实验 3：通过编程获取 IP 地址与 MAC 地址的对应关系

一、实验说明

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 NPcap 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、前期准备

1、学习 WinPcap 的数据包发送方法

本次实验中，以太网发送数据包使用 WinPcap 提供的 pcap_sendpacket(pcap_t *p, u_char buf, int size) 函数，其中的参数：

p 指定函数通过哪块接口网卡发送数据包。

buf 指向需要发送的数据包。其中不包含以太网帧的 CRC 校验和字段。

size 指定发送数据包的大小。

2、ARP 请求获取主机网卡对应 IP 的 MAC 地址基本思想

（1）获取网络接口卡列表，选择需要捕获 MAC 地址的网卡 A（或选择对应的 IP）

（2）伪造 ARP 请求报文 S，内容要求如下：

ARP 请求；

广播；

伪造源 MAC 地址和源 IP 地址；

目的 IP 地址为网卡 A 的 IP 地址；

用网卡 A 发送 ARP 请求报文 S；

（3）对网卡 A 进行流量监听，筛选其中的 ARP 报文（类型为 0x806），捕获网卡 A 的 ARP 响应报文，在响应报文的帧首部源 MAC 地址部分可以看到发送该 ARP 响应的网卡对应的 MAC 地址

3、ARP 请求获取输入 IP 地址对应的 MAC 地址的基本思想

（1）利用已获取的主机对应 IP 的 MAC 地址，构造 ARP 请求报文，内容如下：

ARP 请求；

广播；

源 MAC 地址和源 IP 地址为主机网卡 A 的 MAC 地址和 IP 地址；

目的 IP 地址为输入的 IP 地址；

用主机网卡 A 发送构造的 ARP 请求报文；

（2）对主机网卡 A 进行流量监控，筛选其中的 ARP 报文（类型为 0x806），捕获输入 IP 地址对应主机的 ARP 响应报文，在响应报文的帧首部源 MAC 地址部分可以看到发送该 ARP 响应的 IP 地址对应的 MAC 地址

三、实验过程

1、获取设备列表

```
pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf)
```

利用 pcap_findalldevs 函数获取设备列表，alldevs 为设备列表首部的指针

2、打印网卡信息和 IP 地址

```
for (d = alldevs; d != NULL; d = d->next) { //遍历设备链表，显示 IP
    //获取当前网络接口设备的 IP 地址信息
    i++;
    cout << "网卡 " << i << " " << d->name << endl;
    cout<<"描述信息：" << d->description <<endl;
    for (a = d->addresses; a != NULL; a = a->next) {
        if ((a->addr->sa_family == AF_INET)) {  //判断该地址是否为 IP 地址
            //将 a->addr 转化为 sockaddr_in 类型获取网络字节序的 IP 地址，inet_ntop 将其转化为点分十进制的 IP
            char address[INET_ADDRSTRLEN];
            inet_ntop(AF_INET, &(((struct sockaddr_in*)a->addr)->sin_addr), address, sizeof(address));
            cout << "IP 地址：" << address<< endl;
            inet_ntop(AF_INET, &(((struct sockaddr_in*)a->netmask)->sin_addr), address, sizeof(address));
            cout << "子网掩码：" << address << endl;
        }
    }
    cout << endl;
    }
```

for 循环利用 alldevs 遍历网卡设备列表，输出网卡设备的网卡名称、描述信息、IP 地址和子网掩码。

3、打开主机所在的网络接口卡

```
pcap_handle = pcap_open(d->name, 65536, PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL, errbuf)
```

4、设置过滤规则，过滤 ARP 包

```
//设置过滤规则，过滤 ARP 包
u_int netmask;
netmask = ((sockaddr_in*)(d->addresses->netmask))->sin_addr.S_un.S_addr;
bpf_program fcode;  //存放编译后的规则
char packet_filter[] = "ether proto \\arp";//过滤规则，ether 表示以太网头部，以太网头部 proto 字段值为 0x0806，即 ARP
if (pcap_compile(pcap_handle, &fcode, packet_filter, 1, netmask) < 0)
{
    cout << "无法编译数据包过滤器。检查语法";
    pcap_freealldevs(alldevs);
    return 0;
}
```

```
//设置过滤器
if (pcap_setfilter(pcap_handle, &fcode) < 0)
{
    cout << "过滤器设置错误";
    pcap_freealldevs(alldevs);
    return 0;
    }
```

5、组装 ARP 报文

（1）定义 FrameHeader 以太网帧首部结构和 ARP 帧结构 ARPFrame_t:

```
#pragma pack(1) //以 1 字节对齐
typedef struct FrameHeader_t {  //定义以太网帧首部
    BYTE DesMac[6]; //目的地址
    BYTE SrcMac[6]; //源地址
    WORD FrameType; //帧类型
}FrameHeader_t;

typedef struct ARPFrame_t {//ARP 帧
    FrameHeader_t FrameHeader;
    WORD HardwareType;//硬件类型，以太网接口类型为 1
    WORD ProtocolType;//协议类型，IP 类型为 0800H
    BYTE HLen;//硬件地址长度，物理地址 MAC 长度为 6B
    BYTE PLen;//协议地址长度，IP 地址长度为 4B
    WORD Operation;//操作类型，ARP 请求报文为 1，响应报文为 2
    BYTE SendHa[6];//发送方 MAC 地址
    DWORD SendIP;//发送方 IP 地址
    BYTE RecvHa[6];//接收方 MAC 地址
    DWORD RecvIP;//接收方 IP 地址
}ARPFrame_t;

    #pragma pack()  //恢复默认对齐方式
```

（2）构造 ARP 请求报文，以获取主机网卡 IP 对应的 MAC 地址

```
for (int i = 0; i < 6; i++)
{
    ARPFrame.FrameHeader.DesMac[i] = 0xFF;//设置为本机广播地址
255.255.255.255.255.255
    ARPFrame.FrameHeader.SrcMac[i] = 0x66;//设置为虚拟的 MAC 地址 66-66-66-66-66-66
    ARPFrame.RecvHa[i] = 0;//设置为 0
    ARPFrame.SendHa[i] = 0x66; //设置为虚假的 MAC 地址 66-66-66-66-66-66-66
}
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;//硬件地址长度为 6
```

```
    ARPFrame.PLen = 4; // 协议地址长为 4
    ARPFrame.Operation = htons(0x0001);//操作为 ARP 请求
    SendIP = ARPFrame.SendIP = htonl(0x70707070);//源 IP 地址设置为虚拟的 IP 地址
112.112.112.112
    //将所选择的网卡的 IP 设置为请求的 IP 地址
    for (a = d->addresses; a != NULL; a = a->next){
        if (a->addr->sa_family == AF_INET){
            RevIP = ARPFrame.RecvIP = ((struct sockaddr_in*)a->addr)->sin_addr.s_addr;
        }
    }
```

设置 ARP 请求报文以太网帧首部的目的 MAC 地址为广播地址 255.255.255.255.255.255，源 MAC 地址为一个虚假的 MAC 地址，此处设置为 66-66-66-66-66-66，同样 ARP 帧的源 MAC 地址设置为该虚拟地址 66-66-66-66-66-66，由于此时的 ARP 帧硬件目的 MAC 地址还未知，所以设置为 0。

帧类型为 0x0806，硬件类型 0x0001 表示以太网，协议类型 0x0800 表示 IP 协议，硬件地址长度 HLEN 为 6，协议长度 PLEN 为 4，操作为 0x0001 表示 ARP 请求。

源 IP 地址设置一个虚假的 IP 地址 112.1112.112.112，目的 IP 地址设置为主机所在网卡的 IP 地址。

6、发送 ARP 请求，捕获 ARP 响应，获取本机网卡对应 IP 的 MAC 地址

（1）利用 pcap_sendpacket 函数发送 ARP 请求，其中 pcap_handle 为主机所在的网卡，ARPFrame 为要发送的 ARP 请求包，发送成功返回值为 0；

```
    pcap_sendpacket(pcap_handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)
```

（2）pcap_next_ex 函数捕获 ARP 响应包，捕获数据存储在 pkt_data 中，将其转化为 ARPFrame 结构，输出该响应包的源 IP 地址和源目的地址，即为主机所在网卡的对应 IP 的 MAC 地址。

```
    while (true) {
            int result = pcap_next_ex(pcap_handle, &pkt_header, &pkt_data); //得到
pcap_next_ex 的返回结果
            if (result == 0) {  //未捕获到数据包
                cout << "在指定时间范围（read_timeout)内未捕获到数据包" << endl;
                continue;
            }
            else if (result == -1) {  //调用过程发生错误
                cout << "捕获数据包出错" << endl;
                return 0;
            }
            else {  //result=1，捕获成功
                IPPacket = (ARPFrame_t*)pkt_data;  //捕获到的数据包转化为自定义的
ARPFrame_t 数据包类型
                //判断捕获的 IP 包是否为之前发的 ARP 请求的响应包
                if (IPPacket->RecvIP == SendIP && IPPacket->SendIP == RevIP) {
```

```
                cout << "本机网络接口的 IP 地址与 MAC 地址对应关系如下：" <<endl;
                //二进制 IP 转化为字符串
                char address[INET_ADDRSTRLEN];
                inet_ntop(AF_INET, &IPPacket->SendIP, address, sizeof(address));
                cout << "IP 地址：" << address;
                cout << "   MAC 地址：  ";
                for (int i = 0; i < 6; i++) {
                    if (i < 5)
                        printf("%02x:", IPPacket->SendHa[i]);
                    else
                        printf("%02x", IPPacket->SendHa[i]);
                }
                cout << endl;
                break;
            }
        }
    }
```

7、向网络发送 ARP 请求报文，获取其他主机的 IP 地址和 MAC 地址的对应关系

过程和上述类似，利用已获取的主机网卡对应的 IP 地址和 MAC 地址的映射关系，并输入一个 IP 地址，该 IP 地址作为目的 IP 地址构造 ARP 请求报文，利用主机网卡发送 ARP 请求，捕获收到的 ARP 响应包，从而获取该 IP 地址和 MAC 地址的映射关系。

（1）构造 ARP 请求包，目的 IP 设置为输入的 IP 地址，源 IP 地址设置为主机 IP 地址，源 MAC 地址设置为主机网卡 MAC 地址

```
RevIP = ARPFrame.RecvIP = sa.sin_addr.s_addr;  //设置为请求 IP
SendIP = ARPFrame.SendIP = IPPacket->SendIP; //设置为主机 IP
for (int i = 0; i < 6; i++) { //设置源 MAC 地址为主机 MAC 地址
    ARPFrame.SendHa[i] = ARPFrame.FrameHeader.SrcMac[i] = IPPacket->SendHa[i];
}
```

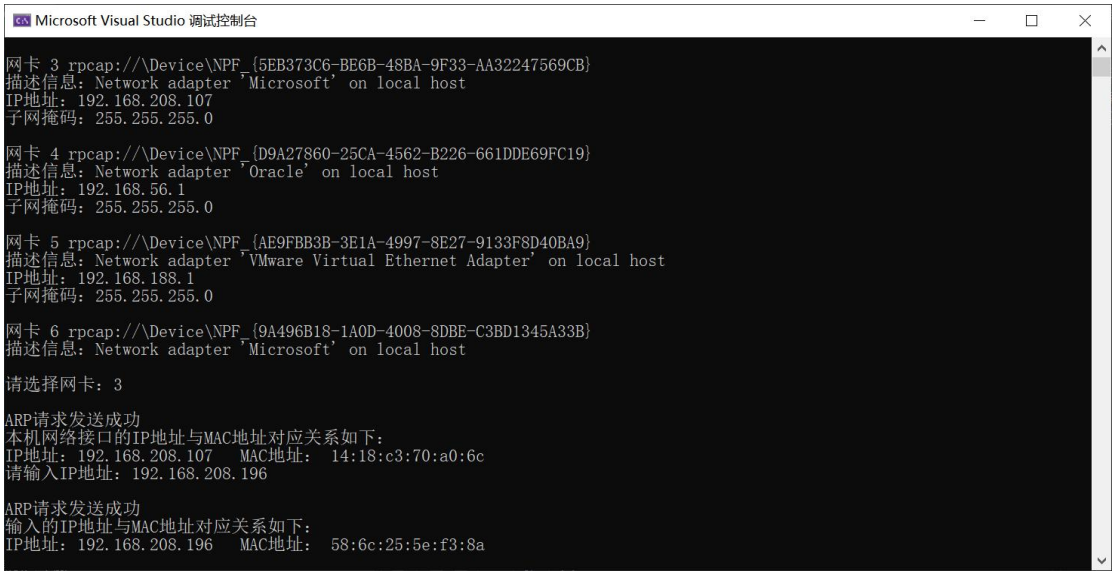
（2）pcap_sendpacket 利用主机网卡发送 ARP 请求包，pcap_next_ex 捕获 ARP 响应包，获取其目的 MAC 地址，即为输入主机 IP 地址的对应 MAC 地址。

```
pcap_sendpacket(pcap_handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t);
pcap_next_ex(pcap_handle, &pkt_header, &pkt_data);
```

8、关闭网卡，释放设备链表

```
pcap_close(pcap_handle);  //关闭当前接口
pcap_freealldevs(alldevs); //释放设备链表
```

四、实验结果



如图所示，输出所有网卡设备的名称、描述、IP 地址和子网掩码，接着输入网卡号以打开主机所在网卡，已知为 3 号，所以打开 3 号网卡，接着看到 ARP 请求发送成功，成功获取主机网卡的对应 IP 地址和 MAC 地址；

接着输入一个 IP 地址，可以看到成功获取到它的对应 MAC 地址。

五、实验总结与思考

本次实验基于上次实验捕获数据包的过程，实现了利用 ARP 请求响应获取主机网卡对应 IP 地址和 MAC 地址的映射关系，进而获得同一局域网中其他主机 IP 地址和 MAC 地址。

实验中发送的 ARP 请求报文的目的 MAC 地址虽然设置为广播类型，但实际上它并不能够获取同一局域网中所有主机的 IP 地址和 MAC 地址映射关系，它需要知道目标主机的 IP 地址才能发起请求，其他主机在收到该请求报文后，会检查自己的 IP 地址是否与请求的 IP 地址匹配。只有 IP 地址与 ARP 请求报文 IP 地址相匹配的主机才会作出 ARP 响应，不匹配的主机会忽略该请求报文，因此一台主句发送一次 ARP 请求报文并不能够获取同一局域网中其他所有主机的 IP 地址和 MAC 地址的映射关系。

通过查阅资料，若想获取同一局域网中所有主机的 IP 地址和 MAC 地址的映射关系，可以借助 ARP 扫描工具，这些工具通过发送大量的 ARP 请求并收集响应来获取网络中各个设备的 IP 地址和 MAC 地址映射关系。ARP 扫描工具通常用于网络管理、安全评估和故障排除等任务。但使用这样的工具需要有足够的权限，并且在某些环境中可能被视为潜在的安全风险。

常用的 ARP 扫描工具有：

（1）arp-scan:

特点： 开源工具，支持跨平台（Linux、Windows、Mac OS 等）。

使用示例： arp-scan --localnet

（2）NetScanTools 网站:

特点： 提供了多种网络工具，包括 ARP 扫描功能。

（3）Angry IP Scanner 网站:

特点： 开源跨平台工具，支持多种扫描方式。

（4）Nmap:

特点： 著名的网络扫描工具，支持多种扫描技术，包括 ARP 扫描。

使用示例： nmap -sn <IP 范围>

（5）Colasoft MAC Scanner 网站:

特点： 提供了图形化用户界面，适用于 Windows 平台。