COMPUTER NETWORKS LAB
WEEK 1

NAME: VISHWA MEHUL MEHTA
SRN: PES2UG20CS389
SECTION: F
DATE: 21/01/2022

Study and understand the basic networking tools - Wireshark, Tcpdump, Ping, Traceroute and Netcat.

Task 1: Linux Interface Configuration (ifconfig / IP command)

Step 1: To display status of all active network interfaces.

ifconfig (or) ip addr show

```
vishwa@pop-os:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1
        inet 10.0.2.15  netmask 255.255.255.0  broadcast
        inet6 fe80::c4c3:1e2:88b7:59c6  prefixlen 64  sco
        ether 08:00:27:4f:0b:16  txqueuelen 1000  (Ethern
        RX packets 212320  bytes 260305880 (260.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98034  bytes 11674264 (11.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  col

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
```

```
vishwa@pop-os:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue sta
n 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdi
efault qlen 1000
    link/ether 08:00:27:4f:0b:16 brd ff:ff:ff:ff:ff:ff
```

Analyze and fill the following table:

ip address table:

Interface name	IP address (IPv4 / IPv6)	MAC address	
lo	127.0.0.1	00:00:00:00:00:00	Loop back device
enp0s3	10.0.2.15	08:00:27:4f:0b:16	Ethernet
WLAN	-	-	-

Step 2: To assign an IP address to an interface, use the following command. **sudo ifconfig interface_name 10.0.your_section.your_sno netmask 255.255.255.0** (or) **sudo ip addr add 10.0.your_section.your_sno /24 dev interface_name**

```
vishwa@pop-os:~$ sudo ifconfig enp0s3 10.0.6.56
vishwa@pop-os:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qle
n 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group d
efault qlen 1000
    link/ether 08:00:27:4f:0b:16 brd ff:ff:ff:ff:ff:ff
    inet 10.0.6.56/8 brd 10.255.255.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::c4c3:1e2:88b7:59c6/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
vishwa@pop-os:~$
```

Step 3: To activate / deactivate a network interface, type.

sudo ifconfig interface_name down

sudo ifconfig interface_name up

```
vishwa@pop-os:~$ sudo ifconfig enp0s3 down
vishwa@pop-os:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:4f:0b:16 brd ff:ff:ff:ff:ff:ff
vishwa@pop-os:~$
```

```
vishwa@pop-os:~$ sudo ifconfig enp0s3 up
vishwa@pop-os:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4f:0b:16 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 86398sec preferred_lft 86398sec
    inet6 fe80::c4c3:1e2:88b7:59c6/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
vishwa@pop-os:~$
```

Step 4: To show the current neighbor table in kernel, type **ip neigh**

```
vishwa@pop-os:~$ ip neigh
10.0.2.2 dev enp0s3 lladdr 52:54:00:12:35:02 STALE
vishwa@pop-os:~$
```

Task 2: Ping PDU (Packet Data Units or Packets) Capture

Step 1: Assign an IP address to the system (Host).

Note: IP address of your system should be 10.0.your_section.your_sno.

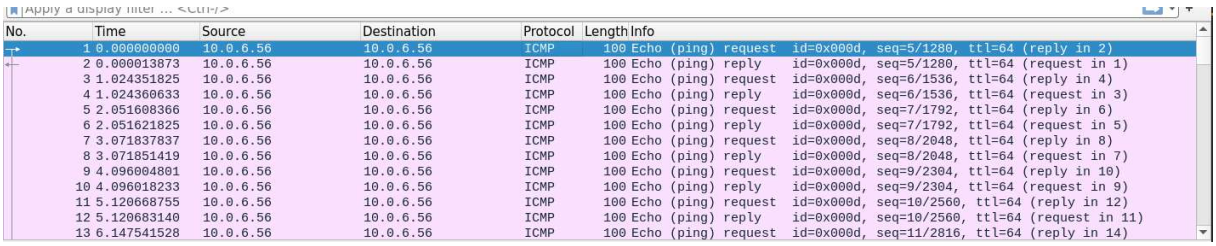
```
vishwa@pop-os:~$ sudo ifconfig enp0s3 10.0.6.56
vishwa@pop-os:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.6.56  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::c4c3:1e2:88b7:59c6  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4f:0b:16  txqueuelen 1000  (Ethernet)
        RX packets 229785  bytes 272227538 (272.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 110926  bytes 15223124 (15.2 MB)
        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 22715  bytes 2295158 (2.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22715  bytes 2295158 (2.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 2: Launch Wireshark and select ‘any’ interface

```
vishwa@pop-os:~$ sudo wireshark
18:45:24.062     Main Warn QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Step 3: In terminal, type **ping 10.0.your_section.your_sno**

```
vishwa@pop-os:~$ ping 10.0.6.56
PING 10.0.6.56 (10.0.6.56) 56(84) bytes of data.
64 bytes from 10.0.6.56: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 10.0.6.56: icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from 10.0.6.56: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 10.0.6.56: icmp_seq=4 ttl=64 time=0.050 ms
64 bytes from 10.0.6.56: icmp_seq=5 ttl=64 time=0.060 ms
64 bytes from 10.0.6.56: icmp_seq=6 ttl=64 time=0.042 ms
64 bytes from 10.0.6.56: icmp_seq=7 ttl=64 time=0.061 ms
64 bytes from 10.0.6.56: icmp_seq=8 ttl=64 time=0.048 ms
64 bytes from 10.0.6.56: icmp_seq=9 ttl=64 time=0.047 ms
64 bytes from 10.0.6.56: icmp_seq=10 ttl=64 time=0.062 ms
64 bytes from 10.0.6.56: icmp_seq=11 ttl=64 time=0.038 ms
64 bytes from 10.0.6.56: icmp_seq=12 ttl=64 time=0.061 ms
64 bytes from 10.0.6.56: icmp_seq=13 ttl=64 time=0.045 ms
64 bytes from 10.0.6.56: icmp_seq=14 ttl=64 time=0.044 ms
64 bytes from 10.0.6.56: icmp_seq=15 ttl=64 time=0.048 ms
64 bytes from 10.0.6.56: icmp_seq=16 ttl=64 time=0.057 ms
64 bytes from 10.0.6.56: icmp_seq=17 ttl=64 time=0.062 ms
64 bytes from 10.0.6.56: icmp_seq=18 ttl=64 time=0.074 ms
64 bytes from 10.0.6.56: icmp_seq=19 ttl=64 time=0.053 ms
^C
--- 10.0.6.56 ping statistics ---
19 packets transmitted, 19 received, 0% packet loss, time 18418ms
rtt min/avg/max/mdev = 0.037/0.051/0.074/0.009 ms
```



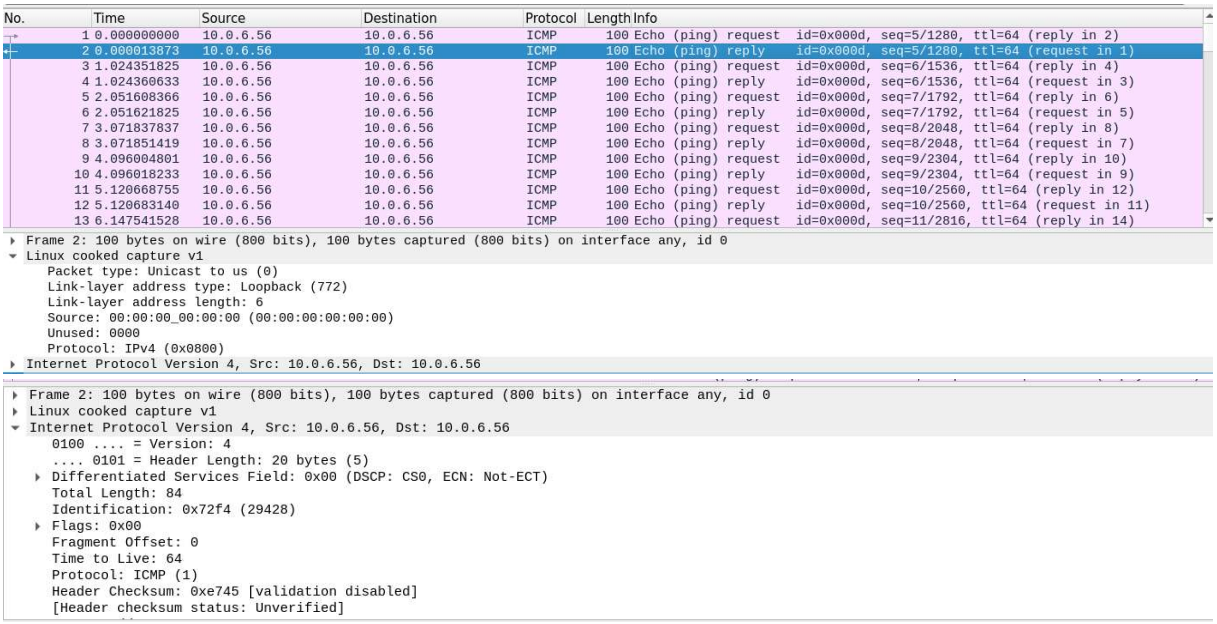
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=5/1280, ttl=64 (reply in 2)
2	0.000013873	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=5/1280, ttl=64 (request in 1)
3	1.024351825	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=6/1536, ttl=64 (reply in 4)
4	1.024360633	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=6/1536, ttl=64 (request in 3)
5	2.051608366	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=7/1792, ttl=64 (reply in 6)
6	2.051621825	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=7/1792, ttl=64 (request in 5)
7	3.071837837	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=8/2048, ttl=64 (reply in 8)
8	3.071851419	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=8/2048, ttl=64 (request in 7)
9	4.096004801	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=9/2304, ttl=64 (reply in 10)
10	4.096018233	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=9/2304, ttl=64 (request in 9)
11	5.120668755	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=10/2560, ttl=64 (reply in 12)
12	5.120683140	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=10/2560, ttl=64 (request in 11)
13	6.147541528	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=11/2816, ttl=64 (reply in 14)

Observations to be made

Step 4: Analyze the following in Terminal

- TTL : **64**
- Protocol used by ping : **ICMP (Internet Control Message Protocol)**
- Time : **18418ms**

Step 5: Analyze the following in Wireshark
On Packet List Pane, select the first echo packet on the list. On Packet Details Pane, click on each of the four “+” to expand the information. Analyze the frames with the first echo request and echo reply and complete the table below.
First echo request:

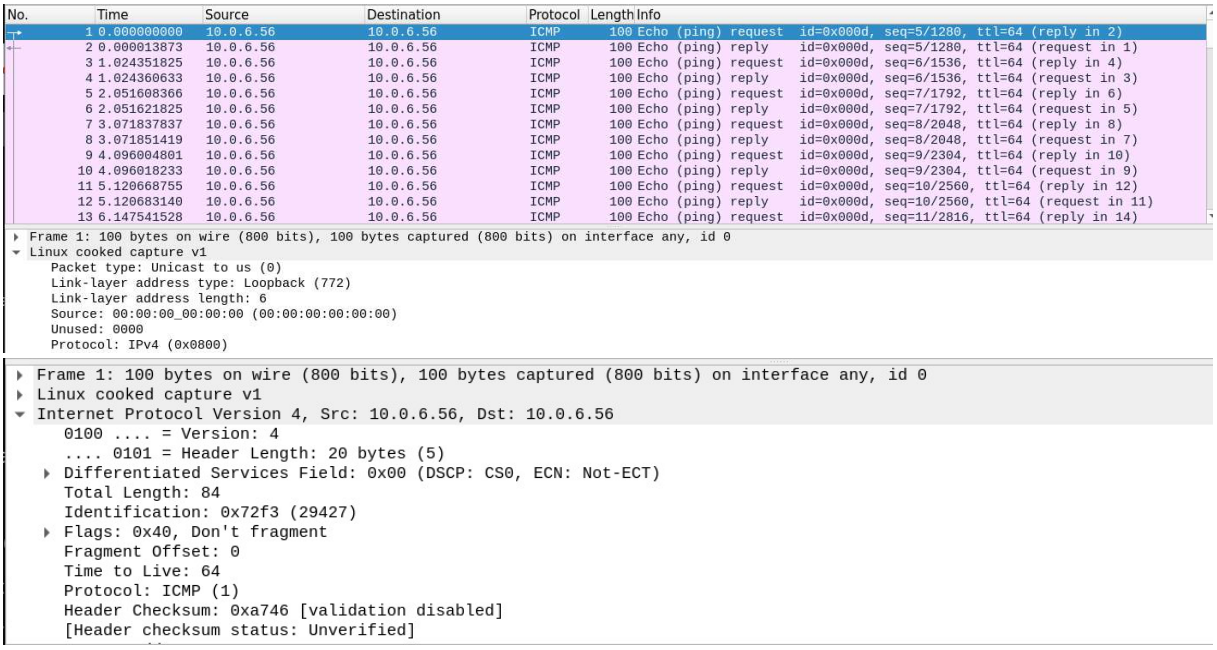


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=5/1280, ttl=64 (reply in 2)
2	0.000013873	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=5/1280, ttl=64 (request in 1)
3	1.024351825	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=6/1536, ttl=64 (reply in 4)
4	1.024360633	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=6/1536, ttl=64 (request in 3)
5	2.051608366	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=7/1792, ttl=64 (reply in 6)
6	2.051621825	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=7/1792, ttl=64 (request in 5)
7	3.071837837	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=8/2048, ttl=64 (reply in 8)
8	3.071851419	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=8/2048, ttl=64 (request in 7)
9	4.096004801	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=9/2304, ttl=64 (reply in 10)
10	4.096018233	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=9/2304, ttl=64 (request in 9)
11	5.120668755	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=10/2560, ttl=64 (reply in 12)
12	5.120683140	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=10/2560, ttl=64 (request in 11)
13	6.147541528	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=11/2816, ttl=64 (reply in 14)

```
▸ Frame 2: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▾ Linux cooked capture v1
    Packet type: Unicast to us (0)
    Link-layer address type: Loopback (772)
    Link-layer address length: 6
    Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Unused: 0000
    Protocol: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.6.56, Dst: 10.0.6.56
```

```
▸ Frame 2: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▸ Linux cooked capture v1
▾ Internet Protocol Version 4, Src: 10.0.6.56, Dst: 10.0.6.56
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x72f4 (29428)
  ▸ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xe745 [validation disabled]
    [Header checksum status: Unverified]
```

First echo reply:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=5/1280, ttl=64 (reply in 2)
2	0.000013873	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=5/1280, ttl=64 (request in 1)
3	1.024351825	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=6/1536, ttl=64 (reply in 4)
4	1.024360633	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=6/1536, ttl=64 (request in 3)
5	2.051608366	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=7/1792, ttl=64 (reply in 6)
6	2.051621825	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=7/1792, ttl=64 (request in 5)
7	3.071837837	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=8/2048, ttl=64 (reply in 8)
8	3.071851419	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=8/2048, ttl=64 (request in 7)
9	4.096004801	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=9/2304, ttl=64 (reply in 10)
10	4.096018233	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=9/2304, ttl=64 (request in 9)
11	5.120668755	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=10/2560, ttl=64 (reply in 12)
12	5.120683140	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) reply id=0x000d, seq=10/2560, ttl=64 (request in 11)
13	6.147541528	10.0.6.56	10.0.6.56	ICMP	100	Echo (ping) request id=0x000d, seq=11/2816, ttl=64 (reply in 14)

```
▸ Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▾ Linux cooked capture v1
    Packet type: Unicast to us (0)
    Link-layer address type: Loopback (772)
    Link-layer address length: 6
    Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Unused: 0000
    Protocol: IPv4 (0x0800)
```

```
▸ Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▸ Linux cooked capture v1
▾ Internet Protocol Version 4, Src: 10.0.6.56, Dst: 10.0.6.56
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x72f3 (29427)
  ▸ Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xa746 [validation disabled]
    [Header checksum status: Unverified]
```

Details	First Echo Request	First Echo Reply
Frame Number	1	2
Source IP address	10.0.6.56	10.0.6.56
Destination IP address	10.0.6.56	10.0.6.56
ICMP Type Value	8	0

ICMP Code Value	0	0
Source Ethernet Address	00:00:00:00:00:00	00:00:00:00:00:00
Destination Ethernet Address	00:00:00:00:00:00	00:00:00:00:00:00
Internet Protocol Version	4	4
Time To Live (TTL) Value	64	64

Task 3: HTTP PDU Capture

Using Wireshark's Filter feature

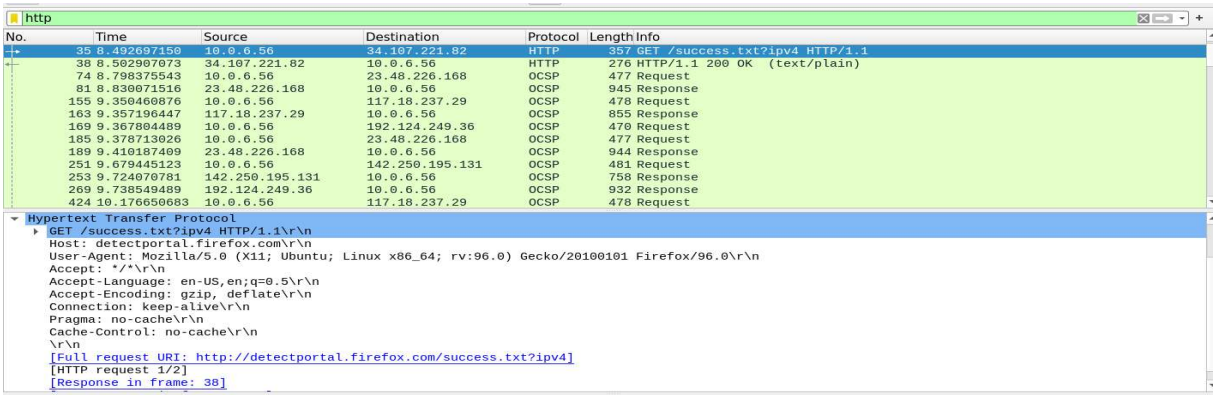
Step 1: Launch Wireshark and select 'any' interface. On the Filter toolbar, type-in 'http' and press enter

Step 2: Open Firefox browser, and browse www.flipkart.com

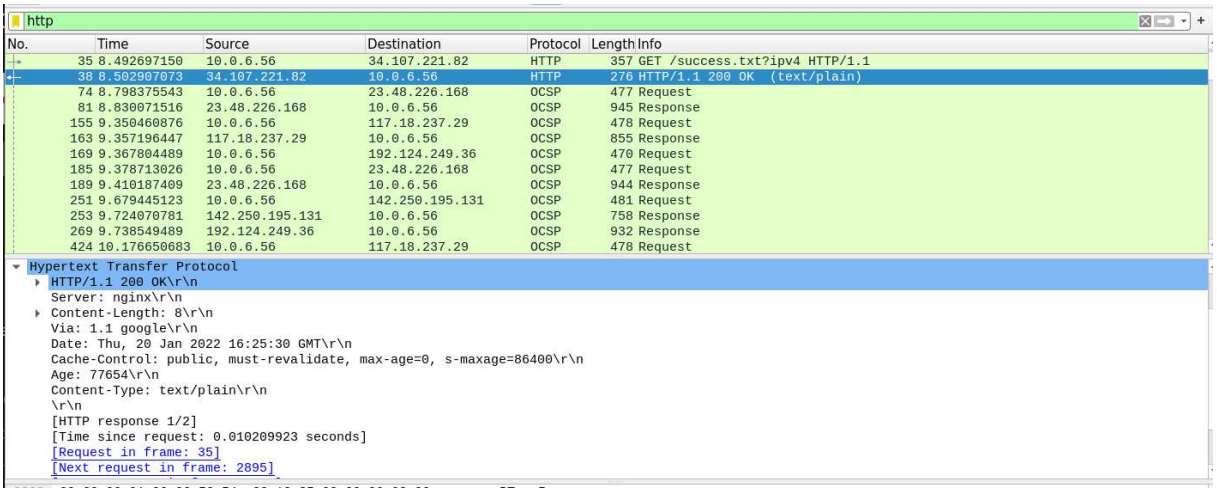
Observations to be made

Step 3: Analyze the first (interaction of host to the web server) and second frame (response of server to the client). By analyzing the filtered frames, complete the table below:

HTTP First Request:



HTTP First Response:



Details	First Echo Request	First Echo Reply
Frame Number	35	38
Source Port	38282	80
Destination Port	80	38282
Source IP address	10.0.6.56	34.107.221.82
Destination IP address	34.107.221.82	10.0.6.56
Source Ethernet Address	08:00:27:4f:0b:16	52:54:00:12:35:02
Destination Ethernet Address	52:54:00:12:35:02	08:00:27:4f:0b:16

Step 4: Analyze the HTTP request and response and complete the table below.

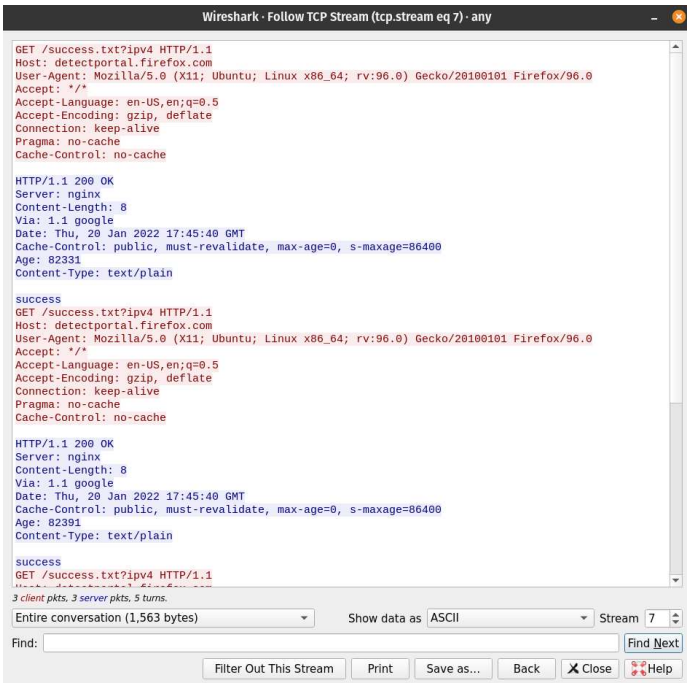
HTTP Request		HTTP Response	
Get	GET /Success.txt?ipv4 HTTP/1.1	Server	nginx\r\n
Host	detectportal.firefox\r\n	Content-Type	text/plain\r\n

User-Agent	**Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:96.0) Gecko/20100101 Firefox/96.0\r\n**	Date	**Thu, 20 Jan 2022 16:25:30 GMT\r\n**
Accept-Language	**en-US,en;q=0.5\r\n**	Location	**-**
Accept-Encoding	**gzip, deflate\r\n**	Content-Length	**8\r\n**
Connection	**keep-alive\r\n**	Connection	**close\r\n**

Using Wireshark's Follow TCP Stream

Step 1: Make sure the filter is blank. Right-click any packet inside the Packet List Pane, then select 'Follow TCP Stream'. For demo purpose, a packet containing the HTTP GET request "GET / HTTP / 1.1" can be selected.

Step 2: Upon following a TCP stream, screenshot the whole window.



Task 4: Capturing packets with tcpdump

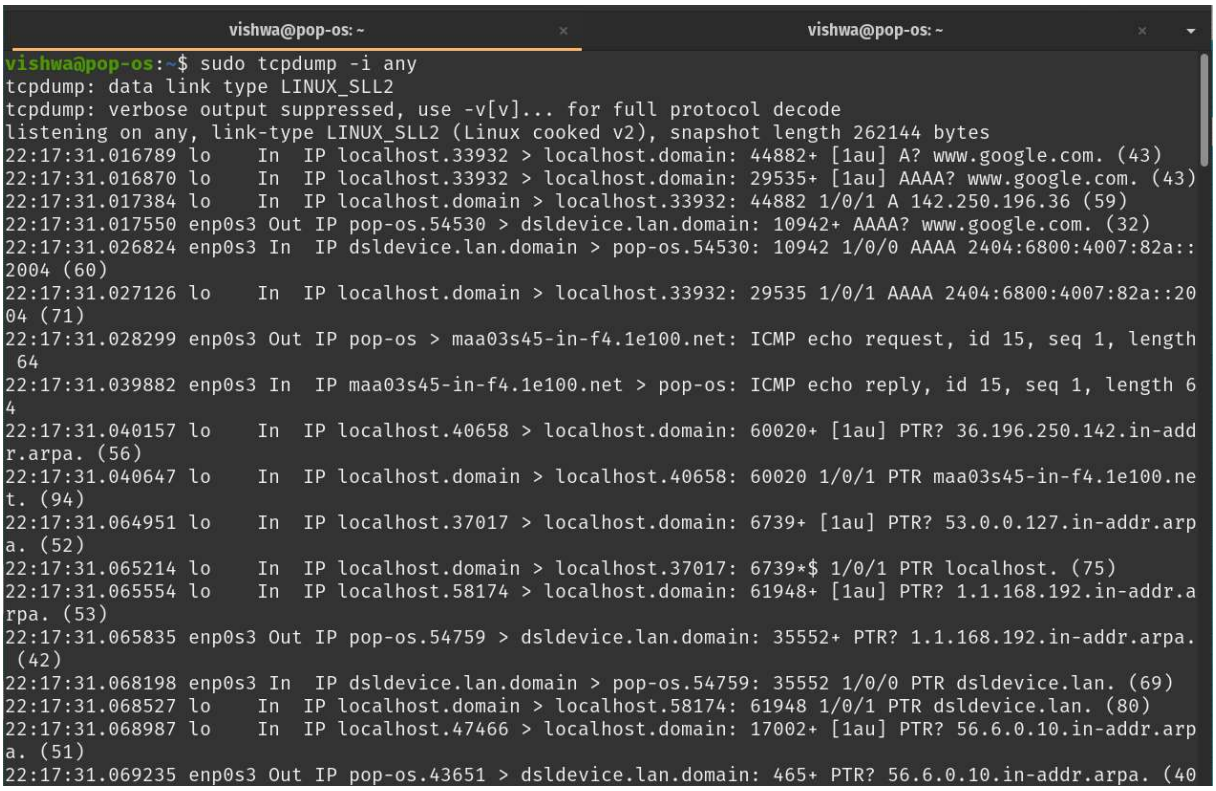
Step 1: Use the command **tcpdump -D** to see which interfaces are available for capture. **sudo tcpdump -D**

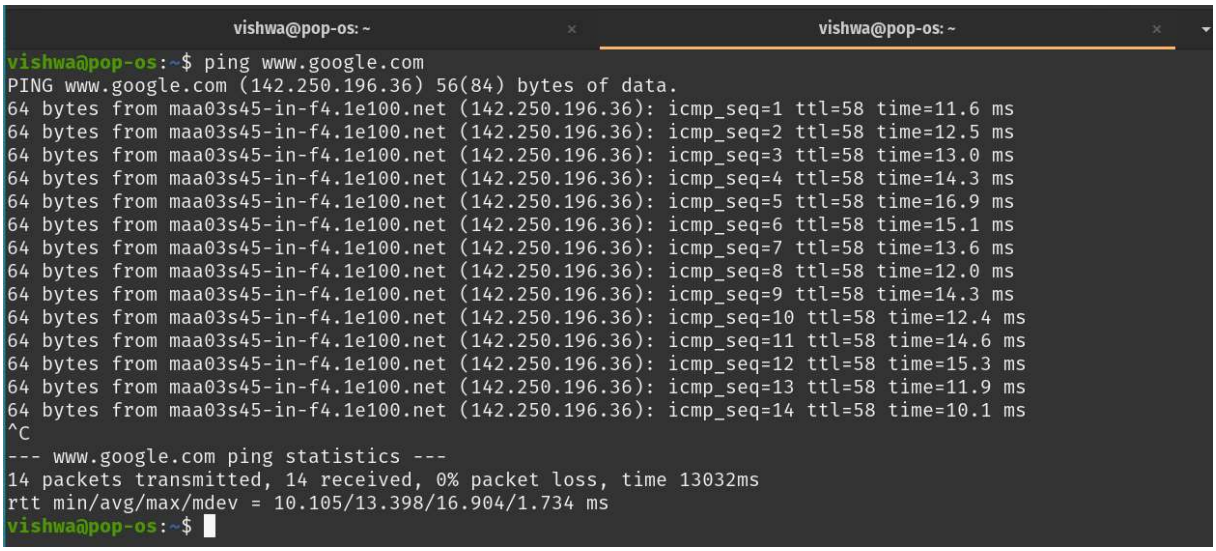
```
vishwa@pop-os:~$ sudo tcpdump -D
[sudo] password for vishwa:
1.enp0s3 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
vishwa@pop-os:~$
```

Step 2: Capture all packets in any interface by running this command:

sudo tcpdump -i any

Note: Perform some pinging operation while giving above command. Also type www.google.com in browser.





Observation

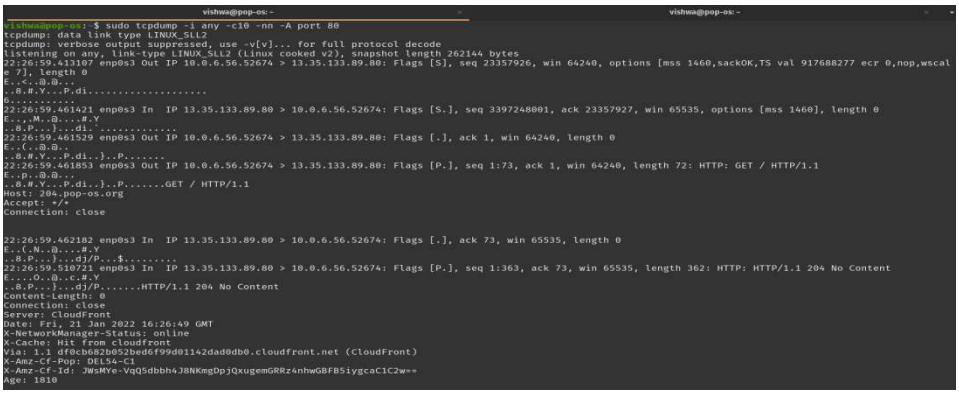
Step 3: Understand the output format.

Step 4: To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:

sudo tcpdump -i any -c5 icmp

```
vishwa@pop-os: ~                                    vishwa@pop-os: ~
vishwa@pop-os:~$ sudo tcpdump -i any -c5 icmp
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
22:22:56.397741 enp0s3 Out IP pop-os > maa03s44-in-f4.1e100.net: ICMP echo request, id 16, seq 1, length
 64
22:22:56.409641 enp0s3 In  IP maa03s44-in-f4.1e100.net > pop-os: ICMP echo reply, id 16, seq 1, length 6
4
22:22:57.399105 enp0s3 Out IP pop-os > maa03s44-in-f4.1e100.net: ICMP echo request, id 16, seq 2, length
 64
22:22:57.412295 enp0s3 In  IP maa03s44-in-f4.1e100.net > pop-os: ICMP echo reply, id 16, seq 2, length 6
4
22:22:58.401853 enp0s3 Out IP pop-os > maa03s44-in-f4.1e100.net: ICMP echo request, id 16, seq 3, length
 64
5 packets captured
6 packets received by filter
0 packets dropped by kernel
vishwa@pop-os:~$
```

Step 5: Check the packet content. For example, inspect the HTTP content of a web request like this: **sudo tcpdump -i any -c10 -nn -A port 80**




```
22:26:59.510745 enp0s3 Out IP 10.0.6.56.52674 > 13.35.133.89.80: Flags [.], ack 363, win 63878, length 0
E..(..@.@...
..8.#.Y...P.dj/.}.lP.......
22:26:59.510932 enp0s3 In  IP 13.35.133.89.80 > 10.0.6.56.52674: Flags [F.], seq 363, ack 73, win 65535, length 0
E..(.P..@.....#.Y
..8.P...}.l.dj/P...#.........
22:26:59.511784 enp0s3 Out IP 10.0.6.56.52674 > 13.35.133.89.80: Flags [F.], seq 73, ack 364, win 63878, length 0
E..(..@.@...
..8.#.Y...P.dj/.}.mP.......
22:26:59.512440 enp0s3 In  IP 13.35.133.89.80 > 10.0.6.56.52674: Flags [.], ack 74, win 65535, length 0
E..(.Q..@.....#.Y
..8.P...}.m.dj0P...#.........
10 packets captured
10 packets received by filter
0 packets dropped by kernel
vishwa@pop-os:~$
```

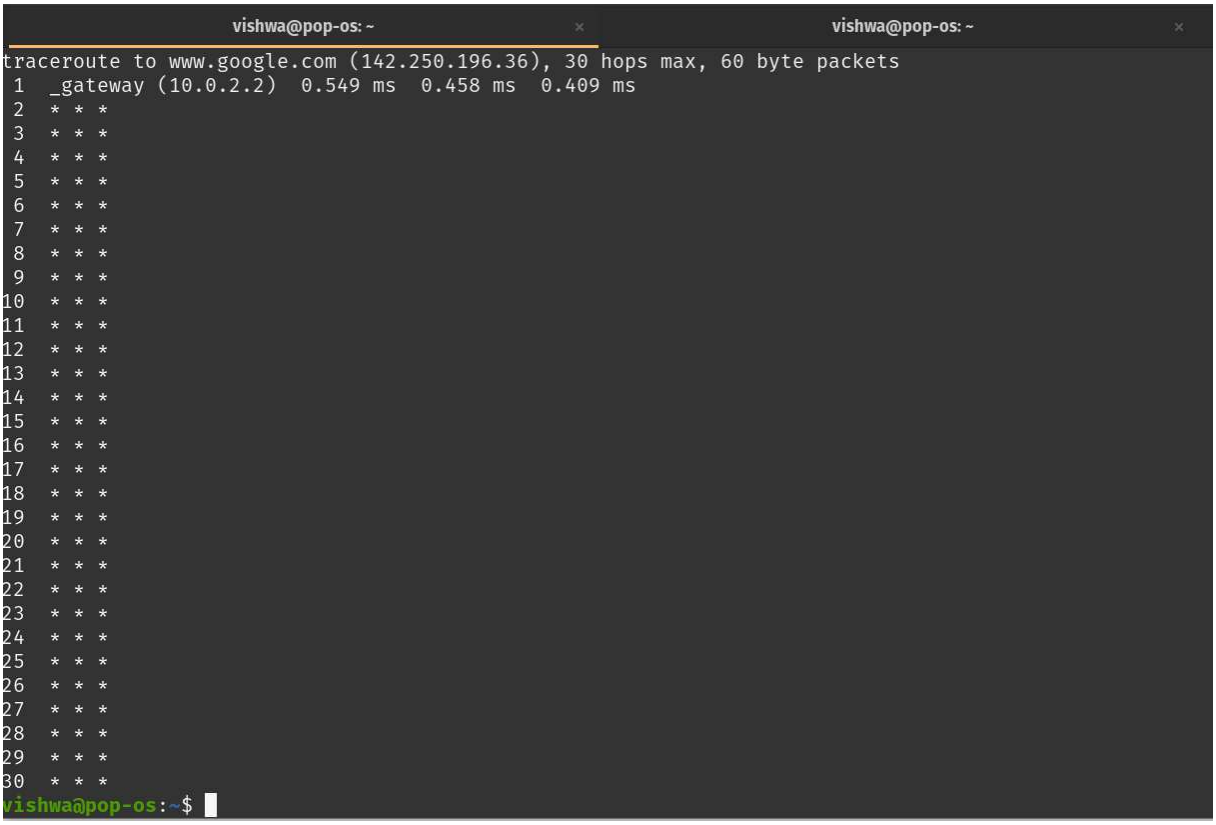
Step 6: To save packets to a file instead of displaying them on screen, use the option -w: **sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80**

```
vishwa@pop-os: ~
vishwa@pop-os:~$ sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
10 packets captured
10 packets received by filter
0 packets dropped by kernel
vishwa@pop-os:~$
```

Task 5: Perform Traceroute checks

Step 1: Run the traceroute using the following command.

sudo traceroute www.google.com

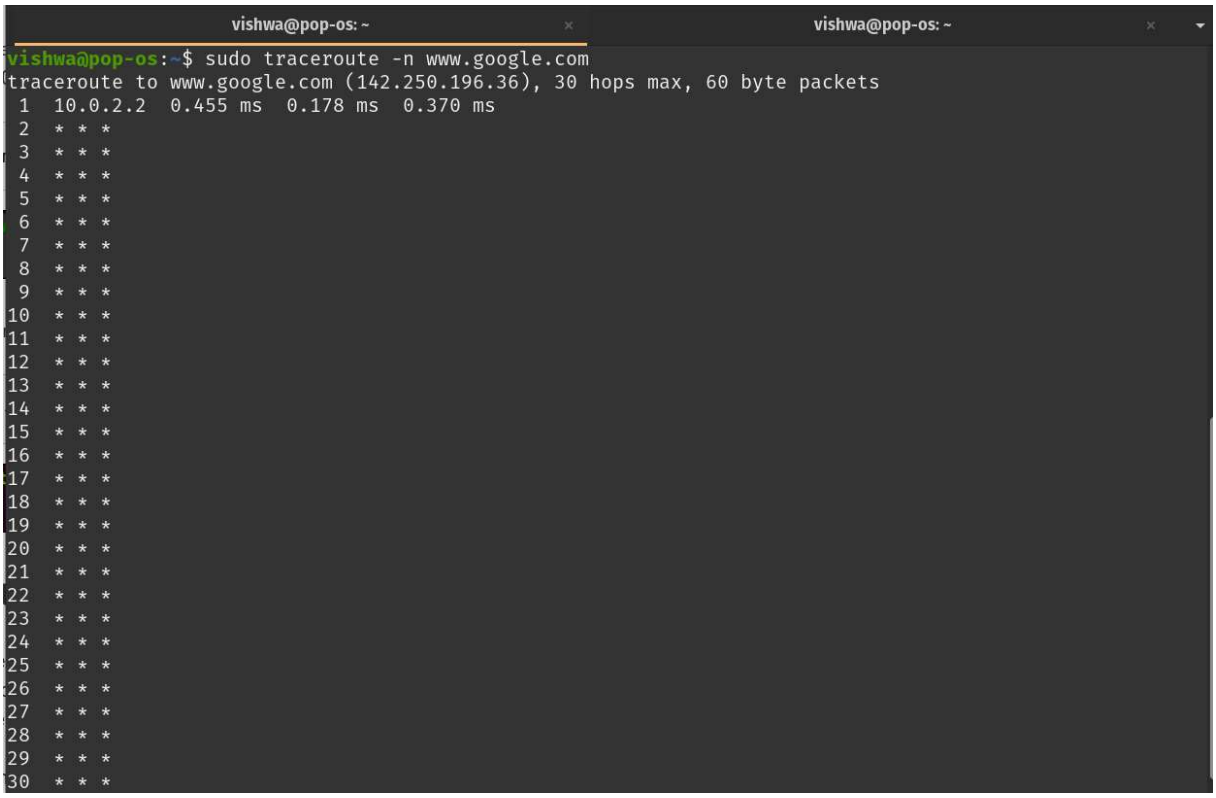


Step 2: Analyze destination address of google.com and no. of hops

Destination: 142.250.196.36 No. of hops: 30

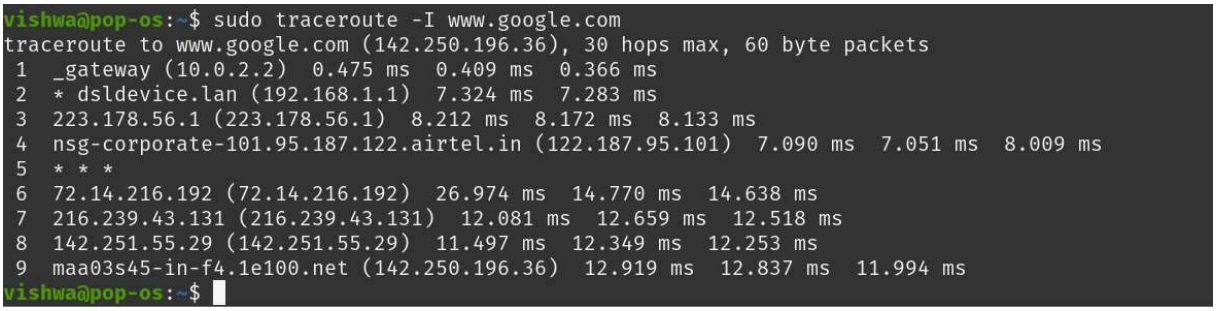
Step 3: To speed up the process, you can disable the mapping of IP addresses with hostnames by using the ***-n*** option

sudo traceroute -n www.google.com



Step 4: The -I option is necessary so that the traceroute uses ICMP.

sudo traceroute -I www.google.com



Step 5: By default, traceroute uses icmp (ping) packets. If you'd rather test a TCP connection to gather data more relevant to web server, you can use the -T flag.

sudo traceroute -T www.google.com

Task 6: Explore an entire network for information (Nmap)

Step 1: You can scan a host using its host name or IP address, for instance. **nmap www.pes.edu**

```
vishwa@pop-os:~$ nmap www.pes.edu
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-21 22:53 IST
Nmap scan report for www.pes.edu (52.172.204.196)
Host is up (0.032s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 10.52 seconds
vishwa@pop-os:~$
```

Step 2: Alternatively, use an IP address to scan.

nmap 163.53.78.128

```
vishwa@pop-os:~$ nmap 163.53.78.128
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-21 22:54 IST
Nmap scan report for 163.53.78.128
Host is up (0.014s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 6.32 seconds
vishwa@pop-os:~$
```

Step 3: Scan multiple IP address or subnet (IPv4)

nmap 192.168.1.1 192.168.1.2 192.168.1.3

```
vishwa@pop-os:~$ nmap 192.168.1.1 192.168.1.2 192.168.1.3
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-21 22:55 IST
Nmap scan report for dsldevice.lan (192.168.1.1)
Host is up (0.0037s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 3 IP addresses (1 host up) scanned in 150.48 seconds
vishwa@pop-os:~$
```

TASK 7 A): NETCAT AS CHAT TOOL

a) Intra system communication (Using 2 terminals in the same system)

Step 1: Open a terminal (Ctrl+Alt+T). This will act as a Server.

Step 2: Type nc -l any_portnum (For eg., nc -l 1234)

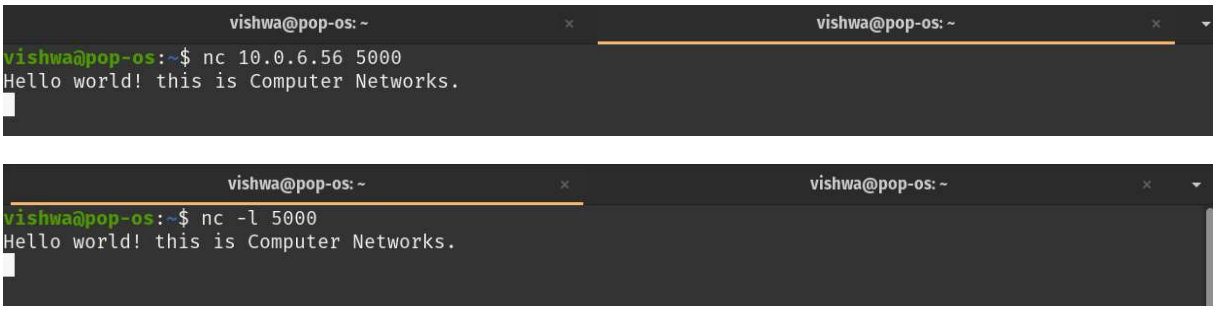
Note: It will goto listening mode

Step 3: Open another terminal and this will act as a client.

Step 4: Type nc <your-system-ip-address> portnum

Note: portnum should be common in both the terminals (for eg., nc 10.0.2.8 1234)

Step 5: Type anything in client will appear in server



Questions on above observations:

1) Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server?

HTTP 1.1 for browser and server.

2) When was the HTML file that you are retrieving last modified at the server?

Thu, 20 Jan 2022 16:25:30 GMT\r\n

3) How to tell ping to exit after a specified number of ECHO_REQUEST packets?

ping ip –v no_of_pings

4) How will you identify remote host apps and OS?

nmap –O –v ip_addr