CS_342 Assignment 1

Akshat Mittal, 200101011

Aug 14, 2022

1. ***ping* command**

a. ping **-c** <no. of echo requests> <destination IP>

b. ping **-i** <interval in sec> <destination IP>

c. Command: ping **-l** <no. of packets to preload> <destination IP>
Normal users can send **at most 3** packets using this command.

d. Command: ping **-s** <data size in bytes> <destination IP>
Total packet size (**without** IPV4 header) = payload size (32 bytes) + ICMP header size (8 bytes) = **40 bytes**
Total packet size (**including** IPV4 header) = payload size (32 bytes) + ICMP header size (8 bytes) + IPV4 header size (20 bytes) = **60 bytes**

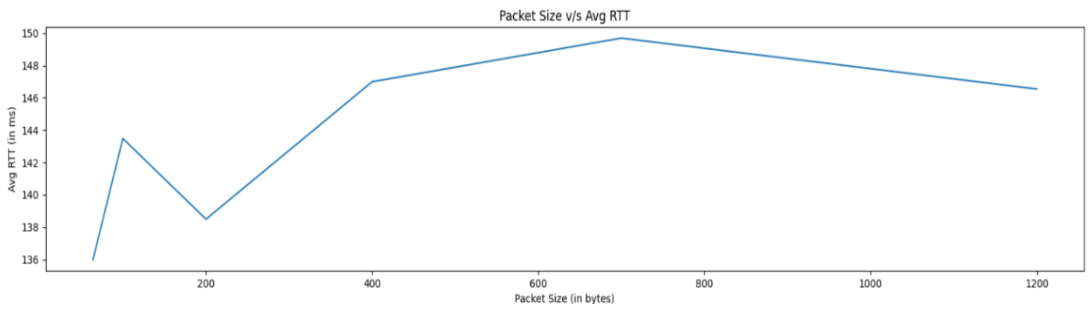
2. **Ping different hosts at different times and record average RTT**

I performed the experiment on 6 different hosts and compiled all the results in following table:

Host Name	**Location**	**Time - 10 am**		**Time - 5 pm**		**Time - 10 pm**		**Avg RTT (in ms)**
		Avg RTT	**Packet Loss**	**Avg RTT**	**Packet Loss**	**Avg RTT**	**Packet Loss**	
google.com (142.250.195.100)	California	121.04 ms	0 %	114.07 ms	0 %	149.35 ms	0 %	128.15 ms
microsoft.com (104.120.173.156)	Telangana	209.73 ms	0 %	197.10 ms	0 %	177.02 ms	0 %	194.62 ms
facebook.com (157.240.23.35)	Tamil Nadu	151.42 ms	0 %	178.03 ms	4 %	174.40 ms	0 %	167.95 ms
codeforces.com (213.248.110.126)	Russia	367.98 ms	0 %	363.37 ms	0 %	483.03 ms	8 %	404.79 ms
github.com (13.234.210.38)	Maharashtra	144.29 ms	0 %	90.88 ms	0 %	98.14 ms	0 %	111.10 ms
Iitg.ac.in (14.139.196.22)	Assam	-	100 %	-	100 %	-	100 %	-

- In general, **RTT (Round Trip Time) increases with increase in geographical distance** as there will be **more hops** required to the destination increasing **propagational**, **computational** and **transmission** delays. But other factors like **network congestion**, **cross traffic,** etc. also play a role leading to inconsistencies in above data.
- There were some packet losses. The reasons could be as follows:
  a. **High network traffic** leading to large queues.
  b. Some of them could be due to **network congestion**, **faulty routers,** or **error in data transmission**.
  c. The packet loss for IITG server is 100% because **ICMP ECHO_REQUESTS are blocked** and therefore, no packets are returned at all.
- I repeated the experiment for github.com with different packet sizes and got following results. **Packets of size more than 1500 bytes were not being received at all**.

Packet Size	**Avg RTT 6 pm**
64 bytes	135.98 ms
100 bytes	143.48 ms
200 bytes	138.49 ms
400 bytes	146.99 ms
700 bytes	149.69 ms
1200 bytes	146.54 ms
1600 bytes	-



- There is an **increase in RTT with increasing packet size** but it not that significant. It is also observed that for packet size more than 1500 bytes, nothing was received because **the default MTU (Maximum Transmission Unit) is 1500 bytes**.
- There are **slight variations in RTT at different times of day** due to variations in network congestion and traffic. The servers of social media sites in India may have more traffic at night than day while Microsoft's server has higher RTT during day due to office work. The California server of Google has higher RTT at night because it is daytime in California.

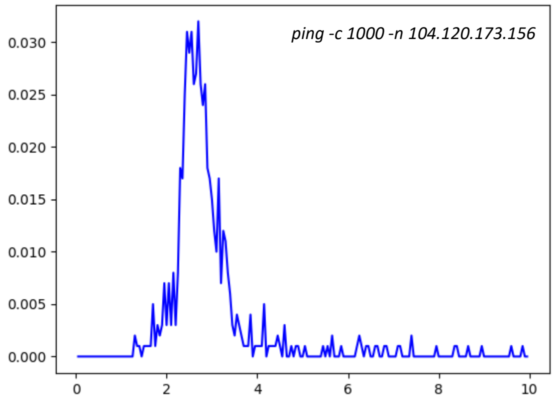
3. ping using *-n* and *-p*

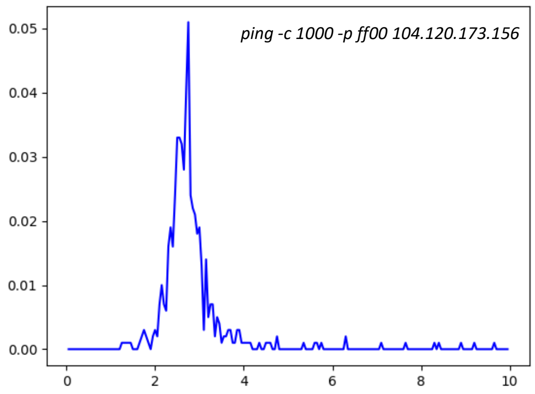
a. Host: intranet.iitg.ac.in (172.17.0.23)
For *ping -c 1000 -n 172.17.0.23*, packet loss rate = 1.7 % (983 received out of 1000)
For *ping -c 1000 -p ff00 172.17.0.23,* packet loss rate = 4.6 % (954 received out of 1000)

b. Following table shows the required result (values are in ***ms***):

Command	Min Latency	Avg Latency	Max Latency	Median Latency
ping -c 1000 -n 104.120.173.156	1.302	2.993	16.083	2.74
ping -c 1000 -p ff00 104.120.173.156	1.257	2.885	15.116	2.71

c. Normal distribution of both is shown below:

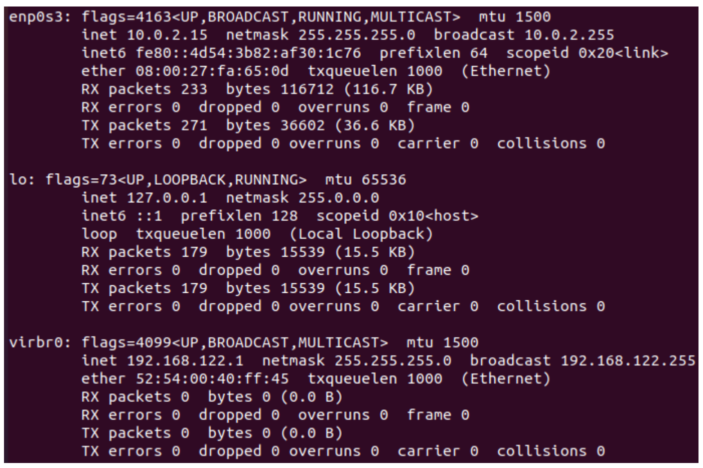




d. Differences between both commands are:
   i. -n : No attempt is made to look up symbolic names for host addresses (in other words there is **no DNS resolution**), this **prevents the overhead for DNS lookup** and causes both the **median** and **average** latency to be lower in this case.
   ii. -p ff00 : The packet **1111111100000000** (8 1's + 8 0's) is sent in this case. This option is generally used to **diagnose data- dependent issues**. Since only a single transition of 1 to 0 is present, it may lead to **synchronization issues** in clock timings thereby **increasing the packet latency rate**.

4. *ifconfig* and *route* commands

a. *ifconfig* is used for diagnosing and configuring network interfaces. It displays **the status of all currently active kernel- resident network interfaces** (those linked with kernel modules). For each interface, it shows:
   i. **status flags** of that interface and its **Maximum Transmission Unit (MTU)**.
   ii. The IP address information of the interface- the **IPv4** (**netmask** and **broadcast** address also) and **IPv6** addresses, its **scope id** and the ethernet layer address.

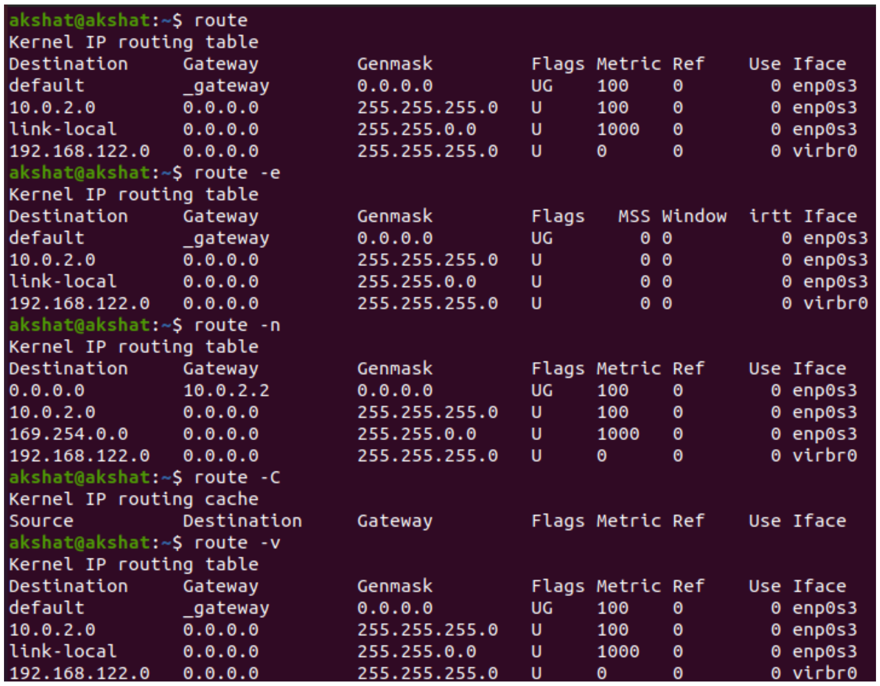


```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::4d54:3b82:af30:1c76  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:fa:65:0d  txqueuelen 1000  (Ethernet)
        RX packets 233  bytes 116712 (116.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 271  bytes 36602 (36.6 KB)
        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 179  bytes 15539 (15.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 179  bytes 15539 (15.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:40:ff:45  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

   iii. Data related to **received and transmitted packets** from the interface.

b. Following options can be provided with ifconfig command:
   i. **-a**: Displays all the interfaces that are available, even if they are not active right now.
   ii. **-s**: Displays the information in lesser detail- the name of each interface, along with important information such as MTU, flags and packet statistics for the interface.
   iii. **-v**: Runs the command in verbose mode, logging additional details in certain error conditions.
   iv. **up <interface name>:** No output; activates an interface if it not already active.
   v. **down <interface name>:** No output; deactivates an interface if it is active.

c. *route* command is used to **work with IP routing table** which store the routing info of some static routes along with the **metric**, i.e., cost associated with that route. We have some **destination routes** along with **genmask** (netmask of destination network), **gateway** (location of the next router one hop away) and **interface** of the network. Each route has **flags** such as **U** (route is up), **H** (target is host) and **G** (use gateway) indicating the status of the routes.



```
akshat@akshat:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
akshat@akshat:~$ route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
192.168.122.0   0.0.0.0         255.255.255.0   U         0 0          0 virbr0
akshat@akshat:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
akshat@akshat:~$ route -C
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
akshat@akshat:~$ route -v
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

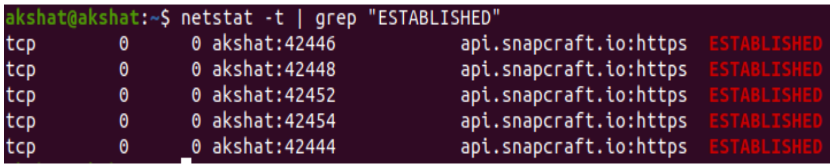
d. Four options of *route* command are:
   i. **-e**: shows output in netstat8 format.
   ii. **-n**: shows numeric address instead of giving symbolic host names.
   iii. **-C**: to operate on kernel's routing cache. Shown output is empty because nothing is there in cache.
   iv. **-v**: gives more verbose description.
   (Output of all options are present in screenshot on right side).

---

5. ***netstat* command**

a. *netstat* command is used to print **network connections**, **routing tables, interface statistics, masquerade connections** and **multicast memberships**. It prints information about the Linux networking subsystem.

b. Command: netstat -t | grep "ESTABLISHED"
   The fields in the output are:



```
akshat@akshat:~$ netstat -t | grep "ESTABLISHED"
tcp        0      0 akshat:42446            api.snapcraft.io:https  ESTABLISHED
tcp        0      0 akshat:42448            api.snapcraft.io:https  ESTABLISHED
tcp        0      0 akshat:42452            api.snapcraft.io:https  ESTABLISHED
tcp        0      0 akshat:42454            api.snapcraft.io:https  ESTABLISHED
tcp        0      0 akshat:42444            api.snapcraft.io:https  ESTABLISHED
```

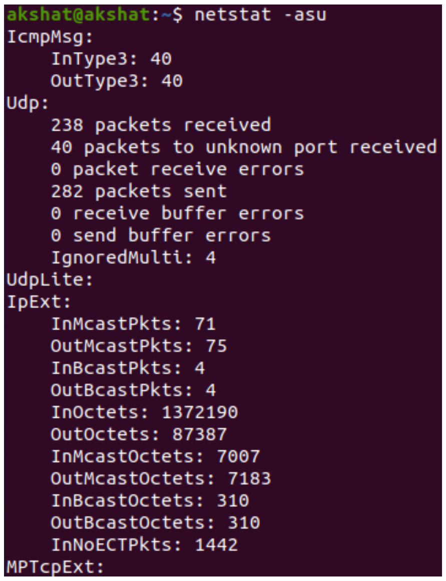
   i. **Proto**: The prototype followed by the socket (TCP in this case).
   ii. **Recv-Q**: The count of bytes not received by the program connected to this socket.
   iii. **Send-Q**: The count of bytes not received by a remote host connected to this socket.
   iv. **Local Address**: Address with port number to connect to the socket from a local end.
   v. **Foreign Address**: Address with port number to which the socket can be connected from a remote end.
   vi. **State**: The status of the socket. **ESTABLISHED** refers to the fact that the socket has established a connection.

c. The command *netstat -r* shows the kernel routing table which consists of following fields:

i. **Destination**: The destination network (or host).
ii. **Gateway**: The gateway used to reach the specified destination.
iii. **Genmask**: The subnet mask used to connect to the destination. 0.0.0.0 represents that there is no mask.
iv. **Flags**: Describe certain characteristics of this route like U (route is up), H (target is host), G (use Gateway), C (cache entry), etc.
v. **MSS**: Default maximum segment size for TCP connections over this route.
vi. **Window**: Default window size for TCP connections over this route.
vii. **irtt**: Initial RTT (Round Trip Time). This is the RTT measured before establishing connection which helps to determine best TCP parameters.
viii. **Iface**: Network Interface to which packets for this route will be sent.

d. Command to display all interfaces: *netstat -i*
Command to count all interfaces: *echo $[$(netstat -i | wc -l) -2]*
Number of interfaces = Number of lines of output – 2 (one line for heading of table, one line for attribute headers) = **3** (in my device).

e. Command: *netstat -asu*
(Output is shown in screenshot on right.)



```
akshat@akshat:~$ netstat -asu
IcmpMsg:
    InType3: 40
    OutType3: 40
Udp:
    238 packets received
    40 packets to unknown port received
    0 packet receive errors
    282 packets sent
    0 receive buffer errors
    0 send buffer errors
    IgnoredMulti: 4
UdpLite:
IpExt:
    InMcastPkts: 71
    OutMcastPkts: 75
    InBcastPkts: 4
    OutBcastPkts: 4
    InOctets: 1372190
    OutOctets: 87387
    InMcastOctets: 7007
    OutMcastOctets: 7183
    InBcastOctets: 310
    OutBcastOctets: 310
    InNoECTPkts: 1442
MPTcpExt:
```

f. The loopback interface is a **virtual interface** used by the computer to communicate with itself. It is always up and available after configuration. Hence, it is used **to identify a router**. Using some other interface may be misleading if that interface itself is down. For the same reason, it is also used as a **termination address** for some routing protocols. Lastly, it is also used for **diagnostics** (e.g. run a web server locally).

6. ***traceroute* tool**

Traceroute is a tool used to track **real-time pathway** of a packet on an IP network from source to destination. It **pings each router on the path** between source and destination. **IP addresses** of all the routers in between along with the **time** taken for **each** hop of packet in its route to the destination is also shown for each router on the path.

a.

Host Name	IP Address	Location	Count – 11 AM	Count - 6 PM	Count – 11 PM
google.com	142.250.195.100	California	14	15	14
microsoft.com	104.120.173.156	Telangana	13	12	14
facebook.com	157.240.23.35	Tamil Nadu	15	16	15
codeforces.com	213.248.110.126	Russia	18	16	16
github.com	13.234.210.38	Maharashtra	17	20	18
Iitg.ac.in	14.139.196.22	Assam	Time Out	Time Out	Time Out

Following hops are common between multiple hosts:
i. **First two** hops were common in all routes, i.e., **192.168.76.191** and **192.168.27.237**.
ii. In google and microsoft, **4th hop** was common, i.e., **192.168.31.201**.
iii. In github and facebook, **5th hop** was common, i.e., **192.168.31.205**.

b. The results of the experiment suggested that there are **changes in the routes** when the hosts are pinged at the different times in a day. The following could be some reasons:
i. Variation in network congestion causes the packets to take different routes at different times of the day to **reduce overall congestion**.

ii. As a method of **load balancing** the destination server may utilize multiple hosts to serve the client requests at different times of the day which changes the destination IP and also the route to the destination.

c. In all routes, there were such cases. This could be mainly because of 3 reasons:
  i. The nodes in between the source and destination are **congested with high ICMP traffic** and thus block any further incoming traffic.
  ii. There may be **firewalls** to block ICMP traffic.
  iii. The **packet had to be dropped** for some reason, and it **could not reach the router**.

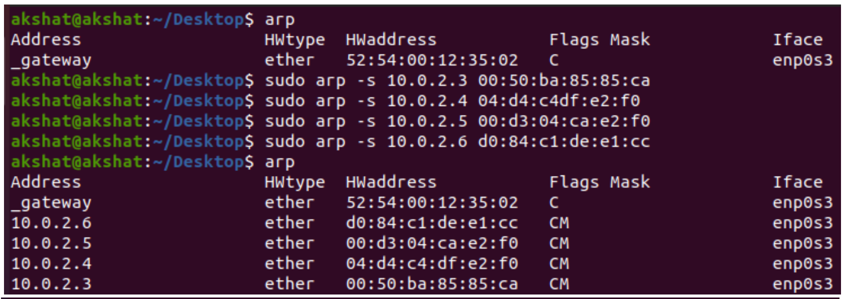
d. Yes, it is possible because of **difference in packets** sent via these commands. *ping* sends ICMP packets while traceroute sends UDP packets. If a host blocks only ICMP requests, this situation is possible.
Traceroute works by **limiting the TTL and incrementing by 1** at each step. When the TTL comes to 0 before the packet is processed, traceroute sends back the packet with a "**Time Exceeded**" message. The source router waits for this message instead of the **standard echo response** that ping receives. This can help traceroute find the path when ping cannot.

---

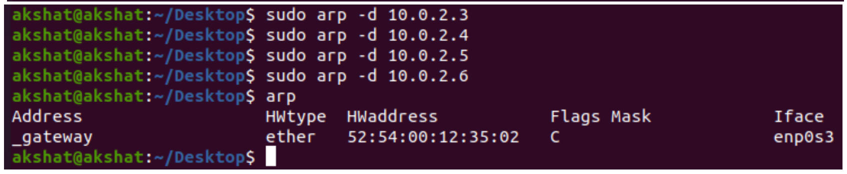
7. ***arp* command**

a. Command: arp
Following are the description of various columns of the table:
  i. **Address**: IP address of the network connection.
  ii. **HWtype**: The hardware type of the device. For eg. ether stands for ethernet.
  iii. **HWaddress**: The MAC address (hardware address) of the hardware device.
  iv. **Flags Mask**: Additional description about type of entry like C (for complete entry), M (for permanent entry) and P (for published entry).
  v. **Iface**: Network interface used for establishing the connection.

b. When we try to add or delete entry in ARP table, it asks us for **root user** privileges. Further, given MAC address must be compatible with given IP address.
Command to add:
sudo arp -s <IP address> <MAC address>
Command to delete:
sudo arp -d <IP address>



```
akshat@akshat:~/Desktop$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:02   C                     enp0s3
akshat@akshat:~/Desktop$ sudo arp -s 10.0.2.3 00:50:ba:85:85:ca
akshat@akshat:~/Desktop$ sudo arp -s 10.0.2.4 04:d4:c4df:e2:f0
akshat@akshat:~/Desktop$ sudo arp -s 10.0.2.5 00:d3:04:ca:e2:f0
akshat@akshat:~/Desktop$ sudo arp -s 10.0.2.6 d0:84:c1:de:e1:cc
akshat@akshat:~/Desktop$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:02   C                     enp0s3
10.0.2.6                 ether   d0:84:c1:de:e1:cc   CM                    enp0s3
10.0.2.5                 ether   00:d3:04:ca:e2:f0   CM                    enp0s3
10.0.2.4                 ether   04:d4:c4:df:e2:f0   CM                    enp0s3
10.0.2.3                 ether   00:50:ba:85:85:ca   CM                    enp0s3
```



```
akshat@akshat:~/Desktop$ sudo arp -d 10.0.2.3
akshat@akshat:~/Desktop$ sudo arp -d 10.0.2.4
akshat@akshat:~/Desktop$ sudo arp -d 10.0.2.5
akshat@akshat:~/Desktop$ sudo arp -d 10.0.2.6
akshat@akshat:~/Desktop$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:02   C                     enp0s3
akshat@akshat:~/Desktop$
```

c. The **ARP timeout** value determines the duration of how long an entry remains in the ARP table cache. It is configurable and can be changed from its default value by specifying it.
A trial-and-error method of finding ARP timeout value is by **adding a new static entry in the ARP table and there on viewing the ARP table in regular intervals by running *arp***.
However, a more straightforward approach would be to run *arp timeout* to see ARP timeout value immediately in the terminal.

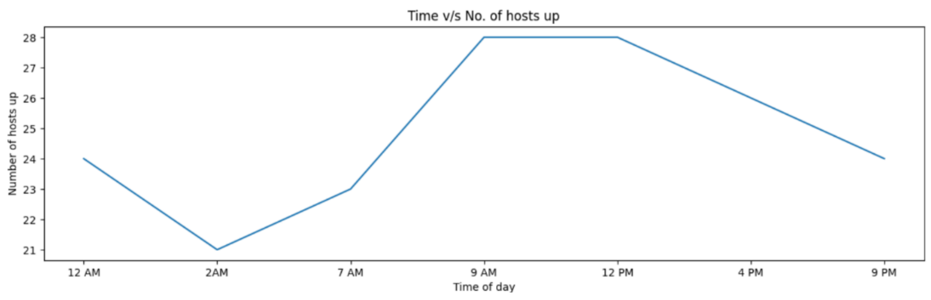
d. **Yes**, it is possible to have two IP addresses mapped to the same ethernet address. All hosts with an IP address are mapped to a network interface device by its MAC address from where it sends/receives packets. If multiple address on same subnet maps to same MAC (Ethernet) address, it may **cause interference and confusion in network transmission** of right packets to the right host.

---

8. ***nmap* command**

- I used the command: *nmap -n -sP 172.16.112.0/26* for this experiment and obtained following results:

Time	Number Of Hosts Up
12 AM	24
2 AM	21
7 AM	23
9 AM	28
12 PM	28
4 PM	26
9 PM	24



- From the above graph, we can see that number of hosts up remains almost same throughout the day.

---

9. ***nslookup* command**

a. To find IP address of a given host, command used is: nslookup <domain>
b. To find Domain name for a given IP address, command used is: nslookup <IP address>
c. To find mail servers for a given domain, command used is: nslookup -type=mx <domain>