

实验报告

一、实验名称

Respond to ARP

二、实验目的

完成路由器（router）的一个功能：回复ARP请求

三、实验内容

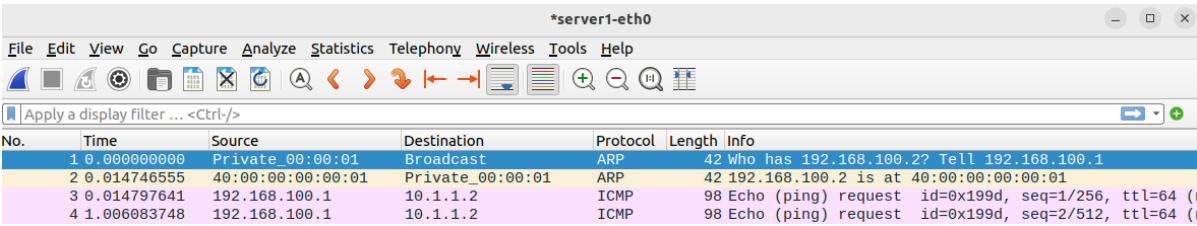
回复ARP请求的逻辑：

收到一个包时，先读取包头，如果是ARP请求，那么进入handle_arp_packet处理：先查看自身的接口，看有没有接口的IP是ARP包的目标IP，如果有，回复该ARP请求：使用create_ip_arp_reply来创建回复包（其中sender方填对应接口的信息，target方填ARP来源方的信息），并发向收到包的那个接口

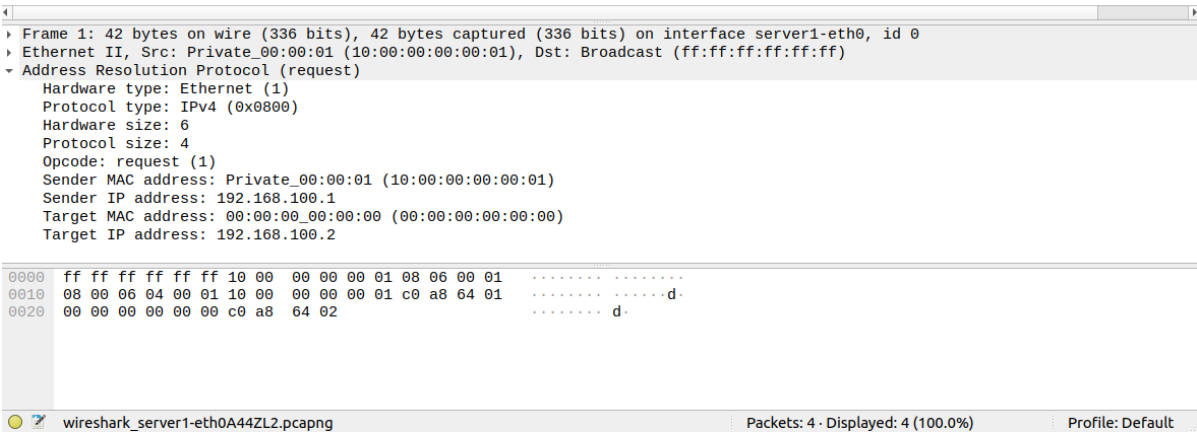
在mininet中测试：

server1 ping -c2 10.1.1.2

第一行：server1广播发了一个针对路由器地址的ARP请求，其中Target MAC address都是0，因为是请求的地址



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	0.014746555	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	0.014797641	192.168.100.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x199d, seq=1/256, ttl=64 (r
4	1.006083748	192.168.100.1	10.1.1.2	ICMP	98	Echo (ping) request id=0x199d, seq=2/512, ttl=64 (r

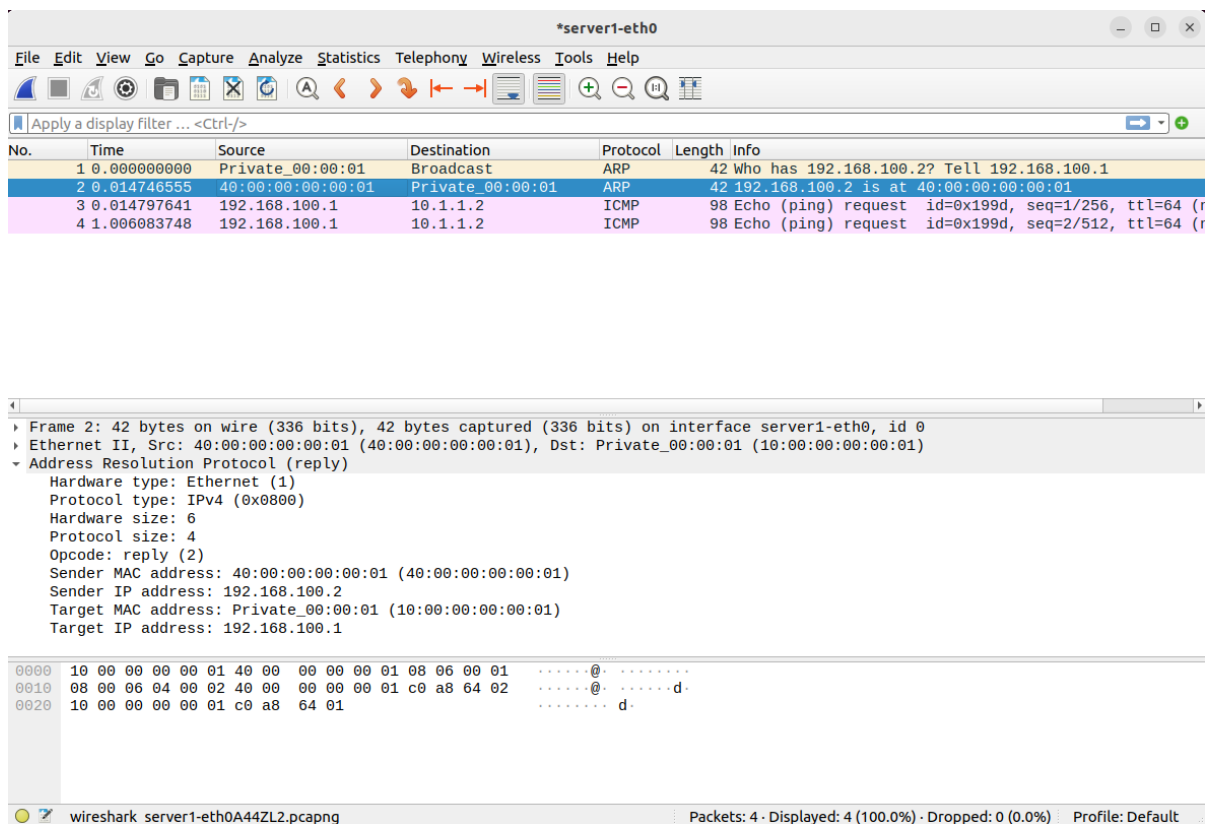


Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface server1-eth0, id 0	
Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)	
Address Resolution Protocol (request)	
Hardware type: Ethernet (1)	
Protocol type: IPv4 (0x0800)	
Hardware size: 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	

0000	ff ff ff ff ff ff 10 00	00 00 00 01 08 06 00 01d.
0010	08 00 06 04 00 01 10 00	00 00 00 01 c0 a8 64 01d.
0020	00 00 00 00 00 00 c0 a8	64 02d.

wireshark_server1-eth0A44ZL2.pcapng Packets: 4 · Displayed: 4 (100.0%) Profile: Default

第二行：路由器给server1发了一个ARP回复，源地址是路由器，并且MAC已经填好，目标地址是client



建立ARP table

在收到一个包时，将发送者的IP地址和MAC地址添加到arp_table中

其中arp_table使用dict数据结构，key是IP，value是MAC

四、实验结果

通过测试样例

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!

建立arp_table

测试样例的arp_table:

```
(syenv) uuu@Ubuntu22:~/net_lab/workspace/lab-3-xkxkzzz$ sudo swyard -t testcases/myrouter1_testscenario
io.srpy myrouter.py
15:47:40 2024/04/14 INFO Starting test scenario testcases/myrouter1_testscenario.srpy
15:47:40 2024/04/14 INFO arp_table: {IPv4Address('192.168.1.100'): EthAddr('30:00:00:00:00:01')}
15:47:40 2024/04/14 INFO arp_table: {IPv4Address('192.168.1.100'): EthAddr('30:00:00:00:00:01'),
IPv4Address('10.10.1.1'): EthAddr('60:00:de:ad:be:ef')}
15:47:40 2024/04/14 INFO arp_table: {IPv4Address('192.168.1.100'): EthAddr('30:00:00:00:00:01'),
IPv4Address('10.10.1.1'): EthAddr('60:00:de:ad:be:ef'), IPv4Address('10.10.5.5'): EthAddr('70:00:ca:f
e:c0:de')}

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

自测:

client ping -c1 router

server1 ping -c1 router

server2 ping -c1 router

```
15:53:45 2024/04/14 INFO arp_table: {IPv4Address('10.1.1.1'): EthAddr('30:00:00:00:00:01')}
15:54:23 2024/04/14 INFO arp_table: {IPv4Address('10.1.1.1'): EthAddr('30:00:00:00:00:01'), IPv4Address('192.168.100.1'): EthAddr('10:00:00:00:00:01')}
15:55:20 2024/04/14 INFO arp_table: {IPv4Address('10.1.1.1'): EthAddr('30:00:00:00:00:01'), IPv4Address('192.168.100.1'): EthAddr('10:00:00:00:00:01'), IPv4Address('192.168.200.1'): EthAddr('20:00:00:00:00:01')}
```

arp_table不断添加表项，分别是client, server1, server2的IP与MAC

五、核心代码

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

    def handle_arp_packet(self, recv):
        timestamp, ifaceName, packet = recv
        arp = packet.get_header(Arp)
        self.arp_table[arp.senderprotoaddr] = arp.senderhwaddr
        log_info("arp_table: {}".format(self.arp_table))
        for interface in self.interfaces:
            if(arp.targetprotoaddr == interface.ipaddr):
                arp_reply = create_ip_arp_reply(interface.ethaddr,
                arp.senderhwaddr, interface.ipaddr, arp.senderprotoaddr)
                self.net.send_packet(ifaceName, arp_reply)
                return

    def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
        timestamp, ifaceName, packet = recv
        # TODO: your logic here
        arp = packet.get_header(Arp)
```

```
if arp:  
    self.handle_arp_packet(recv)  
    return
```