COMP3331

ASSIGNMENT REPORT

Richard Quisumbing

z5330506

1 Program Design

1.1 Client

The client design is simplistic, purposefully done to delegate most of the command functionality to the server. Specifically, the client handles user input and parses the command and its associated arguments which forms the data of the client request. The request is then sent to the server via a TCP connection. Resulting responses from the server are displayed to the user in the client.

As for the UDP command which defines a peer-to-peer interaction utilising a UDP connection, the client implements a server thread (if they are the audience) called UDPServerThread which remains open throughout the user session. On the other hand, if the client is acting as the presenter in the peer-to-peer connection, a new client thread called UDPClientThread is created that sends the desired file to the audience client, which ends as soon as all the bytes of the file has been sent.

1.2 Server

The server is responsible for identifying the type of command inside incoming requests from clients inside its threading class ClientThread, capable of handling multiple clients simultaneously. It is also directly linked to the server data structure defined in the ServerData class in server_data.py. When requests are received, the server identifies the command, and handles the majority of the command functionality including user login. Each command is handled within its own function, usually with the naming scheme that looks like *handle_command_BCM()*. Inside these functions, error checks are first implemented, then interacting with the server data structure to ensure data is preserved, and finally sending the response back to the relevant client.

1.3 Server Data Structure

The server data is a class defined in server_data.py, which is instantiated as soon as the server is up and running, and is responsible for preserving the data sent by clients. Additionally, the data structure automatically loads the users inside credentials.txt, classifying them as registered users as soon as the ServerData class is instantiated. This means that if the server owner wants new users to be able to login, the server would need to be restarted with the new users added to the credentials.txt file.

1.4 Validate.py

Validate.py contains simple validation functions that verify certain aspects of command requests, such as verifying command argument syntax and certain error scenarios.

2 Message Format

Messages are mainly formatted as key-value pairs inside python dictionaries.

2.1 Client Requests

```
request = {
    "type": type,
    "user": user,
    "data": data,
}
```

Type: Specifies the type of request being made (usually just 'command'), which lets the server know how the request is going to be handled.

User: Specifies the username of the user that made the request to the server.

Data

The data for a command request has its own dictionary format:

```
command_data = {
    "command": command,
    "command_args": command_args,
}
```

The "command" key contains the input command supplied by the user, and the "command_args" contains a list of the command arguments that were supplied alongside the command.

2.2 Server Response

```
response = {
    "command": command,
    "user": user,
    "status": status,
    "data": data,
    "keep_alive": keep_alive,
}
```

Command: The command for which the request was initially made for.

User: The username of the user that made the request.

Status: A Boolean value indicating whether the command was successfully executed.

Data: Contains the requested data.

Keep-alive: Contains a Boolean value of whether the connection should still be kept alive after the command has been executed.

3 Program Description

3.1 Version

Python 3.10 which should work for Python 3.7 on CSE machines.

3.2 Usage (On CSE machines)

Server

python3 server.py <server-port> <number-of-consecutive-failed-attempts>

Client

python3 client.py <server-IP> <server-port> <client-udp-server-port>

3.3 Pickling

The programs server.py and client.py utilise python pickling to convert python objects into byte streams, necessary to transport data as bytes since the sockets only send byte streams.

3.4 Authentication

Authentication functionality is handled as soon as a connection is established between the server and the client. The server first requests the client to provide a username, and the client supplies it. This is done infinitely until the username is a valid one. Next, the server then waits for the password for the given username, keeping track of how many more permitted password attempts are available for that particular user. If it the client has exceeded the password attempts, then the user is locked out and cannot log in for another 10 seconds.

3.5 Commands

Each command between the client and the server is handled in the following format:

1. Client sends request.
2. Server identifies command request and calls the relevant function that handles the command.
3. Server then sends a response to the client, notifying the client of whether an error has occurred or the command was successfully handled by the server.

3.6 Peer-to-Peer UDP

Each client can be run on separate directories, and files that the client wishes to send to another client must be inside the same directory as the client program. The client server thread is always running, indicated by the while loop.

During a file transfer

First, the client UDP thread sends a dictionary object to the server UDP thread running on the audience user client, which contains the filename being sent as well as the username of the user who is sending the file. Then, the client UDP thread sends all bytes of the file, as well as a final empty byte to the server UDP thread. The server UDP thread detects the final empty byte to signal that the file transfer is done, and as such will start a new loop for other incoming files.

4 Potential Improvements

4.1 UDP Command Display

Displaying UDP output to client

The UDP command displays to the user the filename to the audience client after it has been successfully received whole. This causes an issue where the displayed message appears over the command input line.

Note: However, commands can still be inputted into the client program despite having text of the downloaded file appearing in the terminal.

Example:

Presenter client sending the file:

```
richq@DESKTOP-09SHN88:/mnt/c/Users/rrqui/OneDrive/Desktop/UNI/COMP3331/Assig
nment/client1$ python3 client.py localhost 2000 2002
Username: yoda
Password: l
Invalid Password. Please try again
Password: k
Welcome to Toom!

================================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> UDP hans example2.mp4
================================================================================
sending example2.mp4
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>>
```

Audience client receiving the file:

```
richq@DESKTOP-09SHN88:/mnt/c/Users/rrqui/OneDrive/Desktop/UNI/COMP3331/Assig
nment/client2$ python3 client.py localhost 2000 2001
Username: hans
Password: k
Welcome to Toom!

================================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>>
Downloaded example2.mp4 from yoda as yoda_example2.mp4

```

Potential Solution

Stop client execution by checking whether the client is alive, and only resuming getting input commands from the user once the file has been fully sent and received. However, this would violate the condition that the commands can still be input by the user while the file is downloading, as specified in the specification.

4.2 UDP Command Code Structure

In code

Since UDP command is a peer-to-peer interaction, the client must also handle the UDP command as opposed to previous commands where the interaction simply involves sending a request and the client printing the data of the response. However, the UDP command breaks this structure in which the command calls the *handle_command_UDP* function inside the client program and sends its own message format, breaking the message format that was previously used.

Solution

Use the same message format to retrieve the user details of audience user from the server.

5 Sample Interaction

5.1 Login

Invalid username supplied

```
nment/client1$ python3 client.py localhost 2000 2002
Username: harry
Unknown username. Please try again
Username:
```

Blocked user after exceeding the permitted number of failed attempts

Server input

```
PS C:\Users\rrqui\OneDrive\Desktop\UNI\COMP3331\Assignment> python .\server.py 2000 2
Server is now running
```

We have specified the number of allowed failed attempts as 2.

Client Output

```
nment/client1$ python3 client.py localhost 2000 2002
Username: hans
Password: p
Invalid Password. Please try again
Password: p
Invalid Password. You have been blocked
```

Hans cannot login after 10 seconds even on another client.

Successful Login

```
richq@DESKTOP-09SHN88:/mnt/c/Users/rrqui/OneDrive/Desktop/UNI/COMP3331/Assig
nment/client1$ python3 client.py localhost 2000 2002
Username: hans
Password: k
Welcome to Toom!

------------------------------------------------------------------------
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>>
```

5.2 RDM Command

Creating separate rooms for the active users

Active users are: yoda, hans vader

Vader creating a room

```
================================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> SRB hans vader

-- [SRB] Response ------------------------------------------------

Status: True
For: vader

Separate chat room (1) created with: hans, vader

--------------------------------------------------------------------
```

Vader cannot create the exact same room

```
================================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> SRB hans vader

-- [SRB] Response ------------------------------------------------

Status: False
For: vader

A separate room (ID: 1) already created for these users

--------------------------------------------------------------------
```

Vader creating another room with him and yoda

```
================================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> SRB vader yoda

-- [SRB] Response ------------------------------------------------

Status: True
For: vader

Separate chat room (2) created with: vader, yoda

--------------------------------------------------------------------
```

Users creating messages in the rooms if they are a part of it

Hans creating a message in room 1

```
================================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> SRM 1 how are we today

-- [SRM] Response ------------------------------------------------

Status: True
For: hans

Chat room message #1 created in room 1 at 04 Aug 2022 22:01:44

--------------------------------------------------------------------
```

Vader displaying all the messages in the rooms that he is a part of

```
======================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> RDM s 1 Jun 2001 0:0:0

-- [RDM] Response ------------------------------------------------

Status: True
For: vader


Room ID 1:
   > ID: #1; 04 Aug 2022 22:01:44; hans; how are we today
   > ID: #2; 04 Aug 2022 22:02:42; vader; hello hans
   > ID: #3; 04 Aug 2022 22:02:51; vader; how are you today

Room ID 2:
   > ID: #1; 04 Aug 2022 22:02:20; yoda; vader my friend
   > ID: #2; 04 Aug 2022 22:02:29; yoda; you there?
   > ID: #3; 04 Aug 2022 22:02:38; vader; hello yoda


----------------------------------------------------------------------
```

Only displaying messages after 4 Aug 2022 22:02:30

```
======================================================================
Enter one of the following commands (BCM, ATU, SRB, SRM, RDM, UDP, OUT):
>> RDM s 4 Aug 2022 22:02:30

-- [RDM] Response ------------------------------------------------

Status: True
For: vader


Room ID 1:
   > ID: #2; 04 Aug 2022 22:02:42; vader; hello hans
   > ID: #3; 04 Aug 2022 22:02:51; vader; how are you today

Room ID 2:
   > ID: #3; 04 Aug 2022 22:02:38; vader; hello yoda


----------------------------------------------------------------------
```