COMPUTER NETWORKS LAB

Practical-7

NAME: VEDANT BHUTADA

ROLL: 69

SECTION: A

BATCH: A4

Aim: Socket Programming - Implement chat server using TCP/UDP

Code:

Tcp_server

```python
import socket
import threading

# Function to handle client connections
def handle_client(client_socket, address):
    while True:
        data = client_socket.recv(1024)
        if not data:
            break
        message = data.decode('utf-8')
        print(f"Received from {address}: {message}")
        # Broadcast the message to all clients
        broadcast(message, client_socket)

    client_socket.close()

# Function to broadcast a message to all connected clients
def broadcast(message, sender_socket):
    for client in clients:
        if client != sender_socket:
            try:
                client.send(message.encode('utf-8'))
            except:
                # Remove the client if unable to send a message
                clients.remove(client)

# Create a socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
server_socket.bind(('localhost', 5555))
server_socket.listen(5)

print("TCP Chat Server is listening on port 5555...")

# List to store client sockets
clients = []

while True:
    client_socket, address = server_socket.accept()
    clients.append(client_socket)
    print(f"Connection from {address}")
    client_handler = threading.Thread(target=handle_client, 
args=(client_socket, address))
    client_handler.start()
```

Tcp_client

```
import socket
import threading

# Function to handle receiving messages
def receive_messages(client_socket):
    while True:
        try:
            data = client_socket.recv(1024)
            if not data:
                break
            message = data.decode('utf-8')
            print(f"Received: {message}")
        except Exception as e:
            print(f"Error receiving message: {e}")
            break

# Connect to the server
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 5555))

# Start a thread to receive messages
receive_thread = threading.Thread(target=receive_messages, 
args=(client_socket,))
receive_thread.start()

# Send messages to the server
while True:
    message = input("Enter a message: ")
    client_socket.send(message.encode('utf-8'))
```

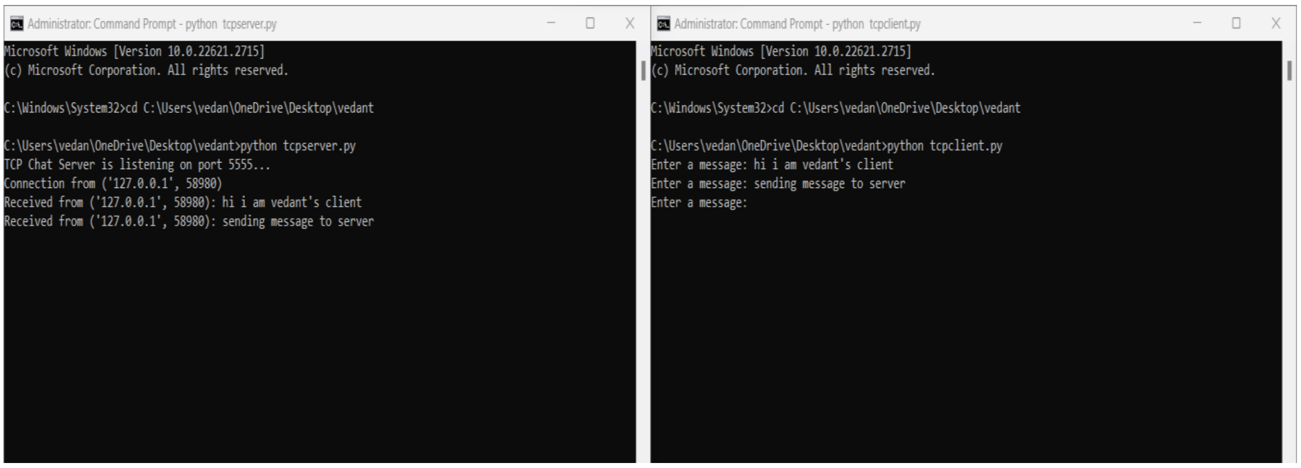
Udp_server

```
import socket
import threading

# Function to handle incoming UDP messages
def handle_udp_messages():
    while True:
        data, address = udp_server_socket.recvfrom(1024)
        message = data.decode('utf-8')
        print(f"Received from {address}: {message}")
        # Broadcast the message to all clients
        broadcast_udp(message, address)

# Function to broadcast a UDP message to all connected clients
def broadcast_udp(message, sender_address):
    for client_address in udp_clients:
        if client_address != sender_address:
            udp_server_socket.sendto(message.encode('utf-8'), client_address)

# Create a UDP socket
udp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp_server_socket.bind(('localhost', 5556))

print("UDP Chat Server is listening on port 5556...")

# List to store client addresses
udp_clients = set()

# Start a thread to handle incoming UDP messages
udp_thread = threading.Thread(target=handle_udp_messages)
udp_thread.start()
```

Udp_client

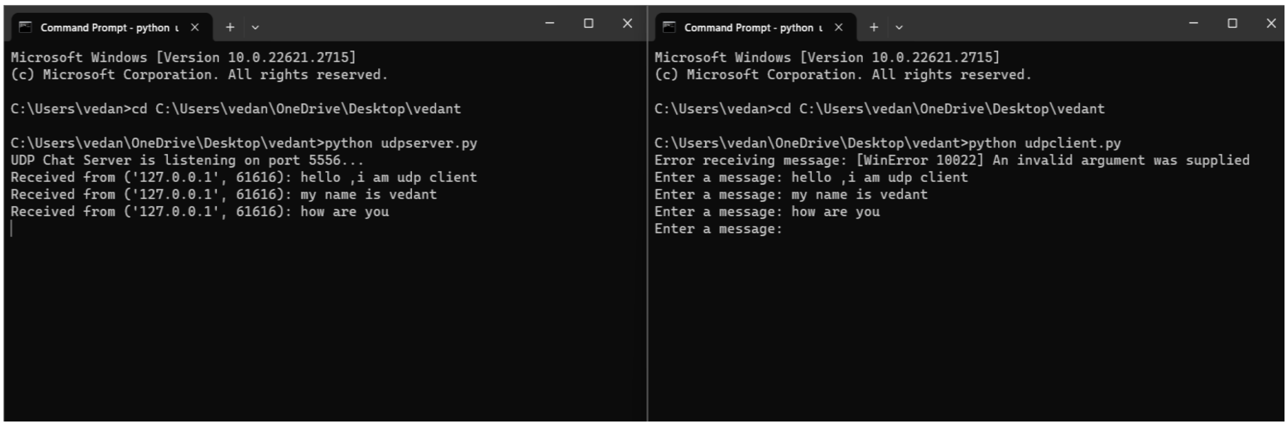
```
import socket
import threading
# Function to handle receiving messages
def receive_messages(client_socket):
    while True:
        try:
            data, address = client_socket.recvfrom(1024)
            message = data.decode('utf-8')
            print(f"Received from {address}: {message}")
        except Exception as e:
            print(f"Error receiving message: {e}")
            break

# Create a UDP socket for the client
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Start a thread to receive messages
receive_thread = threading.Thread(target=receive_messages,
args=(client_socket,))
receive_thread.start()

# Send messages to the server
while True:
    message = input("Enter a message: ")
    client_socket.sendto(message.encode('utf-8'), ('localhost', 5556))
```

Output:


Administrator: Command Prompt - python tcpserver.py
Administrator: Command Prompt - python tcpclient.py

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\vedan\OneDrive\Desktop\vedant

C:\Users\vedan\OneDrive\Desktop\vedant>python tcpserver.py
TCP Chat Server is listening on port 5555...
Connection from ('127.0.0.1', 58980)
Received from ('127.0.0.1', 58980): hi i am vedant's client
Received from ('127.0.0.1', 58980): sending message to server
```

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\vedan\OneDrive\Desktop\vedant

C:\Users\vedan\OneDrive\Desktop\vedant>python tcpclient.py
Enter a message: hi i am vedant's client
Enter a message: sending message to server
Enter a message:
```



```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vedan>cd C:\Users\vedan\OneDrive\Desktop\vedant

C:\Users\vedan\OneDrive\Desktop\vedant>python udpserver.py
UDP Chat Server is listening on port 5556...
Received from ('127.0.0.1', 61616): hello ,i am udp client
Received from ('127.0.0.1', 61616): my name is vedant
Received from ('127.0.0.1', 61616): how are you
```

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vedan>cd C:\Users\vedan\OneDrive\Desktop\vedant

C:\Users\vedan\OneDrive\Desktop\vedant>python udpclient.py
Error receiving message: [WinError 10022] An invalid argument was supplied
Enter a message: hello ,i am udp client
Enter a message: my name is vedant
Enter a message: how are you
Enter a message:
```

Conclusion: In this practical we successfully implemented tcp and udp chat server using socket programming