

# 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：洋溢	
学院	计算机	专业（方向）	计算机科学与技术		
学号	221220142	姓名	欧阳瑞泽		
Email	221220142@smail.nju.edu.cn	开始/完成日期	2024.5.11		

## Lab 5: Respond to ICMP

### 1. 实验目的

在 Lab4 的基础上，实现 router 的 ICMP 功能，包括对 PING 请求的回复，生成 ICMP 错误信息。

### 2. 实验内容

#### Step 1: Responding to ICMP echo requests

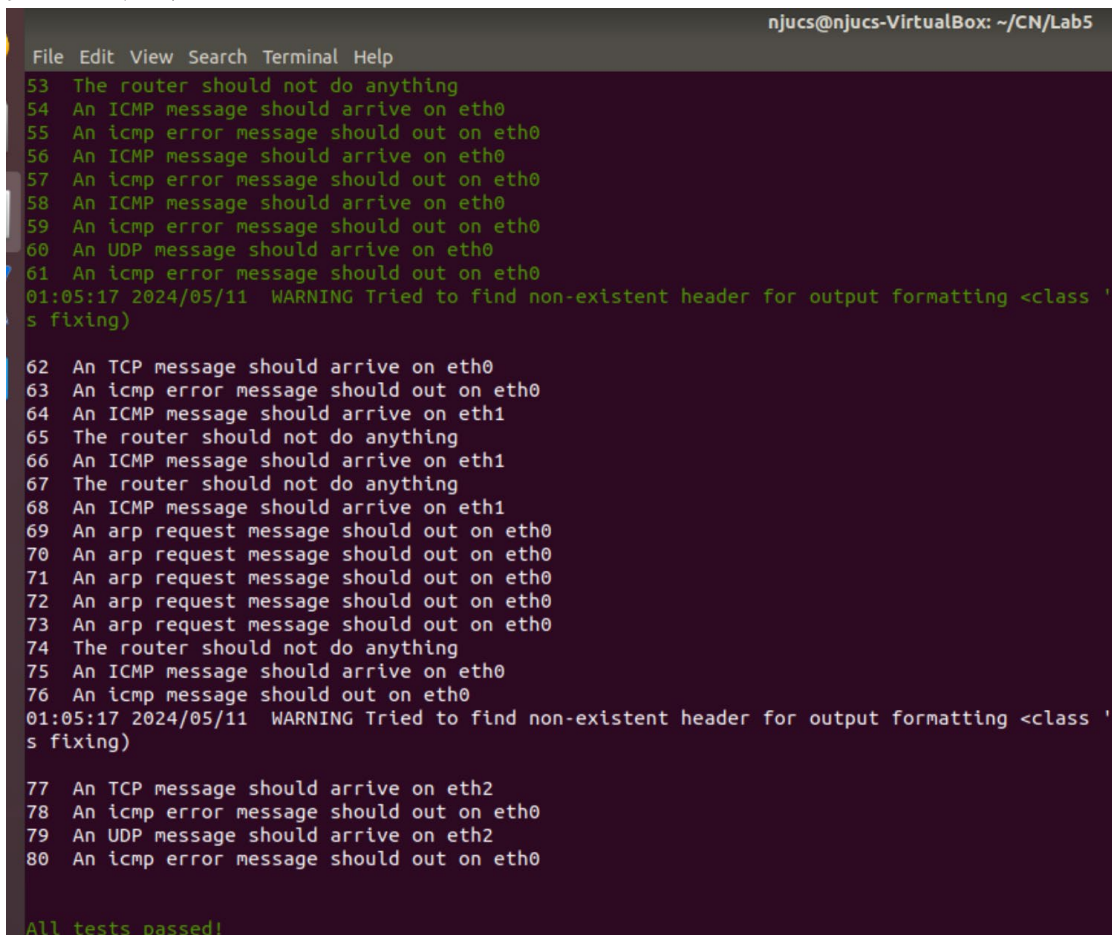
要回应 ICMP request，首先要判断收到的 packet 的目标 ip 是路由器的 port 之一重新制造数据包，修改 ttl，将数据包的 IPv4 头的源地址和目的地址互换以实现回应。发送时仍然需要匹配最长前缀来找到发送 port，而不是直接从接收端口发送，这一部分用递归实现，后面生成发送 ICMP 同理。

#### Step 2: Generating ICMP error messages

对收到的 packet 的 IP 头进行判断，若目的地址是路由器 IP，进行以下判断：如果包是 ICMP request，执行 Step1，其次如果是 ICMP 错误信息，则不进行响应：因为如果一个路由器在收到 ICMP 错误消息时生成另一个 ICMP 错误消息作为响应，可能会导致 ICMP 错误消息的循环，进而造成网络负载增加和不必要的网络流量。然后如果是其他未定义行为，生成

DestinationUnreachable 类型错误 ICMP 数据包回应。若目的地址不是路由器 IP，则路由器需要转发这个包，先匹配最长前缀来寻找发送 port，若未找到则生成 DestinationUnreachable 类型错误 ICMP 数据包回应。然后将 ttl 减一，若 ttl 等于 0，则生成 TimeExceeded 类型错误 ICMP 数据包回应。注意到回应时仍需匹配最长前缀来找到发送 port，所以这一部分用递归实现。

### 3. 实验结果



```
njucs@njucs-VirtualBox: ~/CN/Lab5
File Edit View Search Terminal Help
53 The router should not do anything
54 An ICMP message should arrive on eth0
55 An icmp error message should out on eth0
56 An ICMP message should arrive on eth0
57 An icmp error message should out on eth0
58 An ICMP message should arrive on eth0
59 An icmp error message should out on eth0
60 An UDP message should arrive on eth0
61 An icmp error message should out on eth0
01:05:17 2024/05/11 WARNING Tried to find non-existent header for output formatting <class '
s fixing)
62 An TCP message should arrive on eth0
63 An icmp error message should out on eth0
64 An ICMP message should arrive on eth1
65 The router should not do anything
66 An ICMP message should arrive on eth1
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
01:05:17 2024/05/11 WARNING Tried to find non-existent header for output formatting <class '
s fixing)
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!
```

在 mininet 中部署：

用 wireshark 监听 router 与 client 相连的端口。

用 client ping 10.1.1.2

```
*** client : ('route add -net 192.168.200.0/24 gw 10.1.1.2',)
*** client : ('route add -net 172.16.0.0/16 gw 10.1.1.2',)
*** client : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** client : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** router : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** router : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** server2 : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** server2 : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** Starting controller
*** Starting 0 switches

*** Starting CLI:
mininet> xterm router
mininet> router wireshark &
mininet> client ping -c 1 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data:
64 bytes from 10.1.1.2: icmp_seq=1 ttl=63 time=186 ms

--- 10.1.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 186.533/186.533/186.533/0.000 ms
```

可见 router 收到 Echo Request 后进行回应，发出了 Echo Reply。注意这里要设置 Reply 包 ICMP 头的 .icmpdata.identifier 要和 Request 包的一样否则 ping 不通（client 收到了 Reply 包但无法判别）

用 client ping -c 1 -t 1 192.168.200.1

```
net.ipv6.conf.all.disable_ipv6 = 1
*** server2 : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** Starting controller
*** Starting 0 switches

*** Starting CLI:
mininet> xterm router
mininet> router wireshark &
mininet> client ping -c 1 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data:
64 bytes from 10.1.1.2: icmp_seq=1 ttl=63 time=186 ms

--- 10.1.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 186.533/186.533/186.533/0.000 ms
mininet> client ping -c 1 -t 1 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
From 10.1.1.2: icmp_seq=1 Time to live exceeded

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss
```

可见收到 ping 192.168.200.1 的包后将 TTL 减一。多了一个 ICMP error 包回应 TTL exceeded。

用 client ping -c 1 172.16.1.1

```
mininet> router wireshark &
mininet> client ping -c 1 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data:
64 bytes from 10.1.1.2: icmp_seq=1 ttl=63 time=186 ms

--- 10.1.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 186.533/186.533/186.533/0.000 ms
mininet> client ping -c 1 -t 1 192.168.200.1
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data:
From 10.1.1.2: icmp_seq=1 Time to live exceeded

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss
mininet> client ping -c 1 172.16.1.1
PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data:
From 10.1.1.2: icmp_seq=1 Destination Net Unreachable

--- 172.16.1.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss
```

可见 router 收到 Request 后没有查询到发送端口，生成 DestinationUnreachable 类型错误 ICMP 数据

包回应。

安装 traceroute 后测试：

```
mininet> client traceroute 192.168.100.1
traceroute to 192.168.100.1 (192.168.100.1), 30 hops max, 60 byte packets
 1  10.1.1.2 (10.1.1.2)  68.646 ms  69.174 ms  69.692 ms
 2  192.168.100.1 (192.168.100.1)  278.258 ms  278.638 ms  278.849 ms
mininet> █
```

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	0.025228661	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	0.025254826	10.1.1.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x4e07, seq=1/256, ttl=64 (reply in 4)
4	0.120612927	10.1.1.2	10.1.1.1	ICMP	98	Echo (ping) reply id=0x4e07, seq=1/256, ttl=63 (request in 3)
5	4.231717884	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x4e09, seq=1/256, ttl=1 (no response found!)
6	4.262082791	10.1.1.2	10.1.1.1	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
7	18.577216188	10.1.1.1	172.16.1.1	ICMP	98	Echo (ping) request id=0x4e0c, seq=1/256, ttl=64 (no response found!)
8	18.588849574	10.1.1.2	10.1.1.1	ICMP	70	Destination unreachable (Network unreachable)
9	27.264705187	10.1.1.1	192.168.100.1	UDP	74	42909 → 33434 Len=32
10	27.264807177	10.1.1.1	192.168.100.1	UDP	74	56066 → 33435 Len=32
11	27.264813609	10.1.1.1	192.168.100.1	UDP	74	36086 → 33436 Len=32
12	27.264818424	10.1.1.1	192.168.100.1	UDP	74	46568 → 33437 Len=32
13	27.264823313	10.1.1.1	192.168.100.1	UDP	74	41673 → 33438 Len=32
14	27.264828215	10.1.1.1	192.168.100.1	UDP	74	41821 → 33439 Len=32
15	27.264832562	10.1.1.1	192.168.100.1	UDP	74	32872 → 33440 Len=32
16	27.264836833	10.1.1.1	192.168.100.1	UDP	74	39187 → 33441 Len=32
17	27.264842562	10.1.1.1	192.168.100.1	UDP	74	36038 → 33442 Len=32
18	27.264847202	10.1.1.1	192.168.100.1	UDP	74	43986 → 33443 Len=32
19	27.264852008	10.1.1.1	192.168.100.1	UDP	74	52335 → 33444 Len=32
20	27.264856492	10.1.1.1	192.168.100.1	UDP	74	48411 → 33445 Len=32
21	27.264861306	10.1.1.1	192.168.100.1	UDP	74	51158 → 33446 Len=32
22	27.264866307	10.1.1.1	192.168.100.1	UDP	74	55054 → 33447 Len=32
23	27.264870860	10.1.1.1	192.168.100.1	UDP	74	37465 → 33448 Len=32
24	27.264876902	10.1.1.1	192.168.100.1	UDP	74	50373 → 33449 Len=32
25	27.333423262	10.1.1.2	10.1.1.1	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
26	27.333978585	10.1.1.2	10.1.1.1	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
27	27.334503608	10.1.1.2	10.1.1.1	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
28	27.335031265	10.1.1.1	192.168.100.1	UDP	74	43932 → 33450 Len=32
29	27.335040237	10.1.1.1	192.168.100.1	UDP	74	50369 → 33451 Len=32
30	27.335044937	10.1.1.1	192.168.100.1	UDP	74	51790 → 33452 Len=32
31	27.543071841	192.168.100.1	10.1.1.1	ICMP	102	Destination unreachable (Port unreachable)
32	27.543459283	192.168.100.1	10.1.1.1	ICMP	102	Destination unreachable (Port unreachable)
33	27.543674493	192.168.100.1	10.1.1.1	ICMP	102	Destination unreachable (Port unreachable)
34	27.543855955	192.168.100.1	10.1.1.1	ICMP	102	Destination unreachable (Port unreachable)
35	27.544040257	192.168.100.1	10.1.1.1	ICMP	102	Destination unreachable (Port unreachable)
36	27.544217415	192.168.100.1	10.1.1.1	ICMP	102	Destination unreachable (Port unreachable)

可见traceroute原理是发送TTL快过期的UDP包来探寻整个网络。路由器做出了正确回应。

## 4. 核心代码



```

def handle_IP_packet(self, packet):
    IPv4_header = packet.get_header(IPv4)
    ICMP_header = packet.get_header(ICMP)
    #find minmax prefix

    if IPv4_header.dst in self.myip:
        if packet.get_header(UDP) is not None:
            error_packet = self.create_error_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src, ICMPType.DestinationUnreachable)
            self.handle_IP_packet(error_packet)

        if ICMP_header is None or ICMP_header.icmptype == ICMPType.DestinationUnreachable or ICMP_header.icmptype == ICMPType.TimeExceeded:
            return

        if ICMP_header.icmptype == ICMPType.EchoRequest:
            #for router, teckle in Lab5
            ping_packet = self.create_ping_packet(packet, IPv4_header.dst, IPv4_header.src)
            #ping_packet = self.create_ping_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src)
            self.handle_IP_packet(ping_packet)
        else:
            error_packet = self.create_error_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src, ICMPType.DestinationUnreachable)
            self.handle_IP_packet(error_packet)
        return

    entry = self.prefix_match(IPv4_header.dst)
    if entry is None:
        if packet.get_header(IPv4).src == IPv4Address('0.0.0.0'):
            return

        if ICMP_header and (ICMP_header.icmptype == ICMPType.DestinationUnreachable or ICMP_header.icmptype == ICMPType.TimeExceeded):
            return

        error_packet = self.create_error_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src, ICMPType.DestinationUnreachable, 0)
        self.handle_IP_packet(error_packet)
        self.handle_IP_packet(ping_packet)
    else:
        error_packet = self.create_error_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src, ICMPType.DestinationUnreachable)
        self.handle_IP_packet(error_packet)
        return

    entry = self.prefix_match(IPv4_header.dst)
    if entry is None:
        if packet.get_header(IPv4).src == IPv4Address('0.0.0.0'):
            return

        if ICMP_header and (ICMP_header.icmptype == ICMPType.DestinationUnreachable or ICMP_header.icmptype == ICMPType.TimeExceeded):
            return

        error_packet = self.create_error_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src, ICMPType.DestinationUnreachable, 0)
        self.handle_IP_packet(error_packet)
        return

    IPv4_header.ttl = max(0, IPv4_header.ttl - 1)
    if IPv4_header.ttl == 0:
        #teckle in Lab5
        if ICMP_header and (ICMP_header.icmptype == ICMPType.DestinationUnreachable or ICMP_header.icmptype == ICMPType.TimeExceeded):
            return

        error_packet = self.create_error_packet(packet, IPv4Address('0.0.0.0'), IPv4_header.src, ICMPType.TimeExceeded, 0)
        self.handle_IP_packet(error_packet)
        return

    network_address, subnet_mask, next_hop_address, interface = entry
    Oface = self.net.interface_by_name(interface)
    if IPv4_header.src == IPv4Address('0.0.0.0'):
        IPv4_header.src = Oface.ipaddr
    self.forwarding(packet)
    return

```

## 5. 总结与感想

规则较多比较麻烦，把 Lab4 的代码封装成函数比较方便发送 packet。开始没有意识到 ICMP 错误信息包也需要前缀匹配，发现了过后修改代码，最后用递归实现。