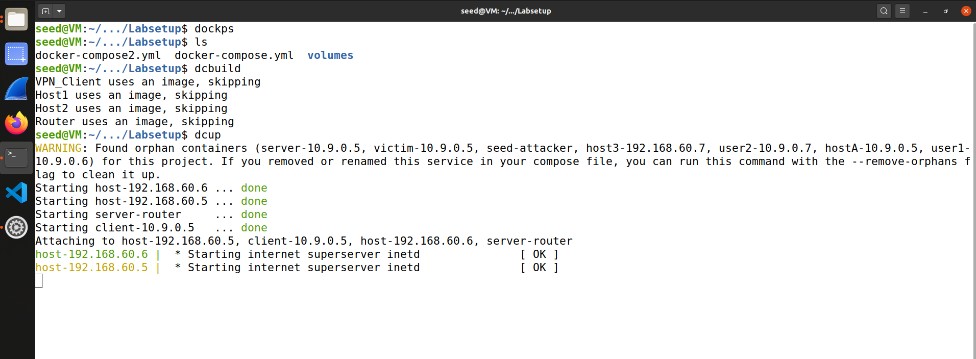
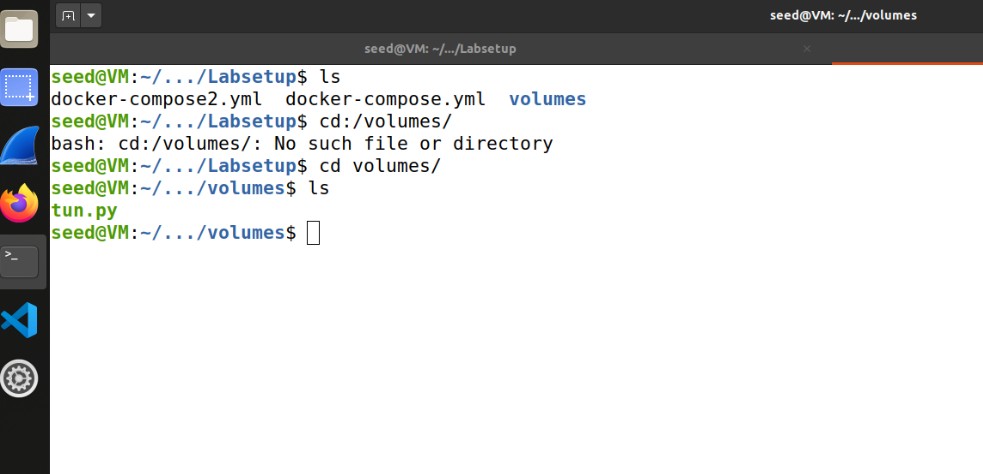
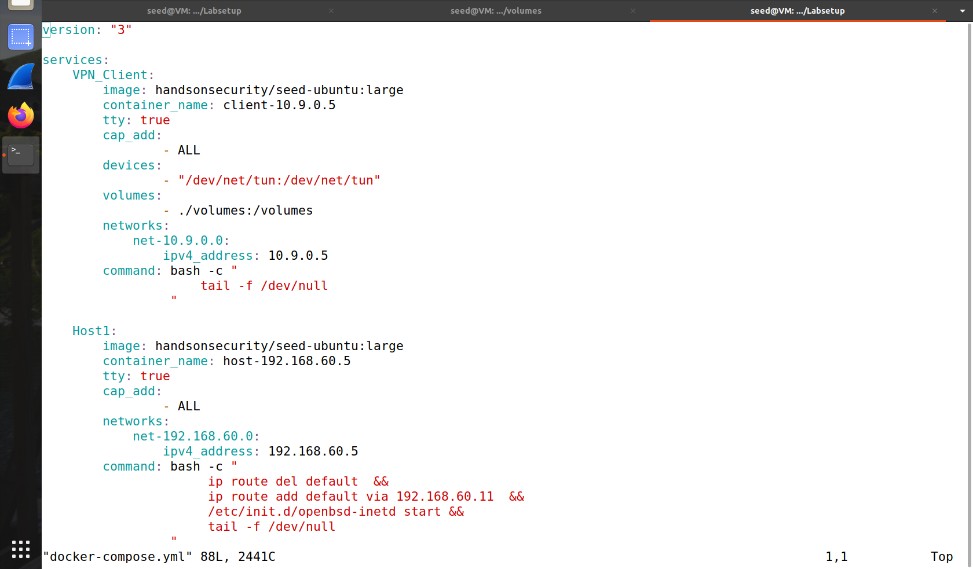
ACS 545 Cryptography and Network Security Lab 6: VPN Lab: The Container Version

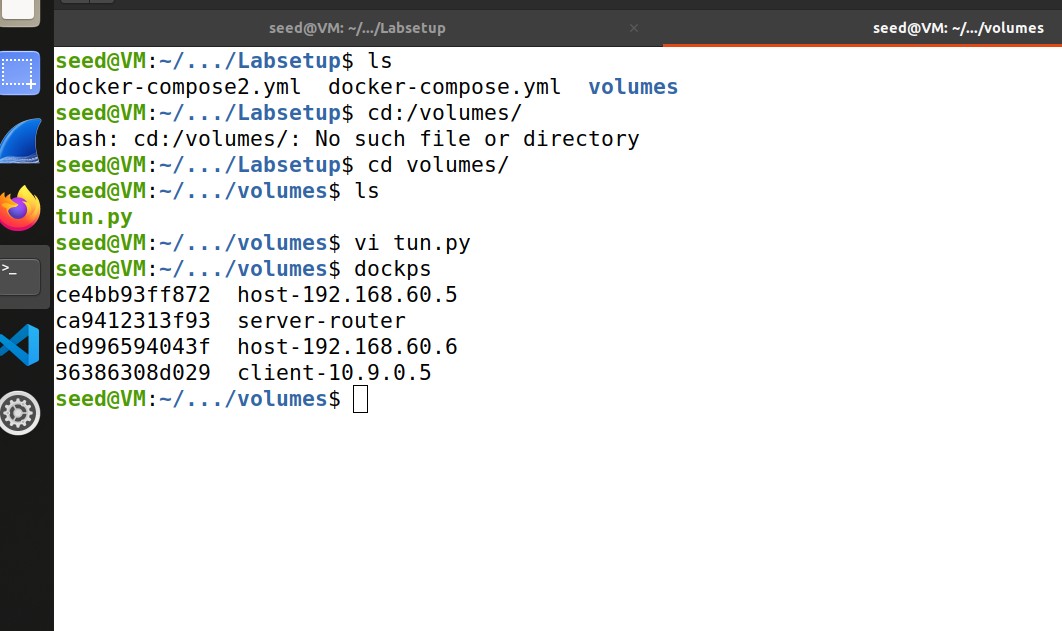
1. Task 1: Network Setup

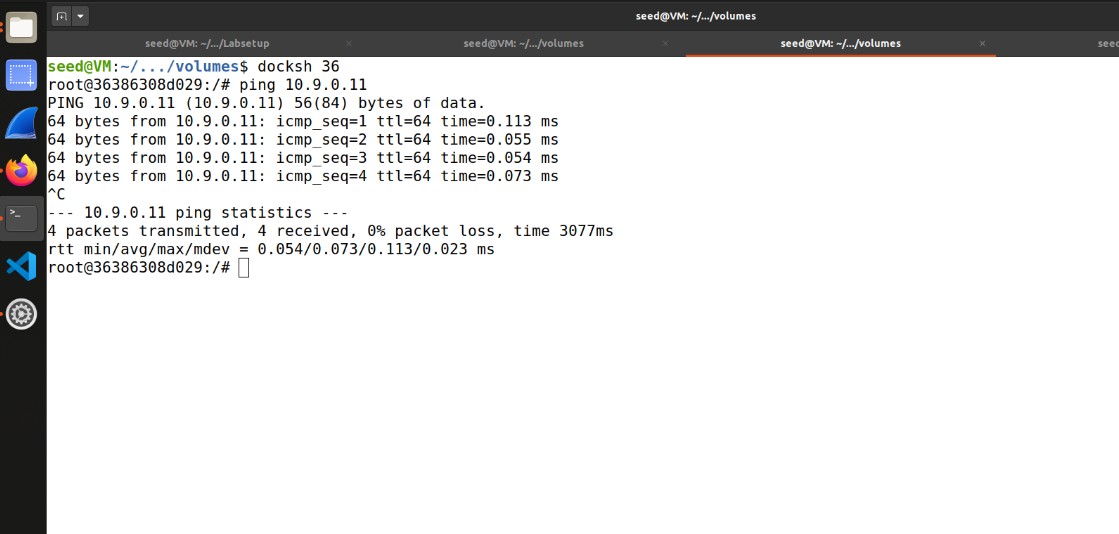
Lab setup: Building the docker- compose file.

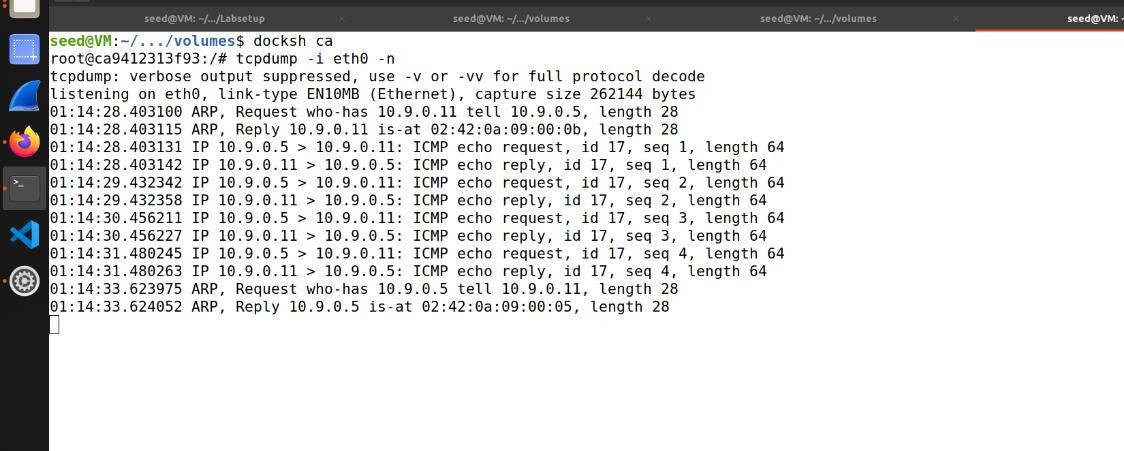
Shared folder: Docker file consists of entry ‘volumes:’ for the VM and contain to

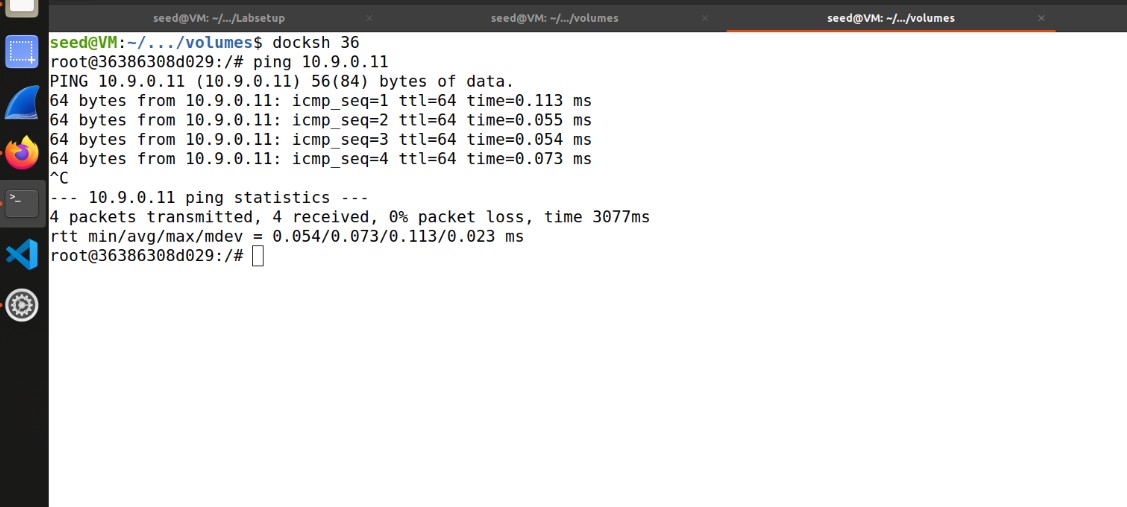
share files.

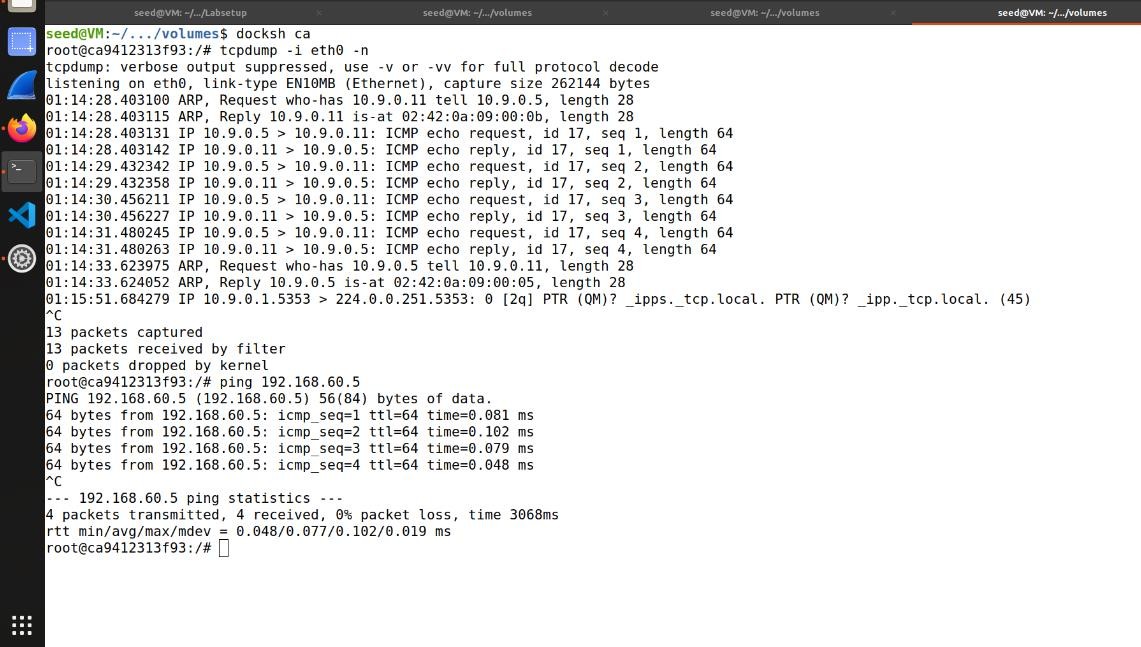


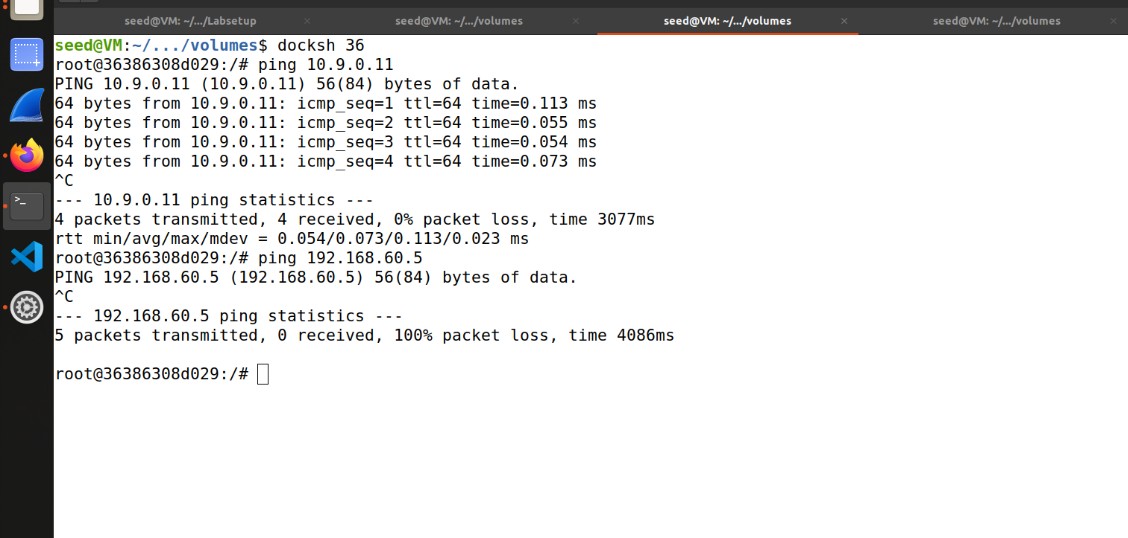
Packet sniffing: Running tcpdump on containers and pinging from client to check packets.

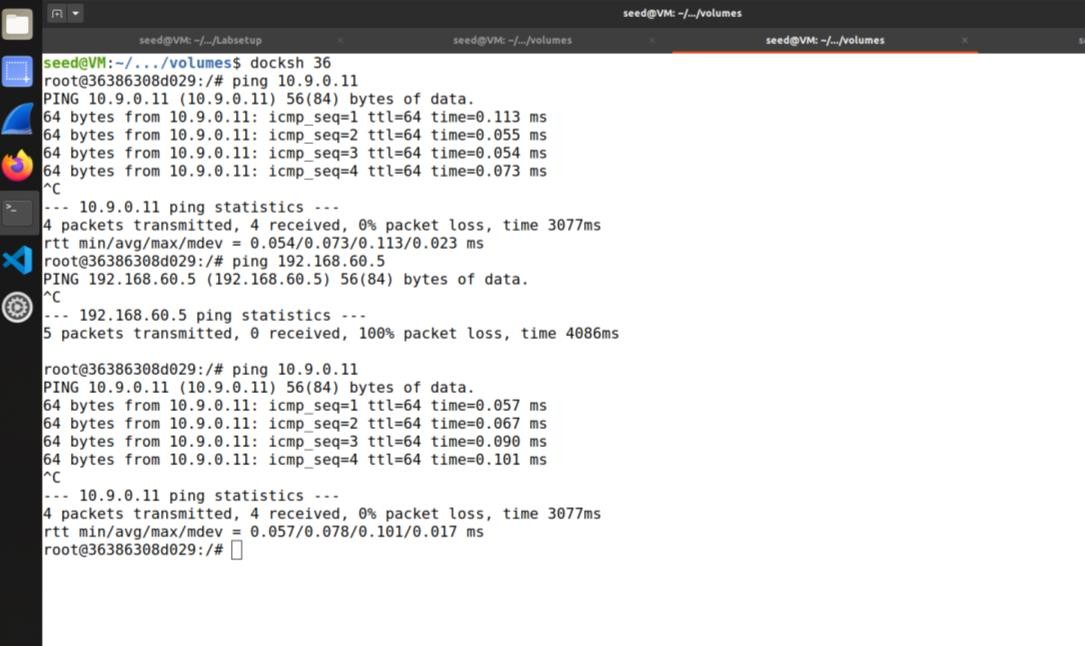


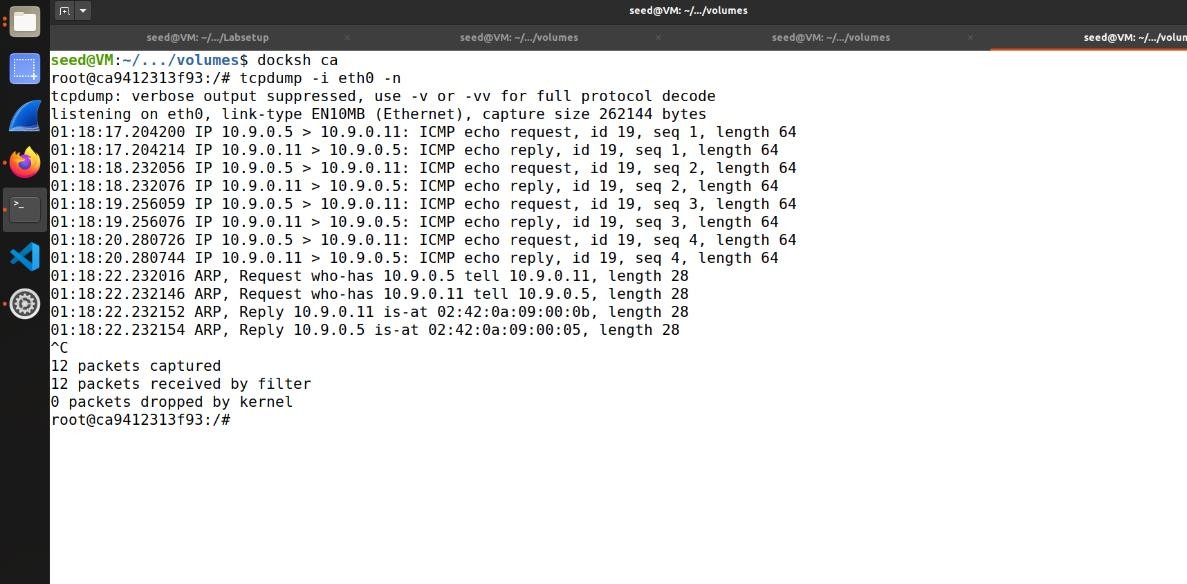


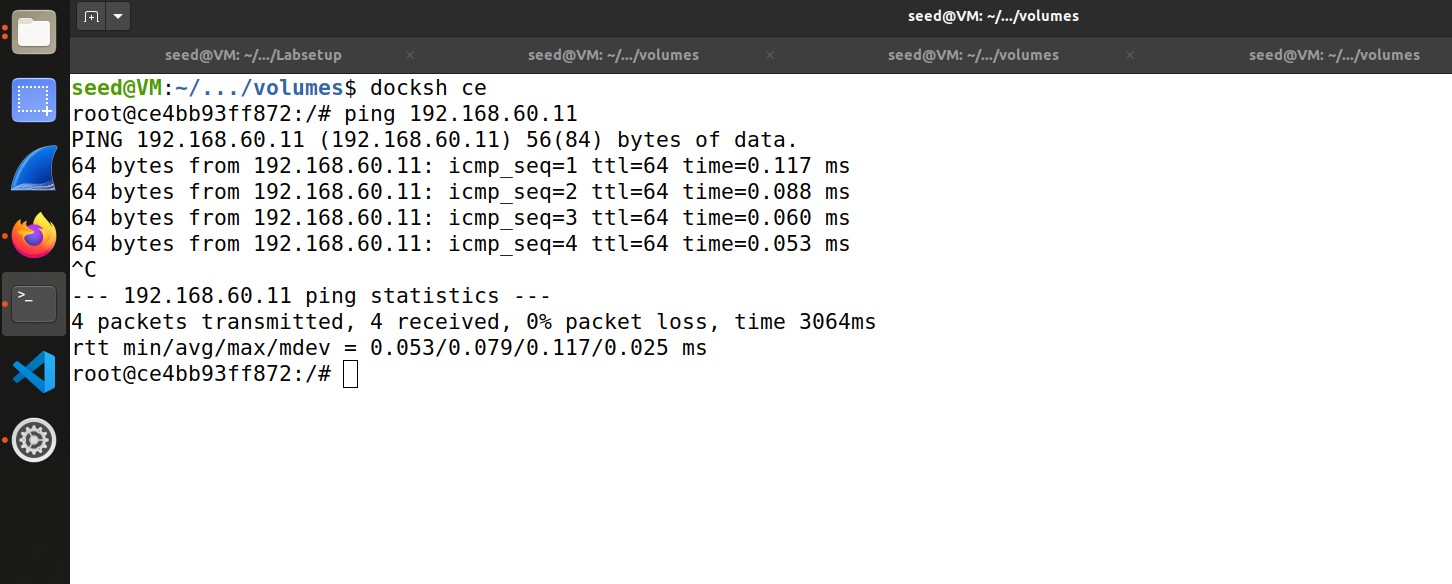
Testing: Host U can communicate with VPN Server.

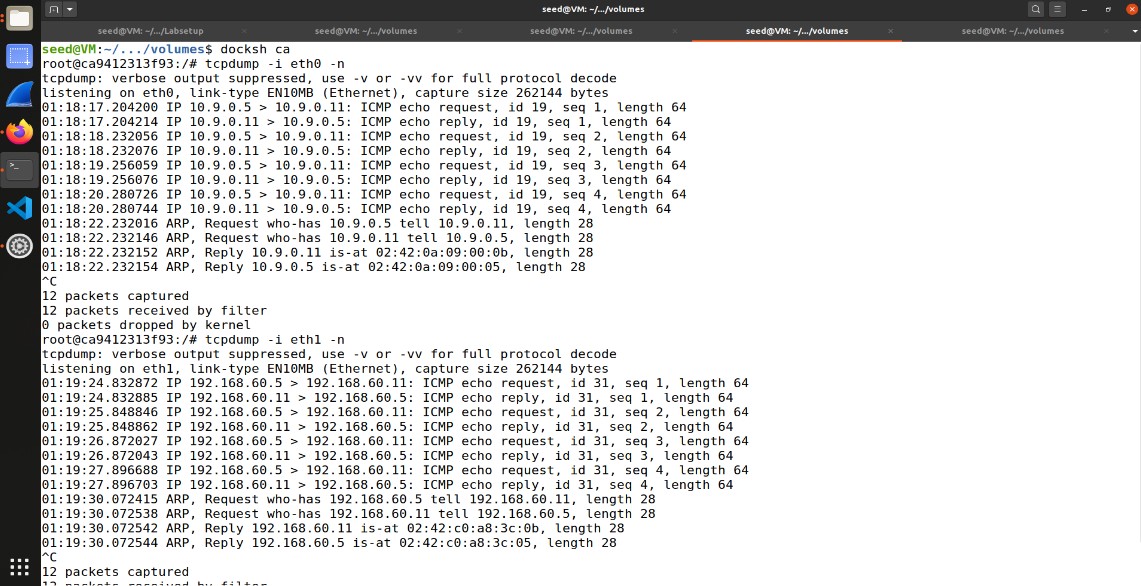
VPN Server can communicate with Host V.

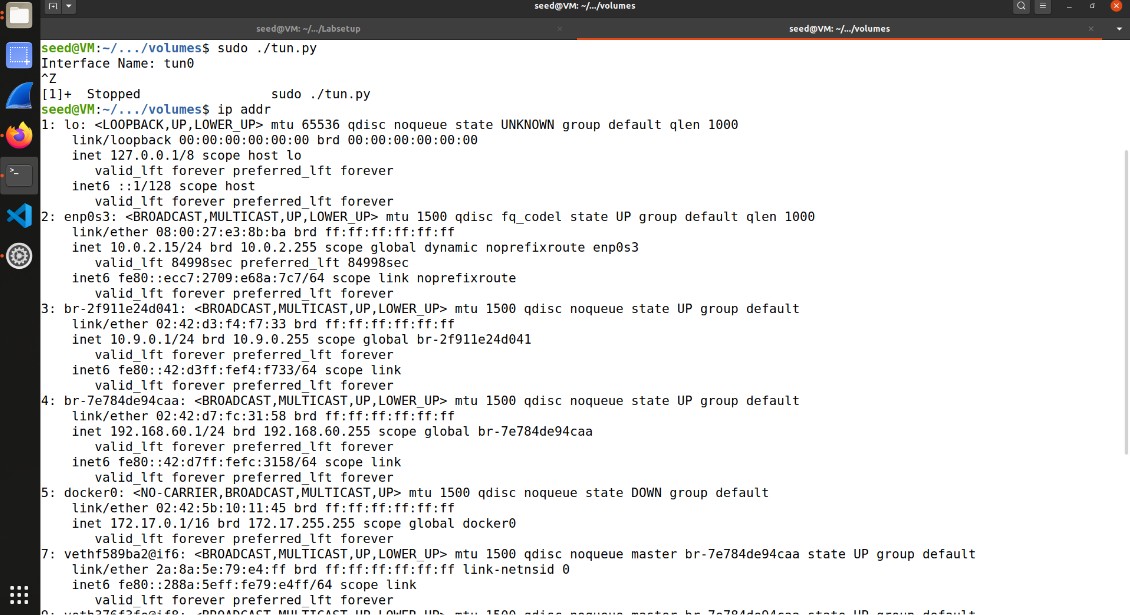
Host U should not be able to communicate with Host V.

Run tcpdump on the router, and sniff the traffic on each of the network.





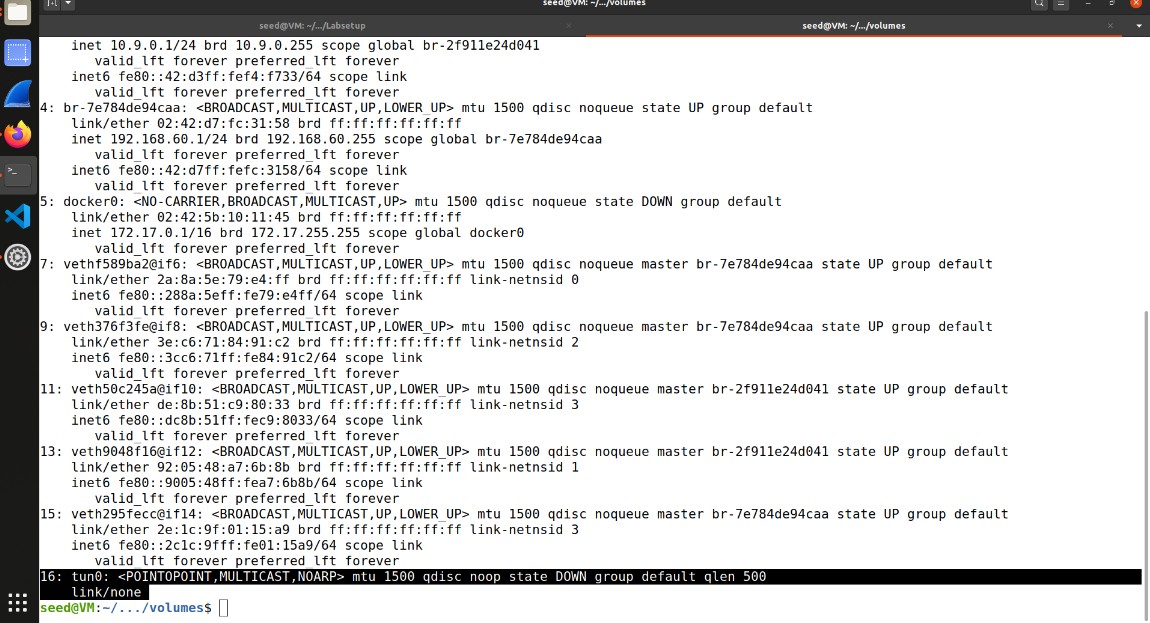


1. Task 2: Create and Configure TUN Interface

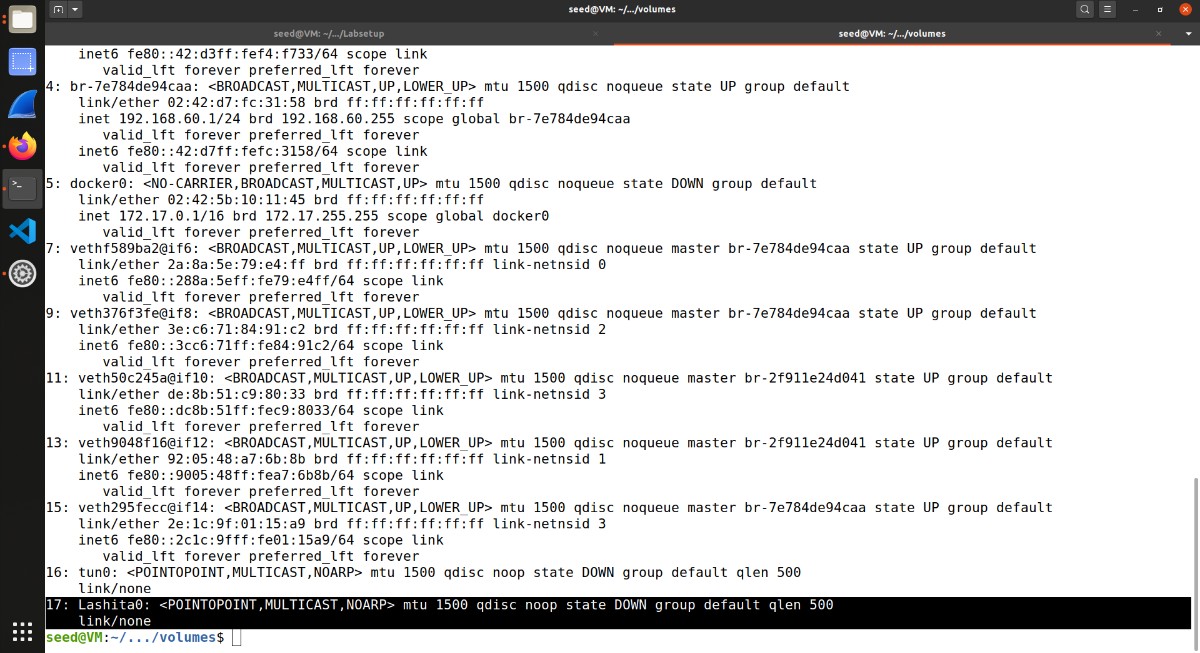
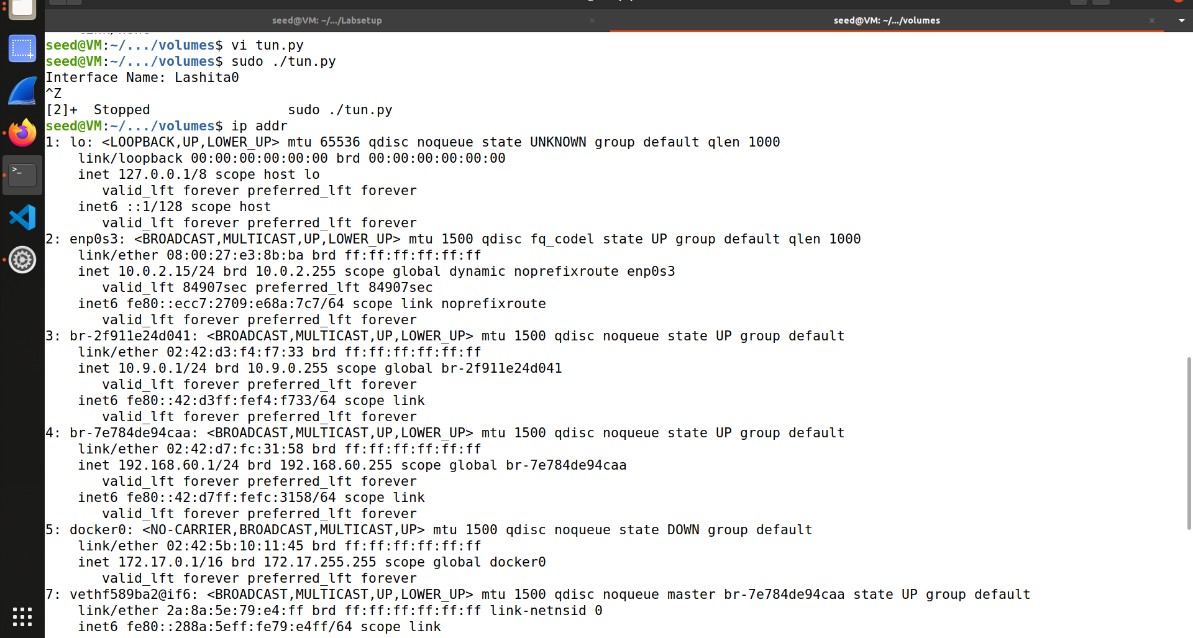
Task 2.a:

Name of the Interface Running the tun.py using command sudo

./tun.py.

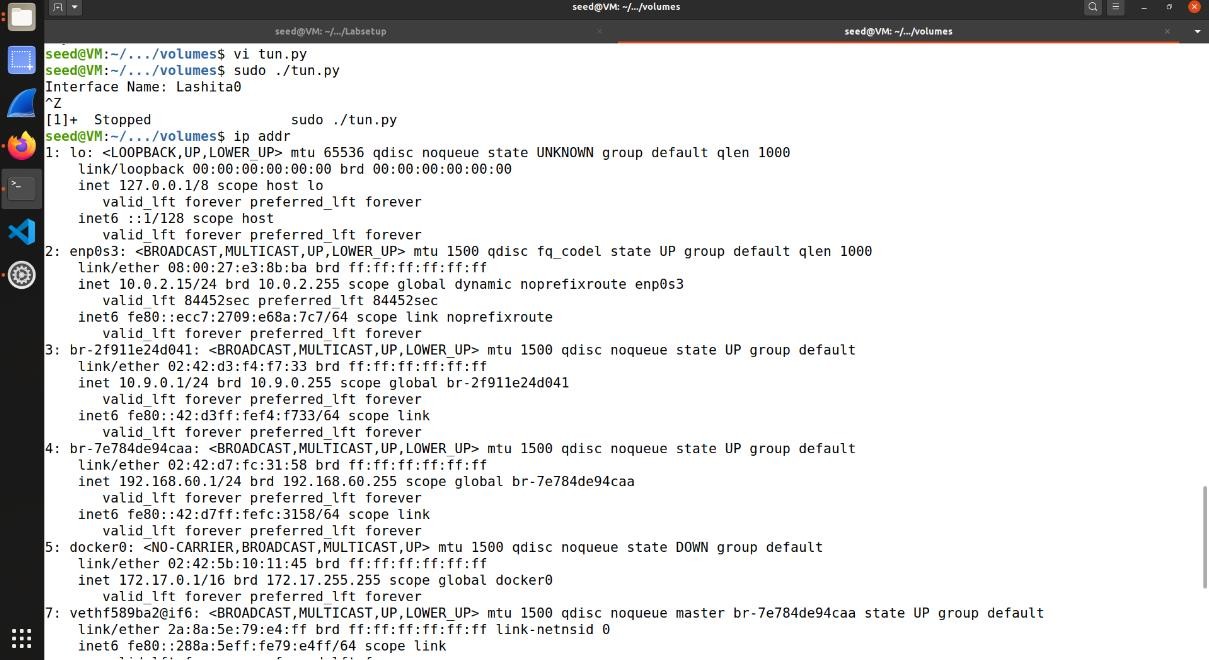


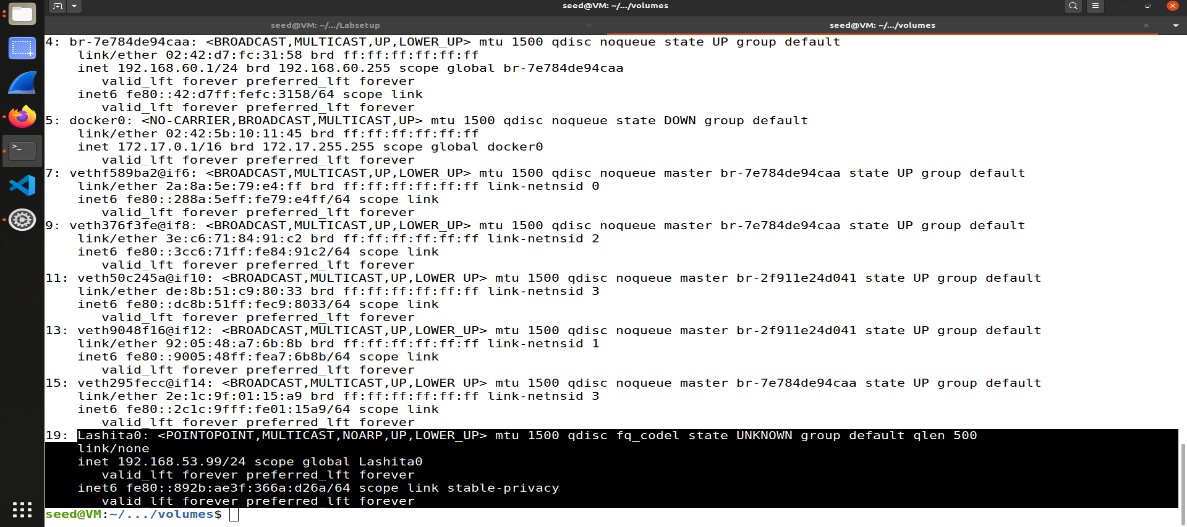
Changing the interface name in tun.py file and executing the file.



Task 2.b: Set up the TUN Interface

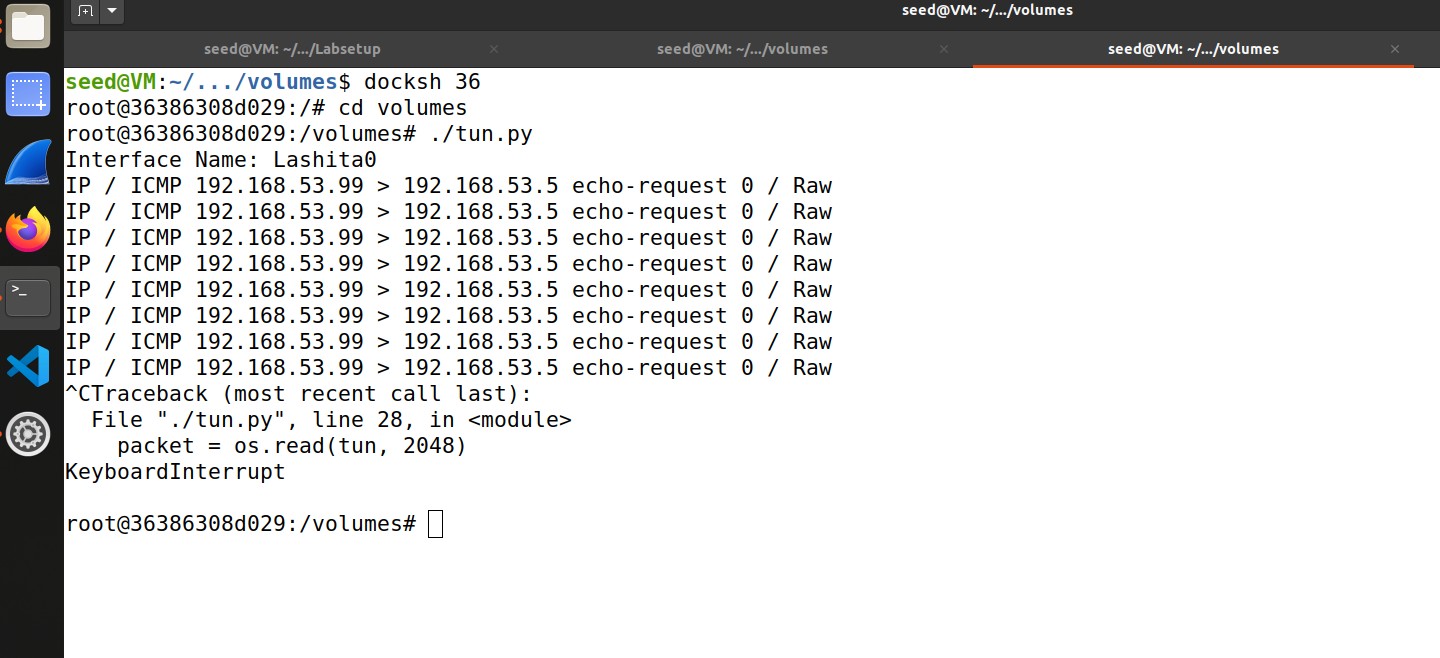
Adding two lines of code to tun.py, so the configuration can be automatically performed by the program.

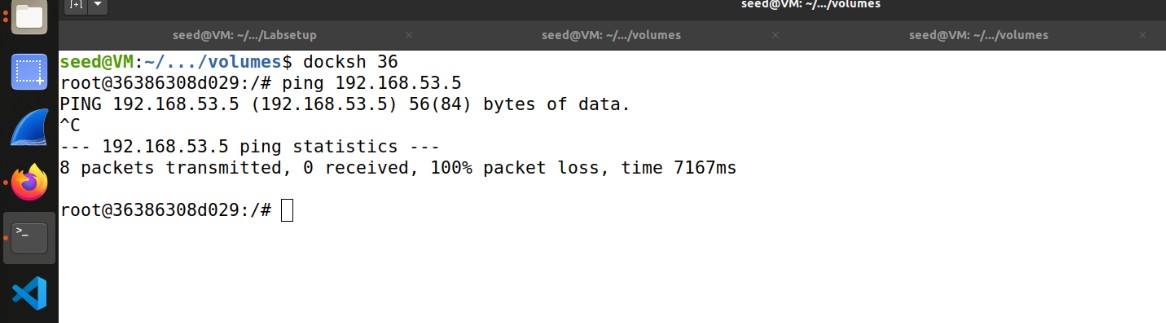
Output after running tun.py and check for interface details using command ip addr.



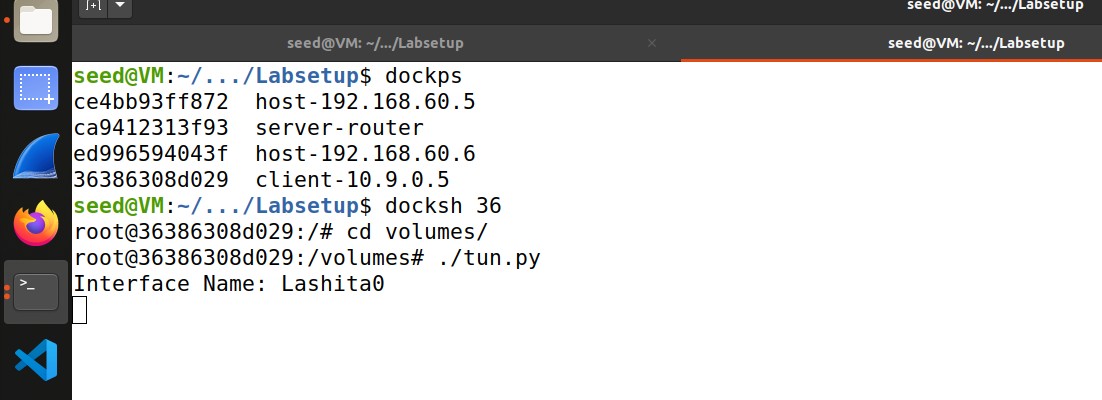
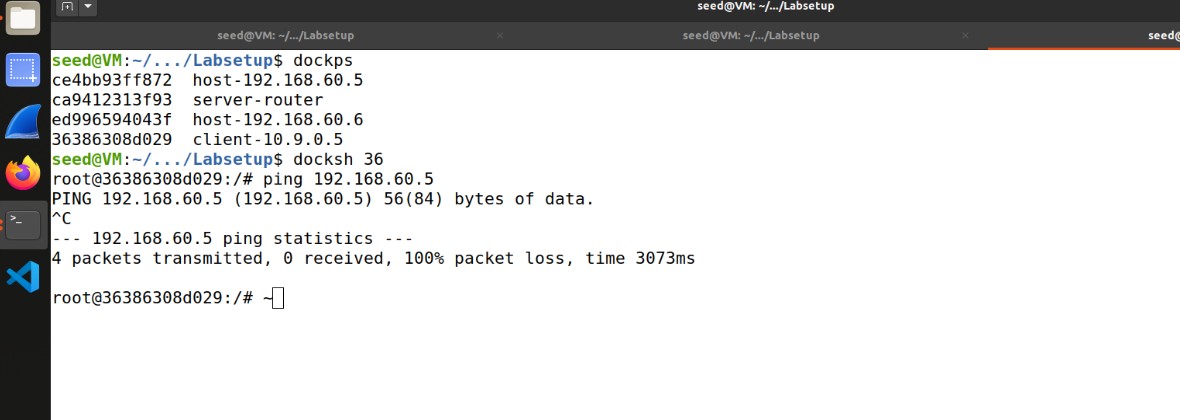
Task 2.c: Read from the TUN Interface

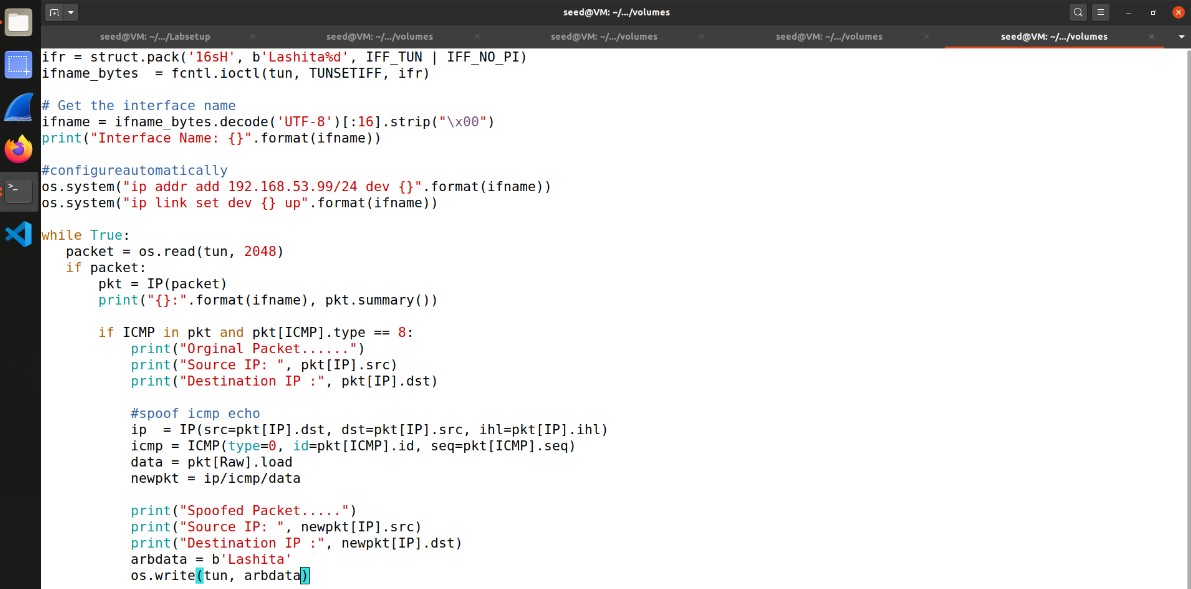
while loop is replaced with the one in tun.py.

Running tun.py file and pinging 192.168.53.5 from Host U.

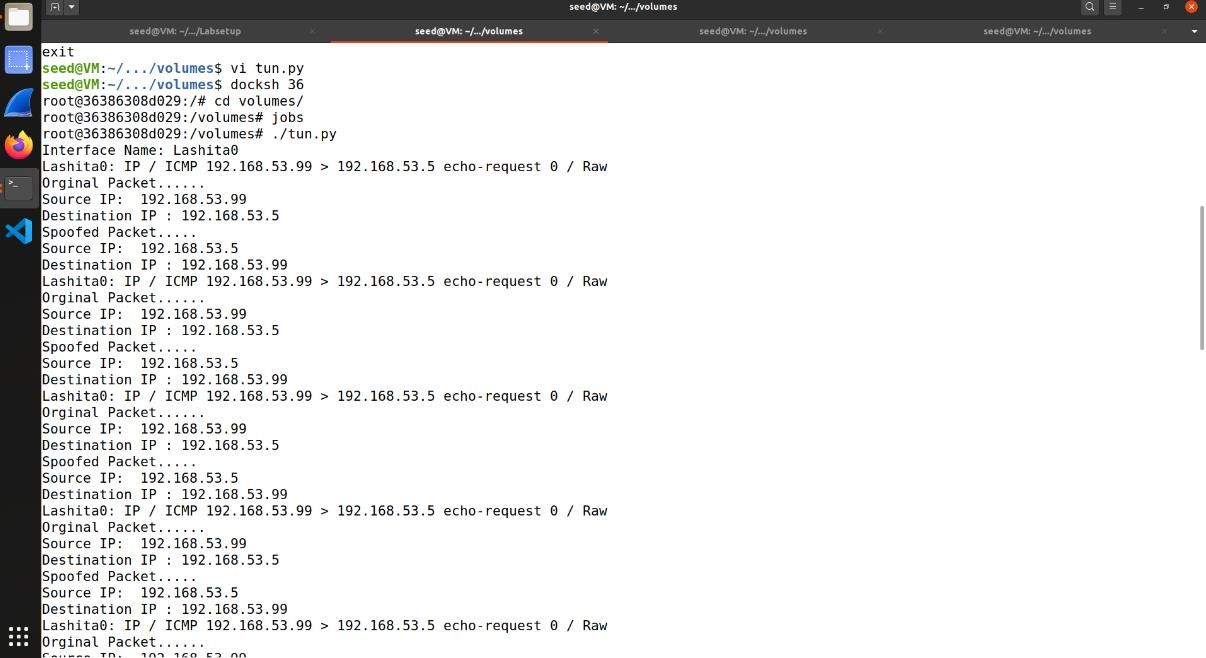


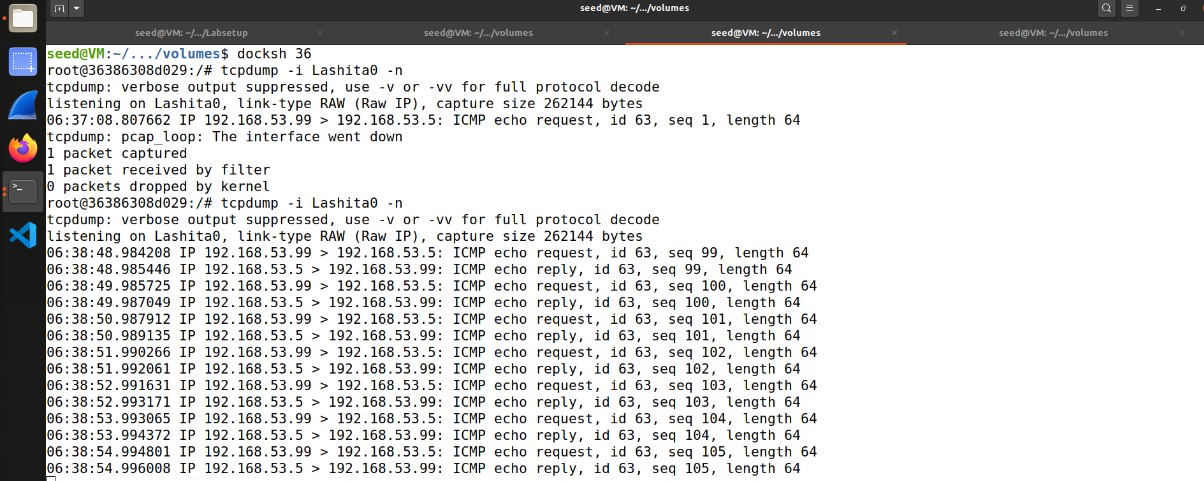
Now ping 192.168.60.5 from Host U while running tun.py



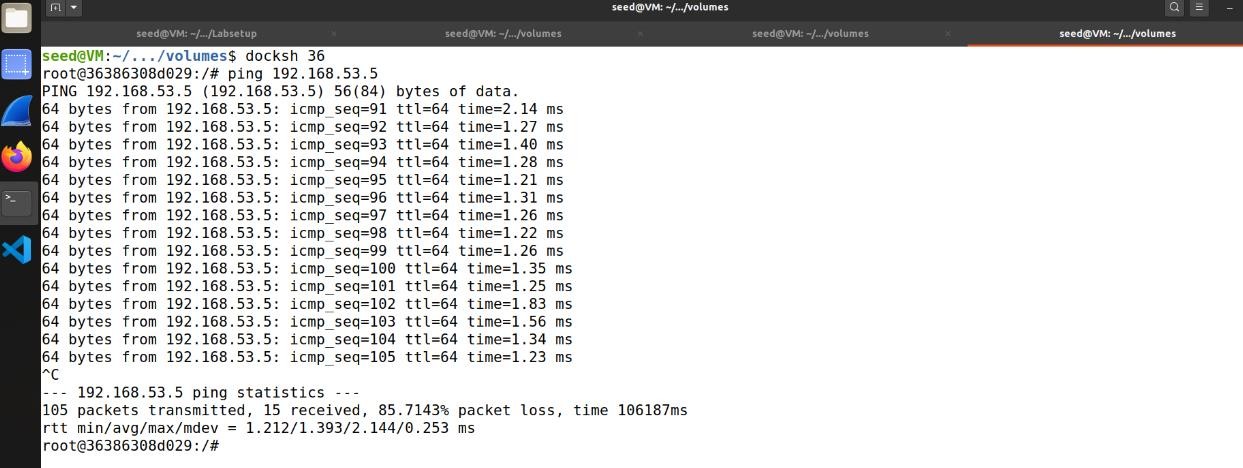
Task 2.d: Write to the TUN Interface

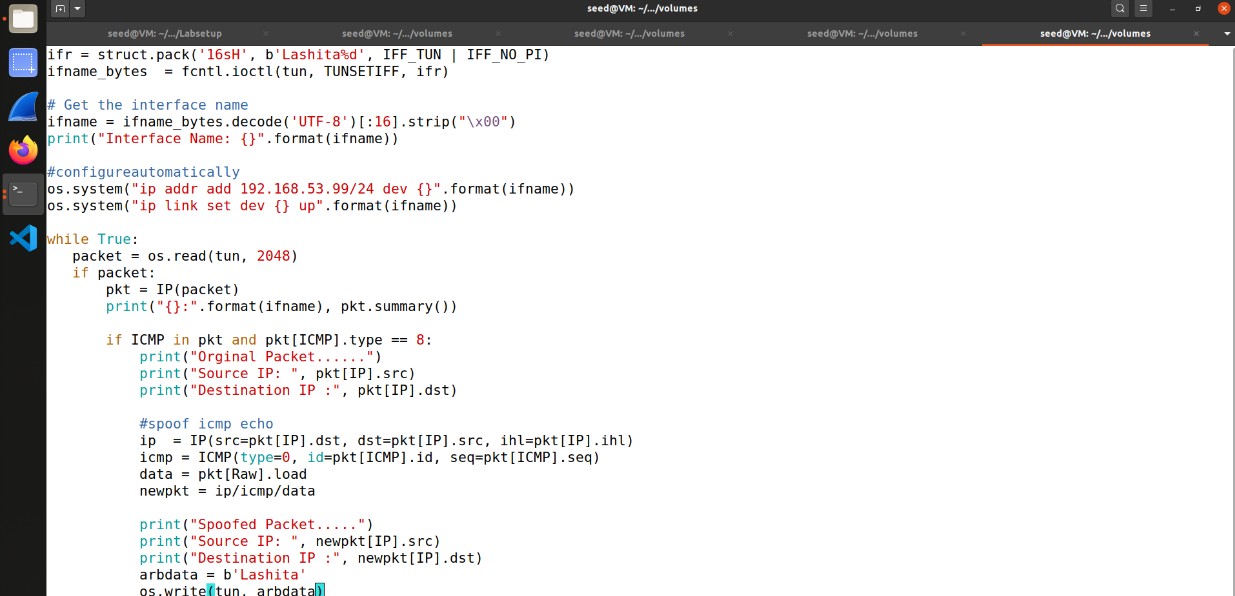
Modifying tun.py for getting a packet from the TUN interface, if this packet is an ICMP echo request packet, construct a corresponding echo reply packet and write it to the TUN interface.

Running tun.py on HostU.

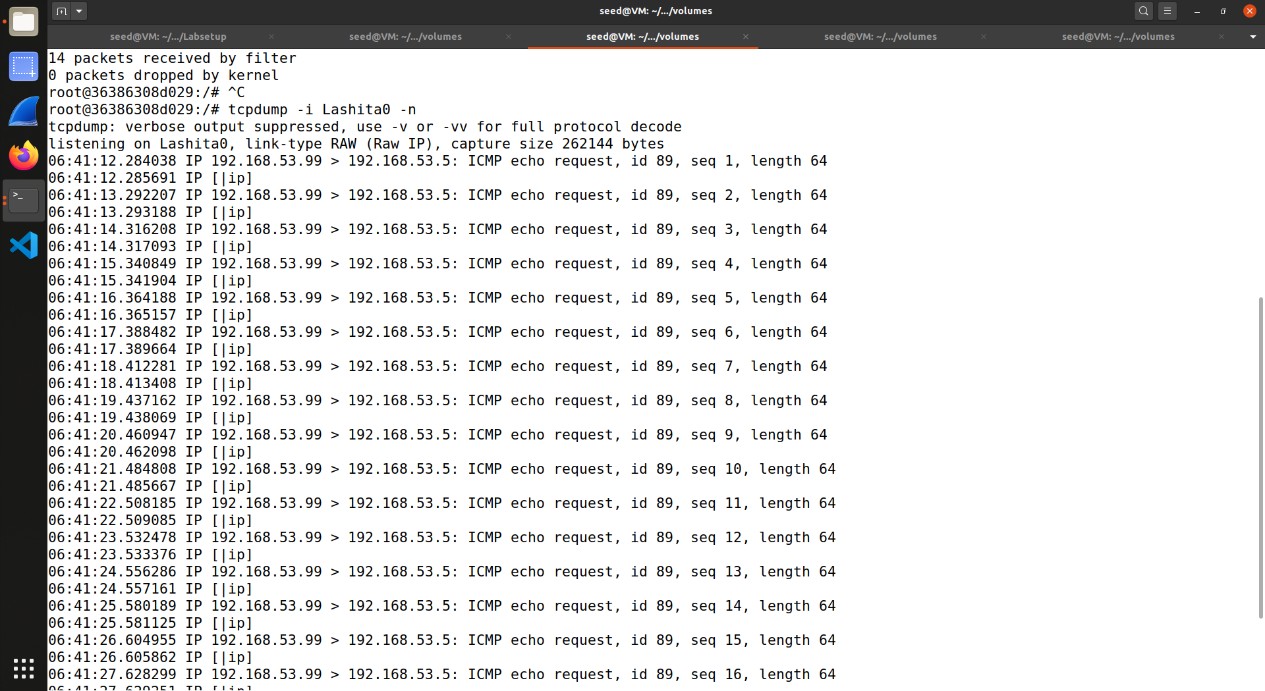
Running tcpdump -i lashita0

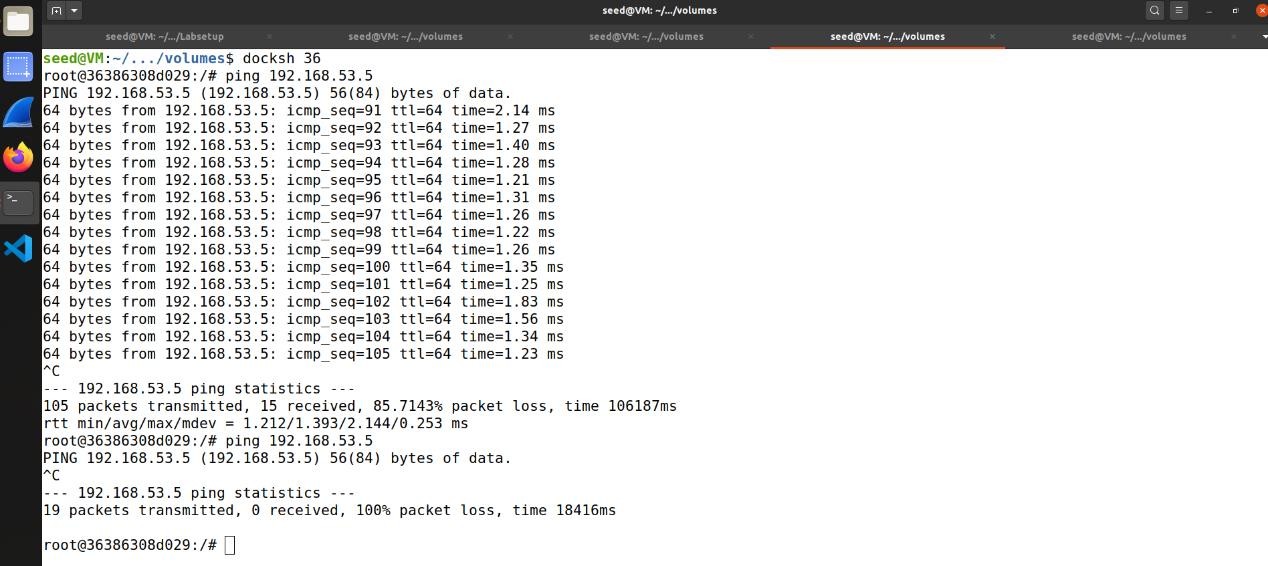
-n.

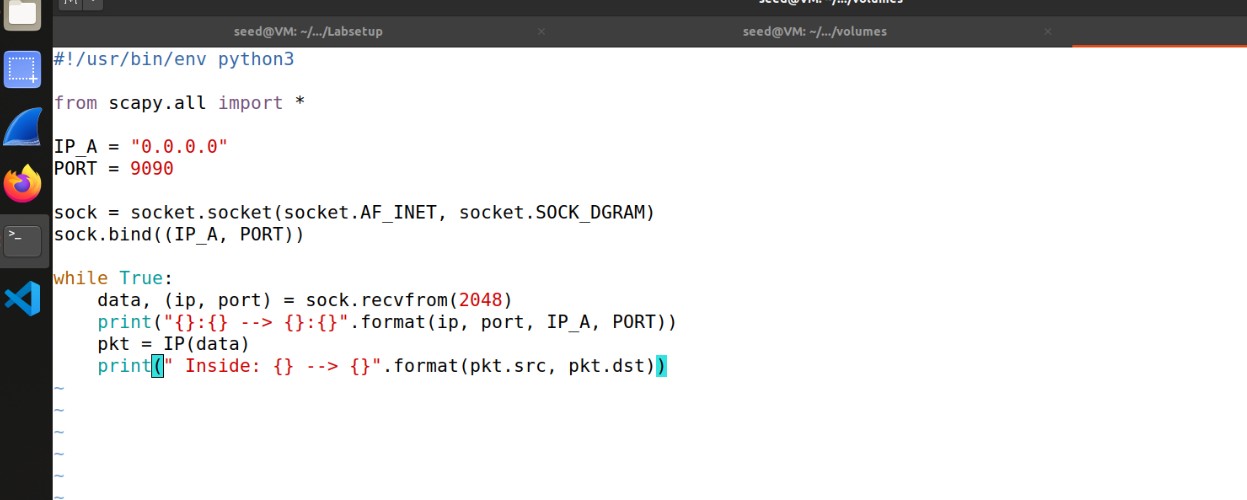
Pinging 192.168.53.5

Writing arbitrary data instead IP packet.

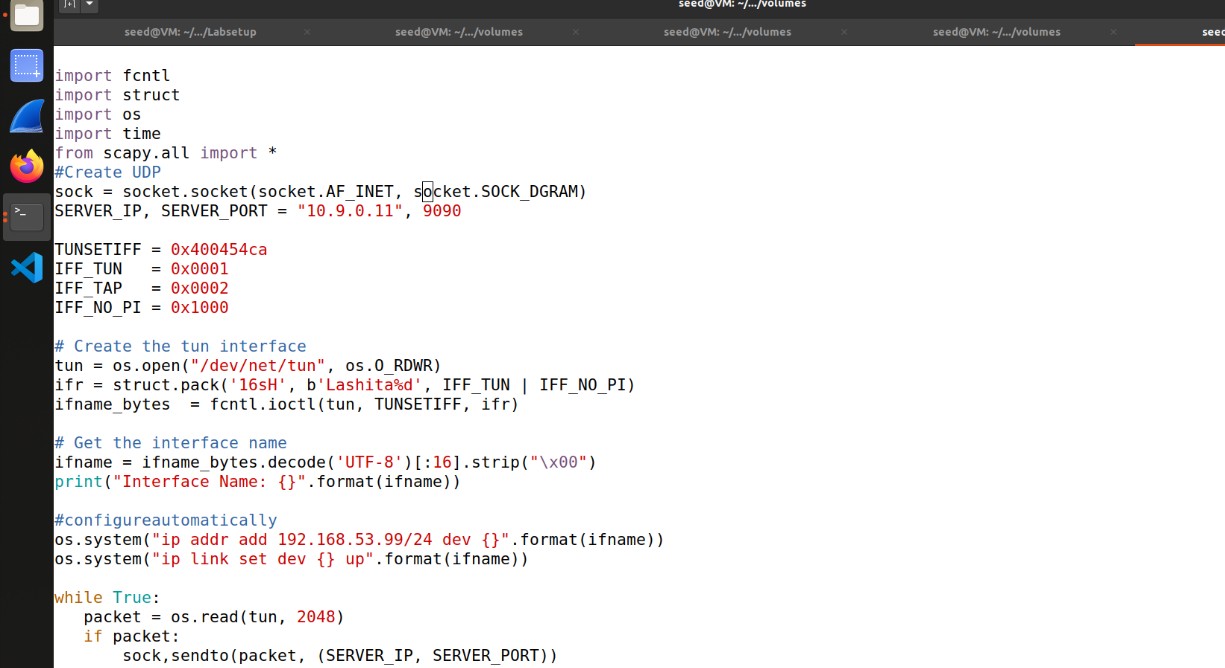
Running tun.py on Host U.

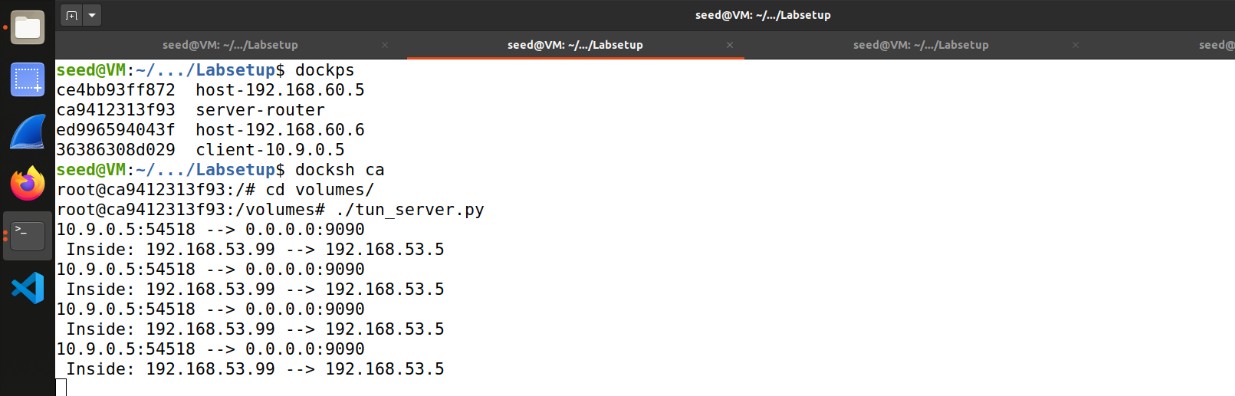
Running tcpdump -i lashita0 -n.

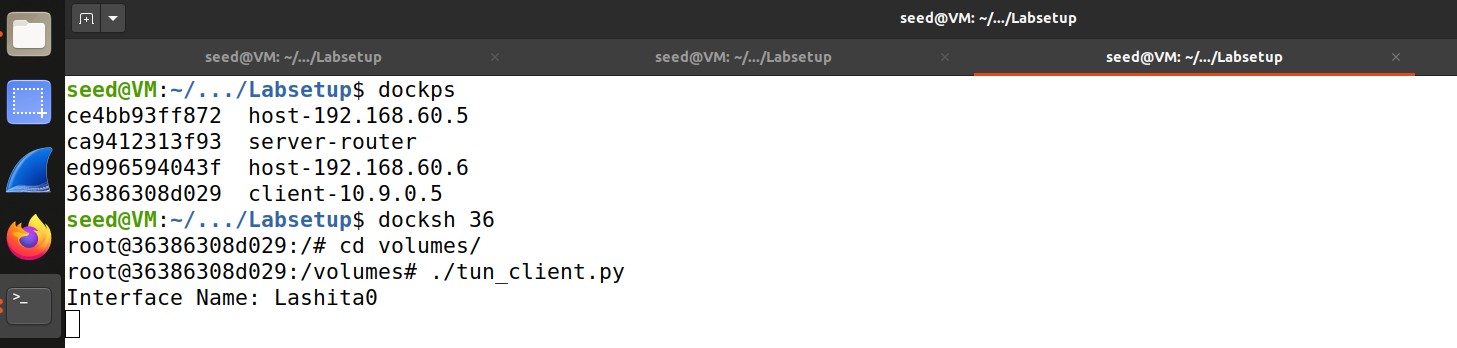
Pinging 192.168.53.5

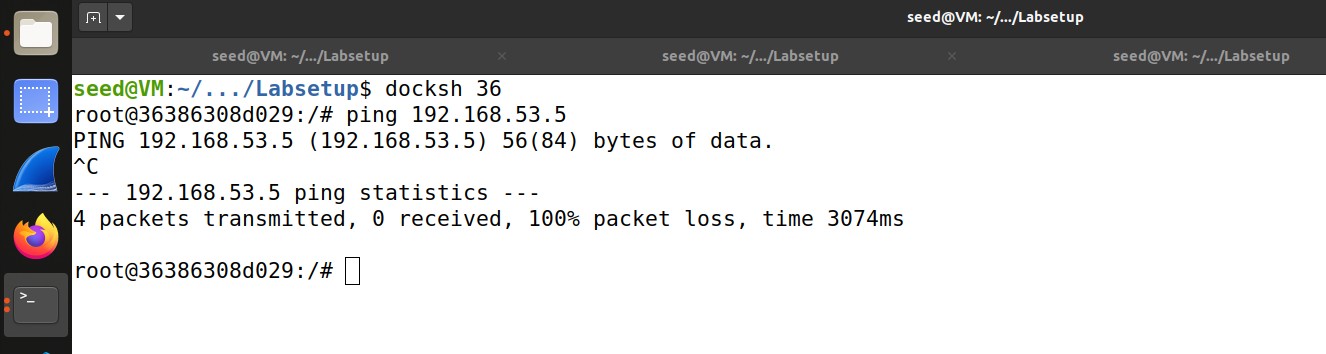
1. Task 3: Send the IP Packet to VPN Server Through a Tunnel

The server program tun\_server.py.

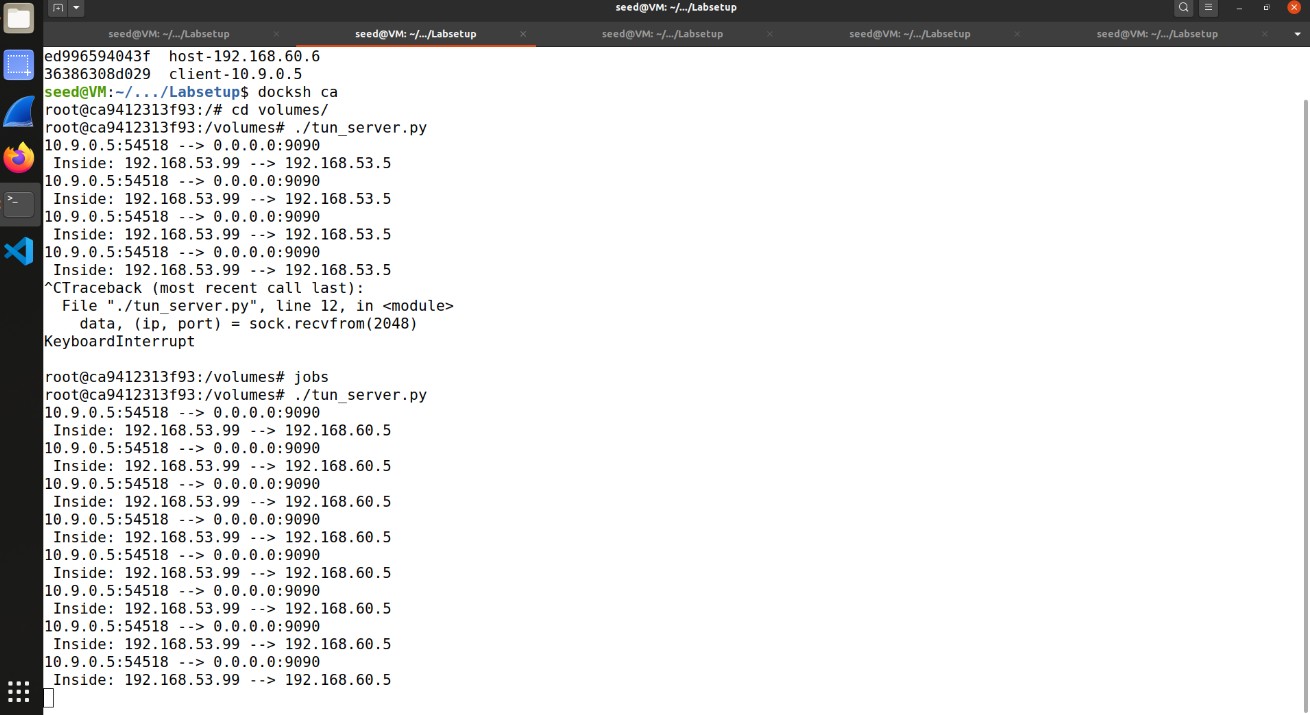
Implementing the client program tun\_client.py.

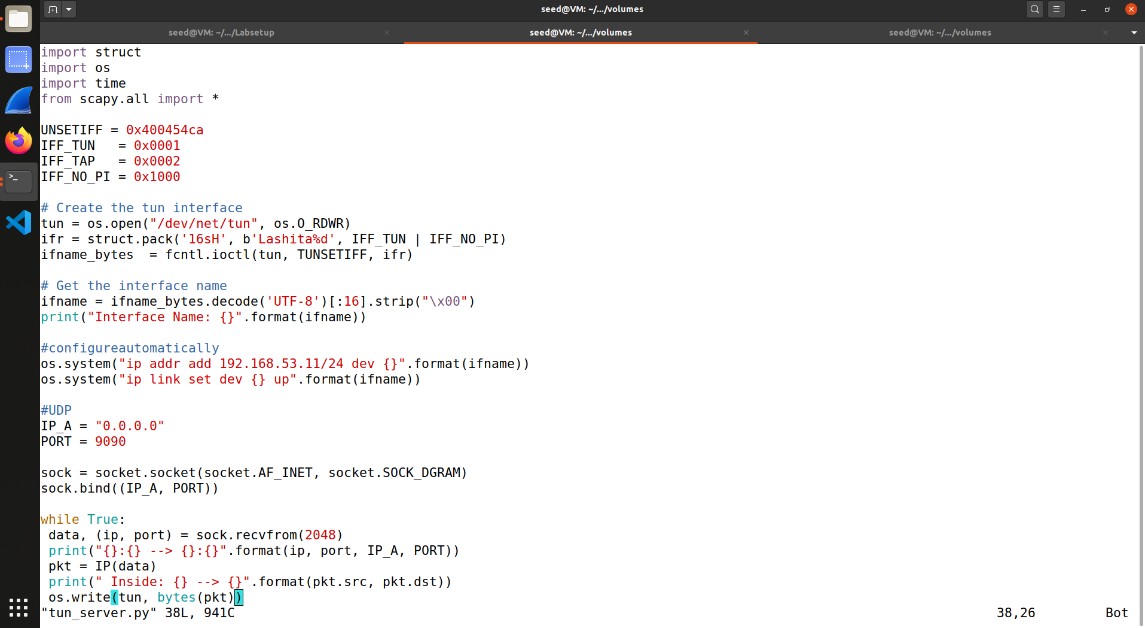
Running the tun\_server.py on VPN server and running tun\_client.py on Host U, then checking ping 192.168.53.0



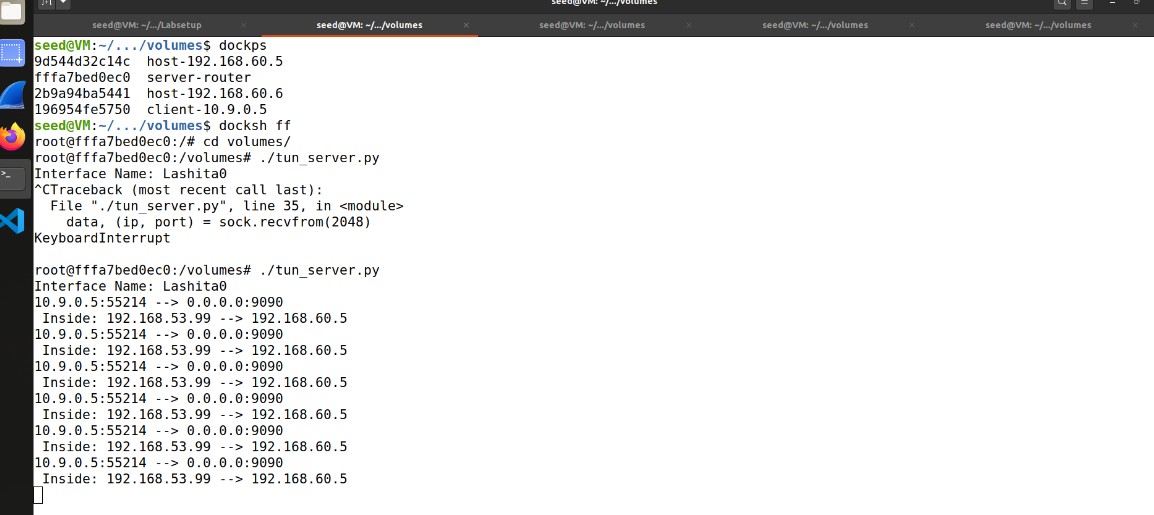


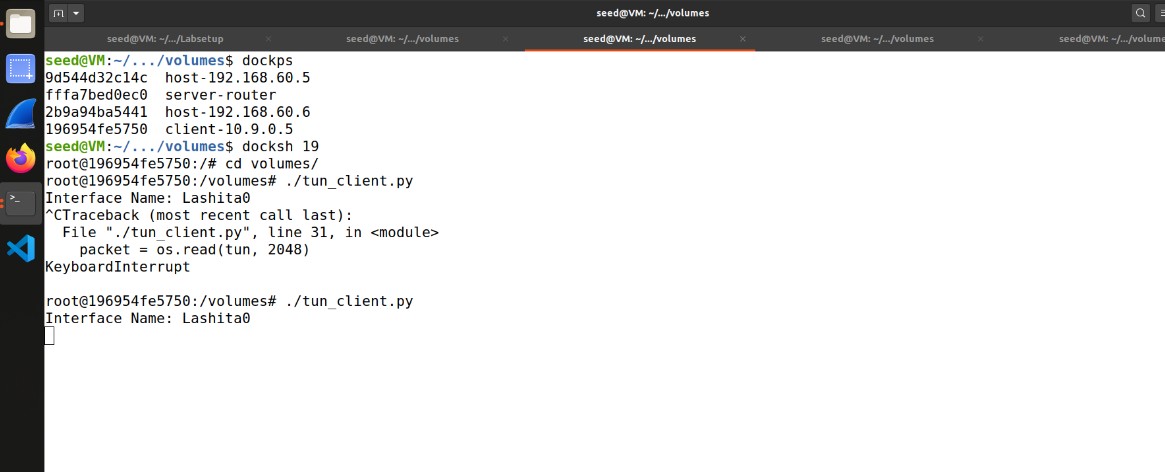
Adding entry to touting table and ping Host V, and checking whether the ICMP packet is sent to VPN Server through the tunnel.

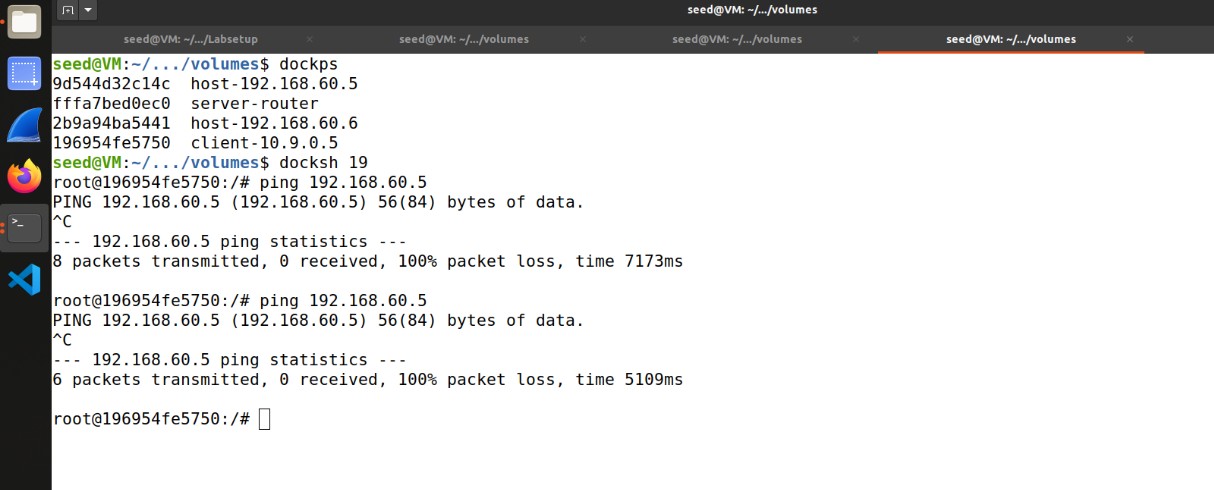


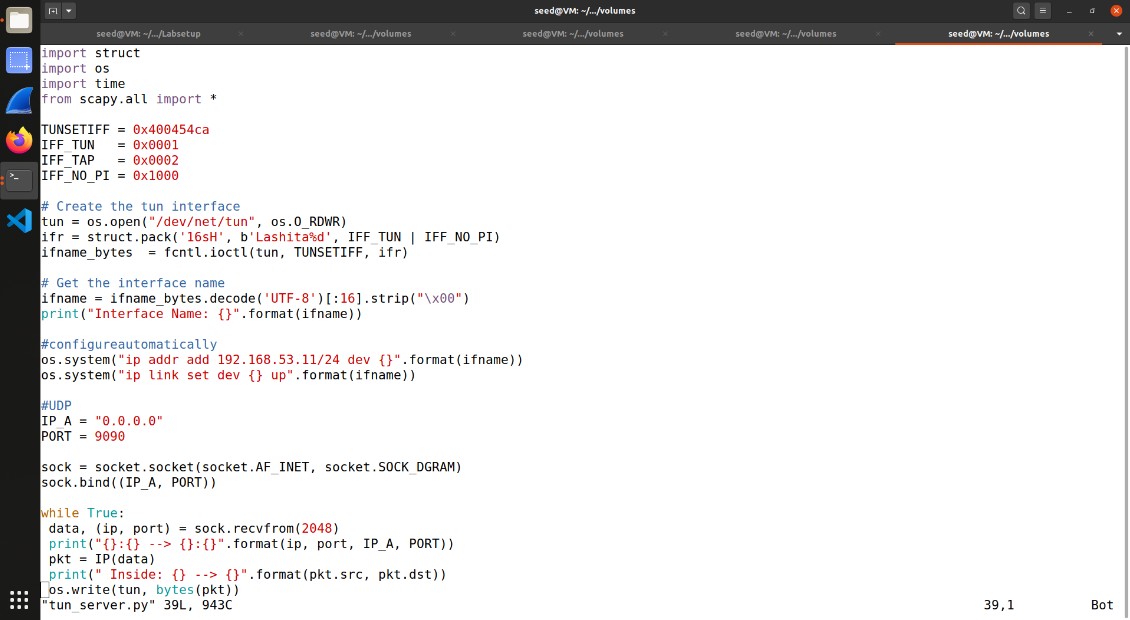
1. Task 4: Set Up the VPN Server

Modifying tun\_server.py and tun\_cliet.py.

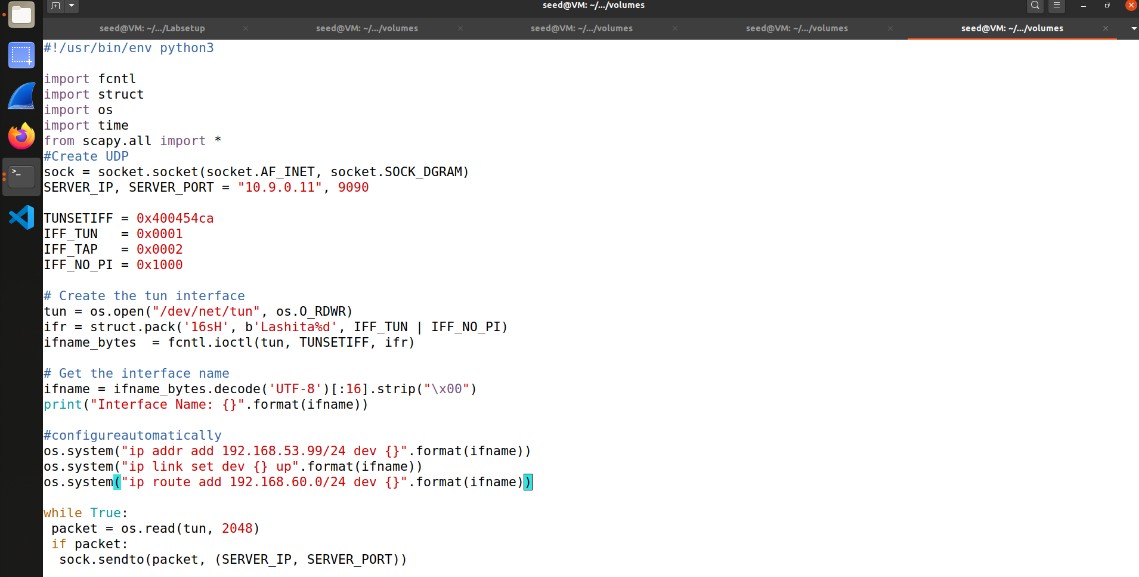
Running tun\_server.py on server.

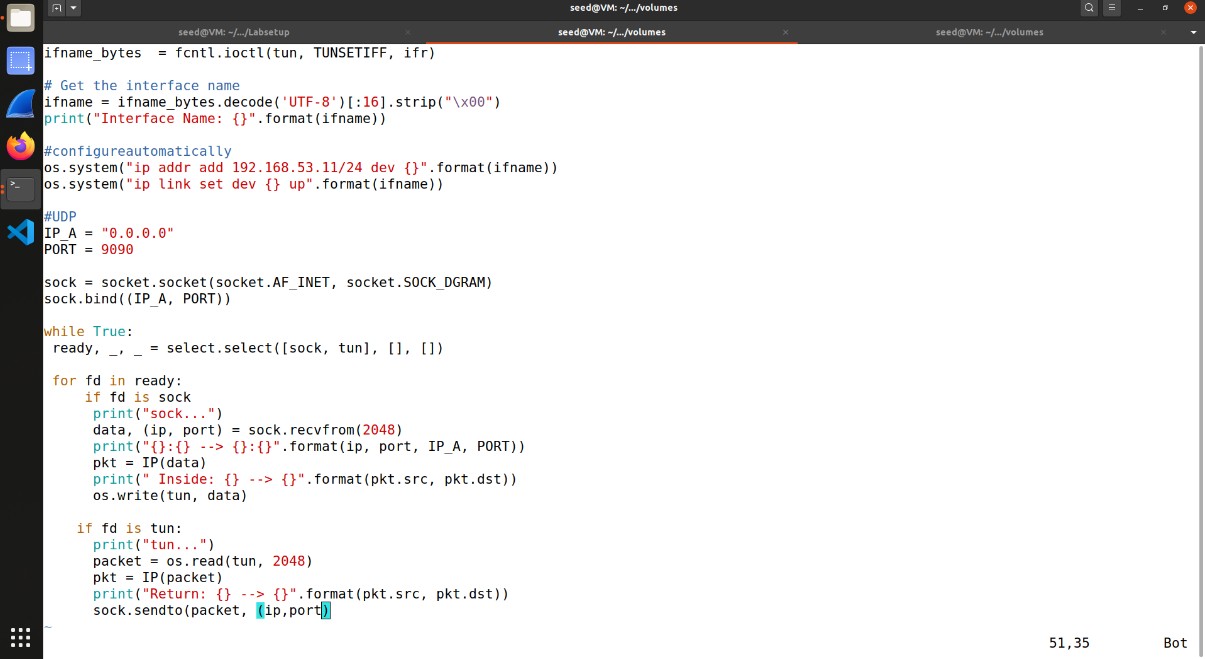
Running ./tun\_client.py on Host U and ping 192.168.60.5



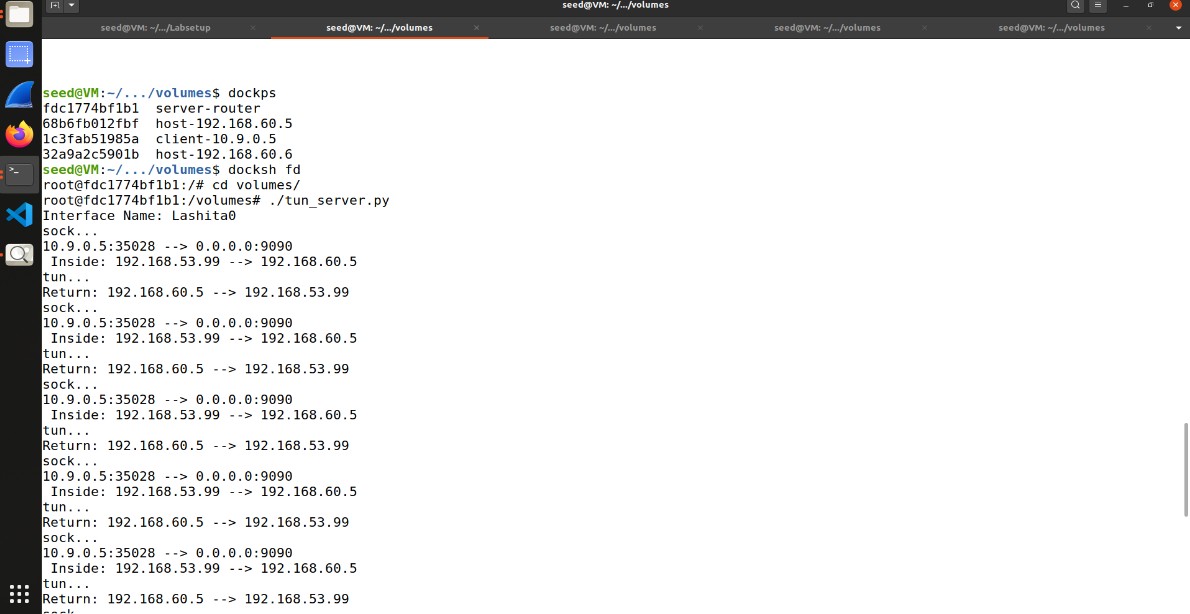
1. Task 5: Handling Traffic in Both Directions.

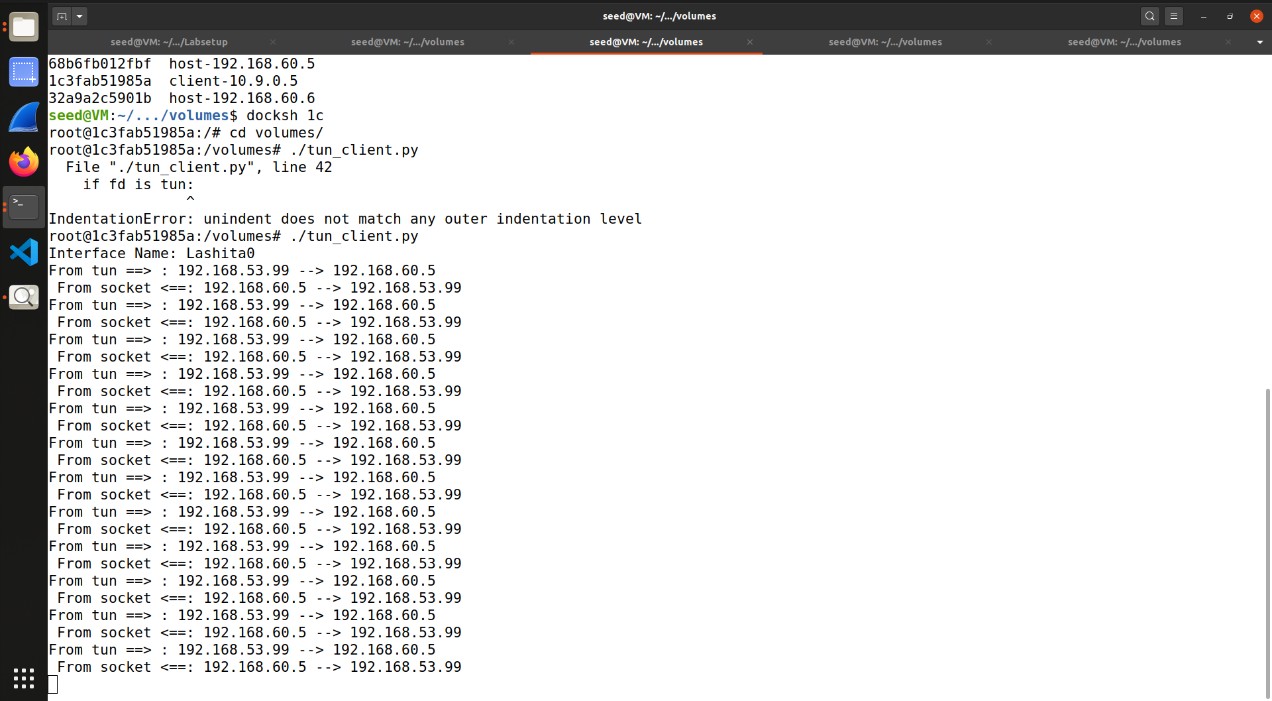
Modifying tun\_server.py and tun\_client.py.

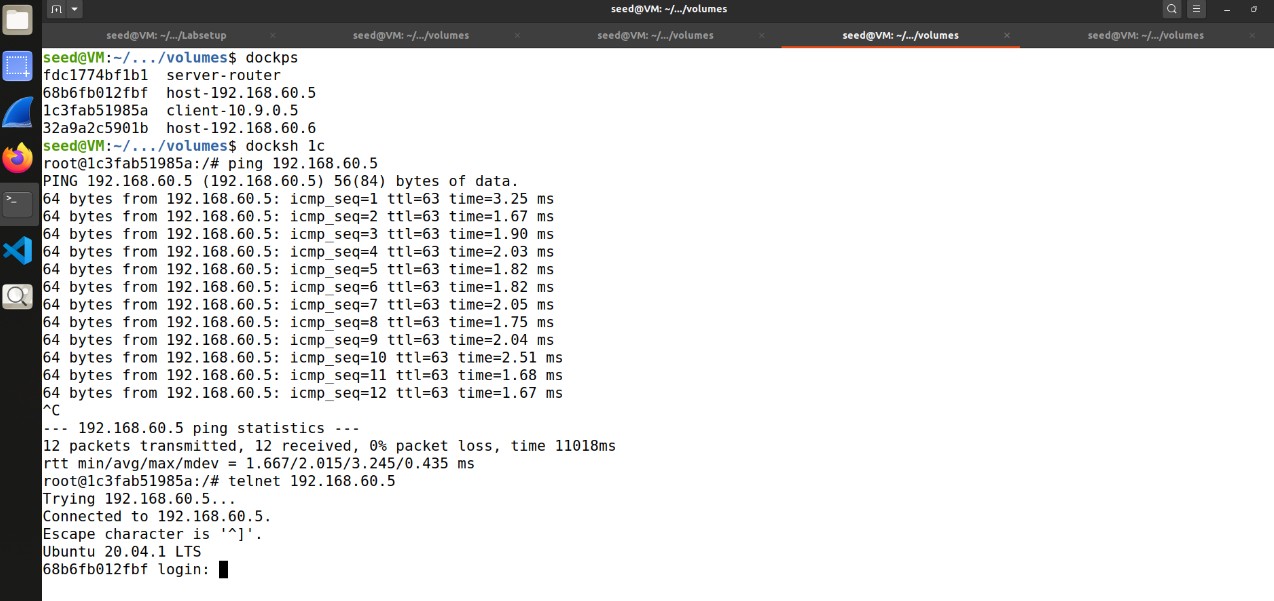
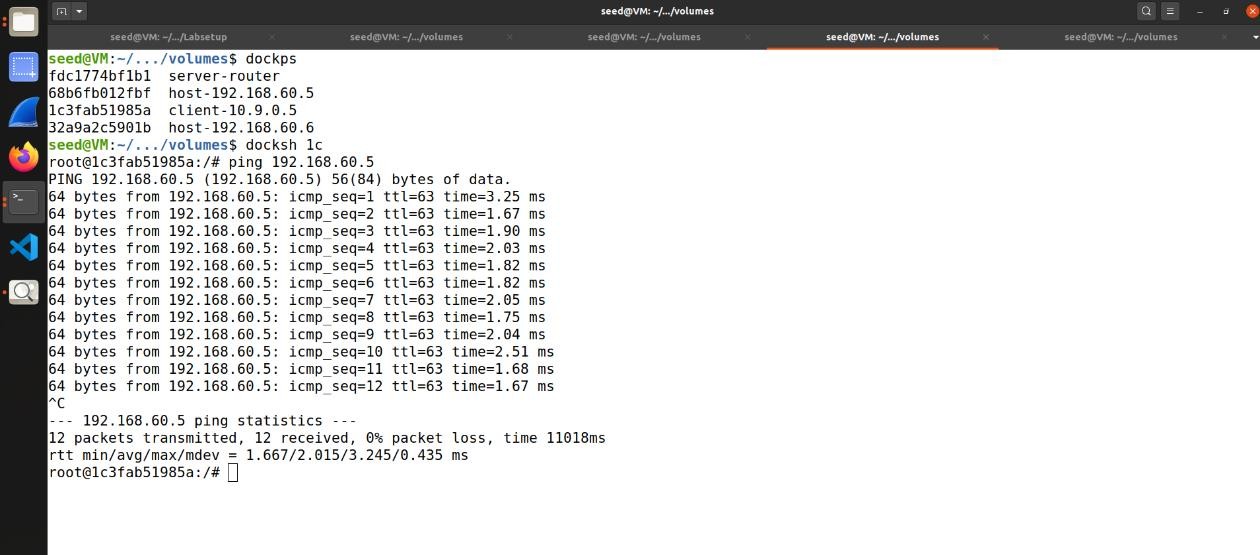




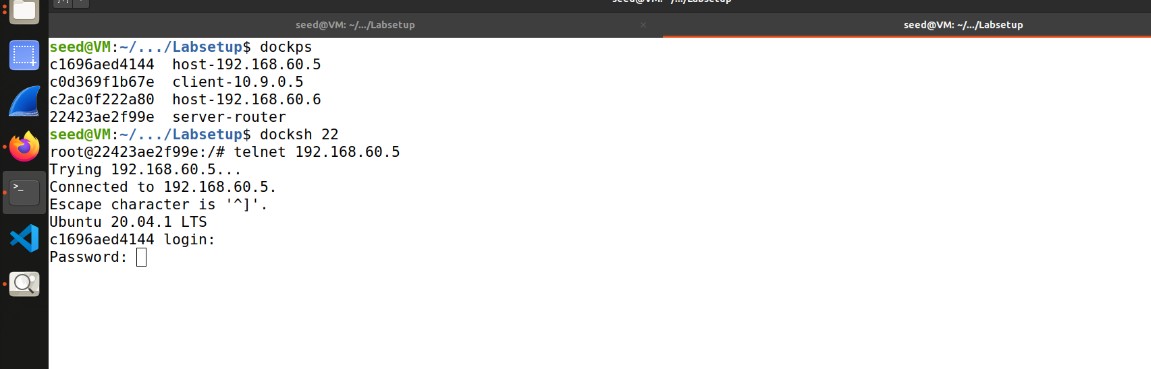


Running tun\_server.py on server.

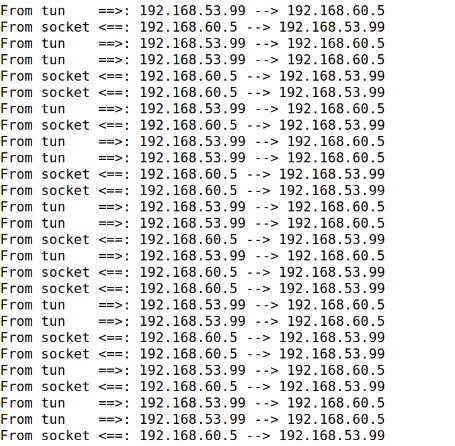
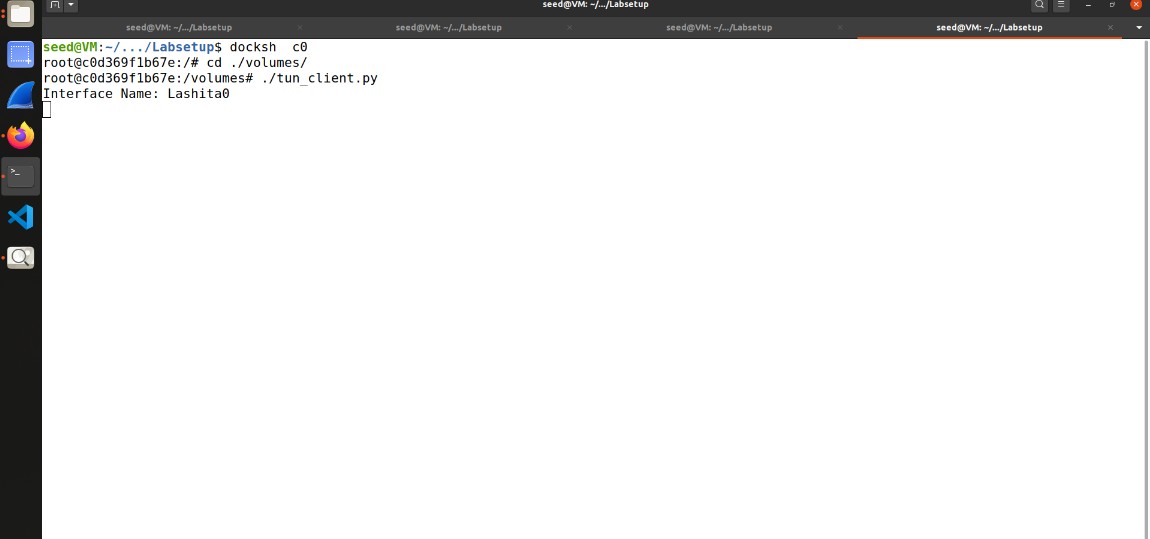
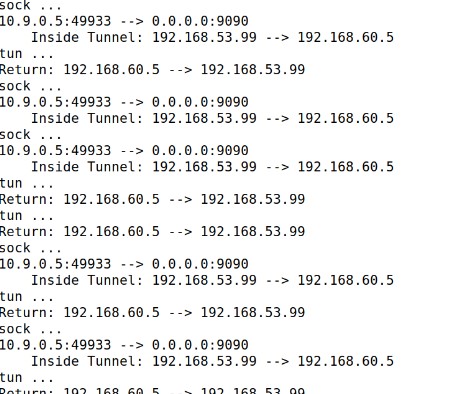
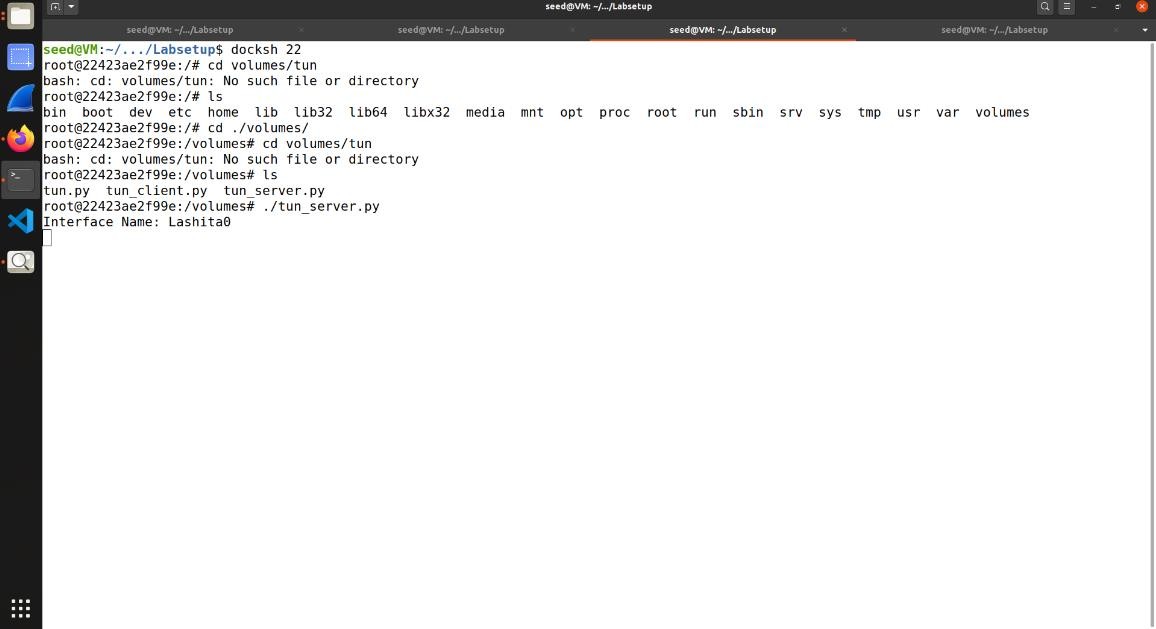
Running tun\_client.py on Host U and checking ping, telnet to 192.168.60.5



1. Task 6: Tunnel-Breaking Experiment.

Establishing telnet from Host U to Host V.

Running tun\_server.py and tun\_client.py and then breaking one connection followed by reconnecting.



1. Task 7: Routing Experiment on Host V.

