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：lzh、lsp、wcx

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	211220027	姓名	王秋博
Email	211220027@nju.edu.cn	开始/完成日期	10.24/11.9

1. 实验名称

Lab4: IP Forwarding Table Lookup

2. 实验目的

实现响应分配给路由器上接口的地址的 ARP（地址解析协议）请求。

3. 实验内容

Task 2：IP Forwarding Table Lookup

Step 1: Coding

实现转发表的数据结构如图：

```
class ForwardTable(object):
    def __init__(self,interfaces):
        self.data=[]
        # debugger()
        for intf in interfaces:
            # debugger()
            key=IPv4Network(str(IPv4Address(int(intf.ipaddr)&int(intf.netmask)))+'/'+str(intf.
            self.data.append([key,IPv4Address('0.0.0.0'),intf.name])
        with open('forwarding_table.txt','r') as fp:
            for line in fp.readlines():
                line=line.strip()
                line=line.split(' ')
                key=IPv4Network(line[0]+'/'+line[1])
                self.data.append([key,IPv4Address(line[2]),line[3]])
        self.data.sort(key=lambda key:key[0].prefixlen,reverse=True)

    def Query(self,address:IPv4Address):
        # debugger()
        for key in self.data:
            if address in key[0]:
                return key[1:]
        return None
```

将转发表存于列表中并根据从上 ipv4network 的 prefixlen 降序排序，则查找时从前往后查找即可查到最大匹配。具体匹配如下

```
packet[ipv4_idx].ttl-=1
# debugger()
info=self.forwardtable.Query(ipv4.dst)
# debugger()
if info==None: pass
else:
    next_ip=ipv4.dst if info[0]==IPv4Address('0.0.0.0') else info[0]
    next_intf=self.net.interface_by_name(info[1])
    next_mac=self.arptable.get(next_ip)
    # debugger()
```

直接查找，返回值不为 None 则表示匹配

Task 3：Forwarding the Packet and ARP

Step 1: Coding

在匹配之后，判断下一跳地址是否存在于 arp 缓存表中，在则直接点对点传输，不在就加入等待队列

```
else:
    next_ip=ipv4.dst if info[0]==IPv4Address('0.0.0.0') else info[0]
    next_intf=self.net.interface_by_name(info[1])
    next_mac=self.arptable.get(next_ip)
    # debugger()
    if(next_mac!=None):
        # debugger()
        ether_idx=packet.get_header_index(Ethernet)
        packet[ether_idx].src=next_intf.ethaddr
        packet[ether_idx].dst=next_mac
        self.net.send_packet(next_intf,packet)
        # debugger()
    else:
        self.arpqueue.put(WaitArpPacket(self.net,packet,next_intf,next_ip))
```

而等待队列数据结构如下，新增了时间和发送次数，并根据手册进行删除

而 handle_reply 一旦收到回复证明数据通路，就把寄存在等待队列的数据包发出

```
class WaitPacket(object):
    def __init__(self,net,packet,intf,next_ip):
        self.net=net
        self.packet=packet
        self.intf=intf
        self.next_ip=next_ip
        self.timestamp=0
        self.count=0

    def send_arp_request(self):
        ether=Ethernet()
        # debugger()
        ether.src=self.intf.ethaddr
        ether.dst='ff:ff:ff:ff:ff:ff'
        ether.ethertype = EtherType.ARP
        arp = Arp(operation=ArpOperation.Request,
                  senderhwaddr=self.intf.ethaddr,
                  senderprotoaddr=self.intf.ipaddr,
                  targethwaddr='ff:ff:ff:ff:ff:ff',
                  targetprotoaddr=self.next_ip)
        arppacket = ether + arp
        self.net.send_packet(self.intf,arppacket)
        self.timestamp=time.time()
        self.count+=1
        return self.count
```

```
class ArpQueue(object):
    def __init__(self,net):
        self.data=[]
        self.sendedarp=[]
        self.sendedip=[]
    #   self.forbidip=[]
        self.net=net

    def put(self,waitpkt):
        self.data.append(waitpkt)

    def handle_reply(self,reply):
        # debugger()
        dstip=reply.senderprotoaddr
        dstmac=reply.senderhwaddr
        if dstip not in self.sendedip: return
        self.sendedip.remove(dstip)
        for waitpkt in self.data[:]:
            if waitpkt.next_ip==dstip:
                ether_idx=waitpkt.packet.get_header_index(Ethernet)
                waitpkt.packet[ether_idx].src=waitpkt.intf.ethaddr
                waitpkt.packet[ether_idx].dst=dstmac
                self.net.send_packet(waitpkt.intf,waitpkt.packet)
                if waitpkt in self.sendedarp:self.sendedarp.remove(waitpkt)
                self.data.remove(waitpkt)
```

Timeout 函数进行循环判断是否同一 ip 查询超过 5 次，通过设立一个列表 sendedip 储存已经发过的 ip，防止多个数据包一起向同一 ip 地址发送请求包

```
    def clearpkts(self,ip):
        for waitpkt in self.data[:]:
            if waitpkt.next_ip==ip:
                self.data.remove(waitpkt)
        #self.forbidip.append(ip)

    def timeout(self):
        for waitpkt in self.data:
            if(time.time()-waitpkt.timestamp>1):
                # debugger()
                # waitpkt.send_arp_request()
                if waitpkt in self.sendedarp:
                    if waitpkt.count>=5:
                        self.clearpkts(waitpkt.next_ip)
                        self.sendedarp.remove(waitpkt)
                        self.sendedip.remove(waitpkt.next_ip)
                    else:
                        waitpkt.send_arp_request()
                else:
                    if waitpkt.next_ip  in self.sendedip:
                        pass
                    else:
                        self.sendedarp.append(waitpkt)
                        self.sendedip.append(waitpkt.next_ip)
                        tries=waitpkt.send_arp_request()
                        assert(tries==1)
```

Step 2: Testing

De 了几天 bug，终于过了，但多线程还是没能实现

```
Results for test scenario IP forwarding and ARP requester tests: 31 passed, 0 fa
iled, 0 pending


Passed:
1   IP packet to be forwarded to 172.16.42.2 should arrive on
    router-eth0
```

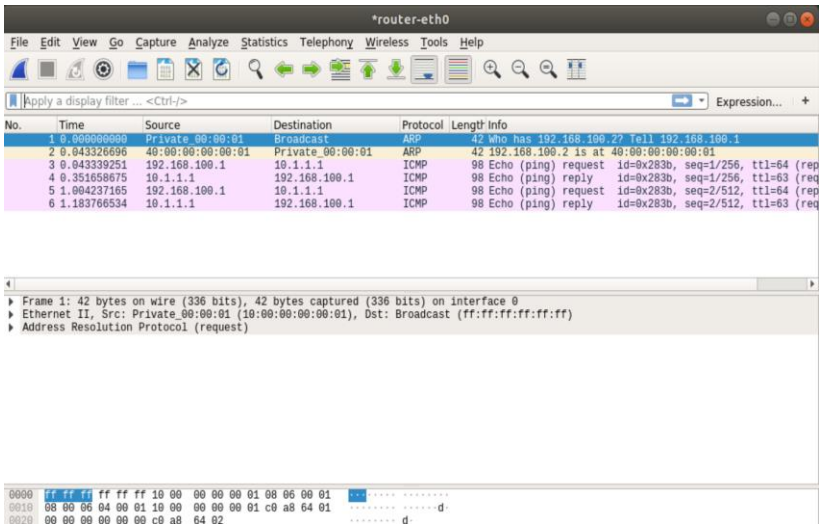
```
1193Ping request from 31.0.5.1 should arrive on eth5
1194Ping request from 31.0.5.1 should arrive on eth5
1195Ping request from 31.0.5.1 should arrive on eth5
1196Ping request from 31.0.5.1 should arrive on eth5
1197Ping request from 31.0.5.1 should arrive on eth5
1198Router should not do anything
1199Ping request from 31.0.6.1 should arrive on eth6
1200Ping request from 31.0.6.1 should arrive on eth6
1201Ping request from 31.0.6.1 should arrive on eth6
1202Ping request from 31.0.6.1 should arrive on eth6
1203Ping request from 31.0.6.1 should arrive on eth6
1204Ping request from 31.0.6.1 should arrive on eth6
1205Router should not do anything
1206Bonus: V2FybSB1cA==
1207Bonus: Q29vbCBkb3du
1208Bonus: V2h1dCBkJyB5YSBob3BlIHQnIGZpbmQgaGVyZT8=


Failed:
    Bonus: Tm90aGluJyBmb3IgeWEgdCcgZmluZCBoZXJlIQ==
        Expected event: recv_packet Ethernet
```

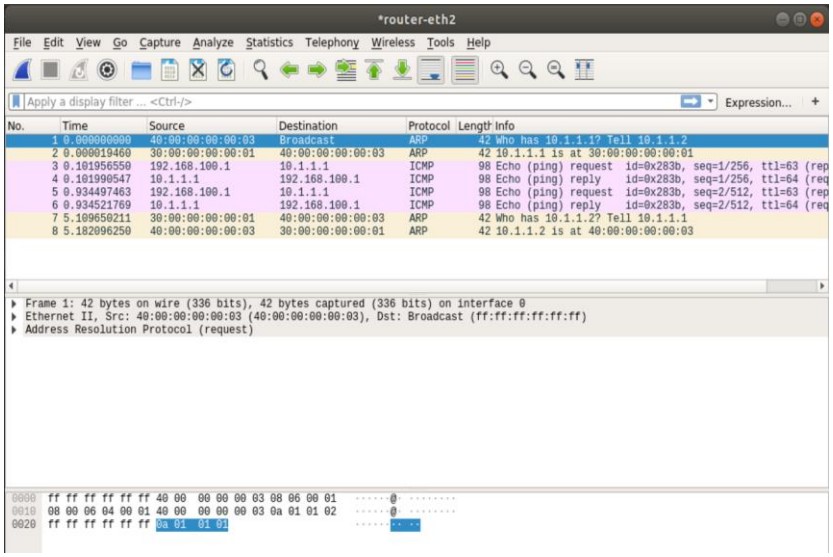
Step 2: Deploying

可以用 server1 ping -c2 client

分别监听 router 与这两端相连的 eth0 和 eth1 接口

结果如下：





如图所示，可以看到，server1 先广播发送 arp 包询问了 router-eth0 的 mac 地址，router 发送回复包后，server1 给 router 发送 icmp 包，router 根据转发表查询向 client 发送 arp 请求包查询 mac 地址，收到回复包后传输数据包，于是有了 ping 两次回显，第二次发包时 arp 缓存表里已经记载双方地址，故直接发送无需事先发送 arp 请求。

4. 实验结果

本节实验结果基本于实验过程中阐述，不再赘述

5. 核心代码

同实验结果

6. 总结与感想

//本次实验破大防，报告还将更新。
实验很破防，de 了很久边界条件，但也确实对于转发机制理解更深了

在此感谢王宸旭助教，真的很用心，鼓舞了我很多。