

网络技术与应用课程实验报告

实验名称：编程获取 IP 地址与 MAC 地址的对应关系

学号： 2211489 姓名： 冯佳明 专业： 物联网工程

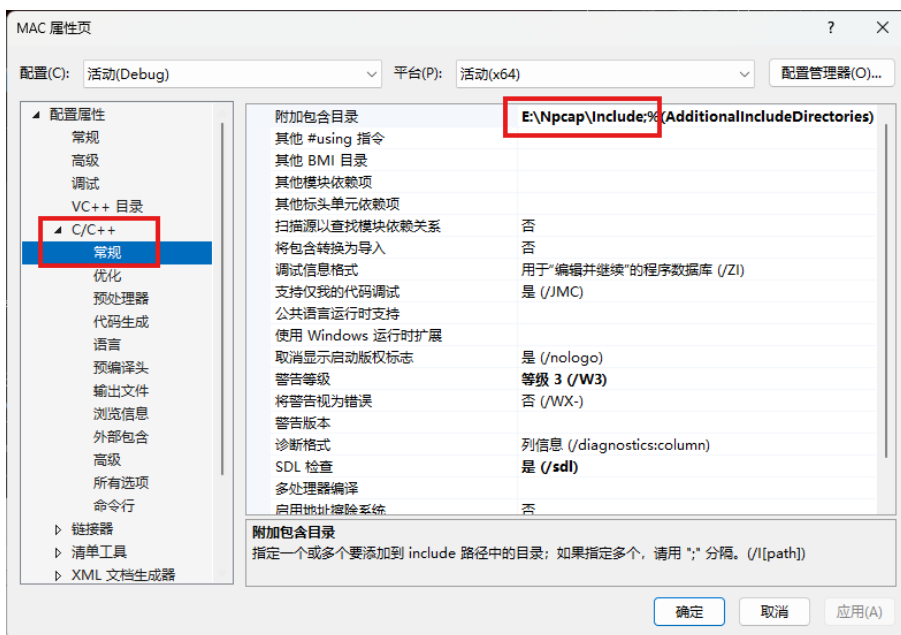
一、实验要求：通过编程获取 IP 地址与 MAC 地址的对应关系

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。

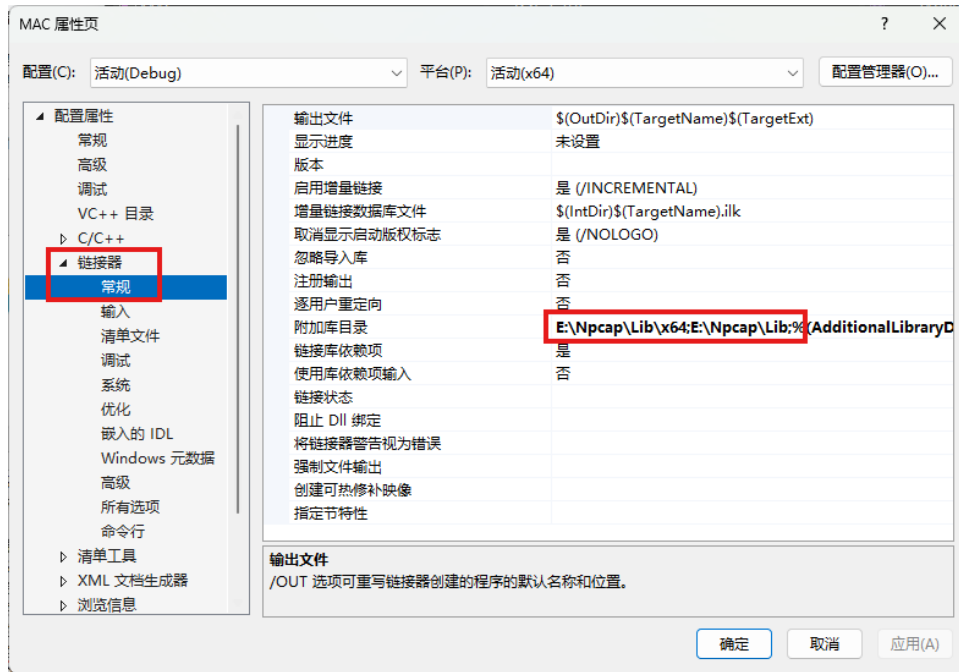
二、实验步骤

1. 准备工作：环境配置

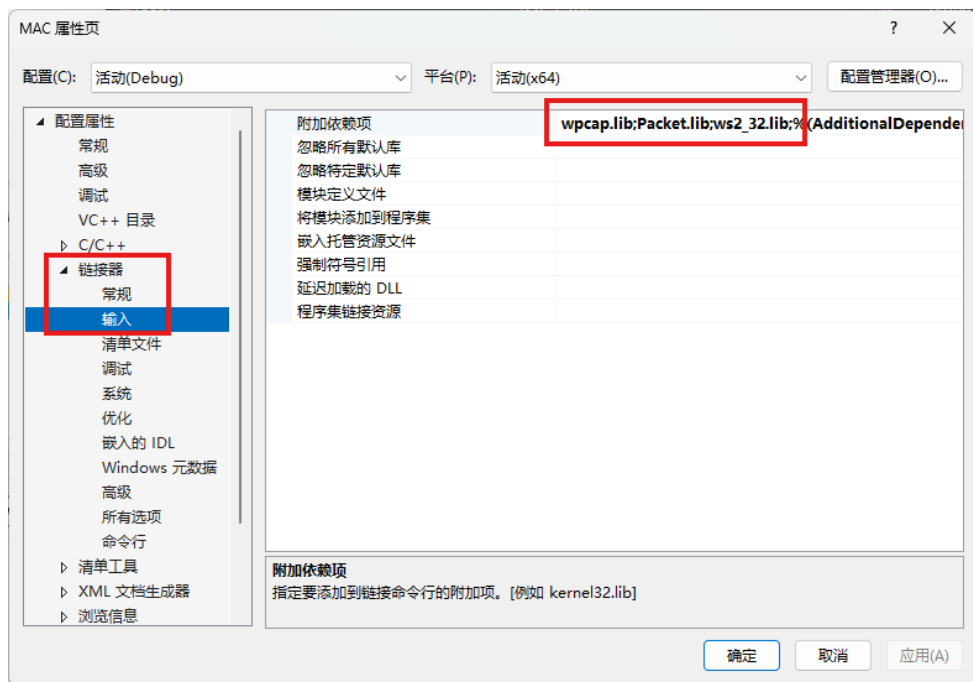
- (1) 添加包含文件目录：项目 - 属性 - 配置属性 - C/C++ - 常规 - 附加包含目录



- (2) 添加库文件目录：项目 - 属性 - 配置属性 - 连接器 - 常规 - 附加库目录



(3) 添加链接时使用的库文件



2. 帧结构定义：依据以太网中 ARP 报文格式，对 ARP 帧结构进行定义

硬件类型：以太网接口类型值为 1

协议类型：IP 协议类型为 0800₍₁₆₎

硬件地址长度：MAC 地址长度为 6B

协议地址长度：IP 地址长度为 4B

操作：ARP 请求为 1，ARP 应答为 2

源 MAC 地址：发送方的 MAC 地址

源 IP 地址：发送方的 IP 地址

目的 MAC 地址：ARP 请求中该字段没有意义，ARP 响应中为接收方的 MAC 地址

目的 IP 地址：ARP 请求中为请求解析的 IP 地址，ARP 响应中为接收方的 IP 地址

0		15		16		31	
硬件类型				协议类型			
硬件地址长度		协议地址长度		操作			
源MAC地址（0-3）							
源MAC地址（4-5）				源IP地址（0-1）			
源IP地址（2-3）				目的MAC地址（0-1）			
目的MAC地址（2-5）							
目的IP地址（0-3）							

```
18 //定义帧首部
19 typedef struct FrameHeader_t {
20     BYTE DesMAC[6]; //目的地址
21     BYTE SrcMAC[6]; //源地址
22     WORD FrameType; //帧类型
23 }FrameHeader_t;
24
25 //定义IP首部
26 typedef struct IPHeader_t {
27     BYTE Ver_HLen; //IP版本和头部长度
28     BYTE TOS; //服务类型
29     WORD TotallLen; //总长度
30     WORD ID; //标识
31     WORD Flag_Segment; //片偏移
32     BYTE TTL; //生存时间
33     BYTE Protocol; //协议
34     WORD Checksum; //首部校验和
35     ULONG SrcIP; //源IP
36     ULONG DstIP; //目的IP
37 }IPHeader_t;
38
39 //定义包含帧首部和IP首部的数据包
40 typedef struct Data_t {
41     FrameHeader_t FrameHeader;
42     IPHeader_t IPHeader;
43 }Data_t;
```

```
45 //定义ARP帧
46 typedef struct ARPFrame_t {
47     FrameHeader_t FrameHeader;
48     WORD HardwareType; //硬件类型
49     WORD ProtocolType; //协议类型
50     BYTE HLen; //硬件地址长度
51     BYTE PLen; //协议地址长度
52     WORD Operation; //操作
53     BYTE SendHa[6]; //源MAC地址
54     DWORD SendIP; //源IP地址
55     BYTE RecvHa[6]; //目的MAC地址
56     DWORD RecvIP; //目的IP地址
57 }ARPFrame_t;
58
59 #pragma pack() //恢复缺省对齐方式
```

3. 获取本机网络接口（同 IP 数据包捕获与分析实验，不再赘述）

```
164 int main()
165 {
166     int i = 0;
167     int num = 0; //接口序号
168     pcap_if_t* alldevs; //指向设备链表首部的指针
169     pcap_if_t* d;
170     pcap_addr_t* a; //表示接口地址指针
171     char errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
172
173     //获取当前网卡列表
174     if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, //获取本机的接口设备
175         NULL, //无需认证
176         &alldevs, //指向设备列表首部
177         errbuf //出错信息保存缓冲区
178     ) == -1)
179     {
180         cout << "获取本机网卡列表时出错:" << errbuf << endl;
181         return 1;
182     }
183
184     //打印网卡的列表
185     for (d = alldevs; d != NULL; d = d->next)
186     {
187         num++;
188         //打印网络接口设备的名字和描述信息
189         cout << num << ". " << d->name << "->" << d->description << ";" << endl;
190     }
191 }
```

4. 选择打开的设备，并打印设备的详细信息（同 IP 数据包捕获与分析实验，不再赘述）

```
192 //选择网卡
193 cout << "请选择设备 (1-" << num << "):" << endl;
194 int dev_select_num = 0;
195 cin >> dev_select_num;
196
197 while (dev_select_num < 1 || dev_select_num > num)
198 {
199     cout << "字符非法，请重新输入 (1-" << num << "):" << endl;
200     cin >> dev_select_num;
201     if (dev_select_num >= 1 && dev_select_num <= num)
202     {
203         break;
204     }
205 }
206
207 //转到选择的设备
208 for (d = alldevs, i = 0; i < dev_select_num - 1; i++)
209 {
210     d = d->next;
211 }
212
213 //打印选择的设备的详细信息
214 cout << "您选择的设备信息为: " << d->name << ";" << endl;
215 cout << "描述信息: " << d->description << endl;
216
217 for (a = d->addresses; a != NULL; a = a->next)
218 {
219     if (a->addr->sa_family == AF_INET)
220     {
221         cout << "IP地址: " << inet_ntoa(((struct sockaddr_in*)a->addr)->sin_addr) << endl;
222         HostIP = inet_ntoa(((struct sockaddr_in*)a->addr)->sin_addr);
223
224         cout << "子网掩码: " << inet_ntoa(((struct sockaddr_in*)a->netmask)->sin_addr) << endl;
225         cout << "广播地址: " << inet_ntoa(((struct sockaddr_in*)a->broadaddr)->sin_addr) << endl;
226         HostBroadaddr = inet_ntoa(((struct sockaddr_in*)a->broadaddr)->sin_addr);
227         cout << endl;
228         break;
229     }
230 }
```

5. 打开所选择的设备（同 IP 数据包捕获与分析实验，不再赘述）

```
232 //打开网络接口
233 pcap_t* handle = pcap_open_live(d->name, BUFSIZ, 1, 1000, errbuf);
234 if (pcap_open == NULL)
235 {
236     cout << "打开设备" << dev_select_num << "的网络接口失败：" << errbuf << endl;
237     return 1;
238 }
```

6. 初始化 ARP 数据包：设置相关参数

```
240 //定义发送的ARP包并初始化
241 ARPFrame_t ARPFrame;
242
243 ARPFrame.FrameHeader.FrameType = htons(0x8086); //帧类型为ARP
244 ARPFrame.HardwareType = htons(0x0001); //硬件类型为以太网
245 ARPFrame.ProtocolType = htons(0x0800); //协议类型为IP
246 ARPFrame.HLen = 6; //硬件地址长度为6
247 ARPFrame.PLen = 4; //协议地址长度为4
248 ARPFrame.Operation = htons(0x0001); //操作为ARP请求
```

7. 获取本机 MAC 地址：构造 ARP 数据包，将目的 IP 地址设置为选择的接口的 IP 地址，发送 ARP 数据包，并循环捕获返回的 ARP 响应包；响应包中的源 IP 地址如果与源 IP 地址一致，则获取响应包中的源 MAC 地址并输出

```
62 void get_host_mac(pcap_t* handle, ARPFrame_t ARPFrame)
63 {
64     struct pcap_pkthdr* packet_header;
65     const u_char* packet_data;
66     Data_t* packet;
67
68     for (int i = 0; i < 6; i++)
69     {
70         ARPFrame.FrameHeader.DesMAC[i] = 0xff; //目的Mac地址设置为广播地址
71         ARPFrame.FrameHeader.SrcMAC[i] = HostMAC[i]; //源MAC地址设置为本机网卡MAC地
72         ARPFrame.SendHa[i] = i;
73         ARPFrame.RecvHa[i] = 0x00; //目的MAC地址设置为0
74     }
75     ARPFrame.SendIP = inet_addr("10.10.10.10"); //源IP地址随便设置一个
76     ARPFrame.RecvIP = inet_addr(HostIP.c_str()); //目的IP地址为本机地址
77
78     //发送数据包
79     if (pcap_sendpacket(handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0);
80
81     //抓取自己发出的包，从而获得自己的MAC地址
82     int res = 0;
83     while ((res = pcap_next_ex(handle, &packet_header, &packet_data)) >= 0)
84     {
85         if (res == 0) continue; //接收数据包超时
86
87         packet = (Data_t*)packet_data;
88         if (inet_ntoa(*(in_addr*)&packet->IPHeader.SrcIP) == HostIP)
89         {
90             for (int i = 0; i < 6; i++)
91             {
92                 HostMAC[i] = packet->FrameHeader.SrcMAC[i];
93             }
94             cout << "本机接口IP地址对应的MAC地址：\n" << HostIP;
95             printf(" --> %02X-%02X-%02X-%02X-%02X-%02X\n\n",
96                 (unsigned int)HostMAC[0],
97                 (unsigned int)HostMAC[1],
98                 (unsigned int)HostMAC[2],
99                 (unsigned int)HostMAC[3],
100                 (unsigned int)HostMAC[4],
101                 (unsigned int)HostMAC[5]);
102             break;
103         }
104     }
105 }
106 }
```

8. 获取输入的 IP 地址所对应的 MAC 地址：处理逻辑与获取本机 MAC 地址类似，将 ARP 请求发送给目标 IP，收到目标的 ARP 响应后，提取其中的源 MAC 地址并输出

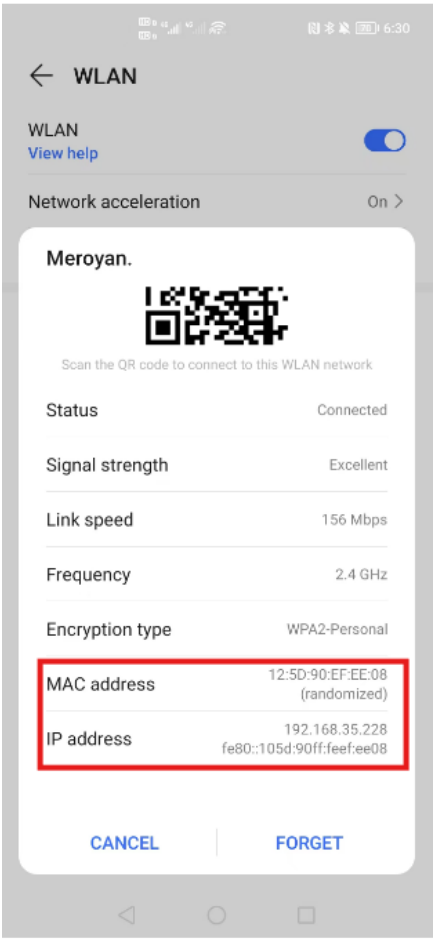
```
108 void GetMAC(pcap_t* adhandle, ARPFrame_t ARPFrame)
109 {
110     struct pcap_pkthdr* packet_header;
111     const u_char* packet_data;
112     Data_t* packet;
113
114     cout << "请输入IP地址: ";
115     string IP;
116     cin >> IP;
117
118     //ARP数据包
119     for (int i = 0; i < 6; i++)
120     {
121         ARPFrame.FrameHeader.DesMAC[i] = 0xff; //目的Mac地址设置为广播地址
122         ARPFrame.FrameHeader.SrcMAC[i] = HostMAC[i]; //源MAC地址设置为本机网卡MAC地
123         ARPFrame.SendHa[i] = HostMAC[i]; //源MAC地址设置为本机网卡MAC地址
124         ARPFrame.RecvHa[i] = 0x00; //目的MAC地址设置为0
125     }
126     ARPFrame.SendIP = inet_addr(HostIP.c_str()); //本机网卡IP地址
127     ARPFrame.RecvIP = inet_addr(IP.c_str()); //目的IP地址
128
129     //发送数据包
130     pcap_sendpacket(adhandle, (u_char*)&ARPFrame, sizeof(ARPFrame_t));
```

```
132     //监听目的IP为本机IP，原始IP为输入IP的数据包
133     int res = 0;
134     while ((res = pcap_next_ex(adhandle, &packet_header, &packet_data)) >= 0)
135     {
136         if (res == 0) continue; //接收数据包超时
137
138         ARPFrame_t* ARP_Packet = (ARPFrame_t*)packet_data;
139         string desIP = inet_ntoa(*(in_addr*)&ARP_Packet->RecvIP);
140         string srcIP = inet_ntoa(*(in_addr*)&ARP_Packet->SendIP);
141         if ((desIP == HostIP) && (srcIP == IP))
142         {
143             BYTE MACs[6];
144             for (int i = 0; i < 6; i++)
145             {
146                 MACs[i] = ARP_Packet->FrameHeader.SrcMAC[i];
147             }
148
149             cout << IP;
150             printf(" --> %02X-%02X-%02X-%02X-%02X-%02X\n\n",
151                 (unsigned int)MACs[0],
152                 (unsigned int)MACs[1],
153                 (unsigned int)MACs[2],
154                 (unsigned int)MACs[3],
155                 (unsigned int)MACs[4],
156                 (unsigned int)MACs[5]);
157
158             break;
159         }
160     }
161 }
```

9. 运行结果展示

我将舍友的手机与我自己的电脑一同连到了我手机的热点之下，运行程序，选择 wifi 接口 4，可以看到返回了该接口的物理地址

随后输入手机的 ip 地址 192.168.35.228，查询手机对应的 mac 地址



运行结果如下图所示，返回结果与手机显示的 MAC 地址一致，证明程序功能正确

```
D:\Code\C++\WangJ\MAC\w
1. rpcap://Device\NPF_{B0FF5B12-1655-415A-B84A-BFE3C34E3C25}->Network adapter 'WAN Miniport (Network Monitor)' on local host;
2. rpcap://Device\NPF_{9775282B-D5FB-487C-B563-0952FE67DBB4}->Network adapter 'WAN Miniport (IPv6)' on local host;
3. rpcap://Device\NPF_{987A8693-8FC6-4F9A-9E66-89EA6DA760B5}->Network adapter 'WAN Miniport (IP)' on local host;
4. rpcap://Device\NPF_{93F890BF-9B73-406B-9F66-730414C3049F}->Network adapter 'Intel(R) Wi-Fi 6E AX211 160MHz' on local host;
5. rpcap://Device\NPF_{B754A8F1-CBF1-41A7-9F07-91B44E924B41}->Network adapter 'VMware Virtual Ethernet Adapter for VMnet8 #2' on local host;
6. rpcap://Device\NPF_{A9C32118-3F34-486B-8CA9-3EA2E9AD2745}->Network adapter 'VMware Virtual Ethernet Adapter for VMnet1 #2' on local host;
7. rpcap://Device\NPF_{634059BE-0654-4F70-BFE6-9156F8412245}->Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host;
8. rpcap://Device\NPF_{11D97A15-FAE2-41CC-9CB9-D86F476DCD3D}->Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter' on local host;
9. rpcap://Device\NPF_Loopback->Network adapter 'Adapter for loopback traffic capture' on local host;
请选择设备 (1-9):
4
您选择的设备信息为: rpcap://Device\NPF_{93F890BF-9B73-406B-9F66-730414C3049F};
描述信息: Network adapter 'Intel(R) Wi-Fi 6E AX211 160MHz' on local host
IP地址: 192.168.35.91
子网掩码: 255.255.255.0
广播地址: 192.168.35.255

本机接口IP地址对应的MAC地址:
192.168.35.91 --> F8-9E-94-EA-19-2E

请输入IP地址: 192.168.35.228
192.168.35.228 --> 12-5D-90-EF-EE-08

请输入IP地址: |
```

在 wireshark 中，设置过滤为 arp，可以看到捕获的数据包如下

1745	28.779511	Intel_ea:19:2e	ea:9b:19:5d:56:d9	ARP	42 192.168.35.91 is at f8:9e:94:ea:19:2e
1828	34.834447	Intel ea:19:2e	12:5d:90:ef:ee:08	ARP	42 Who has 192.168.35.228? Tell 192.168.35.91
1834	34.938578	12:5d:90:ef:ee:08	Intel ea:19:2e	ARP	42 192.168.35.228 is at 12:5d:90:ef:ee:08

三行数据依次为：

主机响应 ARP，返回本机的 MAC 地址；

主机发送 ARP 请求，询问手机的 MAC 地址；

手机响应 ARP，返回手机的 MAC 地址