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**	专业（方向）	**计算机科学与技术**
学号	**201220062**	姓名	**黄子睿**
Email	**201220062@smail.nju.edu.cn**	开始/完成日期	**4/20-4/22**

1. 实验名称：

Lab5 Respond to ICMP

2. 实验目的：

在 lab3 与 lab4 的基础上，实现路由器对 ICMP 的处理。包括 ICMP echo reply 的实现以及 ICMP 错误信息的生成与传输。

3. 实验内容：

Task2 Responding to ICMP echo requests

下面将阐述如何在 lab3 与 lab4 的基础上实现对 ICMP echo request 的处理。首先给出处理框架的伪代码。

```
1.  def handle_packet:
2.      pkt := get packet from net
3.      if pkt is an ARP packet then:
4.          arp := pkt.arp
5.          handle_arppacket(arp);
6.      else if pkt is an IPv4 packet then:
7.          delete pkt.ethernet
8.          updated_pkt, pktdel = self.handle_ippacket(packet)
9.              if pktdel != True then
10.              self.ipqueue.append({'packet':updated_pkt, 'del':False})
11.     else:
12.         Just drop this poacket.
13.
14. While free, try to renew packet in the ipqueue
```

通过调用 handle_ippacket 来处理所有 IPv4 类型的数据报。在这一阶段，我们仅仅需要关注与 ICMP echo request 有关的处理。下面给出这一部分的伪代码：

```
1.  def handle_ippacket:
2.      ipheader := pkt.IPv4_header
3.      if ipheader is None then
4.          Drop the packet and then return
5.      if ipheader.ttl <=0 then
6.          Just drop it for the time being, needs further implement
7.      forwardingIntf, next_hop := prefix_match(ipheader.dst)
8.      if forwadingIntf is None then
9.          Forward this packet, which has been implemented in lab4
10.     else if pkt is aimed at this router and pkt is an ICMP echo reque
    st then:
11.      icmp_pkt := handle_icmppacket(packet, ICMPType.EchoReply, 0)
12.      return handle_ippacket(icmp_pkt)
13.     else:
14.         Needs further implement to handle ICMP error messages
```

关于 handle_ippacket 函数需要补充对的一点是，它的返回值有两个，一个是在此时等待 ARP reply 而无法处理的 packet 与如果该包发送成功指示需要从 ipqueue 中删除的 del 布尔值。返回的 packet 可以被视作 updated_pkt，与原先的 pkt 相比，更新过的包可能是新生成的 ICMP echo reply 或 ICMP error message，即使不是，pkt 的 ttl 也会有所变化。handle_pkt 总会尝试发送，如果发送成功，即 arpCache 中可以匹配到 ip 对应的 mac 地址，则返回 None，True；否则发送 ARP request，并更新 ip_dst_map（一个队列，存储等待 ARP reply 的所有 ip 地址与对应的 cnt 及 time）中对应的表项。

注意到当 pkt 是发向这个路由器本身并且它是一个 ICMP echo request 时，才需要生成 ICMP echo reply。生成的包需要递归调用 handle_ipppacket 发送。而生成 ICMP echo reply 调用了函数 handle_icmppacket，这个函数设计出来用以生成符合要求的 ICMP 包，包括本阶段的 ICMP echo reply 与之后的 ICMP error message。这里给出生成 ICMP echo reply 的源代码。

```
def handle_icmppacket(oripkt, icmptype, icmpcode):
     oripkt_ip = oripkt.get_header(IPv4)
    oripkt_icmp = oripkt.get_header(ICMP)
    ip = IPv4(dst = oripkt_ip.src, src = oripkt_ip.dst, protocol = IPProtocol.ICMP, ttl = 64)
    # init ip header
    icmp = ICMP()
    # init icmp header
    if oripkt_icmp is not None and icmptype == ICMPType.EchoReply:
        icmp.icmptype = ICMPType.EchoReply
        icmp.icmpdata.data = oripkt_icmp.icmpdata.data
        icmp.icmpdata.identifier = oripkt_icmp.icmpdata.identifier
        icmp.icmpdata.sequence = oripkt_icmp.icmpdata.sequence
   else:
        Ignore ICMP error message for the time being
   return ip + icmp
```

Task3 Generating ICMP error message

在 Task2 完善的代码框架上，增加对 ICMP error message 的处理。显然这一部分功能只需要完善 handle_ippacket 与 handle_icmppacket 两个函数就可以了。之后的函数发送、查询工作可以在之前的基础上自动完成。

简而言之，这一任务阶段需要处理的 ICMP error message 有 4 种类型，

1、路由器没有对应的转发接口

2、ttl 降为 0，出现超时

3、5 次 ARP request 后还未收到 ARP reply

4、目的地址为该路由器但是不是 ICMP echo request

首先，需要补充 handle_icmppacket 函数，使得程序可以根据需要正确生成 ICMP error message 的包。具体的思路是，所有的 ICMP error message 都有一样的生成逻辑：

1、根据原来的包 oripkt 的源地址找到对应的转发接口 Intf，填充

icmp_error_pkt 对应 IP 头的源地址与目标地址, ip.src := Intf.ip , ip.dst := oripkt.src

2、根据传入的参数生成 icmptype 与 icmpcode

3、根据 oripkt 生成 28 字节的 raw payload

下面给出伪代码:

```
1.  def handle_icmppacket(oripkt, icmptype, icmpcode):
2.      oripkt_ip = oripkt.get_header(IPv4)
3.      oripkt_icmp = oripkt.get_header(ICMP)
4.      Init ip header ip
5.      Init icmp header icmp
6.      if oripkt_icmp is not None and icmptype == EchoReply:
7.          As implemented above
8.       else:   # handle ICMP error message
9.  +       get forwadingIntf by prefix_match(oripkt_ip.src)
10. +       ip.src := forwadingIntf.ip
11. +       ip.dst := oripkt_ip.ds
12. +       icmp.icmptype := icmptype
13. +       icmp.icmpcode := icmpcode
14. +       implement the 28 bytes of raw payload
15.     return ip + icmp
```

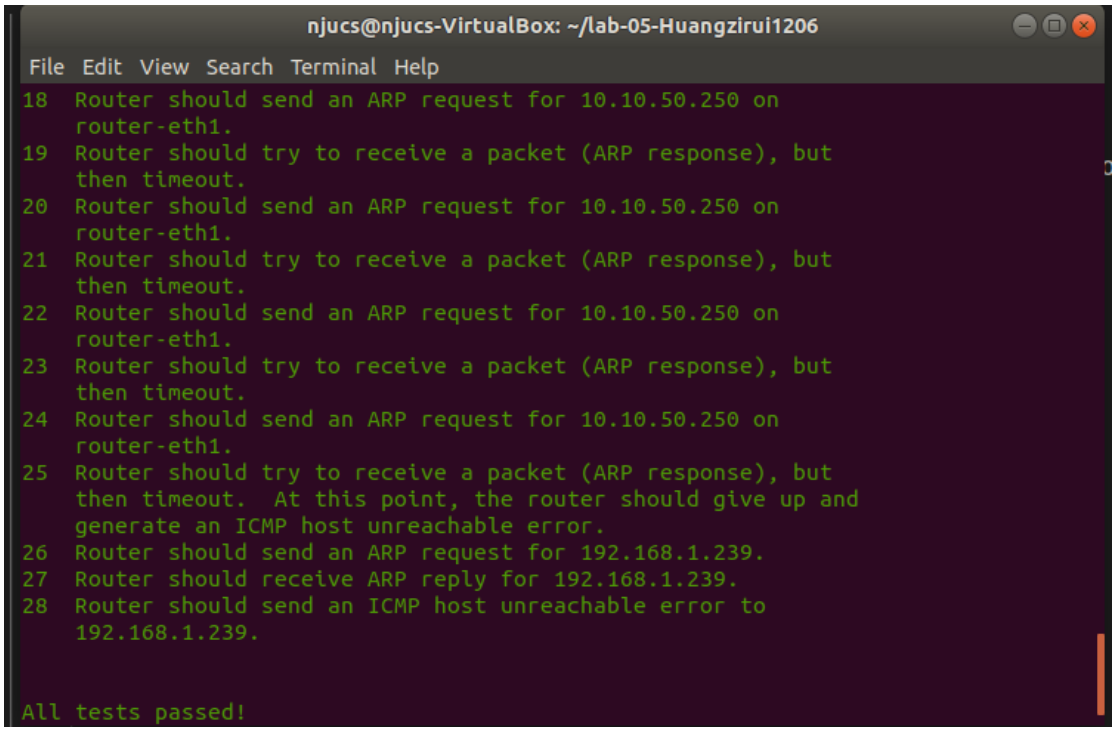
然后需要完善 handle_ippacket 函数, 在合适的地方生成 ICMP error packet,

下面给出伪代码:

```
1.  def handle_ippacket:
2.     ipheader := pkt.IPv4_header
3.     if ipheader is None then
4.        Drop the packet and then return
5.  +   if ipheader.ttl <=0 then  # ttl is less than 0
6.  +     icmp_pkt = self.handle_icmppacket(packet, TimeExceeded, 0)
7.  +     return self.handle_ippacket(icmp_pkt)
8.     forwardingIntf, next_hop := prefix_match(ipheader.dst)
9.     if forwadingIntf is None then
10.       Forward this packet, which has been implemented in lab4
11. +     if send over 5 ARP requests but no reply received then:
12. +       icmptype = ICMPType.DestinationUnreachable
13. +       icmpcode = ICMPTypeCodeMap[icmptype].HostUnreachable
14. +       icmp_pkt = self.handle_icmppacket(packet, icmptype, icmpcode)
15. +       return self.handle_ippacket(icmp_pkt)
16.    else if pkt is aimed at this router and pkt is an ICMP echo reques
    t then:
```

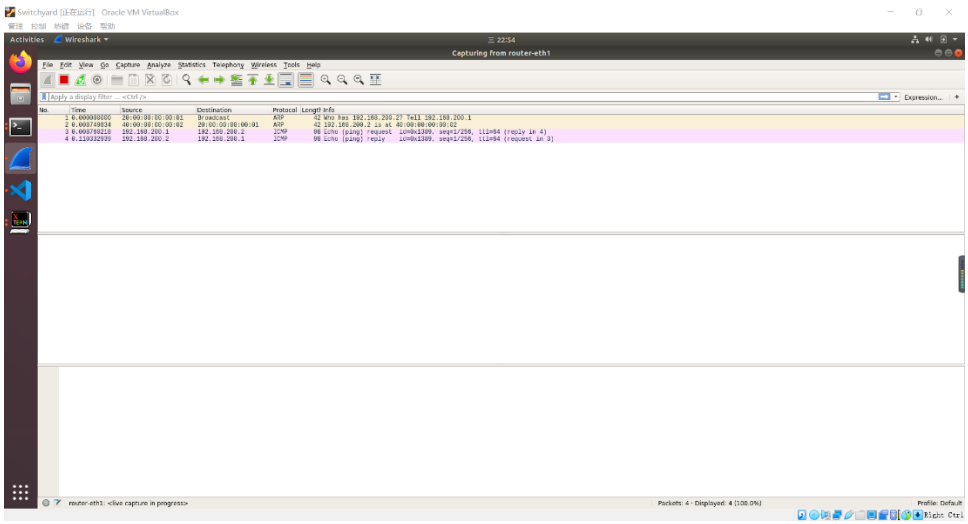
```
17.          icmp_pkt := handle_icmppacket(packet, ICMPType.EchoReply, 0)

18.          return handle_ippacket(icmp_pkt)
19. +  else if dst is not connected to the router then:
20. +      icmptype = ICMPType.DestinationUnreachable
21. +      icmpcode= ICMPTypeCodeMap[icmptype].NetworkUnreachable
22. +      icmp_pkt = self.handle_icmppacket(packet, icmptype, icmpcode)
23. +      return self.handle_ippacket(icmp_pkt)
24. +   else if pkt is aimed at this router but is not an ICMP echo reque
    st then:
25. +      icmptype = ICMPType.DestinationUnreachable
26. +      icmpcode = ICMPTypeCodeMap[icmptype].PortUnreachable
27. +      icmp_pkt = self.handle_icmppacket(packet, icmptype, icmpcode)
28. +      return self.handle_ippacket(icmp_pkt)
```

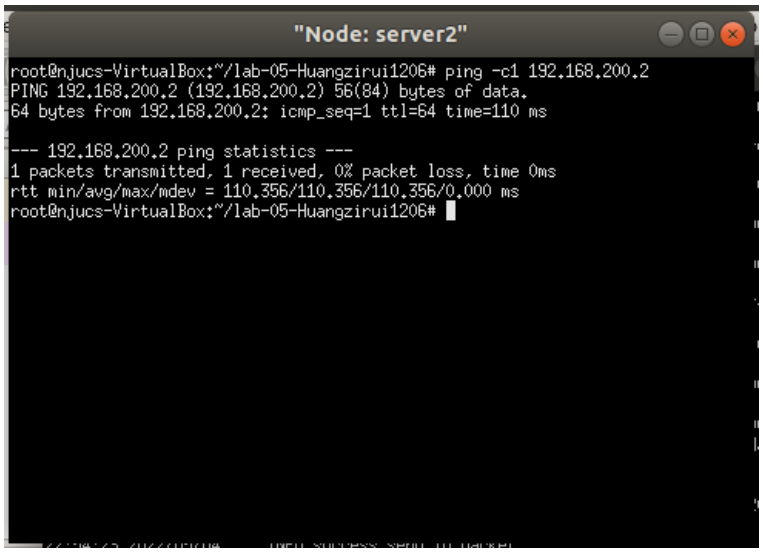
下面给出通过 test_senario 的截图：



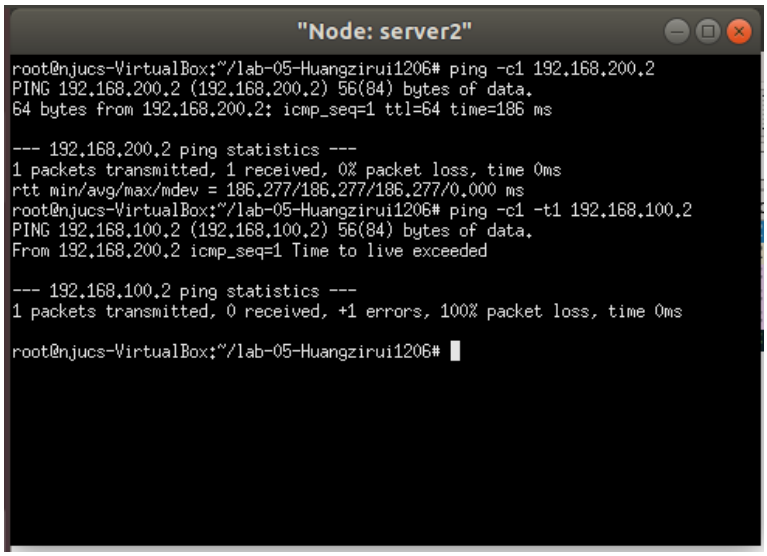
Deploy：

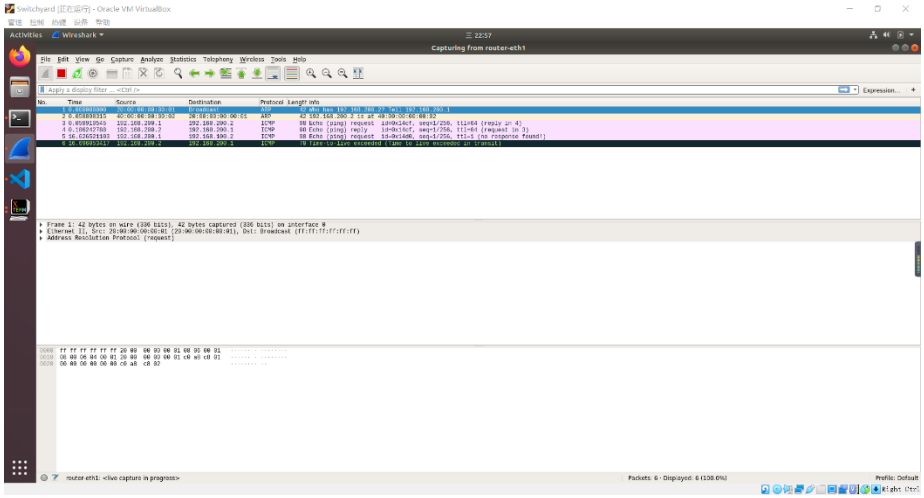
1、用 server2 ping 192.168.200.1（router_eth0）,结果如下所示：



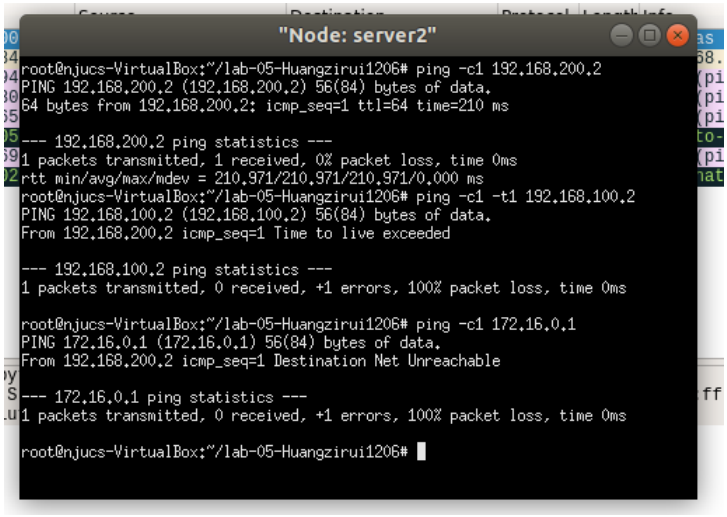


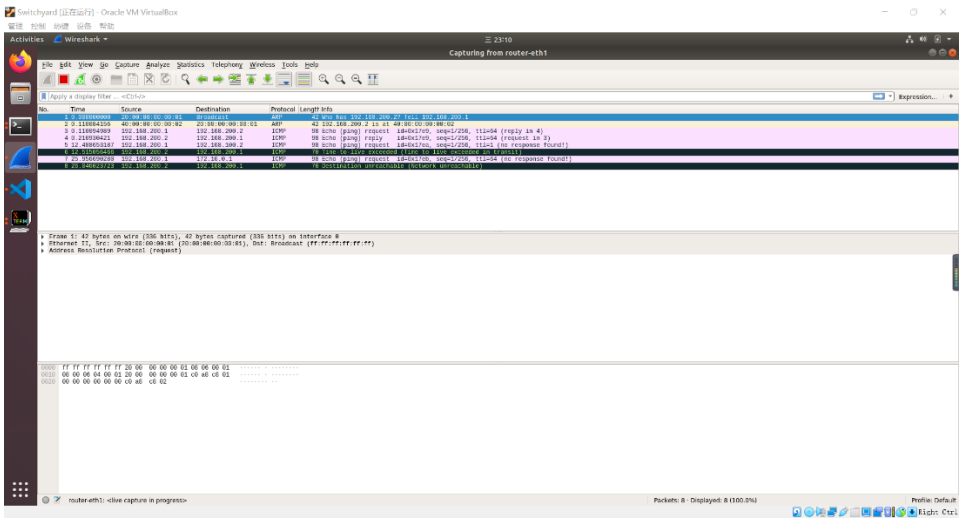
2、尝试测试 ttl 降为 0 的情况，在 xterm server2 中 ping -c1 -t1 192.168.100.2





3、测试一个不在转发表中的地址，为实现这一功能，首先在 start_mininet.py 中增加 set_route(net, 'server2', '172.16.0.0/24', '192.168.200.2')，再在 xterm server2 中 ping -c1 172.16.0.1，结果如下所示：





4、运行 tracetroute，输入 server2 traceroute 192.168.100.1