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	**计算机科学与技术系**	专业（方向）	**计算机科学与技术**
学号	**191220154**	姓名	**张涵之**
Email	**1683762615@qq.com**	开始/完成日期	**2021/4/18**

1. 实验名称：Lab 3: Respond to ARP

2. 实验目的：

Construct ARP reply and cache ARP table (upon receiving ARP request)

3. 实验内容

a) Task 2: Handle ARP Requests
Ready to make ARP work.

b) Task 3: Cached ARP Table
Maintain correlation between MAC address and corresponding IP address.

4. 实验结果

a) Task 2: Handle ARP Requests
Running the given switchyard test scenario.

```
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!
```

Testing my router in Mininet:
Ping the router from another host (server1).

```
root@njucs-VirtualBox:~/switchyard/lab-3-RainTreeCrow# ping -c3 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.

--- 10.1.1.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2031ms
```

The router only replies to ARP request and drop all the other packets. 3 packets (ICMP ping requests?) are transmitted but none of them are “received”, server1 assume they are not received because the router does not reply.

```
15:55:12 2021/04/18     INFO Receive an ARP request.
-----------------------------------
IP              Mac Address
10.1.1.1        30:00:00:00:00:01
-----------------------------------
15:55:12 2021/04/18     INFO Not an ARP packet, dropped.
15:55:13 2021/04/18     INFO Not an ARP packet, dropped.
15:55:14 2021/04/18     INFO Not an ARP packet, dropped.
15:55:33 2021/04/18     INFO Receive an ARP request.
-----------------------------------
IP              Mac Address
10.1.1.1        30:00:00:00:00:01
192.168.100.1   10:00:00:00:00:01
-----------------------------------
15:55:34 2021/04/18     INFO Not an ARP packet, dropped.
15:55:34 2021/04/18     INFO Not an ARP packet, dropped.
15:55:35 2021/04/18     INFO Not an ARP packet, dropped.
```

Here is what the router's Mininet windows shows after it is pinged by the client and server1, each time the router receives an ARP request, an entry is added to the cache in Task 3, and it replies accordingly. The following 3 packets are not ARP packets and are dropped without further management.

Source	Destination	Protocol	Length	Info
Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
40:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 40:00:00:00:00:01
192.168.100.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x17d0, seq=1/256, ttl=64 (no response found!)
192.168.100.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x17d0, seq=2/512, ttl=64 (no response found!)
192.168.100.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x17d0, seq=3/768, ttl=64 (no response found!)

Here are the packets captured by Wireshark, a Broadcast ARP, one private ARP and three ICMP Echo (ping) requests with "no response found".

b) Task 3: Cached ARP Table

The cache table with switchyard tests are presented below

```
(syenv) njucs@njucs-VirtualBox:~/switchyard/lab-3-RainTreeCrow$ swyard -t testcases/myrout
16:16:04 2021/04/18     INFO Starting test scenario testcases/myrouter1_testscenario.srpy
16:16:04 2021/04/18     INFO Receive an ARP request.
-----------------------------------
IP              Mac Address
192.168.1.100   30:00:00:00:00:01
-----------------------------------
16:16:04 2021/04/18     INFO Not an ARP packet, dropped.
16:16:04 2021/04/18     INFO Receive an ARP request.
-----------------------------------
IP              Mac Address
192.168.1.100   30:00:00:00:00:01
10.10.1.1       60:00:de:ad:be:ef
-----------------------------------
16:16:04 2021/04/18     INFO Receive an ARP request.
-----------------------------------
IP              Mac Address
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
-----------------------------------
```

From the log info I can infer that the router received and processed three ARP requests and dropped one none ARP packet, which matches the tests.

5. 核心代码

a) Once you understood the logic, the code is very simple

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

The ARP cache table is designed using a dictionary.

```
    def print_arpTable(self):
        print("---------------------------------")
        print("IP\t\t Mac Address")
        for ip, mac in self.arpTable.items():
            print(ip, "\t", mac)
        print("---------------------------------")
```

Here is how the cache table is printed.

```
def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
    _, ifaceName, packet = recv
    # TODO: your logic here
    arp = packet.get_header(Arp)
    if arp is not None: #ARP packet
        #Update cached ARP table
        self.arpTable[arp.senderprotoaddr] = arp.senderhwaddr
        if arp.operation == 1: #Request
            log_info("Receive an ARP request.")
            targetInterface = None
            targetProto = arp.targetprotoaddr
            #Serching for the target IP address
            for interface in self.interfaces:
                if interface.ipaddr == targetProto:
                    targetInterface = interface
                    break
            if targetInterface is not None: #Found
                #Create ARP reply
                arpReply = create_ip_arp_reply(targetInterface.ethaddr,
                arp.senderhwaddr, targetInterface.ipaddr, arp.senderprotoaddr)
                #Sent out the same interface the ARP request arrived on
                self.net.send_packet(ifaceName, arpReply)
```

Here is how the ARP packet is handled. The sender's IP and Mac addresses are added to the cache table. If the target protocol address can be found among the router's interfaces' IP address, the corresponding interface is used to create the ARP reply packet. The packet is then passed back to the sender, using the very same interface on which the ARP request arrived.

```
    elif arp.operation == 2: #Reply
        log_info("Receive an ARP reply.")
        #Nothing to do yet
    else: #Others
        log_info("Unknown ARP format.")
    #Print ARP table every time it is updated
    self.print_arpTable()
else: #None ARP packet
    log_info("Not an ARP packet, dropped.")
```

Certain other cases (ARP reply and non-ARP packets) are not dealt with in this lab. Log information are used to distinguish such cases, and the ARP table is printed every time the router receives and handles an ARP packet.

6. 总结与感想

a) When I used Wireshark to capture files in the first two labs, I always wondered what the packets meant, what those ARP and ICMP packets, those requests and replies stood for. Through the lectures and labs, I gradually begin to understand their logic and functions, and how the entire computer network operates.

b) Dictionaries can be useful if used correctly.