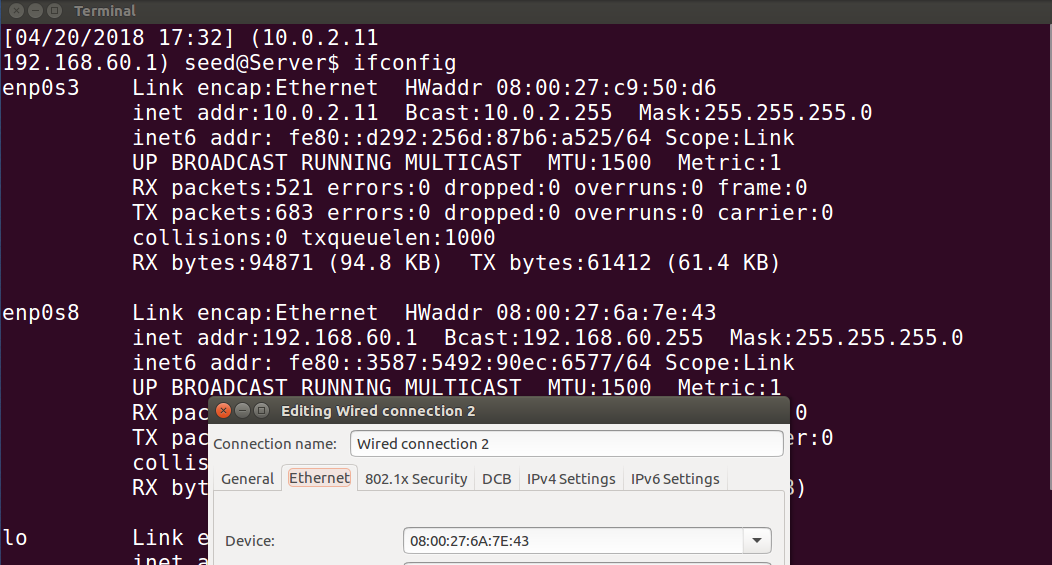
**Lab7实验报告**

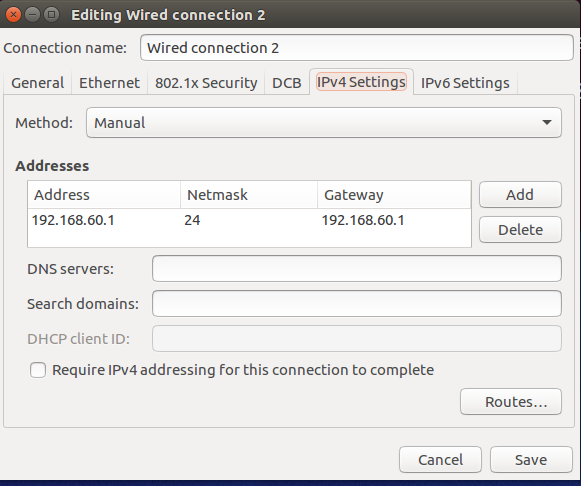
57117138 吴伊杰

**VPN Tunneling Lab**

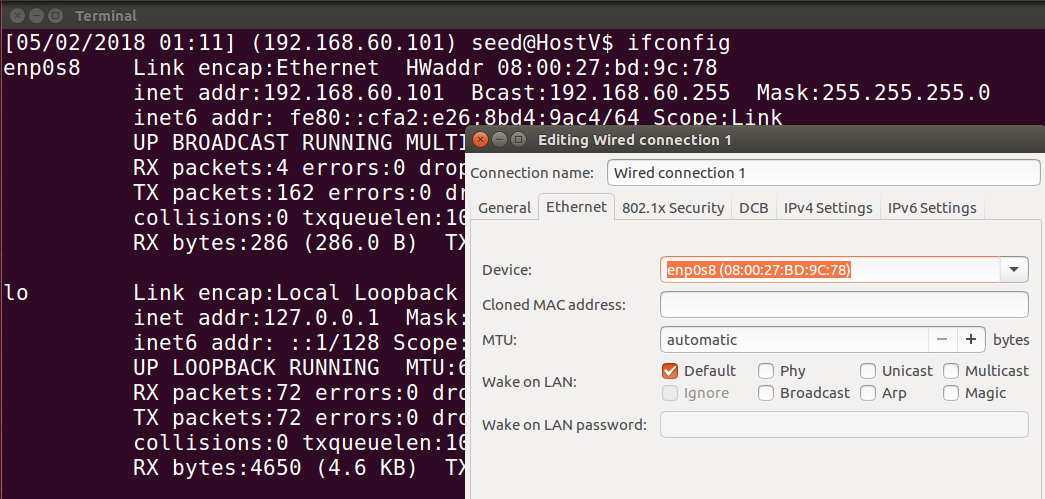
# Task 1: Network Setup

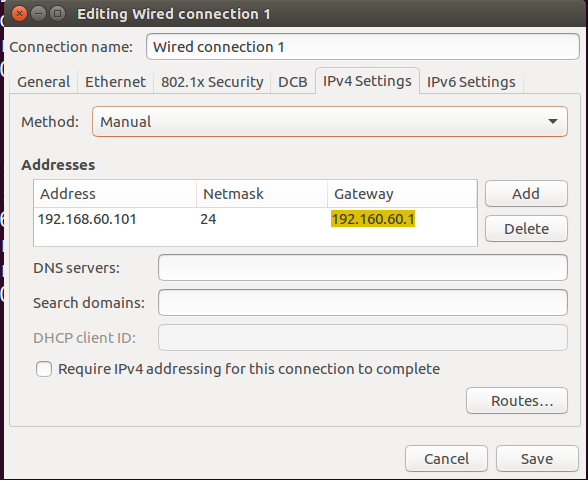
VPN Server: 使用192.168.60.1/24作为VPN网段





Host V:设置IP地址为192.168.60.101，网关采用VPN服务器IP 192.168.60.1





用ping命令测试，测试结果：

|  |  |  |  |
| --- | --- | --- | --- |
| ping | **U** | **VPN** | **V** |
| **U** | \ | 成功 | 失败 |
| **VPN** | 成功 | \ | 成功 |
| **V** | 失败 | 成功 | \ |

# Task 2: Create and Configure TUN Interface

* **Task2.a**

在用户主机U上运行tun.py：

#!/usr/bin/python3

inport fcntl

irport struct

iaport os

iport time

from scapy.all inport\*

TUNSETIFF =ex400454ca

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'seu',IFF\_TUN IFF\_NO\_PI)

ifname\_bytes = fcntl.ioctl(tun,ruNSETIFF, ifr)

#Get the interface nane

ifname= ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifnane)

while True:

time.sleep(10)

运行成功后，创建了一个TUN虚拟接口seu，通过ip address可以查看

* **Task2.b**

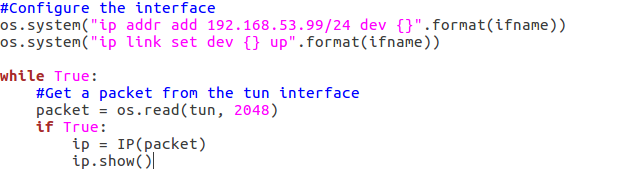
然后，为接口分配IP地址：

sudo ip addr add 192.168.43.159/24 dev seu

sudo ip link set dev seu up

* **Task2.c**

读取数据



运行以上的循环代码，通过ping网内地址进行测试

**1. On Host U, ping a host in the 192.168.53.0/24 network. What are printed out by the tun.py program? What has happened? Why?**

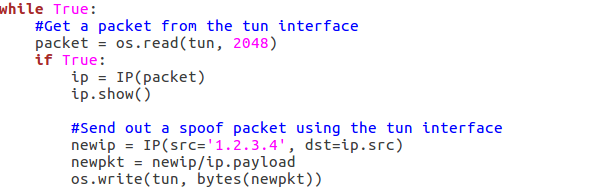
tun.py输出了icmp报文，因为192.168.53.0/24在tuntap设备的网段中。

**2. On Host U, ping a host in the internal network 192.168.60.0/24, Does tun.py print out anything? Why?**

无输出，因为192.168.60.0/24不在tuntap设备的网段中

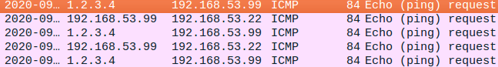
* **Task2.d**

写入数据



运行以上程序，打开wireshark观察

**1. After getting a packet from the TUN interface, send out a new packet to the TUN interface. Please use Wireshark to provide proofs that such packet is sent out successfully.**



两地址之间互相都有请求报文

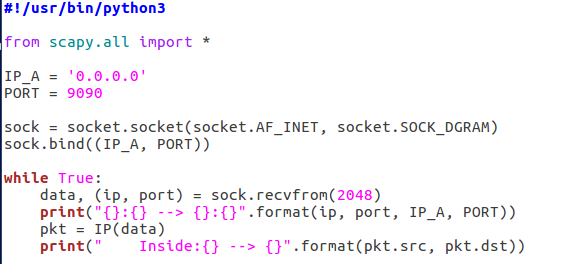
**2. Instead of writing an IP packet to the interface, write some arbitrary data to the interface, and report your observation.**



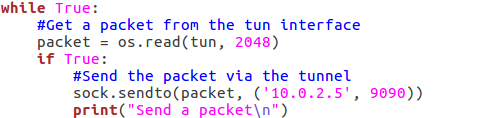
会出现无法识别的报文

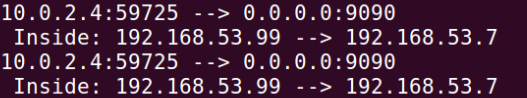
# Task 3: Send the IP Packet to VPN Server Through a Tunnel

运行server.py程序，监听9090端口：

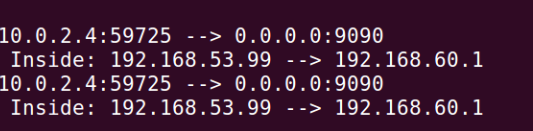


然后在主机U上执行client.py程序：



1. Run the tun server.py program on VPN Server, and then run tun client.py on Host U. To test whether the tunnel works or not, ping any IP address belonging to the 192.168.53.0/24 network. What is printed out on VPN Server? Why?

2. Please provide proofs to demonstrate that when you ping an IP address in the 192.168.60.0/24 network, the ICMP packets are received by tun server.py through the tunnel.



# Task 4: Set Up the VPN Server

IP\_A = ‘0.0.0.0’

PORT = 9090

tun= os.open(" /dew/net/tun" , os.o\_RDWR）

ifr =struct.pack('16sH',bserve’,IFF\_TUN\_IIFF\_NO\_PI)

ifname\_bytes= fcntl.ioctl(tun,TUNSETIFF. Ifr)

ifnane=ifname\_bytes.decode( ‘UTF-8’ )[:16].strip("\x00")

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

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

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

sock =socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.bind((IP\_A, PORT))

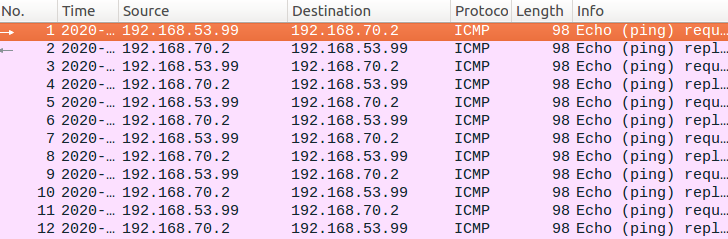
while True:

data, ip. port-sock.recvrror( 2048)

os.write(tun, data)

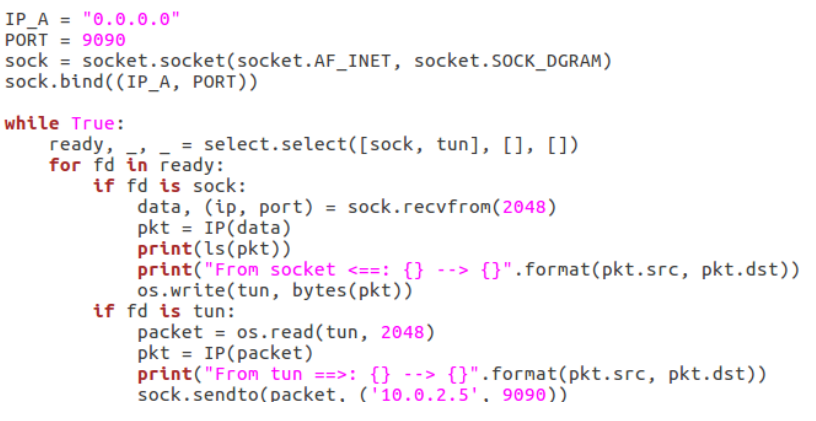
print("Receive a packet\n”)

运行程序，观察wireshark：

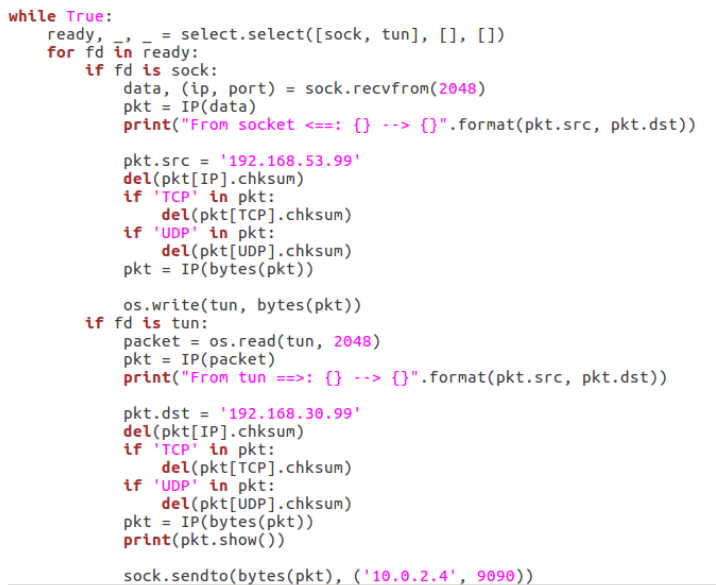


# Task 5: Handling Traffic in Both Directions

client程序的修改：



server程序的修改：



运行测试，此时能够ping通内网，服务器端可以看到通信的交互过程，客户端也能看到类似的报告。

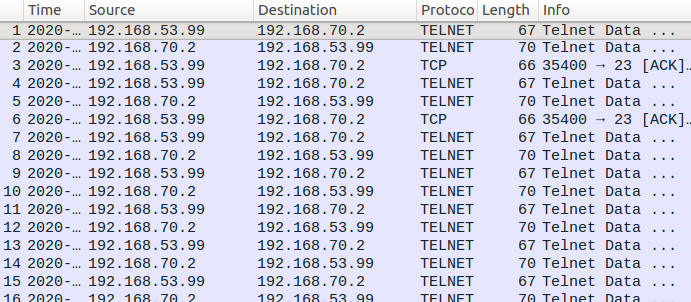
# Task 6: Tunnel-Breaking Experiment

**Once the tunnel is re‐established, what is going to happen to the telnet connection? Please describe and explain your observations.**

打开隧道时进行telnet通信测试，可以正常通信；关闭隧道后进行telent通信，不会有报文传输；但若在短时间内打开隧道并重新连接，会恢复报文。说明隧道的短暂断开没有导致telnet连接也断开。

# Task 7: Routing Experiment on Host V

在主机V上配置路由，然后打开隧道并令U对V进行telnet通信，成功。



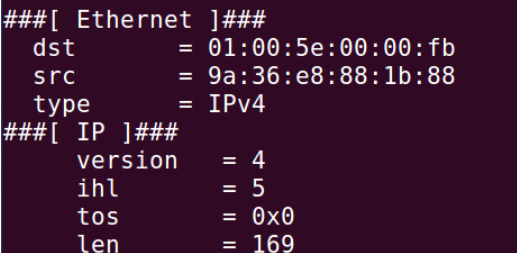
# Task 8: Experiment with the TUN IP Address

修改client程序的IP地址，使其与server端的TUN接口IP地址不在同一网段内，然后进行测试。通过wireshark可以发现因为报文中的src ip与tun不在同一网段，当报文从U抵达VPN服务器后，VPN服务器未能通过TUN将报文发送出去。

解决方法：在服务器端将U的报文写入TUN前，将其源IP修改为与VPN服务器的TUN同一网段的任何IP；在VPN服务器将V的报文写入SOCK前，将其目的IP还原为U的报文中的源IP。

# Task 9: Experiment with the TAP Interface

修改代码然后进行测试



可以看到TAP口确实收到了一个ARP请求