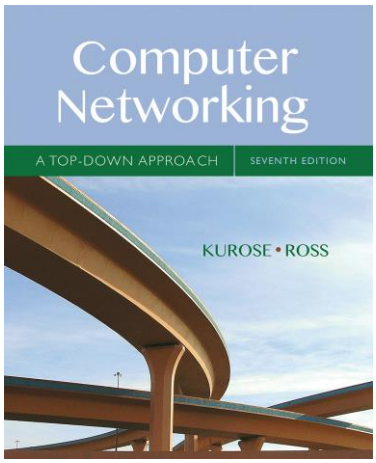
Name: _____Timur Guner_____

Wireshark Lab: IP v7.0

Supplement to *Computer Networking: A Top-Down Approach, 7th ed.*, J.F. Kurose and K.W. Ross



"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb

In this lab, we'll investigate the IP protocol, focusing on the IP datagram. We'll do so by analyzing a trace of IP datagrams sent and received by an execution of the `traceroute` program (the `traceroute` program itself is explored in more detail in the Wireshark ICMP lab). We'll investigate the various fields in the IP datagram, and study IP fragmentation in detail.

Before beginning this lab, you'll probably want to review sections 1.4.3 in the text[1] and section 3.4 of RFC 2151 [ftp://ftp.rfc-editor.org/in-notes/rfc2151.txt] to update yourself on the operation of the `traceroute` program. You'll also want to read Section 4.3 in the text, and probably also have RFC 791 [ftp://ftp.rfc-editor.org/in-notes/rfc791.txt] on hand as well, for a discussion of the IP protocol.

1. Capturing packets from an execution of traceroute

In order to generate a trace of IP datagrams for this lab, we'll use the `traceroute` program to send datagrams of different sizes towards some destination, *X*. Recall that `traceroute` operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1 (actually, RFC 791 says that the router must decrement the TTL by *at least* one). If the TTL reaches 0, the router returns an ICMP message (type 11 – TTL-exceeded) to the sending host. As a result of this behavior, a datagram with a TTL of 1 (sent by the host executing `traceroute`) will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops

[1] References to figures and sections are for the 7th edition of our text, *Computer Networks, A Top-down Approach, 7th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2016.*

away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing `traceroute` can learn the identities of the routers between itself and destination *X* by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

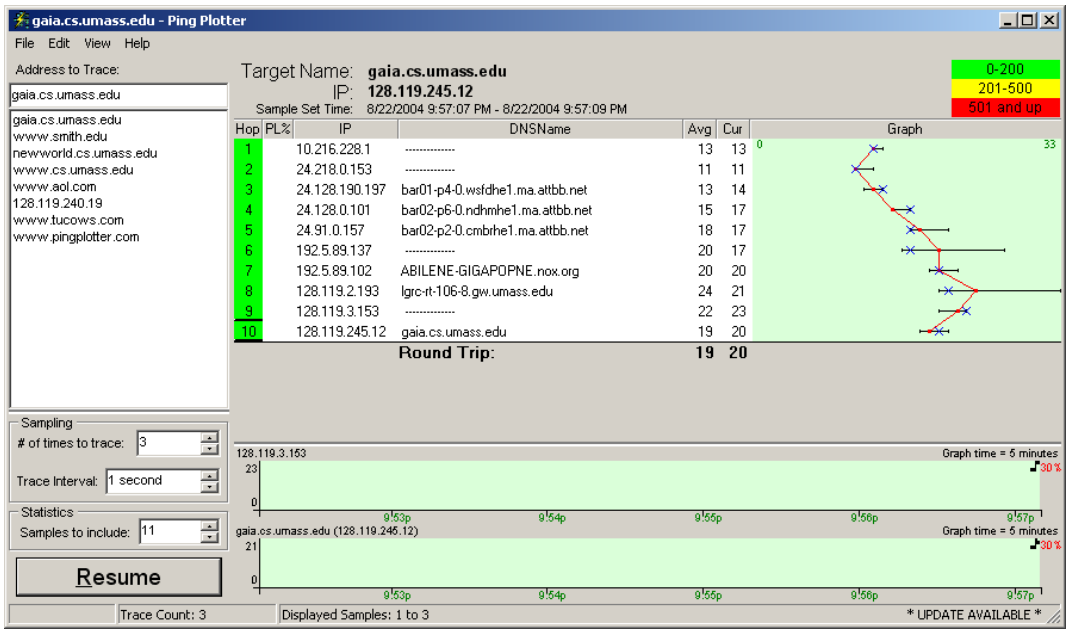
We'll want to run `traceroute` and have it send datagrams of various lengths.

- **Windows.** The `tracert` program (used for our ICMP Wireshark lab) provided with Windows does not allow one to change the size of the ICMP echo request (ping) message sent by the `tracert` program. A nicer Windows `traceroute` program is *pingplotter*, available both in free version and shareware versions at http://www.pingplotter.com. Download and install *pingplotter*, and test it out by performing a few traceroutes to your favorite sites. The size of the ICMP echo request message can be explicitly set in *pingplotter* by selecting the menu item *Edit-> Options->Packet Options* and then filling in the *Packet Size* field. The default packet size is 56 bytes. Once *pingplotter* has sent a series of packets with the increasing TTL values, it restarts the sending process again with a TTL of 1, after waiting *Trace Interval* amount of time. The value of *Trace Interval* and the number of intervals can be explicitly set in *pingplotter*.
- **Linux/Unix/MacOS.** With the Unix/MacOS `traceroute` command, the size of the UDP datagram sent towards the destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the `traceroute` command line immediately after the name or address of the destination. For example, to send `traceroute` datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

```
%traceroute gaia.cs.umass.edu 2000
```

Do the following:

- Start up Wireshark and begin packet capture *(Capture->Start)* and then press *OK* on the Wireshark Packet Capture Options screen (we'll not need to select any options here).
- If you are using a Windows platform, start up *pingplotter* and enter the name of a target destination in the "Address to Trace Window." Enter 3 in the "# of times to Trace" field, so you don't gather too much data. Select the menu item *Edit->Advanced Options->Packet Options* and enter a value of 56 in the *Packet Size* field and then press OK. Then press the Trace button. You should see a *pingplotter* window that looks something like this:



Next, send a set of datagrams with a longer length, by selecting *Edit->Advanced Options->Packet Options* and enter a value of 2000 in the *Packet Size* field and then press OK. Then press the Resume button.

Finally, send a set of datagrams with a longer length, by selecting *Edit->Advanced Options->Packet Options* and enter a value of 3500 in the *Packet Size* field and then press OK. Then press the Resume button.

Stop Wireshark tracing.

- If you are using a Unix or Mac platform, enter three `traceroute` commands, one with a length of 56 bytes, one with a length of 2000 bytes, and one with a length of 3500 bytes.

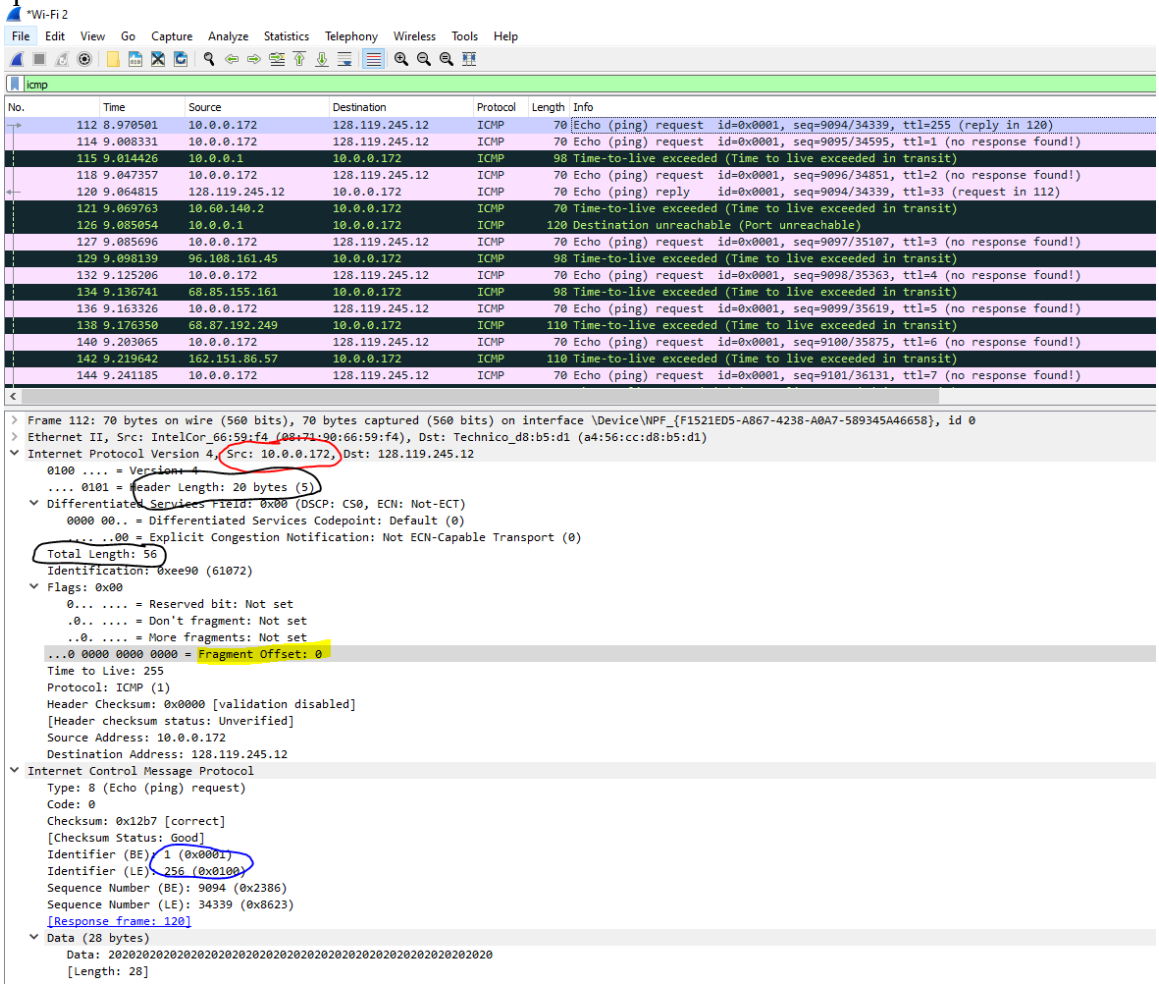
  Stop Wireshark tracing.

If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author's Windows computers[2]. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

[2] Download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file *ip-ethereal-trace-1*. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the ip-ethereal-trace-1 trace file.

2. A look at the captured trace

In your trace, you should be able to see the series of ICMP Echo Request (in the case of Windows machine) or the UDP segment (in the case of Unix) sent by your computer and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers. In the questions below, we'll assume you are using a Windows machine; the corresponding questions for the case of a Unix machine should be clear. Whenever possible, when answering a question below you should include a screenshot of the packet(s) within the trace that you used to answer the question asked. When you submit your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our class, we ask that students annotate electronic copies with text in a colored font). Make sure to include in your screenshot ALL and ONLY the minimum amount of packet detail that you need to answer the question.



1. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window.
   What is the IP address of your computer?
   Circled in red is the IP address is 10.0.0.172
2. Within the IP packet header, what is the value in the upper layer protocol field?
   Circled in blue is the value is 0x0001

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.
   The header is 20 bytes. Total length is 56 so the payload is 56-20=36 bytes. Check the values circled in black
4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.
   The IP has not been fragmented because fragment offset is 0 and the more fragments bit is 0. Highlight shows the more fragment offset and right above it the more fragments set to 0 as well.

Next, sort the traced packets according to IP source address by clicking on the *Source* column header; a small downward pointing arrow should appear next to the word *Source*. If the arrow points up, click on the *Source* column header again. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol portion in the "details of selected packet header" window. In the "listing of captured packets" window, you should see all of the subsequent ICMP messages (perhaps with additional interspersed packets sent by other protocols running on your computer) below this first ICMP. Use the down arrow to move through the ICMP messages sent by your computer.

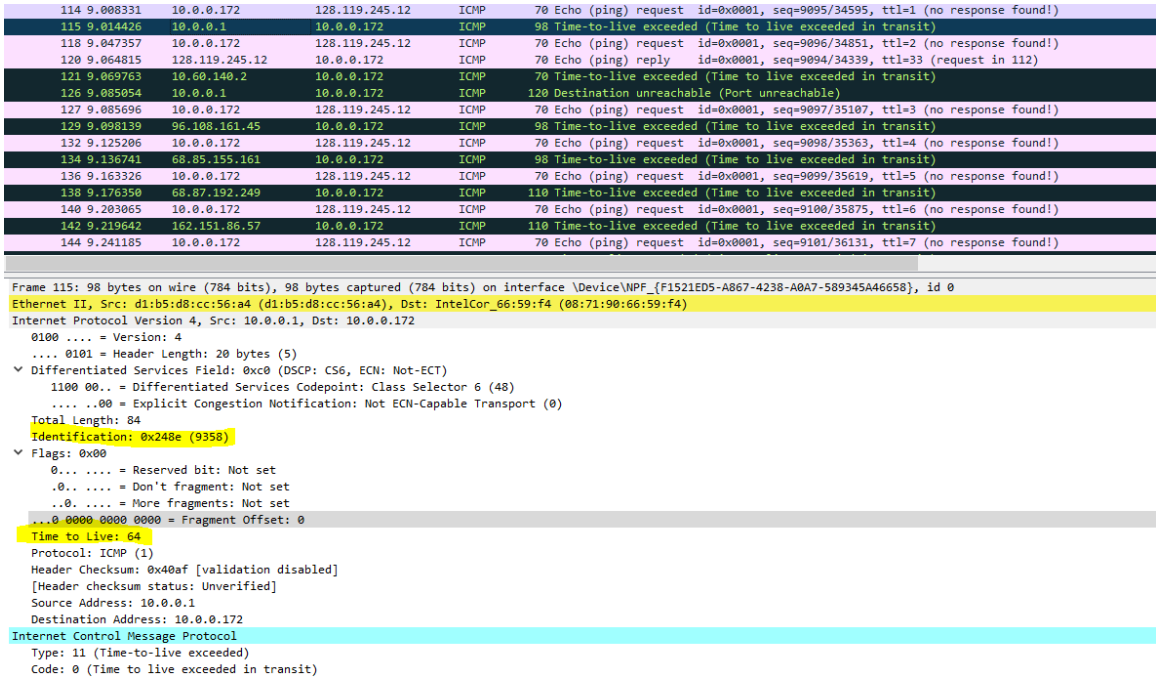
5. Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by your computer?
   Identification, checksum, and time to live
6. Which fields stay constant? Which of the fields *must* stay constant? Which fields must change? Why?
   The fields that stay constant across the IP datagrams are:
   • Version
   • Header length
   • Source IP
   • Destination IP
   • Differentiated Services
   • Upper Layer Protocol (since these are ICMP packets)

   The fields that must stay constant are:
   • Same as above

   The fields that must change are:
   • Identification because the packet ids need to be different
   • Time to live increments with each packet
   • Header checksum since the header changes, the checksum also changes
7. Describe the pattern you see in the values in the Identification field of the IP datagram
   The identification field is increasing by one each time. First request started at 0xee91.

Next (with the packets still sorted by source address) find the series of ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router.

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



Identification number is 0x248e and the TTL is 64

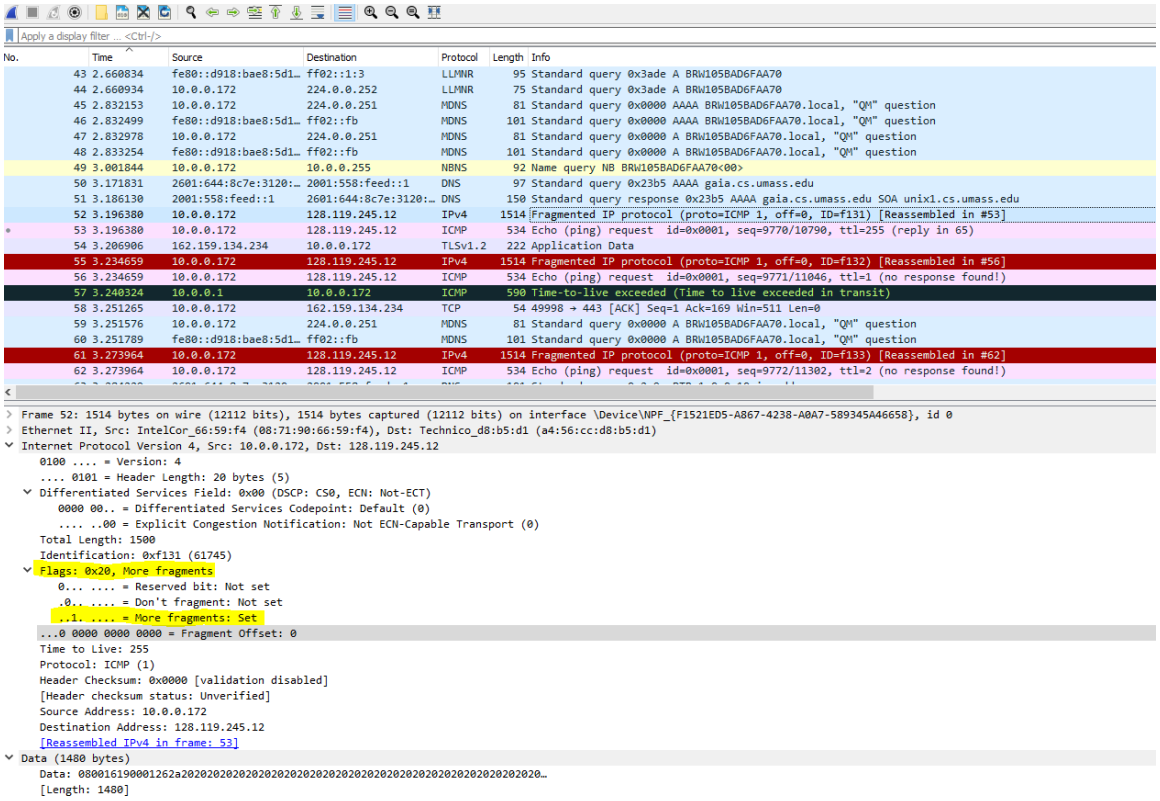
9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?
The identification field will change for all replies because it is a unique value. When two or more datagrams have the same id, it means it is fragmented.

The TTL does not change because it is the same for all first hops

Fragmentation

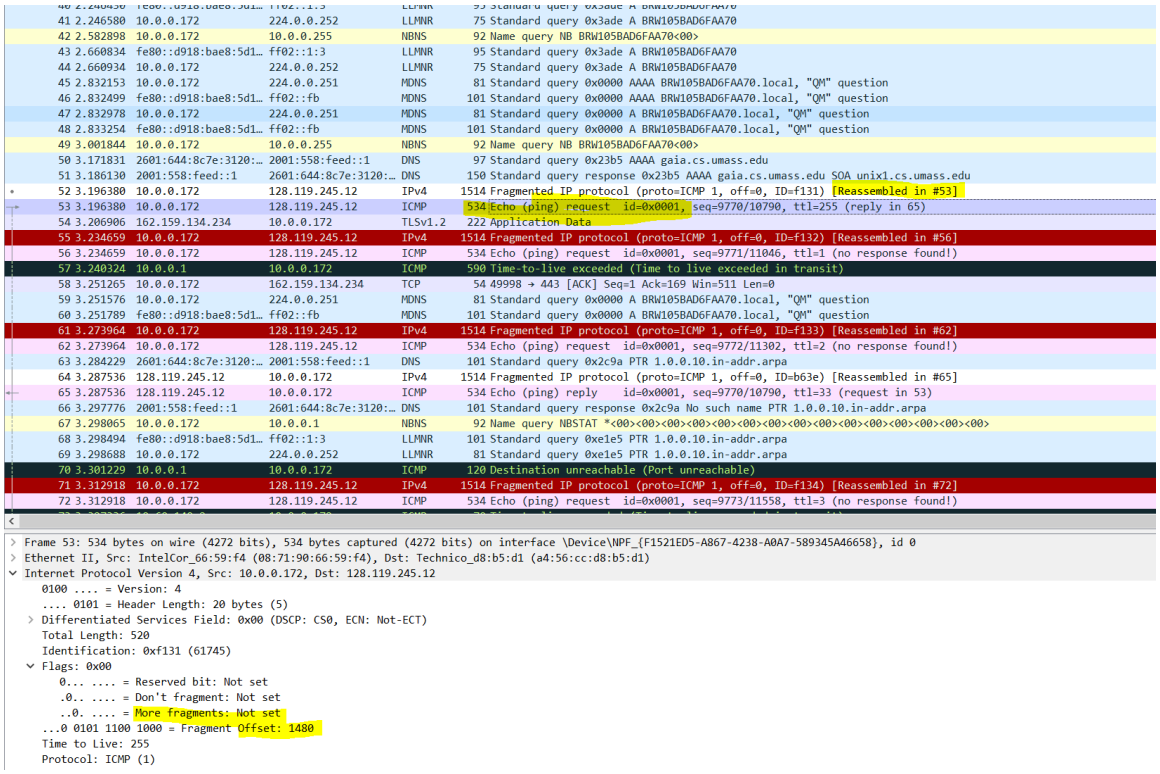
Sort the packet listing according to time again by clicking on the *Time* column.

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?



More fragments bit has been set, so we have fragments

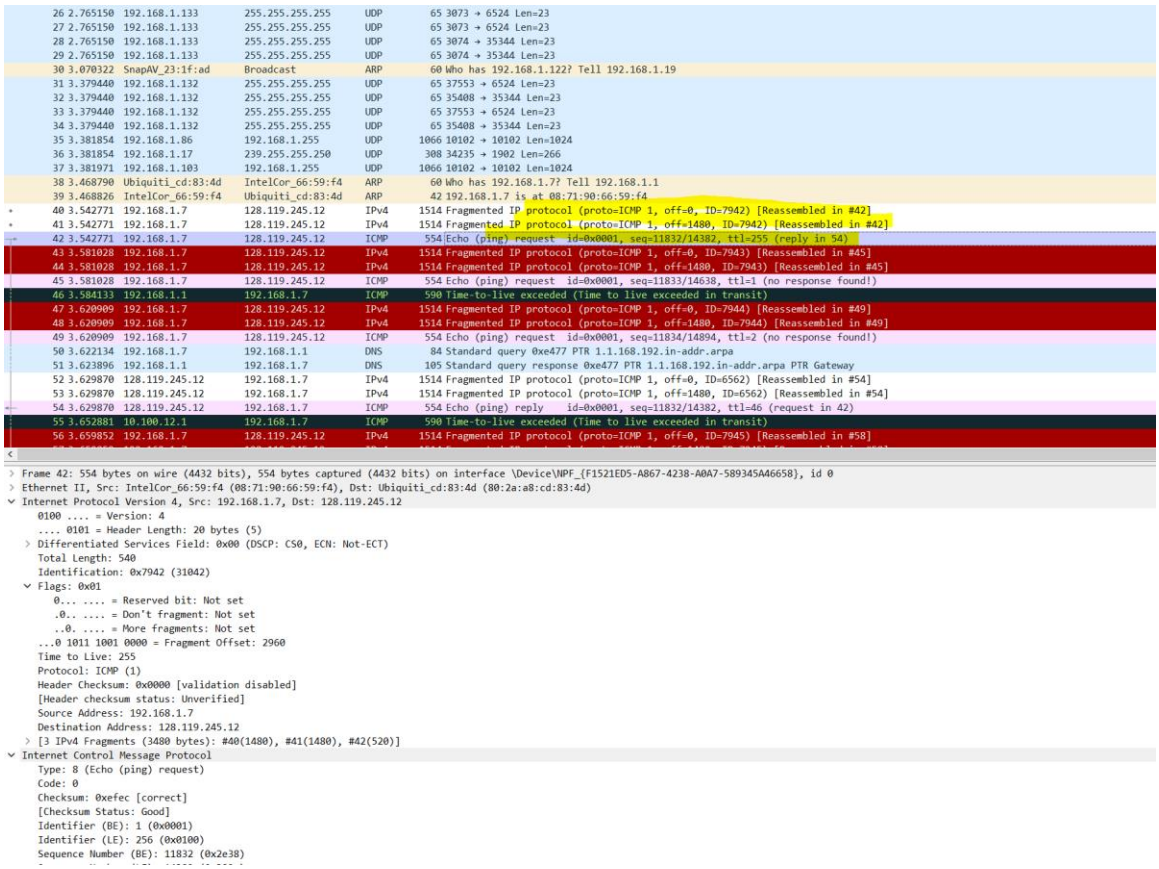
11. Screenshot the first fragment of the fragmented IP datagram (with sufficient details to answer these questions). What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?
    In the above screenshot it shows the more fragments bit has been flagged. We know it's the first portion of the fragment because the fragment offset is 0. The total fragment is 1500
12. Screenshot the second fragment of the fragmented IP datagram (with sufficient details to answer these questions). What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?



The second fragment shows that offset is 1480 and has been reassembled from the previous fragment. We know that there are no more fragments because the more fragments bit is 0

13. What fields change in the IP header between the first and second fragment?
Length, flags set, fragment offset, header checksum.

Now find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 3500.



14. How many fragments were created from the original datagram?
    Three fragments were created from the original data, as shown in the above screenshot.
15. What fields change in the IP header among the fragments?
    The fields that change through all three is the fragment offset (0, 1480, 2960) and the header checksum. The first two fragments have a length of 1500 bytes and the more fragments bit is set. The last fragment has a length of 540 bytes and the more fragment bit is set to 0.