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	**2022.04.18**

实验名称

Lab4

实验目的

完善自己写的路由器功能

实验内容

完成路由器的核心功能：路由和转发，其中包括接收和转发数据包，生成以太网头部，对于未知MAC的IP地址，生成ARP Request

实验结果

Task 2

首先，为了实现在等待对ARP请求的回复时，可以接收和处理传入的数据包，所以要用一个队列实现其中的包缓存。

```
class Info():
    def __init__(self,prefix,mask,nexthop,name):
        self.prefix = prefix
        self.mask = mask
        self.nexthop = nexthop
        self.name = name

class Node():
    def __init__(self,packet,info):
        self.packet = packet
        self.info = info
        self.cnt = 0
        self.time = 0
```

用Info类实现了对forwardTable类的储存，用Node类表示在队列中缓存包的的封装，其中包括包和包的forwardTable以及用于时间和次数处理的cnt和time。

首先要构建IP转发表，我们需要在两个地方构建IP转发表，一种可以通过调用`net.interfaces()`获得路由器接口列表，另一种方式可以是通过读取`forwarding_table.txt`文件的内容。

```
for intf in net.interfaces():
    # 求网络前缀
    prefix = IPv4Network(str(intf.ipaddr) + "/" +
str(intf.netmask),strict=False)
    # 在这里加了strict = False则强制将主机位置零
    node_info = Info(prefix.network_address,intf.netmask,None,intf.name)
    self.forwardTable.append(node_info)

file = open("forwarding_table.txt")
while 1:
    line = file.readline()
    if not line:
        break
    else:
        line = line.strip('\n').split(" ")
        node_info =
Info(IPv4Address(line[0]),IPv4Address(line[1]),IPv4Address(line[2]),line[3])
        self.forwardTable.append(node_info)
        # 要注意类型转换，不要再str和IPv4Address中搞混
```

匹配目标转发表则是在处理一个收到的包时，经路由器接收到的IP数据包中的目标地址应该和转发表进行匹配，如果表中没有匹配项时，则只需丢弃数据包即可，如果数据包是针对路由器本身的，即目标地址时路由器的某个接口地址，则只需丢弃数据包即可。在表中如果有两个项目相匹配时，则应该使用最长的前缀进行匹配，在这里我们用到了`netaddr.prefixlen`函数，则可以进行比较。

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

    head.ttl -= 1
    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
    if best == -1:
        print("There is no match!")
    else:
        queue.append(Node(packet,self.forwardTable[best]))
```

这里的最佳索引，要赋初值`-1`不然会出现错误，而当`best == -1`时，则要单独讨论这个情况。

Task 3

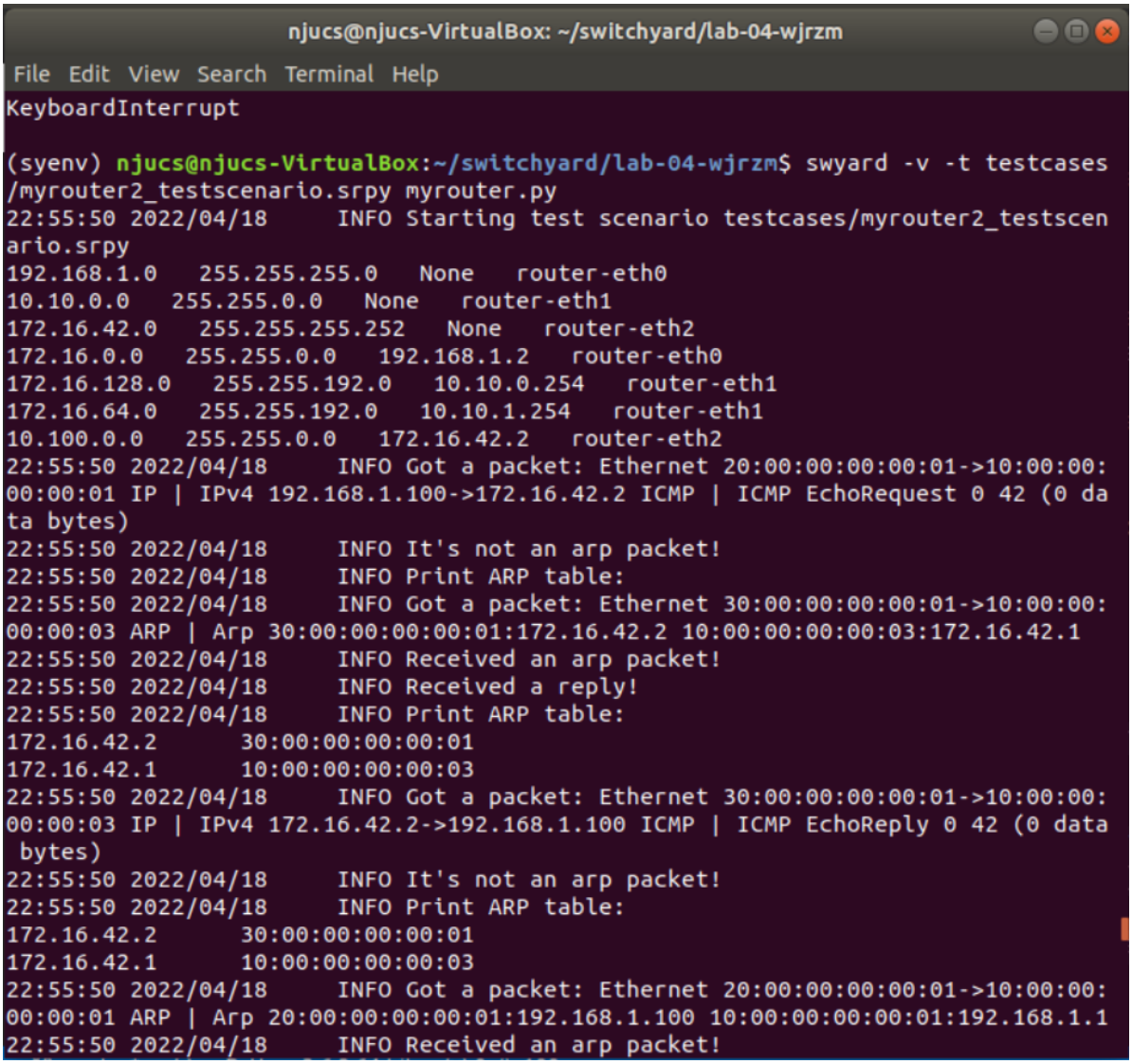
在这个任务中，首先在转发表中查找IP目标地址后，先要将IP表头的TTL项减一，在此项目中我们考虑在ttl递减后仍大于零，其次要为转发的IP数据包创建新的以太网包头，如果下一跳在路由器arp cache表中存在MAC地址，则可以直接发送，如果不知道MAC地址，则需要发送ARP进行查找。

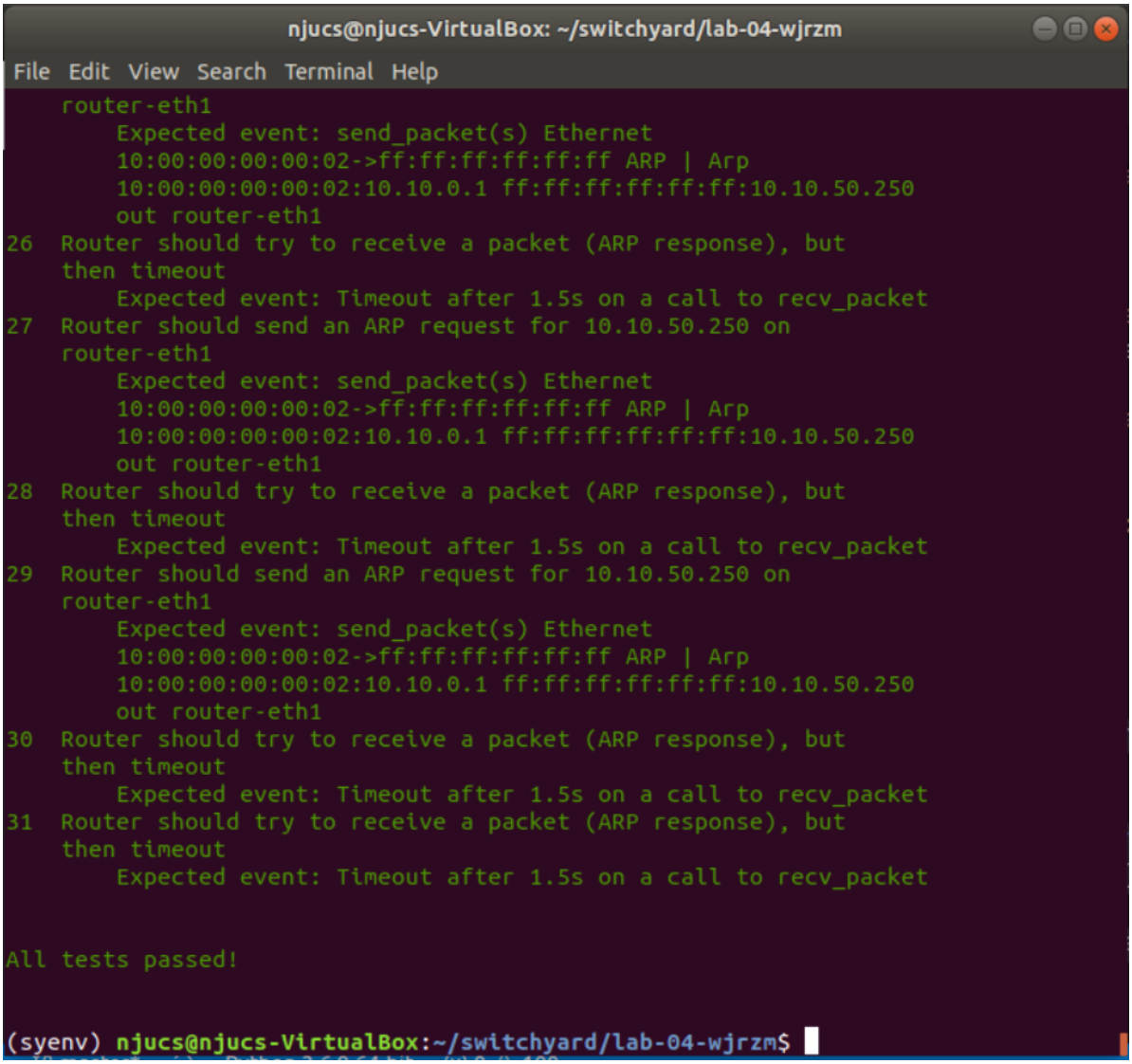
在发送ARP Request的时候，有需要有着对应的策略，当向对应端口发送ARP请求时，如果收到回复则生成包头，转发Packet，缓存MAC地址，如果没有回复则重复发送五次，在五次中如果等待时间超过一秒则直接进行下一次发送。

```
def handle_queue(self,queue):
    if len(queue) != 0:
        for intf in self.interfaces:
            if intf.name == queue[0].info.name:
                port = intf

        if queue[0].info.nexthop is None:
            targetIp = queue[0].packet[IPv4].dst
        else:
            targetIp = queue[0].info.nexthop

        flag = 0

        for (k,v) in self.arpTable.items():
            if k == targetIp:
                queue[0].packet[Ethernet].dst = v
                queue[0].packet[Ethernet].src = port.ethaddr
                self.net.send_packet(port,queue[0].packet)
                del(queue[0])
                flag = 1
                # 判断是否在ARP表中的标志
                break

        if flag == 0:
            if queue[0].cnt >= 5:
                del(queue[0])
            else:
                if (queue[0].cnt == 0) or (time.time() - queue[0].time) > 1:
                    # 根据API生成以太网包头
                    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)
                    queue[0].cnt += 1
                    queue[0].time = time.time()
```

而关于arp处理的代码，是照搬实验三中的，此处不重复叙述。

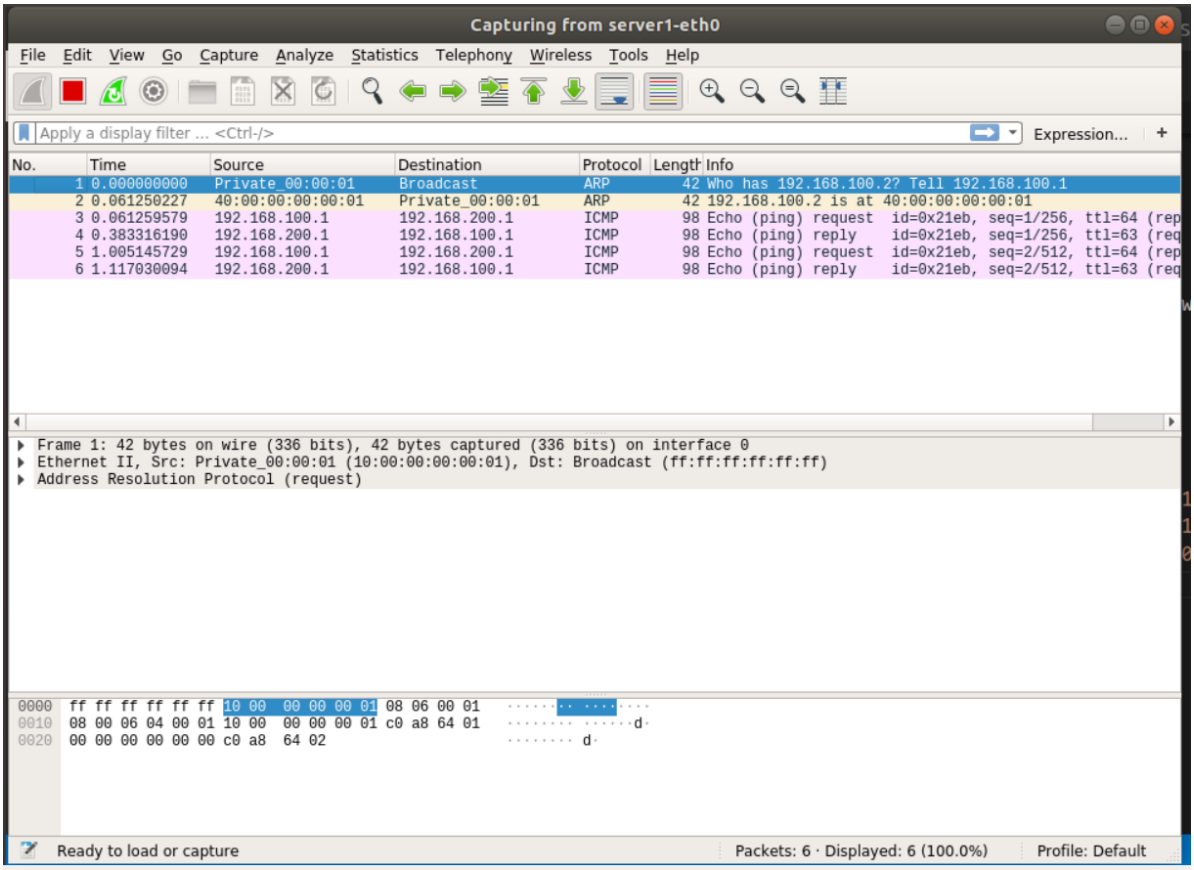
测试结果如下：


njucs@njucs-VirtualBox: ~/switchyard/lab-04-wjrzm
File Edit View Search Terminal Help
KeyboardInterrupt
(syenv) njucs@njucs-VirtualBox:~/switchyard/lab-04-wjrzm$ swyard -v -t testcases
/myrouter2_testscenario.srpy myrouter.py
22:55:50 2022/04/18 INFO Starting test scenario testcases/myrouter2_testscen
ario.srpy
192.168.1.0 255.255.255.0 None router-eth0
10.10.0.0 255.255.0.0 None router-eth1
172.16.42.0 255.255.255.252 None router-eth2
172.16.0.0 255.255.0.0 192.168.1.2 router-eth0
172.16.128.0 255.255.192.0 10.10.0.254 router-eth1
172.16.64.0 255.255.192.0 10.10.1.254 router-eth1
10.100.0.0 255.255.0.0 172.16.42.2 router-eth2
22:55:50 2022/04/18 INFO Got a packet: Ethernet 20:00:00:00:00:01->10:00:00:
00:00:01 IP | IPv4 192.168.1.100->172.16.42.2 ICMP | ICMP EchoRequest 0 42 (0 da
ta bytes)
22:55:50 2022/04/18 INFO It's not an arp packet!
22:55:50 2022/04/18 INFO Print ARP table:
22:55:50 2022/04/18 INFO Got a packet: Ethernet 30:00:00:00:00:01->10:00:00:
00:00:03 ARP | Arp 30:00:00:00:00:01:172.16.42.2 10:00:00:00:00:03:172.16.42.1
22:55:50 2022/04/18 INFO Received an arp packet!
22:55:50 2022/04/18 INFO Received a reply!
22:55:50 2022/04/18 INFO Print ARP table:
172.16.42.2 30:00:00:00:00:01
172.16.42.1 10:00:00:00:00:03
22:55:50 2022/04/18 INFO Got a packet: Ethernet 30:00:00:00:00:01->10:00:00:
00:00:03 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP EchoReply 0 42 (0 data
bytes)
22:55:50 2022/04/18 INFO It's not an arp packet!
22:55:50 2022/04/18 INFO Print ARP table:
172.16.42.2 30:00:00:00:00:01
172.16.42.1 10:00:00:00:00:03
22:55:50 2022/04/18 INFO Got a packet: Ethernet 20:00:00:00:00:01->10:00:00:
00:00:01 ARP | Arp 20:00:00:00:00:01:192.168.1.100 10:00:00:00:00:01:192.168.1.1
22:55:50 2022/04/18 INFO Received an arp packet!



部署至Mininet

在server1端口ping server2，在server1处抓包，在server1的端口发出了arp request，路由器进行回复，server1得到了和server2相连的mac，随后server1发出了ICMP包，路由器处理时，发现本地的arptable中没有于是，从eth1向server2发送了arp request，随后收到了server2 的reply，这时候队列可以进行处理，发出了ICMP的包，路由器收到后进行处理，同理又进行了一次往返处理。



总结

本次实验主要是需要查询一些新的API用法，其余则需要理清实验逻辑即可。