**Networks & Data Communications I**

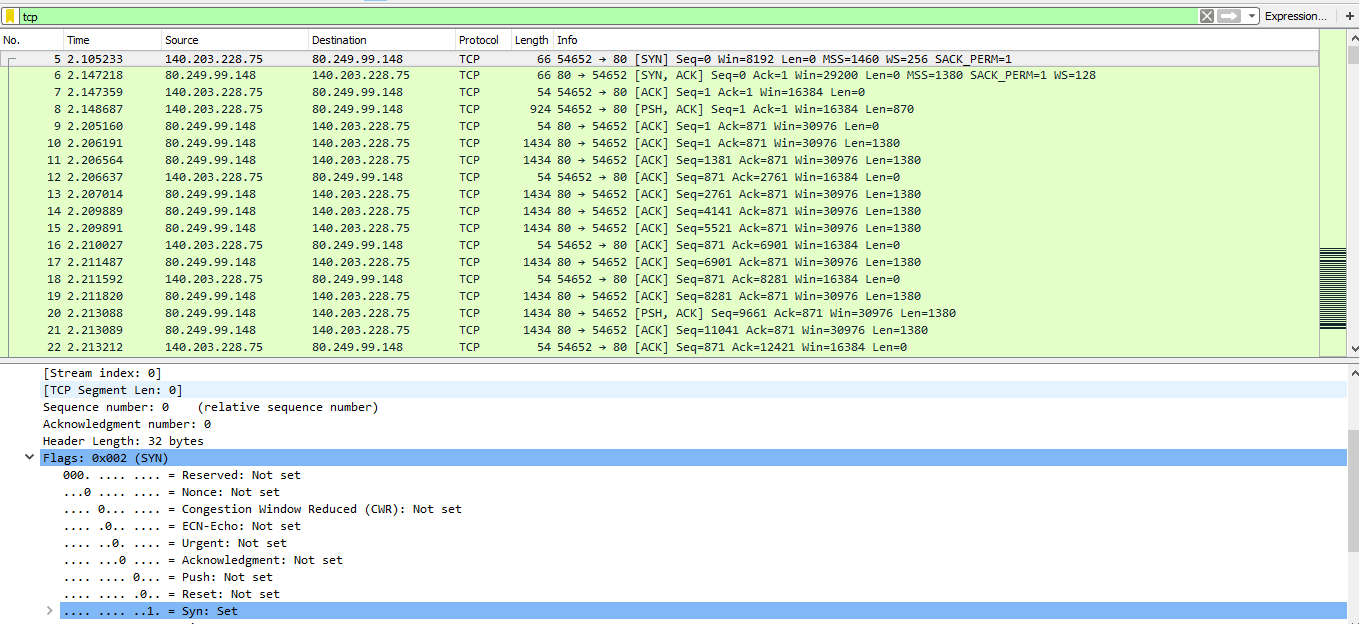
**Assignment 1**

***TCP Protocol Analysis using Wireshark***

* Cillian Fennell – 15407302 – [C.FENNELL3@nuigalway.ie](mailto:C.FENNELL3@nuigalway.ie) – 31/03/2017
* Andre Godinez – 15460718 – [A.GODINEZ1@nuigalway.ie](mailto:A.GODINEZ1@nuigalway.ie) – 31/03/2017

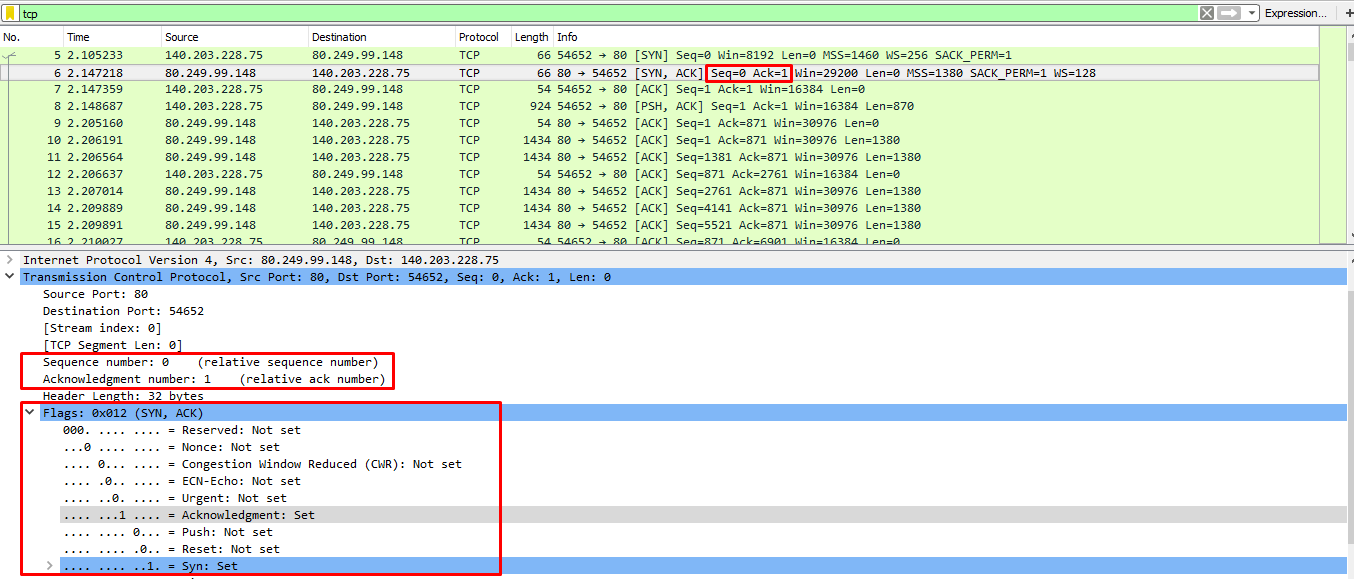
**Q1.**

* The sequence number of the TCP SYN segment that was used to initiate a connection between my computer and the server is 0.
* The segment is identified as a SYN segment as its SYN flag has a value of 1.



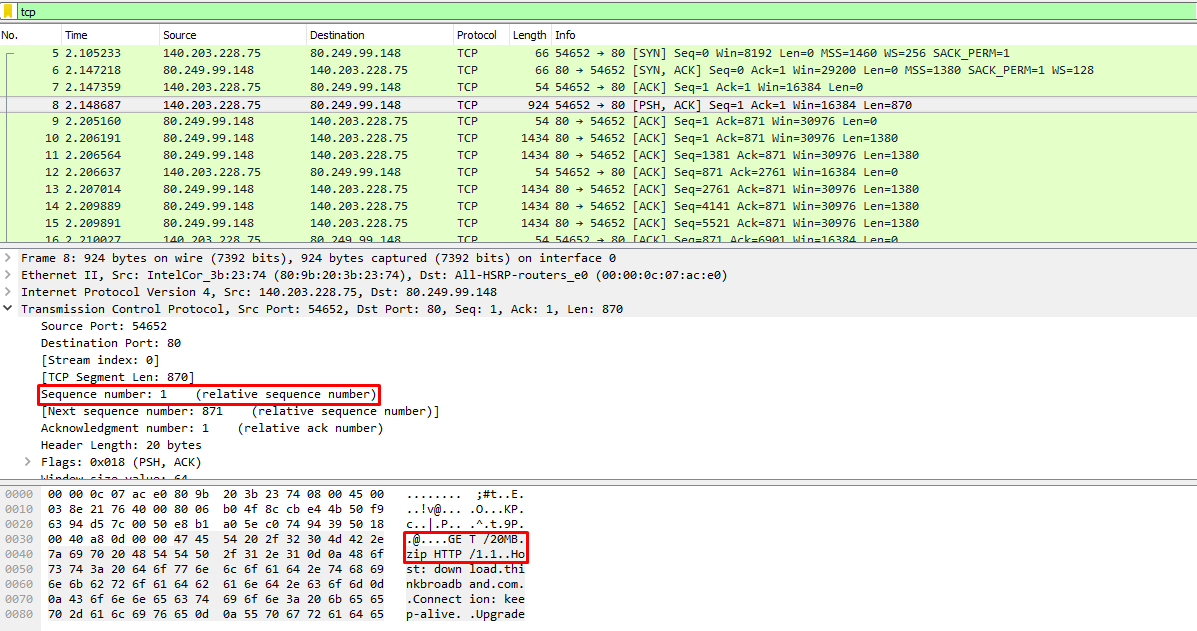
**Q2.**

* The sequence number of the SYN ACK segment sent by the server to my computer in reply to the SYN is 0.
* The value of the Acknowledgement field in the SYN ACK segment is 1.
* This value was found by the sequence number of the SYN segment from **Q1.** plus 1.
* This segment is identified as a SYN ACK segment as both its SYN flag and ACK flag have a value of 1.

****

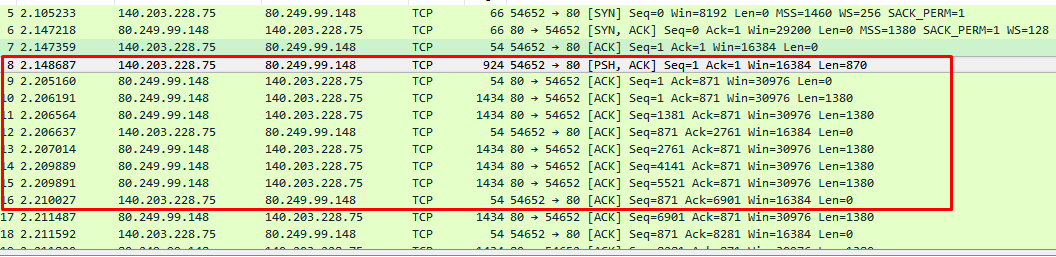
**Q3.**

* The sequence number of the TCP segment containing the initial HTTP GET command is 1.

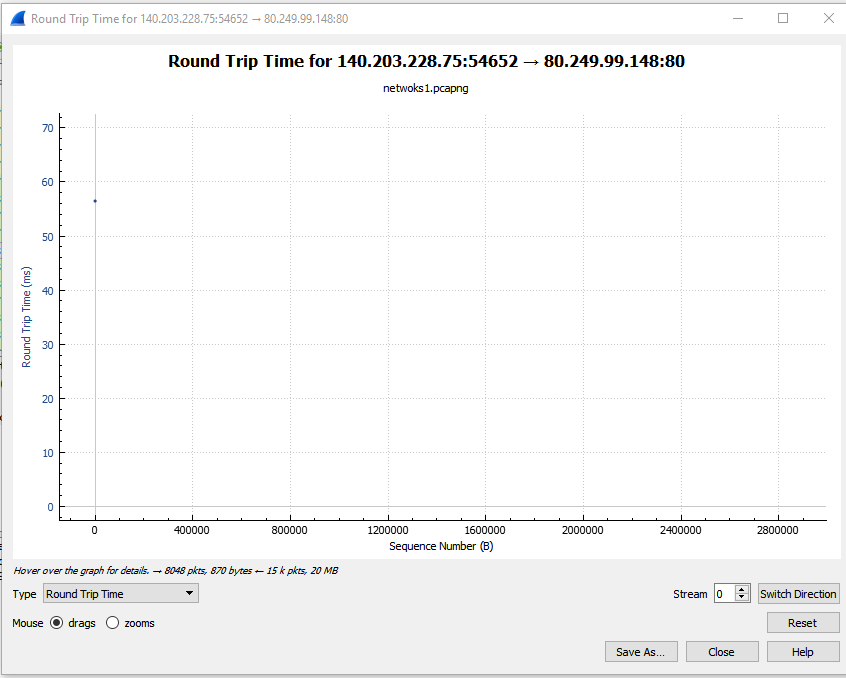
****

**Q4.**

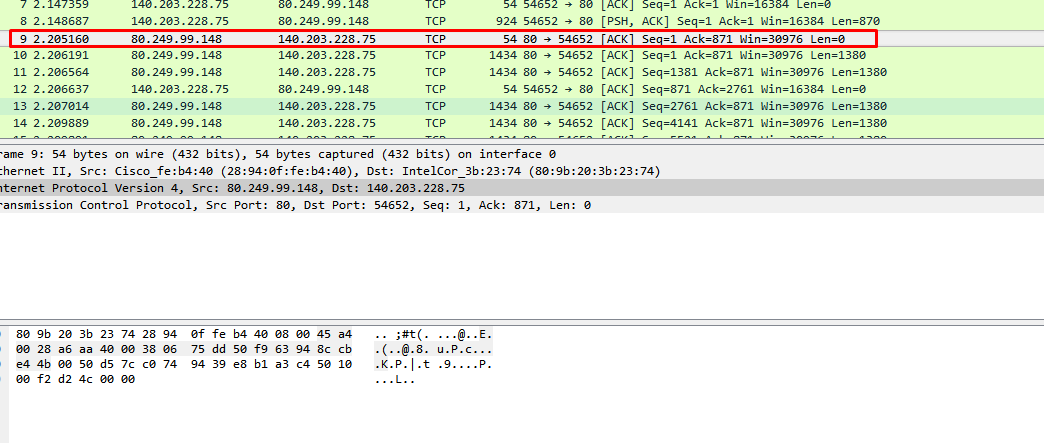
* First 6 sequence numbers of the TCP connection are 1, 1, 1381, 2761, 4141 and 5521.
* Sent at 2.148687, 2.206191, 2.206564, 2.207014, 2.209889 and 2.209891 respectively.
* ACK segments received at 2.205160, 2.206564, 2.206637, 2.209889, 2.209891 and 2.210027 respectively.
* Note: I was confused by this question as I assumed each TCP segment would have a corresponding ACK segment directly after it was sent. Also the amount of TCP segments outweighs the ACK segments. After doing some research online I came to the assumption that data piggybacking is in action here where some ACK response are being sent with some TCP segments as to reduce the amount of segments needing to be sent. This is why some of my ACK segments are received at the same time some of the TCP segments are being sent (as seen above)



**Q5.**

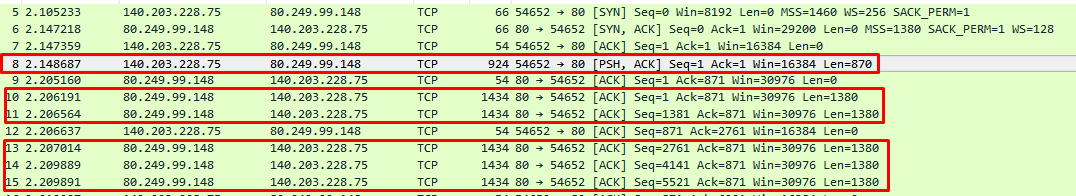


* The round trip time for this TCP segment is approximately 56ms



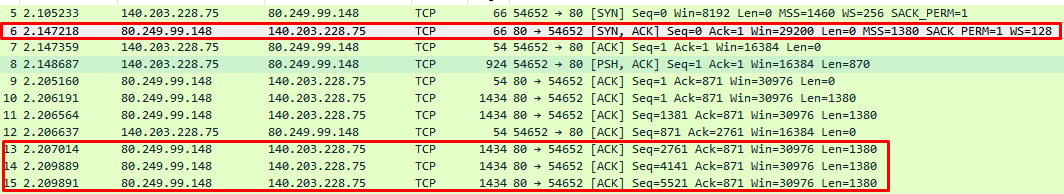
**Q6.**

* The length of the first segment is 870, the lengths of the following five segments are 1380.



**Q7.**

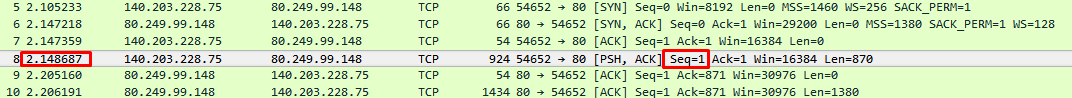
* The typical amount of available buffer space advertised at the receiver for the entire trace is 30976 bytes.

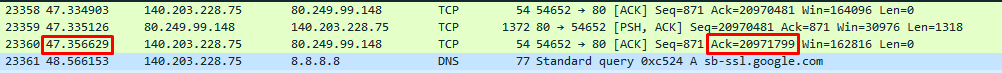


* As seen above the minimum amount of available buffer space advertised at the receiver is 29200 bytes, and also examples of the typical amount available, 30976 bytes.

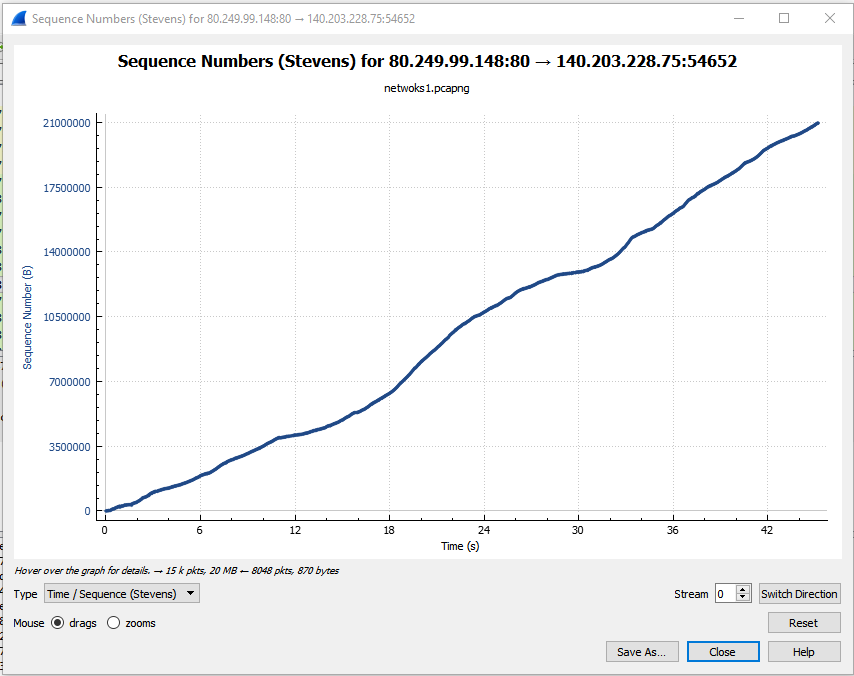
**Q8.**

* Estimated throughput is 463 Kbytes/sec.
* The throughput is the data transmitted divided by the time taken for the data to be transmitted.
* The data transmitted is found by taking away the sequence number of the first TCP segment from the sequence number of the last ACK segment. In this case it is 20,971,799 – 1 = 20,971,798 bytes.
* The time taken is the time between the 2 respective segments. In this case it is 47.356629 – 2.148687 = 45.207942 seconds.
* Throughput = 20,971,798/45.207942 = 463,896.3215 bytes/sec, or, 463.896 Kbytes/seC.

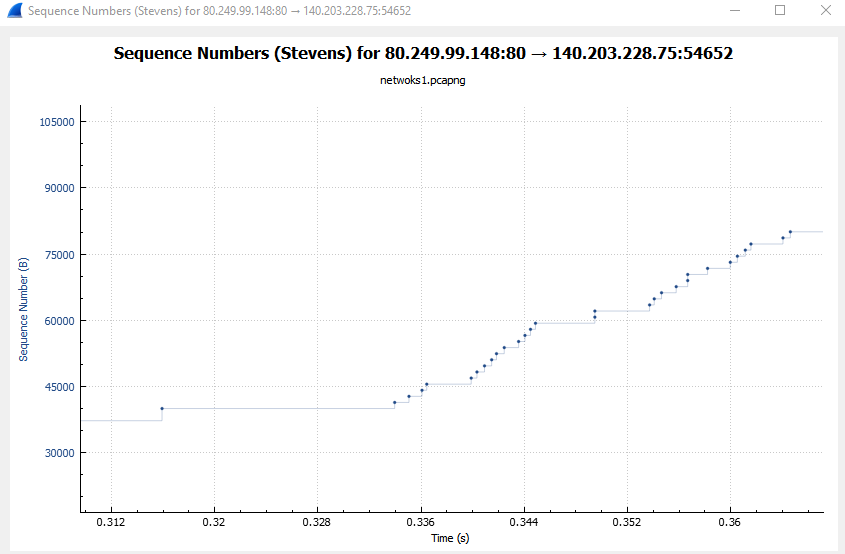




**Q9.**



* The TCP slow start phase starts at the beginning of the connection



* Here the slow start ends at 0.336s and congestion control starts where packets start to be sent in groups

**Q10.**

* My results differ from the idealized behaviour of TCP as it only uses a fraction of the window size instead of the ideal value of around a half.
* The graph for congestion control is more uneven and gradual compared to the ideal vertical, steep graphs