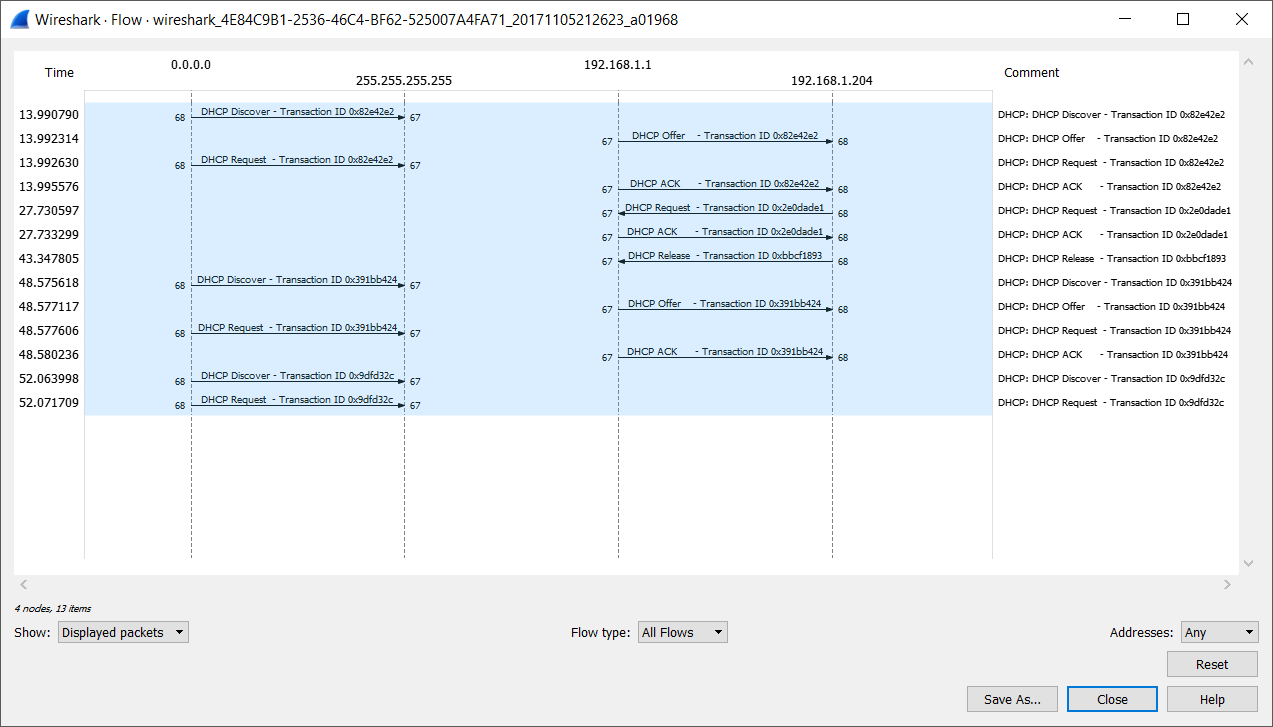
CPE 400/600: Computer Communication Networks

Name: Patrick Austin , Grades \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**HW 3 (Total 20 points)**

**Part 1: DHCP experiment [10]**

1. **[2] Draw a timing datagram illustrating the sequence of the first four-packet Discover/Offer/Request/ACK DHCP exchange between the client and server as captured by wireshark.**

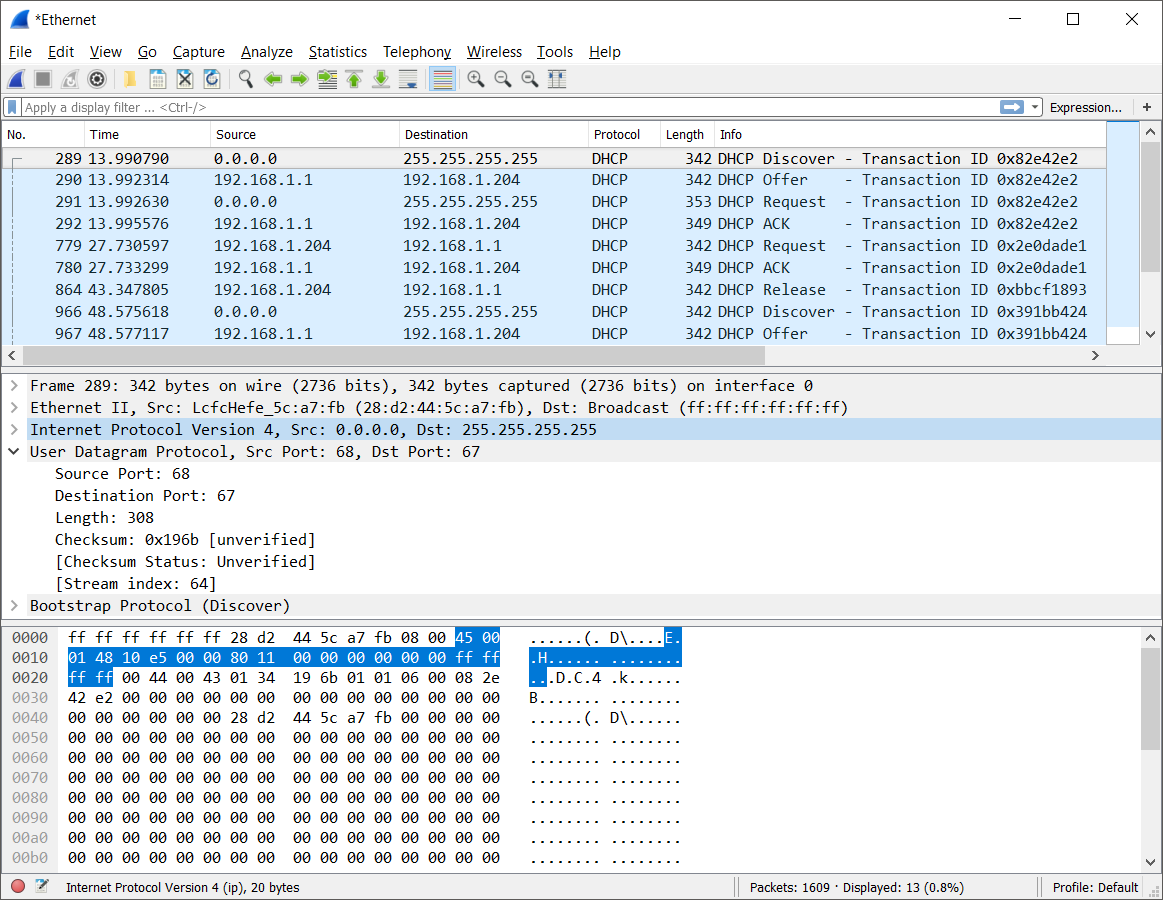


1. **[1] For each packet, indicate the source and destination port numbers. For each of the four DHCP messages (Discover/Offer/Request/ACK DHCP), indicate the source and destination IP addresses that are carried in the encapsulating IP datagram.**

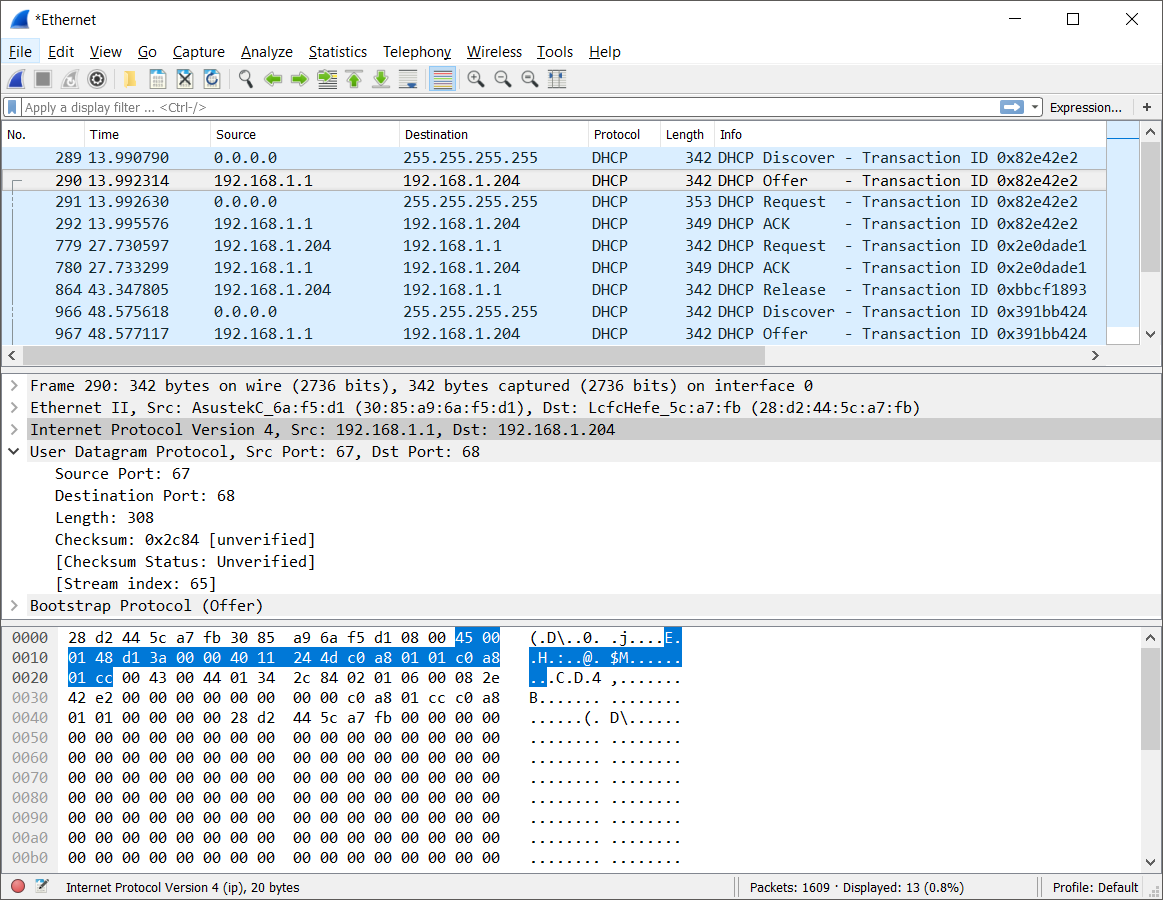
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **#Discover** | **#Offer** | **#Request** | **#ACK DHCP** |
| **Src:** | **68** | **67** | **68** | **67** |
| **Dst:** | **67** | **68** | **67** | **68** |
| **Enc. Src:** | **0.0.0.0** | **192.168.1.1** | **0.0.0.0** | **192.168.1.1** |
| **Enc. Dst:** | **255.255.255.255** | **192.168.1.204** | **255.255.255.255** | **192.168.1.204** |

*Screenshot(s):*

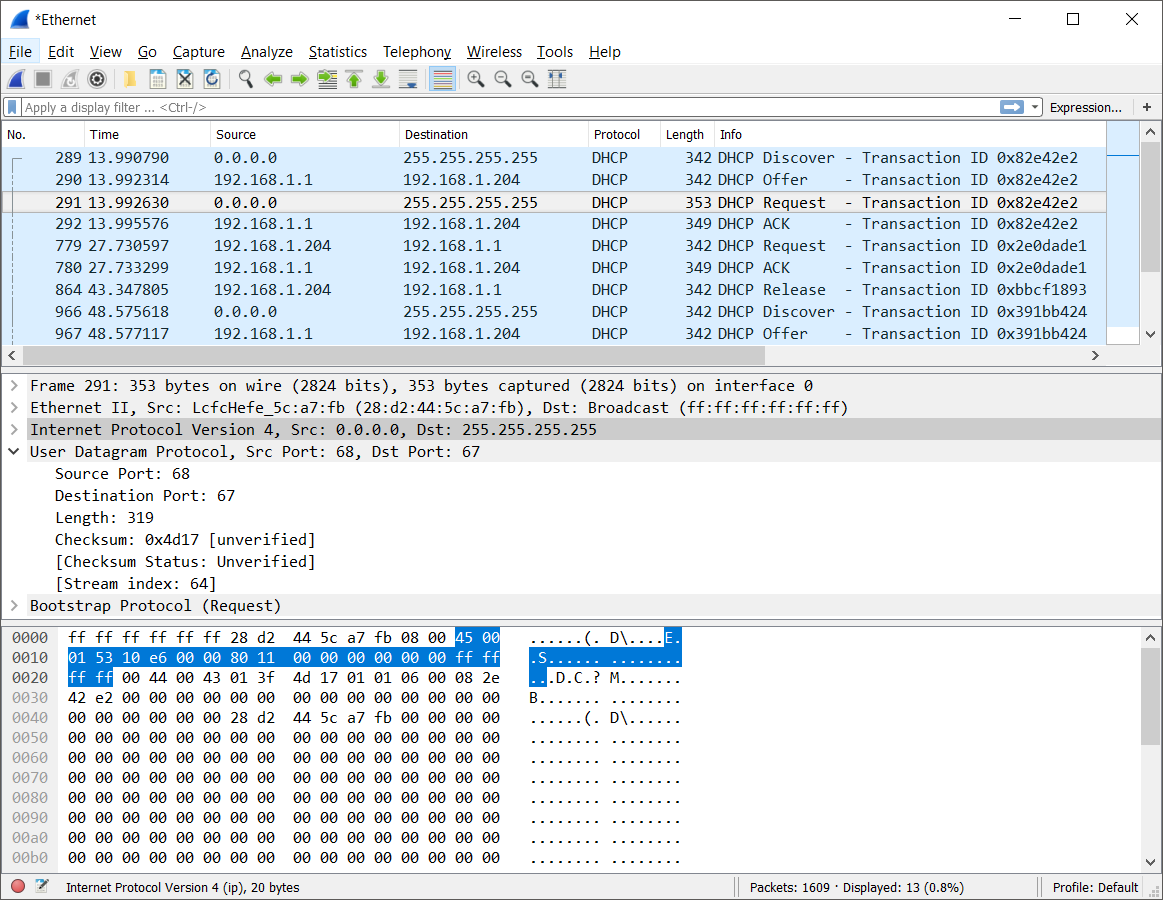
Discover:



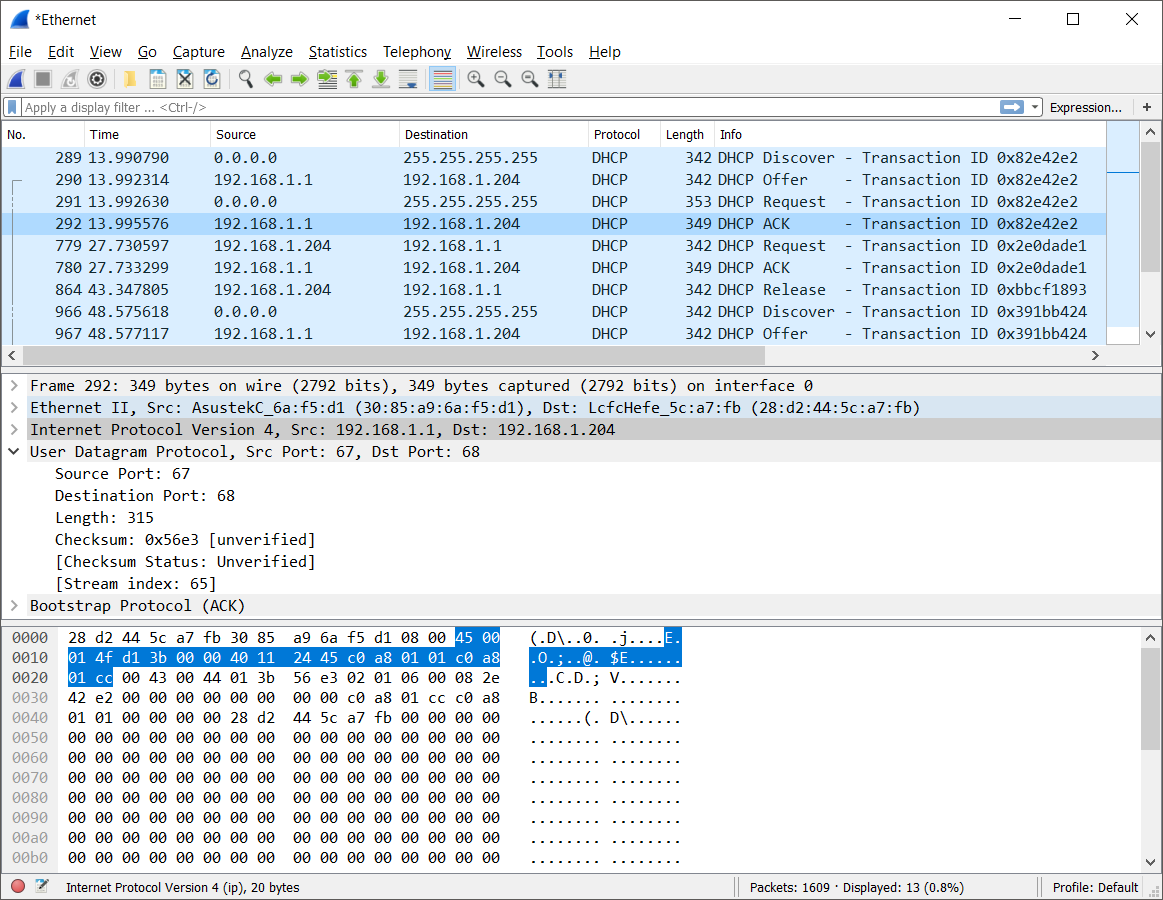
Offer:



Request:



ACK:

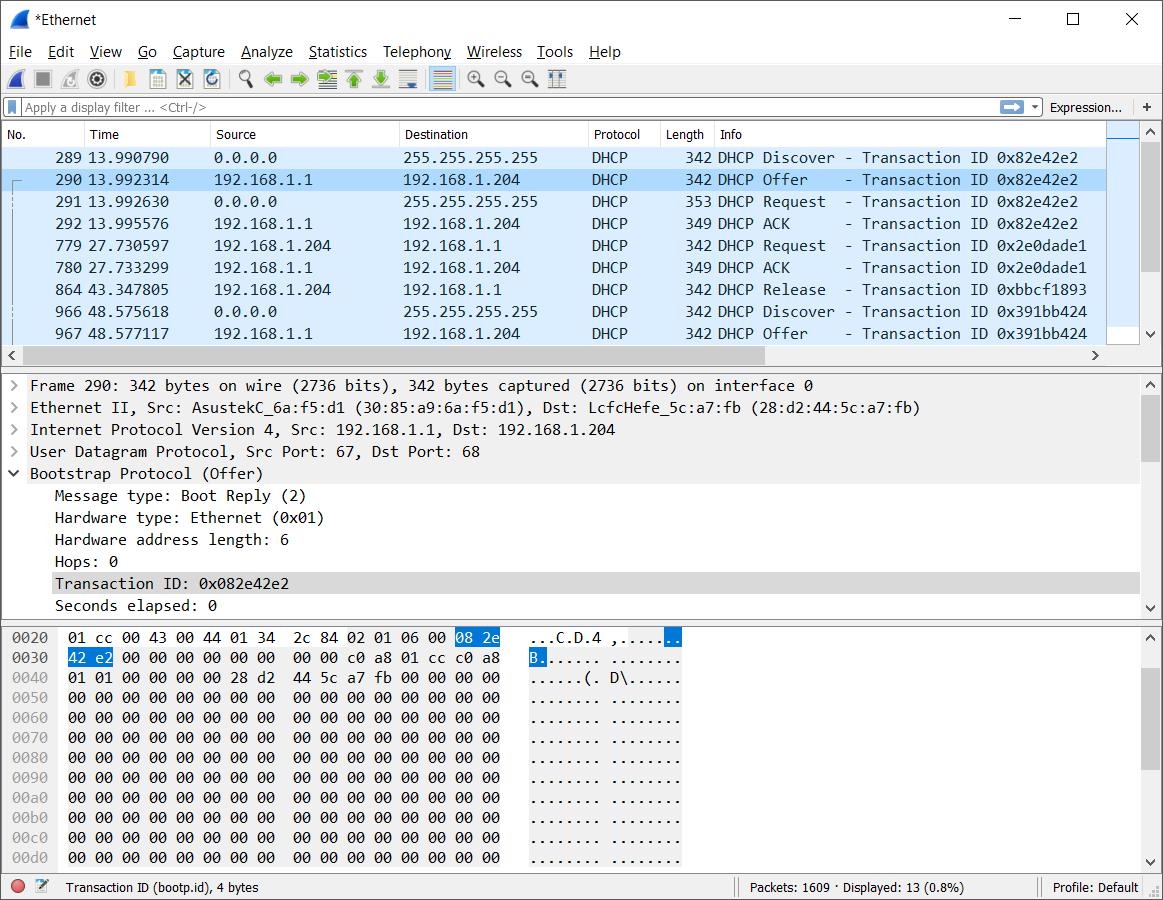
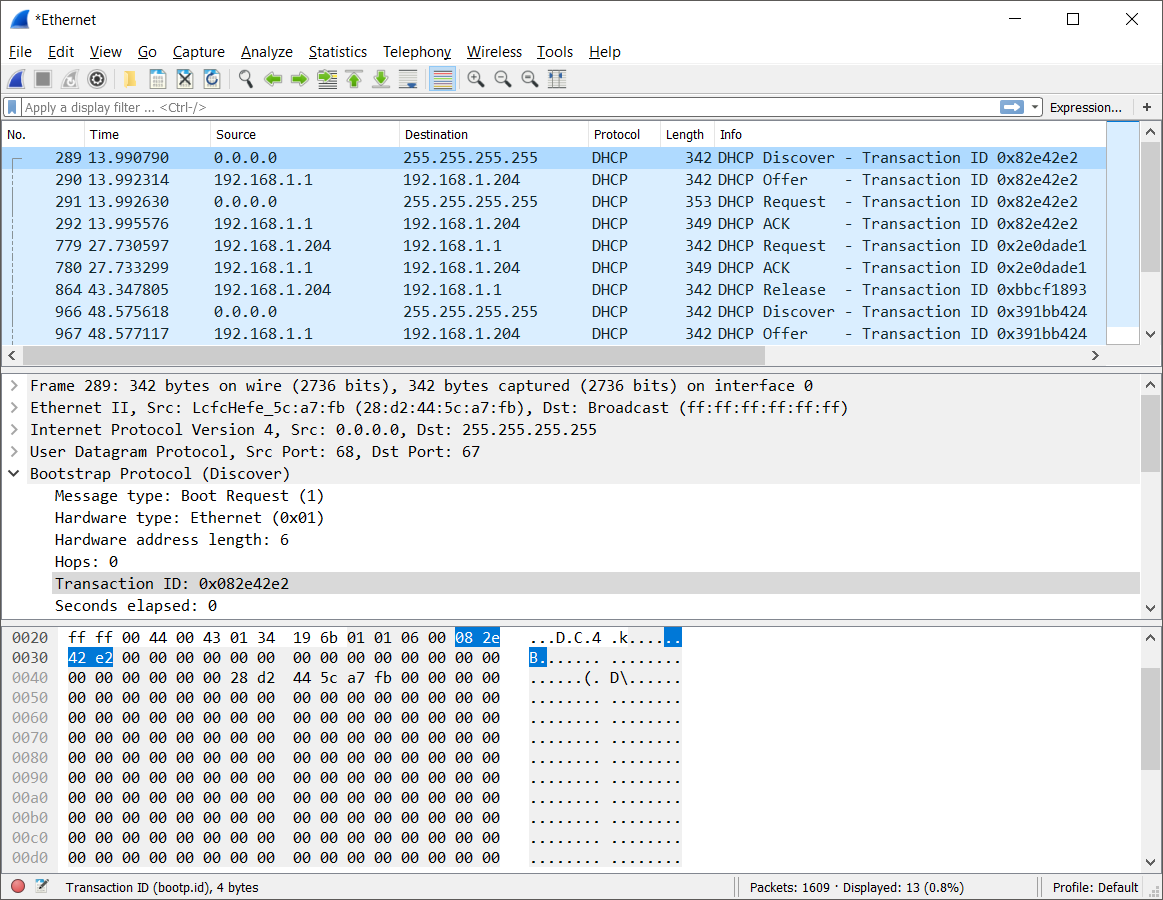


1. **[2] What are the values of the Transaction-ID in each of the first four (Discover/Offer/Request/ACK) DHCP messages?**

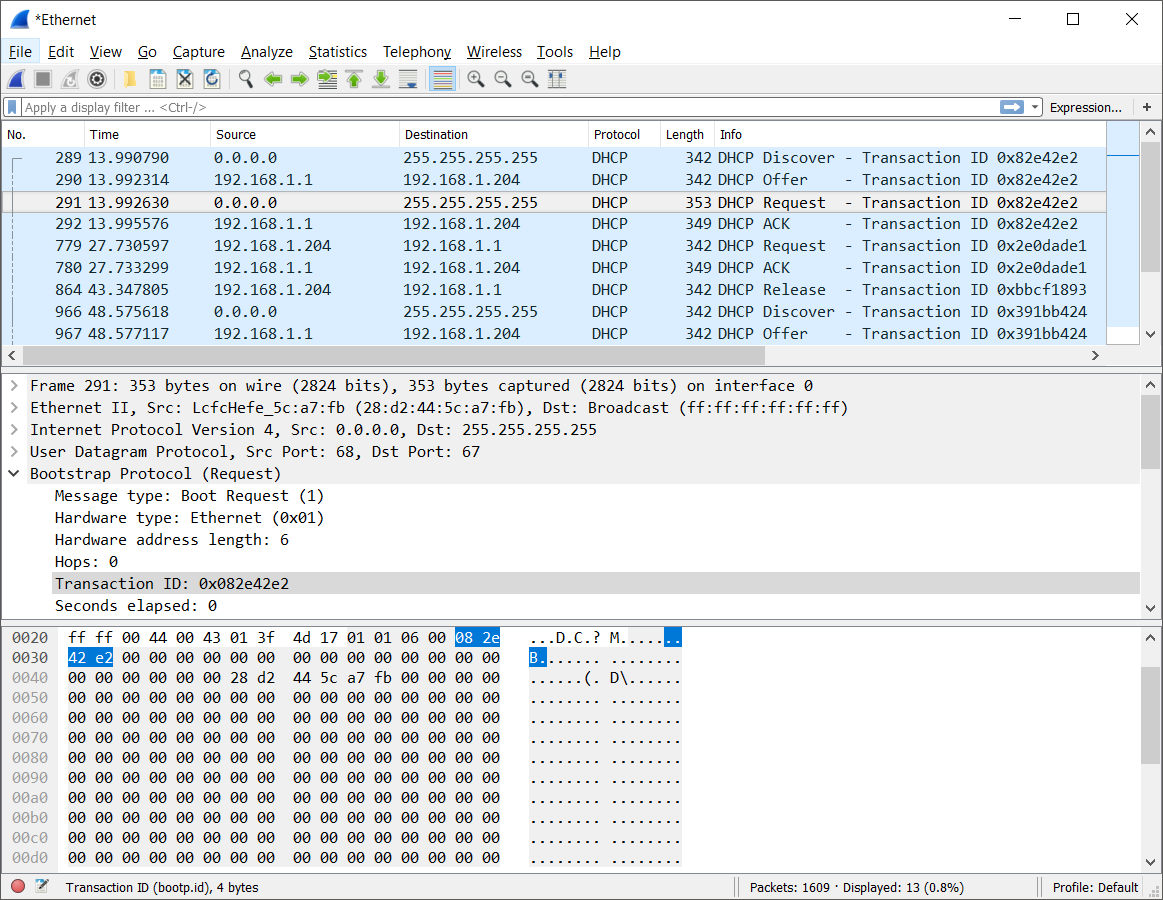
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **#Discover** | **#Offer** | **#Request** | **#ACK DHCP** |
| **T.-ID** | **0x082e42e2** | **0x082e42e2** | **0x082e42e2** | **0x082e42e2** |

*Screenshot(s):*

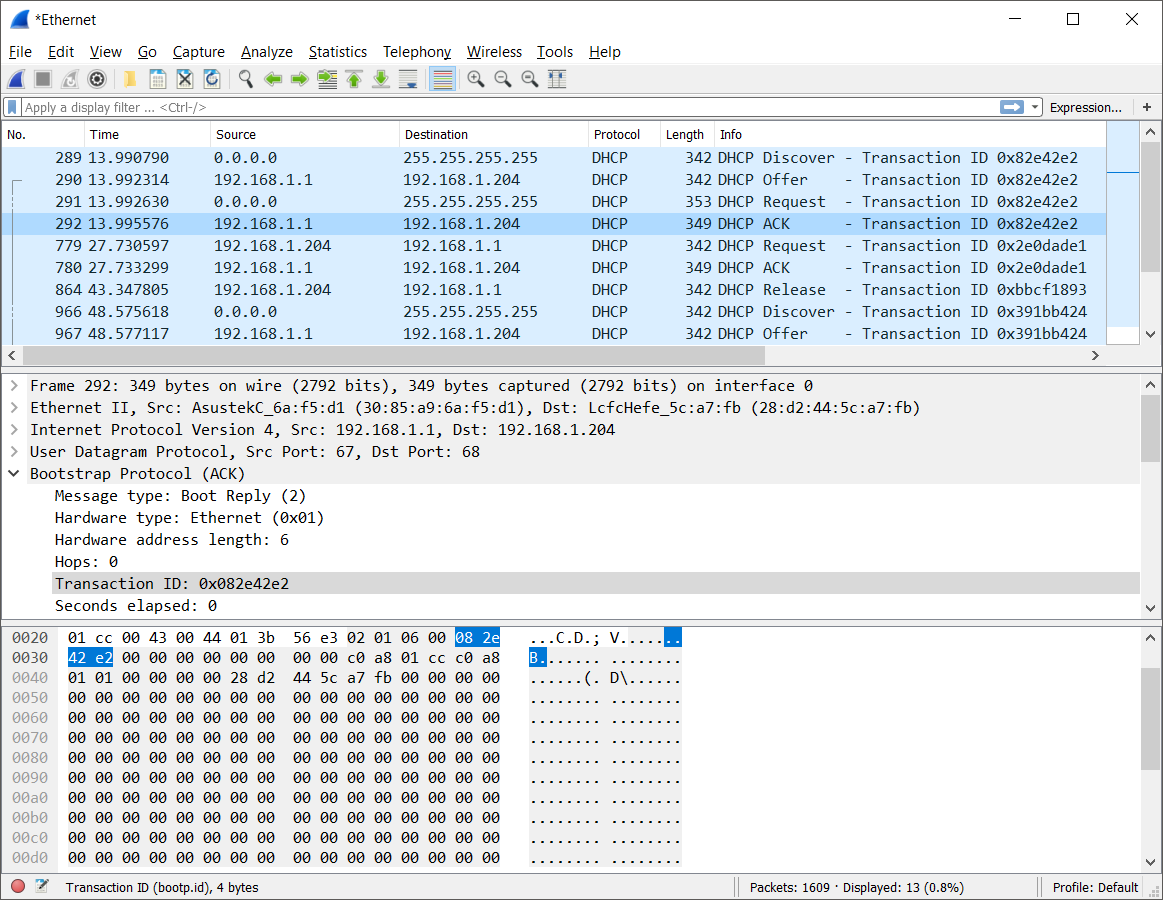
Discover: Offer:



Request:



ACK:



1. **[1] What is the purpose of the Transaction-ID field?**

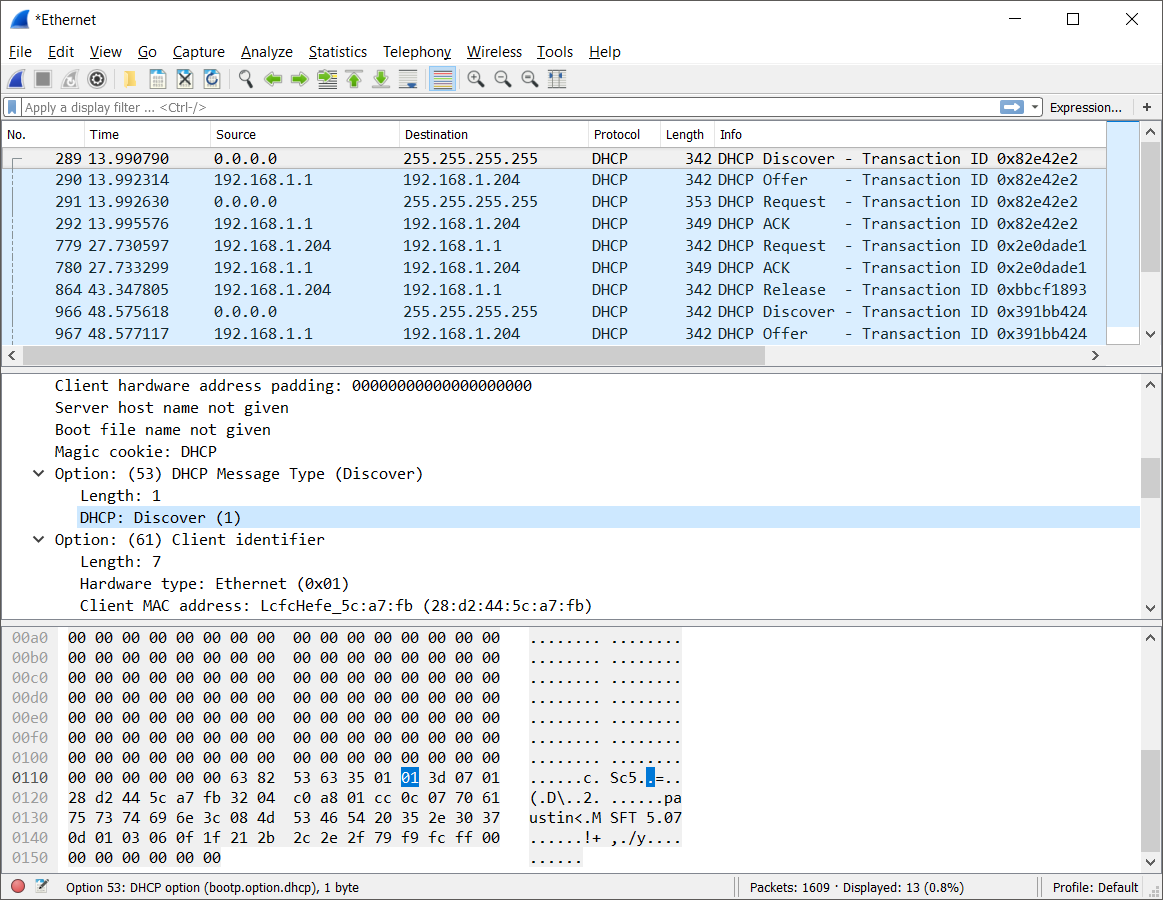
*Answer:* The transaction-ID field is used to specify that a group of DHCP messages are related to a single request or transaction. This allows the client and the server to associate related messages and responses together, and differentiate between multiple transactions.

1. **[1] What values in the DHCP discover message differentiate this message from the DHCP request message?**

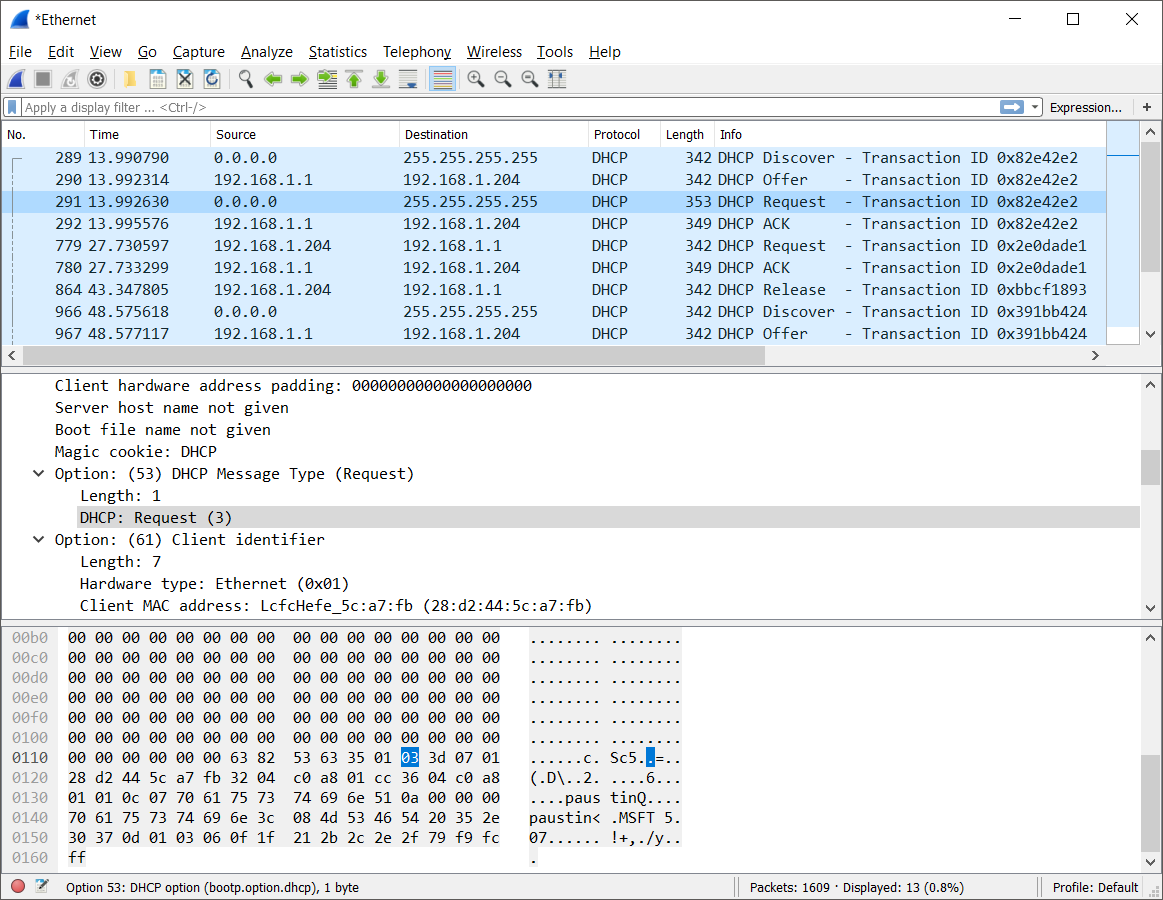
*Answer:* The discover message has a DHCP type value of 1, and the request has a DHCP type value of 3. The messages also differ in terms of the options invoked at the end of the packet.

*Screenshot(s):*

Discover (type 1):



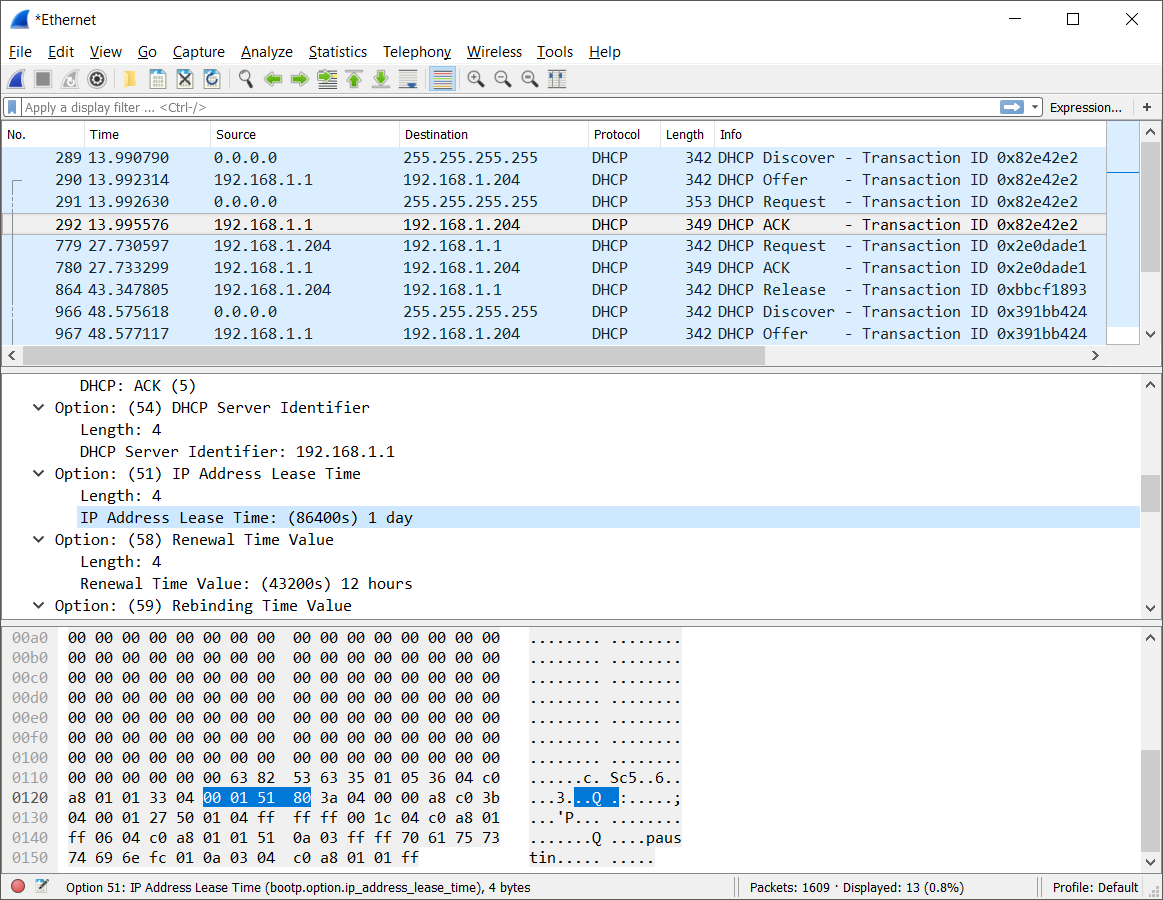
Request (type 3):



1. **[1] Explain the purpose of the lease time. How long is the lease time in your experiment?**

*Answer:* The lease time reflects how long the server is granting the client use of the IP address in question. In this experiment the lease time was 86,400 seconds, or one day, as shown below (next page).

*Screenshot(s):*



1. **[2] What is the purpose of the DHCP release message? Does the DHCP server issue an acknowledgment of receipt of the client’s DHCP request? What would happen if the client’s DHCP release message is lost?**

*Answer 1:* The client sends a DHCP release message to the server when it no longer desires/requires the use of the granted IP address. This way the server can regain control of the IP address, perhaps to be granted to another client.

*Answer 2:* No ACK is sent; no ACK to the release was picked up in Wireshark.

*Answer 3:* If the release message is lost, the client behaves as it would have otherwise, not using the released IP. Instead of recognizing that the IP has returned to the stock of available IPs that can be granted as it would have if the release arrived, the server will not reclaim the IP until the lease time expires.

**Part 2: IP [10]**

1. **[2] 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.**

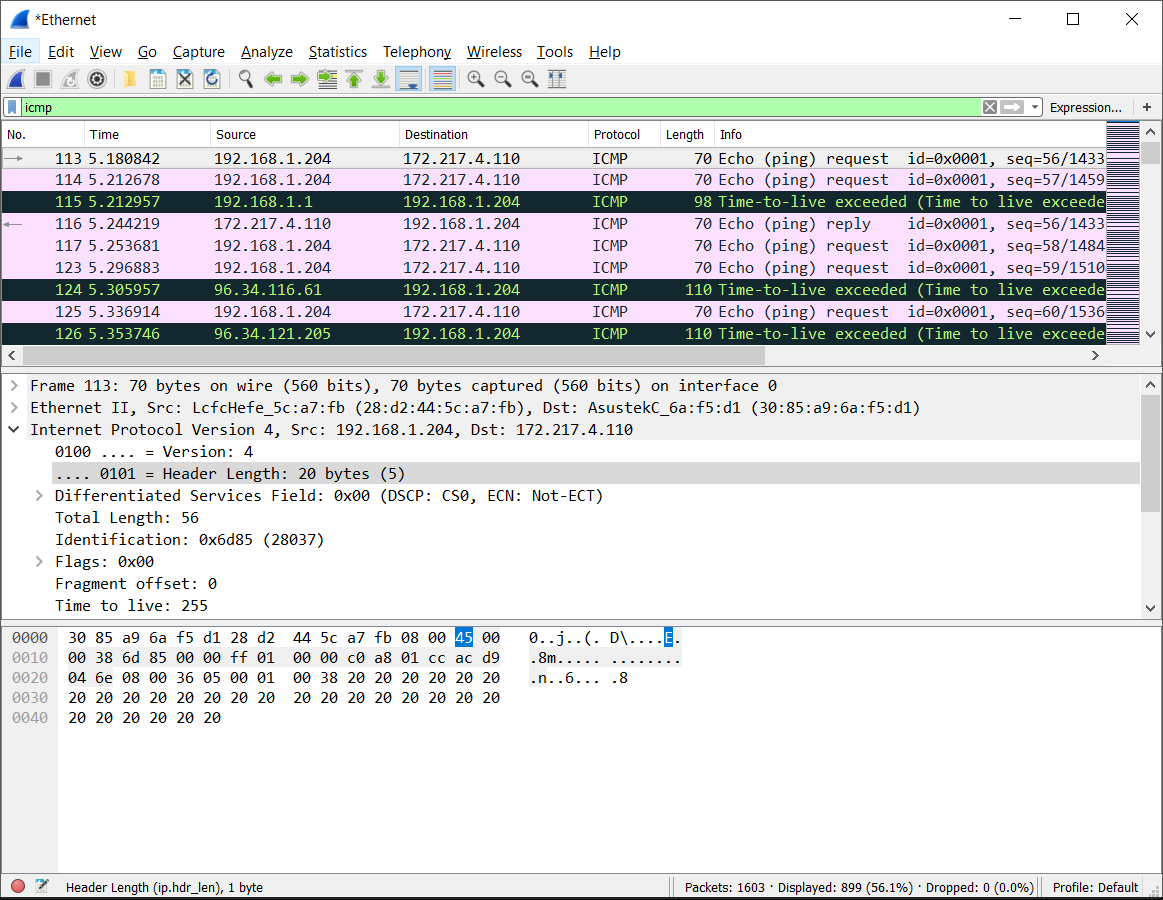
*Answer 1:* The headers have 20 bytes, as show below.

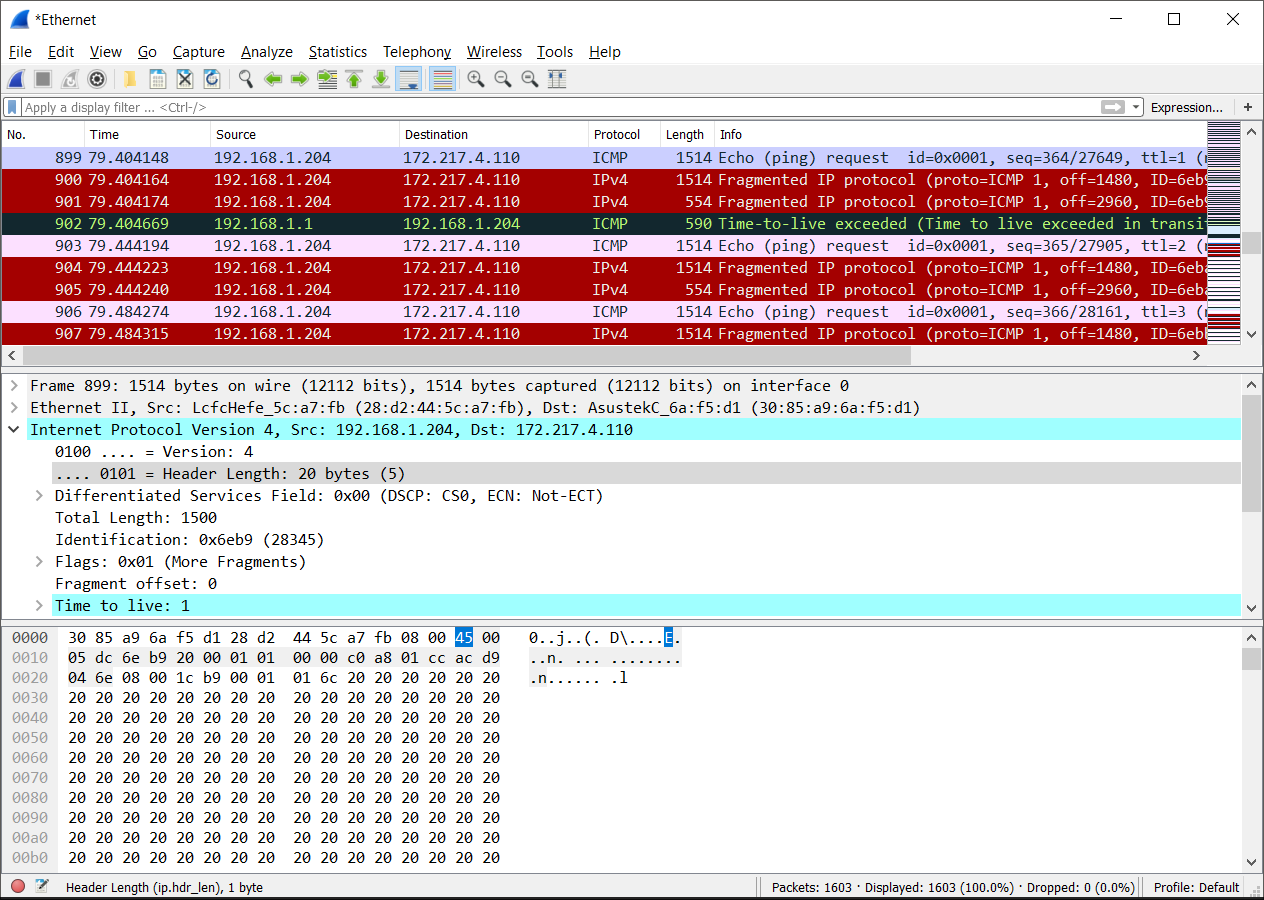
*Answer 2:* The payload has 36 bytes, 980 bytes, and 1480/1480/520 bytes (due to fragmentation) respectively.

*Answer 3:* The total size of the IP datagrams are 56/1000/1500-1500-540 bytes as shown below and there are 20 bytes of header, so the payload is 56-20=36, 1000-20=980, and -1500-20=1480-1480-540-20=520 bytes.

*Screenshot(s):*

IPv4 header length is shown, as well as the total length of the packet. See also the fragment offset, for use in the next problem.





1. **[2] Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.**

*Answer 1:* The first two datagrams are not fragmented and the third is.

*Answer 2:* The first two datagrams have the More Fragments flag down. The first two pieces of the third datagram have it up, and the third piece has a fragmentation offset value. Also, the third datagram is 1500/1400/540 bytes for a 3500 bit transmission, so fragmentation must have occurred, splitting the original message into multiple packets.

*Screenshot(s):* See the More Fragments flags in the screenshots for the previous problem.

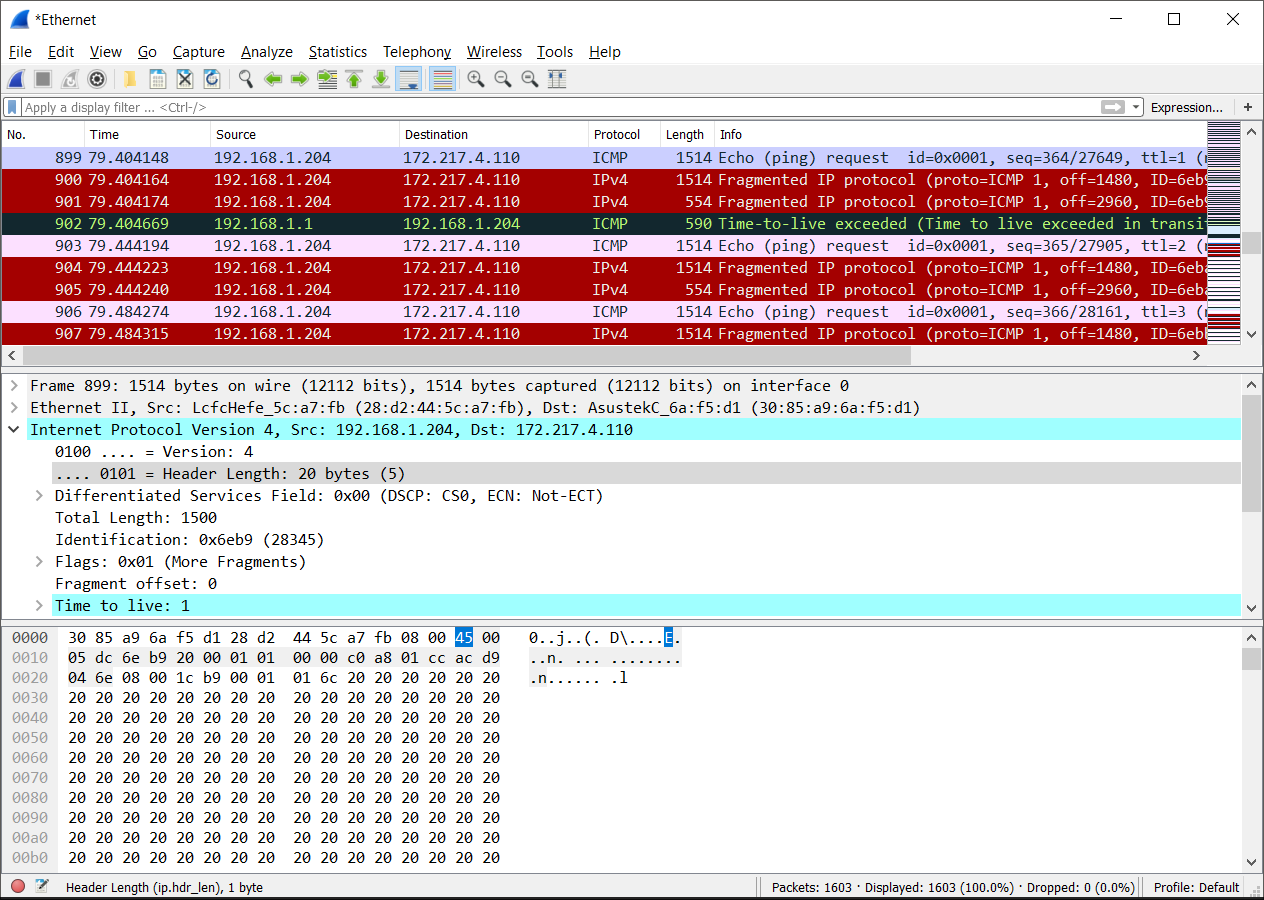
1. **[3] If fragmented, show the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?**

*Answer 1:* The More Fragments flag indicates that the datagram has been fragmented.

*Answer 2:* The fragment offset being 0 on this datagram indicates that this is the first fragment and not a later one, which will have a non-zero offset value.

*Answer 3:* This datagram is 1500 bytes long- 1480 of payload and 20 of header.

*Screenshot(s):*



1. **[3] Show the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are there more fragments? Explain.**

*Answer 1:* The fragment offset value of 1480 denotes that this is not the first datagram fragment- 1480 bytes of the packet have been sent already and should come first.

*Answer 2:* More fragments are to come since the More Fragments flag is up. Indeed, Wireshark shows a third fragment is needed to complete the datagram. Both are shown below.

*Screenshot(s):*

