**Lab7 reporter**

**VPN Tunneling Lab**

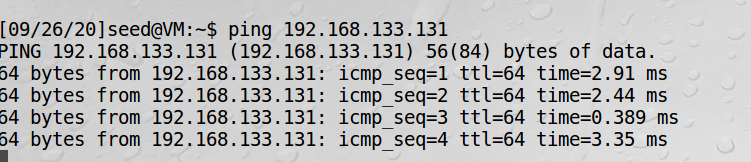
57117107 赵笛言

Set：用户U：192.168.133.130；用户V：192.168.18.131；

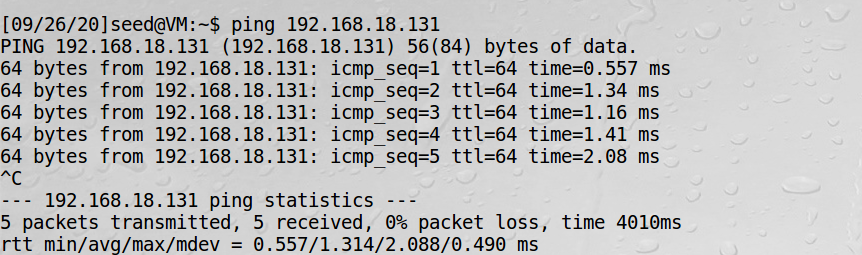
VPN server：192.168.18.129；192.168.133.131

**Task 1: Network Setup**

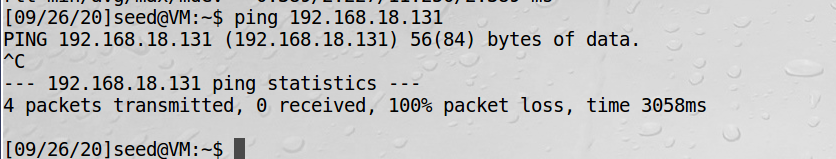
• Host U can communicate with VPN Server.



• VPN Server can communicate with Host V.



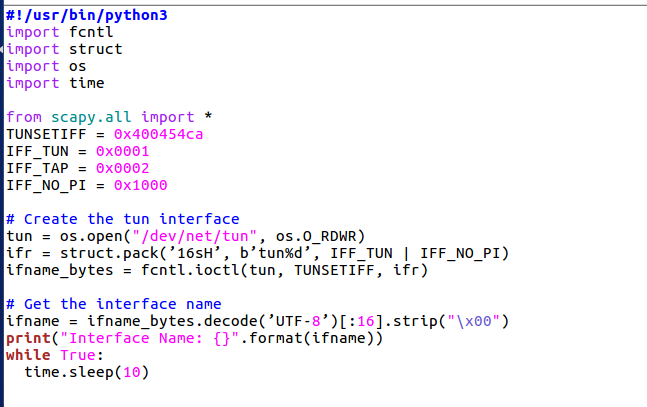
• Host U should not be able to communicate with Host V.

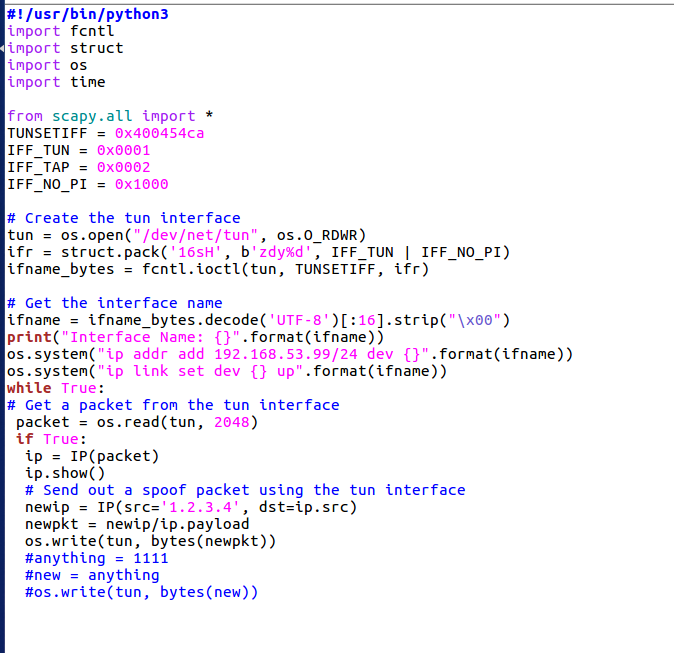


配置完成。

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

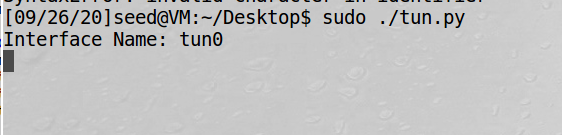
在U上运行tun.py程序添加一个TUN虚拟接口tun0：（并尝试改名）

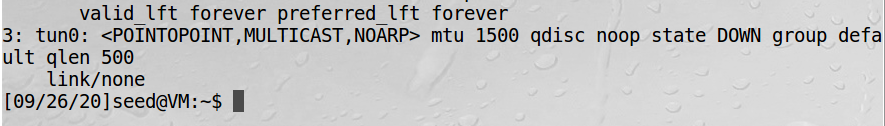




**Task 2.a: Name of the Interface**

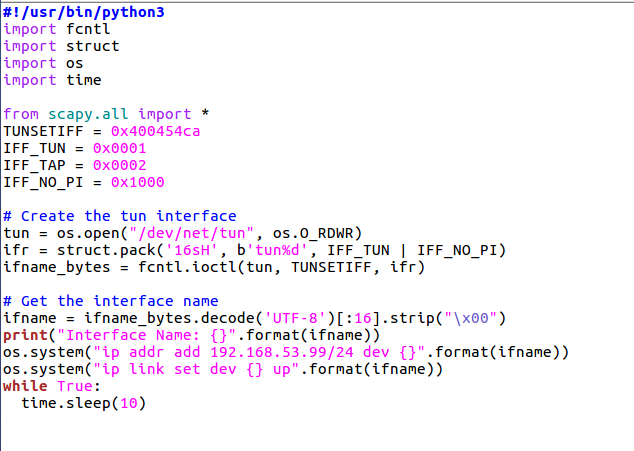
改变tun.py并使用ip address查看：

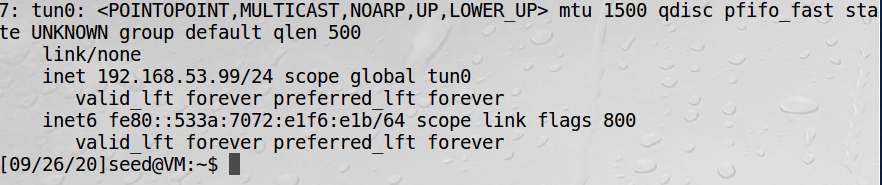




**Task 2.b: Set up the TUN Interface**

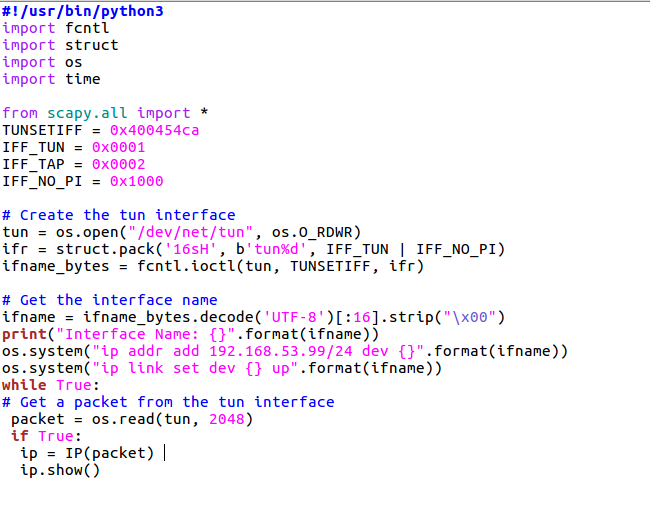
给tun0接口分配IP地址，查看：

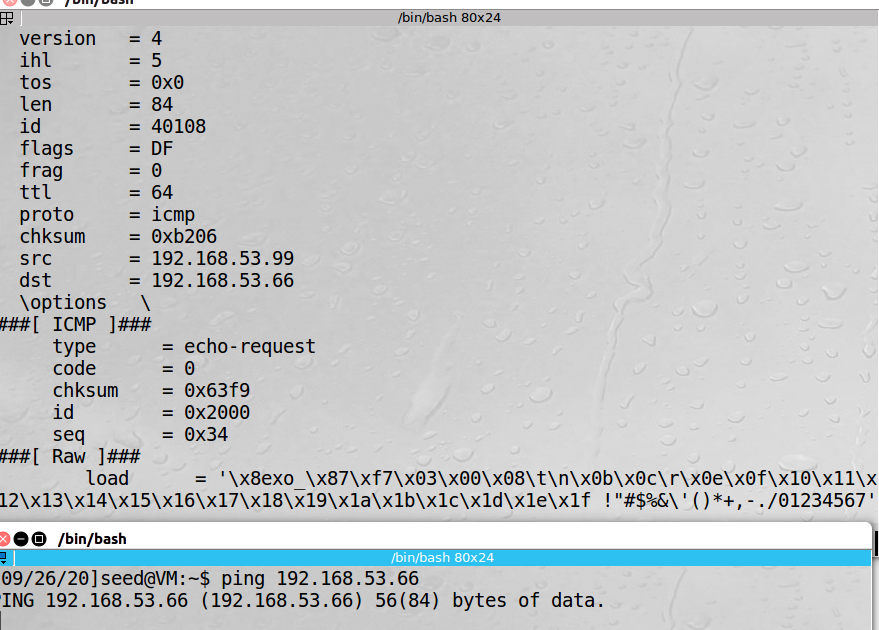




**Task 2.c: Read from the TUN Interface**

添加代码，可监听报文：



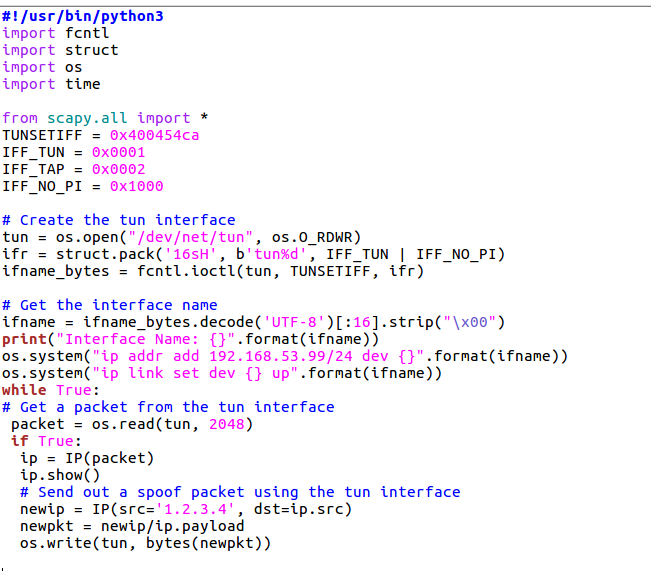


tun.py程序打印出了发往192.168.53.22的数据报，这些报文在路由选择时被发给了tun0虚拟接口。

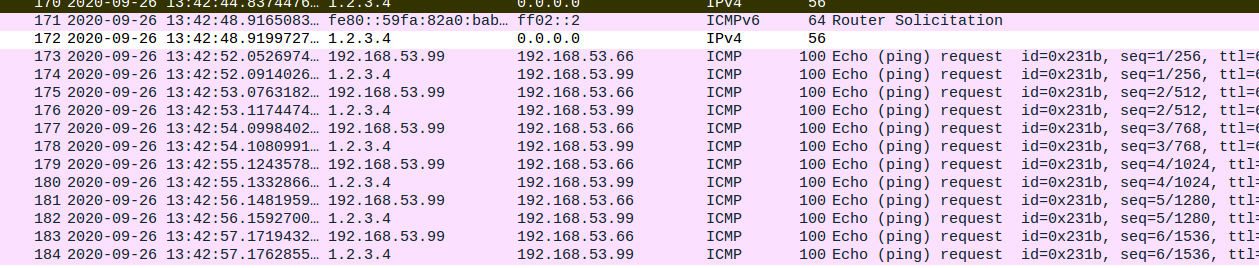
而ping 192.168.60.66进行测试，无结果，此时192.168.60.0/24这一网络还不在tun0的可监听范围中。

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

添加代码：

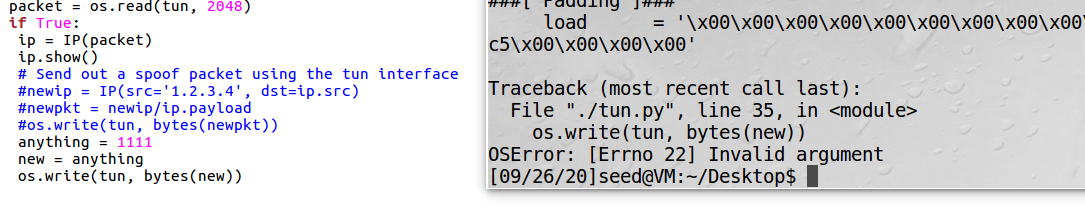


运行程序和wireshark：



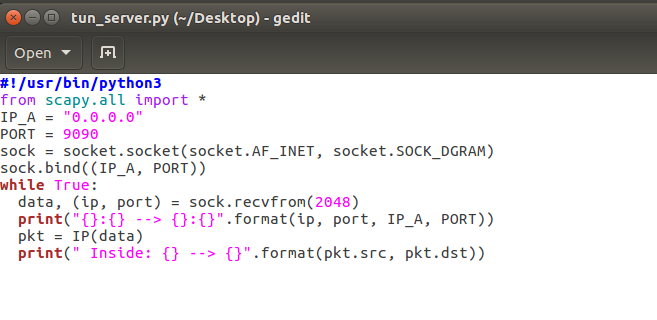
可以看到从192.168.53.99发往192.168.53.66的ICMP请求报文和从1.2.3.4发往192.168.53.99的ICMP请求报文。

而向tun中写入任意数据，产生"Invalid argument"的错误：



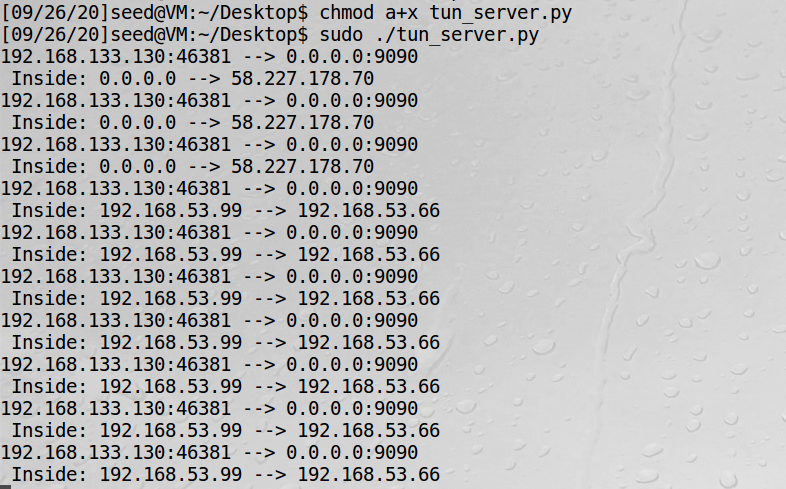
**Task 3: Send the IP Packet to VPN Server Through a Tunnel**

在VPN server上编译tun\_server.py程序：监听9090端口并打印出所有收到的数据



ping 192.168.53.0/24网内任意IP地址（.66）:

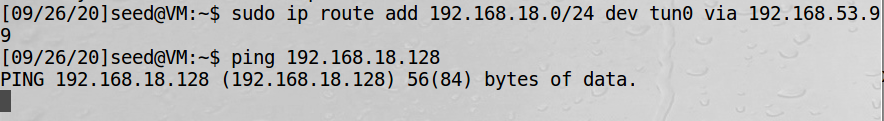
可以看到VPN server输出：



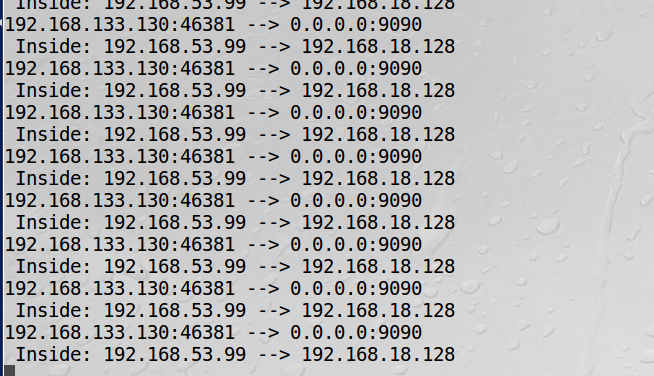
VPN服务器能够收到数据报，外层为192.168.133.130通过46381端口发往自身9090端口的数据报，内层为192.168.53.99发往192.168.53.22的数据报，可见内层数据报通过IP隧道在主机U与VPN服务器间实现了通信。

ping 主机V时则只能看到主机U以0.0.0.0的IP地址发往其他IP地址的数据报。

在主机U上配置路由信息，将192.168.10.0/24网段的出口设为虚拟接口tun0：

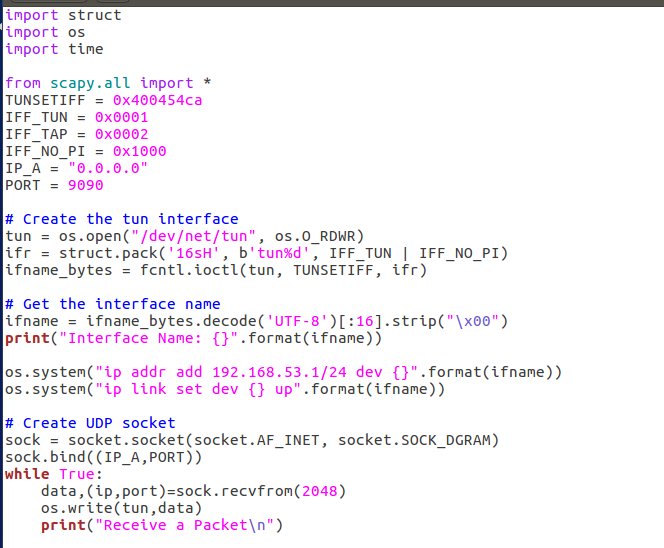


则可以看见发往192.168.18.128的内层数据。



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

编译tun\_server.py：

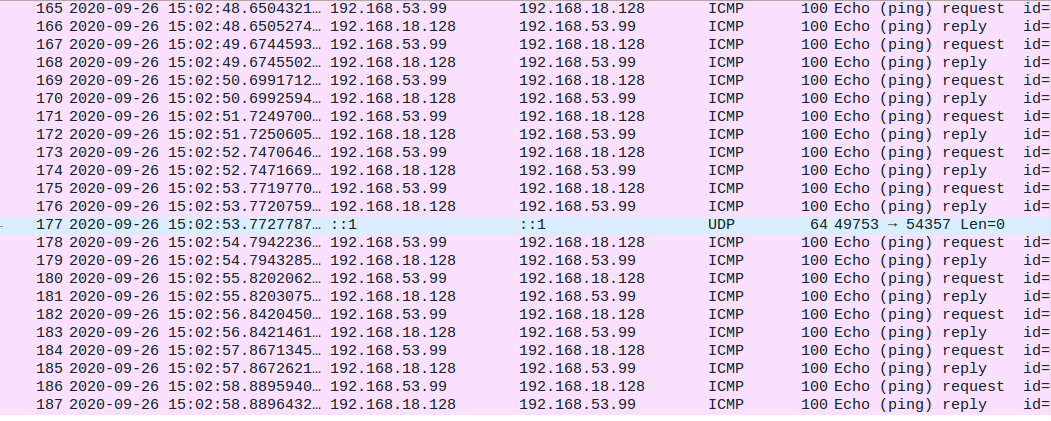


打开路由转发选项（VPN server）：



使用主机U ping V

主机V上打开wireshark：

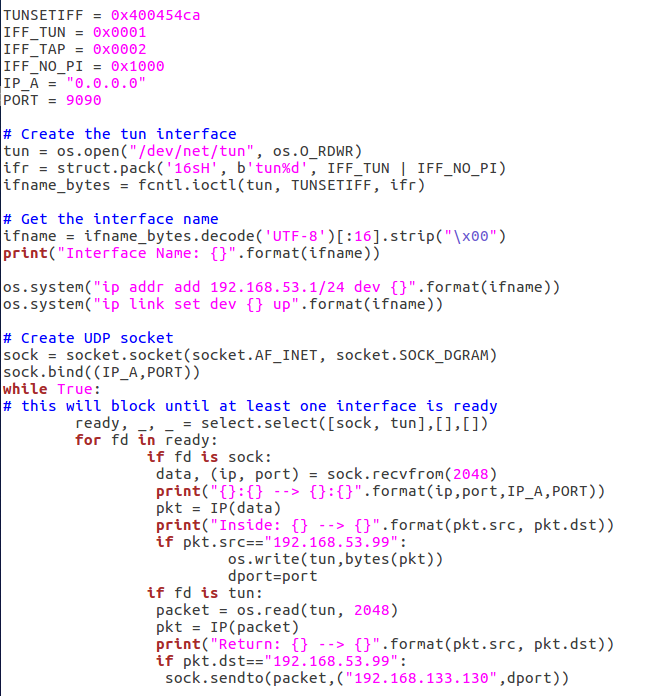


接收到了来自于主机U的ICMP报文，且发送了reply回应。

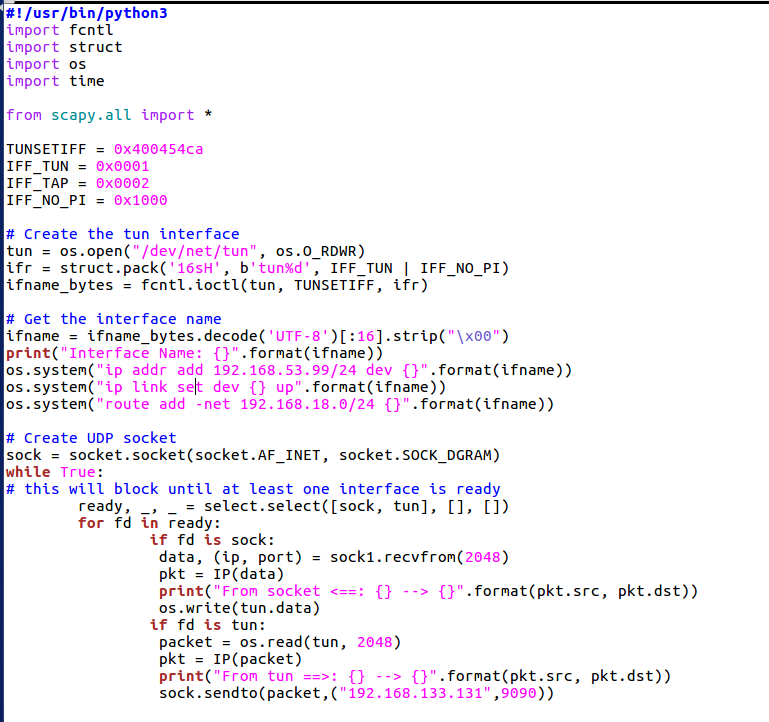
可见已经成功连接，但由于没有设置完毕，返回报文还无法接收。

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

VPN server中编译tun\_server.py：

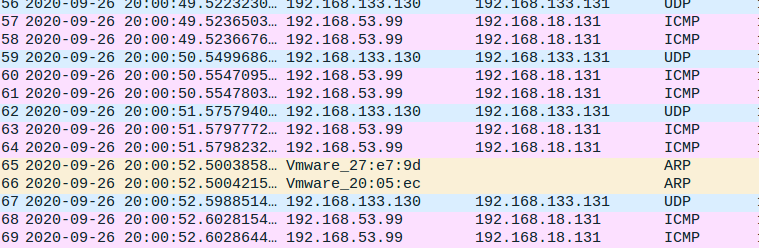


主机U编译tun\_client：



U可以ping通V，

wireshark输出，可见利用了隧道：



Telnet连接也能成功建立。

**Task 6: Tunnel-Breaking Experiment**

建立telent连接后关闭tun\_server.py。

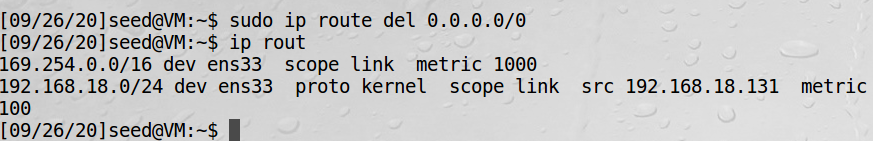
此时无法再输入命令，不再有报文传输。

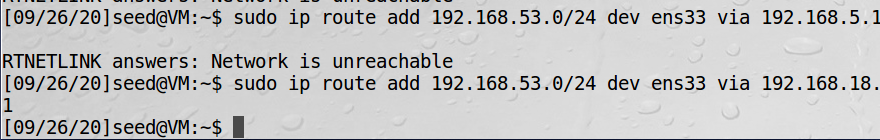
等待一段时间后重新连接，可以正常输入，还会传送之前输入的报文。

当IP隧道断开后，如果能在较短时间内重新连接上，由于报文被暂存在了缓存，可以重新发送。

**Task 7: Routing Experiment on Host V**

清除默认路由并添加路由：

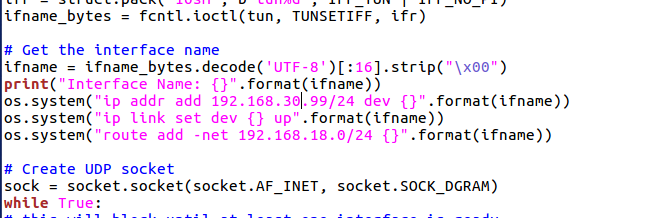




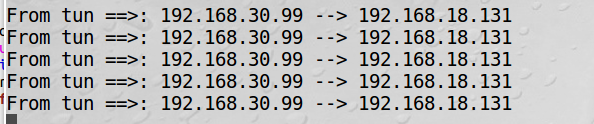
主机U和主机V可以成功通信（ping和telnet均可）。

**Task 8: Experiment with the TUN IP Address**

修改客户端TUN端口IP地址，使其与服务端TUN接口IP不位于同一网段



此时无法ping通。



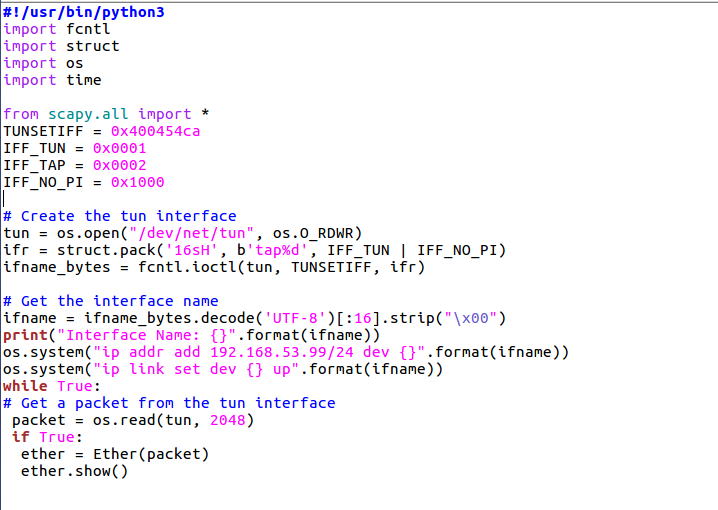
此时观察VPN服务器上wireshark：ens33和tun0端口都分别有收到UDP和ICMP报文，但ens38端口却没有报文发出。

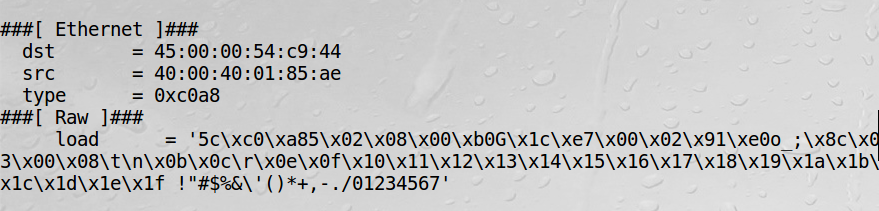
这是由于路由器的反向过滤机制。

而在VPN服务器上添加192.168.30.0/24网络与tun0端口关联的路由表项就可以正常通信。

**Task 9: Experiment with the TAP Interface**

编译基于TAP端口的隧道进行通信：



尝试ping192.168.53.66：  


可以看到ethernet层存在srcMAC地址，看出tap工作在MAC层。