NAME :- ANURAG RAMTEKE
ROLL NUMBER:-150101010
NETWORKS LAB ASSIGNMENT 1

1) **a)** **ping -c [count] [target]** is the command used to specify the number of echo requests to send with ping.

b) **ping -i [interval] [target]** is the command used to specify the time interval in seconds between two successive pings.

c) **i) ping -l [preload][target]** is used to send ECHO_REQUEST packets to the destination one after another without waiting for a reply
ii) 3 packets is the limit for sending such ECHO_REQUEST packets by normal users and the time limit for it is **200 ms**.

d) **i)ping -s [packet_size] [target]** is used to set the ECHO_REQUEST packet size (in bytes)
ii) If the Packet Size is set to 64 bytes then **92 bytes** will be the **total packet size** of which **20 bytes** will be **IP header** and **8 bytes** will be **ICMP header**.

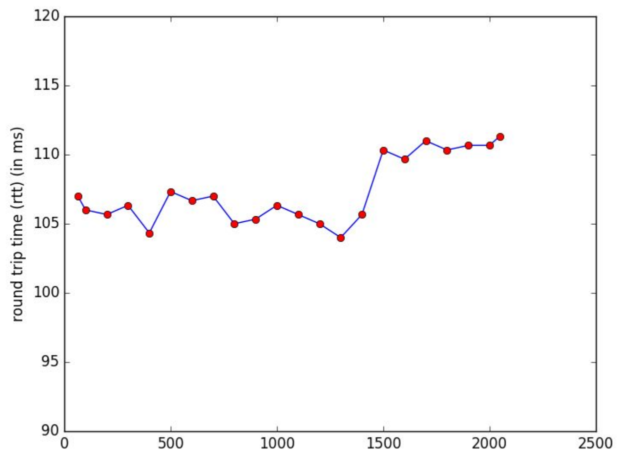
Here, target is the destination address, count is the number of packets to be sent, interval is the interval between sending each packet, preload is the number of packets to be sent not waiting for the reply.

2) The pinging of the hosts are done using the tool ***http://ping.online-domain-tools.com/*** which is located in **Czech Republic.** The pinging of the host is done 3 times within a single day at **1 a.m.**, **10 a.m.** and **6 p.m.** by sending **20 packets** each time. The name of the hosts, their location, distance of the host from Czech Republic, the rtt(round trip time) and the packet loss is given in the table below :-

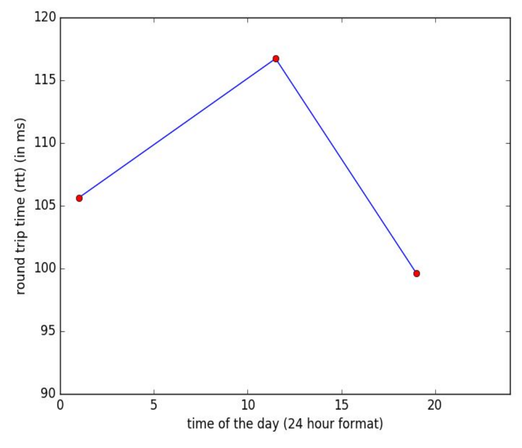
Host_name	Host location	Distance from host(km)	rtt(ms) @ 1 hours	Packet loss @ 1 hours	rtt(ms) @ 10 hours	Packet loss @ 10 hours	rtt(ms) @ 18 hours	Packet loss @ 18 hours	Average rtt(ms)
Instagram.com	USA	8,229	106	0%	117	0%	108	0%	**110.33**
Linkedin.com	USA	8,229	97	0%	116	0%	101	0%	**104.67**
Internshala.com	INDIA	6,393	128	0%	125	0%	126	0%	**126.33**
Hostingchecker.com	CANADA	7,067	87	0%	87	0%	91	0%	**88.33**
Iitg.ac.in	USA	8,229	183	15%	191	5%	182	20%	**185**

Here, ***iitg.ac.in*** shows packet loss greater than 0. The reason for packet loss here is due to network congestion. When content arrives for a sustained period at a given router or network segment at a rate greater than it is possible to send through, there is no other option than to drop packets. If we assume that the router that is responsible for the appearance of a particular pattern has a fixed-size buffer, then the larger the size of the probe packets the fewer the packets that can be queued in the buffer before the buffer fills up. Thus large packets more easily fill the buffer than small packets, and cause packet drop. Furthermore, large packets take longer to be received, queued, and transmitted than small packets. Though not mentioned in the table, ***geeksforgeeks.org*** is such an example where we can see **100% packet loss**. The ping packets are blocked by the user with firewall. However, another reason for such loss could be that one of the routers in between is not working or it has blocked the ping or has given less priority to the pings. Along a given route, RTT usually increases with increasing hop count. However, different routes may have quite different relationships between RTT and hops count. Physically short hops will contribute far less propagation delay than physically long hops. So, it depends on the number of routers or hops between the two systems. With geographical distance number of routers or hops may increase. But as we can see from the above table no relation can be found out between the geographical distance of the two hosts and round trip time of the packets between them. So, even though it is related with geographical distance it is very weak. Therefore, RTTs are weakly correlated with the geographical distance of the hosts. For the next part of the question where one of the above host has to be used to repeat the same experiment with different packet sizes at different times of the day ***linkedin.com*** was used. Here, each ping sends 20 packets for different packet sizes ranging from 64 bytes to 2048 bytes. This experiment has been performed 3 times in a single day. It was conducted at 1 a.m., 11 a.m. and 7 p.m.. The rtt for each packet size and their average is given in the table below :-

Packet size(bytes)	64	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	2048
Rtt @ 1 hours(ms)	97	97	99	97	98	97	97	97	97	98	97	97	97	97	97	104	104	106	105	104	104	106
Rtt @ 11 hours(ms)	116	117	118	116	117	118	116	117	116	116	118	116	116	117	117	116	117	117	117	116	117	117
Rtt @ 19 hours(ms)	108	104	100	106	98	107	107	107	102	102	104	104	102	98	103	111	108	109	110	112	111	111
Average(ms)	107	106	106	106	104	107	107	107	105	105	106	106	105	104	106	110	110	111	111	111	111	111



The graph shown between round trip time in ms and packet size in bytes is the average of all the round trip times at different times for each specific packet size. As can be seen from the graph, when the packet size reaches **1500 bytes**, the round trip time increases above certain point and stays in that region showing there is a difference between the sending of packet size which is smaller than 1500 bytes and the ones which are larger than that. The reason behind this is MTU(maximum transmission unit). The MTU here is 1500 bytes. So, whenever the packet size goes beyond this size, the packet is divided into 2 packets or more depending on the size of the packets which is needed to be sent. Therefore, as the number of packets increase the time increased. Since, there are other factors which affect the round trip time like congestion the time is not double when a packet is divided into 2 packets. However, we can infer from the graph that with the increase in packet size the round trip time will increase.



The graph shown between round trip time in ms and time of the day in 24 hour format is the average of the all the rtts with different packet sizes at one particular time of the day. This experiment has been performed 3 times in a single day and same has been plotted on the graph shown. As it can be seen the highest round trip time is during morning during 11 o'clock followed by midnight and then at night 8 o clock. The reason for high rtt or different rtt during different times of the day is due to the traffic or congestion. During peak times or rather say highest rtt times of the day, the traffic is the highest which contributes to the higher rtt due to congestion. As congestion increases the packets send are increased and so the packets in the queue are increased because of which delay is increased. Therefore, due to different amount of traffic during different time of the day, the rtt is different.

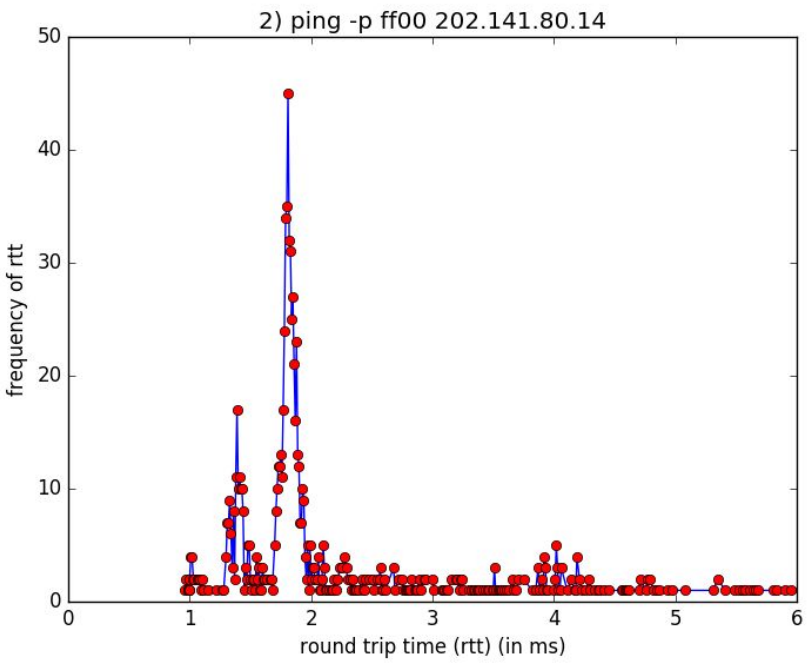
3) For this question **202.141.80.14** has been taken as an I.P. to perform the experiments mentioned in the question. To read and analyze the output came from **ping -n** and **ping -p ff00** for the mentioned I.P. address python script has been created and used to mathematical calculations as mentioned in b part of the question and also to plot graph as mentioned in c part of the question.

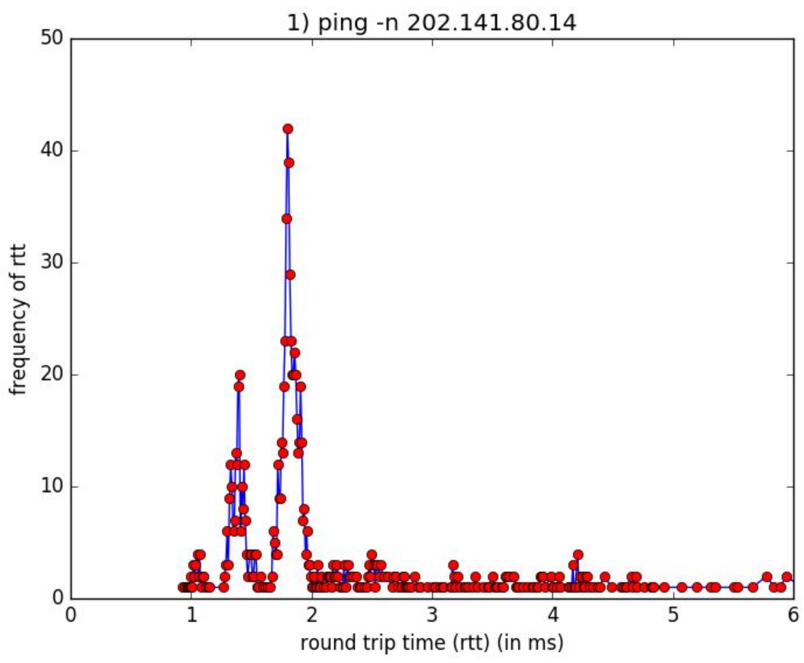
a) Packet loss for the command **ping -n 202.141.80.14** is **1 packet** while the packet loss for the command **ping -p ff00 202.141.80.14** are **2 packets**.

b) The given tables tells the minimum, maximum, mean and median of latency of pings of the two commands mentioned in the problem statement.

commands	Minimum	Maximum	Mean	Median
ping -n	0.93	208.0	3.53	1.82
ping -p ff00	0.96	204.0	4.07	1.84

c)

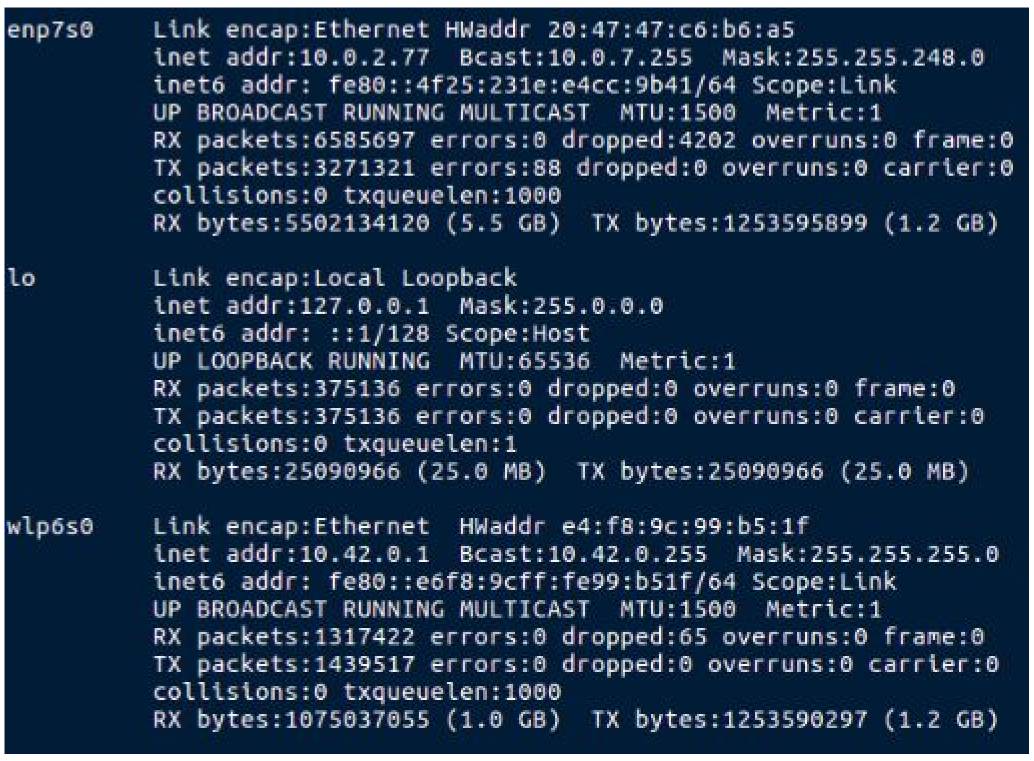




The above graphs are between the frequency of round trip time and that specific rtt. The first graph is for command 1 and second graph is for command 2 as mentioned in the question.As can be seen, from both the graph, both the graphs resemble the graph of normal distribution.

d) In **ping -n 202.141.80.14** only numeric output is given and no attempt is made to lookup symbolic names for host addresses while in **ping -p ff00** 16 pad bytes are filled out in the packet which is sent which is useful for diagnosing data-dependent problems in a network. So, because of this the average time taken by the second command is more than the average time taken by the first command.

4)



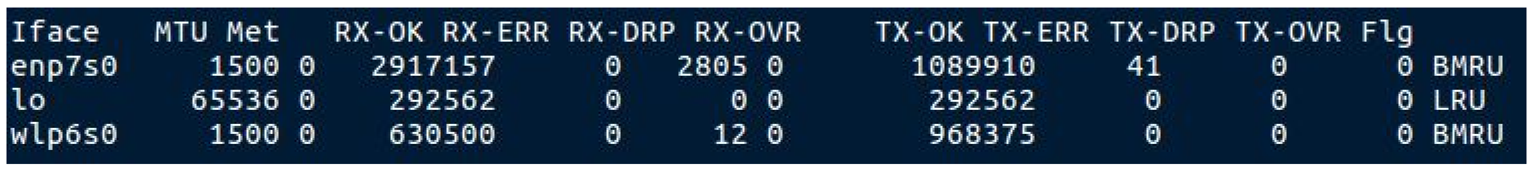
```
enp7s0    Link encap:Ethernet HWaddr 20:47:47:c6:b6:a5
          inet addr:10.0.2.77  Bcast:10.0.7.255  Mask:255.255.248.0
          inet6 addr: fe80::4f25:231e:e4cc:9b41/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6585697 errors:0 dropped:4202 overruns:0 frame:0
          TX packets:3271321 errors:88 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5502134120 (5.5 GB)  TX bytes:1253595899 (1.2 GB)

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:375136 errors:0 dropped:0 overruns:0 frame:0
          TX packets:375136 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:25090966 (25.0 MB)  TX bytes:25090966 (25.0 MB)

wlp6s0    Link encap:Ethernet  HWaddr e4:f8:9c:99:b5:1f
          inet addr:10.42.0.1  Bcast:10.42.0.255  Mask:255.255.255.0
          inet6 addr: fe80::e6f8:9cff:fe99:b51f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1317422 errors:0 dropped:65 overruns:0 frame:0
          TX packets:1439517 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1075037055 (1.0 GB)  TX bytes:1253590297 (1.2 GB)
```

The output of the **ifconfig** command gives the output as shown in the figure.
The command ifconfig shows details of the network interfaces that are up and running in your computer. **ifconfig** is used to configure, or view the configuration of, a network interface. ifconfig stands for "**interface configuration**". Here, enp7s0, lo and wlp6s0 are the names of the active network interfaces on the system. lo is the loopback interface. This is a special network interface that the system uses to communicate with itself.
Link encap:Ethernet denotes that the interface is an Ethernet related device. **HWaddr 20:47:47:c6:b6:a5** is the hardware address or MAC address which is unique to each Ethernet card which is manufactured. Usually, the first half part of this address will contain the manufacturer code which is common for all the Ethernet cards manufactured by the same manufacturer and the rest will denote the device Id which should not be the same for any two devices manufactured at the same place. **inet addr** indicates the machine IP address which is 10.0.2.77 in this case. **Bcast** denotes the broadcast address which is 10.0.7.255 in this case. **Mask** - is the network mask which we passed using the netmask option....A netmask is a 32-bit mask used to divide an IP address into subnets and specify the network's available hosts. In a netmask, two bits are always automatically assigned. For example, in 255.255.225.0, "0" is the assigned network address. In 255.255.255.255, "255" is the assigned broadcast address. The 0 and 255 are always assigned and cannot be used and in this case 255.255.248.0 is the mask. IPv6 addresses(**inet6 addr**) are shown because IPv6 is enabled on network interfaces by default. **Scope:Link** means they can only be used on a single directly attached network.**UP** is a flag which indicates that the kernel modules related to the Ethernet interface has been loaded. **BROADCAST** denotes that the Ethernet device supports broadcasting - a necessary characteristic to obtain IP address via DHCP. **RUNNING** tells that the interface is ready to accept data. **MULTICAST** indicates that the Ethernet interface supports multicasting. Multicasting can be best understood by relating to a radio station. Multiple devices can capture the same signal from the radio station but if and only if they tune to a particular frequency. Multicast allows a

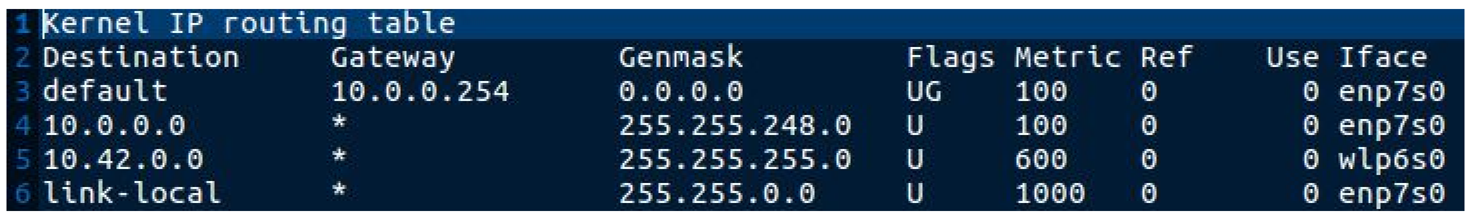
source to send a packet(s) to multiple machines as long as the machines are watching out for that packet. **MTU** short form for Maximum Transmission Unit is the maximum size of each packet received by the Ethernet card. The value of MTU for all Ethernet devices by default is set to 1500 bytes. Though it can changed. However setting this to a higher value could hazard packet fragmentation or buffer overflows. **Metric** option can take a value of 0,1,2,3... with the lower the value the more leverage it has. The value of this property decides the priority of the device. This parameter has significance only while routing packets. For example, if you have two Ethernet cards and you want to forcibly make your machine use one card over the other in sending the data. Then you can set the Metric value of the Ethernet card which you favor lower than that of the other Ethernet card. However in Linux, setting this value using ifconfig has no effect on the priority of the card being chosen as Linux uses the Metric value in its routing table to decide the priority. **RX Packets**, **TX Packets** - The next two lines show the total number of packets received and transmitted respectively. As it can be seen in the output, the total **errors** are 0 in each case, packets are **dropped** during two cases no **overruns** in any case. If the errors or dropped value greater than zero, then it could mean that the Ethernet device is failing or there is some congestion in your network. Frame and carrier also show the errors where frame errors tells the CRC failures on receipt of a frame and carrier tells the loss of link pulse. **collisions** should ideally be 0. If it has a value greater than 0, it could mean that the packets are colliding while traversing the network which is a sure sign of network congestion. **txqueuelen** denotes the length of the transmit queue of the device. It is usually set it to smaller values for slower devices with a high latency such as modem links and ISDN. **RX Bytes**, **TX Bytes** indicate the total amount of data that has passed through the Ethernet interface either way. As long as there is some network traffic being generated via the Ethernet device, both the RX and TX bytes will go on increasing. **ifconfig -a** displays all interfaces which are currently available, even if down while **ifconfig** only shows the interfaces which are up.



```
Iface    MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enp7s0     1500 0   2917157      0   2805 0      1089910     41      0      0 BMRU
lo        65536 0    292562      0      0 0       292562      0      0      0 LRU
wlp6s0     1500 0    630500      0     12 0       968375      0      0      0 BMRU
```

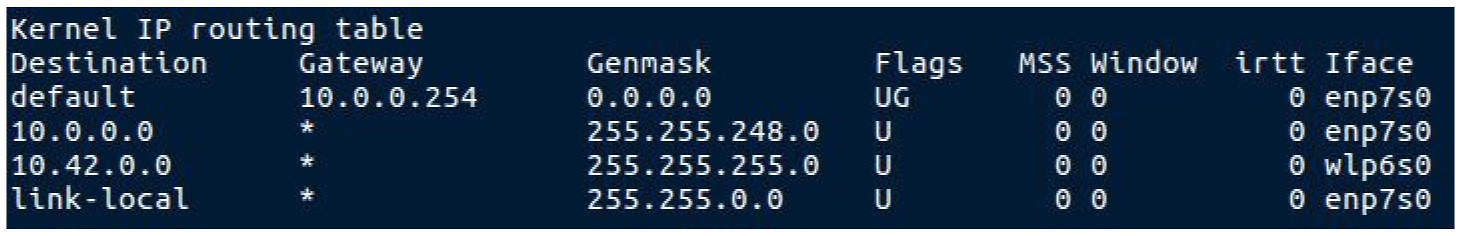
ifconfig -s displays a short list of ifconfig (**netstat -i** gives the same output as ifconfig -s). The output given is shown in the figure.

Here, Iface=interface, MTU - maximum transmission Unit, Met=Metric, RX-OK= packets which are successfully received, RX-ERR=RX error, RX-DRP=RX dropped, RX-OVR=RX overruns, TX-OK= packets which are successfully transmitted, TX-ERR=TX error, TX-DRP=TX dropped, TX-OVR=TX overruns, flg=flag:- The flag values printed by ifconfig correspond more or less to the names of its command line options
The meaning every character in flag is as follows:- B = A broadcast address has been set, L = This interface is a loopback device, M = Supports multicast, O = ARP is turned off for this interface, P = This is a point-to-point connection, R = Interface is running, U = Interface is up.



```
1 Kernel IP routing table
2 Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
3 default         10.0.0.254      0.0.0.0         UG    100    0        0 enp7s0
4 10.0.0.0        *               255.255.248.0   U     100    0        0 enp7s0
5 10.42.0.0       *               255.255.255.0   U     600    0        0 wlp6s0
6 link-local      *               255.255.0.0     U     1000   0        0 enp7s0
```

The output of the **route** commands gives the output as shown in the figure below.
Line 3 says that any packet to a destination without another route will be sent out enp7s0, using 10.0.0.254 as a gateway. Line 4 This says that any packet with a destination of 10.0.0.0 through 10.0.0.255 will be sent out enp7s0 with using a gateway 10.0.0.254. Line 5 This says that any packet with a destination of 10.42.0.0 through 10.42.0.255 will be sent out wlp6s0s0 without using a gateway. Line 6 This says that any packet with a link-local address will be sent out interface enp7s0 with no gateway. **route -A family** uses the specified address family (eg `inet) i.e. **route -A inet. route -F** operates on the kernel's FIB (Forwarding Information Base) routing table which is the default. **route -C** operates on the kernel's routing cache. **Route -v** selects verbose operation. **route -n** shows numerical addresses instead of trying to determine symbolic host names. **route -e** uses netstat-format for displaying the routing table. **route -ee** will generate a very long line with all parameters from the routing table. The output of **route -e** is given below.



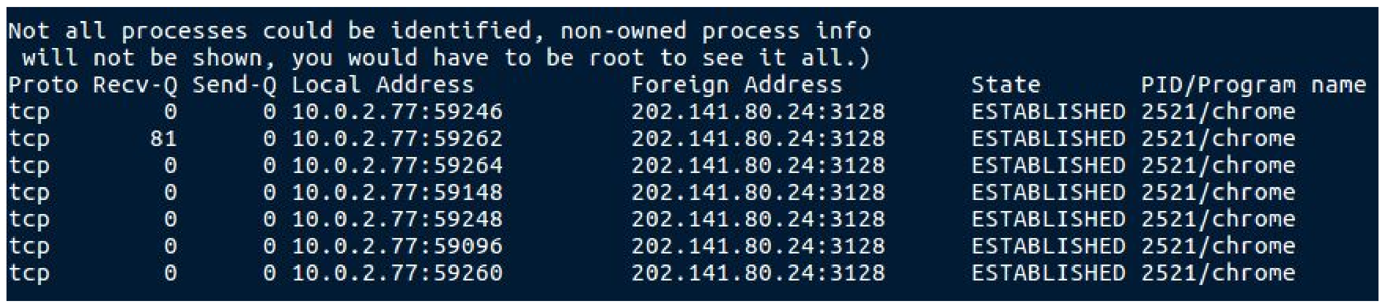
```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.0.254      0.0.0.0         UG        0 0          0 enp7s0
10.0.0.0        *               255.255.248.0   U         0 0          0 enp7s0
10.42.0.0       *               255.255.255.0   U         0 0          0 wlp6s0
link-local      *               255.255.0.0     U         0 0          0 enp7s0
```

Destination is the address of the network that the packet is headed to. The "default" means that if the destination is not found in any of the other rules than use this rule. **Gateway** is the Next hop. This is where the packet will be sent if the destination is not on the same network as the sender. If the value is in "*" or "On-link" or the address of the current device then these all mean the same thing. It means that the packet is addressed to a device that is directly reachable by the current host. In other words, they're on the same network so the gateway won't actually be used because the host will know the data link layer (MAC) address of the destination and be able to send it directly there. **Irtt** is the initial round trip time. **Genmask** is the subnet mask. If there is more than one address in the routing table that works for the outgoing packet, the rule with the higher subnet mask will be used. If those are tied, then metric comes into play but that will be different based on what protocol is being used. **Iface** or Interface is the On-board connection. For example, the laptop I am on currently has three interfaces: Ethernet card, Wireless card, Bluetooth. **Metric** option can take a value of 0,1,2,3... with lower the value the more leverage it has. The value of this property decides the priority of the device. This parameter has significance only while routing packets. For example, if you have two Ethernet cards and you want to forcibly make your machine use one card over the other in sending the data. Then you can set the Metric value of the Ethernet card which you favor lower than that of the other Ethernet card. The **flag 'U'** indicates that this entry is up while the **flag 'G'** indicates that this entry is not a direct entry i.e. the destination indicated in this route entry is not on the same network. The meaning of each flag is as follows. **A** means receive all multicast at this interface. **B** means OK broadcast. **D means** debugging ON. **M means** promiscuous Mode. **O** means no ARP at this interface. **P** means P2P connection at this interface. **R** means interface is running. **U** means Interface is up. **G** means not a direct entry. The **MSS** column tells us what the path MTU discovery has determined for a maximum segment size for the route to this destination. The **Ref** column of netstat's output shows the number of references to this route, that is, how many other routes (e.g. through gateways) rely on the presence of this route. **window** is the default window size for TCP connections over this route. **Use** is the count of lookups for the route and depending on the use of -F and -C this will be either route cache misses (-F) or hits (-C).

5)

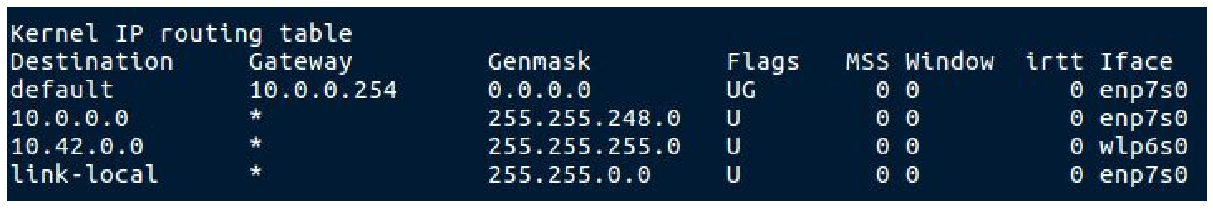
Netstat (network statistics) is a command-line network utility tool that displays network connections for the Transmission Control Protocol (both incoming and outgoing), routing tables, and a number of network interface (network interface controller or software-defined network interface) and network protocol statistics.
It is **used** for finding problems in the network and to determine the amount of traffic on the network as a performance measurement. Netstat provides information and statistics about protocols in use and current TCP/IP network connections.

netstat -atnp | grep ESTA command is used to show all the TCP connections established.



```
Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 10.0.2.77:59246         202.141.80.24:3128      ESTABLISHED 2521/chrome
tcp       81      0 10.0.2.77:59262         202.141.80.24:3128      ESTABLISHED 2521/chrome
tcp        0      0 10.0.2.77:59264         202.141.80.24:3128      ESTABLISHED 2521/chrome
tcp        0      0 10.0.2.77:59148         202.141.80.24:3128      ESTABLISHED 2521/chrome
tcp        0      0 10.0.2.77:59248         202.141.80.24:3128      ESTABLISHED 2521/chrome
tcp        0      0 10.0.2.77:59096         202.141.80.24:3128      ESTABLISHED 2521/chrome
tcp        0      0 10.0.2.77:59260         202.141.80.24:3128      ESTABLISHED 2521/chrome
```

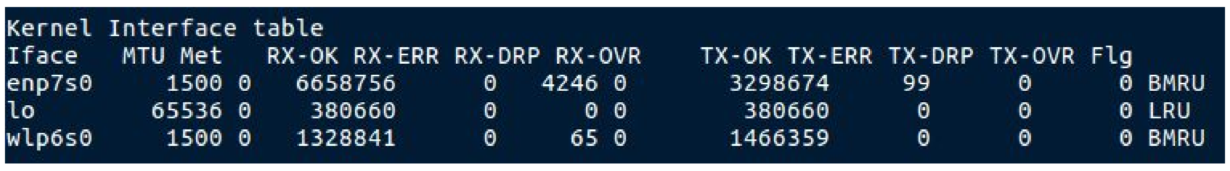
The **Proto** column tells us if the socket listed is TCP or UDP. Those are network protocols. TCP makes reliable connections but slows down dramatically if the network quality is bad. UDP stays fast but may lose a few packets or deliver them in the wrong order. The **Recv-Q** and **Send-Q** columns tell us how much data is in the queue for that socket, waiting to be read (Recv-Q) or sent (Send-Q). In short: if this is 0, everything's ok, if there are non-zero values anywhere, there may be trouble. If you look closely at the example, you'll see that one socket has a Recv-Q with 81 unread bytes in them. The **Local Address** and **Foreign Address** columns tell to which hosts and ports the listed sockets are connected. The local end is always on the computer on which you're running netstat and the foreign end is about the other computer (could be somewhere in the local network or somewhere on the internet). The **State** column tells in which state the listed sockets are. The TCP protocol defines states, including **LISTEN** (wait for some external computer to contact us) and **ESTABLISHED** (ready for communication). The stranger among these is the **CLOSE WAIT** state. This means that the foreign or remote machine has already closed the connection, but that the local program somehow hasn't followed suit. The **PID/Program name** column tells us which pid owns the listed socket and the name of the program running in the process with that pid. So we can see which programs are using the network and to whom they are connecting.



```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.0.254      0.0.0.0         UG        0 0          0 enp7s0
10.0.0.0        *               255.255.248.0   U         0 0          0 enp7s0
10.42.0.0       *               255.255.255.0   U         0 0          0 wlp6s0
link-local      *               255.255.0.0     U         0 0          0 enp7s0
```

netstat -r :-Displays the contents of the IP routing table. For the detail explanation of the output of netstat -r as shown in the figure refer the explanation of the output of the command route -e from question number 4.

netstat -i is used to display network interface status



```
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enp7s0     1500 0   6658756      0   4246 0       3298674     99      0      0 BMRU
lo        65536 0    380660      0      0 0        380660      0      0      0 LRU
wlp6s0     1500 0   1328841      0     65 0       1466359      0      0      0 BMRU
```

From the output of **netstat -i** shown above, we can conclude that the number of interfaces on the machine is 3 namely enp7s0, lo, wlp6s0.

Loopback interface can perform the following functions: **Device identification**—The loopback interface is used to identify the device. While any interface address can be used to determine if the device is online, the loopback address is the preferred method. Whereas interfaces might be removed or addresses changed based on network topology changes, the loopback address never changes. **Routing information**—The loopback address is used by protocols to determine protocol-specific properties for the device or network. **Packet filtering**—Stateless firewall filters can be applied to the loopback address to filter packets originating from, or destined for, the Routing Engine. We can see the loopback in the kernel interface table which is the output of **netstat -i** command and also in the output of ifconfig command.

6) i) For traceroute experiment, same hosts as chosen in the question 2 are used. Here, traceroute has been performed in those hosts at 3 different times of the days. http://ping.eu tool is used to perform traceroute experiment. The table shown below tells the number of hops at different times of the day which are 4 p.m., 8:50 p.m. and 10:50 a.m..

hosts	**instagram.com**	**linkedin.com**	**internshala.com**	**hostingchecker.com**	**iitg.ac.in**
Hops @ 16 hours	15	11	9	13	14
Hops @ 20:50 hours	15	11	9	13	14
Hops @ 10:50 hours	15	11	9	13	14

The common hops between different routes are as follows:-

Hop number	**I.P. address**	**hosts**
2	213.239.245.237	hostingchecker.com, iitg.ac.in, internshala.com, linkedin.com, instagram.com
	213.239.245.241	instagram.com, linkedin,iitg.ac.in
3	213.239.245.218	hostingchecker.com, iitg.ac.in, linkedin.com
	213.239.245.177	linkedin.com, internshala.com
	213.239.224.9	instagram.com, iitg.ac.in
	213.239.224.13	Instagram.com, iitg.ac.in

	213.239.245.221	Instagram.com, linkedin.com
4	213.239.245.10	internshala.com, iitg.ac.in, linkedin.com
5	81.95.15.5	linkedin.com, iitg.ac.in
	80.255.15.121	linkedin.com, iitg.ac.in

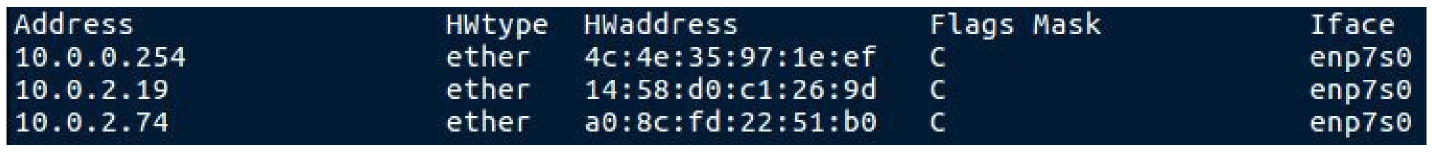
Here, every line means for e.g. first line says that at hop number 2 of all the hosts mentioned there, the I.P. addresses as mentioned is the common route they have with the other hosts which are mentioned with them in the row.

ii) Yes, the routes to different host changes at different times of the day. For e.g. instagram.com at 4:00 p.m. at hop number 8 has route 62.115.122.61 but at 8:50 p.m. it is 80.91.246.68. This is because the destination router has two or more connections to different ISPs and the connections are doing **load balancing**. 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. A traceroute to a router, it will use *process switching* for every package, that is influenced by load balancing and we have **every time a different path**.

iii) Traceroute is not able to find the complete paths to hosts in most of the cases like say linkedin.com, the reason behind this is because of firewall. The packets are sent through multiple routers and after doing so it is stopped by firewall and no packets can further go. Since packets are not able to go further beyond this firewall, the route to the destination I.P. can't be found. So, we will only get route till the firewall and not beyond that.

iv) Yes, it is possible to find the routes to certain hosts which fail to respond with ping experiment. The reason behind this is many times the router has blocked the ping or has given less priority to the pings. So, if less priority is given then most of the packets which are sent through pings are lost and if it is blocked all the packets which are sent through pings are loss. However, the routers have given less priority or blocked only the ping packets. However, the packets which are sent via traceroute aren't blocked. So, even though the site failed to respond to the ping experiment we were able to find the route to that host.

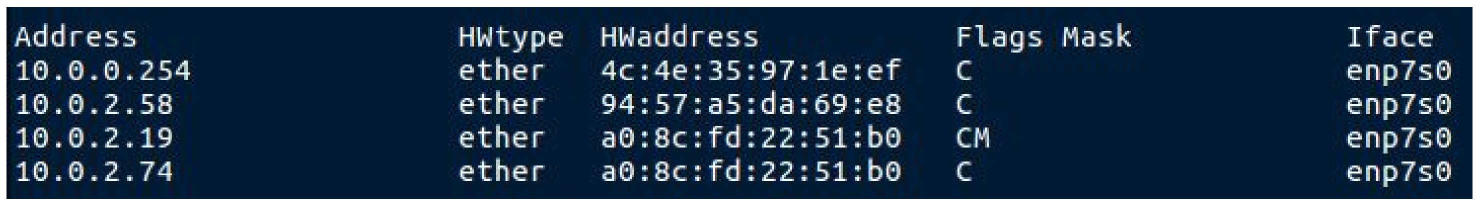
7)



```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     enp7s0
10.0.2.19                ether   14:58:d0:c1:26:9d   C                     enp7s0
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     enp7s0
```

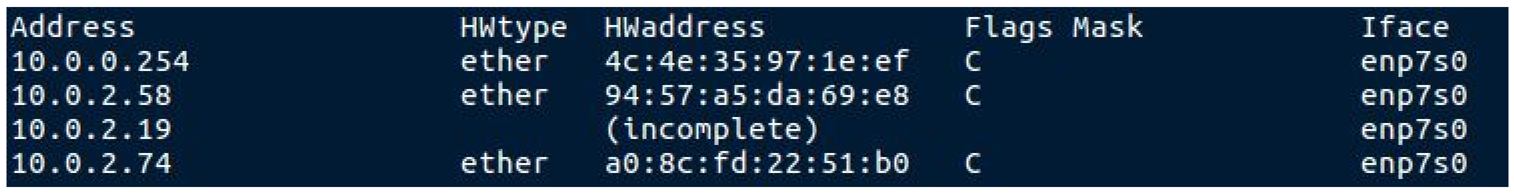
To see the entire ARP table for the machine **arp** command is used. The figure shown is the output of the command **arp.**

Address Resolution Protocol (ARP) is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network. For example, in IP Version 4, the most common level of IP in use today, an address is 32 bits long. In an Ethernet local area network, however, addresses for attached devices are 48 bits long. (The physical machine address is also known as a Media Access Control or MAC address). A table, usually called the **ARP cache**, is used to maintain a correlation between each MAC address and its corresponding IP address. ARP provides the protocol rules for making this correlation and providing address conversion in both directions.Address tells the IP host of the address with which some exchange of packets like pinging or transfer of data has happened. **HWtype** tells the hardware type of the host IP. The type is usually ether (Ethernet), which is the default. **HWaddress** tells the MAC address of the IP address. **incomplete** means that device has sent an ARP for the IP in the list (71.169.191.209) and it has not received a reply. ARP cache entries may be marked with the following flags: **C** (complete), **M** (permanent), and **P** (publish). The **flags** indicate if the mac address has been learned, manually set, published (announced by another node than the requested) or is incomplete. **Mask** also called a **subnet mask** because it is used to identify network address of an IP address. **Iface** tells the network associated with that i.p. Address.



```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     enp7s0
10.0.2.58                ether   94:57:a5:da:69:e8   C                     enp7s0
10.0.2.19                ether   a0:8c:fd:22:51:b0   CM                    enp7s0
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     enp7s0
```

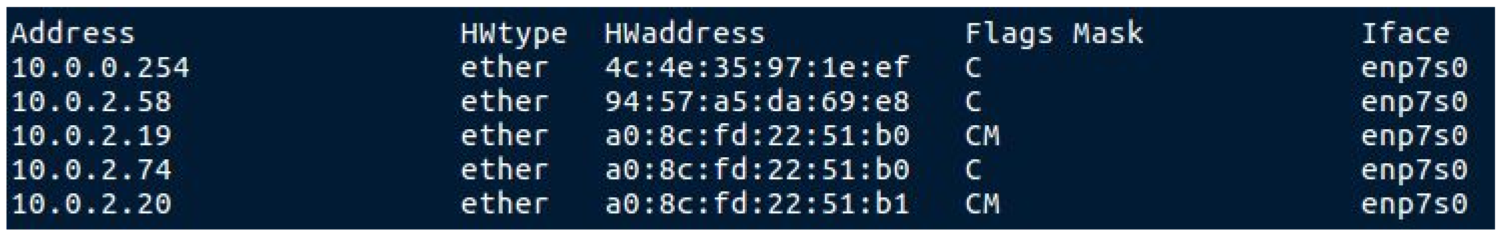
Now, if we try to add an entry in the arp table with the command **sudo arp -s 10.0.2.19 a0:8c:fd:22:51:b0.** Then the arp table will look like this. Here, we will get another entry with IP 10.0.2.19 and the mac address which we inserted into. M flag is used to show that the flag entry is marked permanent. However, without sudo that is non super user cannot add entry into the arp cache.



```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     enp7s0
10.0.2.58                ether   94:57:a5:da:69:e8   C                     enp7s0
10.0.2.19                        (incomplete)                              enp7s0
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     enp7s0
```

Now, if we try to delete the entry, then we will see that flags are removed from the arp cache and the MAC address is removed and only incomplete is shown in the HWaddress. However, IP address is still there in the table. This shows that we have an IP address stored in arp cache but we don't have information about the MAC address corresponding to that IP.

We can add entry in the arp table using the command **arp -s [IP_address] [hw_address]** without square brackets. Where hw_address is the MAC address corresponding to the IP address which is put into the IP_address. To delete entries in the arp table simply use **arp -d [IP_address]**. The MAC address and the flag corresponding to the IP address which is put into the command will be removed. However, the IP address will stay with incomplete status shown in HWaddress. To edit the entries again use **arp -s [IP_address] [hw_address]** command. If the IP address which is put into the command is already there then the corresponding MAC address will be updated(whether there is already a MAC address or it is just showing incomplete). If there is no such entry then a new entry will be made as explained already.

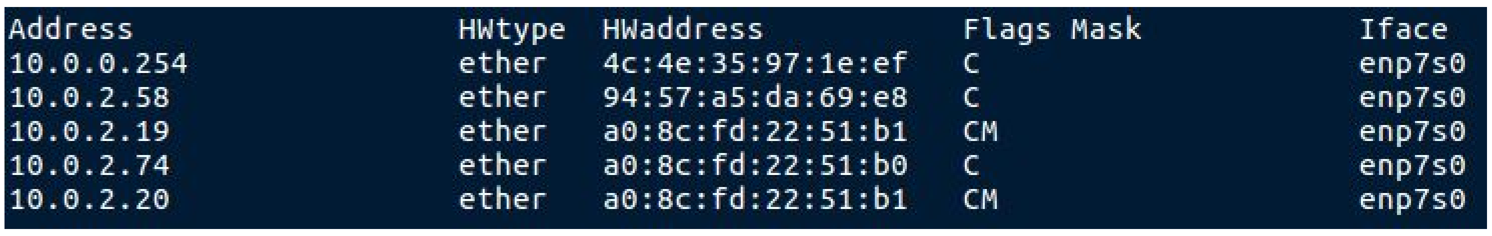


```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     enp7s0
10.0.2.58                ether   94:57:a5:da:69:e8   C                     enp7s0
10.0.2.19                ether   a0:8c:fd:22:51:b0   CM                    enp7s0
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     enp7s0
10.0.2.20                ether   a0:8c:fd:22:51:b1   CM                    enp7s0
```

If we use **sudo arp -s 10.0.2.19 a0:8c:fd:22:51:b0** and **sudo arp -s 10.0.2.20 a0:8c:fd:22:51:b1** commands to add to new entries the resulting output will look like as shown in the figure. The two entries which are added in the table have the IP address as **10.0.2.19** and **10.0.2.20.**

The entries should be there in the arp table for 60 seconds. To do the trial and error method to find out how long the entry stays there in the arp table we can do the binary search type procedure in it that is by taking 100 seconds to find out if the entry is gone or not if yes then take 50 seconds if yes then take 25 else take 75 seconds to find out. Doing this procedure we will finally know that at time 60 seconds the entry in the arp table has been gone.

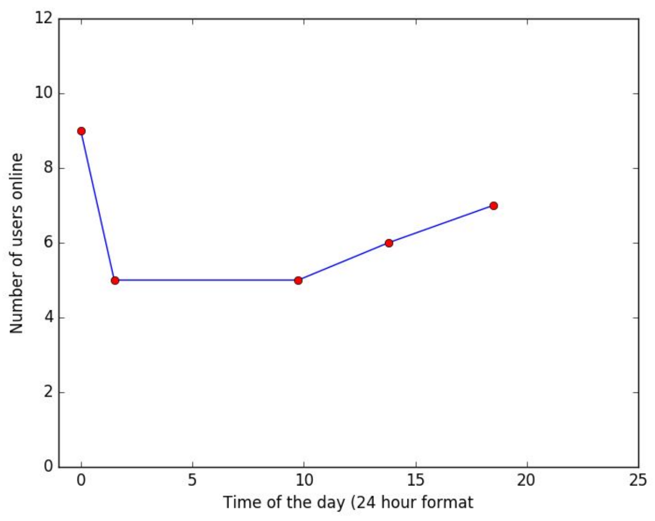
If we add two IP address(10.0.2.19 and 10.0.2.20) with same MAC address then the IP addresses will be added as shown in the figure below. Now, one IP address(10.0.2.19) has correct MAC address associated with the arp cache and the other IP address(10.0.2.20) has the wrong MAC address associated with it. So, if we try to ping the IP address 10.0.2.19 we get the responses from it. However, if we try to ping the IP address 10.0.2.20 then we won't get the response from it unless we put the correct MAC address or it retrieves the correct IP address and put it in the arp cache. The reason behind it is although we are giving the correct IP address to ping. It will retrieve the MAC address corresponding to that IP address then it will search that MAC address. After finding that MAC address it will match the IP address which is wrong so it will drop the packets. This is for the case when the subnet range is local. However, when the external subnet range is used at that time first it will search the IP address of the router and the packet is sent there and then the MAC address is searched for it. So, even if the MAC address and IP address does not match in this case the packets will go that is ping will work in this case.


```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     enp7s0
10.0.2.58                ether   94:57:a5:da:69:e8   C                     enp7s0
10.0.2.19                ether   a0:8c:fd:22:51:b1   CM                    enp7s0
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     enp7s0
10.0.2.20                ether   a0:8c:fd:22:51:b1   CM                    enp7s0
```

8)

Subnet range used here is 10.0.2.77/26. So the command used for this experiment becomes **nmap -sP -n 10.0.2.77/26.** The command is run 5 times during the same day to check the number of users online. The time during which this experiment has been performed is at 12:00 a.m.,1:30 a.m., 9:45 a.m., 1:50 p.m. and 6:30 p.m. for which the number of systems online are shown in the table given below.

Time(24 hour format)	0	1.5	9.75	13.8	18.5
Number of users online	9	5	5	6	7



The graph shown below is between the number of users online and the time at which this has been counted. Time in the graph is in 24 hour format. The number of IP addresses scanned upon using this command is 64. As can be seen from the graph from the midnight the number of users online are decreasing till 10 a.m. and then increasing to reach the peak value again at midnight. The maximum number of systems are online during midnight and the minimum is from night 1 a.m. to morning 10 a.m.