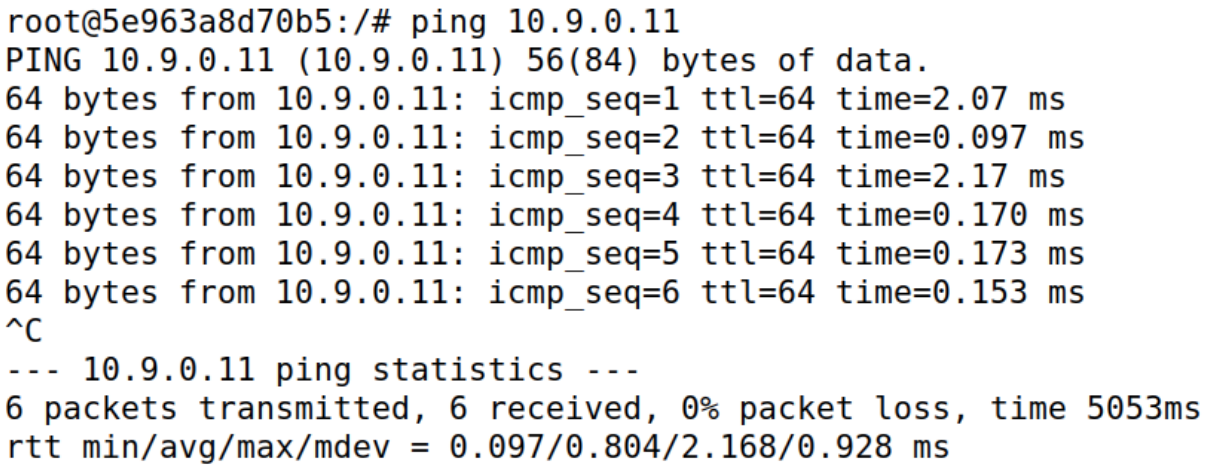
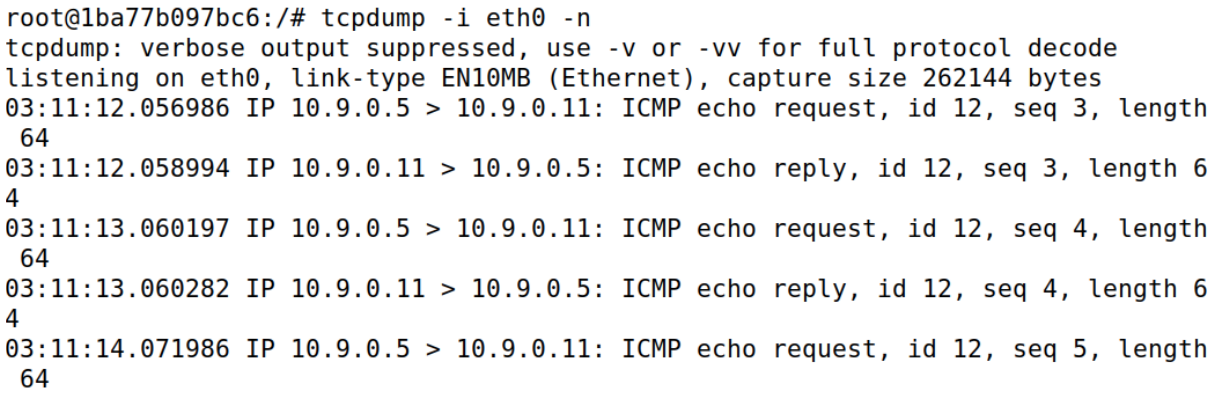
Lab7 VPN Lab: The Container Version

Task 1: Network Setup

1. 在主机U上ping VPN server，并在路由器上捕获报文，结果如下：



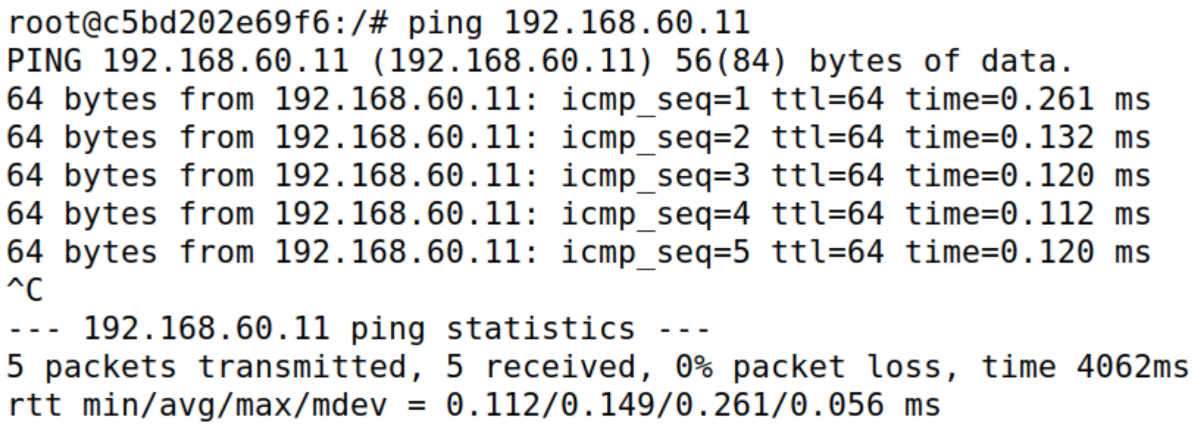
```
root@5e963a8d70b5:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=2.07 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.097 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=2.17 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.170 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.173 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.153 ms
^C
--- 10.9.0.11 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5053ms
rtt min/avg/max/mdev = 0.097/0.804/2.168/0.928 ms
```



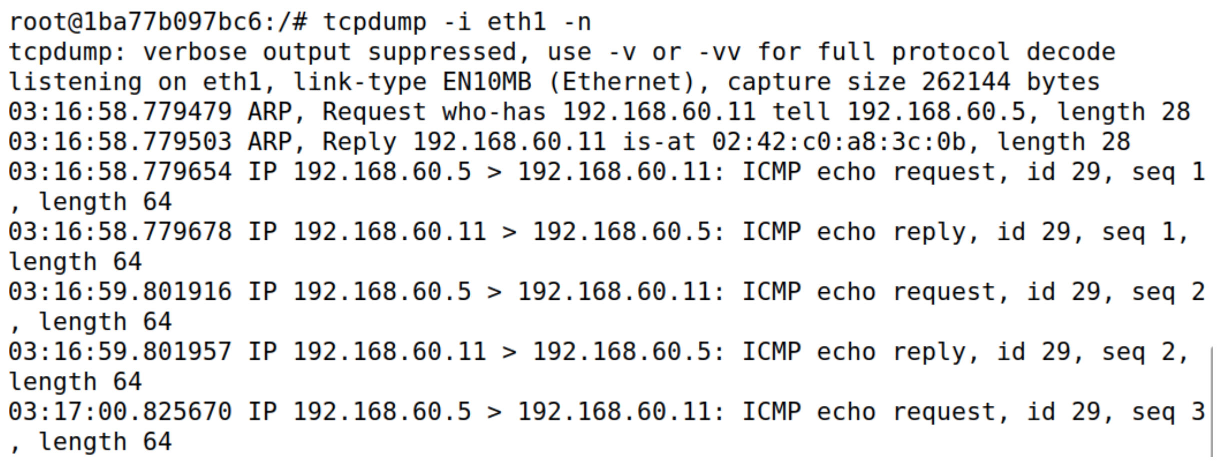
```
root@1ba77b097bc6:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
03:11:12.056986 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 3, length
 64
03:11:12.058994 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 3, length 6
4
03:11:13.060197 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 4, length
 64
03:11:13.060282 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 4, length 6
4
03:11:14.071986 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 5, length
 64
```

可见主机U可以与VPN server通信，并能在路由器上捕获到报文。

2. 在主机V上ping VPN server，并在路由器上捕获报文，结果如下：



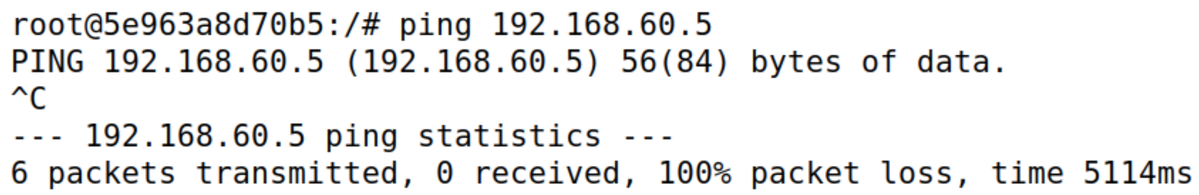
```
root@c5bd202e69f6:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.261 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.132 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.120 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.112 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=64 time=0.120 ms
^C
--- 192.168.60.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4062ms
rtt min/avg/max/mdev = 0.112/0.149/0.261/0.056 ms
```



```
root@1ba77b097bc6:/# tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
03:16:58.779479 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
03:16:58.779503 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
03:16:58.779654 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 29, seq 1
, length 64
03:16:58.779678 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 29, seq 1,
length 64
03:16:59.801916 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 29, seq 2
, length 64
03:16:59.801957 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 29, seq 2,
length 64
03:17:00.825670 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 29, seq 3
, length 64
```

可见主机V可以与VPN server通信，并能在路由器上捕获到报文。

3. 在主机U上ping主机V，结果如下：



```
root@5e963a8d70b5:/# 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 ---
6 packets transmitted, 0 received, 100% packet loss, time 5114ms
```

可见主机U不能和主机V通信。

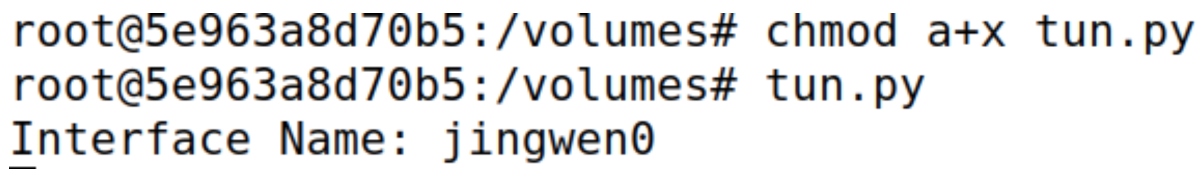
Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

1. tun.py代码更改如下：

ifr = struct.pack('16sH', b'jingwen%d', IFF_TUN | IFF_NO_PI)

2. 在主机U上运行tun.py，结果如下：



```
root@5e963a8d70b5:/volumes# chmod a+x tun.py
root@5e963a8d70b5:/volumes# tun.py
Interface Name: jingwen0
```

3. 在主机U上执行命令ip address查看所有接口，结果如下：

```
root@5e963a8d70b5:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
2: jingwen0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group
default qlen 500
    link/none
73: eth0@if74: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
```

可见tun接口名被改成jingwen0。

Task 2.b: Set up the TUN Interface

1. 在tun.py中添加代码如下：

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

os.system("ip link set dev {} up".format(ifname))

2. 在主机U上运行tun.py，执行命令ip address查看所有接口，结果如下：

```
root@5e963a8d70b5:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
3: jingwen0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel s
tate UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global jingwen0
       valid_lft forever preferred_lft forever
73: eth0@if74: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
```

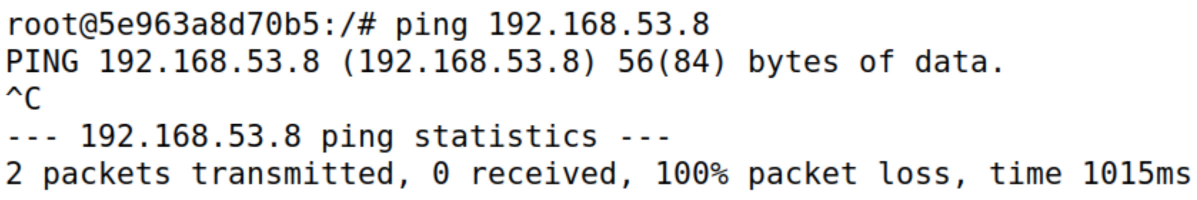
可见tun接口绑定了相应的ip地址。

Task 2.c: Read from the TUN Interface

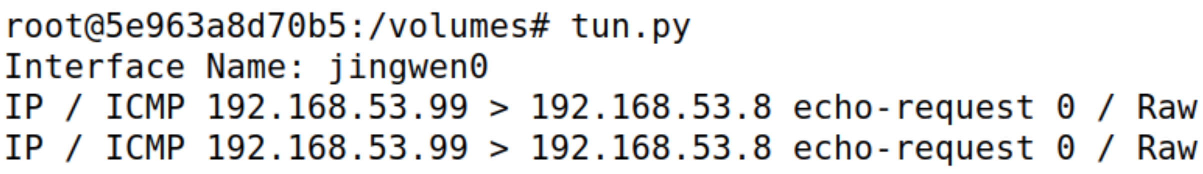
1. 在tun.py中添加代码如下：

```
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                print(ip.summary())
```

2. 在主机U上运行tun.py，ping 192.168.53.8，结果如下：



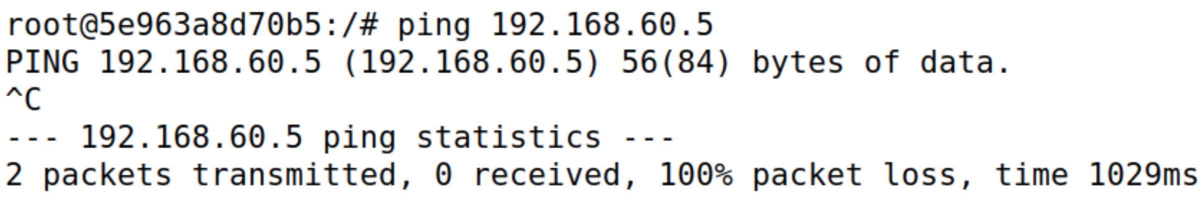
```
root@5e963a8d70b5:/# ping 192.168.53.8
PING 192.168.53.8 (192.168.53.8) 56(84) bytes of data.
^C
--- 192.168.53.8 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1015ms
```



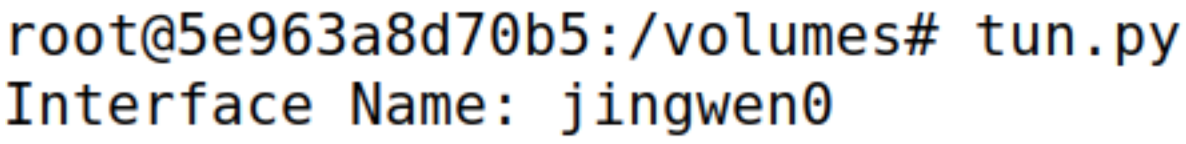
```
root@5e963a8d70b5:/volumes# tun.py
Interface Name: jingwen0
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
```

可见由于主机不存在，所以没有得到相应。

3. 在主机U上运行tun.py，ping 192.168.60.5，结果如下：



```
root@5e963a8d70b5:/# 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 ---
2 packets transmitted, 0 received, 100% packet loss, time 1029ms
```



```
root@5e963a8d70b5:/volumes# tun.py
Interface Name: jingwen0
```

可见由于没有添加路由，程序没有输出。

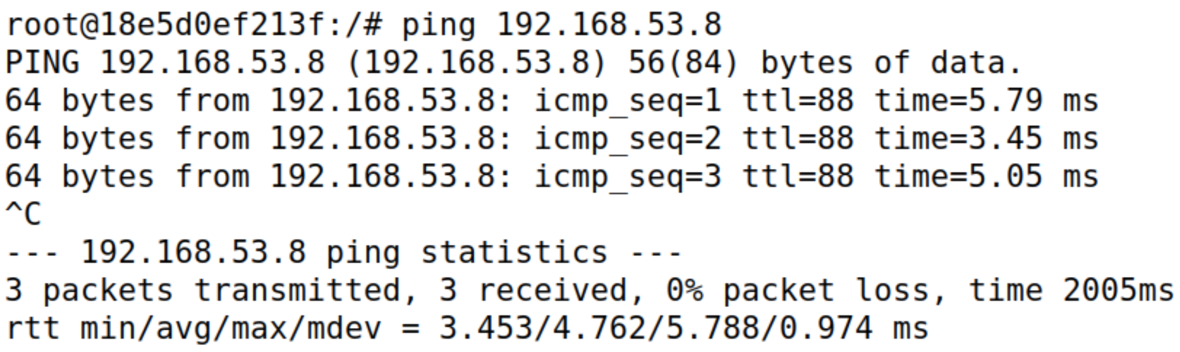
Task 2.d: Write to the TUN Interface

1. tun.py代码更改如下：

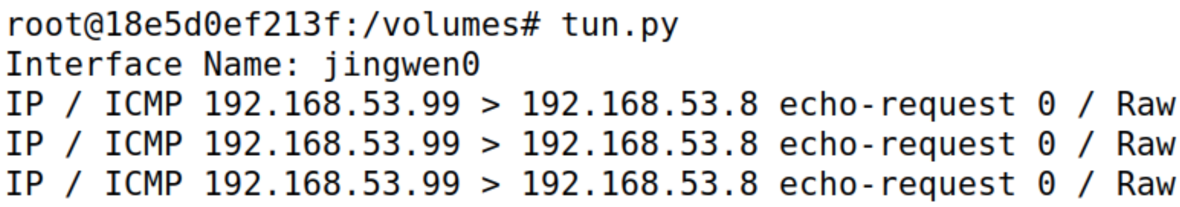
```
#!/usr/bin/env python3


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'jingwen%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))


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

os.system("ip link set dev {} up".format(ifname))
```

```
while True:
    # Get a packet from the tun interface
    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)
        newip.ttl = 88
        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))
```

2. 在主机U上运行tun.py，ping 192.168.53.8，结果如下：



```
root@18e5d0ef213f:/# ping 192.168.53.8
PING 192.168.53.8 (192.168.53.8) 56(84) bytes of data.
64 bytes from 192.168.53.8: icmp_seq=1 ttl=88 time=5.79 ms
64 bytes from 192.168.53.8: icmp_seq=2 ttl=88 time=3.45 ms
64 bytes from 192.168.53.8: icmp_seq=3 ttl=88 time=5.05 ms
^C
--- 192.168.53.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 3.453/4.762/5.788/0.974 ms
```



```
root@18e5d0ef213f:/volumes# tun.py
Interface Name: jingwen0
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
```

可见ICMP报文被响应。

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

1. 客户端程序tun.py代码更改如下：

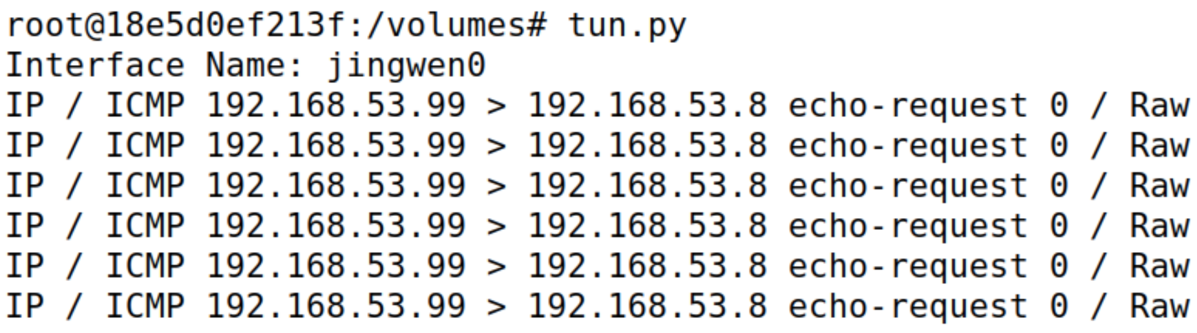
```
#!/usr/bin/env python3

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'jingwen%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))

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 192.168.60.0/24 dev {}".format(ifname))

# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="10.9.0.11"
SERVER_PORT=9090
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                pkt = IP(packet)
                print(pkt.summary())
                sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

2. 服务器端程序tun_server.py代码如下：

```
#!/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'jingwen%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))
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 192.168.60.0/24 dev {}".format(ifname))

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP = "0.0.0.0"
SERVER_PORT = 9090
server.bind((SERVER_IP, SERVER_PORT))

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

3. 在主机U上运行tun.py，在VPN server上运行tun_server.py，在主机U上ping 192.168.53.8，结果如下：

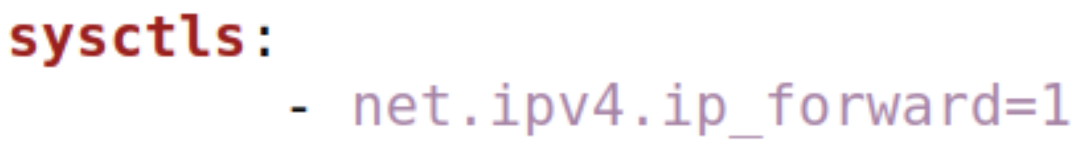


```
root@18e5d0ef213f:/volumes# tun.py
Interface Name: jingwen0
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.8 echo-request 0 / Raw
```

可见管道外部是10.9.0.5->0.0.0.0，管道内部是192.168.53.99->192.168.53.8。

Task 4: Set Up the VPN Server

1. 在路由器上打开IP转发，结果如下：



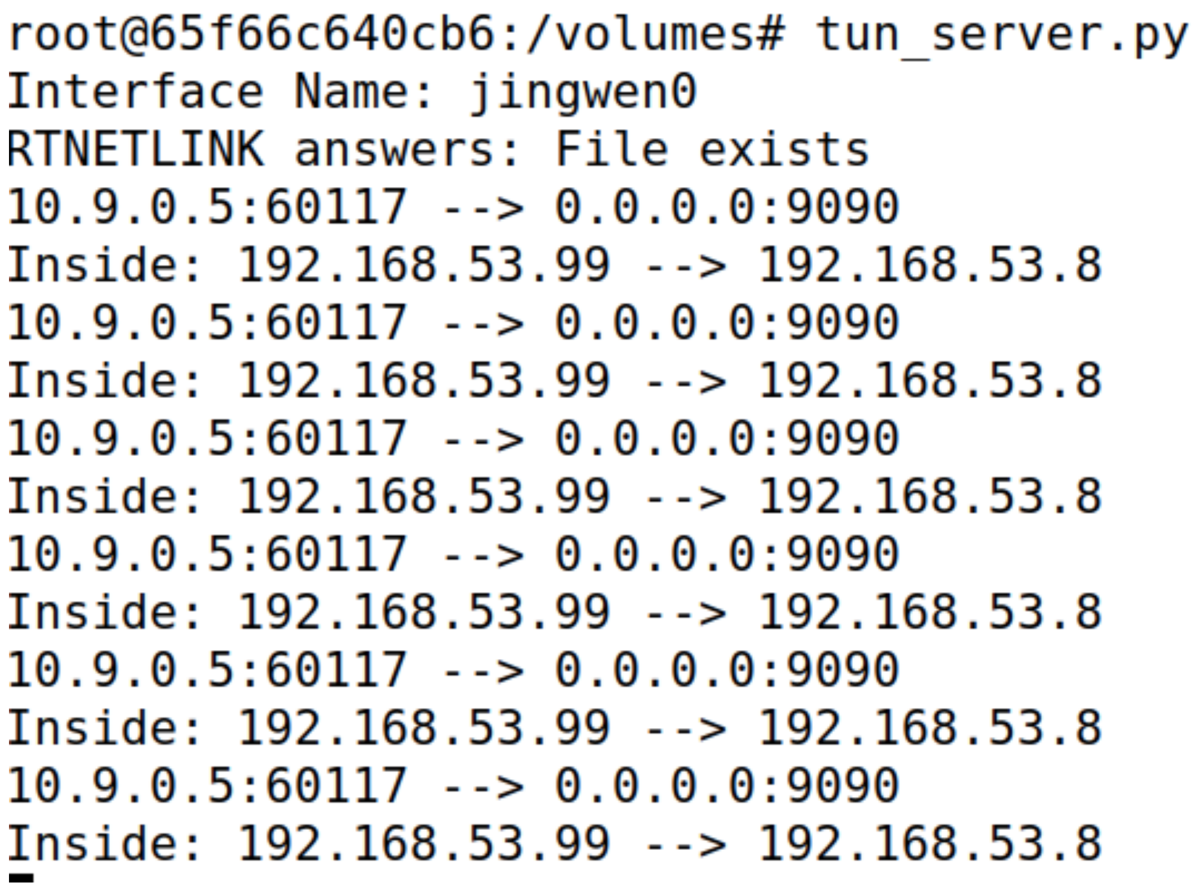
```
sysctls:
        - net.ipv4.ip_forward=1
```

2. 服务器程序tun_server.py代码更改如下：

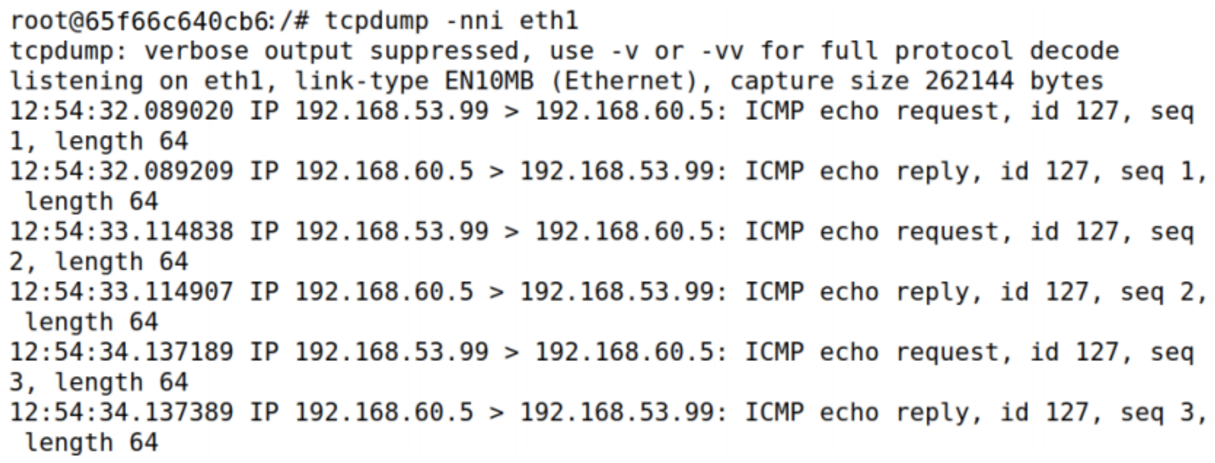
```
#!/usr/bin/env python3
import fcntl
import struct
import os
import time
from scapy.all import *

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'jingwen%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))
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 192.168.60.0/24 dev {}".format(ifname))

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP = "0.0.0.0"
SERVER_PORT = 9090
server.bind((SERVER_IP, SERVER_PORT))

while True:
    data,(ip, port) = server.recvfrom(2048)
    print("{}:{} --> {}:{}".format(ip, port, SERVER_IP, SERVER_PORT))
    pkt = IP(data)
    print("Inside: {} --> {}".format(pkt.src, pkt.dst))
    os.write(tun, data)
    print("write")
```

3. 在主机U上ping 192.168.60.5，在VPN server上执行命令tcpdump -nni eth1，结果如下：



```
root@65f66c640cb6:/volumes# tun_server.py
Interface Name: jingwen0
RTNETLINK answers: File exists
10.9.0.5:60117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.8
10.9.0.5:60117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.8
10.9.0.5:60117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.8
10.9.0.5:60117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.8
10.9.0.5:60117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.8
10.9.0.5:60117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.8
```



```
root@65f66c640cb6:/# tcpdump -nni eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
12:54:32.089020 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 127, seq
1, length 64
12:54:32.089209 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 127, seq 1,
 length 64
12:54:33.114838 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 127, seq
2, length 64
12:54:33.114907 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 127, seq 2,
 length 64
12:54:34.137189 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 127, seq
3, length 64
12:54:34.137389 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 127, seq 3,
 length 64
```

可见有报文返回。

Task 5: Handling Traffic in Both Directions

1. 客户端程序tun.py代码更改如下：

```
#!/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'jingwen%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))


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 192.168.60.0/24 dev jingwen0 via
192.168.53.99".format(ifname))
```

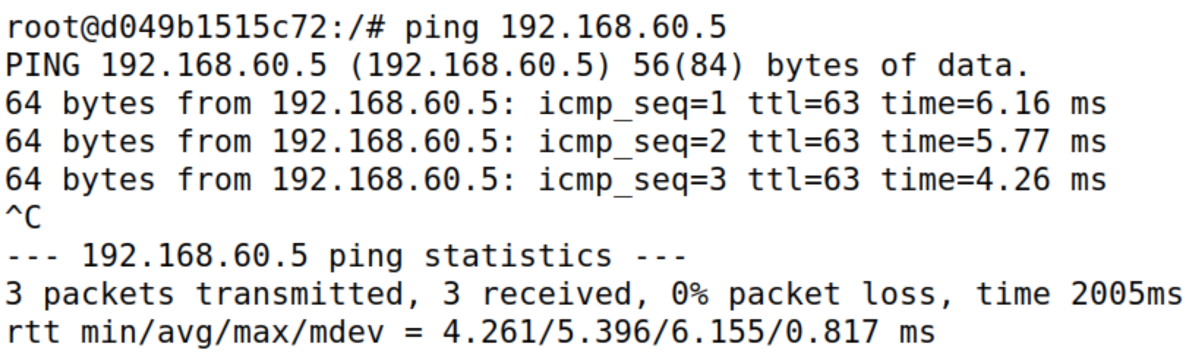
```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
sock.bind((SERVER_IP, SERVER_PORT))

while True:
    # Get a packet from the tun interface
    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, bytes(pkt))
        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))
```

2. 服务器程序tun_server.py代码更改如下：

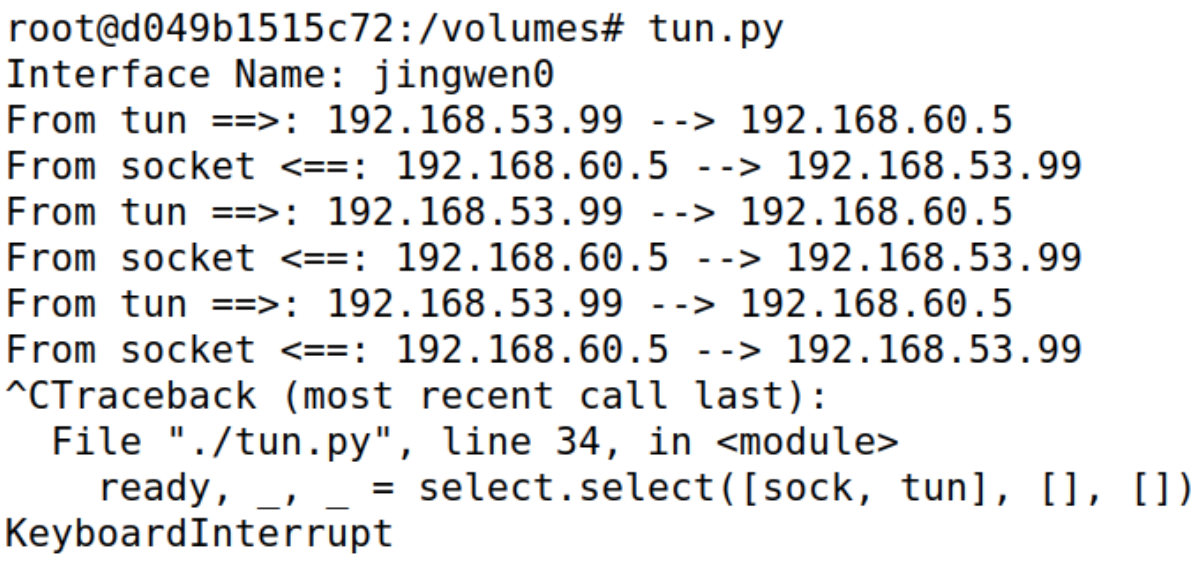
```
#!/usr/bin/env python3
import fcntl
import struct
import os
import time
from scapy.all import *
```

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

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP = "0.0.0.0"
SERVER_PORT = 9090
server.bind((SERVER_IP, SERVER_PORT))

while True:
        ready, _, _ = select.select([server, tun], [], [])
        for fd in ready:
                if fd is server:
                        data, (ip, port) = server.recvfrom(2048)
                        print("{}:{}-->{}:
{}".format('10.9.0.5',9090,SERVER_IP,SERVER_PORT))
                        pkt = IP(data)
                        print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
                        os.write(tun, bytes(pkt))
                if fd is tun:
                        packet = os.read(tun, 2048)
                        pkt = IP(packet)
                        print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
                        server.sendto(packet, ('10.9.0.5', 9090))
```

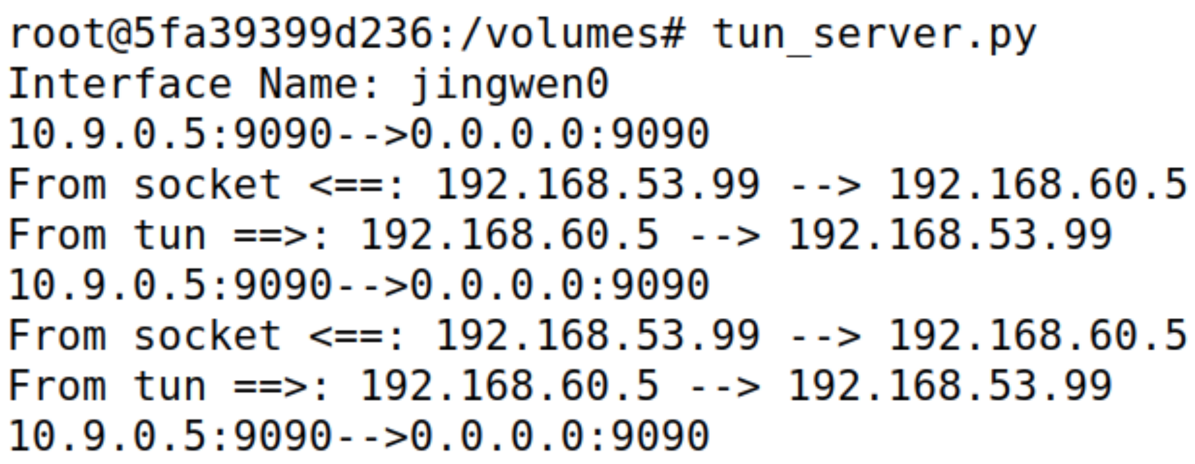
3. 在主机U上运行tun.py，在VPN server上运行tun_server.py，在主机U上ping 主机V，并用Wireshark抓包，结果如下：



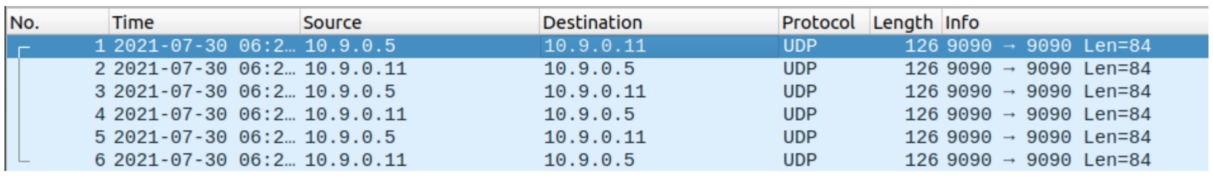
```
root@d049b1515c72:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=6.16 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=5.77 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=4.26 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 4.261/5.396/6.155/0.817 ms
```



```
root@d049b1515c72:/volumes# tun.py
Interface Name: jingwen0
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
^CTraceback (most recent call last):
  File "./tun.py", line 34, in <module>
    ready, _, _ = select.select([sock, tun], [], [])
KeyboardInterrupt
```



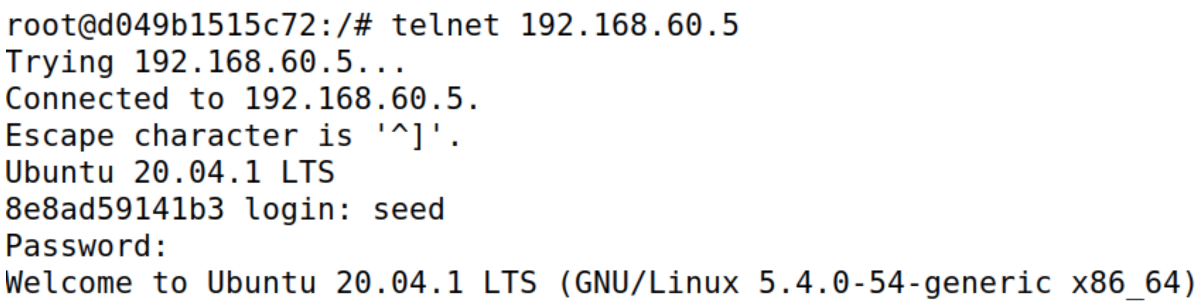
```
root@5fa39399d236:/volumes# tun_server.py
Interface Name: jingwen0
10.9.0.5:9090-->0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
10.9.0.5:9090-->0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
10.9.0.5:9090-->0.0.0.0:9090
```



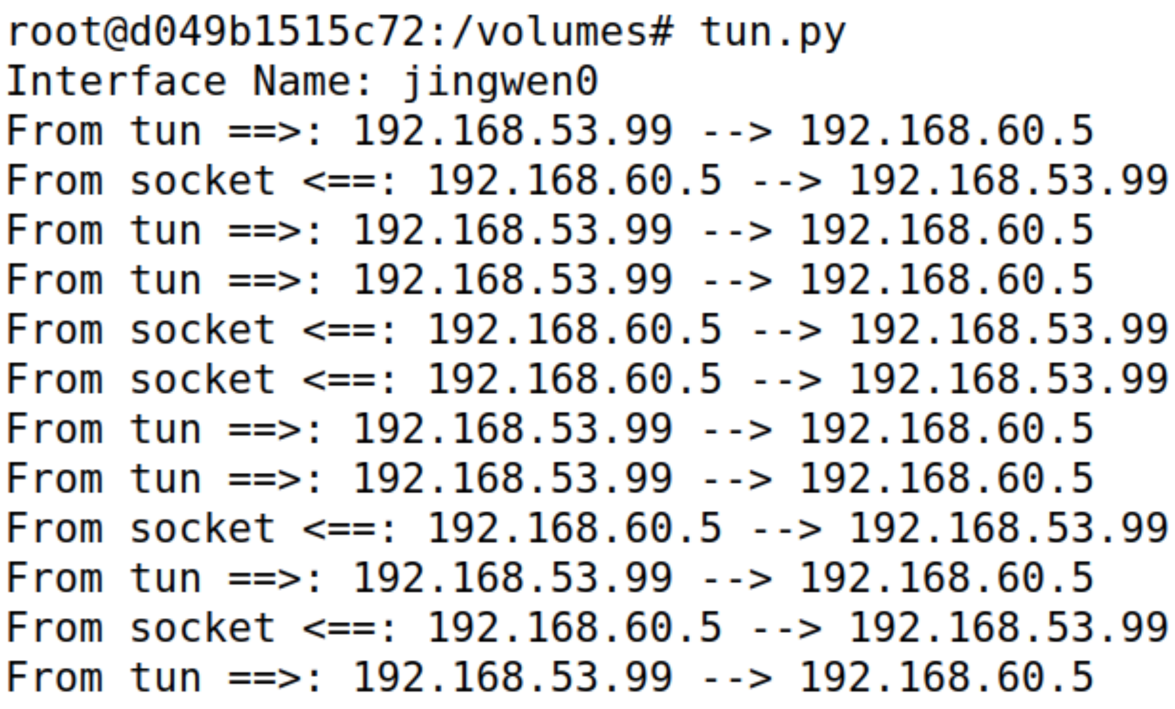
No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-30 06:2…	10.9.0.5	10.9.0.11	UDP	126	9090 → 9090 Len=84
2	2021-07-30 06:2…	10.9.0.11	10.9.0.5	UDP	126	9090 → 9090 Len=84
3	2021-07-30 06:2…	10.9.0.5	10.9.0.11	UDP	126	9090 → 9090 Len=84
4	2021-07-30 06:2…	10.9.0.11	10.9.0.5	UDP	126	9090 → 9090 Len=84
5	2021-07-30 06:2…	10.9.0.5	10.9.0.11	UDP	126	9090 → 9090 Len=84
6	2021-07-30 06:2…	10.9.0.11	10.9.0.5	UDP	126	9090 → 9090 Len=84

可见主机U可以ping主机V。

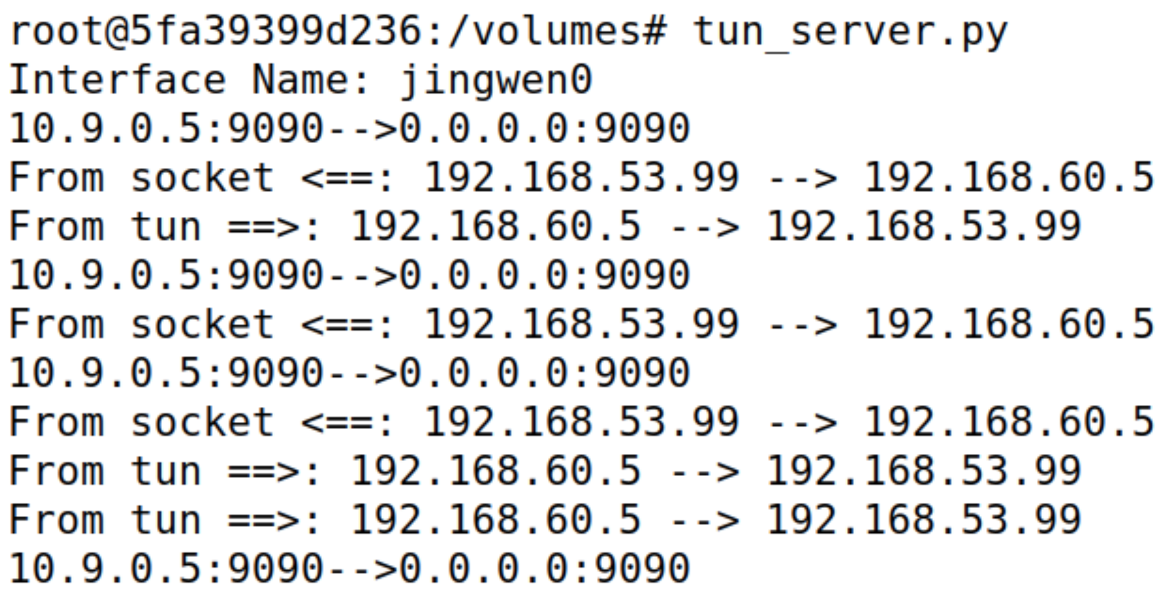
4. 在主机U上telnet 主机V，并用Wireshark抓包，结果如下：



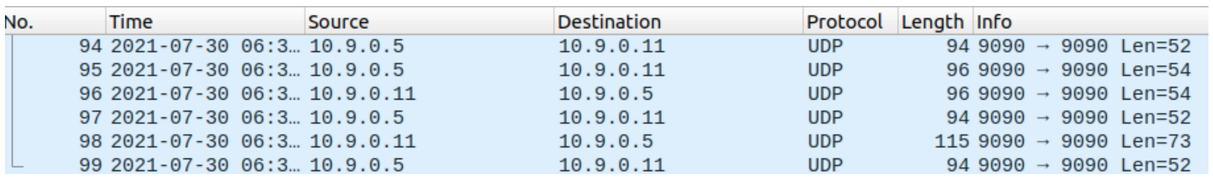
```
root@d049b1515c72:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8e8ad59141b3 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```



```
root@d049b1515c72:/volumes# tun.py
Interface Name: jingwen0
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 tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
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
```



```
root@5fa39399d236:/volumes# tun_server.py
Interface Name: jingwen0
10.9.0.5:9090-->0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
10.9.0.5:9090-->0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
10.9.0.5:9090-->0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
10.9.0.5:9090-->0.0.0.0:9090
```

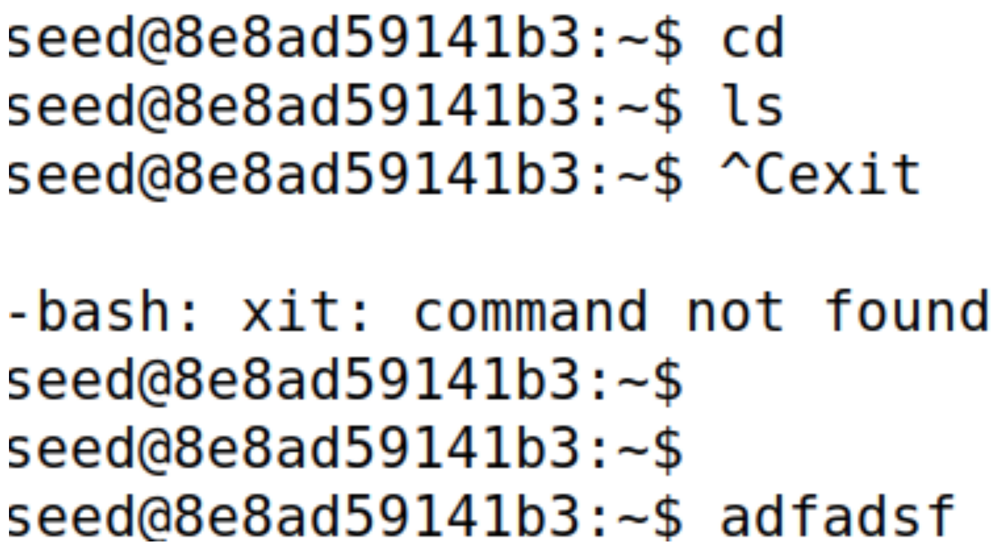


No.	Time	Source	Destination	Protocol	Length	Info
94	2021-07-30 06:3…	10.9.0.5	10.9.0.11	UDP	94	9090 → 9090 Len=52
95	2021-07-30 06:3…	10.9.0.5	10.9.0.11	UDP	96	9090 → 9090 Len=54
96	2021-07-30 06:3…	10.9.0.11	10.9.0.5	UDP	96	9090 → 9090 Len=54
97	2021-07-30 06:3…	10.9.0.5	10.9.0.11	UDP	94	9090 → 9090 Len=52
98	2021-07-30 06:3…	10.9.0.11	10.9.0.5	UDP	115	9090 → 9090 Len=73
99	2021-07-30 06:3…	10.9.0.5	10.9.0.11	UDP	94	9090 → 9090 Len=52

可见主机U可以telnet主机V。

Task 6: Tunnel-Breaking Experiment

1. 断开client或 server程序，telnet也断开，直接卡死。tunnel重新建立后，telnet也会重新建立，可以继续输入命令，结果如下：



```
seed@8e8ad59141b3:~$ cd
seed@8e8ad59141b3:~$ ls
seed@8e8ad59141b3:~$ ^Cexit

-bash: xit: command not found
seed@8e8ad59141b3:~$
seed@8e8ad59141b3:~$
seed@8e8ad59141b3:~$ adfadsf
```