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　　助教：

学院	**计算机科学与技术系**	专业（方向）	**计算机科学与技术**
学号	**221220143**	姓名	**陈雨檬**
Email	**221220143@qq.com**	开始/完成日期	**2024.4.14/2024.4.21**

1. 实验名称

Lab4: 转发数据包

2. 实验目的

实现接收和转发到达链路并发往其他主机的数据包；对未知以太网 mac 地址的 ip 地址发出 arp 请求。

3. 实验内容

Task1：Preparation

Task2：IP 转发表查找

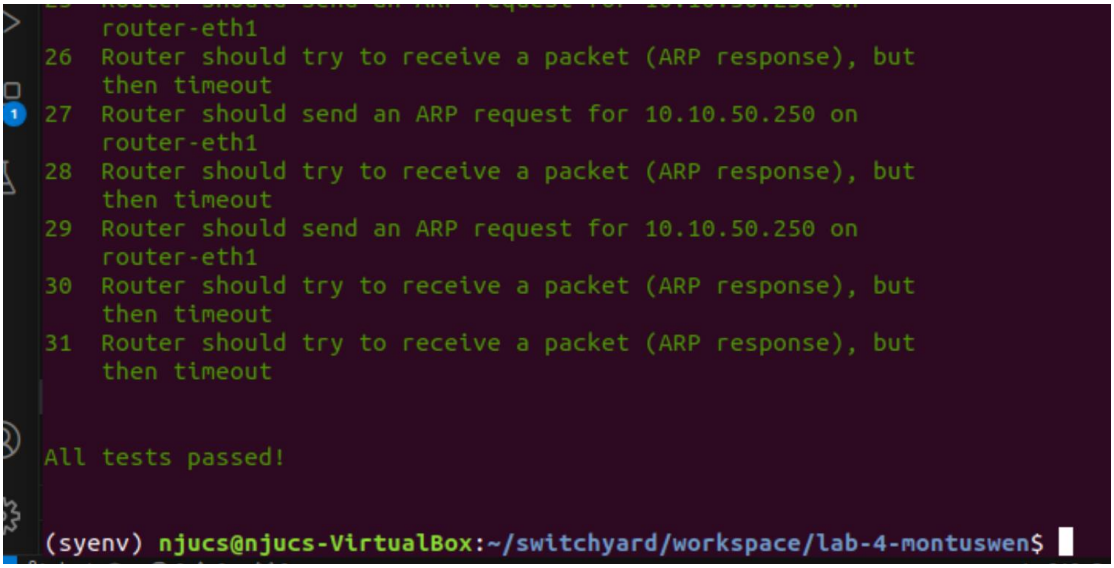
How: 自定义一个类 forward_table 作为 ip 转发表，类中定义一个列表存储所有的表项，每一个表项都是一个自定义自定类 forward_table_item。每一个表项中都存储 IP 和相应的掩码 mask 和下一跳的 IP 与发送的端口。每有一个新的 IP 需要查找转发表时，采用最长前缀匹配找到相应的表项。

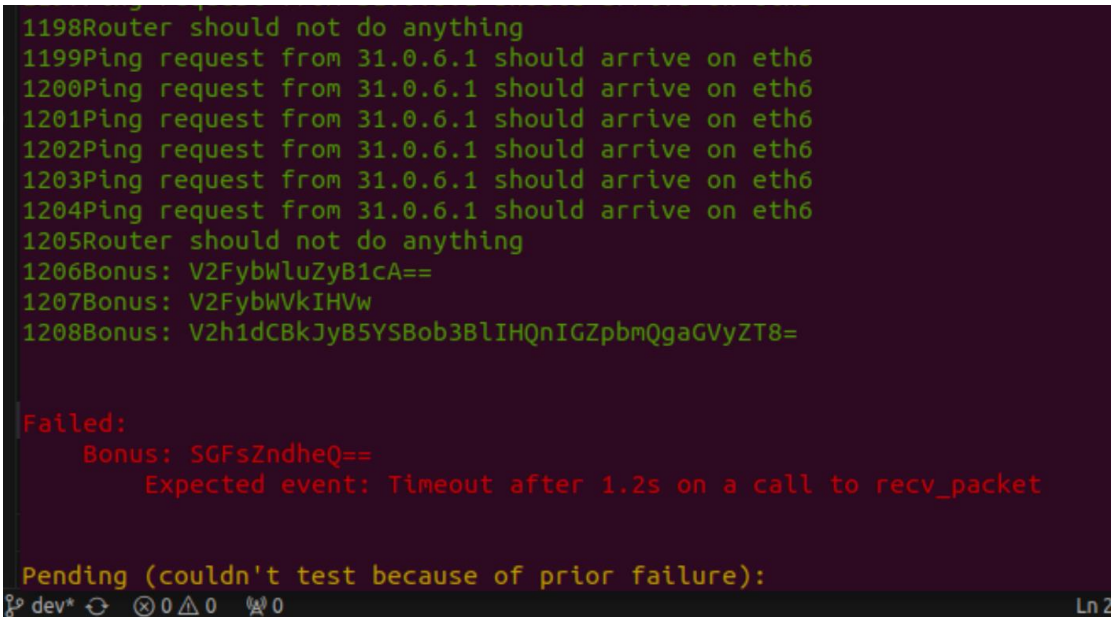
Task3：转发数据包和发送 ARP 请求

How: 如果已知要转发的数据包的目的 IP 对应的 MAC 地址，则填写数据包以太网报头的内容并从相应的端口发出；如果不知道数据包的目的 MAC 地址，则需要构造一个 arp 包并发出 request 请求得到目的 MAC 地址，构造一个队列来存储已经发送的 ARP 包，如果已经收到回答则清除该 arp 包对应所有需要发送到该地址出的 packet，否则如果超过 1s 都没有收到回答再次重发 ARP 包，重发如果超过 5 次则丢弃。类似的，用自定义类 queue 和 queue_item 来存储 ARP 队列和队列中的每一项。

4. 实验结果

首先进行样例测试



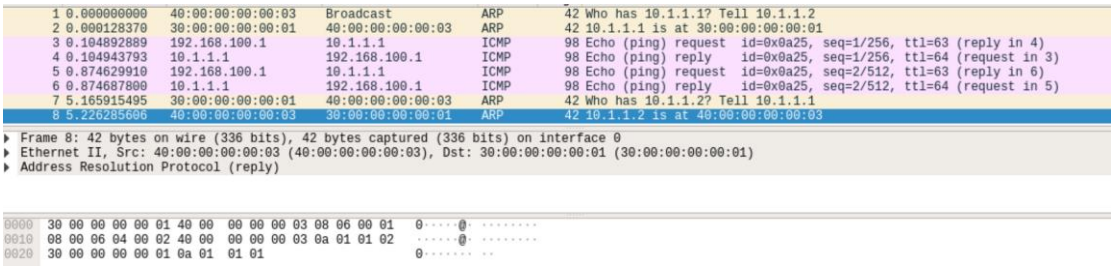


除了 bonus 多线程测试无法通过，其余正确通过

再进行部署测试

Wireshark 监视 router 的 eth2 接口，server1 向 client 发出 2 个 ping。

Wireshark 抓包结果如下：



Server1 先发出 arp 包询问 client 的 mac 地址，client 的 arp 回复先到 router 再到 sever1。Router 在这个过程中缓存了 server1 和 client 的 ip、mac 地址对应关系，因此 router 在接收到 server1 发出的数据包后直接转发给了 client。

5. 核心代码

Task2 核心代码

```
class forward_table_item:
    def __init__(self,ip,mask,next_des,iface):
        ipnum=IPv4Address(ip)
        masknum=IPv4Address(mask)
        self.ip=IPv4Address(int(ipnum)&int(masknum))
        self.mask=IPv4Address(masknum)
        if next_des is not None:
            self.next_des=IPv4Address(next_des)
        else:
            self.next_des=None
        self.iface=iface
        self.prefixnet=IPv4Network(str(self.ip)+'/'+mask)
    def prefixlen(self):
        return self.prefixnet.prefixlen
    def match(self,desaddr):
        return desaddr in self.prefixnet
    def printf(self):
        log_info("ip:{} mask:{} nextdes:{} iface:{}".format(sel
```

```
class forward_table:
    def __init__(self,interfaces):
        self.table=[]
        f=open('forwarding_table.txt','r')
        for lines in f:
            ip,mask,desaddr,iface=lines.strip().split(' ')
            self.table.append(forward_table_item(ip,mask,desaddr,iface))
        f.close()
        for ifa in interfaces:
            self.table.append(forward_table_item(str(ifa.ipaddr),str(ifa.n
    def find_match(self,desipaddr):
        maxlen=-1
        res=None
        for it in self.table:
            if(it.match(desipaddr) and it.prefixlen()>maxlen):
                res=it
                maxlen=it.prefixlen()
        return res
```

如图，在 IP 转发表表项所属的类 forward_table_item 中，prefixlen 函数返回待匹配的 IP 地址与匹配的表项的最大前缀长度，match 函数则用于判断待匹配的 IP 地址是否与某个表项匹配。在 IP 转发表所属的类中 find_match 函数则用于找到匹配的前缀长度最大的表项并返回。

Task3

```
class queue_item:
    def __init__(self,arp,port):
        self.arp=arp
        self.pkt=[]
        self.time=time.time()
        self.times=1
        self.port=port
    def add_packet(self,pkt):
        self.pkt.append(pkt)
    def matches(self,senderaddr):
        return self.arp.get_header(Arp).targetprotoaddr==senderaddr
```

```
class queue:
    def __init__(self,net):
        self.arp_q=[]
        self.net=net
    def add_item(self,arp,pkt,port):
        for item in self.arp_q:
            if item.arp.get_header(Arp).targetprotoaddr==arp.get_header(Ar
                item.add_packet(pkt)
                return True
        log_info("!!!")
        newitem=queue_item(arp,port)
        newitem.add_packet(pkt)
        self.arp_q.append(newitem)
        self.net.send_packet(port,arp)
        return False
```

```
    def getreply(self,arp,interface):
        for it in self.arp_q:
            if(it.matches(arp.senderprotoaddr)):
                for pkt in it.pkt:
                    pkt[0].dst=arp.senderhwaddr
                    pkt[0].src=interface.ethaddr
                    self.net.send_packet(interface,pkt)
                self.arp_q.remove(it)
                return True
    def resend(self):
        now=time.time()
        for it in self.arp_q:
            if it.times>=5:
                self.arp_q.remove(it)
            elif now-it.time>1.0:
                log_info("Fa Le")
                log_info(it.arp.get_header(Arp).targetprotoaddr)
                self.net.send_packet(it.port,it.arp)
                it.time=time.time()
                it.times+=1
```

当 router 收到一个 ARP 回复时，调用 getreply 函数，找到对应的队列中的表项，填写相应全部数据报的内容并发出；resend 函数则用于重发队列中还没有得到回答的 ARP 包。

```
if port!=None:
    if arp.operation==1:
        self.record.add_item(arp.senderprotoaddr,arp.senderhwaddr)
        forward_item=self.forward_record.find_match(arp.senderprotoa
        if forward_item is not None:
            replypkt=create_ip_arp_reply(port.ethaddr,arp.senderhwad
            self.net.send_packet(interface,replypkt)
        else:
            pass
    else:
        if arp.senderhwaddr!='ff:ff:ff:ff:ff:ff':
            self.record.add_item(arp.senderprotoaddr,arp.senderhwadd
            self.arpqueue.getreply(arp,interface)
```

如果收到给 router 自己的回答 ARP 包则调用队列的 getreply 函数；但如果发送方不在转发表提供的网络中则忽略该包。

```
if forward_item.next_ip is None:
    nextip=ipv4.dst
else:
    nextip=forward_item.next_ip
# self.record.printf()
port=self.net.interface_by_name(forward_item.portname)
nextmac=self.record.get_item(nextip)
```

```
if nextmac is not None:
    packet[0].dst=nextmac
    packet[0].src=port.ethaddr
    self.net.send_packet(port,packet)
    log_info("???")
else:
    log_info(ipv4.dst)
    log_info(forward_item.portname)
    log_info(nextip)
    flag=True
    for item in self.interfaces:
        if item.ipaddr==nextip:
            flag=False
    if flag:
        arp_request = create_ip_arp_request(port.ethaddr,port.ipaddr,r
        self.arpqueue.add_item(arp_request,packet,port)
    else:
        pass
```

如果在自己的缓存 ARP 表能找到目的 IP 对应的 MAC 则直接填写数据报报头内容再发送即可，否则则由 router 发送 arp 请求询问。但如果目标就是 router 本身则不做转发。

6. 思考与感受

本次实验比较复杂，要考虑多种特殊情况，应当小心处理便于后续的构建工作。