5 functions in this part. They are **private_server()**, **private_send(), handle_recv(), handle_send()** and **main()**

In the main() function:

Firstly, get IP address and port number from the command line and then create a socket to connect to server**.**

Secondly, create a list to store all possible commands in this project.

```
# Create a command list to store all possible commands
commands = ['message', 'broadcast', 'whoelse', 'whoelsesince',
'block', 'unblock', 'logout', 'startprivate', 'private',
'stopprivate']
```

Thirdly, I use a main loop to do the authentication. In this part, I need to do different operations based on the reply sent by the server. So, I use a flag e_type to record the possible error type.

1. If reply is **'Welcome to the greatest messaging application ever!'**, then authentication done, user log in successfully. This time error type is 1. So, I break the loop to do the following operations.
2. If the reply is **'No such account'**. It means that user input an invalid username, then I need to let use input username and password again, until the authentication is ok. This time error type is 2. So, I should continue loop to let user input again and again.
3. If the reply is **'Already login, cannot login again', 'Your account is blocked due to multiple login failure. Please try again later'** or **'Invalid Password. Your account has been blocked. Please try again later' later'**. It means that user has

tried 3 times, its account is blocked or this account has been logged in already, so I ne ed to shut down the whole process directly, so I just use sys.exit() to do it.

After authentication is done. I create **2 new threads thread_recv and thread_send** and set the target functions as **handle_recv and handle_send** to deal with receiving and sending transactions.

In the handle_recv() function:

This function needs 1 argument socket. The responsibility of this function is easy. It just uses a while loop to receive replies from server and print them out. There are only 3 kinds of condition we need care about:

1. **The reply is log out or timeout.** These 2 replies refer that the user is not active for a while or he/she sends logout command to server and server accept it. So, I need to shut down program at once.
2. **If the header of reply is 'Link'**, it means that this is a response to the 'startprivate <user>' instruction, the reply includes <user>'s IP address and port number.
3. **The header is 'Accept'.** This is a message that the server tells the client that **someone else wants to establish a private connection with you.** So, this client should be ready to serve as a private server, just start a new thread and the target

function of this thread is private_server.

In the handle_send() function:

This function needs 3 arguments: socket, username and commands list. In this part, I use a while loop to send messages. Every time before message is sent out. We need to check whether command is in the commands list or not. If not, this command is invalid. Otherwise, I encapsulate the message as **'sender: <message>' format**. Once user inputs **'private <user>'**. It means that the sender wants to send private message to <user>, **so we start a new thread to build connection with <user >.**

server.py

There are totally 8 functions in this part. They are authentication(), unblock(), unblock_t(), hanle_recv(), handle_send(), notify(), notify_all() and main(). In addition, there is also a class named U ser to record the information of each user.

Most of the work in this section is based on user input instructions to send messages to different clients. We just need to create a dictionary to save all online users' information. When message comes up, server will take different actions according to user instructions, sender and online conditions.

There are several key points of this part:

1. Implement a function with a timer to let the function start after a period of time. To realize it, I start a new thread like this:

```
"""Implement unblock with a timer"""
t = threading.Timer(time_block, unblock, (blocked, username,))
t.start()
```

   The first argument is a time variable. The second is the function we want to run after the set period of time. The third tuple is the argument given to the function we set before.

2. Thread synchronization (lock) mechanism.

   Every time, the handle_send() will stop at

```
if con.acquire():
    # wait to be notified
    con.wait()
```

   Only when the handle_recv() gets message from client and uses the notify()/notify_all() function to tell other thread s to run code after wait() point, the handle_send() will get message from clients and do the following operations.

3. In the handle_recv() function, we need to set a timeout Every time there is a message coming in reset timeout. The mechanism will detect whether this client is active or not. Once time out, send 'timeout' message to client to let it log out (shut down) at once.