

q1:

```
client.py
~/lab-files/lab-tcp

1 import socket
2
3
4 def client_program():
5     # Configure the hostname and port.
6     host = '10.0.0.10'
7     port = 50001
8
9     # TO-DO: Create a new client socket.
10    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11
12    # TO-DO: Connect to the server.
13    client_socket.connect((host, port))
14
15    # Get data from user
16    data = input('> ')
17
18    # Continue sending data to the server until the end command.
19    while data.lower().strip() != 'end':
20        # TO-DO: Send data to the server (data.encode()).
21        client_socket.send(data.encode())
22
23        # TO-DO: Receive data back from the server (up to 1024 bytes).
24        data = client_socket.recv(1024).decode()
25
26        print('Data stream from server (\'' + host + '\', ' + str(port) + '): ' + data)
27
28        # Get data from user
29        data = input('> ')
30
31    # TO-DO: Close connection.
32    client_socket.close()
33
34    print('Connection terminated with server (\'' + host + '\', ' + str(port) + ')')
35
36
37 if __name__ == '__main__':
38     client_program()
```

q2:

*veth2.0.58

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns

No.	Time	Source	Destination	Protocol	Length	Info
2	1.515157413	10.0.1.10	10.0.0.10	TCP	74	51854 → 50001 [SYN]
3	1.515170056	10.0.0.10	10.0.1.10	TCP	74	50001 → 51854 [SYN]
4	1.515185956	10.0.1.10	10.0.0.10	TCP	66	51854 → 50001 [ACK]

Acknowledgment number (raw): 2273871357

1010 = Header Length: 40 bytes (10)

Flags: 0x012 (SYN, ACK)

0000 = Reserved: Not set

....0 = Nonce: Not set

....0 = Congestion Window Reduced (CWR): Not set

....0 = ECN-Echo: Not set

....0 = Urgent: Not set

....1 = Acknowledgment: Set

....0 = Push: Not set

....0 = Reset: Not set

....1 = Syn: Set

....0 = Fin: Not set

[TCP Flags:A..S.]

*veth3.0.58

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
4	4.386317693	10.0.1.10	10.0.0.10	TCP	74	51854 → 50001 [SYN] Seq=0
5	4.386354151	10.0.0.10	10.0.1.10	TCP	74	50001 → 51854 [SYN, ACK] S
6	4.386362997	10.0.1.10	10.0.0.10	TCP	66	51854 → 50001 [ACK] Seq=1

1000 = Header Length: 32 bytes (8)

Flags: 0x010 (ACK)

0000 = Reserved: Not set

....0 = Nonce: Not set

....0 = Congestion Window Reduced (CWR): Not set

....0 = ECN-Echo: Not set

....0 = Urgent: Not set

....1 = Acknowledgment: Set

....0 = Push: Not set

....0 = Reset: Not set

....0 = Syn: Not set

....0 = Fin: Not set

[TCP Flags:A....]

Window size value: 502

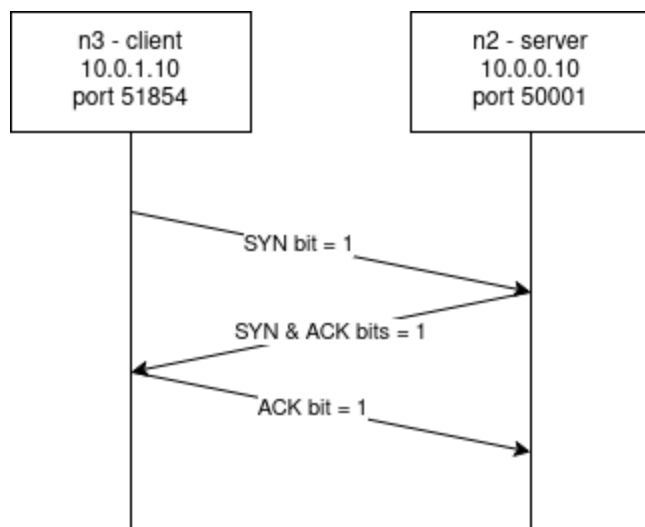
[Calculated window size: 642561]

Transmission Control Protocol: Protocol

Packets: 11 - Displayed: 3 (27.3%) - Dropped: 0 (0.0%) Profile: De

Transmission Control Protocol: Protocol

Packets: 15 - Displayed: 3 (20.0%) - Dropped: 0 (0.0%) Profile: Default



q3:

The server distinguishes both clients by identifying their IP addresses and port numbers. Even if both clients had the same IP address, the server could distinguish them by the port number. The 4 values (Source IP + port, Dest IP + source) identify the packet.

q4:

A ligação termina normalmente com pacotes FYN (e ACK) de cada um dos lados. Não é possível identificar nenhum pacote do tipo FYN no trace file usado.

q5:

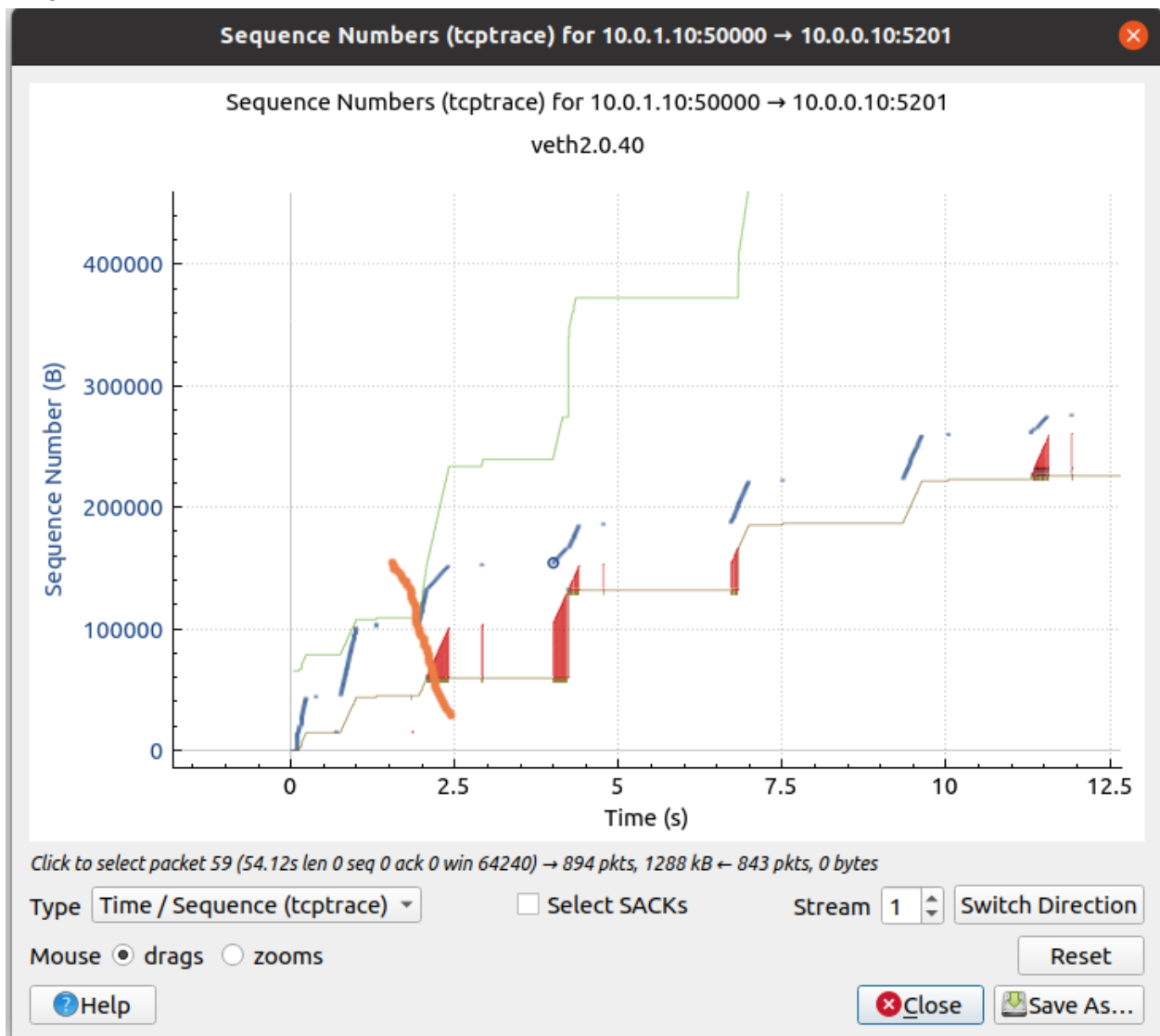
Neste caso, a conexão TCP não é feita porque as portas estão a ser reutilizadas, daí a flag RST (de RESET) estar ativa nos pacotes de resposta para indicar ao cliente que volte a tentar conectar-se.

q6:

Losing a packet in the network can cause duplicate ACKs. This happens because the following packets arriving at the receiver will generate cumulative ACKs and will not include the lost packet, hence generating duplicated ACKs when arriving at the sender. The sender just responds by resending the lost packet, instead of waiting for its timeout. This is called fast retransmitting.

q7:

Antes do 1º conjunto de traços vermelhos (até aos 2 seg +/-), temos slow start. Depois temos congestion avoidance.



q8:

A fast retransmission makes the sender retransmit the lost packet, the receiver gets it and the communication continues normally, with the window size being cut in half. In a regular transmission, after a timeout, the window size is set back to 1 and we have a new slow start, decreasing the speed of the communication.