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	电子科学与工程学院	专业（方向）	电子信息科学与技术
学号	191180102	姓名	任晟昊
Email	191180102@smail.nju.edu.cn	开始/完成日期	2022.05.03-2022.05.04

实验名称

Lab5

实验目的

实现对ICMP包的处理，和对异常情况的处理

实验内容

实现对ICMP包的回复和处理，实现对异常错误的处理

实验结果

Task2

实现对ICMP echo request的回复，通过判断是否有IPv4头以及判断是否有ICMP头，再判断是否是`EchoRequest`，接着构建Ethernet ICMP IP包头。

```
if packet.has_header(IPv4):
    head = (packet[IPv4])

    if head.dst in self.ipList:
        log_info("IP has matched!")
        if packet.has_header(ICMP):
            if packet[ICMP].icmptype == ICMPType.EchoRequest:
                log_info("Received an ICMP request!")

                icmp = ICMP()
                icmp.icmptype = ICMPType.EchoReply
                icmp.icmpcode = ICMPCodeEchoReply.EchoReply
                icmp.icmpdata.sequence = packet[ICMP].icmpdata.sequence
                icmp.icmpdata.data = packet[ICMP].icmpdata.data

                ip = IPv4()
                ip.src = head.dst
                ip.dst = head.src
                ip.protocol = IPProtocol.ICMP
                ip.ttl = 64
                ip.ipid = 0
```

```
                ether = Ethernet()
                ether.ethertype = EtherType.IP

                packet = ether + ip + icmp
                head = (packet[IPv4])
```

Task3

当收到的IPv4包不是ICMP包头时，则发送ICMP error，同时标记`the ICMP type should be destination unreachable, and the ICMP code should be port unreachable.`

```
if packet.has_header(ICMP):
'''
'''
else:
    log_info("Not an ICMP packets!")

    ether = Ethernet()
    ether.src = packet[Ethernet].dst
    ether.dst = packet[Ethernet].src
    ether.ethertype = EtherType.IP

    i = packet.get_header_index(Ethernet)
    del packet[i]

    icmp = ICMP()
    icmp.icmptype = ICMPType.DestinationUnreachable
    icmp.icmpcode = ICMPCodeDestinationUnreachable.PortUnreachable
    icmp.icmpdata.data = packet.to_bytes()[:28]

    ip = IPv4()
    for i in self.interfaces:
        if i.name == ifaceName:
            ip.src = i.ipaddr
            break

    ip.dst = head.src
    ip.protocol = IPProtocol.ICMP
    ip.ttl = 64
    ip.ipid = 0

    packet = ether + ip + icmp
    head = packet[IPv4]
```

当无法在`forwardTable`中找到对应的匹配项时，应将 **ICMP 目标网络无法访问**的错误发送回 IP 数据包中的源地址引用的主机。`the ICMP type should be destination unreachable, and the ICMP code should be network unreachable.`此时再利用原算法查找发挥源地址主机的对应匹配项。

```
if best == -1:
    log_info("There is no match!")

    ether = Ethernet()
    ether.src = packet[Ethernet].dst
    ether.dst = packet[Ethernet].src
    ether.ethertype = EtherType.IP
```

```
    i = packet.get_header_index(Ethernet)
    del packet[i]

    icmp = ICMP()
    icmp.icmptype = ICMPType.DestinationUnreachable
    icmp.icmpcode = ICMPCodeDestinationUnreachable.NetworkUnreachable

    icmp.icmpdata.data = packet.to_bytes()[:28]

    ip = IPv4()
    for i in self.interfaces:
        if i.name == ifaceName:
            ip.src = i.ipaddr
            break

    ip.dst = head.src
    ip.protocol = IPProtocol.ICMP
    ip.ttl = 64
    ip.ipid = 0

    packet = ether + ip + icmp
    head = packet[IPv4]

    prefixlen = 0
    index = 0
    best = -1
    for i in self.forwardTable:
        if (int(head.dst) & int(i.mask)) == int(i.prefix):
            netprefix = IPv4Network(str(i.prefix)+"/"+str(i.mask))
            if netprefix.prefixlen > prefixlen:
                prefixlen = netprefix.prefixlen
                best = index
        index += 1
    queue.append(Node(packet,self.forwardTable[best],icmp_info=ifaceName))
```

在转发过程中递减 IP 数据包的 TTL 值后，TTL 变为零，此时应在这种情况下，**超出 ICMP 时间的**错误消息应发送回 IP 数据包中的源地址所引用的主机。`the ICMP code should be TTL expired`

```
head.ttl -= 1

if head.ttl <= 0:
    log_info("TTL decreased to zero!")

    ether = Ethernet()
    ether.src = packet[Ethernet].dst
    ether.dst = packet[Ethernet].src
    ether.ethertype = EtherType.IP

    i = packet.get_header_index(Ethernet)
    del packet[i]

    icmp = ICMP()
    icmp.icmptype = ICMPType.TimeExceeded
    icmp.icmpcode = ICMPCodeTimeExceeded.TTLExpired
    icmp.icmpdata.data = packet.to_bytes()[:28]
```

```
    ip = IPv4()
    for i in self.interfaces:
        if i.name == ifaceName:
            ip.src = i.ipaddr
            break

    ip.dst = head.src
    ip.protocol = IPProtocol.ICMP
    ip.ttl = 64
    ip.ipid = 0

    packet = ether + ip + icmp
    head = packet[IPv4]
    print("TimeExceed",packet)
```

如果在 5 次重新传输 ARP 请求后，路由器未收到 ARP 回复，则路由器应将**无法访问的 ICMP 目标主机**发送回 IP 数据包中源地址引用的主机。`the ICMP type should be destination unreachable, and the ICMP code should be host unreachable`

```
if queue[0].cnt >= 5:
    log_info("Count number bigger than five!")

    ether = Ethernet()
    ether.src = queue[0].packet[Ethernet].dst
    ether.dst = queue[0].packet[Ethernet].src
    ether.ethertype = EtherType.IP

    i = queue[0].packet.get_header_index(Ethernet)
    del queue[0].packet[i]

    icmp = ICMP()
    icmp.icmptype = ICMPType.DestinationUnreachable
    icmp.icmpcode = ICMPCodeDestinationUnreachable.HostUnreachable
    icmp.icmpdata.data = queue[0].packet.to_bytes()[:28]

    ip = IPv4()
    for i in self.interfaces:
        if i.name == queue[0].icmp_info:
            ip.src = i.ipaddr
            break

    ip.dst = queue[0].packet[IPv4].src
    ip.protocol = IPProtocol.ICMP
    ip.ttl = 64
    ip.ipid = 0

    packet = ether + ip + icmp
    head = packet[IPv4]
```

由于上述代码在Handle queue中，所以还要加上如下代码完成包的发送：

```
prefixlen = 0
index = 0
best = -1
for i in self.forwardTable:
    if (int(head.dst) & int(i.mask)) == int(i.prefix):
```

```
        netprefix = IPv4Network(str(i.prefix)+"/"+str(i.mask))
        if netprefix.prefixlen > prefixlen:
            prefixlen = netprefix.prefixlen
            best = index
    index += 1

immediate = Node(packet,self.forwardTable[best],queue[0].icmp_info)
del(queue[0])

for intf in self.interfaces:
    if intf.name == immediate.info.name:
        port = intf

if immediate.info.nexthop is None:
    targetIp = immediate.packet[IPv4].dst
else:
    targetIp = immediate.info.nexthop

tempFlag = 0

for (k,v) in self.arpTable.items():
    if k == targetIp:
        immediate.packet[Ethernet].dst = v
        immediate.packet[Ethernet].src = port.ethaddr
        self.net.send_packet(port,immediate.packet)
        tempFlag = 1
        break

if tempFlag == 0:
    ether = Ethernet()
    ether.src = port.ethaddr
    ether.dst = 'ff:ff:ff:ff:ff:ff'
    ether.ethertype = EtherType.ARP
    arp = Arp(operation=ArpOperation.Request,
            senderhwaddr=port.ethaddr,
            senderprotoaddr=port.ipaddr,
            targethwaddr='ff:ff:ff:ff:ff:ff',
            targetprotoaddr=targetIp)
    arppacket = ether + arp

    self.net.send_packet(port,arppacket)
    immediate.cnt += 1
    immediate.time = time.time()
    queue.append(immediate)
```

测试结果如下：

```
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
```

```
File  Edit  View  Search  Terminal  Help
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!
```

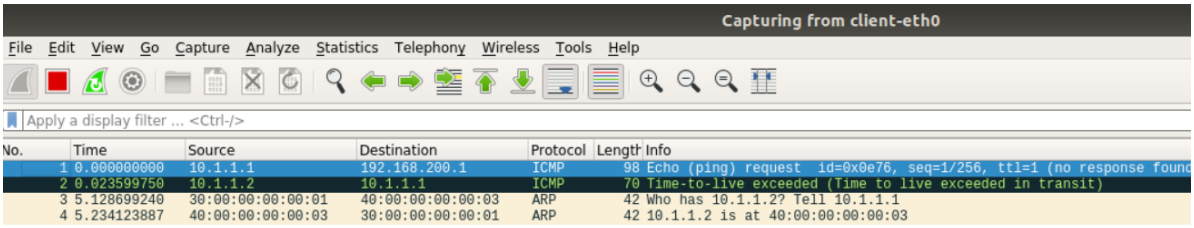
Deploy

首先在client ping server2 `ping -c 2 192.168.200.1`在router-eth1抓包，此时有如下包可见：

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:02	Broadcast	ARP	42	Who has 192.168.200.1? Tell 192.168.200.2
2	0.000020973	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	192.168.200.1 is at 20:00:00:00:00:01
3	0.108197862	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x0cd6, seq=1/256, ttl=63 (rep
4	0.108222009	192.168.200.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x0cd6, seq=1/256, ttl=64 (requ
5	0.956279575	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x0cd6, seq=2/512, ttl=63 (rep
6	0.956301061	192.168.200.1	10.1.1.1	ICMP	98	Echo (ping) reply id=0x0cd6, seq=2/512, ttl=64 (requ
7	5.144107557	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	Who has 192.168.200.2? Tell 192.168.200.1
8	5.240774346	40:00:00:00:00:02	20:00:00:00:00:01	ARP	42	192.168.200.2 is at 40:00:00:00:00:02

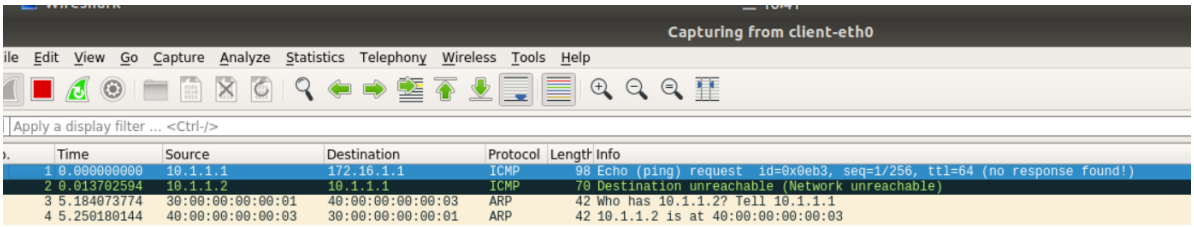
首先是，路由器收到client的arp包，再通过eth2进行回复，eth2收到来自client的ICMP包，此时要转发至server2，不知道server2的mac，所以在router-eth1发送arp包询问server2的mac，收到回答后，路由器则可以构建回复包，最后由于client不知道路由器192.168.200.2的mac地址，所以询问。

`client# ping -c 1 -t 1 192.168.200.1`



这里从10.1.1.2的路由器端口上发来，TTL exceeded错误类型。

```
client# ping -c 1 172.16.1.1
```



这里从10.1.1.2的路由器端口上发来，Destination unreachable(Network unreachable)错误类型。

```
client# traceroute 192.168.100.1
```

```
*** Starting CLI:
mininet> xterm router
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)  166.348 ms  169.148 ms  172.017 ms
 2  192.168.100.1 (192.168.100.1)  4538.956 ms * *
mininet>
```

```
*** Starting CLI:
mininet> xterm router
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)  165.911 ms  167.138 ms  168.536 ms
 2  192.168.100.1 (192.168.100.1)  4561.039 ms * *
mininet> server1 traceroute 192.168.100.1
traceroute to 192.168.100.1 (192.168.100.1), 30 hops max, 60 byte packets
 1  njucs-VirtualBox (192.168.100.1)  0.253 ms  0.227 ms  0.219 ms
mininet> server1 traceroute 10.1.1.1
traceroute to 10.1.1.1 (10.1.1.1), 30 hops max, 60 byte packets
 1  192.168.100.2 (192.168.100.2)  95.047 ms  97.047 ms  99.305 ms
 2  10.1.1.1 (10.1.1.1)  199.900 ms  202.231 ms  204.025 ms
mininet>
```

如上所示，使用server1和server2测试路由器的原理和上面一致，且基本功能均已实现完毕。

总结

本次实验主要难点在于整体代码逻辑和具体api的实现，例如在handle_queue()函数中需要在发送arp包大于五次的判断逻辑中，加入对arp包的处理代码，降低了代码的复用性，才能通过实验测试用例。同时要让icmptype先赋值，在赋值icmpcode不然也会出现错误。

在以后的编程中，应注意整体代码搭建思路的重要性，同时要对api比较熟悉，才能更专注于功能的实现上，在细节处减少错误。