Group 1:

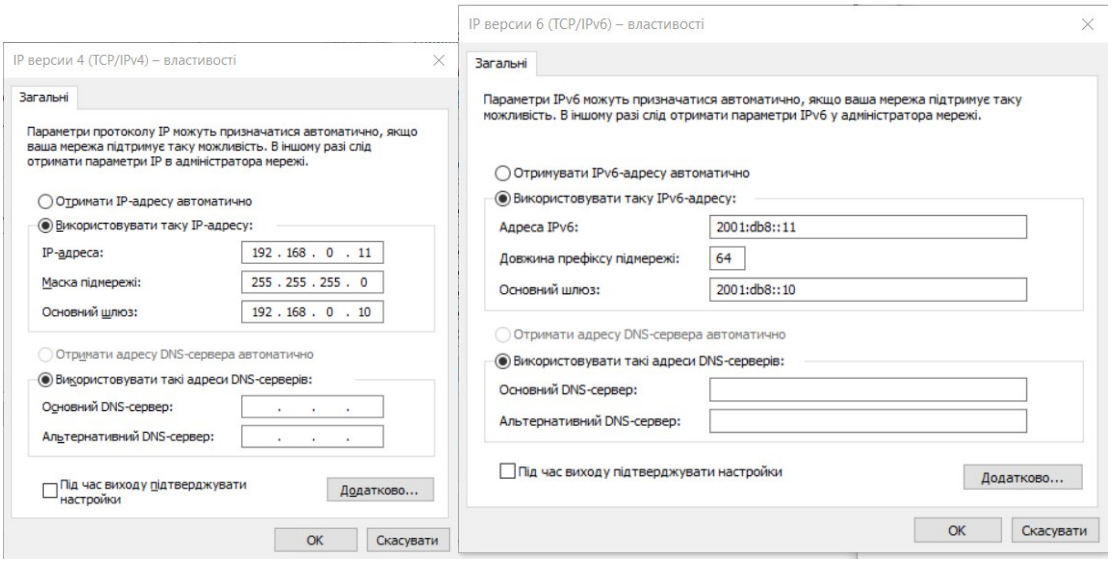
Vasylyshyn Danylo 256711

Nykonchuk Illia 245693

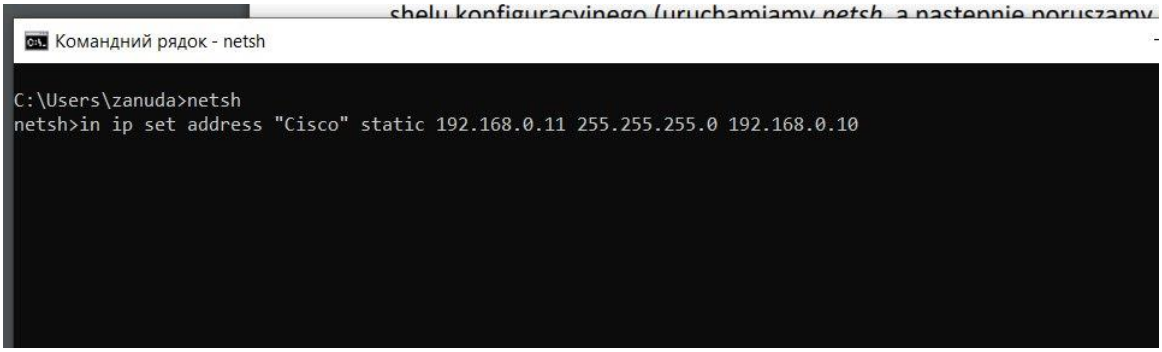
Lab 4

Task 1

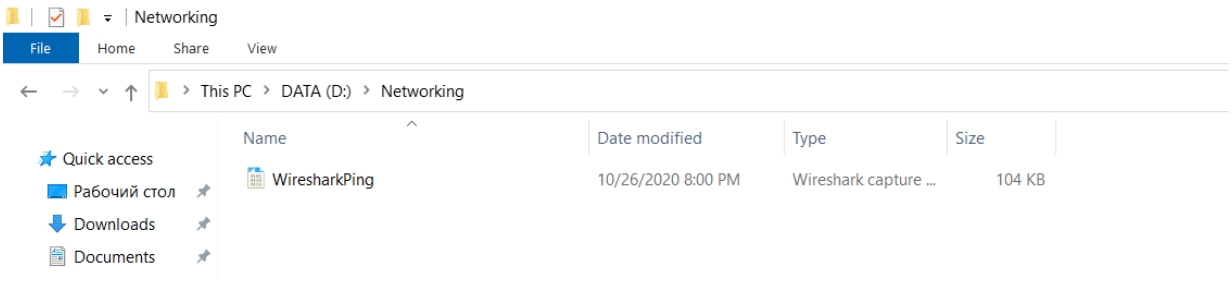
Configuring ip's:

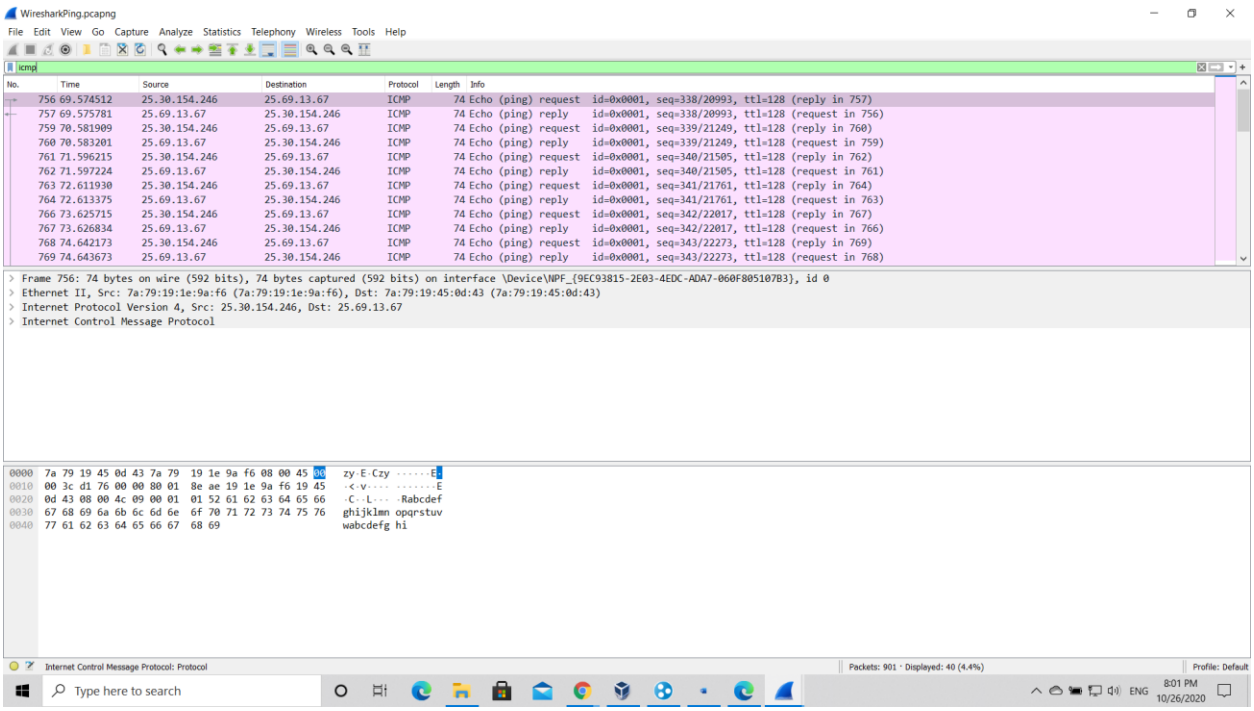


Command for configuring in cmd.



1) pinging my partner capturing ICMP packets with wireshark, save it and then open a saved file:





- Frame id's: 756 –798 (I guess it's just the order number of the frame captured by wireshark)
- My ip – 25.30.154.246, partners ip – 25.69.13.67

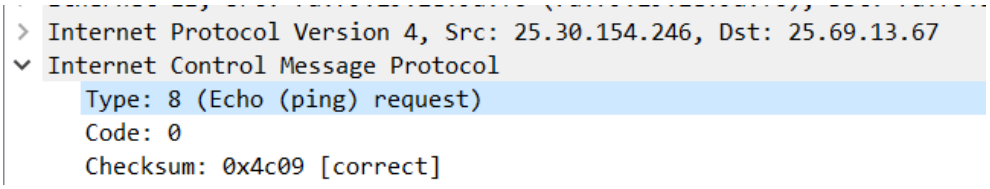
```
756 69.574512    25.30.154.246    25.69.13.67    ICMP    74 Echo (ping) request  id=0x0001, seq=338/20993, ttl=128 (reply in 757)
757 69.575781    25.69.13.67      25.30.154.246  ICMP    74 Echo (ping) reply    id=0x0001, seq=338/20993, ttl=128 (request in 756)
```

- Time to live : 128

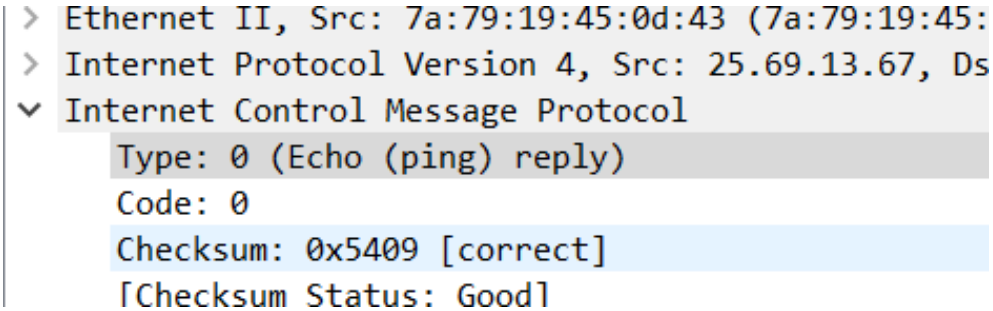
```
Total Length: 60
Identification: 0xd176 (53622)
> Flags: 0x0000
Fragment offset: 0
Time to live: 128
Protocol: ICMP (1)
Header checksum: 0x8eae [validation disabled]
[Header checksum status: Unverified]
```

- Types and names:

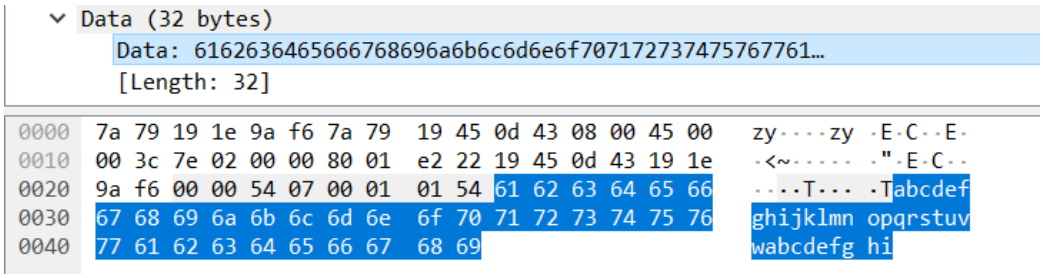
  8 Echo (ping) request



0 Echo (ping response)



- Size and content of ICMP data field:



Size – 32 bytes

Content – alphabet part without xyz

B)

Pinging gogle.com

```
C:\Users\danyl>ping -f -l 1472 google.com

Pinging google.com [216.58.215.78] with 1472 bytes of data:
Reply from 216.58.215.78: bytes=68 (sent 1472) time=9ms TTL=117
Reply from 216.58.215.78: bytes=68 (sent 1472) time=9ms TTL=117
```

-f argument stands for no fragmentation

-l – for size in this case 64 bytes

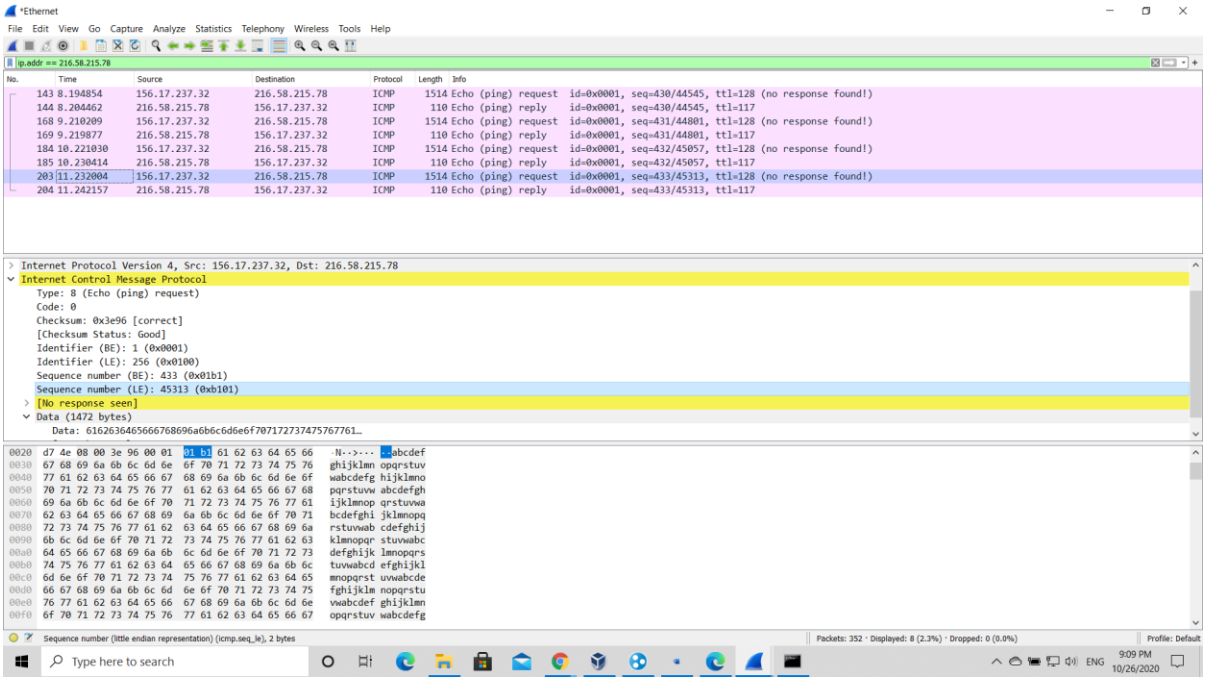
1472 – maximum size of ping packet, when DF set with gogle.com

```
C:\Users\danyl>ping -f -l 1473 google.com

Pinging google.com [216.58.215.78] with 1473 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
```

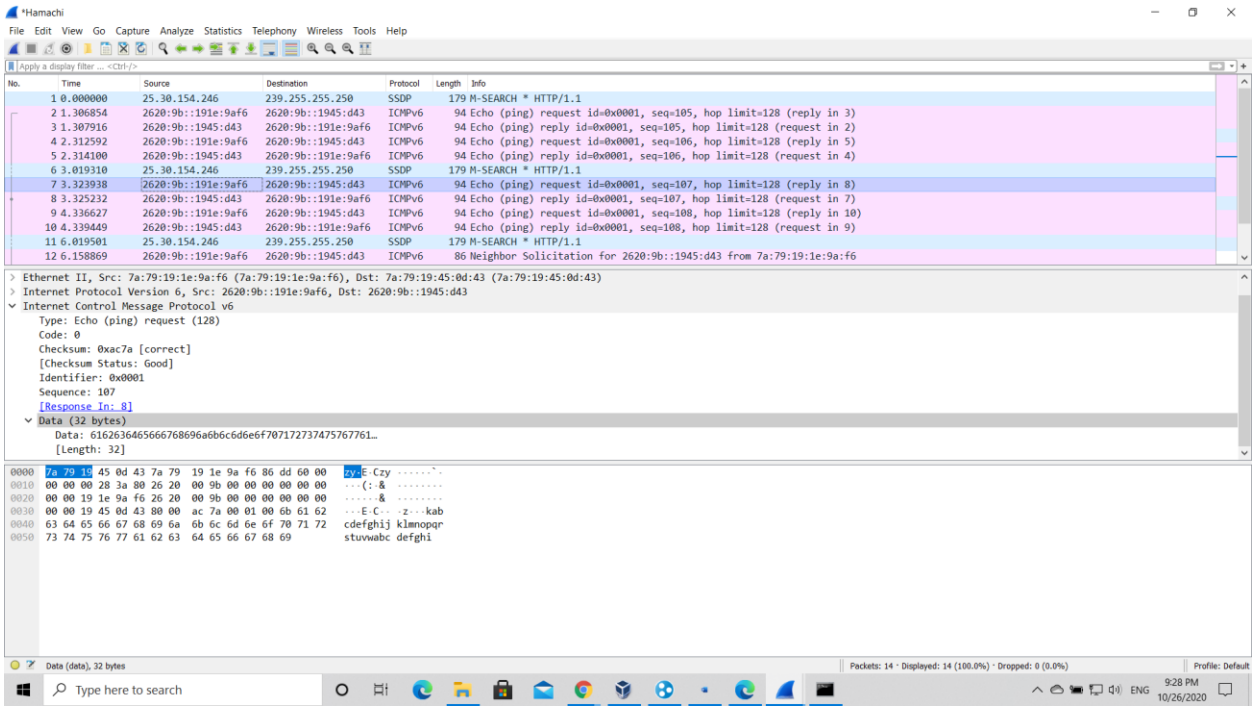
Data from wireshark:

- Source ip: 156.17.237.32
- Destination ip : 216.58.215.78
- Time to live- 128
- Non fragment bit: 1
- The name and type: 8 Echo (ping) request, 0 Echo (ping) reply

- Max size – 1472 bytes without fragmentation, content – alphabet without xyz



Task 2:



- Source ip: 2620:9b::191e:9af6
- Destination ip: 2620:9b::1945:d43

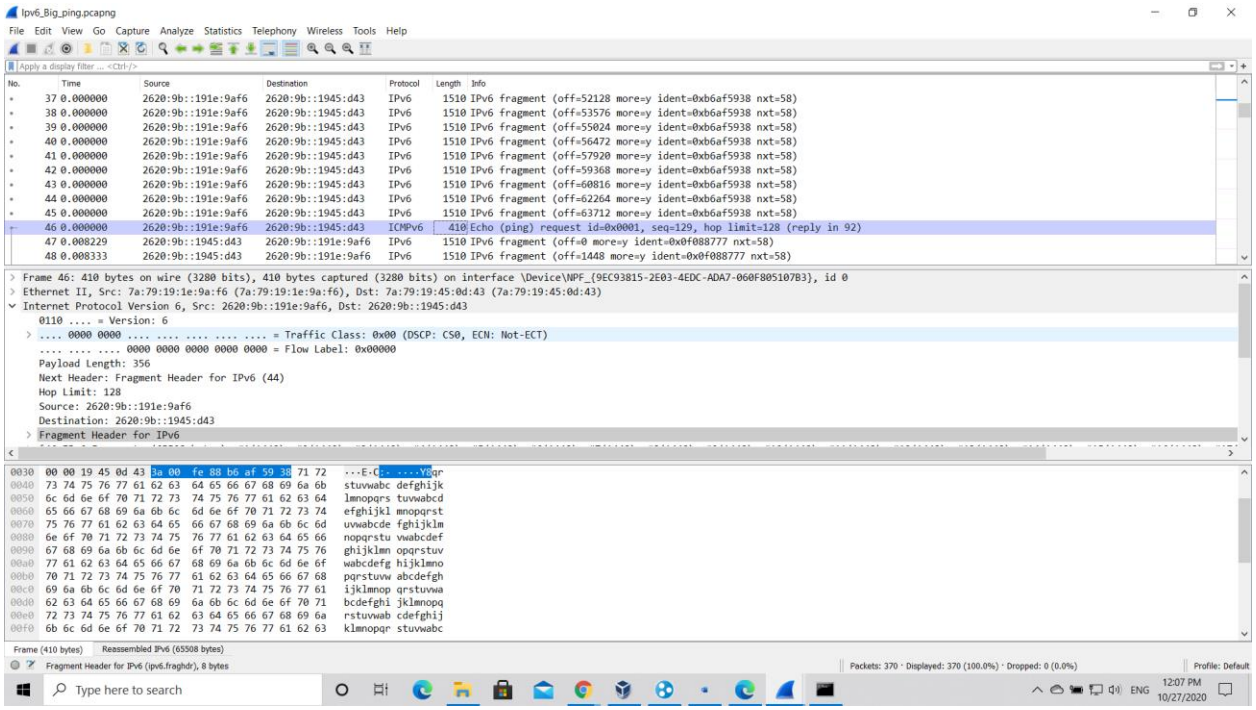
- TTL field is 128 and it's called Hop Limit in IPv6

```
> Frame 7: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface \Device\NPF_{9EC93815-2E03-4EDC-ADA7-060F805107B3}, id 0
> Ethernet II, Src: 7a:79:19:1e:9a:f6 (7a:79:19:1e:9a:f6), Dst: 7a:79:19:45:0d:43 (7a:79:19:45:0d:43)
v Internet Protocol Version 6, Src: 2620:9b::191e:9af6, Dst: 2620:9b::1945:d43
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 40
    Next Header: ICMPv6 (58)
    Hop Limit: 128
    Source: 2620:9b::191e:9af6
    Destination: 2620:9b::1945:d43
> Internet Control Message Protocol v6
```

- Names and types of ICMPv6 packets: Echo(ping) request(128), Echo(ping) reply(129)
- Default data field size: 32 bytes, data – alphabet no xyz

B)

When pinging host by ipv6 address the maximum size of a ping pakcet is 65500 bytes, but it should be fragmented into parts of 1448 bytes



- Source ip: 2620:9b::191e:9af6
- Destination ip: 2620:9b::1945:d43
- Hop limit: 128
- The fragmentation in IPv6 is solved using an ICMPv6 response, and is controlled by the end-host, so we have the fragmentation header field.
- Echo(ping) request (128), Echo(ping) reply (129)
- Maximal size of data field: 65500

Task 3

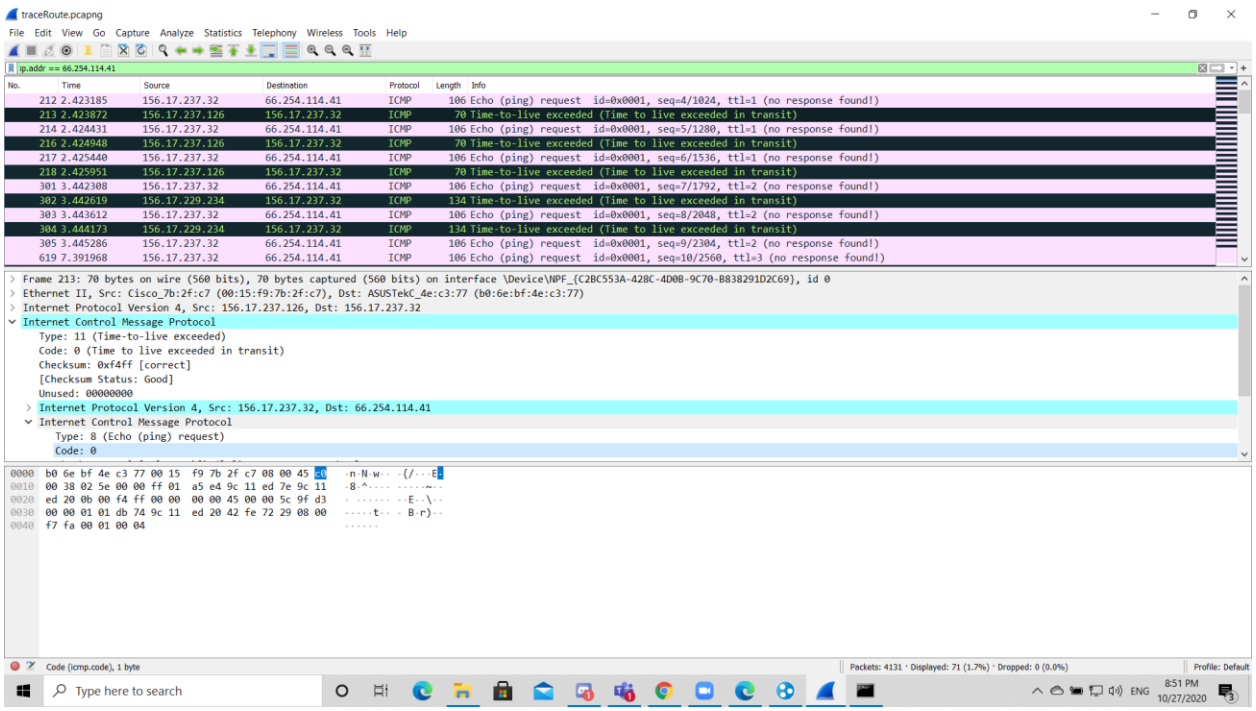
Here's the routing table:

```
Tracing route to pornhub.com [66.254.114.41]
over a maximum of 30 hops:

  1    <1 ms    <1 ms     1 ms  xxxx.t19.ds.pwr.wroc.pl [156.17.237.126]
  2    <1 ms     1 ms     *     234.ds.pwr.wroc.pl [156.17.229.234]
  3    11 ms    12 ms    19 ms  t15-wittiga2.ds.pwr.wroc.pl [156.17.229.241]
  4     2 ms     4 ms     1 ms  156.17.229.255
  5     1 ms    <1 ms     1 ms  pwr-zds-centrum3-vprn.wask.wroc.pl [156.17.254.41]
  6     5 ms     5 ms     5 ms  z-wroclawia.poznan-gw3.10Gb.rtr.pionier.gov.pl [212.191.224.105]
  7    10 ms    10 ms    10 ms  ae100.edge3.Berlin1.Level3.net [212.162.10.81]
  8     *        *        *     Request timed out.
  9    22 ms    22 ms    21 ms  ae3.cr2-fra6.ip4.gtt.net [213.254.196.1]
 10    23 ms    22 ms    22 ms  ae22.cr11-fra2.ip4.gtt.net [89.149.180.226]
 11    22 ms    23 ms    22 ms  ip4.gtt.net [213.254.223.254]
 12    21 ms    21 ms    21 ms  reflectededge.reflected.net [66.254.114.41]

Trace complete.
```

If we examine the packets sent in wireshark:

Outcoming packets:

- Source ipv4 is always: 156.17.237.32 (This is my ipv4)
- Destination ipv4 address – 66.254.114.41 (Ip address of the remote server)
- TTL field value – varies from 1 to 2 in the first 6 packets (3 packets for each intermediary device)
- Types and names: 8 Echo(ping) request
- 64 bytes of 0’s



Incoming packets:

- Source IPv4 address: first three: 156.17.237.126(dormitory router by hostname xxxx.t19.ds.pwr.wroc.pl),

  second three: 156.17.229.234 some device belonging to pwr 234.ds.pwr.wroc.pl

- Destination IPv4 address: 156.17.237.32 (my ip)
- TTL: 255
- Types and names: 11 (Time-to-live exceeded)
- No data field, the package sent is copied back

The middle nodes respond to traceroute packets, because they are the point when time to live decrements to 0 so they are sending back the error by the ICMP packet

Task 4

Ip address show:

```
root@deb10:~# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1
000
    link/ether 08:00:27:64:ff:a6 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 86335sec preferred_lft 86335sec
    inet6 fe80::a00:27ff:fe64:ffa6/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:12:72:2b brd ff:ff:ff:ff:ff:ff
4: enp0s9: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:9d:3d:33 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:86:80:ce brd ff:ff:ff:ff:ff:ff
root@deb10:~# _
```

Ip link show:

```
root@deb10:~# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group d
efault qlen 1000
    link/ether 08:00:27:64:ff:a6 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:00:27:12:72:2b brd ff:ff:ff:ff:ff:ff
4: enp0s9: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:00:27:9d:3d:33 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 100
0
    link/ether 08:00:27:86:80:ce brd ff:ff:ff:ff:ff:ff
root@deb10:~#
```

Ip link list:

```
root@deb10:~# ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group d
efault qlen 1000
    link/ether 08:00:27:64:ff:a6 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:00:27:12:72:2b brd ff:ff:ff:ff:ff:ff
4: enp0s9: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:00:27:9d:3d:33 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 100
0
    link/ether 08:00:27:86:80:ce brd ff:ff:ff:ff:ff:ff
root@deb10:~# _
```

Ip link set dev enp0s3: (no output)

Ip route:

```
default via 10.0.2.2 dev enp0s3
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15
root@deb10:~#
```

Ip route show: same as previous

Ip route list: same but without the first row

Ifconfig: should show the data about ip’s, Mac addresses... But not working on Debian until configured

To set the static ip on debian linux one should open etc/network/interfaces with nano(terminal text editor):

Nano etc/network/interfaces

And instead of the last row place:

iface enp0s3 inet static

address 192.168.0.11

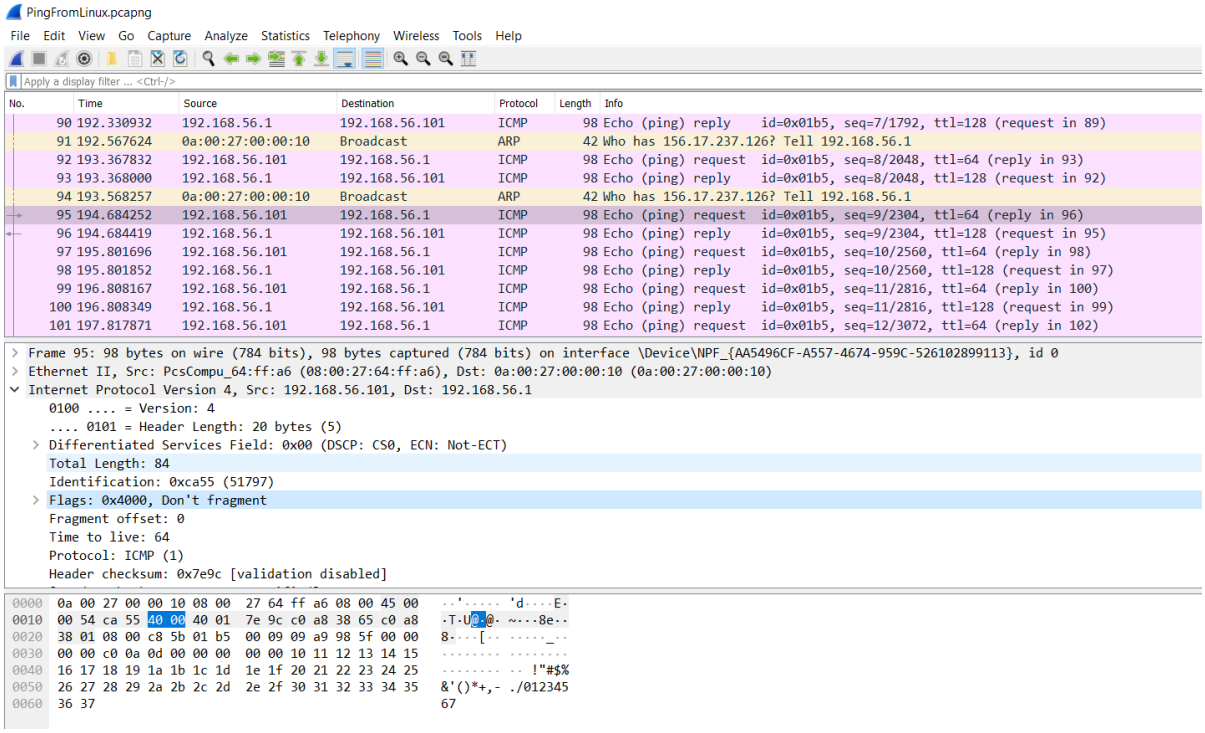
netmask 255.255.255.0

gateway 192.168.0.10

dns-nameservers 8.8.8.8

I won’t apply settings because in non-lab conditions it will brake my connection

Pinging my real machine from a virtual machine:



- Source ip: 192.168.56.101
- Destination ip: 192.168.56.1
- TTL fiedld: 128 on windows 64 on linux
- Types and names: 8 Echo(ping) request, 0 Echo(ping) reply
- 48 bytes, like ascii table

```
stud@deb10:~$ ping 192.168.56.1
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=128 time=0.518 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=128 time=0.745 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=128 time=0.299 ms
64 bytes from 192.168.56.1: icmp_seq=4 ttl=128 time=0.275 ms
64 bytes from 192.168.56.1: icmp_seq=5 ttl=128 time=0.460 ms
64 bytes from 192.168.56.1: icmp_seq=6 ttl=128 time=0.342 ms
64 bytes from 192.168.56.1: icmp_seq=7 ttl=128 time=0.587 ms
64 bytes from 192.168.56.1: icmp_seq=8 ttl=128 time=0.605 ms
64 bytes from 192.168.56.1: icmp_seq=9 ttl=128 time=0.551 ms
64 bytes from 192.168.56.1: icmp_seq=10 ttl=128 time=0.533 ms
64 bytes from 192.168.56.1: icmp_seq=11 ttl=128 time=0.619 ms
64 bytes from 192.168.56.1: icmp_seq=12 ttl=128 time=0.419 ms
64 bytes from 192.168.56.1: icmp_seq=13 ttl=128 time=0.335 ms
64 bytes from 192.168.56.1: icmp_seq=14 ttl=128 time=0.396 ms
64 bytes from 192.168.56.1: icmp_seq=15 ttl=128 time=0.633 ms
64 bytes from 192.168.56.1: icmp_seq=16 ttl=128 time=0.270 ms
64 bytes from 192.168.56.1: icmp_seq=17 ttl=128 time=0.385 ms
64 bytes from 192.168.56.1: icmp_seq=18 ttl=128 time=0.944 ms
^C
--- 192.168.56.1 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 307ms
rtt min/avg/max/mdev = 0.270/0.495/0.944/0.174 ms
stud@deb10:~$ _
```

The difference between the linux and the windows ping packets that I've noticed is following

The default data field size in linux is 48 bytes, whereas it’s 32 in windows, and it send’s the different kind of data (looks like in the order of ascii table)

Also in linux by defauld the non fragment bit is set to 1

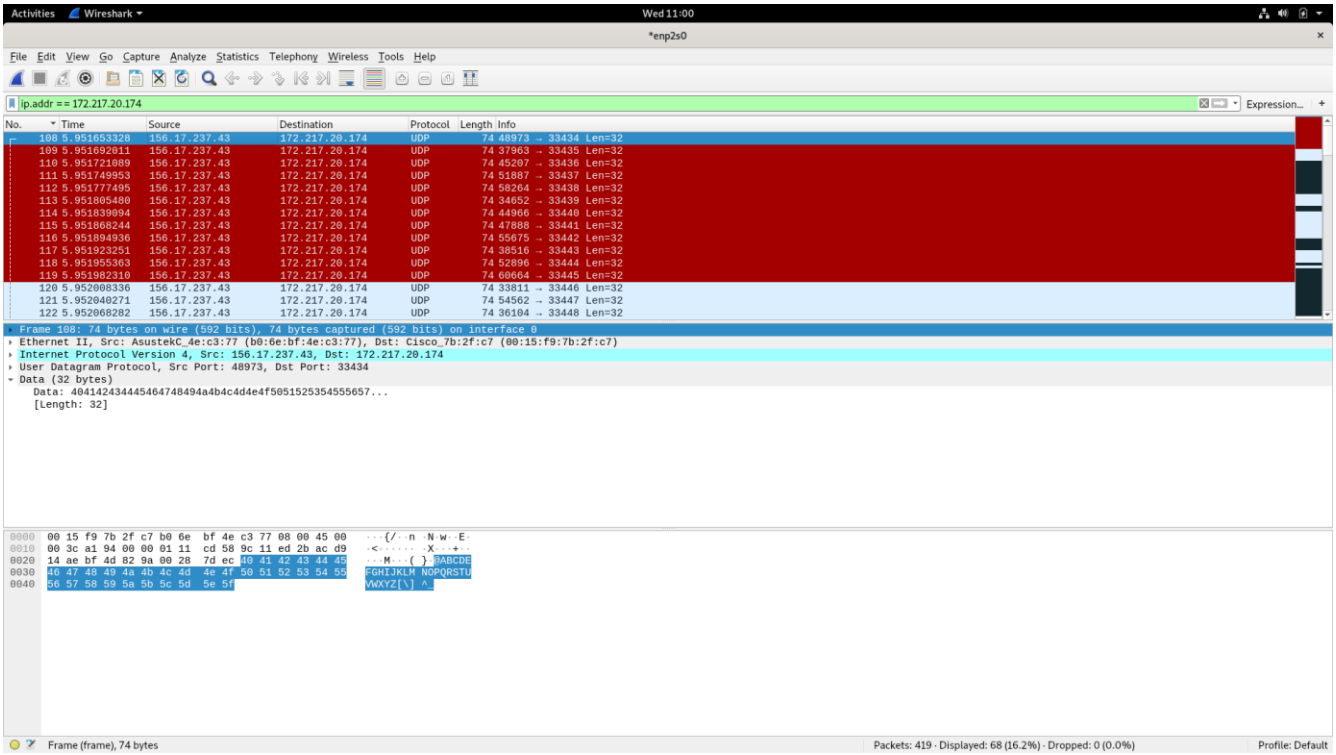
In linux time to live by default is 64 whereas in windows 128

Task 5:

Tracerouting youtube.com on debian linux machine:

```
danylo@danylo:~$ traceroute youtube.com
traceroute to youtube.com (172.217.20.174), 30 hops max, 60 byte packets
 1  xxxx.t19.ds.pwr.wroc.pl (156.17.237.126)  0.752 ms  1.357 ms  1.513 ms
 2  234.ds.pwr.wroc.pl (156.17.229.234)  0.732 ms  0.646 ms  0.626 ms
 3  t15-wittiga2.ds.pwr.wroc.pl (156.17.229.241)  12.599 ms  12.583 ms  12.569 ms
 4  156.17.229.255 (156.17.229.255)  1.299 ms  1.698 ms  2.359 ms
 5  pwr-zds-vprn-centrum3.wask.wroc.pl (156.17.254.40)  2.359 ms pwr-zds-centrum3-vprn.wask.wroc.pl (156.17.254.41)  0.657 ms  0.644 ms
 6  pwr-zds-centrum3-vprn.wask.wroc.pl (156.17.254.41)  0.624 ms z-wroclawia.poznan-gw3.10Gb.rtr.pionier.gov.pl (212.191.224.105)  5.663 ms  5.646 ms
 7  z-wroclawia.poznan-gw3.10Gb.rtr.pionier.gov.pl (212.191.224.105)  5.626 ms  5.606 ms  5.040 ms
 8  108.170.250.193 (108.170.250.193)  9.223 ms  9.178 ms 72.14.203.178 (72.14.203.178)  9.159 ms
 9  108.170.250.193 (108.170.250.193)  9.285 ms 216.239.41.165 (216.239.41.165)  9.062 ms  9.191 ms
10  216.239.41.167 (216.239.41.167)  9.139 ms waw02s07-in-f14.1e100.net (172.217.20.174)  8.984 ms  9.383 ms
danylo@danylo:~$
```

Capturing packets in wireshark:



- Source: 156.17.237.43

- Destination ip: 172.217.20.174

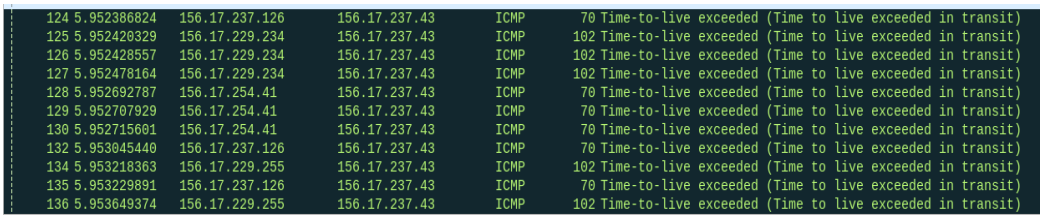
- Just as in windows, in case with linux ttl field goes from 1 and higher because it makes up the principle of work of tracerouting.

-There are no ICMP packets sent, in linux for the tracerouting the UDP (User datagram protocol) packets are sent instead. (There's no types and names, it carries information about the source and the destination port)

-Data field of the UDP protocol is 32 bytes and contains uppercase alphabet and some symbols

ICMP reply packets:

- Source ip's: Varying from the fact on which node the ttl fiedl became 0.



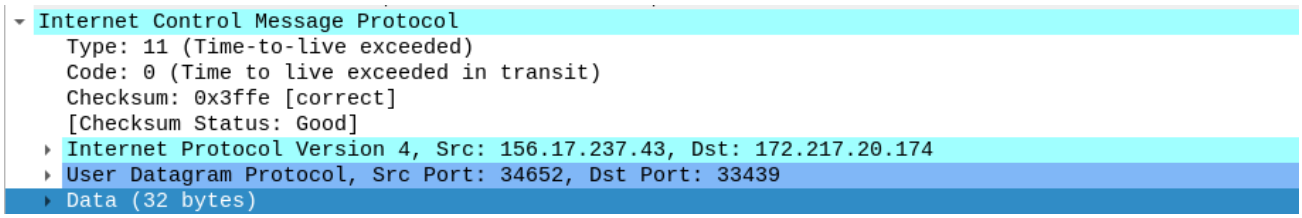
124	5.952386824	156.17.237.126	156.17.237.43	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
125	5.952420329	156.17.229.234	156.17.237.43	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
126	5.952428557	156.17.229.234	156.17.237.43	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
127	5.952478164	156.17.229.234	156.17.237.43	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
128	5.952692787	156.17.254.41	156.17.237.43	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
129	5.952707929	156.17.254.41	156.17.237.43	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
130	5.952715601	156.17.254.41	156.17.237.43	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
132	5.953045440	156.17.237.126	156.17.237.43	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
134	5.953218363	156.17.229.255	156.17.237.43	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
135	5.953229891	156.17.237.126	156.17.237.43	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
136	5.953649374	156.17.229.255	156.17.237.43	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

- Destination: 156.17.237.43 (my ip)

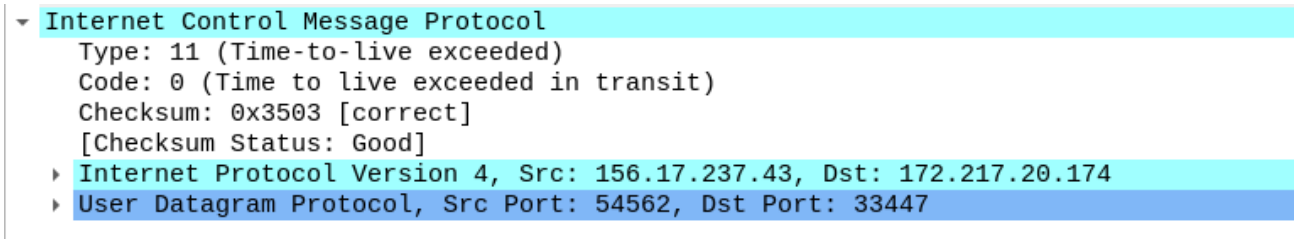
- Value of TTL field: different varying on the node it comes from i've seen 255, 254, 61

- Types and names: 11 (Time-to-live exceeded)

- Size and content of ICMP data field: some don't send any "data fields" but send back the ipv4 and udp headers, some do the same thing but also send back the data coming from the packet.



Or



The question about Why do intermediate nodes respond to traceroute packets was answered in the same task on windows.

-Unlike the traceroute which is usually installed on the machines MTR is the utility that people usually install manually. It includes both ping and traceroute and combines information from these two operations.