Lab 4: Implementing a UDP-Based Messaging System

Creating the UDP Server

The UDP server will bind to a specific IP address and port. It will listen for incoming messages from clients, respond, and continue listening for more messages.

Server Program

```
import socket
def start_udp_server():
    # Create a UDP socket object
    udp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # Bind the socket to a specific IP address and port
    udp_server_socket.bind(('0.0.0.0', 12345))  # '0.0.0.0' allows any network
interface
    print("UDP server is listening on port 12345...")
    while True:
        # Receive data from the client (1024 bytes max)
        client_message, client_address = udp_server_socket.recvfrom(1024)
        client_message = client_message.decode('utf-8')
        print(f"Received message from {client_address}: {client_message}")
        # Send a response back to the client
        server_response = "Hello from the UDP server!"
        udp_server_socket.sendto(server_response.encode('utf-8'), client_address)
if __name__ == "__main__":
    start_udp_server()
```

UDP Client Code:

The UDP client will send a message to the server's IP address and port. It will also wait for the server's response before closing the connection.

Client Program

```
import socket
def start_client():
    # Create a socket object using IPv4 (AF_INET) and TCP (SOCK_STREAM)
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Connect to the server (replace 'server_ip' with the actual server IP
address)
    client_socket.connect(('127.0.0.1', 12345))  # Use 'localhost' for testing on
the same machine
    while True:
        # Send a message to the server
        client_message = input("Client: ")
        client_socket.send(client_message.encode('utf-8'))
        # Receive a response from the server
        server_message = client_socket.recv(1024).decode('utf-8')
        if not server_message:
            break  # Break the loop if the server sends an empty message
        print(f"Server: {server_message}")
    # Close the connection
    client_socket.close()
if __name__ == "__main__":
    start_client()
```