

File transfer ove TCP/IP in CLI

Distributed System Group 4

5/3/2020

Contents

1	Introduction	1
2	How we design out Protocol	1
3	How we organized our System	1
3.1	Open Session	1
3.2	End open listen of server	2
3.3	End open client socket	2
4	Code	2
4.1	Client	2
4.2	Server	3
5	Who does what?	4

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

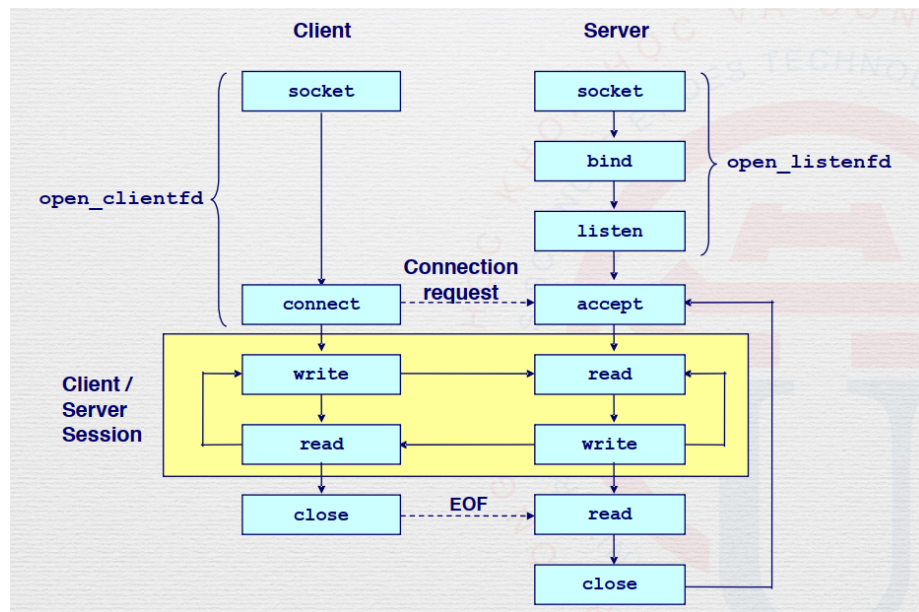


Figure 1: Protocol

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

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netdb.h>
8
9  int main(int argc, char* argv[]) {
10     int so;
11     char s[100];
12     struct sockaddr_in ad;
13
14     socklen_t ad_length = sizeof(ad);
15     struct hostent *hep;
16
17     // create socket
18     int serv = socket(AF_INET, SOCK_STREAM, 0);
19     if (serv == -1) {
20         printf("socket creation failed...\n");
21         exit(0);
22     }
23     else
24         printf("Socket successfully created..\n");
25     // init address
26     hep = gethostbyname(argv[1]);
27     memset(&ad, 0, sizeof(ad));
28     ad.sin_family = AF_INET;
29     ad.sin_addr = *((struct in_addr *)hep->h_addr_list[0]);
30     ad.sin_port = htons(8080);
31
32     // connect to server
33     connect(serv, (struct sockaddr *)&ad, ad_length);
34
35     while (1) {
36         // after connected, it's client turn to chat
37
38         // send some data to server
39         printf("client>");
40         scanf("%s", s);
```

```

41         write(serv, s, strlen(s) + 1);
42
43         // then it's server turn
44         read(serv, s, sizeof(s));
45
46         printf("server says: %s\n", s);
47     }
48 }

```

4.2 Server

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netdb.h>
8
9  int main() {
10     int ss, cli;
11     struct sockaddr_in ad;
12     char s[100];
13     socklen_t ad_length = sizeof(ad);
14
15     // create the socket
16     ss = socket(AF_INET, SOCK_STREAM, 0);
17     if (ss == -1) {
18         printf("socket creation failed...\n");
19         exit(0);
20     }
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
43                 // message from client
44                 read(cli, s, sizeof(s));
45                 printf("client 1 says: %s\n", s);

```

```

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
61             read message from client
62             read(cli, s, sizeof(s));
63             printf("client 1 says: %s\n", s);
64
65             // now it's my (server) turn
66             printf("client 1>%s", s);
67             scanf("%s", s);
68             write(cli, s, strlen(s) + 1);
69         }
70         return 0;
71     }
72     {
73         //this is after work parent
74     }
75 }
76 }
77 // disconnect
78 close(cli);
79
80
81 }

```

5 Who does what?

Everybody do everything :(