Assignment - 1

ELL - 785 Computer Communication Network

Srijan Upadhyay 2019JTM2168
Utkarsh Badal 2019JTM2679
2019-21

A report presented for the assignment on
Socket Programming

Bharti School Of
Telecommunication Technology and
Management

IIT Delhi
India

List of Figures

Objective

Writing a client-server program using C/C++, where clients (instructor or students of a class) access the server (storing students' marks in 5 subjects out of 100) for information about marks in a semester examination.

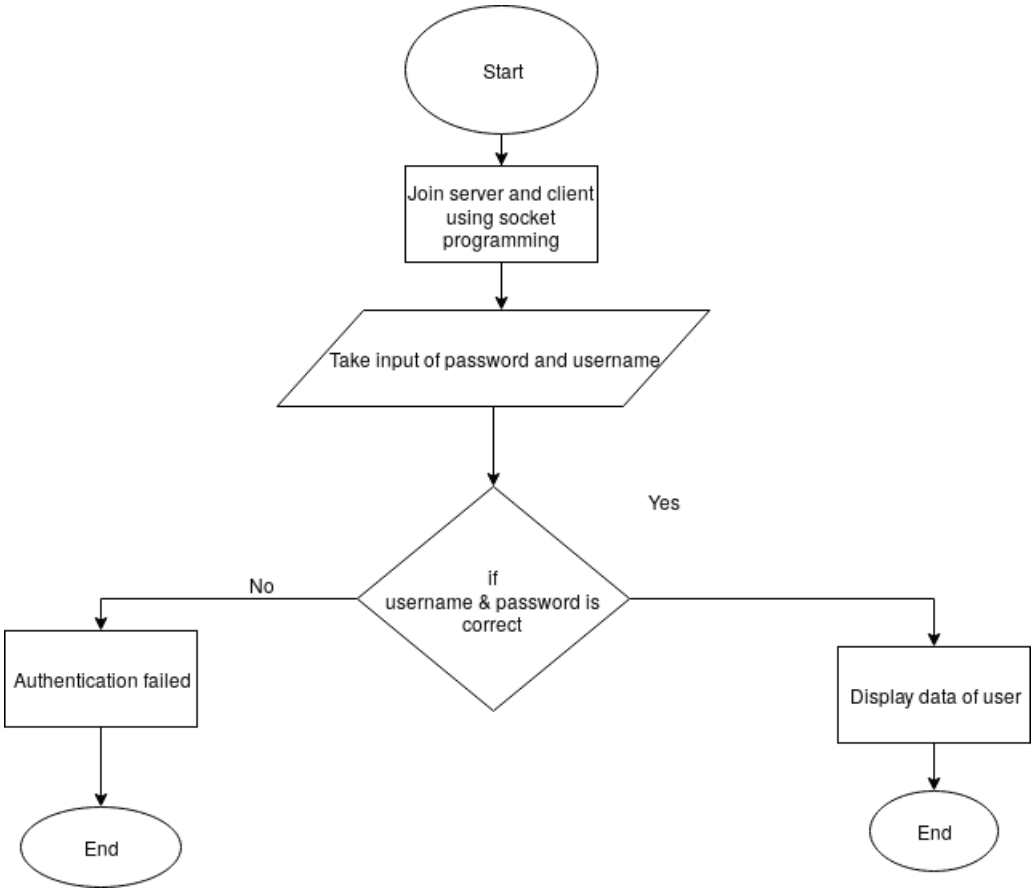
Following tasks have to be performed

1. Client will connect to server and logon through username and password pre-stored on server. Server will refuse connection without proper authentication.
2. If client is logged on using 'instructor' as username, it will have access to marks of all the students in the class.
3. If client is any other user '¡username¿' (i.e. client is student) it will have access to his/her marks only.
4. Client (student) should be able to get information about:
   (a) His/her marks in each subject
   (b) Aggregate percentage
   (c) Subjects with maximum and minimum marks
5. Client (instructor) should be able to get information about:
   (a) Marks (individual and aggregate percentage) of each student
   (b) Class average
   (c) Number of students failed (passing percentage 33.33(d) Name of best and worst performing students
   (e) BONUS Question: Instructor can update the marks of any student if he/she finds a bug (or need for correction). Therefore, create a menu having option 'Update' for 'Instructor' login to update marks of a particular student in a subject.
6. Create '$student_marks'$ file that contains marks of each student and is accessed by server for responding to client queries.
7. Create '$user_pass'$ file to hold data for usernames and passwords (with at least 20 users). This file is accessed by server for authentication
8. Create menu to select required information from client, either at client side or server side.
9. Exception handling is a must.
10. Using Wireshark, analyze packet size and frame size in different TCP/IP layers. Also trace the communication path between client and server machines, and find the number of hops used for communications. Comment on all the observations.

*Algorithm

- first i made a program for client and server to connect them together.
- made a files of user name and password ,students marksheet.

Flowchart



Flowchart

Screenshots

```
srijanupadhyaya@upadhyay:~/ccna$ gcc newserver.c -o test
srijanupadhyaya@upadhyay:~/ccna$ ./test
^[[A^[[AUsername:user4
Password:pwd4
User:user1
```

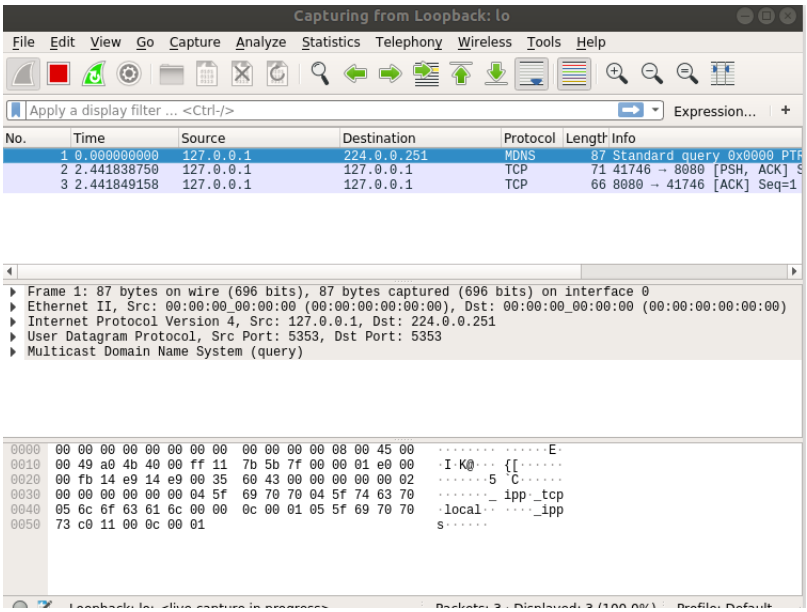
Server 1

```
srijanupadhyaya@upadhyay:~/ccna$ ./test
Username:instructor
Password:ins1
user1
```

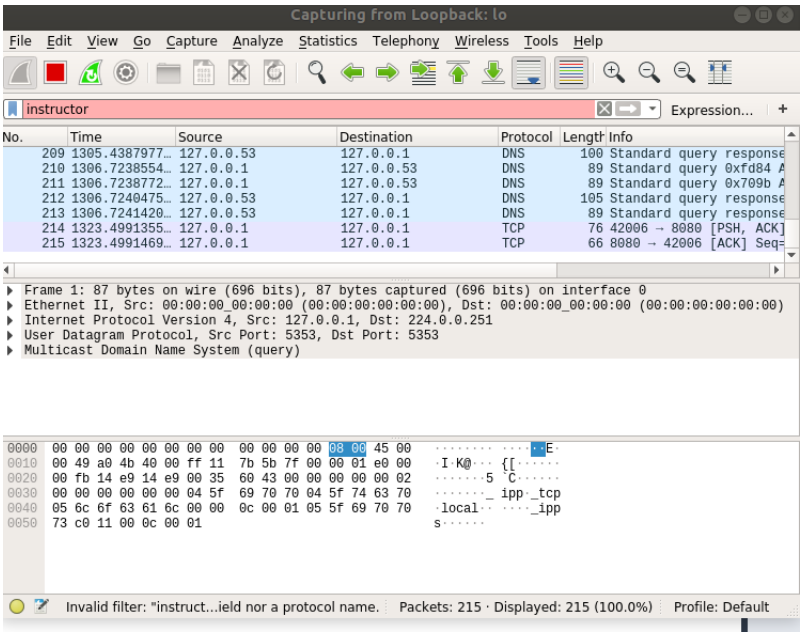
Server 2

```
srijanupadhyaya@upadhyay:~/ccna$ gcc newclient.c -o tes
srijanupadhyaya@upadhyay:~/ccna$ gcc newclient.c -o tes
srijanupadhyaya@upadhyay:~/ccna$ ./tes
Enter the name:instructor
Name Sent
Enter the password:ins1
Password sent
user1
physics 90 chemistry 55 maths 89 signals 76 neurology 67 percentage 75.40 max physics,min chemistry
user2
physics 79 chemistry 78 maths 98 signals 67 neurology 67 percentage 77.80 max maths,min signals,neurology
user3
physics 79 chemistry 89 maths 70 signals 57 neurology 67 percentage 72.40 max chemistry,min signals
user4
physics 78 chemistry 90 maths 76 signals 93 neurology 67 percentage 80.80 max signals,min neurology
user5
physics 84 chemistry 55 maths 87 signals 83 neurology 67 percentage 75.20 max maths,min chemistry
user6
physics 98 chemistry 55 maths 78 signals 99 neurology 67 percentage 79.40 max signals,min chemistry
user7
physics 67 chemistry 55 maths 69 signals 69 neurology 67 percentage 65.40 max chemistry,maths min chemistry
user8
physics 58 chemistry 55 maths 89 signals 79 neurology 67 percentage 69.60 max maths,min chemistry
user9
physics 87 chemistry 55 maths 65 signals 79 neurology 67 percentage 70.60 max physics,min chemistry
user10
physics 89 chemistry 55 maths 74 signals 87 neurology 74 percentage 75.80 max signals,min chemistry
user11
physics 99 chemistry 55 maths 93 signals 87 neurology 64 percentage 79.60 max physics,min chemistry
user12
physics 98 chemistry 55 maths 83 signals 44 neurology 68 percentage 69.60 max physics,min physics
user13
physics 97 chemistry 55 maths 98 signals 78 neurology 66 percentage 78.80 max maths,min chemistry
user14
physics 96 chemistry 55 maths 98 signals 97 neurology 77 percentage 84.60  max maths,min chemistry
user15
physics 95 chemistry 55 maths 78 signals 99 neurology 98 percentage 85.00 max signals,min chemistry
user16
physics 94 chemistry 55 maths 79 signals 69 neurology 78 percentage 75.00 max physics,min chemistry
user17
physics 93 chemistry 55 maths 97 signals 77 neurology 76 percentage 79.60 max maths,min chemistry
user18
physics 92 chemistry 55 maths 99 signals 90 neurology 98 percentage 86.80 max maths,min chemistry
user19
physics 91 chemistry 55 maths 100 signals 79 neurology 56 percentage 76.20 max maths,min chemistry
user20
physics 90 chemistry 55 maths 84 signals 69 neurology 68 percentage 73.20 max physics,min chemistry
class average:55.5 ,number of students failed :00,user 11 is best performer and user 19 is worst performing stude
nts.
```

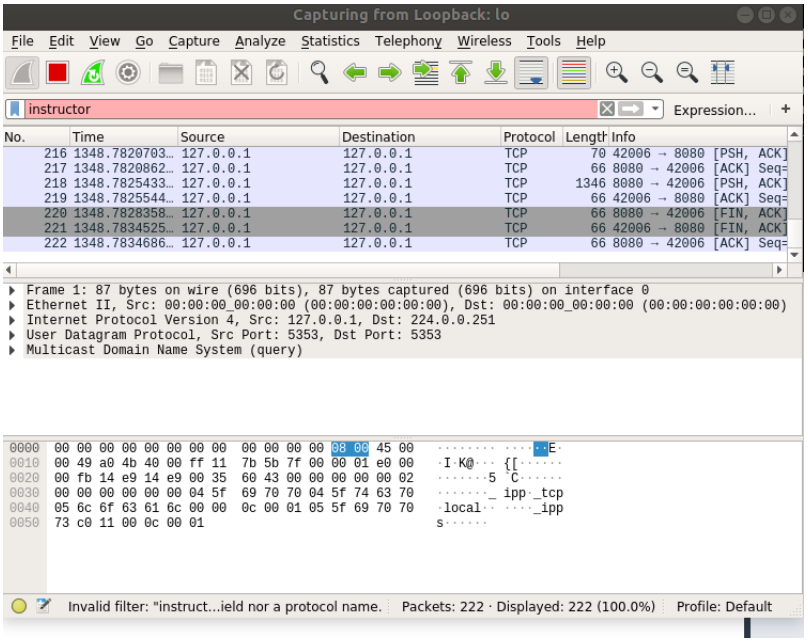
Client 2



Wireshark screenshot 1



Wireshark screenshot 2



Wireshark screenshot 3

Appendix

Client code

```c
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

int main(int argc, char const *argv[])
{
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char *hello = "Hello from client";
    char buffer[1000000] = {0};
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(
    serv_addr)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    char arr[20];
    printf("Enter the name:");
    scanf("%s",arr);
    send(sock , arr , strlen(arr) , 0 );
    printf("Name Sent\n");
    printf("Enter the password:");
    scanf("%s",arr);
    send(sock,arr,strlen(arr),0);
    printf("Password sent\n");
    valread = read( sock , buffer, 1000000);
    printf("%s\n",buffer );
    valread=read(sock,buffer,1024);

    printf("%s\n",buffer);
    return 0;
}
```

server code

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
```

```
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char passwd[1024]={0};
    char result[1000];
    char *hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR |
    SO_REUSEPORT,
                                                  &opt, sizeof(opt)
    ))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address,
                                 sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address
    ,
                        (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    valread = read(new_socket,buffer,1024);
    valread = read(new_socket,passwd,1024);
    printf("Username:%s\n",buffer);
    printf("Password:%s\n",passwd);
FILE *fp;
char aun[20],apw[20];
char *sd="Authenticated Successfully";
fp = fopen("user_pass.txt","r+");
```

```
63 if(fp==0)
64 {printf("can not open file");
65 }
66 while(fscanf(fp,"%s %s",aun,apw)!=EOF)
67 if(strcmp(aun,buffer)==0&&strcmp(apw,passwd)==0)
68 {
69 char * ins="instructor";
70 if(strcmp(ins,aun)==0){
71     char arr[100000];
72     arr[0]='\0';
73     FILE *fp3;
74     fp3 = fopen("student_marks.txt","r+");
75     char temp2[10000];
76     while(fgets(temp2, 10000, (FILE*)fp3)!=NULL){
77         strcat(arr,temp2);
78         temp2[0]='\0';
79
80     }
81     int sos=strlen(arr);
82     arr[sos]='\0';
83     printf("%s\n",arr);
84
85      send(new_socket , arr , strlen(arr) , 0 );
86      exit(0);
87 }
88 send(new_socket,sd ,strlen(sd),0);
89 FILE *fp2;
90 fp2 = fopen("student_marks.txt","r+");
91 if(fp2==0)
92 {printf("can not open file");
93 exit(0);
94 }
95 char curuser[20];
96 size_t len = 0;
97 while(fgets(curuser, 20, (FILE*)fp2)!=NULL)
98 {
99 printf("User:%s\n",curuser);
100 int i;
101     for(i=0;i<20;i++)
102     if(curuser[i]=='\n')
103     {curuser[i]='\0';
104     break;
105     }
106     if(strcmp(curuser,aun)==0)
107     { //printf("Yaha aa gye");
108         fgets(result, 1000, (FILE*)fp2);
109         send(new_socket , result , strlen(result) , 0 );
110         printf("%s",result);
111         exit(0);
112     }
113     fgets(result, 1000, (FILE*)fp2);
114
115     printf("%s",result);
116 }
117 }
118     send(new_socket , hello , strlen(hello) , 0 );
119     //printf("Hello message sent\n");
120     return 0;
121 }
```

References

[1] *The basics of socket programming.*

[2] *Cprogramming.* https://www.cprogramming.com/tutorial/c-tutorial.html.

[3] *Geeka and geeks.*

[4] H Kopka and PW Daly. A guide to {$LaTeX$}− − *document*. 1995.