

# 南京大学本科生实验报告

课程名称： 计算机网络

任课教师：李文中

助教：

学院	计算机科学与技术	专业（方向）	计算机科学与技术
学号	191220029	姓名	傅小龙
Email	<a href="mailto:1830970417@qq.com">1830970417@qq.com</a>	开始/完成日期	2021/5/14 - 2021/5/16

## 1.实验名称

Lab5: Respond to ICMP

## 2.实验目的

设计并实现路由器对ICMP包的回复，并在必要的时候生成ICMP错误信息的相关逻辑。

## 3. 实验内容

### 3.1 Responding to ICMP echo requests

#### 3.1.1 Coding

这里要求实现路由器对收到的 `ICMP echo request` 作出回复。当然，对于不是发给自己的请求则转发之，对于收到的非ICMP请求则发出 `ICMP error` 信息。

执行逻辑如下：

```
Is this packet for me? -----+
|yes                        no      |
|↓                            |
Does this packet have ICMP header? -----+
|yes                        no      |
|↓                            |
Is this packet ICMP echo request? |
|yes                        ↓no     |
|↓          create a ICMP error message-----+
create a ICMP reply packet         |
|                                  |
+-----+
|↓
.....
```

回复包的ICMP包头需要重新生成：序列号和请求包的相同，标识符和请求包的相同，数据和请求包的相同。回复包的IP包头中的目的ip为原来请求包IP包头的源ip,源ip为路由接口的ip。

代码实现如下：

```
for interface in interfaces:
    if ipv4.dst == interface.ipaddr :
        if icmp and icmp.icmptype == ICMPType.EchoRequest: # echo request
```

```

        print("This pkt makes an echoRequest for me.")
        echo_reply = ICMP()
        echo_reply.icmptype = ICMPType.EchoReply
        echo_reply.icmpdata.sequence = icmp.icmpdata.sequence
        echo_reply.icmpdata.identifier = icmp.icmpdata.identifier
        echo_reply.icmpdata.data = icmp.icmpdata.data
        packet[icmp_index] = echo_reply
        packet[ipv4_index].dst = packet[ipv4_index].src
        packet[ipv4_index].src = interface.ipaddr
        ether, ipv4, icmp = self.headers(packet)
        break
    else: # not echo request, send ICMP destination port unreachable error
pkt
        print("oh! this pkt is for me but not ICMP")
        '''Implemented in 3.2.1 Coding part'''
.....

```

## 3.2 Generating ICMP error messages

### 3.2.1 Coding

ICMP错误信息包括原数据包ICMP包头的28字节，错误类型和错误码。ICMP错误信息的生成封装于 `ctreate_icmp_err(self, packet, interface, err_type, err_code)` 函数中，参考实验手册 Task3 Coding环节，其实现如下：

```

def ctreate_icmp_err(self, packet, interface, err_type, err_code):
    i = packet.get_header_index(Ethernet)
    ether = packet[i]
    # remove Ethernet header --- the errored packet contents sent with the
    ICMP error message should not have an Ethernet header
    del packet[i]
    ipv4_index = packet.get_header_index(IPv4)
    #icmp_index = packet.get_header_index(ICMP)

    icmp = ICMP()
    ip = IPv4()
    # protocol defaults to ICMP
    ip.protocol = IPProtocol.ICMP
    ip.ttl = default_ttl
    ip.dst = packet[ipv4_index].src
    ip.src = interface.ipaddr
    icmp.icmptype = err_type
    icmp.icmpcode = err_code
    icmp.icmpdata.data = packet.to_bytes()[28:]
    print(f"icmp_error pkt created:{ether + ip + icmp}")
    return ether + ip + icmp

```

本次实验需要路由器能够生成的ICMP错误信息的情况有以下四种：

#### ① ICMP destination network unreachable

在转发包时，若最长匹配失败，那么需要向发送方发送这一错误信息。相关逻辑如下：

```
if longest prefix match failed
|___then create ICMP error message packet accordingly
|___make a longest prefix match for error message packet
add packet to waiting queue
```

代码实现如下:

```
...(longest prefix match)...
if cur_prefixlen == 0:
    print("oh! prefix macth failed!!")
    for interface in interfaces:
        if interface.name == ifaceName:
            packet = self.ctreate_icmp_err(packet, interface,
            ICMPType.DestinationUnreachable, 0)
            ether, ipv4, icmp = self.headers(packet)
            break
            cur_prefixlen, packet, nhopip, interface = self.longest_prefix_match(packet)
            print(f"packet from {interface.name} with nexthopip {nhopip} waiting to
            forward")
            self.queue.append(WQentry(packet, interface, nhopip))
...(ttl check)...
```

### ② ICMP time exceeded

在IPv4包头的TTL项减为0时, 路由器将抛弃该包, 并向发送方发送这一错误信息. 对TTL项的检查应在最长匹配之前.

代码实现如下:

```
if packet[ipv4_index].ttl <= 0:
    print("oh! this packet has expired!(ttl <= 0)")
    for interface in interfaces:
        if interface.name == ifaceName:
            packet = self.ctreate_icmp_err(packet, interface,
            ICMPType.TimeExceeded, ICMPCodeTimeExceeded.TTLExpired)
            ether, ipv4, icmp = self.headers(packet)
            break
            cur_prefixlen, packet, nhopip, interface = self.longest_prefix_match(packet)
...(add packet to waiting queue)...
```

### ③ ICMP destination host unreachable

在路由器转发包时, 若无法获取下一跳的mac信息, 那么取消所有相同下一跳的包的转发操作, 并向这些包的不同的发送方发送这一错误信息。

在具体实现中, 在发现有包转发失败后, 遍历队列中所有等待转发的包, 若和转发失败的包由相同的下一跳ip, 那么将该项添加到完成队列 (List) finish 等待删除, 并采用 (List) source 来记录该包的发送方ip, 若该ip是新加入的一项, 那么向该ip发送对应的ICMP错误信息。

代码实现如下:

```

'''wqEntry failed to forward:'''
source = [] # senders who send pkts to same next hop
for tmp_entry in self.queue:
    if tmp_entry.nhopip == wqEntry.nhopip:
        finish.append(tmp_entry)
        ipv4_index = tmp_entry.packet.get_header_index(IPv4)
        if tmp_entry.packet[ipv4_index].src not in source:
            source.append(tmp_entry.packet[ipv4_index].src)
            packet = self.ctreate_icmp_err(wqEntry.packet, wqEntry.interface,
            ICMPType.DestinationUnreachable, 1)
            interfaces = self.net.interfaces
            prefixlen, packet, nhopip, interface =
self.longest_prefix_match(packet)
            packet[packet.get_header_index(IPv4)].src = interface.ipaddr
            self.queue.append(wqEntry(packet, interface, nhopip))

```

#### ④ ICMP destination port unreachable

当路由器收到发送给自己的包后，若不是 ICMP echo request，那么需要向发送方发送这一错误信息。这一部分的具体位置在 3.1.1 Coding 环节的代码空缺部分。

代码实现如下：

```

for interface in interfaces:
    if interface.name == ifaceName:
        packet = self.ctreate_icmp_err(packet, interface,
        ICMPType.DestinationUnreachable, 3)
        ether, ipv4, icmp = self.headers(packet)
        break

```

注：每个类型错误信息的错误码在API Reference上可查阅到：

```

>>> list(ICMPTypeCodeMap[ICMPType.DestinationUnreachable])
[ <DestinationUnreachable.HostUnreachable: 1>,
  <DestinationUnreachable.NetworkUnreachable: 0>,
  <DestinationUnreachable.PortUnreachable: 3>
  ...
]
>>> i = ICMP()
>>> i.icmpcode
<ICMPCodeTimeExceeded.TTLExpired: 0>

```

## 3.2.2 Testing

测试结果如下：

```
eth1. The destination address 1.2.3.4 should not match any
entry in the forwarding table.
12 Router should send an ICMP destination network unreachable
error back to 10.10.123.123 out router-eth1.
13 A UDP packet addressed to the router's IP address
192.168.1.1 should arrive on router-eth1. The router cannot
handle this type of packet and should generate an ICMP
destination port unreachable error.
14 The router should send an ICMP destination port unreachable
error back to 172.16.111.222 out router-eth1.
15 An IP packet from 192.168.1.239 for 10.10.50.250 should
arrive on router-eth0. The host 10.10.50.250 is presumed
not to exist, so any attempts to send ARP requests will
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!

```
(syenv) njucs@njucs-VirtualBox:~/cnLab/cnLab05/lab-5-191220029$
```

### 3.2.3 Deploying

#### A. ICMP Echo Request

在 `server1` 端向 `client` 发送一个 ICMP Echo Request:

```
(server1)# ping -c 1 10.1.1.1
```

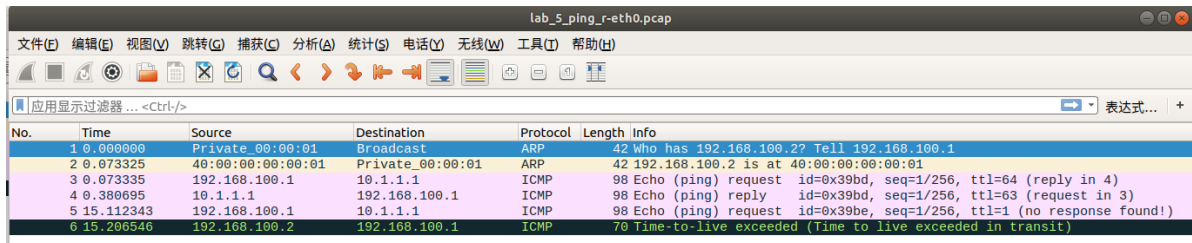
预期能够正确收到 `client` 发来的 ICMP Echo Reply

然后再发送一个ttl为1的ICMP Echo Request:

```
(server1)# ping -c 1 -t 1 10.1.1.1
```

预期将收到 Router 发来的ICMP错误信息:TTL = 0

监听 router-eth0 接口，得到以下结果：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.073325	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	0.073335	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x39bd, seq=1/256, ttl=64 (reply in 4)
4	0.380695	10.1.1.1	192.168.100.1	ICMP	98	Echo (ping) reply id=0x39bd, seq=1/256, ttl=63 (request in 3)
5	15.112343	192.168.100.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x39be, seq=1/256, ttl=1 (no response found!)
6	15.206546	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

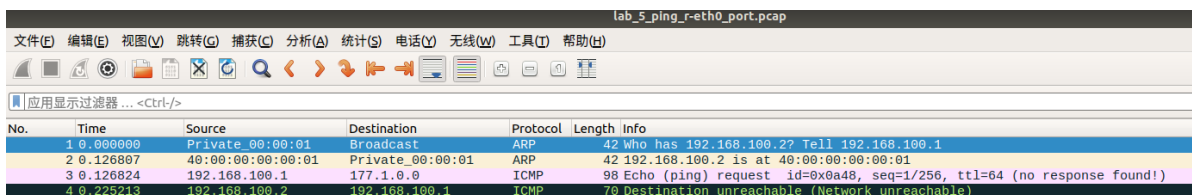
可以发现，第一次的ping操作收到了回复（对应第4个包），而第二次ping操作由于包的ttl减为0，router产生了 ttl exceeded 的ICMP错误信息。

修改 start\_mininet.py，添加一条路由 set\_route(net, 'server1', '177.1.0.0/16', '192.168.100.2')，然后启动mininet，在在 server1 端向 177.1.0.0 发送一个ICMP Echo Request:

```
(server1)# ping -c 1 -t 1 177.1.0.0
```

预期将受到 Router 发送的ICMP错误信息：Destination Port Unreachable.

在 router-eth0 抓包结果如下：



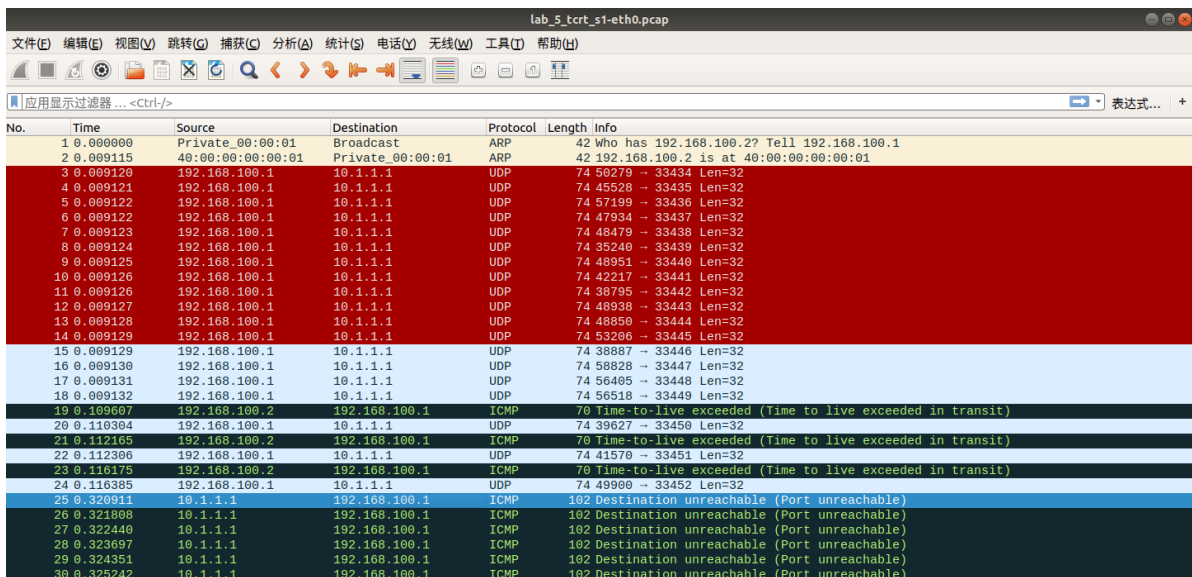
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.126897	40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
3	0.126824	192.168.100.1	177.1.0.0	ICMP	98	Echo (ping) request id=0x0a48, seq=1/256, ttl=64 (no response found!)
4	0.225213	192.168.100.2	192.168.100.1	ICMP	70	Destination unreachable (Network unreachable)

## B. Traceroute

在 server1 端以 client 为目标执行traceroute指令：

```
(server1)# traceroute 10.1.1.1
```

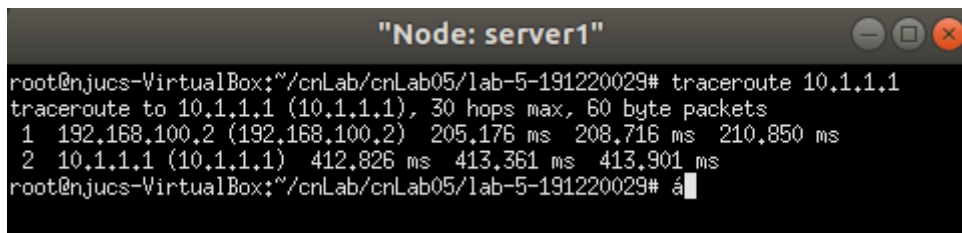
监听 server1-eth0 端口，得到以下结果：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
2	0.009115	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	0.009120	192.168.100.1	10.1.1.1	UDP	74	50279 -> 33434 Len=32
4	0.009121	192.168.100.1	10.1.1.1	UDP	74	45528 -> 33435 Len=32
5	0.009122	192.168.100.1	10.1.1.1	UDP	74	57199 -> 33436 Len=32
6	0.009122	192.168.100.1	10.1.1.1	UDP	74	47934 -> 33437 Len=32
7	0.009123	192.168.100.1	10.1.1.1	UDP	74	48479 -> 33438 Len=32
8	0.009124	192.168.100.1	10.1.1.1	UDP	74	35248 -> 33439 Len=32
9	0.009125	192.168.100.1	10.1.1.1	UDP	74	48951 -> 33440 Len=32
10	0.009126	192.168.100.1	10.1.1.1	UDP	74	42217 -> 33441 Len=32
11	0.009126	192.168.100.1	10.1.1.1	UDP	74	38795 -> 33442 Len=32
12	0.009127	192.168.100.1	10.1.1.1	UDP	74	48938 -> 33443 Len=32
13	0.009128	192.168.100.1	10.1.1.1	UDP	74	48850 -> 33444 Len=32
14	0.009129	192.168.100.1	10.1.1.1	UDP	74	53206 -> 33445 Len=32
15	0.009129	192.168.100.1	10.1.1.1	UDP	74	38887 -> 33446 Len=32
16	0.009130	192.168.100.1	10.1.1.1	UDP	74	58828 -> 33447 Len=32
17	0.009131	192.168.100.1	10.1.1.1	UDP	74	56405 -> 33448 Len=32
18	0.009132	192.168.100.1	10.1.1.1	UDP	74	56518 -> 33449 Len=32
19	0.100007	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
20	0.110304	192.168.100.1	10.1.1.1	UDP	74	39627 -> 33450 Len=32
21	0.112165	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
22	0.112306	192.168.100.1	10.1.1.1	UDP	74	41570 -> 33451 Len=32
23	0.116175	192.168.100.2	192.168.100.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
24	0.116385	192.168.100.1	10.1.1.1	UDP	74	49900 -> 33452 Len=32
25	0.320911	10.1.1.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)
26	0.321808	10.1.1.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)
27	0.322440	10.1.1.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)
28	0.323697	10.1.1.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)
29	0.324351	10.1.1.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)
30	0.325242	10.1.1.1	192.168.100.1	ICMP	102	Destination unreachable (Port unreachable)

server1 在通过ARP知道下一跳(router-eth0)的mac后, 连续发送了19UDP包, 这些UDP包的TTL从1开始, 每3个递增1直到7, 直到收到 client 发来的包才停止. client 最后发送了6个ICMP error(Port unreachable), 中间的3个(No.19, 21, 23)ICMP error包为router发送的, 错误码为ttl exceeded.

traceroute结果如下:



```
root@njucs-VirtualBox:~/cnLab/cnLab05/lab-5-191220029# traceroute 10.1.1.1
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)  205.176 ms  208.716 ms  210.850 ms
 2  10.1.1.1 (10.1.1.1)  412.826 ms  413.361 ms  413.901 ms
root@njucs-VirtualBox:~/cnLab/cnLab05/lab-5-191220029#
```

## 4.实验总结与感想

本次实验完成了IPv4路由的最后一部分: 响应ICMP。通过本次实验进一步认识到了网络层路由的逻辑以及ICMP协议相关的数据结构及原理。

## 5.遇到的问题和思考

在对发送给路由器自己的非ICMP包作出ICMP error message响应时(对应3.2.1 ④ICMP destination port unreachable), 不能直接将该包的目的ip作为ICMP错误信息的源ip, 否则将导致错误。

这是由于在判断包是否是发送给路由器的时候, 条件是目的ip是否是路由器所有接口中的一个, 这个接口可能并不是路由器收到包的那一个接口。ICMP error message包的转发接口应当是路由器收到包的接口才对。