CS23532-COMPUTER NETWORKS-LAB MANUAL

Practical 12

AIM: - a) Implement echo client server using TCP/UDP sockets.

A. TCP Echo Client-Server

TCP Server (tcp_server.py)

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind to local host and port
server_socket.bind(('localhost', 5000))

# Listen for connections
server_socket.listen(1)
print("Server listening on port 5000...")

# Accept a connection
conn, addr = server_socket.accept()
print(f"Connected to client {addr}")

# Communication loop
while True:
    data = conn.recv(1024).decode()
    if not data:
        break
    print("Received from client:", data)

    # Echo back the same message
    conn.send(data.encode())
    print("Echoed back:", data)

# Close connection
conn.close()
server_socket.close()
```

TCP Client (tcp_client.py)

```
import socket

# Create TCP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to server
client_socket.connect(('localhost', 5000))
print("Connected to server on port 5000")

# Send and receive messages
while True:
    msg = input("Enter message (type 'exit' to quit): ")
    if msg.lower() == 'exit':
        break
    client_socket.send(msg.encode())
    data = client_socket.recv(1024).decode()
    print("Received echo:", data)

# Close connection
client_socket.close()
```

B. UDP Echo Client-Server

UDP Server (udp_server.py)

```
import socket

# Create UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind to local address and port
server_socket.bind(('localhost', 5001))
print("UDP Server listening on port 5001...")

while True:
    data, addr = server_socket.recvfrom(1024)
    print("Received from client:", data.decode())

    # Echo message back to client
    server_socket.sendto(data, addr)
    print("Echoed back to client", addr)
```

UDP Client (udp_client.py)

```
import socket

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

```
server_address = ('localhost', 5001)
while True:
    msg = input("Enter message (type 'exit' to quit): ")
    if msg.lower() == 'exit':
        break
    client_socket.sendto(msg.encode(), server_address)
    data, _ = client_socket.recvfrom(1024)
    print("Received echo:", data.decode())
client_socket.close()
```

Input:

Client Input:

```
Enter message (type 'exit' to quit): Hello Server!
```

Output:

Server Side:

```
Server listening on port 5000...
Connected to client ('127.0.0.1', 58734)
Received from client: Hello Server!
Echoed back: Hello Server!
```

Client Side:

```
Connected to server on port 5000
Enter message (type 'exit' to quit): Hello Server!
Received echo: Hello Server!
```

Result:

The Echo Client-Server was successfully implemented using both TCP and UDP sockets. The server receives a message from the client and sends it back (echo), confirming correct data transmission at the Transport Layer of the OSI Model.

CS23532-COMPUTER NETWORKS-LAB MANUAL

CS23532-COMPUTER NETWORKS-LAB MANUAL

Practical 12

AIM: - b) Implement chat client server using TCP/UDP sockets.

A. TCP Chat Client-Server

TCP Server (tcp_chat_server.py)

```
import socket

# Create TCP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind and listen
server_socket.bind(('localhost', 6000))
server_socket.listen(1)
print("TCP Chat Server is running on port 6000...")

# Accept connection
conn, addr = server_socket.accept()
print(f"Connected with client {addr}")

while True:
    # Receive message from client
    client_msg = conn.recv(1024).decode()
    if client_msg.lower() == 'exit':
        print("Client ended the chat.")
        break
    print(f"Client: {client_msg}")

    # Send reply to client
    server_msg = input("Server: ")
    conn.send(server_msg.encode())

    if server_msg.lower() == 'exit':
        print("Chat ended by server.")
        break

conn.close()
server_socket.close()
```

TCP Client (tcp_chat_client.py)

```
import socket
```

Practical 12

```
# Create TCP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to server
client_socket.connect(('localhost', 6000))
print("Connected to TCP Chat Server (type 'exit' to quit)")

while True:
    # Send message
    msg = input("You: ")
    client_socket.send(msg.encode())

    if msg.lower() == 'exit':
        print("Chat ended by you.")
        break

    # Receive reply from server
    data = client_socket.recv(1024).decode()
    if data.lower() == 'exit':
        print("Server ended the chat.")
        break
    print(f"Server: {data}")

client_socket.close()
```

B. UDP Chat Client-Server

UDP Server (udp_chat_server.py)

```
import socket

# Create UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('localhost', 6001))
print("UDP Chat Server running on port 6001...")

while True:
    # Receive message
    data, addr = server_socket.recvfrom(1024)
    msg = data.decode()
    if msg.lower() == 'exit':
        print("Client ended the chat.")
        break
    print(f"Client: {msg}")
```

Practical 12

```
    # Send reply
    reply = input("Server: ")
    server_socket.sendto(reply.encode(), addr)

    if reply.lower() == 'exit':
        print("Chat ended by server.")
        break

server_socket.close()
```

UDP Client (udp_chat_client.py)

```
import socket

# Create UDP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_addr = ('localhost', 6001)

print("Connected to UDP Chat Server (type 'exit' to quit)")

while True:
    # Send message
    msg = input("You: ")
    client_socket.sendto(msg.encode(), server_addr)

    if msg.lower() == 'exit':
        print("Chat ended by you.")
        break

    # Receive reply
    data, _ = client_socket.recvfrom(1024)
    reply = data.decode()
    if reply.lower() == 'exit':
        print("Server ended the chat.")
        break
    print(f"Server: {reply}")

client_socket.close()
```

Client:

```
You: Hi Server!
Server: Hello! How are you?
```

Practical 12

```
You: I'm good, exit
Chat ended by you.
```

Server:

```
TCP Chat Server is running on port 6000...
Connected with client ('127.0.0.1', 58234)
Client: Hi Server!
Server: Hello! How are you?
Client: I'm good, exit
Client ended the chat.
```

Result:

A bidirectional Chat Application was successfully implemented using both TCP and UDP sockets in Python.
Both ends can send and receive messages in real-time until either party types ‘exit’ to terminate the session.