

# p0Computer Network lab5实验报告

姓名	学号
张洋彬	191220169
邮箱	完成日期
<a href="mailto:1016466918@qq.com">1016466918@qq.com</a>	2021.5.14

## p0Computer Network lab5实验报告

- 1、实验名称
  - Lab 5 : Respond to ICMP
- 2、实验目的
- 3、实验进行
  - 3.1 Preparation
  - 3.2 Responding to ICMP echo requests
  - 3.3 Generating ICMP error messages
  - mininet deployment
- 4、实验感想

## 1、实验名称

### Lab 5 : Respond to ICMP

## 2、实验目的

- 1、响应ICMP回显请求
- 2、根据四种不同的情况，生成不同的icmp错误信息

## 3、实验进行

### 3.1 Preparation

将lab4写的文件移植到lab5中，文件结构如下：（无报告的时候）

bbzunyi 1		c91d0c1 31 seconds ago	🕒 4 commits
📁 .github	GitHub Classroom Feedback		7 hours ago
📁 testcases	Initial commit		7 hours ago
📄 .gitignore	Initial commit		7 hours ago
📄 README.md	Initial commit		7 hours ago
📄 forwarding_table.txt	1		31 seconds ago
📄 myrouter.py	1		31 seconds ago
📄 start_mininet.py	1		31 seconds ago

## 3.2 Responding to ICMP echo requests

按照实验介绍的逻辑，大致如下：

1、在对IP数据包做转发决定之前，应该首先检查ip目的地址是否与分配给路由器的接口之一的ip地址一致，然后检查这个packet的类型，如果是ICMP echo request，应该创建一个ICMP echo reply包发送给发送方

```
1  for intf in interfaces:
2      if ipv4.dst == intf.ipaddr:
3          icmp=packet.get_header(ICMP)
4          if icmp is not None:
5              if icmp.icmpdtype == ICMPType.EchoRequest: #下面的代码接这一部分
6                  echo_reply=ICMP()
```

2、ICMP echo reply的创建过程如下：

- 将echo request的sequence复制到echo reply中
- 将echo request的identifier复制到echo reply中
- 将echo request的数据字段设置到echo reply中

```
1  echo_reply.icmpdtype=ICMPType.EchoReply
2  echo_reply.icmpdata.sequence=icmp.icmpdata.sequence
3  echo_reply.icmpdata.identifier=icmp.icmpdata.identifier
```

- 将ip头的将目标地址和原地址互换，数据包의ICMP头设置为新的ICMP头

```
1  ipv4.dst=ipv4.src
2  ipv4.src=intf.ipaddr
3  packet[packet.get_header_index(ICMP)]=echo_reply
4  break
```

- 按照之前写的发送（转发）构造的数据包

## 3.3 Generating ICMP error messages

ICMP错误主要有四种情况：

1、与转发表中的内容不匹配，解决方案：

发送ICMP destination network unreachable error给发送方

Note: the ICMP type should be destination unreachable, and the ICMP code should be network unreachable.

四种情况的代码类似，只是在修改icmpdata和icmptype的时候存在差异，对这种情况进行特别说明

```
1 i = packet.get_header_index(Ethernet)
2 # remove Ethernet header --- the errored packet contents sent with
3 # the ICMP error message should not have an Ethernet header
4 del packet[i]# 去除包的ethernet头
5 icmp = ICMP()#创建新的icmp头
6 icmp.icmptype = ICMPType.DestinationUnreachable
7 icmp.icmpcode = ICMPTypeCodeMap[icmp.icmptype].NetworkUnreachable
8 icmp.icmpdata.data = packet.to_bytes()[28]
9 print(icmp)
10 # ICMP TimeExceeded:TTLExpired 28 bytes of raw payload
11 # (b'E\x00\x00\x1c\x00\x00\x00\x00\x01') OrigDgramLen: 0
12 ip = IPv4()#创建新的ipv4包头
13 ip.protocol = IPProtocol.ICMP
14 longest_prefixlen = 0
15 match = None#将包的源地址作为目的地址发包
16 for entry in self.forwarding_table:
17     entry_netaddr= IPv4Network(entry[0]+'/'+entry[1],strict = False)
18     if packet[IPv4].src in entry_netaddr:
19         if entry_netaddr.prefixlen > longest_prefixlen:
20             longest_prefixlen = entry_netaddr.prefixlen
21             match = entry
22 ip.src = self.net.interface_by_name(match[3]).ipaddr
23 ip.dst = packet[IPv4].src
24 ip.ttl = 64
25 ether = Ethernet()
26 ether.ethertype = EtherType.IPv4
27 new_packet = ether + ip + icmp
28 self.forward_packet(match,new_packet,packet[IPv4].src)#使用转发函数将新产生的包转发
```

在获取packet的ICMP头的部分信息并修改信息后，修改ip头的信息后，形成一个新的包通过转发机制给转发出去。（不对源地址不在forwarding table中进行特殊处理，因为如果不在的话就匹配不到，后续操作也不会有意义）

特别说明：

`forward_packet` 函数是重新封装过的函数，和lab4写的内容一样，只是因为四种情况都需要调用这个函数，所以将其封装起来，以便调用

2、TTL递减为0，超时，解决方案：

```
1 icmp.icmptype = ICMPType.TimeExceeded
2 icmp.icmpcode = ICMPTypeCodeMap[icmp.icmptype].TTLExpired
```

3、对dstip arp五次了之后，依旧不知道目的ip地址的mac地址，将这些包丢弃，解决方案：

```
>>> list(ICMPTypeCodeMap[ICMPType.DestinationUnreachable])
[ <DestinationUnreachable.ProtocolUnreachable: 2>,
  <DestinationUnreachable.SourceHostIsolated: 8>,
  <DestinationUnreachable.FragmentationRequiredDFSet: 4>,
  <DestinationUnreachable.HostUnreachable: 1>,
  <DestinationUnreachable.DestinationNetworkUnknown: 6>,
  <DestinationUnreachable.NetworkUnreachableForTOS: 11>,
  <DestinationUnreachable.HostAdministrativelyProhibited: 10>,
  <DestinationUnreachable.DestinationHostUnknown: 7>,
  <DestinationUnreachable.HostPrecedenceViolation: 14>,
  <DestinationUnreachable.PrecedenceCutoffInEffect: 15>,
  <DestinationUnreachable.NetworkAdministrativelyProhibited: 9>,
  <DestinationUnreachable.NetworkUnreachable: 0>,
  <DestinationUnreachable.SourceRouteFailed: 5>,
  <DestinationUnreachable.PortUnreachable: 3>,
  <DestinationUnreachable.CommunicationAdministrativelyProhibited: 13>,
  <DestinationUnreachable.HostUnreachableForTOS: 12> ]
```

```
1 icmp.icmptype = ICMPType.DestinationUnreachable
2 icmp.icmpcode = ICMPTypeCodeMap[icmp.icmptype].HostUnreachable
```

4、传入的数据包发往分配给路由器接口之一的IP地址，但该数据包不是ICMP echo request

```
1 icmp.icmptype = ICMPType.DestinationUnreachable
2 icmp.icmpcode = ICMPTypeCodeMap[icmp.icmptype].PortUnreachable
```

通过输入 `$ swyard -t testcases/router3_testscenario.srpy myrouter.py` 进行测试，结果如下：

```

    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.

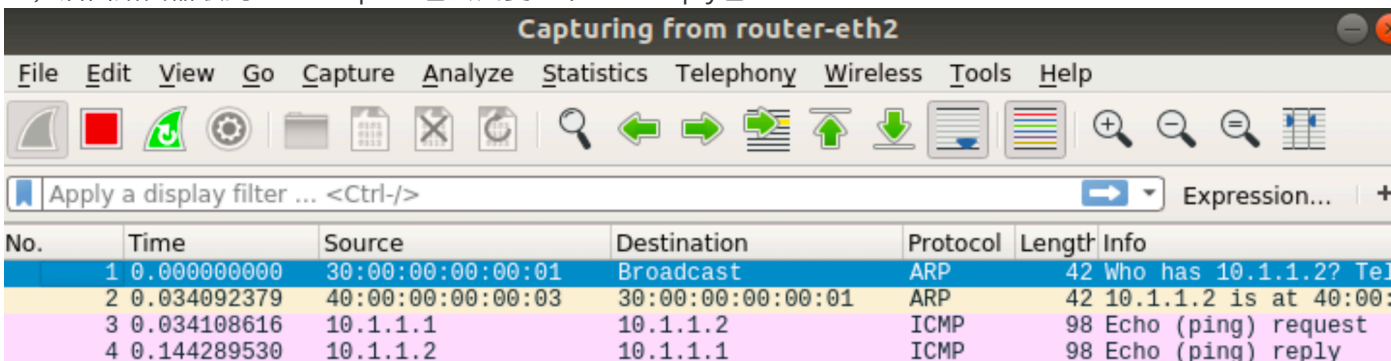
```

All tests passed!

可见，通过了全部测试

## mininet deployment

1、测试echo reply：输入 `client ping -c 1 10.1.1.2`，可见先开始发送了arp请求包询问10.1.1.2的mac地址，后面路由器收到echo request包会回复一个echo reply包



The screenshot shows the Wireshark interface with the title 'Capturing from router-eth2'. The packet list table is as follows:

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	0.034092379	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	0.034108616	10.1.1.1	10.1.1.2	ICMP	98	Echo (ping) request
4	0.144289530	10.1.1.2	10.1.1.1	ICMP	98	Echo (ping) reply

2、测试时间超限错误：

输入 `client# ping -c 1 -t 1 192.168.200.1` 进行测试，没有arp请求包因为上一步已经知道了端口的mac地址，如下图所示，由eth2向client发送一个ICMP error message包

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) request 10.1.1.1
2	0.096276834	10.1.1.2	10.1.1.1	ICMP	70	Time-to-live exceeded
3	5.224132261	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
4	5.341902681	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、测试网络不可达错误：

输入 `client ping -c 1 172.16.1.1` 进行测试，往不在forwarding\_table里的ip地址发包，结果如下：

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.1	172.16.1.1	ICMP	98	Echo (ping) request 10.1.1.1
2	0.065579739	10.1.1.2	10.1.1.1	ICMP	70	Destination unreachable
3	5.105056542	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
4	5.197828756	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

可以发现，eth2回复了一个网络不可达错误包

### 4、traceroute测试：输入 `client# traceroute -N 1 -n 192.168.100.1`，测试结果如下：

```

root@njucs-VirtualBox:~/lab-5-bbzunyi# traceroute -N 1 -n 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  123.673 ms  106.056 ms  102.164 ms
 2  192.168.100.1  392.009 ms  212.753 ms  210.759 ms

```

可见上图符合拓扑路线： `client—router-eth2—server1`

## 4、实验感想

这一阶段比较简单，主要是前两个阶段把这次实验很多要写的代码都囊括了，这次实验只使用封装好的函数就行了。只是需要注意，如果之前的实验的函数没有封装，都在packet\_handle里面的话，可能会带来一些困难。通过这次实验，我明白了主函数的内容越少，对于读代码的人来说，是很愉快的，这次没有对ICMP错误信息进行封装，因为觉得封装也不会减少代码量，但是在下一阶段，将尽量做到让实现功能的函数看起来简洁一些。这一阶段的实验结束了，也学习到了阶段实验应该怎样进行，收获还是满满的。