भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

Computer Networks Lab (CS 342) Lab Assignment - 01

Student Details:

Name: AB Satyaprakash Roll Number: 180123062 Department: Mathematics and Computing

Question 1:

a. The option required is **–c**. Example: *ping –c 2 google.com* (2=number of packets)
b. The option required is **–i**. Example: *ping –i 0.5 google.com* (0.5=0.5s instead of 1s time interval which is the default)
c. The command to send ECHO_REQUEST packets to the destination one after another without waiting for a reply is : ***ping -l <preload> <destination>***. The limit for sending such echo request packets by a normal user is **3**.
d. The command to set the ECHO_REQUEST payload/data size is: ***ping –s <payload size> <hostname-IP>***. The payload size is in bytes and is 56 by default. An extra 28 bytes is sent per header for the IPv4. Thus *60 bytes* (32+28) will be the total packet size if the payload size is set at 32 bytes.

Question 2:

Hosts: (with IP addresses and geographical location)

a. *google.com - 2404:6800:4003:803::200e (geographical location = Singapore)*
b. *oracle.com - 137.254.120.50 (geographical location = San Francisco, California, United States)*
c. *amazon.com - 176.32.98.166(geographical location = Virginia Beach, Virginia, United States)*
d. *wikipedia.org - 103.102.166.224 (geographical location = San Francisco, California, United States)*
e. *sprinklr.com - 13.33.168.0/22 (geographical location = Seattle, United States)*
f. *codeforces.com - 81.27.240.126 (geographical location = Saint Petersburg, Russia)*

Host Name	Location (Distance)	Morning (9:30 – 10:30 AM)	Afternoon (2:30-3:30 PM)	Evening (8:30 – 9:30 PM)
google.com	Singapore 5484.3km	Avg. RTT = 234.436 ms	Avg. RTT = 210.121 ms	Avg. RTT = 184.452 ms
oracle.com	San Francisco 12,903 km	Avg. RTT = 451.618 ms	Avg. RTT = 418.171 ms	Avg. RTT =593.217 ms
amazon.com	Virginia 13,381 km	Avg. RTT = 426.304 ms	Avg. RTT = 400.895 ms	Avg. RTT = 807.259 ms
wikipedia.org	San Francisco 12,903 km	Avg. RTT = 199.220 ms	Avg. RTT = 386.526 ms	Avg. RTT = 281.053 ms
sprinklr.com	Seattle 11,947 km	Avg. RTT = 133.858 ms	Avg. RTT = 222.810 ms	Avg. RTT = 275.723 ms
codeforces.com	St. Petersburg 6,162 km	Avg. RTT = 460.419 ms	100% packet-loss	Avg. RTT =762.250 ms

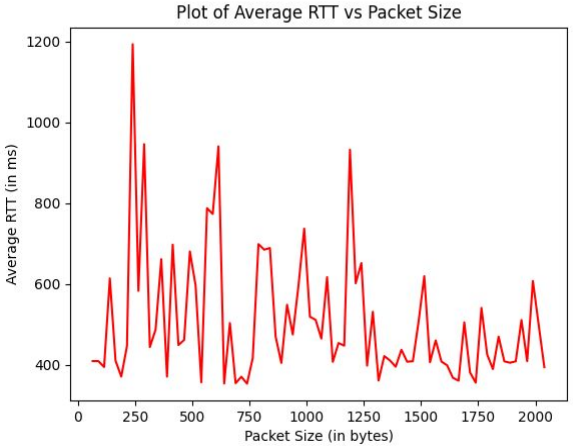
a. The table above shows the **average RTT of 6 different hosts pinged at 3 different times** of the day and also their geographical distance from my current location.
There seems to be **no correlation** as such between the geographical distance and the average RTT. For example, hosts **wikipedia.org** and **oracle.com** are both located at similar distances but **oracle.com** (593.217 ms in the evening) has a much larger RTT as compared to **wikipedia.org** (281.053 ms in the evening). Similarly, hosts **codeforces.com** has a much larger RTT compared to **sprinklr.com**, although it's at almost half the distance of the latter. The average RTT depends more on the number of routers the data has to go through.

a. I experienced **100% packet-loss** with host **codeforces.com**, in the afternoon. I also pinged **codeforces.com** in the evening changing the packet size to 108 bytes and received the following result.

```
--- codeforces.com ping statistics ---
12 packets transmitted, 11 received, 8.33333% packet loss, time 11012ms
rtt min/avg/max/mdev = 284.484/440.333/933.655/177.995 ms
```

Packet loss occurs when one or more **packets** of data travelling across a computer network fail to reach their destination. Packet loss is either caused by *errors in data transmission*, typically across wireless networks, or *network congestion*.

b. I chose **oracle.com**, the host in **San Francisco** and varied the packet size from **64** bytes to **2048** bytes (**25** bytes increased at a time). I obtained the following plot.



c. **Packet Size** : As we can conclude from the plot above, *changing the packet size, affects the average rtt*. The average rtt is seen to first rise and then fall irregularly. Incase of worse network traffic we could have packet loss after a certain packet size.
Time of the day : As seen from the table the same host can have *different RTTs at different times of the day*. This is because RTT depends on the network speed and traffic which can vary during the course of the day.

Question 3:

IP Address = 31.13.79.35 (corresponding to **facebook.com**)

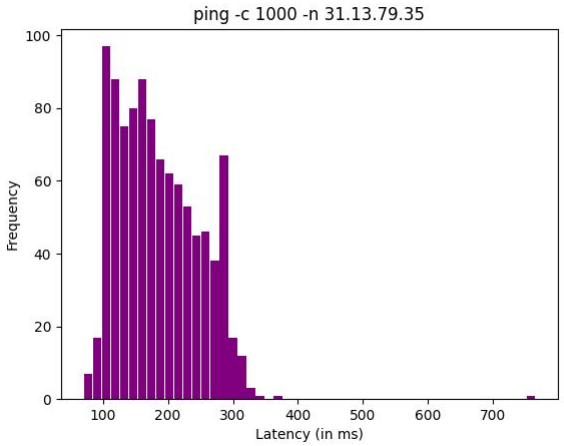
a. The **packet loss rate = 0%** in either case.
b. Latency:

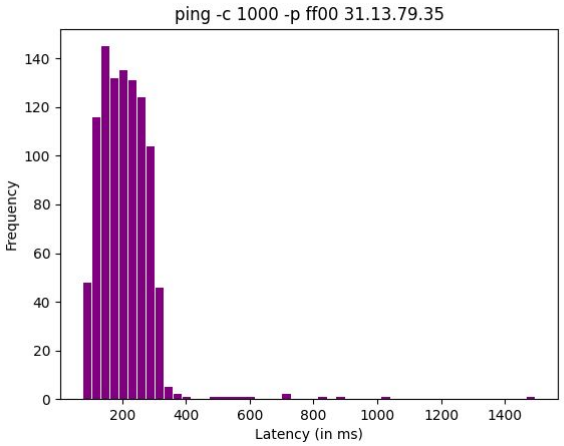
```
imperial_lord@lord:~$ ping -c 1000 -n 31.13.79.35 >> q31.txt
```

- minimum: 70.419 ms
- maximum: 764.680 ms
- mean: 185.642 ms
- median: 175.0 ms

```
imperial_lord@lord:~$ ping -c 1000 -p ff00 31.13.79.35 >> q32.txt
```

- minimum: 76.302 ms
- maximum: 1495.835 ms
- mean: 208.887 ms
- median: 201.0 ms





c.

d. The following are the differences between **-n** and **-p ff00:**
   - Using **-n**, decreases the latency (max values ~ 765 ms), because it removes the need to look up for symbolic names, thus removing the lookup latency.
   - Using **-p** makes it easier to diagnose data dependent problems.

Question 4:

a. After running **ifconfig**, the following is the output:

```
satyapra@E1-Maths72:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr c0:3f:d5:35:fc:a0
          inet addr:172.16.70.72  Bcast:172.16.71.255  Mask:255.255.252.0
          inet6 addr: fe80::a8f2:fc32:72ee:abbc/64 Scope:Link
          inet6 addr: fe80::73b6:ae5f:d477:a2cc/64 Scope:Link
          inet6 addr: fe80::8058:fcc5:faa0:cea8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13634120 errors:0 dropped:18127 overruns:0 frame:0
          TX packets:282269 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1789607704 (1.7 GB)  TX bytes:35938656 (35.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1111 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1111 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:138625 (138.6 KB)  TX bytes:138625 (138.6 KB)
```

The output has been described below:

A. Interfaces :

There are 2 network interfaces on the box:
- **eth0** is a physical interface representing Ethernet network cards.
- **lo** is a special virtual network interface called loopback device.

B. Interface Details :

Let's look closely at details of **ifconfig** output:
- **Link encap** shows how packets are encapsulated for transmission.
- **HWaddr** is the hardware address of the ethernet interface (also known as MAC address).
- **inet addr** is the IPv4 address assigned to the interface.
- **Bcast** is the broadcast address for the interface.
- **Mask** is the network mask for the interface.
- **inet6** addr is the IPv6 address assigned to the interface.
- **Scope** is scope of IPv6 address
- **UP** indicates that kernel modules related to the interface have been loaded and the 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 the interface is ready to accept data.
- **MULTICAST** indicates that the interface supports multicasting.
- **MTU** is the maximum transmission unit.

- **Metric** determines the cost of using the interface. Interfaces with lower cost have higher priority.

C. **Interface Stats** :

- **RX packets** is a total number of packets received.
- **RX errors** show the total number of packets received with error.
- **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 the rate at which a buffer gets full and the 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 packets** is the total number of packets transmitted.
- *TX errors, TX dropped and TX overruns are similar to RX equivalents.*
- **TX carriers** are a number of packets that experienced loss of carriers. This usually happens when a link is flapping.
- **TX collisions** is a number of transmitted packets that experienced Ethernet collisions.
- **TX txqueuelen** is the length of the transmission queue.
- **RX bytes** is a total number of bytes received over interface.
- **TX bytes** is a total number of bytes transmitted over interface.

b. The **options** that can be provided are:

- **-a** This option tells *ifconfig* to show information about all interfaces, both active and inactive.
- **up** This activates an interface if it is not already active.
- **down** The counterpart to up, this deactivates the specified interface.
- **netmask** *[addr]* Using the "netmask" option allows you to set the network mask for a given interface.

c. After running **route**, the following is the output:

```
imperial_lord@lord:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.43.1    0.0.0.0         UG    600    0        0 wlo1
10.0.0.0        172.18.16.3     255.0.0.0       UG    0      0        0 ppp0
agnigarh.iitg.a 192.168.43.1    255.255.255.255 UGH   0      0        0 wlo1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 wlo1
172.16.0.0      172.18.16.3     255.252.0.0     UG    0      0        0 ppp0
192.168.0.0     172.18.16.3     255.255.0.0     UG    0      0        0 ppp0
192.168.43.0    0.0.0.0         255.255.255.0   U     600    0        0 wlo1
```

The **route** command is used to manipulate the kernel's routing tables. Its primary use is to set up static routes to specific hosts or networks. The output has been described below :

- **Destination** column identifies the destination network.
- **Gateway** column identifies the defined gateway for the specified network.
- **Genmask** column shows the netmask for the network
- **Flags** include :
  - **U:** (the route is up)
  - **H:** (target is a host)
  - **G:** (use gateway)
- **Metric**: The distance to the target usually measured in hops.
- **Ref** is the number of references to this route
- **Use** is the 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** column shows the network interface.

d. The options that can be provided are:

- **del** is used to delete a route *[route del <dest>]*
- **add** is used to add a route *[route add <dest> gw <gateway>]*
- **get** is used to look and display the route for a destination *[ip route get <dest>]*
- **-n** shows the routing table for all IPs bound to the server *[route -n]*

```
imperial_lord@lord:~$ sudo route del default
imperial_lord@lord:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        172.18.16.3     255.0.0.0       UG    0      0        0 ppp0
agnigarh.iitg.a 192.168.43.1    255.255.255.255 UGH   0      0        0 wlo1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 wlo1
172.16.0.0      172.18.16.3     255.252.0.0     UG    0      0        0 ppp0
192.168.0.0     172.18.16.3     255.255.0.0     UG    0      0        0 ppp0
192.168.43.0    0.0.0.0         255.255.255.0   U     600    0        0 wlo1
```

```
imperial_lord@lord:~$ sudo route add default gw 192.168.43.1
imperial_lord@lord:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.43.1    0.0.0.0         UG    0      0        0 wlo1
10.0.0.0        172.18.16.3     255.0.0.0       UG    0      0        0 ppp0
agnigarh.iitg.a 192.168.43.1    255.255.255.255 UGH   0      0        0 wlo1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 wlo1
172.16.0.0      172.18.16.3     255.252.0.0     UG    0      0        0 ppp0
192.168.0.0     172.18.16.3     255.255.0.0     UG    0      0        0 ppp0
192.168.43.0    0.0.0.0         255.255.255.0   U     600    0        0 wlo1
```

```
imperial_lord@lord:~$ ip route get default
local 0.0.0.0 dev lo src 127.0.0.1 uid 1000
    cache <local>
```

```
imperial_lord@lord:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.43.1    0.0.0.0         UG    0      0        0 wlo1
10.0.0.0        172.18.16.3     255.0.0.0       UG    0      0        0 ppp0
14.139.196.11   192.168.43.1    255.255.255.255 UGH   0      0        0 wlo1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 wlo1
172.16.0.0      172.18.16.3     255.252.0.0     UG    0      0        0 ppp0
192.168.0.0     172.18.16.3     255.255.0.0     UG    0      0        0 ppp0
192.168.43.0    0.0.0.0         255.255.255.0   U     600    0        0 wlo1
```

Question 5:

a. The **netstat** command generates displays that show network status and protocol statistics. You can display the status of TCP and UDP endpoints in table format, routing table information, and interface information. **netstat** displays various types of network data depending on the command line option selected. These displays are the most useful for system administration. The syntax for this form is: netstat [-m] [-n] [-s] [-i | -r] [-f address_family]

a. **netstat -at | grep ESTABLISHED** is used to show the established TCP connections.

```
satyapra@E1-Maths72:~$ netstat -at | grep ESTABLISHED
tcp        0     36 172.16.70.72:ssh        172.18.16.33:51892      ESTABLISHED
```

b. **netstat -r (netstat --route)** shows the **routing** table. It displays the kernel routing table in the way we've been doing with **route**. A routing table displays the information on how packets should be transferred and compute the next hop for a packet.

```
satyapra@E1-Maths72:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         172.16.68.254   0.0.0.0         UG        0 0          0 eth0
link-local      *               255.255.0.0     U         0 0          0 eth0
172.16.68.0     *               255.255.252.0   U         0 0          0 eth0
```

The output fields are as follows:

- **Destination** column identifies the destination network.
- **Gateway** column identifies the defined gateway for the specified network (* means it's not set).
- **Genmask** column shows the netmask for the network.
- **Flags** include :
  - **U:** (the route is up)
  - **H:** (target is a host)
  - **G:** (use gateway)
- **MSS Window** is the minimum window size for a TCP connection
- **irtt** is the initial round-trip time
- **Iface** column shows the network interface.

c. **netstat -i** is used to display the status of all network interfaces (the option is **-i**).

```
satyapra@E1-Maths72:~$ netstat -i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0  14366452      0  19270 0        305816      0      0      0 BMRU
lo        65536 0      1215      0      0 0          1215      0      0      0 LRU
```

To figure out the number of network interfaces on my computer I used: **netstat -i | tail -n +3 | wc -l** . It's 2.

```
satyapra@E1-Maths72:~$ netstat -i | tail -n +3 | wc -l
2
```

d. **netstat -asu** is used to display the **statistics(s)** of **all(a) UDP(u)** connections (the option is **-asu**).

```
satyapra@E1-Maths72:~$ netstat -asu
IcmpMsg:
    InType0: 14
    InType3: 46
    InType8: 17
    OutType0: 17
    OutType3: 46
    OutType8: 465
Udp:
    3626991 packets received
    24 packets to unknown port received.
    0 packet receive errors
    18465 packets sent
    IgnoredMulti: 1115114
UdpLite:
IpExt:
```

```
    InMcastPkts: 3625624
    OutMcastPkts: 17026
    InBcastPkts: 1115138
    OutBcastPkts: 2
    InOctets: 1190917963
    OutOctets: 34180409
    InMcastOctets: 899009985
    OutMcastOctets: 6980325
    InBcastOctets: 175285036
    OutBcastOctets: 108
    InNoECTPkts: 5792435
```

e. A **loopback** interface is a logical, virtual interface which can be used to emulate a physical interface. Loopback interface's IP Address determines a router's **OSPF** Router ID. It is always up and allows **Border Gateway Protocol (BGP)** neighborship between two routers to stay up even if one of the outbound physical interfaces connected between the routers is down. It's also used for troubleshooting, diagnostics and connecting servers on the local machine.

```
satyapra@E1-Maths72:~$ ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1247 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1247 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:155639 (155.6 KB)  TX bytes:155639 (155.6 KB)
```

Question 6:

a. Traceroute is a network diagnostic **tool used** to track in real-time the pathway taken by a packet on an IP network from source to destination, reporting the IP addresses of all the routers it pinged in between. It also records the time taken for each hop the packet makes during its route to the destination.

b. **The first two hops for all the hosts were the same**. The first hop is to the **home** router. The second hop is through the **ISP's provider** (**airtel** in my case). These are the same in all hosts. Other than these two initial hops, hosts having nearby geographical locations may have some common hops because they might have the same internet paths for long distance transmission.

```
traceroute to google.com (142.250.67.46), 64 hops max
  1   192.168.43.1  2.274ms  2.719ms  2.688ms
  2   192.168.26.1  872.008ms  150.782ms  80.729ms
```

Host Name	Location (Distance)	Morning (9:30 – 10:30 AM)	Afternoon (2:30-3:30 PM)	Evening (8:30 – 9:30 PM)
google.com	Singapore 5484.3km	Hop Count = 11	Hop Count = 09	Hop Count = 10
oracle.com	San Francisco 12,903 km	Hop Count = 12	Hop Count = 12	Hop Count = 12
amazon.com	Virginia 13,381 km	Hop Count = 10	Hop Count = 11	Hop Count = 10
wikipedia.org	San Francisco 12,903 km	Hop Count = 07	Hop Count = 07	Hop Count = 08
sprinklr.com	Seattle 11,947 km	Hop Count = 10	Hop Count = 08	Hop Count = 11
codeforces.com	St. Petersburg 6,162 km	Hop Count = 09	Hop Count = 07	Hop Count = 09

c. The destination router might be connected to 2 or more ISPs. If the connections are doing **load balancing**, it's possible that route to the same host will change. Load balancing means that to avoid that a single connection gets overloaded until the router uses the next one, it uses a mechanism as round-robin to evenly distribute the traffic among all its connections.

d. **Traceroute** requires a response from the target server and each of the intermediate hops to create its output. If a router doesn't generate a **time-to-live (TTL)** exceeded response, traceroute will not know anything about that hop. A hop that outputs * * * means that the router at that hop doesn't respond to the type of packet you were using for the **traceroute**. The reasons why traceroute doesn't find complete paths to a host are:

   i. The target host could not be reached, i.e, the packets could not reach there and return due to **network issues**
   ii. The packets reached but their return was intentionally blocked by a **firewall or other security measures**.

   We can clearly say that the host is reachable if the traceroute stops before reaching the maximum number of hops

e. **Yes**, it is possible to find the route to hosts that don't respond to ping experiments. This is precisely where **traceroute** comes into play. The traceroute in case of Linux uses **UDP packets** whereas **ping** uses **ICMP echo requests**. While the former is seldom blocked, the **latter is usually blocked because of security measures or firewalls**. So if ping experiments fail, traceroute will still succeed.

Question 7:

a. The full **arp** table of a UNIX machine can be seen by using the command `arp -a` . **Address Resolution Protocol (ARP)** is the method for finding a host's Link Layer (**MAC) address** when only its **IP address** is known. The ARP table is used to maintain a correlation between each MAC address and its corresponding IP address.

```
satyapra@E1-Maths72:~$ arp -a
? (172.16.68.183) at 00:03:0f:20:56:94 [ether] on eth0
? (172.16.68.182) at 00:03:0f:20:57:28 [ether] on eth0
? (172.16.68.187) at 00:03:0f:22:51:9e [ether] on eth0
? (172.16.70.53) at d4:3d:7e:ce:84:6c [ether] on eth0
......
```

Each line of the table has the following information (easily understood):

- IP Address is the information in the incoming packet
- MAC Address: The arp table maps each **IP** address to some **MAC** address
- Hardware type is Ethernet ([**ether**]) in our case. It tells the type of network hardware
- Network Interface is **eth0** in our case.

b. We use the following command to add an entry : arp -s <IP address> -i <Network Interface> <MAC address>
We use the following command to delete an entry : arp -d <IP address>
For example I added 2 hosts:
   - IP : **172.16.70.13** and MAC : **64:51:06:43:be:ad** on **eth0**
   - IP : **172.16.70.12** and MAC : **64:51:06:43:be:af** on **eth0**

```
satyapra@E1-Maths72:~$ sudo arp -s 172.16.70.13 -i eth0 64:51:06:43:be:ad
satyapra@E1-Maths72:~$ sudo arp -s 172.16.70.12 -i eth0 64:51:06:43:be:af
satyapra@E1-Maths72:~$ arp -a | grep PERM
? (172.16.70.13) at 64:51:06:43:be:ad [ether] PERM on eth0
? (172.16.70.12) at 64:51:06:43:be:af [ether] PERM on eth0
```

c. **ARP** only works between devices in the **same IP subnet**.
When device A with IP address A needs to send a packet to device B with IP address B, the first thing it does is consulting its routing table to determine if IP address B belongs to a subnet it can directly reach through its network interface(s).If it does, then devices A uses ARP to map IP address B to a physical Ethernet address, and then sends an Ethernet frame to that address. But if the two IP Addresses are on different subnets, the device will follow a completely different logic: it will look in its routing table for a route to the destination network, and then it will send its packet to the appropriate router (or to its default gateway if no more specific route is present); in this scenario, ARP will be used to find the hardware address of the router, because the destination IP address has already be deemed to not be directly reachable, so the packet must be delivered to a router which can take care of it.

d. **Observation:** In my case, I pinged the address **172.16.70.75** once. It was reachable and it responded with 0% packet loss. Then I removed the IP address from the arp table and added it with a different MAC address. The next ping detected the IP address as unreachable.
Reason: The ping command is able to send ECHO requests to the IP address only if the MAC address matches from the arp table. Since the MAC Address of the IP address 172.16.70.75 was forcefully changed in the arp table, on pinging 172.16.70.75, the ICMP request packet is sent to the IP Address but there is a mismatch between the IP address stated in the header of the ICMP request packet and the IP address corresponding to the given MAC in the arp table. This results in the 100% packet loss.

```
satyapra@E1-Maths72:~$ ping -c 1 172.16.70.75
PING 172.16.70.75 (172.16.70.75) 56(84) bytes of data.
64 bytes from 172.16.70.75: icmp_seq=1 ttl=64 time=2.63 ms

--- 172.16.70.75 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.633/2.633/2.633/0.000 ms
satyapra@E1-Maths72:~$ sudo arp -d 172.16.70.75
satyapra@E1-Maths72:~$ sudo arp -s 172.16.70.75 -i eth0 64:51:06:43:be:12
satyapra@E1-Maths72:~$ ping -c 1 172.16.70.75
PING 172.16.70.75 (172.16.70.75) 56(84) bytes of data.

--- 172.16.70.75 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Question 8:

a. The command to check which of my PCs subnet are up is: sudo nmap -sn 172.16.70.72/22

b. The firewall settings of my PC can be detected by nmap -sA <IP Address>

c. The table below shows the number of hosts online at 6 different times of the day. The plot drawn on its side shows that there is no correlation as such between time and the number of PCs online. The number reached a maximum of 249 (at 3 pm) and a minimum of 241 (at 7 pm). (I plotted the time on X axis in a 24 hr format)

Time	Number of hosts online
9:00	246
11:00	241
13:00	245
15:00	249
19:00	241
22:00	243