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	**计算机科学与技术系**	专业（方向）	**计算机科学与技术**
学号	**221220143**	姓名	**陈雨檬**
Email	**221220143@smail.nju.edu.cn**	开始/完成日期	**2024.5.07/2024.5.12**

1. 实验名称

Lab5: Respond to ICMP

2. 实验目的

实现交换机响应 ICMP 消息的功能，回复 EchoRequest；在必要时生成 ICMP 错误消息。

3. 实验内容

Task1：Preparation

How:无

Task2：响应 ICMP 请求

How: （说明有关响应 ICMP echo requests 的逻辑）收到一个 ICMP 包时经过检查（此为 Task3 中的要求）后对 router 接口的 ip 地址进行查找并找到，且此包的确是 EchoRequest 类型包，且 TTL 仍未过期，则确认收到需要回答的 ICMPEchoRequest 包。对需要回复的 ICMP 包创建一个 EchoReply 的报文，并进行查找、转发。具体代码实现过程见第 **5** 节代码内容。

Task3：生成 ICMP 错误信息报文

How: （生成 ICMP 错误消息的逻辑）首先对 ip 包的目标 ip 进行转发表查找，如果没有找到则生成"Network Unreachable"错误讯息并转入错误处理函数；否则则进行检查是否是需要回答的 EchoRequest 报文，如果目标 ip 在 router 接口的 ip 中而包却不是 EchoRequest 包则生成“ICMP destination port unreachable”错误消息并转入错误处理函数；否则检查 ttl，如果 ttl 过期则生成“ICMP time exceeded”错误消息并转入错误处理函数。而对于需要重发的 ARP 报文，如果重发 5 次后仍然没有得到回复则生成"ICMP destination host unreachable"错误讯息并转入错误处理函数。错误处理函数的具体实现见第 5 节代码内容。

（为什么不应生成 ICMP 错误消息来响应任何 ICMP 错误消息）

如果路由器对 ICMP 错误消息中的错误仍产生回应，那么两个路由器之间可能会重复产生 ICMP 错误消息并始终在讯道中传递而无法被消除。

4. 实验结果

测试样例的测试结果：

```
swyard -t testcases/testscenario3.srpy myrouter.py
```

```
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
21:43:34 2024/05/12  WARNING Tried to find non-existent
ting <class 'switchyard.lib.packet.tcp.TCP'> (test scer
g)

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!
```

再进行部署测试:

首先使用 server1 ping router 与 server1 相连的接口 192.168.100.2，server1 的接口收到了一个 EchoReply，说明 router 进行了回复。

1	0.000000000	192.168.100.1	192.168.100.2	ICMP	98 Echo (ping) request id=0x0c6d, seq=1/256, ttl=64 (reply in 2)
2	0.106195449	192.168.100.2	192.168.100.1	ICMP	98 Echo (ping) reply id=0x0c6d, seq=1/256, ttl=64 (request in 1)
3	5.222887869	Private_00:00:01	40:00:00:00:00:01	ARP	42 Who has 192.168.100.2? Tell 192.168.100.1
4	5.251769681	40:00:00:00:00:01	Private_00:00:01	ARP	42 192.168.100.2 is at 40:00:00:00:00:01

随后设置使用 server1 ping server2 并设置 ttl 为 1，收到了一条超时报错信息。

7	191.504216889	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	191.607636137	192.168.100.2	192.168.100.1	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
9	196.710164005	Private_00:00:01	40:00:00:00:00:01	ARP	42 Who has 192.168.100.2? Tell 192.168.100.1
10	196.731979960	40:00:00:00:00:01	Private_00:00:01	ARP	42 192.168.100.2 is at 40:00:00:00:00:01

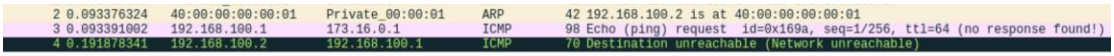
▸ Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▸ Ethernet II, Src: 40:00:00:00:00:01 (40:00:00:00:00:01), Dst: Private_00:00:01 (10:00:00:00:00:01)

（上述两图见 lab_5_test1.pcapng）

在 start_mininet.py 中插入语句

```
set_route(net, 'server1', '173.16.0.0/24', '192.168.100.2')
```

随后再使用 server1 ping 173.16.0.1，收到 ICMP 转发错误讯息。



（见 lab_5_test2.pcapng）

随后使用 server1 进行 traceroute 192.168.200.1

5. 核心代码

Task2:

```
def create_icmp_reply(self,ipv4,icmp_request):
    icmp_reply=ICMP()
    icmp_reply.icmptype=ICMPType.EchoReply
    icmp_reply.icmpdata.data=icmp_request.icmpdata.data
    icmp_reply.icmpdata.identifier=icmp_request.icmpdata.identifier
    icmp_reply.icmpdata.sequence=icmp_request.icmpdata.sequence
    ip_reply=IPv4()
    ip_reply.protocol=IPProtocol.ICMP
    ip_reply.ttl=64
    ip_reply.src=ipv4.dst
    ip_reply.dst=ipv4.src
    eth_reply=Ethernet()
    eth_reply.ethertype=EtherType.IP
    reply_icmp=eth_reply+ip_reply+icmp_reply
    return reply_icmp
```

该函数定义了如何创建一个 EchoReply 包回复 ICMPRequest；

在确认收到 ICMP Request 且无错误后，调用此函数创建一个 Reply 包并调用 lab4 中已实现的 send_pkt 方法发出。

```
def send_pkt(self,pkt,interface):
    log_info("send normal packet")
    dst=pkt.get_header(IPv4).dst
    forward_item=self.forward_record.find_match(dst)
    log_info(forward_item)
    if forward_item.next_ip is None:
        nextip=dst
    else:
        nextip=forward_item.next_ip
    port=self.net.interface_by_name(forward_item.portname)
    nextmac=self.record.get_item(nextip)
    if nextmac is not None:
        pkt[0].dst=nextmac
        pkt[0].src=port.ethaddr
        self.net.send_packet(port,pkt)
    else:
        arp_request = create_ip_arp_request(port.ethaddr,port.ipaddr,nextip)
        self.arpqueue.add_item(arp_request,pkt,port,interface)
```

Task3:

首先说明 router 逻辑：

```
forward_item=self.forward_record.find_match(ipv4.dst)
if forward_item is None:
    log_info("Network Unreachable")
    self.handle_icmp_error("ICMP destination network unreachable",packet,interface)
```

首先进行转发表查找，如果未找到则报错转入错误处理

```
flagicmp=False
for face in self.interfaces:
    if ipv4.dst==face.ipaddr:
        icmp_request=packet.get_header(ICMP)
        if icmp_request is not None and icmp_request.icmptype==ICMPType.EchoRequest:
            flagicmp=True
        else:
            log_info("port unreachable")
            self.handle_icmp_error("ICMP destination port unreachable",packet,interface)
            return
if flagicmp:#icmprequest for me
    reply_icmp=self.create_icmp_reply(ipv4,icmp_request)
    self.send_pkt(reply_icmp,interface)
```

然后进行查找是否是需要回复的 Request 包，如果目标是端口 ip 却不是 Request 则转入错误处理

```
else:
    if packet[i].ttl<=1:
        log_info("TTl -")
        self.handle_icmp_error("ICMP time exceeded",packet,interface)
    else:
        packet[i].ttl-=1
        self.send_pkt(packet,interface)
```

最后再检查 ttl，如果错误则转入 ttl 错误处理。

```
now=time.time()
for it in self.arpqueue.arp_q:
    if it.times>=5:
        for pkt in it.pkt:
            self.handle_icmp_error("ICMP destination host unreachable",pkt,it.interface)
        self.arpqueue.arp_q.remove(it)
    elif now-it.time>1.0:
        self.net.send_packet(it.port,it.arp)
        it.time=time.time()
        it.times+=1
```

同时在 lab4 实现的 resend 中修改代码，添加超送 5 次后的错误处理。

```
def handle_icmp_error(self,error_msg,origin_packet,interface):
    log_info("Handle error")
    icmp=origin_packet.get_header(ICMP)
    if icmp is not None and (icmp.icmptype==3 or icmp.icmptype==11 or icmp.icmptype==12):
        log_info(icmp.icmptype)
        return
    log_info(origin_packet.get_header(IPv4).src)
    pkt_error=self.create_icmp_error(error_msg,origin_packet,interface)
    log_info(pkt_error.get_header(IPv4).dst)
    self.send_error_pkt(pkt_error,interface)
```

该 Handle 函数首先判断错误包是否是 ICMP 错误信息包，如果是则不再进行错误处理；否则先创建一个错误信息包，再通过 send_error_pkt 函数发送错误信息包。

```
def create_icmp_error(self,error_msg,origin_packet,interface):
    icmp_error=ICMP()
    if error_msg=="ICMP destination network unreachable":
        icmp_error.icmptype=ICMPType.DestinationUnreachable
        icmp_error.icmpcode=0
    elif error_msg=="ICMP time exceeded":
        icmp_error.icmptype=ICMPType.TimeExceeded
    elif error_msg=="ICMP destination host unreachable":
        icmp_error.icmptype=ICMPType.DestinationUnreachable
        icmp_error.icmpcode=1
    elif error_msg=="ICMP destination port unreachable":
        icmp_error.icmptype=ICMPType.DestinationUnreachable
        icmp_error.icmpcode=3
    i=origin_packet.get_header_index(Ethernet)
    del origin_packet[i]
    icmp_error.icmpdata.data=origin_packet.to_bytes()[:28]
    icmp_error.icmpdata.origdgramlen = len(origin_packet)
    ip_error=IPv4()
    ip_error.protocol=IPProtocol.ICMP
    ip_error.ttl=32
    ipv4=origin_packet.get_header(IPv4)
    ip_error.dst=ipv4.src
    eth_error=Ethernet()
    eth_error.ethertype=EtherType.IP
    pkt_error=eth_error+ip_error+icmp_error
    return pkt_error
```

创建错误包的过程中首先根据传入的 error_message 判断发生错误的类型并赋予不同 icmptype 和 icmpcode 值，随后再进行相应 data 的拷贝的头部内容的填写。

```
def send_error_pkt(self,error_pkt,interface):
    log_info("send error reply")
    ip_dst=error_pkt.get_header(IPv4).dst
    forward_item=self.forward_record.find_match(ip_dst)
    if forward_item is None:
        pass
    else:
        if forward_item.next_ip is None:
            nextip=ip_dst
        else:
            nextip=forward_item.next_ip
        port=self.net.interface_by_name(forward_item.portname)
        nextmac=self.record.get_item(nextip)
        i=error_pkt.get_header_index(IPv4)
        error_pkt[i].src=port.ipaddr
        if nextmac is not None:
            error_pkt[0].dst=nextmac
            error_pkt[0].src=port.ethaddr
            self.net.send_packet(port,error_pkt)
        else:
            arp_request = create_ip_arp_request(port.ethaddr,port.ipaddr,nextip)
            self.arpqueue.add_item(arp_request,error_pkt,port,interface)
```

错误包发送函数与普通包发送函数大致相同，不过增加了 ip 头部内容信息的填写。

6. 思考与感受

至此，我们已经基本完成了一个具有基础内容的 router，这对我们认识网络中包的转发过程具有重要意义。