Student Name: Gökay Gülsoy
Student Number: 270201072

-CENG312-ASSIGNMENT 3 IP, ICMP, NAT, DHCP-

1.)

a.) When we look at the packet traffic “Fragmentation needed” ICMP messages are triggered in case amount of data that should be carried by datagram exceeds the maximum amount of data can be carried within the datagram.

b.) When endpoints receive the “Fragmentation needed” message endpoints can divide or in other words fragment datagram into smaller IPv4 packets. To identify fragements 16-bit field is used as an identification field and this identification field is copied to fragmented packet header so that on the destination side receiver will be able to reassemble the fragments that belong to a particular frame. In the packet traffic there is a problem related to fragmentation, because even though endpoint with IP address 62.177.254.141 has received ICMP message “Destination unreachable (Fragmentation needed)” three times it was not able to perform fragmentation. (Figure 1 and 2)

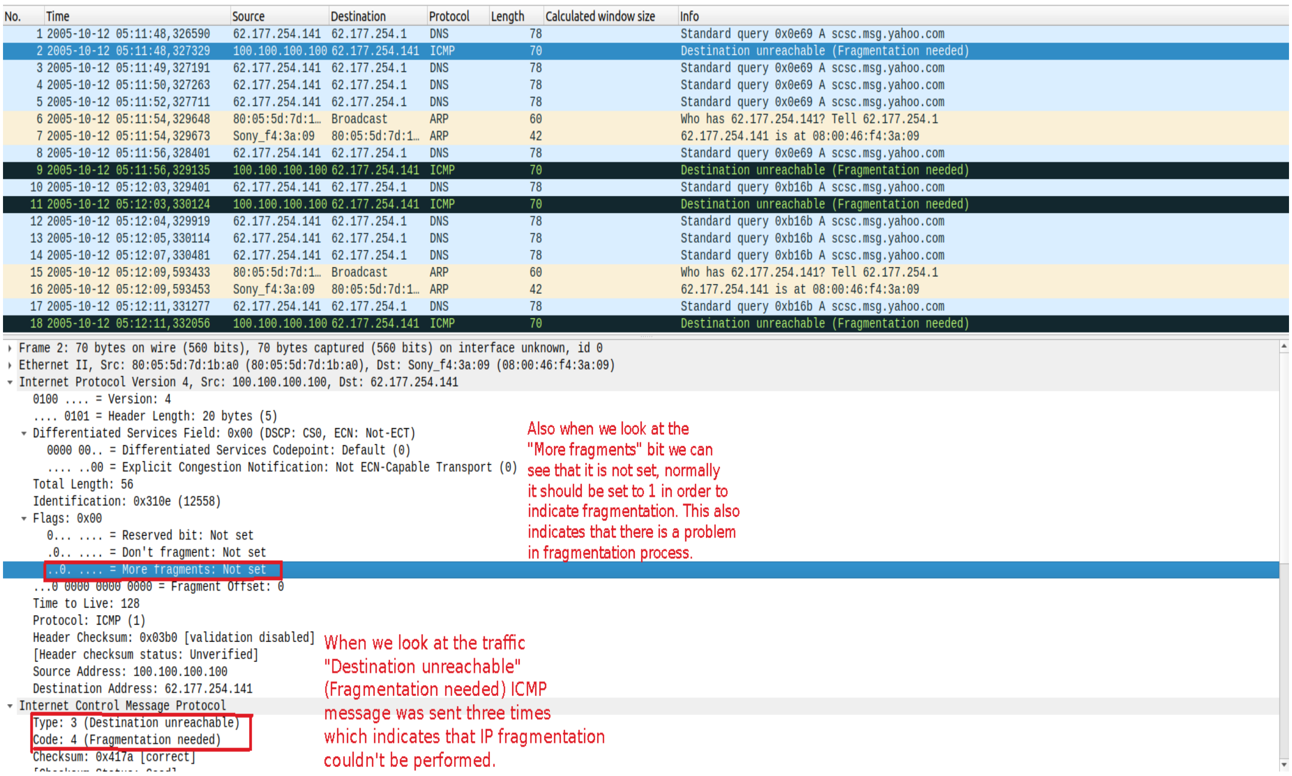


Figure 1: Fragmentation problem in the traffic

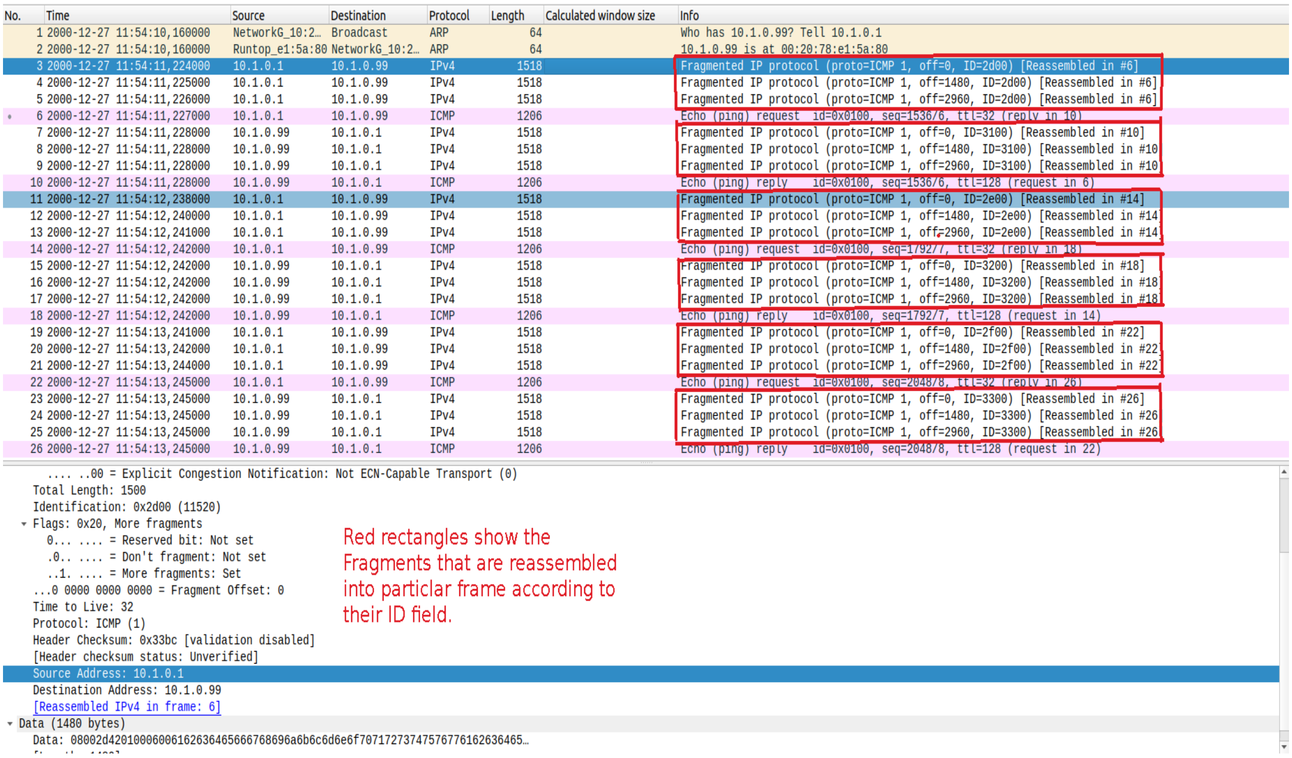


Figure 2: Reassembly operation on fragments is done according to id fields

2.)

a.) In the packet capture Info column, there is an information saying that "Fragmented IP protocol" with fragment's offset number. (Figure 3)

b.) Purpose of IP fragmentation is to divide large datagrams into smaller packets which does not fit into a single datagram previously. Later, fragments will be reassembled at the destination according to identification fields.

c.) When we look at the line numbers 3,4, and 5 fragments with the ID=2d00 are all reassembled into number #6 frame. In the lines 7,8, and 9 fragments with ID=3100 are all reassembled into number #10 frame. In the lines 11,12, and 13 fragments with ID=2e00 are all reassembled into number #14 frame. In the lines 15,16, and 17 fragments with ID=3200 are all reassembled into number #18 frame. In the lines 19,20, and 21 fragments with ID=2f00 are all reassembled into number #22

(Figure 3 and Figure 4)

frame. In the lines 23,24, and 25 with ID=3300 are all reassembled into number #26.

c.) If we look at the line numbers 3,4, and 5, fragments which have ID=2d00 belong to to frame with number #6. Each Fragment has a total length of 1500, so total length is 4500. We can also compute the original packet size by using offset values. First packet's offset is 0 and second packet's offset is 1480, so data length of first packet should be 1480. Second packet offset is 1480 and third packet offset is 2960, so data length of second packet is 1480. Third packet also has the data length 1480, so total data length is 4440 and also each fragment has 20 byte header field so total 60 bytes for headers. Indeed 4440+60 = 4500 is the original IP packet length which conforms to total length information in Wireshark bottom panel. (Figure 4 and Figure 6)

<u>d.)</u> For the fragments whose ID's are 2d00, 2e00, and 2f00 Time to live field is 32 whereas, fragments whose ID's are 3100,3200, and 3300 have Time to live field value 128. So TTL value is different for some fragments. Reason that's why TTL value for some fragments is longer than others can be attributed to importance of fragment, may be some fragments some fragments are given more importance due to role of data they carry. (Figure 5)

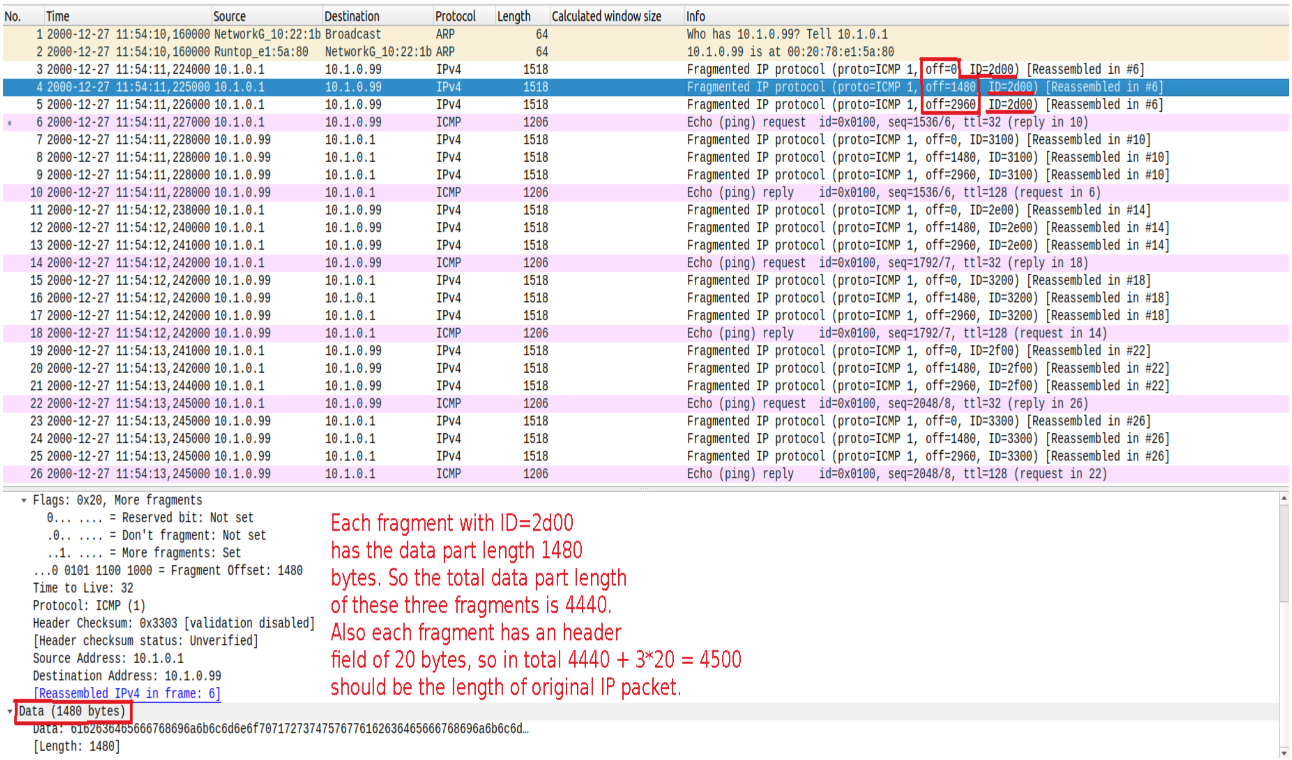


Figure 3: Calculating original packet length with help of offset information

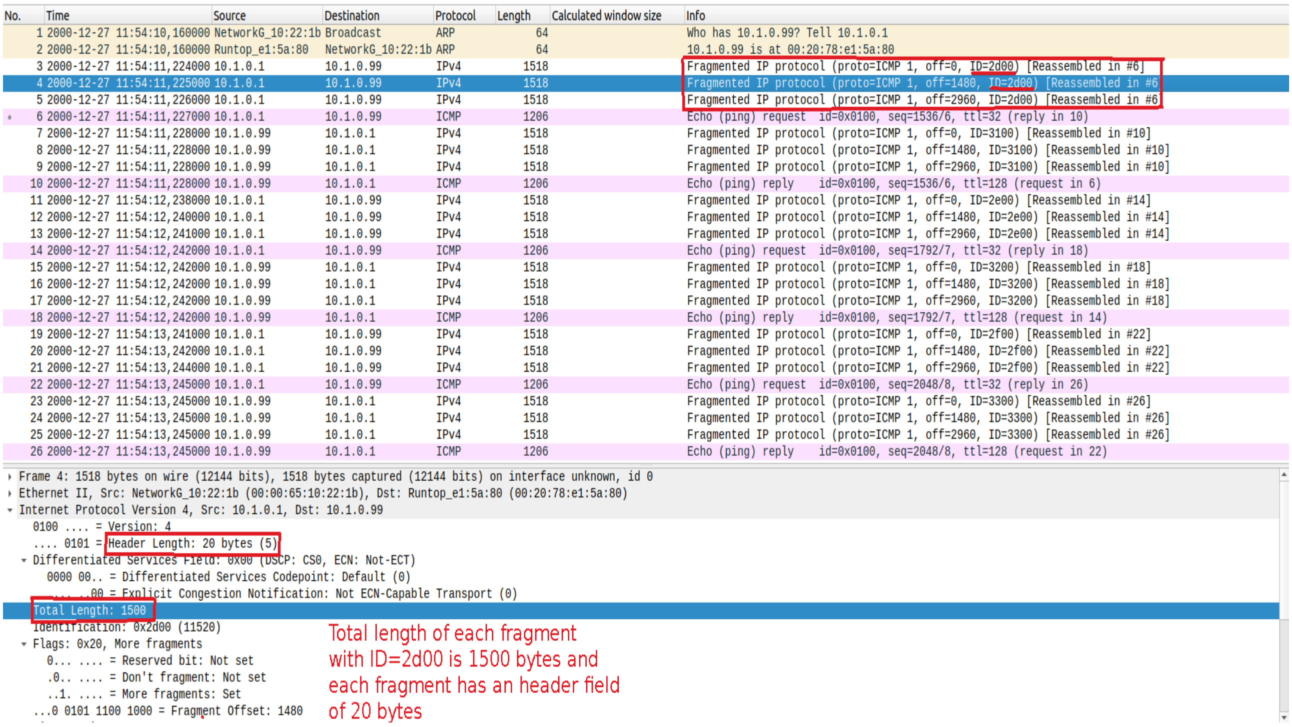


Figure 4: Total length and header length fileds of fragments

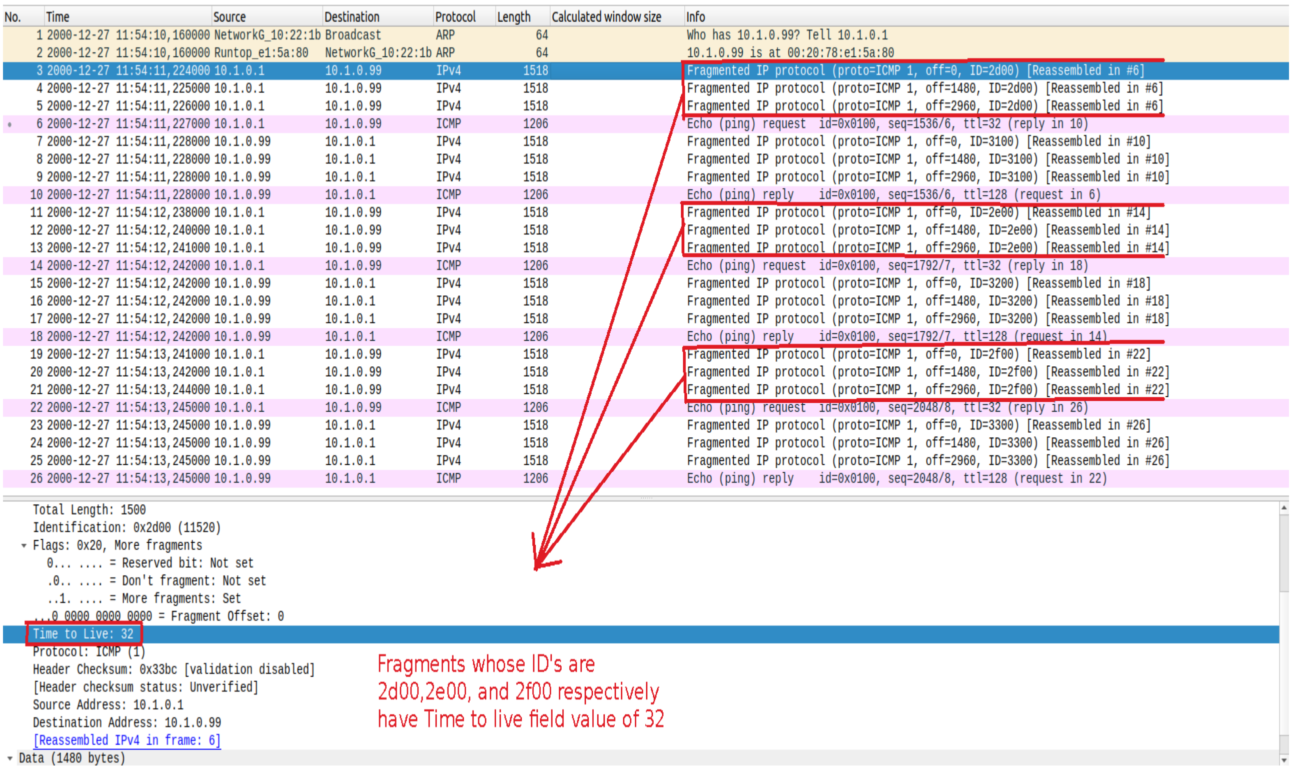


Figure 5: Time to live fields of fragments with ID's 2d00,2e00,2f00 is 32

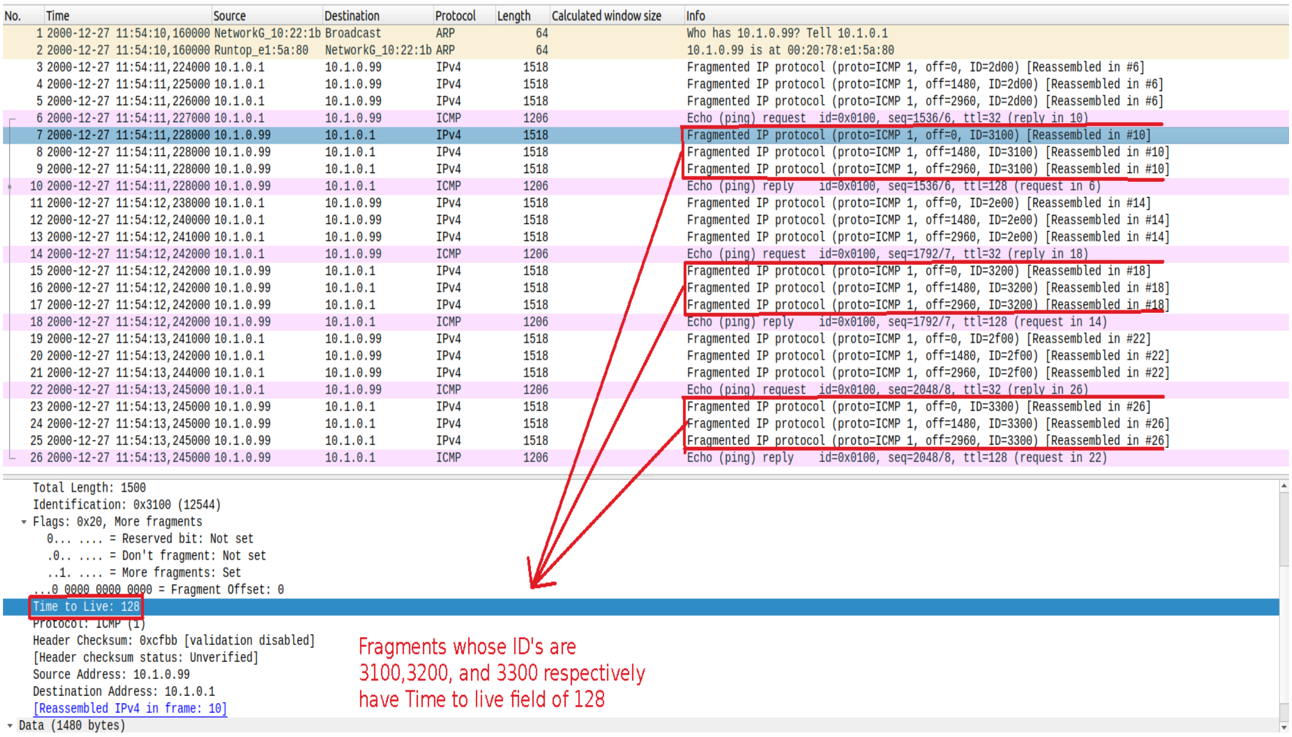


Figure 6: Time to live fields of fragments with ID's 3100,3200,3300 is 128

3.) If we look at the line number 8 we can see in the packet comments section that "IPv6 packets are encapsulated in IPv4" this is the method of tunelling and encapsulation where IPv6 datagrams are embedded inside IPv4 datagram to sent to destination address. Also when we look at the Internet protocol version 4 part below the packet capture table we can see that Protocol field is also IPv6.

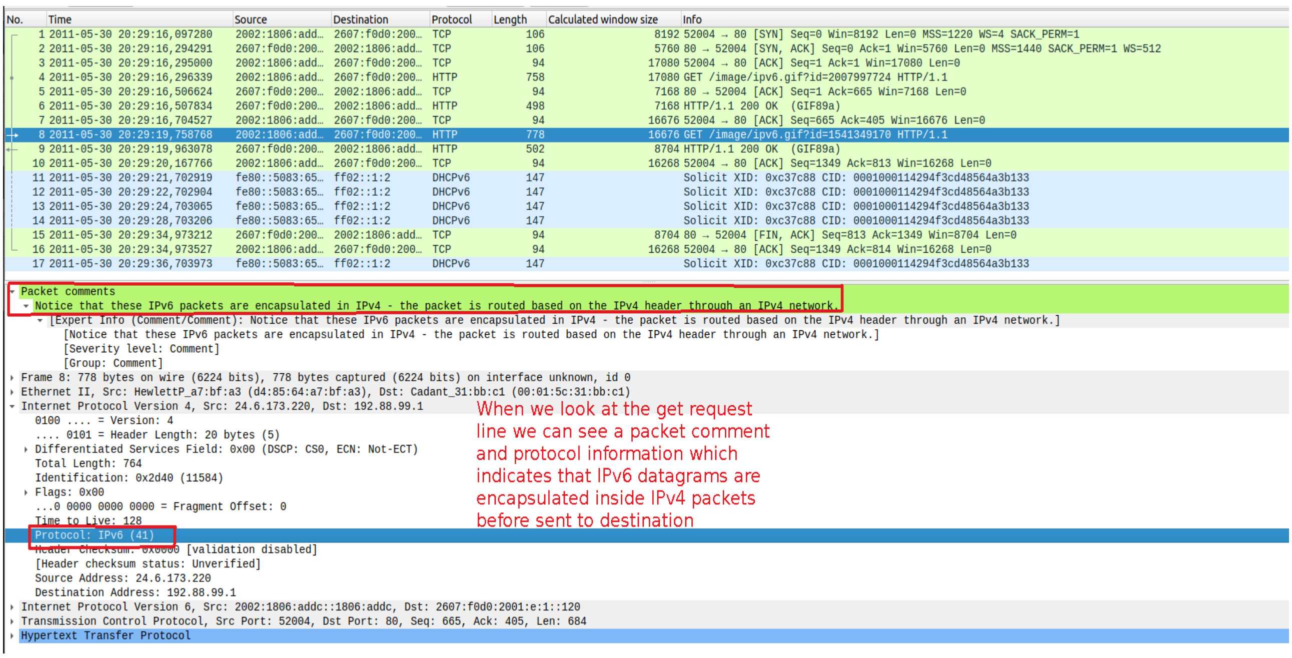


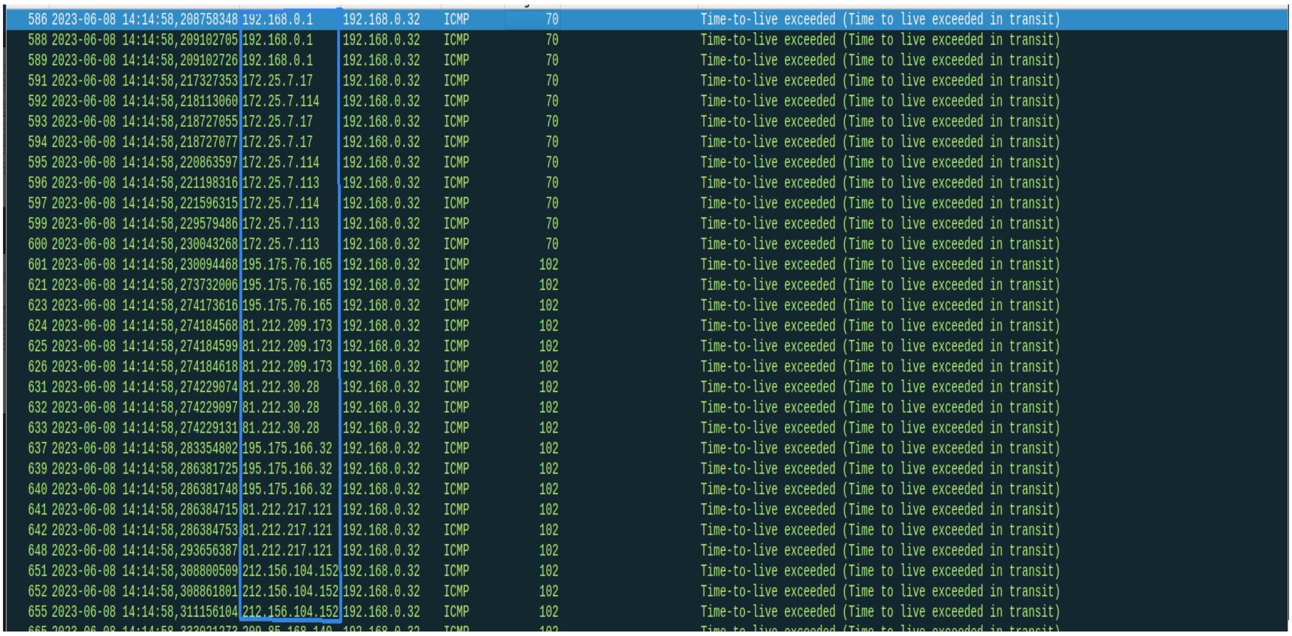
Figure 7: IPv6 packets encapsulated inside IPv4 datagrams

4.)

a.) We can observe two ICMP message types as a result of executing traceroute command from terminal. First one is Type 11 which gives the time exceeded message when a router receives a datagram with TTL of 0 or 1. So by using this TTL value infinite routing loops are prevented. Routers can not forward a datagram that has a TTL of 0 or 1 and packet is dropped. ICMP message of code 0 indicates Time-to-live became equal to 0 during transit not during reassembly. Second is Type 3 ICMP message which indicates that the router cannot find the destination network. Code 3 which says “port uncreachable” indicates that trasport layer protocol we are attempting to communicate is not available on the arriving side.

b.) As a result of executing traceroute google.com command, intermediate network devices are same in both cases because when we look at the hop IP addresses, in the Wireshark and in the output of traceroute they are same. It is intuitive to get same intermediate device IP addresses because we have captured packets during the same traceroute execution. (Figure 10)

c.) Routers with IP addresses given inside blue rectangle in the screenshot below have have been sent 3 packets. Routers with the IP addresses given inside the red rectangle have been sent 1 packet. (Figure 8 and Figure 9)



586	2023-06-08 14:14:58,208758348	192.168.0.1	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
588	2023-06-08 14:14:58,209102705	192.168.0.1	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
589	2023-06-08 14:14:58,209102726	192.168.0.1	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
591	2023-06-08 14:14:58,217327353	172.25.7.17	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
592	2023-06-08 14:14:58,218113060	172.25.7.114	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
593	2023-06-08 14:14:58,218727055	172.25.7.17	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
594	2023-06-08 14:14:58,218727077	172.25.7.17	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
595	2023-06-08 14:14:58,220863597	172.25.7.114	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
596	2023-06-08 14:14:58,221198316	172.25.7.113	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
597	2023-06-08 14:14:58,221596315	172.25.7.114	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
599	2023-06-08 14:14:58,229579486	172.25.7.113	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
600	2023-06-08 14:14:58,230043268	172.25.7.113	192.168.0.32	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
601	2023-06-08 14:14:58,230094468	195.175.76.165	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
621	2023-06-08 14:14:58,273732006	195.175.76.165	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
623	2023-06-08 14:14:58,274173616	195.175.76.165	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
624	2023-06-08 14:14:58,274184568	81.212.209.173	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
625	2023-06-08 14:14:58,274184599	81.212.209.173	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
626	2023-06-08 14:14:58,274184618	81.212.209.173	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
631	2023-06-08 14:14:58,274229074	81.212.30.28	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
632	2023-06-08 14:14:58,274229097	81.212.30.28	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
633	2023-06-08 14:14:58,274229131	81.212.30.28	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
637	2023-06-08 14:14:58,283354802	195.175.166.32	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
639	2023-06-08 14:14:58,286381725	195.175.166.32	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
640	2023-06-08 14:14:58,286381748	195.175.166.32	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
641	2023-06-08 14:14:58,286384715	81.212.217.121	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
642	2023-06-08 14:14:58,286384753	81.212.217.121	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
648	2023-06-08 14:14:58,293656387	81.212.217.121	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
651	2023-06-08 14:14:58,308800509	212.156.104.152	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
652	2023-06-08 14:14:58,308861801	212.156.104.152	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
655	2023-06-08 14:14:58,311156104	212.156.104.152	192.168.0.32	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

Figure 8: Routers with IP addresses given inside blue rectangle have been sent 3 packets

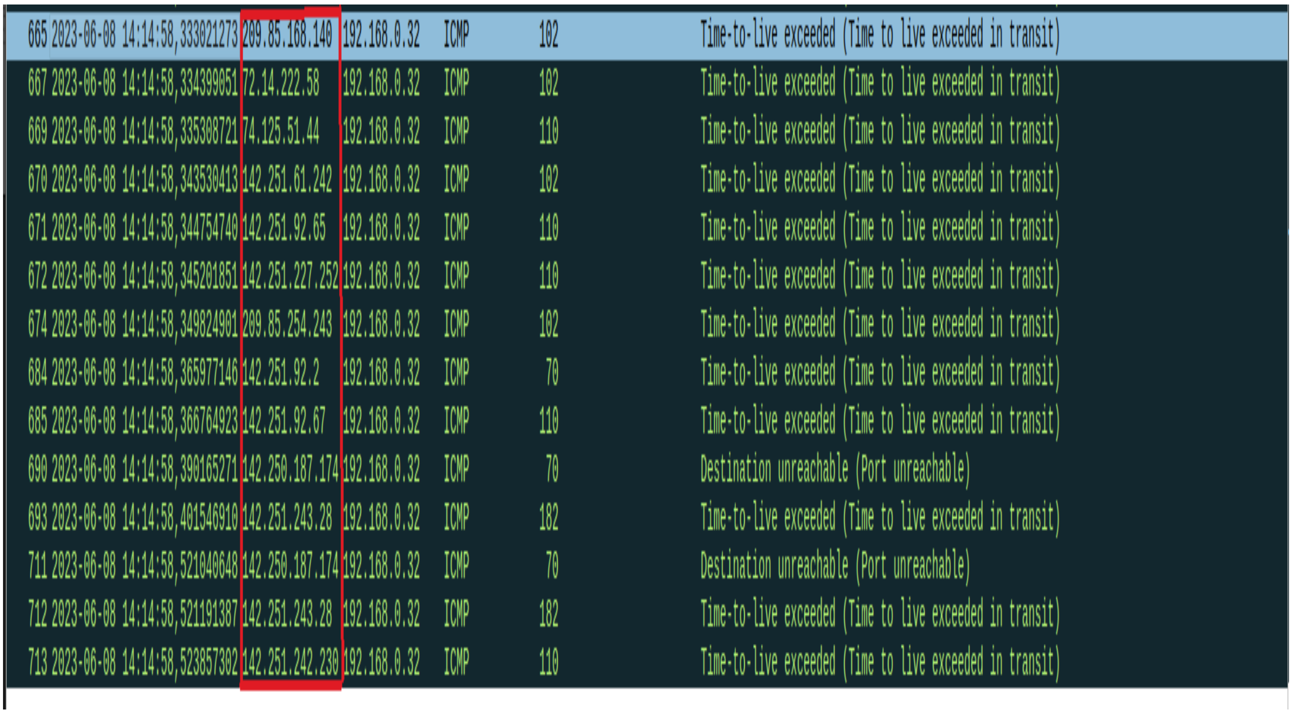


Figure 9: Routers with IP address given inside the red rectangle have been sent 1 packet

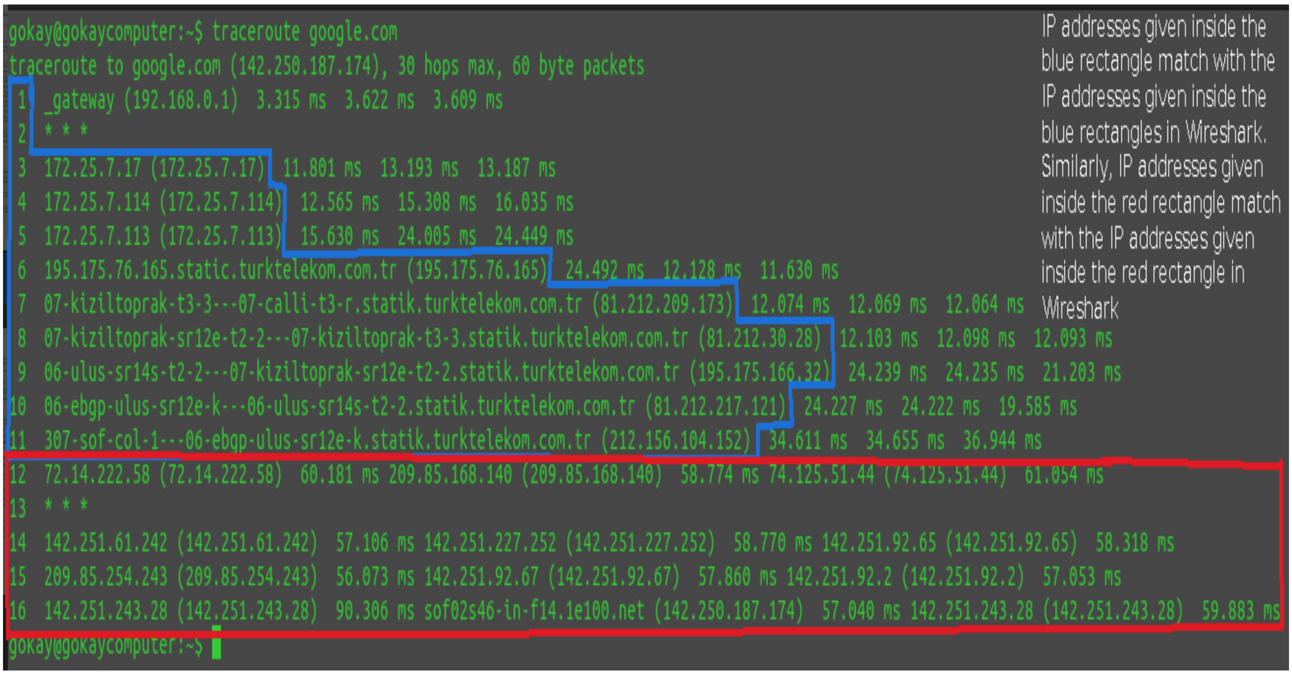


Figure 10: traceroute command output and the hops along with their IP addresses and packets they received

NAT SECTION

1.) NAT protocol stands for Network address translation which allows all devices inside local area network to share just one IP address as far as outer networks are considered , so all datagrams leaving from local network share same NAT IP address, but they are distinguished by different port numbers. The mapping between datagram source IP address and NAT IP address, is implemented in the form of NAT table. NAT protocol also helps IPv4 address exhaustion because just one IP address is needed from ISP for all devices in local network.

2.) When we look at the lan-packets screenshot below, scenario might be as follows: host with IP address 10.142.154.239 and destination with IP address 8.8.8.8 are located in the same local area network and host is trying to establish a TCP connection with server whose IP address is 8.8.8.8. When we look at the wan-packets screenshot below, scenario might be as follows: host with IP address 136.102.83.11 and destination with IP address 8.8.8.8 are located in different networks and host is trying to establish a TCP connection with server whose IP address is 8.8.8.8. This scenarios can be related to NAT protocol as follows: Actual IP address of host may be 10.142.154.239 inside it's local area network but let's say as in WAN scenario server with IP address 8.8.8.8 is located outside the LAN and any host when connecting to any server in LAN is using IP address 10.142.154.139, but when server for which host is going to establish connection is outside the LAN, there is a NAT router which translates IP address of host from 10.142.154.239 to 136.102.83.11. Thus when server sends response to host it sees it's NAT IP which is 136.102.83.11 not it's actual local network IP address which is 10.142.154.239. (Figure 11 and Figure 12)

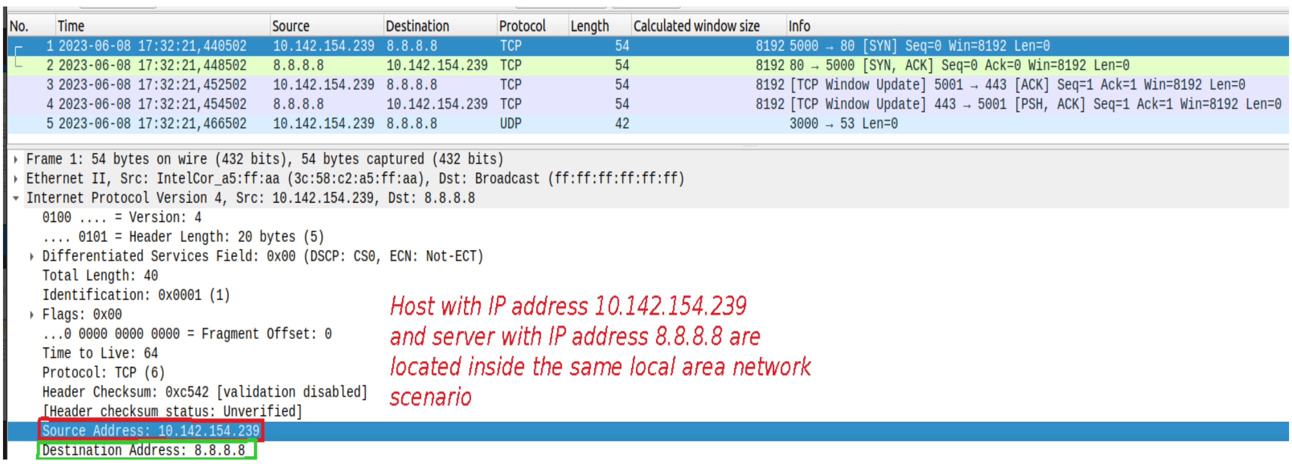


Figure 11: Scenario in which host with IP address 10.142.154.239 and destination with IP 8.8.8.8 in same LAN

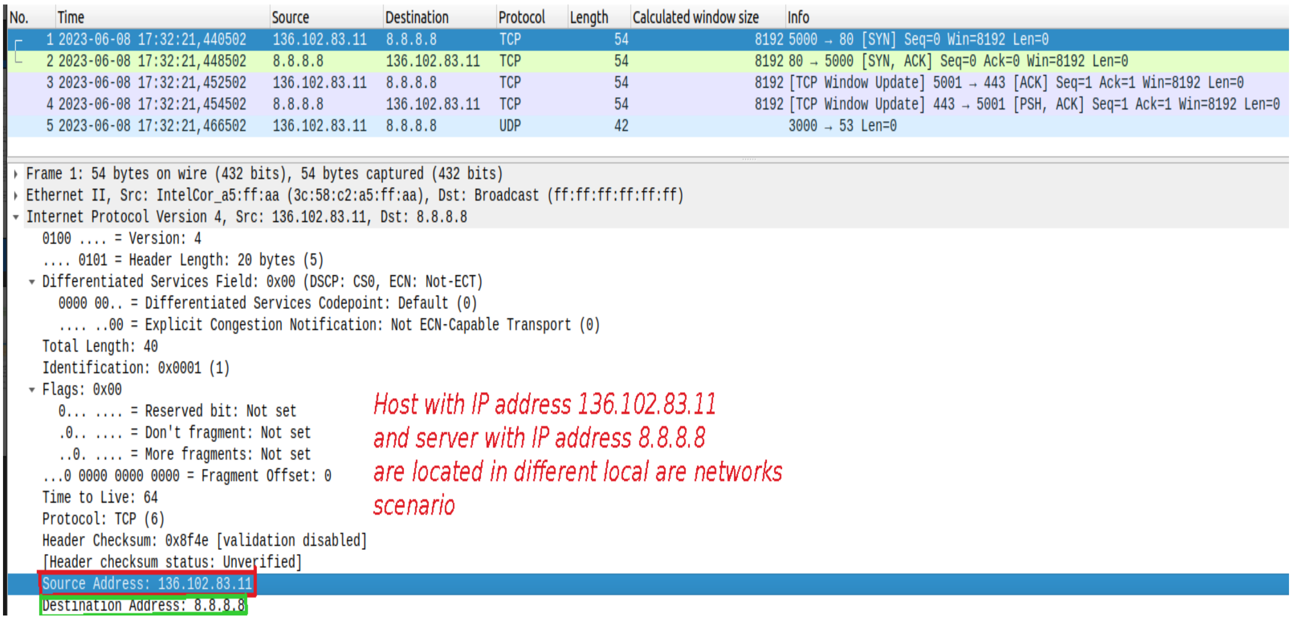


Figure 12: Scenario in which host with IP address 136.102.83.11 and destination with IP 8.8.8.8 in different LANs

DHCP SECTION

1.) ipconfig /release command will inform the DHCP server that we don't want to use previously assigned IP address any larger and we release it. After releasing the previously assigned IP address we can request new IP address from DHCP server with ipconfig /renew command. Note: As I am using **UBUNTU** distribution of Linux, I have used **dhclient** program for releasing and obtaining new IP address with DHCP which is equivalent to ipconfig program on Wİndows.

2.) When we look at the packet capture on Wireshark 1 DHCP packet was captured for the release command, 4 DHCP packets were captured for first renew command. 2 DHCP packets were captured for second renew command.

3.) If we look at the Wireshark packet capture when when I made two new requests respectively to DHCP server both newly assigned IP address in both situtation were same as we can see in the below screenshots. In both situations requested IP address was 192.138.0.1, but this may not always be the case, IP address can differ from previous one.

4.) When we look at the packets as a result of first renew command, we can see four different DHCP message types which are DISCOVER, OFFER, REQUEST, and ACK respectively. DHCP DISCOVER message is sent by client to locate DHCP server when client attempts to connect network for the first time. DHCP OFFER message is sent by server in response to DHCP discover message of client, and it caries configuration information along with the new IP address. DHCP REQUEST message is sent by server. Finally DHCP ACK message is sent by server to acknowledge DHCP REQUEST message sent from DHCP client. After receiving DHCP ACK message client obtains configuration parameters including new IP address.

5.) For first dhclient command (which is equivalent to piconfig /renew on Windows) common option fileds to all packets are as follows: subnet mask which defines the subnet IP of computer that belongs to particular LAN, IP address of DHCP server, and DHCP message type option which indicates the sending purpose of DHCP packet that I have explained detaily in the previous question (4) it can be of type DISCOVER, OFFER, REQUEST, ACK, etc.. to indicate what client wants from DHCP server and DHCP servers response.

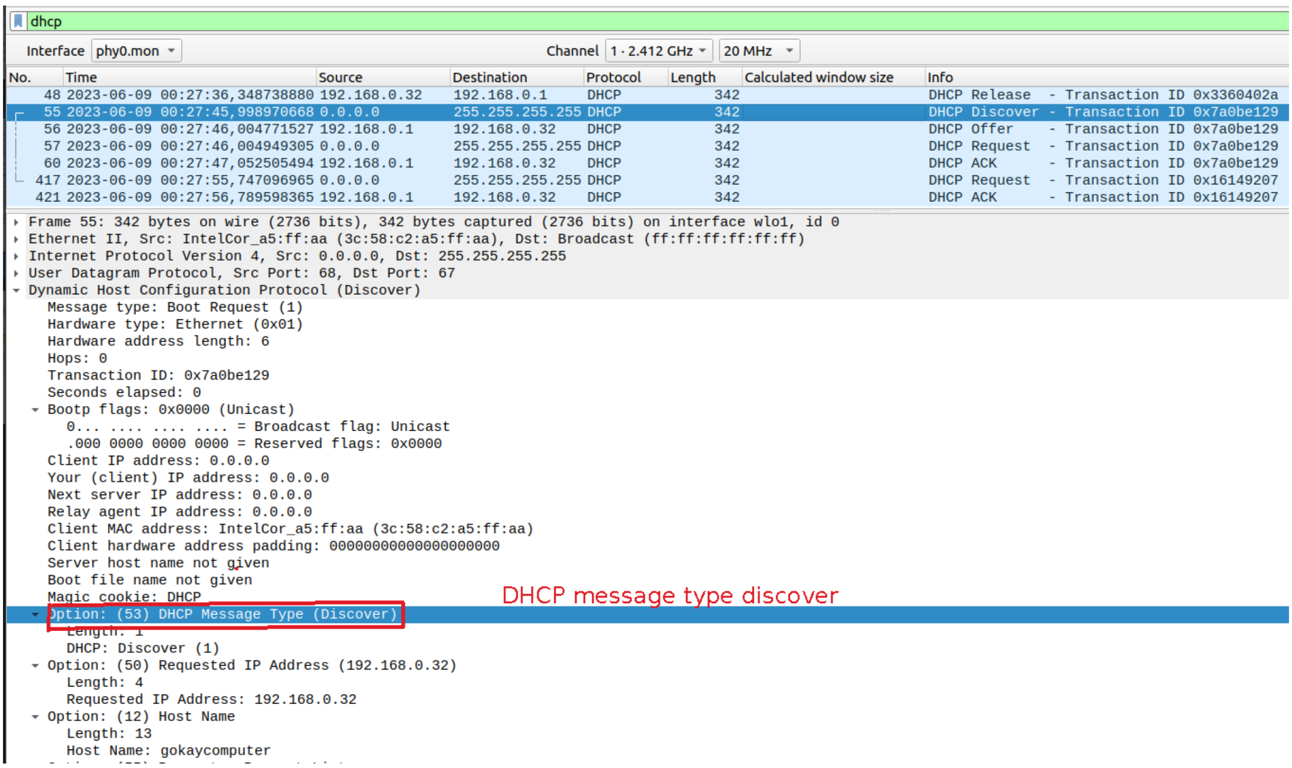


Figure 13: DHCP Discover message

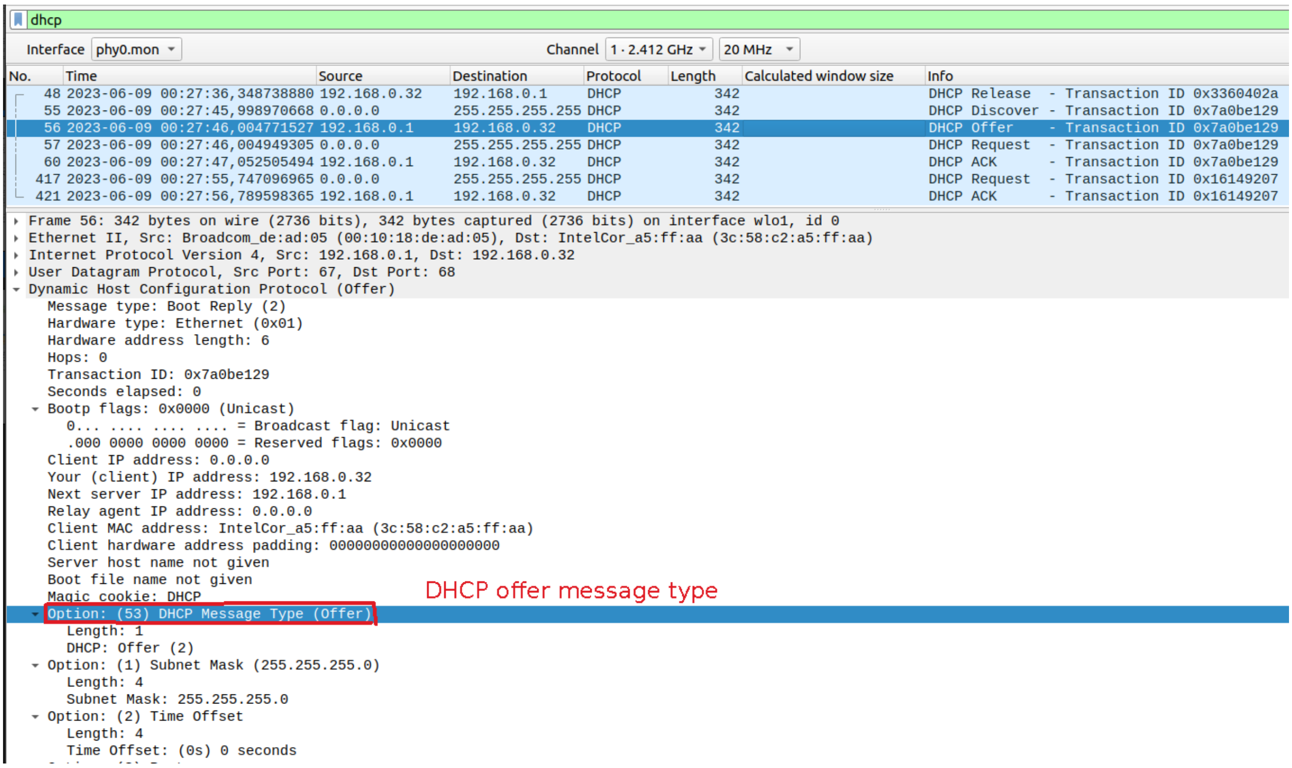


Figure 14: DHCP Offer message type

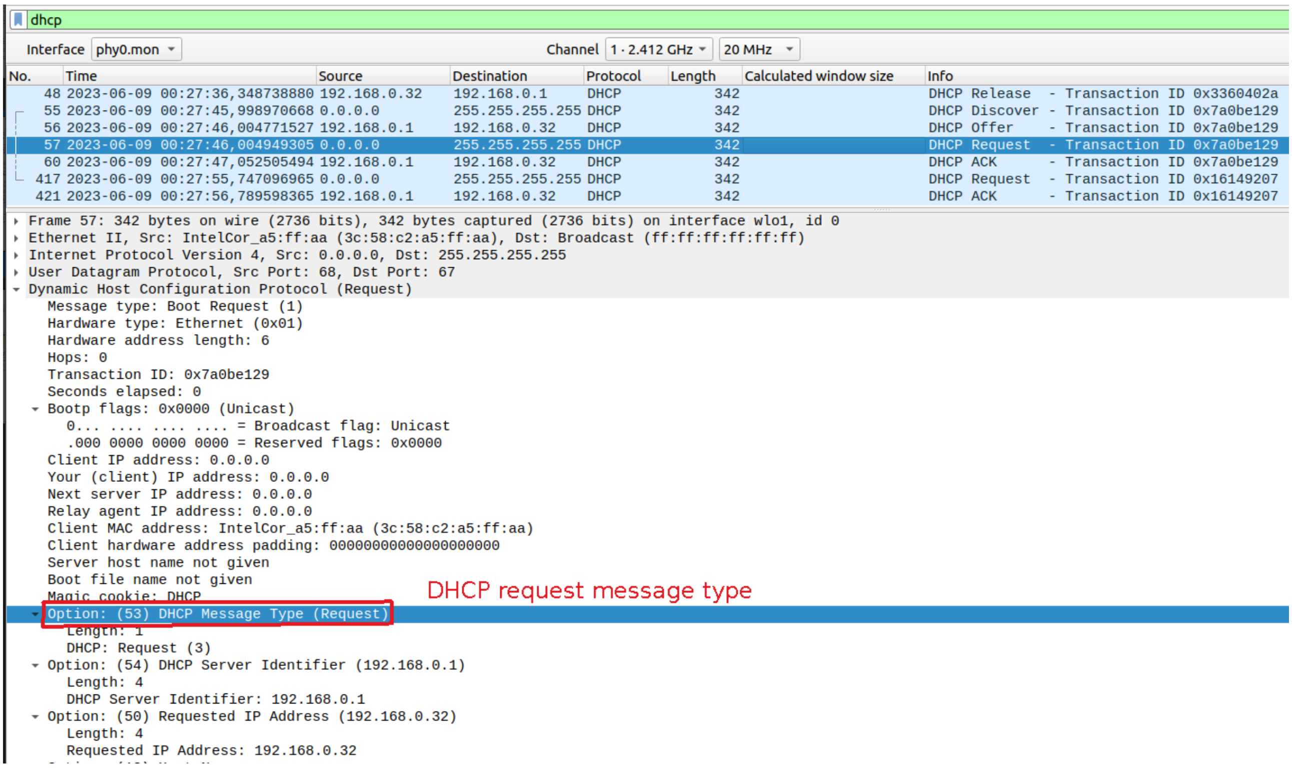


Figure 15: DHCP Request message type

Figure 16: dhclient command equivalent to ipconfig in WIndows for releasing IP address and obtaining new IP address

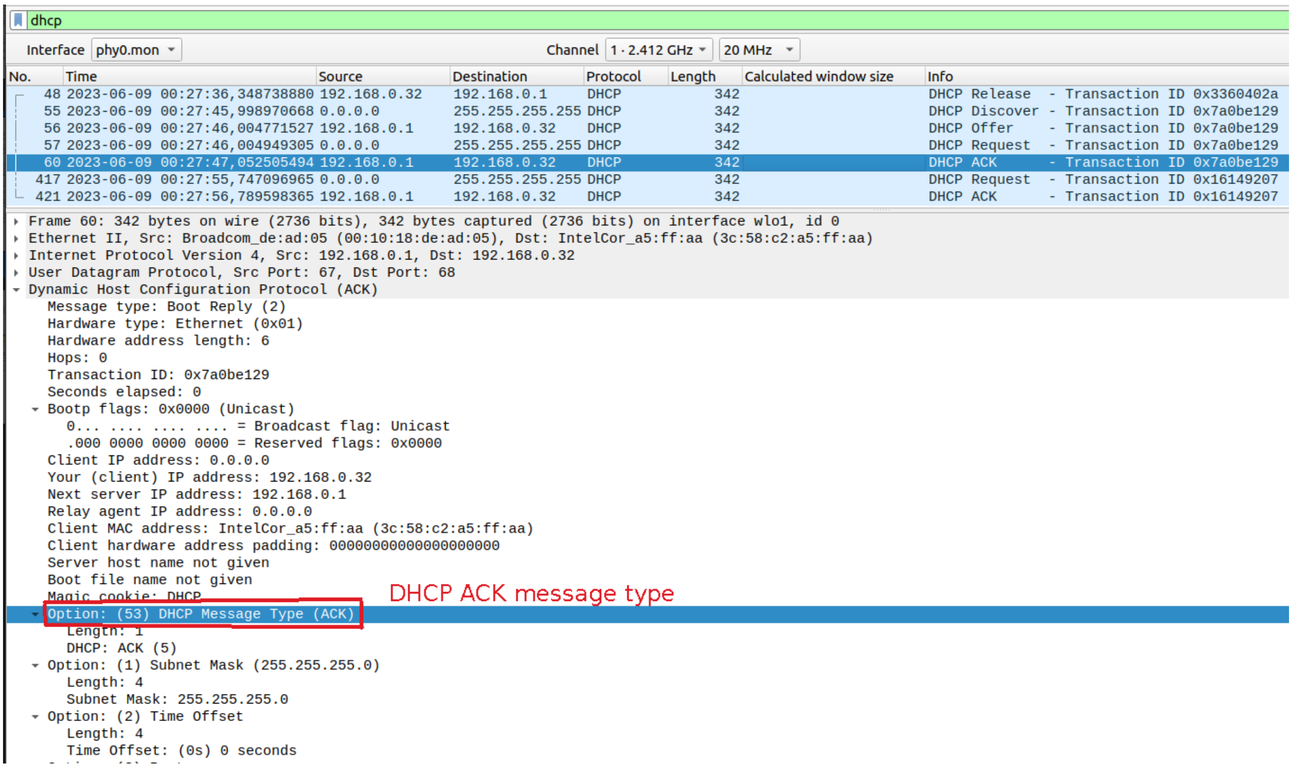


Figure 17: DHCP ACK message for second dhclient command (same as ipconfig /renew)