Solution: TCP Client-Server Echo Program

Overview

This solution demonstrates how to create a simple TCP client-server application in C++ where the server echoes back any message received from the client.

Server Code

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[1024];

    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd < 0) {
        perror("Socket creation failed");
        return 1;
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(5001);

    if (bind(server_fd, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
        perror("Bind failed");
        close(server_fd);
        return 1;
    }

    if (listen(server_fd, 5) < 0) {
        perror("Listen failed");
        close(server_fd);
        return 1;
```

```
    }

    std::cout << "Server listening on port 5001..." << std::endl;

    client_fd = accept(server_fd, (struct sockaddr*)&client_addr,
&addr_len);
    if (client_fd < 0) {
        perror("Accept failed");
        close(server_fd);
        return 1;
    }

    memset(buffer, 0, sizeof(buffer));
    int n = recv(client_fd, buffer, sizeof(buffer), 0);
    if (n > 0) {
        std::cout << "Received: " << buffer << std::endl;
        send(client_fd, buffer, n, 0);
    }

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

Client Code

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[1024];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        return 1;
    }

    server_addr.sin_family = AF_INET;
```

```
    server_addr.sin_port = htons(5001);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(sockfd, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
        perror("Connection failed");
        close(sockfd);
        return 1;
    }

    const char* message = "Hello Server!";
    send(sockfd, message, strlen(message), 0);

    memset(buffer, 0, sizeof(buffer));
    int n = recv(sockfd, buffer, sizeof(buffer), 0);
    if (n > 0) {
        std::cout << "Echo from server: " << buffer << std::endl;
    }

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

Explanation

Server

- **socket()**: Creates a TCP socket.
- **bind()**: Binds the socket to port `5001` and all available IP addresses.
- **listen()**: Puts the server in listening mode, waiting for incoming connections.
- **accept()**: Accepts an incoming client connection.
- **recv()**: Receives data sent by the client.
- **send()**: Sends the same data back to the client.
- **close()**: Closes the client and server sockets.

Client

- **socket()**: Creates a TCP socket.
- **connect()**: Connects to the server at `127.0.0.1:5001`.
- **send()**: Sends a message to the server.
- **recv()**: Receives the echoed message from the server.
- **close()**: Closes the connection.

Summary

This exercise demonstrates how TCP provides a **reliable, connection-oriented communication channel** compared to the stateless nature of UDP. You can extend this exercise to handle multiple clients or implement error handling and connection timeouts.