Wireshark Lab - 2
BSSE 1204

1. 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? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows.

Ans:
Ip 10.100.106.15
Port 443

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

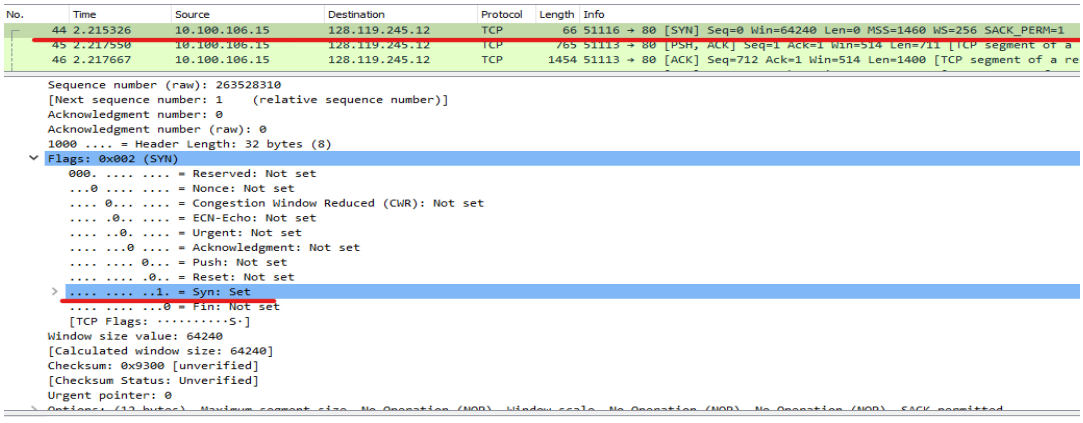
Ans:
IP 128.119.245.12
Sending Port 80
Receiving Port 50792

3. What is the IP address and TCP port number used by your client's computer (source) to transfer the file to gaia.cs.umass.edu?

Ans:
Ip 10.100.106.15
Port 443

4. What is the **sequence number of the TCP SYN segment** that is used to initiate the The TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as an SYN segment?

Ans:

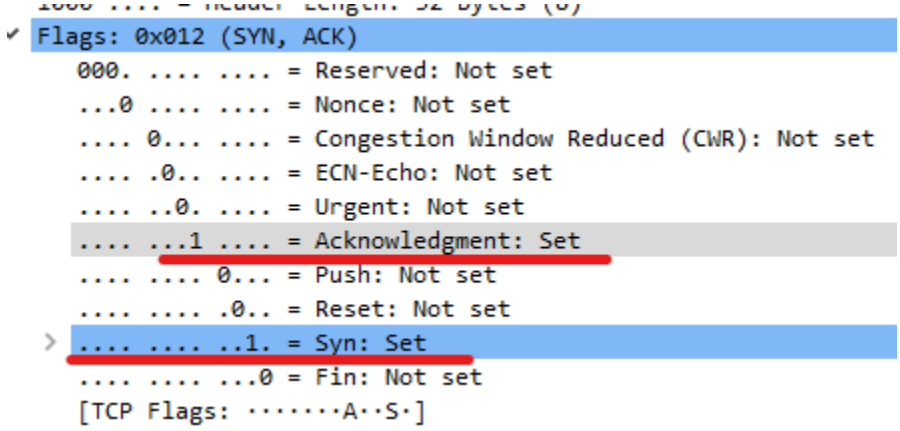


Sequence Number 0

The value of the SYN flag in the Flag segment is set to **1**

5. What is the **sequence number of the SYNACK** segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgment field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Ans:



Sequence Number of SYNACK segment is **0.** The acknowledgment field is **1.** The gaia.cs.umass.edu determines the value by adding 1 to the **initial sequence number** of the SYN segment from the client computer. The **Flag** segment determines SYN and ACK field values as mentioned in the above screenshot.

6. What is the **sequence number of the TCP segment** containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

Ans: **Sequence Number:** 1

7. 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 (including the segment containing the HTTP POST)? At what time was **each segment sent**? When was the ACK for each segment received? Given the difference between when each TCP segment was **sent**, and when its acknowledgment was **received**, what is the RTT value for each of the six Segments? What is the EstimatedRTT value (see Section 3.5.3, page 239 in text) after the receipt of each ACK?

Ans:

```
 764 63071 → 80 [PSH, ACK] Seq=1 Ack=1 Win=257 Len=710
1454 63071 → 80 [ACK] Seq=711 Ack=1 Win=257 Len=1400 [T
1454 63071 → 80 [ACK] Seq=2111 Ack=1 Win=257 Len=1400 [
1454 63071 → 80 [ACK] Seq=3511 Ack=1 Win=257 Len=1400 [
1454 63071 → 80 [ACK] Seq=4911 Ack=1 Win=257 Len=1400 [
1454 63071 → 80 [ACK] Seq=6311 Ack=1 Win=257 Len=1400 [
```

The first six segments including the HTTP POST segments are:

Segment 1: 1
Segment 2: 711
Segment 3: 2111
Segment 4: 3511
Segment 5: 4911
Segment 6: 6311

```
1.361827   128.119.245.12   192.168.43.91   TCP   54 80 → 63071 [ACK] Seq=1 Ack=711 Win=240 Len=0
1.379514   128.119.245.12   192.168.43.91   TCP   54 80 → 63071 [ACK] Seq=1 Ack=2111 Win=263 Len=0
1.399626   128.119.245.12   192.168.43.91   TCP   54 80 → 63071 [ACK] Seq=1 Ack=3511 Win=285 Len=0
1.419418   128.119.245.12   192.168.43.91   TCP   54 80 → 63071 [ACK] Seq=1 Ack=4911 Win=308 Len=0
1.594182   128.119.245.12   192.168.43.91   TCP   54 80 → 63071 [ACK] Seq=1 Ack=6311 Win=331 Len=0
1.594540   128.119.245.12   192.168.43.91   TCP   54 80 → 63071 [ACK] Seq=1 Ack=7711 Win=354 Len=0
```

The **time** for each segment to be **sent** is listed below: (tcp.seq == 1)

Segment 1: 1.026117
Segment 2: 1.026286
Segment 3: 1.026286
Segment 4: 1.026286
Segment 5: 1.026286
Segment 6: 1.026286

```
1.026286   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=711 Ack=1 Win=257 Len=1400
1.026286   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=2111 Ack=1 Win=257 Len=1400
1.026286   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=3511 Ack=1 Win=257 Len=1400
1.026286   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=4911 Ack=1 Win=257 Len=1400
1.026286   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=6311 Ack=1 Win=257 Len=1400
1.026286   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=7711 Ack=1 Win=257 Len=1400
```

The **time** for each segment to be **received** is listed below: (tcp.ack == 1)

Segment 1: 1.340091
Segment 2: 1.361827
Segment 3: 1.379514
Segment 4: 1.399626
Segment 5: 1.419418
Segment 6: 1.594182

The RTT value of each of the segments are:

RTT for segment 1: 0.313974
RTT for segment 2: 0.335541
RTT for segment 3: 0.353228
RTT for segment 4: 0.37334
RTT for segment 5: 0.393132
RTT for segment 6: 0.567896

Formulae for calculating Estimated RTT is:
Estimated RTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT
The estimated RTT for the following six packets is calculated as follows:

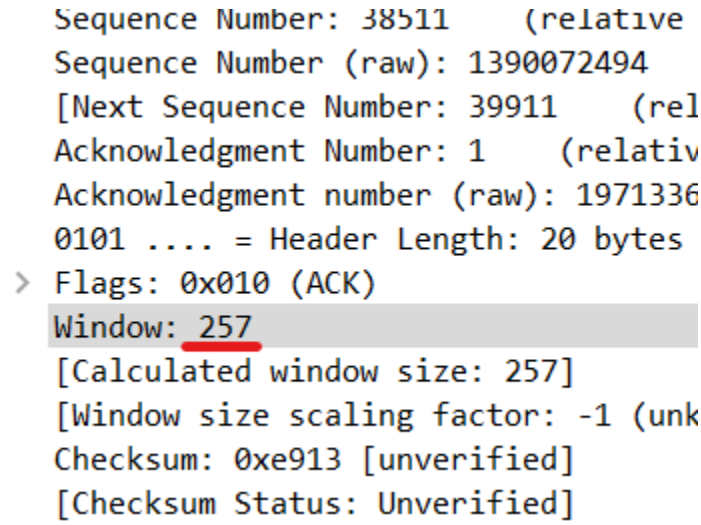
Estimated RTT for Segment 1: 0.313974
Estimated RTT for Segment 2: 0.875 * 0.313974 + 0.125 * 0.335541 = **0.316669875**
Estimated RTT for Segment 3: 0.875 * 0.316669 + 0.125 * 0.353228 = **0.318613**
Estimated RTT for Segment 4: 0.875 * 0.318613 + 0.125 * 0.37334 = **0.325453**
Estimated RTT for Segment 5: 0.875 * 0.325453 + 0.125 * 0.393132 = **0.333912**
Estimated RTT for Segment 6: 0.875 * 0.333912 + 0.125 * 0.567896 = **0.36316**

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

Ans: **1400**

9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the Sender?

Ans:



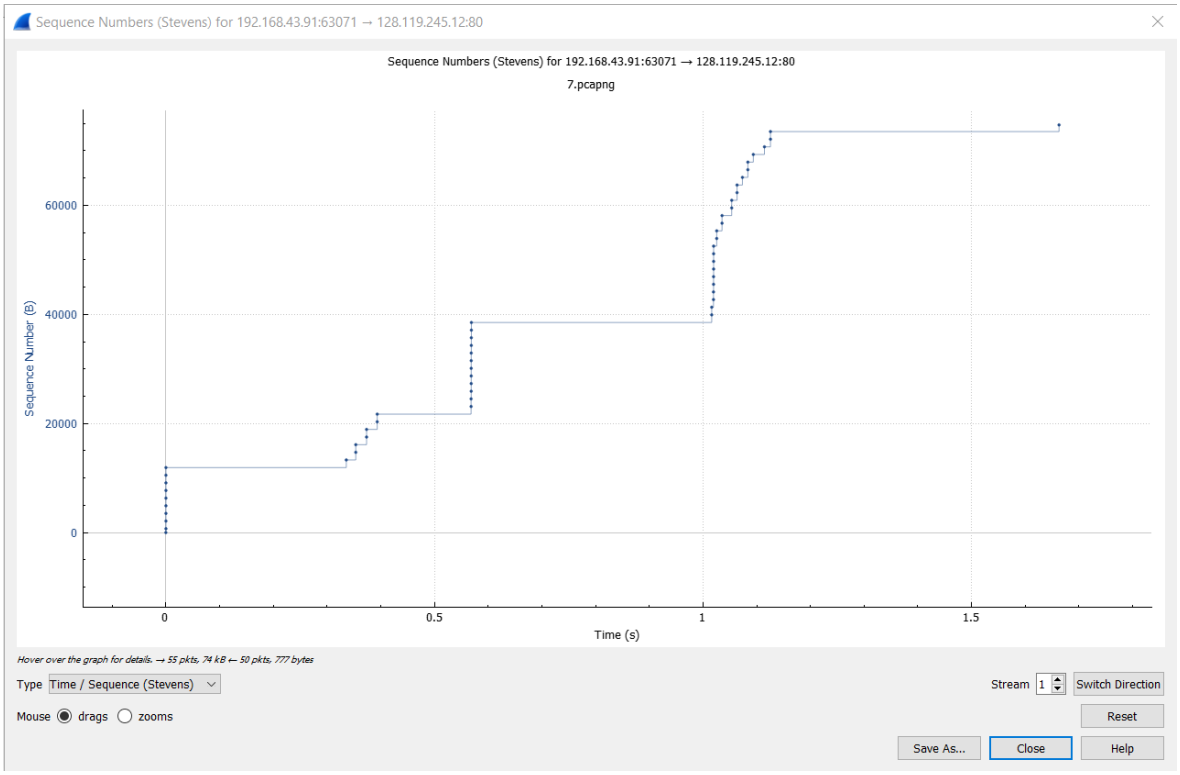
```
70 2.042401   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=39911 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
71 2.042401   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=41311 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
73 2.045659   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=42711 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
74 2.045659   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=44111 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
76 2.045814   192.168.43.91   128.119.245.12   TCP   1454 63071 → 80 [ACK] Seq=45511 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]

Sequence Number: 41311    (relative sequence number)
Sequence Number (raw): 1390075294
[Next Sequence Number: 42711    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 1971336848
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 257
[Calculated window size: 257]
```

The minimum amount of available buffer space at the receiver side is **257.** This receiver window grows up until it overflows the maximum buffer space of 257 bytes. Therefore by inspecting this trace, the sends **need not** throttle due to lacking receiver buffer space.

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

Ans:



No segments need to be retransmitted as the graph is always upward directed. The sequence numbers of the TCP segments in the trace file verify this statement.
The upward-directed graph denotes the sequence number is always non-decreasing, i.e increases monotonically with respect to time, therefore no packet is lost while transmitting the packets from client to server.

11. How much data does the **receiver typically acknowledge in an ACK**? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text)?

Ans:

```
1454 63071 → 80 [ACK] Seq=52511 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=53911 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=55311 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=56711 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=58111 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=59511 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=60911 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=62311 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
1454 63071 → 80 [ACK] Seq=63711 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
```

The receiver typically acknowledges 1400 bytes in an ACK. We can find those cases where the receiver is ACKing every other received segment by calculating the difference between two consecutive ACKs. Here if we calculate the last two ACKs we get:
63711 - 62311 = 1400 which is equal to the received segment size.

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

Ans:

```
 9 1.019885   192.168.43.91   128.119.245.12   TCP    54 63070 → 80 [FIN, ACK] Seq=1 Ack=1 Win=257 Len=0
12 1.026117   192.168.43.91   128.119.245.12   TCP   764 63071 → 80 [PSH, ACK] Seq=1 Ack=1 Win=257 Len=710 [TCP segment of a reassembled PDU]
13 1.026286   192.168.43.91   128.119.245.12   TCP  1454 63071 → 80 [ACK] Seq=711 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
14 1.026286   192.168.43.91   128.119.245.12   TCP  1454 63071 → 80 [ACK] Seq=2111 Ack=1 Win=257 Len=1400 [TCP segment of a reassembled PDU]
```

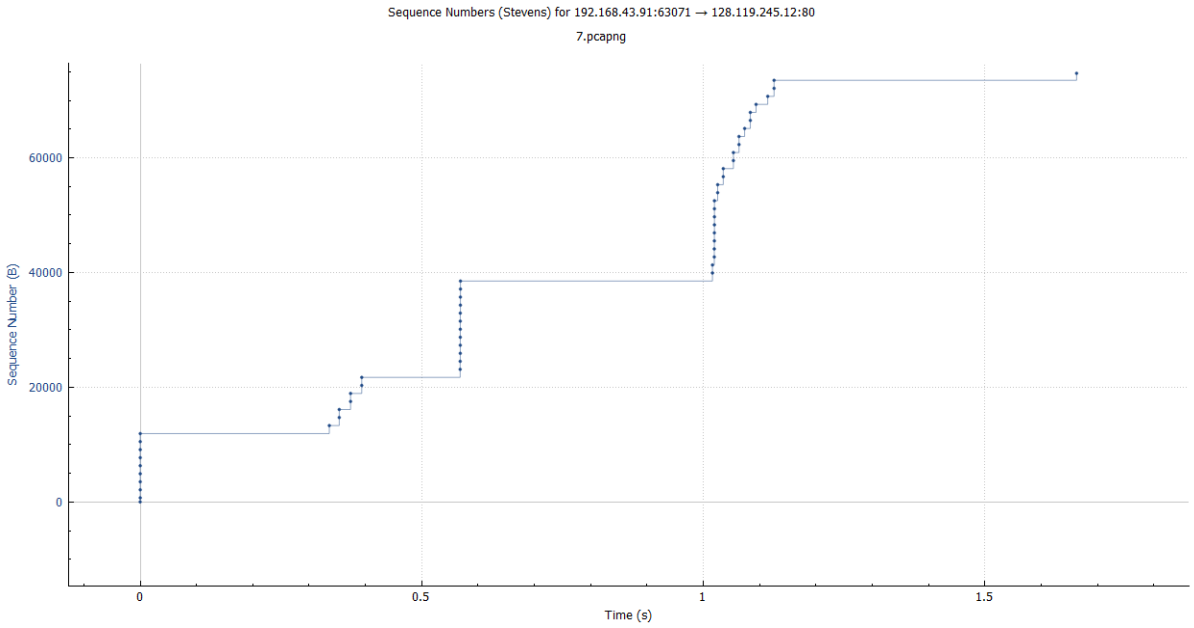
```
137 2.571632   128.119.245.12   192.168.43.91   TCP    54 80 → 63071 [ACK] Seq=1 Ack=67911 Win=1329 Len=0
138 2.571760   128.119.245.12   192.168.43.91   TCP    54 80 → 63071 [ACK] Seq=1 Ack=69311 Win=1352 Len=0
139 2.582476   128.119.245.12   192.168.43.91   TCP    54 80 → 63071 [ACK] Seq=1 Ack=70711 Win=1375 Len=0
140 2.634152   128.119.245.12   192.168.43.91   TCP    54 80 → 63071 [ACK] Seq=1 Ack=74729 Win=1419 Len=0
 12 1.026117   192.168.43.91   128.119.245.12   TCP   764 63071 → 80 [PSH, ACK] Seq=1 Ack=1 Win=257 Len=710
141 2.636950   128.119.245.12   192.168.43.91   HTTP  831 HTTP/1.1 200 OK  (text/html)
```

Throughput = (ACK of last segment - ACK of first segment) / (time for last segment - time for first segment)

Throughput = (74729-1)/(2.634152-1.026117) = 74728 / 1.608035 = **46471.62531**

13. Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus the time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slow-start phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text. Answer each of the two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu.

Ans:



As the graph suggests, it goes in an upward direction. Therefore the whole process is a slow-start phase starting from 0 and ends at 2.634152 where the last segment was sent. And as the slow-start phase takes place over the whole process there is no congestion avoidance in that case.