VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2023-24

School Of Computer Science and Engineering (SCOPE)

Digital Assignment - IV

Computer Networks Lab

Apurva Mishra, 22BCE2791

8 Novemeber 2024

Problem 0.1.
Explain TCP and UDP with socket programming concept.

1. TCP Socket Programming

TCP is a connection-oriented protocol, which means that a connection is established and maintained until the application programs at each end have finished exchanging messages. It provides reliable and ordered delivery of data packets.

server.c (writer)

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

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_address, client_address;
    int port = 12345;
    char buffer[1024] = {0};

    // Create a TCP/IP socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("Error creating socket");
        exit(EXIT_FAILURE);
    }

    // Bind the socket to the host and port
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(port);
    server_address.sin_addr.s_addr = INADDR_ANY;
         if  (bind(server_socket,  (struct  sockaddr  *)&server_address,
sizeof(server_address)) < 0) {
        perror("Error binding socket");
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_socket, 1) < 0) {
        perror("Error listening for connections");
        exit(EXIT_FAILURE);
```

```
    }

    printf("TCP Server listening on port %d\n", port);

    while (1) {
        printf("Waiting for a connection...\n");
                client_socket = accept(server_socket, (struct sockaddr
*)&client_address, (socklen_t *)&(sizeof(client_address)));
        if (client_socket < 0) {
            perror("Error accepting connection");
            exit(EXIT_FAILURE);
        }

        printf("Connection from %s\n", inet_ntoa(client_address.sin_addr));

        // Receive the data in small chunks and retransmit it
        int bytes_read;
        while ((bytes_read = read(client_socket, buffer, 1024)) > 0) {
            printf("Received: %s\n", buffer);
            write(client_socket, buffer, bytes_read);
            memset(buffer, 0, 1024);
        }

        close(client_socket);
    }

    return 0;
}
```

client.c (reader)

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

int main() {
    int client_socket;
    struct sockaddr_in server_address;
    char buffer[1024] = {0};
    int port = 12345;

    // Create a TCP/IP socket
```

```c
    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Error creating socket");
        exit(EXIT_FAILURE);
    }

    // Connect the socket to the server
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(port);
    server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
    if (connect(client_socket, (struct sockaddr *)&server_address,
sizeof(server_address)) < 0) {
        perror("Error connecting to server");
        exit(EXIT_FAILURE);
    }

    // Send data
    char message[] = "Hello, world";
    printf("Sending: %s\n", message);
    write(client_socket, message, strlen(message));

    // Receive the response
    int bytes_read = read(client_socket, buffer, 1024);
    printf("Received: %s\n", buffer);

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

Output: Sender and Receiver

```
da/ass4/q2 via C v16.0.0-clang
❯ ./a.out
TCP Server listening on port 12345
Waiting for a connection...
Connection from 127.0.0.1
Received: Hello, world
Waiting for a connection...
```

The key features of TCP:

- TCP is a connection-oriented protocol, which means that a connection is established and maintained until the application programs at each end have finished exchanging messages.
- The server creates a socket, binds it to a specific port, and listens for incoming connections.
- The client creates a socket and connects to the server's socket.
- The server and client can then exchange data through the established connection.
- The server and client must handle the connection lifecycle, including opening, using, and closing the connection.

2. UDP Socket Programming

UDP is a connectionless protocol, which means that there is no need to establish a connection before sending data. It provides unreliable and unordered delivery of data packets.

server.c (writer)

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

int main() {
    int server_socket;
    struct sockaddr_in server_address, client_address;
    int port = 12345;
    char buffer[1024] = {0};
    socklen_t client_address_len = sizeof(client_address);

    // Create a UDP socket
    server_socket = socket(AF_INET, SOCK_DGRAM, 0);
    if (server_socket < 0) {
        perror("Error creating socket");
        exit(EXIT_FAILURE);
    }

    // Bind the socket to the host and port
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(port);
    server_address.sin_addr.s_addr = INADDR_ANY;
         if   (bind(server_socket,   (struct   sockaddr   *)&server_address,
sizeof(server_address)) < 0) {
        perror("Error binding socket");
        exit(EXIT_FAILURE);
    }

    printf("UDP Server listening on port %d\n", port);

    while (1) {
        printf("Waiting to receive message...\n");
       int bytes_received = recvfrom(server_socket, buffer, 1024, 0, (struct
sockaddr *)&client_address, &client_address_len);
        if (bytes_received < 0) {
            perror("Error receiving message");
            exit(EXIT_FAILURE);
        }
```

```
                printf("Received %d bytes from %s\n", bytes_received,
inet_ntoa(client_address.sin_addr));
        printf("Data: %s\n", buffer);

        printf("Sending acknowledgment\n");
        char ack[] = "ACK";
            sendto(server_socket, ack, strlen(ack), 0, (struct sockaddr
*)&client_address, client_address_len);
    }

    return 0;
}
```

client.c (reader)

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

int main() {
    int client_socket;
    struct sockaddr_in server_address;
    char buffer[1024] = {0};
    int port = 12345;

    // Create a UDP socket
    client_socket = socket(AF_INET, SOCK_DGRAM, 0);
    if (client_socket < 0) {
        perror("Error creating socket");
        exit(EXIT_FAILURE);
    }

    // Set up the server address
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(port);
    server_address.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Send a message to the server
    char message[] = "Hello, world";
    printf("Sending: %s\n", message);
      sendto(client_socket, message, strlen(message), 0, (struct sockaddr
```

```c
*)&server_address, sizeof(server_address));

    // Receive the server's response
    int bytes_received = recvfrom(client_socket, buffer, 1024, 0, NULL, NULL);
    if (bytes_received < 0) {
        perror("Error receiving message");
        exit(EXIT_FAILURE);
    }

    printf("Received: %s\n", buffer);

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

Output: Sender and Receiver

```
da/ass4/q2 via C v16.0.0-clang
❯ ./a.out
UDP Server listening on port 12345
Waiting to receive message...
Received 12 bytes from 127.0.0.1
Data: Hello, world
Sending acknowledgment
Waiting to receive message...
```

The key features of UDP:

- UDP is a connectionless protocol, which means that there is no need to establish a connection before sending data.
- The server creates a socket, binds it to a specific port, and listens for incoming messages.
- The client creates a socket and sends messages to the server's socket.
- The server and client can exchange data, but there is no guarantee that the data will be delivered or arrive in the correct order.
- The server and client do not need to handle the connection lifecycle, as there is no connection to maintain.

3. Difference between TCP and UDP

The main differences between TCP and UDP socket programming in C:

1. Connection-oriented vs. Connectionless:
   TCP is a connection-oriented protocol, which means that a connection is established and maintained until the application programs at each end have finished exchanging messages. UDP is a connectionless protocol, which means that there is no need to establish a connection before sending data.

2. Reliability and Ordering:
   TCP provides reliable and ordered delivery of data packets. UDP provides unreliable and unordered delivery of data packets.

3. Connection Lifecycle:
   In TCP, the server and client must handle the connection lifecycle, including opening, using, and closing the connection. In UDP, the server and client do not need to handle the connection lifecycle, as there is no connection to maintain.