

CSP334: Computer Networks, Lab Assignment No 3, Traceroute ping commands

Rahul Byas Sherwan
Entry No. : 2016UCS0028

1: The First Problem

(a) What if there was no TTL field in the invocation of the traceroute at all?

Traceroute works on the principal of TTL of a packet and loopbacks (that is upto how much time the packet is in the network). If there was no TTL field in the invocation of the traceroute then we will only be able to see the IP address (and other informations such as time etc) of the destination system. We will not be able to see the IPAs of the hops in between the source and destination as no loopbacks is possible .

The other drawback of not having a TTL concept is it will create congestion in the network.

(b) How will the routers in between determine whether the TTL value limit has reached ?

Each router has a to do only two things,i.e.

1. Decrement the TTL by 1.
2. Compare whether the value is zero or not.If it is zero then the value limit has reached , otherwise it will loopback and go to the next hop.

This two steps have to be repeated by each hop.

For example : Suppose a packet has x TTL , then the first hop will decrement its TTL (now after decrementing its TTL is x-1) and forward it to next hop . This process will keep on repeating by each hop unless it has reached to zero . In that case the router will come to know that the TTL value has reached.

(c) Should an intermediate router that receives a traceroute packet always respond with an ICMP TTL exceeded message ? If the answer is a yes, reason why and if the answer is a no, then argue how do we know the address of all the routers/hops in between us and the destination ?

No, not always respond with an ICMP TTL exceeded message.It will respond this message only when the incoming packet has TTL 1.

Since , the IP address of the destination or the last hop which respond with ICMP TTL exceeded message is written on the packet that this IPA has responded with ICMP message. So, in this manner while loopingback each hop just forward it to the nearest hop as the source IP is written on it.

(d) Why does traceroute make use of a destination UDP port number which is

invalid - i.e. it sends a packet to a UDP port in the range **h33434 to 33534i** ?

If we are using traceroute it means we just want to know the IP and we don't want any service . That's why traceroute uses destination UDP port number which is invalid.

(e) How do we know the address of all the routers/hops in between us and the destination when using the traceroute?

Since traceroute uses loopbacks for acknowledging each hop. So, what it does is, it first go to the nearest hop and loopback and after then to the next hop nearest to the previous hop and again loopbacks. It repeats this process again and again untill TTL limit is reached or untill destination is not found. While looping back it gives the address of the perticular hop. So, in this way , we know the address of all the routers/hops between us and the destination when using the traceroute.

(f) How is traceroute latency calculated?

A roundtrip time of a single packet is its latency . So, When we send a packet ,we recorded that time and when the packet is recieved ,we recorded that time too . Then we subtract the initial time from the final time . That difference is the latency.

2: The Second Problem

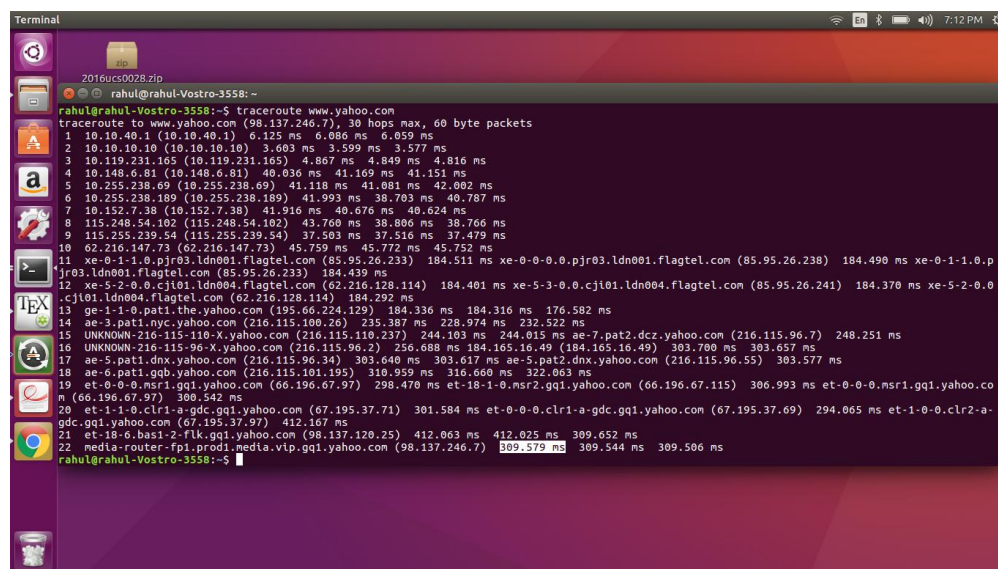


Figure 1: traceroute

As you can see in the image above IP address of yahoo.com that was used for the trace route : 98.137.246.7

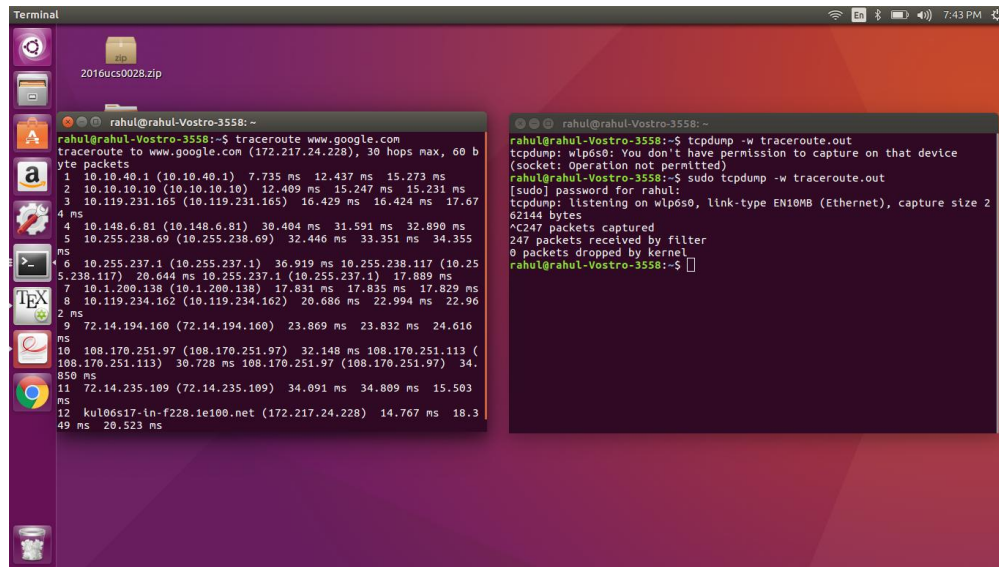
The number of iterations required to determine route : 22

The IP addresses of all the machines between the source and the destination : We can see that in the screendump above.

The average round trip time of the packet that reached the destination: 309.543 ms

3: The Third Problem

After running the command screendump :



The screenshot shows a Linux desktop environment with a terminal window open. The terminal displays the output of a traceroute to google.com and the output of tcpdump. The traceroute shows 12 hops, with the final hop being google.com. The tcpdump output shows 247 packets received by the filter and 0 packets dropped by the kernel.

```
Terminal
2016ucs0028.zip

rahul@rahul-Vostro-3558:~$ traceroute www.google.com
traceroute to www.google.com (172.217.24.228), 30 hops max, 60 b
yte packets
 1 10.10.40.1 (10.10.40.1) 7.735 ms 12.437 ms 15.273 ms
 2 10.10.10.10 (10.10.10.10) 12.409 ms 15.247 ms 15.231 ms
 3 10.119.231.165 (10.119.231.165) 16.429 ms 16.424 ms 17.67
 4 ms
 4 10.148.6.81 (10.148.6.81) 30.404 ms 31.591 ms 32.890 ms
 5 10.255.238.69 (10.255.238.69) 32.446 ms 33.351 ms 34.355
 ms
 6 10.255.237.1 (10.255.237.1) 36.919 ms 10.255.238.117 (10.25
 5.238.117) 20.644 ms 10.255.237.1 (10.255.237.1) 17.889 ms
 7 10.1.200.138 (10.1.200.138) 17.831 ms 17.835 ms 17.829 ms
 8 10.119.234.162 (10.119.234.162) 20.686 ms 22.994 ms 22.96
 2 ms
 9 72.14.194.160 (72.14.194.160) 23.869 ms 23.832 ms 24.616
 ms
10 108.170.251.97 (108.170.251.97) 32.148 ms 108.170.251.113 (
108.170.251.113) 30.728 ms 108.170.251.97 (108.170.251.97) 34.
850 ms
11 72.14.235.109 (72.14.235.109) 34.091 ms 34.809 ms 15.503
 ms
12 kul06s17-in-f228.1e100.net (172.217.24.228) 14.767 ms 18.3
49 ms 20.523 ms

rahul@rahul-Vostro-3558:~$ tcpdump -w traceroute.out
tcpdump: wlp6s0: You don't have permission to capture on that device
(socket: Operation not permitted)
rahul@rahul-Vostro-3558:~$ sudo tcpdump -w traceroute.out
[sudo] password for rahul:
tcpdump: listening on wlp6s0, link-type EN10MB (Ethernet), capture size 2
62144 bytes
^C247 packets captured
247 packets received by filter
0 packets dropped by kernel
rahul@rahul-Vostro-3558:~$
```

Figure 2: tcpdump listening

(a) How many packets are send by traceroute in each iteration ? How can you prove this using the tcpdump output?

Packets that are send by traceroute in each iteration: 3

As we can see in the tcpdump output that 3 packets have been send to IPA 72.14.194.160 and IPA 10.119.234.162 (screenshot below).

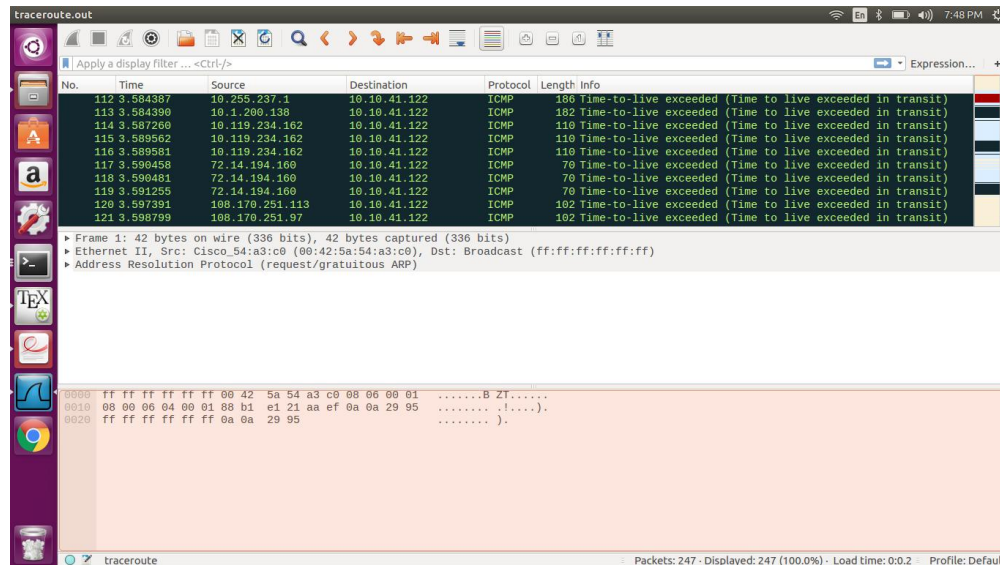


Figure 3: tcpdump output

(b) Consider one specific iteration of traceroute invocation/iteration. For this specific iteration, what are the individual round trip times of each of the three probes sent ? What is the average round trip time ? Does it match with the round trip time returned by traceroute ?

I am considering iteration of 10.10.40.1. The individual round trip time are :

- 1 . 7.735ms (can see in the screenshot below)
2. 12.437 ms
3. 15.273 ms

Average :11.815 ms

Yes, it approximately matches the average round trip time.

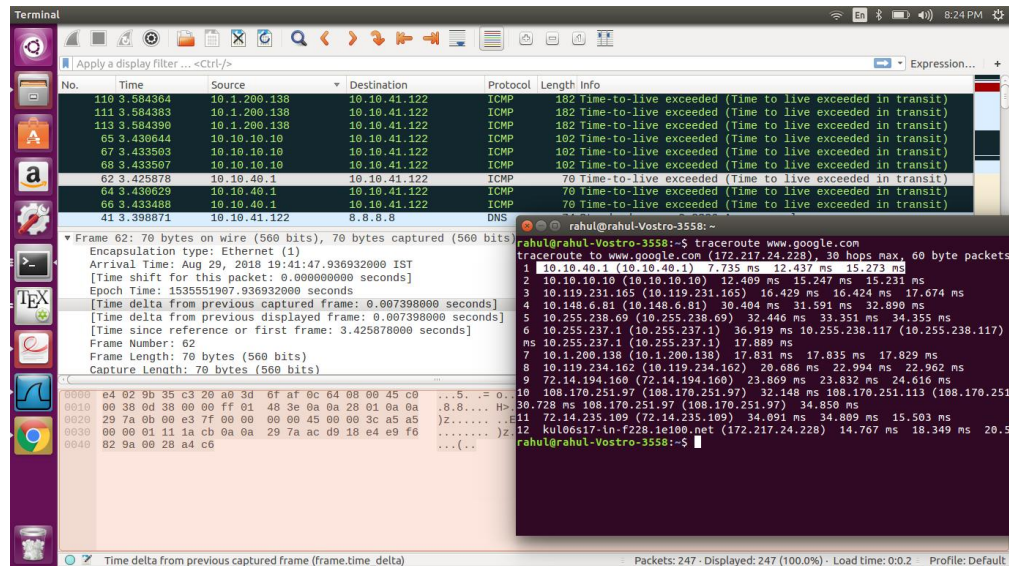


Figure 4: 10.10.40.1 first packet's timestamp

(c) In each iteration of traceroute does it use the same port number for the destination ? IF yes, reason why and if no, then also argue why does it do so.

No, it does not use the same port numbers. This is done basically to check the order in which the packets are sent . Another way to argue about the above condition is that it might be possible that the ports are being utilized by come other application time at the same time , so in such cases it will not respond to our request . And hence we require an another port number to complete our task.

4: The Fourth Problem

open above link and trace the route of google.com

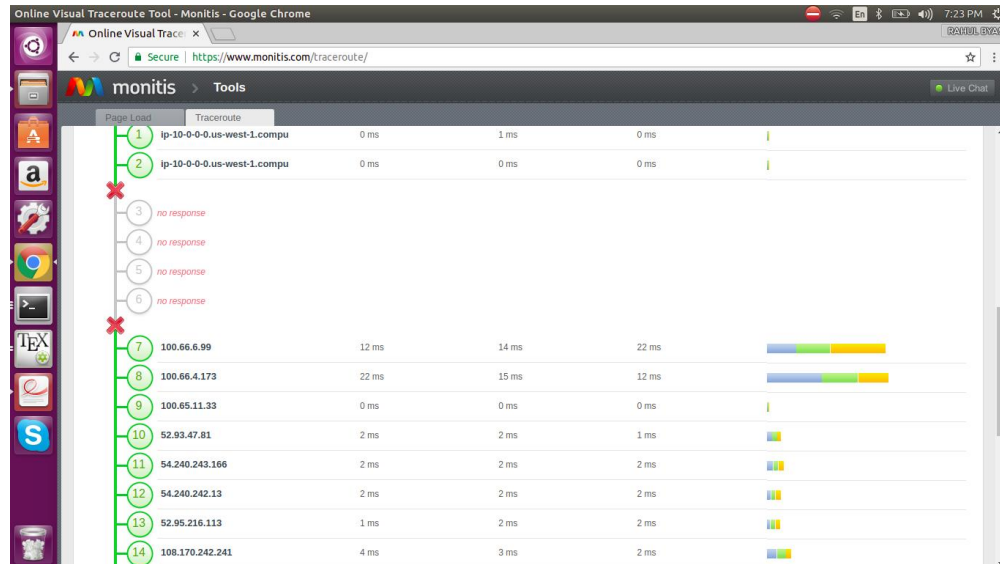


Figure 5: www.monitis.com/traceroute/

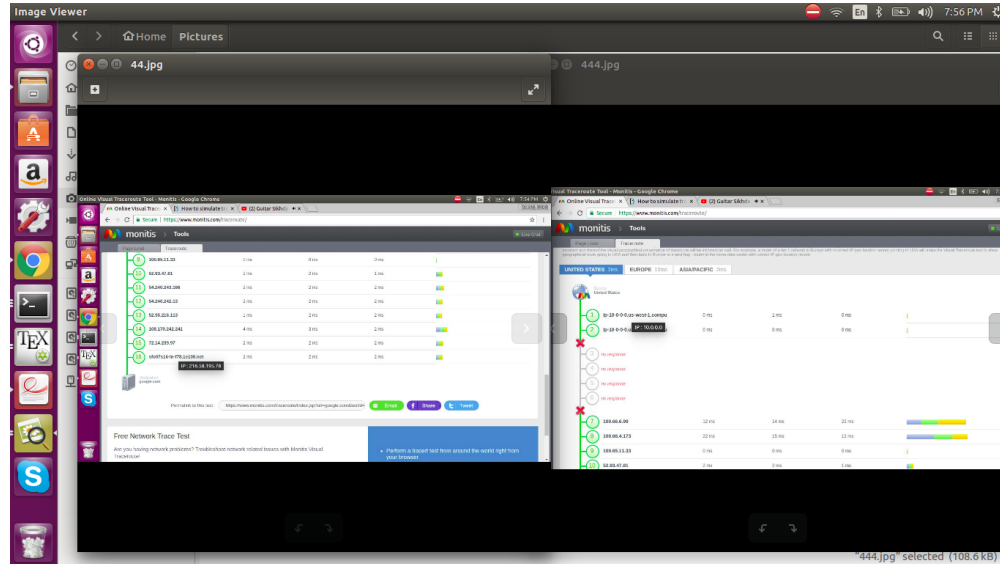


Figure 6: www.monitis.com/traceroute/

source IP:10.0.0.0 destination IP:216.58.195.78

5: The Fifth Problem

Whenever we run a traceroute command for an IP which is protected by firewall then it would not respond any ICMP message.

Instead it will show an *** which has two possibilities ,i.e. either the TTL limit has reached or a firewall has been encountered by the packet in the way to destination.

In order to confirm that it is a firewall , we need to find some IPA related to previous IPA after which *** was acheived.We can use ping sweep to find IPAs related to that IPAs.

If the IPA of the hop just after that IPA is kind of similar to the IPA then it is a firewall.

6: The Sixth Problem

In this case, the last IP appearing indicates the address of the destination.

7: The Seventh Problem

Ping is used to ensure that a host computer the user is trying to reach is actually operating or not. Ping works by sending an Internet Control Message Protocol (ICMP) Echo request to a specified interface on the network and waiting for a reply. Ping can be used for troubleshooting to test connectivity and determine response time. Ping can also be used to gather Information about the maximum frame size on the network.(help : wikipedia)

Usage of ping program :

1. Sends an ICMP message to the argument hostname and wait for the response

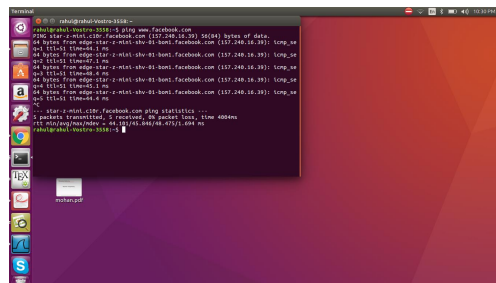


Figure 7: ping

2. Records any loss of packets

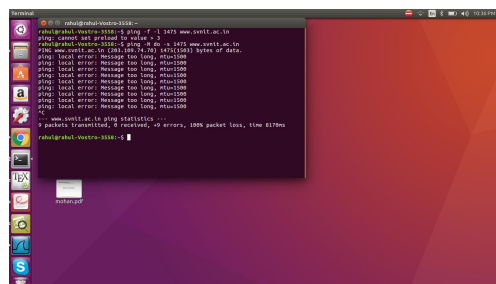


Figure 8: ping

3. The maximum frame size on the network.

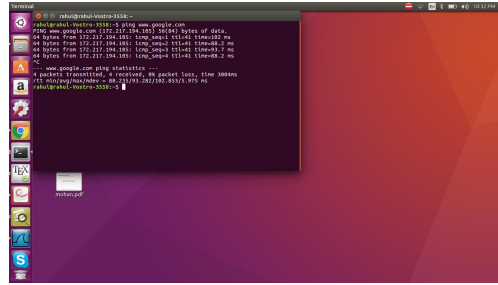


Figure 9: ping

4. Find the IPA of all the routers along the path to the host and the maximum hops to the target

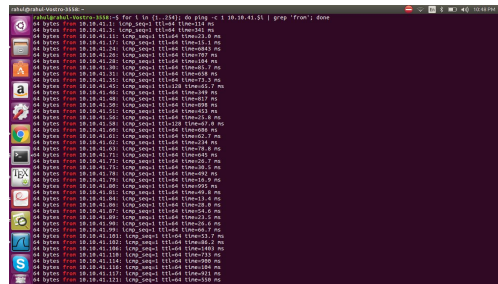


Figure 10: ping

- ## 5. Identifying the presence of a firewall

Run : `ping -v -T tsandaddr www.svnit.ac.in` (No firewall as you can see in the screenshot)

```
Run : ping -v -T tsandaddr www.du.edu (Firewall detected 100 percent packet lost.)
```

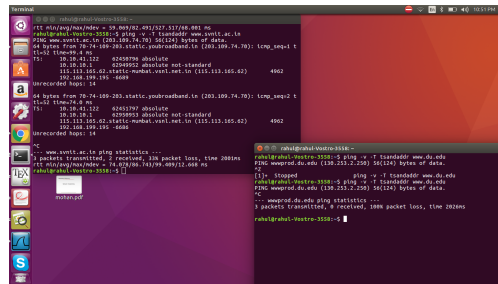


Figure 11: ping

8: The Eighth Problem

Given below is a screenshot in which we have run a small bash script:

EXPLANATION:

Over here it is just a for loop in which i is a variable iterating from 1 to 30 and that i is used for creating different IPAs inside the loop which is used by ping to check its availability . Grep command is given to follow the next IPA from previous IPA.

-c is basically number of counts it is sending to ping request

-t is for TTL of a packet.


```

rahul@rahul-Vostro-3558:~$ ping -v -T tsandaddr www.du.edu
PING wwwprod.du.edu (130.253.2.250) 56(124) bytes of data.
^C
--- wwwprod.du.edu ping statistics ---
123 packets transmitted, 0 received, 100% packet loss, time 124938ms

rahul@rahul-Vostro-3558:~$ for i in {1..30}; do ping -t $i -c 1 google.com; done | grep "Time to live exceeded"
From 10.10.10.10 icmp_seq=1 Time to live exceeded
From 10.119.231.165 icmp_seq=1 Time to live exceeded
From 10.148.0.81 icmp_seq=1 Time to live exceeded
From 10.255.238.69 icmp_seq=1 Time to live exceeded
From 10.255.237.1 icmp_seq=1 Time to live exceeded
From 10.1.200.138 icmp_seq=1 Time to live exceeded
From 10.119.234.162 icmp_seq=1 Time to live exceeded
From 72.14.195.56 icmp_seq=1 Time to live exceeded
From 108.170.251.97 icmp_seq=1 Time to live exceeded
From 72.14.233.217 icmp_seq=1 Time to live exceeded
rahul@rahul-Vostro-3558:~$

```

Figure 12: loop command

9: The Nineth Problem

Helping Reference : www.slashroot.in/what-ping-sweep-and-how-do-ping-sweep

Ping sweep is a way to find that various system which are around us are connected to the network or not. There are various approaches for ping sweep and one can use fping, gping, nmap and simple bash for loop for implementing this technique.

(a) Using nmap:

```

rahul@rahul-Vostro-3558:~$ nm nm-connection-editor nmtui-edit
nm nmap nm-online nmtui-hostname
nm-applet nmtui
nmcli nmtui-connect
rahul@rahul-Vostro-3558:~$ nmap -sp 10.10.41.0-142
Starting Nmap 7.01 ( https://nmap.org ) at 2018-08-27 19:52 IST
Nmap scan report for 10.10.41.12
Host is up (0.044s latency).
Nmap scan report for 10.10.41.16
Host is up (0.035s latency).
Nmap scan report for 10.10.41.18
Host is up (0.014s latency).
Nmap scan report for 10.10.41.27
Host is up (0.010s latency).
Nmap scan report for 10.10.41.31
Host is up (0.079s latency).
Nmap scan report for 10.10.41.47
Host is up (0.12s latency).
Nmap scan report for 10.10.41.53
Host is up (0.0048s latency).
Nmap scan report for 10.10.41.56
Host is up (0.032s latency).
Nmap scan report for 10.10.41.58
Host is up (0.014s latency).
Nmap scan report for 10.10.41.67
Host is up (0.040s latency).
Nmap scan report for 10.10.41.92
Host is up (0.081s latency).
Nmap scan report for 10.10.41.93
Host is up (0.018s latency).
Nmap scan report for 10.10.41.118
Host is up (0.17s latency).
Nmap scan report for 10.10.41.120
Host is up (0.011s latency).
Nmap scan report for 10.10.41.122
Host is up (0.000098s latency).
Nmap scan report for 10.10.41.125
Host is up (0.076s latency).
Nmap scan report for 10.10.41.137
Host is up (0.079s latency).
Nmap done: 143 IP addresses (17 hosts up) scanned in 5.80 seconds
rahul@rahul-Vostro-3558:~$

```

Figure 13: nmap

(b) Using fping:

```

rahul@rahul-Vostro-3558:~$ fping -g 10.10.41.122 10.10.41.126
10.10.41.122 is alive
10.10.41.125 is alive
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.124
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.123
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.126
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.124
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.123
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.124
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.122
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.125
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.123
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.126
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.124
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.122
ICMP Host Unreachable from 10.10.41.122 for ICMP Echo sent to 10.10.41.125
10.10.41.123 is unreachable
10.10.41.124 is unreachable
10.10.41.126 is unreachable
rahul@rahul-Vostro-3558:~$

```

Figure 14: fping

(c) Using simple Bash for loop:

```

rahul@rahul-Vostro-3558:~$ for i in {1..254}; do ping -c 1 10.10.41.$i | grep 'from'; done
64 bytes from 10.10.41.10: icmp_seq=1 ttl=64 time=70.1 ms
64 bytes from 10.10.41.17: icmp_seq=1 ttl=64 time=1286 ms
64 bytes from 10.10.41.18: icmp_seq=1 ttl=64 time=602 ms
64 bytes from 10.10.41.24: icmp_seq=1 ttl=64 time=3019 ms
64 bytes from 10.10.41.27: icmp_seq=1 ttl=64 time=215 ms
64 bytes from 10.10.41.28: icmp_seq=1 ttl=64 time=310 ms
64 bytes from 10.10.41.38: icmp_seq=1 ttl=64 time=631 ms
64 bytes from 10.10.41.35: icmp_seq=1 ttl=64 time=335 ms
64 bytes from 10.10.41.40: icmp_seq=1 ttl=64 time=317 ms
64 bytes from 10.10.41.45: icmp_seq=1 ttl=128 time=17.0 ms
64 bytes from 10.10.41.50: icmp_seq=1 ttl=64 time=176 ms
64 bytes from 10.10.41.51: icmp_seq=1 ttl=64 time=3371 ms
64 bytes from 10.10.41.53: icmp_seq=1 ttl=64 time=99.7 ms
64 bytes from 10.10.41.58: icmp_seq=1 ttl=128 time=22.8 ms
64 bytes from 10.10.41.65: icmp_seq=1 ttl=128 time=281 ms
64 bytes from 10.10.41.67: icmp_seq=1 ttl=64 time=68.3 ms
64 bytes from 10.10.41.75: icmp_seq=1 ttl=64 time=101 ms
64 bytes from 10.10.41.79: icmp_seq=1 ttl=64 time=629 ms
64 bytes from 10.10.41.81: icmp_seq=1 ttl=64 time=867 ms
64 bytes from 10.10.41.84: icmp_seq=1 ttl=64 time=267 ms
64 bytes from 10.10.41.96: icmp_seq=1 ttl=128 time=74.7 ms
64 bytes from 10.10.41.97: icmp_seq=1 ttl=64 time=819 ms
64 bytes from 10.10.41.99: icmp_seq=1 ttl=64 time=569 ms
64 bytes from 10.10.41.102: icmp_seq=1 ttl=64 time=2015 ms
64 bytes from 10.10.41.114: icmp_seq=1 ttl=64 time=1606 ms
64 bytes from 10.10.41.116: icmp_seq=1 ttl=64 time=1537 ms
64 bytes from 10.10.41.120: icmp_seq=1 ttl=64 time=522 ms
64 bytes from 10.10.41.121: icmp_seq=1 ttl=64 time=2071 ms
64 bytes from 10.10.41.122: icmp_seq=1 ttl=64 time=8.040 ms
64 bytes from 10.10.41.126: icmp_seq=1 ttl=64 time=1829 ms
64 bytes from 10.10.41.127: icmp_seq=1 ttl=64 time=4054 ms
64 bytes from 10.10.41.130: icmp_seq=1 ttl=64 time=1037 ms
64 bytes from 10.10.41.131: icmp_seq=1 ttl=64 time=1074 ms
64 bytes from 10.10.41.132: icmp_seq=1 ttl=64 time=1313 ms
64 bytes from 10.10.41.135: icmp_seq=1 ttl=64 time=1827 ms
64 bytes from 10.10.41.149: icmp_seq=1 ttl=64 time=1139 ms
64 bytes from 10.10.41.150: icmp_seq=1 ttl=64 time=594 ms
64 bytes from 10.10.41.172: icmp_seq=1 ttl=64 time=1952 ms
64 bytes from 10.10.41.177: icmp_seq=1 ttl=64 time=523 ms
64 bytes from 10.10.41.179: icmp_seq=1 ttl=64 time=1869 ms

```

Figure 15: loop command