Basant Solanky

12040430

Assignment- 3

Socket Programming

Part 1: Implement SSH using socket programming. [20 Points]

- **Run the server script first:** python Server.py
  Enter the port number (1024-65536)
- **Run the client script**: python Client.py
  Port number for this will be same as above entered.

Client.py

This file contains the client side code executed on the server port which take input from the user as username and the corresponding password. If the password matches in the server.py file then connection will be established.

```python
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
port=int(input("Enter the port number:- "))
client_socket.connect(('127.0.0.1', port))


username = input("Enter Username: ")
password = input("Enter password: ")



client_socket.send(username.encode('utf-8'))

auth_response = client_socket.recv(3).decode('utf-8')


if auth_response == 'yes':
    client_socket.send(password.encode('utf-8'))
    auth_response1 = client_socket.recv(3).decode('utf-8')

    if auth_response1 == 'yes':
        # Display system information
        welcome_message = client_socket.recv(1024).decode('utf-8')
        print(welcome_message)

        while True:
            command = input(f"{username}@remoteserver$ ")
            client_socket.send(command.encode('utf-8'))

            if command == 'exit':
                break

            # Receive and display the command output
            output = client_socket.recv(1024).decode('utf-8')
            print(output)

    else:
        print("Authentication failed")

else:
    print("Authentication failed")

client_socket.close()
```

Server.py

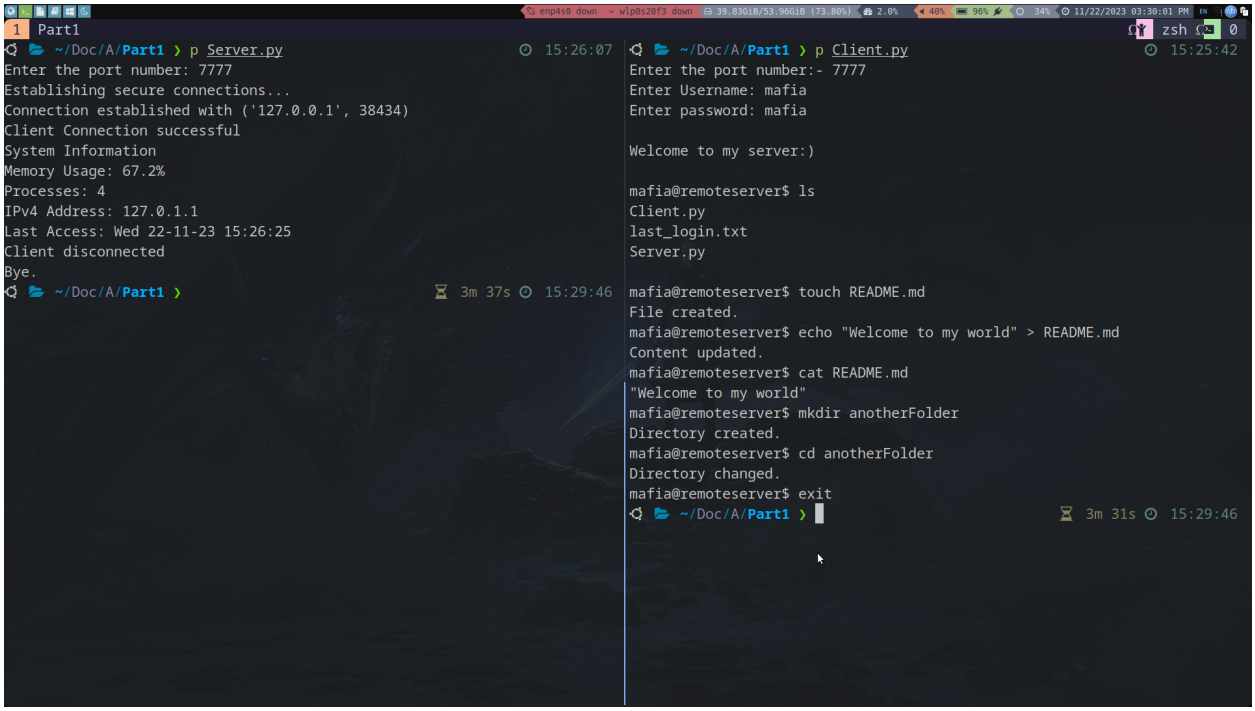
This is server.py file which contains multi-users username and their password and the last login session which is stored in a text file named ‘last_login.txt’. Once the credentials are matched it sends a welcome message to the client, and on server its shows system informations.

```python
import socket
import os
import datetime
import psutil
credentials = {
    "mafia": "mafia",
    "basant": "localhost",
    "arch": "penguin"
    "Admin": "root"
}

last_login_file = "last_login.txt"

def authenticate_username(input_username):
    return input_username in credentials

def authenticate_password(username, input_passwd):
    return credentials.get(username) == input_passwd

def execute_command(command):
    try:
        result = os.popen(command).read()
        return result
    except Exception as e:
        return f"Command Execution Error: {str(e)}"

#
def fetch_system_info():
    memory_usage = psutil.virtual_memory().percent
    processes = psutil.cpu_count(logical=False)
    ipv4_address = socket.gethostbyname(socket.gethostname())
    last_access_time = datetime.datetime.now().strftime("%a %d-%m-%y %H:%M:%S")
    print(f"Client Connection successful ")
    print("System Information")
    print(f"Memory Usage: {memory_usage}%")
    print(f"Processes: {processes}")
    print(f"IPv4 Address: {ipv4_address}")
    print(f"Last Access: {last_access_time}")

    return f"\nWelcome to my server:)\n"

def update_last_login():
    with open(last_login_file, 'w') as file:
        file.write(datetime.datetime.now().strftime("%a %d-%m-%y %H:%M:%S"))
```

```python
def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    port = int(input("Enter the port number: "))
    server_socket.bind(('127.0.0.1', port))
    server_socket.listen(1)

    print("Establishing secure connections...")
    client_socket, client_address = server_socket.accept()
    print(f"Connection established with {client_address}")

    # Authentication
    input_username = client_socket.recv(1024).decode('utf-8')
    if authenticate_username(input_username):
        client_socket.send(b'yes')

        input_passwd = client_socket.recv(1024).decode('utf-8')
        if authenticate_password(input_username,input_passwd):
            client_socket.send(b'yes')

            update_last_login()

            welcome_msg = fetch_system_info()
            client_socket.send(welcome_msg.encode('utf-8'))
```

```python
        while True:
            command = client_socket.recv(1024).decode('utf-8')
            if command == 'exit':
                print("Client disconnected\nBye.")
                break
                # continue

            if command.startswith('cd '):
                os.chdir(command[3:])
                client_socket.send(b'Directory changed.')
            elif command.startswith('mkdir '):
                os.mkdir(command[6:])
                client_socket.send(b'Directory created.')
            elif command.startswith('touch '):
                open(command[6:], 'a').close()
                client_socket.send(b'File created.')
            elif command.startswith('cat '):
                try:
                    with open(command[4:], 'r') as file:
                        content = file.read()
                        client_socket.send(content.encode('utf-8'))
                except FileNotFoundError:
                    client_socket.send(b'File not found.')
            elif command.startswith('echo '):
                try:
                    content = command[5:]

                    if ' > ' in content:
                        parts = content.split(' > ', 1)
                        file_name = parts[-1].strip()
                        with open(file_name, 'w') as file:
                            file.write(parts[0])
                        client_socket.send(b'Content updated.')
                    else:
                        client_socket.send(content.encode('utf-8'))
                except Exception as e:
                    client_socket.send(f'Error: {str(e)}'.encode('utf-8'))
            elif command.startswith('ls') or command.startswith('echo'):
                output = execute_command(command)
                client_socket.send(output.encode('utf-8'))
            else:
                client_socket.send(b'Still learning new commands')

        else:
            client_socket.send(b'no')
            print("Client disconnected")
    else:
        client_socket.send(b'no')
        print("Authentication Failed:(")

    client_socket.close()
    server_socket.close()

if __name__ == "__main__":
    main()
```

This snippet show various command we can use in the ssh server such as:- cd, mkdir, echo, ls, touch, cat.



Workflow with sutaible commands

Part 2: Build a DNS resolver application using socket programming. [15 Points]

Client.py

```python
import socket

hostname = socket.gethostname()
IPA = '127.0.0.1'
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port=int(input("Enter port number: "))
addr = (IPA, port)

c = "Y"
transaction_id = 1

while c.upper() == "Y":
    req_domain = input("Enter domain name: ")
    data_to_send = req_domain
    send = s.sendto(data_to_send.encode(), addr)
    # data_another =transaction_id
    # send_another=s.sendto(data_another.encode(),addr)
    data, address = s.recvfrom(1024)
    reply_ip = data.decode().strip()
    print(f"{transaction_id} {req_domain} {reply_ip}")
    transaction_id += 1
    c = input("Continue? (y/n) ")

s.close()
```

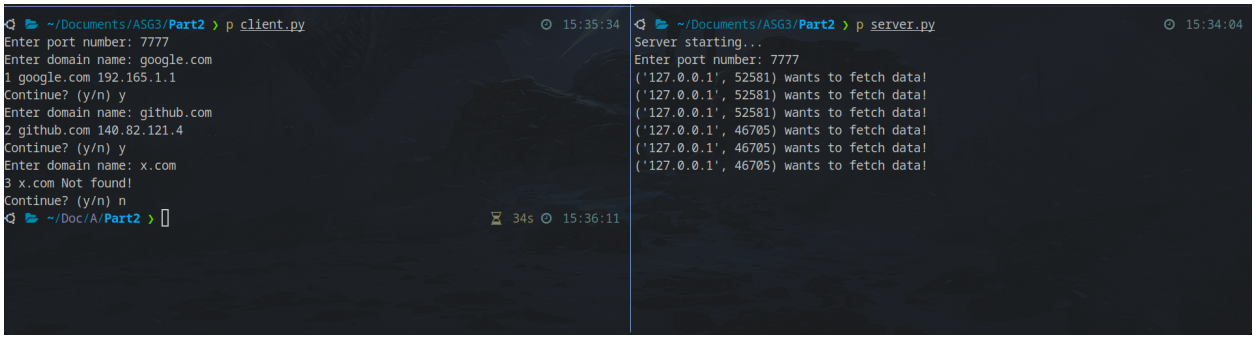
Server.py

```python
import socket

def read_dns_table(file_path):
    dns_table = {}
    with open(file_path, 'r') as file:
        for line in file:
            parts = line.strip().split()
            transaction_id, domain, ip = int(parts[0]), parts[1], parts[2]
            dns_table[domain]=ip
    return dns_table

file_path = 'query'
dns_table = read_dns_table(file_path)

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print("Server starting...")

port=int(input("Enter port number: "))
s.bind(("127.0.0.1", port))

while True:
    data, address = s.recvfrom(1024)
    print(f"{address} wants to fetch data!")
    data = data.decode()
    ip = dns_table.get(data, "Not found!").encode()
    s.sendto(ip, address)
```

DNS queries are stored in a file and server.py fetches the IPAddress whenever client asks for.

```
1 google.com 192.165.1.1
2 example.com 203.0.113.1
3 facebook.com 192.168.0.1
4 amazon.com 172.217.10.14
5 twitter.com 104.244.42.1
6 reddit.com 151.101.1.140
7 linkedin.com 108.174.10.10
8 github.com 140.82.121.4
```

User asking for IPAddress for the domains which exists in the database.



Workflow

Part 3: Making Echo Client/Server “Protocol Independent” [15 Points].

- **Run the server script first:** python Server.py <localhost/ip6-localhost> <port>

- **Run the client script**: python Client.py <localhost/ip6-localhost> <port> <packet_count> <interval> <packet_size>

Server.py

```python
import socket
import sys

if (len(sys.argv) != 3):
    print("To use this file:\n python3 part3_server.py <localhost/ip6-localhost> <port>")
    exit()

# Host received from the user
host = sys.argv[1]
# Port number received from the user
port = sys.argv[2]
buffsize = 1024

# Get the information about the host using getaddrinfo
addr_info = socket.getaddrinfo(host,port)
# Created a UDP socket using the family info received from getaddrinfo
UDPServerSocket = socket.socket(family=addr_info[0][0], type=socket.SOCK_DGRAM)
# Binding the server to the address received from the getaddrinfo
UDPServerSocket.bind(addr_info[0][4])

if (host == "ip6-localhost"):
    print("UDP ipv6 server is running!")
else:
    print("UPD ipv4 server is running")

while (True):
    # Message and Address(client's) recevied from the client
    message,address = UDPServerSocket.recvfrom(buffsize)
    # Send the message back to the client
    UDPServerSocket.sendto(message,address)
```

Client.py

Server.py • Client.py ×

```python
import socket
import sys
import datetime
import time

if (len(sys.argv) != 6):
    print("To use this file, run the file in the following manner:")
    print("python3 part3_client.py <localhost/ip6-localhost> <port> <packet_count> <interval> <packet_size>")
    exit()
# Hostname received from the user (localhost or ip6-localhost)
host = sys.argv[1]
# Packet count from the user
packet_count = int(sys.argv[3])
# Interval between the packets transmission
interval = int(sys.argv[4])
# Packet Size from the user
packet_size = int(sys.argv[5])

# port number of the client (hardcoded)
port = 8000
timeout = 1

# If packet size is greater than 1024 then
if (packet_size > 1024):
    print("Packet Size should be lower than 1024")
    exit()

# UDP client Socket Initialised
UDPClientSocket = socket.socket(family=socket.getaddrinfo(host,port)[0][0], type=socket.SOCK_DGRAM)

loss_count = 0
received_count = 0
total_packets = packet_count
total_time = 0
```

NORMAL Client.py ⊗ 0 ⚠ 0 Ln 35, Col 0 UTF-8 {} python Part3

Server.py • Client.py ×

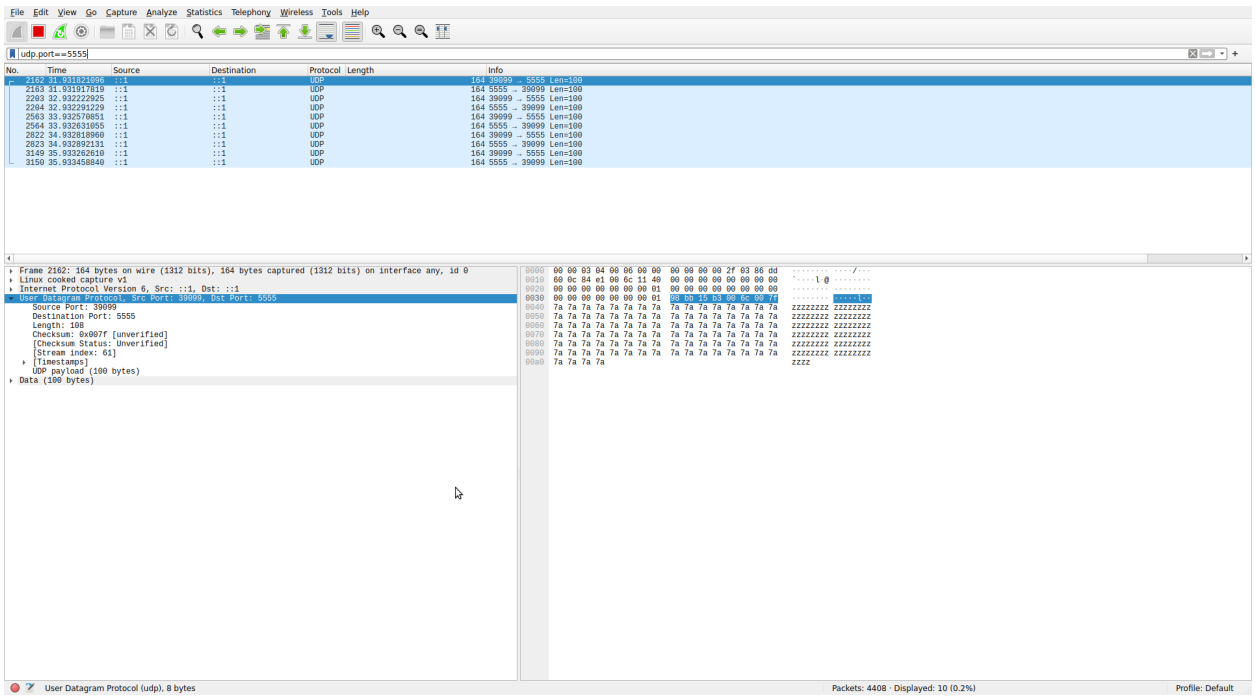
```python
while(packet_count):

    # Message To send
    message = "z"*packet_size
    packet_received = False # Flag to check receiving of the file
    UDPClientSocket.sendto(bytes(message,'ascii'),(host,int(sys.argv[2])))
    start_time = datetime.datetime.now() # Start time flag
    while ((datetime.datetime.now() - start_time).total_seconds() < timeout):
        if (UDPClientSocket.recvfrom(packet_size)):
            end_time = datetime.datetime.now() # End time flag
            packet_received = True
            print(f"RTT => {(end_time - start_time).total_seconds()}")
            print("==========================")
            total_time += (end_time - start_time).total_seconds()
            received_count += 1
            break
    if (packet_received == False):
        print("Packet Loss")
        print("======================")
        loss_count += 1
    packet_count = packet_count - 1
    time.sleep(interval)
print("======== Overall Details ========")
print(f"Packet Received: {received_count}")
print(f"Packet Sent: {total_packets}")
print(f"Loss: {(loss_count/total_packets)*100}%")
print(f"Average RTT: {total_time/total_packets}")
```

NORMAL Client.py ⊗ 0 ⚠ 0 Ln 37, Col 6 UTF-8 {} python Part3

Ipv6-localhost port:- 5555

```
1 part3                                                                    part3.3 2

~/Documents/ASG3/part3  main !302 ?4 ❯ python Server.py ip6-localhost 5555   19:45:27
UDP ipv6 server is running!
```

```
~/Documents/ASG3/part3  main !302 ?4 ❯ python Client.py ip6-localhost 5555 5 1 100   19:44:56 [0/3]
RTT => 5.6e-05
==========================
RTT => 3.9e-05
==========================
RTT => 3.4e-05
==========================
RTT => 3.7e-05
==========================
RTT => 3.7e-05
==========================
======== Overall Details ========
Packet Received: 5
Packet Sent: 5
Loss: 0.0%
Average RTT: 4.06e-05
~/Documents/ASG3/part3  main !302 ?4 ❯                                  5s 19:45:58
```

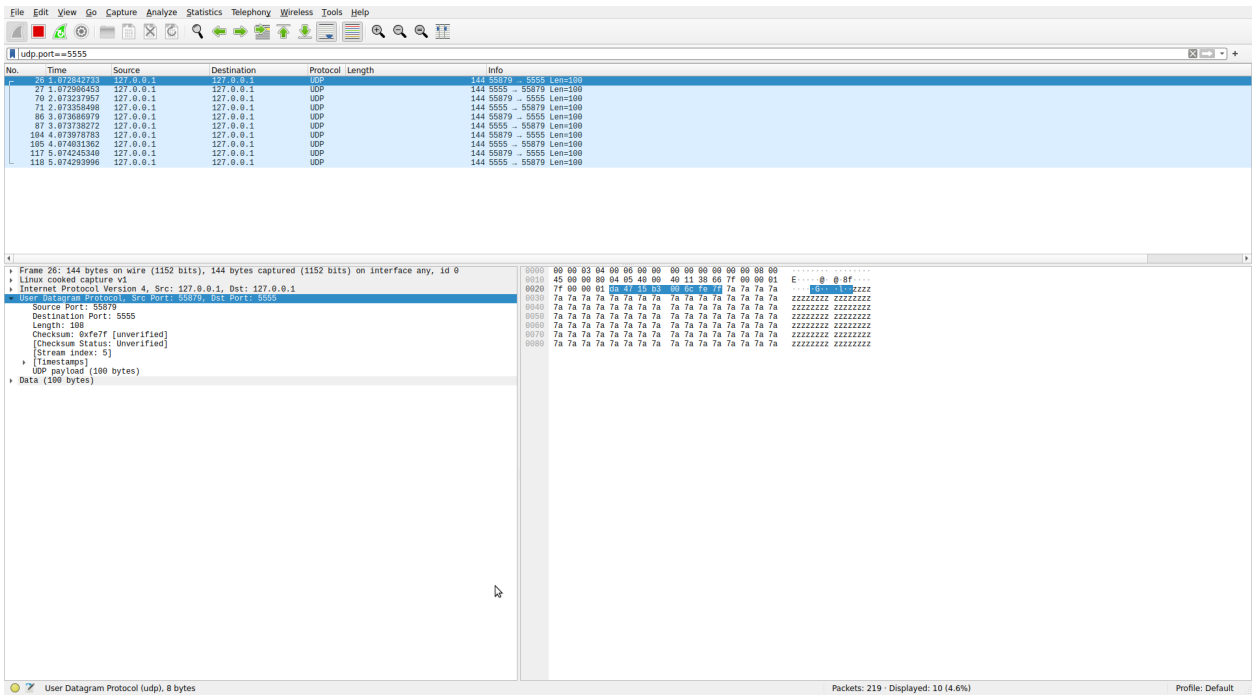


Wireshark Output

localhost **port:- 5555**

```
1 part3                                                                    part3.3 2
~/Documents/ASG3/part3  main !302 ?4 ❯ python Server.py localhost 5555          19:50:45
UPD ipv4 server is running
```

```
~/Documents/ASG3/part3  main !302 ?4 ❯ python Client.py localhost 5555 5 1 100  19:50:43
RTT => 0.000333
==========================
RTT => 7.3e-05
==========================
RTT => 0.000103
==========================
RTT => 0.000119
==========================
RTT => 7.5e-05
==========================
======== Overall Details ========
Packet Received: 5
Packet Sent: 5
Loss: 0.0%
Average RTT: 0.0001406
~/Documents/ASG3/part3  main !302 ?4 ❯                                  5s 19:51:18
```



Wireshark Ouput

The server receives a certain amount of packets from the client and returns each packet to the client via echo. The client measures every packet's round-trip time.

Part 4: Create your own client server application [15 Points]

Add any one feature to Client/Server and demonstrate them. In the report, you must describe the new features with their benefit. Significance of the feature will impact the marks given.

- **Run the server script first:** python Server.py <port>
- **Run the client script**: python Client.py <port>

Client.py

```python
import socket
import select
import sys

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
if len(sys.argv) != 2:
    print ("Correct usage: script, port number")
    exit()
IPA = socket.gethostbyname(socket.gethostname())

Port = int(sys.argv[1])
server.connect((IPA, Port))

while True:

    # maintains a list of possible input streams
    sockets_list = [sys.stdin, server]
    read_sockets,write_socket, error_socket = select.select(sockets_list,[],[])

    for socks in read_sockets:
        if socks == server:
            message = socks.recv(2048)
            print(f"Received Message: {message}")
        else:
            message = sys.stdin.readline()
            server.send(bytes(message,'ascii'))
            sys.stdout.write("<User>")
            sys.stdout.write(message)
            sys.stdout.flush()


```

After running both scripts, the client is connected to the server, and we can connect multiple clients to the server. Every client can see the messages sent from another client, and the server keeps track of the messages with the message's timestamp.

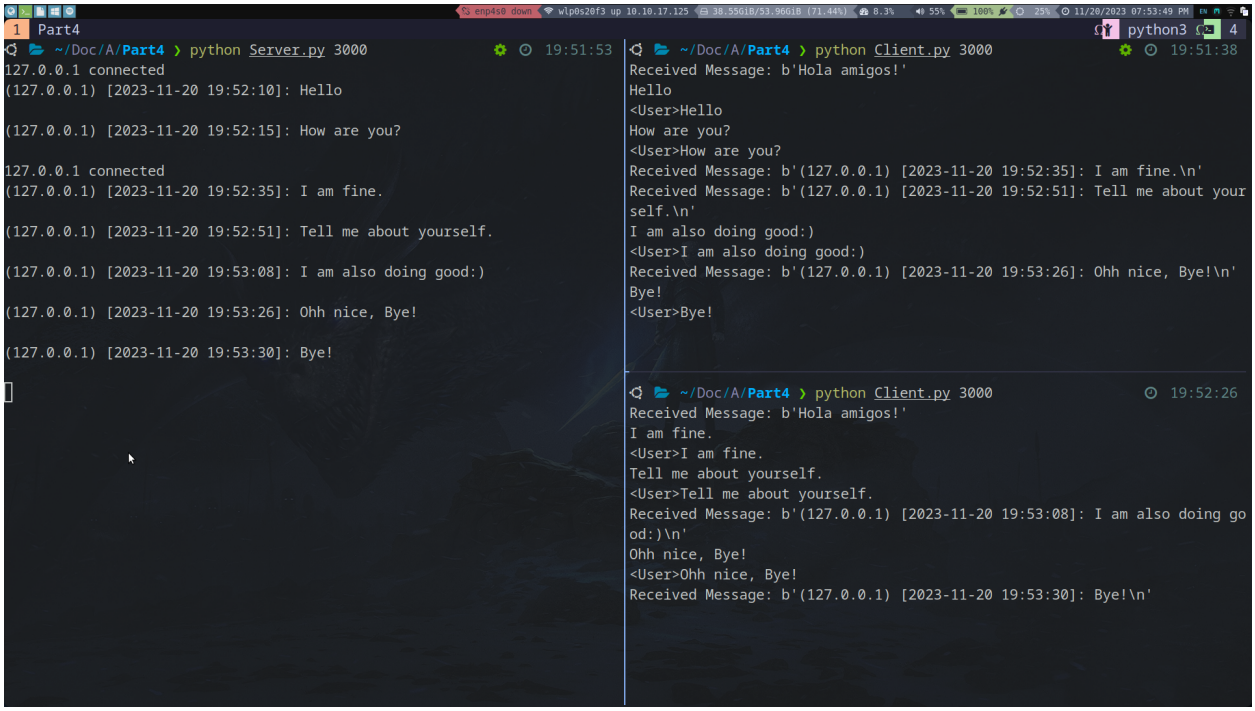
Server.py

```python
import socket
import sys
from _thread import *
import time

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

if len(sys.argv) != 2:
    print("Correct usage: script, port number")
    exit()
IPA = socket.gethostbyname(socket.gethostname())

Port = int(sys.argv[1])
server.bind((IPA, Port))  # We are setting up the ip_address and port number for the server
server.listen(100)  # We can set the number of users to 100
all_clients = []


# New threads will be created for different users for this function
def clientthread(conn, addr):
    conn.send(b"Hola amigos!")
    while True:
        try:
            message = conn.recv(2048)
            if message:
                # Get the current time
                current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

                # Print the message along with the timestamp
                print(f"({addr[0]}) [{current_time}]: {message.decode()}")

                # Calls broadcast function to send message to all
                message_to_send = f"({addr[0]}) [{current_time}]: {message.decode()}"
                broadcast(message_to_send, conn)
            else:
                remove(conn)
        except:
            continue

```

```python
# broadcast the message to all other users
def broadcast(message, connection):
    for clients in all_clients:
        if clients != connection:
            try:
                clients.send(bytes(message, 'ascii'))
            except:
                clients.close()
                # if the link is broken, we remove the client
                remove(clients)


def remove(connection):
    if connection in all_clients:
        all_clients.remove(connection)

while True:
    conn, addr = server.accept()
    all_clients.append(conn)
    # prints the address of the user that just connected
    print(addr[0] + " connected")
    # creates and individual thread for every user
    # that connects
    start_new_thread(clientthread, (conn, addr))
```

Viewing messages from every user connected to the server in one location with this server/client application is possible.

Server view with timestamp



Ref:- https://www.geeksforgeeks.org/simple-chat-room-using-python/

https://youtu.be/Um_oIBPs48w?si=x4n0t2MxcD_BIvbG