

CSE 232: Programming Assignment 3

Using Linux iptables

Anushka Srivastava (2022086)

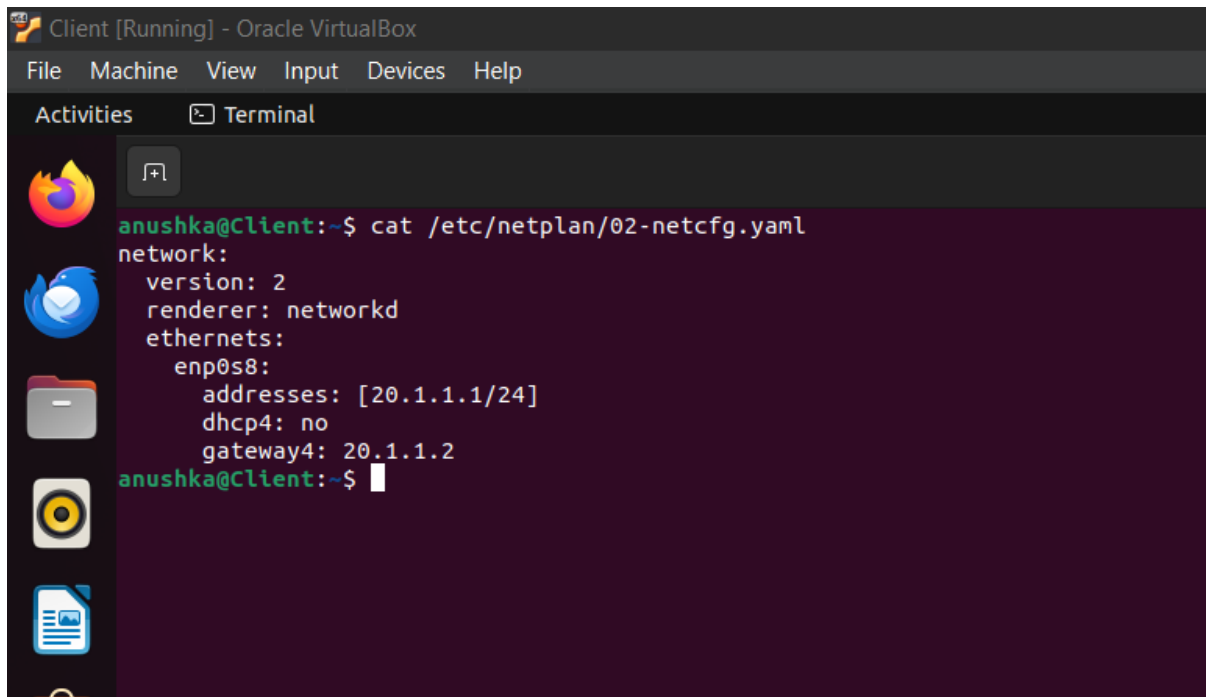
October 2024

Question 1

a.) Configure the IP addresses and routes for all VMs, as shown in the figure.

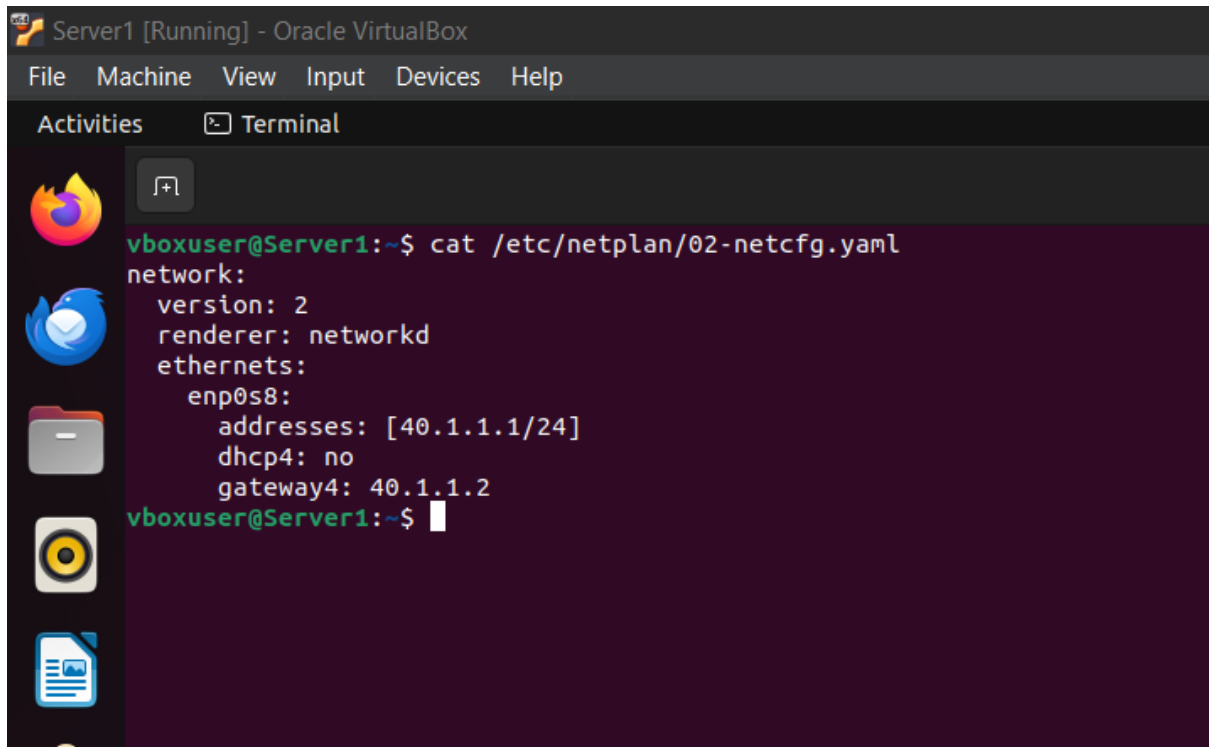
To configure the IP addresses of all client and server VMs:

1. We turn on the second **network adapter** and turn it up using the command `sudo ip link set <adapter-name> up`, which in this case is `enp0s8` for network adapter 2 and `enp0s9` for network adapter 3 (network adapter 3 is only turned on for Gateway VM). The adapters are attached to Host-Only Adapter.
2. We add the configuration information in `/etc/netplan/02-netcfg.yaml` file, as shown in the figures.



```
Client [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
anushka@Client:~$ cat /etc/netplan/02-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses: [20.1.1.1/24]
      dhcp4: no
      gateway4: 20.1.1.2
anushka@Client:~$
```

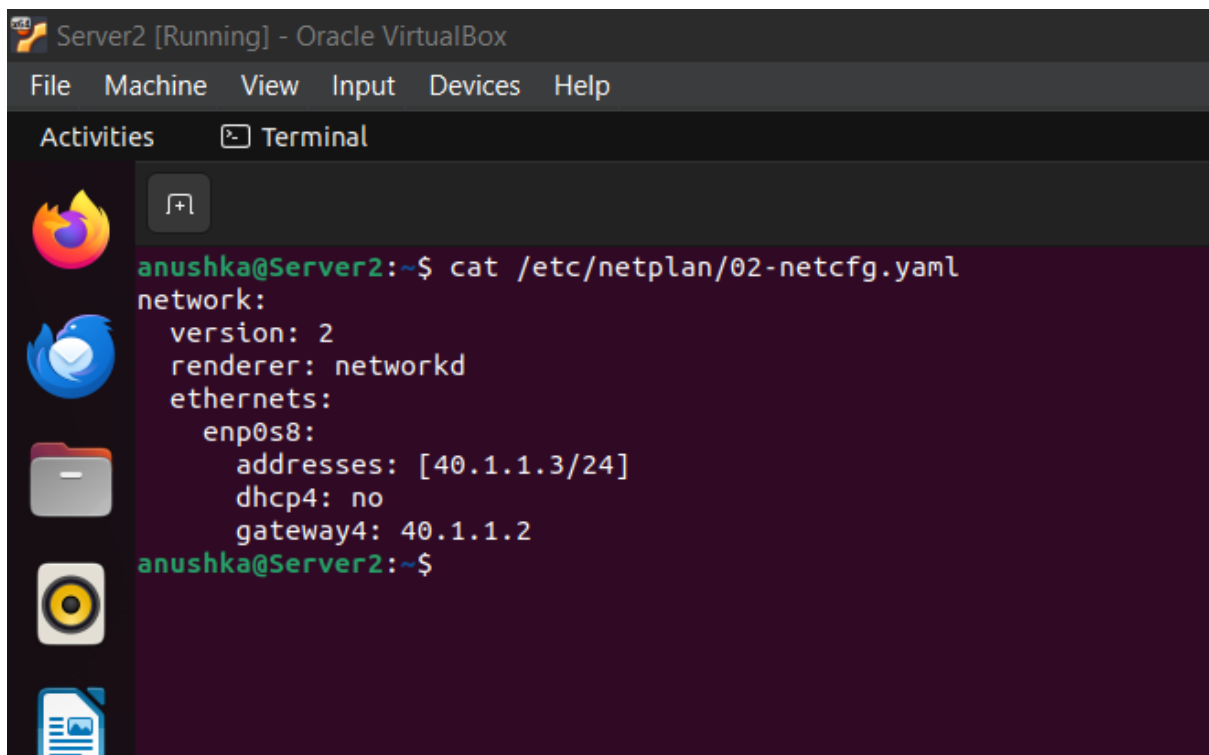
Figure 1: IP Address Configuration for Client VM



The screenshot shows the Oracle VM VirtualBox interface for a VM named 'Server1'. The 'Terminal' window is active, displaying the following netplan configuration for the 'enp0s8' interface:

```
vboxuser@Server1:~$ cat /etc/netplan/02-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses: [40.1.1.1/24]
      dhcp4: no
      gateway4: 40.1.1.2
vboxuser@Server1:~$
```

Figure 2: IP Address Configuration for Server1 VM



The screenshot shows the Oracle VM VirtualBox interface for a VM named 'Server2'. The 'Terminal' window is active, displaying the following netplan configuration for the 'enp0s8' interface:

```
anushka@Server2:~$ cat /etc/netplan/02-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses: [40.1.1.3/24]
      dhcp4: no
      gateway4: 40.1.1.2
anushka@Server2:~$
```

Figure 3: IP Address Configuration for Server2 VM

To configure the routes for client and server VMs, we run the following command in the terminals to add the VMs to the routing table:

```
gateway4: 20.1.1.2
anushka@Client:~$ ip route show
default via 20.1.1.2 dev enp0s8 proto static
20.1.1.0/24 dev enp0s8 proto kernel scope link src 20.1.1.1
anushka@Client:~$ sudo ip route add 40.1.1.0/24 via 20.1.1.2
[sudo] password for anushka:

anushka@Client:~$ ip route show
default via 20.1.1.2 dev enp0s8 proto static
20.1.1.0/24 dev enp0s8 proto kernel scope link src 20.1.1.1
40.1.1.0/24 via 20.1.1.2 dev enp0s8
anushka@Client:~$
```

Figure 4: IP Route Configuration for Client VM

```
gateway4: 40.1.1.2
vboxuser@Server1:~$ sudo ip route add 20.1.1.0/24 via 40.1.1.2
[sudo] password for vboxuser:
vboxuser@Server1:~$ ip route show
default via 40.1.1.2 dev enp0s8 proto static
20.1.1.0/24 via 40.1.1.2 dev enp0s8
40.1.1.0/24 dev enp0s8 proto kernel scope link src 40.1.1.1
vboxuser@Server1:~$
```

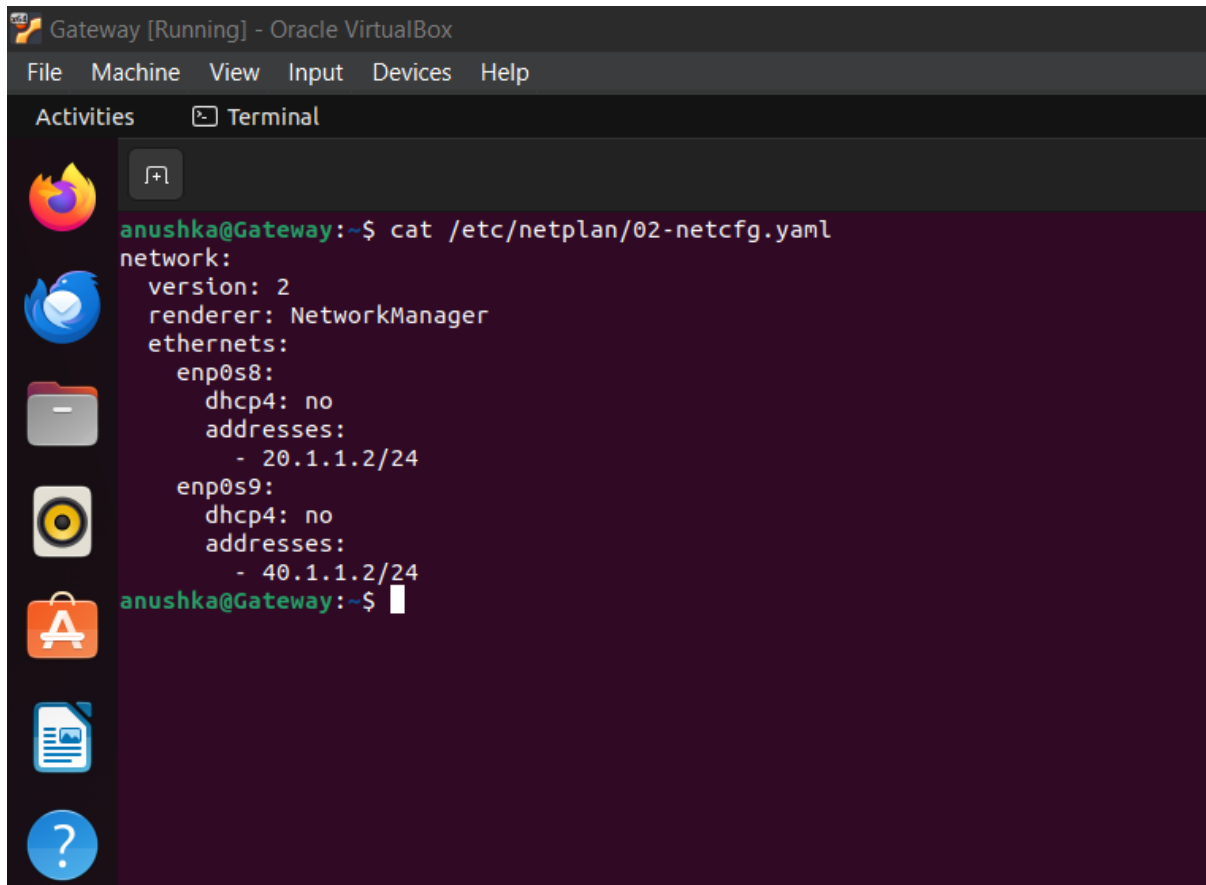
Figure 5: IP Route Configuration for Server1 VM

```
169.254.0.0/16 dev enp0s8 scope link metric 1000
anushka@Server2:~$ sudo ip route add 20.1.1.0/24 via 40.1.1.2
[sudo] password for anushka:
anushka@Server2:~$ ip route show
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
default via 40.1.1.2 dev enp0s8 proto static metric 20101
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
20.1.1.0/24 via 40.1.1.2 dev enp0s8
40.1.1.0/24 dev enp0s8 proto kernel scope link src 40.1.1.3 metric 101
169.254.0.0/16 dev enp0s8 scope link metric 1000
anushka@Server2:~$
```

Figure 6: IP Route Configuration for Server2 VM

b.) Configure VM2 as the gateway such that it can forward the incoming traffic to one of the servers – add forwarding functionality.

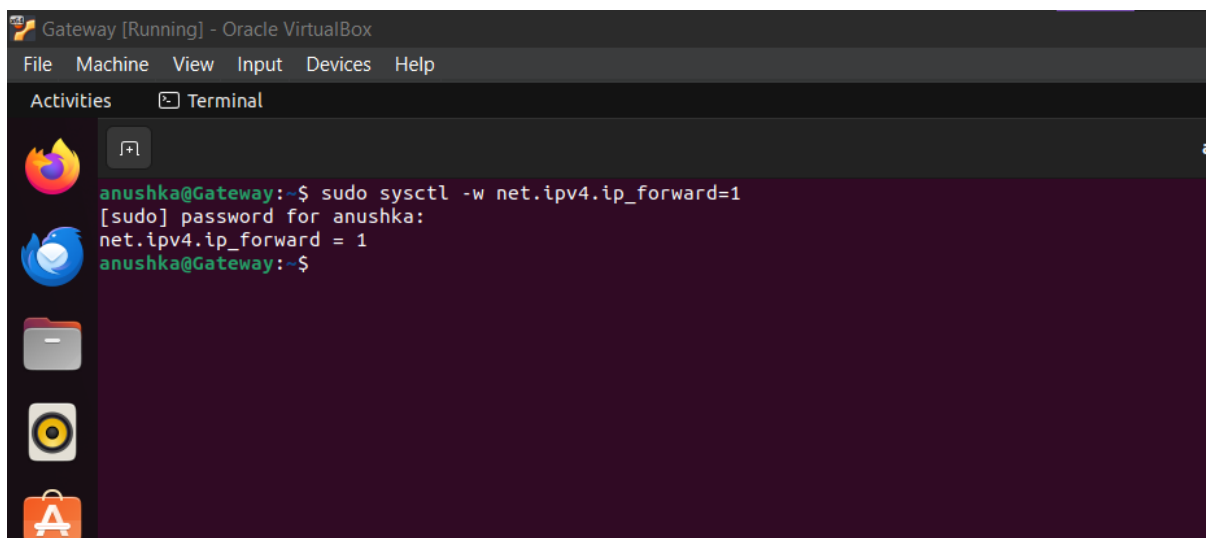
We add the following configuration details in the `/etc/netplan/02-netcfg.yaml` file to configure the Gateway.



```
Gateway [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
anushka@Gateway:~$ cat /etc/netplan/02-netcfg.yaml
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s8:
      dhcp4: no
      addresses:
        - 20.1.1.2/24
    enp0s9:
      dhcp4: no
      addresses:
        - 40.1.1.2/24
anushka@Gateway:~$
```

Figure 7: IP Route Configuration for Server2 VM

To add forwarding functionality, we run the following command in the Gateway VM.



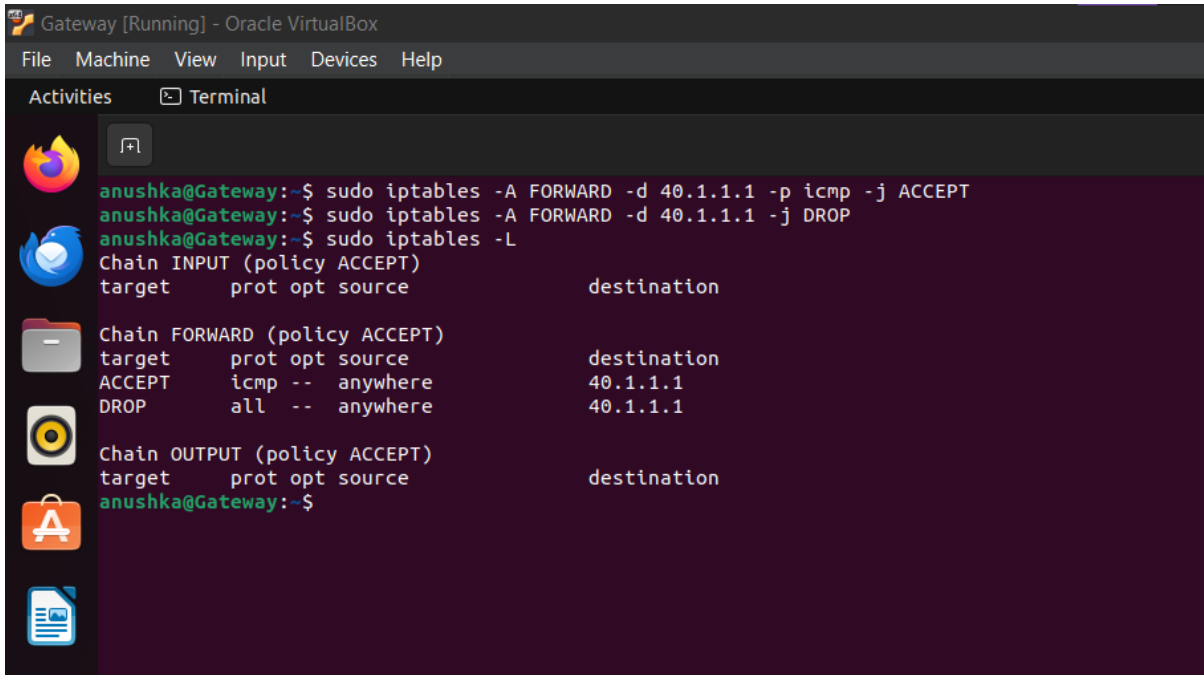
```
Gateway [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
anushka@Gateway:~$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] password for anushka:
net.ipv4.ip_forward = 1
anushka@Gateway:~$
```

Figure 8: Adding forward functionality to the Gateway VM

Question 2

a.) The gateway must block all traffic (except for ping) destined to the server 40.1.1.1/24. Show that this works; attach the screenshot.

To block all traffic except ping, we configure the `iptables` in the following way such that it accepts all ICMP packets sent by ping and drops all other packets.



```
Gateway [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal

anushka@Gateway:~$ sudo iptables -A FORWARD -d 40.1.1.1 -p icmp -j ACCEPT
anushka@Gateway:~$ sudo iptables -A FORWARD -d 40.1.1.1 -j DROP
anushka@Gateway:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
ACCEPT     icmp -- anywhere             40.1.1.1
DROP       all  -- anywhere             40.1.1.1

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
anushka@Gateway:~$
```

Figure 9: iptables configuration for Gateway VM

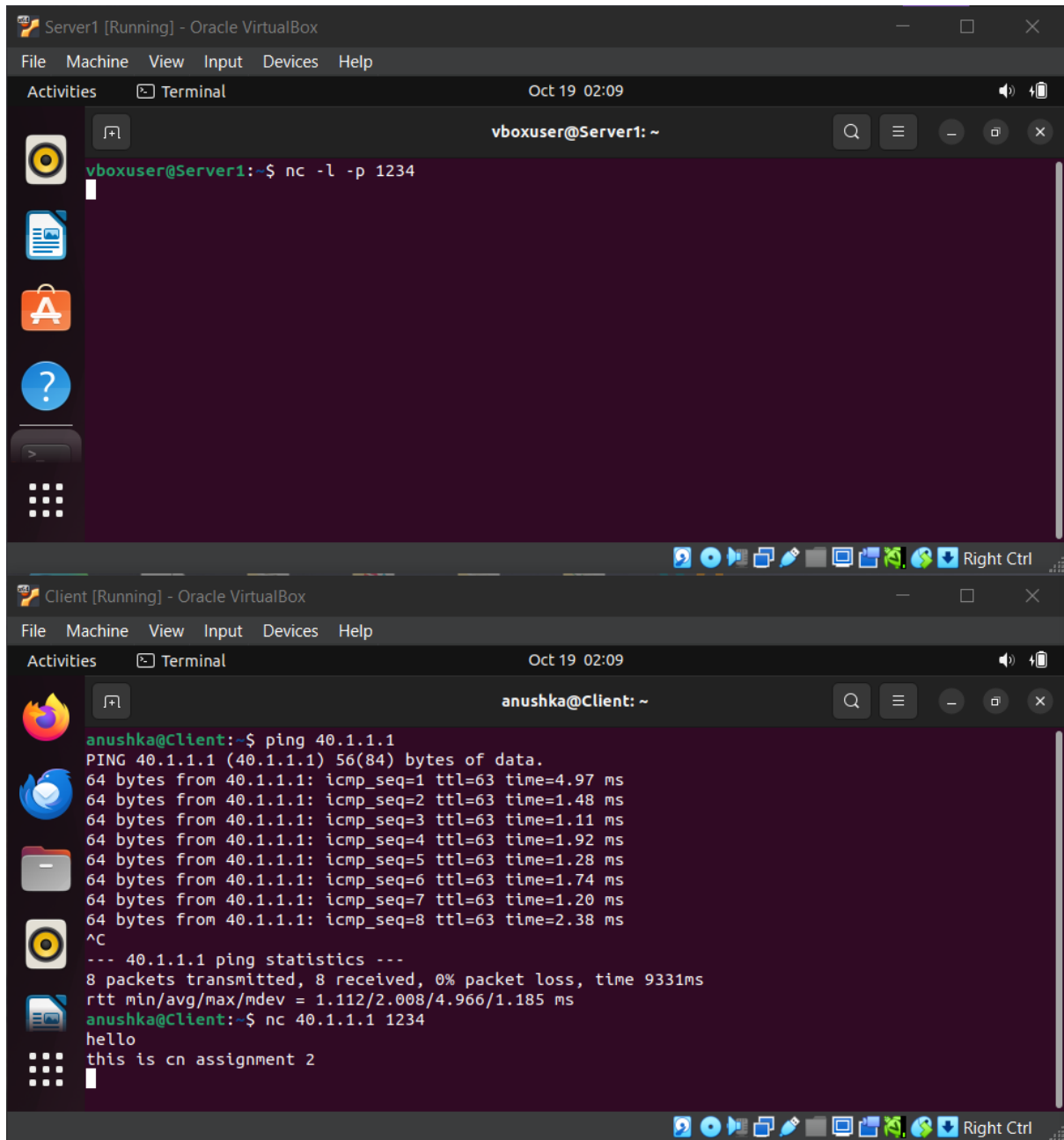


Figure 10: Screenshot showing ping working but netcat connection not working

b.) The gateway must block only TCP traffic initiated by 20.1.1.1/24. Show that this works; attach the screenshot.

To achieve this, we configure the `iptables` such that it drops all packets from tcp connection. We then test both the tcp and udp connection between the server and client and notice the result as shown in the figure. Here, we set up a connection between the client and the gateway as the gateway is responsible for routing the traffic to the servers so it provides the same results.

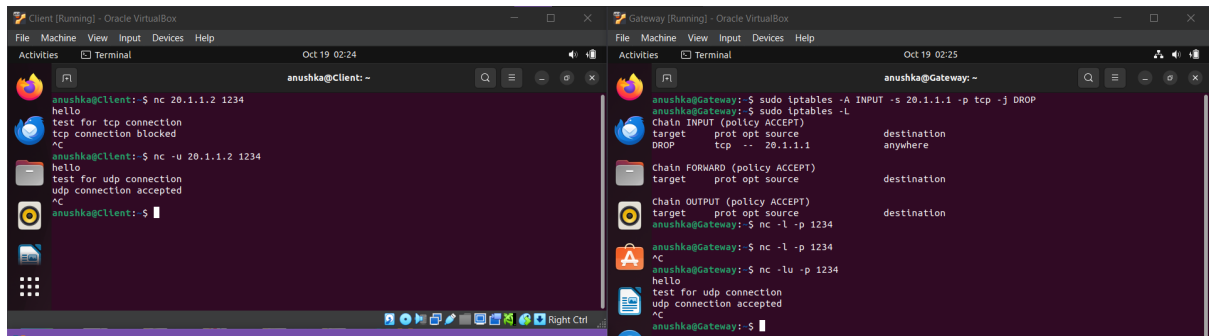


Figure 11: Testing tcp and udp connection between client and gateway

Question 3

a.) Use “iperf” tool to test the TCP and UDP bandwidth between 20.1.1.1/24 and 40.1.1.3/24. Attach the screenshot.

For this question, we use the following configuration for iptables of Gateway VM from Q2:

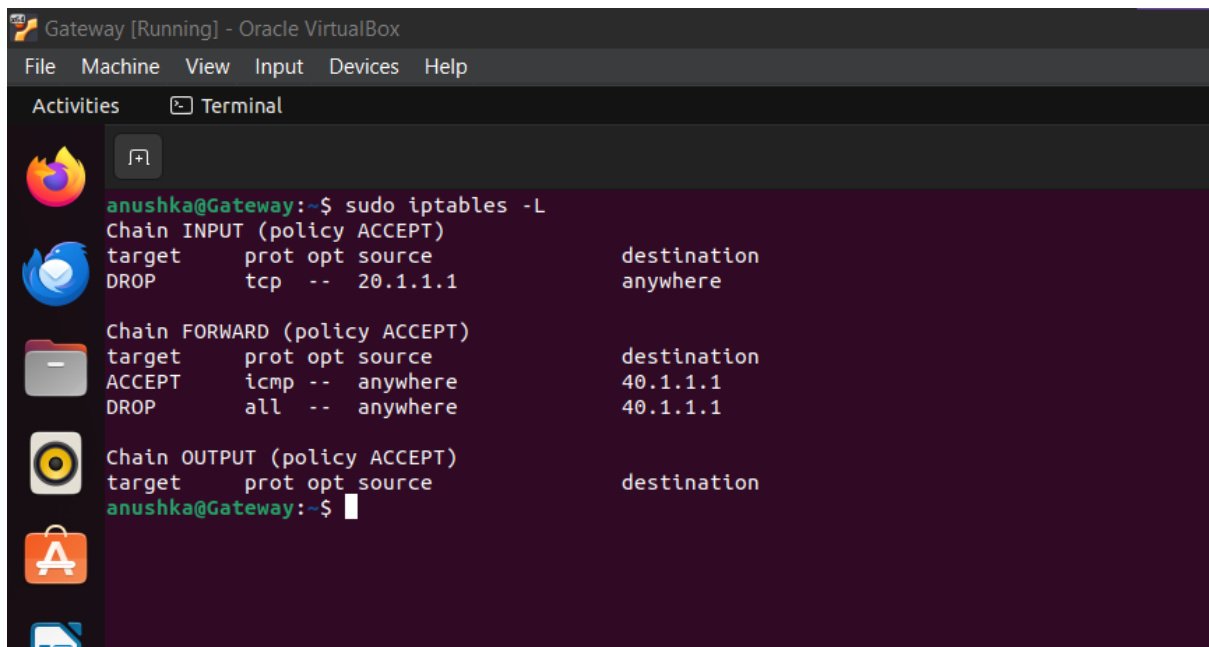


Figure 12: iptables for Gateway VM

First, we test the TCP bandwidth between server 40.1.1.3 and the client.

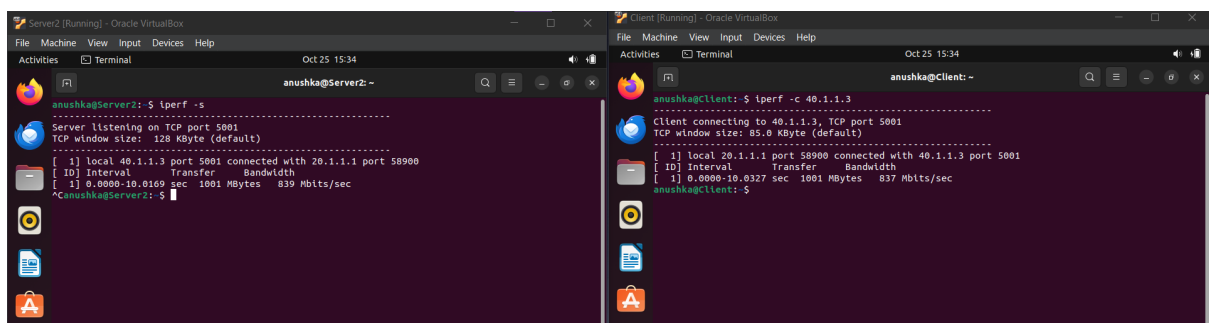


Figure 13: TCP Bandwidth using iperf

Analysis for TCP Bandwidth

1. The TCP connection performed well, with an average bandwidth of over 800 Mbits/sec for both sender and receiver. The connection details are logged for a time period of about 10.0327 seconds.
2. The TCP window size is 128 Kbyte for server and 85 KByte for client.
3. Over 1001 MBytes of data is transferred between the sender and receiver over the network.

Next, we test the UDP bandwidth between server 40.1.1.3 and the client.

```

Server2 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
Oct 25 15:32
anushka@Server2:~$ iperf -s -u
Server listening on UDP port 5001
UDP buffer size: 208 Kbyte (default)
[ 1] local 40.1.1.3 port 5001 connected with 20.1.1.1 port 48477
[ ID] Interval      Transfer      Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-10.0155 sec  1.25 MBytes  1.05 Mbits/sec  0.498 ms  0/895 (0%)
anushka@Server2:~$

Client [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
Oct 25 15:32
anushka@Client:~$ iperf -c 40.1.1.3 -u
Client connecting to 40.1.1.3, UDP port 5001
Sending 1478 byte datagrams, IPQ target: 11215.21 us (kalman adjust)
UDP buffer size: 208 Kbyte (default)
[ 1] local 20.1.1.1 port 48477 connected with 40.1.1.3 port 5001
[ ID] Interval      Transfer      Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-10.0101 sec  1.25 MBytes  1.05 Mbits/sec
[ 1] Sent 896 datagrams
[ 1] Server Report:
[ ID] Interval      Transfer      Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-10.0155 sec  1.25 MBytes  1.05 Mbits/sec  0.497 ms  0/895 (0%)
anushka@Client:~$

```

Figure 14: UDP Bandwidth using iperf

Analysis for UDP Bandwidth

1. The UDP connection did not perform better than TCP connection as it has a bandwidth of over 1.05 Mbits/sec only for both sender and receiver. The connection details are logged for a time period of about 10.0155 seconds.
2. Total of 896 datagrams are transferred over the network. The report shows 0% packet loss.
3. Over 1.25 MBytes of data is transferred between the sender and receiver over the network.

b.) What is the minimum, average, and maximum RTT

i.) from 20.1.1.1/24 to 40.1.1.1/24

```

Client [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
Oct 24 00:51
anushka@Client:~$ ping -c 10 40.1.1.1
PING 40.1.1.1 (40.1.1.1) 56(84) bytes of data:
64 bytes from 40.1.1.1: icmp_seq=1 ttl=63 time=2.82 ms
64 bytes from 40.1.1.1: icmp_seq=2 ttl=63 time=1.41 ms
64 bytes from 40.1.1.1: icmp_seq=3 ttl=63 time=1.32 ms
64 bytes from 40.1.1.1: icmp_seq=4 ttl=63 time=1.67 ms
64 bytes from 40.1.1.1: icmp_seq=5 ttl=63 time=1.29 ms
64 bytes from 40.1.1.1: icmp_seq=6 ttl=63 time=1.36 ms
64 bytes from 40.1.1.1: icmp_seq=7 ttl=63 time=1.18 ms
64 bytes from 40.1.1.1: icmp_seq=8 ttl=63 time=1.27 ms
64 bytes from 40.1.1.1: icmp_seq=9 ttl=63 time=1.27 ms
64 bytes from 40.1.1.1: icmp_seq=10 ttl=63 time=1.34 ms

--- 40.1.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9874ms
rtt min/avg/max/mdev = 1.182/1.491/2.818/0.459 ms
anushka@Client:~$

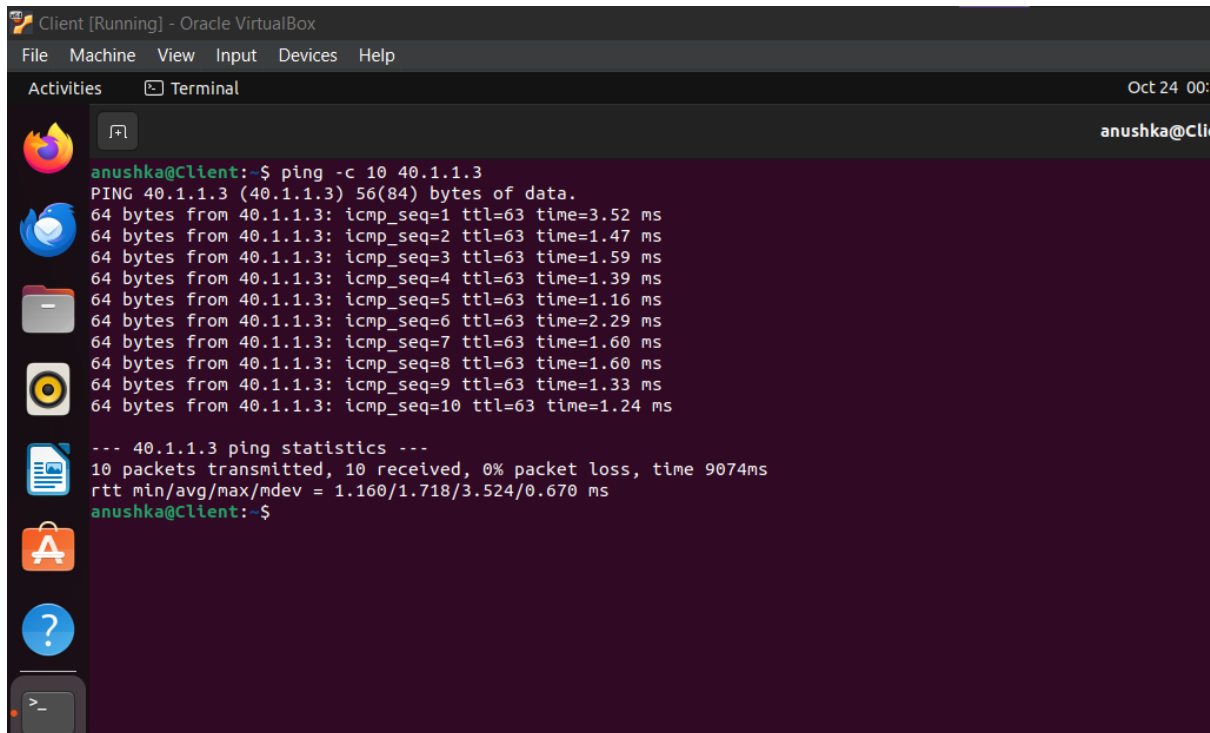
```

Figure 15: ping statistics for server 1

We notice the following:

1. Minimum RTT: 1.182 ms
2. Average RTT: 1.491 ms
3. Maximum RTT: 2.818 ms

ii.) from 20.1.1.1/24 to 40.1.1.3/24



```
Client [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities Terminal
anushka@Client:~$ ping -c 10 40.1.1.3
PING 40.1.1.3 (40.1.1.3) 56(84) bytes of data.
64 bytes from 40.1.1.3: icmp_seq=1 ttl=63 time=3.52 ms
64 bytes from 40.1.1.3: icmp_seq=2 ttl=63 time=1.47 ms
64 bytes from 40.1.1.3: icmp_seq=3 ttl=63 time=1.59 ms
64 bytes from 40.1.1.3: icmp_seq=4 ttl=63 time=1.39 ms
64 bytes from 40.1.1.3: icmp_seq=5 ttl=63 time=1.16 ms
64 bytes from 40.1.1.3: icmp_seq=6 ttl=63 time=2.29 ms
64 bytes from 40.1.1.3: icmp_seq=7 ttl=63 time=1.60 ms
64 bytes from 40.1.1.3: icmp_seq=8 ttl=63 time=1.60 ms
64 bytes from 40.1.1.3: icmp_seq=9 ttl=63 time=1.33 ms
64 bytes from 40.1.1.3: icmp_seq=10 ttl=63 time=1.24 ms
--- 40.1.1.3 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9074ms
rtt min/avg/max/mdev = 1.160/1.718/3.524/0.670 ms
anushka@Client:~$
```

Figure 16: ping statistics for server 2

We notice the following:

1. Minimum RTT: 1.160 ms
2. Average RTT: 1.718 ms
3. Maximum RTT: 3.524 ms

iii.) Did you find a significant difference between (i) and (ii)? If so, why?

We notice that the average RTT and the mean deviation of RTT is slightly higher for server 2 as compared to server 1. The possible reason could be the blocking of all other connections except ping to server 1, minimising the traffic directed to server 1 which in turn results in lower latency and RTT value.

Question 4

a.) Change the source IP address of every packet from 20.1.1.1/24 to 40.1.1.2/24.

To change the source IP address of every packet from 20.1.1.1/24 to 40.1.1.1/24, we use SNAT for the same. The change is visible in the NAT table of iptables and can be tested by opening a TCP connection between client and server and testing the network packets using tcpdump at the Gateway VM.

(Refer to the screenshot in part 3)

b.) When the packet response for the packet from step “a” arrives at the gateway, revert the destination IP address to the original.

To change the destination IP address of every packet from gateway to 20.1.1.1/24, we use DNAT for the same. The change is visible in the NAT table of `iptables` and can be tested by opening a TCP connection between client as the port and connecting the server to the IP address of the Gateway VM at the same port. We notice all the packets being transmitted to the client.

(Refer to the screenshot in part 3)

c.) Validate the above by sending traffic and observing the packets at each VM using Wireshark/tcpdump.

We use `tcpdump` to test the network packets transferred in the above 2 parts.

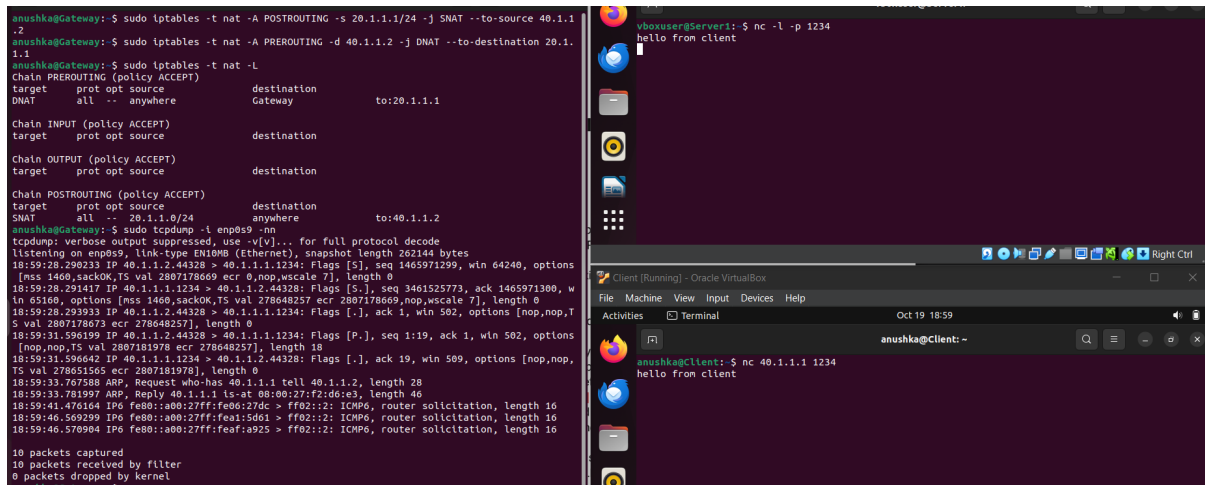


Figure 17: Changing the source IP address

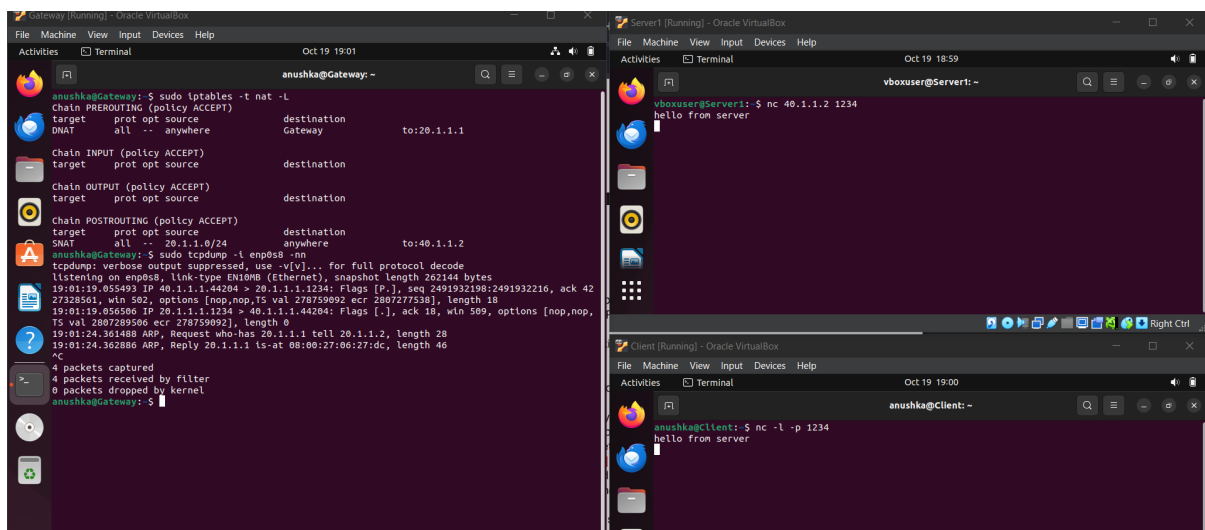


Figure 18: Changing the destination IP address

Question 5

a.) Using the information obtained from Q.3.b., balance the traffic from 20.1.1.1/24 to the servers, 40.1.1.1/24 and 40.1.1.3/24. The probability of assigning the packet to the servers is 0.8 and 0.2, i.e., assign a high probability to the server with lower RTT.

Server 1 has lower RTT so we assign 0.8 to server 1. We use random balancing based on probability to assign probability to both the servers. The NAT table of `iptables` is configured in the following way:

```

anushka@Gateway:~$ sudo iptables -t nat -A PREROUTING -s 20.1.1.1/24 -m statistic --mode random --probability 0.8 -j DNAT --to-destination 40.1.1.1
[sudo] password for anushka:
anushka@Gateway:~$ sudo iptables -t nat -A PREROUTING -s 20.1.1.1/24 -j DNAT --to-destination 40.1.1.3
anushka@Gateway:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination
DNAT       all  --  20.1.1.0/24            anywhere        statistic mode random probability 0
           all  --  20.1.1.0/24            anywhere        to:40.1.1.3
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
anushka@Gateway:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1

```

Figure 19: Load Balancing based on probability

b.) Test the above configuration using a series of “ping” packets.

Since server 2 has been assigned lower probability, we ping server 2 to show the results clearly as the chances of second rule being executed is 20% only. We use `tcpdump` to analyse the network packets transferred between the client and the server and we notice some packets being routed to server 1 with IP address 40.1.1.1 and rest being forwarded to server 2.

```

anushka@Gateway:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
anushka@Gateway:~$ sudo tcpdump -i any icmp -nn
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
00:10:24.437894 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 1, length 64
00:10:24.437736 enp0s9 Out IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 1, length 64
00:10:25.938894 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 2, length 64
00:10:25.938910 enp0s9 Out IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 2, length 64
00:10:27.044231 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 3, length 64
00:10:27.044246 enp0s9 Out IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 3, length 64
00:10:28.236977 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 4, length 64
00:10:28.236992 enp0s9 Out IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 1, seq 4, length 64
00:10:31.907016 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 1, length 64
00:10:31.911826 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 1, length 64
00:10:31.914908 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 1, length 64
00:10:31.915668 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 1, length 64
00:10:32.912138 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 2, length 64
00:10:32.912353 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 2, length 64
00:10:32.912914 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 2, length 64
00:10:32.912921 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 2, length 64
00:10:33.913188 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 3, length 64
00:10:33.913204 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 3, length 64
00:10:33.913767 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 3, length 64
00:10:33.913775 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 3, length 64
00:10:34.919170 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 4, length 64
00:10:34.919193 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 4, length 64
00:10:34.920847 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 4, length 64
00:10:34.920857 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 4, length 64
00:10:35.921230 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 5, length 64
00:10:35.921247 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 5, length 64
00:10:35.921943 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 5, length 64
00:10:35.921957 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 5, length 64
00:10:37.134625 enp0s9 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 6, length 64
00:10:37.134642 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 6, length 64
00:10:37.135305 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 6, length 64
00:10:37.135313 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 6, length 64
00:10:38.438880 enp0s8 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 7, length 64
00:10:38.438895 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 7, length 64
00:10:38.438894 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 7, length 64
00:10:38.438701 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 7, length 64
00:10:39.556027 enp0s8 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 8, length 64
00:10:39.556043 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 8, length 64
00:10:39.556045 enp0s9 In IP 40.1.1.1 > 20.1.1.1: ICMP echo reply, id 2, seq 8, length 64
00:10:39.556056 enp0s8 Out IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2, seq 8, length 64
00:10:40.557703 enp0s8 In IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2, seq 9, length 64
00:10:40.557721 enp0s9 Out IP 20.1.1.1 > 40.1.1.1: ICMP echo request, id 2, seq 9, length 64

```

```

anushkaClient:~$ ping -c 10 40.1.1.3
PING 40.1.1.3 (40.1.1.3) 56(84) bytes of data:
64 bytes from 40.1.1.3: icmp_seq=1 ttl=63 time=2.13 ms
64 bytes from 40.1.1.3: icmp_seq=2 ttl=63 time=12.7 ms
64 bytes from 40.1.1.3: icmp_seq=3 ttl=63 time=2.40 ms
64 bytes from 40.1.1.3: icmp_seq=4 ttl=63 time=1.16 ms
64 bytes from 40.1.1.3: icmp_seq=5 ttl=63 time=6.46 ms
64 bytes from 40.1.1.3: icmp_seq=6 ttl=63 time=1.49 ms
64 bytes from 40.1.1.3: icmp_seq=7 ttl=63 time=1.16 ms
64 bytes from 40.1.1.3: icmp_seq=8 ttl=63 time=1.38 ms
64 bytes from 40.1.1.3: icmp_seq=9 ttl=63 time=1.53 ms
64 bytes from 40.1.1.3: icmp_seq=10 ttl=63 time=1.24 ms

--- 40.1.1.3 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9074ms
rtt min/avg/max/mdev = 1.156/3.166/12.710/3.521 ms
anushkaClient:~$

```

Figure 20: Sending ping packets to server 2

References

- [1] [Scalingo: Turning IPTables into a TCP load balancer for fun and profit](#)
- [2] [Github: iptables-loadbalancer](#)
- [3] [Tutorial 5: iptables](#)