# 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

| 学院 | 计算机科学与技术系 | 学号 | 221220085 |
| --- | --- | --- | --- |
| 姓名 | 时昌军 | Email | 221220085@smail.nju.edu.cn |

## 一、实验名称

Forwarding Packets

## 二、实验目的

1. 对没有已知以太网 MAC 地址的 IP 地址发出 ARP 请求。路由器通常必须向其他主机发送数据包，并且需要MAC地址才能这样做。
2. 接收和转发到达链路并发往其他主机的数据包。转发过程的一部分是在转发信息库中执行地址查找（“最长前缀匹配”查找）。您最终将只在路由器中使用“静态”路由，而不是实现 RIP 或 OSPF 等动态路由协议。

## 三、实验内容&结果

### Task 1: 准备

### Task 2: IP 转发表查找

#### [1] 构建转发表

创建转发表，我采用的是 `list` 结构，在 `router` 类中定义为 `forwarding_table`，每个元素为 `[network address,subnet address，next_ip,interface.name]`，其创建过程定义在 `router` 类中的 `build()` 函数中。

从两个来源构建转发表:

1. 调用`net.interfaces()`

用 `interface` 接口中的 `ipaddr` 和 `netmask` 获取接口的IP和子网掩码，再用 `IPv4Network` 的构造函数就可以生成一个 `IPv4Network` 的对象。 `next_ip`不知道，设置为 `0.0.0.0`。

```
for intf in self.net.interfaces():
    network_address = ip_network(f"{intf.ipaddr}/{intf.netmask}",
strict=False).network_address
    entry = {
        'network address': network_address,
        'subnet address': IPv4Address(intf.netmask),
        'next hop address': None,
        'interface': intf.name
    }
    self.forwarding_table.append(entry)
```

2. 通过读取 `forwarding_table.txt`

其结构使得每行包含 4 个空格分隔项：**网络地址**、**子网掩码**、**下一跳地址**和转发数据包的**接口**。可用 `open()` 方法读取文件，再用 `split()` 方法分割字符串并且处理一下行末的回车。

```
with open('forwarding_table.txt', 'r') as f:
    for line in f:
        network, mask, next_hop, intf_name = line.strip().split()
        network_address = ip_network(f"{network}/{mask}",
strict=False).network_address
        entry = {
            'network address': network_address,
            'subnet address': IPv4Address(mask),
            'next hop address': IPv4Address(next_hop) if next_hop != '-'
else None,
            'interface': intf_name
        }
        self.forwarding_table.append(entry)
```

### [2] 将目标 IP 地址与转发表进行匹配

构建转发表（启动时应执行一次）后，路由器接收的 IP 数据包中的目标地址应与转发表匹配。如果表中有两个项目匹配，则应使用最长的前缀匹配。需要考虑的三种特殊情况：

1. 如果以太网目标既不是广播地址也不是传入端口的 MAC，则路由器应始终将其丢弃，而不是执行查找过程。
2. 如果表中没有匹配项，暂时删除数据包。
3. 如果数据包是针对路由器本身的（即目标地址在路由器的接口中），则只需丢弃该数据包即可。

```
best_match = None
max_prefix_length = -1
for entry in self.forwarding_table:
    network = IPv4Network(f"{entry['network address']}/{entry['subnet
address']}")
    if waiting_packet.packet.get_header(IPv4).dst in network and
network.prefixlen > max_prefix_length:
        best_match = entry
        max_prefix_length = network.prefixlen

...
ip_header = packet.get_header(IPv4)
if ip_header is None or ip_header.dst in self.interface_ips:
    return
```

## Task 3: 转发数据包和 ARP

接下来我们要为待转发的 IP 数据包创建一个以太网标头，因此我们需要知道与应将数据包转发到的主机相对应的目标以太网 MAC 地址。为此我们需要发送 ARP 查询以获取与下一跳 IP 地址对应的 以太网地址。我们的步骤如下：

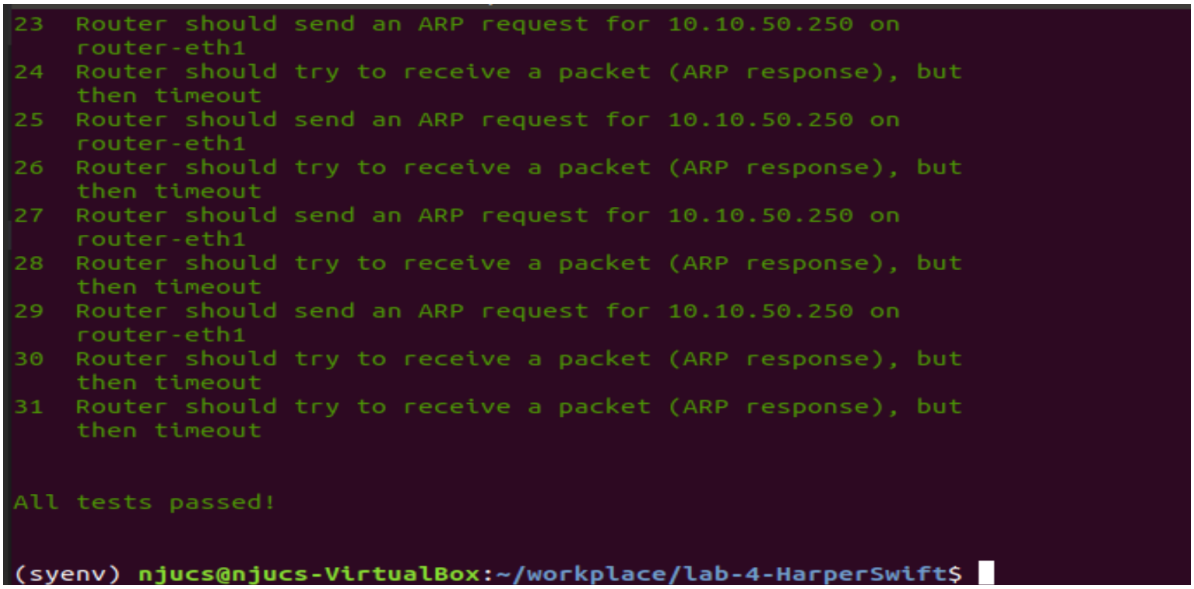
首先，为需要解析的 IP 地址（即相应以太网地址的 IP 地址） 创建并发送 ARP 请求。接着，在收到 ARP 回复后，为对应的 IP 数据包添加以太网标头，完成转发工作，同时将回复中的 IP-MAC 键值对加入到 ARP 缓存表中。如果 `arp_table` 有所更新，能够找到对应的mac地址，则将数据包发送出去，并且从等待队列中移除。但是要注意，如果一秒内没有收到相应请求的 ARP 回复，则再次发送 ARP 请求，直到同一个 IP 地址对应的发送 ARP 请求个数已经达到五个，此时放弃转发并丢弃这个数据包。

```python
def needs_arp_request(self, current_time):
    return current_time - self.last_request_time > 1 and self.request_count < 4 and current_time - 1

...

if next_hop_ip in self.ARPtable:
    dest_mac = self.ARPtable[next_hop_ip]
    ethernet_header = Ethernet(dst=dest_mac, src=self.net.interface_by_name(interface_name).ethaddr, ethertype=EtherType.IPv4)
    packet[0] = ethernet_header
    self.net.send_packet(interface_name, packet)
else:
    # MAC is unknown
    wp=WaitingPacket(packet, next_hop_ip)
    self.waiting_packets.append(wp)
    if wp.needs_arp_request(time.time()):
        last_arp_request_time[wp.dst_ip] = time.time()
        #send_arp_request
        interface=self.net.interface_by_name(interface_name)
        arp_request = create_ip_arp_request(interface.ethaddr, interface.ipaddr, next_hop_ip)
        self.net.send_packet(interface.name, arp_request)
```

**测试样例通过图：**



```
njucs@njucs-VirtualBox: ~/workplace/lab-4-HarperSwift
File  Edit  View  Search  Terminal  Help
1191Router should not do anything
1192Ping request from 31.0.5.1 should arrive on eth5
1193Ping request from 31.0.5.1 should arrive on eth5
1194Ping request from 31.0.5.1 should arrive on eth5
1195Ping request from 31.0.5.1 should arrive on eth5
1196Ping request from 31.0.5.1 should arrive on eth5
1197Ping request from 31.0.5.1 should arrive on eth5
1198Router should not do anything
1199Ping request from 31.0.6.1 should arrive on eth6
1200Ping request from 31.0.6.1 should arrive on eth6
1201Ping request from 31.0.6.1 should arrive on eth6
1202Ping request from 31.0.6.1 should arrive on eth6
1203Ping request from 31.0.6.1 should arrive on eth6
1204Ping request from 31.0.6.1 should arrive on eth6
1205Router should not do anything
1206Bonus: V2FybWluZyB1cA==
1207Bonus: V2FybWVkIHVw
1208Bonus: V2h1dCBkJyB5YSBob3BlIHQnIGZpbmQgaGVyZT8=
1209Bonus: SGFsZndheQ==
1210Bonus: Tm90aGluJyBmb3IgeWEgdCcgZmluZCBoZXJlIQ==
1211Bonus: Q29uZ3JhdHMh

All tests passed!
```

## 部署:

```
mininet> xterm router
mininet> router wireshark &
mininet> router wireshark &
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
mininet> xterm server1
mininet> server1 ping -c2 10.1.1.1
```

Capturing from router-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 | 0.055258964 | 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 | 0.055272450 | 192.168.100.1 | 10.1.1.1 | ICMP | 98 | Echo (ping) request id=0x183b, seq=1/256, ttl=64 (reply in 4) |
| 4 | 0.363425757 | 10.1.1.1 | 192.168.100.1 | ICMP | 98 | Echo (ping) reply id=0x183b, seq=1/256, ttl=63 (request in 3) |
| 5 | 1.001460905 | 192.168.100.1 | 10.1.1.1 | ICMP | 98 | Echo (ping) request id=0x183b, seq=2/512, ttl=64 (reply in 6) |
| 6 | 1.196015996 | 10.1.1.1 | 192.168.100.1 | ICMP | 98 | Echo (ping) reply id=0x183b, seq=2/512, ttl=63 (request in 5) |

```
▸ 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)
```

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

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 | 40:00:00:00:00:03 | Broadcast | ARP | 42 | Who has 10.1.1.1? Tell 10.1.1.2 |
| 2 | 0.000039495 | 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 |
| 3 | 0.103788047 | 192.168.100.1 | 10.1.1.1 | ICMP | 98 | Echo (ping) request id=0x183b, seq=1/256, ttl=63 (reply in 4) |
| 4 | 0.103849894 | 10.1.1.1 | 192.168.100.1 | ICMP | 98 | Echo (ping) reply id=0x183b, seq=1/256, ttl=64 (request in 3) |
| 5 | 0.936161267 | 192.168.100.1 | 10.1.1.1 | ICMP | 98 | Echo (ping) request id=0x183b, seq=2/512, ttl=63 (reply in 6) |
| 6 | 0.936233744 | 10.1.1.1 | 192.168.100.1 | ICMP | 98 | Echo (ping) reply id=0x183b, seq=2/512, ttl=64 (request in 5) |
| 7 | 5.158822182 | 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 | 5.223786799 | 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 |

```
▸ Frame 1: 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)
▸ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 40 00  00 00 00 03 08 06 00 01   ......@. ........
0010  08 00 06 04 00 01 40 00  00 00 00 03 0a 01 01 02   ......@. ........
0020  ff ff ff ff ff ff 0a 01  01 01                     ........ ..
```

其中 eth2 端发送了一个 ARP 请求，这是由于路由器收到 server1 发给 client 的数据包时，不知道 client 的 MAC 地址导致的，当它收到 来自 client 的 ARP 回复时，会将这个数据包正确的转发给 client。

# 四、实验心得

写的时候好难，好坐牢，好折磨，但是写出来好开心。