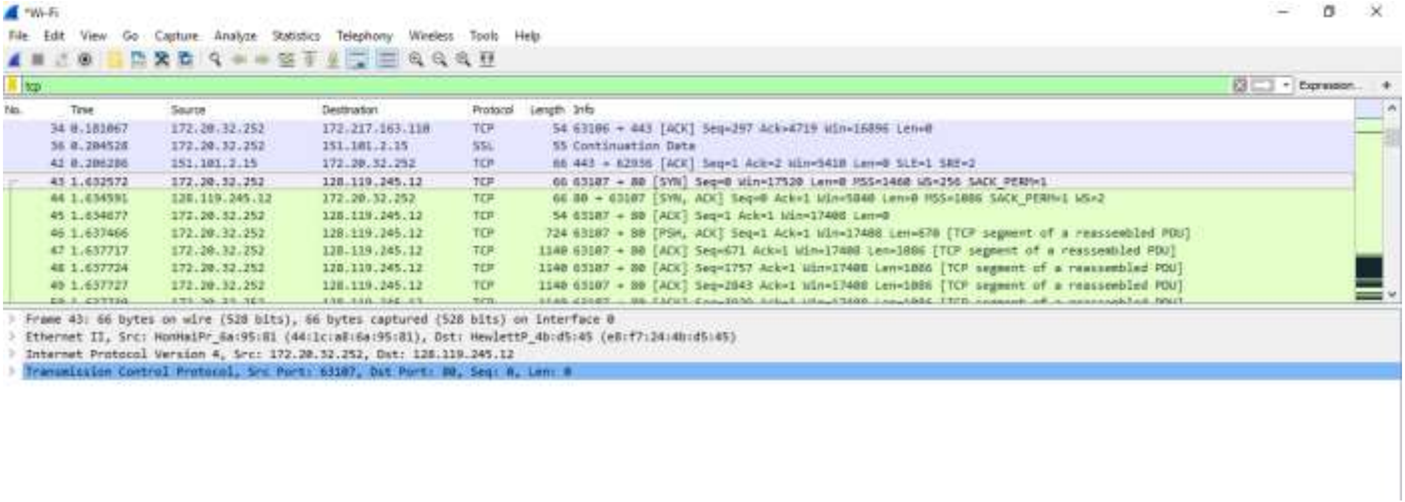
DHRUV GARG 16 BCE 1190

NETWORK LAB: WIRESHARK

TCP

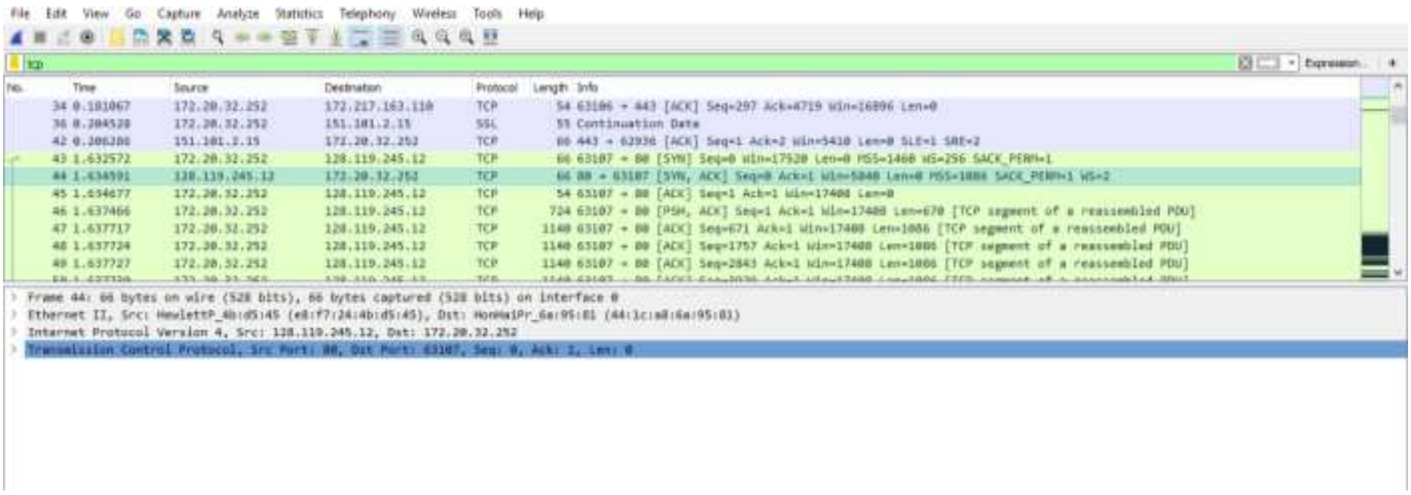
1. **1.** What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details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

   <u>**ANS:**</u> The client IP address is *172.20.20.352*, TCP port number is *63017*



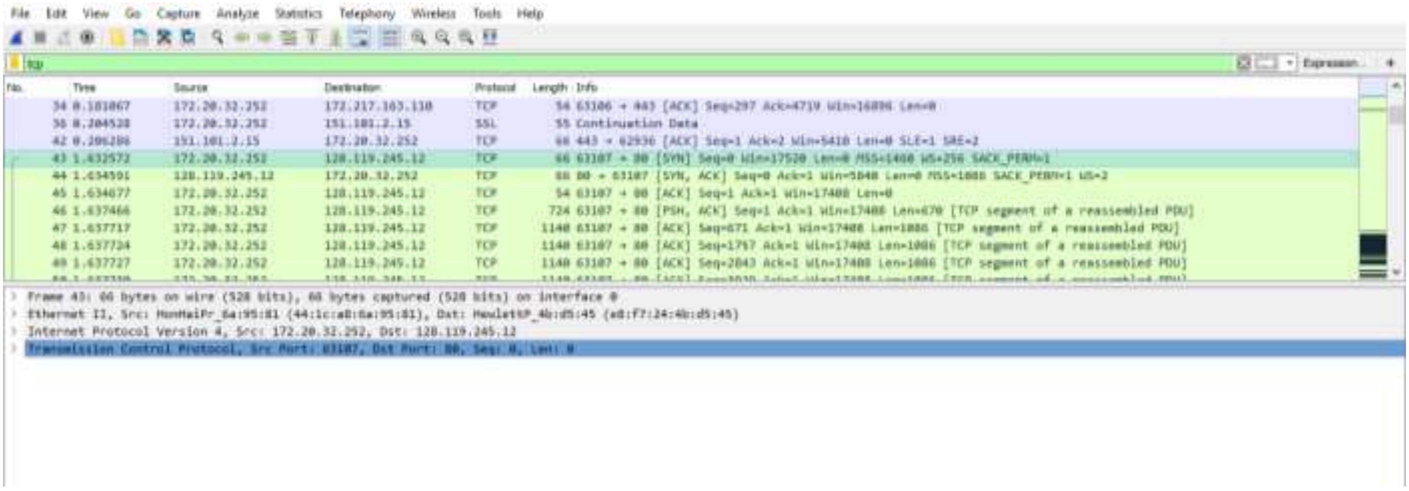
2. **2.** What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

   <u>**ANS:**</u> gaia.cs.umass.edu's IP address is *128.110.245.12*, port number is *80*



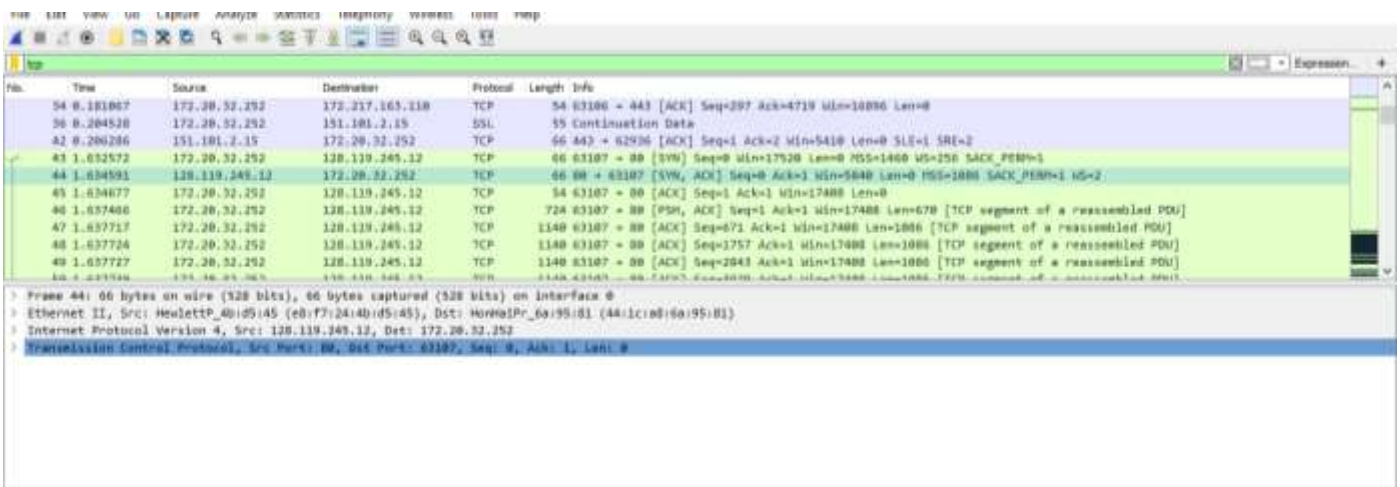
3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

<u>**ANS:** </u>The sequence number of the TCP SYN segment is 0 since it is used to imitate the TCP connection between the client computer and gaia.cs.umass.edu. According to the screenshot below, in the Flags section, the SYN flag is set to 1 which indicates that this segment is a SYN segment.



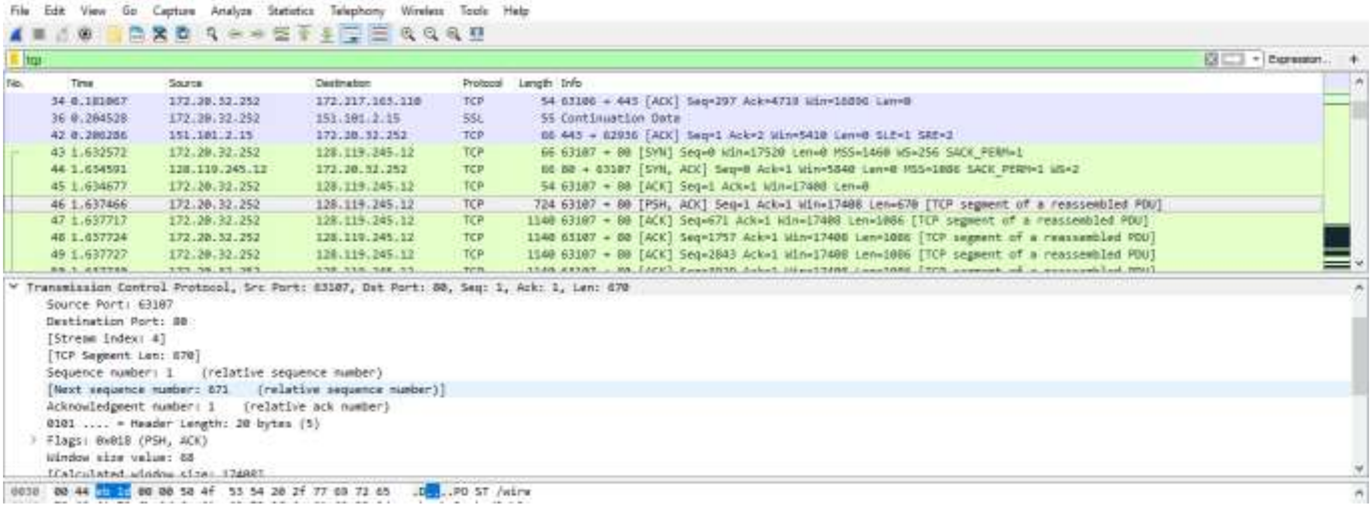
4. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK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

<u>**ANS:** </u>According to the screenshot below, the sequence number of the SYN_ACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0. The value of the acknowledgement field in the SYN_ACK segment is determined by the server gaia.cs.umass.edu. The server adds 1 to the initial sequence number of the SYN segment from the client computer. For this case, the initial sequence number of the SYN segment from the client computer is 0, thus the value of the acknowledgement field in the SYN_ACK segment is 1. A segment will be identified as a SYN_ACK segment if both SYN flag and Acknowledgement flag in the segment are set to 1.



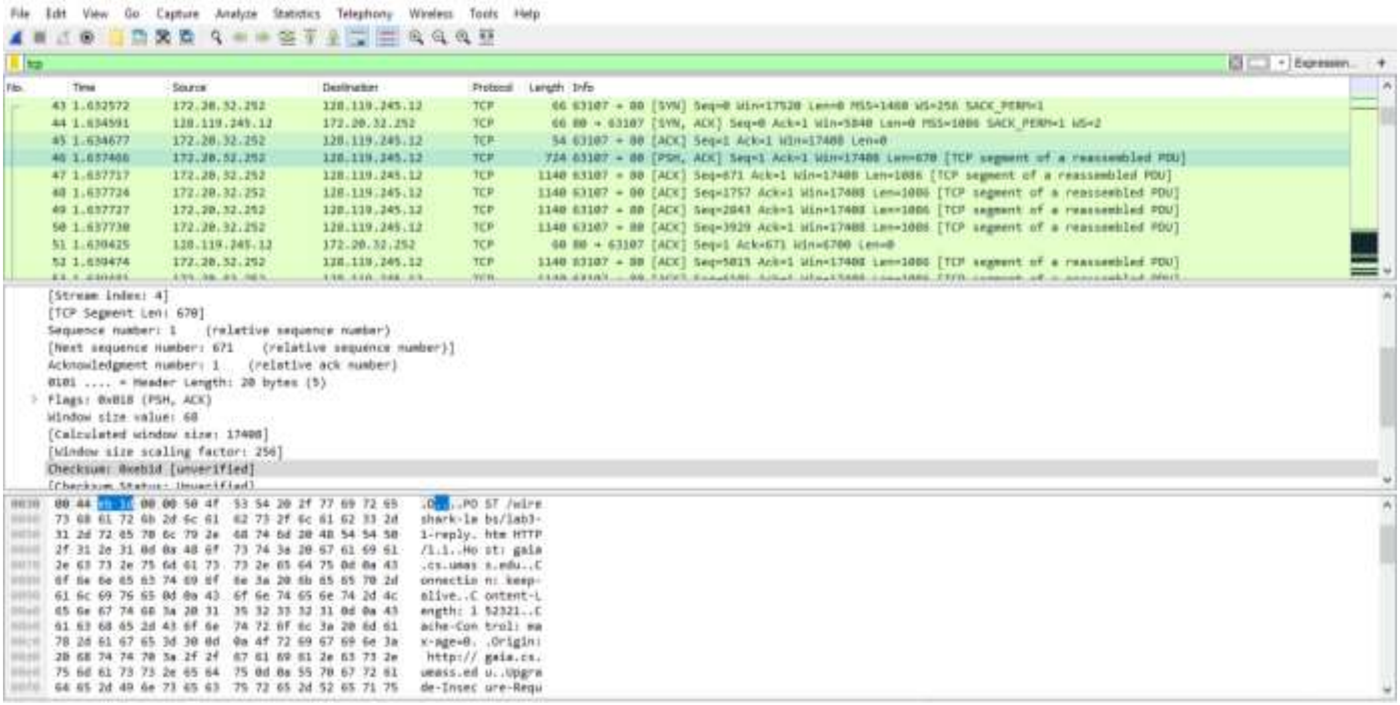
5. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

ANS: The sequence number of the TCP segment containing the HTTP Post command is 1.



6. Consider the TCP connection.
   a. What are the sequence numbers of the first six segments in the TCP connection?

ANS: Sequence number for segment 1 is 1, sequence number for segment 2 is 671 , for segment 3 is 1757, for segment 4 is 2843, for segment 5 is 3939.

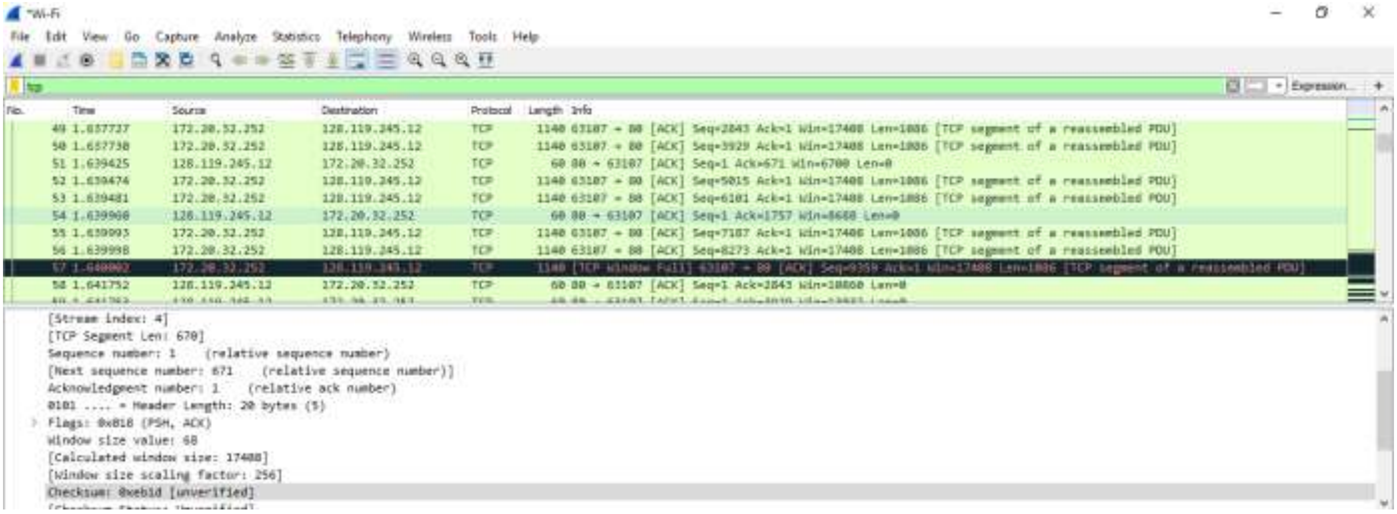


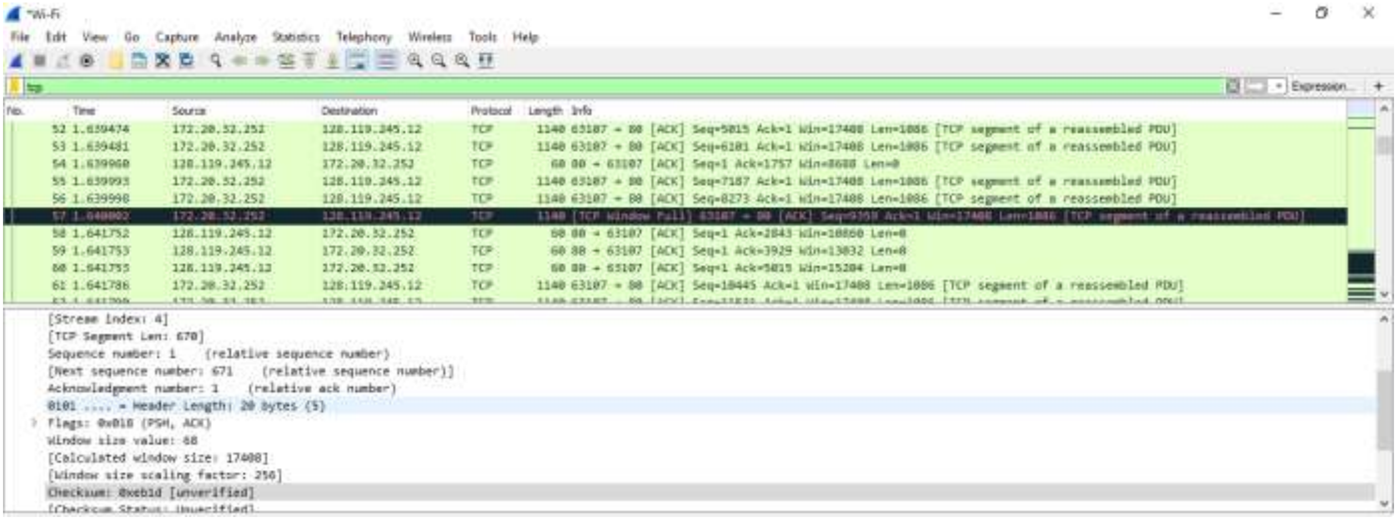
   b. At what time was each segment sent?

ANS: 1.637466 s for segment 1, 1.637717 s for segment 2,1.637724s for segment 3,1.637727s for segment 4, 1.637730s for segment 5. Screenshot same as above.

c. When was the ACK for each segment received?

ANS: ACK for segment 1 was received at 1.639425 s, ACK for segment 2 is received at 1.639960 s, ACK for segment 3 is received at 1.64172 s, ACK for segment 4 is received at 1.64173 s, ACK for segment 5 is received at 1.64173 s.



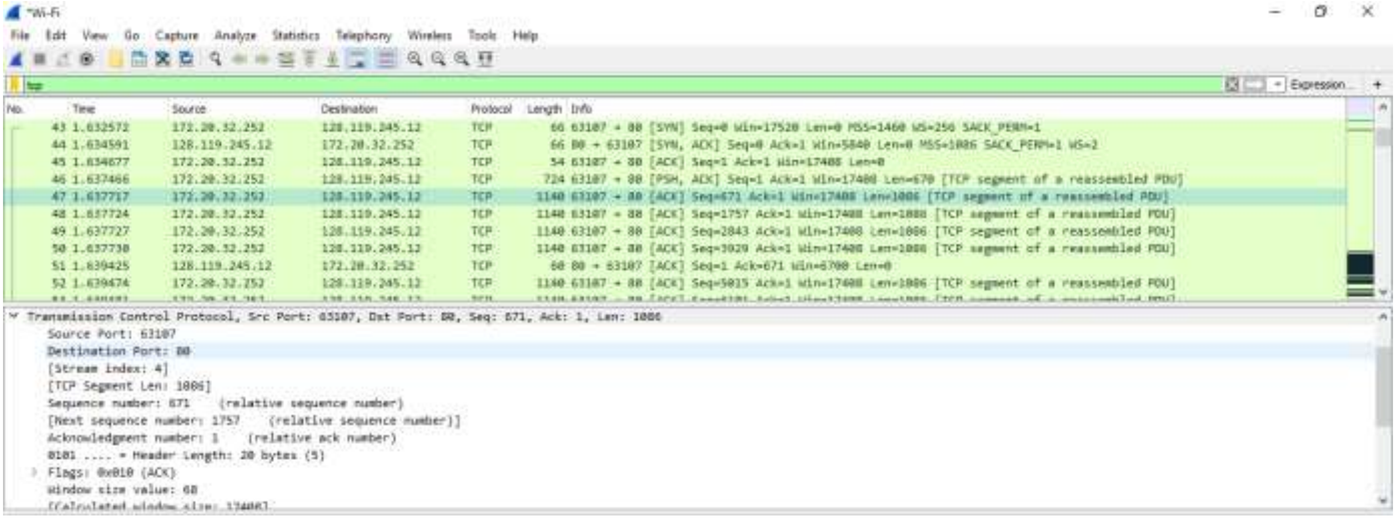


d. Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments?

ANS: RTT for segment 1 is 0.001959 seconds, RTT for segment 2 is 0.002243 seconds, RTT for segment 3 is 0.003996 seconds, RTT for segment 4 is 0.004003 seconds, RTT for segment 5 is 0.004 seconds.

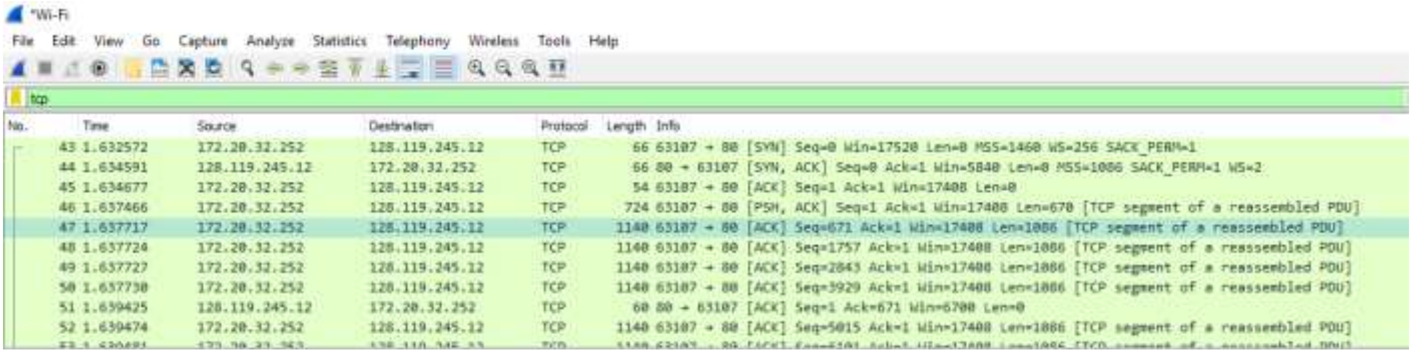
7. What is the length of each of the first six TCP segments?

ANS: The length of each of the first 6 TCP segments is 1086 bytes.



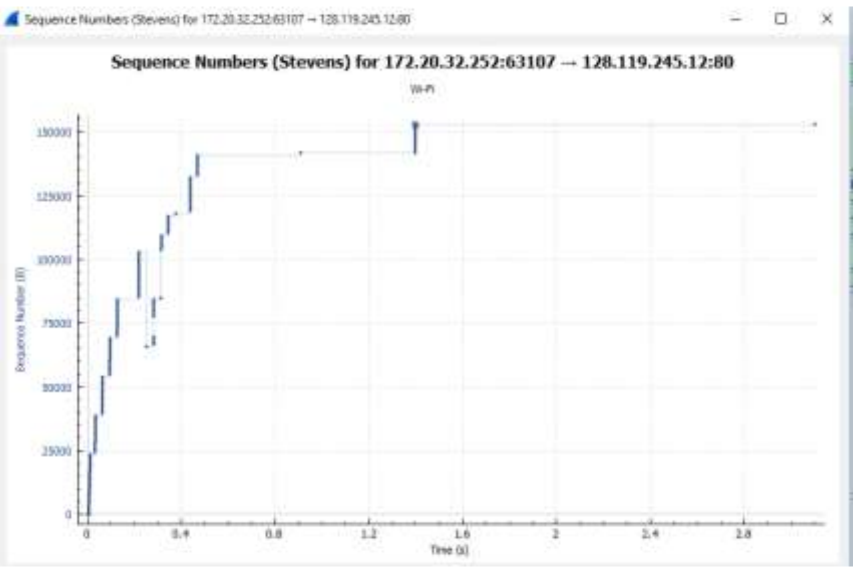
8. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

ANS: The minimum amount of available buffer space advertised at the received is 17408 bytes



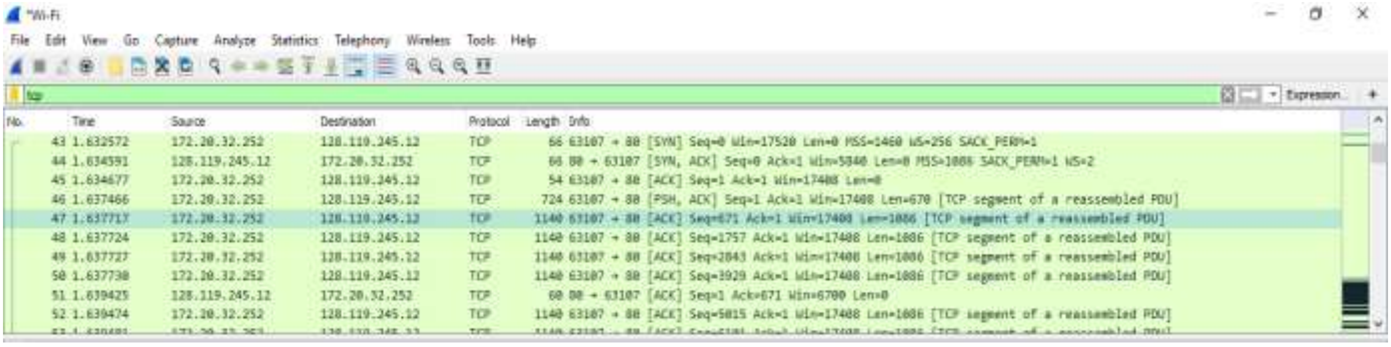
9. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

ANS: Yes, there is no retransmitted segments in the trace file. This can be explained by packets with same sequence number at different time are found.



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

ANS: According to the screenshot below, we can see that the ACK numbers increase in the sequence of 671, 1757, 2849, and so on. The ACK numbers increases by 670,1086,1086 and so on. Receiver is acknowledging 1086 data after the first packet.



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

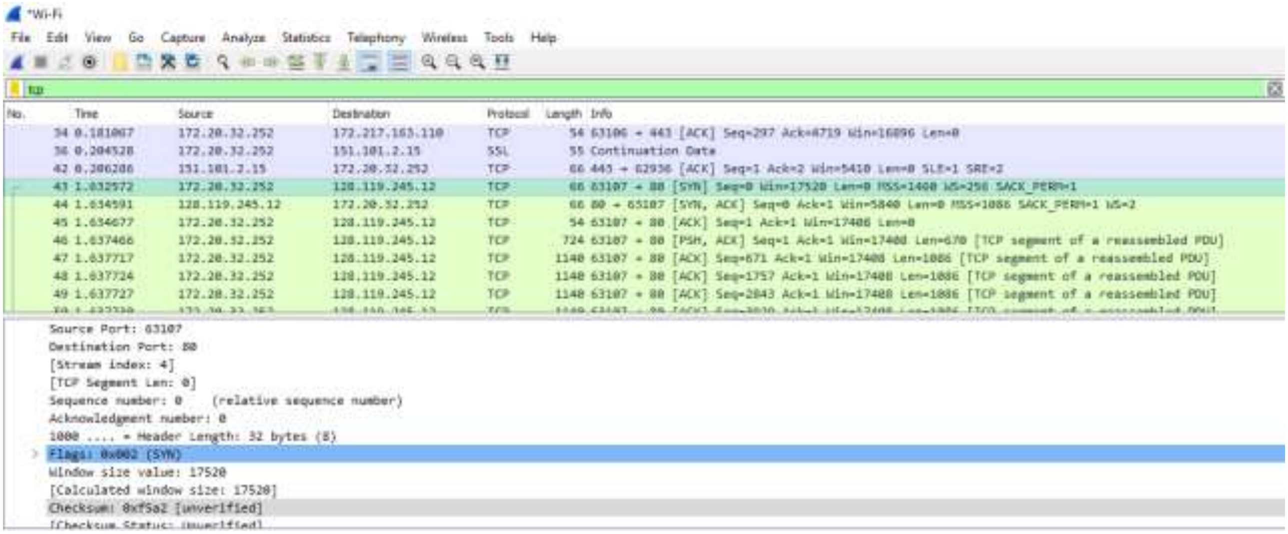
ANS:

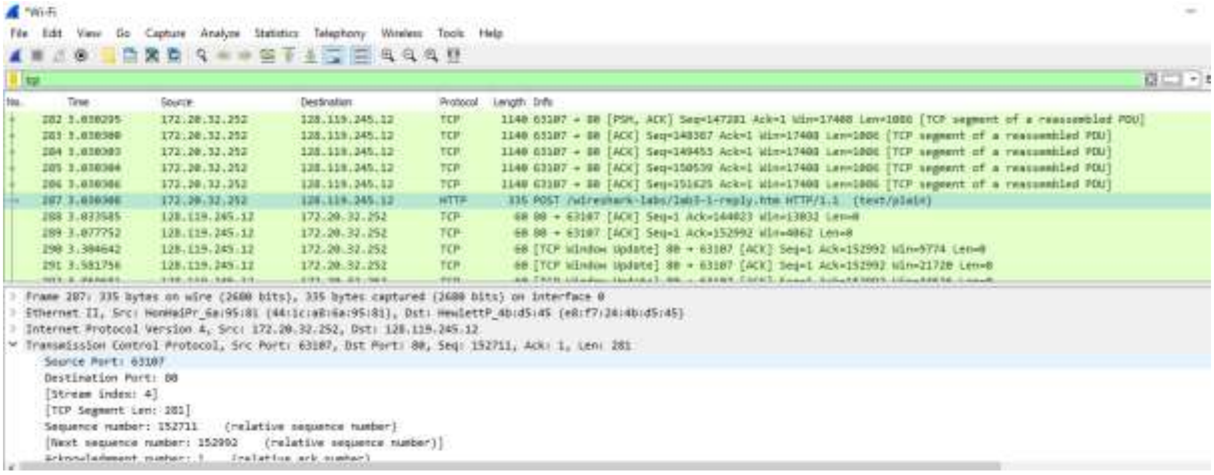
Throughput= Amount of data sent/time incurred

Time = 3.030308-1.632572=1.397736

Data sent = 152711 bytes

Throughput=152711/1.397736= 109255.9682228 bytes/se

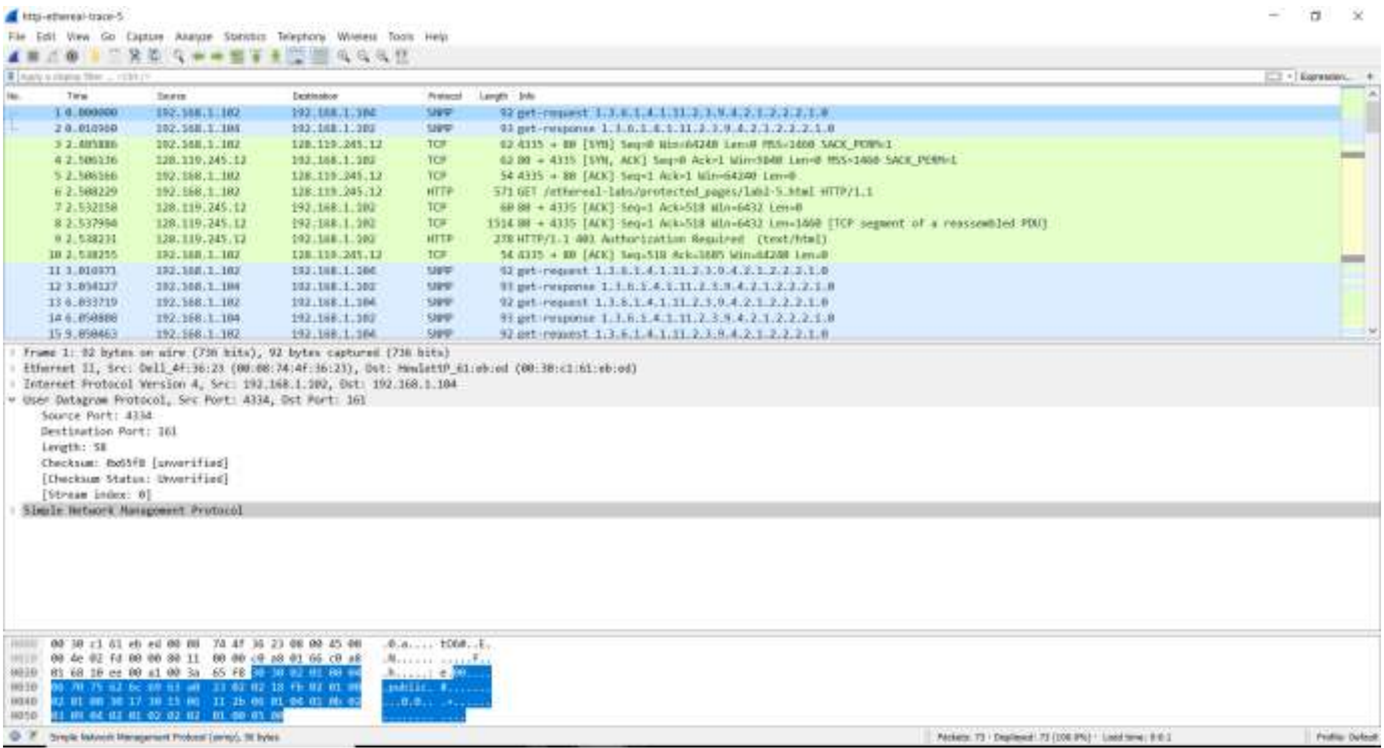




12. Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slow start phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behaviour of TCP that we've studied in the text.

ANS: By observing the plot, we can see that the slow-start phase only lasts for first 1-1.5 second. Afterwards, it seems that the TCP session is always in congestion avoidance state. In this case, we do not observe the expected linear increase behaviour, i.e. the TCP transmit window does not grow linearly during this phase. In fact, it appears that the sender transmits packets in batches of 6. This does not seem to be caused by flow control since the receiver advertised window is significantly larger than 5 packets. The reason for this behaviour might be due to the fact that the HTTP server has enforced a rate-limit of some sort.

UDP



DHRUV GARG 16 BCE 1190

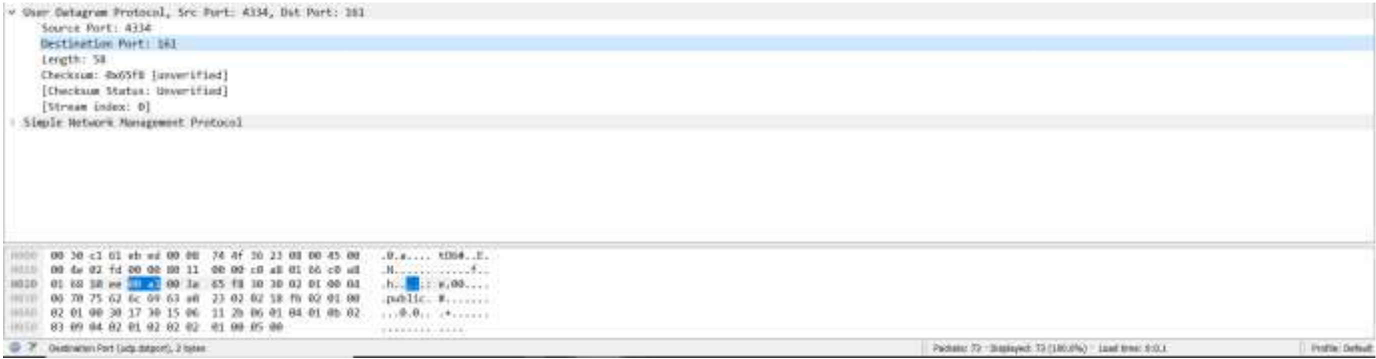
1. Select *one* UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields.

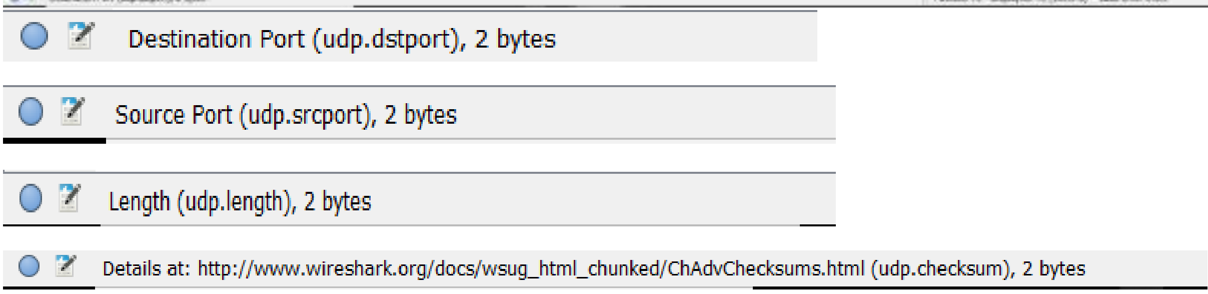
ANS:

Four fields – 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.

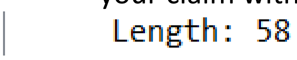
ANS:





Two Bytes

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: 8 bytes UDP packet header added with 50 bytes payload Simple Network Management Protocol equals to the length of 58 bytes.

```
v User Datagram Protocol, Src Port: 4334, Dst Port: 161
    Source Port: 4334
    Destination Port: 161
    Length: 58
    Checksum: 0x65f8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
> Simple Network Management Protocol

0000  00 30 c1 61 eb ed 00 08  74 4f 36 23 08 00 45 00   .0.a.... tO6#..E.
0010  00 4e 02 fd 00 00 80 11  00 00 c0 a8 01 66 c0 a8   .N...... .....f..
0020  01 68 10 ee 00 a1 00 3a  65 f8 30 30 02 01 00 04   .h.....: e.00....
0030  06 70 75 62 6c 69 63 a0  23 02 02 18 fb 02 01 00   .public. #.......
0040  02 01 00 30 17 30 15 06  11 2b 06 01 04 01 0b 02   ...0.0.. .+......
0050  03 09 04 02 01 02 02 02  01 00 05 00               ........ ....

User Datagram Protocol (udp), 8 bytes
```

```
v User Datagram Protocol, Src Port: 4334, Dst Port: 161
    Source Port: 4334
    Destination Port: 161
    Length: 58
    Checksum: 0x65f8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
> Simple Network Management Protocol

0000  00 30 c1 61 eb ed 00 08  74 4f 36 23 08 00 45 00   .0.a.... tO6#..E.
0010  00 4e 02 fd 00 00 80 11  00 00 c0 a8 01 66 c0 a8   .N...... .....f..
0020  01 68 10 ee 00 a1 00 3a  65 f8 30 30 02 01 00 04   .h.....: e.00....
0030  06 70 75 62 6c 69 63 a0  23 02 02 18 fb 02 01 00   .public. #.......
0040  02 01 00 30 17 30 15 06  11 2b 06 01 04 01 0b 02   ...0.0.. .+......
0050  03 09 04 02 01 02 02 02  01 00 05 00               ........ ....

Simple Network Management Protocol (snmp), 50 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: Length field size is 2 bytes = 16 bits.
Maximum number of bytes that can be included in UDP = 2^{16} -1 less the header bytes. This gives 65535-8=65527 bytes

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

ANS: 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. 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:

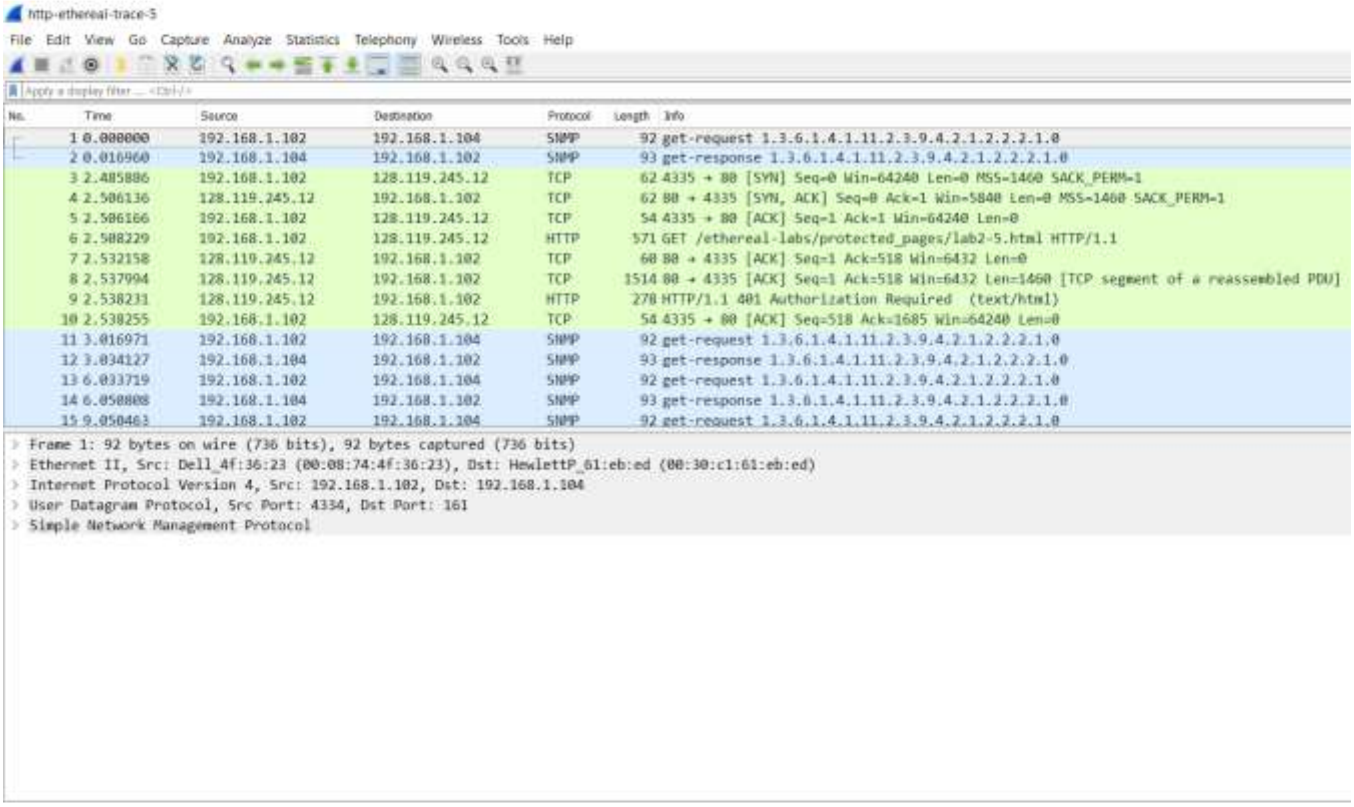
Protocol number in decimals is 17 and in hexadecimal is 11

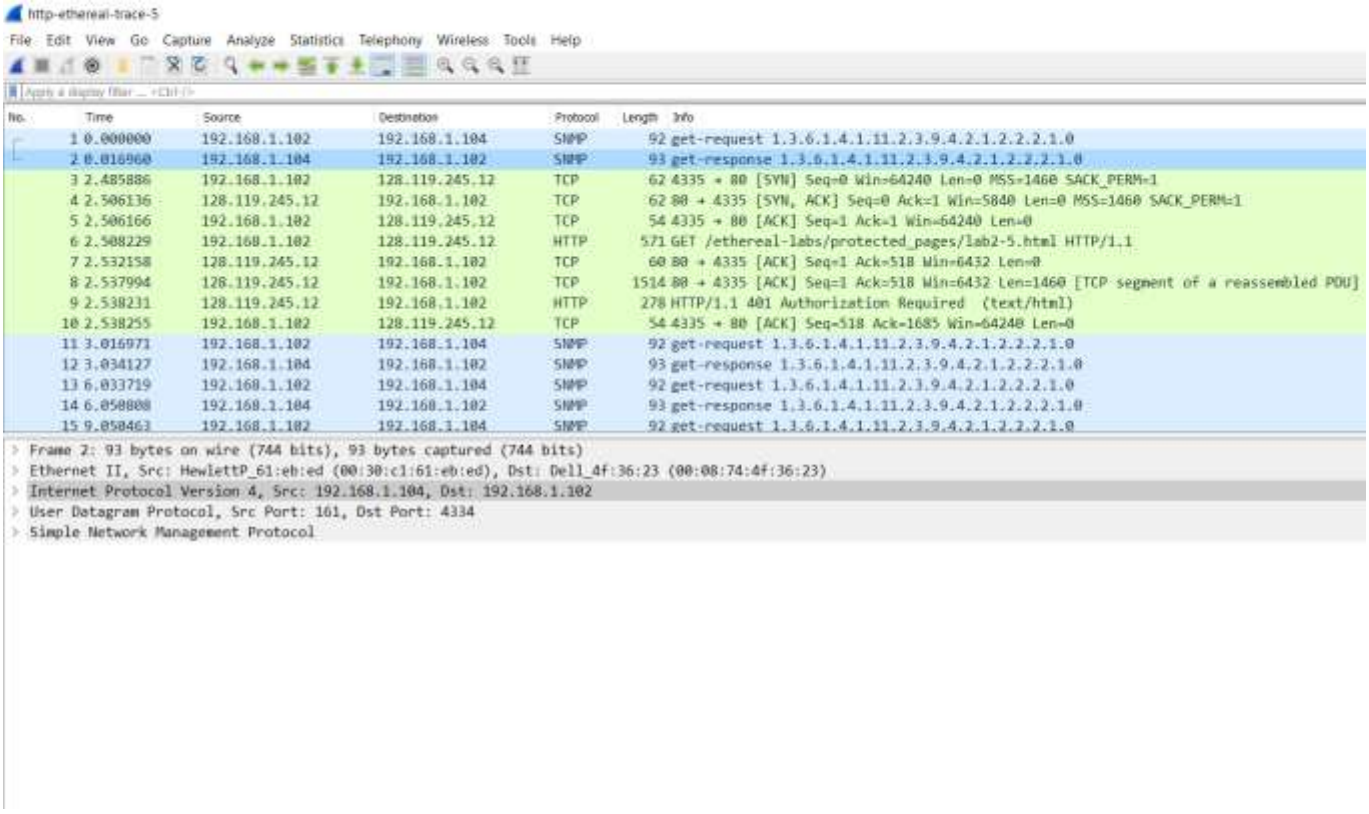
```
> Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
> Ethernet II, Src: Dell_4f:36:23 (00:08:74:4f:36:23), Dst: HewlettP_61:eb:ed (00:30:c1:61:eb:ed)
v Internet Protocol Version 4, Src: 192.168.1.102, Dst: 192.168.1.104
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 78
    Identification: 0x02fd (765)
  > Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (17)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.102
    Destination: 192.168.1.104
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]

0000  00 30 c1 61 eb ed 00 08  74 4f 36 23 08 00 45 00   .0.a.... tO6#..E.
0010  00 4e 02 fd 00 00 80 11  00 00 c0 a8 01 66 c0 a8   .N...... .....f..
0020  01 68 10 ee 00 a1 00 3a  65 f8 30 30 02 01 00 04   .h.....: e.00....
0030  06 70 75 62 6c 69 63 a0  23 02 02 18 fb 02 01 00   .public. #.......
0040  02 01 00 30 17 30 15 06  11 2b 06 01 04 01 0b 02   ...0.0.. .+......
0050  03 09 04 02 01 02 02 02  01 00 05 00               ........ ....

Protocol (ip.proto), 1 byte
```

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.

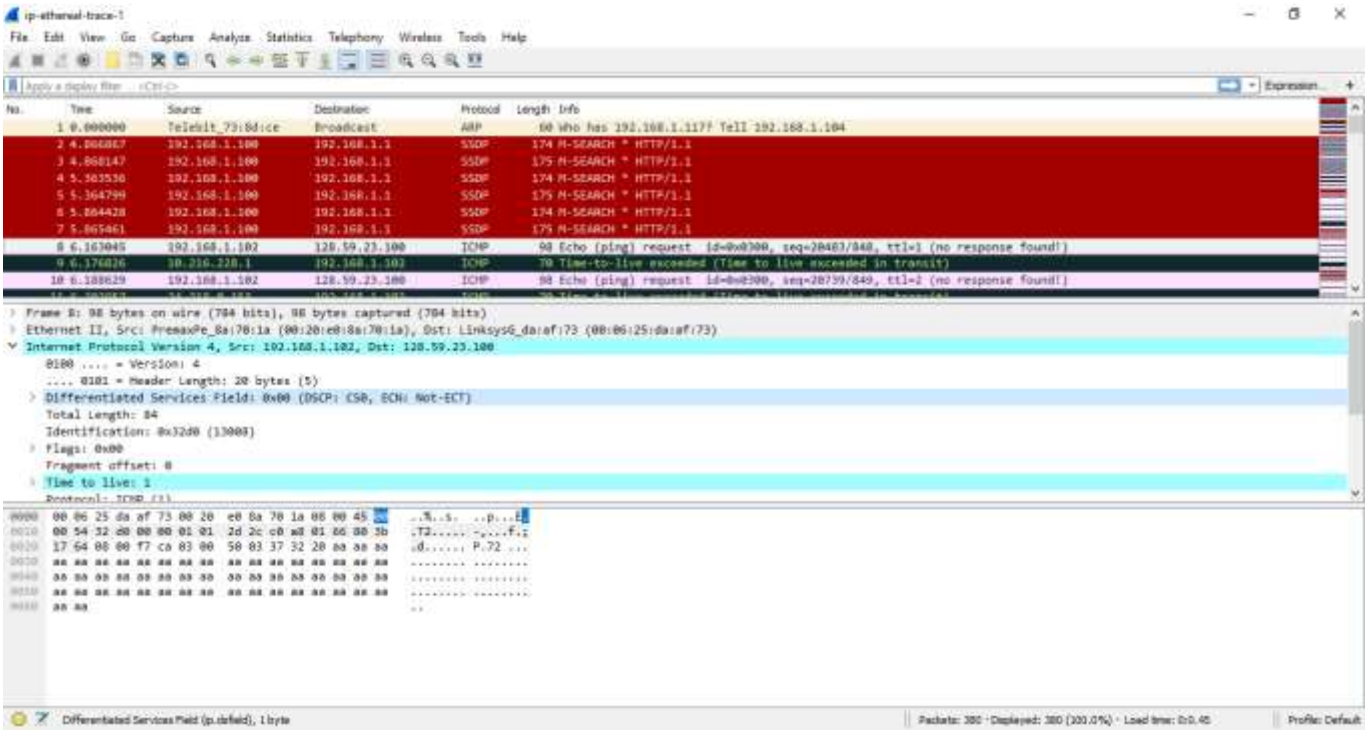
ANS: The source port number(4334) from the source IP sends the request packet to the destination IP's destination port number(161). During the sending of a response, the source IP that sent the request packet becomes the destination and it's source port becomes the destination port. The response sender's IP and port number turns to the source.

DHRUV GARG 16 BCE 1190





IP



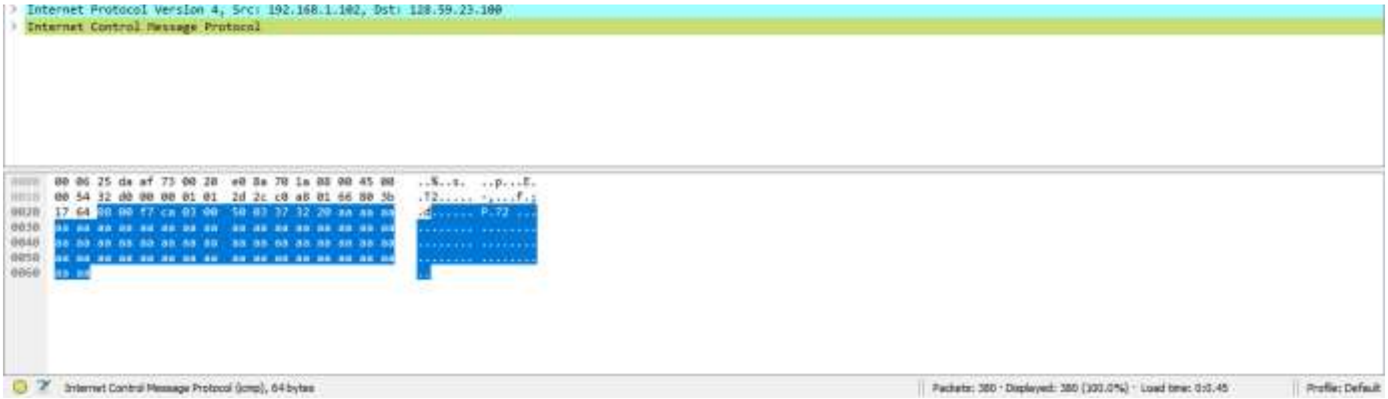
1. What is the IP address of your computer?
   ANS:

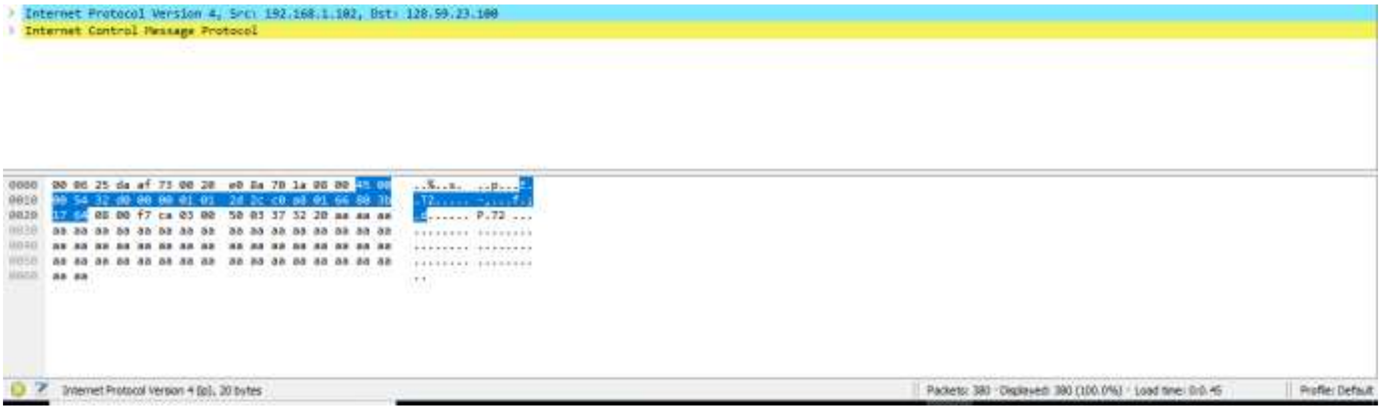
```
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
```

2. Within the IP packet header, what is the value in the upper layer protocol field?
   ANS:

```
Protocol: ICMP (1)
Header checksum: 0x2d2c [validation disabled]
[Header checksum status: Unverified]
```

3. How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.
   ANS:





ANS: There are 20 bytes in the IP header, and 64 bytes total length, this gives 84 bytes in the payload of the IP datagram.

4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.
   ANS: The more fragments bit = 0, so the data is not fragmented.

```
> Flags: 0x00
  Fragment offset: 0
```

5. Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by your computer?
   ANS: Identification, Time to live and Header checksum always change.

6. Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?

ANS: The fields that stay constant across the IP datagrams are:

Version (since we are using IPv4 for all packets), header length (since these are ICMP packets), source IP (since we are sending from the same source), destination IP (since we are sending to the same dest), Differentiated Services (since all packets are ICMP they use the same Type of Service class), Upper Layer Protocol (since these are ICMP packets)

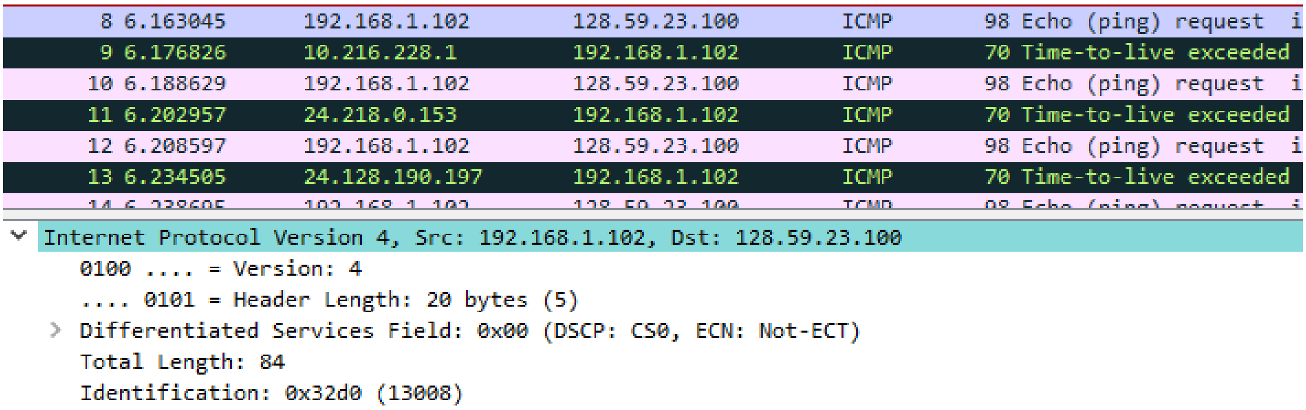
The fields that must stay constant are:

Version (since we are using IPv4 for all packets), header length (since these are ICMP packets), source IP (since we are sending from the same source), destination IP (since we are sending to the same dest), Differentiated Services (since all packets are ICMP they use the same Type of Service class), Upper Layer Protocol (since these are ICMP packets)

The fields that must change are:

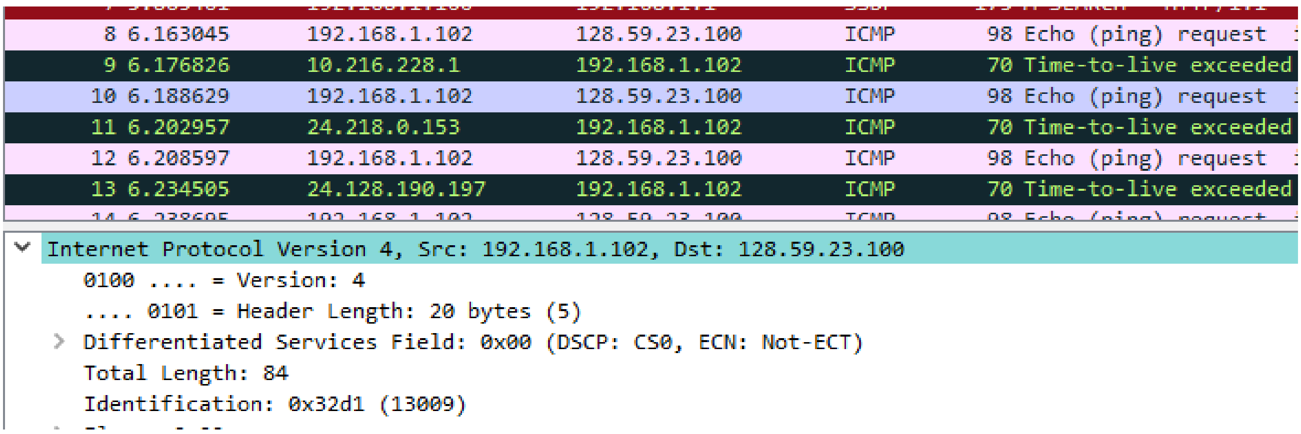
Identification(IP packets must have different ids), Time to live (traceroute increments each subsequent packet), Header checksum (since header changes, so must checksum)

7. Describe the pattern you see in the values in the Identification field of the IP datagram.
ANS:



8	6.163045	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request i
9	6.176826	10.216.228.1	192.168.1.102	ICMP	70 Time-to-live exceeded
10	6.188629	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request i
11	6.202957	24.218.0.153	192.168.1.102	ICMP	70 Time-to-live exceeded
12	6.208597	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request i
13	6.234505	24.128.190.197	192.168.1.102	ICMP	70 Time-to-live exceeded

Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x32d0 (13008)



8	6.163045	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request
9	6.176826	10.216.228.1	192.168.1.102	ICMP	70 Time-to-live exceeded
10	6.188629	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request
11	6.202957	24.218.0.153	192.168.1.102	ICMP	70 Time-to-live exceeded
12	6.208597	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request
13	6.234505	24.128.190.197	192.168.1.102	ICMP	70 Time-to-live exceeded

Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x32d1 (13009)

8 6.163045	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request i
9 6.176826	10.216.228.1	192.168.1.102	ICMP	70 Time-to-live exceeded
10 6.188629	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request i
11 6.202957	24.218.0.153	192.168.1.102	ICMP	70 Time-to-live exceeded
12 6.208597	192.168.1.102	128.59.23.100	ICMP	98 Echo (ping) request i
13 6.234505	24.128.190.197	192.168.1.102	ICMP	70 Time-to-live exceeded

```
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x32d2 (13010)
```

The pattern is that the IP header Identification fields increment with each ICMP Echo (ping) request.

8. What is the value in the Identification field and the TTL field?
   ANS:

```
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x32d0 (13008)
  > Flags: 0x00
    Fragment offset: 0
  > Time to live: 1
```

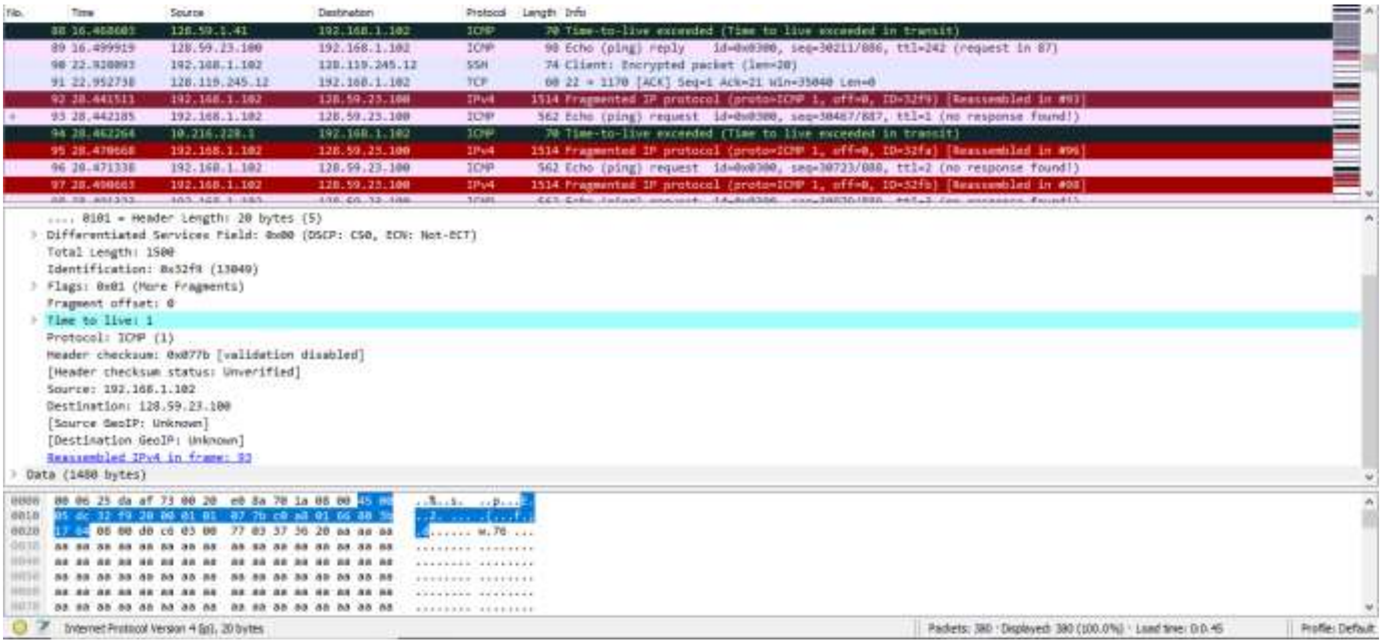
9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

   ANS: The identification field changes for all the ICMP TTL-exceeded replies because the identification field is a unique value. When two or more IP datagrams have the same identification value, then it means that these IP datagrams are fragments of a single large IP datagram. The TTL field remains unchanged because the TTL for the first hop router is always the same.

10. Find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 2000. Has that message been fragmented across more than one IP datagram?

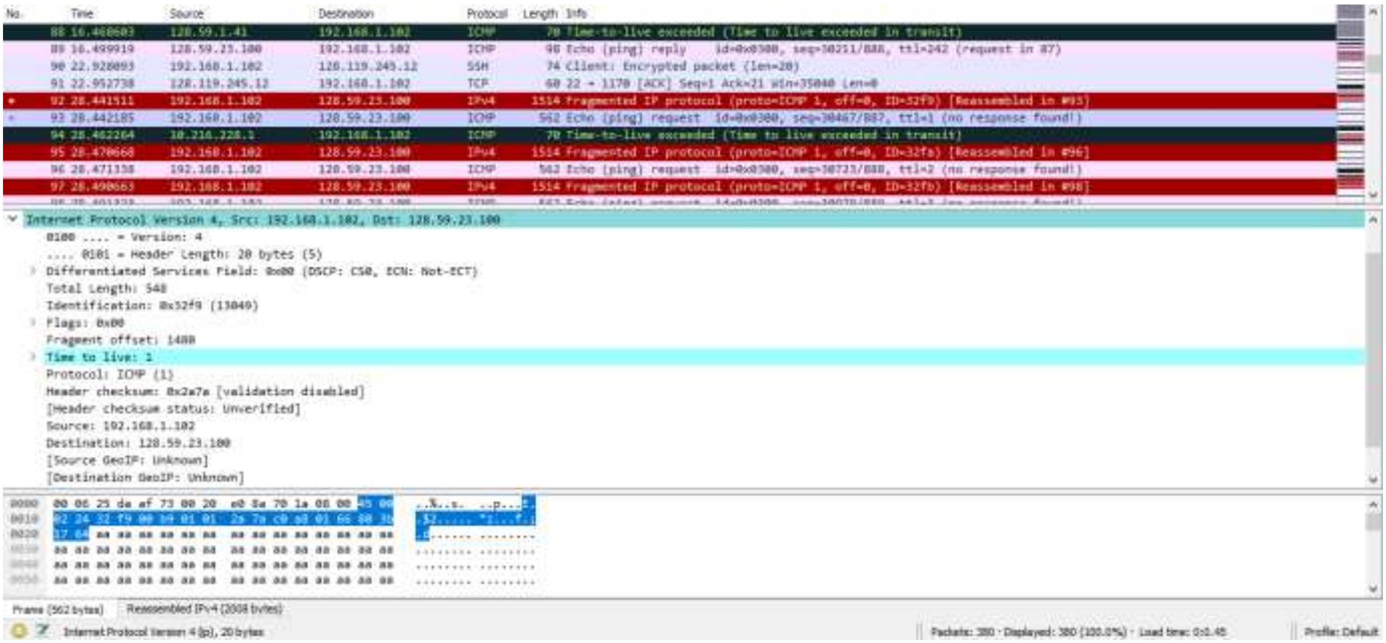
    ANS: Yes, this packet has been fragmented across more than one IP datagram

11. Print out the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?
    ANS:



12. Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?

<u>ANS:</u>

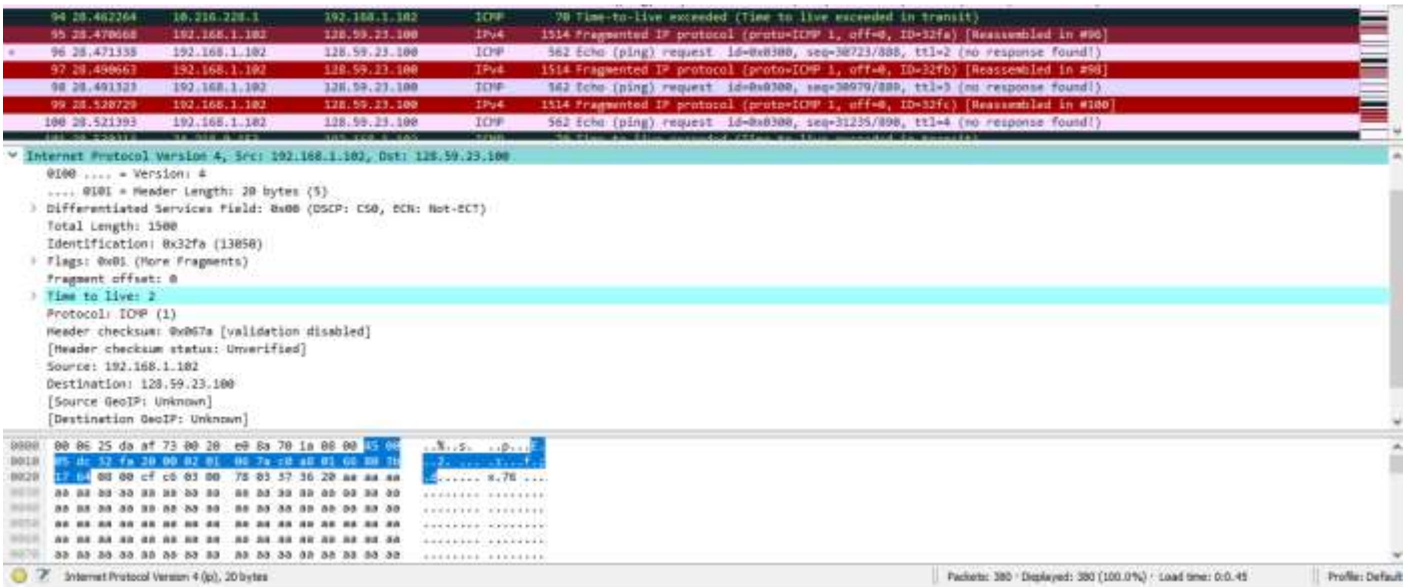


13. What fields change in the IP header between the first and second fragment?

<u>ANS:</u> The IP header fields that changed between the fragments are: total length, flags, fragment offset, and checksum.

14. How many fragments were created from the original datagram?

<u>ANS:</u>



15. What fields change in the IP header among the fragments?

<u>ANS:</u> The IP header fields that changed between all of the packets are: fragment offset, and checksum. Between the first two packets and the last packet, we see a change in total length, and also in the flags. The first two packets have a total length of 1500, with the more fragments bit set to 1, and the last packet has a total length of 548, with the more fragments bit set to 0.