Lab7 实验报告

57117230 刘玉洁

VPN Tunneling Lab

本实验需要用到三台虚拟机

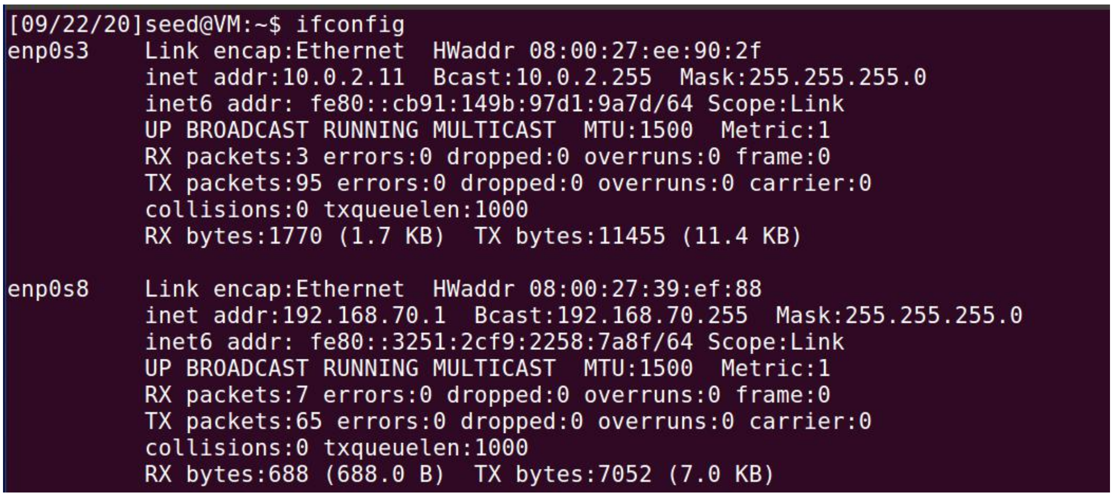
主机 U：NAT 网络 10.0.2.8

VPN 服务器：NAT 网络 10.0.2.11，内部网络 192.168.70.1

主机 V：内部网络 192.168.70.101

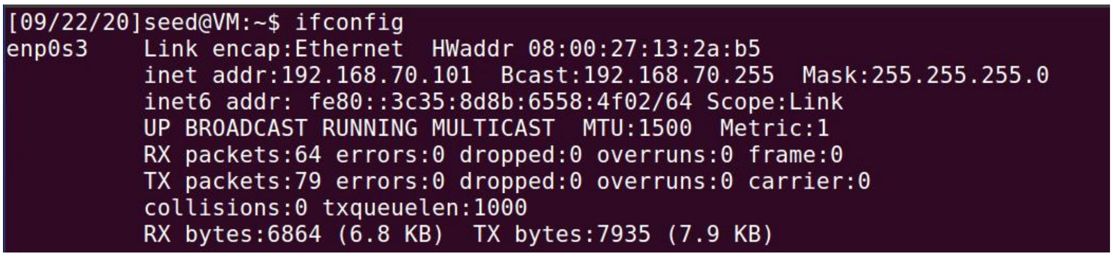
Task 1: Network Setup

配置 VPN 服务器内部网络：



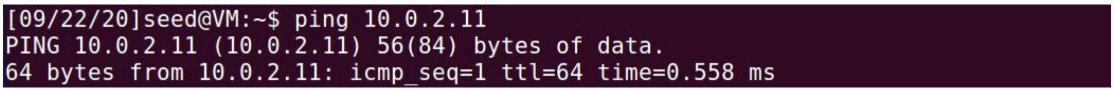
```
[09/22/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:ee:90:2f
          inet addr:10.0.2.11  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::cb91:149b:97d1:9a7d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1770 (1.7 KB)  TX bytes:11455 (11.4 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:39:ef:88
          inet addr:192.168.70.1  Bcast:192.168.70.255  Mask:255.255.255.0
          inet6 addr: fe80::3251:2cf9:2258:7a8f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:688 (688.0 B)  TX bytes:7052 (7.0 KB)
```

配置主机 V 内部网络：

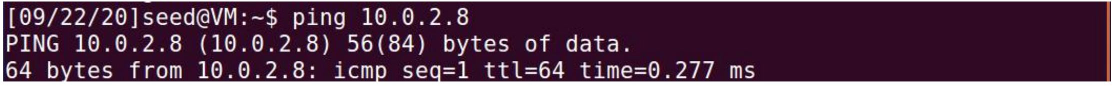


```
[09/22/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:13:2a:b5
          inet addr:192.168.70.101  Bcast:192.168.70.255  Mask:255.255.255.0
          inet6 addr: fe80::3c35:8d8b:6558:4f02/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64 errors:0 dropped:0 overruns:0 frame:0
          TX packets:79 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6864 (6.8 KB)  TX bytes:7935 (7.9 KB)
```

- 主机 U 和 VPN 服务器通信：

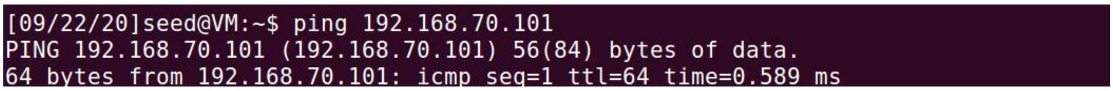


```
[09/22/20]seed@VM:~$ ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
64 bytes from 10.0.2.11: icmp_seq=1 ttl=64 time=0.558 ms
```

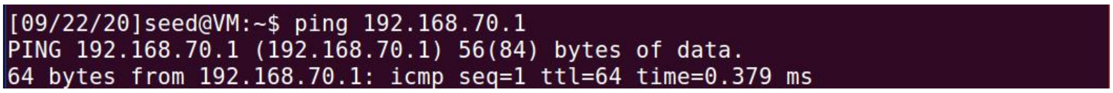


```
[09/22/20]seed@VM:~$ ping 10.0.2.8
PING 10.0.2.8 (10.0.2.8) 56(84) bytes of data.
64 bytes from 10.0.2.8: icmp_seq=1 ttl=64 time=0.277 ms
```

- 主机 V 和 VPN 服务器通信：

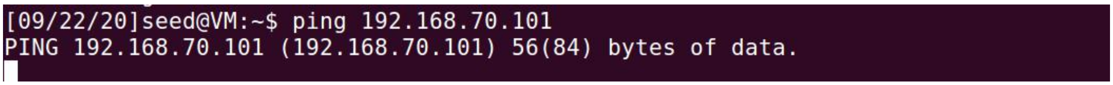


```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
64 bytes from 192.168.70.101: icmp_seq=1 ttl=64 time=0.589 ms
```

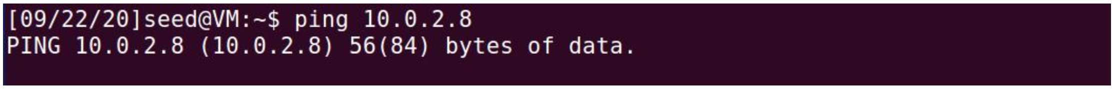


```
[09/22/20]seed@VM:~$ ping 192.168.70.1
PING 192.168.70.1 (192.168.70.1) 56(84) bytes of data.
64 bytes from 192.168.70.1: icmp_seq=1 ttl=64 time=0.379 ms
```

- 主机 V 和主机 U 无法通信：



```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
```



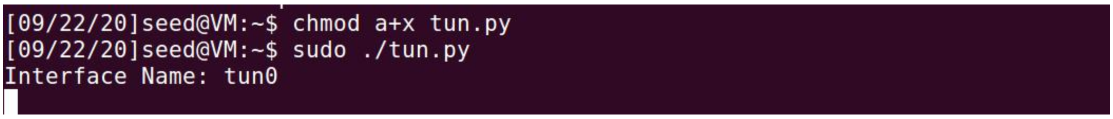
```
[09/22/20]seed@VM:~$ ping 10.0.2.8
PING 10.0.2.8 (10.0.2.8) 56(84) bytes of data.
```

网络已配置恰当。

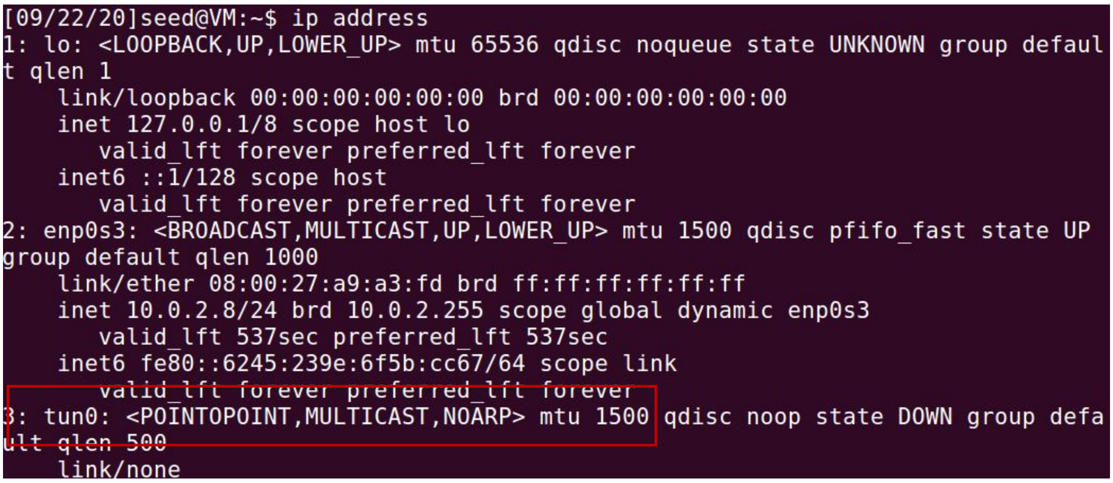
Task 2: Create and Confifigure TUN Interface

Task 2.a: Name of the Interface

运行 tun.py 程序，在另一终端执行 ip address 查看接口名称 tun0：

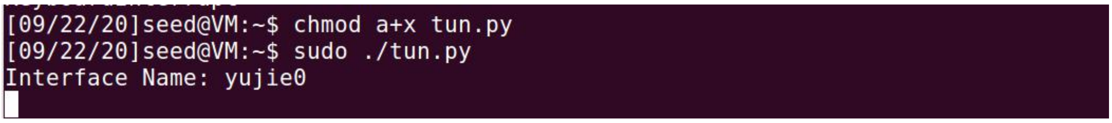


```
[09/22/20]seed@VM:~$ chmod a+x tun.py
[09/22/20]seed@VM:~$ sudo ./tun.py
Interface Name: tun0
```

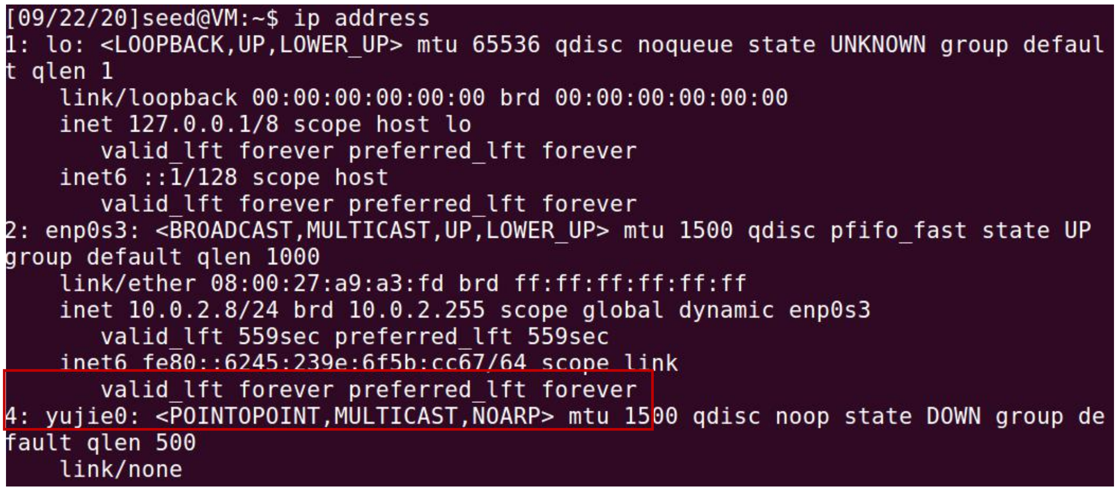


```
[09/22/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:a9:a3:fd brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.8/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 537sec preferred_lft 537sec
    inet6 fe80::6245:239e:6f5b:cc67/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defa
ult qlen 500
    link/none
```

将接口名称进行更改 yujie0，重复上述操作：



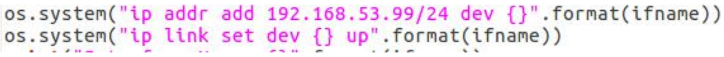
```
[09/22/20]seed@VM:~$ chmod a+x tun.py
[09/22/20]seed@VM:~$ sudo ./tun.py
Interface Name: yujie0
```



```
[09/22/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:a9:a3:fd brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.8/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 559sec preferred_lft 559sec
    inet6 fe80::6245:239e:6f5b:cc67/64 scope link
       valid_lft forever preferred_lft forever
4: yujie0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group de
fault qlen 500
    link/none
```

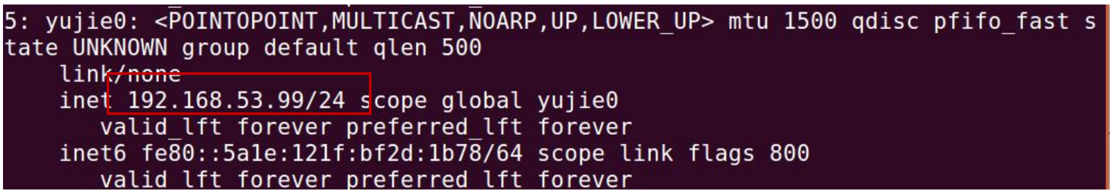
Task 2.b: Set up the TUN Interface

设置接口 IP 地址：



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

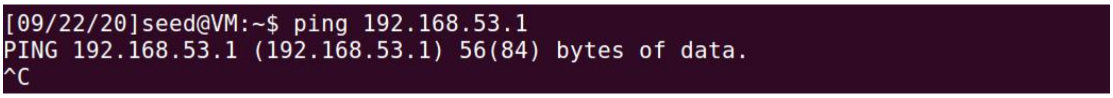
运行 tun.py 程序，在另一终端执行 ip address 查看接口信息，出现了接口 IP 地址：



```
5: yujie0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s
tate UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global yujie0
       valid_lft forever preferred_lft forever
    inet6 fe80::5a1e:121f:bf2d:1b78/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

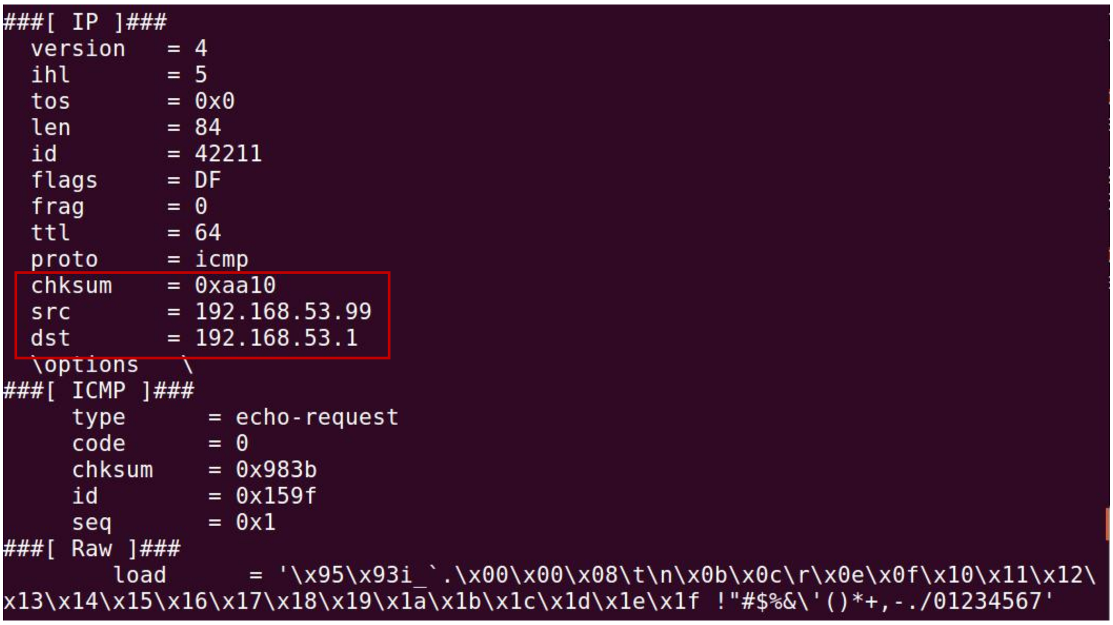
Task 2.c: Read from the TUN Interface

- ping 192.168.53.0/24 网络主机：



```
[09/22/20]seed@VM:~$ ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
```

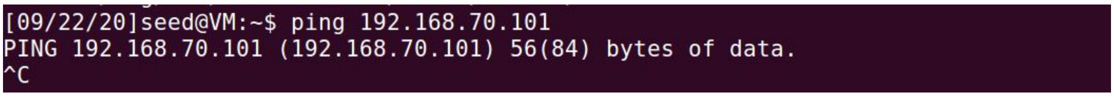
程序打印出嗅探到的报文信息：



```
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 42211
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0xaa10
  src       = 192.168.53.99
  dst       = 192.168.53.1
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x983b
     id        = 0x159f
     seq       = 0x1
###[ Raw ]###
        load      = '\x95\x93i_`.\x00\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\
x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

原因：该 tun 通道的接口设置了 192.168.53.0/24 网段的 IP 地址，该网段报文信息可以通过。

- ping 192.168.70.0/24 网络主机：



```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
^C
```

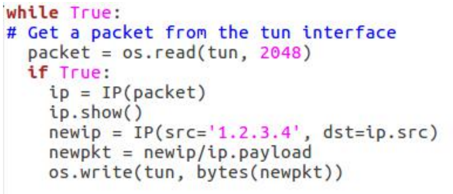
程序没有打印出与该网段相关的报文信息。

原因：该 tun 通道的接口只设置了 192.168.53.0/24 网段的 IP 地址，其他网段报文信息无法通过。

Task 2.d: Write to the TUN Interface

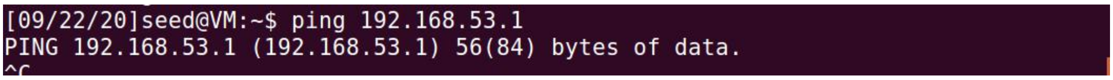
- 将 IP 数据包写入接口

添加发送虚假报文的代码：

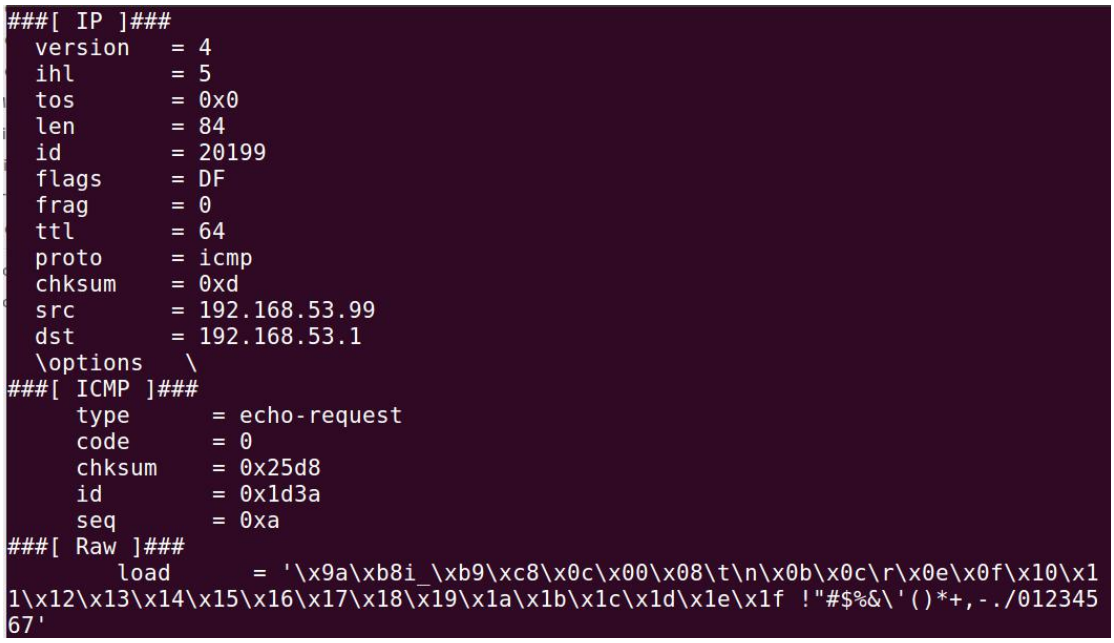


```
while True:
# Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if True:
    ip = IP(packet)
    ip.show()
    newip = IP(src='1.2.3.4', dst=ip.src)
    newpkt = newip/ip.payload
    os.write(tun, bytes(newpkt))
```

执行程序，程序打印出嗅探到的报文信息：

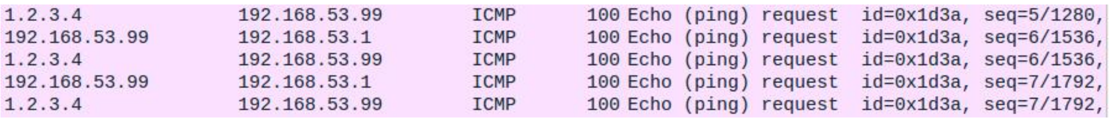


```
[09/22/20]seed@VM:~$ ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
```



```
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 20199
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0xd
  src       = 192.168.53.99
  dst       = 192.168.53.1
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x25d8
     id        = 0x1d3a
     seq       = 0xa
###[ Raw ]###
        load      = '\x9a\xb8i_\xb9\xc8\x0c\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x1
1\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./012345
67'
```

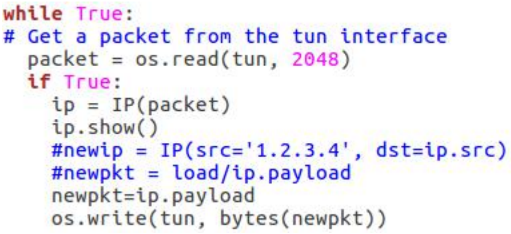
同时，通过查看到了实验构造的虚假报文：



```
1.2.3.4           192.168.53.99     ICMP     100 Echo (ping) request  id=0x1d3a, seq=5/1280,
192.168.53.99     192.168.53.1      ICMP     100 Echo (ping) request  id=0x1d3a, seq=6/1536,
1.2.3.4           192.168.53.99     ICMP     100 Echo (ping) request  id=0x1d3a, seq=6/1536,
192.168.53.99     192.168.53.1      ICMP     100 Echo (ping) request  id=0x1d3a, seq=7/1792,
1.2.3.4           192.168.53.99     ICMP     100 Echo (ping) request  id=0x1d3a, seq=7/1792,
```

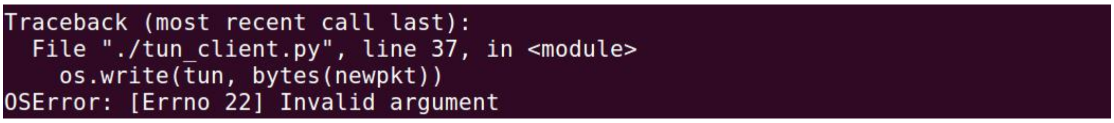
欺骗报文发送成功。

- 将任意数据写入接口



```
while True:
# Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if True:
    ip = IP(packet)
    ip.show()
    #newip = IP(src='1.2.3.4', dst=ip.src)
    #newpkt = load/ip.payload
    newpkt=ip.payload
    os.write(tun, bytes(newpkt))
```

发送任意数据出现格式错误：



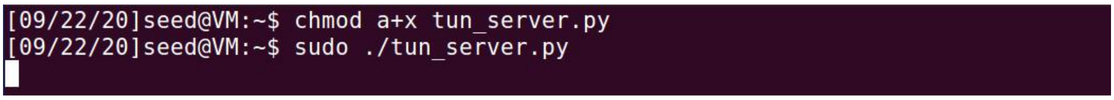
```
Traceback (most recent call last):
  File "./tun_client.py", line 37, in <module>
    os.write(tun, bytes(newpkt))
OSError: [Errno 22] Invalid argument
```

故只能发送 IP 数据包。

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

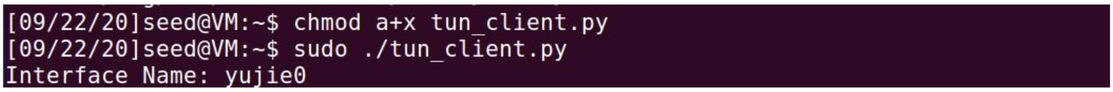
- 在隧道未打开的情况下

VPN 服务器监听不到报文：



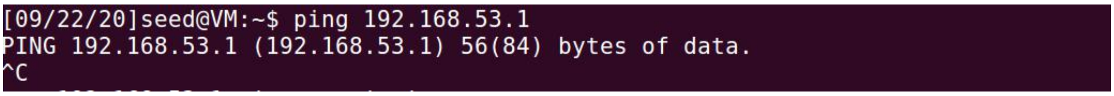
```
[09/22/20]seed@VM:~$ chmod a+x tun_server.py
[09/22/20]seed@VM:~$ sudo ./tun_server.py
█
```

- 打开隧道



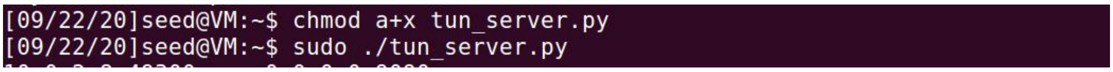
```
[09/22/20]seed@VM:~$ chmod a+x tun_client.py
[09/22/20]seed@VM:~$ sudo ./tun_client.py
Interface Name: yujie0
```

在主机 U 上 ping 192.168.53.0/24 网段地址：

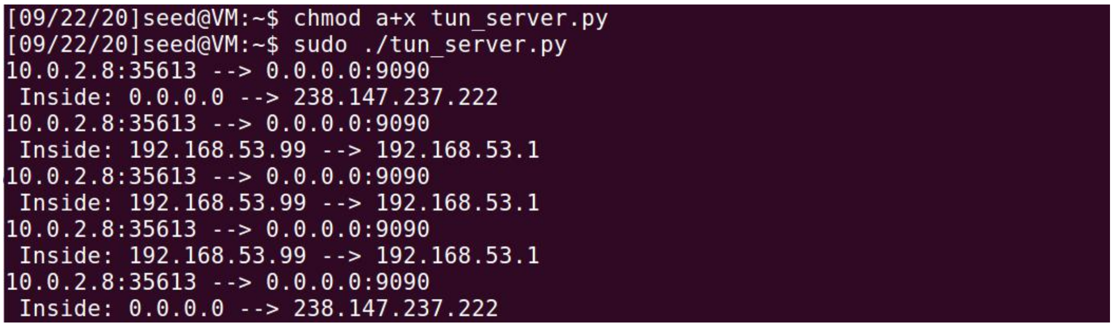


```
[09/22/20]seed@VM:~$ ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
```

VPN 服务器打印出监听到的 IP 数据包源地址和目的地址：



```
[09/22/20]seed@VM:~$ chmod a+x tun_server.py
[09/22/20]seed@VM:~$ sudo ./tun_server.py
```

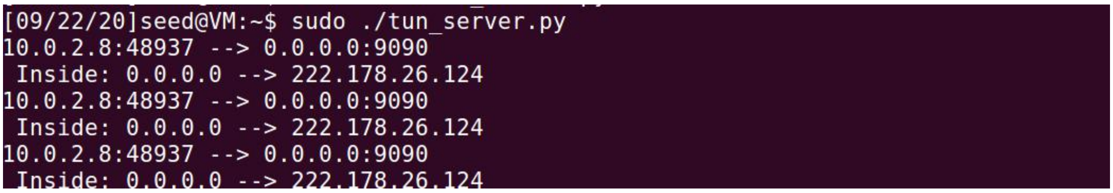


```
[09/22/20]seed@VM:~$ chmod a+x tun_server.py
[09/22/20]seed@VM:~$ sudo ./tun_server.py
10.0.2.8:35613 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 238.147.237.222
10.0.2.8:35613 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.8:35613 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.8:35613 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.8:35613 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 238.147.237.222
```

由于 server 程序只监听 9090 端口，因此只有发送到 9090 端口的数据包可以被监听到并打印出来。

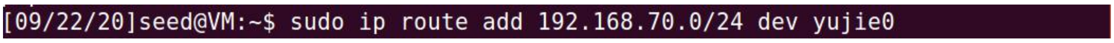
- 在主机 U 上 ping 主机 V

一开始无法接通，且没有 ICMP 报文发送：



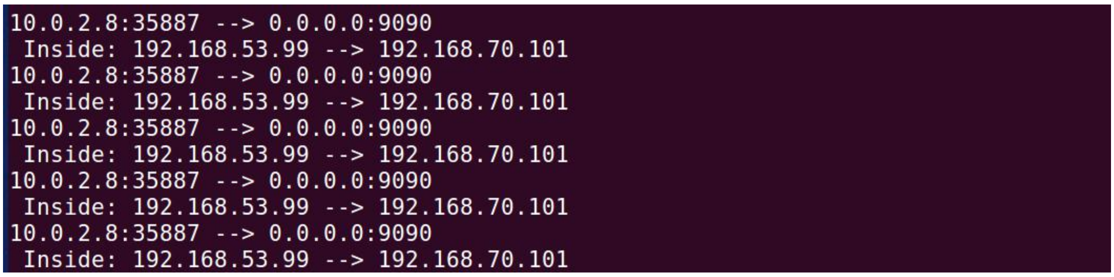
```
[09/22/20]seed@VM:~$ sudo ./tun_server.py
10.0.2.8:48937 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 222.178.26.124
10.0.2.8:48937 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 222.178.26.124
10.0.2.8:48937 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 222.178.26.124
```

然后配置路由信息：



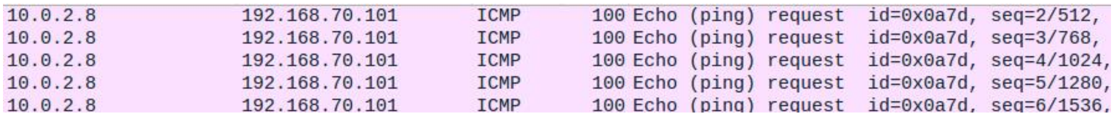
```
[09/22/20]seed@VM:~$ sudo ip route add 192.168.70.0/24 dev yujie0
```

此时服务器程序接收到 ICMP 报文信息：



```
10.0.2.8:35887 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.70.101
10.0.2.8:35887 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.70.101
10.0.2.8:35887 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.70.101
10.0.2.8:35887 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.70.101
10.0.2.8:35887 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.70.101
```

同时在客户端可以监听到发送的 ICMP 报文：

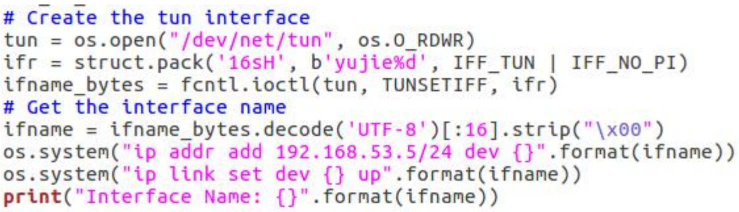


```
10.0.2.8            192.168.70.101      ICMP      100 Echo (ping) request  id=0x0a7d, seq=2/512,
10.0.2.8            192.168.70.101      ICMP      100 Echo (ping) request  id=0x0a7d, seq=3/768,
10.0.2.8            192.168.70.101      ICMP      100 Echo (ping) request  id=0x0a7d, seq=4/1024,
10.0.2.8            192.168.70.101      ICMP      100 Echo (ping) request  id=0x0a7d, seq=5/1280,
10.0.2.8            192.168.70.101      ICMP      100 Echo (ping) request  id=0x0a7d, seq=6/1536,
```

ICMP 报文已通过隧道发送给 VPN 服务器。

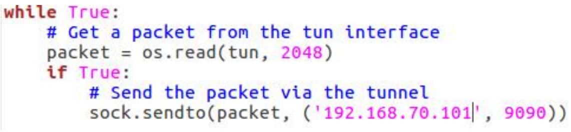
Task 4: Set Up the VPN Server

在 VPN 服务器上配置隧道接口：



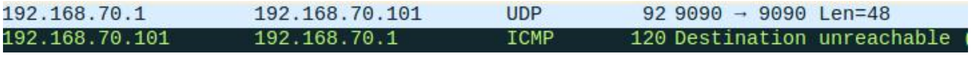
```python
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'yujie%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.5/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
```

将收到的 ICMP 报文进行转发：



```python
while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        # Send the packet via the tunnel
        sock.sendto(packet, ('192.168.70.101', 9090))
```

通过主机 V 的 wireshark 查看：

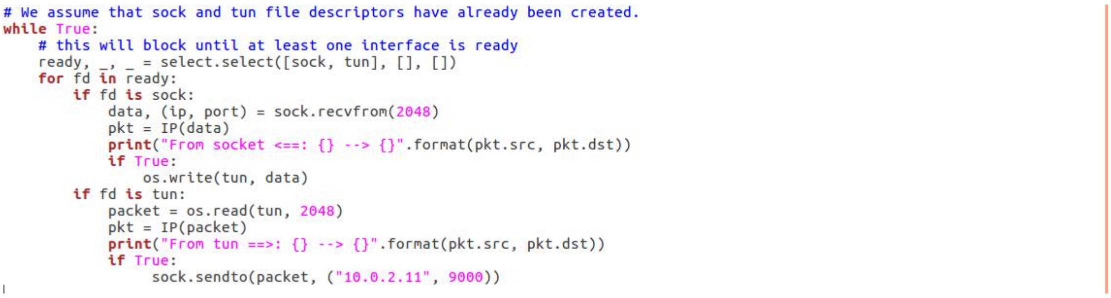


```
192.168.70.1        192.168.70.101      UDP        92 9090 → 9090 Len=48
192.168.70.101      192.168.70.1        ICMP      120 Destination unreachable (
```

主机 V 试图回复主机 U，这说明主机 V 已经收到主机 U 的 ICMP 报文，单向隧道配置成功。但由于反向通道还未进行配置，故 ICMP 响应不可达。

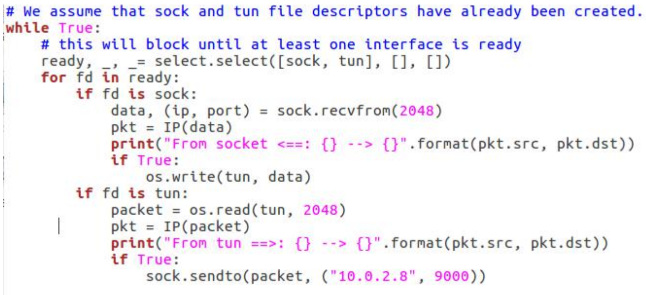
Task 5: Handling Traffific in Both Directions

配置主机 U：



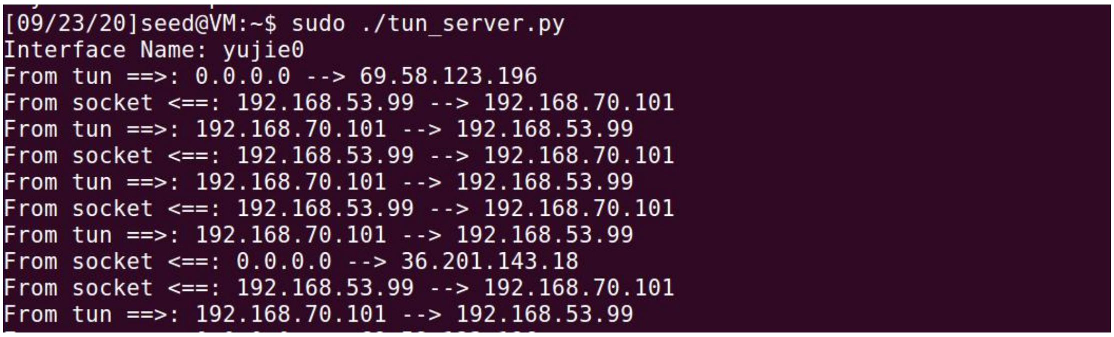
```python
# We assume that sock and tun file descriptors have already been created.
while True:
    # this will block until at least one interface is ready
    ready, _, _ = select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            if True:
                os.write(tun, data)
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            if True:
                 sock.sendto(packet, ("10.0.2.11", 9000))
```

配置 VPN 服务器：



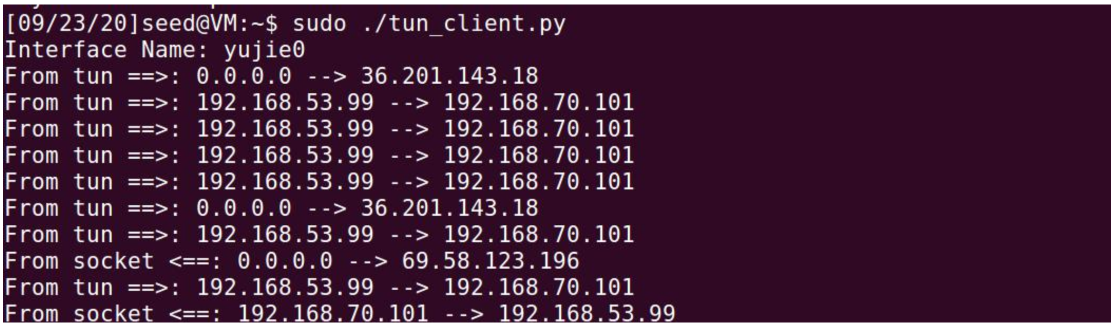
```python
# We assume that sock and tun file descriptors have already been created.
while True:
    # this will block until at least one interface is ready
    ready, _, _= select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            if True:
                os.write(tun, data)
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            if True:
                sock.sendto(packet, ("10.0.2.8", 9000))
```

运行服务器程序：



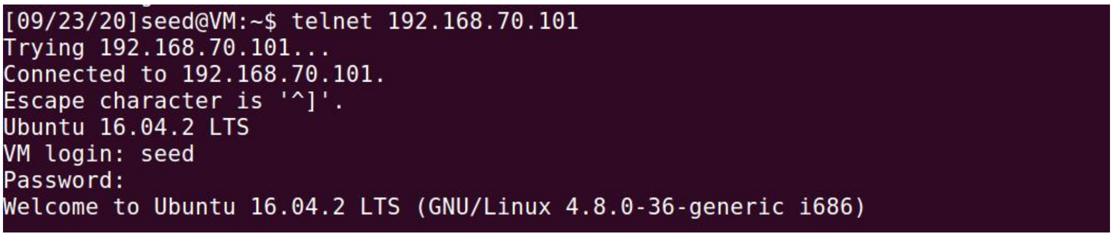
```
[09/23/20]seed@VM:~$ sudo ./tun_server.py
Interface Name: yujie0
From tun ==>: 0.0.0.0 --> 69.58.123.196
From socket <==: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.53.99
From socket <==: 0.0.0.0 --> 36.201.143.18
From socket <==: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.53.99
```

运行客户端程序：



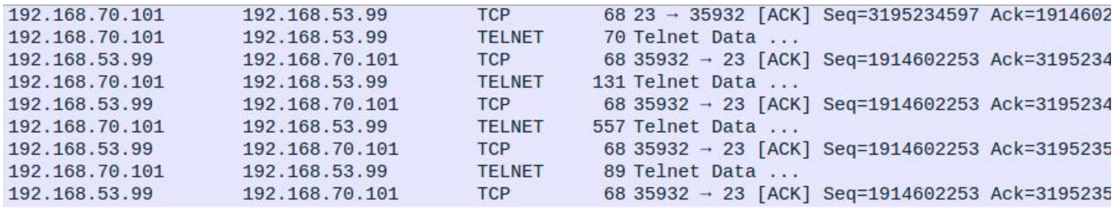
```
[09/23/20]seed@VM:~$ sudo ./tun_client.py
Interface Name: yujie0
From tun ==>: 0.0.0.0 --> 36.201.143.18
From tun ==>: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.53.99 --> 192.168.70.101
From tun ==>: 192.168.53.99 --> 192.168.70.101
From tun ==>: 0.0.0.0 --> 36.201.143.18
From tun ==>: 192.168.53.99 --> 192.168.70.101
From socket <==: 0.0.0.0 --> 69.58.123.196
From tun ==>: 192.168.53.99 --> 192.168.70.101
From socket <==: 192.168.70.101 --> 192.168.53.99
```

- 主机 U 与主机 V 建立 Telnet 连接：



```
[09/23/20]seed@VM:~$ telnet 192.168.70.101
Trying 192.168.70.101...
Connected to 192.168.70.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

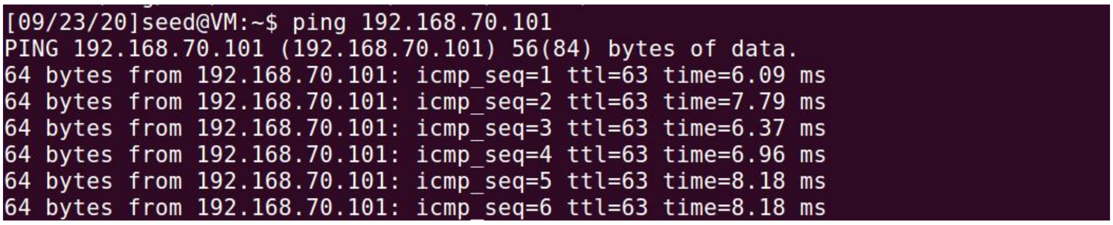
通过主机 V 的 wireshark 观察到主机 U 和主机 V 的 TCP 和 TELNET 报文：



```
192.168.70.101       192.168.53.99        TCP        68 23 → 35932 [ACK] Seq=3195234597 Ack=1914602
192.168.70.101       192.168.53.99        TELNET     70 Telnet Data ...
192.168.53.99        192.168.70.101       TCP        68 35932 → 23 [ACK] Seq=1914602253 Ack=3195234
192.168.70.101       192.168.53.99        TELNET    131 Telnet Data ...
192.168.53.99        192.168.70.101       TCP        68 35932 → 23 [ACK] Seq=1914602253 Ack=3195234
192.168.70.101       192.168.53.99        TELNET    557 Telnet Data ...
192.168.53.99        192.168.70.101       TCP        68 35932 → 23 [ACK] Seq=1914602253 Ack=3195235
192.168.70.101       192.168.53.99        TELNET     89 Telnet Data ...
192.168.53.99        192.168.70.101       TCP        68 35932 → 23 [ACK] Seq=1914602253 Ack=3195235
```

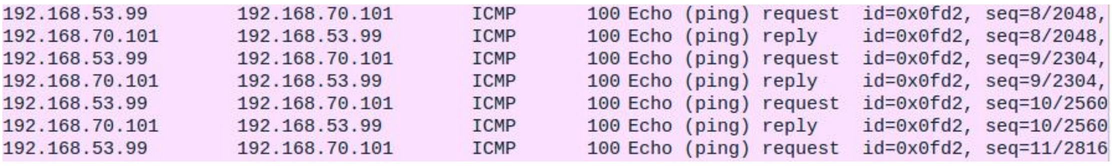
Telnet 连接成功。

- 主机 U ping 主机 V，可以 ping 通：



```
[09/23/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
64 bytes from 192.168.70.101: icmp_seq=1 ttl=63 time=6.09 ms
64 bytes from 192.168.70.101: icmp_seq=2 ttl=63 time=7.79 ms
64 bytes from 192.168.70.101: icmp_seq=3 ttl=63 time=6.37 ms
64 bytes from 192.168.70.101: icmp_seq=4 ttl=63 time=6.96 ms
64 bytes from 192.168.70.101: icmp_seq=5 ttl=63 time=8.18 ms
64 bytes from 192.168.70.101: icmp_seq=6 ttl=63 time=8.18 ms
```

通过主机 V 的 wireshark 观察到主机 V 发送的 ICMP 响应报文：



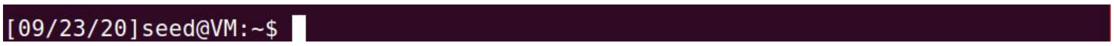
```
192.168.53.99        192.168.70.101       ICMP      100 Echo (ping) request  id=0x0fd2, seq=8/2048,
192.168.70.101       192.168.53.99        ICMP      100 Echo (ping) reply    id=0x0fd2, seq=8/2048,
192.168.53.99        192.168.70.101       ICMP      100 Echo (ping) request  id=0x0fd2, seq=9/2304,
192.168.70.101       192.168.53.99        ICMP      100 Echo (ping) reply    id=0x0fd2, seq=9/2304,
192.168.53.99        192.168.70.101       ICMP      100 Echo (ping) request  id=0x0fd2, seq=10/2560
192.168.70.101       192.168.53.99        ICMP      100 Echo (ping) reply    id=0x0fd2, seq=10/2560
192.168.53.99        192.168.70.101       ICMP      100 Echo (ping) request  id=0x0fd2, seq=11/2816
```

ping 连接过程中的报文流向：

①主机 U 构造目的地址为主机 V 的 ICMP 报文，交给路由器；
②路由器将报文交给 tun 接口，建立 socket 套接字，构造 IP 报文交给 VPN 服务器；
③VPN 服务器收到 IP 报文，交付 9000 端口进程；
④进程根据目的地址交付主机 V；
⑤主机 V 收到 ICMP 报文，构造 ICMP 响应报文进行回复，目的地址为主机 U；
⑥路由器将报文交给 tun 接口，建立 socket 套接字，构造 IP 报文交给 VPN 服务器；
⑦VPN 服务器收到 IP 报文，交付 9000 端口进程；
⑧主机 U 收到 ICMP 响应报文，完成 ping 连接。
至此，VPN 隧道建立完成，主机 U 和主机 V 可以相互通信。

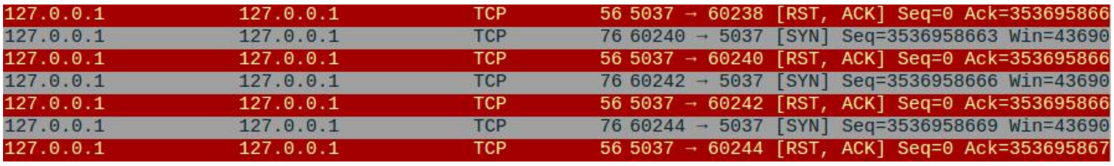
Task 6: Tunnel-Breaking Experiment

主机 U 与主机 V 建立 Telnet 连接时，断开 VPN 隧道，在终端输入的字符无法显示：



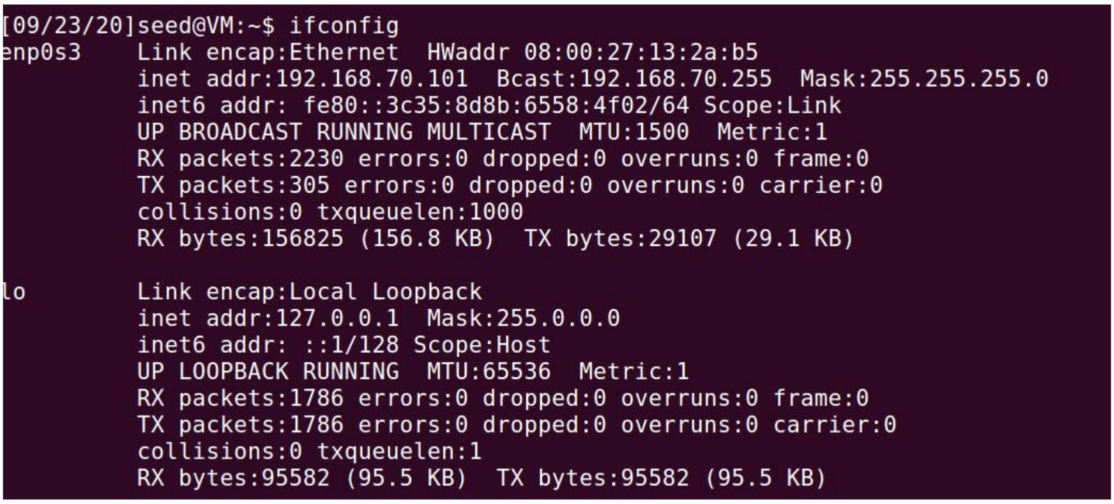
```
[09/23/20]seed@VM:~$ 
```

查看此时客户机的 wireshark：



```
127.0.0.1            127.0.0.1            TCP        56 5037 → 60238 [RST, ACK] Seq=0 Ack=353695866
127.0.0.1            127.0.0.1            TCP        76 60240 → 5037 [SYN] Seq=3536958663 Win=43690
127.0.0.1            127.0.0.1            TCP        56 5037 → 60240 [RST, ACK] Seq=0 Ack=353695866
127.0.0.1            127.0.0.1            TCP        76 60242 → 5037 [SYN] Seq=3536958666 Win=43690
127.0.0.1            127.0.0.1            TCP        56 5037 → 60242 [RST, ACK] Seq=0 Ack=353695866
127.0.0.1            127.0.0.1            TCP        76 60244 → 5037 [SYN] Seq=3536958669 Win=43690
127.0.0.1            127.0.0.1            TCP        56 5037 → 60244 [RST, ACK] Seq=0 Ack=353695867
```

重新建立 VPN 隧道，刚刚输入的字符重新显示：

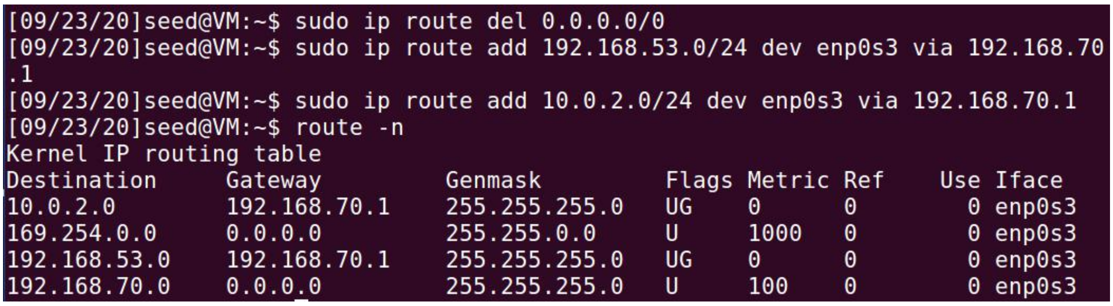


```
[09/23/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:13:2a:b5
          inet addr:192.168.70.101  Bcast:192.168.70.255  Mask:255.255.255.0
          inet6 addr: fe80::3c35:8d8b:6558:4f02/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2230 errors:0 dropped:0 overruns:0 frame:0
          TX packets:305 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:156825 (156.8 KB)  TX bytes:29107 (29.1 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:1786 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1786 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:95582 (95.5 KB)  TX bytes:95582 (95.5 KB)
```

可以看到，随着 VPN 隧道的中断，telnet 连接没有中断。TCP 将继续重新发送数据包，但由于隧道被破坏，它们无法被发送。我们在 telnet 中输入的任何内容都会被 TCP 缓冲，不会丢失，但我们看不到任何内容。一旦我们重新连接隧道，我们输入的所有信息都会显示出来。

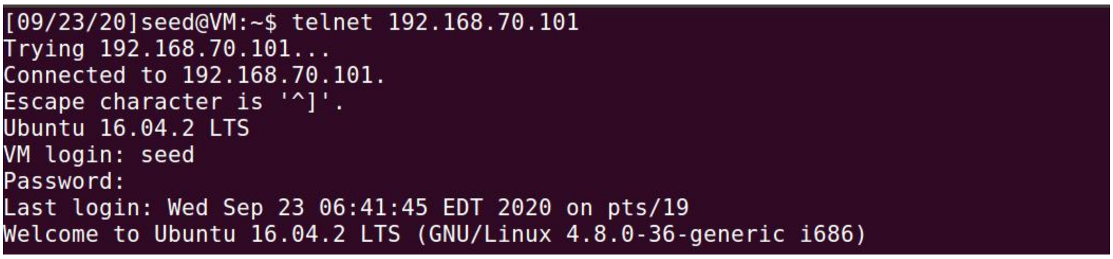
Task 7: Routing Experiment on Host V

配置主机 V 的路由器：



```
[09/23/20]seed@VM:~$ sudo ip route del 0.0.0.0/0
[09/23/20]seed@VM:~$ sudo ip route add 192.168.53.0/24 dev enp0s3 via 192.168.70
.1
[09/23/20]seed@VM:~$ sudo ip route add 10.0.2.0/24 dev enp0s3 via 192.168.70.1
[09/23/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.2.0        192.168.70.1    255.255.255.0   UG    0      0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.53.0    192.168.70.1    255.255.255.0   UG    0      0        0 enp0s3
192.168.70.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
```

连通 VPN 隧道后，主机 U 与主机 V 建立 Telnet 连接：

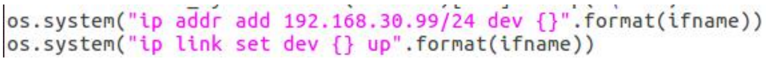


```
[09/23/20]seed@VM:~$ telnet 192.168.70.101
Trying 192.168.70.101...
Connected to 192.168.70.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Sep 23 06:41:45 EDT 2020 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

可以正常通信。

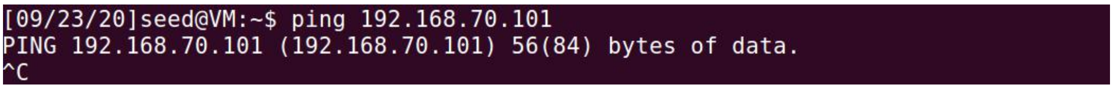
Task 8: Experiment with the TUN IP Address

修改隧道接口地址：



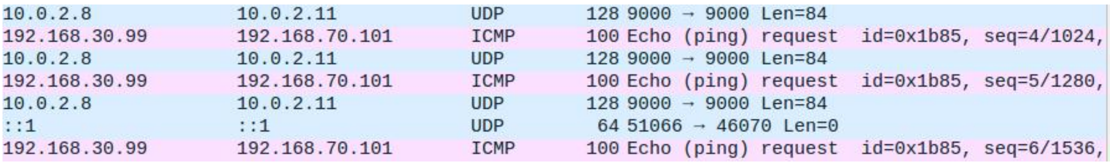
```
os.system("ip addr add 192.168.30.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
```

此时主机 U 无法 ping 通主机 V：



```
[09/23/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
^C
```

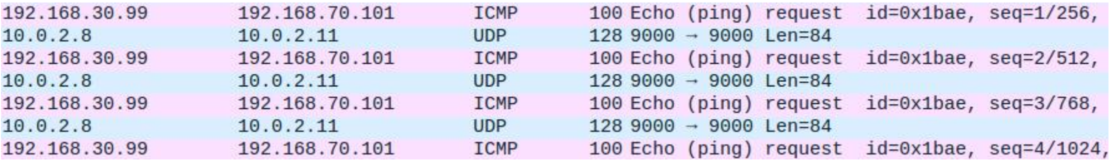
主机 V 的 Wireshark 显示有 ICMP 报文发出，然而主机 U 并没有接收到，说明数据包被丢弃：



```
10.0.2.8         10.0.2.11        UDP      128 9000 → 9000 Len=84
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1b85, seq=4/1024,
10.0.2.8         10.0.2.11        UDP      128 9000 → 9000 Len=84
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1b85, seq=5/1280,
10.0.2.8         10.0.2.11        UDP      128 9000 → 9000 Len=84
::1              ::1              UDP       64 51066 → 46070 Len=0
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1b85, seq=6/1536,
```

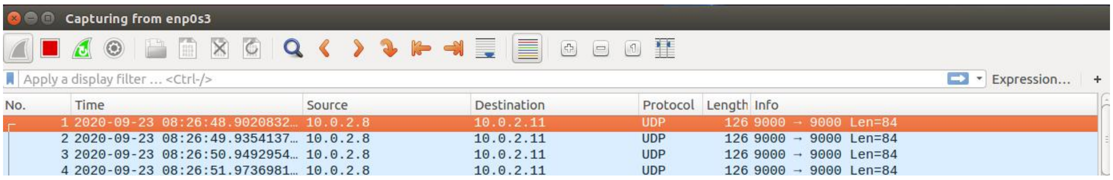
- Where are the packets dropped?

查看 VPN 服务器的 Wireshark，tun 端口（yujie0）有 ICMP 报文和 UDP 报文：



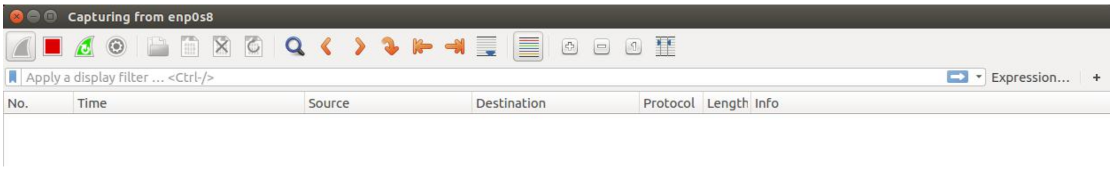
```
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1bae, seq=1/256,
10.0.2.8         10.0.2.11        UDP      128 9000 → 9000 Len=84
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1bae, seq=2/512,
10.0.2.8         10.0.2.11        UDP      128 9000 → 9000 Len=84
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1bae, seq=3/768,
10.0.2.8         10.0.2.11        UDP      128 9000 → 9000 Len=84
192.168.30.99    192.168.70.101   ICMP     100 Echo (ping) request  id=0x1bae, seq=4/1024,
```

enp0s3 端口也有报文通过：



```
No.   Time                          Source      Destination  Protocol  Length Info
    1 2020-09-23 08:26:48.9020832… 10.0.2.8    10.0.2.11    UDP          126 9000 → 9000 Len=84
    2 2020-09-23 08:26:49.9354137… 10.0.2.8    10.0.2.11    UDP          126 9000 → 9000 Len=84
    3 2020-09-23 08:26:50.9492954… 10.0.2.8    10.0.2.11    UDP          126 9000 → 9000 Len=84
    4 2020-09-23 08:26:51.9736981… 10.0.2.8    10.0.2.11    UDP          126 9000 → 9000 Len=84
```

而 enp0s8 端口没有报文信息：

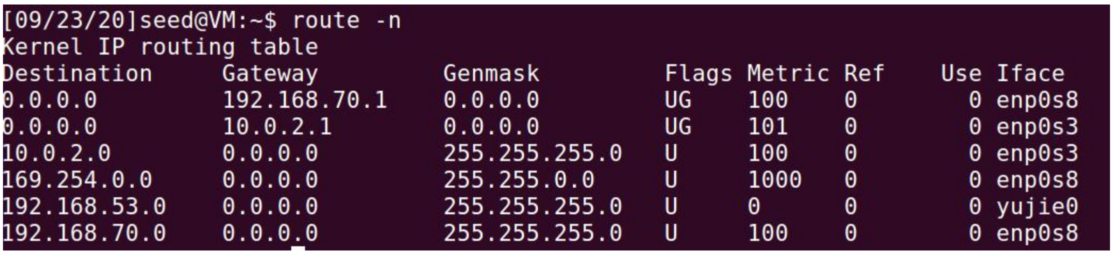


这说明服务器没有将报文从 enp0s8 端口转发到主机 V，报文在服务器被丢弃。

- Why are the packets dropped?

因为 Linux 内核具有反向路由检查机制，基本原理是根据包的源地址查找路由的出接口，然后比较包的原始入接口是否和查到的出接口一致，如果一致则放过，如果不一致则丢弃。

查看 VPN 服务器的路由表：

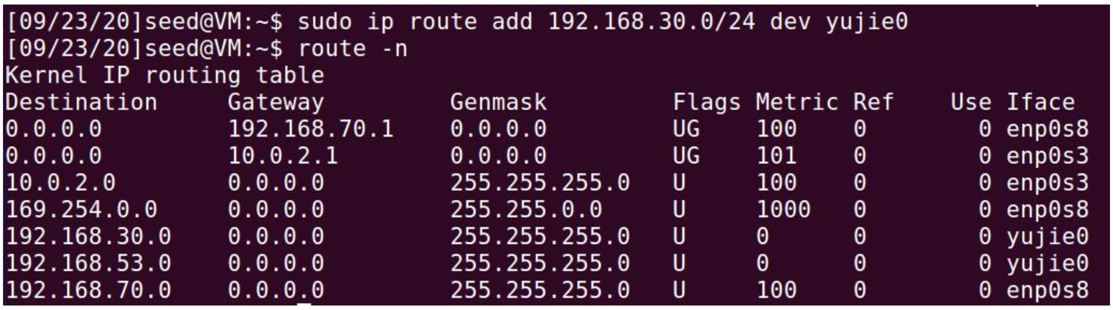


```
[09/23/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.70.1    0.0.0.0         UG    100    0        0 enp0s8
0.0.0.0         10.0.2.1        0.0.0.0         UG    101    0        0 enp0s3
10.0.2.0        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 yujie0
192.168.70.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s8
```

主机 U 上修改后的隧道端口 IP 地址为 192.168.30.99，来源于 192.168.30.0/24 网络，经 VPN 服务器的路由查找，其应通过默认路由项 0.0.0.0/0 转发，即通过 enp0s3 端口，但实际中服务器是由 tun 端口（yujie0）接收到 ICMP 报文，出接口不一致，则服务器丢弃。

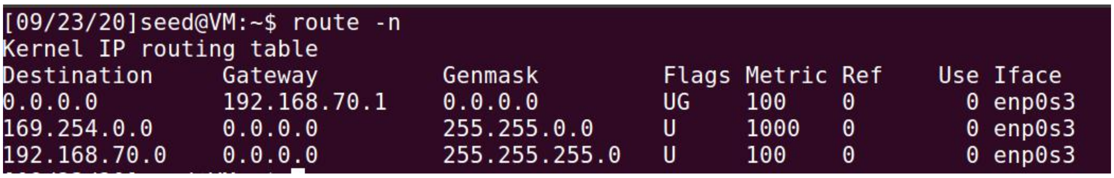
- How to solve this problem?

为 VPN 服务器配置和 192.168.30.0/24 网络一致的 tun 端口（yujie0）：



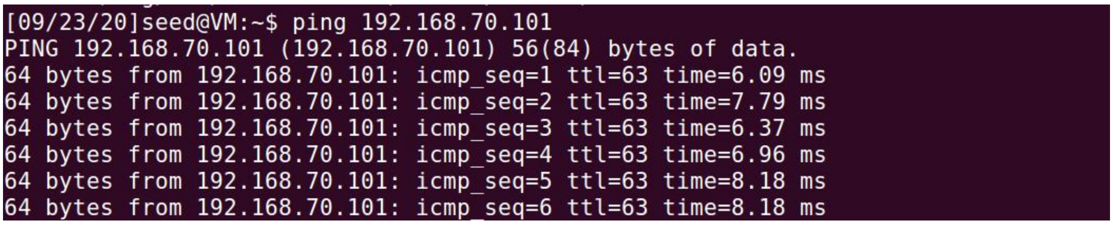
```
[09/23/20]seed@VM:~$ sudo ip route add 192.168.30.0/24 dev yujie0
[09/23/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.70.1    0.0.0.0         UG    100    0        0 enp0s8
0.0.0.0         10.0.2.1        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
192.168.30.0    0.0.0.0         255.255.255.0   U     0      0        0 yujie0
192.168.53.0    0.0.0.0         255.255.255.0   U     0      0        0 yujie0
192.168.70.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s8
```

为主机 V 配置默认端口：



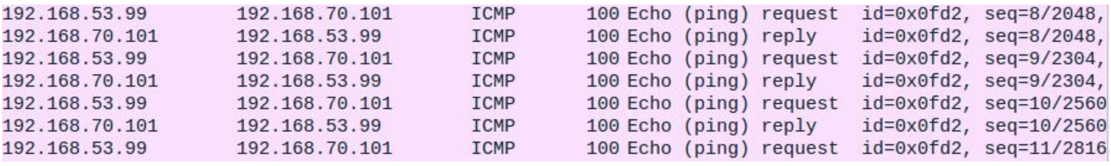
```
[09/23/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.70.1    0.0.0.0         UG    100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.70.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
```

此时在主机 U 上 ping 主机 V，可以连通：



```
[09/23/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
64 bytes from 192.168.70.101: icmp_seq=1 ttl=63 time=6.09 ms
64 bytes from 192.168.70.101: icmp_seq=2 ttl=63 time=7.79 ms
64 bytes from 192.168.70.101: icmp_seq=3 ttl=63 time=6.37 ms
64 bytes from 192.168.70.101: icmp_seq=4 ttl=63 time=6.96 ms
64 bytes from 192.168.70.101: icmp_seq=5 ttl=63 time=8.18 ms
64 bytes from 192.168.70.101: icmp_seq=6 ttl=63 time=8.18 ms
```

通过主机 V 的 wireshark 观察到 ICMP 报文：

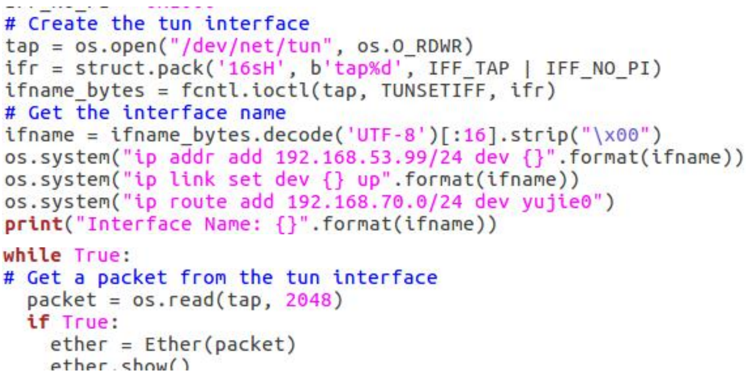


```
192.168.53.99     192.168.70.101    ICMP     100 Echo (ping) request  id=0x0fd2, seq=8/2048,
192.168.70.101    192.168.53.99     ICMP     100 Echo (ping) reply    id=0x0fd2, seq=8/2048,
192.168.53.99     192.168.70.101    ICMP     100 Echo (ping) request  id=0x0fd2, seq=9/2304,
192.168.70.101    192.168.53.99     ICMP     100 Echo (ping) reply    id=0x0fd2, seq=9/2304,
192.168.53.99     192.168.70.101    ICMP     100 Echo (ping) request  id=0x0fd2, seq=10/2560
192.168.70.101    192.168.53.99     ICMP     100 Echo (ping) reply    id=0x0fd2, seq=10/2560
192.168.53.99     192.168.70.101    ICMP     100 Echo (ping) request  id=0x0fd2, seq=11/2816
```

主机 U 和主机 V 实现通信。

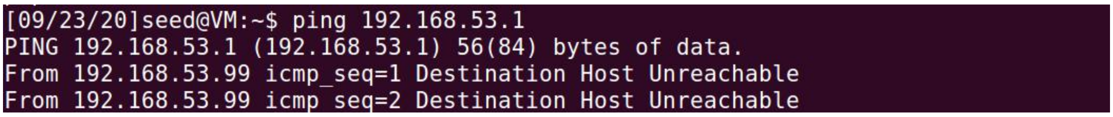
Task 9: Experiment with the TAP Interface

创建并配置 tap 端口：



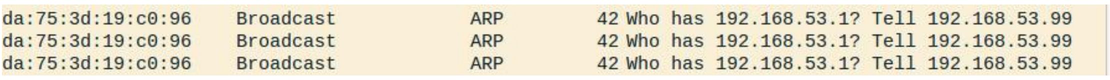
```
# Create the tun interface
tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tap%d', IFF_TAP | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tap, TUNSETIFF, ifr)
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.70.0/24 dev yujie0")
print("Interface Name: {}".format(ifname))

while True:
# Get a packet from the tun interface
  packet = os.read(tap, 2048)
  if True:
    ether = Ether(packet)
    ether.show()
```

在客户机上 ping 192.168.53.0/24 网络，主机不可达：



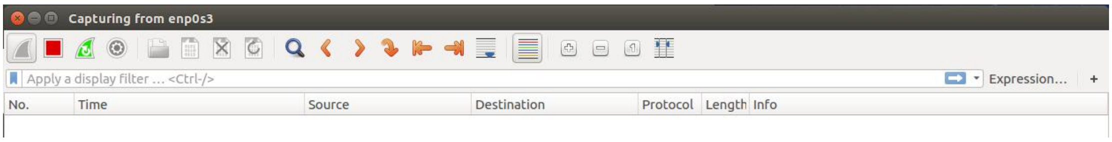
```
[09/23/20]seed@VM:~$ ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
From 192.168.53.99 icmp_seq=1 Destination Host Unreachable
From 192.168.53.99 icmp_seq=2 Destination Host Unreachable
```

通过 wireshark 查看 tap0 端口发送 ARP 报文：



```
da:75:3d:19:c0:96    Broadcast         ARP      42 Who has 192.168.53.1? Tell 192.168.53.99
da:75:3d:19:c0:96    Broadcast         ARP      42 Who has 192.168.53.1? Tell 192.168.53.99
da:75:3d:19:c0:96    Broadcast         ARP      42 Who has 192.168.53.1? Tell 192.168.53.99
```

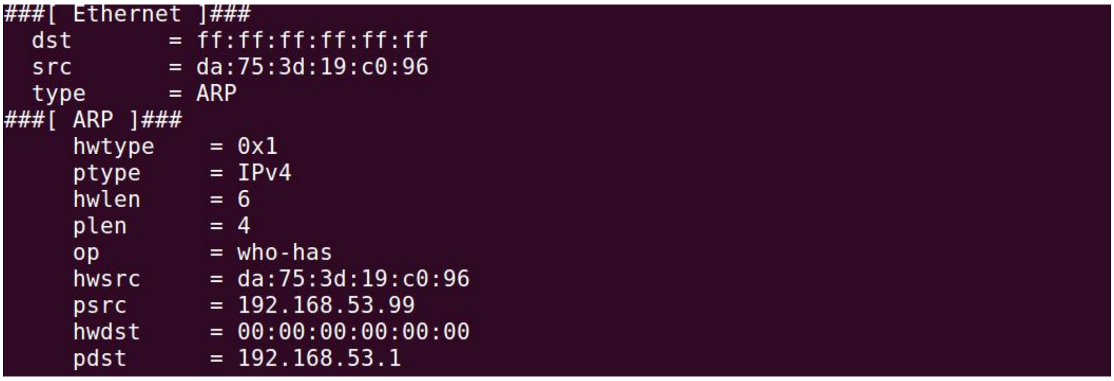
通过 wireshark 查看 enp0s3 端口无报文发出：



No.	Time	Source	Destination	Protocol	Length	Info

因此，ARP 报文全部通过 tap0 端口发出。

查看 tap0 端口打印出的 ARP 报文：



```
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = da:75:3d:19:c0:96
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = da:75:3d:19:c0:96
     psrc      = 192.168.53.99
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.53.1
```

可以看到，ARP 请求直接通过广播形式查找 192.168.53.1 的 MAC 地址，而该地址为虚拟地址，无主机响应，因此无法收到报文回复。因此，TAP 对 MAC 层的处理需要更复杂的通信。