

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	专业（方向）	计算机科学与技术
学号	201220062	姓名	黄子睿
Email	201220062@smail.nju.edu.cn	开始/完成日期	3/26-3/28

1. 实验名称：

Lab3 Respond to ARP

2. 实验目的：

学习理解 ARP 协议的内容以及 router 处理 ARP REQUEST 的方式与具体过程。

3. 实验内容：

本次实验需要解决 router 如何处理 ARP 协议，即需要处理 ARP request。Arp

协议报文格式的结构如下所示：

硬件类型		协议类型
硬件长度	协议长度	操作码（请求为1，响应为2）
源硬件地址		
源逻辑地址		
目的硬件地址		
目的逻辑地址		

在接收到报文后，首先判断它是否是 ARP 报文，即它是否拥有 ARP 头。

在确定它是 ARP 报文后，再判断它的操作类型(arp.operation)，具体是请求还是回复。

如果它是 ARP Request，则进行如下操作：

首先判断 arp.targetprotoaddr 是否是 router 自身某个接口的 ip 地址。如果是，则将该接口的 ipaddr 与 etheraddr 封装进 ARP Reply 中，从接收 request 的接口发送回去。

如果不是，则需要查询它是否与 router 的某个接口相连(在代码模拟中，具体指调用 interface_by_ipaddr 函数)。如果该目标地址与 router 不相连，则丢弃该包，并向 LOG 输出报错信息(在程序模拟中指用 try-except 捕捉 KeyError)。若相连，则构造 ARP Request 报文从对应接口向目标 ip 地址发送。

此外，在 router 中需要构造记录 ipaddr 与 etheraddr 对应关系的 Cache。Cache 有两种记录方式，一种是静态的(static NAT)，ip 地址与以太网地址的映射关系始终记录在表中；另一种是动态的(Dynammmic NAT)，router 维护一个已注册 ip 地址池（a pool of registered IP addresses），表中的信息可以更新并且一段时间未访问的信息将被移出 Cache。

设计一个可以同时实现两种功能的 ARPCache 类：

```
1. class ARPCache(object):
2.     def __init__(self, timeout = 180):
3.         self.timeout = timeout
4.         self.cache = {} # Ipaddr:[hwaddr,timestamp]
5.             # If timestamp is negative, the entry is static
6.             # Otherwise it's dynamic.
7.
8.     def put(self, key, value, timestamp = -1):
9.         # The default entry type is static
10.        log_info(f"The cache puts in entry {key} -- {value}")
```

```

11.         self.cache[key] = [value,timestamp]
12.
13.     def get(self, key):
14.         if key in self.cache:
15.             if self.cache[key][1] != -1:
16.                 self.cache[key][1] = time.time()
17.                 return self.cache[key][0]
18.         else:
19.             return None
20.
21.     def refresh(): # Evict those dynamic entries that are out of time
22.         cur_time = time.time()
23.         for key in list(self.cache):
24.             if cur_time - self.cache[key][1] >= self.timeout:
25.                 # Output to the log
26.                 log_info(f"The entry \"{key}:{self.cache[key]}\" is e
victed At time {cur_time}")
27.                 del self.cache[key]

```

在本次实验中，只需要用到其中的静态表项。

结合 ARPCache，实现 router 处理 ARP Request 的功能。

首先，将 router 自身所有接口的 ipaddr 与 etheraddr 信息在初始化时记录入表中，即修改__init__函数为：

```

1. def __init__(self, net: switchyard.llnetbase.LLNetBase):
2.     self.net = net
3.     self.arpCache = ARPCache()
4.     # Record own ip/ethernet addr
5.     for intf in self.net.interfaces():
6.         self.arpCache.put(intf.ipaddr, intf.ethaddr)

```

这样可以不用特判 targetprotoaddr 是否是 router 自身的接口地址，可以直接统一处理，简化了程序逻辑。

下面设计具体的 handle_packet 函数：

```

1. def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
2.     timestamp, ifaceName, packet = recv
3.     arp = packet.get_header(Arp, None)
4.     log_info(f"The arp is {arp}")
5.     if arp is not None: # Handle ARP packets
6.         # self-learning

```

```

7.          # Use arpCache as a static table, so no need to refresh.

8.          # put the sender ipaddr--etheraddr into cache
9.          self.arpCache.put(arp.senderprotoaddr, arp.senderhwaddr)

10.         # Forward ARP packet
11.         if arp.operation == ArpOperation.Request:
12.             # Search the cache to see whether the targethwaddr is
already in cache
13.             targethwaddr = self.arpCache.get(arp.targetprotoaddr)

14.             if targethwaddr == None: # The targethwaddr is not in
the cache
15.                 forwardIntf = None
16.                 try:
17.                     forwardIntf = self.net.interface_by_ipaddr(ar
p.targetprotoaddr)
18.                     # Get the forwarding interface.
19.                     pkt = create_ip_arp_request(arp.senderhwaddr,
arp.senderprotoaddr, arp.targetprotoaddr)
20.                     self.net.send_packet(forwardIntf, pkt)
21.                 except KeyError:
22.                     # The target ipaddr is not combined to any in
terface, just drop it.
23.                     log_info(f"The ipaddr {arp.targetprotoaddr} i
s not connected to the router.")
24.                     pass
25.                 else: # The targethwaddr is already in the cache
26.                     pkt = create_ip_arp_reply(targethwaddr, arp.sende
rhwaddr, arp.targetprotoaddr, arp.senderprotoaddr)
27.                     for forwardIntf in self.net.interfaces():
28.                         if forwardIntf.name == ifaceName:
29.                             self.net.send_packet(forwardIntf, pkt)
30.                             break
31.                 elif arp.operation == ArpOperation.Reply: # Handle ARP Re
ply, but drop them for now according to the document
32.                     pass
33.                 else: # Drop all kinds of other packets for now
34.                     pass

```

代码完成后，需要检验运行的结果：

首先在 xterm router 中运行 `swyard router.py`，可以看到初始化时 cache 填入

router 的接口信息：

```
"Node: router"
root@njucs-VirtualBox:~/lab-03-Huangzirui1206# source ~/switchyard/syenw/bin/activate
(syenv) root@njucs-VirtualBox:~/lab-03-Huangzirui1206# swyard myrouter.py
10:26:53 2022/03/28 INFO Saving iptables state and installing switchyard rules
10:26:54 2022/03/28 INFO Using network devices: router-eth1 router-eth2 router-eth0
10:26:54 2022/03/28 INFO The cache puts in entry 192.168.200.2:40:00:00:00:00:02
10:26:54 2022/03/28 INFO The cache puts in entry 10.1.1.2:40:00:00:00:00:03
10:26:54 2022/03/28 INFO The cache puts in entry 192.168.100.2:40:00:00:00:00:01
```

然后执行 client ping -c3 10.1.1.2，注意到 10.1.1.2 是 router 的一个接口，因此通过查询 cache，可以得到 10.1.1.2 的 etheraddr 信息。Wireshark 抓包后，观察 ARP Request 与 Arp Reply 的具体信息：

```
▶ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▶ Ethernet II, Src: 30:00:00:00:00:01 (30:00:00:00:00:01), 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: 30:00:00:00:00:01 (30:00:00:00:00:01)
  Sender IP address: 10.1.1.1
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.1.1.2
```

观察 Arp Request 的相关信息，可以看到此时 target mac address 是 0，意味着此时是一个广播报文，是需要告知的信息。

```
▶ Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▶ Ethernet II, Src: 40:00:00:00:00:03 (40:00:00:00:00:03), Dst: 30:00:00:00:00:01 (30:00:00:00:00:01)
▼ Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: 40:00:00:00:00:03 (40:00:00:00:00:03)
  Sender IP address: 10.1.1.2
  Target MAC address: 30:00:00:00:00:01 (30:00:00:00:00:01)
  Target IP address: 10.1.1.1
```

观察 Arp Reply 的信息，可以看到此时的发送者就是 arp 请求的目标，此时它的 mac 地址已经填上了。

而观察总的 arp 应答，可以看到非 ARP 的 ICMP 包此时被丢弃了：

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	30:00:00:00:00:01	Broadcast	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
2	0.052850219	40:00:00:00:00:03	30:00:00:00:00:01	ARP	42	10.1.1.2 is at 40:00:00:00:00:03
3	0.052859353	10.1.1.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x277b, seq=1/256, ttl=64 (no response found!)
4	1.018669682	10.1.1.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x277b, seq=2/512, ttl=64 (no response found!)
5	2.061112927	10.1.1.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x277b, seq=3/768, ttl=64 (no response found!)

而后通过 log 信息考察 cache 的运行情况。再执行 server1 ping -c1 client, log 信息中可以看到 server1 的地址信息将被新记录入 cache:

(The cache puts in entry ...)

```
"Node: router"
(syenv) root@njucs-VirtualBox:~/lab-03-Huangzirui1206# swyard myrouter.py
11:43:21 2022/03/28 INFO Saving iptables state and installing switchyard rules
11:43:22 2022/03/28 INFO Using network devices: router-eth0 router-eth2 router-eth1
11:43:22 2022/03/28 INFO The cache puts in entry 192.168.100.2 -- 40:00:00:00:00:01
11:43:22 2022/03/28 INFO The cache puts in entry 10.1.1.2 -- 40:00:00:00:00:03
11:43:22 2022/03/28 INFO The cache puts in entry 192.168.200.2 -- 40:00:00:00:00:02
11:43:36 2022/03/28 INFO The arp is Arp 30:00:00:00:00:01:10.1.1.1 00:00:00:00:00:00:10.1.1.2
11:43:36 2022/03/28 INFO The cache puts in entry 10.1.1.1 -- 30:00:00:00:00:01
11:43:36 2022/03/28 INFO The arp is None
11:43:37 2022/03/28 INFO The arp is None
11:43:38 2022/03/28 INFO The arp is None
11:44:54 2022/03/28 INFO The arp is Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.100.2
11:44:54 2022/03/28 INFO The cache puts in entry 192.168.100.1 -- 10:00:00:00:00:01
11:44:54 2022/03/28 INFO The arp is None
```