CS349 NETWORKS LAB – ASSIGNMENT 1

Roopansh Bansal (150101053)

1 PING COMMAND – BASICS

- The option required to specify the number of echo requests to send with ping is **'c'.**
- The time interval between two successive ping requests can be set using **'i'** option (time in seconds).
- We can send packets to the destination one after another without waiting for a reply by using the **'–l'** option. The normal users can only send 3 packets using this. Alternatively, The destination can be flooded with ping requests without waiting using **'-f'** option. Also time interval can be set 0 seconds using **'i'**. Normal users have a limit of **200ms.**
- The packet size can be set with the **'s'** option (size in bytes). The actual packet size will be slightly larger than what we enter due to the addition of the ICMP header(8 Bytes) and IP Headers(20 Bytes). Hence, total packet size will be **92 bytes** if size is set as **64 bytes.**

2 PING EXPERIMENT

- The readings are taken at **1.00pm**, **7.00pm** and **1.00am (IST)** respectively.
- **Test PC was connected to DIGITALOCEAN VPN (Bangalore,India) while performing the experiment.**
- **'justpakit.com'** is chosen for experimenting with **packets of size from 64 bytes to 2048 bytes**.

DESTINATION HOST ADDRESS	IP ADDRESS	GEOGRAPHIC LOCATION	Avg. RTT 1 (ms)	Avg. RTT 2 (ms)	Avg. RTT 3 (ms)	Total Avg. RTT (ms)
digitalocean.com	104.16.109.208	Arizona, United States	76.061	91.875	85.680	84.539
iitg.ac.in	14.139.196.22	Guwahati, India	114.658	127.167	121.000	120.942
Justpakit.com	139.59.87.253	Bangalore, India	61.091	70.817	60.193	64.034
Seedr.cc	95.211.185.133	Amsterdam, Netherlands	331.955	358.695	247.127	312.592
Youtube.com	172.217.31.206	California, United States	68.653	90.701	75.473	78.276

Table 1: Round Trip Times of 5 Hosts

Size (Bytes)	64	256	512	768	1024	1280	1536	1792	2048
Avg. RTT 1 (ms)	62.097	56.879	61.477	56.938	58.868	61.406	111.777	112.239	116.622
Avg. RTT 2 (ms)	83.886	70.155	84.152	79.518	79.684	74.881	166.073	131.965	141.431
Avg. RTT 3 (ms)	64.798	59.951	60.219	60.438	65.219	62.550	132.313	120.825	119.195

Table 2: Round Trip Time vs Packet Size

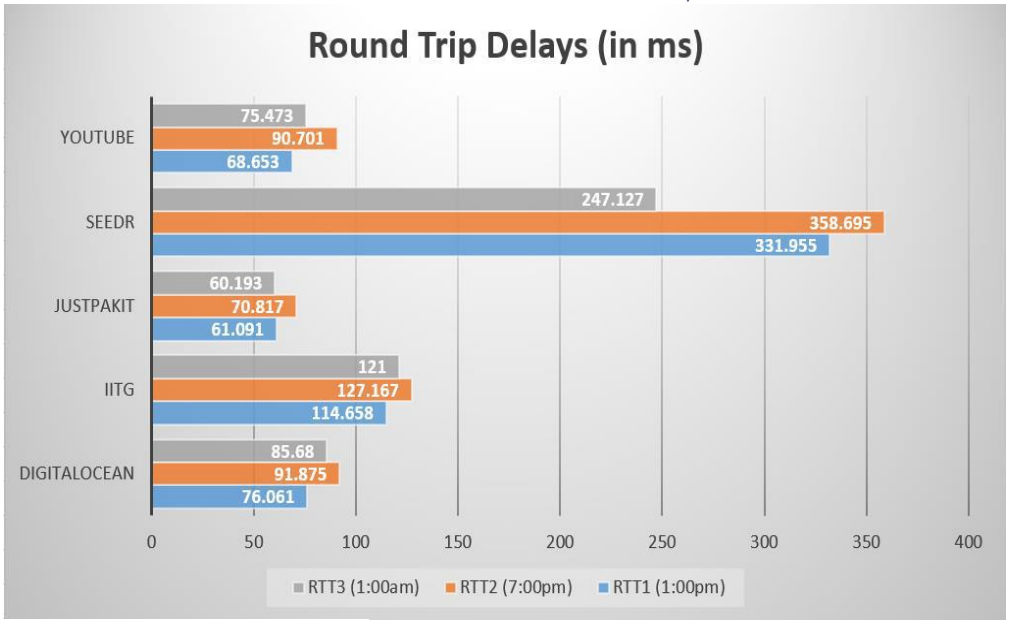


Figure 1: Round Trip Time of 5 Hosts

RTT V/S DISTANCE : From the above table, we can conclude that there exists ***weekly positive*** correlation between the Geographic distance and Round Trip Time (RTT). They are correlated because of reasons like increased number of hops and increased propagation delay. Larger the distance, longer it takes for a packets to propogate(*Propagation delay*). Also, the packets have to go through more number of nodes and at each nodes there may be a delay(*processing delay*). Hence, more the routers, the longer is the RTT. It's a week relation because there are many other factors on which it depends like network traffic and the server capacities. **PACKET LOSS :** In my experiment, no cases of packet loss greater than 0% was found. But, in general packet loss can be greater than 0% because of network congestion and traffic. Some packets may collide with other packets in the network and result

in packet loss. The ICMP Packets have lower priority. So they might take longer time to process in some destination server's queue. Sometimes, there may be 100% packet loss. In such cases, generally the destination server drops all the ping ICMP packets. **RTT V/S DAY OF TIME:** From both the above tables, we observe that RTT's vary with time of the day. At different times, the congestion in network is different. From the observations, we can say that RTT is least at 1:00 am and maximum at 7:00pm. Around 1:00pm, RTT is little less than that at 7:00pm. Hence, we can conclude that the network traffic is high around 7:00pm.

RTT V/S PACKET SIZE : From the above table, we can clearly observe that the Round Trip Time(RTT) is almost the same for packets upto size 1280 Bytes. After that, we observe a sudden jump in the RTT (almost Doubles). This can be explained by the fact that Maximum Transmission Unit (MTU)is 1500 Bytes by default. If the packet size is less than 1500 Bytes, then the data is padded to make the size 1500 Bytes. Hence, for packets with size less than 1500 Bytes, the RTT is same. If packet size is more than 1500 Bytes, then the packet is broken into two frames of size 1500Bytes. Hence we observe an almost twice the RTT for packets of size 1536 bytes to 2048 bytes.

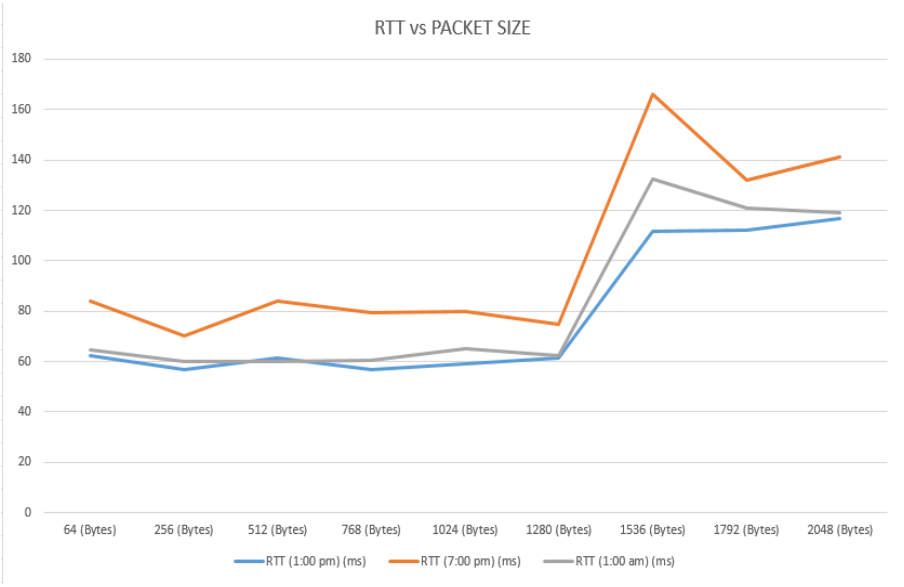


Figure 2: Round Trip Time VS Packet Size

3 2 DIFFERENT SCENARIOS OF PING

Command	Packets Sent	Packets Received	Packet Loss Rate	Minimum Latency	Max Latency	Mean Latency	Median Latency
$ ping –n –c 1000 ***172.16.112.1***	1000	1000	0.0%	1.552ms	18.984	1.988	1.68ms
$ ping –p ff00 –c 1000 ***172.16.112.1***	1000	998	0.2%	1.546	14.484	2.087	1.71ms

Table 3: Question 3 Readings

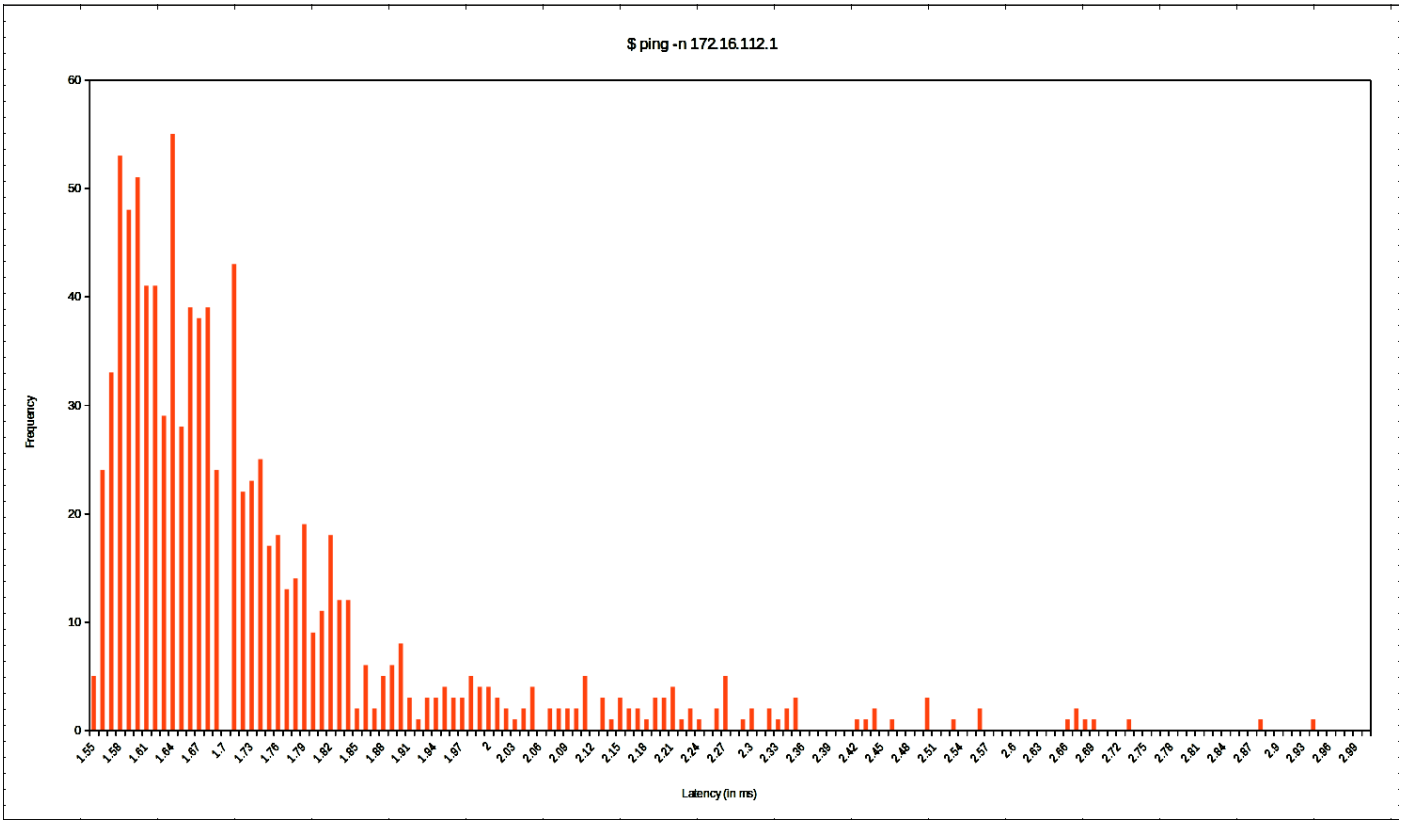


Figure 3: Histogram - $ ping -n

On plotting the curves as histograms, we observe that they resemble the shape of **Normal Distritbution. Note that a Log-Normal Distibution Curve would have better depicted the latencies.** The two cases are very similar to each other except in two aspects.

Firstly, no attempt will be made to lookup symbolic names for host addresses when using *'-n'*, hence it will be faster. So, the mean Latency is higher in second case than the mean latency in the first case. Secondly, *'-p ff00'* will cause the sent packet to be filled with the pattern *11111100000000* which is useful for diagnosing data-dependent problems in a network. This will cause problems with the synchronisation of the clocks because only one transition is present in the padding, from 1 to 0. Hence, the clocks are more likely to go out of synchronisation in second case and we observe that the packet loss is higher in the second case.

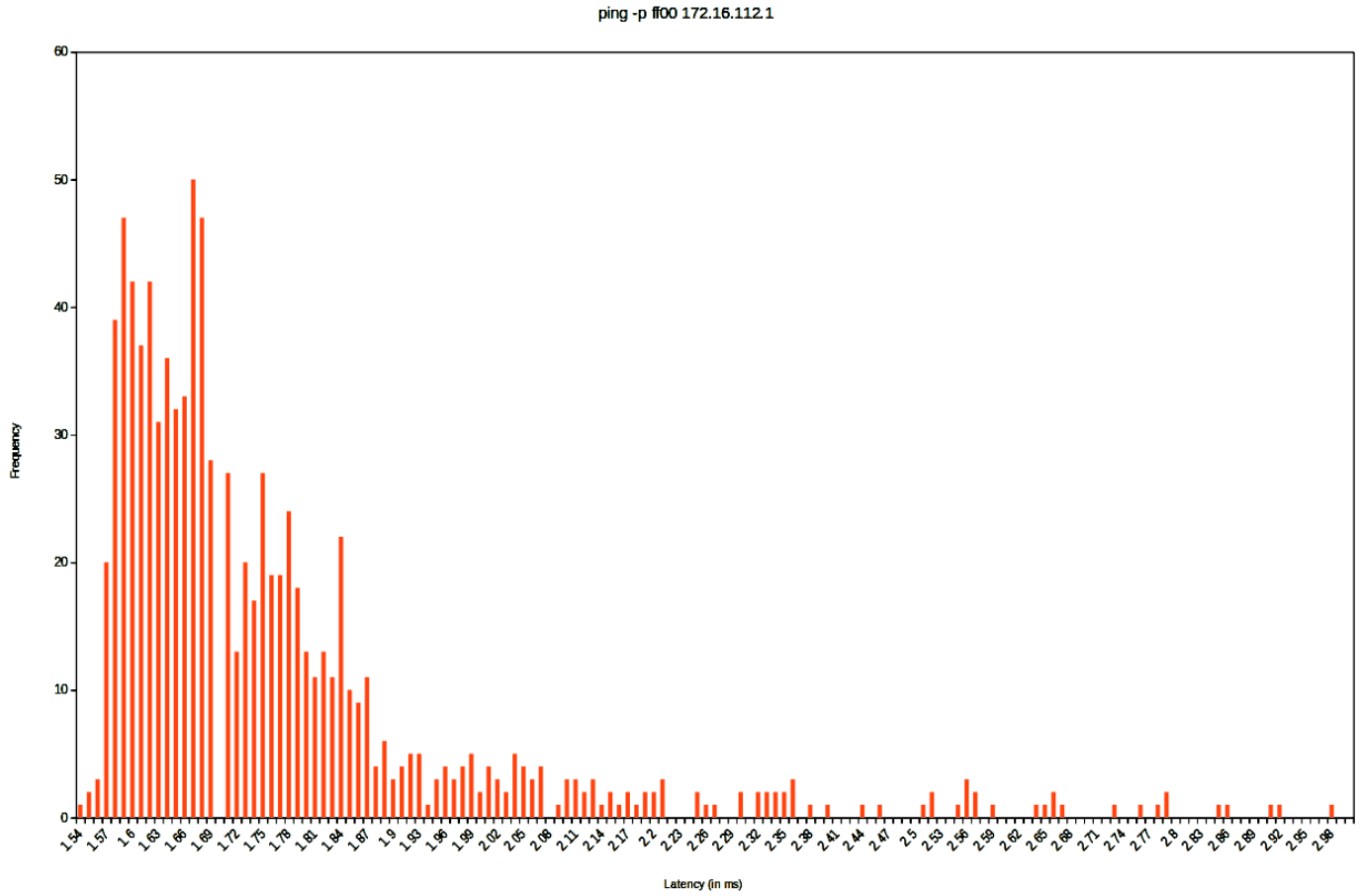


Figure 4: Histogram - $ ping -p ff00

4 IFCONFIG & ROUTE

The command ***'ifconfig'*** shows details of the network interfaces that are up and running in the computer. In my machine, I get the output as show in **Error! Reference source not found.**. My machine has a wired ethernet interface *(eno1)*, a loopback interface *(lo)* and a wireless ethernet interface *(wlo1)*. *Link encap - Ethernet* denotes that the interface is an Ethernet related device and *Loopback* indicates that interface is local loopback related. *Hwaddr* is the Hardware Address or the MAC address. *inet addr & inet6 addr* indicates the machine IPv4 and IPv6 address associated with that network interface respectively. *Bcast* denotes the broadcast address (The address required to broadcast on the network connected through that interface of the machine). *Mask* is the network mask which decides the potential size of the network. *Scope* is the scope of the area where the address is valid. Link means valid only on this device and host means valid only inside the host(my machine). *UP* flag indicates that the kernel modules related to the Ethernet interface has been loaded. *BROADCAST & MULTICAST* denotes that the Ethernet device supports broadcasting and multicasting respectively. *RUNNING* means that the interface is ready to accept data. *MTU*(Maximum Transmission Unit) is the size of each packet received by the Ethernet card. *Metric* take a value of 0,1,2… It decides the priority of the device. Lower the value the more leverage it has. *RX Packets, TX Packets* show the total number of packets received and transmitted respectively. It also shows the number of packets dropped and the overruns. *Collision* shows the number of packets that are colliding while traversing the

```
roopansh@roopansh-HP:~$ ifconfig -a -v
eno1      Link encap:Ethernet  HWaddr 3c:a8:2a:a5:66:73
          inet addr:10.0.2.84  Bcast:10.0.3.255  Mask:255.255.252.0
          inet6 addr: fe80::3ea8:2aff:fea5:6673/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27067 errors:0 dropped:35 overruns:0 frame:0
          TX packets:77068 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2185392 (2.1 MB)  TX bytes:5373362 (5.3 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:11434 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11434 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:915040 (915.0 KB)  TX bytes:915040 (915.0 KB)

wlo1      Link encap:Ethernet  HWaddr c4:8e:8f:a8:96:d5
          inet addr:192.168.31.136  Bcast:192.168.31.255  Mask:255.255.255.0
          inet6 addr: fe80::67d5:3c5b:a4db:17aa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:763 (763.0 B)  TX bytes:8900 (8.9 KB)
```

Figure 5: ifconfig

network due to network congestion. *RX bytes, TX bytes* indicates the total amount of data that has passed through the interface either way. *Txqueuelen* denotes the length of the transmit queue of the device.

```
roopansh@roopansh-HP:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.0.254      0.0.0.0         UG    100    0        0 eno1
0.0.0.0         192.168.31.1    0.0.0.0         UG    600    0        0 wlo1
10.0.0.0        0.0.0.0         255.255.252.0   U     100    0        0 eno1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eno1
192.168.31.0    0.0.0.0         255.255.255.0   U     600    0        0 wlo1
```

Figure 6: route

The ***'route'*** command shows the routing table of the device. **Error! Reference source not found.** shows the routing table of my machine. The *Destination* column identifies the destination network or destination host. The *Gateway* column identifies the defined gateway for the specified network. An asterisk (*) appears in this column if no forwarding gateway is needed for the network. The *Genmask* column shows the netmask for the network. The *Iface* column shows the network interface. *'eno1'* is for the Ethernet device and *'wlo1'* is for the Wireless Ethernet device. Under the Flags section, the *U flag* means the route is up, and the *G flag* means that specified gateway should be used for this route. *Metric* is the distance to the target (usually counted in hops). *Ref* is the number of references to this route. Route command has many options – *'-n'* is used to display the numerical IP address, list the kernel's routing cache information by using *'-C', '-v'* to select verbose operation, *'del'* to delete a route and *'add'* to add a route, *'-net'* specifies that the target is a network and *'-host'* specifies that the target is a host.

5 NETSTAT

'netstat' (network statistics) is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc. It is one of the most basic network service debugging tools, which tells us which ports are open and whether any programs are listening on ports.

$ netstat –at # All the TCP Connections

```
roopansh@roopansh-HP:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:mysql         *:*                     LISTEN
tcp        0      0 roopansh-HP:domain      *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 localhost:postgresql    *:*                     LISTEN
tcp        0      0 10.0.2.84:36000         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36006         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35980         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35908         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35990         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36002         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36004         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36046         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35970         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36040         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36074         202.141.80.24:3128      TIME_WAIT
tcp        0      0 10.0.2.84:36008         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35988         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35998         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36086         202.141.80.24:3128      TIME_WAIT
tcp        0      0 10.0.2.84:36058         202.141.80.24:3128      ESTABLISHED
tcp      390      0 10.0.2.84:36056         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36026         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36028         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36062         202.141.80.24:3128      TIME_WAIT
tcp        0      0 10.0.2.84:36064         202.141.80.24:3128      TIME_WAIT
tcp        0      0 10.0.2.84:35994         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:35996         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36042         202.141.80.24:3128      ESTABLISHED
tcp        0      0 10.0.2.84:36042         202.141.80.24:3128      ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
```

Figure 7: All TCP Connections

The *Proto* column tell us if the socket listed is TCP or UDP (Protocol). The *Recv-Q* and *Send-Q* columns tell us how much data is in the queue for that socket, waiting to be read or sent. The *Local Address* and *Foreign Address* columns tell to which hosts and ports the listed sockets are connected. The local end is my machine on which we run netstat, and the foreign end is the other computer (could be somewhere in the local network or on the internet). The *state* of the socket can be *listen* (waiting for an incoming connection), *established* (connections which are established), and *time wait* (the foreign or remote machine has already closed the connection, but that the local program somehow hasn't followed suit). If the *Foreign Address* is *:* (with *TCP* sockets & *LISTEN* state), a socket is usually waiting for some remote host to send the first data. When connecting to a foreign host, a program on the computer usually doesn't care which local port is used for the connection. That's why the port on the local side isn't usually recognized and translated to a protocol like *"https"* or *"www".*

$ netstat –r # Routing Table

```
roopansh@roopansh-Ubuntu:~$ netstat -r
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 eno1
default         XiaoQiang       0.0.0.0         UG        0 0          0 wlo1
10.0.0.0        *               255.255.252.0   U         0 0          0 eno1
link-local      *               255.255.0.0     U         0 0          0 eno1
192.168.31.0    *               255.255.255.0   U         0 0          0 wlo1
```

Figure 8: Kernel IP Routing Table

It shows the Kernel Routing Table of the Machine. The *Destination* column indicates the pattern that the destination of a packet is compared to. When a packet has to be sent over the network, this table is examined top to bottom, and the first line with a matching destination is then used to determine where to send the packet. The *Gateway* column tells the computer where to send a packet that matches the destination of the same line. An *asterisk (*)* here means *send locally*.The *Genmask* column tells how many bits from the start of the ip address are used to identify the subnet. As a rule of thumb, it is 255 for non-zero part and 0 for other parts of the

destination. The *Flags* column displays the flags that describe the route - **G***(route uses a gateway)*, **U***(interface is up)*, **H***(Only a single host can be reached through the route)*, **D***(route is dynamically created)*, **M***(route is set if the table entry was modified by an ICMP redirect message)*, **!***(route is a reject route and datagrams will be dropped)*. The next three columns show the *MSS*, *Window* and *irtt* that will be applied to TCP connections established via this route. The *MSS*(*Maximum Segment Size)* is the size of the largest datagram the kernel will construct for transmission via this route. The *Window* is the maximum amount of data the system will accept in a single burst from a remote host. The acronym irtt is the *initial round trip time*. The *Iface* column tells which network interface should be used for sending packets that match the destination.

$netstat –i	# Network Interface Status

```
roopansh@roopansh-Ubuntu:~$ 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
eno1      1500 0       664      0      2 0           121      0      0      0 BMRU
lo       65536 0       696      0      0 0           696      0      0      0 LRU
wlo1      1500 0       790      0      0 0           912      0      0      0 BMRU
```

Figure 9: Kernel Interface Table

My machine has 3 interfaces, which are *eno1* (Wired Ethernet), *lo* (loopback device), *wlo1* (Wireless Ethernet). The *MTU* and *Met* fields show the current MTU and metric values for that interface. The *RX* and *TX* columns show how many packets have been received or transmitted *error-free(RX-OK/TX-OK)*, *damaged(RX-ERR/TX-ERR)*, *dropped(RX-DRP/TX-DRP)* and *lost because of an overrun (RX-OVR/TX-OVR)*. The last column shows the flags that have been set for this interface- **B***(broadcast address has been set)*, **L***(it is a loopback device)*, **M***(all packets are received, i.e. promiscuous mode)*, **O***(ARP is turned off for this interface)*, **P***(This is a point-to-point connection)*, **R***(Interface is running)*, **U***(Interface is up)*.

The **loopback device** is a special, virtual network interface that the computer uses to communicate with itself. It is used mainly for diagnostics and troubleshooting, and to connect to servers running on the local machine. When a network interface is disconnected, no communication on that interface is possible, not even communication between the computer and itself. The loopback interface does not represent any actual hardware, but exists so applications running on the computer can always connect to servers on the same machine. For example, if you run a web server, you have all your web documents and could examine them file by file on the local machine. For IPv4, the loopback interface is assigned all the IPs in the 127.0.0.0/8 address block (i.e.127.0.0.1 through 127.255.255.254).

6 TRACEROUTE

- The readings are taken at **1.00pm**, **7.00pm** and **1.00am (IST)** respectively.
- Test PC was connected to **DIGITALOCEAN VPN (Bangalore,India)** while performing the experiment.

	digitalocean.com	iitg.ac.in	Justpakit.com	Seedr.cc	Youtube.com
Hop Count #1	8	12	3	8	10
Hop Count #2	8	12	3	9	10
Hop Count #3	8	12	3	8	10

Table 4: Hopcounts of 5 Servers

The obvious common hops found were – *10.8.0.1 (My machine)* and *139.59.64.253(VPN Service Provider's IP)* which were common in all the experiments. *138.197.249.14* was also a common hop between all the hosts except justpakit.com. *138.197.249.18 & 138.197.249.22* was a common hop between digitalocean, seedr and youtube. *124.124.67.154* was a common hop found between *digitalocean.com* & *youtube.com*. Hops are common because of the reason that routes to these destinations pass through the same internet circles and hence overlap.

The route to the hosts changes at different times of the day in the experiments because of network congestion. The packets are redirected by the nodes to take a route having less traffic. The load balancing is done to reduce the load of congested path.

Sometimes, traceroute might not find a complete path to some host. Some servers/hosts along the path may have not been configured to respond to the ICMP Traffic or may have set up firewalls which block the ICMP Traffic. However, they still send the data to the next hop as there are results that follow. Many network providers disable ICMP traffic if their network is under heavy load.

It is possible to find the route to certain hosts which fail to respond with ping experiment. The *ping* and *traceroute* both use the ICMP Packets but there working is different. Ping is straight ICMP from point A to point B, that traverses networks via routing rules and expects a ICMP Reply from the host. Most probably the server is blocking the reply. On the other hand, Traceroute sends packets with TTL values that gradually increase from packet to packet. Routers decrement TTL values of packets by one and discard packets whose TTL value has reached zero, returning the ICMP error (ICMP Time Exceeded). Traceroute looks for the ICMP Time exceeded packet and not the ICMP Reply Packet, and that is why it might be possible.

7 ARP

```
roopansh@roopansh-Ubuntu:~$ sudo arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.76                ether   70:4d:7b:bd:2e:99   C                     eno1
10.0.1.39                ether   c8:d3:ff:cb:09:b4   C                     eno1
10.0.2.83                        (incomplete)                              eno1
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     eno1
10.0.2.68                ether   d0:bf:9c:0f:7f:18   C                     eno1
10.0.1.37                ether   f4:8e:38:ef:d4:a1   C                     eno1
10.0.0.60                ether   d4:81:d7:b3:f2:29   C                     eno1
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     eno1
Entries: 8      Skipped: 0      Found: 8
roopansh@roopansh-Ubuntu:~$ sudo arp -sv 10.0.0.1 ff:ff:ff:ff:ff:ff
arp: SIOCSARP()
roopansh@roopansh-Ubuntu:~$ sudo arp -sv 10.0.0.2 ff:ff:ff:ff:00:00
arp: SIOCSARP()
roopansh@roopansh-Ubuntu:~$ sudo arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.254               ether   4c:4e:35:97:1e:ef   C                     eno1
10.0.2.68                ether   d0:bf:9c:0f:7f:18   C                     eno1
10.0.2.83                        (incomplete)                              eno1
10.0.0.60                ether   d4:81:d7:b3:f2:29   C                     eno1
10.0.2.76                ether   70:4d:7b:bd:2e:99   C                     eno1
10.0.0.2                 ether   ff:ff:ff:ff:00:00   CM                    eno1
10.0.0.1                 ether   ff:ff:ff:ff:ff:ff   CM                    eno1
10.0.1.39                ether   c8:d3:ff:cb:09:b4   C                     eno1
10.0.2.74                ether   a0:8c:fd:22:51:b0   C                     eno1
10.0.1.37                ether   f4:8e:38:ef:d4:a1   C                     eno1
Entries: 10     Skipped: 0      Found: 10
```

Figure 10: Adding Entries in ARP Table

ARP stands for Address Resolution Protocol. ***$ arp –a*** shows the complete ARP Table of the machine. It shows the *IP-Address*, the corresponding *MAC Address* and the *Network Interface*. When we try to ping an IP address on our local network, say 10.0.0.1, the system has to turn the IP address 10.0.0.1 into a MAC address. This involves using ARP Table to resolve it. An entry for the IP address can be deleted from the ARP table using the command ***"$ arp –d <address>"***. If you want to make a specific MAC address be used for an IP, use the command: ***"$ arp -s <ip_addr> <MAC_addr>"***. You need to run it as a root user(use sudo).

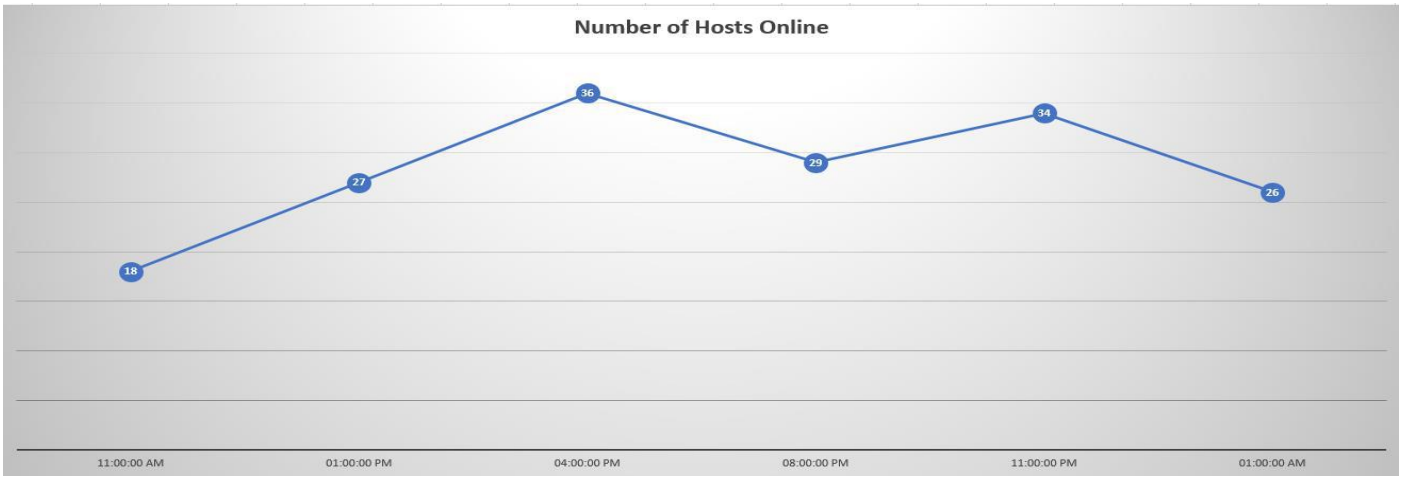
Entries stay cached in the ARP table for **60 seconds**. A trial & error method to discover the timeout value is to add a temporary entry in the arp table and keep on checking the arp table after fixed intervals of time (say 5 seconds). The time after which it dissapears is approximately the required cache timeout. For a better appproximation, decrease the interval length. Alternatively, one can use binary search also for finding the cache time, for e.g. – Add a temporary entry in ARP and check after 5000ms. If then entry has been deleted, then add the entry again and check after 2500ms. Continue in Binary sense to find the cache time.

The scenario where two IP's can map to same Ethernet Address is when a router or a gateway connects two or more subnet ranges. When communicating with machines on the same subnet range, MAC address is used for directing the packages. In the ARP Table, the IP's of the devices which are connected in the other subnet range have the ethernet address/MAC address as that of the Router or Gateway which connects the two subnet ranges. ARP table is referred to convert these IP addresses to the MAC address and packets are sent to it(router/gateway). The router then uses it's routing table and sends the packet further to the correct device.

8 NMAP

The following command is used for this question. The IP's analysed are that of **Dihing Hostel.**

```
$ nmap -n -sP 10.0.0.254/22
```



From the above graph, one can easily notice that the number of hosts are low in the early morning around 11.00 AM and steadily increase till the afternoon (around 4.00PM). After that there is slight decrease or almost same number of hosts online till evening around 11:00PM. The number of hosts online starts to decrease after that.