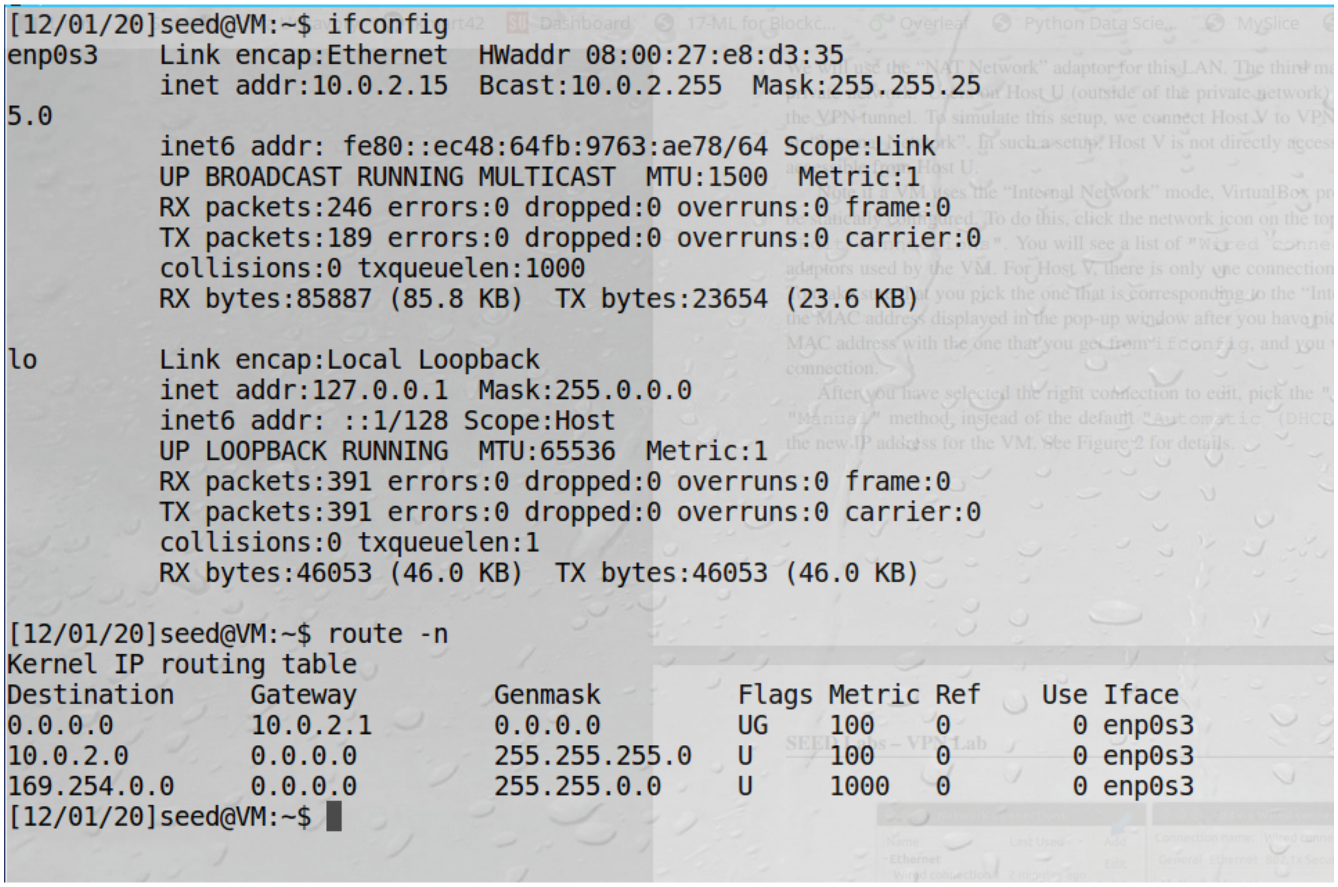
Lab 7 – Kevin Martin

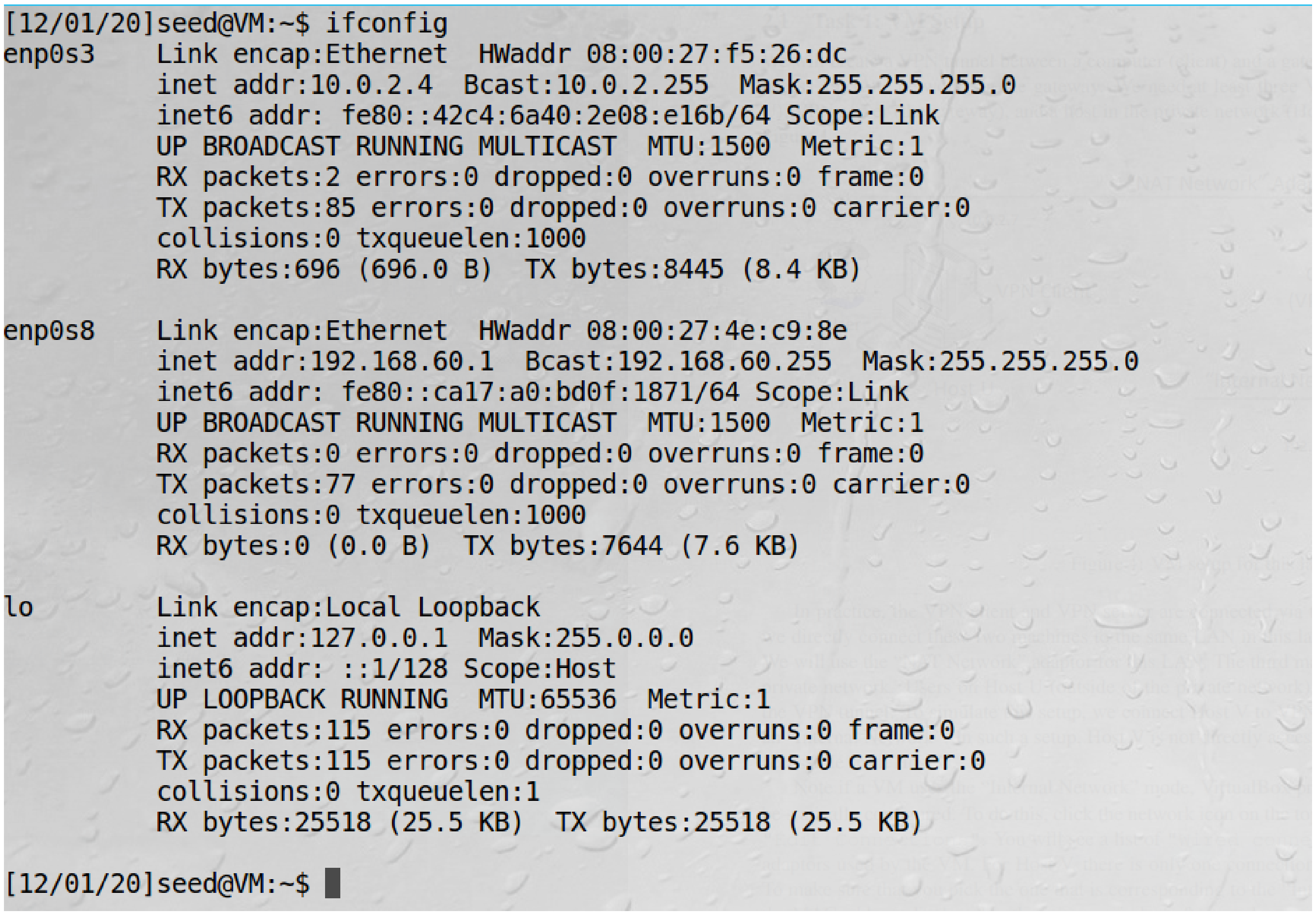
Task 1 – VM Setup

To set up the VPN, first we have to configure the three machines. Host U will be VM A, the VPN Server/Gateway will be VM C, and the destination/Host V will be VM C.

Host U (VM A), the NAT Network and IP 10.0.2.15:


```
[12/01/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:e8:d3:35
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.25
5.0
          inet6 addr: fe80::ec48:64fb:9763:ae78/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:246 errors:0 dropped:0 overruns:0 frame:0
          TX packets:189 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:85887 (85.8 KB)  TX bytes:23654 (23.6 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
          RX packets:391 errors:0 dropped:0 overruns:0 frame:0
          TX packets:391 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:46053 (46.0 KB)  TX bytes:46053 (46.0 KB)

[12/01/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
[12/01/20]seed@VM:~$
```

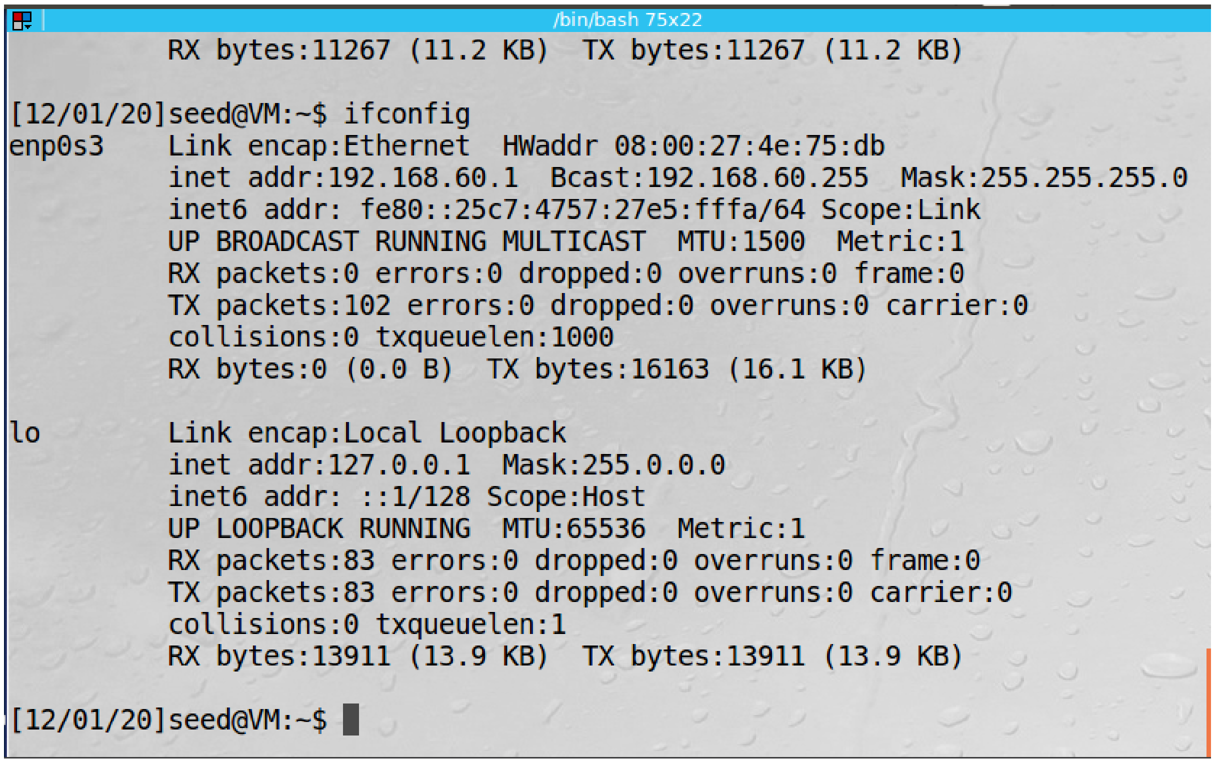
VPN Server (VM B) needs both the shared ethernet network as well as the statically configured connection. This is done through the ip4 settings of the VM. However, the **internal network** had to be configured outside of the VM first, then it was recognized in the booted VM. Once correctly added, the output is can be seen in the **ifconfig** screenshot:



```
[12/01/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:f5:26:dc
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::42c4:6a40:2e08:e16b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:696 (696.0 B)  TX bytes:8445 (8.4 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:4e:c9:8e
          inet addr:192.168.60.1  Bcast:192.168.60.255  Mask:255.255.255.0
          inet6 addr: fe80::ca17:a0:bd0f:1871/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:7644 (7.6 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
          RX packets:115 errors:0 dropped:0 overruns:0 frame:0
          TX packets:115 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:25518 (25.5 KB)  TX bytes:25518 (25.5 KB)

[12/01/20]seed@VM:~$ 
```

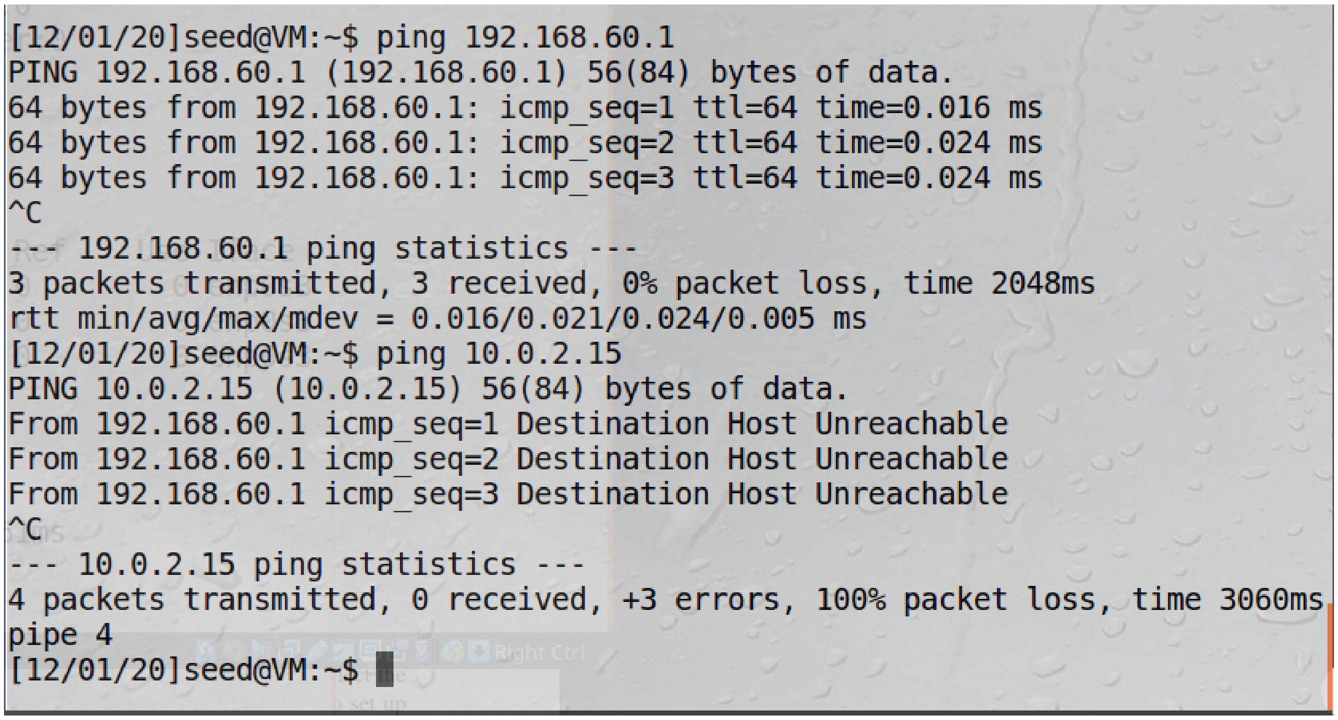
Note the VM's IP address as 10.0.2.4, as it has always been, and the new connection at 192.168.60.1.

Finally, Host V (VM C) is configured to ONLY be on the internal network. Again, this was first done outside the VM, and is now recognized inside the booted instance via **ifconfig**:



```
/bin/bash 75x22
          RX bytes:11267 (11.2 KB)  TX bytes:11267 (11.2 KB)

[12/01/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:4e:75:db
          inet addr:192.168.60.1  Bcast:192.168.60.255  Mask:255.255.255.0
          inet6 addr: fe80::25c7:4757:27e5:fffa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:102 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:16163 (16.1 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
          RX packets:83 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:13911 (13.9 KB)  TX bytes:13911 (13.9 KB)

[12/01/20]seed@VM:~$ 
```

From Host V (VM C), we can only ping the VPN server (VM B) but not Host U (VM A). Before modifying the connections, Host V would have been reachable:



```
[12/01/20]seed@VM:~$ ping 192.168.60.1
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
64 bytes from 192.168.60.1: icmp_seq=1 ttl=64 time=0.016 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.024 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.024 ms
^C
--- 192.168.60.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.016/0.021/0.024/0.005 ms
[12/01/20]seed@VM:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
From 192.168.60.1 icmp_seq=1 Destination Host Unreachable
From 192.168.60.1 icmp_seq=2 Destination Host Unreachable
From 192.168.60.1 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.2.15 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3060ms
pipe 4
[12/01/20]seed@VM:~$ 
```

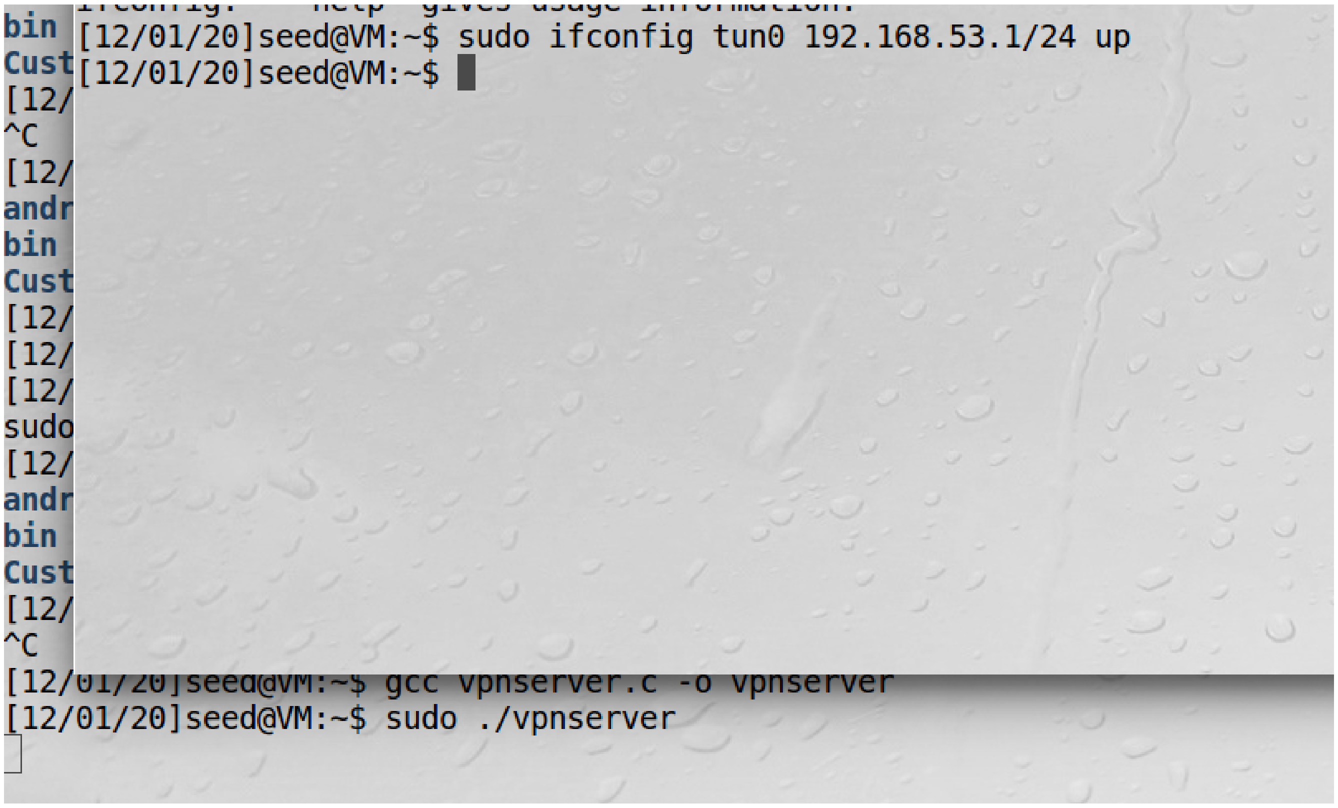
Observation: We set up a secondary "private" connection between the VPN Server and Host V. This network is not accessible for Host U. The VPN Server has connections to both the normal NAT

Network as well as this newly established private network. Host V and Host U are, at the moment, unable to communicate with each other.

Explanation: The new private network allows connection between VPN Server and Host V to simulate a private network for something like a company's internal network. Host U is simulating the user and will try to connect to V using the VPN tunnel.

Task 2 – Creating a VPN Tunnel using TUN/TAP

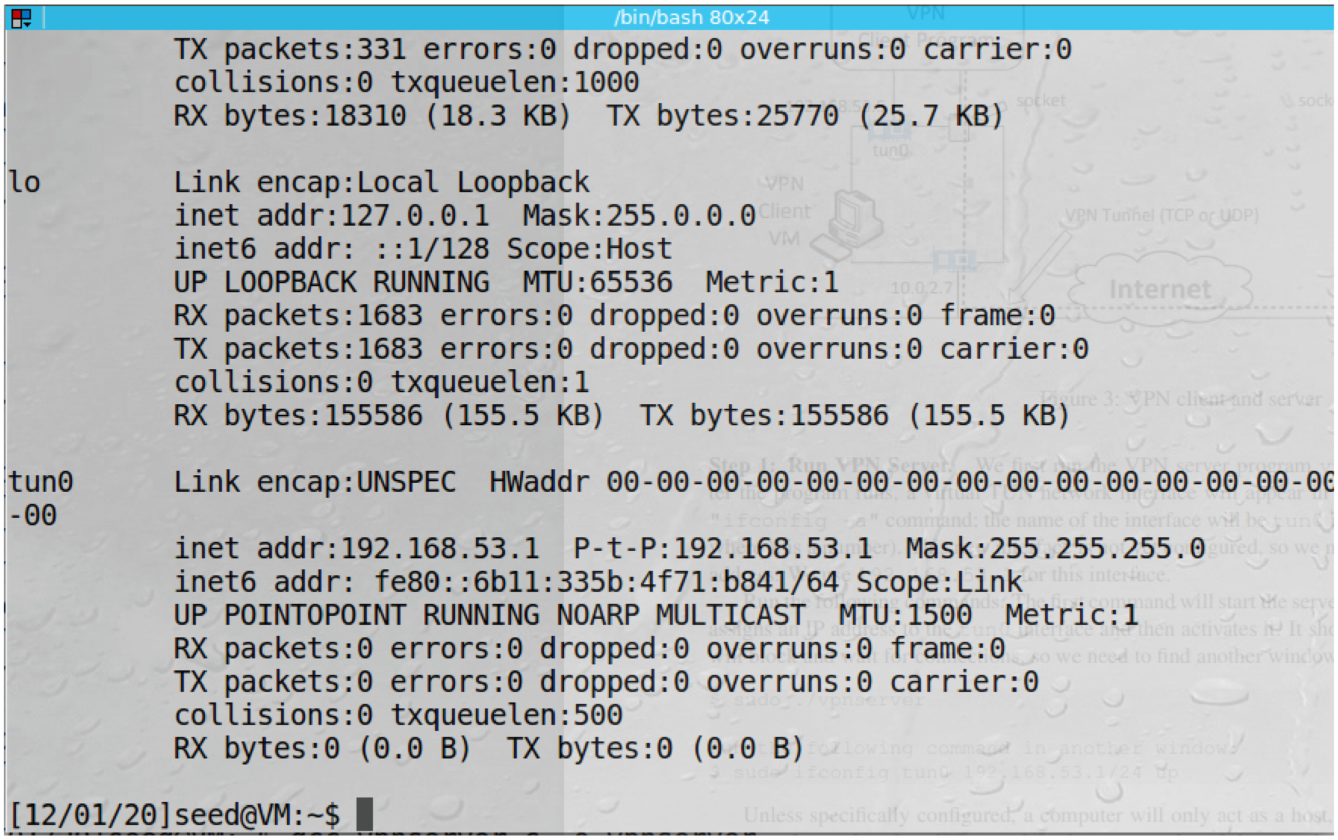
To create a VPN tunnel, we will use the **vpnserver** program provided. First, we will add the IP address in the VPN Server (VM B) and compile the program **vpn_server.c**, and the **vpnserver** program can be ran. Finally it is confirmed using **ifconfig**:



```
[12/01/20]seed@VM:~$ sudo ifconfig tun0 192.168.53.1/24 up
[12/01/20]seed@VM:~$
```

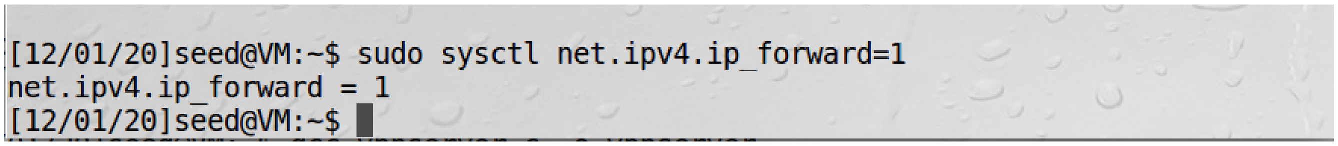
```
bin
Cust
[12/
^C
[12/
andr
bin
Cust
[12/
[12/
[12/
sudo
[12/
andr
bin
Cust
[12/
^C
[12/01/20]seed@VM:~$ gcc vpnserver.c -o vpnserver
[12/01/20]seed@VM:~$ sudo ./vpnserver
```

We can also see the tunnel (tun0) from a standard **ifconfig**:



```
/bin/bash 80x24
          TX packets:331 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18310 (18.3 KB)  TX bytes:25770 (25.7 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
          RX packets:1683 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1683 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:155586 (155.5 KB)  TX bytes:155586 (155.5 KB)

tun0      Link encap:UNSPEC  HWaddr 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  Mask:255.255.255.0
          inet6 addr: fe80::6b11:335b:4f71:b841/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[12/01/20]seed@VM:~$ 
```

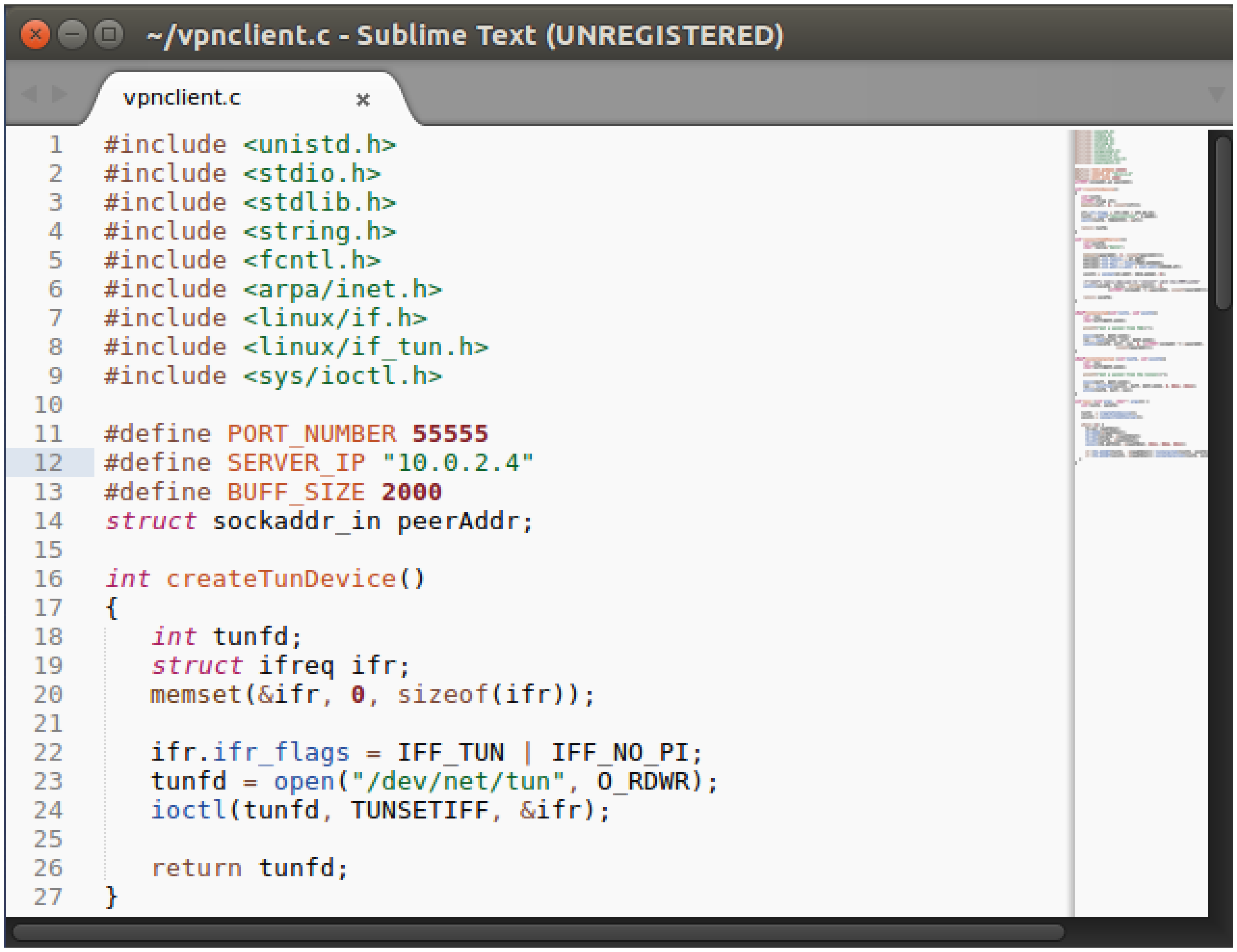
To complete the setup of the VPN server, we must able forwarding so the server will act as a gateway. As it is currently configured, it is only a host. The following **sysctl** command will fix:



```
[12/01/20]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[12/01/20]seed@VM:~$ 
```

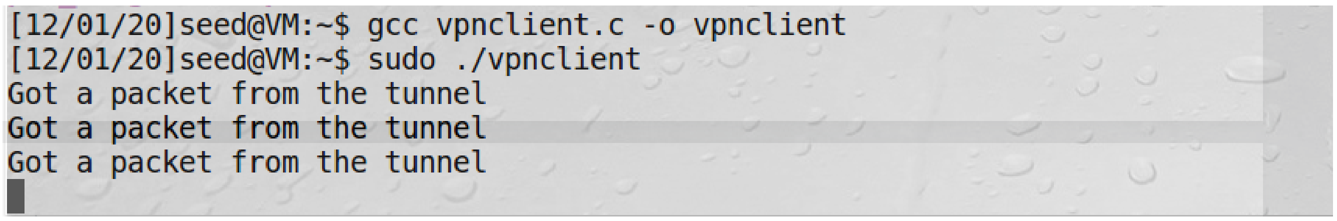
Step 2: Run the VPN Client

In order to run the VPN client, we will use the provided **vpnclient.c** program on Host U (VM A). We will set the server IP to the VPN Server (VM B):



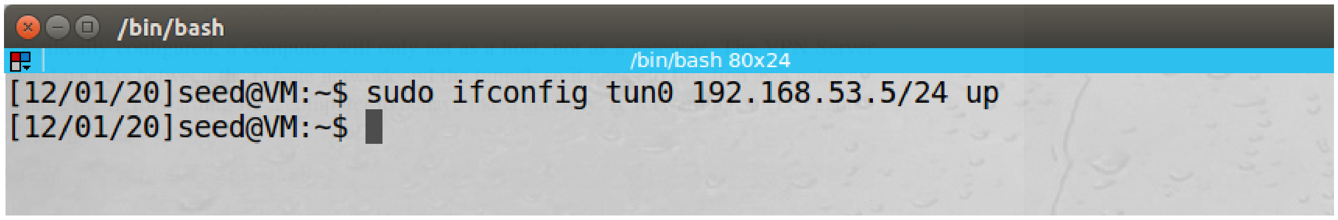
```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>

#define PORT_NUMBER 55555
#define SERVER_IP "10.0.2.4"
#define BUFF_SIZE 2000
struct sockaddr_in peerAddr;

int createTunDevice()
{
    int tunfd;
    struct ifreq ifr;
    memset(&ifr, 0, sizeof(ifr));

    ifr.ifr_flags = IFF_TUN | IFF_NO_PI;
    tunfd = open("/dev/net/tun", O_RDWR);
    ioctl(tunfd, TUNSETIFF, &ifr);

    return tunfd;
}
```

The **vpnclient** running:



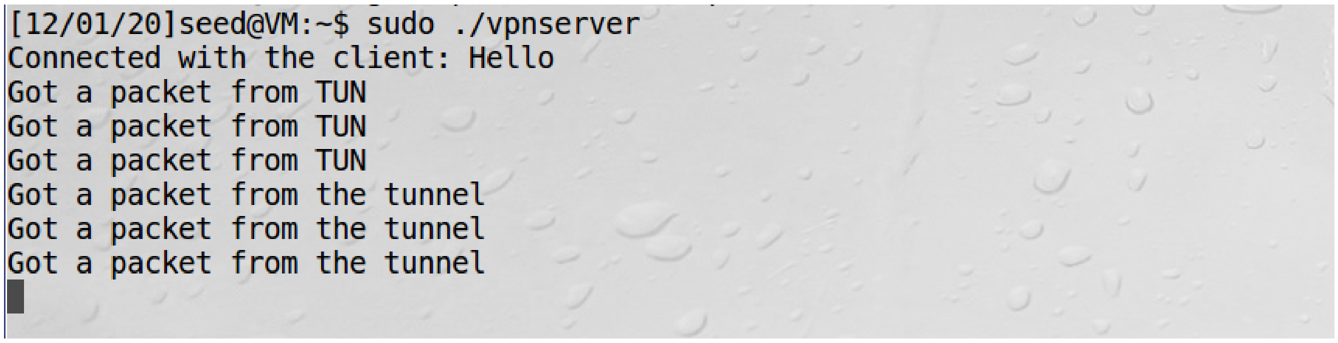
```
[12/01/20]seed@VM:~$ gcc vpnclient.c -o vpnclient
[12/01/20]seed@VM:~$ sudo ./vpnclient
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

Next we configure the tun0 interface to the previously configured IP address at the VPN Server:



```
[12/01/20]seed@VM:~$ sudo ifconfig tun0 192.168.53.5/24 up
[12/01/20]seed@VM:~$ 
```

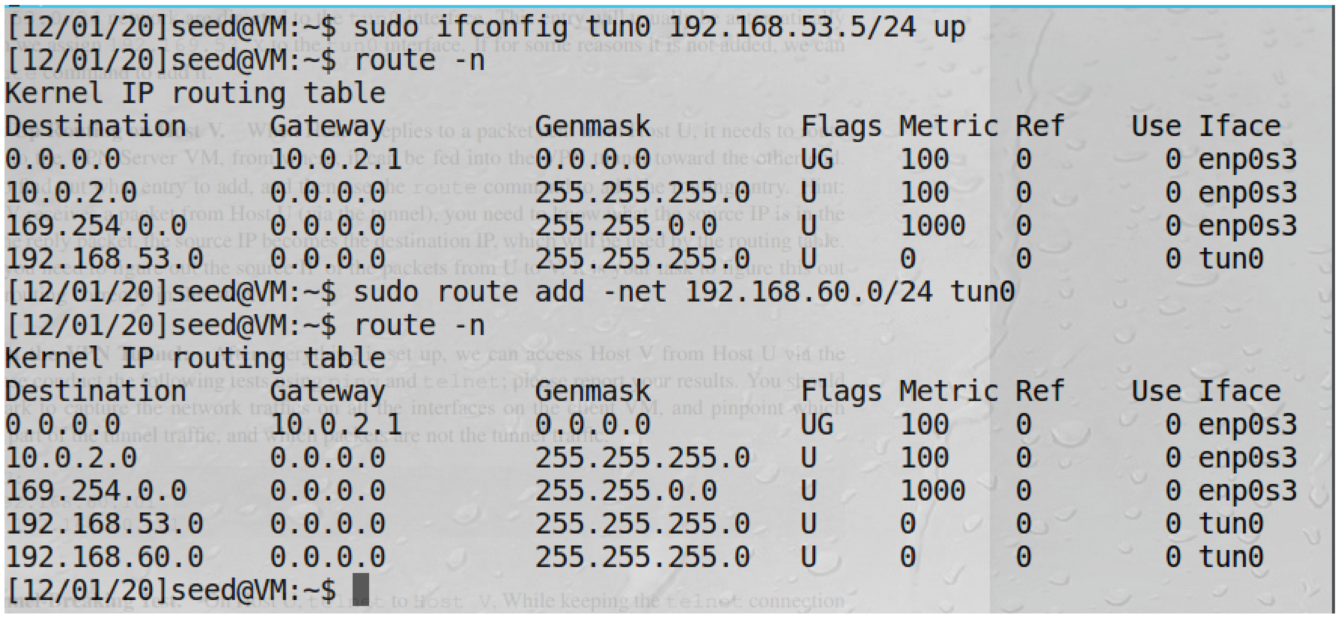
Back on the VPN Server, we can see a successful connection:



```
[12/01/20]seed@VM:~$ 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
```

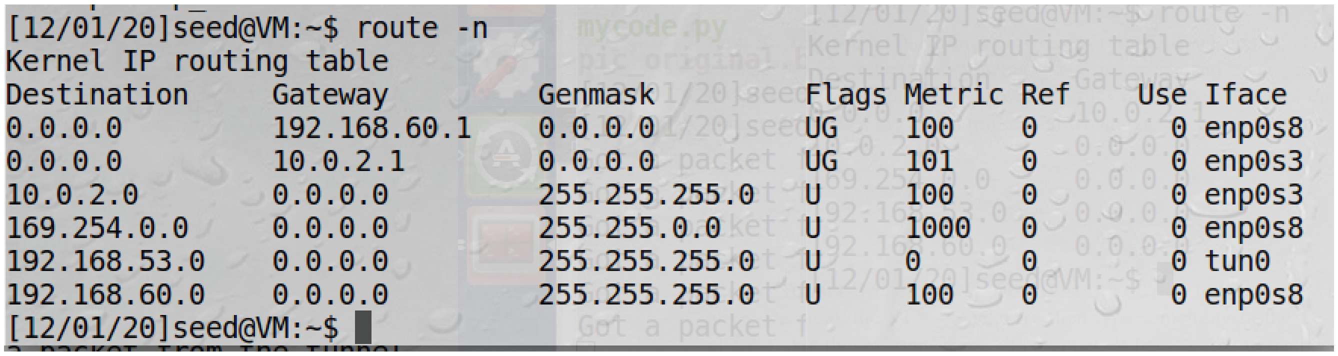
Step 3: Routing on the Client and Server VMs

Now that the connection has been established, we must direct traffic to the tunnel or else it is useless. On Host U, we use the **route** command to establish this protocol. Note the before and after output of running **route -n**, the later showing the last entry of the newly established tunnel:



```
[12/01/20]seed@VM:~$ sudo ifconfig tun0 192.168.53.5/24 up
[12/01/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.53.0    0.0.0.0         255.255.255.0   U     0      0        0 tun0
[12/01/20]seed@VM:~$ sudo route add -net 192.168.60.0/24 tun0
[12/01/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.53.0    0.0.0.0         255.255.255.0   U     0      0        0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U     0      0        0 tun0
[12/01/20]seed@VM:~$ 
```

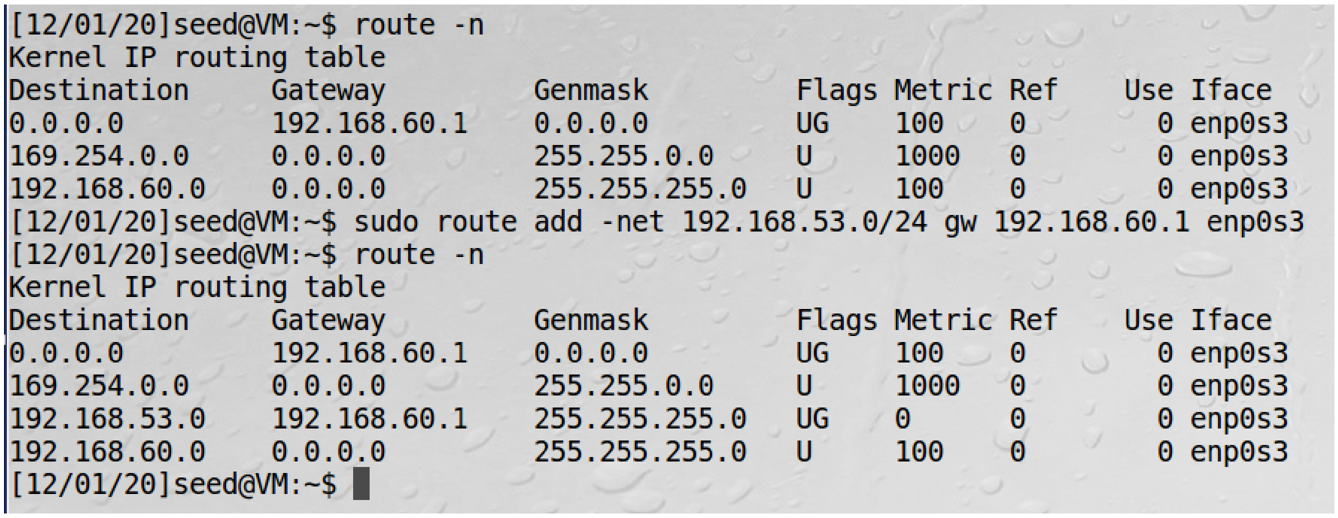
Back on the VPN Server, we can see that this route is recognized:



```
[12/01/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.60.1    0.0.0.0         UG    100    0        0 enp0s8
0.0.0.0         10.0.2.1        0.0.0.0         UG    101    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s8
192.168.53.0    0.0.0.0         255.255.255.0   U     0      0        0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s8
[12/01/20]seed@VM:~$ 
```

Step 4: Set up Routing Host V

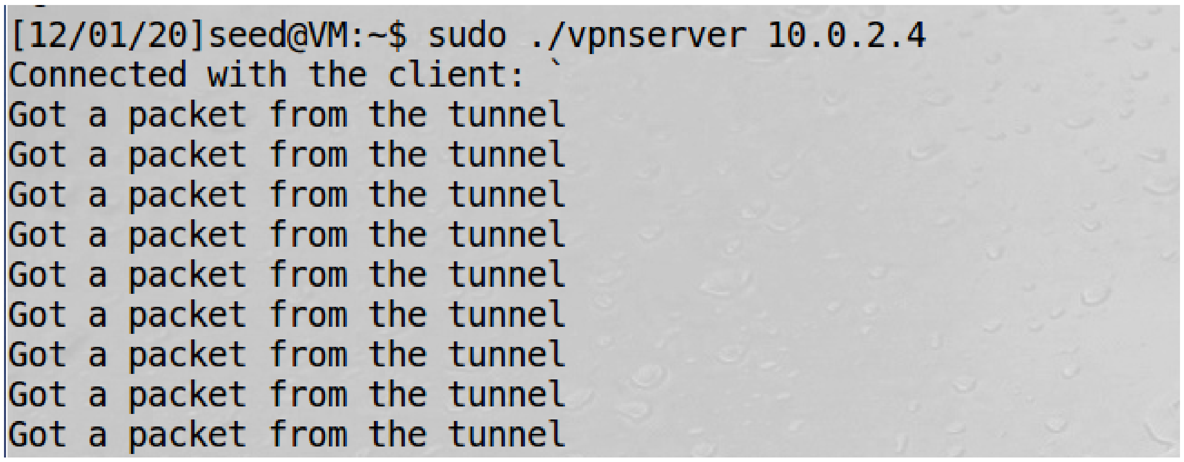
To allow for packets to be sent back to Host U from Host V, we will need to establish this route via the **route** command on VM C, specifiying our new network:



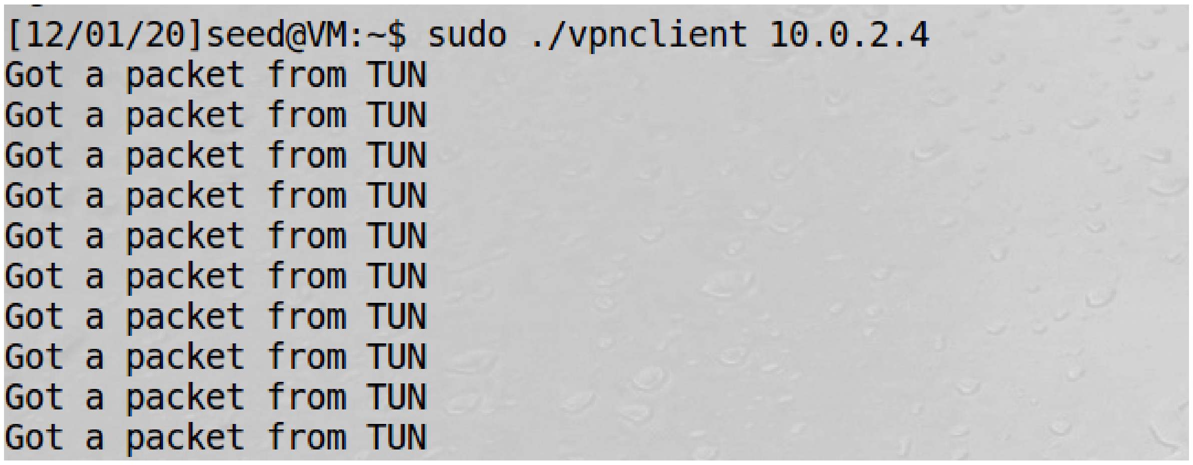
```
[12/01/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.60.1    0.0.0.0         UG    100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.60.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
[12/01/20]seed@VM:~$ sudo route add -net 192.168.53.0/24 gw 192.168.60.1 enp0s3
[12/01/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.60.1    0.0.0.0         UG    100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.53.0    192.168.60.1    255.255.255.0   UG    0      0        0 enp0s3
192.168.60.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
[12/01/20]seed@VM:~$ 
```

Step 5: Test the VPN

With everything in place and the routes established, we can now reach Host V from Host U by first using the **ping** command. Unfortunately I was not able to successfully transfer packets, however I did see the message “Got a packet from the tunnel” on both Host U and the VPN Server:



```
[12/01/20]seed@VM:~$ sudo ./vpnserver 10.0.2.4
Connected with the client: `
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

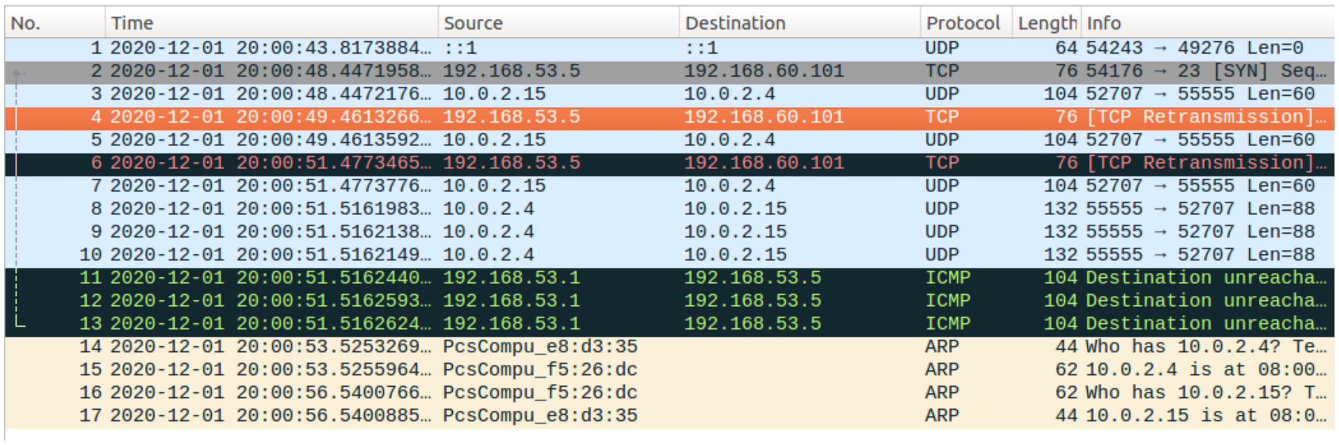


```
[12/01/20]seed@VM:~$ sudo ./vpnclient 10.0.2.4
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

So the tunnel is working, but the fully established route does not seem to be. Telnet was also not able to connect.

Step 6: Tunnel-Breaking Test

Unfortunately I was not able to establish a proper Telnet connection. However, using Wireshark, we can see the same effect as if there had been a connection which was then broken. The output from Host V:



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-12-01 20:00:43.8173884…	::1	::1	UDP	64	54243 → 49276 Len=0
2	2020-12-01 20:00:48.4471958…	192.168.53.5	192.168.60.101	TCP	76	54176 → 23 [SYN] Seq…
3	2020-12-01 20:00:48.4472176…	10.0.2.15	10.0.2.4	UDP	104	52707 → 55555 Len=60
4	2020-12-01 20:00:49.4613266…	192.168.53.5	192.168.60.101	TCP	76	[TCP Retransmission]…
5	2020-12-01 20:00:49.4613592…	10.0.2.15	10.0.2.4	UDP	104	52707 → 55555 Len=60
6	2020-12-01 20:00:51.4773465…	192.168.53.5	192.168.60.101	TCP	76	[TCP Retransmission]…
7	2020-12-01 20:00:51.4773776…	10.0.2.15	10.0.2.4	UDP	104	52707 → 55555 Len=60
8	2020-12-01 20:00:51.5161983…	10.0.2.4	10.0.2.15	UDP	132	55555 → 52707 Len=88
9	2020-12-01 20:00:51.5162138…	10.0.2.4	10.0.2.15	UDP	132	55555 → 52707 Len=88
10	2020-12-01 20:00:51.5162149…	10.0.2.4	10.0.2.15	UDP	132	55555 → 52707 Len=88
11	2020-12-01 20:00:51.5162440…	192.168.53.1	192.168.53.5	ICMP	104	Destination unreacha…
12	2020-12-01 20:00:51.5162593…	192.168.53.1	192.168.53.5	ICMP	104	Destination unreacha…
13	2020-12-01 20:00:51.5162624…	192.168.53.1	192.168.53.5	ICMP	104	Destination unreacha…
14	2020-12-01 20:00:53.5253269…	PcsCompu_e8:d3:35		ARP	44	Who has 10.0.2.4? Te…
15	2020-12-01 20:00:53.5255964…	PcsCompu_f5:26:dc		ARP	62	10.0.2.4 is at 08:00…
16	2020-12-01 20:00:56.5400766…	PcsCompu_f5:26:dc		ARP	62	Who has 10.0.2.15? T…
17	2020-12-01 20:00:56.5400885…	PcsCompu_e8:d3:35		ARP	44	10.0.2.15 is at 08:0…

The source and destination appear to be correct, and we can see that output is similar to the expected broken Telnet output.

Observation: In order to properly connect to Host U and make use of the private network established between VPN Server and Host U, we must properly configure the entire route. That includes specifying the types of the connections and the IP address (actual) of the VPN Server, as well as the newly defined tunnel IP Addresses.

Explanation: Unfortunatley the final path did not work, but we were able to see the conneection attempted. All three VM’s appeared to be set up correctly and working as intended. The expectation was that a ping from VM A would reach VM C through VM B, but that did not quite happen.