MIT WORLD PEACE UNIVERSITY

Computer Networks
Second Year B. Tech, Semester 3

UDP SOCKET PROGRAMMING

PRACTICAL REPORT
ASSIGNMENT 9

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 29, 2022

Contents

1 Aim and Objectives

To understand Concept of UDP Socket programming.

2 Problem Statement

Write a C Program for wired network using udp socket to perform Reversing of a String.

3 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Programs Used: Cisco Packet Tracer v6.0.1

4 Code

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>

void reverse_string(char *str)
{
    int i, j;
    char temp;
    for (i = 0, j = strlen(str) - 1; i < j; i++, j--)
    {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}

int main(void)
{
    int socket_desc;
    struct sockaddr_in server_addr, client_addr;
    char server_message[2000], client_message[2000];
    int client_struct_length = sizeof(client_addr);

    // Clean buffers:
    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    // Create UDP socket:
    socket_desc = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    if (socket_desc < 0)
    {
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");

```

```c
    // Set port and IP:
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Bind to the set port and IP:
    if (bind(socket_desc, (struct sockaddr *)&server_addr, sizeof(server_addr)) <
   0)
    {
        printf("Couldn't bind to the port\n");
        return -1;
    }
    printf("Done with binding\n");

    printf("Listening for incoming messages...\n\n");

    // Receive client's message:
    if (recvfrom(socket_desc, client_message, sizeof(client_message), 0,
                 (struct sockaddr *)&client_addr, &client_struct_length) < 0)
    {
        printf("Couldn't receive\n");
        return -1;
    }
    printf("Received message from IP: %s and port: %i\n",
           inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

    reverse_string(client_message);
    printf("Msg from client: %s\n", client_message);

    // Respond to client:
    strcpy(server_message, client_message);

    if (sendto(socket_desc, server_message, strlen(server_message), 0,
               (struct sockaddr *)&client_addr, client_struct_length) < 0)
    {
        printf("Can't send\n");
        return -1;
    }

    // Close the socket:
    close(socket_desc);

    return 0;
}
```

Listing 1: Server

```c
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(void)
{
    int socket_desc;
    struct sockaddr_in server_addr;
    char server_message[2000], client_message[2000];
    int server_struct_length = sizeof(server_addr);

    // Clean buffers:
    memset(server_message, ’\0’, sizeof(server_message));
    memset(client_message, ’\0’, sizeof(client_message));

    // Create socket:
    socket_desc = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    if (socket_desc < 0)
    {
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");

    // Set port and IP:
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Get input from the user:
    printf("Enter message: ");
    gets(client_message);

    // Send the message to server:
    if (sendto(socket_desc, client_message, strlen(client_message), 0,
               (struct sockaddr *)&server_addr, server_struct_length) < 0)
    {
        printf("Unable to send message\n");
        return -1;
    }

    // Receive the server’s response:
    if (recvfrom(socket_desc, server_message, sizeof(server_message), 0,
                 (struct sockaddr *)&server_addr, &server_struct_length) < 0)
    {
        printf("Error while receiving server’s msg\n");
        return -1;
    }

    printf("Server’s response: %s\n", server_message);

    // Close the socket:
    close(socket_desc);

    return 0;
}
```

Listing 2: Client

5 Output

SERVER

```
Socket created successfully
Done with binding
Listening for incoming messages...
```

```
Received message from IP: 127.0.0.1 and port: 34067
Msg from client: olleh

CLIENT

Socket created successfully
Enter message: hello
Server’s response: olleh
```