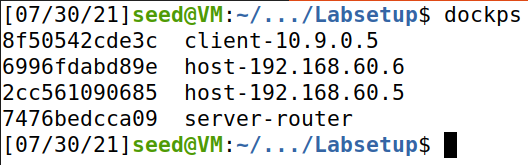
**VPN Lab: The Container Version**

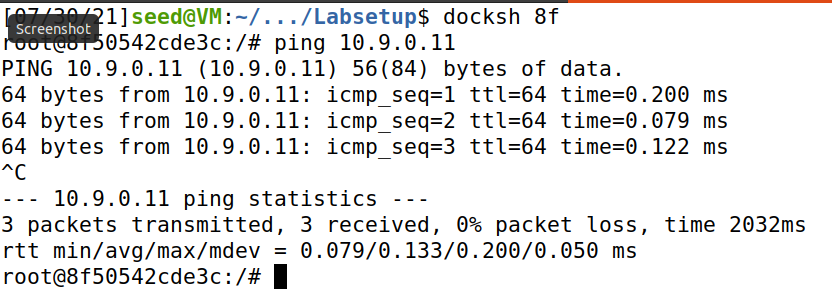
**57118113 蔡义涵**

**Task 1: Network Setup**

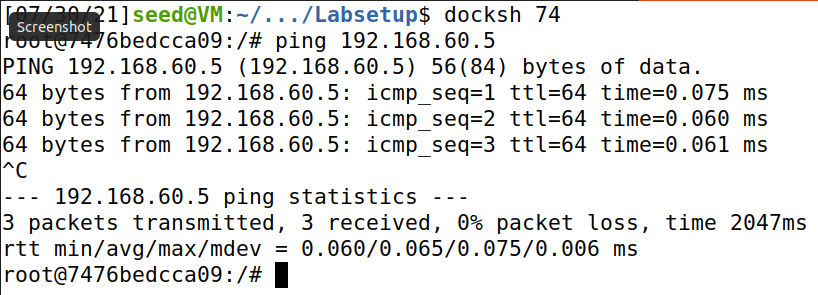
建立docker环境后，直接对docker进行测试



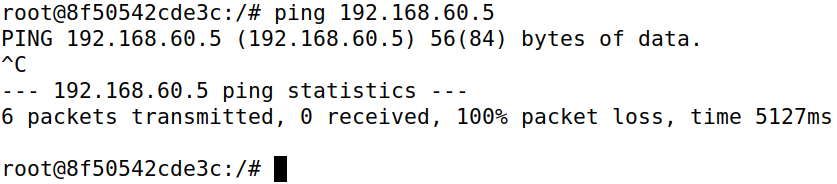
**Host U can communicate with VPN Server**



**VPN Server can communicate with Host V**

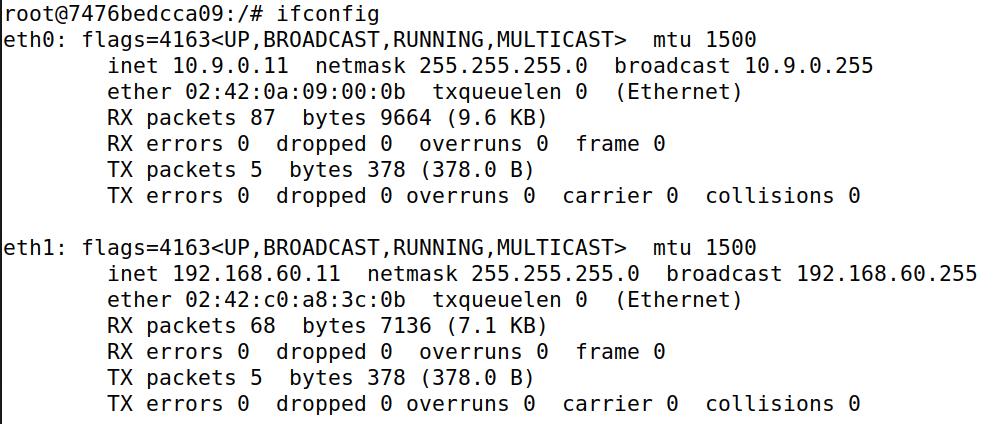


**Host U should not be able to communicate with Host V**

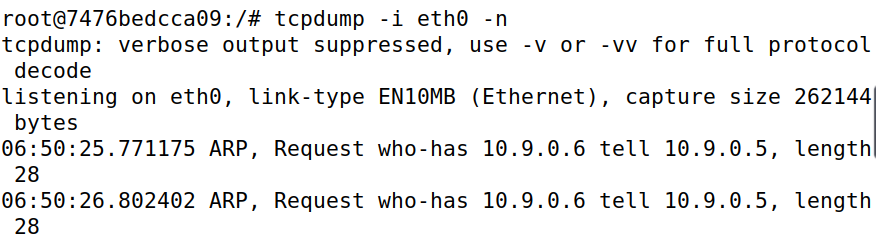


**Run tcpdump on the router, and sniff the traffic on each of the network.**

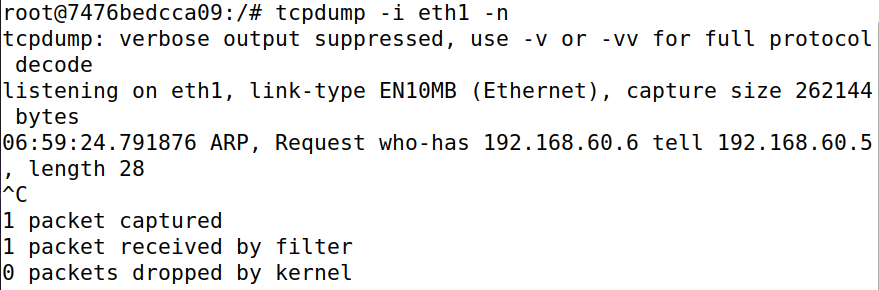
查看router的端口信息：



10.9.0.0/24子网：



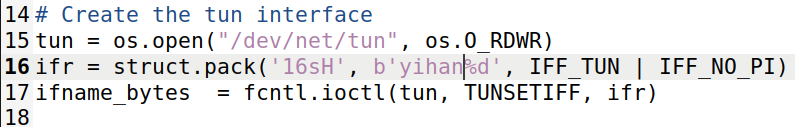
192.168.60.0/24子网：



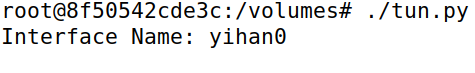
**Task 2: Create and Configure TUN Interface**

**Task 2.a: Name of the Interface**

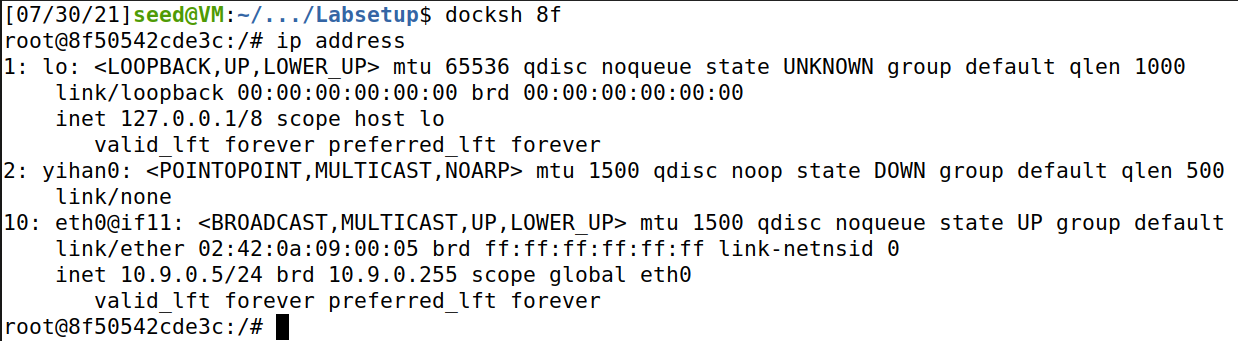
修改tun.py中的接口名称为’yihan’，并将其变为可执行文件。



在Host.U上执行tun.py：



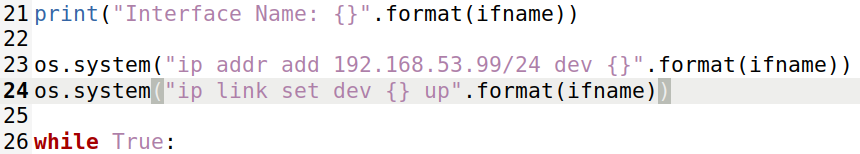
开启另一个终端，查看Host.U的端口信息：



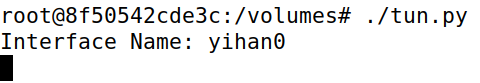
可以看到TUN端口成功添加。

**Task 2.b: Set up the TUN Interface**

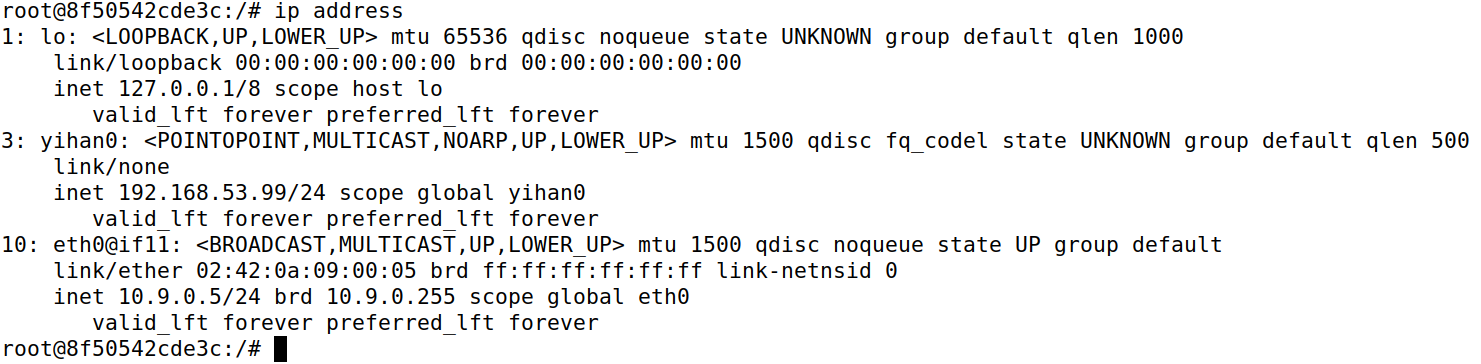
在tun.py中增加以下代码：



重新在Host.U上执行tun.py：



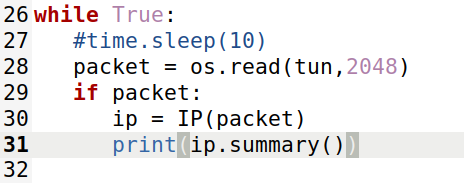
查看Host.U的ip address：



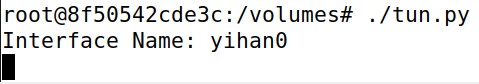
发现接口已经具备了ip地址，且不处于DOWN的状态。

**Task 2.c: Read from the TUN Interface**

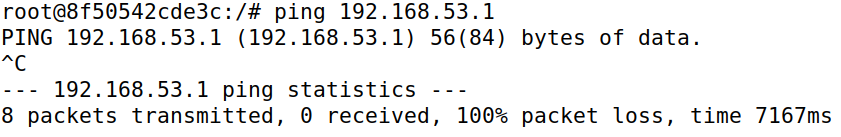
修改tun.py的while循环：

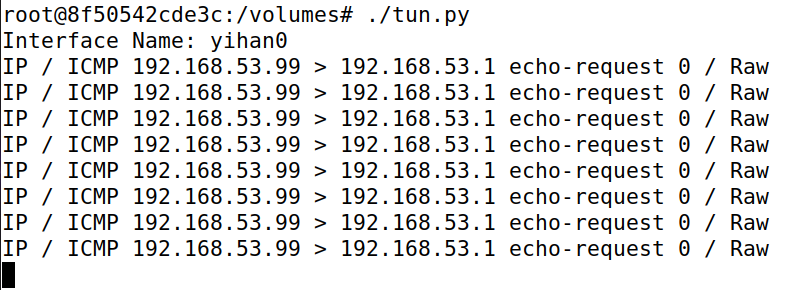


重新在Host.U上执行tun.py：



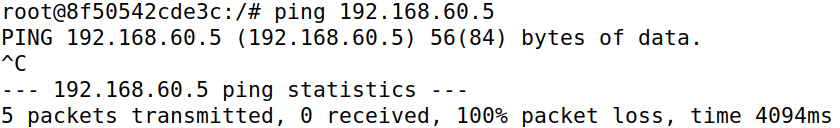
**On Host U, ping a host in the 192.168.53.0/24 network**

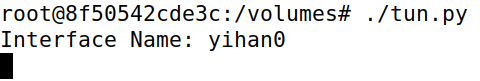




可以接收到相应的报文。

**On Host U, ping a host in the internal network 192.168.60.0/24**

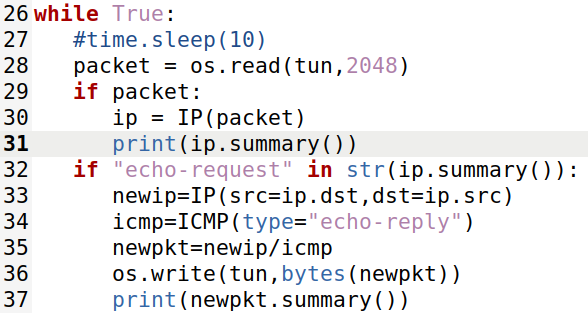




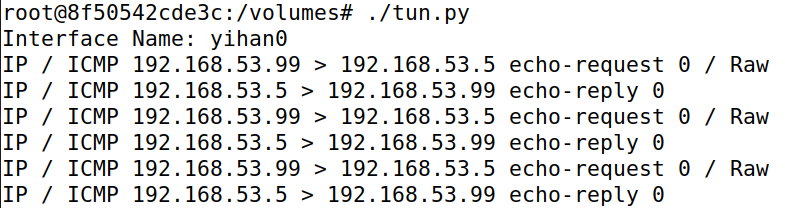
可以看到不能接收到相应的报文。

**Task 2.d: Write to the TUN Interface**

修改tun.py程序，使其在TUN接口获取到数据包后，能够构造出新的数据包：

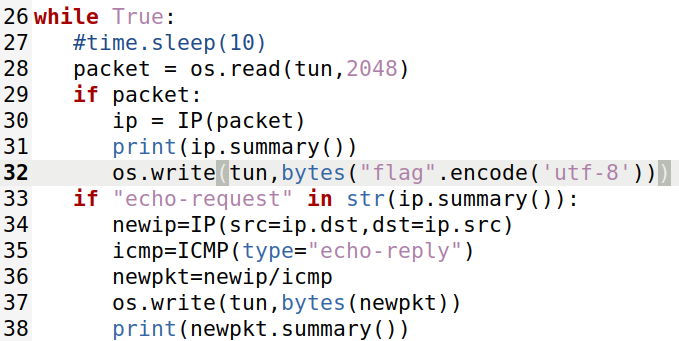


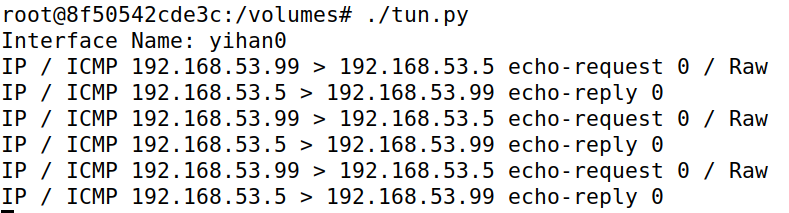
重新在Host.U上执行tun.py，同时ping192.168.53.5：



发现有报文构造的结果。

在tun.py中随便写一些字符串输出：

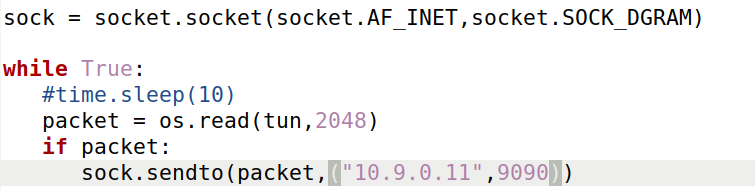




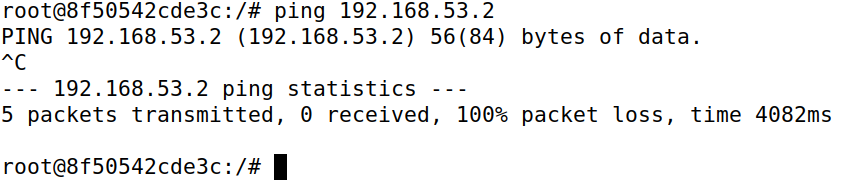
可以看到字符串输出成功。

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

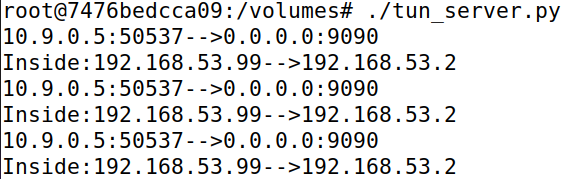
完成tun\_server.py和tun\_client.py，其中tun\_client.py部分代码为：



在server上运行tun\_server.py程序，在Host.U上运行tun\_client.py，然后在Host.U上ping192.168.53.0/24网段上的ip：



Server收到了相应的报文：



若在Host.U上ping 192.168.60.0/24网段：

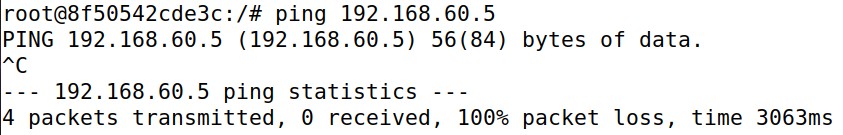


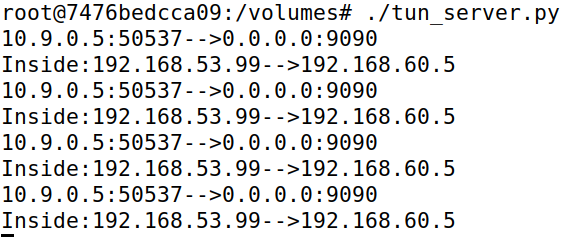
Server没有收到报文。

在Host.U上增加一条路由：



然后再执行ping命令：

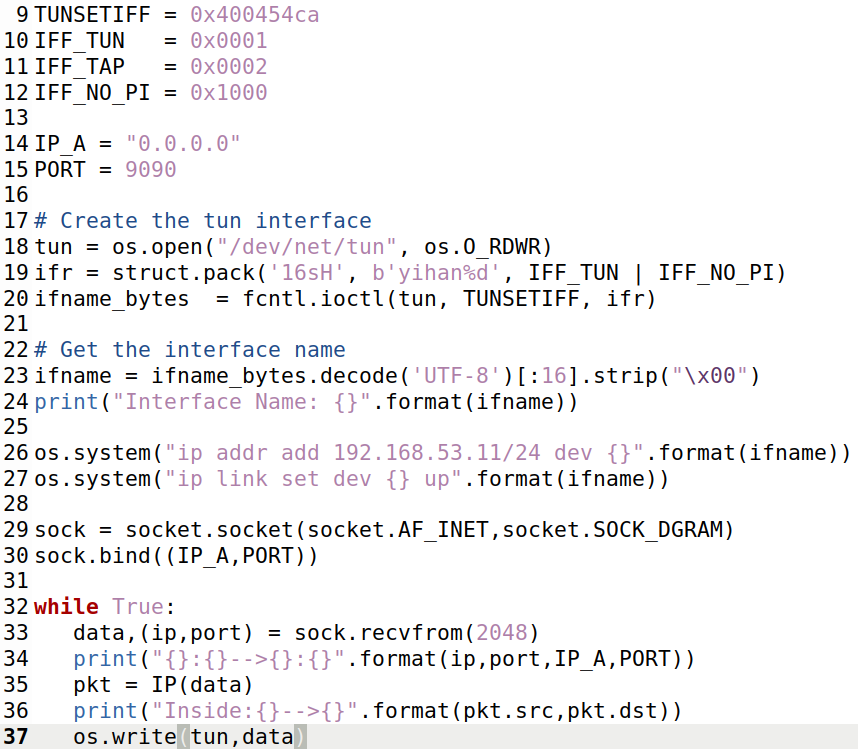




可以看到报文被server接收。

**Task 4: Set Up the VPN Server**

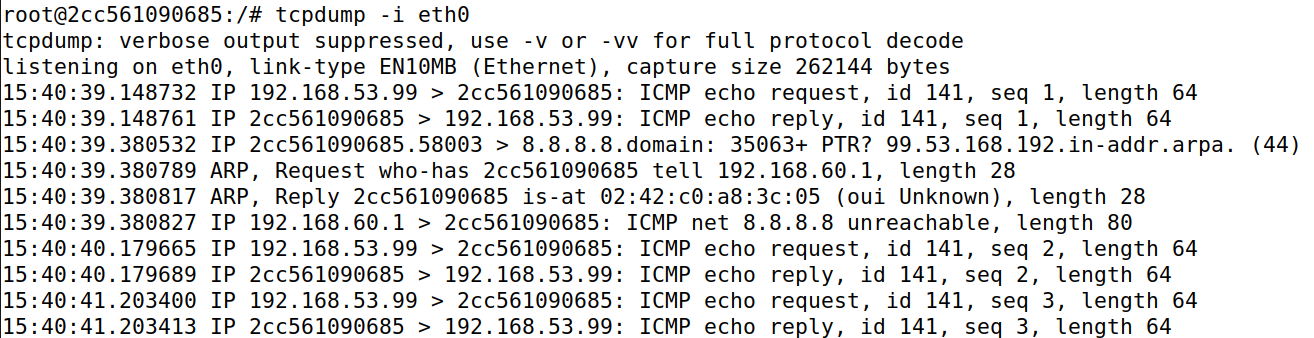
修改tun\_server.py：



检查router的net.ipv4.ip\_forward=1：

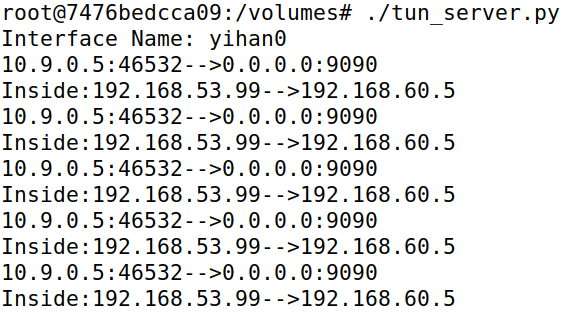


接下来在Host.U上执行tun\_client.py，在server/router上执行tun\_server.py。使用Host.U去ping Host.V，同时在Host.V上执行tcpdump命令：



可以看到Host.V接收了相应的ICMP request，并发出了reply。

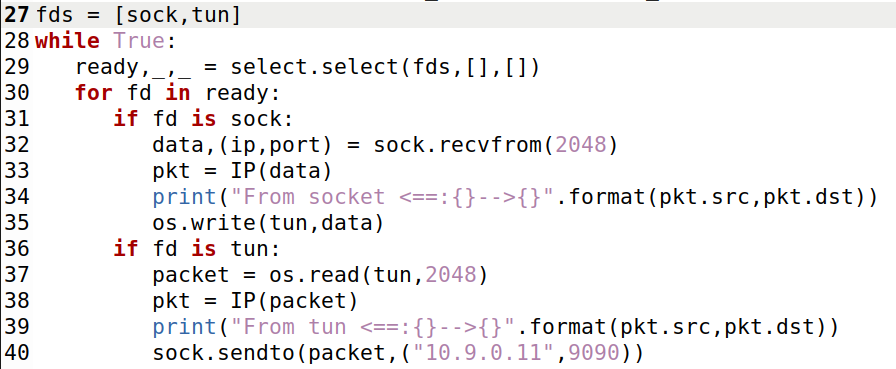
Server上也可以看到报文的一些信息：



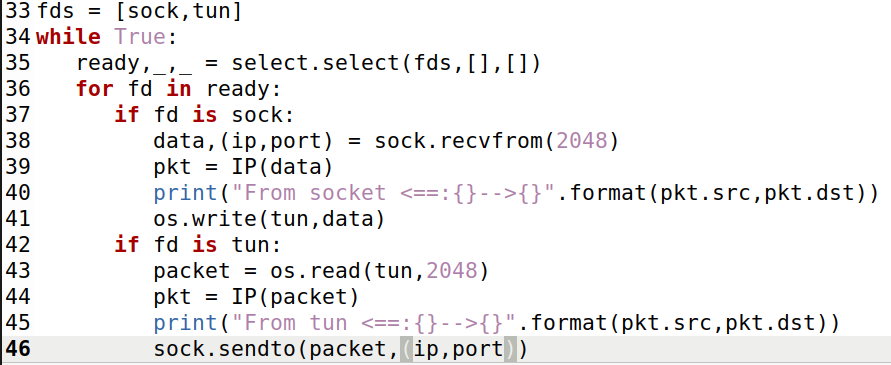
**Task 5: Handling Traffic in Both Directions**

分别修改两个程序为：

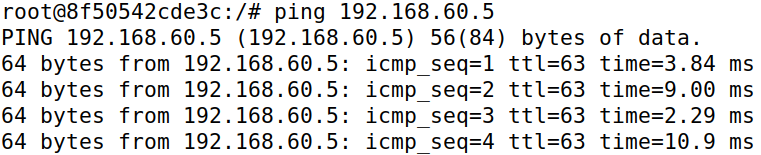
**tun\_client.py**



**tun\_server.py**

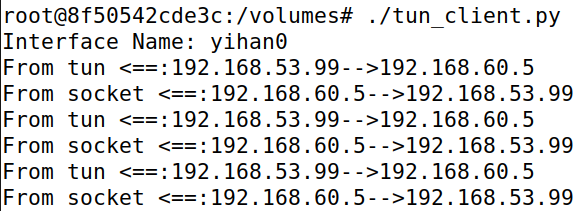


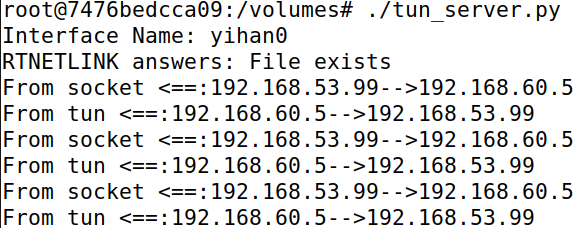
在Host.U上ping Host.V：



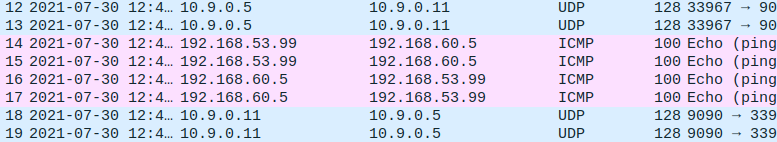
可以看到已经成功。

查看以上两个程序的输出信息：



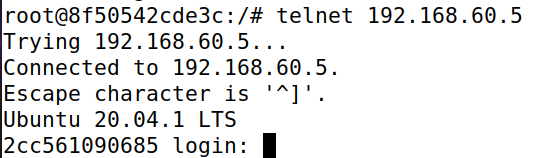


查看wireshark的抓包结果：



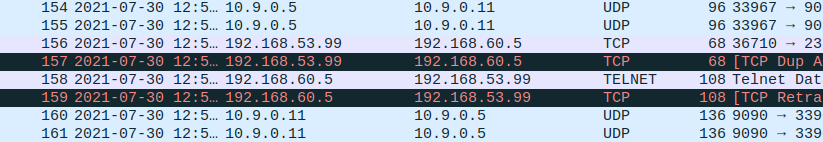
可以看到报文传输路径为：先是10.9.0.5传输给10.9.0.11，然后进入VPN，为192.168.53.99发往192.168.60.5，然后原路返回。

在Host.U上telnet Host.V：



同样成功。

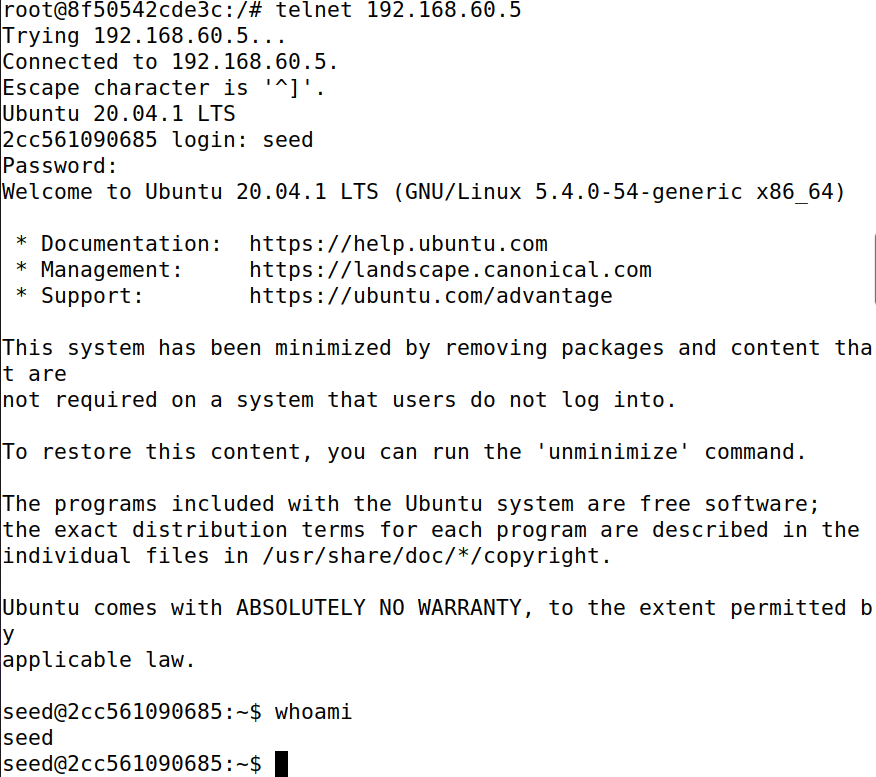
查看wireshark：



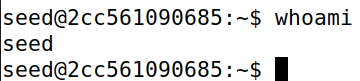
同样可以看到VPN下ip地址变换的过程，由10.9.0.0/24网段内通信变为192.168.53.99发往192.168.60.5，返回过程同理。

**Task 6: Tunnel-Breaking Experiment**

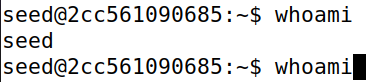
在vpn成功建立的前提下，建立Host.U与Host.V之间的telnet连接：



中断tun\_client.py程序后：



键入字母无法显示，重新开启tun\_client.py后：



发现键入的字母又显示出来了。

原因是程序是从tun接口上发送报文的，程序终止时报文会cunchuzai 这些接口的buffer上，等待程序运行时再处理。