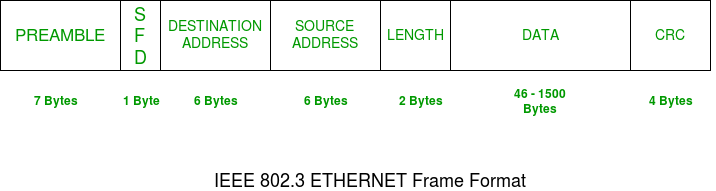
1. Ethernet protocol
   1. Frame structure (IEEE 802.3):



The basic format for ethernet protocol which is needed for all mac implementation which is an IEEE 802.3 standard.

* + 1. PREAMBLE:

preamble was introduced to allow for the loss of a few bits due to signal delays. But today’s high-speed Ethernet doesn’t need Preamble to protect the frame bits.

* + 1. SFD (Start of frame delimiter):

This is a 1-Byte field that is always set to 10101011. SFD indicates that upcoming bits are starting the frame, Sometimes SFD is considered part of PREAMBLE, this is the reason Preamble is described as 8 Bytes in many places.

* + 1. Destination address:

This is a 6-Byte field that contains the MAC address of the machine for which data is destined.

* + 1. Source address:

This is a 6-Byte field that contains the MAC address of the source machine.

* + 1. Length:

Length is a 2-Byte field, which indicates the length of the entire Ethernet frame.

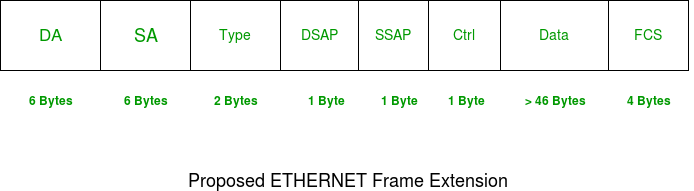
* + 1. Data:

This is the place where actual data is inserted, also known as Payload. Both IP header and data will be inserted here if Internet Protocol is used over Ethernet.

* + 1. CRC (Cyclic Redundancy Check):

CRC is 4 Byte field. This field contains a 32-bits hash code of data, which is generated over the Destination Address, Source Address, Length, and Data field. If the checksum computed by destination is not the same as sent checksum value, data received is corrupted.

* 1. Frame structure (Ethernet II):



Unlike basic version of ethernet protocol you can have a larger payload for your data.

* + 1. DA:

Destination MAC Address, size: 6bytes

* + 1. SA:

Source MAC Address, size: 6bytes

* + 1. Type:

Ethertype, size: 2bytes

* + 1. DSAP:

Destination Service Access Point, size: 1byte

* + 1. SSAP:

Source Service Access Point, size: 1byte

* + 1. Ctrl:

Control field, size: 1byte

* + 1. Data:

Data, size: gt 46bytes

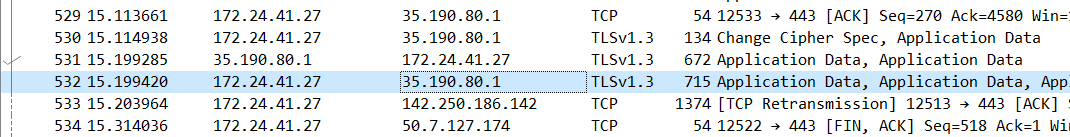
* + 1. FCS:

Frame checksum, size: 4bytes

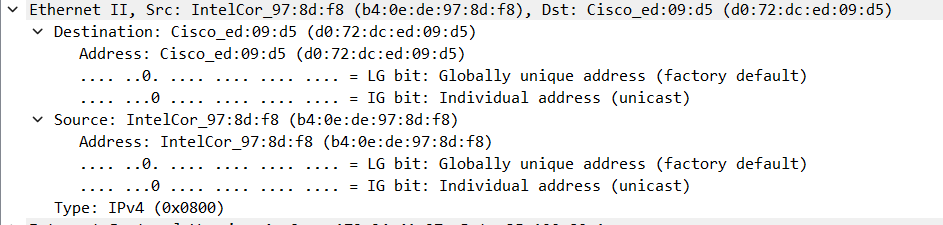
* 1. Capturing packets:

Ethernet protocol is observable over any packets sent and received but various of inner protocols:

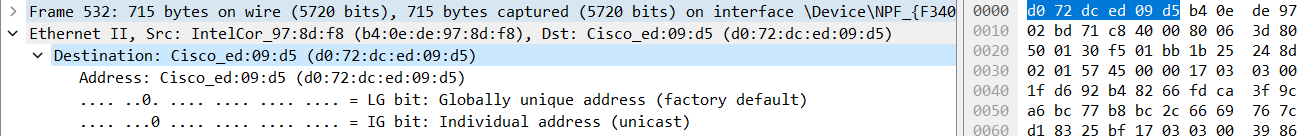
* This is a random packet for example:



* In frame sections you can see further details about requests:
* For example



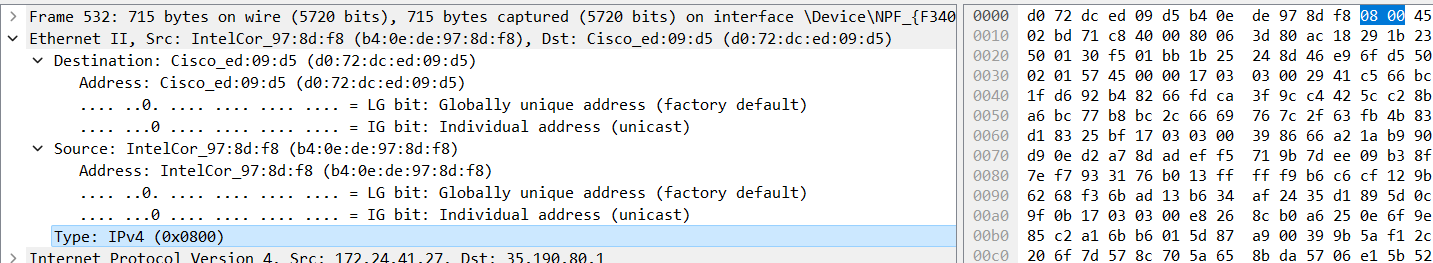
* As you can see there is destination, source and type as we saw in ethernet protocol structure
* Destinations field:



6 bytes length for destination as we saw

Source structure is same as destination.

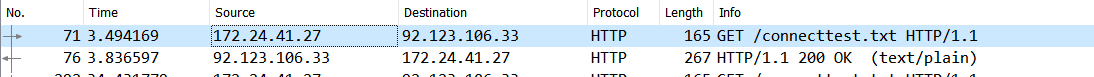
* Type field:



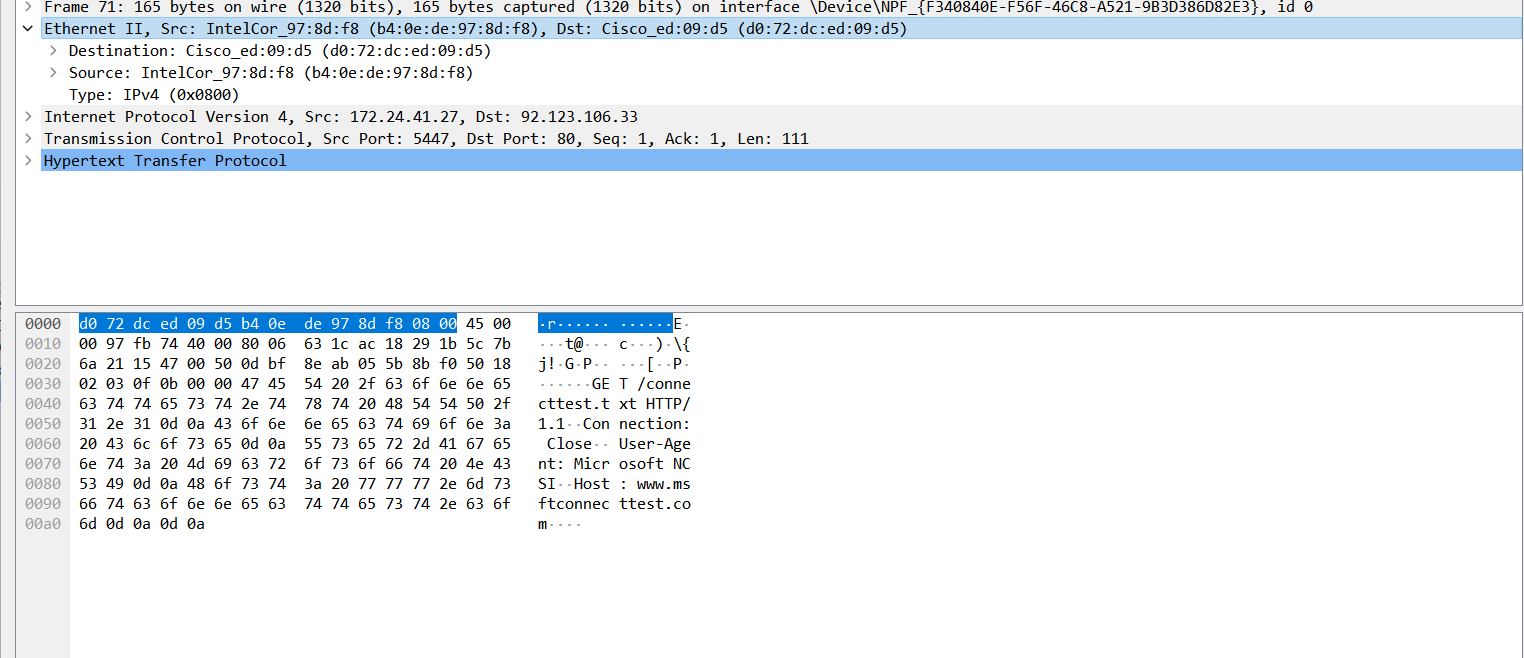
2 bytes for type

Another example:

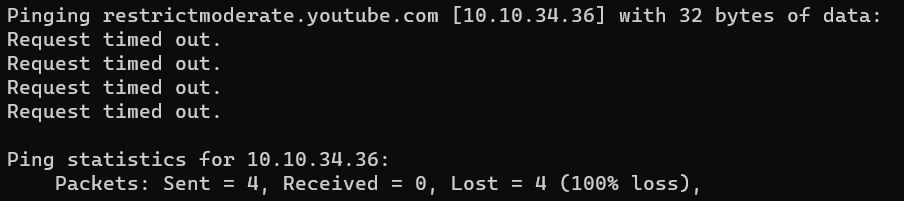
* A http request:



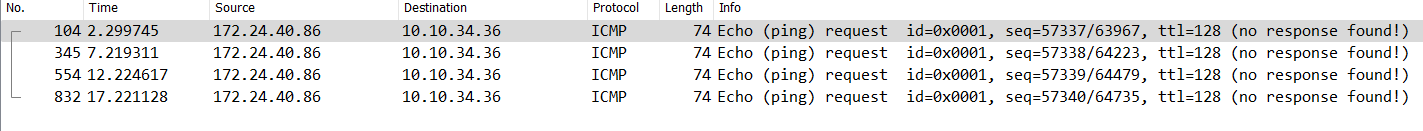
* Ethernet section



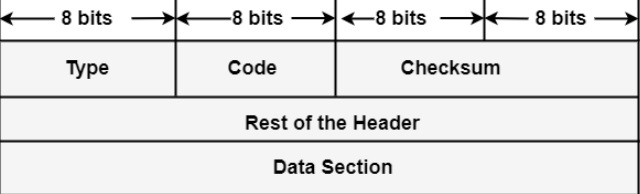
There is total of 14 bytes here, first 6 for destination, second 6 for source and last 2 for type.



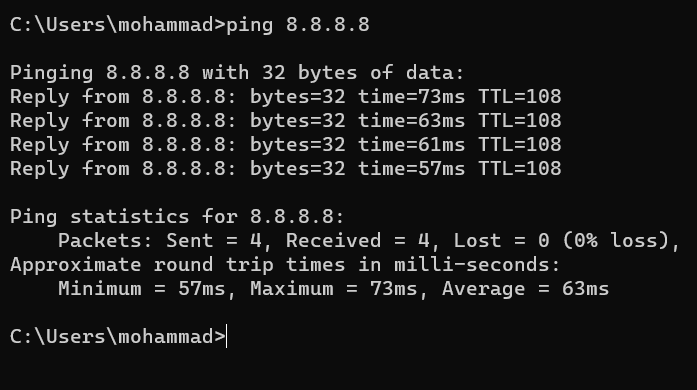
There is no reply’s because destination is unreachable:



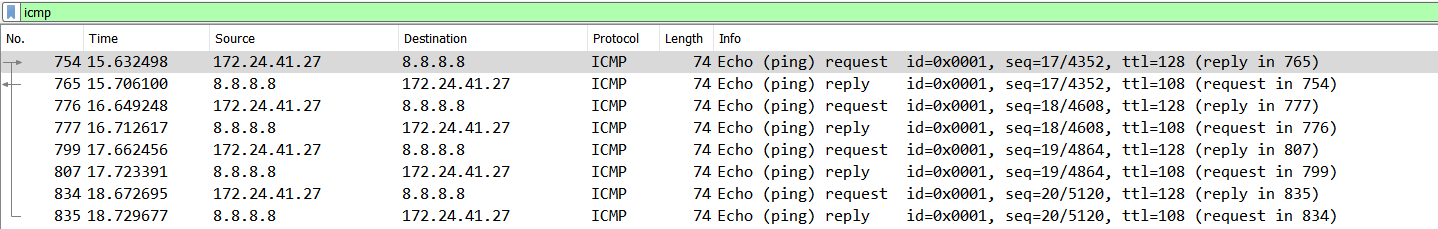
1. ICMP:
   1. Frame structure:



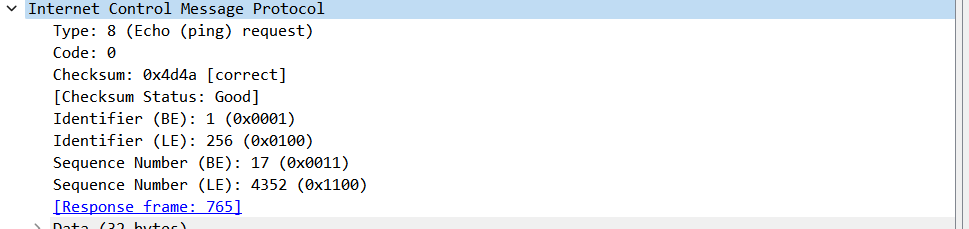
* + 1. Type: It is an 8-bit field. It represents the ICMP message type. The values area from 0 to 127 are described for ICMPv6, and the values from 128 to 255 are the data messages.
    2. Code: It is an 8-bit field that represents the subtype of the ICMP message.
    3. Checksum: It is a 16-bit field to recognize whether the error exists in the message or not.
  1. Capture section:
* We can use ping command on CMD to test the ICMP protocol:

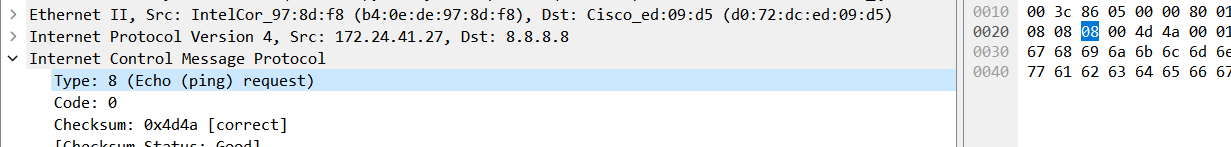


* There are 4 requests for ping to 8.8.8.8 server and if the given IP is a reachable IP there should be 4 replies too.
* Let’s check the wireshark:
* Search for ICMP protocols in wireshark

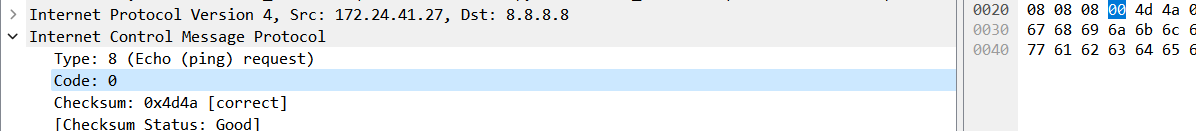


* There are 4 requests and for replies here
* For the request the source is our device and the destination is the given IP (here 8.8.8.8)
* And for replies the source and destination has been replaced with each other

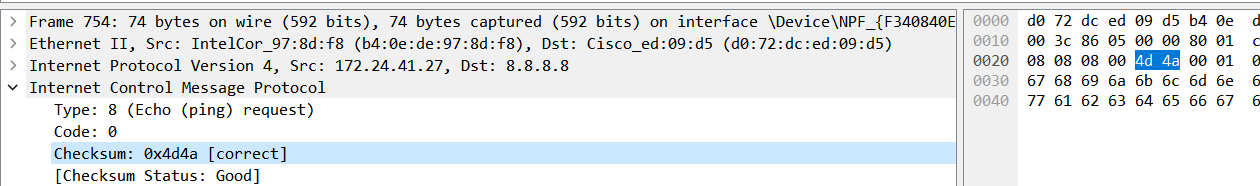




Type field: 1byte or 8bits

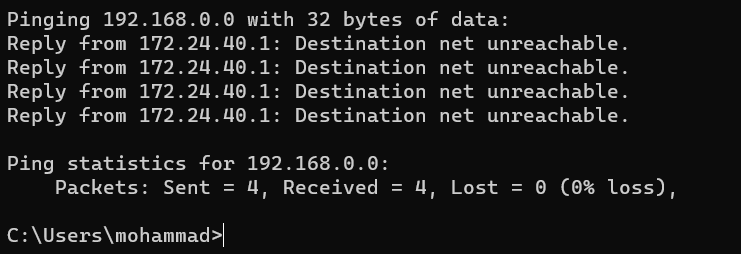


Code field: same as type field it’s 8bits

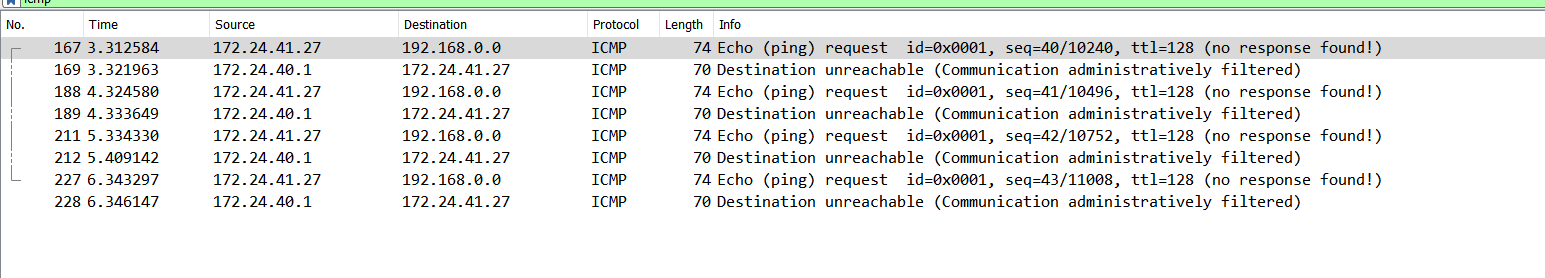


Checksum field: 16bits of data

* In second example we use an unreachable IP:

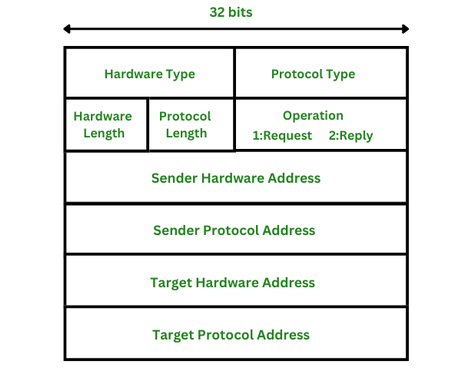


* There is 0 percent loss and there is the unreachable reply for given IP
* It means the request and reply was sent and received successfully but the given IP is not usable

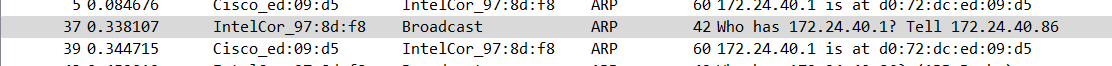


* Requests and replies

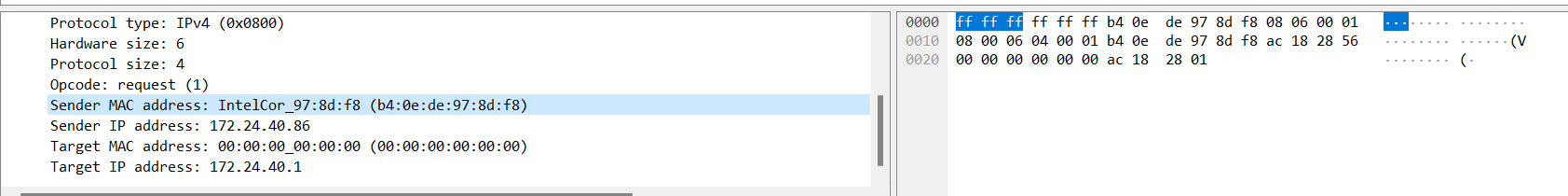
1. ARP (Address Resolution Protocol):
   1. Frame structure:



* + 1. Hardware type: This is 16 bits field defining the type of the network on which ARP is running. Ethernet is given type 1.
    2. Protocol type: This is 16 bits field defining the protocol. The value of this field for the IPv4 protocol is 0800H.
    3. Hardware length: This is an 8 bits field defining the length of the physical address in bytes. Ethernet is the value 6.
    4. Protocol length: This is an 8 bits field defining the length of the logical address in bytes. For the IPv4 protocol, the value is 4.
    5. Operation (request or reply): This is a 16 bits field defining the type of packet. Packet types are ARP request (1), and ARP reply (2).
    6. Sender hardware address: This is a variable length field defining the physical address of the sender. For example, for Ethernet, this field is 6 bytes long.
    7. Sender protocol address: This is also a variable length field defining the logical address of the sender For the IP protocol, this field is 4 bytes long.
    8. Target hardware address: This is a variable length field defining the physical address of the target. For Ethernet, this field is 6 bytes long. For the ARP request messages, this field is all Os because the sender does not know the physical address of the target.
    9. Target protocol address: This is also a variable length field defining the logical address of the target. For the IPv4 protocol, this field is 4 bytes long.
  1. Capturing packets:
* At first turn off the wi-fi.
* The start capturing in wireshark over wi-fi
* Then turn on the wifi
* Stop capturing in wireshark
* Search for arp protocols
* Example of sending arp request:

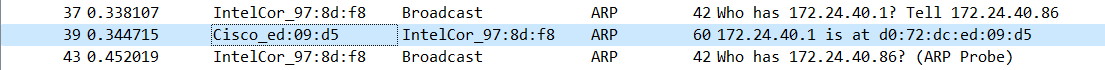


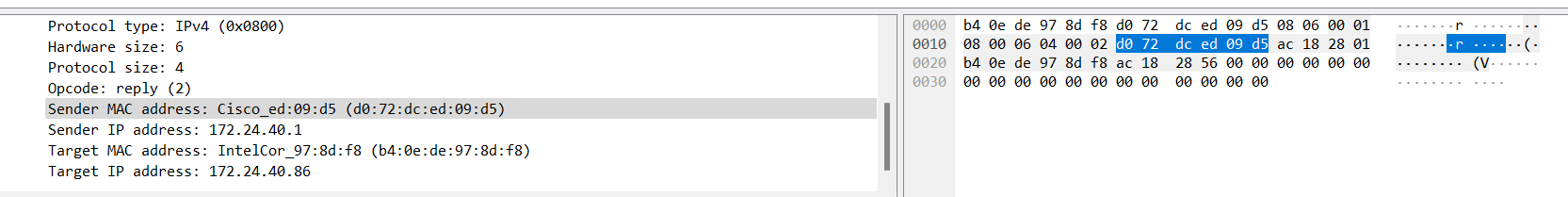
* In the frame sections you will see the frame structure of this protocol:



As you see the source address is my pc and there is no specific destination for arp protocol, packet travels to all needed destination by itself (broadcast).

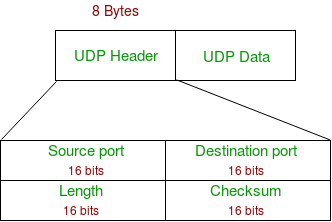
* Answer for above request:



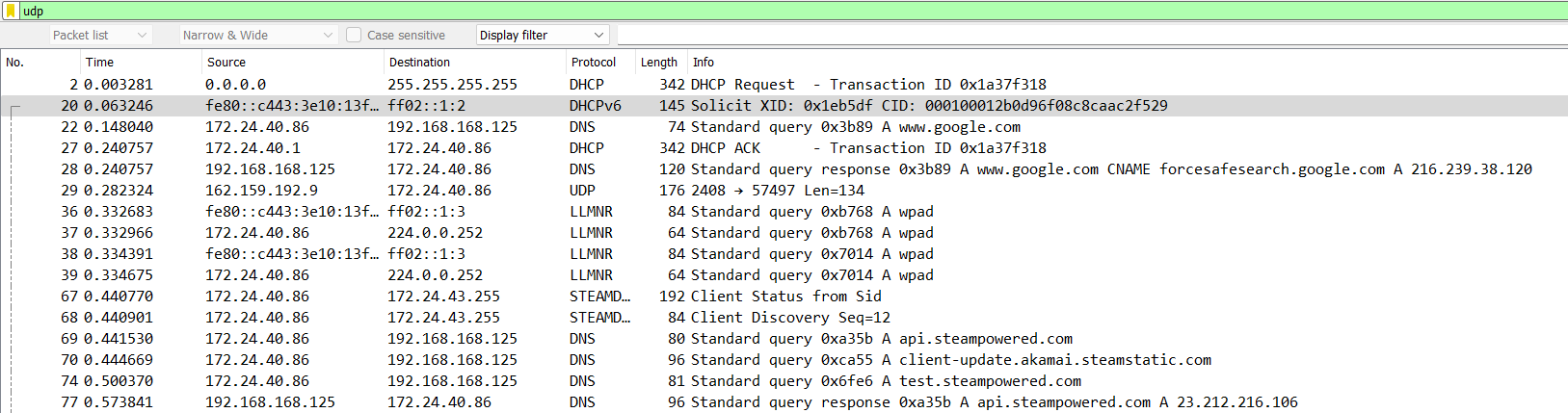


In the answer you see that the destination is changed to my address and the source address is now changed to receiver of last request.

1. UDP (User Datagram Protocol):
   1. Frame structure:

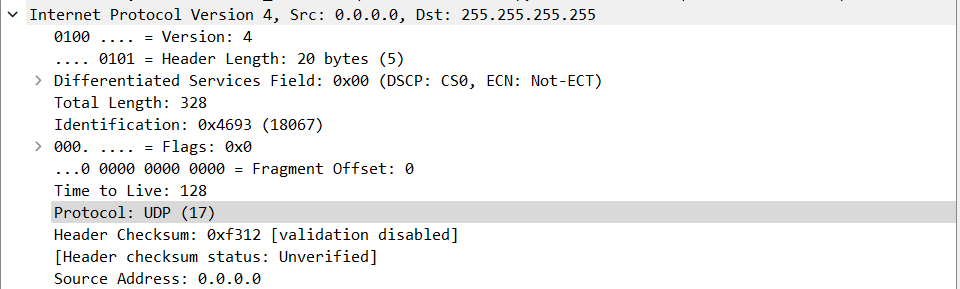


* + 1. Source Port: Source Port is a 2 Byte long field used to identify the port number of the source.
    2. Destination Port: It is a 2 Byte long field, used to identify the port of the destined packet.
    3. Length: Length is the length of UDP including the header and the data. It is a 16-bits field.
    4. Checksum: Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
  1. Capturing packets:
* Turn off the wi-fi
* Start capturing over wi-fi
* Turn on the wi-fi
* Search for udp protocls
* Now basically you can see all frames that are using udp itself or they are running over udp protocol like an inner protocol.

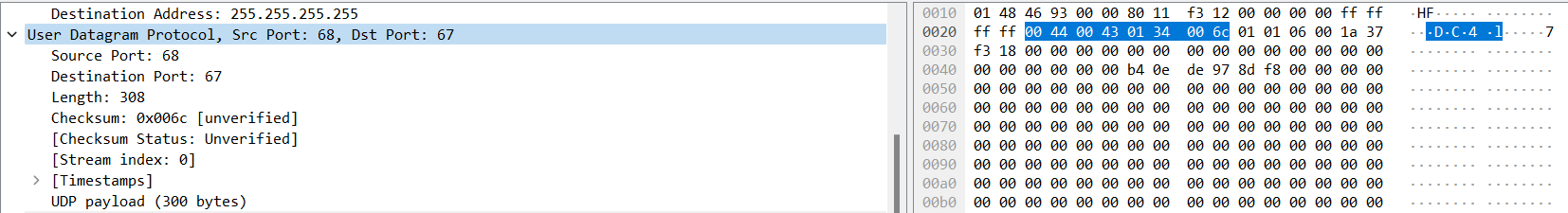


* As you see there are many protocols that are running udp as inner protocol like: DHCP, DHCPV6, DNS , LLMNR, MDNS, …
* For example take a look at a DHCP:



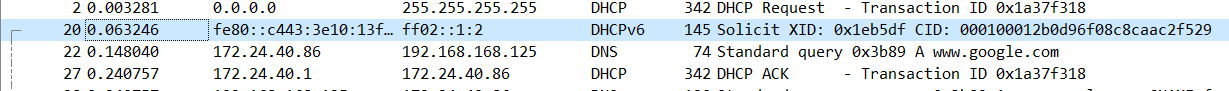


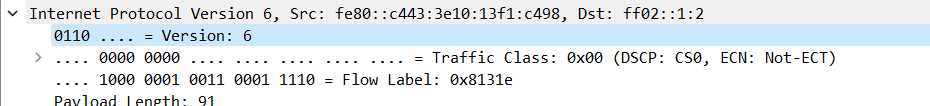
Inner protocol which is udp



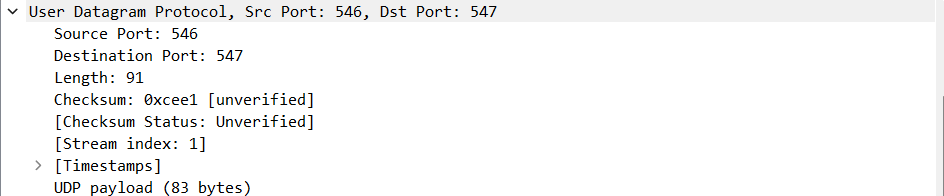
All udp sections of frame are shown

* Another example is DHCPV6 which uses version 6 protocol



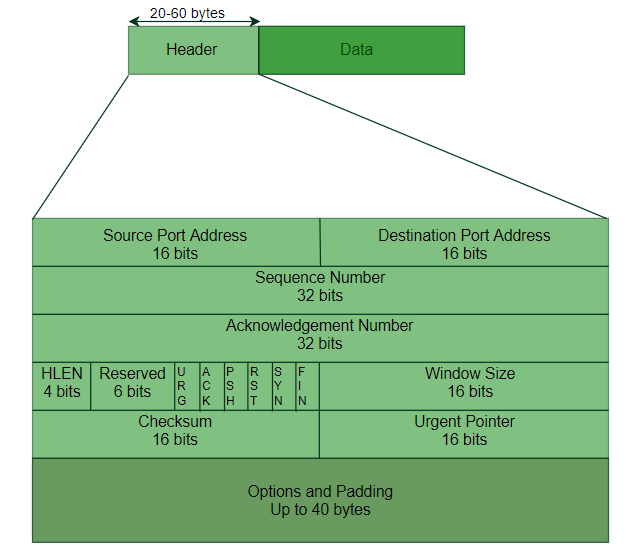


Version 6



Same udp frame

1. TCP (Transmission Control Protocol):
   1. Frame structure:



* + 1. Source Port Address:

a 16-bit field that holds the port address of the application that is sending the data segment.

* + 1. Destination Port Address:

a 16-bit field that holds the port address of the application in the host that is receiving the data segment.

* + 1. Sequence Number:

a 32-bit field that holds the sequence number, i.e, the byte number of the first byte that is sent in that particular segment. It is used to reassemble the message at the receiving end of the segments that are received out of order.

* + 1. Acknowledgement Number:

a 32-bit field that holds the acknowledgement number, i.e, the byte number that the receiver expects to receive next. It is an acknowledgement for the previous bytes being received successfully.

* + 1. Header Length (HLEN):

This is a 4-bit field that indicates the length of the TCP header by a number of 4-byte words in the header, i.e if the header is 20 bytes(min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it’ll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.

* + 1. Control flags:

These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc.

* + 1. Window size:

This field tells the window size of the sending TCP in bytes.

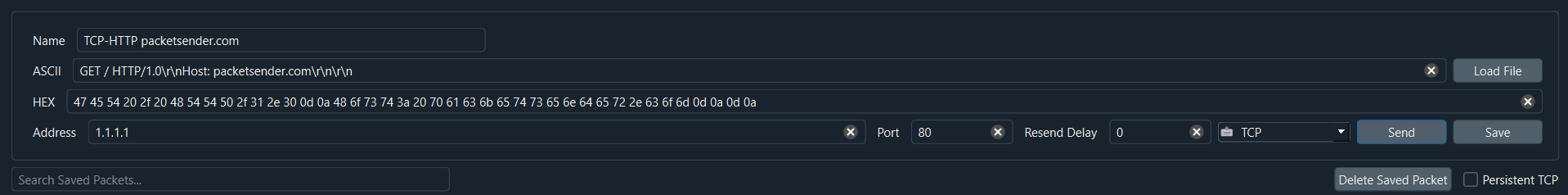
* + 1. Checksum:

This field holds the checksum for error control. It is mandatory in TCP as opposed to UDP.

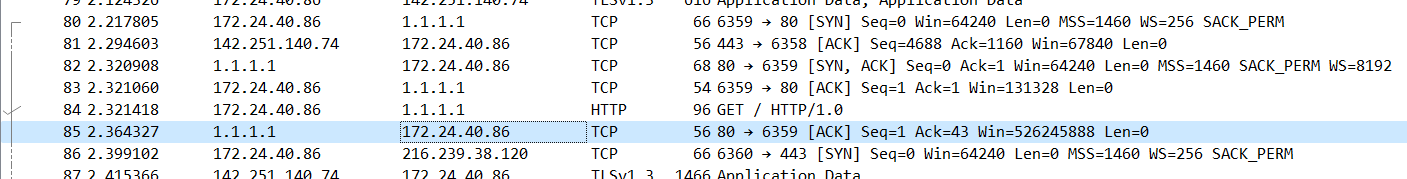
* + 1. Urgent pointer:

This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

* 1. Capturing frames:
* We use packet sender app to send a tcp frame
* Then we capture it on wireshark

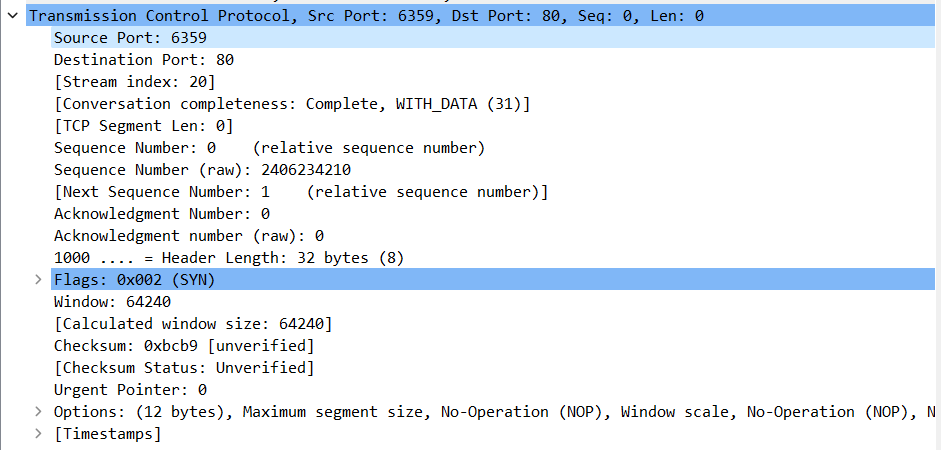


Sent segment



Logs for tcp protocol

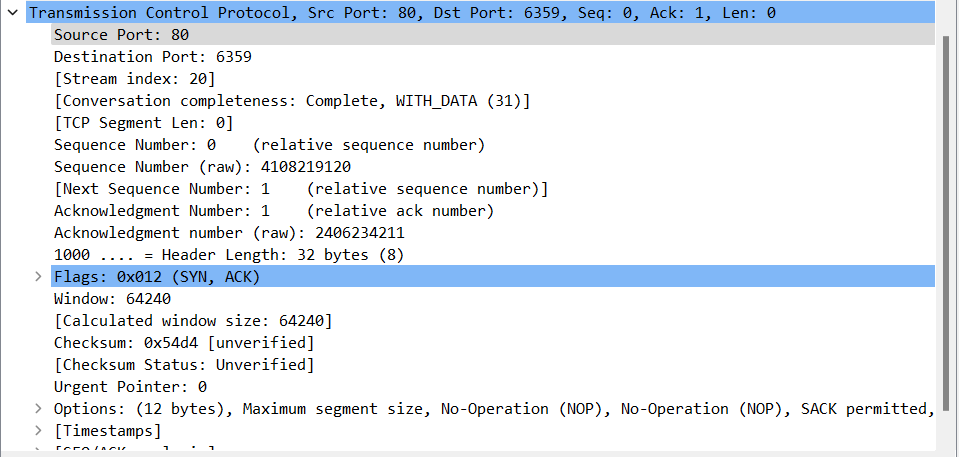
* At first we see a request from our device to 1.1.1.1 and after that we see the response came back with our device as the destination and 1.1.1.1 as sender



Tcp segment of request

As you see the sequence number is set to zero which it means the connection has just started

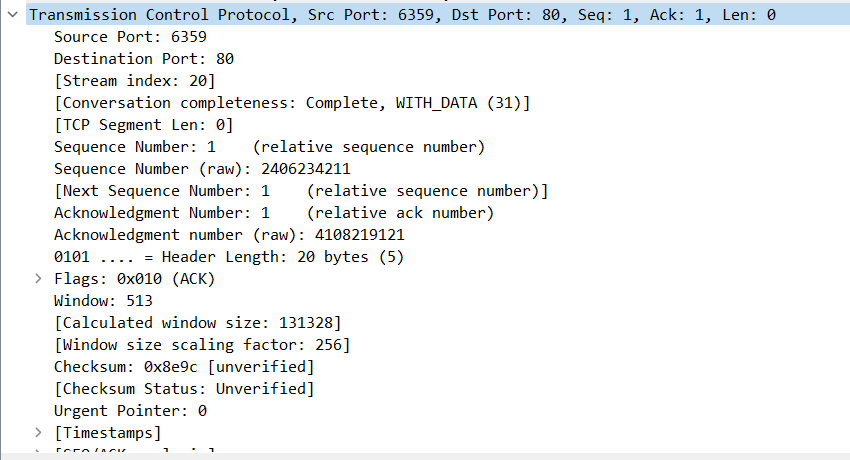
And we have the syn flag for the Synchronize sequence numbers



Here you see the acknowledgment for the our request (ack flag is up too)

Note that the place of source and destination port has changed.

* Now we see the second segment which was sent just after the first segment

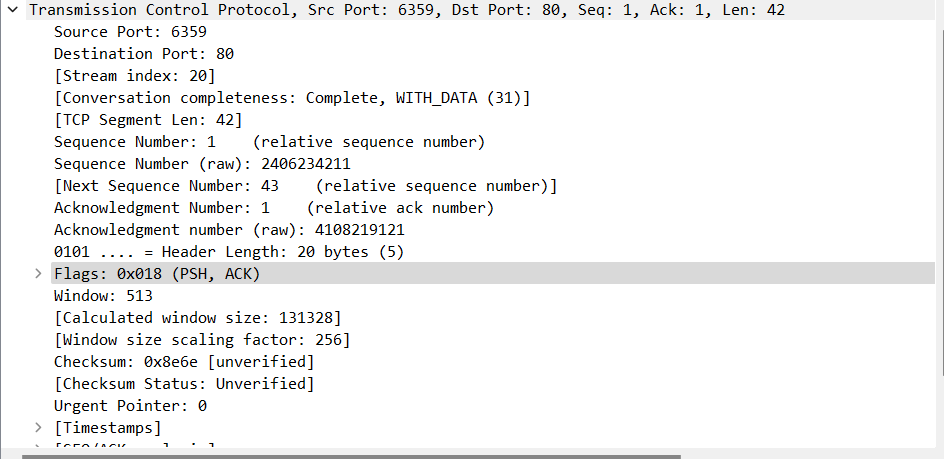


As we can see the sequence number has incremented by one and the syn flag is no longer up (because synchronizes has been done at last segment)

Another thing to mention is that window size has gone up.

Then after this ack which was sent by out device we see another segment:



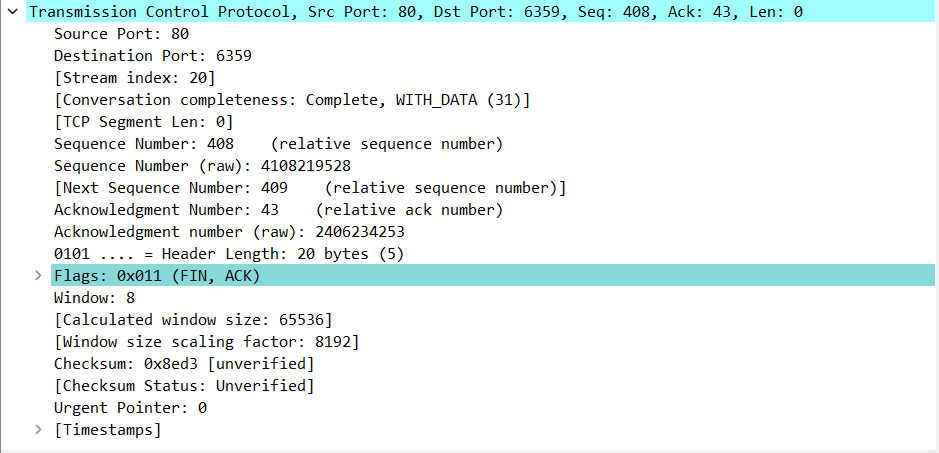


As you see the psh(request for push) flag has been added is to send data

After all communication between us and server

The server sent this segment to terminate the connection:





You can see that the FIN flag has been added which means termination of connection

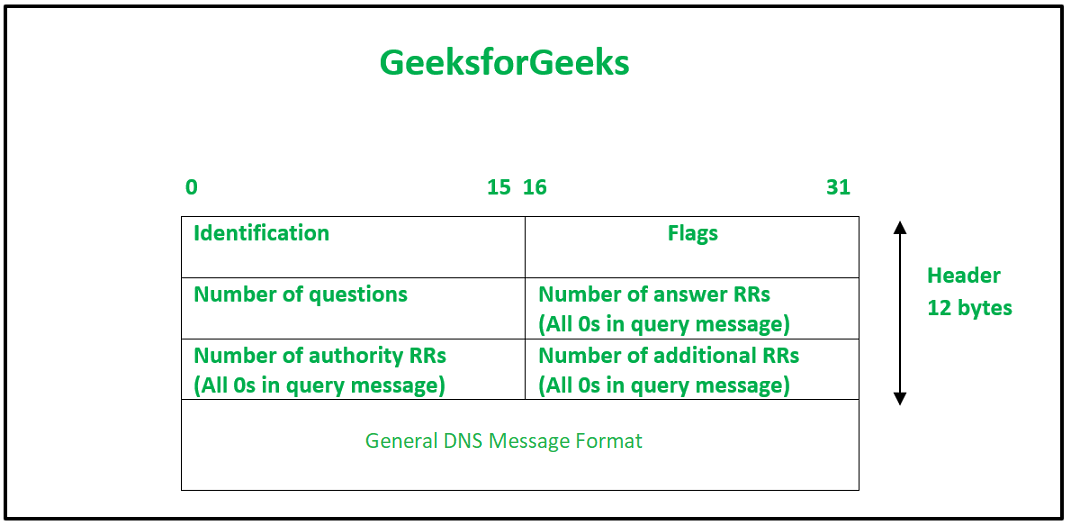
1. DNS (Domain Name System):
   1. DNS structure:

DNS allows you to interact with devices on the Internet without having to remember long strings of numbers. Changing of information between client and server is carried out by two types of DNS messages:

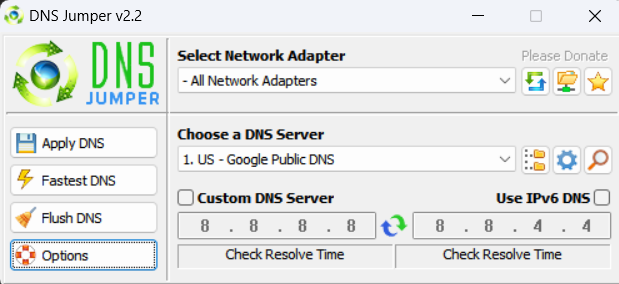
Query message - Response message.

Query is sent by us and response is received by us intervally.

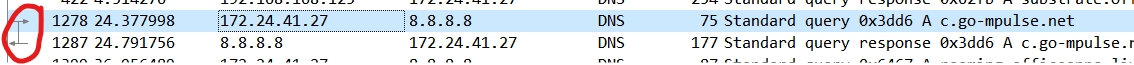
Query and response (almost) share the same structure:



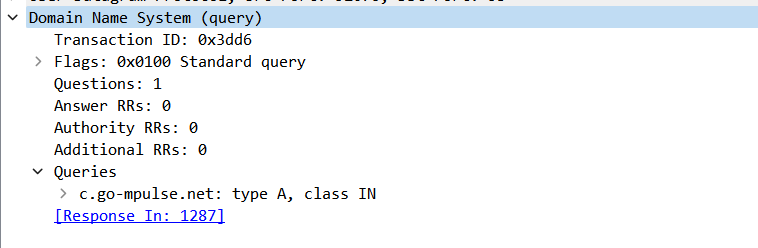
* + 1. Identification: The identification field is made up of 16 bits which are used to match the response with the request sent from the client-side. The matching is carried out by this field as the server copies the 16-bit value of identification in the response message so the client device can match the queries with the corresponding response received from the server-side.
    2. Number of Questions- It is a 16-bit field to specify the count of questions in the Question Section of the message. It is present in both query and response messages.
    3. A number of answer RRs- It is a 16-bit field that specifies the count of answer records in the Answer section of the message. This section has a value of 0 in query messages. The server answers the query received from the client. It is available only in response messages.
    4. A number of authority RRs- It is a 16-bit field that gives the count of the resource records in the Authoritative section of the message. This section has a value of 0 in query messages. It is available only in response messages. It gives information that comprises domain names about one or more authoritative servers.
    5. A number of additional RRs– It is a 16-bit field that holds additional records to keep additional information to help the resolver. This section has a value of 0 in query messages. It is available only in response messages.
  1. Capture section:
* On this section to check some DNS requests we try to change our DNS.
* You can change your DNS manually or you can change it using third-party softwars.
* I will be using DNS-jumper here:



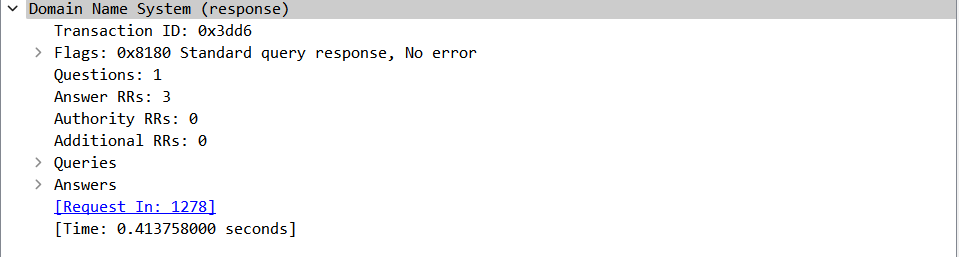
* Select a server and then apply.



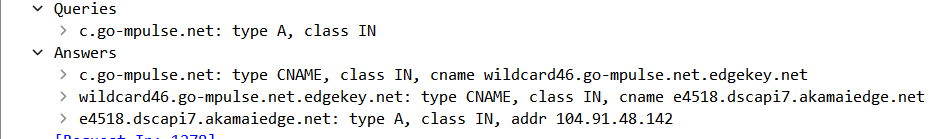
* These two sections shows the query and response message.
* For the query message the source is our IP and the destination is the IP of the selected server (8.8.8.8 or google.com).
* And for the response message the destination is our IP.



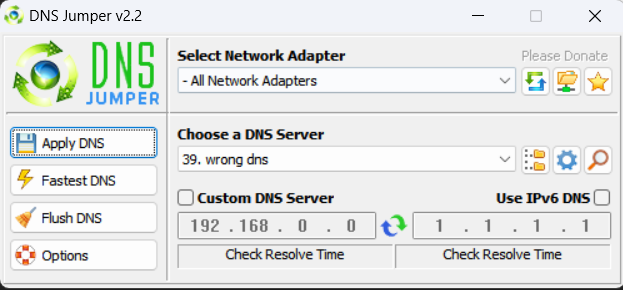
* Query:
* All sections match the known headers.
* There is a flag section that has multiple flags.
* For the query all you need is the standard query flag.
* As you can see you can directly reach to the response message which was below the query message.



* This is the response message.
* As you can see there is extra flags that determines the type of the message (for example you know that this one is a response and It has no errors)



* The query section is the same query as the query message.
* And answer sections is filled too.
* Another example for DNS:
* This time we try to use an unreachable IP for our DNS:





* Our devices tries to reach to first IP
* But it was unreachable and there is no answer like the last example.
* After not reaching the first IP it tries the second IP.
* Which this time is reachable

