

Distributed System Labwork 1



Group 1 - ICT

University of Science and Technology of Hanoi

January, 2022

Contents

1	Introduction	2
1.1	Overview	2
1.2	Protocol	2
1.3	System organization	3
1.4	Implementation	3
1.5	Contribution	7

1 Introduction

1.1 Overview

In this labwork, we try to build a file transfer over TCP/IP in CLI, based on the provided chat system.

1.2 Protocol

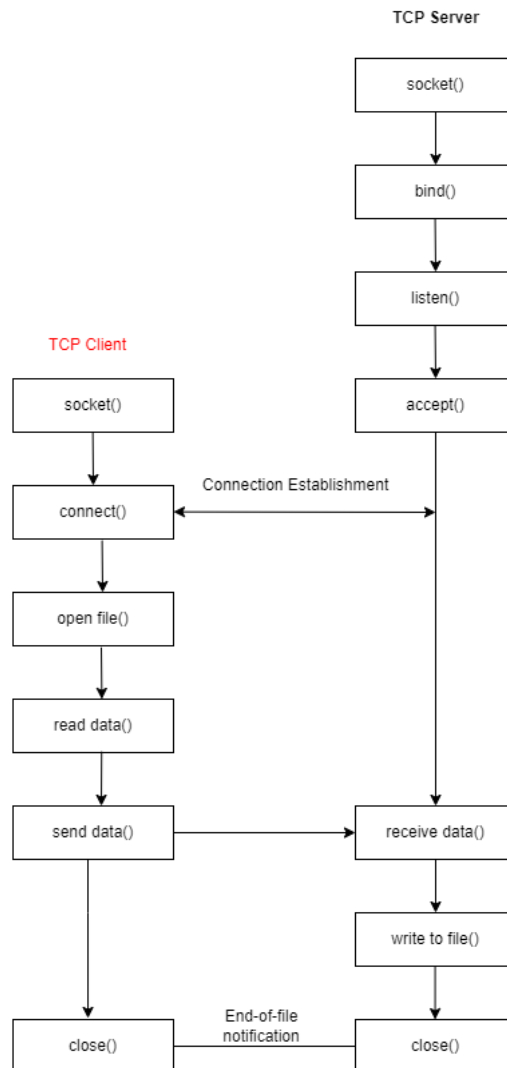


Figure 1: Protocol diagram

1.3 System organization

The server creates a specific port, for eg: 9819. From the client CLI, it takes 2 arguments, IP address and the port to connect to the server. One client connects to one server only. The client send data through a character array called buffer. The buffer has a maximum length of 255. After receiving the buffer, the server will writes it to the file. The client will notice the server there is nothing left after reaching end-of-file. Afterwards, both server and client will close.

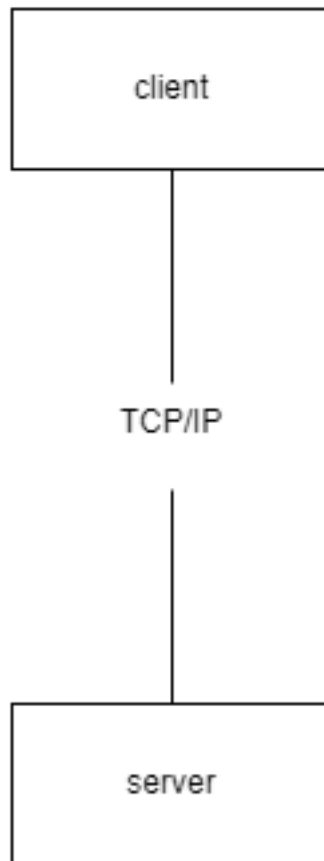


Figure 2: System organization

1.4 Implementation

We have implemented the client side:

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

```

#include <string.h>
#include <netdb.h>
void error(const char* msg) {
    //print system error message
    perror(msg);
    exit(1);
}

void send_file(FILE* f, int sockfd) {
    char buffer[1024] = { 0 };
    while (fgets(buffer, sizeof(buffer), f) != NULL) {
        int i = send(sockfd, buffer, sizeof(buffer), 0);
        if (i == -1) {
            error("Error in sending data");
        }
        memset(&buffer, 0, sizeof(buffer));
    }
}

int main(int argc, char* argv[]) {
    int so;
    char s[100];
    struct sockaddr_in ad;

    socklen_t ad_length = sizeof(ad);
    struct hostent* hep;

    // create socket
    int serv = socket(AF_INET, SOCK_STREAM, 0);

    // init address
    hep = gethostbyname(argv[1]);
    memset(&ad, 0, sizeof(ad));
    ad.sin_family = AF_INET;
    ad.sin_addr = *(struct in_addr*)hep->h_addr_list[0];
    ad.sin_port = htons(12345);

    // connect to server
    connect(serv, (struct sockaddr*)&ad, ad_length);

    memset(&s, 0, 100);
    FILE* f;
    f = fopen("send.txt", "r");
    if (f == NULL) {
        error("Error in reading file.");
    }
}

```

```

    }
    else {
        printf("Reading file successfully..\n");
    }
    rewind(f);
    send_file(f, serv);
    printf("The file has been transfered successfully...\n");
    printf("Close...\n");
    close(serv);
    return 0;
}

```

We have implemented the server side:

```

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

void error(const char* msg) {
    //print system error message
    perror(msg);
    exit(1);
}

void write_file(FILE* fp, int sockfd) {
    char buffer[1024];

    while (1) {
        int n = recv(sockfd, buffer, sizeof(buffer), 0);
        if (n <= 0) {
            break;
        }
        fprintf(fp, "%s", buffer);
        memset(&buffer, 0, sizeof(buffer));
    }
    return;
}

int main() {
    int ss, cli, pid;
    struct sockaddr_in ad;
    char s[100];
    socklen_t ad_length = sizeof(ad);

```

```

FILE* fp;
// create the socket
ss = socket(AF_INET, SOCK_STREAM, 0);

// bind the socket to port 12345
memset(&ad, 0, sizeof(ad));
ad.sin_family = AF_INET;
ad.sin_addr.s_addr = INADDR_ANY;
ad.sin_port = htons(12345);
bind(ss, (struct sockaddr*)&ad, ad_length);

// then listen
listen(ss, 0);

while (1) {
    // an incoming connection
    cli = accept(ss, (struct sockaddr*)&ad, &ad_length);

    pid = fork();
    if (pid == 0) {
        printf("client connected\n");

        fp = fopen("received.txt", "w");
        if (fp == NULL) {
            error("Error in reading file.");
        }
        else {
            printf("Reading file successfully..\n");
        }
        write_file(fp, cli);
        printf("Received file.");

        return 0;
    }
    else {
        // continue the loop to accept more clients
        continue;
    }
}

// disconnect
close(cli);
close(ss);
}

```

1.5 Contribution

Member	Contribution
Nguyen Xuan Tung	Client code
Nguyen Quang Anh	Server code
Lu Khanh Huyen	Design Protocol
Tran Hong Quan	Design Architecture
Vu Duc Chinh	Report

Table 1: Contribution Table