

# 南京大学本科生实验报告

课程名称：计算机网络	任课教师：田臣 助教：方毓楚/于沛文/郑浩/陈伟/李国浩/李睿宸/杨溢...
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：201220096	姓名：钟亚晨
Email: <a href="mailto:2991908515@qq.com">2991908515@qq.com</a>	开始/完成日期：2022.04.05-2022.04.05

## 1.实验名称

IPv4 Router: Respond to ARP

完成清单：

- ✧ 完成了 IPv4 Router 对 ARP Requests 的接收和处理，并通过了框架所给的测试用例；
- ✧ 编写了对 IPv4 Router 工作逻辑的自定义测试用例，并通过了所有测试用例；
- ✧ 完成了 IPv4 Router 的 ARP Table 工作逻辑，并且在基础要求之上添加了超时机制；
- ✧ 利用 log\_info 结合终端对所实现的超时机制进行了验证；

## 2.实验目的

- ✧ 实现 IPv4 Router 对 ARP Requests 的响应逻辑；
- ✧ 掌握 ARP 请求的完整工作机制；
- ✧ 掌握路由器中 ARP 表的相关知识；
- ✧ 进一步巩固 switchyard 框架的使用以及官方文档的查阅，同时熟练测试用例的设计和编写；

## 3.实验内容

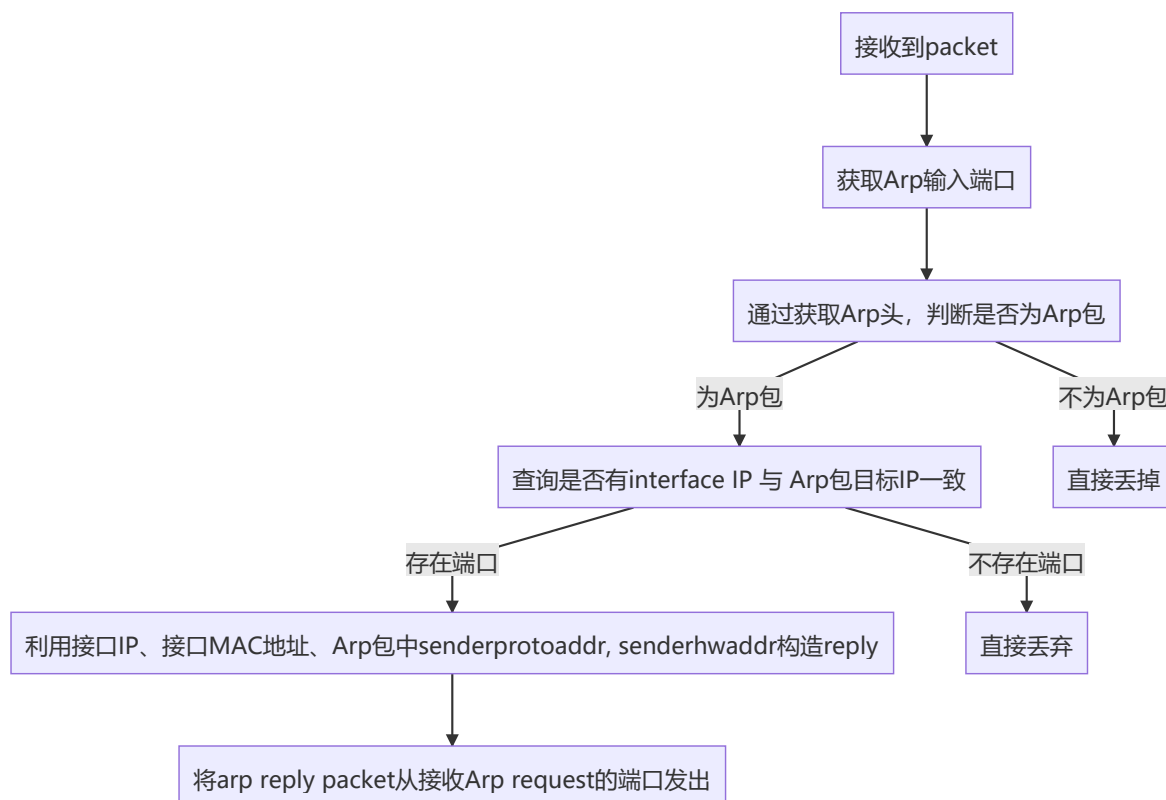
- ✧ 完成 IPv4 Router 对 ARP Request 的响应逻辑，并通过测试；
- ✧ 初步完成 IPv4 Router 的 ARP Table 逻辑；

## 4.实验结果

简单说明，以下对实验说明中的五个问题进行回答。

## 0x01 Show how you implement the logic of responding to the ARP request.

对于 IPv4 Router 响应 ARP requests 的工作逻辑可以由如下流程图进行显示：



则实现逻辑也是如图所示：

```
13 class Router(object):
14     def __init__(self, net: switchyard.llnetbase.LLNetBase):
15         self.net = net
16         self.interfaces = self.net.interfaces()
17         self.arp_table = {}
18
19     def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
20         timestamp, ifaceName, packet = recv
21         # TODO: your logic here
22         for key in list(self.arp_table.keys()):
23             if(time.time() >= self.arp_table[key][1]):
24                 self.arp_table.pop(key)
25         arp = packet.get_header(Arp)
26         if arp:
27             self.arp_table[arp.senderprotoaddr] = [arp.senderhwaddr, time.time() + 1200] # TTL: current time + 20 minutes
28             for it in self.interfaces:
29                 if it.ipaddr == arp.targetprotoaddr:
30                     reply_pkt = create_ip_arp_reply(it.ethaddr, arp.senderhwaddr, it.ipaddr, arp.senderprotoaddr)
31                     self.net.send_packet(ifaceName, reply_pkt)
```

①第16行：为对象初始化一个包含所有端口的列表，对于列表中每一个表项是什么类型、有什么属性，可以通过官方文档的如下说明知道：

`interfaces()`

Return a list of interfaces incident on this node/router. Each item in the list is an Interface object, each of which includes name, ethaddr, ipaddr, and netmask attributes.

[50]

②第25-27行：实现 IPv4 Router 对 ARP Request 的响应逻辑。当接收到一个报文时，首先分解出其 packet，而后通过方法 get\_header(Arp) 来获取 Arp 头，而同样阅读官方文档可知，get\_header(Arp) 会返回获取到的首个 Arp 头，如果没有就返回 None，而通过此我们就可以判断接收到的包是否为一个 Arp Request 了，如果是的话，那么显然 arp = packet.get\_header(Arp) 不为 None，否则其就不为一个 Arp 报文，直接丢弃。

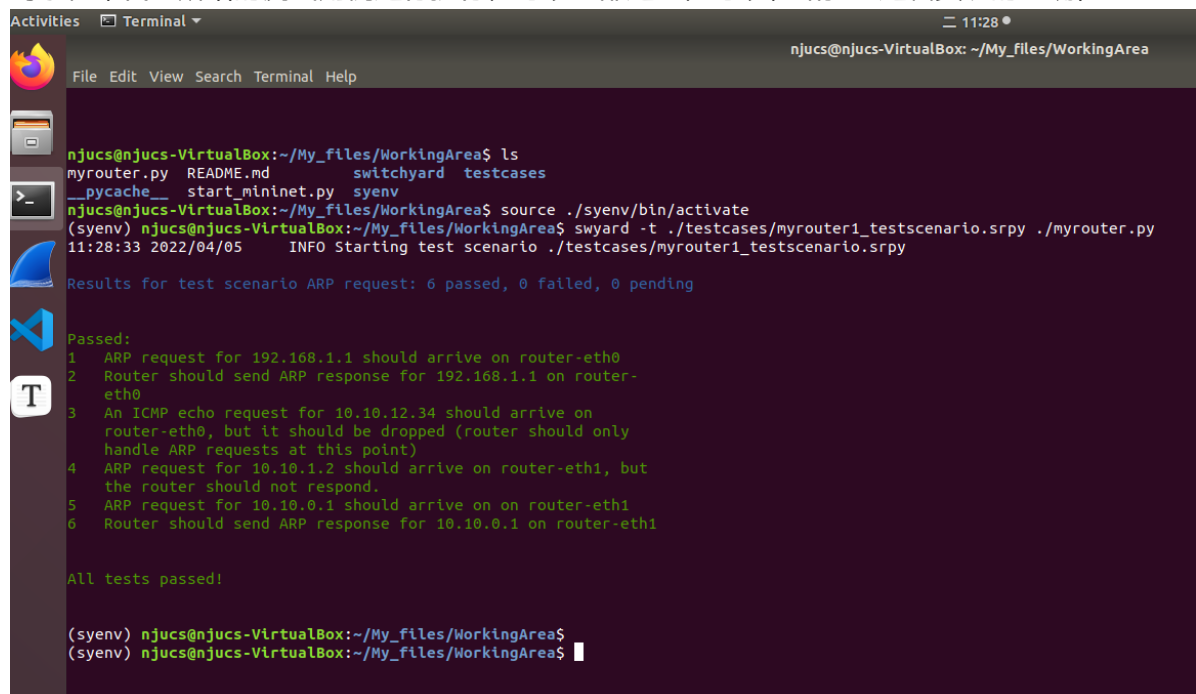
③第28-31行：遍历先前的端口列表，查询是否有端口的IP地址与 Arp 报文的目标IP地址 targetprotoaddr 相同，如果找到了响应的表项，则通过四个属性构造出 Arp Reply Packet：端口IP地址（作为源IP地址），端口MAC地址(作为源MAC地址，Arp Request 当中唯独空缺的字段，也是ARP报文所求取的字段)，Arp Request 发送方的IP地址（即 senderprotoaddr，作为目标IP地址），Arp Request 发送方的MAC地址（即 senderhwaddr，作为目标MAC地址）。然后将该包通过接收请求的端口（the same interface on which the ARP request arrived）向回发送出去即可。

A->B发送 Arp Request，想要求得B的MAC地址，此时A为源，而B为目标；  
B接收到了A的 Request；  
B->A发送 Arp Reply以回复A的 Request，此时B为源，而A为目标；

综上，就实现了 IPv4 Router 对于 Arp Request 的响应逻辑。

## 0x02 In the report, show the test result of your router. (Optional) If you have written the test files yourself, show how you test your ARP responding logic.

对于框架代码所给的测试用例进行执行，可以全部通过，可以粗略验证逻辑实现的正确性：



```
Activities  Terminal  11:28
njucs@njucs-VirtualBox: ~/My_files/WorkingArea

File Edit View Search Terminal Help

njucs@njucs-VirtualBox:~/My_files/WorkingArea$ ls
myrouter.py  README.md      switchyard  testcases
__pycache__  start_mininet.py  syenv
njucs@njucs-VirtualBox:~/My_files/WorkingArea$ source ./syenv/bin/activate
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$ swyard -t ./testcases/myrouter1_testscenario.srpy ./myrouter.py
11:28:33 2022/04/05 INFO Starting test scenario ./testcases/myrouter1_testscenario.srpy

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 router-eth1
6 Router should send ARP response for 10.10.0.1 on router-eth1

All tests passed!

(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

(可选)编写自己的测试用例，并展示测试用例逻辑以及结果：

自己编写的测试用例逻辑：

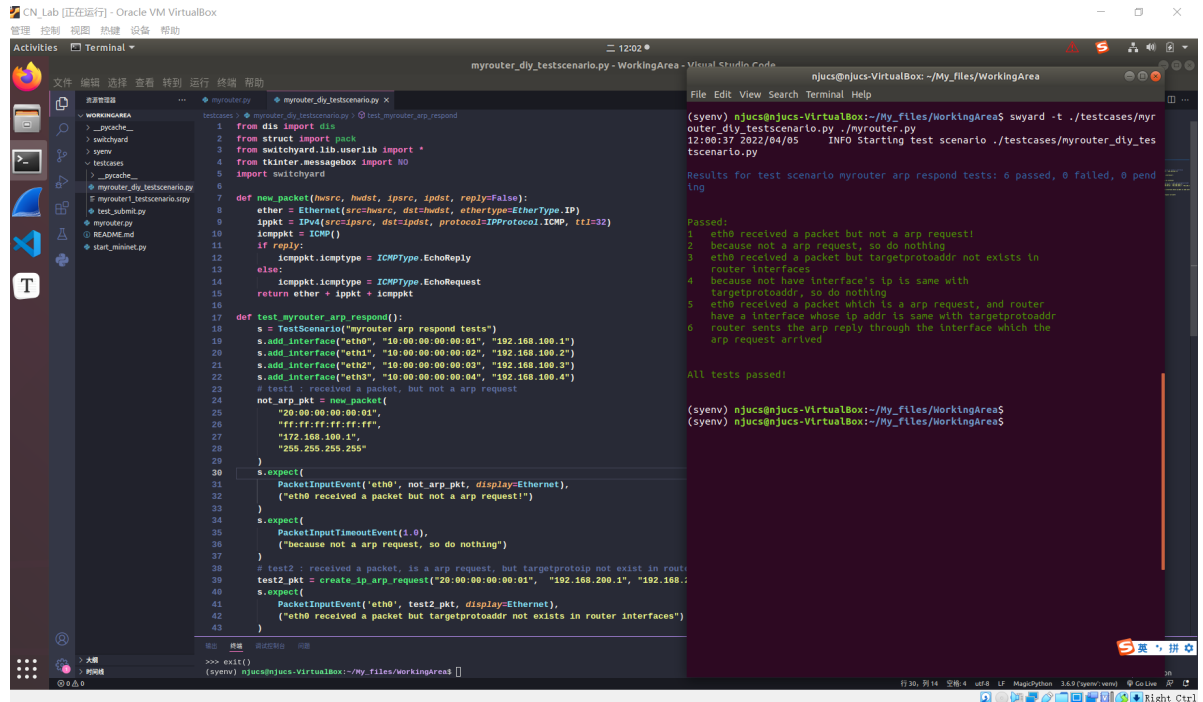
test case 01：接收到非ARP Request，此时期望所有的端口都不发包，什么也不做；

test case 02：接收到ARP Request，但是没有端口的IP地址与 targetprotoaddr 相同，此时同样期望什么也不做；

test case 03：接收到ARP Request，并且有端口的IP地址与 targetprotoaddr 相同，此时期望


从接收端口发出正确的ARP Reply；（同时检验其它端口行为，以及ARP Reply当中的内容正确性）

运行结果如下图所示：

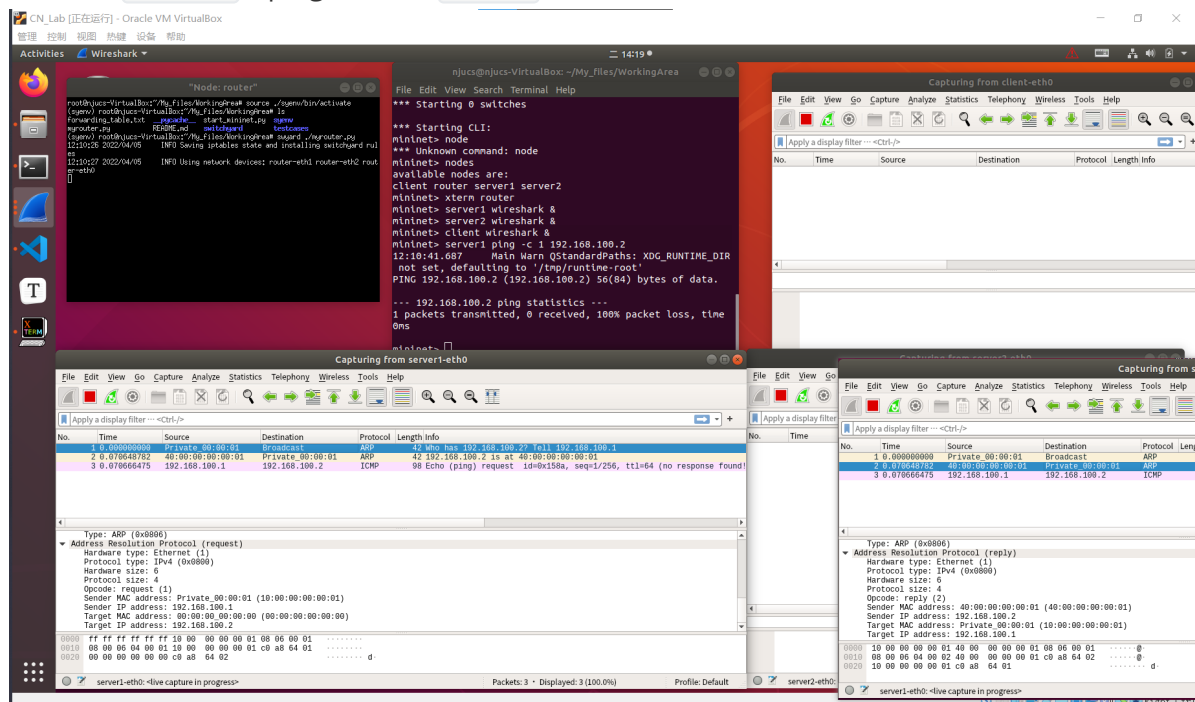


```
myrouter_dly_testscenario.py - WorkingArea - Visual Studio Code
myrouter_dly_testscenario.py
1 from dis import dis
2 from struct import pack
3 from switchyard.lib.userlib import *
4 from tkinter.messagebox import NO
5 import switchyard
6
7 def new_packet(hwsrc, hwdst, ipsrc, ipdst, reply=False):
8     ether = Ethernet(src=hwsrc, dst=hwdst, ethertype=EtherType.IP)
9     ippkt = IPv4(src=ipsrc, dst=ipdst, protocol=IPProtocol.ICMP, ttl=32)
10    icmppkt = ICMP()
11    if reply:
12        icmppkt.icmptype = ICMPType.EchoReply
13    else:
14        icmppkt.icmptype = ICMPType.EchoRequest
15    return ether + ippkt + icmppkt
16
17 def test_myrouter_arp_respond():
18     s = TestScenario("myrouter arp respond tests")
19     s.add_interface("eth0", "10:00:00:00:00:01", "192.168.100.1")
20     s.add_interface("eth1", "10:00:00:00:00:02", "192.168.100.2")
21     s.add_interface("eth2", "10:00:00:00:00:03", "192.168.100.3")
22     s.add_interface("eth3", "10:00:00:00:00:04", "192.168.100.4")
23     s.test1 = receive_a_packet, but not a arp request
24     not_arp_pkt = new_packet(
25         "20:00:00:00:00:01",
26         "ff:ff:ff:ff:ff:ff",
27         "192.168.100.1",
28         "255.255.255.255"
29     )
30     s.expect(
31         PacketInputEvent("eth0", not_arp_pkt, display=Ethernet),
32         ("eth0 received a packet but not a arp request!")
33     )
34     s.expect(
35         PacketInputTimeoutEvent(1.0),
36         ("because not a arp request, so do nothing")
37     )
38     # test2: received a packet, is a arp request, but targetprotoaddr not exist in router interfaces
39     test2_pkt = create_ip_request("20:00:00:00:00:01", "192.168.200.1", "192.168.100.1")
40     s.expect(
41         PacketInputEvent("eth0", test2_pkt, display=Ethernet),
42         ("eth0 received a packet but targetprotoaddr not exists in router interfaces")
43     )
44
45 if __name__ == '__main__':
46     test_myrouter_arp_respond()
47
48 >>> exit()
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

测试用例也是根据实现逻辑当中的分支所在进行测试，一共有三个点可以进行测试，从而编写了三个针对不同方面的测试用例。可以看到通过了所有自定义测试用例，因而可以进一步确定实现逻辑的正确性。

**0x03**  Your task is: ping the router from another host (server1 or server2). Using Wireshark to prove that you have handled ARP requests well. Write the procedure and analysis in your report with screenshots.

这里使用 `server1` 来ping所实现的 `router`，效果如下：



可以见到左下角为从 `server1` 向 `router` 的连接 `server1` 端口发送的ARP Request，而右下角为 `router` 连接 `server1` 端口发送回的ARP Reply。通过对比两个数据包当中的内容，可以发现：ARP Request进行广播，其源MAC地址和IP地址都为 `server1` 的MAC地址和IP地址，而目标MAC地址则为 `192.168.100.2` 即 `router` 连接 `server1` 的端口的IP地址，而目标MAC地址则为空。

实际上ARP Request请求就是进行泛洪，对于IP地址与ARP Request里目标IP地址相同的设备，其需要返回一个ARP Reply，其中包含了ARP Request所请求的MAC地址。（该ARP Reply可以通过右下角抓包数据查看到）

当具有目标IP地址的设备接收到ARP Request时，其会将自己的IP地址作为 `senderprotoaddr`，将自己的MAC地址作为 `senderhwaddr`，而将ARP Request当中的源MAC地址作为目标MAC地址，源IP地址作为目标IP地址，封装成一个ARP Reply，向接收ARP Request的端口发出。

另外，在这个过程当中，其它路由器端口和主机都未有动作。

从而我们可以初步验证本实验中对交换机向ARP Request响应逻辑实现的正确性。

## 0x04 In your report, show how you construct the ARP table.

实现逻辑如图所示：

```
13 class Router(object):
14     def __init__(self, net: switchyard.llnetbase.LLNetBase):
15         self.net = net
16         self.interfaces = self.net.interfaces()
17         self.arp_table = {}
18
19     def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
20         timestamp, ifaceName, packet = recv
21         # TODO: your logic here
22         for key in list(self.arp_table.keys()):
23             if(time.time() >= self.arp_table[key][1]):
24                 self.arp_table.pop(key)
25         arp = packet.get_header(Arp)
26         if arp:
27             self.arp_table[arp.senderprotoaddr] = [arp.senderhwaddr, time.time() + 1200] # TTL: current time + 20 minutes
28             for it in self.interfaces:
29                 if it.ipaddr == arp.targetprotoaddr:
30                     reply_pkt = create_ip_arp_reply(it.ethaddr, arp.senderhwaddr, it.ipaddr, arp.senderprotoaddr)
31                     self.net.send_packet(ifaceName, reply_pkt)
```

①第17行：为对象增添一个字典类型的属性 `arp_table` 作为该路由器的ARP table，规定表项格式为：键：IP地址，值：列表【MAC地址，TTL（拓展部分）】；

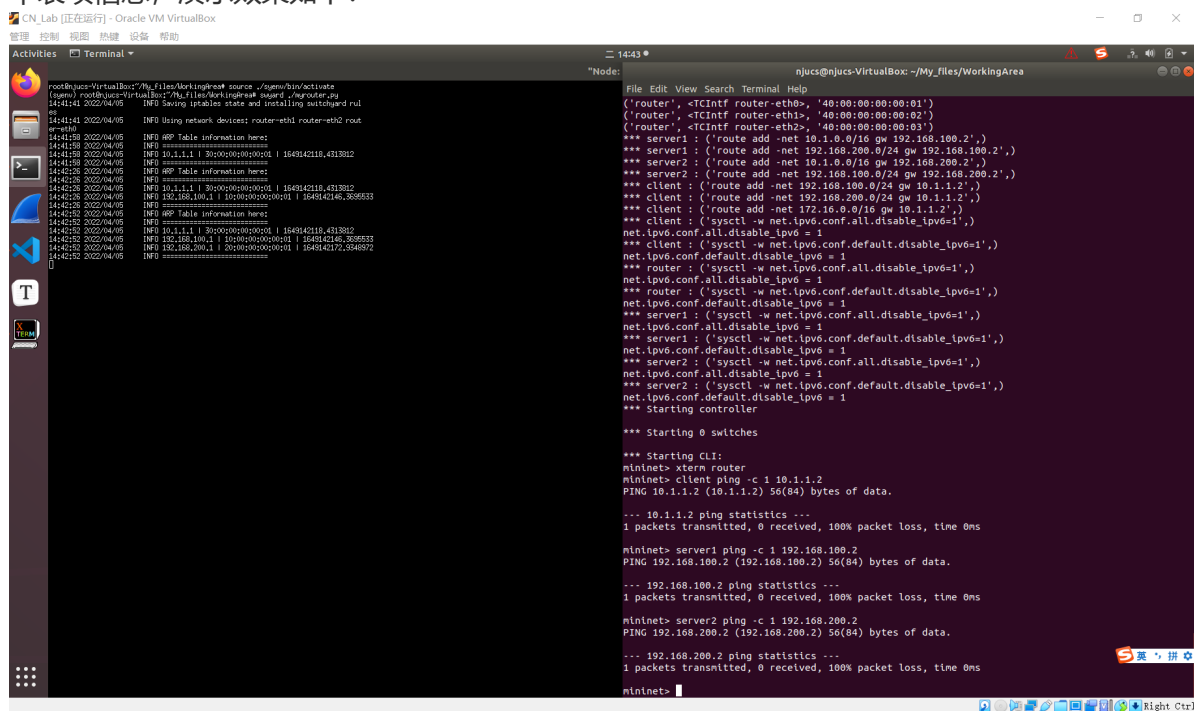
②第26行：当接收到ARP请求时，如果有同名IP则修改表项，如果没有则向 `arp_table` 中新增表项（字典操作正好与逻辑完美匹配），TTL设定为20分钟后；

③第22-24行：将已过TTL或者已经到达TTL的表项进行删除（拓展部分，增添超时机制）

具体行为就是，对于接收到的 `Arp Request`，根据其 `senderprotoaddr` 和 `senderhwaddr` 修改本路由器的 `arp_table`。

## 0x05 In your report, show the cached ARP table with screenshots. Explain how entries have changed.

利用 `mininet` 进行仿真测试，并且在源代码中嵌入 `log_info` 来进行信息输出以展示ARP Table中表项信息，演示效果如下：

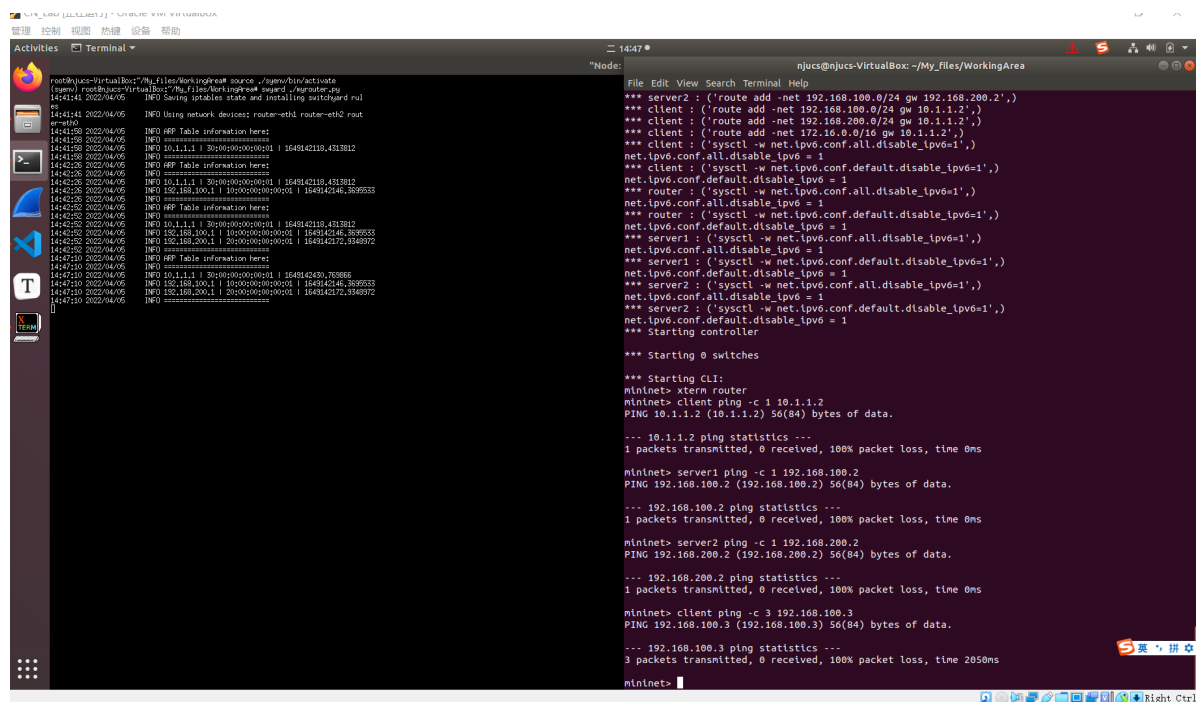


①首先利用 `client` 来Ping `router` 的一个端口，此时接收到来自新IP地址的ARP Request，ARP Table当中追加 `client` 的表项：IP、MAC、TTL；

②而后利用 `server1` 来Ping `router` 的一个端口，此时接收到来自新IP地址的ARP Request，ARP Table当中追加 `server1` 的表项：IP、MAC、TTL；

③最后利用 `server2` 来Ping `router` 的一个端口，此时接收到来自新IP地址的ARP Request，ARP Table当中追加 `server2` 的表项：IP、MAC、TTL；

最后利用 `client` 随意Ping一个IP地址，可以观察到ARP当中仍旧是符合要求的：



综上，可以基本验证在本实验中ARP Table行为逻辑实现的正确性。

## 5.核心代码

```
1 class Router(object):  
2     def __init__(self, net: switchyard.llnetbase.LLNetBase):  
3         self.net = net  
4         self.interfaces = self.net.interfaces()  
5         self.arp_table = {}  
  
6  
7     def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):  
8         timestamp, ifaceName, packet = recv  
9         # TODO: your logic here  
10        for key in list(self.arp_table.keys()):  
11            if(time.time() >= self.arp_table[key][1]):  
12                self.arp_table.pop(key)  
13            arp = packet.get_header(Arp)  
14            if arp:  
15                self.arp_table[arp.senderprotoaddr] = [arp.senderhwaddr,  
time.time() + 1200]    # TTL: current time + 20 minutes  
16                # show ARP Table information  
17                log_info(f"ARP Table information here:")  
18                log_info("=====  
19                for it in self.arp_table.keys():  
20                    log_info(f"{it} | {self.arp_table[it][0]} |  
{self.arp_table[it][1]}")  
21                log_info("=====")  
22                for it in self.interfaces:  
23                    if it.ipaddr == arp.targetprotoaddr:  
24                        reply_pkt = create_ip_arp_reply(it.ethaddr,  
arp.senderhwaddr, it.ipaddr, arp.senderprotoaddr)  
25                        self.net.send_packet(ifaceName, reply_pkt)
```



## 6.总结与感想

---

### 总结：

- 🧐对于ARP协议以及ARP Request \ ARP Reply有了进一步的理解；
- 🧐阅读查阅 `switchyard` 文档越发熟练，可以看到API的实现逻辑真的很有用！；
- 🧐三种测试方法：
  - ①利用 `switchyard` 编写 `testscenario` 自动测试；
  - ②使用 `mininet` 结合 `wireshark` 模拟测试；
  - ③使用 `mininet` 结合 `log_info` 模拟测试

### 感想：

- 🧐本次实验较短，无感想，略🤔