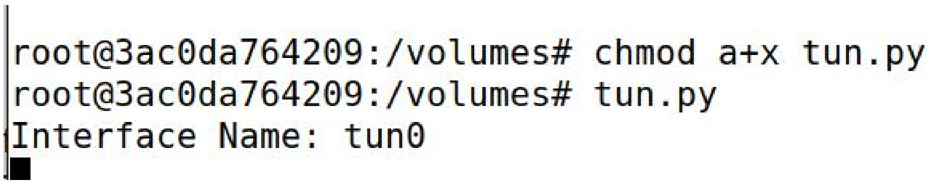
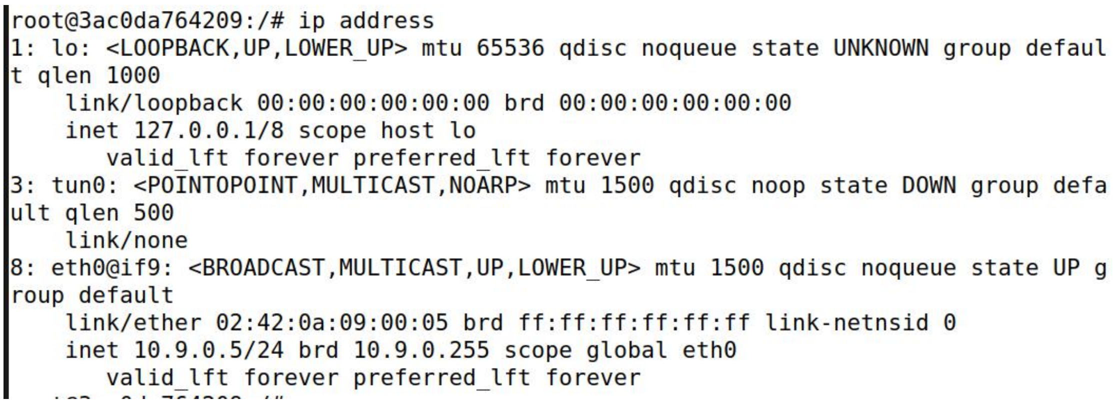
57118231 向颖

Task2.A



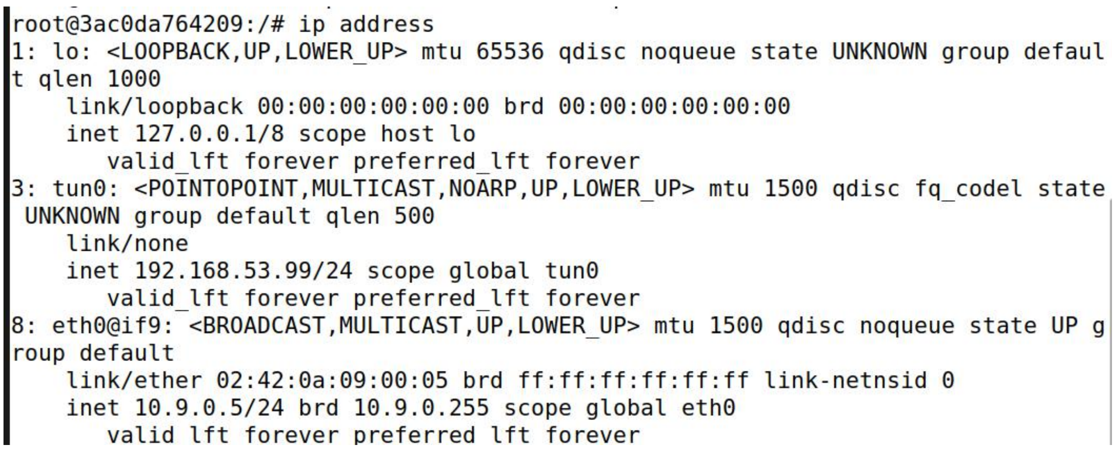
```
root@3ac0da764209:/volumes# chmod a+x tun.py
root@3ac0da764209:/volumes# tun.py
Interface Name: tun0
```



```
root@3ac0da764209:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    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
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defa
ult qlen 500
    link/none
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Task2.B

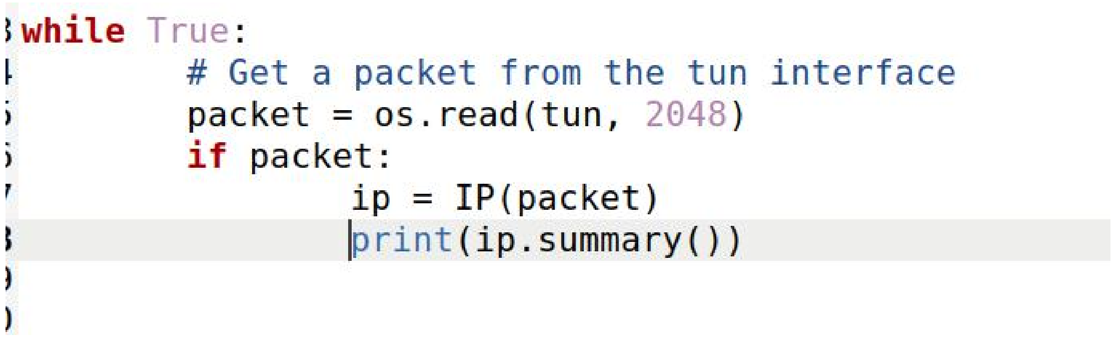
执行两个指令后查看 ip address，如图，可以看到 tun0 有了 ip 地址



```
root@3ac0da764209:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    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
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
 UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global tun0
       valid_lft forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
```

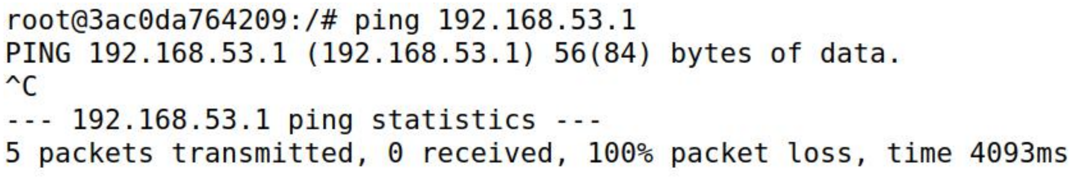
Task2.C

修改代码

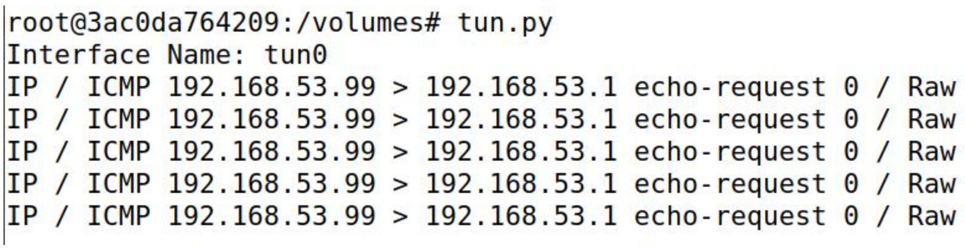


```python
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                print(ip.summary())
```

重新执行 tun.py，配置接口地址并开启接口，ping192.168.53.0/24 网段，如图所示



```
root@3ac0da764209:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4093ms
```

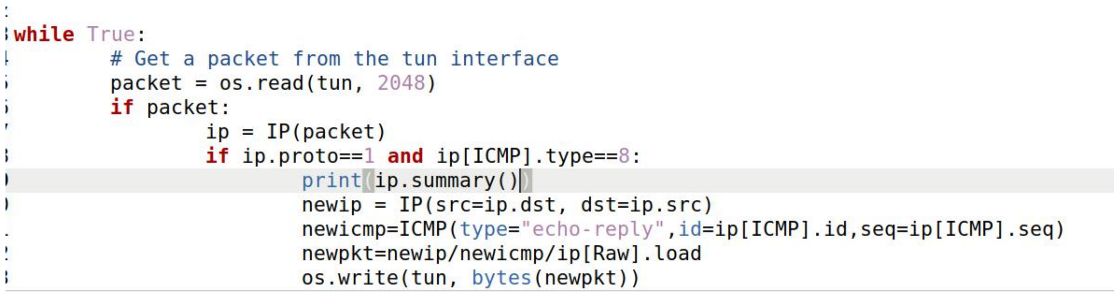


```
root@3ac0da764209:/volumes# tun.py
Interface Name: tun0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

Ping192.168.60.0/24 网段，无法 ping 通且没有输出，因为目前没有到达 192.168.60.0/24 网段的路由

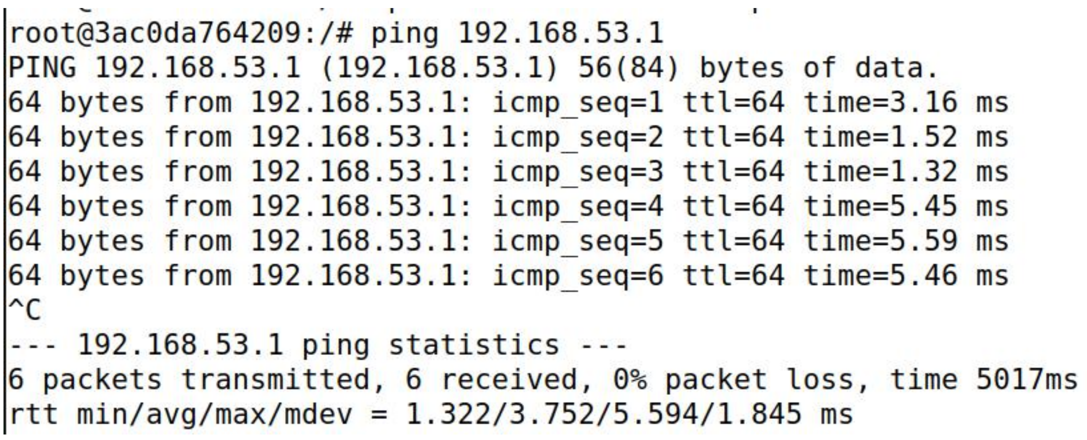
Task2.D

代码如下



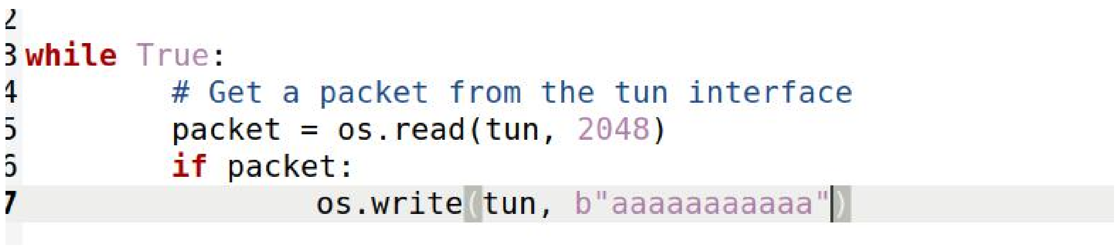
```python
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                if ip.proto==1 and ip[ICMP].type==8:
                        print(ip.summary())
                        newip = IP(src=ip.dst, dst=ip.src)
                        newicmp=ICMP(type="echo-reply",id=ip[ICMP].id,seq=ip[ICMP].seq)
                        newpkt=newip/newicmp/ip[Raw].load
                        os.write(tun, bytes(newpkt))
```

未执行 tun.py 时无法 Ping 通 192.168.53.1，执行后如图所示，能够 ping 通



```
root@3ac0da764209:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
64 bytes from 192.168.53.1: icmp_seq=1 ttl=64 time=3.16 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=64 time=1.52 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=64 time=1.32 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=64 time=5.45 ms
64 bytes from 192.168.53.1: icmp_seq=5 ttl=64 time=5.59 ms
64 bytes from 192.168.53.1: icmp_seq=6 ttl=64 time=5.46 ms
^C
--- 192.168.53.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5017ms
rtt min/avg/max/mdev = 1.322/3.752/5.594/1.845 ms
```

修改 tun.py，如图所示，无法 ping 通，写入的任意字符串会被当做伪造的 IP 报文

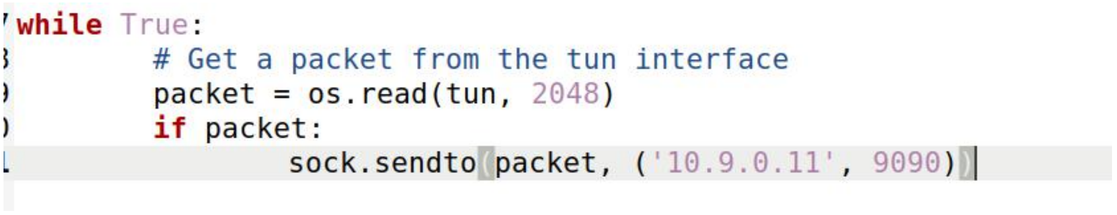


```python
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                os.write(tun, b"aaaaaaaaaaa")
```

task3

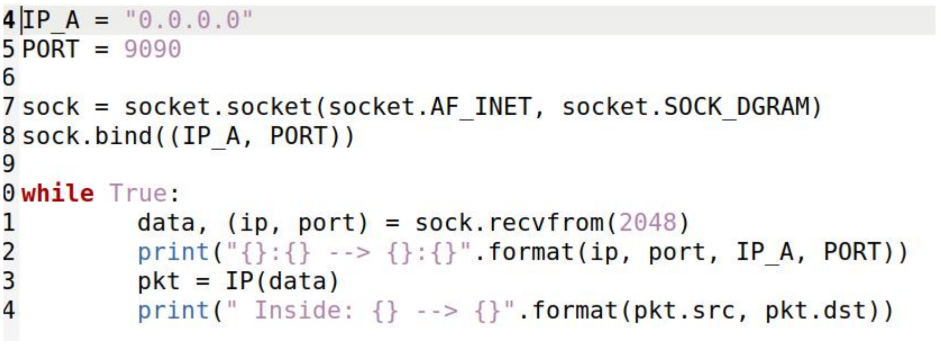
代码如下

Tun_client:



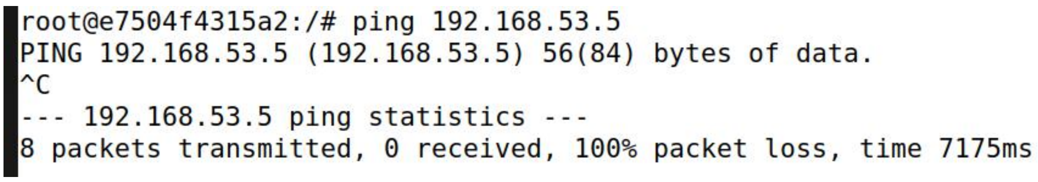
```
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                sock.sendto(packet, ('10.9.0.11', 9090))
```

Tun_server:



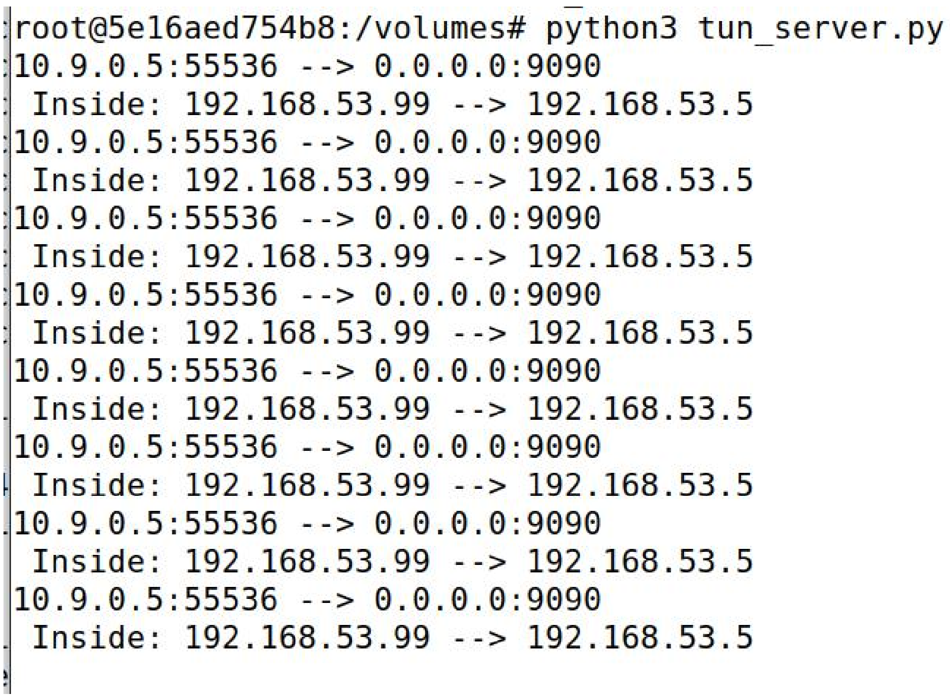
```
IP_A = "0.0.0.0"
PORT = 9090

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

while True:
        data, (ip, port) = sock.recvfrom(2048)
        print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
        pkt = IP(data)
        print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
```

Ping 192.168.53.5



```
root@e7504f4315a2:/# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
^C
--- 192.168.53.5 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7175ms
```

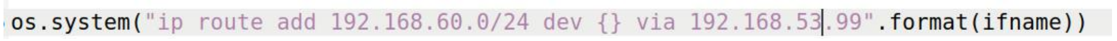
在 vpn server 上有如下输出



```
root@5e16aed754b8:/volumes# python3 tun_server.py
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:55536 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
```

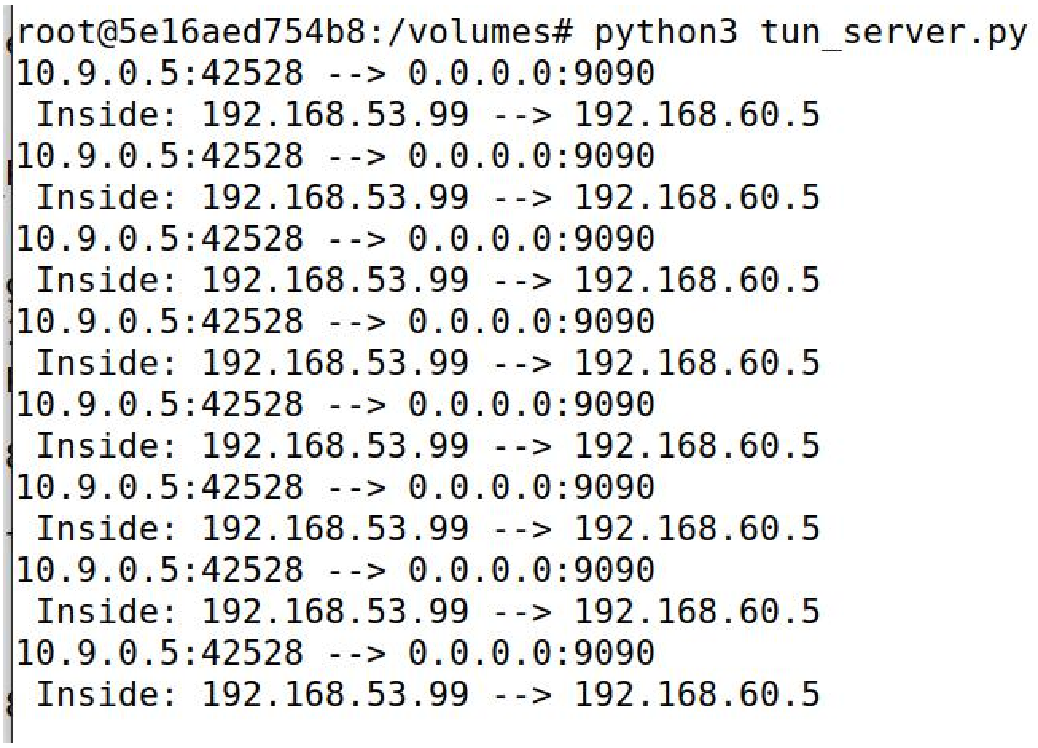
现在 ping 192.168.60.0/24 网段不通，因为没有路由

修改 tun_client.py，添加如下代码



```
os.system("ip route add 192.168.60.0/24 dev {} via 192.168.53.99".format(ifname))
```

现在 ping 192.168.60.5，在 vpn server 上有如下输出

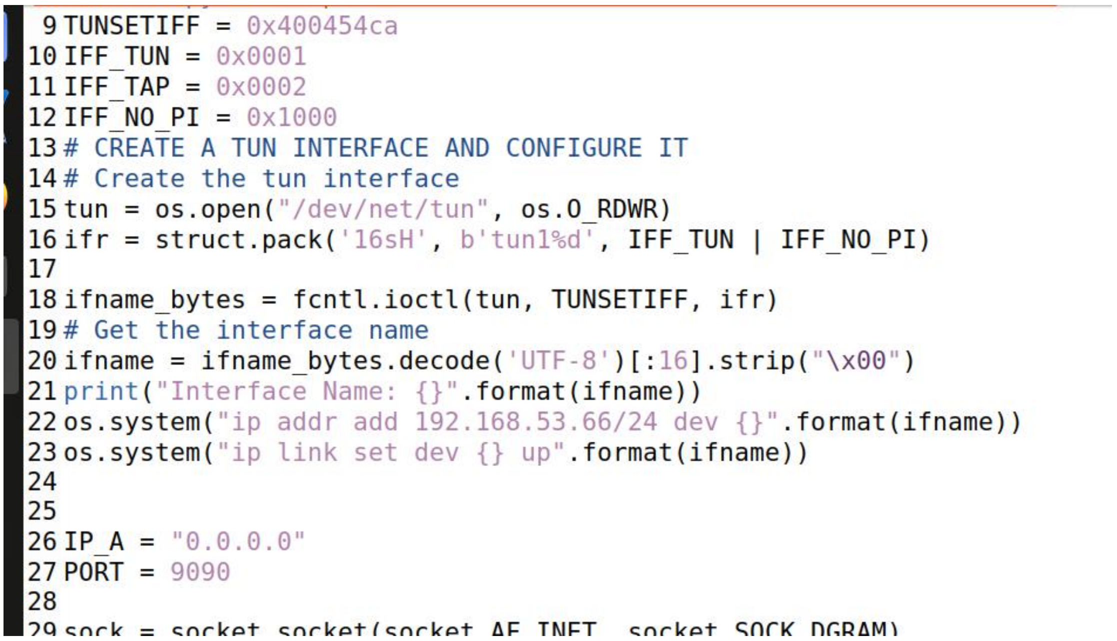


```
root@5e16aed754b8:/volumes# python3 tun_server.py
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42528 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

Task4

程序如下

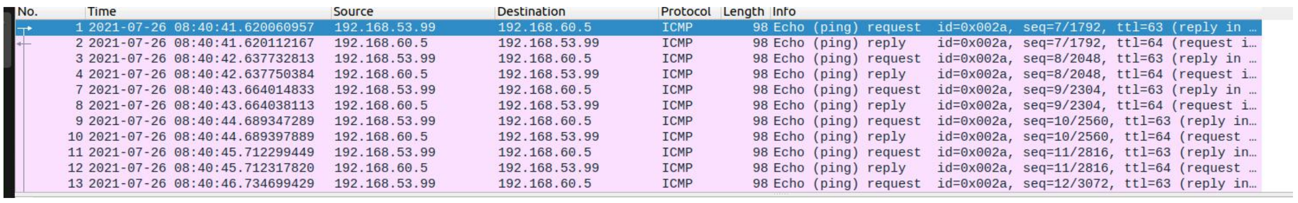
Client_server:



```
TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
# CREATE A TUN INTERFACE AND CONFIGURE IT
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tun1%d', IFF_TUN | IFF_NO_PI)

ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.66/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))


IP_A = "0.0.0.0"
PORT = 9090

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

开启服务端和客户端后在客户端 ping 192.168.60.5，查看 wireshark

下图是 wireshark 监听 192.168.60.5 的结果，可以看到收到了 icmp request 并返回了 reply，但并没有 ping 通，说明 reply 没有返回到 10.9.0.5，因为没有配置好路由

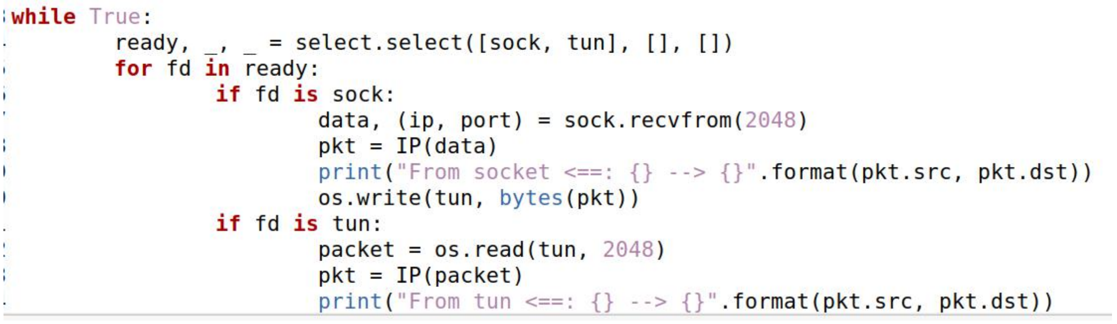


```
No.  Time                           Source          Destination     Protocol Length Info
   1 2021-07-26 08:40:41.620060957  192.168.53.99   192.168.60.5    ICMP        98 Echo (ping) request  id=0x002a, seq=7/1792, ttl=63 (reply in …
   2 2021-07-26 08:40:41.620112167  192.168.60.5    192.168.53.99   ICMP        98 Echo (ping) reply    id=0x002a, seq=7/1792, ttl=64 (request i…
   3 2021-07-26 08:40:42.637732813  192.168.53.99   192.168.60.5    ICMP        98 Echo (ping) request  id=0x002a, seq=8/2048, ttl=63 (reply in …
   4 2021-07-26 08:40:42.637750384  192.168.60.5    192.168.53.99   ICMP        98 Echo (ping) reply    id=0x002a, seq=8/2048, ttl=64 (request i…
   7 2021-07-26 08:40:43.664014833  192.168.53.99   192.168.60.5    ICMP        98 Echo (ping) request  id=0x002a, seq=9/2304, ttl=63 (reply in …
   8 2021-07-26 08:40:43.664038113  192.168.60.5    192.168.53.99   ICMP        98 Echo (ping) reply    id=0x002a, seq=9/2304, ttl=64 (request i…
   9 2021-07-26 08:40:44.689347289  192.168.53.99   192.168.60.5    ICMP        98 Echo (ping) request  id=0x002a, seq=10/2560, ttl=63 (reply in…
  10 2021-07-26 08:40:44.689397889  192.168.60.5    192.168.53.99   ICMP        98 Echo (ping) reply    id=0x002a, seq=10/2560, ttl=64 (request …
  11 2021-07-26 08:40:45.712299449  192.168.53.99   192.168.60.5    ICMP        98 Echo (ping) request  id=0x002a, seq=11/2816, ttl=63 (reply in…
  12 2021-07-26 08:40:45.712317820  192.168.60.5    192.168.53.99   ICMP        98 Echo (ping) reply    id=0x002a, seq=11/2816, ttl=64 (request …
  13 2021-07-26 08:40:46.734699429  192.168.53.99   192.168.60.5    ICMP        98 Echo (ping) request  id=0x002a, seq=12/3072, ttl=63 (reply in…
```

Task5

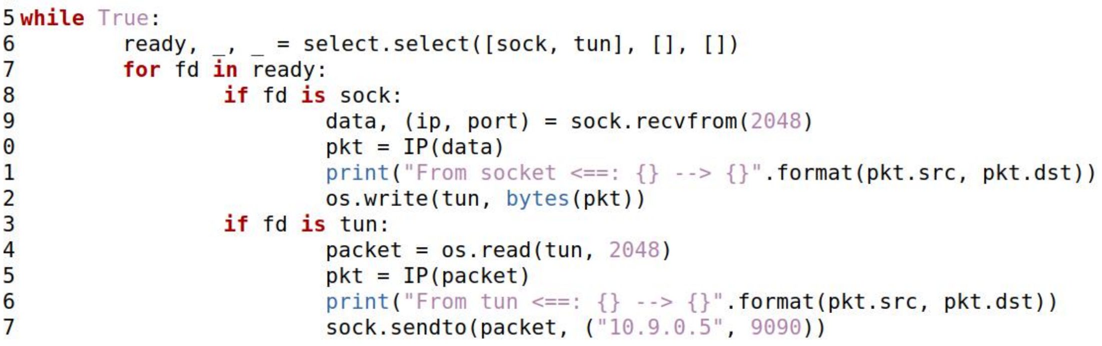
程序如下

Tun_client



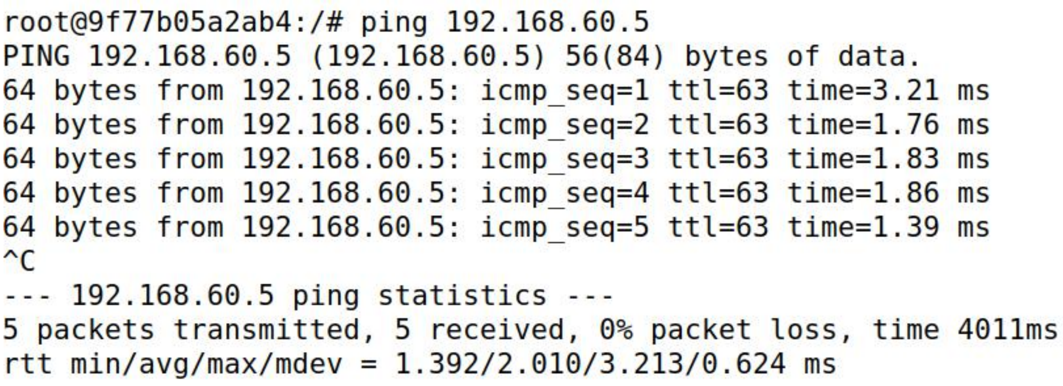
```python
while True:
        ready, _, _ = select.select([sock, tun], [], [])
        for fd in ready:
                if fd is sock:
                        data, (ip, port) = sock.recvfrom(2048)
                        pkt = IP(data)
                        print("From socket <==: {} --> {}".format(pkt.src,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

Tun_server

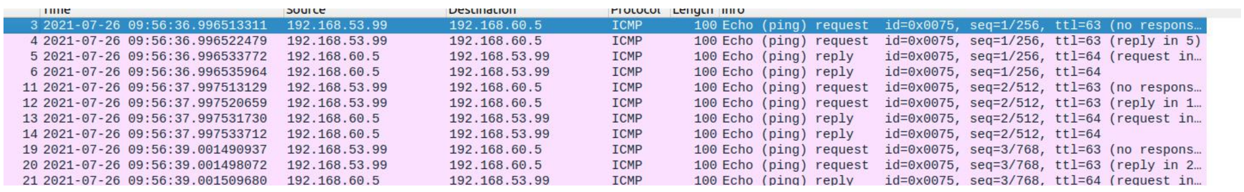


```python
while True:
        ready, _, _ = select.select([sock, tun], [], [])
        for fd in ready:
                if fd is sock:
                        data, (ip, port) = sock.recvfrom(2048)
                        pkt = IP(data)
                        print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
                        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, ("10.9.0.5", 9090))
```

10.9.0.5ping192.168.60.5,如图所示，ping 通

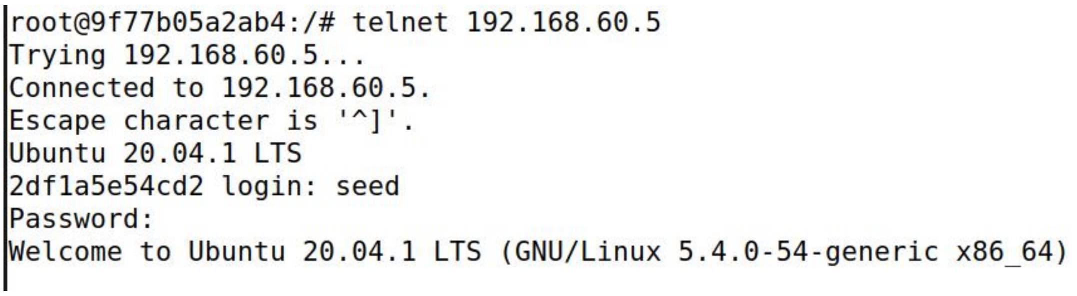


```
root@9f77b05a2ab4:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=3.21 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=1.76 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=1.83 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=1.86 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=1.39 ms
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 1.392/2.010/3.213/0.624 ms
```

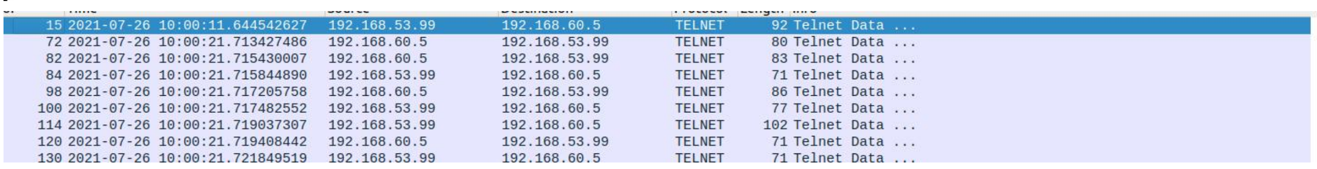


```
     Time                           Source          Destination     Protocol Length Info
   3 2021-07-26 09:56:36.996513311  192.168.53.99   192.168.60.5    ICMP       100 Echo (ping) request  id=0x0075, seq=1/256, ttl=63 (no respons…
   4 2021-07-26 09:56:36.996522479  192.168.53.99   192.168.60.5    ICMP       100 Echo (ping) request  id=0x0075, seq=1/256, ttl=63 (reply in 5)
   5 2021-07-26 09:56:36.996533772  192.168.60.5    192.168.53.99   ICMP       100 Echo (ping) reply    id=0x0075, seq=1/256, ttl=64 (request in…
   6 2021-07-26 09:56:36.996535964  192.168.60.5    192.168.53.99   ICMP       100 Echo (ping) reply    id=0x0075, seq=1/256, ttl=64
  11 2021-07-26 09:56:37.997513129  192.168.53.99   192.168.60.5    ICMP       100 Echo (ping) request  id=0x0075, seq=2/512, ttl=63 (no respons…
  12 2021-07-26 09:56:37.997520659  192.168.53.99   192.168.60.5    ICMP       100 Echo (ping) request  id=0x0075, seq=2/512, ttl=63 (reply in 1…
  13 2021-07-26 09:56:37.997531730  192.168.60.5    192.168.53.99   ICMP       100 Echo (ping) reply    id=0x0075, seq=2/512, ttl=64 (request in…
  14 2021-07-26 09:56:37.997533712  192.168.60.5    192.168.53.99   ICMP       100 Echo (ping) reply    id=0x0075, seq=2/512, ttl=64
  19 2021-07-26 09:56:39.001490937  192.168.53.99   192.168.60.5    ICMP       100 Echo (ping) request  id=0x0075, seq=3/768, ttl=63 (no respons…
  20 2021-07-26 09:56:39.001498072  192.168.53.99   192.168.60.5    ICMP       100 Echo (ping) request  id=0x0075, seq=3/768, ttl=63 (reply in 2…
  21 2021-07-26 09:56:39.001509680  192.168.60.5    192.168.53.99   ICMP       100 Echo (ping) reply    id=0x0075, seq=3/768, ttl=64 (request in…
```

telnet 连接成功，如图所示



```
root@9f77b05a2ab4:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
2df1a5e54cd2 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

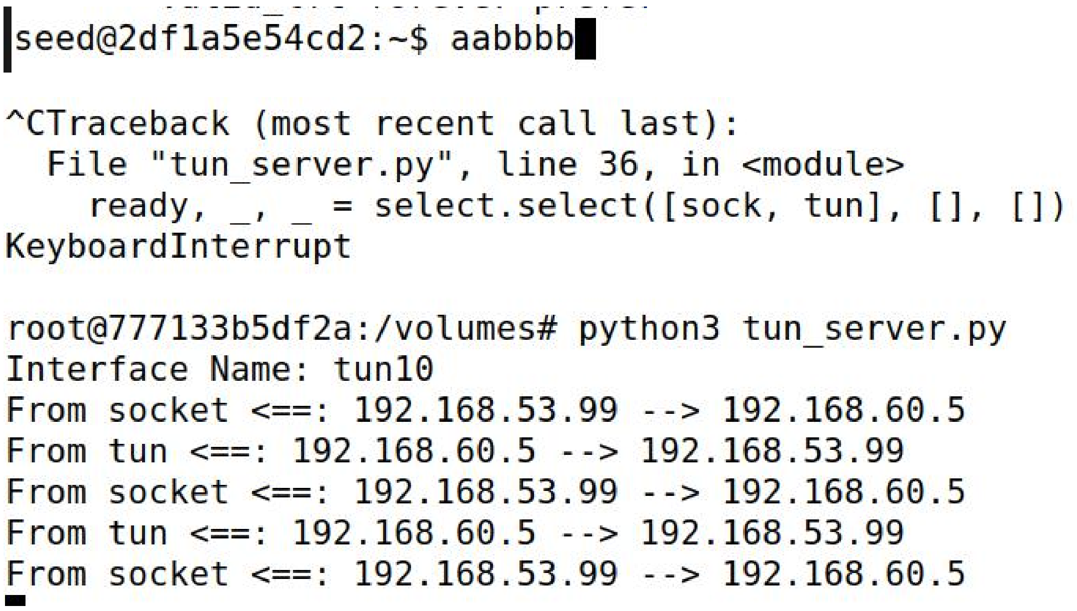


```
 15 2021-07-26 10:00:11.644542627  192.168.53.99  192.168.60.5   TELNET  92 Telnet Data ...
 72 2021-07-26 10:00:21.713427486  192.168.60.5   192.168.53.99  TELNET  80 Telnet Data ...
 82 2021-07-26 10:00:21.715430007  192.168.60.5   192.168.53.99  TELNET  83 Telnet Data ...
 84 2021-07-26 10:00:21.715844890  192.168.53.99  192.168.60.5   TELNET  71 Telnet Data ...
 98 2021-07-26 10:00:21.717205758  192.168.60.5   192.168.53.99  TELNET  86 Telnet Data ...
100 2021-07-26 10:00:21.717482552  192.168.53.99  192.168.60.5   TELNET  77 Telnet Data ...
114 2021-07-26 10:00:21.719037307  192.168.53.99  192.168.60.5   TELNET 102 Telnet Data ...
120 2021-07-26 10:00:21.719408442  192.168.60.5   192.168.53.99  TELNET  71 Telnet Data ...
130 2021-07-26 10:00:21.721849519  192.168.53.99  192.168.60.5   TELNET  71 Telnet Data ...
```

报文流向：
从 10.9.0.5 发向 192.169.60.5 的报文经过路由配置由网卡 tun1 发送到 10.9.0.11，然后由于 10.9.0.11 开启了路由转发，报文被发送到 192.168.60.5，应答报文发送到路由器后由于目的地址 192.168.53.99 与网卡 tun10 地址 192.168.53.66 处于同一网段，报文被转发到网卡 tun10，再由 tun10 将其发送到 10.9.0.5，由网卡 tun0 解析，整个过程结束。

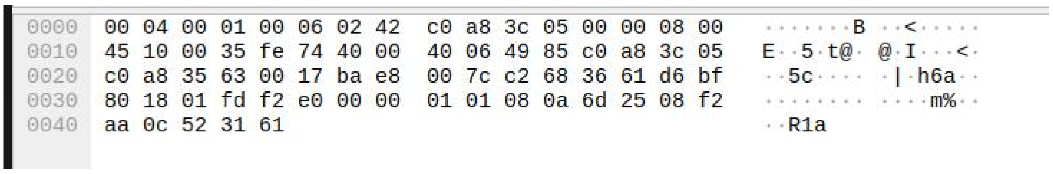
Task6

Telnet 连接后终止 server，会无法输入内容，这时候重新启动 server 仍能显示刚才键入内容，

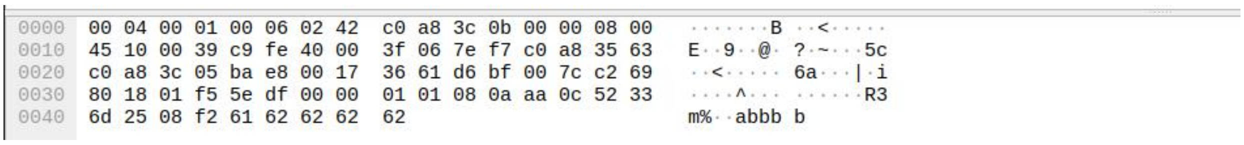


```
seed@2df1a5e54cd2:~$ aabbbb

^CTraceback (most recent call last):
  File "tun_server.py", line 36, in <module>
    ready, _, _ = select.select([sock, tun], [], [])
KeyboardInterrupt

root@777133b5df2a:/volumes# python3 tun_server.py
Interface Name: tun10
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
```

查看 wireshark，发现 tcp 会持续重发包含第一个字符的报文，当该报文被收到后直接将后续所有键入内容放在一个报文中发送，如图所示



```
0000  00 04 00 01 00 06 02 42  c0 a8 3c 05 00 00 08 00   ·······B ··<·····
0010  45 10 00 35 fe 74 40 00  40 06 49 85 c0 a8 3c 05   E··5·t@· @·I···<·
0020  c0 a8 35 63 00 17 ba e8  00 7c c2 68 36 61 d6 bf   ··5c···· ·|·h6a··
0030  80 18 01 fd f2 e0 00 00  01 01 08 0a 6d 25 08 f2   ········ ····m%··
0040  aa 0c 52 31 61                                     ··R1a
```



```
0000  00 04 00 01 00 06 02 42  c0 a8 3c 0b 00 00 08 00   ·······B ··<·····
0010  45 10 00 39 c9 fe 40 00  3f 06 7e f7 c0 a8 35 63   E··9··@· ?·~···5c
0020  c0 a8 3c 05 ba e8 00 17  36 61 d6 bf 00 7c c2 69   ··<····· 6a···|·i
0030  80 18 01 f5 5e df 00 00  01 01 08 0a aa 0c 52 33   ····^··· ······R3
0040  6d 25 08 f2 61 62 62 62  62                        m%··abbb b
```