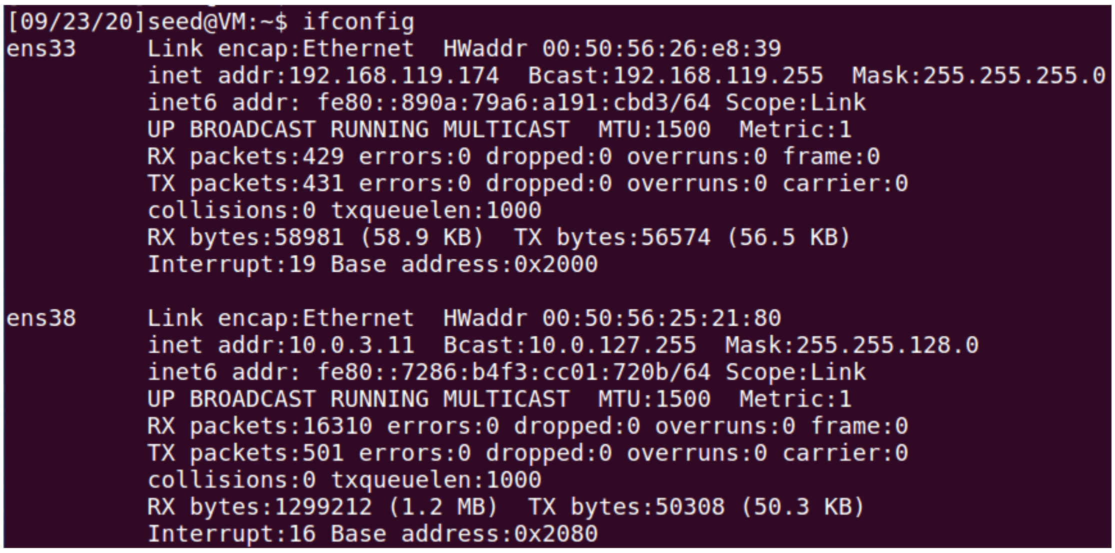
Taks 1

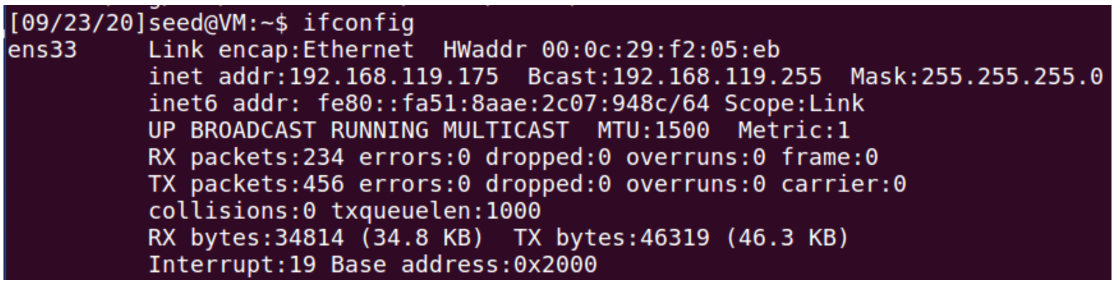
实验环境

VPN Server　192.168.119.174　10.0.3.11



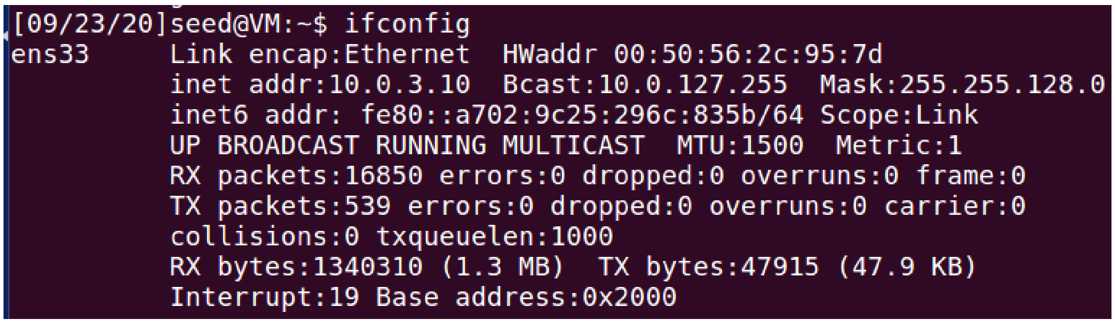
```
[09/23/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:50:56:26:e8:39
          inet addr:192.168.119.174  Bcast:192.168.119.255  Mask:255.255.255.0
          inet6 addr: fe80::890a:79a6:a191:cbd3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:429 errors:0 dropped:0 overruns:0 frame:0
          TX packets:431 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:58981 (58.9 KB)  TX bytes:56574 (56.5 KB)
          Interrupt:19 Base address:0x2000

ens38     Link encap:Ethernet  HWaddr 00:50:56:25:21:80
          inet addr:10.0.3.11  Bcast:10.0.127.255  Mask:255.255.128.0
          inet6 addr: fe80::7286:b4f3:cc01:720b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16310 errors:0 dropped:0 overruns:0 frame:0
          TX packets:501 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1299212 (1.2 MB)  TX bytes:50308 (50.3 KB)
          Interrupt:16 Base address:0x2080
```

Host V　192.168.119.175



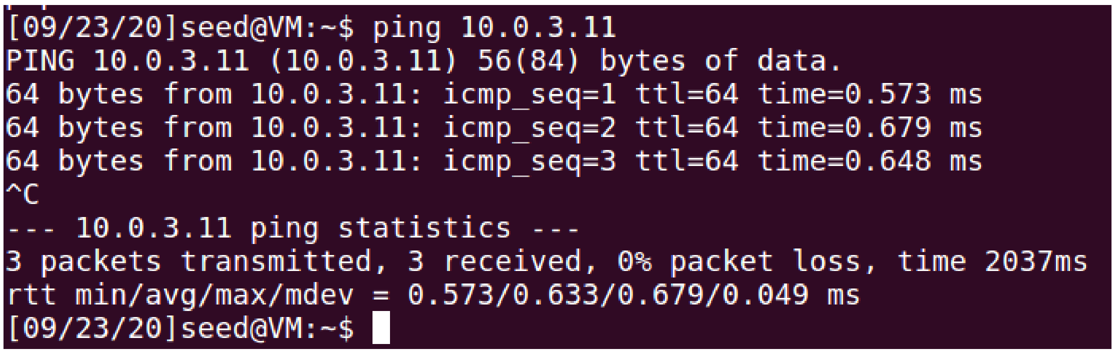
```
[09/23/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:f2:05:eb
          inet addr:192.168.119.175  Bcast:192.168.119.255  Mask:255.255.255.0
          inet6 addr: fe80::fa51:8aae:2c07:948c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:234 errors:0 dropped:0 overruns:0 frame:0
          TX packets:456 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34814 (34.8 KB)  TX bytes:46319 (46.3 KB)
          Interrupt:19 Base address:0x2000
```

Host U　10.0.3.10



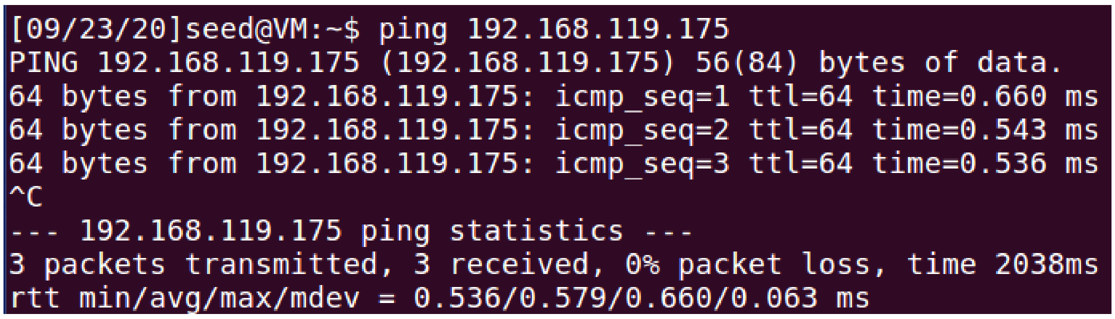
```
[09/23/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:50:56:2c:95:7d
          inet addr:10.0.3.10  Bcast:10.0.127.255  Mask:255.255.128.0
          inet6 addr: fe80::a702:9c25:296c:835b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16850 errors:0 dropped:0 overruns:0 frame:0
          TX packets:539 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1340310 (1.3 MB)  TX bytes:47915 (47.9 KB)
          Interrupt:19 Base address:0x2000
```

Host U ping VPN Server，可以 ping 通



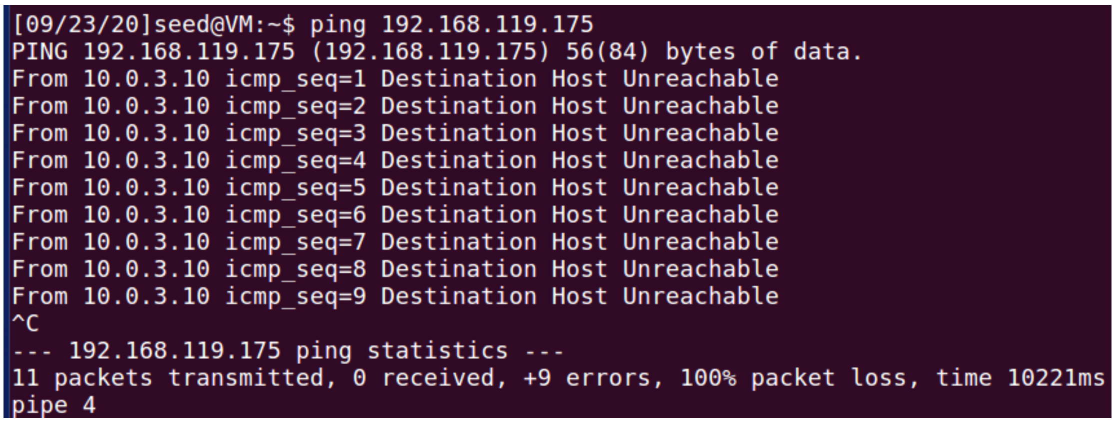
```
[09/23/20]seed@VM:~$ ping 10.0.3.11
PING 10.0.3.11 (10.0.3.11) 56(84) bytes of data.
64 bytes from 10.0.3.11: icmp_seq=1 ttl=64 time=0.573 ms
64 bytes from 10.0.3.11: icmp_seq=2 ttl=64 time=0.679 ms
64 bytes from 10.0.3.11: icmp_seq=3 ttl=64 time=0.648 ms
^C
--- 10.0.3.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.573/0.633/0.679/0.049 ms
[09/23/20]seed@VM:~$
```

VPN Server ping Host V，可以 ping 通



```
[09/23/20]seed@VM:~$ ping 192.168.119.175
PING 192.168.119.175 (192.168.119.175) 56(84) bytes of data.
64 bytes from 192.168.119.175: icmp_seq=1 ttl=64 time=0.660 ms
64 bytes from 192.168.119.175: icmp_seq=2 ttl=64 time=0.543 ms
64 bytes from 192.168.119.175: icmp_seq=3 ttl=64 time=0.536 ms
^C
--- 192.168.119.175 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2038ms
rtt min/avg/max/mdev = 0.536/0.579/0.660/0.063 ms
```

Host U ping Host V，ping 不通

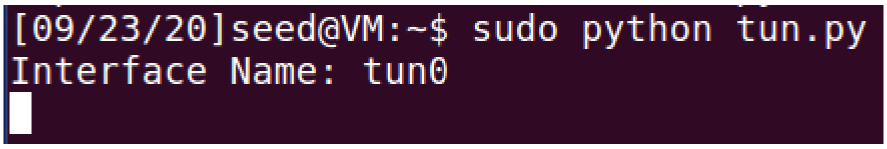


```
[09/23/20]seed@VM:~$ ping 192.168.119.175
PING 192.168.119.175 (192.168.119.175) 56(84) bytes of data.
From 10.0.3.10 icmp_seq=1 Destination Host Unreachable
From 10.0.3.10 icmp_seq=2 Destination Host Unreachable
From 10.0.3.10 icmp_seq=3 Destination Host Unreachable
From 10.0.3.10 icmp_seq=4 Destination Host Unreachable
From 10.0.3.10 icmp_seq=5 Destination Host Unreachable
From 10.0.3.10 icmp_seq=6 Destination Host Unreachable
From 10.0.3.10 icmp_seq=7 Destination Host Unreachable
From 10.0.3.10 icmp_seq=8 Destination Host Unreachable
From 10.0.3.10 icmp_seq=9 Destination Host Unreachable
^C
--- 192.168.119.175 ping statistics ---
11 packets transmitted, 0 received, +9 errors, 100% packet loss, time 10221ms
pipe 4
```

Task 2

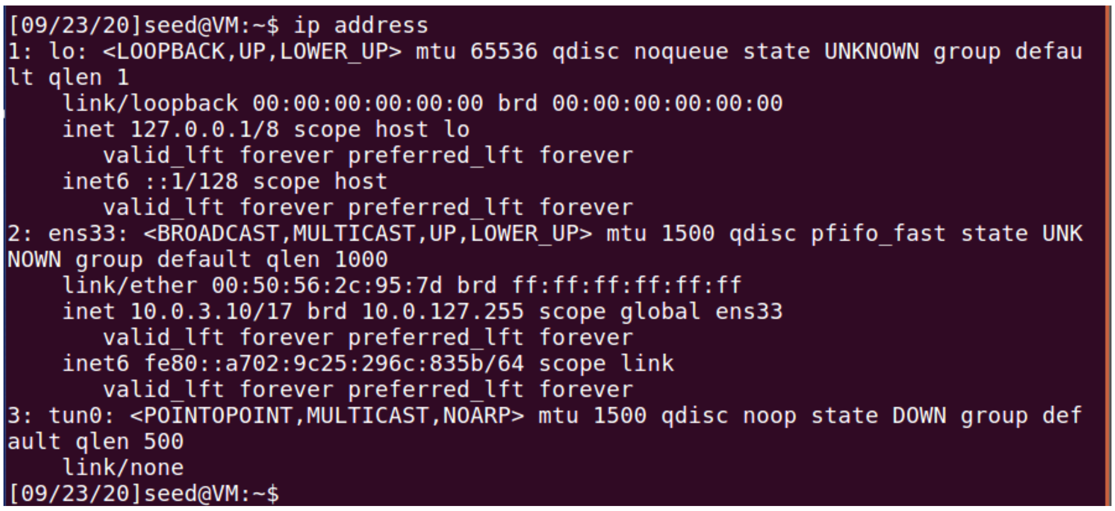
Task 2.a

在 Host U 上运行 tun.py



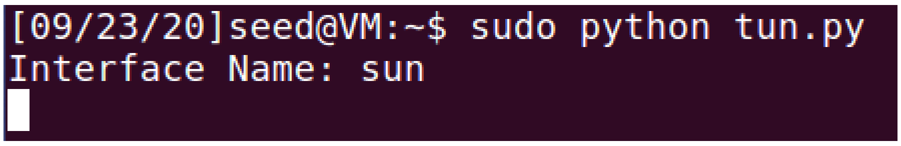
```
[09/23/20]seed@VM:~$ sudo python tun.py
Interface Name: tun0
```

另起一个终端，运行 ip address



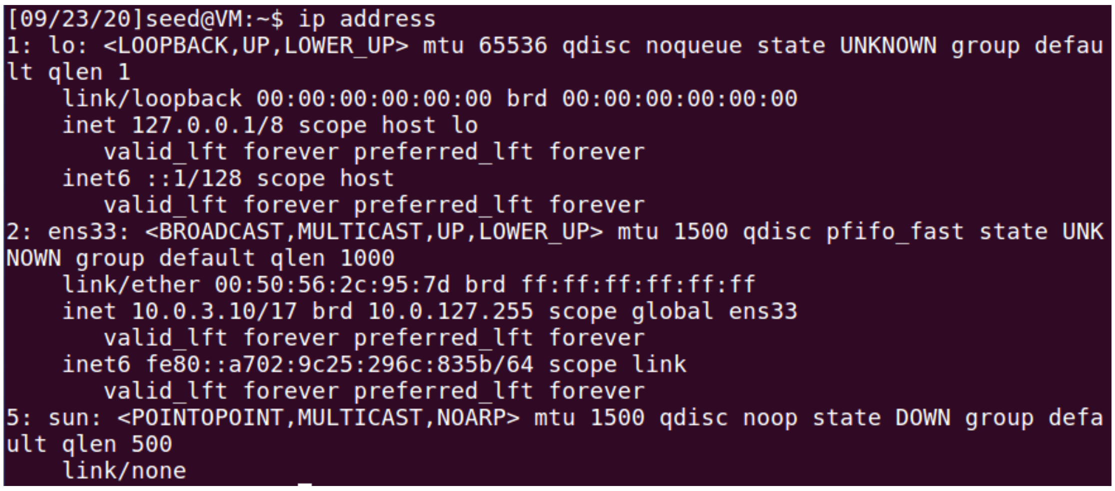
```
[09/23/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNK
NOWN group default qlen 1000
    link/ether 00:50:56:2c:95:7d brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.10/17 brd 10.0.127.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::a702:9c25:296c:835b/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group def
ault qlen 500
    link/none
[09/23/20]seed@VM:~$
```

修改 tun.py 后运行



```
[09/23/20]seed@VM:~$ sudo python tun.py
Interface Name: sun
```

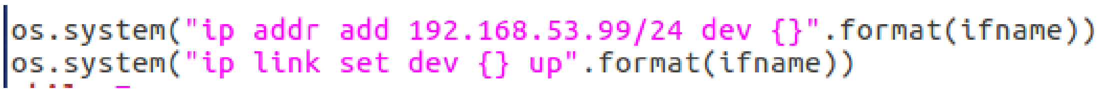
在另一终端查看



```
[09/23/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt 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 state UNK
NOWN group default qlen 1000
    link/ether 00:50:56:2c:95:7d brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.10/17 brd 10.0.127.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::a702:9c25:296c:835b/64 scope link
       valid_lft forever preferred_lft forever
5: sun: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defa
ult qlen 500
    link/none
```

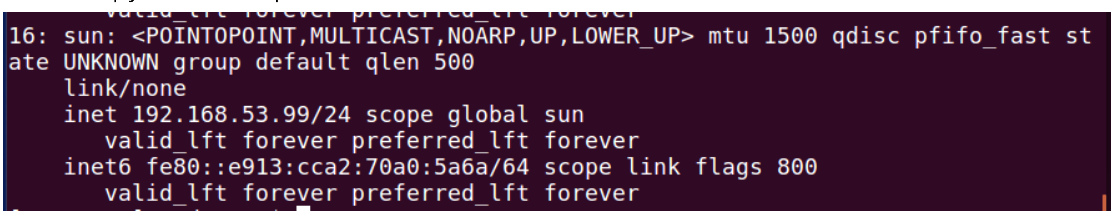
Task 2.b

在 tun.py 中加入以下代码



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

运行 tun.py 后，使用 ip address 查看

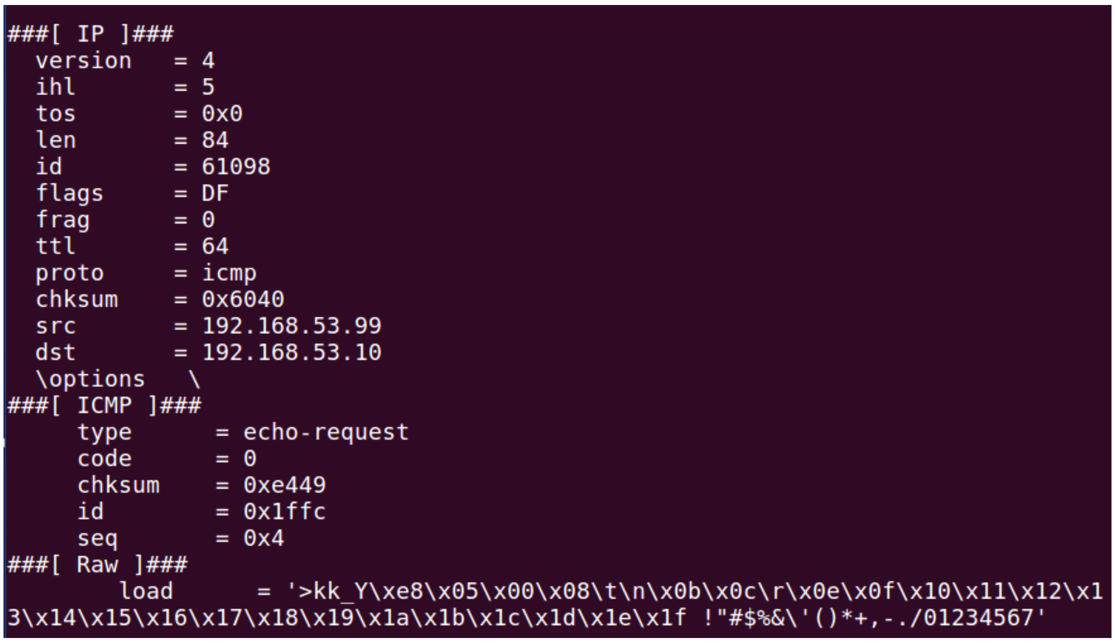


```
16: sun: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast st
ate UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global sun
       valid_lft forever preferred_lft forever
    inet6 fe80::e913:cca2:70a0:5a6a/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

可以看到接口增加了 ip 地址

Task 2.c

在 Host U 上 ping 192.168.53.10， tun.py 输出如下，

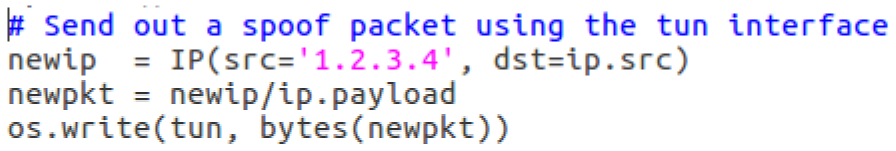


```
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 61098
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0x6040
  src       = 192.168.53.99
  dst       = 192.168.53.10
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0xe449
     id        = 0x1ffc
     seq       = 0x4
###[ Raw ]###
        load      = '>kk_Y\xe8\x05\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x1
3\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

在 Host U 上 ping 192.168.119.160，tun.py 没有输出

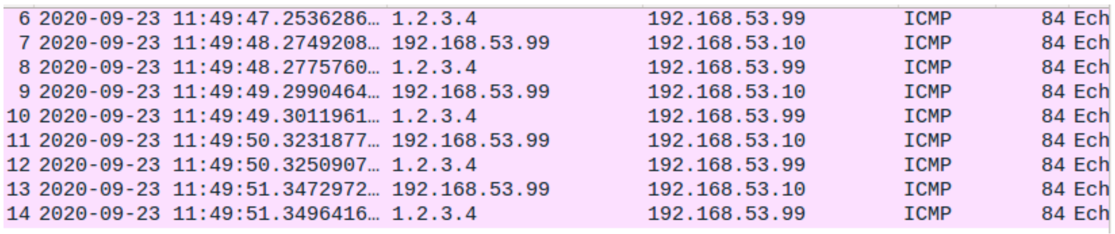
Task 2.d

在 tun.py 增加以下代码



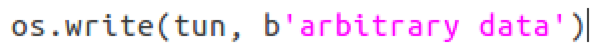
```
# Send out a spoof packet using the tun interface
newip  = IP(src='1.2.3.4', dst=ip.src)
newpkt = newip/ip.payload
os.write(tun, bytes(newpkt))
```

运行 tun.py，再 ping 192.168.53.10，wireshark 抓包结果如下。



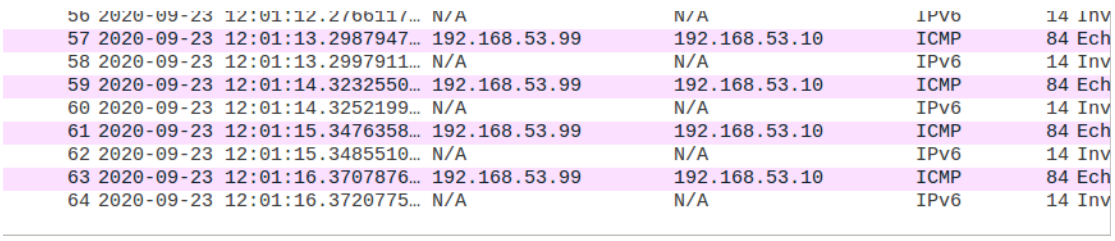
```
 6 2020-09-23 11:49:47.2536286… 1.2.3.4              192.168.53.99      ICMP      84 Ech
 7 2020-09-23 11:49:48.2749208… 192.168.53.99        192.168.53.10      ICMP      84 Ech
 8 2020-09-23 11:49:48.2775760… 1.2.3.4              192.168.53.99      ICMP      84 Ech
 9 2020-09-23 11:49:49.2990464… 192.168.53.99        192.168.53.10      ICMP      84 Ech
10 2020-09-23 11:49:49.3011961… 1.2.3.4              192.168.53.99      ICMP      84 Ech
11 2020-09-23 11:49:50.3231877… 192.168.53.99        192.168.53.10      ICMP      84 Ech
12 2020-09-23 11:49:50.3250907… 1.2.3.4              192.168.53.99      ICMP      84 Ech
13 2020-09-23 11:49:51.3472972… 192.168.53.99        192.168.53.10      ICMP      84 Ech
14 2020-09-23 11:49:51.3496416… 1.2.3.4              192.168.53.99      ICMP      84 Ech
```

将写入的内容修改为



```
os.write(tun, b'arbitrary data')
```

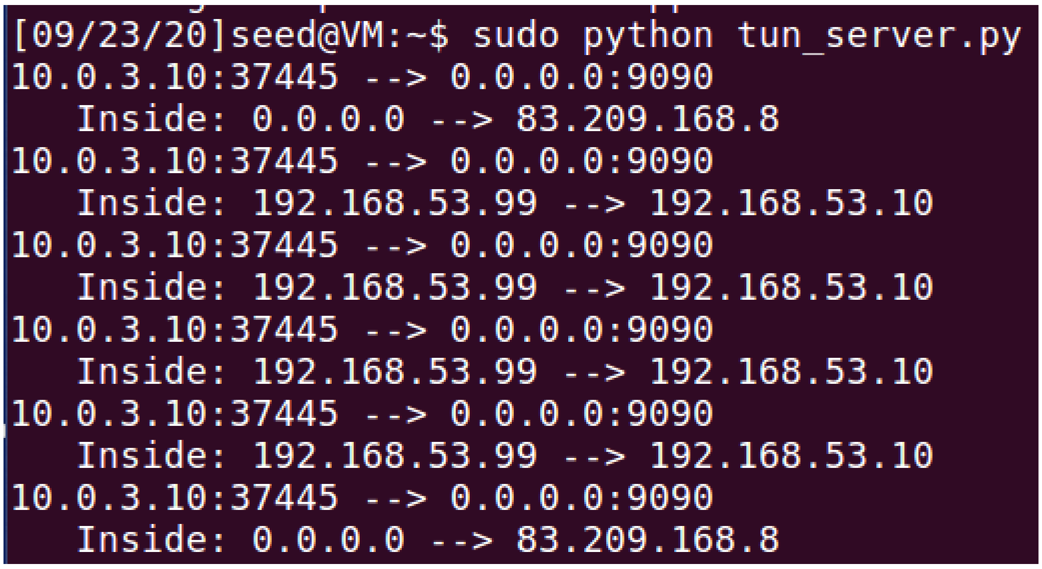
重复上述操作，wireshark 抓包结果如下。



```
56 2020-09-23 12:01:12.2766117… N/A                  N/A                IPv6      14 Inv
57 2020-09-23 12:01:13.2987947… 192.168.53.99        192.168.53.10      ICMP      84 Ech
58 2020-09-23 12:01:13.2997911… N/A                  N/A                IPv6      14 Inv
59 2020-09-23 12:01:14.3232550… 192.168.53.99        192.168.53.10      ICMP      84 Ech
60 2020-09-23 12:01:14.3252199… N/A                  N/A                IPv6      14 Inv
61 2020-09-23 12:01:15.3476358… 192.168.53.99        192.168.53.10      ICMP      84 Ech
62 2020-09-23 12:01:15.3485510… N/A                  N/A                IPv6      14 Inv
63 2020-09-23 12:01:16.3707876… 192.168.53.99        192.168.53.10      ICMP      84 Ech
64 2020-09-23 12:01:16.3720775… N/A                  N/A                IPv6      14 Inv
```

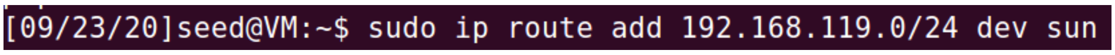
Task 3

在 VPN Server 上运行 tun_server.py，Host U 上运行 tun_client.py，再 ping 192.168.53.10，VPN Server 上输出如下。



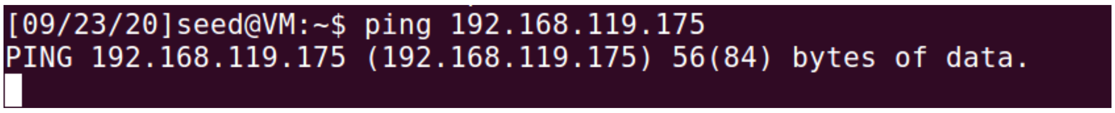
```
[09/23/20]seed@VM:~$ sudo python tun_server.py
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 0.0.0.0 --> 83.209.168.8
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.10
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.10
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.10
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.10
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 0.0.0.0 --> 83.209.168.8
```

在 Host U 中添加路由



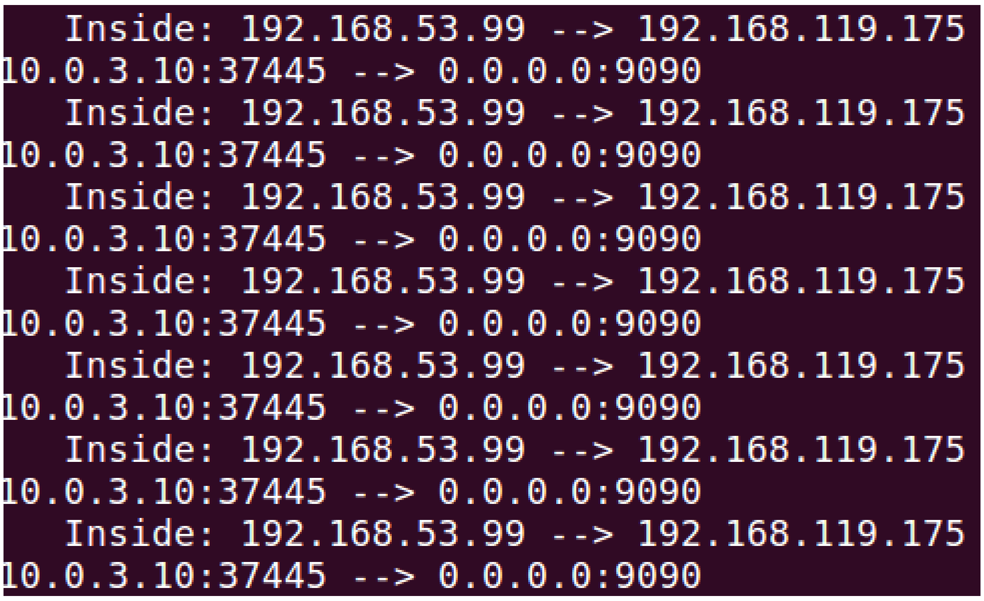
```
[09/23/20]seed@VM:~$ sudo ip route add 192.168.119.0/24 dev sun
```

再 ping Host V



```
[09/23/20]seed@VM:~$ ping 192.168.119.175
PING 192.168.119.175 (192.168.119.175) 56(84) bytes of data.
```

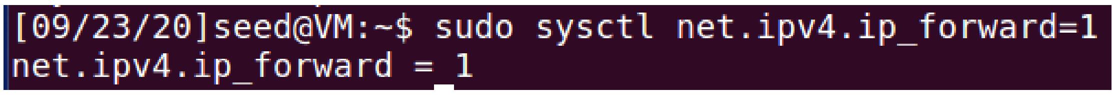
VPN Server 显示以下输出，与预期效果一致。



```
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:37445 --> 0.0.0.0:9090
```

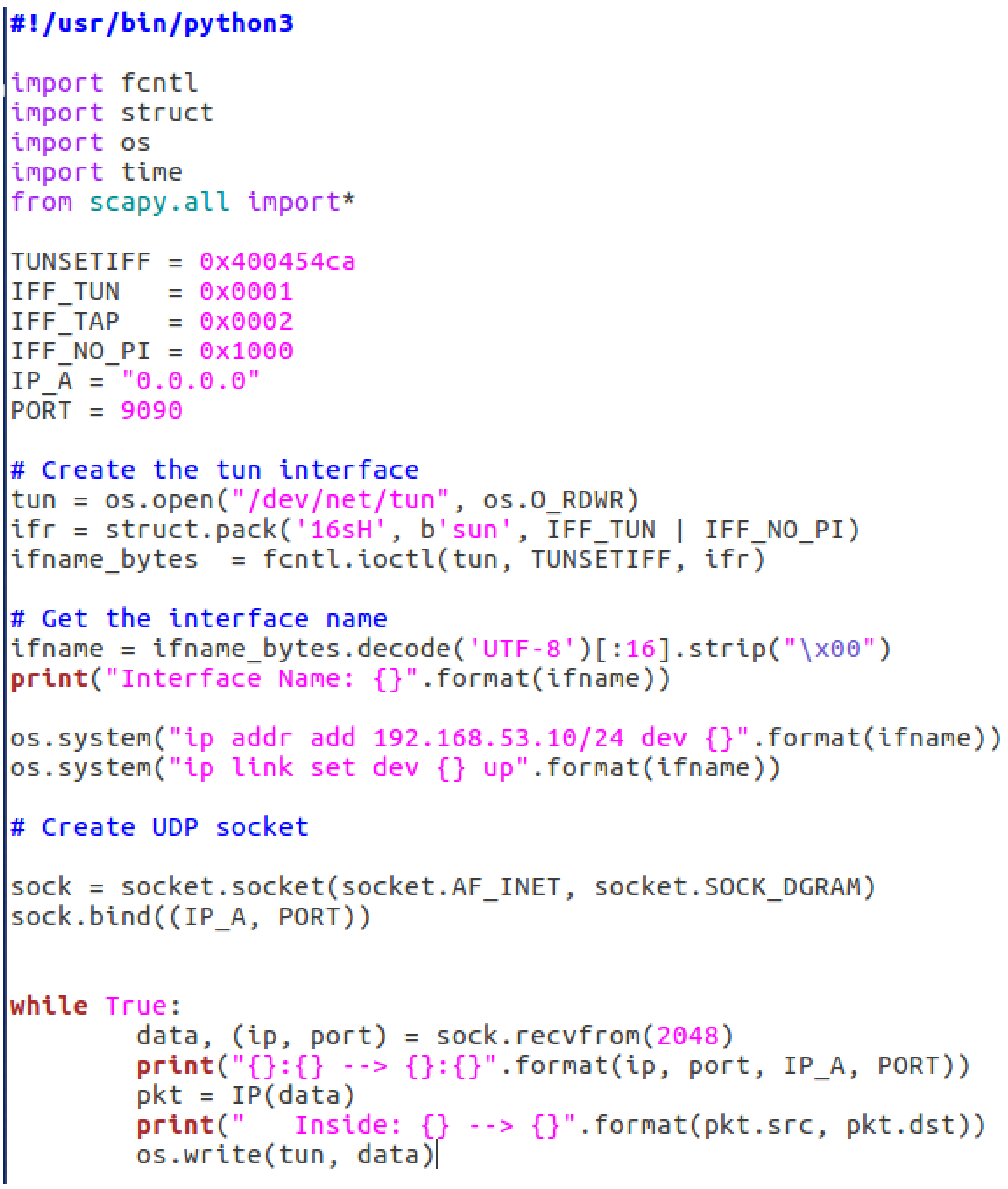
Task 4

在 VPN Server 上开启 IP 转发



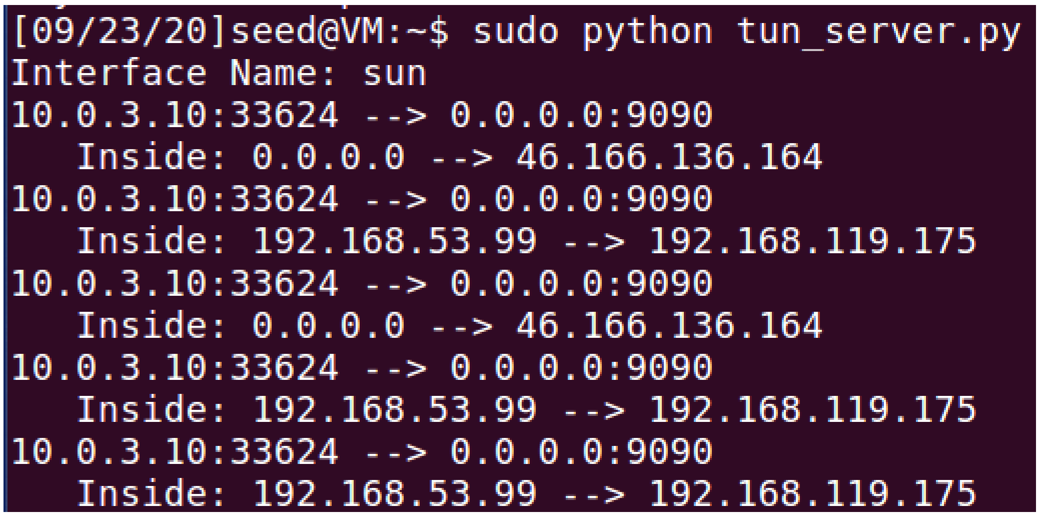
```
[09/23/20]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

修改 tun_server.py 后并运行，且 Host U 运行 tun_client.py



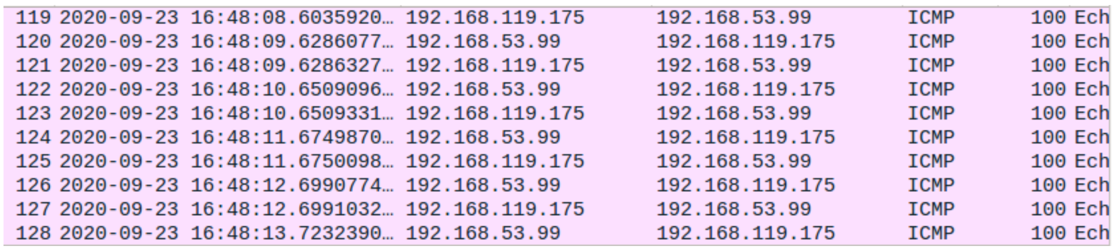
```python
#!/usr/bin/python3

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

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000
IP_A = "0.0.0.0"
PORT = 9090

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'sun', 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 192.168.53.10/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Create UDP socket

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))
        os.write(tun, data)
```

Host U ping V，VPN Server 上有以下输出



```
[09/23/20]seed@VM:~$ sudo python tun_server.py
Interface Name: sun
10.0.3.10:33624 --> 0.0.0.0:9090
   Inside: 0.0.0.0 --> 46.166.136.164
10.0.3.10:33624 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:33624 --> 0.0.0.0:9090
   Inside: 0.0.0.0 --> 46.166.136.164
10.0.3.10:33624 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
10.0.3.10:33624 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.119.175
```

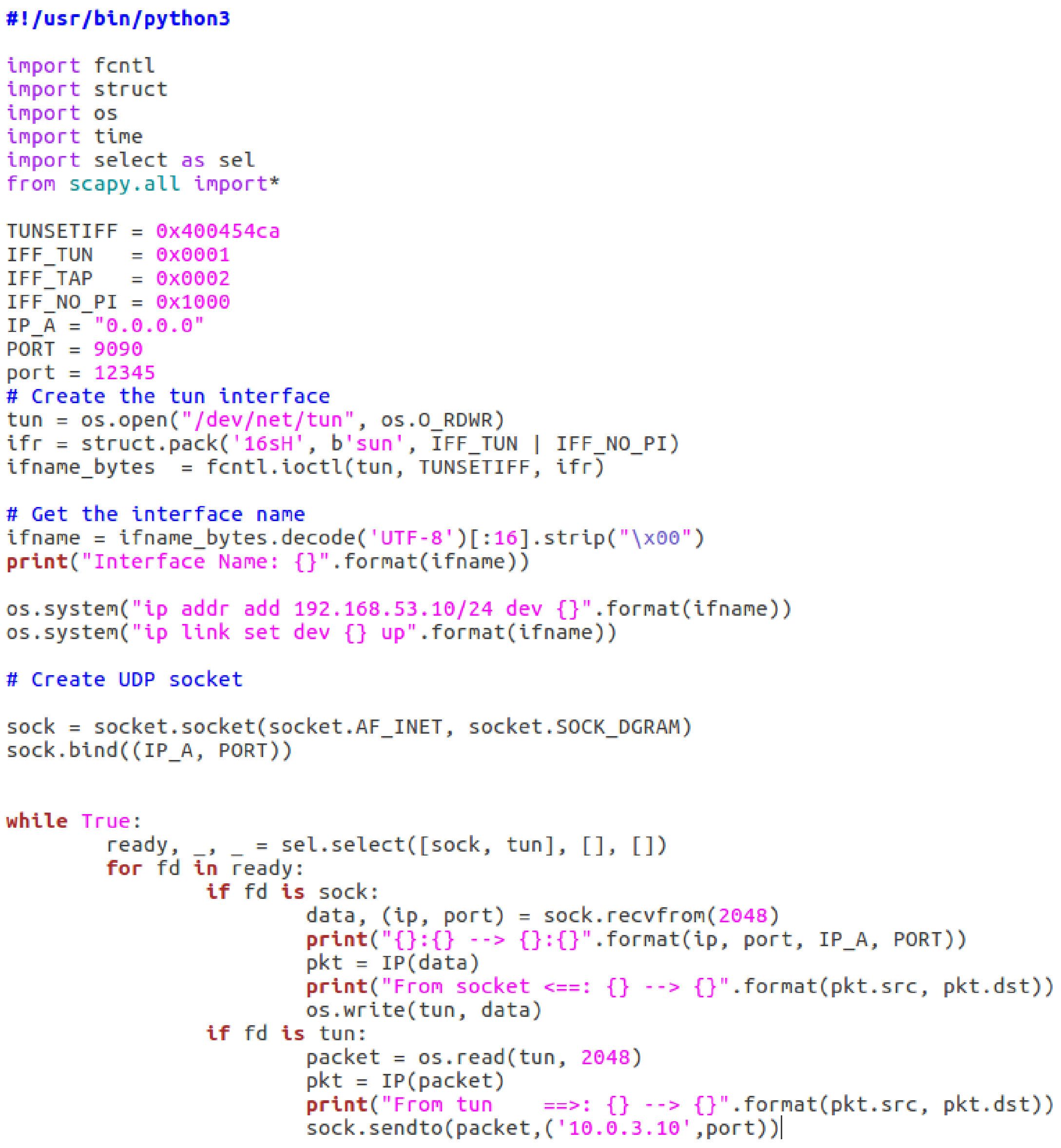
Host V 中 wireshark 抓包结果如下，可以看到 Host V 收到了来自 U 的 ICMP 报文



```
119 2020-09-23 16:48:08.6035920… 192.168.119.175     192.168.53.99       ICMP      100 Ech
120 2020-09-23 16:48:09.6286077… 192.168.53.99       192.168.119.175     ICMP      100 Ech
121 2020-09-23 16:48:09.6286327… 192.168.119.175     192.168.53.99       ICMP      100 Ech
122 2020-09-23 16:48:10.6509096… 192.168.53.99       192.168.119.175     ICMP      100 Ech
123 2020-09-23 16:48:10.6509331… 192.168.119.175     192.168.53.99       ICMP      100 Ech
124 2020-09-23 16:48:11.6749870… 192.168.53.99       192.168.119.175     ICMP      100 Ech
125 2020-09-23 16:48:11.6750098… 192.168.119.175     192.168.53.99       ICMP      100 Ech
126 2020-09-23 16:48:12.6990774… 192.168.53.99       192.168.119.175     ICMP      100 Ech
127 2020-09-23 16:48:12.6991032… 192.168.119.175     192.168.53.99       ICMP      100 Ech
128 2020-09-23 16:48:13.7232390… 192.168.53.99       192.168.119.175     ICMP      100 Ech
```

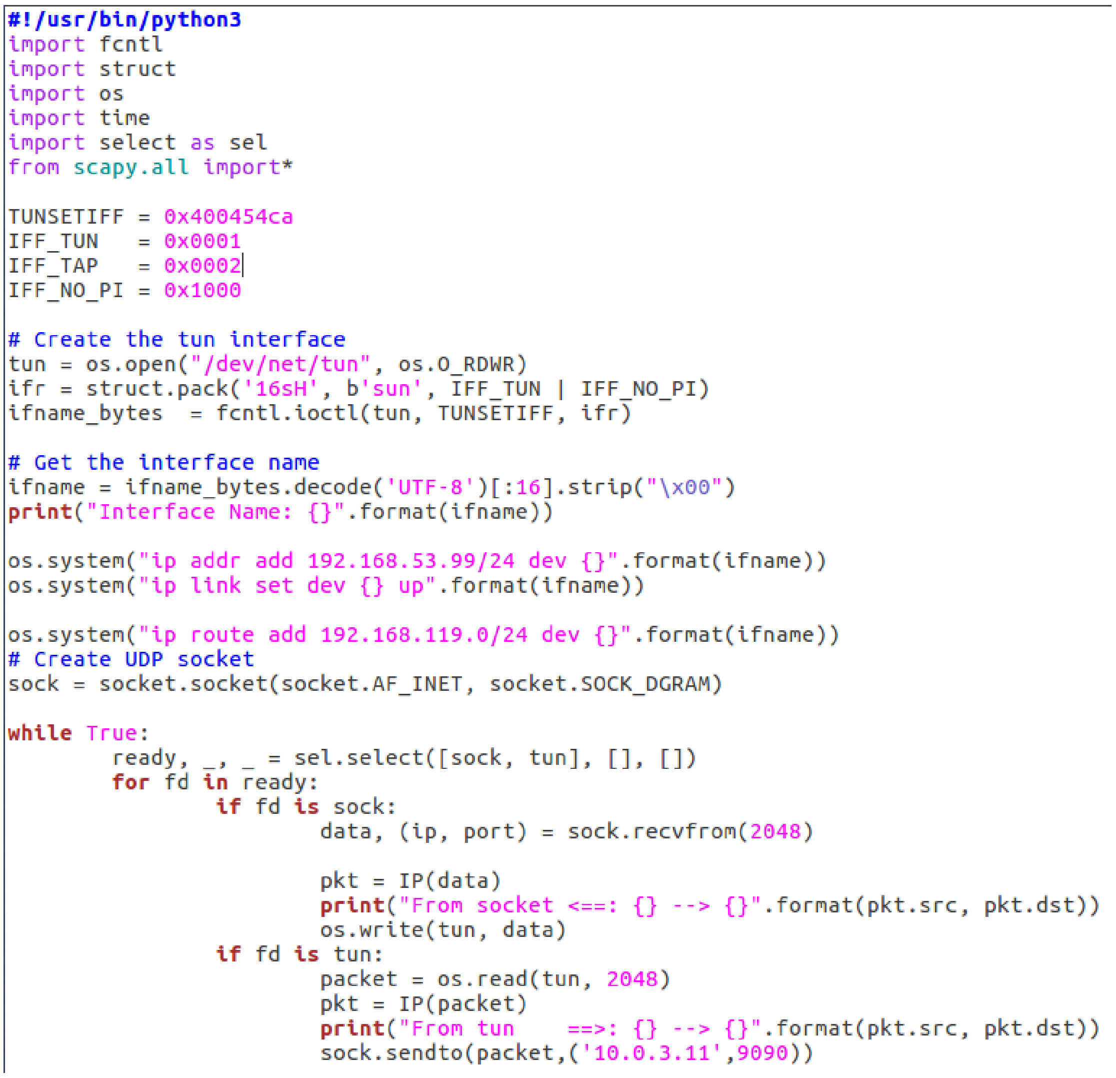
Task5

VPN Server 中，修改 tun_server.py



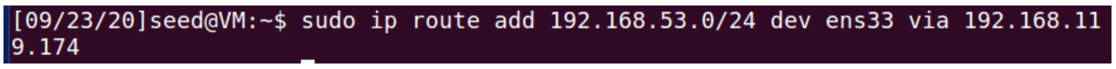
```python
#!/usr/bin/python3

import fcntl
import struct
import os
import time
import select as sel
from scapy.all import*

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000
IP_A = "0.0.0.0"
PORT = 9090
port = 12345
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'sun', 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 192.168.53.10/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Create UDP socket

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


while True:
        ready, _, _ = sel.select([sock, tun], [], [])
        for fd in ready:
                if fd is sock:
                        data, (ip, port) = sock.recvfrom(2048)
                        print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
                        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,('10.0.3.10',port))
```

Host U 中，修改 tun_client.py



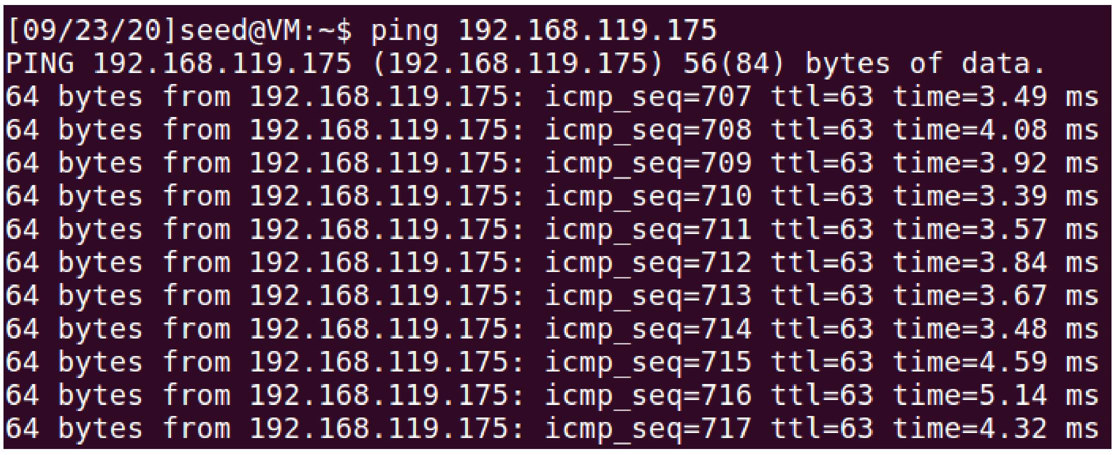
```python
#!/usr/bin/python3
import fcntl
import struct
import os
import time
import select as sel
from scapy.all import*

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'sun', 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))

os.system("ip route add 192.168.119.0/24 dev {}".format(ifname))
# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
        ready, _, _ = sel.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,('10.0.3.11',9090))
```

在 Host V 中添加 VPN Server 的 ip



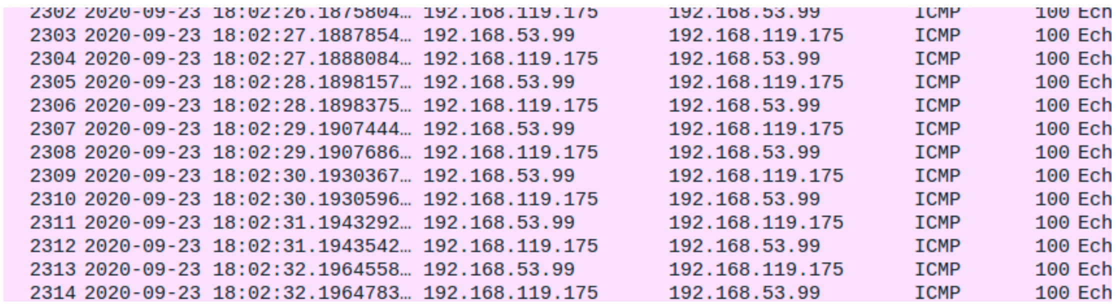
```
[09/23/20]seed@VM:~$ sudo ip route add 192.168.53.0/24 dev ens33 via 192.168.11
9.174
```

运行 tun_server.py tun_client.py，Host U ping V



```
[09/23/20]seed@VM:~$ ping 192.168.119.175
PING 192.168.119.175 (192.168.119.175) 56(84) bytes of data.
64 bytes from 192.168.119.175: icmp_seq=707 ttl=63 time=3.49 ms
64 bytes from 192.168.119.175: icmp_seq=708 ttl=63 time=4.08 ms
64 bytes from 192.168.119.175: icmp_seq=709 ttl=63 time=3.92 ms
64 bytes from 192.168.119.175: icmp_seq=710 ttl=63 time=3.39 ms
64 bytes from 192.168.119.175: icmp_seq=711 ttl=63 time=3.57 ms
64 bytes from 192.168.119.175: icmp_seq=712 ttl=63 time=3.84 ms
64 bytes from 192.168.119.175: icmp_seq=713 ttl=63 time=3.67 ms
64 bytes from 192.168.119.175: icmp_seq=714 ttl=63 time=3.48 ms
64 bytes from 192.168.119.175: icmp_seq=715 ttl=63 time=4.59 ms
64 bytes from 192.168.119.175: icmp_seq=716 ttl=63 time=5.14 ms
64 bytes from 192.168.119.175: icmp_seq=717 ttl=63 time=4.32 ms
```

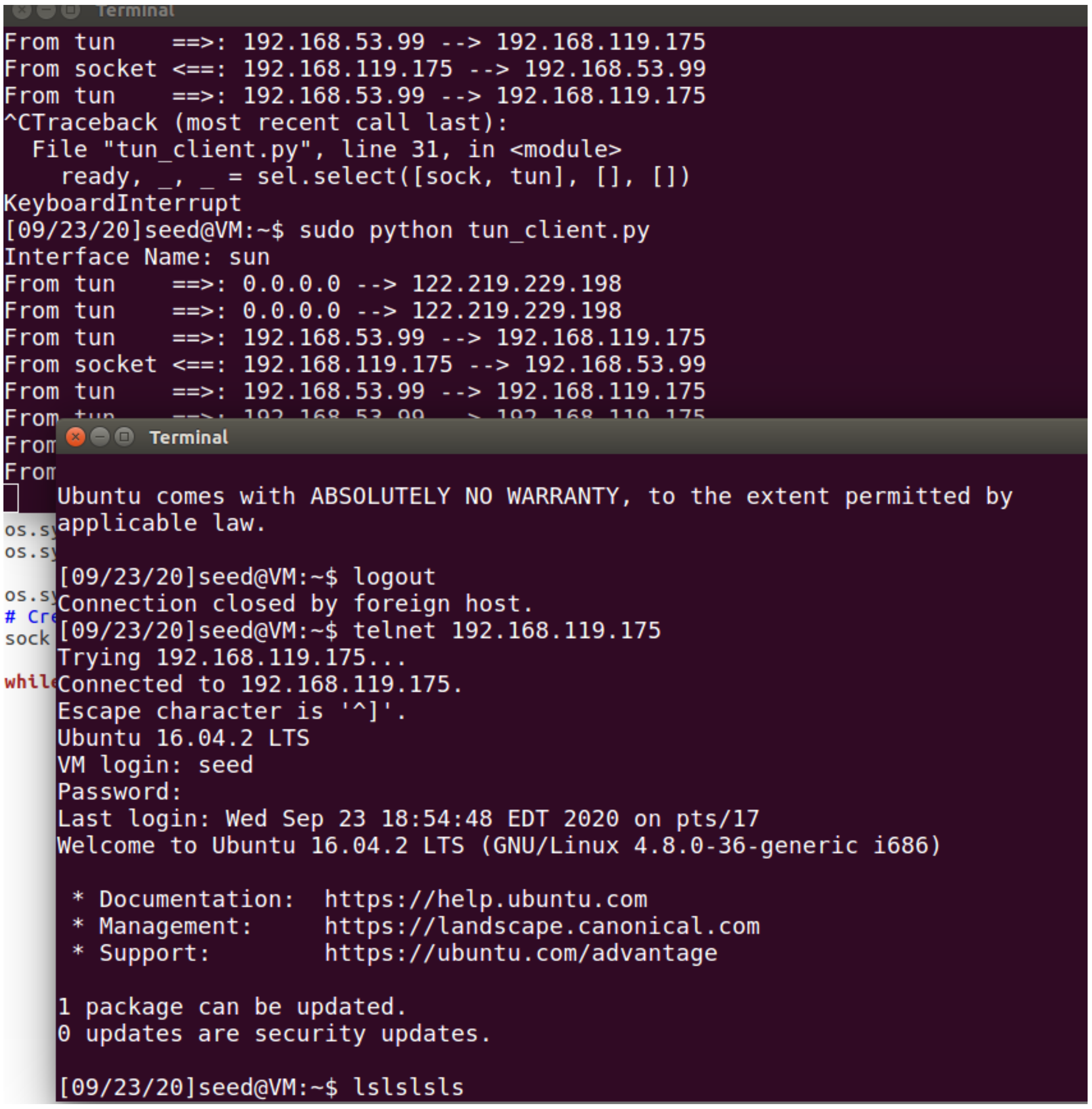
且 wireshark 抓包也能看到 ICMP 的回复报文



```
2302 2020-09-23 18:02:26.1875804… 192.168.119.175    192.168.53.99      ICMP     100 Ech
2303 2020-09-23 18:02:27.1887854… 192.168.53.99      192.168.119.175    ICMP     100 Ech
2304 2020-09-23 18:02:27.1888084… 192.168.119.175    192.168.53.99      ICMP     100 Ech
2305 2020-09-23 18:02:28.1898157… 192.168.53.99      192.168.119.175    ICMP     100 Ech
2306 2020-09-23 18:02:28.1898375… 192.168.119.175    192.168.53.99      ICMP     100 Ech
2307 2020-09-23 18:02:29.1907444… 192.168.53.99      192.168.119.175    ICMP     100 Ech
2308 2020-09-23 18:02:29.1907686… 192.168.119.175    192.168.53.99      ICMP     100 Ech
2309 2020-09-23 18:02:30.1930367… 192.168.53.99      192.168.119.175    ICMP     100 Ech
2310 2020-09-23 18:02:30.1930596… 192.168.119.175    192.168.53.99      ICMP     100 Ech
2311 2020-09-23 18:02:31.1943292… 192.168.53.99      192.168.119.175    ICMP     100 Ech
2312 2020-09-23 18:02:31.1943542… 192.168.119.175    192.168.53.99      ICMP     100 Ech
2313 2020-09-23 18:02:32.1964558… 192.168.53.99      192.168.119.175    ICMP     100 Ech
2314 2020-09-23 18:02:32.1964783… 192.168.119.175    192.168.53.99      ICMP     100 Ech
```

Task 6

中断 tun_client.py，telnet 连接的终端无法输入，重新运行 tun_client.py 后恢复正常。



```
From tun    ==>: 192.168.53.99 --> 192.168.119.175
From socket <==: 192.168.119.175 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.119.175
^CTraceback (most recent call last):
  File "tun_client.py", line 31, in <module>
    ready, _, _ = sel.select([sock, tun], [], [])
KeyboardInterrupt
[09/23/20]seed@VM:~$ sudo python tun_client.py
Interface Name: sun
From tun    ==>: 0.0.0.0 --> 122.219.229.198
From tun    ==>: 0.0.0.0 --> 122.219.229.198
From tun    ==>: 192.168.53.99 --> 192.168.119.175
From socket <==: 192.168.119.175 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.119.175
```

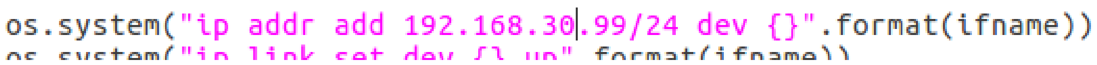
```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[09/23/20]seed@VM:~$ logout
Connection closed by foreign host.
[09/23/20]seed@VM:~$ telnet 192.168.119.175
Trying 192.168.119.175...
Connected to 192.168.119.175.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Sep 23 18:54:48 EDT 2020 on pts/17
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.

[09/23/20]seed@VM:~$ lslslsls
```

Task 7

见 Task 5 对 Host V 的操作

Task 8

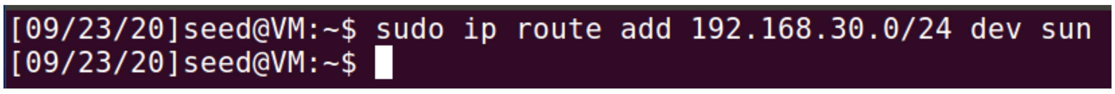
在 Host U 中修改 tun_client.py，将接口的 IP 地址改为 192.168.30.99



```
os.system("ip addr add 192.168.30.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
```

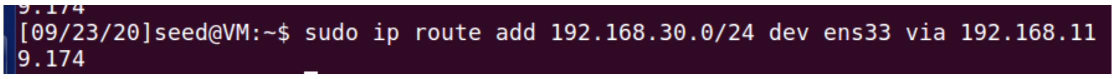
重新运行 tun_client.py，VPN Server 可以收到 UDP 包，而 Host V 收不到 ICMP 报文。

在 VPN Server 中添加路由



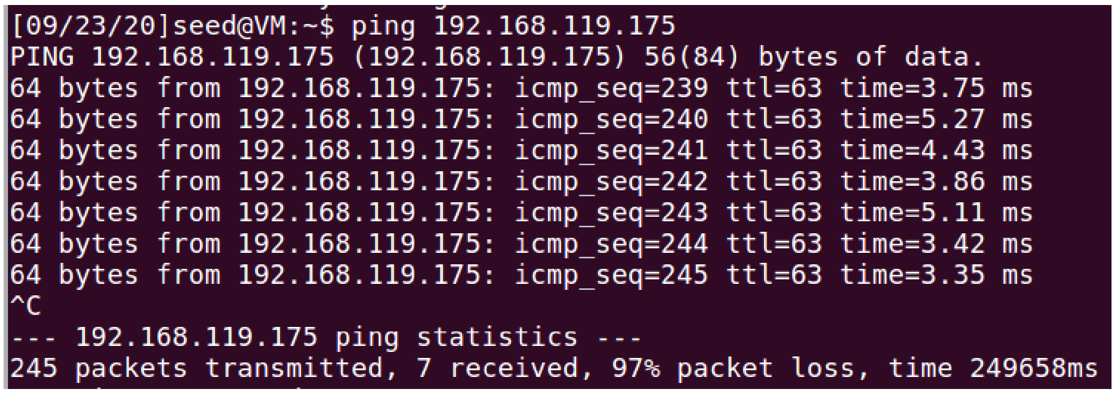
```
[09/23/20]seed@VM:~$ sudo ip route add 192.168.30.0/24 dev sun
[09/23/20]seed@VM:~$ 
```

在 Host V 中添加路由



```
[09/23/20]seed@VM:~$ sudo ip route add 192.168.30.0/24 dev ens33 via 192.168.11
9.174
```

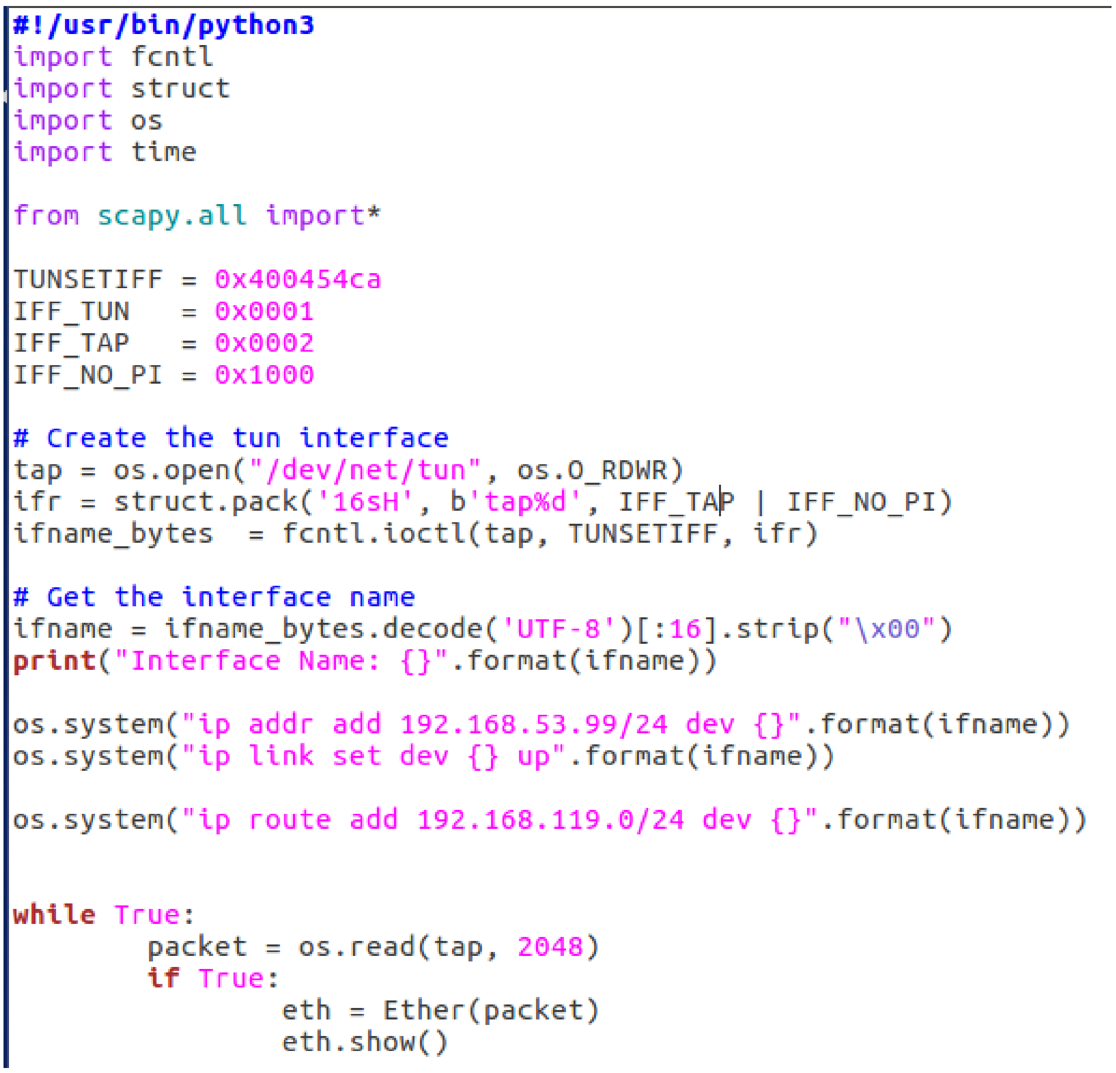
成功 ping 通



```
[09/23/20]seed@VM:~$ ping 192.168.119.175
PING 192.168.119.175 (192.168.119.175) 56(84) bytes of data.
64 bytes from 192.168.119.175: icmp_seq=239 ttl=63 time=3.75 ms
64 bytes from 192.168.119.175: icmp_seq=240 ttl=63 time=5.27 ms
64 bytes from 192.168.119.175: icmp_seq=241 ttl=63 time=4.43 ms
64 bytes from 192.168.119.175: icmp_seq=242 ttl=63 time=3.86 ms
64 bytes from 192.168.119.175: icmp_seq=243 ttl=63 time=5.11 ms
64 bytes from 192.168.119.175: icmp_seq=244 ttl=63 time=3.42 ms
64 bytes from 192.168.119.175: icmp_seq=245 ttl=63 time=3.35 ms
^C
--- 192.168.119.175 ping statistics ---
245 packets transmitted, 7 received, 97% packet loss, time 249658ms
```

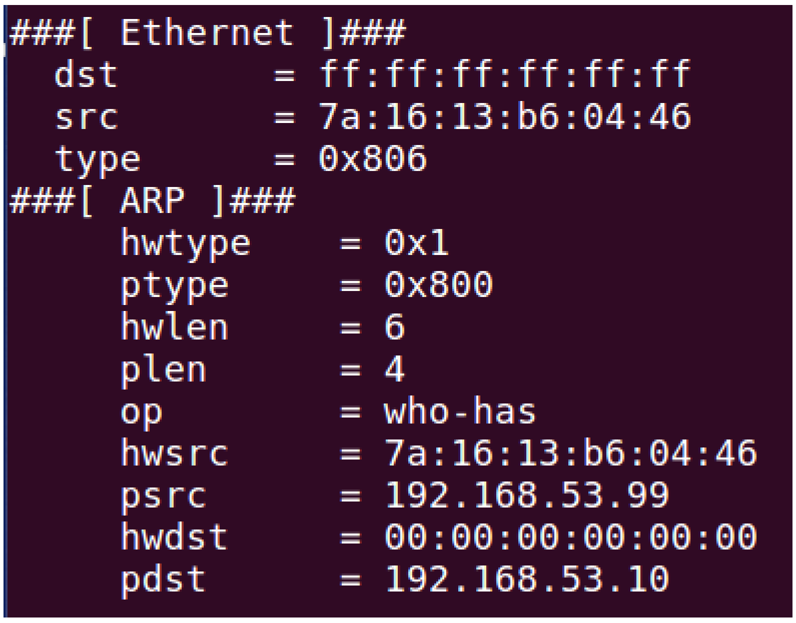
Task 9

在 Host U 上，编写并运行以下代码



```
#!/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
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 192.168.119.0/24 dev {}".format(ifname))


while True:
        packet = os.read(tap, 2048)
        if True:
                eth = Ether(packet)
                eth.show()
```

ping 192.168.53.10，结果如下



```
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 7a:16:13:b6:04:46
  type      = 0x806
###[ ARP ]###
     hwtype    = 0x1
     ptype     = 0x800
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 7a:16:13:b6:04:46
     psrc      = 192.168.53.99
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.53.10
```

可以看到显示了 ping 指令发出的 ARP 请求，查询此 IP 为那个设备所拥有。