Senanur Güvercinoğlu
150119740

WireShark TCP Lab

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.

Client computer (source) IP address: 192.168.1.102
TCP port number:1161

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?

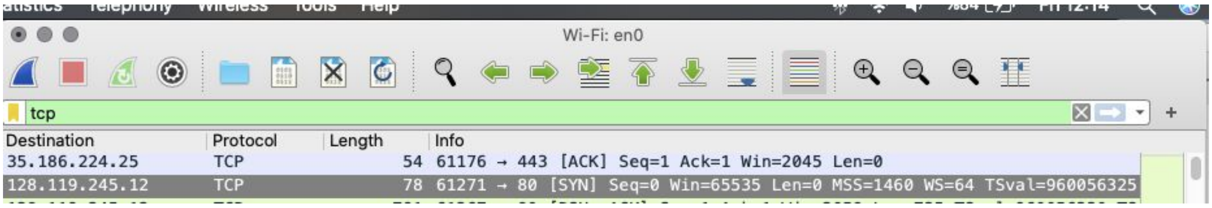
Destination computer: gaia.cs.umass.edu IP address:128.119.245.12
TCP port number:80

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?

Client computer (source) IP address:192.168.0.20
TCP port number:61271

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?

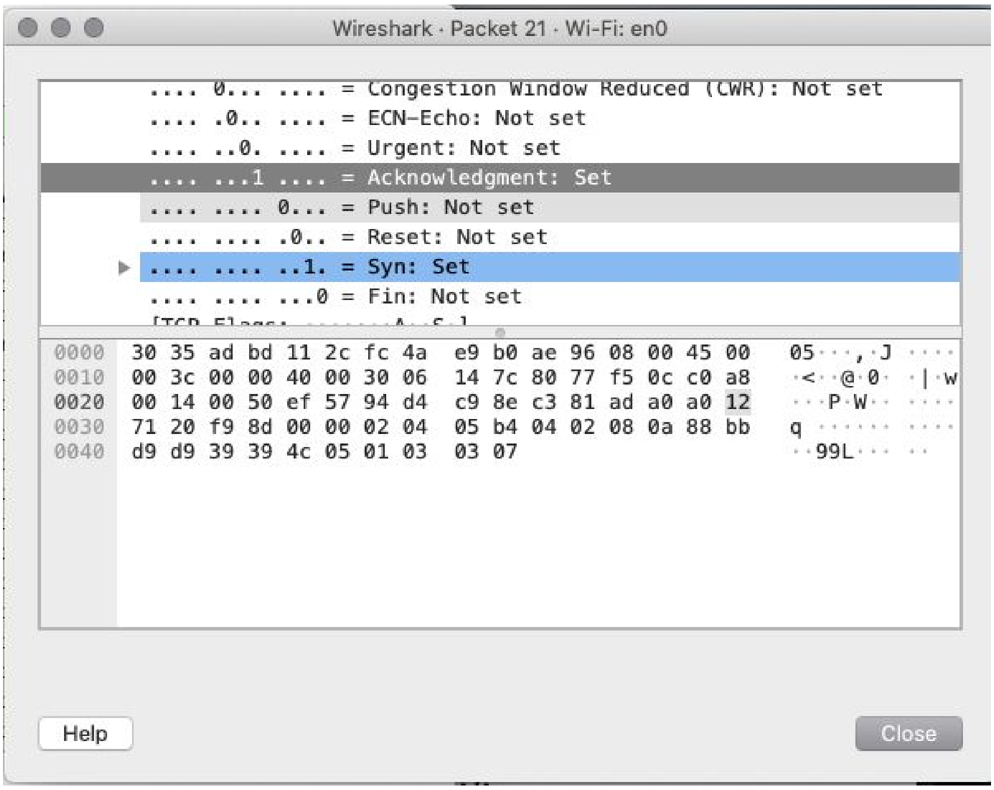
Sequence number of the TCP SYN segment is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu. The value is 0 in this trace.



```
tcp
Destination        Protocol   Length   Info
35.186.224.25      TCP            54   61176 → 443 [ACK] Seq=1 Ack=1 Win=2045 Len=0
128.119.245.12     TCP            78   61271 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=960056325
```

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?

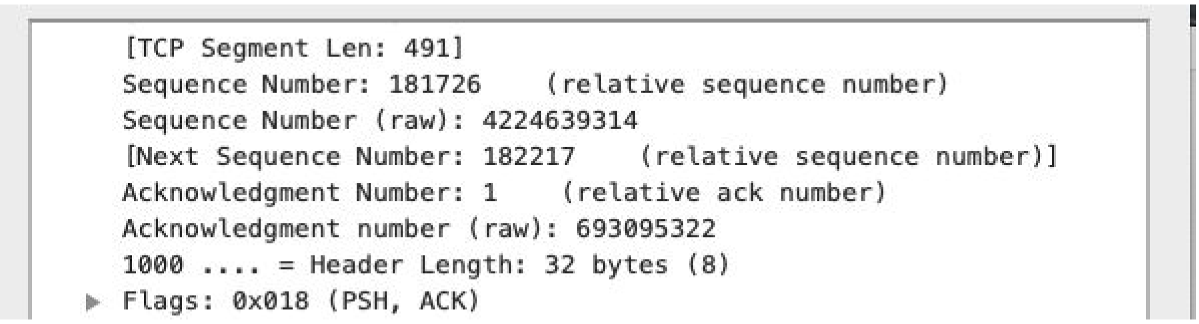
Sequence number of the SYNACK segment from gaia.cs.umass.edu to the client computer in reply to the SYN has the value of 0 in this trace. The value of the ACKnowledgement field is 1.The value is determined by initial sequence number +1.The message carries flags that show it to be a SYNACK message.



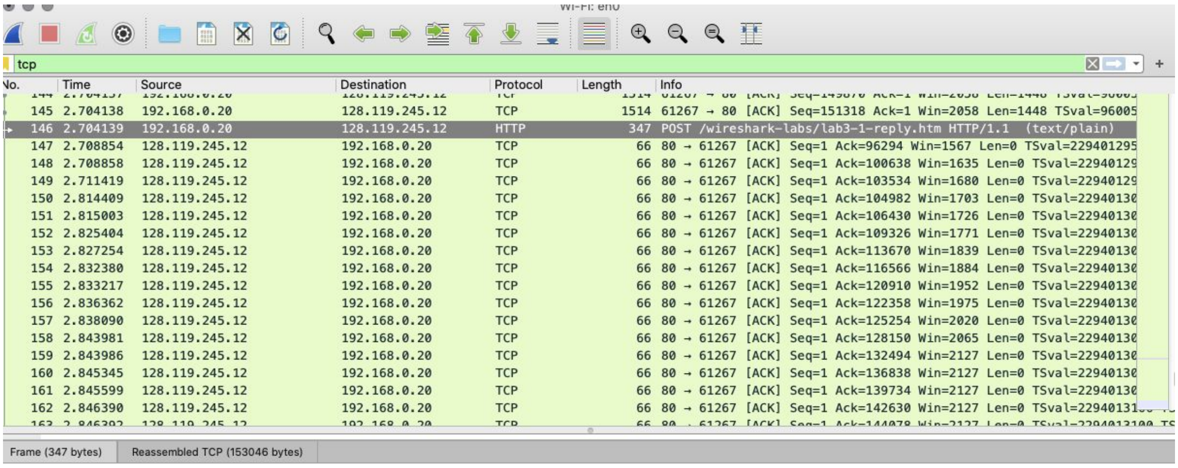
```
.... 0... .... = Congestion Window Reduced (CWR): Not set
.... .0.. .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
.... ...1 .... = Acknowledgment: Set
.... .... 0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..1. = Syn: Set
.... .... ...0 = Fin: Not set
[TCP Flags: ·······A··S·]

0000  30 35 ad bd 11 2c fc 4a  e9 b0 ae 96 08 00 45 00   05···,·J ····
0010  00 3c 00 00 40 00 30 06  14 7c 80 77 f5 0c c0 a8   ·<··@·0· ·|·w
0020  00 14 00 50 ef 57 94 d4  c9 8e c3 81 ad a0 a0 12   ···P·W·· ····
0030  71 20 f9 8d 00 00 02 04  05 b4 04 02 08 0a 88 bb   q······· ····
0040  d9 d9 39 39 4c 05 01 03  03 07                     ··99L··· ··
```

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.

No. 146 segment is the TCP segment containing the HTTP POST command. The sequence number of this segment has the value of 181726.



```
[TCP Segment Len: 491]
Sequence Number: 181726    (relative sequence number)
Sequence Number (raw): 4224639314
[Next Sequence Number: 182217    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 693095322
1000 .... = Header Length: 32 bytes (8)
▶ Flags: 0x018 (PSH, ACK)
```

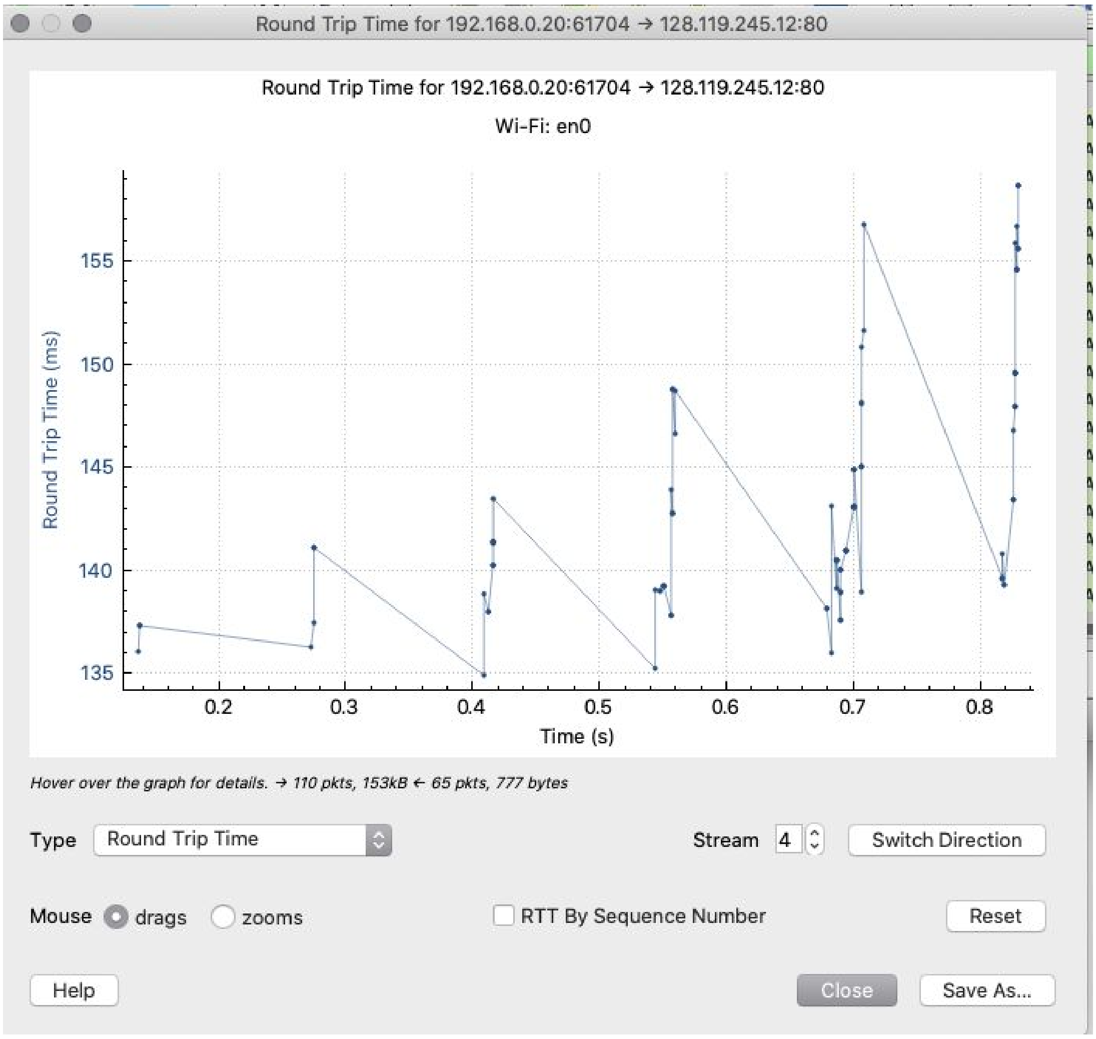


No.	Time	Source	Destination	Protocol	Length	Info
144	2.704137	192.168.0.20	128.119.245.12	TCP	1514	61267 → 80 [ACK] Seq=149870 Ack=1 Win=2058 Len=1448 TSval=96005
145	2.704138	192.168.0.20	128.119.245.12	TCP	1514	61267 → 80 [ACK] Seq=151318 Ack=1 Win=2058 Len=1448 TSval=96005
146	2.704139	192.168.0.20	128.119.245.12	HTTP	347	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
147	2.708854	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=96294 Win=1567 Len=0 TSval=229401295
148	2.708858	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=100638 Win=1635 Len=0 TSval=22940129
149	2.711419	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=103534 Win=1680 Len=0 TSval=22940129
150	2.814409	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=104982 Win=1703 Len=0 TSval=22940130
151	2.815003	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=106430 Win=1726 Len=0 TSval=22940130
152	2.825404	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=109326 Win=1771 Len=0 TSval=22940130
153	2.827254	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=113670 Win=1839 Len=0 TSval=22940130
154	2.832380	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=116566 Win=1884 Len=0 TSval=22940130
155	2.833217	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=120910 Win=1952 Len=0 TSval=22940130
156	2.836362	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=122358 Win=1975 Len=0 TSval=22940130
157	2.838090	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=125254 Win=2020 Len=0 TSval=22940130
158	2.843981	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=128150 Win=2065 Len=0 TSval=22940130
159	2.843986	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=132494 Win=2127 Len=0 TSval=22940130
160	2.845345	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=136838 Win=2127 Len=0 TSval=22940130
161	2.845599	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=139734 Win=2127 Len=0 TSval=22940130
162	2.846390	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=142630 Win=2127 Len=0 TSval=22940131
163	2.846392	128.119.245.12	192.168.0.20	TCP	66	80 → 61267 [ACK] Seq=1 Ack=144078 Win=2127 Len=0 TSval=229401310 TS

Frame (347 bytes) | Reassembled TCP (153046 bytes)

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 242 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 242 for all subsequent segments.

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 gaia.cs.umass.edu server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph.*



Hover over the graph for details. → 110 pkts, 153kB ← 65 pkts, 777 bytes

Type Round Trip Time

Stream 4

Mouse ● drags ○ zooms

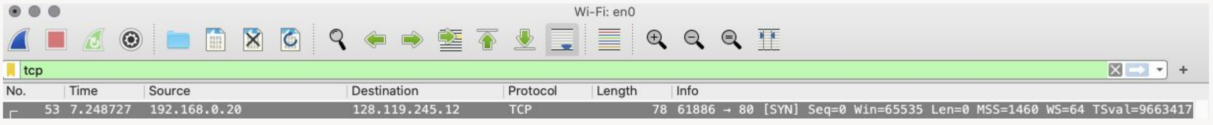
□ RTT By Sequence Number

8. What is the length of each of the first six TCP segments?[4]

The length of each of the first TCP segment is 152766.The following segments are all 1.

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 available buffer space is listed 65535.The sende is never throttled because we never reach full capacity of the window.



tcp

No.	Time	Source	Destination	Protocol	Length	Info
53	7.248727	192.168.0.20	128.119.245.12	TCP	78	61886 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=9663417

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

No,no segments were ever retransmitted.This is shown by the fact that an old ACK number was never re sent in order to re request former packets.

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 250 in the text).

The receiver is typically acking 432 bits. There are cases where the receiver acks every other segment.This is shown when more than one ack occurs in a row.

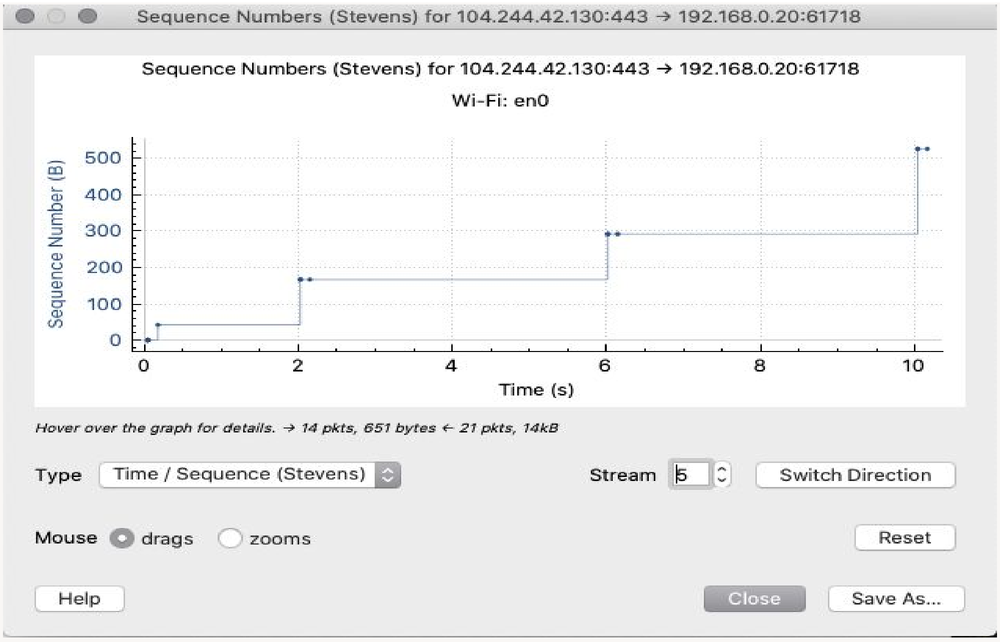
12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

The throughput can be calculated by using the value of the last ack(149,629)-the first sequence number(1) divided by the time since first frame(1.6)=93517.6 bps.

4. TCP congestion control in action

13. Use the *Time-Sequence-Graph(Stevens)* plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart 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.

The TCP slow start phase begins at sequence number 0, and ends just before sequence number 40.Congestion avoidance takes over at 40.



Sequence Numbers (Stevens) for 104.244.42.130:443 → 192.168.0.20:61718

Hover over the graph for details. → 14 pkts, 651 bytes ← 21 pkts, 14kB

Type Time / Sequence (Stevens)

Stream 5

Mouse ● drags ○ zooms