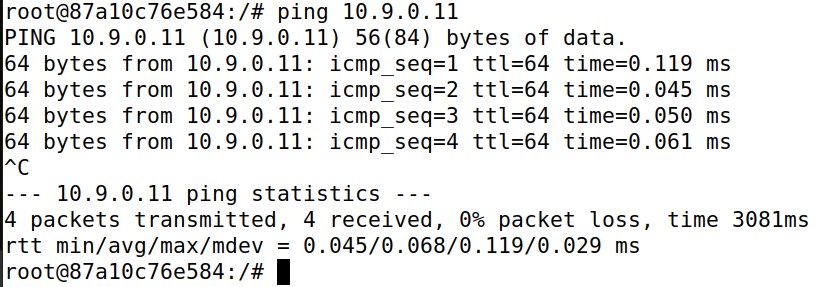
**VPN Lab: The Container Version**

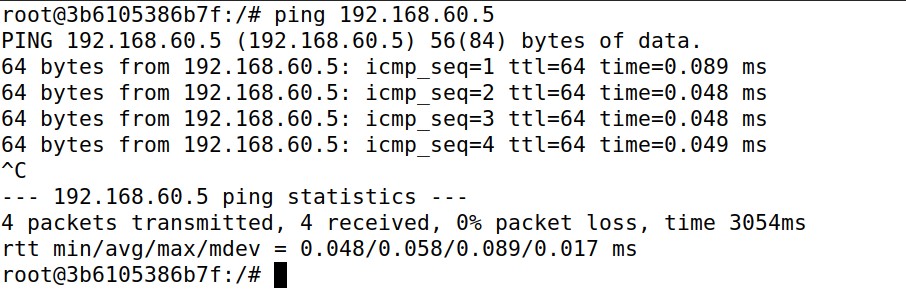
# Task1

主机 U：10.9.0.5，主机 V：192.168.60.5，VPN 服务器：10.9.0.11

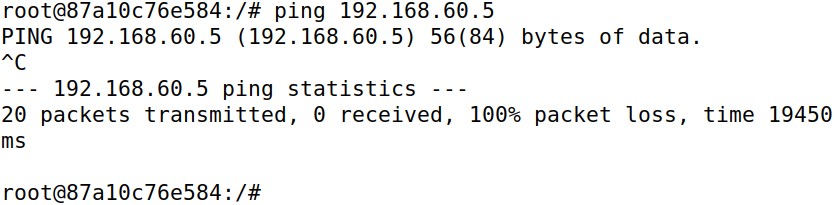
主机 U 和 VPN 服务器可以通：



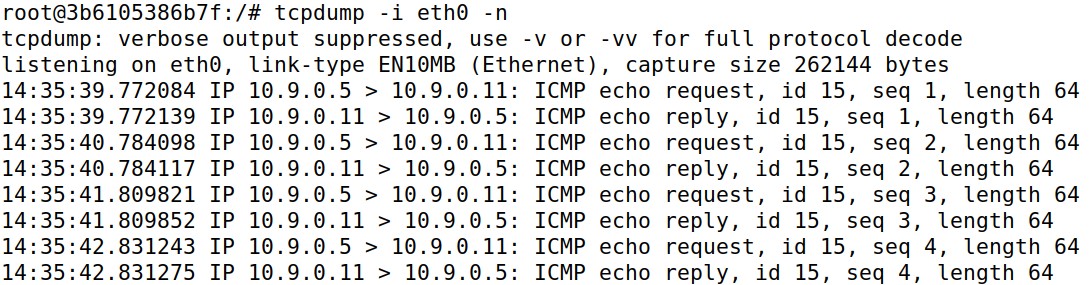
VPN 服务器和主机 V 可以通：



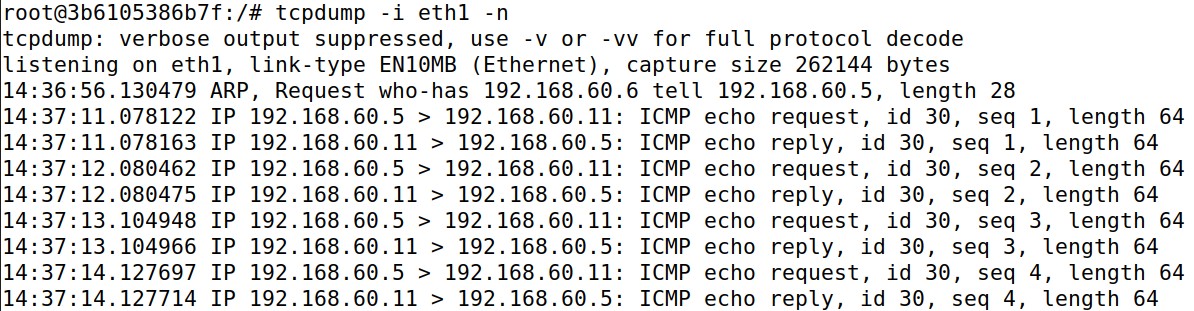
主机 U 和主机 V 不能通：



用 tcpdump 观察 eth0 接口，并在 U 上 ping VPN 服务器：



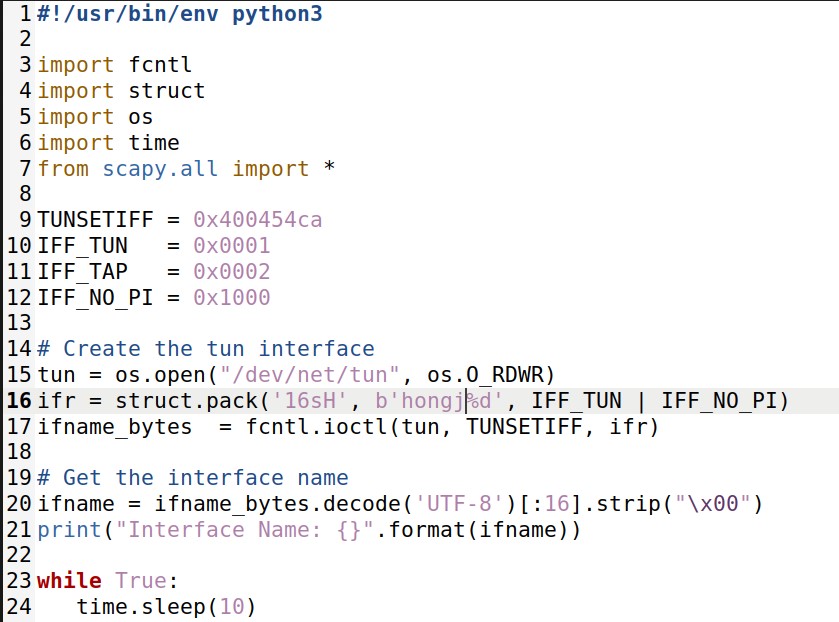
用 tcpdump 观察 eth1 接口，并在 V 上 ping 192.168.60.11：



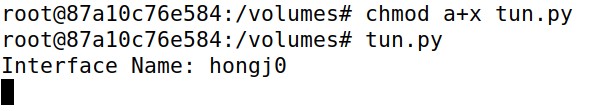
# Task2

## Task2.a

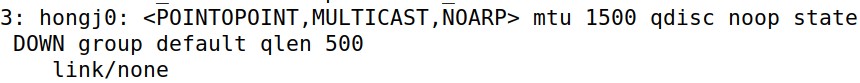
将第 16 行处代码改成自己的名字：



运行代码：

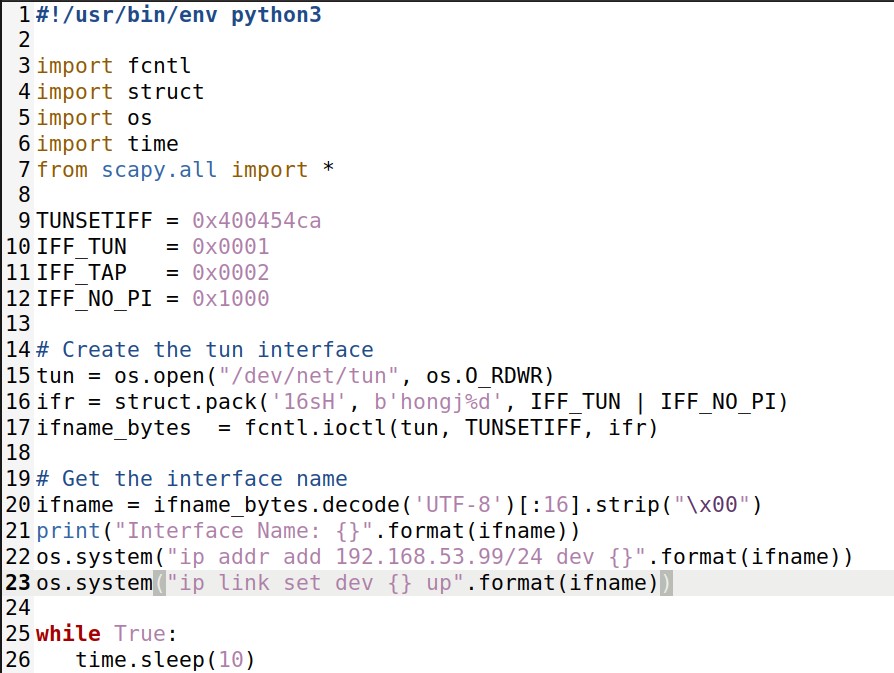


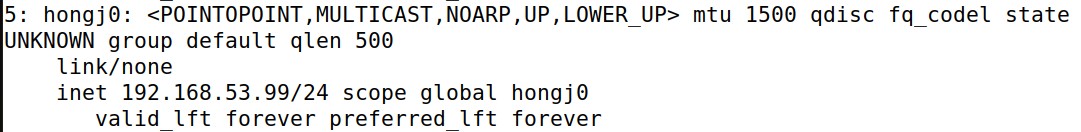
重开一个终端，输入 ip address 命令，看到已经有自定义的接口：



## Task2.b

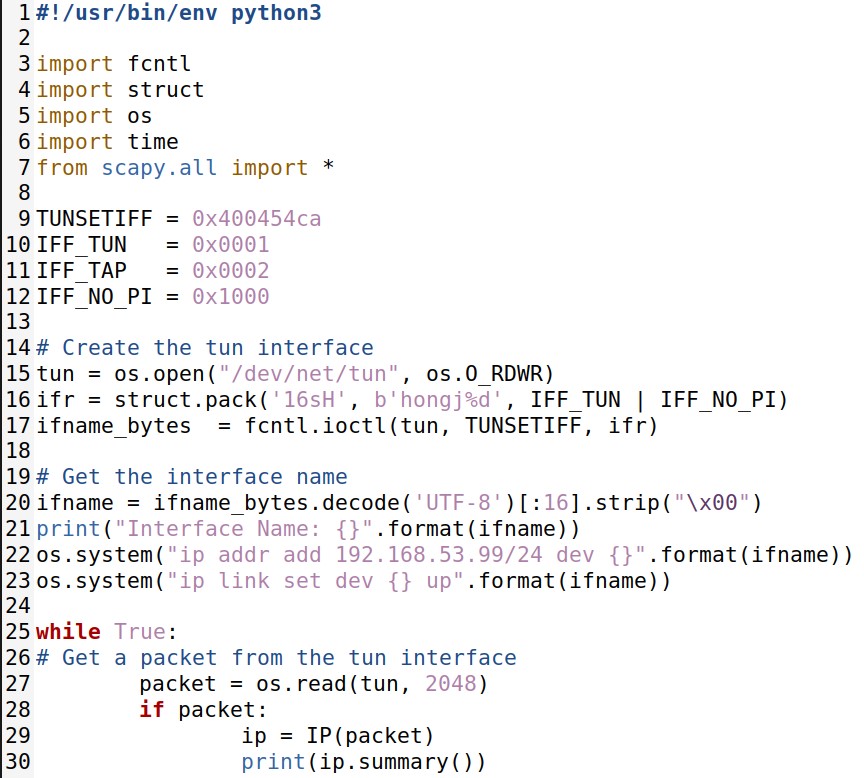
在原代码中 22 和 23 行加入手册所给代码：

 再次运行代码并查看，发现自定义的接口已经分配了地址。

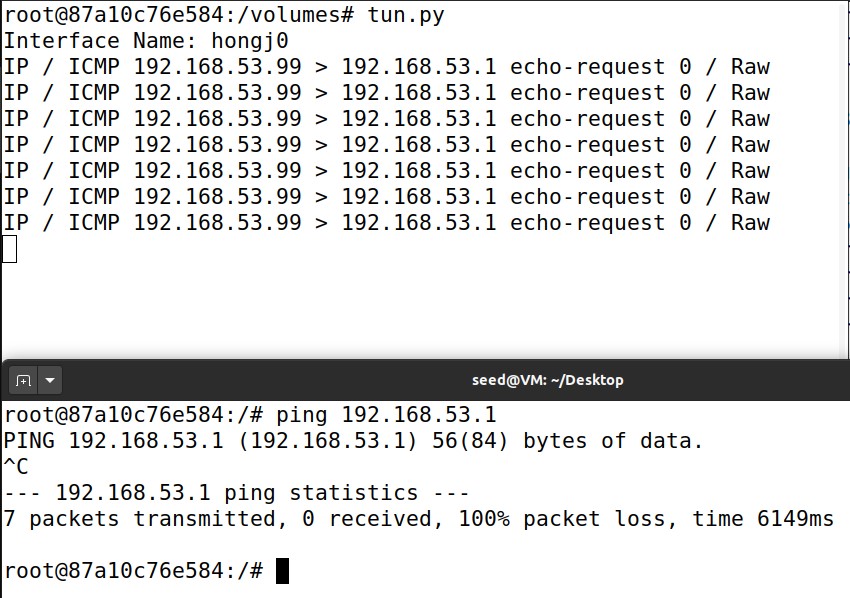


## Task2.c

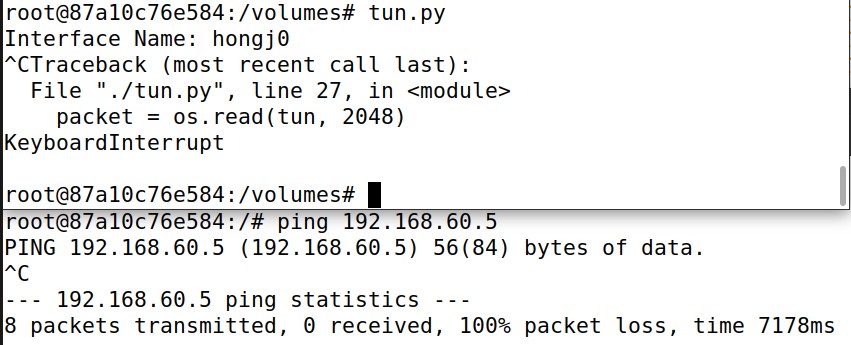
用手册所给 while 循环替换原代码：



运行代码并在 U 上 ping192.168.53.0/24 下的地址，代码出现响应，但是 ping 不通，因为实际主机并不存在。

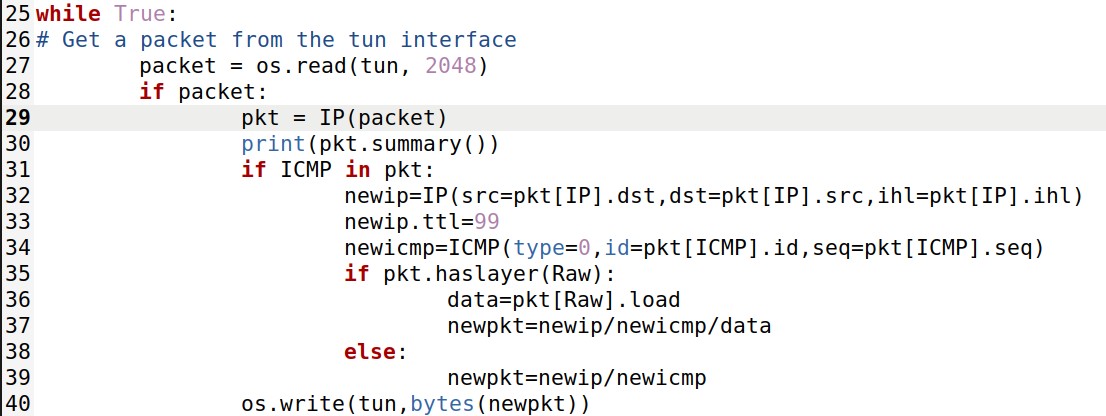


运行代码并在 U 上 ping192.168.60.5，代码未响应，且 ping 不通，因为未添加对应路由。

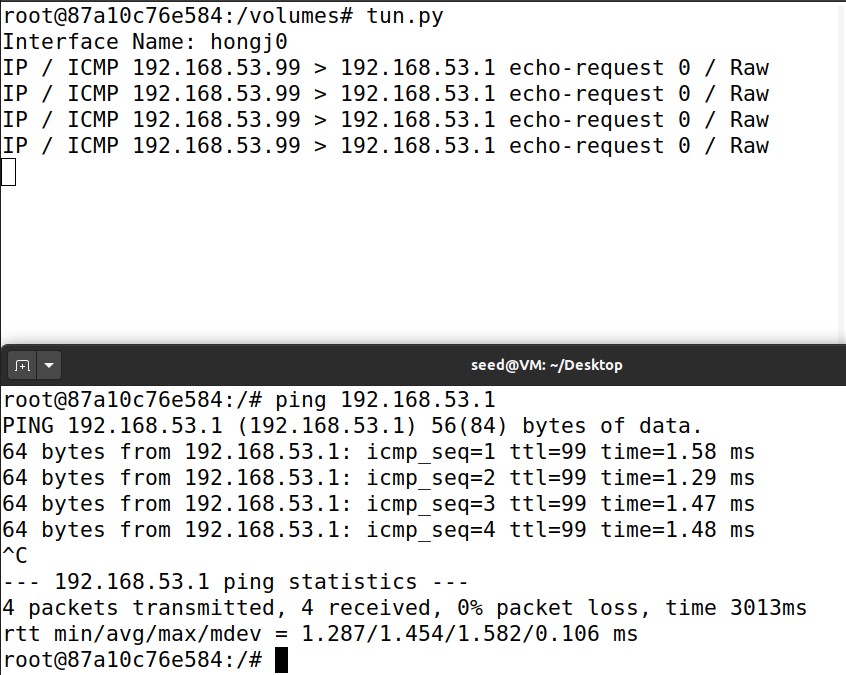


## Task2.d

修改 while 循环如下：

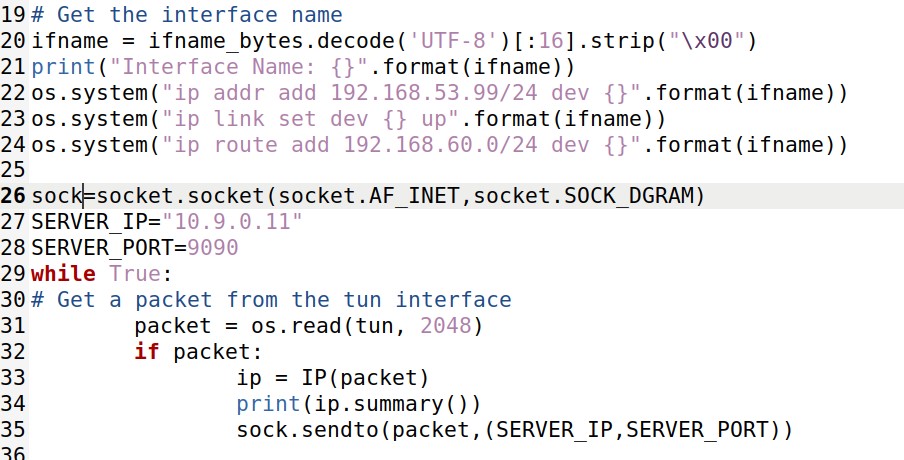


运行代码并在 U 上 ping 192.168.53.0/24 下面的地址，观察到返回的是构造的报文，且有 IP/ICMP/Raw 三层。

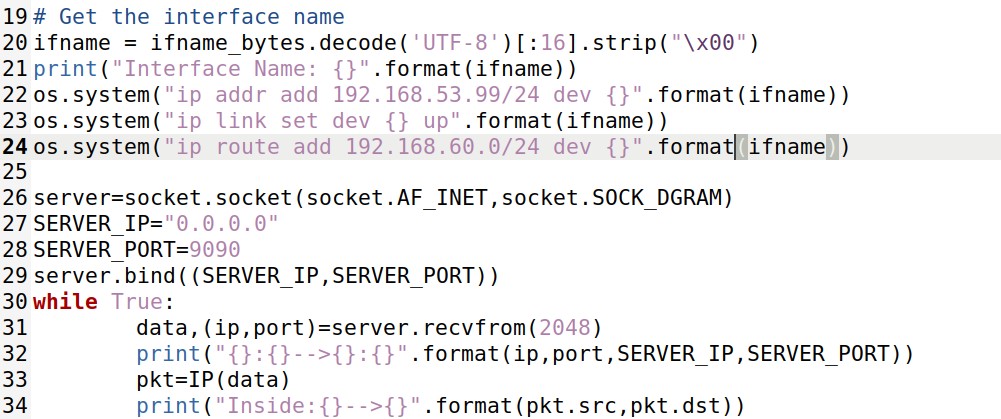


# Task3

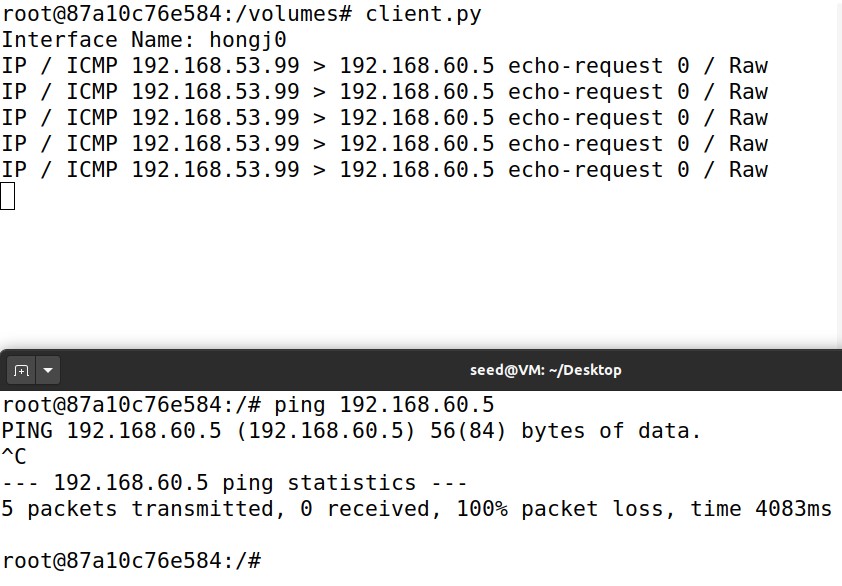
修改 tun，重命名为 client，修改代码如下：



修改 tun，重命名为 server，修改代码如下：

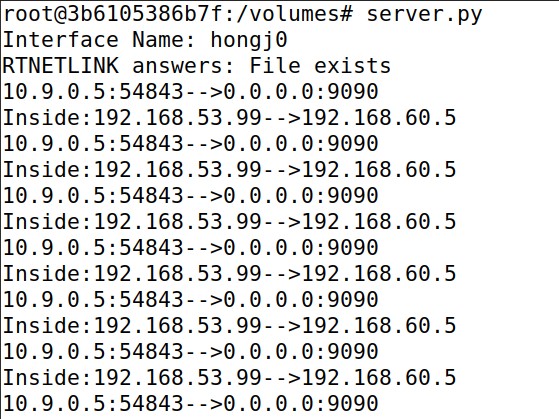


在 U 上运行 client，并 ping 192.168.60.5，看到代码响应未发生变化，依然 ping 不通。



在上一步的基础上，在 VPN 服务器运行 server，看到管道外部是

10.9.0.5->0.0.0.0，管道内部是 192.168.53.99->192.168.60.5。

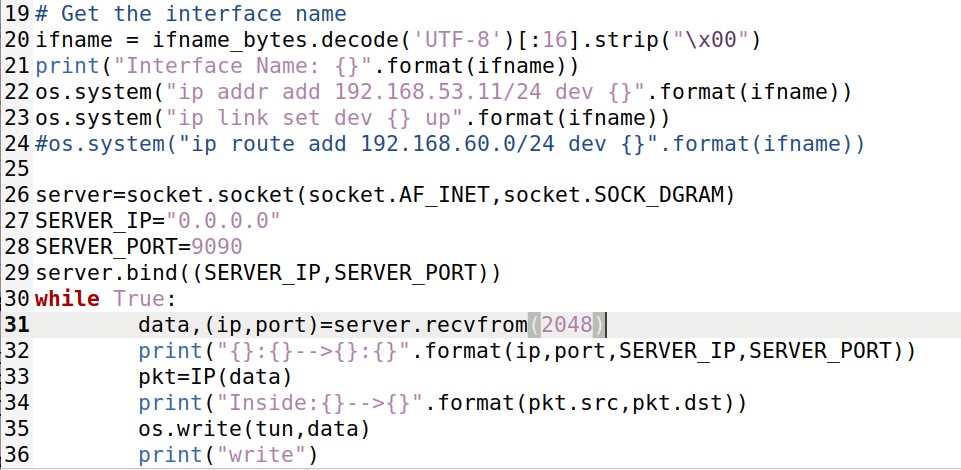


# Task4

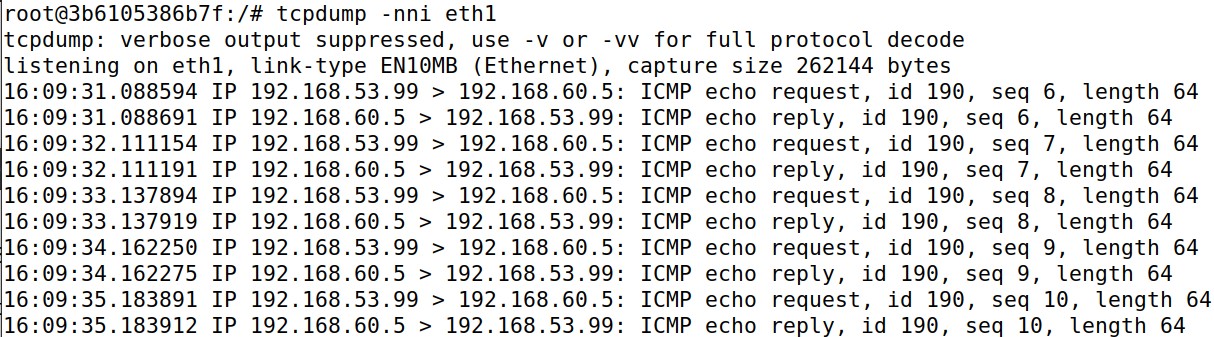
首先打开 docker-compose.yml，确保 IP 转发处于打开状态。



修改 server 代码如下：

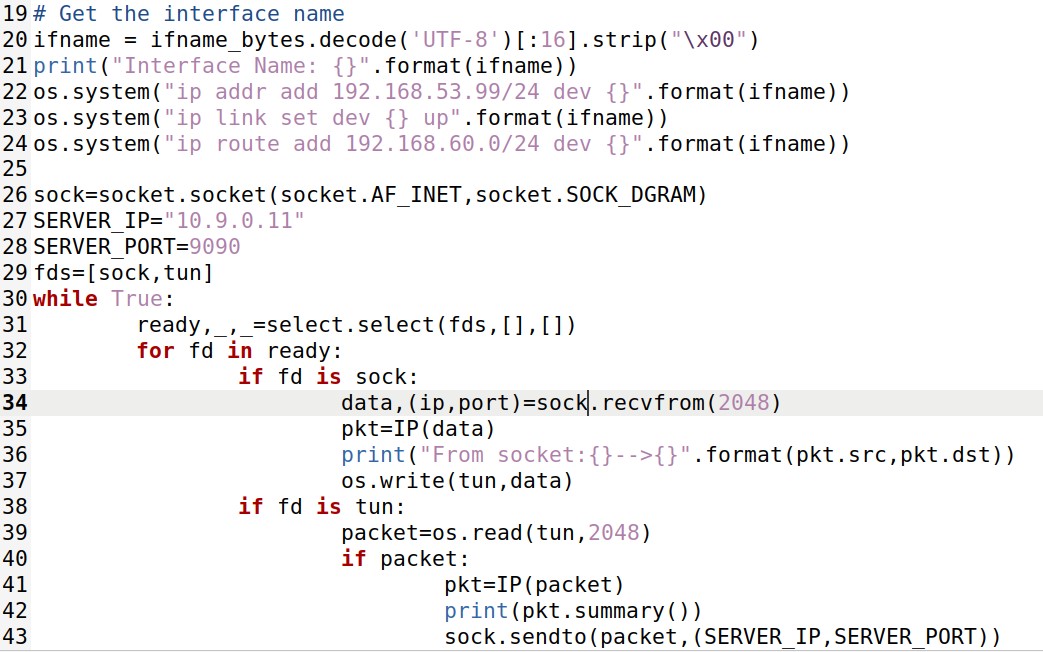


运行 server，client，在 U 上 ping192.168.60.5，此时在 VPN 服务器上用 tcpdump -nni eth1 命令看到 eth1 接口收到返回。

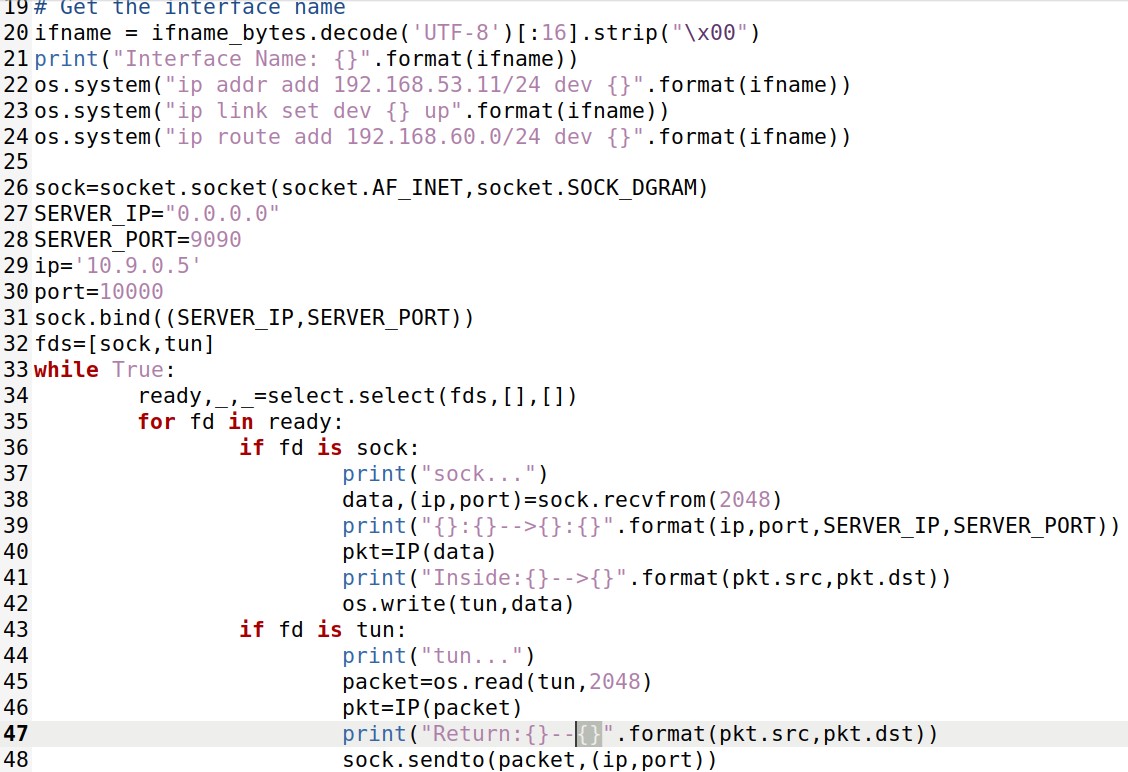


# Task5

修改 client 代码如下：

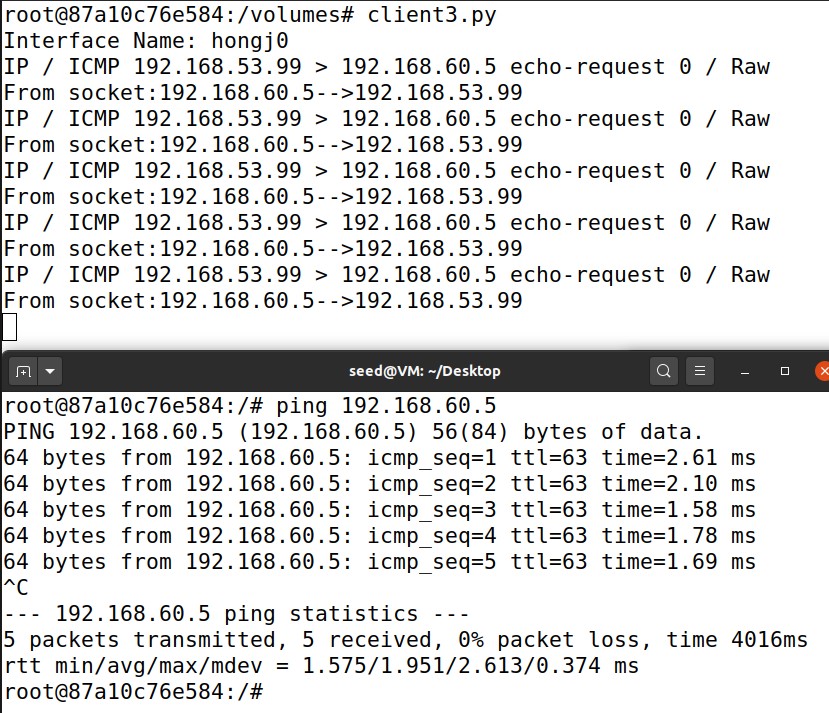


修改 server 代码如下：

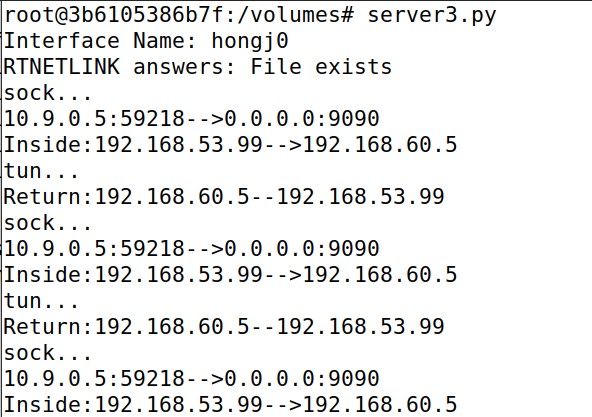


运行 client，server，在 U 上 ping192.168.60.5，此时能够 ping 通，代

码响应如下。



VPN 服务器上的响应如下：



# Task6

保持 client 和 server 运行，在 U 上 telnet192.168.60.5，成功进入，观察到每输入一个字符，client 和 server 都有一个响应。断开 client 或 server 之一，继续在 telnet 输入，此时输入无响应，且 client 和 server 中只有未被断开的有响应，重新连上断开的 client 或 server，telnet 中之前输入的字符又再次出现了，因为之前输入的字符在缓冲区不断重发。

