File transfer ove TCP/IP in CLI

Distributed System Group 4

5/3/2020

Contents

1 Introduction

If we are creating a connection between client and server using TCP then it has few functionality like, TCP is suited for applications that require high reliability, and transmission time is relatively less critical. It is used by other protocols like HTTP, HTTPs, FTP, SMTP, Telnet. TCP rearranges data packets in the order specified. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent. TCP does Flow Control and requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control. It also does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.

2 How we design out Protocol

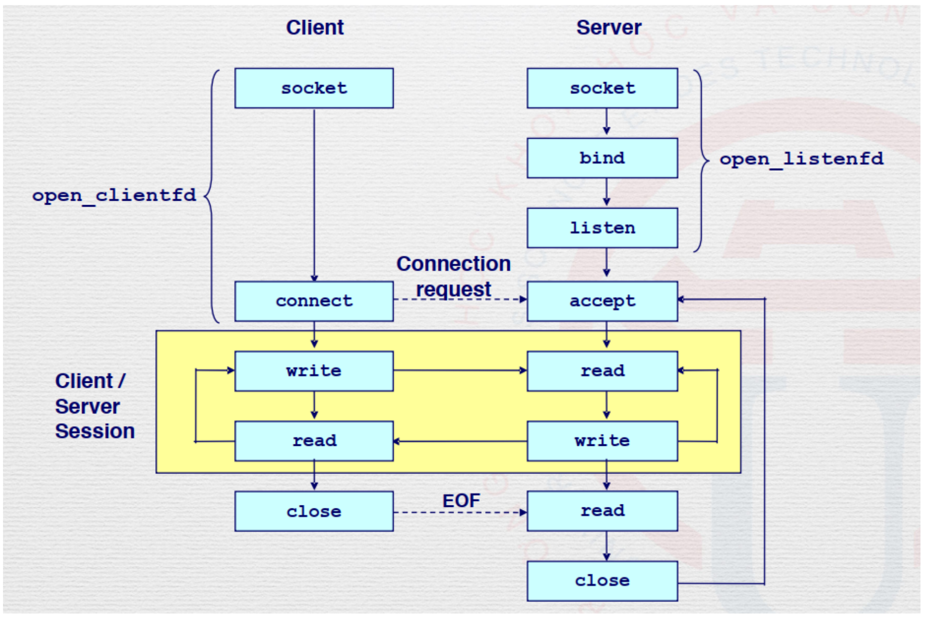

Figure 1: Protocol

3 How we organized our System

3.1 Open Session

- The server socket will be bound to port 8080.
- The server socket then listening to y message / data received.

3.2 End open listen of server

- After getting the server socket to listening, the client socket will try to connect to the server.

3.3 End open client socket

- In Client/Server session, both client and server sending eachother messages.

4 Code

4.1 Client

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

 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);
     if (serv == -1) {
         printf("socket creation failed...\n");
         exit(0);
     }
     else
         printf("Socket successfully created..\n");
     // 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(8080);

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

     while (1) {
         // after connected, it's client turn to chat

         // send some data to server
         printf("client>");
         scanf("%s", s);
```

```c
            write(serv, s, strlen(s) + 1);

            // then it's server turn
            read(serv, s, sizeof(s));

            printf("server says: %s\n", s);
        }
}
```

4.2 Server

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

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

        // create the socket
        ss = socket(AF_INET, SOCK_STREAM, 0);
        if (ss == -1) {
            printf("socket creation failed...\n");
            exit(0);
        }
        else
            printf("Socket successfully created..\n");
        // 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(8080);
        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);

            int pid = fork();
            if (pid == 0) {
                // I'm the son, I'll serve this client
                printf("client 1 connected\n");
                while (1) {
                    // it's client turn to chat, I wait and read message from client
                    read(cli, s, sizeof(s));
                    printf("client 1 says: %s\n",s);

                    // now it's my (server) turn
                    printf("client 1>%s", s);
                    scanf("%s", s);
                    write(cli, s, strlen(s) + 1);
                }
                return 0;
            }
            else {
                waitpid(pid, NULL, 0);
                int pid1 = fork();
                if (pid1 == 0) {
                    // I'm the son, I'll serve this client
                    printf("client 1 connected\n");
                    while (1) {
                        // it's client turn to chat, I wait and read message from client
                        read(cli, s, sizeof(s));
                        printf("client 1 says: %s\n",s);

                        // now it's my (server) turn
                        printf("client 1>%s", s);
                        scanf("%s", s);
                        write(cli, s, strlen(s) + 1);
                    }
                    return 0;
                }
                else
                {
                    //this is after work parent
                }

            }
        }
        // disconnect
        close(cli);

    }
```

5 Who does what?

Bui Quang Huy : Rewrite the code from Dr.Son source code and execute it.
Nguyen Viet Dung and Nguyen Quang Trung : Design and write the report.
Do Minh Hoang: Research about the protocol.