# EXPERIMENT-32

**IMPLEMENTATION OF SERVER – CLIENT USING UDP SOCKET PROGRAMMING**

**Aim:**

To implement a server-client communication model using UDP socket programming in C.

**Software/Apparatus Required:**

* C Compiler (GCC or any compatible compiler)
* Linux-based OS (or any OS supporting POSIX sockets)
* Text editor (e.g., Vim, Nano, or any IDE)

**Procedure:**

**Step 1: Write the Server-Side Code**

1. Open a text editor and write the server-side C program as provided.
2. Save the file as udp\_server.c.

**Step 2: Write the Client-Side Code**

1. Open a text editor and write the client-side C program as provided.
2. Save the file as udp\_client.c. **Step 3: Compile the Programs**
3. Open the terminal and navigate to the directory containing the udp\_server.c and udp\_client.c files.
4. Compile the server program using the following command: gcc udp\_server.c -o udp\_server
5. Compile the client program using the following command:

gcc udp\_client.c -o udp\_client

**Step 4: Run the Server**

1. Execute the server program using the following command:

./udp\_server

1. The server will start listening on port 5000.

**Step 5: Run the Client**

1. Open another terminal window and navigate to the same directory.
2. Execute the client program using the following command:

./udp\_client

1. The client will send a message to the server running on 127.0.0.1 (localhost) and port 5000.

**Step 6: Test the Communication**

1. The client sends a message ("Hello Server") to the server.
2. The server receives the message, prints it, and sends a response ("Hello Client") back to the client.
3. The client receives the server's response and prints it.

**Code:**

Implementation of server – client using UDP socket programming

// server program for udp connection

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include<netinet/in.h>

#define PORT 5000

#define MAXLINE 1000

// Driver code int main()

{ char buffer[100]; char \*message = "Hello Client"; int listenfd, len;

struct sockaddr\_in servaddr, cliaddr; bzero(&servaddr, sizeof(servaddr));

// Create a UDP Socket listenfd = socket(AF\_INET, SOCK\_DGRAM, 0); servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY); servaddr.sin\_port = htons(PORT); servaddr.sin\_family = AF\_INET;

// bind server address to socket descriptor

bind(listenfd, (struct sockaddr\*)&servaddr, sizeof(servaddr));

//receive the datagram len = sizeof(cliaddr); int n = recvfrom(listenfd, buffer, sizeof(buffer),

0, (struct sockaddr\*)&cliaddr,&len); //receive message from server buffer[n] = '\0'; puts(buffer);

// send the response sendto(listenfd, message, MAXLINE, 0,

(struct sockaddr\*)&cliaddr, sizeof(cliaddr));

}

// udp client driver program

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include<netinet/in.h>

#include<unistd.h>

#include<stdlib.h>

#define PORT 5000

#define MAXLINE 1000

// Driver code int main()

{ char buffer[100]; char \*message = "Hello Server"; int sockfd, n;

struct sockaddr\_in servaddr;

// clear servaddr bzero(&servaddr, sizeof(servaddr)); servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); servaddr.sin\_port = htons(PORT); servaddr.sin\_family = AF\_INET;

// create datagram socket sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

// connect to server if(connect(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)

{

printf("\n Error : Connect Failed \n"); exit(0);

}

// request to send datagram

// no need to specify server address in sendto // connect stores the peers IP and port sendto(sockfd, message, MAXLINE, 0, (struct sockaddr\*)NULL, sizeof(servaddr));

// waiting for response

recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr\*)NULL, NULL); puts(buffer);

// close the descriptor close(sockfd);

}

**Output:**

1. Server-side output:

Hello Server

1. Client-side output:

Hello Client

**Result:**

Thus, the server-client communication using UDP socket programming was implemented successfully.