Lab 3: Respond to ARP

181180050 孔孟荀

Task 1: Preparation

略

Task 2: Handle ARP Request

coding：

```
class Router(object):
    def __init__(self, net: switchyard.llnetbase.LLNetBase):
        self.net = net
        # other initialization stuff here

    def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
        timestamp, ifaceName, packet = recv

        if packet.has_header(Arp) == False:
            log_info("Received a non-arp packet")
        else:
            arp = packet.get_header(Arp)
            #itface = self.net.interface_by_ipaddr(arp.targetprotoaddr)

            for  itface in self.net.interfaces():
                if arp.targetprotoaddr == itface.ipaddr:
                    pket=create_ip_arp_reply(itface.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprotoaddr
                    self.net.send_packet(ifaceName,pket)
```

思路&&碰到的问题:

- 首先判断有没有arp包，一开始我直接用arp=packet.get_header(Arp),判断返回的是

不是None，但是会报Attribute error,后来干脆用has_header()函数，解决了问题

- 然后判断给的ip address在不在router的接口对应的ipaddress里，一开始我直接用itface = self.net.interface_by_ipaddr(arp.targetprotoaddr)，又想判断这个itface是不是None，但是发现这个函数的源码没有返回None这个操作，而是返回一个报错KeyError，所以干脆放弃了这个函数

```
def interface_by_ipaddr(self, ipaddr):
    '''
    Given an IP address, return the interface that 'owns' this address
    '''
    ipaddr = IPAddr(ipaddr)
    for devname,iface in self._devinfo.items():
        if iface.ipaddr == ipaddr:
            return iface
    raise KeyError("No device has IP address {}".format(ipaddr))
```

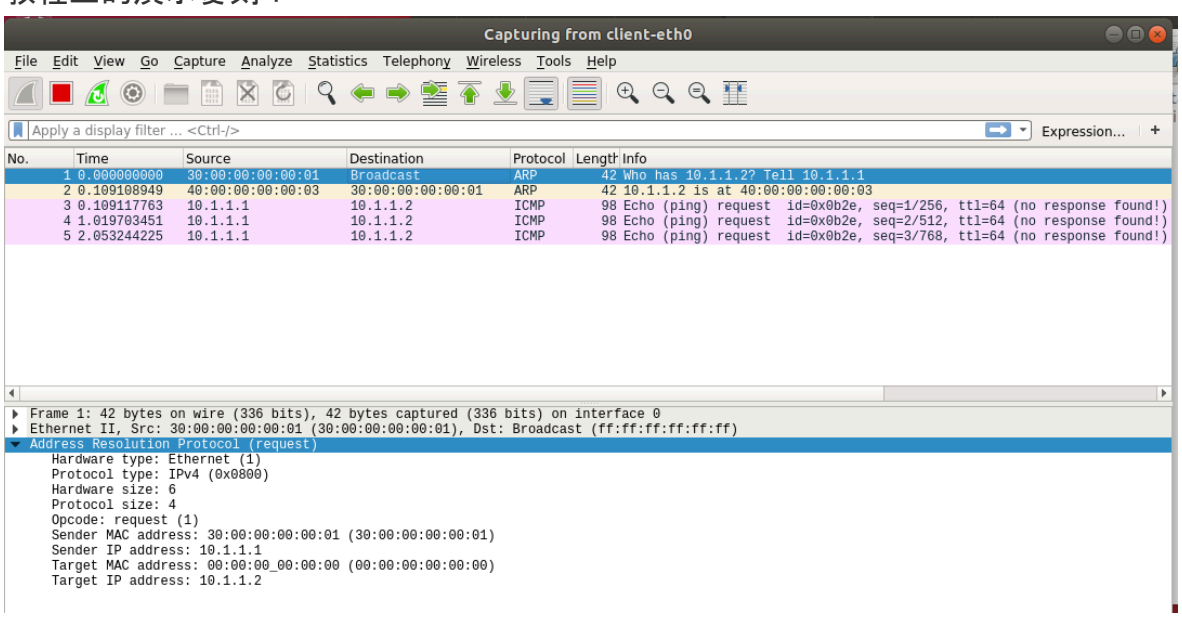
- 最后还要用提供的create_ip_arp_reply函数构造一个arp reply包并且返回，只要补充最开始的request包缺少的 destination Ethernet address信息(现在是source了)即可

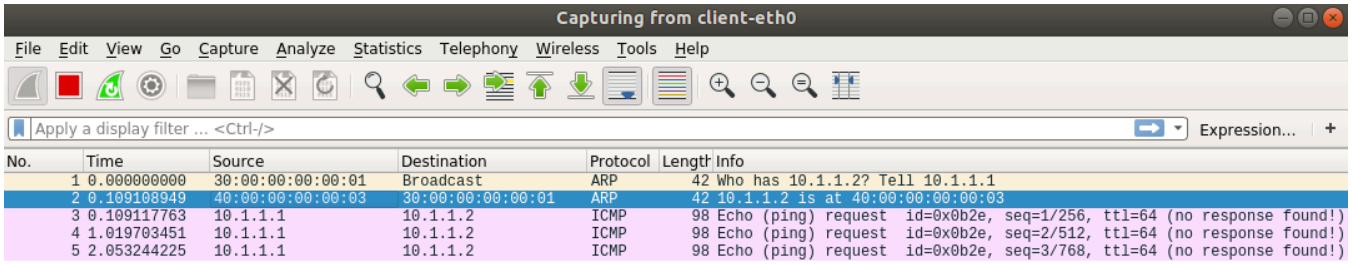
testing:

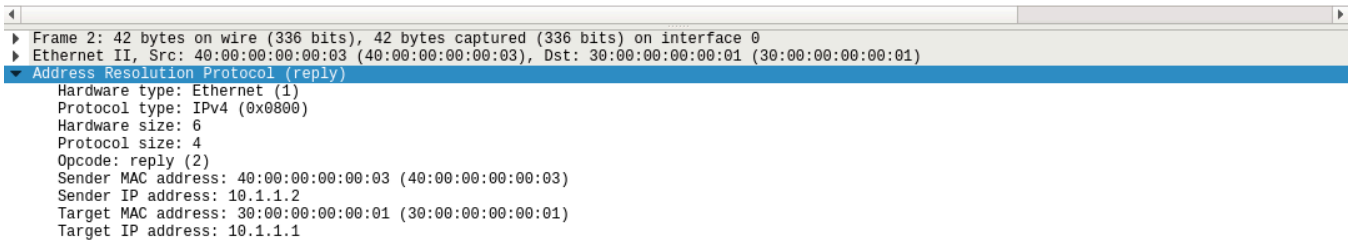
```
njucs@njucs-VirtualBox: ~/networklab/lab-3-saltfishmx
File  Edit  View  Search  Terminal  Help
16:29:43 2021/04/07     INFO Starting test scenario testcases/myrouter1_testsena
rio.srpy
16:29:43 2021/04/07     INFO Received a non-arp packet

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:~/networklab/lab-3-saltfishmx$
```

deploying:

1. 教程上的演示复刻：







- arp request的dest mac address 为全0
- arp reply 的 四个地址都已经填好，且与arp request 发生了一些互换
- 因为现在的router只能处理arp，对ping没有回应

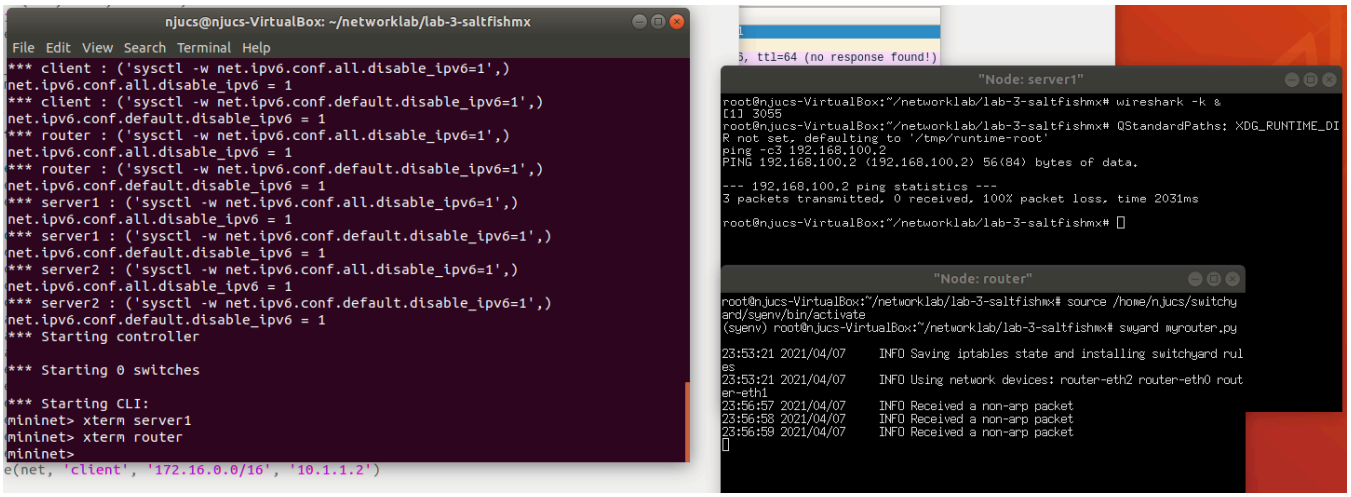
2. task：ping the router from another host (server1 or server2). Using Wireshark to prove that you have handled ARP requests well.

我决定从server1开ping

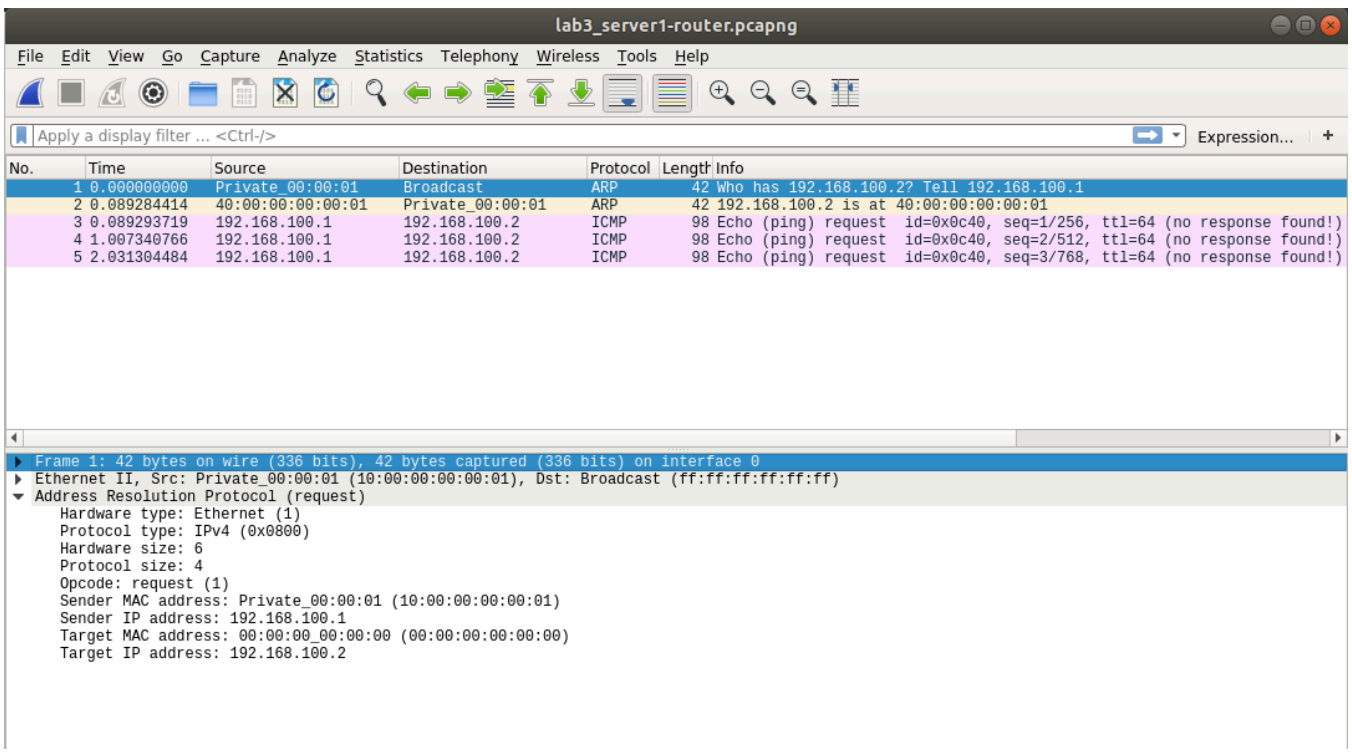
```
def setup_addressing(net):
    reset_macs(net, 'server1', '10:00:00:00:00:{:02x}')
    reset_macs(net, 'server2', '20:00:00:00:00:{:02x}')
    reset_macs(net, 'client', '30:00:00:00:00:{:02x}')
    reset_macs(net, 'router', '40:00:00:00:00:{:02x}')
    set_ip_pair(net, 'server1', 'router', '192.168.100.1/30', '192.168.100.2/30')
    set_ip_pair(net, 'server2', 'router', '192.168.200.1/30', '192.168.200.2/30')
    set_ip_pair(net, 'client', 'router', '10.1.1.1/30', '10.1.1.2/30')
    set_route(net, 'server1', '10.1.0.0/16', '192.168.100.2')
    set_route(net, 'server1', '192.168.200.0/24', '192.168.100.2')
    set_route(net, 'server2', '10.1.0.0/16', '192.168.200.2')
    set_route(net, 'server2', '192.168.100.0/24', '192.168.200.2')
    set_route(net, 'client', '192.168.100.0/24', '10.1.1.2')
    set_route(net, 'client', '192.168.200.0/24', '10.1.1.2')
    set_route(net, 'client', '172.16.0.0/16', '10.1.1.2')
```

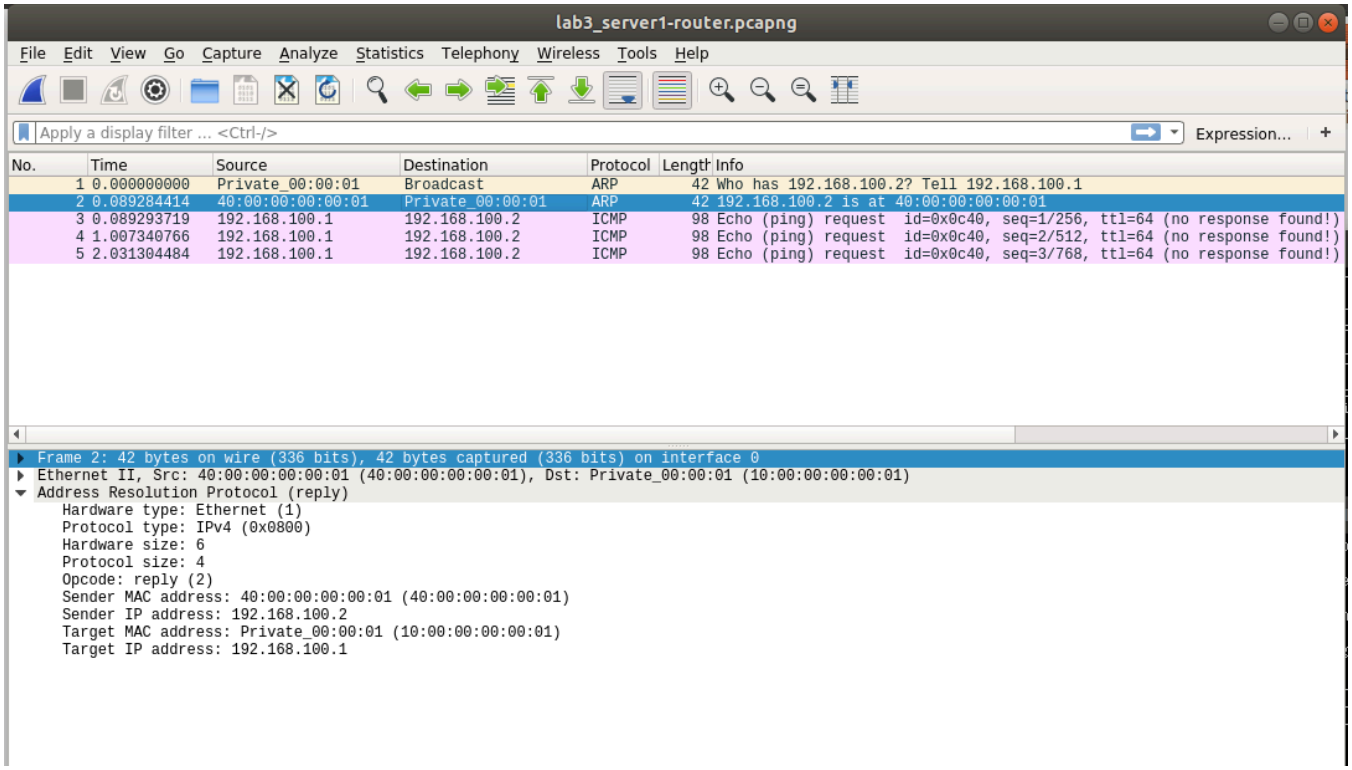
从start_mininet.py的地址设定（上图）可以看到，对于server1，要ping router， 可以用地址192.168.100.2（还是用10.1.1.2也行）

指令如下：


njucs@njucs-VirtualBox: ~/networklab/lab-3-saltfishmx
File Edit View Search Terminal Help
*** client : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** client : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** router : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** router : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** server2 : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** server2 : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** Starting controller
*** Starting 0 switches
*** Starting CLI:
mininet> xterm server1
mininet> xterm router
mininet>
e(net, 'client', '172.16.0.0/16', '10.1.1.2')
3, ttl=64 (no response found!)
"Node: server1"
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# wireshark -k &
[1] 3055
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# QStandardPaths: XDG_RUNTIME_DI
R not set, defaulting to '/tmp/runtime-root'
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 2031ms
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx#
"Node: router"
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# source /home/njucs/switchy
ard/syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# swyard myrouter.py
23:53:21 2021/04/07 INFO Saving iptables state and installing switchyard rul
es
23:53:21 2021/04/07 INFO Using network devices: router-eth2 router-eth0 rout
er-eth1
23:56:57 2021/04/07 INFO Received a non-arp packet
23:56:58 2021/04/07 INFO Received a non-arp packet
23:56:59 2021/04/07 INFO Received a non-arp packet

抓包结果如下：


lab3_server1-router.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
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 0.089284414 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 0.089293719 192.168.100.1 192.168.100.2 ICMP 98 Echo (ping) request id=0x0c40, seq=1/256, ttl=64 (no response found!)
4 1.007340766 192.168.100.1 192.168.100.2 ICMP 98 Echo (ping) request id=0x0c40, seq=2/512, ttl=64 (no response found!)
5 2.031304484 192.168.100.1 192.168.100.2 ICMP 98 Echo (ping) request id=0x0c40, seq=3/768, ttl=64 (no response found!)
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)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: Private_00:00:01 (10:00:00:00:00:01)
Sender IP address: 192.168.100.1
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.100.2



可见，router成功完成了根据发来的arp request 填上四个地址，并且发送arp reply的任务，而且对于icmp 的ping包，没有做响应，说明实现成功。

Task 3: Cached ARP Table

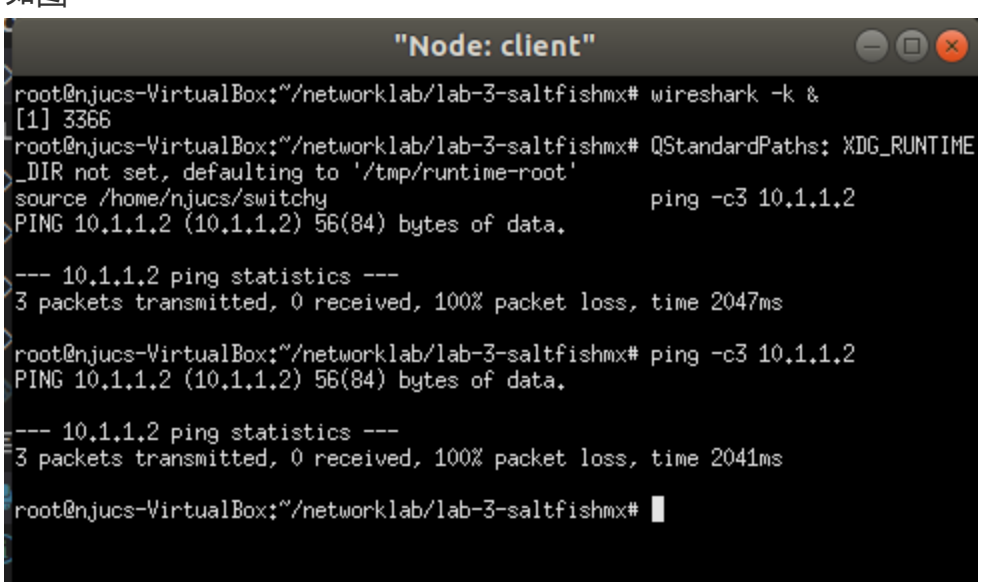
coding:

采用和lab2里switch_table一样的方法，用一个字典，每个ip 对应一个mac address和一个包收到的时间

```
class Router(object):
    def __init__(self, net: switchyard.llnetbase.LLNetBase):
        self.net = net
        self.arp_table={}
        # other initialization stuff here
    def update_table(self,duration):
        for key in list(self.arp_table):
            if time.time()-self.arp_table[key][1]>duration:
                #log_info(f"out of time , now poping {key}--{arp_table[key]} out of arp_table")
                self.arp_table.pop(key)

    def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
        timestamp, ifaceName, packet = recv

        if packet.has_header(Arp) == False:
            log_info("Received a non-arp packet")
        else:
            arp = packet.get_header(Arp)
            #itface = self.net.interface_by_ipaddr(arp.targetprotoaddr)
            self.arp_table[arp.senderprotoaddr]=[arp.senderhwaddr,time.time()]
            self.update_table(10)
            log_info(f"now the arp_table looked like:{self.arp_table}")
            for  itface in self.net.interfaces():
                if arp.targetprotoaddr == itface.ipaddr:
                    pket=create_ip_arp_reply(itface.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprotoaddr
                    self.net.send_packet(ifaceName,pket)
```

testing:

如图



```
"Node: server1"
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# wireshark -k &
[1] 3308
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# QStandardPaths: XDG_RUNTIME
_DIR not set, defaulting to '/tmp/runtime-root'
source /home/njucs/switchy                          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 2056ms

root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx#
```

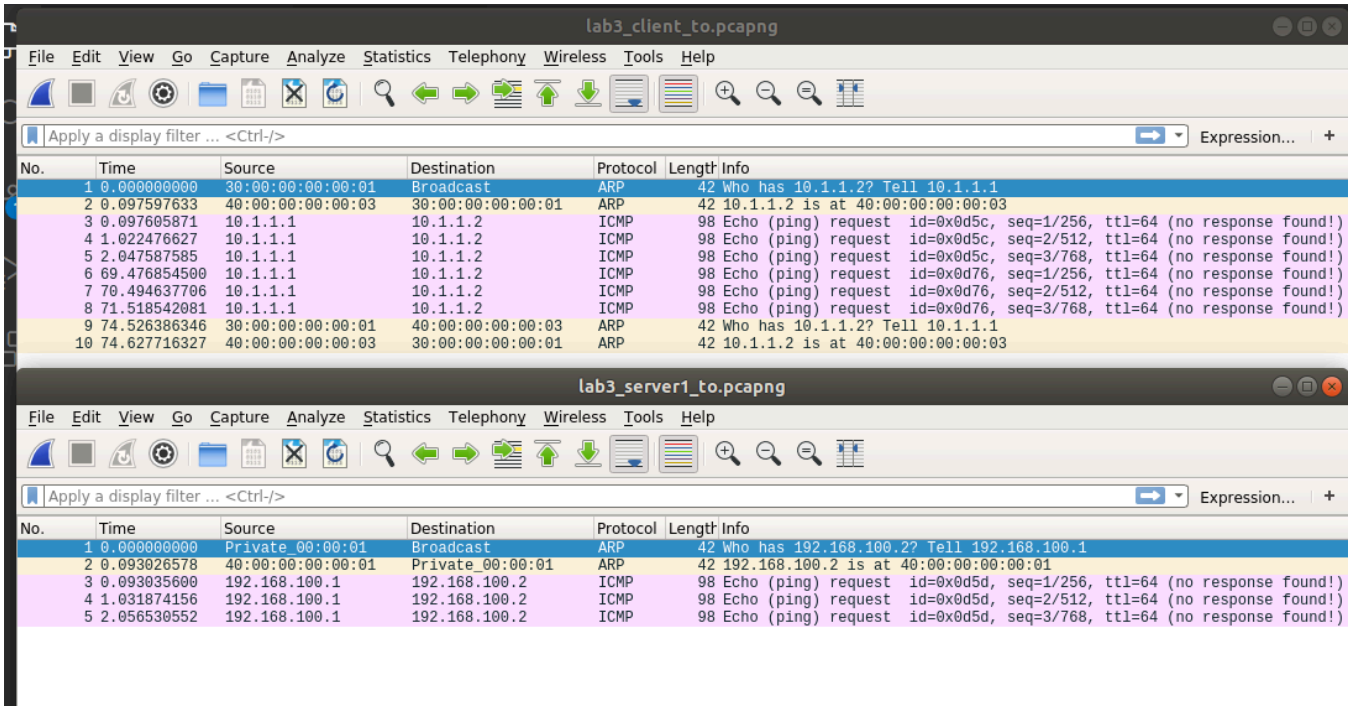
让client ping router，紧接着server1 ping router，可以见到router的arp_table的entry从0变为1再变为2

由于我实现了timeout策略，duration设置为10s，过了10s多我再让client ping server1，在update 以后，**可以见到server1的entry已经被删除了**，arp_table此时保存了client的entry，并且更新了它的时间戳

所以说明**arp_table实现成功**

```
"Node: router"
root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# source /home/njucs/switchy
ard/syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/networklab/lab-3-saltfishmx# swyard myrouter.py

10:56:02 2021/04/08     INFO Saving iptables state and installing switchyard rul
es
10:56:02 2021/04/08     INFO Using network devices: router-eth0 router-eth1 rout
er-eth2
10:56:46 2021/04/08     INFO now the arp_table looked like:{IPv4Address('10.1.1.
1'): [EthAddr('30:00:00:00:00:01'), 1617850606.765625]}
10:56:46 2021/04/08     INFO Received a non-arp packet
10:56:47 2021/04/08     INFO Received a non-arp packet
10:56:48 2021/04/08     INFO Received a non-arp packet
10:56:56 2021/04/08     INFO now the arp_table looked like:{IPv4Address('10.1.1.
1'): [EthAddr('30:00:00:00:00:01'), 1617850606.765625], IPv4Address('192.168.100
.1'): [EthAddr('10:00:00:00:00:01'), 1617850616.7359476]}
10:56:56 2021/04/08     INFO Received a non-arp packet
10:56:57 2021/04/08     INFO Received a non-arp packet
10:56:58 2021/04/08     INFO Received a non-arp packet
10:57:56 2021/04/08     INFO Received a non-arp packet
10:57:57 2021/04/08     INFO Received a non-arp packet
10:57:58 2021/04/08     INFO Received a non-arp packet
10:58:01 2021/04/08     INFO now the arp_table looked like:{IPv4Address('10.1.1.
1'): [EthAddr('30:00:00:00:00:01'), 1617850681.2958326]}
```

下图为抓包结果，与上述流程一致



实验总结

本次实验有了上次写switch_table的经验，完成的比较顺利，中途遇到的问题主要是attribute error，已经在上文提及

收工