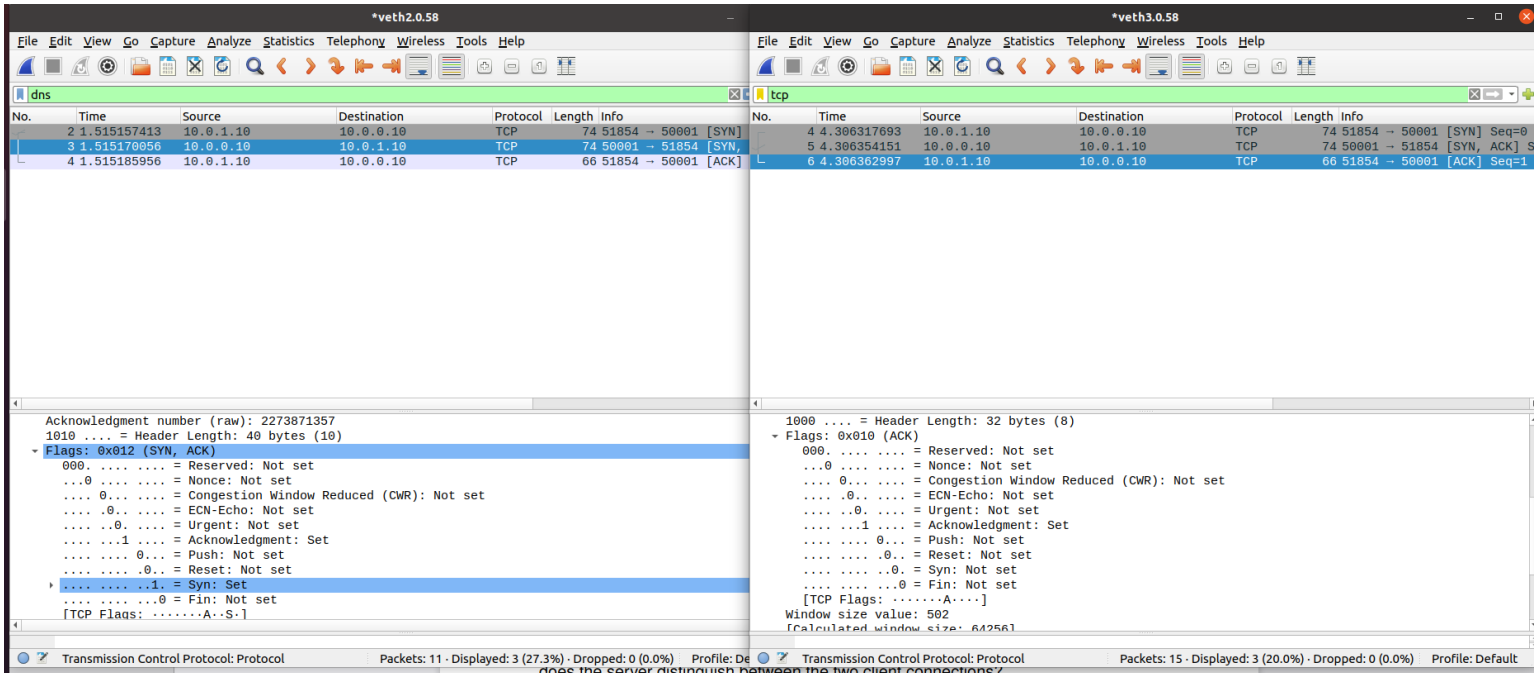
q1:

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

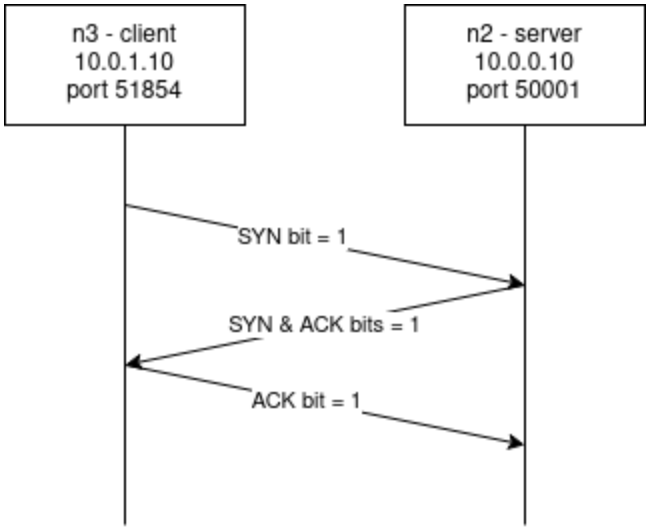
```python
import socket

def client_program():
    # Configure the hostname and port.
    host = '10.0.0.10'
    port = 50001

    # TO-DO: Create a new client socket.
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # TO-DO: Connect to the server.
    client_socket.connect((host, port))

    # Get data from user
    data = input('> ')

    # Continue sending data to the server until the end command.
    while data.lower().strip() != 'end':
        # TO-DO: Send data to the server (data.encode()).
        client_socket.send(data.encode())

        # TO-DO: Receive data back from the server (up to 1024 bytes).
        data = client_socket.recv(1024).decode()

        print('Data stream from server (\'' + host + '\',  ' + str(port) + '): ' + data)

        # Get data from user
        data = input('> ')

    # TO-DO: Close connection.
    client_socket.close()

    print('Connection terminated with server (\'' + host + '\', ' + str(port) + ')')


if __name__ == '__main__':
    client_program()
```

Python Tab Width: 8 Ln 3, Col 1 INS

q2:



does the server distinguish between the two client connections?



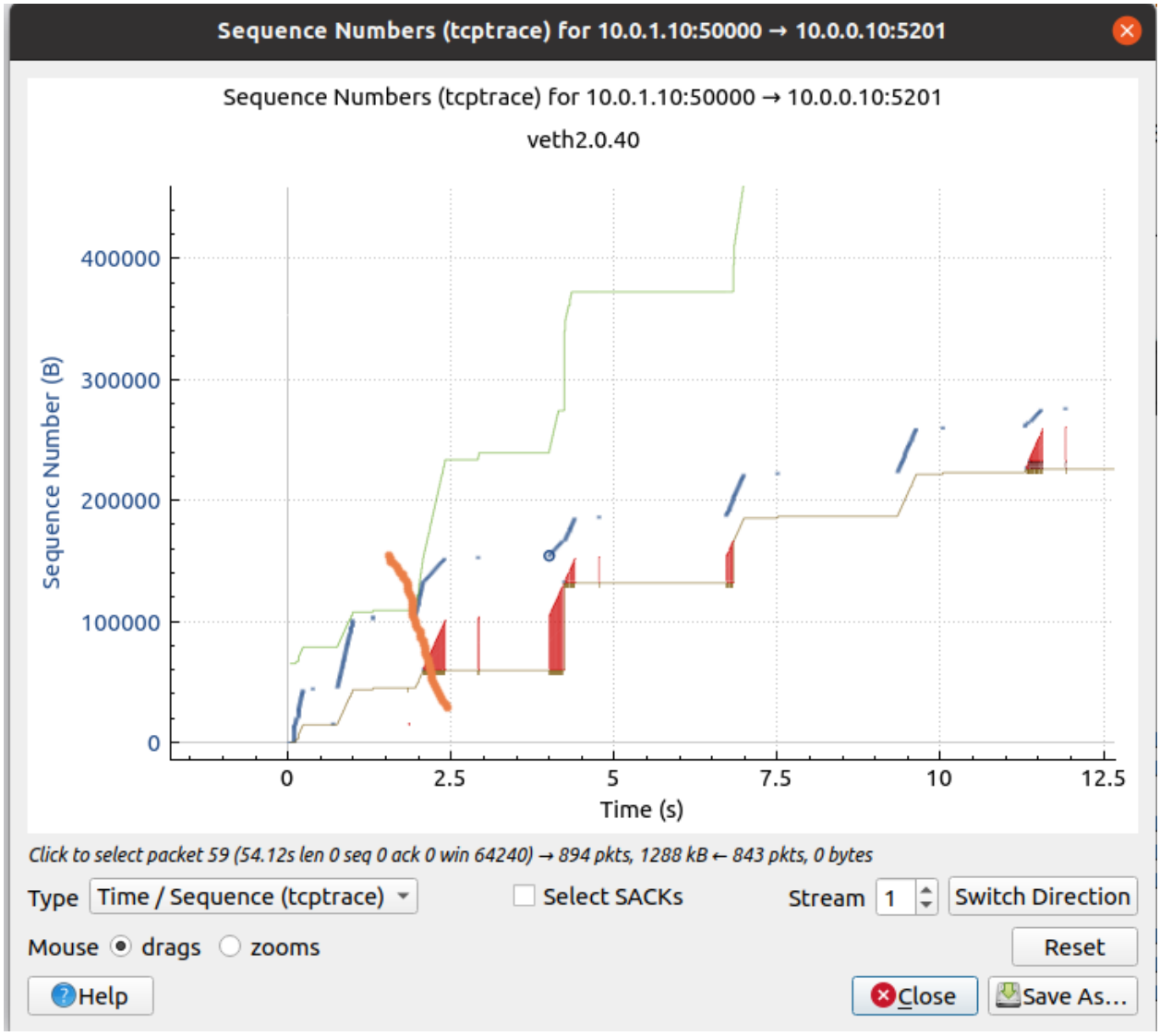
q3:
The server distinguishes both clients by identifying their IP addresses and port numbers. Even if both clients had the same IP address, the server could distinguish them by the port number. The 4 values (Source IP + port, Dest IP + source) identify the packet.

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 it’s 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.