

Computer Network

Assignment – 1

Question 1:

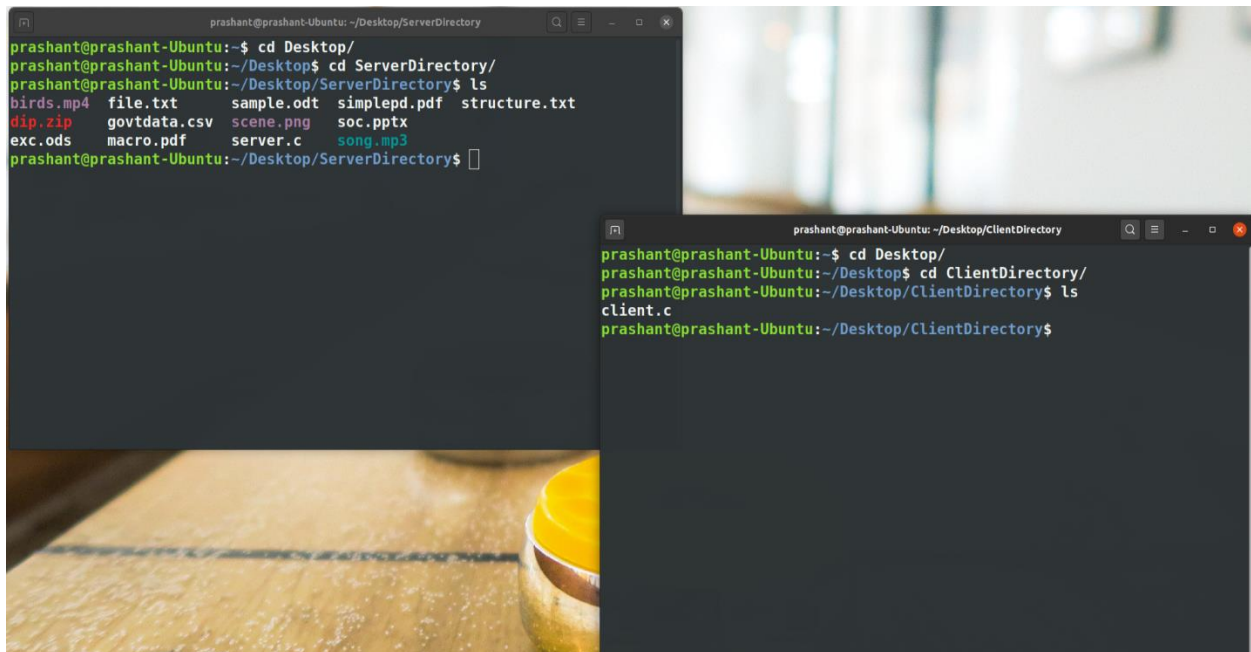
The solution of this problem is totally based on Socket programming (C programming).

The submitted .zip folder contains the file server.c and client.c, and their code explanation is written in the code as a comment.

Here in this document, just a demo of compiling and running of these two files (server.c and client.c) by some screenshots.

Firstly, we have to create two directories, ServerDirectory for server and ClientDirectory for client and also their .c file present in their respective directory.

As we can see in the server directory contains server.c file and some sample files, and client directory contains only client.c



The image displays two terminal windows from a Linux system. The top window shows the user navigating to the Desktop, then to a newly created 'ServerDirectory', and listing its contents. The directory contains various sample files like birds.mp4, file.txt, sample.odt, simplepd.pdf, structure.txt, dip.zip, govtdata.csv, scene.png, soc.pptx, exc.ods, macro.pdf, server.c, and song.mp3. The bottom window shows the user navigating to the Desktop, then to a newly created 'ClientDirectory', and listing its contents, which only includes the 'client.c' file.

```
prashant@prashant-Ubuntu: ~/Desktop/ServerDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ServerDirectory/
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ls
birds.mp4  file.txt      sample.odt  simplepd.pdf  structure.txt
dip.zip    govtdata.csv  scene.png   soc.pptx
exc.ods    macro.pdf     server.c     song.mp3
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$

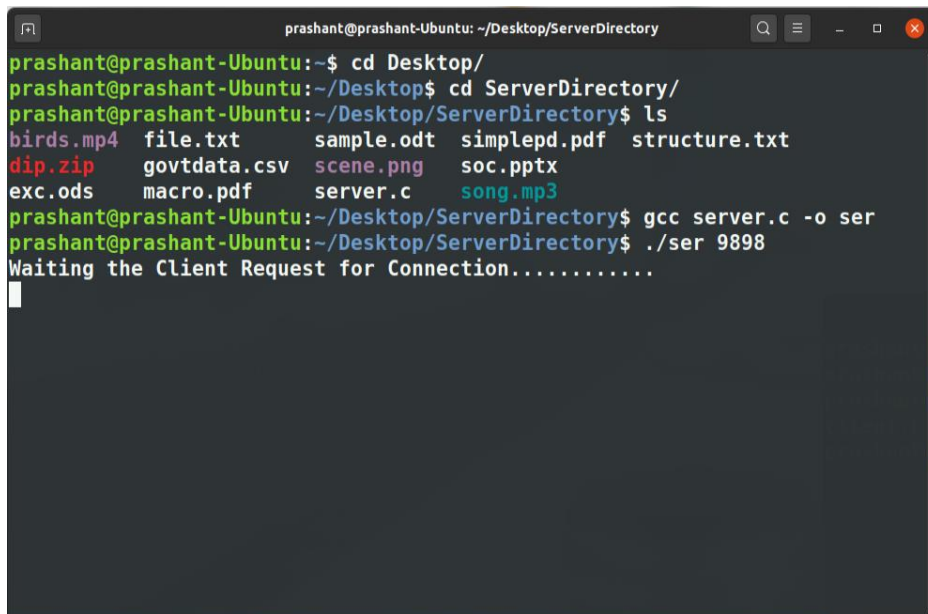
prashant@prashant-Ubuntu: ~/Desktop/ClientDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ClientDirectory/
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
client.c
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$
```

PRASHANT
2018360

Instruction for Running the server.c

This code takes two argument at the time of Running of code
1. Executable file name or may be called as output file name
2. PORT NO. (always takes greater than 5000)

Compiling and Running of an code by example:
gcc server.c -o ser --- Compiling
./ser 9898 --- Running

A screenshot of a terminal window titled 'prashant@prashant-Ubuntu: ~/Desktop/ServerDirectory'. The terminal shows the following commands and output:

```
prashant@prashant-Ubuntu:~$ cd Desktop/  
prashant@prashant-Ubuntu:~/Desktop$ cd ServerDirectory/  
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ls  
birds.mp4  file.txt      sample.odt  simplepd.pdf  structure.txt  
dip.zip    govtdata.csv  scene.png   soc.pptx  
exc.ods    macro.pdf     server.c     song.mp3  
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ gcc server.c -o ser  
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ./ser 9898  
Waiting the Client Request for Connection.....
```

Instruction for Running the client.c

This code takes three argument at the time of Running of code
1. Executable file name or may be called as output file name
2. IP address of the server (if we are working on same machine then
enter loopback ip address which is 127.0.0.1 otherwise for different machines
then enter the server's address)
3. PORT NO. (Just to be make sure that PORT NO. is same as the Server's PORT NO.)

Compiling and Running of an code by example:
gcc client.c -o cl --- Compiling
./cl 127.0.0.1 9898 --- Running

PRASHANT
2018360

```
prashant@prashant-Ubuntu: ~/Desktop/ClientDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ClientDirectory/
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
client.c
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ gcc client.c -o cl
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ./cl 127.0.0.1 9898
```

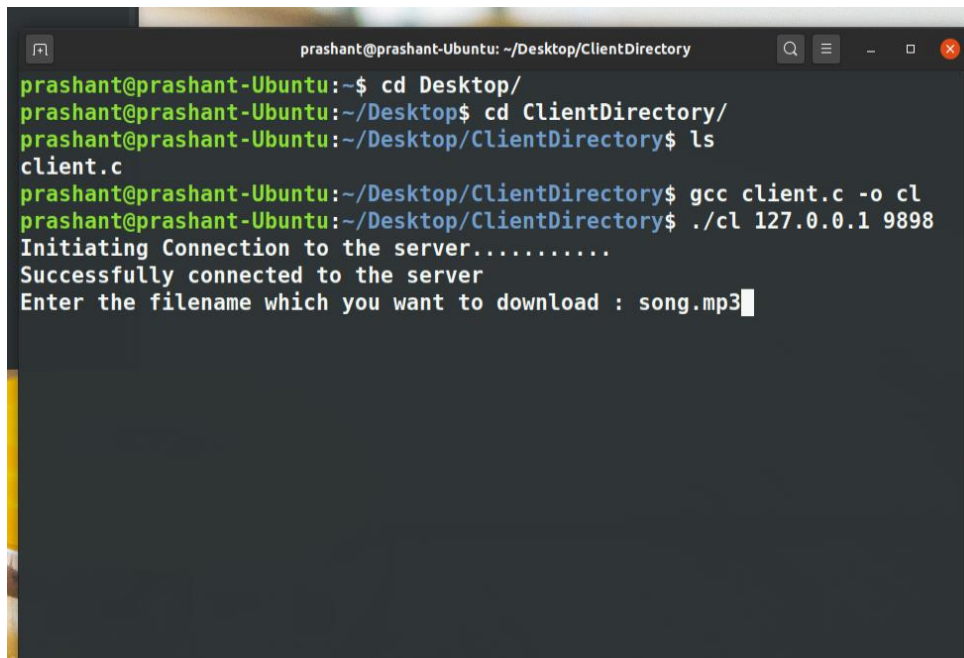
After client connected to the server, client takes the input as filename

```
prashant@prashant-Ubuntu: ~/Desktop/ServerDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ServerDirectory/
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ls
birds.mp4  file.txt  sample.odt  simplepd.pdf  structure.txt
dip.zip    govtdata.csv  scene.png  soc.pptx
exc.ods    macro.pdf  server.c    song.mp3
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ gcc server.c -o ser
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ./ser 9898
Waiting the Client Request for Connection.....
Receiving request from Client for Connection..
Client successfully Connected

prashant@prashant-Ubuntu: ~/Desktop/ClientDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ClientDirectory/
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
client.c
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ gcc client.c -o cl
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ./cl 127.0.0.1 9898
Initiating Connection to the server.....
Successfully connected to the server
Enter the filename which you want to download :
```

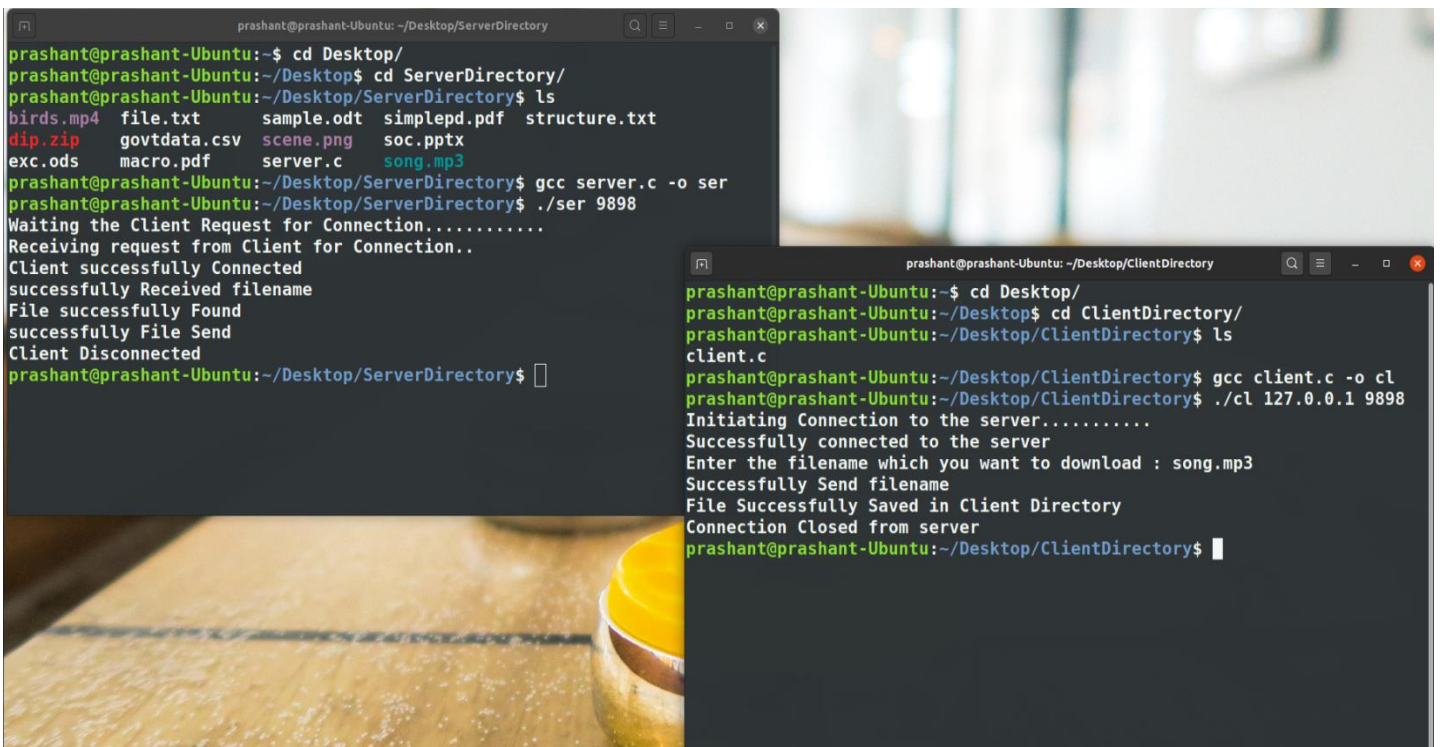
PRASHANT 2018360

Give the input song.mp3, which is present in the server directory, I'm here just using a .mp3 file for an example, you can use any type of file whether it is .txt, .pdf, .pptx, .csv, .png, .jpeg, .mp4, .zip, etc.



```
prashant@prashant-Ubuntu: ~/Desktop/ClientDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ClientDirectory/
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
client.c
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ gcc client.c -o cl
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ./cl 127.0.0.1 9898
Initiating Connection to the server.....
Successfully connected to the server
Enter the filename which you want to download : song.mp3
```

Now, successful message pop out that the server sends the file to the client

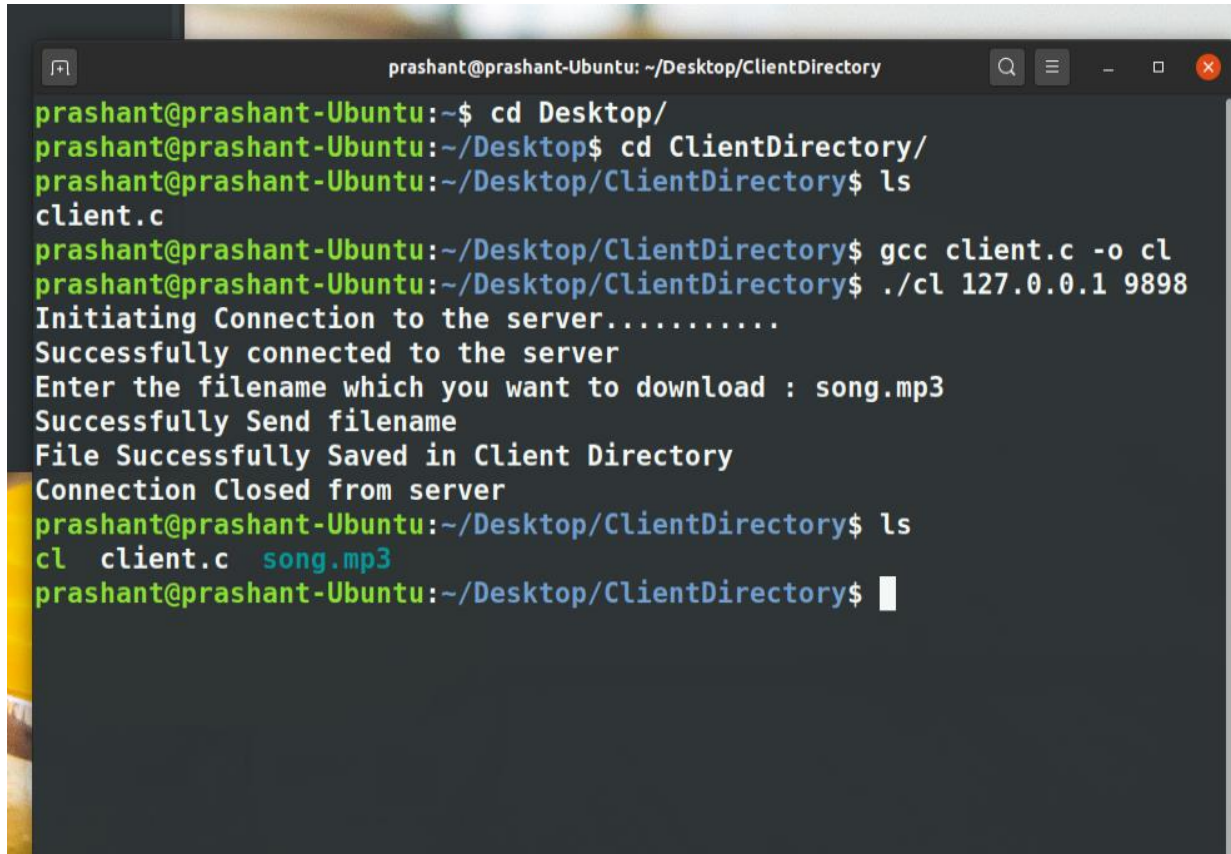


```
prashant@prashant-Ubuntu: ~/Desktop/ServerDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ServerDirectory/
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ls
birds.mp4  file.txt  sample.odt  simplepd.pdf  structure.txt
dip.zip    govtdata.csv  scene.png  soc.pptx
exc.ods    macro.pdf  server.c    song.mp3
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ gcc server.c -o ser
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$ ./ser 9898
Waiting the Client Request for Connection.....
Receiving request from Client for Connection..
Client successfully Connected
successfully Received filename
File successfully Found
successfully File Send
Client Disconnected
prashant@prashant-Ubuntu:~/Desktop/ServerDirectory$
```

```
prashant@prashant-Ubuntu: ~/Desktop/ClientDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ClientDirectory/
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
client.c
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ gcc client.c -o cl
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ./cl 127.0.0.1 9898
Initiating Connection to the server.....
Successfully connected to the server
Enter the filename which you want to download : song.mp3
Successfully Send filename
File Successfully Saved in Client Directory
Connection Closed from server
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$
```


PRASHANT
2018360

As we can see, Now song.mp3 file is present in local directory of client



```
prashant@prashant-Ubuntu: ~/Desktop/ClientDirectory
prashant@prashant-Ubuntu:~$ cd Desktop/
prashant@prashant-Ubuntu:~/Desktop$ cd ClientDirectory/
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
client.c
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ gcc client.c -o cl
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ./cl 127.0.0.1 9898
Initiating Connection to the server.....
Successfully connected to the server
Enter the filename which you want to download : song.mp3
Successfully Send filename
File Successfully Saved in Client Directory
Connection Closed from server
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$ ls
cl  client.c  song.mp3
prashant@prashant-Ubuntu:~/Desktop/ClientDirectory$
```

PRASHANT
2018360

Question 2:

Wireshark - Capture File Properties - Loopback: lo

File Edit View Go Capture Analyze

Apply a display filter ... <Ctrl-/>

Vo. Time Source

324 8.298726735 127.0.0.1

325 8.322719221 127.0.0.1

326 8.322734252 127.0.0.1

327 8.322741096 127.0.0.1

328 8.324708257 127.0.0.1

329 8.324713904 127.0.0.1

330 8.365908623 127.0.0.1

331 8.366029907 127.0.0.1

332 8.406130816 127.0.0.1

333 8.406205103 127.0.0.1

334 8.419402381 127.0.0.1

335 8.419413212 127.0.0.1

Frame 332: 66 bytes on wire (528 bits) captured (528 bits) on interface 0 (0.0%)

Interface id: 0 (lo)

Interface name: lo

Encapsulation type: Ethernet

Arrival Time: Sep 9, 2020 15:26:17.000000000

[Time shift for this packet: 0.000000000]

Epoch Time: 1599645385.778677

[Time delta from previous capture: 0.000000000]

[Time delta from previous display: 0.000000000]

[Time since reference or first packet: 0.000000000]

Frame Number: 332

Frame Length: 66 bytes (528 bits)

Capture Length: 66 bytes (528 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ethertype:ip:tcp]

[Coloring Rule Name: TCP]

[Coloring Rule String: tcp]

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0010 00 34 0e 42 40 00 00 00 00 00 00 00 00 00 00

0020 00 01 e1 c8 26 aa 6e 2d 6e 2d 6e 2d 6e 2d 6e

0030 02 00 fe 28 00 00 01 01 00 00 00 00 00 00 00

0040 70 03

Details

File

Name: /tmp/wireshark_lo_20200909152542_uFBLwu.pcapng

Length: 1,833 kB

Hash (SHA256): 3b8d1de2d845241ade727547a0faeb17354d8395c7f19098aca1f4b0857cf68

Hash (RIPEMD160): 669d60acedd46af1da13d48d1d52045758970436

Hash (SHA1): 316735edad0baf44aaccc1d1e62aff541c846b4f

Format: Wireshark/... - pcapng

Encapsulation: Ethernet

Time

First packet: 2020-09-09 15:26:17

Last packet: 2020-09-09 15:26:25

Elapsed: 00:00:08

Capture

Hardware: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (with SSE4.2)

OS: Linux 5.4.0-45-generic

Application: Dumpcap (Wireshark) 3.2.3 (Git v3.2.3 packaged as 3.2.3-1)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit
lo	0 (0.0%)	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	335	335 (100.0%)	—
Time span, s	8.419	8.419	—
Average pps	39.8	39.8	—
Average packet size, B	5439	5439	—
Bytes	1822103	1822103 (100.0%)	0
Average bytes/s	216 k	216 k	—
Average bits/s	1,731 k	1,731 k	—

Capture file comments

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	57800 → 9898 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=...
2	0.000022225	127.0.0.1	127.0.0.1	TCP	74	9898 → 57800 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495...
3	0.000039609	127.0.0.1	127.0.0.1	TCP	66	57800 → 9898 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=19841227...
4	7.023985380	127.0.0.1	127.0.0.1	TCP	166	57800 → 9898 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=100 TSval=1...
5	7.024009629	127.0.0.1	127.0.0.1	TCP	66	9898 → 57800 [ACK] Seq=1 Ack=101 Win=65408 Len=0 TSval=198412...
6	7.024146890	127.0.0.1	127.0.0.1	TCP	74	9898 → 57800 [PSH, ACK] Seq=1 Ack=101 Win=65536 Len=8 TSval=1...
7	7.024169125	127.0.0.1	127.0.0.1	TCP	66	57800 → 9898 [ACK] Seq=101 Ack=9 Win=65536 Len=0 TSval=198412...
8	7.026033023	127.0.0.1	127.0.0.1	TCP	67	9898 → 57800 [PSH, ACK] Seq=9 Ack=101 Win=65536 Len=1 TSval=1...
9	7.026039973	127.0.0.1	127.0.0.1	TCP	66	57800 → 9898 [ACK] Seq=101 Ack=10 Win=65536 Len=0 TSval=19841...
10	7.026061988	127.0.0.1	127.0.0.1	TCP	67	9898 → 57800 [PSH, ACK] Seq=10 Ack=101 Win=65536 Len=1 TSval=...
11	7.026065640	127.0.0.1	127.0.0.1	TCP	66	57800 → 9898 [ACK] Seq=101 Ack=11 Win=65536 Len=0 TSval=19841...
12	7.026073575	127.0.0.1	127.0.0.1	TCP	67	9898 → 57800 [PSH, ACK] Seq=11 Ack=101 Win=65536 Len=1 TSval=...
13	7.026076600	127.0.0.1	127.0.0.1	TCP	66	57800 → 9898 [ACK] Seq=101 Ack=12 Win=65536 Len=0 TSval=19841...

PRASHANT

2018360

Below are the answers of some questions

Answers:

- a) Only 1 TCP connection are made
- b) Server PORTNO.: 9898
Client PORTNO.: 57800
- c) Total packet exchange between client & server: 335
- d) Total time taken for download the file (in sec): 8.419

I have also attached the packet.csv file in the submitted folder, which contains the packet dissection data from the Wireshark Analysis