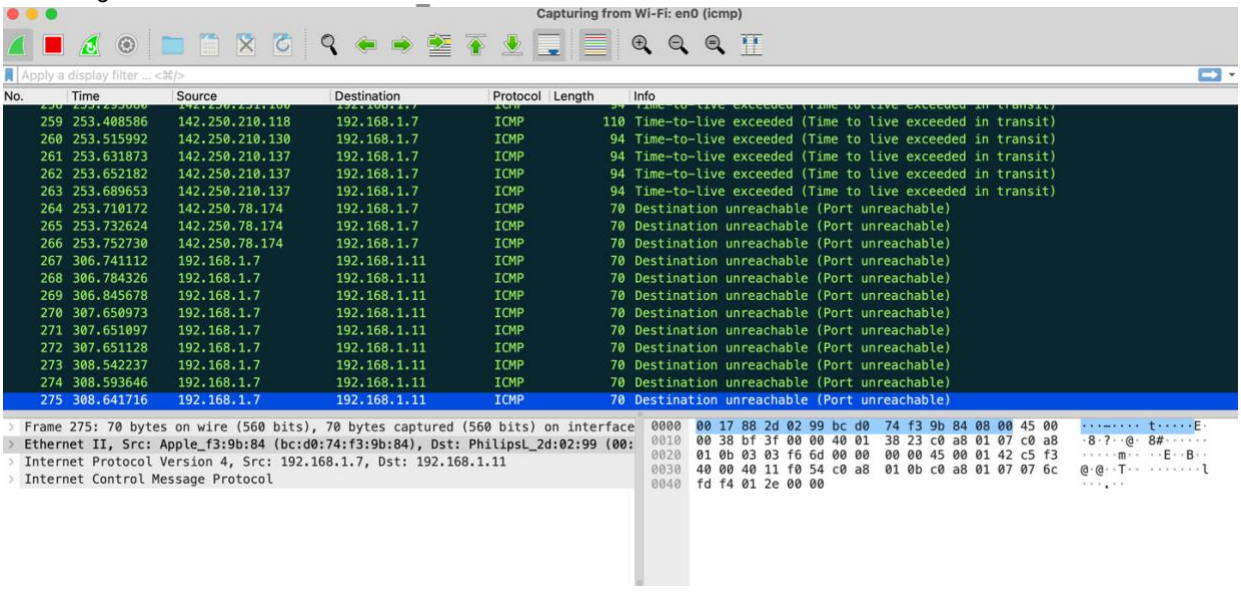
CMP-4005 – Homework 2

Sebastián Romero (00216765)

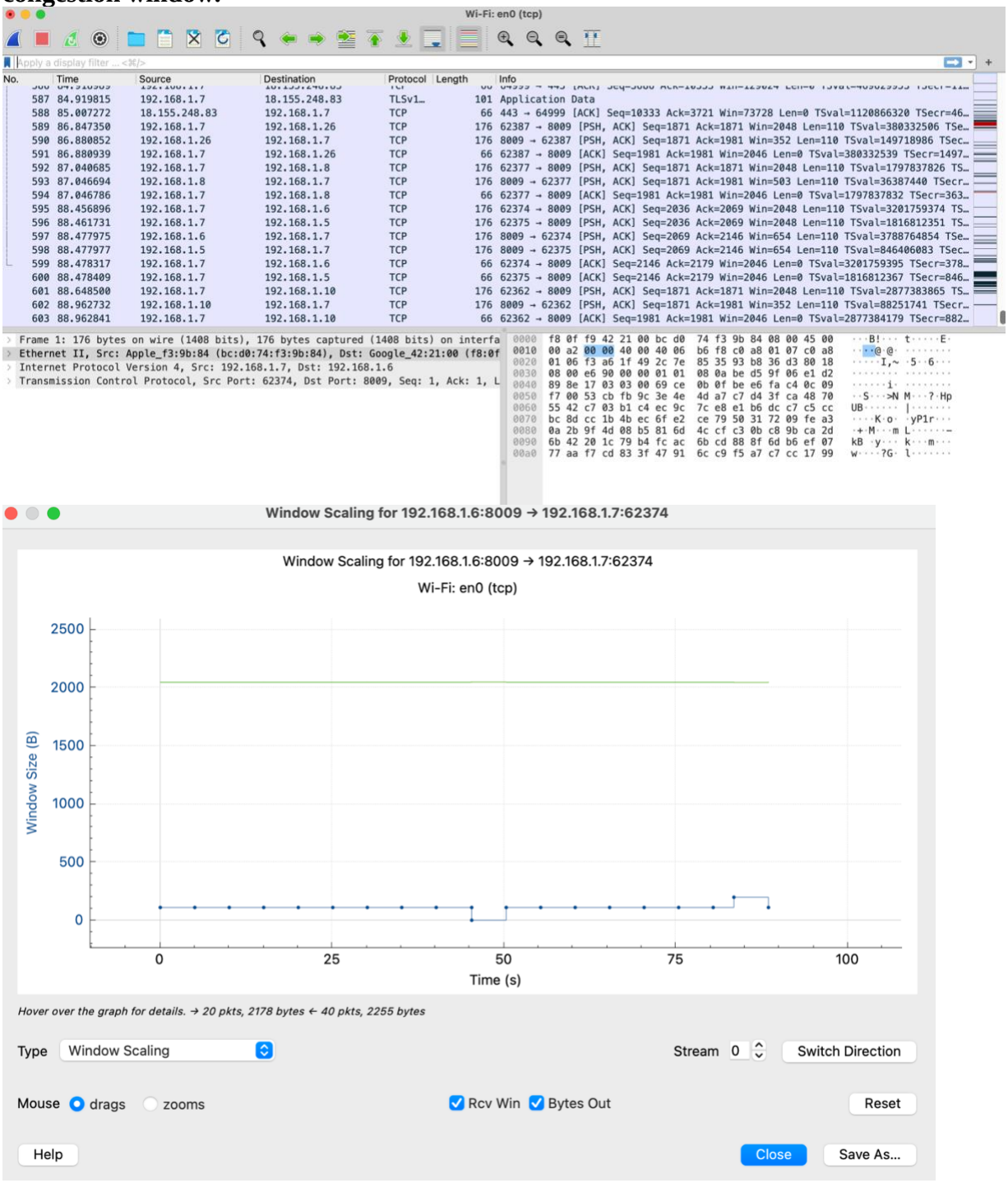
1) Read the following Wireshark tutorial, and use it to capture traffic from the following scenarios. Use screenshots to show your results.

a) Run 10 traceroute commands against google.com

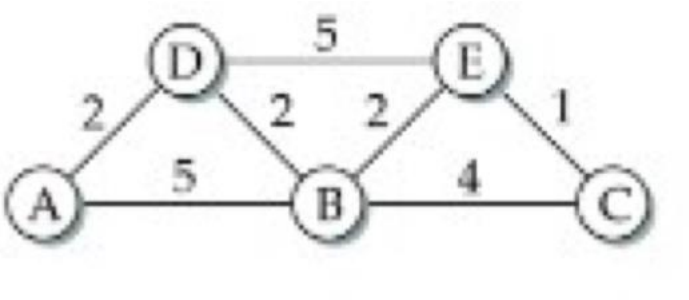
```
sebas@Sebass-MacBook-Pro ~ % traceroute google.com
traceroute to google.com (142.250.78.174), 64 hops max, 52 byte packets
 1  192.168.1.1 (192.168.1.1)  3.247 ms  2.509 ms  19.830 ms
 2  192.168.100.1 (192.168.100.1)  3.068 ms  2.829 ms  2.593 ms
 3  181.39.211.129 (181.39.211.129)  9.411 ms  7.783 ms  8.217 ms
 4  10.224.51.134 (10.224.51.134)  5.936 ms  5.489 ms  5.832 ms
 5  186.3.125.46 (186.3.125.46)  5.843 ms
    142.250.163.94 (142.250.163.94)  5.470 ms
    186.3.125.46 (186.3.125.46)  7.258 ms
 6  186.3.125.47 (186.3.125.47)  24.963 ms
    142.250.163.95 (142.250.163.95)  18.485 ms
    186.3.125.47 (186.3.125.47)  19.146 ms
 7  * * *
 8  142.250.231.160 (142.250.231.160)  20.509 ms
    142.250.210.118 (142.250.210.118)  22.130 ms
    142.250.210.130 (142.250.210.130)  21.353 ms
 9  142.250.210.137 (142.250.210.137)  18.716 ms  18.916 ms  37.494 ms
10  bog02s19-in-f14.1e100.net (142.250.78.174)  20.495 ms  20.541 ms  20.052 ms
```



b) Watch a video from youtube.com. Capture the TCP handshake, and the congestion window.



2)Use Dijkstra's to get the routing tables for nodes A, B and E.



A

Step	Confirmed	Tentative
1	(A,0,-)	
2	(A,0,-)	(D,2,D) (B,5,B)
3	(A,0,-) (D,2,D)	(B,4,D) (E,7,D)
4	(A,0,-) (D,2,D) (B,4,D)	(E,6,D) (C,8,D)
5	(A,0,-) (D,2,D) (B,4,D) (E,6,D)	(C,7,D)
6	(A,0,-) (D,2,D) (B,4,D) (E,6,D) (C,7,D)	

B

Step	Confirmed	Tentative
1	(B,0,-)	(A,5,A) (D,2,D) (E,2,E) (C,4,C)
2	(B,0,-) (D,2,D)	(A,4,D) (E,7,D)
3	(B,0,-) (D,2,D) (A,4,D)	
4	(B,0,-) (E,2,E)	(D,7,E) (C,3,E)
5	(B,0,-) (E,2,E) (C,3,E)	

E

Step	Confirmed	Tentative
1	(E,0,-)	(C,1,C) (B,2,B) (D,5,D)
2	(E,0,-) (B,2,B)	(C,6,B) (D,4,B) (A,7,B)
3	(E,0,-) (B,2,B) (D,4,B)	(A,6,B)
4	(E,0,-) (B,2,B) (D,4,B) (A,6,B)	

3) Suppose a host wants to establish the reliability of a link by sending packets and measuring the percentage that are received; routers, for example, do this. Explain the difficulty of doing this over a TCP connection.

Performing this measurement over a TCP connection presents some difficulties.

TCP is a connection-oriented protocol that guarantees reliable data transmission by using a sequence number and acknowledgement mechanism. When a TCP sender sends a packet, it waits for an acknowledgement from the receiver before sending the next packet. If the sender does not receive an acknowledgement within a certain amount of time, it will retransmit the packet until it receives an acknowledgement. This ensures that all packets are delivered reliably and in order.

While this reliability mechanism is beneficial for data transmission, it can make it difficult to measure the percentage of packets that are received. This is because, if a packet is lost or not acknowledged, the sender will keep retransmitting the packet until it receives an acknowledgement. This can make it difficult to distinguish between packets that were lost and packets that were simply delayed in transit. As a result, the measurement of the percentage of received packets can be inaccurate and may not provide an accurate reflection of the quality of the link.

4) Consider a simple congestion control algorithm that uses linear increase and multiplicative decrease (no slow start). Assume the congestion window size is in units of packets rather than bytes, and it is one packet initially.
a) Give a detailed sketch of this algorithm.
b) Assume the delay is latency only, and that when a group of packets is sent, only a single ACK is returned.
c) Plot the congestion window as a function of RTT for the situation in which the following packets are lost: 9, 25, 30, 38 and 50. For simplicity, assume a perfect timeout mechanism that detects a lost packet exactly 1 RTT after it is transmitted.

a) The algorithm can be summarized as follows:

- Initially, the congestion window size is set to 1 packet.
- For each successful round-trip transmission of all packets, increase the congestion window size by 1 packet.
- If a packet loss is detected, decrease the congestion window size by half
- Repeat the above steps until the desired throughput or congestion avoidance is achieved.

b) Since only a single ACK is returned for a group of packets, the congestion window will be increased by the number of packets acknowledged by each ACK. For example, if 5 packets are sent and only 1 ACK is received, then the congestion window will be increased by 1 packet.

c)

RTT	Window size	Pkt transmitted
1	1	1
2	2	2-3
3	3	4-6
4	4	7-10
5	2	9-10
6	3	11-13
7	4	14-17
8	5	18-22
9	6	23-28
10	3	25-27
11	4	28-31
12	2	30-31
13	3	32-34
14	4	35-38
15	2	38-39
16	3	40-42
17	4	43-46
18	5	47-51
19	2	50-51