《网络技术与应用》课程大作业

实验二：利用 ARP 协议获取 IP 地址与 MAC 地址的对应关系

姓名：王科

学号：1310583

专业：计算机科学与技术

完成日期：2015/12/5

一、 实验要求介绍

在以太网中，获取 MAC 地址常常是其他工作的前提。本实验要求利用 WinPcap 提供的功能获取以太网中主机的 MAC 地址。通过本实验不但可以学习 ARP 的工作过程，而且可以深入了解 IP 地址和 MAC 地址的有关概念。

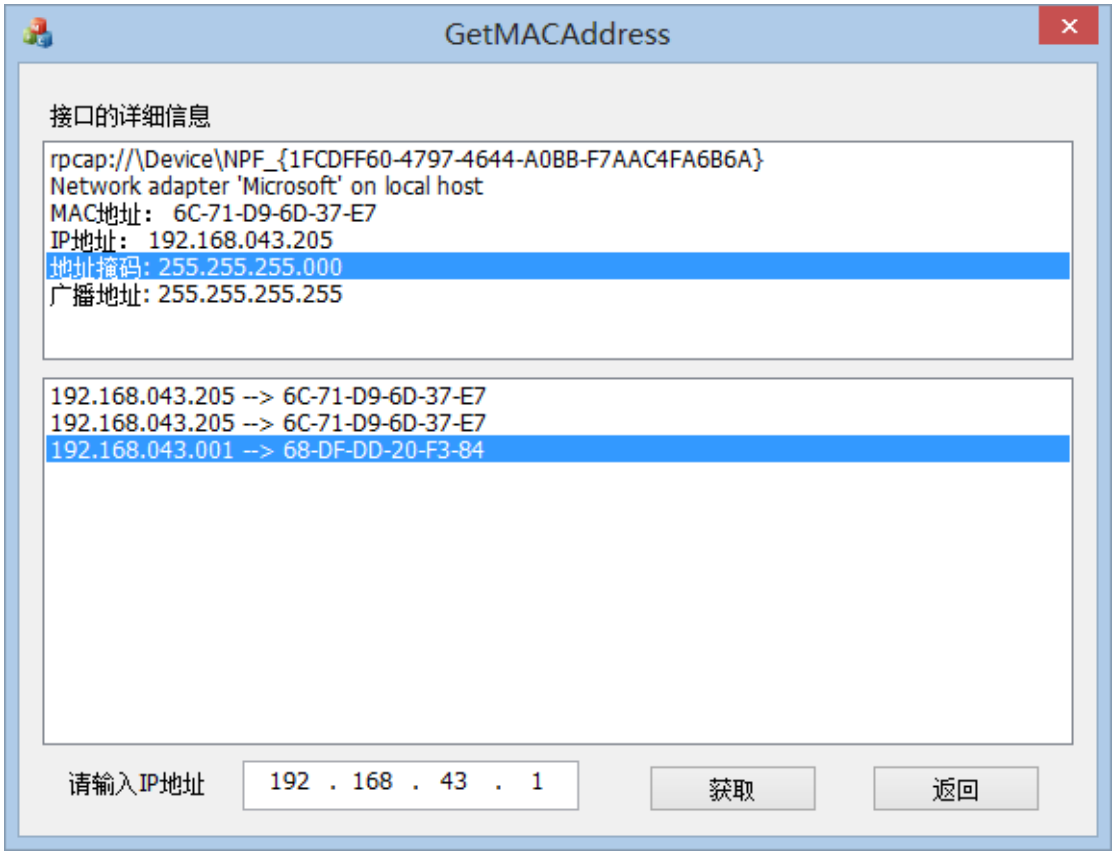
本实验的目的是获取以太网中主机的 MAC 地址，因此以太网在该实验中是必不可少的。本实验使用的以太网既可以是共享式以太网，也可以是交换式以太网。

二、 实验编译运行环境

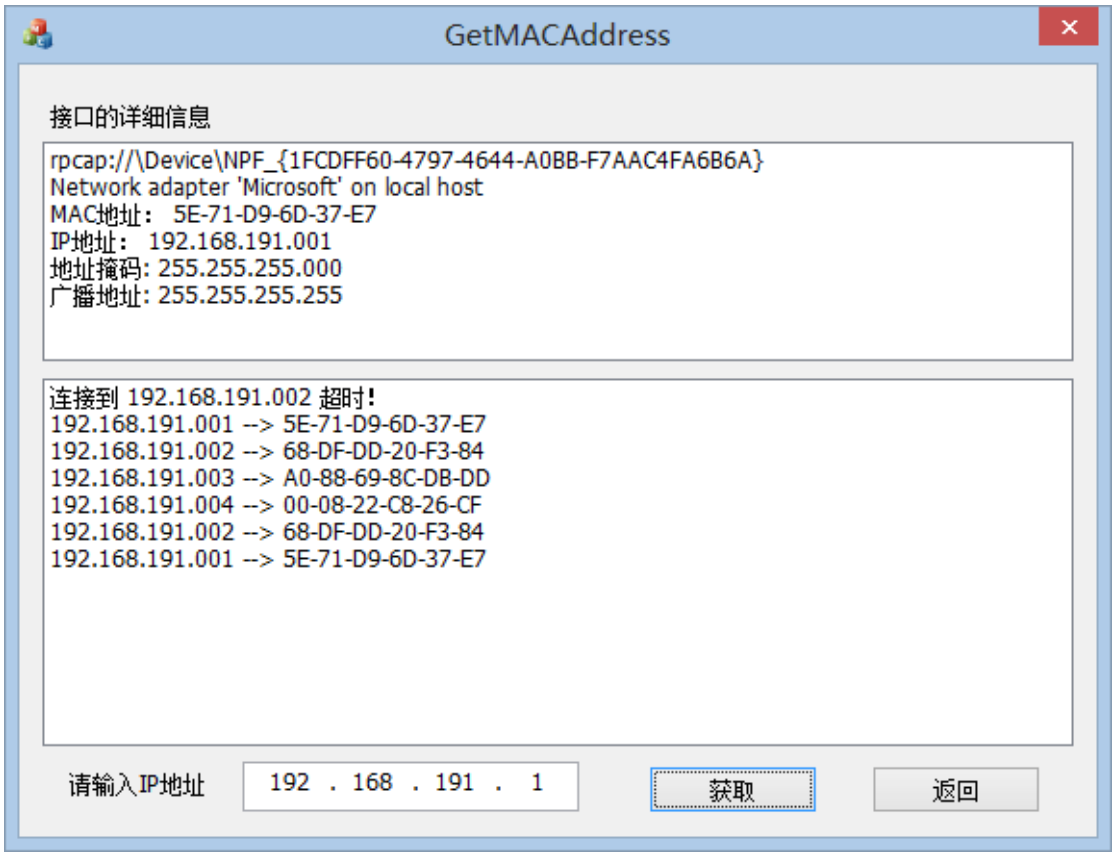
本程序编译环境是：Visual Studio 2012；系统环境是：Windows 8（64 位）；

三、 编写 IP 数据报捕获与分析软件运行效果

程序最终界面如下：



程序在以太网中测试界面如下：



程序首先在上面的窗口显示接口的详细信息，包括网络接口设备的名字、设备的描述信息、MAC 地址、IP 地址、地址掩码和广播地址。

在下面的窗口显示该以太网中输入的 IP 地址与对应的 MAC 地址。其中通过下方的 IP 地址控件输入想要获得对应 MAC 地址的 IP 地址，通过点击“获取”按钮获取对应关系，若获取成功则显示“xxx.xxx.xxx.xxx(IP 地址) → xx-xx-xx-xx-xx-xx(MAC 地址)”，否则则显示“连接到 xxx.xxx.xxx.xxx(IP 地址)超时!”。

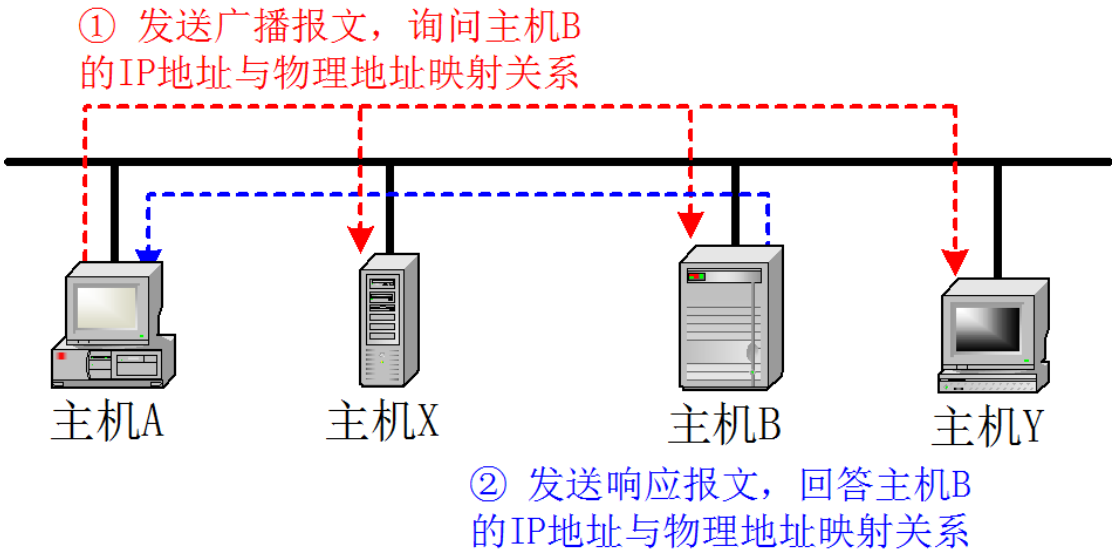
当点击“返回”按钮之后会清空列表框中的内容。

这就是本次编写程序主要实现的功能，下面详细介绍编写的原理思路和步骤。

四、 实验原理

1. ARP 协议的基本思想

假定在一个以太网中，主机 A 欲获得主机 B 的 IP 地址 I_B 与 MAC 地址 P_B 的映射关系。ARP 协议的工作过程如下图所示：



①主机 A 广播发送一个带有 I_B 的请求信息包，请求主机 B 用它的 IP 地址 I_B 和 MAC 地址 P_B 的映射关系进行响应。

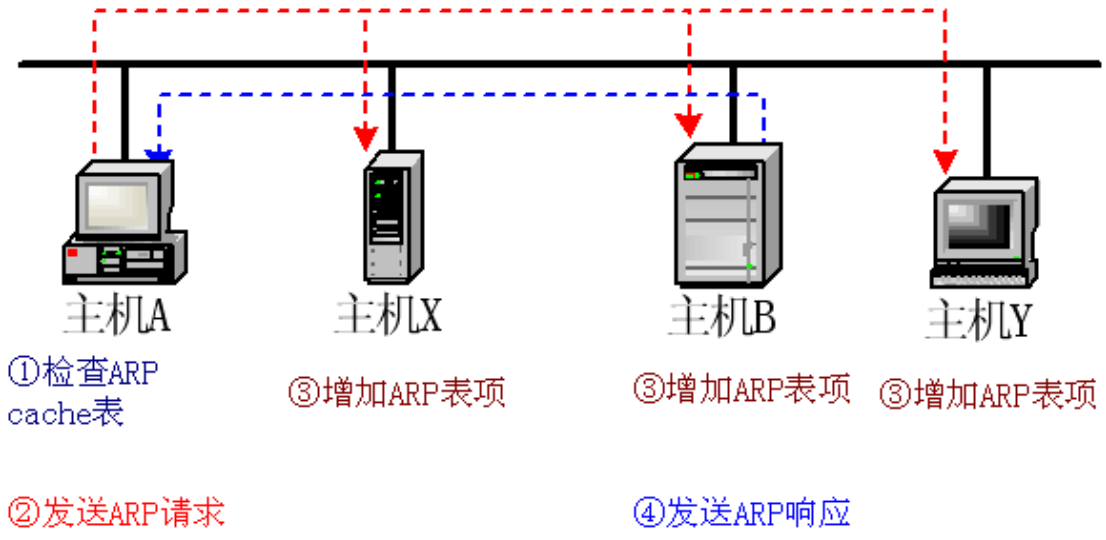
②以太网上的所有主机接收到这个请求信息（包括主机 B 在内）。

③主机 B 识别该请求信息，并向主机 A 发送带有主机 B 的 IP 地址 I_B 和 MAC 地址 P_B 映射关系的响应信息包。

④主机 A 得到 I_B 与 P_B 的映射关系，并可以在随后的发送过程中使用该映射关系。

一个完整的 ARP 工作过程如下图：

假设以太网上有四台计算机，分别是计算机 A、B、X 和 Y（如下图）。现在，计算机 A 的应用程序需要和计算机 B 的应用程序交换数据。在计算机 A 发送信息前，必须首先得到计算机 B 的 IP 地址与 MAC 地址的映射关系。一个完整的 ARP 软件工作过程如下图：



①计算机 A 检查自己高速 cache 中的 ARP 表，判断 ARP 表中是否存在计算机 B 的 IP 地址与 MAC 地址的映射关系。如果找到，则完成 ARP 地址解析；如果没有找到，则转至下一步。

②计算机 A 广播含有自身 IP 地址与 MAC 地址映射关系的请求信息包，请求解析计算机 B 的 IP 地址与 MAC 地址的映射关系。

③包括计算机 B 在内的所有计算机接收到计算机 A 的请求信息，然后将计算机 A 的 IP 地址与 MAC 地址的映射关系存入各自的 ARP 表中。

④计算机 B 发送 ARP 的响应信息给计算机 A，通知自己的 IP 地址与 MAC 地址的映射关系。

⑤计算机 A 收到计算机 B 的响应信息，并将计算机 B 的 IP 地址与 MAC 地址的映射关系存入自己的 ARP 表中，从而完成计算机 B 的 ARP 地址解析。

2. ARP 数据的封装和报文格式

当 ARP 报文在以太网中传送时，需要将它们封装在以太网数据帧中。为了使接收方能够容易的识别该数据帧携带的是 ARP 数据，发送方需要将以太网数据帧首部的长度/类型字段指定是 0806H。由于 ARP 请求和应答分别采用广播方式和单播方式发送，因此封装 ARP 请求的数据帧的目的地址为全 1 形式的广播地址，而封装 ARP 响应的数据帧的目的地址为接收结点的单播地址。

ARP 是一种适应性非常强的协议，它既可以在以太网中使用，也可以在其他类型的物理网络中使用。在以太网中，ARP 数据包的格式如下图：

其中各字段的意义如下：

- 硬件协议： 以太网接口类型为 1
- 协议类型：IP 协议类型为 0800H
- 操作：ARP 请求为 1，ARP 应答为 2
- 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
- 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

3. 利用 WinPcap 获取 IP-MAC 的对应关系

➢ 获取本机网络接口的 IP 地址：

pcap_findalldevs_ex()这个函数获得本地的网卡列表，这个 API 返回一个 pcap_if 结构的链表，链表的每项内容都含有全面网卡信息。

其代码如下：

```
pcap_if_t       *alldevs;  //指向设备链表首部的指针
pcap_if_t       *d;
pcap_addr_t         *a;
char            errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区

if(pcap_findalldevs_ex( …… ) == -1){ …… }//获得本机的设备列表


for(d= alldevs; d != NULL; d= d->next) //显示接口列表

{    ……      //利用 d->name 获取该网络接口设备的名字
     ……      //利用 d->description 获取该网络接口设备的描述信息
     for(a=d->addresses; a!=NULL; a=addr->next) //获取 IP 地址
     {
          if(a->addr->sa_family==AF_INET)  //该地址是否 IP 地址
          {
               ……      //利用 a->addr 获取 IP 地址
               ……      //利用 a->netmask 获取网络掩码
               ……      //利用 a->broadaddr 获取广播地址
               ……      //利用 a->dstaddr)获取目的地址
          }
     }
pcap_freealldevs(alldevs); //释放设备列表
```

➢ 发送 ARP 请求：

根据 ARP 数据报格式和以太网帧结构格式构造 ARP 数据报文数据结构如下：

```
#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_tFrameHeader;  //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
```

发送 ARP 数据报代码如下：

```
ARPFrame_t    ARPFrame;
//将 ARPFrame.FrameHeader.DesMAC 设置为广播地址。
//将 ARPFrame.FrameHeader.SrcMAC 设置为本机网卡的 MAC 地址。
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
//将 ARPFrame.SendHa 设置为本机网卡的 MAC 地址。
//将 ARPFrame.SendIP 设置为本机网卡上绑定的 IP 地址。
//将 ARPFrame.RecvHa 设置为 0。
//将 ARPFrame.RecvIP 设置为请求的 IP 地址;
if (pcap_sendpacket(adhandle, (u_char *) &ARPFrame,
      sizeof(ARPFrame_t)!= 0)
      {…… //发送错误处理       }
else  {     …… //发送成功       }
```

- 捕获本机接口的数据包：

捕获网卡上的数据包需要使用数据包捕获工作者线程,使用 pcap_open（）打开接口，使用 pcap_next_ex（）函数捕获数据包。

```
//数据包捕获工作者线程
UINT Capturer(LPVOID pParm)
{
    pcap_open()    //打开该设备网卡接口
    //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*/
    while(res = pcap_next_ex(afx_adhandle,&afx_header,&afx_pkt_data)>=0){
        ……
    }
    return 0;
}
```

- 获取本机网卡的 MAC：

1. 获取本机网络接口和接口上绑定的 IP 地址。

2. 发送 ARP 请求，请求本机网络接口上绑定的 IP 地址与 MAC 地址的对应关系：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。

3. 捕获本机的 ARP 响应，获取本机网络接口的 MAC 地址。

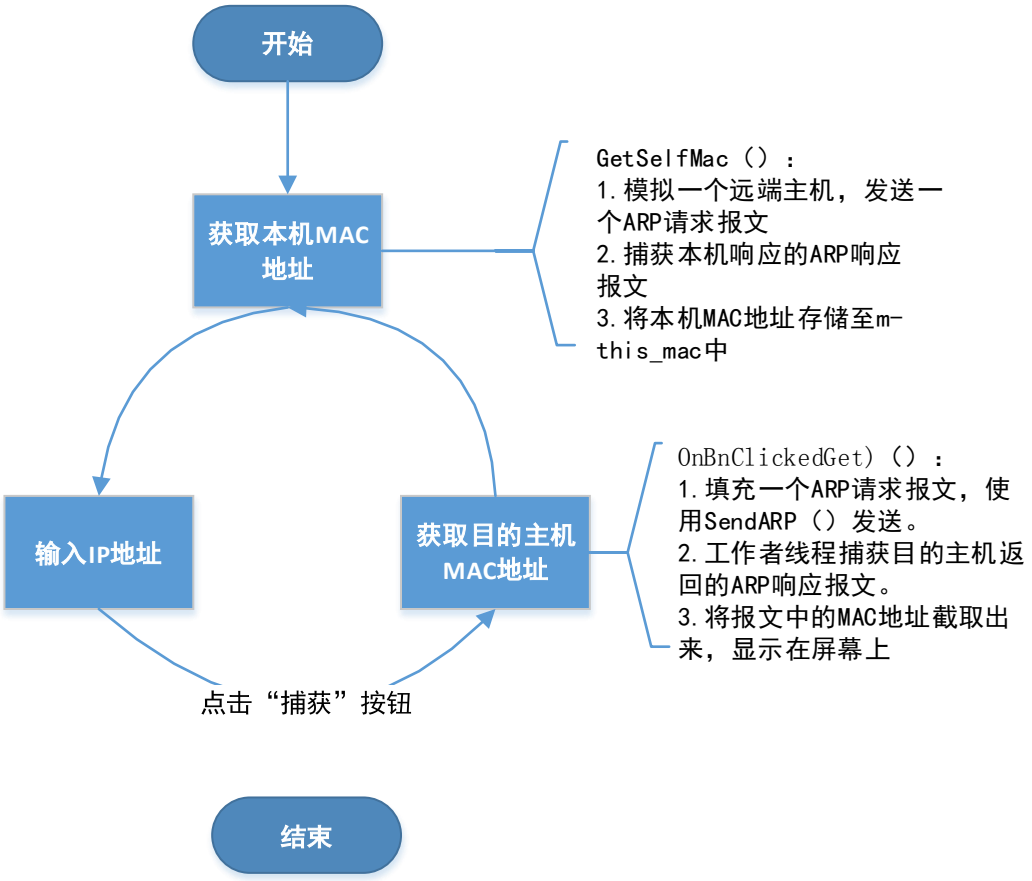
五、 程序实现步骤及代码

本次编写“利用 ARP 协议获取 IP 地址与 MAC 地址的对应关系”程序的过程中，主要有两方面内容：

第一是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 地址的对应关系然后捕获本机的 ARP 响应报文获取本机的 MAC 地址。

第二是将上面获取的本机的 MAC 地址填入想要获取目标 IP 地址主机的 ARP 请求报文中，发送出去，然后捕获目标主机返回的 ARP 响应报文获取目标主机的 MAC 地址。

其逻辑图如下：



1. 程序中所用变量如下图：

```
CListBox mc_Message;     //捕获接口的详细信息
CListBox m_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
afx_msg void OnClose();
int SendARP(BYTE* SrcMAC,BYTE* SendHa,DWORD SendIp,DWORD RecvIp);          // 发送 ARP 请求函数
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

DWORD    ip2long (CString in);      /*  将字符串类型的 IP 地址转换成数字类型的  */
CString    long2ip(DWORD in); /*  将数字类型的 IP 地址转换成字符串类型的  */
CWinThread* m_Capturer;           /*工作者线程*/
int GetSelfMac(void);            /*获取自己主机的 MAC 地址*/

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

//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. 程序构造 ARP 报文数据结构：

```
#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_tFrameHeader;  //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
```

3. 发送报文函数 SendARP（）

```
int CGetMACAddressDlg::SendARP(BYTE* SrcMAC,BYTE* SendHa,DWORD SendIp,DWORD
RecvIp)
{


      CGetMACAddressDlg* dlg = (CGetMACAddressDlg*)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
        m_list.AddString(L"获取 IP 地址：  "+long2ip(RecvIp)+L"  的 MAC 地址失败！");
    }
    return 0;
}
```

4. 数据包捕获工作者线程 UINT Capturer(LPVOID pParm)

```
//数据包捕获工作者线程
UINT Capturer(LPVOID pParm){
    CGetMACAddressDlg* dlg = (CGetMACAddressDlg*)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];
            //AfxMessageBox(L"获得本机 MAC 地址成功!",MB_OK);
        }
    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];
            //AfxMessageBox(L"获得目的主机 MAC 地址成功!",MB_OK);
        }
    }
    if(res==-1)  //获取数据包错误{  AfxMessageBox(L"获取数据包错误!",MB_OK|MB_ICONERROR);  }
```

5. 获取自己主机的 MAC 地址 GetSelfMac()

```
//获取自己主机的 MAC 地址
int CGetMACAddressDlg::GetSelfMac(void)
{
	/*********************************************************
	 *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_PRIORIT
Y_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=inet_addr("112.112.112.112"); //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(3000);//等待获取成功
	if(m_get_state)
		m_if_get_this_mac=true;

	return 0;
}
```

6. 点击“获取”按钮：OnBnClickedGet()

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

	if(m_IP==m_this_ip){
		m_list.AddString(long2ip(m_IP)+L" --> "+char2mac(m_this_mac));
		m_list.SetCurSel(m_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
		m_list.AddString(long2ip(m_IP)+L" --> "+char2mac(m_MAC));
	}
	else{
		m_list.AddString(L"连接到 "+long2ip(m_IP)+ L" 超时！ ");
	}
	//将光标设定在最后一行
	m_list.SetCurSel(m_list.GetCount()-1);
}
```

六、 实验总结

通过这次“利用 ARP 协议获取 IP 地址与 MAC 地址的对应关系”实验使我对于 MAC 地址、IP 地址和网络传输协议有了更深的理解，不但学习了 ARP 的工作过程，而且更深入了解 IP 地址和 MAC 地址的有关概念。

我通过查阅各种资料终于完成了这次的实验，感觉自己在课程中所学的 ARP 数据报文格式、ARP 响应和请求、MAC 地址、IP 地址等理论知识在实践编程中得到了确确实实的应用，当看到自己编写的程序能够成功捕获 ARP 响应数据包时突然感觉到一种有所收获的喜悦，对于网络传输的协议等有了更深的兴趣，不仅仅加深了对理论知识的理解，更提高了我对于探求网络技术的兴致。

最后感谢老师不厌其烦的为我们解答疑惑，也感谢我的室友们耽搁自己的事情一遍又一遍的帮助我搭建局域网调试程序，谢谢！