Task 1

1. What does the above ping command do?

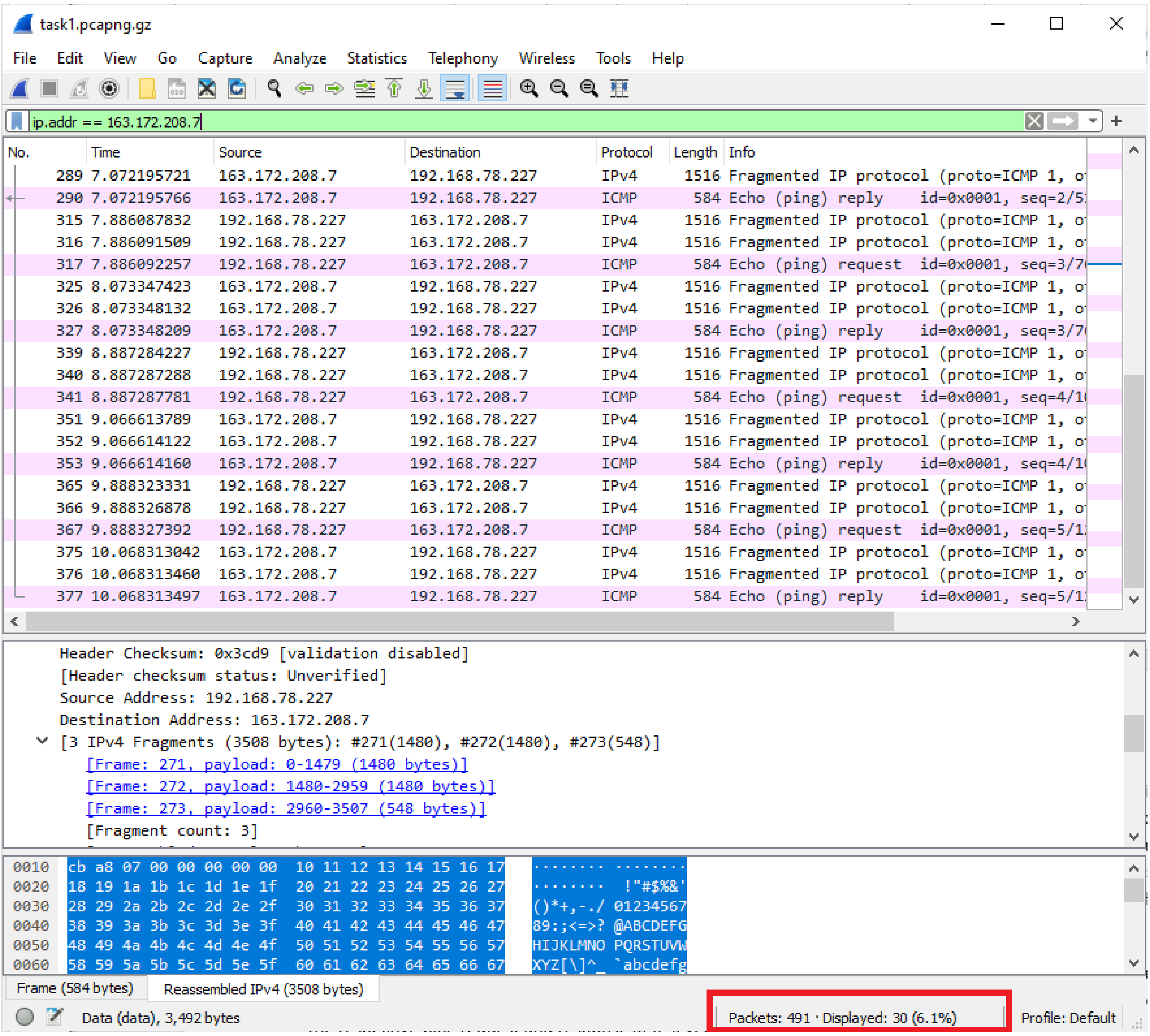
   The ping command sends a ICMP echo request to the defined server. It uses ICMP protocol's ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host.
   The options used in the command are:

   -s: Specifies the size of ping packet to be sent. We have set this to 3500 bytes which contains 8 bytes of ICMP header data and 3492 bytes of data.

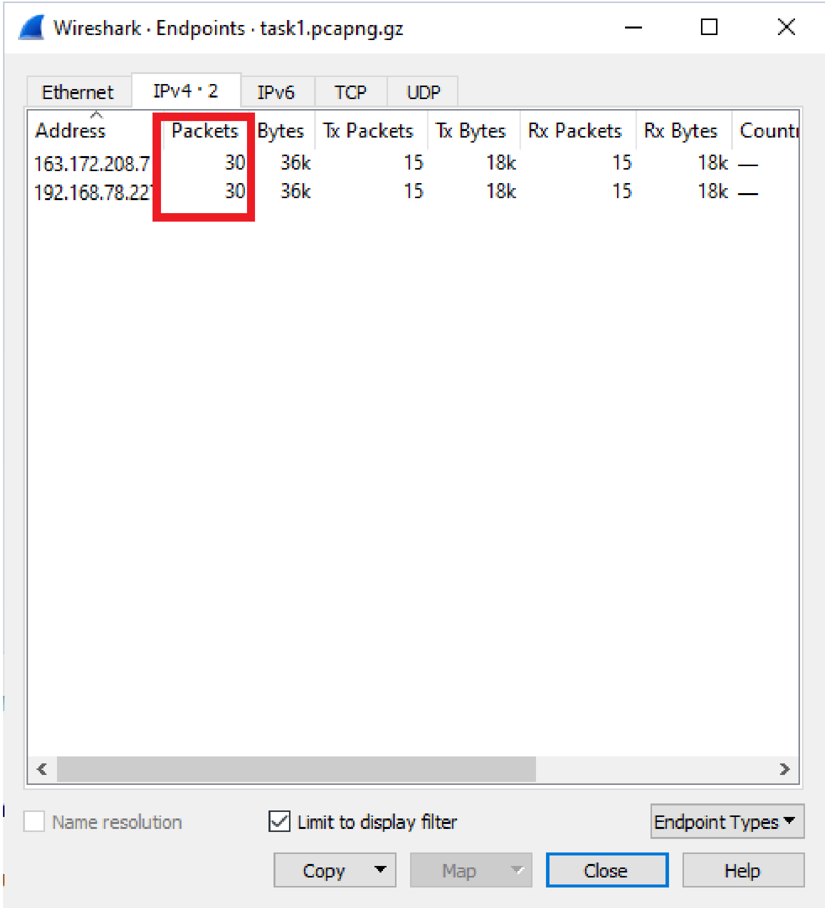
   -c: It defines the count of the number of ping packets to be sent. We have defined 5 so a total of 5 ping packets are sent to the host.

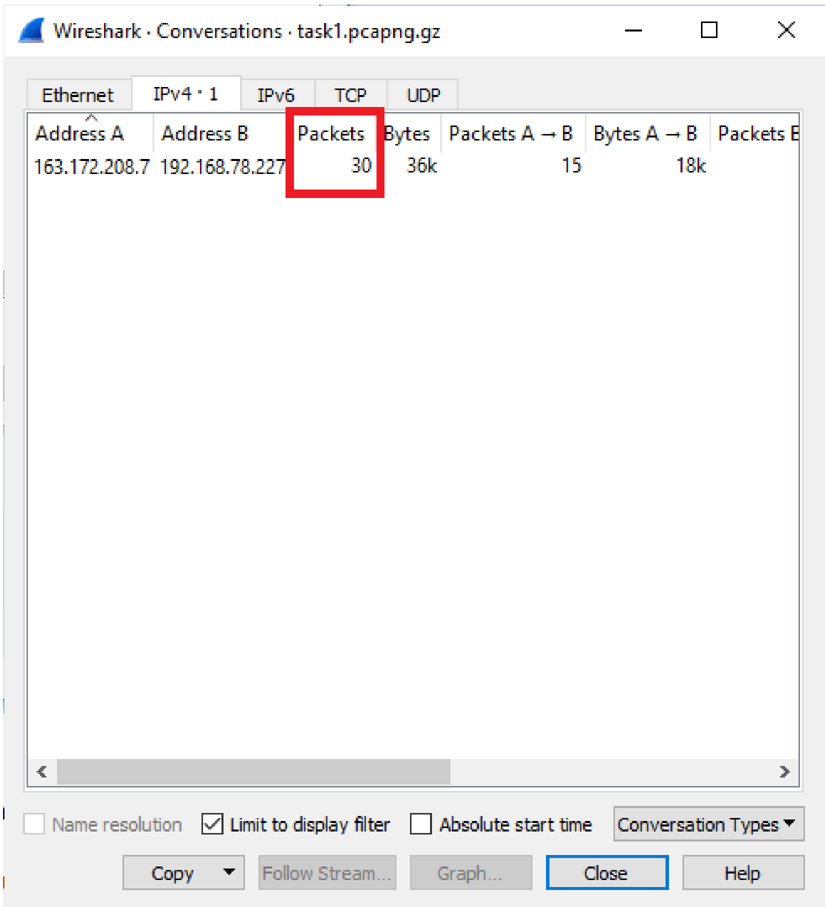
2. How many total IP packets are exchanged in the communication between your host and the remote server representing **ping-ams1.online.net ? count fragments**

   We find the IP address of ping-ams1.online.net is 163.172.208.7. We use the filter *ip.addr == 163.172.208.7* to filter packets from our capture. We find that a total of 30 packets were sent / received during this communication between our host and the server.



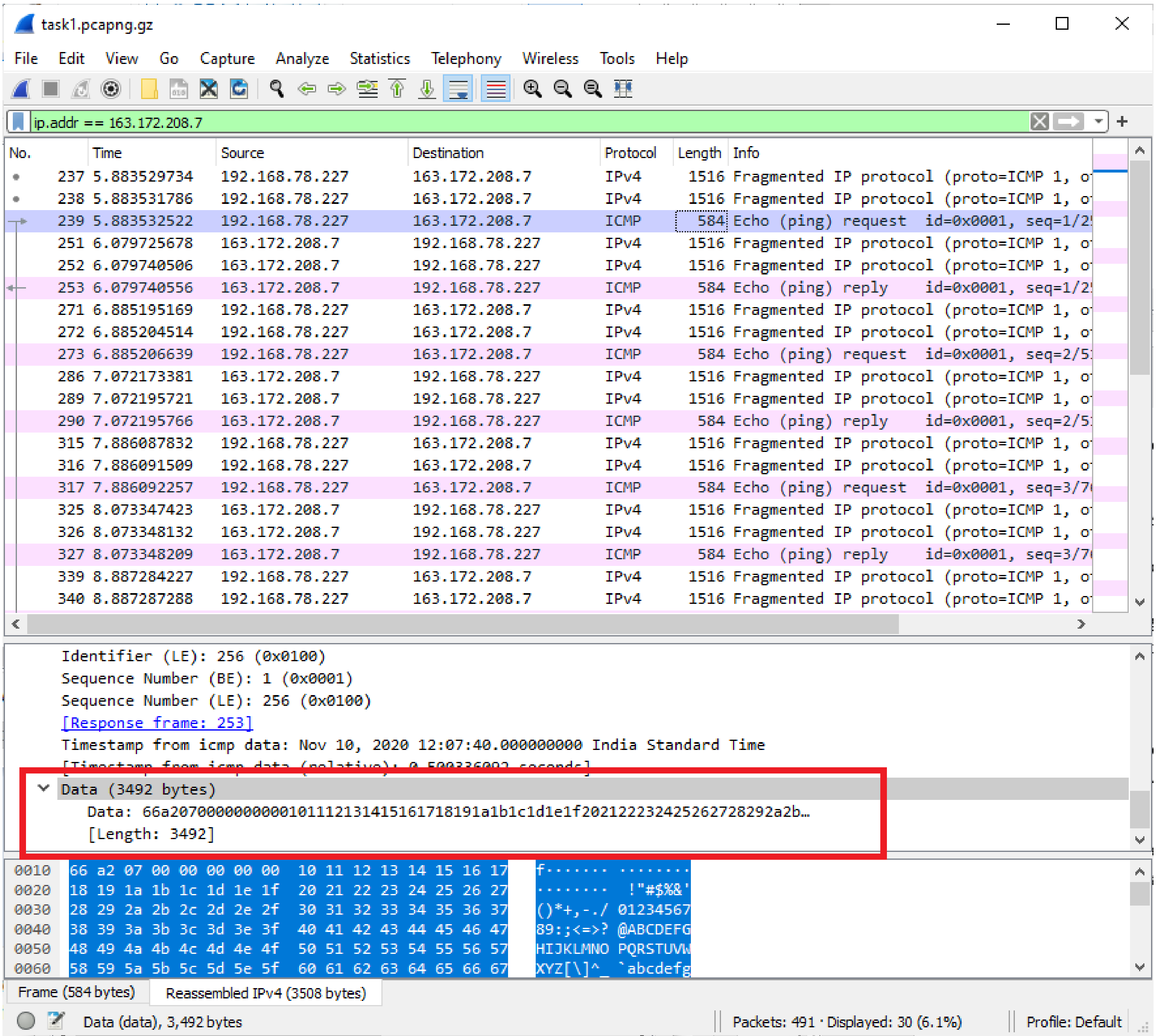
The same can also be verified by checking the packet count in conversations and endpoints.





3. What is the size of each ping request sent from your host to remote server ?

We check the data size in a request packet. We find the data size is 3492 bytes. ICMP header of 8 bytes is added over this data. So our final packet size is 3500 bytes which we defined in the ping command.



4. Make a table for each ping request packet sent from your host to remote, the respective field indicating it, if the request packet is fragmented or not. If packet is fragmented (add details on number of IP fragments and on each fragment), Time of sending each individual fragment/packet, length of the individual fragment/packet), time of receiving ping response, the respective field indicating if response packet is fragmented or not, if response packet is fragmented, include the number of IP fragments, total actual length of data carried by the respective fragment in respective ping request and response.

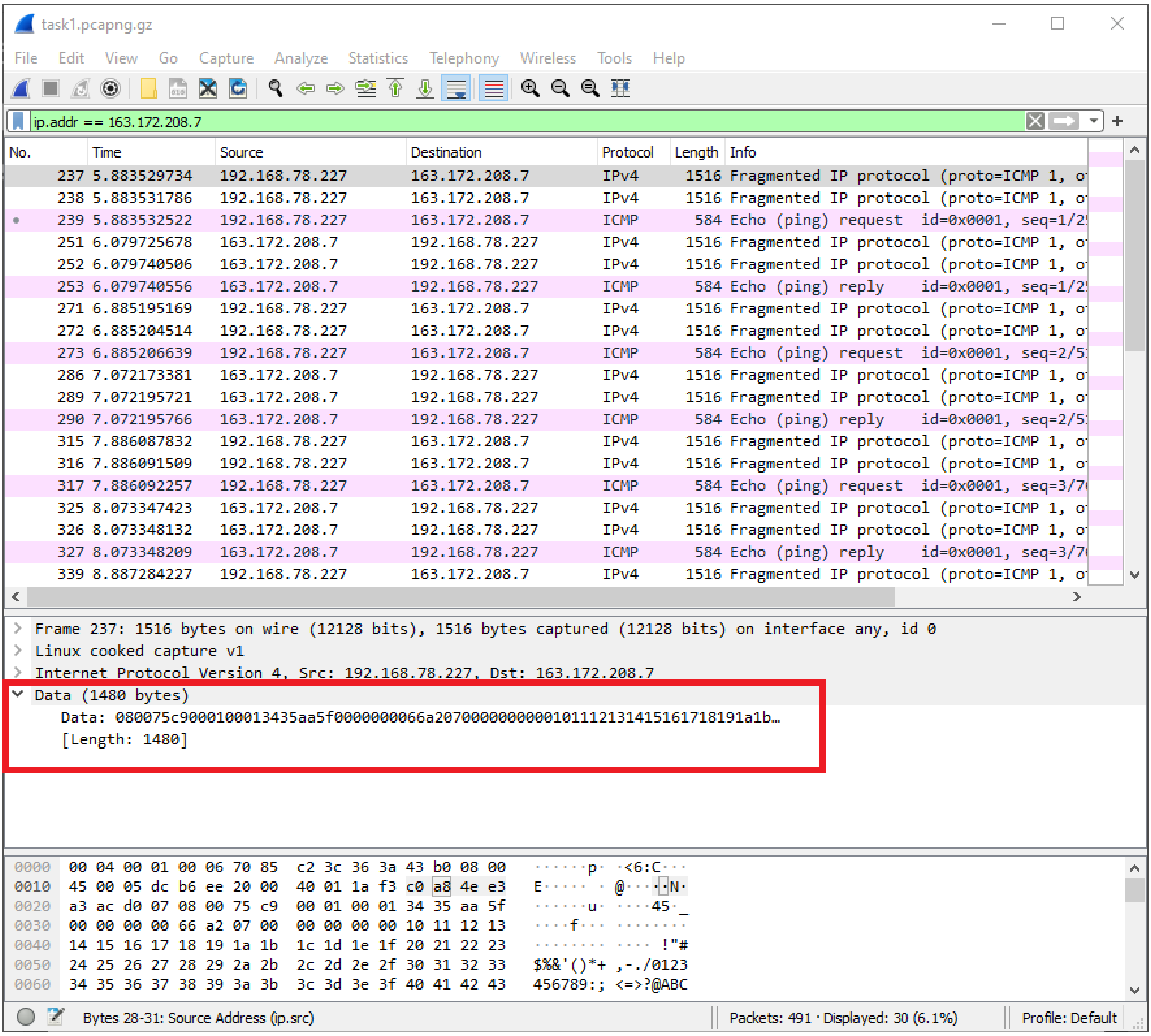
Request Packet	Field indicating request packet	Details of request packet fragments	Time of receiving ping response	Field indicating if the response is fragmented	Details of response packet fragments	Actual data carried by each fragment.
239	Internet Control	3 fragments	6.079740556	Internet Protocol	3 fragments	Packet 237: 1480 bytes

	Message Protocol , Type : 8 (Echo(ping) request)	Packet 237: Time – 5.883529734, Length – 1500 bytes Packet 238: Time – 5.883531786, Length – 1500 bytes Packet 239: Time – 5.883532522, Length – 568 bytes		Version 4, IPv4 Fragments	Packet 251, 252 and 253	Packet 238: 1480 bytes Packet 239: 548 bytes Packet 251: 1480 bytes Packet 252: 1480 bytes Packet 253: 548 bytes
273	Internet Control Message Protocol , Type : 8 (Echo(ping) request)	3 fragments Packet 271: Time – 6.885195169, Length – 1500 bytes Packet 272: Time – 6.885204514, Length – 1500 bytes Packet 273: Time – 6.885206639, Length – 568 bytes	7.072195766	Internet Protocol Version 4, IPv4 Fragments	3 fragments Packet 286, 289 and 290	Packet 271: 1480 bytes Packet 272: 1480 bytes Packet 273: 548 bytes Packet 286: 1480 bytes Packet 289: 1480 bytes Packet 290: 548 bytes
317	Internet Control Message Protocol , Type : 8 (Echo(ping) request)	3 fragments Packet 315: Time – 7.886087832, Length – 1500 bytes Packet 316: Time – 7.886091509, Length – 1500 Packet 317: Time – 7.886092257, Length – 568 bytes	8.073348209	Internet Protocol Version 4, IPv4 Fragments	3 fragments Packet 325, 326 and 327	Packet 315: 1480 bytes Packet 316: 1480 bytes Packet 317: 548 bytes Packet 325: 1480 bytes Packet 326: 1480 bytes Packet 327: 548 bytes

341	Internet Control Message Protocol , Type : 8 (Echo(ping) request)	3 fragments Packet 339: Time – 8.887284227, Length – 1500 bytes Packet 340: Time – 8.887287288, Length – 1500 bytes Packet 341: Time – 8.8872877781, Length – 568 bytes	9.066614160	Internet Protocol Version 4, IPv4 Fragments	3 fragments Packet 351, 352 and 353	Packet 339: 1480 bytes Packet 340: 1480 bytes Packet 341: 548 bytes Packet 351: 1480 bytes Packet 352: 1480 bytes Packet 353: 548 bytes
367	Internet Control Message Protocol , Type : 8 (Echo(ping) request)	3 fragments Packet 365: Time – 9.88323331, Length – 1500 bytes Packet 366: Time – 9.88326878, Length – 1500 bytes Packet 367: Time – 9.88327392, Length – 568 bytes	10.068313497	Internet Protocol Version 4, IPv4 Fragments	3 fragments Packet 375, 376 and 377	Packet 365: 1480 bytes Packet 366: 1480 bytes Packet 367: 548 bytes Packet 375: 1480 bytes Packet 376: 1480 bytes Packet 377: 548 bytes

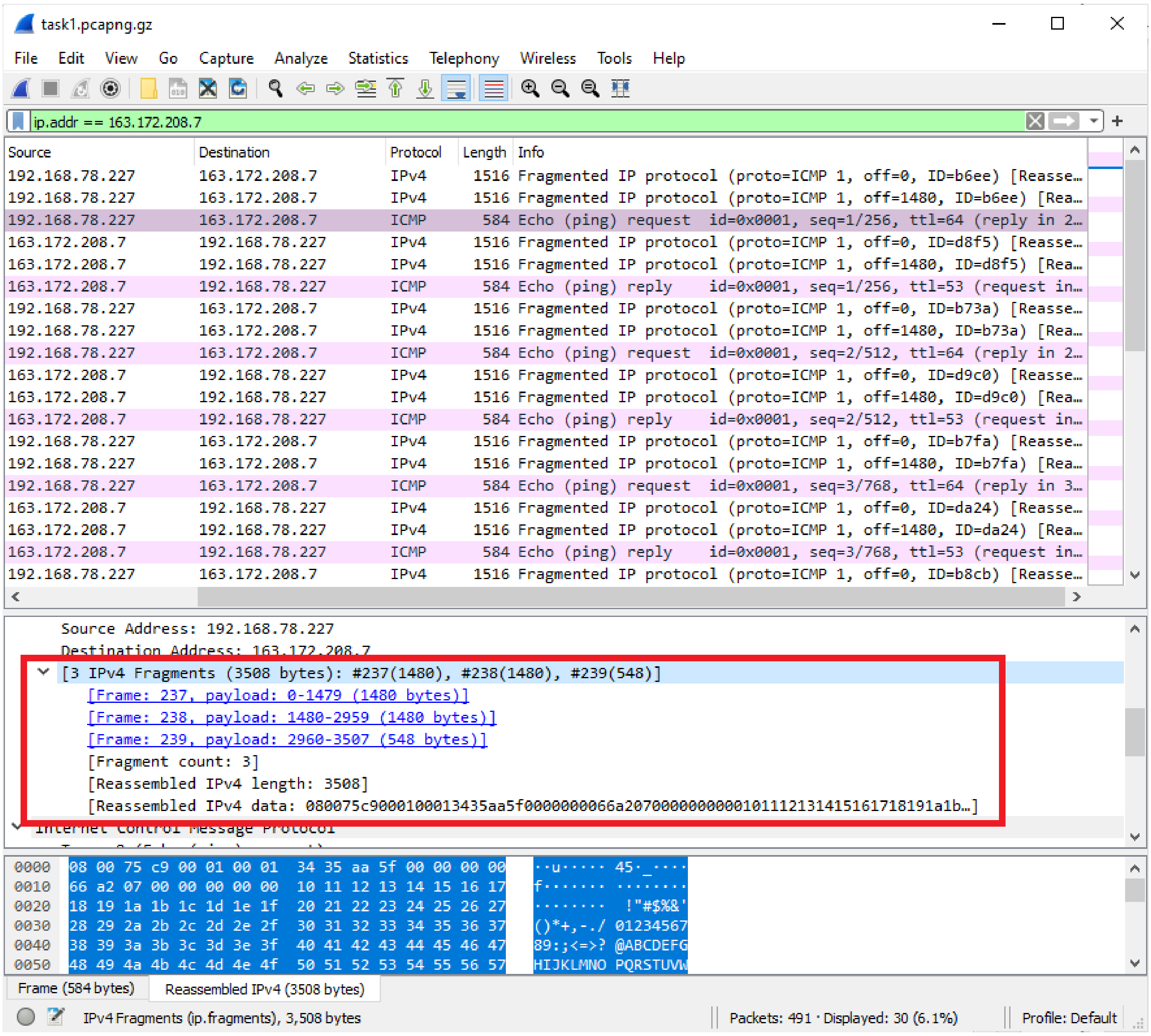
5. Pick any fragmented ping request and response used in question #4. Explain how you find the length of actual data in individual fragments of the associated ping request and response ? Where is the total/final length of the respective ping request and response at IP level visible in Wireshark ?

   We can check the Data field in Wireshark which shows the actual data after removing headers. For packet 237, we see the data is 1480 bytes.

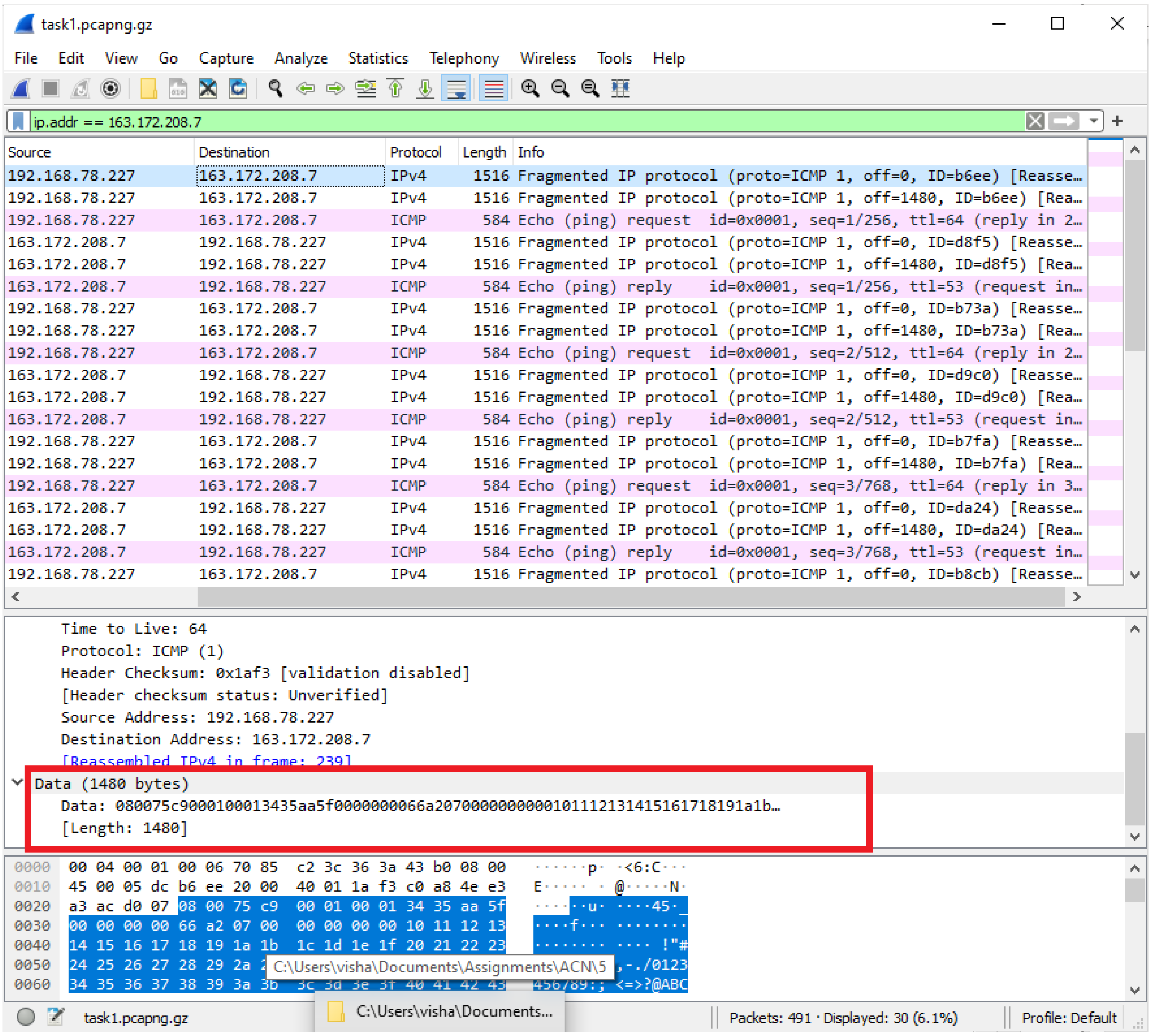


Similarly we can check for other fragments as well. We find that the last fragment shows the total of all fragments. To view the data in last fragment we check the IPv4 fragments field where it shows payload size of all fragments.

We can check IPv4 fragments under Internet Protocol version 4 for any ping request or response. It shows us the fragments and the data transmitted in each fragment.



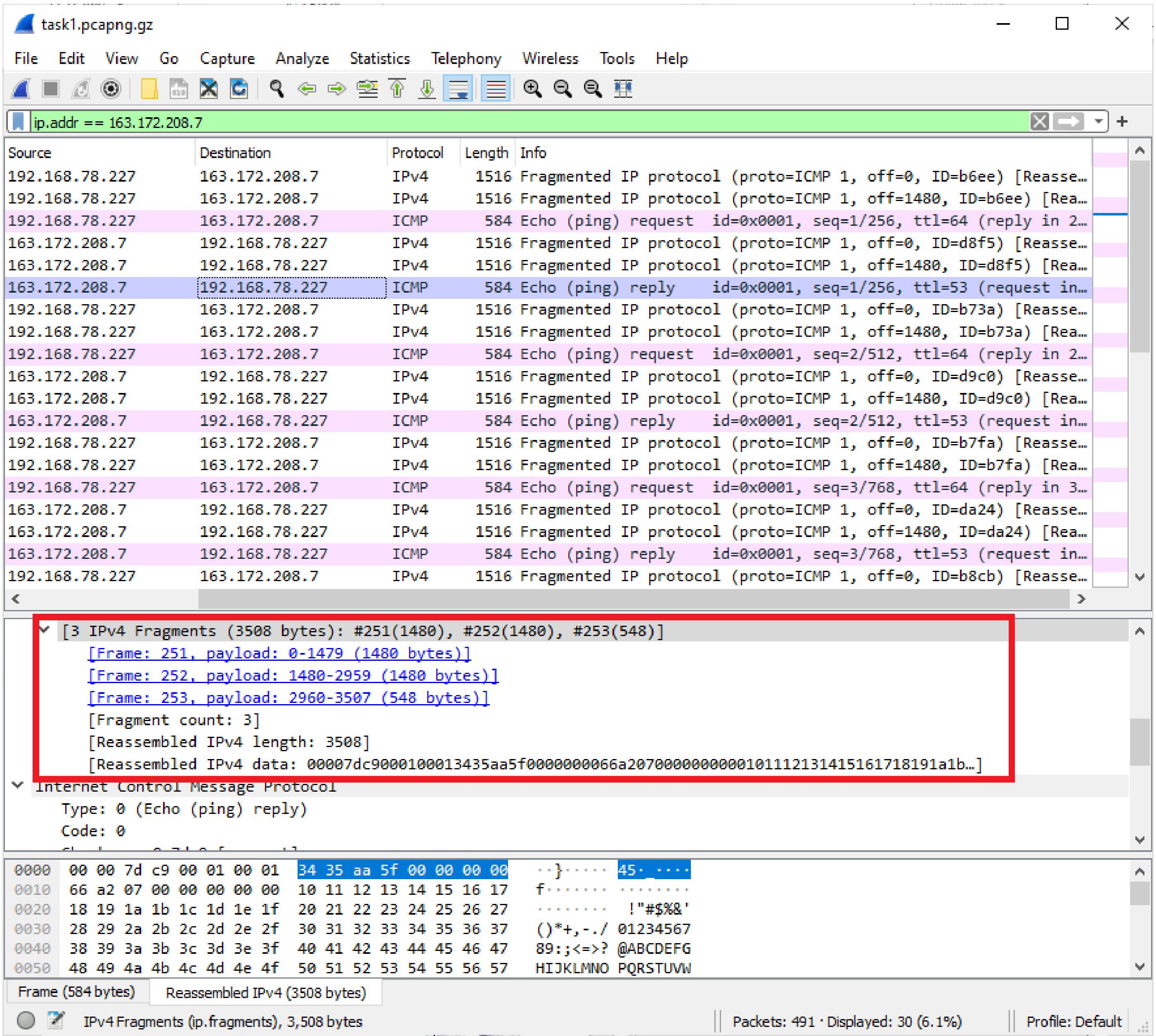
We can also check individual fragments and can verify the same data is transmitted in each fragment.



We can add the payload data in these fragments to get the final size of request packet sent. Adding the three fragments we get 1480 + 1480 + 548 = 3508 bytes.

The final size of request is 3508 bytes at the IP level which is also shown in Reassembled IPv4 length.

Similarly for reply packet,



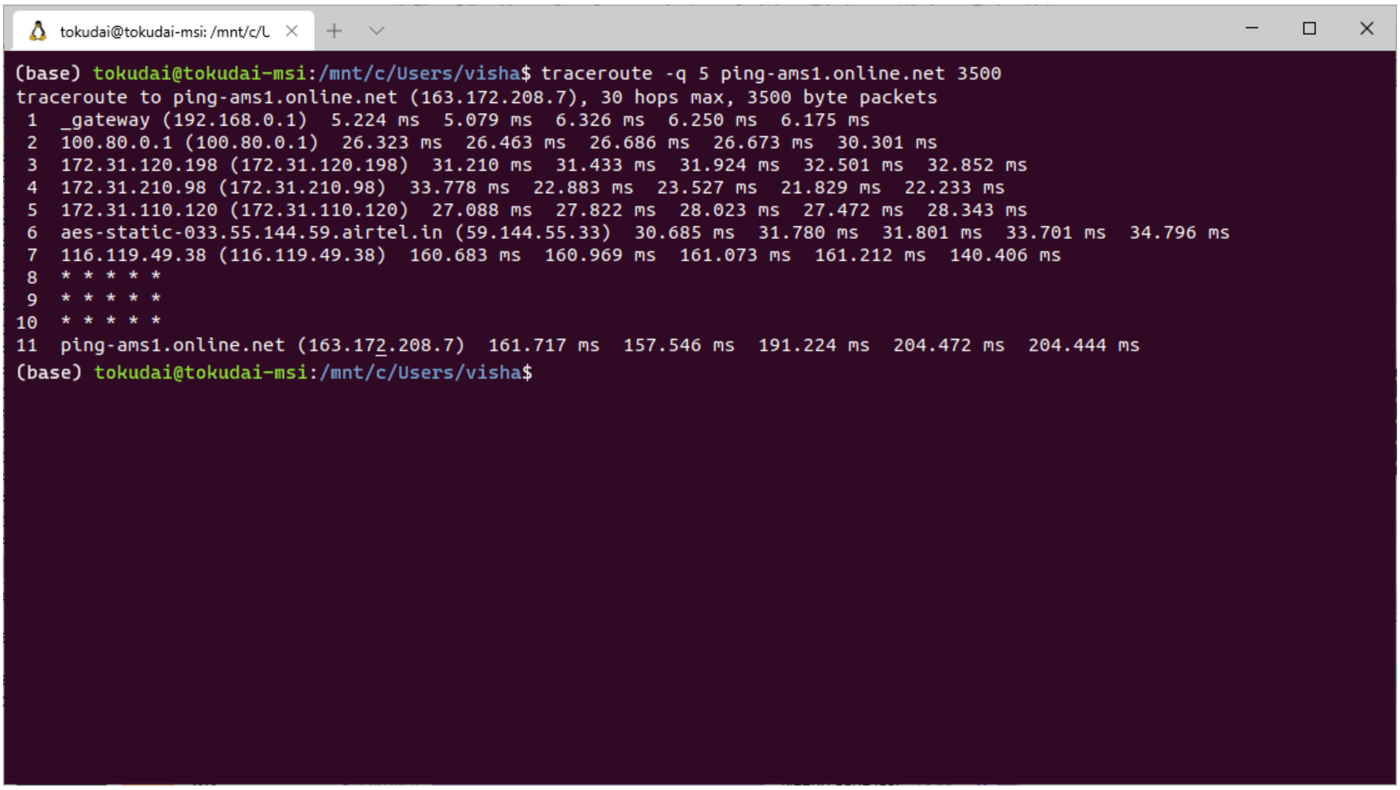
We can add the fragment sizes to get 1480 + 1480 + 548 bytes of data at IP level.

6. In the saved file used for analysis, export only those IP packets involved in either direction (in communication) using appropriate Wireshark display filters and save it to another file to upload during submission.

File saved as task1_filtered.pcapng.gz

Task 2

1. What is the IP address of **ping-ams1.online.net**?



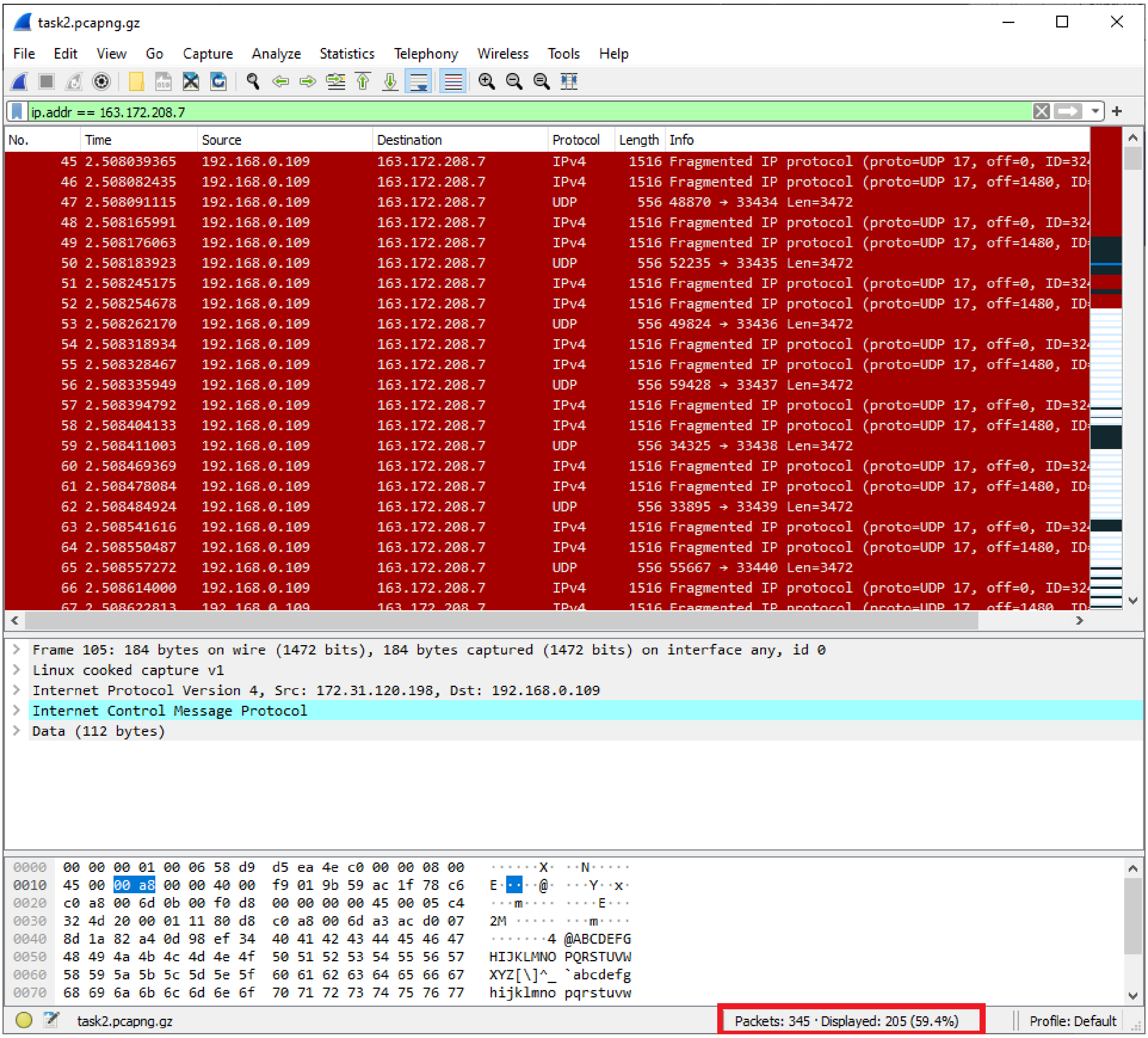
From the traceroute screenshot given above; we can see the IP address for ping-ams1.online.net is 163.172.208.7

2. How many hops are involved in finding the route to this **ping-ams1.online.net?**

There are 11 hops involved in finding the route to ping-ams1.online.net where 11^{th} hop is the destination we intend to reach.

3. How many total IP packets are exchanged in the communication to get the final traceroute output of **ping-ams1.online.net** ? How many of them are sent from client to remote machine (server/router) ? How many of them are sent from the remote machine (hop/server/router) to the local client ? Tabulate this with an entry for a router/server and the client too.

A total of 205 packets are exchanged to get the final output of traceroute. We do this by searching all the packets send to / received from ping-ams1.online.net using *ip.addr == 162.172.208.7* filter. This is shown in the screenshot given below.



4. Why and how does the hop/router involved send the response to the packet sent by your client machine ?

   The IP protocol has a TTL (time to live) field in header of IPv4 packet which describes how many hops the packet can take before being discarded. When a packet reaches end of TTL, it is dropped and a response is sent from the dropping router/hop to the client which sent the packet.

   For traceroute, we increase the TTL by 1 for every next hop. When the packet is dropped the, the router will send a response which will contains its IP address in source field. This way we can know the hops through which the packet passes.

5. Which upper layer protocol is used in sending the packet from local client to remote machine ? Which upper layer protocol is used in sending the packet from remote server/router to local client ? Identify within the protocol the type/name of this message (within the protocol) from server to client?

   Within the header, the value in the upper layer protocol field from client to server is IP,UDP,Payload. The value in the upper layer protocol field from server to client is IP,ICMP,Payload.

6. Which fields in the IP datagram always change from one datagram to the next within this series of IP packets sent by your host/client ? Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?

   The fields that always change in IP datagram are:

   Identification – All IP packets must have a different ID.

   Time to live – Traceroute increments the TTL for each subsequent packet to find the next hop.

   Header checksum – Since the header changes for every packet, so the checksum needs to change for each packet.

   The fields that change for every packet are:

   Version – Since we are using IPv4 for for all packets, the version should remain same for all packets.

   Header length – Since all the packets are ICMP packets, the header length always remains constant.

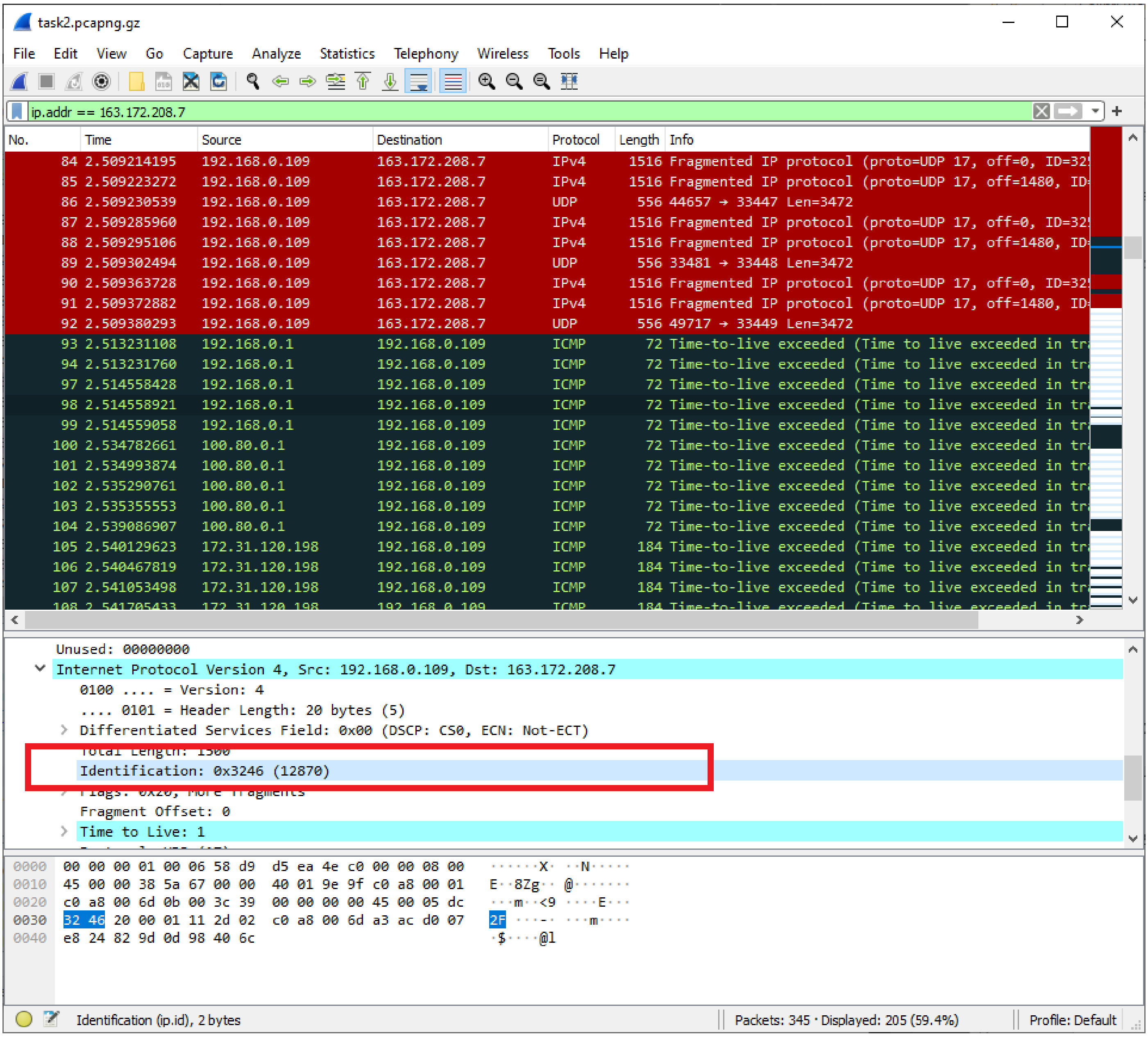
   Source IP – Since all packets are being sent from our client, the source IP will remain same for all packets sent from our system.

   Destination IP – Since all packets are being sent to final destination we need to reach, albeit with different TTL, the destination IP address should remain same for all packets.

   Upper Layer Protocol – Since the entire conversation uses ICMP packets, the upper layer protocol should remain same for all packets.

7. Describe the pattern you see in the values in the Identification field of the IP datagram both from client to server and hop/router/server to your client?

   We observe that the Identification flag under Internet Protocol version 4 under Internet control message protocol is incremented by 1 for every response. When the response comes from a new hop, the value may increase by more than 1.



8. Make a table with an entry for each request sent from your client/host, listing what is the value in the IP Identification field and the TTL field for the request and respective response (Include entries if the packet is fragmented). Do these values remain unchanged for all of the replies sent to your computer by the respective (hop) router? If yes why? If not why ?

   A few values are sown below as it was not possible to show all values due to the large number of entries that will be made.

Request Identification	Request TTL	Response Identification	Response TTL
12867	1	23140	64
12874	2	0	254
12880	3	0	249
12898	4	59409	251

We observe that for a specific hop, the identification number keeps incrementing by 1 for every new request and response whereas the TTL remains the same for request. The TTL remains same as we need to get response from the same hop every time.

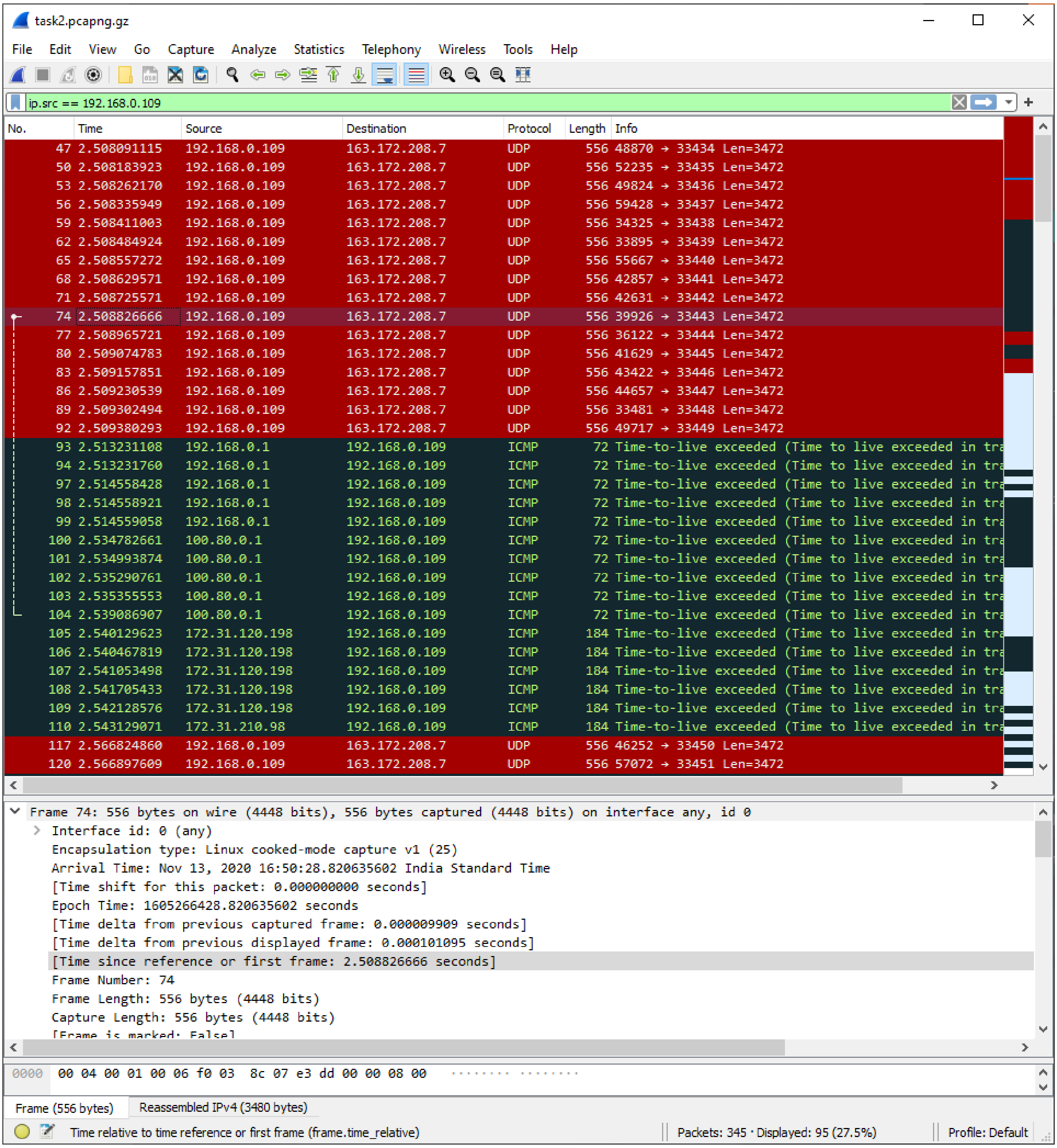
For response from a single hop, the identification keeps increasing by 1 whereas the TTL remains the same for all packets. The identification keeps changing so that the other side knows that it is a new packet and not the same packet transmitted again due to congestion of loss of packet.

9. Calculate the average RTT for each request sent by traceroute w.r.t its respective response (from the related hop) using the different IP fields and wireshark display filters. There shall be a proof of screenshot(s) showing this calculation for at least 1 packet with respective response(s). Plot a graph of hop name/IP address which sent the response versus the RTT using the calculations done.

We use filter *ip.addr == 192.168.0.109 && ip.addr == 163.172.208.7 && (udp || icmp)* to filter out the packets that are sent to server and the ICMP response received from different hosts.

Hop	Send time for each packet	Response time for each packet	RTT for each packet	Average RTT
192.168.0.1	2.508091115, 2.508183923, 2.508262170, 2.5083335949, 2.508411003	2.513231108, 2.513231760, 2.514558428, 2.514558921, 2.514559058	0.005139993, 0.005047837, 0.006296258, 0.0062253261, 0.006148055	0.00577149382
100.80.0.1	2.508484924, 2.508557272, 2.508629571, 2.508725571, 2.508826666	2.534782661, 2.534993874, 2.535290761, 2.535355553, 2.539086907	0.026297737, 0.026436602, 0.02666119, 0.026629982, 0.030260241	0.0272571504
172.31.120.198	2.508965721, 2.509074783, 2.509157851, 2.509230539, 2.509302494	2.540129623, 2.540467819, 2.541053498, 2.541705433, 2.542128576	0.031163902, 0.031393036, 0.031895647, 0.032474894, 0.032826082	0.0319507122
172.31.210.98	2.509380293, 2.566824860, 2.566897609, 3.023102009, 3.023180429	2.543129071, 2.589636522, 2.590400886, 3.044864773, 3.045387225	0.033748778, 0.022811662, 0.023503277, 0.021762764, 0.022206796	0.0248066554
172.31.110.120	3.023252596, 3.023321026, 3.023393692, 3.023464267, 3.023534957	3.050315340, 3.050909835, 3.051120016, 3.051391002, 3.051850891	0.027062744, 0.027588809, 0.027726324, 0.027926735, 0.028315934	0.0277241092

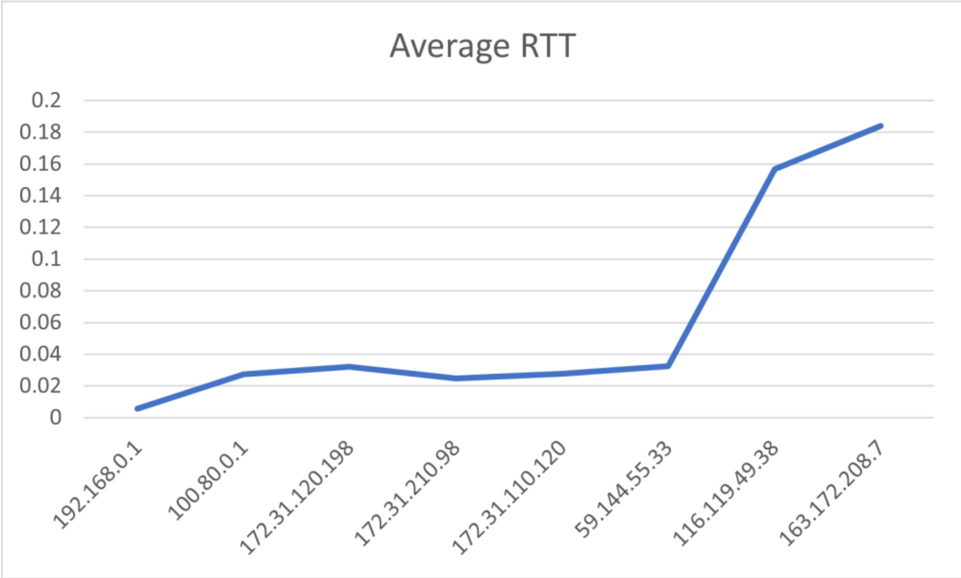
59.144.55.33	3.023604144, 3.023677099, 3.023745330, 3.023814871, 3.023886436	3.054263926, 3.055429171, 3.055521678, 3.057491170, 3.058656346	0.030659782, 0.031752072, 0.031776348, 0.033676299, 0.03476991	0.0325268822
116.119.49.38	3.023958083, 3.024028478, 3.024205662, 3.024277839, 3.045294679	3.184613414, 3.184971264, 3.185246120, 3.185463351, 3.185583429	0.160655331, 0.160942786, 0.161040458, 0.161185512, 0.14028875	0.1568225674
163.172.208.7	3.278266581, 3.440277891, 3.598109224, 3.789742012, 3.994534308	3.439953895, 3.597734655, 3.789255632, 3.994101766, 4.198802957	0.161687314, 0.157456764, 0.191146408, 0.204359754, 0.204268649	0.1837837778



We see that for packet 74, we get a response at packet 104 (shown by dotted line in Wireshark). We see the time difference is 2.539086907 – 2.508826666 = 0.

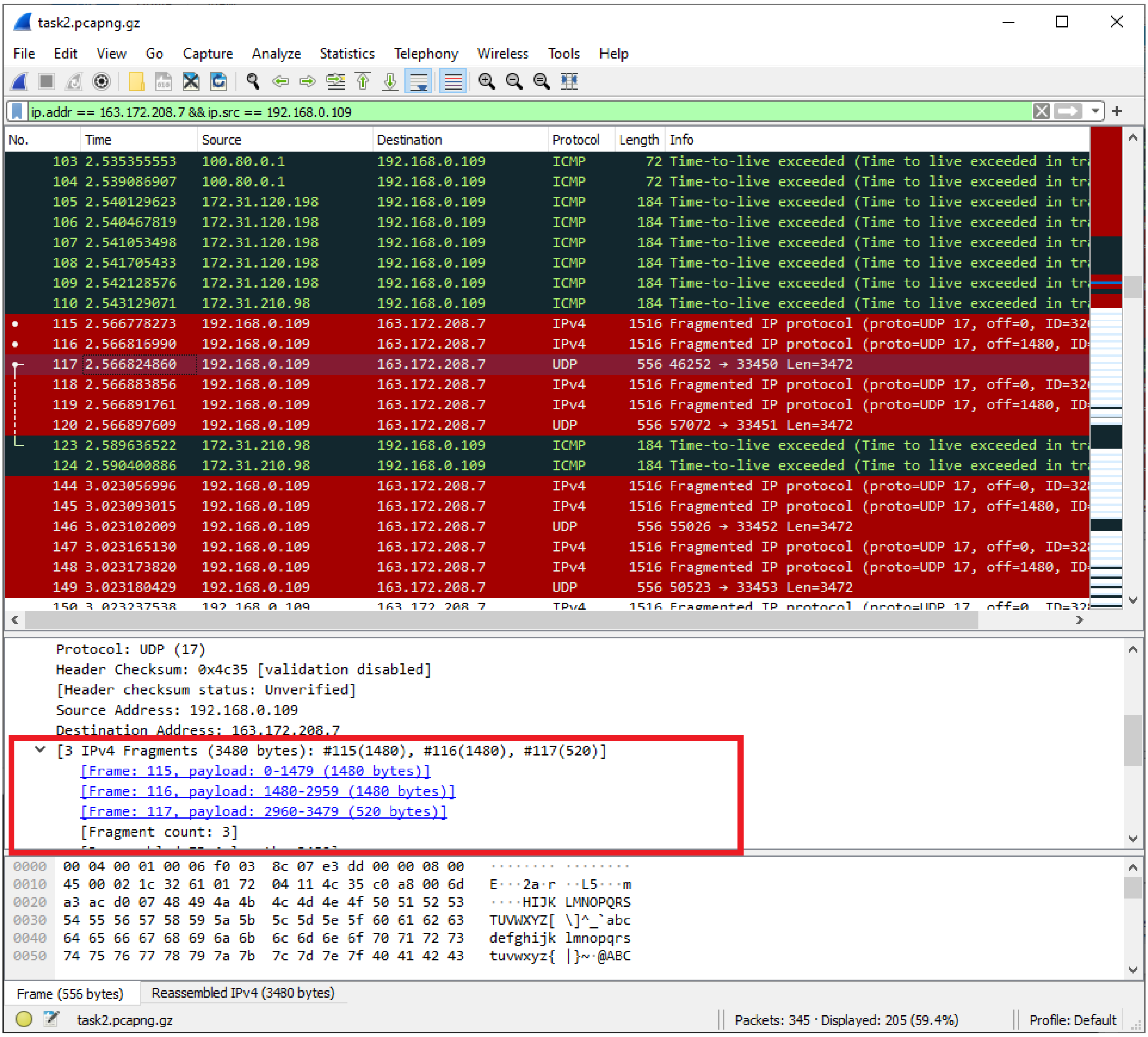
Similarly we can find the RTT for each packet for each hop and calculate the average RTT.

The graph for Hop vs Average RTT is given below:

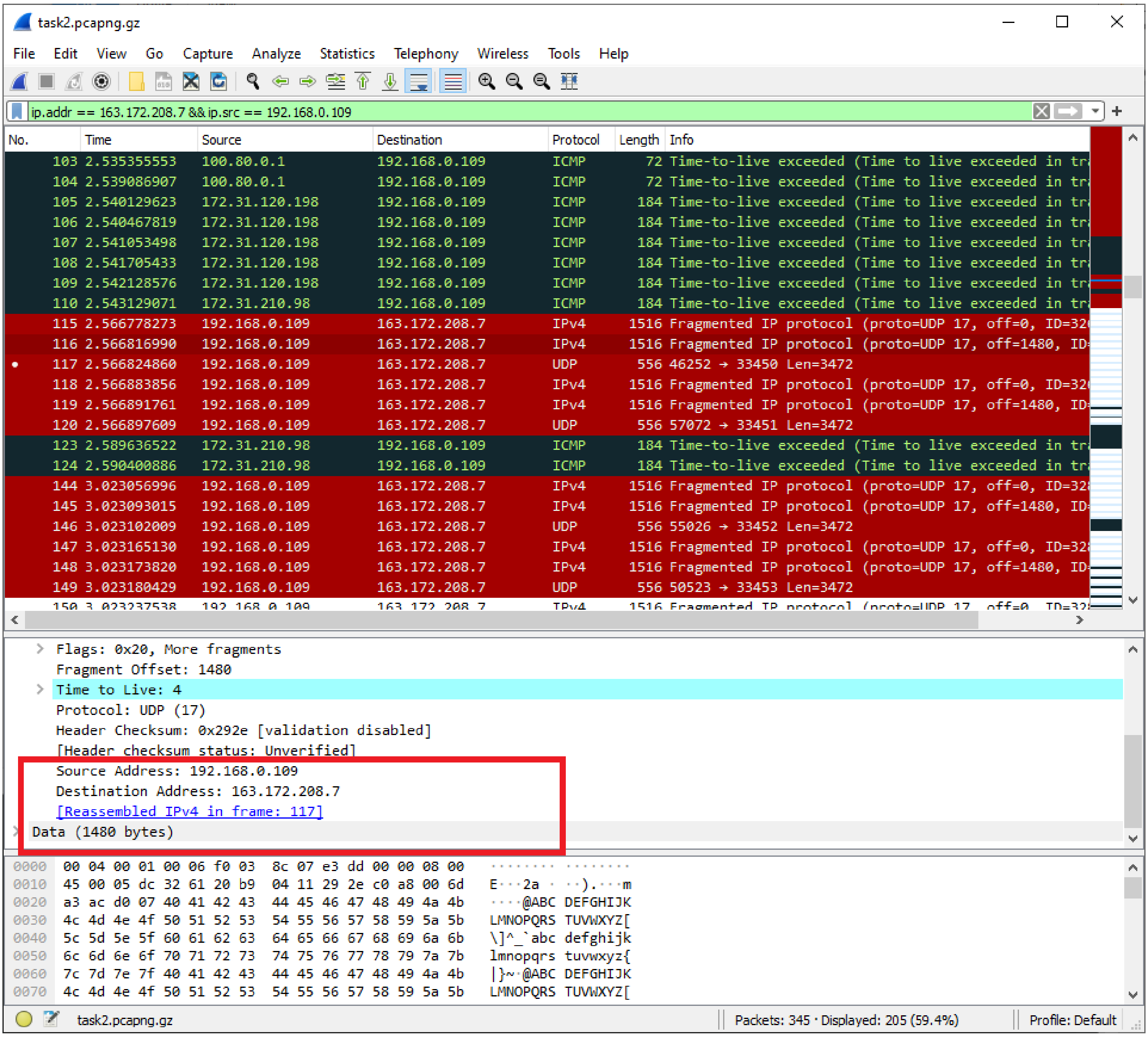


10. Pick any packet from client towards server. Has this IP datagram been fragmented?

    We pick packet 117 from the capture which is from our host to server. We can check the Internet Protocol version 4 filed in Wireshark where it shows the number of fragments of the packet, their lengths as well as the fragments which are combined to form a single packet.



If the packet is not a end of fragment packet, such as packet 116, wireshark shows the frame in which the fragment is reassembled to form a packet.



11. Explain how you determine whether or not the datagram has been fragmented and where does the fragmentation end.

    As explained in previous answer, we can check Internet Protocol version 4 filed in Wireshark to see if the datagram is fragmented. For a datagram that is not ending datagram, we can see the datagram number where the datagrams are reassembled to form a single packet.

    To check where which frame is the last one in fragments, we can check MF flag. When More Fragments flag is 0 we know that there are no more fragments left.

12. In the saved file used for analysis, export only those IP packets involved in either direction (in communication) using appropriate Wireshark display filters and save it to another file to upload during submission.

    File is uploaded as task2_filtered.pcapng.gz

Task 3

1. Comment on your understanding of differences between traceroute and ping command executed for **ping-ams1.online.net** (in **Task1 and Task2**) using the respective wireshark traces. List the IP fields which indicate the differences in the behaviour of execution. Place at least 1 screenshot of wireshark from respective traces showing relevant display filters as a proof of explaining the difference.

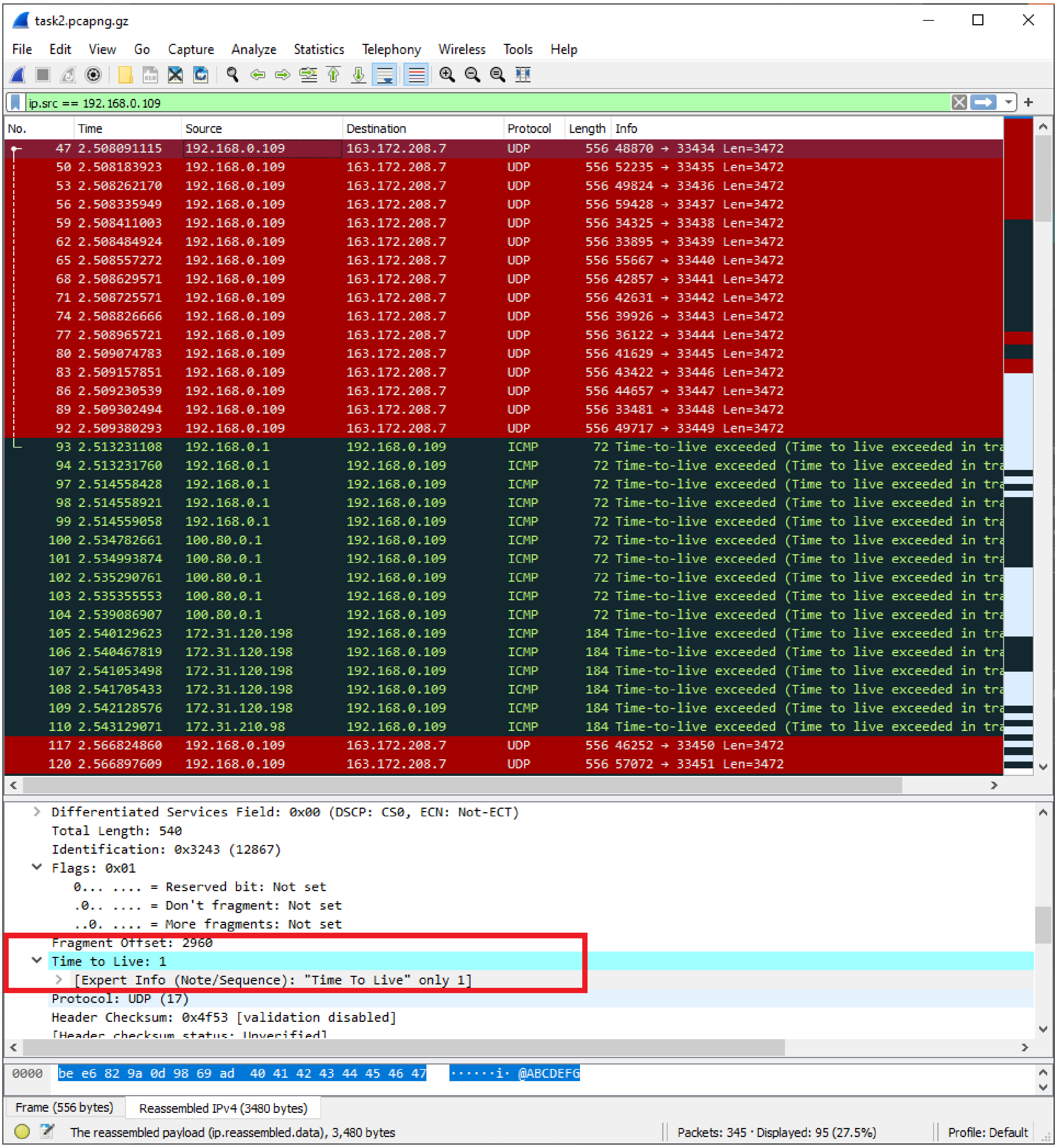
   The following observations were made from the ping and traceroute.

   Ping command works over the IP layer. This command work on ECHO-Request and ECHO-Reply. This command is used to find whether a host is available or not. For this client send the ECHO request using ICMP packet and the destination host will reply with an ECHO packet if it is available.

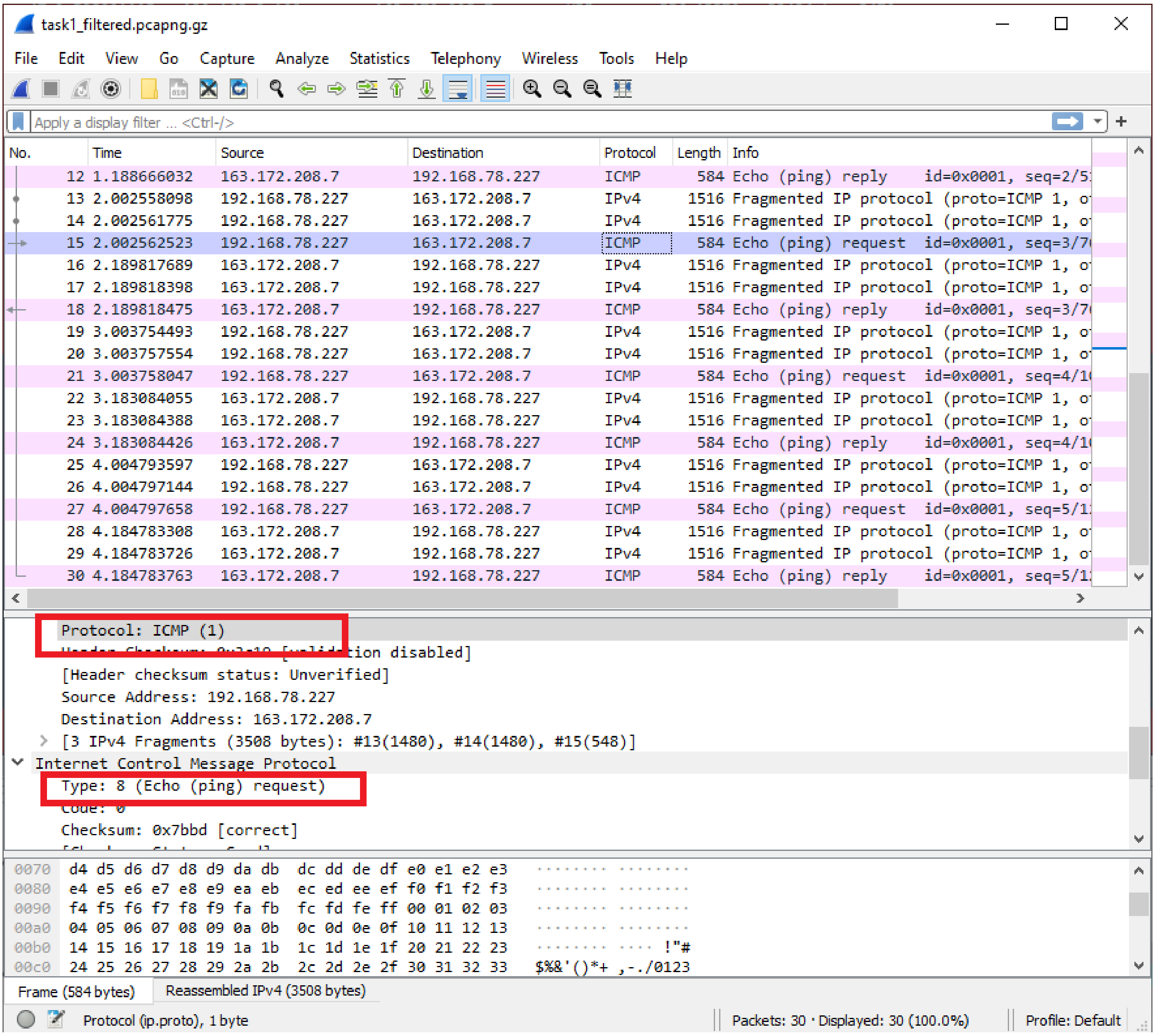
   Traceroute is used to find the path and number of hops in between source and destination. This protocol uses ICMP, IPV4 and UDP protocol for this mechanism. This process uses various messages like TTL exceeded, port unreachable etc to find the hosts that act as hop between the client and server.

   The IP fields which indicate the difference are:

   For traceroute, we use TTL filed to set the number of hops. By setting the number of hops to 1 and increasing it by 1 for every next hop, we can find all the hops from which the packet passes. This is achieved as when the TTL gets to 0, the hop which drops the packet will send a ICMP response saying TTL exceeded back to the host. We can check the IP address from where the response came and find the hops.



For Ping requests, we set the Protocol in IPv4 to 1 and Type in ICMP to 8 for ping request or 0 for ping reply.



I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.