

# 网络技术与应用编程实验（2）报告

学号：2012482    姓名：董伊萌    年级：20级    专业：信息安全

## 一.实验内容

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.实验前期准备

1. 安装WinPcap驱动程序和DLL程序。
2. 创建基于WinPcap的应用程序，主要包括：
  - 添加pcap.h包含文件；
  - 增加与WinPcap有关的预处理器定义：将标号WPCAP和HAVE\_REMOTE添加到预处理器定义中；也可以通过
  - 添加包含文件目录：将pcap.h所在的Include文件夹添加到IDE中；
  - 添加wpcap.lib库文件：将wpcap.lib所在的Lib文件夹添加到IDE中。
3. 学习WinPcap的设备列表获取方法、网卡设备打开方法、数据包捕获方法以及向网络发送数据包的方法，应用在编程中。

## 三.实验过程

### 1. 应用的主要数据结构

- pcap\_if\_t：网络接口信息

```
1 struct pcap_if
2 {
3     struct pcap_if *next;           /*多个网卡时使用来显示各个网卡的信息*/
4     char *name;                     /* name to hand to "pcap_open_live()" */
5     char *description;              /*就是网卡的型号、名字等*/
6     struct pcap_addr *addresses;    //pcap_addr 结构体 地址信息
7     bpf_u_int32 flags;              /* PCAP_IF_ interface flags 接口标志*/
8 };
9
```

调用WinPcap的pcap\_findalldevs\_ex()函数后，参数alldevs指向的链表中包含了主机中安装的网络设备列表。

pcap\_addr结构体如下：

```
1 struct pcap_addr
2 {
3     struct pcap_addr *next;
4     struct sockaddr *addr;           /* address  IP地址 */
5     struct sockaddr *netmask;       /* netmask for that address 子网掩码*/
6     struct sockaddr *broadaddr;     /* broadcast address for that address */
7     struct sockaddr *dstaddr;       /* P2P destination address for that address */
8 };
9
```

- 帧首部 FrameHeader\_t

```
1 typedef struct FrameHeader_t {
2     BYTE DesMAC[6]; //源MAC地址
3     BYTE SrcMAC[6]; //目的MAC地址
4     WORD FrameType; //帧类型
5 }FrameHeader_t;
6
```

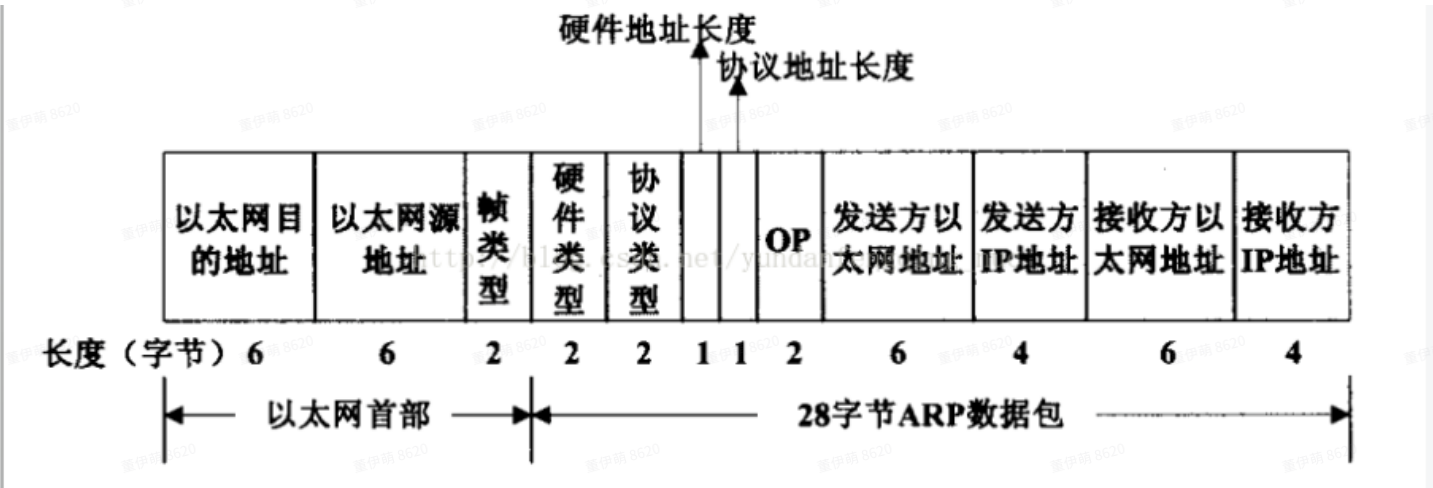
包含48位源MAC地址，源目的MAC地址，。帧类型

- ARP帧 ARPFrame\_t

```
1 typedef struct ARPFrame_t {
2     FrameHeader_t FrameHeader; //帧首部
3     WORD HardwareType; //硬件类型
4     WORD ProtocolType; //协议类型
5     BYTE HLen; //硬件地址长度
6     BYTE PLen; //协议地址长度
7     WORD Operation; //操作类型
8     BYTE SendHa[6]; //源MAC地址
9     DWORD SendIP; //源IP地址
10    BYTE RecvHa[6]; //目的MAC地址
11    DWORD RecvIP; //目的IP地址
12 }ARPFrame_t
```

- const u\_char \*pkt\_data

是传递的参数 指的是ARP数据包中的除头外的数据内容,其结构如下所示：



所以pkt\_data + 12指向的是帧类型，如果为ARP的话，应为 htons(0x0806)，pkt\_data + 22指向的是发送方的MAC地址，长度为6，pkt\_data + 28指向的是发送方的IP地址，长度为4，pkt\_data + 32指向

的是接收方的以太网地址，pkt\_data + 38指向的是接收方的IP地址。

- Struct sockaddr结构

```
1 struct sockaddr
2 {
3     sa_family_t sa_family;    //所选协议族AF_INET
4     char        sa_data[14]; //ip地址及端口号
5 }
```

- sockaddr\_in结构

```
1 struct sockaddr_in {
2     short int sin_family; /* Address family */
3     unsigned short int sin_port; /* Port number */
4     struct in_addr sin_addr; /* Internet address */
5     unsigned char sin_zero[8]; /* Same size as struct sockaddr */
6 };
7 sin_family: 指代协议族，在socket编程中只能是AF_INET
8 sin_port: 存储端口号（使用网络字节顺序）
9 sin_addr: 存储IP地址，使用in_addr这个数据结构
10 sin_zero: 是为了让sockaddr与sockaddr_in两个数据结构保持大小相同而保留的空字节。
```

## 2. 程序主要思路 and 重要代码部分

首先简述一下本次实验程序的实验思路：

- 首先获取本机网络接口和接口上绑定的IP地址

这里主要用到了自定义函数 pcap\_if\_t\* CAPLIST()，其中调用WinPcap的函数 pcap\_findalldevs () 函数，

并根据 pcap\_if\_t 和 pcap\_addr 结构获取网络设备列表及其绑定的IP相关信息

- 选择发送数据包的网卡，发送ARP请求，请求本机网络接口上绑定的IP地址和MAC地址的对应关系，本地主机模拟一个远端主机，发送一个ARP请求报文，该请求报文请求本机网络接口上绑定的IP地址与MAC地址的对应关系。

这里首先根据上一步骤中获取的本机网络接口和接口上绑定的IP地址选择发送数据包的网卡，打开用户选择设备的网卡，获取该网络接口卡绑定的IP地址存储到ip数组中。

然后设置要发送的ARP数据包 ARPF\_Send，为了获取以太网中其他主机的IP地址与MAC地址，需要向以太网广播ARP请求，设置ARP数据包中相关内容。

最后利用 pcap\_sendpacket() 函数向网络发送数据包。

- 程序捕获本机的ARP响应，获取本机网络接口卡的MAC地址。

这里主要是利用 pcap\_next\_ex() 函数，pkt\_data指向捕获到的网络数据包，判断是否为正确的ARP响应，如果正确就输出ARP数据包中的内容，获取本机MAC地址。

- 得到本机网络接口的MAC地址和其上绑定的IP地址后，程序可以组装和发送ARP请求报文，请求以太网中其他主机的IP地址与MAC地址的对应关系。

主要利用的函数为 pcap\_next\_ex，pcap\_sendpacke t输入目的主机的IP地址，发送构造好的数据包并进行数据包捕获，获取目的主机的MAC地址。

然后说明一下程序中使用的重要的函数：

## ●自定义函数：

`string CoutIp(unsigned long u) :`

对IP输出格式的更改，调用函数 `inet_ntoa` 将internet地址结构转换成以“.”间隔的诸如“a.b.c.d”的字符串形式。

```
1 string CoutIp(unsigned long u) {
2     in_addr addr;
3     memcpy(&addr, &u, sizeof(u));
4     return inet_ntoa(addr);
5 }
6
```

`string* Byte2Hex(unsigned char bArray[], int bArray_len) :`

将地址由BYTE形式转换为16进制字符串类型，便于后续程序运行结果进行比较。

```
1 string* Byte2Hex(unsigned char bArray[], int bArray_len)
2 {
3     string* strHex = new string();
4     int nIndex = 0;
5     for (int i = 0; i < bArray_len; i++)
6     {
7         char hex1;
8         char hex2;
9         int value = bArray[i];
10        int S = value / 16;
11        int Y = value % 16;
12        if (S >= 0 && S <= 9)
13            hex1 = (char)(48 + S);
14        else
15            hex1 = (char)(55 + S);
16        if (Y >= 0 && Y <= 9)
17            hex2 = (char)(48 + Y);
18        else
19            hex2 = (char)(55 + Y);
20        if (i != bArray_len - 1) {
21            *strHex = *strHex + hex1 + hex2 + "-";
22        }
23        else
24            *strHex = *strHex + hex1 + hex2;
25    }
26
27    return strHex;
28 }
```

`void* get_in_addr(struct sockaddr* sa) :`

返回要转换为字符串的网络字节中的 IP 地址的指针。

```
1 void* get_in_addr(struct sockaddr* sa)
2 {
3     //判断一下是否为IP
4     if (sa->sa_family == AF_INET)
5         return &(((struct sockaddr_in*)sa)->sin_addr);
```

```

6     return &(((struct sockaddr_in6*)sa)->sin6_addr);
7 }

```

**void ARP\_show(struct pcap\_pkthdr\* header, const u\_char\* pkt\_data) :**

输出捕获的APR包中的相关信息，主要包括源MAC地址、源IP地址、目的MAC地址、目的IP地址。

```

1 void ARP_show(struct pcap_pkthdr* header, const u_char* pkt_data)
2 {
3     struct ARPFrame_t* arp_protocol;
4     arp_protocol = (struct ARPFrame_t*)(pkt_data);
5
6     cout << "源MAC地址:  " << *(Byte2Hex(arp_protocol->FrameHeader.SrcMAC, 6)) <
7     cout << "源IP地址:  " << CoutIp(arp_protocol->SendIP) << endl;
8     cout << "目的MAC地址: " << *(Byte2Hex(arp_protocol->FrameHeader.DesMAC, 6)) <
9     cout << "目的IP地址 " << CoutIp(arp_protocol->RecvIP) << endl;
10    cout << endl;
11 }

```

**pcap\_if\_t\* CAPLIST()**

获取本机网络接口的MAC地址和IP地址，这里主要调用了函数pcap\_findalldevs

(PCAP\_SRC\_IF\_STRING, NULL, &alldevs, errbuf)，alldevs指向的链表包含主机中安装的网络接口设备列表，在此要进行一下判断，pcap\_findalldevs () 函数的返回值如果等于-1，则说明获取失败，打印错误信息，函数返回。如果获取成功，通过利用alldevs指针，进行循环遍历所有的网络接口，打印每个网络接口设备的名字和描述信息。并针对每一个网络设备接口，循环遍历绑定的IP信息，对IP地址，网络掩码和广播地址进行输出。

```

1 pcap_if_t* CAPLIST() {
2     pcap_if_t* alldevs;    //指向设备链表首部的指针
3     pcap_if_t* d;
4     pcap_addr_t* a;
5     int        n = 1;
6     char        errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
7
8     //获取本机的设备列表
9     //调用pcap_findalldevs () 函数, alldevs指向的链表包含主机中安装的网络接口设备列表
10    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1)
11    {
12        cout << stderr << "Error in pcap_findalldevs_ex:" << errbuf << endl;
13        return 0;
14    }
15
16    //显示接口列表
17
18    for (d = alldevs; d != NULL; d = d->next)
19    {
20
21        cout << n++ << "." << d->name;
22        if (d->description)
23            cout << "(" << d->description << ")" << endl;
24        else
25            cout << "(No description )\n";
26        //获取该网络接口的IP地址信息
27        for (a = d->addresses; a != NULL; a = a->next) {

```

```

28 //判断该地址是否为IP地址
29 if (a->addr->sa_family == AF_INET) {
30     //输出网络接口卡上绑定的多个IP地址的相关信息
31     char str[INET_ADDRSTRLEN];
32     inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), str,
33     cout << "IP地址: " << str << endl;
34     inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->netmask), st
35     cout << "网络掩码: " << str << endl;
36     inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->broadaddr),
37     cout << "广播地址: " << str << endl;
38
39     }
40 }
41 }
42 if (n == 0)
43 {
44     cout << "\nERROR!\n";
45
46 }
47 return alldevs;//pcap_findalldevs_ex函数调用成功后, alldevs参数指向获取的网络接口
48 }
49

```

**void SET\_ARP\_Frame\_HOST(ARPFrame\_t &ARPFrame1, char ip[INET\_ADDRSTRLEN]) :**

设置在本地主机模拟一个远端主机，发送一个ARP请求报文时，对ARP组装报文的设置构造要发送的APR数据包，此时源MAC地址字段和源IP地址字段需要使用虚假的MAC地址和IP地址，所以在这里SrcMac、SendHa、SendIP随意设置；RecvHa设置为0；RecvIP设置为本机的IP地址（这里由函数的参数 char ip[INET\_ADDRSTRLEN] 提供，协议类型为IP，硬件类型为以太网、硬件地址长度为6、协议地址长度为4、操作为ARP请求。

```

1 void SET_ARP_Frame_HOST(ARPFrame_t &ARPFrame1, char ip[INET_ADDRSTRLEN]) {
2     for (int i = 0; i < 6; i++) {
3         ARPFrame1.FrameHeader.DesMAC[i] = 0xff;
4         ARPFrame1.FrameHeader.SrcMAC[i] = 0x0f;
5         ARPFrame1.SendHa[i] = 0x0f;
6         ARPFrame1.RecvHa[i] = 0x00;
7     }
8
9     ARPFrame1.FrameHeader.FrameType = htons(0x0806);
10    ARPFrame1.HardwareType = htons(0x0001);
11    ARPFrame1.ProtocolType = htons(0x0800);
12    ARPFrame1.HLen = 6;
13    ARPFrame1.PLen = 4;
14    ARPFrame1.Operation = htons(0x0001);
15    ARPFrame1.SendIP = inet_addr("10.10.10.10");
16    ARPFrame1.RecvIP = inet_addr(ip);
17 }

```

**void SET\_ARP\_Frame\_DEST(ARPFrame\_t& ARPFrame , char ip[INET\_ADDRSTRLEN], unsigned char\*mac, unsigned char\*desmac) :**



请求以太网中其他主机的IP地址与MAC地址的对应关系时对ARP请求报文的组装，DesMAC设置为广播地址；SrcMAC和SendHa为本机MAC地址，用获得的本机AC地址（在此为参数unsigned char\*mac）填充；RecvHa设置为0；SendIP为本机的IP地址（在此为参数ip[INET\_ADDRSTRLEN]）；RecvIP设置为目的主机的IP地址，由用户输入（在此函数中为参数unsigned char\*desmac）；其他关于类型的设置均与上面相同。

```
1 void SET_ARP_Frame_DEST(ARPFrame_t& ARPFrame , char ip[INET_ADDRSTRLEN], unsigned
2     for (int i = 0; i < 6; i++) {
3         ARPFrame.FrameHeader.DesMAC[i] = 0xff;
4         ARPFrame.RecvHa[i] = 0x00;
5         ARPFrame.FrameHeader.SrcMAC[i] = mac[i]; // 设置为本网卡的MAC地址
6         ARPFrame.SendHa[i] = mac[i]; // 设置为本网卡的MAC地址
7     }
8     ARPFrame.FrameHeader.FrameType = htons(0x0806);
9     ARPFrame.HardwareType = htons(0x0001);
10    ARPFrame.ProtocolType = htons(0x0800);
11    ARPFrame.HLen = 6;
12    ARPFrame.PLen = 4;
13    ARPFrame.Operation = htons(0x0001);
14    ARPFrame.SendIP = inet_addr(ip);
15
16 }
```

### 3. 代码展示

程序所有代码如下所示：

```
1 #define WIN32
2 #define HAVE_REMOTE
3 #include "pcap.h"
4 #include <iostream>
5 #include <WinSock2.h>
6 #include <bitset>
7 #include <process.h>
8 using namespace std;
9 #pragma comment(lib, "wpcap.lib")
10 #pragma comment(lib, "ws2_32.lib")
11 #pragma warning(disable:4996)
12
13
14 // IP输出格式更改
15 string CoutIp(unsigned long u) {
16     in_addr addr;
17     memcpy(&addr, &u, sizeof(u));
18     return inet_ntoa(addr);
19 }
20
21 // 将地址由BYTE形式转换为16进制字符串类型
22 string* Byte2Hex(unsigned char bArray[], int bArray_len)
```

```

23 {
24     string* strHex = new string();
25     int nIndex = 0;
26     for (int i = 0; i < bArray_len; i++)
27     {
28         char hex1;
29         char hex2;
30         int value = bArray[i];
31         int S = value / 16;
32         int Y = value % 16;
33         if (S >= 0 && S <= 9)
34             hex1 = (char)(48 + S);
35         else
36             hex1 = (char)(55 + S);
37         if (Y >= 0 && Y <= 9)
38             hex2 = (char)(48 + Y);
39         else
40             hex2 = (char)(55 + Y);
41         if (i != bArray_len - 1) {
42             *strHex = *strHex + hex1 + hex2 + "-";
43         }
44         else
45             *strHex = *strHex + hex1 + hex2;
46     }
47
48     return strHex;
49 }
50
51
52 //指向要转换为字符串的网络字节中的 IP 地址的指针
53 void* get_in_addr(struct sockaddr* sa)
54 {
55     //判断一下是否为IP
56     if (sa->sa_family == AF_INET)
57         return &(((struct sockaddr_in*)sa)->sin_addr);
58     return &(((struct sockaddr_in6*)sa)->sin6_addr);
59 }
60
61 #pragma pack(1)
62 #define BYTE unsigned char
63
64 //帧首部
65 typedef struct FrameHeader_t {
66     BYTE DesMAC[6]; //源MAC地址
67     BYTE SrcMAC[6]; //目的MAC地址
68     WORD FrameType; //帧类型
69 }FrameHeader_t;
70
71 //ARP帧
72 typedef struct ARPFrame_t {
73     FrameHeader_t FrameHeader; //帧首部
74     WORD HardwareType; //硬件类型
75     WORD ProtocolType; //协议类型
76     BYTE HLen; //硬件地址长度
77     BYTE PLen; //协议地址长度
78     WORD Operation; //操作类型
79     BYTE SendHa[6]; //源MAC地址
80     DWORD SendIP; //源IP地址
81     BYTE RecvHa[6]; //目的MAC地址

```



```

82     DWORD RecvIP; //目的IP地址
83 }ARPFrame_t;
84
85
86 #pragma pack()
87 ARPFramet ARPFrame; //要发送的APR数据包(其他主机)
88 ARPFramet ARPFSend; //要发送的APR数据包(本机)
89 unsigned char mac[48], desmac[48]; //目的主机和其他主机的mac
90 pcap_t* choosed_dev; //选择的网络接口
91
92 void ARP_show(struct pcap_pkthdr* header, const u_char* pkt_data)
93 {
94     struct ARPFramet* arp_protocol;
95     arp_protocol = (struct ARPFramet*)(pkt_data);
96
97     cout << "源MAC地址: " << *(Byte2Hex(arp_protocol->FrameHeader.SrcMAC, 6)) <
98     cout << "源IP地址: " << CoutIp(arp_protocol->SendIP) << endl;
99     cout << "目的MAC地址: " << *(Byte2Hex(arp_protocol->FrameHeader.DesMAC, 6)) <
100    cout << "目的IP地址 " << CoutIp(arp_protocol->RecvIP) << endl;
101    cout << endl;
102 }
103
104 //获取本机网络接口的MAC地址和IP地址
105 pcap_if_t* CAPLIST() {
106     pcap_if_t* alldevs; //指向设备链表首部的指针
107     pcap_if_t* d;
108     pcap_addr_t* a;
109     int n = 1;
110     char errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
111
112     //获取本机的设备列表
113     //调用pcap_findalldevs()函数, alldevs指向的链表包含主机中安装的网络接口设备列表
114     if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1)
115     {
116         cout << stderr << "Error in pcap_findalldevs_ex:" << errbuf << endl;
117         return 0;
118     }
119
120
121     //显示接口列表
122
123     for (d = alldevs; d != NULL; d = d->next)
124     {
125         cout << n++ << "." << d->name;
126         if (d->description)
127             cout << "(" << d->description << ")" << endl;
128         else
129             cout << "(No description )\n";
130         //获取该网络接口的IP地址信息
131         for (a = d->addresses; a != NULL; a = a->next) {
132             //判断该地址是否为IP地址
133             if (a->addr->sa_family == AF_INET) {
134                 //输出网络接口卡上绑定的多个IP地址的相关信息
135                 char str[INET_ADDRSTRLEN];
136                 inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), str,
137                 cout << "IP地址: " << str << endl;
138                 inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->netmask), st
139                 cout << "网络掩码: " << str << endl;
140                 inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->broadaddr),

```

```

141         cout << "广播地址: " << str << endl;
142
143     }
144 }
145 }
146 if (n == 0)
147 {
148     cout << "\nERROR!\n";
149     // return 0;
150 }
151 return alldevs; //pcap_findalldevs_ex函数调用成功后, alldevs参数指向获取的网络接口
152 }
153
154
155 void SET_ARP_Frame_HOST(ARPFrame_t &ARPFrame1, char ip[INET_ADDRSTRLEN]) {
156     for (int i = 0; i < 6; i++) {
157         ARPFrame1.FrameHeader.DesMAC[i] = 0xff;
158         ARPFrame1.FrameHeader.SrcMAC[i] = 0x0f;
159         ARPFrame1.SendHa[i] = 0x0f;
160         ARPFrame1.RecvHa[i] = 0x00;
161     }
162
163     ARPFrame1.FrameHeader.FrameType = htons(0x0806);
164     ARPFrame1.HardwareType = htons(0x0001);
165     ARPFrame1.ProtocolType = htons(0x0800);
166     ARPFrame1.HLen = 6;
167     ARPFrame1.PLen = 4;
168     ARPFrame1.Operation = htons(0x0001);
169     ARPFrame1.SendIP = inet_addr("10.10.10.10");
170     ARPFrame1.RecvIP = inet_addr(ip);
171 }
172 void SET_ARP_Frame_DEST(ARPFrame_t& ARPFrame , char ip[INET_ADDRSTRLEN], unsigned
173     for (int i = 0; i < 6; i++) {
174         ARPFrame.FrameHeader.DesMAC[i] = 0xff;
175         ARPFrame.RecvHa[i] = 0x00;
176         ARPFrame.FrameHeader.SrcMAC[i] = mac[i]; //设置为本机网卡的MAC地址
177         ARPFrame.SendHa[i] = mac[i]; //设置为本机网卡的MAC地址
178     }
179     ARPFrame.FrameHeader.FrameType = htons(0x0806);
180     ARPFrame.HardwareType = htons(0x0001);
181     ARPFrame.ProtocolType = htons(0x0800);
182     ARPFrame.HLen = 6;
183     ARPFrame.PLen = 4;
184     ARPFrame.Operation = htons(0x0001);
185     ARPFrame.SendIP = inet_addr(ip);
186
187 }
188
189
190 int main() {
191
192     pcap_if_t* alldevs; //指向设备链表首部的指针
193     pcap_if_t* d;
194     pcap_addr_t* a;
195
196     char errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
197     alldevs = CAPLIST();
198     cout << "-----
199

```

```

200 //设备链表首部的指针
201 d = alldevs;
202
203 int j;
204 cout << "请选择发送数据包的网卡: ";
205 cin >> j;
206 int i = 0;
207 //获取指向选择发送数据包网卡的指针
208
209 while (i < j - 1) {
210     i++;
211     d = d->next;
212 }
213
214
215 //打开用户选择设备的网卡
216 choosed_dev = pcap_open(d->name, 100, PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL,
217
218 if (choosed_dev == NULL) {
219     cout << "Error in pcap_open: " << errbuf << endl;
220     //失败就释放设备列表;
221     pcap_freealldevs(alldevs);
222     return 0;
223 }
224 //保存网卡的ip地址 (指向缓冲区的指针, 用于存储 IP 地址的 NULL 终止字符串表示形式。)
225 char ip[INET_ADDRSTRLEN];
226
227
228 for (a = d->addresses; a != NULL; a = a->next) {
229     //判断该地址是否为IP地址
230     if (a->addr->sa_family == AF_INET) {
231         //InetNtop 函数将 IPv4 或 IPv6 Internet 网络地址转换为采用 Internet 标准格式
232         inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), ip, sizeof
233     }
234 }
235 cout << ip;
236 cout << endl << d->name << endl;
237
238 //获取本机的MAC地址
239
240 //设置ARP帧相关
241 SET_ARP_Frame_HOST(ARPF_Send, ip);
242
243 struct pcap_pkthdr* pkt_header;
244 const u_char* pkt_data;
245 struct pcap_pkthdr* header = new pcap_pkthdr;
246 int k;
247 //发送构造好的数据包
248 //用pcap_next_ex()捕获数据包, pkt_data指向捕获到的网络数据包
249 while ((k = pcap_next_ex(choosed_dev, &pkt_header, &pkt_data)) >= 0) {
250     //发送数据包
251
252     pcap_sendpacket(choosed_dev, (u_char*)&ARPF_Send, sizeof(ARPFFrame_t));
253     struct ARPFrame_t* arp_message;
254     arp_message = (struct ARPFrame_t*)(pkt_data);
255     if (k == 0) continue;
256     else
257     {
258         if(arp_message->FrameHeader.FrameType==htons(0x0806)&& arp_messag

```

```

259         cout << "ARP数据包: \n";
260         ARP_show(header, pkt_data); //打印相应的信息
261         //用MAC地址记录本机的MAC地址, 用于后续构造ARP数据包
262         for (int i = 0; i < 6; i++) {
263             mac[i] = *(unsigned char*)(pkt_data + 22 + i);
264         }
265         cout << "获取自己主机的MAC地址成功, 本机MAC地址为: " << *(Byte2Hex(
266             break;
267     }
268 }
269 }
270
271 if (k < 0) {
272     cout << "Error in pcap_next_ex." << endl;
273 }
274 cout << "-----"
275
276 //设置ARP帧
277
278 SET_ARP_Frame_DEST(ARPFrame, ip, mac, desmac);
279
280 cout << "请输入目的主机的IP地址: ";
281 char desip[INET_ADDRSTRLEN];
282 cin >> desip;
283 ARPFrame.RecvIP = inet_addr(desip); //设置为请求的IP地址
284
285 while ((k = pcap_next_ex(choosed_dev, &pkt_header, &pkt_data)) >= 0) {
286     //pcap_sendpacket () 发送构造好的数据包
287
288     pcap_sendpacket(choosed_dev, (u_char*)&ARPFrame, sizeof(ARPFrame_t));
289     struct ARPFrame_t* arp_message;
290     arp_message = (struct ARPFrame_t*)(pkt_data);
291     if (k == 0) continue;
292     else
293         if (arp_message->FrameHeader.FrameType == htons(0x0806) && arp_messa
294             cout << "ARP数据包: \n";
295             ARP_show(header, pkt_data);
296             for (int i = 0; i < 6; i++) {
297                 //记录得到的目的主机的MAC地址
298                 desmac[i] = *(unsigned char*)(pkt_data + 22 + i);
299             }
300             cout << "获取目的主机的MAC地址成功, 目的主机的MAC地址为: " << *(Byte2Hex(
301                 break;
302         }
303     }
304     pcap_freealldevs(alldevs);
305
306 }
307
308
309
310

```

## 四.实验运行结果

获取设备列表

```
C:\Users\baxia\source\repos\网络技术与应用作业二\Debug\网络技术与应用作业二.exe

1.rpcap://\Device\NPF_{E941206A-BA06-4926-SDFC-EA6574117B13} (Network adapter 'WAN Miniport (Network Monitor)' on local host)
2.rpcap://\Device\NPF_{EC846750-C96E-43FF-B12C-D9EADEBB9C3A} (Network adapter 'WAN Miniport (IPv6)' on local host)
3.rpcap://\Device\NPF_{91344B84-04F6-443F-AA87-9FEBF58B839A} (Network adapter 'WAN Miniport (IP)' on local host)
4.rpcap://\Device\NPF_{A2FFA96F-C397-410C-85A8-E552502394A8} (Network adapter 'Bluetooth Device (Personal Area Network)' on local host)
IP地址: 169.254.54.163
网络掩码: 255.255.0.0
广播地址: 169.254.255.255
5.rpcap://\Device\NPF_{39BC0F55-EBB2-4B46-AD85-D387B72E7A02} (Network adapter 'Intel(R) Wireless-AC 9462' on local host)
IP地址: 192.168.43.133
网络掩码: 255.255.255.0
广播地址: 192.168.43.255
6.rpcap://\Device\NPF_{0FD07C5E-B154-4EE6-9BD5-9E2C90A787D8} (Network adapter 'VMware Virtual Ethernet Adapter for VMnet8' on local host)
IP地址: 192.168.227.1
网络掩码: 255.255.255.0
广播地址: 192.168.227.255
7.rpcap://\Device\NPF_{56A1075C-5697-4F5B-B5BE-F7C649BB2AF5} (Network adapter 'VMware Virtual Ethernet Adapter for VMnet1' on local host)
IP地址: 192.168.111.1
网络掩码: 255.255.255.0
广播地址: 192.168.111.255
8.rpcap://\Device\NPF_{553FB601-1CFF-493F-8370-6A61658EB3CE} (Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host)
IP地址: 169.254.37.66
网络掩码: 255.255.0.0
广播地址: 169.254.255.255
9.rpcap://\Device\NPF_{7E046905-F329-480C-8578-C0214CE3663F} (Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter' on local host)
```

获取本机IP地址与MAC地址的映射关系

```
选择 命令提示符

默认网关. . . . . : 192.168.227.254
DHCP 服务器 . . . . . : 654331990
DHCPv6 IAID . . . . . : 00-01-00-01-27-FC-D8-C1-AC-67-5D-A2-97-7F
DHCPv6 客户端 DUID . . . . . : 192.168.227.2
主 WINS 服务器 . . . . . : 已启用
TCP/IP 上的 NetBIOS . . . . . : 已启用

无线局域网适配器 WLAN:

连接特定的 DNS 后缀 . . . . . : Intel(R) Wireless-AC 9462
描述 . . . . . : AC-67-5D-A2-97-7F
物理地址. . . . . : DHCP 已启用. . . . . : 是
自动配置已启用. . . . . : 是
IPv6 地址 . . . . . : 2408:8411:6038:e507:c09:c89d:bb22:6784(首选)
临时 IPv6 地址. . . . . : 2408:8411:6038:e507:6db0:3f43:cd4a:feb9(首选)
本地链接 IPv6 地址. . . . . : fe80::c09:c89d:bb22:6784%9(首选)
IPv4 地址 . . . . . : 192.168.43.133(首选)
子网掩码 . . . . . : 255.255.255.0
获得租约的时间 . . . . . : 2022年11月10日 20:58:23
租约过期的时间 . . . . . : 2022年11月10日 21:58:23
默认网关. . . . . : fe80::24e0:8319:a17b:f72f%9
192.168.43.1
DHCP 服务器 . . . . . : 192.168.43.1
DHCPv6 IAID . . . . . : 95184733
DHCPv6 客户端 DUID . . . . . : 00-01-00-01-27-FC-D8-C1-AC-67-5D-A2-97-7F
DNS 服务器 . . . . . : 192.168.43.1
TCP/IP 上的 NetBIOS . . . . . : 已启用

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 . . . . . : Bluetooth Device (Personal Area Network)
描述 . . . . . : AC-67-5D-A2-97-83
物理地址. . . . . : DHCP 已启用. . . . . : 是
自动配置已启用. . . . . : 是

C:\Users\baxia>
```

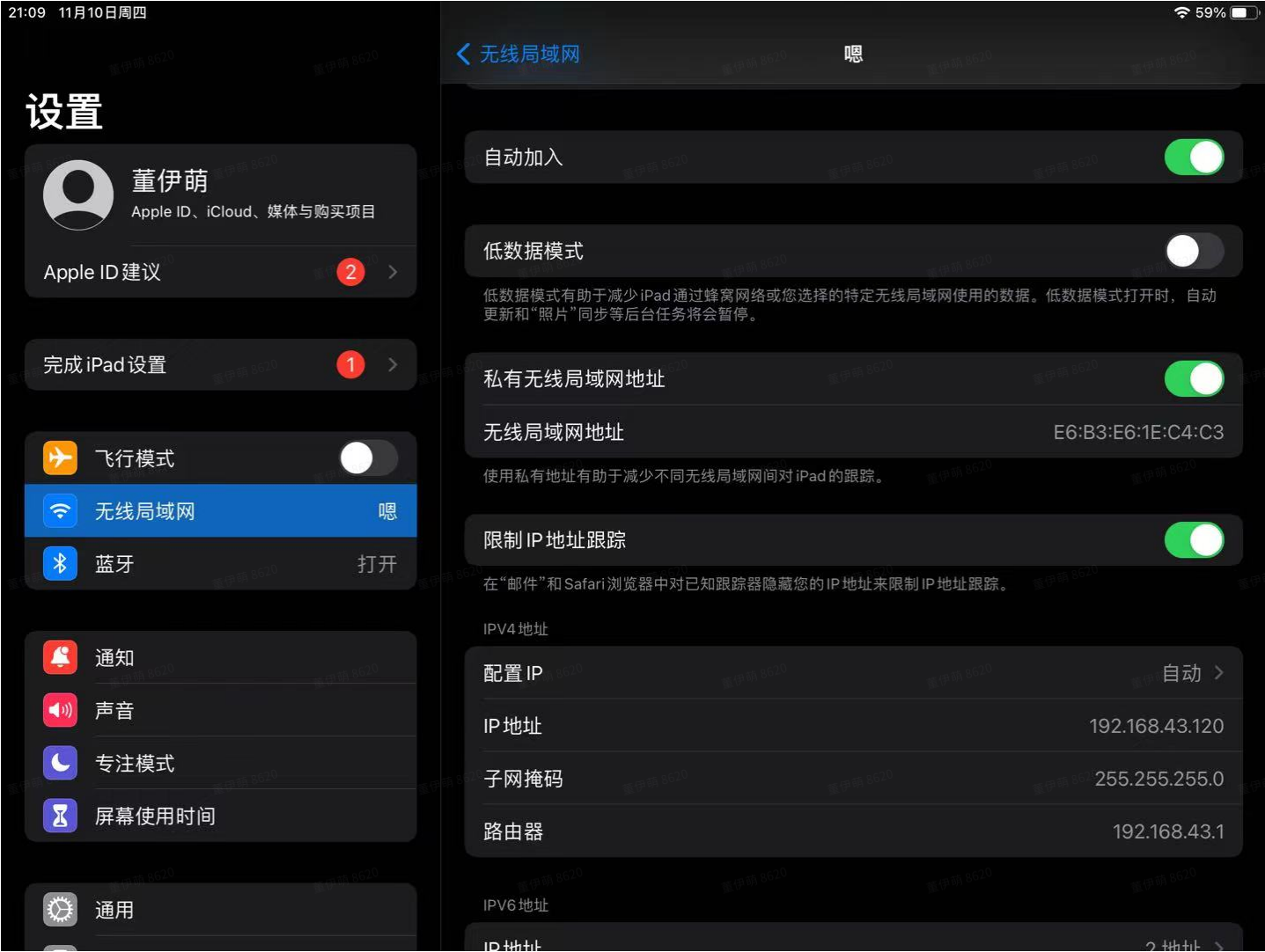
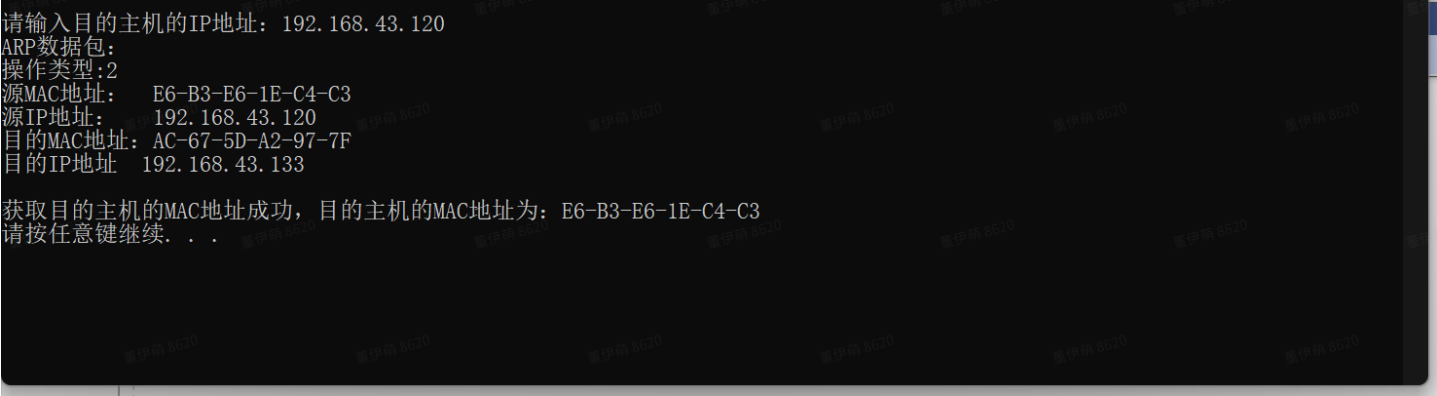
先用命令行找到无线局域网适配器WLAN的物理地址如上图：AC-67-5D-A2-97-7F

在设备列表中找到对应的IP地址，选择这个为发送数据包的网卡，结果如下：

```
请选择发送数据包的网卡: 5
192.168.43.133
rpcap://\Device\NPF_{39BC0F55-EBB2-4B46-AD85-D387B72E7A02}
ARP数据包:
源MAC地址: AC-67-5D-A2-97-7F
源IP地址: 192.168.43.133
目的MAC地址: 0F-0F-0F-0F-0F-0F
目的IP地址 10.10.10.10

获取自己主机的MAC地址成功，本机MAC地址为: AC-67-5D-A2-97-7F
```

本机和ipad分别连接手机热点，结果如下：







对照后发现物理地址和IP地址是匹配的，程序运行成功。