COMP3331 LAB 4
Abanob Tawfik
Z5075490

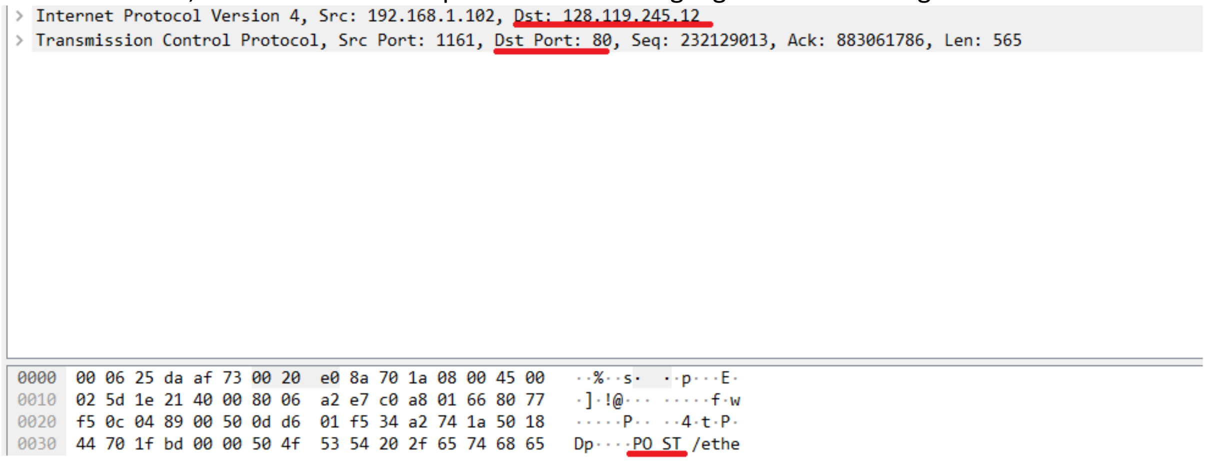
Contents

Exercise 1: Understanding TCP using Wireshark

Question 1

What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

The 4th HTTP message is a post from MIT (source) to Gaia (destination). The destination IP address is **128.119.245.12**, and the destination port is **80**. This is highlighted below in Figure 1.

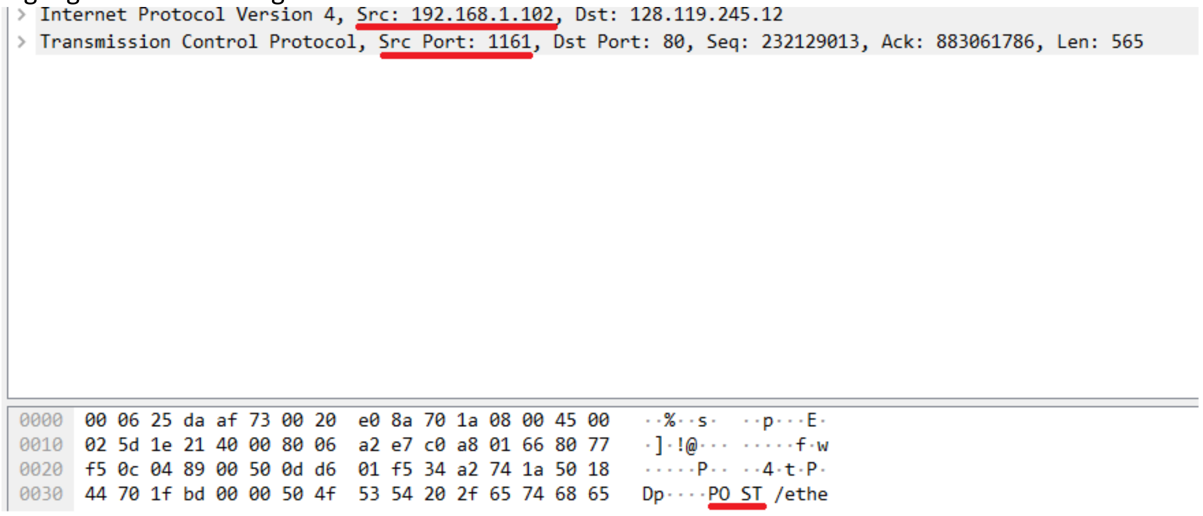


```
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 232129013, Ack: 883061786, Len: 565

0000  00 06 25 da af 73 00 20  e0 8a 70 1a 08 00 45 00   ··%··s·  ··p···E·
0010  02 5d 1e 21 40 00 80 06  a2 e7 c0 a8 01 66 80 77   ·]·!@···  ·····f·w
0020  f5 0c 04 89 00 50 0d d6  01 f5 34 a2 74 1a 50 18   ·····P··  ··4·t·P·
0030  44 70 1f bd 00 00 50 4f  53 54 20 2f 65 74 68 65   Dp····PO ST /ethe
```

Figure 1 the IP address and port number for gaia.cs.umass.edu.

What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

Similar to above, the source IP address is **192.168.1.102** and the source port is **1161**. This is highlighted below in Figure 2.



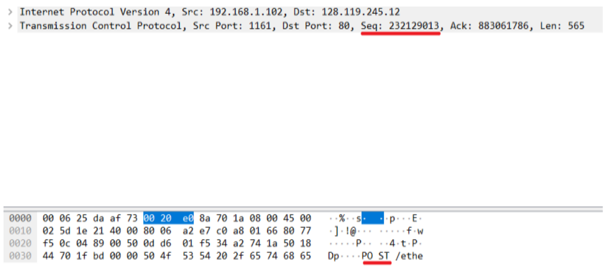
```
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 232129013, Ack: 883061786, Len: 565

0000  00 06 25 da af 73 00 20  e0 8a 70 1a 08 00 45 00   ··%··s·  ··p···E·
0010  02 5d 1e 21 40 00 80 06  a2 e7 c0 a8 01 66 80 77   ·]·!@···  ·····f·w
0020  f5 0c 04 89 00 50 0d d6  01 f5 34 a2 74 1a 50 18   ·····P··  ··4·t·P·
0030  44 70 1f bd 00 00 50 4f  53 54 20 2f 65 74 68 65   Dp····PO ST /ethe
```

Figure 2 the IP address and port number for host in MIT.

Question 2

What is the sequence number of the TCP segment containing the HTTP POST command?

The sequence number for the TCP segment containing the HTTP post command is **232129013**, this is highlighted below in Figure 3. The reason it is also 232129013 is that the initial sequence number for the SYN was 232129012 and 1 bit is added to the sequence number for the acknowledgment of the SYN.



```
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 232129013, Ack: 883061786, Len: 565
```

```
0000  00 06 25 da af 73 00 20  e0 8a 70 1a 08 00 45 00   ··%··s·  ·p···E·
0010  02 5d 1e 21 40 00 80 06  a2 e7 c0 a8 01 66 80 77   ·]·!@···  ·····f·w
0020  f5 0c 04 89 00 50 0d d6  01 f5 34 a2 74 1a 50 18   ·····P··  ··4·t·P·
0030  44 70 1f bd 00 00 50 4f  53 54 20 2f 65 74 68 65   Dp····PO ST /ethe
```

Figure 3 the sequence number as seen in the HTTP post message.

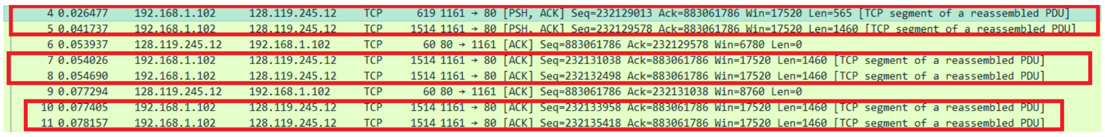
Question 3

Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection sent from the client to the web server?

The first six segments including the first HTTP post have the following sequence numbers

1. **232129013**
2. **232129578**
3. **232131038**
4. **232132498**
5. **232133958**
6. **232135418**

This is highlighted in Figure 4, since the TCP segments from client to web server have source **192.168.1.102** and destination **128.119.245.12**



```
  4 0.026477   192.168.1.102    128.119.245.12   TCP    619 1161 → 80 [PSH, ACK] Seq=232129013 Ack=883061786 Win=17520 Len=565 [TCP segment of a reassembled PDU]
  5 0.041737   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [PSH, ACK] Seq=232129578 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
  6 0.053937   128.119.245.12   192.168.1.102    TCP     60 80 → 1161 [ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0
  7 0.054026   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
  8 0.054690   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
  9 0.077294   128.119.245.12   192.168.1.102    TCP     60 80 → 1161 [ACK] Seq=883061786 Ack=232131038 Win=8760 Len=0
 10 0.077405   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 11 0.078157   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
```

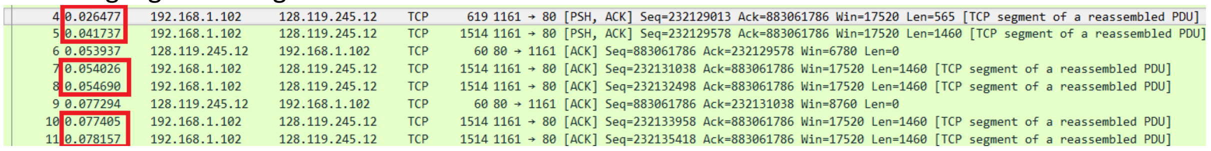
Figure 4 the 6 segments from client to web server highlighted.

At what time was each segment sent?

The time each segment was sent is given below

- segment 1 → **0.026477**
- segment 2 → **0.041737**
- segment 3 → **0.054026**
- segment 4 → **0.054690**
- segment 5 → **0.077405**
- segment 6 → **0.078157**

This is highlighted in Figure 5 under the time column in Wireshark.



```
  4 0.026477   192.168.1.102    128.119.245.12   TCP    619 1161 → 80 [PSH, ACK] Seq=232129013 Ack=883061786 Win=17520 Len=565 [TCP segment of a reassembled PDU]
  5 0.041737   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [PSH, ACK] Seq=232129578 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
  6 0.053937   128.119.245.12   192.168.1.102    TCP     60 80 → 1161 [ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0
  7 0.054026   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
  8 0.054690   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
  9 0.077294   128.119.245.12   192.168.1.102    TCP     60 80 → 1161 [ACK] Seq=883061786 Ack=232131038 Win=8760 Len=0
 10 0.077405   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 11 0.078157   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
```

Figure 5 the time for each segment highlighted in the red box.

When was the ACK for each segment received?

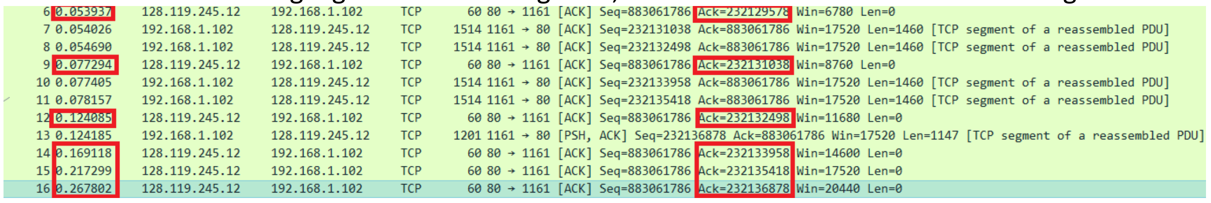
To check when the ack was received for each segment, the length of the segment was added to the sequence number, and I found the ACK with that ACK number.

- For segment 1 – sequence number = 232129013, length = 565,
  - ACK = 232129013 + 565 = 232129578
- For segment 2 – sequence number = 232129578, length = 1460
  - ACK = 232129578 + 1460 = 232131038
- For segment 3 – sequence number = 232131038, length = 1460
  - ACK = 232131038 + 1460 = 232132498
- For segment 4 – sequence number = 232132498, length = 1460
  - ACK = 232132498 + 1460 = 232133958
- For segment 5 – sequence number = 232133958, length = 1460
  - ACK = 232133958 + 1460 = 232135418
- For segment 6 – sequence number = 232135418, length = 1460
  - ACK = 232135418 + 1460 = 232136878

The time for when each ACK was received is the following

- segment 1 → **0.053937**
- segment 2 → **0.077294**
- segment 3 → **0.124085**
- segment 4 → **0.169118**
- segment 5 → **0.217299**
- segment 6 → **0.267802**

The ACK numbers are highlighted below in Figure 6, as well as the time column for the segments.



```
 6 0.053937  128.119.245.12  192.168.1.102   TCP    60 80 → 1161 [ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0
 7 0.054026  192.168.1.102   128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 8 0.054690  192.168.1.102   128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 9 0.077294  128.119.245.12  192.168.1.102   TCP    60 80 → 1161 [ACK] Seq=883061786 Ack=232131038 Win=8760 Len=0
10 0.077405  192.168.1.102   128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
11 0.078157  192.168.1.102   128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
12 0.124085  128.119.245.12  192.168.1.102   TCP    60 80 → 1161 [ACK] Seq=883061786 Ack=232132498 Win=11680 Len=0
13 0.124185  192.168.1.102   128.119.245.12  TCP  1201 1161 → 80 [PSH, ACK] Seq=232136878 Ack=883061786 Win=17520 Len=1147 [TCP segment of a reassembled PDU]
14 0.169118  128.119.245.12  192.168.1.102   TCP    60 80 → 1161 [ACK] Seq=883061786 Ack=232133958 Win=14600 Len=0
15 0.217299  128.119.245.12  192.168.1.102   TCP    60 80 → 1161 [ACK] Seq=883061786 Ack=232135418 Win=17520 Len=0
16 0.267802  128.119.245.12  192.168.1.102   TCP    60 80 → 1161 [ACK] Seq=883061786 Ack=232136878 Win=20440 Len=0
```

Figure 6 Highlighted ACK numbers and time of receive.

What is the RTT value for each of the six segments?

The RTT value for each segment can be calculated from subtracting the time the segment was sent from the time the ack was sent.

The RTT values for each segment are the following

- segment 1 → 0.053937 - 0.026477
  - RTT = **0.02746** seconds
- segment 2 → 0.077294 - 0.041737
  - RTT = **0.035557** seconds
- segment 3 → 0.124085 - 0.054026
  - RTT = **0.070059** seconds
- segment 4 → 0.169118 - 0.054690
  - RTT = **0.114428** seconds
- segment 5 → 0.217299 - 0.077405
  - RTT = **0.139894** seconds
- segment 6 → 0.267802 - 0.078157
  - RTT = **0.189645** seconds

What is the *EstimatedRTT* value after the receipt of each ACK?

The formula for estimated RTT is given below as
Estimated RTT = (1- alpha) *EstimatedRTT + alpha*SampleRTT
For segment 1, we say EstimatedRTT = SampleRTT
Assuming that value alpha = 0.125

- segment 1 → (1-0.125) * (0.02746) + (0.125) * (0.02746)
    - Estimated RTT = **0.02746** seconds
- segment 2 → (1-0.125) * (0.02746) + (0.125) * (0.035557)
    - Estimated RTT = **0.028472125** seconds
- segment 3 → (1-0.125) * (0.028472125) + (0.125) * (0.070059)
    - Estimated RTT = **0.03367048437** seconds
- segment 4 → (1-0.125) * (0.03367048437) + (0.125) * (0.114428)
    - Estimated RTT = **0.04376517382** seconds
- segment 5 → (1-0.125) * (0.04376517382) + (0.125) * (0.139894)
    - Estimated RTT = **0.05578127709** seconds
- segment 6 → (1-0.125) * (0.05578127709) + (0.125) * (0.189645)
    - Estimated RTT = **0.07251424245** seconds

Table for Question 3

Table 1.
Data used in Question 3.

Segment Number	*Sequence Number*	*Time of Send* (seconds)	*Time Ack for segment was Received* (seconds)	*RTT for segment* (seconds)	*Estimated RTT for segment* (seconds)
1	232129013	0.026477	0.053937	0.02746	0.02746
2	232129578	0.041737	0.077294	0.035557	0.028472125
3	232131038	0.054026	0.124085	0.070059	0. 03367048437
4	232132498	0.054690	0.169118	0.114428	0.04376517382
5	232133958	0.077405	0.217299	0.139894	0.05578127709
6	232135418	0.078157	0.267802	0.189645	0.07251424245

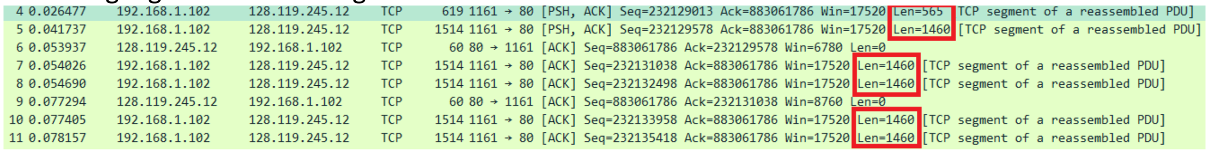
Question 4

What is the length of each of the first six TCP segments?

The length of the first six TCP segments are the following

- segment 1 → **565** bytes
- segment 2 → **1460** bytes
- segment 3 → **1460** bytes
- segment 4 → **1460** bytes
- segment 5 → **1460** bytes
- segment 6 → **1460** bytes

This is highlighted below in Figure 7



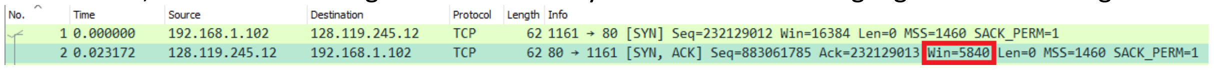
```
 4 0.026477   192.168.1.102    128.119.245.12   TCP    619 1161 → 80 [PSH, ACK] Seq=232129013 Ack=883061786 Win=17520 Len=565  TCP segment of a reassembled PDU]
 5 0.041737   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [PSH, ACK] Seq=232129578 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 6 0.053937   128.119.245.12   192.168.1.102    TCP     60 80 → 1161 [ACK] Seq=883061786 Ack=232129578 Win=6780 Len=0
 7 0.054026   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232131038 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 8 0.054690   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232132498 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
 9 0.077294   128.119.245.12   192.168.1.102    TCP     60 80 → 1161 [ACK] Seq=883061786 Ack=232131038 Win=8760 Len=0
10 0.077405   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232133958 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
11 0.078157   192.168.1.102    128.119.245.12   TCP   1514 1161 → 80 [ACK] Seq=232135418 Ack=883061786 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
```

Figure 7 Length of segment highlighted in wireshark.

Question 5

What is the minimum amount of available buffer space advertised at the receiver for the entire trace?

The minimum amount of buffer advertised at the receiver for the trace is **5840** bytes, this can be seen in SYN, ACK at the 2nd stage of the three-way handshake. This is highlighted below in Figure 8

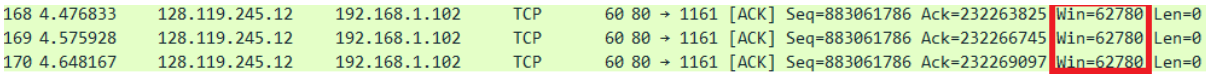


```
No.   Time       Source          Destination     Protocol Length Info
    1 0.000000   192.168.1.102   128.119.245.12  TCP      62 1161 → 80 [SYN] Seq=232129012 Win=16384 Len=0 MSS=1460 SACK_PERM=1
    2 0.023172   128.119.245.12  192.168.1.102   TCP      62 80 → 1161 [SYN, ACK] Seq=883061785 Ack=232129013 Win=5840 Len=0 MSS=1460 SACK_PERM=1
```

Figure 8 The minimum amount of buffer space given in WIN.

Does the lack of receiver buffer space ever throttle the sender?

The receiver window size grows throughout the duration of the trace, and reaches a maximum of **62780** bytes as seen in Figure 9, the increase in window size is designed to prevent throttle, hence the sender will never be throttled.



```
168 4.476833   128.119.245.12   192.168.1.102   TCP   60 80 → 1161 [ACK] Seq=883061786 Ack=232263825 Win=62780 Len=0
169 4.575928   128.119.245.12   192.168.1.102   TCP   60 80 → 1161 [ACK] Seq=883061786 Ack=232266745 Win=62780 Len=0
170 4.648167   128.119.245.12   192.168.1.102   TCP   60 80 → 1161 [ACK] Seq=883061786 Ack=232269097 Win=62780 Len=0
```

Figure 9 maximum amount of buffer space observed in the trace.

Question 6

Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

There were no retransmitted segments in the trace file. To check for re-transmitted segments, we check for segments with the same sequence number, implying that we are re-sending a previously sent segment, as segments are identified by their sequence number. This can be simply done in Wireshark by sorting the data by clicking on "info" and check for duplicates, (much easier to see when sorted by seq number).

Question 7

How much data does the receiver typically acknowledge in an ACK?

Typically, the amount of data acknowledged in an ACK is **1460** bytes, mostly likely due to the fact that MSS = 1460 bytes. This can be observed below in the following Table.

Table 2.
Data used in Question 3.

ACK Number	*ACK Sequence Number*	Length of acknowledged data
1	232129013	None
2	232129578	565
3	232131038	1460
4	232132498	1460
5	232133958	1460
6	232135418	1460
7	232136878	1460
8	232138025	1147
…	…	…
59	232164061	1460
60	232166981	2920

… implies many more in-between

Can you identify cases where the receiver is ACKing every other received segment?

There are cases where the receiver is asking every other segment, and this can be seen when the jump in ack number is 2*1460, this occurs in segment 59-60, 69-70, 79-80, and in the case of 88-89 where we get 2*1176 in the ack. There are multiple cases that occur where the receiver acks every other received segment.

Question 8

What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

To workout the bytes transferred per unit of time we need to first

1. workout total number of bytes sent
2. total transmission time

total bytes sent is = (ACK number of last segment) – (sequence number of the very first segment)
ack number of last segment - 232293103 (seen in number 202)
sequence number of the very first segment – 232129012

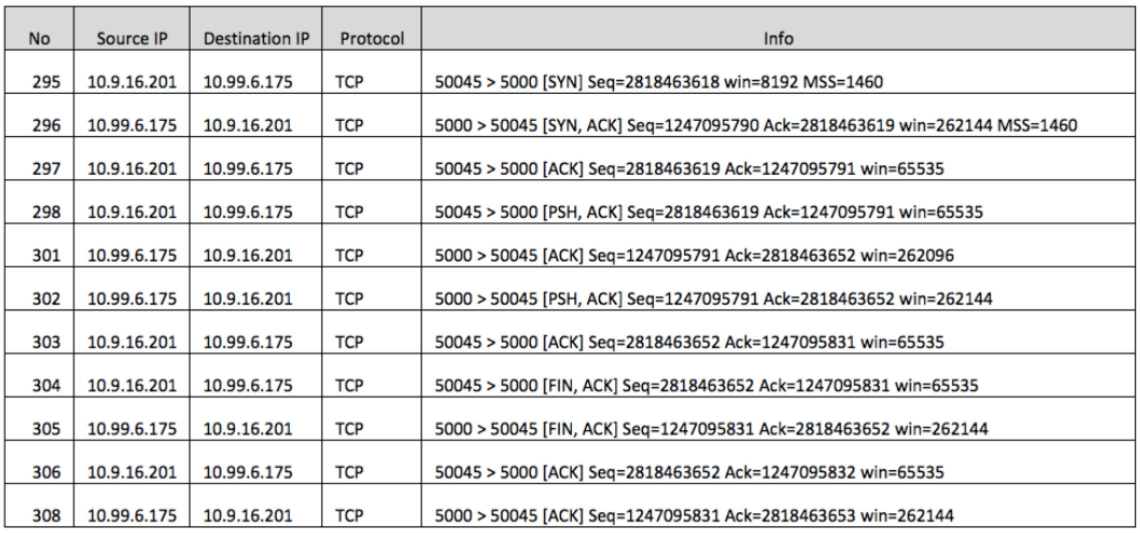
total transmission time = (time at last segment) – (time at first segment)
time at last segment – 5.455830
time at first segment – 0.026477

total number of bytes sent = 232293103 - 232129012
= 164090 bytes
Total transmission time = 5.455830 – 0.026477
= 5.429353 seconds

Throughput = (total number of bytes sent) / (total transmission time) bytes/second
Throughput = 164090/5.429353 bytes/second
Throughput = **30222.7539819** bytes/second

Exercise 2: TCP Connection Management

Consider the following TCP transaction between a client (10.9.16.201) and a server (10.99.6.175).



No	Source IP	Destination IP	Protocol	Info
295	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [SYN] Seq=2818463618 win=8192 MSS=1460
296	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [SYN, ACK] Seq=1247095790 Ack=2818463619 win=262144 MSS=1460
297	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463619 Ack=1247095791 win=65535
298	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [PSH, ACK] Seq=2818463619 Ack=1247095791 win=65535
301	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [ACK] Seq=1247095791 Ack=2818463652 win=262096
302	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [PSH, ACK] Seq=1247095791 Ack=2818463652 win=262144
303	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463652 Ack=1247095831 win=65535
304	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [FIN, ACK] Seq=2818463652 Ack=1247095831 win=65535
305	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [FIN, ACK] Seq=1247095831 Ack=2818463652 win=262144
306	10.9.16.201	10.99.6.175	TCP	50045 > 5000 [ACK] Seq=2818463652 Ack=1247095832 win=65535
308	10.99.6.175	10.9.16.201	TCP	5000 > 50045 [ACK] Seq=1247095831 Ack=2818463653 win=262144

Question 1

What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and server?

The sequence number of segment used to initiate the connection between client and server is **2818463618**

Question 2

What is the sequence number of the SYNACK segment sent by the server to the client computer in reply to the SYN?

The sequence number of the segment sent as a reply to question 1 is **1247095790**

What is the value of the Acknowledgement field in the SYNACK segment? How did the server determine that value?

The value in the acknowledgement field for the SYNACK segment is **2818463619** and the server determined this value by adding 1 to the sequence number, since the SYN is 1 byte. This informs the client when it receives this value that the next sequence number it is expecting is from **2818463619** implying a successful transmission of the SYN

Question 3

What is the sequence number of the ACK segment sent by the client computer in response to the SYNACK?

The sequence number in the ack segment (the final stage for the TCP handshake) is **2818463619,** this is since the last ack received back was **2818463619.**

What is the value of the Acknowledgment field in this ACK segment? Does this segment contain any data?

The value of the acknowledgement field is **1247095791**, and this is due to the SYN being 1 byte. This segment does not contain any data, and this can be seen by comparing the sequence number in the proceeding segment which is **2818463619**, implying no data is sent.

Question 4

Who has done the active close? client or the server? how you have determined this?

The client has done the active close, and this was determined by checking the source ip address for the first FIN, ACK segment **(10.9.16.201)**

What type of closure has been performed? 3 Segment (FIN/FINACK/ACK), 4 Segment (FIN/ACK/FIN/ACK) or Simultaneous close?

The type of closure performed was a 3-segment closure, as piggybacking is utilised to both acknowledge the FIN and also send out its own FIN seen in number 304,305

Question 5

How many data bytes have been transferred from the client to the server and from the server to the client during the whole duration of the connection?

The total data bytes transferred from client to server is
Final ACK number for client - the client ISN
Final ack number for client = 2818463653
Client ISN = 2818463618
Total data bytes transferred = 2818463653 – 2818463618 bytes
= **35** bytes

What relationship does this have with the Initial Sequence Number and the final ACK received from the other side?

The relationship between total number of bytes sent with the initial sequence and the final ack received is that,
Total number of bytes sent = (ACK number from final ack) – (initial sequence number)