lab7

1. VPN Tunneling Lab

1.1. Task 1: Network Setup

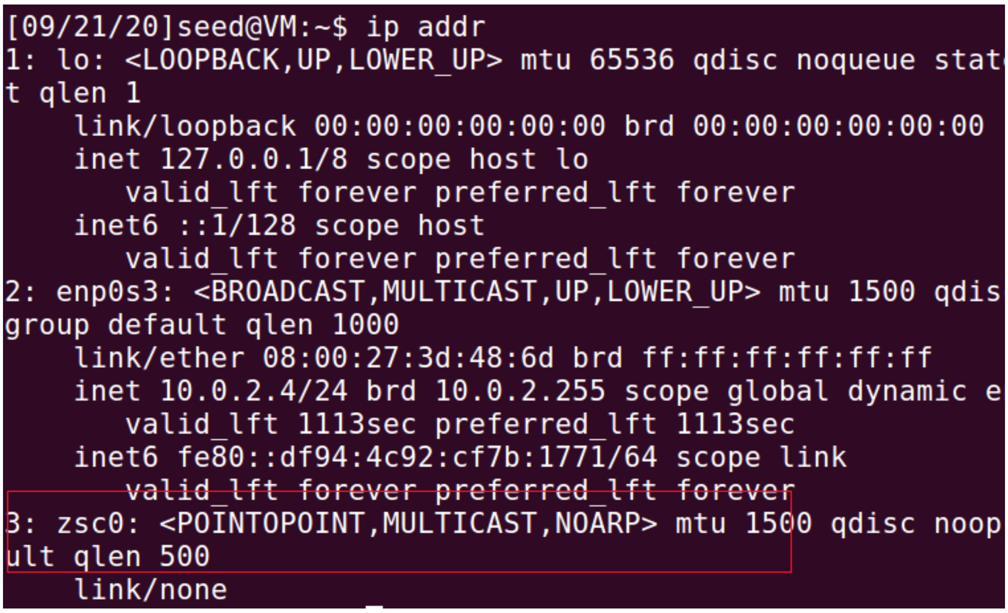
Testing. Please conduct the following testings to ensure that the network setup is performed correctly:

1. Host U can communicate with VPN Server.
2. VPN Server can communicate with Host V.
3. Host U should not be able to communicate with Host V.

1.2. Task 2: Create and Configure TUN Interface

1.2.1. Task 2.a: Name of the Interface

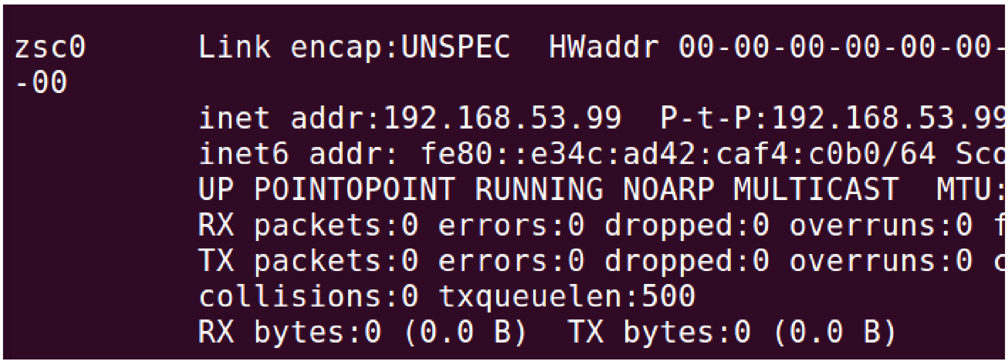
运行程序后，出现了名为`zsc0`的新接口，新接口没有ip。



```
[09/21/20]seed@VM:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue stat
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdis
group default qlen 1000
    link/ether 08:00:27:3d:48:6d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic e
       valid_lft 1113sec preferred_lft 1113sec
    inet6 fe80::df94:4c92:cf7b:1771/64 scope link
       valid_lft forever preferred_lft forever
3: zsc0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop
ult qlen 500
    link/none
```

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

运行新的程序后，此接口下出现了ip：



```
zsc0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-
-00
          inet addr:192.168.53.99  P-t-P:192.168.53.99
          inet6 addr: fe80::e34c:ad42:caf4:c0b0/64 Sco
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:
          RX packets:0 errors:0 dropped:0 overruns:0 f
          TX packets:0 errors:0 dropped:0 overruns:0 c
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

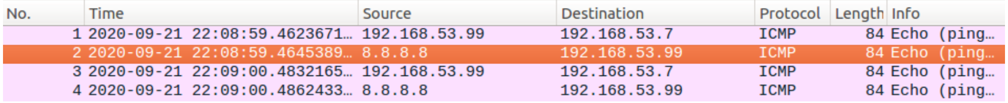
1.2.3. Task 2.c: Read from the TUN Interface

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

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

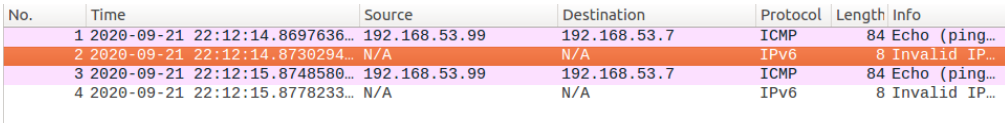
Please modify the tun.py code according to the following requirements:

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.



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-21 22:08:59.4623671…	192.168.53.99	192.168.53.7	ICMP	84	Echo (ping…
2	2020-09-21 22:08:59.4645389…	8.8.8.8	192.168.53.99	ICMP	84	Echo (ping…
3	2020-09-21 22:09:00.4832165…	192.168.53.99	192.168.53.7	ICMP	84	Echo (ping…
4	2020-09-21 22:09:00.4862433…	8.8.8.8	192.168.53.99	ICMP	84	Echo (ping…

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

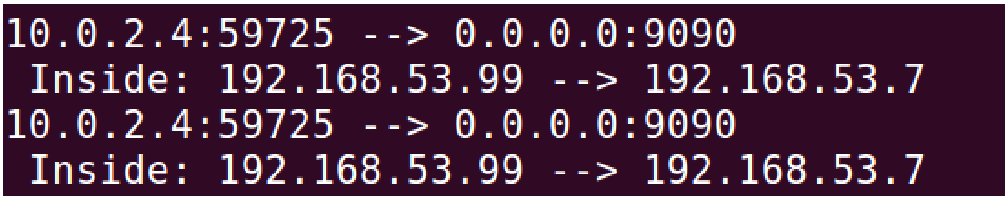


No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-21 22:12:14.8697636…	192.168.53.99	192.168.53.7	ICMP	84	Echo (ping…
2	2020-09-21 22:12:14.8730294…	N/A	N/A	IPv6	8	Invalid IP…
3	2020-09-21 22:12:15.8748580…	192.168.53.99	192.168.53.7	ICMP	84	Echo (ping…
4	2020-09-21 22:12:15.8778233…	N/A	N/A	IPv6	8	Invalid IP…

   出现了无法识别的报文。

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

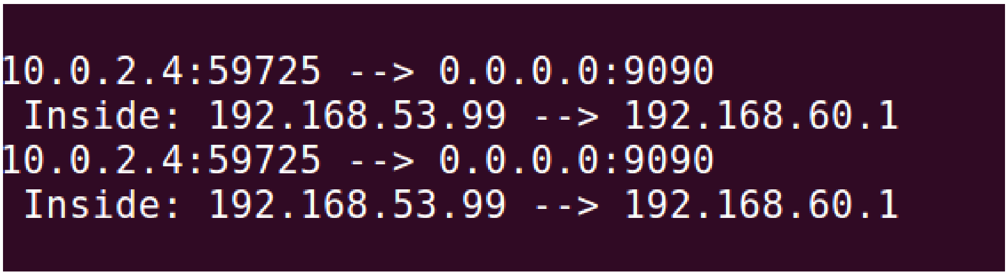
1. Run the tun server.py program on VPN Server, and then run tun client.py on Host U. To test whether the tunnel works or not, ping any IP address belonging to the 192.168.53.0/24 network. What is printed out on VPN Server? Why?



```
10.0.2.4:59725 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.7
10.0.2.4:59725 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.7
```

tun_client.py将icmp报文作为udp报文的payload发送给了tun_server.py，因此inside中的dst&src应为原先icmp报文的dst&src，即192.168.53.1/24。

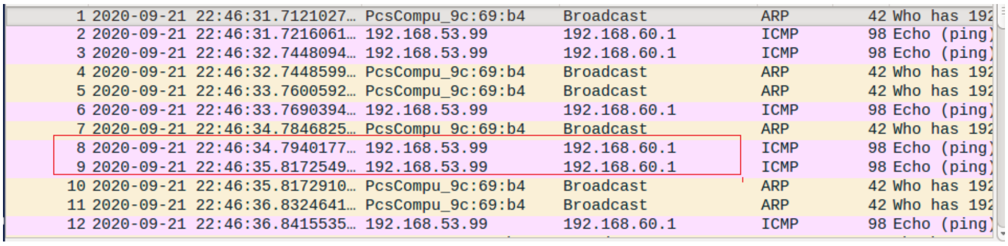
2. Please provide proofs to demonstrate that when you ping an IP address in the 192.168.60.0/24 network, the ICMP packets are received by tun server.py through the tunnel.



```
10.0.2.4:59725 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
10.0.2.4:59725 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
```

1.4. Task 4: Set Up the VPN Server

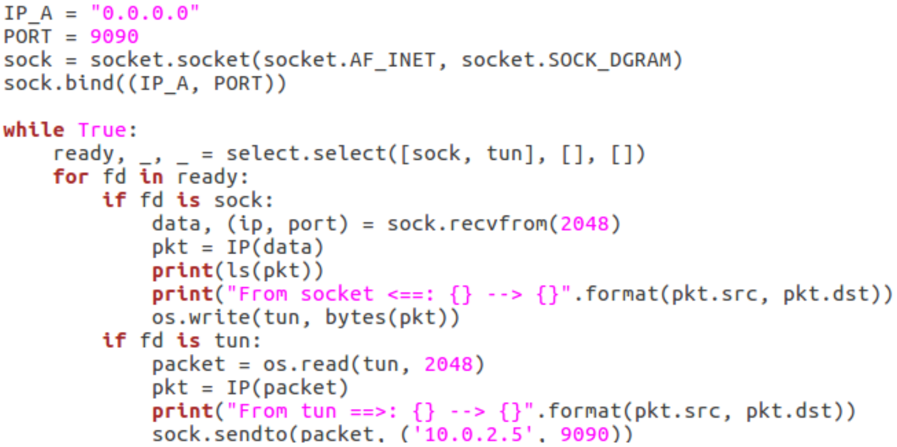
Testing. If everything is set up properly, we can ping Host V from Host U. The ICMP echo request packets should eventually arrive at Host V through the tunnel. Please show your proof. It should be noted that although Host V will respond to the ICMP packets, the reply will not get back to Host U, because we have not set up everything yet. Therefore, for this task, it is sufficient to show (using Wireshark) that the ICMP packets have arrived at Host V.



```
 1 2020-09-21 22:46:31.7121027… PcsCompu_9c:69:b4   Broadcast      ARP     42 Who has 192
 2 2020-09-21 22:46:31.7216061… 192.168.53.99       192.168.60.1   ICMP    98 Echo (ping)
 3 2020-09-21 22:46:32.7448094… 192.168.53.99       192.168.60.1   ICMP    98 Echo (ping)
 4 2020-09-21 22:46:32.7448599… PcsCompu_9c:69:b4   Broadcast      ARP     42 Who has 192
 5 2020-09-21 22:46:33.7600592… PcsCompu_9c:69:b4   Broadcast      ARP     42 Who has 192
 6 2020-09-21 22:46:33.7690394… 192.168.53.99       192.168.60.1   ICMP    98 Echo (ping)
 7 2020-09-21 22:46:34.7846825… PcsCompu_9c:69:b4   Broadcast      ARP     42 Who has 192
 8 2020-09-21 22:46:34.7940177… 192.168.53.99       192.168.60.1   ICMP    98 Echo (ping)
 9 2020-09-21 22:46:35.8172549… 192.168.53.99       192.168.60.1   ICMP    98 Echo (ping)
10 2020-09-21 22:46:35.8172910… PcsCompu_9c:69:b4   Broadcast      ARP     42 Who has 192
11 2020-09-21 22:46:36.8324641… PcsCompu_9c:69:b4   Broadcast      ARP     42 Who has 192
12 2020-09-21 22:46:36.8415535… 192.168.53.99       192.168.60.1   ICMP    98 Echo (ping)
```

1.5. Task 5: Handling Traffic in Both Directions

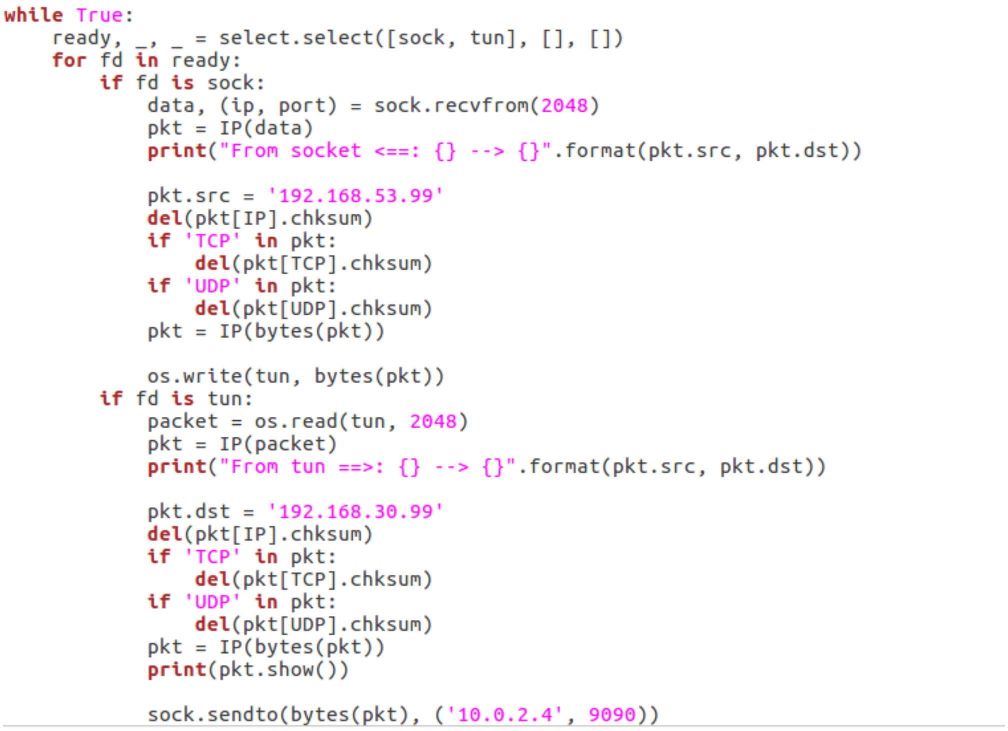
1. client



```
IP_A = "0.0.0.0"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

while True:
    ready, _, _ = select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print(ls(pkt))
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            os.write(tun, bytes(pkt))
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            sock.sendto(packet, ('10.0.2.5', 9090))
```

2. server



```
while True:
    ready, _, _ = select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))

            pkt.src = '192.168.53.99'
            del(pkt[IP].chksum)
            if 'TCP' in pkt:
                del(pkt[TCP].chksum)
            if 'UDP' in pkt:
                del(pkt[UDP].chksum)
            pkt = IP(bytes(pkt))

            os.write(tun, bytes(pkt))
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))

            pkt.dst = '192.168.30.99'
            del(pkt[IP].chksum)
            if 'TCP' in pkt:
                del(pkt[TCP].chksum)
            if 'UDP' in pkt:
                del(pkt[UDP].chksum)
            pkt = IP(bytes(pkt))
            print(pkt.show())

            sock.sendto(bytes(pkt), ('10.0.2.4', 9090))
```

1.6. Task 6: Tunnel-Breaking Experiment

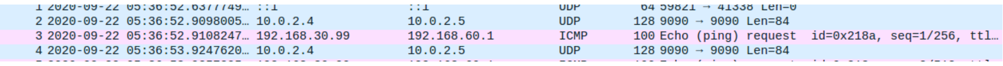
Once the tunnel is re-established, what is going to happen to the telnet connection? Please describe and explain your observations.
tunnel短暂断开没有导致telnet connection也断开。

1.7. Task 7: Routing Experiment on Host V

这一步可以跳过，在VPN Server上使用snat是更好的选择。

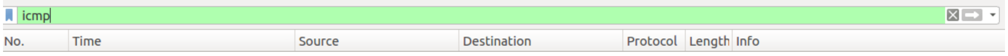
1.8. Task 8: Experiment with the TUN IP Address

1. Where are the packets dropped? Please provide evidence using Wireshark traces
   当packets从U抵达VPN Server后，VPN Server未能通过tun将packets发送出去。



```
1 2020-09-22 05:36:52.6377749… ::1             ::1             UDP     64 59821 → 41338 Len=0
2 2020-09-22 05:36:52.9098005… 10.0.2.4        10.0.2.5        UDP    128 9090 → 9090 Len=84
3 2020-09-22 05:36:52.9108247… 192.168.30.99   192.168.60.1    ICMP   100 Echo (ping) request  id=0x218a, seq=1/256, ttl…
4 2020-09-22 05:36:53.9247620… 10.0.2.4        10.0.2.5        UDP    128 9090 → 9090 Len=84
```

报文抵达VPN Server



```
icmp
No.  Time  Source  Destination  Protocol  Length  Info
```

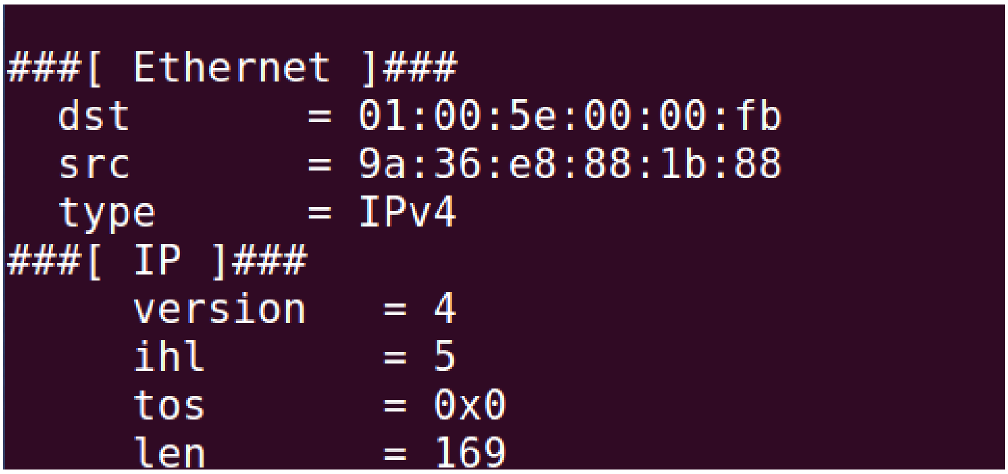
报文没有抵达V

2. Why are the packets dropped? See the hint below.
   因为报文中的src ip与tun不在同一网段
3. How to solve this problem? You are not allowed to change the IP address on any of the TUN interfaces? Please demonstrate that your solution works.

在VPN Server将U的报文写入tun前，将其源ip修改为与VPN Server的tun同一网段的任何ip。
在VPN Server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。
双管齐下，解决问题。

1.9. Task 9: Experiment with the TAP Interface

report and explain your observations.
报文中多出了额外的eth层信息：



```
###[ Ethernet ]###
  dst       = 01:00:5e:00:00:fb
  src       = 9a:36:e8:88:1b:88
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 169
```