

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/李文中 助教：

学院	电子科学与工程学院	专业（方向）	电子信息科学与技术
学号	191180102	姓名	任晟昊
Email	191180102@smail.nju.edu.cn	开始/完成日期	2022.04.05

实验名称

Lab3

实验目的

实现路由器的基本功能

实验内容

进行对ARP的应答和缓存表的实现

实验结果

Task2

在实验二中午需要完成对arp的应答，所以构建了如下数据结构：

```
def __init__(self, net: switchyard.llnetbase.LLNetBase):
    self.net = net
    # other initialization stuff here
    self.ipList = [intf.ipaddr for intf in net.interfaces()]
    self.macList = [intf.ethaddr for intf in net.interfaces()]
```

具体的处理逻辑如下：

首先，用 `arp = packet.get_header(Arp)` 获取包头，判断arp头是否为空，在这里我开始使用 `if arp == None` 进行判断，但是发现一直出错，后来经过查阅资料，发现

▲ The answer is explained [here](#).

400 To quote:

▼
✓ A class is free to implement comparison any way it chooses, and it can choose to make comparison against None mean something (which actually makes sense; if someone told you to implement the None object from scratch, how else would you get it to compare True against itself?).
🔄

Practically-speaking, there is not much difference since custom comparison operators are rare. But you should use as a general rule. `is None`

由于我们这里的arp返回的是一个类，所以不应该在数值上比较他们，而是应该在类型上比较两者，故应该使用 `if arp is None:`。接着判断arp的类型是 `request` or `reply` 通过查阅API可见，

Request = 1

Reply = 2

所以，对应Request查找他的target是否存在路由器的缓存表中，如果是则返回arp需要的信息，如果不是则丢弃。对应reply则将其返回的target的ip和mac地址加入到路由器表中。

最后如果两种类型都不是，则丢弃。

```
# TODO: your logic here
log_info("Got a packet: {}".format(str(packet)))
arp = packet.get_header(Arp)
if arp is None:
    log_info("It's not an arp packet!")
else:
    log_info("Received an arp packet!")
    self.arpTable[arp.senderprotoaddr] = arp.senderhwaddr
    if arp.operation == 1:
        log_info("Received a request!")
        for i in range(len(self.ipList)):
            if self.ipList[i] == arp.targetprotoaddr:
                log_info("I got it!")
                answer =
create_ip_arp_reply(self.macList[i], arp.senderhwaddr, self.ipList[i], arp.senderpr
otoaddr)

                self.net.send_packet(ifaceName, answer)
                break
    else:
        if arp.operation == 2:
            log_info("Received a reply!")
            self.arpTable[arp.targetprotoaddr] = arp.targethwaddr
        else:
            log_info("Received an arp which is not a request or a reply!")
```

测试结果如下:

```
(syenv) njucs@njucs-VirtualBox:~/switchyard/lab-03-wjrzms$ swyard -t testcases/my
router1_testscenario.srpy myrouter.py
15:10:26 2022/04/05      INFO Starting test scenario testcases/myrouter1_testscen
ario.srpy
15:10:26 2022/04/05      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
15:10:26 2022/04/05      INFO Received an arp packet!
15:10:26 2022/04/05      INFO Received a request!
15:10:26 2022/04/05      INFO I got it!
15:10:26 2022/04/05      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 d
ata bytes)
15:10:26 2022/04/05      INFO It's not an arp packet!
15:10:26 2022/04/05      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
15:10:26 2022/04/05      INFO Received an arp packet!
15:10:26 2022/04/05      INFO Received a request!
15:10:26 2022/04/05      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
15:10:26 2022/04/05      INFO Received an arp packet!
15:10:26 2022/04/05      INFO Received a request!
15:10:26 2022/04/05      INFO I got it!
```

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!

部署至Mininet

在server1中ping -c3 192.168.200.1, 可以看到server1中的抓包如下:

The image shows a Wireshark packet capture window titled "Capturing from server1-eth0". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar with various icons for file operations, capture control, and analysis. A display filter bar shows "Apply a display filter ... <Ctrl-/>".

The packet list pane displays five captured packets:

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	0.066874749	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	0.066883419	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x151f, seq=1/256, ttl=64 (no
4	1.032871807	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x151f, seq=2/512, ttl=64 (no
5	2.056857023	192.168.100.1	192.168.200.1	ICMP	98	Echo (ping) request id=0x151f, seq=3/768, ttl=64 (no

The packet details pane for the selected packet (Frame 2) shows:

- Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
- Ethernet II, Src: 40:00:00:00:00:01 (40:00:00:00:00:01), Dst: Private_00:00:01 (10:00:00:00:00:01)
- Address Resolution Protocol (reply)
 - Hardware type: Ethernet (1)
 - Protocol type: IPv4 (0x0800)
 - Hardware size: 6
 - Protocol size: 4
 - Opcode: reply (2)
 - Sender MAC address: 40:00:00:00:00:01 (40:00:00:00:00:01)
 - Sender IP address: 192.168.100.2
 - Target MAC address: Private_00:00:01 (10:00:00:00:00:01)
 - Target IP address: 192.168.100.1

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  10 00 00 00 00 01 40 00 00 00 00 01 08 06 00 01  ....@.....
0010  08 00 06 04 00 02 40 00 00 00 00 01 c0 a8 64 02  ....@.....d.
0020  10 00 00 00 00 01 c0 a8 64 01  ....d.
```

The status bar at the bottom indicates "Ready to load or capture", "Packets: 5 · Displayed: 5 (100.0%)", and "Profile: Default".

```
(syenv) root@njucs-VirtualBox:~/switchyard/lab-03-wjrz# swyard myrouter.py
15:46:29 2022/04/05      INFO Saving iptables state and installing switchyard rules
15:46:29 2022/04/05      INFO Using network devices: router-eth2 router-eth0 router-eth1
15:47:13 2022/04/05      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
15:47:13 2022/04/05      INFO Received an arp packet!
15:47:13 2022/04/05      INFO Received a request!
15:47:13 2022/04/05      INFO I got it!
15:47:13 2022/04/05      INFO Sent an answer: Ethernet 40:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 40:00:00:00:00:01:192.168.100.2 10:00:00:00:00:01:192.168.100.1
15:47:13 2022/04/05      INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 ICMP | ICMP EchoRequest 5407 1 (56 data bytes)
15:47:13 2022/04/05      INFO It's not an arp packet!
15:47:14 2022/04/05      INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 ICMP | ICMP EchoRequest 5407 2 (56 data bytes)
15:47:14 2022/04/05      INFO It's not an arp packet!
15:47:15 2022/04/05      INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 ICMP | ICMP EchoRequest 5407 3 (56 data bytes)
15:47:15 2022/04/05      INFO It's not an arp packet!
```

可以看到路由器对这个arp包进行了request的回复，并进行处理，同时进行arp的回复，而ICMP包不在处理范围之内，不做处理。

Task3

加入 `self.arpTable = {}`，当收到一个arp包时，`self.arpTable[arp.senderprotoaddr] = arp.senderhwaddr`，当判断该包为reply时，`self.arpTable[arp.targetprotoaddr] = arp.targethwaddr`。

在处理一个新包时，加入打印信息：

```
for (k,v) in self.arpTable.items():
    print("%s \t" % k,v)
```

```

16:11:33 2022/04/05      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
16:11:33 2022/04/05      INFO Received an arp packet!
16:11:33 2022/04/05      INFO Received a request!
16:11:33 2022/04/05      INFO I got it!
16:11:33 2022/04/05      INFO Sent an answer: Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.1.1 30:00:00:00:00:01:192.168.1.1
00
16:11:33 2022/04/05      INFO Print ARP table:
192.168.1.100      30:00:00:00:00:01
16:11:33 2022/04/05      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 d
ata bytes)
16:11:33 2022/04/05      INFO It's not an arp packet!
16:11:33 2022/04/05      INFO Print ARP table:
192.168.1.100      30:00:00:00:00:01
16:11:33 2022/04/05      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
16:11:33 2022/04/05      INFO Received an arp packet!
16:11:33 2022/04/05      INFO Received a request!
16:11:33 2022/04/05      INFO Print ARP table:
192.168.1.100      30:00:00:00:00:01
10.10.1.1          60:00:de:ad:be:ef
16:11:33 2022/04/05      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
16:11:33 2022/04/05      INFO Received an arp packet!
16:11:33 2022/04/05      INFO Received a request!
16:11:33 2022/04/05      INFO I got it!
16:11:33 2022/04/05      INFO Sent an answer: Ethernet 10:00:00:00:00:02->70:00:c
a: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
16:11:33 2022/04/05      INFO Print ARP table:
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

```

在每一次接收到ARP包的时候，就会产生更新。