Algorithm for UDP Client-Server Communication

Server Algorithm (UDP Server)

1. **Start the server.**
2. **Create a UDP socket.**
    - Use socket(AF_INET, SOCK_DGRAM, 0).
3. **Bind the socket to an IP address and port.**
    - Use bind() to assign the server socket to INADDR_ANY and PORT 8080.
4. **Wait for a client to send an initial message.**
    - Use recvfrom() to receive data from the client.
    - Store the client's address for future communication.
5. **Enter an infinite loop to send messages to the client.**
    - Prompt the user to enter a message.
    - Read input using fgets().
    - Send the message to the client using sendto().
6. **Repeat step 5 until terminated manually.**
7. **Close the socket using close().**

Client Algorithm (UDP Client)

1. **Start the client.**
2. **Create a UDP socket.**
    - Use socket(AF_INET, SOCK_DGRAM, 0).
3. **Set up the server's address and port.**
    - Use inet_addr("127.0.0.1") for localhost or specify another IP if needed.
    - Set PORT 8080.
4. **Send an initial message to register with the server.**
    - Use sendto() to inform the server of the client's address.
5. **Enter an infinite loop to receive messages from the server.**

- Use recvfrom() to receive messages.
- Display the received message.

6. **Repeat step 5 until terminated manually.**
7. **Close the socket using close().**

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int server_sock;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];

    // Create UDP socket
    if ((server_sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Server address setup
```

```
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind socket to address
    if (bind(server_sock, (const struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Bind failed");
        close(server_sock);
        exit(EXIT_FAILURE);
    }

    printf("UDP Server is running on port %d...\n", PORT);

    // Wait for the first message from the client
    recvfrom(server_sock, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&client_addr,
&client_len);
    printf("Client connected!\n");

    while (1) {
        printf("Enter message to send: ");
        fgets(buffer, BUFFER_SIZE, stdin);

        // Send message to client
        sendto(server_sock, buffer, strlen(buffer), 0, (struct sockaddr *)&client_addr,
client_len);
        printf("Message sent: %s", buffer);
    }

    close(server_sock);
    return 0;
```

```
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define SERVER_IP "127.0.0.1"  // Change this if server runs on another machine
#define BUFFER_SIZE 1024

int main() {
    int client_sock;
    struct sockaddr_in server_addr;
    socklen_t addr_len = sizeof(server_addr);
    char buffer[BUFFER_SIZE] = "Hello, server!";  // Initial message to register client

    // Create UDP socket
    if ((client_sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Server address setup
    memset(&server_addr, 0, sizeof(server_addr));
```

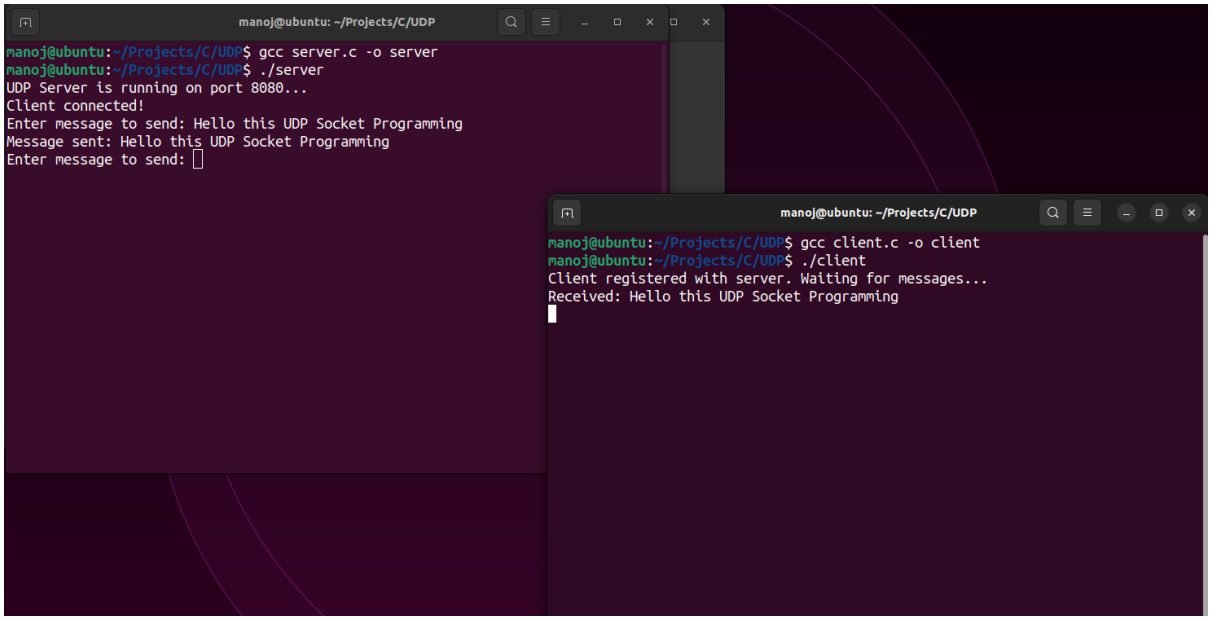
```
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

    // Send an initial message to server
    sendto(client_sock, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr, addr_len);
    printf("Client registered with server. Waiting for messages...\n");

    while (1) {
        memset(buffer, 0, BUFFER_SIZE);

        // Receive message from server
        recvfrom(client_sock, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&server_addr,
&addr_len);
        printf("Received: %s", buffer);
    }

    close(client_sock);
    return 0;
}
```


manoj@ubuntu: ~/Projects/C/UDP
manoj@ubuntu:~/Projects/C/UDP$ gcc server.c -o server
manoj@ubuntu:~/Projects/C/UDP$ ./server
UDP Server is running on port 8080...
Client connected!
Enter message to send: Hello this UDP Socket Programming
Message sent: Hello this UDP Socket Programming
Enter message to send:
manoj@ubuntu: ~/Projects/C/UDP
manoj@ubuntu:~/Projects/C/UDP$ gcc client.c -o client
manoj@ubuntu:~/Projects/C/UDP$ ./client
Client registered with server. Waiting for messages...
Received: Hello this UDP Socket Programming