Report

File transfer ove TCP/IP in CLI

Distributed System

Group 4

by Nguyễn Lê Thanh Hà, Trần Thanh Long, Nguyễn Minh Hoàng, Trần Trung Nhật, Đỗ Đức Mạnh

February 26, 2021

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

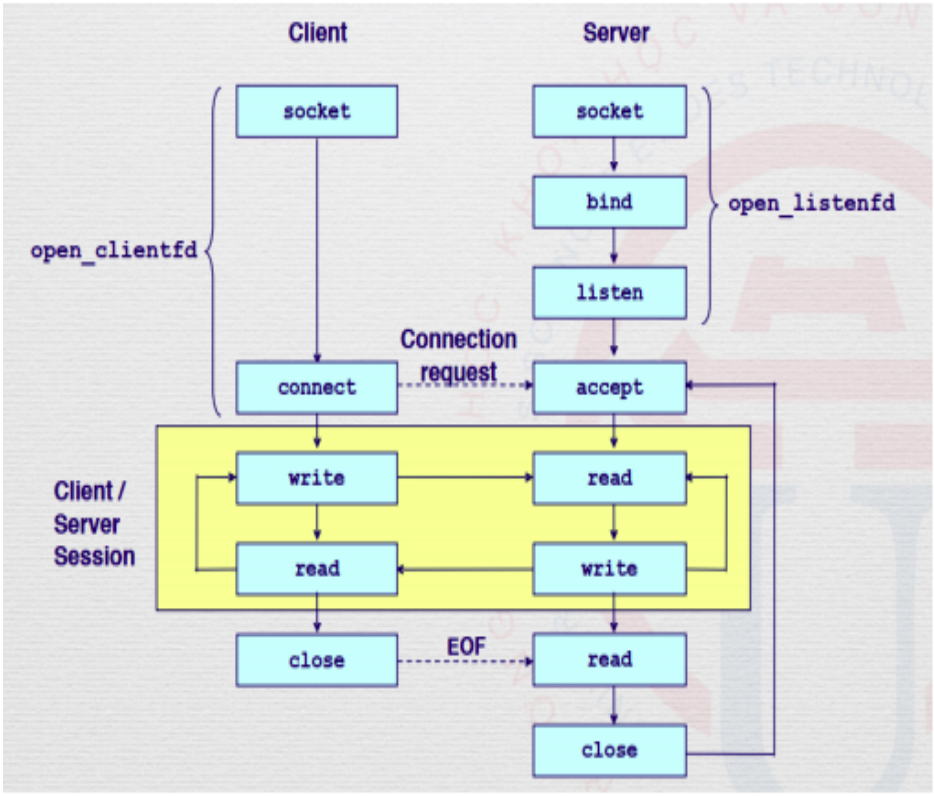

Figure 1: Protocol

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

```
#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);
            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

```
#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);
    }
```

```
21          else
22              printf("Socket successfully created..\n");
23          // bind the socket to port 12345
24          memset(&ad, 0, sizeof(ad));
25          ad.sin_family = AF_INET;
26          ad.sin_addr.s_addr = INADDR_ANY;
27          ad.sin_port = htons(8080);
28          bind(ss, (struct sockaddr *)&ad, ad_length);
29
30          // then listen
31          listen(ss, 0);
32
33          while (1) {
34              // an incoming connection
35              cli = accept(ss, (struct sockaddr *)&ad, &ad_length)
    ;
36
37              int pid = fork();
38              if (pid == 0) {
39                  // I'm the son, I'll serve this client
40                  printf("client 1 connected\n");
41                  while (1) {
42                      // it's client turn to chat, I wait and read
     message from client
43                      read(cli, s, sizeof(s));
44                      printf("client 1 says: %s\n",s);
45
46                      // now it's my (server) turn
47                      printf("client 1>%s", s);
48                      scanf("%s", s);
49                      write(cli, s, strlen(s) + 1);
50                  }
51                  return 0;
52              }
53              else {
54                  waitpid(pid, NULL, 0);
55                  int pid1 = fork();
56                  if (pid1 == 0) {
57                      // I'm the son, I'll serve this client
58                      printf("client 1 connected\n");
59                      while (1) {
60                          // it's client turn to chat, I wait and
    read message from client
61                          read(cli, s, sizeof(s));
62                          printf("client 1 says: %s\n",s);
63
64                          // now it's my (server) turn
65                          printf("client 1>%s", s);
66                          scanf("%s", s);
```

```
67                          write(cli, s, strlen(s) + 1);
68                      }
69                      return 0;
70                  }
71                  else
72                  {
73                      //this is after work parent
74                  }
75
76              }
77          }
78          // disconnect
79          close(cli);
80
81      }
```