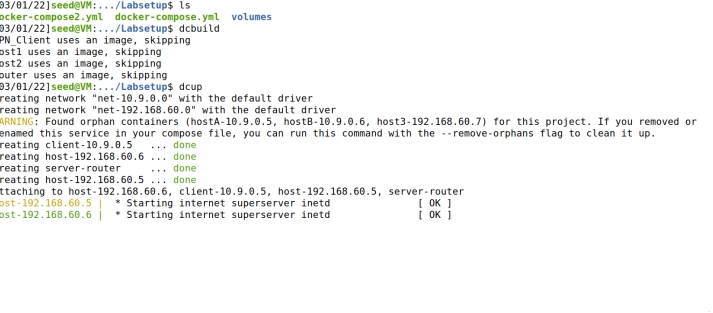
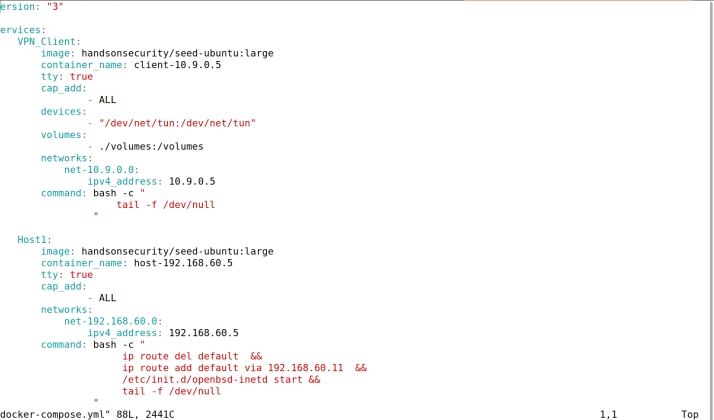
Lab 6: VPN Lab: The Container Version

Dhanusha Mallavajjala

**Task 1: Network Setup**

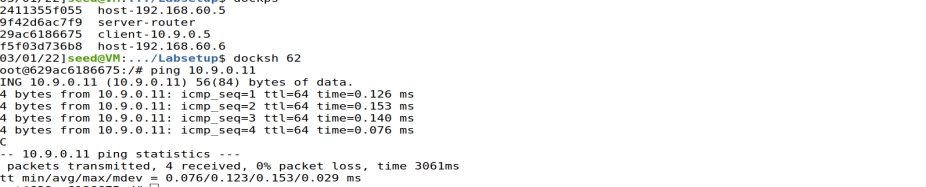


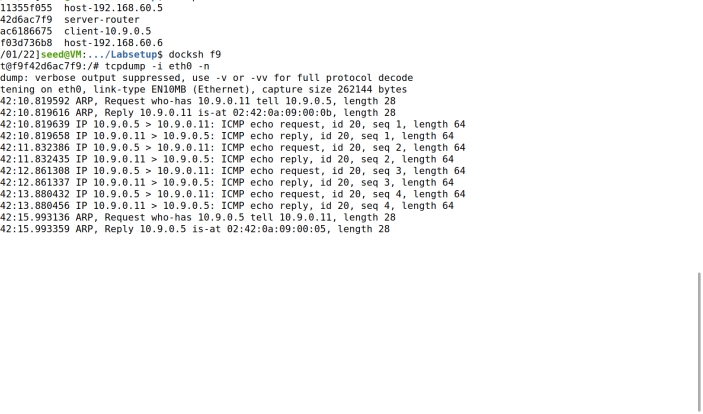


Packet sniffing:

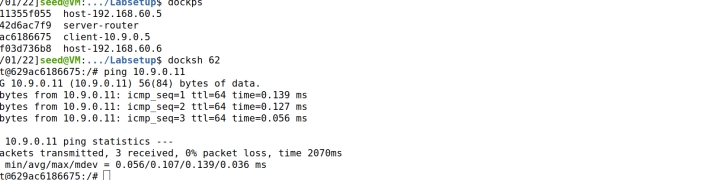
Running tcpdump on containers with below command and then ping from client to see number of packets

tcpdump -i eth0 -n

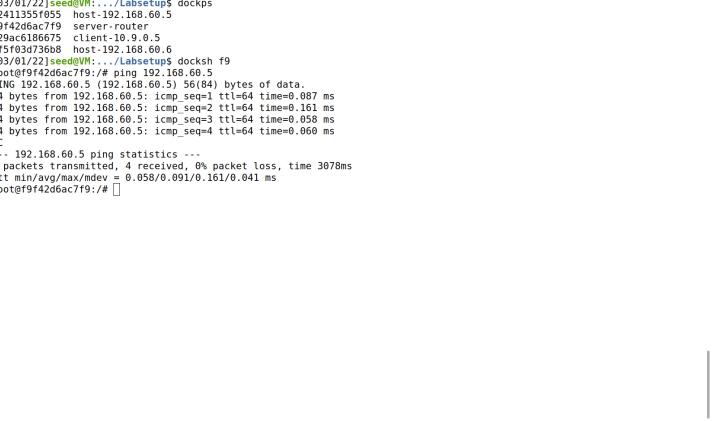




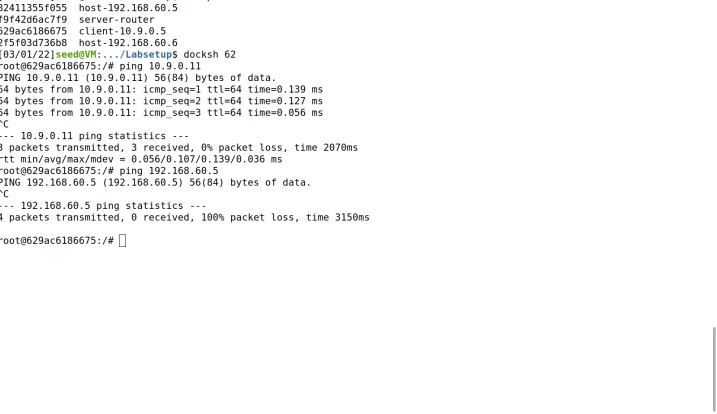
Host U can communicate with VPN Server



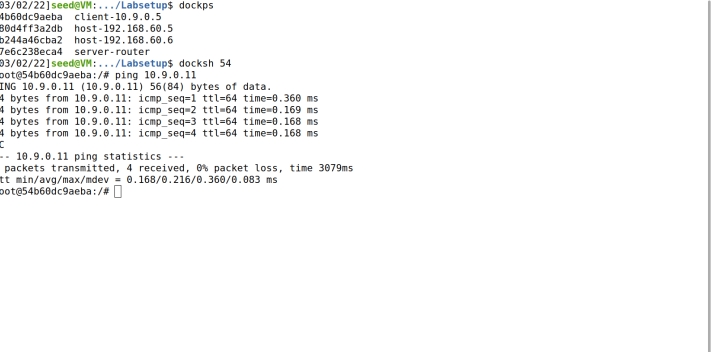
We can see that VPN Server can communicate with Host V.

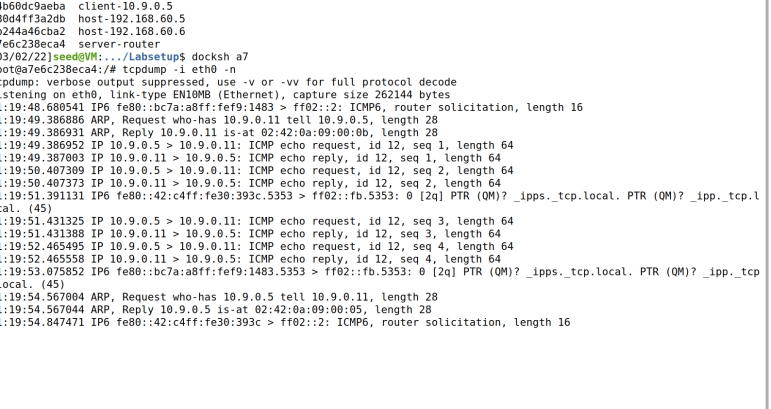


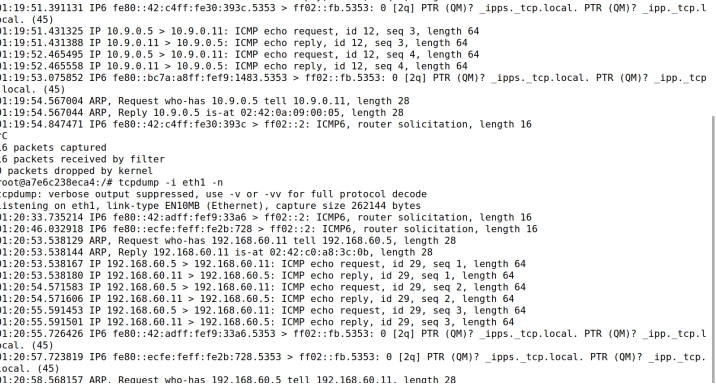
But Host U should not be able to communicate with Host V



Now lets run tcpdump on the router, and sniff the traffic on each of the network







**Task 2: Create and Configure TUN Interface**

**Task 2.a**: Name of the Interface

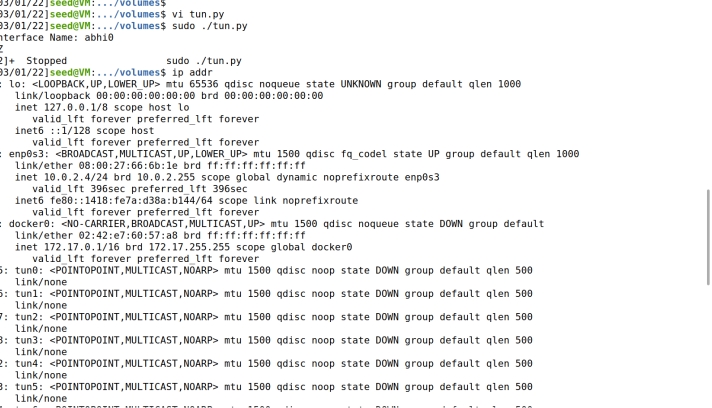
Running the tun.py using command sudo ./tun.py

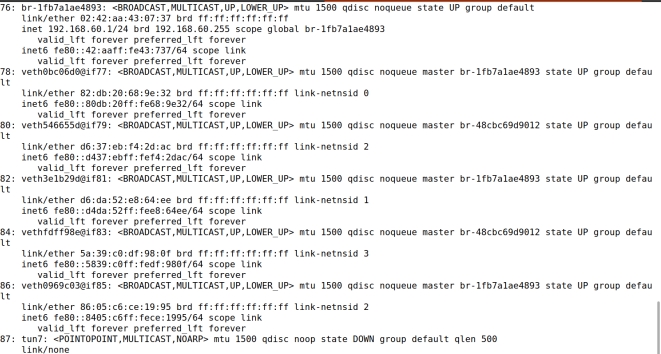


Changing the interface name in tun.py file and executing the file









**Task 2.b:** Set up the TUN Interface add the following two lines of code to tun.py, so the configuration can be automatically performed by the program

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))

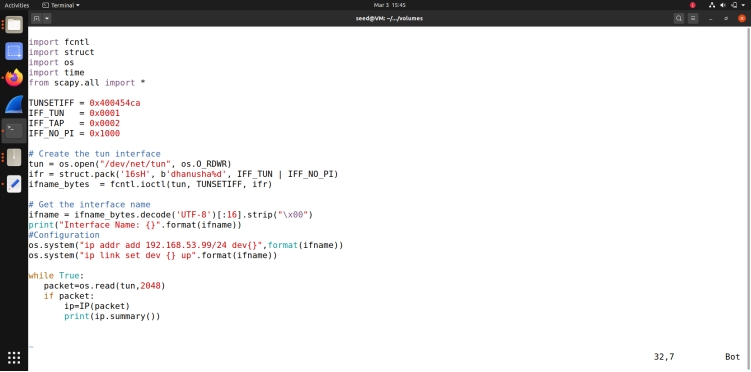
os.system("ip link set dev {} up".format(ifname))



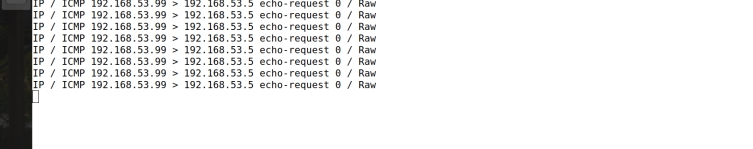
* Output after running tun.py and check for interface details using command ip addr

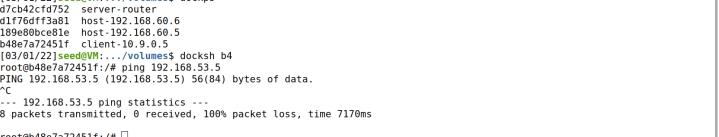


* **Task 2.c**: Read from the TUN Interface
* while loop is replaced with the one in tun.py



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





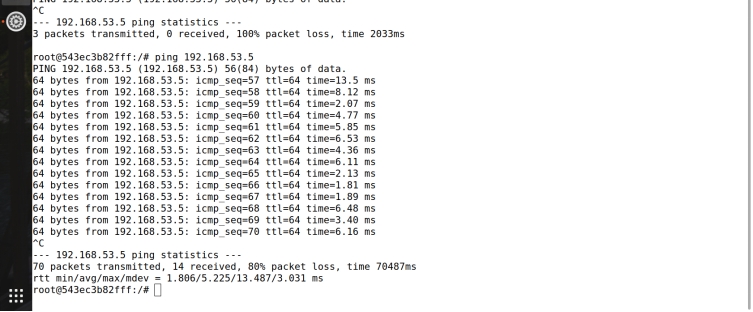
Packets are printed out to our echo request as 192.168.53.5 doesn’t exist so we receive zero replies

Now ping 192.168.60.5 from Host U while running tun.py

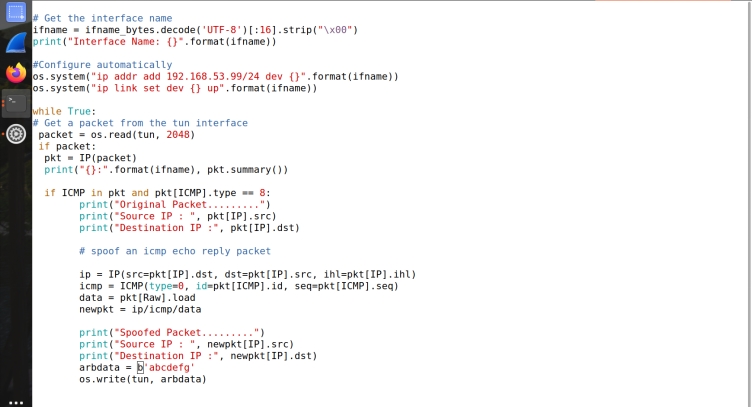
The packets are transmitted but not sent to the time interface, therefore we didn’t any print out.

**Task 2.d: Write to the TUN Interface**

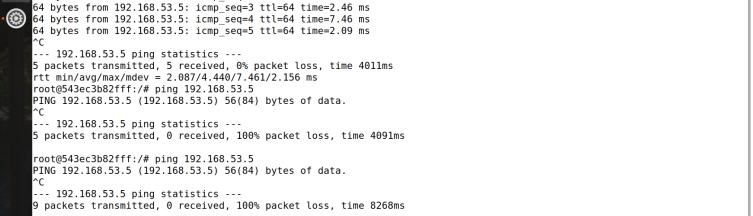
* Modifying tun.py for getting a packet from the TUN interface, if this packet is an ICMP echo request packet, construct a corresponding echo reply packet and write it to the TUN interface.
* Running tun.py on HostU,
* Running tcpdump -I dhanusha0 -n
* Ping 192.168.53.5



Writing arbitrary data instead IP packet

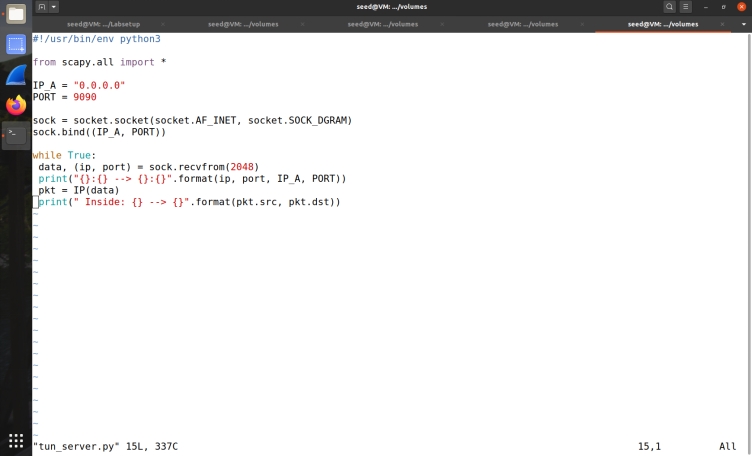


* Running tun.py on HostU,
* Running tcpdump -i dhanusha0 -n
* Ping 192.168.53.5

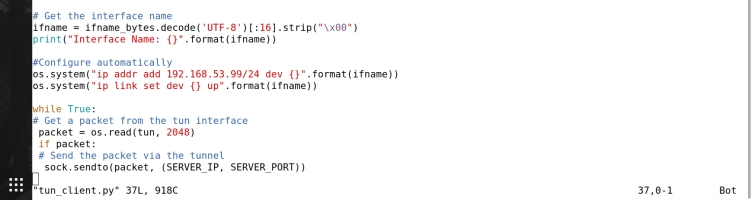


**Task 3: Send the IP Packet to VPN Server Through a Tunnel**

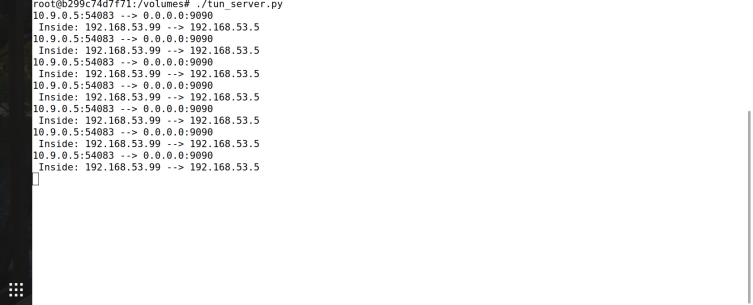
* The server program tun\_server.py.

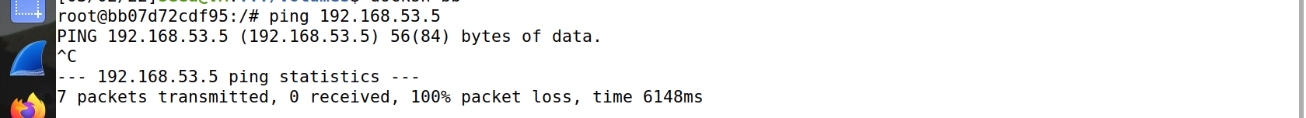


* Implementing the client program tun\_client.py



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





The tunnel is working as the packet is put inside the udp payload and is received by the server.

* **Task 4: Set Up the VPN Server**
* Modifying tun\_server.py and tun\_cliet.py

Client side code-

#!/usr/bin/env python3

import fcntl

import struct

import os

import time

from scapy.all import \*

SERVER\_IP = "10.9.0.11"

SERVER\_PORT = 9090

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

# Create the tun interface

tun = os.open("/dev/net/tun", os.O\_RDWR)

ifr = struct.pack('16sH', b'tun%d', IFF\_TUN | IFF\_NO\_PI)

ifname\_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name

ifname = ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))

os.system("ip link set dev {} up".format(ifname))

# Set up routing

os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

# Create UDP socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

while True:

# this will block until at least one interface is ready

ready, \_, \_ = select.select([tun, sock], [], [])

for fd in ready:

if fd is sock:

data, (ip, port) = sock.recvfrom(2048)

pkt = IP(data)

print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))

os.write(tun, data)

if fd is tun:

packet = os.read(tun, 2048)

pkt = IP(packet)

print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))

sock.sendto(packet, (SERVER\_IP, SERVER\_PORT))

Server side code-

#!/usr/bin/env python3

import fcntl

import struct

import os

import time

from scapy.all import \*

IP\_A = "0.0.0.0"

PORT = 9090

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

# Create the tun interface

tun = os.open("/dev/net/tun", os.O\_RDWR)

ifr = struct.pack('16sH', b'tun%d', IFF\_TUN | IFF\_NO\_PI)

ifname\_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name

ifname = ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.1/24 dev {}".format(ifname))

os.system("ip link set dev {} up".format(ifname))

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.bind((IP\_A, PORT))

while True:

# this will block until at least one interface is ready

ready, \_, \_ = select.select([sock, tun], [], [])

for fd in ready:

if fd is sock:

print("sock ...")

data, (ip, port) = sock.recvfrom(2048)

pkt = IP(data)

print("{}:{} --> {}:{}".format(ip, port, IP\_A, PORT))

print(" Inside Tunnel: {} --> {}".format(pkt.src, pkt.dst))

os.write(tun, data)

if fd is tun:

print("tun ...")

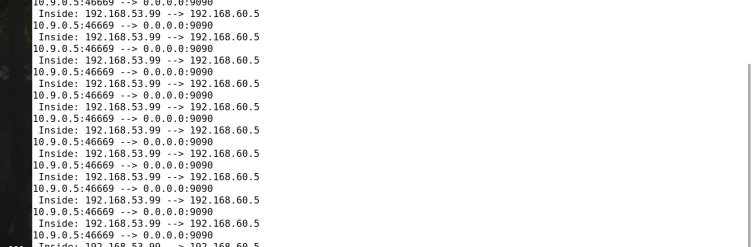
packet = os.read(tun, 2048)

pkt = IP(packet)

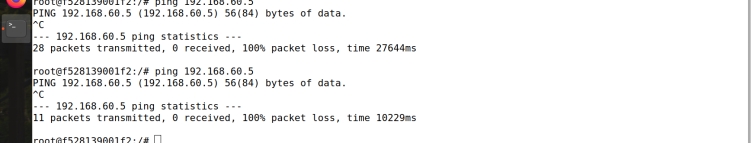
print("Return: {} --> {}".format(pkt.src, pkt.dst))

sock.sendto(packet, (IP\_A,PORT))

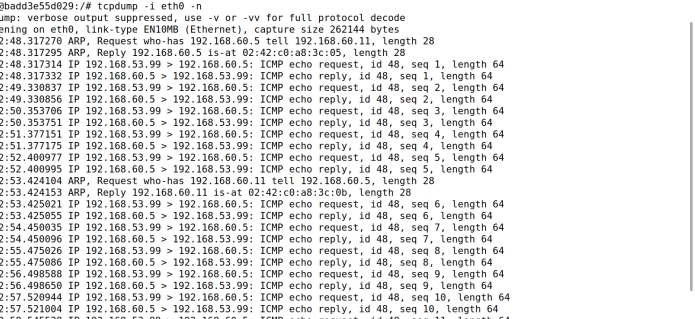
running tun\_server.py on server



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

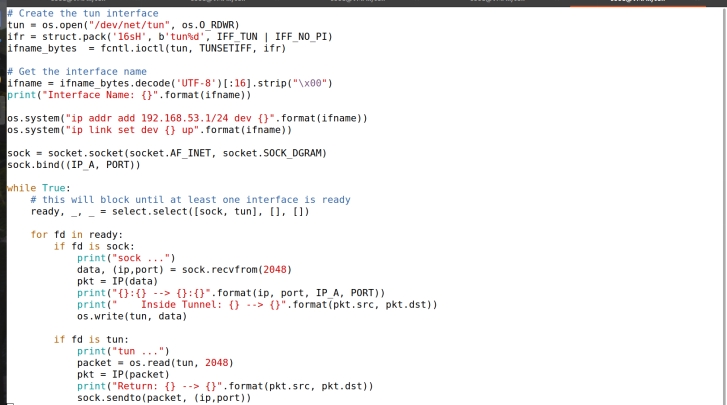


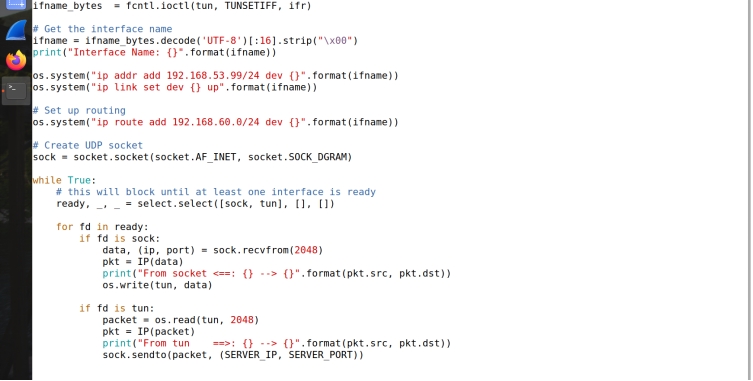
using tcp dump checking packets receiving



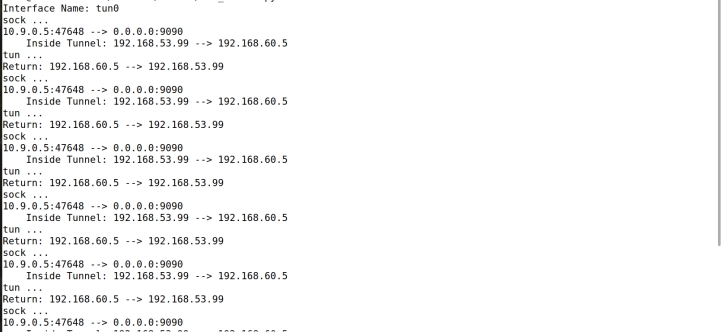
**Task 5: Handling Traffic in Both Directions**

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

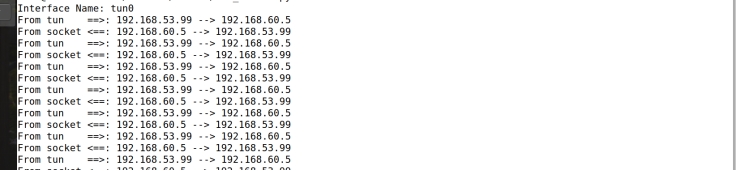


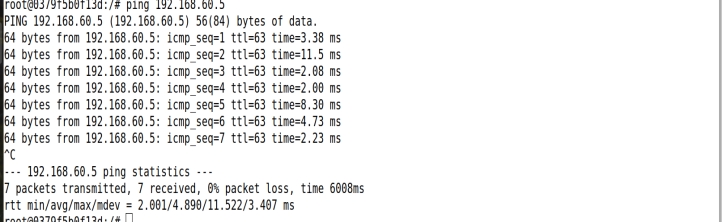


* Running tun\_server.py on server



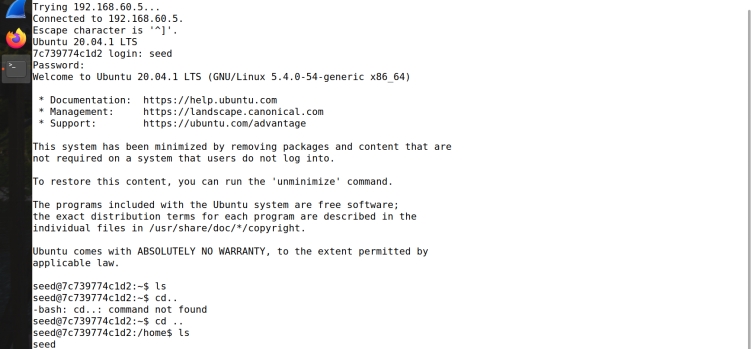
Running tun\_client.py on Host U and checking ping telnet to 192.168.60.5





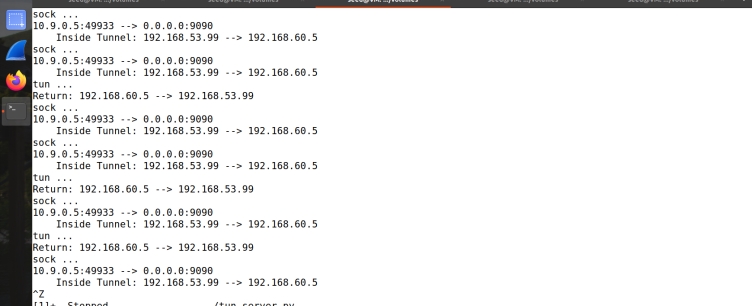
**Task 6: Tunnel-Breaking Experiment**

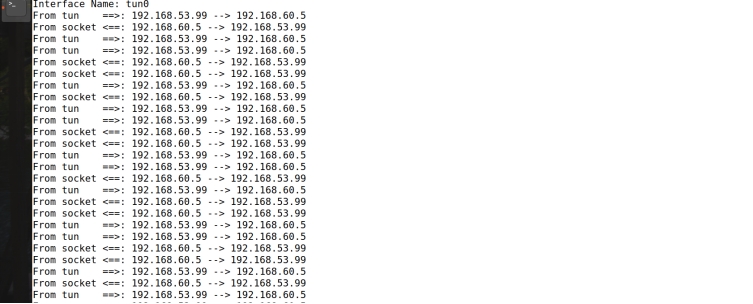
* Establishing telnet from Host U to Host V



Running tun\_server.py and tun\_client.py and then we should breaking one connection and then reconnect it.







**Task 7: Routing Experiment on Host V**

Running following command to enter new entries

1) ip route del default

2) ip route add 192.168.53.0/24 via 192.168.60.11

