CSE232: Programming Assignment 3

Submission by: Swapnil Panigrahi (2022522)

Contents

1 Virtual Machine Setup **2**
1.1 Configuration Commands . . . 3

2 Traffic Filtering at the Gateway VM **5**
2.0.1 Block All Traffic Except Ping to Server 1 . . . 6
2.0.2 Block Only TCP Traffic from Client . . . 7

3 Performance Testing with `iperf` **8**
3.1 TCP and UDP Bandwidth Test with `iperf` . . . 8

4 Question 4: NAT Configuration and Verification **12**
4.1 Commands for NAT Configuration on Gateway . . . 12
4.2 Traffic Capture Commands Using `tcpdump` . . . 12
4.3 Verification Using `iperf` . . . 13
4.4 Network Address Translation (NAT) Verification . . . 14

5 Network Address Translation and Testing at the Gateway VM **15**
5.1 Configuration Commands . . . 15
5.2 Inference from tcpdump . . . 17

Prerequisites

1. 4 virtual machines should set up and running, one should have 2 adapters
2. Adapter should be set to Host-Only, thus all packages necessary should be installed before hand

```
sudo apt upgrade
sudo apt install net-tools traceroute curl telnetd iperf
```

3. Telnet server should be running on 2 of the VMs

```
sudo nano /etc/inetd.conf
#Uncomment the line which begins with #<off>#
sudo systemctl restart inetd.service
nestast -tulnp
#Verify if a server is running on port 23
```

or HTTP server for curl

```
python3 -m http.server 80 &
```

1 Virtual Machine Setup

Q.1 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.

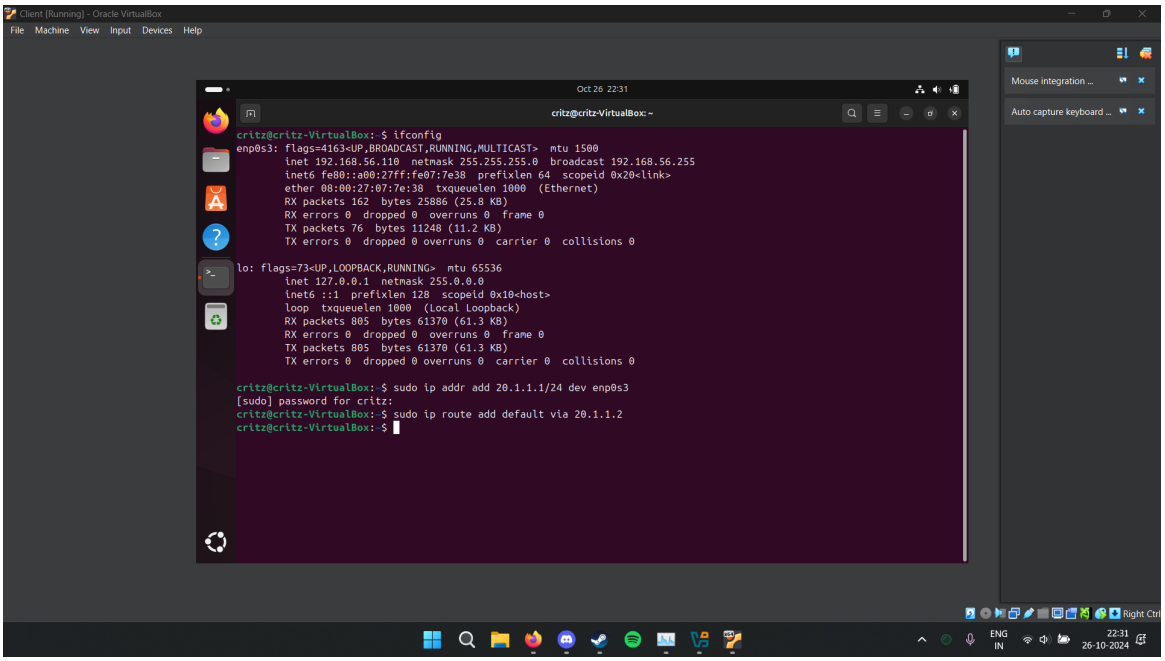

Figure 1: Client `ifconfig`

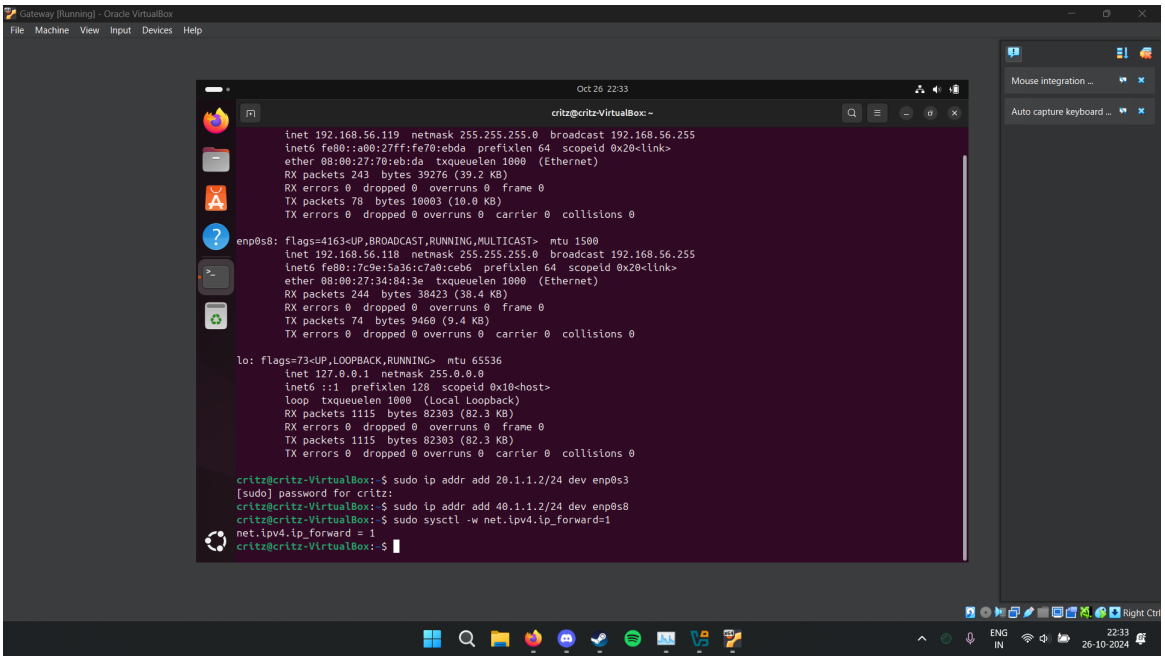

Figure 2: Gateway `ifconfig`

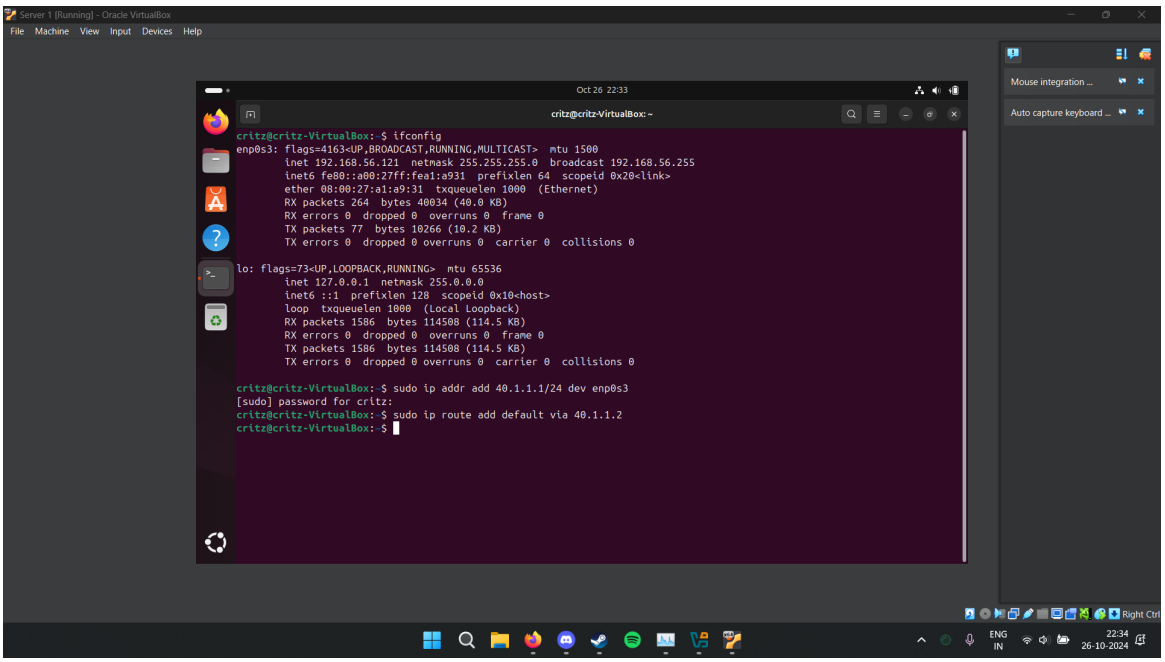

Figure 3: Server `ifconfig`

1.1 Configuration Commands

Commands for Client

```
1  sudo ip addr add 20.1.1.1/24 dev enp0s3
2  sudo ip link set enp0s3 up
3  sudo ip route add default via 20.1.1.2
```

Commands for Server 1

```
sudo ip addr add 40.1.1.1/24 dev enp0s3
sudo ip link set enp0s3 up
sudo ip route add default via 40.1.1.2
```

Commands for Server 2

```
sudo ip addr add 40.1.1.3/24 dev enp0s3
sudo ip link set enp0s3 up
sudo ip route add default via 40.1.1.2
```

Commands for Gateway

```
sudo ip addr add 20.1.1.2/24 dev enp0s3
sudo ip link set enp0s3 up
sudo ip addr add 40.1.1.2/24 dev enp0s8
sudo ip link set enp0s8 up
sudo sysctl -w net.ipv4.ip_forward=1
```

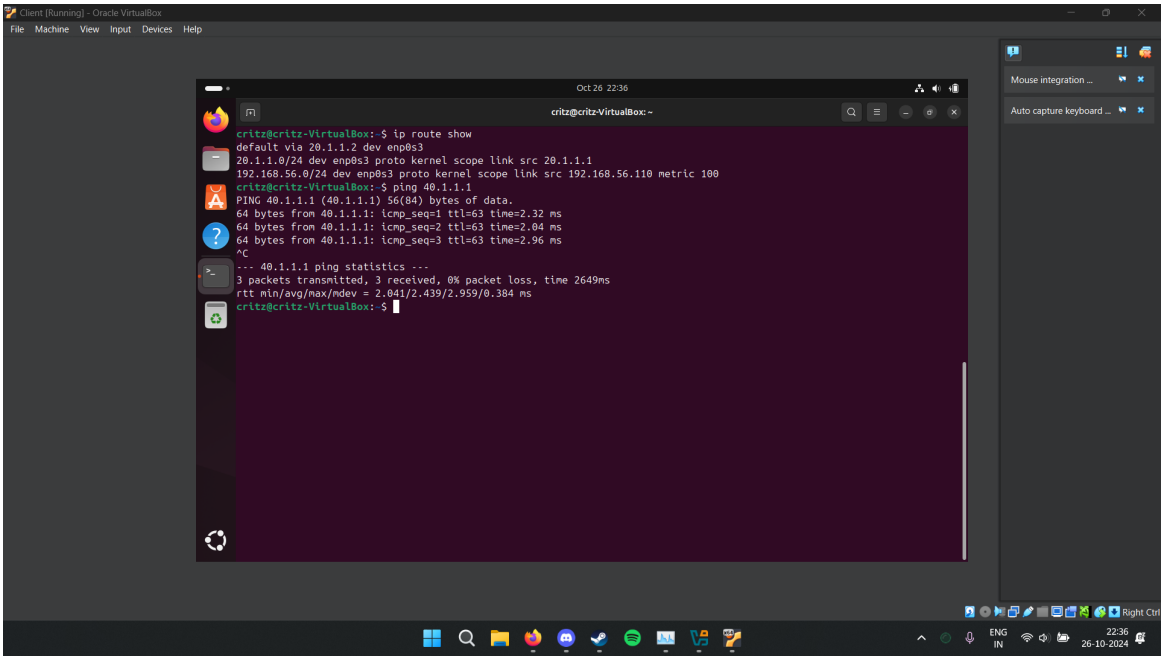

Figure 4: Default Gateway Setup

2 Traffic Filtering at the Gateway VM

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`. 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.

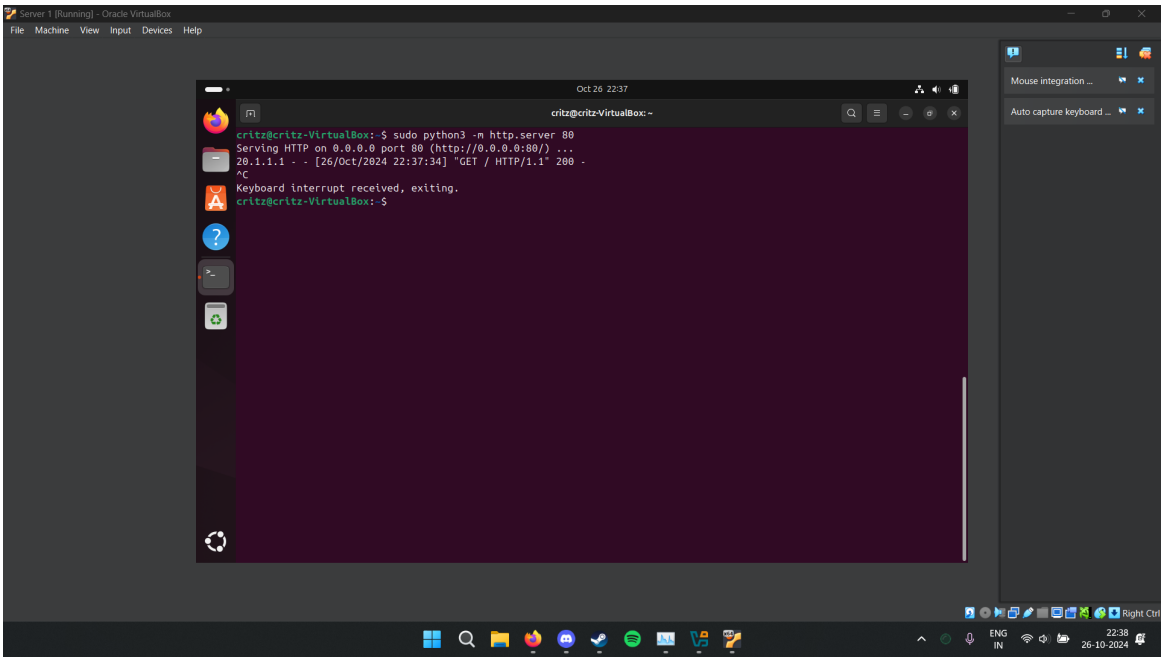

Figure 5: `HTTP` Server setup

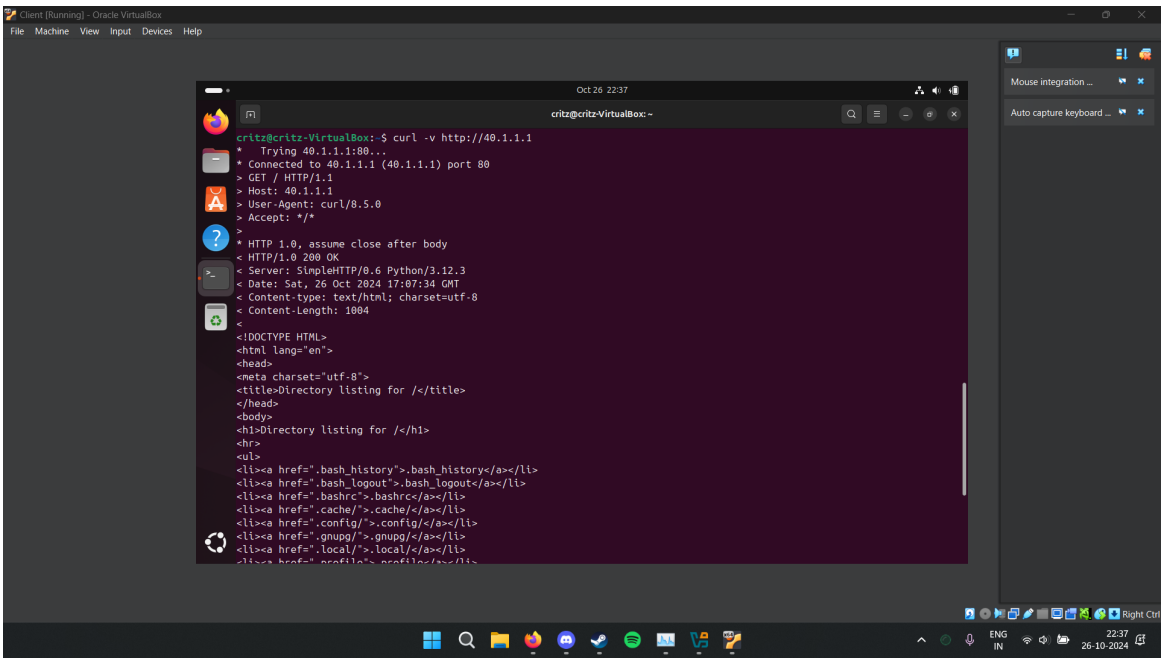

Figure 6: `curl` working before setting up `iptables`

2.0.1 Block All Traffic Except Ping to Server 1

Commands

```
sudo iptables -A FORWARD -p icmp -d 40.1.1.1 -j ACCEPT
sudo iptables -A FORWARD -d 40.1.1.1 -j DROP
sudo iptables -L -v -n
```

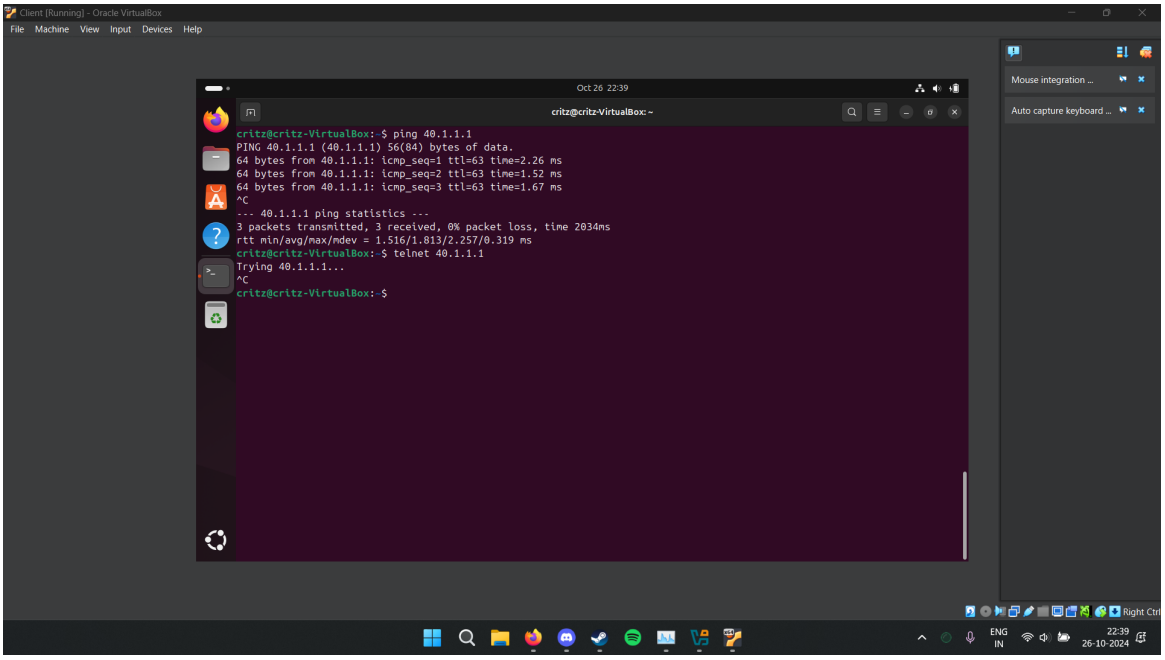

Figure 7: `ping` to Server 1 works while `telnet` doesn't

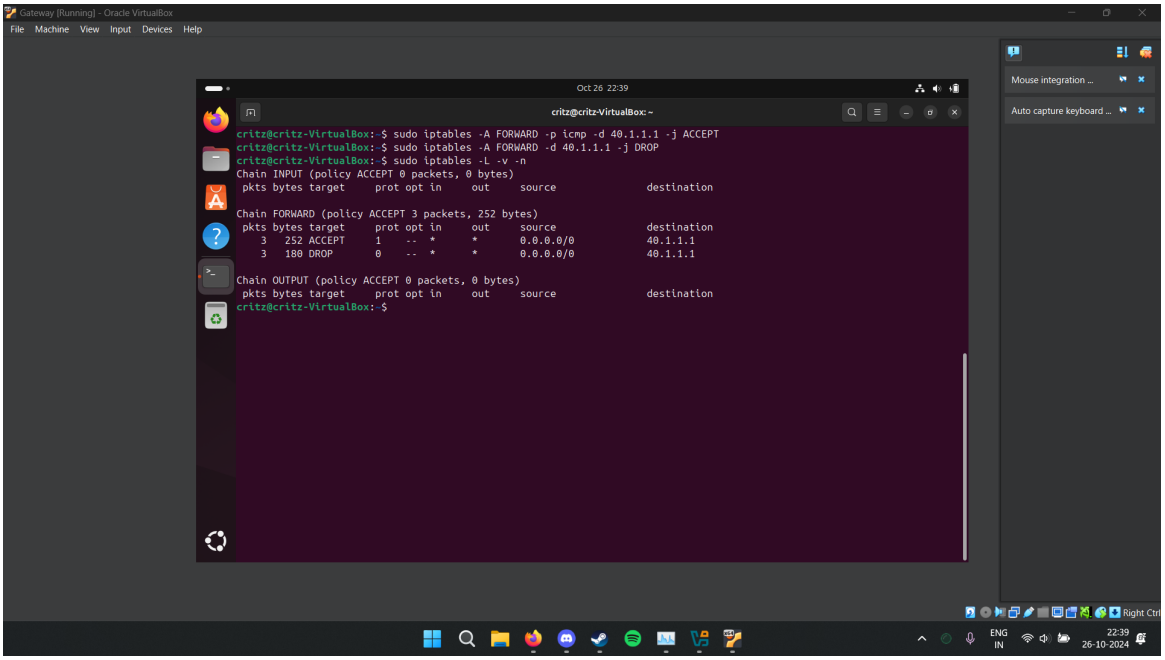

Figure 8: `iptables` on Gateway

2.0.2 Block Only TCP Traffic from Client

Commands

```
sudo iptables -F
sudo iptables -A FORWARD -p tcp -s 20.1.1.1 -j DROP
sudo iptables -A FORWARD -s 20.1.1.1 -j ACCEPT
sudo iptables -L -v -n
```

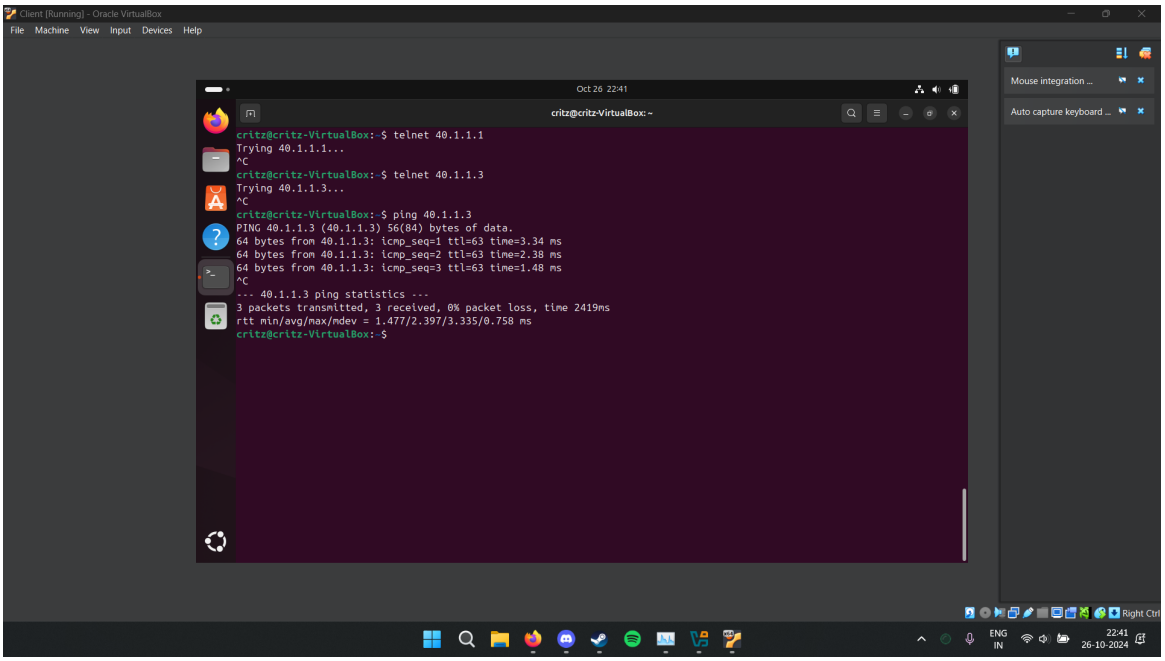

Figure 9: `telnet` doesn't work to either server

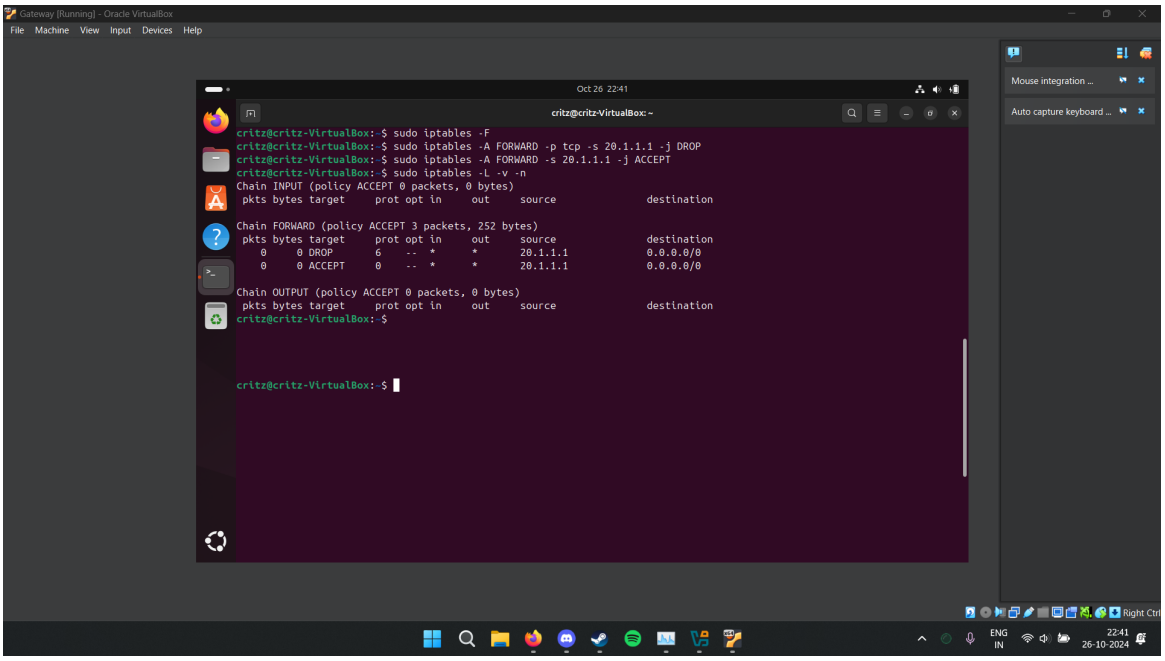

Figure 10: `iptables` on Gateway

3 Performance Testing with `iperf`

Q.3 Use the configuration obtained in Q.2 to solve this question.

(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.

(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?

3.1 TCP and UDP Bandwidth Test with `iperf`

Commands for TCP Bandwidth Testing with `iperf`

```
1  iperf -s #On Server 1
2  iperf -s #On Server 2
3
4  iperf -c 40.1.1.1
5  iperf -c 40.1.1.3
```

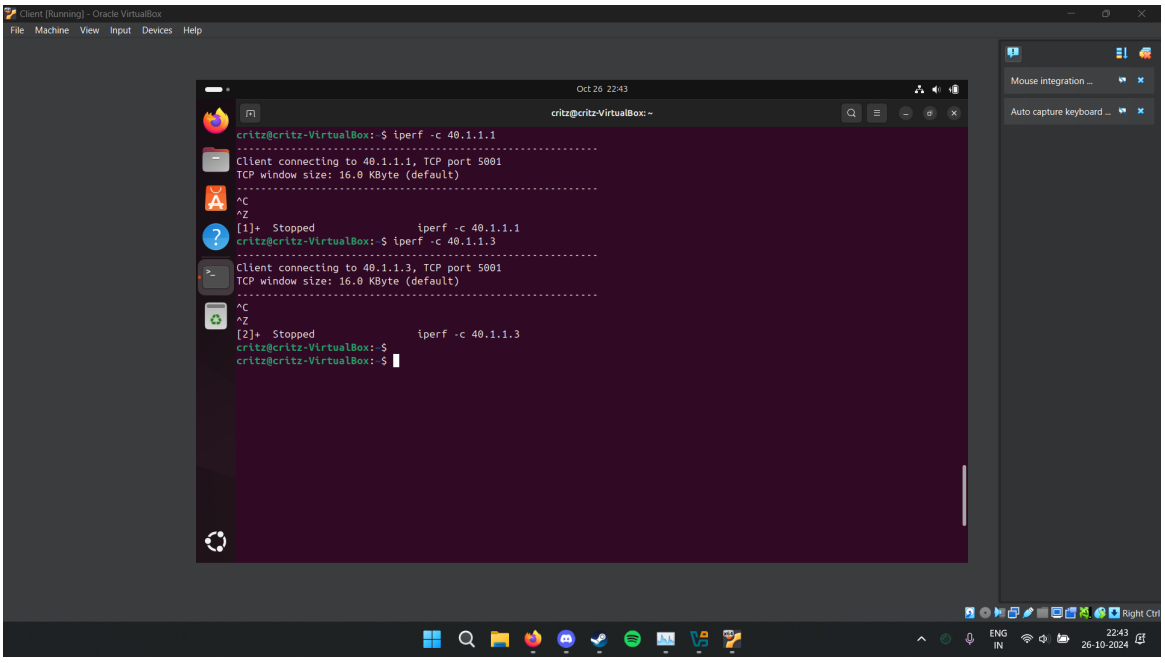

Figure 11: `iperf` on Client

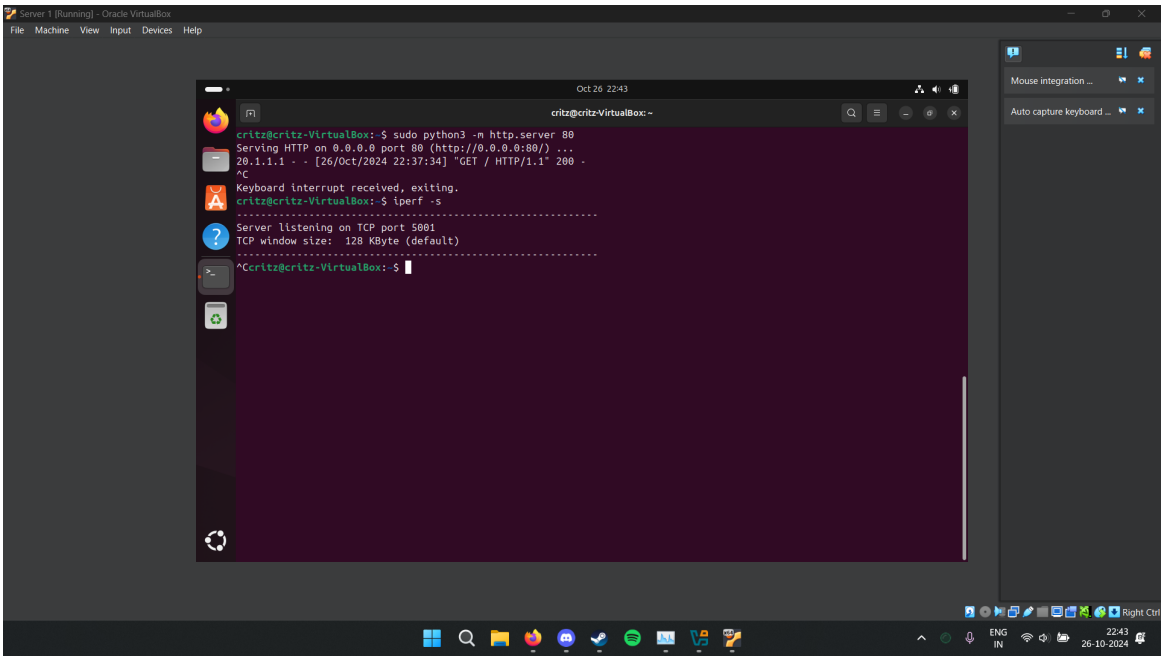

Figure 12: `iperf` on Server 1

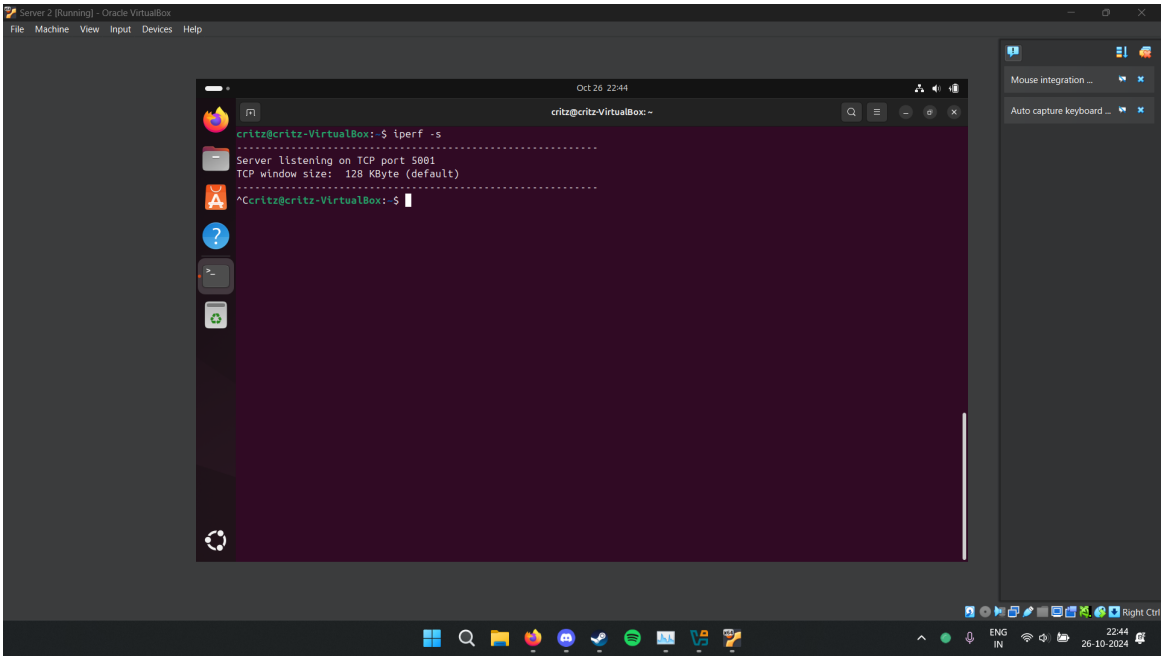

Figure 13: `iperf` on Server 2

Commands for UDP Bandwidth Testing with `iperf`

```
iperf -s -u
iperf -c 40.1.1.1 -u
```

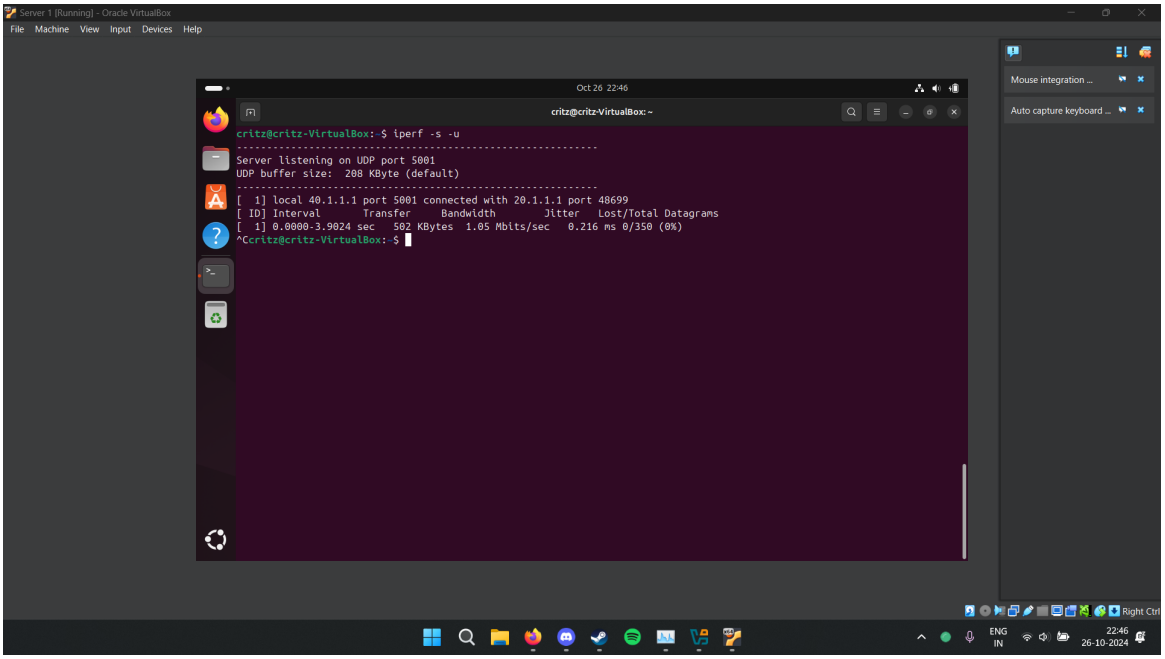

Figure 14: `iperf` on Client with UDP

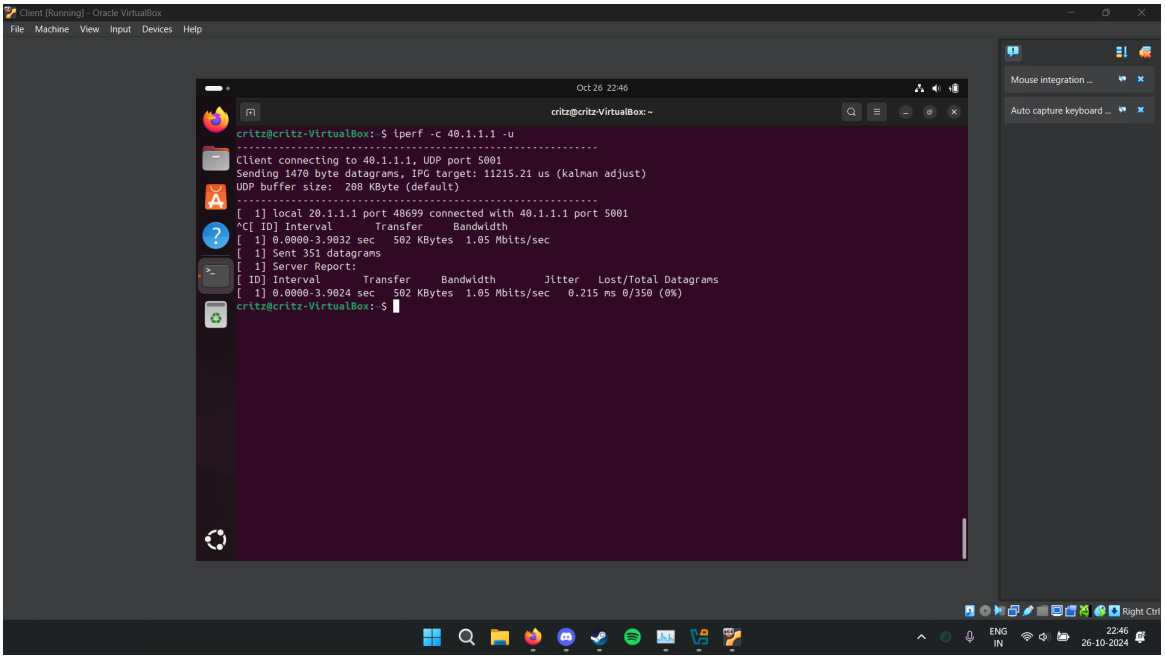

Figure 15: `iperf` on Server with UDP

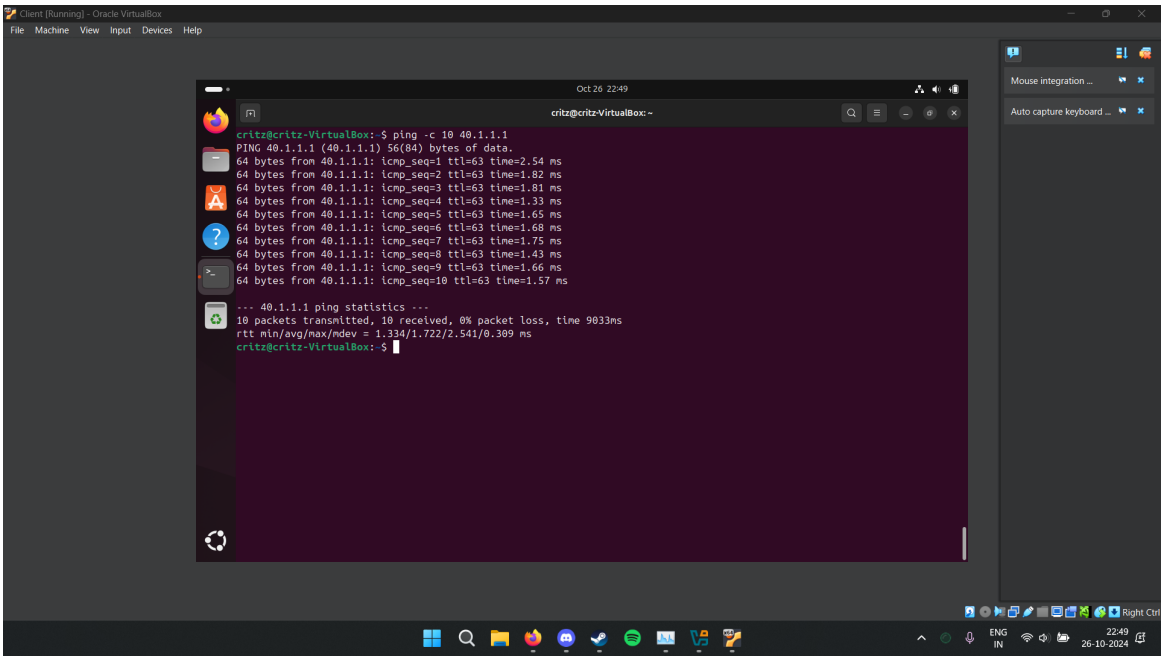

Figure 16: `ping` to Server 1

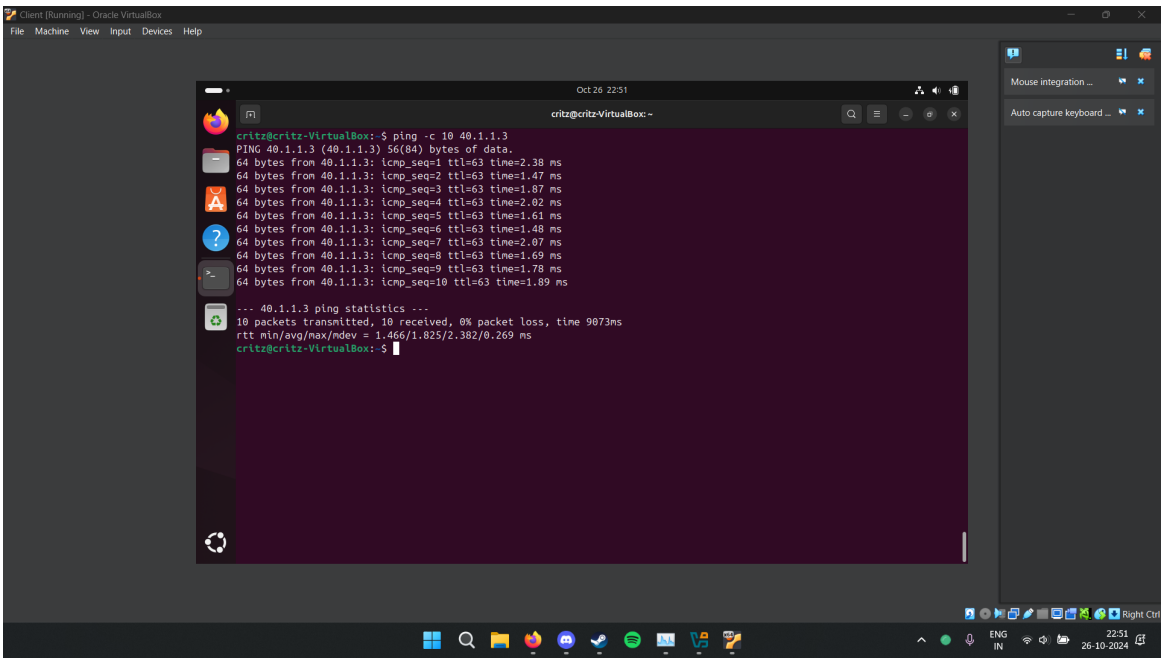

Figure 17: `ping` to Server 2

Commands for RTT Testing

```
ping -c 50 40.1.1.1
ping -c 50 40.1.1.3
```

Note: The slight difference in RTT between `40.1.1.1` and `40.1.1.3` could be due to network load balancing configurations and iptables processing time.

4 Question 4: NAT Configuration and Verification

Q.4 Configure NAT rules on the gateway for traffic manipulation and verify using `tcpdump` and `iperf`.

(a) Set up a `POSTROUTING` rule on the gateway VM to modify the source IP of outgoing packets from the client.

(b) Configure a `PREROUTING` rule to forward incoming packets to the client.

4.1 Commands for NAT Configuration on Gateway

```
sudo iptables -t nat -A POSTROUTING -s 20.1.1.1 -j SNAT --to-source 40.1.1.2
sudo iptables -t nat -A PREROUTING -d 40.1.1.2 -j DNAT --to-destination 20.1.1.1
```

4.2 Traffic Capture Commands Using `tcpdump`

To verify the NAT configuration, capture packets on the respective interfaces on the gateway, server, and client using the following commands.

```
# On the Gateway to capture traffic destined for 20.1.1.1
sudo tcpdump -i enp0s3 'dst 20.1.1.1'

# On the Server to capture packets to/from 40.1.1.2
sudo tcpdump -i enp0s3 'host 40.1.1.2'

# On the Client to capture packets to/from 40.1.1.1
sudo tcpdump -i enp0s3 'host 40.1.1.1'
```

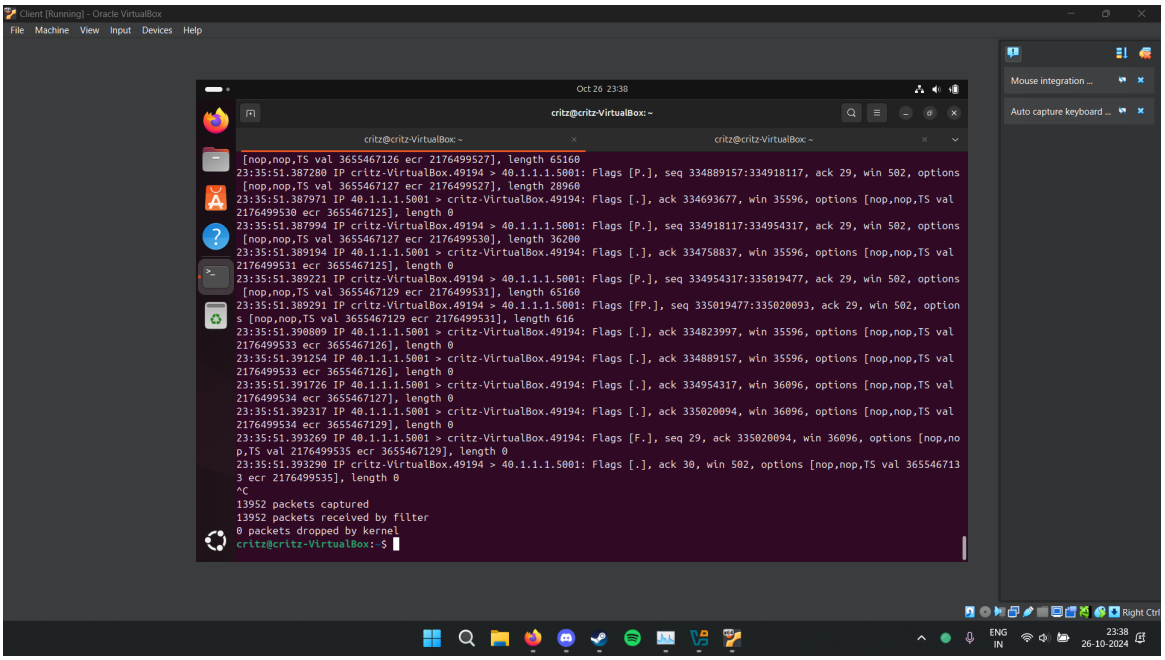

Figure 18: `tcpdump` on Client

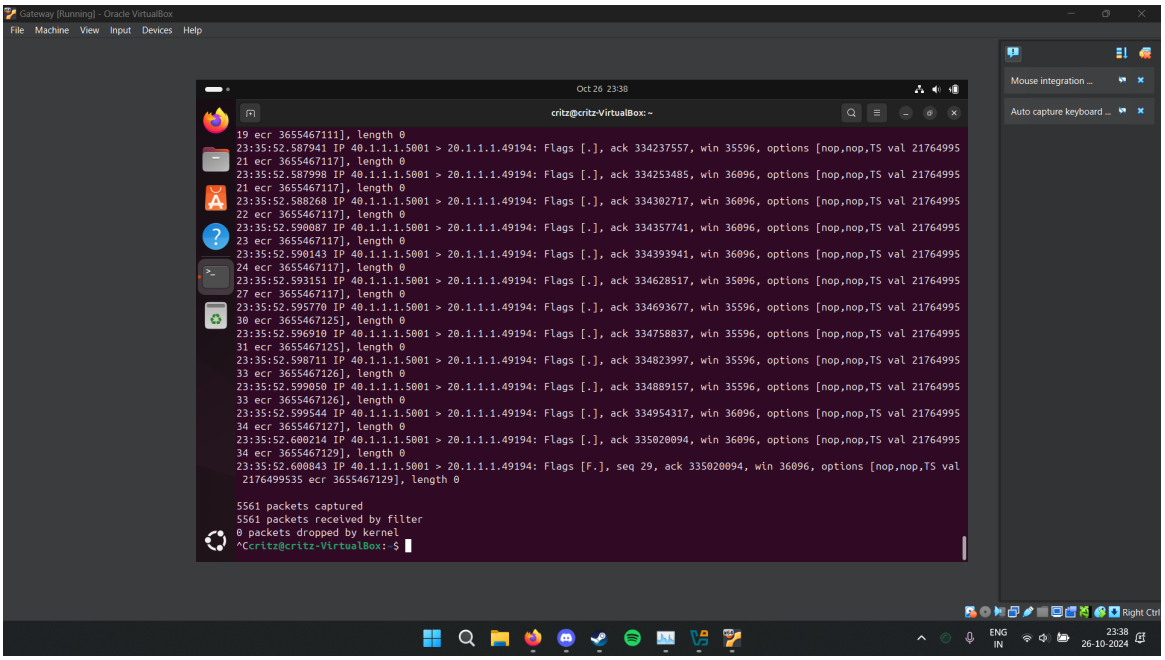

Figure 19: `tcpdump` on Gateway

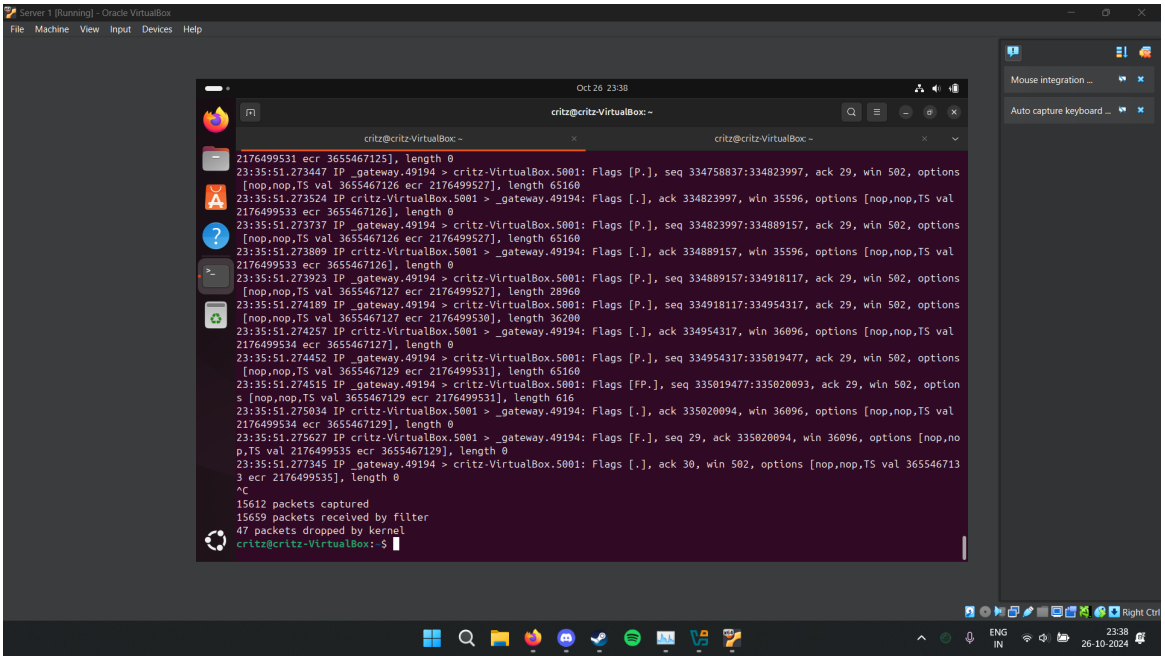

Figure 20: `tcpdump` on Server

4.3 Verification Using `iperf`

To verify the network behavior, use `iperf` to generate traffic between the client and server:

```
1  iperf -s #On Server
2  iperf -c 40.1.1.1 #On Client
```

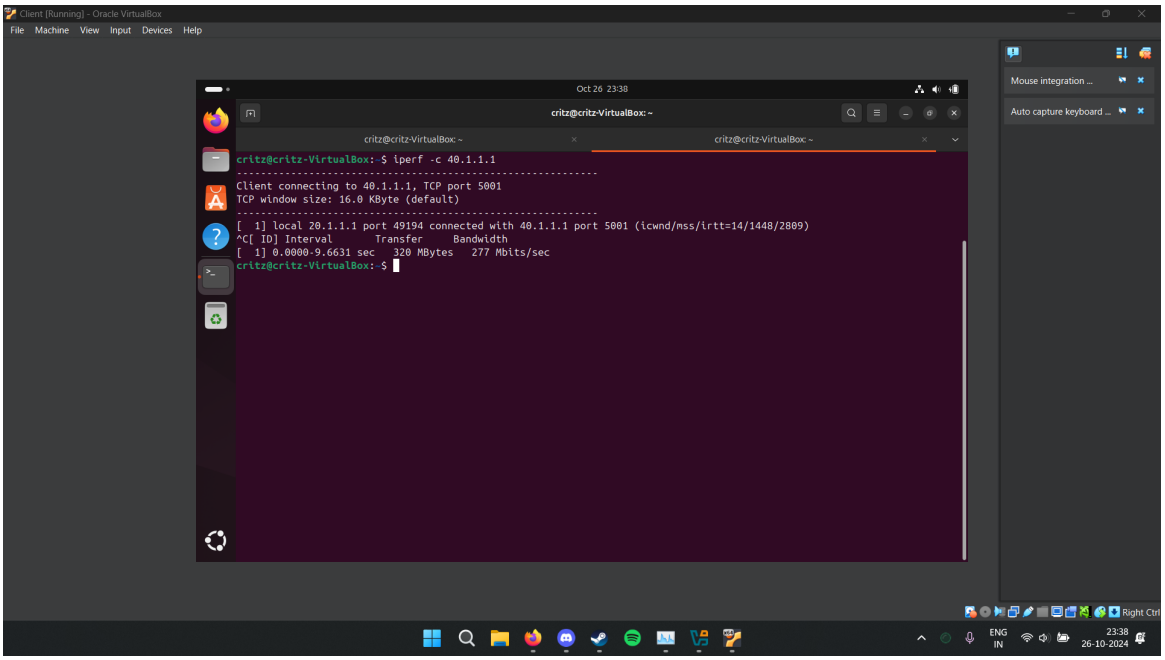

Figure 21: `iperf` on Client for verification

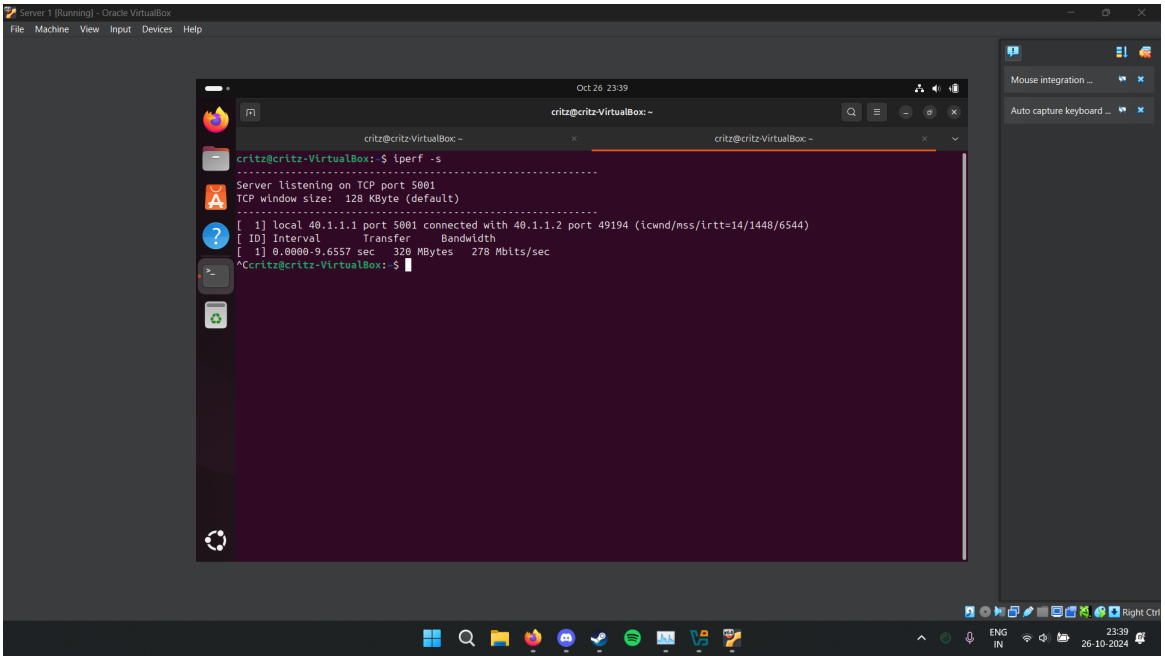

Figure 22: `iperf` on Server for verification

4.4 Network Address Translation (NAT) Verification

To demonstrate how the Network Address Translation (NAT) is functioning in the question, the tcpdump outputs from each VM are analyzed with specific examples as follows:

1. **Source IP Translation (20.1.1.1 → 40.1.1.2):**
   - In the tcpdump output on the *Gateway VM*, packets originating from `20.1.1.1` are translated to `40.1.1.2` when forwarded to the servers (`40.1.1.1` or `40.1.1.3`). This demonstrates that the SNAT rule is correctly changing the source IP, making the packet appear as if it originated from `40.1.1.2`.

2. **Destination IP Reversion (40.1.1.1 → 20.1.1.1):**

   - In the tcpdump output on the *Client VM*, responses arriving from `40.1.1.1` have their destination IP reverted to `20.1.1.1`, as per the DNAT rule. This confirms that the gateway is translating the destination back to the original client IP, allowing the client to receive the packet as if it were sent directly to it.

5 Network Address Translation and Testing at the Gateway VM

Q.5 Validate NAT configuration at the Gateway VM by observing the packet flow using tcpdump and testing with ICMP requests.

(a) Configure iptables on the Gateway VM to forward ICMP packets with randomized routing between servers.

(b) Capture and analyze ICMP traffic to ensure proper translation and response.

5.1 Configuration Commands

Commands for Gateway

```
sudo iptables -t nat -F PREROUTING

sudo iptables -t nat -A PREROUTING -s 20.1.1.1 -d 40.1.1.2 -p icmp -m statistic --mode random
 ↪  --probability 0.8 -j DNAT --to-destination 40.1.1.1
sudo iptables -t nat -A PREROUTING -s 20.1.1.1 -d 40.1.1.2 -p icmp -j DNAT --to-destination
 ↪  40.1.1.3

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

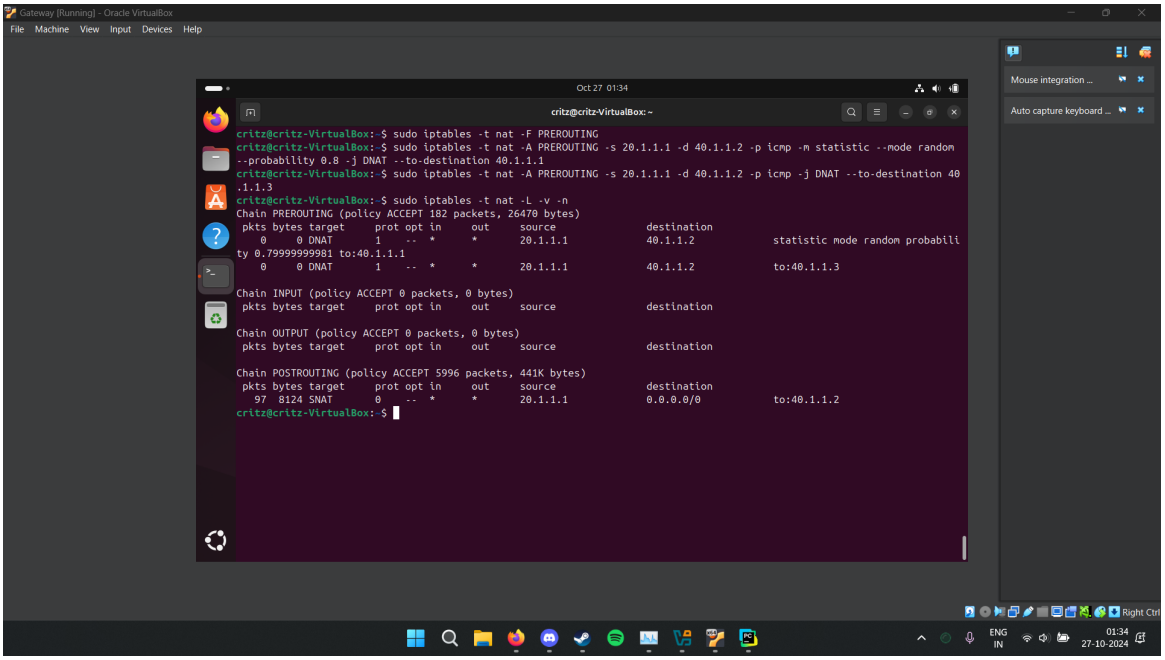

Figure 23: NAT Table on Gateway

Commands on Server for ICMP Capture

```
sudo tcpdump -i enp0s3 icmp -w ~/icmp_capture_server1.pcap
sudo tcpdump -i enp0s3 icmp -w ~/icmp_capture_server2.pcap
```

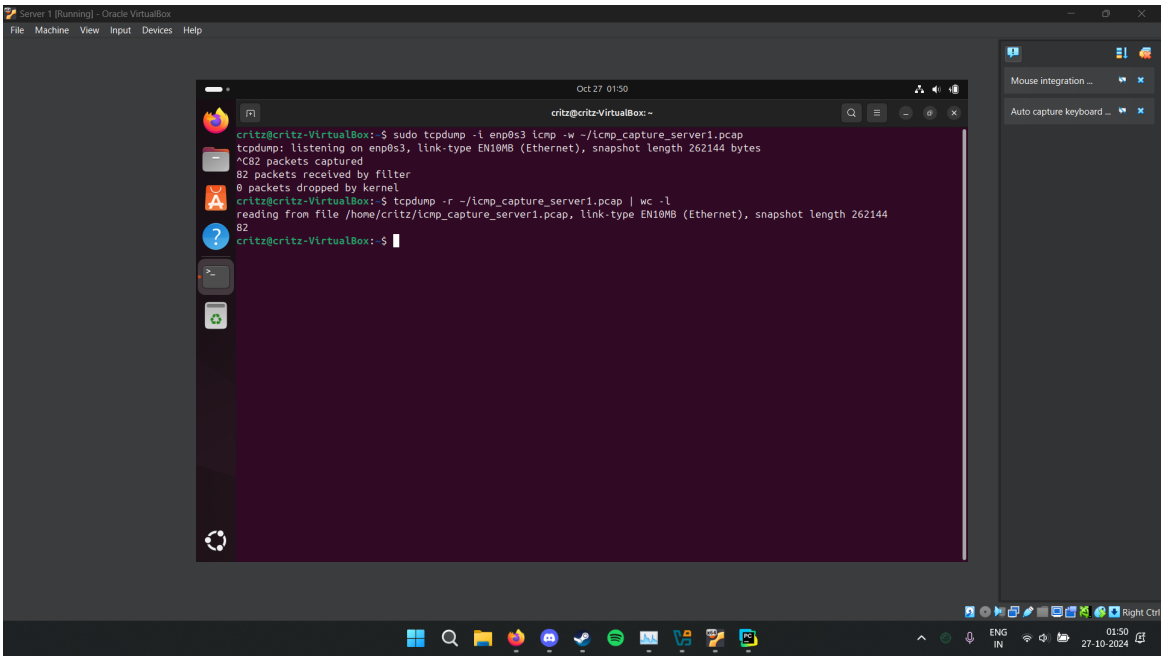

Figure 24: `tcpdump` on Server 1

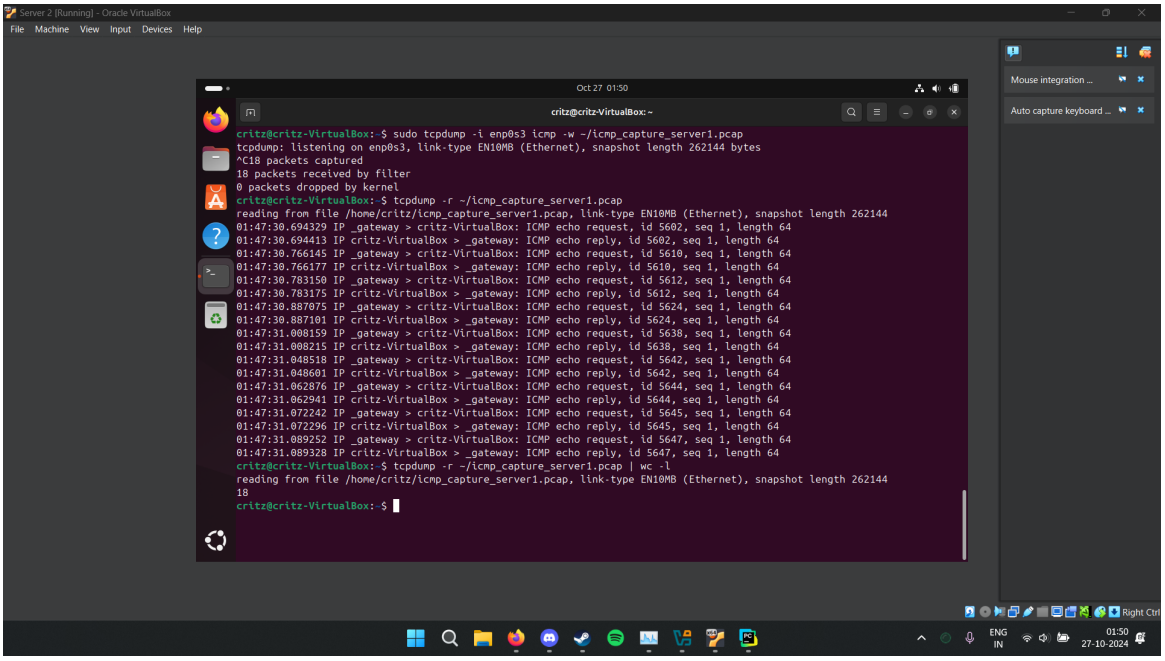

Figure 25: `tcpdump` on Server 2

Commands on Client to Test Connectivity

```
for i in {1..50}; do ping -c 1 40.1.1.2; done
```

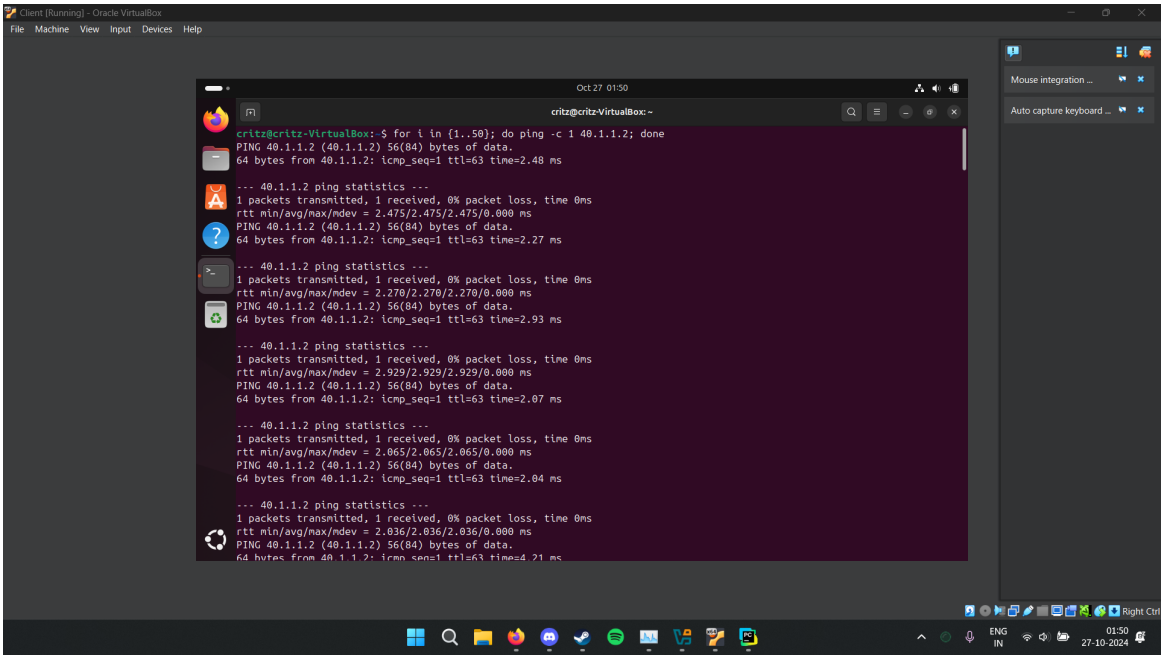

Figure 26: Enter Caption

5.2 Inference from tcpdump

The tcpdump captures provide insight into the behavior of the NAT setup:

- From the screenshot (Fig. 25 on Server 1), we observe ICMP echo requests and replies indicating that packets are successfully reaching Server 1 and being returned.
- Similarly, in Fig. 26 (Server 2), ICMP requests are observed, verifying that packets are also routed to Server 2 as per the random distribution set by iptables.
- The packet count in each capture file (`icmp_capture_server1.pcap` and `icmp_capture_server2.pcap`) shows roughly 80% of packets directed to Server 1 and 20% to Server 2, aligning with the probability set in the iptables rule.