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：方毓楚、郑浩、陈伟等（排名不分先后）

学院	地理与海洋科学学院	专业	地理信息科学
学号	191830173	姓名	徐嵩
Email	song.xv@outlook.com	开始/完成日期	2022/4/14

1. 实验名称

Lab 4: Forwarding Packets

2. 实验目的

1. 根据静态路由表按最长前缀匹配规则转发分组；
2. 创建 ARP request，查找目的 IP 对应的 MAC 地址。

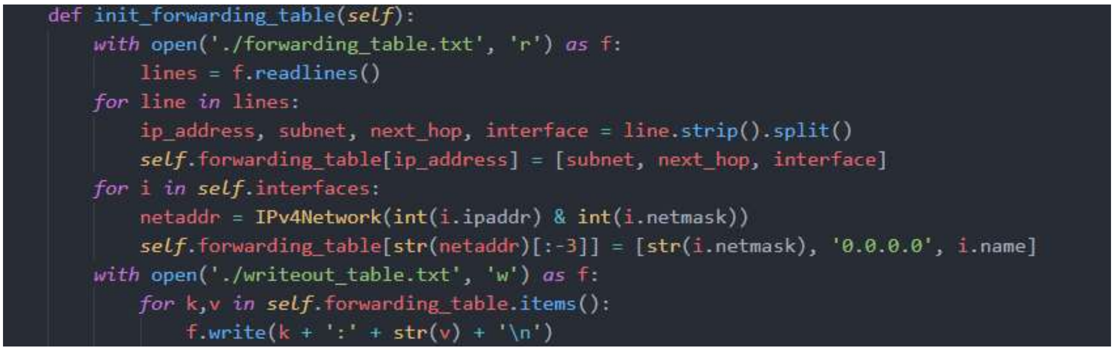
3. 实验内容

1. 构建路由转发表，根据当前工作目录和路由器自身端口的信息，将网络地址、子网掩码、下一跳地址和接口名称作为一项加入转发表；
2. 根据最长匹配原则对目的 IP 地址进行匹配，如果没有找到匹配项或目的 IP 是路由器自身接口，则不做处理；

4. 实验结果

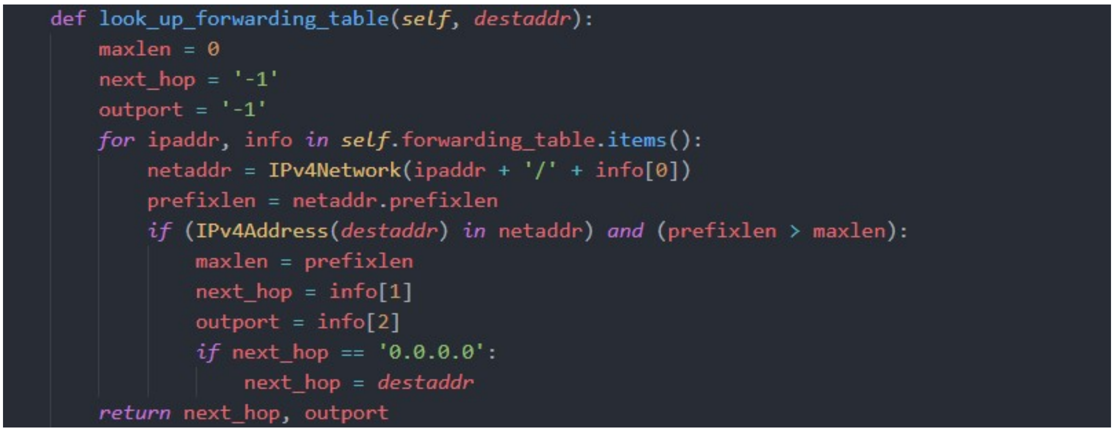
In the report, show how you implement the logic of building IP forwarding table and matching the destination IP addresses.

创建转发表分两步，一是从 forwarding_table.txt 中获取，二是从自身接口获取。要注意从自身接口获取的下一跳地址设置为’0.0.0.0’表示可以直接连接到该网络。



```python
def init_forwarding_table(self):
    with open('./forwarding_table.txt', 'r') as f:
        lines = f.readlines()
    for line in lines:
        ip_address, subnet, next_hop, interface = line.strip().split()
        self.forwarding_table[ip_address] = [subnet, next_hop, interface]
    for i in self.interfaces:
        netaddr = IPv4Network(int(i.ipaddr) & int(i.netmask))
        self.forwarding_table[str(netaddr)[:-3]] = [str(i.netmask), '0.0.0.0', i.name]
    with open('./writeout_table.txt', 'w') as f:
        for k,v in self.forwarding_table.items():
            f.write(k + ':' + str(v) + '\n')
```

查找转发表的时候，根据转发表的每一项创建一个 IPv4Network 对象，分别求出目的 IP 是否在该网络对象中，以及匹配的网络前缀长度，最后返回最长前缀匹配的网络对应的下一跳 IP 地址和接口名称。特别注意，当下一跳地址为’0.0.0.0’时，要将下一跳 IP 直接替换为目的 IP，表示下一跳就能到达目的 IP。

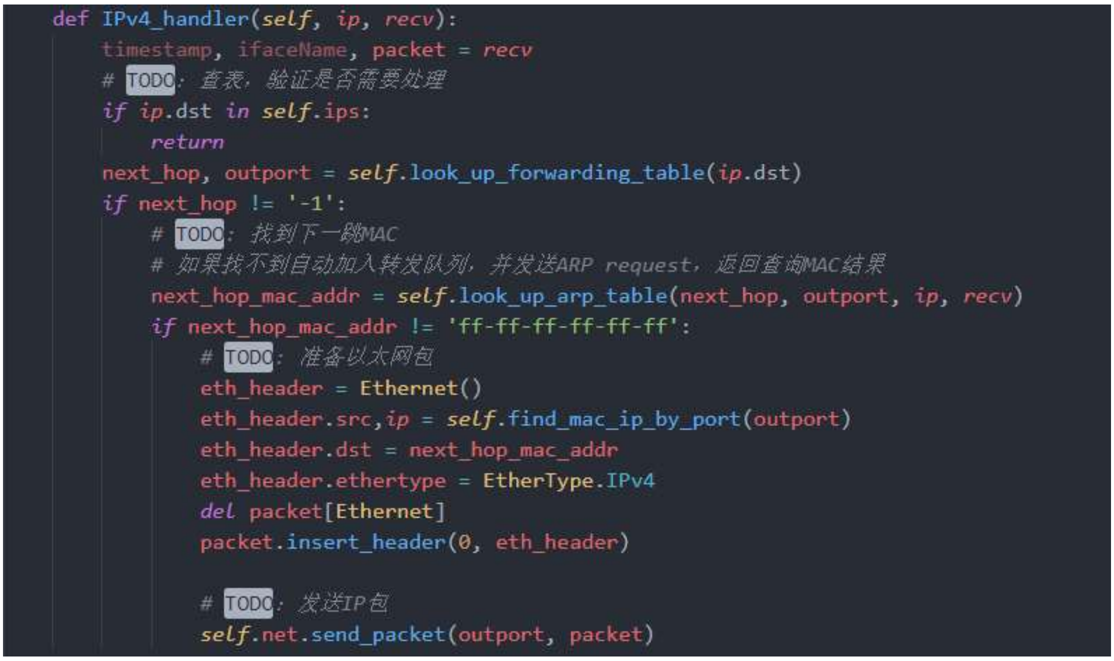


```python
def look_up_forwarding_table(self, destaddr):
    maxlen = 0
    next_hop = '-1'
    outport = '-1'
    for ipaddr, info in self.forwarding_table.items():
        netaddr = IPv4Network(ipaddr + '/' + info[0])
        prefixlen = netaddr.prefixlen
        if (IPv4Address(destaddr) in netaddr) and (prefixlen > maxlen):
            maxlen = prefixlen
            next_hop = info[1]
            outport = info[2]
            if next_hop == '0.0.0.0':
                next_hop = destaddr
    return next_hop, outport
```

In the report, show how you implement the logic of forwarding the packet and ARP.

处理 IPv4 分组，转发分组和发送 ARP

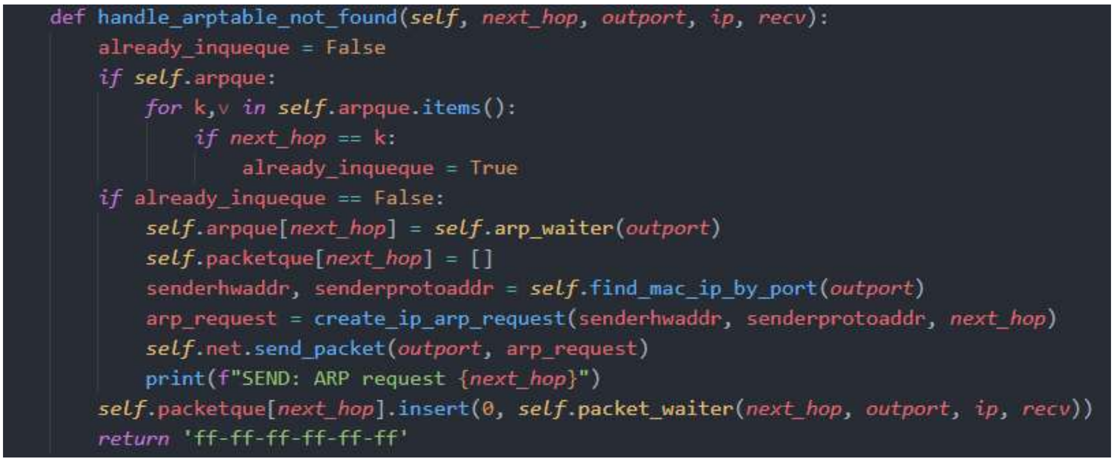
1. 查看目的地址是否是自身接口，是则结束处理
2. 查看目的地址是否在转发表中，不是则结束处理
3. 查找 ARP 表中下一跳对应的 MAC 地址，如果没找到，加入等待队列
4. 如果找到立即发送



```python
def IPv4_handler(self, ip, recv):
    timestamp, ifaceName, packet = recv
    # TODO: 查表，验证是否需要处理
    if ip.dst in self.ips:
        return
    next_hop, outport = self.look_up_forwarding_table(ip.dst)
    if next_hop != '-1':
        # TODO: 找到下一跳MAC
        # 如果找不到自动加入转发队列，并发送ARP request，返回查询MAC结果
        next_hop_mac_addr = self.look_up_arp_table(next_hop, outport, ip, recv)
        if next_hop_mac_addr != 'ff-ff-ff-ff-ff-ff':
            # TODO: 准备以太网包
            eth_header = Ethernet()
            eth_header.src,ip = self.find_mac_ip_by_port(outport)
            eth_header.dst = next_hop_mac_addr
            eth_header.ethertype = EtherType.IPv4
            del packet[Ethernet]
            packet.insert_header(0, eth_header)

            # TODO: 发送IP包
            self.net.send_packet(outport, packet)
```

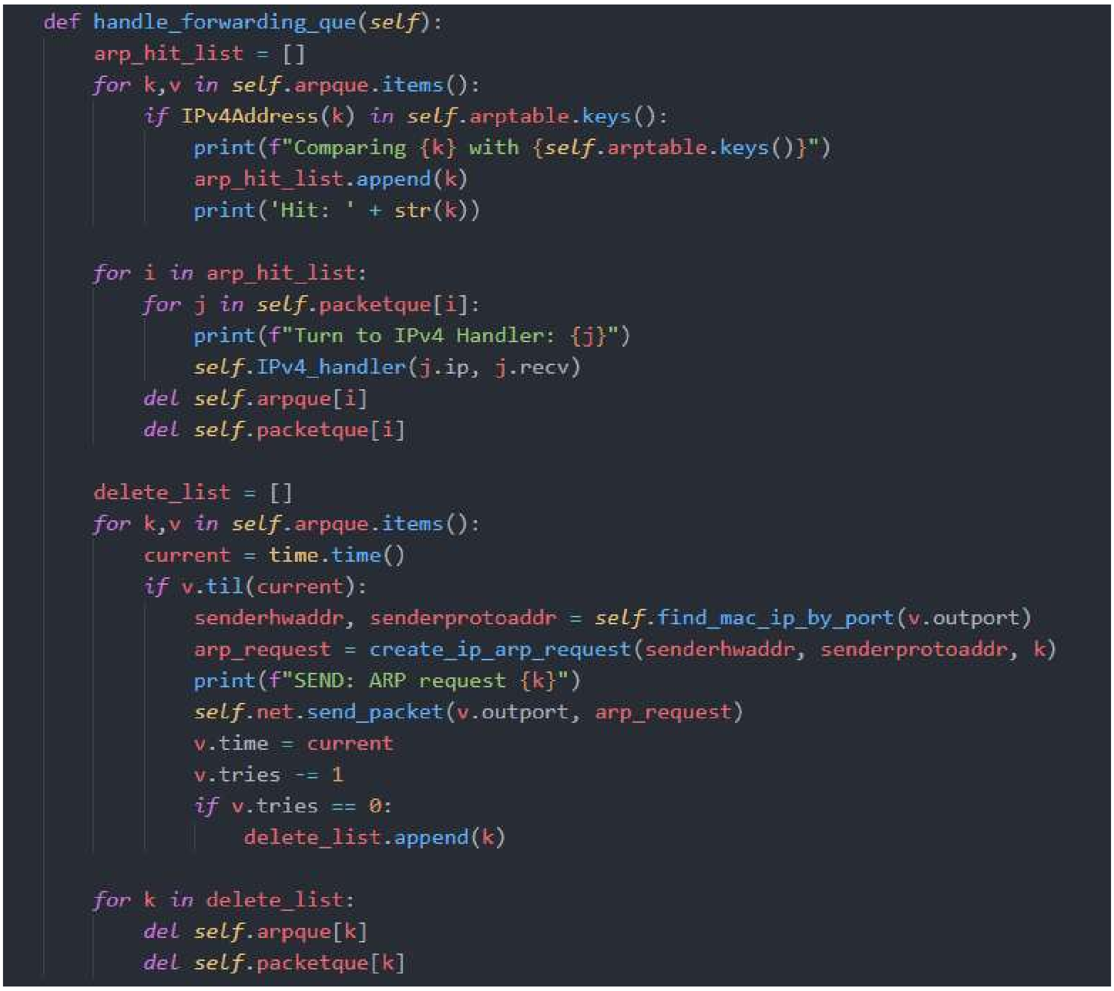
通过 ARP 查找 MAC 地址，在未命中的情况下：

1. 创建并发送代表转发表中某项(next hop)的 ARP request；
2. 创建对应 next hop 的分组 arp 请求标记和 packet 发送队列；
3. arp 请求标记中存放 arp_waiter 对象，每个 next hop 对应一个 arp_waiter，以 next hop 为键，记录了 request 剩余重发机会、上次发送时间以及出口端口名称；
4. 将对应 next hop 的分组存入 packet_waiter 队列中，packet_waiter 记录了分组的下一跳地址、出口端口名称、分组本身和 IP 包头。



```
def handle_arptable_not_found(self, next_hop, outport, ip, recv):
    already_inqueque = False
    if self.arpque:
        for k,v in self.arpque.items():
            if next_hop == k:
                already_inqueque = True
    if already_inqueque == False:
        self.arpque[next_hop] = self.arp_waiter(outport)
        self.packetque[next_hop] = []
        senderhwaddr, senderprotoaddr = self.find_mac_ip_by_port(outport)
        arp_request = create_ip_arp_request(senderhwaddr, senderprotoaddr, next_hop)
        self.net.send_packet(outport, arp_request)
        print(f"SEND: ARP request {next_hop}")
    self.packetque[next_hop].insert(0, self.packet_waiter(next_hop, outport, ip, recv))
    return 'ff-ff-ff-ff-ff-ff'
```

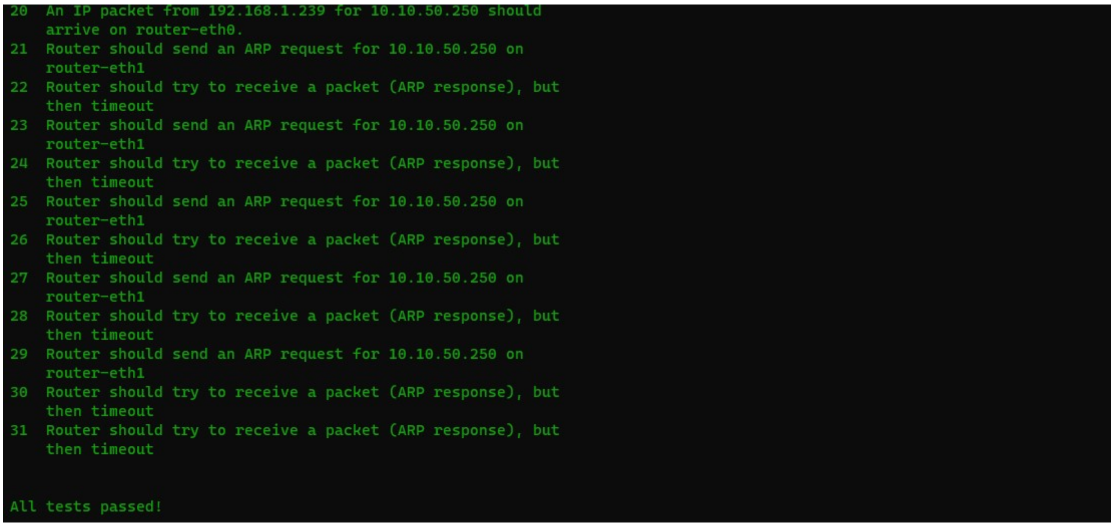
1. 每轮循环开始时，检查等待队列的下一跳 mac 地址有没有命中 arp 表；
2. 如果命中，将该 mac 地址加入命中列表，根据命中列表按顺序将排队的分组转发，删除对应等待队列；
3. 剩余 arp que 表项重发机会-1，等于 0 直接删除；
4. 剩余 arp que 中没有命中的项发送 arp 分组；



```
def handle_forwarding_que(self):
    arp_hit_list = []
    for k,v in self.arpque.items():
        if IPv4Address(k) in self.arptable.keys():
            print(f"Comparing {k} with {self.arptable.keys()}")
            arp_hit_list.append(k)
            print('Hit: ' + str(k))

    for i in arp_hit_list:
        for j in self.packetque[i]:
            print(f"Turn to IPv4 Handler: {j}")
            self.IPv4_handler(j.ip, j.recv)
        del self.arpque[i]
        del self.packetque[i]

    delete_list = []
    for k,v in self.arpque.items():
        current = time.time()
        if v.til(current):
            senderhwaddr, senderprotoaddr = self.find_mac_ip_by_port(v.outport)
            arp_request = create_ip_arp_request(senderhwaddr, senderprotoaddr, k)
            print(f"SEND: ARP request {k}")
            self.net.send_packet(v.outport, arp_request)
            v.time = current
            v.tries -= 1
            if v.tries == 0:
                delete_list.append(k)

    for k in delete_list:
        del self.arpque[k]
        del self.packetque[k]
```

In the report, show the test result of your router.

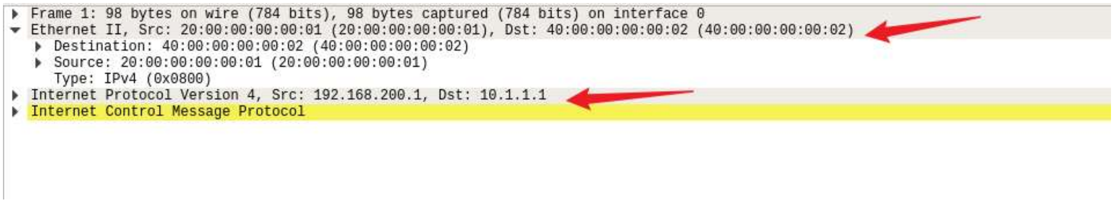
测试结果如下：



```
20  An IP packet from 192.168.1.239 for 10.10.50.250 should
    arrive on router-eth0.
21  Router should send an ARP request for 10.10.50.250 on
    router-eth1
22  Router should try to receive a packet (ARP response), but
    then timeout
23  Router should send an ARP request for 10.10.50.250 on
    router-eth1
24  Router should try to receive a packet (ARP response), but
    then timeout
25  Router should send an ARP request for 10.10.50.250 on
    router-eth1
26  Router should try to receive a packet (ARP response), but
    then timeout
27  Router should send an ARP request for 10.10.50.250 on
    router-eth1
28  Router should try to receive a packet (ARP response), but
    then timeout
29  Router should send an ARP request for 10.10.50.250 on
    router-eth1
30  Router should try to receive a packet (ARP response), but
    then timeout
31  Router should try to receive a packet (ARP response), but
    then timeout


All tests passed!
```

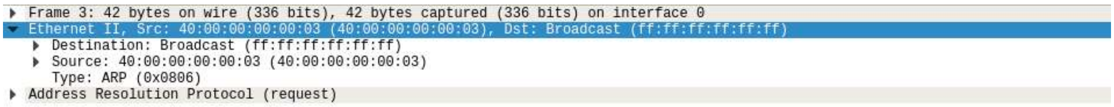
Write the procedure and analysis in your report with screenshots.

首先使用命令：server2 ping -c2 client，server2 首先向路由器的 mac 地址发送分组，目的 IP 地址是 client，如下图：

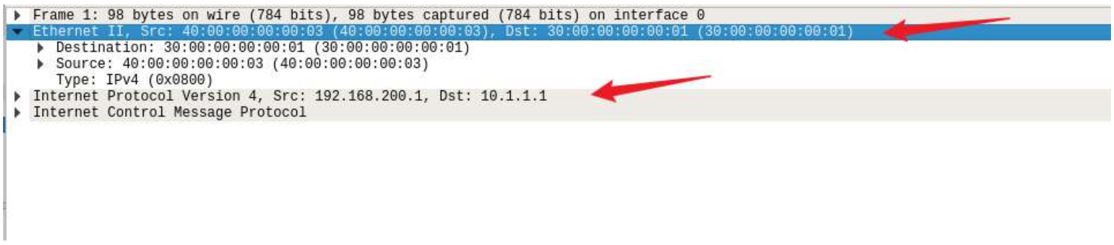


```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: 20:00:00:00:00:01 (20:00:00:00:00:01), Dst: 40:00:00:00:00:02 (40:00:00:00:00:02)
  ▸ Destination: 40:00:00:00:00:02 (40:00:00:00:00:02)
  ▸ Source: 20:00:00:00:00:01 (20:00:00:00:00:01)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.200.1, Dst: 10.1.1.1
▸ Internet Control Message Protocol
```

路由器收到之后，根据转发表，先发送 arp request 获取 client 的 mac 地址，然后在 client 对应的端口发送分组，源 mac 地址是自身接口的 mac，目的 mac 地址是 client 的 mac，IP 地址不变，如下图所示：



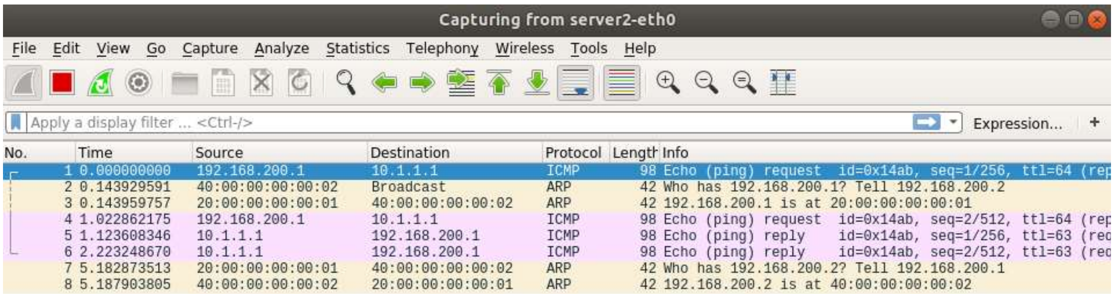
```
▸ Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▾ Ethernet II, Src: 40:00:00:00:00:03 (40:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Source: 40:00:00:00:00:03 (40:00:00:00:00:03)
    Type: ARP (0x0806)
▸ Address Resolution Protocol (request)
```



```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 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)
  ▸ Destination: 30:00:00:00:00:01 (30:00:00:00:00:01)
  ▸ Source: 40:00:00:00:00:03 (40:00:00:00:00:03)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.200.1, Dst: 10.1.1.1
▸ Internet Control Message Protocol
```

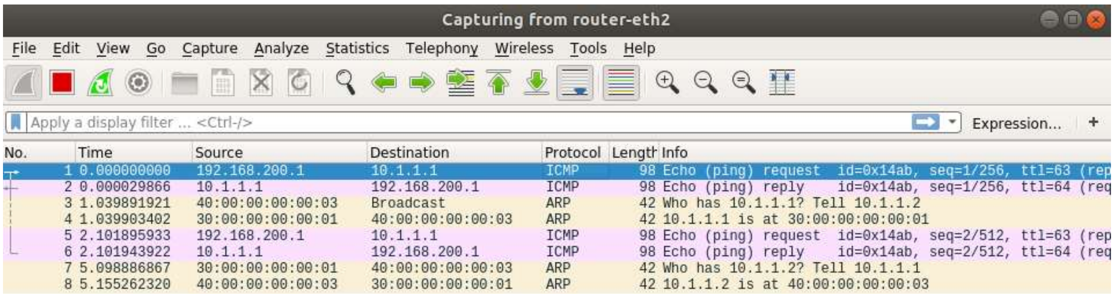
ICMP reply 即 ping 的回复分组和发送分组同理。

Server2 wireshark 抓包如下：


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	192.168.200.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x14ab, seq=1/256, ttl=64 (rep
2	0.143929591	40:00:00:00:00:02	Broadcast	ARP	42	Who has 192.168.200.1? Tell 192.168.200.2
3	0.143959757	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	192.168.200.1 is at 20:00:00:00:00:01
4	1.022862175	192.168.200.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x14ab, seq=2/512, ttl=64 (rep
5	1.123608346	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) reply id=0x14ab, seq=1/256, ttl=63 (req
6	2.223248670	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) reply id=0x14ab, seq=2/512, ttl=63 (req
7	5.182873513	20:00:00:00:00:01	40:00:00:00:00:02	ARP	42	Who has 192.168.200.2? Tell 192.168.200.1
8	5.187903805	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

Router-eth2 即路由到 cilent 端口抓包如下：


Capturing from router-eth2
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	192.168.200.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x14ab, seq=1/256, ttl=63 (rep
2	0.000029866	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) reply id=0x14ab, seq=1/256, ttl=64 (req
3	1.039891921	40:00:00:00:00:03	Broadcast	ARP	42	Who has 10.1.1.1? Tell 10.1.1.2
4	1.039903402	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	10.1.1.1 is at 30:00:00:00:00:01
5	2.101895933	192.168.200.1	10.1.1.1	ICMP	98	Echo (ping) request id=0x14ab, seq=2/512, ttl=63 (rep
6	2.101943922	10.1.1.1	192.168.200.1	ICMP	98	Echo (ping) reply id=0x14ab, seq=2/512, ttl=64 (req
7	5.098886867	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
8	5.155262320	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

5. 核心代码

6. 总结与感想

掌握了路由器处理静态转发分组的方法。