Lab Practical #06: Study Client-Server Socket programming - TCP & UDP

Student Name: Dhairya Adroja

Enrollment No: 24010101602

Course: B.Tech. CSE

Aim/Objective

To implement Client-Server Socket Programming using TCP and UDP protocols in C/Java.

Theory

Socket programming enables communication between processes over a network. TCP provides reliable, connection-oriented communication while UDP offers faster, connectionless communication.

Procedure

1. TCP Socket Programming

TCP Server Program:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    char buffer[1024] = {0};
    char *hello = "Hello from TCP server";

    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(8080);

    bind(server_fd, (struct sockaddr *)&address, sizeof(address));
    listen(server_fd, 3);

    new_socket = accept(server_fd, NULL, NULL);
    read(new_socket, buffer, 1024);
    printf("Message from client: %s\n", buffer);
    send(new_socket, hello, strlen(hello), 0);

    close(new_socket);
    close(server_fd);
    return 0;
}
```

TCP Client Program:

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

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char *hello = "Hello from TCP client";
    char buffer[1024] = {0};

    sock = socket(AF_INET, SOCK_STREAM, 0);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(8080);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);

    connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    send(sock, hello, strlen(hello), 0);
    read(sock, buffer, 1024);
    printf("Message from server: %s\n", buffer);

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

Testing Results:

- Server listens on port 8080
- Client connects and exchanges messages
- Reliable data transmission verified

2. UDP Socket Programming

UDP Server Program:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

int main() {
    int sockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buffer[1024];
    char *hello = "Hello from UDP server";
    socklen_t len = sizeof(cliaddr);

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(8081);

    bind(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

    int n = recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&cliaddr, &len);
    buffer[n] = '\0';
    printf("Client: %s\n", buffer);

    sendto(sockfd, hello, strlen(hello), 0, (struct sockaddr*)&cliaddr, len);

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

UDP Client Program:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

int main() {
    int sockfd;
    struct sockaddr_in servaddr;
    char *hello = "Hello from UDP client";
    char buffer[1024];
    socklen_t len = sizeof(servaddr);

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(8081);
    inet_pton(AF_INET, "127.0.0.1", &servaddr.sin_addr);

    sendto(sockfd, hello, strlen(hello), 0, (struct sockaddr*)&servaddr, sizeof(servaddr));

    int n = recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&servaddr, &len);
    buffer[n] = '\0';
    printf("Server: %s\n", buffer);

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

Testing Results:

- UDP server listens on port 8081
- Client sends datagram and receives response
- Connectionless communication verified

Compilation and Execution

TCP:

```
gcc tcp_server.c -o tcp_server
gcc tcp_client.c -o tcp_client
./tcp_server    # Terminal 1
./tcp_client    # Terminal 2
```

UDP:

```
gcc udp_server.c -o udp_server
gcc udp_client.c -o udp_client
./udp_server    # Terminal 1
./udp_client    # Terminal 2
```

Conclusion

Successfully implemented TCP and UDP socket programming. TCP provides reliable, ordered data delivery while UDP offers faster, lightweight communication for applications where speed is more important than reliability.