网络技术与应用课程第 3 次实验报告

实验名称：通过编程获取 IP 地址与 MAC 地址的对应关系

学号： 1711409 姓名： 张嘉尧 班次： 周二上午

一、实验目的和要求

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。且编写的程序应结构清晰，具有较好的可读性。

二、实验过程

（一）程序工作原理和执行过程

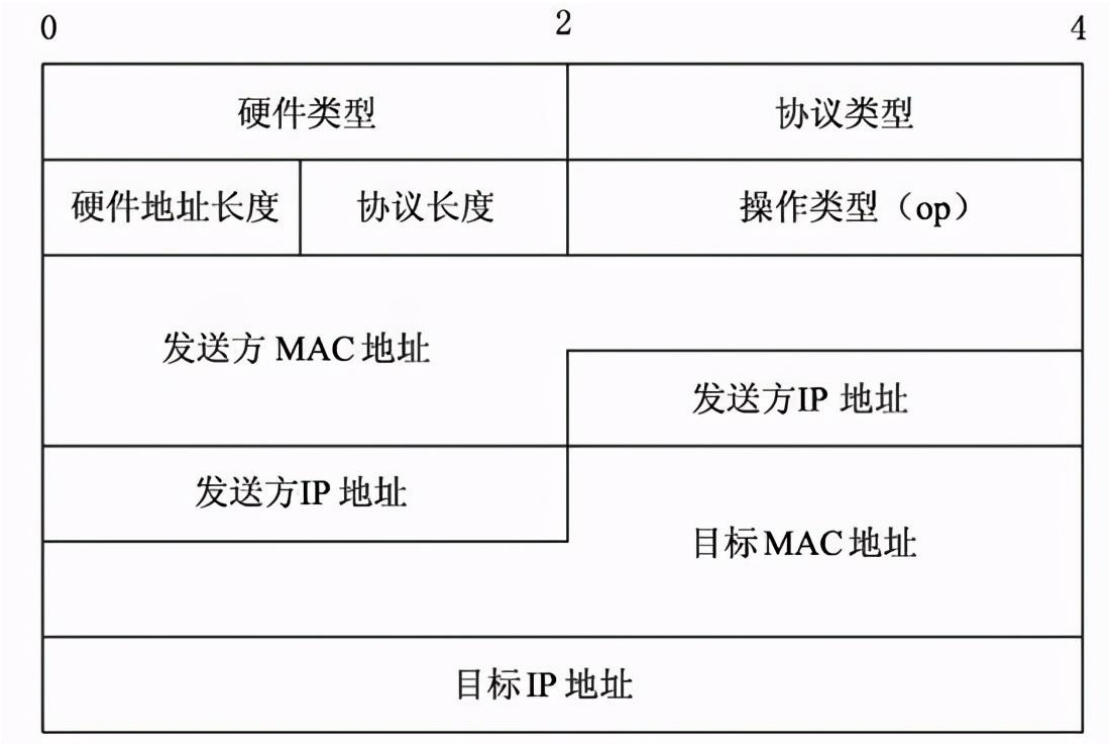
ARP 的基本思想是主机 A 广播发送带有主机 B 的 IP 地址的 ARP 请求，主机 B 识别请求，并向 A 发送带有主机 B 的 IP 地址和 MAC 地址映射关系的 ARP 响应，主机 A 得到 ARP 响应报文，得到该映射关系并继续使用。

通过 NPcap 能够打开本机的设备列表，获取本设备的网卡名和 IP 地址等信息，然后通过获得的 IP 地址，通过 NPcap 的 pcap_sendpacket()函数虚拟一个主机向本机发送 ARP 请求（虚拟主机的 IP 和 MAC 地址随便定），然后编程截获本机的 ARP 响应即可获得本机的 IP 和 MAC 地址对应关系。之后可以通过获取的本机 IP 和 MAC 地址以及输入的目的主机 IP 地址发送 ARP 请求获取对应的 IP 与 MAC 地址对应关系。

- 通过 pcap_findalldevs()获取所有的设备列表
- 选择一个设备扫描其上的地址信息，找到 IP 地址
- 通过找到的 IP 地址向本机发送一个 ARP 请求
- 通过 while 循环截获本机的 ARP 响应报文，解析本机 IP 与 MAC 地址对应关系
- 客户端输入发送 ARP 请求的命令，根据输入的 IP 和本机的 IP、MAC 地址组成 ARP 请求报文并广播到同一个网络中

• 循环接收数据报，触发解析报文函数进行解析并显示到界面上，完成获取 IP 与 MAC 地址对应关系

ARP 报文格式如图：



（二）实验过程

本次实验定义的 ARP 报文格式如下：

```
#pragma pack(1)
struct FrameHeader_t //帧首部
{
    BYTE DesMAC[6];  //目的地址
    BYTE SrcMAC[6];  //源地址
    WORD FrameType;  //帧类型
};

struct ARPFrame_t               //ARP帧
{
    FrameHeader_t FrameHeader;  //帧头部
    WORD HardwareType;          //硬件类型
    WORD ProtocolType;             //协议类型
    BYTE HLen;                  //地址长度MAC
    BYTE PLen;                  //地址长度IP
    WORD Operation;             //协议动作
    BYTE SendHa[6];             //源地址MAC
    DWORD SendIP;               //源地址IP
```

```
    BYTE RecvHa[6];             //目的地址MAC
    DWORD RecvIP;               //目的地址IP
};
    #pragma pack()          //恢复缺省对齐方式
```

获取本机所有网卡设备信息列表 NPcap 的 pcap_findalldevs 函数：

```
//获得本机的设备列表
    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1)
    {
        cout << "获取网络接口时发生错误:" << errbuf << endl;
        return 0;
    }
```

显示接口列表，打印网卡信息、IP 和子网掩码。

```
//显示接口列表
    for (ptr = alldevs; ptr != NULL; ptr = ptr->next)
    {
        cout << "网卡" << index + 1 << "\t" << ptr->name << endl;
        cout << "描述信息：" << ptr->description << endl;

        for (a = ptr->addresses; a != NULL; a = a->next)
        {

            if (a->addr->sa_family == AF_INET)
            {

                cout << "  IP地址：" << inet_ntoa(((struct
sockaddr_in*)(a->addr))->sin_addr) << endl;
                cout << "  子网掩码：" << inet_ntoa(((struct
sockaddr_in*)(a->netmask))->sin_addr) << endl;

            }
        }

        index++;
    }
```

选择其中一个网卡打开：

```
int num;
    cout << "请选要打开的网卡号：";
    cin >> num;
    ptr = alldevs;
    for (int i = 1; i < num; i++)
    {
```

```
        ptr = ptr->next;
    }

    pcap_t* pcap_handle = pcap_open(ptr->name, 1024, PCAP_OPENFLAG_PROMISCUOUS, 1000,
NULL, errbuf);//打开网卡
    if (pcap_handle == NULL)
    {
        cout << "打开网卡时发生错误：" << errbuf << endl;
        return 0;
    }
    else
    {
        cout << "成功打开该网卡" << endl;
    }
```

编译设置过滤器，只捕获 ARP 包：

```
//编译过滤器，只捕获ARP包
    u_int netmask;
    netmask = ((sockaddr_in*)(ptr->addresses->netmask))->sin_addr.S_un.S_addr;
    bpf_program fcode;
    char packet_filter[] = "ether proto \\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;
    }
```

组装要发送的 ARP 报文：

```
//组装报文
    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-
66
        ARPFrame.RecvHa[i] = 0;//设置为0
```

```
        ARPFrame.SendHa[i] = 0x66;
    }
    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
112.112.112.112.112.112
```

用选中的网卡发送报文：

```
pcap_sendpacket(pcap_handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t));
cout << "ARP 请求发送成功" << endl;
```

循环捕获返回的数据包：

```
while (true)
    {
        int rtn = pcap_next_ex(pcap_handle, &pkt_header, &pkt_data);
        if (rtn == -1)
        {
            cout << "  捕获数据包时发生错误：" << errbuf << endl;
            return 0;
        }
        else
        {
            if (rtn == 0)
            {
                cout << "  没有捕获到数据报" << endl;
            }

            else
            {
                IPPacket = (ARPFrame_t*)pkt_data;
                if (IPPacket->RecvIP == SendIP && IPPacket->SendIP == RevIP)//判断是
不是一开始发的包
                {

                    cout << " 捕获到回复的数据报,请求IP与其MAC地址对应关系如下：" <<
endl;
                    printIP(IPPacket->SendIP);
                    cout << "     -----     ";
                    printMAC(IPPacket->SendHa);
```

```
                cout << endl;
                break;
            }
        }
    }
}
```

封装 ARP 请求使用本机网卡的 IP 和 MAC 地址，将本机 IP 和 MAC 填入报文，重新发送 ARP 请求，返回网关和远程主机的 MAC 地址：

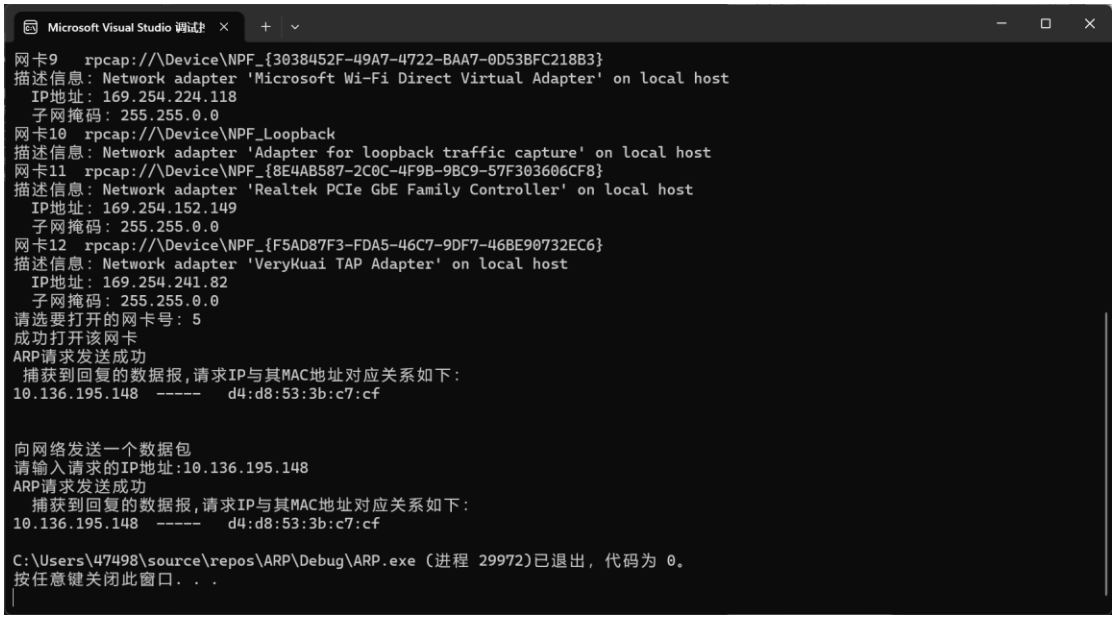
```
//向网络发送数据包
    cout << "\n" << endl;
    cout << "向网络发送一个数据包" << endl;
    cout << "请输入请求的IP地址:";
    char str[15];
    cin >> str;
    RevIP = ARPFrame.RecvIP = inet_addr(str);
    SendIP = ARPFrame.SendIP = IPPacket->SendIP;//将本机IP赋值给数据报的源IP
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.SendHa[i] = ARPFrame.FrameHeader.SrcMAC[i] = IPPacket->SendHa[i];
    }

    if (pcap_sendpacket(pcap_handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0)
    {
        cout << "ARP请求发送失败" << endl;
    }
    else
    {
        cout << "ARP请求发送成功" << endl;

        while (true)
        {
            int n = pcap_next_ex(pcap_handle, &pkt_header, &pkt_data);
            if (n == -1)
            {
                cout << "  捕获数据包时发生错误：" << errbuf << endl;
                return 0;
            }
            else
            {
                if (n == 0)
                {
                    cout << "  没有捕获到数据报" << endl;
                }
```

```
                else
                {
                    IPPacket = (ARPFrame_t*)pkt_data;
                    if (IPPacket->RecvIP == SendIP && IPPacket->SendIP == RevIP)
                    {
                        //  cout<<"  捕获到网络向自己回复的数据报，其信息如下：
"<<endl;
                        //  printARP(IPPacket);
                        cout << "  捕获到回复的数据报,请求IP与其MAC地址对应关系如
下：" << endl;
                        printIP(IPPacket->SendIP);
                        cout << "     -----     ";
                        printMAC(IPPacket->SendHa);
                        cout << endl;
                        break;
                    }
                }
            }
        }
    }
```

（三）程序运行界面展示



Github 链接:

https://github.com/zhayao99/Net_Tech