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

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

```

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：

```
struct pcap_addr
{
        struct pcap_addr *next;
        struct sockaddr *addr;                  /* address  IP地址 */
        struct sockaddr *netmask;         /* netmask for that address  子网掩码*/
        struct sockaddr *broadaddr;          /* broadcast address for that addre
        struct sockaddr *dstaddr;          /* P2P destination address for that a
};

```

- 帧首部 FrameHeader_t

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

```

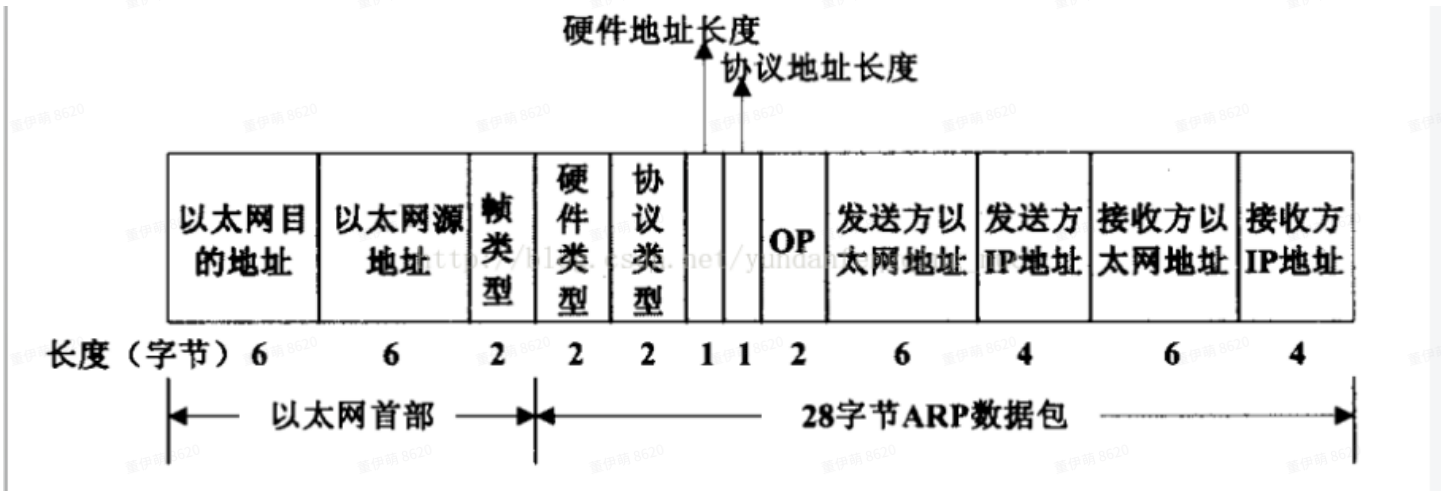
包含48位源MAC地址，源目的MAC地址，。帧类型

- ARP帧 ARPFrame_t

```
typedef struct ARPFrame_t {
    FrameHeader_t FrameHeader; //帧首部
    WORD HardwareType; //硬件类型
    WORD ProtocolType; //协议类型
    BYTE HLen;//硬件地址长度
    BYTE PLen;//协议地址长度
    WORD Operation;//操作类型
    BYTE SendHa[6];//源MAC地址
    DWORD SendIP;//源IP地址
    BYTE RecvHa[6];//目的MAC地址
    DWORD RecvIP;//目的IP地址
}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

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

- sockaddr_in结构

```
struct sockaddr_in {
    short int sin_family; /* Address family */
    unsigned short int sin_port; /* Port number */
    struct in_addr sin_addr; /* Internet address */
    unsigned char sin_zero[8]; /* Same size as struct sockaddr */
    };
    sin_family：指代协议族，在socket编程中只能是AF_INET
    sin_port：存储端口号（使用网络字节顺序）
    sin_addr：存储IP地址，使用in_addr这个数据结构
    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. 程序主要思路和重要代码部分

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`l将nternet地址结构转换成以“.” 间隔的诸如“a.b.c.d”的字符串形式。

```
string CoutIp(unsigned long u) {
    in_addr addr;
    memcpy(&addr, &u, sizeof(u));
    return inet_ntoa(addr);
}

```

`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。

```
string* Byte2Hex(unsigned char bArray[], int bArray_len)
{
    string* strHex = new string();
    int nIndex = 0;
    for (int i = 0; i < bArray_len; i++)
    {
        char hex1;
        char hex2;
        int value = bArray[i];
        int S = value / 16;
        int Y = value % 16;
        if (S >= 0 && S <= 9)
            hex1 = (char)(48 + S);
        else
            hex1 = (char)(55 + S);
        if (Y >= 0 && Y <= 9)
            hex2 = (char)(48 + Y);
        else
            hex2 = (char)(55 + Y);
        if (i != bArray_len - 1) {
            *strHex = *strHex + hex1 + hex2 + "-";
        }
        else
            *strHex = *strHex + hex1 + hex2;
    }

    return strHex;
}
```

`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。

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

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

`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。

```
void ARP_show(struct pcap_pkthdr* header, const u_char* pkt_data)
{
    struct ARPFrame_t* arp_protocol;
    arp_protocol = (struct ARPFrame_t*)(pkt_data);

    cout << "源MAC地址:   " << *(Byte2Hex(arp_protocol->FrameHeader.SrcMAC, 6)) <
    cout << "源IP地址:    " << CoutIp(arp_protocol->SendIP) << endl;
    cout << "目的MAC地址: " << *(Byte2Hex(arp_protocol->FrameHeader.DesMAC, 6)) <
    cout << "目的IP地址   " << CoutIp(arp_protocol->RecvIP) << endl;
    cout << endl;
}
```

`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。

```
pcap_if_t* CAPLIST() {
    pcap_if_t* alldevs;     //指向设备链表首部的指针
    pcap_if_t* d;
    pcap_addr_t* a;
    int          n = 1;
    char        errbuf[PCAP_ERRBUF_SIZE];//错误信息缓冲区

   //获取本机的设备列表
   //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
    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1)
    {
        cout << stderr << "Error in pcap_findalldevs_ex:" << errbuf << endl;
        return 0;
    }


    //显示接口列表

    for (d = alldevs; d != NULL; d = d->next)
    {
        cout << n++ << "." << d->name;
        if (d->description)
            cout << "(" << d->description << ")" << endl;
        else
            cout << "(No description )\n";
        //获取该网络接口的IP地址信息
        for (a = d->addresses; a != NULL; a = a->next) {
```

```
            //判断该地址是否为IP地址
            if (a->addr->sa_family == AF_INET) {
                //输出网络接口卡上绑定的多个IP地址的相关信息
                char str[INET_ADDRSTRLEN];
                inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), str,
                cout << "IP地址: " << str << endl;
                inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->netmask), st
                cout << "网络掩码: " << str << endl;
                inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->broadaddr),
                cout << "广播地址: " << str << endl;

            }
        }
    }
    if (n == 0)
    {
        cout << "\nERROR!\n";

    }
    return alldevs;//pcap_findalldevs_ex函数调用成功后，alldevs参数指向获取的网络接口
}

```

`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。

```
void SET_ARP_Frame_HOST(ARPFrame_t &ARPFrame1, char ip[INET_ADDRSTRLEN]) {
    for (int i = 0; i < 6; i++) {
        ARPFrame1.FrameHeader.DesMAC[i] = 0xff;
        ARPFrame1.FrameHeader.SrcMAC[i] = 0x0f;
        ARPFrame1.SendHa[i] = 0x0f;
        ARPFrame1.RecvHa[i] = 0x00;
    }

    ARPFrame1.FrameHeader.FrameType = htons(0x0806);
    ARPFrame1.HardwareType = htons(0x0001);
    ARPFrame1.ProtocolType = htons(0x0800);
    ARPFrame1.HLen = 6;
    ARPFrame1.PLen = 4;
    ARPFrame1.Operation = htons(0x0001);
    ARPFrame1.SendIP = inet_addr("10.10.10.10");
    ARPFrame1.RecvIP = inet_addr(ip);
}
```

`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APR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。

```
void SET_ARP_Frame_DEST(ARPFrame_t& ARPFrame , char ip[INET_ADDRSTRLEN],unsigned
    for (int i = 0; i < 6; i++) {
        ARPFrame.FrameHeader.DesMAC[i] = 0xff;
        ARPFrame.RecvHa[i] = 0x00;
        ARPFrame.FrameHeader.SrcMAC[i] = mac[i];//设置为本机网卡的MAC地址
        ARPFrame.SendHa[i] = mac[i];//设置为本机网卡的MAC地址
    }
    ARPFrame.FrameHeader.FrameType = htons(0x0806);
    ARPFrame.HardwareType = htons(0x0001);
    ARPFrame.ProtocolType = htons(0x0800);
    ARPFrame.HLen = 6;
    ARPFrame.PLen = 4;
    ARPFrame.Operation = htons(0x0001);
    ARPFrame.SendIP = inet_addr(ip);

}
```

3. 代码展示

程序所有代码如下所示：

```
#define WIN32
#define HAVE_REMOTE
#include "pcap.h"
#include <iostream>
#include <WinSock2.h>
#include <bitset>
#include <process.h>
using namespace std;
#pragma comment(lib,"wpcap.lib")
#pragma comment(lib,"ws2_32.lib")
#pragma warning(disable:4996)


//IP输出格式更改
string CoutIp(unsigned long u) {
    in_addr addr;
    memcpy(&addr, &u, sizeof(u));
    return inet_ntoa(addr);
}

//将地址由BYTE形式转换为16进制字符串类型
string* Byte2Hex(unsigned char bArray[], int bArray_len)
```

```
{
    string* strHex = new string();
    int nIndex = 0;
    for (int i = 0; i < bArray_len; i++)
    {
        char hex1;
        char hex2;
        int value = bArray[i];
        int S = value / 16;
        int Y = value % 16;
        if (S >= 0 && S <= 9)
            hex1 = (char)(48 + S);
        else
            hex1 = (char)(55 + S);
        if (Y >= 0 && Y <= 9)
            hex2 = (char)(48 + Y);
        else
            hex2 = (char)(55 + Y);
        if (i != bArray_len - 1) {
            *strHex = *strHex + hex1 + hex2 + "-";
        }
        else
            *strHex = *strHex + hex1 + hex2;
    }

    return strHex;
}


//指向要转换为字符串的网络字节中的 IP 地址的指针
void* get_in_addr(struct sockaddr* sa)
{
   //判断一下是否为IP
    if (sa->sa_family == AF_INET)
        return &(((struct sockaddr_in*)sa)->sin_addr);
    return &(((struct sockaddr_in6*)sa)->sin6_addr);
}

#pragma pack(1)
#define BYTE unsigned char

//帧首部
typedef struct FrameHeader_t {
    BYTE DesMAC[6]; //源MAC地址
    BYTE SrcMAC[6]; //目的MAC地址
    WORD FrameType; //帧类型
}FrameHeader_t;

//ARP帧
typedef struct ARPFrame_t {
    FrameHeader_t FrameHeader; //帧首部
    WORD HardwareType; //硬件类型
    WORD ProtocolType; //协议类型
    BYTE HLen;//硬件地址长度
    BYTE PLen;//协议地址长度
    WORD Operation;//操作类型
    BYTE SendHa[6];//源MAC地址
    DWORD SendIP;//源IP地址
    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  ARPFrame_t ARPFrame;//要发送的APR数据包(其他主机)
88  ARPFrame_t ARPF_Send;//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 ARPFrame_t* arp_protocol;
95      arp_protocol = (struct ARPFrame_t*)(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),
```

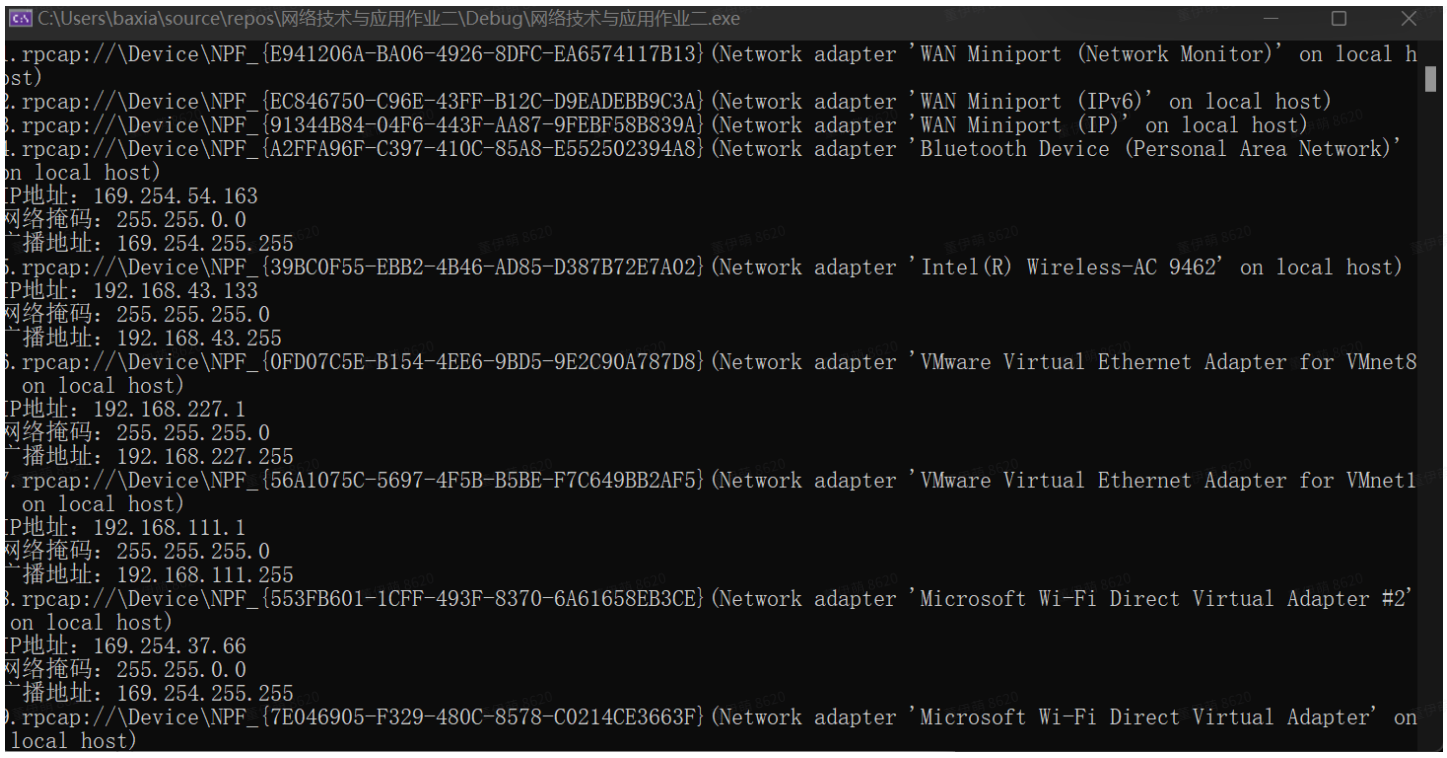
```
141                 cout << "广播地址: " << str << endl;
142
143             }
144         }
145     }
146     if (n == 0)
147     {
148         cout << "\nERROR!\n";
149        // return 0;
150     }
151     return alldevs;//pcap_findalldevs_ex函数调用成功后，alldevs参数指向获取的网络接口
152 }
153
154
155 void SET_ARP_Frame_HOST(ARPFrame_t &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
```

```
    //设备链表首部的指针
    d = alldevs;

    int j;
    cout << "请选择发送数据包的网卡: ";
    cin >> j;
    int i = 0;
    //获取指向选择发送数据包网卡的指针

    while (i < j - 1) {
        i++;
        d = d->next;
    }


    //打开用户选择设备的网卡
    choosed_dev = pcap_open(d->name, 100, PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL,

    if (choosed_dev == NULL) {
        cout << "Error in pcap_open: " << errbuf << endl;
        //失败就释放设备列表;
        pcap_freealldevs(alldevs);
        return 0;
    }
    //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。）
    char ip[INET_ADDRSTRLEN];


    for (a = d->addresses; a != NULL; a = a->next) {
        //判断该地址是否为IP地址
        if (a->addr->sa_family == AF_INET) {
        //InetNtop 函数将 IPv4 或 IPv6 Internet 网络地址转换为采用 Internet 标准格式
            inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), ip, sizeo
        }
    }
    cout << ip;
    cout << endl << d->name << endl;

    //获取本机的MAC地址

    //设置ARP帧相关
    SET_ARP_Frame_HOST(ARPF_Send, ip);

    struct pcap_pkthdr* pkt_header;
    const u_char* pkt_data;
    struct pcap_pkthdr* header = new pcap_pkthdr;
    int k;
    //发送构造好的数据包
    //用pcap_next_ex()捕获数据包，pkt_data指向捕获到的网络数据包
    while ((k = pcap_next_ex(choosed_dev, &pkt_header, &pkt_data)) >= 0) {
        //发送数据包

        pcap_sendpacket(choosed_dev, (u_char*)&ARPF_Send, sizeof(ARPFrame_t));
        struct ARPFrame_t* arp_message;
        arp_message = (struct ARPFrame_t*)(pkt_data);
           if (k == 0)continue;
           else
           {
               if(arp_message->FrameHeader.FrameType==htons(0x0806)&& arp_messag
```

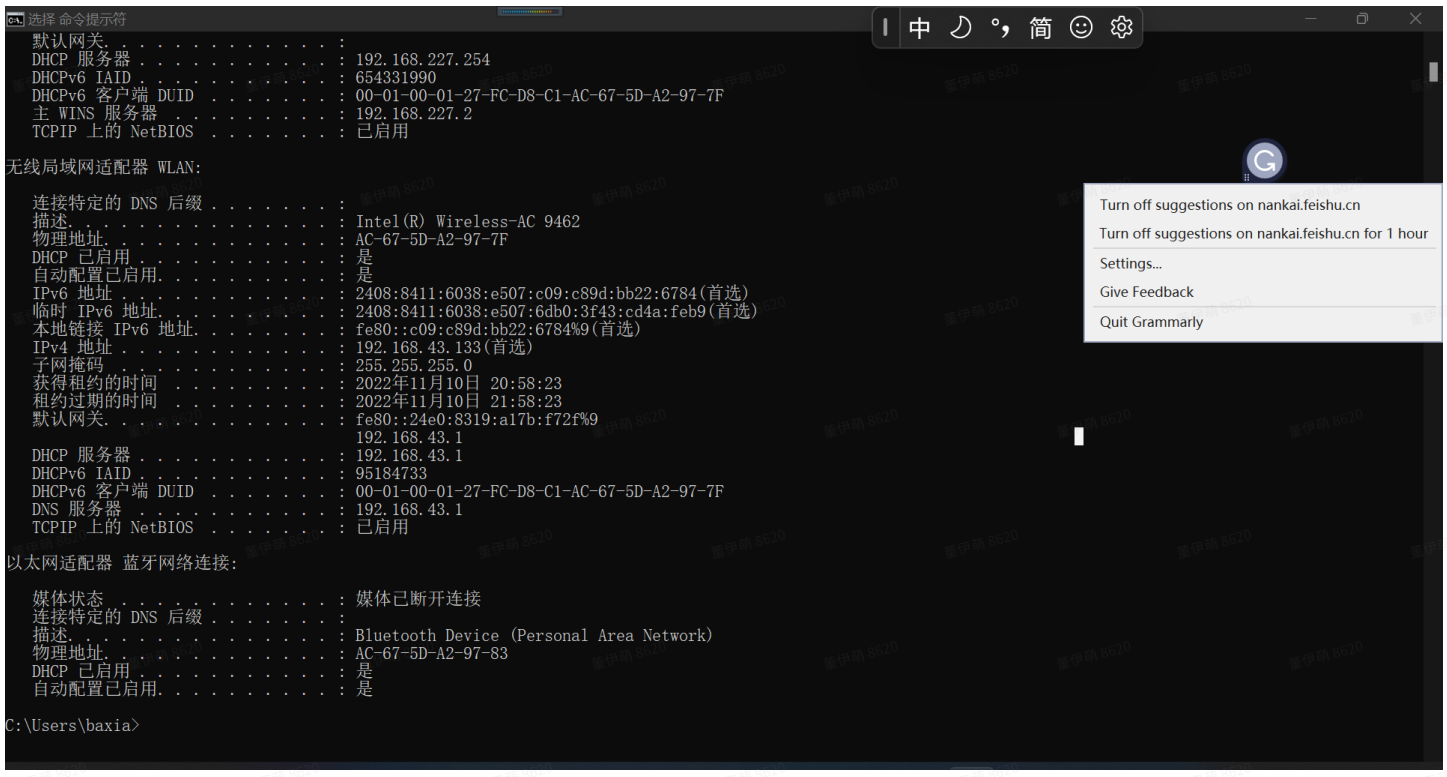
```
259                     cout << "ARP数据包: \n";
260                     ARP_show(header, pkt_data);//打印相应的信息
261                     //用MAC地址记录本机的MAC地址，用于后续构造ARP数据包
262                     for (int i = 0; i < 6; i++) {
263                         mac[i] = *(unsigned char*)(pkt_data + 22 + i);
264                     }
265                     cout << "获取自己主机的MAC地址成功，本机MAC地址为: " << *(Byte2He
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

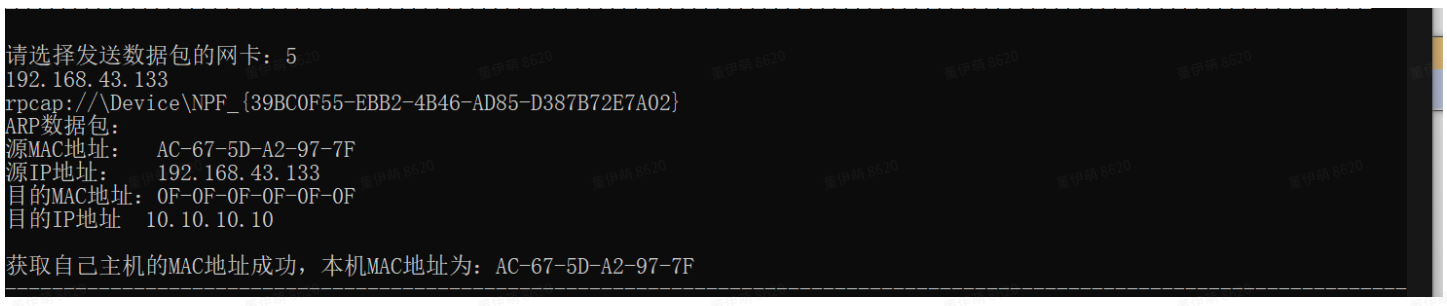


获取本机IP地址与MAC地址的映射关系



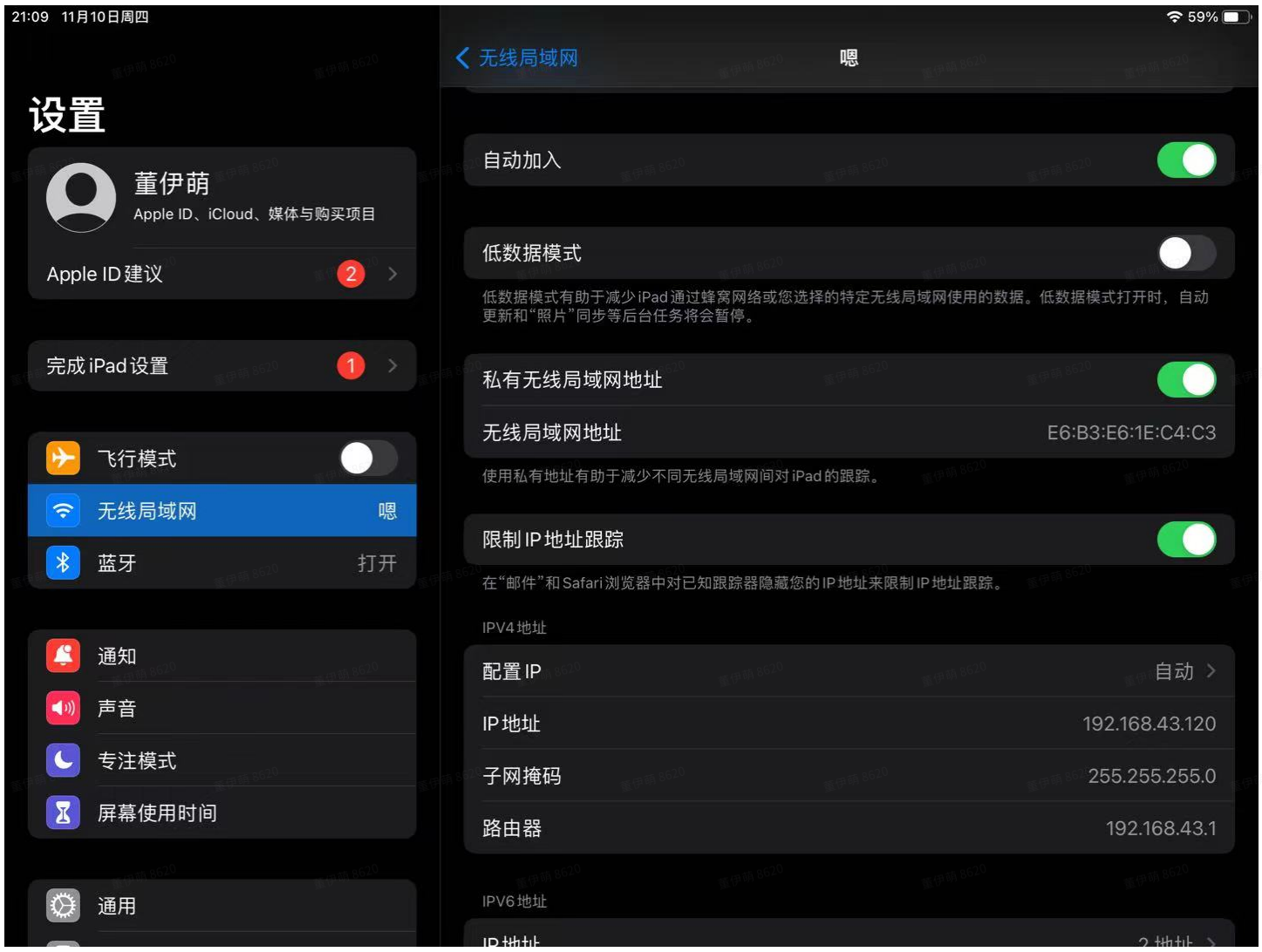
先用命令行找到无线局域网适配器WLAN的物理地址如上图：AC-67-5D-A2-97-7F

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：



本机和ipad分别连接手机热点，结果如下：

```
请输入目的主机的IP地址：192.168.43.120
ARP数据包：
操作类型:2
源MAC地址：    E6-B3-E6-1E-C4-C3
源IP地址：     192.168.43.120
目的MAC地址：AC-67-5D-A2-97-7F
目的IP地址   192.168.43.133

获取目的主机的MAC地址成功，目的主机的MAC地址为：E6-B3-E6-1E-C4-C3
请按任意键继续. . .
```



对照后发现物理地址和IP地址是匹配的，程序运行成功。