Smit Sutariya **21BCP142**

Assignment: 7

Part A

Class B Tech CSE 3rd Year **Sub: Computer Networks**

Aim:
Introduction to Socket Programming- Design and Implement client-server elements of a few network applications e.g. Echo client and server, Time client and server, Online Quiz and Buzzer Application, etc.

Prerequisite: Nil

Outcome: To impart knowledge of Socket Programming

Theory:

Socket programming is like the language computers use to talk to each other over a network. It's the way computers share information. Think of a socket as a phone number for a computer, and you can use it to call or answer calls.

In socket programming, one computer can be like a phone that waits for calls (the server), and others can be like phones that make calls (the clients). There are two types of calls: one where you talk and make sure the other person hears you clearly (TCP), and another where you speak quickly, but you're not sure if the other person got all your words (UDP).

The phones can do many things: create a phone, tell the phone where to listen for calls, pick up calls, make calls, send and receive messages, and hang up when done.

So, socket programming helps computers chat over the internet by creating these virtual phones and making sure the conversation goes smoothly. It's like making sure you and your friend can talk on the phone without missing any important details.

Procedure:

1. Write Simple Client Server Program using Java/Python Programming Language
2. Execute the program using appropriate compiler.
3. Verify the working of the program.

Part B

Steps:

Server_lab7.py

```python
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(("127.0.0.1", 12345))
server_socket.listen(5)
print("Server is listening...")
while True:
    client_socket, client_address = server_socket.accept()
    print(f"Accepted connection from {client_address}")
```

```
    while True:
        data = client_socket.recv(1024)
        if not data:
            break
        message = data.decode()
        print(f"Received from client: {message}")

        # Echo the received message back to the client
        client_socket.send(data)

    client_socket.close()
```

client_lab7.py

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(("127.0.0.1", 12345))
try:
    while True:
        message = input("Enter a message to send to the server (or type 'exit' to
quit): ")
        client_socket.send(message.encode())

        if message.lower() == "exit":
            break

        data = client_socket.recv(1024)
        print(f"Received from server: {data.decode()}")
except KeyboardInterrupt:
    pass
client_socket.close()
```

Output:

```
PS D:\Sem-5\Network Lab\Lab_7> python -u "d:\Sem-5\Network Lab\Lab_7\server.py"
Server is listening...
```

```
Server is listening...
Accepted connection from ('127.0.0.1', 56601)
Received from client: heyy
```

```
Enter a message to send to the server (or type 'exit' to quit): i am smit sutariya
Received from server: i am smit sutariya
Enter a message to send to the server (or type 'exit' to quit):
```

```
D:\Sem-5\Network Lab\Lab_7>python client.py
Enter a message to send to the server (or type 'exit' to quit): heyy
Received from server: heyy
Enter a message to send to the server (or type 'exit' to quit): i am smit sutariya
Received from server: i am smit sutariya
Enter a message to send to the server (or type 'exit' to quit): i am first roll no. 21BCP142
Received from server: i am first roll no. 21BCP142
Enter a message to send to the server (or type 'exit' to quit): done with day
Received from server: done with day
Enter a message to send to the server (or type 'exit' to quit):
```

```
PS D:\Sem-5\Network Lab\Lab_7> python -u "d:\Sem-5\Network Lab\Lab_7\server.py"
Server is listening...
Accepted connection from ('127.0.0.1', 56601)
Received from client: heyy
Received from client: i am smit sutariya
Received from client: i am first roll no. 21BCP142
Received from client: done with day
```

Observation & Learning:

Socket programming is the foundation of network communication, enabling two devices, such as clients and servers, to exchange data over a network.

- **Server:** We set up a server script (server_lab7.py) using Python's socket library. It listened on IP 127.0.0.1 and port 12345, welcoming incoming connections.
- **Client:** We developed a client script (client_lab7.py) to connect to the server on the same IP and port. It allowed users to input messages for the server and offered an exit option.
- **Message Handling:** The server echoed messages it received from the client, illustrating data transfer between them.

- **Outcome:** This experiment demonstrated successful communication between the client and server through socket connections, serving as a foundation for understanding network applications.

Conclusion:
In conclusion, this experiment provided valuable insights into socket programming and the development of basic network applications. We observed that the server and client successfully connected, sent, and received messages, highlighting the underlying principles of socket-based communication. This hands-on experience serves as a foundation for understanding more complex network applications and protocols.

Questions:

1. **What is Socket?**
   ANS: A socket is a software endpoint that allows communication between two computers over a network. It provides a mechanism for processes (programs) running on different devices to exchange data. Sockets can be used to establish connections, send and receive data, and manage network communication.

2. **Which socket is used for the communication between the client and server?**
   ANS: In the context of client-server communication, both the client and server use sockets. The server creates a socket that listens for incoming connections, while the client creates a socket to initiate a connection to the server. Typically, the server socket is used to accept incoming connections, and the client socket is used to establish a connection with the server.

3. **What are the different operation supported on sockets?**
   ANS: Sockets support various operations, including:

   - **Socket Creation:** Creating a socket to prepare for communication.
   - **Binding:** Associating a socket with a specific address and port.
   - **Listening:** Setting a socket to listen for incoming connections (usually on the server side).
   - **Accepting:** Accepting incoming client connections (server-side).
   - **Connecting:** Initiating a connection to a server (client-side).
   - **Sending:** Transmitting data from one end to the other.
   - **Receiving:** Receiving data at the other end.
   - **Closing:** Closing the socket connection once the communication is done.