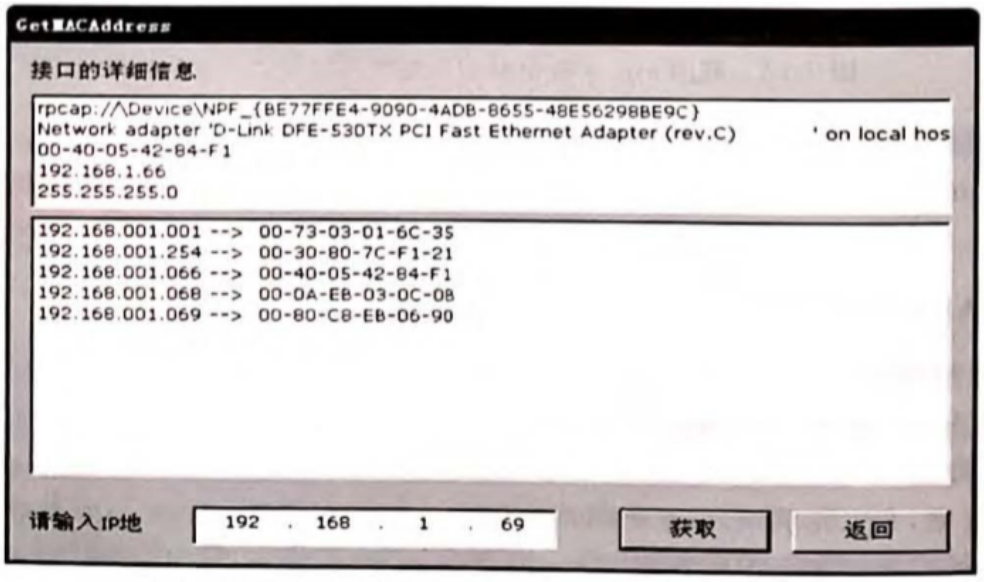
网络技术与应用-ARP数据包捕获 实验报告

1 实验目的

1. 学习获取以太网中主机的MAC地址
2. 掌握ARP数据报格式
3. 深入了解IP地址和MAC地址的有关概念

2 实验要求

1. 利用WinPcap 提供的功能获取以太网中主机的 MAC 地址。
2. 本实验使用的以太网既可以是共享式以太网，也可以是交换式以太网。
3. 所写的程序必须可以在实验室机器的环境下独立运行，现场测试时不接受调试项目文件，只允许运行exe程序。（配置：Windows 7 SP1 32位操作系统，已安装Winpcap）
4. 程序需拥有可视化界面，可以用Qt/MFC等框架。参考界面如下：



3 实验环境

Windows 10 、visual studio 2019 preview、MFC、release X86

4 实验原理

4.1 ARP运行机制

在A不知道B的MAC地址的情况下，A就广播一个ARP请求包，请求包中填有B的IP(192.168.1.2)，以太网中的所有计算机都会接收这个请求，而正常的情况下只有B会给出ARP应答包，包中就填充上了B的MAC地址，并回复给A。

A得到ARP应答后，将B的MAC地址放入本机缓存，便于下次使用。

本机MAC缓存是有生存期的，生存期结束后，将再次重复上面的过程。

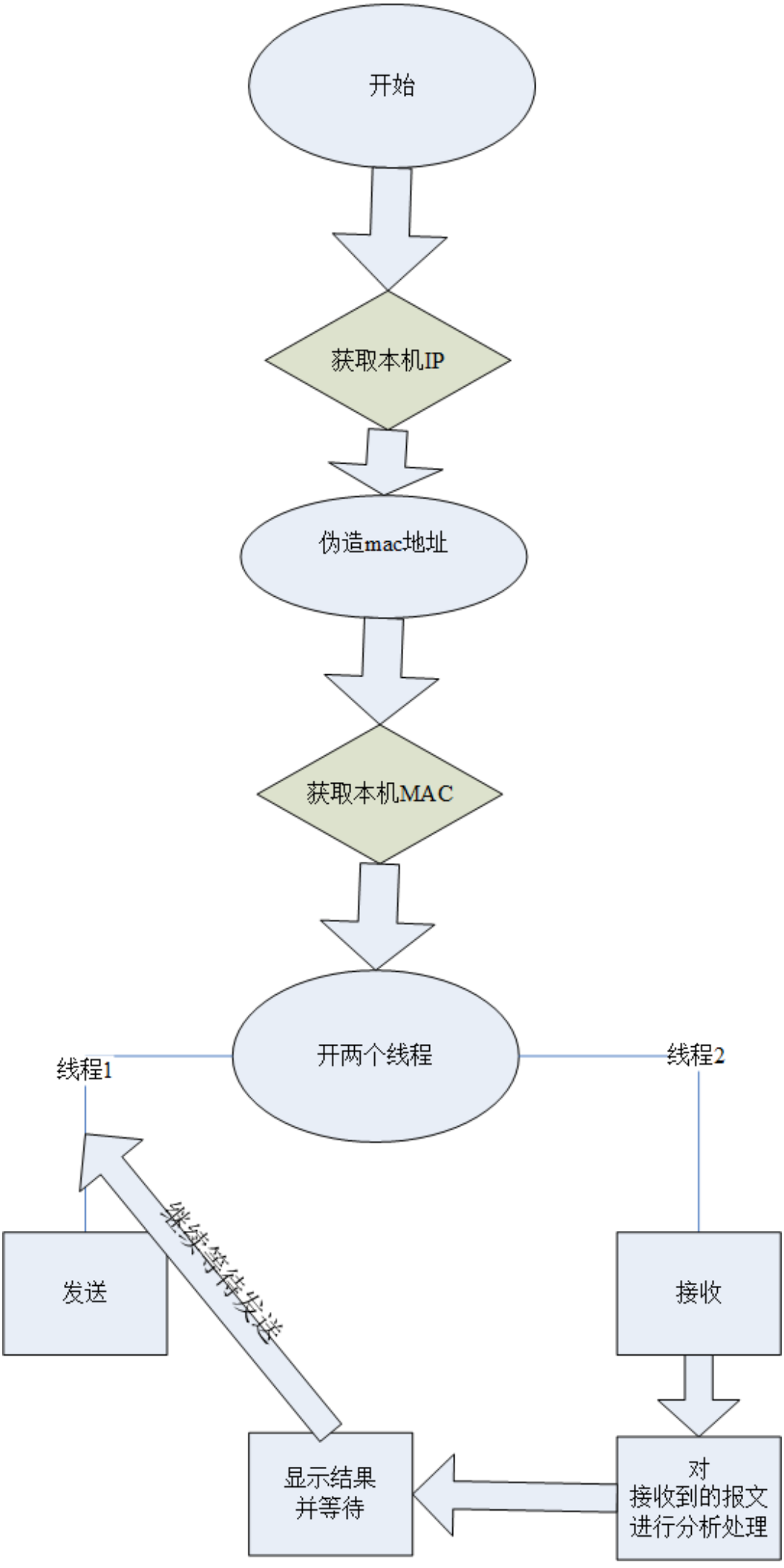
ARP协议并不只在发送了ARP请求才接收ARP应答。当计算机接收到ARP应答数据包的时候，就会对本地的ARP缓存进行更新，将应答中的IP和MAC地址存储在ARP缓存中。因此，当局域网中的某台机器B向A发送一个自己伪造的ARP应答，而如果这个应答是B冒充C伪造来的，即IP地址为C的IP，而MAC地址是伪造的，则当A接收到B伪造的ARP应答后，就会更新本地的ARP缓存，这样在A看来C的IP地址没有变，而它的MAC地址已经不是原来那个了。由于局域网的网络流通不是根据IP地址进行，而是按照MAC地址进行传输。所以，那个伪造出来的MAC地址在A上被改变成一个不存在的MAC地址，这样就会造成网络不通，导致A不能Ping通C！这就是一个简单的ARP欺骗。

在网络执法官中，要想限制某台机器上网，只要点击“网卡”菜单中的“权限”，选择指定的网卡号或在用户列表中点击该网卡所在行，从右键菜单中选择“权限”，在弹出的对话框中即可限制该用户的权限。对于未登记网卡，可以这样限定其上线：只要设定好所有已知用户（登记）后，将网卡的默认权限改为禁止上线即可阻止所有未知的网卡上线。使用这两个功能就可限制用户上网。其原理是通过ARP欺骗发给被攻击的电脑一个假的网关IP地址对应的MAC，使其找不到网关真正的MAC地址，这样就可以禁止其上网。

4.2 ARP数据报格式

- 硬件类型：占两字节，表示ARP报文可以在哪种类型的网络上传输，值为1时表示为以太网地址。
- 上层协议类型：占两字节，表示硬件地址要映射的协议地址类型，映射IP地址时的值为0x0800。
- MAC地址长度：占一字节，标识MAC地址长度，以字节为单位，此处为6。
- IP协议地址长度：占一字节，标识IP得知长度，以字节为单位，此处为4。
- 操作类型：占2字节，指定本次ARP报文类型。1标识ARP请求报文，2标识ARP应答报文。
- 源MAC地址：占6字节，标识发送设备的硬件地址。
- 源IP地址：占4字节，标识发送方设备的IP地址。
- 目的MAC地址：占6字节，表示接收方设备的硬件地址，在请求报文中该字段值全为0，即00-00-00-00-00-00，表示任意地址，因为现在不知道这个MAC地址。
- 目的IP地址：占4字节，表示接受方的IP地址。

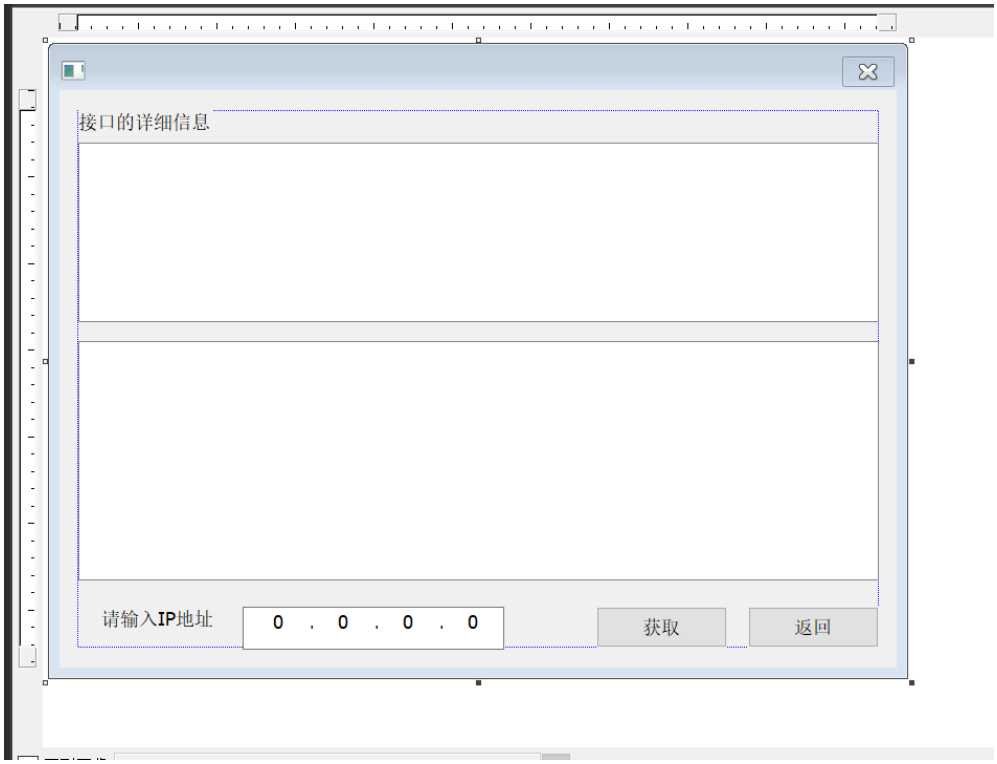
5 实验设计思路


开始
获取本机IP
伪造mac地址
获取本机MAC
开两个线程
线程1
线程2
发送
继续等待发送
接收
显示结果
并等待
对
接收到的报文
进行分析处理

6 程序实现步骤

6.1 创建MFC应用程序

1. 编译环境修改为release X86。
2. 在资源视图中构建以下界面。



3. 修改控件的ID与其他属性值，这里不再一一赘述。

6.2 创建基于WinPcap的应用程序

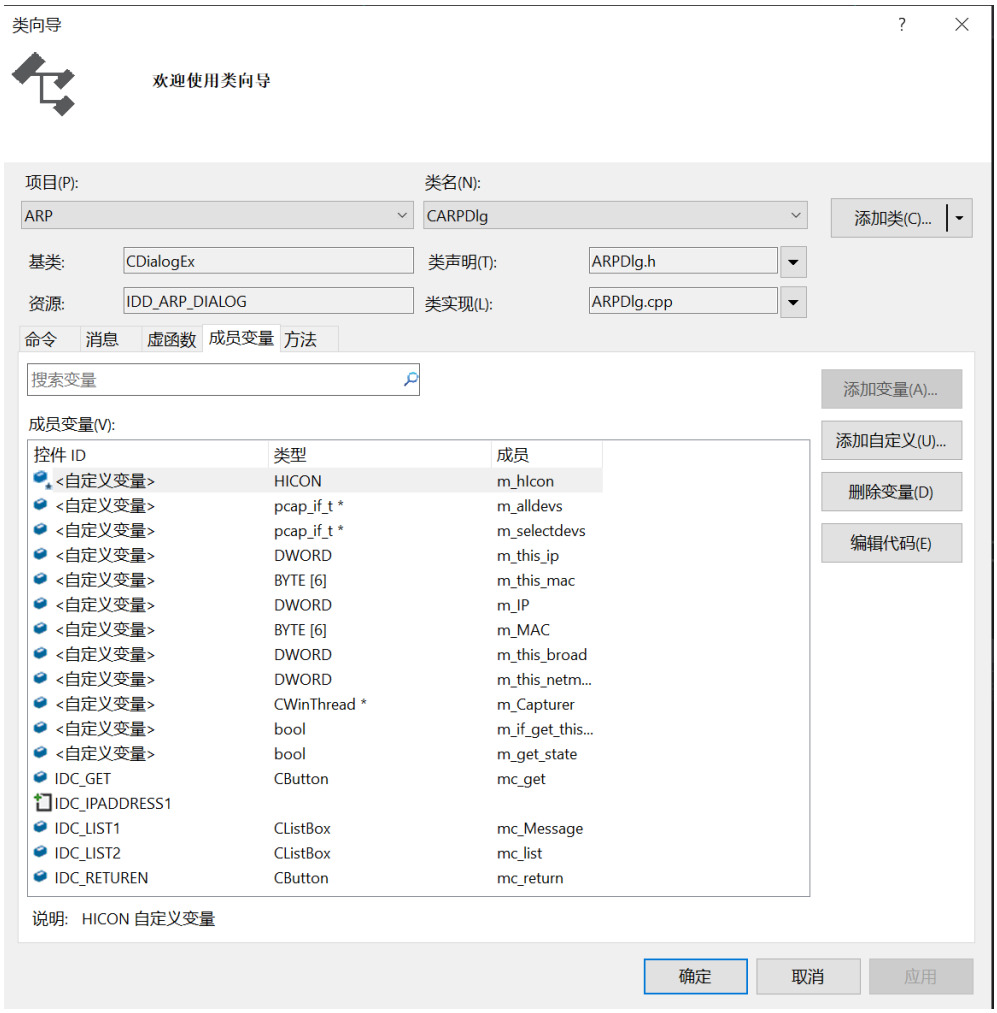
1. 添加 pcnp. h 包含文件：如果一个源文件使用了 WinPcap 提供的函数，那么就能 要在该文件的开始位置增加pcap. h 包含文件。
2. 增加与 WinPcap 有关的预处理都定义：需要将 WPCAP 和HAVE_REMOTE 两个标号添加到预处理指定义中。
3. 把WinPcap的开发者工具包里的包含文件目录，与库文件添加到项目。

6.3 变量声明

1. 先声明全局变量

```
//全局变量
pcap_t* afx_adhandle;          //当前打开的网络接口
struct pcap_pkthdr* afx_header;      //捕获数据报的头部
const u_char* afx_pkt_data;          //捕获数据报数据
```

2. 然后使用类向导申明成员变量绑定相应的组件



3. 部分变量的含义如下：

```
public:
    CListBox mc_Message;
    CListBox mc_list;
    pcap_if_t* m_alldevs;       //指向设备列表首部的指针
    pcap_if_t* m_selectdevs;    //当前选择的设备列表的指针
    DWORD m_this_ip;          //本机IP地址
    BYTE  m_this_mac[6];      //本机物理地址
    DWORD m_IP;               //查询的IP地址
    BYTE  m_MAC[6];           //查询获得的物理地址
    DWORD m_this_broad;       //本机广播地址
    DWORD m_this_netmask;     //本机子网掩码

    CWinThread* m_Capturer;     /*工作者线程*/
    bool  m_if_get_this_mac;        //标记是否已经获得本机MAC地址
    bool  m_get_state;              //标记是否已经获得请求的MAC地
址
    DWORD ip2long(CString in);
//  CSring long2ip(DWORD in);
    CString long2ip(DWORD in);
    int GetSelfMac();/*获取自己主机的MAC地址*/
    int SendARP(BYTE* SrcMAC, BYTE* SendHa, DWORD SendIp,
DWORD RecvIp);//发送ARP请求函数
```

```
// 将char*类型的MAC地址转换为字符串类型
    CString char2mac(BYTE* MAC);
    CButton mc_get;
    CButton mc_return;
```

6.4 创建数据结构

```
#pragma pack(1)     //进入字节对齐方式
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;    //帧头部结构体
    WORD            HardwareType;   //硬件类型
    WORD            ProtocolType;   //协议类型
    BYTE            HLen;           //硬件地址长度
    BYTE            PLen;           //协议地址长度
    WORD            Operation;      //操作字段
    BYTE            SendHa[6];      //源mac地址
    DWORD           SendIP;         //源ip地址
    BYTE            RecvHa[6];      //目的mac地址
    DWORD           RecvIP;         //目的ip地址
} ARPFrame_t;
#pragma pack()      //恢复缺省对齐方式
```

6.5 构造函数

```
CARPDlg::CARPDlg(CWnd* pParent /*=nullptr*/)
    : CDialogEx(IDD_ARP_DIALOG, pParent)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    m_this_ip = 0;
    //获得本机的设备列表
    char  errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING,     //获取本机的接
口设备
        NULL,                   //无需认证
        &m_alldevs,                 //指向设备列表首部
        errbuf                  //出错信息保存缓存区
    ) == -1) {
        /*错误处理,结果为-1代表出现获取适配器列表失败*/
        MessageBox(L"获取本机设备列表失败: " + CString(errbuf),
MB_OK);
    }

    m_selectdevs = m_alldevs;
```

```
}
```

6.6 初始化函数

在`BOOL CARPDlg::OnInitDialog()`中添加以下代码

```
// TODO: 在此添加额外的初始化代码
    mc_Message.ResetContent();//清除原有框的内容
    mc_Message.AddString(CString(m_selectdevs->name));
//显示该网络接口设备的名字
    mc_Message.AddString(CString(m_selectdevs->description));
//显示该网络接口设备的描述信息

    pcap_addr_t* a;
    CString output1, output2, output3, output4;
    bool ok = 0;
    while (1) {
        for (a = m_selectdevs->addresses; a != NULL; a = a->next)
{
            if (a->addr->sa_family == AF_INET) {  //判断该地址是否
IP地址

                m_this_ip = ntohl(((struct sockaddr_in*)a->addr)-
>sin_addr.s_addr);  //显示IP地址

                output2 = L"IP地址:  " + long2ip(m_this_ip);

                m_this_netmask = ntohl(((struct sockaddr_in*)a-
>netmask)->sin_addr.s_addr);  //显示地址掩码
                if (m_this_netmask != ((((255 * 256) + 255) * 256
+ 255) * 256 + 0))    //获取地址掩码为255.255.255.0的网络

                    break;
                ok = 1;
                output3 = L"地址掩码: " + long2ip(m_this_netmask);

                m_this_broad = ntohl(((struct sockaddr_in*)a-
>broadaddr)->sin_addr.s_addr);  //显示广播地址
                output4 = L"广播地址: " + long2ip(m_this_broad);
            }
        }
        if (ok)
            break;
        m_selectdevs = m_selectdevs->next;
    }

    //获得本机MAC地址
    GetSelfMac();

    if (!m_if_get_this_mac) {
```

```
        AfxMessageBox(L"获取本机MAC地址失败!", MB_OK |
MB_ICONERROR);
        mc_get.EnableWindow(false);
    }
    else {
        for (int i = 0; i < 6; i++)
            m_this_mac[i] = m_MAC[i];
    }

    output1.Format(L"MAC地址:  " + char2mac(m_this_mac));

    mc_Message.AddString(output1);
    mc_Message.AddString(output2);
    mc_Message.AddString(output3);
    mc_Message.AddString(output4);

    m_IP = m_this_ip;
    UpdateData(FALSE);
    return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
}
```

6.7 发送报文函数

```
int CARPDlg::SendARP(BYTE* SrcMAC, BYTE* SendHa, DWORD SendIp,
DWORD RecvIp)
{
    // TODO: 在此处添加实现代码.
    CARPDlg* dlg = (CARPDlg*)theApp.m_pMainWnd; //获取对话框句柄
    char errbuff[1000];
    memset(errbuff, 0, sizeof(errbuff));
    if ((afx_adhandle = pcap_open(dlg->m_selectdevs->name,  // 设
备名称
        65536,    // WinPcap获取网络数据包的最大长度
        PCAP_OPENFLAG_PROMISCUOUS,  // 混杂模式
        1000,     // 读超时为1秒
        NULL,
        errbuff // error buffer
    )) == NULL)
    {
        AfxMessageBox(L"打开该设备网卡接口失败!", MB_OK |
MB_ICONERROR);
        return -1;
    }

    ARPFrame_t  ARPFrame;
    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;                                  //协议地址长度为4
    ARPFrame.Operation = htons(0x0001);                 //操作为ARP请求
    ARPFrame.SendIP = SendIp;//将ARPFrame.SendIP设置为本机网卡上绑定的IP地址。
    ARPFrame.RecvIP = RecvIp;       //将ARPFrame.RecvIP设置为请求的IP地址;


    for (int i = 0; i < 6; i++)
    {
        ARPFrame.FrameHeader.DesMAC[i] = 0xff;  //将ARPFrame.FrameHeader.DesMAC设置为广播地址。
        ARPFrame.FrameHeader.SrcMAC[i] = SrcMAC[i];
        ARPFrame.SendHa[i] = SendHa[i]; //将ARPFrame.SendHa设置为本机网卡的MAC地址。
        ARPFrame.RecvHa[i] = 0x00;      //将ARPFrame.RecvHa设置为0。
    }

    if (pcap_sendpacket(afx_adhandle, (u_char*)&ARPFrame,
        sizeof(ARPFrame_t)) != 0)
    {
        //发送错误处理
        mc_list.AddString(L"获取IP地址:  " + long2ip(RecvIp) + L"的MAC地址失败! ");
    }
    return 0;
}
```

6.8 数据包捕获工作者线程

```
UINT Capturer(LPVOID pParm)
{
    CARPDlg* dlg = (CARPDlg*)theApp.m_pMainWnd; //获取对话框句柄

    char errbuff[1000];
    memset(errbuff, 0, sizeof(errbuff));

    if ((afx_adhandle = pcap_open(dlg->m_selectdevs->name,  // 设备名称
        65536,    // WinPcap获取网络数据包的最大长度
        PCAP_OPENFLAG_PROMISCUOUS,  // 混杂模式
        1000,     // 读超时为1秒
        NULL,
        errbuff // error buffer
    )) == NULL)
    {
        AfxMessageBox(L"打开该设备网卡接口失败!", MB_OK | MB_ICONERROR);
```

```
        return -1;
    }

    //利用pcap_next_ex函数捕获数据包
    /* 此处循环调用 pcap_next_ex来接受数据报*/
    int res;
    while (res = pcap_next_ex(afx_adhandle, &afx_header,
&afx_pkt_data) >= 0) {
        if (res == 0)  //超时情况
            continue;

        ARPFrame_t* arp = (ARPFrame_t*)afx_pkt_data;
        if (!dlg->m_if_get_this_mac && (ntohs(arp-
>FrameHeader.FrameType) == 0x0806) && ((arp->Operation) ==
htons(0x0002)) && (arp->RecvIP == htonl(dlg->m_IP)))
/*0x0002:ARP应答*/
        {
            dlg->m_get_state = true;
            for (int i = 0; i < 6; i++)
                dlg->m_MAC[i] = arp->SendHa[i];
        }
        if (dlg->m_if_get_this_mac && (ntohs(arp-
>FrameHeader.FrameType) == 0x0806) && ((arp->Operation) ==
htons(0x0002)) && (arp->SendIP == htonl(dlg->m_IP)))
        {
            dlg->m_get_state = true;
            for (int i = 0; i < 6; i++)
                dlg->m_MAC[i] = arp->SendHa[i];
        }

    }
    if (res == -1)  //获取数据包错误
    {
        AfxMessageBox(L"获取数据包错误!", MB_OK | MB_ICONERROR);
    }

    return 0;
}
```

6.9 获取自己的MAC地址

```
int CARPDlg::GetSelfMac()
{
    // TODO: 在此处添加实现代码.
/************************************************************
     *本地主机模拟一个远端主机，发送一个ARP请求报文，
     *该请求报文请求本机网络接口上绑定的IP地址与MAC地址的对应关系

************************************************************/

    //创建工作者线程
```

```
    m_Capturer = AfxBeginThread((AFX_THREADPROC)Capturer, NULL,
THREAD_PRIORITY_NORMAL);
    if (m_Capturer == NULL) {
        AfxMessageBox(L"启动捕获数据包线程失败!", MB_OK |
MB_ICONERROR);
        return FALSE;
    }


    //随机设置源MAC地址
    BYTE SrcMAC[6], SendHa[6];
    for (int i = 0; i < 6; i++) {
        SrcMAC[i] = 0x66;
        SendHa[i] = 0x66;
    }

    DWORD SendIp, RecvIp;
    SendIp=0;
    inet_pton(AF_INET,"112.112.112.112",(void*)SendIp); //随便设的
请求方ip
    RecvIp = htonl(m_this_ip);      //将接受方IP设置成本机IP
    m_IP = SendIp;
    ::Sleep(40);
    m_get_state = false;
    SendARP(SrcMAC, SendHa, SendIp, RecvIp);    //发送ARP请求报
    ::Sleep(5000);//等待获取成功
    if (m_get_state)
        m_if_get_this_mac = true;

    return 0;
}
```

6.10 点击获取按钮函数

```
void CARPDlg::OnClickedGet()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(true);

    if (m_IP == m_this_ip) {
        mc_list.AddString(long2ip(m_IP) + L" --> " +
char2mac(m_this_mac));
        mc_list.SetCurSel(mc_list.GetCount() - 1);
        return;
    }

    BYTE SrcMAC[6], SendHa[6];
    DWORD SendIp, RecvIp;

    //将SendHa,SrcMAC设置为本机网卡的MAC地址
```

```
    for (int i = 0; i < 6; i++) {
        SrcMAC[i] = m_this_mac[i];
        SendHa[i] = m_this_mac[i];
    }

    //将RecvIP设置为请求的IP地址;
    RecvIp = htonl(m_IP);

    //将SendIP设置为本机网卡上绑定的IP地址
    SendIp = htonl(m_this_ip);

    m_get_state = false;
    SendARP(SrcMAC, SendHa, SendIp, RecvIp);    //发送ARP请求报
    ::Sleep(2000);//等待获取成功
    if (m_get_state) {
        //获取成功
        mc_list.AddString(long2ip(m_IP) + L" --> " +
char2mac(m_MAC));
    }
    else {
        mc_list.AddString(L"连接到 " + long2ip(m_IP) + L" 超时! ");
    }
    //将光标设定在最后一行
    mc_list.SetCurSel(mc_list.GetCount() - 1);
}
```

6.11 点击返回按钮函数

```
void CARPDlg::OnClickedReturen()
{
    // TODO: 在此添加控件通知处理程序代码
    mc_list.ResetContent();//清除原有框的内容
}
```

6.12 格式转换函数

6.12.1 将char*类型的MAC地址转换成字符串类型

```
CString CARPDlg::char2mac(BYTE* MAC)
{
    // TODO: 在此处添加实现代码.


    CString ans;
    ans.Format(L"%02X-%02X-%02X-%02X-%02X-%02X", int(MAC[0]),
int(MAC[1]), int(MAC[2]), int(MAC[3]), int(MAC[4]), int(MAC[5]));
    return ans;
}
```

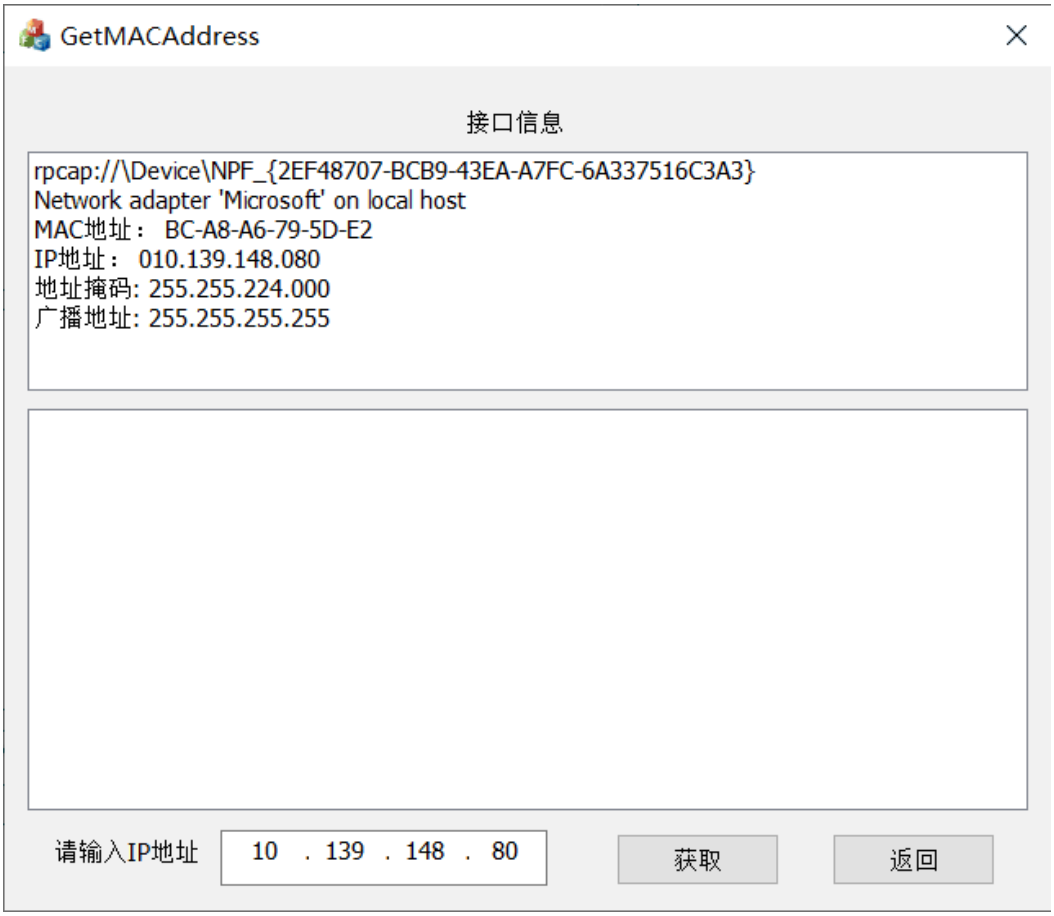
6.12.2 将数字类型的IP地址转化为字符串类型

```
CString CARPDlg::long2ip(DWORD in)
{
    // TODO: 在此处添加实现代码.
    DWORD mask[] = { 0xFF000000,0x00FF0000,0x0000FF00,0x000000FF
};
    DWORD num[4];

    num[0] = in & mask[0];
    num[0] = num[0] >> 24;

    num[1] = in & mask[1];
    num[1] = num[1] >> 16;

    num[2] = in & mask[2];
    num[2] = num[2] >> 8;

    num[3] = in & mask[3];

    CString ans;
    ans.Format(L"%03d.%03d.%03d.%03d", num[0], num[1], num[2],
num[3]);
    return ans;
}
```

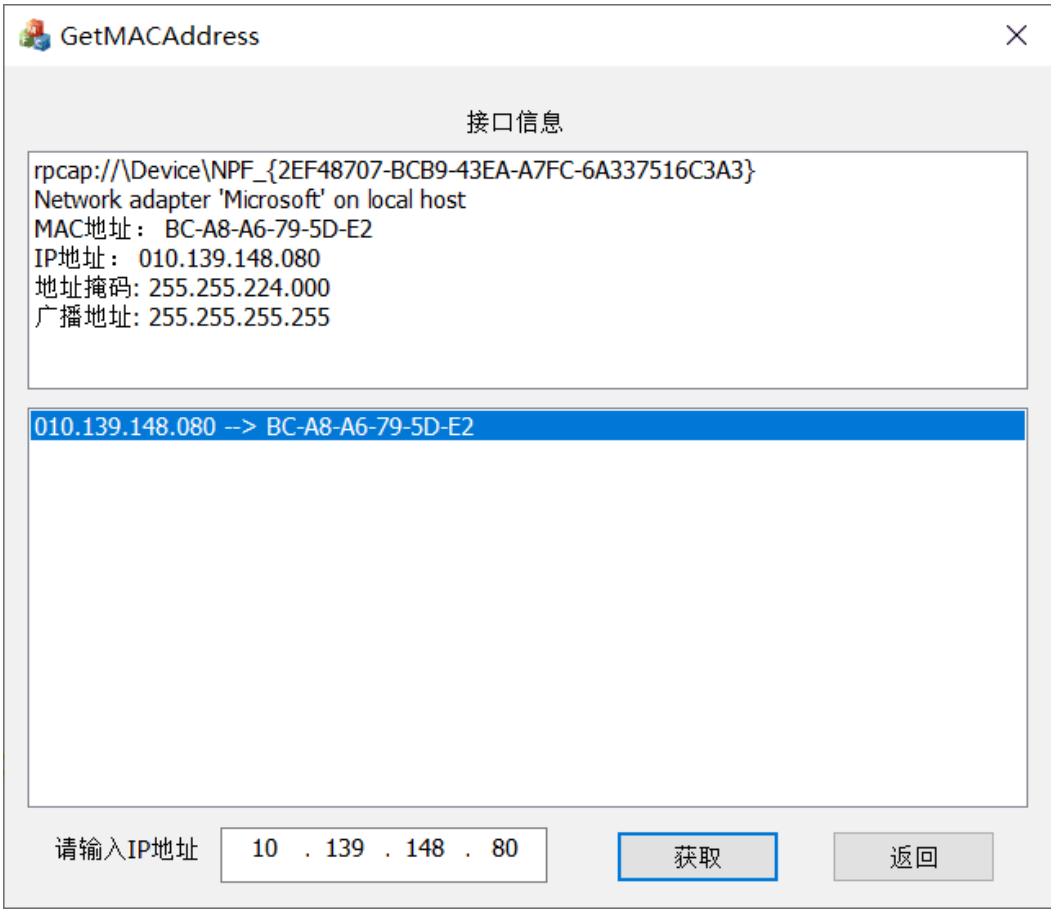
6.12.3 将字符串类型的IP地址转化为数字类型

```
DWORD CARPDlg::ip2long(CString in)
{
    // TODO: 在此处添加实现代码.
    DWORD ans = 0, temp;
    int size = in.GetLength();

    for (int i = 0; i < size; i++)
    {
        if (in[i] == '.') {
            ans = ans * 256 + temp;
            temp = 0;
            continue;
        }
        temp = temp * 10 + in[i] - '0';
    }
    ans = ans * 256 + temp;
    return ans;
}
```

7 效果演示


GetMACAddress
接口信息
rpcap://\Device\NPF_{2EF48707-BCB9-43EA-A7FC-6A337516C3A3}
Network adapter 'Microsoft' on local host
MAC地址： BC-A8-A6-79-5D-E2
IP地址： 010.139.148.080
地址掩码: 255.255.224.000
广播地址: 255.255.255.255
请输入IP地址
10 . 139 . 148 . 80
获取
返回

点击获取按钮后


GetMACAddress
接口信息
rpcap://\Device\NPF_{2EF48707-BCB9-43EA-A7FC-6A337516C3A3}
Network adapter 'Microsoft' on local host
MAC地址： BC-A8-A6-79-5D-E2
IP地址： 010.139.148.080
地址掩码: 255.255.224.000
广播地址: 255.255.255.255
010.139.148.080 --> BC-A8-A6-79-5D-E2
请输入IP地址
10 . 139 . 148 . 80
获取
返回