**Experiment 07- Implement socket programming**

**Learning Objective:** Implement basic socket programming

**Tools:**  VS Code, python ‘socket’ library

**Theory:**

* **Socket Programming**: Facilitates communication between computers over a network using endpoints called sockets.
* **TCP/IP Protocol**: Ensures reliable data transmission with connection-oriented communication (TCP).
* **Client-Server Model**: Server listens for connections and handles multiple clients, while clients connect to the server to exchange data.

#### **Code Description:**

**server.py**:

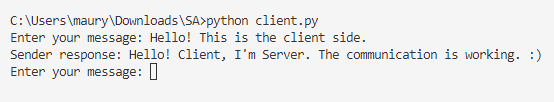
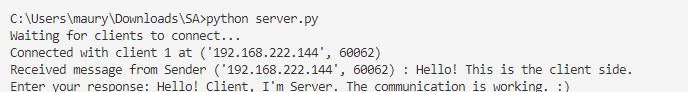
* **Function**: Implements a TCP server that listens on port 8080.
* **Process**: Accepts a single client connection, receives messages from the client, and responds based on user input. The server closes the connection after the client disconnects.

**client.py**:

* **Function**: Implements a TCP client that connects to the server's IP address and port 8080.
* **Process**: Sends user-inputted messages to the server and prints the responses received from the server. The client continues this process until closed by the user.

| server.py | client.py |
| --- | --- |
| import socket  if \_\_name\_\_ == '\_\_main\_\_':  host = '0.0.0.0'  port = 8080  totalclient = 1  sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  sock.bind((host, port))  sock.listen(totalclient)  connections = []  print('Waiting for clients to connect...')  for i in range(totalclient):  conn, addr = sock.accept()  connections.append((conn, addr))  print('Connected with client', i+1, 'at', addr)  for conn, addr in connections:  while True:  data = conn.recv(1024).decode()  if not data:  break  print('Received message from Sender', addr, ':', data)  response = input('Enter your response: ')  conn.send(response.encode())  conn.close()  sock.close() | import socket  if \_\_name\_\_ == '\_\_main\_\_':  host = '192.168.222.144'  port = 8080  sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  sock.connect((host, port))  while True:  message = input('Enter your message: ')  sock.sendall(message.encode())  response = sock.recv(1024).decode()  print('Sender response:', response)  sock.close() |

Output:

**Learning Outcomes:** The student should have the ability to:

LO 1: Understand and explain non-functional requirements in the context of networked applications.

LO 2: Identify and elaborate on various non-functional requirements such as performance, scalability, and reliability in socket programming.

**Course Outcomes:** Upon completion of the course students will be able to understand and explain the fundamentals of network programming, including socket creation, connection management, and data exchange using TCP/IP.

**Conclusion:**

**For Faculty Use**

| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** | **Total** |
| --- | --- | --- | --- | --- |
| **Marks Obtained** |  |  |  |  |