Computer Networks: lab05

Router3（ICMP Echo Reply & ICMP Error Message）

任课教师：田臣　　助教：方毓楚，郑浩等

学院	计算机科学与技术系	专业	计算机科学与技术
学号	201220154	姓名	王紫萁
Email	201220154@smail.nju.edu.cn	开始/完成时间	2022. 4.21/2022.4.25

1. 实验目的

在之前的实验中， 我们实现了路由器的绝大部分基础功能，但是并没有特别处理一些路由器无法处理或者数据报出错的情况。于是本次实验中，我们为路由器增加对发送给自己端口的 `ICMP Request` 数据报以及针对错误情况的响应。

2. 构造ICMP Reply

在lab04中我们针对目的IP不是路由器端口的数据报做了查询和转发处理，而对于发送给路由器端口之一的数据报直接丢弃了。实际情况是，路由器唯一能接受并处理的数据报是 `ICMP Request` 报文，通常用于检查主机和路由器之间的连接情况，并获取路由器的端口信息。当路由器的其中一个端口收到发送给自己的数据报后，它会构造一个ICMP回复报文，其中 `sequence number` 以及 `identifier` ， `data` 段都是拷贝的 `ICMP Request` 的对应段，目的地址 `IP/MAC` 设置为请求报文的源地址，发送给哪个端口，哪个端口就充当源地址

构造的函数是 `ICMPPackets.py/create_icmp_EchoReply` ，它会根据ICMP请求报文packet，源地址（接收端口）返回一个完整的 `ICMP Request` 报文，函数接口如下（具体实现就是将上面的赋值逻辑转换成代码即可，在报告中不做赘述，需要注意 `icmpcode` 的类型为ICMP code echo reply的Echo reply）：

```
def create_icmp_EchoReply(packet:Packet, srcMAC:EthAddr, srcIP:IPAddr)
```

3. 构造ICMP ERROR Message

我们的路由器处理四种形式的错误，分别会在以下情况下发送 `Error Message`

- 子网不可达：在转发表中没有相应表项，发送 `Destination Network Unreachable` 错误
- TTL exceeded：在查找完转发表后跳数耗尽，发送 `Time To Live exceeded` 错误
- 主机不可达：在路由器尝试向目的IP发送5次ARP请求后没有收到 `ARP Reply` 说明在该子网中目的IP不可达，于是发送 `Destination Host Unreachable` 错误

- 端口不可达：发给路由器端口的数据报只能是 `ICMP Echo Request` 类型的数据报，对于其他数据报路由器没有能力处理因此需要发送 `Dstination Port Unreachable` 错误

构造的函数是 `ICMPPackets.py/create_icmp_ErrorMsg` 函数，它会接收两个参数：枚举类型 `ICMPErrorType` 指明需要构造哪一种类型的错误信息，以及发生错误的数据报 `packet`

函数通过不同的类型赋予不同的错误类型，需要注意的是ICMP报文的 `payload` 字段只不包含出错包的Ethernet字段的信息因此需要删除，而该函数也只返回合适的IPv4报头和ICMP报头的组合，在端口发送数据报时会自动为缺少Ethernet报头的数据报添加符合发送端口信息的报头

```python

class ICMPErrorType(Enum):
    NetworkUnreachable = 0
    TimeLimitExceeded = 1
    HostUnreachable = 2
    PortUnreachable = 3


def create_icmp_ErrorMsg(errortype: ICMPErrorType, packet: Packet):
    i = packet.get_header_index(Ethernet)
    del packet[i]
    replyICMP = ICMP()
    if errortype == ICMPErrorType.NetworkUnreachable:
        replyICMP.icmptype = ICMPType.DestinationUnreachable
        replyICMP.icmpcode = ICMPCodeDestinationUnreachable.NetworkUnreachable
    elif errortype == ICMPErrorType.TimeLimitExceeded:
        replyICMP.icmptype = ICMPType.TimeExceeded
        replyICMP.icmpcode = ICMPCodeTimeExceeded.TTLExpired
    elif errortype == ICMPErrorType.HostUnreachable:
        replyICMP.icmptype = ICMPType.DestinationUnreachable
        replyICMP.icmpcode = ICMPCodeDestinationUnreachable.HostUnreachable
    elif errortype == ICMPErrorType.PortUnreachable:
        replyICMP.icmptype = ICMPType.DestinationUnreachable
        replyICMP.icmpcode = ICMPCodeDestinationUnreachable.PortUnreachable
    replyICMP.icmpdata.data = packet.to_bytes()[:28]
    replyIPv4 = IPv4()
    replyIPv4.protocol = IPProtocol.ICMP
    replyIPv4.src = packet.get_header(IPv4).dst
    replyIPv4.dst = packet.get_header(IPv4).src
    replyIPv4.ttl = 63

    return replyIPv4 + replyICMP
```

4.在正确的地方发送ICMP报文

有了第2，3节的两个工具后我们就可以在需要发送相应类型的ICMP报文的地方构造正确的数据报并由路由器负责决定将这个ICMP数据报加入到哪个端口等待队列中（lab03中实现）

4.1. Echo Reply & Port Unreachable

```python
elif packet.has_header(IPv4):  # process other packets
    dstIP = packet.get_header(IPv4).dst
    if dstIP in self._DictIpIntf.keys():    # destinate to one of the interface
        if packet.has_header(ICMP):
            icmp = packet.get_header(ICMP)
            if icmp.icmptype == ICMPType.EchoRequest:   #need to send Echo Reply
                replyPkt = create_icmp_EchoReply(
                    packet=packet,
                    srcMAC=self._DictIpIntf[dstIP].ethaddr,
                    srcIP=dstIP)
                self.forward_IPv4(replyPkt) #send IPv4 packet
            else:  # got ICMP echo reply, router shouldn't handle this
                wrongType = create_icmp_ErrorMsg(ICMPErrorType.PortUnreachable, packet) # generate port unreachable error message
                self.forward_IPv4(wrongType)
        else:  # not ICMP, shouldn't handle this, either
            wrongType = create_icmp_ErrorMsg(ICMPErrorType.PortUnreachable, packet)
            self.forward_IPv4(wrongType)
    else:  # packet is not for the router interfaces
        self.forward_IPv4(packet)
```

在处理IPv4数据报的分支中，对于发给自己端口的 `Echo Request` 数据报，我们将该端口的地址作为源地址并发送 `Echo Reply` 。而对于发给路由器的其他类型的IPv4数据报我们返回一个Port Unreachable的message

4.2. TTL Exceeded & Network Unreachable

```python
def forward_IPv4(self, packet: Packet):
    dstIP = packet.get_header(IPv4).dst
    entry = self._fwTable.lookUp(dstIP=dstIP)
    packet.get_header(IPv4).ttl -= 1  # ttl should be decreased after look-up
    if entry is not None and packet.get_header(IPv4).ttl > 0:
        #ready to forward
        '''
            this branch adds packet to an appropriate interface queue and
            decides whether to send ARP Request, look up lab4 for more details
        '''
    elif entry is None: # can't find matched entry in the forwarding table
        icmpUnreachable = create_icmp_ErrorMsg(ICMPErrorType.NetworkUnreachable, packet)
        self.forward_IPv4(icmpUnreachable)
    else:  # ttl == 0
        # ttl exceeded, forward ttl-exceeded message
        icmpTTL = create_icmp_ErrorMsg(ICMPErrorType.TimeLimitExceeded, packet)
        self.forward_IPv4(icmpTTL)
```

当查找转发表项失败时，需要发送NetWork Unreachable的信息；当发现跳数耗尽时发送TTL Exceeded的数据报

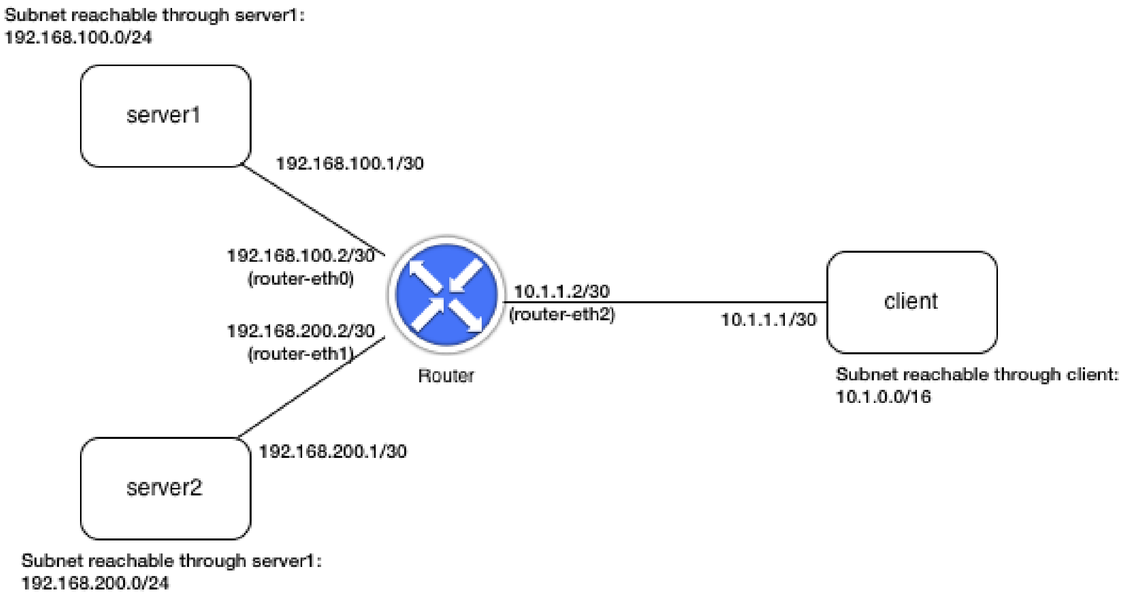
4.3.Host Unreachable

```
def update(self):
    # update ARP cache and interface queues
    self._ARPcache.update()
    for queue in self._INTFQueues.values():
        noHostList = queue.update_all_request()
        self.handleNoHostPackets(noHostList)

def handleNoHostPackets(self, noHostList):
    # send host-unreachable icmp packet for every packet
    # that raised no-host error to the destination demonstrated
    # in srcIP/MAC segment in the error packet
    for packetList in noHostList:
        for packet in packetList:
            icmpHostUnreachable =
create_icmp_ErrorMsg(ICMPErrorType.HostUnreachable, packet)
            self.forward_IPv4(icmpHostUnreachable)
```

每一个端口等待队列在发现目标IP不可达时，会将所有包的信息打包成一个废旧包列表，并将该列表发送给隶属的端口等待队列组。在路由器 `update` 时（包括更新缓存表时间戳，以及让每个未收到ARP Reply的端口再次发送ARP请求），如果收到了队列组打包的废旧包列表组，就按照列表组中的顺序为每一个包发送 `Host Unreachable` 的错误信息给源端

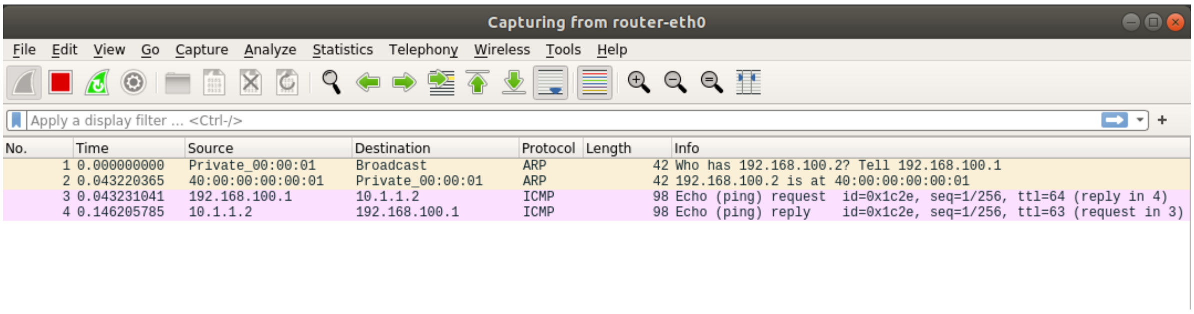
5. wireshark抓包测试



在本次实验测试中，我们以 `server1` 为发包的源头并观察 `router-eth0` 端口的收发包情况

5.1. ICMP Echo Reply

```
$server1> ping -c 1 10.1.1.2
```


Capturing from router-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

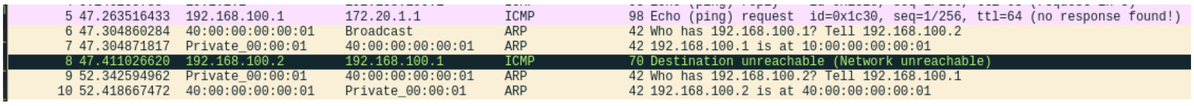
```
No.  Time          Source             Destination        Protocol Length Info
   1 0.000000000   Private_00:00:01   Broadcast          ARP          42 Who has 192.168.100.2? Tell 192.168.100.1
   2 0.043220365   40:00:00:00:00:01  Private_00:00:01   ARP          42 192.168.100.2 is at 40:00:00:00:00:01
   3 0.043231041   192.168.100.1      10.1.1.2           ICMP         98 Echo (ping) request  id=0x1c2e, seq=1/256, ttl=64 (reply in 4)
   4 0.146205785   10.1.1.2           192.168.100.1      ICMP         98 Echo (ping) reply    id=0x1c2e, seq=1/256, ttl=63 (request in 3)
```

我们观察到， `server1` 发送给 `eth2` 端口的ICMP请求报文（3号包）在4号包得到回应

5.2. ICMP Network Unreachable

我们为server1添加了一条172.20.0.0/16的路由，但是在路由表中并没有和该子网的转发表项，因此在执行下面命令后，会无法收到ICMP回复，并且由路由器回复 `ICMP NetWork Unreachable` 的错误信息

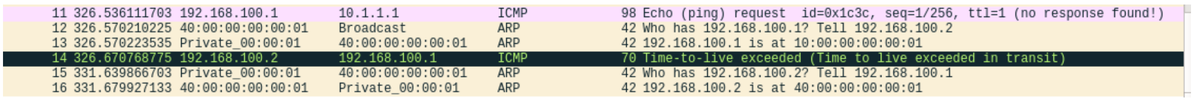
```
$server1> ping -c 1 172.20.1.1
```



```
 5 47.263516433  192.168.100.1      172.20.1.1         ICMP     98 Echo (ping) request  id=0x1c30, seq=1/256, ttl=64 (no response found!)
 6 47.304860284  40:00:00:00:00:01  Broadcast          ARP      42 Who has 192.168.100.1? Tell 192.168.100.2
 7 47.304871817  Private_00:00:01   40:00:00:00:00:01  ARP      42 192.168.100.1 is at 10:00:00:00:00:01
 8 47.411026620  192.168.100.2      192.168.100.1      ICMP     70 Destination unreachable (Network unreachable)
 9 52.342594962  Private_00:00:01   40:00:00:00:00:01  ARP      42 Who has 192.168.100.2? Tell 192.168.100.1
10 52.418667472  40:00:00:00:00:01  Private_00:00:01   ARP      42 192.168.100.2 is at 40:00:00:00:00:01
```

5.3. ICMP Time To Live Exceeded

```
$server1> ping -c 1 -t 1 10.1.1.1
```



```
11 326.536111703 192.168.100.1      10.1.1.1           ICMP     98 Echo (ping) request  id=0x1c3c, seq=1/256, ttl=1 (no response found!)
12 326.570210225 40:00:00:00:00:01  Broadcast          ARP      42 Who has 192.168.100.1? Tell 192.168.100.2
13 326.570223535 Private_00:00:01   40:00:00:00:00:01  ARP      42 192.168.100.1 is at 10:00:00:00:00:01
14 326.670768775 192.168.100.2      192.168.100.1      ICMP     70 Time-to-live exceeded (Time to live exceeded in transit)
15 331.639866703 Private_00:00:01   40:00:00:00:00:01  ARP      42 Who has 192.168.100.2? Tell 192.168.100.1
16 331.679927133 40:00:00:00:00:01  Private_00:00:01   ARP      42 192.168.100.2 is at 40:00:00:00:00:01
```

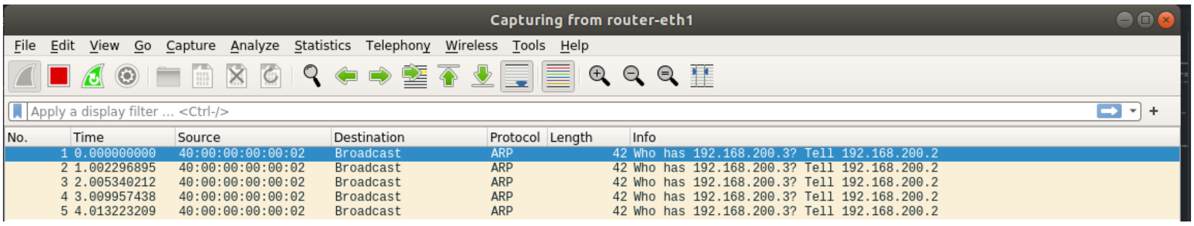
构建一个给 `client` 的但是只有一条的ICMP请求，于是 `router-eth0` 需要发送 `ttl exceeded` 错误信息给到 `server1`

5.4 ICMP Host Unreachable

```
$server1> ping -c 1 192.168.200.3
```

router-eth1抓包情况

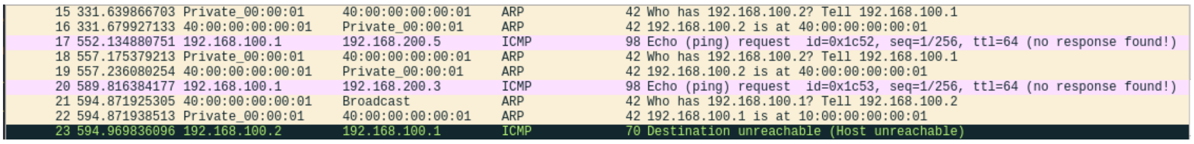
（后来单独测试Host Unreachable时想要找到路由器发送的5次ARP请求，因此之前eth1端口的收发包情况并没有记录）



Capturing from router-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time          Source              Destination   Protocol Length  Info
    1 0.000000000   40:00:00:00:00:02   Broadcast     ARP        42 Who has 192.168.200.3? Tell 192.168.200.2
    2 1.002296895   40:00:00:00:00:02   Broadcast     ARP        42 Who has 192.168.200.3? Tell 192.168.200.2
    3 2.005340212   40:00:00:00:00:02   Broadcast     ARP        42 Who has 192.168.200.3? Tell 192.168.200.2
    4 3.009957438   40:00:00:00:00:02   Broadcast     ARP        42 Who has 192.168.200.3? Tell 192.168.200.2
    5 4.013223209   40:00:00:00:00:02   Broadcast     ARP        42 Who has 192.168.200.3? Tell 192.168.200.2
```

router-eth0抓包情况



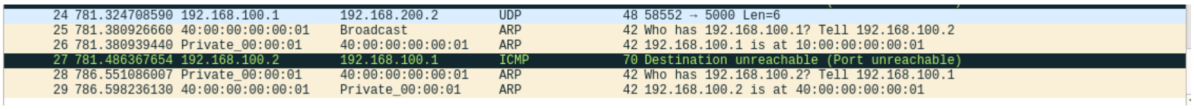
```
   15 331.639866703 Private_00:00:01   40:00:00:00:00:01   ARP    42 Who has 192.168.100.2? Tell 192.168.100.1
   16 331.679927133 40:00:00:00:00:01  Private_00:00:01    ARP    42 192.168.100.2 is at 40:00:00:00:00:01
   17 552.134880751 192.168.100.1      192.168.200.5       ICMP   98 Echo (ping) request  id=0x1c52, seq=1/256, ttl=64 (no response found!)
   18 557.175379213 Private_00:00:01   40:00:00:00:00:01   ARP    42 Who has 192.168.100.2? Tell 192.168.100.1
   19 557.236080254 40:00:00:00:00:01  Private_00:00:01    ARP    42 192.168.100.2 is at 40:00:00:00:00:01
   20 589.816384177 192.168.100.1      192.168.200.3       ICMP   98 Echo (ping) request  id=0x1c53, seq=1/256, ttl=64 (no response found!)
   21 594.871925305 40:00:00:00:00:01  Broadcast           ARP    42 Who has 192.168.100.1? Tell 192.168.100.2
   22 594.871938513 Private_00:00:01   40:00:00:00:00:01   ARP    42 192.168.100.1 is at 10:00:00:00:00:01
   23 594.969836096 192.168.100.2      192.168.100.1       ICMP   70 Destination unreachable (Host unreachable)
```

（在此之前ping过192.168.200.5）

尝试给 `192.168.100.3` 发送5次ARP请求，由于该子网下并没有该主机，于是会给server1发送 `Host Unreachable` 错误信息

5.5. ICMP Port Unreachable

```
$server1> echo "hello" | socat - udp4-datagram:192.168.200.2:5000
```

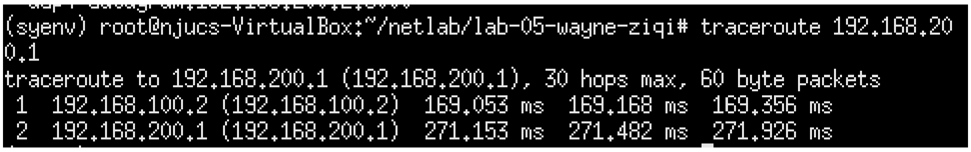


```
   24 781.324708590 192.168.100.1      192.168.200.2       UDP    48 58552 → 5000 Len=6
   25 781.380926660 40:00:00:00:00:01  Broadcast           ARP    42 Who has 192.168.100.1? Tell 192.168.100.2
   26 781.380939440 Private_00:00:01   40:00:00:00:00:01   ARP    42 192.168.100.1 is at 10:00:00:00:00:01
   27 781.486367654 192.168.100.2      192.168.100.1       ICMP   70 Destination unreachable (Port unreachable)
   28 786.551086007 Private_00:00:01   40:00:00:00:00:01   ARP    42 Who has 192.168.100.2? Tell 192.168.100.1
   29 786.598236130 40:00:00:00:00:01  Private_00:00:01    ARP    42 192.168.100.2 is at 40:00:00:00:00:01
```

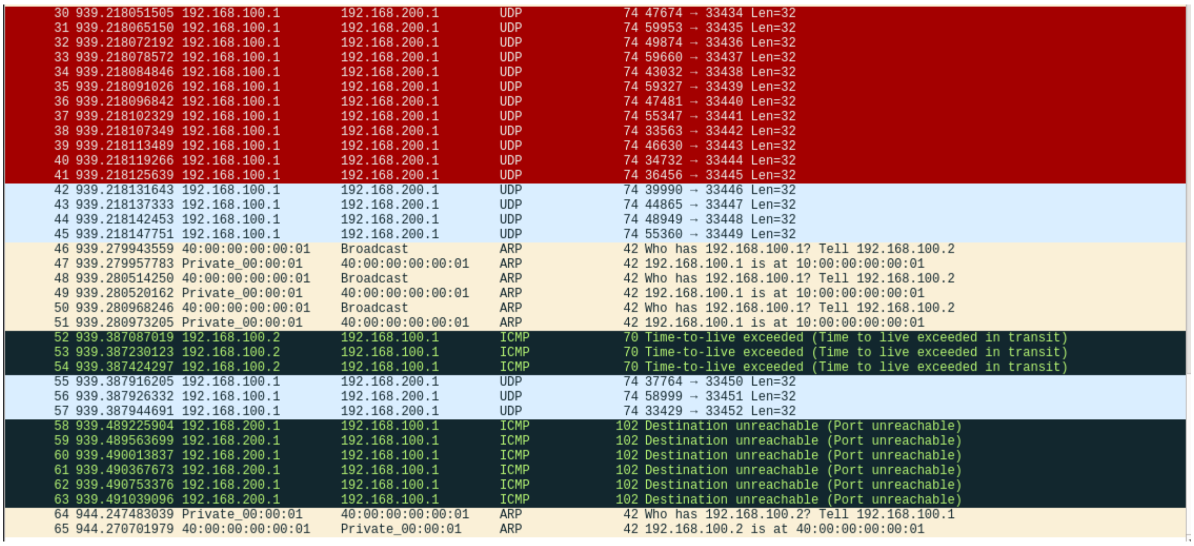
构建一个指向192.168.200.2路由器端口的UDP报文，由于路由器只能处理Echo Request报文因此会发给server1 `Port Unreachable` 的信息

5.6. traceroute

```
$server1> traceroute 192.168.200.1
```



```
(syenv) root@njucs-VirtualBox:~/netlab/lab-05-wayne-ziqi# traceroute 192.168.20
0.1
traceroute to 192.168.200.1 (192.168.200.1), 30 hops max, 60 byte packets
 1  192.168.100.2 (192.168.100.2)  169.053 ms  169.168 ms  169.356 ms
 2  192.168.200.1 (192.168.200.1)  271.153 ms  271.482 ms  271.926 ms
```



```
   30 939.218051505 192.168.100.1      192.168.200.1       UDP    74 47674 → 33434 Len=32
   31 939.218065150 192.168.100.1      192.168.200.1       UDP    74 59953 → 33435 Len=32
   32 939.218072192 192.168.100.1      192.168.200.1       UDP    74 49874 → 33436 Len=32
   33 939.218078572 192.168.100.1      192.168.200.1       UDP    74 59660 → 33437 Len=32
   34 939.218084846 192.168.100.1      192.168.200.1       UDP    74 43032 → 33438 Len=32
   35 939.218091026 192.168.100.1      192.168.200.1       UDP    74 59327 → 33439 Len=32
   36 939.218096842 192.168.100.1      192.168.200.1       UDP    74 47481 → 33440 Len=32
   37 939.218102329 192.168.100.1      192.168.200.1       UDP    74 55347 → 33441 Len=32
   38 939.218107349 192.168.100.1      192.168.200.1       UDP    74 33563 → 33442 Len=32
   39 939.218113489 192.168.100.1      192.168.200.1       UDP    74 46630 → 33443 Len=32
   40 939.218119266 192.168.100.1      192.168.200.1       UDP    74 34732 → 33444 Len=32
   41 939.218125639 192.168.100.1      192.168.200.1       UDP    74 36456 → 33445 Len=32
   42 939.218131643 192.168.100.1      192.168.200.1       UDP    74 39990 → 33446 Len=32
   43 939.218137333 192.168.100.1      192.168.200.1       UDP    74 44865 → 33447 Len=32
   44 939.218142453 192.168.100.1      192.168.200.1       UDP    74 48949 → 33448 Len=32
   45 939.218147751 192.168.100.1      192.168.200.1       UDP    74 55360 → 33449 Len=32
   46 939.279943559 40:00:00:00:00:01  Broadcast           ARP    42 Who has 192.168.100.1? Tell 192.168.100.2
   47 939.279957783 Private_00:00:01   40:00:00:00:00:01   ARP    42 192.168.100.1 is at 10:00:00:00:00:01
   48 939.280514250 40:00:00:00:00:01  Broadcast           ARP    42 Who has 192.168.100.1? Tell 192.168.100.2
   49 939.280520162 Private_00:00:01   40:00:00:00:00:01   ARP    42 192.168.100.1 is at 10:00:00:00:00:01
   50 939.280968246 40:00:00:00:00:01  Broadcast           ARP    42 Who has 192.168.100.1? Tell 192.168.100.2
   51 939.280973205 Private_00:00:01   40:00:00:00:00:01   ARP    42 192.168.100.1 is at 10:00:00:00:00:01
   52 939.387087019 192.168.100.2      192.168.100.1       ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
   53 939.387230123 192.168.100.2      192.168.100.1       ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
   54 939.387424297 192.168.100.2      192.168.100.1       ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
   55 939.387916205 192.168.100.1      192.168.200.1       UDP    74 37764 → 33450 Len=32
   56 939.387926332 192.168.100.1      192.168.200.1       UDP    74 58999 → 33451 Len=32
   57 939.387944691 192.168.100.1      192.168.200.1       UDP    74 33429 → 33452 Len=32
   58 939.489225904 192.168.200.1      192.168.100.1       ICMP  102 Destination unreachable (Port unreachable)
   59 939.489563699 192.168.200.1      192.168.100.1       ICMP  102 Destination unreachable (Port unreachable)
   60 939.490013837 192.168.200.1      192.168.100.1       ICMP  102 Destination unreachable (Port unreachable)
   61 939.490367673 192.168.200.1      192.168.100.1       ICMP  102 Destination unreachable (Port unreachable)
   62 939.490753376 192.168.200.1      192.168.100.1       ICMP  102 Destination unreachable (Port unreachable)
   63 939.491039096 192.168.200.1      192.168.100.1       ICMP  102 Destination unreachable (Port unreachable)
   64 944.247483039 Private_00:00:01   40:00:00:00:00:01   ARP    42 Who has 192.168.100.2? Tell 192.168.100.1
   65 944.270701979 40:00:00:00:00:01  Private_00:00:01    ARP    42 192.168.100.2 is at 40:00:00:00:00:01
```

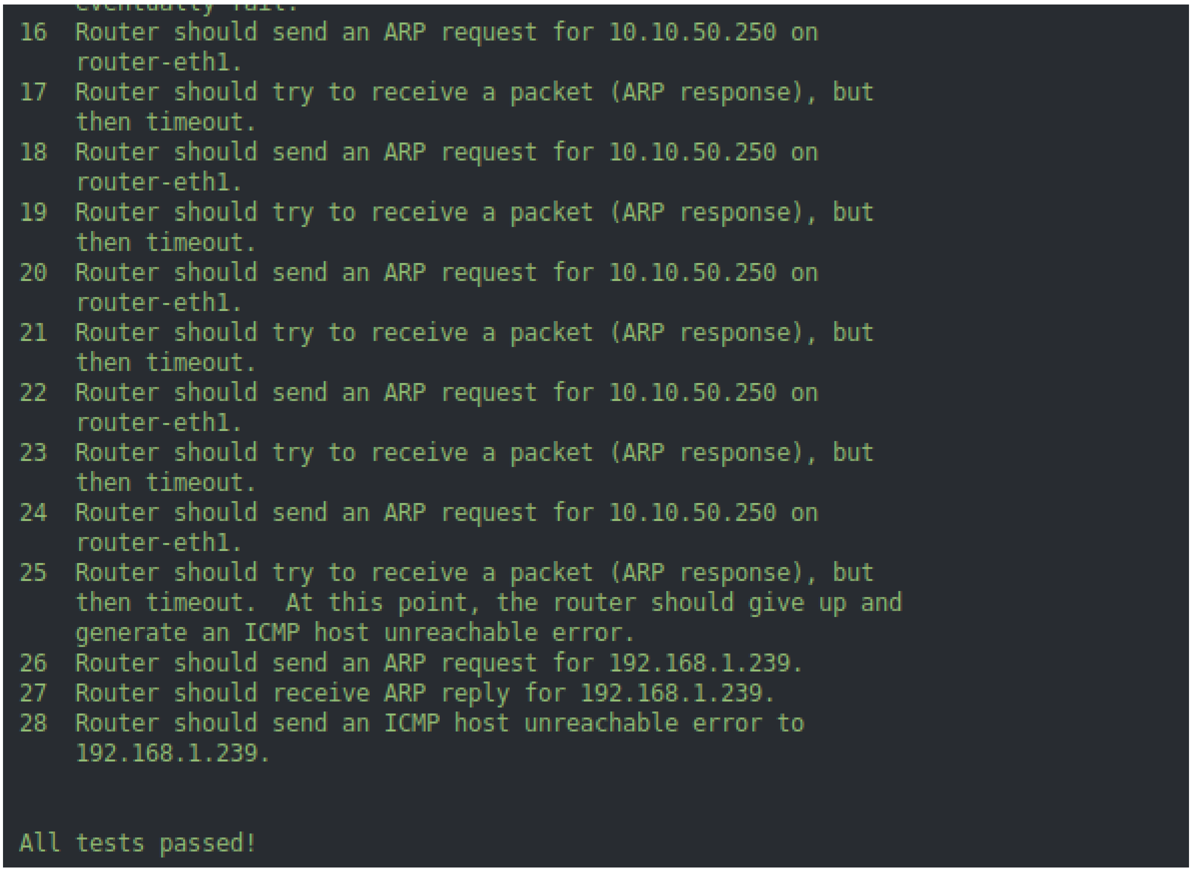
最后，我们用一条 traceroute测试路由器的功能

6. 实验总结

至此，IPv4路由器就算实现完成了，我们回过头来再总结一下具有静态路由功能的路由器需要完成那些工作

- 回复ARP Request：对于发送给路由器端口之一的ARP请求，路由器需要将该端口的目的地址（IP，MAC）告知源端。（lab3）
- 发送ARP Request：对于下一跳IP地址没有缓存到ARP表中的地址，路由器需要向该转发表项指定的端口发送ARP Request，请求这一目的IP的MAC地址，并且会每隔1秒发一次总共发送5次，仍未收到回复会发送ICMP 错误信息。（lab4，lab5）
- 接收ARP Reply：路由器会根据接收到的ARP报文维护ARP表项，再ARP报文的src段就包含了我们需要的答案（lab4）
- 转发路由：查找传发表，确定下一跳地址和发送端口，如果转发表中找不到子网的表项发送ICMP错误信息（lab4，lab5）。
- 回复 `ICMP Request` ：发送给路由器端口之一的 `ICMP Request` 路由器会构造一个 `ICMP Reply` 回复给源端，证明连接正确；发给端口之一的其他类型的报文由于不属于路由器的处理范畴因此会发送ICMP错误信息（lab5）

附：测试正确截图



```
    eventually fail.
16  Router should send an ARP request for 10.10.50.250 on
    router-eth1.
17  Router should try to receive a packet (ARP response), but
    then timeout.
18  Router should send an ARP request for 10.10.50.250 on
    router-eth1.
19  Router should try to receive a packet (ARP response), but
    then timeout.
20  Router should send an ARP request for 10.10.50.250 on
    router-eth1.
21  Router should try to receive a packet (ARP response), but
    then timeout.
22  Router should send an ARP request for 10.10.50.250 on
    router-eth1.
23  Router should try to receive a packet (ARP response), but
    then timeout.
24  Router should send an ARP request for 10.10.50.250 on
    router-eth1.
25  Router should try to receive a packet (ARP response), but
    then timeout.  At this point, the router should give up and
    generate an ICMP host unreachable error.
26  Router should send an ARP request for 192.168.1.239.
27  Router should receive ARP reply for 192.168.1.239.
28  Router should send an ICMP host unreachable error to
    192.168.1.239.


All tests passed!
```