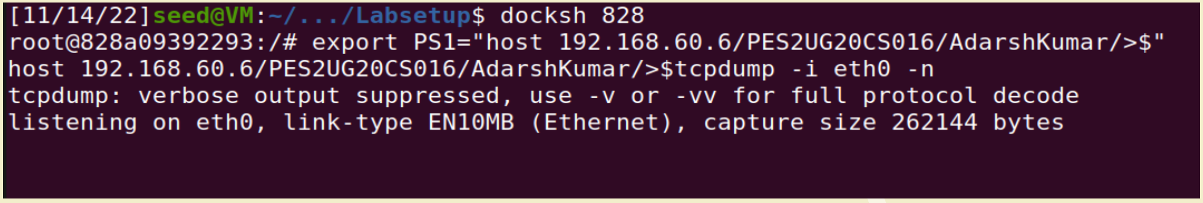
Name: Adarsh Kumar	SRN No: PES2UG20CS016	Assignment No: 09
	Section: B	Date: 11/11/2022

Task 1:	Network Setup
Screenshot	Checking packet sniffing using tcpdump command

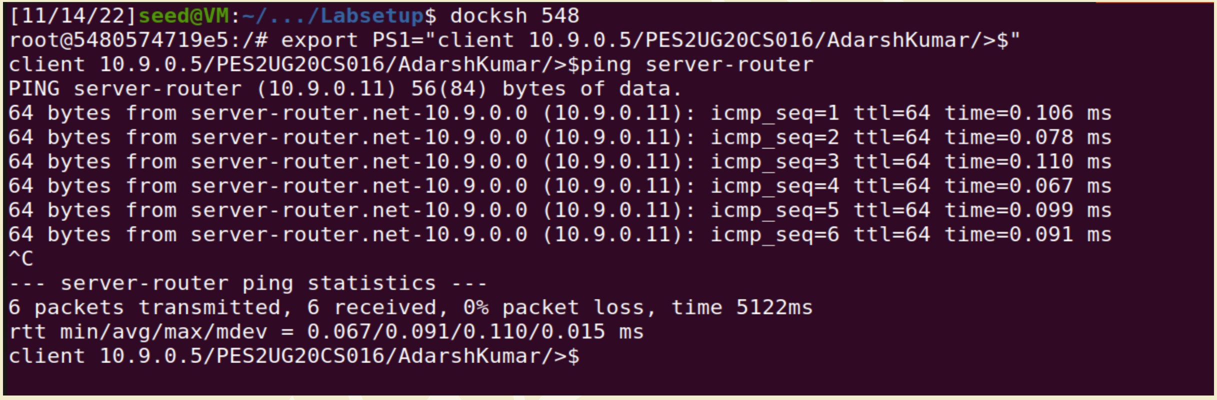


```
[11/14/22]seed@VM:~/.../Labsetup$ docksh 828
root@828a09392293:/# export PS1="host 192.168.60.6/PES2UG20CS016/AdarshKumar/>$"
host 192.168.60.6/PES2UG20CS016/AdarshKumar/>$tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

we can see that tcpdump is active and listening at interface eth0.

Testing

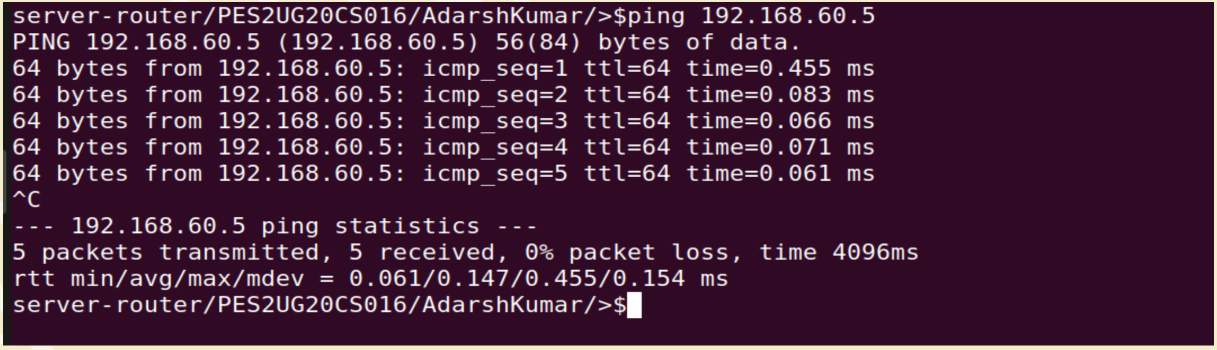
Pinging to server-router for client 10.9.0.5



```
[11/14/22]seed@VM:~/.../Labsetup$ docksh 548
root@5480574719e5:/# export PS1="client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$"
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping server-router
PING server-router (10.9.0.11) 56(84) bytes of data.
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.106 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.078 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.110 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.067 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.099 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=6 ttl=64 time=0.091 ms
^C
--- server-router ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5122ms
rtt min/avg/max/mdev = 0.067/0.091/0.110/0.015 ms
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

The connection is successfully established between server-router and client.

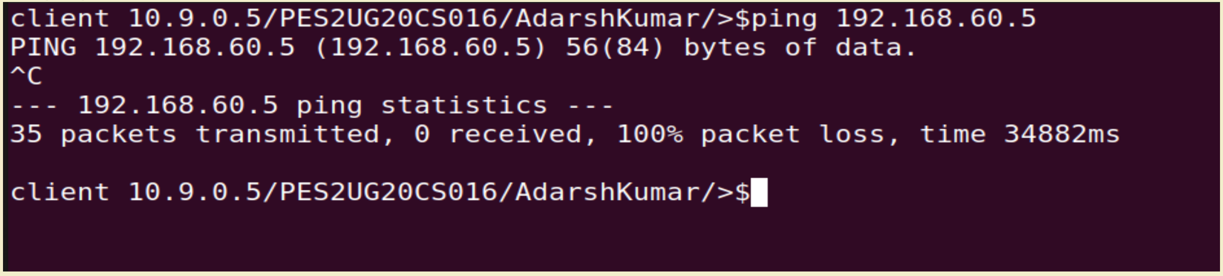
Pinging to Host V 192.168.60.5 from server-router



```
server-router/PES2UG20CS016/AdarshKumar/>$ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.455 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.066 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.061 ms
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4096ms
rtt min/avg/max/mdev = 0.061/0.147/0.455/0.154 ms
server-router/PES2UG20CS016/AdarshKumar/>$
```

As we can see that VPN Server can successfully established a connection to Host V (private network)

Pinging to Host V 192.168.60.5 from Host U 10.9.0.5

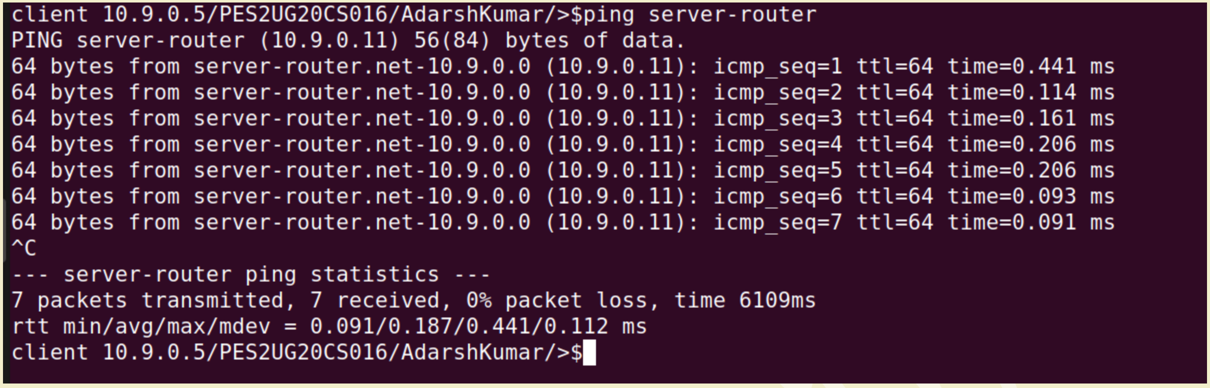


```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
35 packets transmitted, 0 received, 100% packet loss, time 34882ms

client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

All our sent packets are lost and we are not able to establish a connection between Host U & Host V.

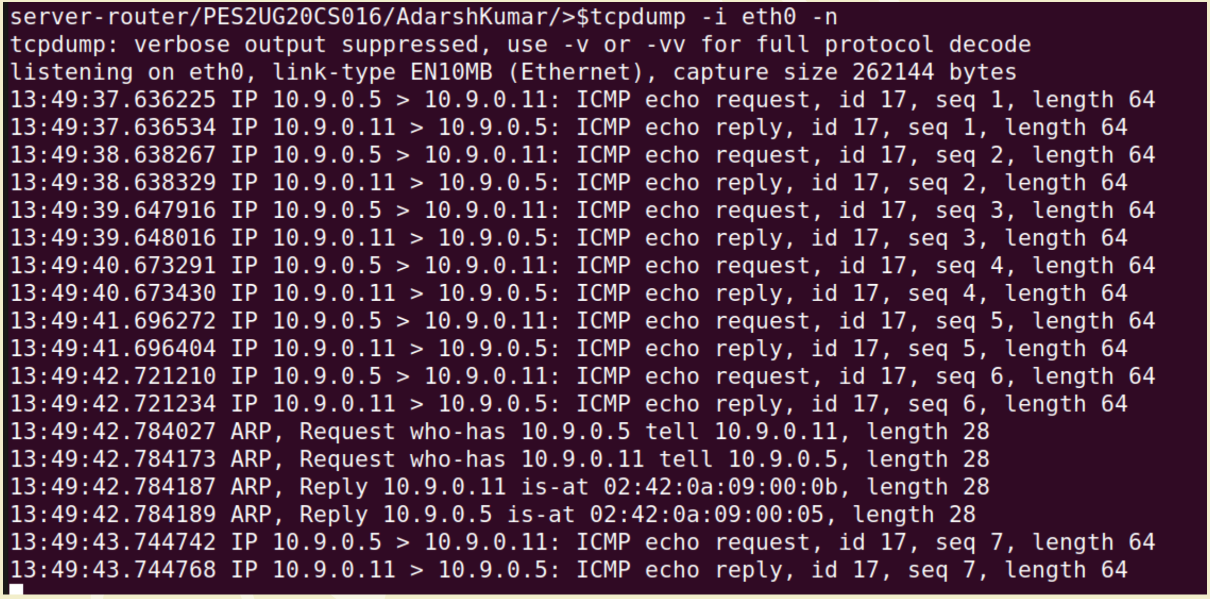
Sniffing the packet on network

Pinging to VPN Server-router from Client 10.9.0.5



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping server-router
PING server-router (10.9.0.11) 56(84) bytes of data.
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.441 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.114 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.161 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.206 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.206 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=6 ttl=64 time=0.093 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=7 ttl=64 time=0.091 ms
^C
--- server-router ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6109ms
rtt min/avg/max/mdev = 0.091/0.187/0.441/0.112 ms
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

Running tcpdump sniffer command on the VPN server-router.



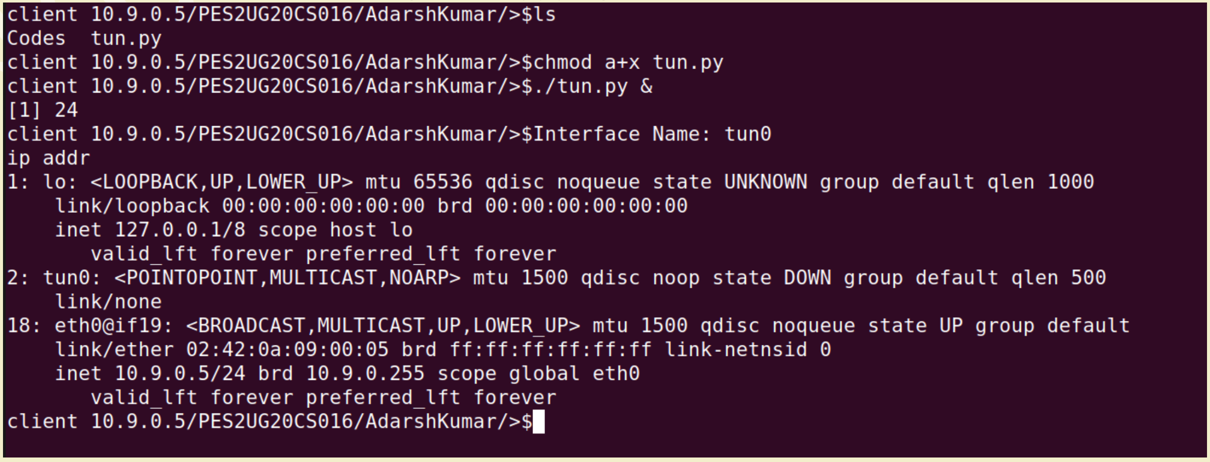
```
server-router/PES2UG20CS016/AdarshKumar/>$tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:49:37.636225 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 1, length 64
13:49:37.636534 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 1, length 64
13:49:38.638267 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 2, length 64
13:49:38.638329 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 2, length 64
13:49:39.647916 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 3, length 64
13:49:39.648016 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 3, length 64
13:49:40.673291 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 4, length 64
13:49:40.673430 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 4, length 64
13:49:41.696272 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 5, length 64
13:49:41.696404 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 5, length 64
13:49:42.721210 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 6, length 64
13:49:42.721234 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 6, length 64
13:49:42.784027 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
13:49:42.784173 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
13:49:42.784187 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
13:49:42.784189 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
13:49:43.744742 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 7, length 64
13:49:43.744768 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 7, length 64
```

We can see that when client 10.9.0.5 is pinging to server-router then we are able to sniff the packet.

Task 2	**Create and Configure TUN Interface**
Task 2.a:	Name of the Interface

Screenshot

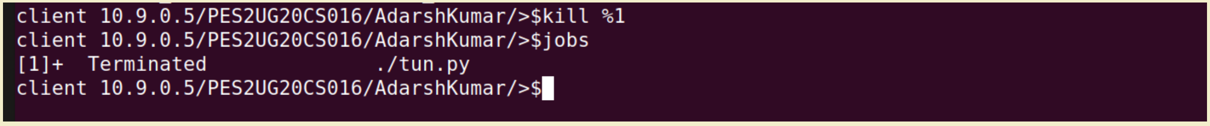
On Client - 10.9.0.5 we are running the tun.py program



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ls
Codes  tun.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$chmod a+x tun.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun.py &
[1] 24
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$Interface Name: tun0
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
2: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
18: eth0@if19: <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
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

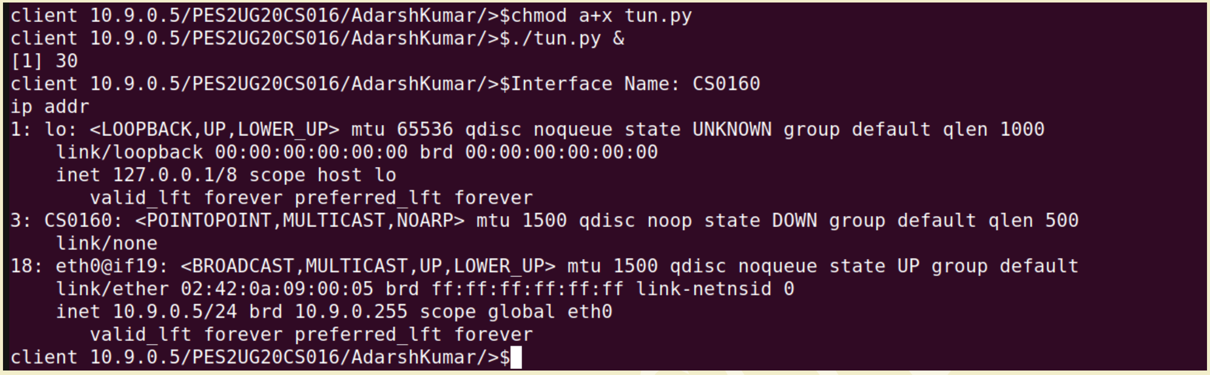
we can see that a new interface is created by the name of tun0 and presently it is in down state.

Killing the established tunnel



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$kill %1
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$jobs
[1]+  Terminated              ./tun.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

Now we are changing the name of interface

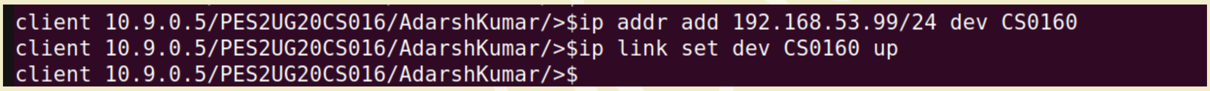


```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$chmod a+x tun.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun.py &
[1] 30
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$Interface Name: CS0160
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: CS0160: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
18: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

We can see that the interface name is CS0160 now SRN+0

Task 2.b: Set up the TUN Interface

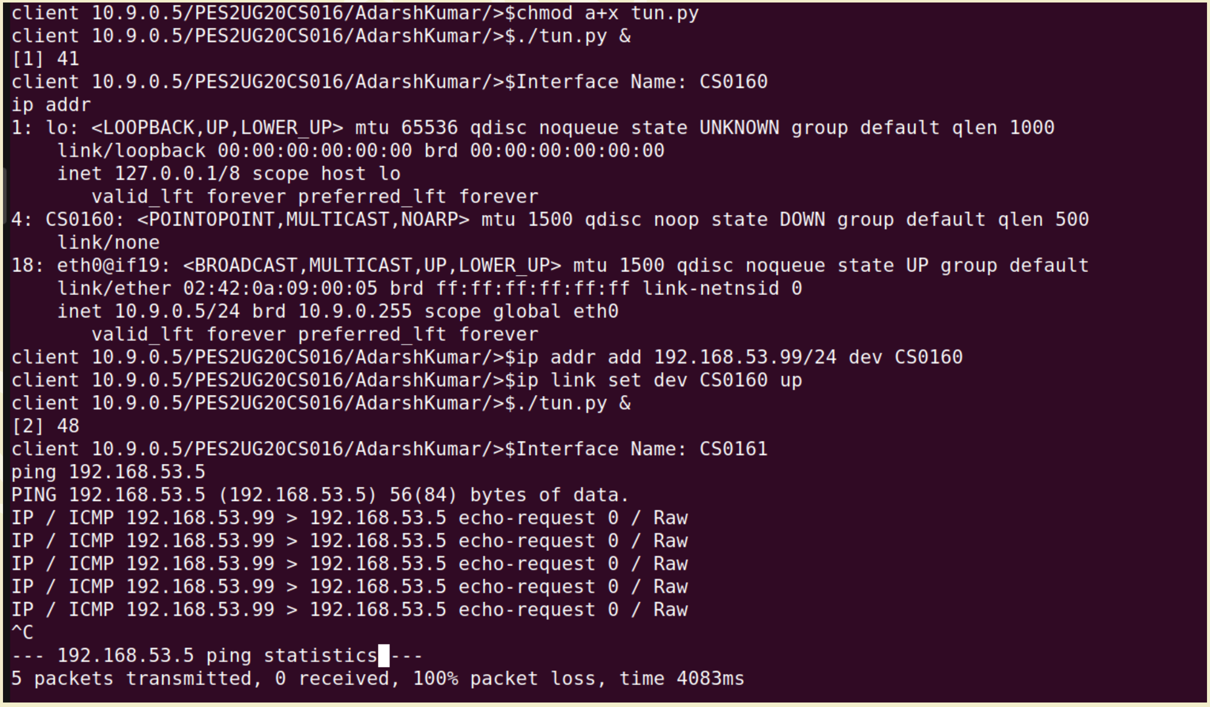
Screenshot: The Our interface in UP state



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ip addr add 192.168.53.99/24 dev CS0160
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ip link set dev CS0160 up
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

Task 2.c: Read from the TUN Interface

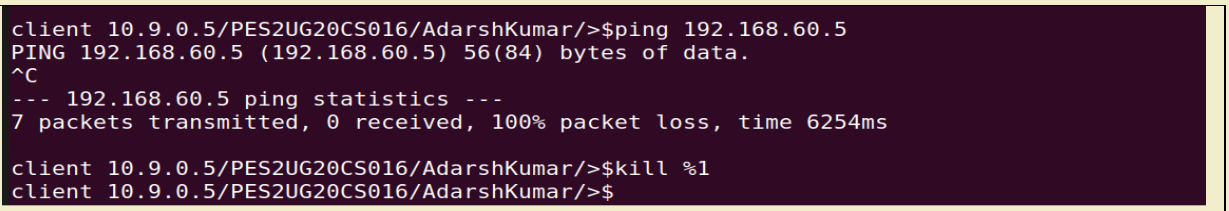
Screenshot: Now we are trying to read data passing through tun interface.



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$chmod a+x tun.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun.py &
[1] 41
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$Interface Name: CS0160
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
4: CS0160: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
18: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ip addr add 192.168.53.99/24 dev CS0160
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ip link set dev CS0160 up
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun.py &
[2] 48
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$Interface Name: CS0161
ping 192.168.53.5
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
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
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
^C
--- 192.168.53.5 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4083ms
```

On Host U, ping a host in the 192.168.53.0/24 network. What is printed out by the tun.py program? What has happened? Why?

Ans: This is because another end of the tunnel is not setup yet, the packet found a route for 192.168.53.0/24 via tun0, so the packet pass top this interface and the application able to get the packet. For 192.168.53.99, the ping command success, because it is the address of local adapter, so the ICMP message goes into loopback interface.



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$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 ---
7 packets transmitted, 0 received, 100% packet loss, time 6254ms

client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$kill %1
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

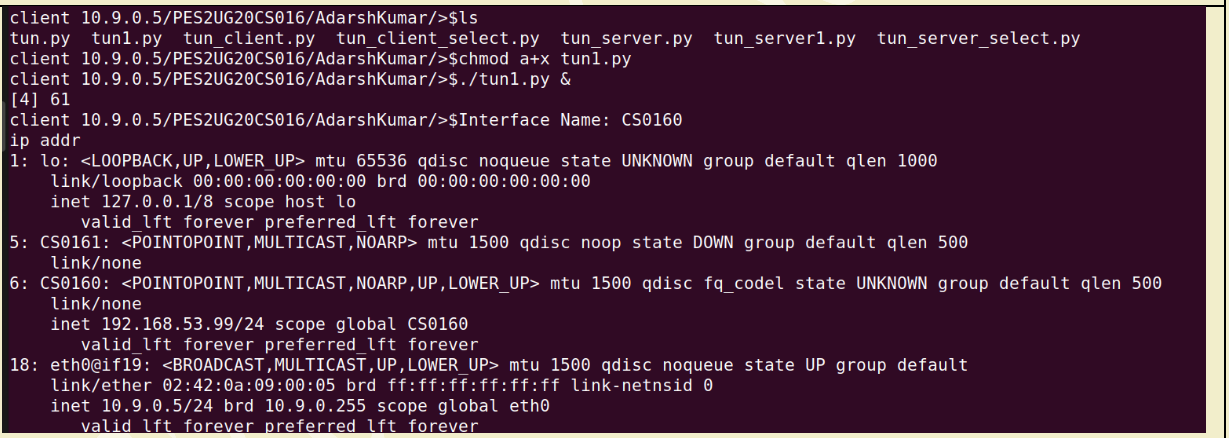
On Host U, ping a host in the internal network 192.168.60.0/24, Does tun.py print out anything? Why?

Ans: A ping request to any address under 192.168.60.0/24 is unresponsive on our tunnel interface as no data is being sent or received from that interface. It happened because which command ping execute to IP address 192.168.60.1, the OS lookup the routing table and found the packet should pass via the physical interface ens33, so the packet did not pass through tun0 interface and the application cannot capture the packet.

It is worth noting that when ping192.168.53.99 is our pipe interface, it will not receive packets on this interface, but the terminal that sends the ping request can receive packets. The guess is because the system found that it was sent to its own address, so it was replaced with a loopback address. Use tcpdump to monitor the loopback address, as expected:

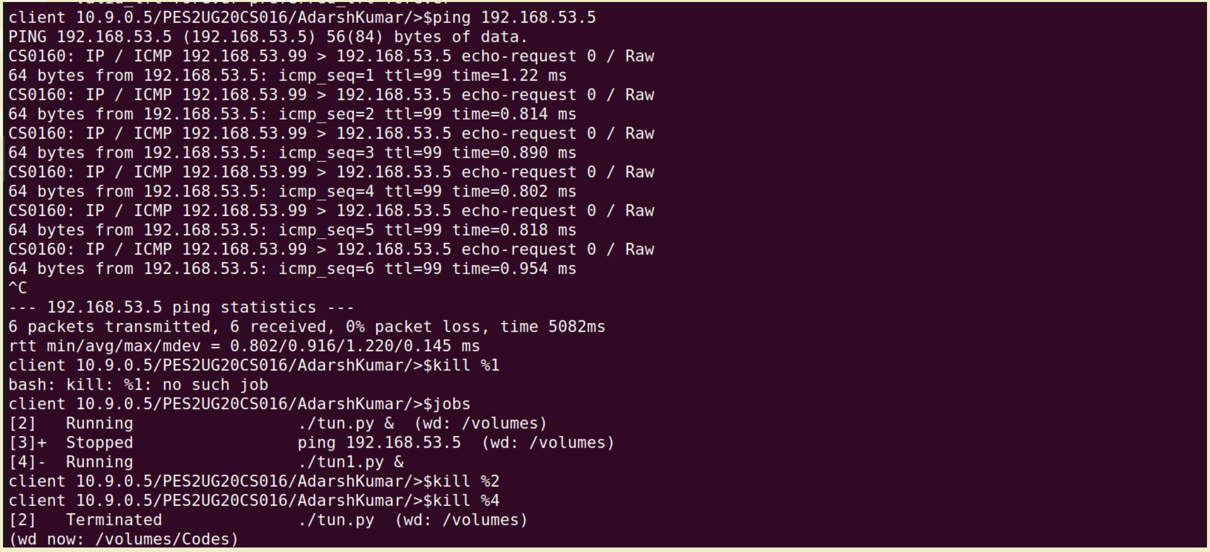
Task 2.d: Write to the TUN Interface

Screenshot



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ls
tun.py  tun1.py  tun_client.py  tun_client_select.py  tun_server.py  tun_server1.py  tun_server_select.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$chmod a+x tun1.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun1.py &
[4] 61
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$Interface Name: CS0160
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
5: CS0161: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
6: CS0160: <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 CS0160
       valid_lft forever preferred_lft forever
18: eth0@if19: <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
```

After 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.



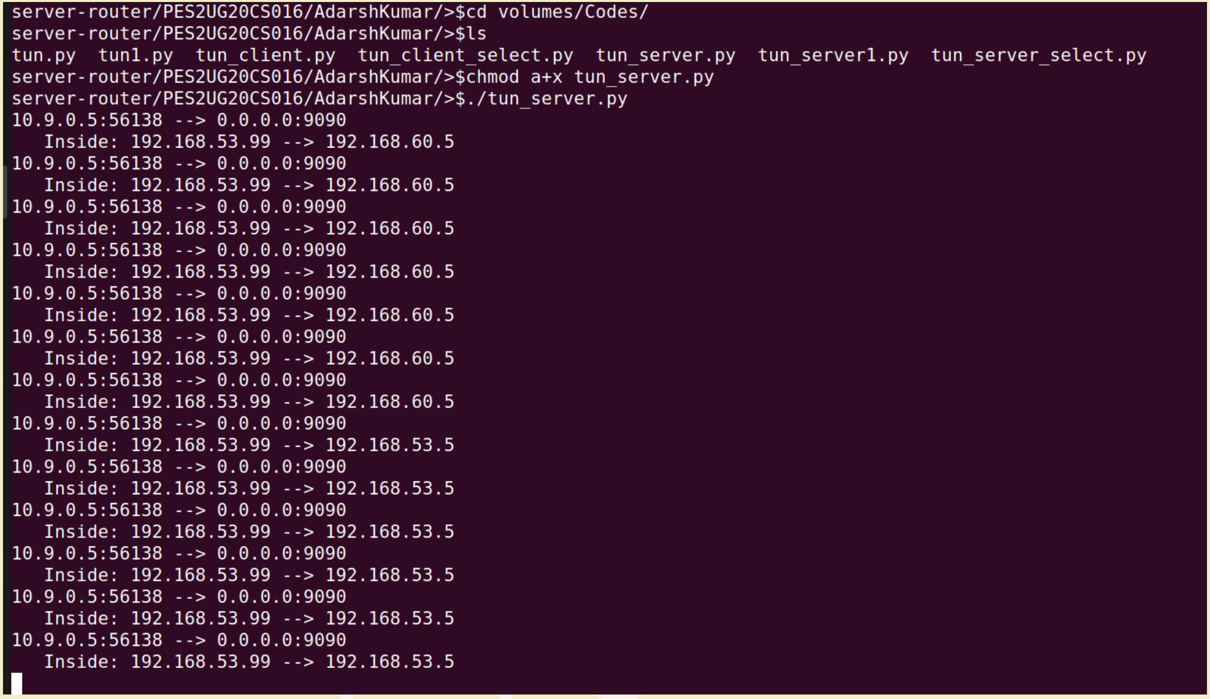
```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
CS0160: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=1 ttl=99 time=1.22 ms
CS0160: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=2 ttl=99 time=0.814 ms
CS0160: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=3 ttl=99 time=0.890 ms
CS0160: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=4 ttl=99 time=0.802 ms
CS0160: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=5 ttl=99 time=0.818 ms
CS0160: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=6 ttl=99 time=0.954 ms
^C
--- 192.168.53.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5082ms
rtt min/avg/max/mdev = 0.802/0.916/1.220/0.145 ms
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$kill %1
bash: kill: %1: no such job
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$jobs
[2]   Running                 ./tun.py &  (wd: /volumes)
[3]+  Stopped                 ping 192.168.53.5  (wd: /volumes)
[4]-  Running                 ./tun1.py &
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$kill %2
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$kill %4
[2]   Terminated              ./tun.py  (wd: /volumes)
(wd now: /volumes/Codes)
```

It can be seen tun successfully received the message and returned the corresponding ICMP message.

Since any IP starting point and ending point is the local machine, the message will be received by the kernel (known by task 4), and the message that does not meet the requirements of the network segment will also be rejected, so the messages sent and received here are only used to show the code

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

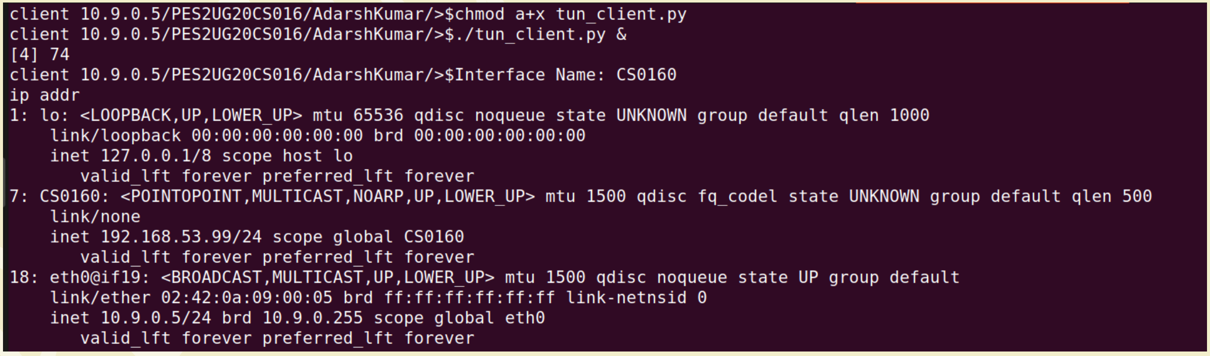
Listing to Server-router of VPN



```
server-router/PES2UG20CS016/AdarshKumar/>$cd volumes/Codes/
server-router/PES2UG20CS016/AdarshKumar/>$ls
tun.py  tun1.py  tun_client.py  tun_client_select.py  tun_server.py  tun_server1.py  tun_server_select.py
server-router/PES2UG20CS016/AdarshKumar/>$chmod a+x tun_server.py
server-router/PES2UG20CS016/AdarshKumar/>$./tun_server.py
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
```

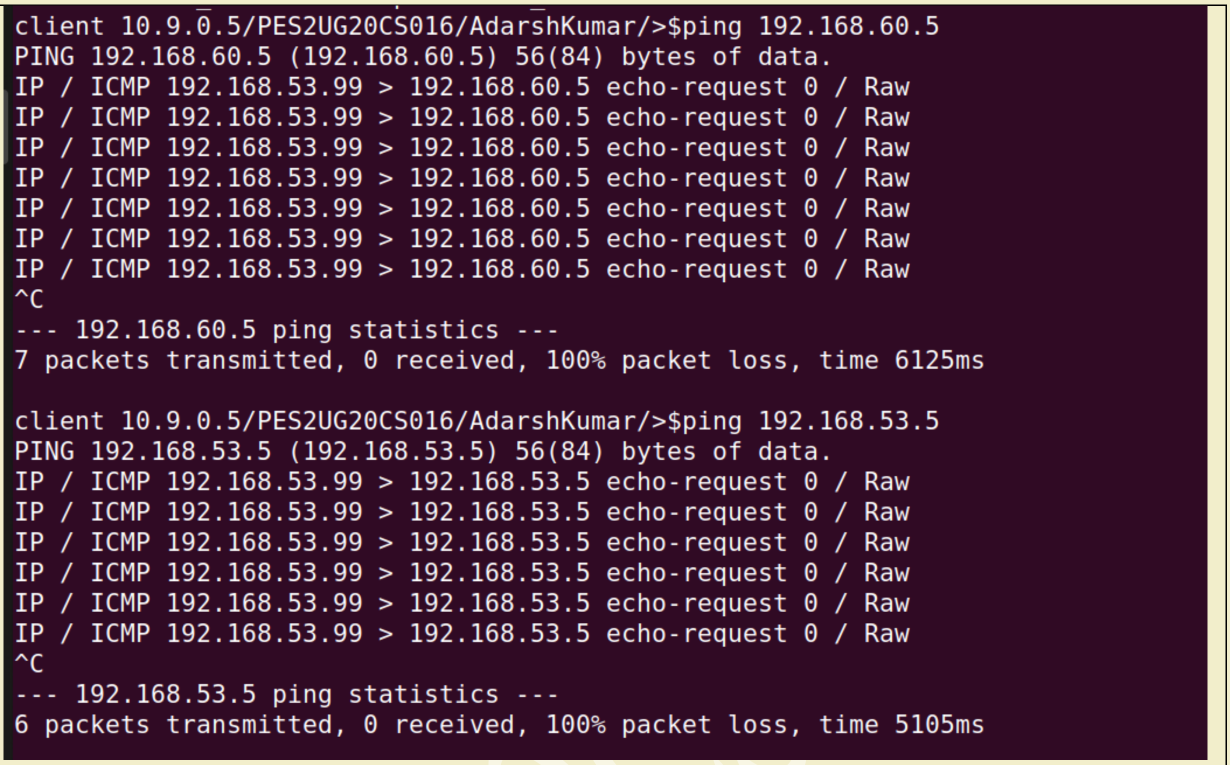
As you can see, the VPN-SERVER has successfully received and is ready to forward.

Establishing a tunnel from client side HOST U



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$chmod a+x tun_client.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun_client.py &
[4] 74
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$Interface Name: CS0160
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
7: CS0160: <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 CS0160
       valid_lft forever preferred_lft forever
18: eth0@if19: <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
```

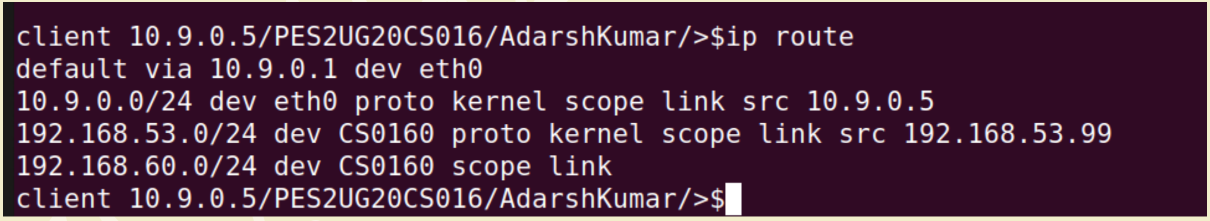
the client will automatically transfer 192.168.53.0/24 to the sun0 interface we set before, but it is not a real (virtual) interface and cannot really send packets. But our program can read it and package it and send it to the server we set. The server unpacks it and reads that message which we sent actually.



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6125ms

client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping 192.168.53.5
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
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
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
^C
--- 192.168.53.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5105ms
```

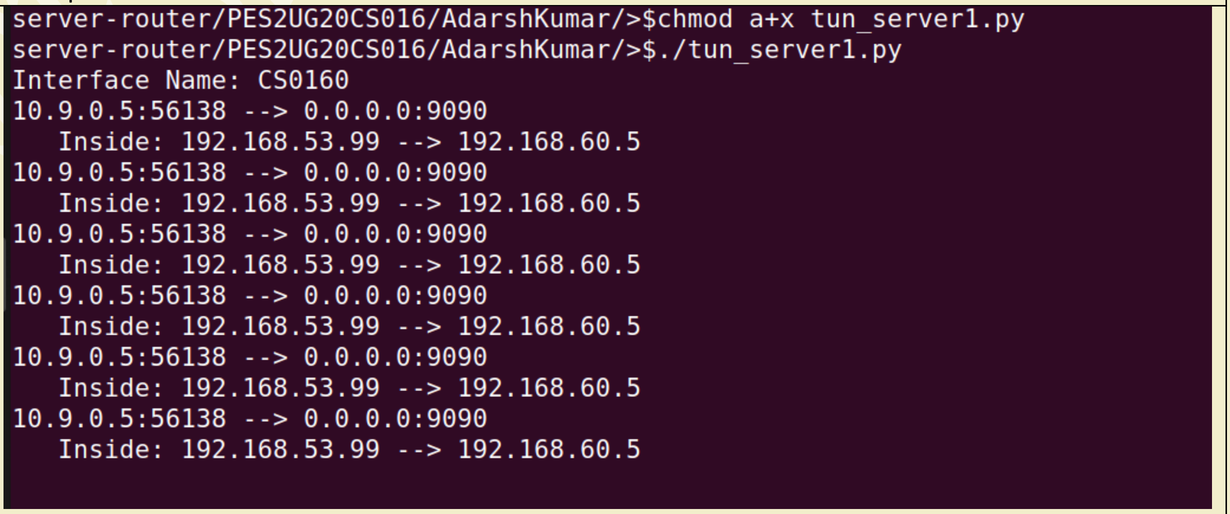
There is output for packet send to the 192.168.53.5 & 192.168.60.5. But ping test fail. The reason behind is that there is only one way traffic from HOST U to VPN Server the tunnel works in one way and no IP address assigned as 192.168.53.100 at VPN Server, and you can observe the ping packet is encapsulated inside UDP packet.

We want the packets going to HOST-V to go through tun, so we need to configure the routing table:



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev CS0160 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev CS0160 scope link
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

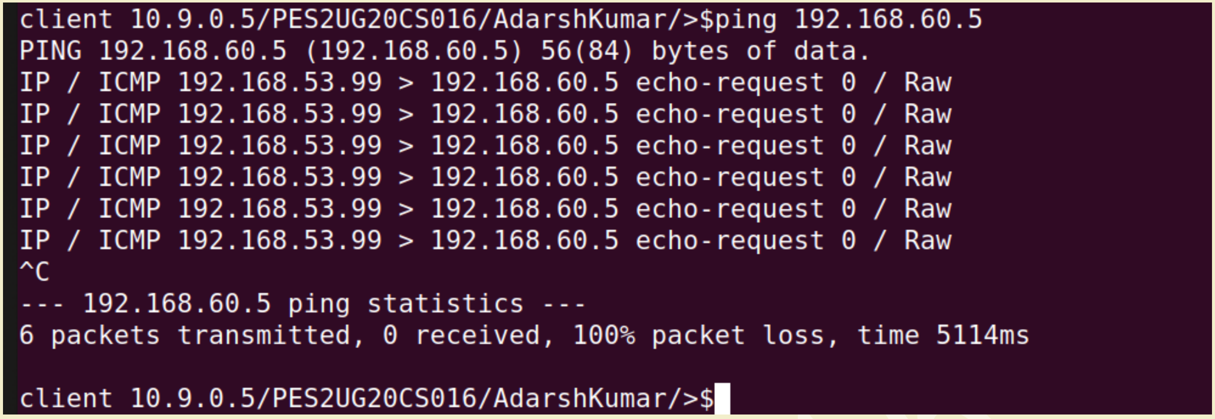
Task 4: Set Up the VPN Server



```
server-router/PES2UG20CS016/AdarshKumar/>$chmod a+x tun_server1.py
server-router/PES2UG20CS016/AdarshKumar/>$./tun_server1.py
Interface Name: CS0160
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56138 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
```

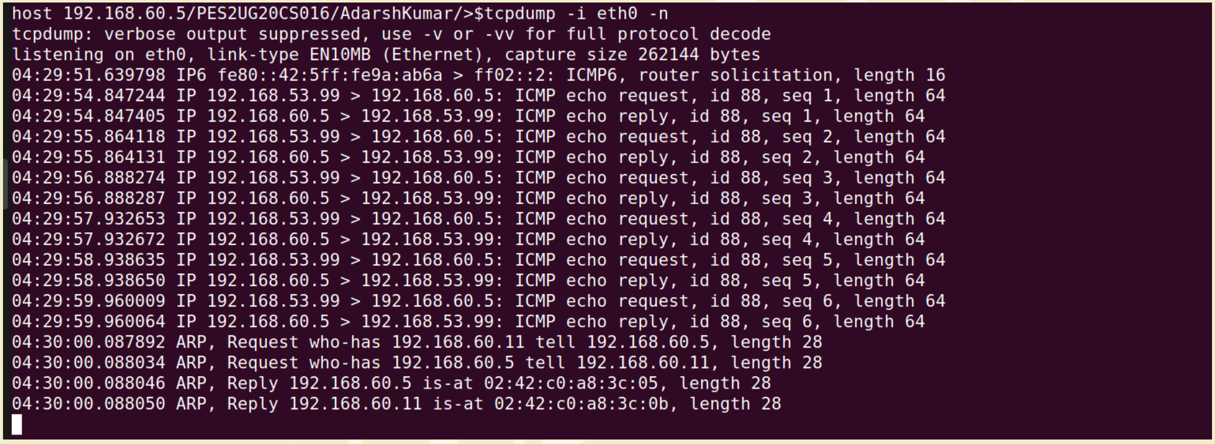
From the output we can see that the code runs correctly and sends the ICMP packets sent in the tunnel.

Ping the Private network (192.168.60.5) from Client 10.9.0.5



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5114ms

client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

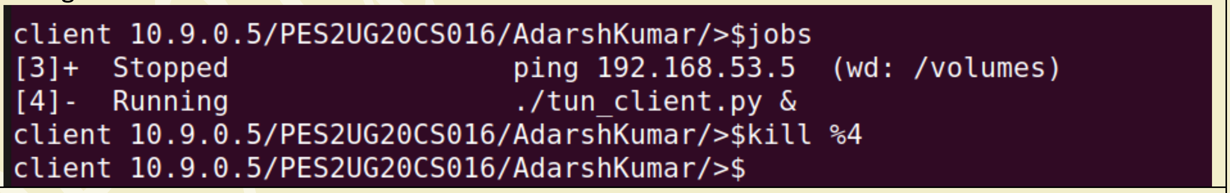
On Host 192.168.60.5 running the tcpdump to capture packet



```
host 192.168.60.5/PES2UG20CS016/AdarshKumar/>$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
04:29:51.639798 IP6 fe80::42:5ff:fe9a:ab6a > ff02::2: ICMP6, router solicitation, length 16
04:29:54.847244 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 88, seq 1, length 64
04:29:54.847405 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 88, seq 1, length 64
04:29:55.864118 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 88, seq 2, length 64
04:29:55.864131 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 88, seq 2, length 64
04:29:56.888274 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 88, seq 3, length 64
04:29:56.888287 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 88, seq 3, length 64
04:29:57.932653 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 88, seq 4, length 64
04:29:57.932672 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 88, seq 4, length 64
04:29:58.938635 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 88, seq 5, length 64
04:29:58.938650 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 88, seq 5, length 64
04:29:59.960009 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 88, seq 6, length 64
04:29:59.960064 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 88, seq 6, length 64
04:30:00.087892 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
04:30:00.088034 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
04:30:00.088046 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
04:30:00.088050 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

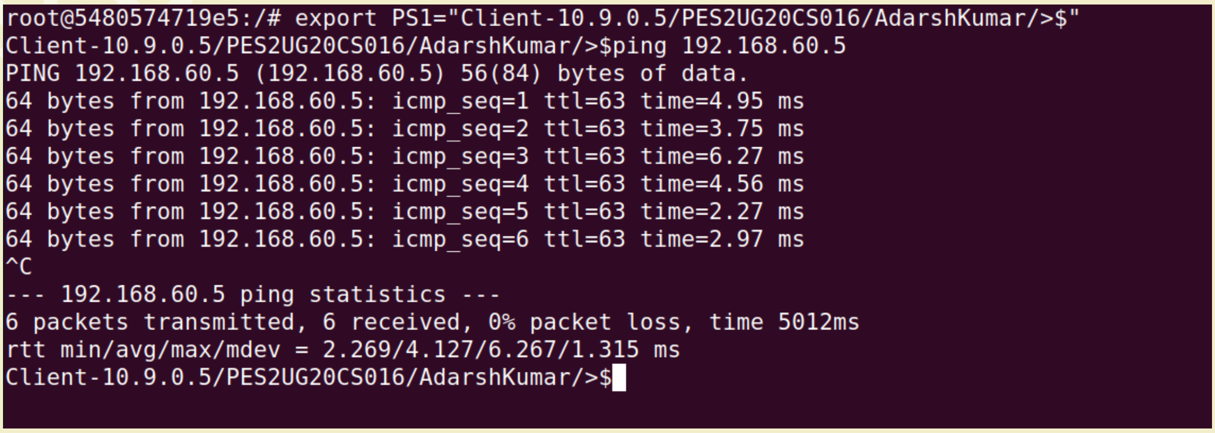
It can be seen that although there is no return function at present, the message has been correctly sent to HOST-V.

Killing the earlier Tunnel Process -



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$jobs
[3]+  Stopped                 ping 192.168.53.5  (wd: /volumes)
[4]-  Running                 ./tun_client.py &
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$kill %4
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

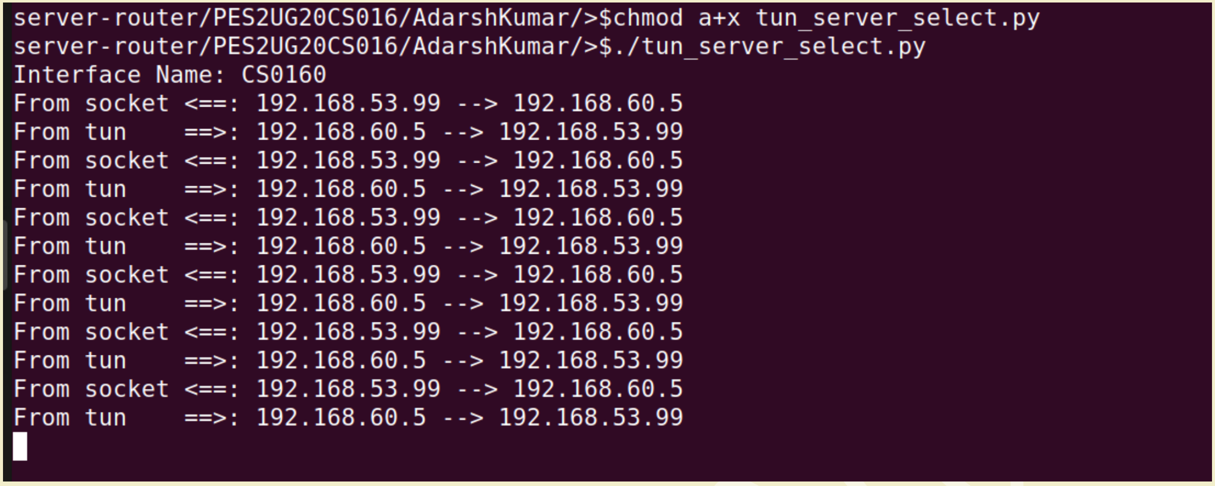
Task 5: Handling Traffic in Both Directions



```
root@5480574719e5:/# export PS1="Client-10.9.0.5/PES2UG20CS016/AdarshKumar/>$"
Client-10.9.0.5/PES2UG20CS016/AdarshKumar/>$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=4.95 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=3.75 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=6.27 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=4.56 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=2.27 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=2.97 ms
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5012ms
rtt min/avg/max/mdev = 2.269/4.127/6.267/1.315 ms
Client-10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

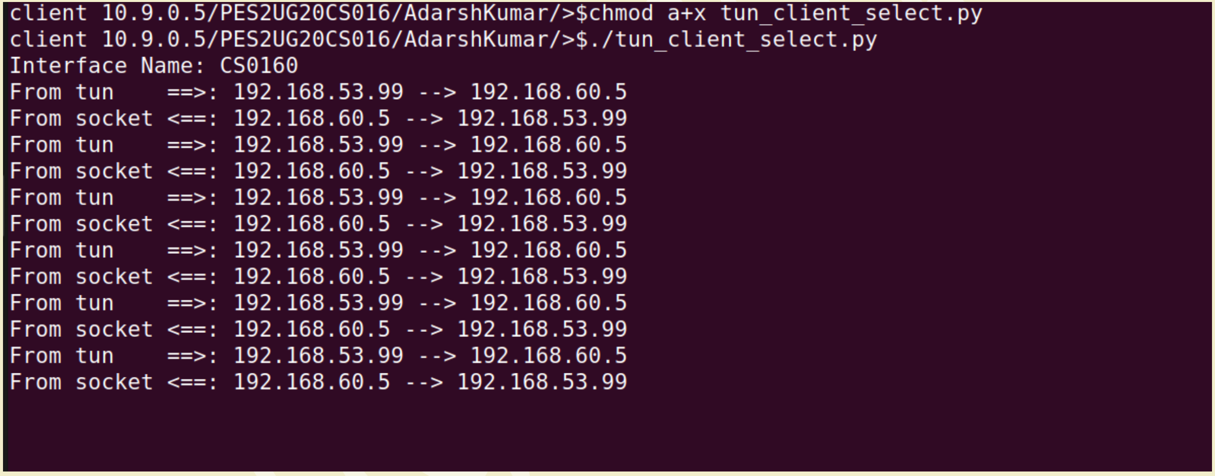
The client Host-U has been able to ping to Host-V normally:

Server is listening via Tun interface



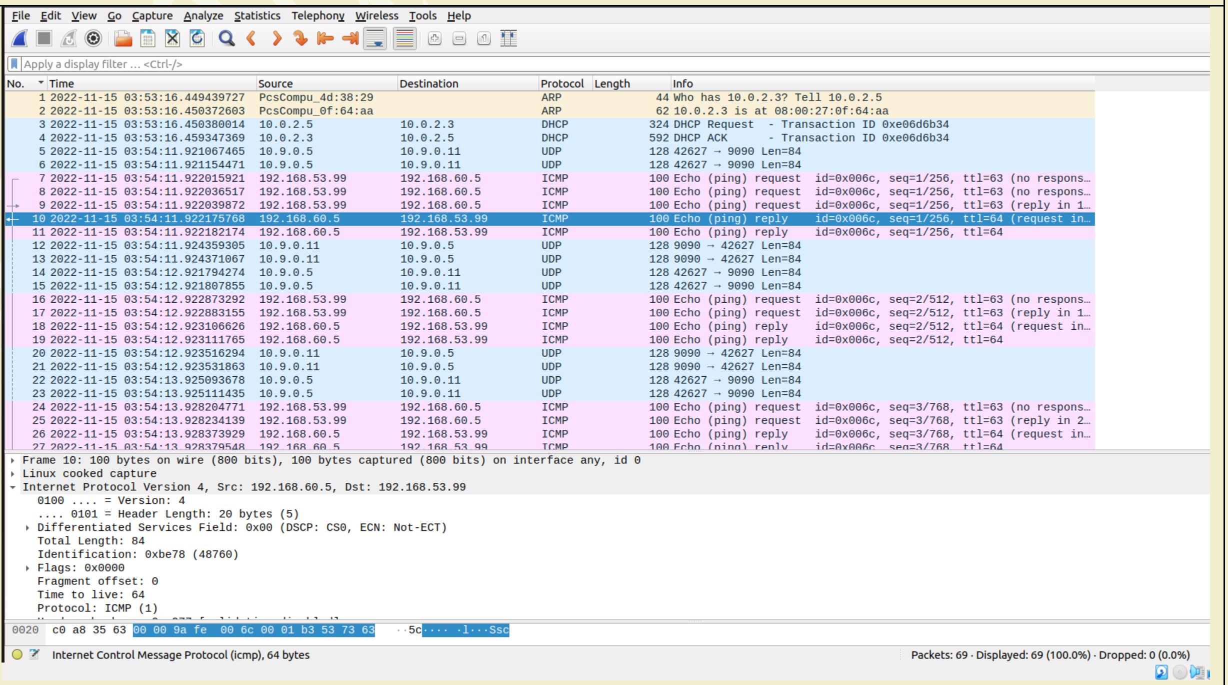
```
server-router/PES2UG20CS016/AdarshKumar/>$chmod a+x tun_server_select.py
server-router/PES2UG20CS016/AdarshKumar/>$./tun_server_select.py
Interface Name: CS0160
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
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
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
```

Client Is listening via Tun interface



```
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$chmod a+x tun_client_select.py
client 10.9.0.5/PES2UG20CS016/AdarshKumar/>$./tun_client_select.py
Interface Name: CS0160
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
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
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
From socket <==: 192.168.60.5 --> 192.168.53.99
```

Wireshark Screenshot



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

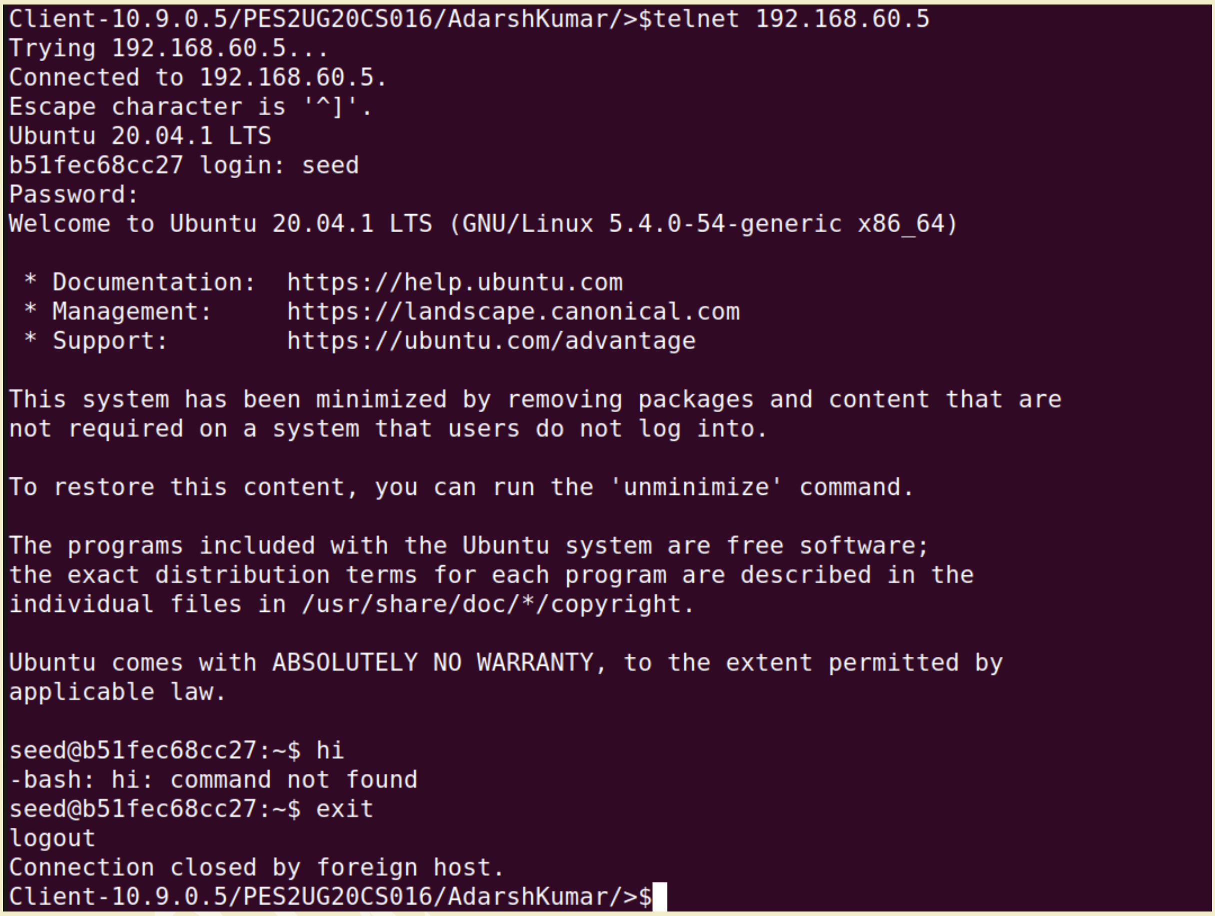
```
No.  Time                          Source              Destination     Protocol Length Info
   1 2022-11-15 03:53:16.449439727 PcsCompu_4d:38:29                   ARP         44 Who has 10.0.2.3? Tell 10.0.2.5
   2 2022-11-15 03:53:16.450372603 PcsCompu_0f:64:aa                   ARP         62 10.0.2.3 is at 08:00:27:0f:64:aa
   3 2022-11-15 03:53:16.450380014 10.0.2.5            10.0.2.3        DHCP       324 DHCP Request  - Transaction ID 0xe06d6b34
   4 2022-11-15 03:53:16.459347369 10.0.2.3            10.0.2.5        DHCP       592 DHCP ACK      - Transaction ID 0xe06d6b34
   5 2022-11-15 03:54:11.921067465 10.9.0.5            10.9.0.11       UDP        128 42627 → 9090 Len=84
   6 2022-11-15 03:54:11.921154471 10.9.0.5            10.9.0.11       UDP        128 42627 → 9090 Len=84
   7 2022-11-15 03:54:11.922015921 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=1/256, ttl=63 (no respons…
   8 2022-11-15 03:54:11.922036517 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=1/256, ttl=63 (no respons…
   9 2022-11-15 03:54:11.922039872 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=1/256, ttl=63 (reply in 1…
  10 2022-11-15 03:54:11.922175768 192.168.60.5        192.168.53.99   ICMP       100 Echo (ping) reply    id=0x006c, seq=1/256, ttl=64 (request in…
  11 2022-11-15 03:54:11.922182174 192.168.60.5        192.168.53.99   ICMP       100 Echo (ping) reply    id=0x006c, seq=1/256, ttl=64
  12 2022-11-15 03:54:11.924359305 10.9.0.11           10.9.0.5        UDP        128 9090 → 42627 Len=84
  13 2022-11-15 03:54:11.924371067 10.9.0.11           10.9.0.5        UDP        128 9090 → 42627 Len=84
  14 2022-11-15 03:54:12.921794274 10.9.0.5            10.9.0.11       UDP        128 42627 → 9090 Len=84
  15 2022-11-15 03:54:12.921807855 10.9.0.5            10.9.0.11       UDP        128 42627 → 9090 Len=84
  16 2022-11-15 03:54:12.922873292 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=2/512, ttl=63 (no respons…
  17 2022-11-15 03:54:12.922883155 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=2/512, ttl=63 (reply in 1…
  18 2022-11-15 03:54:12.923106626 192.168.60.5        192.168.53.99   ICMP       100 Echo (ping) reply    id=0x006c, seq=2/512, ttl=64 (request in…
  19 2022-11-15 03:54:12.923111765 192.168.60.5        192.168.53.99   ICMP       100 Echo (ping) reply    id=0x006c, seq=2/512, ttl=64
  20 2022-11-15 03:54:12.923516294 10.9.0.11           10.9.0.5        UDP        128 9090 → 42627 Len=84
  21 2022-11-15 03:54:12.923531863 10.9.0.11           10.9.0.5        UDP        128 9090 → 42627 Len=84
  22 2022-11-15 03:54:13.925093678 10.9.0.5            10.9.0.11       UDP        128 42627 → 9090 Len=84
  23 2022-11-15 03:54:13.925111435 10.9.0.5            10.9.0.11       UDP        128 42627 → 9090 Len=84
  24 2022-11-15 03:54:13.928204771 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=3/768, ttl=63 (no respons…
  25 2022-11-15 03:54:13.928234139 192.168.53.99       192.168.60.5    ICMP       100 Echo (ping) request  id=0x006c, seq=3/768, ttl=63 (reply in 2…
  26 2022-11-15 03:54:13.928373929 192.168.60.5        192.168.53.99   ICMP       100 Echo (ping) reply    id=0x006c, seq=3/768, ttl=64 (request in…
  27 2022-11-15 03:54:13.928379548 192.168.60.5        192.168.53.99   ICMP       100 Echo (ping) reply    id=0x006c, seq=3/768, ttl=64
```

```
› Frame 10: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
› Linux cooked capture
˅ Internet Protocol Version 4, Src: 192.168.60.5, Dst: 192.168.53.99
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xbe78 (48760)
  › Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)

0020  c0 a8 35 63 00 00 9a fe  00 6c 00 01 b3 53 73 63   ··5c···· ·l···Ssc

Internet Control Message Protocol (icmp), 64 bytes          Packets: 69 · Displayed: 69 (100.0%) · Dropped: 0 (0.0%)
```

It can be seen that ping can be roughly divided into four processes:

HOST-U sent to VPN-SERVER
VPN-SERVER sends ping request to HOST-V
HOST-V replies to VPN-SERVER's ping request
VPN-SERVER sends reply back to HOST-U

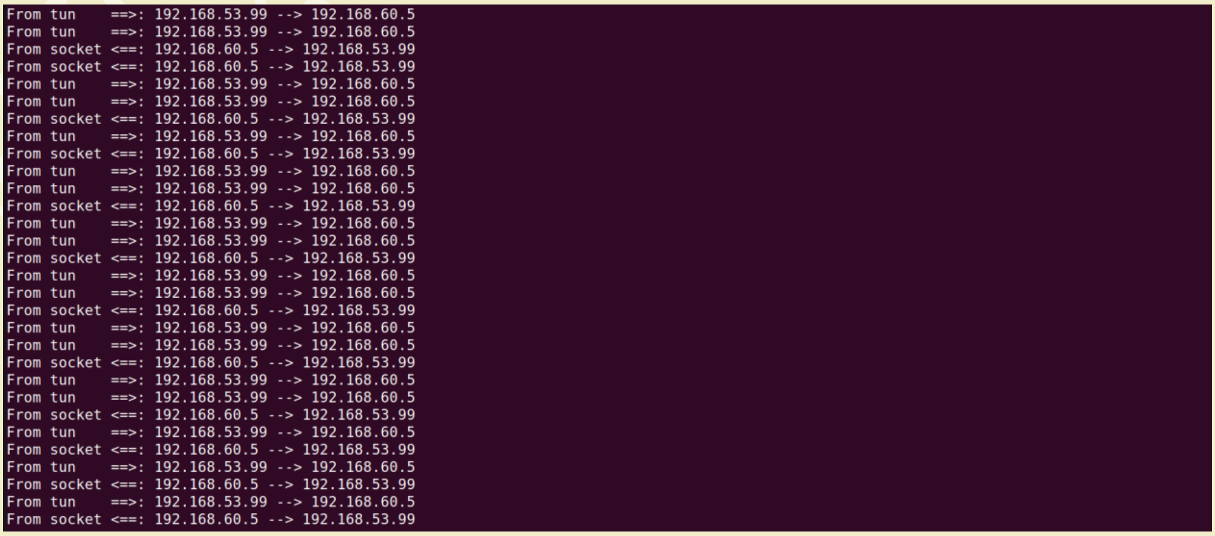
Screenshot for telnet test success



```
Client-10.9.0.5/PES2UG20CS016/AdarshKumar/>$telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
b51fec68cc27 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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@b51fec68cc27:~$ hi
-bash: hi: command not found
seed@b51fec68cc27:~$ exit
logout
Connection closed by foreign host.
Client-10.9.0.5/PES2UG20CS016/AdarshKumar/>$
```

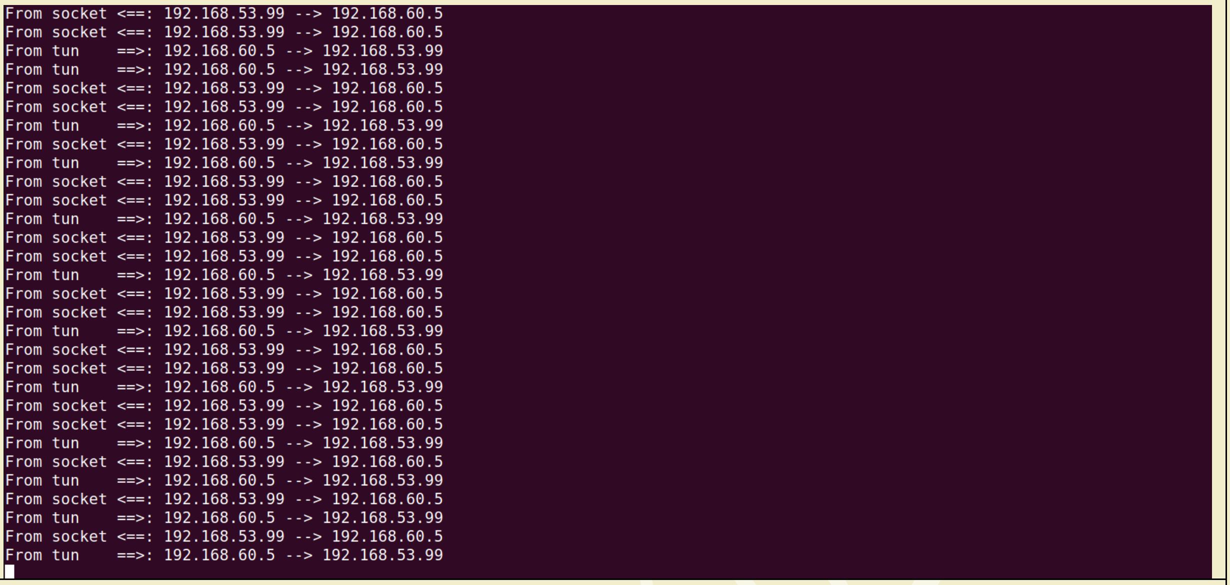
At the same time, remote login can also be completed

Client



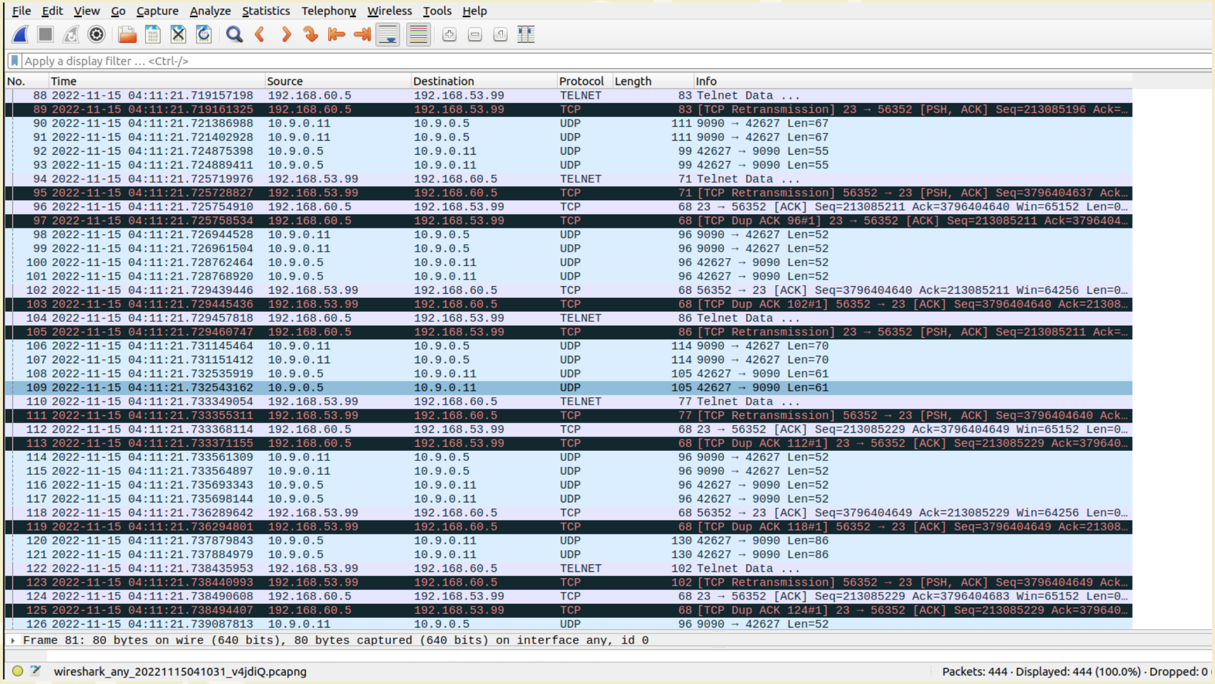
```
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
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
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
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
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
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
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
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
From socket <==: 192.168.60.5 --> 192.168.53.99
```

Server



```
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 tun    ==>: 192.168.60.5 --> 192.168.53.99
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
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 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 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 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 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
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
```

Packet capture at HOST V of the telnet test


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
88	2022-11-15 04:11:21.719157198	192.168.60.5	192.168.53.99	TELNET	83	Telnet Data ...
89	2022-11-15 04:11:21.719161325	192.168.60.5	192.168.53.99	TCP	83	[TCP Retransmission] 23 → 56352 [PSH, ACK] Seq=213085196 Ack=…
90	2022-11-15 04:11:21.721386988	10.9.0.11	10.9.0.5	UDP	111	9090 → 42627 Len=67
91	2022-11-15 04:11:21.721402928	10.9.0.11	10.9.0.5	UDP	111	9090 → 42627 Len=67
92	2022-11-15 04:11:21.724875398	10.9.0.5	10.9.0.11	UDP	99	42627 → 9090 Len=55
93	2022-11-15 04:11:21.724889411	10.9.0.5	10.9.0.11	UDP	99	42627 → 9090 Len=55
94	2022-11-15 04:11:21.725719976	192.168.53.99	192.168.60.5	TELNET	71	Telnet Data ...
95	2022-11-15 04:11:21.725728827	192.168.53.99	192.168.60.5	TCP	71	[TCP Retransmission] 56352 → 23 [PSH, ACK] Seq=3796404637 Ack…
96	2022-11-15 04:11:21.725754910	192.168.60.5	192.168.53.99	TCP	68	23 → 56352 [ACK] Seq=213085211 Ack=3796404640 Win=65152 Len=0…
97	2022-11-15 04:11:21.725758534	192.168.60.5	192.168.53.99	TCP	68	[TCP Dup ACK 96#1] 23 → 56352 [ACK] Seq=213085211 Ack=3796404…
98	2022-11-15 04:11:21.726944528	10.9.0.11	10.9.0.5	UDP	96	9090 → 42627 Len=52
99	2022-11-15 04:11:21.726961504	10.9.0.11	10.9.0.5	UDP	96	9090 → 42627 Len=52
100	2022-11-15 04:11:21.728762464	10.9.0.5	10.9.0.11	UDP	96	42627 → 9090 Len=52
101	2022-11-15 04:11:21.728768920	10.9.0.5	10.9.0.11	UDP	96	42627 → 9090 Len=52
102	2022-11-15 04:11:21.729439446	192.168.53.99	192.168.60.5	TCP	68	56352 → 23 [ACK] Seq=3796404640 Ack=213085211 Win=64256 Len=0…
103	2022-11-15 04:11:21.729445436	192.168.53.99	192.168.60.5	TCP	68	[TCP Dup ACK 102#1] 56352 → 23 [ACK] Seq=3796404640 Ack=21308…
104	2022-11-15 04:11:21.729457818	192.168.60.5	192.168.53.99	TELNET	86	Telnet Data ...
105	2022-11-15 04:11:21.729460747	192.168.60.5	192.168.53.99	TCP	86	[TCP Retransmission] 23 → 56352 [PSH, ACK] Seq=213085211 Ack=…
106	2022-11-15 04:11:21.731145464	10.9.0.11	10.9.0.5	UDP	114	9090 → 42627 Len=70
107	2022-11-15 04:11:21.731151412	10.9.0.11	10.9.0.5	UDP	114	9090 → 42627 Len=70
108	2022-11-15 04:11:21.732535919	10.9.0.5	10.9.0.11	UDP	105	42627 → 9090 Len=61
109	2022-11-15 04:11:21.732543162	10.9.0.5	10.9.0.11	UDP	105	42627 → 9090 Len=61
110	2022-11-15 04:11:21.733349054	192.168.53.99	192.168.60.5	TELNET	77	Telnet Data ...
111	2022-11-15 04:11:21.733355311	192.168.53.99	192.168.60.5	TCP	77	[TCP Retransmission] 56352 → 23 [PSH, ACK] Seq=3796404640 Ack…
112	2022-11-15 04:11:21.733368114	192.168.60.5	192.168.53.99	TCP	68	23 → 56352 [ACK] Seq=213085229 Ack=3796404649 Win=65152 Len=0…
113	2022-11-15 04:11:21.733371155	192.168.60.5	192.168.53.99	TCP	68	[TCP Dup ACK 112#1] 23 → 56352 [ACK] Seq=213085229 Ack=379640…
114	2022-11-15 04:11:21.733561309	10.9.0.11	10.9.0.5	UDP	96	9090 → 42627 Len=52
115	2022-11-15 04:11:21.733564897	10.9.0.11	10.9.0.5	UDP	96	9090 → 42627 Len=52
116	2022-11-15 04:11:21.735693343	10.9.0.5	10.9.0.11	UDP	96	42627 → 9090 Len=52
117	2022-11-15 04:11:21.735698144	10.9.0.5	10.9.0.11	UDP	96	42627 → 9090 Len=52
118	2022-11-15 04:11:21.736289642	192.168.53.99	192.168.60.5	TCP	68	56352 → 23 [ACK] Seq=3796404649 Ack=213085229 Win=64256 Len=0…
119	2022-11-15 04:11:21.736294801	192.168.53.99	192.168.60.5	TCP	68	[TCP Dup ACK 118#1] 56352 → 23 [ACK] Seq=3796404649 Ack=21308…
120	2022-11-15 04:11:21.737879843	10.9.0.5	10.9.0.11	UDP	130	42627 → 9090 Len=86
121	2022-11-15 04:11:21.737884979	10.9.0.5	10.9.0.11	UDP	130	42627 → 9090 Len=86
122	2022-11-15 04:11:21.738435953	192.168.53.99	192.168.60.5	TELNET	102	Telnet Data ...
123	2022-11-15 04:11:21.738440993	192.168.53.99	192.168.60.5	TCP	102	[TCP Retransmission] 56352 → 23 [PSH, ACK] Seq=3796404649 Ack…
124	2022-11-15 04:11:21.738490608	192.168.60.5	192.168.53.99	TCP	68	23 → 56352 [ACK] Seq=213085229 Ack=3796404683 Win=65152 Len=0…
125	2022-11-15 04:11:21.738494407	192.168.60.5	192.168.53.99	TCP	68	[TCP Dup ACK 124#1] 23 → 56352 [ACK] Seq=213085229 Ack=379640…
126	2022-11-15 04:11:21.739087813	10.9.0.11	10.9.0.5	UDP	96	9090 → 42627 Len=52

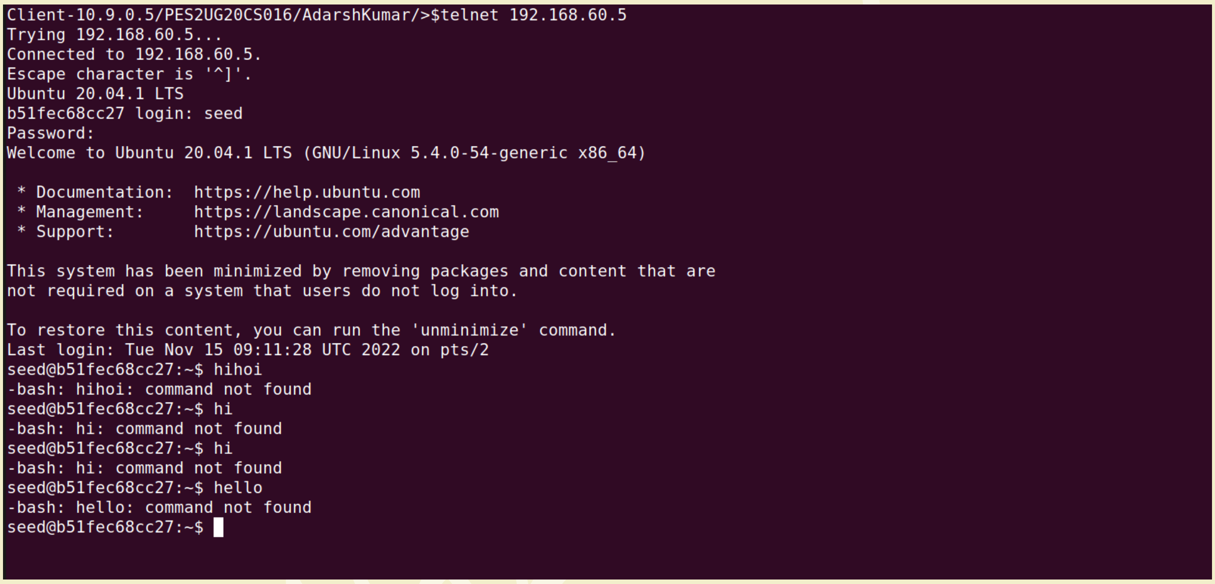
```
› Frame 81: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface any, id 0
```

wireshark_any_20221115041031_v4jdiQ.pcapng   Packets: 444 · Displayed: 444 (100.0%) · Dropped: 0

Description

In ping test, the icmp packet at Host U is target to 192.168.60.5, by the static route setup, the packet will route to interface tun0 and next hop address is 192.168.59.100. the packet will delivery to tun 0 and capture by the application tun_client.py. Then tun_client.py encapsulate the icmp packet with a udp packet and deliver to 10.9.0.11 with destination port 9090. As the application tun_server.py is started and the udp socket is listening at port 9090. The UDP packet to 10.9.0.11:9090 will captured by the tun_server.py and the application will decapsulate the UDP packet and extract its payload to become the IP packet. After that the application will pass the packet to tun0 right away. By the IP forward function is enabled at VPN server's kernel. The decapsulated packet will forward to 192.168.60.5 according to the destination address at ip header.

	Vice versa, the HOST V server receive the ICMP request and response with ICMP reply to VPN server. According to the routing table at VPN Server, the VPN Server receive an icmp reply packet with src ip 192.168.60.5 and destination 192.168.53.99. the VPN server will forward this packet to tun0 interface and capture by tun_server.py application. The application will encapsulate the packet with a UDP packet with destination ip 10.9.0.11 and destination port 9090 Via ens33 interface at VPN server. At next step, the ens33 interface at Host U will receive the UDP packet and the application has a udp socket that is listening at 9090 port. The application will receive the packet and decapsulate to the icmp reply packet. Finally, it passes back to tun0 interface and OS will pass to ping application with success result.
Task 6:	Tunnel-Breaking Experiment
Screenshot	Telnet to client to establish internet connection



```
Client-10.9.0.5/PES2UG20CS016/AdarshKumar/>$telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
b51fec68cc27 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 15 09:11:28 UTC 2022 on pts/2
seed@b51fec68cc27:~$ hihoi
-bash: hihoi: command not found
seed@b51fec68cc27:~$ hi
-bash: hi: command not found
seed@b51fec68cc27:~$ hi
-bash: hi: command not found
seed@b51fec68cc27:~$ hello
-bash: hello: command not found
seed@b51fec68cc27:~$ 
```

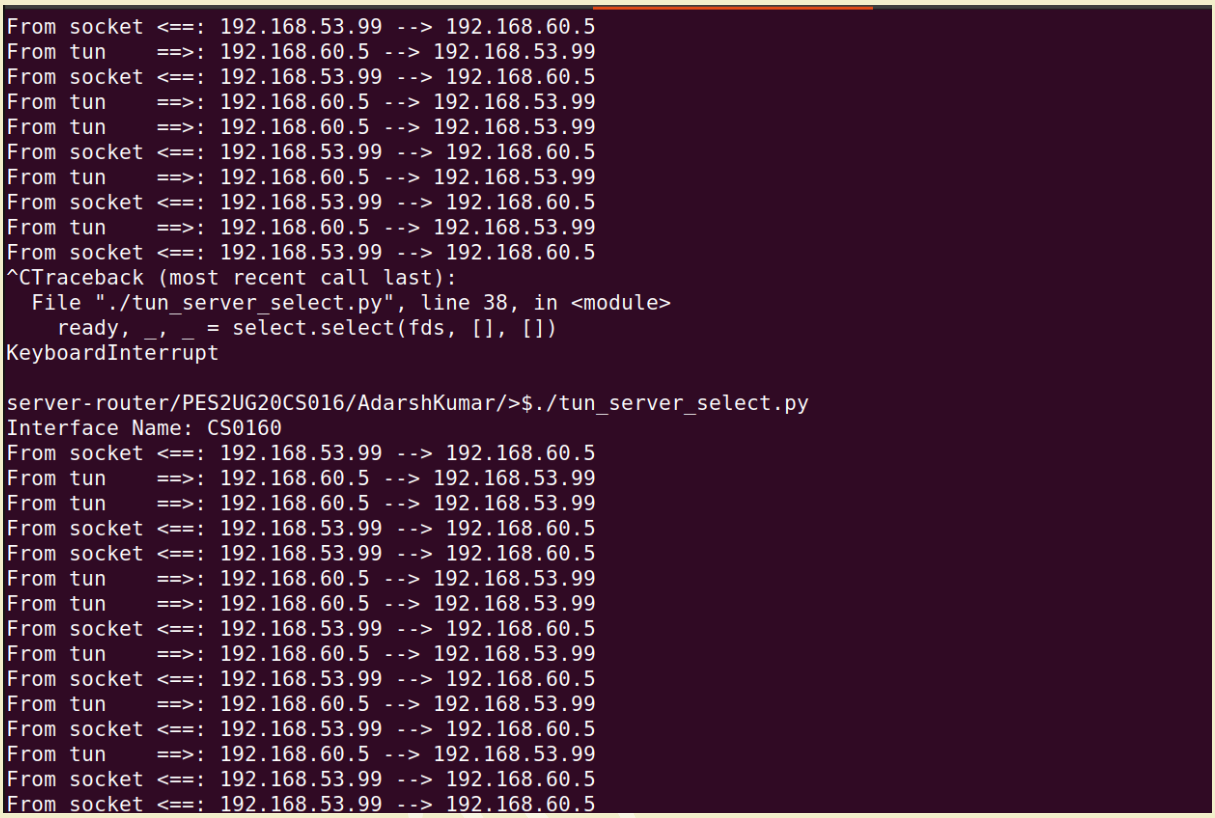
Do you see what you type? What happens to the TCP connection? Is the connection broken?

- We found that no matter what was entered, nothing was displayed. The TCP connection is not broken. the connection can resume if the disconnection time is not too long. The character typed after tunnel breaks can resume and send to the telent session. And telnet session is resumed without issue.
- From my understanding telnet use TCP as protocol, the packet send without ack from remote end will resend within certain time windows. That makes the connection persistent and recoverable even the UDP tunnel breaks. However, the disconnect time beyond the Retransmission timeout RTO. In RFC 1122, the recommendation is at least 100 seconds for the timeout, which corresponds to a value of at least 8, ubuntu default at 15. Another recommendation is at least 3 retransmissions, which is the default at ubuntu.

Once the tunnel is re-established, what is going to happen to the telnet connection? Please describe and explain your observations?

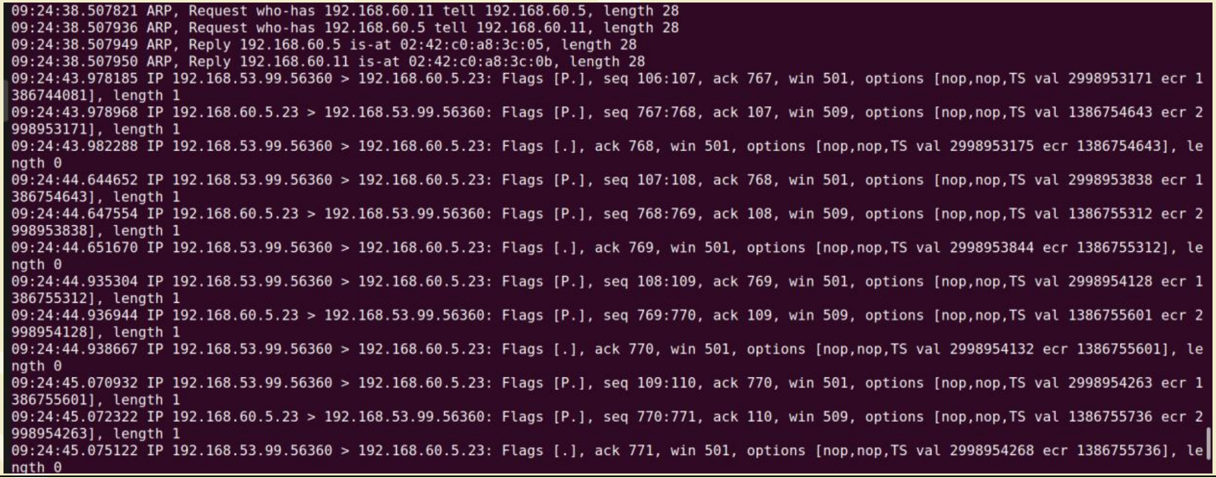
- While keeping the remote login online, the same situation occurs when the tunnel service of the client or server is broken, that is, no text can be entered in the remote login interface, and there will be no new output. When the service is reconnected for a short period of time, the backlogged packet buffers in the TUN file will be released one by one. As shown in the figure below:

On VPN Server-Router



```
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 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
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_server_select.py", line 38, in <module>
    ready, _, _ = select.select(fds, [], [])
KeyboardInterrupt

server-router/PES2UG20CS016/AdarshKumar/>$./tun_server_select.py
Interface Name: CS0160
From socket <==: 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 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 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
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
From socket <==: 192.168.53.99 --> 192.168.60.5
```

On the private network we can see that packets are also receiving



```
09:24:38.507821 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
09:24:38.507936 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
09:24:38.507949 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
09:24:38.507950 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
09:24:43.978185 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [P.], seq 106:107, ack 767, win 501, options [nop,nop,TS val 2998953171 ecr 1
386744081], length 1
09:24:43.978968 IP 192.168.60.5.23 > 192.168.53.99.56360: Flags [P.], seq 767:768, ack 107, win 509, options [nop,nop,TS val 1386754643 ecr 2
998953171], length 1
09:24:43.982288 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [.], ack 768, win 501, options [nop,nop,TS val 2998953175 ecr 1386754643], le
ngth 0
09:24:44.644652 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [P.], seq 107:108, ack 768, win 501, options [nop,nop,TS val 2998953838 ecr 1
386754643], length 1
09:24:44.647554 IP 192.168.60.5.23 > 192.168.53.99.56360: Flags [P.], seq 768:769, ack 108, win 509, options [nop,nop,TS val 1386755312 ecr 2
998953838], length 1
09:24:44.651670 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [.], ack 769, win 501, options [nop,nop,TS val 2998953844 ecr 1386755312], le
ngth 0
09:24:44.935304 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [P.], seq 108:109, ack 769, win 501, options [nop,nop,TS val 2998954128 ecr 1
386755312], length 1
09:24:44.936944 IP 192.168.60.5.23 > 192.168.53.99.56360: Flags [P.], seq 769:770, ack 109, win 509, options [nop,nop,TS val 1386755601 ecr 2
998954128], length 1
09:24:44.938667 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [.], ack 770, win 501, options [nop,nop,TS val 2998954132 ecr 1386755601], le
ngth 0
09:24:45.070932 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [P.], seq 109:110, ack 770, win 501, options [nop,nop,TS val 2998954263 ecr 1
386755601], length 1
09:24:45.072322 IP 192.168.60.5.23 > 192.168.53.99.56360: Flags [P.], seq 770:771, ack 110, win 509, options [nop,nop,TS val 1386755736 ecr 2
998954263], length 1
09:24:45.075122 IP 192.168.53.99.56360 > 192.168.60.5.23: Flags [.], ack 771, win 501, options [nop,nop,TS val 2998954268 ecr 1386755736], le
ngth 0
```

THE END