Network Protocols

Dr. Ayman A. Abdel-Hamid

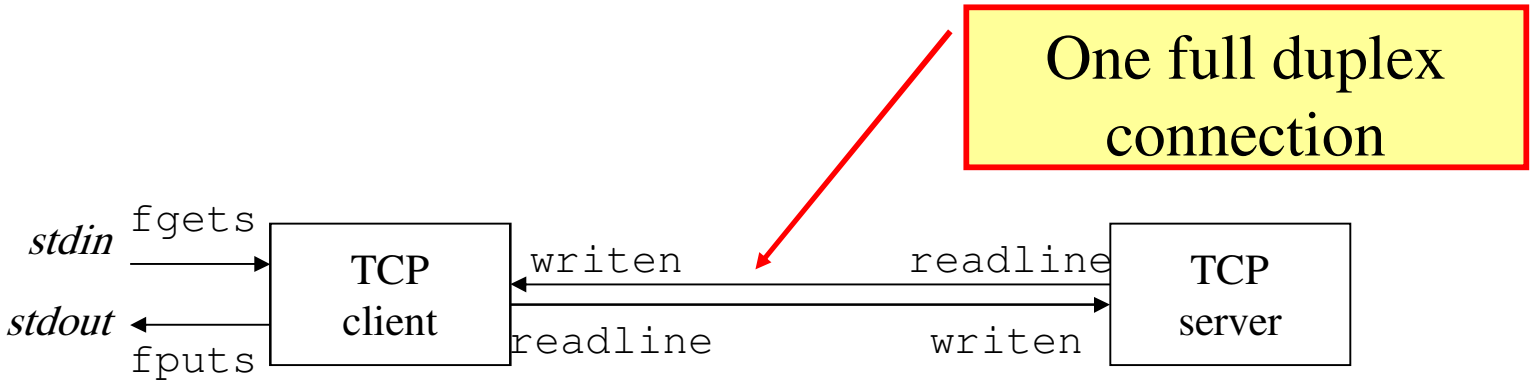
College of Computing and Information Technology
Arab Academy for Science & Technology and
Maritime Transport

TCP Client/Server Example

Outline

- TCP Client Server Example (Chapter 5)
  - Introduction
  - TCP Echo Server
  - TCP Echo Client

Introduction



•The Client reads a line of text from its standard input and writes the line to the server.
•The server reads the line from its network input and echoes the line back to the client.
•The client reads the echoed line and prints it on its standard output.

TCP Echo Server: main

tcpcliserv/tcpserv01.c

```
 1 #include    "unp.h"
 2 int
 3 main(int argc, char **argv)
 4 {
 5     int     listenfd, connfd;
 6     pid_t   childpid;
 7     socklen_t clilen;
 8     struct sockaddr_in cliaddr, servaddr;
 9     listenfd = Socket(AF_INET, SOCK_STREAM, 0);
10     bzero(&servaddr, sizeof(servaddr));
11     servaddr.sin_family = AF_INET;
12     servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
13     servaddr.sin_port = htons(SERV_PORT);
14     Bind(listenfd, (SA *) &servaddr, sizeof(servaddr));
15     Listen(listenfd, LISTENQ);
16     for ( ; ; ) {
17         clilen = sizeof(cliaddr);
18         connfd = Accept(listenfd, (SA *) &cliaddr, &clilen);
19         if ( (childpid = Fork()) == 0) {     /* child process */
20             Close(listenfd);    /* close listening socket */
21             str_echo(connfd);   /* process the request */
22             exit(0);
23         }
24         Close(connfd);          /* parent closes connected socket */
25     }
26 }
```

SERVPORT defined to be 9877 in unp.h

tcpcliserv/tcpserv01.c

Figure 5.2 TCP echo server.

TCP Echo Server: **str_echo** function

```
/* code in lib/str_echo.c */
#include "unp.h"

void
str_echo(int sockfd)
{
        ssize_t         n;
        char            buf[MAXLINE];

again:
        while ( (n = read(sockfd, buf, MAXLINE)) > 0)
                Writen(sockfd, buf, n);

        if (n < 0 && errno == EINTR)
                goto again;
        else if (n < 0)
                err_sys("str_echo: read error");
}
```

TCP Echo Client: **main**

```
#include "unp.h"
int
main(int argc, char **argv)
{
        int                     sockfd;
        struct sockaddr_in      servaddr;

        if (argc != 2)
                err_quit("usage: tcpcli <IPaddress>");

        sockfd = Socket(AF_INET, SOCK_STREAM, 0);
        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(SERV_PORT);
        Inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

        Connect(sockfd, (SA *) &servaddr, sizeof(servaddr));
        str_cli(stdin, sockfd);                 /* do it all */
        exit(0);
}
```

TCP Echo Client: **str_cli** function

```
// code in lib/str_cli.c
#include "unp.h"
void
str_cli(FILE *fp, int sockfd)
{
        char    sendline[MAXLINE], recvline[MAXLINE];

        while (Fgets(sendline, MAXLINE, fp) != NULL) {

                Writen(sockfd, sendline, strlen(sendline));

                if (Readline(sockfd, recvline, MAXLINE) == 0)
                        err_quit("str_cli: server terminated prematurely");

                Fputs(recvline, stdout);
        }
}
```

Normal Startup

- Normal Startup
  - Check TCP state using *netstat –a*
  - Check processes using *ps –ef*
  - Check with 1 client and 2 clients
  - What do you conclude?

Normal Termination

- Normal Termination (see page 128)
  - When child process (server child) terminates, the parent process receives a SIGCHLD signal (software interrupt)
  - The code does not handle the signal, hence default action is to be ignored
  - Child process enters *zombie state*
  - Check status of processes using *ps –ef*
  - Use *netstat* for TCP states
  - Try the following scenarios
    - ✓Client terminates first
    - ✓Server terminates first