COMPUTER SYSTEMS LAB CS 558
Assignment 01 (Network Diagnostic Commands)

Vijay Purohit | 214101058

M.Tech CSE, IITG

Contents

1 *ping* Command

The Internet Ping command bounces a small packet(s) to test network communications, and then shows how long this packet(s) took to make the round trip. The Internet Ping program works much like a sonar echo- location, sending a small packet of information containing an ICMP ECHO_REQUEST to a specified computer, which then sends an ECHO_REPLY packet in return. Explore more about the ping command and answer the following questions (Unix or GNU/Linux version only):

(a) What is the option required to specify the number of echo requests to send with ping command?
The option required to specify the number of echo request is `-c` which stands for `count`.
`ping -c 2 www.google.com`

(b) What is the option required to set time interval (in seconds), rather than the default one second interval, between two successive ping ECHO_REQUESTs?
To increase or decrease time interval we have to use the `-i` switch which stands for `interval`.
`ping -i 2 www.google.com`

(c) What is the command to send ECHO_REQUEST packets to the destination one after another without waiting for a reply? What is the limit for sending such ECHO_REQUEST packets by normal users (not super user)?
The command to send packets to the destination one after another without waiting for a reply is `-l` switch which stand for `preload`. `ping -l 2 www.google.com`
The limit of sending such packets by normal users is **3**.

(d) What is the command to set the ECHO_REQUEST packet size (in bytes)? If the PacketSize is set to 32 bytes, what will be the total packet size?
The command to set the packet size is `-s` switch. `ping -s 32 www.google.com`
Packet size 32B, ICMP Header size 8B then Total Packet Size = 40 Bytes.

2 *ping* Experiment With Six Host

Select six hosts of your choice in the Internet (mention the list in your report) and experiment with pinging each host 25 times at three different hours of the day. Check if there exist cases, which show packet loss greater than 0% and provide reasoning. Find out average RTT for each host and explain whether measured RTTs are strongly or weakly correlated with the geographical distance of the hosts. Pick one of the above used hosts and repeat the experiment with different packet sizes ranging from 64 bytes to 2048 bytes. Plot the average RTT, and explain how change in packet size and time of the day impact RTT. You can use the following online tools for this experiment:
i) `http://www.spfld.com/ping.html`
ii) `https://www.subnetonline.com/pages/network-tools/online-ping-ipv4.php`

Table 1: Hosts Taken

1	CodeForces (213.248.110.126)	Saint Petersburg, Sankt-Peterburg, Russian Federation
2	Instagram (2a03:2880:f212:e5:face:b00c:0:4420)	Dublin, Dublin, Ireland
3	Spotify (35.186.224.25)	Kansas City, Missouri, USA
4	YouTube (142.250.64.110)	Mountain View, California, USA
5	BookMyShow (104.16.123.37)	San Francisco, California, USA
6	Twitter (104.244.42.1)	San Francisco, California, USA

Table 2: Hosts AVG RTT (msec) at three different hours of the day (22 March 2022) with Default packet Size.

Hosts	09:00	13:00	20:00	24:00
BookMyShow	174.859	314.785	604.841	237.305
YouTube	574.014	640.428	622.783	725.882
Twitter	219.607	309.642	306.901	384.554
Spotify	144.692	553.554	329.335	393.053
Instagram	610.696	646.329	949.457	655.963
CodeForces	397.447	742.212	492.919	595.635

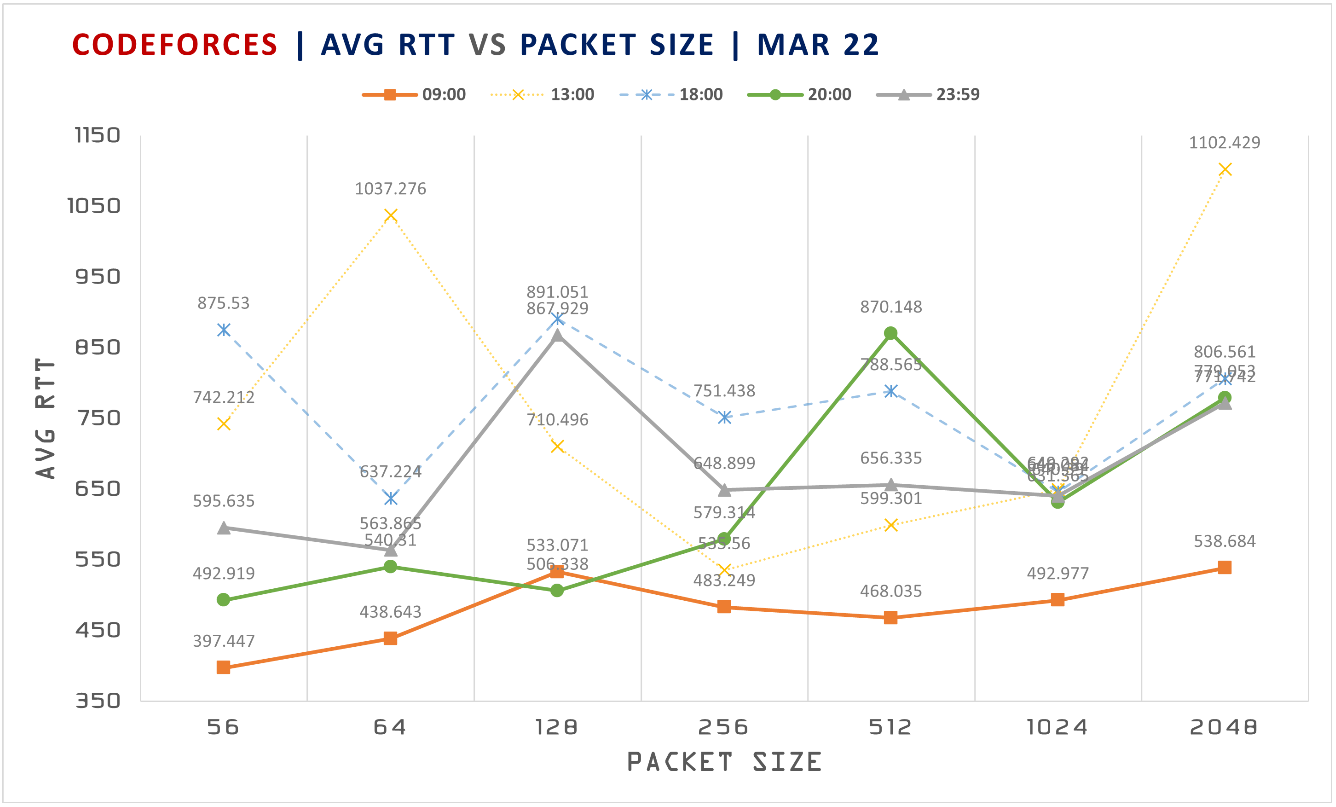


Figure 1: Avg RTT vs Packet Size

For the default packet size with the above 6 hosts, there was no packet loss greater than 0%. But the experiment conducted with larger packet size 2048B and in some other cases were showing packet loss. There are many causes of packet loss, most of them unintentional: Network Congestion, Problems With Network Hardware, Software Bugs, Overloaded Devices, Security Threats, reduce throughput.
There is a positive correlation between distance and RTT. There can be number of reasons for this: an increased hop count. The packets have to go through more routers, at each router there may be a delay. More routers, the longer the RTT. But it is less as compared to linearized distance. So RTTs are weakly correlated to geographic distance.

Table 3: Single Host AVG RTT with respect to change in packet size

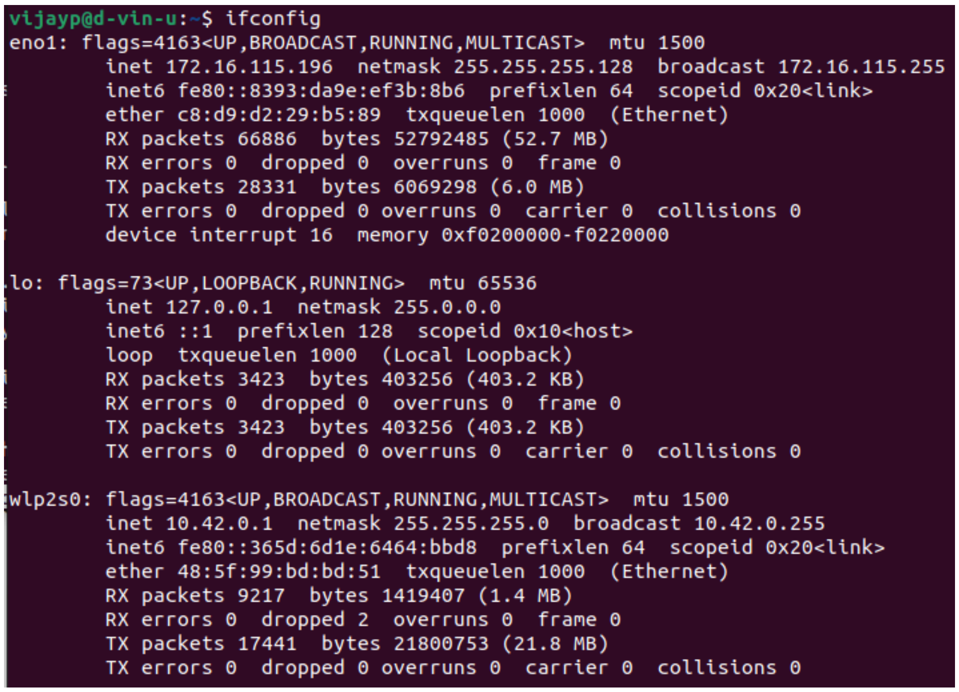
Time\Size	56B	64B	128B	256B	512B	1024B	2048B
09:00	397.447	438.643	533.071	483.249	468.035	492.977	538.684
13:00	742.212	1037.276	710.496	535.56	599.301	649.292	1102.429
18:00	875.53	637.224	891.051	751.438	788.565	646.084	806.561
20:00	492.919	540.31	506.338	579.314	870.148	631.365	779.053
23:59	595.635	563.865	867.929	648.899	656.335	640.39	771.742

Varying packet size from 64 bytes to 2048 bytes shows that for some of the hosts for higher packet size there was 100% packet loss. While for some it is showing valid RTT value. Therefore, we cannot make any strong conclusion here.

3 *ifconfig* and *route* Commands

With regard to ifconfig and route commands, an swer the following questions:

(a) Run *ifconfig* command and describe its output (identify and explain as much of what is printed on the screen as you can).
`ifconfig` (interface configuration) is used to view and configure the network interfaces. It is used at boot time to set up interfaces as necessary. If no arguments are given, `ifconfig` displays the status of the currently active interfaces. If a single interface argument is given, it displays the status of the given interface only; if a single -a argument is given, it displays the status of all interfaces, even those that are down.



```
vijayp@d-vin-u:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.115.196  netmask 255.255.255.128  broadcast 172.16.115.255
        inet6 fe80::8393:da9e:ef3b:8b6  prefixlen 64  scopeid 0x20<link>
        ether c8:d9:d2:29:b5:89  txqueuelen 1000  (Ethernet)
        RX packets 66886  bytes 52792485 (52.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28331  bytes 6069298 (6.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xf0200000-f0220000

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 3423  bytes 403256 (403.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3423  bytes 403256 (403.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.42.0.1  netmask 255.255.255.0  broadcast 10.42.0.255
        inet6 fe80::365d:6d1e:6464:bbd8  prefixlen 64  scopeid 0x20<link>
        ether 48:5f:99:bd:bd:51  txqueuelen 1000  (Ethernet)
        RX packets 9217  bytes 1419407 (1.4 MB)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 17441  bytes 21800753 (21.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 2: ifconfig output

Here, **eno1, lo and wlp2s0** are the names of the active network interfaces.

- `eno1` is the name of embedded NIC (Network Interface Card). It is a regular physical network interface. The number is a firmware/BIOS provided index.
- `lo` is the loopback interface. This is a special network interface that the system uses to communicate with itself.
- `wlp2s0` is the wireless lan controller (pci bus 2 slot 0)

Interface Details:

- `UP` indicates that kernel modules related to the interface have been loaded and interface is activated.
- `BROADCAST` indicates that interface is configured to handle broadcast packets, which is required for obtaining IP address via DHCP.
- `RUNNING` indicates that interface is ready to accept data.
- `MULTICAST` indicates that interface supports multicasting.
- `mtu` is maximum transmission unit. IP datagrams larger than MTU bytes will be fragmented into multiple Ethernet frames.
- `inet addr` is IPv4 address assigned to the interface.
- `broadcast` is broadcast address for the interface.
- `netmask` is network mask for the interface.
- `inet6 addr` is IPv6 address assigned to the interface.
- `ether` is hardware address of the ethernet interface (also known as MAC address).
- `scopeid` is scope of IPv6 address. It can be link-local or global.
- `Metric` determines the cost of using the interface. Interfaces with lower cost have higher priority.

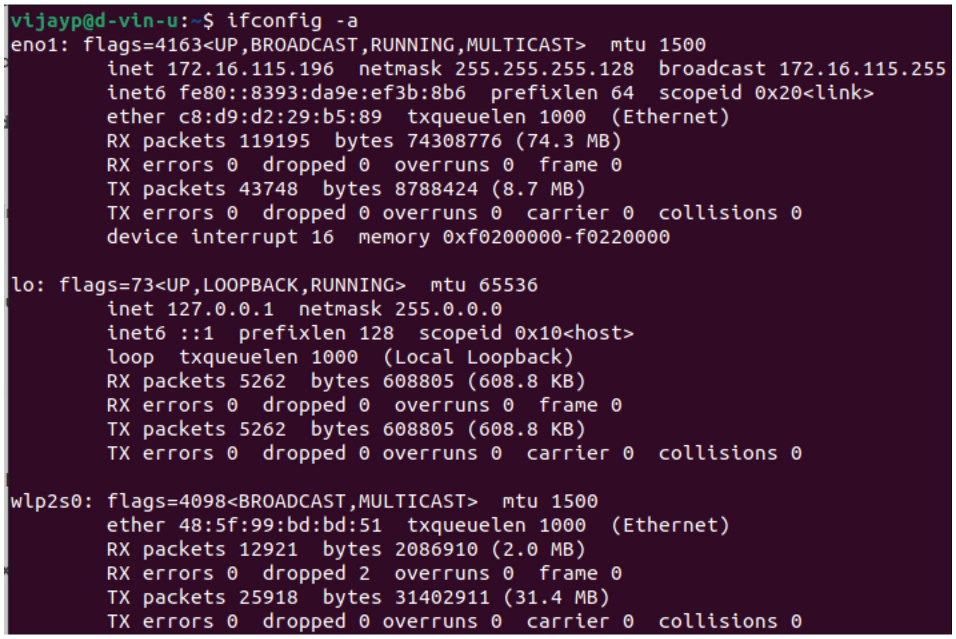
Interface stats:

- `txqueuelen` is length of transmission queue.
- `RX packets, TX packets` is a total number of packets received and transmitted respectively.
- `RX bytes/TX bytes` is a total number of bytes received/transmitted over interface.
- `RX errors/TX errors` shows a total number of packets received/transmitted with error. This includes too-long-frames errors, ring-buffer overflow errors, CRC errors, frame alignment errors, fifo overruns, and missed packets.
- `RX dropped` is a number of dropped packets due to unintended VLAN tags or receiving IPv6 frames when interface is not configured for IPv6.
- `RX overruns` is a number of received packets that experienced fifo overruns, caused by rate at which a buffer gets full and kernel isn't able to empty it.
- `RX frame` is a number of misaligned frames, i.e. frames with length not divisible by 8.
- `TX carriers` is a number of packets that experienced loss of carriers. This usually happens when link is flapping.
- `TX collisions` is a number of transmitted packets that experienced Ethernet collisions.
- `interrupt` it is the interrupt number used by NIC.

(b) What options can be provided with the *ifconfig* command? Mention and explain at least four options.
Options that can be provided with `ifconfig` are

-a, -s, -v, interface_name, up, down, [-]arp, [-]promisc, [-]allmulti, mtu N, dstaddr addr, netmask addr, add addr, del addr, media type, address .

- `-a`: display all interfaces which are currently available, even if down. `ifconfig -a`.



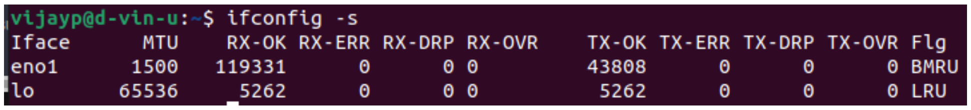
```
vijayp@d-vin-u:~$ ifconfig -a
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.115.196  netmask 255.255.255.128  broadcast 172.16.115.255
        inet6 fe80::8393:da9e:ef3b:8b6  prefixlen 64  scopeid 0x20<link>
        ether c8:d9:d2:29:b5:89  txqueuelen 1000  (Ethernet)
        RX packets 119195  bytes 74308776 (74.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 43748  bytes 8788424 (8.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xf0200000-f0220000

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 5262  bytes 608805 (608.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5262  bytes 608805 (608.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 48:5f:99:bd:bd:51  txqueuelen 1000  (Ethernet)
        RX packets 12921  bytes 2086910 (2.0 MB)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 25918  bytes 31402911 (31.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3: ifconfig -a output

- `-s`: display a short list. `ifconfig -s`.

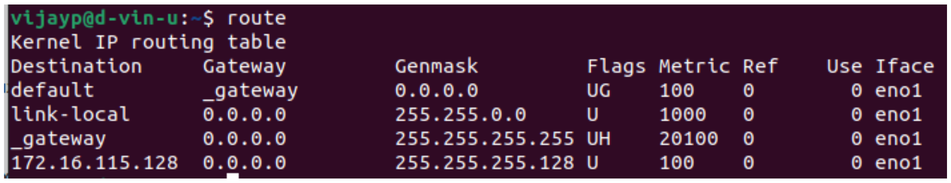


```
vijayp@d-vin-u:~$ ifconfig -s
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eno1      1500   119331      0      0 0         43808      0      0      0 BMRU
lo       65536     5262      0      0 0          5262      0      0      0 LRU
```

Figure 4: ifconfig -s output

- `-v`: be more verbose for some error conditions. `ifconfig -v`.
- `down`: This flag causes the driver for this interface to be shut down. `ifconfig interface down`.
- `up`: This flag causes the interface to be activated. `ifconfig interface up`.
- `add addr/prefixlen`: Add an IPv6 address to an interface. `ifconfig interface add addr/prefixlen`.
- `del addr/prefixlen`: Remove an IPv6 address from an interface. `ifconfig interface del addr/prefixlen`.

(c) Explain the output of *route* command.
The output of the kernel routing table is organized in the following columns



```
vijayp@d-vin-u:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 eno1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 eno1
_gateway        0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
```

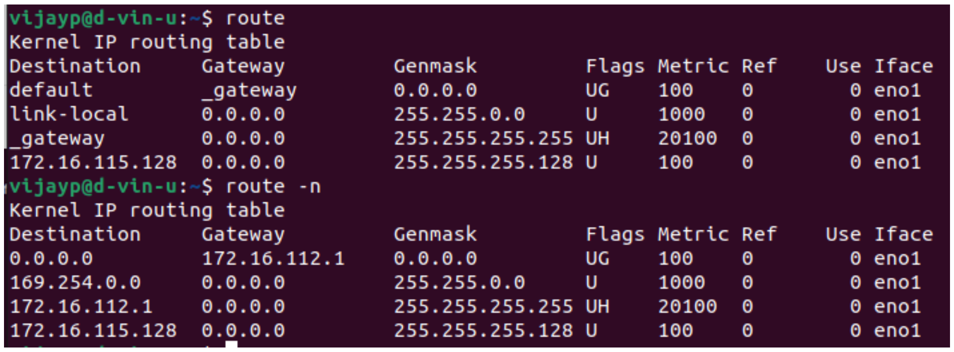
Figure 5: route output

- `Destination`: The destination network or destination host.
- `Gateway`: The gateway address or '*' if none set.
- `Genmask`: The netmask for the destination net; '255.255.255.255' for a host destination and '0.0.0.0' for the default route.
- `Flags`: Possible flags include
  - U (route is up)
  - H (target is a host)
  - G (use gateway)
  - R (reinstate route for dynamic routing)
  - D (dynamically installed by daemon or redirect)
  - M (modified from routing daemon or redirect)
  - A (installed by addrconf)
  - C (cache entry)
  - ! (reject route)
- `Metric`: The 'distance' to the target (usually counted in hops).
- `Ref`: Number of references to this route. (Not used in the Linux kernel.)
- `Use`: Count of lookups for the route. Depending on the use of -F and -C this will be either route cache misses (-F) or hits (-C).
- `Iface`: Interface to which packets for this route will be sent.
- `MSS`: Default maximum segment size for TCP connections over this route.
- `Window`: Window Default window size for TCP connections over this route.
- `irtt`: Initial RTT (Round Trip Time). The kernel uses this to guess about the best TCP protocol parameters without waiting on (possibly slow) answers.

(d) Mention and explain at least four options of the *route* command. Execute the *route* command with these four options and show the output

Options of `route` command are as follows:

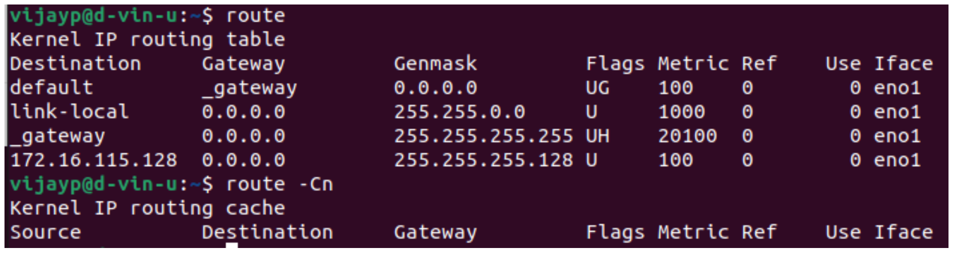
- `-n`: show numerical addresses instead of trying to determine symbolic host names. This is useful if you are trying to determine why the route to your name-server has vanished.



```
vijayp@d-vin-u:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 eno1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 eno1
_gateway        0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
vijayp@d-vin-u:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.112.1    0.0.0.0         UG    100    0        0 eno1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eno1
172.16.112.1    0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
```

Figure 6: route -n output

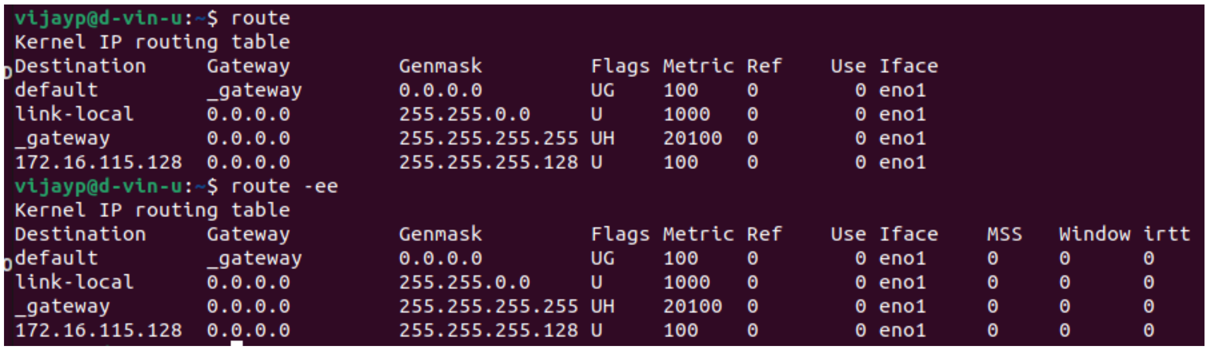
- `-C`: operate on the kernel's routing cache.



```
vijayp@d-vin-u:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 eno1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 eno1
_gateway        0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
vijayp@d-vin-u:~$ route -Cn
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
```

Figure 7: route -C output

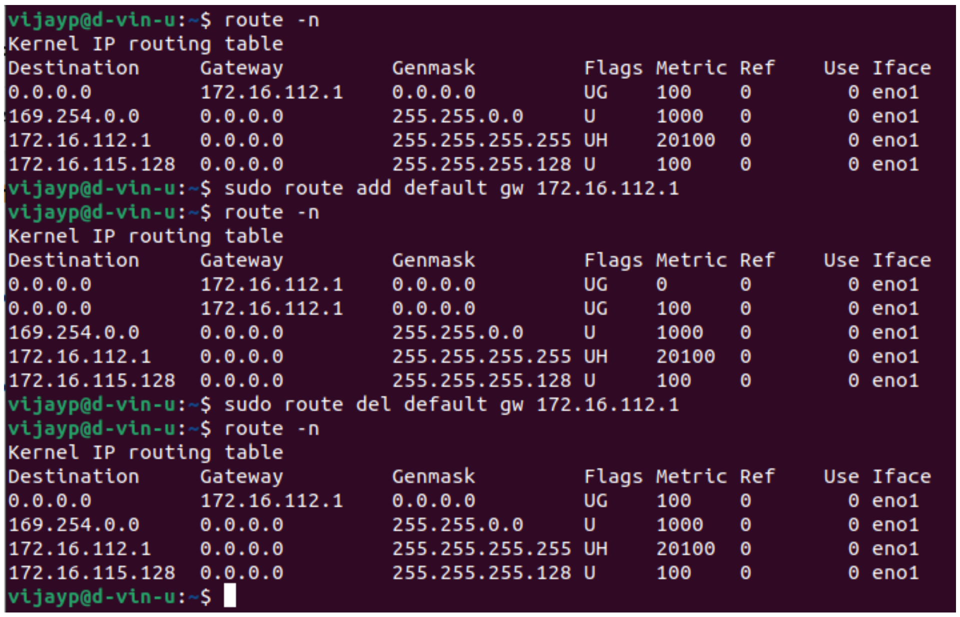
- `-ee`: will generate a very long line with all parameters from the routing table.



```
vijayp@d-vin-u:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 eno1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 eno1
_gateway        0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
vijayp@d-vin-u:~$ route -ee
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface    MSS   Window irtt
default         _gateway        0.0.0.0         UG    100    0        0 eno1     0     0      0
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 eno1     0     0      0
_gateway        0.0.0.0         255.255.255.255 UH    20100  0        0 eno1     0     0      0
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1     0     0      0
```

Figure 8: route -ee output

- `add`: add a new route.
- `del`: delete a route.



```
vijayp@d-vin-u:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.112.1    0.0.0.0         UG    100    0        0 eno1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eno1
172.16.112.1    0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
vijayp@d-vin-u:~$ sudo route add default gw 172.16.112.1
vijayp@d-vin-u:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.112.1    0.0.0.0         UG    0      0        0 eno1
0.0.0.0         172.16.112.1    0.0.0.0         UG    100    0        0 eno1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eno1
172.16.112.1    0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
vijayp@d-vin-u:~$ sudo route del default gw 172.16.112.1
vijayp@d-vin-u:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.112.1    0.0.0.0         UG    100    0        0 eno1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eno1
172.16.112.1    0.0.0.0         255.255.255.255 UH    20100  0        0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U     100    0        0 eno1
vijayp@d-vin-u:~$ 
```

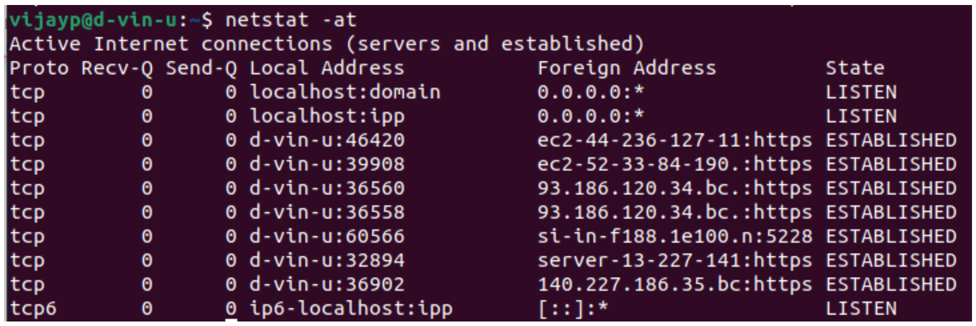
Figure 9: route add and delete output

4 *netstat* Command

Answer the following questions related to netstat command.

(a) What is the command *netstat* used for?
`netstat` prints information about the Linux networking subsystem. The type of information printed is controlled by the first argument. By default, netstat displays a list of open sockets (tcp, udp). If we don't specify any address families, then the active sockets of all configured address families will be printed. It is also used to display the kernel routing tables (`-r`), display multi-cast group membership information for IPv4 and IPv6 (`-g`), display a table of all network interfaces (`-i`), display a list of masqueraded connections (`-M`), display summary statistics for each protocol (`-s`).

(b) What parameters for *netstat* should you use to show all the established TCP connections? Include a screenshot of this list for your computer and explain all the fields of the table in the output.
To show all the established TCP connections, we have to use option `-at` where it will list all listening and non listening tcp ports.



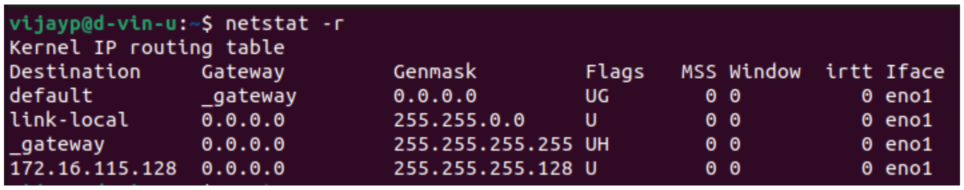
```
vijayp@d-vin-u:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0      0 d-vin-u:46420           ec2-44-236-127-11:https ESTABLISHED
tcp        0      0 d-vin-u:39908           ec2-52-33-84-190.:https ESTABLISHED
tcp        0      0 d-vin-u:36560           93.186.120.34.bc.:https ESTABLISHED
tcp        0      0 d-vin-u:36558           93.186.120.34.bc.:https ESTABLISHED
tcp        0      0 d-vin-u:60566           si-in-f188.1e100.n:5228 ESTABLISHED
tcp        0      0 d-vin-u:32894           server-13-227-141:https ESTABLISHED
tcp        0      0 d-vin-u:36902           140.227.186.35.bc:https ESTABLISHED
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
```

Figure 10: netstat -at command output

- `Proto`: The protocol (tcp, udp, udpl, raw) used by the socket.
- `Recv-Q`: Established: The count of bytes not copied by the user program connected to this socket. Listening: Since Kernel 2.6.18 this column contains the current syn backlog.
- `Send-Q` Established: The count of bytes not acknowledged by the remote host. Listening: Since Kernel 2.6.18 this column contains the maximum size of the syn backlog.
- `Local Address` Address and port number of the local end of the socket. Unless the –numeric (-n) option is specified, the socket address is resolved to its canonical host name (FQDN), and the port number is translated into the corresponding service name.
- `Foreign Address` Address and port number of the remote end of the socket. Analogous to “Local Address”.
- `State` The state of the socket. Since there are no states in raw mode and usually no states used in UDP and UDPLite, this column may be left blank. Normally this can be one of several values:
  - ESTABLISHED: The socket has an established connection.
  - LISTEN: The socket is listening for incoming connections.
  - SYN_SENT The socket is actively attempting to establish a connection.
  - SYN_RECV A connection request has been received from the network.
  - FIN_WAIT1 The socket is closed, and the connection is shutting down.
  - FIN_WAIT2 Connection is closed, and the socket is waiting for a shutdown from the remote end.
  - TIME_WAIT The socket is waiting after close to handle packets still in the network.
  - CLOSE The socket is not being used.
  - CLOSE_WAIT The remote end has shut down, waiting for the socket to close.
  - LAST_ACK The remote end has shut down, and the socket is closed. Waiting for acknowledgement.
  - LISTEN The socket is listening for incoming connections. Such sockets are not included in the output unless you specify the –listening (-l) or –all (-a) option.
  - CLOSING Both sockets are shut down but we still don’t have all our data sent.
  - UNKNOWN The state of the socket is unknown

- `PID/Program name` Slash-separated pair of the process id (PID) and process name of the process that owns the socket.

(c) What does "*netstat -r*" show? Explain all the fields of the output.
"*netstat -r*" display the kernel routing tables. `netstat -r and route -e` produce the same output.

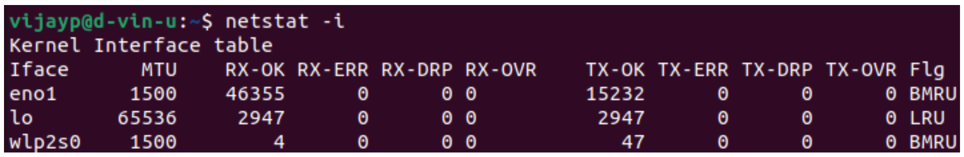


```
vijayp@d-vin-u:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 eno1
link-local      0.0.0.0         255.255.0.0     U         0 0          0 eno1
_gateway        0.0.0.0         255.255.255.255 UH        0 0          0 eno1
172.16.115.128  0.0.0.0         255.255.255.128 U         0 0          0 eno1
```

Figure 11: netstat -r command output

- `Destination`: The destination network or destination host.
- `Gateway`: The gateway address or '*' if none set.
- `Genmask`: The netmask for the destination net; '255.255.255.255' for a host destination and '0.0.0.0' for the default route.
- `Flags`: Possible flags include
  - U (route is up)
  - H (target is a host)
  - G (use gateway)
  - R (reinstate route for dynamic routing)
  - D (dynamically installed by daemon or redirect)
  - M (modified from routing daemon or redirect)
  - A (installed by addrconf)
  - C (cache entry)
  - ! (reject route)
- `MSS`: Default maximum segment size for TCP connections over this route.
- `Window`: Window Default window size for TCP connections over this route.
- `irtt`: Initial RTT (Round Trip Time). The kernel uses this to guess about the best TCP protocol parameters without waiting on (possibly slow) answers.
- `Iface`: Interface to which packets for this route will be sent.

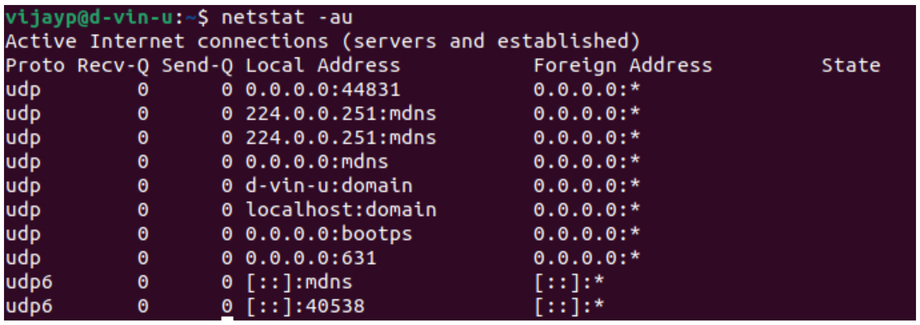
(d) What option of *netstat* can be used to display the status of all network interfaces? By using *netstat*, figure out the number of interfaces on your computer.
`netstat` with option `-i` can be used to display the status of all network interfaces. `netstat -ie` can be used to display extended information on the interfaces. Currently, there are 3 network interfaces running in my computer.



```
vijayp@d-vin-u:~$ netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eno1      1500    46355      0      0 0         15232      0      0      0 BMRU
lo       65536     2947      0      0 0          2947      0      0      0 LRU
wlp2s0    1500        4      0      0 0            47      0      0      0 BMRU
```

Figure 12: netstat -i command output

(e) What option of *netstat* can be used to show the statistics of all UDP connections? Run the command for this purpose on your computer and show the output.
`netstat` with option `-au` can be used to show the statistics of all UDP connections.



```
vijayp@d-vin-u:~$ netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:44831           0.0.0.0:*
udp        0      0 224.0.0.251:mdns        0.0.0.0:*
udp        0      0 224.0.0.251:mdns        0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 d-vin-u:domain          0.0.0.0:*
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootps          0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp6       0      0 [::]:mdns               [::]:*
udp6       0      0 [::]:40538              [::]:*
```

Figure 13: netstat -au command output

(f) Show and explain the function of loopback interface.
`lo` is the loopback interface. This is a special network interface that the system uses to communicate with itself and to represent the loopback facility. Any traffic that a computer program sends on the loopback network is addressed to the same computer. The most commonly used IP address on the loopback network is 127.0.0.1/8 for IPv4 and ::1 for IPv6. The standard domain name for the address is localhost. It is used mainly for diagnostics and troubleshooting, and to connect to servers running on the local machine.

5 *traceroute* Tool

What is a `traceroute` tool used for? Perform a `traceroute` experiment (with same hosts used in Q2) at three different hours of the day, and then answer the questions below. Use any one of the following online tools for this experiment:

- `http://ping.eu`
- `http://www.cogentco.com/en/network/looking-glass`
- `https://www.ultratools.com/tools/traceRoute`
- `http://network-tools.com`

traceroute tracks the route packets taken from an IP network on their way to a given host. It utilizes the IP protocol's time to live (TTL) field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to the host.

(a) List out the hop counts for each host in each time slot. Determine the common hops between two routes if they exist.

Table 4: Hosts vs Hope Count in each time slot. Date: 22-March-2022

Hosts	HC_1 (13:00)	HC_2 (18:00)	HC_3 (23:00)
BookMyShow	10	10	10
YouTube	19	19	19
Twitter	15	15	15
Spotify	11	11	11
Instagram	14	13	11
CodeForces	19	16	16

Most common hops between two routes are: _gateway (192.168.244.90), 192.168.25.231 (192.168.25.231), dsl-tn-dynamic-221.222.22.125.airtelbroadband.in.

(b) Check and explain the reason if route to same host changes at different times of the day.
The reason for different route every time can be: As `traceroute` to a router address uses process switching (checking the table and making a decision) for every packet, that is influenced by load balancing (distribute traffic among all its connections), and we have every time a different path. There can be case also where destination router has two or more connections to different ISPs and the connections are doing load balancing. Another possible reason can be a roter might just be misconfigured.

(c) Inspect the cases when *traceroute* does not find complete paths to some hosts and provide reasoning.
Any form of traceroute works by incrementing the TTL of an IP packet by one and host then send out an error message via ICMP (time to live exceeded).
There are different cases why we don't see a complete paths:

- Firewall might have blocked the ICMP protocol.
- Firewall might have blocked the UDP Ports for tracing hosts.
- Router might be busy routing packets and not have the resources to send out the ICMP packets.
- The interface receiving the ICMP TTL exceeded messages does not have routing back to that address then its not displayed in the traceroute results.

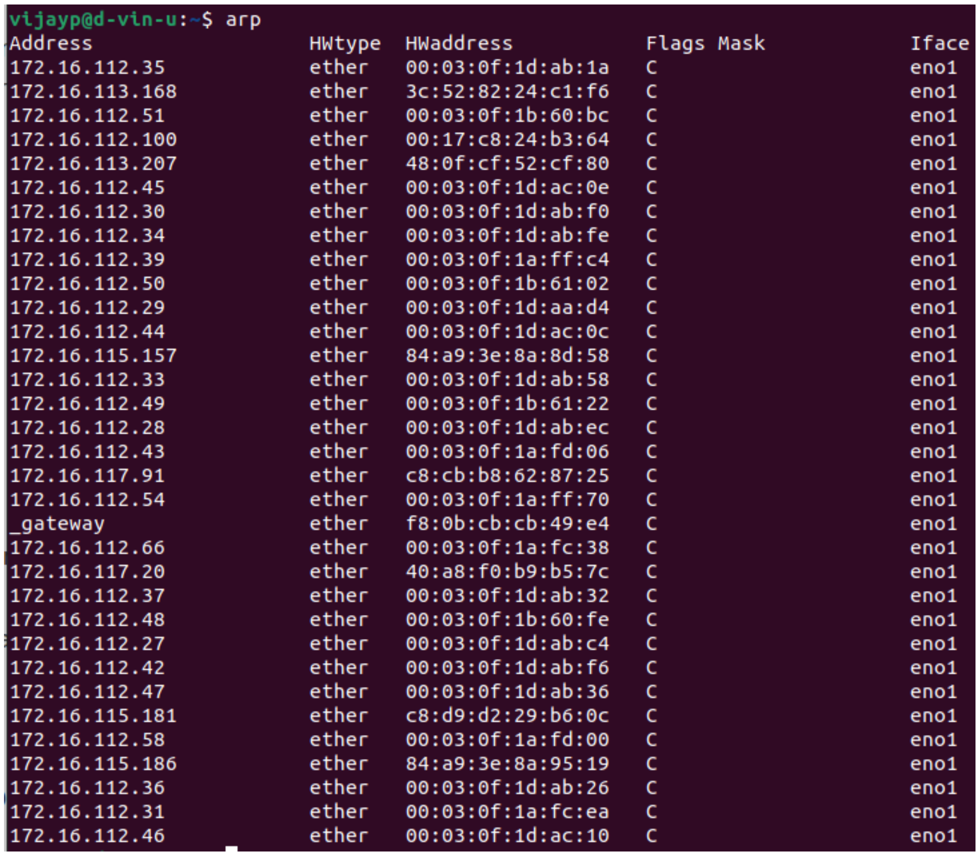
(d) Is it possible to find the route to certain hosts which fail to respond with ping experiment? Give reasoning.
Yes, it is possible to find the route to certain hosts by using the option `-T`. If some filters are present in the network path, then most probably any “unlikely” udp ports (as for default method) or even icmp echoes (as for icmp) are filtered, and whole tracerouting will just stop at such a firewall. To bypass a network filter, we more likely -T -p 25 can reach it, even when -I can not.

6 Network Addresses

Answer the following questions with regard to network addresses.

(a) How do you show the full ARP table for your machine? Explain each column of the ARP table.
Using the following command we can show the full ARP table for the machine.

- `arp`: arp with no mode specifier will print the current content of the table. It is possible to limit the number of entries printed, by specifying an hardware address type, interface name or host address.
- `arp -a`: Use alternate BSD style output format (with no fixed columns).
- `arp -e`: Use default Linux style output format (with fixed columns).



```
vijayp@d-vin-u:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.112.35            ether   00:03:0f:1d:ab:1a   C                     eno1
172.16.113.168           ether   3c:52:82:24:c1:f6   C                     eno1
172.16.112.51            ether   00:03:0f:1b:60:bc   C                     eno1
172.16.112.100           ether   00:17:c8:24:b3:64   C                     eno1
172.16.113.207           ether   48:0f:cf:52:cf:80   C                     eno1
172.16.112.45            ether   00:03:0f:1d:ac:0e   C                     eno1
172.16.112.30            ether   00:03:0f:1d:ab:f0   C                     eno1
172.16.112.34            ether   00:03:0f:1d:ab:fe   C                     eno1
172.16.112.39            ether   00:03:0f:1a:ff:c4   C                     eno1
172.16.112.50            ether   00:03:0f:1b:61:02   C                     eno1
172.16.112.29            ether   00:03:0f:1d:aa:d4   C                     eno1
172.16.112.44            ether   00:03:0f:1d:ac:0c   C                     eno1
172.16.115.157           ether   84:a9:3e:8a:8d:58   C                     eno1
172.16.112.33            ether   00:03:0f:1d:ab:58   C                     eno1
172.16.112.49            ether   00:03:0f:1b:61:22   C                     eno1
172.16.112.28            ether   00:03:0f:1d:ab:ec   C                     eno1
172.16.112.43            ether   00:03:0f:1a:fd:06   C                     eno1
172.16.117.91            ether   c8:cb:b8:62:87:25   C                     eno1
172.16.112.54            ether   00:03:0f:1a:ff:70   C                     eno1
_gateway                 ether   f8:0b:cb:cb:49:e4   C                     eno1
172.16.112.66            ether   00:03:0f:1a:fc:38   C                     eno1
172.16.117.20            ether   40:a8:f0:b9:b5:7c   C                     eno1
172.16.112.37            ether   00:03:0f:1d:ab:32   C                     eno1
172.16.112.48            ether   00:03:0f:1b:60:fe   C                     eno1
172.16.112.27            ether   00:03:0f:1d:ab:c4   C                     eno1
172.16.112.42            ether   00:03:0f:1d:ab:f6   C                     eno1
172.16.112.47            ether   00:03:0f:1d:ab:36   C                     eno1
172.16.115.181           ether   c8:d9:d2:29:b6:0c   C                     eno1
172.16.112.58            ether   00:03:0f:1a:fd:00   C                     eno1
172.16.115.186           ether   84:a9:3e:8a:95:19   C                     eno1
172.16.112.36            ether   00:03:0f:1d:ab:26   C                     eno1
172.16.112.31            ether   00:03:0f:1a:fc:ea   C                     eno1
172.16.112.46            ether   00:03:0f:1d:ac:10   C                     eno1
```

Figure 14: arp command output

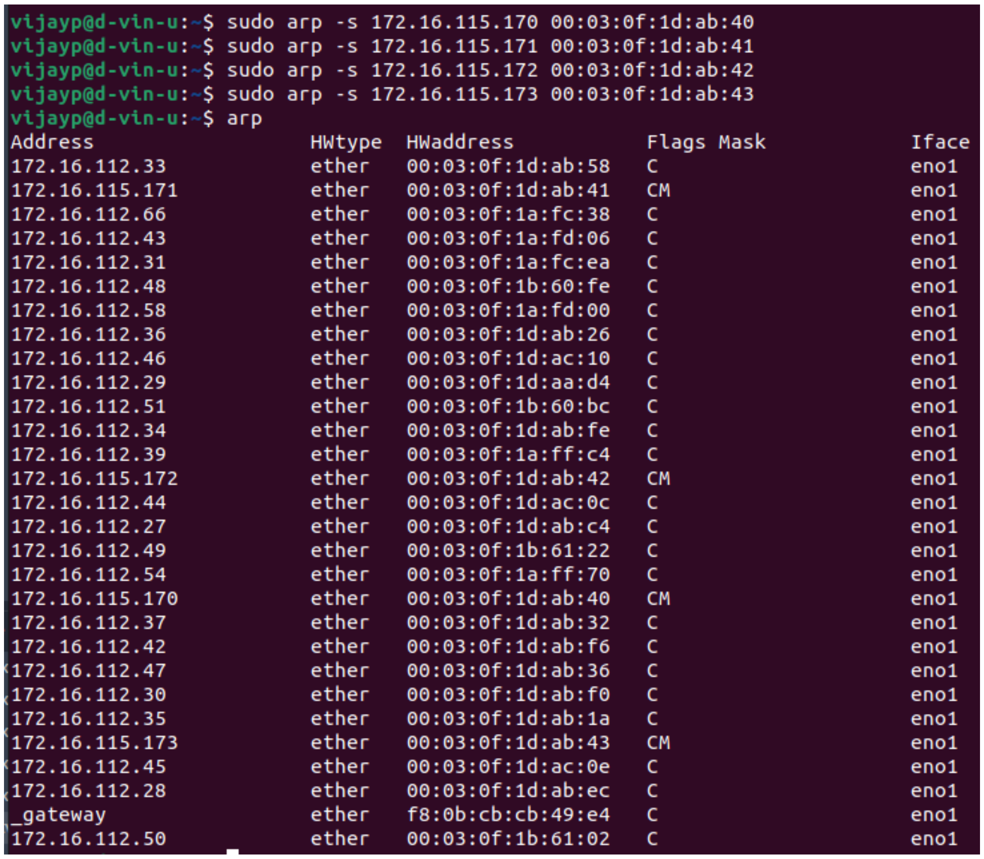
Table Column Description:

- `Address`: The IPv4 address of the machine.
- `HWtype`: The type of connection. In my system it is through ethernet
- `HWaddress`: The MAC address of the machine.
- `Flags Mask`: It tells the address is manually set, learned, published (announced by another node than the requested) or is incomplete.
  - **C**: It tells that entries are dynamically learned by `arp` protocol. Each complete entry in the ARP cache will be marked with the C flag.
  - **M**: It tells that entries have been manually entered/added in the memory instead of dynamically learned from `arp` protocol. Permanent entries are marked with M.
  - **P**: (Published) It tells the host to respond to packets which are ARP request and ARP response.
- `Iface`: It is the interface name.

(b) Check and explain what happens if you try and use the arp command to add or delete an entry to the ARP table. Find out how to add, delete or change entries in the ARP table. Use this mechanism to add at least four new hosts to the ARP table and include a printout.

If network is reachable, arp can be used to make a static entry into the ARP table, otherwise it will show network is unreachable. Similarly, deletion of entry will remove it from arp table.

- `add entry`: To add entry into ARP table, we have to use option `arp -s address hw_addr`. The format of the hw_addr parameter is dependent on the hardware class, but for most classes one can assume that the usual presentation can be used. For the Ethernet class, this is 6 bytes in hexadecimal, separated by colons.
- `delete entry`: To delete entry from ARP table, we have to use option `arp -d address`. Root or net-admin privilege is required to do this. The entry is found by IP address. If a hostname is given, it will be resolved before looking up the entry in the ARP table.



```
vijayp@d-vin-u:~$ sudo arp -s 172.16.115.170 00:03:0f:1d:ab:40
vijayp@d-vin-u:~$ sudo arp -s 172.16.115.171 00:03:0f:1d:ab:41
vijayp@d-vin-u:~$ sudo arp -s 172.16.115.172 00:03:0f:1d:ab:42
vijayp@d-vin-u:~$ sudo arp -s 172.16.115.173 00:03:0f:1d:ab:43
vijayp@d-vin-u:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.112.33            ether   00:03:0f:1d:ab:58   C                     eno1
172.16.115.171           ether   00:03:0f:1d:ab:41   CM                    eno1
172.16.112.66            ether   00:03:0f:1a:fc:38   C                     eno1
172.16.112.43            ether   00:03:0f:1a:fd:06   C                     eno1
172.16.112.31            ether   00:03:0f:1a:fc:ea   C                     eno1
172.16.112.48            ether   00:03:0f:1b:60:fe   C                     eno1
172.16.112.58            ether   00:03:0f:1a:fd:00   C                     eno1
172.16.112.36            ether   00:03:0f:1d:ab:26   C                     eno1
172.16.112.46            ether   00:03:0f:1d:ac:10   C                     eno1
172.16.112.29            ether   00:03:0f:1d:aa:d4   C                     eno1
172.16.112.51            ether   00:03:0f:1b:60:bc   C                     eno1
172.16.112.34            ether   00:03:0f:1d:ab:fe   C                     eno1
172.16.112.39            ether   00:03:0f:1a:ff:c4   C                     eno1
172.16.115.172           ether   00:03:0f:1d:ab:42   CM                    eno1
172.16.112.44            ether   00:03:0f:1d:ac:0c   C                     eno1
172.16.112.27            ether   00:03:0f:1d:ab:c4   C                     eno1
172.16.112.49            ether   00:03:0f:1b:61:22   C                     eno1
172.16.112.54            ether   00:03:0f:1a:ff:70   C                     eno1
172.16.115.170           ether   00:03:0f:1d:ab:40   CM                    eno1
172.16.112.37            ether   00:03:0f:1d:ab:32   C                     eno1
172.16.112.42            ether   00:03:0f:1d:ab:f6   C                     eno1
172.16.112.47            ether   00:03:0f:1d:ab:36   C                     eno1
172.16.112.30            ether   00:03:0f:1d:ab:f0   C                     eno1
172.16.112.35            ether   00:03:0f:1d:ab:1a   C                     eno1
172.16.115.173           ether   00:03:0f:1d:ab:43   CM                    eno1
172.16.112.45            ether   00:03:0f:1d:ac:0e   C                     eno1
172.16.112.28            ether   00:03:0f:1d:ab:ec   C                     eno1
_gateway                 ether   f8:0b:cb:cb:49:e4   C                     eno1
172.16.112.50            ether   00:03:0f:1b:61:02   C                     eno1
```

Figure 15: arp add command output

(c) What are the parameters that determine how long the entries in the cache of the ARP module of the kernel remain valid and when they get deleted from the cache? Describe a trial-and-error method to discover the timeout value for the ARP cache entries.
ARP supports a range of `/proc` interfaces to configure parameters on a global or per-interface basis. The interfaces can be accessed by reading or writing the */proc/sys/net/ipv4/neigh/*/** files.
`base_reachable_time`: Once a neighbor has been found, the entry is considered to be valid for at least a random value between `base_reachable_time/2 and 3base_reachable_time/2`.

An entry's validity will be extended if it receives positive feedback from higher level protocols. Defaults to 30 seconds.
`locktime`: The minimum number of jiffies to keep an ARP entry in the cache. This prevents ARP cache thrashing if there is more than one potential mapping (generally due to network misconfiguration). Defaults to 1 second.

At some point between `base_reachable_time/2 and 3base_reachable_time/2`, the entry will still be in the cache, but it will be marked with a state of STALE. We should be able to view the state with `ip -s neighbor show`.
If the entry hasn't been used and is stale for `gc_stale_time` seconds, it should be eligible to be removed. If `gc_stale_time` passed and marked the entry as okay to be removed, it will be removed when the garbage collector runs (usually after `gc_interval` seconds). Now the problem is that the neighbor entry will not be deleted if it's being referenced. There's a lot of complicated garbage collection stuff, but the important thing to note is that the garbage collector for the route cache only expires entries every 5 minutes (*/proc/sys/net/ipv4/route/gc_timeout* seconds) on a lot of kernels. This means the neighbor entry will have to be marked as stale (maybe 30 seconds, depending on `base_reachable_time`), then 5 minutes will have to go by before the route cache stops referencing the entry, followed by some combination of `gc_stale_time` and `gc_interval` passing before it actually gets cleaned up (so, overall, somewhere between 5-10 minutes will pass).

(d) What will happen if two IP addresses map to the same Ethernet address? Be specific on how all hosts on the subnet operate.
In order for a network device to be able to communicate, the MAC Address it is using must be unique. No other device on that local network subnet can use that MAC Address as Ethernet is using MAC for the addressing. If two devices have the same MAC Address neither computer can communicate properly. It then depends on how the routers and systems on the network are configured. On an Ethernet LAN, this will cause a high number of collisions. Duplicate MAC Addresses on the same LAN are a problem. We can start seeing unreachable host errors due to the collisions as well. The overall results really do vary depending on when the duplicate machine is coming online and how the current infrastructure is setup to handle such items.

7 Local Network Analysis

Local network analysis: Query your LAN using the nmap command to discover which hosts are online. Use a command such as: *nmap –n –sP <Subnet Range >(e.g., 172.16.112.0/26)*
You can choose a different LAN subnet address as well (make sure you report the same in your report explicitly).
Now run the command repeatedly at different times of the day, and find the number of hosts online. Do it for at least 6 times with sufficient time gap. Plot a graph against time to see if there are any hourly trends for when computers are switched ON or OFF in your LAN.

Subnet Range Chosen: **172.16.112.0/22**
Date: 22 March,2022 - 23 March, 2022

Table 5: Local Network Analysis

Time	Num of Hosts Online
00:35	154
10:44	162
15:30	162
18:45	166
23:46	163
10:00	140

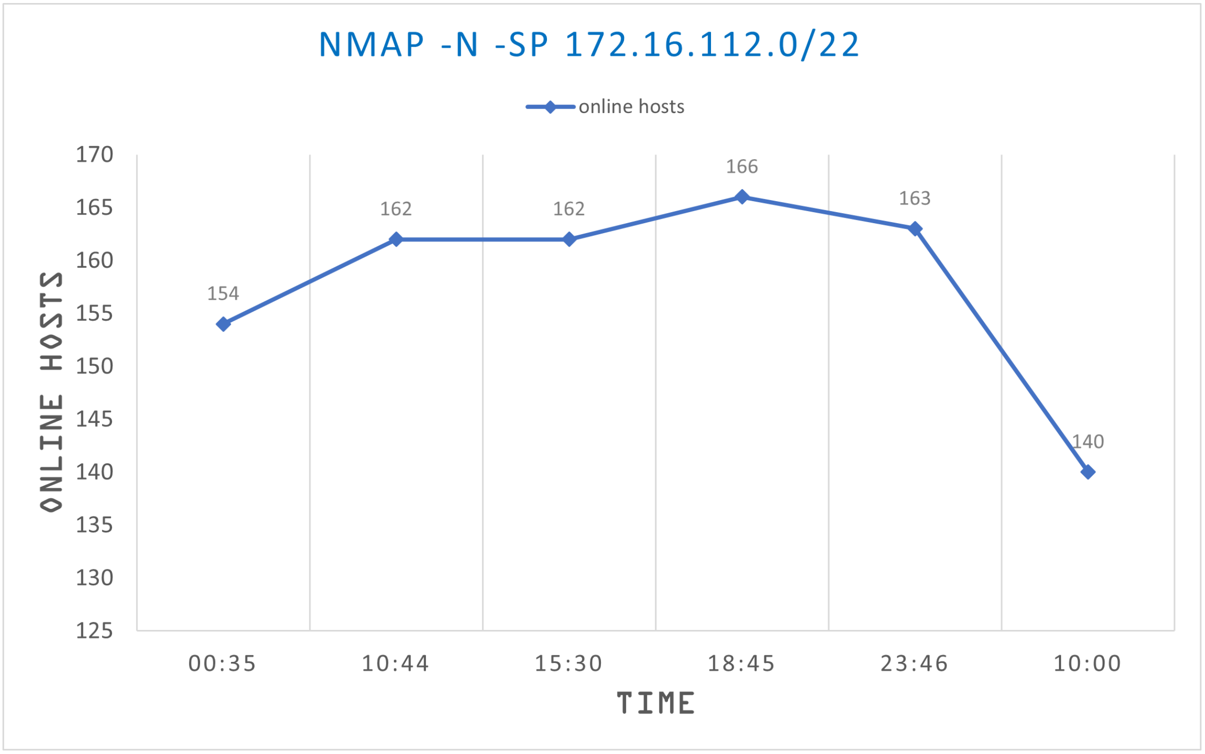


Figure 16: Time vs Online Hosts