南開大學
Nankai University

实验3：通过编程获取IP地址与MAC地址的对应关系

- 姓名：陈睿颖
- 学号：2013544
- 专业：计算机科学与技术

1. 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：

a. 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。

b. 通过Npcap编程，获取IP地址与MAC地址的映射关系。

c. 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。

d. 编写的程序应结构清晰，具有较好的可读性。

2. 实验准备

2.1 实验环境准备

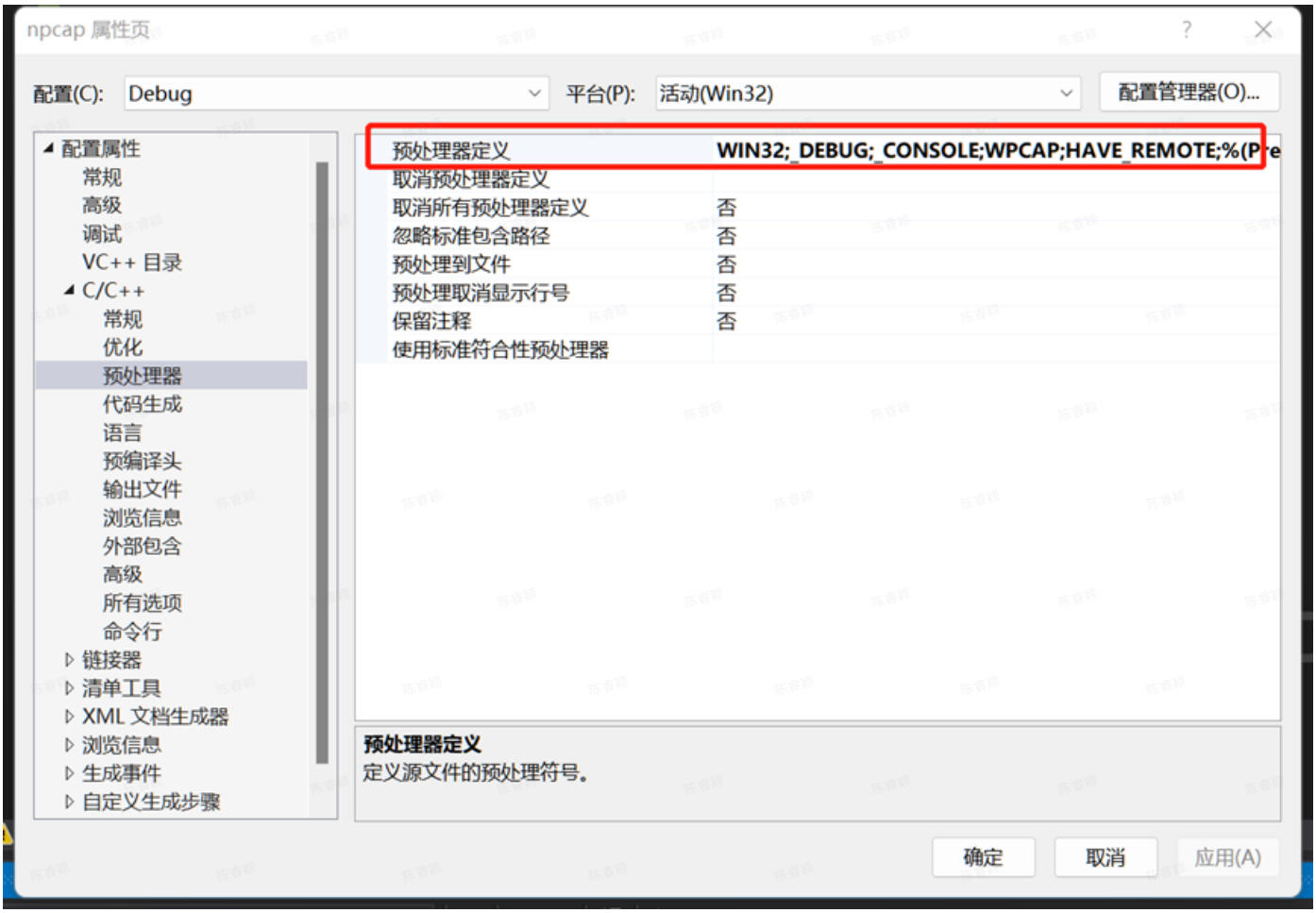
2.1.1 编程环境准备

a. 添加pcap.h包含文件：程序中使用了WinPcap提供的函数，所以需要包含头文件：

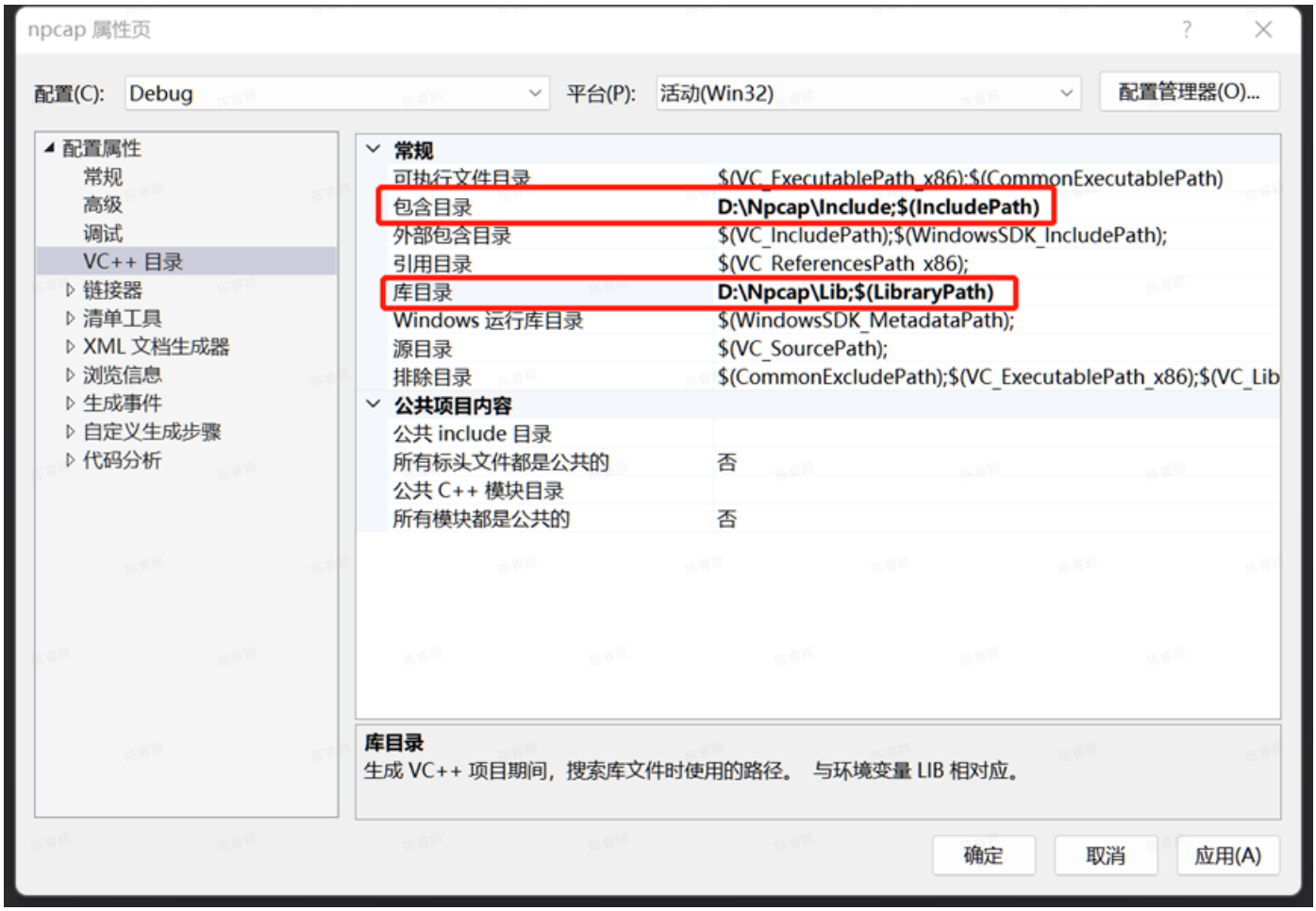
```
#include “pcap.h”
```

b. 增加与WinPcap有关的预处理器定义：

```
WPCAP
HAVE_REMOTE
```



c. 添加包含文件目录与npcap.lib文件目录：



2.1.2 实验验证环境准备

本次实验运行验证结果时，准备一台手机、一台平板电脑、一台笔记本电脑。其中手机联网使用移动数据，平板电脑与笔记本电脑连接在手机热点上。具体信息如下：

“LAPTOP-LADTBSMC”为笔记本电脑，“*”为平板电脑。

可以获得的信息整合如下表：

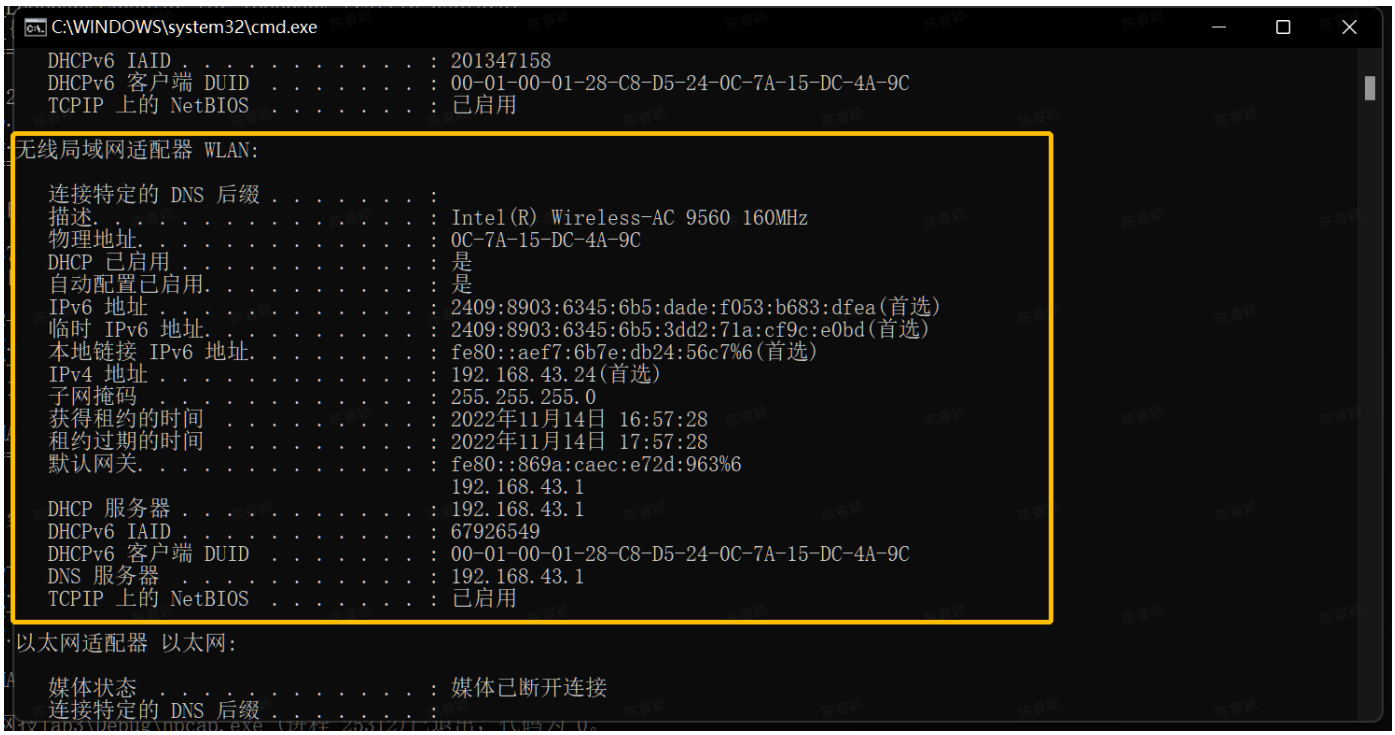
设备	设备名称	IP地址	MAC地址

笔记本电脑	LAPTOP-LADTBSMC	192.168.43.24	0C-7A-15-DC-4A-9C
平板电脑	*	192.168.43.244	76-91-DA-D3-CF-BE

此外，再通过命令

```
1 ipconfig /all
```

查明主机（笔记本电脑）信息如下：



3. 实验过程

3.1 实验中使用的数据结构

- `FrameHeader_t`：

  帧首部，其中包含48位的源MAC地址、目的MAC地址和帧类型。

- `ARPFrame_t: ARP`

  帧，包含帧首部、硬件类型、协议类型、硬件地址长度、协议地址长度、操作类型、源MAC地址、源IP地址、目的MAC地址和目的IP地址。

- `string* ByteToHexStr(unsigned char byte_arr[], int arr_len)`：

自定义函数，用于将Byte类型转换为十六进制的字符串，主要用来处理源MAC地址和目的MAC地址。

- `string GetIp(unsigned long u)`

  自定义函数，将源IP地址和目的IP地址转换成“.”点隔的字符串格式。

- `void PrintWordInBit(WORD b)`

  自定义函数，将16位的校验和转换成二进制的形式并打印在屏幕上。

- `void arp_protool_packet(struct pcap_pkthdr* header, const u_char* pkt_data)`

  自定义函数，用来打印捕获到的ARP数据包的主要内容。

3.2 程序设计思路

1. 用`pcap_findalldevs()`函数获取网络接口设备列表。如果获取成功，则继续向下进行；否则，打印errbuf里的错误信息，函数返回。

```
if (pcap_findalldevs(&allDevices, errbuf) == -1)
{
        cout << stderr << "寻找设备错误" << errbuf << endl;
        return 0;
}
```

2. 上一个函数调用成功后，`allDevices`参数指向获取的网络接口列表的第一个元素，下面通过指针和循环，遍历所有的网络接口，并打印设备的名字和描述。使用循环获取该网络接口设备的IP地址信息，打印IP地址、网络掩码和广播地址。

```
for (currentDevice = allDevices; currentDevice; currentDevice = currentDevice->n
    cout << ++i << ". " << currentDevice->name;
    if (currentDevice->description)
            cout << "(" << currentDevice->description << ")" << endl;
    else
            cout << "(无可用描述)\n";
    for (a = currentDevice->addresses; a != NULL; a = a->next) {
            if (a->addr->sa_family == AF_INET) {
                    cout << "=================================================
                    char str[INET_ADDRSTRLEN];
                    inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), s
                    cout << "IP 地址:" << str << endl;
                    inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->netmask)
```

```
                    cout << "网络掩码:" << str << endl;
                    inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->broadadd
                    cout << "广播地址:" << str << endl;

            }
    }
        }
```

3. 由用户输入选择指定的设备，并调用函数 `pcap_open` 打开对应的网卡，成功则继续进行；否则，打印 `errbuf` 里的错误信息，调用 `pcap_freealldevs()` 释放该设备列表，函数返回。并使用 `char ip[INET_ADDRSTRLEN]` 保存该网卡的ip地址，在后面构造ARP数据包时使用

```
targetDevice = pcap_open(currentDevice->name, 100, PCAP_OPENFLAG_PROMISCUOUS, 10
if (targetDevice == NULL) {
        cout << "打开设备错误 " << errbuf << endl;
        pcap_freealldevs(allDevices);
        return 0;
}
char ip[INET_ADDRSTRLEN];
    for (a = d->addresses; a != NULL; a = a->next) {
        if (a->addr->sa_family == AF_INET) {
            inet_ntop(AF_INET, get_in_addr((struct sockaddr*)a->addr), ip, sizeo
        }
    }
```

4. 获取本机的MAC地址，步骤如下：

- 首先构造要发送的ARP数据包 `ARPFrame1` ： `DesMAC` 设置为广播地址； `SrcMAC` 、 `SendHa` 和 `SendIP` 随意设置； `RecvHa` 设置为0； `RecvIP` 设置为本机的IP地址，用上面得到的ip设置；帧类型为ARP；硬件类型为以太网；协议类型为IP；硬件地址长度为6；协议地址长度为4；操作为ARP请求。

```
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);
```

- 在循环中使用 `pcap_sendpacket` 发送构造好的数据包，并使用 `pcap_next_ex` 捕获数据包。设置过滤条件为“帧类型为ARP且操作类型为ARP响应且 `SendIP` 为发送的数据包中的 `RecvIP` （即本机IP地址）”，捕获到符合条件的数据包就调用 `arp_protool_packet` 打印相应信息并跳出循环。

同时，用mac数组记录本机的MAC地址，用于后续构造ARP数据包。

```
while ((k = pcap_next_ex(targetDevice, &pkt_header, &pkt_data)) >= 0) {
    pcap_sendpacket(targetDevice, (u_char*)&ARPFrame1, sizeof(ARPFrame_t));

    if (k == 0)continue;
    else if (*(unsigned short*)(pkt_data + 12) == htons(0x0806) && *(unsignedshort*)(pkt_data + 20) == htons(0x0002)&& *(unsigned long*)(pkt_data + 28) == ARPFrame1.RecvIP) {
            cout << "ARP 数据包中主要内容: \n";
            arp_protool_packet(header, pkt_data);
            for (int i = 0; i < 6; i++) {
                    mac[i] = *(unsigned char*)(pkt_data + 22 + i);
            }
            cout << "获取自己主机的 MAC 地址成功，本机 MAC 地址为: " <<
                    *(ByteToHexStr(mac, 6)) << endl;
            break;
    }
}
```

5. 获取目的主机（在本实验中即为平板电脑）的MAC地址，步骤如下：

- 首先构造要发送的ARP数据包 `ARPFrame：DesMAC` 设置为广播地址； `SrcMAC` 和 `SendHa` 为本机MAC地址，用上面获得的mac数组填充； `RecvHa` 设置为0； `SendIP` 为本机的IP地址，用上面得到的ip设置； `RecvIP` 设置为目的主机的IP地址，由用户输入；其他关于类型的设置均与上面相同。

```
for (int i = 0; i < 6; i++) {
        ARPFrame.FrameHeader.DesMAC[i] = 0xff;  //设置为广播地址
        ARPFrame.RecvHa[i] = 0x00;              //设置为0
        ARPFrame.FrameHeader.SrcMAC[i] = mac[i]; //设置为本机的mac地址
        ARPFrame.SendHa[i] = mac[i];            //设置为本机的mac地址
```

```
}
ARPFrame.FrameHeader.FrameType = htons(0x0806);  //帧类型为ARP
ARPFrame.HardwareType = htons(0x0001);           //硬件类型为以太网
ARPFrame.ProtocolType = htons(0x0800);           //协议类型为IP
ARPFrame.HLen = 6;                               //硬件地址为6
ARPFrame.PLen = 4;                               //协议地址长度为4
ARPFrame.Operation = htons(0x0001);              //操作为ARP请求
ARPFrame.SendIP = inet_addr(ip);                 //设置为本机网卡上绑定的ip地址
cout << "请输入目的主机的 IP 地址:";
char s[INET_ADDRSTRLEN];
cin >> s;
ARPFrame.RecvIP = inet_addr(s);                  //设置为请求的ip地址
```

- 在循环中使用 `pcap_sendpacket` 发送构造好的数据包，并使用 `pcap_next_ex` 捕获数据包。设置过滤条件为“帧类型为ARP且操作类型为ARP响应且SendIP为发送的数据包中的RecvIP（即目的主机的IP地址）”，捕获到符合条件的数据包就调用 `arp_protool_packet` 打印相应信息并跳出循环。

  同时，用desmac数组记录得到的目的主机的MAC地址。

```
while ((k = pcap_next_ex(targetDevice, &pkt_header, &pkt_data)) >= 0) {
        //进行数据包捕获
        pcap_sendpacket(targetDevice, (u_char*)&ARPFrame, sizeof(ARPFrame_t));


if (i == 0)continue;
else if (*(unsigned short*)(pkt_data + 12) == htons(0x0806)
        && *(unsigned short*)(pkt_data + 20) == htons(0x0002)
        && *(unsigned long*)(pkt_data + 28) == ARPFrame.RecvIP) {
        cout << "ARP 数据包主要内容\n";
        arp_protool_packet(header, pkt_data);
        for (int i = 0; i < 6; i++) {
                desmac[i] = *(unsigned char*)(pkt_data + 22 + i);
        }
        cout << "获取目的主机的 MAC 地址成功，目的主机的 MAC 地址为: " <<
                *(ByteToHexStr(desmac, 6)) << endl;
        break;
}

}
```

6. 最后调用 `pcap_freealldevs(alldevs)` 释放所有设备。

3.3 实验结果验证

运行程序后，根据**2.1.2**节的主机网卡信息选择如下网卡设备：

```
IP 地址:169.254.191.148
网络掩码:255.255.0.0
广播地址:169.254.255.255
6. \Device\NPF_{3337A2ED-C76E-4AD7-91EA-461B663C2C70}(Intel(R) Wireless-AC 9560 160MHz)
==========================================================================
```

接着输入平板电脑的IP地址：

```
1 192.168.43.244
```

```
请选择发送数据包的网卡: 6
192.168.43.24
\Device\NPF_{3337A2ED-C76E-4AD7-91EA-461B663C2C70}
ARP 数据包中主要内容:
操作类型:2
源 MAC 地址:  0C-7A-15-DC-4A-9C
源 IP 地址:  192.168.43.24
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: 0C-7A-15-DC-4A-9C
==========================================================================

请输入目的主机的 IP 地址:192.168.43.244
ARP 数据包主要内容
操作类型:2
源 MAC 地址:  76-91-DA-D3-CF-BE
源 IP 地址:  192.168.43.244
目的 MAC 地址:0C-7A-15-DC-4A-9C
目的 IP 地址 192.168.43.24

获取目的主机的 MAC 地址成功，目的主机的 MAC 地址为: 76-91-DA-D3-CF-BE

D:\111programs\网技lab3\Debug\npcap.exe (进程 25312)已退出，代码为 0。
按任意键关闭此窗口. . .
```

对照下表：

设备	设备名称	IP地址	MAC地址
笔记本电脑	LAPTOP-LADTBSMC	192.168.43.24	0C-7A-15-DC-4A-9C
平板电脑	*	192.168.43.244	76-91-DA-D3-CF-BE

实验结果正确！