Computer Networking-lab1-Repot

课程名称：**计算机网络** 任课教师：**田臣/李文中**

学院	Dept. of Computer Science and Technology	专业（方向）	CS
学号	181830044	姓名	董宸郅
Email	pdon#foxmail.com	开始/完成日期	3.26/4.8

实验名称：IPv4 Router: Respond to ARP

实验目的

- 深入理解 **ARP** 的原理
- 初步实现具有**缓存**功能的 **Router**
- 熟练运用 switchyard 提供的各种接口，并能自主寻找适合的方法
- 加强在实验环境下调试代码的能力

实验内容

理论知识

ARP

地址解析协议(Address Resolution Protocol)，是根据IP地址获取物理地址的一个TCP/IP协议。主机发送信息时将**包含目标IP地址的ARP请求广播到局域网络上的所有主机**，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址**存入本机ARP缓存中并保留一定时间**，下次请求时直接查询ARP缓存以节约资源。

实验步骤（含测试结果与关键代码）

Task 2: Handle ARP Request

Coding

添加intfs（router的所有端口）、ipaddrs（属于router的所有IP地址）两个数据成员，方便后续操作：

```
class Router(object):
    def __init__(self, net):
        self.net = net
        # other initialization stuff here
        self.intfs = net.interfaces()
        # init ipaddrs of all intfs
        self.ipaddrs = []
        for intf in self.intfs:
            self.ipaddrs.append(intf.ipaddr)
```

接收packet并判断其是否为Request ARP，其次判断目标IP地址是否属于自己：

```
arp = pkt.get_header(Arp)
# drop this time if it's not an ARP request
if (arp) and (arp.operation == ArpOperation.Request):
    # drop if target ip does not exist here
    if arp.targetprotoaddr in self.ipaddrs:
        wanted_macaddr =
self.net.interface_by_ipaddr(arp.targetprotoaddr).ethaddr
        arp_reply = create_ip_arp_reply(wanted_macaddr, arp.senderhwaddr,
arp.targetprotoaddr, arp.senderprotoaddr)
        self.net.send_packet(dev, arp_reply)
```

两个实际编程中的疑问：

- 获取到一个包（命名为arp）之后要判断是否为ARP，如果使用arp != None 作为判断条件就一直报错（AttributeError: 'NoneType' object has no attribute 'hardwaretype'），直接用 arp 却好了，这个我很困惑，因为这里好像没有涉及到hardwaretype这个属性。
- 在收ARP包时，使用 get_header(hdrclass, returnval=None) 可以正常接收，但使用 get_header_by_name(hdrname) 就有问题，这个我也比较困惑。

Testing

```
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!
```

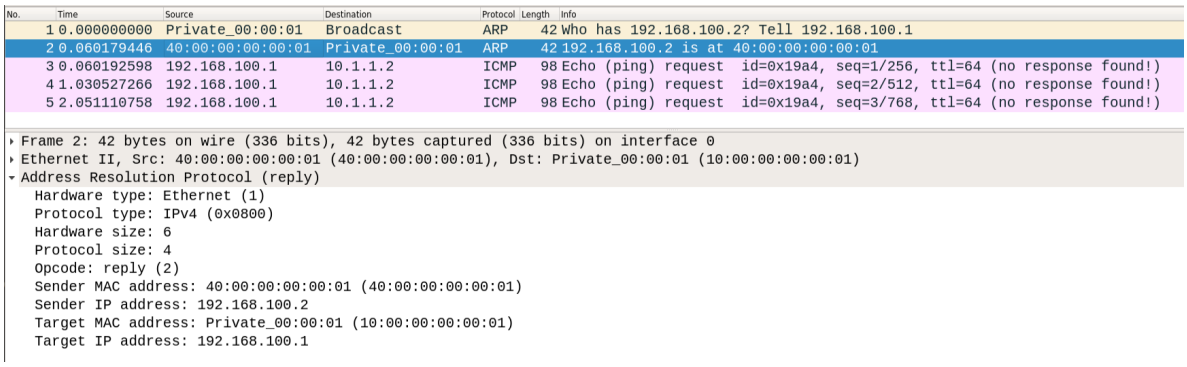
Deploying

Request：

```
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 0.060179446 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 0.060192598 192.168.100.1       10.1.1.2            ICMP     98 Echo (ping) request  id=0x19a4, seq=1/256, ttl=64 (no response found!)
     4 1.030527266 192.168.100.1       10.1.1.2            ICMP     98 Echo (ping) request  id=0x19a4, seq=2/512, ttl=64 (no response found!)
     5 2.051110758 192.168.100.1       10.1.1.2            ICMP     98 Echo (ping) request  id=0x19a4, seq=3/768, ttl=64 (no response found!)

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), 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: Private_00:00:01 (10:00:00:00:00:01)
    Sender IP address: 192.168.100.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.100.2
```

Reply:



在server1的cli中输入以下命令，以向router发出两次请求：

```
ping -c3 10.1.1.2
```

根据上述截图，server1通过ARP询问router的MAC地址。

在server1发出去的Request ARP中，Sender MAC address 与 Sender IP address 分别对应server1自己的MAC和IP地址，Target IP address 对应需要询问的对象（即此处的router）的IP地址，需要特别留意的是 Target MAC address 处为全0，因为这正是server1要询问的MAC地址。

在server1收到的Reply ARP中，Target MAC address 与 Target IP address 分别对应server1自己的MAC和IP地址，Sender IP address 对应router的IP地址，注意到 Sender MAC address 处即为server1想要的MAC地址。

又，由于目前Router类仅实现了对ARP包的处理，所以对于ICMP（如Echo）直接drop，对应上图中的‘no response found’。

Task 3: Cached ARP Table（with TIMEOUT feature)

Coding

新增一个dict作为Cached ARP Table：

（每个表项的格式为：IP-{MAC，last_using_time})

```
class Router(object):
    def __init__(self, net):
        ...
        # init the ARP table----IP-{MAC,last_time}
        self.tab = {}
```

当收到一个ARP请求时，新增或更新相应的表项：

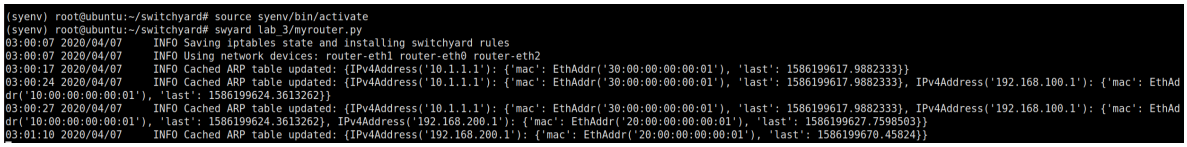
（如果某个IP-MAC超过20s没有再次出现，将之删除）

```
arp = pkt.get_header(Arp)
# drop this time if it's not an ARP request
if (arp) and (arp.operation == ArpOperation.Request):
    # add a new entry into the table or just update a recorded one
    self.tab[arp.senderprotoaddr] = {'mac': arp.senderhwaddr, 'last':
time.time()}
    # delete old entries after 20s of idleness
    for host in list(self.tab):
        if time.time() - self.tab[host]['last'] > 20:
            del self.tab[host]
```

```
    log_info("Cached ARP table updated: {}".format(str(self.tab)))

    # drop if target ip does not exist here
    if arp.targetprotoaddr in self.ipaddrs:
        wanted_macaddr = self.net.interface_by_ipaddr(arp.targetprotoaddr).ethaddr
        arp_reply = create_ip_arp_reply(wanted_macaddr, arp.senderhwaddr, arp.targetprotoaddr, arp.senderprotoaddr)
        self.net.send_packet(dev, arp_reply)
```

Testing



首先，依次在client、server1、server2的cli中输入以下命令，且前后间隔不超过20s：

```
ping -c3 10.1.1.2
```

从上图的log_info可知，三者的IP、MAC、last_using_time被依次计入表中。

其次，在间隔20s以上后，在server2中再次输入以上命令。

此次的log_info显示，表中只有server2的相关信息，client和server1的信息已被删除，从而验证了timeout功能。

总结与感想

这次实验总体的代码量比较少，而且有了lab2的铺垫，难度就没那么大了，估计之后两次就不会这么轻松了。

对于switchyard方法的调用以及Python语言的使用，相比前两次已经熟练了不少，但依然会遇到一些自己想不通的玄学问题（如Task 2中提到的那样）。

在理论学习方面，已经基本完全适应了纯英文课本的学习，发现少去翻译上的障碍后学起来反而轻松了不少，也希望利用当前的一段相对空闲期多看点教材内容，以便更好地适应教学进度。