



南开大学  
Nankai University

南 开 大 学

计 算 机 学 院

网络技术与应用实验报告

---

## 通过编程获取 IP 地址与 MAC 地址的对应关系

---

李佩诺

年级：2020 级

专业：信息安全

指导教师：张建忠

2022 年 11 月 19 日

# 目录

<b>一、 实验内容说明</b>	<b>1</b>
<b>二、 实验准备</b>	<b>1</b>
(一) 项目配置 . . . . .	1
(二) 查看 IP 与 MAC 对应关系 . . . . .	1
(三) ARP 协议 . . . . .	2
1. ARP 帧 . . . . .	2
2. ARP 过程解析 . . . . .	2
<b>三、 实验过程</b>	<b>3</b>
(一) 项目设计思路 . . . . .	3
(二) 关键代码分析 . . . . .	4
1. 遍历设备存储信息并显示 . . . . .	4
2. 找到输入 IP 所在的网卡 . . . . .	4
(三) 获取 MAC . . . . .	5
1. 伪造 ARP . . . . .	6
2. 发包 . . . . .	7
3. 收包并解析 . . . . .	7
4. 实验结果展示 . . . . .	8
<b>四、 特殊现象分析</b>	<b>9</b>
(一) 发包后接收不到的问题 . . . . .	9
(二) pcap_sendpacket 发包失败返回 31 问题 . . . . .	9

## 一、 实验内容说明

通过编程获取 IP 地址与 MAC 地址的对应关系实验，要求如下：

1. 在 IP 数据报捕获与分析编程实验的基础上，学习 WinPcap 的数据包发送方法。
2. 通过 Npcap 编程，获取 IP 地址与 MAC 地址的映射关系。
3. 程序要具有输入 IP 地址，显示输入 IP 地址与获取的 MAC 地址对应关系界面。界面可以是命令行界面，也可以是图形界面，但应以简单明了的方式在屏幕上显示。
4. 编写的程序应结构清晰，具有较好的可读性。

## 二、 实验准备

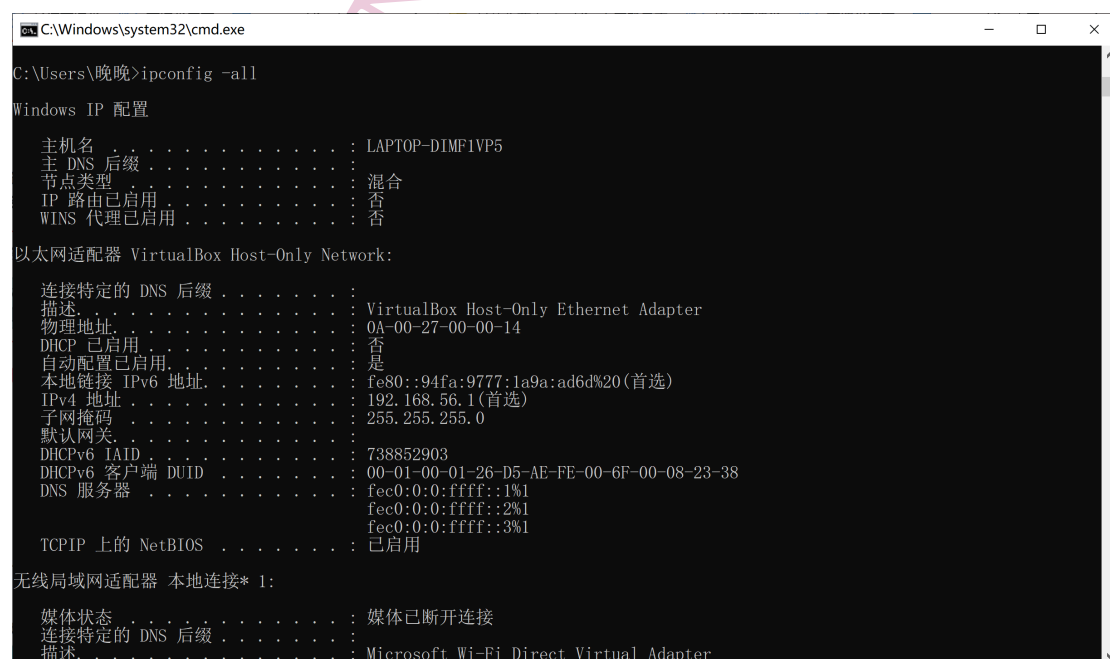
### (一) 项目配置

本次实验需要安装 Npcap1.71、npcap-sdk-1.13, 在成功安装后需要对 Visual Studio 的项目属性配置：

- 在 C/C++ 中添加附加包含目录：D:\npcap\Include;
- 在预处理器中添加预处理器定义：WPCAP;HAVE\_REMOTE;
- 在链接器-> 常规中添加附加库目录：D:\npcap\Lib;
- 在链接器-> 输入中添加附加依赖项：Packet.lib;wpcap.lib;

### (二) 查看 IP 与 MAC 对应关系

在 cmd 输入 ipconfig, 查看本机网卡与 IP 的对应关系：



```
C:\Windows\system32\cmd.exe
C:\Users\晚晚>ipconfig -all

Windows IP 配置

   主机名 . . . . . : LAPTOP-DIMF1VP5
   主 DNS 后缀 . . . . . :
   节点类型 . . . . . : 混合
   IP 路由已启用 . . . . . : 否
   WINS 代理已启用 . . . . . : 否

以太网适配器 VirtualBox Host-Only Network:

   连接特定的 DNS 后缀 . . . . . :
   描述. . . . . : VirtualBox Host-Only Ethernet Adapter
   物理地址. . . . . : 0A-00-27-00-00-14
   DHCP 已启用 . . . . . : 否
   自动配置已启用 . . . . . : 是
   本地链接 IPv6 地址. . . . . : fe80::94fa:9777:1a9a:ad6d%20(首选)
   IPv4 地址 . . . . . : 192.168.56.1(首选)
   子网掩码 . . . . . : 255.255.255.0
   默认网关. . . . . :
   DHCPv6 IAID . . . . . : 738852903
   DHCPv6 客户端 DUID . . . . . : 00-01-00-01-26-D5-AE-FE-00-6F-00-08-23-38
   DNS 服务器 . . . . . : fec0:0:0:ffff::1%1
                           fec0:0:0:ffff::2%1
                           fec0:0:0:ffff::3%1
   TCPIP 上的 NetBIOS . . . . . : 已启用

无线局域网适配器 本地连接* 1:

   媒体状态 . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . :
   描述. . . . . : Microsoft Wi-Fi Direct Virtual Adapter
```

图 1: IP 与 MAC 对应关系

### (三) ARP 协议

在理论课的学习中,了解到 IP 地址只是一个逻辑地址,如果要想在链路层进行数据传播,需要知道 MAC 物理地址,而 IP 数据包中只有 IP 地址,所以 ARP 协议的作用是**通过 IP 地址获取对应 MAC 地址**。

#### 1. ARP 帧

ARP 请求会发送下图格式的帧:

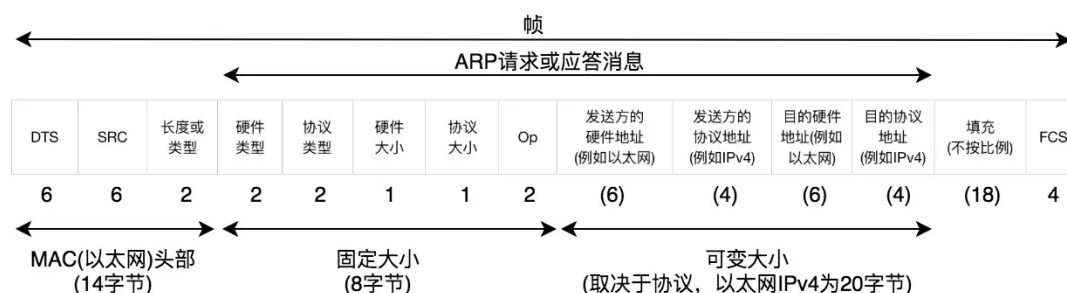


图 2: ARP 帧

前 14 字节是以太网帧, 里面包含发送帧的源 MAC 地址和目的 MAC 地址以及帧类型, 当目的 MAC 地址为 FFFFFFFF 时, 表示发送的是广播帧。

ARP 帧占 28 字节, 其中的关键字段有 OP (表示操作类型, 1 为请求, 2 为应答)、发送端 MAC、发送端 IP、目的 MAC (本次实验中获得)、目的 IP。

#### 2. ARP 过程解析

假设有主机 A 与主机 B 在同一网段, 主机 A (IP1、MAC1), 主机 B (IP2、MAC2), 主机 A 想要与主机 B 建立连接。

主机 A 首先查看自己的 ARP 缓存表中是否存在 ip2 对应的 MAC2, 如果缓存表中存在直接返回该地址, 如果不存在, 主机 A 发送 **ARP 广播帧**, 其中源 MAC 地址为 MAC1, **目的 MAC 地址为 FFFFFFFF**, OP 为 1, 发送端 MAC 为 MAC1, 发送端 IP 为 ip1, 目标 IP 为 ip2, **目标 mac 为空** (即要获得的 MAC2 地址的值)。

该帧会广播发送到该网段的**所有主机**, 当主机收到帧的时候, 识别到是 ARP 请求后, **会比较自己的 IP 与 ARP 中的目的 IP**, 如果不一致则丢弃这个包, 如果一致, 也就是主机 B 收到这个请求后, 首先将 ARP 信息中的发送端 IP 和发送端 MAC 加入自己的 ARP 缓存中, 然后主机 B 会发送一个 **ARP 应答帧**给主机 A, 这个 ARP 帧**填入了主机 B 的 IP2 和 MAC2**, 待这个 ARP 应答包到达主机 A 时, 主机 A 就可以获取到主机 B 的 IP 地址和 MAC 地址的对应关系, 然后存入 ARP 缓存中。

### 三、 实验过程

#### (一) 项目设计思路

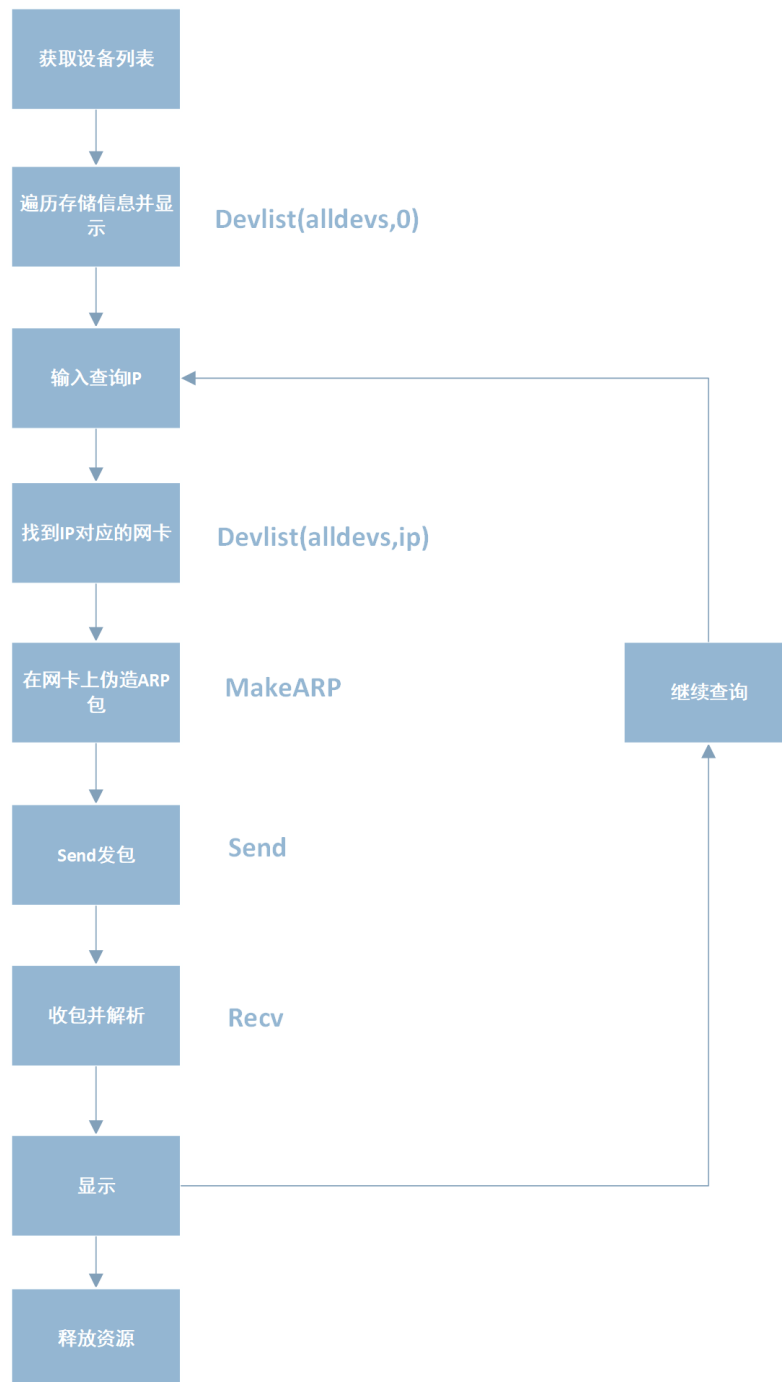


图 3: 实验过程和相关函数

## (二) 关键代码分析

### 1. 遍历设备存储信息并显示

Devslis 函数，当第二个参数为 0 时，遍历获取到的设备列表，并且其 IP 信息存到 iplist 中，方便之后的查询。

遍历

```

1 //遍历接口列表
2 void Devslis(pcap_if_t* alldevs, int ip_i) {
3 //ip_i=0 全部显示
4 //ip_i!=0 找特定ip
5 i = 0;
6 for (pcap_if_t* d = alldevs; d != nullptr; d = d->next) //显示接口列表
7 {
8 //获取该网络接口设备的ip地址信息
9 for (pcap_addr_t* a = d->addresses; a != nullptr; a = a->next)
10 {
11     if (((struct sockaddr_in*)a->addr)->sin_family == AF_INET && a->addr)
12     { //打印ip地址
13         i++;
14         if (ip_i == 0)
15         {
16             //打印相关信息
17             //inet_ntoa将ip地址转成字符串格式
18             printf("%d\n", i);
19             printf("%s\t\t%s\n", "name:", d->name, "
                description:", d->description);
20             printf("%s\t\t%s\n", "IP地址:", inet_ntoa(((
                structsockaddr_in*)a->addr)->sin_addr));
21             iplist[i] = inet_ntoa(((structsockaddr_in*)a->addr)->
                sin_addr);
22         }
23         else { .... //寻找特定IP }
24     }
25 }
}

```

### 2. 找到输入 IP 所在的网卡

当用户输入一个 IP 后，会根据 IP 遍历查询它在 iplist 中的位置，该位置即该 IP 所对应的网卡：

遍历寻找 ip 对应的网卡

```

1 //输入想查询的IP地址
2 string ip;
3 cout << "-----" << endl << "请输入需要查询的
    ip地址: ";
4 while (cin >> ip) {
5     int ip_i;

```

```

6      for (int j = 1; j <= i; j++) {
7          //在iplist里找到了
8          if (iplist[j] == ip) {
9              ip_i = j;
10             break;
11         }
12         //没找到
13         else if (j == i && ip != iplist[j]) { cout << "输入错误";
14             return 0; }
15     }
16     //查询输入ip并显示所有信息
17     Devslist(alldevs, ip_i);
18     cout << "-----" << endl;
19     cout << "请输入需要查询的ip地址: ";
20 }

```

找到之后，将所对应的网卡的序号存入 ip\_i，将 ip\_i 作为第二个参数传入 Devslist 函数，此时就不会对整个设备列表进行遍历，而是去寻找 ip\_i 网卡。

#### 输入 IP 查看 MAC 功能

```

1 void Devslist(pcap_if_t* alldevs, int ip_i) {
2     //ip_i=0 全部显示
3     //ip_i!=0 找特定ip
4     ...
5     //输入IP查看MAC功能
6     else if(ip_i==i){
7         printf("%s\t%s\n%s\t%s\n", "name:", d->name, "description:", d->
            description);
8         printf("%s\t%s\n", "IP地址:", inet_ntoa(((structsockaddr_in*)a->addr)
            ->sin_addr));
9         //在当前网卡上伪造一个包
10        ARPFrame_t ARPFrame = MakeARP(a);
11        //打开该网卡的网络接口
12        adhandle = pcap_open(d->name, 655340, PCAP_OPENFLAG_PROMISCUOUS, 1000, 0,
            0);
13        if (adhandle == NULL) { cout << "打开接口失败"; }
14        //发包
15        if (Send(adhandle, ARPFrame) == 0) { break; };
16        //收包
17        Recv(adhandle);
18    }
19    ...}

```

### (三) 获取 MAC

刚刚我们已经找到了输入 IP 所在的网卡，接下来要获取该网卡的 MAC 地址，根据 ARP 协议，我们应该发送一个 ARP 广播帧（以太网帧目的 MAC 为 FFFFFFFF），其中目的 ip 是所在网卡的 ip，目的 MAC 应为空，源 MAC、发送端 MAC、发送端 IP 任意填写，我们需

要让网卡接收到这个 ARP 广播帧后将自己的 MAC 填入 ARP 应答包，我们再去抓这个包，从而可以获得网卡 IP 与 MAC 地址的对应关系。

### 1. 伪造 ARP

首先先定义好 ARP 帧和以太网帧的结构体：

以太网帧和 ARP 请求/应答消息

```

1 #pragma pack (1) //进入字节对齐方式
2 //以太网帧 14字节
3 typedef struct FrameHeader_t {
4     BYTE DesMAC[6]; // 目的地址
5     BYTE SrcMAC[6]; //源地址
6     WORD FrameType; //帧类型
7 }FrameHeader_t;
8 //ARP请求/应答 28字节
9 typedef struct ARPFrame_t {
10     FrameHeader_t FrameHeader; //以太网帧头
11     WORD HardwareType; //硬件类型
12     WORD ProtocolType; //协议类型
13     BYTE HLen; //硬件地址长度
14     BYTE PLen; //协议地址长度
15     WORD Operation;
16     BYTE SendHa[6]; //发送端以太网地址
17     DWORD SendIP; //发送端IP地址
18     BYTE RecvHa[6]; //目的以太网地址
19     DWORD RecvIP; //目的IP地址
20 } ARPFrame_t;
21 #pragma pack ()

```

通过 MakeARP() 函数完成 ARP 包的伪造，其中，如果要获取本机 IP 地址对应的 MAC 地址，应伪造广播帧（以太网帧目的 MAC 为 FFFFFFFF），其中目的 ip 是所在网卡的 ip，目的 MAC 应为空，源 MAC、发送端 MAC、发送端 IP 任意填；

如果要获取远程 IP 地址的 IP，应伪造的 ARP 包的源 MAC 和发送端 MAC 硬填入所在网卡的 MAC 地址，发送端 IP 也应该填写所在网卡的 IP，目的 IP 应填写远程 IP 地址，以太网帧目的 MAC 地址依然填写 FFFFFFFF，ARP 请求中的 MAC 地址为空。以本机 IP 地址的包为例子，伪造 ARP 包代码如下：

ARP 伪造

```

1 ARPFrame_t MakeARP(pcap_addr* a) {
2     ARPFrame_t ARPFrame;
3     for (int i = 0; i < 6; i++)
4         ARPFrame.FrameHeader.DesMAC[i] = 0xff; //表示广播
5     //将ARPFrame.FrameHeader.SrcMAC设置为本机网卡的MAC地址
6     for (int i = 0; i < 6; i++)
7         ARPFrame.FrameHeader.SrcMAC[i] = 0x0f;
8
9     ARPFrame.FrameHeader.FrameType = htons(0x806); //帧类型为ARP

```



```

10     ARPFrame.HardwareType = htons(0x0001); //硬件类型为以太网
11     ARPFrame.ProtocolType = htons(0x0800); //协议类型为IP
12     ARPFrame.HLen = 6; //硬件地址长度为6
13     ARPFrame.PLen = 4; //协议地址长为4
14     ARPFrame.Operation = htons(0x0001); //操作为ARP请求
15
16     //获取远程IP与MAC对应时应该将ARPFrame.SendHa设置为本机网卡的MAC地址
17     for (int i = 0; i < 6; i++)
18         ARPFrame.SendHa[i] = 0x0f;
19     //获取远程IP与MAC对应时应该将ARPFrame.SendIP设置为本机网卡上绑定的
20     IP地址
21     ARPFrame.SendIP = inet_addr("100.100.100.100");
22     //获取远程IP与MAC对应时应该将ARPFrame.RecvHa设置为0
23     for (int i = 0; i < 6; i++)
24         ARPFrame.RecvHa[i] = 0; //表示目的地址未知
25     //获取远程IP与MAC对应时应该将ARPFrame.RecvIP设置为请求的IP地址
26     ARPFrame.RecvIP = inet_addr(inet_ntoa(((struct sockaddr_in*)a->addr)
27         ->sin_addr));
28     return ARPFrame;
29 }

```

## 2. 发包

发送数据包直接用 pcap\_sendpacket() 函数, 如果返回值为 0 即代表发包成功, 在接收到包之后, 本机网卡或远程网卡都会发送一个 ARP 应答包, 下面我们要做的就是收到并解析这个 ARP 应答包。

## 3. 收包并解析

打开网卡之后在网卡上发送广播帧后, 需要进行收包, 此时对所有收到的包进行筛选, 需要得到刚刚我们伪造 IP 地址的数据包, 所以需要增加筛选条件 `*(unsigned long*)(pkt_data + 38) == inet_addr("伪造 ip 地址")` 和 `*(unsigned short*)(pkt_data + 12) == htons(0x0806)`, 确保收到的是我们刚刚伪造并发送的 ARP 数据包。

收到包后需要对其进行解析, 我们直接打印出收到包中的 FrameHeader.SrcMAC 的值, 所有相关代码如下:

### 收包并解析

```

1 //收包
2 void Recv(pcap_t* adhandle) {
3     struct pcap_pkthdr* pkt_header;
4     const u_char* pkt_data;
5     int res;
6     while ((res = pcap_next_ex(adhandle, &pkt_header, &pkt_data)) >= 0){
7         ARPFrame_t* RecPacket = (ARPFrame_t*)pkt_data;
8         if (
9             *(unsigned short*)(pkt_data + 12) == htons(0x0806)
10            //0x0806为以太网帧类型表示后面数据的类型, 对于ARP请求或应答来
            说, 该字段的值为x0806

```

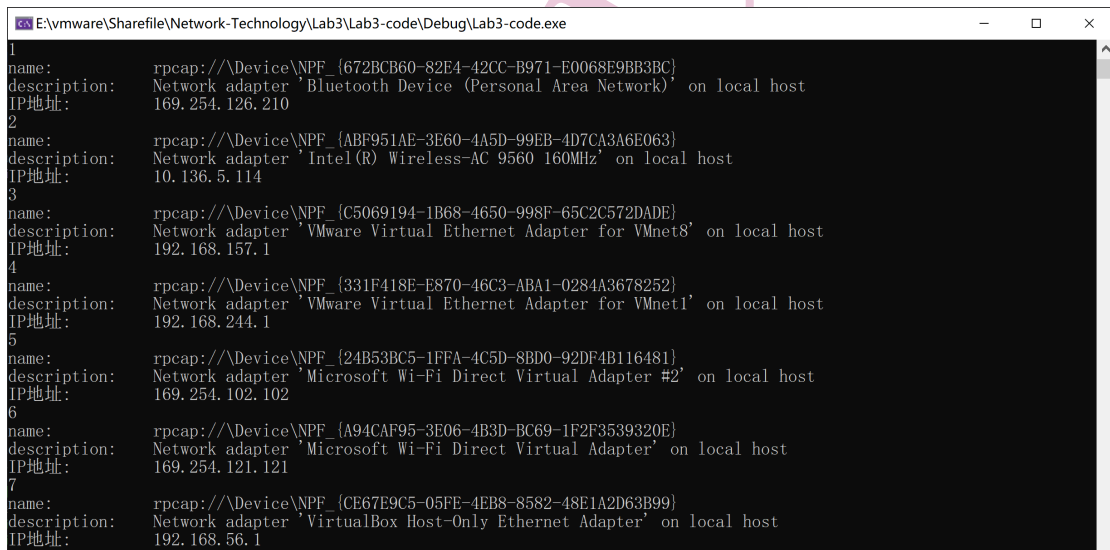
```

11         && *((unsigned short*)(pkt_data + 20)) == htons(2) //ARP应答
12         && *((unsigned long*)(pkt_data + 38)) == inet_addr("
           100.100.100.100")
13     )
14     {
15         printf("%s:\t%02x-%02x-%02x-%02x-%02x-%02x\n", "MAC地址",
16             RecPacket->FrameHeader.SrcMAC[0],
17             RecPacket->FrameHeader.SrcMAC[1],
18             RecPacket->FrameHeader.SrcMAC[2],
19             RecPacket->FrameHeader.SrcMAC[3],
20             RecPacket->FrameHeader.SrcMAC[4],
21             RecPacket->FrameHeader.SrcMAC[5]);
22         break;
23     }
24 }

```

#### 4. 实验结果展示

##### 1. 获取本机网卡与 IP 列表



```

E:\vmware\Sharefile\Network-Technology\Lab3\Lab3-code\Debug\Lab3-code.exe
1
name:                rpcap://\Device\NPF_{672BCB60-82E4-42CC-B971-E0068E9BB3BC}
description:         Network adapter 'Bluetooth Device (Personal Area Network)' on local host
IP地址:              169.254.126.210
2
name:                rpcap://\Device\NPF_{ABF951AE-3E60-4A5D-99EB-4D7CA3A6E063}
description:         Network adapter 'Intel(R) Wireless-AC 9560 160MHz' on local host
IP地址:              10.136.5.114
3
name:                rpcap://\Device\NPF_{C5069194-1B68-4650-998F-65C2C572DADE}
description:         Network adapter 'VMware Virtual Ethernet Adapter for VMnet8' on local host
IP地址:              192.168.157.1
4
name:                rpcap://\Device\NPF_{331F418E-E870-46C3-ABA1-0284A3678252}
description:         Network adapter 'VMware Virtual Ethernet Adapter for VMnet1' on local host
IP地址:              192.168.244.1
5
name:                rpcap://\Device\NPF_{24B53BC5-1FFA-4C5D-8BD0-92DF4B116481}
description:         Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host
IP地址:              169.254.102.102
6
name:                rpcap://\Device\NPF_{A94CAF95-3E06-4B3D-BC69-1F2F3539320E}
description:         Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter' on local host
IP地址:              169.254.121.121
7
name:                rpcap://\Device\NPF_{CE67E9C5-05FE-4EB8-8582-48E1A2D63B99}
description:         Network adapter 'VirtualBox Host-Only Ethernet Adapter' on local host
IP地址:              192.168.56.1

```

图 4: 获取本机网卡与 IP 列表

##### 2. 获取 IP 与 MAC 地址的对应关系

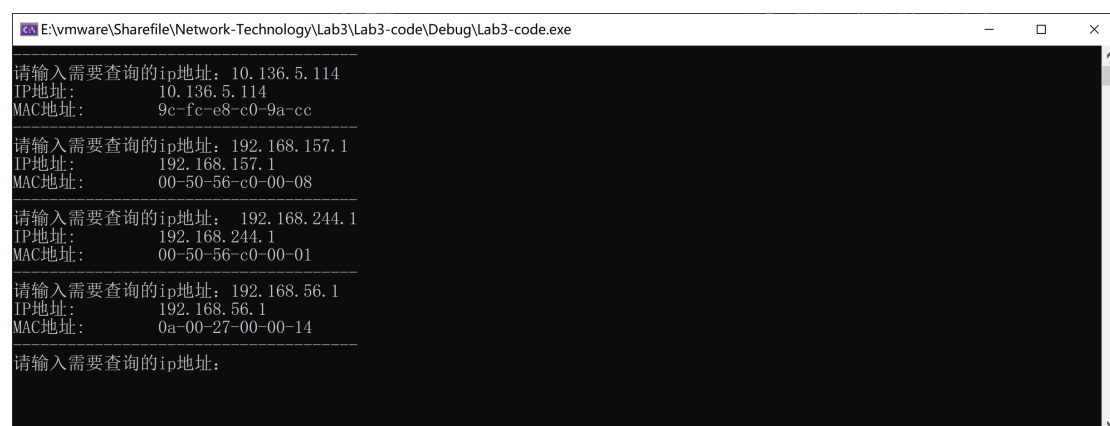


图 5: 获取 IP 与 MAC 地址的对应关系

## 四、特殊现象分析

### (一) 发包后接收不到的问题

在初次试验代码能否成功时, 打开了蓝牙网卡, 但是发包成功后并未接收到应答包, 用 Wireshark 查看情况, 发现网卡能收到自己伪造的应答包:

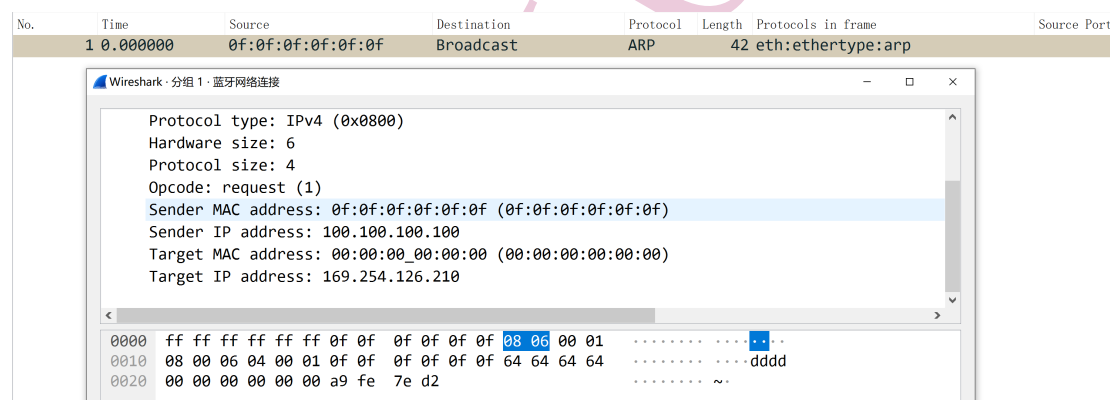


图 6: Wireshark 查看

最后发现是由于网卡未打开的原因, 更换打开的网卡就可以收到了。

以太网适配器 蓝牙网络连接:

```
媒体状态 . . . . . : 媒体已断开连接
连接特定的 DNS 后缀 . . . . . :
```

图 7: 蓝牙网卡并未打开

### (二) pcap\_sendpacket 发包失败返回 31 问题

在成功打开本机网卡之后, 发包时会出现 pcap\_sendpacket 发包失败返回 31 的问题, 含义是“连到系统上的设备没有发挥作用”:

```
请输入需要查询的ip地址: 10.136.5.114
IP地址: 10.136.5.114
返回错误码:31
```

图 8: 31 问题

但是如果用 Wireshark 进行抓包却可以抓到:

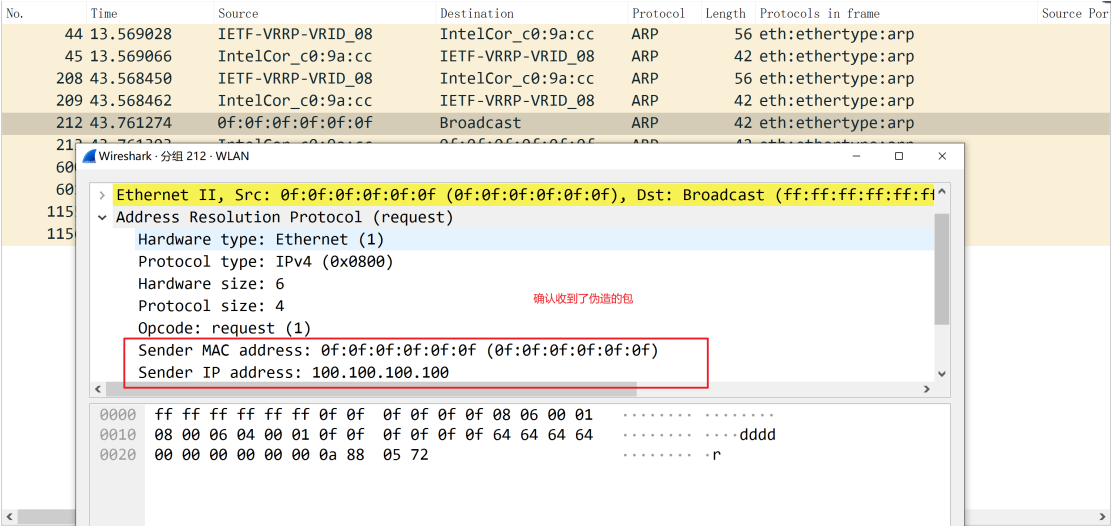


图 9: Wireshark 抓到了 arp 请求包

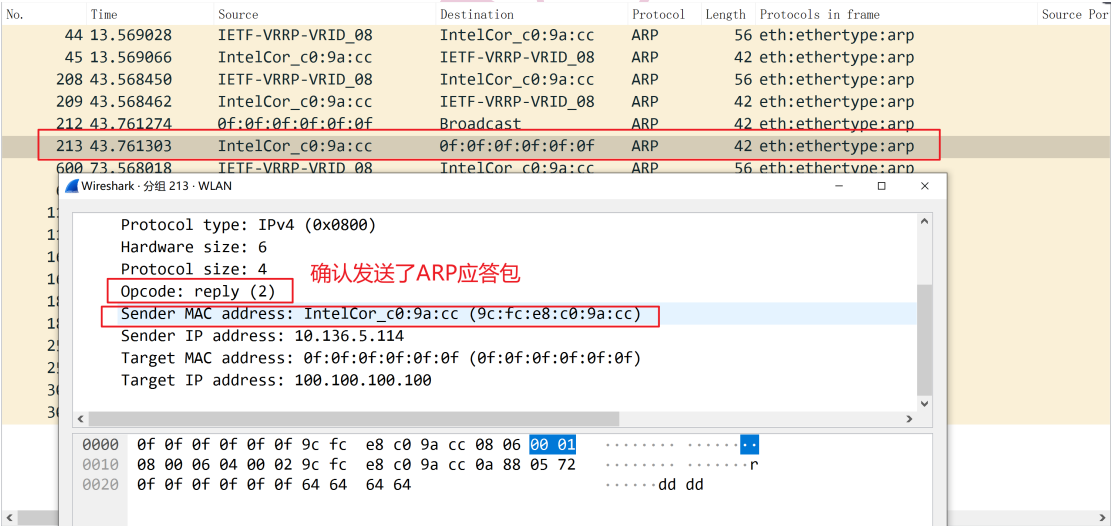


图 10: Wireshark 抓到了 arp 应答包

所以按照 Wireshark 的抓包结果, 应该是可以抓到这个 ARP 应答包并且可以解析出请求 IP 地址对应的 MAC 地址的, 我们可以选择先忽略这个返回值问题, 继续往下进行:

```
-----  
请输入需要查询的ip地址: 10.136.5.114  
IP地址:      10.136.5.114  
MAC地址:     9c-fc-e8-c0-9a-cc  
-----
```

图 11: 重新抓包结果

实验代码已放进Github:<https://github.com/Wanwan0407/Network-Technology/tree/main/Lab3>

NIKU