Computer Networks Lab

Week 1

Vishal R

PES1UG19CS571

I Section

January 27,2021

1 Linux Interface Configuration

1.1 Display status of all active network interfaces

Using `ip addr show` or `ifconfig` command we can list all the interfaces avaliable

```
vishalr@ubuntu:~$ 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:a9:22:d8 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.0.128/24 brd 192.168.0.255 scope global dynamic noprefixroute ens33
       valid_lft 85679sec preferred_lft 85679sec
    inet6 fe80::6030:541e:83ca:9ccd/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enx00e04c534458: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:e0:4c:53:44:58 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.114/24 brd 192.168.0.255 scope global dynamic noprefixroute enx00e04c534458
       valid_lft 86389sec preferred_lft 86389sec
    inet6 fe80::4505:d8e8:87d4:dd82/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
vishalr@ubuntu:~$
```

Figure 1: The above figure shows all the available network interfaces.

IP Address Table :

Interface Name	IP Address (IPv4 / IPv6)	MAC Address
lo	127.0.0.1/::1	00:00:00:00:00:00
ens33	192.168.0.128/fe80::6030:541e:83ca:9ccd	00:0c:29:a9:22:d8
enx00e04c534458	192.168.0.114/fe80::4505:d8e8:87d4:dd82	00:e0:4c:53:44:58

Note : `enx00e04c534458` is also known as `eth0` in some systems.

1.2 Assigning IP Address to an interface

Since DHCP assigns an unique IP Address to each system in a network, we need not execute this step. The IP Address assigned by DHCP to my system is `192.168.0.128`

1.3 Activating and Deactivating a network interface

To deactivate a network interface we execute `$ sudo ifconfig interface_name down` and to activate a network interface we execute `$ sudo ifconfig interface_name up`

```
vishalr@ubuntu:~$ sudo ifconfig lo down
vishalr@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.128  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6030:541e:83ca:9ccd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a9:22:d8  txqueuelen 1000  (Ethernet)
        RX packets 6285  bytes 7938158 (7.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2637  bytes 312465 (312.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enx00e04c534458: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.114  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::4505:d8e8:87d4:dd82  prefixlen 64  scopeid 0x20<link>
        ether 00:e0:4c:53:44:58  txqueuelen 1000  (Ethernet)
        RX packets 118  bytes 6747 (6.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 151  bytes 10944 (10.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vishalr@ubuntu:~$
```

Figure 2: Deactivating `lo` interface and checking for available network interfaces using `ifconfig`.

From the above figure, `lo` interface has been deactivated on executing `$ sudo ifconfig lo down`. This can be verified by typing `ifconfig`. We can see that `lo` has not been listed in the list of available network interfaces after executing `ifconfig` command.

```
vishalr@ubuntu:~$ sudo ifconfig lo up
vishalr@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.128  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6030:541e:83ca:9ccd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a9:22:d8  txqueuelen 1000  (Ethernet)
        RX packets 6328  bytes 7947303 (7.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2677  bytes 316918 (316.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enx00e04c534458: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.114  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::4505:d8e8:87d4:dd82  prefixlen 64  scopeid 0x20<link>
        ether 00:e0:4c:53:44:58  txqueuelen 1000  (Ethernet)
        RX packets 151  bytes 8451 (8.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 187  bytes 12892 (12.8 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 705  bytes 68827 (68.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 705  bytes 68827 (68.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vishalr@ubuntu:~$
```

Figure 3: Activating `lo` interface and checking for available network interfaces using `ifconfig`.

In the above figure, I have executed the command `$ sudo ifconfig lo up`. This command activates the interface `lo`. We can verify it by using `ifconfig`. In the above figure we can see that `lo` interface has been listed in the list of available network interfaces.

We can execute this command for any network interface. For this experiment, I have used only `lo` interface.

1.4 To show the neighbor table in the kernel

We can show the neighbor table by typing the command `ip neigh`.

```
vishalr@ubuntu:~$ ip neigh
192.168.0.1 dev ens33 lladdr 50:2b:73:2c:29:20 REACHABLE
192.168.0.1 dev enx00e04c534458 lladdr 50:2b:73:2c:29:20 STALE
vishalr@ubuntu:~$
```

Figure 4: Neighbor Table in kernel.

The above picture shows all the interfaces in the neighbor table.
`REACHABLE` - indicates that the neighbor entry is valid until the reachability timeout expires.
`STALE` - indicates that the neighbour entry is valid but suspicious.

2 Ping PDU (Packet Data Units) Capture

2.1 Assign IP Address to the system.

As mentioned earlier, we will be using the IP Address assigned by DHCP as it will be unique to all systems in a network.

2.2 Launch Wireshark and select 'any' interface.

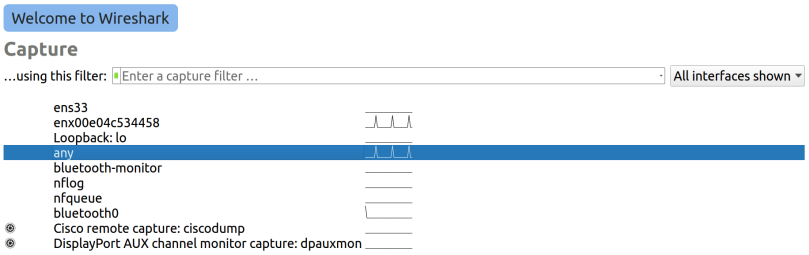


Figure 5: Selecting any interface in wireshark

In the above figure, we have opened wireshark for PDU Capture and we will select 'any' interface to capture packets.

2.3 Ping to `192.168.0.128`.

We can ping to a server or website by typing the command `ping ip_address`.

```
vishalr@ubuntu:~$ ping 192.168.0.128
PING 192.168.0.128 (192.168.0.128) 56(84) bytes of data.
64 bytes from 192.168.0.128: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 192.168.0.128: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 192.168.0.128: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 192.168.0.128: icmp_seq=4 ttl=64 time=0.034 ms
64 bytes from 192.168.0.128: icmp_seq=5 ttl=64 time=0.034 ms
64 bytes from 192.168.0.128: icmp_seq=6 ttl=64 time=0.036 ms
64 bytes from 192.168.0.128: icmp_seq=7 ttl=64 time=0.035 ms
^C
--- 192.168.0.128 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6131ms
rtt min/avg/max/mdev = 0.024/0.032/0.036/0.003 ms
vishalr@ubuntu:~$
```

Figure 6: Performing ping operation to `192.168.0.128`.

In the above figure, I have pinged to my local machine (`192.168.0.128`). From the terminal, we can make the following observations :

TTL	64
Protocol used by ping	ICMP
Time	0.032 ms

2.4 Additional Analysis on Wireshark

We will select the first echo request packet in wireshark.

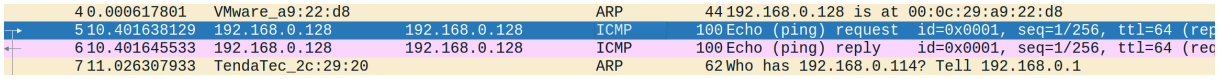


Figure 7: Selecting first echo ping request in wireshark.

Expanding the four tabs we can get additional information about the packets.

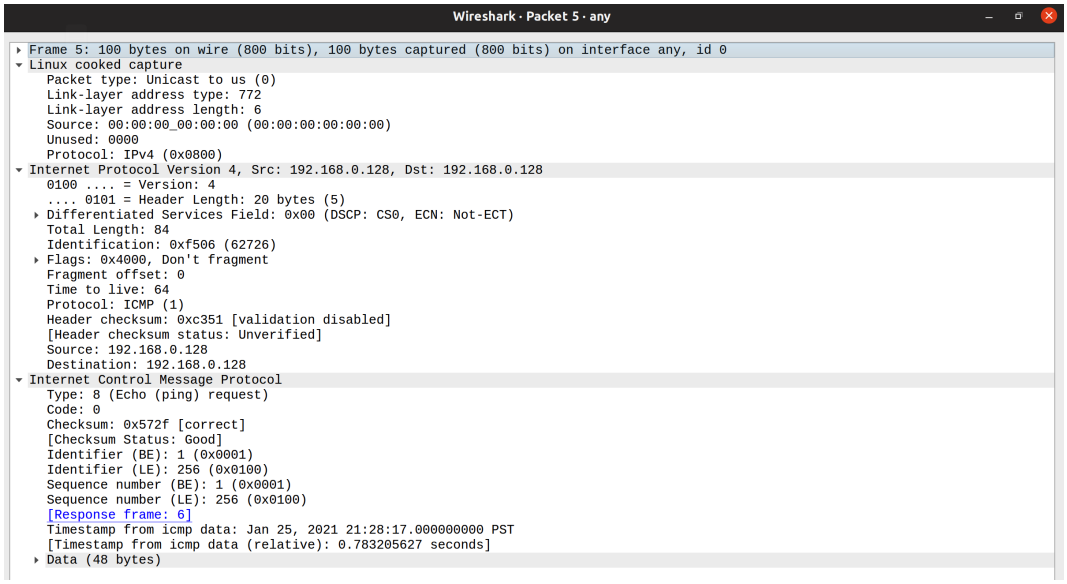


Figure 8: Expanding the four layer tabs, we can get additional information about the ping request we made to `192.168.0.128`.

We will select the first echo reply packet in wireshark.

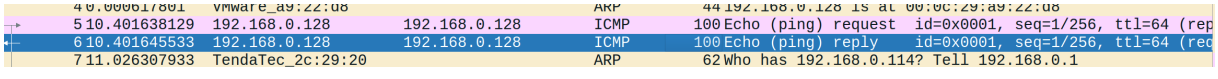


Figure 9: Selecting first echo ping reply in wireshark.

Again, expanding the four tabs we can get additional information about the packets.

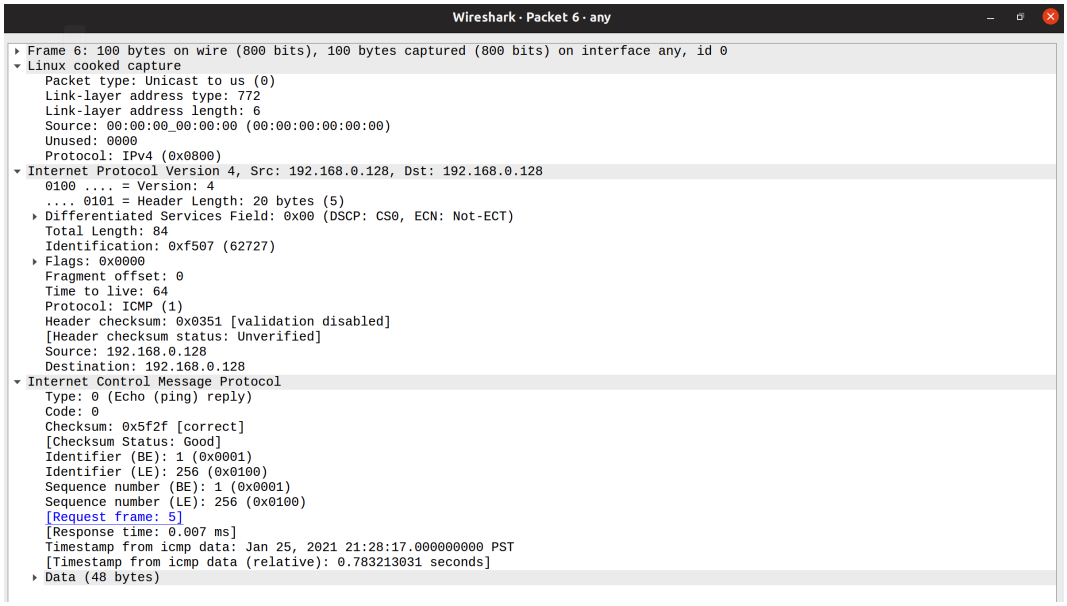


Figure 10: Expanding the four layer tabs, we can get additional information about the ping reply we got from `192.168.0.128`.

From the above figures, we can make the following observations :

Details	First Echo Request	First Echo Reply
Frame Number	5	6
Source IP Address	192.168.0.128	192.168.0.128
Destination IP Address	192.168.0.128	192.168.0.128
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	IPv4	IPv4
Time To Live (TTL)	64	64

3 HTTP Packet Data Unit Capture

3.1 Using Wireshark's filter feature

We will use wireshark's filter feature to observe only HTTP Packets. To achieve this, we will open wireshark and observe packets in 'any' interface.
In the filter options, we will type or select `http` so that we can filter out only http packets to make things easier.

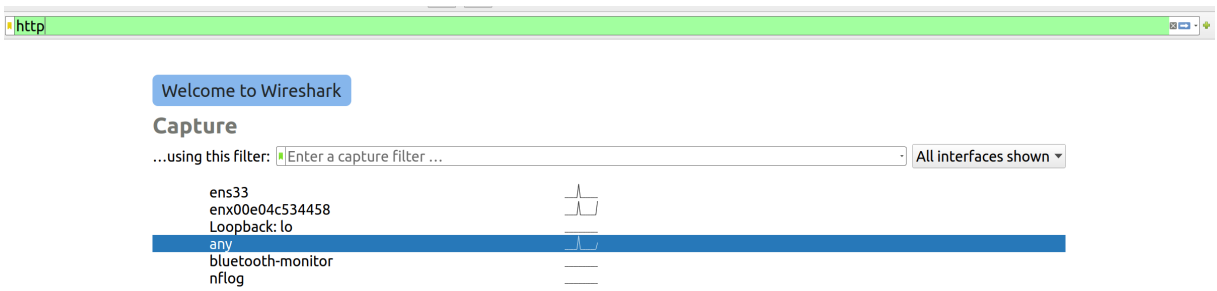


Figure 11: Selecting 'any' interface with http filter enabled.

We will open a web browser and browse flipkart. We will come back to wireshark to observe the HTTP Packets we got from flipkart.

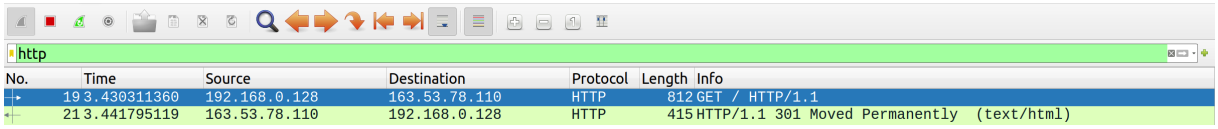


Figure 12: HTTP Packets received from flipkart.

We have received two packets from flipkart. The following observations have been made by opening the four tabs in wireshark.

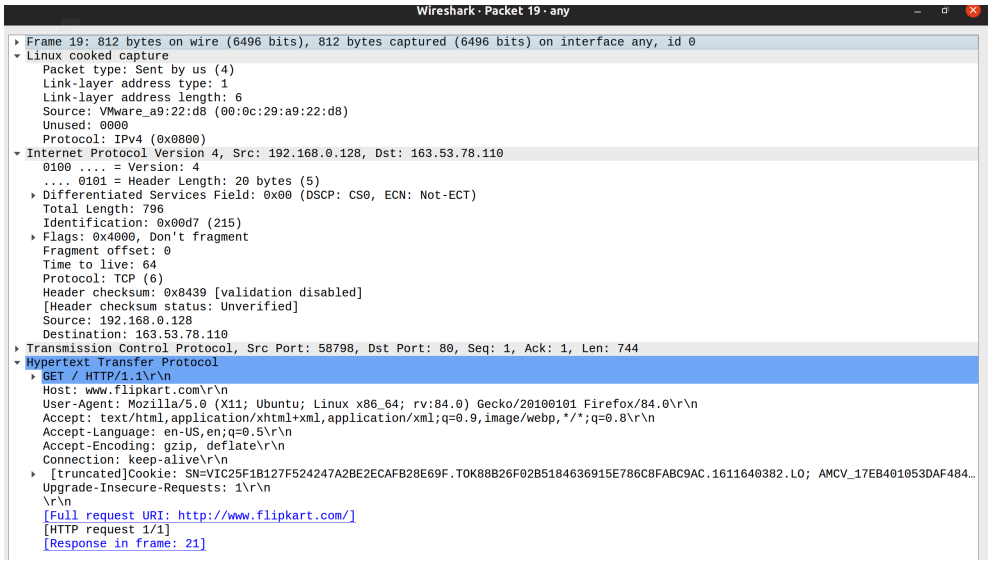


Figure 13: HTTP Get Request Packet Data.

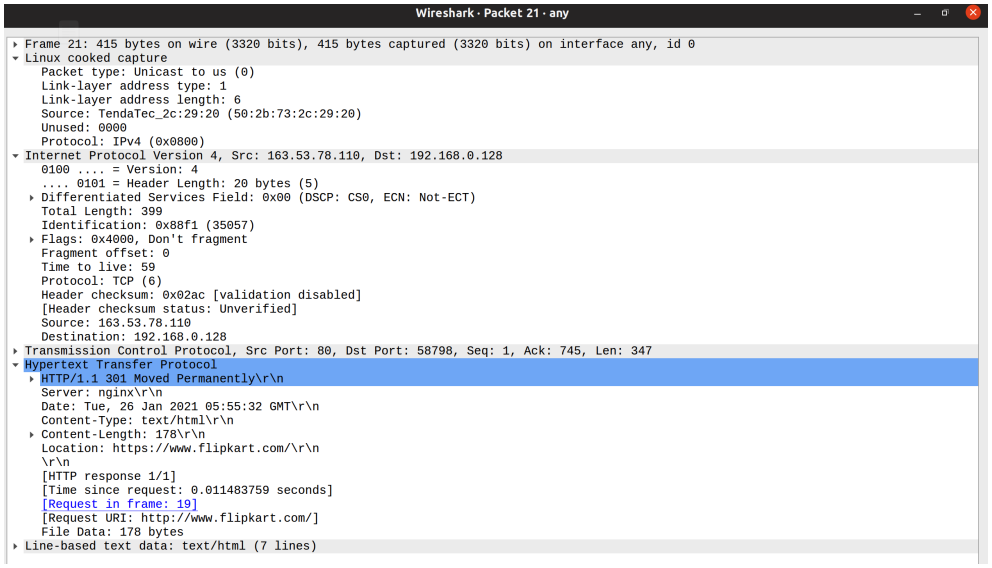


Figure 14: HTTP Response Packet Data.

From the packets we received from flipkart, we can make the following observations :

Details	First Echo Request	First Echo Reply
Frame Number	19	21
Source Port	58798	192.168.0.128
Destination Port	80	192.168.0.128
Source IP Address	192.168.0.128	192.168.0.128
Destination IP Address	163.53.78.110	192.168.0.128
Source Ethernet Address	00:0c:29:a9:22:d8	50:2b:73:2c:29:20
Destination Ethernet Address	50:2b:73:2c:29:20	00:0c:29:a9:22:d8

Additional analysis :

HTTP Request		**HTTP Response**	
GET	GET / HTTP/1.1	**Server**	nginx
Host	www.flipkart.com	**Content-Type**	text/html
User-Agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0	**Date**	Tue, 26 Jan 2021 05:55:32 GMT
Accept-Language	en-US,en;q=0.5	**Location**	https://www.flipkart.com/
Accept-Encoding	gzip, deflate	**Content-Length**	178
Connection	keep-alive	**Connection**	keep-alive

3.2 Using Wireshark's Follow TCP Stream

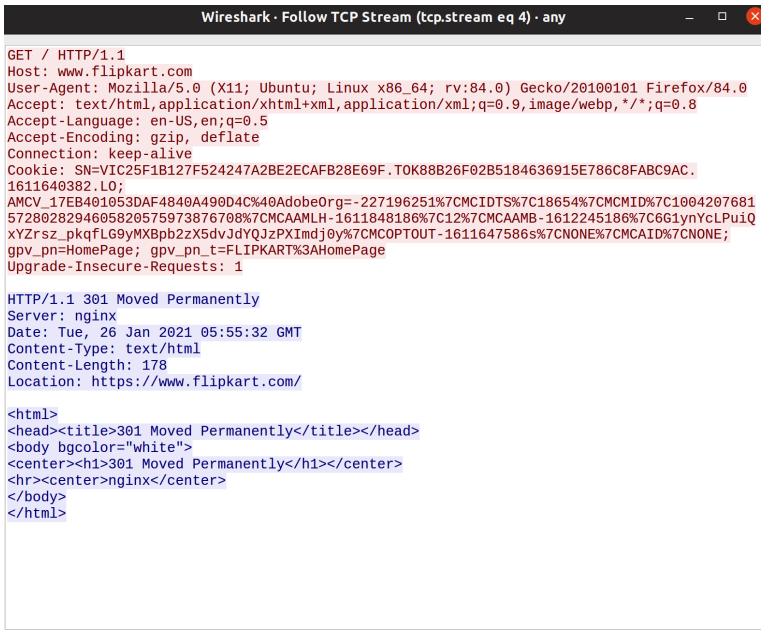


Figure 15: TCP Stream of GET / HTTP/1.1 Packet.

The above figure is a screen shot of what I got after clicking Follow TCP stream.

4 Capturing packets with `tcpdump`

4.1 Check the avaliable interfaces for Capture

We can check the available interface for capture by typing the command `$ sudo tcpdump -D`.

```
vishalr@ubuntu:~$ sudo tcpdump -D
1.ens33 [Up, Running]
2.enx00e04c534458 [Up, Running]
3.lo [Up, Running, Loopback]
4.any (Pseudo-device that captures on all interfaces) [Up, Running]
5.bluetooth-monitor (Bluetooth Linux Monitor) [none]
6.nflog (Linux netfilter log (NFLOG) interface) [none]
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
8.bluetooth0 (Bluetooth adapter number 0) [none]
vishalr@ubuntu:~$
```

Figure 16: Checking for available interfaces for capture using `tcpdump`.

4.2 Capturing all packets in 'any' interface

We can capture all the packets in 'any' interface by typing the command `$ sudo tcpdump -i any`.
Note : Pinging operations were performed while executing the above command and Google was browsed in the web browser.

```
vishalr@ubuntu:~$ sudo tcpdump -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
23:07:26.405201 IP ubuntu > broadband.actcorp.in: ICMP echo request, id 1, seq 43, length 64
23:07:26.405857 IP localhost.50940 > localhost.domain: 35132+ [1au] PTR? 192.145.51.106.in-addr.arpa. (56)
23:07:26.406076 IP ubuntu.40381 > broadband.actcorp.in.domain: 19706+ [1au] PTR? 192.145.51.106.in-addr.arpa. (56)
23:07:26.406384 IP localhost.domain > localhost.50940: 35132 1/0/1 PTR broadband.actcorp.in. (90)
23:07:26.406559 IP localhost.51821 > localhost.domain: 32883+ [1au] PTR? 128.0.168.192.in-addr.arpa. (55)
23:07:26.413020 IP localhost.45135 > localhost.domain: 19306+ [1au] PTR? 53.0.0.127.in-addr.arpa. (52)
23:07:26.415440 IP broadband.actcorp.in.domain > ubuntu.47718: 1084 1/0/1 PTR broadband.actcorp.in. (89)
23:07:26.415488 IP ubuntu > broadband.actcorp.in: ICMP ubuntu udp port 47718 unreachable, length 125
23:07:27.406495 IP ubuntu > broadband.actcorp.in: ICMP echo request, id 1, seq 44, length 64
23:07:27.409738 IP broadband.actcorp.in > ubuntu: ICMP echo reply, id 1, seq 44, length 64
23:07:28.408078 IP ubuntu > broadband.actcorp.in: ICMP echo request, id 1, seq 45, length 64
23:07:28.410168 IP broadband.actcorp.in > ubuntu: ICMP echo reply, id 1, seq 45, length 64
23:07:29.268262 ARP, Request who-has ubuntu tell _gateway, length 46
23:07:29.268295 ARP, Reply ubuntu is-at 00:e0:4c:53:44:58 (oui Unknown), length 28
23:07:29.268616 IP localhost.49840 > localhost.domain: 5173+ [1au] PTR? 114.0.168.192.in-addr.arpa. (55)
23:07:29.268953 ARP, Request who-has ubuntu tell _gateway, length 46
23:07:29.268964 ARP, Reply ubuntu is-at 00:0c:29:a9:22:d8 (oui Unknown), length 28
23:07:29.269227 IP ubuntu.40380 > broadband.actcorp.in.domain: 55597+ [1au] PTR? 114.0.168.192.in-addr.arpa. (55)
23:07:29.269454 IP ubuntu.45411 > broadband.actcorp.in.domain: 30761+ [1au] PTR? 114.0.168.192.in-addr.arpa. (55)
23:07:29.409571 IP ubuntu > broadband.actcorp.in: ICMP echo request, id 1, seq 46, length 64
23:07:29.412158 IP broadband.actcorp.in > ubuntu: ICMP echo reply, id 1, seq 46, length 64
^C
21 packets captured
66 packets received by filter
39 packets dropped by kernel
vishalr@ubuntu:~$
```

Figure 17: Capturing packets in `any` interface using `tcpdump`.

4.3 Filtering Packets based on Protocol

We can filter the packets based on protocol used by typing the command `$ sudo tcpdump -i any -c5 icmp`.
Here `-i any` allows us to use 'any' interface to capture packets. `-c5` says that 5 packets must be captured. `icmp` says that the captured packets must use `icmp` protocol.

Note : Pinging operations were performed while executing the above command and Google was browsed in the web browser.

```
vishalr@ubuntu:~$ sudo tcpdump -i any -c5 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
23:13:02.155783 IP ubuntu > broadband.actcorp.in: ICMP echo request, id 2, seq 10, length 64
23:13:02.158024 IP broadband.actcorp.in > ubuntu: ICMP echo reply, id 2, seq 10, length 64
23:13:02.159897 IP ubuntu > broadband.actcorp.in: ICMP ubuntu udp port 35459 unreachable, length 126
23:13:02.168670 IP ubuntu > broadband.actcorp.in: ICMP ubuntu udp port 59189 unreachable, length 125
23:13:03.157675 IP ubuntu > broadband.actcorp.in: ICMP echo request, id 2, seq 11, length 64
5 packets captured
5 packets received by filter
0 packets dropped by kernel
vishalr@ubuntu:~$
```

Figure 18: Capturing ICMP packets in `any` interface using `tcpdump`.

4.4 Checking Packet Contents using `tcpdump`.

We can inspect the HTTP content of a web request by typing the command `$ sudo tcpdump -i any -c10 -nn -A port 80`.
Here `-i any` allows us to use 'any' interface to capture packets. `-c10` says that 10 packets in total must be captured.
Note : Pinging operations were performed while executing the above command and Google was browsed in the web browser.

```
vishalr@ubuntu:~$ sudo tcpdump -i any -c10 -nn -A port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
23:16:09.282490 IP 192.168.0.128.39438 > 117.18.237.29.80: Flags [.], ack 3565215097, win 501, options [nop,nop,TS val 518424999 ecr
 997396747], length 0
E..4..@.@.......u......P..z
...y....#......
....;s..
23:16:09.286114 IP 117.18.237.29.80 > 192.168.0.128.39438: Flags [.], ack 1, win 147, options [nop,nop,TS val 997406983 ecr 51840324
9], length 0
E..4@...9..ru........P.....y..z......y.....
;s9...4.
23:16:19.522496 IP 192.168.0.128.39438 > 117.18.237.29.80: Flags [.], ack 1, win 501, options [nop,nop,TS val 518435239 ecr 99740698
3], length 0
E..4..@.@.......u......P..z
...y....#......
....;s9.
23:16:19.526445 IP 117.18.237.29.80 > 192.168.0.128.39438: Flags [.], ack 1, win 147, options [nop,nop,TS val 997417222 ecr 51840324
9], length 0
E..4E...9...u........P.....y..z......z.....
;sa...4.
23:16:24.210702 IP 192.168.0.128.39438 > 117.18.237.29.80: Flags [F.], seq 1, ack 1, win 501, options [nop,nop,TS val 518439927 ecr
997417222], length 0
E..4..@.@.......u......P..z....y....#......
....;sa.
23:16:24.215103 IP 117.18.237.29.80 > 192.168.0.128.39438: Flags [F.], seq 1, ack 2, win 147, options [nop,nop,TS val 997421911 ecr
518439927], length 0
E..4Fy..8...u........P.....y..z............
;ssW....
23:16:24.215154 IP 192.168.0.128.39438 > 117.18.237.29.80: Flags [.], ack 2, win 501, options [nop,nop,TS val 518439931 ecr 99742191
1], length 0
E..4..@.@.......u......P..z....z....#......
....;ssW
23:17:51.728353 IP 192.168.0.128.55796 > 34.107.221.82.80: Flags [S], seq 512535451, win 64240, options [mss 1460,sackOK,TS val 2698
741831 ecr 0,nop,wscale 7], length 0
E..<
.@.@.o.....k.R...P.....................
...G........
23:17:51.739279 IP 34.107.221.82.80 > 192.168.0.128.55796: Flags [S.], seq 3778289684, ack 512535452, win 65535, options [mss 1430,s
ackOK,TS val 1121329666 ecr 2698741831,nop,wscale 8], length 0
E..<.N..7...k.R.....P...4 .........Su.........
B.....G....
23:17:51.739335 IP 192.168.0.128.55796 > 34.107.221.82.80: Flags [.], ack 1, win 502, options [nop,nop,TS val 2698741842 ecr 1121329
666], length 0
E..4
.@.@.o.....k.R...P.....4 ............
...RB.".
10 packets captured
10 packets received by filter
0 packets dropped by kernel
vishalr@ubuntu:~$
```

Figure 19: Inspecting HTTP Content of a web request.

4.5 Saving packets to a file instead of displaying on screen using `tcpdump`.

Instead of displaying the packet contents on the screen, we can store them in a `.pcap` file by typing the command `$ sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80`.
Again, `-i any` allows us to use 'any' interface to capture packets. `-c10` says that 10 packets in total must be captured. `webserver.pcap` is the file in which the contents of the packet is stored.
Note : Pinging operations were performed while executing the above command and Google was browsed in the web browser.

```
vishalr@ubuntu:~$ sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
10 packets captured
14 packets received by filter
0 packets dropped by kernel
vishalr@ubuntu:~$
```

Figure 20: Inspecting HTTP Content of a web request and storing in `webserver.pcap` file using `tcpdump`.

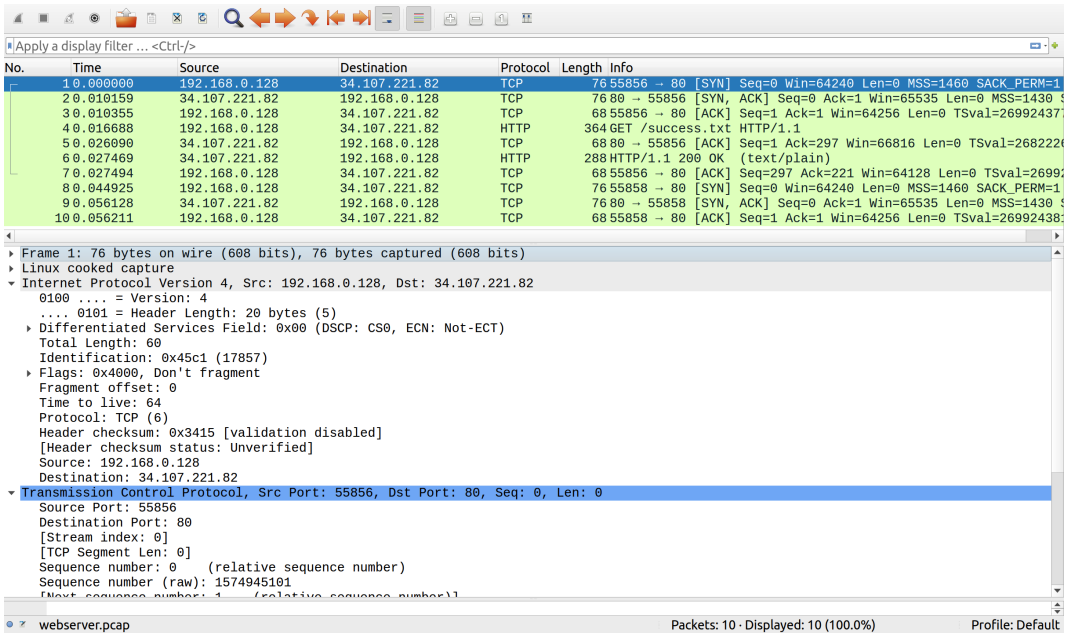


Figure 21: Contents of `webserver.pcap` viewed in wireshark.

From the above figure, we can see that the 10 packets captured by `tcpdump` has been saved in the file `webserver.pcap`.

5 Perform Traceroute Checks

5.1 Run `traceroute` command

`traceroute` is a command that is used to track in real-time the path taken by a packet from source to destination on an IP Network. It also reports all the IP addresses of all the routers it pinged in between the transmission.

We will now check the path taken by a packet to reach Google. This can be done by executing the command `$ sudo traceroute www.google.com`.

```
vishalr@ubuntu:~$ sudo traceroute www.google.com
traceroute to www.google.com (142.250.71.36), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  2.021 ms  1.866 ms  2.392 ms
 2  * * *
 3  14.142.183.201.static-Bangalore.vsnl.net.in (14.142.183.201)  4.691 ms  4.592 ms  4.496 ms
 4  * 172.29.209.114 (172.29.209.114)  9.769 ms 172.31.167.50 (172.31.167.50)  9.425 ms
 5  121.240.1.46 (121.240.1.46)  17.234 ms  11.597 ms  11.139 ms
 6  108.170.253.113 (108.170.253.113)  11.032 ms  10.093 ms 108.170.253.97 (108.170.253.97)  11.044 ms
 7  142.250.233.145 (142.250.233.145)  12.429 ms  10.690 ms 142.250.233.143 (142.250.233.143)  11.906 ms
 8  maa03s35-in-f4.1e100.net (142.250.71.36)  10.134 ms  11.456 ms  10.007 ms
vishalr@ubuntu:~$
```

Figure 22: Executing `traceroute` command to `www.google.com`.

5.2 Analyzing destination address of Google and number of hops taken.

From the above figure, we can see that the destination address of Google is `142.250.71.36` and the number of hops taken is 30.

5.3 Speeding up by disabling host mapping with `traceroute`.

We can speed up the tracing process by disabling the mapping of IP addresses with their host names.

This can be done by typing the following command `$ sudo traceroute -n www.google.com`.
`-n` option allows us to disable mapping of IP addresses with host names.

```
vishalr@ubuntu:~$ sudo traceroute -n www.google.com
traceroute to www.google.com (142.250.71.36), 30 hops max, 60 byte packets
 1  192.168.0.1  1.769 ms  1.685 ms  2.856 ms
 2  * * *
 3  14.142.183.201  4.807 ms  4.689 ms  4.627 ms
 4  * * *
 5  121.240.1.46  11.881 ms  13.775 ms  12.539 ms
 6  108.170.253.97  14.683 ms  14.027 ms 108.170.253.113  10.801 ms
 7  142.250.233.143  11.834 ms 142.250.233.145  9.566 ms 142.250.233.143  10.647 ms
 8  142.250.71.36  10.562 ms  11.204 ms  9.644 ms
vishalr@ubuntu:~$
```

Figure 23: Executing `traceroute` command to `www.google.com` with IP address and host name mapping disabled.

5.4 Makeing `traceroute` use ICMP.

We can make `traceroute` use ICMP protocol by using the following command `$ sudo traceroute -I www.google.com`. `-I` option allows or specifies `traceroute` to use only ICMP protocols.

```
vishalr@ubuntu:~$ sudo traceroute -I www.google.com
traceroute to www.google.com (142.250.71.36), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  2.146 ms  1.965 ms  1.837 ms
 2  10.240.0.1 (10.240.0.1)  3.088 ms  2.942 ms  2.803 ms
 3  14.142.183.201.static-Bangalore.vsnl.net.in (14.142.183.201)  4.447 ms  4.307 ms  4.188 ms
 4  * * *
 5  121.240.1.46 (121.240.1.46)  13.030 ms  12.904 ms  12.783 ms
 6  108.170.253.113 (108.170.253.113)  11.201 ms  11.115 ms  10.874 ms
 7  142.250.233.143 (142.250.233.143)  11.882 ms  12.029 ms  11.900 ms
 8  maa03s35-in-f4.1e100.net (142.250.71.36)  10.684 ms  10.613 ms  10.271 ms
vishalr@ubuntu:~$
```

Figure 24: Executing `traceroute` command to `www.google.com` and forcing it to use ICMP protocols only.

5.5 Makeing `traceroute` use TCP.

We can make `traceroute` use TCP protocol by using the following command `$ sudo traceroute -T www.google.com`. `-T` option allows or specifies `traceroute` to use only TCP protocols which will be useful to get data relevant to web server.

```
vishalr@ubuntu:~$ sudo traceroute -T www.google.com
traceroute to www.google.com (142.250.71.36), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  1.717 ms  1.621 ms  2.757 ms
 2  10.240.0.1 (10.240.0.1)  5.488 ms  5.437 ms  5.335 ms
 3  14.142.183.201.static-Bangalore.vsnl.net.in (14.142.183.201)  5.259 ms  5.118 ms  5.034 ms
 4  * * *
 5  121.240.1.46 (121.240.1.46)  13.764 ms  13.714 ms  14.796 ms
 6  108.170.253.97 (108.170.253.97)  13.548 ms  11.180 ms 108.170.253.113 (108.170.253.113)  9.727 ms
 7  142.250.233.145 (142.250.233.145)  9.897 ms  9.759 ms 142.250.233.143 (142.250.233.143)  13.542 ms
 8  maa03s35-in-f4.1e100.net (142.250.71.36)  10.109 ms  10.042 ms  9.943 ms
vishalr@ubuntu:~$
```

Figure 25: Executing `traceroute` command to `www.google.com` and forcing it to use TCP protocols only.

6 Exploring an entire network for information (`Nmap`)

`nmap` is not available by default. `nmap` can be installed using the command `$ sudo apt-get install nmap`. `nmap` is used to scan large networks rapidly and is used as an exploration tool and security / port scanner.

6.1 Scanning a host or IP Address using `nmap`.

We will scan PES University by using the command `sudo nmap www.pes.edu` and observe what we get.

```
vishalr@ubuntu:~$ sudo nmap www.pes.edu
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-25 23:52 PST
Nmap scan report for www.pes.edu (13.71.123.138)
Host is up (0.011s 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 4.96 seconds
vishalr@ubuntu:~$
```

Figure 26: Executing `sudo nmap www.pes.edu` command.

6.2 Using `nmap` to scan IP addresses

We will scan `163.53.78.128` by using the command `$ sudo nmap 163.53.78.128` and observe what we get.

```
vishalr@ubuntu:~$ sudo nmap 163.53.78.128
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-25 23:54 PST
Nmap scan report for 163.53.78.128
Host is up (0.0091s 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 5.10 seconds
vishalr@ubuntu:~$
```

Figure 27: Executing `nmap` command with an IP address to scan.

6.3 Using `nmap` to scan multiple IP address or subnet (IPv4)

We will scan `192.168.1.1 192.168.1.2 192.168.1.3` by using the command `$ sudo nmap 192.168.1.1 192.168.1.2 192.168.1.3` and observe what we get.

```
vishalr@ubuntu:~$ sudo nmap 192.168.1.1 192.168.1.2 192.168.1.3
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-25 23:57 PST
Nmap done: 3 IP addresses (0 hosts up) scanned in 5.12 seconds
vishalr@ubuntu:~$
```

Figure 28: Executing `nmap` command with multiple IP addresses to scan.

7 Netcat as Chat tool

7.1 Intra System Communication.

In the server's terminal, we will type the command `nc -l portnum`. `portnum` indicates the port on which the server must be hosted or listened. In my example I have used port `8080`.

Hence typing `nc -l 8080` makes the server go into listening mode.

```
vishalr@ubuntu:~$ nc -l 8080
```

Figure 29: Server in listening mode after executing the command `nc -l 8080`.

In another terminal, which acts as a client, we will type the following command `nc ipaddr portnum`. `portnum` must be the same in both terminals. In my example, we need to type `nc 192.168.0.128 8080`.

```
vishalr@ubuntu:~$ nc 192.168.0.128 8080
```

Figure 30: Client in listening mode after executing the command `nc 192.168.0.128 8080`.

Typing anything in the client will appear in the server as well as shown in the figure below.

```
vishalr@ubuntu:~$ nc 192.168.0.128 8080
This is a message from the client side.
Hello World!
^C
vishalr@ubuntu:~$
```

(a) Client Terminal

```
vishalr@ubuntu:~$ nc -l 8080
This is a message from the client side.
Hello World!
vishalr@ubuntu:~$
```

(b) Server Terminal

Figure 31: The above two figures shows the use of Netcat as a chat tool.

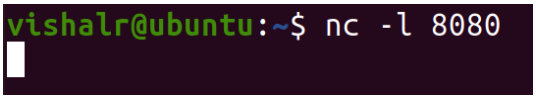
7.2 Inter System Communication

A simple switched network of 2 systems were setup with one acting as a Web Server. IP addresses were not assigned as the two systems will have their own unique IP address assigned by DHCP.

In server's terminal, type the following command `nc -l portnum`. This will make the server go into listening mode.

In client's terminal type the following command `nc server-ip-addr portnum`. This will make a connection with the server on the specified `portnum`. The client will go into listening mode and typing anything in client will appear in server as well.

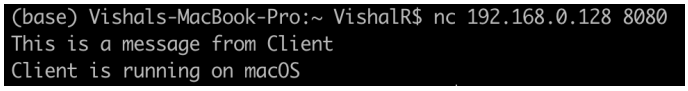
(a) Client Terminal on macOS



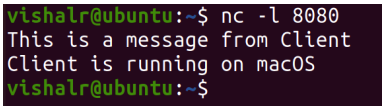
(b) Server Terminal on Ubuntu

Figure 32: (a) Client in listening mode on one system. (b) Server in listening mode on another system.

Since both client and server are in listening mode, we can conclude that they both are connected successfully.



(a) Client Terminal on macOS



(b) Server Terminal on Ubuntu

Figure 33: The above two figures shows the use of Netcat as an inter system communication tool.

7.3 Using Netcat to Transfer Files

Netcat utility can be used to transfer files from client to server.

At the server side, an empty file `test.txt` was created and the command `sudo nc -l 555 > test.txt` was executed in terminal. Here `555` is the port number.

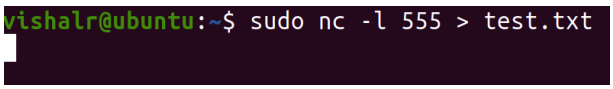


Figure 34: Server in listening mode after executing the command `sudo nc -l 555 > test.txt`.

At the client side, `testfile.txt` was created and some contents were added to the same. Then the command `sudo nc 192.168.0.128 555 < testfile.txt` was executed in the client's terminal. Here `192.168.0.128` is the server's IP address.

Figure 35: Executing the command `sudo nc 192.168.0.128 555 < testfile.txt` on client side (macOS)

After executing the above command, the contents of `testfile.txt` will be copied to `test.txt` and we can verify it by typing the command `cat test.txt` in server's terminal.

```
vishalr@ubuntu:~$ sudo nc -l 555 > test.txt
vishalr@ubuntu:~$ cat test.txt
Computer Networks Lab

Week1

PES1UG19CS571

Vishal R

Document created in LaTeX.
vishalr@ubuntu:~$
```

Figure 36: Verifying that the contents of `testfile.txt` on client side has been transfered to `test.txt` on server side.

This shows that Netcat can be used to transfer files from clients to servers.

7.4 Other Commands in Netcat

7.4.1 Testing if a particular TCP port of remote host is open.

We can check if a TCP port of a remote host is open or not by typing the command `nc -vn 192.168.0.128 555`. Here `555` is the port number. Before executing this command, we need to make sure that the server is running on the same port.

```
vishalr@ubuntu:~$ nc -vn 192.168.0.128 555
Connection to 192.168.0.128 555 port [tcp/*] succeeded!
^C
vishalr@ubuntu:~$
```

Figure 37: Executing the command `nc -vn 192.168.0.128 555` to test if a TCP port of a remote host is open.

Note : `nc -l 555` was executed to make the server listen on port `555`.

7.4.2 Run a web server with a static web page.

We can run a web server with a static web page using netcat. A sample html file called `test.html` was created at the server's side and some html content was added to it. We can make the server serve this html page by typing the command `while true; do sudo nc -lp 80 < test.html; done`. This will host the server on port 80.

```
vishalr@ubuntu:~$ while true; do sudo nc -lp 80 < test.html; done
```

Figure 38: Executing the command `while true; do sudo nc -lp 80 < test.html; done` to serve `test.html`

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Vishal R's blog page</h1>
6
7 <p>This page is hosted on the server running on port 80.</p>
8
9 </body>
10 </html>
11
12
```

Figure 39: Contents of `test.html`.

On another host, we will open `http://192.168.0.128/test.html` in the web browser.

Vishal R's blog page

This page is hosted on the server running on port 80.

Figure 40: `test.html` being accessed from another host.

We now will check our server's terminal.

```
vishalr@ubuntu:~$ while true; do sudo nc -lp 80 < test.html; done
GET /test.html HTTP/1.1
Host: 192.168.0.128
Connection: keep-alive
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_1_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Figure 41: Details of GET request made by the client being displayed on the server's terminal.

8 Questions on above Observations :

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

The browser is running HTTP version 1.1 . This can be verified by taking a look at the HTTP GET request we made to flipkart.

```
Destination: 163.53.78.110
▸ Transmission Control Protocol, Src Port: 58798, Dst Port: 80, Seq: 1, Ack: 1, Len: 744
▾ Hypertext Transfer Protocol
  ▸ GET / HTTP/1.1\r\n
    Host: www.flipkart.com\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
```

Figure 42: Shows the version of HTTP used to make GET request to flipkart.

The server also uses HTTP version 1.1 . We can verify this by seeing the HTTP response sent by flipkart back to the browser.

```
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 58798, Seq: 1, Ack: 745, Len: 347
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 301 Moved Permanently\r\n
    Server: nginx\r\n
    Date: Tue, 26 Jan 2021 05:55:32 GMT\r\n
    Content-Type: text/html\r\n
```

Figure 43: Shows the version of HTTP used by server to send response back to browser.

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

We can find the last modified time of the HTML file at server by taking a look at the `Last-Modified` property of the HTTP response object. The `Last-Modified` property contains time which indicates the time when the file was last modifed.

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

We can specify `ping` command to capture a specific number of `ECHO_REQUEST` packets by using the `-c` option. If the `-c` is not used, `ping` will continuously keep sending packets until an interrupt signal is received.

Example : `ping -c5 www.pes.edu` will send 5 `ECHO_REQUEST` to `www.pes.edu`.

8.4 How will you identify remote host apps and OS?

Using `nmap`, we can scan the network to find information about the remote host apps and the OS.

Example : Executing the command `sudo nmap -O -v www.tesla.com` will give the following results.

```
vishalr@ubuntu:~$ sudo nmap -O -v www.tesla.com
[sudo] password for vishalr:
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-26 19:13 PST
Initiating Ping Scan at 19:13
Scanning www.tesla.com (23.35.62.248) [4 ports]
Completed Ping Scan at 19:13, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:13
Completed Parallel DNS resolution of 1 host. at 19:13, 0.01s elapsed
Initiating SYN Stealth Scan at 19:13
Scanning www.tesla.com (23.35.62.248) [1000 ports]
Discovered open port 443/tcp on 23.35.62.248
Discovered open port 80/tcp on 23.35.62.248
Completed SYN Stealth Scan at 19:13, 4.80s elapsed (1000 total ports)
Initiating OS detection (try #1) against www.tesla.com (23.35.62.248)
Retrying OS detection (try #2) against www.tesla.com (23.35.62.248)
Nmap scan report for www.tesla.com (23.35.62.248)
Host is up (0.010s latency).
Other addresses for www.tesla.com (not scanned): 2600:140f:3:1184::700 2600:140f:3:119b::700
rDNS record for 23.35.62.248: a23-35-62-248.deploy.static.akamaitechnologies.com
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Linux 2.6.X (86%)
OS CPE: cpe:/o:linux:linux_kernel:2.6
Aggressive OS guesses: Linux 2.6.18 - 2.6.22 (86%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 0.000 days (since Tue Jan 26 19:13:30 2021)
TCP Sequence Prediction: Difficulty=234 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/bin/../share/nmap
OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.23 seconds
           Raw packets sent: 2077 (95.072KB) | Rcvd: 24 (1.768KB)
vishalr@ubuntu:~$
```