Assignment 1

ELL785 Computer Communication Network

Sudhanshu Chaudhary
2019JTM2207

Santosh Kumar
2019JTM2088

Akanksha Jaiswal
2019EEZ8333

2019-2021

A report presented for the assignment on
Network Programming Using Internet Sockets

Bharti School Of
Telecommunication Technology and Management
IIT Delhi
India
August 16, 2019

Contents

List of Figures

1 Problem Statement-1

1.1 Problem Statement

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 percentage) in each subject
   (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.

1.2 Algorithm and Implementation on Server Side

1. Socket creation:
2. Set socket port:
3. Bind:
4. Listen:
5. User name and Password validation Check:
6. Accept:
7. Applied Delay:
8. File Handling:
9. Read data:
10. Send data:

1.3 Algorithm and Implementation on Client Side

1. Socket connection:
2. Connect:
3. User name and Password input:
4. Validating user name and password from server:
5. Read data:
6. Send data:
7. Converting string to integer and calculate percentage and class average:
8. Find the the maximum and minimum marks of each student:

1.4 Program Structure

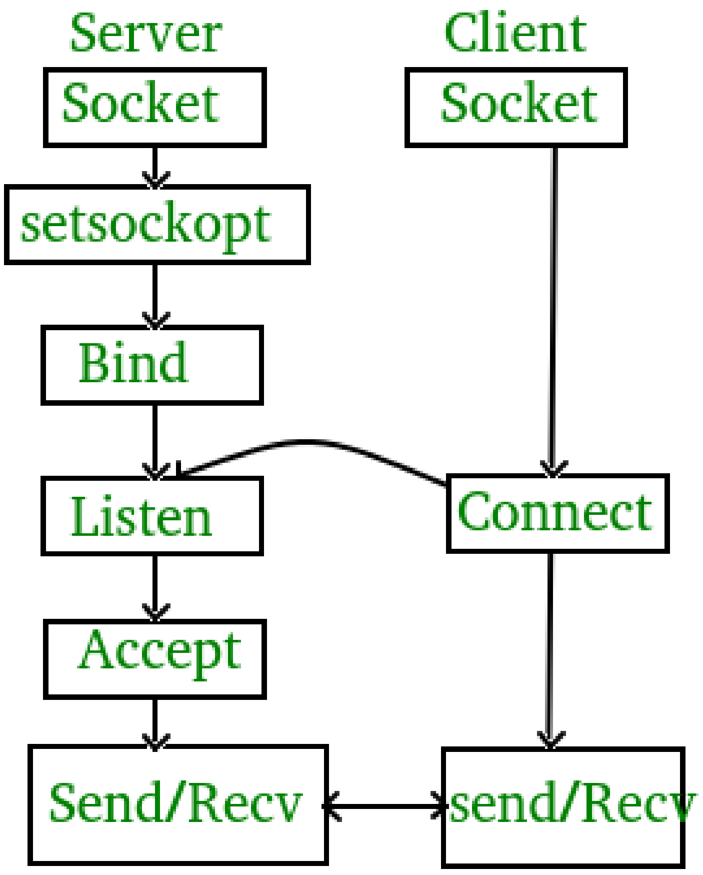


Figure 1: State diagram for server and client model

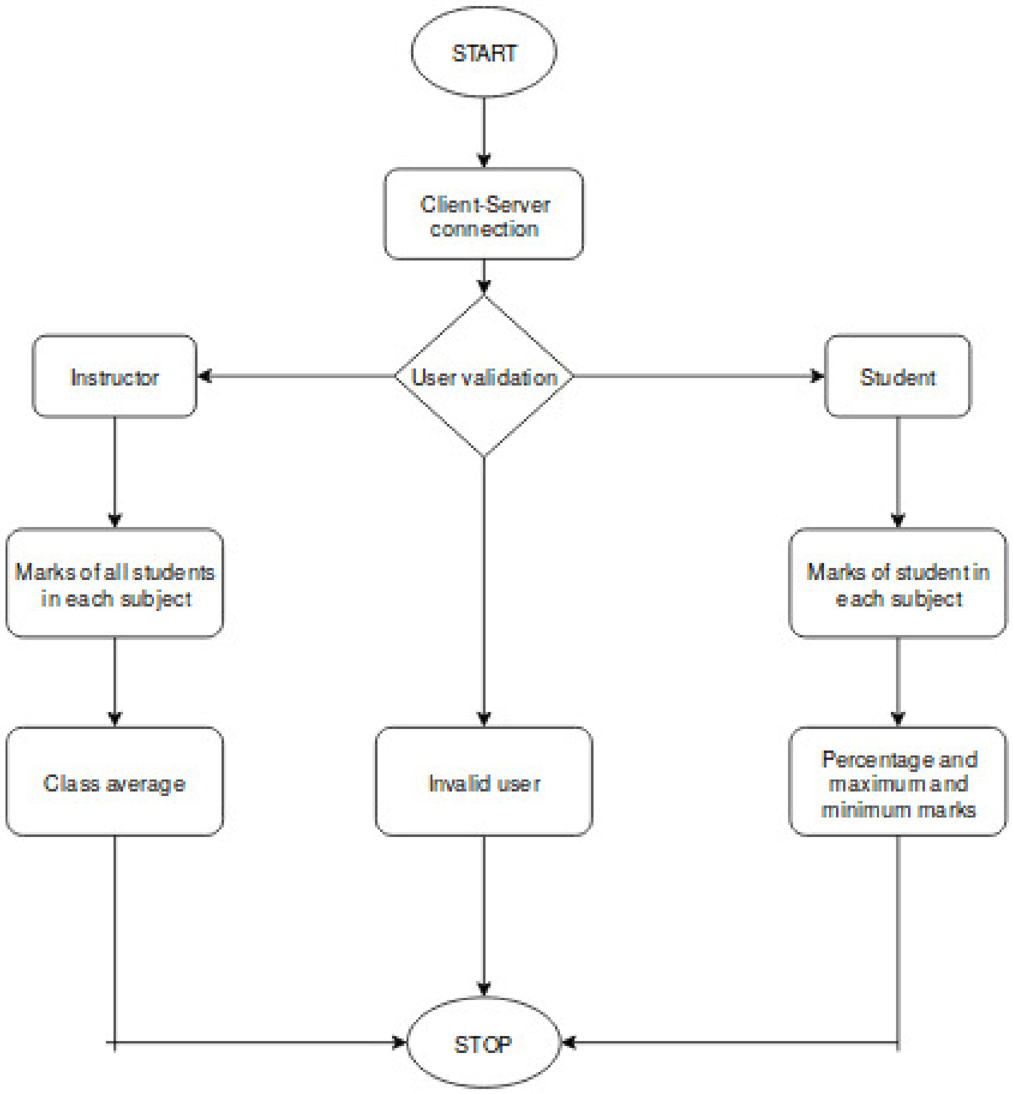
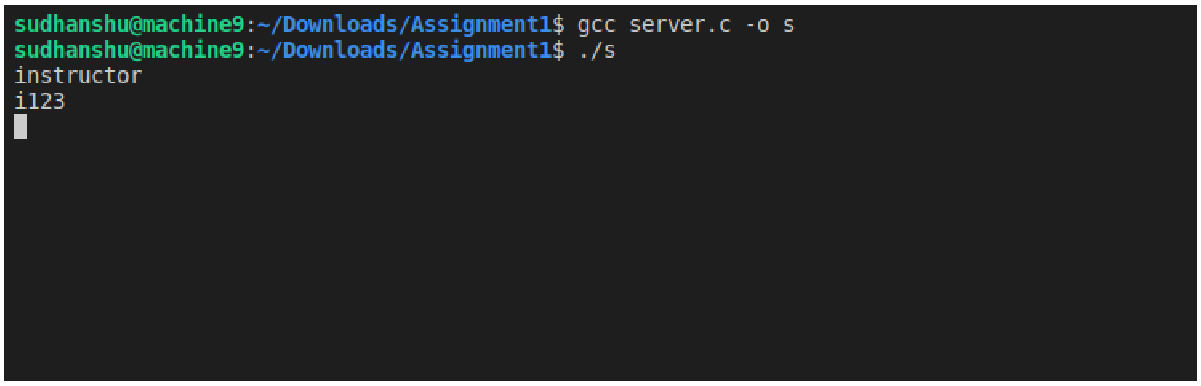

START
Client-Server connection
User validation
Instructor
Student
Marks of all students in each subject
Marks of student in each subject
Class average
Invalid user
Percentage and maximum and minimum marks
STOP

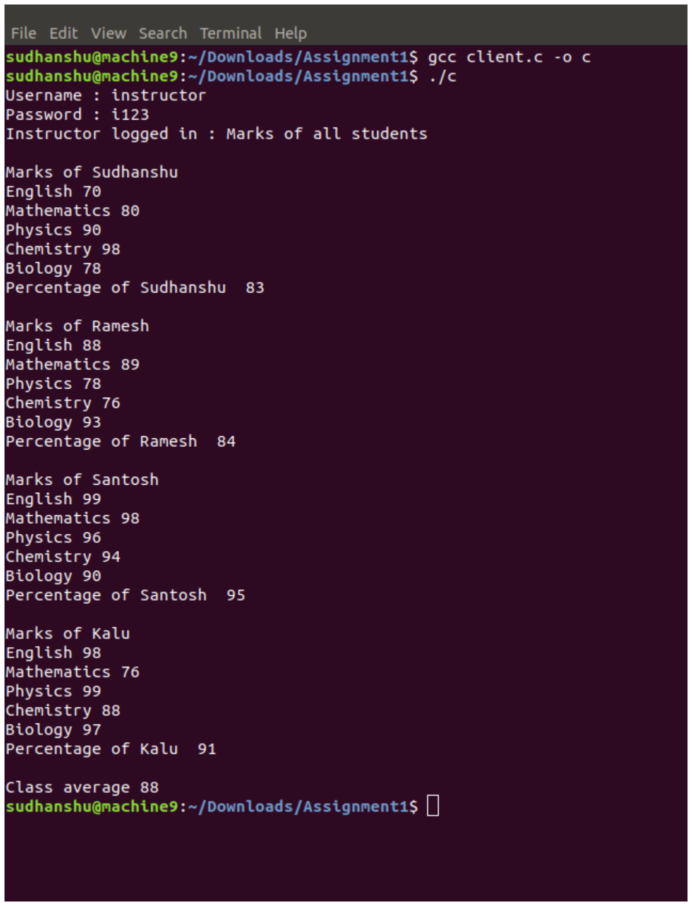
Figure 2: Flow chart



```
sudhanshu@machine9:~/Downloads/Assignment1$ gcc server.c -o s
sudhanshu@machine9:~/Downloads/Assignment1$ ./s
instructor
i123
```

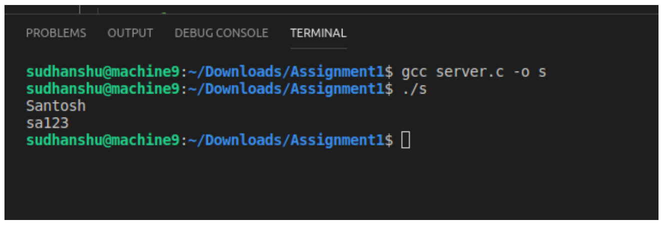
Figure 3: Result on Server log in with Instructor

1.5 Screen shots



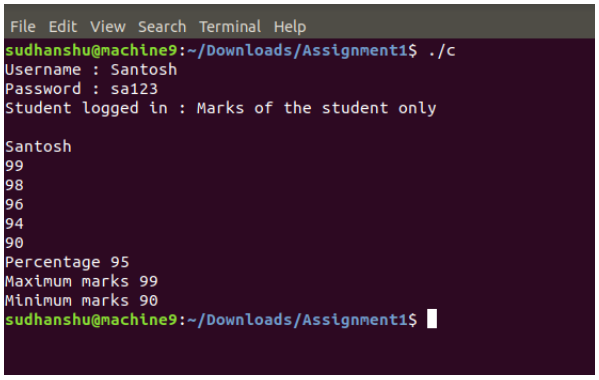
```
sudhanshu@machine9:~/Downloads/Assignment1$ gcc client.c -o c
sudhanshu@machine9:~/Downloads/Assignment1$ ./c
Username : instructor
Password : i123
Instructor logged in : Marks of all students

Marks of Sudhanshu
English 70
Mathematics 80
Physics 90
Chemistry 98
Biology 78
Percentage of Sudhanshu  83

Marks of Ramesh
English 88
Mathematics 89
Physics 78
Chemistry 76
Biology 93
Percentage of Ramesh  84

Marks of Santosh
English 99
Mathematics 98
Physics 96
Chemistry 94
Biology 90
Percentage of Santosh  95

Marks of Kalu
English 98
Mathematics 76
Physics 99
Chemistry 88
Biology 97
Percentage of Kalu  91

Class average 88
sudhanshu@machine9:~/Downloads/Assignment1$ 
```

Figure 4: Log in on Client side with Instructor (shows all data)(a) Marks (individual and aggregate percentage) of each student (b) Class average



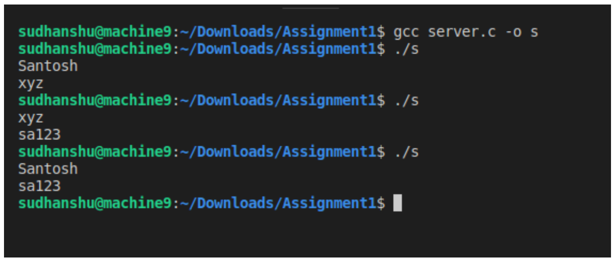
```
sudhanshu@machine9:~/Downloads/Assignment1$ gcc server.c -o s
sudhanshu@machine9:~/Downloads/Assignment1$ ./s
Santosh
sa123
sudhanshu@machine9:~/Downloads/Assignment1$ 
```

Figure 5: Result on Server log in with Student



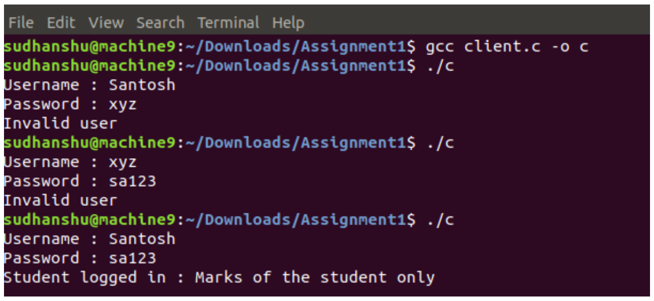
```
sudhanshu@machine9:~/Downloads/Assignment1$ ./c
Username : Santosh
Password : sa123
Student logged in : Marks of the student only

Santosh
99
98
96
94
90
Percentage 95
Maximum marks 99
Minimum marks 90
sudhanshu@machine9:~/Downloads/Assignment1$
```

Figure 6: Log in on Client side with Student (shows all data of Student)
(a)His/her marks in each subject (b)Aggregate percentage (c)Subjects with maximum and minimum marks



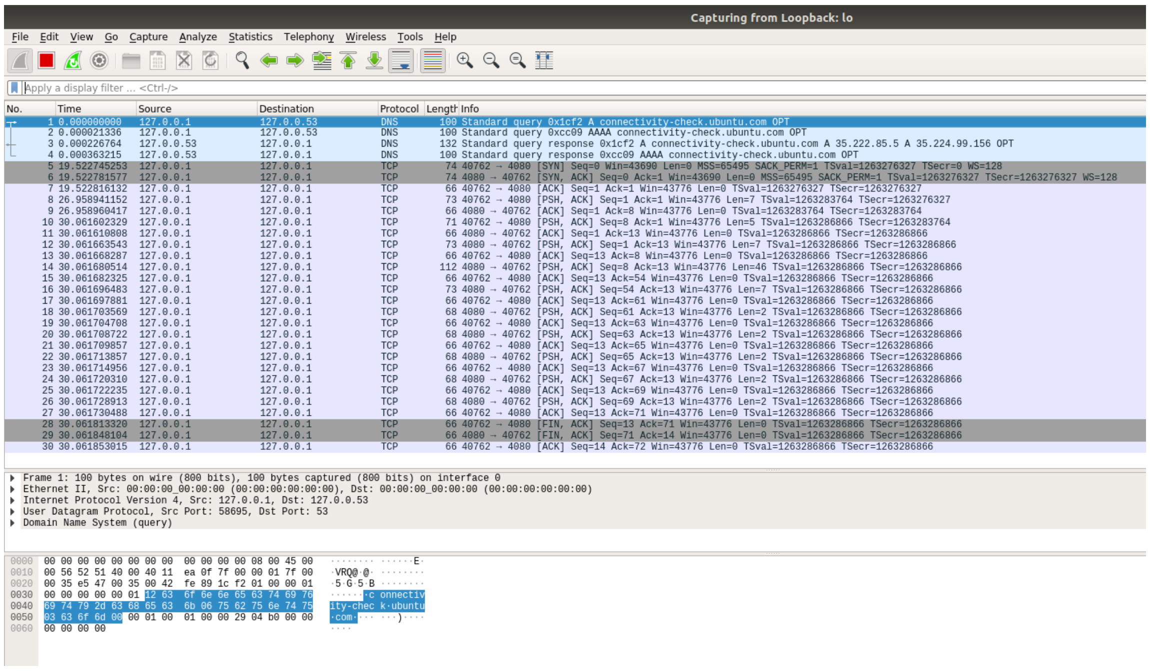
```
sudhanshu@machine9:~/Downloads/Assignment1$ gcc server.c -o s
sudhanshu@machine9:~/Downloads/Assignment1$ ./s
Santosh
xyz
sudhanshu@machine9:~/Downloads/Assignment1$ ./s
xyz
sa123
sudhanshu@machine9:~/Downloads/Assignment1$ ./s
Santosh
sa123
sudhanshu@machine9:~/Downloads/Assignment1$
```

Figure 7: Result on Server when logging with wrong/wright Credentials



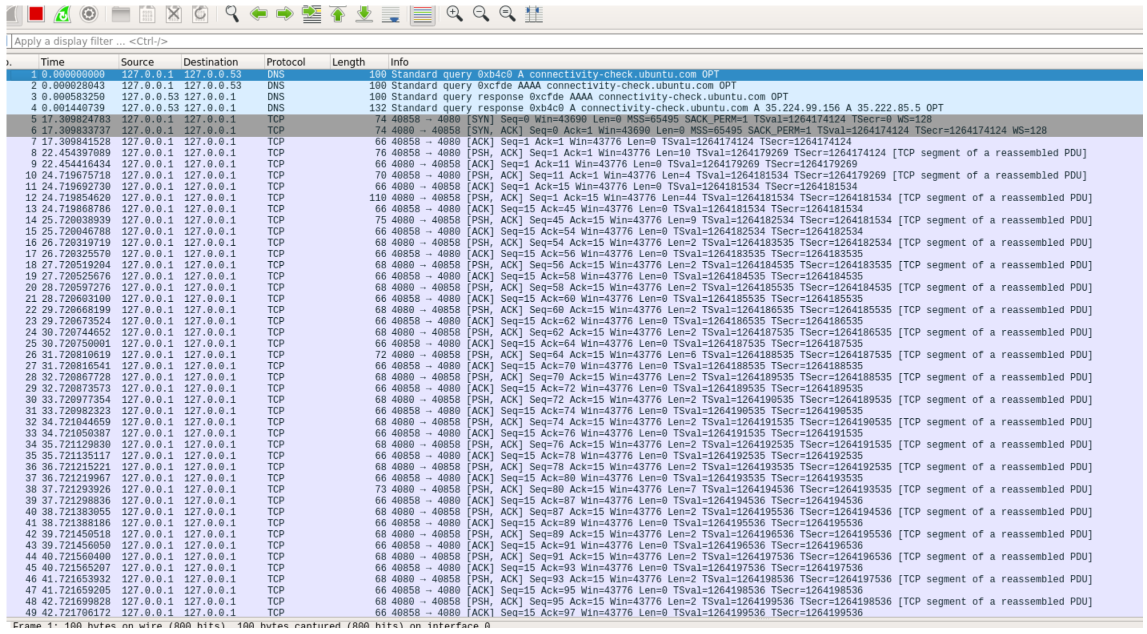
```
sudhanshu@machine9:~/Downloads/Assignment1$ gcc client.c -o c
sudhanshu@machine9:~/Downloads/Assignment1$ ./c
Username : Santosh
Password : xyz
Invalid user
sudhanshu@machine9:~/Downloads/Assignment1$ ./c
Username : xyz
Password : sa123
Invalid user
sudhanshu@machine9:~/Downloads/Assignment1$ ./c
Username : Santosh
Password : sa123
Student logged in : Marks of the student only
```

Figure 8: Result on Client side when logging with wrong/wright Credentials


Capturing from Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

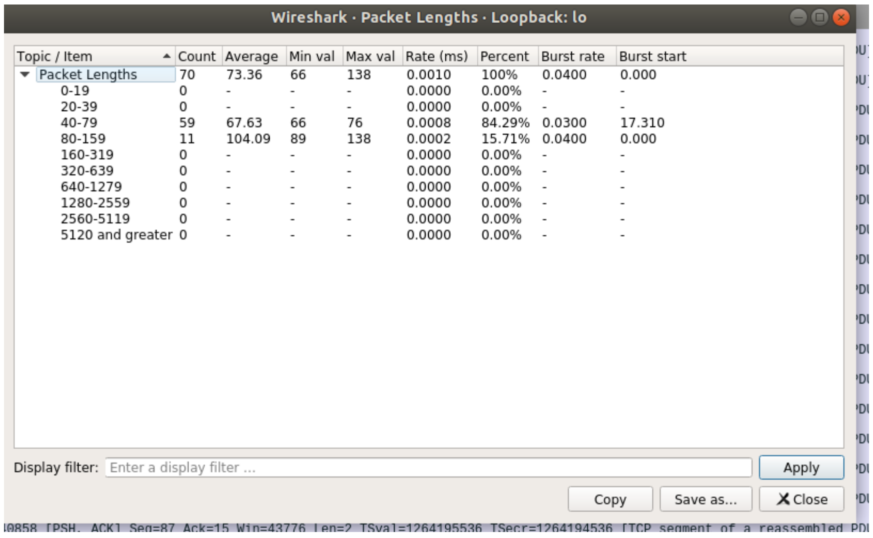
```
No.  Time          Source      Destination  Protocol Length Info
   1 0.000000000   127.0.0.1   127.0.0.53   DNS      100 Standard query 0x1cf2 A connectivity-check.ubuntu.com OPT
   2 0.000021336   127.0.0.1   127.0.0.53   DNS      100 Standard query 0xcc09 AAAA connectivity-check.ubuntu.com OPT
   3 0.000226764   127.0.0.53  127.0.0.1    DNS      132 Standard query response 0x1cf2 A connectivity-check.ubuntu.com A 35.222.85.5 A 35.224.99.156 OPT
   4 0.000363215   127.0.0.53  127.0.0.1    DNS      100 Standard query response 0xcc09 AAAA connectivity-check.ubuntu.com OPT
   5 19.522745253  127.0.0.1   127.0.0.1    TCP       74 40762 → 4080 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1263276327 TSecr=0 WS=128
   6 19.522781577  127.0.0.1   127.0.0.1    TCP       74 4080 → 40762 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1263276327 TSecr=1263276327 WS=128
   7 19.522816132  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=1263276327 TSecr=1263276327
   8 26.958941152  127.0.0.1   127.0.0.1    TCP       73 40762 → 4080 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=7 TSval=1263283764 TSecr=1263276327
   9 26.958960417  127.0.0.1   127.0.0.1    TCP       66 4080 → 40762 [ACK] Seq=1 Ack=8 Win=43776 Len=0 TSval=1263283764 TSecr=1263283764
  10 30.061602329  127.0.0.1   127.0.0.1    TCP       71 40762 → 4080 [PSH, ACK] Seq=8 Ack=1 Win=43776 Len=5 TSval=1263286866 TSecr=1263283764
  11 30.061610808  127.0.0.1   127.0.0.1    TCP       66 4080 → 40762 [ACK] Seq=1 Ack=13 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  12 30.061663543  127.0.0.1   127.0.0.1    TCP       73 4080 → 40762 [PSH, ACK] Seq=1 Ack=13 Win=43776 Len=7 TSval=1263286866 TSecr=1263286866
  13 30.061668287  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=8 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  14 30.061680514  127.0.0.1   127.0.0.1    TCP      112 4080 → 40762 [PSH, ACK] Seq=8 Ack=13 Win=43776 Len=46 TSval=1263286866 TSecr=1263286866
  15 30.061682325  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=54 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  16 30.061696483  127.0.0.1   127.0.0.1    TCP       73 4080 → 40762 [PSH, ACK] Seq=54 Ack=13 Win=43776 Len=7 TSval=1263286866 TSecr=1263286866
  17 30.061697881  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=61 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  18 30.061703569  127.0.0.1   127.0.0.1    TCP       68 4080 → 40762 [PSH, ACK] Seq=61 Ack=13 Win=43776 Len=2 TSval=1263286866 TSecr=1263286866
  19 30.061704708  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=63 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  20 30.061708722  127.0.0.1   127.0.0.1    TCP       68 4080 → 40762 [PSH, ACK] Seq=63 Ack=13 Win=43776 Len=2 TSval=1263286866 TSecr=1263286866
  21 30.061709857  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=65 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  22 30.061713857  127.0.0.1   127.0.0.1    TCP       68 4080 → 40762 [PSH, ACK] Seq=65 Ack=13 Win=43776 Len=2 TSval=1263286866 TSecr=1263286866
  23 30.061714956  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=67 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  24 30.061720310  127.0.0.1   127.0.0.1    TCP       68 4080 → 40762 [PSH, ACK] Seq=67 Ack=13 Win=43776 Len=2 TSval=1263286866 TSecr=1263286866
  25 30.061722235  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=69 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  26 30.061728913  127.0.0.1   127.0.0.1    TCP       68 4080 → 40762 [PSH, ACK] Seq=69 Ack=13 Win=43776 Len=2 TSval=1263286866 TSecr=1263286866
  27 30.061730488  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=13 Ack=71 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  28 30.061813320  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [FIN, ACK] Seq=13 Ack=71 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  29 30.061848104  127.0.0.1   127.0.0.1    TCP       66 4080 → 40762 [FIN, ACK] Seq=71 Ack=14 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866
  30 30.061853015  127.0.0.1   127.0.0.1    TCP       66 40762 → 4080 [ACK] Seq=14 Ack=72 Win=43776 Len=0 TSval=1263286866 TSecr=1263286866

▸ Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
▸ User Datagram Protocol, Src Port: 58695, Dst Port: 53
▸ Domain Name System (query)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 56 52 51 40 00 40 11  ea 0f 7f 00 00 01 7f 00   ·VRQ@·@· ········
0020  00 35 e5 47 00 35 00 42  fe 89 1c f2 01 00 00 01   ·5·G·5·B ········
0030  00 00 00 00 00 01 12 63  6f 6e 6e 65 63 74 69 76   ·······c onnectiv
0040  69 74 79 2d 63 68 65 63  6b 06 75 62 75 6e 74 75   ity-chec k·ubuntu
0050  03 63 6f 6d 00 00 01 00  01 00 00 29 04 b0 00 00   ·com···· ···)····
0060  00 00 00 00                                        ····
```

Figure 9: Wireshark



```
).  Time          Source      Destination  Protocol  Length  Info
   1 0.000000000  127.0.0.1   127.0.0.53   DNS        100 Standard query 0xb4c0 A connectivity-check.ubuntu.com OPT
   2 0.000028043  127.0.0.1   127.0.0.53   DNS        100 Standard query 0xcfde AAAA connectivity-check.ubuntu.com OPT
   3 0.000583250  127.0.0.53  127.0.0.1    DNS        100 Standard query response 0xcfde AAAA connectivity-check.ubuntu.com OPT
   4 0.001440739  127.0.0.53  127.0.0.1    DNS        132 Standard query response 0xb4c0 A connectivity-check.ubuntu.com A 35.224.99.156 A 35.222.85.5 OPT
   5 17.309824783 127.0.0.1   127.0.0.1    TCP         74 40858 → 4080 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1264174124 TSecr=0 WS=128
   6 17.309833737 127.0.0.1   127.0.0.1    TCP         74 4080 → 40858 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1264174124 TSecr=1264174124 WS=128
   7 17.309841528 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=1264174124 TSecr=1264174124
   8 22.454397089 127.0.0.1   127.0.0.1    TCP         76 40858 → 4080 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=10 TSval=1264179269 TSecr=1264174124 [TCP segment of a reassembled PDU]
   9 22.454416434 127.0.0.1   127.0.0.1    TCP         66 4080 → 40858 [ACK] Seq=1 Ack=11 Win=43776 Len=0 TSval=1264179269 TSecr=1264179269
  10 24.719675718 127.0.0.1   127.0.0.1    TCP         70 40858 → 4080 [PSH, ACK] Seq=11 Ack=1 Win=43776 Len=4 TSval=1264181534 TSecr=1264179269 [TCP segment of a reassembled PDU]
  11 24.719692730 127.0.0.1   127.0.0.1    TCP         66 4080 → 40858 [ACK] Seq=1 Ack=15 Win=43776 Len=0 TSval=1264181534 TSecr=1264181534
  12 24.719854620 127.0.0.1   127.0.0.1    TCP        110 4080 → 40858 [PSH, ACK] Seq=1 Ack=15 Win=43776 Len=44 TSval=1264181534 TSecr=1264181534 [TCP segment of a reassembled PDU]
  13 24.719868786 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=45 Win=43776 Len=0 TSval=1264181534 TSecr=1264181534
  14 25.720038939 127.0.0.1   127.0.0.1    TCP         75 4080 → 40858 [PSH, ACK] Seq=45 Ack=15 Win=43776 Len=9 TSval=1264182534 TSecr=1264181534 [TCP segment of a reassembled PDU]
  15 25.720046788 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=54 Win=43776 Len=0 TSval=1264182534 TSecr=1264182534
  16 26.720319719 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=54 Ack=15 Win=43776 Len=2 TSval=1264183535 TSecr=1264182534 [TCP segment of a reassembled PDU]
  17 26.720325570 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=56 Win=43776 Len=0 TSval=1264183535 TSecr=1264183535
  18 27.720519204 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=56 Ack=15 Win=43776 Len=2 TSval=1264184535 TSecr=1264183535 [TCP segment of a reassembled PDU]
  19 27.720525676 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=58 Win=43776 Len=0 TSval=1264184535 TSecr=1264184535
  20 28.720597276 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=58 Ack=15 Win=43776 Len=2 TSval=1264185535 TSecr=1264184535 [TCP segment of a reassembled PDU]
  21 28.720603100 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=60 Win=43776 Len=0 TSval=1264185535 TSecr=1264185535
  22 29.720668199 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=60 Ack=15 Win=43776 Len=2 TSval=1264186535 TSecr=1264185535 [TCP segment of a reassembled PDU]
  23 29.720673524 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=62 Win=43776 Len=0 TSval=1264186535 TSecr=1264186535
  24 30.720744652 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=62 Ack=15 Win=43776 Len=2 TSval=1264187535 TSecr=1264186535 [TCP segment of a reassembled PDU]
  25 30.720750001 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=64 Win=43776 Len=0 TSval=1264187535 TSecr=1264187535
  26 31.720810619 127.0.0.1   127.0.0.1    TCP         72 4080 → 40858 [PSH, ACK] Seq=64 Ack=15 Win=43776 Len=6 TSval=1264188535 TSecr=1264187535 [TCP segment of a reassembled PDU]
  27 31.720816541 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=70 Win=43776 Len=0 TSval=1264188535 TSecr=1264188535
  28 32.720867728 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=70 Ack=15 Win=43776 Len=2 TSval=1264189535 TSecr=1264188535 [TCP segment of a reassembled PDU]
  29 32.720873573 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=72 Win=43776 Len=0 TSval=1264189535 TSecr=1264189535
  30 33.720977354 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=72 Ack=15 Win=43776 Len=2 TSval=1264190535 TSecr=1264189535 [TCP segment of a reassembled PDU]
  31 33.720982323 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=74 Win=43776 Len=0 TSval=1264190535 TSecr=1264190535
  32 34.721044659 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=74 Ack=15 Win=43776 Len=2 TSval=1264191535 TSecr=1264190535 [TCP segment of a reassembled PDU]
  33 34.721050387 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=76 Win=43776 Len=0 TSval=1264191535 TSecr=1264191535
  34 35.721129830 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=76 Ack=15 Win=43776 Len=2 TSval=1264192535 TSecr=1264191535 [TCP segment of a reassembled PDU]
  35 35.721135117 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=78 Win=43776 Len=0 TSval=1264192535 TSecr=1264192535
  36 36.721215221 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=78 Ack=15 Win=43776 Len=2 TSval=1264193535 TSecr=1264192535 [TCP segment of a reassembled PDU]
  37 36.721219967 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=80 Win=43776 Len=0 TSval=1264193535 TSecr=1264193535
  38 37.721293926 127.0.0.1   127.0.0.1    TCP         73 4080 → 40858 [PSH, ACK] Seq=80 Ack=15 Win=43776 Len=7 TSval=1264194536 TSecr=1264193535 [TCP segment of a reassembled PDU]
  39 37.721298836 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=87 Win=43776 Len=0 TSval=1264194536 TSecr=1264194536
  40 38.721383055 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=87 Ack=15 Win=43776 Len=2 TSval=1264195536 TSecr=1264194536 [TCP segment of a reassembled PDU]
  41 38.721388186 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=89 Win=43776 Len=0 TSval=1264195536 TSecr=1264195536
  42 39.721450518 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=89 Ack=15 Win=43776 Len=2 TSval=1264196536 TSecr=1264195536 [TCP segment of a reassembled PDU]
  43 39.721456050 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=91 Win=43776 Len=0 TSval=1264196536 TSecr=1264196536
  44 40.721560400 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=91 Ack=15 Win=43776 Len=2 TSval=1264197536 TSecr=1264196536 [TCP segment of a reassembled PDU]
  45 40.721565207 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=93 Win=43776 Len=0 TSval=1264197536 TSecr=1264197536
  46 41.721653932 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=93 Ack=15 Win=43776 Len=2 TSval=1264198536 TSecr=1264197536 [TCP segment of a reassembled PDU]
  47 41.721659205 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=95 Win=43776 Len=0 TSval=1264198536 TSecr=1264198536
  48 42.721699828 127.0.0.1   127.0.0.1    TCP         68 4080 → 40858 [PSH, ACK] Seq=95 Ack=15 Win=43776 Len=2 TSval=1264199536 TSecr=1264198536 [TCP segment of a reassembled PDU]
  49 42.721706172 127.0.0.1   127.0.0.1    TCP         66 40858 → 4080 [ACK] Seq=15 Ack=97 Win=43776 Len=0 TSval=1264199536 TSecr=1264199536
Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
```

Figure 10: Analyze packet size and frame size in TCP/IP layers



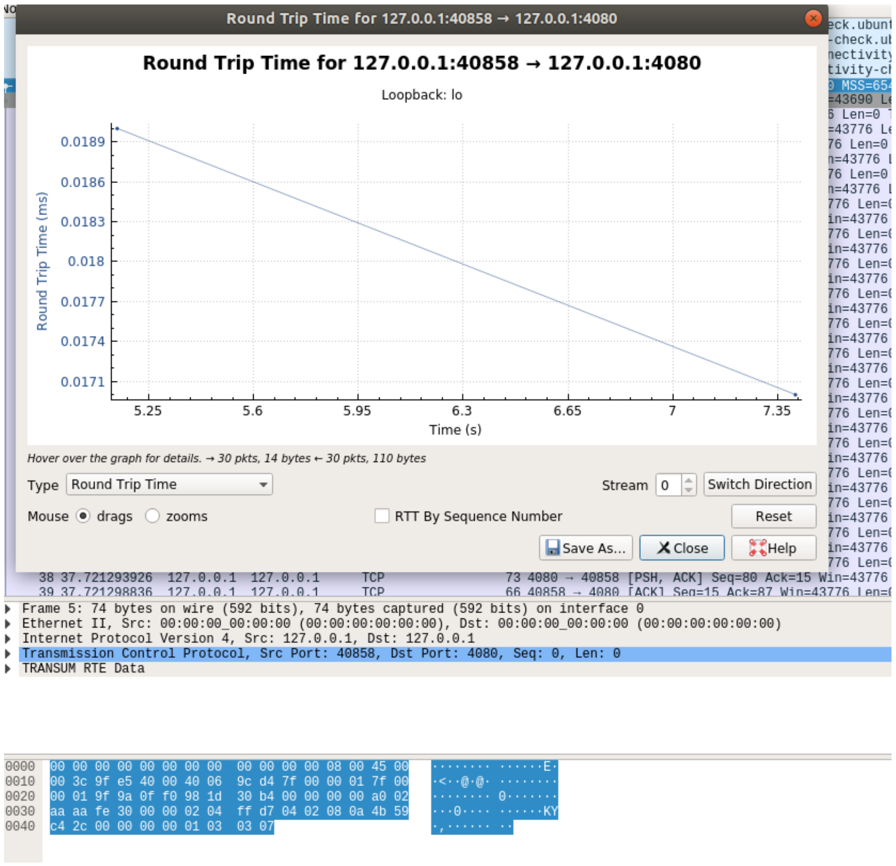
Wireshark · Packet Lengths · Loopback: lo

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	70	73.36	66	138	0.0010	100%	0.0400	0.000
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	59	67.63	66	76	0.0008	84.29%	0.0300	17.310
80-159	11	104.09	89	138	0.0002	15.71%	0.0400	0.000
160-319	0	-	-	-	0.0000	0.00%	-	-
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	0	-	-	-	0.0000	0.00%	-	-
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter: Enter a display filter ...

```
l0858 [PSH, ACK] Seq=87 Ack=15 Win=43776 Len=2 TSval=1264195536 TSecr=1264194536 [TCP segment of a reassembled PDU
```

Figure 11: Packet lenth



Hover over the graph for details. → 30 pkts, 14 bytes ← 30 pkts, 110 bytes

Type: Round Trip Time

Stream: 0

Mouse: [✓ drags] zooms

☐ RTT By Sequence Number

```
38 37.721293926  127.0.0.1  127.0.0.1    TCP      73 4080 → 40858 [PSH, ACK] Seq=80 Ack=15 Win=43776
39 37.721298836  127.0.0.1  127.0.0.1    TCP      66 40858 → 4080 [ACK] Seq=15 Ack=87 Win=43776 Len=
Frame 5: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 40858, Dst Port: 4080, Seq: 0, Len: 0
TRANSUM RTE Data

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 3c 9f e5 40 00 40 06  9c d4 7f 00 00 01 7f 00   ·<··@·@· ········
0020  00 01 9f 9a 0f f0 98 1d  30 b4 00 00 00 00 a0 02   ········ 0·······
0030  aa aa fe 30 00 00 02 04  ff d7 04 02 08 0a 4b 59   ···0···· ······KY
0040  c4 2c 00 00 00 00 01 03  03 07                     ·,······ ··
```

Figure 12: Packet round trip time from server to client

A Appendix

A.1 Code of Server

```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 4080

// C function showing how to do time delay
// To use time library of C
#include <time.h>

void delay(int number_of_seconds)
{
// Converting time into milli_seconds
int milli_seconds = 1000000 * number_of_seconds;

// Stroing start time
clock_t start_time = clock();

// looping till required time is not acheived
while (clock() < start_time + milli_seconds)
;
}


int main()
{

//client server program//

    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[500] = {0};

// 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
```

```
    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 4080

    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);
    }

//username and password validiation check//

    char user[20],pwd[20], user1[20], pwd1[20];
    int utype;

    valread = read( new_socket , buffer, 500);
    strcpy(user,buffer);
    printf("%s\n",user);
    if(strcmp(user, "instructor")==0){
        utype = 1;
    }
    else
    {
        utype = 2;
    }

    bzero(buffer,sizeof(buffer));

    valread = read( new_socket , buffer, 500);
    strcpy(pwd,buffer);
    printf("%s\n",pwd);
    bzero(buffer,sizeof(buffer));

//file open //

    FILE *fptr;
    fptr = fopen("user_pass.txt","r");

    if (fptr == NULL){
      printf("error \n");
      exit(1);}

    else{

    while(fscanf(fptr, "%s %s", user1, pwd1)!=EOF){

        if(strcmp(user1, user)==0){
            if(strcmp(pwd1, pwd)==0){
            //printf("valid user\n");
                if (utype !=1)
                {


                    char print1[50] = "Student logged in : Marks of the student only\n";
                    send(new_socket , print1 , strlen(print1) , 0 );



    char student[20],m1[3], m2[3], m3[3], m4[3], m5[3];
            FILE *fptr1;
              fptr1 = fopen("student_marks.txt","r");

             if (fptr1 == NULL){
                    printf("error \n");
                  exit(1);}

                else{

             while(fscanf(fptr1, "%s %s %s %s %s %s ", student, m1,m2,m3,m4,m5)!=EOF){

                 if(strcmp(student, user)==0){

              send(new_socket , student , strlen(student) , 0 );
               delay(0.01);
              send(new_socket , m1 , strlen(m1) , 0 );
               delay(0.01);
              send(new_socket , m2 , strlen(m2) , 0 );
               delay(0.01);
              send(new_socket , m3 , strlen(m3) , 0 );
               delay(0.01);
              send(new_socket , m4 , strlen(m4) , 0 );
               delay(0.01);
```

```
          send(new_socket , m5 , strlen(m5) , 0 );
           delay(0.1);

            }


     }


}
fclose(fptr1);


            }
            else
            {   char print2[50] = "Instructor logged in : Marks of all students";
                send(new_socket , print2 , strlen(print2) , 0 );
                delay(1);

    char student[20],m1[3], m2[3], m3[3], m4[3], m5[3];
                FILE *fptr1;
                fptr1 = fopen("student_marks.txt","r");

                if (fptr1 == NULL){
                         printf("error \n");
                    exit(1);}

                else{

         while(fscanf(fptr1, "%s %s %s %s %s %s ", student, m1,m2,m3,m4,m5)!=EOF){
          send(new_socket , student , strlen(student) , 0 );
           delay(1);
          send(new_socket , m1 , strlen(m1) , 0 );
           delay(1);
          send(new_socket , m2 , strlen(m2) , 0 );
           delay(1);
          send(new_socket , m3 , strlen(m3) , 0 );
           delay(1);
          send(new_socket , m4 , strlen(m4) , 0 );
           delay(1);
          send(new_socket , m5 , strlen(m5) , 0 );
           delay(1);
     }


}
fclose(fptr1);



            }
            }
            else
            {
                //printf("invalid user\n");
            }
     }
        else{
            // char print3[50] = "Invalid user\n";
             // send(new_socket , print3 , strlen(print3) , 0 );
             printf("invalid user\n");
     }

    }
    }

    fclose(fptr);

 //display marks//

    //char student[20],m1[3], m2[3], m3[3], m4[3], m5[3];



//reading student marks file//

   /*/ FILE *fptr1;

switch (utype)
{
case 1:
    printf("instructor");
    break;

case 2:

    fptr1 = fopen("student_marks.txt","r");

    if (fptr1 == NULL){
      printf("error \n");
      exit(1);}

    else{
         while(fscanf(fptr, "%s %s %s %s %s %s ", student, m1,m2,m3,m4,m5)!=EOF){
              send(new_socket , student , strlen(student) , 0 );
              send(new_socket , m1 , strlen(m1) , 0 );
              send(new_socket , m2 , strlen(m2) , 0 );
              send(new_socket , m3 , strlen(m3) , 0 );
              send(new_socket , m4 , strlen(m4) , 0 );
              send(new_socket , m5 , strlen(m5) , 0 );
         }


    }
    fclose(fptr1);

        break;

default:
    break;
}*/

    return 0;
}
```

A.2 Code of Client

```
// Client side C/C++ program to demonstrate Socket programming
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#define PORT 4080

// C function showing how to do time delay
// To use time library of C
#include <time.h>

void delay(int number_of_seconds)
{
// Converting time into milli_seconds
int milli_seconds = 1000000 * number_of_seconds;

// Stroing start time
clock_t start_time = clock();

// looping till required time is not acheived
while (clock() < start_time + milli_seconds)
;
}


int main()
{

//client server program//

    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char buffer[500] = {0};

//socket connection//

    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;
    }

//username and password input//

    char user[20], pwd[20],student1[20];

    printf("Username : ");
    scanf("%s",user);
    send(sock , user , strlen(user) , 0 );


    printf("Password : ");
    scanf("%s",pwd);
    send(sock , pwd , strlen(pwd) , 0 );




if(strcmp(user, "instructor")==0 && strcmp(pwd, "i123")==0)
{

char print2[50];
    valread = read( sock , buffer, 500);
    strcpy(print2,buffer);
    printf("%s\n",print2);
    printf("\n");
    bzero(buffer,sizeof(buffer));

    char student[20],m1[3], m2[3], m3[3], m4[3], m5[3];
    int count , avg = 0;
for(count =0; count <20;count++)
  {  valread = read( sock , buffer, 500);
    strcpy(student,buffer);
    printf("Marks of %s",student);
    printf("\n");


      valread = read( sock , buffer, 500);
    strcpy(m1,buffer);
    printf("English %s",m1);
    printf("\n");
    bzero(buffer,sizeof(buffer));
```

```
      valread = read( sock , buffer, 500);
    strcpy(m2,buffer);
    printf("Mathematics %s",m2);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m3,buffer);
    printf("Physics %s",m3);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m4,buffer);
    printf("Chemistry %s",m4);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m5,buffer);
    printf("Biology %s",m5);
    printf("\n");
    bzero(buffer,sizeof(buffer));

//////////aggregate percentage of each student///////////

  int sum = 0,per = 0;
  int x1 = atoi(m1);
  int x2 = atoi(m2);
  int x3 = atoi(m3);
  int x4 = atoi(m4);
  int x5 = atoi(m5);
   sum=(x1+x2+x3+x4+x5);
   per = sum/5;
   avg = avg + per;
   printf("Percentage of %s  %d\n",student, per);
   printf("\n");

}
  printf("Class average %d\n",avg/20);
}
else /* if ( strcmp(user, student1)==0 )*/ {
  char print1[50];
    valread = read( sock , buffer, 500);
    strcpy(print1,buffer);
    printf("%s\n",print1);
    bzero(buffer,sizeof(buffer));

    char student[20],m1[3], m2[3], m3[3], m4[3], m5[3];
    valread = read( sock , buffer, 500);
    strcpy(student,buffer);
    printf("%s",student);
    printf("\n");
    bzero(buffer,sizeof(buffer));


      valread = read( sock , buffer, 500);
    strcpy(m1,buffer);
    printf("%s",m1);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m2,buffer);
    printf("%s",m2);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m3,buffer);
    printf("%s",m3);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m4,buffer);
    printf("%s",m4);
    printf("\n");
    bzero(buffer,sizeof(buffer));

      valread = read( sock , buffer, 500);
    strcpy(m5,buffer);
    printf("%s",m5);
    printf("\n");
    bzero(buffer,sizeof(buffer));

  int i, sum = 0,avg = 0;
  int x1 = atoi(m1);
  int x2 = atoi(m2);
  int x3 = atoi(m3);
  int x4 = atoi(m4);
  int x5 = atoi(m5);
   sum=(x1+x2+x3+x4+x5);
   avg = (sum/5);
   printf("Percentage %d\n",avg);

////////////////////maximum marks///////////////////////////
   if(x1>x2&&x1>x3&&x1>x4&&x1>x5){
     printf("Maximum marks %d\n",x1);
   }
   else if(x2>x1&&x2>x3&&x2>x4&&x2>x5){
     printf("Maximum marks %d\n",x2);
   }
   else if(x3>x2&&x3>x4&&x3>x5&&x3>x1){
     printf("Maximum marks %d\n",x3);
   }
   else if(x4>x1&&x4>x3&&x4>x2&&x4>x5){
     printf("Maximum marks %d\n",x4);
   }
   else {
     printf("Maximum marks %d\n",x5);
   }
//////////////////minimum marks////////////////////

    if(x1<x2&&x1<x3&&x1<x4&&x1<x5){
     printf("Minimum marks %d\n",x1);
   }
   else if(x2<x1&&x2<x3&&x2<x4&&x2<x5){
     printf("Minimum marks %d\n",x2);
   }
   else if(x3<x2&&x3<x4&&x3<x5&&x3<x1){
     printf("Minimum marks %d\n",x3);
   }
   else if(x4<x1&&x4<x3&&x4<x2&&x4<x5){
     printf("Minimum marks %d\n",x4);
   }
   else {
     printf("Minimum marks %d\n",x5);
   }
}
//else{
 // printf("Invalid user\n");
//}

/*

//////////////////////////////////Choose your option ///////////////////////////////////////////////

int code;
  printf("Choose your option : 1. His/her marks in each subject
  2. Aggregate percentage 3.Subjects with maximum and minimum marks");

  scanf("%d",&code);

  switch(code)
  {
    case 1:
      puts("His/her marks in each subject ");


      break;
    case 2:
      puts("Aggregate percentage ");


      break;
    default:
      puts("Subjects with maximum and minimum marks");


  }
}
*/



    return 0;
}
```

References

[1] *The basics of socket programming are given at.* `http://www.linuxhowtos.org/C_C++/socket.htm`.

[2] *Beej network programming guide.* `http://beej.us/guide/bgnet/output/html/multipage/index.html`.

[3] *Network programming by Richard Stevens.* .

[4] *Socket programming.* `https://www.geeksforgeeks.org/socket-programming-cc/`.