# 南京大学计算机网络实验报告

**任课教师:田臣**

# 实验三 Respond to ARP

**计算机科学与技术系**

181860055 **刘国涛**

邮箱: 181860055@smail.nju.edu.cn

**2020年4月1日**

# 实验目的

- 学习router对ARP的响应机制
- 实现带缓存表的router类

# 实验内容

## TASK 2 Handle ARP Request

**任务概述** router接收到包的时候判断是否是ARP包，并对属于自己的ARP包进行答复

**任务实现**

1.如何判断接收到的是ARP包

通过判断ARP的数据包头是否存在来判断包是否属于ARP包：

```python
arp = pkt.get_header(Arp)
if arp is not None: # packet is a ARP packet
    #do something
```

2.如何对属于自己的ARP包答复

通过调用API `create_ip_arp_reply` 创建ARP reply packet，然后调用 `send_packet` 发送

```python
try:
    interface = 
self.net.interface_by_ipaddr(arp.targetprotoaddr)
except KeyError:
    interface = None

if interface is not None: # intended for me
    reply = 
create_ip_arp_reply(interface.ethaddr,arp.senderhwaddr,
            arp.targetprotoaddr,arp.senderprotoaddr)
    self.net.send_packet(dev, reply)
```

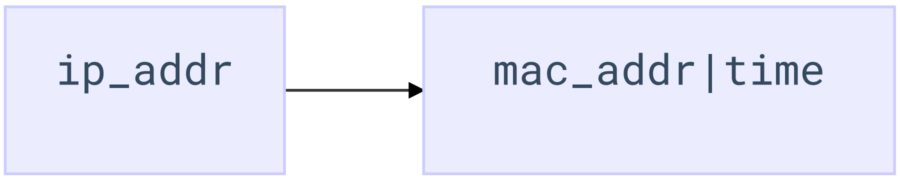
## TASK 3 Cache ARP Table

**任务概述** router每次接收到ARP包时缓存下包内的ip到mac的映射关系，并加入Timeout机制

**任务实现**

1.实现一个缓存表类

如图，这是表内的一项：



代码实现为：

```python
class ip2mac_table_item():
    def __init__(self,mac):
        self.time = time.time()
        self.value = mac
    def timeout(self):
        return time.time()-self.time > 10
    def __str__(self):
        return "mac:{} time:{}".format(self.value,self.time)
```

整张表使用dict进行实现，包含 `get` 和 `set` 方法

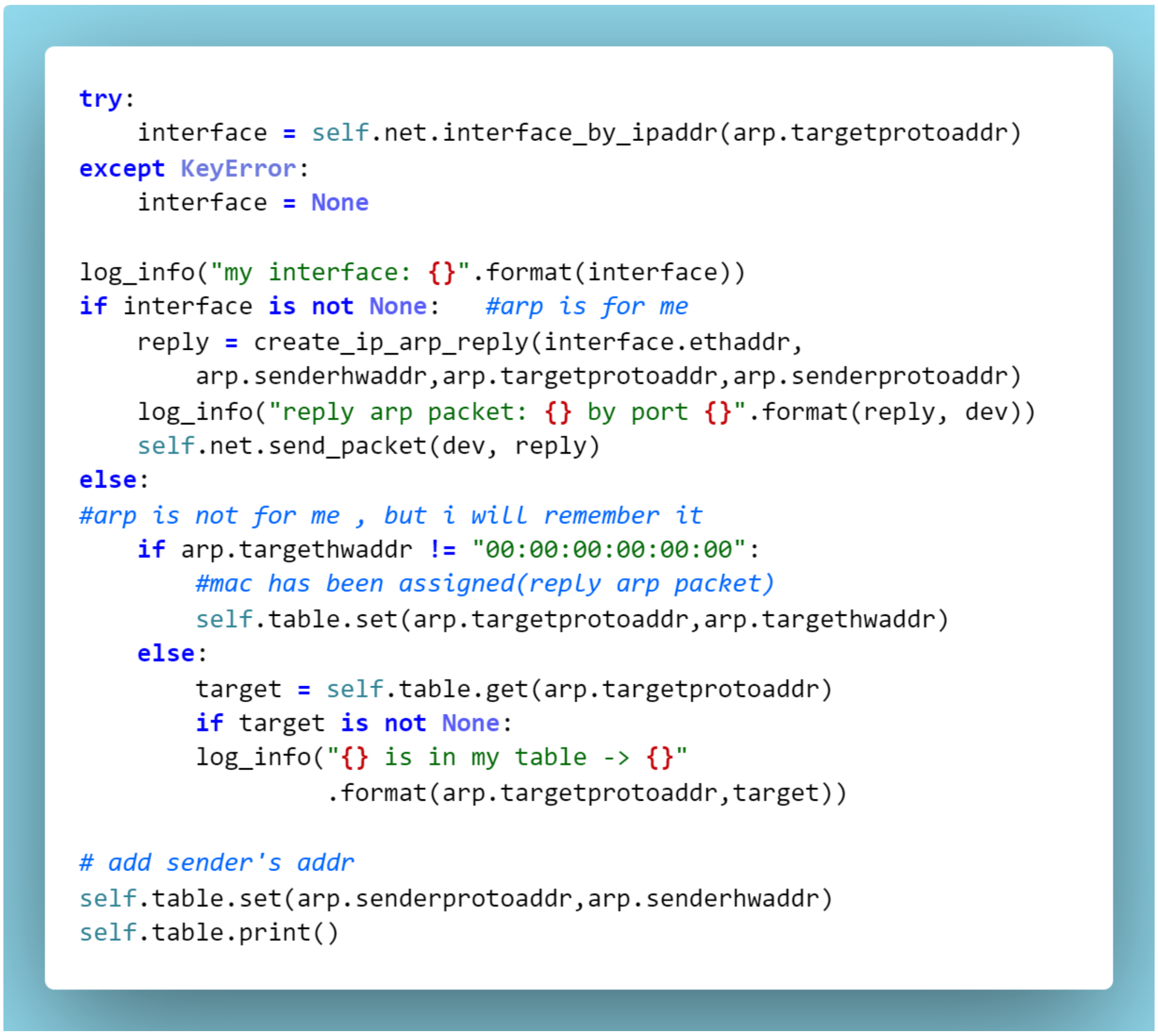
```python
class ip2mac_table():
    def __init__(self):
        self.table = {}

    def get(self,key):
        if key in self.table: #get and update
```

```
            value = self.table[key]
            if value.timeout():
                log_info("Timeout Item:{}".format(value))
                value = None
                self.table.pop(key)
            else:
                value = value.value
        else:
            value = None
        return value

    def set(self,key,value):
        self.table[key] = ip2mac_table_item(value)

```

2.用缓存表实现相应的存储逻辑

对于router收到的一个ARP包，做如下处理：

1. 把sender的ip和mac加入缓存表
2. 如果target的ip和mac已分配，则加入缓存表

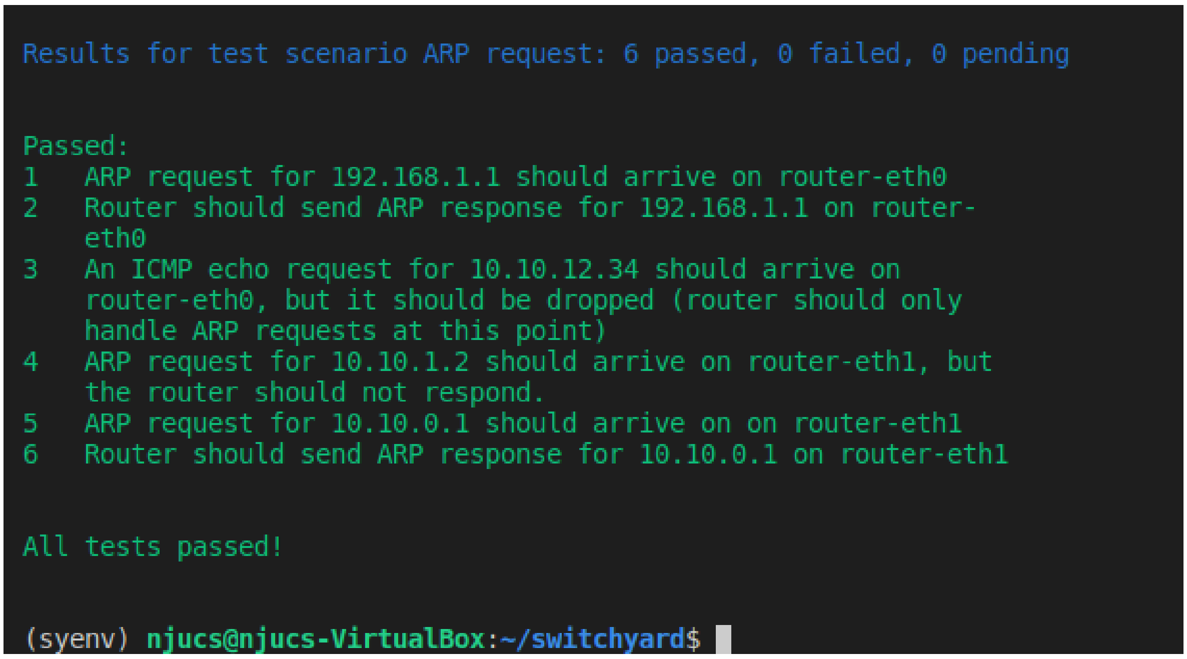


```python
try:
    interface = self.net.interface_by_ipaddr(arp.targetprotoaddr)
except KeyError:
    interface = None

log_info("my interface: {}".format(interface))
if interface is not None:   #arp is for me
    reply = create_ip_arp_reply(interface.ethaddr,
        arp.senderhwaddr,arp.targetprotoaddr,arp.senderprotoaddr)
    log_info("reply arp packet: {} by port {}".format(reply, dev))
    self.net.send_packet(dev, reply)
else:
#arp is not for me , but i will remember it
    if arp.targethwaddr != "00:00:00:00:00:00":
        #mac has been assigned(reply arp packet)
        self.table.set(arp.targetprotoaddr,arp.targethwaddr)
    else:
        target = self.table.get(arp.targetprotoaddr)
        if target is not None:
        log_info("{} is in my table -> {}"
                 .format(arp.targetprotoaddr,target))

# add sender's addr
self.table.set(arp.senderprotoaddr,arp.senderhwaddr)
self.table.print()
```

# 实验结果

## TASK 2

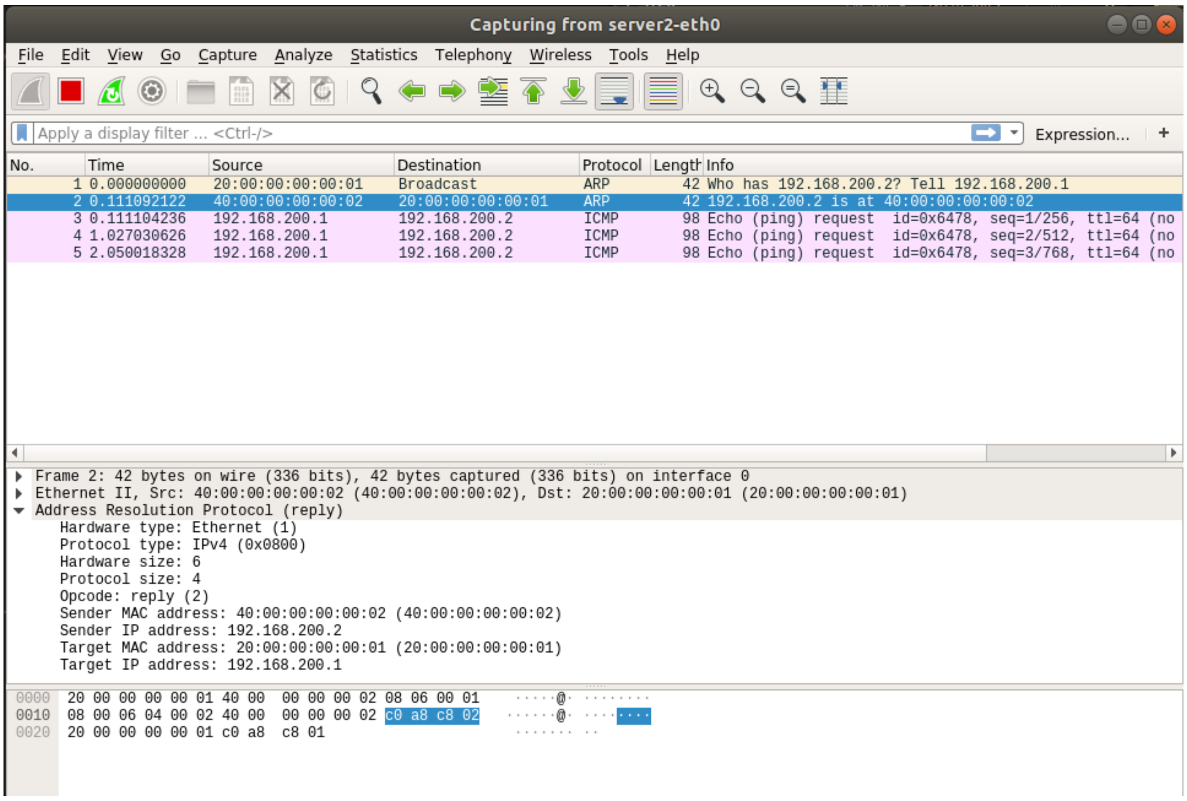
**Testing:**



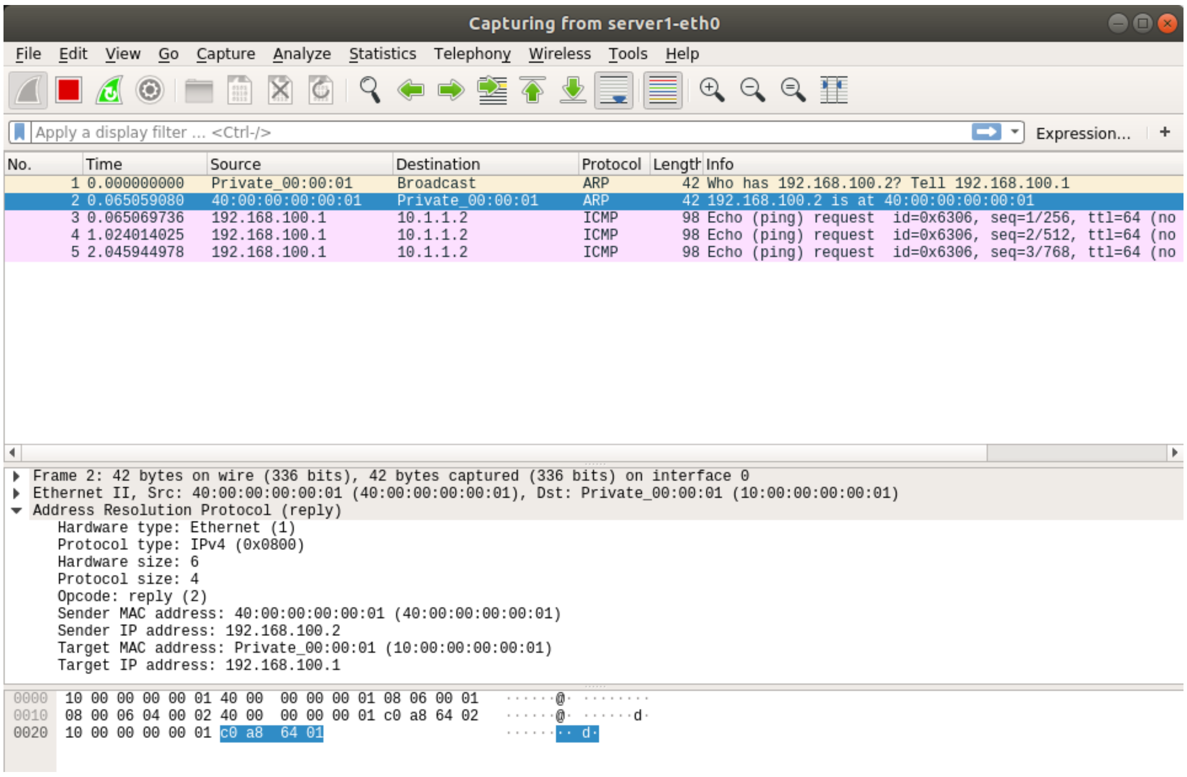
```
Results for test scenario ARP request: 6 passed, 0 failed, 0 pending


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!


(syenv) njucs@njucs-VirtualBox:~/switchyard$
```

**Deploying:**

通过在 `mininet` 中分别让client/server1/server2对router ping，得到如下结果：


Capturing from server2-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

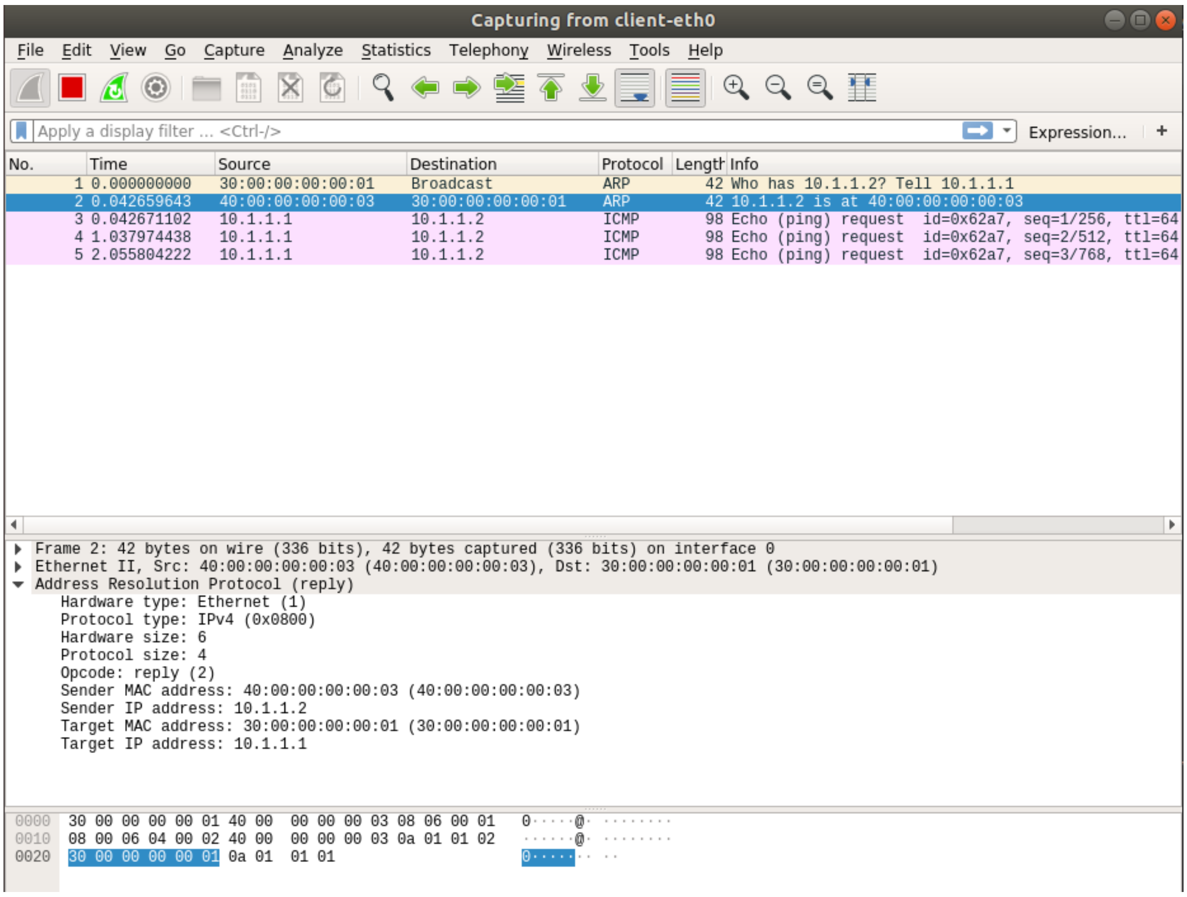
```
No.  Time         Source             Destination        Protocol Length Info
   1 0.000000000  20:00:00:00:00:01  Broadcast          ARP       42 Who has 192.168.200.2? Tell 192.168.200.1
   2 0.111092122  40:00:00:00:00:02  20:00:00:00:00:01  ARP       42 192.168.200.2 is at 40:00:00:00:00:02
   3 0.111104236  192.168.200.1      192.168.200.2      ICMP      98 Echo (ping) request  id=0x6478, seq=1/256, ttl=64 (no
   4 1.027030626  192.168.200.1      192.168.200.2      ICMP      98 Echo (ping) request  id=0x6478, seq=2/512, ttl=64 (no
   5 2.050018328  192.168.200.1      192.168.200.2      ICMP      98 Echo (ping) request  id=0x6478, seq=3/768, ttl=64 (no

▸ Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: 40:00:00:00:00:02 (40:00:00:00:00:02), Dst: 20:00:00:00:00:01 (20:00:00:00:00:01)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 40:00:00:00:00:02 (40:00:00:00:00:02)
    Sender IP address: 192.168.200.2
    Target MAC address: 20:00:00:00:00:01 (20:00:00:00:00:01)
    Target IP address: 192.168.200.1

0000  20 00 00 00 00 01 40 00  00 00 00 02 08 06 00 01    ······@· ········
0010  08 00 06 04 00 02 40 00  00 00 00 02 c0 a8 c8 02    ······@· ········
0020  20 00 00 00 00 01 c0 a8  c8 01                       ········ ··
```


Capturing from server1-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
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 0.065059080  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 0.065069736  192.168.100.1      10.1.1.2           ICMP       98 Echo (ping) request  id=0x6306, seq=1/256, ttl=64 (no
   4 1.024014025  192.168.100.1      10.1.1.2           ICMP       98 Echo (ping) request  id=0x6306, seq=2/512, ttl=64 (no
   5 2.045944978  192.168.100.1      10.1.1.2           ICMP       98 Echo (ping) request  id=0x6306, seq=3/768, ttl=64 (no
```

```
▸ 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
```

```
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·
```


Capturing from client-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.  Time         Source             Destination        Protocol Length Info
   1 0.000000000  30:00:00:00:00:01  Broadcast          ARP        42 Who has 10.1.1.2? Tell 10.1.1.1
   2 0.042659643  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 0.042671102  10.1.1.1           10.1.1.2           ICMP       98 Echo (ping) request  id=0x62a7, seq=1/256, ttl=64
   4 1.037974438  10.1.1.1           10.1.1.2           ICMP       98 Echo (ping) request  id=0x62a7, seq=2/512, ttl=64
   5 2.055804222  10.1.1.1           10.1.1.2           ICMP       98 Echo (ping) request  id=0x62a7, seq=3/768, ttl=64
```

```
▸ 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
```

```
0000  30 00 00 00 00 01 40 00  00 00 00 03 08 06 00 01   0·····@· ········
0010  08 00 06 04 00 02 40 00  00 00 00 03 0a 01 01 02   ······@· ········
0020  30 00 00 00 00 01 0a 01  01 01                     0······· ··
```

过程分析：

以client为例，

1. 在client要向路由发送ICMP包之前，先广播ARP包询问了路由的MAC地址

2. 路由收到ARP包后，查找interfaces发现是询问自己的包，于是构建ARP reply packet，发送给client
3. client接收到了reply，并被wireshark捕获

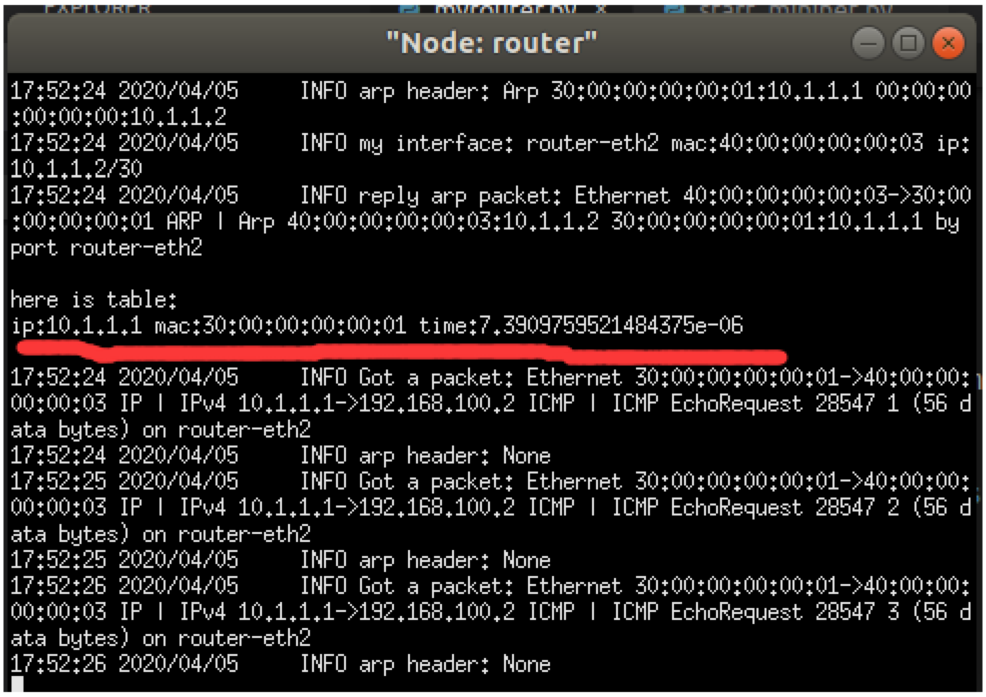
## TASK 3

**Deploying:**

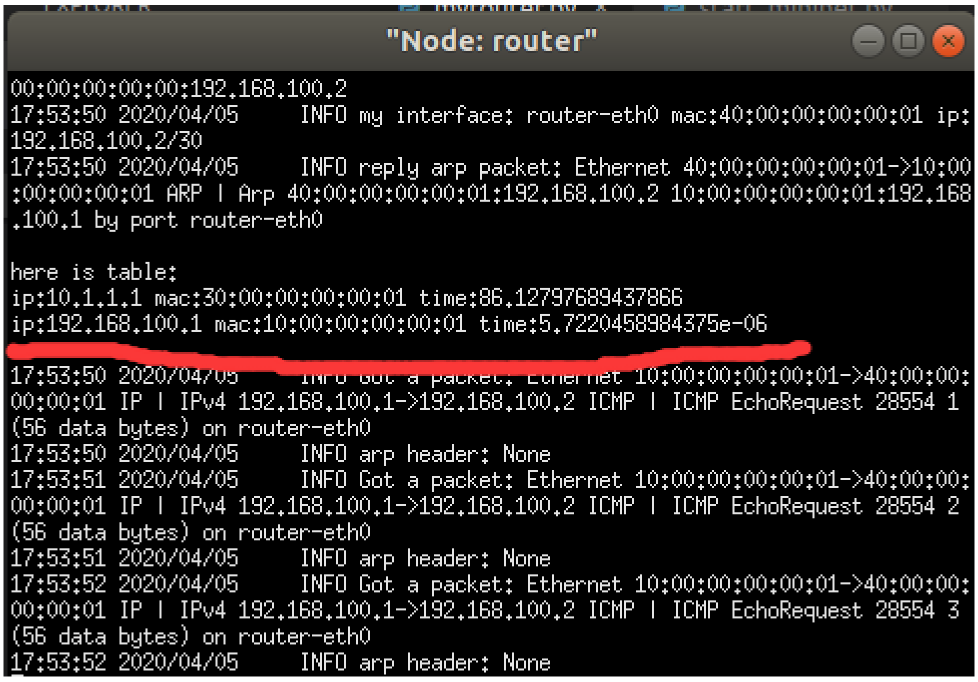
首先测试表是否能够正确缓存下收到的ARP包，在mininet中让client/server1/server2依次对router ping

```
client ping -c3 router
server1 ping -c3 router
server2 ping -c3 router
```

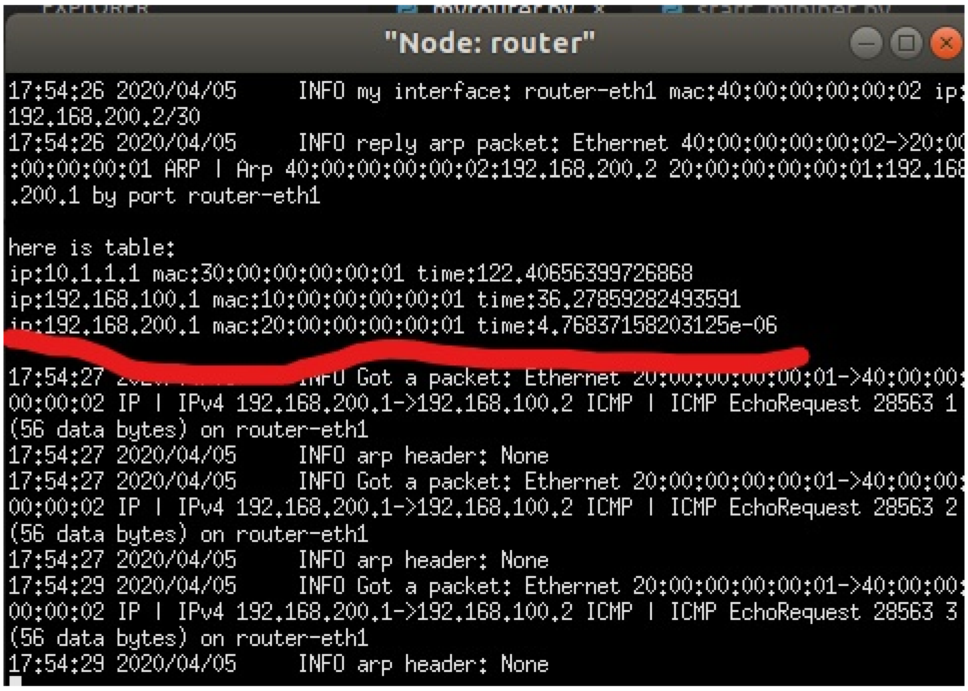
得到的结果依次如下图（`time` 表示表项的创建时长）



```
17:52:24 2020/04/05     INFO arp header: Arp 30:00:00:00:00:01:10.1.1.1 00:00:00
:00:00:00:10.1.1.2
17:52:24 2020/04/05     INFO my interface: router-eth2 mac:40:00:00:00:00:03 ip:
10.1.1.2/30
17:52:24 2020/04/05     INFO reply arp packet: 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 by
port router-eth2

here is table:
ip:10.1.1.1 mac:30:00:00:00:00:01 time:7.3909759521484375e-06

17:52:24 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 IP | IPv4 10.1.1.1->192.168.100.2 ICMP | ICMP EchoRequest 28547 1 (56 d
ata bytes) on router-eth2
17:52:24 2020/04/05     INFO arp header: None
17:52:25 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 IP | IPv4 10.1.1.1->192.168.100.2 ICMP | ICMP EchoRequest 28547 2 (56 d
ata bytes) on router-eth2
17:52:25 2020/04/05     INFO arp header: None
17:52:26 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 IP | IPv4 10.1.1.1->192.168.100.2 ICMP | ICMP EchoRequest 28547 3 (56 d
ata bytes) on router-eth2
17:52:26 2020/04/05     INFO arp header: None
```



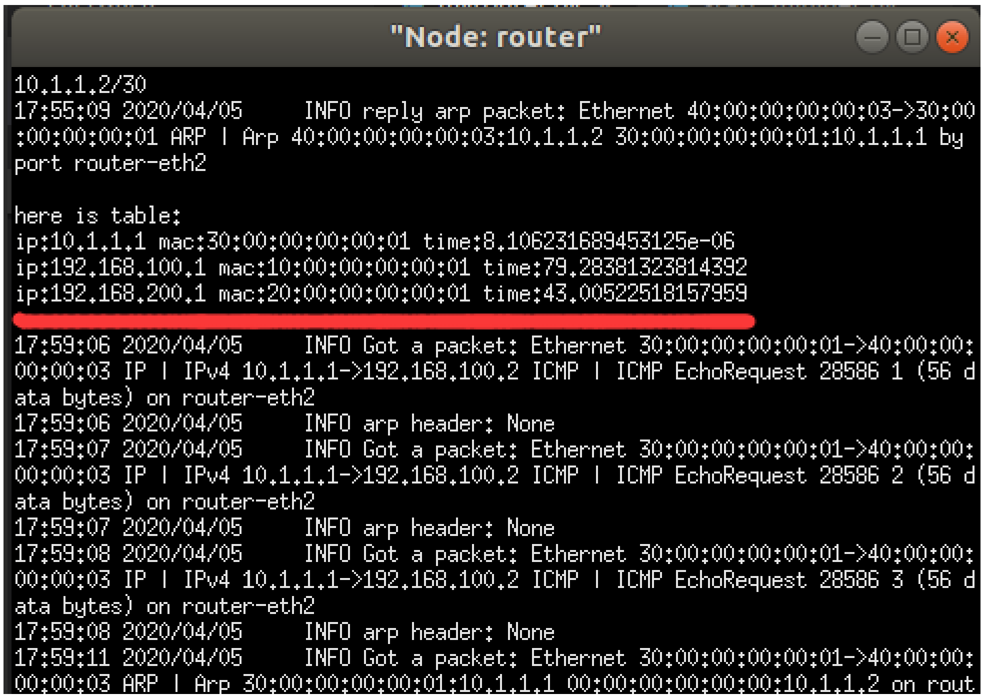
```
"Node: router"
00:00:00:00:00:192.168.100.2
17:53:50 2020/04/05     INFO my interface: router-eth0 mac:40:00:00:00:00:01 ip:
192.168.100.2/30
17:53:50 2020/04/05     INFO reply arp packet: Ethernet 40:00:00:00:00:01->10:00
:00:00:00:01 ARP | Arp 40:00:00:00:00:01:192.168.100.2 10:00:00:00:00:01:192.168
.100.1 by port router-eth0

here is table:
ip:10.1.1.1 mac:30:00:00:00:00:01 time:86.12797689437866
ip:192.168.100.1 mac:10:00:00:00:00:01 time:5.7220458984375e-06
17:53:50 2020/04/05     INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:
00:00:01 IP | IPv4 192.168.100.1->192.168.100.2 ICMP | ICMP EchoRequest 28554 1
(56 data bytes) on router-eth0
17:53:50 2020/04/05     INFO arp header: None
17:53:51 2020/04/05     INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:
00:00:01 IP | IPv4 192.168.100.1->192.168.100.2 ICMP | ICMP EchoRequest 28554 2
(56 data bytes) on router-eth0
17:53:51 2020/04/05     INFO arp header: None
17:53:52 2020/04/05     INFO Got a packet: Ethernet 10:00:00:00:00:01->40:00:00:
00:00:01 IP | IPv4 192.168.100.1->192.168.100.2 ICMP | ICMP EchoRequest 28554 3
(56 data bytes) on router-eth0
17:53:52 2020/04/05     INFO arp header: None
```



```
"Node: router"
17:54:26 2020/04/05     INFO my interface: router-eth1 mac:40:00:00:00:00:02 ip:
192.168.200.2/30
17:54:26 2020/04/05     INFO reply arp packet: Ethernet 40:00:00:00:00:02->20:00
:00:00:00:01 ARP | Arp 40:00:00:00:00:02:192.168.200.2 20:00:00:00:00:01:192.168
.200.1 by port router-eth1

here is table:
ip:10.1.1.1 mac:30:00:00:00:00:01 time:122.40656399726868
ip:192.168.100.1 mac:10:00:00:00:00:01 time:36.27859282493591
ip:192.168.200.1 mac:20:00:00:00:00:01 time:4.76837158203125e-06
17:54:27 2020/04/05     INFO Got a packet: Ethernet 20:00:00:00:00:01->40:00:00:
00:00:02 IP | IPv4 192.168.200.1->192.168.100.2 ICMP | ICMP EchoRequest 28563 1
(56 data bytes) on router-eth1
17:54:27 2020/04/05     INFO arp header: None
17:54:27 2020/04/05     INFO Got a packet: Ethernet 20:00:00:00:00:01->40:00:00:
00:00:02 IP | IPv4 192.168.200.1->192.168.100.2 ICMP | ICMP EchoRequest 28563 2
(56 data bytes) on router-eth1
17:54:27 2020/04/05     INFO arp header: None
17:54:29 2020/04/05     INFO Got a packet: Ethernet 20:00:00:00:00:01->40:00:00:
00:00:02 IP | IPv4 192.168.200.1->192.168.100.2 ICMP | ICMP EchoRequest 28563 3
(56 data bytes) on router-eth1
17:54:29 2020/04/05     INFO arp header: None
```

可以看到，缓存表成功记录下了需要的信息

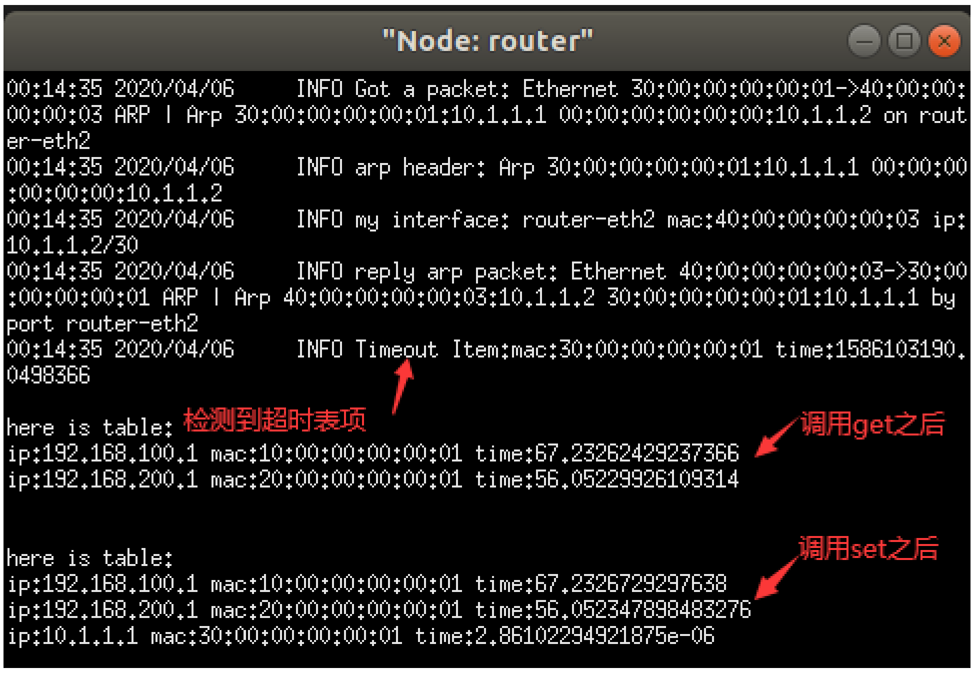
接着再用client ping router，可以看到client对应的表项的时间更新了



```
10.1.1.2/30
17:55:09 2020/04/05     INFO reply arp packet: 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 by
port router-eth2

here is table:
ip:10.1.1.1 mac:30:00:00:00:00:01 time:8.106231689453125e-06
ip:192.168.100.1 mac:10:00:00:00:00:01 time:79.28381323814392
ip:192.168.200.1 mac:20:00:00:00:00:01 time:43.00522518157959
17:59:06 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 IP | IPv4 10.1.1.1->192.168.100.2 ICMP | ICMP EchoRequest 28586 1 (56 d
ata bytes) on router-eth2
17:59:06 2020/04/05     INFO arp header: None
17:59:07 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 IP | IPv4 10.1.1.1->192.168.100.2 ICMP | ICMP EchoRequest 28586 2 (56 d
ata bytes) on router-eth2
17:59:07 2020/04/05     INFO arp header: None
17:59:08 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 IP | IPv4 10.1.1.1->192.168.100.2 ICMP | ICMP EchoRequest 28586 3 (56 d
ata bytes) on router-eth2
17:59:08 2020/04/05     INFO arp header: None
17:59:11 2020/04/05     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 ARP | Arp 30:00:00:00:00:01:10.1.1.1 00:00:00:00:00:00:10.1.1.2 on rout
```

接着测试Timeout：由于实验中没有合适的场景（需要调用get方法才会判断是否超时）测试这种机制，**为了达到演示效果**，这里在set方法中**无意义地**调用一次get

```python
class ip2mac_table():
    def __init__(self):
        #省略

    def get(self,key):
        #省略

    def set(self,key,value):
        self.get(key)
        self.print()
        self.table[key] = ip2mac_table_item(value)
```



```
00:14:35 2020/04/06     INFO Got a packet: Ethernet 30:00:00:00:00:01->40:00:00:
00:00:03 ARP | Arp 30:00:00:00:00:01:10.1.1.1 00:00:00:00:00:00:10.1.1.2 on rout
er-eth2
00:14:35 2020/04/06     INFO arp header: Arp 30:00:00:00:00:01:10.1.1.1 00:00:00
:00:00:00:10.1.1.2
00:14:35 2020/04/06     INFO my interface: router-eth2 mac:40:00:00:00:00:03 ip:
10.1.1.2/30
00:14:35 2020/04/06     INFO reply arp packet: 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 by
port router-eth2
00:14:35 2020/04/06     INFO Timeout Item:mac:30:00:00:00:00:01 time:1586103190.
0498366

here is table:
ip:192.168.100.1 mac:10:00:00:00:00:01 time:67.23262429237366
ip:192.168.200.1 mac:20:00:00:00:00:01 time:56.05229926109314


here is table:
ip:192.168.100.1 mac:10:00:00:00:00:01 time:67.2326729297638
ip:192.168.200.1 mac:20:00:00:00:00:01 time:56.052347898483276
ip:10.1.1.1 mac:30:00:00:00:00:01 time:2.86102294921875e-06
```

如图，在get里检测到了超时表项并将其pop，在set中加入新的表项

## 总结与感想

router的缓存方式和switch有些类似，只不过两者工作于不同的层。本次学习了解到了router对于ARP的响应方式，但是显然这还不是router的全部。期待下一次实验对于router的学习。