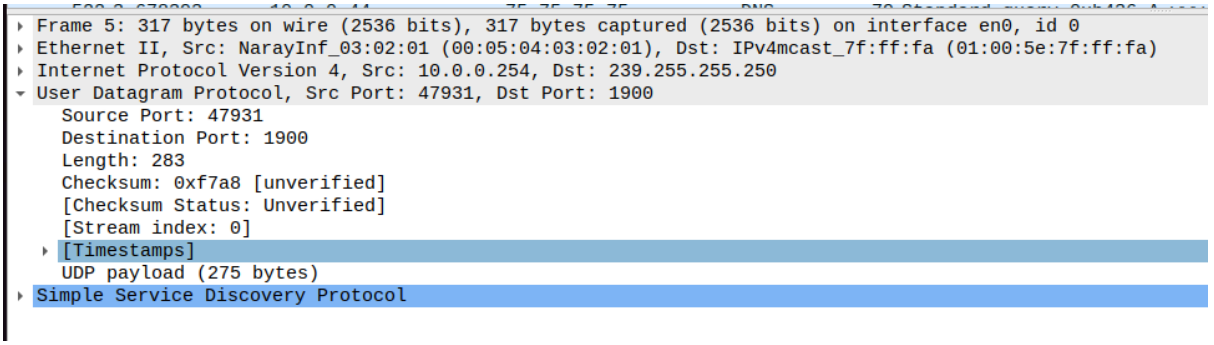
UDP:

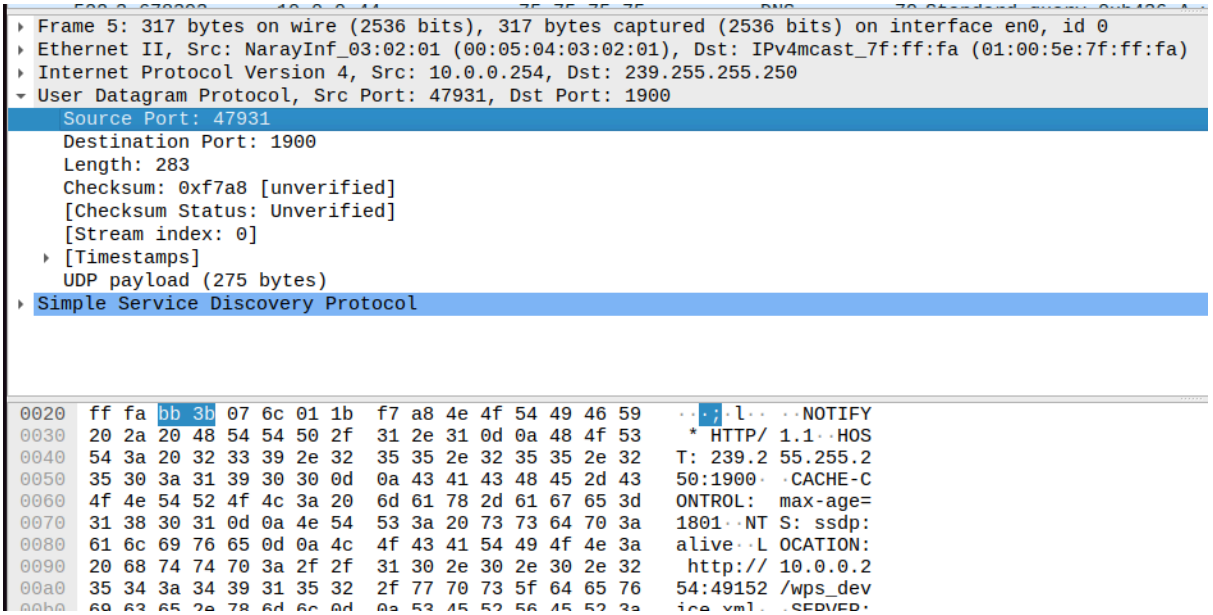
1. Select the first UDP segment in your trace. What is the packet number of this segment in the trace file? What type of application-layer payload or protocol message is being carried in this UDP segment? Look at the details of this packet in Wireshark. How many fields there are in the UDP header? (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) What are the names of these fields?

   Ans:
   Packet number : 5
   Application-layer protocol: Simple Service Discovery Protocol (SSDP)
   Fields: 4
   1. Source Port
   2. Destination Port
   3. Length
   4. Checksum



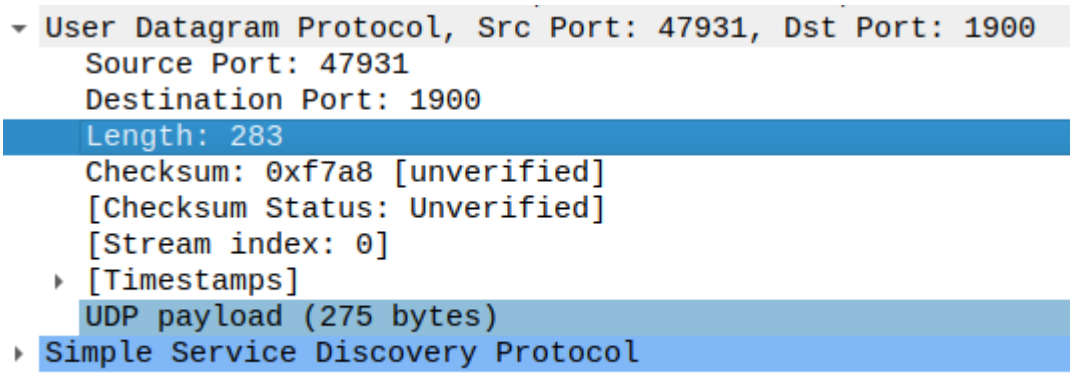
2. By consulting the displayed information in Wireshark's packet content field for this packet (or by consulting the textbook), what is the length (in bytes) of each of the UDP header fields?

   Ans: Source Port : 2 byte (bb 3b)
   Destination Port : 2 byte (07 6c)
   Length : 2 byte (01 1b)
   Checksum : 2 byte (f7 a8)



3. The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet.

Ans: UDP Length field is the total length in bytes of the UDP headed (8 bytes) + UDP Payload (275 bytes) = total 283 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)

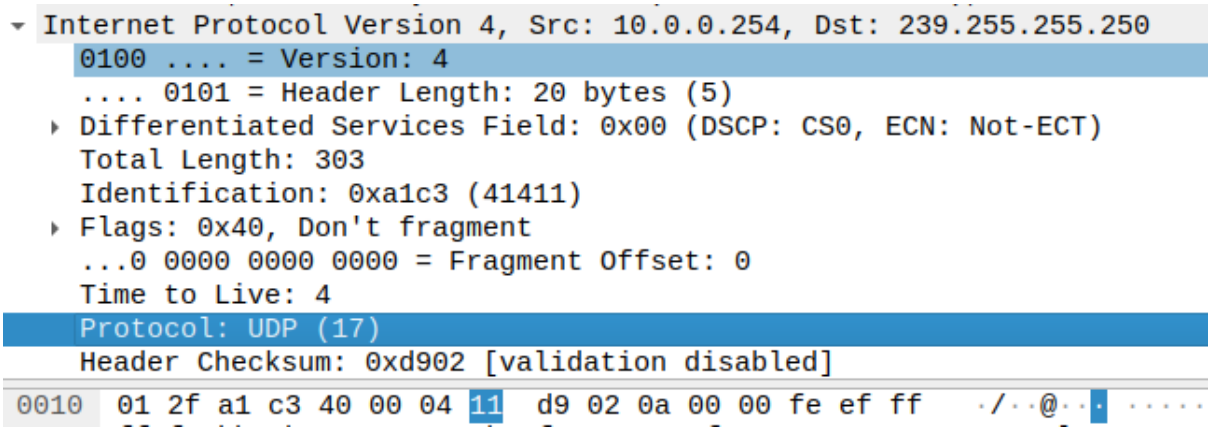
Ans: Maximum number of bytes = 65536 (2^{16}) - 1 - 8 (header bytes) = 65527

5. What is the largest possible source port number? (Hint: see the hint in 4.)

Ans: Ports are 16-bit unsigned values, so their numeric range is 0–65,535 (0 is reserved / uncommon). The maximum numeric value is therefore 65535.

6. What is the protocol number for UDP? Give your answer in decimal notation. To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment (see Figure 4.13 in the text, and the discussion of IP header fields).

Ans: Protocol number : UDP (17) - 11



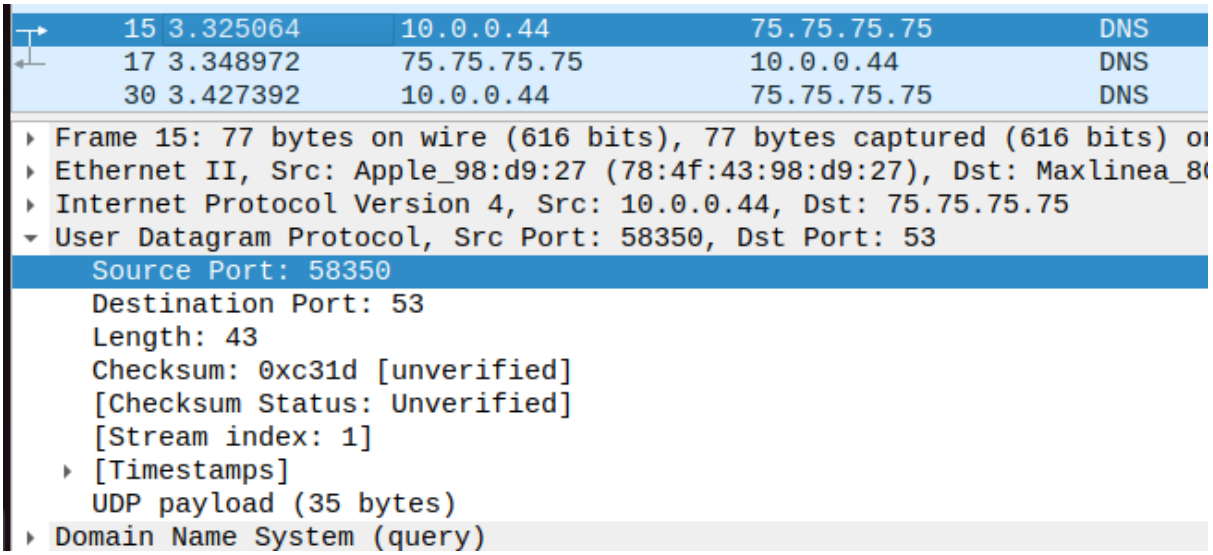
7. Examine the pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). What is the packet number of the first of these two UDP segments in the trace file? What is the value in the source port field in this UDP segment? What is the value in the destination port field in this UDP segment? What is the packet number of the second of these two UDP segments in the trace file? What is the value in the source port field in this second UDP segment? What is the value in the destination port field in this

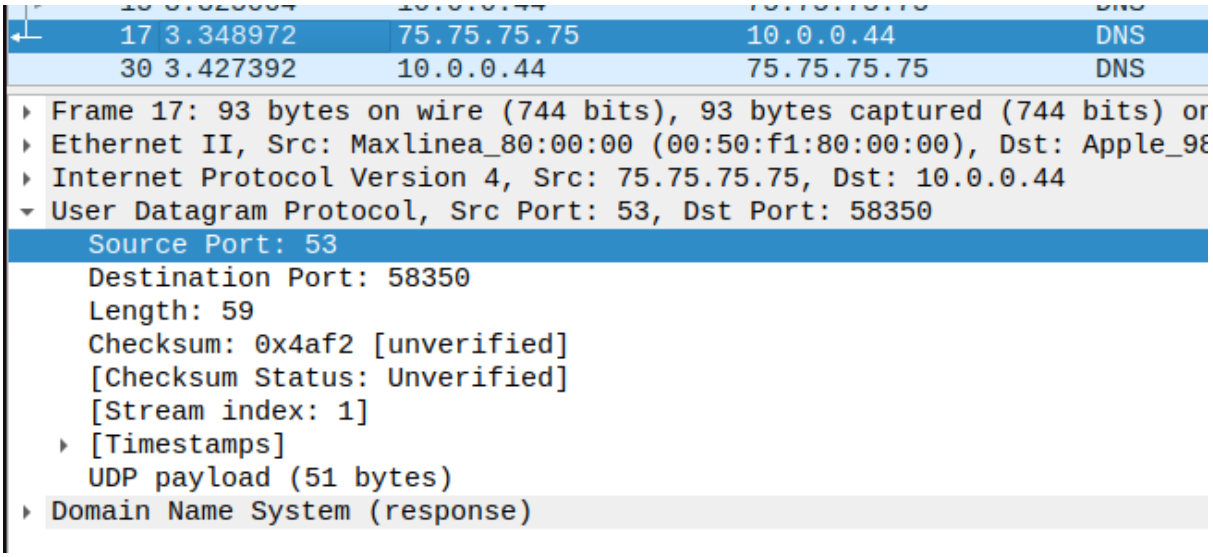
second UDP segment? Describe the relationship between the port numbers in the two packets.

Ans: Packet number of the first UDP segment : 15
Source port field : 58350 (SP1)
Destination port field : 53 (DP1)

Packet number of the second UDP segment : 17
Source port field : 53 (SP2)
Destination port field : 58350 (DP2)

Relationship : SP1 = DP2 , SP2 = DP1

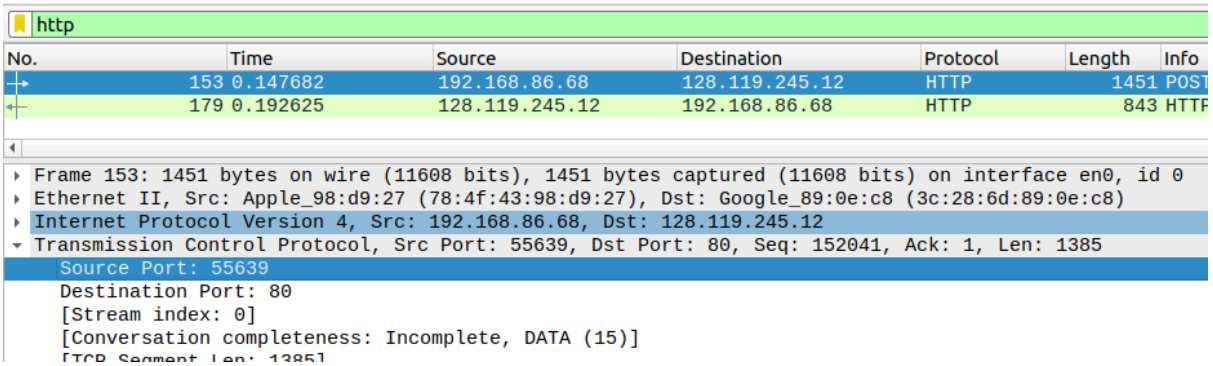




TCP :

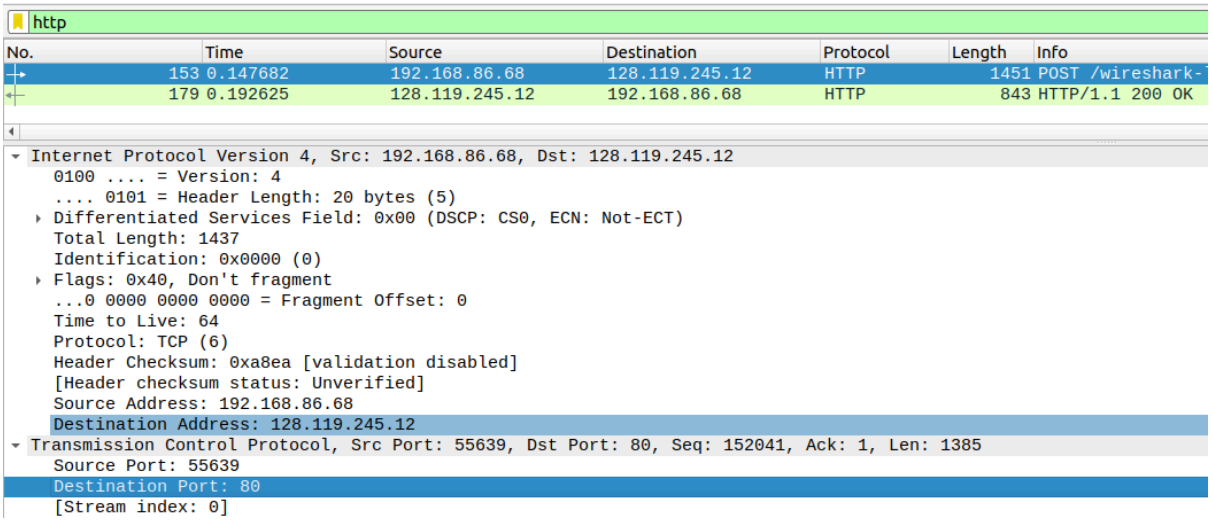
1. What is the IP address and TCP port number used by the client computer (source) that is transferring the alice.txt file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "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 Address : 192.168.86.68
   Port : 55639



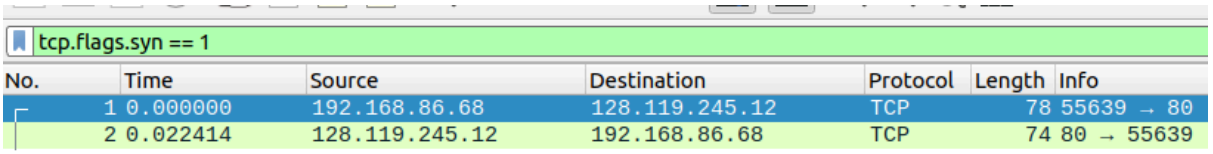
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

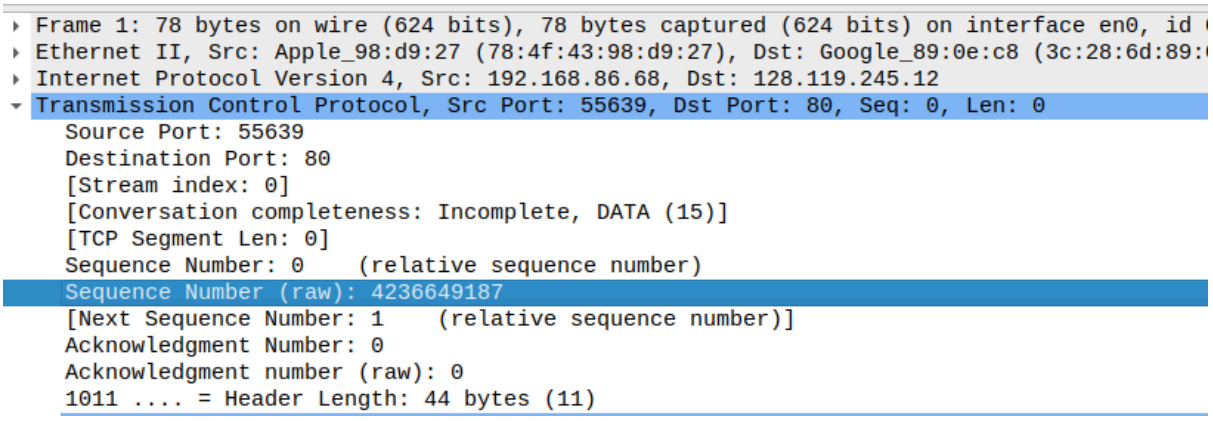
   Ans: IP Address : 128.119.245.12
   Port : 80



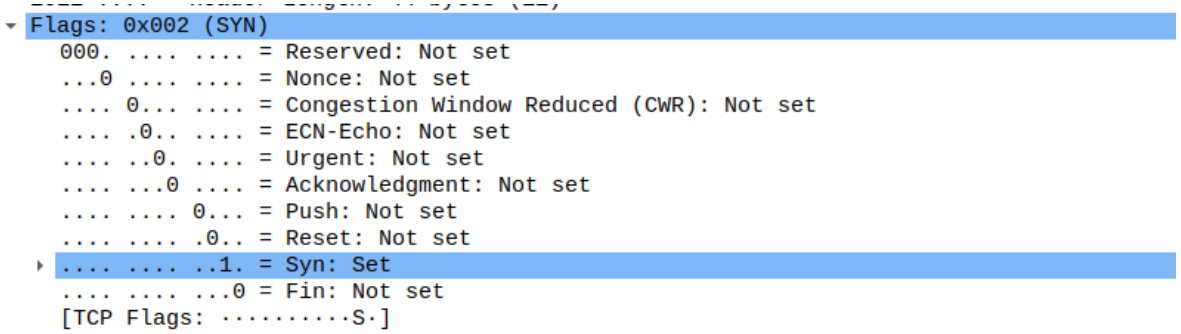
3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? (Note: this is the “raw” sequence number carried in the TCP segment itself; it is NOT the packet # in the “No.” column in the Wireshark window. Remember there is no such thing as a “packet number” in TCP or UDP; as you know, there are sequence numbers in TCP and that’s what we’re after here. Also note that this is not the relative sequence number with respect to the starting sequence number of this TCP session.). What is it in this TCP segment that identifies the segment as a SYN segment? Will the TCP receiver in this session be able to use Selective Acknowledgments (allowing TCP to function a bit more like a “selective repeat” receiver, see section 3.4.5 in the text)?

Ans: Sequence number (raw) : 4236649187

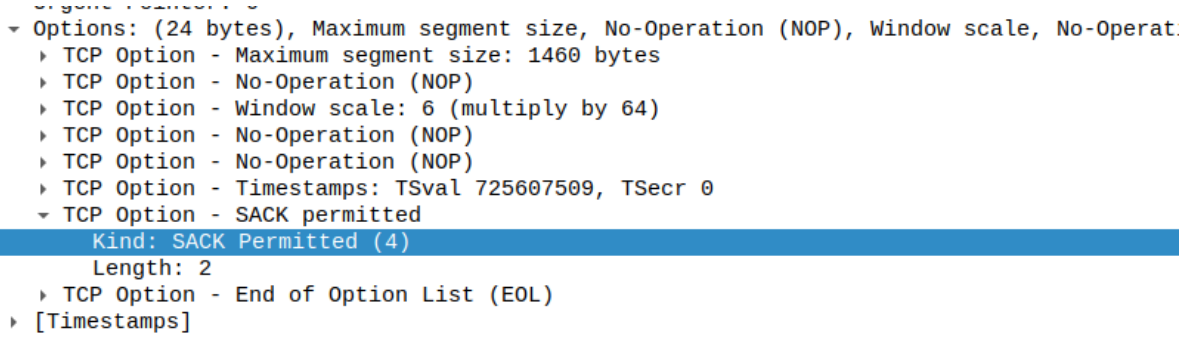




SYN flag : Set to 1 → This is a SYN segment
ACK flag : Set to 0 → Not SYNACK segment



TCP Option - SACK Permitted

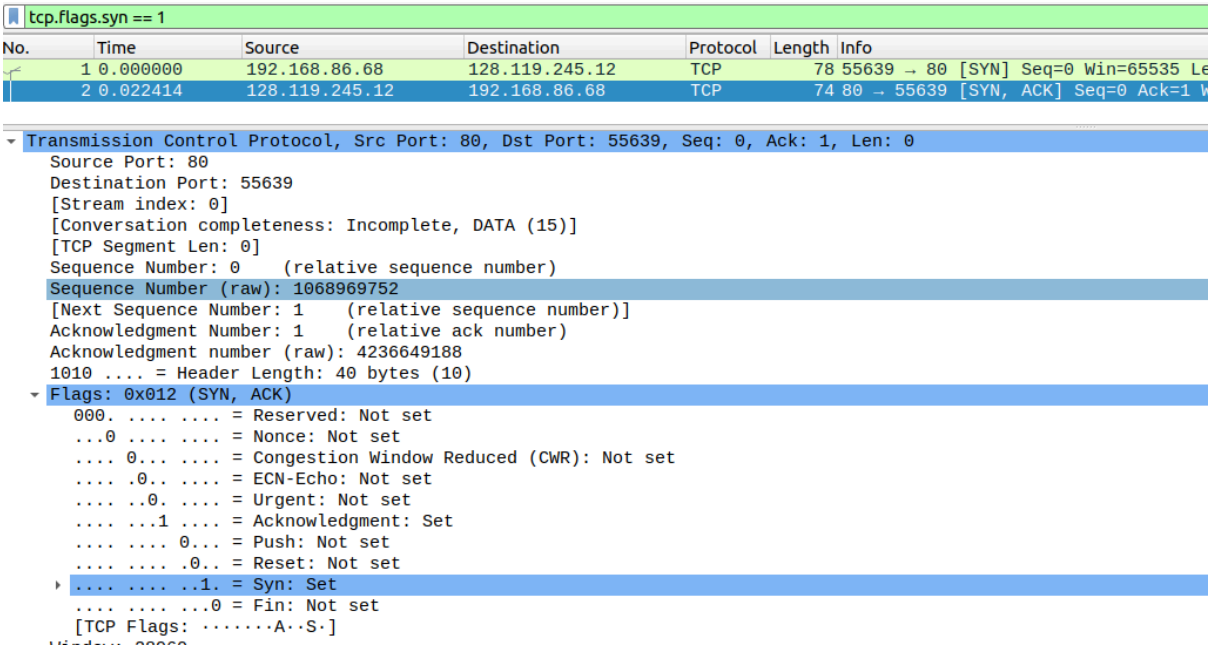


4. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is it in the segment that identifies the segment as a SYNACK segment? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine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

   Ans: Sequence number (relative) : 0
   Sequence number (raw) : 1068969752
   SYN flag = Set to 1 → This is a SYN segment
   ACK flag = Set to 1 → This is a SYNACK segment
   ACK field : 4236649188

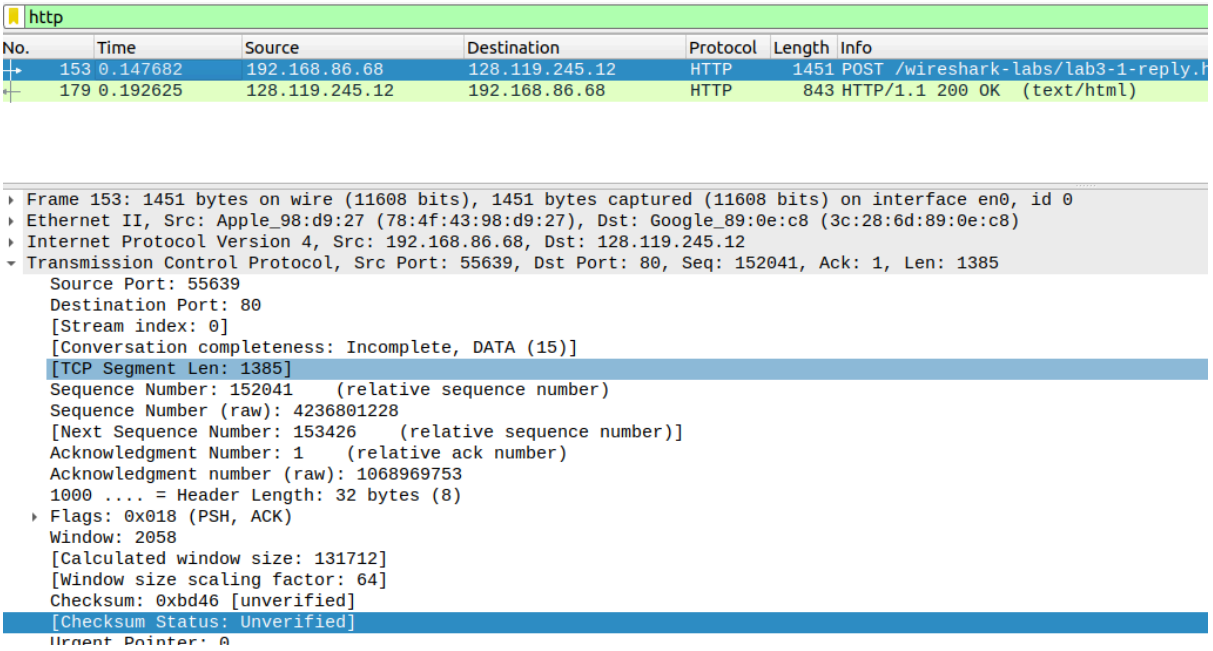
   How it is determined :
   TCP rule:
   To acknowledge a segment, set Ack = (received Seq) + 1

   Received sequence : 4236649187
   ACK Sequence : 4236649187 + 1 = 4236649188

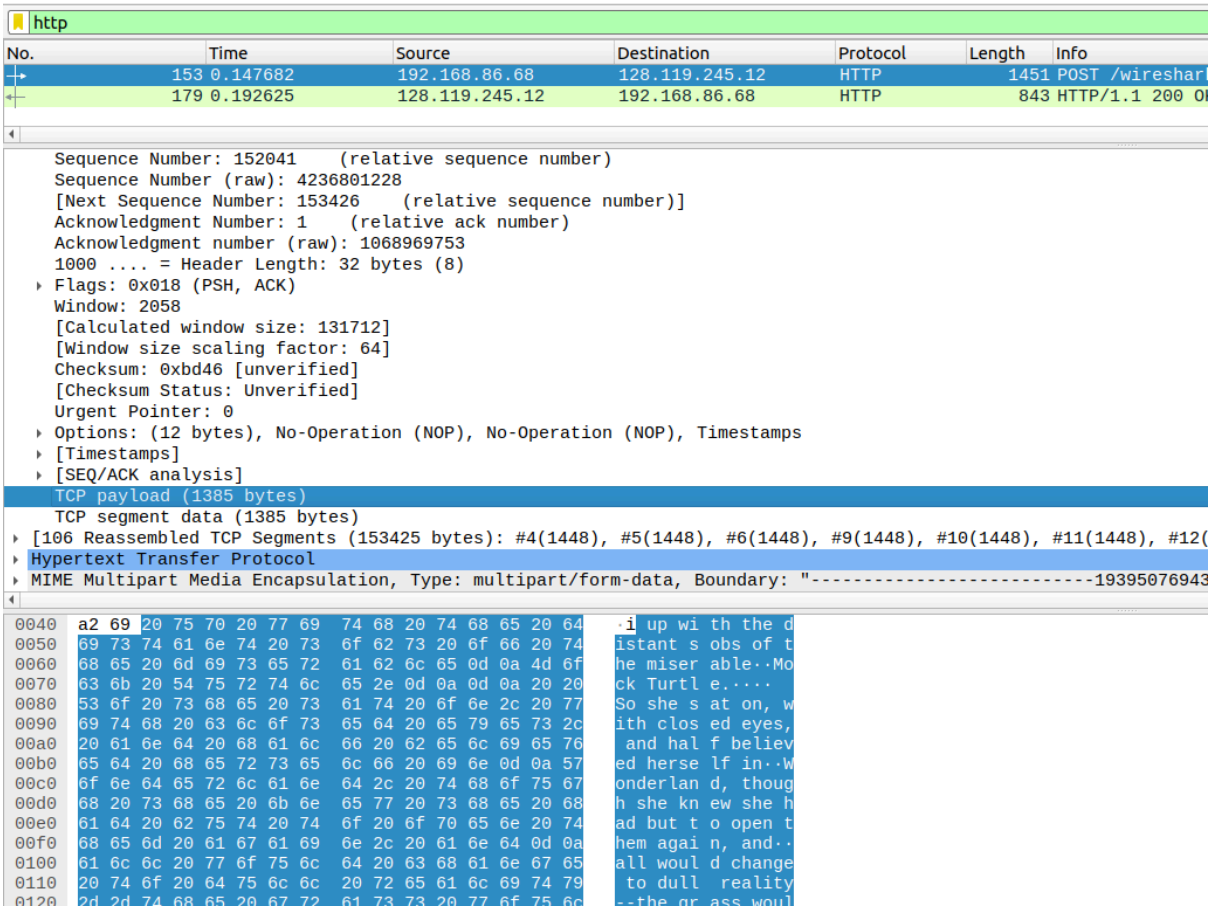


5. What is the sequence number of the TCP segment containing the header of the HTTP POST command? Note that in order to find the POST message header, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with the ASCII text "POST" within its DATA field4,5. How many bytes of data are contained in the payload (data) field of this TCP segment? Did all of the data in the transferred file alice.txt fit into this single segment?

Ans: Sequence number (relative) : 152041
Sequence number (raw) : 4236801228

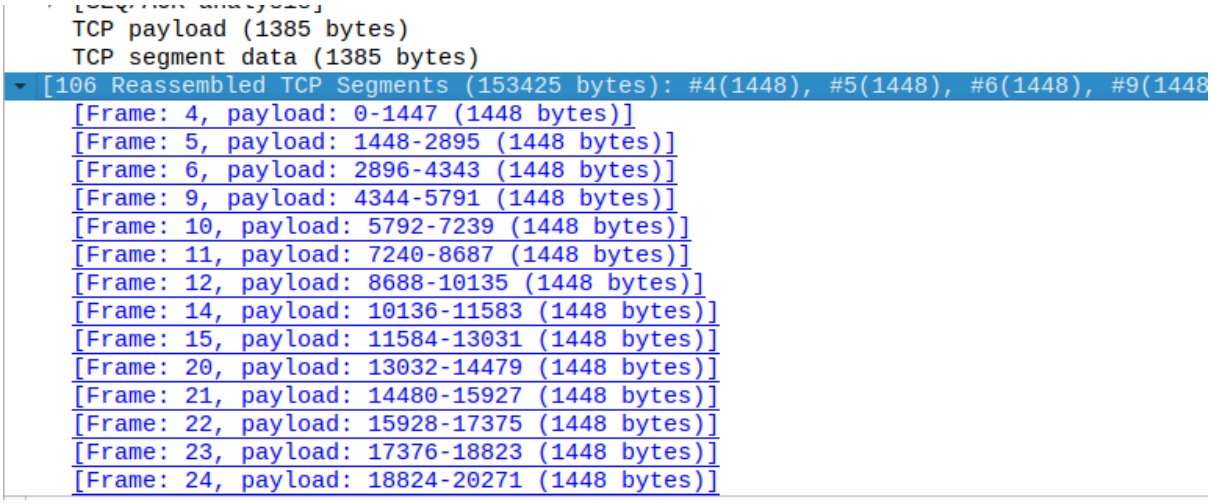


Payload field : 1385



From pdf - size of alice.txt = 153425 bytes

The file was divided into 106 segments (each frame's limit is 1448 bytes)



6. Consider the TCP segment containing the HTTP “POST” as the first segment in the data transfer part of the TCP connection.
   • At what time was the first segment (the one containing the HTTP POST) in the data-transfer part of the TCP connection sent?
   • At what time was the ACK for this first data-containing segment received?
   • What is the RTT for this first data-containing segment?
   • What is the RTT value the second data-carrying TCP segment and its ACK?
   • What is the EstimatedRTT value (see Section 3.5.3, in the text) after the ACK for the second data-carrying segment is received? Assume that in making this calculation after the received of the ACK for the second segment, that the initial value of EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 242, and a value of α = 0.125.

Ans: First segment in the data-transfer part of the TCP connection is sent at 0.024647s. Its sequence number is 1 and length is 1448
That means the ACK for this segment is the one containing ACK number (1+1448) = 1449, as the next segment’s sequence number is 1449. So the ACK for the first segment is sent at 0.052671s.

So, the RTT for this first data-containing segment is:
RTT_1 = 0.052671s - 0.024647s = 0.028624s = 28.624 ms

tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.86.68	128.119.245.12	TCP	78	55639 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=725607509 TSecr=0 SACK_PERM=
2	0.022414	128.119.245.12	192.168.86.68	TCP	74	80 → 55639 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3913851370
3	0.022505	192.168.86.68	128.119.245.12	TCP	66	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=725607531 TSecr=3913851370
4	0.024047	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TCP s
5	0.024048	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1449 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TC
6	0.024049	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=2897 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TC
7	0.052671	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=1449 Win=31872 Len=0 TSval=3913851399 TSecr=725607532
8	0.052676	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=2897 Win=34816 Len=0 TSval=3913851400 TSecr=725607532

Second segment in the data-transfer part of the TCP connection is sent at 0.024048s. Its sequence number is 1449 and length is 1448

That means the ACK for this segment is the one containing ACK number (1449+1448) = 2897, as the next segment's sequence number is 2897. So the ACK for the first segment is sent at 0.052676s.

So, the RTT for this first data-containing segment is:
RTT_2 = 0.052676s - 0.024048s = 0.028628s = 28.628 ms

tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.86.68	128.119.245.12	TCP	78	55639 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=725607509 TSecr=0 SACK_PERM=
2	0.022414	128.119.245.12	192.168.86.68	TCP	74	80 → 55639 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3913851370
3	0.022505	192.168.86.68	128.119.245.12	TCP	66	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=725607531 TSecr=3913851370
4	0.024047	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TCP s
5	0.024048	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1449 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TC
6	0.024049	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=2897 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TC
7	0.052671	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=1449 Win=31872 Len=0 TSval=3913851399 TSecr=725607532
8	0.052676	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=2897 Win=34816 Len=0 TSval=3913851400 TSecr=725607532
9	0.052774	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=4345 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851399 [TC

Estimated RTT value after the ACK received for the second data-carrying segment:

$$\begin{aligned} \text{EstimatedRTT}_2 &= (1 - \alpha) * RTT_1 + \alpha * RTT_2 \\ &= (1 - 0.125) * 28.624 + 0.125 * 28.628 \\ &= 28.6245 \text{ ms} \end{aligned}$$

7. What is the length (header plus payload) of each of the first four data-carrying TCP segments?

Ans: The length of each of the first four data-carrying TCP segments (at #4, #5, #6, #9) is 1514

tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.86.68	128.119.245.12	TCP	78	55639 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=725607509 TSecr=0 SACK_PERM=
2	0.022414	128.119.245.12	192.168.86.68	TCP	74	80 → 55639 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3913851370
3	0.022505	192.168.86.68	128.119.245.12	TCP	66	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=725607531 TSecr=3913851370
4	0.024047	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TCP s
5	0.024048	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1449 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TC
6	0.024049	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=2897 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TC
7	0.052671	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=1449 Win=31872 Len=0 TSval=3913851399 TSecr=725607532
8	0.052676	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=2897 Win=34816 Len=0 TSval=3913851400 TSecr=725607532
9	0.052774	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=4345 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851399 [TC
10	0.052775	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=5793 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851399 [TC

8. What is the minimum amount of available buffer space advertised to the client by gaia.cs.umass.edu among these first four data-carrying TCP segments? Does the lack of receiver buffer space ever throttle the sender for these first four data- carrying segments?

Ans: The minimum amount of available buffer space advertised among these first four data-carrying TCP segments (at #4, #5, #6, #9) is 131712 bytes, as that is the calculated window size for all of these first four segments.

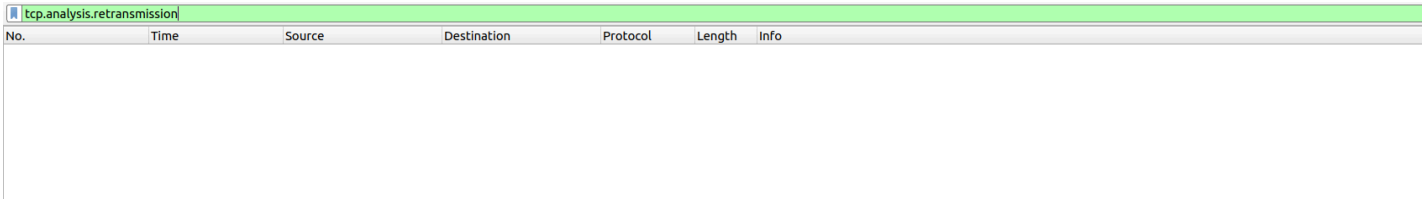
The lack of receiver buffer space does not throttle the sender for the first four data-carrying TCP segments.

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

Ans: There is no retransmitted segments in the trace file.

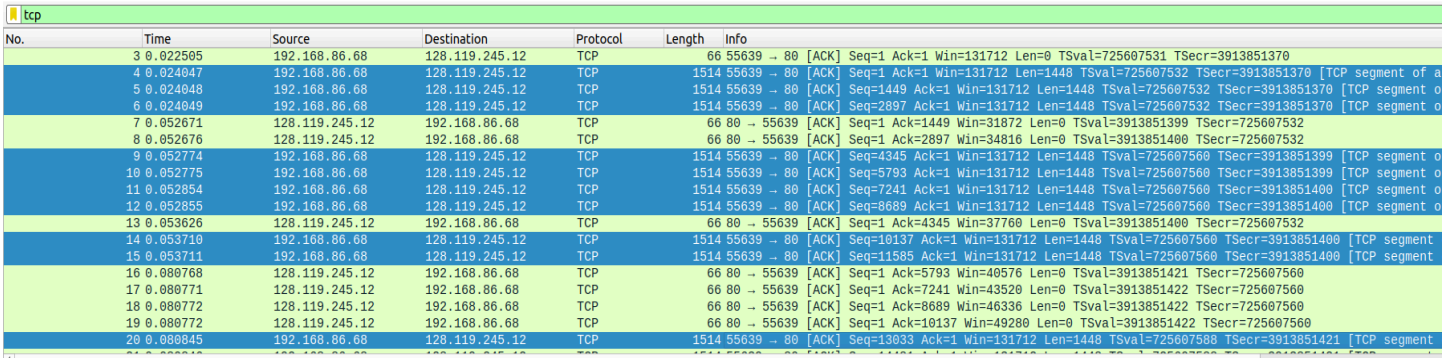
How to check :

1. Use filter “tcp.analysis.retransmission” and see if there is any data segment that is retransmitted or not.
2. Manually check through every TCP data segment inspecting their sequence numbers and duplicate ACKs.



10. How much data does the receiver typically acknowledge in an ACK among the first ten data-carrying segments sent from the client to gaia.cs.umass.edu? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 in the text) among these first ten data-carrying segments?

Ans: The receiver typically acknowledges 1448 bytes of data in an ACK among the first ten data-carrying segments sent from the client. As the length of each segments is 1448 bytes.



tcp

No.	Time	Source	Destination	Protocol	Length	Info
3	0.022505	192.168.86.68	128.119.245.12	TCP	66	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=725607531 TSecr=3913851370
4	0.024047	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TCP segment of a
5	0.024048	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1449 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TCP segment o
6	0.024049	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=2897 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=3913851370 [TCP segment o
7	0.052671	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=1449 Win=31872 Len=0 TSval=3913851399 TSecr=725607532
8	0.052676	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=2897 Win=34816 Len=0 TSval=3913851400 TSecr=725607532
9	0.052774	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=4345 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851399 [TCP segment o
10	0.052775	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=5793 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851399 [TCP segment o
11	0.052854	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=7241 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851400 [TCP segment o
12	0.052855	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=8689 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851400 [TCP segment o
13	0.053626	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=4345 Win=37760 Len=0 TSval=3913851400 TSecr=725607532
14	0.053710	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=10137 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851400 [TCP segment
15	0.053711	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=11585 Ack=1 Win=131712 Len=1448 TSval=725607560 TSecr=3913851400 [TCP segment
16	0.080768	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=5793 Win=40576 Len=0 TSval=3913851421 TSecr=725607560
17	0.080771	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=7241 Win=43520 Len=0 TSval=3913851422 TSecr=725607560
18	0.080772	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=8689 Win=46336 Len=0 TSval=3913851422 TSecr=725607560
19	0.080772	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=10137 Win=49280 Len=0 TSval=3913851422 TSecr=725607560
20	0.080845	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=13033 Ack=1 Win=131712 Len=1448 TSval=725607588 TSecr=3913851421 [TCP segment

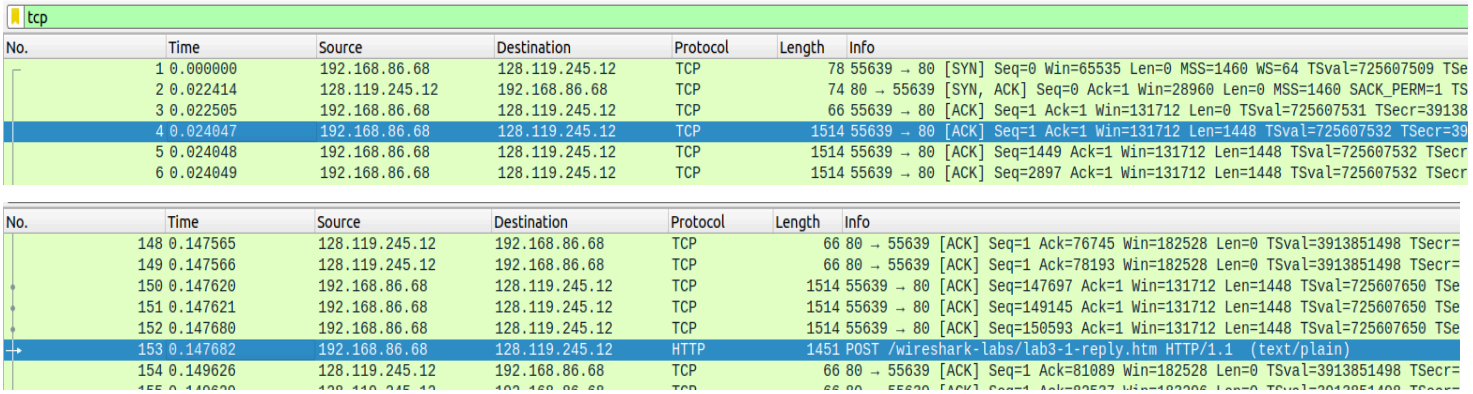
We can identify cases where the receiver is ACKing every other received segment among these first ten data-carrying segments.

- The receiver does sends an ACK for every single segment.
- Because, it acknowledges every segments (after receiving 1448 bytes)

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

Ans: First data-carrying packet : pkt #4 (time = 0.024047s)
Last data segment (HTTP POST end) : pkt #153 (time = 0.147682s)
Total data transferred = 153425 bytes

Throughput = Total bytes / transfer duration
= 153425 / (0.147682 - 0.024047)
= 1240951.187 bytes/s



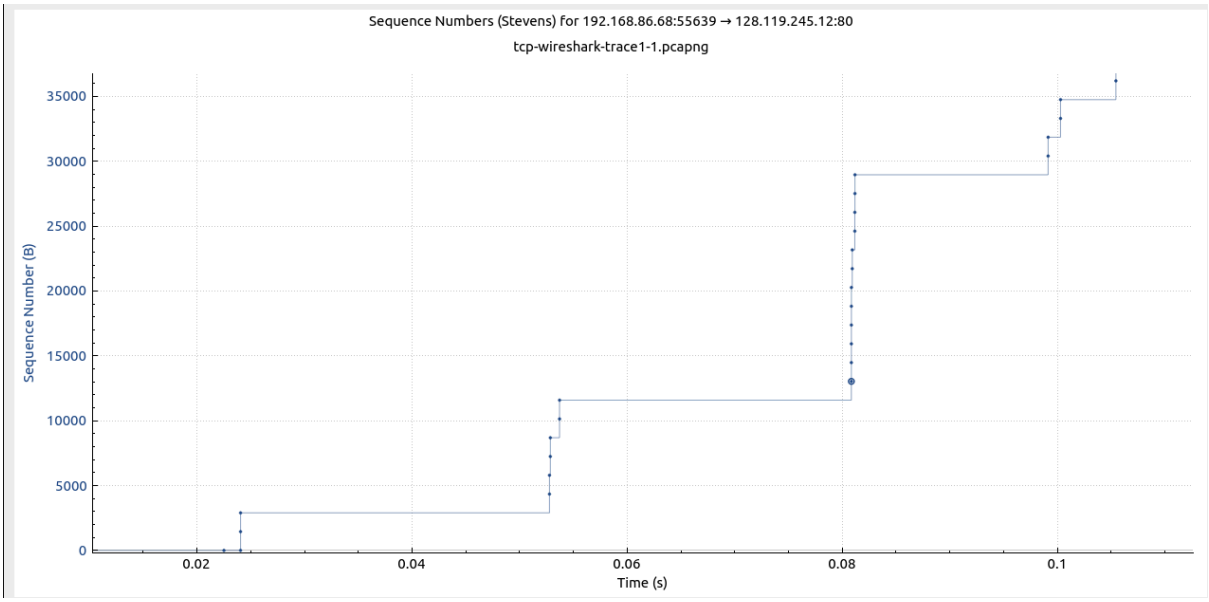
tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.86.68	128.119.245.12	TCP	78	55639 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=725607509 TSe
2	0.022414	128.119.245.12	192.168.86.68	TCP	74	80 → 55639 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TS
3	0.022505	192.168.86.68	128.119.245.12	TCP	66	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=725607531 TSecr=39138
4	0.024047	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr=39
5	0.024048	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=1449 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr
6	0.024049	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=2897 Ack=1 Win=131712 Len=1448 TSval=725607532 TSecr

No.	Time	Source	Destination	Protocol	Length	Info
148	0.147565	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=76745 Win=182528 Len=0 TSval=3913851498 TSecr=
149	0.147566	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=78193 Win=182528 Len=0 TSval=3913851498 TSecr=
150	0.147620	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=147697 Ack=1 Win=131712 Len=1448 TSval=725607650 TSe
151	0.147621	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=149145 Ack=1 Win=131712 Len=1448 TSval=725607650 TSe
152	0.147680	192.168.86.68	128.119.245.12	TCP	1514	55639 → 80 [ACK] Seq=150593 Ack=1 Win=131712 Len=1448 TSval=725607650 TSe
153	0.147682	192.168.86.68	128.119.245.12	HTTP	1451	POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
154	0.149626	128.119.245.12	192.168.86.68	TCP	66	80 → 55639 [ACK] Seq=1 Ack=81089 Win=182528 Len=0 TSval=3913851498 TSecr=

12. 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. Consider the “fleets” of packets sent around t = 0.025, t = 0.053, t = 0.082 and t = 0.1. Comment on whether this looks as if TCP is in its slow start phase, congestion avoidance phase or some other phase. Figure 6 shows a slightly different view of this data.

Ans:



Interpretation:

- The number of segments per fleet doubles early (slow start behavior).

- Then, fleets are spaced more evenly and of similar size - transition to congestion avoidance.

So the connection starts in Slow Start, then shifts toward Congestion Avoidance.

13. These “fleets” of segments appear to have some periodicity. What can you say about the period?

Ans: Approximate fleet intervals:

0.053 - 0.025 = 0.028 s
0.082 - 0.053 = 0.029 s
0.100 - 0.082 = 0.018 s

Average period ≈ 0.025 seconds (25 ms)
This matches the RTT from earlier (~28 ms), showing each fleet corresponds to one RTT’s worth of ACK-triggered transmissions — ACK-clocked behavior.