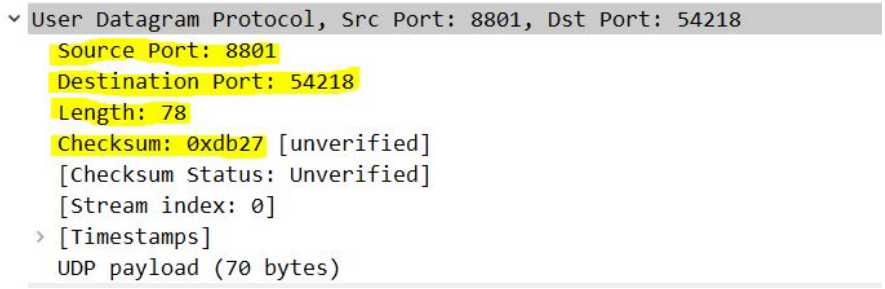
Part 1 – UDP Transport Layer Protocol

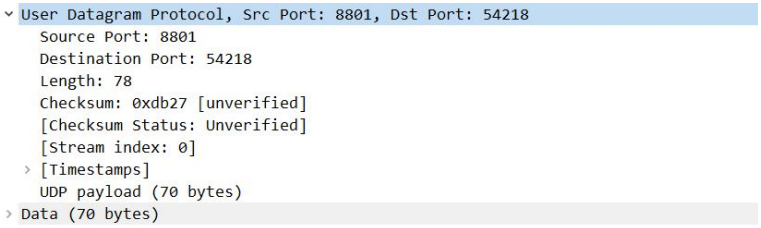
1.Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. Name these fields.

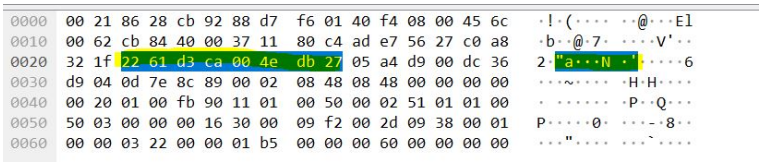
There are 4 fields in the UDP header which are: Source Port, Destination Port, Length, Checksum.



2.By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.

Each of the header fields are 2 bytes long since the UDP header always has a fixed amount of 8 bytes.





3.The value in the Length field is the length of what? (Consult the textbook for this answer). Verify your claim with your captured UDP packet. Show your work details.

The value in the length field is the sum of the 8 bytes from the header plus 70 bytes from the data.

```
User Datagram Protocol, Src Port: 8801, Dst Port: 54218
    Source Port: 8801
    Destination Port: 54218
    Length: 78
    Checksum: 0xdb27 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
    UDP payload (70 bytes)
> Data (70 bytes)
```

4.What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)

The maximum number of bytes that can be included in a UDP payload is 2^{16}-8 = 65527 bytes.

5.What is the largest possible source port number?

The largest possible source port number is $2^{16} - 1 = 65535$.

6.What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation.

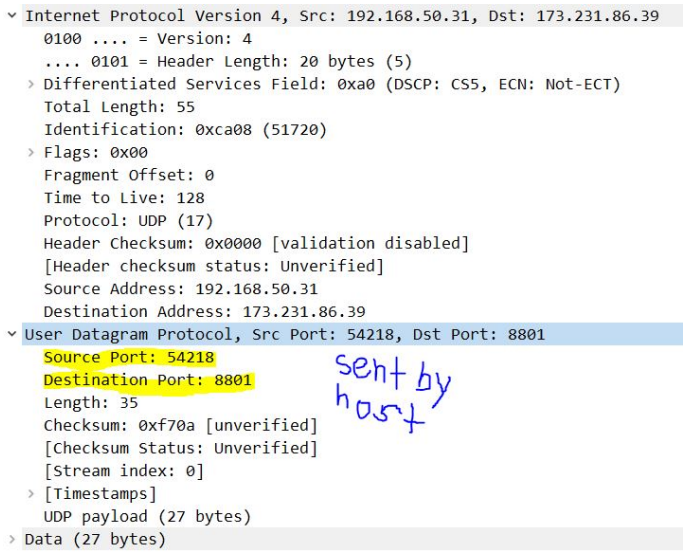
The protocol number for UDP is 17 and 0x11 hex.

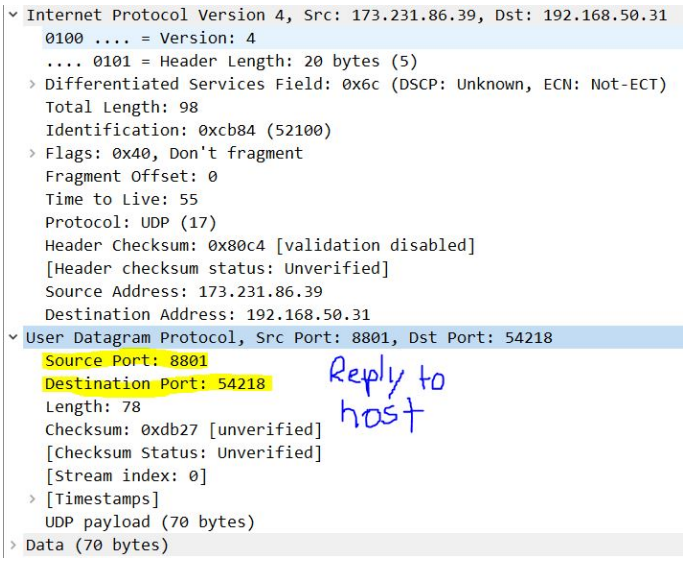
```
Internet Protocol Version 4, Src: 173.231.86.39, Dst: 192.168.50.31
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x6c (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 98
    Identification: 0xcb84 (52100)
  > Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 55
    Protocol: UDP (17)
    Header Checksum: 0x80c4 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 173.231.86.39
    Destination Address: 192.168.50.31
User Datagram Protocol, Src Port: 8801, Dst Port: 54218
    Source Port: 8801
    Destination Port: 54218
    Length: 78
    Checksum: 0xdb27 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
```

```
0000  00 21 86 28 cb 92 88 d7  f6 01 40 f4 08 00 45 6c   ·!·(···· ··@···El
0010  00 62 cb 84 40 00 37 11  80 c4 ad e7 56 27 c0 a8   ·b··@·7· ····V'··
0020  32 1f 22 61 d3 ca 00 4e  db 27 05 a4 d9 00 dc 36   2·"a···N ·'·····6
0030  d9 04 0d 7e 8c 89 00 02  08 48 08 48 00 00 00 00   ···~···· ·H·H····
0040  00 20 01 00 fb 90 11 01  00 50 00 02 51 01 01 00   · ····· ·P··Q···
0050  50 03 00 00 00 16 30 00  09 f2 00 2d 09 38 00 01   P·····0· ···-·8··
0060  00 00 03 22 00 00 01 b5  00 00 00 60 00 00 00 00   ···"···· ···`····
```

7.Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). Describe the relationship between the port numbers in the two packets.

The source port in the UDP packet that was sent by the host is the same as the destination port of the reply UDP packet. And the destination port in the UDP packet that was sent by the host is the same as the source port of the reply UDP packet

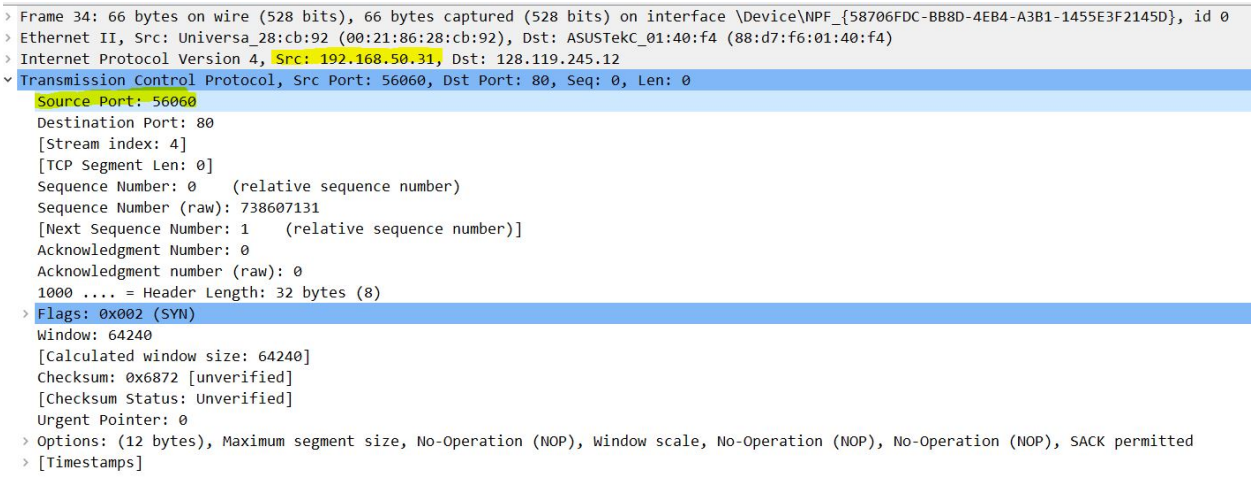




Part 2 – TCP Transport Layer Protocool

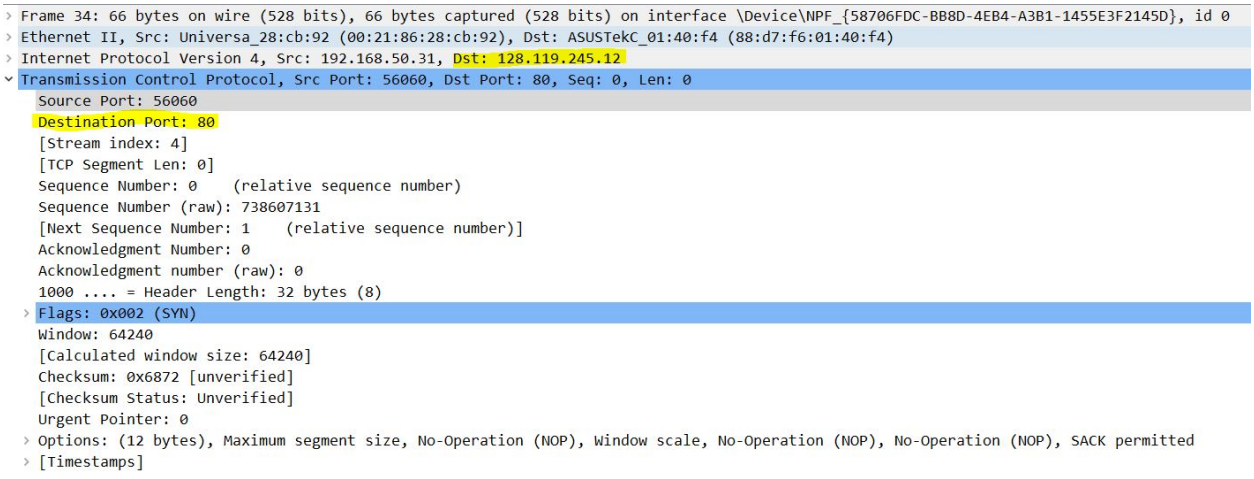
1. From the packet that contains the initial three-way handshake SYN message, what is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to enable HTTP protocol analysis, filter for only http, and 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".

The IP address is 192.168.50.31 and the source port is 56060.



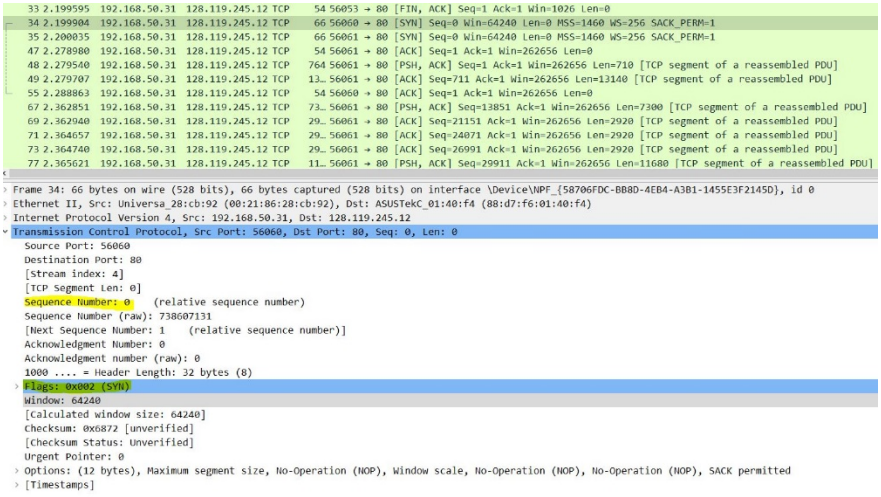
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

The IP address is 128.119.245.12 and the destination port is 80.



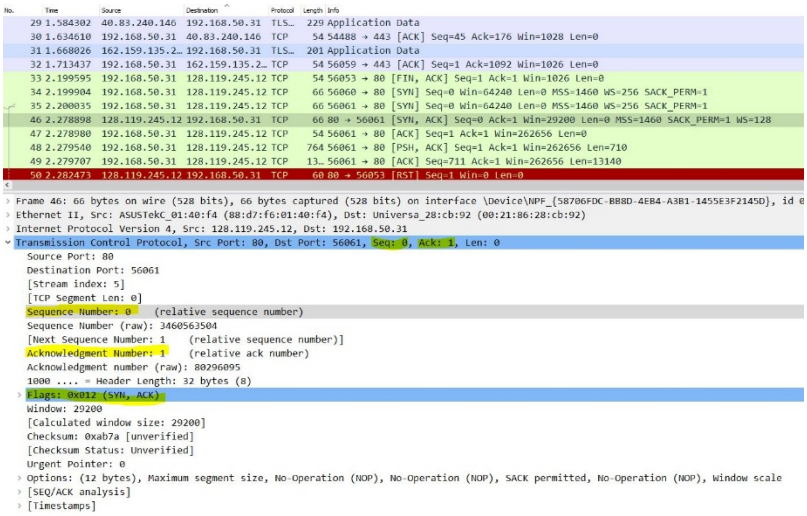
3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

The sequence number that is used to initiate the TCP connection is 0. It shows that it contains a SYN flag.



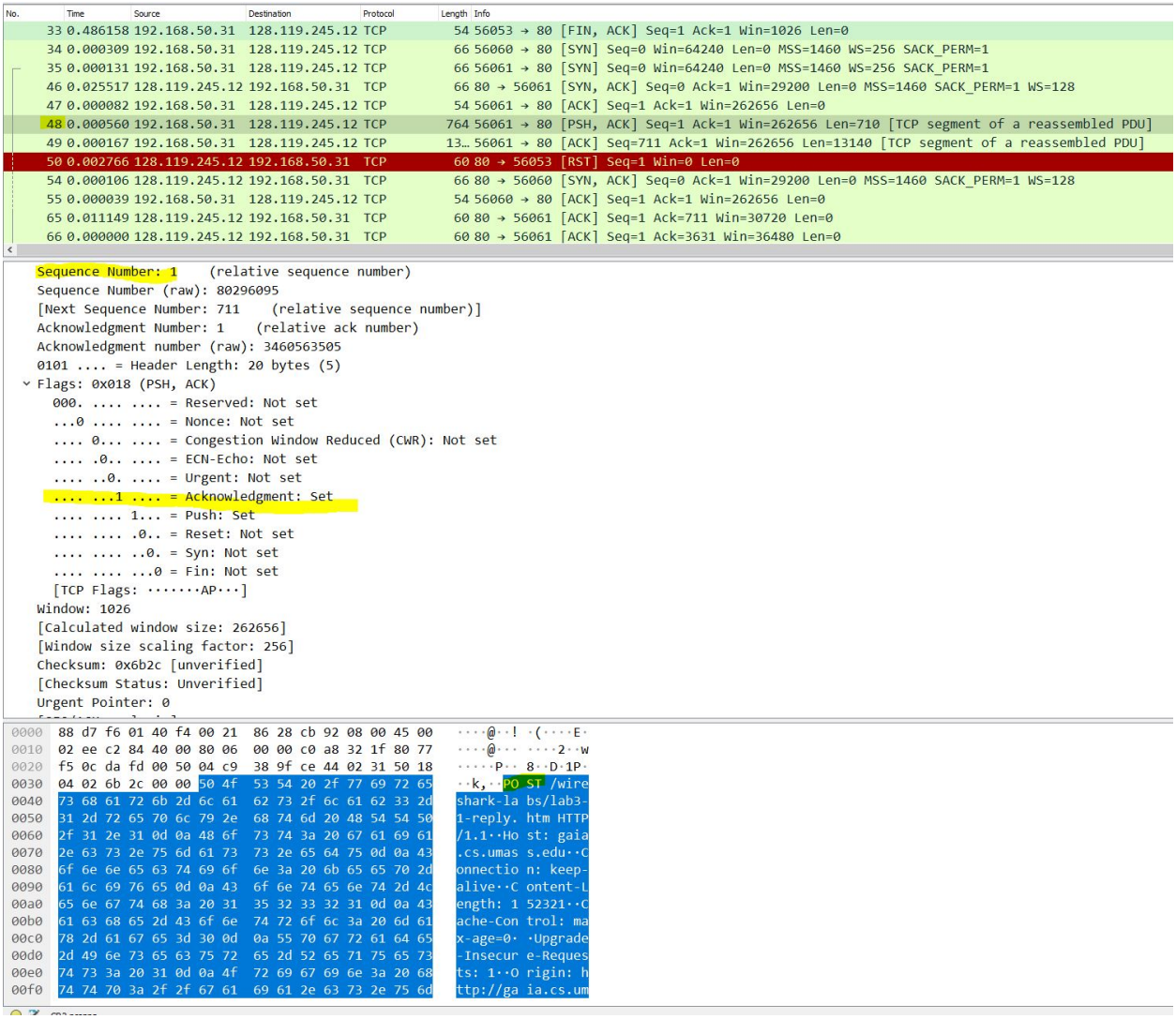
4. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

The sequence number of the SYNACK segment is 0. The value of the acknowledgement field is 1. It is determined by (initial sequence number + 1). The flags show that it is a SYNACK segment.



5. What is the sequence number of the TCP segment containing the HTTP POST command? Note that 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.

Packet number 48 is the TCP segment containing the HTTP POST command. The sequence number has a value of 1.



No.	Time	Source	Destination	Protocol	Length	Info
33	0.486158	192.168.50.31	128.119.245.12	TCP	54	56053 → 80 [FIN, ACK] Seq=1 Ack=1 Win=1026 Len=0
34	0.000309	192.168.50.31	128.119.245.12	TCP	66	56060 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
35	0.000131	192.168.50.31	128.119.245.12	TCP	66	56061 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
46	0.025517	128.119.245.12	192.168.50.31	TCP	66	80 → 56061 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
47	0.000082	192.168.50.31	128.119.245.12	TCP	54	56061 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
48	0.000560	192.168.50.31	128.119.245.12	TCP	764	56061 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656 Len=710 [TCP segment of a reassembled PDU]
49	0.000167	192.168.50.31	128.119.245.12	TCP	13…	56061 → 80 [ACK] Seq=711 Ack=1 Win=262656 Len=13140 [TCP segment of a reassembled PDU]
50	0.002766	128.119.245.12	192.168.50.31	TCP	60	80 → 56053 [RST] Seq=1 Win=0 Len=0
54	0.000106	128.119.245.12	192.168.50.31	TCP	66	80 → 56060 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
55	0.000039	192.168.50.31	128.119.245.12	TCP	54	56060 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
65	0.011149	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=711 Win=30720 Len=0
66	0.000000	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=3631 Win=36480 Len=0

```
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 80296095
[Next Sequence Number: 711    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 3460563505
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, 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
   .... .... 1... = Push: Set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····AP···]
Window: 1026
[Calculated window size: 262656]
[Window size scaling factor: 256]
Checksum: 0x6b2c [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
```

```
0000  88 d7 f6 01 40 f4 00 21  86 28 cb 92 08 00 45 00   ····@··! ·(····E·
0010  02 ee c2 84 40 00 80 06  00 00 c0 a8 32 1f 80 77   ····@··· ····2··w
0020  f5 0c da fd 00 50 04 c9  38 9f ce 44 02 31 50 18   ·····P·· 8··D·1P·
0030  04 02 6b 2c 00 00 50 4f  53 54 20 2f 77 69 72 65   ··k,··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 55 70 67 72 61 64 65   x-age=0· ·Upgrade
00d0  2d 49 6e 73 65 63 75 72  65 2d 52 65 71 75 65 73   -Insecur e-Reques
00e0  74 73 3a 20 31 0d 0a 4f  72 69 67 69 6e 3a 20 68   ts: 1··O rigin: h
00f0  74 74 70 3a 2f 2f 67 61  69 61 2e 63 73 2e 75 6d   ttp://ga ia.cs.um
```

6. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection.

Create a table to show the following data (from a to g below):

a. What are the sequence numbers of the first six segments in the TCP connection (starting with the segment containing the HTTP POST)?

Segment 1: sequence number 1, (raw) 80296095

Segment 2: sequence number 711, (raw) 80296805

Segment 3: sequence number 13851, (raw) 80309945

Segment 4: sequence number 21151, (raw) 80317245

Segment 5: sequence number 24071, (raw) 80320165

Segment 6: sequence number 26991, (raw) 80323085

b. At what time was each segment sent?

Segment 1: 2.279540s

Segment 2: 2.279707s

Segment 3: 2.362851s

Segment 4: 2.362940s

Segment 5: 2.364657s

Segment 6: 2.364740s

c. When was the ACK for each segment received?

Segment 1: ACK 711, (raw) 80296805, 2.362781s

Segment 2: ACK 3631, (raw) 80299725, 2.362781s

Segment 3: ACK 5091, (raw) 80301185, 2.362920s

Segment 4: ACK 6551, (raw) 80302645, 2.364636s

Segment 5: ACK 8011, (raw) 80304105, 2.364705s

Segment 6: ACK 10931, (raw) 80307025, 2.365601s

d. 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?

Segment 1: 0.083241s

Segment 2: 0.083074s

Segment 3: 0s

Segment 4: 0.001696s

Segment 5: 0.000048s

Segment 6: 0.000861s

e. What is the EstimatedRTT value (see Section 3.5.3, page 242 in textbook) 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 242 for all subsequent segments. Show your work details.

EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT

Segment 1: 0.083241s

Segment 2: 0.875 * 0.083241 + 0.125 * 0.083074 = 0.0832s

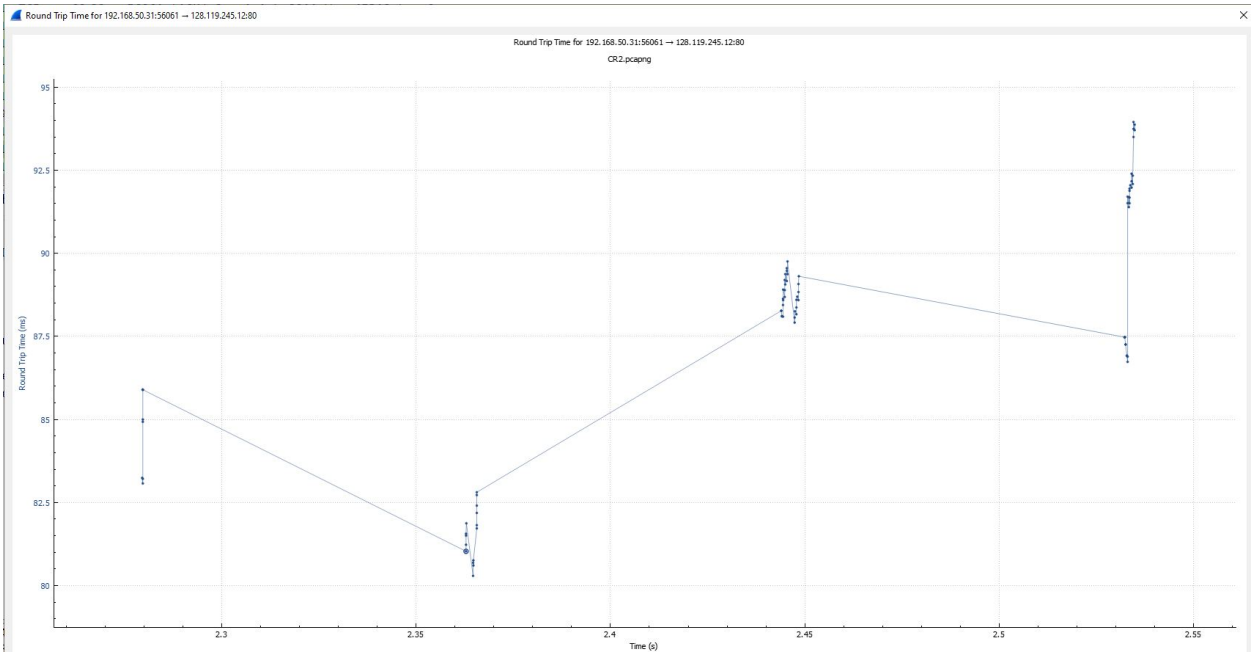
Segment 3: 0.875 * 0.0832 + 0.125 * 0 = 0.0728s

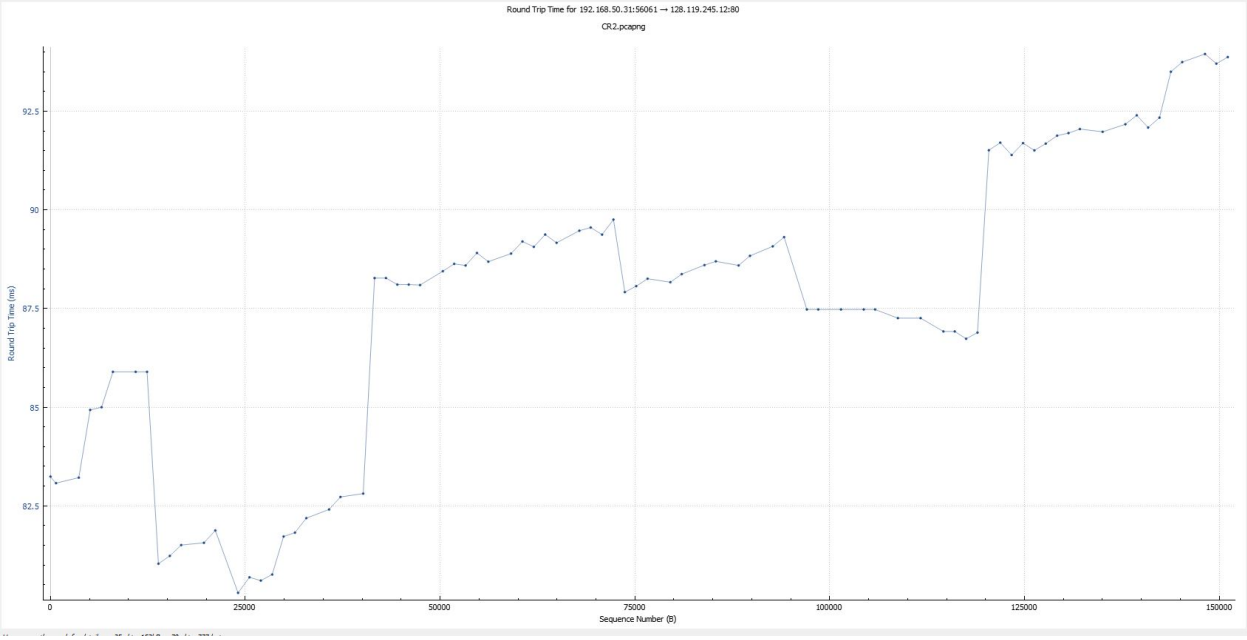
Segment 4: 0.875 * 0.0728 + 0.125 * 0.001696 = 0.0639s

Segment 5: 0.875 * 0.0639 + 0.125 * 0.000048 = 0.0559s

Segment 6: 0.875 * 0.0559 + 0.125 * 0.000861 = 0.049s

f. Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph->Round Trip Time Graph. Include the roundtrip Time Graph in your report.





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

Segment 1: 710 bytes

Segment 2: 13140 bytes

Segment 3: 7300 bytes

Segment 4: 2920 bytes

Segment 5: 2920 bytes

Segment 6: 2920 bytes

48	0.000560	192.168.50.31	128.119.245.12	TCP	764	56061 → 80	[PSH, ACK] Seq=1 Ack=1 Win=262656 Len=710 [TCP segment of a reassembled PDU]
49	0.000167	192.168.50.31	128.119.245.12	TCP	13…	56061 → 80	[ACK] Seq=711 Ack=1 Win=262656 Len=13140 [TCP segment of a reassembled PDU]
50	0.002766	128.119.245.12	192.168.50.31	TCP	60	80 → 56053	[RST] Seq=1 Win=0 Len=0
54	0.000106	128.119.245.12	192.168.50.31	TCP	66	80 → 56060	[SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
55	0.000039	192.168.50.31	128.119.245.12	TCP	54	56060 → 80	[ACK] Seq=1 Ack=1 Win=262656 Len=0
65	0.011149	128.119.245.12	192.168.50.31	TCP	60	80 → 56061	[ACK] Seq=1 Ack=711 Win=30720 Len=0
66	0.000000	128.119.245.12	192.168.50.31	TCP	60	80 → 56061	[ACK] Seq=1 Ack=3631 Win=36480 Len=0
67	0.000070	192.168.50.31	128.119.245.12	TCP	73…	56061 → 80	[PSH, ACK] Seq=13851 Ack=1 Win=262656 Len=7300 [TCP segment of a reassembled PDU]
68	0.000069	128.119.245.12	192.168.50.31	TCP	60	80 → 56061	[ACK] Seq=1 Ack=5091 Win=39424 Len=0
69	0.000020	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80	[ACK] Seq=21151 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
70	0.001696	128.119.245.12	192.168.50.31	TCP	60	80 → 56061	[ACK] Seq=1 Ack=6551 Win=42368 Len=0
71	0.000021	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80	[ACK] Seq=24071 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
72	0.000048	128.119.245.12	192.168.50.31	TCP	60	80 → 56061	[ACK] Seq=1 Ack=8011 Win=45312 Len=0
73	0.000035	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80	[ACK] Seq=26991 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]

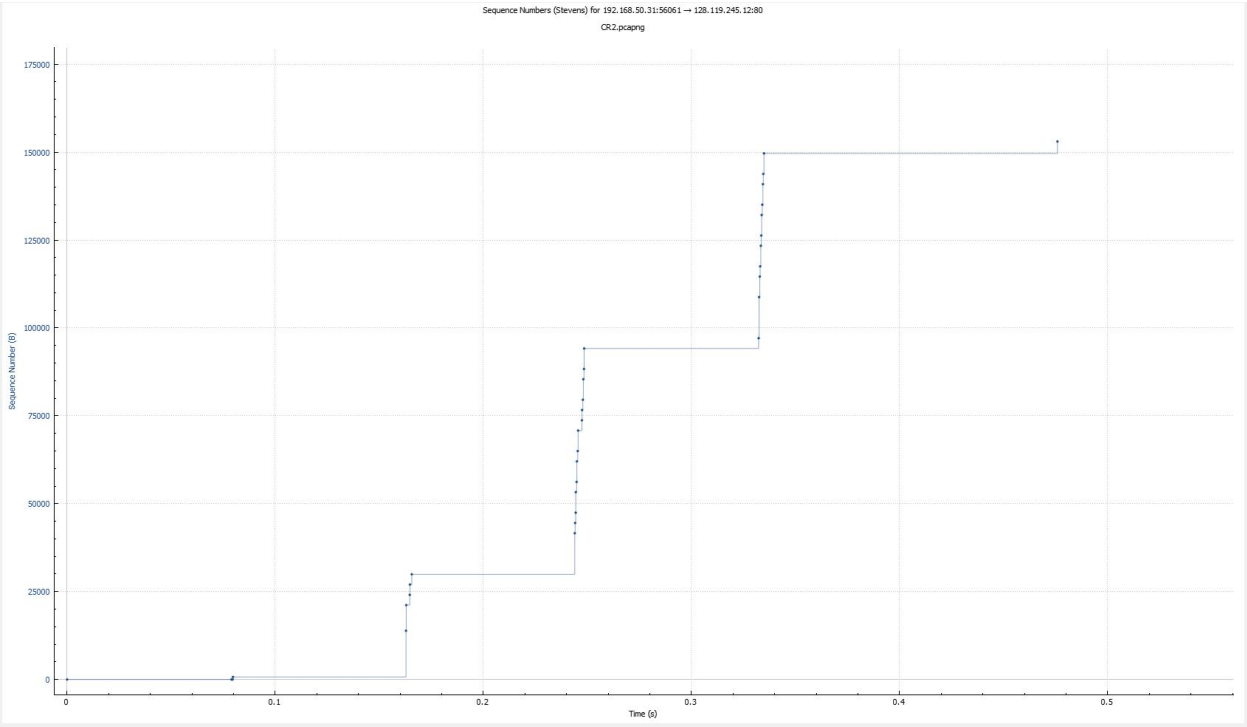
7. What is the minimum amount of available buffer space advertised at the received for the entire trace? Indicate which captured packet number in the trace. Does the lack of receiver buffer space ever throttle the sender? Explain.

The minimum amount of available buffer space advertised is 29200 bytes which is indicated in the 46 packet capture. No, the lack of receiver buffer space never throttled because we never get close to the max capacity of the window.

No.	Time	Source	Destination	Protocol	Length	Info
33	0.486158	192.168.50.31	128.119.245.12	TCP	54	56053 → 80 [FIN, ACK] Seq=1 Ack=1 Win=1026 Len=0
34	0.000309	192.168.50.31	128.119.245.12	TCP	66	56060 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
35	0.000131	192.168.50.31	128.119.245.12	TCP	66	56061 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
46	0.025517	128.119.245.12	192.168.50.31	TCP	66	80 → 56061 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
47	0.000082	192.168.50.31	128.119.245.12	TCP	54	56061 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
48	0.000560	192.168.50.31	128.119.245.12	TCP	764	56061 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656 Len=710 [TCP segment of a reassembled PDU]
49	0.000167	192.168.50.31	128.119.245.12	TCP	13…	56061 → 80 [ACK] Seq=711 Ack=1 Win=262656 Len=13140 [TCP segment of a reassembled PDU]
50	0.002766	128.119.245.12	192.168.50.31	TCP	60	80 → 56053 [RST] Seq=1 Win=0 Len=0
54	0.000106	128.119.245.12	192.168.50.31	TCP	66	80 → 56060 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
55	0.000039	192.168.50.31	128.119.245.12	TCP	54	56060 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
65	0.011149	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=711 Win=30720 Len=0
66	0.000000	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=3631 Win=36480 Len=0
67	0.000070	192.168.50.31	128.119.245.12	TCP	73…	56061 → 80 [PSH, ACK] Seq=13851 Ack=1 Win=262656 Len=7300 [TCP segment of a reassembled PDU]
68	0.000069	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=5091 Win=39424 Len=0
69	0.000020	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80 [ACK] Seq=21151 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
70	0.001696	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=6551 Win=42368 Len=0
71	0.000021	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80 [ACK] Seq=24071 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
72	0.000048	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=8011 Win=45312 Len=0
73	0.000035	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80 [ACK] Seq=26991 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
74	0.000861	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=10931 Win=51072 Len=0
75	0.000000	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=12391 Win=54016 Len=0
76	0.000000	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=13851 Win=56960 Len=0
77	0.000020	192.168.50.31	128.119.245.12	TCP	11…	56061 → 80 [PSH, ACK] Seq=29911 Ack=1 Win=262656 Len=11680 [TCP segment of a reassembled PDU]
80	0.057533	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=15311 Win=59904 Len=0
81	0.000056	192.168.50.31	128.119.245.12	TCP	29…	56061 → 80 [ACK] Seq=41591 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
82	0.000142	128.119.245.12	192.168.50.31	TCP	60	80 → 56061 [ACK] Seq=1 Ack=16771 Win=62848 Len=0

8. Are there any retransmitted segments in the trace file? What did you check for (in the trace) to answer this question? Explain based on and include the Statistics/TCP Stream Graphs/Time Sequence (Stevens) graph.

There is no retransmitted segment in the trace file. The way I was able to check was by looking at the time sequence graph and seeing the graph increase monotonically with respect to time and not seeing any segment lower than then the previous one.



9. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment?

	Acknowledge sequence number	Acknowledge data
ACK 1	711	711
ACK 2	3631	2920
ACK 3	5091	1460
ACK 4	6551	1460
ACK 5	8011	1460

The receiver typically acknowledges 1460 bytes. There are also some cases where the receiver is ACKing every other received segment such as in segment number 65-66 and 109-111 acknowledged data with 2920 bytes.

```
 65 0.011149 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=711 Win=30720 Len=0
 66 0.000000 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=3631 Win=36480 Len=0
 67 0.000070 192.168.50.31  128.119.245.12 TCP    73… 56061 → 80 [PSH, ACK] Seq=13851 Ack=1 Win=262656 Len=7300 [TCP segment of a reassembled PDU]
 68 0.000069 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=5091 Win=39424 Len=0
 69 0.000020 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=21151 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 70 0.001696 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=6551 Win=42368 Len=0
 71 0.000021 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=24071 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 72 0.000048 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=8011 Win=45312 Len=0
 73 0.000035 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=26991 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 74 0.000861 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=10931 Win=51072 Len=0
 75 0.000000 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=12391 Win=54016 Len=0
 76 0.000000 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=13851 Win=56960 Len=0
 77 0.000020 192.168.50.31  128.119.245.12 TCP    11… 56061 → 80 [PSH, ACK] Seq=29911 Ack=1 Win=262656 Len=11680 [TCP segment of a reassembled PDU]
 80 0.057533 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=15311 Win=59904 Len=0
 81 0.000056 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=41591 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 82 0.000142 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=16771 Win=62848 Len=0
 83 0.000023 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=44511 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 84 0.000252 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=19691 Win=68608 Len=0
 85 0.000027 192.168.50.31  128.119.245.12 TCP    58… 56061 → 80 [PSH, ACK] Seq=47431 Ack=1 Win=262656 Len=5840 [TCP segment of a reassembled PDU]
 86 0.000030 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=21151 Win=71552 Len=0
 87 0.000011 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=53271 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 88 0.000391 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=24071 Win=77440 Len=0
 89 0.000014 192.168.50.31  128.119.245.12 TCP    58… 56061 → 80 [ACK] Seq=56191 Ack=1 Win=262656 Len=5840 [TCP segment of a reassembled PDU]
 90 0.000122 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=25531 Win=80384 Len=0
 91 0.000014 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=62031 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 92 0.000380 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=26991 Win=83200 Len=0
 93 0.000000 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=28451 Win=86144 Len=0
 94 0.000015 192.168.50.31  128.119.245.12 TCP    58… 56061 → 80 [PSH, ACK] Seq=64951 Ack=1 Win=262656 Len=5840 [TCP segment of a reassembled PDU]
 95 0.000139 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=29911 Win=89088 Len=0
 96 0.000041 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=70791 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 97 0.001803 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=31371 Win=92032 Len=0
 98 0.000036 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=73711 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
 99 0.000061 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=32831 Win=94976 Len=0
100 0.000013 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=76631 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
101 0.000355 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=35751 Win=100736 Len=0
102 0.000021 192.168.50.31  128.119.245.12 TCP    58… 56061 → 80 [PSH, ACK] Seq=79551 Ack=1 Win=262656 Len=5840 [TCP segment of a reassembled PDU]
103 0.000199 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=37211 Win=103680 Len=0
104 0.000016 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=85391 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
105 0.000301 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=40131 Win=109568 Len=0
106 0.000017 192.168.50.31  128.119.245.12 TCP    58… 56061 → 80 [ACK] Seq=88311 Ack=1 Win=262656 Len=5840 [TCP segment of a reassembled PDU]
107 0.000068 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=41591 Win=112384 Len=0
108 0.000019 192.168.50.31  128.119.245.12 TCP    29… 56061 → 80 [ACK] Seq=94151 Ack=1 Win=262656 Len=2920 [TCP segment of a reassembled PDU]
109 0.083760 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=43051 Win=115328 Len=0
110 0.000000 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=45971 Win=121216 Len=0
111 0.000000 128.119.245.12 192.168.50.31  TCP      60 80 → 56061 [ACK] Seq=1 Ack=47431 Win=124160 Len=0
```

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

Throughput = amount of data transmitted/time incurred

Amount of data transmitted = 149631bytes

Time incurred = 2.534842 – 2.362781 = 0.172061s

Throughput = 149631/0.172061 = 848.536 Kbits/second

The amount of data transmitted can be found by the sequence number of segment 133. To find the time it incurred you take the difference between the first TCP segment number 65 and the last segment number 133.

11. From the Time Sequence Graph (Stevens) plotting showing the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

a. Identify where TCP's Slow-Start phase begins and ends?

It seems to begin the slow-start phase at 0 seconds to 0.15 seconds.

b. Identify where Congestion Avoidance takes over?

Congestion avoidance takes over at 0.25 seconds to 0.35 seconds.

c. Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the textbook.

It differs from the perfect exponential plotted slow start graph we see in the textbook and the better indicated vertical graphs showing congestion avoidance.

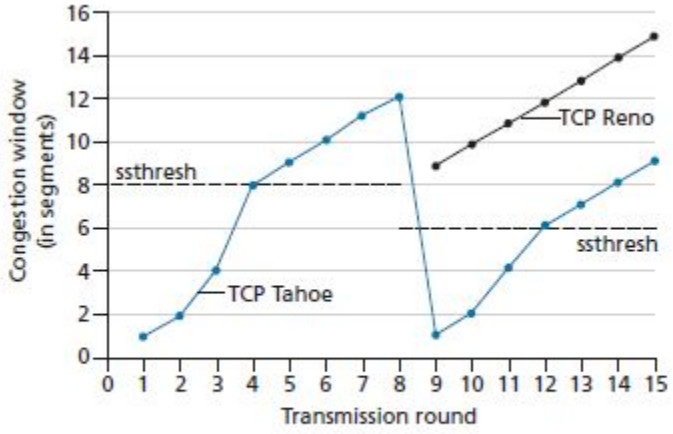


Figure 3.52 ♦ Evolution of TCP's congestion window (Tahoe and Reno)