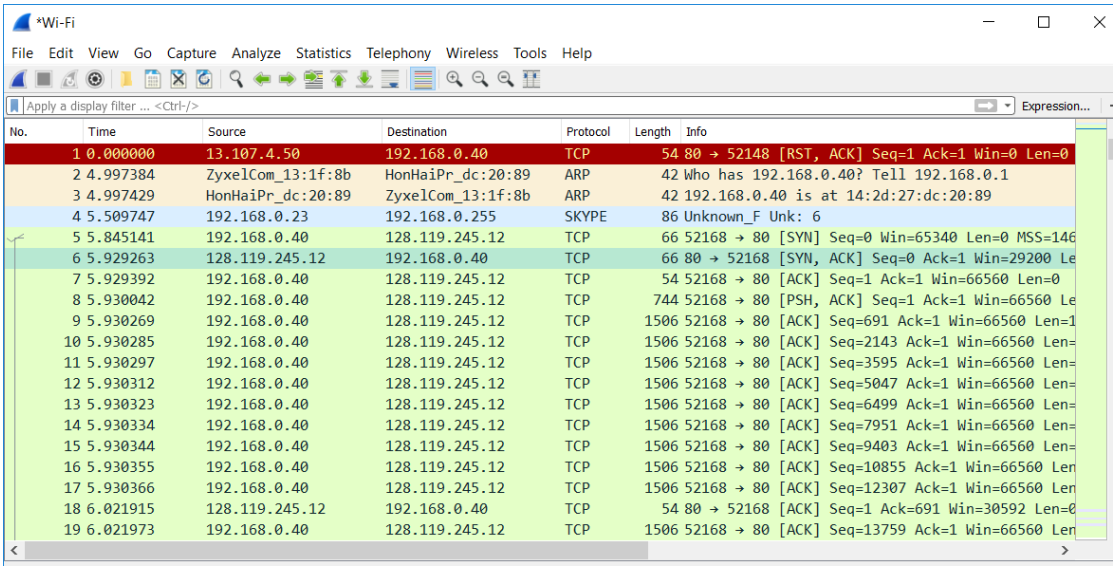
Assignment 2: Investigating TCP with Wireshark (30 Points)

Akshay Aravind Nayak

23368873

1. Capturing a bulk TCP transfer from your computer to a remote server



2. A first look at the captured trace

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:

```
> Internet Protocol Version 4, Src: 192.168.0.40, Dst: 128.119.245.12
v Transmission Control Protocol, Src Port: 52168, Dst Port: 80, Seq: 152389, Ack: 1, Len: 623
    Source Port: 52168
    Destination Port: 80
```

IP Address: 192.168.0.40
Port No : 52168

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?

```
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.0.40
Transmission Control Protocol, Src Port: 80, Dst Port: 52168, Seq: 1, Ack: 153012, Len: 777
   Source Port: 80
   Destination Port: 52168
   [Stream index: 1]
```

IP address: 128.119.245.12

Port No: 80

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

```
> Internet Protocol Version 4, Src: 192.168.0.40, Dst: 128.119.245.12
v Transmission Control Protocol, Src Port: 52168, Dst Port: 80, Seq: 152389, Ack: 1, Len: 623
    Source Port: 52168
    Destination Port: 80
```

IP Address: 192.168.0.40
Port No : 52168

Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages.

5	5.845141	192.168.0.40	128.119.245.12	TCP	66 52168 → 80 [
6	5.929263	128.119.245.12	192.168.0.40	TCP	66 80 → 52168 [
7	5.929392	192.168.0.40	128.119.245.12	TCP	54 52168 → 80 [
8	5.930042	192.168.0.40	128.119.245.12	TCP	744 52168 → 80 [
9	5.930269	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
10	5.930285	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
11	5.930297	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
12	5.930312	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
13	5.930323	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
14	5.930334	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
15	5.930344	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
16	5.930355	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
17	5.930366	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
18	6.021915	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [
19	6.021973	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [
20	6.037270	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [
21	6.037325	192.168.0.40	128.119.245.12	TCP	1506 52168 → 80 [

3. TCP Basics

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

```
v Transmission Control Protocol, Src Port: 52168, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 52168
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
```

```
Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ··········S·]
```

The sequence Number of TCP SYN segment is 0
In the Flags section SYN flag is set to 1 this identifies it as a SYN segment.

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 Acknowledgement 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?

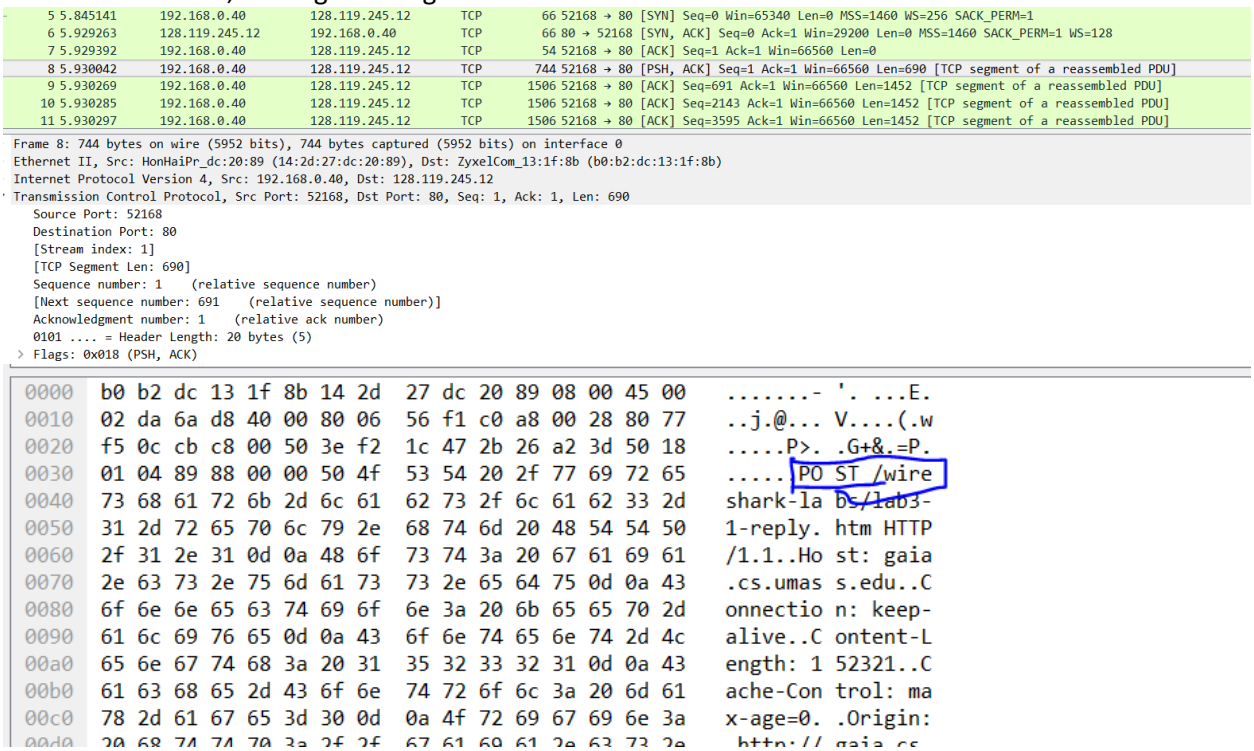
```
Transmission Control Protocol, Src Port: 80, Dst Port: 52168, Seq: 0, Ack: 1, Len: 0
   Source Port: 80
   Destination Port: 52168
   [Stream index: 1]
   [TCP Segment Len: 0]
   Sequence number: 0    (relative sequence number)
   Acknowledgment number: 1    (relative ack number)
   1000 .... = Header Length: 32 bytes (8)
 > Flags: 0x012 (SYN, ACK)
```

```
Flags: 0x012 (SYN, ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
 > .... .... ..1. = Syn: Set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····A··S·]
Window size value: 29200
[Calculated window size: 29200]
```

The sequence number of the SYN / ACK segment is 0
The Acknowledgement number of the SYN / ACK is 1
The Acknowledgement number is set to 1 to indicate that it received a packet with sequence number 0 from the client and the next packet it is accepting has a number of 1.

A segment is identified as a SYN/ACK segment if SYN and ACK flags are both set to 1

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.



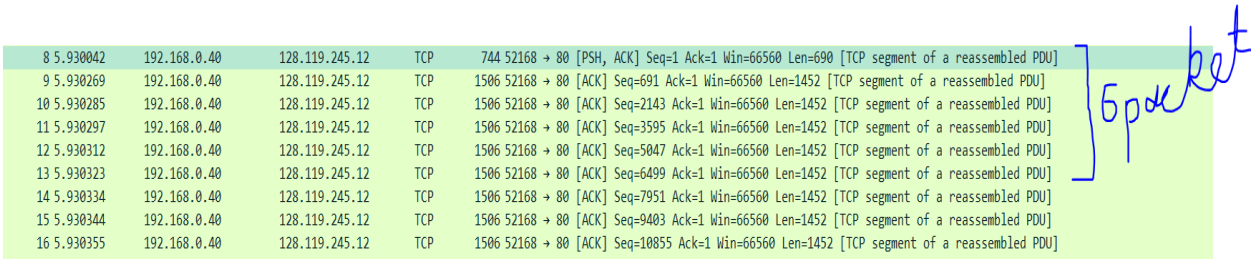
```
    5 5.845141    192.168.0.40      128.119.245.12    TCP      66 52168 → 80 [SYN] Seq=0 Win=65340 Len=0 MSS=1460 WS=256 SACK_PERM=1
    6 5.929263    128.119.245.12    192.168.0.40      TCP      66 80 → 52168 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
    7 5.929392    192.168.0.40      128.119.245.12    TCP      54 52168 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
    8 5.930042    192.168.0.40      128.119.245.12    TCP     744 52168 → 80 [PSH, ACK] Seq=1 Ack=1 Win=66560 Len=690 [TCP segment of a reassembled PDU]
    9 5.930269    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=691 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   10 5.930285    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=2143 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   11 5.930297    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=3595 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]

Frame 8: 744 bytes on wire (5952 bits), 744 bytes captured (5952 bits) on interface 0
Ethernet II, Src: HonHaiPr_dc:20:89 (14:2d:27:dc:20:89), Dst: ZyxelCom_13:1f:8b (b0:b2:dc:13:1f:8b)
Internet Protocol Version 4, Src: 192.168.0.40, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 52168, Dst Port: 80, Seq: 1, Ack: 1, Len: 690
    Source Port: 52168
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 690]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 691    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)

0000  b0 b2 dc 13 1f 8b 14 2d  27 dc 20 89 08 00 45 00   .......- '. ...E.
0010  02 da 6a d8 40 00 80 06  56 f1 c0 a8 00 28 80 77   ..j.@... V....(.w
0020  f5 0c cb c8 00 50 3e f2  1c 47 2b 26 a2 3d 50 18   .....P>. .G+&.=P.
0030  01 04 89 88 00 00 50 4f  53 54 20 2f 77 69 72 65   ......PO ST /wire
0040  73 68 61 72 6b 2d 6c 61  62 73 2f 6c 61 62 33 2d   shark-la bs/lab3-
0050  31 2d 72 65 70 6c 79 2e  68 74 6d 20 48 54 54 50   1-reply. htm HTTP
0060  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 67 61 69 61   /1.1..Ho st: gaia
0070  2e 63 73 2e 75 6d 61 73  73 2e 65 64 75 0d 0a 43   .cs.umas s.edu..C
0080  6f 6e 6e 65 63 74 69 6f  6e 3a 20 6b 65 65 70 2d   onnectio n: keep-
0090  61 6c 69 76 65 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   alive..C ontent-L
00a0  65 6e 67 74 68 3a 20 31  35 32 33 32 31 0d 0a 43   ength: 1 52321..C
00b0  61 63 68 65 2d 43 6f 6e  74 72 6f 6c 3a 20 6d 61   ache-Con trol: ma
00c0  78 2d 61 67 65 3d 30 0d  0a 4f 72 69 67 69 6e 3a   x-age=0. .Origin:
00d0  20 68 74 74 70 3a 2f 2f  67 61 69 61 2e 63 73 2e    http:// gaia.cs.
```

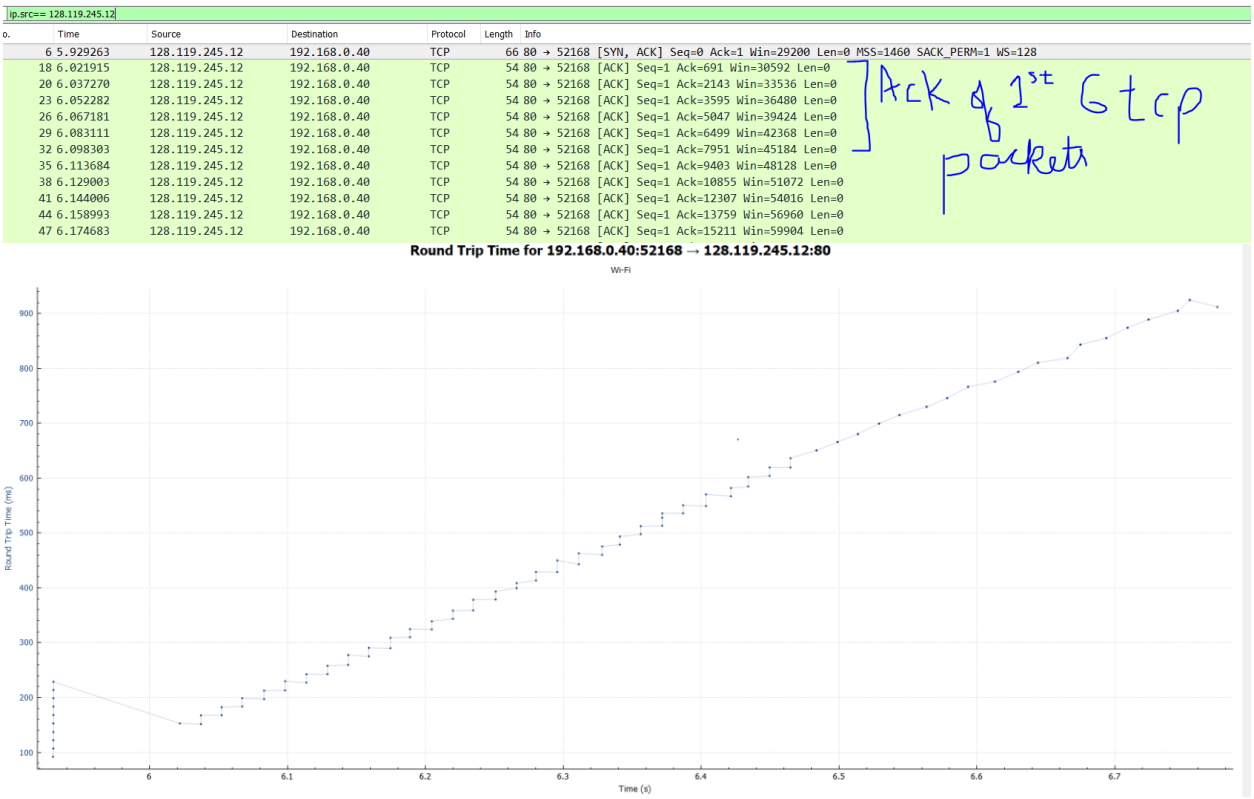
sequence number of the TCP segment containing the HTTP POST command is 1

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 acknowledgement 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? Assume that the value of the `EstimatedRTT` is equal to the measured RTT for the first segment, and then is computed using the `EstimatedRTT` equation on page 239 for all subsequent segments.



```
    8 5.930042    192.168.0.40      128.119.245.12    TCP     744 52168 → 80 [PSH, ACK] Seq=1 Ack=1 Win=66560 Len=690 [TCP segment of a reassembled PDU]
    9 5.930269    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=691 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   10 5.930285    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=2143 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   11 5.930297    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=3595 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   12 5.930312    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=5047 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   13 5.930323    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=6499 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   14 5.930334    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=7951 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   15 5.930344    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=9403 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
   16 5.930355    192.168.0.40      128.119.245.12    TCP    1506 52168 → 80 [ACK] Seq=10855 Ack=1 Win=66560 Len=1452 [TCP segment of a reassembled PDU]
```

6 packet



EstimatedRTT = 0.875 • EstimatedRTT + 0.125 • SampleRTT

	Packet Sent ti	ACK received	RTT	Estimated RT
Segment 1	5.930042	6.021915	0.091873	0.091873
Segment 2	5.930269	6.037270	0.107001	0.093764
Segment 3	5.930285	6.052282	0.121997	0.108876
Segment 4	5.930297	6.067181	0.136884	0.123858
Segment 5	5.930312	6.083111	0.152799	0.138873
Segment 6	5.930323	6.098303	0.16798	0.154697
Segment 6	5.930334	6.113684	0.18335	0.169901

Estimated RTT after the receipt of ACK of segment 1:

Estimated RTT = RTT for segment 1 = 0.091873

Estimated RTT after the receipt of ACK of segment 2:

Estimated RTT = 0.875 • 0.09183 + 0.125 • 0.107001 = 0.093764

Estimated RTT after the receipt of ACK of segment 3:

Estimated RTT = 0.875 • 0.093764 + 0.125 • 0.121997 = **0.108876**

Estimated RTT after the receipt of ACK of segment 4:

Estimated RTT = 0.875 • 0.108876 + 0.125 • 0.136884 = **0.123858**

Estimated RTT after the receipt of ACK of segment 5:

Estimated RTT = 0.875 • 0.123858 + 0.125 • 0.152799 = **0.138873**

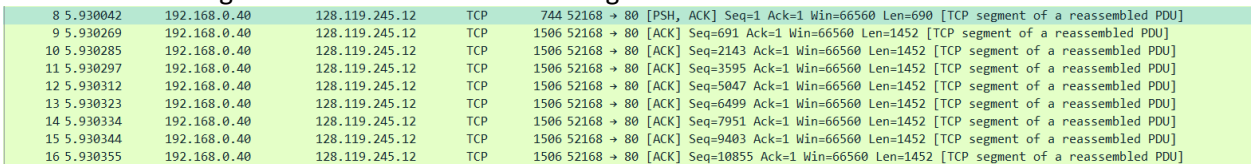
Estimated RTT after the receipt of ACK of segment 6:

Estimated RTT = 0.875 • 0.138873 + 0.125 • 0.16798 = **0.154697**

Estimated RTT after the receipt of ACK of segment 6:

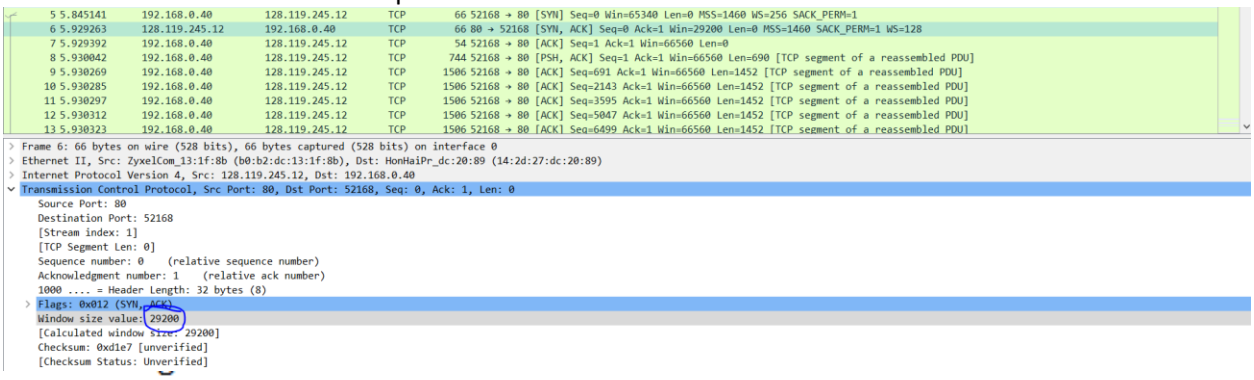
Estimated RTT = 0.875 • 0.154697 + 0.125 • 0.18335 = **0.169901**

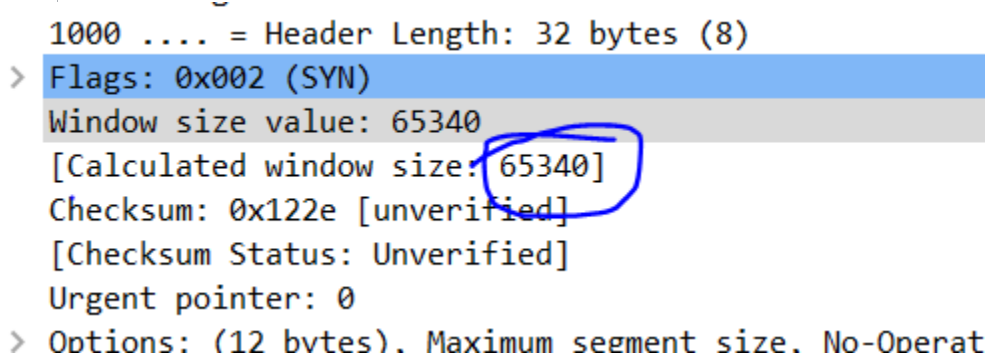
8. What is the length of each of the first six TCP segments?[1]



The length of first TCP segment is 690 bits and the next 5 TCP segment is 1452

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?

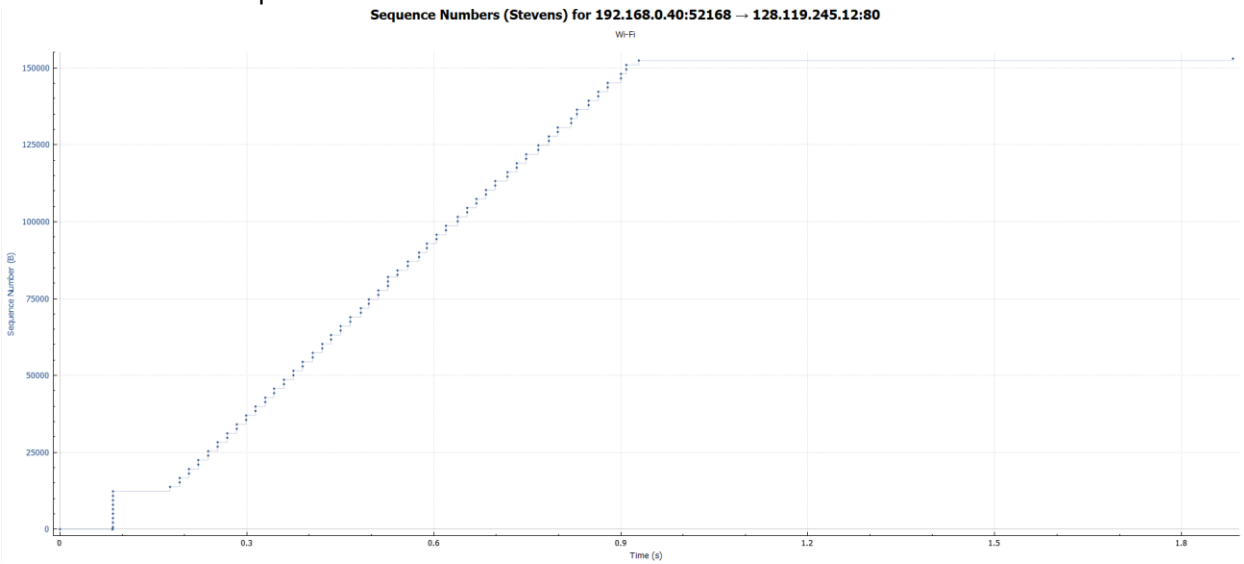




The minimum amount of buffer space (receiver window) advertised at gaia.cs.umass.edu for the entire trace is 29200 bytes, which shows in the first acknowledgement from the server. This receiver window

grows steadily until a maximum receiver buffer size of 65320 bytes. The sender is never throttled due to lacking of receiver buffer space by inspecting this trace.

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



The sequence numbers are increasing this proves the fact that there are no retransmitted segments in the trace file.

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).

18	6.021915	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=691 Win=30592 Len=0
20	6.037270	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=2143 Win=33536 Len=0
23	6.052282	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=3595 Win=36480 Len=0
26	6.067181	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=5047 Win=39424 Len=0
29	6.083111	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=6499 Win=42368 Len=0
32	6.098303	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=7951 Win=45184 Len=0
35	6.113684	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=9403 Win=48128 Len=0
38	6.129003	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=10855 Win=51072 Len=0
41	6.144006	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=12307 Win=54016 Len=0
44	6.158993	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=13759 Win=56960 Len=0
47	6.174683	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=15211 Win=59904 Len=0
50	6.188799	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=16663 Win=62720 Len=0
53	6.204584	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=18115 Win=65664 Len=0
56	6.219913	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=19567 Win=68608 Len=0
59	6.234644	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=21019 Win=71552 Len=0
62	6.250885	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=22471 Win=74496 Len=0
65	6.265929	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=23923 Win=77312 Len=0
68	6.280195	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=25375 Win=80256 Len=0
71	6.295576	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=26827 Win=83200 Len=0
74	6.311203	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=28279 Win=86144 Len=0
77	6.328127	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=29731 Win=89088 Len=0
80	6.340921	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=31183 Win=91904 Len=0
83	6.356107	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=32635 Win=94848 Len=0
86	6.371630	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=34087 Win=97792 Len=0
90	6.386800	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=35539 Win=100736 Len=0
93	6.403401	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=36991 Win=103680 Len=0
96	6.421477	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=38443 Win=106624 Len=0
99	6.433999	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=39895 Win=109440 Len=0
102	6.449509	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=41347 Win=112384 Len=0
105	6.464618	128.119.245.12	192.168.0.40	TCP	54 80 → 52168 [ACK] Seq=1 Ack=42799 Win=115328 Len=0

The difference between the next ACK number is 1452 , the receiver is Acknowledging 1452 bits of data.