CSE 232: Programming Assignment 1
Using command-line utilities for network debugging

Anushka Srivastava (2022086)

August 2024

Question 1

a.) Learn to use the ifconfig command, and figure out the IP address of your network interface. Put a screenshot.

```
(base) anushka_04@LAPTOP-5JMBES8H:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.21.107.19  netmask 255.255.240.0  broadcast 172.21.111.255
        inet6 fe80::215:5dff:fecb:8f1f  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:cb:8f:1f  txqueuelen 1000  (Ethernet)
        RX packets 467  bytes 903686 (903.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 295  bytes 24166 (24.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 1: Running the ifconfig command.

The IP address of our network interface is represented by the `inet` value under the `eth0` network interface, which is the name of our ethernet network interface.
The IP address of my network interface from this image is **172.21.107.19**.

b.) Go to the webpage https://www.whatismyip.com and find out what IP is shown for your machine. Are they identical or different? Why?

Figure 2: Results on https://www.whatismyip.com.

The IP address on the given website is represented by **My Public IPv4** value which in this case is **49.36.176.152**.

The IP addresses represented by both the images are **different**. This is due to the distinction between private and public IP addresses. The IP address returned by `ifconfig` command is the **private** IP address of our device, which is a unique identifier for our device on a local network assigned by our network routers.
The IP address returned by the given website is the **public** IP address of our device, which is unique identifier for our device over the internet, assigned to our network router by the Internet Service Provider.

Question 2

a.) Change the IP address of your network interface using the command line. Put a screenshot that shows the change. Revert to the original IP address.

We can use the `sudo ifconfig eth0 <IP address>` command to change the IP address of our network interface.

```
(base) anushka_04@LAPTOP-5JMBES8H:~$ sudo ifconfig eth0 192.168.1.100
[sudo] password for anushka_04:
(base) anushka_04@LAPTOP-5JMBES8H:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::215:5dff:fecb:8f1f  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:cb:8f:1f  txqueuelen 1000  (Ethernet)
        RX packets 8999  bytes 19007323 (19.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5024  bytes 482258 (482.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3: Changing the IP address to 192.168.1.100

To revert to the original IP address, we can use the same command and enter our original IP address.

```
(base) anushka_04@LAPTOP-5JMBES8H:~$ sudo ifconfig eth0 172.21.107.19
(base) anushka_04@LAPTOP-5JMBES8H:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.21.107.19  netmask 255.255.0.0  broadcast 172.21.255.255
        inet6 fe80::215:5dff:fecb:8f1f  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:cb:8f:1f  txqueuelen 1000  (Ethernet)
        RX packets 8999  bytes 19007323 (19.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5024  bytes 482258 (482.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4: Reverting to original IP Address

Question 3

a.) Use "netcat" to set up a TCP client/server connection between your VM and host machine. If you are not using a VM, you can set up the connection with localhost. Put a screenshot.

We use the `netcat` command to set up a connection between server *(Figure 5)* and client *(Figure 6)*, by using our `localhost` on port `1234`.

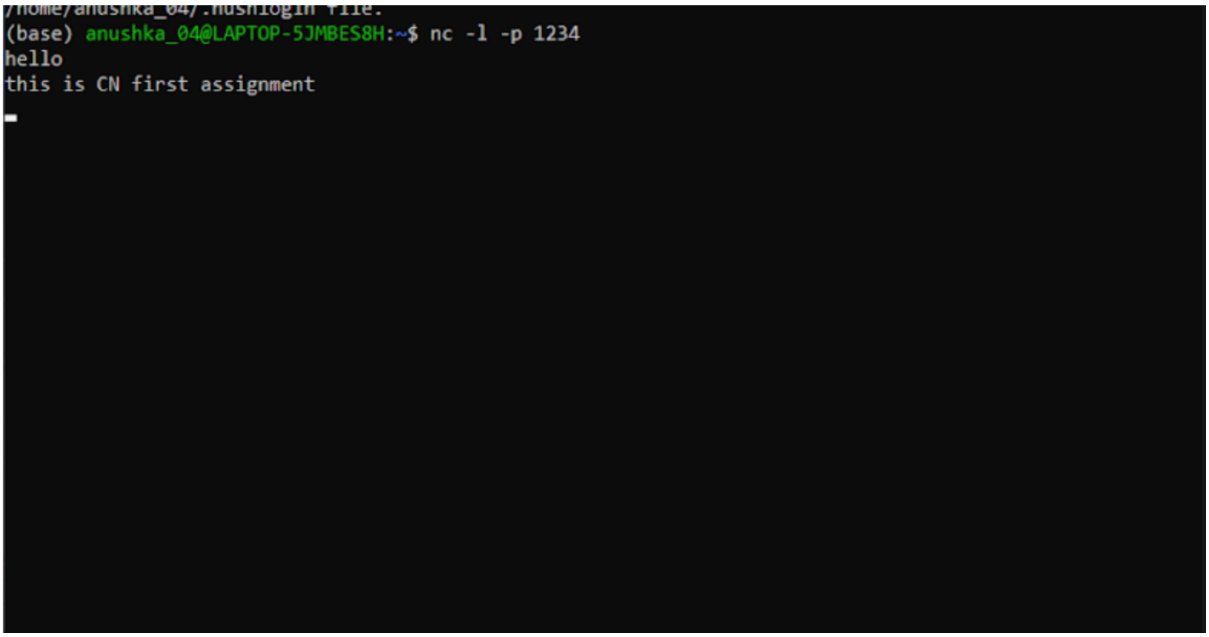

Figure 5: Server

Figure 6: Client

b.) Determine the state of this TCP connection(s) at the client node. Put a screenshot.

We can use `netsat` command with `-t` flag to display all TCP connections. For our port `1234`, we see that the state of the TCP connection at both client and server node is `ESTABLISHED`.

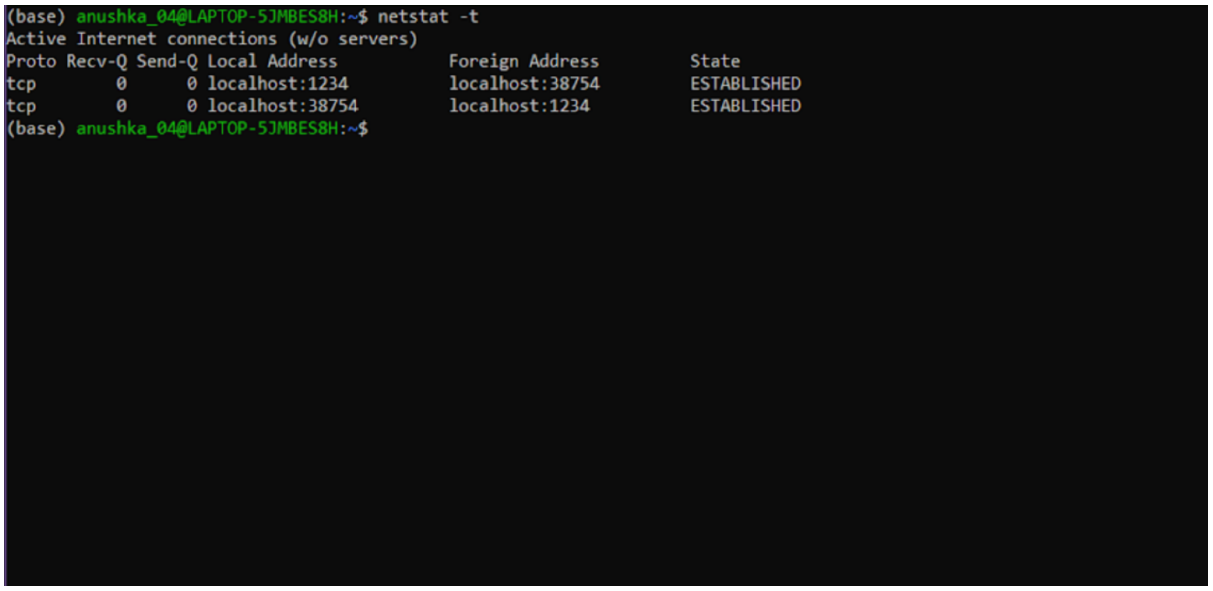

Figure 7: State of TCP connection using `netstat -t` command.

Question 4

a.) Get an authoritative result for "google.in" using nslookup. Put a screenshot. Explain how you did it.

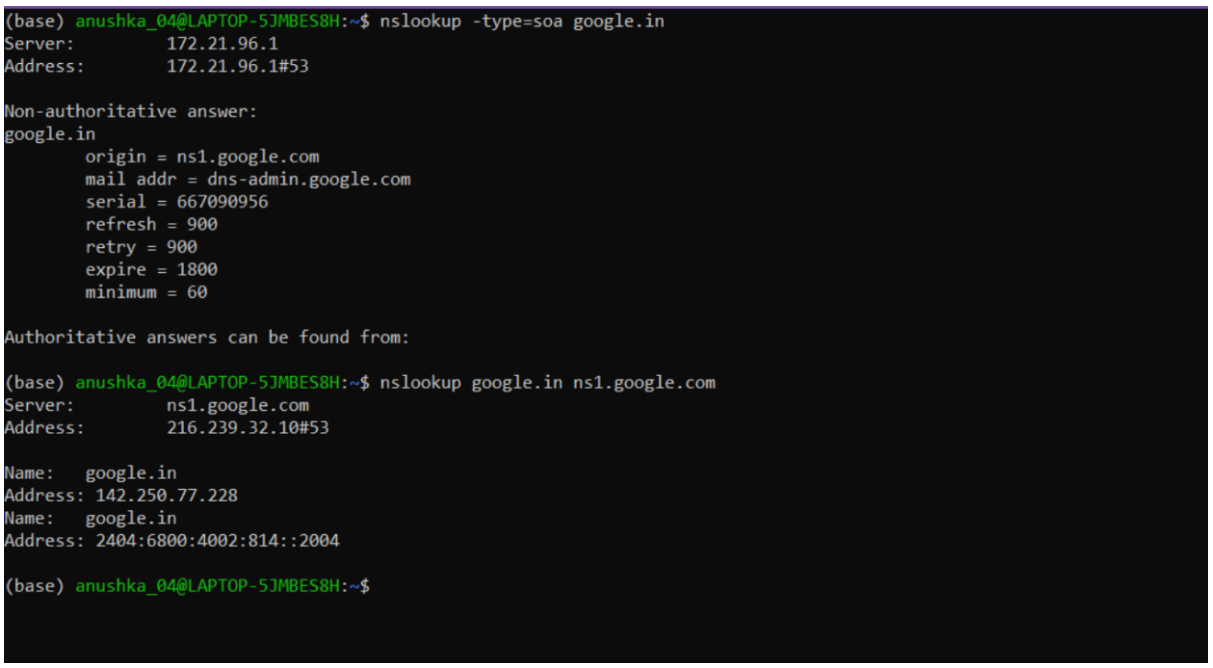

Figure 8: Authoritative result for `google.in` using `nslookup`

On using the command `nslookup google.in`, we get the **non-authoritative** result from the cache of a DNS server around the internet and not from the authoritative server of `google.in`. To get the authoritative result, we follow these steps:

1. Run the `nslookup -type=soa google.in` command to find the authoritative name-server for the domain name. The **SOA Record (State of Authority)** contains administrative information about the zone, including the primary authoritative name server and contact details and various timers. Hence, we use this DNS record with the `-type` flag.

2. We receive a response specifying the primary name server and associated information. We see that the primary name server for `google.in` is `ns1.google.com`.

3. To receive an authoritative answer, we send the query to the authoritative server of google.in. This is a response we get directly from the primary DNS server holding the master copy of the zone file.

4. We run the command `nslookup <sub domain name> <name-server>`, that is we run `nslookup google.in ns1.google.com` to get an authoritative result, as displayed in the image.

Reference: Baeldung CS: How to Find the Authoritative Name Server for a Domain?

b.) Find out the time to live for any website on the local DNS. Put a screenshot. Explain in words (with unit) after how much time this entry would expire from the local DNS server.

We find out the **time to live** for `google.in` using the `dig` command. The answer is displayed in the **ANSWER** section, which is **300 seconds**.

```
(base) anushka_04@LAPTOP-5JMBES8H:~$ dig +nssearch google.in
;; UDP setup with 2001:4860:4802:32::a#53(2001:4860:4802:32::a) for google.in failed: network unreachable.
;; UDP setup with 2001:4860:4802:38::a#53(2001:4860:4802:38::a) for google.in failed: network unreachable.
;; UDP setup with 2001:4860:4802:36::a#53(2001:4860:4802:36::a) for google.in failed: network unreachable.
;; UDP setup with 2001:4860:4802:34::a#53(2001:4860:4802:34::a) for google.in failed: network unreachable.
SOA ns1.google.com. dns-admin.google.com. 667287868 900 900 1800 60 from server 216.239.34.10 in 70 ms.
SOA ns1.google.com. dns-admin.google.com. 667287868 900 900 1800 60 from server 216.239.38.10 in 90 ms.
SOA ns1.google.com. dns-admin.google.com. 667287868 900 900 1800 60 from server 216.239.32.10 in 90 ms.
SOA ns1.google.com. dns-admin.google.com. 667519583 900 900 1800 60 from server 216.239.36.10 in 120 ms.
(base) anushka_04@LAPTOP-5JMBES8H:~$ dig @ns1.google.com +noall +answer google.in
google.in.              300     IN      A       142.250.77.228
(base) anushka_04@LAPTOP-5JMBES8H:~$
```

Figure 9: Time to Live for `google.in` (Method 1)

```
(base) anushka_04@LAPTOP-5JMBES8H:~$ dig @ns1.google.com google.in

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> @ns1.google.com google.in
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64385
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.in.                     IN      A

;; ANSWER SECTION:
google.in.              300     IN      A       142.250.77.228

;; Query time: 99 msec
;; SERVER: 216.239.32.10#53(ns1.google.com) (UDP)
;; WHEN: Tue Aug 27 00:39:41 IST 2024
;; MSG SIZE  rcvd: 54

(base) anushka_04@LAPTOP-5JMBES8H:~$
```

Figure 10: Time to Live for `google.in` (Method 2)

TTL is a value associated with a DNS record that tells other servers how long it takes to cache the DNS record before reaching out to the authoritative DNS server and getting a new copy of the record. From our results, we can see that it takes **300 seconds** for an entry to expire from the local DNS server.

Hence, the DNS entry for `google.in` will expire from the local DNS server **300 seconds** after it was first cached, requiring the DNS server to fetch a fresh record from an authoritative server if another request is made after this time.

Question 5

a.) Run the command, traceroute google.in. How many intermediate hosts do you see? What are the IP addresses? Compute the average latency to each intermediate host. Put a screenshot.

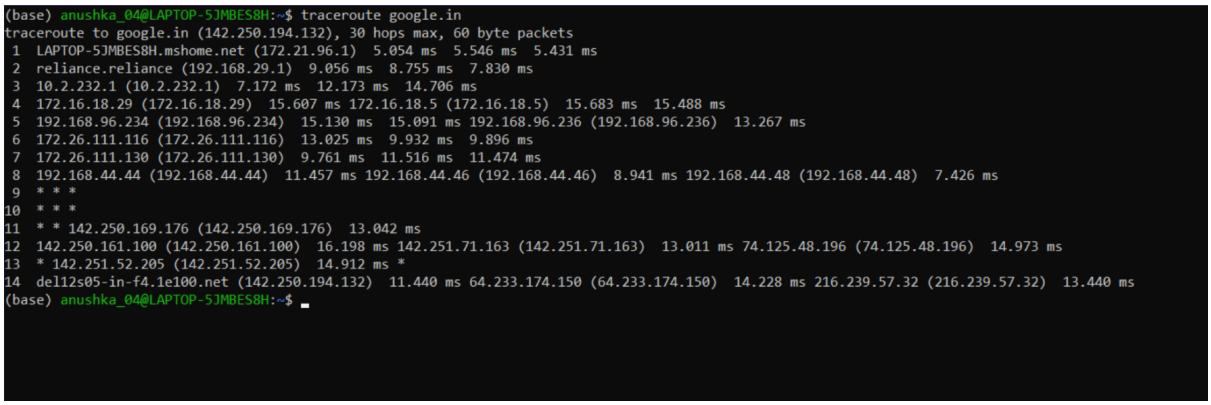

Figure 11: Running command `traceroute google.in`

From the attached image, we see that there are **12** intermediate hosts, excluding the source and destination IP.

The `traceroute` command sends 3 probes per hop. According to the path taken by each probe packet, it might result in multiple IP addresses for each hop.
All the IP addresses for each probe packet in each hop are listed in the following table:

Hop	Probe 1	Probe 2	Probe 3
1	172.21.96.1	172.21.96.1	172.21.96.1
2	192.168.29.1	192.168.29.1	192.168.29.1
3	10.2.232.1	10.2.232.1	10.2.232.1
4	172.16.18.29	172.16.18.5	172.16.18.5
5	192.168.96.234	192.168.96.234	192.168.96.236
6	172.26.111.116	172.26.111.116	172.26.111.116
7	172.26.111.130	172.26.111.130	172.26.111.130
8	192.168.44.44	192.168.44.46	192.168.44.48
9	N/A	N/A	N/A
10	N/A	N/A	N/A
11	N/A	N/A	142.250.169.176
12	142.250.161.100	142.251.71.163	74.125.48.196
13	N/A	142.251.52.205	N/A
14	142.250.194.132	64.233.174.150	216.239.57.32

Table 1: IP Address of each probe in each hop

The average latency of each hop can be calculated by adding up the latency values of each probe packet and dividing it by the number of probe packets that are available for that particular hop.
The following table displays the average latency for each intermediate host, excluding the source and destination IP:

Hop	Probe 1	Probe 2	Probe 3	Average Latency
2	9.056	8.755	7.830	8.547
3	7.172	12.173	14.706	11.350
4	15.607	15.683	15.488	15.593
5	15.130	15.091	13.267	14.496
6	13.025	9.932	9.896	10.951
7	9.761	11.516	11.474	10.917
8	11.457	8.941	7.426	9.275
9	N/A	N/A	N/A	N/A
10	N/A	N/A	N/A	N/A
11	N/A	N/A	13.042	13.042
12	16.198	13.011	14.973	14.727
13	N/A	14.912	N/A	14.912

Table 2: Average latency for each hop (in ms)

b.) Send 50 ping messages to google.in, Determine the average latency. Put a screenshot.

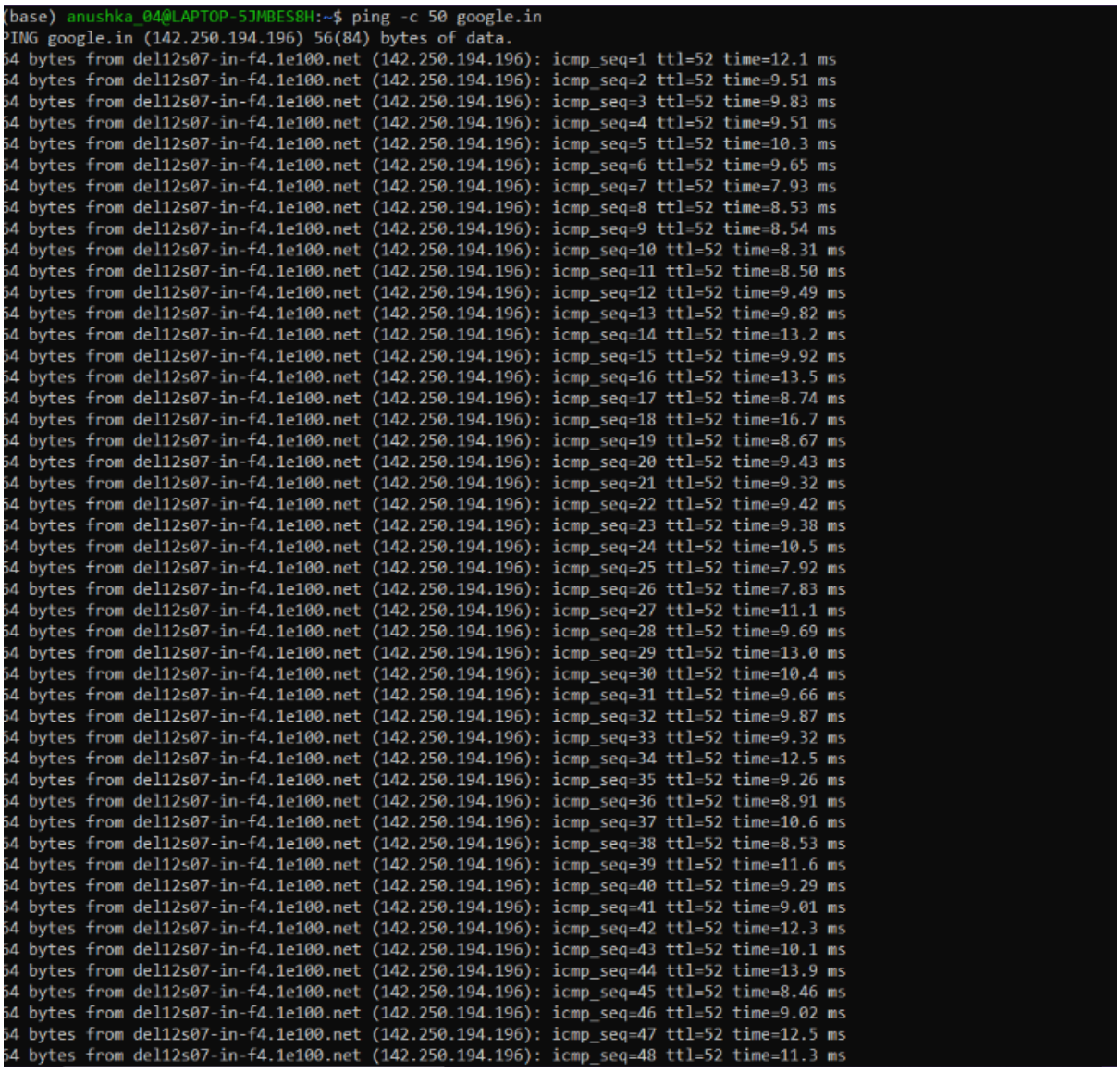

Figure 12: Running `ping -c 50 google.in`

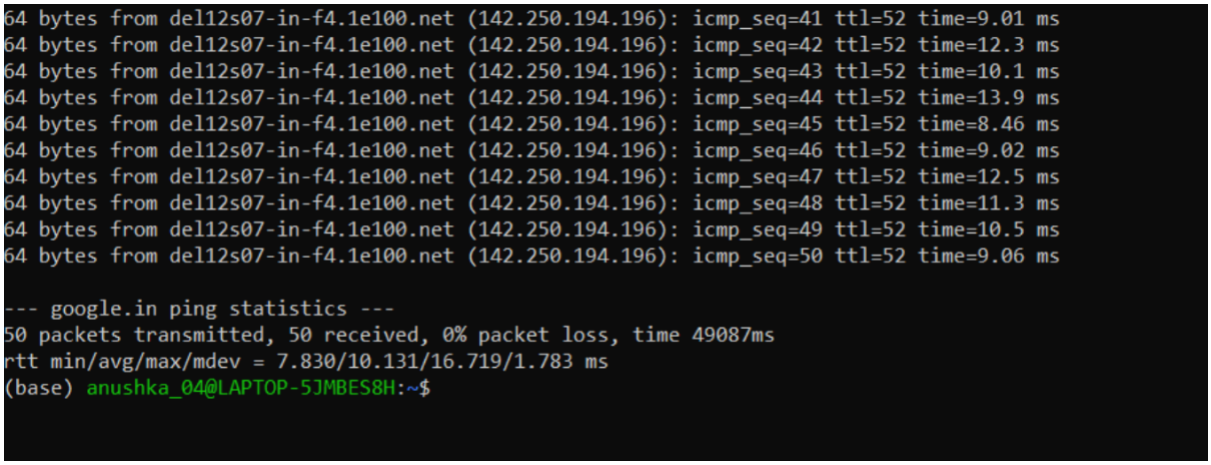

Figure 13: Running `ping -c 50 google.in`

From the second image, we see that the average latency of all ping messages is **10.131 ms**.

c.) Add up the ping latency of all the intermediate hosts obtained in (a) and compare with (b). Are they matching, explain?

On adding up the average ping latency of all the intermediate hosts from **(a) Table 2**, we get the answer as **123.813 seconds**.
This value is not equivalent to the average latency obtained using `ping` command.

The latency in each hop for the `traceroute` command is the resultant sum of:

- The time taken to forward the packet to the **displayed router hop**.
- The time taken for the router to generate an ICMP response packet.
- The time taken for that ICMP packet to reach the sender.

On the other hand, the latency in each result for the `ping` command is the resultant sum of:

- The time taken to send one or more ICMP Echo Request packet to the specified **destination IP**.
- The time taken for the router to generate an ICMP Echo Request packet.
- The time taken for the destination to respond with an ICMP echo reply.

Hence, the sum of all average latencies in `traceroute` is not equivalent to the average latency of `ping` command because the first sum returns the total round trip time for each packet sent to each **intermediate host** while the second value returns the round trip time for the echo packet sent to the **destination IP**.

d.) Take the maximum ping latency amongst the intermediate hosts (in (a)) and compare it with (b). Are they matching, explain?

The maximum ping latency amongst the intermediate hosts is **15.593 ms**.
This value is not equivalent to the average latency obtained using `ping` command. As explained in **(c)**, the first value represents the average round trip time from the source IP to the intermediate host 3 (hop number 4), including the time to wait for the response packet. The second value represents the round trip time for the echo packet sent to the **destination IP**. Hence, both values are not equivalent.

e.) You may see multiple entries for a single hop while using the traceroute command. What do these entries mean?

The `traceroute` commands send multiple probe packets per hop. The default value for this count is 3. Each probe is a completely **independent** trial, which means that each packet may be forwarded down a completely different path to reach the destination IP. This is visible to the user as multiple IP addresses for each hop. Hence, multiple entries for a single hop may be seen.

f.) Send 50 ping messages to stanford.edu, Determine the average latency. Put a screenshot.

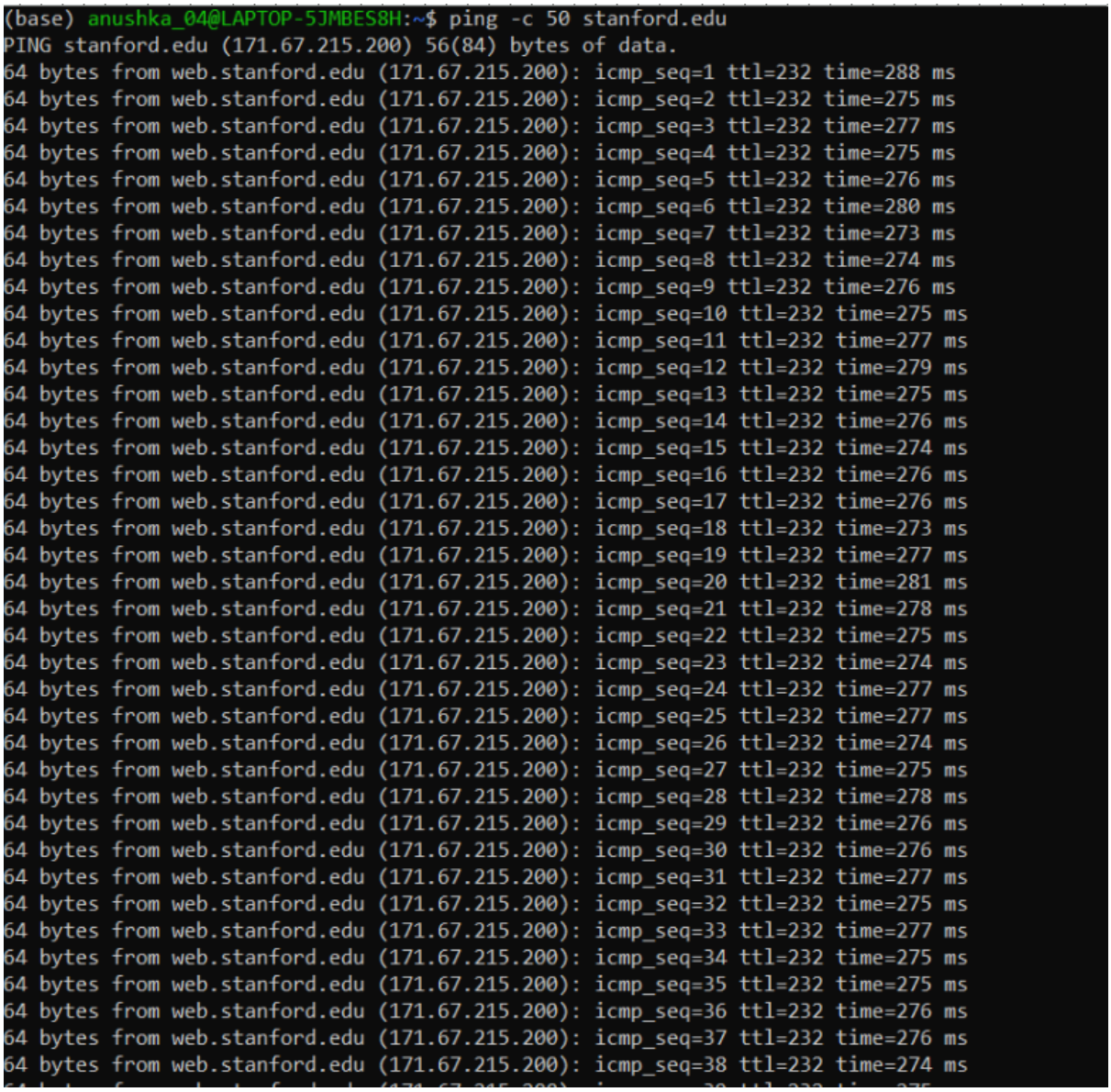

Figure 14: Running `ping -c 50 stanford.edu`

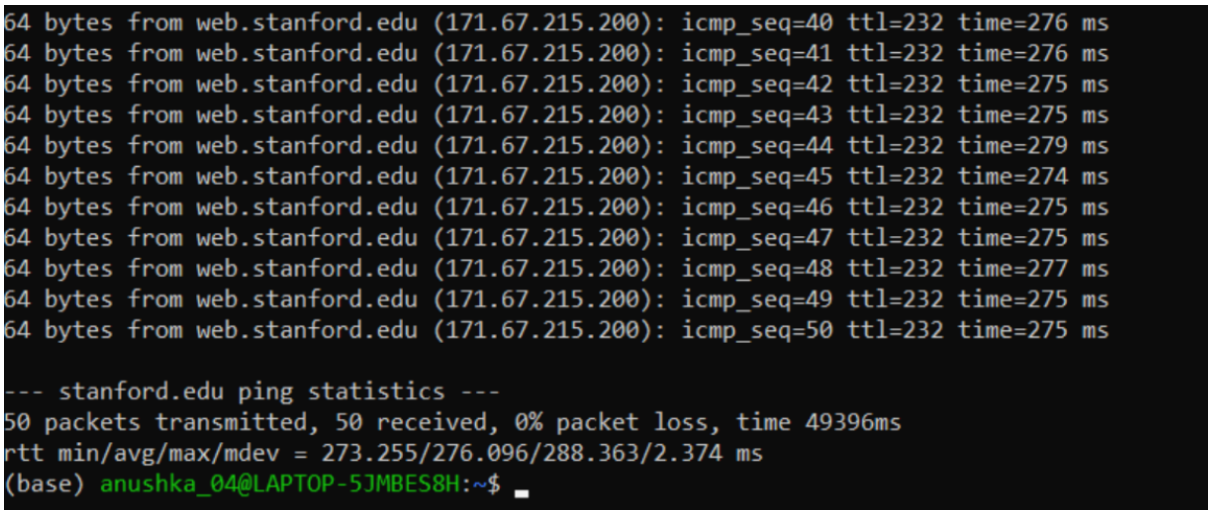

Figure 15: Running `ping -c 50 stanford.edu`

From the second image, we see that the average latency of all ping messages is **276.096 ms**.

g.) Run the command, traceroute stanford.edu. Compare the number of hops between google.in and stanford.edu (between the traceroute result of google.in and stanford.edu).

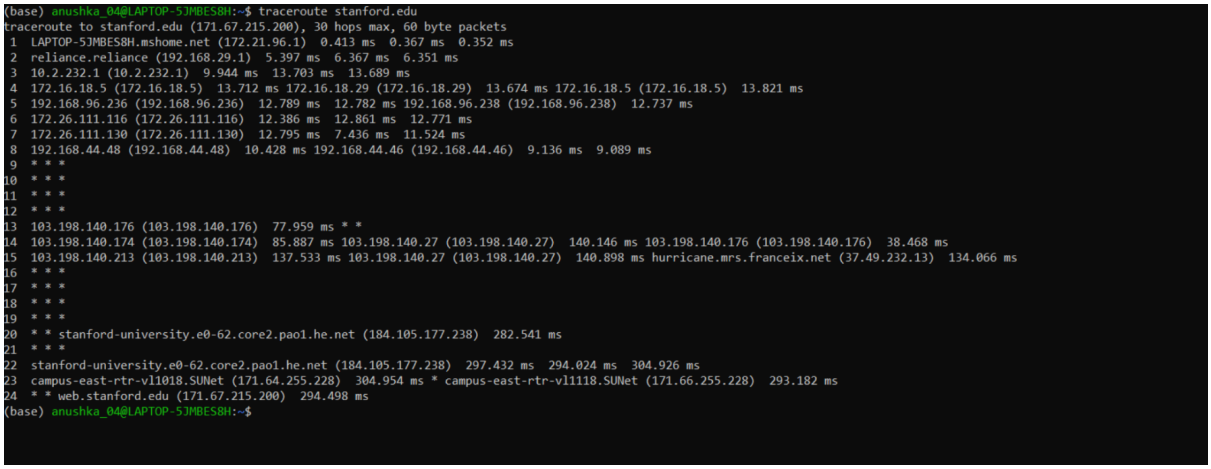

Figure 16: Running `traceroute stanford.edu`

We see that `traceroute google.in` returned **14 hops** and `traceroute stanford.edu` returned **24 hops**. The number of hops returned by **stanford.edu** is more than **google.in** due to a difference in physical locations between both the destinations. The physical location for the server of **stanford.edu** is farther than the server of **google.in**, as compared with the source IP.

h.) Can you explain the reason for the latency difference between google.in and stanford.edu (see (b) & (f))?

From **(b)** and **(f)**, we see that the latency for **stanford.edu** is more than **google.in**. This is due to the increase in geographical distance between the source and the destination, which further results in increased number of hops for the farther destination, in this case **stanford.edu**.

Question 6

a.) Make your ping command fail for 127.0.0.1 (with 100% packet loss). Explain how you do it. Put a screenshot that it failed.

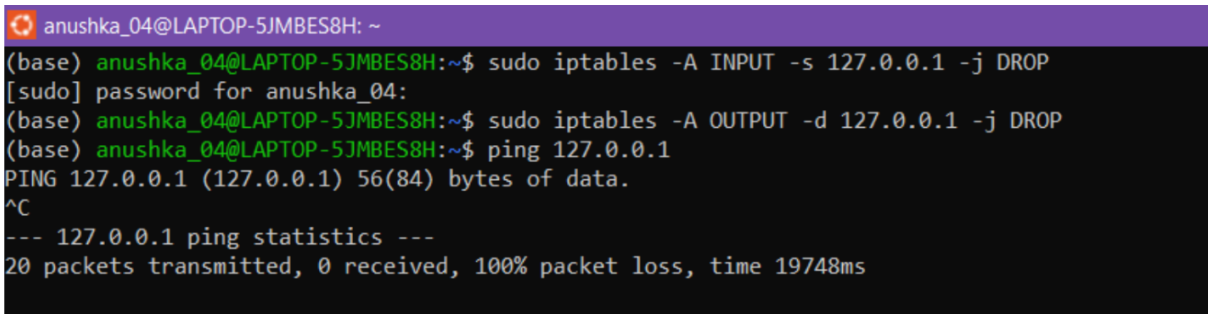

Figure 17: `ping` command failing with 100% packet loss.

A `ping` command can fail when there is no response to the packets that are being transmitted.

To achieve this, we use the `iptables` command to configure the Linux kernel firewall such that it **blocks** all network connections that originate from the specified IP address, in this case `127.0.0.1`. It drops all incoming and outgoing packets for the IP address.

We add two rules to the INPUT chain and OUTPUT chain, handling incoming and outgoing packets, respectively, and configure them to drop the packets.

Now, on running `ping 127.0.0.1`, we see that the command has failed with 100% packet loss as the firewall is **dropping all the packets** going to and from the specified address.

Reference: Iptables Essentials: Common Firewall Rules and Commands

References

[1] Tecmint Linux Blog: 15 Useful "ifconfig" Commands to Configure Network Interface in Linux

[2] Baeldung CS: How to Find the Authoritative Name Server for a Domain?

[3] Linode: A Complete Guide to the nslookup Command

[4] VIP Documentation: Check DNS record time to live (TTL)

[5] Richard A Steenbergen: A Practical Guide to (Correctly) Troubleshooting with Traceroute

[6] Iptables Essentials: Common Firewall Rules and Commands