Name- Harshit Agrawal SUID: 912538842

Task 1: Network Setup

```
root@d2227e2180d6:/# export PS1="U-10.9.0.5:\w\n\$>"
U-10.9.0.5:/
$>ping 10.9.0.11 -c 2
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.270 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.169 ms

--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.169/0.219/0.270/0.050 ms
U-10.9.0.5:/
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1012ms

U-10.9.0.5:/
```

```
[03/06/23]seed@VM:~/.../Labsetup$ docksh server-router
root@c0bb8b997c95:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
35: eth1@if36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever
37: eth0@if38: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
root@c0bb8b997c95:/# export PS1="router-10.9.0.11-192.168.60.11:\w\n\$>"
router-10.9.0.11-192.168.60.11:/
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.169 ms
```

```
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.169 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.119 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1032ms
rtt min/avg/max/mdev = 0.119/0.144/0.169/0.025 ms
router-10.9.0.11-192.168.60.11:/
$>tcdump -i eth0 -n
bash: tcdump: command not found
router-10.9.0.11-192.168.60.11:/
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:41:03.573210 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 1, length
 64
17:41:03.573406 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 1, length 6
4
17:41:04.590581 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 2, length
 64
17:41:04.590800 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 2, length 6
4
17:41:08.815035 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
17:41:08.815234 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
17:41:08.815243 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
17:41:08.815247 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
^C
8 packets captured
8 packets received by filter
0 packets dropped by kernel
router-10.9.0.11-192.168.60.11:/
$>tcpdump -i eth1 -n
```

```
root@3319119343db:/# export PS1="V-192.168.60.5:\w\n\$>"
V-192.168.60.5:/
$>ping 192.168.69.11 -c 2
PING 192.168.69.11 (192.168.69.11) 56(84) bytes of data.

--- 192.168.69.11 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1048ms

V-192.168.60.5:/
$>ping 192.168.60.6 -c 2
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
64 bytes from 192.168.60.6: icmp_seq=1 ttl=64 time=1.40 ms
64 bytes from 192.168.60.6: icmp_seq=2 ttl=64 time=0.138 ms

--- 192.168.60.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.138/0.768/1.399/0.630 ms
V-192.168.60.5:/
$>ping 192.168.60.6 -c 2
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
64 bytes from 192.168.60.6: icmp_seq=1 ttl=64 time=0.108 ms
64 bytes from 192.168.60.6: icmp_seq=2 ttl=64 time=0.120 ms

--- 192.168.60.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.108/0.114/0.120/0.006 ms
V-192.168.60.5:/
```

Task2 : Create and Configure TUN Interface

```python

import fcntl
import struct
import os
import time
from scapy.all import *

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'agrawal%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))

while True:
   time.sleep(10)

```

```
$>./tun.py &
[1] 32
U-10.9.0.5:/volumes
$>Interface Name: agrawal0

U-10.9.0.5:/volumes
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
5: agrawal0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group
default qlen 500
    link/none
31: eth0@if32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
U-10.9.0.5:/volumes
$>
```

```
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
5: agrawal0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group
default qlen 500
    link/none
31: eth0@if32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
U-10.9.0.5:/volumes
$>jobs
[1]+  Running                 ./tun.py &
U-10.9.0.5:/volumes
$>kill %1
```

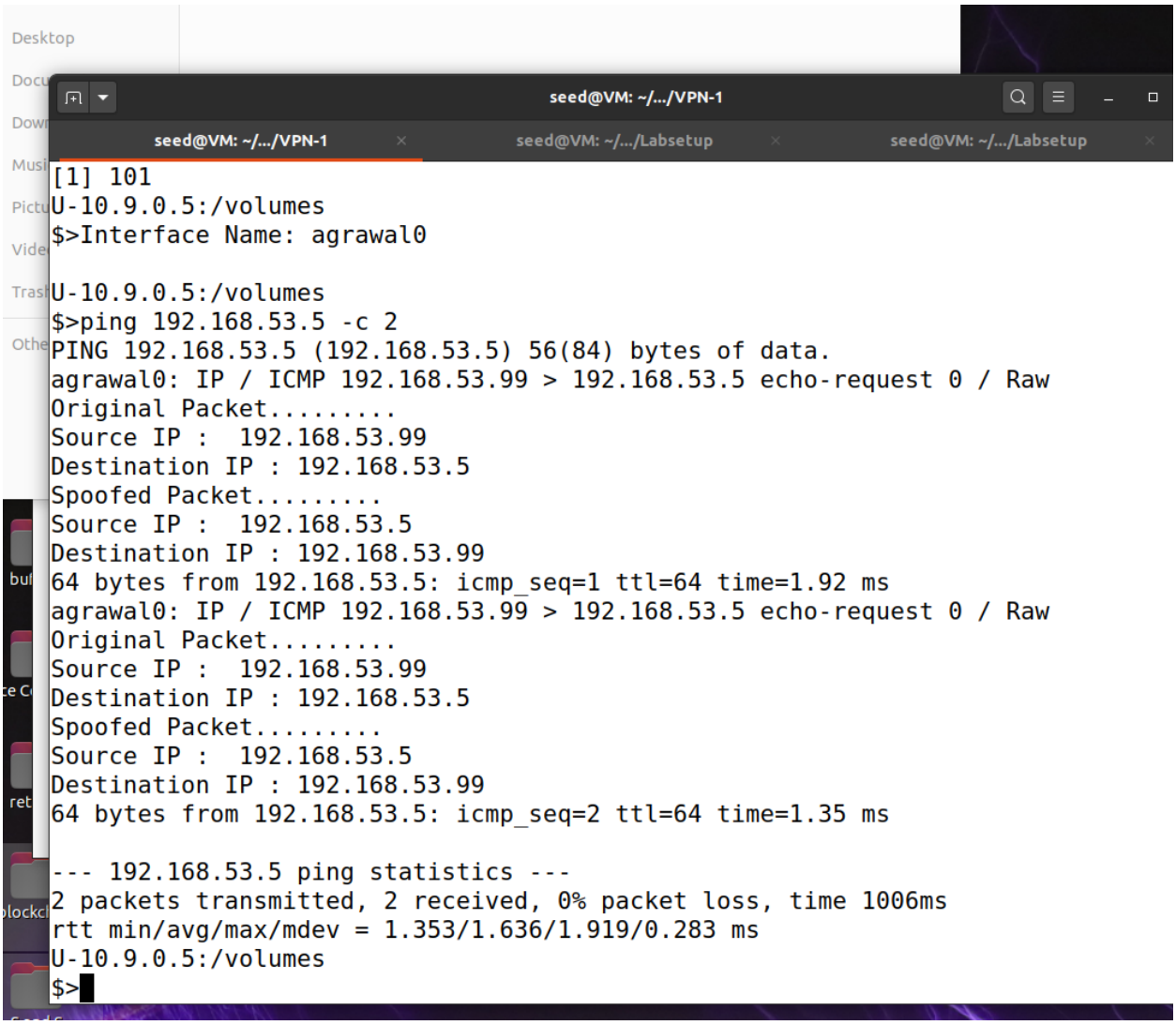
```python
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'agrawal%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

#config the tun inerface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))


while True:
   time.sleep(10)
```

```
U-10.9.0.5:/volumes
$>./tun.py &
[1] 37
U-10.9.0.5:/volumes
$>Interface Name: agrawal0

U-10.9.0.5:/volumes
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
6: agrawal0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel s
tate UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global agrawal0
       valid_lft forever preferred_lft forever
31: eth0@if32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
U-10.9.0.5:/volumes
$>
```

```python
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))

#config the tun inerface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))


while True:
  # Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
    ip = IP(packet)
    print("{}:".format(ifname), ip.summary())
```

```
U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
agrawal0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
agrawal0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1021ms

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1018ms

U-10.9.0.5:/volumes
$>
```

```
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

#config the tun inerface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))


while True:
  # Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
    pkt = IP(packet)
    print("{}:".format(ifname), ip.summary())

    # Send out a spoof packet using the tun interface
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("Original Packet.........")
        print("Source IP : ", pkt[IP].src)
        print("Destination IP :", pkt[IP].dst)

        # spoof an icmp echo reply packet
        # swap srcip and dstip
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data

        print("Spoofed Packet.........")
        print("Source IP : ", newpkt[IP].src)
        print("Destination IP :", newpkt[IP].dst)

        os.write(tun, bytes(newpkt))
```


seed@VM: ~/.../VPN-1
seed@VM: ~/.../VPN-1
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
[1] 101
U-10.9.0.5:/volumes
$>Interface Name: agrawal0
U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
agrawal0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Original Packet.........
Source IP : 192.168.53.99
Destination IP : 192.168.53.5
Spoofed Packet.........
Source IP : 192.168.53.5
Destination IP : 192.168.53.99
64 bytes from 192.168.53.5: icmp_seq=1 ttl=64 time=1.92 ms
agrawal0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Original Packet.........
Source IP : 192.168.53.99
Destination IP : 192.168.53.5
Spoofed Packet.........
Source IP : 192.168.53.5
Destination IP : 192.168.53.99
64 bytes from 192.168.53.5: icmp_seq=2 ttl=64 time=1.35 ms
--- 192.168.53.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 1.353/1.636/1.919/0.283 ms
U-10.9.0.5:/volumes
$>

```
U-10.9.0.5:/volumes
$>jobs
[1]+  Running                 ./tun.py &
U-10.9.0.5:/volumes
$>tcpdump -i last0 -n 2>/dev/null &
[2] 222
U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 1
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
agrawal0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Original Packet.........
Source IP :  192.168.53.99
Destination IP : 192.168.53.5
Spoofed Packet.........
Source IP :  192.168.53.5
Destination IP : 192.168.53.99
Destination IP : 192.168.53.99
16:20:01.083358 IP 192.168.53.99 > 192.168.53.5: ICMP echo request, id 82, seq 1
, length 64
16:20:01.084871 IP truncated-ip - 29435 bytes missing! 114.105.116.105 > 110.103
.32.97: ip-proto-102

--- 192.168.53.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

U-10.9.0.5:/volumes
$>
```

```python
  # Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
    pkt = IP(packet)
    print("{}:".format(ifname), pkt.summary())

    # Send out a spoof packet using the tun interface
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("Original Packet.........")
        print("Source IP : ", pkt[IP].src)
        print("Destination IP :", pkt[IP].dst)

        # spoof an icmp echo reply packet
        # swap srcip and dstip
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data

        print("Spoofed Packet.........")
        print("Source IP : ", newpkt[IP].src)
        print("Destination IP :", newpkt[IP].dst)
        arbdata = b'wcuyabuyawbeubvql for the arb data wabefuyawbgubvcuyabuyaw beubv
 qlf or the arb data hjbahjc bWBCK'
        os.write(tun, arbdata)
```

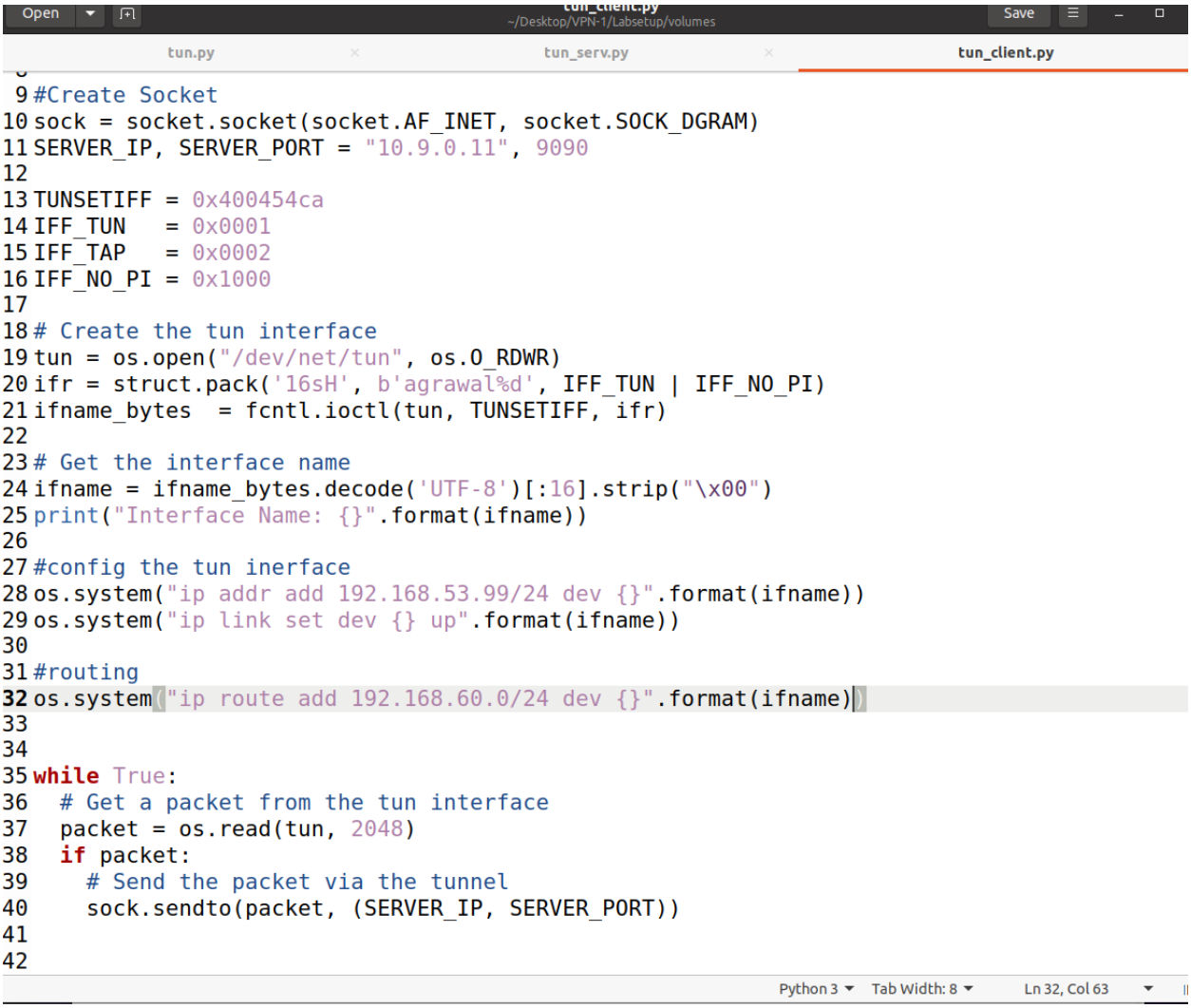
Task-3 Send the IP Packet to VPN Server Through a Tunnel

```python
import time
from scapy.all import *

#Create Socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP, SERVER_PORT = "10.9.0.11", 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'agrawal%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))

#config the tun inerface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))


while True:
  # Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
    # Send the packet via the tunnel
    sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

```python
#!/usr/bin/env python3

from scapy.all import *

IP_A = '0.0.0.0'
PORT = 9090

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

while True:
  data, (ip,port) = sock.recvfrom(2048)
  print("{}:{} --> {}:{}".format(ip,port, IP_A, PORT))
  pkt = IP(data)
  print("  Inside: {} --> {}".format(pkt.src, pkt.dst))
```

```
$>ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
router-10.9.0.11-192.168.60.11:/
$>cd volumes/
router-10.9.0.11-192.168.60.11:/volumes
$>ls
tun.py  tun_client.py  tun_serv.py
router-10.9.0.11-192.168.60.11:/volumes
$>chmod a+x tun_serv.py
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
10.9.0.5:37192 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:37192 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.5
```

```
U-10.9.0.5:/volumes
$>./tun_client.py &
[1] 227
U-10.9.0.5:/volumes
$>Interface Name: agrawal0

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1027ms

U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1003ms

U-10.9.0.5:/volumes
$>
```


Open
tun_client.py
~/Desktop/VPN-1/Labsetup/volumes
Save
tun.py
tun_serv.py
tun_client.py
#Create Socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP, SERVER_PORT = "10.9.0.11", 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'agrawal%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))
#config the tun inerface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
#routing
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
while True:
# Get a packet from the tun interface
packet = os.read(tun, 2048)
if packet:
# Send the packet via the tunnel
sock.sendto(packet, (SERVER_IP, SERVER_PORT))
Python 3
Tab Width: 8
Ln 32, Col 63

tun.py | tun_serv.py | tun_cli

```
12 IFF_TAP   = 0x0002
13 IFF_NO_PI = 0x1000
14
15 # Create the tun interface
16 tun = os.open("/dev/net/tun", os.O_RDWR)
17 ifr = struct.pack('16sH', b'agrawal%d', IFF_TUN | IFF_NO_PI)
18 ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
19
20 # Get the interface name
21 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
22 print("Interface Name: {}".format(ifname))
23
24 #config the tun inerface
25 os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
26 os.system("ip link set dev {} up".format(ifname))
27
28 #routing
29
30
31 #UDP server
32
33 IP_A = '0.0.0.0'
34 PORT = 9090
35
36 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
37 sock.bind((IP_A, PORT))
38
39 while True:
40   data, (ip,port) = sock.recvfrom(2048)
41   print("{}:{} --> {}:{}".format(ip,port, IP_A, PORT))
42   pkt = IP(data)
43   print("  Inside: {} --> {}".format(pkt.src, pkt.dst))
44
45   os.write(tun, bytes(pkt))
```

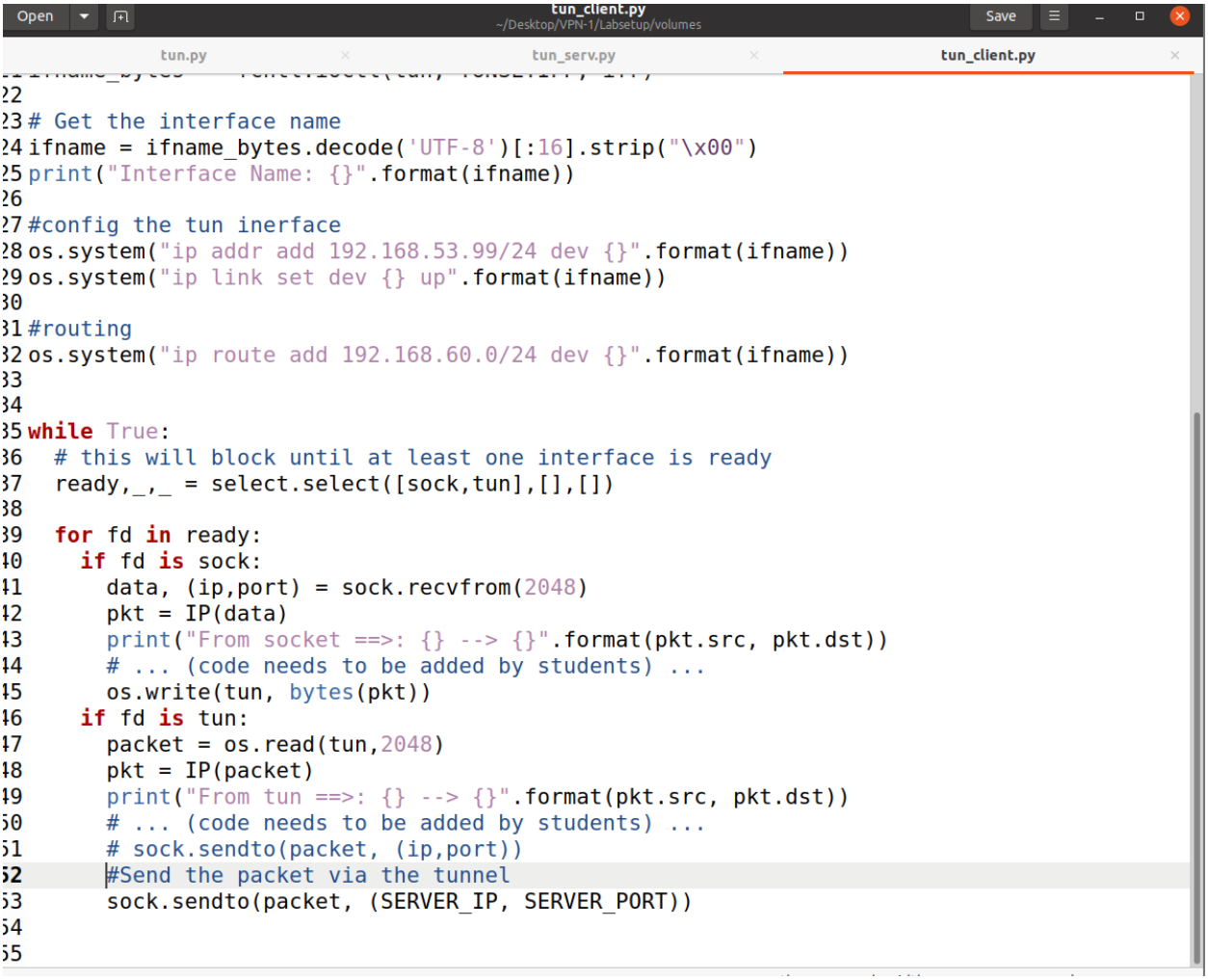
Python 3 | Tab Width: 8

```
U-10.9.0.5:/volumes
$>./tun_client.py &
[1] 241
U-10.9.0.5:/volumes
$>Interface Name: agrawal0

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1031ms

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1021ms

U-10.9.0.5:/volumes
$>
```

```
$>jobs
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
Interface Name: agrawal0

10.9.0.5:54921 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:54921 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

```
V-192.168.60.5:/
$>tcpdump -i eth0 -n 2>/dev/null
00:26:21.872620 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 252, seq
1, length 64
00:26:21.872647 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 252, seq 1,
 length 64
00:26:22.893891 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 252, seq
2, length 64
00:26:22.893936 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 252, seq 2,
 length 64
00:26:27.018636 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
00:26:27.018903 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
00:26:27.018936 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
00:26:27.018942 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

Task-4 Set Up the VPN Server


Open
tun_client.py
~/Desktop/VPN-1/Labsetup/volumes
Save
tun.py
tun_serv.py
tun_client.py
22
23 # Get the interface name
24 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
25 print("Interface Name: {}".format(ifname))
26
27 #config the tun inerface
28 os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
29 os.system("ip link set dev {} up".format(ifname))
30
31 #routing
32 os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
33
34
35 while True:
36   # this will block until at least one interface is ready
37   ready,_,_ = select.select([sock,tun],[],[])
38
39   for fd in ready:
40   if fd is sock:
41   data, (ip,port) = sock.recvfrom(2048)
42   pkt = IP(data)
43   print("From socket ==>: {} --> {}".format(pkt.src, pkt.dst))
44   # ... (code needs to be added by students) ...
45   os.write(tun, bytes(pkt))
46   if fd is tun:
47   packet = os.read(tun,2048)
48   pkt = IP(packet)
49   print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
50   # ... (code needs to be added by students) ...
51   # sock.sendto(packet, (ip,port))
52   #Send the packet via the tunnel
53   sock.sendto(packet, (SERVER_IP, SERVER_PORT))
54
55

tun.py | tun_serv.py | tun_client.py

```
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'agrawal%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))

#config the tun inerface
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

#routing


#UDP server

IP_A = '0.0.0.0'
PORT = 9090

ip,port = '10.9.0.5',12345

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:
      data, (ip,port) = sock.recvfrom(2048)
      pkt = IP(data)
      print("From socket ==>: {} --> {}".format(pkt.src, pkt.dst))
      # ... (code needs to be added by students) ...
      os.write(tun, bytes(pkt))
    if fd is tun:
      packet = os.read(tun,2048)
      pkt = IP(packet)
      print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
      # ... (code needs to be added by students) ...
      sock.sendto(packet, (ip,port))
```

```
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
Interface Name: agrawal0

10.9.0.5:54921 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:54921 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "./tun_serv.py", line 40, in <module>

KeyboardInterrupt

router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
Interface Name: agrawal0
From socket ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

```
U-10.9.0.5:/volumes
$>./tun_client.py &
[1] 253
U-10.9.0.5:/volumes
$>Interface Name: agrawal0

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=12.9 ms
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=4.37 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 4.368/8.643/12.919/4.275 ms
U-10.9.0.5:/volumes
$>
```

```
 length 64
00:26:27.018636 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
00:26:27.018903 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
00:26:27.018936 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
00:26:27.018942 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
00:49:53.996493 IP6 fe80::3cd0:b0ff:fe9f:cba7 > ff02::2: ICMP6, router solicitat
ion, length 16
00:51:09.687089 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 263, seq
1, length 64
00:51:09.687202 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 263, seq 1,
 length 64
00:51:10.687995 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 263, seq
2, length 64
00:51:10.688025 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 263, seq 2,
 length 64
00:51:14.891196 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
00:51:14.891456 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
00:51:14.891467 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
00:51:14.891482 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
00:52:05.067748 IP6 fe80::42:e2ff:fe68:f6df > ff02::2: ICMP6, router solicitatio
n, length 16
```

Task-5 Handling Traffic in Both Directions

tun.py | tun_serv.py | tun_client.py

```
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))

#config the tun inerface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

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


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:
      data, (ip,port) = sock.recvfrom(2048)
      print("From UDP {}:{} --> {}".format(ip,port,"10.9.0.5"))
      pkt = IP(data)
      print("From socket(IP) ==>: {} --> {}".format(pkt.src, pkt.dst))
      # ... (code needs to be added by students) ...
      os.write(tun, bytes(pkt))
    if fd is tun:
      packet = os.read(tun,2048)
      pkt = IP(packet)
      print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
      # ... (code needs to be added by students) ...
      # sock.sendto(packet, (ip,port))
      #Send the packet via the tunnel
      sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

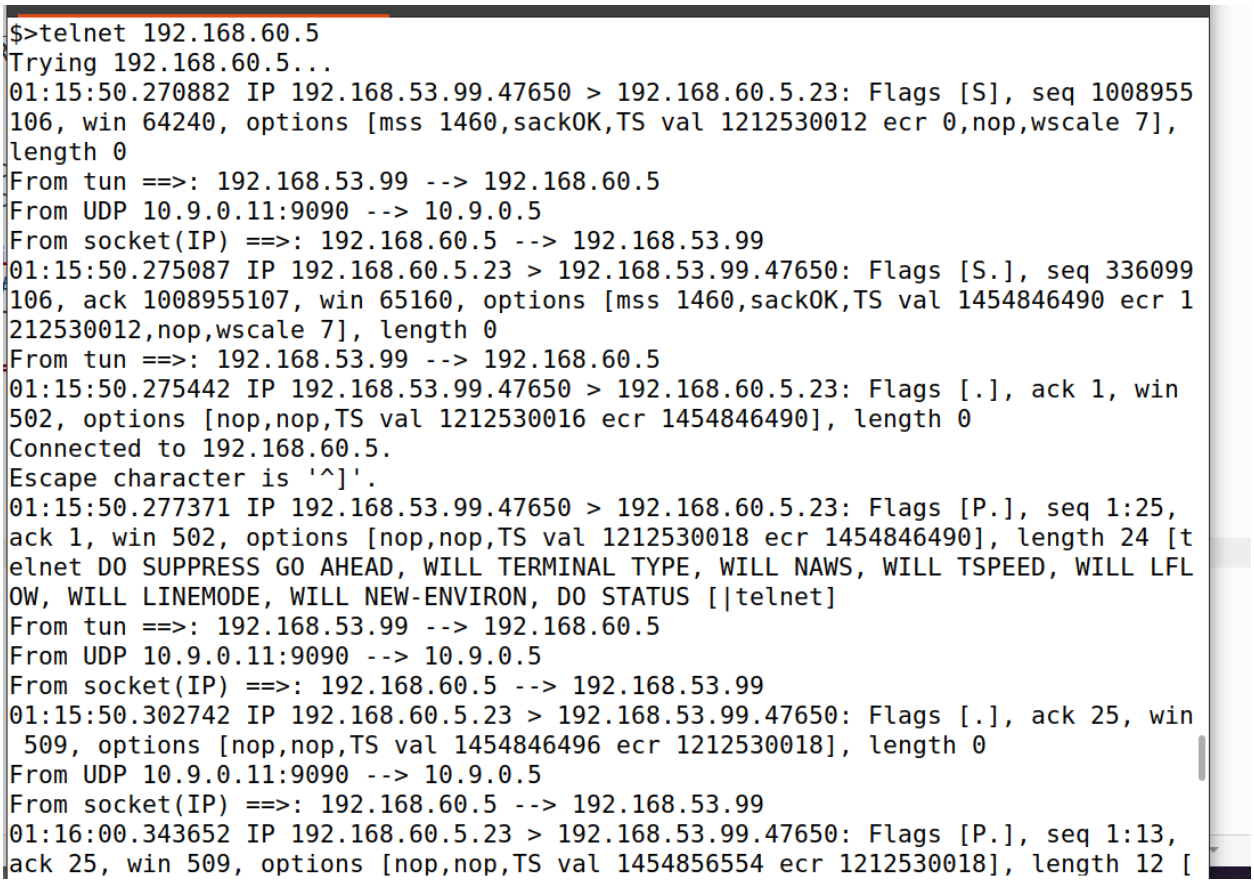
tun.py | tun_serv.py | tun_client.py

```
25 os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
26 os.system("ip link set dev {} up".format(ifname))
27
28 #routing
29
30
31 #UDP server
32
33 IP_A = '0.0.0.0'
34 PORT = 9090
35
36 ip,port = '10.9.0.5',12345
37
38 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
39 sock.bind((IP_A, PORT))
40
41 while True:
42   # this will block until at least one interface is ready
43   ready,_,_ = select.select([sock,tun],[],[])
44
45   for fd in ready:
46     if fd is sock:
47
48       data, (ip,port) = sock.recvfrom(2048)
49       print("From UDP {}:{} --> {}:{}".format(ip,port,IP_A,PORT))
50       pkt = IP(data)
51       print("From socket(IP) ==>: {} --> {}".format(pkt.src, pkt.dst))
52       # ... (code needs to be added by students) ...
53       os.write(tun, bytes(pkt))
54     if fd is tun:
55       packet = os.read(tun,2048)
56       pkt = IP(packet)
57       print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
58       # ... (code needs to be added by students) ...
59       sock.sendto(packet, (ip,port))
60
```

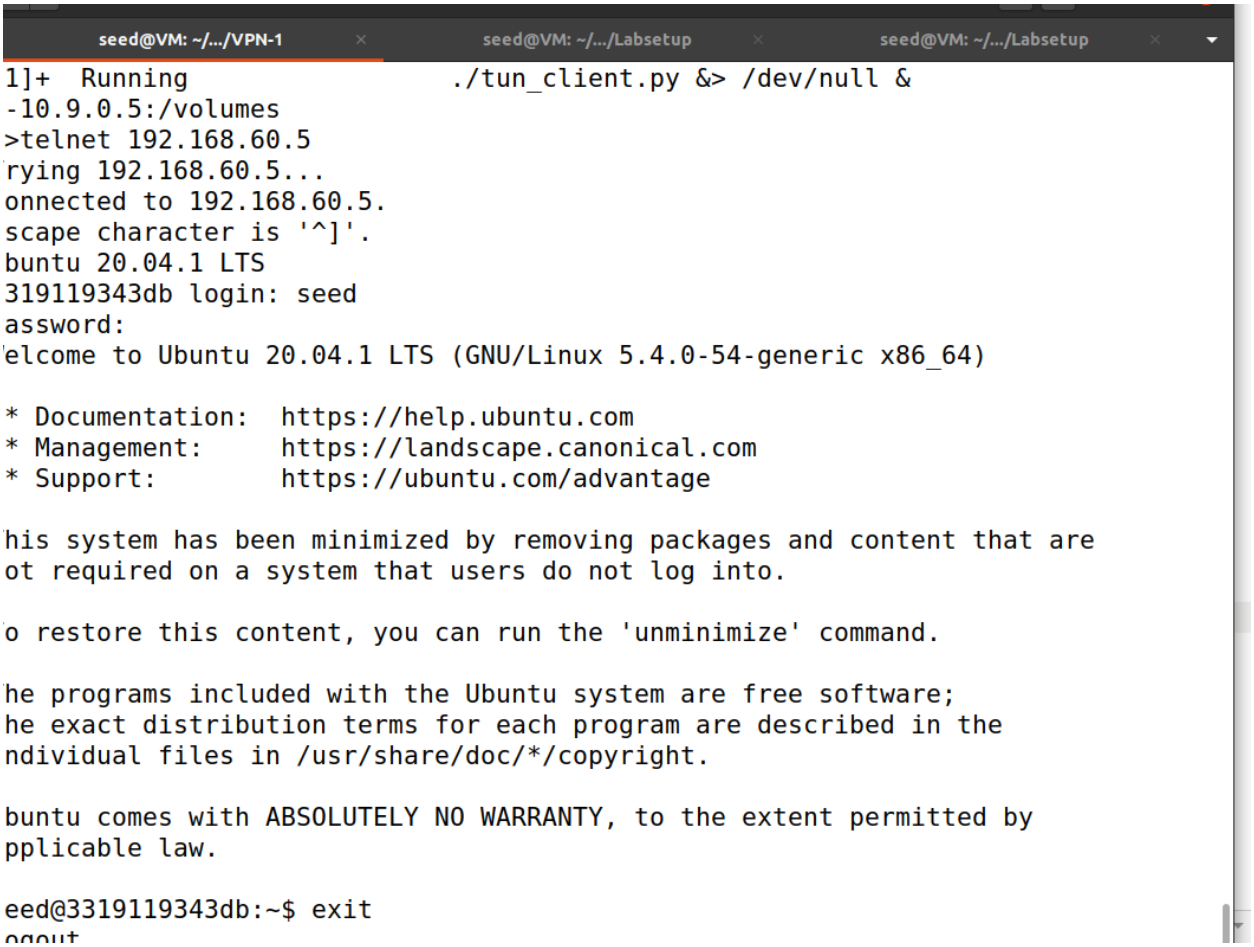
```
$>./tun_client.py &
[1] 264
U-10.9.0.5:/volumes
$>Interface Name: agrawal0

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=6.01 ms
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=6.09 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 6.005/6.046/6.087/0.041 ms
U-10.9.0.5:/volumes
$>


router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
Interface Name: agrawal0
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

```
00:51:10.688025 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 263, seq 2,
 length 64
00:51:14.891196 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
00:51:14.891456 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
00:51:14.891467 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
00:51:14.891482 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
00:52:05.067748 IP6 fe80::42:e2ff:fe68:f6df > ff02::2: ICMP6, router solicitatio
n, length 16
01:09:27.867431 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 274, seq
1, length 64
01:09:27.867527 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 274, seq 1,
 length 64
01:09:28.867512 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 274, seq
2, length 64
01:09:28.867539 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 274, seq 2,
 length 64
01:09:32.876831 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
01:09:32.877116 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
01:09:32.877126 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
01:09:32.877142 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

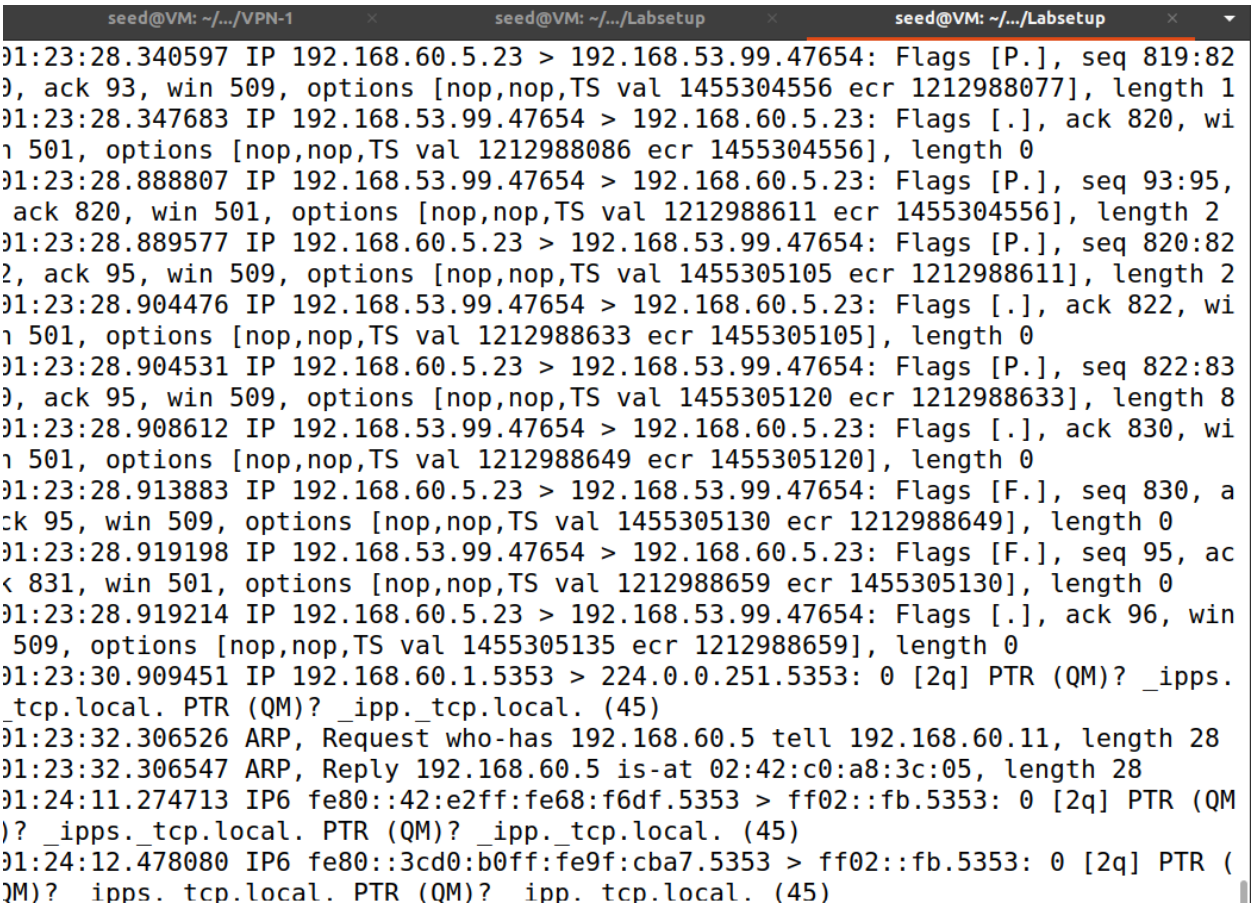
```
U-10.9.0.5:/volumes
$>tcpdump -i agrawal0 -n 2>/dev/null &
[2] 275
U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=6.19 ms
01:14:21.488110 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 276, seq
1, length 64
01:14:21.494261 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 276, seq 1,
 length 64
01:14:22.490909 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 276, seq
2, length 64
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
01:14:22.495864 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 276, seq 2,
 length 64
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=5.03 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 5.026/5.608/6.191/0.582 ms
U-10.9.0.5:/volumes
$>
```


$>telnet 192.168.60.5
Trying 192.168.60.5...
01:15:50.270882 IP 192.168.53.99.47650 > 192.168.60.5.23: Flags [S], seq 1008955
106, win 64240, options [mss 1460,sackOK,TS val 1212530012 ecr 0,nop,wscale 7],
length 0
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
01:15:50.275087 IP 192.168.60.5.23 > 192.168.53.99.47650: Flags [S.], seq 336099
106, ack 1008955107, win 65160, options [mss 1460,sackOK,TS val 1454846490 ecr 1
212530012,nop,wscale 7], length 0
From tun ==>: 192.168.53.99 --> 192.168.60.5
01:15:50.275442 IP 192.168.53.99.47650 > 192.168.60.5.23: Flags [.], ack 1, win
502, options [nop,nop,TS val 1212530016 ecr 1454846490], length 0
Connected to 192.168.60.5.
Escape character is '^]'.
01:15:50.277371 IP 192.168.53.99.47650 > 192.168.60.5.23: Flags [P.], seq 1:25,
ack 1, win 502, options [nop,nop,TS val 1212530018 ecr 1454846490], length 24 [t
elnet DO SUPPRESS GO AHEAD, WILL TERMINAL TYPE, WILL NAWS, WILL TSPEED, WILL LFL
OW, WILL LINEMODE, WILL NEW-ENVIRON, DO STATUS [|telnet]
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
01:15:50.302742 IP 192.168.60.5.23 > 192.168.53.99.47650: Flags [.], ack 25, win
509, options [nop,nop,TS val 1454846496 ecr 1212530018], length 0
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
01:16:00.343652 IP 192.168.60.5.23 > 192.168.53.99.47650: Flags [P.], seq 1:13,
ack 25, win 509, options [nop,nop,TS val 1454856554 ecr 1212530018], length 12 [

```
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
Interface Name: agrawal0
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:42753 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```


seed@VM: ~/.../VPN-1
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
1]+ Running   ./tun_client.py &> /dev/null &
-10.9.0.5:/volumes
>telnet 192.168.60.5
rying 192.168.60.5...
onnected to 192.168.60.5.
scape character is '^]'.
buntu 20.04.1 LTS
319119343db login: seed
assword:
elcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
his system has been minimized by removing packages and content that are
ot required on a system that users do not log into.
o restore this content, you can run the 'unminimize' command.
he programs included with the Ubuntu system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.
buntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
pplicable law.
eed@3319119343db:~$ exit
ogout

```
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```



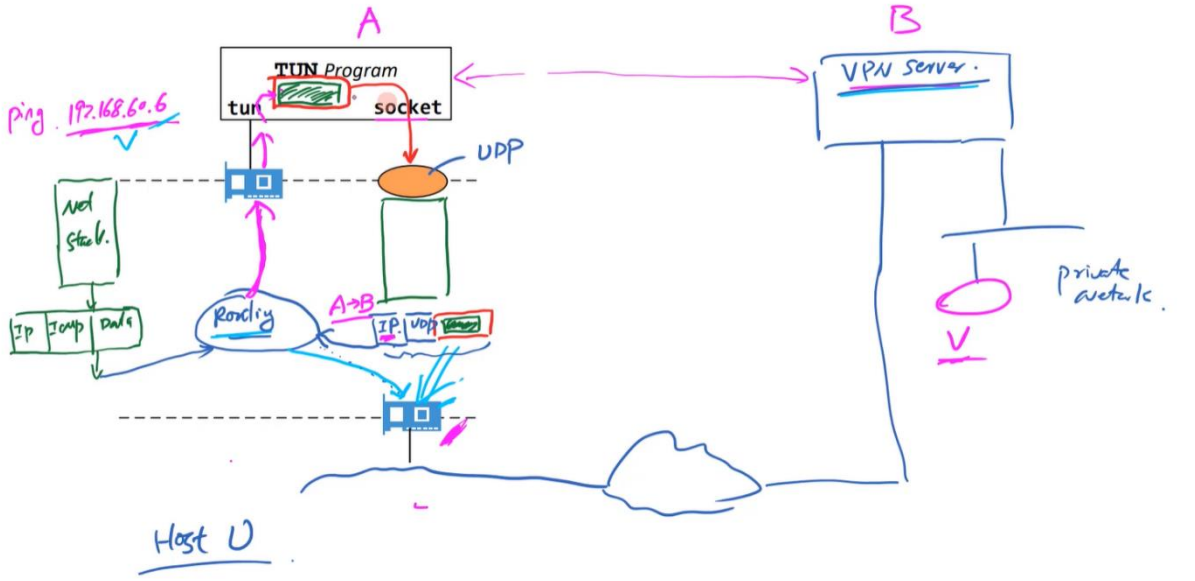
Task-6 : Tunnel-Breaking Experiment

```
root@d2227e2180d6:/# export PS1="U-10.9.0.5:\w\n\$>"
U-10.9.0.5:/
$>ping 10.9.0.11 -c 2
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.270 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.169 ms

--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.169/0.219/0.270/0.050 ms
U-10.9.0.5:/
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1012ms

U-10.9.0.5:/
$>telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
3319119343db login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Mar  7 01:22:53 UTC 2023 on pts/2
seed@3319119343db:~$ date
Tue Mar  7 01:26:23 UTC 2023
seed@3319119343db:~$ date
Tue Mar  7 01:28:40 UTC 2023
seed@3319119343db:~$
```

```
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_serv.py
Interface Name: agrawal0

From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:45402 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
```

Some important Takeaways:

Client: Put IP Packet Inside Tunnel



The Whole Picture