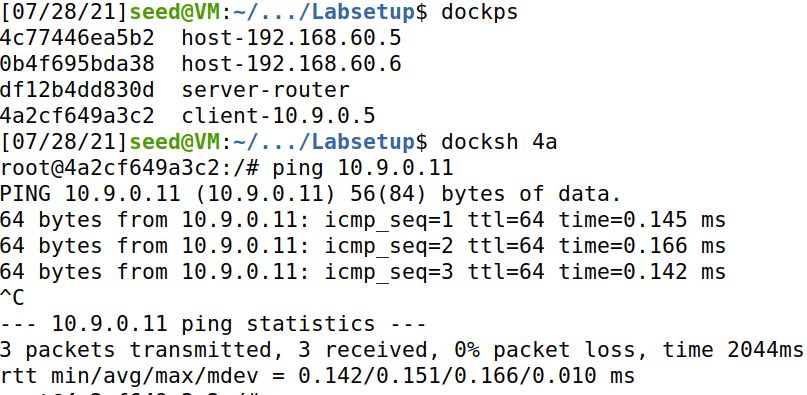
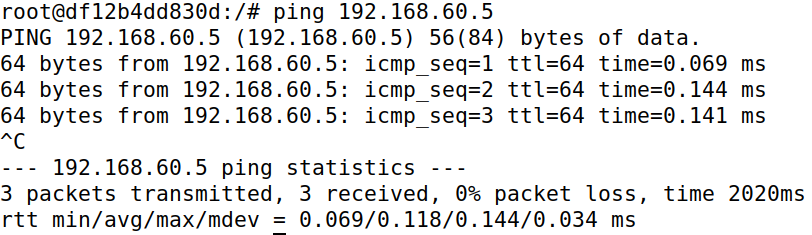
**57118110 杨紫瑄**

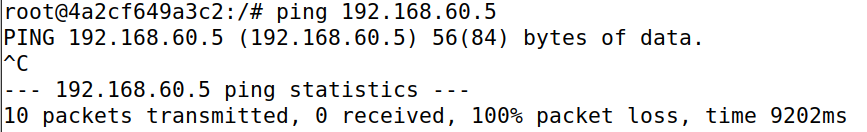
**Task 1 Network Setup**



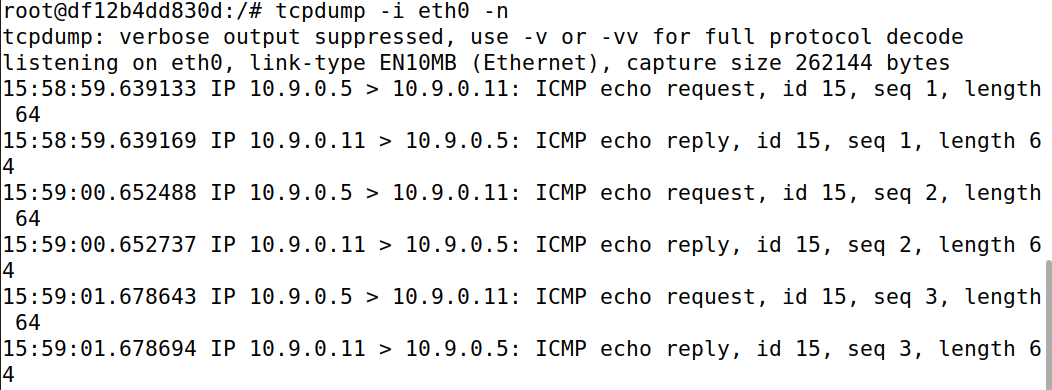
Host U可以与VPN通信



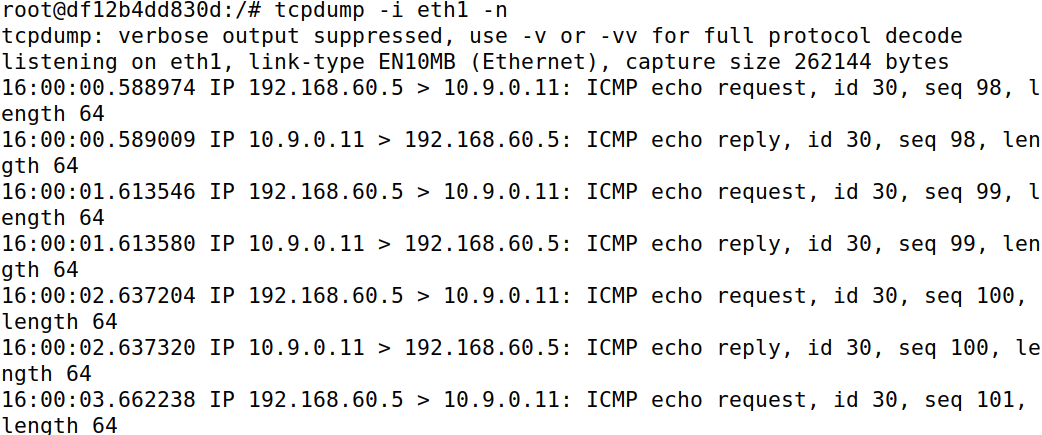
VPN可以与Host V通信



Host U不能和Host V通信



嗅探接口eth0



嗅探接口eth1，两个接口都可以正常嗅探

配置正常

**Task 2 Create and Configure TUN Interface**

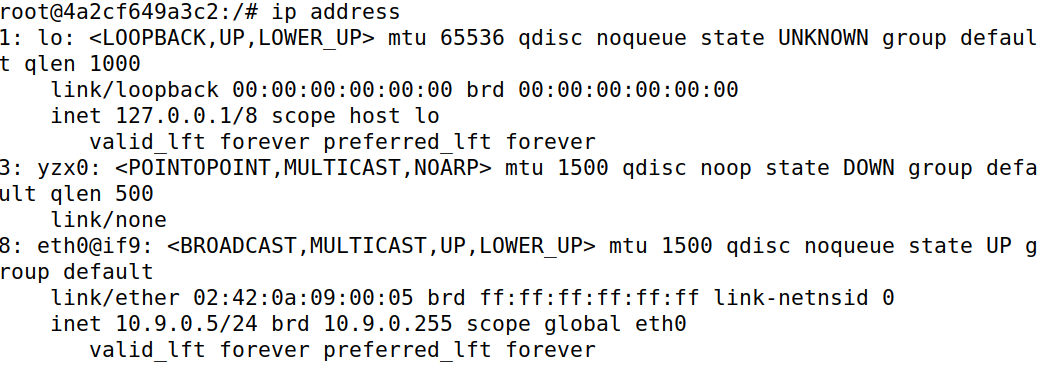
**Task 2.A Name of the Interface**



将文件tun.py中端口名修改为yzx



在Host U中运行程序后发现端口名改为了yzx0

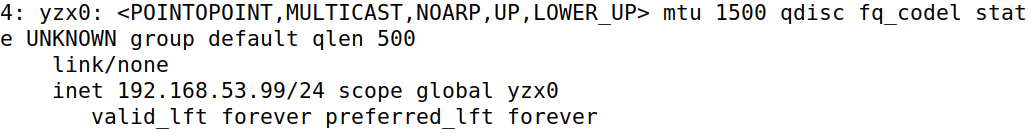


此时新打开一个终端查看发现ip address命令下的端口名也改为了yzx0

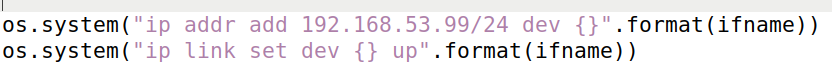
**Task 2.B Set up the TUN Interface**



设置yzx0接口信息，为其安排一个ip地址，并将其设为up

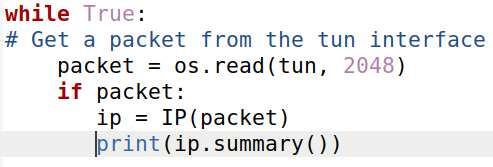


执行ip address命令时可以发现端口已被成功分配了地址

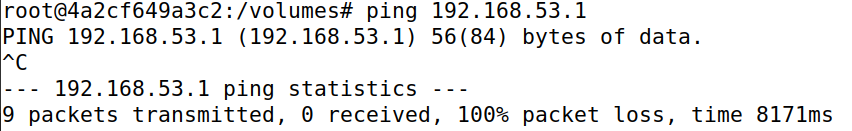


也可以将上图两条命令加入tun.py文件中，能够达到同样的效果

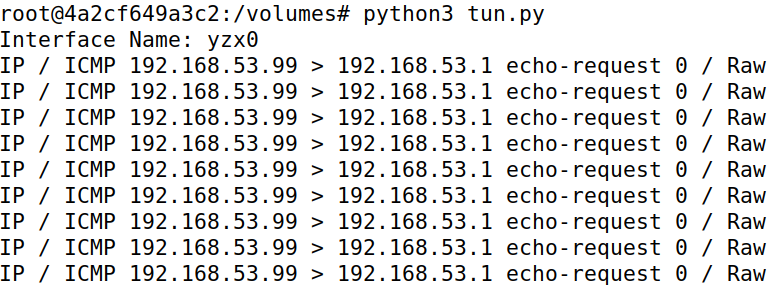
**Task 2.C Read from the TUN Interface**



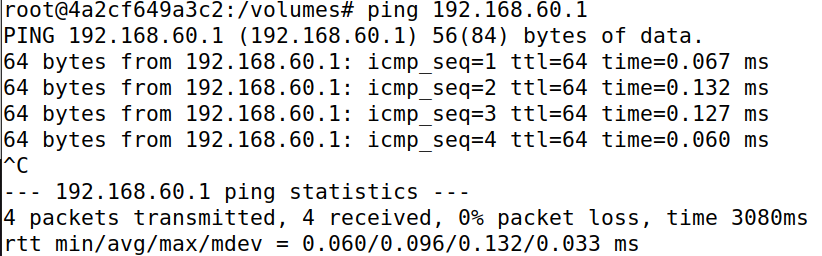
将tun.py文件中的while循环修改如上图



再次执行程序后ping 192.168.53.0/24网段中任一主机

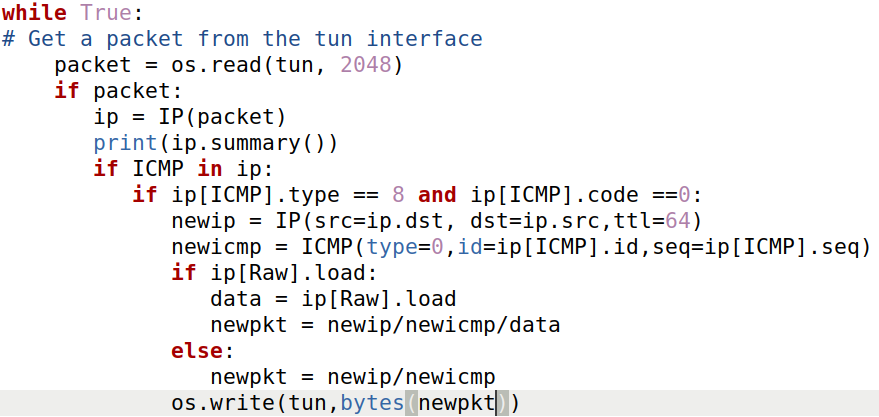


此时ping不通，并在执行程序的命令下打印出如上信息，因为此时的ICMP请求报文都被端口捕获了

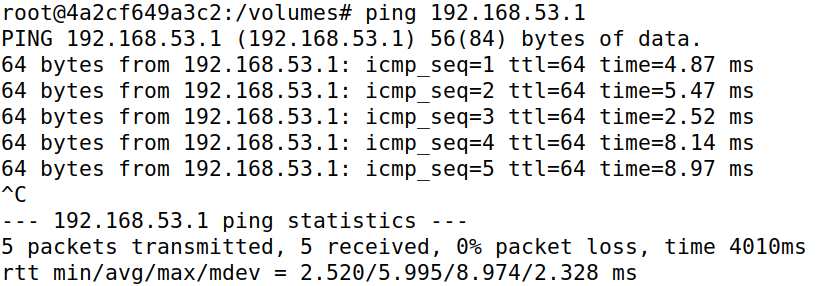


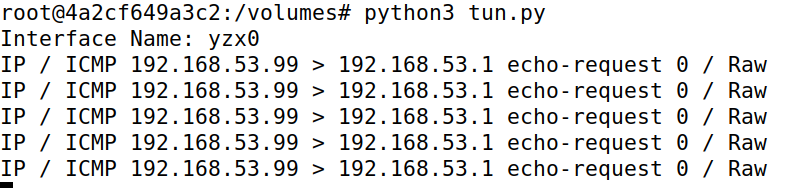
而ping 192.168.60.0/24网段的任一主机能够ping通，此时执行程序的命令下无任何输出，这是因为发送给192.168.60.0/24的报文不经过yzx0端口，所以不会被捕获

**Task 2.D Write to the TUN Interface**



将while循环修改如上图所示，首先判断报文是否为ICMP报文，再判断是否为回送请求报文，若是则将请求报文中源地址与目的地址交换，构造ICMP响应报文，加上原数据包的负载后写入TUN端口

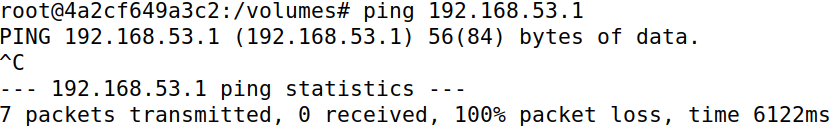


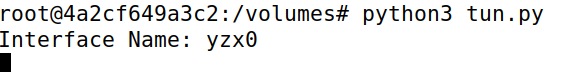


运行程序后再ping192.168.53.1，此时能够ping通说明伪造响应报文成功，运行程序命令下也有捕获的原始报文信息



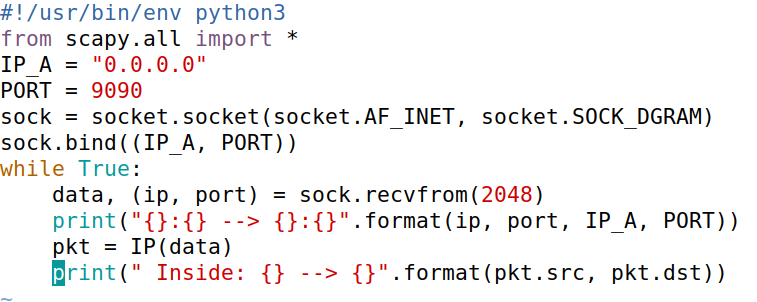
修改while循环，不写入数据包而是任意数据



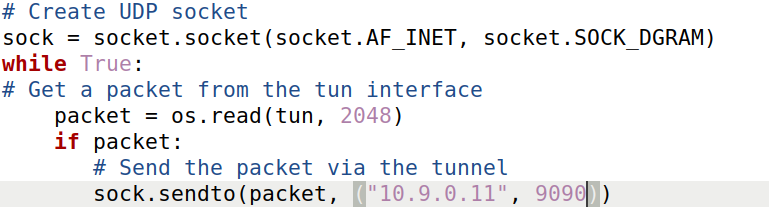


再次运行程序后ping 192.168.53.1发现ping不通,运行程序命令下也没有任何输出

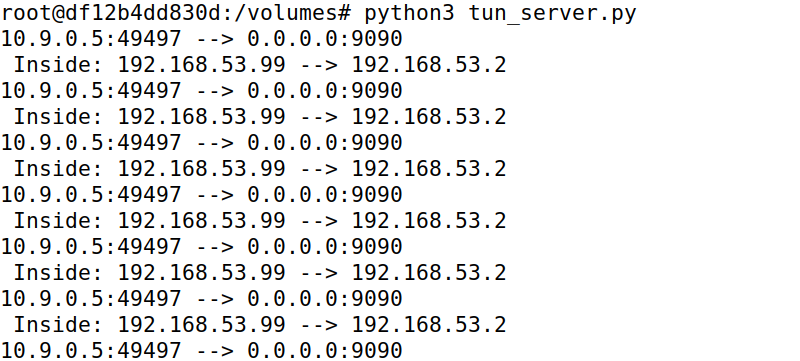
**Task 3 Send the IP Packet to VPN Server Through a Tunnel**



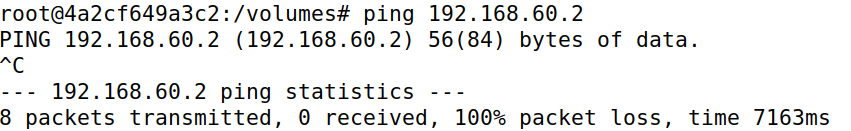
创建tun\_server.py文件，内容如上图所示



将tun.py文件修改为上图并改为tun\_client.py



在VPN上运行tun\_server.py,主机U上运行tun\_client.py，并在主机U上ping 192.168.53.0/24任一主机，此时VPN运行程序命令下输出如上图所示，此时VPN服务器成功捕获了报文

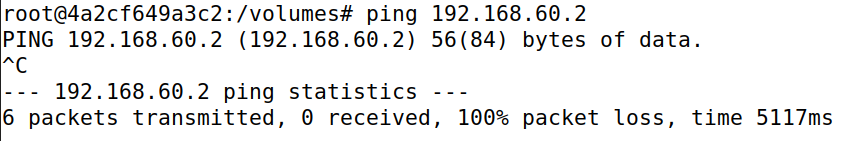


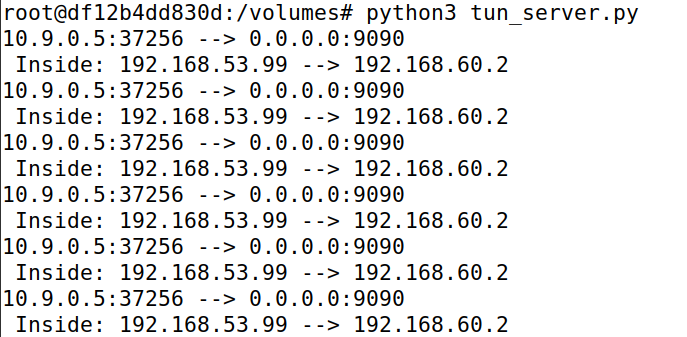


未配置路由时，在主机U上ping 192.168.60.0/24任一主机时，VPN上无任何输出



在tun\_client.py文件中加入上图所示代码来自动配置路由



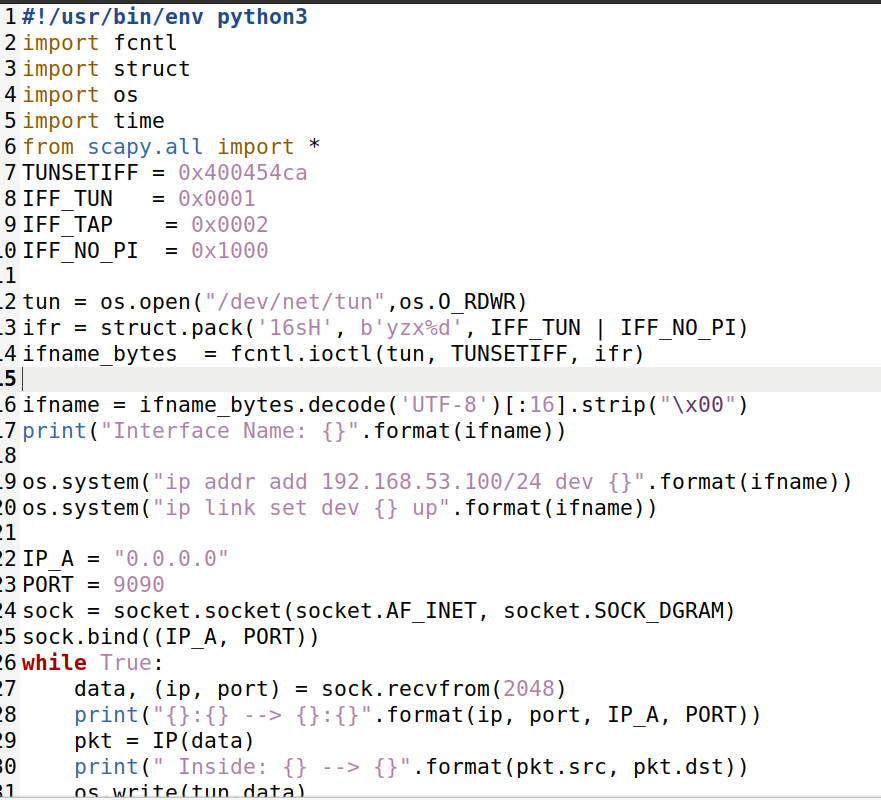


重新执行程序后再次在主机U上ping 192.168.60.2,此时VPN有输出，说明tun\_server通过隧道接收到报文

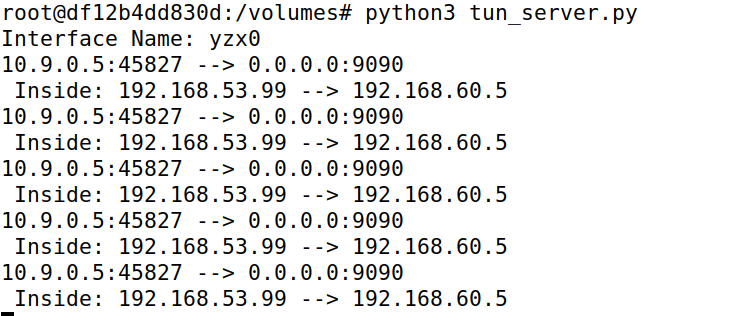
**Task 4 Set Up the VPN Server**



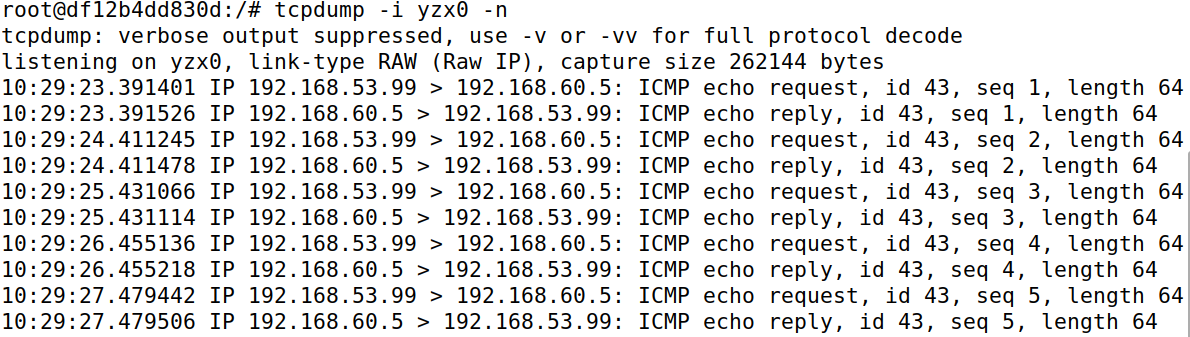
查看文件docker\_compose.yml中发现net.ipv4.ip\_forward已设为1



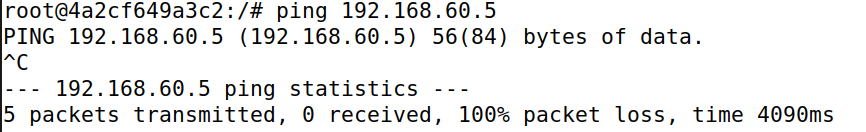
修改tun\_server.py文件如上图所示，建立一个tun接口将数据包路由到最终目的地



重复Task 3中的步骤，在VPN中运行tun\_server.py，在主机U中运行tun\_client.py的同时ping 192.168.60.5输出如上图所示

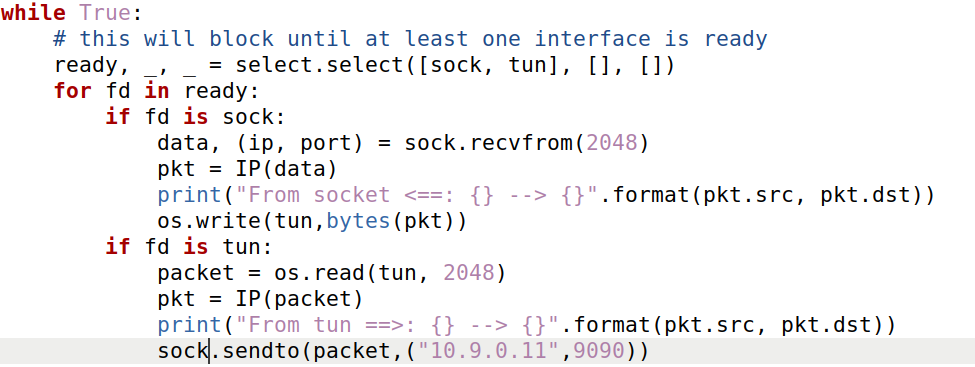


在VPN中嗅探yzx0端口，发现ICMP请求包成功通过隧道到达主机V，并接收到了主机V的响应包

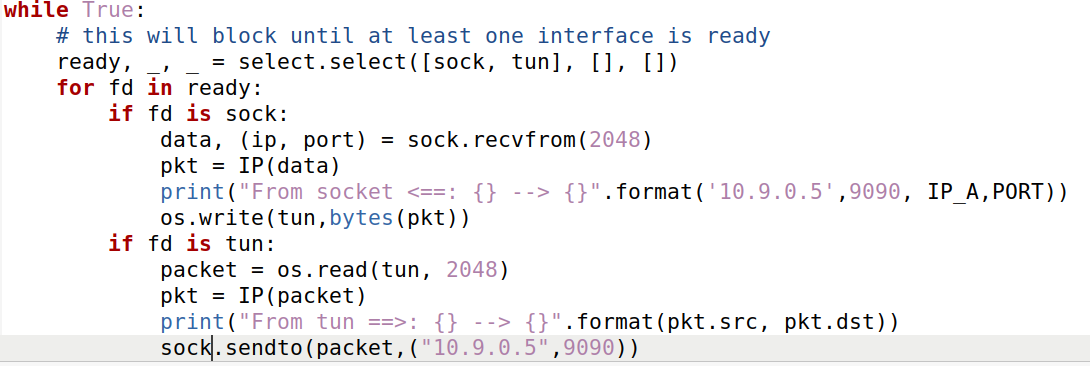


此时ICMP响应包还未返回，所以主机U中接收不到响应

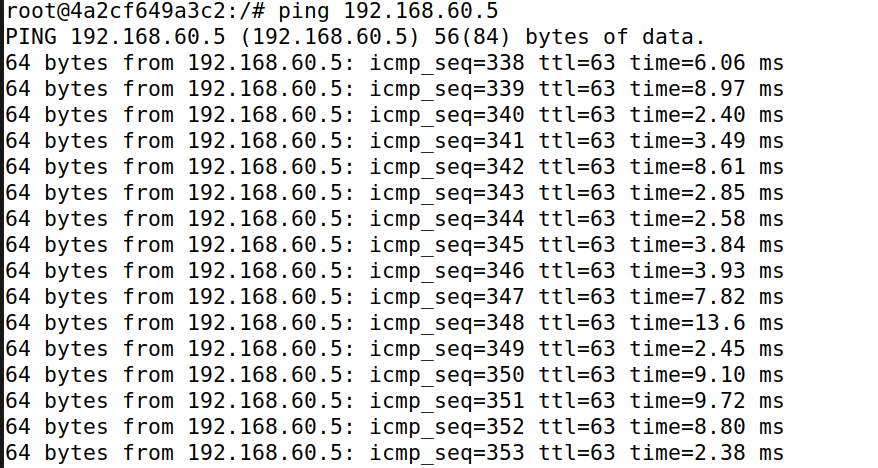
**Task 5 Handling Traffic in Both Directions**



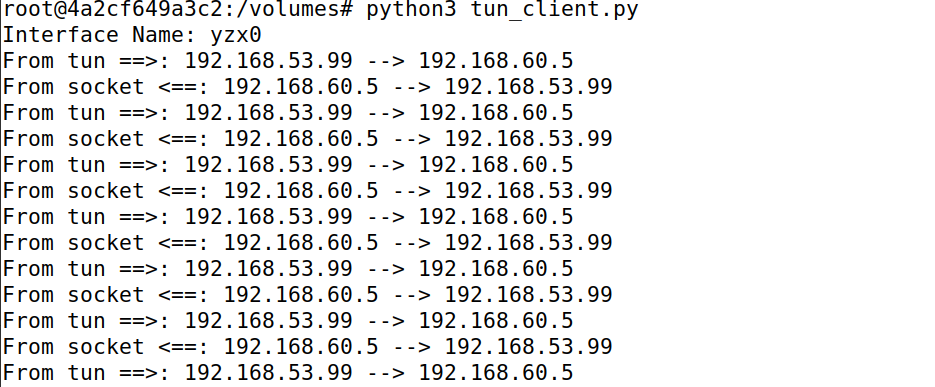
将tun\_client.py文件的while循环修改如上图，如果数据包来着tun接口，则发给主机U，如果来自socket接口，则发给隧道

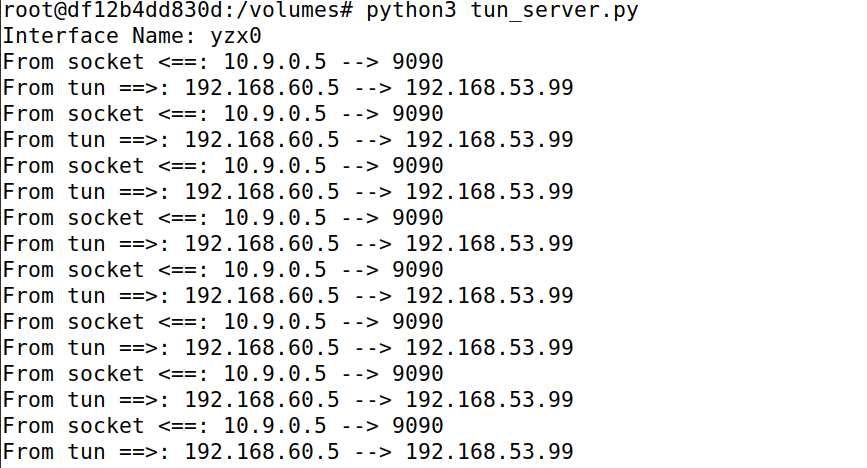


tun\_server.py文件中while循环修改如上图，如果数据包来自tun接口则发给主机V，若来自socket接口则发给隧道

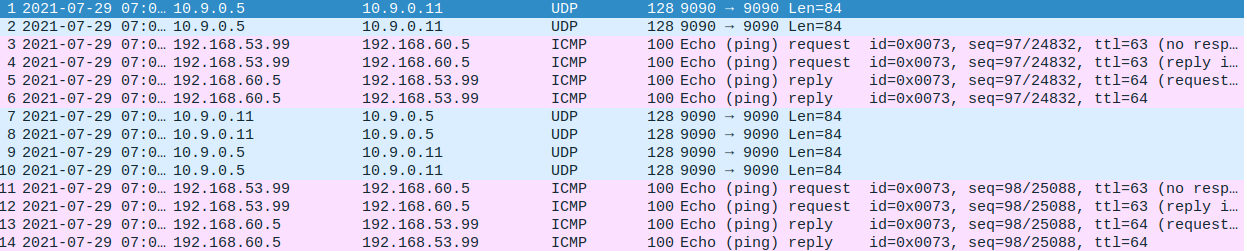


再次重复Task 3中操作，发现能够成功ping通

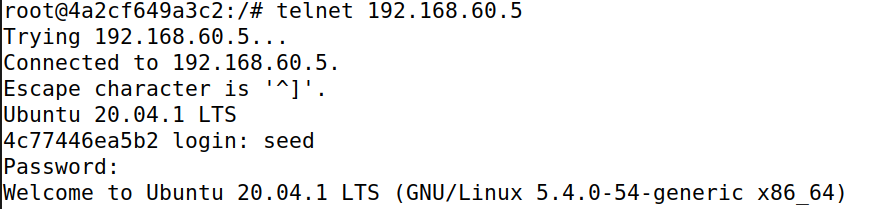




tun\_server.py与tun\_client.py运行命令下的输出入上两图所示

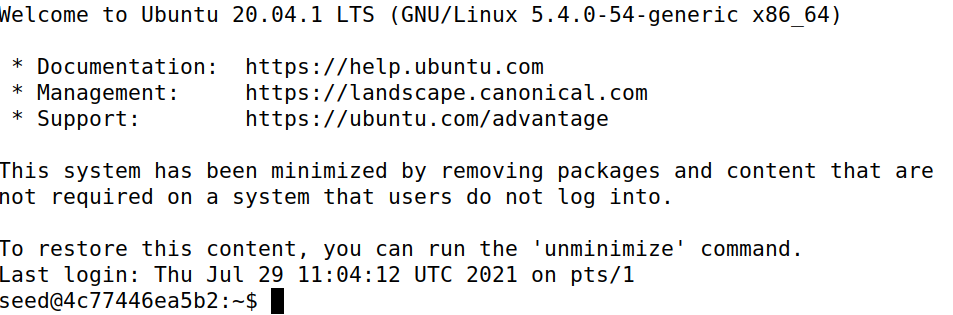


ping的过程中使用wireshark抓包，发现数据包从主机U先通过隧道到达VPN，然后VPN通过隧道发往主机V报文，主机V返回的响应报文通过隧道到达VPN，VPN又通过隧道发送给主机U，从而完成了主机U到主机V的通信

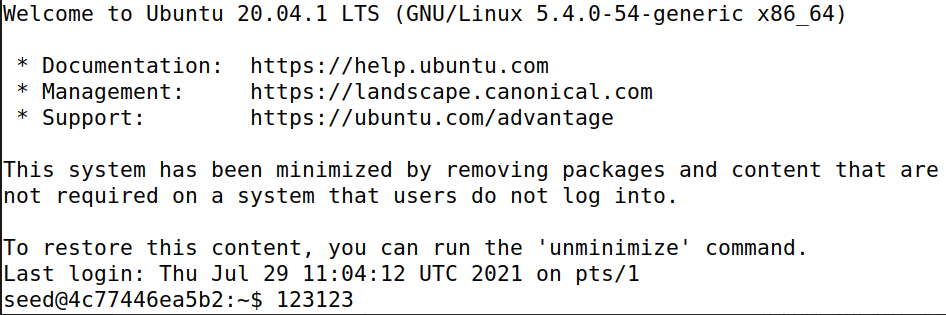


建立Telnet连接发现也可以成功连接

**Task 6 Tunnel-Breaking Experiment**



主机U向主机V建立Telnet连接，成功后终止tun\_server.py程序，发现已建立的连接中无法输入任何字符，这是因为隧道断开，数据包无法到达



短时间内再次运行tun\_server.py程序，发现前面输入未显示的字符会再次显示，因为终止程序后的输入会一直发送报文，短时间内恢复隧道后就仍可以显示之前的输入