INTERNET PROTOCOL LAB ASSIGNMNET -12

Name: Siriparapu Sparshika

Roll No: CYS22006

```
[01/16/23]seed@VM:~/.../Labsetup$ dcbuild
VPN_Client uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Router uses an image, skipping
[01/16/23]seed@VM:~/.../Labsetup$ dcup
host-192.168.60.5 is up-to-date
client-10.9.0.5 is up-to-date
server-router is up-to-date
host-192.168.60.6 is up-to-date
Attaching to host-192.168.60.5, client-10.9.0.5, server-router, hos
t-192.168.60.6
host-192.168.60.5 |  * Starting internet superserver inetd        [
 OK ]
host-192.168.60.6 |  * Starting internet superserver inetd        [
 OK ]
```

```
[01/16/23]seed@VM:~/.../Labsetup$ dockps
9524588f8e3f  client-10.9.0.5
66226dbed01a  server-router
80ede6ecb53f  host-192.168.60.6
10a07fb19793  host-192.168.60.5
[01/16/23]seed@VM:~/.../Labsetup$
```

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh client-10.9.0.5
root@9524588f8e3f:/# echo $PS1
\u@\h:\w\$
root@9524588f8e3f:/# export PS1="U-10.9.0.5:\w\n\$>"
U-10.9.0.5:/
$>
```

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh server-router
root@66226dbed01a:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
 group default 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
13: eth1@if14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue 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
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue 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@66226dbed01a:/# export PS1="router-10.9.0.11-192.168.60.11:\w\
n\$>"
router-10.9.0.11-192.168.60.11:/
$>
```

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh host-192.168.60.5
root@10a07fb19793:/# export PS1="V-192.168.60.5:\w\n$>"
V-192.168.60.5:/
$>
```

Host U can communicate with VPN server

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh client-10.9.0.5
root@9524588f8e3f:/# echo $PS1
\u@\h:\w\$
root@9524588f8e3f:/# 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=0.183 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.095 ms

--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1018ms
rtt min/avg/max/mdev = 0.095/0.139/0.183/0.044 ms
U-10.9.0.5:/
$>
```

VPN server can communicate with Host- V

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh server-router
root@66226dbed01a:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
 group default 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
13: eth1@if14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue 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
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue 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@66226dbed01a:/# 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=0.116 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.135 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1043ms
rtt min/avg/max/mdev = 0.116/0.125/0.135/0.009 ms
router-10.9.0.11-192.168.60.11:/
$>
```

Host U should not be able to communicate with Host V

Open docksh client and ping

```
$>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 1009ms

U-10.9.0.5:/
$>
```

Run TCP dump on the router and sniff the traffic on each network and show that You can capture the packets.

Eth0:

After performing TCP dump in in server router we should ping 10.9.0.11 at client

```
$>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
13:18:50.616134 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 27,
seq 1, length 64
13:18:50.616159 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 27, se
q 1, length 64
13:18:51.631780 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 27,
seq 2, length 64
13:18:51.631808 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 27, se
q 2, length 64
13:18:55.759831 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, lengt
h 28
13:18:55.759994 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, lengt
h 28
13:18:55.760000 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, lengt
h 28
13:18:55.760002 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:/
```

```
$> 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=0.114 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.127 ms

--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1016ms
rtt min/avg/max/mdev = 0.114/0.120/0.127/0.006 ms
U-10.9.0.5:/
$>
```

Now to check for proper private network we should change the eth0 – eth1 “ value

```
$>tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol
 decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144
 bytes
13:23:30.160511 IP 192.168.60.5 > 192.168.60.11: ICMP echo request,
 id 36, seq 1, length 64
13:23:30.160537 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, i
d 36, seq 1, length 64
13:23:31.183619 IP 192.168.60.5 > 192.168.60.11: ICMP echo request,
 id 36, seq 2, length 64
13:23:31.183663 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, i
d 36, seq 2, length 64
13:23:35.311536 ARP, Request who-has 192.168.60.5 tell 192.168.60.1
1, length 28
13:23:35.311617 ARP, Request who-has 192.168.60.11 tell 192.168.60.
5, length 28
13:23:35.311622 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, l
ength 28
13:23:35.311624 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, le
ngth 28
```

(Here it will ask for physical address)

Here , after tcpdump we should ping 192.168.60.11 in the docsh - host

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh host-192.168.60.5
root@10a07fb19793:/# export PS1="V-192.168.60.5:\w\n$>"
V-192.168.60.5:/
$>pinf 192.168.60.11 -c 2
bash: pinf: command not found
V-192.168.60.5:/
$>ping  192.168.60.11 -c 2
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.129 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.119 ms

--- 192.168.60.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1023ms
rtt min/avg/max/mdev = 0.119/0.124/0.129/0.005 ms
V-192.168.60.5:/
$>
```

So again, if we ping another .6 private network we don't have any packets captured because from .5 to .6 . From host v to another private host –6

```
$>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=0.099 ms
64 bytes from 192.168.60.6: icmp_seq=2 ttl=64 time=0.115 ms

--- 192.168.60.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1028ms
rtt min/avg/max/mdev = 0.099/0.107/0.115/0.008 ms
V-192.168.60.5:/
$>
```

TASK-2

2a.

```
$>cd volumes/
U-10.9.0.5:/volumes
$>ls
tun.py
U-10.9.0.5:/volumes
$>chmod a+x tun.py
U-10.9.0.5:/volumes
$>./tun.py
Interface Name: tun0
```

We stop running the program

```
$>./tun.py
Interface Name: tun0
^Z
[1]+  Stopped                 ./tun.py
U-10.9.0.5:/volumes
$>jobs
[1]+  Stopped                 ./tun.py
U-10.9.0.5:/volumes
$>kill %1

[1]+  Stopped                 ./tun.py
U-10.9.0.5:/volumes
$>./tun.py &
[2] 38
[1]   Terminated              ./tun.py
U-10.9.0.5:/volumes
```

Here we & to make the program run in the background.

```
$>./tun.py &
[2] 38
[1]   Terminated              ./tun.py
U-10.9.0.5:/volumes
$>Interface Name: tun0

U-10.9.0.5:/volumes
$>jobs
[2]+  Running                 ./tun.py &
U-10.9.0.5:/volumes
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N group default 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
4: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state D
OWN group default qlen 500
    link/none
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsi
d 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
$>
```

And we don't find any ip address , and we need to terminate the process.

```
$>jobs
[2]+  Running                 ./tun.py &
U-10.9.0.5:/volumes
$>kill %2
U-10.9.0.5:/volumes
$>jobs
[2]+  Terminated              ./tun.py
U-10.9.0.5:/volumes
$>
```

Change name tun to the last name of yours

We should update the name "tun to last" in the code of tun.py

```
# 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))
```

So, after updating we can see that the interface name got changed.

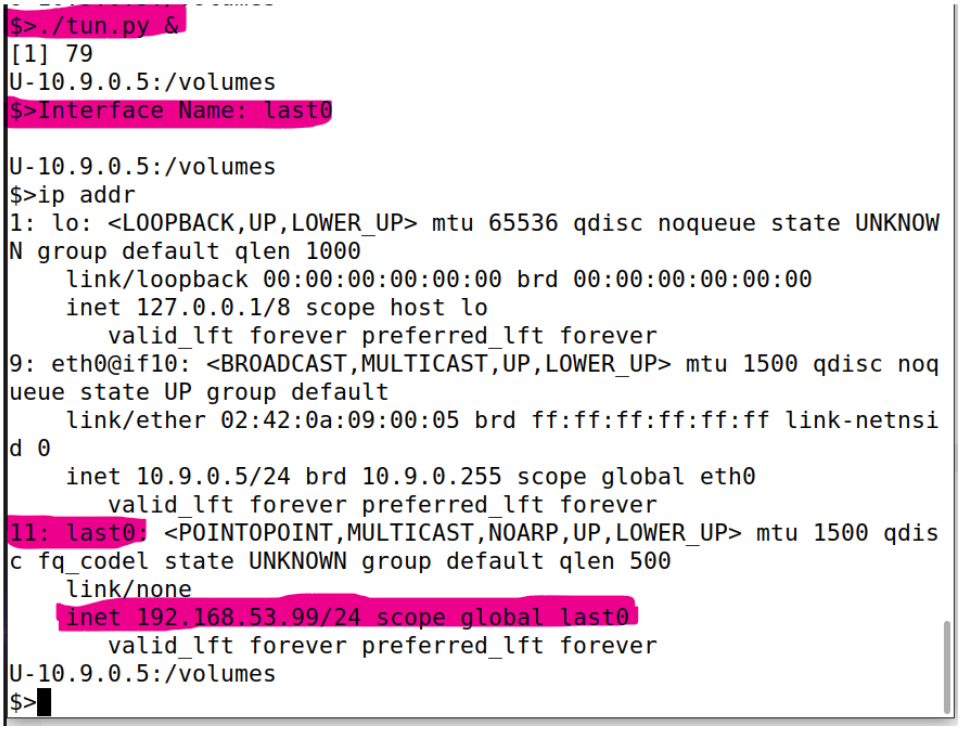
```
$>./tun.py &
[1] 43
U-10.9.0.5:/volumes
$>Interface Name: last0

U-10.9.0.5:/volumes
$>
```

We can see that the name in the ip addr also got updated .

```
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N group default 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: last0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state
DOWN group default qlen 500
    link/none
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsi
d 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
$>
```

Task 2b : Set up the tun interface.

After adding the 2 lines of code and run the code and when we check in the ip addrr we can see that the last0 is UP



And when we compare the result with previous result we can see that ip addrr is up in the second execution while its not up in the first last0.

Task2c: Read From the TUN interface.

Update the while loop.

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

Ping a host of same network, and we didn't received any packets because .5 doesn't exists.

```
$>./tun.py &
[1] 92
U-10.9.0.5:/volumes
$>Interface Name: last0

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.
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
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 1013ms

U-10.9.0.5:/volumes
```

print("{}:".format(ifname) after we update this printf and run the ip addr again we can see that

```
$>./tun.py &
[1] 109
U-10.9.0.5:/volumes
$>Interface Name: last0

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.
last0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
last0: 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 1023ms

U-10.9.0.5:/volumes
$>
```

Now we will ping host V

```
$>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 1006ms

U-10.9.0.5:/volumes
$>
```

The packers are transmitted but not sent to the tun interface because we have not set up the router.

Task2.d: Write to tun interface

#Send out a spoof packet using the tun interface

After getting a packet from the TUN interface, if this packet is an ICMP echo request packet, constructa corresponding echo reply packet and write it to the TUN interface. Please provide evidence to showthat the code works as expected

```
while True:
   #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
     #sniff and print out icmp echo request packet
   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))
```

After updating the code to only icmp packets and then pinning we get the following output

```
$>./tun.py &
[1] 154
U-10.9.0.5:/volumes
$>Interface Name: last0

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.
last0: 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=1.73 ms
last0: 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=1.82 ms

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.725/1.773/1.821/0.048 ms
U-10.9.0.5:/volumes
$>
```

Instead of writing an IP packet to the interface, write some arbitrary data to the interface, and report your observation.

```
$>tcpdump -i eth0 -n 2>/dev/null &
[2] 194
U-10.9.0.5:/volumes
$>jobs
[1]-  Running                 ./tun.py &
[2]+  Running                 tcpdump -i eth0 -n 2> /dev/null &
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.
last0: 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

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

U-10.9.0.5:/volumes
$>
```

Output by the tcpdump (pink highlight) its an optional data we wrote for the tun interface.

114 --- source and 110 --- destination ip

```
$>kill %2
U-10.9.0.5:/volumes

$>jobs
[1]-  Running                 ./tun.py &
[2]+  Done                    tcpdump -i eth0 -n 2> /dev/null
U-10.9.0.5:/volumes
$>tcpdump -i last0 -n 2>/dev/null &
[2] 196
U-10.9.0.5:/volumes
$>jobs
[1]-  Running                 ./tun.py &
[2]+  Running                 tcpdump -i last0 -n 2> /dev/null &
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.
last0: 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
16:33:48.416415 IP 192.168.53.99 > 192.168.53.5: ICMP echo request,
 id 197, seq 1, length 64
16:33:48.417308 IP truncated-ip - 29433 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
$>
```

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

Now we will write both server and client codes and ping

Server is pinged on docksh server and same client on client dock

```
$>cd volumes/
router-10.9.0.11-192.168.60.11:/volumes
$>ls
tun.py  tun_client.py
router-10.9.0.11-192.168.60.11:/volumes
$>ls
tun.py  tun_client.py  tun_server.py
router-10.9.0.11-192.168.60.11:/volumes
$>chmod a+x tun_server.py
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_server.py
```

```
$>./tun_client.py &
[1] 201
U-10.9.0.5:/volumes
$>Interface Name: last0

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 1006ms

U-10.9.0.5:/volumes
$>
```

So now we cam see that there is no response on the server side . So we now change ip addr on the client and when we run we can see that the server has captured some information.

Now the tun worked because Ip packet is put inside the payload and received by the server.

```
$>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
```

```
router-10.9.0.11-192.168.60.11:/volumes
$>./tun_server.py
10.9.0.5:46193 -> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:46193 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.5
```

… .53 is a virutal address used by vpn server

And vpn clinet packet will be reverted to the last0

```
$>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 1006ms

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
$>ip route add 192.168.60.0/24 dev last0
U-10.9.0.5:/volumes
$>ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev last0 proto kernel scope link src 192.168.53.99

192.168.60.0/24 dev last0 scope link
U-10.9.0.5:/volumes
$>
```

And again when we ping on client side 192.168.60.5 nothing will be printed instead on the server side we see the o/p

```
  Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:46193 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:46193 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

This is inside a udp packer and these are the packets went through the tunnel from client to server

TASK-4 : Setup the VPN server.

We proved that packets will got through the tunnel through the host

And host will send back.

Here first we will run the command at the host and then run in server and then ping in the client

@client

```
$>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 1017ms

U-10.9.0.5:/volumes
$>
```

@server

```
$>./tun_server.py
Interface Name: last0
10.9.0.5:37493 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:37493 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

@ Host

```
$>tcpdump -i eth0 -n 2>/dev/null
17:28:48.261179 IP6 fe80::42:83ff:fe2c:eca1.5353 > ff02::fb.5353: 0
 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
17:28:49.072816 IP6 fe80::ac:25ff:fe6a:b862.5353 > ff02::fb.5353: 0
 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
17:32:13.484271 IP 192.168.53.99 > 192.168.60.5: ICMP echo request,
 id 226, seq 1, length 64
17:32:13.484300 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, i
d 226, seq 1, length 64
17:32:14.501314 IP 192.168.53.99 > 192.168.60.5: ICMP echo request,
 id 226, seq 2, length 64
17:32:14.501344 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, i
d 226, seq 2, length 64
17:32:18.692284 ARP, Request who-has 192.168.60.11 tell 192.168.60.
5, length 28
17:32:18.692392 ARP, Request who-has 192.168.60.5 tell 192.168.60.1
1, length 28
17:32:18.692422 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, le
ngth 28
17:32:18.692425 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, l
ength 28
```

TASK 5 : Handling traffic in both direction

On client we can see

```
$>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=2.35 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=3.02 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.345/2.683/3.021/0.338 ms
U-10.9.0.5:/volumes
$>
```

On server we can see

```
$>./tun_server.py
Interface Name: last0
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
```

And on Host we can see

```
17:47:21.976742 IP 192.168.53.99 > 192.168.60.5: ICMP echo request,
 id 237, seq 1, length 64
17:47:21.976767 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, i
d 237, seq 1, length 64
17:47:22.981201 IP 192.168.53.99 > 192.168.60.5: ICMP echo request,
 id 237, seq 2, length 64
17:47:22.981233 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, i
d 237, seq 2, length 64
17:47:26.980666 ARP, Request who-has 192.168.60.11 tell 192.168.60.
5, length 28
17:47:26.980825 ARP, Request who-has 192.168.60.5 tell 192.168.60.1
1, length 28
17:47:26.980902 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, le
ngth 28
17:47:26.980909 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, l
ength 28
```