Lab 6

Student ID: 1005288

Task 1

Host U (10.9.0.5) can communicate with VPN Server (10.9.0.11), Host U was able to ping the VPN server

```
[11/13/23]seed@VM:~/.../Labsetup$ dockps
2fa830be2a88  host-192.168.60.6
1d83feb72a1d  server-router
f76b447d3196  host-192.168.60.5
7b4a0b92ed9a  client-10.9.0.5
[11/13/23]seed@VM:~/.../Labsetup$ docksh 7b
root@7b4a0b92ed9a:/# ping 10.9.0.11
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=0.095 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.068 ms
^C
--- 10.9.0.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.068/0.079/0.095/0.011 ms
```

VPN Server (192.168.60.11) can communicate with Host V (192.168.60.5), VPN server was able to ping Host V

```
[11/13/23]seed@VM:~/.../Labsetup$ dockps
2fa830be2a88  host-192.168.60.6
1d83feb72a1d  server-router
f76b447d3196  host-192.168.60.5
7b4a0b92ed9a  client-10.9.0.5
[11/13/23]seed@VM:~/.../Labsetup$ docksh 1d
root@1d83feb72a1d:/# ping 192.168.60.5
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=0.066 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.071 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2044ms
rtt min/avg/max/mdev = 0.066/0.072/0.081/0.006 ms
```

Host U should not be able to communicate with Host V – no ping replies

```
[11/13/23]seed@VM:~/.../Labsetup$ dockps
2fa830be2a88  host-192.168.60.6
1d83feb72a1d  server-router
f76b447d3196  host-192.168.60.5
7b4a0b92ed9a  client-10.9.0.5
[11/13/23]seed@VM:~/.../Labsetup$ docksh f7
root@f76b447d3196:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1011ms
```

Host V was also unable ping host U

```
[11/13/23]seed@VM:~/.../Labsetup$ dockps
2fa830be2a88  host-192.168.60.6
1d83feb72a1d  server-router
f76b447d3196  host-192.168.60.5
7b4a0b92ed9a  client-10.9.0.5
[11/13/23]seed@VM:~/.../Labsetup$ docksh 7b
root@7b4a0b92ed9a:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
44 packets transmitted, 0 received, 100% packet loss, time 44092ms
```

Run ifconfig to see which subnet each interface is in

```
root@1d83feb72a1d:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.11  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:0b  txqueuelen 0  (Ethernet)
        RX packets 191  bytes 22437 (22.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 90  bytes 7234 (7.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.11  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:0b  txqueuelen 0  (Ethernet)
        RX packets 180  bytes 20690 (20.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34  bytes 2408 (2.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Run tcpdump -i eth0 and tcpdump -i eth1 separately on the router, host U and host V ping the server-router. Both results showed that host U and host V were able to communicate with the server-router.

```
root@7b4a0b92ed9a:/# echo "Network 10.9.0.0/24 eth0"
Network 10.9.0.0/24 eth0
root@7b4a0b92ed9a:/# ping 10.9.0.11 -c 1
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=0.101 ms

--- 10.9.0.11 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.101/0.101/0.101/0.000 ms


root@f76b447d3196:/# echo "Network 192.168.60.0/24 eth1"
Network 192.168.60.0/24 eth1
root@f76b447d3196:/# ping 192.168.60.11 -c 1
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=0.088 ms

--- 192.168.60.11 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.088/0.088/0.088/0.000 ms
```

The router was able to sniff packets within the interface as shown by the number of packets captured

```
[11/13/23]seed@VM:~/.../Labsetup$ dockps
2fa830be2a88  host-192.168.60.6
1d83feb72a1d  server-router
f76b447d3196  host-192.168.60.5
7b4a0b92ed9a  client-10.9.0.5
[11/13/23]seed@VM:~/.../Labsetup$ docksh 1d
root@1d83feb72a1d:/# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:09:31.203343 IP client-10.9.0.5.net-10.9.0.0 > 1d83feb72a1d: ICMP echo request, id 100, seq 1, length
 64
15:09:31.203367 IP 1d83feb72a1d > client-10.9.0.5.net-10.9.0.0: ICMP echo reply, id 100, seq 1, length 6
4
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@1d83feb72a1d:/# tcpdump -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:10:01.549349 IP host-192.168.60.5.net-192.168.60.0 > 1d83feb72a1d: ICMP echo request, id 74, seq 1, l
ength 64
15:10:01.549371 IP 1d83feb72a1d > host-192.168.60.5.net-192.168.60.0: ICMP echo reply, id 74, seq 1, len
gth 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

Task 2

Part A

We run tun.py in server-router container, then open another terminal to run ip address, check that the interface name had changed from tun0 to Lim0

```
[11/13/23]seed@VM:~/.../task2$ dockps
2fa830be2a88  host-192.168.60.6
1d83feb72a1d  server-router
f76b447d3196  host-192.168.60.5
7b4a0b92ed9a  client-10.9.0.5
[11/13/23]seed@VM:~/.../task2$ docksh 1d
root@1d83feb72a1d:/# cd volumes/
root@1d83feb72a1d:/volumes# ls
tun.py
root@1d83feb72a1d:/volumes# python3 tun.py
Interface Name: tun0
^CTraceback (most recent call last):
  File "tun.py", line 24, in <module>
    time.sleep(10)
KeyboardInterrupt

root@1d83feb72a1d:/volumes# nano tun.py
root@1d83feb72a1d:/volumes# python3 tun.py
Interface Name: Lim0
^CTraceback (most recent call last):
  File "tun.py", line 24, in <module>
    time.sleep(10)
KeyboardInterrupt
```

Running tun.py with interface name “tun”

```
root@1d83feb72a1d:/# ip address
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
7: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
12: eth1@if13: <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
```

After changing the interface name to “Lim”

```
root@1d83feb72a1d:/# ip address
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
8: Lim0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
12: eth1@if13: <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
```

States of both interfaces are ***DOWN*** at this point

Part B

After adding the lines
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
into the python script (see configured_tun.py), we then run configured_tun.py

```
root@c55ae5a533c7:/volumes# nano configured_tun.py
root@c55ae5a533c7:/volumes# nano configured_tun.py
root@c55ae5a533c7:/volumes# python3 configured_tun.py
Interface Name: Lim0
```

The interface state became ***UNKNOWN,*** hence the state is not ***DOWN*** anymore and is open for connection, with its IP address 192.168.53.99

```
[11/13/23]seed@VM:~/.../B$ docksh c5
root@c55ae5a533c7:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
000
    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: Lim0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN
 group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global Lim0
       valid_lft forever preferred_lft forever
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group d
efault
    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
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group d
efault
    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
```

Part C

The configuration code to get the interface up has to be included in the script, alongside the new while loop. After running the script on host U, the tunnel was able to capture packets.

When host U pings a host in the subnet 192.168.53.0/24 (192.168.53.17), there was no ping reply as the tunnel was not configured to reply host U yet.

```
root@8a054057ad45:/# ping 192.168.53.17 -c 1
PING 192.168.53.17 (192.168.53.17) 56(84) bytes of data.
^C
--- 192.168.53.17 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Since 192.168.53.17 is on the same subnet as the server-router, and host U is on another subnet with the server router (10.9.0.0/24), it was able to capture ping packets coming from host U.

```
[11/13/23]seed@VM:~/.../C$ dockps
c55ae5a533c7  server-router
a8b3b8b3d11f  host-192.168.60.5
8a054057ad45  client-10.9.0.5
f5f876a84a34  host-192.168.60.6
[11/13/23]seed@VM:~/.../C$ docksh 8a
root@8a054057ad45:/# cd volumes/
root@8a054057ad45:/volumes# python3 task2c.py
Interface Name: Lim0
READING PKT.....
IP / ICMP 192.168.53.99 > 192.168.53.17 echo-request 0 / Raw
READING PKT.....
```

However, packets were not captured when host U pings host V (192.168.60.5)in the subnet 192.168.60.0/24, this is because the tunnel between host U and host V was not set up and host V was unreachable.

```
root@8a054057ad45:/# ping 192.168.60.5 -c 1
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```


```
[11/13/23]seed@VM:~/.../C$ dockps
c55ae5a533c7  server-router
a8b3b8b3d11f  host-192.168.60.5
8a054057ad45  client-10.9.0.5
f5f876a84a34  host-192.168.60.6
[11/13/23]seed@VM:~/.../C$ docksh 8a
root@8a054057ad45:/# cd volumes/
root@8a054057ad45:/volumes# python3 task2c.py
Interface Name: Lim0
READING PKT.....
```


Part D

We can check if ICMP is used by looking for "ICMP" and "echo-request" as a substring in ip.summary()

If these are not in the IP packet summary, then the condition will not be met and a reply will not be constructed.

Building on the knowledge from Lab1 where we spoof icmp packets, we swap the src and dst ip addresses, and a new ICMP packet is constructed with the ICMP type set as reply (type=0), seqnum and id remains unchanged.

After running the new script, host U got a ping reply from the subnet 192.168.53.0/24 after a ping to 192.168.53.17, and was unable to telnet into 192.168.53.17 since telnet uses TCP

```
[11/13/23]seed@VM:~/.../D$ dockps
c55ae5a533c7  server-router
a8b3b8b3d11f  host-192.168.60.5
8a054057ad45  client-10.9.0.5
f5f876a84a34  host-192.168.60.6
[11/13/23]seed@VM:~/.../D$ docksh 8a
root@8a054057ad45:/# ping 192.168.53.17 -c 1
PING 192.168.53.17 (192.168.53.17) 56(84) bytes of data.
64 bytes from 192.168.53.17: icmp_seq=1 ttl=64 time=1.51 ms

--- 192.168.53.17 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.514/1.514/1.514/0.000 ms
root@8a054057ad45:/# telnet 192.168.53.17
Trying 192.168.53.17...
^C
```

As shown, the tunnel was able to detect that an echo-request was sent and constructed an icmp reply, but when it received a TCP packet for telnet, it did not construct any reply. Which resulted in the telnet connection not being established.

```
[11/13/23]seed@VM:~/.../D$ dockps
c55ae5a533c7  server-router
a8b3b8b3d11f  host-192.168.60.5
8a054057ad45  client-10.9.0.5
f5f876a84a34  host-192.168.60.6
[11/13/23]seed@VM:~/.../D$ docksh 8a
root@8a054057ad45:/# cd volumes/
root@8a054057ad45:/volumes# python3 task2d.py
Interface Name: Lim0
READING PKT.....
IP / ICMP 192.168.53.99 > 192.168.53.17 echo-request 0 / Raw
constructing ICMP reply...
READING PKT.....
IP / TCP 192.168.53.99:40158 > 192.168.53.17:telnet S
READING PKT.....
IP / TCP 192.168.53.99:40158 > 192.168.53.17:telnet S
READING PKT.....
IP / TCP 192.168.53.99:40158 > 192.168.53.17:telnet S
READING PKT.....
```

By changing the code to send the arbitrary data instead of the constructed reply packet, host U was unable to get a ping reply like before.

```
root@8a054057ad45:/# ping 192.168.53.17 -c 1
PING 192.168.53.17 (192.168.53.17) 56(84) bytes of data.
^C^C
--- 192.168.53.17 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

As we can see, an icmp reply was constructed, but the packet was not sent. Which was why host U did not get a ping reply.

```
root@8a054057ad45:/volumes# python3 task2d.py
Interface Name: Lim0
READING PKT.....
IP / ICMP 192.168.53.99 > 192.168.53.17 echo-request 0 / Raw
constructing ICMP reply...
READING PKT.....
```

Task 3

The tunnel interface configuration code should be included, Host U will connect with the VPN server (10.9.0.11) using UDP.

This part shows that host U is able to communicate with a host in the same subnet as the VPN server. In tun_client.py, we set the UDP socket to connect to 10.9.0.11 as it is the VPN server IP, which is the VPN server that was set up in the previous tasks

Host U pings a host in the same subnet as the VPN server and host V.

```
[11/14/23]seed@VM:~/.../task3$ dockps
b2a082948ce4  server-router
ba2fcd3bfe73  client-10.9.0.5
a0b2e37d8050  host-192.168.60.6
70c3c9ccba04  host-192.168.60.5
[11/14/23]seed@VM:~/.../task3$ docksh ba
root@ba2fcd3bfe73:/# ping 192.168.53.17
PING 192.168.53.17 (192.168.53.17) 56(84) bytes of data.
^C
--- 192.168.53.17 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2037ms

root@ba2fcd3bfe73:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2054ms
```

The VPN server prints out the ping request, with src 10.9.0.5, to the virtual IP 0.0.0.0, with the encapsulating IP addresses of the VPN server (192.168.53.99) and dst 192.168.53.17, but not when host U pings host V.

```
root@b2a082948ce4:/volumes# python3 tun_server.py
10.9.0.5:52860 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.17
10.9.0.5:52860 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.17
10.9.0.5:52860 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.17
```

As seen from the results, the VPN did not capture packets when host U pings host V before adding the routing information. This is because the VPN server was not unable to route to host V.

We will run ip route add 192.168.60.0/24 dev Lim0 via 192.168.53.99, after which, the routing table had a new entry to route to 192.168.60.0/24 through the server-router IP (192.168.53.99)

```
root@aabbc9e02a55:/# echo "HOST U"
HOST U
root@aabbc9e02a55:/# 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 Lim0 proto kernel scope link src 192.168.53.99
root@aabbc9e02a55:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4095ms

root@aabbc9e02a55:/# ip route add 192.168.60.0/24 dev Lim0 via 192.168.53.99
root@aabbc9e02a55:/# 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 Lim0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 via 192.168.53.99 dev Lim0
root@aabbc9e02a55:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5118ms
```

```
root@4e9d65e0fed1:/# echo "server-router"
server-router
root@4e9d65e0fed1:/# cd volumes/
root@4e9d65e0fed1:/volumes# python3 tun_server.py
10.9.0.5:40983 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40983 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40983 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "tun_server.py", line 10, in <module>
    data, (ip, port) = sock.recvfrom(2048)
KeyboardInterrupt
```

After running on the ip route command on host U, the VPN server was able to capture ICMP packets from host U (10.9.0.5) to host V (192.168.60.5), the output structure is similar to the one above.

Task 4

Setup: run tun_server2.py on server-router, tun_client2.py on host U, add route on host U, tcpdump on host V

Configuring the VPN server will be similar to task 2D, but there is no need to use os.read() since the packet will be received through UDP at port 9090, and there is no need to check if it is an ICMP packet.

After configuring the VPN server, we configure the client to read a packet from the kernel since that is where the tunnel will write to

We need to run ip add route 192.168.60.0/24 dev Lim0 via 192.168.53.99 on host U

```
root@aabbc9e02a55:/# echo "HOST U"
HOST U
root@aabbc9e02a55:/# ip route add 192.168.60.0/24 dev Lim0 via 192.168.53.99
root@aabbc9e02a55:/# ping 192.168.60.5 -c 5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4104ms
```

Run tcpdump on host V to check if it captured a ping packet, it did capture a echo-request packet

```
root@9c20e5fffdbf:/# echo "HOST V"
HOST V
root@9c20e5fffdbf:/# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:07:27.085974 IP 192.168.53.99 > 9c20e5fffdbf: ICMP echo request, id 108, seq 1, length 64
06:07:27.085995 IP 9c20e5fffdbf > 192.168.53.99: ICMP echo reply, id 108, seq 1, length 64
06:07:27.086493 IP 9c20e5fffdbf.44823 > 192.168.1.254.domain: 24223+ PTR? 99.53.168.192.in-addr.arpa. (
44)
06:07:28.114413 IP 192.168.53.99 > 9c20e5fffdbf: ICMP echo request, id 108, seq 2, length 64
```

```
root@4e9d65e0fed1:/volumes# echo "server-router"
server-router
root@4e9d65e0fed1:/volumes# python3 tun_server2.py
Interface Name: Lim0
10.9.0.5:46044 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:46044 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:46044 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:46044 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:46044 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "tun_server2.py", line 31, in <module>
    data, (ip, port) = sock.recvfrom(2048)
KeyboardInterrupt
```

As seen from the results of tcpdump, host V did receive an echo-request packet from the VPN server (192.168.53.99) which was sent by host U(10.9.0.5). In conclusion, the ping packet reached host V from host U through the tunneling from the VPN server.

Task 5

For tun_client3.py, we add a new line to automatically run ip route add 192.168.60.0/24 dev Lim0, since it was always needed when the tunnel scripts were run.

Main points:

- If the file descriptor is a socket, get the packet from the UDP socket
- If the file descriptor is a tunnel, get the packet from the tunnel
- Ensure that ports and IP addresses are set up in the scripts so that the client and server can connect from the port.

We will run tun_server3.py on server-router, tun_client3.py on host U (10.9.0.5), and use host U to ping and telnet host V (192.168.60.5) in another window
Used wireshark to analyze packets, using the 'any' option → no filter, no addresses

ping
host U ping host V through the VPN server
Wireshark capture:

1	2023-11-14 02:1…	10.9.0.5	10.9.0.11	UDP	128	47917 → 9090 Len=84
2	2023-11-14 02:1…	10.9.0.5	10.9.0.11	UDP	128	47917 → 9090 Len=84
3	2023-11-14 02:1…	192.168.53.99	192.168.60.5	ICMP	100	Echo (ping) request
4	2023-11-14 02:1…	192.168.53.99	192.168.60.5	ICMP	100	Echo (ping) request
5	2023-11-14 02:1…	192.168.60.5	192.168.53.99	ICMP	100	Echo (ping) reply
6	2023-11-14 02:1…	192.168.60.5	192.168.53.99	ICMP	100	Echo (ping) reply
7	2023-11-14 02:1…	10.9.0.11	10.9.0.5	UDP	128	9090 → 47917 Len=84
8	2023-11-14 02:1…	10.9.0.11	10.9.0.5	UDP	128	9090 → 47917 Len=84

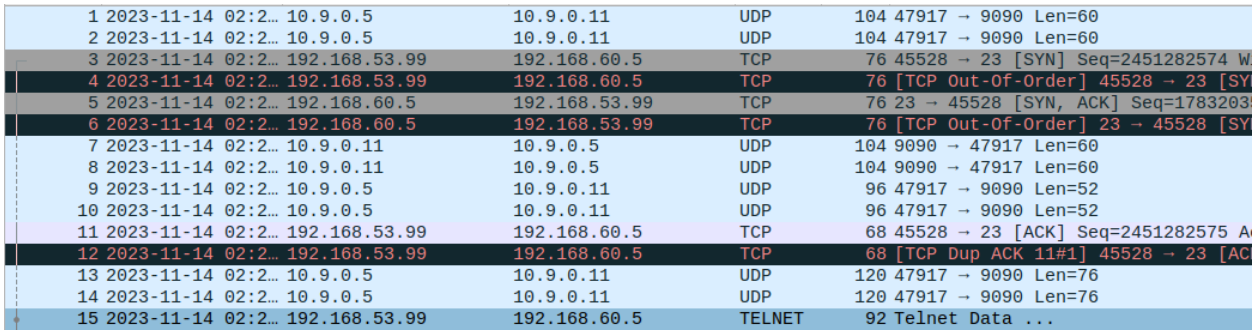
- ICMP packet goes from host U (10.9.0.5) to server-router (10.9.0.11) which is also 192.168.53.99
- server-router already has a route to host V (192.168.60.5)
- server-router forwards the ping echo-request to host V
- host V receives the request, and sends a ping reply to server-router
- server-router forwards the request to host U through the tunnel
- host U receives the echo-reply

This was successful as seen from the results in host U terminal, where there was 0% packet loss.

```
root@aabbc9e02a55:/# ping 192.168.60.5
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=63 time=1.93 ms
^C
--- 192.168.60.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.932/1.932/1.932/0.000 ms
root@aabbc9e02a55:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
9c20e5fffdbf 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 Nov 14 07:21:21 UTC 2023 on pts/1
seed@9c20e5fffdbf:~$ exit
```

telnet

host U telnet into host V



1	2023-11-14 02:2…	10.9.0.5	10.9.0.11	UDP	104	47917 → 9090 Len=60
2	2023-11-14 02:2…	10.9.0.5	10.9.0.11	UDP	104	47917 → 9090 Len=60
3	2023-11-14 02:2…	192.168.53.99	192.168.60.5	TCP	76	45528 → 23 [SYN] Seq=2451282574 W
4	2023-11-14 02:2…	192.168.53.99	192.168.60.5	TCP	76	[TCP Out-Of-Order] 45528 → 23 [SY
5	2023-11-14 02:2…	192.168.60.5	192.168.53.99	TCP	76	23 → 45528 [SYN, ACK] Seq=1783203
6	2023-11-14 02:2…	192.168.60.5	192.168.53.99	TCP	76	[TCP Out-Of-Order] 23 → 45528 [SY
7	2023-11-14 02:2…	10.9.0.11	10.9.0.5	UDP	104	9090 → 47917 Len=60
8	2023-11-14 02:2…	10.9.0.11	10.9.0.5	UDP	104	9090 → 47917 Len=60
9	2023-11-14 02:2…	10.9.0.5	10.9.0.11	UDP	96	47917 → 9090 Len=52
10	2023-11-14 02:2…	10.9.0.5	10.9.0.11	UDP	96	47917 → 9090 Len=52
11	2023-11-14 02:2…	192.168.53.99	192.168.60.5	TCP	68	45528 → 23 [ACK] Seq=2451282575 A
12	2023-11-14 02:2…	192.168.53.99	192.168.60.5	TCP	68	[TCP Dup ACK 11#1] 45528 → 23 [AC
13	2023-11-14 02:2…	10.9.0.5	10.9.0.11	UDP	120	47917 → 9090 Len=76
14	2023-11-14 02:2…	10.9.0.5	10.9.0.11	UDP	120	47917 → 9090 Len=76
15	2023-11-14 02:2…	192.168.53.99	192.168.60.5	TELNET	92	Telnet Data ...

- UDP packet containing telnet command is sent from host U (10.9.0.5) to server-router (10.9.0.11) which is also 192.168.53.99
- server-router sends the packet to host V (192.168.60.5) to initiate a telnet connection with TCP
- Thereafter the three-way handshake was done between server-router and host V
- With the previous tunneling set up from host U to server-router, a tunnel was created between host U and host V.
- With the help of server-router, wireshark was only able to capture packets between server-router and host V / host U, but not between host V and host U

The telnet was successful as seen from the image above where host U was able to login to host V

Task 6

We will reuse the scripts from task5 to establish a telnet connection between host U and host V
As seen, the telnet was successful

```
root@aabbc9e02a55:/# echo "HOST U"
HOST U
root@aabbc9e02a55:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
9c20e5fffdbf 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 Nov 14 08:05:36 UTC 2023 on pts/1
```

After the connection was broken on the client side, we were unable to type anything into the buffer.

```
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "tun_client3.py", line 34, in <module>
    ready, _, _ = select.select([sock, tun], [], [])
KeyboardInterrupt

root@aabbc9e02a55:/volumes# python3 tun_client3.py
Interface Name: Lim0
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
```

We quickly rerun the script to re-establish the telnet connection, and anything that was typed during the disconnection then filled up the buffer.

```
root@aabbc9e02a55:/# echo "HOST U"
HOST U
root@aabbc9e02a55:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
9c20e5fffdbf 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 Nov 14 08:05:36 UTC 2023 on pts/1
seed@9c20e5fffdbf:~$ asdasdascasd
```

The same also happened when we broke the connection from the server side

```
From socket <==: 192.168.53.99 --> 192.168.60.5
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
From socket <==: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "tun_server3.py", line 33, in <module>
    ready, _, _ = select.select([sock, tun], [], [])
KeyboardInterrupt

root@4e9d65e0fed1:/volumes# python3 tun_server3.py
Interface Name: Lim0
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
From socket <==: 192.168.53.99 --> 192.168.60.5
```

```
root@aabbc9e02a55:/# echo "HOST U"
HOST U
root@aabbc9e02a55:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
9c20e5fffdbf 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 Nov 14 08:05:36 UTC 2023 on pts/1
seed@9c20e5fffdbf:~$ broken from server
```

This can be due to the parties involved having a TCP connection state, as mentioned in the previous lab on firewall exploration, the TCP TIME_WAIT field will begin counting down from around 110s. After which, the TCP telnet connection is fully broken. Essentially, if we rerun the

script before the 110s countdown reaches 0, the telnet connection can still be re-established, and reconnected without problems.