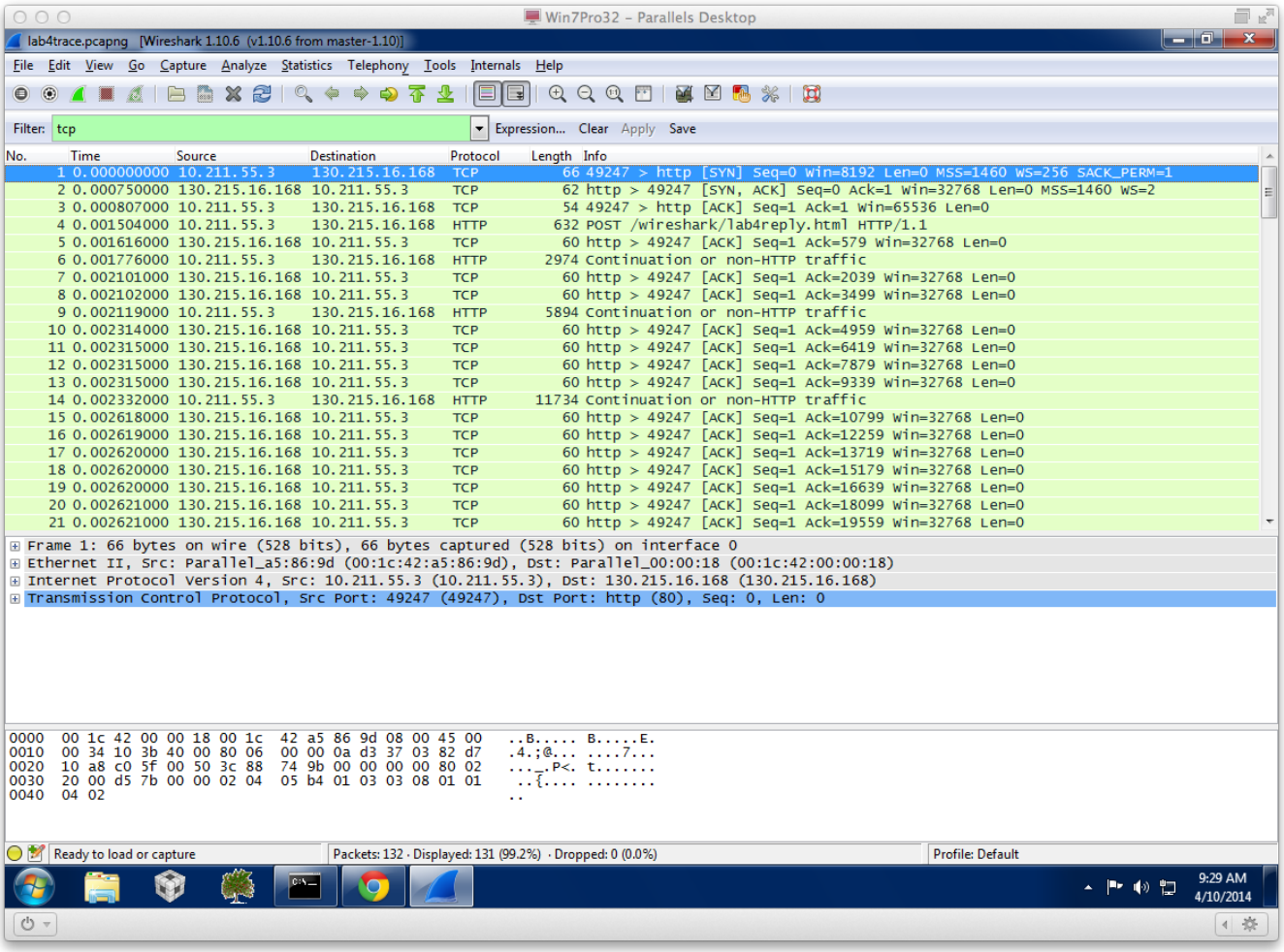
1. What is the IP address and TCP port number used by your client computer (source) to transfer the file to `spinlab.wpi.edu`?

   My computer is at 10.211.55.3. The source port is 49247. See screenshot below.

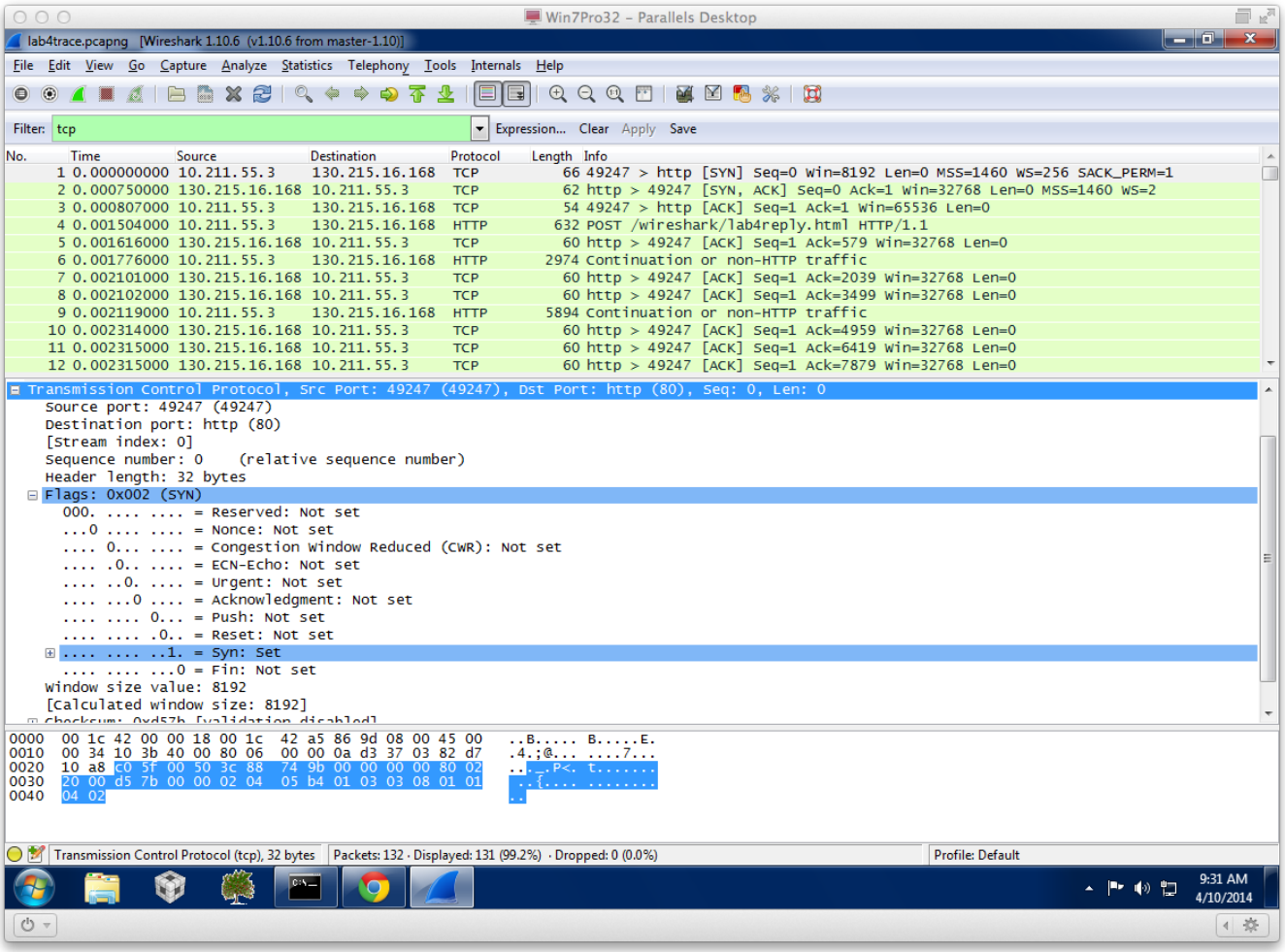


2. What is the IP address of `spinlab.wpi.edu`? On what port number is it sending and receiving TCP segments for this connection?

   Spinlab is at 130.215.16.168 and is using port 80 (destination port). See previous screenshot.

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

   The (relative) sequence number is zero (Seq=0) and the flags are set such that SYN is set (see screenshot below).



4. What is the sequence number of the SYNACK segment sent by `spinlab.wpi.edu` to the client computer in reply to the SYN?

   Also Seq=0 (relative sequence number, see screenshot below).

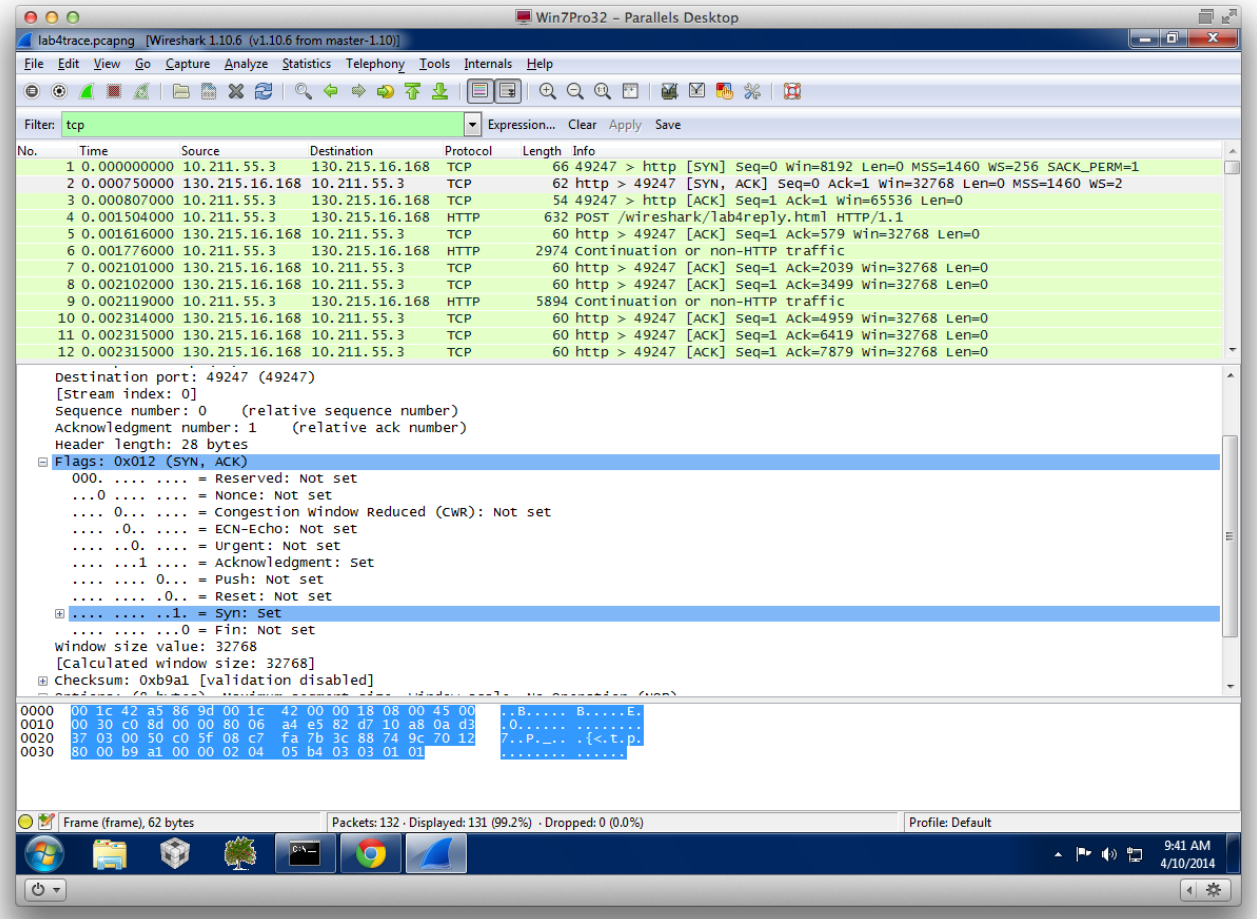
   What is the acknowledgement number (Ack=?) in the SYNACK segment?

   Ack = 1 (relative acknowledgement number, see screenshot below)

   How did `spinlab.wpi.edu` determine that value?

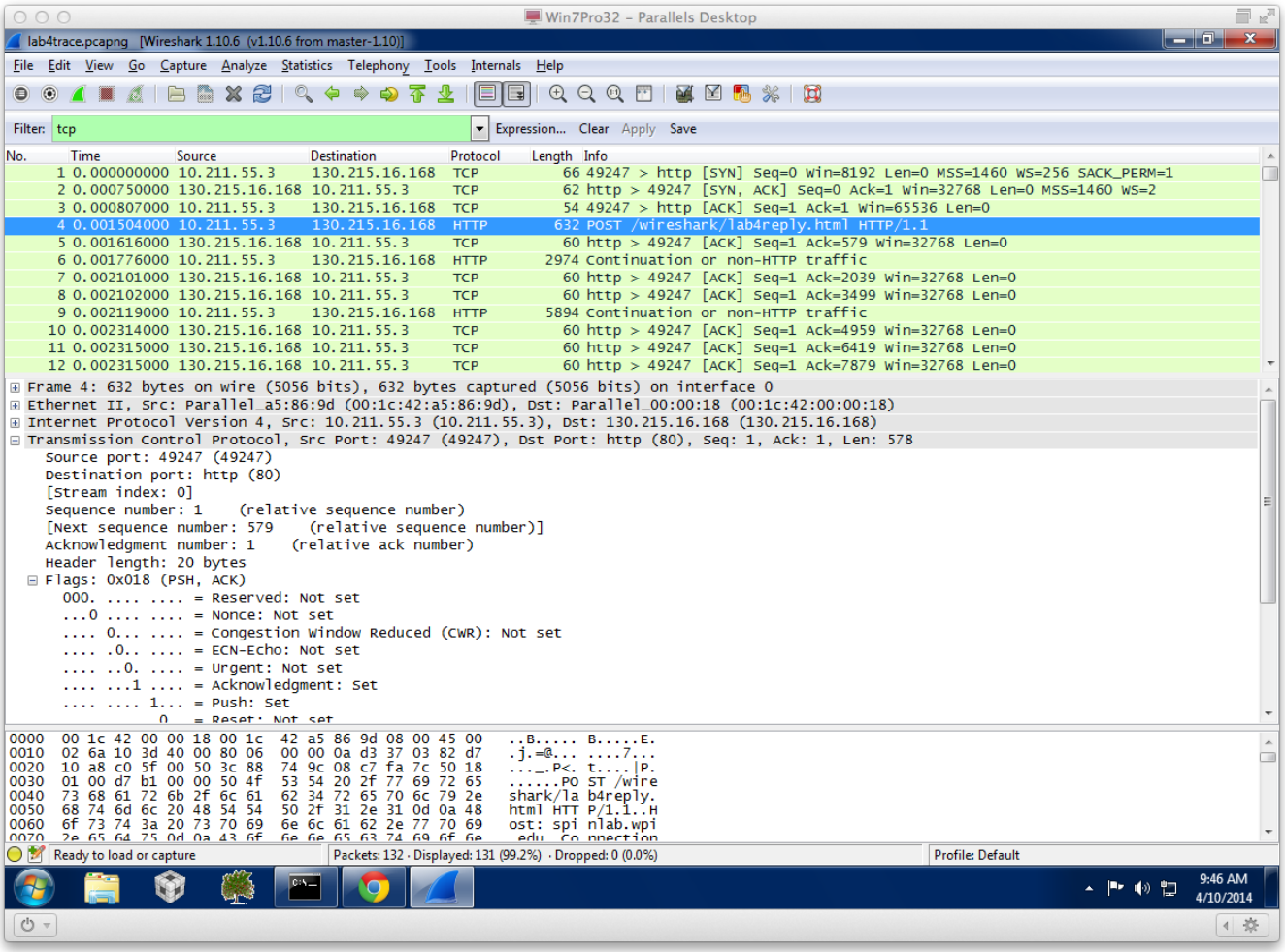
   As discussed in the TCP Mechanisms of the textbook, for a SYN message with sequence number X, the SYNACK message will response with acknowledgement number $X + 1$.

   What is it in the segment that identifies the segment as a SYNACK segment? Both SYN and ACK flags are set (see screens shot below)



5. 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 either 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 or prevent Wireshark from reassembling the packets and displaying them as one response, rather than as multiple continuation packets. This can be disabled by going to Edit→ Preferences → Protocols → HTTP and unchecking the "Reassemble HTTP bodies spanning multiple TCP segments" box.

   Sequence number is 1 (see screenshot below)



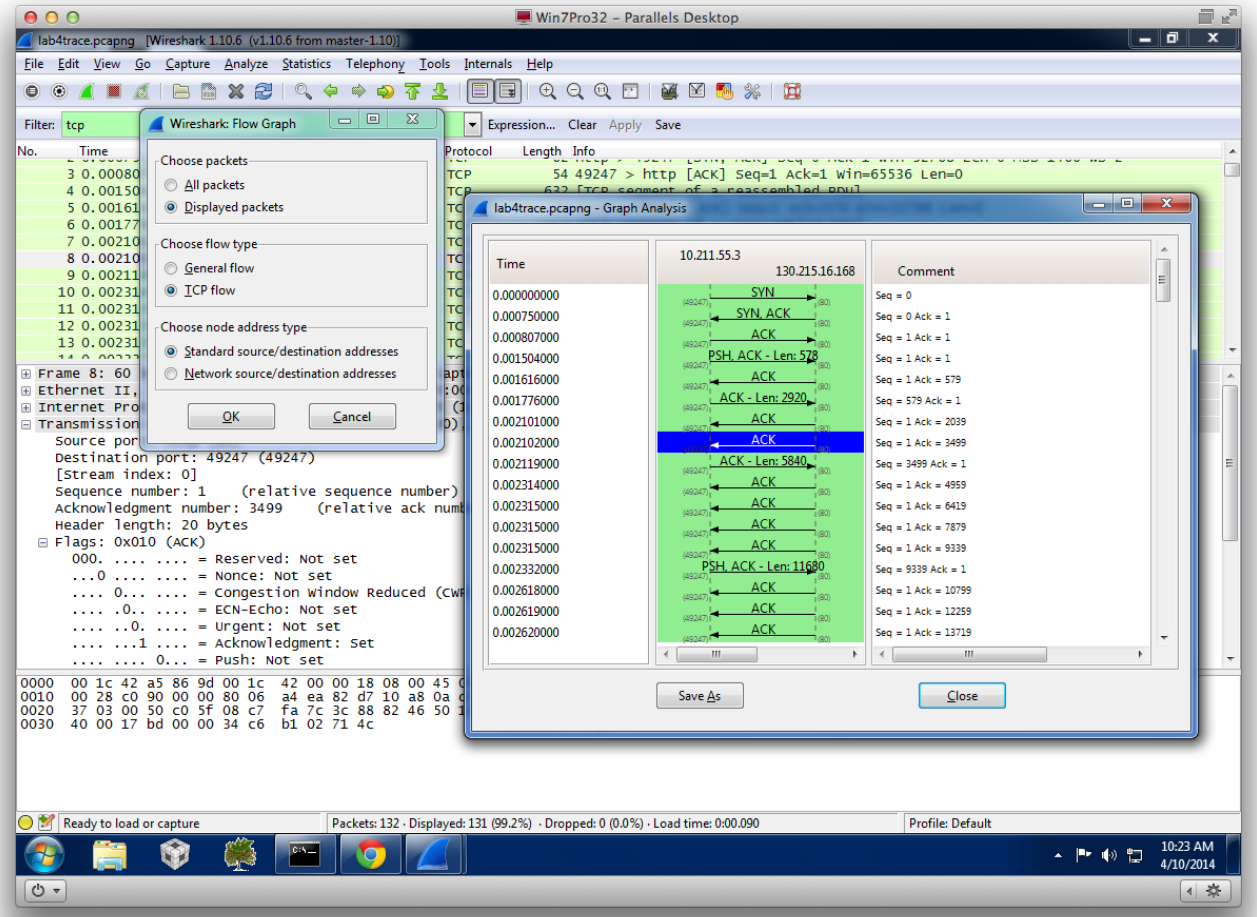
6. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. Calculate the Round Trip Time (RTT). Note that the RTT time is the time difference between the time of the POST message and the corresponding ACK.

   As seen in the previous screenshot, the post occurred at 0.001504 and the ACK occurred at time 0.001616, for a RTT of 0.000112 s or 0.112 ms.

7. 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? Note: you may want to re-enable "Reassemble HTTP bodies spanning multiple TCP segments" if you disabled this setting previously. Also note that there may be multiple ACKs associated with each TCP segment. You should list the time of the final ACK for each segment. The Statistics → Flow Graph → TCP flow view can also be useful.

   - 1) Seq = 1 sent at 0.001504; ACK received at 0.001616
   - 2) Seq = 579 sent at 0.001776; ACK received at 0.002102
   - 3) Seq = 3499 sent at 0.002119; ACK received at 0.002315
   - 4) Seq = 9399 sent at 0.002332; ACK received at 0.002621
   - 5) Seq = 21019 sent at 0.002648; ACK received at 0.002906

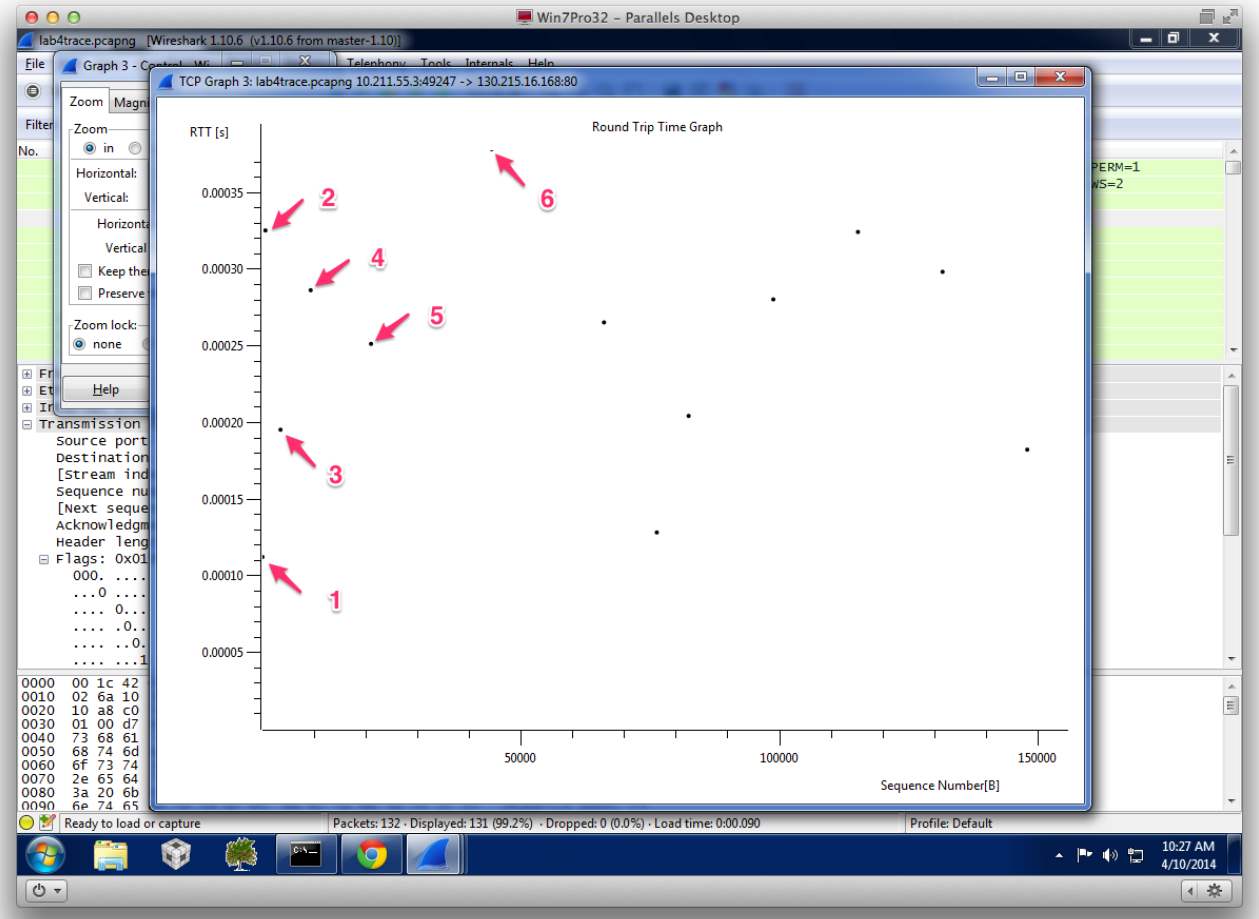
- 6) Seq = 44379 sent at 0.002927; ACK received at 0.003310



8. 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?

   - 1) Seq = 1 RTT = 0.112 ms
   - 2) Seq = 579 RTT = 0.326 ms
   - 3) Seq = 3499 RTT = 0.196 ms
   - 4) Seq = 9399 RTT = 0.289 ms
   - 5) Seq = 21019 RTT = 0.258 ms
   - 6) Seq = 44379 RTT = 0.383 ms

9. Plot the Round Trip Time Graph.Note: 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 `spinlab.wpi.edu` server. Then select: Statistics→TCP Stream Graph→Round Trip Time Graph.



For (a) and (b), fill in this table for 6 segments. For (c), hand in the graph.

Segment Seq. #	Sent time	ACK Receive Time	Actual RTT
Seq = 1	0.001504	0.001616	0.112 ms
Seq = 579	0.001776	0.002102	0.326 ms
Seq = 3499	0.002119	0.002315	0.196 ms
Seq = 9399	0.002332	0.002621	0.289 ms
Seq = 21019	0.002648	0.002906	0.258 ms
Seq = 44379	0.002927	0.003310	0.383 ms

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

    Note: Generally, the TCP segments will all be less that 1460 bytes. This is because most computers have an Ethernet card that limits the length of the maximum IP packet to 1500 bytes (40 bytes of TCP/IP header data and 1460 bytes of TCP payload). This 1500 byte value is the standard maximum length allowed by Ethernet. If your trace indicates a TCP length greater than 1500 bytes, and your computer is using an Ethernet connection, then Wireshark is reporting the wrong TCP segment length; it will likely also show only one large TCP segment rather than multiple smaller segments. Your computer is indeed probably sending multiple smaller segments, as indicated by the ACKs it receives. This inconsistency

in reported segment lengths is due to the interaction between the Ethernet driver and the Wireshark software.

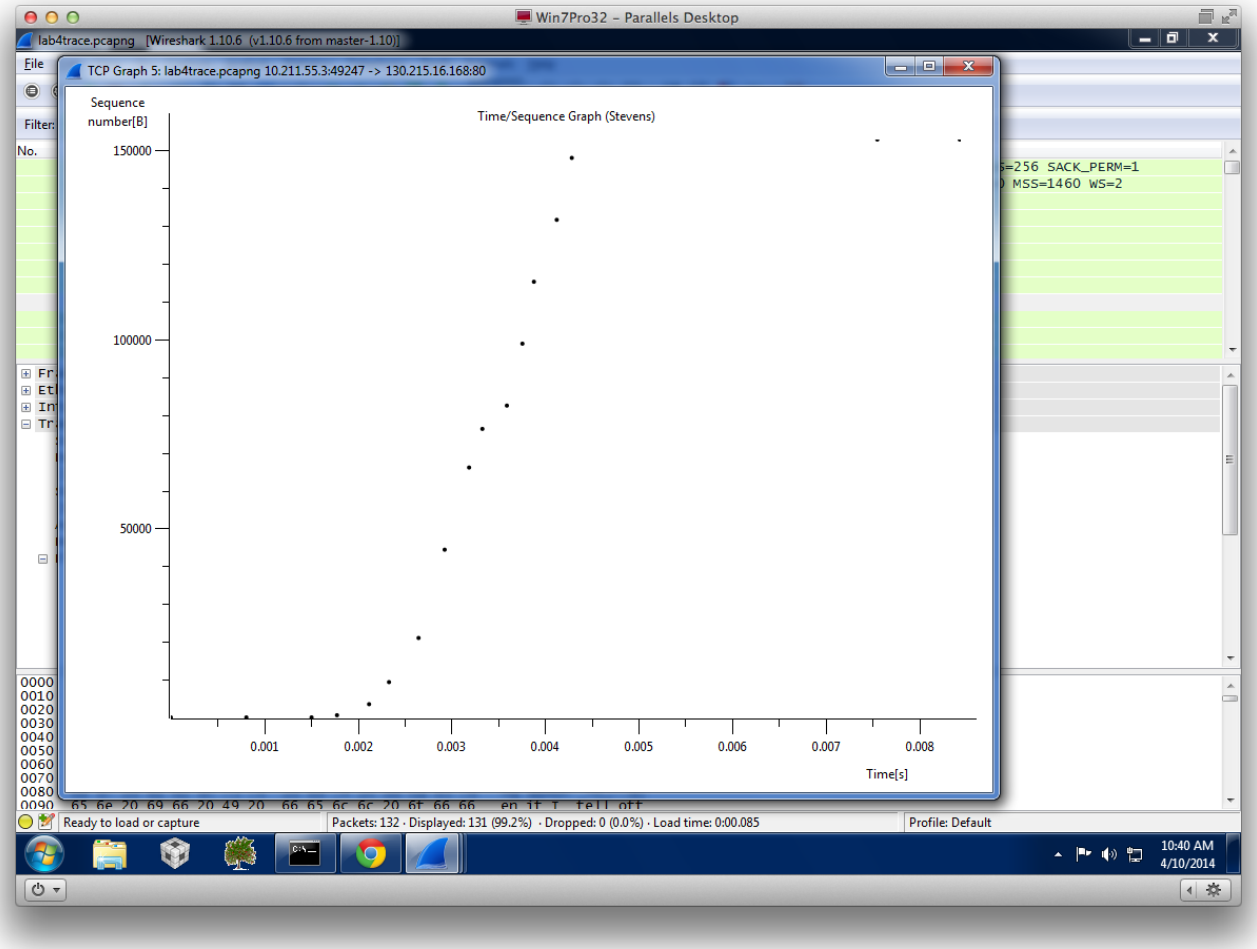
My results show "too long" TCP segments. Even disabling the "Reassemble HTTP bodies spanning multiple TCP segments" doesn't seem to fix this.

- 1) Seq = 1 length = 632
- 2) Seq = 579 length = 2974
- 3) Seq = 3499 length = 5894
- 4) Seq = 9399 length = 11734
- 5) Seq = 21019 length = 23414
- 6) Seq = 44379 length = 21790

11. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question? (hint: plot the time sequence graph from the statistics menu)

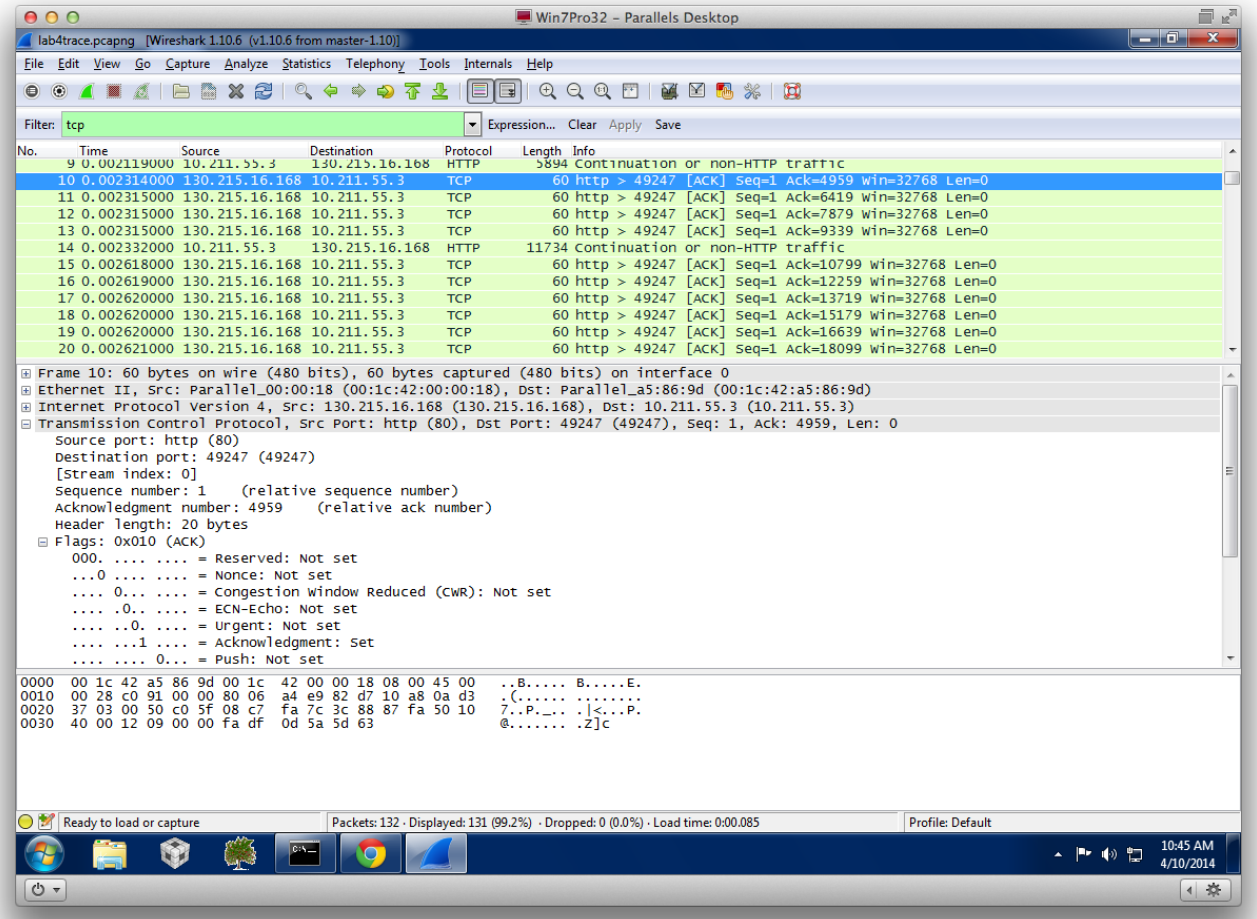
    Note: Select a TCP segment sent from your computer to the server in the Wireshark's "listing of captured-packets" window. Then select the menu: Statistics→TCP Stream Graph→Time-Sequence- Graph (Stevens). You should see a plot that looks similar to the plot in Figure 3. Each dot represents a TCP segment 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.

    No retransmitted segments. See screenshot below. My results look different than in the lab assignment since Wireshark appears to be lumping multiple smaller TCP segments into large segments.



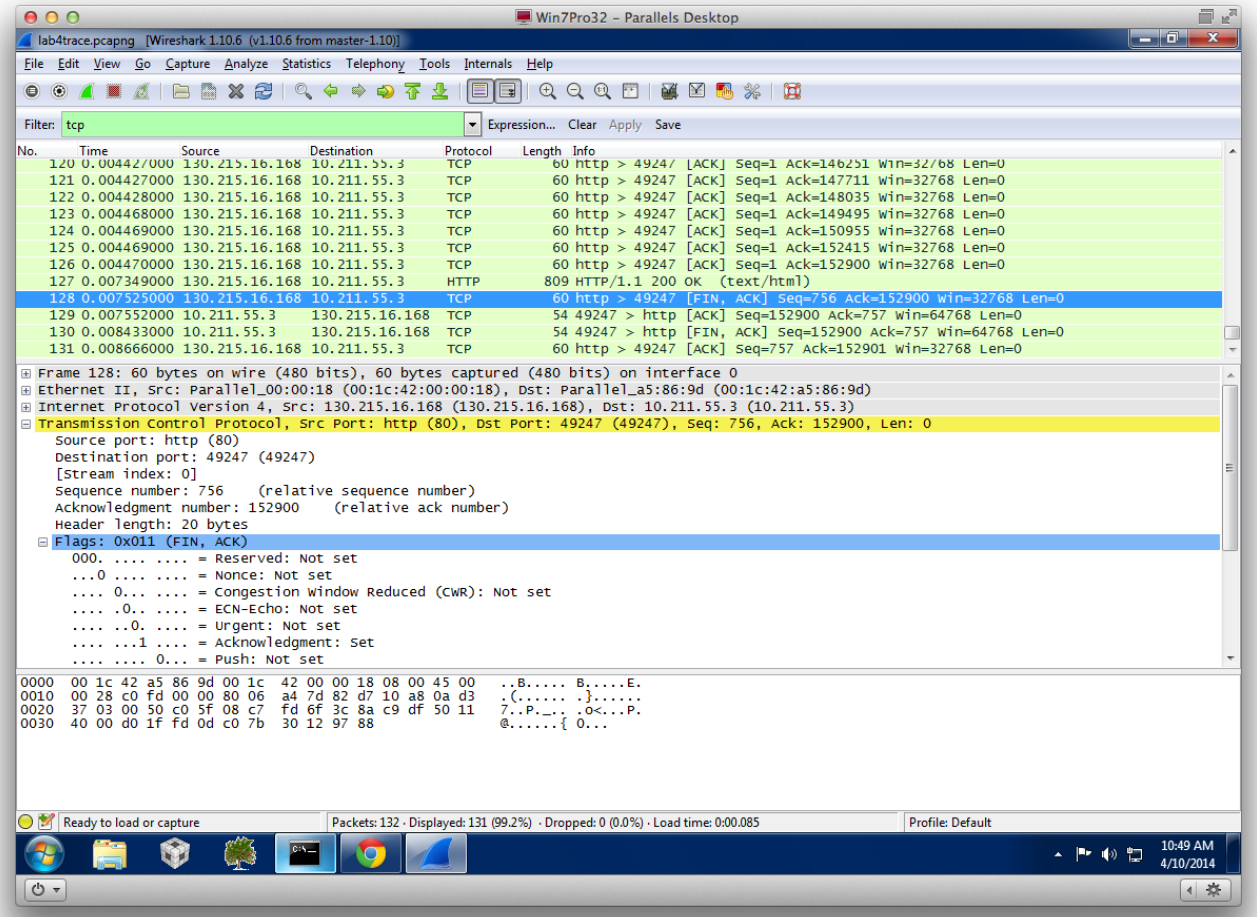
12. How much data does the receiver typically acknowledge in an ACK? Show an example.

    From the screenshot below, we see that the ACK numbers increase in the sequence $10799, 12259, 13719, \ldots$ Note that the ACK numbers increase by 1460 each time, indicating that the receiver is acknowledging 1460 bytes.



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

    I looked to the FINACK packet which shows a acknowledgement number of 152900, meaning that 152900 bytes were acknowledged (this is consistent with the length of the alice.txt file). The time on this message is 0.007525. So an approximate average throughput can be calculated as $\frac{152900 \text{ bytes}}{0.007525 \text{ seconds}} \approx 2.032$ MBps (mega bytes per second) for this connection. See screenshot below.


Win7Pro32 – Parallels Desktop
lab4trace.pcapng [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
120 0.004427000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=146251 Win=32768 Len=0
121 0.004427000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=147711 Win=32768 Len=0
122 0.004428000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=148035 Win=32768 Len=0
123 0.004468000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=149495 Win=32768 Len=0
124 0.004469000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=150955 Win=32768 Len=0
125 0.004469000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=152415 Win=32768 Len=0
126 0.004470000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=1 Ack=152900 Win=32768 Len=0
127 0.007349000 130.215.16.168 10.211.55.3 HTTP 809 HTTP/1.1 200 OK (text/html)
128 0.007525000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [FIN, ACK] Seq=756 Ack=152900 Win=32768 Len=0
129 0.007552000 10.211.55.3 130.215.16.168 TCP 54 49247 > http [ACK] Seq=152900 Ack=757 Win=64768 Len=0
130 0.008433000 10.211.55.3 130.215.16.168 TCP 54 49247 > http [FIN, ACK] Seq=152900 Ack=757 Win=64768 Len=0
131 0.008666000 130.215.16.168 10.211.55.3 TCP 60 http > 49247 [ACK] Seq=757 Ack=152901 Win=32768 Len=0
Frame 128: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: Parallel_00:00:18 (00:1c:42:00:00:18), Dst: Parallel_a5:86:9d (00:1c:42:a5:86:9d)
Internet Protocol Version 4, Src: 130.215.16.168 (130.215.16.168), Dst: 10.211.55.3 (10.211.55.3)
Transmission Control Protocol, Src Port: http (80), Dst Port: 49247 (49247), Seq: 756, Ack: 152900, Len: 0
Source port: http (80)
Destination port: 49247 (49247)
[Stream index: 0]
Sequence number: 756 (relative sequence number)
Acknowledgment number: 152900 (relative ack number)
Header length: 20 bytes
Flags: 0x011 (FIN, 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
0000 00 1c 42 a5 86 9d 00 1c 42 00 00 18 08 00 45 00 ..B..... B.....E.
0010 00 28 c0 fd 00 00 80 06 a4 7d 82 d7 10 a8 0a d3 .(..... .}......
0020 37 03 00 50 c0 5f 08 c7 fd 6f 3c 8a c9 df 50 11 7..P._.. .o<...P.
0030 40 00 d0 1f fd 0d c0 7b 30 12 97 88 @......{ 0...
Ready to load or capture
Packets: 132 · Displayed: 131 (99.2%) · Dropped: 0 (0.0%) · Load time: 0:00.085
Profile: Default
10:49 AM
4/10/2014