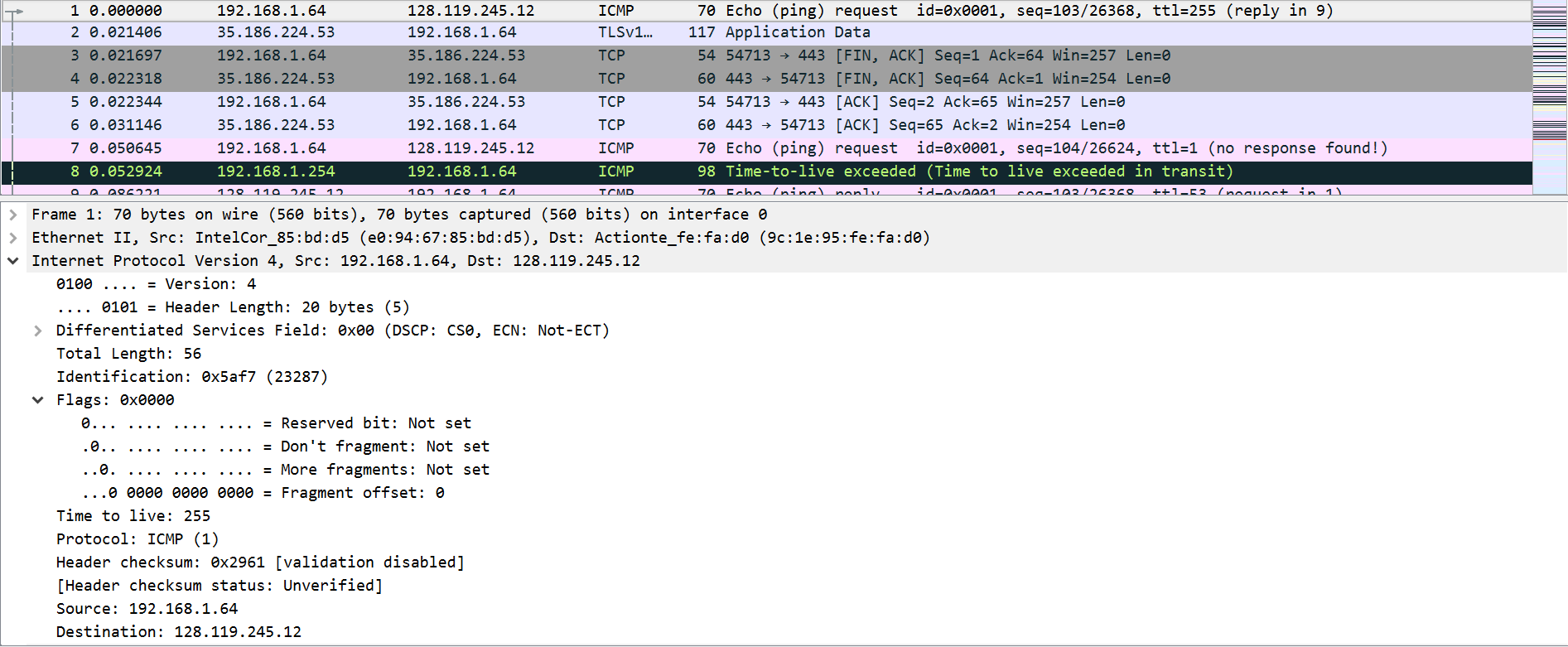
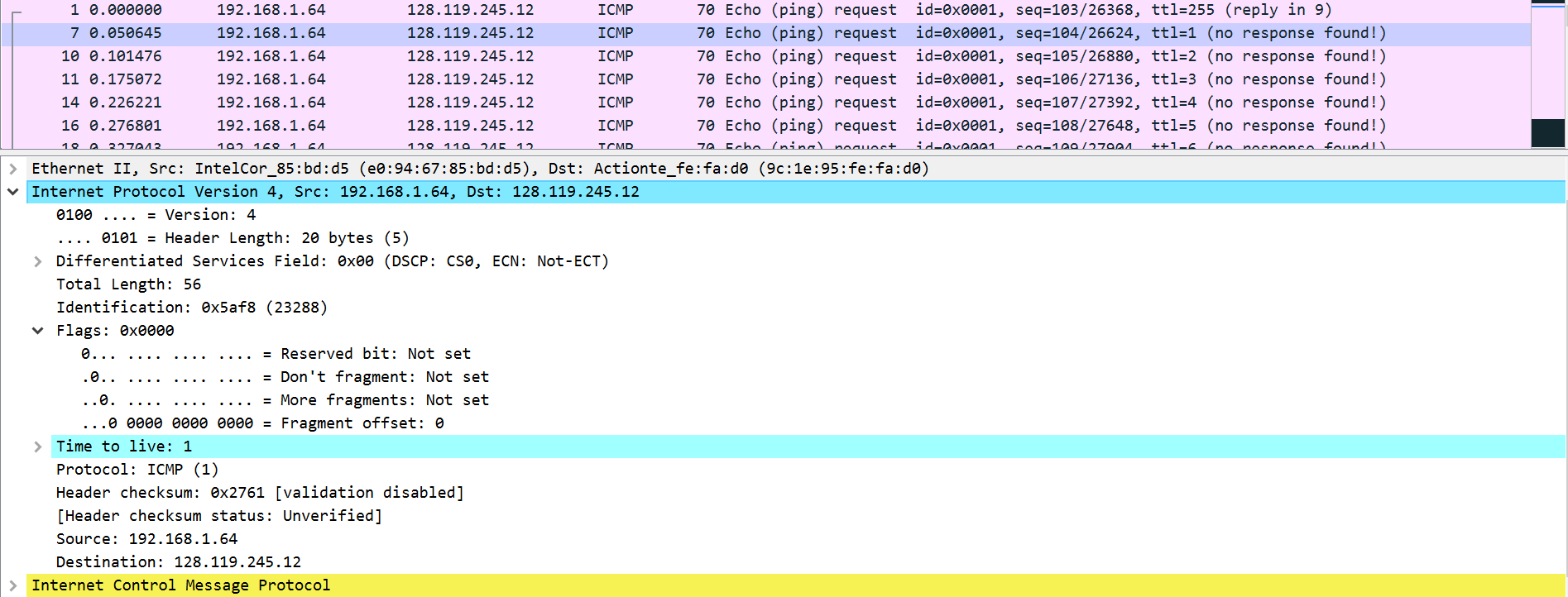
# Wireshark IP



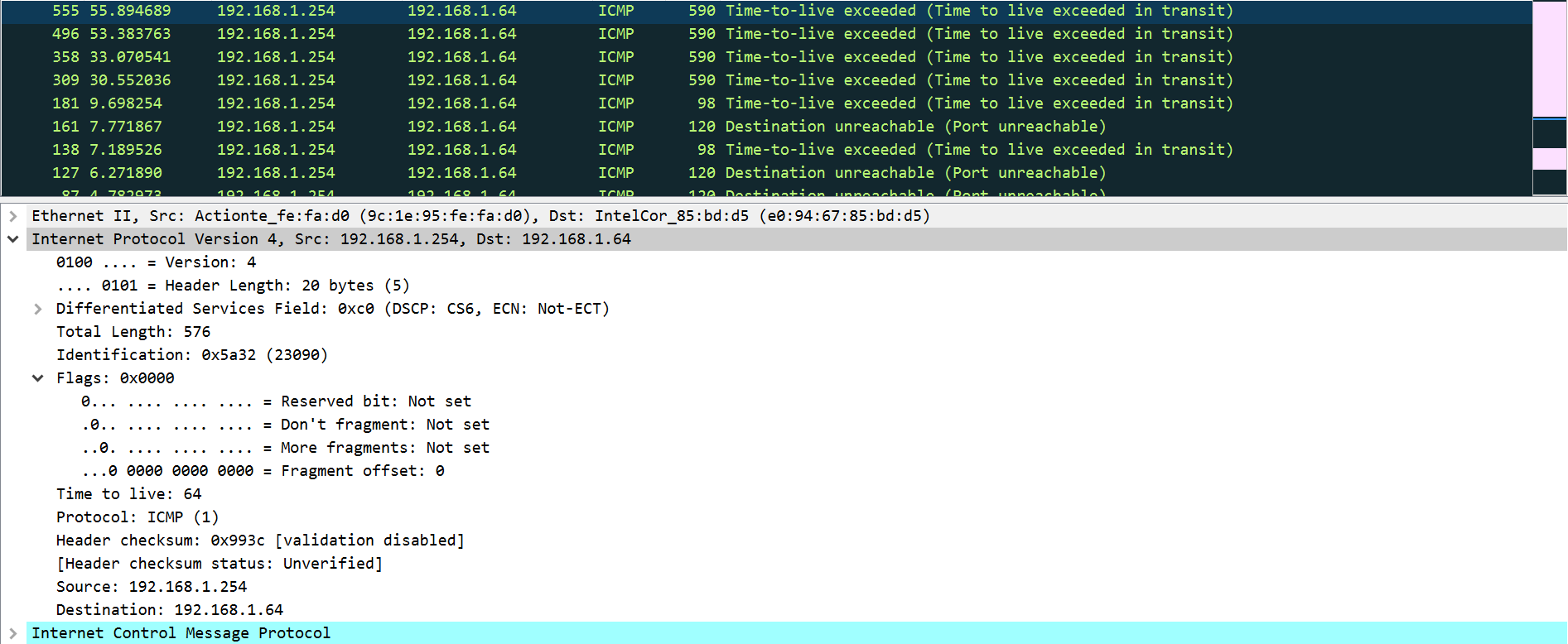
## Figure #1: First ICMP Echo Request

1. The IP address of my computer is 192.168.1.64
2. The value of the upper layer protocol is ICMP (0x01)
3. The header has 20 bytes as indicated by the Header Length field. The total packet length is 56 bytes as indicated by the Total Length field, leaving 36 bytes for the payload.
4. No. This IP datagram has not been fragmented as indicated by the More Fragments bit.



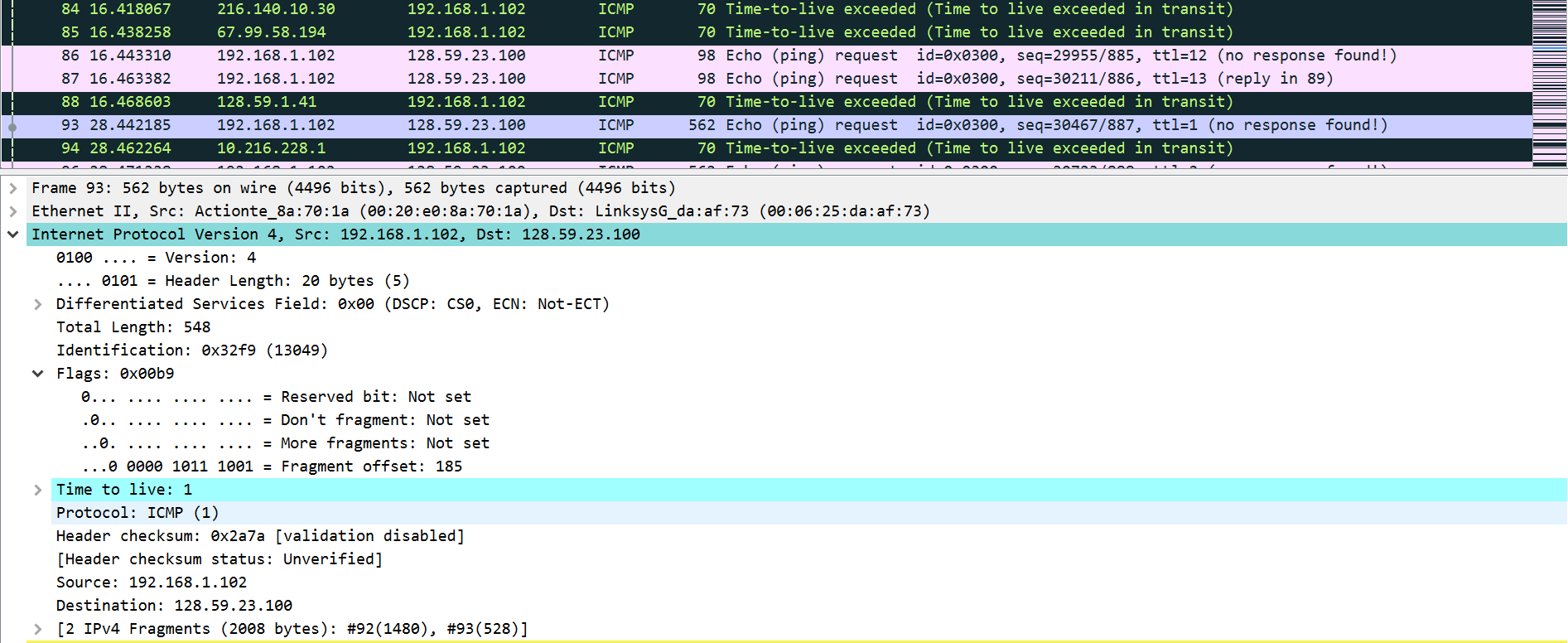
## Figure #2: Second ICMP Echo Request

1. The fields that always change are:
   1. ID
   2. TTL
   3. Header checksum
   4. The fields that stay constant are:
      1. Version – Same version of IP (IPv4)
      2. Header Length – All packets are ICMP, so header length does not change
      3. Differentiated Services – All packets are ICMP, so same service class
      4. Source and Destination IP – Same host and same destination
      5. Upper Layer Protocol – All packets are ICMP packets
   5. The fields that must stay constant are:
      1. Same as a).
   6. The fields that must change are:
      1. ID – IP packets must have different ID
      2. TTL – This is incremented at every router
      3. Header checksum – Since TTL changes, the checksum must also change
2. The pattern I observe is that the value of the ID field increments after each ICMP request.

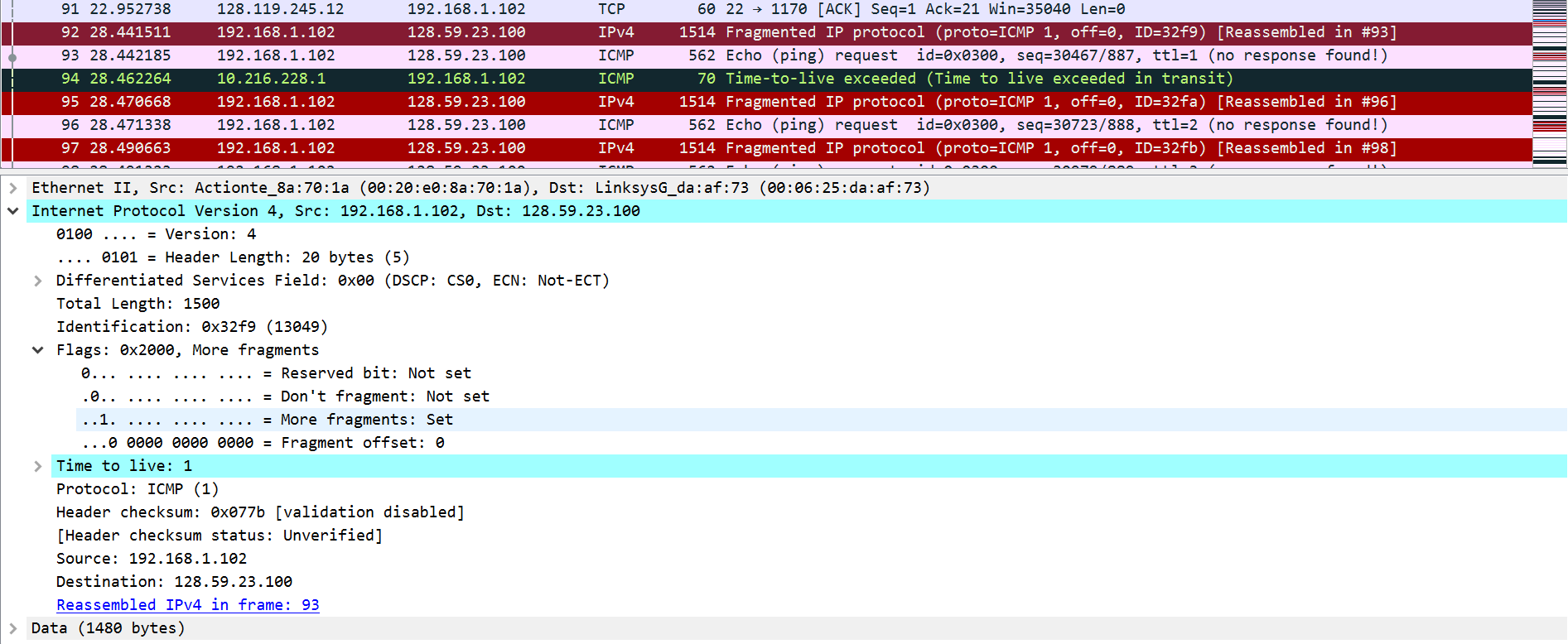


## Figure #3: First ICMP Echo Response – TTL Expired

1. The value of the ID field is 23090 and the value of the TTL field is 64.
2. The value of the ID field changes as it gets incremented. The value of the TTL field does not change however because this is all for the “first-hop router.”

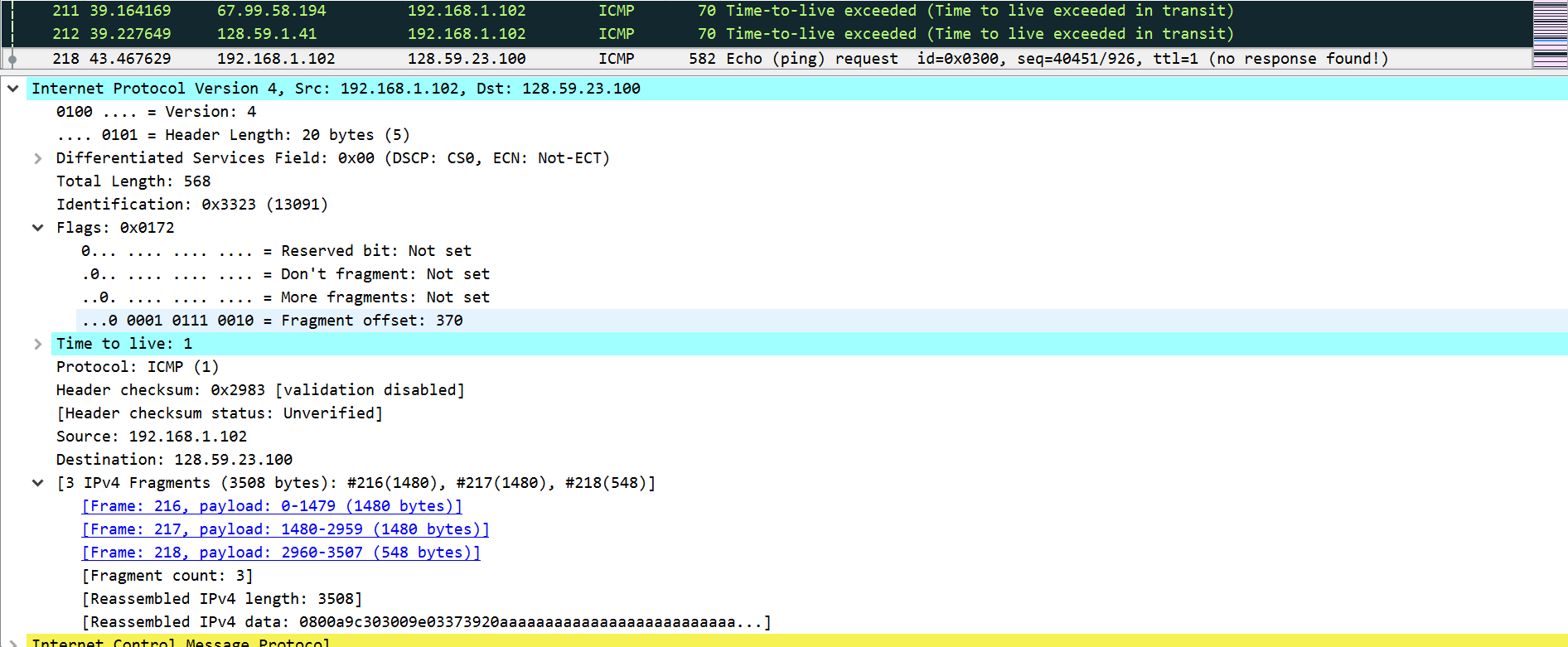


## Figure #4: Fragmented ICMP Request



## Figure #5: First fragment of the Fragmented Datagram (2000 Bytes)

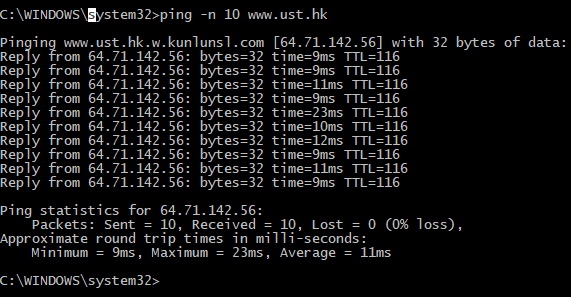
1. I will be using the provided trace for these questions. Here, the packet was fragmented into 2 IPv4 fragments totalling up to 2000 bytes. (Fig 4)
2. The header has the “More fragments” flag set to 1, meaning it has been fragmented. Looking at the fragment offset field, we see that it is 0, meaning that it is the first fragment. (Fig 5). The unfragmented IP datagram is 2000 bytes but is 2020 bytes long due to headers. The first fragment is 1500 bytes (1480 payload and 20 header).
3. The “More fragments” flag was not set and we see the offset has been correctly set. Not only that, it mentions that the first fragment is #92 (Fig 5) and second fragment is #93 (Fig 4).
4. Between the two fragments, the total length, flags (more fragments and fragment offset), and checksum have changed.



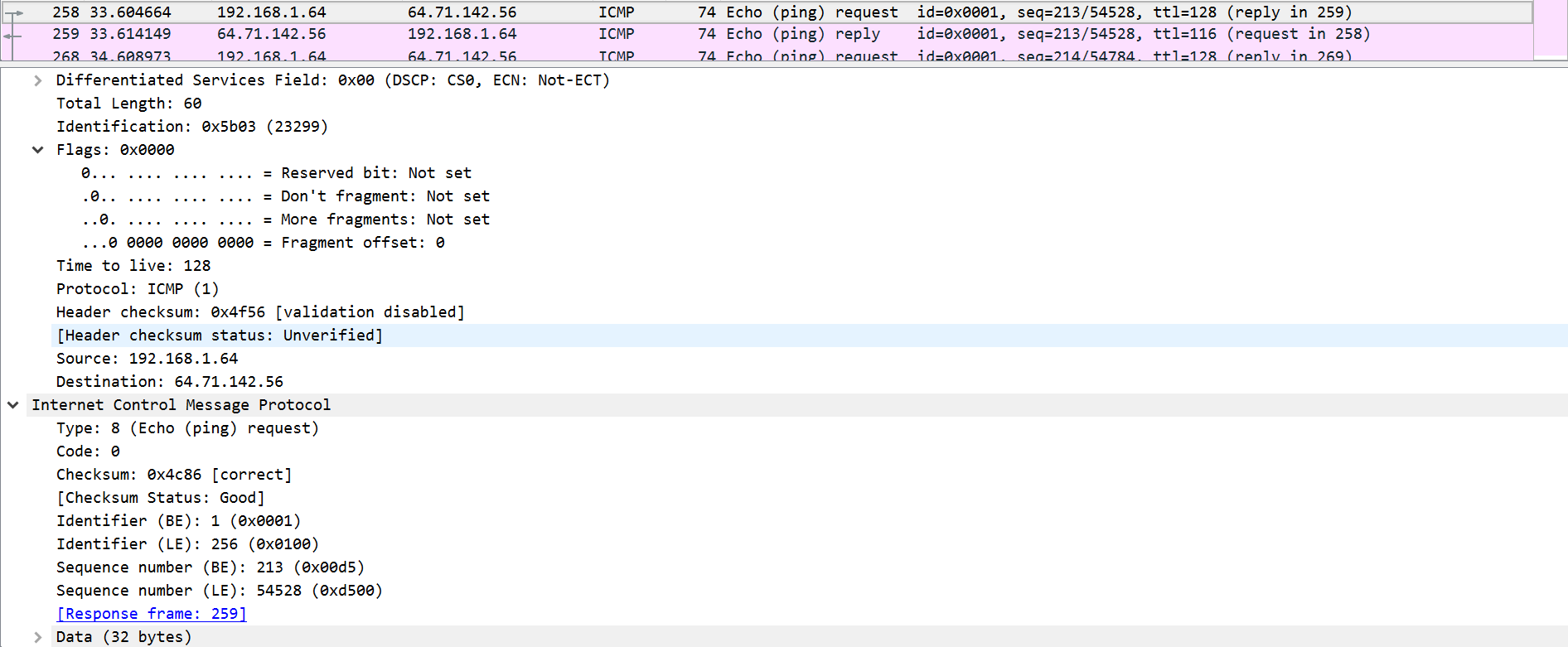
## Figure #6: First fragment of the Fragmented Datagram (3500 Bytes)

1. 3 fragments were created.
2. The fields that change among the fragments are:
   1. More fragments
   2. Fragment offset
   3. Checksum
   4. Length (last packet only has 568 while the other have 1500)

# Wireshark ICMP



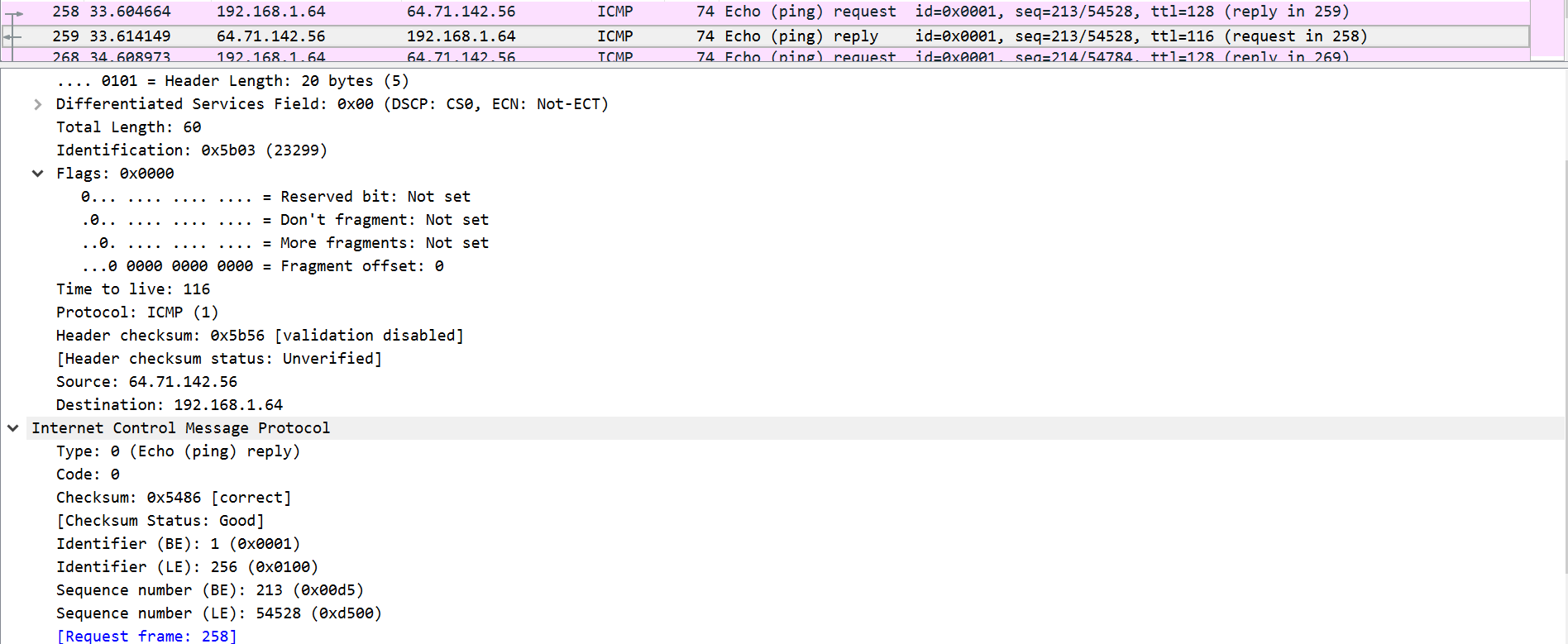
## Figure #7: Ping results



## Figure #8: ICMP PING request

1. As before, the IP address of my host is 192.168.1.64 and the IP address of the destination host is 64.71.142.56.
2. An ICMP packet does not have source and destination ports because ICMP is part of the network layer with IP instead of the transport layer with TCP / UDP. As it is an integral part of the network layer, it was designed to communicate network layer information between hosts and routers and not between application layer processes.
3. Referring to Fig 8, we see that the Type is 8 for Echo (ping) request and Code is 0. It has a few other fields, namely checksum, identifier, sequence number, and data fields. There are two versions of both sequence number and identifier, for little and big endian.

The checksum, sequence number, and identifier fields are all 2 bytes each.



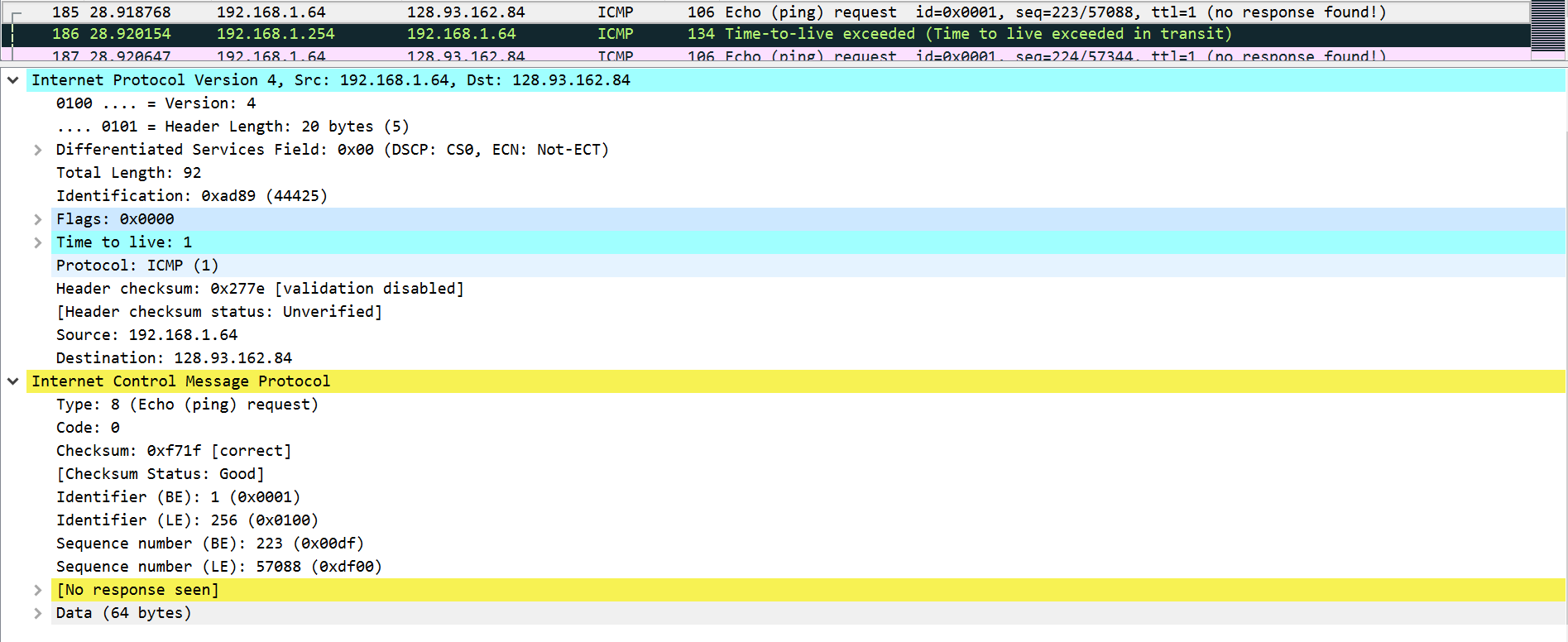
## Figure #9: ICMP PING reply

1. For the reply, Type is 0 for Echo (ping) reply and Code is 0. It has the same fields as the request above, namely checksum, identifier, sequence number, and data. In addition, the checksum, sequence number, and identifier fields are all 2 bytes each.



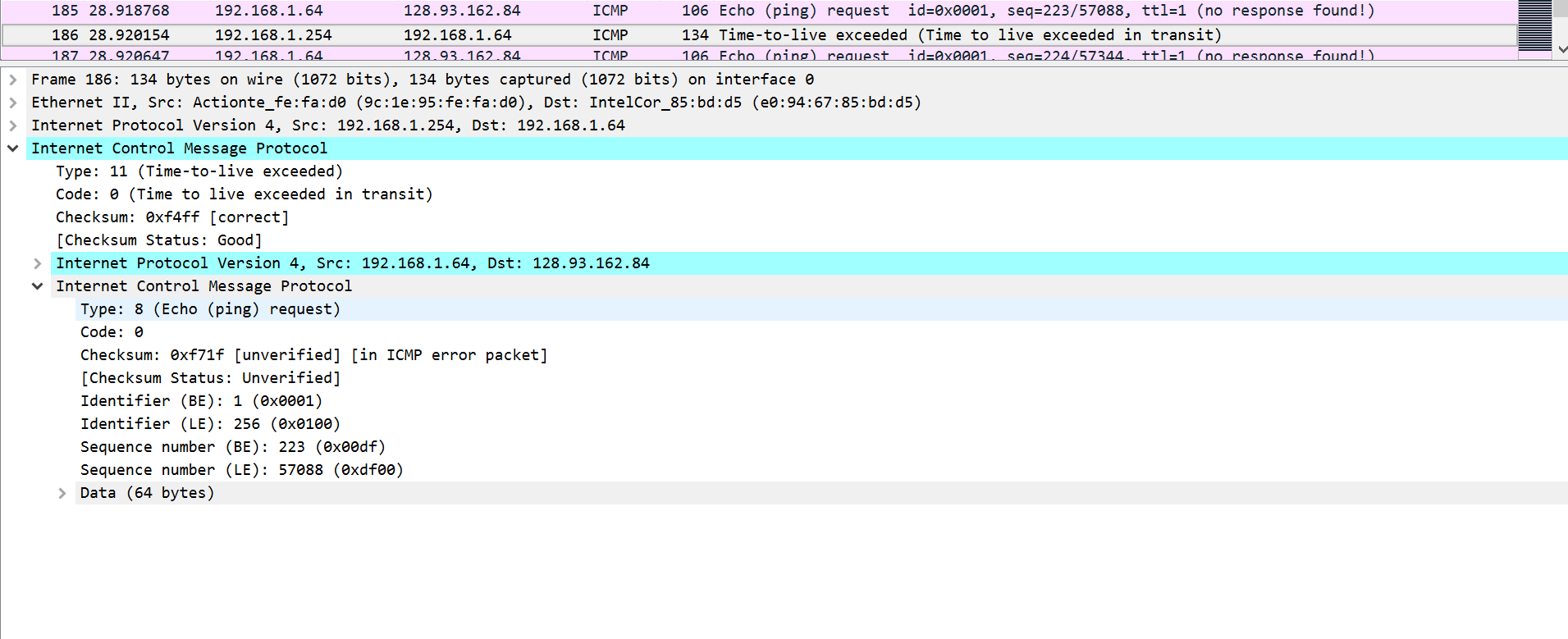
## Figure #10: Tracert results

1. As before, the IP address of my host is 192.168.1.64 and the IP address of the destination host is 128.93.162.84.
2. No, if ICMP sent UDP packets instead, the protocol field in the IP header should be 0x11 for UDP (17).



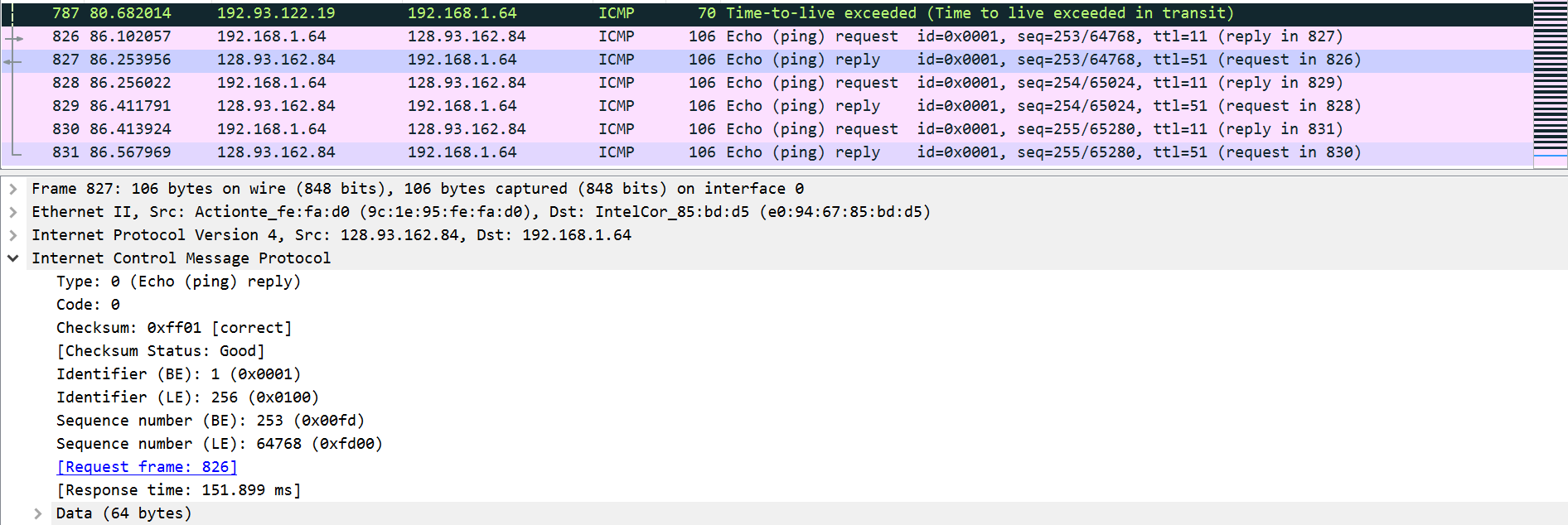
## Figure #11: Tracert Request

1. The ICMP echo packet seems to have the same fields as the ping query packets from above.



## Figure #12: Tracert Error (TTL expired)

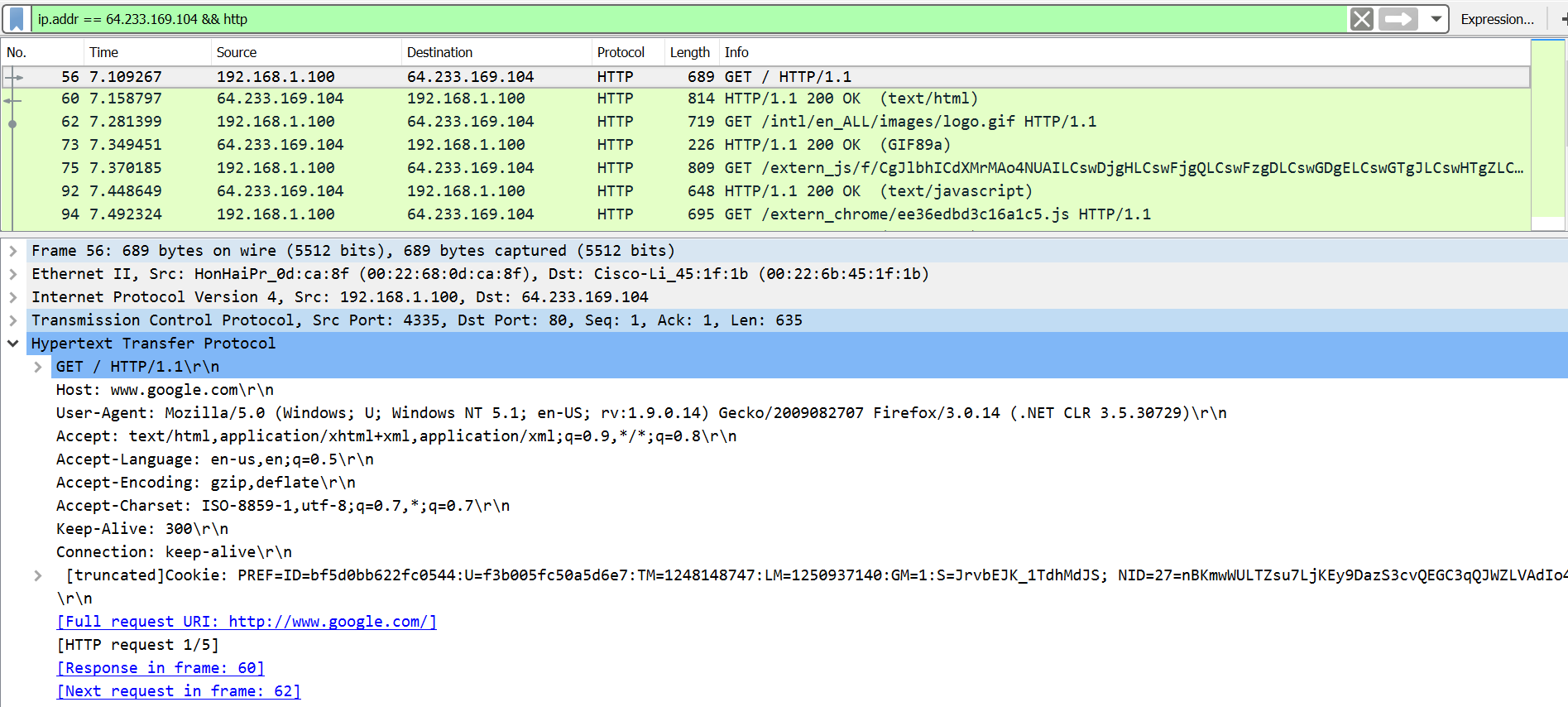
1. There are a few extra fields in the ICMP error packet – it includes the IP header and the first 8 bytes of the ICMP request the error packet corresponds to.



## Figure #13: Last three ICMP packets

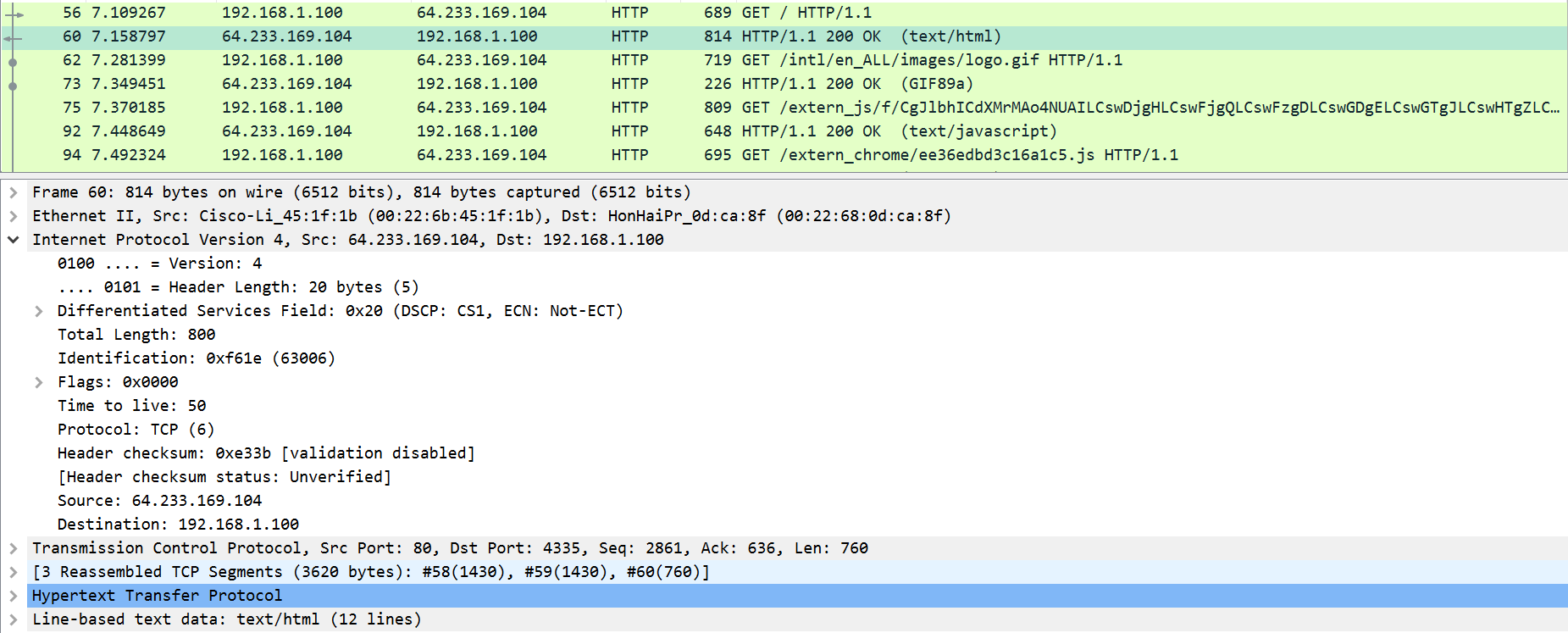
1. The last three ICMP packets are different as they have Type 0 for Echo (ping) reply instead of 11 for TTL expiration. They are different from the ICMP error packets because these datagrams have reached the destination before the TTL expired.
2. Yes. Referring to Fig 10, we see that between steps 6 and 7, there is a sudden increase in RTT. Based on the router name, I would assume that par7 is referring to Paris as the website is a French website (.fr).

# Wireshark NAT



## Figure #14: HTTP Request and Filter

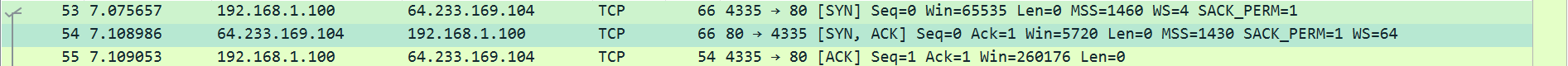
1. The IP address of the client is 192.168.1.100
2. Filtering above (Fig 14)
3. Source: 192.168.1.100, 4335 and Destination: 64.233.169.104, 80 (A.B.C.D, Port Number)



## Figure #15: HTTP 200 Response

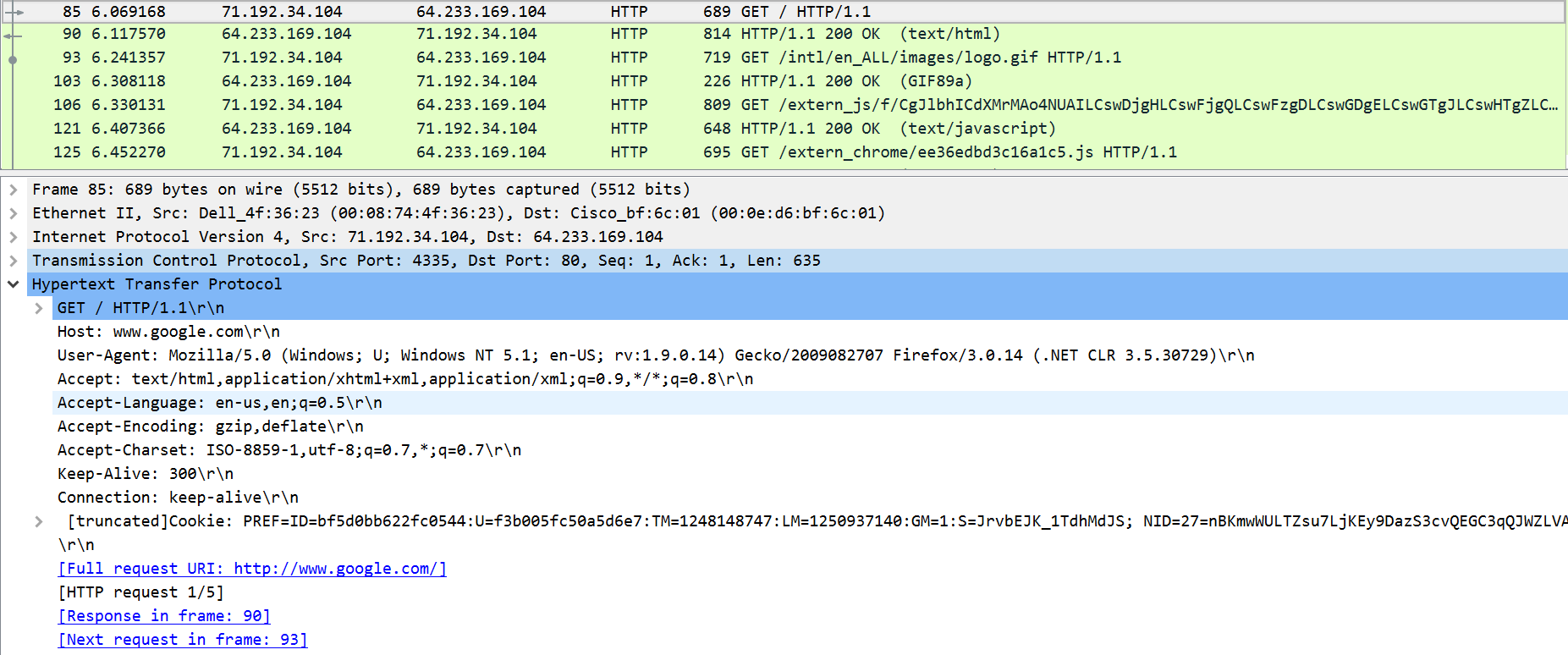
1. The 200 OK HTTP message was received at time t = 7.158797.

Source: 64.233.169.104, 80 and Destination: 192.168.1.100, 4335 -> reverse of 3.



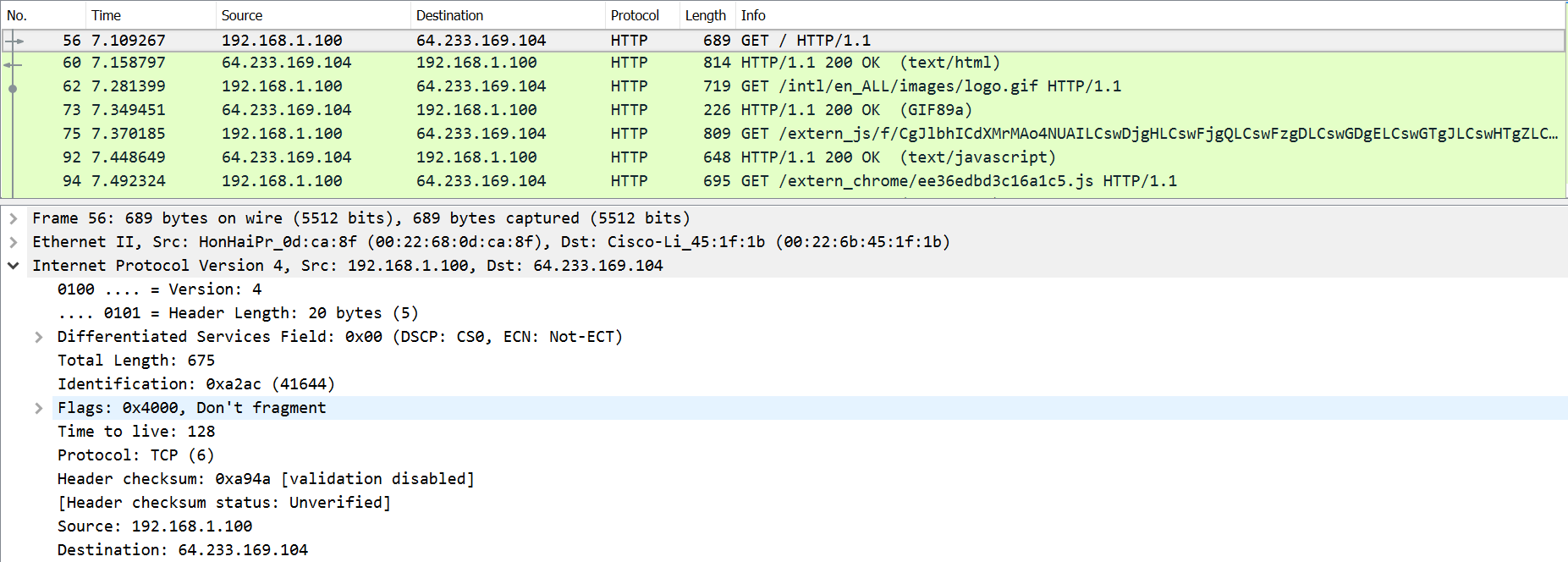
## Figure #16: TCP SYN/ACK segments

1. The TCP SYN segment was sent at t = 7.075657. The TCP ACK was received at t = 7.108986.
   1. TCP SYN segment:
      1. Source: 192.168.1.100, 4335 and Destination: 64.233.169.104, 80
   2. TCP ACK response:
      1. Source: 64.233.169.104, 80 and Destination: 192.168.1.100, 4335

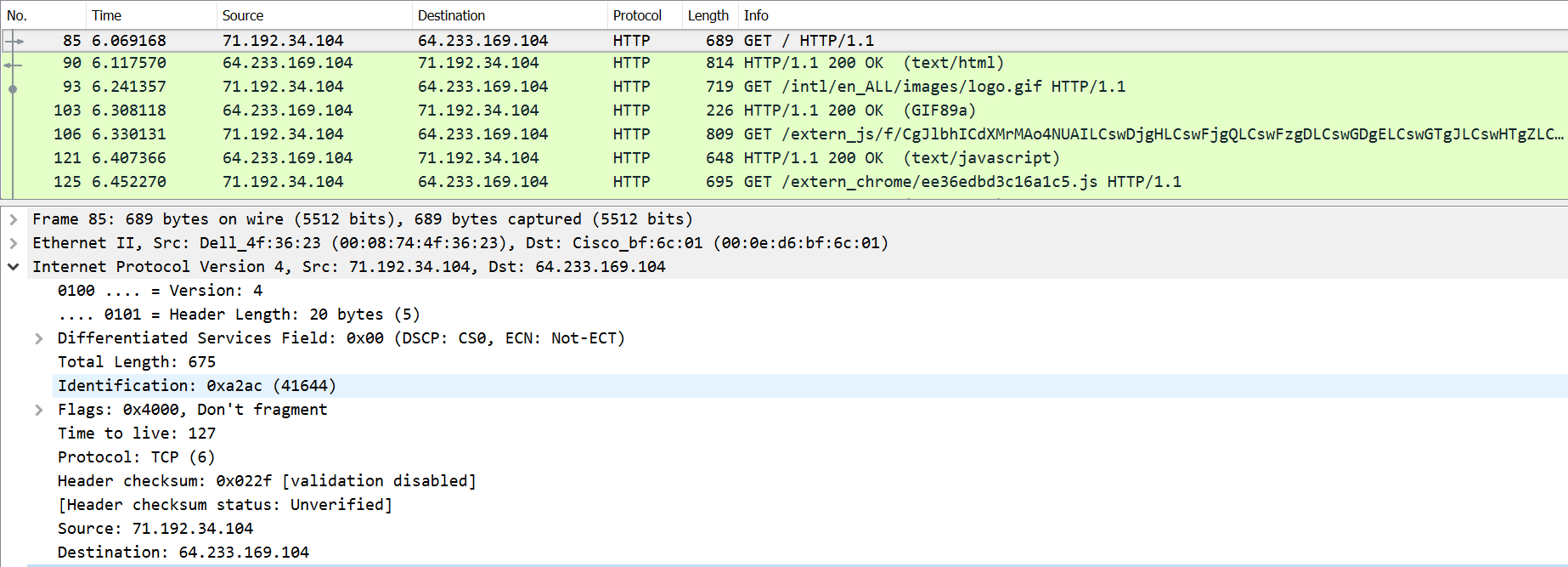


## Figure #17: NAT ISP Side of the GET request

1. The request appears on the ISP side at t = 6.069168. For this request, the destination IP and Port and source Port are the same, only the source IP has changed.
   1. Source: 71.192.34.104, 4335 and Destination: 64.233.169.104, 80

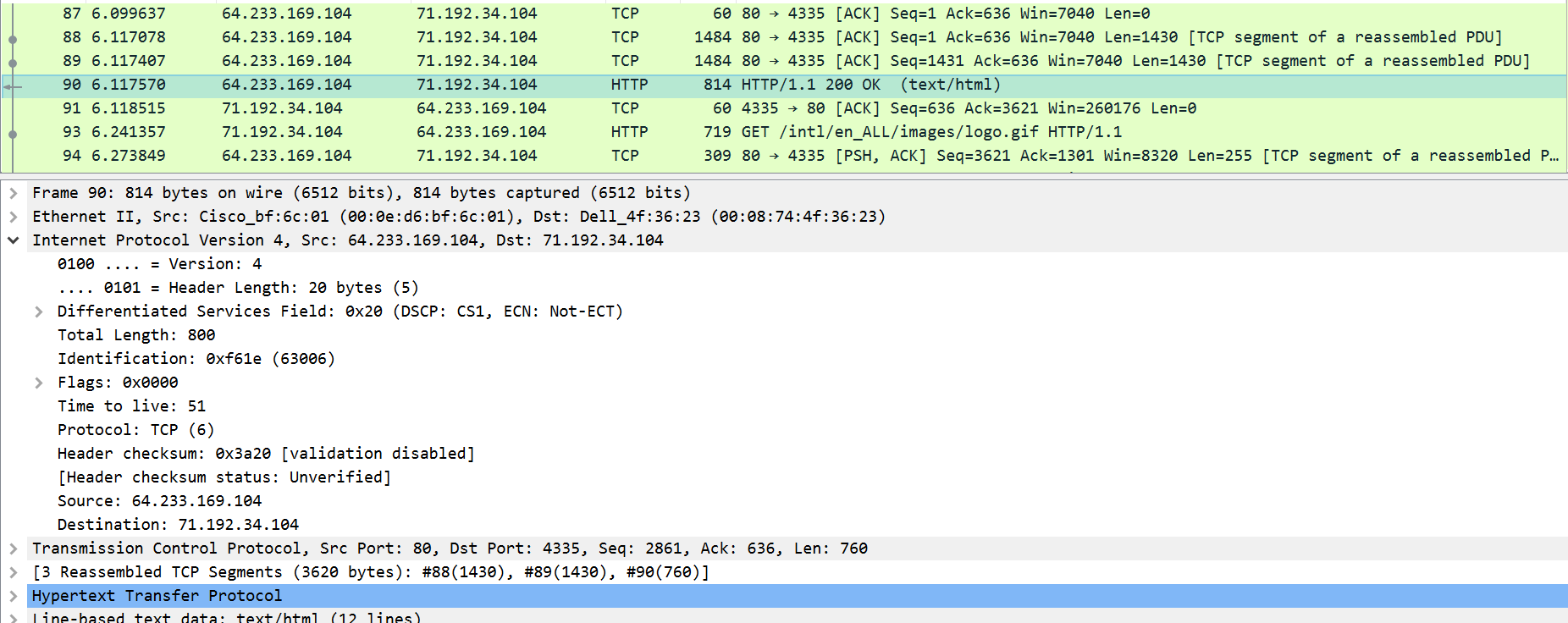


## Figure #18: NAT Home GET Request IP Datagram



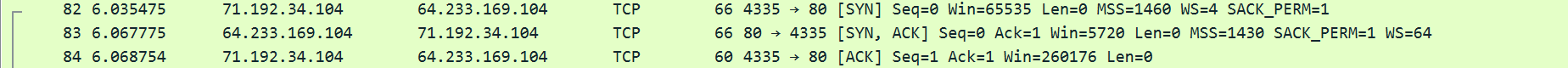
## Figure #19: NAT ISP GET Request IP Datagram

1. No, nothing in the HTTP GET message was changed. This can be confirmed by referring to Figures 14 and 17. Regarding the IP datagram carrying the HTTP GET, the Version, Header Length, and Flags were not changed. The only thing that was changed was checksum as both IP address and TTL changed. These changes come from a NAT table lookup.



## Figure #20: NAT ISP 200 OK IP Datagram

1. The first 200 OK HTTP message came arrived at t = 6.117570. Comparing Figures 15 and 20, it looks like the only things that are different are the Destination IP address, TTL, and checksum.
   1. Source: 64.233.169.104, 80 and Destination: 71.192.34.104, 4335



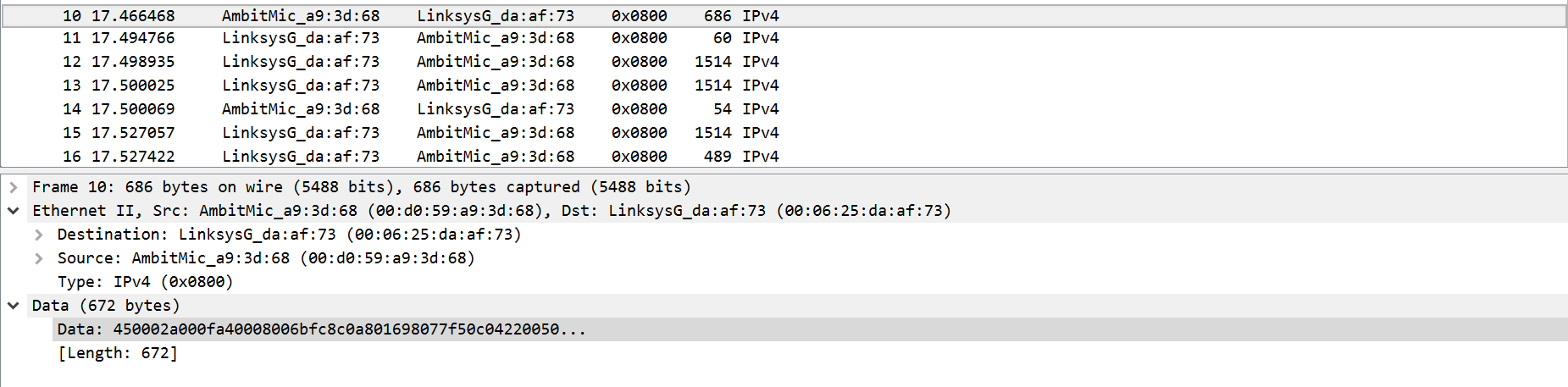
## Figure #21: TCP SYN/ACK Segments

1. The TCP SYN segment was captured at t = 6.035475 and the TCP ACK segment was captured at t = 6.067775.
   1. TCP SYN:
      1. Source IP changed
   2. TCP ACK:
      1. Destination IP changed
2. NAT Table

|  |  |
| --- | --- |
| NAT translation table | |
| WAN side addr | LAN side addr |
| 71.192.34.104, 4335 | 192.168.1.100, 4335 |

# Wireshark Ethernet and ARP

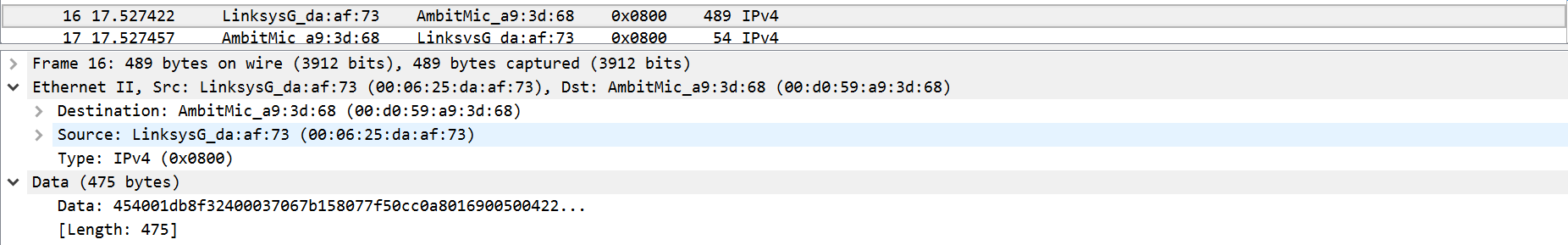
Note: I will be using the provided ethereal trace for this part of the assignment.



## Figure #22: Ethernet Frame for HTTP GET Request

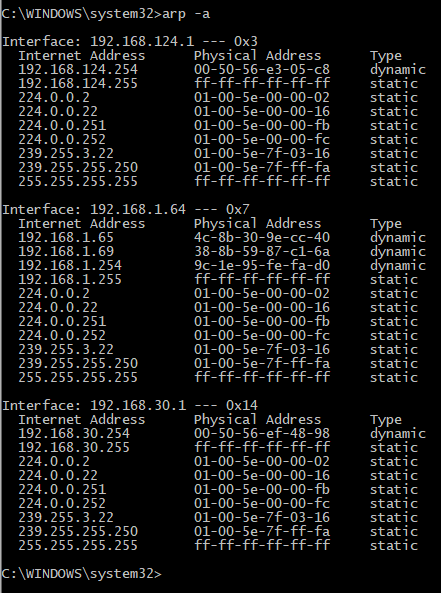
1. The 48-bit Ethernet address of my computer is 00:d0:59:a9:3d:68.
2. The 48-bit destination Ethernet address is 00:06:25:da:af:73. This corresponds to the Linksys router that is used to get off the local subnet.
3. The hexadecimal value for the Frame Type is 0x0800, corresponding to the IPv4 protocol.
4. As the ASCII G is the first thing in the payload, it would appear 54 bytes from the very start of the Ethernet frame.

This is because the HTTP GET message is carried inside of a TCP segment, which is carried inside an IP datagram, which is finally carried inside the Ethernet frame. Thus, we must consider 20 bytes of header from the TCP segment, 20 bytes from the IP datagram, and finally 14 bytes for Ethernet frame’s type and source and destination address.



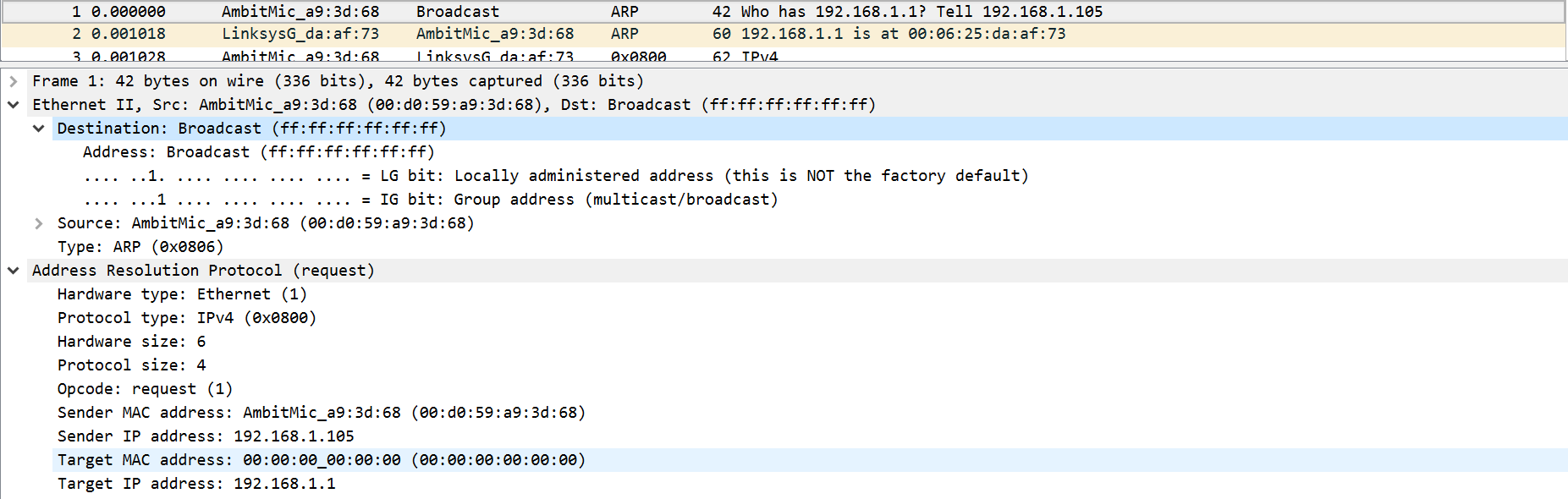
## Figure #23: Ethernet Frame for HTTP GET Response

1. The Ethernet source address is 00:06:25:da:af:73, which is the Ethernet address of the Linksys router as mentioned in Question 2.
2. The Ethernet destination address is 00:d0:59:a9:3d:68, the address of my computer.
3. The hexadecimal value for the Frame Type is 0x0800, corresponding to the IPv4 protocol.
4. Similar to Question 4, as the O in OK is the first character, it will be 54 bytes away from the very start of the Ethernet frame. The calculations are the exact as Question 4.



## Figure #24: ARP Table

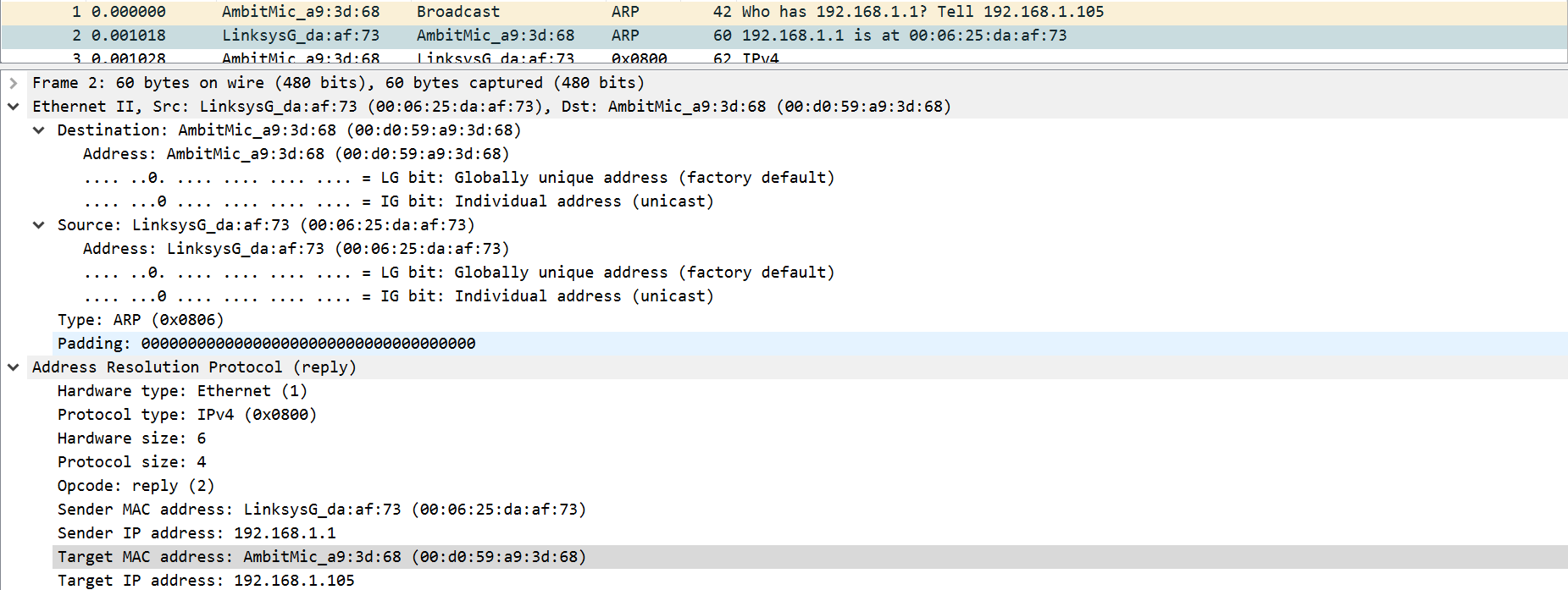
|  |  |
| --- | --- |
| Internet Address | IP Address |
| Physical Address | MAC Address |
| Type | Protocol Type |



## Figure #25: ARP Request Message

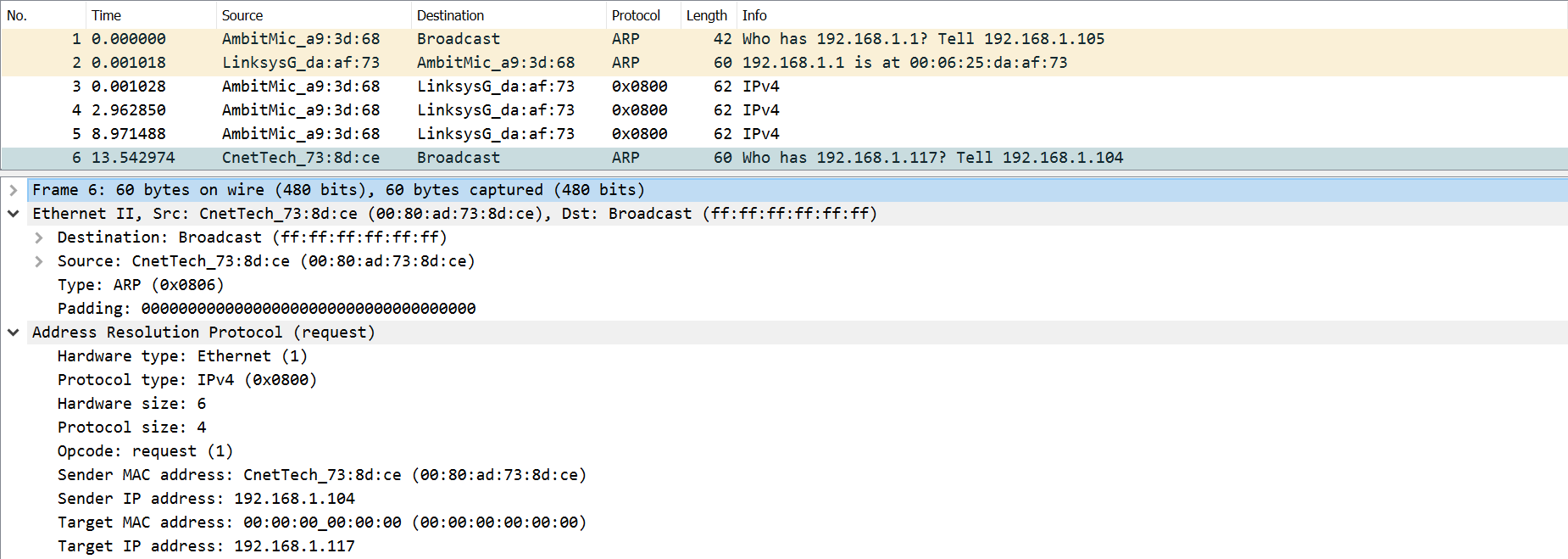
* 1. Source: 00:d0:59:a9:3d:68.
  2. Destination: ff:ff:ff:ff:ff:ff, for broadcast.

1. The hexadecimal value for the Frame Type is 0x806, corresponding to the ARP protocol.
   1. The ARP opcode field begins 20 bytes from the very beginning of the Ethernet frame.
   2. Referring to fig 25, we see that the opcode is 1 (0x0001) for request.
   3. Yes, the ARP message contains the “Sender IP address” field.
   4. In the ARP message, the Target IP address field corresponds to the IP address being queried, and thus the Target MAC address field is set to 00:00:00:00:00:00 to “question” the machine whose IP is the value of the Target IP address field.



## Figure #26: ARP Response Message

* 1. Same answer as question 12 a), being 20 bytes from the very beginning.
  2. Referring to fig 25, we see that the opcode is 2 (0x0002) for reply
  3. The “answer” to the previous query appears in the “Sender MAC address” field.
  4. Source: 00:06:25:da:af:73
  5. Destination: 00:d0:59:a9:3d:68



## Figure #26: Unreplied ARP Request Message

1. There is no reply on our trace because while the ARP request is similar to the DHCP request in that they are both broadcasts, the ARP reply is a unicast reply. As the ARP request was not made by us (different source IP and MAC address), we are unable to capture the ARP reply packet.