Lab 6 VPN Tunneling

Task 1: Network Setup

主机	IP
Host U	192.168.11.140
VPN Server	192.168.11.143 192.168.22.1
Host V	192.168.22.2

Host U

```
[09/25/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:9e:84:54
          inet addr:192.168.11.140  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::f3a5:fd02:c0f0:646/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:711 errors:0 dropped:0 overruns:0 frame:0
          TX packets:216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:79520 (79.5 KB)  TX bytes:30627 (30.6 KB)
          Interrupt:19 Base address:0x2000

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:537 errors:0 dropped:0 overruns:0 frame:0
          TX packets:537 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:69808 (69.8 KB)  TX bytes:69808 (69.8 KB)
```

```
[09/25/20]seed@VM:~$ ping 192.168.11.143
PING 192.168.11.143 (192.168.11.143) 56(84) bytes of data.
64 bytes from 192.168.11.143: icmp_seq=1 ttl=64 time=6.98 ms
64 bytes from 192.168.11.143: icmp_seq=2 ttl=64 time=0.543 ms

--- 192.168.11.143 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.543/3.764/6.985/3.221 ms
[09/25/20]seed@VM:~$ ping 192.168.22.1
PING 192.168.22.1 (192.168.22.1) 56(84) bytes of data.

--- 192.168.22.1 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13301ms

^C[09/25/20]seed@VM:~$ ping 192.168.22.2
PING 192.168.22.2 (192.168.22.2) 56(84) bytes of data.
^C
--- 192.168.22.2 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6149ms
```

VPN Server

```
[09/25/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:85:08:da
          inet addr:192.168.11.143  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::71ec:5111:bbe0:d6a4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:460 errors:0 dropped:0 overruns:0 frame:0
          TX packets:303 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:62005 (62.0 KB)  TX bytes:27922 (27.9 KB)
          Interrupt:19 Base address:0x2000

ens38     Link encap:Ethernet  HWaddr 00:0c:29:85:08:e4
          inet addr:192.168.22.1  Bcast:192.168.22.255  Mask:255.255.255.0
          inet6 addr: fe80::e183:d745:f0f4:9f16/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1089 errors:0 dropped:0 overruns:0 frame:0
          TX packets:525 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:105144 (105.1 KB)  TX bytes:80771 (80.7 KB)
          Interrupt:16 Base address:0x2400

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:545 errors:0 dropped:0 overruns:0 frame:0
          TX packets:545 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:78453 (78.4 KB)  TX bytes:78453 (78.4 KB)
```

```
[09/25/20]seed@VM:~$ ping 192.168.11.140
PING 192.168.11.140 (192.168.11.140) 56(84) bytes of data.
64 bytes from 192.168.11.140: icmp_seq=1 ttl=64 time=50.2 ms
64 bytes from 192.168.11.140: icmp_seq=2 ttl=64 time=0.602 ms
64 bytes from 192.168.11.140: icmp_seq=3 ttl=64 time=0.545 ms
^C
--- 192.168.11.140 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 0.545/17.138/50.269/23.427 ms
[09/25/20]seed@VM:~$ ping 192.168.22.2
PING 192.168.22.2 (192.168.22.2) 56(84) bytes of data.
64 bytes from 192.168.22.2: icmp_seq=1 ttl=64 time=95.1 ms
64 bytes from 192.168.22.2: icmp_seq=2 ttl=64 time=0.382 ms
64 bytes from 192.168.22.2: icmp_seq=3 ttl=64 time=0.367 ms
^C
--- 192.168.22.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 0.367/31.963/95.142/44.674 ms
```

Host V

```
[09/25/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:aa:f9:5a
          inet addr:192.168.22.2  Bcast:192.168.22.255  Mask:255.255.255.0
          inet6 addr: fe80::c3db:d48a:3e70:eb1c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:524 errors:0 dropped:0 overruns:0 frame:0
          TX packets:380 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109650 (109.6 KB)  TX bytes:50899 (50.8 KB)
          Interrupt:19 Base address:0x2000

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:436 errors:0 dropped:0 overruns:0 frame:0
          TX packets:436 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:59323 (59.3 KB)  TX bytes:59323 (59.3 KB)
```

```
[09/25/20]seed@VM:~$ ping 192.168.11.140
PING 192.168.11.140 (192.168.11.140) 56(84) bytes of data.
^C
--- 192.168.11.140 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3051ms

[09/25/20]seed@VM:~$ ping 192.168.22.1
PING 192.168.22.1 (192.168.22.1) 56(84) bytes of data.
64 bytes from 192.168.22.1: icmp_seq=1 ttl=64 time=0.426 ms
64 bytes from 192.168.22.1: icmp_seq=2 ttl=64 time=0.599 ms
64 bytes from 192.168.22.1: icmp_seq=3 ttl=64 time=0.467 ms
^C
--- 192.168.22.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 0.426/0.497/0.599/0.076 ms
```

网络设置完成。

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

```python
#!/usr/bin/python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'groot', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
```

```
print("Interface Name: {}".format(ifname))

while True:
    time.sleep(10)
```

运行脚本添加接口

```
[09/25/20]seed@VM:~/.../week4$ sudo ./tun.py
Interface Name: groot
```

观察端口信息

```
[09/25/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
 default qlen 1
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP group default qlen 1000
    link/ether 00:0c:29:9e:84:54 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.140/24 brd 192.168.11.255 scope global dynamic ens33
       valid_lft 1167sec preferred_lft 1167sec
    inet6 fe80::f3a5:fd02:c0f0:646/64 scope link
       valid_lft forever preferred_lft forever
4: groot: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN gr
oup default qlen 500
    link/none
```

观察到出现虚拟端口groot

Task 2.b: Set up the TUN Interface

```
$ sudo ip addr add 192.168.53.99/24 dev groot
$ sudo ip link set dev groot up
```

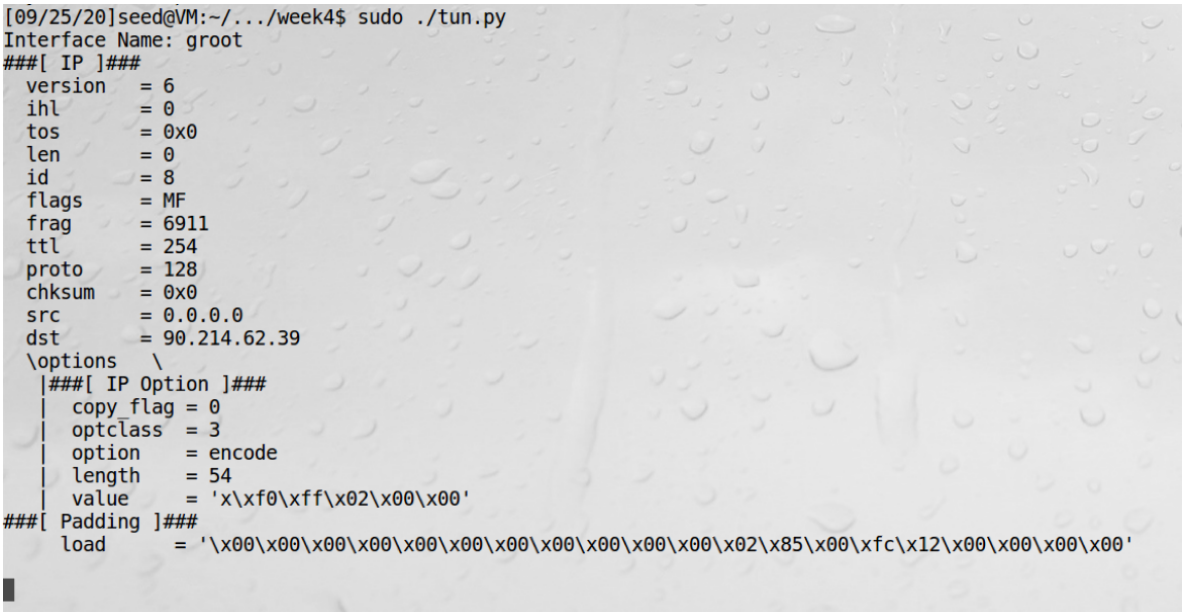
接口配置并开启成功

```
[09/25/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
 default qlen 1
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP group default qlen 1000
    link/ether 00:0c:29:9e:84:54 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.140/24 brd 192.168.11.255 scope global dynamic ens33
       valid_lft 1571sec preferred_lft 1571sec
    inet6 fe80::f3a5:fd02:c0f0:646/64 scope link
       valid_lft forever preferred_lft forever
4: groot: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global groot
       valid_lft forever preferred_lft forever
    inet6 fe80::f3a5:bcbc:4c05:1a95/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

添加代码后自动执行

```
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
```

Task 2.c: Read from the TUN Interface




```
$ ping 192.168.53.22
```

看到打印出发往192.168.53.22的数据包发送给虚拟接口，源地址被修改为fancy端口的IP地址，而非192.168.11.140

```
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x2df7
     id        = 0x218c
     seq       = 0x6
###[ Raw ]###
        load      = 'b\xabm_\xe9h\x04\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x
11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-
./01234567'

###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 44696
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0xa046
  src       = 192.168.53.99
  dst       = 192.168.53.22
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x8c99
     id        = 0x218c
     seq       = 0x7
###[ Raw ]###
        load      = 'c\xabm_\x89\xc5\x04\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x1
0\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*
+,-./01234567'
```

ping 192.168.22.2时发现没有信息产生

Task 2.d: Write to the TUN Interface

```
# Send out a spoof packet using the tun interface
newip = IP(src='1.2.3.4', dst=ip.src)
newpkt = newip/ip.payload
os.write(tun, bytes(newpkt))
```

添加代码后在wireshark发现有不同方向的报文出现，TUN运行成功。

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-25 04:57:05.5078315…	::1	::1	UDP	64	49608 → 44721 Len=0
2	2020-09-25 04:57:11.3686887…	192.168.53.99	192.168.53.22	ICMP	100	Echo (ping) request id=0x248f,
3	2020-09-25 04:57:11.3707098…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x248f,
4	2020-09-25 04:57:12.4006623…	192.168.53.99	192.168.53.22	ICMP	100	Echo (ping) request id=0x248f,
5	2020-09-25 04:57:12.4110120…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x248f,
6	2020-09-25 04:57:13.4250566…	192.168.53.99	192.168.53.22	ICMP	100	Echo (ping) request id=0x248f,
7	2020-09-25 04:57:13.4340050…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x248f,
8	2020-09-25 04:57:14.4493635…	192.168.53.99	192.168.53.22	ICMP	100	Echo (ping) request id=0x248f,
9	2020-09-25 04:57:14.4585965…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x248f,
10	2020-09-25 04:57:15.4729126…	192.168.53.99	192.168.53.22	ICMP	100	Echo (ping) request id=0x248f,
11	2020-09-25 04:57:15.4832881…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x248f,

随意写入数据而非IP数据报发现会报错

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

在VPN Server中运行程序

```python
#!/usr/bin/python3

from scapy.all import *

IP_A = "0.0.0.0"
PORT = 9090

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

while True:
    data, (ip, port) = sock.recvfrom(2048)
    print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
    pkt = IP(data)
    print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
```

在Host U中编辑tun程序

```python
    if True:
        # Send the packet via the tunnel
        sock.sendto(packet, ("192.168.11.143", 9090))
        print("Sent.")
```

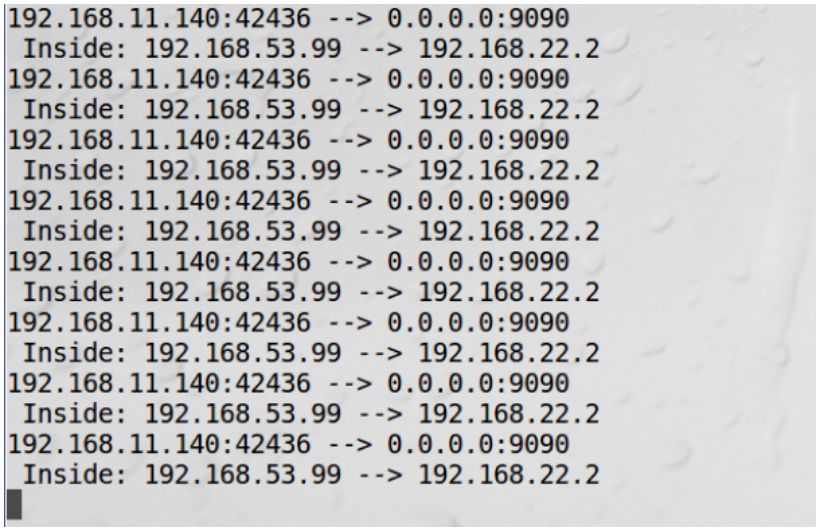
双方同时运行程序，并ping 192.168.53.13

```
[09/25/20]seed@VM:~/.../week4$ sudo ./tun_server.py
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
192.168.11.140:42436 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.13
```

得到外层由192.168.11.140:42436发往自己的IP数据报，内部为192.168.53.99发往192.168.53.13的数据报。由此通过隧道在Host U和VPN Server间实现了通信。而ping 192.168.22.2没有响应，配置Host U的路由信息，将192.168.22.0/24的出口设置为groot，让其走groot接口

```
$ sudo ip route add 192.168.22.0/24 dev groot via 192.168.53.99
```

在Host U端ping 192.168.22.2，在VPN Server端正常接收到了Host U发往192.168.22.2的报文



可以看到两端对IP数据报的操作能够使其经过去除头部后得到原本的IP数据报，并且IP与所使用的端口对应

Task 4: Set Up the VPN Server

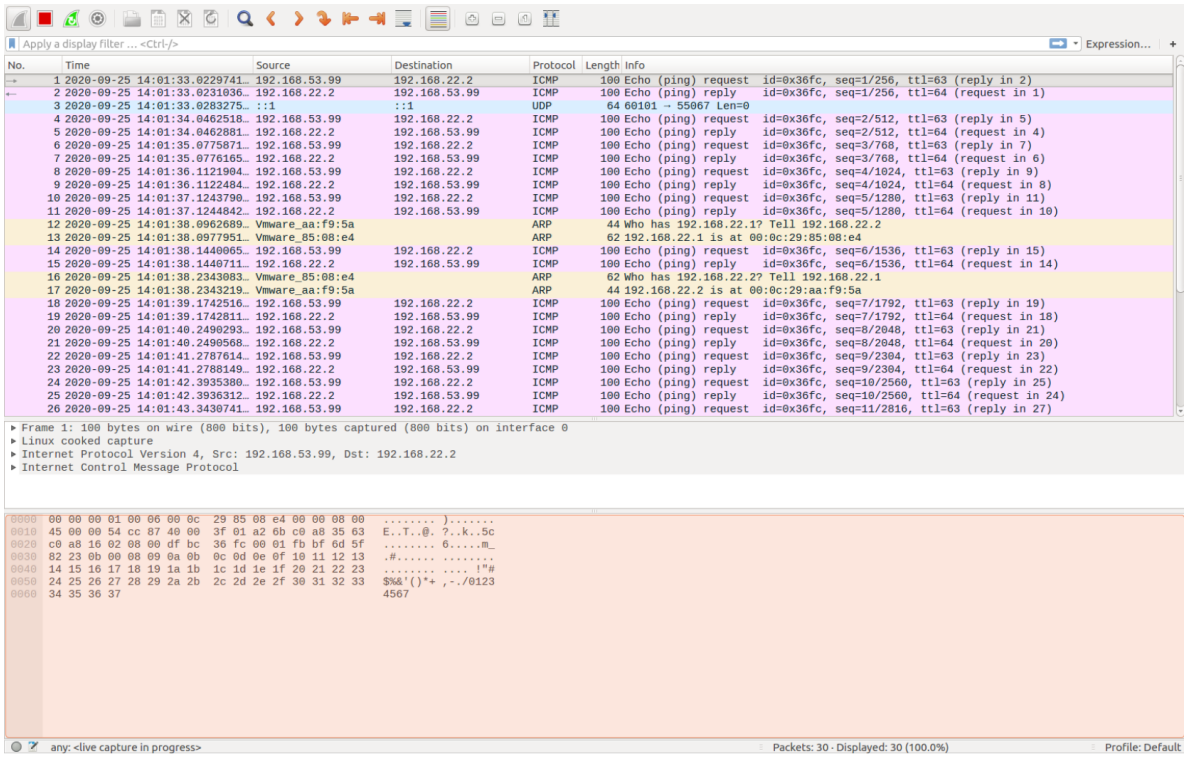
修改tun_server.py

```python
#!/usr/bin/python3

import fcntl
import struct
import os
import time
from scapy.all import *

IP_A = '0.0.0.0'
PORT = 9090

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'server', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.1/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

while True:
    data, (ip, port) = sock.recvfrom(2048)
    os.write(tun,data)
    print("Got one.")
```

打开VPN Server的路由选项

```
$ sudo sysctl net.ipv4.ip_forward=1
```

可以观察到在Host V端已经收到了来自Host U的ICMP请求报文



但由于只实现了单方向的TUN，所以无法传回ICMP响应报文。

Task 5: Handling Traffic in Both Directions

修改Host U与VPN Server的程序

Host U

```
#!/usr/bin/python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

IP_C = "0.0.0.0"
PORT = 9090
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'client', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

```

```
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_C,PORT))
while True:
    # this will block until at least one interface is ready
    ready, _, _ = select.select([sock, tun], [], [])

    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            os.write(tun,data)

        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            sock.sendto(packet,("192.168.11.143",9090))
```

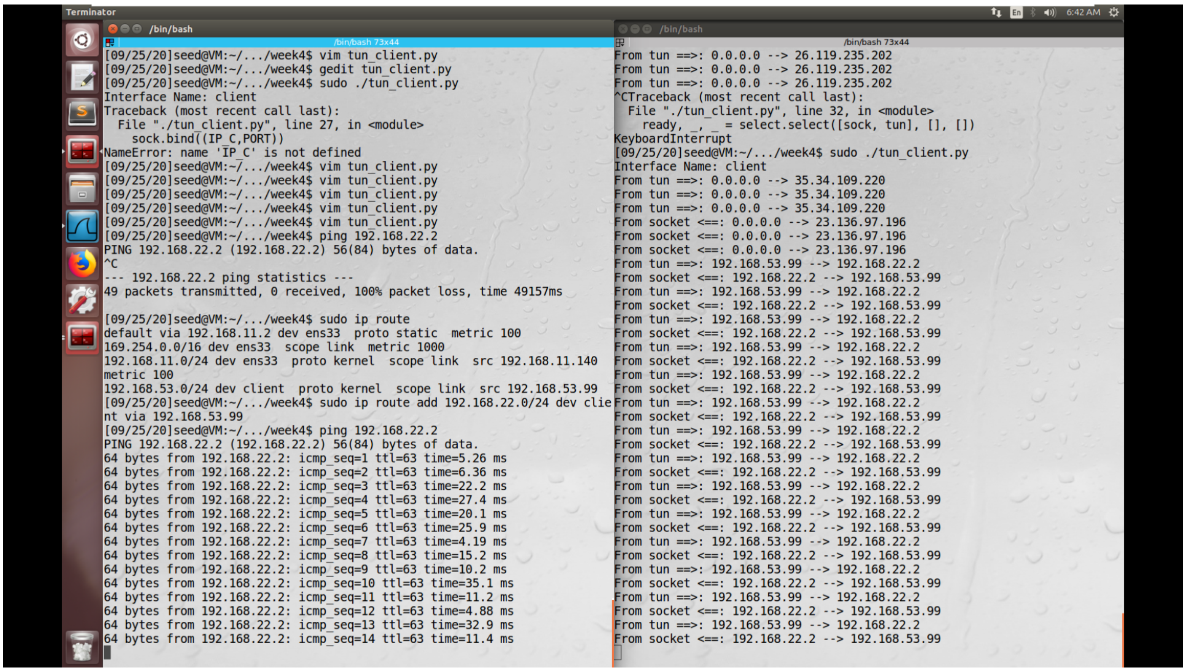
VPN Server

```
#!/usr/bin/python3

import fcntl
import struct
import os
import time
from scapy.all import *

IP_A = '0.0.0.0'
PORT = 9090

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'server', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.1/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

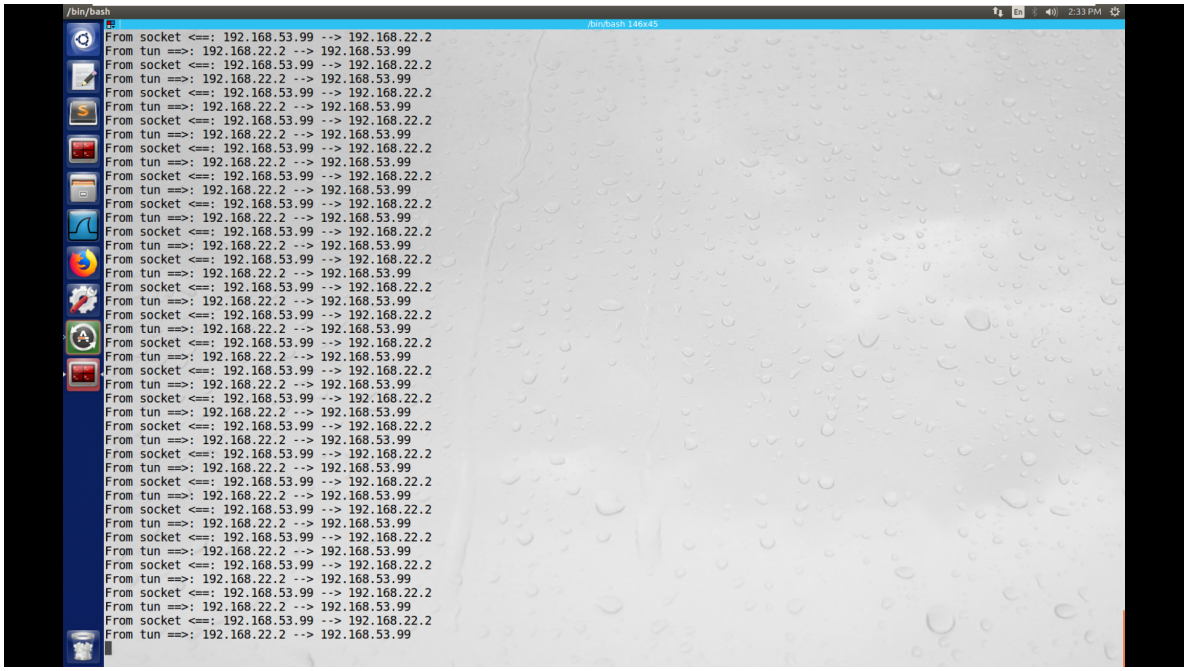
```python
sock.bind((IP_A, PORT))

while True:
    # this will block until at least one interface is ready
    ready, _, _ = select.select([sock, tun], [], [])

    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            os.write(tun,data)

        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            sock.sendto(packet,("192.168.11.140",9090))
```

两端运行后在Host U执行ping 192.168.22.2，这一次能够看到双向流量

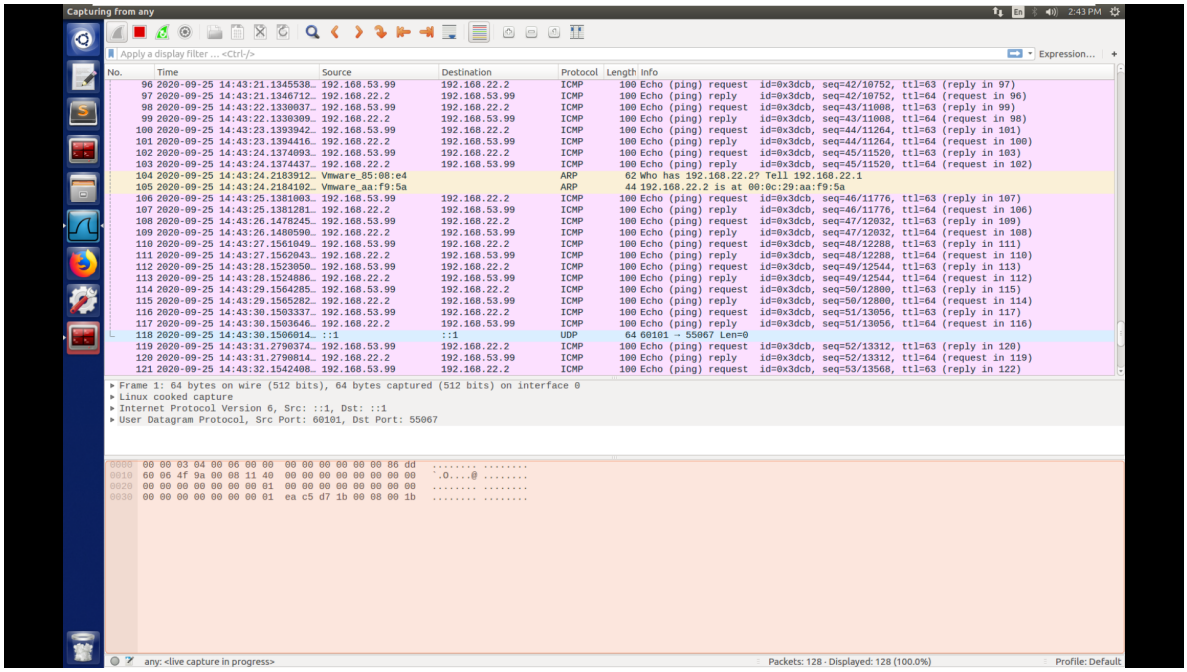
Host U



VPN Server

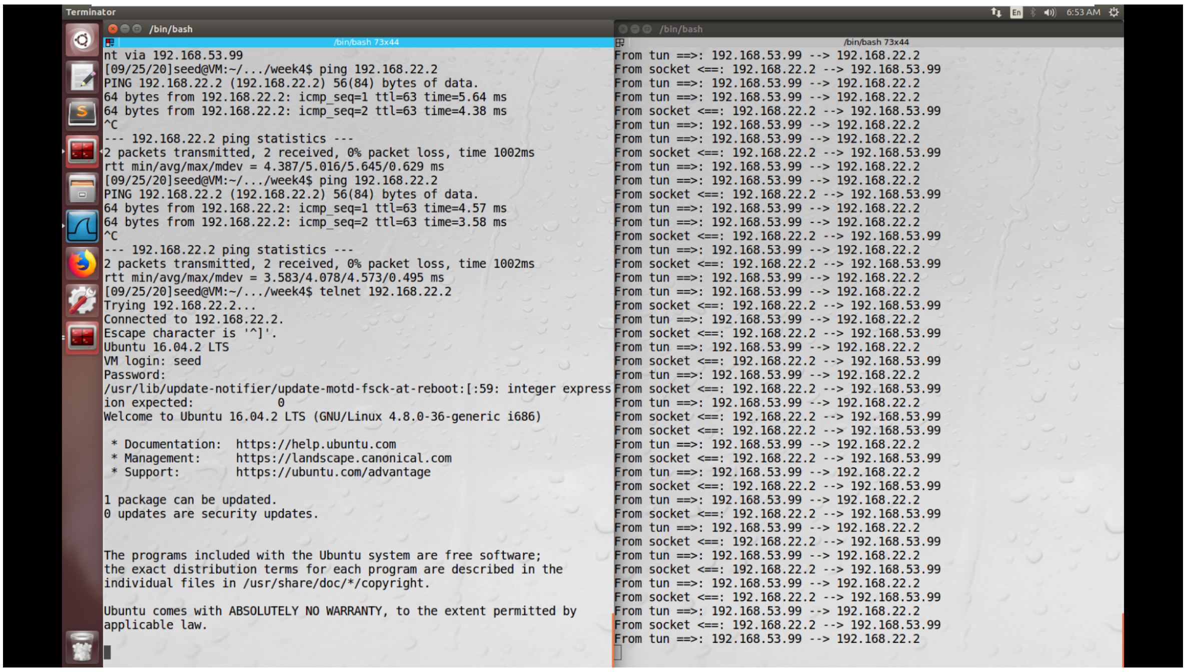


Host V

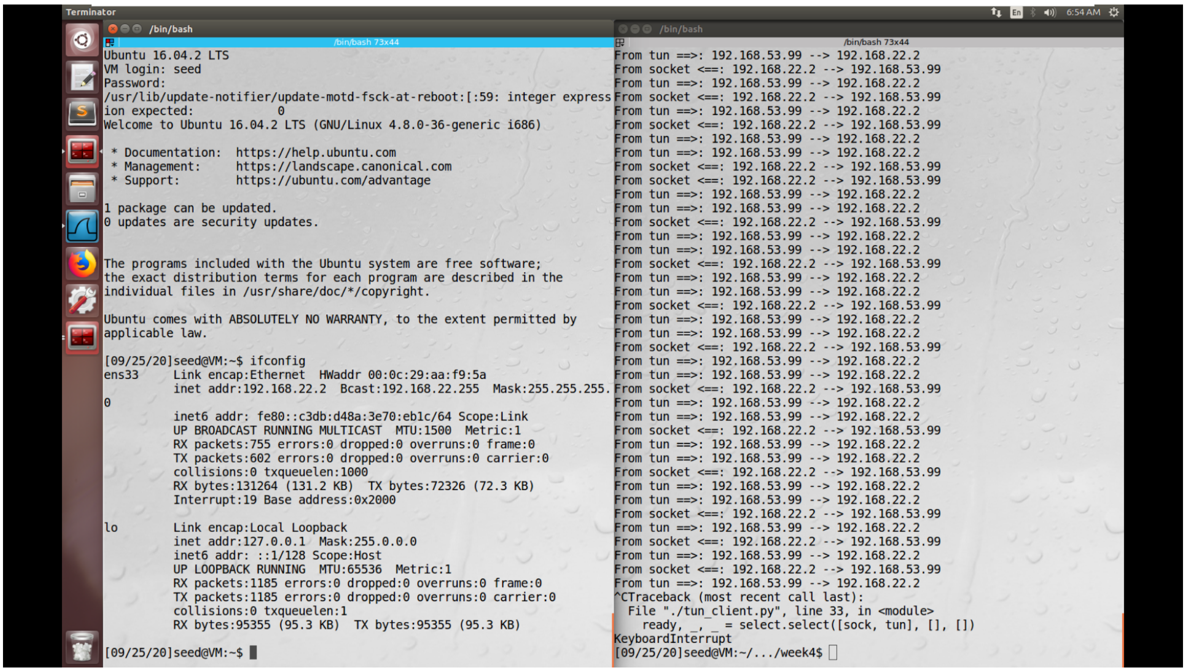


Task 6: Tunnel-Breaking Experiment

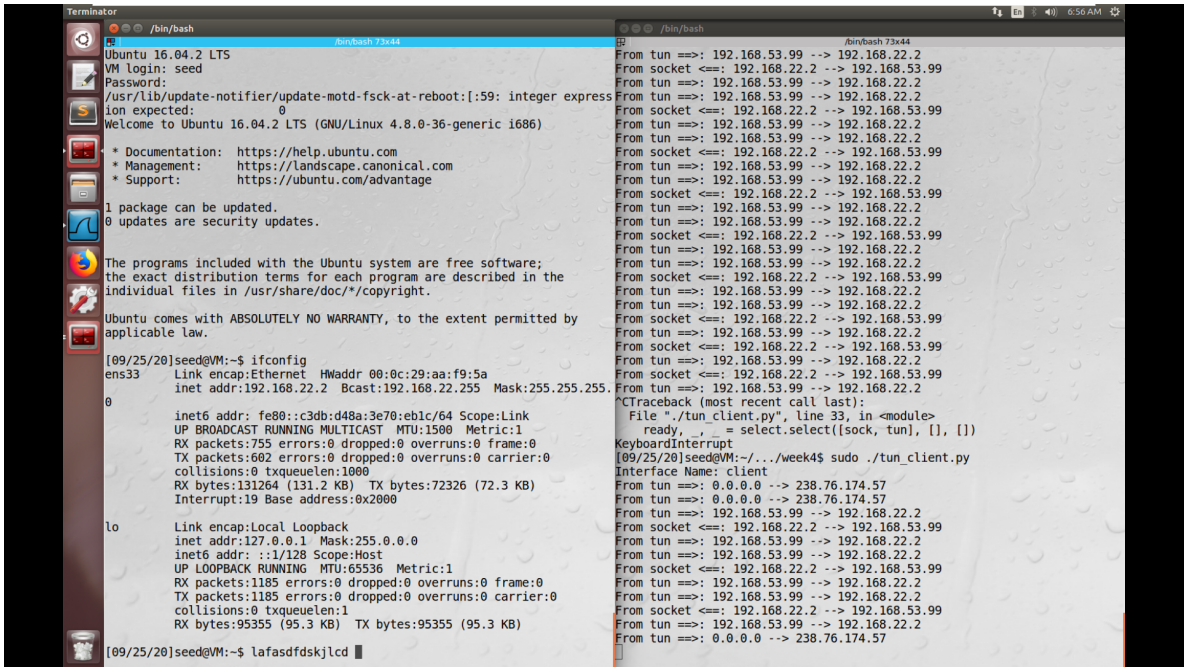
首先在Host U对Host V发起telnet，可正常输入指令



断开tun_client后不能再输入指令



短时间内恢复tun_client连接后telnet恢复正常



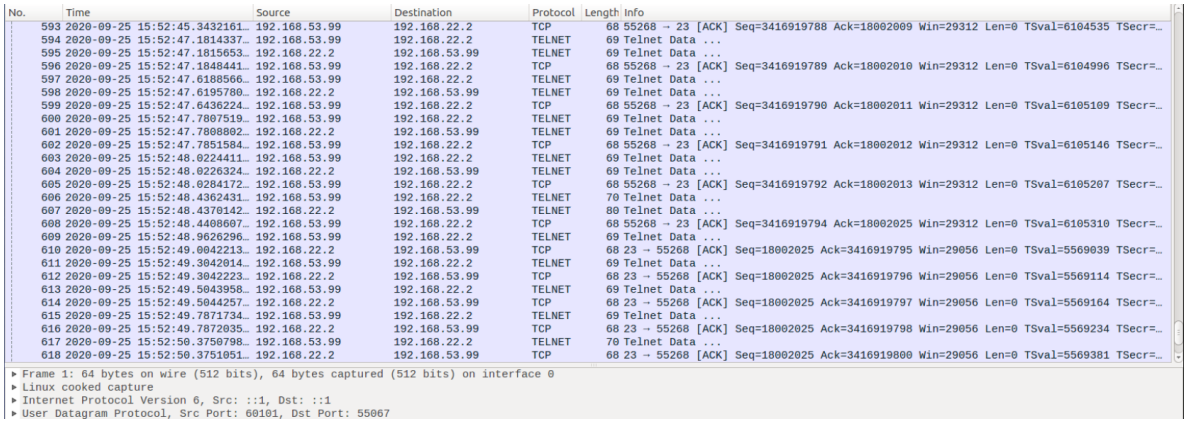
Task 7: Routing Experiment on Host V

对Host V进行路由配置

```
[09/25/20]seed@VM:~$ sudo ip route del 0.0.0.0/0
[09/25/20]seed@VM:~$ sudo ip route add 192.168.53.0/24 dev ens33 via 192.168.22.1
[09/25/20]seed@VM:~$ sudo ip route add 192.168.11.0/24 dev ens33 via 192.168.22.1
[09/25/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 ens33
192.168.11.0    192.168.22.1    255.255.255.0   UG    0      0        0 ens33
192.168.22.0    0.0.0.0         255.255.255.0   U     100    0        0 ens33
192.168.53.0    192.168.22.1    255.255.255.0   UG    0      0        0 ens33
[09/25/20]seed@VM:~$
```

由Host U向Host V发起telnet进行测试

```
[09/25/20]seed@VM:~/.../week4$ telnet 192.168.22.2
Trying 192.168.22.2...
Connected to 192.168.22.2.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Sep 25 14:53:24 EDT 2020 from 192.168.53.99 on pts/18
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.

[09/25/20]seed@VM:~$
```



No.	Time	Source	Destination	Protocol	Length	Info
593	2020-09-25 15:52:45.3432161…	192.168.53.99	192.168.22.2	TCP	68	55268 → 23 [ACK] Seq=3416919788 Ack=18002009 Win=29312 Len=0 TSval=6104535 TSecr=…
594	2020-09-25 15:52:47.1814337…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
595	2020-09-25 15:52:47.1815653…	192.168.22.2	192.168.53.99	TELNET	69	Telnet Data ...
596	2020-09-25 15:52:47.1848441…	192.168.53.99	192.168.22.2	TCP	68	55268 → 23 [ACK] Seq=3416919789 Ack=18002010 Win=29312 Len=0 TSval=6104996 TSecr=…
597	2020-09-25 15:52:47.6188566…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
598	2020-09-25 15:52:47.6195780…	192.168.22.2	192.168.53.99	TELNET	69	Telnet Data ...
599	2020-09-25 15:52:47.6436224…	192.168.53.99	192.168.22.2	TCP	68	55268 → 23 [ACK] Seq=3416919790 Ack=18002011 Win=29312 Len=0 TSval=6105109 TSecr=…
600	2020-09-25 15:52:47.7807519…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
601	2020-09-25 15:52:47.7808802…	192.168.22.2	192.168.53.99	TELNET	69	Telnet Data ...
602	2020-09-25 15:52:47.7851584…	192.168.53.99	192.168.22.2	TCP	68	55268 → 23 [ACK] Seq=3416919791 Ack=18002012 Win=29312 Len=0 TSval=6105146 TSecr=…
603	2020-09-25 15:52:48.0224411…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
604	2020-09-25 15:52:48.0226324…	192.168.22.2	192.168.53.99	TELNET	69	Telnet Data ...
605	2020-09-25 15:52:48.0284172…	192.168.53.99	192.168.22.2	TCP	68	55268 → 23 [ACK] Seq=3416919792 Ack=18002013 Win=29312 Len=0 TSval=6105207 TSecr=…
606	2020-09-25 15:52:48.4362431…	192.168.53.99	192.168.22.2	TELNET	70	Telnet Data ...
607	2020-09-25 15:52:48.4370142…	192.168.22.2	192.168.53.99	TELNET	80	Telnet Data ...
608	2020-09-25 15:52:48.4408607…	192.168.53.99	192.168.22.2	TCP	68	55268 → 23 [ACK] Seq=3416919794 Ack=18002025 Win=29312 Len=0 TSval=6105310 TSecr=…
609	2020-09-25 15:52:48.9626296…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
610	2020-09-25 15:52:49.0042213…	192.168.22.2	192.168.53.99	TCP	68	23 → 55268 [ACK] Seq=18002025 Ack=3416919795 Win=29056 Len=0 TSval=5569039 TSecr=…
611	2020-09-25 15:52:49.3042014…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
612	2020-09-25 15:52:49.3042223…	192.168.22.2	192.168.53.99	TCP	68	23 → 55268 [ACK] Seq=18002025 Ack=3416919796 Win=29056 Len=0 TSval=5569114 TSecr=…
613	2020-09-25 15:52:49.5043958…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
614	2020-09-25 15:52:49.5044257…	192.168.22.2	192.168.53.99	TCP	68	23 → 55268 [ACK] Seq=18002025 Ack=3416919797 Win=29056 Len=0 TSval=5569164 TSecr=…
615	2020-09-25 15:52:49.7871734…	192.168.53.99	192.168.22.2	TELNET	69	Telnet Data ...
616	2020-09-25 15:52:49.7872035…	192.168.22.2	192.168.53.99	TCP	68	23 → 55268 [ACK] Seq=18002025 Ack=3416919798 Win=29056 Len=0 TSval=5569234 TSecr=…
617	2020-09-25 15:52:50.3750798…	192.168.53.99	192.168.22.2	TELNET	70	Telnet Data ...
618	2020-09-25 15:52:50.3751051…	192.168.22.2	192.168.53.99	TCP	68	23 → 55268 [ACK] Seq=18002025 Ack=3416919800 Win=29056 Len=0 TSval=5569381 TSecr=…

▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 6, Src: ::1, Dst: ::1
▸ User Datagram Protocol, Src Port: 60101, Dst Port: 55067

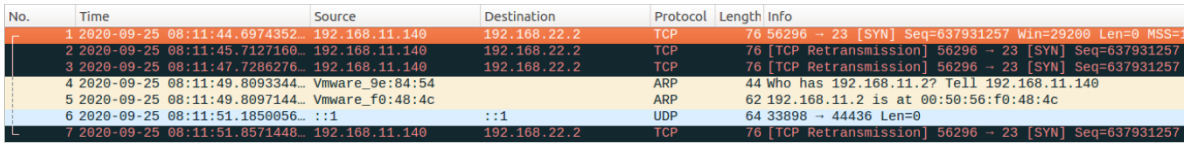
双方成功通信，且经VPN Server路由。

Task 8: Experiment with the TUN IP Address

修改Host U中的网段地址

```
25 os.system("ip addr add 192.168.13.99/24 dev {}".format(ifname))
26 os.system("ip link set dev {} up".format(ifname))
27 os.system("ip route add 192.168.22.0/24 dev {} via 192.168.53.99".for
   mat(ifname))
```

对Host V发起telnet，连接失败，查看wireshark，发现在VPN Server上有收到报文，但是没有进行转发



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-25 08:11:44.6974352…	192.168.11.140	192.168.22.2	TCP	76	56296 → 23 [SYN] Seq=637931257 Win=29200 Len=0 MSS=1
2	2020-09-25 08:11:45.7127160…	192.168.11.140	192.168.22.2	TCP	76	[TCP Retransmission] 56296 → 23 [SYN] Seq=637931257
3	2020-09-25 08:11:47.7286276…	192.168.11.140	192.168.22.2	TCP	76	[TCP Retransmission] 56296 → 23 [SYN] Seq=637931257
4	2020-09-25 08:11:49.8093344…	Vmware_9e:84:54		ARP	44	Who has 192.168.11.2? Tell 192.168.11.140
5	2020-09-25 08:11:49.8097144…	Vmware_f0:48:4c		ARP	62	192.168.11.2 is at 00:50:56:f0:48:4c
6	2020-09-25 08:11:51.1850056…	::1	::1	UDP	64	33898 → 44436 Len=0
7	2020-09-25 08:11:51.8571448…	192.168.11.140	192.168.22.2	TCP	76	[TCP Retransmission] 56296 → 23 [SYN] Seq=637931257

因为Linux内核中的RPC机制，使得其会对IP数据报的源IP在路由表中进行反向路由查找，如果端口与来源不符，则视作伪造报文，将其丢弃，VPN Server上的路由信息如下

```
[09/25/20]seed@VM:~/.../week4$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.11.2    0.0.0.0         UG    100    0        0 ens33
0.0.0.0         192.168.22.1    0.0.0.0         UG    101    0        0 ens38
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 ens33
192.168.11.0    0.0.0.0         255.255.255.0   U     100    0        0 ens33
192.168.22.0    0.0.0.0         255.255.255.0   U     100    0        0 ens38
192.168.53.0    0.0.0.0         255.255.255.0   U     0      0        0 server
```

```
1 $ sudo ip route add 192.168.13.0/24 dev server
```

修改后Host U与Host V能够成功通信。

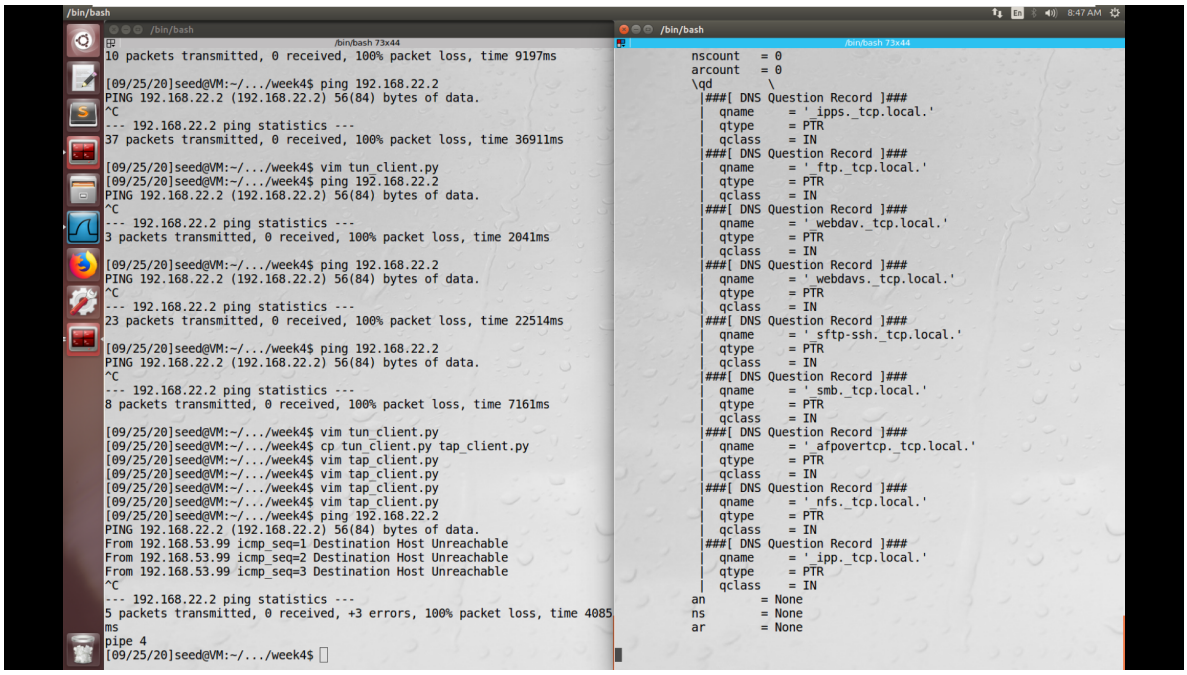
Task 9: Experiment with the TAP Interface

tap_client.py

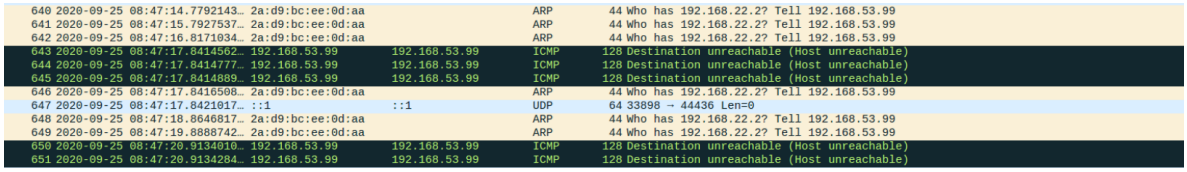
```
1 #!/usr/bin/python3
2
3 import fcntl
4 import struct
5 import os
6 import time
7 from scapy.all import *
8
```

```
TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

IP_C = "0.0.0.0"
PORT = 9090

# Create the tun interface
tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tap%d', IFF_TAP | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tap, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.22.0/24 dev {} via
192.168.53.99".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_C,PORT))
while True:
    # this will block until at least one interface is ready
    packet = os.read(tap, 2048)
    if True:
        ether = Ether(packet)
        ether.show()
```

测试发现无法ping通



原因是TAP在MAC层接管，不断查询192.168.22.2的物理地址，但由于是虚拟网络，所以不响应ARP，因此无法发出报文



640	2020-09-25 08:47:14.7792143…	2a:d9:bc:ee:0d:aa		ARP	44	Who has 192.168.22.2? Tell 192.168.53.99
641	2020-09-25 08:47:15.7927537…	2a:d9:bc:ee:0d:aa		ARP	44	Who has 192.168.22.2? Tell 192.168.53.99
642	2020-09-25 08:47:16.8171034…	2a:d9:bc:ee:0d:aa		ARP	44	Who has 192.168.22.2? Tell 192.168.53.99
643	2020-09-25 08:47:17.8414562…	192.168.53.99	192.168.53.99	ICMP	128	Destination unreachable (Host unreachable)
644	2020-09-25 08:47:17.8414777…	192.168.53.99	192.168.53.99	ICMP	128	Destination unreachable (Host unreachable)
645	2020-09-25 08:47:17.8414889…	192.168.53.99	192.168.53.99	ICMP	128	Destination unreachable (Host unreachable)
646	2020-09-25 08:47:17.8416508…	2a:d9:bc:ee:0d:aa		ARP	44	Who has 192.168.22.2? Tell 192.168.53.99
647	2020-09-25 08:47:17.8421017…	::1	::1	UDP	64	33898 → 44436 Len=0
648	2020-09-25 08:47:18.8646817…	2a:d9:bc:ee:0d:aa		ARP	44	Who has 192.168.22.2? Tell 192.168.53.99
649	2020-09-25 08:47:19.8888742…	2a:d9:bc:ee:0d:aa		ARP	44	Who has 192.168.22.2? Tell 192.168.53.99
650	2020-09-25 08:47:20.9134010…	192.168.53.99	192.168.53.99	ICMP	128	Destination unreachable (Host unreachable)
651	2020-09-25 08:47:20.9134284…	192.168.53.99	192.168.53.99	ICMP	128	Destination unreachable (Host unreachable)