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**

学院	计算机科学与技术系	专业（方向）	计算机科学与技术系
学号	181860109	姓名	吴润泽
Email	181860109@smail.nju.edu.cn	开始/完成日期	2020/4/22-2020/4/24

1. 实验名称： Respond to ICMP

2. 实验目的

1. 掌握差错控制报文协议

---

3. 实验过程

Task 2 响应ICMP请求

a. 实现原理

1. 收到的IPv4报文目的地址是路由器的某一端口并且是ICMP请求报文，则进行响应；
2. 其回复报文中，发送地址为请求报文中的目的地址，并构造相应的回复报文；
3. 将报文转交给转发IP报文的函数即可，在lab4中已经实现。

b. 代码编写

```
def forward_packet(self, port, packet):#接收数据报
        ...
        elif packet[Ethernet].ethertype == EtherType.IPv4:
            if packet[IPv4].dst in self.ip_mac: #目的地址是端口
                if packet[IPv4].protocol == IPProtocol.ICMP:
                    #收到ICMP 请求报文
                    if packet[ICMP].icmptype == ICMPType.EchoRequest:
                        self.Icmp_reply(packet, True, port)
def Icmp_reply(self, packet, legal=True, port=None):
    #legal 来标识是否是合理的请求报文
        ...#对原请求报文的预处理
        icmp = ICMP()
        ip = IPv4(protocol=IPProtocol.ICMP,
                  ttl=10,
                  dst=packet[IPv4].src)
        if legal:
            ... #ICMP 报文的填写
            ip.src = packet[IPv4].dst # IP报文源地址为请求报文的目的地址
        reply_pkt = Ethernet() + ip + icmp #构造待发送的数据报
        #转交给发送IP报文的函数进行处理即可
        self.process_IP_Packet(reply_pkt, True,legal == False, port)
```

Task3 产生ICMP差错报文

a. 实现原理

1. 每当到达一个IP数据包，则将其目的地址与转发表匹配，如果匹配失败则发送ICMP目标网络不可达报文，并根据原报文增加相关信息以及对应的ICMP类型和类

型号；
2. 如果匹配成功，但是该报文的ttl字段已经小于2，则发送ICMP ttl过期的报文；
3. 如果对一个地址的ARP查询达到5次后仍没有回复，则发送ICMP目标主机不可达报文；
4. 目的地址是路由器端口，但是IP报文不上ICMP请求类型，则发送ICMP目的端口不可达报文；
5. 如果ICMP差错报文的目的地址（即原报文的源地址）没有匹配项，则直接将差错报文从接收到原报文的端口发出；
6. 对于所有的差错类型的报文，将其进行标记，在发送该报文时，将IP头的发送地址赋值为发出端口的发送地址。

H5 b. 代码编写

沿用lab4的结构，将处理数据报和待查询ARP队列分开。同时在队列的原有数据中额外记录是否为差错报文以及原报文的接收端口。

```
class PktCache:
    def AddPacket(self,src_mac,src_ip,dst_ip,packet,port,
                  IsErrorIcmp=False,#是否是差错报文
                  come_port=None):#原报文的接收端口
        if dst_ip not in self.cache_packet:
            self.cache_packet[dst_ip] = list()
            self.cache_packet[dst_ip].append(
                [time.time(), 1, src_ip, src_mac, port])
        self.cache_packet[dst_ip].append((packet, IsErrorIcmp,
come_port))
        return (create_ip_arp_request(src_mac, src_ip, dst_ip), port)
```

构造差错报文的代码，情况与构造ICMP回复报文类似，且手册中有示例代码，故不在报告中列出。

H6 case 1：ARP报文发送达上限

```
def GapArpQuery(self):
        ...
        icmp_packets = list()# 缓存查询不到ARP的差错报文
        for item in cache_packet_list:
            ... #正常的ARP请求
            elif item[1][0][1] >= 5 or now - item[1][0][0] > 4.0:#已经
超过发送上限
                for num in range(1, len(item[1])):
                    come_port=item[1][num][-1]#接收到原报文的端口
                    icmp_packets.append((self.GetArpIcmp(item[1][num]
[0]),come_port))
                    #进行ICMP目标主机不可达报文的构造，同时为了保证如果
与转发表匹配失败，从原端口发出，接收到原报文的端口也作为参数一并传递
                self.cache_packet.pop(item[0])
        return arp_packets, icmp_packets
```

H6 case2： 目的端口不可达

```
def forward_packet(self, port, packet):#接收数据报
        ...
        elif packet[Ethernet].ethertype == EtherType.IPv4:
            if packet[IPv4].dst in self.ip_mac: #目的地址是端口
                ...#收到ICMP请求报文的处理
                self.Icmp_reply(packet, False, port)#否则发送目的端口不
可达报文
```

H6 case3、4： ttl超时和匹配表项失败

```
 def process_IP_Packet(self, packet, IsIcmp=False,IsErrorIcmp=False,
port=None):
        #port记录了原始数据报的接收端口，两个标志位用来与普通的数据报进
行区分
        dst_ip = packet[IPv4].dst
        tar_route = self.match_subnet(dst_ip)
        if tar_route is None: #匹配失败
            if IsIcmp:#如果已经是构造出的ICMP报文，则从接收端口发出
                self.IP_forward(packet,port,'ff:ff:ff:ff:ff:ff',True)
            else:# 目的网络不可达报文
                self.route_no_match(packet,port)
            return
        else:
            if packet[IPv4].ttl <= 1:#ttl 过期报文
                self.ttl_ended(packet,port)
                return
```

H5 c. 实现测试

H6 I. test scenario测试

给定测试文件测试结果

运行$swyard\ -t\ routertests2.srpy\ myrouter.py$，结果如下：

```
Results for test scenario IP forwarding and ARP requester tests: 28 passed, 0 failed, 0 pending


Passed:
1   ICMP echo request (PING) for the router IP address
    192.168.1.1 should arrive on router-eth0.  This PING is
    directed at the router, and the router should respond with
    an ICMP echo reply.
2   Router should send an ARP request for 10.10.1.254 out
    router-eth1.
3   Router should receive ARP reply for 10.10.1.254 on router-
    eth1.
4   Router should send ICMP echo reply (PING) to 172.16.111.222
    out router-eth1 (that's right: ping reply goes out a
    different interface than the request).
5   ICMP echo request (PING) for the router IP address 10.10.0.1
    should arrive on router-eth1.
6   Router should send ICMP echo reply (PING) to 172.16.111.222
    out router-eth1.
7   ICMP echo request (PING) for 10.100.1.1 with a TTL of 1
    should arrive on router-eth1.  The router should decrement
    the TTL to 0 then see that the packet has "expired" and
    generate an ICMP time exceeded error.
8   Router should send ARP request for 10.10.123.123 out router-
    eth1.
9   Router should receive ARP reply for 10.10.123.123 on router-
    eth1.
10  Router should send ICMP time exceeded error back to
    10.10.123.123 on router-eth1.
11  A packet to be forwarded to 1.2.3.4 should arrive on router-
    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
```

测试文档说明

路由器端口沿用所给定的测试模板，同时测试代码行数较多且较为相似，故仅说明其主要功能。

测试代码检测以下方面的正确性：

1. 路由器端口可以发送正确的ICMP回复；
2. 四种情况的差错报文的正确构造和发送；

- case 1：发送的目的 ip 为路由器端口，并且是ICMP请求报文；
- case 2：发送的目的 ip 为路由器端口，但不是ICMP请求报文，发送对应的差错报文；

- case 3：接收数据报的ttl小于2，应该发送ttl过期报文；
- case 4：匹配路由器表项失败，同时ttl小于2，应该发送目的网路不可达报文；
- case 5：目的地址相同的多个数据包到达路由器，，在发送5次ARP差错报文后，仍为收到回复，对于每个待发送的数据包都应构造相应的目的主机不可达报文。

测试结果

运行$swyard\ -t\ lab_5routertests.py\ myrouter_to$，结果如下：

```
Results for test scenario IP forwarding and ARP requester tests: 33 passed, 0 failed, 0 pending


Passed:
1   send a ping request to 172.16.42.1(interface eth2) arrive on
    router-eth0
2   send Arp request for 10.10.1.254 leave out on router-eth1
3   Router receive an ARP response for 10.10.0.254 on router-
    eth1 and prepare send ping reply to 10.10.1.254
4   router eth1 send the icmp reply to 10.10.0.254
5   send a ping reply to 172.16.42.1(interface eth2) arrive on
    router-eth0
6   router eth1 send the icmp error packet to 10.10.0.254
7   send a ping reply to 192.168.1.1(interface eth0) arrive on
    router-eth0
8   router eth1 send the icmp error packet to 10.10.0.254
9   send a ping reply to 3.3.3.3(an error dest) arrive on
    router-eth0
10  router eth1 send the icmp error packet to 10.10.0.254
11  172.16.42.2 ping for 172.16.64.1 arrive on router-eth2
12  look up nexthop:10.10.1.254's mac
13  10.10.1.254 ping for 172.16.64.1 arrive on router-eth1
14  waiting for arp reply
15  repeat send arp request for nexthop:10.10.1.254's mac
16  172.16.42.2 ping for 172.16.64.1 arrive on router-eth2
17  waiting for arp reply
18  repeat send arp request for nexthop:10.10.1.254's mac
19  waiting for arp reply
20  repeat send arp request for nexthop:10.10.1.254's mac
21  waiting for arp reply
22  repeat send arp request for nexthop:10.10.1.254's mac
23  waiting for arp reply
24  send arp request for 172.16.42.2's mac
25  send arp request for 10.10.1.254's mac
26  waiting for arp reply
27  send arp request for 172.16.42.2's mac
28  send arp request for 10.10.1.254's mac
29  172.16.42.2 arp reply arrive an router-eth2
30  send HostUnreachable ICMP packet to 172.16.42.2
31  send HostUnreachable ICMP packet to 172.16.42.2
32  10.10.1.254 arp reply arrive an router-eth1
33  send HostUnreachable ICMP packet to 10.10.1.254


All tests passed!
```

H6 II. 抓包测试

在$start_mininet$中，构造的topo结构中，只有 $client$ 的可达子网中存在与转发表项不匹配的情况。如果在 $server1$ 或者 $server2$ 上无法构造目的地址不在转发表中的情况，（因为如果目的地址主机自身转发表都不匹配的情况下，不会发送给路由器），因此需要为主机添加路由项。

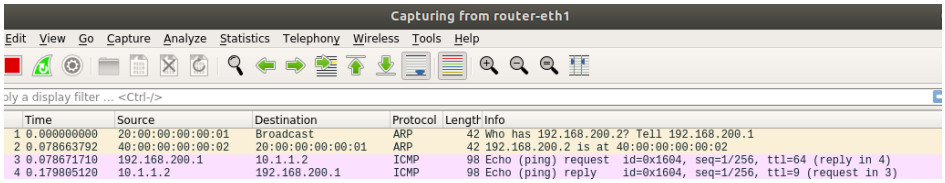
本次抓包测试中，选择server2进行测试，故对其添加路由项，发往 $7.7.0.0/16$ 的数据报首先发往 $192.168.200.2$ 端口。

```
    set_route(net,'server2','7.7.0.0/16','192.168.200.2')
```

运行mininet，开启router运行$swyard\ myrouter.py$，并在wireshark中监视 $eth1$和$eth2$ 的抓包情况；

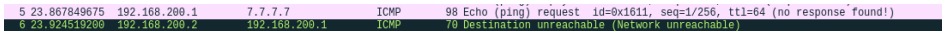
运行$xterm\ server2$，在 $server2$ 中进行通信测试。

- **case 1：** $ping\ -c\ 1\ 10.1.1.2$，对路由器端口发送ICMP请求，观察两个端口抓包结果：

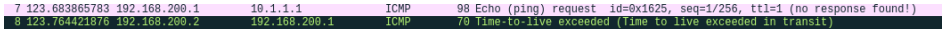


其中 $eth2$ 没有报文发送和接收故没有记录；对于 $eth1$ ， $server2$ 首先应查询其应发送到的路由器端口的mac地址，故发送ARP请求，$eth1$ 接收后进行相应的回复。$server2$ 收到ARP回复后，便将ICMP请求报发给路由器，并接收到从 $eth1$ 发出的源地址为 $eth2$ 的ICMP回复报文。

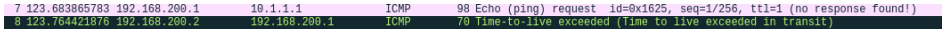
- **case 2：** $ping -c 1\ 7.7.7.7$，路由器没有匹配的表项，应发送目的网路不可达报文。



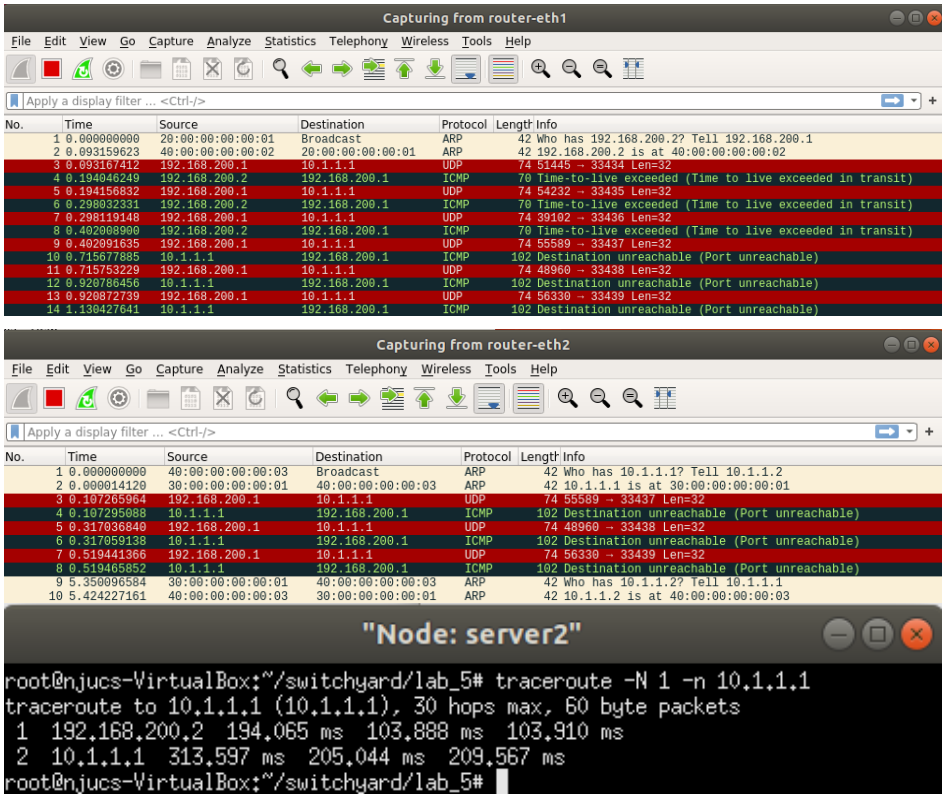
- **case 3：** $ping -c 1\ -t\ 1\ 10.1.1.1$，ttl为1，故应发送ttl过期报文。



- **case 4：** $ping -c 1\ 10.1.1.1$，与 $client$ 进行正常通信。



- **case 5：** $traceroute -N\ 1\ -n\ 10.1.1.1$，追踪到达 $10.1.1.1$ 的途径的路由信息。



traceroute逐次提高ttl值来逐一确定下一跳的地址，直到到达目的地址，但由于UDP中设置的端口不合法，故又会发送端口不可达报文。每一次试探都会发送三次相同的UDP报文。

综上，路由器的ICMP差错控制报文协议验证符合期望。

4. 总结与感想

这次实验的难度有所下降，如果lab4的设计框架耦合足够松弛，相信需要添加的代码会更少。

实验过程中还出现了一个小插曲，因为某些神奇的原因导致虚拟机无法联网，在一顿操作后，成功将虚拟机整崩溃，报告的书写一度中止。重装之后才深刻体会到，备份和快照的重要性。

实验主要是学习了ICMP差错报文协议的基本原理和工作流程，以及由其产生的查看路由信息的工具traceroute。网络层作为框架重要一环，复杂程度是教材描述和实验体现的千倍，等待探索的领域让我着迷。实验的完成让人很有成就感，希望可以保持对这种知识探究的渴望和乐趣。

---

5. 文档结构