Using Linux IPTables

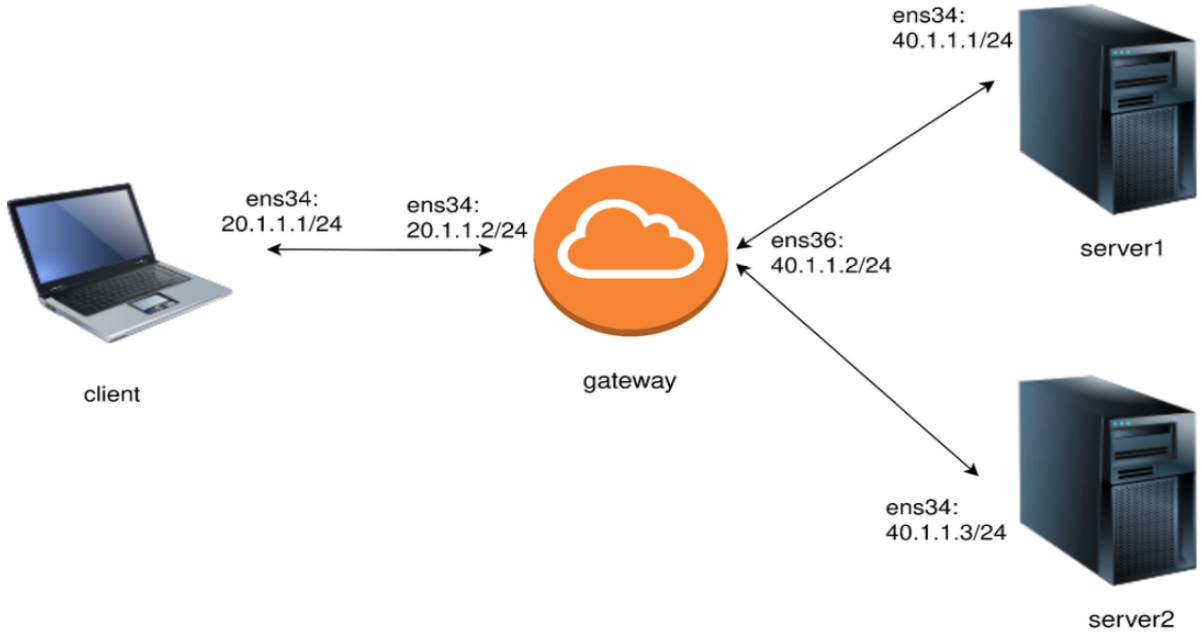
Saksham Singh
2022434
CSE 232 - Computer Networks - PA03

Q1 - Setting up the VMs and Port Forwarding

Deliverables

Set up four VMs as shown in the figure. Use the same setup for the entire assignment.

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



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

Methodology

- I have created 4 VMs using VirtualBox and installed Ubuntu 20.04 on all of them.
- I have configured the IP addresses and routes for all VMs as shown in the figure using VirtualBox's network settings for adding a NAT network. (as shown in the image above), and then configured the IP addresses and routes using the following commands:

```
sudo nano /etc/netplan/01-netcfg.yaml
```

then added the following configuration:

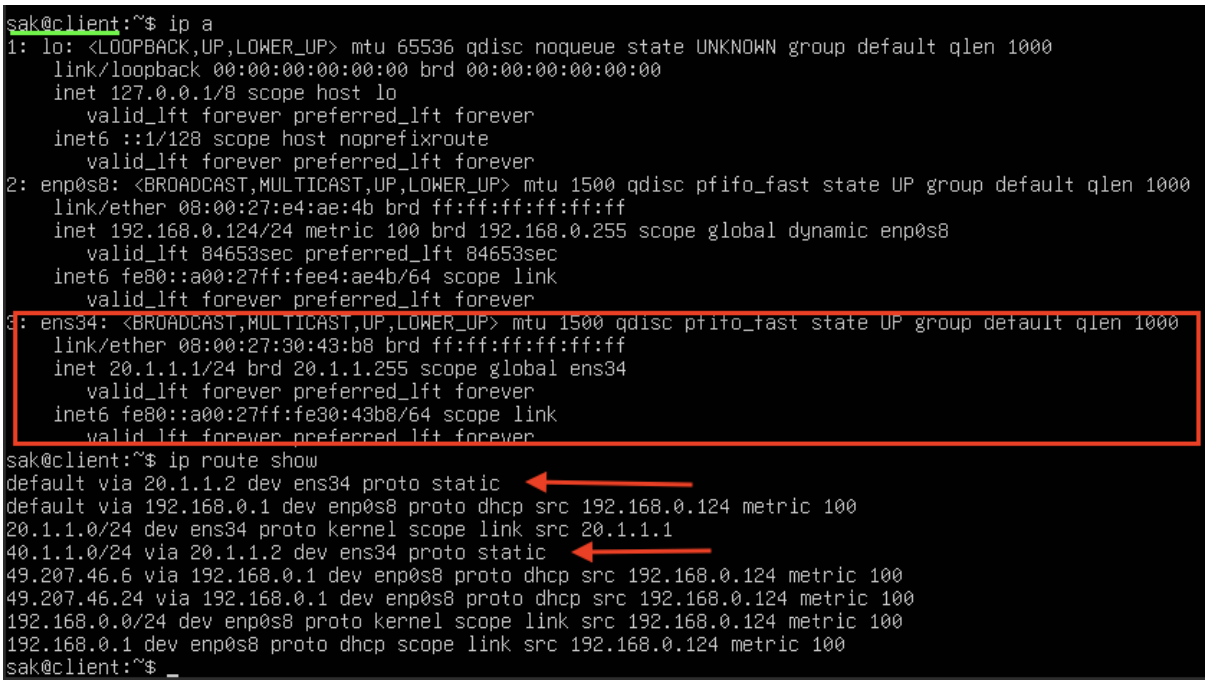
```
network:
    version: 2
```

```
        renderer: networkd
        ethernets:
        [port name]:
            match:
                macaddress: [mac address]
            set-name: [new port name]
            addresses:
            - [IP address]/[subnet mask]
            gateway4: [gateway IP]
            dhcp4: false
            routes:
            - to: [destination network addr]/[subnet mask]
                via: [gateway IP]
```

then applied the changes using the following command:

```
sudo netplan apply
```

- Using the above method, I have configured the IP addresses and routes for all the VMs and interfaces as shown in the figure. This way the configurations and routes persist even after a reboot.
- The following are the results
    - VM1 - Client:



port `ens34` has IP address `20.1.1.1/24` and default gateway `20.1.1.2`, with a route to `40.1.1.0/24` via `20.1.1.2`

- VM2 - Gateway:

```
sak@gateway:~$ ip a
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 noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:5e:ec:3e brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.125/24 metric 100 brd 192.168.0.255 scope global dynamic enp0s8
       valid_lft 84151sec preferred_lft 84151sec
    inet6 fe80::a00:27ff:fe5e:ec3e/64 scope link
       valid_lft forever preferred_lft forever
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:42:19:82 brd ff:ff:ff:ff:ff:ff
    inet 20.1.1.2/24 brd 20.1.1.255 scope global ens34
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe42:1982/64 scope link
       valid_lft forever preferred_lft forever
4: ens36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:96:05:c6 brd ff:ff:ff:ff:ff:ff
    inet 40.1.1.2/24 brd 40.1.1.255 scope global ens36
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe96:5c6/64 scope link
       valid_lft forever preferred_lft forever
```

port `ens34` has IP address `20.1.1.2/24` and port `ens36` has IP address `40.1.1.2` -- acting as a gateway for the networks `20.1.1.0/24` and `40.1.1.0/24` respectively.

- VM3 - Server 1:

```
sak@server1:~$ ip a
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 noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:12:36:e2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.127/24 metric 100 brd 192.168.0.255 scope global dynamic enp0s8
       valid_lft 83978sec preferred_lft 83978sec
    inet6 fe80::a00:27ff:fe12:36e2/64 scope link
       valid_lft forever preferred_lft forever
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:b3:14:8b brd ff:ff:ff:ff:ff:ff
    inet 40.1.1.1/24 brd 40.1.1.255 scope global ens34
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feb3:148b/64 scope link
       valid_lft forever preferred_lft forever
sak@server1:~$ ip route show
default via 40.1.1.2 dev ens34 proto static
default via 192.168.0.1 dev enp0s8 proto dhcp src 192.168.0.127 metric 100
20.1.1.0/24 via 40.1.1.2 dev ens34 proto static
40.1.1.0/24 dev ens34 proto kernel scope link src 40.1.1.1
49.207.46.6 via 192.168.0.1 dev enp0s8 proto dhcp src 192.168.0.127 metric 100
49.207.46.24 via 192.168.0.1 dev enp0s8 proto dhcp src 192.168.0.127 metric 100
192.168.0.0/24 dev enp0s8 proto kernel scope link src 192.168.0.127 metric 100
192.168.0.1 dev enp0s8 proto dhcp scope link src 192.168.0.127 metric 100
sak@server1:~$ _
```

port `ens34` has IP address `40.1.1.1/24` and default gateway `40.1.1.2`, with a route to `20.1.1.0/24` via `40.1.1.2`

- VM4 - Server 2:

```
sak@server2:~$ ip a
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 noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:05:d5:8b brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.126/24 metric 100 brd 192.168.0.255 scope global dynamic enp0s8
       valid_lft 83860sec preferred_lft 83860sec
    inet6 fe80::a00:27ff:fe05:d58b/64 scope link
       valid_lft forever preferred_lft forever
3: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:56:53:85 brd ff:ff:ff:ff:ff:ff
    inet 40.1.1.3/24 brd 40.1.1.255 scope global ens34
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe56:5385/64 scope link
       valid_lft forever preferred_lft forever
sak@server2:~$ ip route show
default via 40.1.1.2 dev ens34 proto static
default via 192.168.0.1 dev enp0s8 proto dhcp src 192.168.0.126 metric 100
20.1.1.0/24 via 40.1.1.2 dev ens34 proto static
40.1.1.0/24 dev ens34 proto kernel scope link src 40.1.1.3
49.207.46.6 via 192.168.0.1 dev enp0s8 proto dhcp src 192.168.0.126 metric 100
49.207.46.24 via 192.168.0.1 dev enp0s8 proto dhcp src 192.168.0.126 metric 100
192.168.0.0/24 dev enp0s8 proto kernel scope link src 192.168.0.126 metric 100
192.168.0.1 dev enp0s8 proto dhcp scope link src 192.168.0.126 metric 100
sak@server2:~$
```

  port `ens34` has IP address `40.1.1.3/24` and default gateway `40.1.1.2`, with a route to `20.1.1.0/24` via `40.1.1.2`

- Each VM has a port `enp0s8` which is a Bridged Adapter connected to the host network for internet access.

- I have configured VM2 as the gateway such that it can forward the incoming traffic to one of the servers using the following commands:

```
sudo nano /etc/sysctl.conf
```

  then uncommented the following line:

```
net.ipv4.ip_forward=1
```

  then applied the changes using the following command:

```
sudo sysctl -p
```

```
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguration
#  based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

- The following can be verified using `ping` command from VM1 to VM3 and VM4:
    - VM1 to VM3:

```
sak@client:~$ ping 40.1.1.1 -c 10
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=5.62 ms
64 bytes from 40.1.1.1: icmp_seq=2 ttl=63 time=1.39 ms
64 bytes from 40.1.1.1: icmp_seq=3 ttl=63 time=0.741 ms
64 bytes from 40.1.1.1: icmp_seq=4 ttl=63 time=0.726 ms
64 bytes from 40.1.1.1: icmp_seq=5 ttl=63 time=1.11 ms
64 bytes from 40.1.1.1: icmp_seq=6 ttl=63 time=0.665 ms
64 bytes from 40.1.1.1: icmp_seq=7 ttl=63 time=0.688 ms
64 bytes from 40.1.1.1: icmp_seq=8 ttl=63 time=0.738 ms
64 bytes from 40.1.1.1: icmp_seq=9 ttl=63 time=0.789 ms
64 bytes from 40.1.1.1: icmp_seq=10 ttl=63 time=0.699 ms

--- 40.1.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9201ms
rtt min/avg/max/mdev = 0.665/1.316/5.615/1.449 ms
sak@client:~$ _
```

    - VM1 to VM4:

```
sak@client:~$ ping 40.1.1.3 -c 10
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=1.75 ms
64 bytes from 40.1.1.3: icmp_seq=2 ttl=63 time=0.714 ms
64 bytes from 40.1.1.3: icmp_seq=3 ttl=63 time=0.821 ms
64 bytes from 40.1.1.3: icmp_seq=4 ttl=63 time=0.651 ms
64 bytes from 40.1.1.3: icmp_seq=5 ttl=63 time=0.632 ms
64 bytes from 40.1.1.3: icmp_seq=6 ttl=63 time=0.734 ms
64 bytes from 40.1.1.3: icmp_seq=7 ttl=63 time=0.808 ms
64 bytes from 40.1.1.3: icmp_seq=8 ttl=63 time=0.728 ms
64 bytes from 40.1.1.3: icmp_seq=9 ttl=63 time=0.684 ms
64 bytes from 40.1.1.3: icmp_seq=10 ttl=63 time=0.770 ms

--- 40.1.1.3 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9202ms
rtt min/avg/max/mdev = 0.632/0.829/1.749/0.312 ms
```

- The above results show that the configurations are correct and the traffic is being forwarded correctly.

Q2 - Traffic filtering at the gateway VM

Deliverables

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

Methodology

- Using IPTables, I have configured the gateway VM to block all traffic (except for ping) destined to the server `40.1.1.1/24` using `OUTPUT` and `FORWARD` chain.
    - The `OUTPUT` chain is used to filter packets that are being sent out of the local system. This chain is used when a packet that creates a new connection is being routed from the local system.
    - The `FORWARD` chain is used to filter packets that are being routed through the local system. This chain is used when a packet that creates a new connection is being routed through the local system.
    - The following commands were used to configure the IPTables:

```
sudo iptables -A FORWARD -d 40.1.1.1 -j DROP
sudo iptables -A FORWARD -d 40.1.1.1 -p icmp --icmp-echo-request
-j ACCEPT
sudo iptables -A FORWARD -d 40.1.1.1 -p icmp --icmp-echo-reply -j
ACCEPT

sudo iptables -A OUTPUT -d 40.1.1.1 -j DROP
sudo iptables -A OUTPUT -d 40.1.1.1 -p icmp --icmp-echo-request -
j ACCEPT
sudo iptables -A OUTPUT -d 40.1.1.1 -p icmp --icmp-echo-reply -j
ACCEPT
```

```
sak@gateway:~$ sudo iptables -A FORWARD -d 40.1.1.1 -p icmp --icmp-type echo-request -j ACCEPT
sak@gateway:~$ sudo iptables -A FORWARD -d 40.1.1.1 -p icmp --icmp-type echo-reply -j ACCEPT
sak@gateway:~$ sudo iptables -A FORWARD -d 40.1.1.1 -j DROP
sak@gateway:~$ sudo iptables -A OUTPUT -d 40.1.1.1 -p icmp --icmp-type echo-reply -j ACCEPT
sak@gateway:~$ sudo iptables -A OUTPUT -d 40.1.1.1 -p icmp --icmp-type echo-request -j ACCEPT
sak@gateway:~$ sudo iptables -A OUTPUT -d 40.1.1.1 -p -j DROP
iptables v1.8.10 (nf_tables): unknown protocol "-j" specified
Try `iptables -h' or 'iptables --help' for more information.
sak@gateway:~$ sudo iptables -A OUTPUT -d 40.1.1.1 -j DROP
sak@gateway:~$ sudo iptables -l
iptables v1.8.10 (nf_tables): unknown option "-l"
Try `iptables -h' or 'iptables --help' for more information.
sak@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             icmp echo-request
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-reply
DROP       all  --  anywhere             40.1.1.1

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-reply
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-request
DROP       all  --  anywhere             40.1.1.1
sak@gateway:~$
```

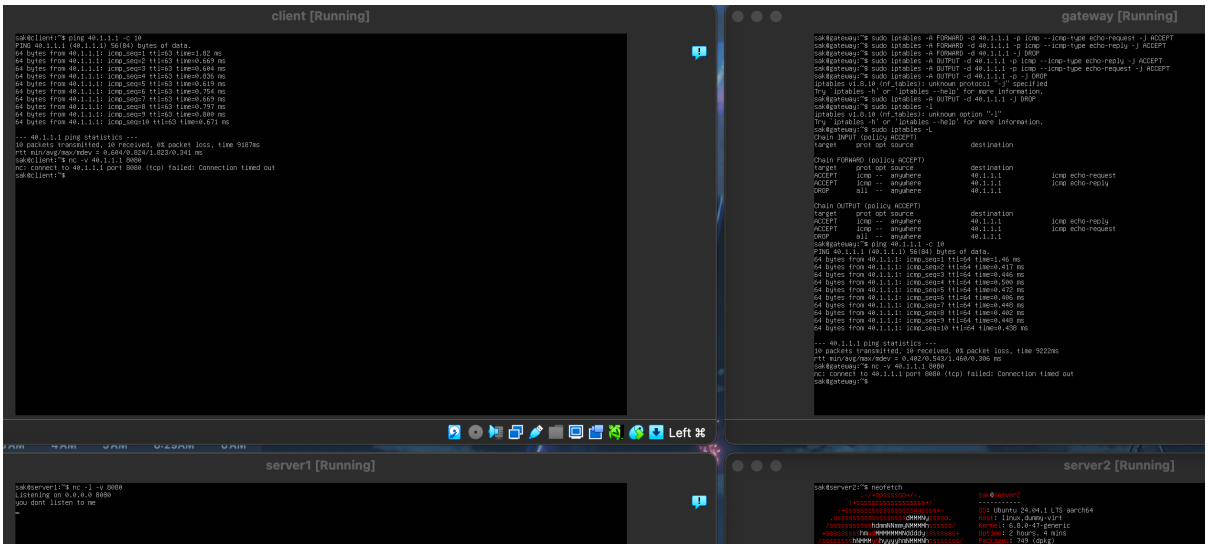
- when using the `ping` command from VM1 to VM3 and VM2 to VM3:

```
sak@client:~$ ping 40.1.1.1 -c 10
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=1.82 ms
64 bytes from 40.1.1.1: icmp_seq=2 ttl=63 time=0.669 ms
64 bytes from 40.1.1.1: icmp_seq=3 ttl=63 time=0.604 ms
64 bytes from 40.1.1.1: icmp_seq=4 ttl=63 time=0.836 ms
64 bytes from 40.1.1.1: icmp_seq=5 ttl=63 time=0.619 ms
64 bytes from 40.1.1.1: icmp_seq=6 ttl=63 time=0.754 ms
64 bytes from 40.1.1.1: icmp_seq=7 ttl=63 time=0.669 ms
64 bytes from 40.1.1.1: icmp_seq=8 ttl=63 time=0.797 ms
64 bytes from 40.1.1.1: icmp_seq=9 ttl=63 time=0.800 ms
64 bytes from 40.1.1.1: icmp_seq=10 ttl=63 time=0.671 ms

--- 40.1.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9187ms
rtt min/avg/max/mdev = 0.604/0.824/1.823/0.341 ms
sak@client:~$
```

```
sak@gateway:~$ ping 40.1.1.1 -c 10
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=64 time=1.46 ms
64 bytes from 40.1.1.1: icmp_seq=2 ttl=64 time=0.417 ms
64 bytes from 40.1.1.1: icmp_seq=3 ttl=64 time=0.446 ms
64 bytes from 40.1.1.1: icmp_seq=4 ttl=64 time=0.500 ms
64 bytes from 40.1.1.1: icmp_seq=5 ttl=64 time=0.472 ms
64 bytes from 40.1.1.1: icmp_seq=6 ttl=64 time=0.406 ms
64 bytes from 40.1.1.1: icmp_seq=7 ttl=64 time=0.448 ms
64 bytes from 40.1.1.1: icmp_seq=8 ttl=64 time=0.402 ms
64 bytes from 40.1.1.1: icmp_seq=9 ttl=64 time=0.448 ms
64 bytes from 40.1.1.1: icmp_seq=10 ttl=64 time=0.438 ms

--- 40.1.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9222ms
rtt min/avg/max/mdev = 0.402/0.543/1.460/0.306 ms
sak@gateway:~$
```

packets are being transmitted and received correctly as requested.

- but when using `nc` command (which uses TCP) to listen on port `8080` on VM3 and trying to connect to it from VM1 and VM2:




```
rtt min/avg/max/mdev = 0.604/0.824/1.823/0.341 ms
sak@client:~$ nc -v 40.1.1.1 8080
nc: connect to 40.1.1.1 port 8080 (tcp) failed: Connection timed out
sak@client:~$
```

```
rtt min/avg/max/mdev = 0.402/0.543/1.460/0.306 ms
sak@gateway:~$ nc -v 40.1.1.1 8080
nc: connect to 40.1.1.1 port 8080 (tcp) failed: Connection timed out
sak@gateway:~$ _
```

```
sak@server1:~$ nc -l -v 8080
Listening on 0.0.0.0 8080
you dont listen to me
_
```

  the connection is not established as requested.

- the same can be verified for UDP traffic using the `nc` command with option `-u`, it gives the same result as TCP traffic.

```
sak@client:~$ nc -u 40.1.1.1 8080
hello
hiii
_
```

```
sak@server1:~$ nc -u -v  -l 8080
Bound on 0.0.0.0 8080
you dont speak either
```

- I have configured the gateway VM to block only TCP traffic initiated by `20.1.1.1/24` using `INPUT` chain and `FORWARD` chain.
  - The `INPUT` chain is used to filter packets that are being sent to the local system. This chain is used when a packet that creates a new connection is being routed to the local system.
  - The following commands were used to configure the IPTables:

```
sudo iptables -A INPUT -s 20.1.1.1 -p tcp -j DROP
sudo iptables -A FORWARD -s 20.1.1.1 -p tcp -j DROP
```

```
sak@gateway:~$ sudo iptables -A INPUT -s 20.1.1.1 -p tcp -j DROP
sak@gateway:~$ sudo iptables -A FORWARD -s 20.1.1.1 -p tcp -j DROP
sak@gateway:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  20.1.1.1             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-request
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-reply
DROP       all  --  anywhere             40.1.1.1
DROP       tcp  --  20.1.1.1             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-reply
ACCEPT     icmp --  anywhere             40.1.1.1             icmp echo-request
DROP       all  --  anywhere             40.1.1.1
sak@gateway:~$ _
```

- when using the `nc` command to listen on port `8080` on VM4 and trying to connect to it from VM1:

```
sak@client:~$ nc -v 40.1.1.3 8080
nc: connect to 40.1.1.3 port 8080 (tcp) failed: Connection timed out
sak@client:~$
```

```
sak@server2:~$ nc -v -l 8080
Listening on 0.0.0.0 8080
doesnt talk still
```

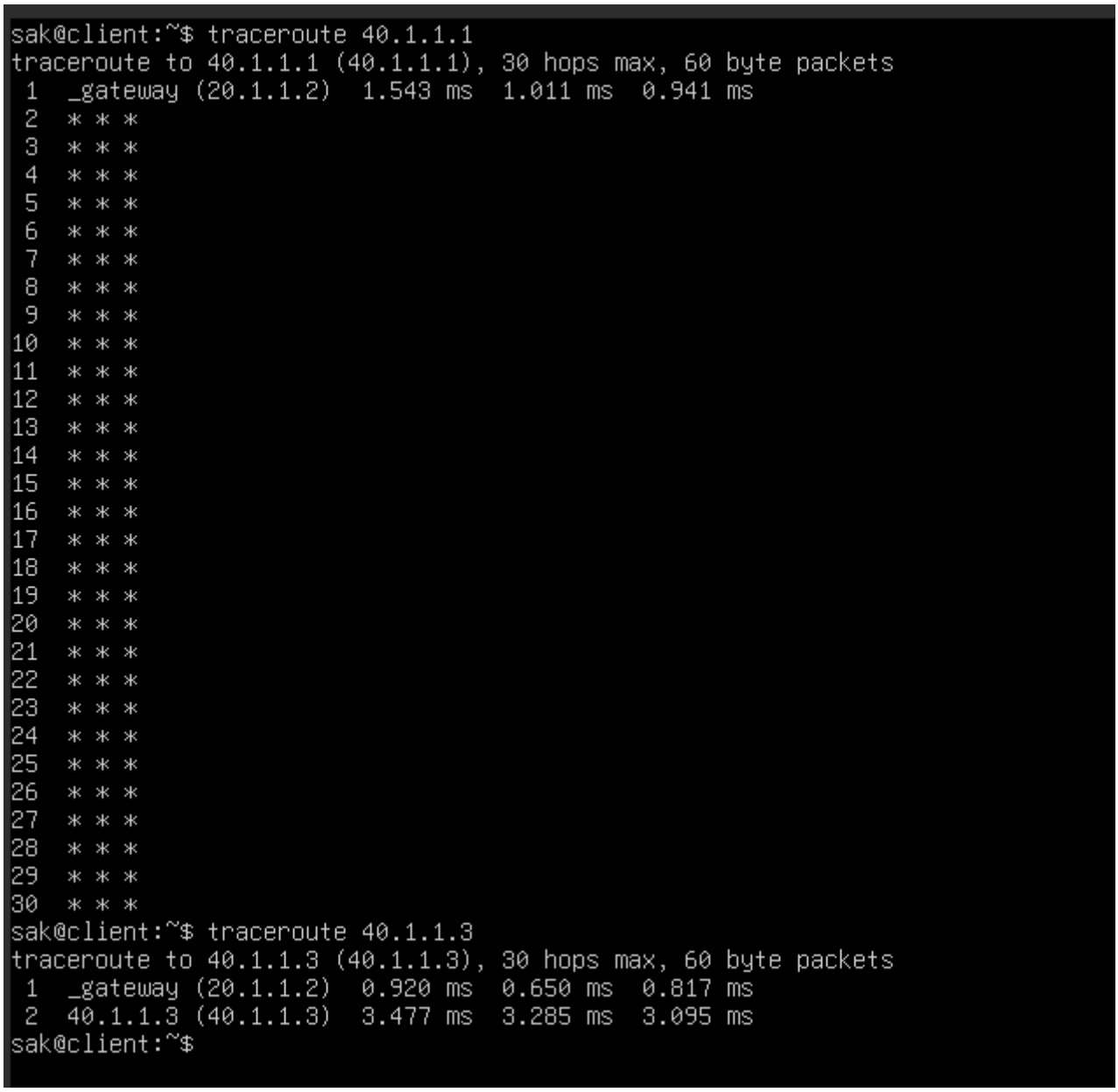
thus blocking the TCP traffic.

- the same can be verified for UDP traffic using the `nc` command with option `-u`, it is not blocked as requested.

```
sak@client:~$ nc -v 40.1.1.3 8080
nc: connect to 40.1.1.3 port 8080 (tcp) failed: Connection timed out
sak@client:~$ nc -v -u 40.1.1.3 8080
Connection to 40.1.1.3 8080 port [udp/*] succeeded!
hwllooww
hii
you finally listened
```

```
sak@server2:~$ nc -v -u -l 8080
Bound on 0.0.0.0 8080
Connection received on 20.1.1.1 58794
XXXXXhwllooww
hii
you finally listened
```

- The above results show that the configurations are correct and the traffic is being filtered correctly as requested.



as shown in the above image, the traffic is blocked by the gateway as seen by the `traceroute` command from VM1 to VM3. Even though it uses `icmp` packets, only ping packets are allowed, and the rest are blocked.

- The rules can be saved using

  ```
  sudo netfilter-persistent save
  ```

  and can be loaded using

  ```
  sudo netfilter-persistent reload
  ```

  to persist the rules even after a reboot.

- The rules can be flushed using

```
sudo iptables -F
```

to remove all the rules.

Q3 - Use the configuration obtained in Q.2. to measure the performance of the gateway

Deliverables

- a) Use “iperf2” tool to test the TCP and UDP bandwidth between 20.1.1.1/24 and 40.1.1.3/24. Attach the screenshot.
- b) What is the minimum, average, and maximum RTT (Attach the screenshot)
  - (i) from 20.1.1.1/24 to 40.1.1.1/24
  - (ii) from 20.1.1.1/24 to 40.1.1.3/24
  - (iii) Did you find a significant difference between (i) and (ii)? If so, why?

Methodology

- I have used the `iperf2` tool to test the TCP and UDP bandwidth between the client `20.1.1.1/24` and the server `40.1.1.3` using the following commands:
  - TCP bandwidth:

    ```
    iperf -s
    ```

    on the server and

    ```
    iperf -c 40.1.1.3
    ```

    on the client.

    ```
    sak@server2:~$ sudo iperf -s
    [sudo] password for sak:
    ------------------------------------------------------------
    Server listening on TCP port 5001
    TCP window size:  128 KByte (default)
    ------------------------------------------------------------
    ```

    ```
    sak@client:~$ sudo iperf -c 40.1.1.3
    [sudo] password for sak:
    ------------------------------------------------------------
    Client connecting to 40.1.1.3, TCP port 5001
    TCP window size: 16.0 KByte (default)
    ------------------------------------------------------------
    ```

  - Since the TCP traffic is blocked for the client, the connection is not established, and no report is generated.

```
sak@server2:~$ sudo iperf -s
[sudo] password for sak:
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
ummmm you there client??
^Csak@server2:~$
```

```
sak@client:~$ sudo iperf -c 40.1.1.3
[sudo] password for sak:
------------------------------------------------------------
Client connecting to 40.1.1.3, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
hello
hii
doesnt do jack shit
^C^Csak@client:~$ _
```

- UDP bandwidth:

  ```
  iperf -s -u
  ```

  on the server and

  ```
  iperf -c 40.1.1.3 -u
  ```

  on the client.

- The following results are obtained:

```
sak@server2:~$ sudo 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 49614
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0154 sec  1.25 MBytes  1.05 Mbits/sec   0.100 ms 0/895 (0%)
^Csak@server2:~$ _
```

```
sak@client:~$ sudo iperf -c 40.1.1.3 -u
------------------------------------------------------------
Client connecting to 40.1.1.3, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 20.1.1.1 port 49614 connected with 40.1.1.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0154 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0154 sec  1.25 MBytes  1.05 Mbits/sec   0.099 ms 0/895 (0%)
sak@client:~$ _
```

    - The above results show that the UDP traffic is not blocked and the bandwidth is measured correctly. The `iperf` tool sends `895` packets of size `1470` bytes each, in nearly `10` seconds, giving a transfer size of `1.24 MBytes` and a bandwidth of `1.05 Mbits/sec`.

- For measuring the RTT, I have used the `ping` command from the client to servers for `100` packets and then calculated the minimum, average, and maximum RTT.

    - from `20.1.1.1/24` to `40.1.1.1/24`:

```
--- 40.1.1.1 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 100860ms
rtt min/avg/max/mdev = 0.599/0.870/3.859/0.351 ms
sak@client:~$ _
```

      min RTT : `0.599 ms`, avg RTT : `0.870 ms`, max RTT : `3.859 ms`, loss : `0%`

    - from `20.1.1.1/24` to `40.1.1.3/24`:

```
--- 40.1.1.3 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 100786ms
rtt min/avg/max/mdev = 0.624/0.827/2.032/0.208 ms
sak@client:~$
```

      min RTT : `0.624 ms`, avg RTT : `0.827 ms`, max RTT : `2.032 ms`, loss : `0%`

    - The above results show that there is no significant difference between the RTT of the two servers, as the RTT is nearly the same for both the servers. This is because the traffic is being forwarded correctly, and there are no rules to block `icmp` traffic between the client and the servers. And since both the servers are in the same network, and both are connected to the same gateway and one jump away, the RTT is nearly the same for both the servers.

Q4 - Network address translation at the gateway VM

Deliverables

- a) Change the source IP address of every packet from 20.1.1.1/24 to 40.1.1.2/24
- b) When the packet response for the packet from step “a” arrives at the gateway, revert the destination IP address to the original.
- c) Validate the above by sending traffic and observing the packets at each VM using Wireshark/tcpdump. Attach the screenshot.

Methodology

- I have used IPTables to configure the gateway VM to change the source IP address of every packet from `20.1.1.1/24` to `40.1.1.2/24` using the `POSTROUTING` chain.

    - The `POSTROUTING` chain is used to filter packets that are being sent out of the local system. This chain is used when a packet that creates a new connection is being routed from the local system.
    - The following commands were used to configure the IPTables:

```
sudo iptables -t nat -A POSTROUTING -s 20.1.1.1 -j SNAT --to-source 40.1.1.2
```

- Now when the packet response for the packet from step “a” arrives at the gateway, the destination IP address is reverted to the original using the `PREROUTING` chain.
    - The `PREROUTING` chain is used to filter packets that are being routed through the local system. This chain is used when a packet that creates a new connection is being routed through the local system.
    - The following commands were used to configure the IPTables:

```
sudo iptables -t nat -A PREROUTING -d 40.1.1.2 -j DNAT --to-destination 20.1.1.1
```

```
sak@gateway:~$ sudo iptables -t nat -A POSTROUTING -s 20.1.1.1 -j SNAT --to-source 40.1.1.2
sak@gateway:~$ sudo iptables -t nat -A PREROUTING -d 40.1.1.2 -j DNAT --to-destination 20.1.1.1
sak@gateway:~$ sudo iptables -t nat -L --line-numbers
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    DNAT       all  --  anywhere             gateway              to:20.1.1.1

Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    SNAT       all  --  20.1.1.1             anywhere             to:40.1.1.2
sak@gateway:~$ _
```

- Once these rules are applied, we can verify its working by running `tcpdump` on the client, server VMs and using a simple command like `ping` only.
    - on the client VM (for pinging from server2 to client):

```
sak@server2:~$ ping 20.1.1.1 -c 10
PING 20.1.1.1 (20.1.1.1) 56(84) bytes of data.
64 bytes from 20.1.1.1: icmp_seq=1 ttl=63 time=1.78 ms
64 bytes from 20.1.1.1: icmp_seq=2 ttl=63 time=1.39 ms
64 bytes from 20.1.1.1: icmp_seq=3 ttl=63 time=0.757 ms
64 bytes from 20.1.1.1: icmp_seq=4 ttl=63 time=0.789 ms
64 bytes from 20.1.1.1: icmp_seq=5 ttl=63 time=0.917 ms
64 bytes from 20.1.1.1: icmp_seq=6 ttl=63 time=1.36 ms
64 bytes from 20.1.1.1: icmp_seq=7 ttl=63 time=0.882 ms
64 bytes from 20.1.1.1: icmp_seq=8 ttl=63 time=0.773 ms
64 bytes from 20.1.1.1: icmp_seq=9 ttl=63 time=0.807 ms
64 bytes from 20.1.1.1: icmp_seq=10 ttl=63 time=0.866 ms

--- 20.1.1.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9155ms
rtt min/avg/max/mdev = 0.757/1.032/1.781/0.333 ms
sak@server2:~$
```

```
sak@client:~$ sudo tcpdump -i ens34
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens34, link-type EN10MB (Ethernet), snapshot length 262144 bytes
11:44:52.767346 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 1, length 64
11:44:52.767525 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 1, length 64
11:44:53.768674 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 2, length 64
11:44:53.768697 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 2, length 64
11:44:54.770545 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 3, length 64
11:44:54.770566 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 3, length 64
11:44:55.794112 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 4, length 64
11:44:55.794137 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 4, length 64
11:44:56.817880 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 5, length 64
11:44:56.817905 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 5, length 64
11:44:57.841824 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 6, length 64
11:44:57.841872 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 6, length 64
11:44:58.052963 ARP, Request who-has _gateway tell client, length 28
11:44:58.053325 ARP, Reply _gateway is-at 08:00:27:42:19:82 (oui Unknown), length 46
11:44:58.174988 ARP, Request who-has client tell _gateway, length 46
11:44:58.175022 ARP, Reply client is-at 08:00:27:30:43:b8 (oui Unknown), length 28
11:44:58.842704 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 7, length 64
11:44:58.842726 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 7, length 64
11:44:59.889562 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 8, length 64
11:44:59.889585 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 8, length 64
11:45:00.914570 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 9, length 64
11:45:00.914592 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 9, length 64
11:45:01.938621 IP 40.1.1.3 > client: ICMP echo request, id 2382, seq 10, length 64
11:45:01.938647 IP client > 40.1.1.3: ICMP echo reply, id 2382, seq 10, length 64
```

when the packet is sent from server2 to the client, the IP address of the source and destination are not changed and `ping` command works as usual.

- on the server2 VM (for pinging from client to server2):

```
sak@client:~$ ping 40.1.1.3 -c 10
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=1.41 ms
64 bytes from 40.1.1.3: icmp_seq=2 ttl=63 time=0.919 ms
64 bytes from 40.1.1.3: icmp_seq=3 ttl=63 time=0.796 ms
64 bytes from 40.1.1.3: icmp_seq=4 ttl=63 time=0.676 ms
64 bytes from 40.1.1.3: icmp_seq=5 ttl=63 time=0.951 ms
64 bytes from 40.1.1.3: icmp_seq=6 ttl=63 time=0.980 ms
64 bytes from 40.1.1.3: icmp_seq=7 ttl=63 time=0.829 ms
64 bytes from 40.1.1.3: icmp_seq=8 ttl=63 time=0.757 ms
64 bytes from 40.1.1.3: icmp_seq=9 ttl=63 time=0.805 ms
64 bytes from 40.1.1.3: icmp_seq=10 ttl=63 time=0.675 ms

--- 40.1.1.3 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9088ms
rtt min/avg/max/mdev = 0.675/0.879/1.407/0.202 ms
sak@client:~$
```

```
sak@server2:~$ sudo tcpdump -i ens34
[sudo] password for sak:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens34, link-type EN10MB (Ethernet), snapshot length 262144 bytes
11:47:26.598894 IP _gateway > server2: ICMP echo request, id 2663, seq 1, length 64
11:47:26.598947 IP server2 > _gateway: ICMP echo reply, id 2663, seq 1, length 64
11:47:27.601082 IP _gateway > server2: ICMP echo request, id 2663, seq 2, length 64
11:47:27.601106 IP server2 > _gateway: ICMP echo reply, id 2663, seq 2, length 64
11:47:28.602826 IP _gateway > server2: ICMP echo request, id 2663, seq 3, length 64
11:47:28.602847 IP server2 > _gateway: ICMP echo reply, id 2663, seq 3, length 64
11:47:29.606471 IP _gateway > server2: ICMP echo request, id 2663, seq 4, length 64
11:47:29.606494 IP server2 > _gateway: ICMP echo reply, id 2663, seq 4, length 64
11:47:30.631067 IP _gateway > server2: ICMP echo request, id 2663, seq 5, length 64
11:47:30.631104 IP server2 > _gateway: ICMP echo reply, id 2663, seq 5, length 64
11:47:31.632947 IP _gateway > server2: ICMP echo request, id 2663, seq 6, length 64
11:47:31.632977 IP server2 > _gateway: ICMP echo reply, id 2663, seq 6, length 64
11:47:31.966915 ARP, Request who-has server2 tell _gateway, length 46
11:47:31.966929 ARP, Reply server2 is-at 08:00:27:56:53:85 (oui Unknown), length 28
11:47:32.077988 ARP, Request who-has _gateway tell server2, length 28
11:47:32.078353 ARP, Reply _gateway is-at 08:00:27:96:05:c6 (oui Unknown), length 46
11:47:32.634040 IP _gateway > server2: ICMP echo request, id 2663, seq 7, length 64
11:47:32.634068 IP server2 > _gateway: ICMP echo reply, id 2663, seq 7, length 64
11:47:33.639173 IP _gateway > server2: ICMP echo request, id 2663, seq 8, length 64
11:47:33.639195 IP server2 > _gateway: ICMP echo reply, id 2663, seq 8, length 64
11:47:34.662785 IP _gateway > server2: ICMP echo request, id 2663, seq 9, length 64
11:47:34.662805 IP server2 > _gateway: ICMP echo reply, id 2663, seq 9, length 64
11:47:35.686441 IP _gateway > server2: ICMP echo request, id 2663, seq 10, length 64
11:47:35.686462 IP server2 > _gateway: ICMP echo reply, id 2663, seq 10, length 64
```

but when the packet is sent from the client to server2, the IP address of the source is changed to `40.1.1.2 (_gateway)` and every echo reply packet has the destination IP address changed to `40.1.1.2 (_gateway)`. Even with these changes of destinations, the `ping` command works as usual with no loss of packets.

- on the gateway VM (for pinging from client to server2):
  - for interface `ens34`:

```
sak@gateway:~$ sudo tcpdump -i ens34 -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens34, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:01:31.495873 IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2690, seq 1, length 64
12:01:31.497024 IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2690, seq 1, length 64
12:01:32.497298 IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2690, seq 2, length 64
12:01:32.497733 IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2690, seq 2, length 64
12:01:33.555543 IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2690, seq 3, length 64
12:01:33.555877 IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2690, seq 3, length 64
12:01:34.579667 IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2690, seq 4, length 64
12:01:34.580222 IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2690, seq 4, length 64
12:01:35.581584 IP 20.1.1.1 > 40.1.1.3: ICMP echo request, id 2690, seq 5, length 64
12:01:35.582315 IP 40.1.1.3 > 20.1.1.1: ICMP echo reply, id 2690, seq 5, length 64
12:01:36.499738 ARP, Request who-has 20.1.1.2 tell 20.1.1.1, length 46
12:01:36.499753 ARP, Reply 20.1.1.2 is-at 08:00:27:42:19:82, length 28
12:01:36.614148 ARP, Request who-has 20.1.1.1 tell 20.1.1.2, length 28
12:01:36.614532 ARP, Reply 20.1.1.1 is-at 08:00:27:30:43:b8, length 46
```

    the client side port of the gateway doesn't show any changes in the IP address of the packets.

- for interface `ens36`:

```
sak@gateway:~$ sudo tcpdump -i ens36 -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens36, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:02:55.480153 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 2691, seq 1, length 64
12:02:55.481785 IP 40.1.1.3 > 40.1.1.2: ICMP echo reply, id 2691, seq 1, length 64
12:02:56.480810 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 2691, seq 2, length 64
12:02:56.481176 IP 40.1.1.3 > 40.1.1.2: ICMP echo reply, id 2691, seq 2, length 64
12:02:57.524163 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 2691, seq 3, length 64
12:02:57.524523 IP 40.1.1.3 > 40.1.1.2: ICMP echo reply, id 2691, seq 3, length 64
12:02:58.548452 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 2691, seq 4, length 64
12:02:58.548870 IP 40.1.1.3 > 40.1.1.2: ICMP echo reply, id 2691, seq 4, length 64
12:02:59.549505 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 2691, seq 5, length 64
12:02:59.549793 IP 40.1.1.3 > 40.1.1.2: ICMP echo reply, id 2691, seq 5, length 64
12:03:00.582086 ARP, Request who-has 40.1.1.3 tell 40.1.1.2, length 28
12:03:00.582472 ARP, Reply 40.1.1.3 is-at 08:00:27:56:53:85, length 46
12:03:00.669641 ARP, Request who-has 40.1.1.2 tell 40.1.1.3, length 46
12:03:00.669657 ARP, Reply 40.1.1.2 is-at 08:00:27:96:05:c6, length 28
```

  the server side port of the gateway shows the changes in the IP address of the packets as expected. Hence all the changes and NAT process happens during the forwarding of the packets from the client to the server2 and is reverted back when the response is sent back to the client.
- The IP address changes are done correctly and the packets are forwarded correctly, as seen by the `ping` command from the client to server2.

```
sak@client:~$ ping 40.1.1.3 -c 5
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=1.90 ms
64 bytes from 40.1.1.3: icmp_seq=2 ttl=63 time=0.884 ms
64 bytes from 40.1.1.3: icmp_seq=3 ttl=63 time=0.672 ms
64 bytes from 40.1.1.3: icmp_seq=4 ttl=63 time=0.993 ms
64 bytes from 40.1.1.3: icmp_seq=5 ttl=63 time=1.22 ms

--- 40.1.1.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.672/1.134/1.901/0.422 ms
sak@client:~$ ping 40.1.1.3 -c 5
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=3.27 ms
64 bytes from 40.1.1.3: icmp_seq=2 ttl=63 time=0.829 ms
64 bytes from 40.1.1.3: icmp_seq=3 ttl=63 time=0.774 ms
64 bytes from 40.1.1.3: icmp_seq=4 ttl=63 time=0.841 ms
64 bytes from 40.1.1.3: icmp_seq=5 ttl=63 time=0.661 ms

--- 40.1.1.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4070ms
rtt min/avg/max/mdev = 0.661/1.275/3.270/0.999 ms
```

- The same results can be obtained using `tshark` command to capture the packets and analyze them, and they will be similar if communication is done using `nc` commmand or between client and server1.

Q5 - Load balancing at the gateway VM

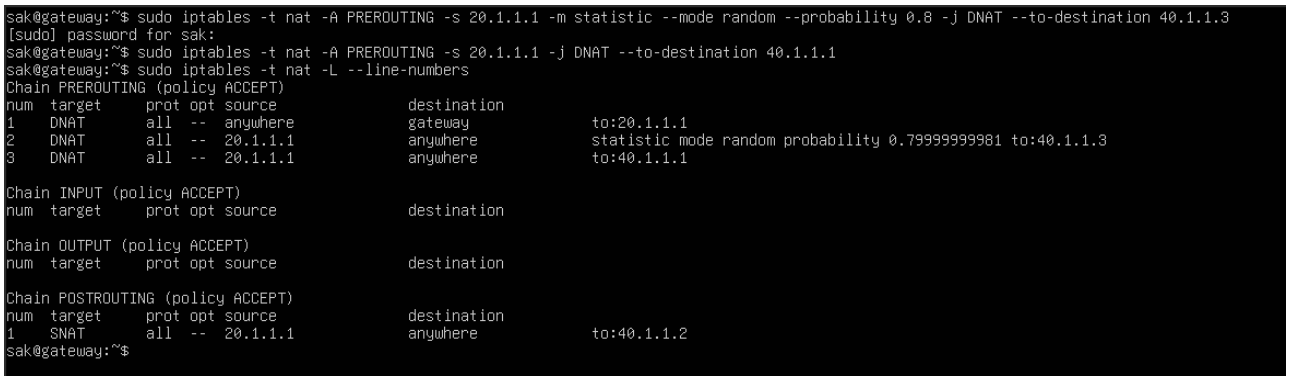
Deliverables

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

Methodology

- Using IP tables, load balancing can be achieved by using the `PREROUTING` chain and `DNAT` target to change the destination IP address of the packets.
- The server2 is assigned a higher probability of `0.8` and server1 is assigned a lower probability of `0.2` as requested.
- The following commands were used to configure the IPTables:

```
sudo iptables -t nat -A PREROUTING -s 20.1.1.1 -m statistic --mode
random --probability 0.8 -j DNAT --to-destination 40.1.1.3
sudo iptables -t nat -A PREROUTING -s 20.1.1.1 -j DNAT --to-
destination 40.1.1.1
```



- Once these rules are applied, we can verify its working by running `tcpdump` on the server VMs and using `ping` commands on client VM.
- On the server VMs, run the command

```
sudo tcpdump -i ens34 -n icmp and src host 40.1.1.2
```

and on the client VM, run the command

```
for i in {1..100}; do
  ping -c 1
done
```

- The following results are obtained:

- on server1 VM:

```
sak@server1:~$ sudo tcpdump -i ens34 -n icmp and src host 40.1.1.2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens34, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:15:53.737977 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3529, seq 1, length 64
14:15:53.877630 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3550, seq 1, length 64
14:15:53.951299 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3561, seq 1, length 64
14:15:53.989188 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3567, seq 1, length 64
14:15:53.995934 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3568, seq 1, length 64
14:15:54.004052 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3569, seq 1, length 64
14:15:54.032479 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3574, seq 1, length 64
14:15:54.083941 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3584, seq 1, length 64
14:15:54.102063 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3587, seq 1, length 64
14:15:54.120457 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3590, seq 1, length 64
14:15:54.127629 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3591, seq 1, length 64
14:15:54.182979 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3597, seq 1, length 64
14:15:54.194878 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3599, seq 1, length 64
14:15:54.223463 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3603, seq 1, length 64
14:15:54.262970 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3610, seq 1, length 64
14:15:54.285241 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3614, seq 1, length 64
14:15:54.298041 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3616, seq 1, length 64
14:15:54.354902 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3625, seq 1, length 64
14:15:54.370702 IP 40.1.1.2 > 40.1.1.1: ICMP echo request, id 3627, seq 1, length 64
^C
19 packets captured
19 packets received by filter
0 packets dropped by kernel
sak@server1:~$
```

  the server1 receives only `19` packets out of `100` packets, which is around `0.2` probability as expected.

- on server2 VM:

```
14:15:54.845142 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 3621, seq 1, length 64
14:15:54.850611 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 3622, seq 1, length 64
14:15:54.855868 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 3623, seq 1, length 64
14:15:54.866362 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 3624, seq 1, length 64
14:15:54.885618 IP 40.1.1.2 > 40.1.1.3: ICMP echo request, id 3626, seq 1, length 64
^C
81 packets captured
81 packets received by filter
0 packets dropped by kernel
sak@server2:~$ _
```

  the server2 receives `81` packets out of `100` packets, as expected.

- The below table shows the packets received by each server for different trials.

Packets Sent	Server1	Server2
100	19	81
100	21	79
100	17	83
100	22	78
100	18	82
100	21	79
-	-	-
Average	**19.6**	**80.4**

- The above results show that the load balancing is working correctly and the packets are being forwarded to the servers based on the probabilities assigned.