

INZ004407L

## Computer Networks

### Network layer of the ISO-OSI model

Group Number = 4

Mustafa Tayyip BAYRAM 257639 => IPv4 = 25.49.179.40 => IPv6 = 2620:9b::1931:b328 => PC\_42

Furkan ÖCALAN 257638 => IPv4 = 25.48.259.244 => IPv6 = 2620:9b::1930:f9f4 => PC\_41

LinuxPC => IPv4 => 25.41.182.15 (Hamachi NIC) <=> 192.168.137.182(Ethernet NIC )

#### Task 1 – Connectivity test in IPv4 network. ICMP protocol

2. Configure the static IP addresses on internal interfaces (Cisco) under MS Windows system. Use the addresses given in the instruction. Use the netsh command. Save the exact configuration formula. Check the correctness of configuration using ipconfig command.

[ We cannot configure IP addresses as a static because we are connected through Hamachi.]

4. Make the connectivity test between PC\_X1 and PC\_X2 computers. Use the ping command.

*We are connected each other through Hamachi. We are tested it using ICMP protocol and Wireshark program.*

```
C:\Users\burak>ping 25.49.179.40

Pinging 25.49.179.40 with 32 bytes of data:
Reply from 25.49.179.40: bytes=32 time<1ms TTL=128
Reply from 25.49.179.40: bytes=32 time<1ms TTL=128
Reply from 25.49.179.40: bytes=32 time=1ms TTL=128
Reply from 25.49.179.40: bytes=32 time<1ms TTL=128

Ping statistics for 25.49.179.40:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

6. Find inside the Wireshark listing ICMP packets exchanged between PC\_X1 and PC\_X2. Note the frames IDs.

Request frame ID's {1 ,3, 5}

Reply frame ID's {2 ,4, 6}

7. Analyze the IPv4 and ICMP headers. Find and save the following fields values:

- Source IPv4 address; 25.48.249.244
  - Destination IPv4 address; 25.49.179.40
  - Value of TTL field; 128
  - Types and names of ICMP packets; Type 0 (Echo(ping) reply )) and Type 8 (Echo(ping) request )
  - Size and content of ICMP data field.; Data (32 bytes)
- Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
- [Length: 32]

8. Make the connectivity test to google.com server. Enlarge the size of ping packet and set the non-fragmentation bit. Check and save the maximum size of ping packet

```
C:\Users\burak>ping -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
Reply from 216.58.215.78: bytes=68 (sent 1472) time=10ms TTL=117
Reply from 216.58.215.78: bytes=68 (sent 1472) time=9ms TTL=117

Ping statistics for 216.58.215.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 9ms, Maximum = 10ms, Average = 9ms

C:\Users\burak>ping -l 1473 google.com

Pinging google.com [216.58.215.78] with 1473 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 216.58.215.78:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

9. Start the Wireshark on internal NIC. Capture the transmitted frames with increased size. Save the result in the user folder.

- Source IPv4 address; 25.48.249.244
  - Destination IPv4 address; 239.255.255.250
  - Value of TTL field; 117
  - Value of the non-fragment bit; 0
  - Types and names of ICMP packets; Type 0 (Echo(ping) reply )) and Type 8 (Echo(ping) request )
  - Maximal size and content of ICMP data field.; Data (32 bytes)
- Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
- [Length: 32]

## Task 2 – Connectivity test in IPv6 network. ICMPv6 protocol.

1. Create a text document in the user's home directory and save the course of exercise in it.

4. Make the connectivity test between PC\_X1 and PC\_X2 computers. Use the ping command.

```
C:\Windows\system32>ping 2620:9b::1930:f9f4
```

```
Pinging 2620:9b::1930:f9f4 with 32 bytes of data:
Reply from 2620:9b::1930:f9f4: time=108ms
Reply from 2620:9b::1930:f9f4: time=24ms
Reply from 2620:9b::1930:f9f4: time=40ms
Reply from 2620:9b::1930:f9f4: time=58ms
```

6. Find inside the Wireshark listing ICMPv6 packets exchanged between PC\_X1 and PC\_X2. Note the frames IDs.

Request frame ID's {1 ,3, 5, 7}

Reply frame ID's {2 ,4, 6,8}

7. Analyze the IPv6 and ICMP headers. Find and save the following fields values:

– Source IPv6 address; *2620:9b::1930:f9f4*

– Destination IPv6 address; *2620:9b::1931:b328*

– Value of TTL field; - Hop Limit: 128 [*TTL is not actually thrown out in IPv6, it has just been renamed. The field is called the "hop limit" and has the same function as the TTL field in IPv4.*]

– Types and names of ICMPv6 packets; Type 129 (Echo(ping) reply )) and Type 128 (Echo(ping) request )

– Size and content of ICMPv6 data field.;

Data (32 bytes)

Data: 6162636465666768696a6b6c6d6e6f707172737475767761...

[Length: 32]

8. Make the connectivity test to PC\_X2. Enlarge the size of ping packet and set the non-fragmentation bit.

```
C:\Users\burak>ping -l 65500 2620:9b::1931:b328
```

```
Pinging 2620:9b::1931:b328 with 65500 bytes of data:
Reply from 2620:9b::1931:b328: time=14ms
Reply from 2620:9b::1931:b328: time=15ms
Reply from 2620:9b::1931:b328: time=15ms
Reply from 2620:9b::1931:b328: time=14ms
```

```
Ping statistics for 2620:9b::1931:b328:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 15ms, Average = 14ms
```

Check and save the maximum size of ping packet.

Note: Packet fragmentation in the IPv6 protocol works on a different way than in the IPv4 protocol.

Nonfragment

Maximal size and content of ICMPv6 data field

9. Start the Wireshark on internal NIC. Capture the transmitted frames with increased size. Save the result in the user folder.

### Task 3 – Tracking the path in IPv4 network. Traceroute (tracert) command

4. In MS Windows system trace the network path to the any chosen Web server using tracert command.

```
C:\Users\burak>tracert google.com

Tracing route to google.com [216.58.215.78]
over a maximum of 30 hops:

  0  1 ms  <1 ms  <1 ms  ipb254.t17.ds.pwr.wroc.pl [156.17.235.254]
  1  1 ms  <1 ms  *      234.ds.pwr.wroc.pl [156.17.229.234]
  2  14 ms  21 ms  9 ms  t15-wittiga2.ds.pwr.wroc.pl [156.17.229.241]
  3  3 ms  1 ms  1 ms  156.17.229.255
  4  1 ms  1 ms  <1 ms  pwr-zds-centrum3-vprn.wask.wroc.pl [156.17.254.41]
  5  6 ms  5 ms  5 ms  z-wroclawia.poznan-gw3.10Gb.rtr.pionier.gov.pl [212.191.224.105]
  6  10 ms  10 ms  10 ms  72.14.203.178
  7  10 ms  11 ms  11 ms  108.170.250.209
  8  10 ms  10 ms  10 ms  108.170.234.103
  9  10 ms  10 ms  10 ms  waw02s16-in-f14.1e100.net [216.58.215.78]

Trace complete.
```

6. Find inside the Wireshark listing traceroute packets exchanged between PC\_X1 and Web server. Note the frames IDs.

Sent Frame ID's : 34, 36, 38, 55, 57, 59, 117, 119, 121, 145, 148, 150, 267, 269, 272, 291, 293, 295

Received Frame ID's :35, 37, 39, 56, 58, 118, 120, 124, 147, 149, 151, 268, 270, 273, 292, 294, 296

7. Analyze the headers of first six sent *traceroute* packets. Find and save the values of the following fields:

8. Analyze the headers of firs six received traceroute packets. Find and save the values of the following fields:

9. Answer the question: Why do intermediate nodes respond to traceroute packets even though they are not the destinations of these packets?

Actually in traceroute programs, we can say every intermediate nodes are destinations by natura. Because its purpose is checking the path. It must take responds from intermediary nodes because calculating some properties such as response time.

## Task 4 – Connectivity test in IPv4 network between Linux hosts. ICMP protocol

7. Make the connectivity test between PC\_X1 and PC\_X2 using *ping* command.

We are connected. [ TASK4\_7 ] file.

8. Make the connectivity test to google.com Web Server.

[ TASK4\_8 ]

11. Analyze the IPv4 and ICMP headers. Find and save the values of the following fields:

Request Frame ID's :2,4,7,9,11

Received Frame ID's :3,5,8,10,12

[ TASK4\_7 ]

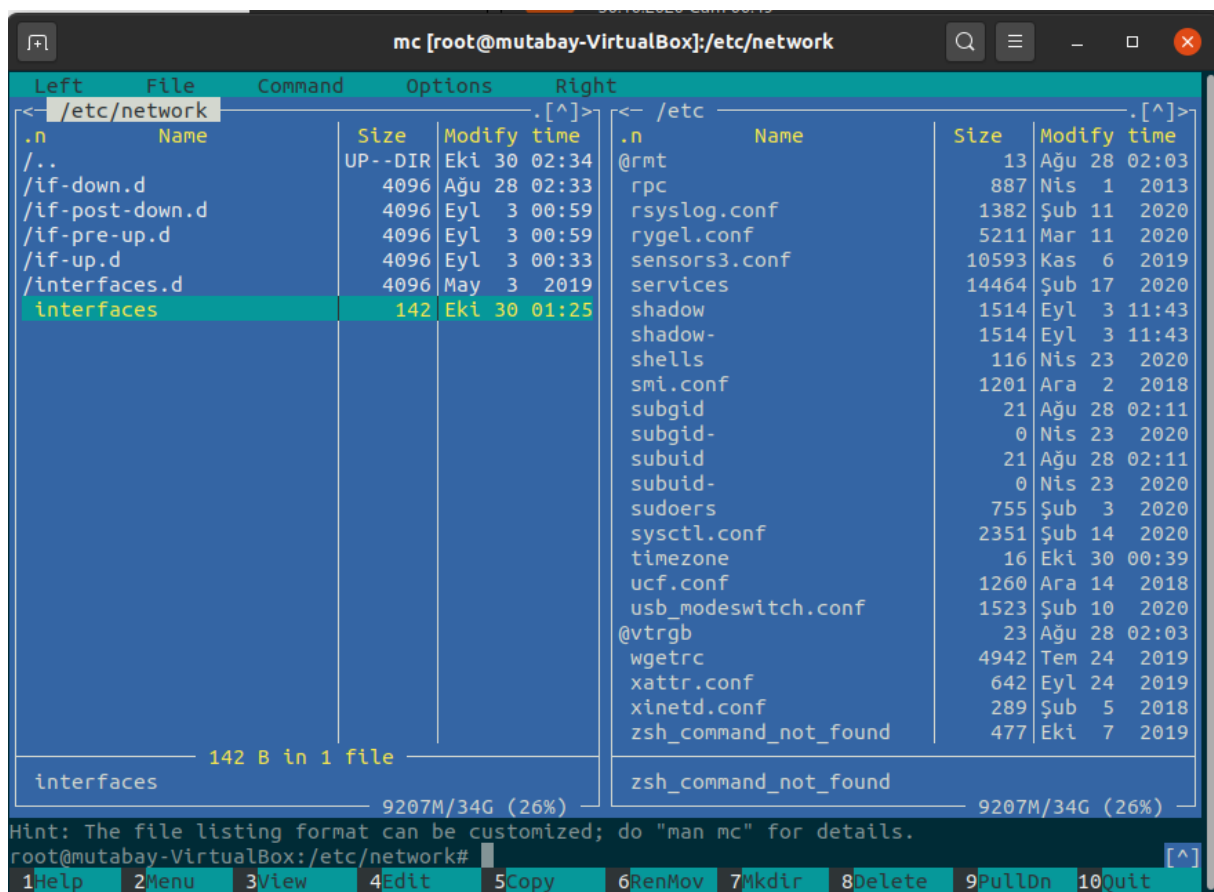
12. Find and write down the differences in the structure of Linux and MS Windows ICMP packets.

TTL values are different. TTL is 128 when requested from windows but TTL is 64 when requested from linux . Data sizes are different. Main difference is ping command. Ping command has differences in Linux and Windows.

## Task 5 – Tracking the IPv4 network path from Linux OS.

### Traceroute command

5. To edit the file, you can use the built-in text editor from the **mc** program. Configure the static IPv4 address on internal NIC (Cisco). Configure dynamic IPv4 address on external NIC (Internet). Use the addresses given in the instructions. Save the created configuration. Reset the interfaces. Check the correctness of the new configuration with command:



```
mc [root@mutabay-VirtualBox]:/etc/network

Left      File      Command  Options  Right
<- /etc/network .[^]> <- /etc .[^]>
.n      Name      Size      Modify   time
UP--DIR Eki 30 02:34
/if-down.d      4096     Ağu 28 02:33
/if-post-down.d 4096     Eyl 3 00:59
/if-pre-up.d    4096     Eyl 3 00:59
/if-up.d        4096     Eyl 3 00:33
/interfaces.d    4096     May 3 2019
interfaces      142     Eki 30 01:25

142 B in 1 file

9207M/34G (26%)

Hint: The file listing format can be customized; do "man mc" for details.
root@mutabay-VirtualBox:/etc/network#

1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Quit
```

9. Find inside the Wireshark listing *traceroute* packets exchanged between PC\_X1 and Web server. Note the frames IDs. Use the frame filter.

Request Frame ID's : 4,5,6,7,8,9,1

,11,12,13,14,15,16,17,18,19,20,21,30,31

Received Frame ID's : 22,23,24,25,26,27

10. Analyze the headers of first six sent *traceroute* packets. Find and save the values of the following fields:

[ TASK5\_9 ]

11. Analyze the headers of first six received *traceroute* packets. Find and save the values of the following fields:

[ TASK5\_9 ]

12. Write down the differences in the operation of the *tracert* (MS Windows) and *traceroute* (Linux) programs.

Linux *traceroute* command and Windows *tracert* commands both trace the network paths, but they do it in slightly different ways. Difference is that *traceroute* uses UDP packets to a random high port number, while Microsoft Windows uses ICMP packets. ( Default )

13. Answer the question: Why do intermediate nodes respond to *traceroute* packets even though they are not the destinations of these packets?

Actually in *traceroute* programs, we can say every intermediate node is a destination by nature. Because its purpose is checking the path. It must take responses from intermediary nodes because calculating some properties such as response time.

14. Check the *mtr* program operation. Write down the differences in the operation of *traceroute* and *mtr*. In the absence of a program, try to install it:

```
aptitude install -R mtr
```

*mtr* combines the functionality of the *traceroute* and *ping* programs in a single network diagnostic tool. Advantage of *mtr* over *traceroute* is it shows where exactly the packet loss is happening in the route to the destination host in real-time. Also *mtr* is faster than *traceroute* because *traceroute* sends at least 3 packets for each hop, *mtr* discovers the hops in the route first, and then sends packets to each node in parallel.