|  |
| --- |
| ECE 358 S20 |
| Encapsulation and Network Utilities |
| Lab 3 |

|  |
| --- |
| Arjun Bawa, 20711916  (no partner)  7-20-2020 |

Contents

[Question 1 2](#_Toc46096952)

[Frame 6 2](#_Toc46096953)

[Link Layer (Layer 2/Ethernet header) 2](#_Toc46096954)

[Network Layer (Layer 3/IP header) 2](#_Toc46096955)

[Transport Layer (Layer 4/UDP header) 3](#_Toc46096956)

[Application Layer (Layer 5/Data) 3](#_Toc46096957)

[Frame 16 4](#_Toc46096958)

[Link Layer (Layer 2/Ethernet header) 4](#_Toc46096959)

[Network Layer (Layer 3/IP header) 4](#_Toc46096960)

[Transport Layer (Layer 4/TCP header) 5](#_Toc46096961)

[Application Layer (Layer 5/Data) 5](#_Toc46096962)

[Question 2 (arp) 6](#_Toc46096963)

[Question 3 (ifconfig) 6](#_Toc46096964)

[Question 4 (netstat) 7](#_Toc46096965)

[Question 5 (nslookup) 10](#_Toc46096966)

[Question 6 (ping) 11](#_Toc46096967)

[Question 7 (traceroute) 12](#_Toc46096968)

# Question 1

I’m examining frames 6 and 16.

## Frame 6

UDP is the highest layer protocol here.

00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00

00 ca 5c 0d 00 00 80 11 00 00 0a 20 1b a0 0a 20

7f ff eb 35 00 8a 00 b6 8e c0 11 0a 80 22 0a 20

1b a0 00 8a 00 a0 00 00 20 46 47 45 4a 46 43 46

45 46 46 45 42 45 4d 46 49 46 41 43 4e 44 49 44

45 44 41 44 46 44 44 41 41 00 20 46 48 45 50 46

43 45 4c 45 48 46 43 45 50 46 46 46 41 43 41 43

41 43 41 43 41 43 41 43 41 42 4e 00 ff 53 4d 42

25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 11 00 00 06

00 00 00 00 00 00 00 00 00 e8 03 00 00 00 00 00

00 00 00 06 00 56 00 03 00 01 00 01 00 02 00 17

00 5c 4d 41 49 4c 53 4c 4f 54 5c 42 52 4f 57 53

45 00 09 04 02 00 00 00

### Link Layer (Layer 2/Ethernet header)

Destination Address: **00 00 00 00 00 00** 00 00 00 00 00 00 08 00

* 0x000000000000 means localhost

Source Address: 00 00 00 00 00 00 **00 00 00 00 00 00** 08 00

* 0x000000000000 means localhost

Type of Payload: 00 00 00 00 00 00 00 00 00 00 00 00 **08 00**

* 0x0800 means type is IPv4

### Network Layer (Layer 3/IP header)

IP Version: **4**5 00 00 ca 5c 0d 00 00 80 11 00 00 0a 20 1b a0 0a 20 7f ff

* 4 🡪 IPv4

Internet Header Length: 4**5** 00 00 ca 5c 0d 00 00 80 11 00 00 0a 20 1b a0 0a 20 7f ff

* Header length is 5 × 4 = 20 bytes 🡪 there is no *options* field in this header

Type of Service: 45 **00** 00 ca 5c 0d 00 00 80 11 00 00 0a 20 1b a0 0a 20 7f ff

* In binary: 0 0 0 0 0 0 0 0
* **0 0 0** 0 0 0 0 0 🡪 The datagram has *routine* (the lowest) precedence.
* 0 0 0 **0** 0 0 0 0 🡪 Normal Delay
* 0 0 0 0 **0** 0 0 0 🡪 Normal Throughput
* 0 0 0 0 0 **0** 0 0 🡪 Normal Reliability
* 0 0 0 0 0 0 **0 0** 🡪 Unused bits

Total Length: 45 00 **00 ca** 5c 0d 00 00 80 11 00 00 0a 20 1b a0 0a 20 7f ff

* 0x00ca = 202 in decimal 🡪 IP datagram is 202 bytes long

Identification: 45 00 00 ca **5c 0d** 00 00 80 11 00 00 0a 20 1b a0 0a 20 7f ff

* 0x5c0d is identity assigned by the sender for reconstruction of a fragmented datagram

Flags & Fragment Offset: 45 00 00 ca 5c 0d **00 00** 80 11 00 00 0a 20 1b a0 0a 20 7f ff

* In binary: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
* **0** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 🡪 Reserved bit (must be zero)
* 0 **0** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 🡪 May fragment
* 0 0 **0** 0 0 0 0 0 0 0 0 0 0 0 0 0 🡪 Last fragment
* Fragment offset is zero, and we know there are no fragments after this datagram (last fragment bit from earlier). This means while the datagram could’ve been fragmented (may fragment bit from earlier), it wasn’t, and this is the last and only fragment of the datagram (because the offset is zero).

Time to Live: 45 00 00 ca 5c 0d 00 00 **80** 11 00 00 0a 20 1b a0 0a 20 7f ff

* 0x80 = 128 in decimal; datagram will exist for 128 more hops

Protocol: 45 00 00 ca 5c 0d 00 00 80 **11** 00 00 0a 20 1b a0 0a 20 7f ff

* 0x11 🡪 UDP

Header Checksum: 45 00 00 ca 5c 0d 00 00 80 11 **00 00** 0a 20 1b a0 0a 20 7f ff

* Since this is UDP, 0x0000 means the checksum isn’t calculated

Source Address: 45 00 00 ca 5c 0d 00 00 80 11 00 00 **0a 20 1b a0** 0a 20 7f ff

* Source IP address is 10.32.27.160

Destination Address: 45 00 00 ca 5c 0d 00 00 80 11 00 00 0a 20 1b a0 **0a 20 7f ff**

* Destination IP address is 10.32.127.255

### Transport Layer (Layer 4/UDP header)

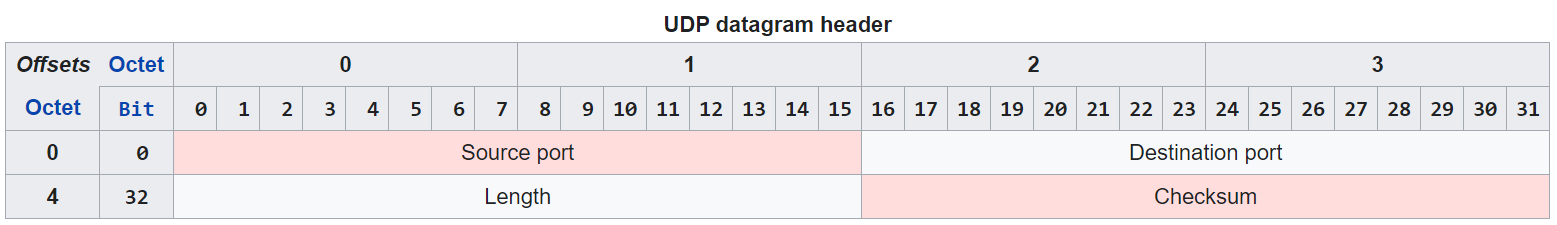
[](https://en.wikipedia.org/wiki/User_Datagram_Protocol#UDP_datagram_structure)

Figure 1 UDP datagram structure, Wikipedia contributors. (2020, June 22). User Datagram Protocol. In Wikipedia, The Free Encyclopedia. Retrieved 19:33, July 17, 2020, from <https://en.wikipedia.org/w/index.php?title=User_Datagram_Protocol>

Source Port: **eb 35** 00 8a 00 b6 8e c0

* 0xeb35 = 60213 in decimal; source port is 60213

Destination Port: eb 35 **00 8a** 00 b6 8e c0

* 0x008a = 138 in decimal; destination port is 138

Length: eb 35 00 8a **00 b6** 8e c0

* 0x00b6 = 182 in decimal 🡪 UDP datagram is 182 bytes

Checksum: eb 35 00 8a 00 b6 **8e c0**

* 0x8ec0 is the checksum of the UDP datagram

### Application Layer (Layer 5/Data)

11 0a 80 22 0a 20 1b a0 00 8a 00 a0 00 00 20 46

47 45 4a 46 43 46 45 46 46 45 42 45 4d 46 49 46

41 43 4e 44 49 44 45 44 41 44 46 44 44 41 41 00

20 46 48 45 50 46 43 45 4c 45 48 46 43 45 50 46

46 46 41 43 41 43 41 43 41 43 41 43 41 43 41 42

4e 00 ff 53 4d 42 25 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 11 00 00 06 00 00 00 00 00 00 00 00 00 e8

03 00 00 00 00 00 00 00 00 06 00 56 00 03 00 01

00 01 00 02 00 17 00 5c 4d 41 49 4c 53 4c 4f 54

5c 42 52 4f 57 53 45 00 09 04 02 00 00 00

## Frame 16

TCP is the higher layer protocol here.

00 1d 7e 46 ec 49 00 22 19 f4 30 97 08 00 45 08

0b 84 01 04 40 00 40 06 b2 72 c0 a8 01 88 81 61

38 64 bc 0d 00 14 7e 3f 10 c4 e3 1f 73 21 80 10

00 e3 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16

96 ff …

### Link Layer (Layer 2/Ethernet header)

Destination Address: **00 1d 7e 46 ec 49** 00 22 19 f4 30 97 08 00

Source Address: 00 1d 7e 46 ec 49 **00 22 19 f4 30 97** 08 00

Type of Payload: 00 1d 7e 46 ec 49 00 22 19 f4 30 97 **08 00**

* 0x0800 means type is IPv4

### Network Layer (Layer 3/IP header)

IP Version: **4**5 08 0b 84 01 04 40 00 40 06 b2 72 c0 a8 01 88 81 61 38 64

* 4 🡪 IPv4

Internet Header Length: 4**5** 08 0b 84 01 04 40 00 40 06 b2 72 c0 a8 01 88 81 61 38 64

* Header length is 5 × 4 = 20 bytes 🡪 there is no *options* field in this header

Type of Service: 45 **08** 0b 84 01 04 40 00 40 06 b2 72 c0 a8 01 88 81 61 38 64

* 0x08 🡪 0 0 0 0 1 0 0 0 in binary
* **0 0 0** 0 1 0 0 0 🡪 The datagram has *routine* (the lowest) precedence.
* 0 0 0 **0** 0 0 0 0 🡪 Normal Delay
* 0 0 0 0 **1** 0 0 0 🡪 High Throughput
* 0 0 0 0 0 **0** 0 0 🡪 Normal Reliability
* 0 0 0 0 0 0 **0 0** 🡪 Unused bits

Total Length: 45 08 **0b 84** 01 04 40 00 40 06 b2 72 c0 a8 01 88 81 61 38 64

* 0x0b84 = 2948 in decimal 🡪 IP datagram is 2948 bytes long

Identification: 45 08 0b 84 **01 04** 40 00 40 06 b2 72 c0 a8 01 88 81 61 38 64

* 0x0104 is identity assigned by the sender for reconstruction of a fragmented datagram

Flags & Fragment Offset: 45 08 0b 84 01 04 **40 00** 40 06 b2 72 c0 a8 01 88 81 61 38 64

* In binary: 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
* **0** 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 🡪 Reserved bit (must be zero)
* 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 🡪 Don’t fragment
* 0 0 **0** 0 0 0 0 0 0 0 0 0 0 0 0 0 🡪 Last fragment
* Fragment offset is zero, and we know there are no other fragments after this datagram (don’t fragment bit & last fragment bit). This means the datagram could not have been fragmented and this is the last and only fragment of the datagram (by definition & the offset being zero).

Time to Live: 45 08 0b 84 01 04 40 00 **40** 06 b2 72 c0 a8 01 88 81 61 38 64

* 0x40 = 64 in decimal; datagram will exist for 64 more hops

Protocol: 45 08 0b 84 01 04 40 00 40 **06** b2 72 c0 a8 01 88 81 61 38 64

* 0x06 🡪 TCP

Header Checksum: 45 08 0b 84 01 04 40 00 40 06 **b2 72** c0 a8 01 88 81 61 38 64

* 0xb272 is the header’s checksum

Source Address: 45 08 0b 84 01 04 40 00 40 06 b2 72 **c0 a8 01 88** 81 61 38 64

* Source IP address is 192.168.1.136

Destination Address: 45 08 0b 84 01 04 40 00 40 06 b2 72 c0 a8 01 88 **81 61 38 64**

* Destination IP address is 129.97.56.100

### Transport Layer (Layer 4/TCP header)

Source Port: **bc 0d** 00 14 7e 3f 10 c4 e3 1f 73 21 80 10 00 e3 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* 0xbc0d = 48141 in decimal; source port is 48141

Destination Port: bc 0d **00 14** 7e 3f 10 c4 e3 1f 73 21 80 10 00 e3 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* 0x0014 = 20 in decimal; destination port is 20

Sequence Number: bc 0d 00 14 **7e 3f 10 c4** e3 1f 73 21 80 10 00 e3 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* 0x7e3f10c4 = 2118062276; sequence number is 2118062276

Acknowledgement Number: bc 0d 00 14 7e 3f 10 c4 **e3 1f 73 21** 80 10 00 e3 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* 0xe31f7321 = 3810489121; acknowledgement number is 3810489121

Data Offset & Control Bits: bc 0d 00 14 7e 3f 10 c4 e3 1f 73 21 **80 10** 00 e3 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* In binary: 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
* Data Offset (4 bits): 1 0 0 0 = 8 in decimal 🡪 8 × 4 = 32 bytes; Length of TCP header is 32 bytes
* Reserved (6 bits): 1 0 0 0 **0 0 0 0 0 0** 0 1 0 0 0 0
* Control Bits (6 bits): 0 1 0 0 0 0 🡪 URG flag 0, *ACK flag 1*, PSH flag 0, RST flag 0, SYN flag 0, FIN flag 0
  + Indicates successful receipt of the packet by host; acknowledgement number field has a valid number

Window: bc 0d 00 14 7e 3f 10 c4 e3 1f 73 21 80 10 **00 e3** 87 6c 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* 0x00e3 = 227 in decimal; receiver window size is 227 bytes

Checksum: bc 0d 00 14 7e 3f 10 c4 e3 1f 73 21 80 10 00 e3 **87 6c** 00 00 01 01 08 0a 00 14 71 bf 09 16 96 ff

* Checksum of entire TCP segment is 0x876c

Urgent Pointer: bc 0d 00 14 7e 3f 10 c4 e3 1f 73 21 80 10 00 e3 87 6c **00 00** 01 01 08 0a 00 14 71 bf 09 16 96 ff

* 0x0000 🡪 not used for this segment

Rest of header bytes: bc 0d 00 14 7e 3f 10 c4 e3 1f 73 21 80 10 00 e3 87 6c 00 00 **01 01 08 0a 00 14 71 bf 09 16 96 ff**

* These bytes are part of the options and padding, indicated by the total length of the TCP header

### Application Layer (Layer 5/Data)

Omitted here for brevity.

# Question 2 (arp)

1. Explain the functions of the utility
   * ARP (Address Resolution Protocol) is a type of communication protocol for determining the MAC address associated with some IP address. It is a mapping between the Internet Protocol address and the Media Access Control address. When a network device maps a MAC address to an IPv4 address, it caches it in it’s ARP table. The *arp* command is used to manipulate the entries in this table. It can add, delete and display the current entries in the table.
2. Use the command */sbin/arp –a* to see the ARP table of the machine on which you are logged in. Include the output of the command in your report and explain it.

exsw02-circuitnet.uwaterloo.ca (129.97.56.1) at b4:99:ba:52:2c:00 [ether] on eno1

ecelinux.uwaterloo.ca (129.97.56.13) at 00:0a:cd:2a:f9:1e [ether] on eno1

sca.uwaterloo.ca (129.97.56.46) at 52:54:00:f6:3d:9b [ether] on eno1

ecelinux1.uwaterloo.ca (129.97.56.15) at 00:0a:cd:2a:f8:ee [ether] on eno1

ece252-3.uwaterloo.ca (129.97.56.53) at 52:54:00:9d:b5:bb [ether] on eno1

eceserv1.uwaterloo.ca (129.97.56.9) at 00:25:90:5d:b6:2d [ether] on eno1

ecetesla3.uwaterloo.ca (129.97.56.11) at 00:0a:cd:2a:f9:fd [ether] on eno1

eceubuntu.uwaterloo.ca (129.97.56.12) at 00:0a:cd:2a:fa:f3 [ether] on eno1

ece252-1.uwaterloo.ca (129.97.56.51) at 52:54:00:10:f8:27 [ether] on eno1

ecelinux4.uwaterloo.ca (129.97.56.16) at 52:54:00:d3:2d:36 [ether] on eno1

ece252-2.uwaterloo.ca (129.97.56.52) at 52:54:00:79:c2:41 [ether] on eno1

ecelinux4.uwaterloo.ca (129.97.56.14) at 52:54:00:51:c8:11 [ether] on eno1

ecesystem.uwaterloo.ca (129.97.56.7) at 52:54:00:0c:98:ec [ether] on eno1

* The *-a* option means all; displays arp cache tables for all the interfaces
* The format of this output is [*hostname] [IP address]* at *[physical/MAC address]* *[connection type]* on *[interface]*
* **[ether]** means the connection between the current machine and a given host is through ethernet
* **eno1** means the interface is an onboard ethernet adapter on the 1st interface.

# Question 3 (ifconfig)

1. Explain the functions of the utility
   * The *ifconfig* (interface configuration) command is used for viewing and modifying configurations for network interfaces. Its used to assign IP addresses and subnet masks to interfaces, or disable certain interfaces.
2. Use the command */sbin/ifconfig –a*. Include the output in your report and explain it.

enp6s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 129.97.110.138 netmask 255.255.255.0 broadcast 129.97.110.255

inet6 fe80::cc21:a09e:a2c2:8d96 prefixlen 64 scopeid 0x20<link>

ether 88:d7:f6:7d:d0:4e txqueuelen 1000 (Ethernet)

RX packets 89786512 bytes 15487192288 (15.4 GB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 97235857 bytes 22472691129 (22.4 GB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536

inet 127.0.0.1 netmask 255.0.0.0

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 1000 (Local Loopback)

RX packets 39451917 bytes 5314369228 (5.3 GB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 39451917 bytes 5314369228 (5.3 GB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

* **enp6s0** is the ethernet network interface. **lo** is a virtual interface that represents the loopback device. It is used by the system to communicate with itself (for example, if some service is hosted on the localhost)
* **UP** flag indicates the kernel modules for the ethernet interface have been loaded
* **BROADCAST** flag indicates the ethernet device broadcasts; used to obtain IP addresses with DHCP
* **RUNNING** flag indicates the interface is ready to take data
* **MULTICAST** flag indicates the ethernet interface does multicasting. Multicasting lets a source send packets to multiple systems if the systems are listening for transmissions
* **MTU** means Maximum Transfer Unit, the size of the packets received by the ethernet NIC. By default, it is set to 1500 for ethernet devices.
* **inet 129.97.110.138 netmask 255.255.255.0 broadcast 129.97.110.255** displays the IPv4 address of the machine, the subnet mask address and the broadcast address respectively.
* **inet6 fe80::cc21:a09e:a2c2:8d96 prefixlen 64 scopeid 0x20<link>** describes the IPv6 address of the machine and the scope of it. The scope is the area on the network where the IPv6 address can be used as a unique identifier among the other network devices.
* **ether 88:d7:f6:7d:d0:4e txqueuelen 1000 (Ethernet)** describes the ethernet adapter’s physical address and the max number of packets allowed in the queue for the interface
* **RX packets** indicates the number of packets received
* **RX errors** indicates how many of those packets had errors in them (invalid CRC, etc.)
  + **dropped** shows how many of the packets were dropped
  + **overruns** show how many of the packets suffered from FIFO overrun, which happens when the rate of the buffer being filled is greater than the rate at which the kernel can empty the buffer
  + **frame** shows how many of the packets had misaligned frames (wrong offsets, not in the correct format, etc)
* **TX packets** indicates the number of packets transmitted
* **TX errors, dropped** and **overruns** are analogous to the RX cases
* **TX collisions** show the number of packets transmitted where collisions occurred
* **TX carriers** describes how many transmitted packets lost their carriers. It can happen due to modulation errors, duplex mode incompatibilities, etc.

# Question 4 (netstat)

1. Explain the functions of the utility.
2. Use the command *netstat -in*. Change *–in* to *–s* to get some statistics. Include the output in your report and explain it.
   * *netstat -in*

Kernel Interface table

Iface MTU RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg

enp6s0 1500 109432710 0 0 0 117196328 0 0 0 BMRU

lo 65536 39453865 0 0 0 39453865 0 0 0 LRU

This is the Kernel Interface Table. It has statistics for all the network interfaces.

We see some information also displayed in the *ifconfig* command, including a new field *Flg* for each interface. The **BMRU** flag means a broadcast address is set (**B**), the interface supports multicast (**M**), the interface is running (**R**) and the interface is up (**U**). The **L** in the **LRU** flag means the interface is a loopback device.

* *netstat -s*

Ip:

Forwarding: 2

148725616 total packets received

1881 with invalid addresses

0 forwarded

0 incoming packets discarded

148565687 incoming packets delivered

148510554 requests sent out

2041 outgoing packets dropped

13 dropped because of missing route

Icmp:

3749 ICMP messages received

27 input ICMP message failed

ICMP input histogram:

destination unreachable: 75

echo requests: 3674

3749 ICMP messages sent

0 ICMP messages failed

ICMP output histogram:

destination unreachable: 75

echo replies: 3674

IcmpMsg:

InType3: 75

InType8: 3674

OutType0: 3674

OutType3: 75

Tcp:

53470 active connection openings

91521 passive connection openings

7538 failed connection attempts

2215 connection resets received

14 connections established

148042276 segments received

156423613 segments sent out

3076 segments retransmitted

0 bad segments received

48557 resets sent

Udp:

247080 packets received

21 packets to unknown port received

0 packet receive errors

220909 packets sent

0 receive buffer errors

0 send buffer errors

IgnoredMulti: 271031

UdpLite:

TcpExt:

3 resets received for embryonic SYN\_RECV sockets

54478 TCP sockets finished time wait in fast timer

71189 delayed acks sent

239 delayed acks further delayed because of locked socket

Quick ack mode was activated 827 times

32 times the listen queue of a socket overflowed

32 SYNs to LISTEN sockets dropped

141305374 packet headers predicted

738867 acknowledgments not containing data payload received

140639795 predicted acknowledgments

TCPSackRecovery: 727

Detected reordering 50598 times using SACK

Detected reordering 215 times using time stamp

29 congestion windows fully recovered without slow start

210 congestion windows partially recovered using Hoe heuristic

TCPDSACKUndo: 38

13 congestion windows recovered without slow start after partial ack

TCPLostRetransmit: 113

2091 fast retransmits

TCPTimeouts: 229

TCPLossProbes: 861

TCPLossProbeRecovery: 29

TCPBacklogCoalesce: 29521

TCPDSACKOldSent: 829

TCPDSACKRecv: 700

TCPDSACKOfoRecv: 1

2465 connections reset due to unexpected data

501 connections reset due to early user close

17 connections aborted due to timeout

TCPDSACKIgnoredOld: 1

TCPDSACKIgnoredNoUndo: 490

TCPSackShifted: 33779

TCPSackMerged: 6907

TCPSackShiftFallback: 97629

TCPRcvCoalesce: 606083

TCPOFOQueue: 53965

TCPSpuriousRtxHostQueues: 36

TCPAutoCorking: 1917166

TCPFromZeroWindowAdv: 44

TCPToZeroWindowAdv: 44

TCPWantZeroWindowAdv: 4901

TCPSynRetrans: 54

TCPOrigDataSent: 151675483

TCPHystartTrainDetect: 115

TCPHystartTrainCwnd: 2265

TCPHystartDelayDetect: 12

TCPHystartDelayCwnd: 495

TCPWinProbe: 1

TCPKeepAlive: 1744

TCPDelivered: 151720268

TCPAckCompressed: 50221

IpExt:

InMcastPkts: 120315

OutMcastPkts: 268

InBcastPkts: 271226

InOctets: 21559342296

OutOctets: 27959359974

InMcastOctets: 8217765

OutMcastOctets: 15369

InBcastOctets: 26727877

InNoECTPkts: 148839145

InECT0Pkts: 20573

This command verbosely displays all the available statistics for network connections on the system. We see it categorizes the statistics into TCP, UDP, Ip, Icmp, etc.

1. Use the command *netstat -r*. Include the output in your report and explain it.

Kernel IP routing table

Destination Gateway Genmask Flags MSS Window irtt Iface

default exsw02-circuitn 0.0.0.0 UG 0 0 0 enp2s0

129.97.56.0 0.0.0.0 255.255.255.0 U 0 0 0 enp2s0

link-local 0.0.0.0 255.255.0.0 U 0 0 0 enp2s0

* This command shows the Kernel IP Routing Table. It’s entries store information about the mapping between interfaces and IP addresses. We can see that enp2s0 is the default interface.

# Question 5 (nslookup)

1. Explain, in your own words, what the utility does.
   * *nslookup* is used to find the IP address corresponding to a certain host name.
2. Use the command to obtain the IP addresses of the following hosts and explain what you get.
   * 1. ecelinux.uwaterloo.ca (do it twice)

Server: 127.0.0.53

Address: 127.0.0.53#53

Non-authoritative answer:

Name: ecelinux.uwaterloo.ca

Address: 129.97.56.15

Server: 127.0.0.53

Address: 127.0.0.53#53

Non-authoritative answer:

Name: ecelinux.uwaterloo.ca

Address: 129.97.56.15

The IP address of the host name here is 129.97.56.15. The server address 127.0.0.53 is the address that the host will query to get the IP address. The system will resolve the host name to the IP address shown by this command. The port number is 53. In this case both instances of running this command returned the same IP address. If there were more than one IP address that resolved to the given host name, the *nslookup* command would return them one by one every time the command is run.

* + 2. [www.mit.edu](http://www.mit.edu)

Server: 127.0.0.53

Address: 127.0.0.53#53

Non-authoritative answer:

www.mit.edu canonical name = www.mit.edu.edgekey.net.

www.mit.edu.edgekey.net canonical name = e9566.dscb.akamaiedge.net.

Name: e9566.dscb.akamaiedge.net

Address: 23.15.221.100

Name: e9566.dscb.akamaiedge.net

Address: 2600:140a:6000:29f::255e

Name: e9566.dscb.akamaiedge.net

Address: 2600:140a:6000:293::255e

In this case the canonical name is shown. Essentially the canonical name is the “true” domain name that the DNS maps the queried host name to. We also see that the true domain name is spread across multiple physical addresses, meaning there are multiple machines hosting this web service.

* + 3. [www.gmail.com](http://www.gmail.com)

Server: 127.0.0.53

Address: 127.0.0.53#53

Non-authoritative answer:

www.gmail.com canonical name = mail.google.com.

mail.google.com canonical name = googlemail.l.google.com.

Name: googlemail.l.google.com

Address: 172.217.164.197

Name: googlemail.l.google.com

Address: 2607:f8b0:400b:800::2005

* + 4. [www.facebook.com](http://www.facebook.com)

Server: 127.0.0.53

Address: 127.0.0.53#53

Non-authoritative answer:

www.facebook.com canonical name = star-mini.c10r.facebook.com.

Name: star-mini.c10r.facebook.com

Address: 31.13.80.36

Name: star-mini.c10r.facebook.com

Address: 2a03:2880:f10e:83:face:b00c:0:25de

# Question 6 (ping)

1. Explain the functions of the utility.
   * This command is used to troubleshoot connectivity, reachability and domain name resolution. The command sends a test request to a specific device (resolved from the host name given to it as a parameter) and returns the result of whether the request was met with a response. This command uses the Internet Control Message Protocol (ICMP) to send an echo request. A response is sent back as an ICMP echo response. The results include parameters indicating datagram sequence number, time to live and round-trip time. Statistics about transmitted, received and lost packets are also computed and shown in addition to max/min/avg times.
   * The *-c10* specifies how many echo requests to do. If the command is run without this option, it will keep pinging the host until a user interrupt occurs.
2. Use *ping –c10 hostname* to estimate the average round-trip-time from the machine on which you are logged in to the following hosts. Include the output in your report and explain what you get.
   1. [www.ualberta.ca](http://www.ualberta.ca)

PING prod.cds.ualberta.cloud (13.226.143.78) 56(84) bytes of data.

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=1 ttl=240 time=4.43 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=2 ttl=240 time=4.72 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=3 ttl=240 time=4.45 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=4 ttl=240 time=4.45 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=5 ttl=240 time=4.38 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=6 ttl=240 time=4.34 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=7 ttl=240 time=8.17 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=8 ttl=240 time=4.37 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=9 ttl=240 time=4.38 ms

64 bytes from server-13-226-143-78.yto50.r.cloudfront.net (13.226.143.78): icmp\_seq=10 ttl=240 time=4.40 ms

--- prod.cds.ualberta.cloud ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9013ms

rtt min/avg/max/mdev = 4.344/4.812/8.171/1.124 ms

* 1. [www.lemonde.fr](http://www.lemonde.fr)

PING s2.shared.global.fastly.net (151.101.126.217) 56(84) bytes of data.

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=1 ttl=52 time=5.11 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=2 ttl=52 time=5.07 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=3 ttl=52 time=5.11 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=4 ttl=52 time=5.13 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=5 ttl=52 time=5.13 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=6 ttl=52 time=5.06 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=7 ttl=52 time=5.15 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=8 ttl=52 time=5.16 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=9 ttl=52 time=5.13 ms

64 bytes from 151.101.126.217 (151.101.126.217): icmp\_seq=10 ttl=52 time=5.09 ms

--- s2.shared.global.fastly.net ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9010ms

rtt min/avg/max/mdev = 5.063/5.116/5.163/0.077 ms

* 1. [www.ucla.edu](http://www.ucla.edu)

PING gateway.lb.it.ucla.edu (164.67.228.152) 56(84) bytes of data.

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=1 ttl=42 time=77.7 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=2 ttl=42 time=77.8 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=3 ttl=42 time=77.7 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=4 ttl=42 time=77.9 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=5 ttl=42 time=77.8 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=6 ttl=42 time=77.9 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=7 ttl=42 time=77.8 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=8 ttl=42 time=77.9 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=9 ttl=42 time=77.7 ms

64 bytes from spotlight.ucla.edu (164.67.228.152): icmp\_seq=10 ttl=42 time=78.0 ms

--- gateway.lb.it.ucla.edu ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9013ms

rtt min/avg/max/mdev = 77.756/77.867/78.004/0.263 ms

# Question 7 (traceroute)

1. Explain the functions of the utility.
   * This command is a diagnostic tool for tracking the path taken by a transmitted packet from source to destination. It shows the IP address of each of the routers/machines it pings during its travel, in addition to the time taken between hops.
2. Use */usr/sbin/traceroute hostname* to find out how many hops there are between the machine on which you are logged in and the following hosts. Include the outputs in your report and explain what you get.
   1. [www.uwaterloo.ca](http://www.uwaterloo.ca)

traceroute to www.uwaterloo.ca (129.97.208.23), 30 hops max, 60 byte packets

1 exsw02-circuitnet.uwaterloo.ca (129.97.56.1) 3.379 ms 3.356 ms 3.340 ms

2 v490-eng-rt-e2.ns.uwaterloo.ca (172.16.32.193) 4.650 ms 4.716 ms 4.704 ms

3 te4-3-dist-rt-phy.ns.uwaterloo.ca (172.18.7.21) 0.721 ms 0.829 ms 0.865 ms

4 xe1-0-0-u11-dist-sa-mc-trust.ns.uwaterloo.ca (172.31.0.149) 0.474 ms 0.477 ms 0.465 ms

5 te2-12-dist-rt-mc-global.ns.uwaterloo.ca (172.31.0.161) 1.336 ms 1.227 ms 1.129 ms

6 te2-16-cn-rt-rac.ns.uwaterloo.ca (172.16.31.113) 0.776 ms 0.845 ms 0.782 ms

7 e1-1-cr-rt-bb2.ns.uwaterloo.ca (172.16.16.39) 1.107 ms 1.299 ms 1.222 ms

8 xe1-0-1-22-cr-sa-bb2.ns.uwaterloo.ca (172.16.16.5) 1.072 ms 1.051 ms 1.119 ms

9 e1-25-20-cr-rt-bb2-area2.ns.uwaterloo.ca (172.16.16.13) 1.642 ms 2.038 ms 1.935 ms

10 wms.uwaterloo.ca (129.97.208.23) 1.270 ms 1.228 ms 1.434 ms

11 wms.uwaterloo.ca (129.97.208.23) 1.404 ms !N 1.286 ms !N 1.350 ms !N

* 1. [www.youtube.com](http://www.youtube.com)

traceroute to www.youtube.com (172.217.164.206), 30 hops max, 60 byte packets

1 exsw02-circuitnet.uwaterloo.ca (129.97.56.1) 3.428 ms 3.415 ms 3.445 ms

2 v490-eng-rt-e2.ns.uwaterloo.ca (172.16.32.193) 4.323 ms 4.316 ms 4.380 ms

3 te4-3-dist-rt-mc.ns.uwaterloo.ca (172.18.7.17) 0.880 ms 0.930 ms 1.252 ms

4 xe1-0-0-u10-dist-sa-mc-trust.ns.uwaterloo.ca (172.31.0.145) 0.501 ms 0.487 ms 0.441 ms

5 te2-12-dist-rt-mc-global.ns.uwaterloo.ca (172.31.0.161) 1.459 ms 1.717 ms 1.499 ms

6 \* \* \*

7 te0-0-2-0-ext-rt-mc.ns.uwaterloo.ca (172.16.32.149) 1.841 ms 1.642 ms 1.664 ms

8 unallocated-static.rogers.com (72.142.108.181) 1.134 ms 1.175 ms 1.139 ms

9 24.156.146.217 (24.156.146.217) 2.790 ms 2.126 ms 2.124 ms

10 24.156.146.197 (24.156.146.197) 3.839 ms 3.821 ms 3.692 ms

11 9044-cgw01.mtnk.asr9k.rmgt.net.rogers.com (209.148.230.53) 3.899 ms 4.031 ms 4.080 ms

12 209.148.235.42 (209.148.235.42) 4.852 ms 209.148.235.145 (209.148.235.145) 5.109 ms 4.740 ms

13 \* 72.14.222.87 (72.14.222.87) 6.326 ms 72.14.209.126 (72.14.209.126) 5.545 ms

14 \* 74.125.244.145 (74.125.244.145) 6.183 ms 74.125.244.161 (74.125.244.161) 4.485 ms

15 216.239.35.232 (216.239.35.232) 5.519 ms 216.239.35.234 (216.239.35.234) 5.514 ms 216.239.42.159 (216.239.42.159) 5.470 ms

16 216.239.42.159 (216.239.42.159) 5.263 ms yyz12s04-in-f14.1e100.net (172.217.164.206) 5.763 ms 108.170.250.227 (108.170.250.227) 5.753 ms

Certain hops have asterisks shown. That means a response is not heard from these routers within the timeout. Since the *traceroute* command sends UDP packets, the packets can be blocked by certain firewalls set up in routers, leading them to drop the packets.

* 1. [www.nytimes.com](http://www.nytimes.com)

traceroute to www.nytimes.com (151.101.125.164), 30 hops max, 60 byte packets

1 exsw02-circuitnet.uwaterloo.ca (129.97.56.1) 3.459 ms 3.580 ms 3.566 ms

2 v490-eng-rt-e2.ns.uwaterloo.ca (172.16.32.193) 4.438 ms 4.436 ms 4.570 ms

3 te4-3-dist-rt-mc.ns.uwaterloo.ca (172.18.7.17) 0.700 ms 0.829 ms 0.929 ms

4 xe1-0-0-u10-dist-sa-mc-trust.ns.uwaterloo.ca (172.31.0.145) 0.449 ms 0.453 ms 0.438 ms

5 te2-12-dist-rt-mc-global.ns.uwaterloo.ca (172.31.0.161) 1.123 ms 1.367 ms 1.206 ms

6 te2-16-cn-rt-rac.ns.uwaterloo.ca (172.16.31.113) 0.872 ms 0.806 ms 0.847 ms

7 te-0-0-2-1-ext-rt-mc.ns.uwaterloo.ca (172.16.31.229) 1.582 ms 3.761 ms 3.639 ms

8 unallocated-static.rogers.com (72.142.108.181) 1.118 ms 1.146 ms 1.146 ms

9 24.156.146.217 (24.156.146.217) 2.020 ms 1.982 ms 1.944 ms

10 24.156.146.197 (24.156.146.197) 5.344 ms 4.691 ms 4.621 ms

11 9044-cgw01.mtnk.asr9k.rmgt.net.rogers.com (209.148.230.53) 4.983 ms 4.960 ms 4.461 ms

12 209.148.235.222 (209.148.235.222) 5.266 ms 5.570 ms 5.182 ms

13 \* \* \*

14 \* \* \*

15 \* \* \*

16 \* \* \*

17 \* \* \*

18 \* \* \*

19 \* \* \*

20 \* \* \*

21 \* \* \*

22 \* \* \*

23 \* \* \*

24 \* \* \*

25 \* \* \*

26 \* \* \*

27 \* \* \*

28 \* \* \*

29 \* \* \*

30 \* \* \*

We see clearly here that after hop 12, each subsequent router fails to respond to the ping in time. This is likely because the routers are set up to block UDP.