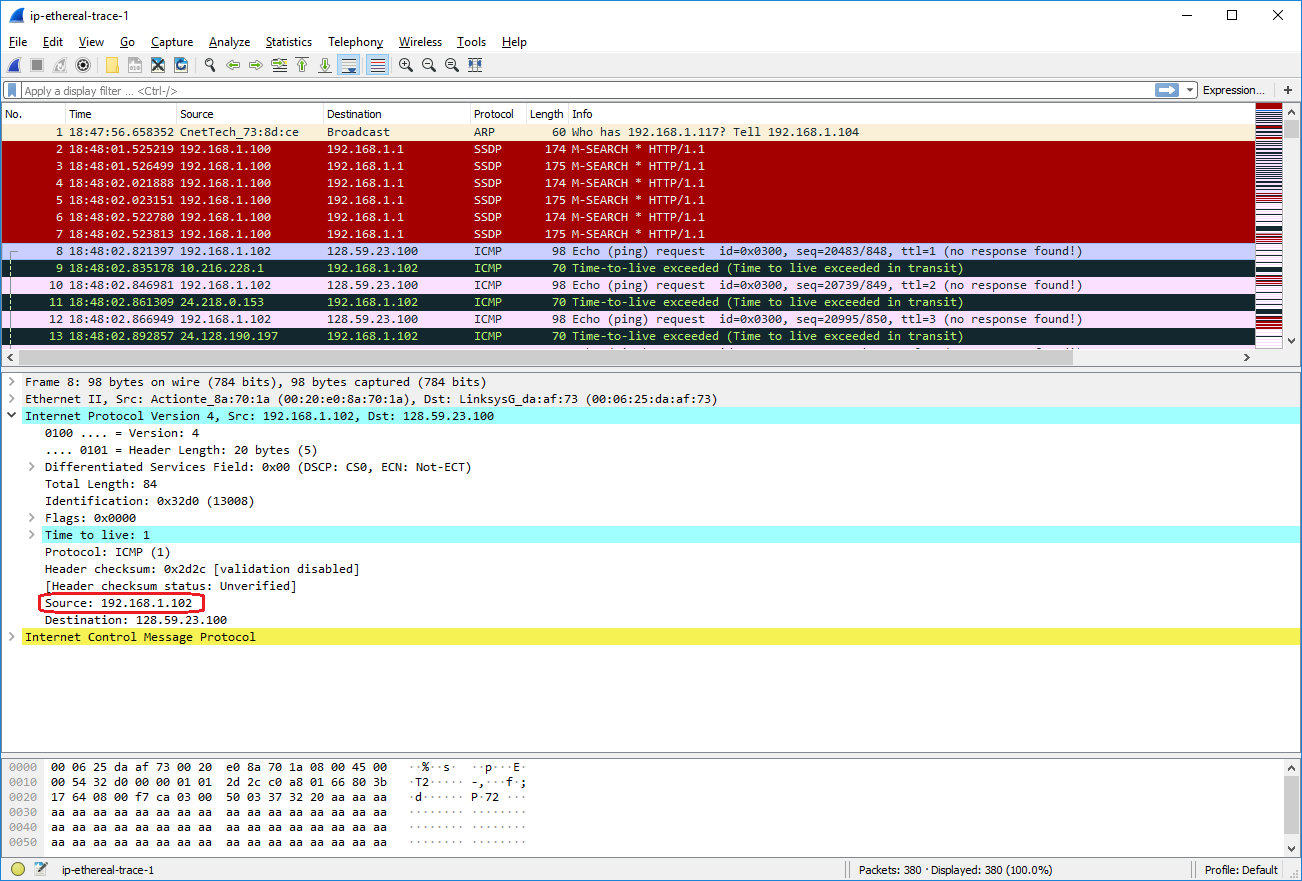
|  |  |
| --- | --- |
| CS372-400 | Edmund Dea |
| 11/18/2019 | ID# 933280343 |

**Lab 4: Wireshark**

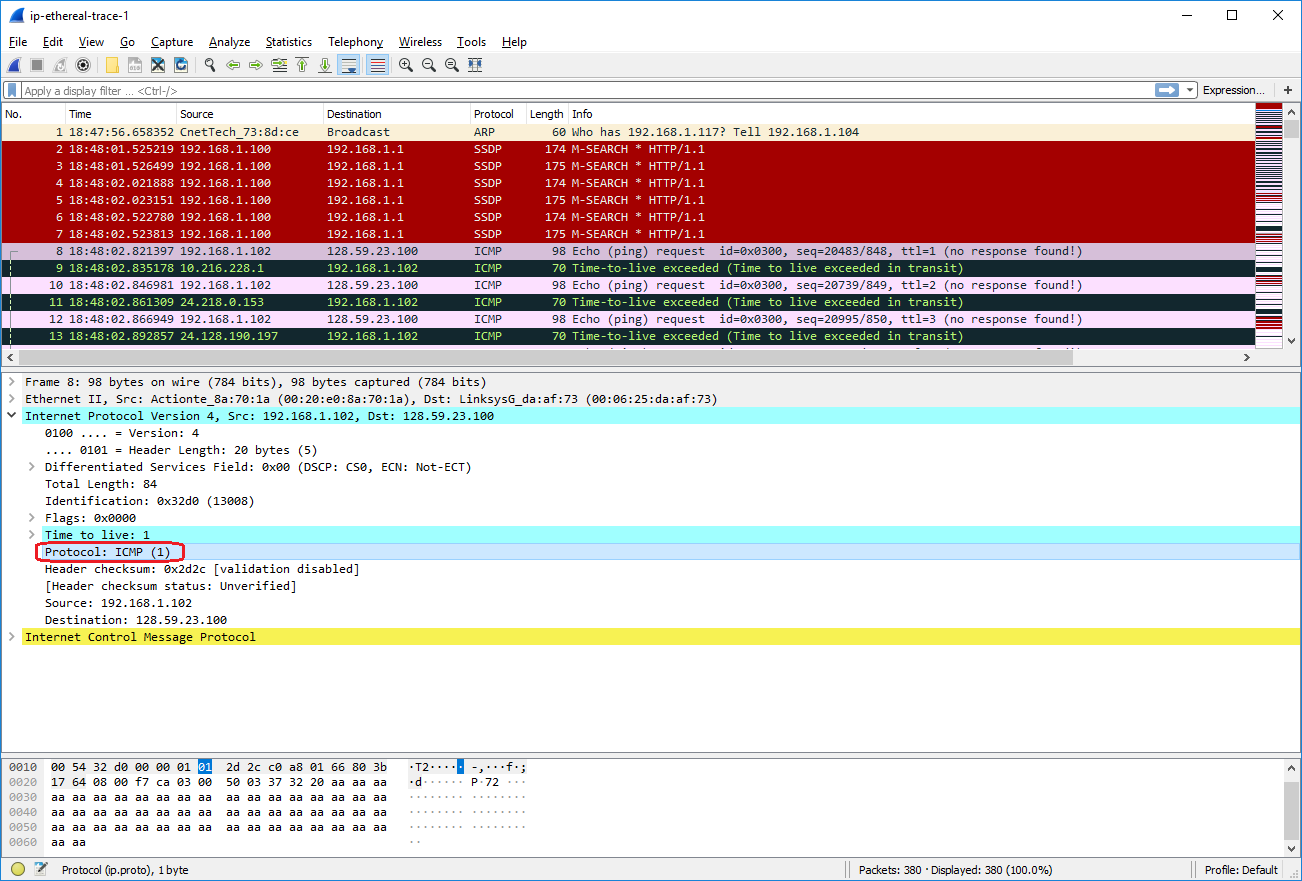
1. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?

Using the author’s trace, the IP address of my computer is 192.168.1.102.



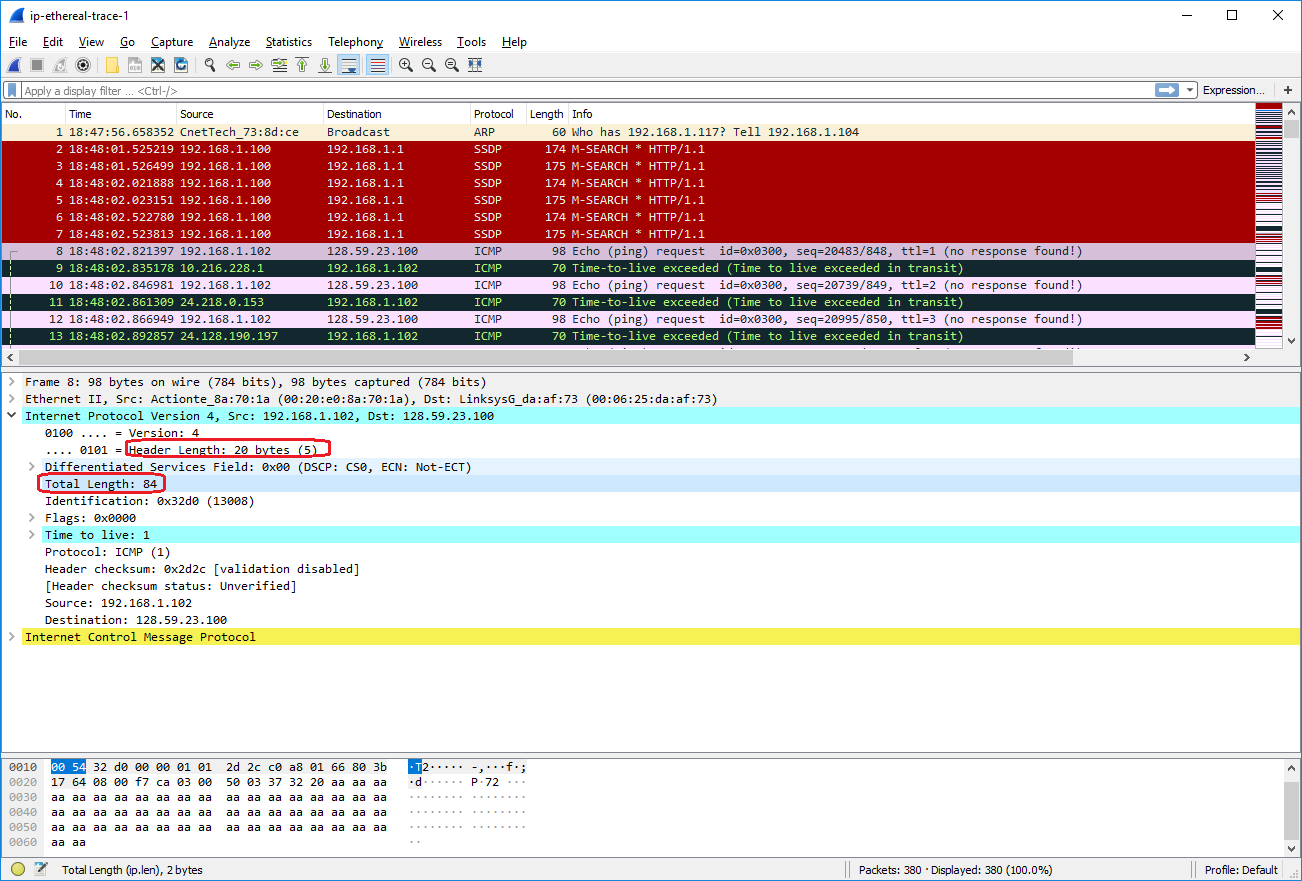
1. Within the IP packet header, what is the value in the upper layer protocol field?

The value in the upper layer protocol field is ICMP (1).



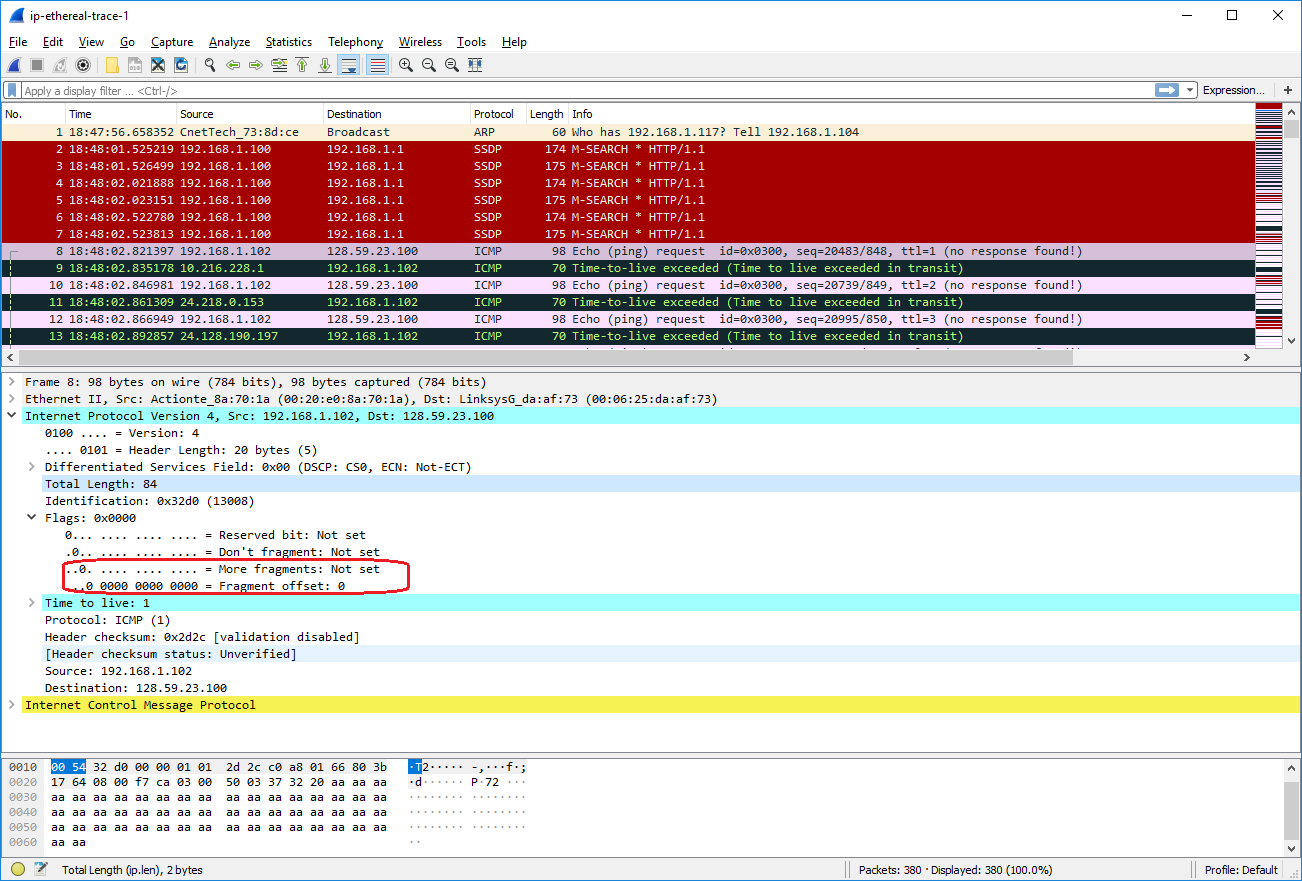
1. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.

There are 20 bytes are in the IP header. This IP packet’s total length is 84 bytes. Therefore, the payload of the IP datagram is 64 bytes (84 – 20 bytes = 64 bytes).



1. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

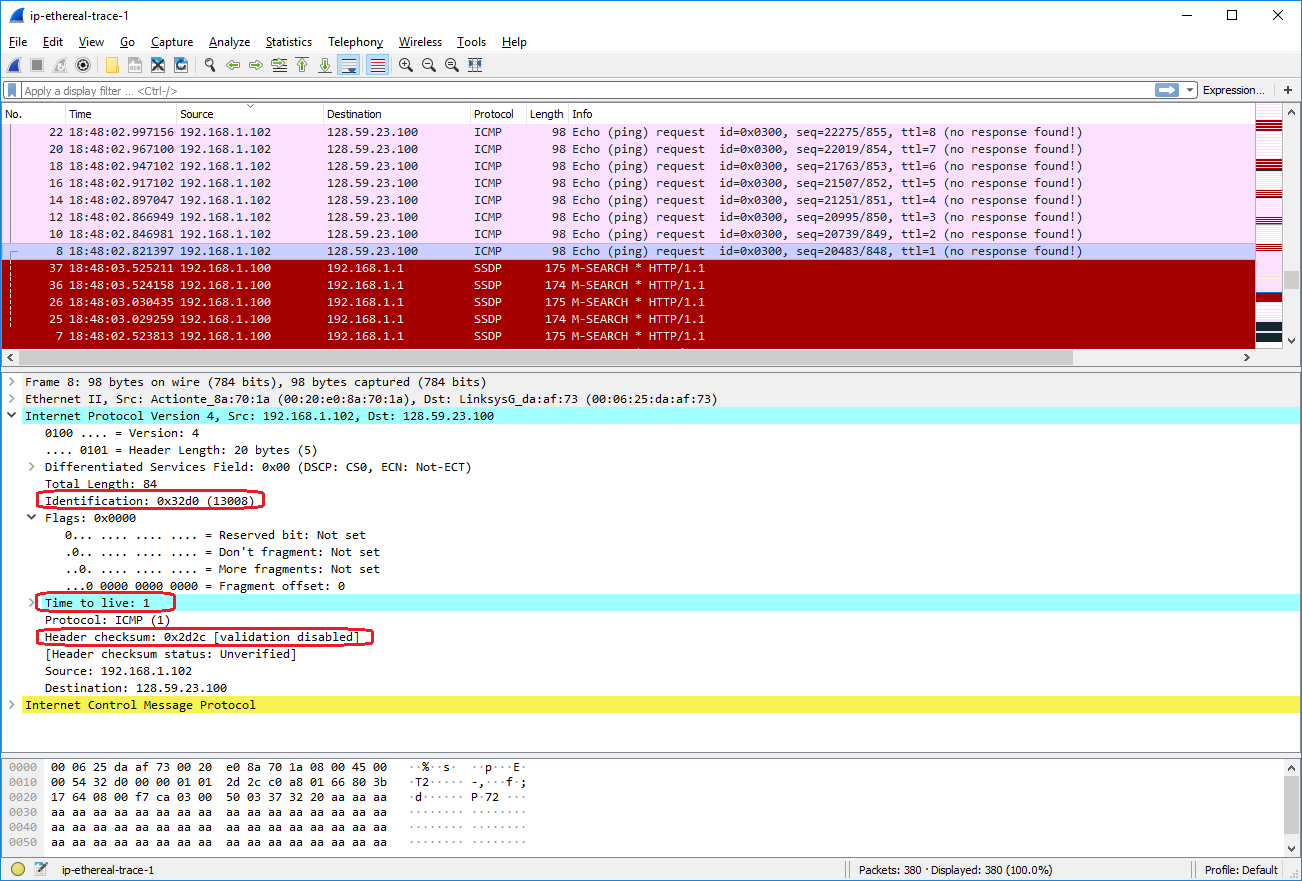
No, this IP datagram has not been fragmented. If it was fragmented, then this first IP datagram would either have the “More fragments” bit set or have a non-zero fragment offset, and the bottom of this IP datagram would have a list of IP fragments. Since this IP datagram does not have the More Fragments bit set and does not have a non-zero fragment offset and there is no list of fragments, it is not fragmented.

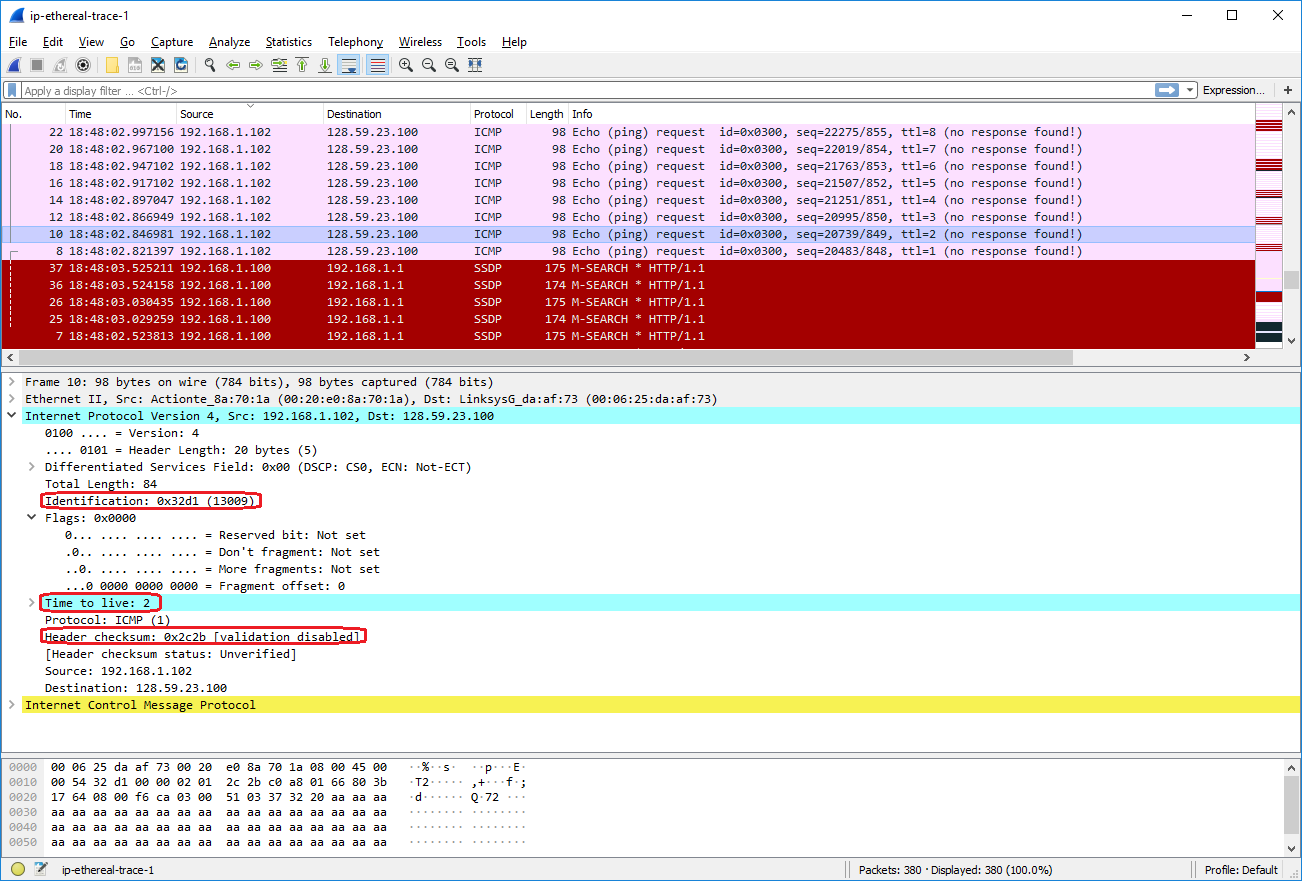


Next, sort the traced packets according to IP source address by clicking on the *Source* column header; a small downward pointing arrow should appear next to the word *Source*. If the arrow points up, click on the *Source* column header again. Select the first ICMP Echo Request message sent by your computer, 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.

1. Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by your computer?

The identification, time to live, and header checksum fields always change from one datagram to the next in this series of ICMP messages, as shown below.

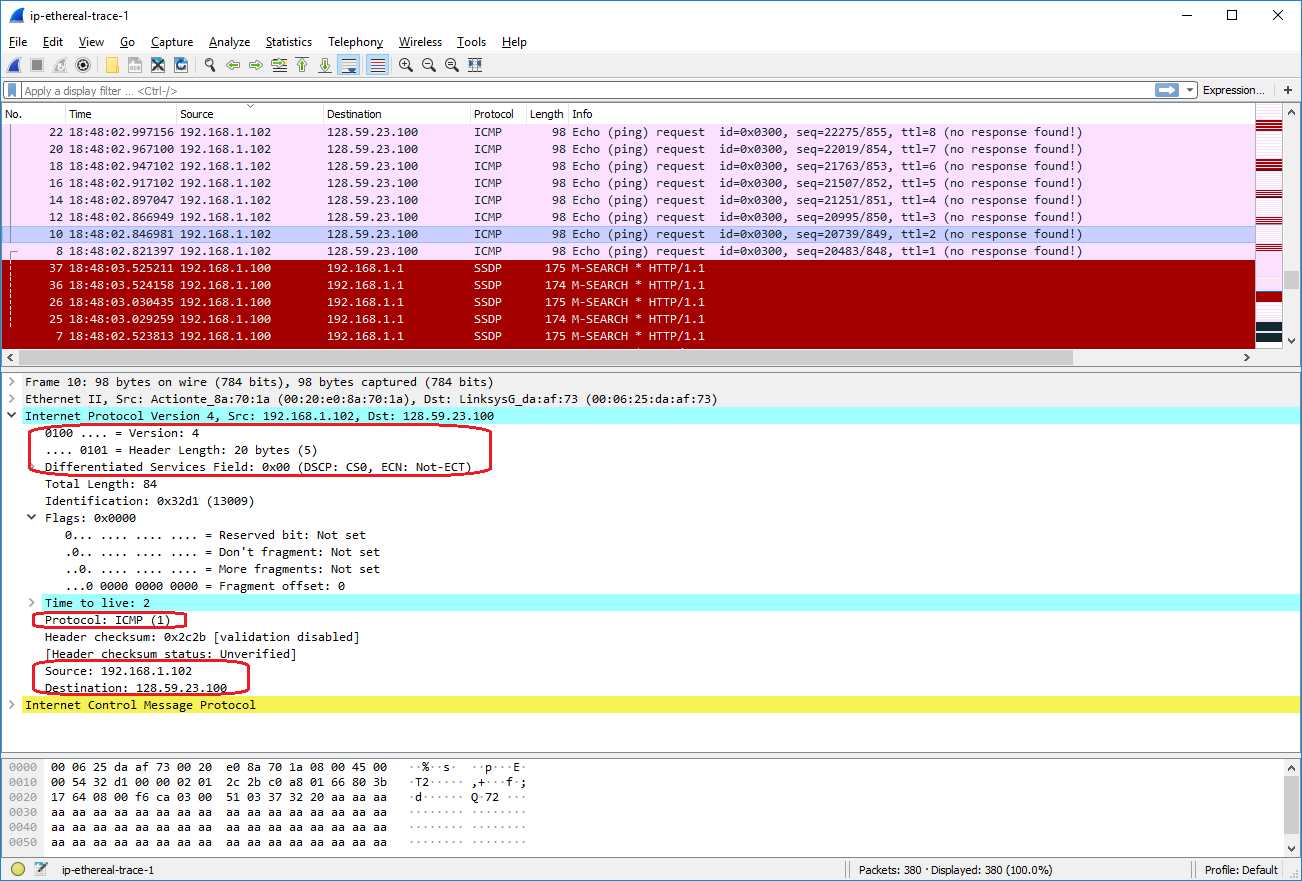




1. Which fields stay constant? Which of the fields *must* stay constant? Which fields must change? Why?

The Version, Header Length, Differentiated Services Field, Protocol, Source, and Destination fields stay constant.

* Version does not change since all the ICMP echo packets are using the same IPv4 network protocol.
* Header Length does not change since these are all ICMP packets, which has a fixed size of 20 bytes.
* Differentiated Services does not change because



The Protocol, Source, and Destination fields must stay constant.

The fields must change.

1. Describe the pattern you see in the values in the Identification field of the IP datagram.

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.

1. What is the value in the Identification field and the TTL field?
2. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

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

1. 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? [Note: if you find your packet has not been fragmented, you should download the zip file [http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip](http://gaia.cs.umass.edu/ethereal-labs/ethereal-traces.zip) and extract the *ip-ethereal-trace-1*packet trace. If your computer has an Ethernet interface, a packet size of 2000 *should* cause fragmentation.[[1]](#footnote-1)]
2. 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?
3. 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?
4. What fields change in the IP header between the first and second fragment?

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.

1. How many fragments were created from the original datagram?
2. What fields change in the IP header among the fragments?

1. The packets in the *ip-ethereal-trace-1* trace file in [http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip](http://gaia.cs.umass.edu/ethereal-labs/ethereal-traces.zip) are all less that 1500 bytes. This is because the computer on which the trace was gathered has an Ethernet card that limits the length of the maximum IP packet to 1500 bytes (40 bytes of TCP/IP header data and 1460 bytes of upper-layer protocol payload). This 1500 byte value is the standard maximum length allowed by Ethernet. If your trace indicates a datagram longer 1500 bytes, and your computer is using an Ethernet connection, then Wireshark is reporting the wrong IP datagram length; it will likely also show only one large IP datagram rather than multiple smaller datagrams.. This inconsistency in reported lengths is due to the interaction between the Ethernet driver and the Wireshark software. We recommend that if you have this inconsistency, that you perform this lab using the *ip-ethereal-trace-1* trace file. [↑](#footnote-ref-1)