CS3262 - Embedded Networks
Lab 01 – Socket Programming
210113L – Akindu Induwara

Client and Server Programs

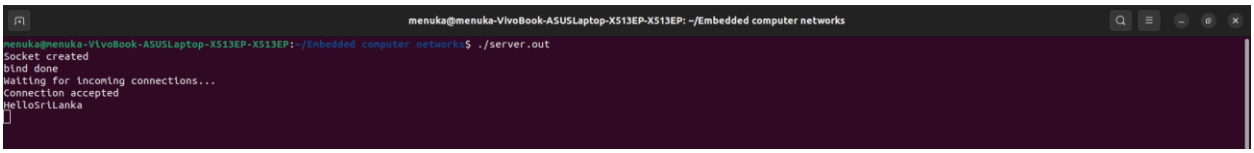


Figure1 - Server

Figure 1 shows the server terminal running the socket server program.

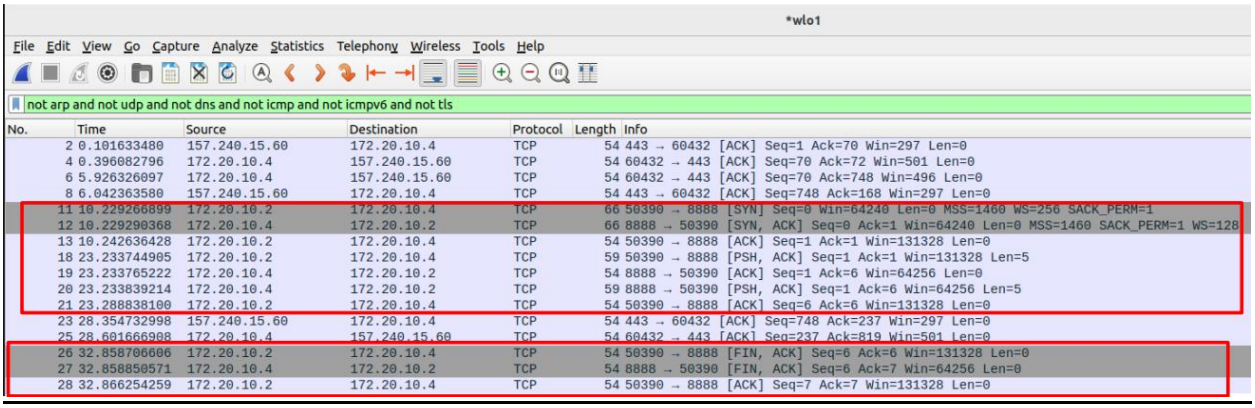
- The program starts by creating a socket using the socket() function and if successful prints “socket created” in the terminal
- The program then binds the socket to a specific IP address and port number using the bind() function and if successful prints “bind done” in the terminal.
- Then the program listens for incoming connections on the socket using the listen() function. The input parameters of the function (socket_desc,3) defines the socket and maximum number of connections that can wait while the server is handling a specific connection. "Waiting for incoming connections..." is printed on the screen while waiting for a connection.
- When a connection request is received, the server accepts the connection using the accept() function. If it is successful, "Connection accepted" is printed on the terminal
- When the connection is established, the server receives messages(data) from the client using the recv() function. It stores the data into the "client_message" buffer. Then, it prints the received message on the terminal and sends the same message back to the client using the write() function. This continues until client is disconnected.

Figure 2 - Client

Figure 2 shows the client terminal initiating a connection to the server and sending messages to the server.

- The program starts by creating a socket using the socket() function and if successful prints “socket created” in the terminal
- Using the connect() function, Client tries to connect to the server with the relevant IP address, port number, and address family specified in the program. If client connects to the specified server successfully, “Connected” is printed in the terminal
- The programs then shows "Enter message :" prompting for input. It reads the message entered by the user using scanf() function and sends it to the server using the send() function.
- Then, it waits for a reply from the server using the recv() function. When a reply is received it will be displayed in the terminal. This continues until the client is disconnected.

Packet Capture using Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
2	0.101633480	157.240.15.60	172.20.10.4	TCP	54	443 → 60432 [ACK] Seq=1 Ack=70 Win=297 Len=0
4	0.396082796	172.20.10.4	157.240.15.60	TCP	54	60432 → 443 [ACK] Seq=70 Ack=72 Win=501 Len=0
6	5.926326097	172.20.10.4	157.240.15.60	TCP	54	60432 → 443 [ACK] Seq=70 Ack=748 Win=496 Len=0
8	6.042363580	157.240.15.60	172.20.10.4	TCP	54	443 → 60432 [ACK] Seq=748 Ack=168 Win=297 Len=0
11	10.229266899	172.20.10.2	172.20.10.4	TCP	66	50390 → 8888 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	10.229290368	172.20.10.4	172.20.10.2	TCP	66	8888 → 50390 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
13	10.242636428	172.20.10.2	172.20.10.4	TCP	54	50390 → 8888 [ACK] Seq=1 Ack=1 Win=131328 Len=0
18	23.233744905	172.20.10.2	172.20.10.4	TCP	59	50390 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
19	23.233765222	172.20.10.4	172.20.10.2	TCP	54	8888 → 50390 [ACK] Seq=1 Ack=6 Win=64256 Len=0
20	23.233839214	172.20.10.4	172.20.10.2	TCP	59	8888 → 50390 [PSH, ACK] Seq=1 Ack=6 Win=64256 Len=5
21	23.288838100	172.20.10.2	172.20.10.4	TCP	54	50390 → 8888 [ACK] Seq=6 Ack=6 Win=131328 Len=0
23	28.354732998	157.240.15.60	172.20.10.4	TCP	54	443 → 60432 [ACK] Seq=748 Ack=237 Win=297 Len=0
25	28.601666908	172.20.10.4	157.240.15.60	TCP	54	60432 → 443 [ACK] Seq=237 Ack=819 Win=501 Len=0
26	32.858706606	172.20.10.2	172.20.10.4	TCP	54	50390 → 8888 [FIN, ACK] Seq=6 Ack=6 Win=131328 Len=0
27	32.858850571	172.20.10.4	172.20.10.2	TCP	54	8888 → 50390 [FIN, ACK] Seq=6 Ack=7 Win=64256 Len=0
28	32.866254259	172.20.10.2	172.20.10.4	TCP	54	50390 → 8888 [ACK] Seq=7 Ack=7 Win=131328 Len=0

Client IP is 172.20.10.2

Server Ip is 172.20.10.4

Packets transferred between client and server are shown inside rectangles.

TCP Protocol uses 3 way hand shake when establishing a connection. It ensures that both client and server are ready to send and receive data.

11	10.229266899	172.20.10.2	172.20.10.4	TCP	66	50390 → 8888 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	10.229290368	172.20.10.4	172.20.10.2	TCP	66	8888 → 50390 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
13	10.242636428	172.20.10.2	172.20.10.4	TCP	54	50390 → 8888 [ACK] Seq=1 Ack=1 Win=131328 Len=0

Packet 11 - SYN (Synchronize)

The client initiates the connection by sending a SYN packet to the server. This packet contains a sequence number chosen by the client to start the communication.

Packet 12 - SYN-ACK (Synchronize-Acknowledge)

After receiving the SYN packet, if the server is ready to establish a connection, it responds with a SYN-ACK packet. This packet acknowledges the receipt of the client's SYN packet and also contains a sequence number chosen by the server.

Packet 13 - ACK (Acknowledge)

At last, the client acknowledges the receipt of the server's SYN-ACK packet by sending an ACK packet. This packet contains the next sequence number, confirming the server's acknowledgment. At this point, the connection is established, and both parties can begin sending data

After the 3-way handshake the client and server are ready to communicate.

18	23.233744905	172.20.10.2	172.20.10.4	TCP	59	50390 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
19	23.233765222	172.20.10.4	172.20.10.2	TCP	54	8888 → 50390 [ACK] Seq=1 Ack=6 Win=64256 Len=0
20	23.233839214	172.20.10.4	172.20.10.2	TCP	59	8888 → 50390 [PSH, ACK] Seq=1 Ack=6 Win=64256 Len=5
21	23.288838100	172.20.10.2	172.20.10.4	TCP	54	50390 → 8888 [ACK] Seq=6 Ack=6 Win=131328 Len=0

Packet 18

The client sends the message “Hello” to the server. This packet uses PSH flag.

```
18 23.233744905  172.20.10.2   172.20.10.4   TCP   59 50390 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=5
▸ Frame 18: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface wlo1, id 0
▸ Ethernet II, Src: IntelCor_3e:c9:61 (84:5c:f3:3e:c9:61), Dst: IntelCor_23:87:78 (64:79:f0:23:87:78)
▸ Internet Protocol Version 4, Src: 172.20.10.2, Dst: 172.20.10.4
▸ Transmission Control Protocol, Src Port: 50390, Dst Port: 8888, Seq: 1, Ack: 1, Len: 5
▾ Data (5 bytes)
    Data: 68656c6c6f
    [Length: 5]

0000  64 79 f0 23 87 78 84 5c  f3 3e c9 61 08 00 45 00   dy·#·x·\ ·>·a··E·
0010  00 2d aa d1 40 00 80 06  e3 ca ac 14 0a 02 ac 14   ·-··@··· ########
0020  0a 04 c4 d6 22 b8 82 8b  3a 4c b6 48 e3 0b 50 18   ····"··· :L·H··P·
0030  02 01 c0 0a 00 00 68 65  6c 6c 6f                  ######he llo
```

Packet 19

Then the server sends an Acknowledgment to the client indicating a successful receive of the data. this packet uses ACK flag.

Packet 20

After that the server also returns the messages received from the client to the client using PSH flag.

```
20 23.233839214  172.20.10.4   172.20.10.2   TCP   59 8888 → 50390 [PSH, ACK] Seq=1 Ack=6 Win=64256 Len=5
▸ Frame 20: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface wlo1, id 0
▸ Ethernet II, Src: IntelCor_23:87:78 (64:79:f0:23:87:78), Dst: IntelCor_3e:c9:61 (84:5c:f3:3e:c9:61)
▸ Internet Protocol Version 4, Src: 172.20.10.4, Dst: 172.20.10.2
▸ Transmission Control Protocol, Src Port: 8888, Dst Port: 50390, Seq: 1, Ack: 6, Len: 5
▾ Data (5 bytes)
    Data: 68656c6c6f
    [Length: 5]

0000  84 5c f3 3e c9 61 64 79  f0 23 87 78 08 00 45 00   ·\·>·ady ·#·x··E·
0010  00 2d cd 6f 40 00 40 06  01 2d ac 14 0a 04 ac 14   ··o@·@·· ########
0020  0a 02 22 b8 c4 d6 b6 48  e3 0b 82 8b 3a 51 50 18   ··"····H ····:QP·
0030  01 f6 6c 4e 00 00 68 65  6c 6c 6f                  ··lN··he llo
```

Packet 21

Then the client also sends an Acknowledgment to the server indicating a successful receive of the data using ACK flag.

When the users want to close the communication between client and server, following packets are interchanged bet ween the client and the server.

```
26 32.858706606  172.20.10.2   172.20.10.4   TCP   54 50390 → 8888 [FIN, ACK] Seq=6 Ack=6 Win=131328 Len=0
27 32.858850571  172.20.10.4   172.20.10.2   TCP   54 8888 → 50390 [FIN, ACK] Seq=6 Ack=7 Win=64256 Len=0
28 32.866254259  172.20.10.2   172.20.10.4   TCP   54 50390 → 8888 [ACK] Seq=7 Ack=7 Win=131328 Len=0
```

Packet 26

The client sends FIN flag to signal that it has finished sending data and want to terminate the connection. The connection remains open for any remaining data to be transmitted in the other direction.

Packet 27

After sending the FIN segment, the Server sends a FIN and ACK flag to the client confirming the termination request. The connection remains open to allow any remaining data to be received.

Packet 28

Then the Client will send an ACK Flag to the Server. Then both client and server will go to Closed state by termination the connection.