COL 672: Computer Networks Assignment-2

Sarang Drugkar - 2021JCS2244

Chat Application:

How to execute the code:

1. There are 2 python files present inside the folders namely server.py and client.py.
2. First run the server.py file from command prompt using the following command.

   python server.py

3. After running the server.py file open the required number of command prompts for as many numbers of users we want to chat in our system.
4. Use the following command in every command prompt to run client.py file.

   python client.py

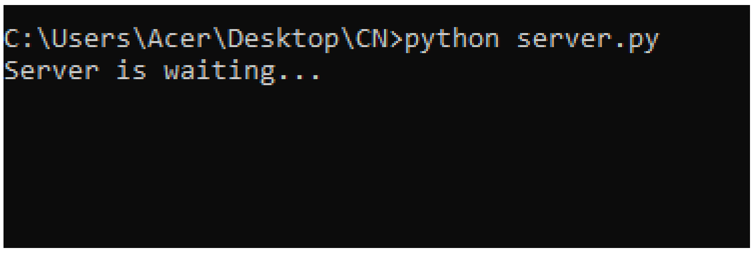
5. After running the client.py file, it will ask for the username and server_IP as an input. Type the **username** as you want and **server_IP** as the local host as **127.0.0.1**.
6. Finally, the user will get registered to send and receive the messages and can exchange the messages using the format specified in Assignment PDF.

Implementation:

1. We start implementation of our chat application from server side. In server.py file we first created the port number which we are going to use throughout our message exchange. We have selected the port no. as 5500 in both server and client file. We are maintaining the list of all clients in the map. We are then creating the sockets in both server and client side.
2. In client side, we are taking the username and server_IP as an input. Now to register the client with the server we are creating the function ***'send_reg_message'*** which takes the username as argument. It will send the registration request to server.
3. On server side we are receiving this request in ***'Implementation'*** function and then passing it to ***'Register'*** function where we are checking the received request and comparing with the correct format as *'REGISTER TOSEND [username]'.* If it is correct, then we are sending the response message to client *'REGISTERED TOSEND [username]'*. Otherwise, respective error message is sent back to client.
4. After the registration of clients, if we want to send the message from one client to other then we are using the ***'snd_sock_generated'*** function where we start threading and calling ***'msg_send'*** function in which we are taking the input message from username in the format *'@[username]: message'* and splitting it. We are then sending this message to server again where it checks for the correct format as *'SEND [recipient username] Content-length:[length] [message of length given in the header]'* in ***'fwd_svr'*** function. If the correct format is received then it forwards the message to the intended client, otherwise it sends the respective error message back to client. Client received this message from server using the function ***'msg_receive'***.
5. When the other client receives message then it also sends the acknowledgment to server where server forwards this acknowledgement to original client and it receives this acknowledgment using ***'svr_acknd'*** function and finally prints 'message successfully sent to [username]'.
6. For broadcast messages only difference is we are using ***'brd_svr'*** function in server.

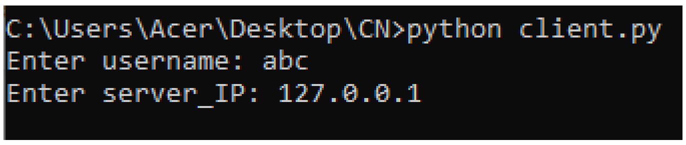
Implementation Results:

1. Run server.py file

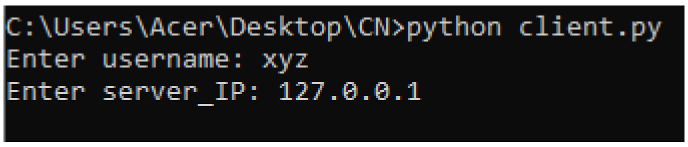


```
C:\Users\Acer\Desktop\CN>python server.py
Server is waiting...
```

2. Run client.py in two command prompts as we want to chat between 2 users.

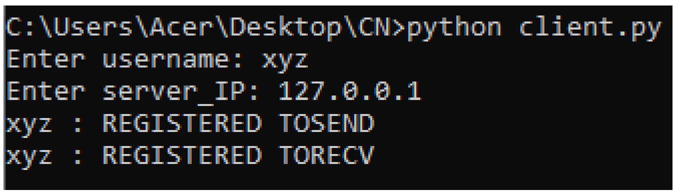


```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: abc
Enter server_IP: 127.0.0.1
```

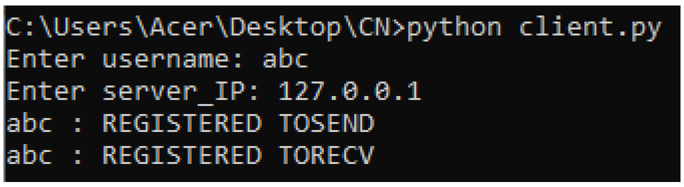


```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: xyz
Enter server_IP: 127.0.0.1
```

3. After entering the username and server IP, press enter

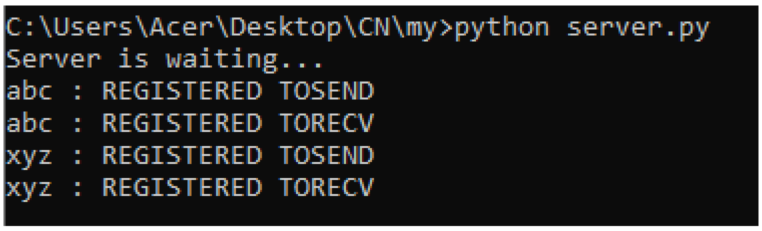


```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: xyz
Enter server_IP: 127.0.0.1
xyz : REGISTERED TOSEND
xyz : REGISTERED TORECV
```



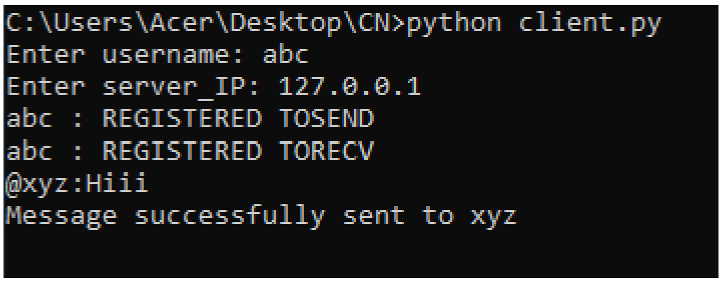
```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: abc
Enter server_IP: 127.0.0.1
abc : REGISTERED TOSEND
abc : REGISTERED TORECV
```

Both the Clients are registered for send and receive messages.

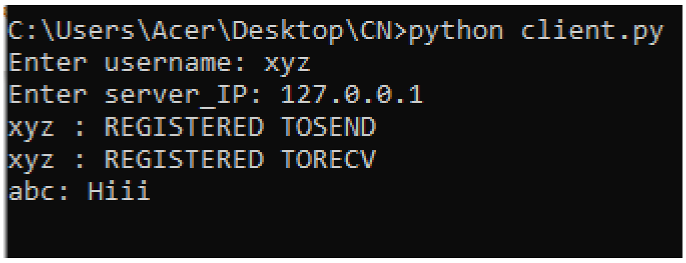


```
C:\Users\Acer\Desktop\CN\my>python server.py
Server is waiting...
abc : REGISTERED TOSEND
abc : REGISTERED TORECV
xyz : REGISTERED TOSEND
xyz : REGISTERED TORECV
```

4. Sending message from abc to xyz,

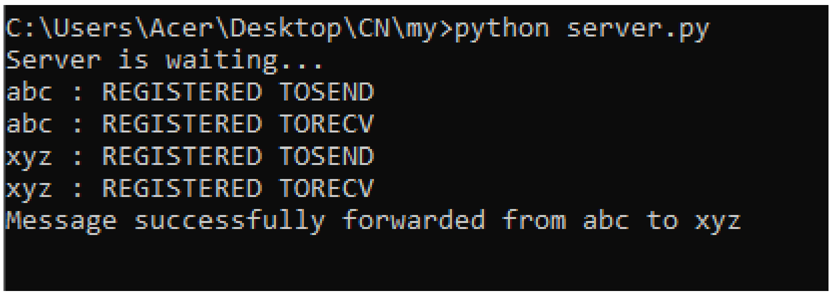


```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: abc
Enter server_IP: 127.0.0.1
abc : REGISTERED TOSEND
abc : REGISTERED TORECV
@xyz:Hiii
Message successfully sent to xyz
```



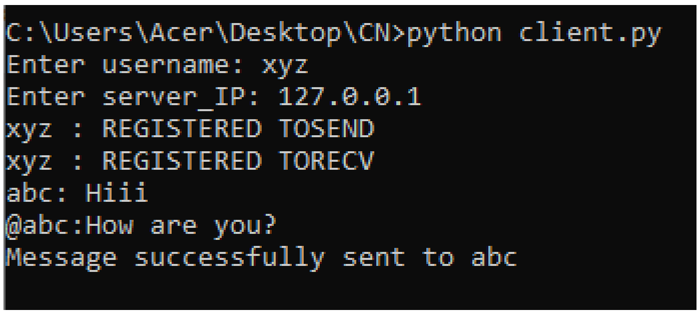
```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: xyz
Enter server_IP: 127.0.0.1
xyz : REGISTERED TOSEND
xyz : REGISTERED TORECV
abc: Hiii
```

For server,

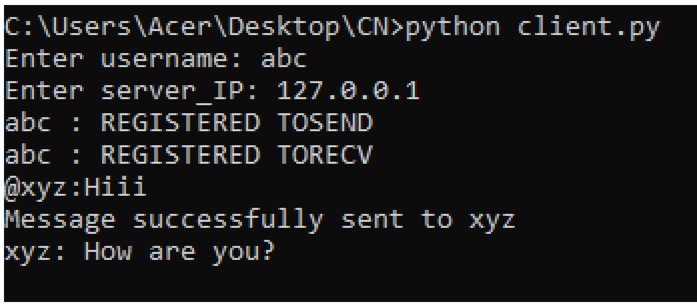


```
C:\Users\Acer\Desktop\CN\my>python server.py
Server is waiting...
abc : REGISTERED TOSEND
abc : REGISTERED TORECV
xyz : REGISTERED TOSEND
xyz : REGISTERED TORECV
Message successfully forwarded from abc to xyz
```

5. Reply from xyz to abc,

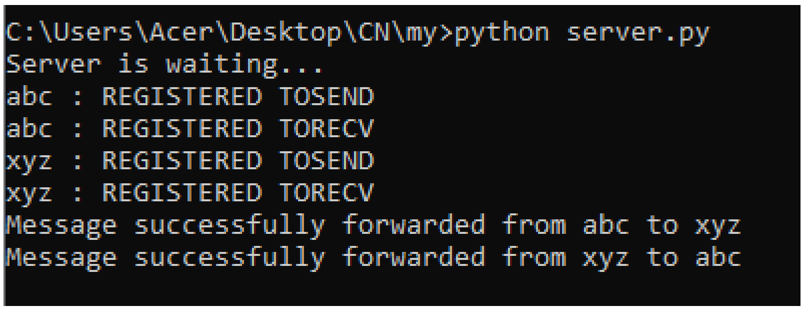


```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: xyz
Enter server_IP: 127.0.0.1
xyz : REGISTERED TOSEND
xyz : REGISTERED TORECV
abc: Hiii
@abc:How are you?
Message successfully sent to abc
```



```
C:\Users\Acer\Desktop\CN>python client.py
Enter username: abc
Enter server_IP: 127.0.0.1
abc : REGISTERED TOSEND
abc : REGISTERED TORECV
@xyz:Hiii
Message successfully sent to xyz
xyz: How are you?
```

For server,



```
C:\Users\Acer\Desktop\CN\my>python server.py
Server is waiting...
abc : REGISTERED TOSEND
abc : REGISTERED TORECV
xyz : REGISTERED TOSEND
xyz : REGISTERED TORECV
Message successfully forwarded from abc to xyz
Message successfully forwarded from xyz to abc
```