实验3-响应ARP

231880038 张国良

1. 实验名称

响应ARP

2. 实验目的

模拟响应 ARP（地址解析协议）请求，获取分配给路由器上接口的地址

3. 实验内容

Task 1: Preparation

- 下载Lab3模板代码

Task 2: Handle ARP Requests

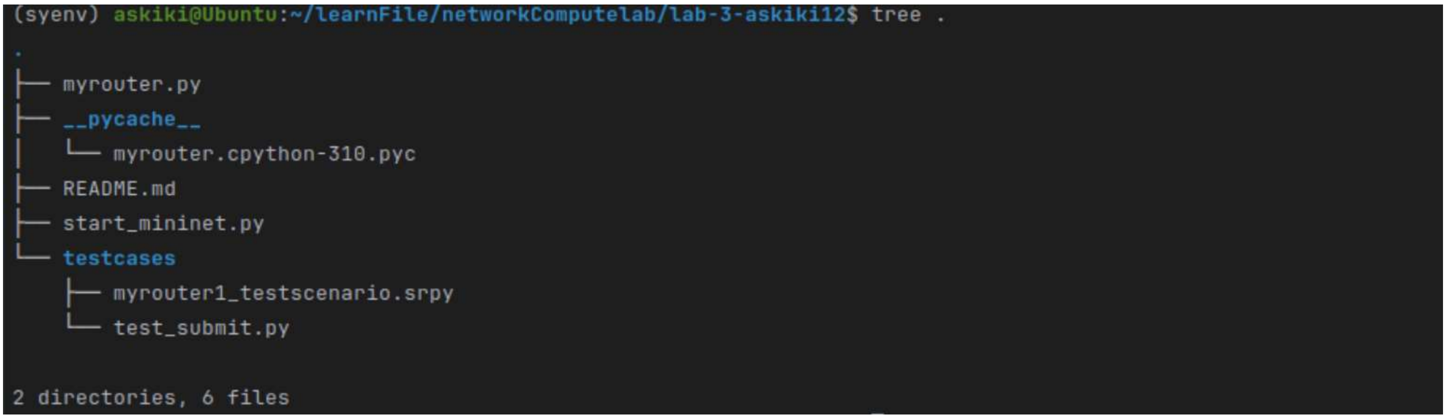
- 收到报文后，根据包头判断是否是ARP包，若不是则抛弃
- 对于每个 ARP 请求，应该确定 ARP 标头中的 `targetprotoaddr` 字段（IP 地址目标）是否是分配给路由器上接口之一的 IP 地址
- 如果目标 IP 地址是分配给路由器接口的地址，则应创建并发送适当的 ARP 回复。 （如果目标 IP 地址未分配给路由器的接口之一，则不应使用 ARP 回复进行响应）
- 如果在路由器中收到的数据包不是 ARP 请求，您应该暂时忽略它（丢弃它）
- `swyard -t testcases/myrouter1_testscenario.srpy myrouter.py` 测试代码
- 部署原理

Task 3: Cached ARP Table

- 修改 myrouter.py文件，添加cached table，更新时，打印 cached ARP table

4. 实验结果

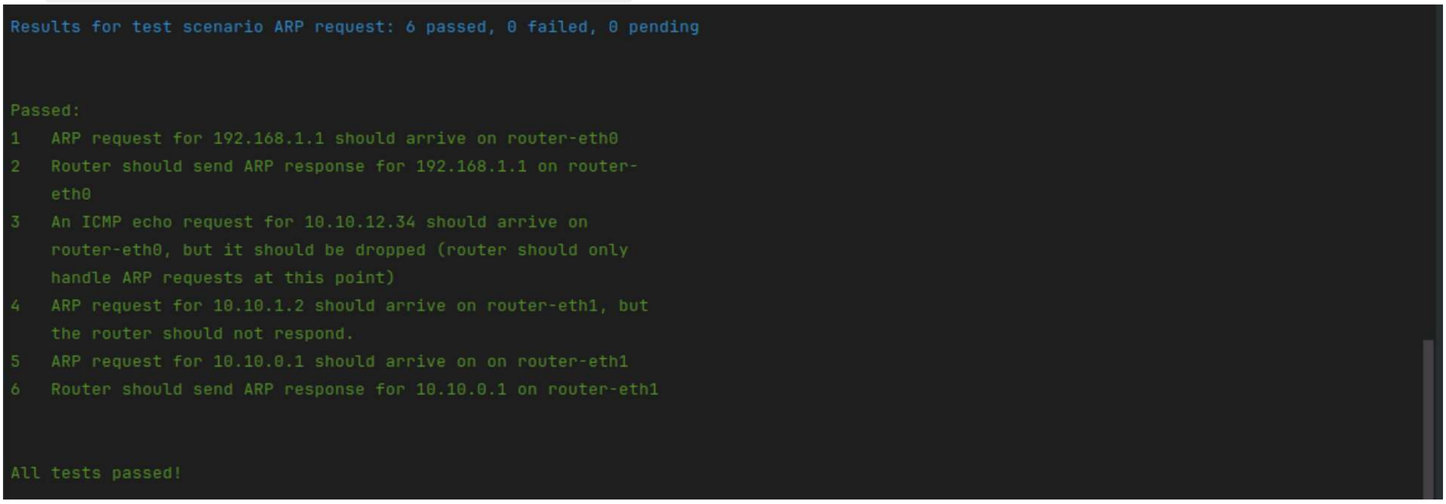
克隆实验目录：



```
(syenv) askiki@Ubuntu:~/learnFile/networkComputelab/lab-3-askiki12$ tree .
.
├── myrouter.py
├── __pycache__
│   └── myrouter.cpython-310.pyc
├── README.md
├── start_mininet.py
└── testcases
    ├── myrouter1_testscenario.srpy
    └── test_submit.py

2 directories, 6 files
```

输入 `swyard -t testcases/myrouter1_testscenario.srpy myrouter.py` 进行测试的结果：

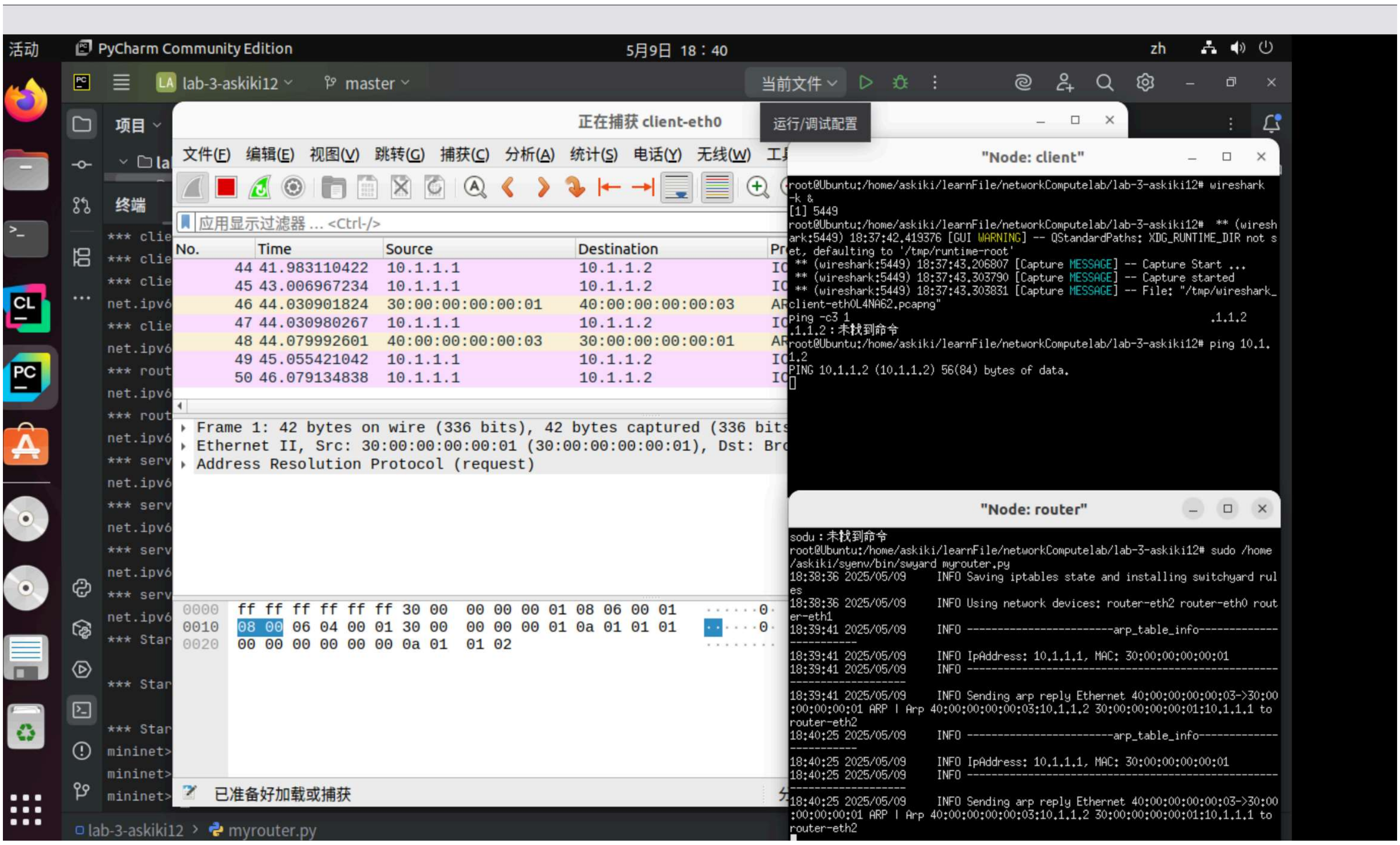


```
Results for test scenario ARP request: 6 passed, 0 failed, 0 pending


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!
```

部署后的结果：



活动
PyCharm Community Edition
5月9日 18：40
zh
lab-3-askiki12
master
当前文件
运行/调试配置
项目
终端
正在捕获 client-eth0
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工
应用显示过滤器 … <Ctrl-/>
已准备好加载或捕获
lab-3-askiki12 > myrouter.py

No.	Time	Source	Destination	Pr
44	41.983110422	10.1.1.1	10.1.1.2	IC
45	43.006967234	10.1.1.1	10.1.1.2	IC
46	44.030901824	30:00:00:00:00:01	40:00:00:00:00:03	AR
47	44.030980267	10.1.1.1	10.1.1.2	IC
48	44.079992601	40:00:00:00:00:03	30:00:00:00:00:01	AR
49	45.055421042	10.1.1.1	10.1.1.2	IC
50	46.079134838	10.1.1.1	10.1.1.2	IC

```
› Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits
› Ethernet II, Src: 30:00:00:00:00:01 (30:00:00:00:00:01), Dst: Bro
› Address Resolution Protocol (request)
```

```
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                     ········
```

```
*** clie
*** clie
*** clie
net.ipv6
*** clie
net.ipv6
*** rout
net.ipv6
*** rout
net.ipv6
*** serv
net.ipv6
*** serv
net.ipv6
*** serv
net.ipv6
*** serv
net.ipv6
*** Star
*** Star
*** Star
mininet>
mininet>
mininet>
```

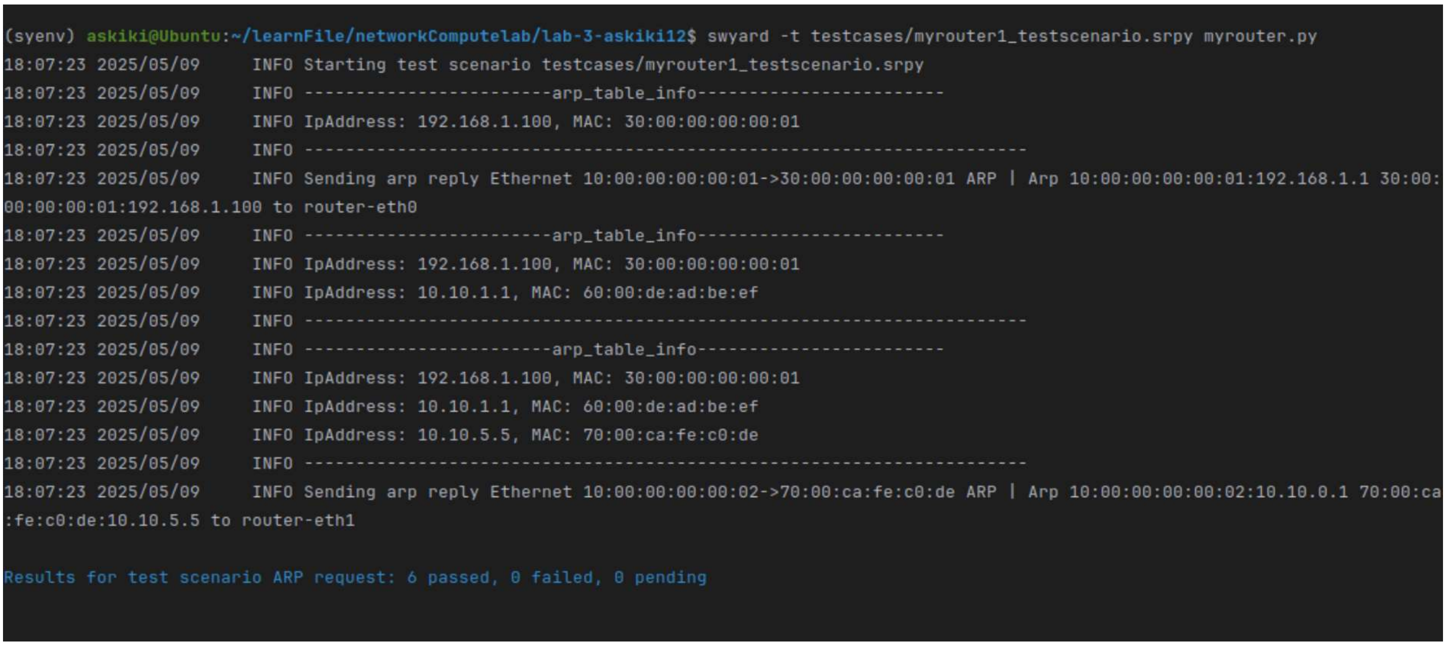
"Node: client"

```
root@Ubuntu:/home/askiki/learnFile/networkComputelab/lab-3-askiki12# wireshark
-k &
[1] 5449
root@Ubuntu:/home/askiki/learnFile/networkComputelab/lab-3-askiki12#  ** (wiresh
ark:5449) 18:37:42.419376 [GUI WARNING] -- QStandardPaths: XDG_RUNTIME_DIR not s
et, defaulting to '/tmp/runtime-root'
 ** (wireshark:5449) 18:37:43.206807 [Capture MESSAGE] -- Capture Start ...
 ** (wireshark:5449) 18:37:43.303790 [Capture MESSAGE] -- Capture started
 ** (wireshark:5449) 18:37:43.303831 [Capture MESSAGE] -- File: "/tmp/wireshark_
client-eth0L4NA62.pcapng"
ping -c3 1                                                             .1.1.2
.1.1.2：未找到命令
root@Ubuntu:/home/askiki/learnFile/networkComputelab/lab-3-askiki12# ping 10.1.
1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
```

"Node: router"

```
sodu：未找到命令
root@Ubuntu:/home/askiki/learnFile/networkComputelab/lab-3-askiki12# sudo /home
/askiki/syenv/bin/swyard myrouter.py
18:38:36 2025/05/09     INFO Saving iptables state and installing switchyard rul
es
18:38:36 2025/05/09     INFO Using network devices: router-eth2 router-eth0 rout
er-eth1
18:39:41 2025/05/09     INFO -----------------------arp_table_info------------
-----------
18:39:41 2025/05/09     INFO IpAddress: 10.1.1.1, MAC: 30:00:00:00:00:01
18:39:41 2025/05/09     INFO -------------------------------------------------
------------------
18:39:41 2025/05/09     INFO Sending arp reply 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 to
router-eth2
18:40:25 2025/05/09     INFO -----------------------arp_table_info------------
-----------
18:40:25 2025/05/09     INFO IpAddress: 10.1.1.1, MAC: 30:00:00:00:00:01
18:40:25 2025/05/09     INFO -------------------------------------------------
------------------
18:40:25 2025/05/09     INFO Sending arp reply 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 to
router-eth2
```

打印出的arp表：



```
(syenv) askiki@Ubuntu:~/learnFile/networkComputelab/lab-3-askiki12$ swyard -t testcases/myrouter1_testscenario.srpy myrouter.py
18:07:23 2025/05/09     INFO Starting test scenario testcases/myrouter1_testscenario.srpy
18:07:23 2025/05/09     INFO -----------------------arp_table_info-----------------------
18:07:23 2025/05/09     INFO IpAddress: 192.168.1.100, MAC: 30:00:00:00:00:01
18:07:23 2025/05/09     INFO -------------------------------------------------------------------
18:07:23 2025/05/09     INFO Sending arp reply 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 to router-eth0
18:07:23 2025/05/09     INFO -----------------------arp_table_info-----------------------
18:07:23 2025/05/09     INFO IpAddress: 192.168.1.100, MAC: 30:00:00:00:00:01
18:07:23 2025/05/09     INFO IpAddress: 10.10.1.1, MAC: 60:00:de:ad:be:ef
18:07:23 2025/05/09     INFO -------------------------------------------------------------------
18:07:23 2025/05/09     INFO -----------------------arp_table_info-----------------------
18:07:23 2025/05/09     INFO IpAddress: 192.168.1.100, MAC: 30:00:00:00:00:01
18:07:23 2025/05/09     INFO IpAddress: 10.10.1.1, MAC: 60:00:de:ad:be:ef
18:07:23 2025/05/09     INFO IpAddress: 10.10.5.5, MAC: 70:00:ca:fe:c0:de
18:07:23 2025/05/09     INFO -------------------------------------------------------------------
18:07:23 2025/05/09     INFO Sending arp reply 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 to router-eth1

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

5. 核心代码

```python
#!/usr/bin/env python3

'''
Basic IPv4 router (static routing) in Python.
'''

import time
import switchyard
from switchyard.lib.userlib import *


class Router(object):
    def __init__(self, net: switchyard.llnetbase.LLNetBase):
        self.net = net
        # other initialization stuff here
        self.interfaces = net.interfaces()
        self.arp_table = {}
        self.ip_list = []
        self.eth_list = []
        for i in self.interfaces:
            self.ip_list.append(i.ipaddr)
            self.eth_list.append(i.ethaddr)
        self.arp_timeout = 20*60

    def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
        timestamp, ifaceName, packet = recv
        # TODO: your logic here
        arp = packet.get_header(Arp)
        ipv4 = packet.get_header(IPv4)
        input_port = self.net.interface_by_name(ifaceName)
        if arp is not None:
            self.update_arp_table()
            self.arp_table[arp.senderprotoaddr] = [arp.senderhwaddr, time.time()]
            log_info("------------------------arp_table_info------------------------")
            for k, v in self.arp_table.items():
                log_info(f"IpAddress: {k}, MAC: {v[0]}")
            log_info("----------------------------------------------------------------------")
            if arp.operation == ArpOperation.Request:
                for i in self.ip_list:
                    if i == arp.targetprotoaddr:
                        arp_reply_pkt = create_ip_arp_reply(input_port.ethaddr, arp.senderhwaddr, arp.targetprotoaddr,
                                                            arp.senderprotoaddr)
                        self.net.send_packet(ifaceName, arp_reply_pkt)
                        log_info(f"Sending arp reply {arp_reply_pkt} to {ifaceName}")
                        return

    def update_arp_table(self):
        current_time = time.time()
        for ip in list(self.arp_table.keys()):
            mac, last_update_time = self.arp_table[ip]
            if current_time - last_update_time > self.arp_timeout:
                del self.arp_table[ip]

    def start(self):
        '''A running daemon of the router.
        Receive packets until the end of time.
        '''
        while True:
            try:
                recv = self.net.recv_packet(timeout=1.0)
            except NoPackets:
                continue
            except Shutdown:
                break

            self.handle_packet(recv)

        self.stop()
    def stop(self):
        self.net.shutdown()


def main(net):
    '''
    Main entry point for router.  Just create Router
    object and get it going.
    '''
    router = Router(net)
    router.start()
```

根据要求判断是否是ARP包，若不是则抛弃，并确定 ARP 标头中的 `targetprotoaddr` 字段（IP 地址目标）是否是分配给路由器上接口之一的 IP 地址，是则回复不是则暂时忽略，同时每次收到ARP请求，更新ARP表，并打印ARP表，记录时间戳，ARP过期时间设置为20分钟。

6. 实验总结

本次实验我模拟了路由器中ARP的收发逻辑和表的更新逻辑，加强了我对局域网通信转发的理解