CHAT ROOM CREATION
USING TCP AND UDP SOCKET PROGRAMMING

20CST51 COMPUTER

NETWORKS

A Mini Project Report submitted by

SARAN S

(21CSR180)

AIM:

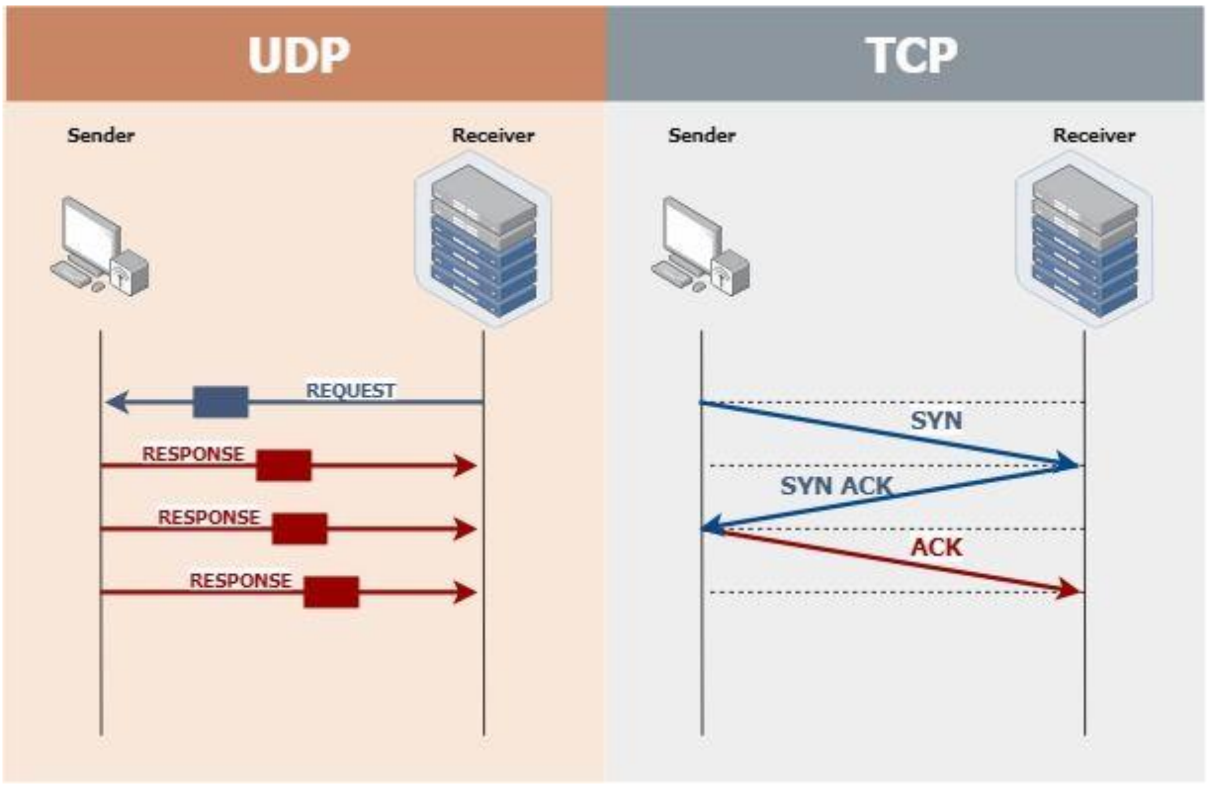
Develop a client-server socket program to create an interactive chatroom where users can share messages. The client allows users to input messages, which are transmitted to the server. The server receives the messages, and broadcasts them to all connected clients, creating a real-time chat experience. Clients can seamlessly participate in the conversation, providing an interactive environment for communication in a networked setting.

INTRODUCTION:

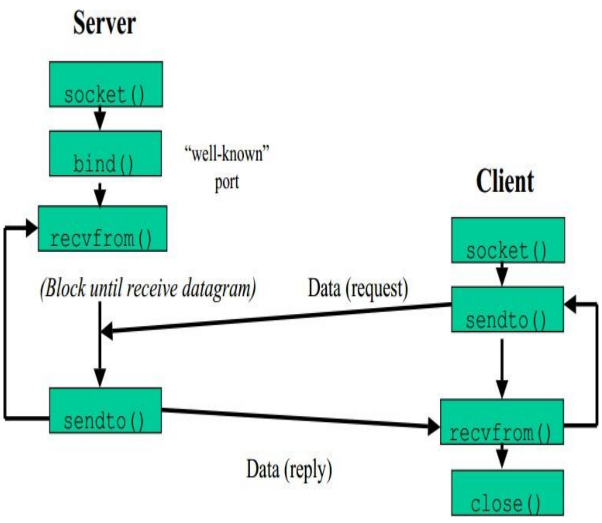
Socket programming forms the backbone of networked communication, enabling computers to exchange messages within a network. At its core are sockets, integral endpoints facilitating the seamless transmission of data between processes. This paradigm is particularly instrumental in crafting real-time, interactive applications like chatrooms, online gaming platforms, and distributed systems. Sockets are diverse, offering types such as stream and datagram sockets to accommodate varied communication requirements. Leveraging socket APIs, developers establish, configure, and manage these channels, fostering dynamic communication.

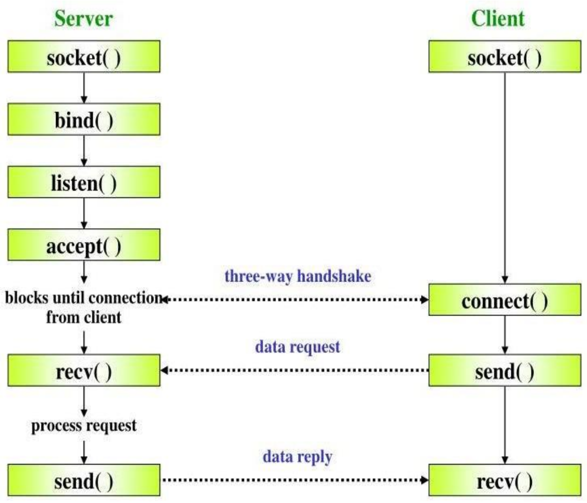
The client-server model, a prevalent application of sockets, unfolds with servers anticipating incoming connections while clients initiate and participate in ongoing conversations. Socket programming is not only ubiquitous but also pivotal in web development, serving as the foundation for protocols like HTTP and WebSocket. Beyond that, it plays a critical role in the Internet of Things (IoT) and facilitates seamless communication between processes on a local machine. Proficiency in socket programming empowers developers to construct responsive, scalable, and interconnected chatroom systems, establishing it as an essential skill in the domain of networked applications.

UDP VS TCP(WORKING):



UDP VS UDP(FUNCTIONS):





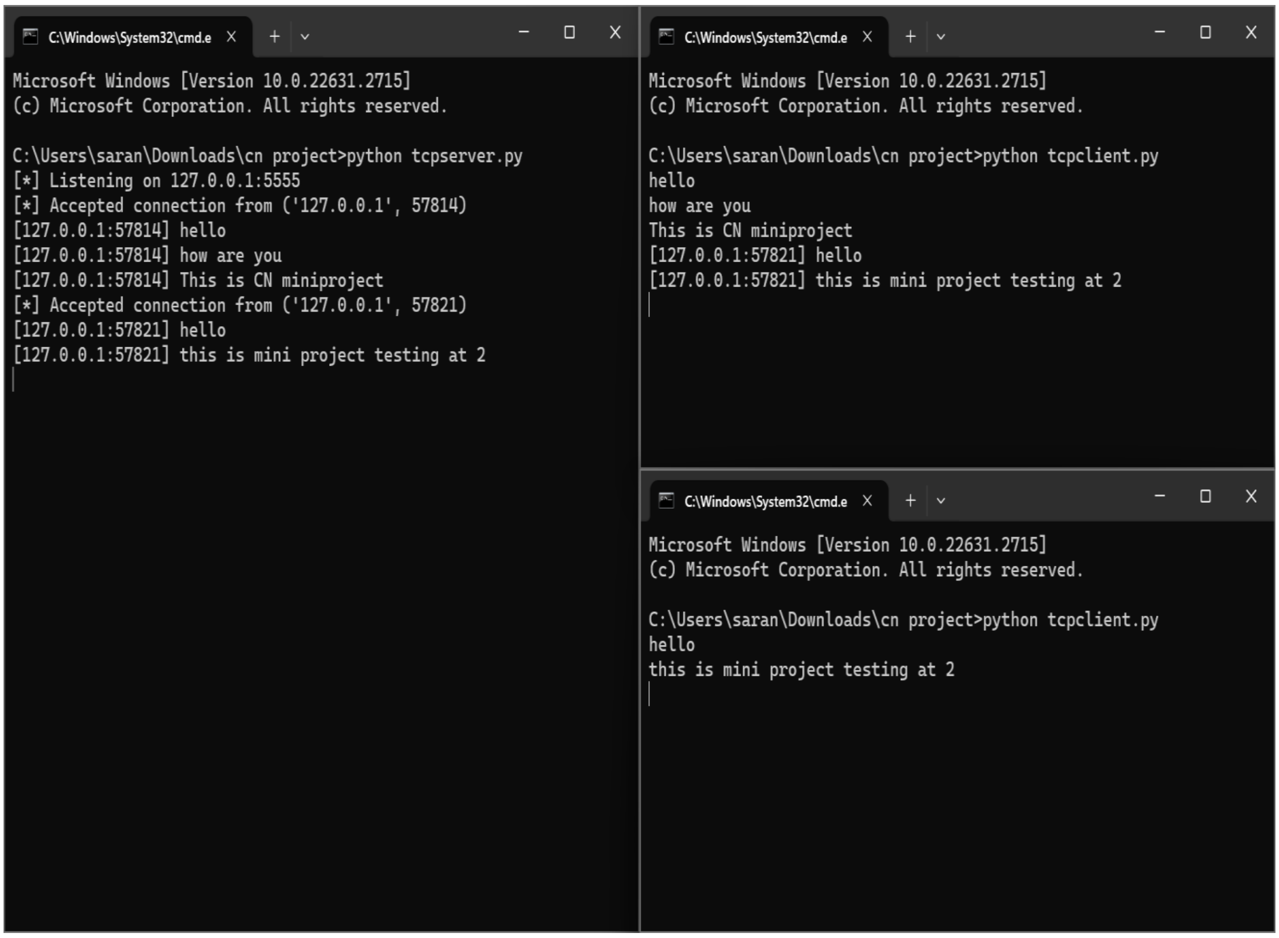
TCP Server -tcpserver.py:

```
import socket
import threading

def handle_client(client_socket, addr, clients):
    try:
        while True:
            data = client_socket.recv(1024)
            if not data:
                print(f"Connection closed by {addr}")
                clients.remove(client_socket)
                client_socket.close()
                break

            message = f"[{addr[0]}:{addr[1]}] {data.decode()}"
            print(message)

            # Broadcast the message to all clients
            for client in clients:
                if client != client_socket:
                    client.send(message.encode())
    except Exception as e:
        print(f"Error handling client {addr}: {e}")
        clients.remove(client_socket)
        client_socket.close()

def start_server():
    host = '127.0.0.1'
    port = 5555

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)

    print(f"[*] Listening on {host}:{port}")

    clients = []

    while True:
        client_socket, addr = server.accept()
        clients.append(client_socket)
        print(f"[*] Accepted connection from {addr}")
        client_handler = threading.Thread(target=handle_client,
args=(client_socket, addr, clients))
        client_handler.start()

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

TCP Client – tcpclient.py:

```python
import socket
import threading

def receive_messages(client_socket):
    while True:
        try:
            data = client_socket.recv(1024)
            if not data:
                break
            print(data.decode())
        except Exception as e:
            print(f"Error receiving message: {e}")
            break

def start_client():
    host = '127.0.0.1'
    port = 5555

    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    receive_thread = threading.Thread(target=receive_messages, args=(client,))
    receive_thread.start()

    while True:
        message = input()
        client.send(message.encode())

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

OUTPUT:



```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saran\Downloads\cn project>python tcpserver.py
[*] Listening on 127.0.0.1:5555
[*] Accepted connection from ('127.0.0.1', 57814)
[127.0.0.1:57814] hello
[127.0.0.1:57814] how are you
[127.0.0.1:57814] This is CN miniproject
[*] Accepted connection from ('127.0.0.1', 57821)
[127.0.0.1:57821] hello
[127.0.0.1:57821] this is mini project testing at 2
```

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saran\Downloads\cn project>python tcpclient.py
hello
how are you
This is CN miniproject
[127.0.0.1:57821] hello
[127.0.0.1:57821] this is mini project testing at 2
```

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saran\Downloads\cn project>python tcpclient.py
hello
this is mini project testing at 2
```

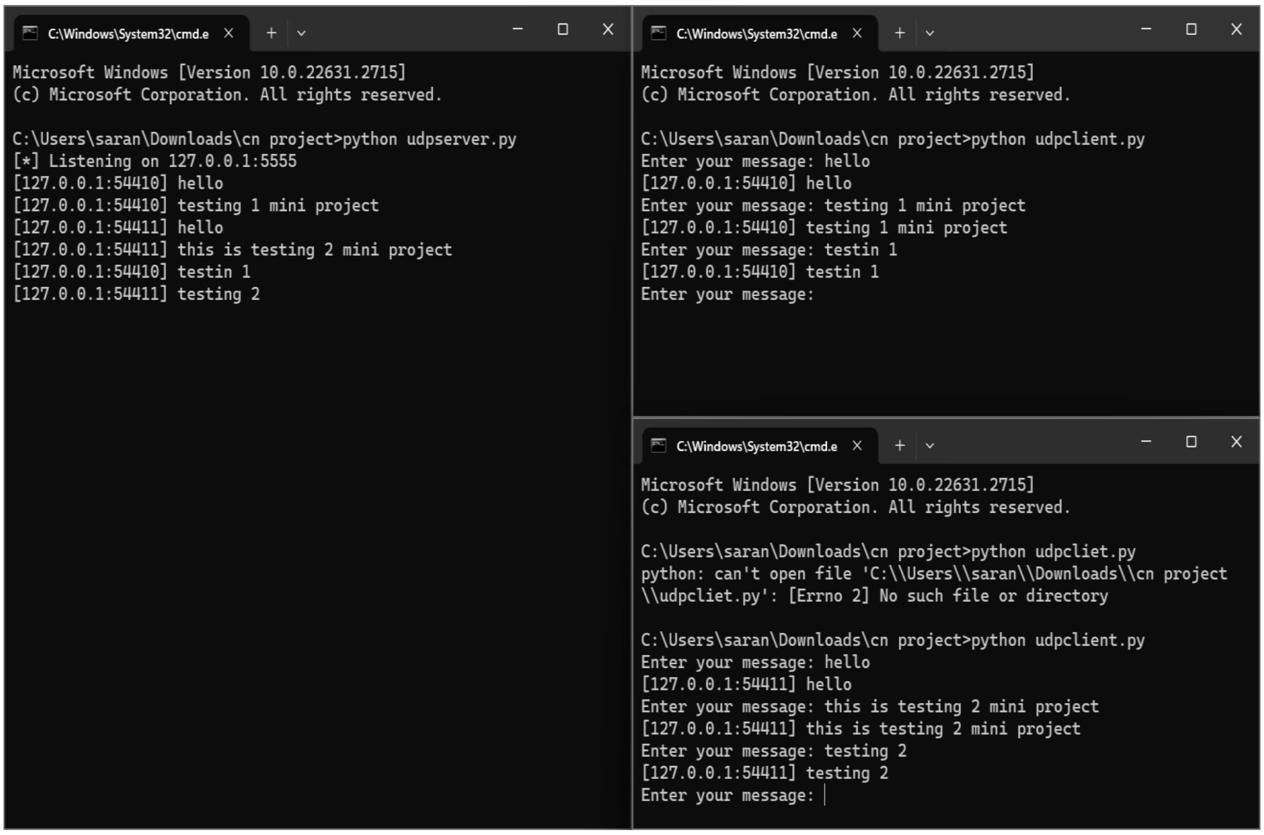
UDP Server – udpserver.py:

```python
import socket

def handle_client(data, client_addr, server_socket):
    try:
        message = f"[{client_addr[0]}:{client_addr[1]}] {data.decode()}"
        print(message)

        # Broadcast the message to all clients
        server_socket.sendto(message.encode(), client_addr)
    except Exception as e:
        print(f"Error handling client {client_addr}: {e}")

def start_server():
    host = '127.0.0.1'
    port = 5555

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind((host, port))

    print(f"[*] Listening on {host}:{port}")

    while True:
        data, client_addr = server_socket.recvfrom(1024)
        handle_client(data, client_addr, server_socket)

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

UDP Client – client.py:

```
import socket

def start_client():
    host = '127.0.0.1'
    port = 5555

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    while True:
        try:
            message = input("Enter your message: ")
            client_socket.sendto(message.encode(), (host, port))

            data, _ = client_socket.recvfrom(1024)
            print(data.decode())
        except Exception as e:
            print(f"Error receiving/sending message: {e}")

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

OUTPUT:



```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saran\Downloads\cn project>python udpserver.py
[*] Listening on 127.0.0.1:5555
[127.0.0.1:54410] hello
[127.0.0.1:54410] testing 1 mini project
[127.0.0.1:54411] hello
[127.0.0.1:54411] this is testing 2 mini project
[127.0.0.1:54410] testin 1
[127.0.0.1:54411] testing 2
```

```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saran\Downloads\cn project>python udpclient.py
Enter your message: hello
[127.0.0.1:54410] hello
Enter your message: testing 1 mini project
[127.0.0.1:54410] testing 1 mini project
Enter your message: testin 1
[127.0.0.1:54410] testin 1
Enter your message:
```

```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saran\Downloads\cn project>python udpcliet.py
python: can't open file 'C:\\Users\\saran\\Downloads\\cn project
\\udpcliet.py': [Errno 2] No such file or directory

C:\Users\saran\Downloads\cn project>python udpclient.py
Enter your message: hello
[127.0.0.1:54411] hello
Enter your message: this is testing 2 mini project
[127.0.0.1:54411] this is testing 2 mini project
Enter your message: testing 2
[127.0.0.1:54411] testing 2
Enter your message:
```

RESULT:

We have successfully implemented socket programming for the "Creation of the Chatroom" using UDP and TCP.