

CNS LAB

LAB - 6

Virtual Private Network Lab

NAME : SIRI S

SEMESTER : 5

SECTION :H

SRN : PES1UG19CS485

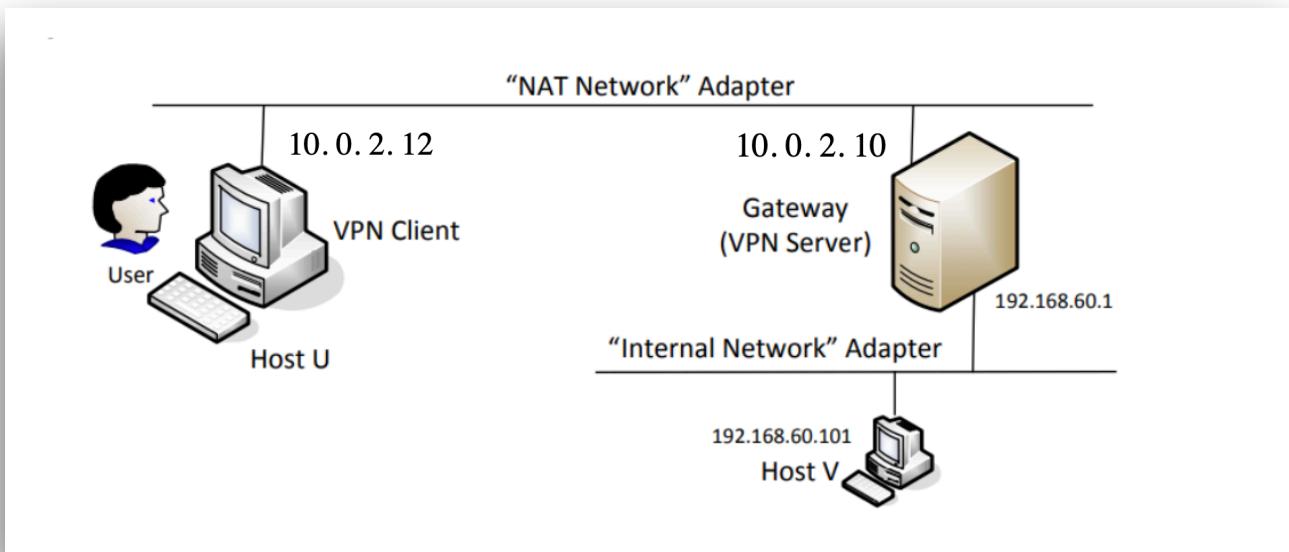
LAB SETUP:

VPN SERVER: 10. 0. 2. 10

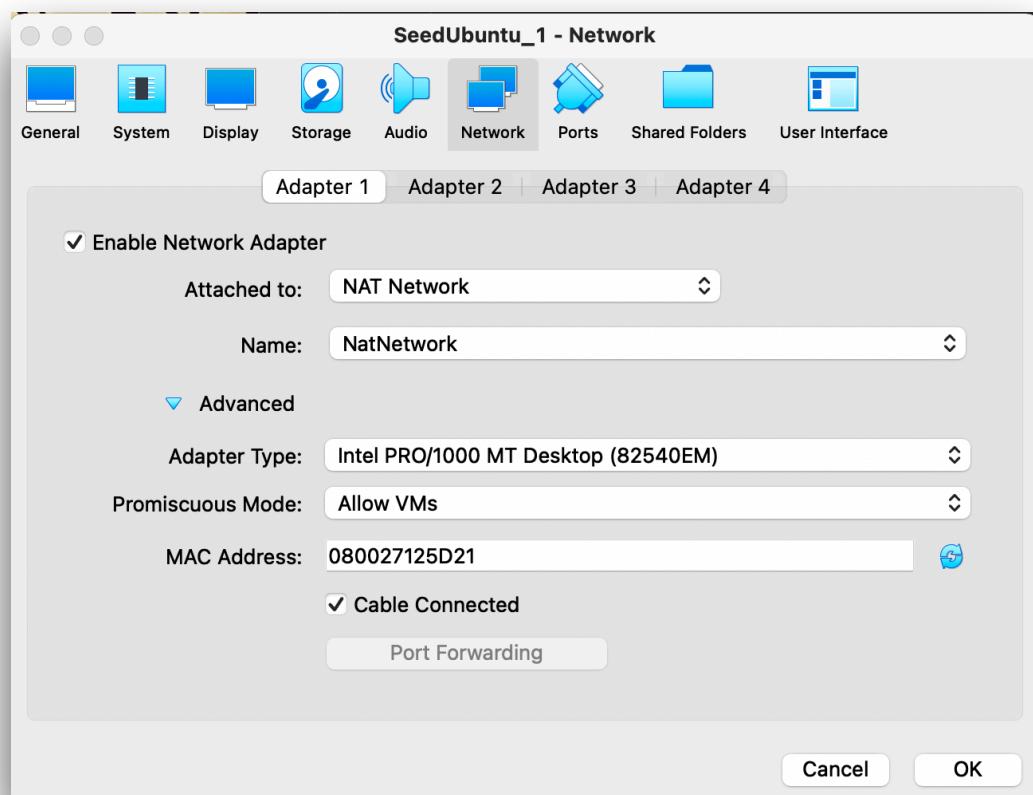
VPN CLIENT: 10. 0. 2. 12

VPN HOST: 10. 0. 2. 11

Task 1: VM Setup

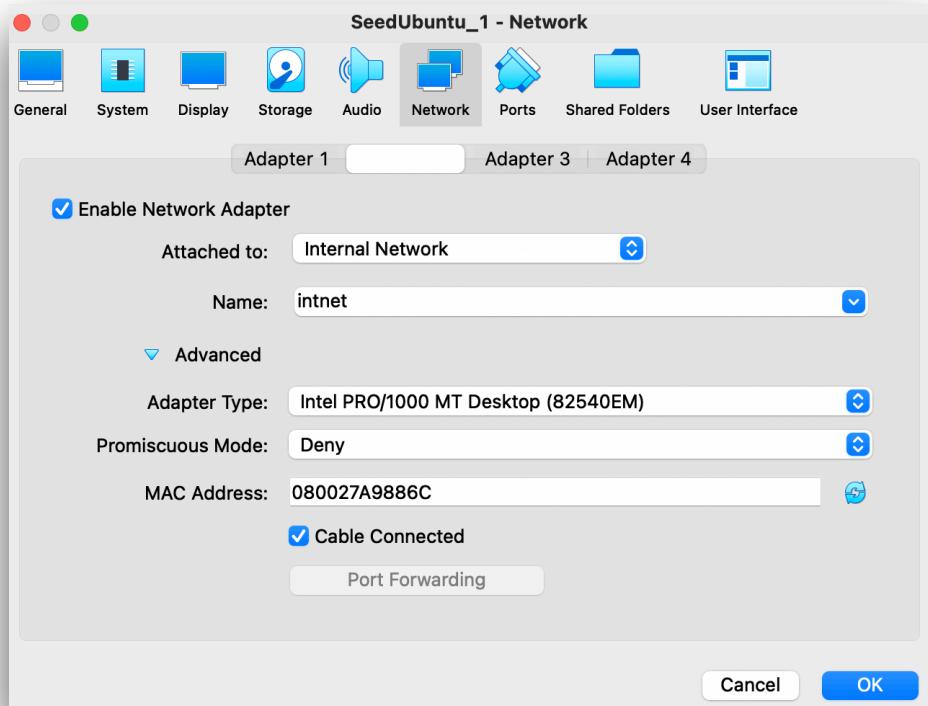


We need to establish a set up like above. In order to do this, the client and the server will have a NAT NETWORK as usual:



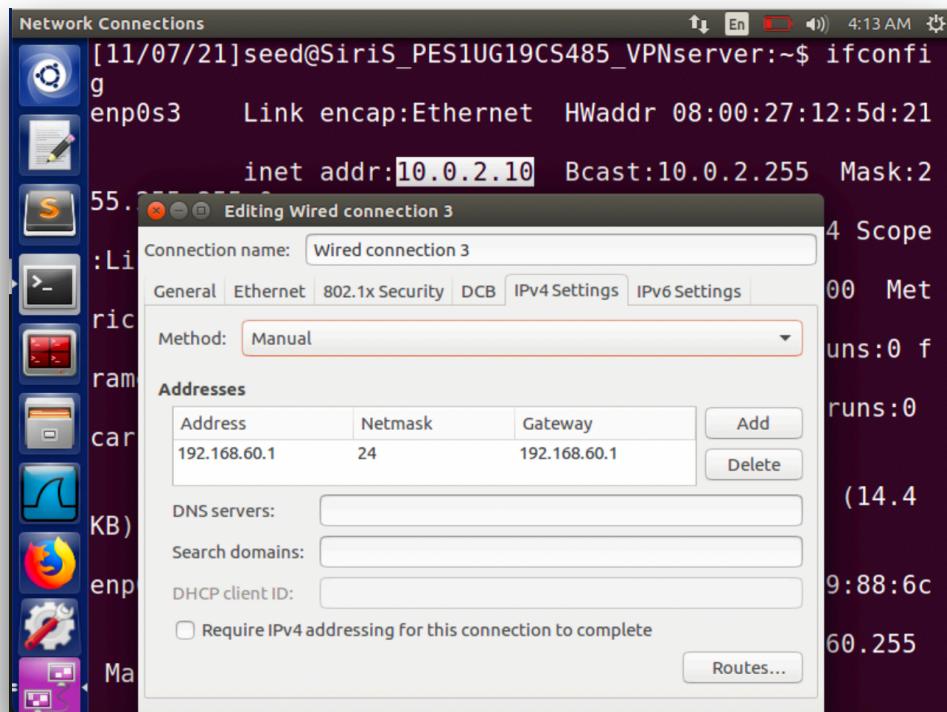
Screenshot of the Network settings in Server VM

However, the server and the host machines will be connected through an ‘internal network’ so that the client and the host have no connection:

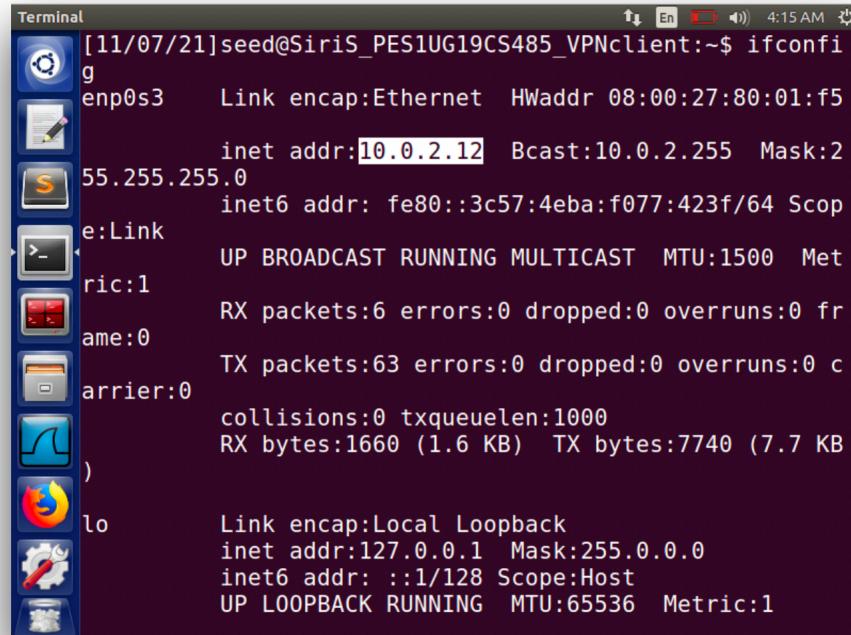


Screenshot of the Network settings in Server VM

Now we will make changes in the edit connections:

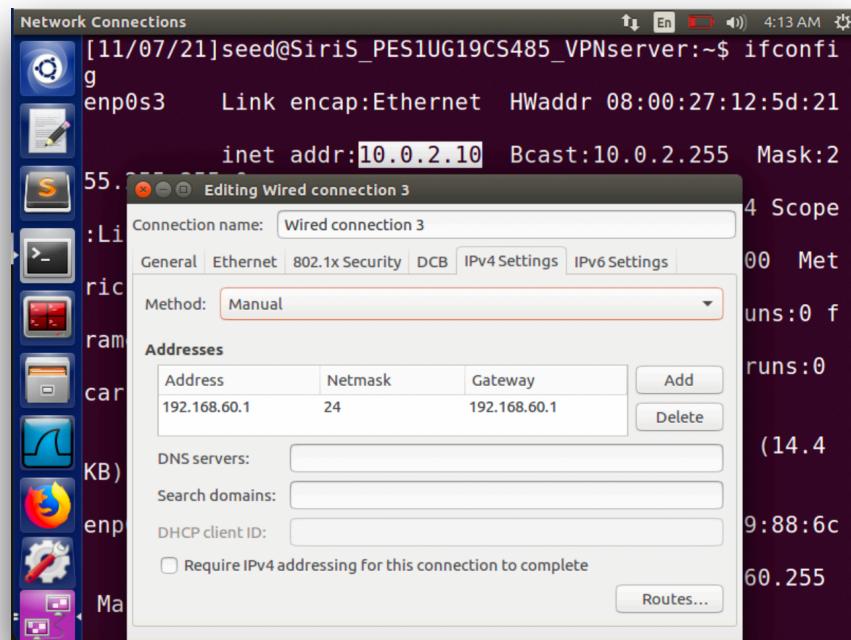


Server VM



```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:80:01:f5
              inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
              inet6 addr: fe80::3c57:4eba:f077:423f/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:6 errors:0 dropped:0 overruns:0 frame:0
              TX packets:63 errors:0 dropped:0 overruns:0 collisions:0
              RX bytes:1660 (1.6 KB)  TX bytes:7740 (7.7 KB)
lo          Link encap:Local Loopback
              inet addr:127.0.0.1  Mask:255.0.0.0
              inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

Client VM



Server VM

Therefore, from the above screenshots, we can see that:

VPN client – Adapter 1 – NAT Network

VPN Server – Adapter 1 – NAT network, Adapter 2 – Internal Network

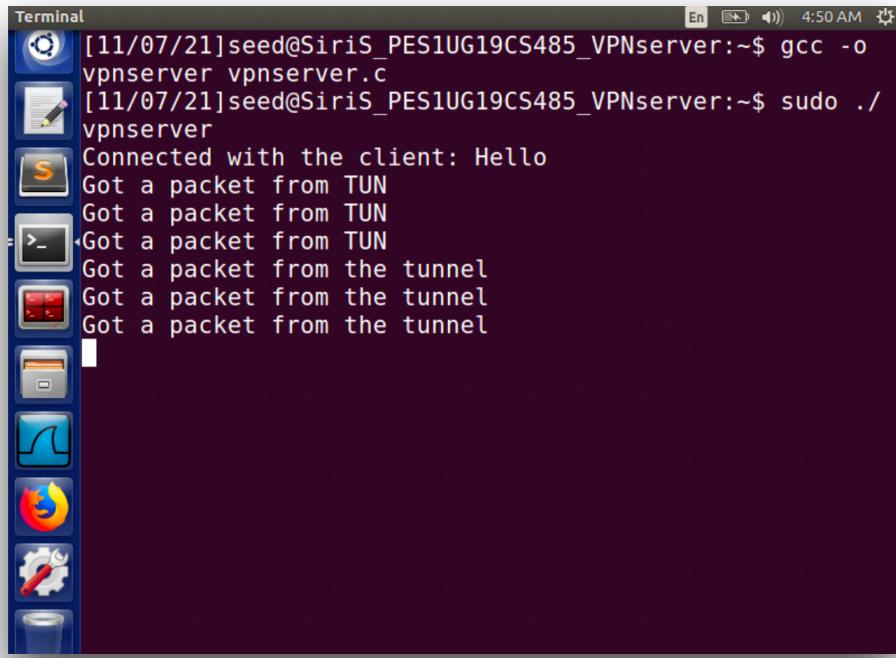
HOST V – Adapter 1 – Internal Network

These connections are established

Task 2: Creating a VPN Tunnel using TUN/TAP

Step 1: Run VPN server and set it's IP address of the interface

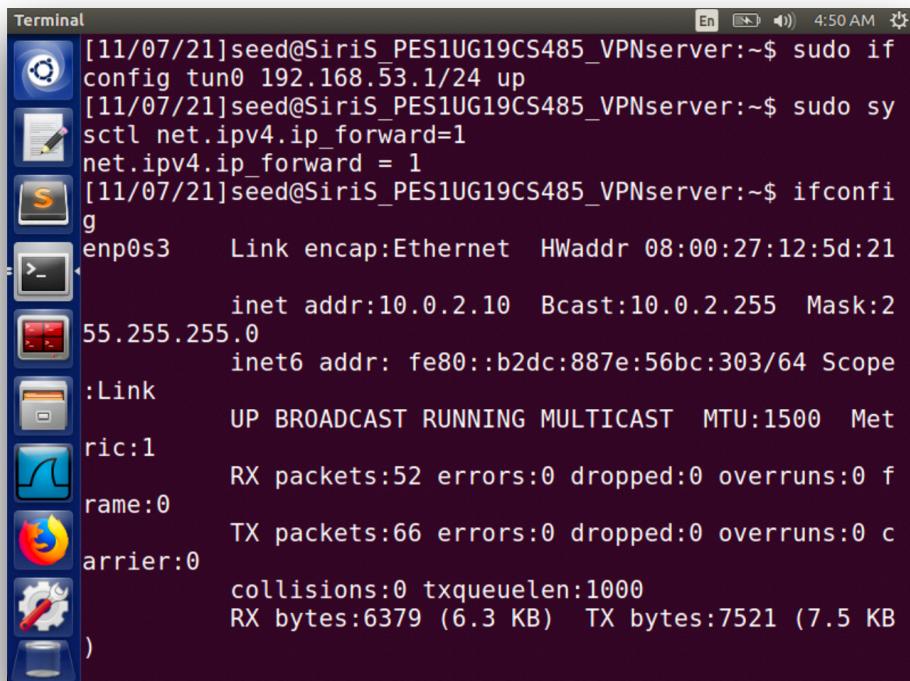
Now we run the vpnserver.c code on the server machine.

A screenshot of a Linux desktop environment, likely Ubuntu. On the left is a dock with icons for Terminal, Dash, Nautilus (file browser), System Settings, and Dash. The main window is a terminal window titled 'Terminal' with the command history:

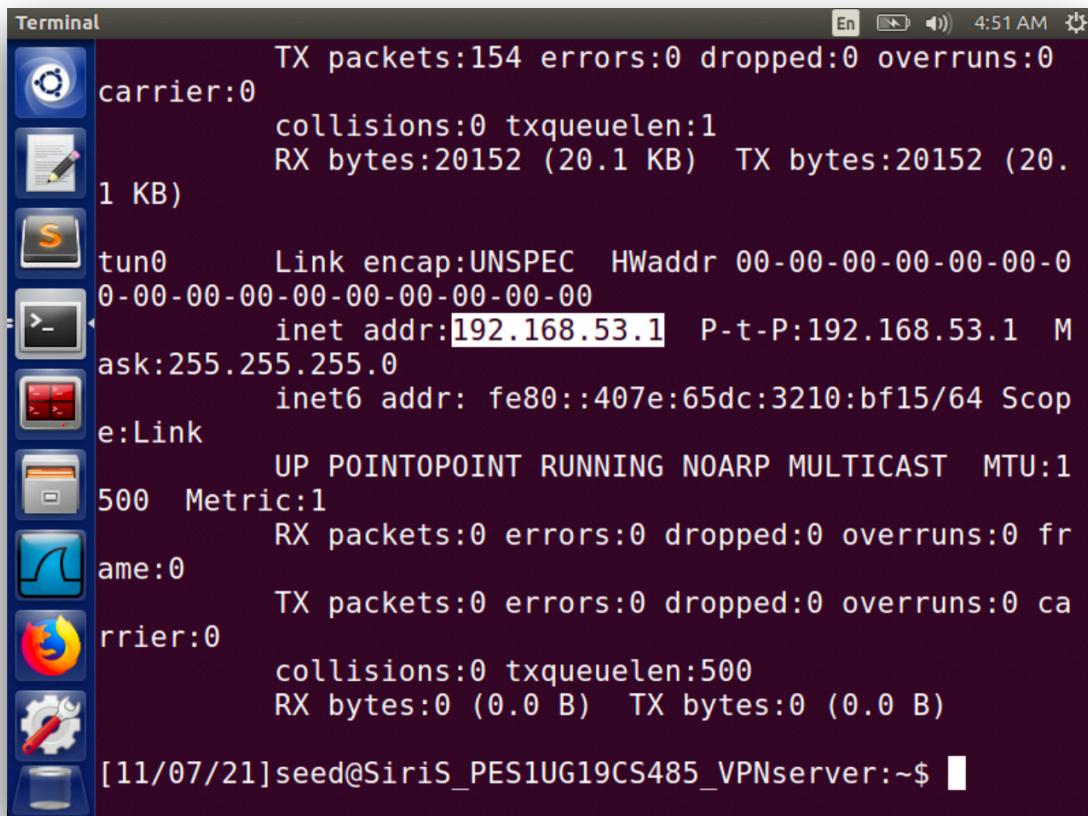
```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ gcc -o
vpnserver vpnserver.c
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ sudo ./vpnserver
Connected with the client: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

Then we assign an IP address to the tun0 interface and activate it.
IP Address assigned: 192.168.53.1/24. We also enable port forwarding.

Upon checking ifconfig -a : we have an established tunnel:

A screenshot of a Linux desktop environment, likely Ubuntu. On the left is a dock with icons for Terminal, Dash, Nautilus (file browser), System Settings, and Dash. The main window is a terminal window titled 'Terminal' with the command history:

```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ sudo if
config tun0 192.168.53.1/24 up
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ sudo sy
sctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:12:5d:21
              inet addr:10.0.2.10  Bcast:10.0.2.255  Mask:255.255.255.0
              inet6 addr: fe80::b2dc:887e:56bc:303/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:52 errors:0 dropped:0 overruns:0 frame:0
              TX packets:66 errors:0 dropped:0 overruns:0 collisions:0 txqueuelen:1000
              RX bytes:6379 (6.3 KB)  TX bytes:7521 (7.5 KB)
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal displays network interface statistics. The output shows two interfaces: "carrier:0" and "tun0". The "carrier:0" interface has TX packets: 154, errors: 0, dropped: 0, overruns: 0, collisions: 0, txqueuelen: 1, RX bytes: 20152 (20.1 KB), and TX bytes: 20152 (20.1 KB). The "tun0" interface has Link encap: UNSPEC, HWaddr 00-00-00-00-00-00, inet addr: 192.168.53.1, P-t-P: 192.168.53.1, Mask: 255.255.255.0, inet6 addr: fe80::407e:65dc:3210:bf15/64, Scop: e:Link, UP, POINTOPOINT, RUNNING, NOARP, MULTICAST, MTU: 1500, Metric: 1. The "tun0" interface also shows TX packets: 0, errors: 0, dropped: 0, overruns: 0, collisions: 0, txqueuelen: 500, RX bytes: 0 (0.0 B), and TX bytes: 0 (0.0 B). The terminal prompt is [11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~\$.

```
carrier:0      TX packets:154 errors:0 dropped:0 overruns:0
               collisions:0 txqueuelen:1
               RX bytes:20152 (20.1 KB)  TX bytes:20152 (20.
               1 KB)

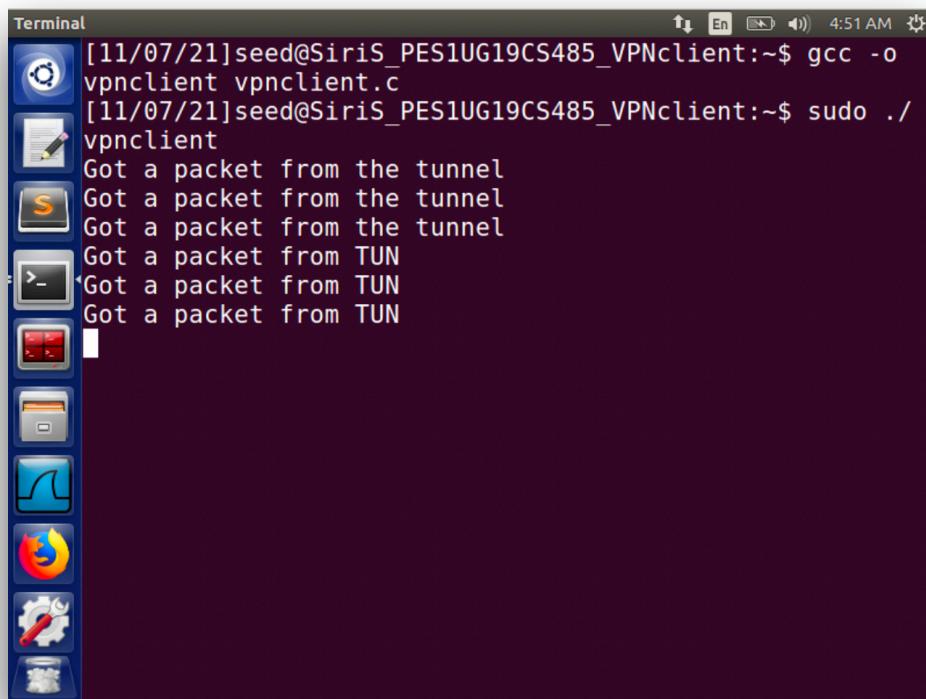
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet  addr:192.168.53.1  P-t-P:192.168.53.1  M
          Mask:255.255.255.0
          inet6 addr: fe80::407e:65dc:3210:bf15/64 Scop
          e:Link
          UP  POINTOPOINT RUNNING NOARP MULTICAST  MTU:1
          500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 fr
          ame:0      TX packets:0 errors:0 dropped:0 overruns:0 ca
          rrier:0      collisions:0 txqueuelen:500
                      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$
```

We can see that the tunnel is active. The VPN Server needs to forward packets to other destinations, so it needs to function as a gateway. We need to enable the IP forwarding for a computer to behave like a gateway.

Step 2: Run VPN Client and set IP address of the interface

Now we run the vpnclient.c code on the client machine.

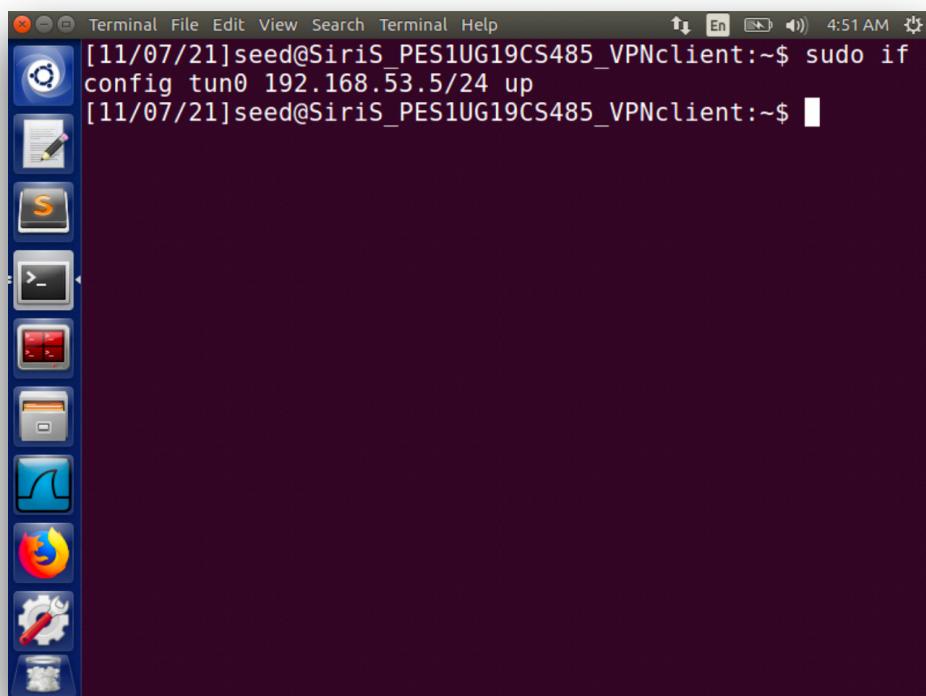
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal window has a dark background and contains the following text:

```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ gcc -o vpnclient vpnclient.c
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ sudo ./vpnclient
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

The terminal window is positioned over a desktop background with various icons for applications like the Dash, Home, and System settings.

Then we assign an IP address to the tun0 interface and activate it.

IP Address assigned: 192.168.53.5/24

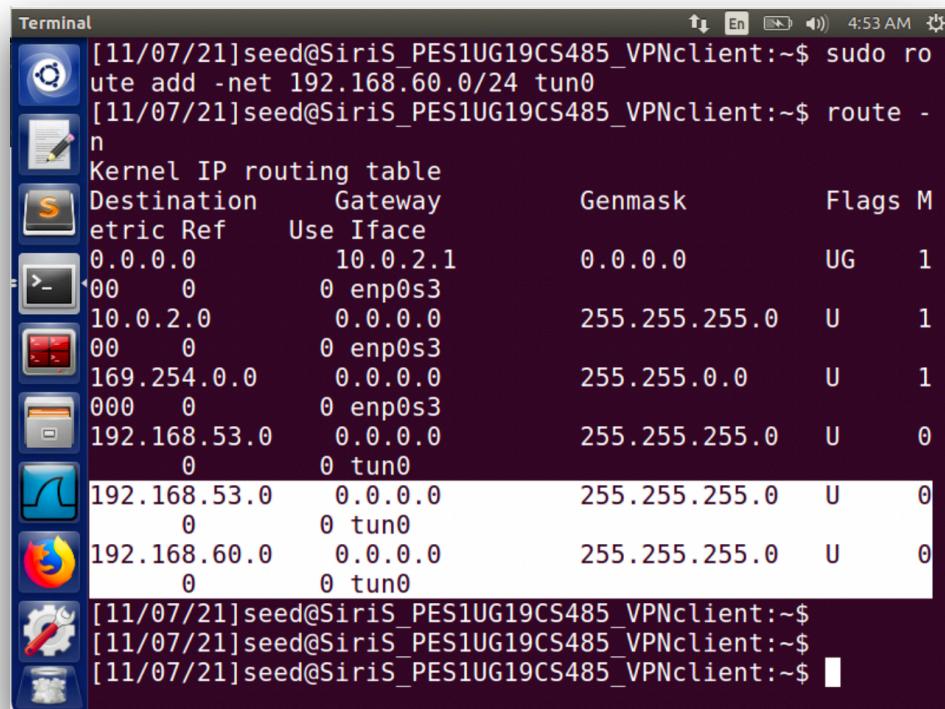
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal window has a dark background and contains the following text:

```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ sudo ifconfig tun0 192.168.53.5/24 up
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$
```

The terminal window is positioned over a desktop background with various icons for applications like the Dash, Home, and System settings.

Step 3: Set up routing on Client and Server VMs

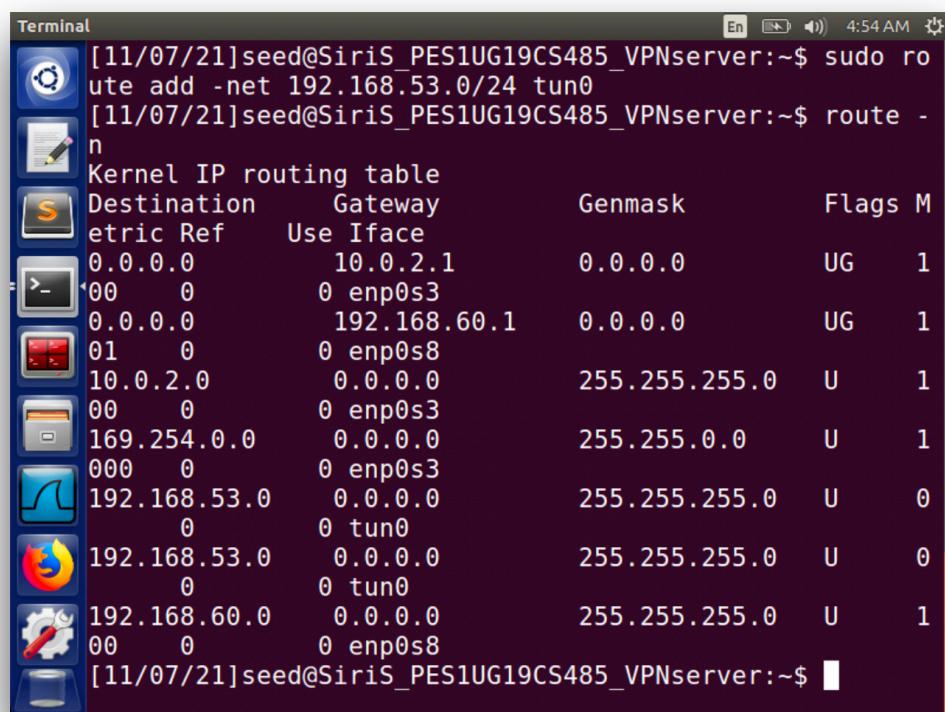
Setting up a routing table on the client VM:



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal displays the command "sudo route add -net 192.168.60.0/24 tun0" followed by the output of the "route -n" command. The output shows the kernel's IP routing table with several routes. One route is highlighted in yellow, indicating it was just added. The highlighted route is for network 192.168.60.0/24 via interface tun0. Other routes listed include 0.0.0.0/0 via enp0s3, 10.0.2.0/24 via 0.0.0.0, and various local loopback and bridge interfaces.

```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ sudo route add -net 192.168.60.0/24 tun0
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags M
          0.0.0.0      10.0.2.1    0.0.0.0       UG   1
          0.0.0.0      0.0.0.0     255.255.255.0  U     1
          10.0.2.0     0.0.0.0     255.255.0.0   U     1
          169.254.0.0   0.0.0.0     255.255.0.0   U     1
          192.168.53.0 0.0.0.0     255.255.255.0  U     0
          192.168.53.0 0.0.0.0     255.255.255.0  U     0
          192.168.60.0 0.0.0.0     255.255.255.0  U     0
          192.168.60.0 0.0.0.0     255.255.255.0  U     0
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ 
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ 
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ 
```

Setting up a routing table on the Server VM:

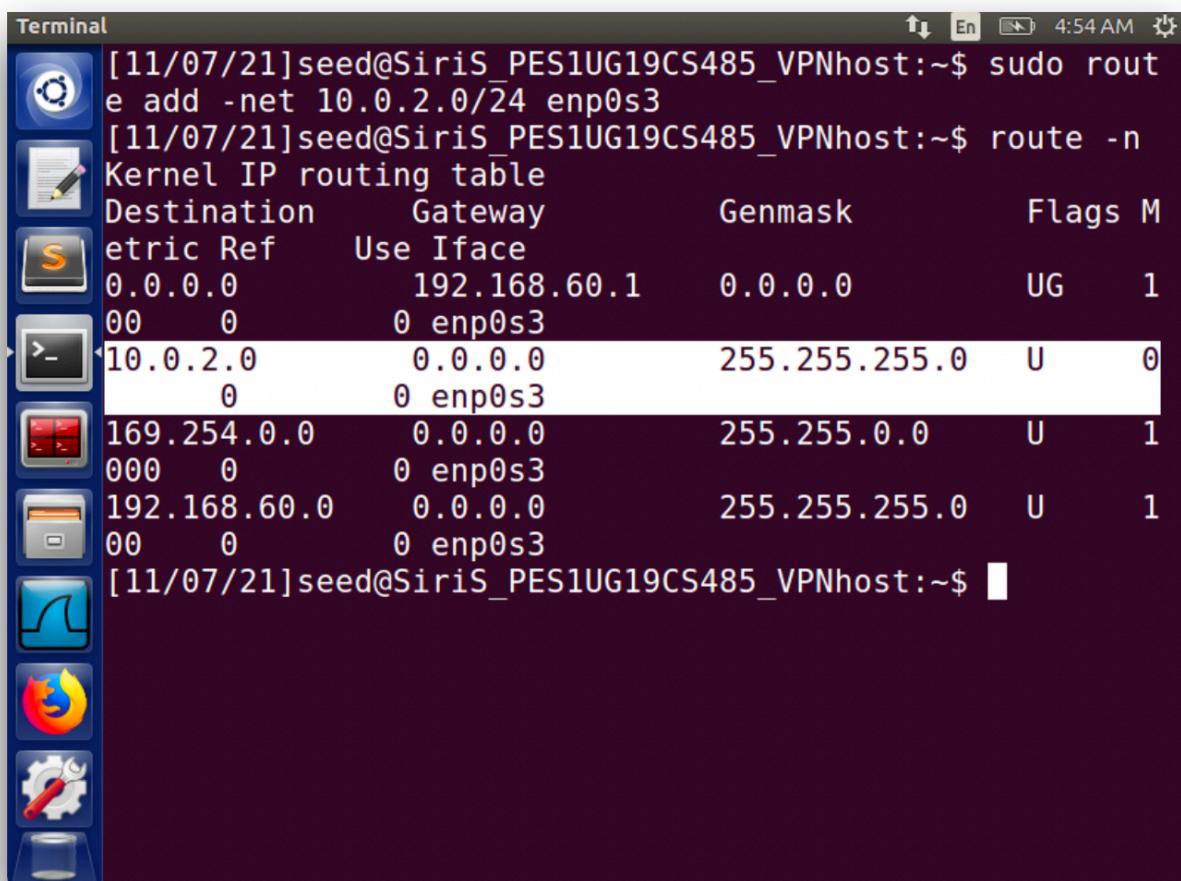


A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal displays the command "sudo route add -net 192.168.53.0/24 tun0" followed by the output of the "route -n" command. The output shows the kernel's IP routing table with several routes. One route is highlighted in yellow, indicating it was just added. The highlighted route is for network 192.168.53.0/24 via interface tun0. Other routes listed include 0.0.0.0/0 via enp0s3, 10.0.2.0/24 via 192.168.60.1, and various local loopback and bridge interfaces.

```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ sudo route add -net 192.168.53.0/24 tun0
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags M
          0.0.0.0      10.0.2.1    0.0.0.0       UG   1
          0.0.0.0      192.168.60.1  0.0.0.0       UG   1
          10.0.2.0     0.0.0.0     255.255.255.0  U     1
          169.254.0.0   0.0.0.0     255.255.0.0   U     1
          192.168.53.0 0.0.0.0     255.255.255.0  U     0
          192.168.53.0 0.0.0.0     255.255.255.0  U     0
          192.168.60.0 0.0.0.0     255.255.255.0  U     1
          192.168.60.0 0.0.0.0     255.255.255.0  U     1
[11/07/21]seed@SiriS_PES1UG19CS485_VPNserver:~$ 
```

Step 4: Set up routing on HOST V

Setting up a routing table on the Host VM:



The screenshot shows a terminal window titled "Terminal" with a dark background. At the top, there are icons for file, copy, paste, and settings, along with the date and time "4:54 AM". The terminal output is as follows:

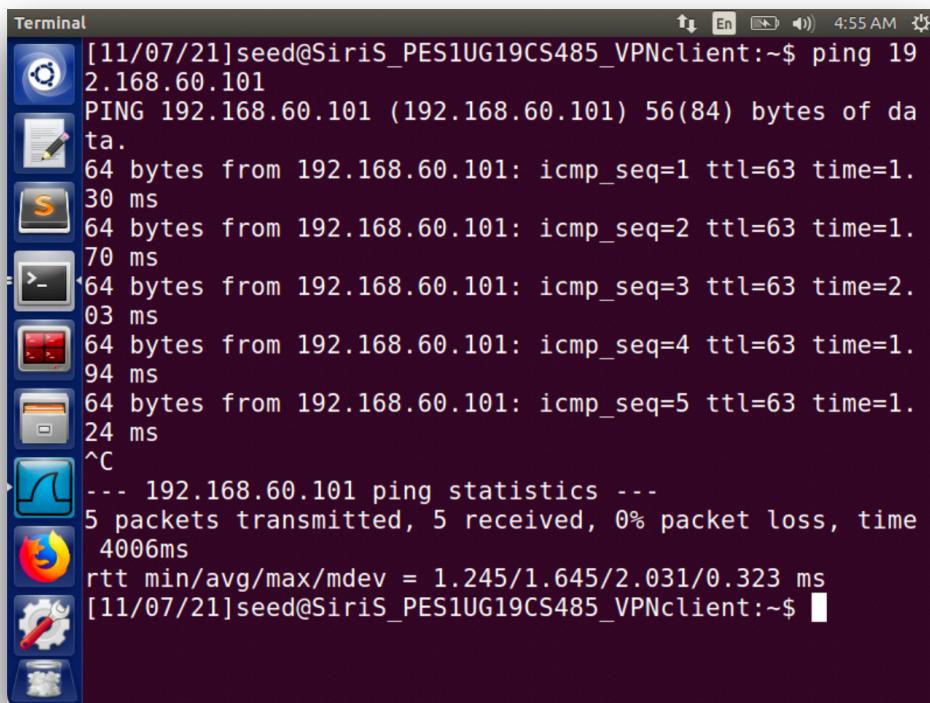
```
[11/07/21]seed@Siris_PES1UG19CS485_VPNhost:~$ sudo route add -net 10.0.2.0/24 enp0s3
[11/07/21]seed@Siris_PES1UG19CS485_VPNhost:~$ route -n
Kernel IP routing table
Destination      Gateway         Genmask        Flags M
Metric Ref      Use Iface
0.0.0.0          192.168.60.1   0.0.0.0       UG    1
00      0          0 enp0s3
10.0.2.0          0.0.0.0        255.255.255.0  U     0
          0          0 enp0s3
169.254.0.0       0.0.0.0        255.255.0.0    U     1
000     0          0 enp0s3
192.168.60.0      0.0.0.0        255.255.255.0  U     1
00      0          0 enp0s3
[11/07/21]seed@Siris_PES1UG19CS485_VPNhost:~$
```

The terminal window has a vertical toolbar on the left with icons for file, copy, paste, settings, and others.

In this step we set up the routing path on host V. This is so, that when host V replies to Host U it needs to go through the routing path through the VPN server. Since, Host V being an internal network does not know where to send the packets and cannot let the packets go through any arbitrary route else the packet is not protected.

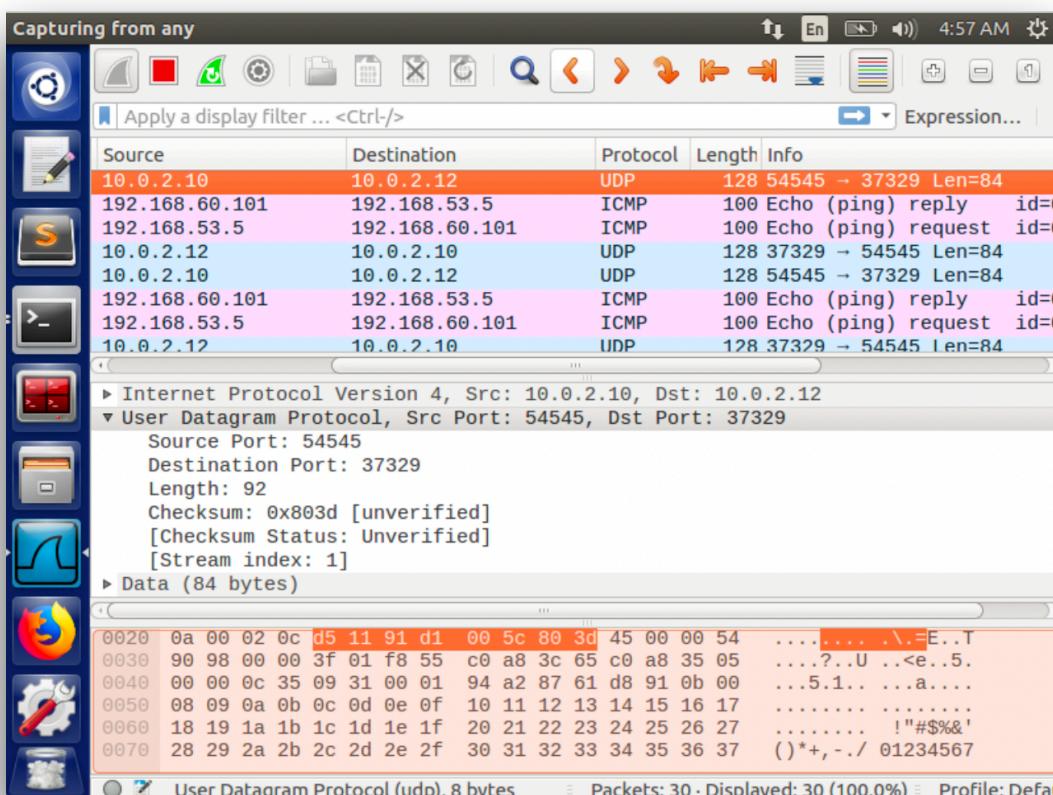
Step 5: Test the VPN tunnel (ping and telnet)

First we will perform the ping command to see if the VPN tunnel has been established:

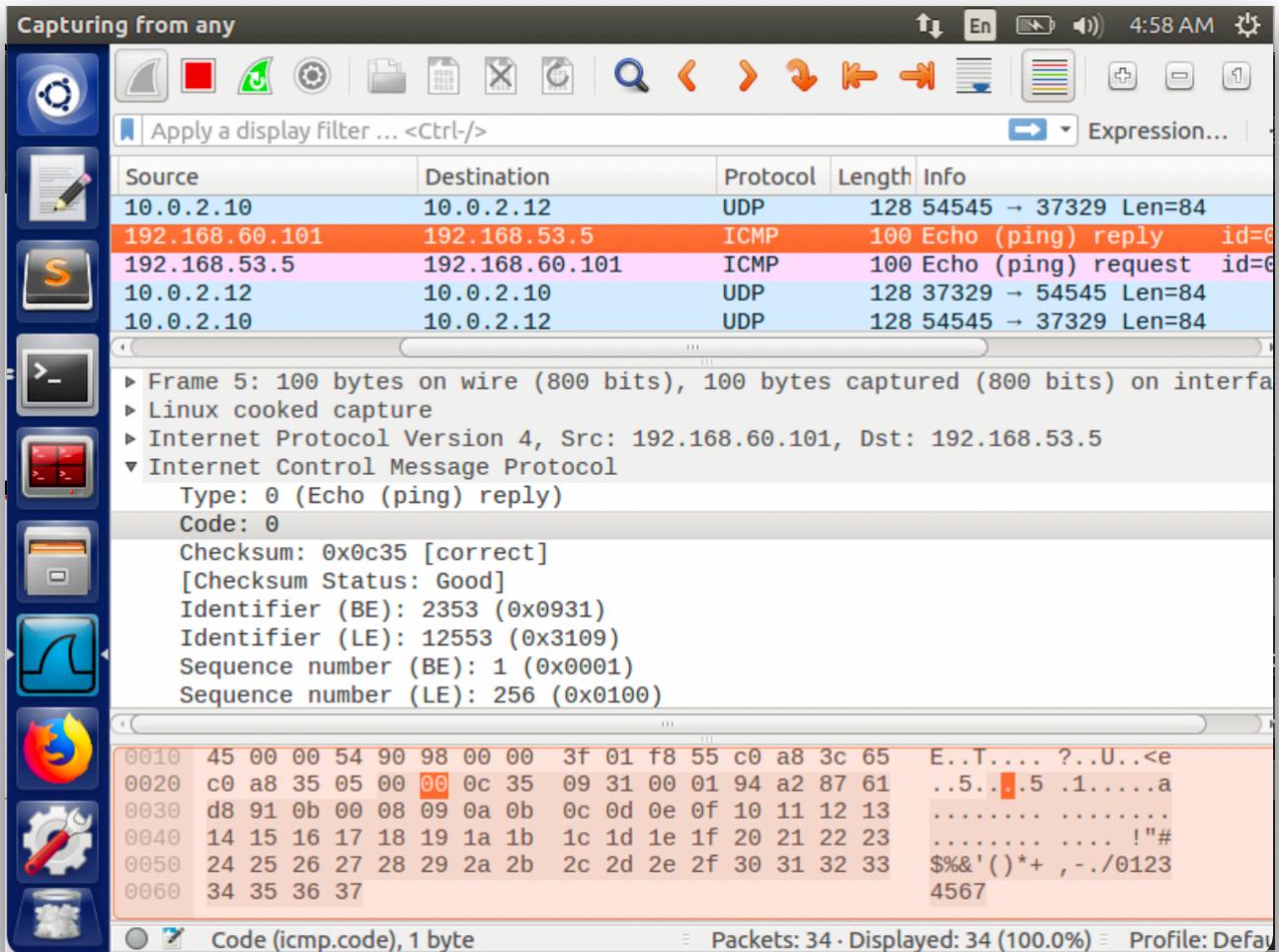


```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=1.30 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=1.70 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=2.03 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=1.94 ms
64 bytes from 192.168.60.101: icmp_seq=5 ttl=63 time=1.24 ms
^C
--- 192.168.60.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time
4006ms
rtt min/avg/max/mdev = 1.245/1.645/2.031/0.323 ms
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$
```

We are successfully able to ping as seen. The Wireshark screenshot explaining the UDP packet:

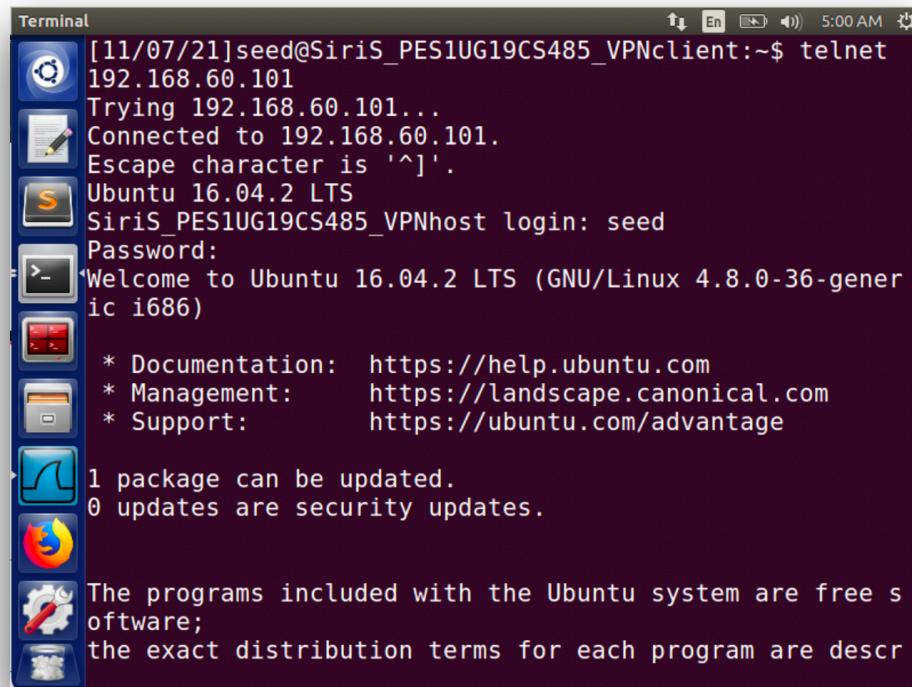


The Wireshark screenshot explaining the ICMP packet:



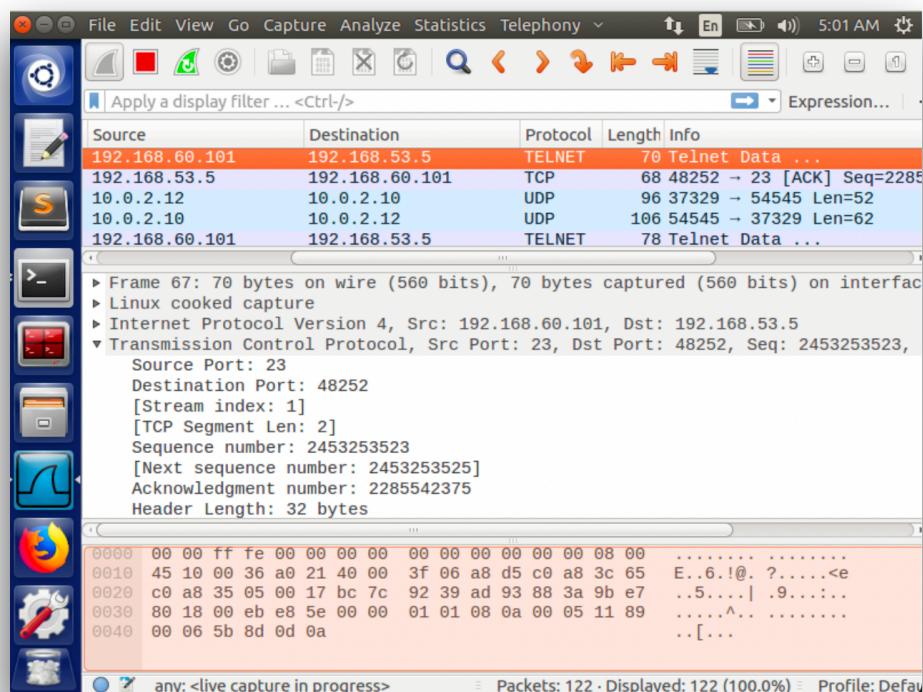
We can see from the wireshark screenshot that packet source and destination 192.168.53.5 (Client- tun0) and 192.168.60.101 (Host V) are the tunnel traffic and the rest are the normal traffic.

Next we will perform the telnet connection to verify if the VPN tunnel has been established:



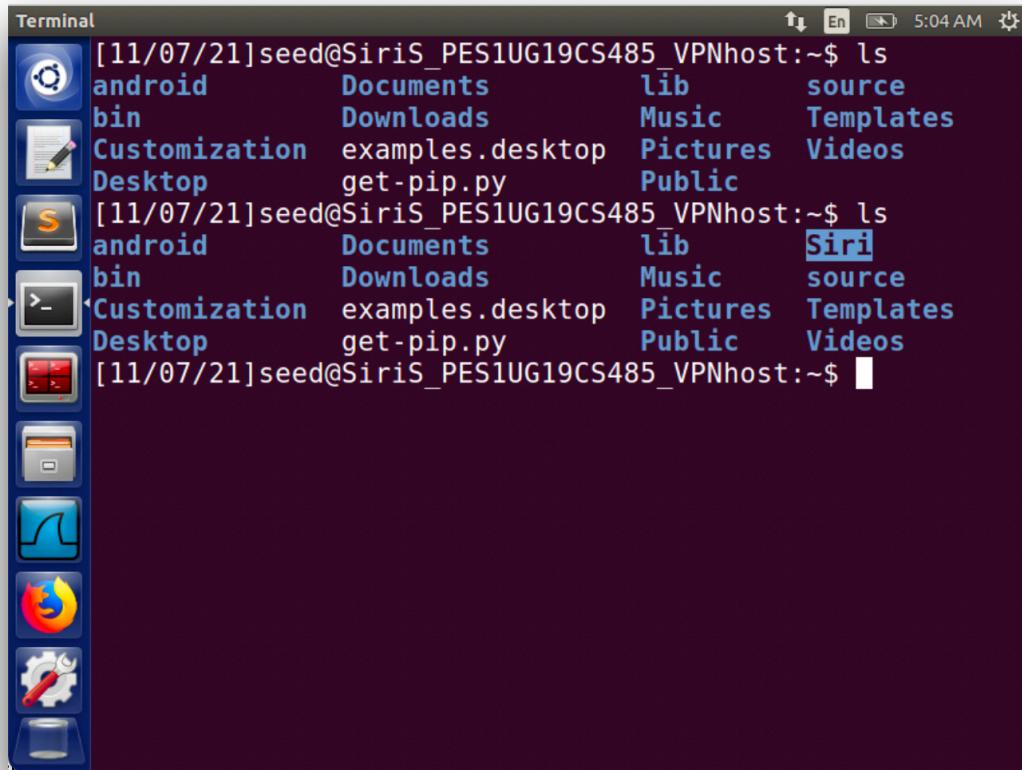
```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNclient:~$ telnet  
192.168.60.101  
Trying 192.168.60.101...  
Connected to 192.168.60.101.  
Escape character is '^]'.  
Ubuntu 16.04.2 LTS  
SiriS_PES1UG19CS485_VPNhost login: seed  
Password:  
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
1 package can be updated.  
0 updates are security updates.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are descr
```

We can see that we are successfully able to establish the telnet connection. Wireshark screenshot to prove it:



From the above screenshot we can conclude that we were successful in establishing the VPN connection.

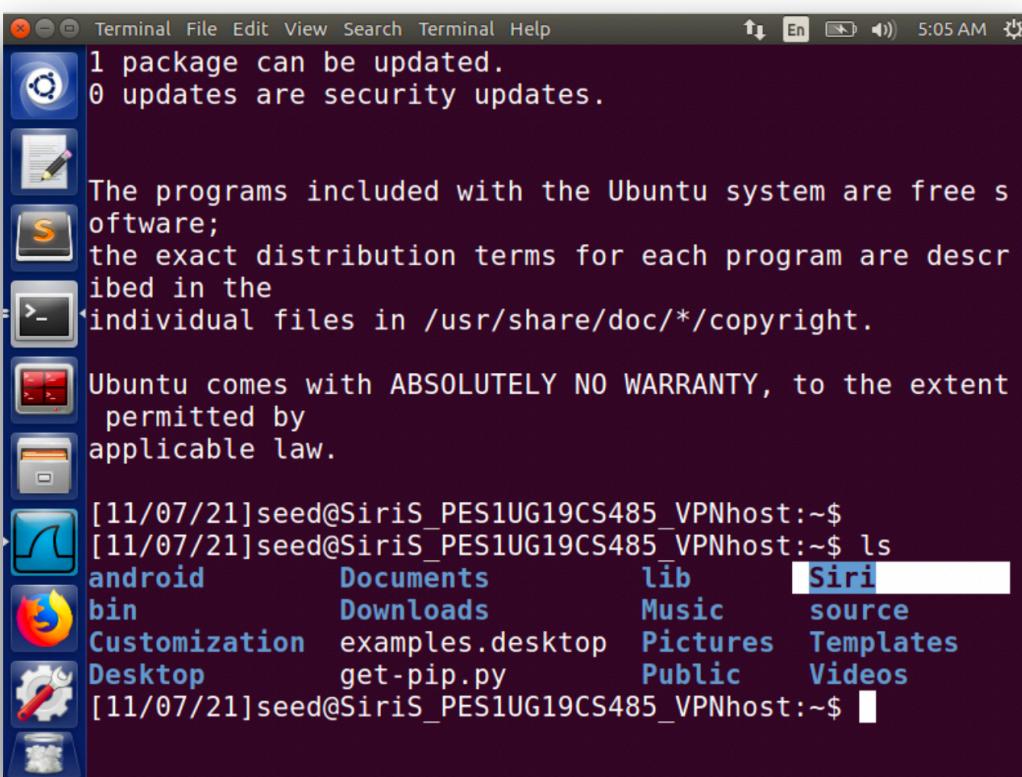
We run ‘ls’ command on the Host VM and in addition, create a new folder called ‘Siri’ as seen in the screenshot:



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal". The terminal content shows three separate "ls" commands being run at the prompt [11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~\$. The first two commands show standard directory contents: android, Documents, lib, source, bin, Downloads, Music, Templates, Customization, examples.desktop, Pictures, Videos, Desktop, and get-pip.py. The third command, run after a new folder "Siri" has been created, includes the "Siri" folder in the list. The desktop interface includes a dock with icons for various applications like a terminal, file manager, and browser.

```
[11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~$ ls
android      Documents      lib      source
bin          Downloads       Music    Templates
Customization examples.desktop Pictures  Videos
Desktop      get-pip.py     Public
[11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~$ ls
android      Documents      lib      Siri
bin          Downloads       Music    source
Customization examples.desktop Pictures  Templates
Desktop      get-pip.py     Public   Videos
[11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~$
```

Now when we run ‘ls’ command on the telnet connection, we are able to notice that the new folder create is visible:



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal". The terminal content shows the output of a "dpkg" command, indicating "1 package can be updated" and "0 updates are security updates". Below this, there is a block of text about the nature of the software and warranty. Finally, the terminal shows the result of an "ls" command at the prompt [11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~\$, which includes the newly created "Siri" folder. The desktop interface includes a dock with icons for various applications like a terminal, file manager, and browser.

```
1 package can be updated.
0 updates are security updates.

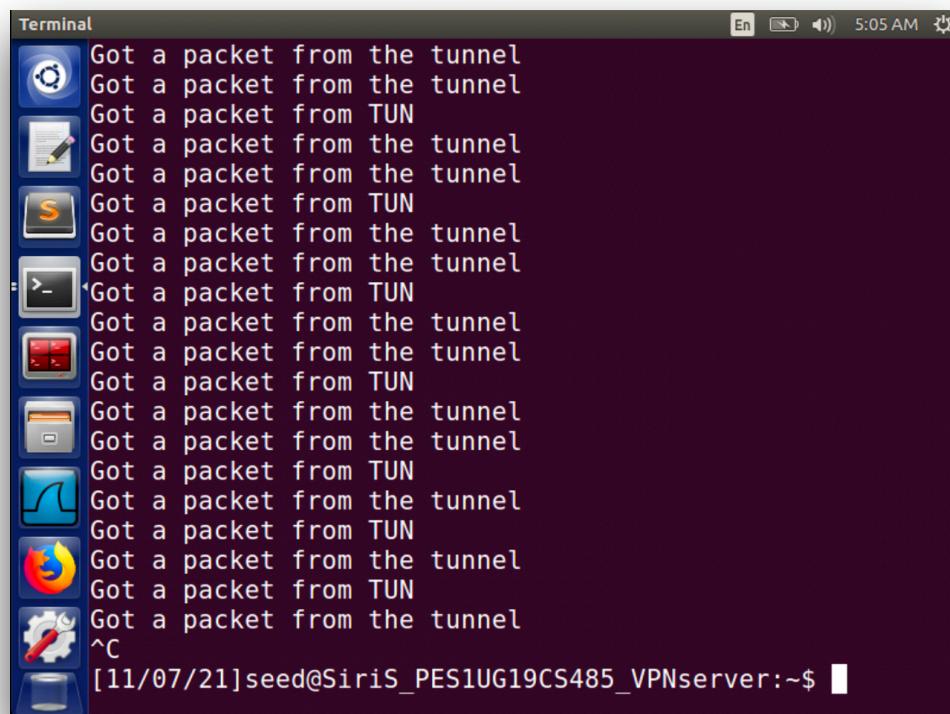
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

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

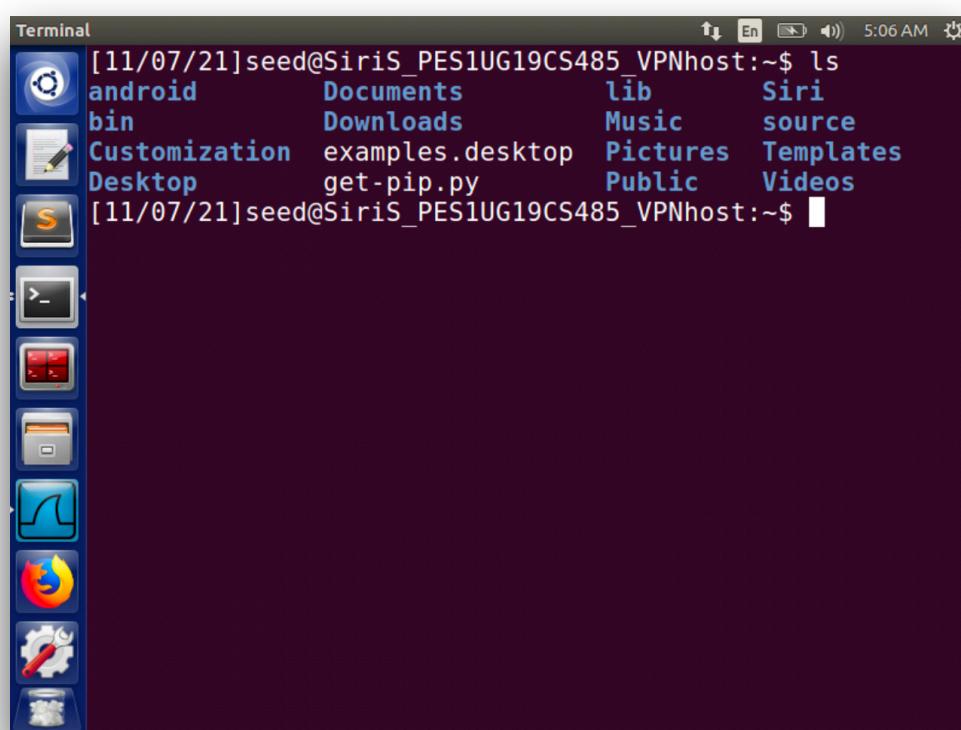
[11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~$ ls
android      Documents      lib      Siri
bin          Downloads       Music    source
Customization examples.desktop Pictures  Templates
Desktop      get-pip.py     Public   Videos
[11/07/21]seed@SiriS_PES1UG19CS485_VPNhost:~$
```

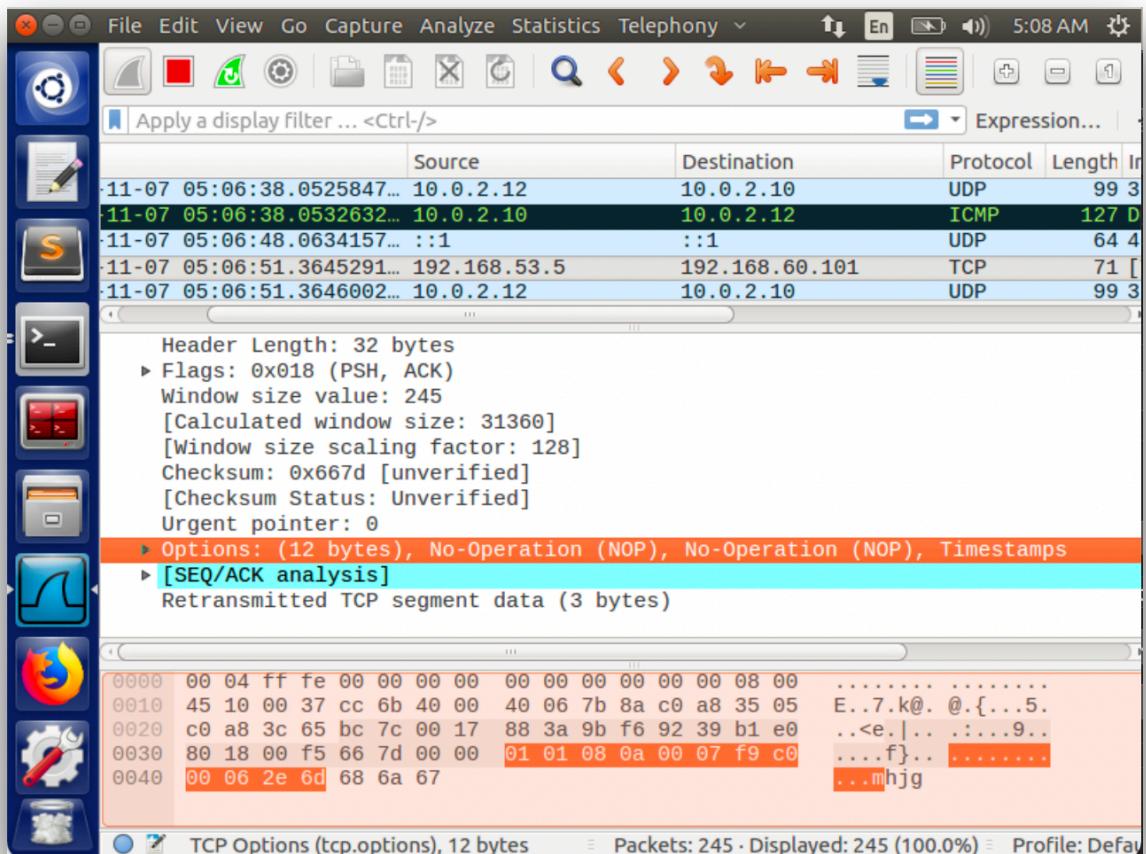
Step 6: Tunnel-Breaking Test

We disconnect the `vpnserver` program in order to break the VPN tunnel connection as shown in the screenshot:



Now when we try to type ls in the telnet connection, we unable to do so:





As we can see from the Wireshark screenshot, we are getting a TCP redirect message.

And now if we again connect the server machine we can see that the randomly typed command on host U machine is popped up.

The telnet becomes unresponsive when we break the connection and then after reconnecting all the typed characters we had given reaches the telnet server which we are later able to see on VPN client.