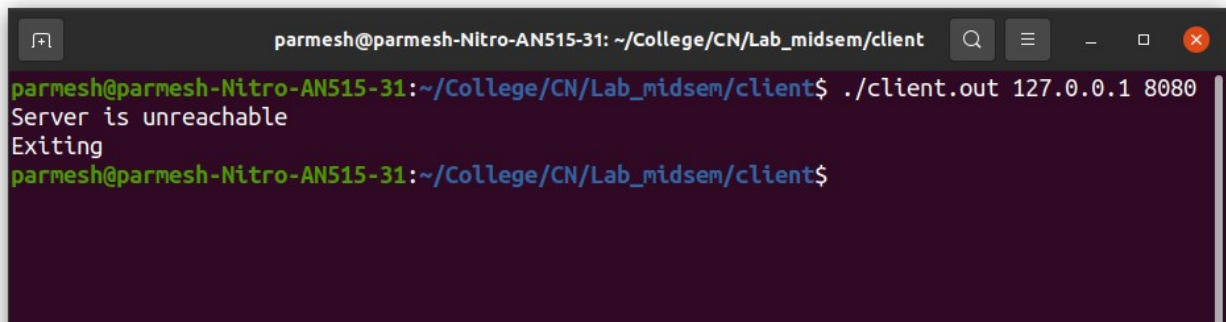


Lab Midsem

Parmesh Mathur

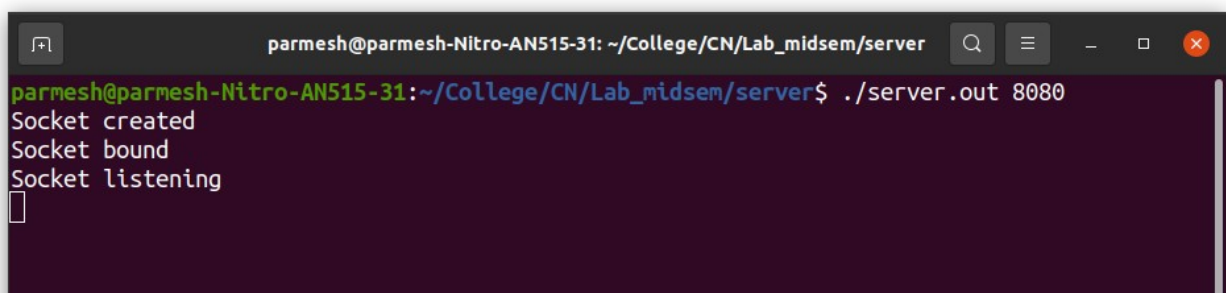
2018A7PS0133G

When the server is not running, the client does not connect to it and exits immediately.

A terminal window titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client'. The prompt is 'parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/client\$'. The user enters './client.out 127.0.0.1 8080'. The output is 'Server is unreachable' followed by 'Exiting' on the next line. The prompt returns.

```
parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client
parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Server is unreachable
Exiting
parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/client$
```

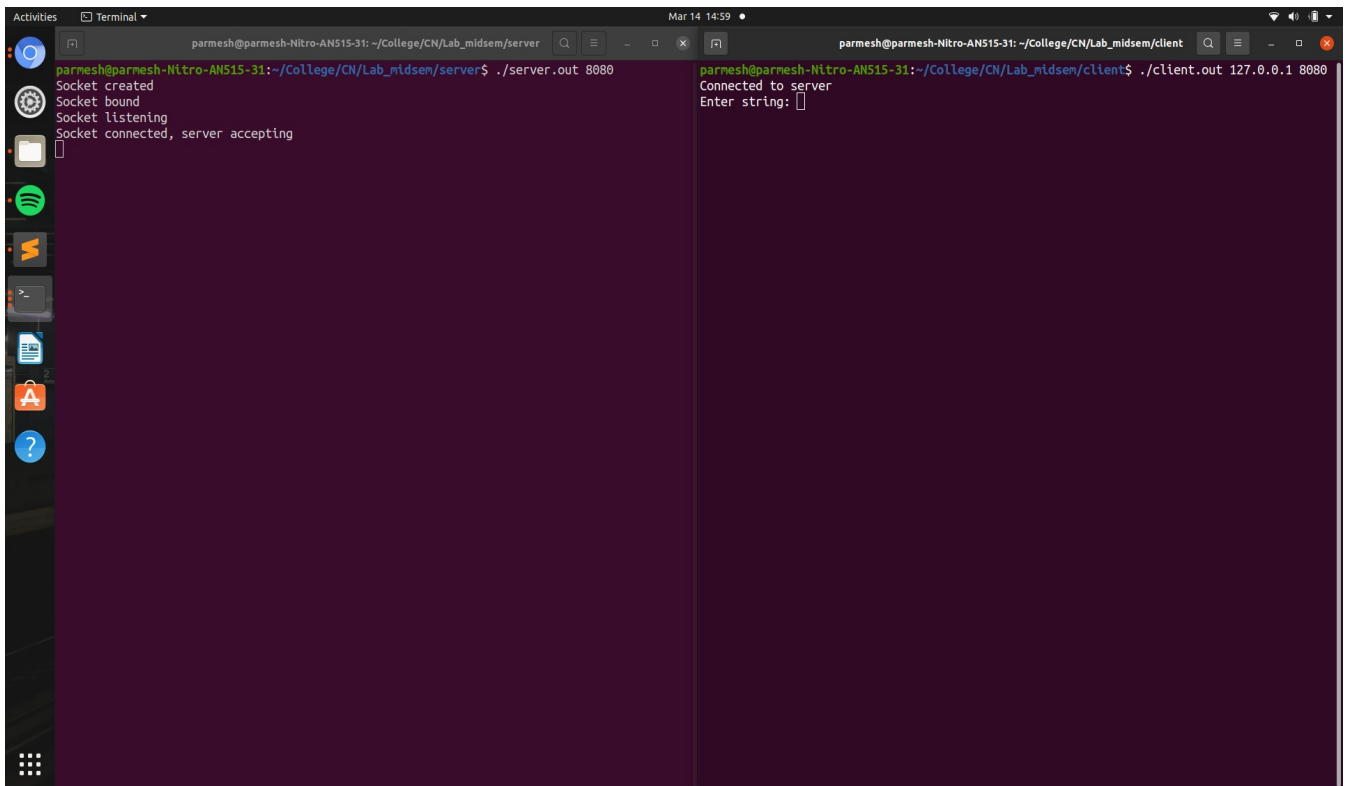
The server will create a socket and listens to it when it is executed. It is then ready to communicate with an incoming client.

A terminal window titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server'. The prompt is 'parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/server\$'. The user enters './server.out 8080'. The output is 'Socket created', 'Socket bound', and 'Socket listening' on three separate lines. A cursor is visible on the line following the last output.

```
parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server
parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/server$ ./server.out 8080
Socket created
Socket bound
Socket listening
█
```

P.T.O

When the client is executed while the server is running, the socket is connected and it is ready to receive a string from the user (stdin).



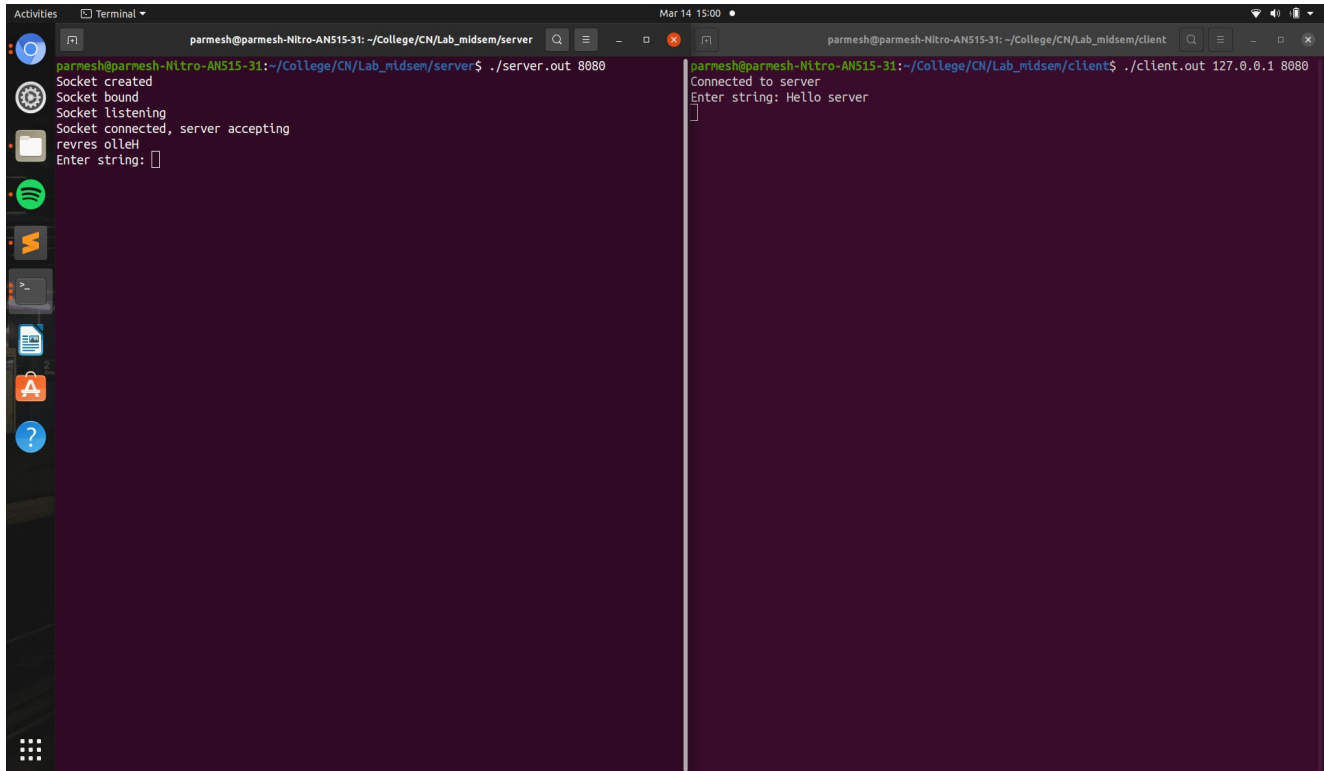
The image shows two terminal windows side-by-side. The left window is titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server' and shows the execution of './server.out 8080'. The output is: 'Socket created', 'Socket bound', 'Socket listening', and 'Socket connected, server accepting'. The right window is titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client' and shows the execution of './client.out 127.0.0.1 8080'. The output is: 'Connected to server' and 'Enter string: ' with a cursor. The desktop background is dark purple with a sidebar on the left containing various application icons.

```
parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server$ ./server.out 8080
Socket created
Socket bound
Socket listening
Socket connected, server accepting

parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: 
```

P.T.O.

Once the string is sent to the server, it displays it in reverse order to the user (stdout). The server is now ready to receive its string input from stdin.



The image shows two terminal windows side-by-side. The left window is titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server' and shows the execution of './server.out 8080'. The output indicates that the socket was created, bound, listening, and then a connection was accepted. It prompts 'Enter string: ' with a cursor. The right window is titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client' and shows the execution of './client.out 127.0.0.1 8080'. The output shows 'Connected to server' and 'Enter string: Hello server', followed by a blank line where the reversed string would be displayed.

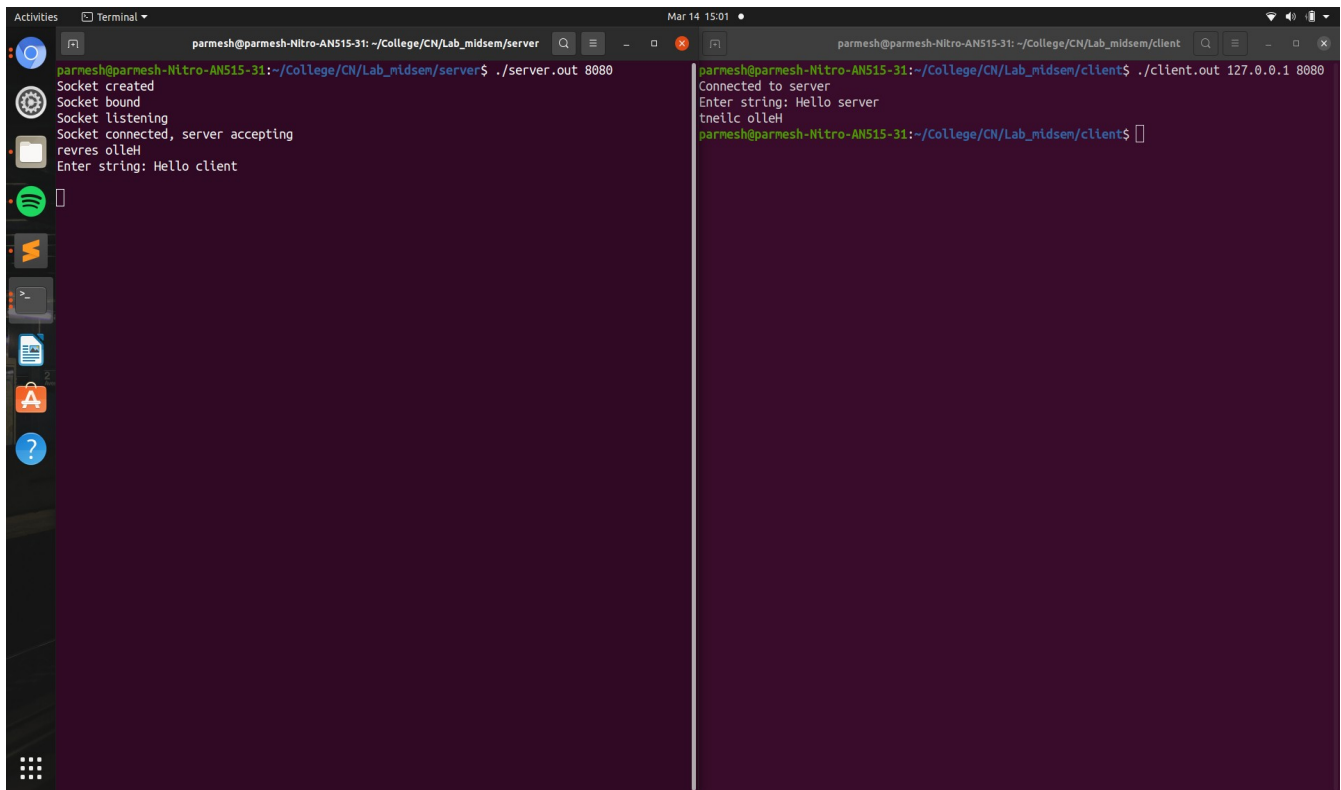
```
parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server$ ./server.out 8080
Socket created
Socket bound
Socket listening
Socket connected, server accepting
revres olleH
Enter string: 
```

```
parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Hello server

```

P.T.O.

Once the server receives the string from the user, it is sent to the client. The client then reverses the string to display it and then exits.



The image shows two terminal windows side-by-side. The left window is titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/server' and shows the execution of a server program. The right window is titled 'parmesh@parmesh-Nitro-AN515-31: ~/College/CN/Lab_midsem/client' and shows the execution of a client program.

```
parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/server$ ./server.out 8080
Socket created
Socket bound
Socket listening
Socket connected, server accepting
revres olleH
Enter string: Hello client
```

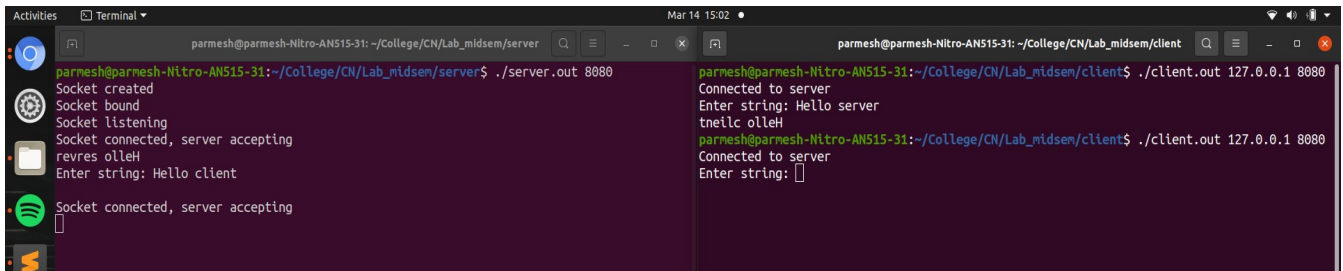
```
parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Hello server
tnelc olleH
parmesh@parmesh-Nitro-AN515-31:~/College/CN/Lab_midsem/client$
```

Meanwhile, the server is still running and is now ready to accept another client process.

P.T.O.

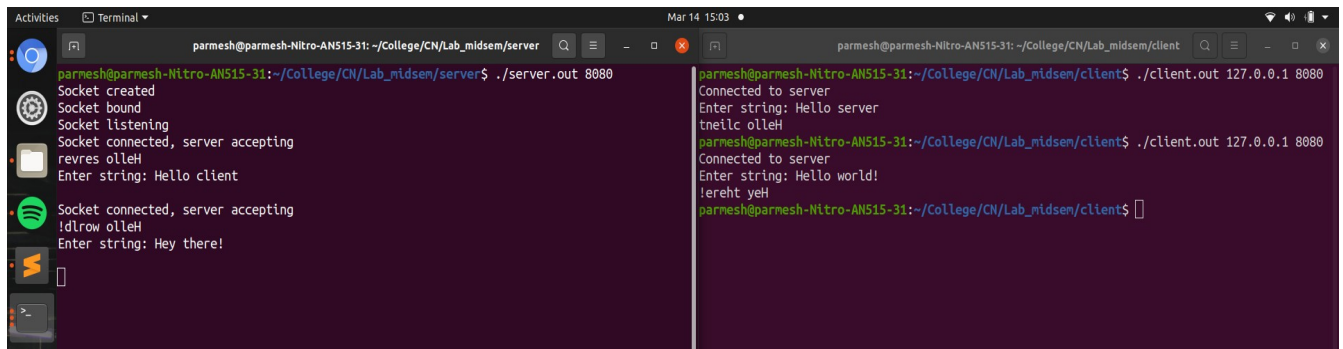
When another client comes in consecutively, the whole process of sending and receiving messages repeats until the client exits again.

The client connects to the running server process.



The screenshot shows two terminal windows side-by-side. The left window is titled 'parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/server' and shows the server process running. It starts with 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/server\$./server.out 8080'. The output shows: 'Socket created', 'Socket bound', 'Socket listening', 'Socket connected, server accepting', 'revres olleH', 'Enter string: Hello client', and 'Socket connected, server accepting'. The right window is titled 'parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/client' and shows the client process running. It starts with 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client\$./client.out 127.0.0.1 8080'. The output shows: 'Connected to server', 'Enter string: Hello server', 'tneilc olleH', and 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client\$./client.out 127.0.0.1 8080'. The client then shows 'Connected to server' and 'Enter string: '.

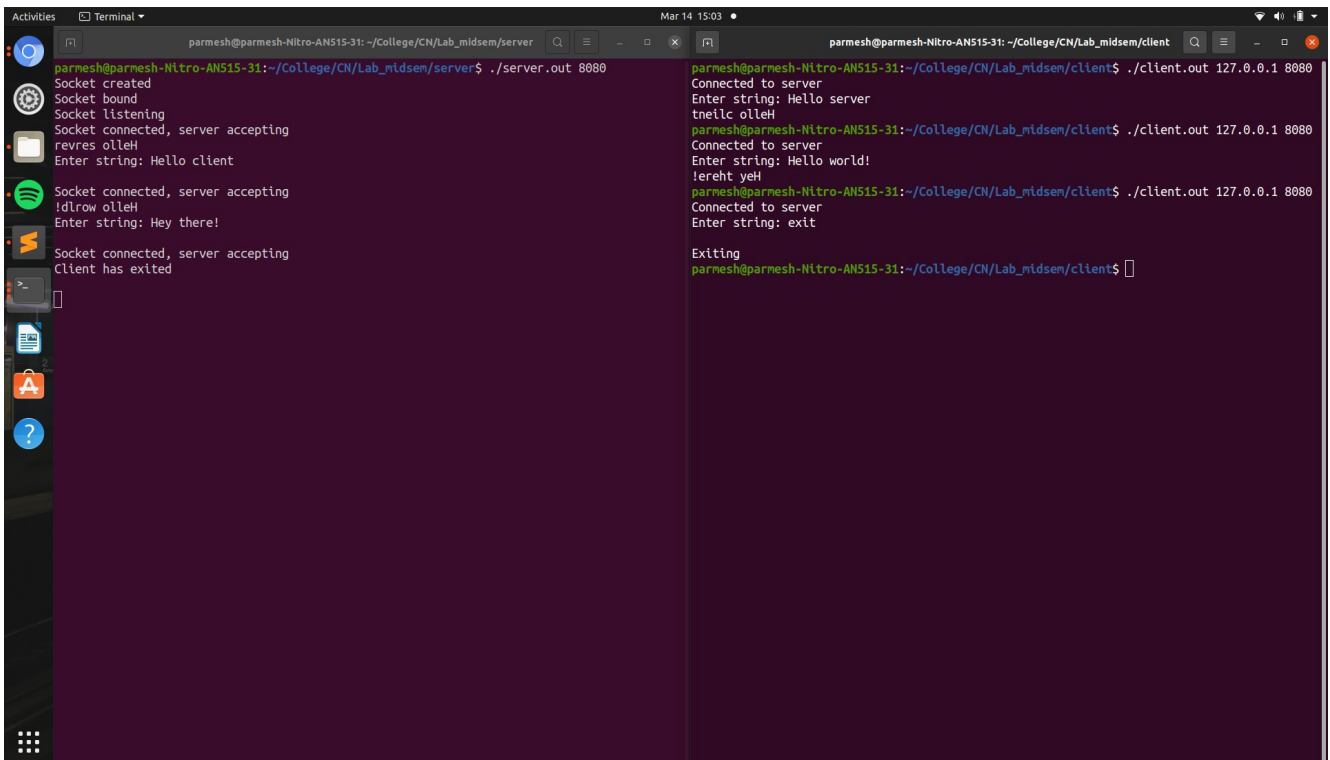
The client and server exchange messages, and then the client exits again.



The screenshot shows two terminal windows side-by-side. The left window is titled 'parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/server' and shows the server process running. It starts with 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/server\$./server.out 8080'. The output shows: 'Socket created', 'Socket bound', 'Socket listening', 'Socket connected, server accepting', 'revres olleH', 'Enter string: Hello client', 'Socket connected, server accepting', 'ldrow olleH', and 'Enter string: Hey there!'. The right window is titled 'parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/client' and shows the client process running. It starts with 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client\$./client.out 127.0.0.1 8080'. The output shows: 'Connected to server', 'Enter string: Hello server', 'tneilc olleH', 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client\$./client.out 127.0.0.1 8080', 'Connected to server', 'Enter string: Hello world!', 'lereht yeH', and 'parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client\$ '.

The server is again ready to accept another client.

After the client connects, if the user enters the exit string (“exit”), the client will exit immediately.



```
parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/server$ ./server.out 8080
Socket created
Socket bound
Socket listening
Socket connected, server accepting
revres olleH
Enter string: Hello client
Socket connected, server accepting
!dlrow olleH
Enter string: Hey there!
Socket connected, server accepting
Client has exited

parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Hello server
tnelc olleH
parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Hello world!
!ereht yeH
parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: exit
Exiting
parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/client$
```

The server continues running waiting for the next client.

P.T.O.

When it is the server's turn to send the message, if the user inputs the exit string, the server exits immediately. Consequently the client also exits as it can't run without a server.

```
parmesh@parmesh-Nitro-ANS15-31: ~/College/CN/Lab_midsem/server$ ./server.out 8080
Socket created
Socket bound
Socket listening
Socket connected, server accepting
revres olleH
Enter string: Hello client
Socket connected, server accepting
!dlrow olleH
Enter string: Hey there!
Socket connected, server accepting
Client has exited
Socket connected, server accepting
egassen tseT
Enter string: exit
Exiting
parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/server$

parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Hello server
tnellec olleH
parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Hello world!
!ereht yeH
parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: exit
Exiting
parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client$ ./client.out 127.0.0.1 8080
Connected to server
Enter string: Test message
parmesh@parmesh-Nitro-ANS15-31:~/College/CN/Lab_midsem/client$
```

P.T.O.

Using Wireshark

All the packets between the two processes were captured on Wireshark on the

As the file might contain some non-tcp packets, the filter used to obtain only the packets transactions between the two processes was:

```
ip.addr==127.0.0.1 && tcp.port==8080
```

This filter is used to make sure that only the messages on the specified IP address and port number are shown. It also ensures that packets are only shown if the protocol being used is TCP.

2018A7P50133G_Lab_midsem.pcap

FileEditViewGoCaptureAnalyzeStatisticsTelephonyWirelessToolsHelp

ip.addr == 127.0.0.1 && tcp.port==8080

Time	Source IP	Source port	Dest IP	Dest port	Protocol	TCP Segment Len	Length	Info
2021-03-14 10:57:48.634417	127.0.0.1	37916	127.0.0.1	8080	TCP	0	74	37916 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65...
2021-03-14 10:57:48.634445	127.0.0.1	8080	127.0.0.1	37916	TCP	0	74	8080 → 37916 [SYN, ACK] Seq=0 Ack=1 Win=65483 L...
2021-03-14 10:57:48.634468	127.0.0.1	37916	127.0.0.1	8080	TCP	0	66	37916 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 ...
2021-03-14 10:57:55.299924	127.0.0.1	37916	127.0.0.1	8080	TCP	12	78	37916 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 L...
2021-03-14 10:57:55.299984	127.0.0.1	8080	127.0.0.1	37916	TCP	0	66	8080 → 37916 [ACK] Seq=1 Ack=13 Win=65536 Len=0...
2021-03-14 10:58:01.045336	127.0.0.1	8080	127.0.0.1	37916	TCP	12	78	8080 → 37916 [PSH, ACK] Seq=1 Ack=13 Win=65536 ...
2021-03-14 10:58:01.045379	127.0.0.1	37916	127.0.0.1	8080	TCP	0	66	37916 → 8080 [ACK] Seq=13 Ack=13 Win=65536 Len=...
2021-03-14 10:58:01.045428	127.0.0.1	8080	127.0.0.1	37916	TCP	0	66	8080 → 37916 [FIN, ACK] Seq=13 Ack=13 Win=65536...
2021-03-14 10:58:01.045521	127.0.0.1	37916	127.0.0.1	8080	TCP	0	66	37916 → 8080 [FIN, ACK] Seq=13 Ack=14 Win=65536...
2021-03-14 10:58:01.045543	127.0.0.1	8080	127.0.0.1	37916	TCP	0	66	8080 → 37916 [ACK] Seq=14 Ack=14 Win=65536 Len=...
2021-03-14 10:58:03.941594	127.0.0.1	37918	127.0.0.1	8080	TCP	0	74	37918 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65...
2021-03-14 10:58:03.941627	127.0.0.1	8080	127.0.0.1	37918	TCP	0	74	8080 → 37918 [SYN, ACK] Seq=0 Ack=1 Win=65483 L...
2021-03-14 10:58:03.941657	127.0.0.1	37918	127.0.0.1	8080	TCP	0	66	37918 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 ...
2021-03-14 10:58:10.061715	127.0.0.1	37918	127.0.0.1	8080	TCP	12	78	37918 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 L...
2021-03-14 10:58:10.061765	127.0.0.1	8080	127.0.0.1	37918	TCP	0	66	8080 → 37918 [ACK] Seq=1 Ack=13 Win=65536 Len=0...
2021-03-14 10:58:19.048226	127.0.0.1	8080	127.0.0.1	37918	TCP	10	76	8080 → 37918 [PSH, ACK] Seq=1 Ack=13 Win=65536 ...
2021-03-14 10:58:19.048275	127.0.0.1	37918	127.0.0.1	8080	TCP	0	66	37918 → 8080 [ACK] Seq=13 Ack=11 Win=65536 Len=...
2021-03-14 10:58:19.048348	127.0.0.1	8080	127.0.0.1	37918	TCP	0	66	8080 → 37918 [FIN, ACK] Seq=11 Ack=13 Win=65536...
2021-03-14 10:58:19.048426	127.0.0.1	37918	127.0.0.1	8080	TCP	0	66	37918 → 8080 [FIN, ACK] Seq=13 Ack=12 Win=65536...
2021-03-14 10:58:19.048443	127.0.0.1	8080	127.0.0.1	37918	TCP	0	66	8080 → 37918 [ACK] Seq=12 Ack=14 Win=65536 Len=...
2021-03-14 10:58:21.661286	127.0.0.1	37920	127.0.0.1	8080	TCP	0	74	37920 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65...
2021-03-14 10:58:21.661312	127.0.0.1	8080	127.0.0.1	37920	TCP	0	74	8080 → 37920 [SYN, ACK] Seq=0 Ack=1 Win=65483 L...
2021-03-14 10:58:21.661334	127.0.0.1	37920	127.0.0.1	8080	TCP	0	66	37920 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 ...
2021-03-14 10:58:26.439968	127.0.0.1	37920	127.0.0.1	8080	TCP	4	70	37920 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 L...
2021-03-14 10:58:26.440015	127.0.0.1	8080	127.0.0.1	37920	TCP	0	66	8080 → 37920 [ACK] Seq=1 Ack=5 Win=65536 Len=0 ...
2021-03-14 10:58:26.440078	127.0.0.1	37920	127.0.0.1	8080	TCP	0	66	37920 → 8080 [FIN, ACK] Seq=5 Ack=1 Win=65536 L...
2021-03-14 10:58:26.440188	127.0.0.1	8080	127.0.0.1	37920	TCP	0	66	8080 → 37920 [FIN, ACK] Seq=1 Ack=6 Win=65536 L...
2021-03-14 10:58:26.440266	127.0.0.1	37920	127.0.0.1	8080	TCP	0	66	37920 → 8080 [ACK] Seq=6 Ack=2 Win=65536 Len=0 ...
2021-03-14 10:58:28.737619	127.0.0.1	37922	127.0.0.1	8080	TCP	0	74	37922 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65...
2021-03-14 10:58:28.737652	127.0.0.1	8080	127.0.0.1	37922	TCP	0	74	8080 → 37922 [SYN, ACK] Seq=0 Ack=1 Win=65483 L...
2021-03-14 10:58:28.737680	127.0.0.1	37922	127.0.0.1	8080	TCP	0	66	37922 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 ...
2021-03-14 10:58:33.358434	127.0.0.1	37922	127.0.0.1	8080	TCP	12	78	37922 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 L...
2021-03-14 10:58:33.358478	127.0.0.1	8080	127.0.0.1	37922	TCP	0	66	8080 → 37922 [ACK] Seq=1 Ack=13 Win=65536 Len=0...
2021-03-14 10:58:37.802667	127.0.0.1	8080	127.0.0.1	37922	TCP	0	66	8080 → 37922 [FIN, ACK] Seq=1 Ack=13 Win=65536 ...
2021-03-14 10:58:37.802271	127.0.0.1	37922	127.0.0.1	8080	TCP	0	66	37922 → 8080 [FIN, ACK] Seq=13 Ack=2 Win=65536 ...
2021-03-14 10:58:37.802302	127.0.0.1	8080	127.0.0.1	37922	TCP	0	66	8080 → 37922 [ACK] Seq=2 Ack=14 Win=65536 Len=0...

▶ Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)

▶ 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: 37916, Dst Port: 8080, Seq: 0, Len: 0

0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00E

0010 00 3c 0e 8e 40 00 00 06 2e 2c 7f 00 00 01 7f 00 <...@.....

0020 00 01 94 1c 1f 90 27 61 b7 dc 00 00 00 00 a0 02a.....

0030 ff d7 fe 30 00 00 02 04 ff d7 04 02 08 0a 55 e5 ...0.....U..

0040 97 86 00 00 00 00 01 03 03 07

2018A7P50133G_Lab_midsem.pcapPackets: 40 - Displayed: 36 (90.0%)Profile: Default

To further filter out packets that were sent specifically from the client to the server and contained messages, we use the previous filter and extend it to:

```
ip.addr==127.0.0.1 && tcp.dstport==8080 && tcp.len>0
```

The image shows a Wireshark packet capture window titled "2018A7PS0133G_Lab_midsem.pcap". The filter bar at the top contains the filter: `ip.addr==127.0.0.1 && tcp.dstport==8080 && tcp.len>0`. The packet list shows four packets, all of which are TCP segments from 127.0.0.1 to 127.0.0.1 on port 8080. The selected packet (Frame 6) is a TCP segment with sequence number 37916, acknowledgment number 1, and a length of 12 bytes. The packet details pane shows the following structure:

- Frame 6: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
- 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: 37916, Dst Port: 8080, Seq: 1, Ack: 1, Len: 12

The packet bytes pane shows the raw data of the packet, which is the string "Hello server".

This filters out messages that do not contain any messages, as they have TCP Segment lengths of zero. We also observe that the TCP segment length is the same as the length of the message being sent (as shown with example above where message being sent is 12 characters [Hello server] and the TCP segment Length is also 12 bytes).

The difference in size of the TCP segment length and the packet length (displayed under the column 'length' in the window) of 66 bytes is due to the headers used by each protocol while sending packets, regardless of whether the message contains any data in it at the transport layer level.

Here 32 bytes is contributed by the TCP layer header. The remaining 34 bytes are contributed by the layers below it:

- 20 bytes in the network layer (by the IP protocol).
- 14 bytes in the data link and physical layers (by the Ethernet protocol).

Refer to the README file to view instructions on how to run the processes.