南 开 大 学

计算机学院

网络技术与应用课程报告

第 5 次实验报告

学号：1911406
姓名：丁彦添
年级：2019 级
专业：计科

2021 年 11 月 13 日

第1节　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：

（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 WinPcap 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。

第2节　实验准备

1. 继续使用实验 2 配置好的 NPcap 环境。NPcap 编程环境配置大体同上一个实验，需要在属性 VC++目录上添加 NPcap 包含目录和 NPcap 库目录；C++关闭 SDL 检查；关闭所有警告；链接器附加依赖项添加 wpcap.lib 和 ws2_32.lib。

2. 结合教材，了解 ARP 的工作原理，实现方式。

3. 通过编程获取 IP 地址与 MAC 地址的对应关系：学习调用 Winpcap 的 pcap_findalldevs_ex()函数获取网络接口设备列表之后，在 alldevs 链表中，按照链表结构利用这个 pcap_findalldevs_ex()函数获取本级网络接口卡和网卡上绑定的 IP 地址。

4. 用 pcap_sendpacket()函数向网络中发送数据包，pcap_sendpacket()函数的各个参数意义：

P：指定 pacp_sendpacket()函数通过那块网卡发送数据包；

Buf：只想需要发送的数据包，需包含各层头部信息；

size：发送数据包的大小。

5. 在测试时，需要另外开一台终端设备进行通信。

第3节　实验过程

项目设计思路：

关键代码分析：

帧首部和 ARP 帧参考教材上的代码

```
typedef struct FrameHeader_t//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;
    WORD ProtocolType;
    BYTE HLen;
    BYTE PLen;
    WORD Operation;
    BYTE SendHa[6];
    DWORD SendIP;
    BYTE RecvHa[6];
    DWORD RecvIP;
}ARPFrame_t;
```

对 MAC 和 IP 地址进行输出的函数，在本次实验中 MAC 地址不能直接使用 cout 进行输出，故对得到的地址进行解码，hex（16 进制），缺省补零的输出，完成输出函数 pMACaddr 和 pIPaddr 函数

```
int pMACaddr(BYTE MACaddr[6])//输出MAC地址
{
    int i = 0;
```

```
    while (i <= 5)
    {
        cout << setw(2) << setfill('0') << hex << (int)MACaddr[i];
        if (i != 5)
            cout << "-";
        else
            cout << endl;
        i++;
    }
    return i;
}
int pIPaddr(DWORD IPaddr)//输出IP地址
{
    BYTE* p = (BYTE*)&IPaddr;
    int i = 0;
    while (i <= 3)
    {
        cout << dec << (int)*p;

        if (i != 3)
            cout << ".";
        else
            cout << endl;
        p++;
        i++;
    }
    return i;
}
```

根据 ARP 帧结构，输出 ARP 帧内容。

```
int pARPframe(ARPFrame_t* IPPacket)//输出ARP帧
{
    cout << "本次捕获得到的ARP帧内容如下：" << endl;
    cout << "目的MAC地址："<< endl;
    pMACaddr(IPPacket->FrameHeader.DesMAC);
    cout << "源MAC地址：" << endl;
    pMACaddr(IPPacket->FrameHeader.SrcMAC);
    //ntoh():将一个16位数由网络字节顺序转换为主机字节顺序
    cout << "帧类型：" << hex << ntohs(IPPacket->FrameHeader.FrameType) << endl;
    cout << "硬件类型：" << hex << ntohs(IPPacket->HardwareType) << endl;
    cout << "协议类型：" << hex << ntohs(IPPacket->ProtocolType) << endl;
    cout << "硬件地址长度：" << hex << (int)IPPacket->HLen << endl;
    cout << "协议地址长度：" << hex << (int)IPPacket->PLen << endl;
    cout << "报文类型：" << hex << ntohs(IPPacket->Operation) << endl;
```

```
    //Operation 表示这个报文的类型，ARP 请求为 1，ARP 响应为 2，RARP 请求为 3，RARP
响应为 4。
    cout << "发送端 MAC 地址: ";
    pMACaddr(IPPacket->SendHa);
    cout << "发送端 IP 地址: ";
    pIPaddr(IPPacket->SendIP);
    cout << "目的端 MAC 地址: ";
    pMACaddr(IPPacket->RecvHa);
    cout << "目的端 IP 地址: ";
    pIPaddr(IPPacket->RecvIP);
    return 0;
}
```

获取设备列表的代码基本和上次实验的一样。为了输出设备列表对应的 IP 地址、网络掩码和广播地址，需要新加上代码用于输出。

```
        a = d->addresses;
JDG:if (a != NULL) //相对第一次试验，增加输出IP地址，掩码，广播地址的代码
        {
            if (a->addr->sa_family == AF_INET)
            {
                cout << "  IP地址: " << inet_ntoa(((struct
sockaddr_in*)(a->addr))->sin_addr) << endl;
                cout << "  网络掩码: " << inet_ntoa(((struct
sockaddr_in*)(a->netmask))->sin_addr) << endl;
                cout << "  广播地址: " << inet_ntoa(((struct
sockaddr_in*)(a->broadaddr))->sin_addr) << endl;
            }
                a = a->next;
                goto JDG;
        }
    }
```

跳到选中的网络适配器和将网卡设为混杂模式的代码和上次实验的一样。但是本次实验只捕获 ARP 数据包，需要在代码中设置

```
    //编写过滤器代码使其只捕获ARP数据包，对错误进行错误处理
    netmask = ((sockaddr_in*)(d->addresses->netmask))->sin_addr.S_un.S_addr;
    struct bpf_program fcode;
    char packet_filter[] = "ether proto \\arp";
    if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) < 0)
    {
        cout << "编译数据包过滤器时遇到错误！请检查过滤器语法" << endl;
        pcap_freealldevs(alldevs);
```

```
    throw -5;
}
if (pcap_setfilter(adhandle, &fcode) < 0)
{
    cout << "设置过滤器时发生错误！"<<endl;
    pcap_freealldevs(alldevs);
    throw -6;
}
```

使用巧妙的方式获取本机的 MAC 地址和 IP 地址。获取主机网卡中对应 IP 的 MAC 地址，可以利用 ARP 请求方法，过程如下：

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：
3. ARP 请求
4. 广播
5. 伪造源 MAC 地址和源 IP 地址
6. 目的 IP 地址为网卡 A 的 IP 地址
7. 用网卡 A 发送报文 S
8. 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

```
//预处理将要发送的 ARP 数据报
for (i = 0; i < 6; i++)
{
    ARPFrame.FrameHeader.DesMAC[i] = 0xFF;//本机广播地址
    ARPFrame.FrameHeader.SrcMAC[i] = 0x00;//设置为任意 MAC 地址
    ARPFrame.RecvHa[i] = 0;//置0
    ARPFrame.SendHa[i] = 0x11;//设置为任意 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);
    SIP = ARPFrame.SendIP = htonl(0x00000000);//设置为任意 IP 地址
    //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)
        {
            ReIP = ARPFrame.RecvIP = inet_addr(inet_ntoa(((struct
sockaddr_in*)(a->addr))->sin_addr));
        }
    }
    struct pcap_pkthdr* adhandleheader;
    const u_char* adhandledata;
    int tjdg = 0;
    if (pcap_sendpacket(adhandle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0)
    {
        cout << "ARP数据包发送失败！" << endl;
        pcap_freealldevs(alldevs);
        throw -7;
    }
    else
    {
        cout << "ARP数据包发送成功！" << endl;
GET_LOCAL_MAC:  int jdg_catch_re_arp_p = pcap_next_ex(adhandle,
&adhandleheader, &adhandledata);
        if (jdg_catch_re_arp_p == -1)
        {
            cout << "捕获ARP返回数据包时发生错误！" << endl;
            pcap_freealldevs(alldevs);
            throw -8;
        }
        else
            if (jdg_catch_re_arp_p == 0)
            {
                cout << "暂未获得数据报，请稍候" << endl;
                cout << "已尝试次数：" << ++tjdg << endl;
                if (tjdg > 20)
                {
                    cout << "已多次尝试接收，请确认端口是否正常" << endl;
                    pcap_freealldevs(alldevs);
                    throw -9;
```

```
                }
                goto GET_LOCAL_MAC;
            }
            else
            {
                IPPacket = (ARPFrame_t*)adhandledata;
                if(SIP==IPPacket->SendIP)
                    if (ReIP == IPPacket->RecvIP)
                    {
                        cout << "确认正常！" << endl;
                        goto GET_LOCAL_MAC;
                    }
                if(SIP == IPPacket->RecvIP)
                    if (ReIP == IPPacket->SendIP)
                    {
                        cout << "成功获取回复的数据报！" << endl;
                        pARPframe(IPPacket);
                        cout << endl;
                        cout << "获取到本机IP地址与MAC地址的对应关系如下：" <<
endl << "IP：";
                        pIPaddr(IPPacket->SendIP);
                        cout << "MAC：";
                        pMACaddr(IPPacket->SendHa);
                        cout << endl;
                    }
                    else
                        goto GET_LOCAL_MAC;
                else
                    goto GET_LOCAL_MAC;
            }
    }
```

使用标签 GET_DISTANT_MAC 和 goto 语句，对未捕获到数据报的情况进行计时，时间过长未捕获到数据包则提示超时并抛出异常，提醒用户程序或端口可能存在问题。（在实验中一般不会出现问题）若捕获到远端的 MAC 地址，直接输出并结束程序。

```
    cout << "向网络发送数据包" << endl;
    char pip[16];
    cout << "请输入目的IP地址" << endl;
    cin >> pip;
    ReIP = ARPFrame.RecvIP = inet_addr(pip);
```

```
    SIP = ARPFrame.SendIP = IPPacket->SendIP;
    for (i = 0; i < 6; i++)
    {
        ARPFrame.SendHa[i] = ARPFrame.FrameHeader.SrcMAC[i] =
IPPacket->SendHa[i];
    }
    if (pcap_sendpacket(adhandle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0)
    {
        cout << "ARP数据包发送失败！" << endl;
        pcap_freealldevs(alldevs);
        throw -10;
    }
    else
    {
        cout << "ARP数据包发送成功！" << endl;
        inum = 0;
GET_DISTANT_MAC:int jdg_catch_re_arp_p = pcap_next_ex(adhandle,
&adhandleheader, &adhandledata);
        if (jdg_catch_re_arp_p == -1)
        {
            cout << "捕获ARP返回数据包时发生错误！" << endl;
            pcap_freealldevs(alldevs);
            throw -11;
        }
        else
            if (jdg_catch_re_arp_p == 0)
            {
                cout << "暂未获得数据报，请稍候" << endl;
                cout << "已尝试次数：" <<dec<< ++inum << endl;
                if (inum > 20)
                {
                    cout << "已多次尝试接收，请确认端口是否正常" << endl;
                    pcap_freealldevs(alldevs);
                    throw -12;
                }
                goto GET_DISTANT_MAC;
            }
            else
            {
                IPPacket = (ARPFrame_t*)adhandledata;
                if (SIP == IPPacket->SendIP)
                    if (ReIP == IPPacket->RecvIP)
                    {
                        cout << "确认正常！" << endl;
```

```
                    goto GET_DISTANT_MAC;
                }
            if (SIP == IPPacket->RecvIP)
                if (ReIP == IPPacket->SendIP)
                {
                    cout << "成功获取回复的数据报！" << endl;
                    pARPframe(IPPacket);
                    cout << endl;
                    cout << "获取到IP地址与MAC地址的对应关系如下：" << endl <<
"IP：";
                    pIPaddr(IPPacket->SendIP);
                    cout << "MAC：";
                    pMACaddr(IPPacket->SendHa);
                    cout << endl;
                }
                else
                    goto GET_DISTANT_MAC;
            else
                goto GET_DISTANT_MAC;
        }
    }
    return i;
```

第4节　实验操作过程

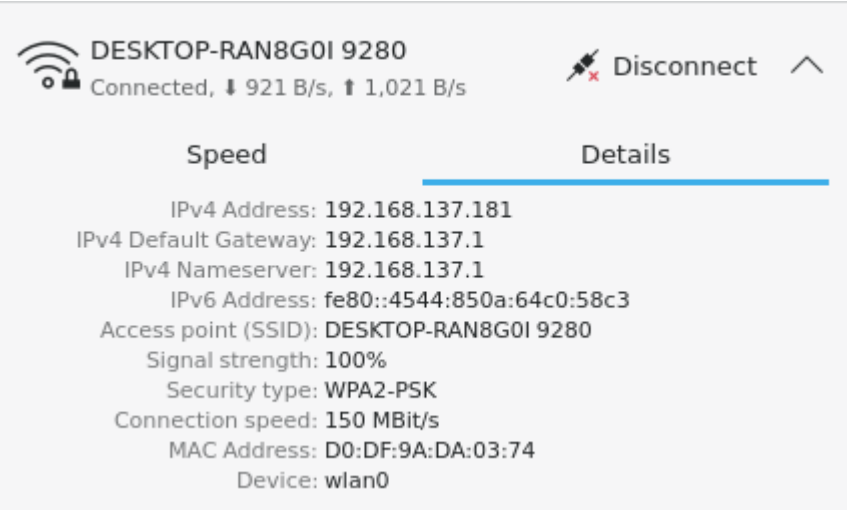
打开本机热点，使用另一台计算机连接此热点。

在本机的命令提示符中输入 ipconfig /all 查看专用热点的网卡，得到结果如下：

```
无线局域网适配器 本地连接* 2:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
   物理地址. . . . . . . . . . . . . : C2-23-43-D7-66-4F
   DHCP 已启用 . . . . . . . . . . . : 否
   自动配置已启用. . . . . . . . . . : 是
   IPv4 地址 . . . . . . . . . . . . : 192.168.137.1(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :
   TCPIP 上的 NetBIOS . . . . . . . : 已启用
```

在另一台计算机的网络连接设置上可以看到其 ip 地址和 MAC 地址，具体为 192.168.137.181 和 D0-DF-9A-DA-03-74。



运行代码编译出的程序，可以看到获取到的端口设备列表，选择无线热点网卡对应的编号，在测试时是 5 号。

```
2 rpcap://\Device\NPF_{BABD972E-EE2B-4223-B0D8-5E6583E9E15D}
Network adapter 'WAN Miniport (IPv6)' on local host
3 rpcap://\Device\NPF_{0BC18797-99B9-4EB6-80D6-01129BA7160B}
Network adapter 'WAN Miniport (IP)' on local host
4 rpcap://\Device\NPF_{C78B61E6-1364-4AFD-8E27-FE68F47C19A1}
Network adapter 'Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC' on local host
  IP地址: 10.130.182.140
5 rpcap://\Device\NPF_{F653009C-4D70-4614-8D32-AB17789B9EBC}
Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host
  IP地址: 192.168.137.1
6 rpcap://\Device\NPF_{94BCEE79-5354-4109-8150-C5D7F55002AE}
Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter' on local host
  IP地址: 169.254.218.234
7 rpcap://\Device\NPF_{7C53B443-28DB-484D-AE46-0D2B135508DD}
Network adapter 'VirtualBox Host-Only Ethernet Adapter' on local host
  IP地址: 192.168.56.1
8 rpcap://\Device\NPF_Loopback
Network adapter 'Adapter for loopback traffic capture' on local host
9 rpcap://\Device\NPF_{431C2C7A-077A-460B-9BD1-D712C7A4A7A3}
Network adapter 'Netease UU TAP-Win32 Adapter V9.21' on local host
  IP地址: 169.254.247.125
10 rpcap://\Device\NPF_{1D223EEF-EF26-4AE0-9E51-CA152363B6F6}
Network adapter 'TAP-Windows Adapter V9 #5' on local host
  IP地址: 169.254.68.141
11 rpcap://\Device\NPF_{028F9220-3492-4510-AD32-9B440EE79ECC}
Network adapter 'TAP-Windows Adapter V9 #4' on local host
  IP地址: 169.254.112.61
12 rpcap://\Device\NPF_{CE684674-ACA1-4AD0-A807-C7FE7A1877B5}
Network adapter 'TAP-Windows Adapter V9 #3' on local host
```

可以看到接着发送 ARP 请求请求本机网络接口上绑定的 IP 地址和 MAC 地址。

```
Enter the interface number:(range:1-14)
请输入进入的端口号:(范围：1-14)
5
接入端口：5 Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host
ARP数据包发送成功!
确认正常!
成功获取回复的数据报!
本次捕获得到的ARP帧内容如下：
目的MAC地址：
11-11-11-11-11-11
源MAC地址：
c2-23-43-d7-66-4f
帧类型：806
硬件类型：1
协议类型：800
硬件地址长度：6
协议地址长度：4
报文类型：2
发送端 MAC 地址：c2-23-43-d7-66-4f
发送端 IP 地址：192.168.137.1
目的端 MAC 地址：11-11-11-11-11-11
目的端 IP 地址：0.0.0.0
```

通过巧妙的方法获取到了本机的 IP 地址和 MAC 地址。接着向热点中的另一台终端设备发送 ARP 协议，请求其 MAC 地址。

```
向网络发送数据包
请输入目的IP地址
192.168.137.181
ARP数据包发送成功!
成功获取回复的数据报!
本次捕获得到的ARP帧内容如下：
目的MAC地址：
ff-ff-ff-ff-ff-ff
源MAC地址：
d0-df-9a-da-03-74
帧类型：806
硬件类型：1
协议类型：800
硬件地址长度：6
协议地址长度：4
报文类型：1
发送端 MAC 地址：d0-df-9a-da-03-74
发送端 IP 地址：192.168.137.181
目的端 MAC 地址：00-00-00-00-00-00
目的端 IP 地址：192.168.137.1

获取到IP地址与MAC地址的对应关系如下：
IP：192.168.137.181
MAC：d0-df-9a-da-03-74
```

比对程序运行结果，可以发现请求到的 IP 和 MAC 地址正确无误。