Client–Server Communication (TCP) in C++ — Step-by-Step Guide

NAME : UBAID KUNDLIK ASSIGNMENT : 3 PRN :12310428. ROLL NO : 52. DIV : CS-AIML-B

PROBLEM STATEMENT
3a) Write the client server programs using TCP Berkeley socket primitives for wired /wireless network for following
a. To say Hello to Each other

b. File transfer

This document explains how to build, run, and test simple TCP client–server programs in C++ on macOS using VS Code. It covers:

- Hello message (basic client/server)
- Peer-to-peer file transfer (server sends file to client)

Prerequisites

1. **macOS** (you already have).
2. **Xcode Command Line Tools** (compiler):

```
xcode-select --install
```

1.

2. **Visual Studio Code** (editor) and the **C/C++** extension by Microsoft.
3. (Optional) code command installed in PATH so you can open folders from Terminal.

Project folder & files

Create the project folder and open it in VS Code:

```
mkdir -p ~/projects/networking
cd ~/projects/networking
code .
```

Files you will create:

```
networking/
  ├─ hello_server.cpp   # simple hello message server
  ├─ hello_client.cpp   # simple hello message client
  ├─ file_server.cpp    # send file to client
  ├─ file_client.cpp    # request file and save
  ├─ server.cpp         # threaded multi-client server (chat)
  └─ client.cpp         # client for chat
```

---

1) Hello Message (basic TCP)

hello_server.cpp

```
#include <iostream>
#include <unistd.h>
#include <string.h>
#include <netinet/in.h>

#define PORT 8080

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
```

```
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    const char *hello = "Hello from server";

    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
&opt, sizeof(opt));

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

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

    std::cout << "Server listening on port " << PORT << "...\n";
    new_socket = accept(server_fd, (struct sockaddr*)&address, (socklen_t
*)&addrlen);

    read(new_socket, buffer, 1024);
    std::cout << "Client: " << buffer << "\n";

    send(new_socket, hello, strlen(hello), 0);
    std::cout << "Hello message sent\n";

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

hello_client.cpp

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

```
#define PORT 8080

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

    sock = socket(AF_INET, SOCK_STREAM, 0);

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    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);
    std::cout << "Hello message sent\n";

    read(sock, buffer, 1024);
    std::cout << "Server: " << buffer << "\n";

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

Compile & run

- Terminal 1 (server):

```
g++ hello_server.cpp -o hello_server && ./hello_server
```

-

- Terminal 2 (client):

```
g++ hello_client.cpp -o hello_client && ./hello_client
```

Expected output: server prints client message and client prints server message.

---

2) Peer-to-peer File Transfer (server sends file)

file_server.cpp

```
#include <iostream>
#include <fstream>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <cstring>

#define PORT 9090

int main() {
    int server_fd = socket(AF_INET, SOCK_STREAM, 0);
    int opt = 1;
    setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

    sockaddr_in address{};
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

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

    std::cout << "Waiting for file request on port " << PORT << std::endl;
    int new_socket = accept(server_fd, (sockaddr*)&address, (socklen_t*) &
(socklen_t){sizeof(address)});

    char buffer[1024] = {0};
    ssize_t r = read(new_socket, buffer, sizeof(buffer)-1);
    buffer[r] = '\0';
    std::string filename = buffer;

    std::ifstream file(filename, std::ios::binary);
    if (!file) {
```

```
        const char* msg = "ERROR: file not found";
        send(new_socket, msg, strlen(msg), 0);
        close(new_socket);
        close(server_fd);
        return 1;
    }

    while (!file.eof()) {
        file.read(buffer, sizeof(buffer));
        std::streamsize got = file.gcount();
        if (got > 0) send(new_socket, buffer, (size_t)got, 0);
    }

    std::cout << "File sent successfully." << std::endl;
    file.close();
    close(new_socket);
    close(server_fd);
    return 0;
}
```

file_client.cpp

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

#define PORT 9090

int main() {
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in serv_addr{};
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);
```

```
    connect(sock, (sockaddr*)&serv_addr, sizeof(serv_addr));

    std::string filename = "sample.txt";
    send(sock, filename.c_str(), filename.size(), 0);

    std::ofstream outfile("received_" + filename, std::ios::binary);
    char buffer[1024];
    ssize_t bytesRead;
    while ((bytesRead = read(sock, buffer, sizeof(buffer))) > 0) {
        outfile.write(buffer, bytesRead);
    }

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

How to run

1. Place sample.txt in the server folder.
2. Terminal 1:

```
g++ file_server.cpp -o file_server && ./file_server
```

1.
2. Terminal 2:

```
g++ file_client.cpp -o file_client && ./file_client
```

1.
2. After completion, received_sample.txt will be created on the client side.

---

5. Conclusion

- TCP sockets provide reliable, connection-oriented communication between client and server.

- This project demonstrated two essential TCP applications: simple message exchange and file transfer.
- These programs work on wired/wireless networks as long as IP connectivity exists.
