COMP28112 Distributed Systems Exercise 2

MORE ROBUST MESSAGING SYSTEM FOR HEALTHCARE PROFESSIONALS

March 15, 2024

Joseph Rhodes

University of Manchester

Description of the Code and Protocol

Task 1: Your own basic server

Find the implementation in the myserver.py file.

Task 2: Counting clients

Find the implementation in the myserver.py file.
To verify that it works, look in the terminal where the server was started. It will show we a client connect/disconnects and the total number of clients in the messaging system.
onConnect will add/increment the number of connected clients and onDisconnect will remove/decrement the number of connected clients.

Task 3: Accepting and parsing commands

Find the implementation in the myserver.py file.
When utilizing the messaging system, you will find that using the commands and the responses satisfies the requirements for this task.

Task 4: Designing the protocol

Here are the available commands that clients are able to use:

1. register <screen name>: Register a screen name.
2. message <message>: Send a message to all registered clients.
3. dm <screen name> <message>: Send a direct message to a specific client.
4. list: List all registered clients.
5. help: Display available commands and their usage.

If the client does not use one of the available commands the client will receive an error message saying 'Invalid command. Type 'help' for a list of available commands.'

```
if Command then
    if Register then
        Will register the client if the screen name is not already registered or if the client is not already registered. One limitation of the screen name that is it can only be one word.
    else
        Will respond with an error message if the register command is not used correctly. i.e. 'register' with no parameter, 'register Joe Rhodes' can only be one word, 'register Steve' if Steve is already registered, or 'register Bob' if that client is already registered under another screen name.
    end if
    if Message then
        Will send a message to every client connected to the server. A client is able to receive messages when a client is not registered but cannot send messages.
    else
        Will respond with an error message if the message command is not used correctly i.e. 'message' with no parameter or 'message Joe Rhodes' if the client is not registered yet
    end if
    if DM then
        Sends a direct message to the specified client. Similar rules to the message command where the client has to be registered to send a DM. In this program I have allowed registered clients to DM themselves.
    else
```

```
        Will respond with an error message if the DM command is not used correctly i.e. 'DM screen name' with no parameter, 'DM screen name' and the screen name specified is not registered or the registered client put in the wrong screen name.
    end if
    if List then
        Will return all of the registered clients on the server. Will return an empty list if the command is called with no registered clients.
    else
        If the List command is called with a parameter i.e. 'List hello' there will be an error message telling you to use the List command without any parameters.
    end if
    if Help then
        Will return all of the available commands in the Messaging system.
    else
        If the Help command is called with a parameter i.e. 'Help hello' there will be an error message telling you to use the help command without any parameters.
    end if
end if
```

Task 5: Implement your protocol in the server

Find the implementation of the protocol from Task 4 in the myserver.py file.

Task 6: Writing a Client Application

Find the implementation in the myclient.py file.

I created a GUI for the client, making the messaging system more user-friendly and visually appealing. To use the GUI, run the client command in the terminal, and a pop-up window will appear. It is preferable to run two clients so they can communicate with each other. The client will be able to use all of the available commands in the message window and communicate with the other clients.

Running and Testing Description

Running the Code

Running the Server

python3 ./myserver.py localhost 8090

Running the Client

python3 ./myclient.py localhost 8090

Running the client will open up a pop-up window allowing the client to register, message, and all of the following commands implemented in this program.

Testing the Program

Testing the Register Command

Case 1: Just using the command 'register' will result in an error message saying 'To register, please input a screen name.'

Case 2: Trying to have non-single word screen name i.e. 'register Joe Rhodes' will result in an error message saying 'Invalid command format. Please provide a single-word screen name.'

Case 3: 'register Joe' when Joe is not already a registered client will result in a successful registration and a message saying 'Successfully registered screen name: Joe'

Case 4: Trying to register a screen name that is already registered will result in a error message saying 'Screen name 'Joe' is already registered.'

Testing the List Command

Case 1: 'List hello' results in an error message saying 'Invalid command. Please use 'list' without any parameters.'

Case 2: When there are no Registered clients: 'List' will result in 'No registered Clients.'

Case 3: When the register clients are Joe and Steve: 'list' will result in 'Registered Clients: Joe, Steve'. You can do this is as few as 1 registered client or as many as you want to test.

Case 4: Lets say the registered clients are Joe and Steve but Steve disconnects: 'list' will result in 'Registered Clients: Joe'

Testing the Help Command

Case 1: 'Help hello' results in an error message saying 'Invalid command. Please use 'help' without any parameters.'

Case 2: 'Help' results in the display of all of the available commands.

Testing the Message Command

Case 1: With an unregistered client 'message hello' will result in 'You need to register a screen name before sending messages'

Case 2: With one registered client 'message hello' will display as <screen name>: hello on the screen of that one registered client.

Case 3: With more than one registered client 'message hello' will display as <screen name>: hello on the screen of all of the registered clients.

Case 4: Typing 'message' with no parameters will result in an invalid command.

Testing the DM Command

Case 1: Trying to send a direct message without being registered will result in a message saying 'you need to register a screen name before sending a direct message'

Case 2: One registered user sending a direct message to themselves 'DM Joe hello there' will result in them receiving a message saying 'DM from Joe: hello there' along with the direct message they sent 'DM to Joe: hello there.'

Case 3: For Cases 3-4 there are 2 registered clients Joe and Steve. Lets say Joe uses the command 'DM Rob hello' since Rob is not registered Joe will receive an error message saying 'No client registered with the name 'Rob''

Case 4: Joe sends a direct message to Steve 'DM Steve Hello Steve', Steve will receive the message 'DM from Joe: Hello Steve' and on Joe's screen it will say 'DM to Steve: Hello Steve' so they can see what message they had sent.

case 5: Lets say that there are 3 registered clients Joe, Steve, and Rob. If Joe and Steve are direct messaging each each Rob will have no clue and will not receive any of the messages between Steve and Joe.