南京大学本科生实验报告

181860097 计算机科学与技术系 王明骁
Email：1164151483@qq.com
开始日期：2020年4月18日 结束日期：2020年4月20日
lab4

实验名称

Lab 4: Forwarding Packets

实验目的

- 为构建一个完整的路由器编写其转发数据包部分
- 理解路由器的转发工作原理
- 学会处理路由器在转发数据包时遇到的各种问题

task1

1. **实验内容**

   task1为准备阶段，只需要按照实验手册上的说明，将相关文件copy到lab_4文件夹并做好命名即可

task2

1. 实验内容

   task2要求实现转发中的查找部分，该部分分为以下两个小部分

   - 建立转发表：

     转发表的来源有两个，一个是从文件forwardingtable.txt中直接读取获取，另一个是从路由器的端口信息中读取与其直接相连的子网信息来获取

     ```
     for line in fp.readlines():      You, 17 hours ago • fix bug and pass 3 tests
         if len(line) > 10:
             item1,item2,item3,item4 = line.split()
             prefixnet = IPv4Network('{}/{}'.format(item1,item2))
             fwtemp = forwardingitem(item1,item2,item3,item4,prefixnet)
             #debugger()
             forwardingtable.append(fwtemp)

     fp.close()
     for intf in intftable:
         prefixnet = IPv4Network('{}/{}'.format(intf.ipaddr,intf.netmask),False)
         fwtemp = forwardingitem(intf.ipaddr,intf.netmask,None,intf.name,prefixnet)
         forwardingtable.append(fwtemp)
     ```

     而整个表采用一个list来获取，其基元素为forwardingitem类，类中除了包含实验说明中要求的网络前缀，子网掩码，下一跳地址，端口名字之外，为了方便后续的编程，还额外增加了一个IPv4Network对象

   - 将收到的ipv4的包与转发表中的内容进行匹配：

```
if ipv4 is not None:
    #debugger()
    pkt[IPv4].ttl -= 1
    pretime = time.time()
    #debugger()
    flag = 1
    maxlength = 0
    for index in forwardingtable:
        #debugger()
        matches = ipv4.dst in index.prefixnet
        if matches == True:
            if index.prefixnet.prefixlen > maxlength:
                maxlength = index.prefixnet.prefixlen
                dest = index
                flag = 0
```

采取最长前缀匹配的方式，按照实验说明中提供的方法进行匹配，如果匹配上则计算网络前缀长度，取其中最长的一个网络前缀为匹配成功的项，同时用flag标识目标ip是否和表中某一项匹配，如果目标ip不和表中任何一项匹配，则路由器直接丢弃这个包

2. 实验结果

由于路由器的转发数据包功能还未完全实现，因此无法测试上述代码是否正确，只有等待task3完成后一起测试

3. 核心代码

forwardingitem类的定义：

```
class forwardingitem(object):
    def __init__(self,netaddr,subnet_mask,nextip,intfname,prefixnet):
        self.netaddr = netaddr
        self.subnet_mask = subnet_mask
        self.nextip = nextip
        self.intfname = intfname
        self.prefixnet = prefixnet
```

建立转发表：

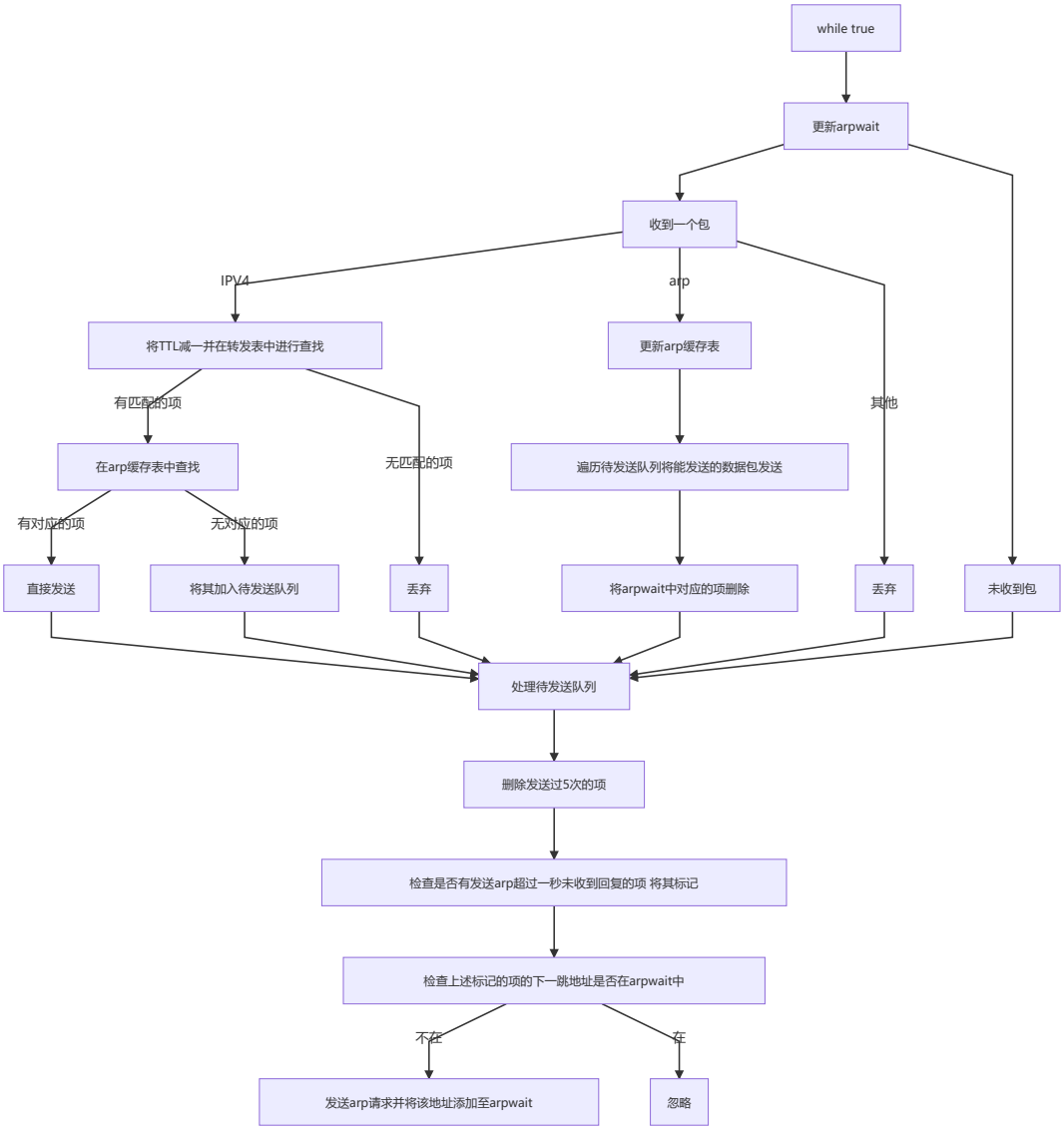
```
for line in fp.readlines():      You, 17 hours ago • fix bug and pass 3 tests
    if len(line) > 10:
        item1,item2,item3,item4 = line.split()
        prefixnet = IPv4Network('{}/{}'.format(item1,item2))
        fwtemp = forwardingitem(item1,item2,item3,item4,prefixnet)
        #debugger()
        forwardingtable.append(fwtemp)

fp.close()
for intf in intftable:
    prefixnet = IPv4Network('{}/{}'.format(intf.ipaddr,intf.netmask),False)
    fwtemp = forwardingitem(intf.ipaddr,intf.netmask,None,intf.name,prefixnet)
    forwardingtable.append(fwtemp)
```

将收到的数据包与转发表的每一项进行最长前缀匹配：

```
if ipv4 is not None:
    #debugger()
    pkt[IPv4].ttl -= 1
    pretime = time.time()
    #debugger()
    flag = 1
    maxlength = 0
    for index in forwardingtable:
        #debugger()
        matches = ipv4.dst in index.prefixnet
        if matches == True:
            if index.prefixnet.prefixlen > maxlength:
                maxlength = index.prefixnet.prefixlen
                dest = index
                flag = 0
```

task3

1. 实验内容：task3要求真正实现路由器的转发数据包功能，其具体实现逻辑为：



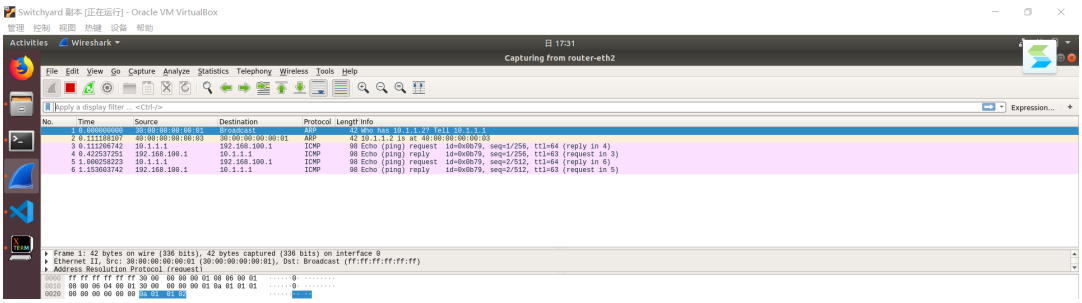
为了应对之前就已发送过arp请求还未得到回复，在这段时间内又收到发往同一个地址的数据包的情况，增加了arpwait队列用来标识还未得到回复的arp队列，如果一个数据包的目的地址在arp队列中，则说明已发送过arp_request并且还未过去一秒，此时将不再发送arp请求，同时，在每次收到数据包之前对arpwait进行更新，将超过一秒的项删除

2. 实验结果：

运行测试样例：可以看到所有测试均已通过

```
(syenv) njucs@njucs-VirtualBox:~/switchyard/lab_4$ swyard -t routertests2.srpy myrouter.py
16:55:03 2020/04/19     INFO Starting test scenario routertests2.srpy
16:55:03 2020/04/19     INFO ip: 172.16.42.2    mac:30:00:00:00:00:01

16:55:03 2020/04/19     INFO ip: 172.16.42.2    mac:30:00:00:00:00:01

16:55:03 2020/04/19     INFO ip: 192.168.1.100  mac:20:00:00:00:00:01

16:55:04 2020/04/19     INFO ip: 172.16.42.2    mac:30:00:00:00:00:01

16:55:04 2020/04/19     INFO ip: 192.168.1.100  mac:20:00:00:00:00:01

16:55:04 2020/04/19     INFO ip: 10.10.1.254    mac:11:22:33:44:55:66

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

Passed:
1   IP packet to be forwarded to 172.16.42.2 should arrive on
    router-eth0
2   Router should send ARP request for 172.16.42.2 out router-
    eth2 interface
3   Router should receive ARP response for 172.16.42.2 on
    router-eth2 interface
4   IP packet should be forwarded to 172.16.42.2 out router-eth2
5   IP packet to be forwarded to 192.168.1.100 should arrive on
    router-eth2
6   Router should send ARP request for 192.168.1.100 out router-
    eth0
7   Router should receive ARP response for 192.168.1.100 on
    router-eth0
8   IP packet should be forwarded to 192.168.1.100 out router-
    eth0
9   Another IP packet for 172.16.42.2 should arrive on router-
    eth0
10  IP packet should be forwarded to 172.16.42.2 out router-eth2
    (no ARP request should be necessary since the information
    from a recent ARP request should be cached)
11  IP packet to be forwarded to 192.168.1.100 should arrive on
    router-eth2
12  IP packet should be forwarded to 192.168.1.100 out router-
    eth0 (again, no ARP request should be necessary since the
    information from a recent ARP request should be cached)
13  An IP packet from 10.100.1.55 to 172.16.64.35 should arrive
    on router-eth1
14  Router should send an ARP request for 10.10.1.254 on router-
    eth1
15  Application should try to receive a packet, but then timeout
16  Router should send another an ARP request for 10.10.1.254 on
    router-eth1 because of a slow response
17  Router should receive an ARP response for 10.10.1.254 on
    router-eth1
18  IP packet destined to 172.16.64.35 should be forwarded on
```

在mininet上部署，并使用wireshark进行抓包，执行实验说明上提供的示例：



可以看到在路由器的端口2上正确抓取到一开始路由器发送的arp请求，得到的arp回复，转发的四个ICMP的包

由此可以说明该路由器的转发功能基本正确实现

3. 核心代码：

   待发送队列和arpwait中每一项的类定义：

```
class queueitem(object):
    def __init__(self,pkt,time,dstip,cnt,intfname):
        self.pkt = pkt
        self.time = time
        self.dstip = dstip
        self.cnt = cnt
        self.intfname = intfname


You, 15 hours ago | 1 author (You)
class arpwaititem(object):
    def __init__(self,ipaddr,time):
        self.ipaddr = ipaddr
        self.time = time
```

其中，待转发队列中每一项除了包含pkt之外还包含其上次发送arp请求的时间，目的地址，下一跳的ip，发送arp请求的次数和端口号；而arpwait中仅包含发送arp请求时请求的ip和时间

每次收到arp包时更新arp缓存表之后，遍历待发送队列中的每一项，将得到arp回复（或者得到对应ip的arp申请）的包发送，同时更新arpwait：

```
for queueindex in sendqueue:
    #debugger()
    if str(arp.senderprotoaddr) == str(queueindex.dstip):
        queueindex.pkt[0].dst = arp.senderhwaddr
        self.net.send_packet(dev, queueindex.pkt)
        sendqueue.remove(queueindex)

for index in arpwait:
    if str(index.ipaddr) == str(arp.senderprotoaddr):
        arpwait.remove(index)
```

收到ipv4数据包后，且数据包能与转发表的某一项匹配上时：

```
if flag == 0:
    port = self.net.interface_by_name(dest.intfname)
    if dest.nextip is None:
        dest.nextip = ipv4.dst
    pkt[Ethernet].src = port.ethaddr
    #debugger()

    flag = 1
    for searchindex in arptable:
        if searchindex.ip == dest.nextip:
            pkt[0].dst = searchindex.mac
            self.net.send_packet(dest.intfname,pkt)
            flag = 0
            break

    if flag == 1:
        #debugger()
        sendtemp = queueitem(pkt,pretime - 1,dest.nextip,1,dest.intfname)
        sendqueue.append(sendtemp)
```

首先在arptable中查找是否有对应的项，如果有则可以直接发送，否则将其加入到待发送队列之中，值得一提的是，这里将添加到队列中的时间初始化为收到包的时间 - 1，是为了可以在处理待发送队列时将刚接收到，还未发送过arp请求的数据包处理，让其可以发送一次arp请求，发送之后时间更新为发送arp请求的时间，即可与发送过arp请求的数据包一致

对待发送队列的处理：

```
if len(sendqueue) != 0 :
    nowtime = time.time()
    for index in sendqueue:
        if index.cnt > 5:
            sendqueue.remove(index)
            continue

        #debugger()

        if nowtime - index.time > 1:
            flag = 1
            for index in arpwait:
                if index.ipaddr == dest.nextip:
                    flag = 0
                    break
            if flag == 1:
            #if index.dstip not in arpwait:
                self.port = self.net.interface_by_name(index.intfname)
                arppkt = create_ip_arp_request(port.ethaddr,port.ipaddr,index.dstip)
                #debugger()
                self.net.send_packet(index.intfname,arppkt)
                index.cnt = index.cnt + 1
                temp = arpwaititem(index.dstip,nowtime)
                arpwait.append(temp)
        index.time = nowtime
```

首先删除发送次数超过5次的包，然后检查距离上次发送arp请求是否超过一秒，如果没有超过则不做处理，否则检查其是否在arpwait队列中，如果在则说明之前刚发送过一个arp请求，这次不应该再发送，否则发送一个arp请求并将其添加到arpwait中

每次对arpwait的更新：

```
nowtime = time.time()
for index in arpwait:
    if nowtime - index.time > 1:
        arpwait.remove(index)
```

在每次判断是否收到包之前，对arpwait进行更新，将超时的表项删除

总结与感想

lab4相比之前的lab无论是复杂程度还是难度都提升了不少，因此对路由器转发工作原理的理解显得尤为重要，一开始我对转发过程的理解不是很透彻，因此犯了不少原理上的错误，之后在进行实验时应该更注重对理论部分的理解