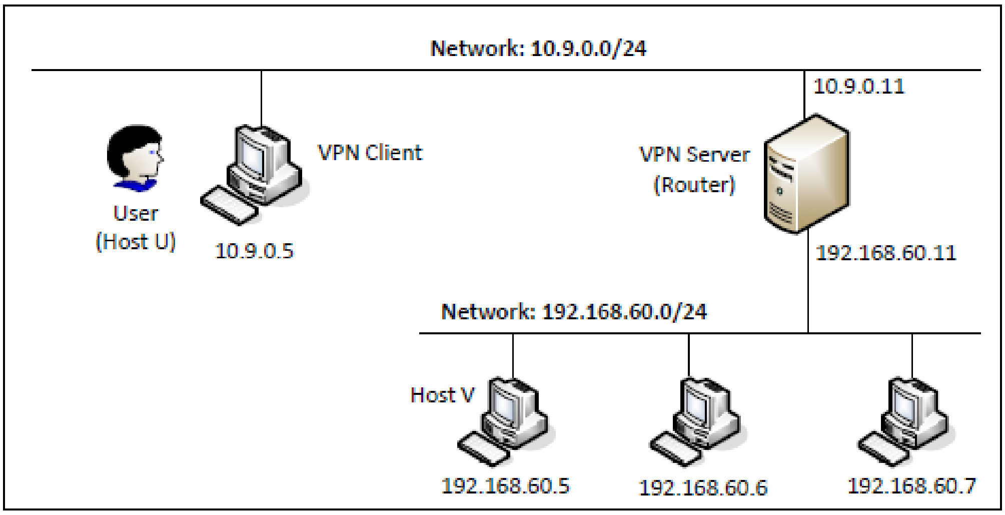
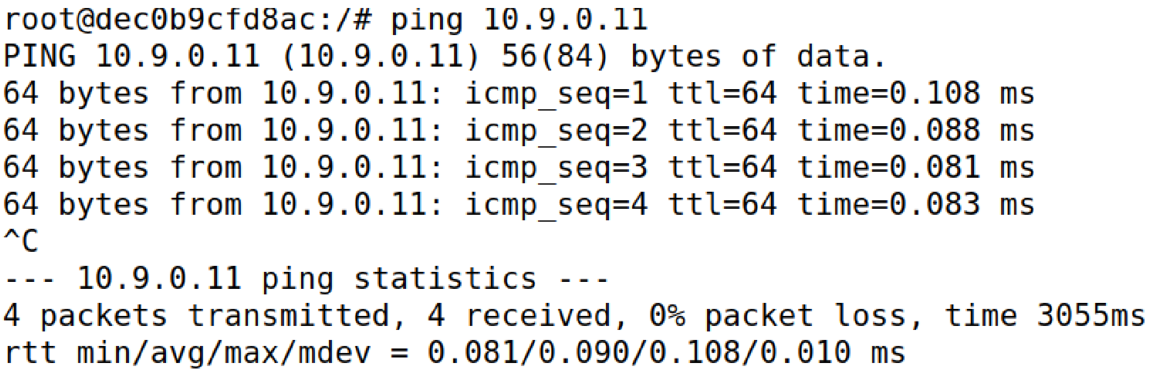
lab7-report

57118115 陈烨



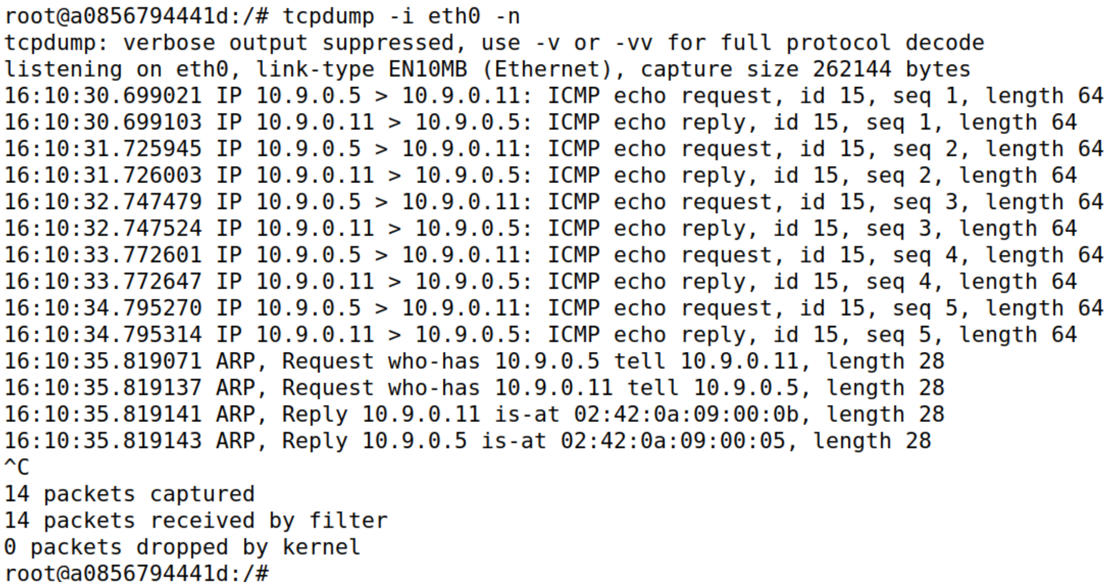
Task1

在主机U上ping服务器，得到结果如下，可知能够连接:



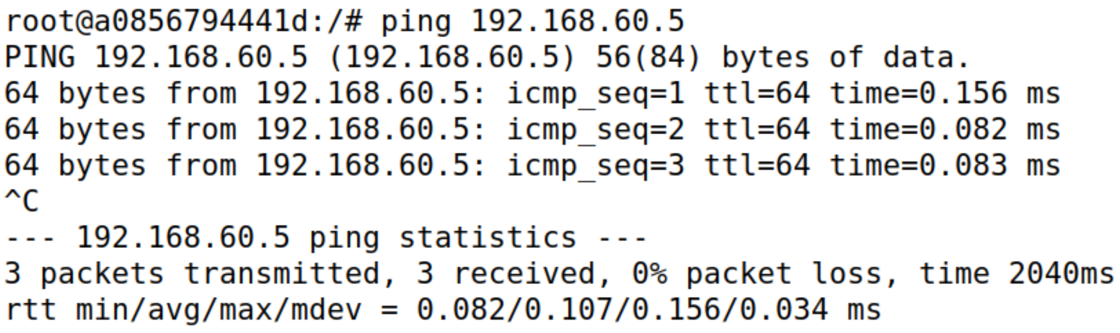
```
root@dec0b9cfd8ac:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.108 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.081 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.083 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.081/0.090/0.108/0.010 ms
```

在VPN服务器上利用tcpdump命令抓取数据包，得到结果如下:



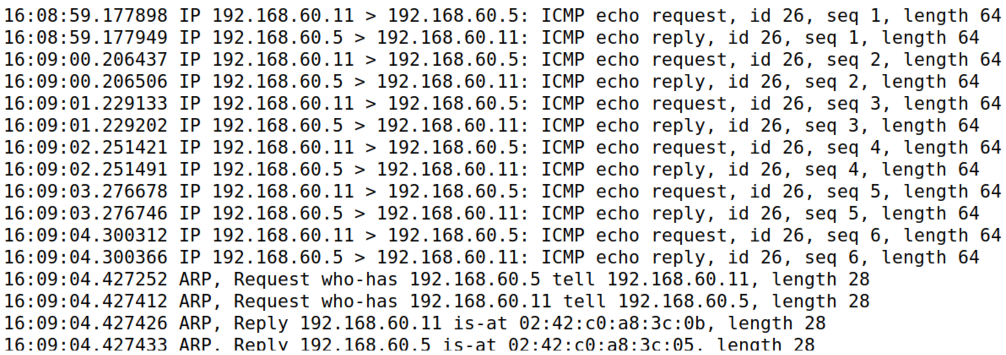
```
root@a0856794441d:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:10:30.699021 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 1, length 64
16:10:30.699103 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 1, length 64
16:10:31.725945 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 2, length 64
16:10:31.726003 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 2, length 64
16:10:32.747479 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 3, length 64
16:10:32.747524 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 3, length 64
16:10:33.772601 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 4, length 64
16:10:33.772647 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 4, length 64
16:10:34.795270 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 5, length 64
16:10:34.795314 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 5, length 64
16:10:35.819071 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
16:10:35.819137 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
16:10:35.819141 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
16:10:35.819143 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@a0856794441d:/#
```

在VPN服务器上ping主机V，得到结果如下，可知能够连接:



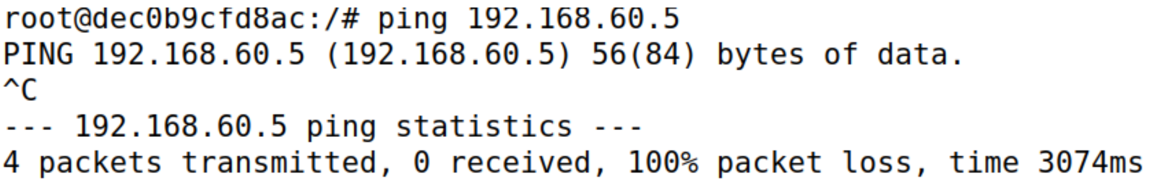
```
root@a0856794441d:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.156 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.083 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.082/0.107/0.156/0.034 ms
```

在VPN服务器上利用tcpdump命令抓取数据包，得到结果如下:



```
16:08:59.177898 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 26, seq 1, length 64
16:08:59.177949 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 26, seq 1, length 64
16:09:00.206437 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 26, seq 2, length 64
16:09:00.206506 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 26, seq 2, length 64
16:09:01.229133 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 26, seq 3, length 64
16:09:01.229202 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 26, seq 3, length 64
16:09:02.251421 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 26, seq 4, length 64
16:09:02.251491 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 26, seq 4, length 64
16:09:03.276678 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 26, seq 5, length 64
16:09:03.276746 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 26, seq 5, length 64
16:09:04.300312 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 26, seq 6, length 64
16:09:04.300366 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 26, seq 6, length 64
16:09:04.427252 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
16:09:04.427412 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
16:09:04.427426 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
16:09:04.427433 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
```

在主机U上ping主机V，得到结果如下，可知无法连接:



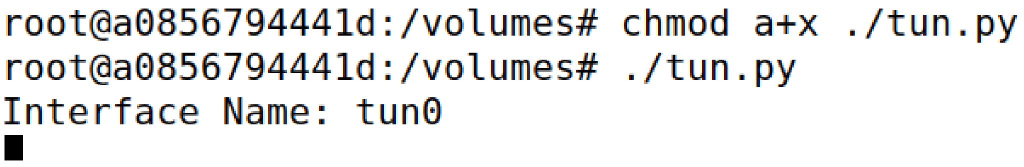
```
root@dec0b9cfd8ac:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3074ms
```

Task2

熟悉TUN/TAP技术

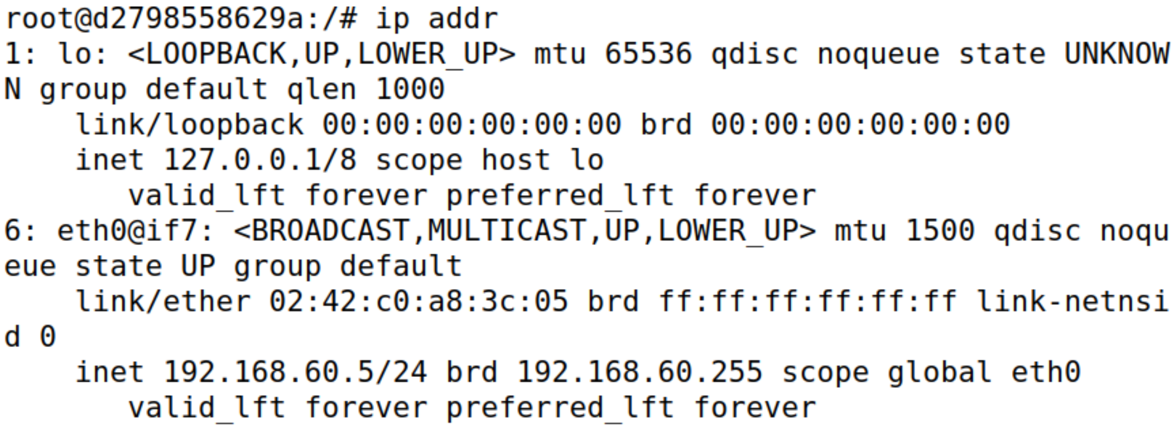
Task 2a

vpn server运行代码，创建通道:



```
root@a0856794441d:/volumes# chmod a+x ./tun.py
root@a0856794441d:/volumes# ./tun.py
Interface Name:  tun0
█
```

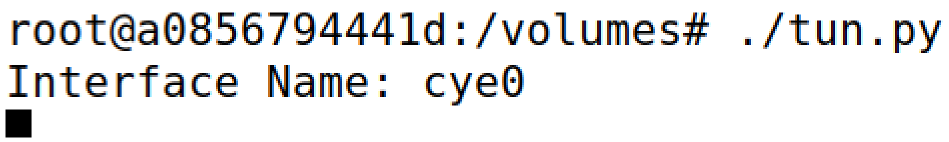
保持程序运行，查看发现有通道



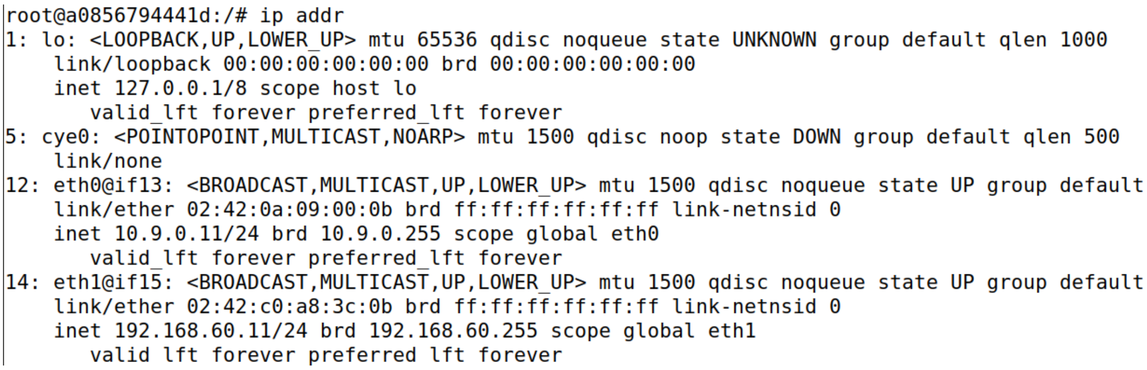
```
root@d2798558629a:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqu
eue state UP group default
    link/ether 02:42:c0:a8:3c:05 brd ff:ff:ff:ff:ff:ff link-netnsi
d 0
    inet 192.168.60.5/24 brd 192.168.60.255 scope global eth0
       valid_lft forever preferred_lft forever
```

```
#!/usr/bin/env 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'cye%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")
print("Interface Name: {}".format(ifname))

while True:
    time.sleep(10)

```

改写代码换成自己名字，再次运行：



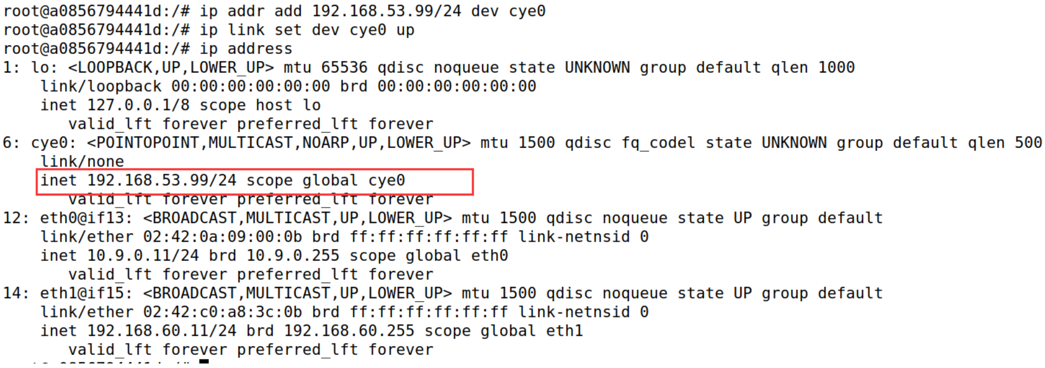
```
root@a0856794441d:/volumes# ./tun.py
Interface Name: cye0
```



```
root@a0856794441d:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
5: cye0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid lft forever preferred lft forever
```

Task2b

TUN接口是不可用的，因为它还没有配置,需要给它分配一个IP地址,启动接口，因为接口仍然处于down状态。



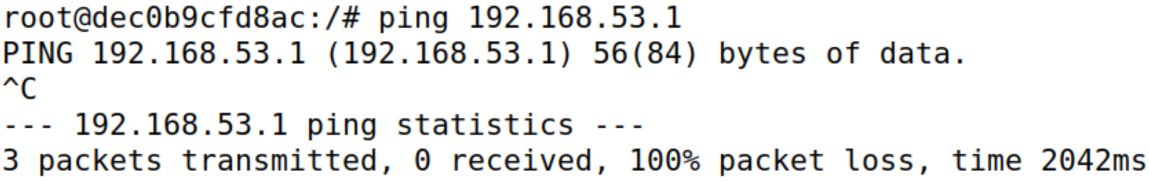
```
root@a0856794441d:/# ip addr add 192.168.53.99/24 dev cye0
root@a0856794441d:/# ip link set dev cye0 up
root@a0856794441d:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
6: cye0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global cye0
       valid_lft forever preferred_lft forever
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever
```

Task2c

host U 上执行代码

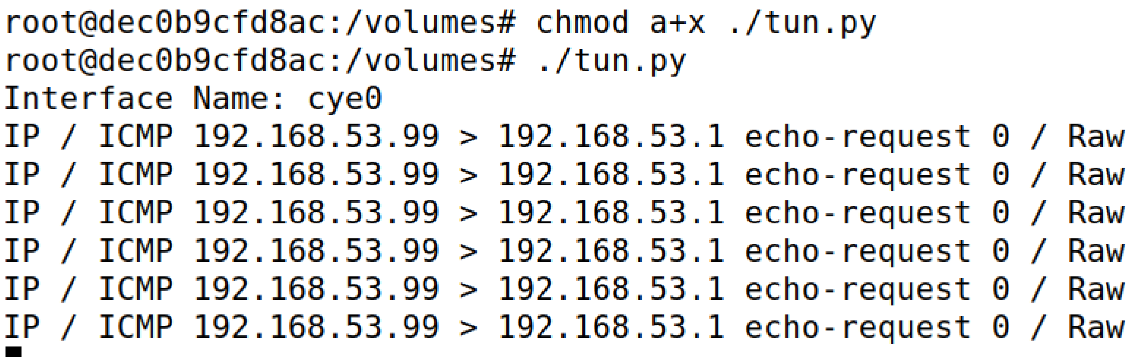
```python
#!/usr/bin/env python3
import fcntl
import struct
import os
import time
from scapy.all import *
TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'cye%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
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))
print("Interface Name: {}".format(ifname))
while True:
    packet = os.read(tun, 2048)
    if packet:
        ip = IP(packet)
        print(ip.summary())
```

在主机U上ping主机192.168.53.1，得到结果如下，可知无法连接:



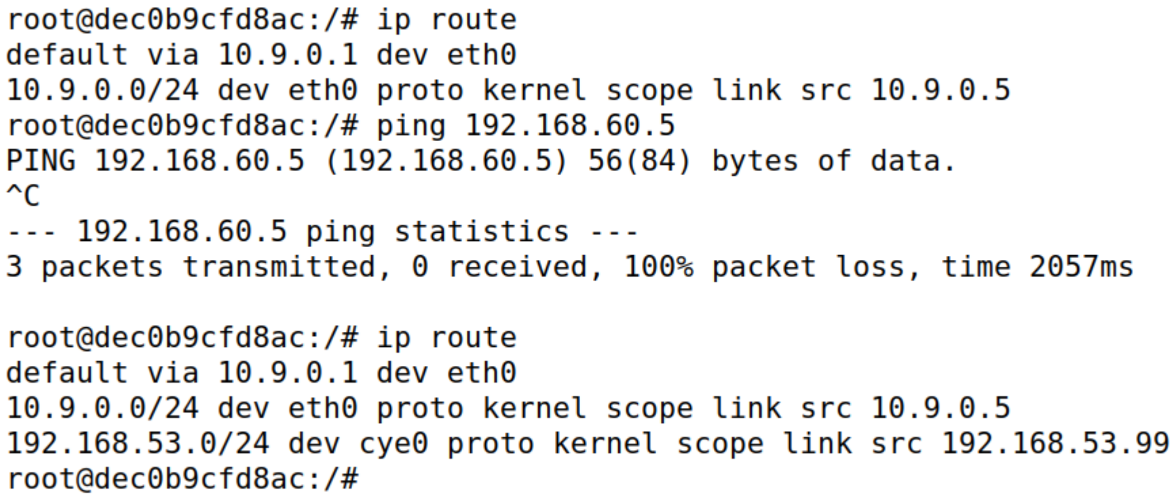
```
root@dec0b9cfd8ac:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2042ms
```

利用root权限运行该程序后，得到结果如下，可知icmp请求报文成功发送，但IP地址为192.168.53.1的主机不存在，导致ping无法连接。



```
root@dec0b9cfd8ac:/volumes# chmod a+x ./tun.py
root@dec0b9cfd8ac:/volumes# ./tun.py
Interface Name: cye0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

在主机U上ping主机V，无法连接：



```
root@dec0b9cfd8ac:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
root@dec0b9cfd8ac:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2057ms

root@dec0b9cfd8ac:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev cye0 proto kernel scope link src 192.168.53.99
root@dec0b9cfd8ac:/#
```

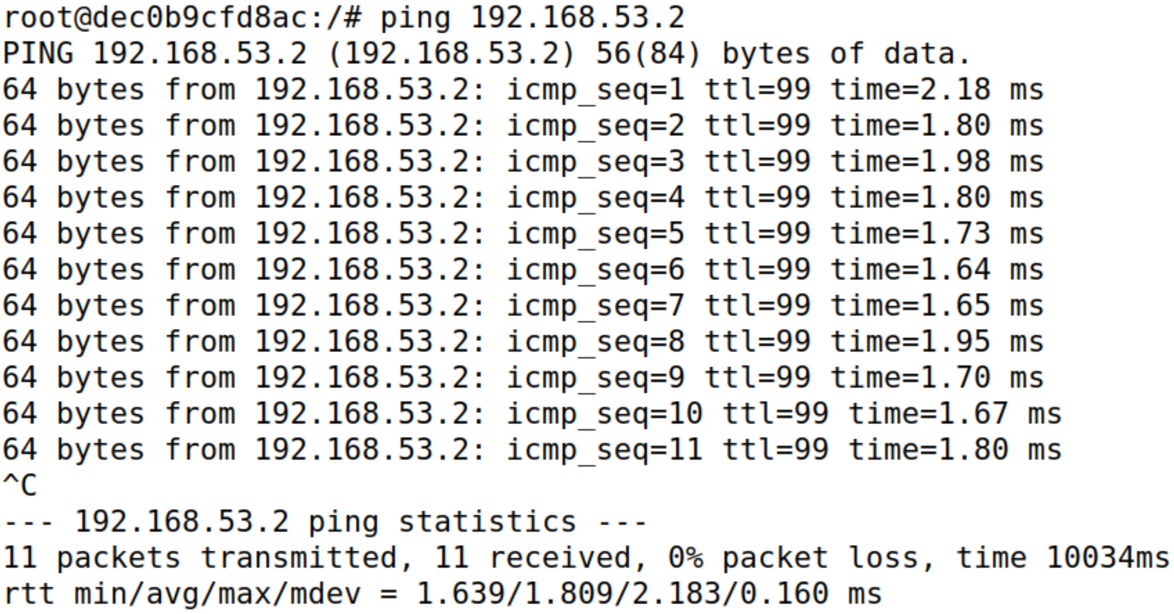
ip route命令查看路由信息，可知192.168.60.0/24的路由经过接口eth0，而非接口cye0。从10.9.0.5成功发送ICMP请求报文，但未收到ICMP响应报文。

Task2d

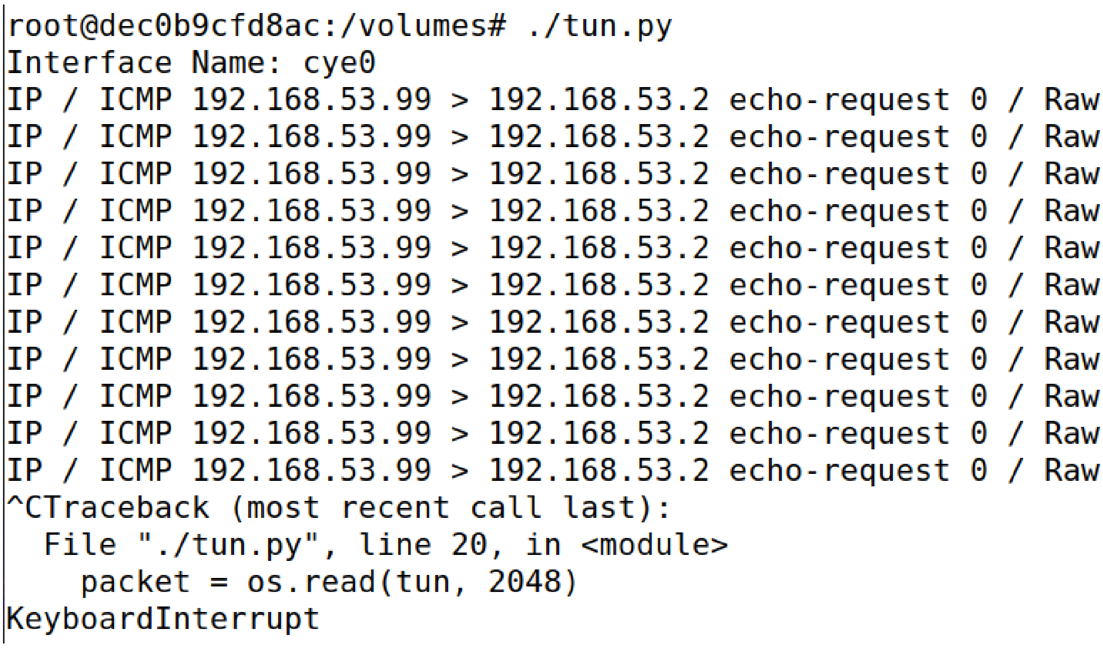
代码：

```python

#!/usr/bin/env 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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'cye%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
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))
print("Interface Name: {}".format(ifname))
while True:
    packet = os.read(tun, 2048)
    if packet:
        pkt = IP(packet)
        print(pkt.summary())
        if ICMP in pkt:
            newip = IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl,ttl=99)
            newicmp = ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
            if pkt.haslayer(Raw):
                data = pkt[Raw].load
                newpkt = newip/newicmp/data
            else:
                newpkt = newip/newicmp
        os.write(tun, bytes(newpkt))
```

在主机U上运行代码并且ping主机192.168.53.2:

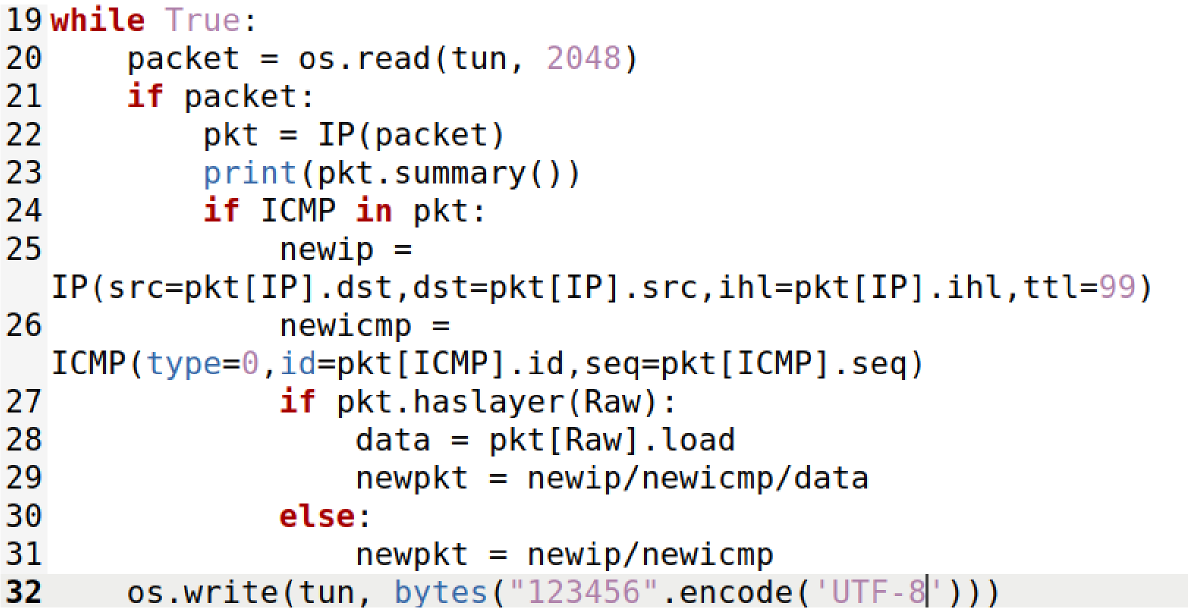


```
root@dec0b9cfd8ac:/# ping 192.168.53.2
PING 192.168.53.2 (192.168.53.2) 56(84) bytes of data.
64 bytes from 192.168.53.2: icmp_seq=1 ttl=99 time=2.18 ms
64 bytes from 192.168.53.2: icmp_seq=2 ttl=99 time=1.80 ms
64 bytes from 192.168.53.2: icmp_seq=3 ttl=99 time=1.98 ms
64 bytes from 192.168.53.2: icmp_seq=4 ttl=99 time=1.80 ms
64 bytes from 192.168.53.2: icmp_seq=5 ttl=99 time=1.73 ms
64 bytes from 192.168.53.2: icmp_seq=6 ttl=99 time=1.64 ms
64 bytes from 192.168.53.2: icmp_seq=7 ttl=99 time=1.65 ms
64 bytes from 192.168.53.2: icmp_seq=8 ttl=99 time=1.95 ms
64 bytes from 192.168.53.2: icmp_seq=9 ttl=99 time=1.70 ms
64 bytes from 192.168.53.2: icmp_seq=10 ttl=99 time=1.67 ms
64 bytes from 192.168.53.2: icmp_seq=11 ttl=99 time=1.80 ms
^C
--- 192.168.53.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10034ms
rtt min/avg/max/mdev = 1.639/1.809/2.183/0.160 ms
```



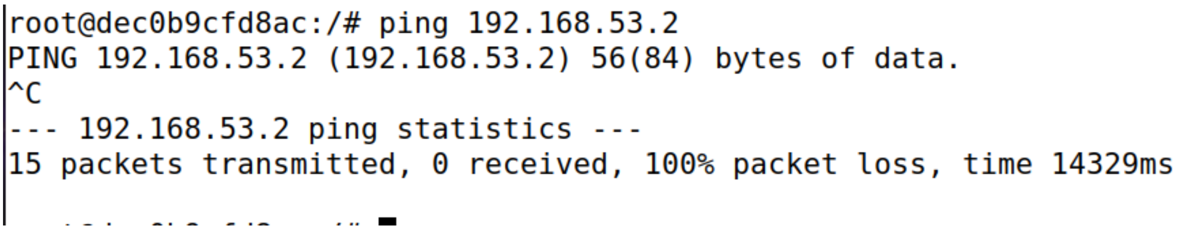
```
root@dec0b9cfd8ac:/volumes# ./tun.py
Interface Name: cye0
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
^CTraceback (most recent call last):
  File "./tun.py", line 20, in <module>
    packet = os.read(tun, 2048)
KeyboardInterrupt
```

ping 192.168.60.0/24无法捕获。修改代码为改为其他填充:



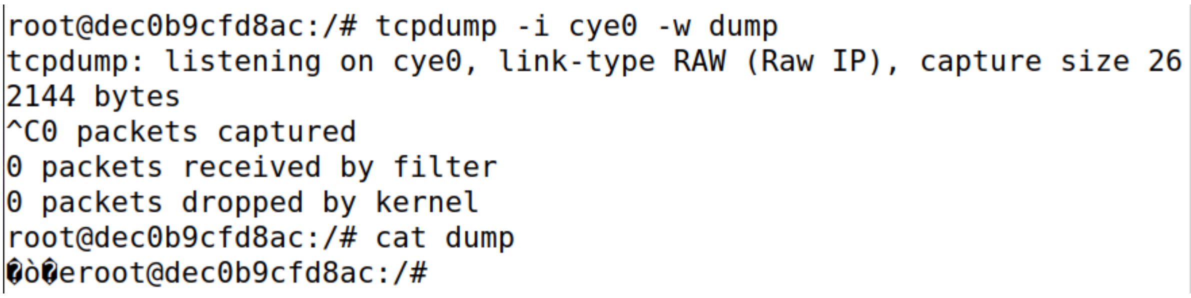
```
while True:
    packet = os.read(tun, 2048)
    if packet:
        pkt = IP(packet)
        print(pkt.summary())
        if ICMP in pkt:
            newip = IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl,ttl=99)
            newicmp = ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
            if pkt.haslayer(Raw):
                data = pkt[Raw].load
                newpkt = newip/newicmp/data
            else:
                newpkt = newip/newicmp
    os.write(tun, bytes("123456".encode('UTF-8')))
```

发现ping不通



```
root@dec0b9cfd8ac:/# ping 192.168.53.2
PING 192.168.53.2 (192.168.53.2) 56(84) bytes of data.
^C
--- 192.168.53.2 ping statistics ---
15 packets transmitted, 0 received, 100% packet loss, time 14329ms
```

在主机U上利用tcpdump命令抓取数据包，得到结果如下：



```
root@dec0b9cfd8ac:/# tcpdump -i cye0 -w dump
tcpdump: listening on cye0, link-type RAW (Raw IP), capture size 26
2144 bytes
^C0 packets captured
0 packets received by filter
0 packets dropped by kernel
root@dec0b9cfd8ac:/# cat dump
�ò�eroot@dec0b9cfd8ac:/#
```

Task3

client运行tun_client.py

```
#!/usr/bin/env 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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'cye%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
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))
print("Interface Name: {}".format(ifname))
```

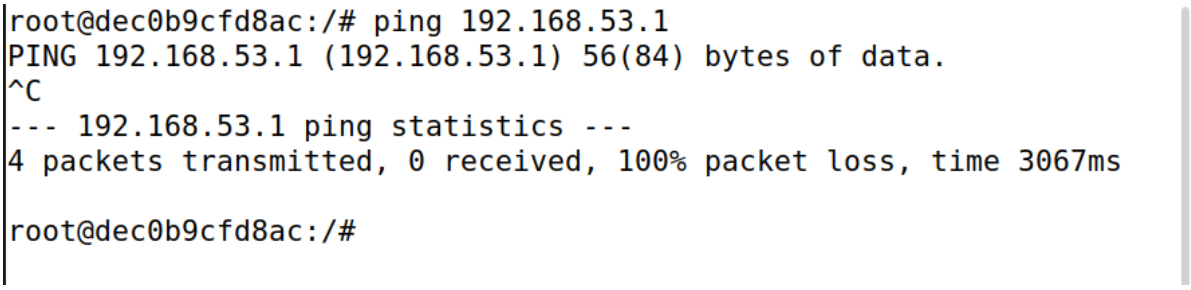
```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    packet = os.read(tun, 2048)
    if packet:
        sock.sendto(packet, ("10.9.0.11", 9090))
```

server运行server.py

```
#!/usr/bin/env 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))
```

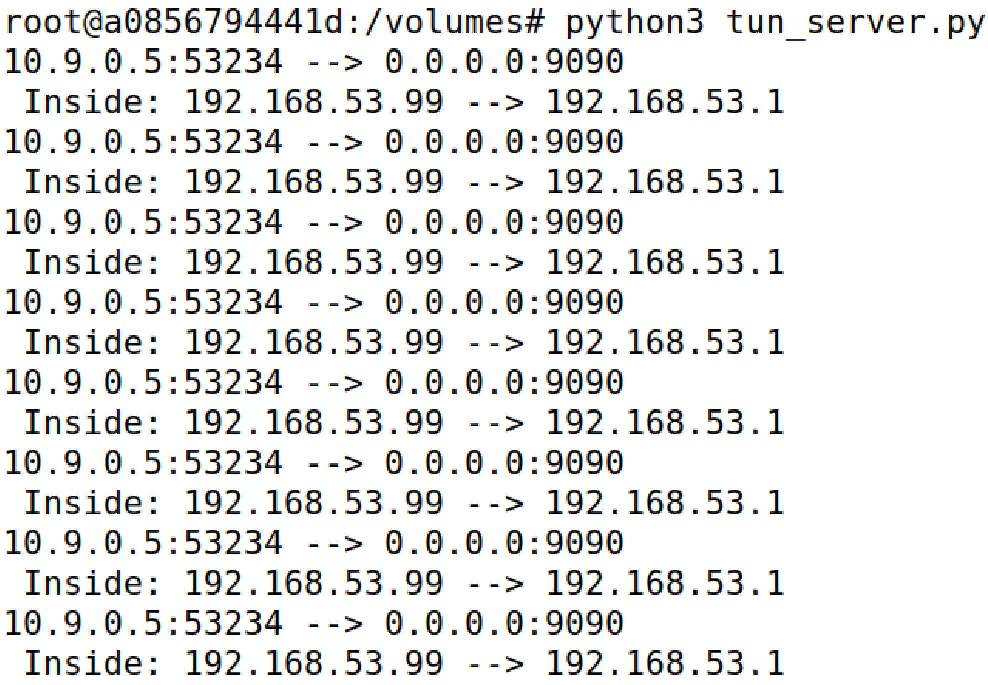
在主机U上利用root权限运行tun_client程序后，创建tun接口:

在主机U上ping主机192.168.53.1



```
root@dec0b9cfd8ac:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3067ms

root@dec0b9cfd8ac:/#
```

在VPN服务器上利用root权限运行tun_server程序后



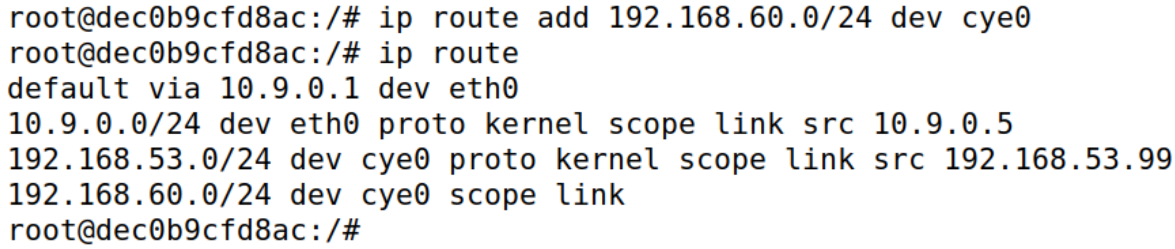
```
root@a0856794441d:/volumes# python3 tun_server.py
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:53234 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
```

192.168.53.0/24的路由经过接口cye0,成功通过隧道发送udp报文

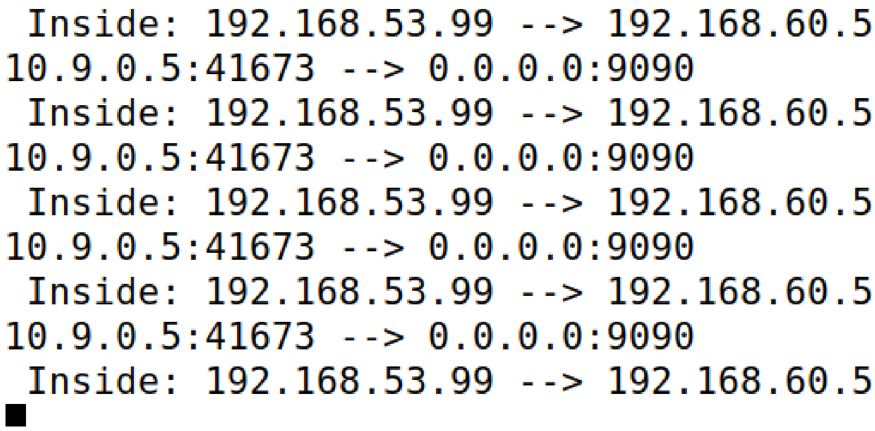
在主机U上ping主机V，得到结果如下，可知无法连接,因为192.168.60.0/24的路由不经过接口cye0。

在主机U上利用ip route命令设置192.168.60.0/24的路由经过接口cye0,在主机U上ping主机V并

在VPN服务器上利用root权限运行tun_server程序，得到



```
root@dec0b9cfd8ac:/# ip route add 192.168.60.0/24 dev cye0
root@dec0b9cfd8ac:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev cye0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev cye0 scope link
root@dec0b9cfd8ac:/#
```



```
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:41673 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:41673 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:41673 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:41673 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

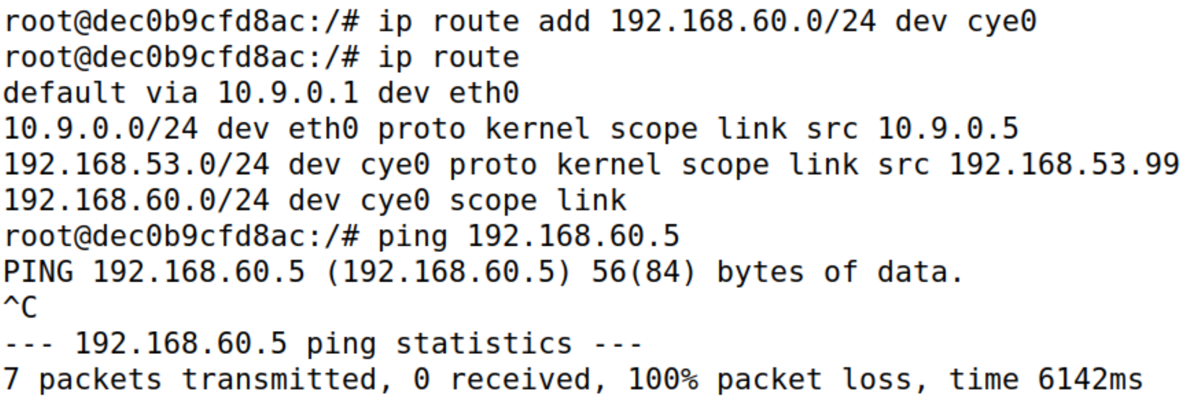
Task4

修改tun_server.py：

```
#!/usr/bin/env python3
import fcntl
import struct
import os
import time
```

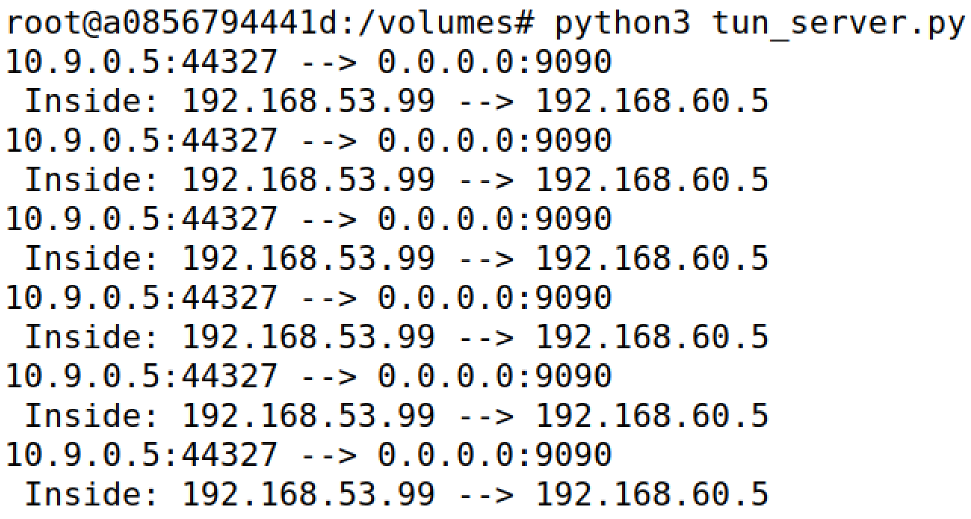
```
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'cye%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.1/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
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))
    os.write(tun, bytes(pkt))
```

在主机U上利用root权限运行tun_client程序后，创建tun接口,并且设置192.168.60.0/24的路由经过接口，在主机U上ping主机V:



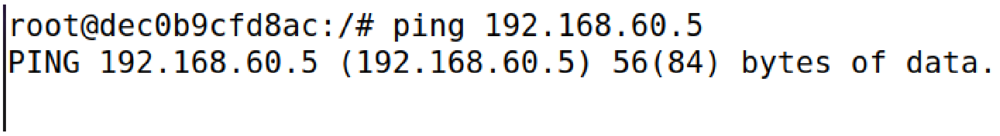
```
root@dec0b9cfd8ac:/# ip route add 192.168.60.0/24 dev cye0
root@dec0b9cfd8ac:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev cye0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev cye0 scope link
root@dec0b9cfd8ac:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6142ms
```

在VPN服务器上利用root权限运行tun_server程序后:



```
root@a0856794441d:/volumes# python3 tun_server.py
10.9.0.5:44327 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:44327 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:44327 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:44327 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:44327 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:44327 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

可知ICMP请求和响应报文都已经发送，但主机U未收到ICMP响应报文。



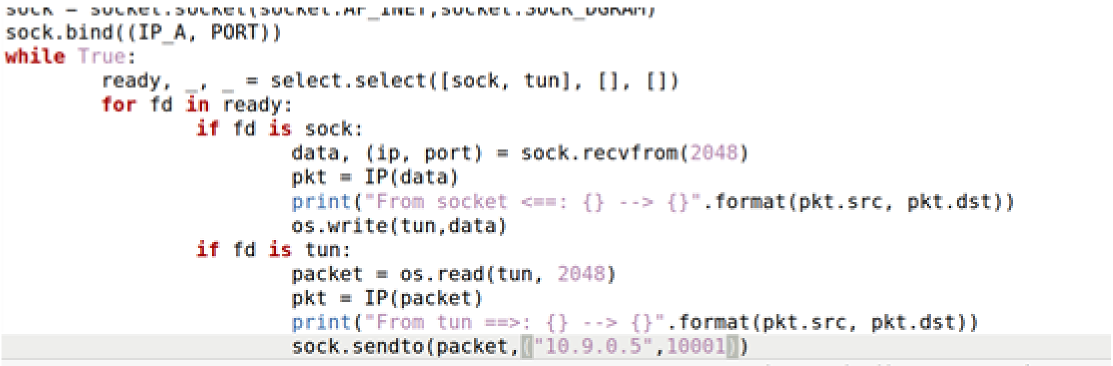
```
root@dec0b9cfd8ac:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

Task5

隧道的一个方向就完成了.我们可以看到Host V已经发送了响应，但是数据包被丢到了某个地方。

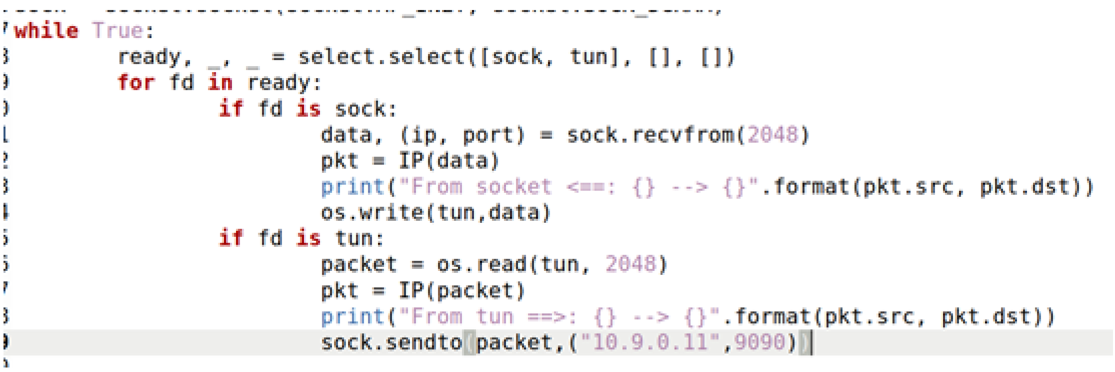
我们需要设置它的另一个方向，这样返回的流量可以通过隧道回到主机U。TUN客户机和服务器程序需要从两个接口读取数据，即TUN接口和套接字接口。

tun_server.py的程序，将while部分修改如下，将在隧道中的报文发给10.9.0.5的10001端口



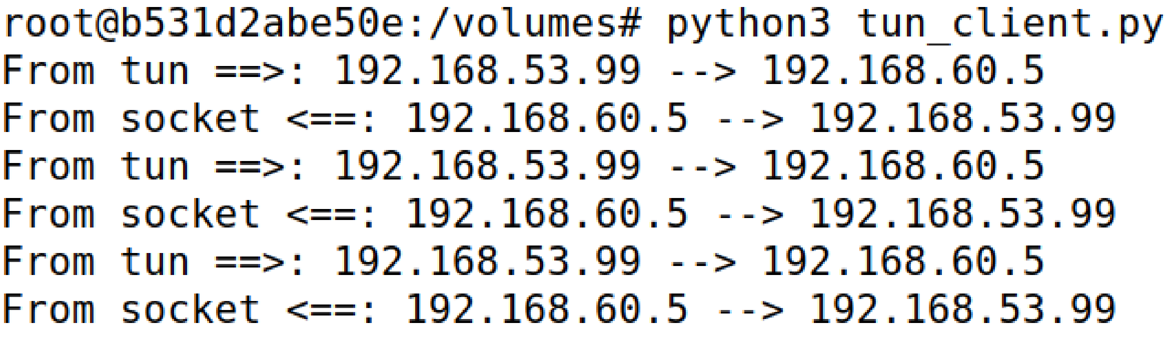
```
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
while True:
        ready, _, _ = select.select([sock, tun], [], [])
        for fd in ready:
                if fd is sock:
                        data, (ip, port) = sock.recvfrom(2048)
                        pkt = 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,("10.9.0.5",10001))
```

tun_client.py程序，将隧道中的报文发给router的10.9.0.11的9090端口



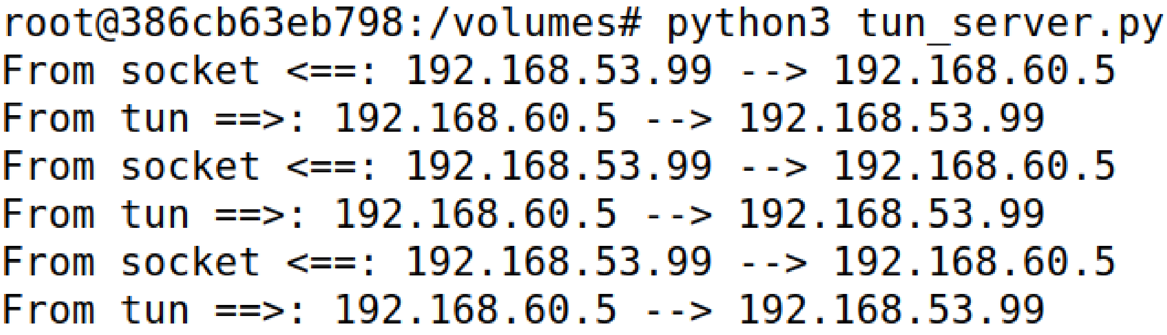
```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
        ready, _, _ = select.select([sock, tun], [], [])
        for fd in ready:
                if fd is sock:
                        data, (ip, port) = sock.recvfrom(2048)
                        pkt = IP(data)
                        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,("10.9.0.11",9090))
```

主机U上ping主机V，发现可以ping通，在主机U上利用root权限运行tun_client程序后：



```
root@b531d2abe50e:/volumes# python3 tun_client.py
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
```

在VPN服务器上利用root权限运行tun_server程序

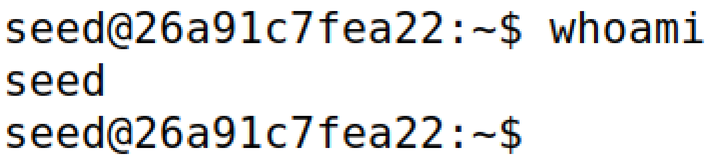


```
root@386cb63eb798:/volumes# python3 tun_server.py
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

在主机U上telnet远程连接主机V，连接成功。

Task6

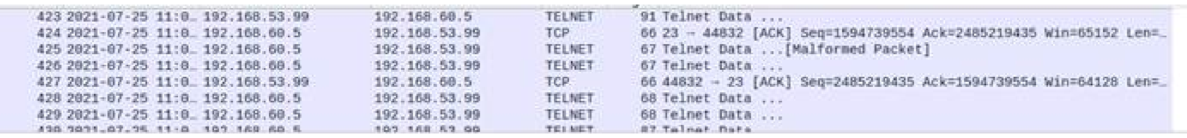
首先打开隧道，可以成功telnet，之后关闭tun_client.py，发现在shell输入命令无法输入，发现连接中断。



```
seed@26a91c7fea22:~$ whoami
seed
seed@26a91c7fea22:~$
```

之后重新运行tun_client.py后，可以重新连接。tun_server程序停止后，无法传输报文。

抓包可得，二者重建立TCP连接



```
423 2021-07-25 11:0… 192.168.53.99     192.168.60.5      TELNET     91 Telnet Data ...
424 2021-07-25 11:0… 192.168.60.5      192.168.53.99     TCP        66 23 → 44832 [ACK] Seq=1594739554 Ack=2485219435 Win=65152 Len=…
425 2021-07-25 11:0… 192.168.60.5      192.168.53.99     TELNET     67 Telnet Data ...[Malformed Packet]
426 2021-07-25 11:0… 192.168.60.5      192.168.53.99     TELNET     67 Telnet Data ...
427 2021-07-25 11:0… 192.168.53.99     192.168.60.5      TCP        66 44832 → 23 [ACK] Seq=2485219435 Ack=1594739554 Win=64128 Len=…
428 2021-07-25 11:0… 192.168.60.5      192.168.53.99     TELNET     68 Telnet Data ...
429 2021-07-25 11:0… 192.168.60.5      192.168.53.99     TELNET     68 Telnet Data ...
430 2021-07-25 11:0… 192.168.60.5      192.168.53.99     TELNET     87 Telnet Data
```