Homework-4: CS425A Computer Networks

Kuldeep Singh Chouhan

200530

The zip file contains two python files along with this README. The two files make a socket programming for UDP client server chatting application. Here, the server and client can chat until one of them says ‘bye’. To run this application, open two different terminals and run

```
python server.py
```

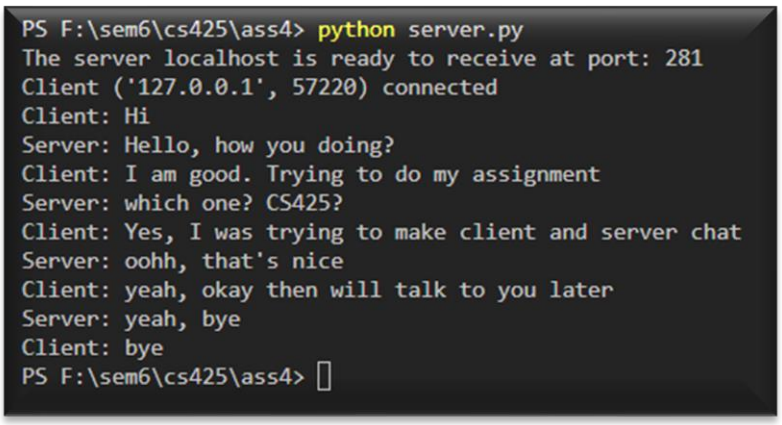
in one of them. This terminal will be our server and will wait for the client to connect with it. Then in other terminal, run

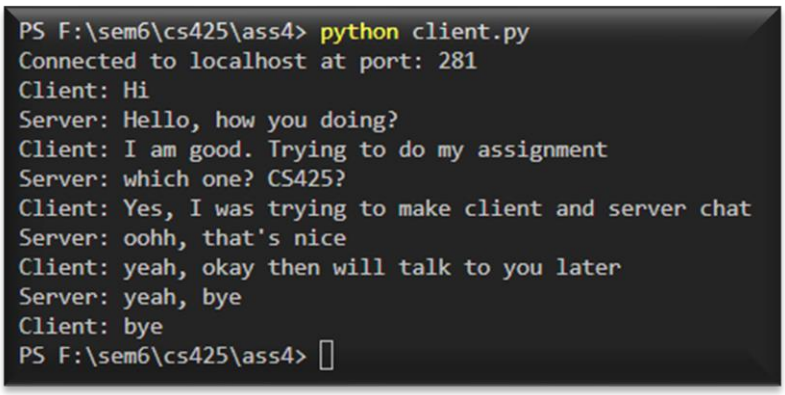
```
python client.py
```

This terminal will be our client. The first message will be sent by the client.

Note: one cannot close the server/client through a keyboard interrupt, when it’s waiting for a response message.

One example:





Client

```
from socket import *

# server details to be connected to
serverName = 'localhost'
serverPort = 281

# connecting to server
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
client_socket.connect((serverName, serverPort))
print("Connected to", serverName, "at port:", serverPort)

# send and receive messages to/from the server
while True:
    # send a message to the server
    message = input('Client: ')
    client_socket.send(message.encode())

    # if the message is 'bye', close the connection
    if message == 'bye':
        client_socket.close()
        break

    # receive a response message from the server
    response_message = client_socket.recv(1024).decode()

    # print the response message
    print('Server: {}'.format(response_message))

    # if the response message is 'bye', close the connection
    if response_message == 'bye':
        client_socket.close()
        break

# close the client socket
client_socket.close()
```

Server

```
from socket import *

# server details
serverName = 'localhost'
serverPort = 281

# server ready to receive
server_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
server_socket.bind((serverName, serverPort))
server_socket.listen()
print ("The server", serverName,  "is ready to receive at
port:",serverPort)

# wait for a client to connect
client_socket, client_address = server_socket.accept()
print('Client {} connected'.format(client_address))

# receive and send messages to/from the client
while True:
    # receive message from the client
    message = client_socket.recv(1024).decode()

    # print the message
    print('Client: {}'.format( message))

    # if the message is 'bye', close the connection
    if message == 'bye':
        client_socket.close()
        break

    # send a response message to the client
    response_message = input('Server: ')
    client_socket.send(response_message.encode())

    # if the message is 'bye', close the connection
    if response_message == 'bye':
        client_socket.close()
        break

# close the server socket
server_socket.close()
```