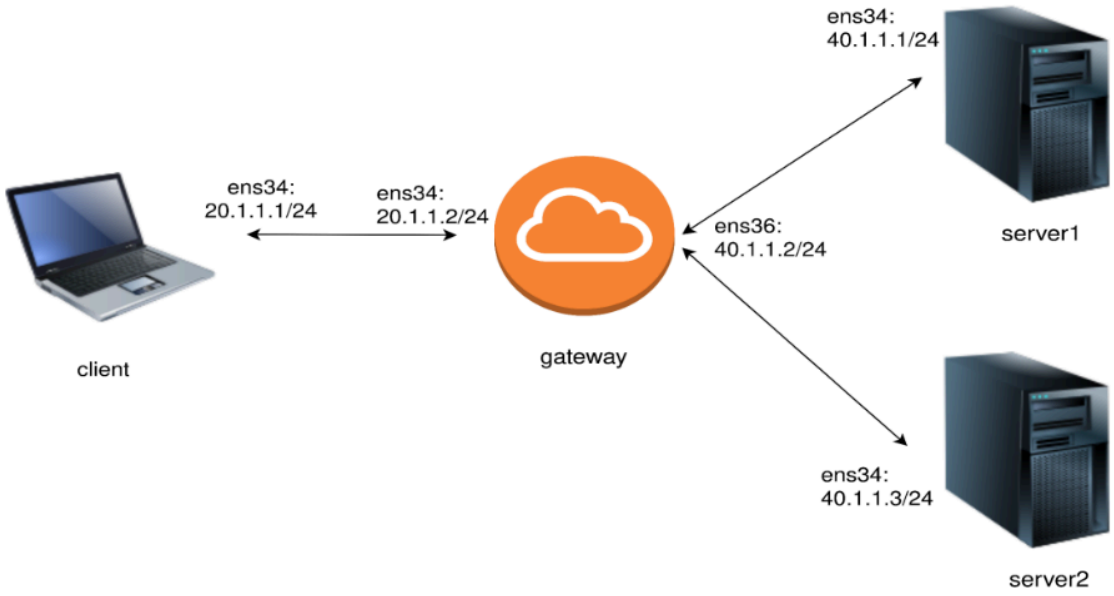
CSE 232: Programming Assignment 3
Using Linux iptables

Himanshu Raj (2022216)

October 27, 2024

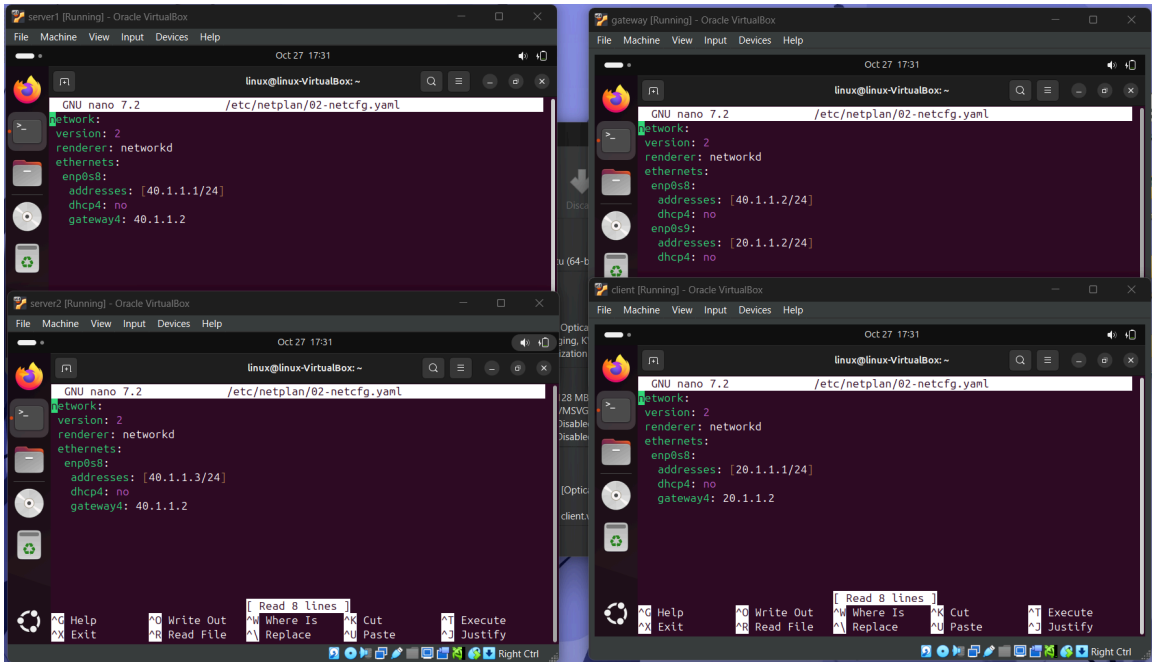
This assignment allows you to gain hands-on experience with `iptables` for NAT, firewall, port forwarding, and load balancing in a multi-VM environment.



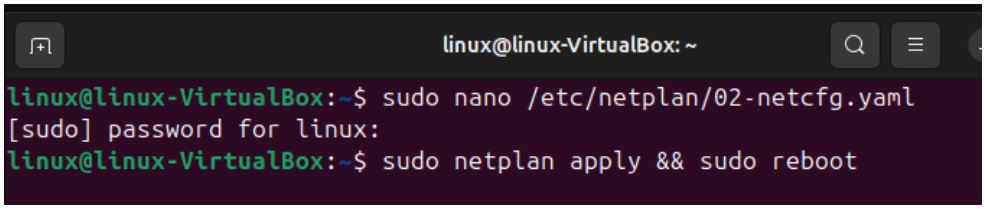
Q.1. Multi VM setup

a) Configure the IP addresses and routes for all VMs.

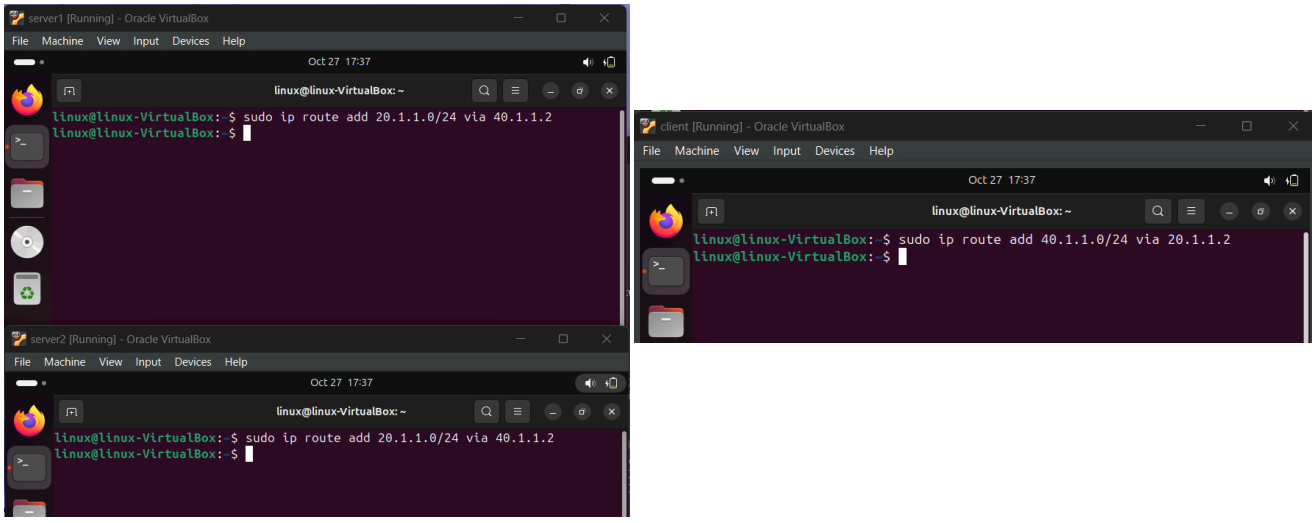
- Ensure the network interface is enabled, if not use
  ip link set <interface name> up
- Create <02-netcfg.yaml> file in </etc/netplan> in all VMs using
  sudo nano /etc/netplan/02-netcfg.yaml
- Enter the content shown in the screenshot below for each VM and save the yaml files. Enter your interface name if it is different from enp0s8 and enp0s9.



- Apply changes using **sudo netplan apply** and **sudo reboot** .

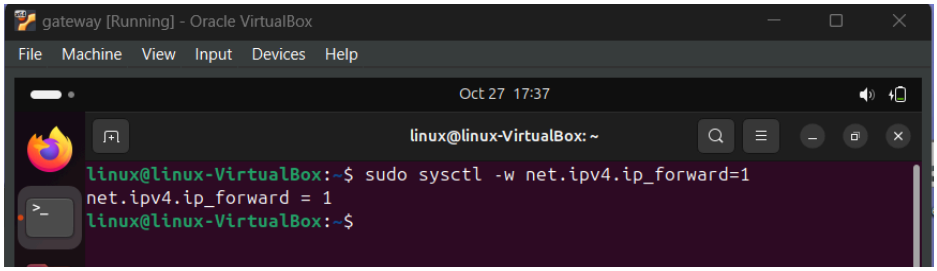


- Add an entry as shown in screenshots in the routing table to allow communication between VMs. (no entry for gateway, only for servers and client)
  sudo ip route add 20.1.1.0/24 via 40.1.1.2 (for both servers)
  sudo ip route add 40.1.1.0/24 via 20.1.1.2 (for client)

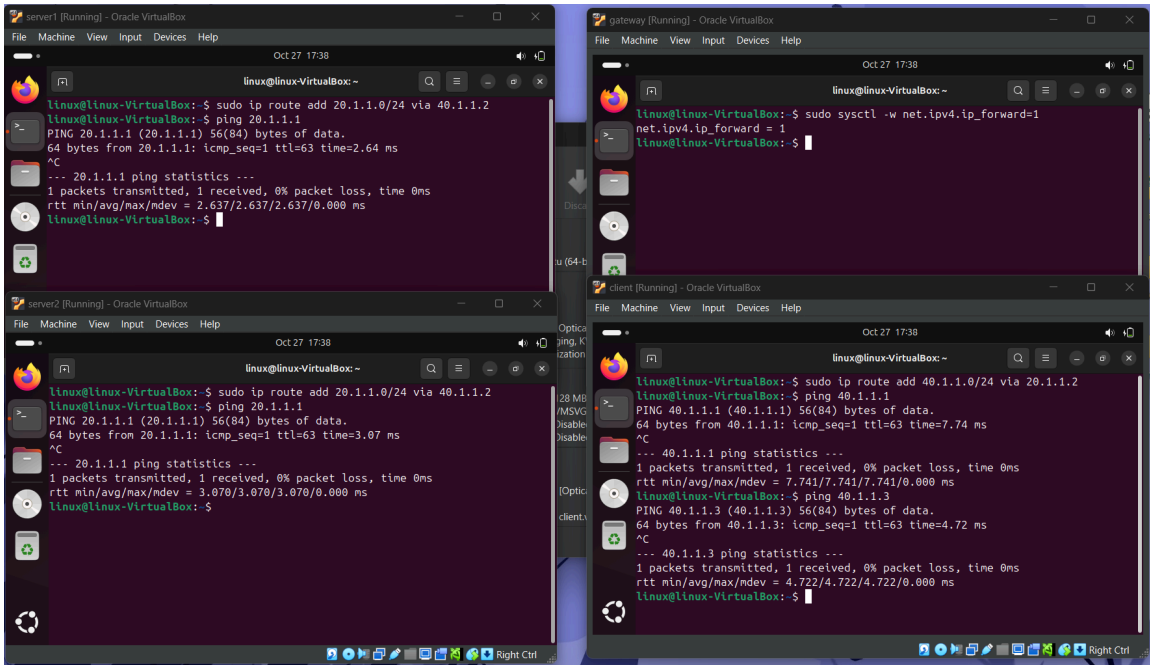


b) Add forwarding functionality to gateway VM.

Use the command below on the gateway VM terminal to add forwarding functionality.
sudo sysctl -w net.ipv4.ip_forward=1



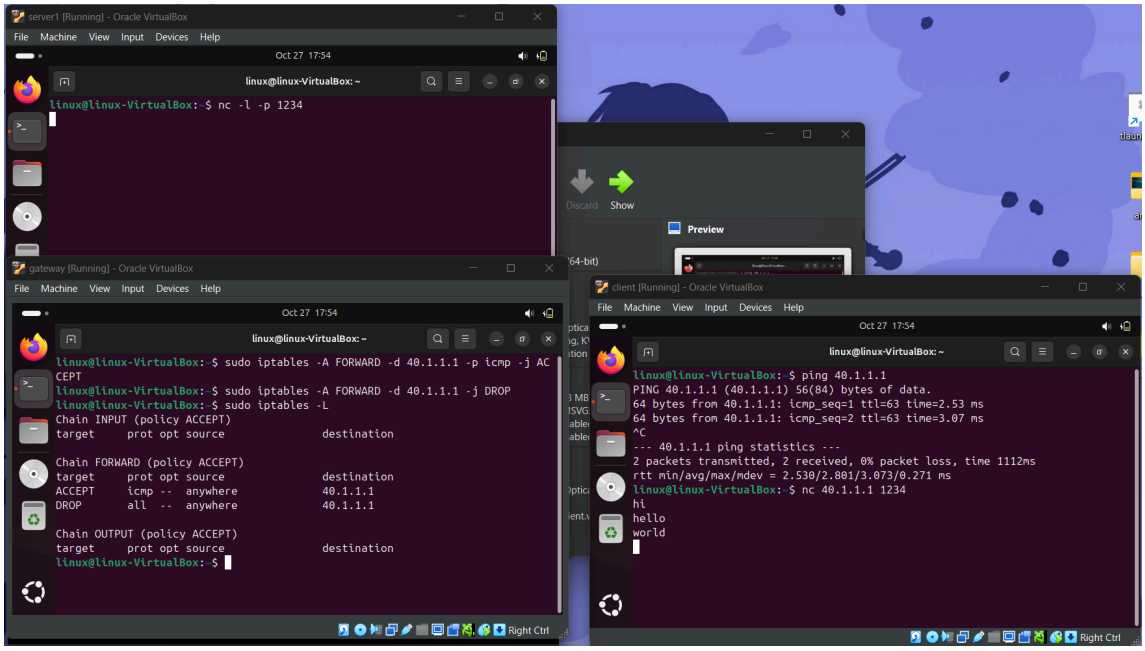
The screenshot below shows that after adding forwarding, the client is able to communicate with servers.



Q.2. Traffic filtering at the gateway VM

a) The gateway must block all traffic (except for ping) destined to the server 40.1.1.1/24.

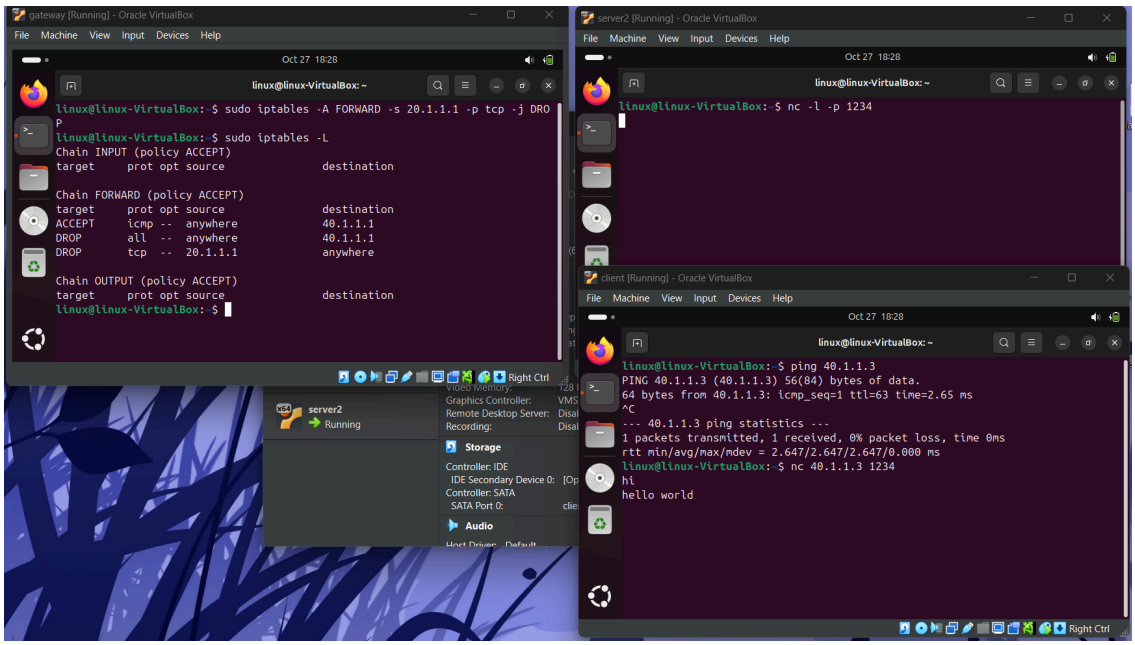
This is done by executing the following commands on the gateway VM terminal.
sudo iptables -A FORWARD -d 40.1.1.1 -p icmp -j ACCEPT (this command means accept packets destined to 40.1.1.1 if it follows icmp protocol, ping uses icmp)
sudo iptables -A FORWARD -d 40.1.1.1 -j DROP (this command drops all packets destined to 40.1.1.1 unless they follow icmp protocol which is accepted by above rule)



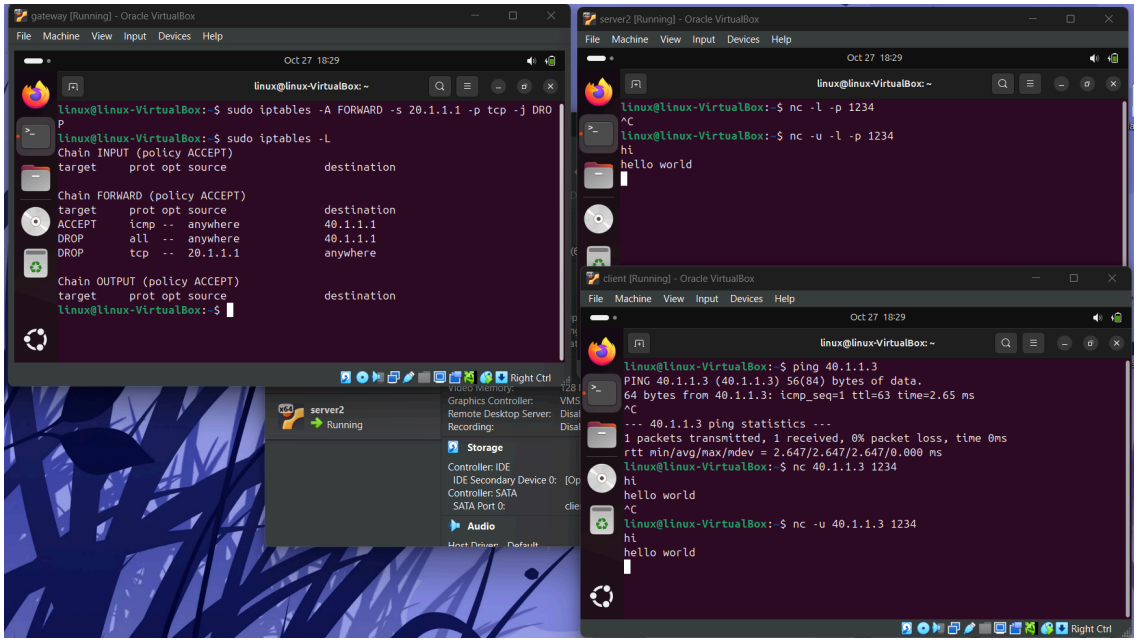
This screenshot shows that the client can only ping server1, but no other traffic (tcp/udp) is allowed.

b) The gateway must block only TCP traffic initiated by 20.1.1.1/24.

This is done by executing the following command on the gateway VM terminal. **sudo iptables -A FORWARD -s 20.1.1.1 -p tcp -j DROP** (this command means drop packet from source 20.1.1.1 that follows tcp protocol)



This screenshot shows that we can ping server2, but we can't communicate over a tcp connection.

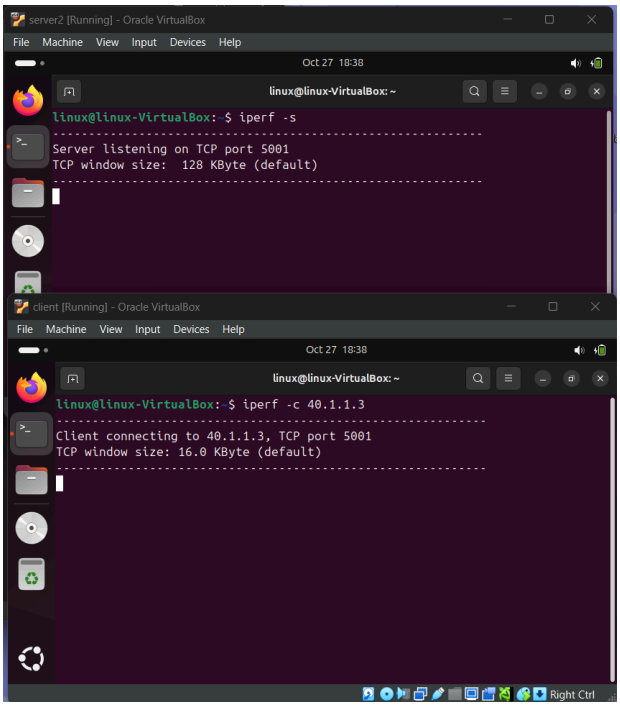


This screenshot shows that we can communicate with server2 using a udp connection, signifying only tcp traffic is blocked via gateway.

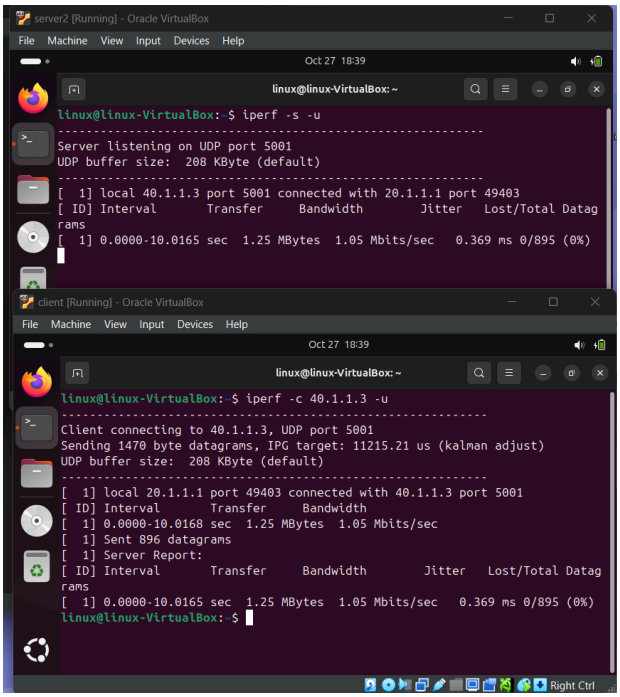
Q.3. Use the configuration obtained in Q.2. for this question

a) Use “iperf2” tool to test the TCP and UDP bandwidth between 20.1.1.1/24 and 40.1.1.3/24.

Because we are blocking tcp traffic initiated by 20.1.1.1 (client), we don’t see any records or logs by iperf with the command.

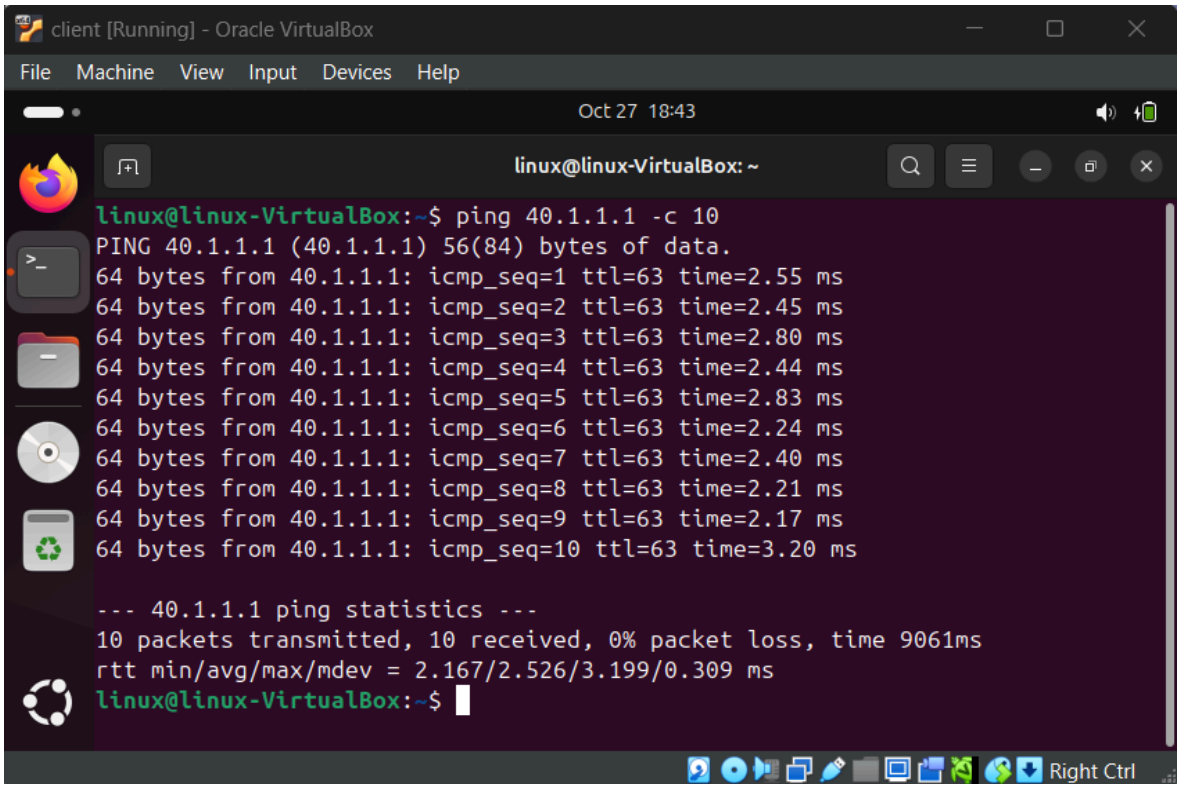


We see records and logs by iperf with the command in case of udp connection.
The amount of data transferred is 1.25 MB.
The bandwidth of the connection is 1.05 Mbps.



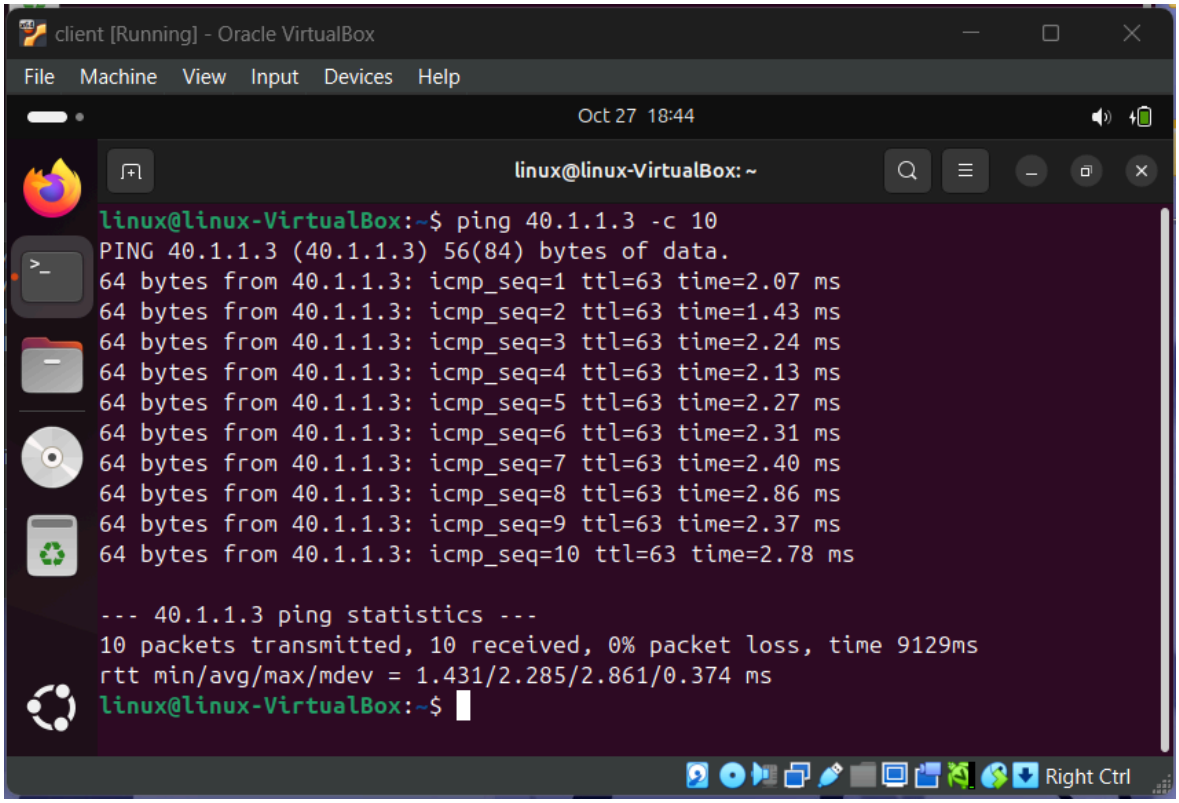
b) Find the minimum, average, and maximum RTT:

(i) from 20.1.1.1/24 to 40.1.1.1/24



min RTT = 2.167 ms, avg RTT = 2.536ms, max RTT = 3.199 ms

(ii) from 20.1.1.1/24 to 40.1.1.3/24



min RTT = 1.431 ms, avg RTT = 2.285ms, max RTT = 2.861 ms

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

A very small difference is observed in RTT values.
The slightly more time taken in the case of server1 (40.1.1.1) can be explained by the time taken to process and forward packets at the gateway due to the filtering rules we applied in Q.2.a where we check the protocol of every packet destined for 40.1.1.1 and then forward the packet.

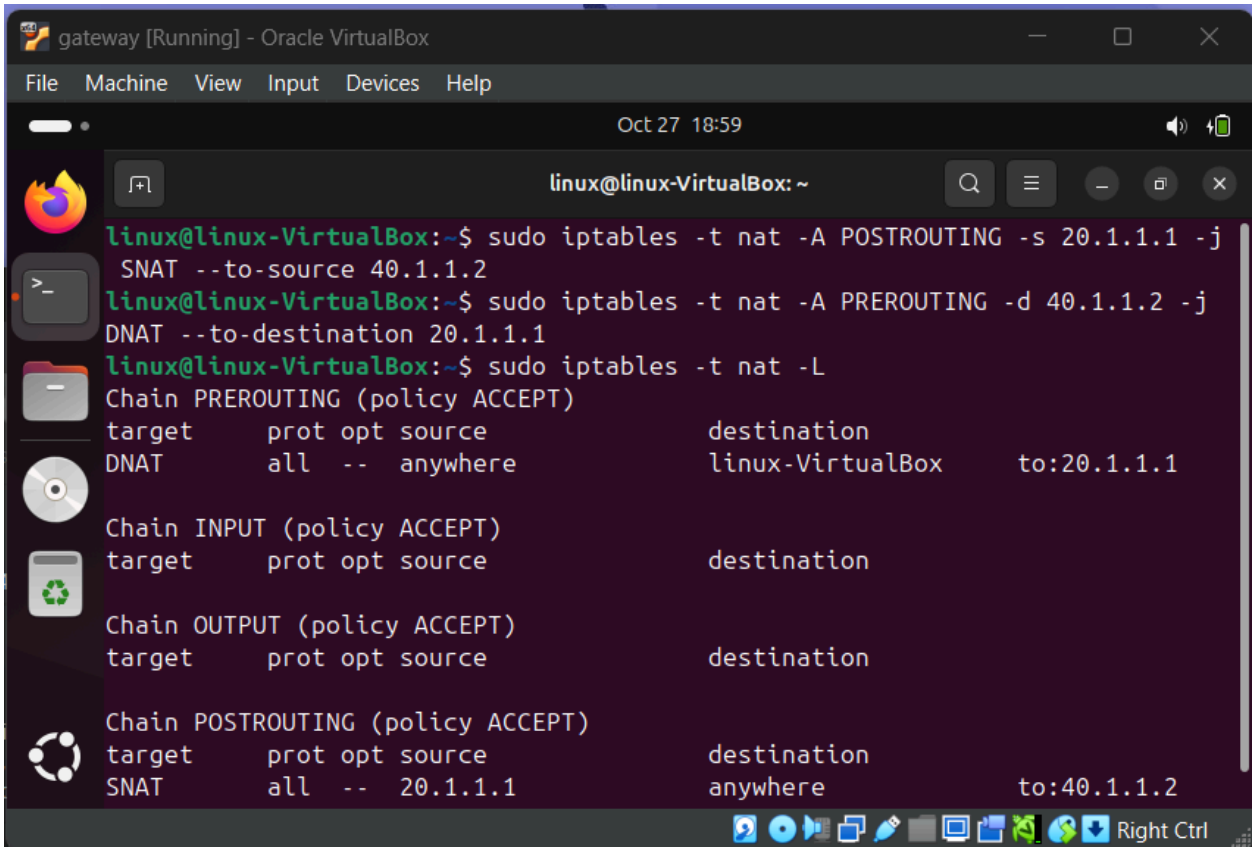
Q.4. Network address translation at the gateway VM

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

This is done by executing the following command on the gateway VM terminal.
sudo iptables -t nat -A POSTROUTING -s 20.1.1.1 -j SNAT --to-source 40.1.1.2
(this command means that translate the source address of every outgoing packet from source 20.1.1.1 to 40.1.1.2)

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

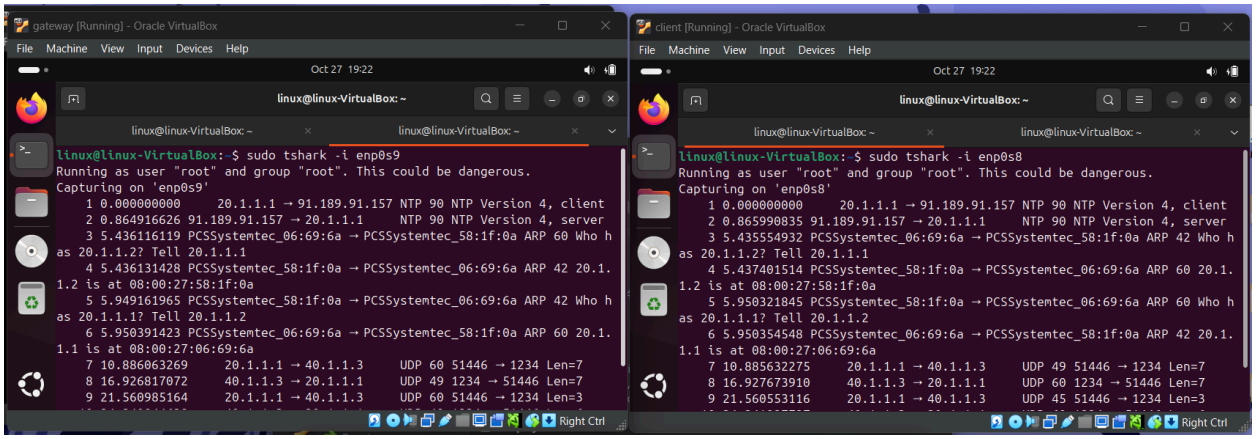
This is done by executing the following command on the gateway VM terminal.
sudo iptables -t nat -A PREROUTING -d 40.1.1.2 -j DNAT --to-destination 20.1.1.1
(this command means that translate the destination address of every incoming packet destined to 40.1.1.2 to 20.1.1.1)



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

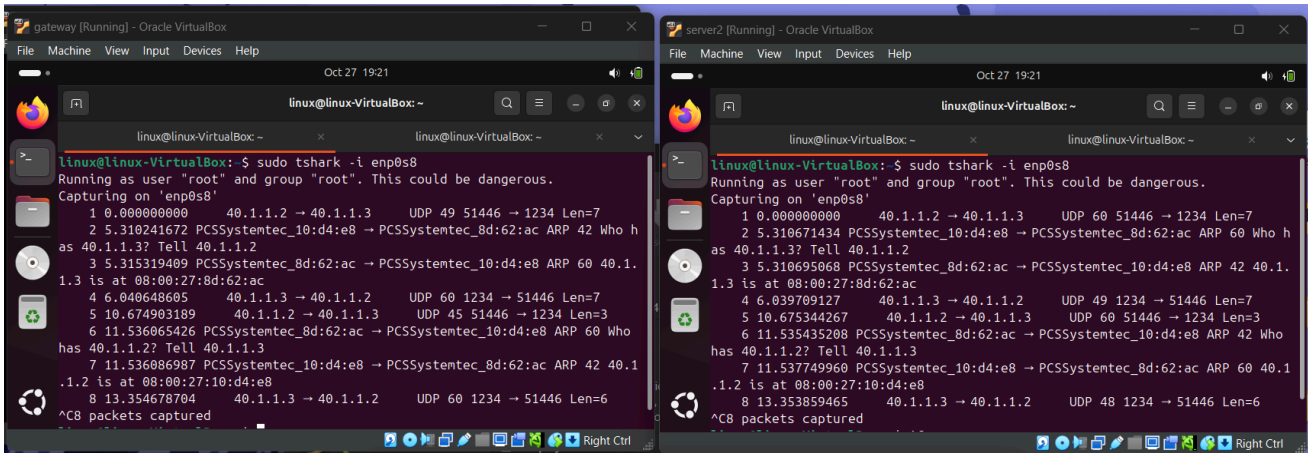
The enp0s8 adapter on the gateway is configured for the servers, and the enp0s9 adapter on the gateway is configured for the client.
The screenshots below show the packets captured by tshark during a udp connection between server2 and the client.

tshark output for client



we can see that the client sends requests to server2 with source address 20.1.1.1 and receives replies from the server with destination address 20.1.1.1

tshark output for server



we can see that server2 receives request packets with the source address as 40.1.1.2 instead of 20.1.1.1 and sends response packets with the destination address as 40.1.1.2 which is then translated to 20.1.1.1 as shown in the previous screenshot

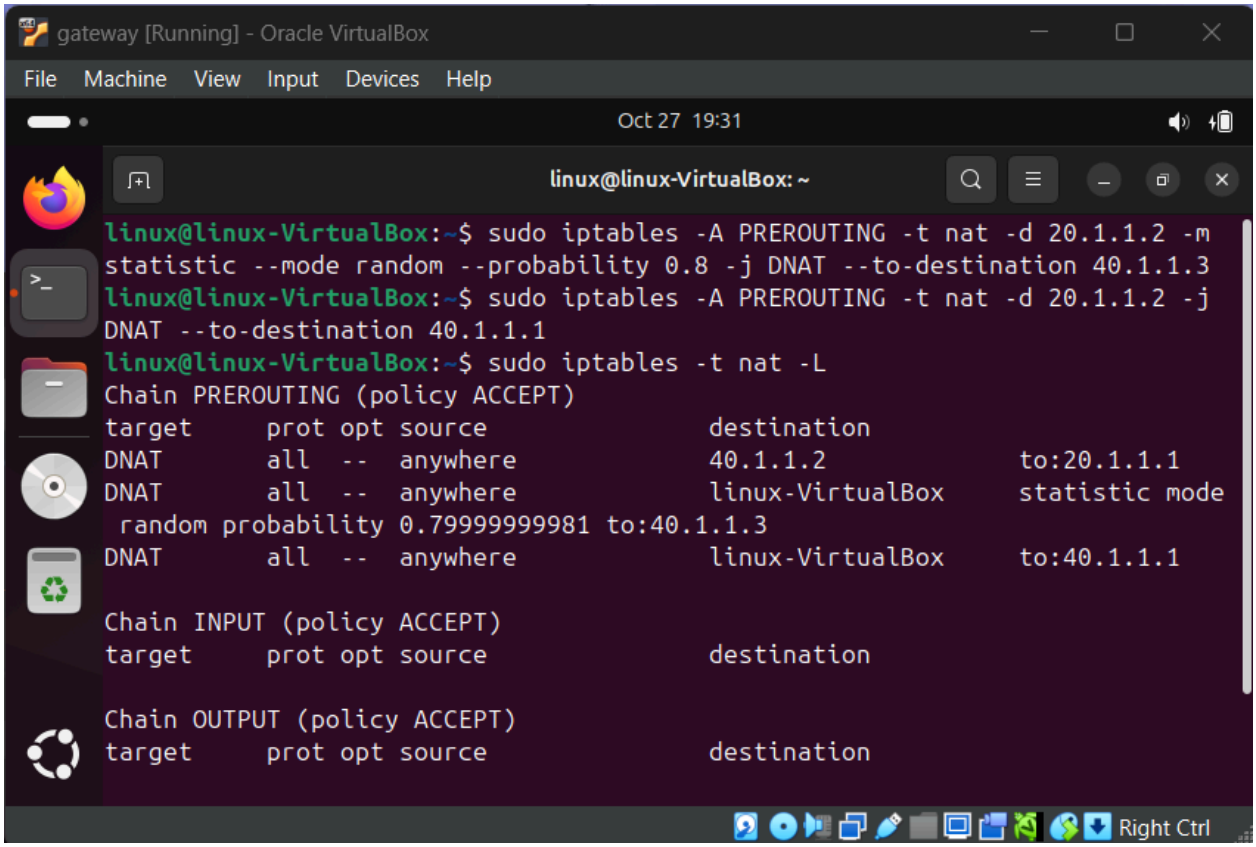
Q.5. Load balancing at the gateway VM

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.

server2 has a lower RTT value so it will be assigned 0.8 probability and server 1 will be assigned 0.2 probability.
This is done by executing the following commands on the gateway VM terminal.

sudo iptables -A PREROUTING -t nat -d 20.1.1.2 -m statistic --mode random --probability 0.8 -j DNAT -- to-destination 40.1.1.3
(this command means all packets destined to 20.1.1.2 will have their destination translated to 40.1.1.3 with a probability of 0.8)

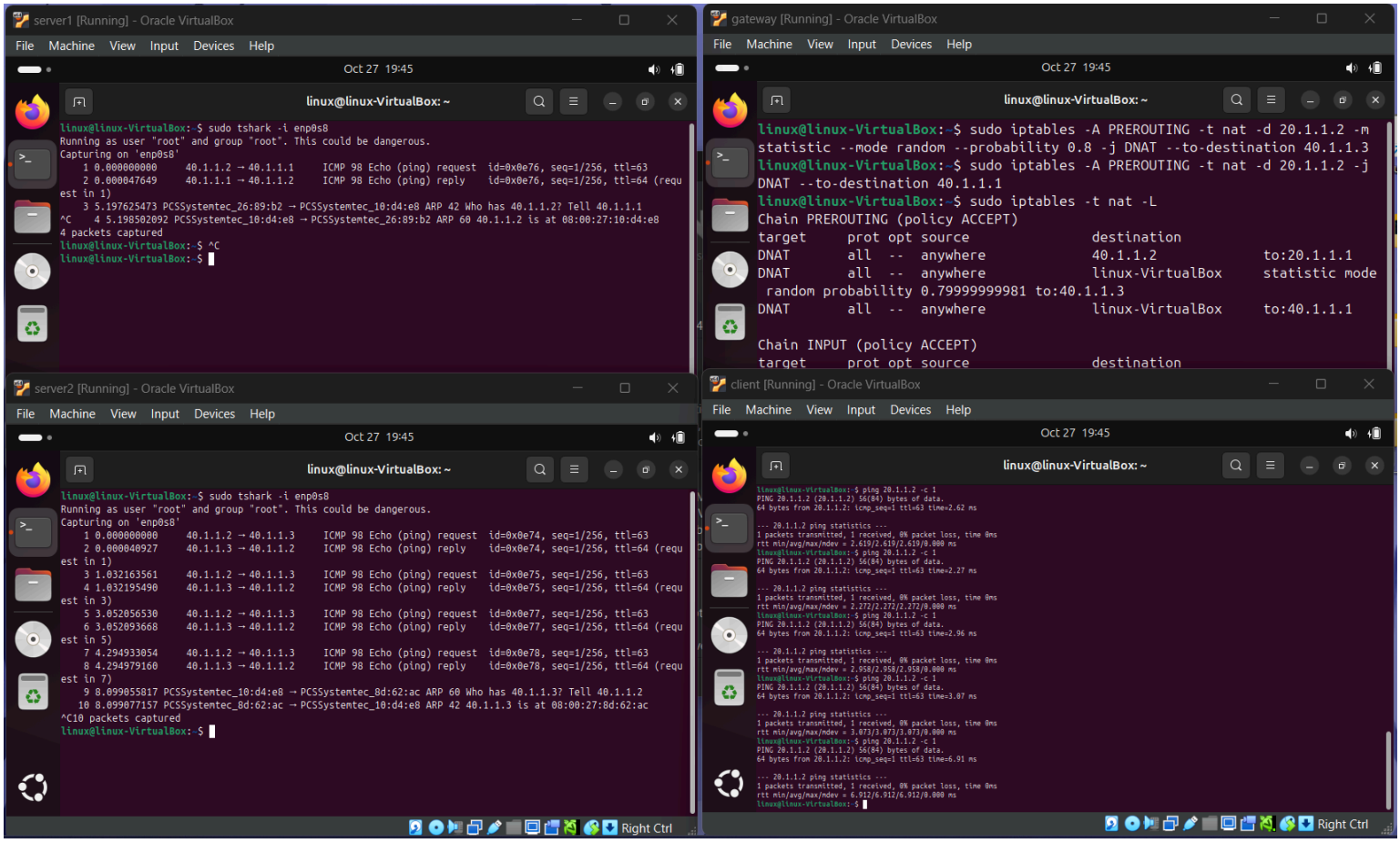
sudo iptables -A PREROUTING -t nat -d 20.1.1.2 -j DNAT -- to-destination 40.1.1.1
(this command means all packets destined to 20.1.1.2 will have their destination translated to 40.1.1.1 with a probability of 0.2 {1-0.8})



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

We send one ping at a time from the client to destination 20.1.1.2 which is the gateway port enp0s9 assigned to handle clients.
The screenshot below shows that out of 5 ping requests by the client, 4 are received by server 2 and 1 is received by server1.

P(server2) = 4/5 = 0.8
P(server1) = 1/5 = 0.2


server1 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Oct 27 19:45
linux@linux-VirtualBox: ~
linux@linux-VirtualBox:~$ sudo tshark -i enp0s8
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s8'
1 0.000000000 40.1.1.2 → 40.1.1.1 ICMP 98 Echo (ping) request id=0x0e76, seq=1/256, ttl=63
2 0.000047649 40.1.1.1 → 40.1.1.2 ICMP 98 Echo (ping) reply id=0x0e76, seq=1/256, ttl=64 (request in 1)
3 5.197625473 PCSSystemtec_26:89:b2 → PCSSystemtec_10:d4:e8 ARP 42 Who has 40.1.1.2? Tell 40.1.1.1
^C 4 5.198502092 PCSSystemtec_10:d4:e8 → PCSSystemtec_26:89:b2 ARP 60 40.1.1.2 is at 08:00:27:10:d4:e8
4 packets captured
linux@linux-VirtualBox:~$ ^C
linux@linux-VirtualBox:~$
gateway [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Oct 27 19:45
linux@linux-VirtualBox: ~
linux@linux-VirtualBox:~$ sudo iptables -A PREROUTING -t nat -d 20.1.1.2 -m statistic --mode random --probability 0.8 -j DNAT --to-destination 40.1.1.3
linux@linux-VirtualBox:~$ sudo iptables -A PREROUTING -t nat -d 20.1.1.2 -j DNAT --to-destination 40.1.1.1
linux@linux-VirtualBox:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
DNAT all -- anywhere 40.1.1.2 to:20.1.1.1
DNAT all -- anywhere linux-VirtualBox statistic mode random probability 0.79999999981 to:40.1.1.3
DNAT all -- anywhere linux-VirtualBox to:40.1.1.1
Chain INPUT (policy ACCEPT)
target prot opt source destination
server2 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Oct 27 19:45
linux@linux-VirtualBox: ~
linux@linux-VirtualBox:~$ sudo tshark -i enp0s8
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s8'
1 0.000000000 40.1.1.2 → 40.1.1.3 ICMP 98 Echo (ping) request id=0x0e74, seq=1/256, ttl=63
2 0.000040927 40.1.1.3 → 40.1.1.2 ICMP 98 Echo (ping) reply id=0x0e74, seq=1/256, ttl=64 (request in 1)
3 1.032163561 40.1.1.2 → 40.1.1.3 ICMP 98 Echo (ping) request id=0x0e75, seq=1/256, ttl=63
4 1.032195490 40.1.1.3 → 40.1.1.2 ICMP 98 Echo (ping) reply id=0x0e75, seq=1/256, ttl=64 (request in 3)
5 3.052056530 40.1.1.2 → 40.1.1.3 ICMP 98 Echo (ping) request id=0x0e77, seq=1/256, ttl=63
6 3.052093668 40.1.1.3 → 40.1.1.2 ICMP 98 Echo (ping) reply id=0x0e77, seq=1/256, ttl=64 (request in 5)
7 4.294933054 40.1.1.2 → 40.1.1.3 ICMP 98 Echo (ping) request id=0x0e78, seq=1/256, ttl=63
8 4.294979160 40.1.1.3 → 40.1.1.2 ICMP 98 Echo (ping) reply id=0x0e78, seq=1/256, ttl=64 (request in 7)
9 8.099055817 PCSSystemtec_10:d4:e8 → PCSSystemtec_8d:62:ac ARP 60 Who has 40.1.1.3? Tell 40.1.1.2
10 8.099077157 PCSSystemtec_8d:62:ac → PCSSystemtec_10:d4:e8 ARP 42 40.1.1.3 is at 08:00:27:8d:62:ac
^C10 packets captured
linux@linux-VirtualBox:~$
client [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Oct 27 19:45
linux@linux-VirtualBox: ~
Right Ctrl