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4日 结束日期：2020年4月6日
lab3

实验名称

Lab 3: Respond to ARP

实验目的

- 学习和掌握路由器的基本工作原理
- 掌握怎样在switchyard中构建一个arp包
- 掌握arp包的收发过程，及各种api的使用

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_3文件夹并做好命名即可

task2

1. 实验内容

   task2要求实现一个可以回复arp包的路由器，针对arp_requset做出回复，其具体做法为：在路由器自己的端口上查找是否分配过这个ip给对应的端口，如果分配过，则根据对应的mac地址和ip构造arp包，并通过发送来arp_request的端口发出去，达到arp回复的功能

```
if gotpkt:
    log_debug("Got a packet: {}".format(str(pkt)))
    arp = pkt.get_header(Arp)
    if arp is not None:
        intftable = self.net.interfaces()
        for index in intftable:
            if index.ipaddr == arp.targetprotoaddr:
                packet = create_ip_arp_reply(index.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprot
                # packet = create_ip_arp_reply(arp.senderhwaddr,index.ethaddr,arp.senderprotoaddr,arp.targetpr
                self.net.send_packet(dev,packet)

        flag = 0
        for index in arptable:
            if index.ip == arp.senderprotoaddr and index.mac != arp.senderhwaddr:
                arptable.remove(index) #item is changed      You, a few seconds ago • Uncommitted changes
            elif index.ip == arp.senderprotoaddr and index.mac == arp.senderhwaddr:
                flag = 1
                break
        if flag == 0:
            temp = arpitem(arp.senderprotoaddr,arp.senderhwaddr)
            arptable.append(temp)
        for index2 in arptable:
            log_info("ip: {}\tmac:{}\n".format(str(index2.ip),str(index2.mac)))
```

   具体实现为：首先判断该包是不是arp包，如果不是，在现阶段不做处理，如果是arp包，则到端口里查找是否分配过对应的ip，如果没有则不做处理，否则按照分配过ip的端口对应的mac地址构造arp包并进行回复即可

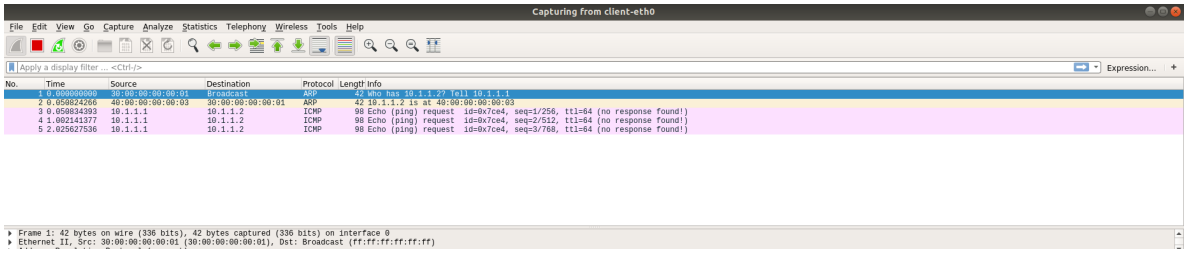
2. 实验结果

首先是运行助教提供的两个测试样例，可以看到均已通过测试

```
(syenv) njucs@njucs-VirtualBox:~/switchyard$ swyard -t lab_3/routertests1.srpy lab_3/myrouter.py
09:06:15 2020/04/06     INFO Starting test scenario lab_3/routertests1.srpy
09:06:15 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:15 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:15 2020/04/06     INFO ip: 10.10.1.1      mac:60:00:de:ad:be:ef

09:06:15 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:15 2020/04/06     INFO ip: 10.10.1.1      mac:60:00:de:ad:be:ef

09:06:15 2020/04/06     INFO ip: 10.10.5.5      mac:70:00:ca:fe:c0:de


Results for test scenario ARP request: 6 passed, 0 failed, 0 pending


Passed:
1   ARP request for 192.168.1.1 should arrive on router-eth0
2   Router should send ARP response for 192.168.1.1 on router-
    eth0
3   An ICMP echo request for 10.10.12.34 should arrive on
    router-eth0, but it should be dropped (router should only
    handle ARP requests at this point)
4   ARP request for 10.10.1.2 should arrive on router-eth1, but
    the router should not respond.
5   ARP request for 10.10.0.1 should arrive on on router-eth1
6   Router should send ARP response for 10.10.0.1 on router-eth1


All tests passed!
```

```
(syenv) njucs@njucs-VirtualBox:~/switchyard$ swyard -t lab_3/routertests1full.srpy lab_3/myrouter.py
09:06:51 2020/04/06     INFO Starting test scenario lab_3/routertests1full.srpy
09:06:51 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 172.16.42.2    mac:50:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 172.16.42.2    mac:50:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 10.10.1.1      mac:60:00:de:ad:be:ef

09:06:51 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 172.16.42.2    mac:50:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 10.10.1.1      mac:60:00:de:ad:be:ef

09:06:51 2020/04/06     INFO ip: 10.10.5.5      mac:70:00:ca:fe:c0:de

09:06:51 2020/04/06     INFO ip: 192.168.1.100  mac:30:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 172.16.42.2    mac:50:00:00:00:00:01

09:06:51 2020/04/06     INFO ip: 10.10.1.1      mac:60:00:de:ad:be:ef

09:06:51 2020/04/06     INFO ip: 10.10.5.5      mac:70:00:ca:fe:c0:de


Results for test scenario ARP request: 9 passed, 0 failed, 0 pending


Passed:
1   ARP request for 192.168.1.1 should arrive on router-eth0
2   Router should send ARP response for 192.168.1.1 on router-
    eth0
3   An ICMP echo request for 10.10.12.34 should arrive on
    router-eth0, but it should be dropped (router should only
    handle ARP requests at this point)
4   ARP request for 172.16.42.1 should arrive on router-eth2
5   Router should send ARP response for 172.16.42.1 on router-
    eth2
6   ARP request for 10.10.1.2 should arrive on router-eth1, but
    the router should not respond.
7   ARP request for 10.10.1.1 should arrive on on router-eth1
8   ARP request for 10.10.0.1 should arrive on on router-eth1
9   Router should send ARP response for 10.10.0.1 on router-eth1


All tests passed!
```

然后是在mininet中运行该网络拓扑，并用wireshark对client进行抓包，可以看到arp包被正确回复



3. 核心代码

```
if gotpkt:
    log_debug("Got a packet: {}".format(str(pkt)))
    arp = pkt.get_header(Arp)
    if arp is not None:
        intftable = self.net.interfaces()
        for index in intftable:
            if index.ipaddr == arp.targetprotoaddr:
                packet = create_ip_arp_reply(index.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprot
                # packet = create_ip_arp_reply(arp.senderhwaddr,index.ethaddr,arp.senderprotoaddr,arp.targetpr
                self.net.send_packet(dev,packet)

        flag = 0
        for index in arptable:
            if index.ip == arp.senderprotoaddr and index.mac != arp.senderhwaddr:
                arptable.remove(index) #item is changed      You, a few seconds ago • Uncommitted changes
            elif index.ip == arp.senderprotoaddr and index.mac == arp.senderhwaddr:
                flag = 1
                break
        if flag == 0:
            temp = arpitem(arp.senderprotoaddr,arp.senderhwaddr)
            arptable.append(temp)
        for index2 in arptable:
            log_info("ip: {}\tmac:{}\n".format(str(index2.ip),str(index2.mac)))
```

具体逻辑在实验内容中已经说明

task3

1. 实验内容：task要求构造一个arp缓存表，对于发送过arp_request的节点，记录下他的ip和对应的mac地址，其具体的构造方式为使用一个类作为list的一项，每项存放一个ip和其对应的mac地址，这样当后续实验中路由器需要发送arp申请时可以首先在自己的arp缓存表中查找，提高效率

```
if gotpkt:
    log_debug("Got a packet: {}".format(str(pkt)))
    arp = pkt.get_header(Arp)
    if arp is not None:
        intftable = self.net.interfaces()
        for index in intftable:
            if index.ipaddr == arp.targetprotoaddr:
                packet = create_ip_arp_reply(index.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprot
                # packet = create_ip_arp_reply(arp.senderhwaddr,index.ethaddr,arp.senderprotoaddr,arp.targetpr
                self.net.send_packet(dev,packet)

        flag = 0
        for index in arptable:
            if index.ip == arp.senderprotoaddr and index.mac != arp.senderhwaddr:
                arptable.remove(index) #item is changed      You, a few seconds ago • Uncommitted changes
            elif index.ip == arp.senderprotoaddr and index.mac == arp.senderhwaddr:
                flag = 1
                break
        if flag == 0:
            temp = arpitem(arp.senderprotoaddr,arp.senderhwaddr)
            arptable.append(temp)
        for index2 in arptable:
            log_info("ip: {}\tmac:{}\n".format(str(index2.ip),str(index2.mac)))
```

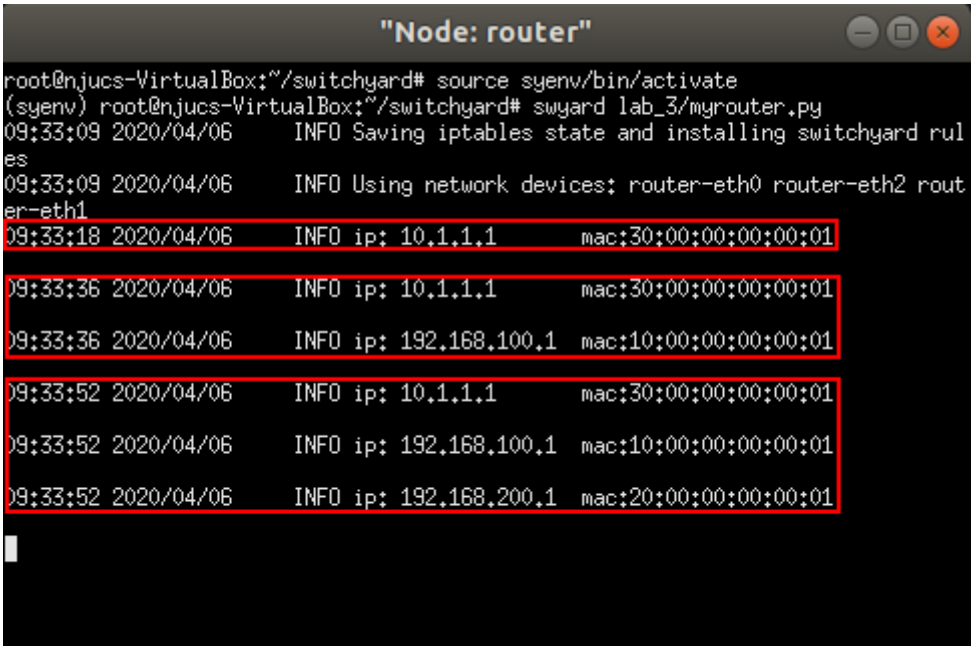
具体实现为：每次收到一个arp包时，遍历一遍arptable，查看是否缓存过该表项，如果表中有对应的ip，但是mac地址不同，则说明该项发送了变化，因此需要将其删除后再添加以达到更新的目的，否则如果表中有对应的ip和mac地址，则说明已存在，因此不需要进行任何操作，如果以上情况均不满足则说明表中没有这一项，则要进行添加

2. 实验结果：

   由于在代码中加入了每次收到arp包，则将arp缓存表输出的代码，因此构造以下流量对其进行测试：

```
mininet> xterm router
mininet> client ping -c1 server1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> server1 ping -c1 server2
PING 192.168.200.1 (192.168.200.1) 56(84) bytes of data.

--- 192.168.200.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> server2 ping -c1 server1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

即首先让client ping server1，然后让server1 ping server2，最后让server2 ping server1，xterm中的输出信息如下：



每个红框对应每次ping，可以看到arp缓存表正确构建

3. 核心代码：

定义类arpitem用于存放每个表项：

```
class arpitem(object):
    def __init__(self,ip ,mac):
        self.ip = ip
        self.mac = mac
```

每个表项包含ip和mac

具体实现逻辑：

```
if gotpkt:
    log_debug("Got a packet: {}".format(str(pkt)))
    arp = pkt.get_header(Arp)
    if arp is not None:
        intftable = self.net.interfaces()
        for index in intftable:
            if index.ipaddr == arp.targetprotoaddr:
                packet = create_ip_arp_reply(index.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprot
                # packet = create_ip_arp_reply(arp.senderhwaddr,index.ethaddr,arp.senderprotoaddr,arp.targetpr
                self.net.send_packet(dev,packet)

        flag = 0
        for index in arptable:
            if index.ip == arp.senderprotoaddr and index.mac != arp.senderhwaddr:
                arptable.remove(index) #item is changed      You, a few seconds ago • Uncommitted changes
            elif index.ip == arp.senderprotoaddr and index.mac == arp.senderhwaddr:
                flag = 1
                break
        if flag == 0:
            temp = arpitem(arp.senderprotoaddr,arp.senderhwaddr)
            arptable.append(temp)
        for index2 in arptable:
            log_info("ip: {}\tmac:{}\n".format(str(index2.ip),str(index2.mac)))
```

具体逻辑在实验内容中已经说明

总结与感想

lab3总体而言工作量不大，其中最难的部分是阅读一些相关文档查看相关api的说明，这在我们之前包括大一的课程中都很少遇到，一开始会感觉有点不知所措，但是后面习惯之后确实感觉很有收获，之后在工作中应该也是这样，而不是由别人告诉我们什么东西可以怎样得到，很有意义！