Lab3_Report

课程名称：**计算机网络** 任课教师：田臣/李文中 助教：

学院	计算机	专业（方向）	计算机
学号	181860077	姓名	佘帅杰
Email	3121416933@qq.com	开始完成日期	2020.4.3

实验名称

计算机网络试验3

实验目的

实现路由器

实验内容

router基本功能：对arp request的实现以及arp table cache的实现

实验结果与核心代码解读（合并了原模板的两个模块）

注：本次的第三个Task比较小，直接合并为一个任务整体测试和分析（task3也是在task2的基础上增加功能的）

代码解读

数据结构

下面是数据结构的初始化

使用了interface保存router的接口

使用ip_list保存所有的接口的ip地址

使用mac_list保存所有的ip对应的mac的地址

最后初始化一个空的字典，存储所需的arp cache table

```
    def __init__(self, net):
        self.net = net
        # other initialization stuff here
        self.interfaces = net.interfaces()
        self.ip_list=[intf.ipaddr for intf in self.interfaces]
        self.mac_list=[intf.ethaddr for intf in self.interfaces]
        self.arp_table={}
```

处理逻辑

具体的实现思路见下面的实现代码的注释

```python
if gotpkt:
     log_debug("Got a packet: {}".format(str(pkt)))
     log_info("Got a packet: {}".format(str(pkt)))
     arp = pkt.get_header(Arp)
     #拿到arp头
     if arp is None:
           log_info("Not arp Packet")#arp头可能是空的，空的就不是arp的包了
     else:
           log_info("operation kind {}".format(str(arp.operation)))
           self.arp_table[arp.senderprotoaddr]=arp.senderhwaddr#是arp包，更新表
           if arp.operation == 1:#对应的1就是request类型，见API文档
                log_info("recive arp requests")
                index=-1#初始化匹配下标
                for i in range(len(self.ip_list)):#循环找匹配
                    if self.ip_list[i]==arp.targetprotoaddr:
                         index=i
                         break
               if index != -1:#找到了
                  log_info("match the packet")
                  answer=create_ip_arp_reply(self.mac_list[index], arp.senderhwaddr, self.ip_list[index],arp.senderprotoaddr)#构造reply包，具体的结构参考了API文档
                  self.net.send_packet(dev, answer)#原路发送
                  log_info("send answer: {}".format(str(answer)))
               else:
                   log_info("no match")#没有匹配
            else:
               if arp.operation == 2:#是reply的类型
                    log_info("recive arp reply")#reply类型的两边都有信息
                    self.arp_table[arp.targetprotoaddr]=arp.targethwaddr#都更新table
                else:
                    log_info("recive unk arp")#上述两种都不是（我也不知道是啥包了）
     log_info("Table Shown as follows")#打印出当前的table
     for (k,v) in  self.arp_table.items():
          print ("%s     " % k,v )
```

测试

测试样例

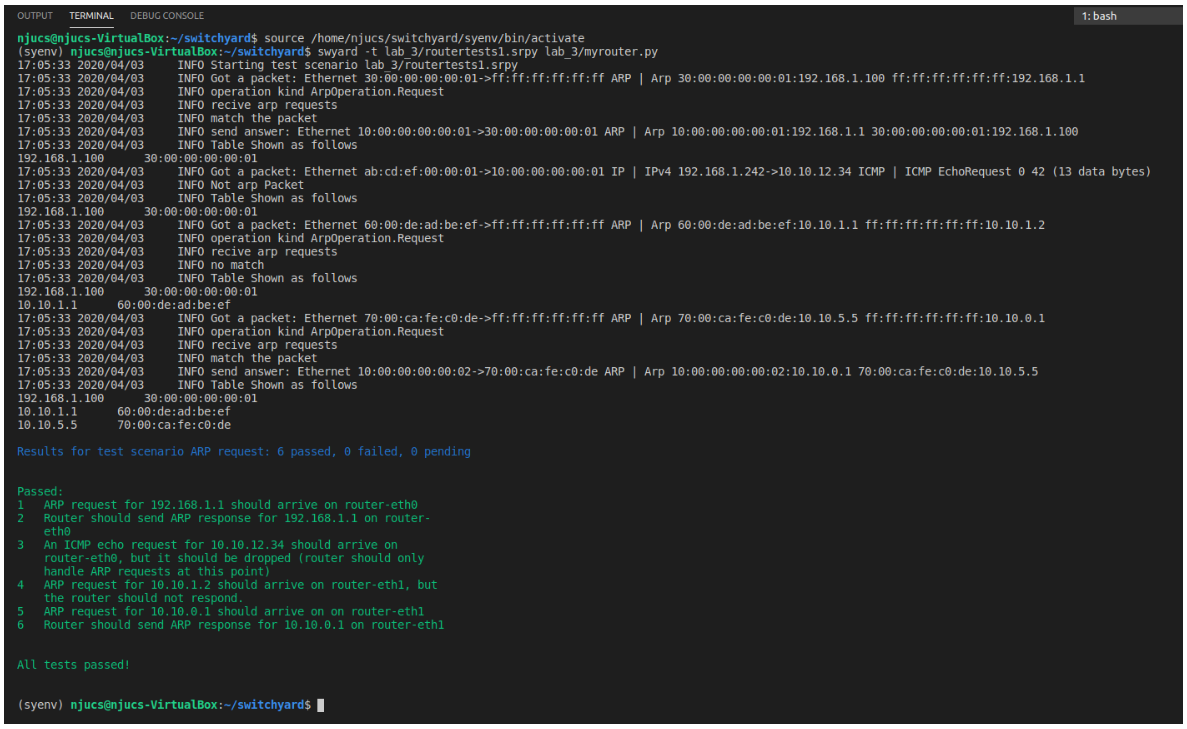
routertests1.srpy

下面利用测试的info信息来分析一下处理逻辑的正确性

```
17:05:33 2020/04/03     INFO Starting test scenario lab_3/routertests1.srpy
启动测试

17:05:33 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.1.100 ff:ff:ff:ff:ff:ff:192.168.1.1
收到广播的包

17:05:33 2020/04/03     INFO operation kind ArpOperation.Request
识别出是request的类型
```

```

17:05:33 2020/04/03     INFO recive arp requests
17:05:33 2020/04/03     INFO match the packet
匹配到对应的端口

17:05:33 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.1.1 30:00:00:00:00:01:192.168.1.100
发出应答，可以看到原来的ffff的地址被填上了结果

17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
这里可以看到request的源地址对被保存了

17:05:33 2020/04/03     INFO Got a packet: Ethernet ab:cd:ef:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.1.242->10.10.12.34 ICMP | ICMP EchoRequest 0 42 (13 data bytes)
收到了ICMP request包，显示不是处理的范围之内的，应该省略

17:05:33 2020/04/03     INFO Not arp Packet
果然识别出了

17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
Table没有改变

17:05:33 2020/04/03     INFO Got a packet: Ethernet 60:00:de:ad:be:ef->ff:ff:ff:ff:ff:ff ARP | Arp 60:00:de:ad:be:ef:10.10.1.1 ff:ff:ff:ff:ff:ff:10.10.1.2
其实和第一个时间是一样的，都是一个arp request

17:05:33 2020/04/03     INFO operation kind ArpOperation.Request
17:05:33 2020/04/03     INFO recive arp requests
识别出了是request的包

17:05:33 2020/04/03     INFO no match
但是没有匹配，按照规则应该丢弃

17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
在这个过程中又有对应的地址被保存

17:05:33 2020/04/03     INFO Got a packet: Ethernet 70:00:ca:fe:c0:de->ff:ff:ff:ff:ff:ff ARP | Arp 70:00:ca:fe:c0:de:10.10.5.5 ff:ff:ff:ff:ff:ff:10.10.0.1
收到了request的类型包

17:05:33 2020/04/03     INFO operation kind ArpOperation.Request
17:05:33 2020/04/03     INFO recive arp requests
17:05:33 2020/04/03     INFO match the packet
判断类型，并得到了匹配

17:05:33 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:02->70:00:ca:fe:c0:de ARP | Arp 10:00:00:00:00:02:10.10.0.1 70:00:ca:fe:c0:de:10.10.5.5
应答

```

```
17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:efc
10.10.5.5      70:00:ca:fe:c0:de
又多了一个数据对
```

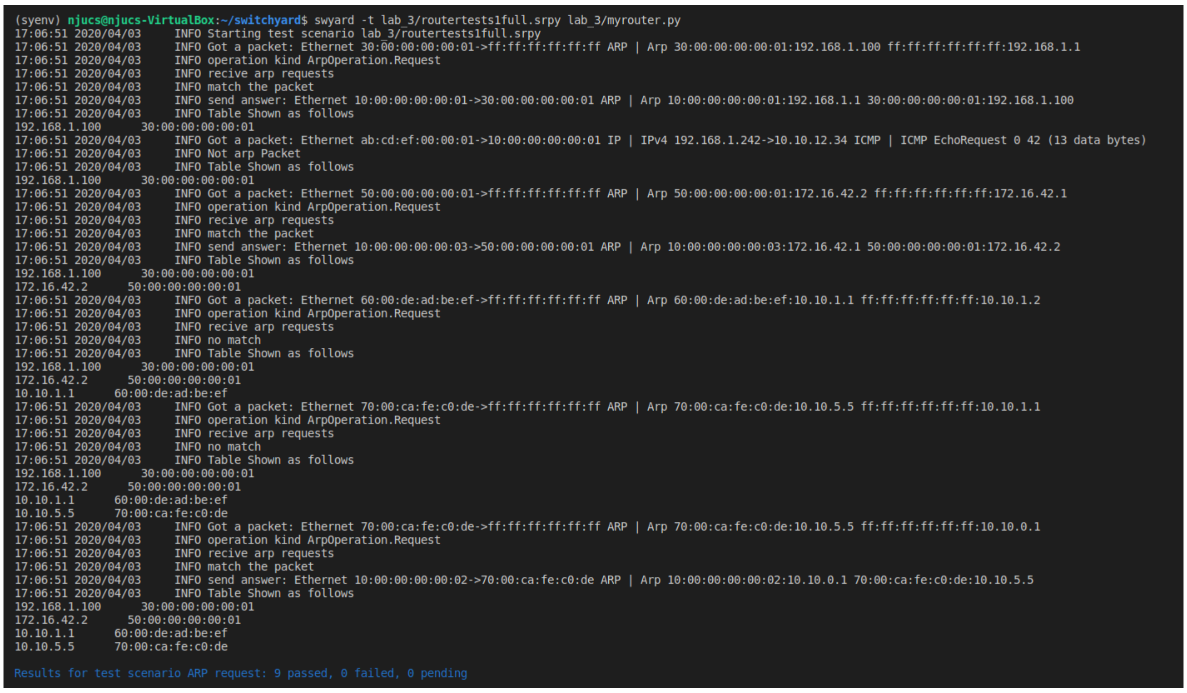


```
njucs@njucs-VirtualBox:~/switchyard$ source /home/njucs/switchyard/syenv/bin/activate
(syenv) njucs@njucs-VirtualBox:~/switchyard$ swyard -t lab_3/routertests1.srpy lab_3/myrouter.py
17:05:33 2020/04/03     INFO Starting test scenario lab_3/routertests1.srpy
17:05:33 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.1.100 ff:ff:ff:ff:ff:ff:192.168.1.1
17:05:33 2020/04/03     INFO operation kind ArpOperation.Request
17:05:33 2020/04/03     INFO recive arp requests
17:05:33 2020/04/03     INFO match the packet
17:05:33 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.1.1 30:00:00:00:00:01:192.168.1.100
17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
17:05:33 2020/04/03     INFO Got a packet: Ethernet ab:cd:ef:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.1.242->10.10.12.34 ICMP | ICMP EchoRequest 0 42 (13 data bytes)
17:05:33 2020/04/03     INFO Not arp Packet
17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
17:05:33 2020/04/03     INFO Got a packet: Ethernet 60:00:de:ad:be:ef->ff:ff:ff:ff:ff:ff ARP | Arp 60:00:de:ad:be:ef:10.10.1.1 ff:ff:ff:ff:ff:ff:10.10.1.2
17:05:33 2020/04/03     INFO operation kind ArpOperation.Request
17:05:33 2020/04/03     INFO recive arp requests
17:05:33 2020/04/03     INFO no match
17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
17:05:33 2020/04/03     INFO Got a packet: Ethernet 70:00:ca:fe:c0:de->ff:ff:ff:ff:ff:ff ARP | Arp 70:00:ca:fe:c0:de:10.10.5.5 ff:ff:ff:ff:ff:ff:10.10.0.1
17:05:33 2020/04/03     INFO operation kind ArpOperation.Request
17:05:33 2020/04/03     INFO recive arp requests
17:05:33 2020/04/03     INFO match the packet
17:05:33 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:02->70:00:ca:fe:c0:de ARP | Arp 10:00:00:00:00:02:10.10.0.1 70:00:ca:fe:c0:de:10.10.5.5
17:05:33 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
10.10.5.5      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$
```

routertests1full.srpy

```
17:06:51 2020/04/03     INFO Starting test scenario lab_3/routertests1full.srpy
17:06:51 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.1.100 ff:ff:ff:ff:ff:ff:192.168.1.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO match the packet
17:06:51 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.1.1 30:00:00:00:00:01:192.168.1.100
上述就是收到了包，并识别出了是request类型，并发送了对应的应答结果

17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
经过了这一次在cache table里有了新的表项

17:06:51 2020/04/03     INFO Got a packet: Ethernet ab:cd:ef:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.1.242->10.10.12.34 ICMP | ICMP EchoRequest 0 42 (13 data bytes)
17:06:51 2020/04/03     INFO Not arp Packet
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
类型是ICMP EchoRequest，无需操作，table不变

```

```
17:06:51 2020/04/03     INFO Got a packet: Ethernet 50:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 50:00:00:00:00:01:172.16.42.2 ff:ff:ff:ff:ff:ff:172.16.42.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO match the packet
17:06:51 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:03->50:00:00:00:00:01 ARP | Arp 10:00:00:00:00:03:172.16.42.1 50:00:00:00:00:01:172.16.42.2
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
收到了request包且匹配上了，发送应答，并在表项里加上了新的一项

17:06:51 2020/04/03     INFO Got a packet: Ethernet 60:00:de:ad:be:ef->ff:ff:ff:ff:ff:ff ARP | Arp 60:00:de:ad:be:ef:10.10.1.1 ff:ff:ff:ff:ff:ff:10.10.1.2
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO no match
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
没有符合的，直接丢弃，并在表项里更新

17:06:51 2020/04/03     INFO Got a packet: Ethernet 70:00:ca:fe:c0:de->ff:ff:ff:ff:ff:ff ARP | Arp 70:00:ca:fe:c0:de:10.10.5.5 ff:ff:ff:ff:ff:ff:10.10.1.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO no match
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
10.10.5.5      70:00:ca:fe:c0:de
没有匹配的。同上述丢弃并更新

17:06:51 2020/04/03     INFO Got a packet: Ethernet 70:00:ca:fe:c0:de->ff:ff:ff:ff:ff:ff ARP | Arp 70:00:ca:fe:c0:de:10.10.5.5 ff:ff:ff:ff:ff:ff:10.10.0.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO match the packet
17:06:51 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:02->70:00:ca:fe:c0:de ARP | Arp 10:00:00:00:00:02:10.10.0.1 70:00:ca:fe:c0:de:10.10.5.5
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
10.10.5.5      70:00:ca:fe:c0:de
匹配，发送应答并更新，同时发现这个记录过了，不再添加
```



```
(syenv) njucs@njucs-VirtualBox:~/switchyard$ swyard -t lab_3/routertests1full.srpy lab_3/myrouter.py
17:06:51 2020/04/03     INFO Starting test scenario lab_3/routertests1full.srpy
17:06:51 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.1.100 ff:ff:ff:ff:ff:ff:192.168.1.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO match the packet
17:06:51 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.1.1 30:00:00:00:00:01:192.168.1.100
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
17:06:51 2020/04/03     INFO Got a packet: Ethernet ab:cd:ef:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.1.242->10.10.12.34 ICMP | ICMP EchoRequest 0 42 (13 data bytes)
17:06:51 2020/04/03     INFO Not arp Packet
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
17:06:51 2020/04/03     INFO Got a packet: Ethernet 50:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 50:00:00:00:00:01:172.16.42.2 ff:ff:ff:ff:ff:ff:172.16.42.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO match the packet
17:06:51 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:03->50:00:00:00:00:01 ARP | Arp 10:00:00:00:00:03:172.16.42.1 50:00:00:00:00:01:172.16.42.2
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
17:06:51 2020/04/03     INFO Got a packet: Ethernet 60:00:de:ad:be:ef->ff:ff:ff:ff:ff:ff ARP | Arp 60:00:de:ad:be:ef:10.10.1.1 ff:ff:ff:ff:ff:ff:10.10.1.2
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO no match
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
17:06:51 2020/04/03     INFO Got a packet: Ethernet 70:00:ca:fe:c0:de->ff:ff:ff:ff:ff:ff ARP | Arp 70:00:ca:fe:c0:de:10.10.5.5 ff:ff:ff:ff:ff:ff:10.10.1.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO no match
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
10.10.5.5      70:00:ca:fe:c0:de
17:06:51 2020/04/03     INFO Got a packet: Ethernet 70:00:ca:fe:c0:de->ff:ff:ff:ff:ff:ff ARP | Arp 70:00:ca:fe:c0:de:10.10.5.5 ff:ff:ff:ff:ff:ff:10.10.0.1
17:06:51 2020/04/03     INFO operation kind ArpOperation.Request
17:06:51 2020/04/03     INFO recive arp requests
17:06:51 2020/04/03     INFO match the packet
17:06:51 2020/04/03     INFO send answer: Ethernet 10:00:00:00:00:02->70:00:ca:fe:c0:de ARP | Arp 10:00:00:00:00:02:10.10.0.1 70:00:ca:fe:c0:de:10.10.5.5
17:06:51 2020/04/03     INFO Table Shown as follows
192.168.1.100      30:00:00:00:00:01
172.16.42.2      50:00:00:00:00:01
10.10.1.1      60:00:de:ad:be:ef
10.10.5.5      70:00:ca:fe:c0:de

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

部署至mininet

example

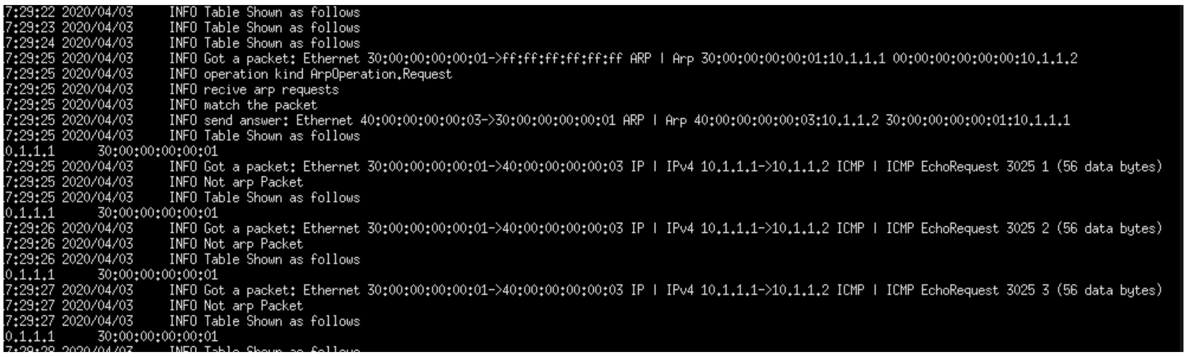
利用文档里的example进行测试

就是利用client ping -c3 来测试，一开始的request被router接受并处理，发回应答包

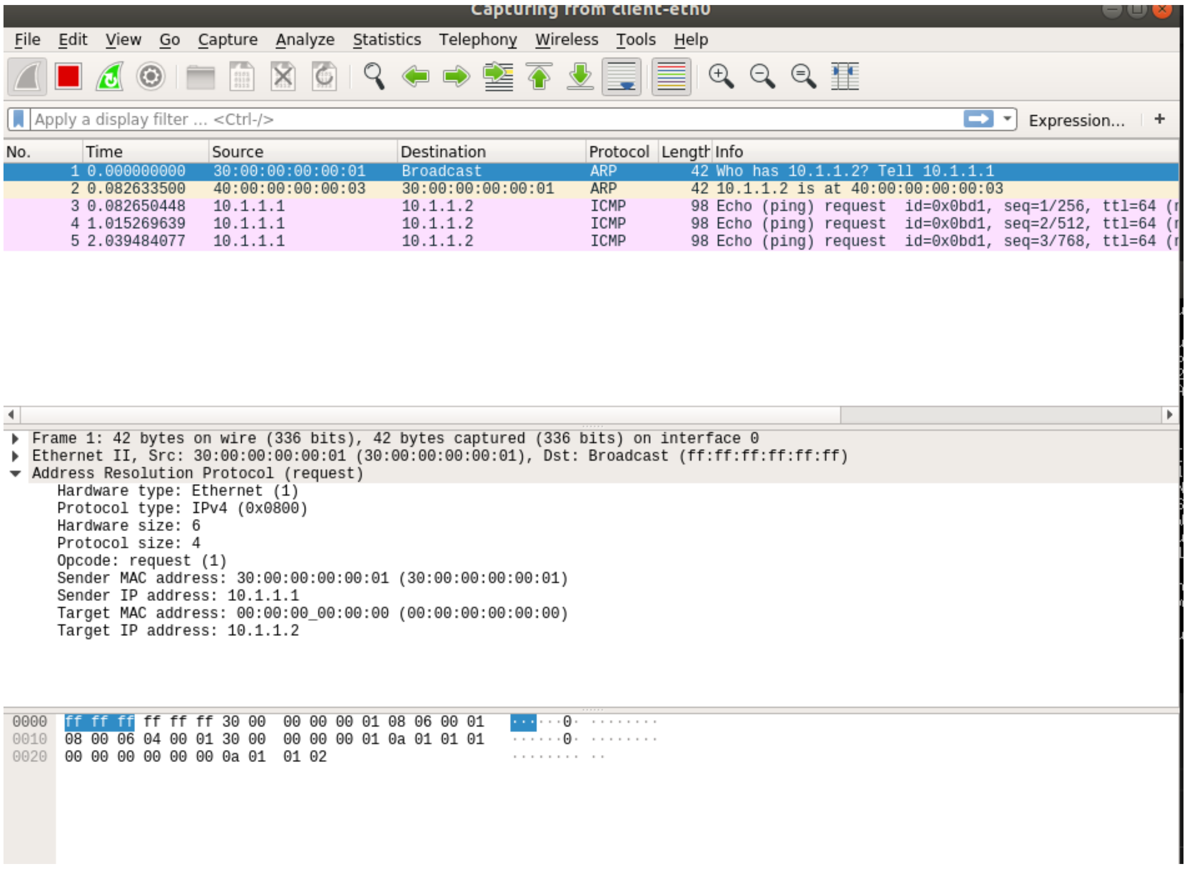
随后的三个包不在处理范围跳过

后面的wireshark结果也印证了

（**注**：可以看到下面反复的打印“Table Shown as follow”，这个是测试的时候设置的每一个循环都打印列表，在提交版本里进行了修改，只会在收到包了才打印）



```
7:29:22 2020/04/03     INFO Table Shown as follows
7:29:23 2020/04/03     INFO Table Shown as follows
7:29:24 2020/04/03     INFO Table Shown as follows
7:29:25 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:10.1.1.1 00:00:00:00:00:00:10.1.1.2
7:29:25 2020/04/03     INFO operation kind ArpOperation.Request
7:29:25 2020/04/03     INFO recive arp requests
7:29:25 2020/04/03     INFO match the packet
7:29:25 2020/04/03     INFO send answer: Ethernet 40:00:00:00:00:03->30:00:00:00:00:01 ARP | Arp 40:00:00:00:00:03:10.1.1.2 30:00:00:00:00:01:10.1.1.1
7:29:25 2020/04/03     INFO Table Shown as follows
0.1.1.1      30:00:00:00:00:01
7:29:25 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:00:00:03 IP | IPv4 10.1.1.1->10.1.1.2 ICMP | ICMP EchoRequest 3025 1 (56 data bytes)
7:29:25 2020/04/03     INFO Not arp Packet
7:29:25 2020/04/03     INFO Table Shown as follows
0.1.1.1      30:00:00:00:00:01
7:29:26 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:00:00:03 IP | IPv4 10.1.1.1->10.1.1.2 ICMP | ICMP EchoRequest 3025 2 (56 data bytes)
7:29:26 2020/04/03     INFO Not arp Packet
7:29:26 2020/04/03     INFO Table Shown as follows
0.1.1.1      30:00:00:00:00:01
7:29:27 2020/04/03     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:00:00:03 IP | IPv4 10.1.1.1->10.1.1.2 ICMP | ICMP EchoRequest 3025 3 (56 data bytes)
7:29:27 2020/04/03     INFO Not arp Packet
7:29:27 2020/04/03     INFO Table Shown as follows
0.1.1.1      30:00:00:00:00:01
```


```
No.  Time         Source              Destination         Protocol Length Info
   1 0.000000000  30:00:00:00:00:01   Broadcast           ARP      42 Who has 10.1.1.2? Tell 10.1.1.1
   2 0.082633500  40:00:00:00:00:03   30:00:00:00:00:01   ARP      42 10.1.1.2 is at 40:00:00:00:00:03
   3 0.082650448  10.1.1.1            10.1.1.2            ICMP     98 Echo (ping) request  id=0x0bd1, seq=1/256, ttl=64 (
   4 1.015269639  10.1.1.1            10.1.1.2            ICMP     98 Echo (ping) request  id=0x0bd1, seq=2/512, ttl=64 (
   5 2.039484077  10.1.1.1            10.1.1.2            ICMP     98 Echo (ping) request  id=0x0bd1, seq=3/768, ttl=64 (

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: 30:00:00:00:00:01 (30:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 30:00:00:00:00:01 (30:00:00:00:00:01)
    Sender IP address: 10.1.1.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.1.1.2

0000  ff ff ff ff ff ff 30 00  00 00 00 01 08 06 00 01   ······0· ········
0010  08 00 06 04 00 01 30 00  00 00 00 01 0a 01 01 01   ······0· ········
0020  00 00 00 00 00 00 0a 01  01 02                     ········ ··
```

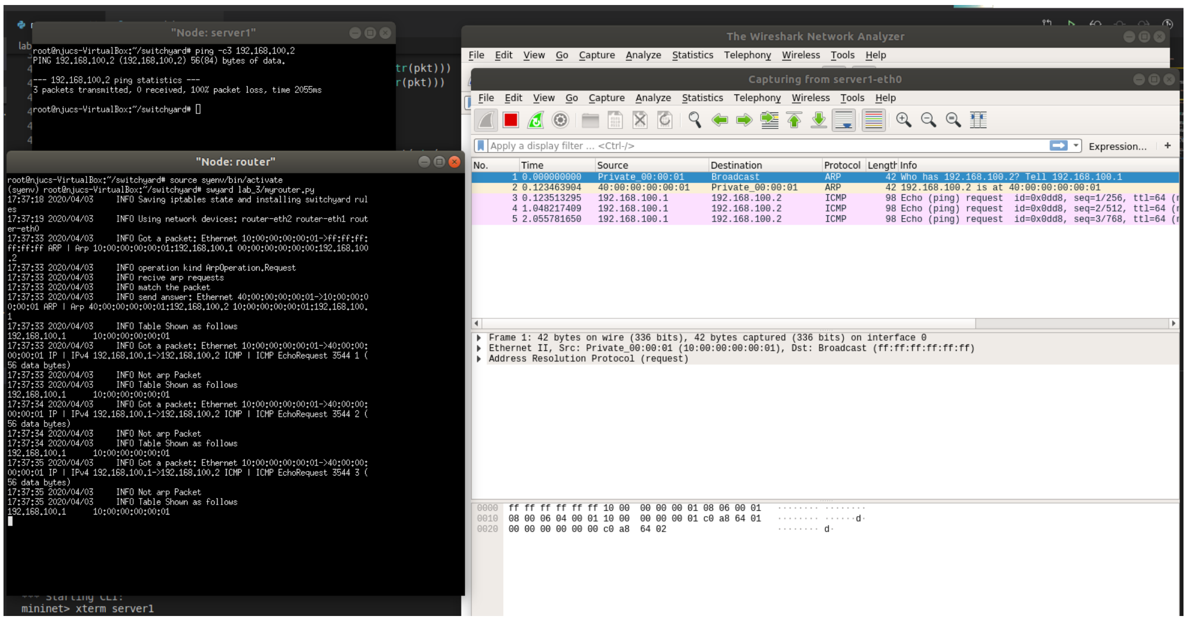
my_example

使用server1 ping -c3 192.168.100.2来对router进行流量构造

最后的结果如下所示

可以见到wireshark抓包的结果是和上面client的测试是一样的

就是先广播（只知道ip不知道mac）然后路由器得到了这个arp的request，进行处理（处理的流程上面说过），基本就是识别出包的类别，然后构造出一个arp的reply然后原路放回，同时在Table里写下（更新）新的表项。随后建立的ICMP包不在处理范围之内，也得到了识别并不做处理。


```
root@njucs-VirtualBox:~/switchyard# ping -c3 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.

--- 192.168.100.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2055ms

root@njucs-VirtualBox:~/switchyard# 
```

```
root@njucs-VirtualBox:~/switchyard# source syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/switchyard# swyard lab_3/myrouter.py
17:37:18 2020/04/03     INFO Saving iptables state and installing switchyard rul
es
17:37:19 2020/04/03     INFO Using network devices: router-eth2 router-eth1 rout
er-eth0
17:37:33 2020/04/03     INFO Got a packet: Ethernet 10:00:00:00:00:01->ff:ff:ff:
ff:ff:ff ARP | Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.100
.2
17:37:33 2020/04/03     INFO operation kind ArpOperation.Request
17:37:33 2020/04/03     INFO recive arp requests
17:37:33 2020/04/03     INFO match the packet
17:37:33 2020/04/03     INFO send answer: Ethernet 40:00:00:00:00:01->10:00:00:0
0:00:01 ARP | Arp 40:00:00:00:00:01:192.168.100.2 10:00:00:00:00:01:192.168.100.
1
17:37:33 2020/04/03     INFO Table Shown as follows
192.168.100.1      10:00:00:00:00:01
17:37:33 2020/04/03     INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:
00:00:01 IP | IPv4 192.168.100.1->192.168.100.2 ICMP | ICMP EchoRequest 3544 1 (
56 data bytes)
17:37:33 2020/04/03     INFO Not arp Packet
17:37:33 2020/04/03     INFO Table Shown as follows
192.168.100.1      10:00:00:00:00:01
17:37:34 2020/04/03     INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:
00:00:01 IP | IPv4 192.168.100.1->192.168.100.2 ICMP | ICMP EchoRequest 3544 2 (
56 data bytes)
17:37:34 2020/04/03     INFO Not arp Packet
17:37:34 2020/04/03     INFO Table Shown as follows
192.168.100.1      10:00:00:00:00:01
17:37:35 2020/04/03     INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:
00:00:01 IP | IPv4 192.168.100.1->192.168.100.2 ICMP | ICMP EchoRequest 3544 3 (
56 data bytes)
17:37:35 2020/04/03     INFO Not arp Packet
17:37:35 2020/04/03     INFO Table Shown as follows
192.168.100.1      10:00:00:00:00:01
```

```
mininet> xterm server1
```

```
No.  Time         Source              Destination         Protocol Length Info
   1 0.000000000  Private_00:00:01    Broadcast           ARP      42 Who has 192.168.100.2? Tell 192.168.100.1
   2 0.123463904  40:00:00:00:00:01   Private_00:00:01    ARP      42 192.168.100.2 is at 40:00:00:00:00:01
   3 0.123513295  192.168.100.1       192.168.100.2       ICMP     98 Echo (ping) request  id=0x0dd8, seq=1/256, ttl=64 (
   4 1.048217409  192.168.100.1       192.168.100.2       ICMP     98 Echo (ping) request  id=0x0dd8, seq=2/512, ttl=64 (
   5 2.055781650  192.168.100.1       192.168.100.2       ICMP     98 Echo (ping) request  id=0x0dd8, seq=3/768, ttl=64 (

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 10 00  00 00 00 01 08 06 00 01   ········ ········
0010  08 00 06 04 00 01 10 00  00 00 00 01 c0 a8 64 01   ········ ······d·
0020  00 00 00 00 00 00 c0 a8  64 02                     ········ d·
```

总结与感想

暂无。。。