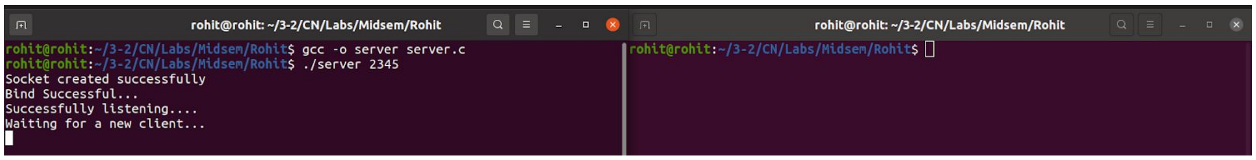
**ROHIT GARG**
**2018A7PS0193G**

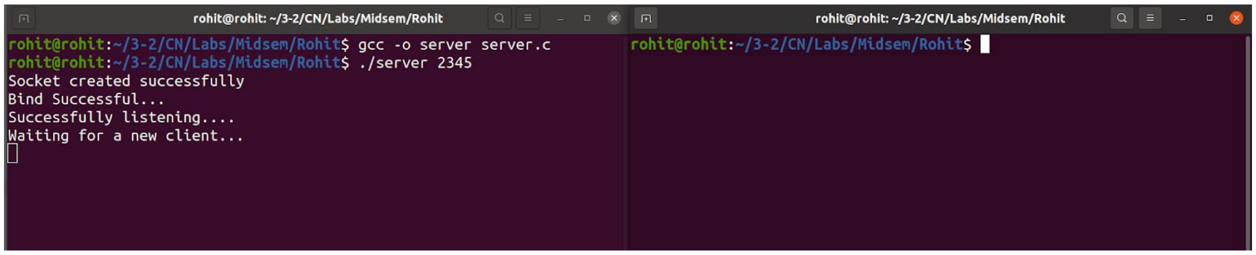
# COMPUTER NETWORKS
# CS F303
# MID-SEMESTER LAB

**1. The server as a command line argument accepts the port number to which it should bind.**



```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
```

```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```

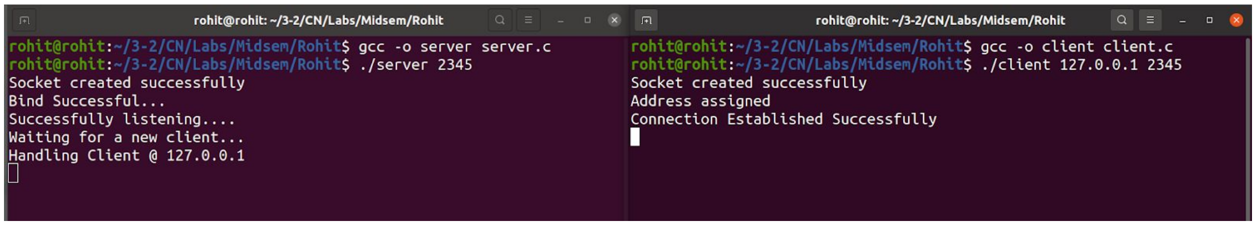


```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
```

```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```

The screenshot shows that server accepts a valid port number as command line argument given to it i.e. 2345. Server creates a new socket, binds the socket to the server's IP address and listens for any connection request in passive mode.

**2. The client, as command line arguments, accepts the IP address and the port number at which it will find the server.**

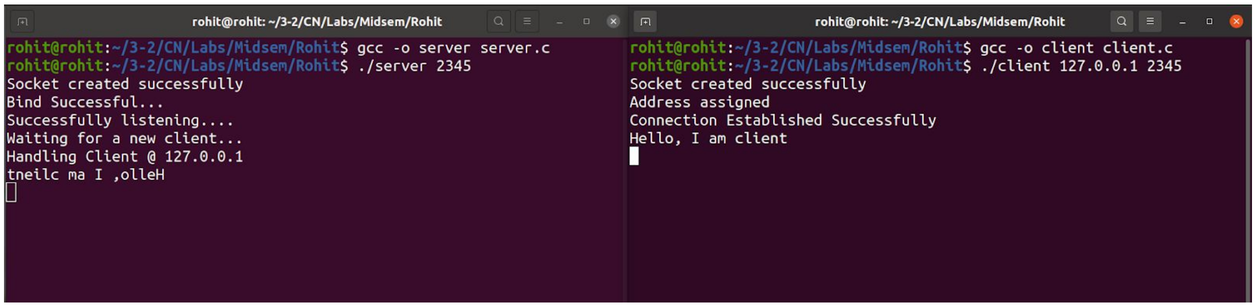


```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
```

```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
```

The screenshot below shows that client accepts server IP address and server port number as command-line arguments. Then, a client creates a socket and then sends a connection request to server. If the server accepts the connection, connection is successful and server waits for client to send it a string that will be stdin input to client.

**3. After connecting to the server, the client reads a line from the standard input and sends it to the server. (2 marks)**

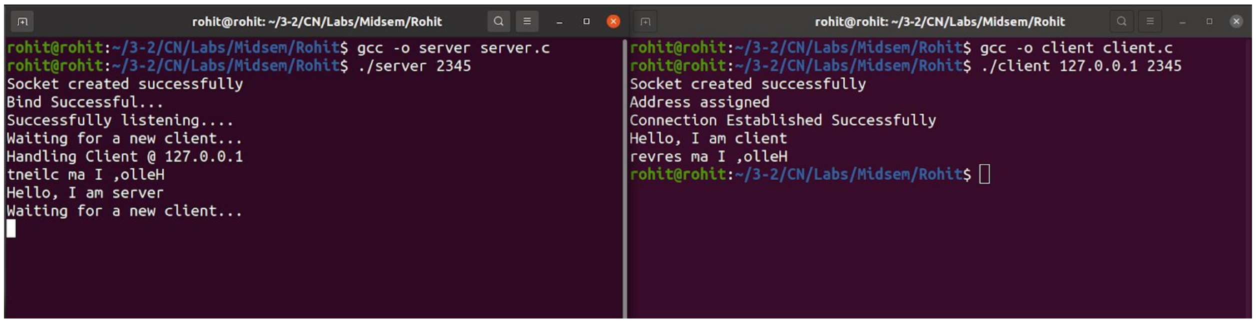


```
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
tneilc ma I ,olleH
```

```
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
Hello, I am client
```

Client reads line “Hello, I am client” from stdin input on client’s terminal, sends the line to the server. The server receives the line and prints it in reverse manner i.e. “tneilc ma I ,olleH” on server’s terminal.

**4. The server prints the received line in the reverser order (2 marks) and reads a line from the standard input and sends it to the client. (2 marks)**

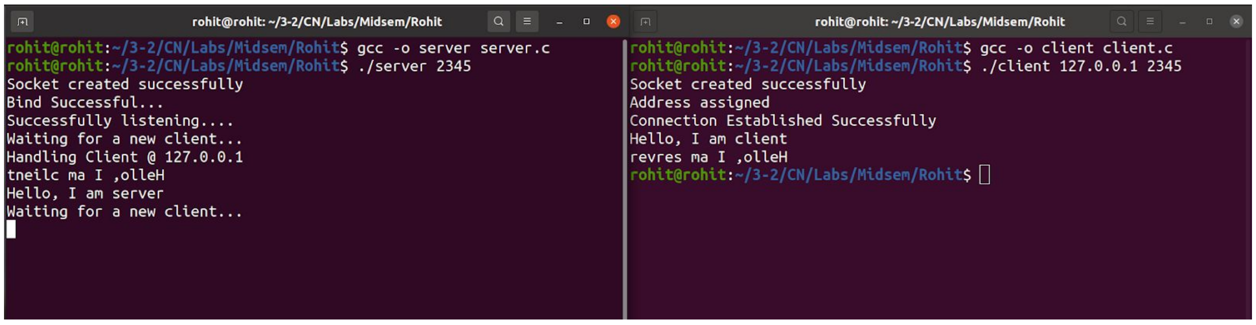


```
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
tneilc ma I ,olleH
Hello, I am server
Waiting for a new client...
```

```
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
Hello, I am client
revres ma I ,olleH
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```

The server receives the line “Hello, I am client” from client and prints it in reverse manner i.e. “tneilc ma I ,olleH” on server’s terminal. Server then reads a line from the standard input “Hello, I am server” and sends it to the client.

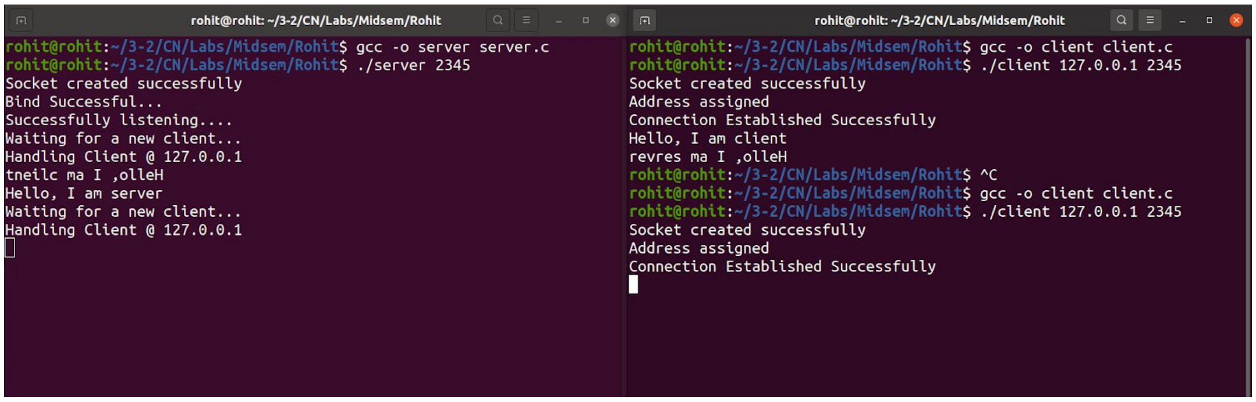
**5. The client prints the received line in the reverse order and exits. The server is ready to accept a new client. (2 marks)**



```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
tneilc ma I ,olleH
Hello, I am server
Waiting for a new client...
```

```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
Hello, I am client
revres ma I ,olleH
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```

The client receives the line “Hello, I am server”  sent by the server and prints it in reverse order as “revres ma I ,olleH” and then exits. However, server still runs and waits for a new client to connect to it.



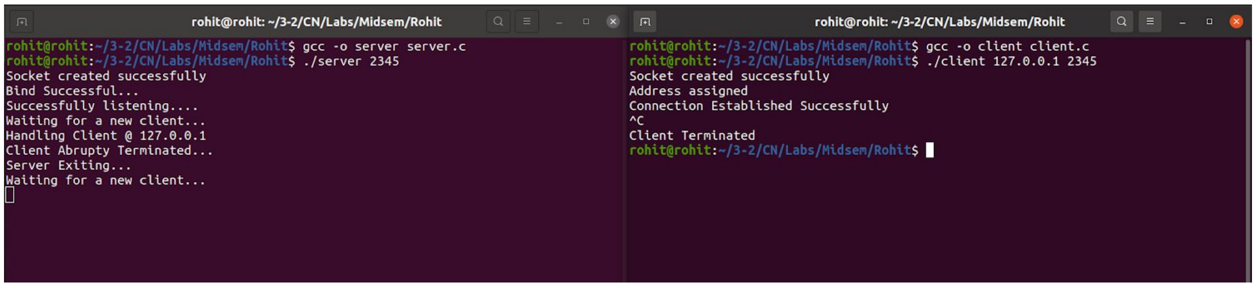
```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
tneilc ma I ,olleH
Hello, I am server
Waiting for a new client...
Handling Client @ 127.0.0.1
```

```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
Hello, I am client
revres ma I ,olleH
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ^C
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
```

This screenshot shows that another client can successfully connect to the server to start a new session. A similar procedure will follow.

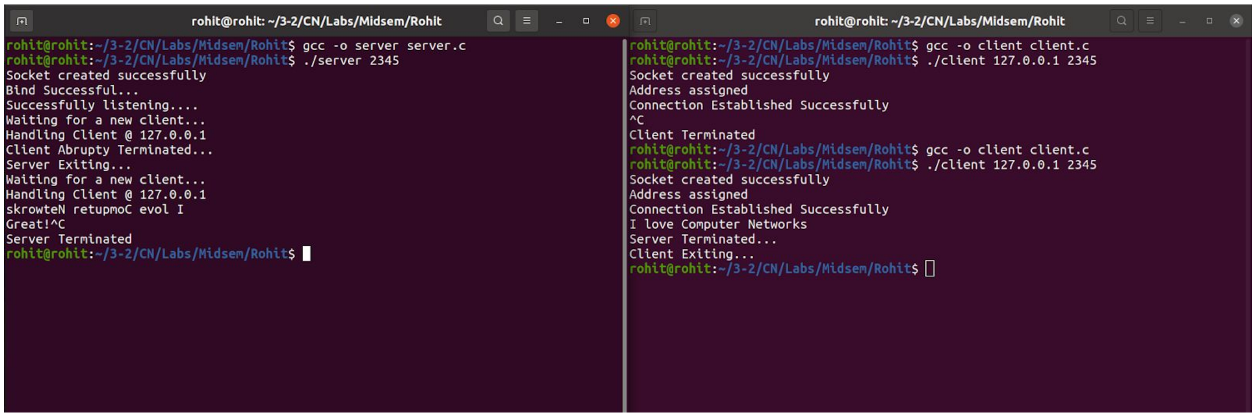
**Handling Abrupt Termination**

The two screenshots below show how the code handles ctrl+C during a session. If during a session, the client side has a SIGINT, then it is handled so that the server side doesn’t hang and continues normally. Similarly if the server side has a SIGINT, then it handles it so that the client side doesn’t hang and exits normally.



```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
Client Abrupty Terminated...
Server Exiting...
Waiting for a new client...
```

```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
^C
Client Terminated
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```



```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
Client Abrupty Terminated...
Server Exiting...
Waiting for a new client...
Handling Client @ 127.0.0.1
skrowteN retupmoC evol I
Great!^C
Server Terminated
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```

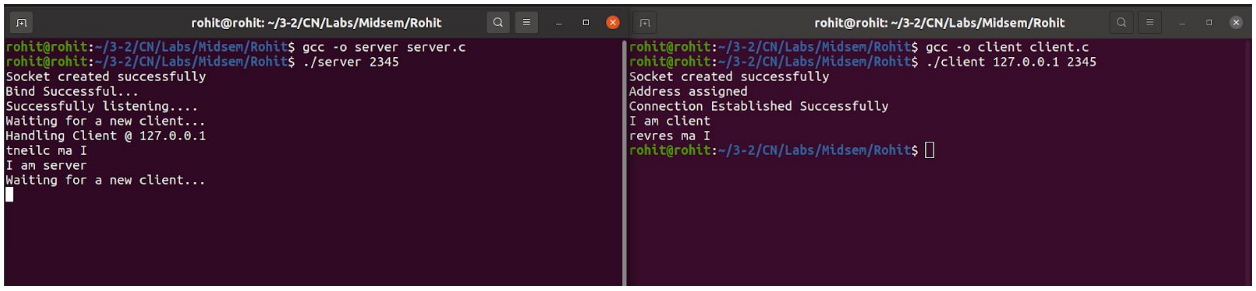
```
rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
^C
Client Terminated
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
I love Computer Networks
Server Terminated...
Client Exiting...
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```

**6. Use Wireshark to capture the packets (2 marks) and write a filter that will find what were the sizes of the TCP segment sent from the client to the server. (2 marks) Compare the number of bytes in the line with those observed in the TCP segment. Justify how the values match. (2 marks)**

Start capturing on Loopback (Lo) interface i.e. interface on localhost. Filter results by using "ip.dst==127.0.0.1 && tcp.dstport==2345" which is the server IP and server port . The TCP segment lengths are displayed using the "tcp.len" filter which is applied as a column. Applying an and on these filters shows desired results.

The number of bytes in the TCP segment length are the same as the number of characters for the message – The message from client to server was "I am client" which has 11 characters . And the same is reflected in the size of the TCP segment. The segment length is 11 bytes.

The TCP header length for this packet is 32 bytes. The IP header length is 20 bytes. The total length = TCP Segment Length + TCP header + IP header lengths = 11 + 32 + 20 = 63 bytes. We see all messages sent from client[Port No. 51634] to server[Port No. 2345], including the first SYN, and the 3-way handshake ACK. Followed by the data segment and the FIN, indicating client leaving, also the FIN, ACK showing acknowledgement of the client to server.

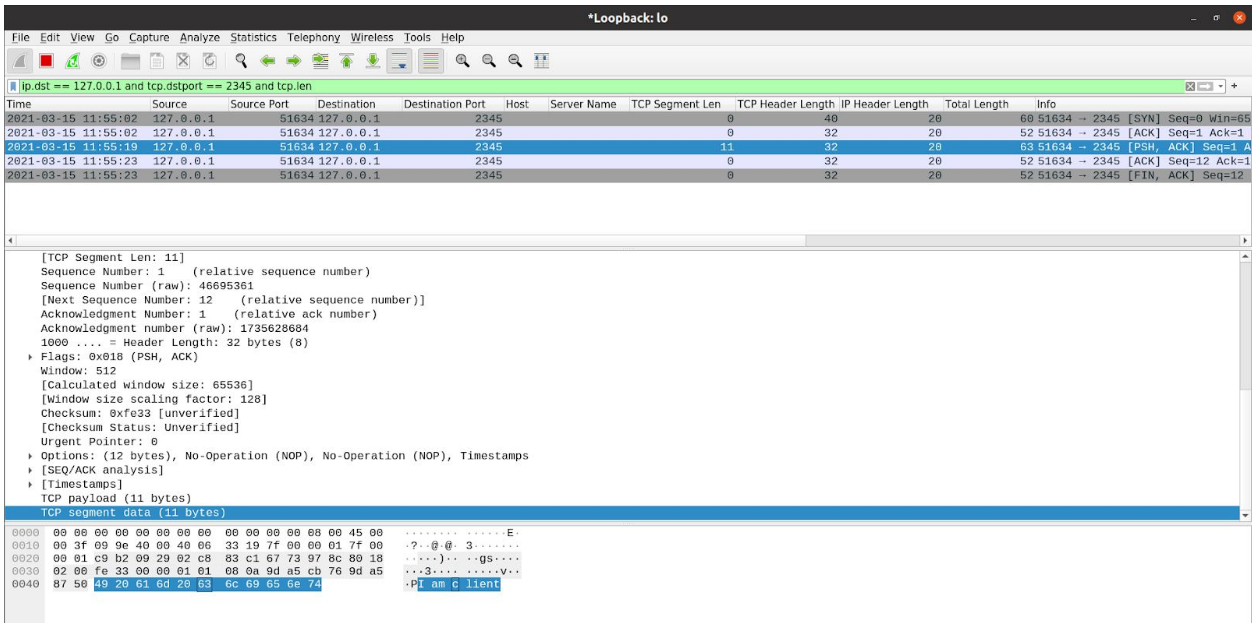


rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit

```
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o server server.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./server 2345
Socket created successfully
Bind Successful...
Successfully listening....
Waiting for a new client...
Handling Client @ 127.0.0.1
tneilc ma I
I am server
Waiting for a new client...
```

rohit@rohit: ~/3-2/CN/Labs/Midsem/Rohit

```
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ gcc -o client client.c
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$ ./client 127.0.0.1 2345
Socket created successfully
Address assigned
Connection Established Successfully
I am client
revres ma I
rohit@rohit:~/3-2/CN/Labs/Midsem/Rohit$
```



*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

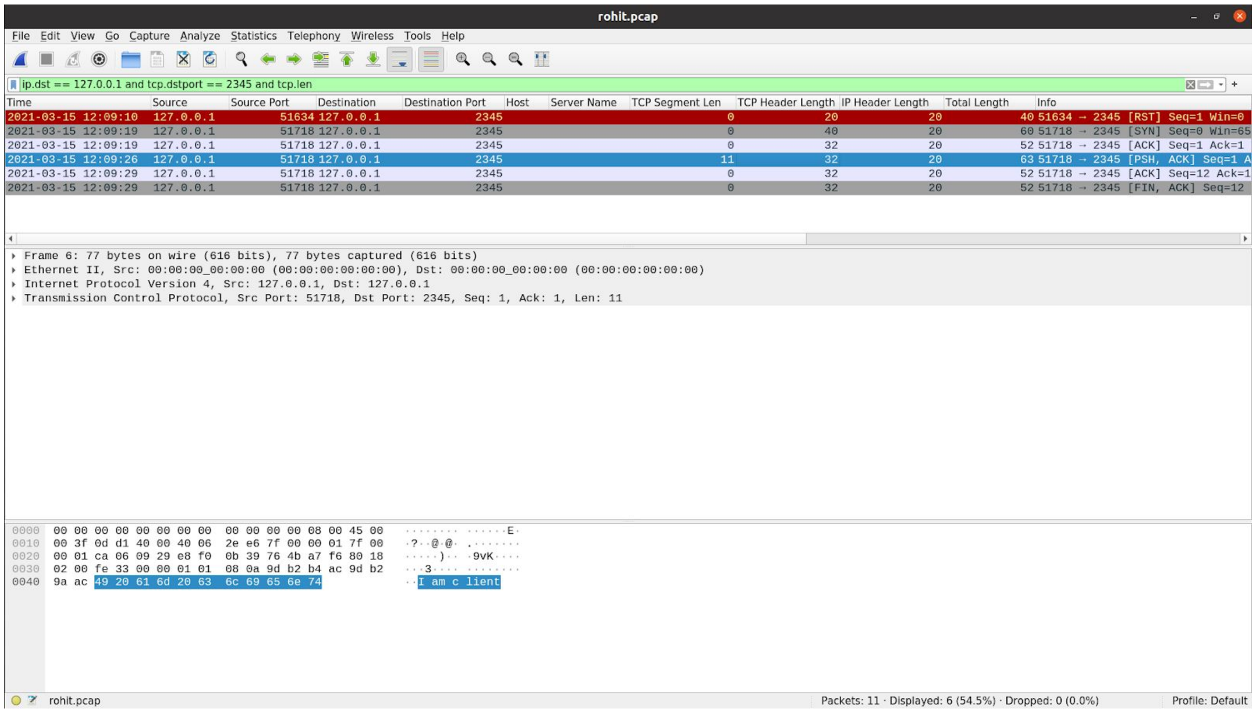
ip.dst == 127.0.0.1 and tcp.dstport == 2345 and tcp.len

| Time | Source | Source Port | Destination | Destination Port | Host | Server Name | TCP Segment Len | TCP Header Length | IP Header Length | Total Length | Info |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-03-15 11:55:02 | 127.0.0.1 | 51634 | 127.0.0.1 | 2345 | | | 0 | 40 | 20 | 60 | 51634 → 2345 [SYN] Seq=0 Win=65 |
| 2021-03-15 11:55:02 | 127.0.0.1 | 51634 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 51634 → 2345 [ACK] Seq=1 Ack=1 |
| 2021-03-15 11:55:19 | 127.0.0.1 | 51634 | 127.0.0.1 | 2345 | | | 11 | 32 | 20 | 63 | 51634 → 2345 [PSH, ACK] Seq=1 A |
| 2021-03-15 11:55:23 | 127.0.0.1 | 51634 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 51634 → 2345 [ACK] Seq=12 Ack=1 |
| 2021-03-15 11:55:23 | 127.0.0.1 | 51634 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 51634 → 2345 [FIN, ACK] Seq=12 |

```
    [TCP Segment Len: 11]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 46695361
    [Next Sequence Number: 12    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1735628684
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0xfe33 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (11 bytes)
    TCP segment data (11 bytes)
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 3f 09 9e 40 00 40 06  33 19 7f 00 00 01 7f 00   ·?··@·@· 3·······
0020  00 01 c9 b2 09 29 02 c8  83 c1 67 73 97 8c 80 18   ·····)·· ··gs····
0030  02 00 fe 33 00 00 01 01  08 0a 9d a5 cb 76 9d a5   ···3···· ·····v··
0040  87 50 49 20 61 6d 20 63  6c 69 65 6e 74            ·PI am c lient
```

**The captured packets are exported as rohit.pcap**


rohit.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.dst == 127.0.0.1 and tcp.dstport == 2345 and tcp.len

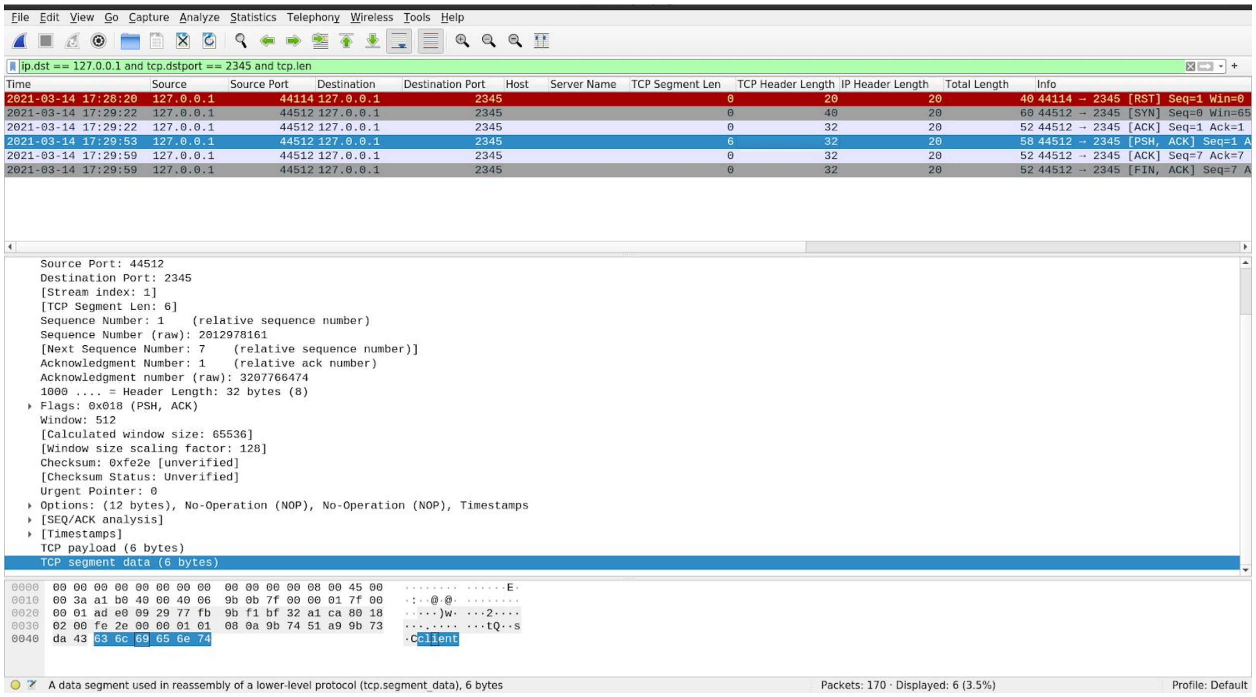
| Time | Source | Source Port | Destination | Destination Port | Host | Server Name | TCP Segment Len | TCP Header Length | IP Header Length | Total Length | Info |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-03-15 12:09:10 | 127.0.0.1 | 51634 | 127.0.0.1 | 2345 | | | 0 | 20 | 20 | 40 | 51634 → 2345 [RST] Seq=1 Win=0 |
| 2021-03-15 12:09:19 | 127.0.0.1 | 51718 | 127.0.0.1 | 2345 | | | 0 | 40 | 20 | 60 | 51718 → 2345 [SYN] Seq=0 Win=65 |
| 2021-03-15 12:09:19 | 127.0.0.1 | 51718 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 51718 → 2345 [ACK] Seq=1 Ack=1 |
| 2021-03-15 12:09:26 | 127.0.0.1 | 51718 | 127.0.0.1 | 2345 | | | 11 | 32 | 20 | 63 | 51718 → 2345 [PSH, ACK] Seq=1 A |
| 2021-03-15 12:09:29 | 127.0.0.1 | 51718 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 51718 → 2345 [ACK] Seq=12 Ack=1 |
| 2021-03-15 12:09:29 | 127.0.0.1 | 51718 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 51718 → 2345 [FIN, ACK] Seq=12 |

```
Frame 6: 77 bytes on wire (616 bits), 77 bytes captured (616 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: 51718, Dst Port: 2345, Seq: 1, Ack: 1, Len: 11
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 3f 0d d1 40 00 40 06  2e e6 7f 00 00 01 7f 00   ·?··@·@· .·······
0020  00 01 ca 06 09 29 e8 f0  0b 39 76 4b a7 f6 80 18   ·····)·· ·9vK····
0030  02 00 fe 33 00 00 01 01  08 0a 9d b2 b4 ac 9d b2   ···3···· ········
0040  9a ac 49 20 61 6d 20 63  6c 69 65 6e 74            ··I am c lient
```

rohit.pcap  Packets: 11 · Displayed: 6 (54.5%) · Dropped: 0 (0.0%)  Profile: Default

Another example of a message “client” sent by the client program. Clearly the client’s message is 6 chars in size so is the size of tcp segment length/data i.e. 6 bytes.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

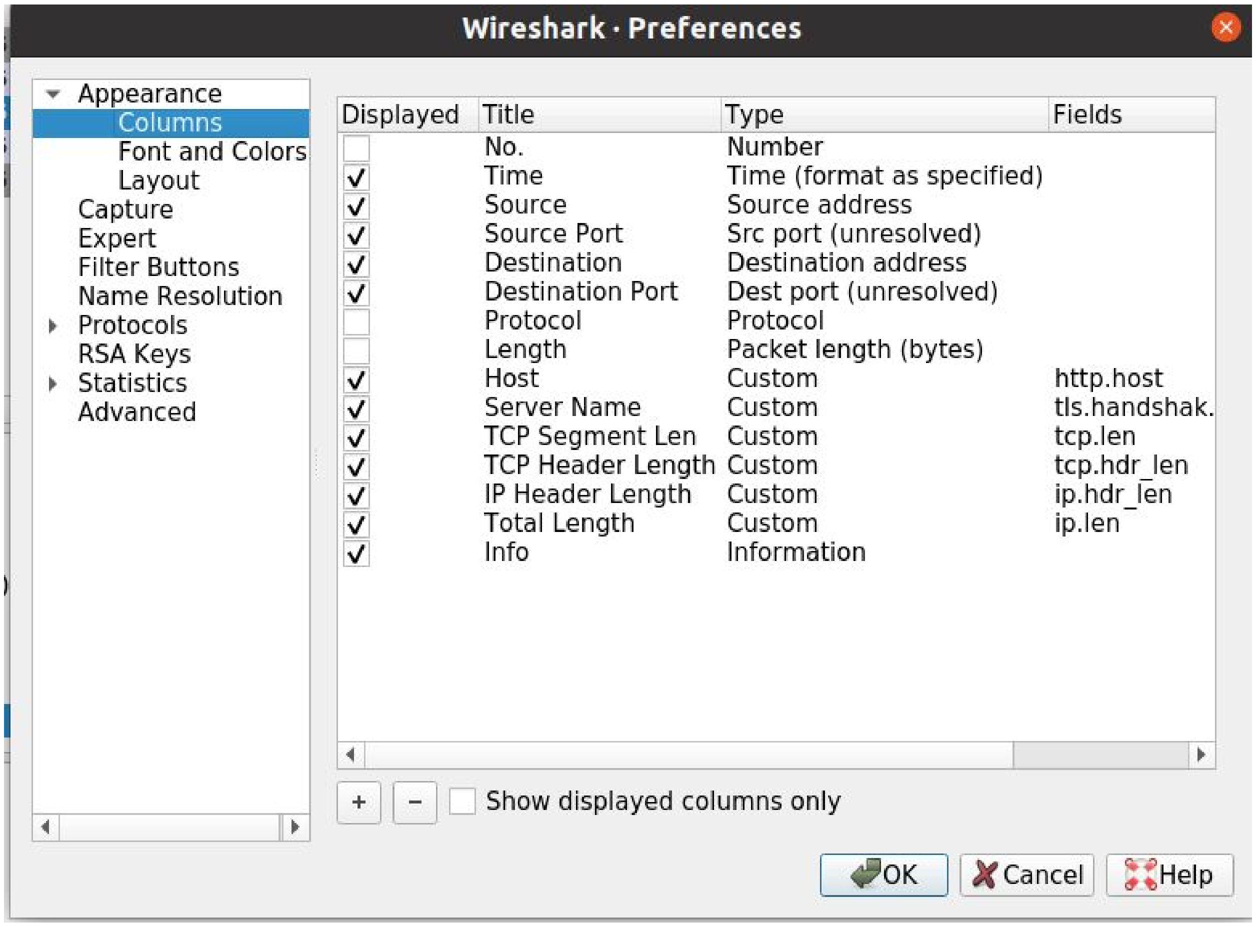
ip.dst == 127.0.0.1 and tcp.dstport == 2345 and tcp.len

| Time | Source | Source Port | Destination | Destination Port | Host | Server Name | TCP Segment Len | TCP Header Length | IP Header Length | Total Length | Info |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-03-14 17:28:20 | 127.0.0.1 | 44114 | 127.0.0.1 | 2345 | | | 0 | 20 | 20 | 40 | 44114 → 2345 [RST] Seq=1 Win=0 |
| 2021-03-14 17:29:22 | 127.0.0.1 | 44512 | 127.0.0.1 | 2345 | | | 0 | 40 | 20 | 60 | 44512 → 2345 [SYN] Seq=0 Win=65 |
| 2021-03-14 17:29:22 | 127.0.0.1 | 44512 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 44512 → 2345 [ACK] Seq=1 Ack=1 |
| 2021-03-14 17:29:53 | 127.0.0.1 | 44512 | 127.0.0.1 | 2345 | | | 6 | 32 | 20 | 58 | 44512 → 2345 [PSH, ACK] Seq=1 A |
| 2021-03-14 17:29:59 | 127.0.0.1 | 44512 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 44512 → 2345 [ACK] Seq=7 Ack=7 |
| 2021-03-14 17:29:59 | 127.0.0.1 | 44512 | 127.0.0.1 | 2345 | | | 0 | 32 | 20 | 52 | 44512 → 2345 [FIN, ACK] Seq=7 A |

```
    Source Port: 44512
    Destination Port: 2345
    [Stream index: 1]
    [TCP Segment Len: 6]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2012978161
    [Next Sequence Number: 7    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3207766474
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x018 (PSH, ACK)
    Window: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0xfe2e [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  [SEQ/ACK analysis]
  [Timestamps]
    TCP payload (6 bytes)
    TCP segment data (6 bytes)
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 3a a1 b0 40 00 40 06  9b 0b 7f 00 00 01 7f 00   ·:··@·@· ········
0020  00 01 ad e0 09 29 77 fb  9b f1 bf 32 a1 ca 80 18   ·····)w· ···2····
0030  02 00 fe 2e 00 00 01 01  08 0a 9b 74 51 a9 9b 73   ···.···· ···tQ··s
0040  da 43 63 6c 69 65 6e 74                            ·Cclient
```

A data segment used in reassembly of a lower-level protocol (tcp.segment_data), 6 bytes  Packets: 170 · Displayed: 6 (3.5%)  Profile: Default


Wireshark · Preferences

| Displayed | Title | Type | Fields |
|---|---|---|---|
| ☐ | No. | Number | |
| ✓ | Time | Time (format as specified) | |
| ✓ | Source | Source address | |
| ✓ | Source Port | Src port (unresolved) | |
| ✓ | Destination | Destination address | |
| ✓ | Destination Port | Dest port (unresolved) | |
| ☐ | Protocol | Protocol | |
| ☐ | Length | Packet length (bytes) | |
| ✓ | Host | Custom | http.host |
| ✓ | Server Name | Custom | tls.handshak. |
| ✓ | TCP Segment Len | Custom | tcp.len |
| ✓ | TCP Header Length | Custom | tcp.hdr_len |
| ✓ | IP Header Length | Custom | ip.hdr_len |
| ✓ | Total Length | Custom | ip.len |
| ✓ | Info | Information | |