南京大学本科生实验报告

课程名称：**计算机网络**　　任课教师：田臣/李文中　　助教：lzh、lsp、wcx

学院	**计算机科学与技术系**	专业（方向）	计算机科学与技术系
学号	211220027	姓名	王秋博
Email	211220027@nju.edu.cn	开始/完成日期	11.9/11.23

1. 实验名称

Lab5: **Respond to ICMP**

2. 实验目的

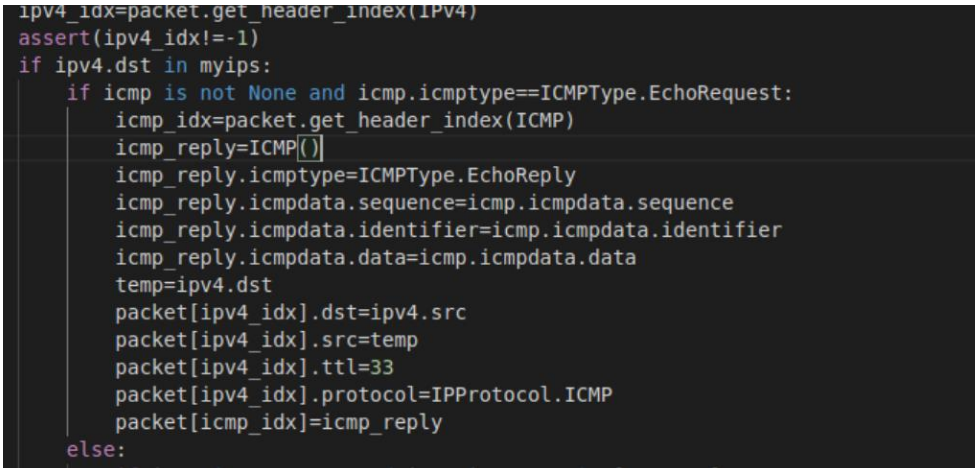
响应 ICMP 消息，如回显请求（“ping”），必要时生成 ICMP 错误消息。

3. 实验内容

Task 2：**IP Forwarding Table Lookup**

Step 1: Coding

响应 ICMP 回显请求的逻辑如图：

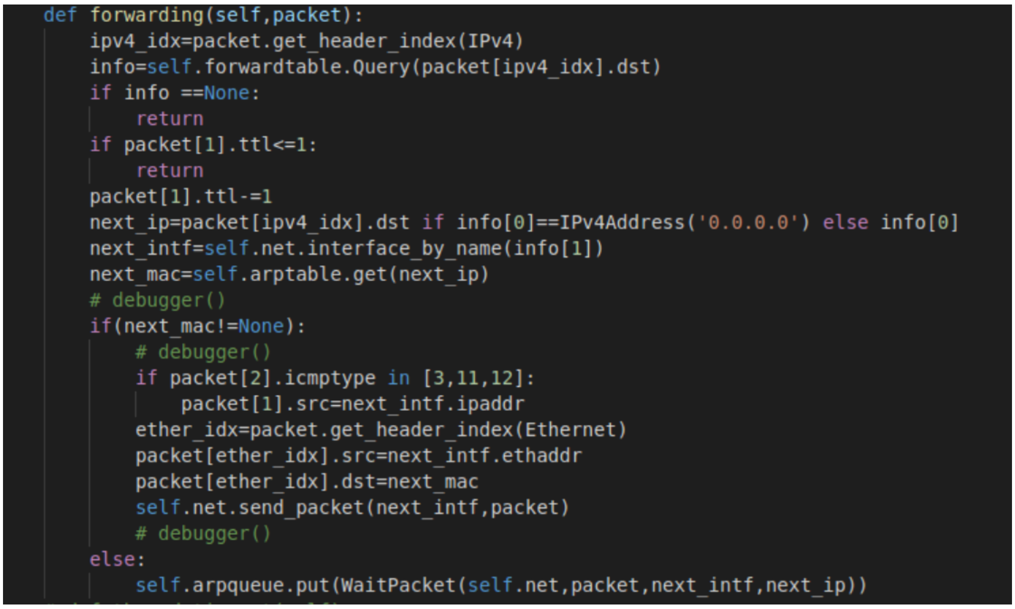


```
ipv4_idx=packet.get_header_index(IPv4)
assert(ipv4_idx!=-1)
if ipv4.dst in myips:
    if icmp is not None and icmp.icmptype==ICMPType.EchoRequest:
        icmp_idx=packet.get_header_index(ICMP)
        icmp_reply=ICMP()
        icmp_reply.icmptype=ICMPType.EchoReply
        icmp_reply.icmpdata.sequence=icmp.icmpdata.sequence
        icmp_reply.icmpdata.identifier=icmp.icmpdata.identifier
        icmp_reply.icmpdata.data=icmp.icmpdata.data
        temp=ipv4.dst
        packet[ipv4_idx].dst=ipv4.src
        packet[ipv4_idx].src=temp
        packet[ipv4_idx].ttl=33
        packet[ipv4_idx].protocol=IPProtocol.ICMP
        packet[icmp_idx]=icmp_reply
    else:
```

先设置 icmptype，再依次将序列号、标识符、数据字段从请求复制到回复中
再构建 ip 标头，src 和 dst 即分别为 request 的 dst 和 src，ttl 设置>=32

发送我所构建的数据包这部分整合在下图的一个函数中，转发逻辑大致与 lab4 当中一样，从构建完数据包到转发数据包判断了几种错误情况，它们也调用了这个转发函数。故防止构建的错误包仍然出现错误，故前面增加了判断。

且由于错误消息转发是 ip.src 应为转发表查找接口的 ipaddr，而 reply 不是，故，修改

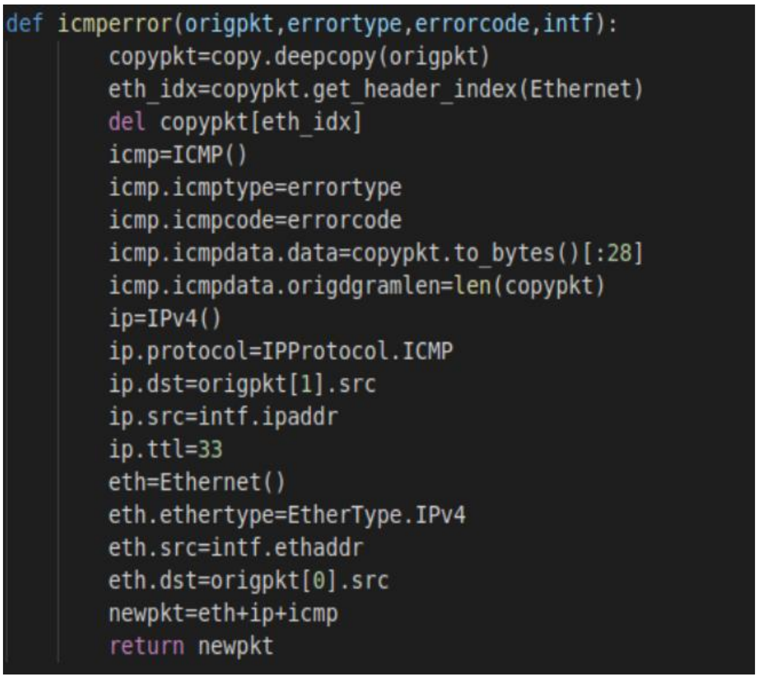


```
def forwarding(self,packet):
    ipv4_idx=packet.get_header_index(IPv4)
    info=self.forwardtable.Query(packet[ipv4_idx].dst)
    if info ==None:
        return
    if packet[1].ttl<=1:
        return
    packet[1].ttl-=1
    next_ip=packet[ipv4_idx].dst if info[0]==IPv4Address('0.0.0.0') else info[0]
    next_intf=self.net.interface_by_name(info[1])
    next_mac=self.arptable.get(next_ip)
    # debugger()
    if(next_mac!=None):
        # debugger()
        if packet[2].icmptype in [3,11,12]:
            packet[1].src=next_intf.ipaddr
        ether_idx=packet.get_header_index(Ethernet)
        packet[ether_idx].src=next_intf.ethaddr
        packet[ether_idx].dst=next_mac
        self.net.send_packet(next_intf,packet)
        # debugger()
    else:
        self.arpqueue.put(WaitPacket(self.net,packet,next_intf,next_ip))
```

Task 3：**Generating ICMP error messages**

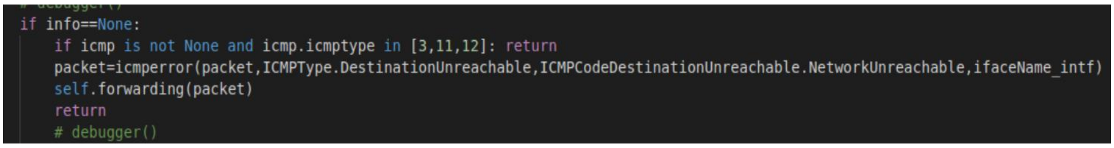
Step 1: Coding

构建错误消息：ip.src 在需在转发是根据转发表查找的接口 ip 修改，此处 intf 其实只主要影响 eth 的 src，后续均传入 ifacename 对应 intf



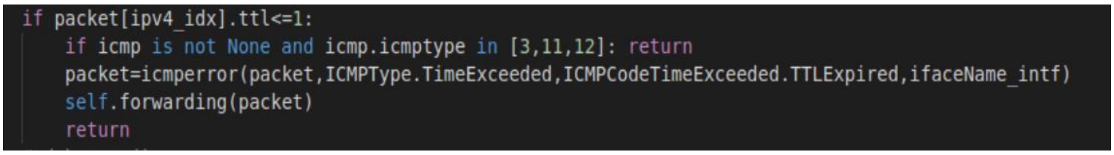
```
def icmperror(origpkt,errortype,errorcode,intf):
        copypkt=copy.deepcopy(origpkt)
        eth_idx=copypkt.get_header_index(Ethernet)
        del copypkt[eth_idx]
        icmp=ICMP()
        icmp.icmptype=errortype
        icmp.icmpcode=errorcode
        icmp.icmpdata.data=copypkt.to_bytes()[:28]
        icmp.icmpdata.origdgramlen=len(copypkt)
        ip=IPv4()
        ip.protocol=IPProtocol.ICMP
        ip.dst=origpkt[1].src
        ip.src=intf.ipaddr
        ip.ttl=33
        eth=Ethernet()
        eth.ethertype=EtherType.IPv4
        eth.src=intf.ethaddr
        eth.dst=origpkt[0].src
        newpkt=eth+ip+icmp
        return newpkt
```

Error1：没有匹配的条目：



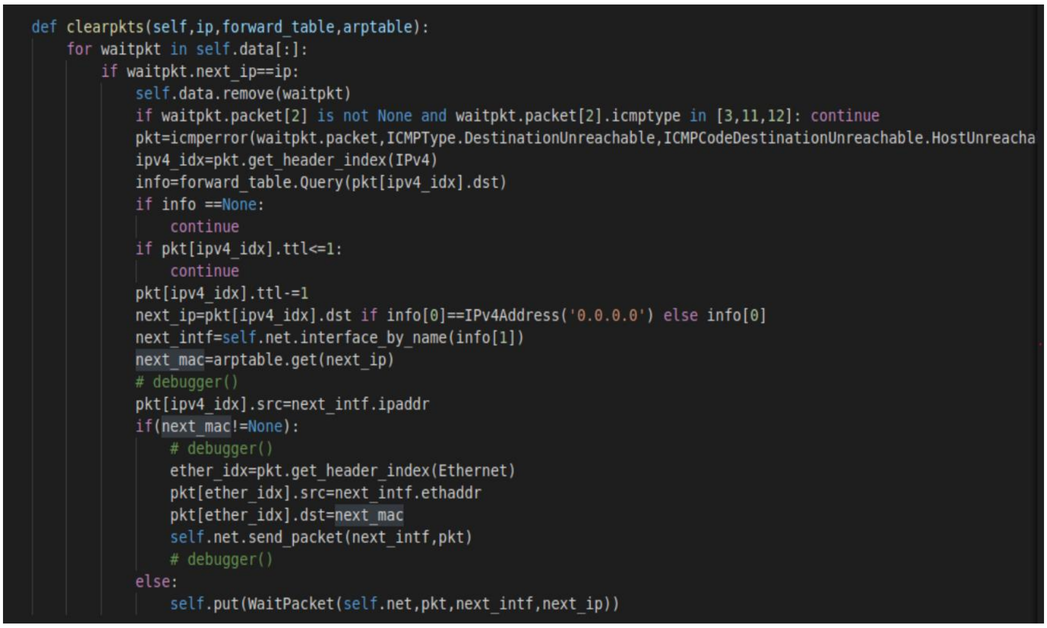
```
if info==None:
    if icmp is not None and icmp.icmptype in [3,11,12]: return
    packet=icmperror(packet,ICMPType.DestinationUnreachable,ICMPCodeDestinationUnreachable.NetworkUnreachable,ifaceName_intf)
    self.forwarding(packet)
    return
    # debugger()
```

Error2：TTL 过期：



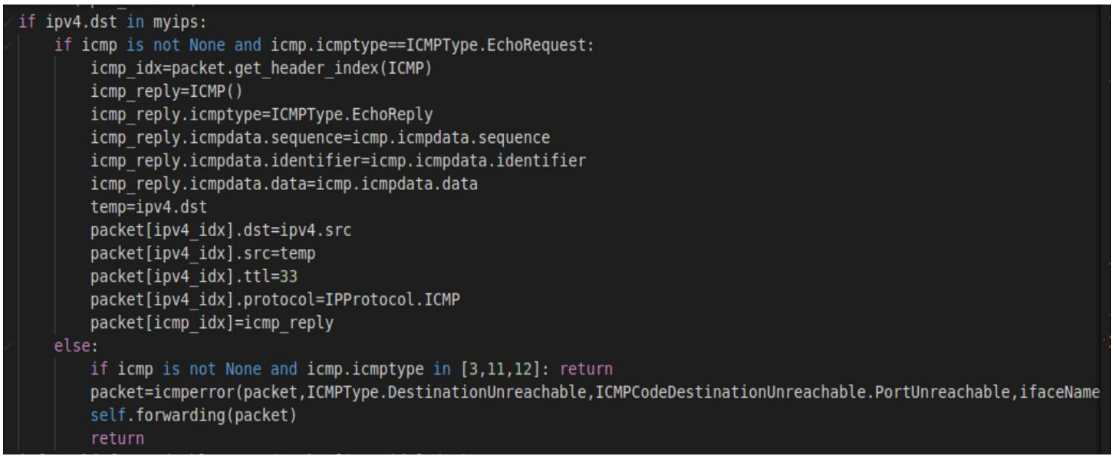
```
if packet[ipv4_idx].ttl<=1:
    if icmp is not None and icmp.icmptype in [3,11,12]: return
    packet=icmperror(packet,ICMPType.TimeExceeded,ICMPCodeTimeExceeded.TTLExpired,ifaceName_intf)
    self.forwarding(packet)
    return
```

Error3：ARP 请求失败：逻辑与 forwarding 基本一样，因类的问题复制了一遍



```
def clearpkts(self,ip,forward_table,arptable):
    for waitpkt in self.data[:]:
        if waitpkt.next_ip==ip:
            self.data.remove(waitpkt)
            if waitpkt.packet[2] is not None and waitpkt.packet[2].icmptype in [3,11,12]: continue
            pkt=icmperror(waitpkt.packet,ICMPType.DestinationUnreachable,ICMPCodeDestinationUnreachable.HostUnreacha
            ipv4_idx=pkt.get_header_index(IPv4)
            info=forward_table.Query(pkt[ipv4_idx].dst)
            if info ==None:
                continue
            if pkt[ipv4_idx].ttl<=1:
                continue
            pkt[ipv4_idx].ttl-=1
            next_ip=pkt[ipv4_idx].dst if info[0]==IPv4Address('0.0.0.0') else info[0]
            next_intf=self.net.interface_by_name(info[1])
            next_mac=arptable.get(next_ip)
            # debugger()
            pkt[ipv4_idx].src=next_intf.ipaddr
            if(next_mac!=None):
                # debugger()
                ether_idx=pkt.get_header_index(Ethernet)
                pkt[ether_idx].src=next_intf.ethaddr
                pkt[ether_idx].dst=next_mac
                self.net.send_packet(next_intf,pkt)
                # debugger()
            else:
                self.put(WaitPacket(self.net,pkt,next_intf,next_ip))
```

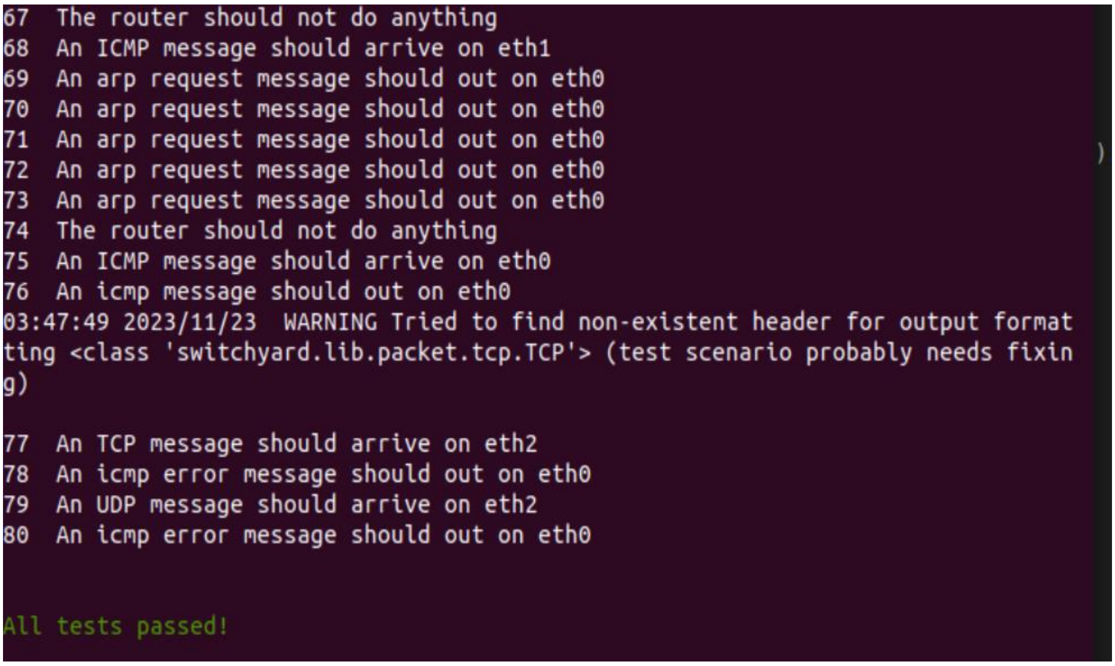
Error4：不支持的功能：



```
if ipv4.dst in myips:
    if icmp is not None and icmp.icmptype==ICMPType.EchoRequest:
        icmp_idx=packet.get_header_index(ICMP)
        icmp_reply=ICMP()
        icmp_reply.icmptype=ICMPType.EchoReply
        icmp_reply.icmpdata.sequence=icmp.icmpdata.sequence
        icmp_reply.icmpdata.identifier=icmp.icmpdata.identifier
        icmp_reply.icmpdata.data=icmp.icmpdata.data
        temp=ipv4.dst
        packet[ipv4_idx].dst=ipv4.src
        packet[ipv4_idx].src=temp
        packet[ipv4_idx].ttl=33
        packet[ipv4_idx].protocol=IPProtocol.ICMP
        packet[icmp_idx]=icmp_reply
    else:
        if icmp is not None and icmp.icmptype in [3,11,12]: return
        packet=icmperror(packet,ICMPType.DestinationUnreachable,ICMPCodeDestinationUnreachable.PortUnreachable,ifaceName
        self.forwarding(packet)
        return
```

关于为什么不响应错误消息：可能造成连锁的错误，如果路由器对其他 ICMP 错误消息生成 ICMP 错误消息，可能导致一个连锁反应，其中路径中的每个路由器都会响应前一个路由器生成的 ICMP 错误，从而导致无休止的循环。

Step 2: Testing



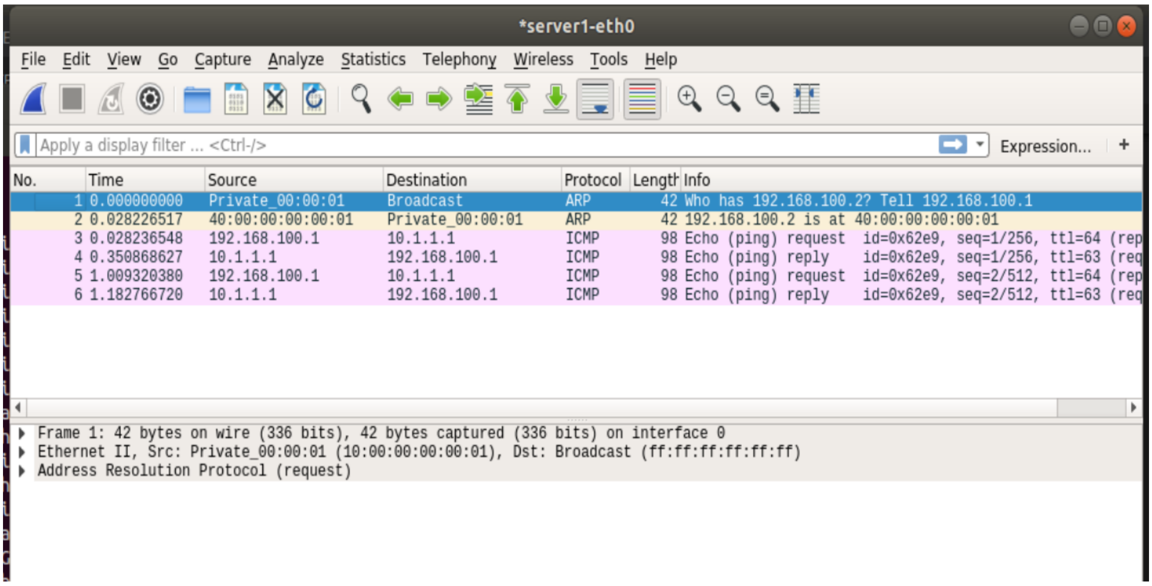
```
67  The router should not do anything
68  An ICMP message should arrive on eth1
69  An arp request message should out on eth0
70  An arp request message should out on eth0
71  An arp request message should out on eth0
72  An arp request message should out on eth0
73  An arp request message should out on eth0
74  The router should not do anything
75  An ICMP message should arrive on eth0
76  An icmp message should out on eth0
03:47:49 2023/11/23  WARNING Tried to find non-existent header for output format
ting <class 'switchyard.lib.packet.tcp.TCP'> (test scenario probably needs fixin
g)

77  An TCP message should arrive on eth2
78  An icmp error message should out on eth0
79  An UDP message should arrive on eth2
80  An icmp error message should out on eth0


All tests passed!
```

Step 3: Deploying

Server1 ping -c2 client

监听 server1 的 eth0 接口

结果如下：



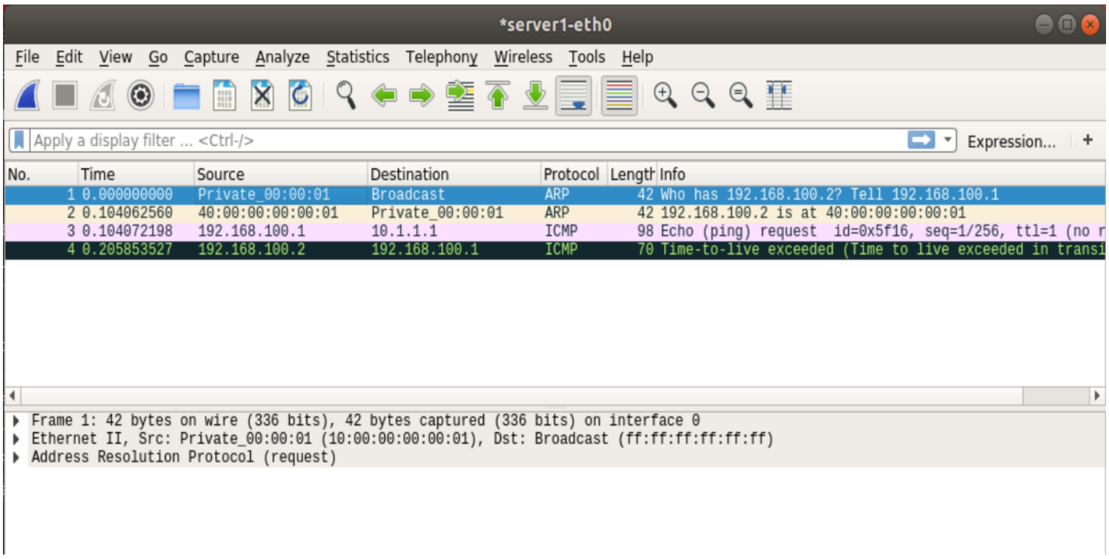
*server1-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.   Time         Source              Destination        Protocol Length Info
    1 0.000000000  Private_00:00:01    Broadcast          ARP      42 Who has 192.168.100.2? Tell 192.168.100.1
    2 0.028226517  40:00:00:00:00:01   Private_00:00:01   ARP      42 192.168.100.2 is at 40:00:00:00:00:01
    3 0.028236548  192.168.100.1       10.1.1.1           ICMP     98 Echo (ping) request  id=0x62e9, seq=1/256, ttl=64 (rep
    4 0.350868627  10.1.1.1            192.168.100.1      ICMP     98 Echo (ping) reply    id=0x62e9, seq=1/256, ttl=63 (req
    5 1.009320380  192.168.100.1       10.1.1.1           ICMP     98 Echo (ping) request  id=0x62e9, seq=2/512, ttl=64 (rep
    6 1.182766720  10.1.1.1            192.168.100.1      ICMP     98 Echo (ping) reply    id=0x62e9, seq=2/512, ttl=63 (req

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

server1 ping -c1 -t 1 client

监听 server1 的 eth0 接口

结果如下（ttl 过期）：


*server1-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.104062560	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
3	0.104072198	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x5f16, seq=1/256, ttl=1 (no r
4	0.205853527	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transi

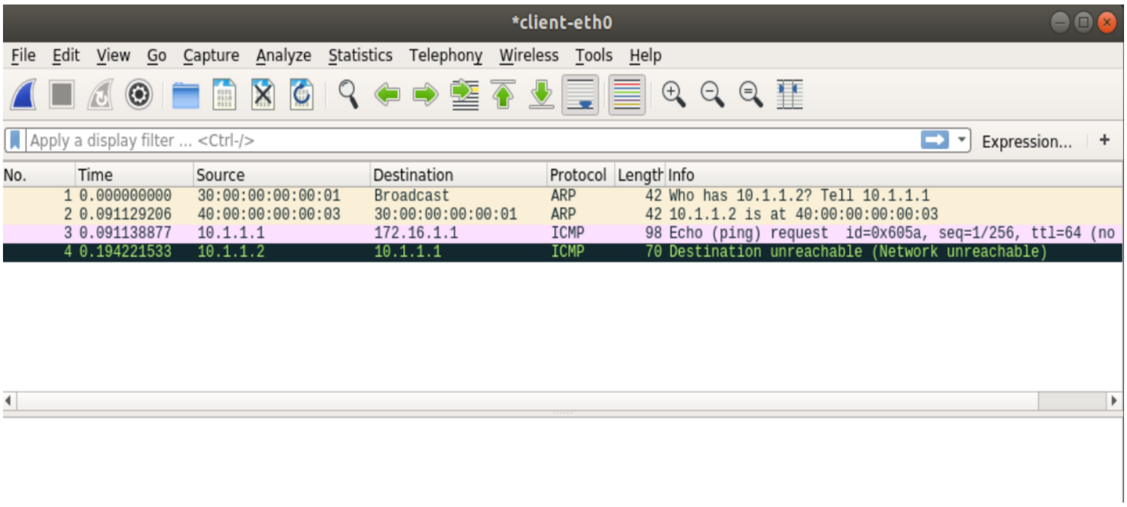
```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

client ping -c1 172.16.1.1

监听 client 的 eth0 接口

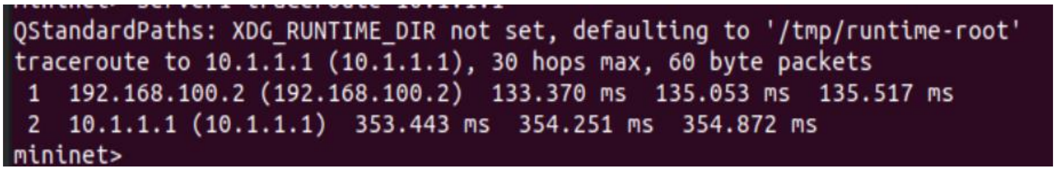
结果如下（目标不可达）：

tr

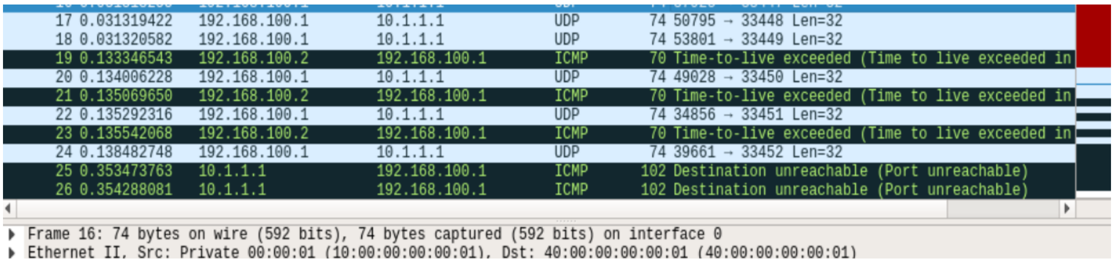

*client-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	30:00:00:00:00:01	Broadcast	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
2	0.091129206	40:00:00:00:00:03	30:00:00:00:00:01	ARP	42	10.1.1.2 is at 40:00:00:00:00:03
3	0.091138877	10.1.1.1	172.16.1.1	ICMP	98	Echo (ping) request id=0x605a, seq=1/256, ttl=64 (no
4	0.194221533	10.1.1.2	10.1.1.1	ICMP	70	Destination unreachable (Network unreachable)

server1 traceroute 10.1.1.1



```
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
traceroute to 10.1.1.1 (10.1.1.1), 30 hops max, 60 byte packets
 1  192.168.100.2 (192.168.100.2)  133.370 ms  135.053 ms  135.517 ms
 2  10.1.1.1 (10.1.1.1)  353.443 ms  354.251 ms  354.872 ms
mininet>
```



```
17 0.031319422  192.168.100.1  10.1.1.1       UDP    74 50795 → 33448 Len=32
18 0.031320582  192.168.100.1  10.1.1.1       UDP    74 53801 → 33449 Len=32
19 0.133346543  192.168.100.2  192.168.100.1  ICMP   70 Time-to-live exceeded (Time to live exceeded in
20 0.134006228  192.168.100.1  10.1.1.1       UDP    74 49028 → 33450 Len=32
21 0.135069650  192.168.100.2  192.168.100.1  ICMP   70 Time-to-live exceeded (Time to live exceeded in
22 0.135292316  192.168.100.1  10.1.1.1       UDP    74 34856 → 33451 Len=32
23 0.135542068  192.168.100.2  192.168.100.1  ICMP   70 Time-to-live exceeded (Time to live exceeded in
24 0.138482748  192.168.100.1  10.1.1.1       UDP    74 39661 → 33452 Len=32
25 0.353473763  10.1.1.1       192.168.100.1  ICMP  102 Destination unreachable (Port unreachable)
26 0.354288081  10.1.1.1       192.168.100.1  ICMP  102 Destination unreachable (Port unreachable)
```

```
▸ Frame 16: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: Private 00:00:01 (10:00:00:00:00:01), Dst: 40:00:00:00:00:01 (40:00:00:00:00:01)
```

Traceroute 的实现借助了 TTL：通过向目的地址发送一系列的探测包，设置探测包的 TTL 初始值分别为 1,2,3…，根据返回的超时通知得到源地址与目的地址之间的每一跳路由信息

4. 实验结果

本节实验结果基本于实验过程中阐述，不再赘述

5. 核心代码

同实验结果

6. 总结与感想

比 lab4 好很多，小卡