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：

学院	计算机科学与技术系	专业（方向）	计算机科学与技术
学号	201220102	姓名	武雅琛
Email	201220102@smail.nju.edu.cn	开始/完成日期	2022.5.3

1. 实验名称

Respond to ICMP

2. 实验目的

- **进一步熟悉Switchyard框架。**

  通过自行学习Switchyard相关**API的实现和接口**进一步熟悉lab提供的Switchyard框架。

- **了解IPv4网络ICMP协议的内容和功能**

  根据实验手册和课本学习IP数据报的重要成员ICMP**报文结构**，并且了解它在网络层的**作用**和提供的服务。

- **学习路由器如何配合ICMP协议实现错误侦察与回报机制**

  在了解ICMP报文结构的基础上学习它如何与路由器配合在**网络层之间侦察错误并回报错误。**

3. 实验内容

task 1:Preparation

task 2:Responding to ICMP echo requests

- 这一步要求路由器实现ICMP报文在网络层**响应请求**的功能。也就是在ping**的底层功能**，**侦察网络连线和远程主机。**

  又分为三个任务：**分析ICMP请求报文分组**、**构造ICMP回复分组**和**发送ICMP回复分组。**

  最后一个任务在lab4中以子线程发包的形式实现，向上层提供发包接口，不再赘述。

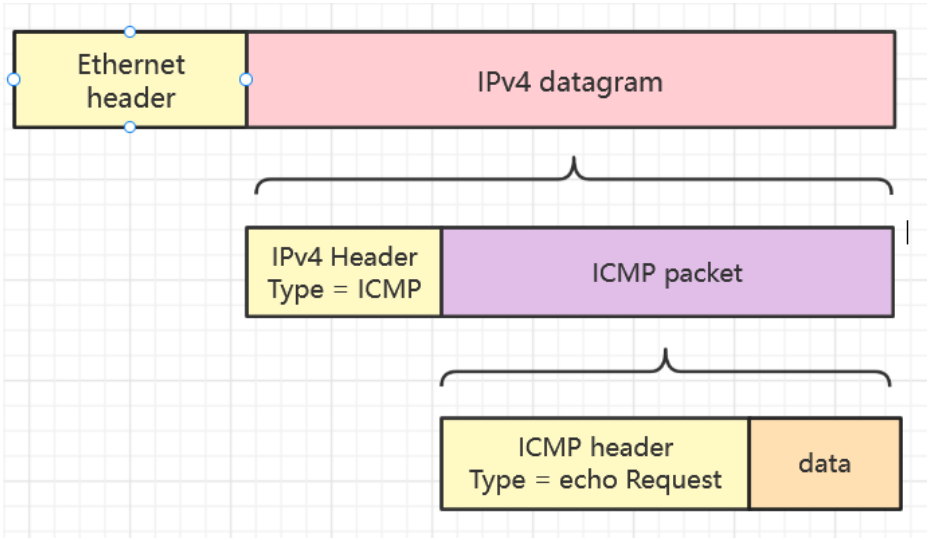
- **分析ICMP请求报文分组**
  - **判断ICMP请求报文分组**

    判断一个IPv4分组的**目的IP地址**是路由器某个**端口的IP地址**，并且恰好IPv4的**Type为ICMP请求**回应报文。那么就需要构造对应的回复分组。

```
 if IPv4:# IPv4 packet
     ...
     if IPv4.ttl > 0:
        ...
        if prefixlen != 0:#exist entry in forwarding_table
             if IPv4.dst not in my_ip:# not send to interfaces of
router
                 ...
             else: # send to interfaces of router
                  icmp_header = packet.get_header_by_name('ICMP')
                   if IPv4.protocol == IPProtocol.ICMP:#ICMP
                           if icmp_header.icmptype ==
ICMPType.EchoRequest:#if it is a ICMP echo
                              #符合要求，构造并发送ICMP回复
```

- **构造ICMP回复分组**
  - 根据ICMP协议，从逻辑上回复分组结构如下：

    即以太网包头封装IPv4数据报，IPv4数据报的**数据部分为ICMP控制报文。**



  - 在实现上需要Switchyard提供的API分别构造**以太网包头**，**IPv4包头**和**ICMP报文。**

    比较需要注意的是，IPv4包头和ICMP报文的内容需要根据**到来的请求分组**填写，而以太网包头主要是在**端口发送分组之前根据ARP缓存**的内容填写。

- **发送ICMP回复分组**

  这一部分直接将构造好的报文加入队列，等待子线程按照转发报文的方式从正确的端口发出即可。发包逻辑在lab4已经涉及。

task 3:Generating ICMP error messages

- 除了响应请求之外，ICMP协议在网络层还承担了网络层传输错误信息的功能，但是封装了ICMP报文的分组结构和task2提到的结构基本一致，不再赘述，可以通过修改ICMP报文头部**Type和code域**的内容来标识不同的错误类型。

  我们需要处理四种错误信息，分别讨论：

1.ICMP destination network unreachable

- 即对于到来的报文查询**转发表**后**没有**找到对应的**表项**，需要向源IP地址返回一个错误信息。
  - 错误的检测比较容易，在查询转发表后判断匹配的前缀长度，**如果**`prefix`**=0**，说明没有可以匹配的表项。
  - **IPv4 header**

    IPv4包头的source IP address 即为收到报文的端口地址，destination IP address即为错误分组的源IP地址。
  - **ICMP Packet**

    修改ICMP报文的**type**和**code**域对应到错误信息，并且注意同时拷贝错误分组的数据部分到回复分组中。

```
error_icmp_header.icmptype = ICMPType.DestinationUnreachable
        error_icmp_header.icmpcode = ICMPTypeCodeMap[ICMPType.DestinationUnreachable].NetworkUnreachable
        error_icmp_header.icmpdata.data = error_packet.to_bytes()[:28]
        log_info(f"{error_icmp_header.icmpdata.data}")
```

2. ICMP time exceeded

- 即到来的IPv4报文ttl - 1 = 0，即**跳数耗尽**，发生超时。
  - 在ttl - 1后对ttl作出判断，如果ttl <= 0，发送错误信息。
  - **IPv4 header**

    IPv4包头的source IP address 即为收到报文的端口地址，destination IP address即为错误分组的源IP地址。
  - **ICMP Packet**

    修改ICMP报文的**type**和**code**域对应到错误信息，并且注意同时拷贝错误分组的数据部分到回复分组中。

```
 error_icmp_header.icmptype = ICMPType.TimeExceeded
        error_icmp_header.icmpcode = ICMPTypeCodeMap[ICMPType.TimeExceeded].TTLExpired
        error_icmp_header.icmpdata.data = error_packet.to_bytes()[:28]
        log_info(f"{error_icmp_header.icmpdata.data}")
```

3.ICMP destination host unreachable

- 在子线程发送了五次ARP请求之后再次超时，没有收到ARP回复时，向源IP地址发送错误信息。
- 比较特殊的是，这里会在子线程发现错误，构造ICMP分组放入队列。
  - **IPv4 header**

    IPv4包头的source IP address 即为收到报文的端口地址，destination IP address即为错误分组的源IP地址。
  - **ICMP Packet**

    修改ICMP报文的**type**和**code**域对应到错误信息，并且注意同时拷贝错误分组的数据部分到回复分组中。

```
 error_icmp_header.icmptype = ICMPType.DestinationUnreachable
        error_icmp_header.icmpcode = ICMPTypeCodeMap[ICMPType.DestinationUnreachable].HostUnreachable
        error_icmp_header.icmpdata.data = error_packet.to_bytes()[:28]
        log_info(f"{error_icmp_header.icmpdata.data}")
```

4.ICMP destination port unreachable

- 即收到了发送到端口的非ICMP echo request报文，向源端口回复错误信息。
  - **IPv4 header**

    IPv4包头的source IP address 即为收到报文的端口地址，destination IP address即为错误分组的源IP地址。
  - **ICMP Packet**

    修改ICMP报文的**type**和**code**域对应到错误信息，并且注意同时拷贝错误分组的数据部分到回复分组中。

```
 error_icmp_header.icmptype = ICMPType.DestinationUnreachable
        error_icmp_header.icmpcode = ICMPTypeCodeMap[ICMPType.DestinationUnreachable].PortUnreachable
        error_icmp_header.icmpdata.data = error_packet.to_bytes()[:28]
        log_info(f"{error_icmp_header.icmpdata.data}")
```

4. 实验结果

task2 & task 3:Responding to ICMP echo requests & Generating ICMP error messages

Testing

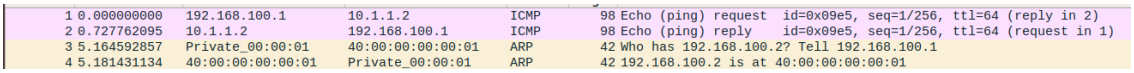
```
File  Edit  View  Search  Terminal  Help
    handle this type of packet and should generate an ICMP
    destination port unreachable error.
14  The router should send an ICMP destination port unreachable
    error back to 172.16.111.222 out router-eth1.
15  An IP packet from 192.168.1.239 for 10.10.50.250 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!
```

Deploying

- 创建默认的Mininet拓扑网络。
- 在Mininet中输入以下命令。
  - **reply**

```
mininet> server1 ping -c 1 10.1.1.2#即router和client在一个子网的端口地址
```

在server1-eth0端口打开wireshark捕获到分组，可以看到server1和IP地址为10.1.1.2的端口发生了依次应答。



  - **exceed time**

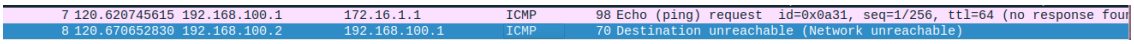
```
mininet> server1 ping -c 1 -t 1 10.1.1.2#即router和client在一个子网的端口地址
```

由于指定了发送的报文ttl为1，则必定会在router超时。

  - **Network unreachable**

```
mininet> server1 ping -c 1 -t 1 172.16.1.1#这是一个在router转发表中无法匹配到的IP地址
```

router会发送一个Network unreachable错误报文。

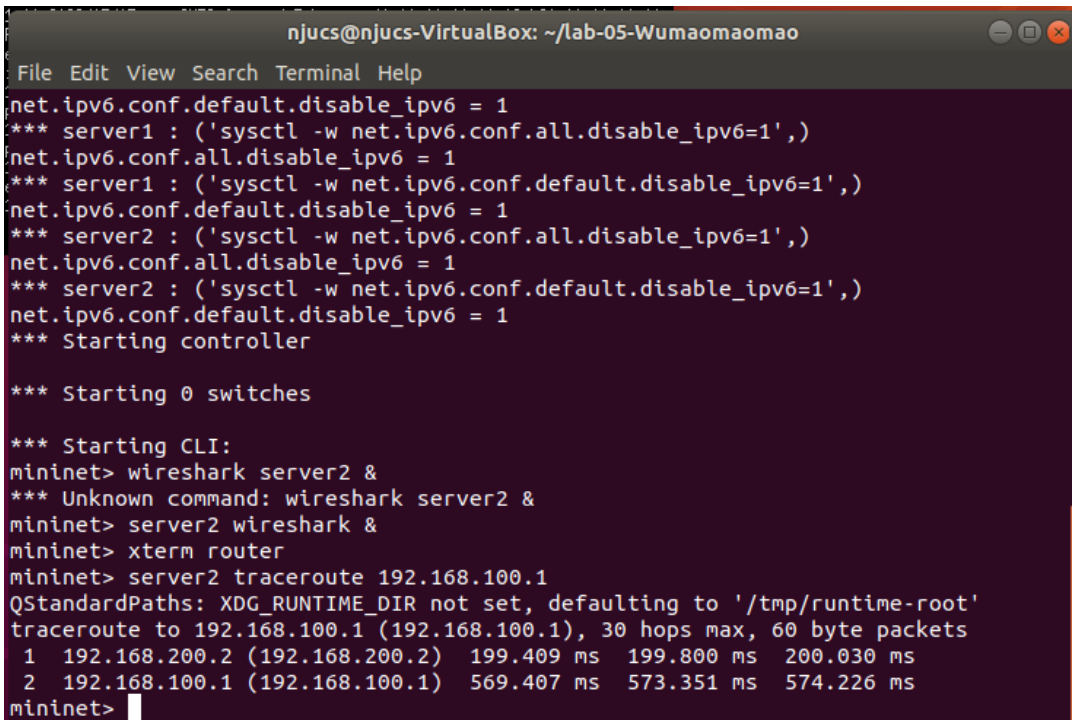


  - **traceroute**

    利用router的超时机制查找沿途的路由器IP地址。

```
mininet> server2 routetrace 192.168.100.1
```

    得到结果：



说明server2通过ttl = 1的ICMP报文首先访问到了192.168.200.2端口，然后增加ttl访问到了更远的192.168

100.1端口。

5.核心代码

task 2:Responding to ICMP echo requests

- **判断ICMP请求报文分组**

判断一个IPv4分组的**目的IP地址**是路由器某个**端口的IP地址**，并且恰好IPv4的**Type为ICMP请求**回应报文。那么就需要构造对应的回复分组。

```
if IPv4:# IPv4 packet
    ...
    if IPv4.ttl > 0:
       ...
       if prefixlen != 0:#exist entry in forwarding_table
            if IPv4.dst not in my_ip:# not send to interfaces of router
                ...
            else: # send to interfaces of router
                 icmp_header = packet.get_header_by_name('ICMP')
                  if IPv4.protocol == IPProtocol.ICMP:#ICMP
                         if icmp_header.icmptype ==
ICMPType.EchoRequest:#if it is a ICMP echo
                             #符合要求，构造并发送ICMP回复
```

- **构造ICMP echo reply分组**

```
#myrouter.py/create_icmp_echo_reply()
        # icmp header
        reply_icmp_header = ICMP()
        # log_info(f"{reply_icmp_header}")
        # log_info(f"{request_icmp_header.icmpdata.data}")
        reply_icmp_header.icmptype = ICMPType.EchoReply
        reply_icmp_header.icmpdata.sequence =
request_icmp_header.icmpdata.sequence
        reply_icmp_header.icmpdata.identifier =
request_icmp_header.icmpdata.identifier
        reply_icmp_header.icmpdata.data =
request_icmp_header.icmpdata.data
        # log_info(f"{reply_icmp_header.icmpdata.data}")
        # ipv4 header
        reply_ip_header = IPv4()
        reply_ip_header.src = request_ip_header.dst
        reply_ip_header.dst = request_ip_header.src
        reply_ip_header.protocol = IPProtocol.ICMP
        reply_ip_header.ttl = 64
        #Ethernet_header
        reply_Ethernet_header = Ethernet()
        reply_Ethernet_header.ethertype = EtherType.IPv4
        #Union packet
        reply_packet = Packet()
        reply_packet += reply_Ethernet_header
        reply_packet += reply_ip_header
        reply_packet += reply_icmp_header
```

task 3:Generating ICMP error messages

- 在构造错误信息回复的时候沿用了reply的大部分模板，有异的部分在实验内容部分给出，不再赘述。

1.ICMP destination network unreachable

```
#查找转发表，检查匹配的最大前缀长度，如果为0则匹配失败
prefixlen, nexthop, forward_intf =
self.forwarding_table.lookup(format(IPv4.dst))
                    if prefixlen != 0:#exist entry in forwarding_table
                    ...
                    else:#dst ip not in forwarding_table
                        intf = self.net.interface_by_name(ifaceName)
                        icmp_error_packet =
self.create_icmp_network_unreachable(packet, intf)

                        log_info(f"send a icmp error echo:
{icmp_error_packet}")
                        self.wait_queue_insert(icmp_error_packet, ifaceName)
```

2.ICMP time exceeded

```
if IPv4.ttl > 0:
...
else:#time exceed
    intf = self.net.interface_by_name(ifaceName)
    icmp_error_packet = self.create_icmp_timeexeceed(packet, intf)

    log_info(f"send a icmp error echo:{icmp_error_packet}")
    self.wait_queue_insert(icmp_error_packet,ifaceName)
```

3.ICMP destination host unreachable

```
 if timestamp + 1 < time.time():
        if (send_cnt < 5):
            ...
        else:#注意当一个IP过期的时候，对应的分组队列里所有的包都要返回一个错误信息
            for packet in wait_packet[nextip]:
                intf = packet[1]
                icmp_error_packet =
self.create_icmp_host_unreachable(packet[0], intf)
                log_info(f"send a icmp error echo:{icmp_error_packet}")
                self.wait_queue_insert(icmp_error_packet,intf.name)
             wait_packet.pop(nextip)
```

4.ICMP destination port unreachable

```
#对于发往路由器端口的分组：
if IPv4.protocol == IPProtocol.ICMP:
    if icmp_header.icmptype == ICMPType.EchoRequest:#if it is a ICMP echo

        ...
    else:#是ICMP报文但不是请求分组
```

```
            intf = self.net.interface_by_name(ifaceName)
            icmp_error_packet = self.create_icmp_port_unreacheable(packet,
intf)

            log_info(f"send a icmp error echo:{icmp_error_packet}")
            self.wait_queue_insert(icmp_error_packet,ifaceName)
else:#不是ICMP报文
           intf = self.net.interface_by_name(ifaceName)
           icmp_error_packet = self.create_icmp_port_unreacheable(packet,
intf)

           log_info(f"send a icmp error echo:{icmp_error_packet}")
            self.wait_queue_insert(icmp_error_packet,ifaceName)
```

6.总结与感想

- 深入了解了ICMP协议报文结构，学习了路由器和ICMP如何配合实现错误信息的传输。