

《计算机网络》课程大作业

实验一：简单的客户/服务器程序 实验报告

姓名：王科

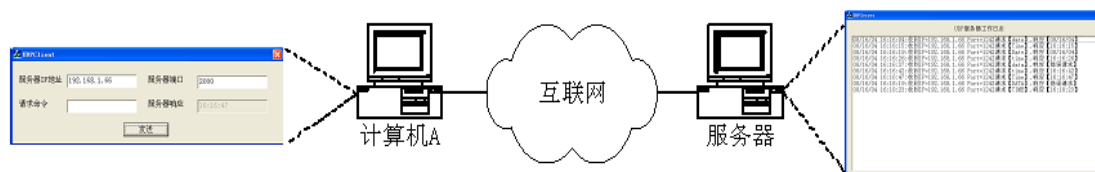
学号：1310583

专业：计算机科学与技术

完成日期：2015/10/19

一、 实验要求介绍

本实验要求利用 CAsyncSocket 类编写一个简单的客户/服务器程序，客户/服务器之间使用数据报方式传递信息，服务器在收到客户发来的 Time 或 Date 请求后，利用本地的时间日期分别进行响应，如下图所示。通过该编程实验，可以加深对客户/服务器交互模型的理解，学习简单的 socket 编程方法。



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、 本程序的时间服务器应用层协议介绍

本客户端/服务器应用层协议分成两部分：

- 客户端：

客户端应用层协议设计格式

不区分大小写的字符串（date、time、其他）

客户端可以发出不分大小写的字符串命令：

当是“date”或“time”时，会收到来自服务器正确的 date 或 time 格式回复命令，当是其他字符串时，收到“错误请求”回复命令。

- 服务器：

服务器应用层协议设计格式

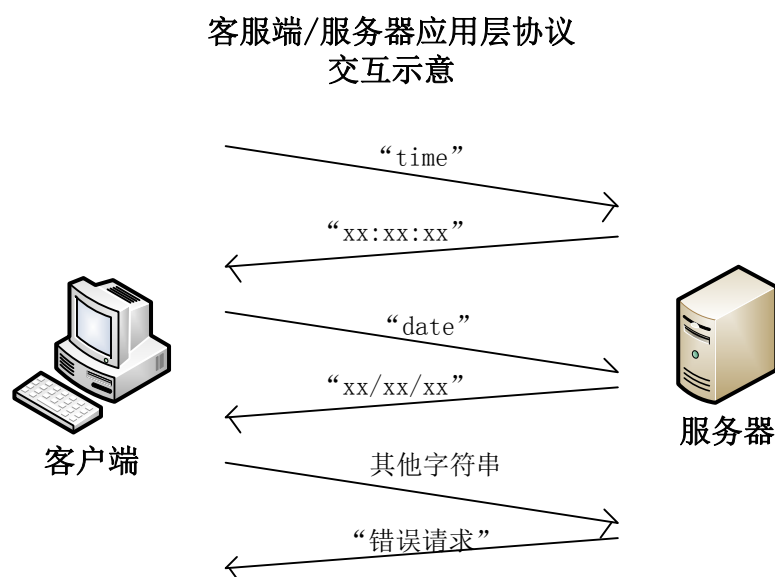
字符串（time: “xx: xx: xx”、
date: “xx/xx/xx”、other: “错误请求”）

当收到请求是不区分大小写的 date 字符串时，返回本机日期：格式是“xx/xx/xx”。

当收到请求是不区分大小写的 time 字符串时，返回本机时间：格式是“xx: xx: xx”。

当收到请求是其他字符串时，返回“错误请求”字符串。

- 客户端、服务器交换协议示意图：



四、 实验实现功能介绍

1. 服务器端效果显示:

服务器“UDPServer.exe”打开后显示界面如下:



- 【1】. 服务器 IP: 在此控件上会显示本程序当前运行环境的 IP 地址列表, 并取获得的第一个 IP 地址创建服务器 Socket 类。
- 【2】. 端口号: 会以此控件上显示的端口号创建服务器 Socket 类, 默认不输入之前端口号是 2000。
- 【3】. 日期: 在此控件上会显示最近一次 UDP 服务器动作时的本地的日期值。

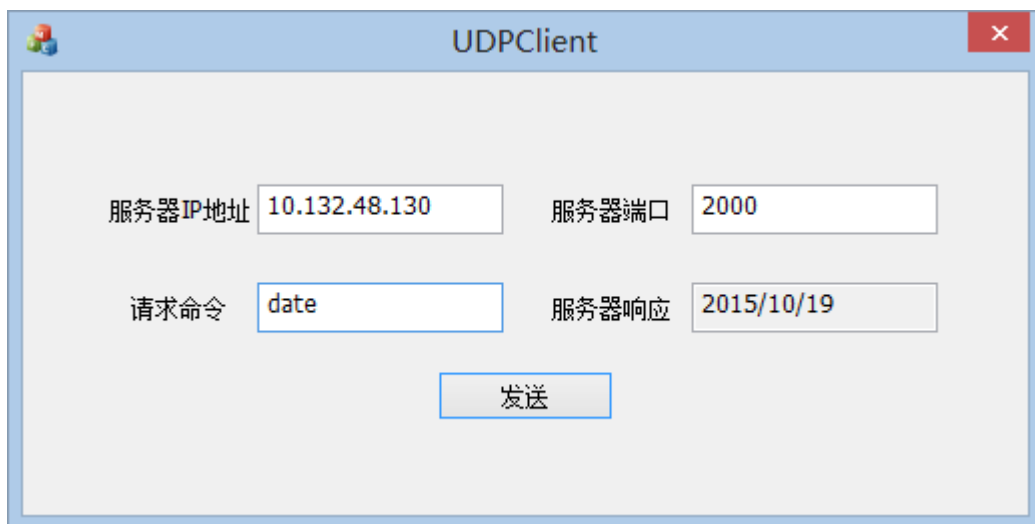
【4】. 时间：在此控件上会显示最近一次 UDP 服务器动作时的本地的时间值。

【5】. UDP 服务器工作日志：此部分会按照动作执行的顺序显示服务器的工作日志，在开始阶段会显示“初始化成功或失败”，接着会显示本机的主机名，然后逐条显示本机的所有可用的 IP 地址，最后会显示以 IP 地址列表里的第一个 IP 地址和端口号显示的数值创建服务器 Socket 类。然后就会以下的格式显示客户端连接记录。

*日期 时间：*收到IP = xx.xxx.xx.xxx Port = xxxx请求 **【xxx】**，响应 **【XXX】**

2. 客户端效果显示：

客户端“UDPClient.exe”打开后显示界面如下：



The screenshot shows a Windows application window titled "UDPClient". The window contains four text input fields arranged in a 2x2 grid. The first field is labeled "服务器IP地址" (Server IP Address) and contains the text "10.132.48.130". The second field is labeled "服务器端口" (Server Port) and contains the text "2000". The third field is labeled "请求命令" (Request Command) and contains the text "date". The fourth field is labeled "服务器响应" (Server Response) and contains the text "2015/10/19". Below these fields is a single button labeled "发送" (Send).

【1】. 服务器 IP 地址：此控件上由用户输入 Ip 地址值作为即将发送的服务器 IP 地址。

【2】. 服务器端口：此控件上由用户输入端口值作为即将发送的服务器端口值。

【3】. 请求命令：在此控件上由用户输入请求命令作为数据报发往服务器。按照此程序设计协议，请求命令可以是 date 和 time（不区分大小写）会返回服务器本地的日期和时间，其余命令都会返回“错误请求”。

【4】. 服务器响应：在此控件上会显示服务器响应发送请求命令的

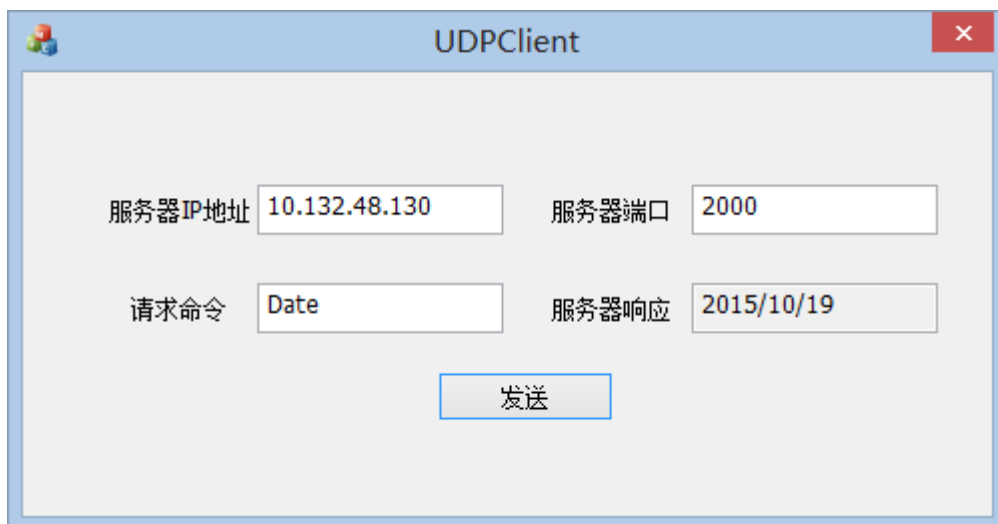
值。按照此程序设计协议，响应会是服务器本地的日期、时间和“错误请求”三种情况。

【5】. **发送：** 用户点击此按钮之后客户端会将请求命令按照服务器 IP 地址值和端口发送到服务器，并在**服务器响应**控件上显示服务器响应内容。

3. 客户/服务器交互效果显示：

【1】. 客户端发送正确的 date 指令，服务器显示客户端连接日志，然后响应客户端并在客户端上显示响应日期内容：

➤ 客户端：



UDPClient

服务器IP地址 10.132.48.130 服务器端口 2000

请求命令 Date 服务器响应 2015/10/19

发送

➤ 服务器端：



UDPServer

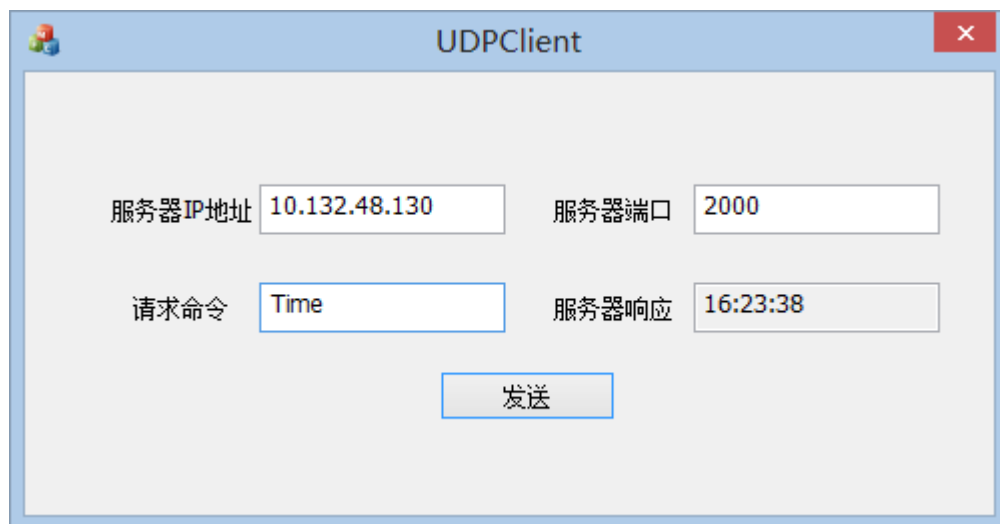
UDP服务器工作日志

服务器IP 10.132.48.130 端口号 2000 日期 2015/10/19 时间 16:19:20

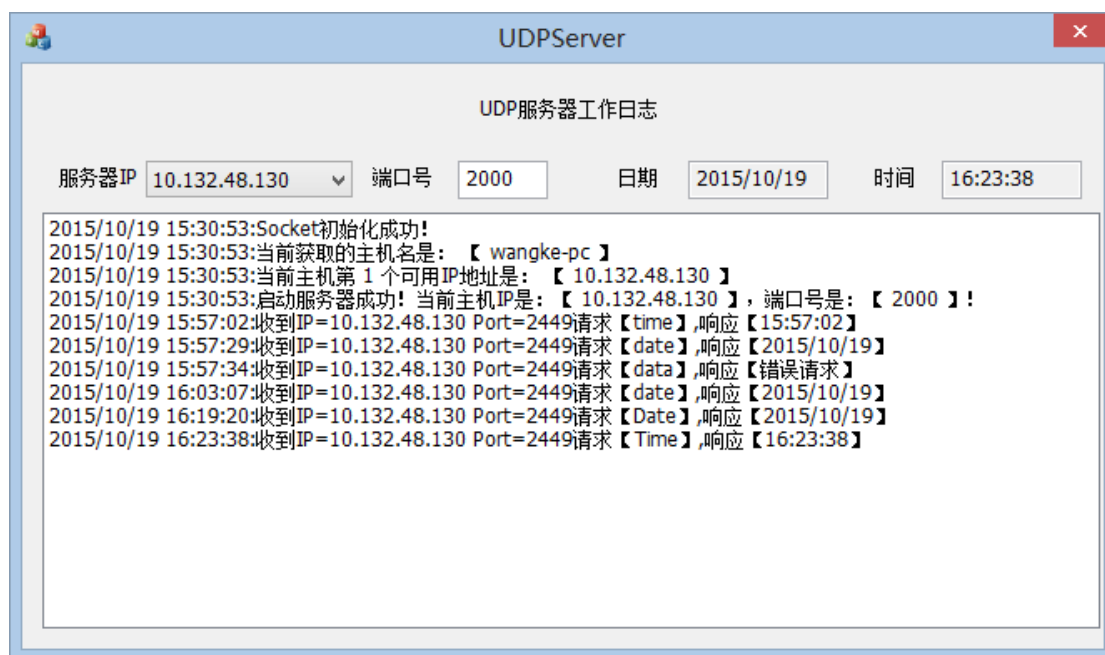
2015/10/19 15:30:53:Socket初始化成功!
2015/10/19 15:30:53:当前获取的主机名是: 【 wangke-pc 】
2015/10/19 15:30:53:当前主机第 1 个可用IP地址是: 【 10.132.48.130 】
2015/10/19 15:30:53:启动服务器成功! 当前主机IP是: 【 10.132.48.130 】, 端口号是: 【 2000 】!
2015/10/19 15:57:02:收到IP=10.132.48.130 Port=2449请求 【time】,响应 【15:57:02】
2015/10/19 15:57:29:收到IP=10.132.48.130 Port=2449请求 【date】,响应 【2015/10/19】
2015/10/19 15:57:34:收到IP=10.132.48.130 Port=2449请求 【data】,响应 【错误请求】
2015/10/19 16:03:07:收到IP=10.132.48.130 Port=2449请求 【date】,响应 【2015/10/19】
2015/10/19 16:19:20:收到IP=10.132.48.130 Port=2449请求 【Date】,响应 【2015/10/19】

【2】. 客户端发送正确的 time 指令，服务器显示客户端连接日志，然后响应客户端并在客户端上显示响应时间内容：

➤ 客户端：

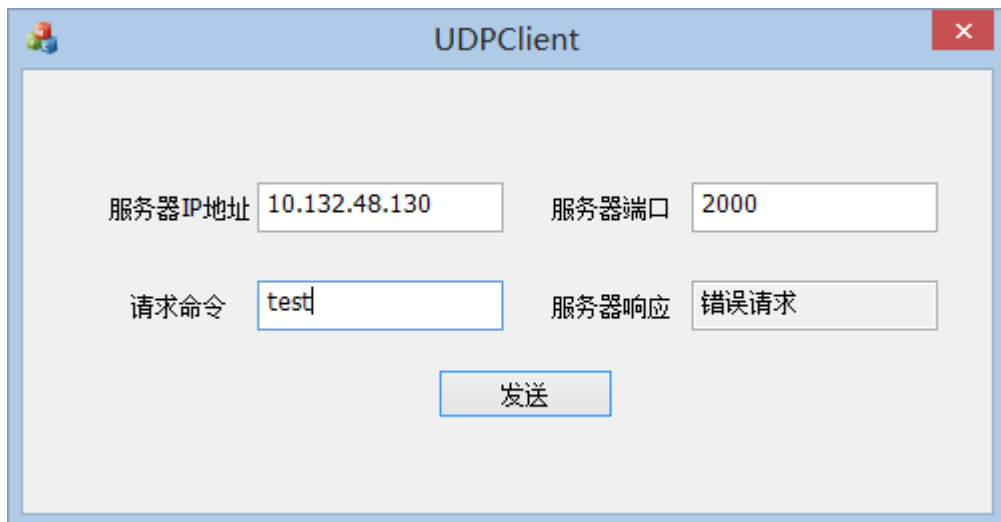


➤ 服务器端：

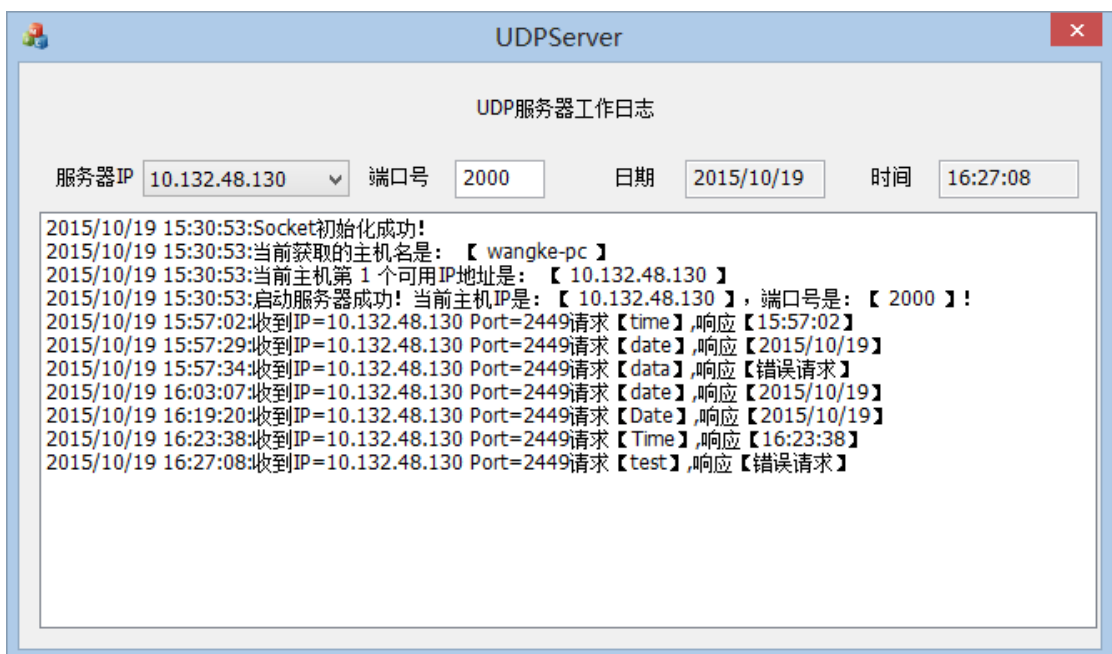


【3】. 客户端发送错误的指令，服务器显示客户端连接日志，然后响应客户端并在客户端上显示响应“请求错误”：

➤ 客户端：



➤ 服务器端：

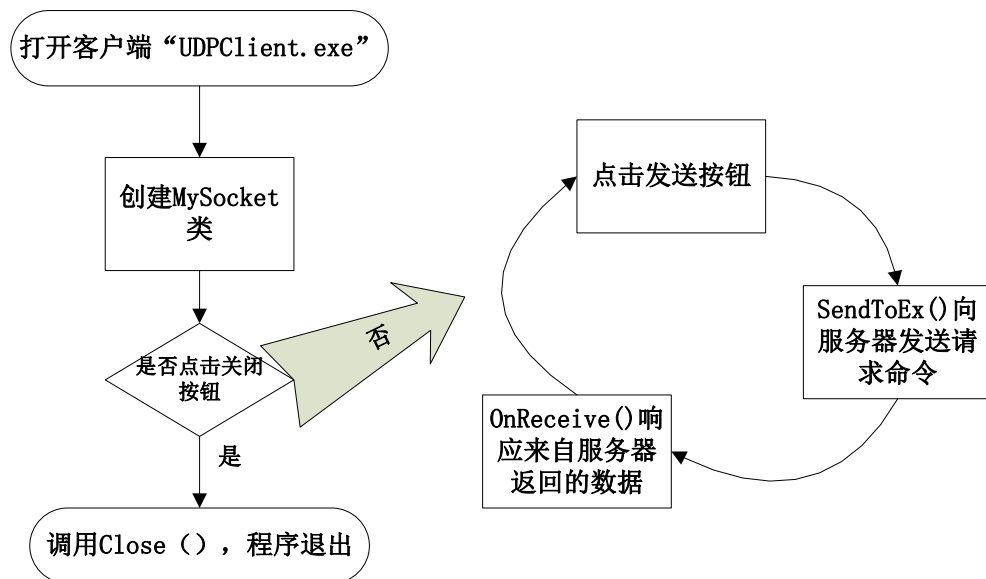


五、 程序实现步骤介绍

1. 客户端：

在客户端主要的流程是当客户端程序初始化创建的时候会创建一个继承于 CAsyncSocket 类的自定义类 MySocket，当用户点击“发送”按钮的时候，会触发一个响应函数，在此函数内会使用 SendToEx（）函数以 UDP 数据报方式向当前控件上显示的服务器 IP 和端口号的服务器发送请求命令。然后再重载 MySocket 类里面的 OnReceive（）函数，在其中执行当收到来自服务器的响应数据报时的操作，将其响应内容显示在控件上面。

其消息结构图如下：



1) 程序初始化操作：

1. 新建类 MySocket，继承自 CAsyncSocket 类。
2. 对话框类的构造函数内对变量初始化如下：

```
CUdpServerDlg::CUdpServerDlg(CWnd* pParent /*=NULL*/)
: CDialogEx(CUdpServerDlg::IDD, pParent)
, m_time(_T(""))
, m_date(_T(""))
, m_port(2000)
, m_ip(_T(""))
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
```

3. 在对话框类初始化 OnInitDialog()内添加如下代码：

```
// TODO: 在此添加额外的初始化代码

if (!AfxSocketInit()) { //初始化
    MessageBox(L"初始化失败", L"提示", MB_OK|MB_ICONSTOP);
}

//随机产生一个客户端口号
srand((unsigned)time(NULL));
int max=RAND_MAX;
int client_port=(int)(rand()*(5000)/max+2002);

if (!m_mysocket.Create(client_port, SOCK_DGRAM, FD_READ)) { //以数据报方式创建socket
    MessageBox(L"Socket套接字创建失败", L"错误", MB_OK|MB_ICONSTOP);
};
```

2) 点击发送按钮操作：创建点击“发送”按钮操作函数 OnClickedIIdsend()


```

void CUDPClientDlg::OnClickedIdsend()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(true); //更新控件对应的变量值
    if(m_ServerIp.IsEmpty())
    {MessageBox(_T("无服务器Ip地址!"),_T("错误!"),MB_OK|MB_ICONEXCLAMATION); return ;}
    if(m_ServerPort <=0 || m_ServerPort> 65535)
    {MessageBox(_T("无请求命令!"),_T("错误!"),MB_OK|MB_ICONEXCLAMATION); return ;}
    if(m_ServerPort <=0 || m_ServerPort> 65535)
    {MessageBox(_T("端口值不正确!"),_T("错误!"),MB_OK|MB_ICONEXCLAMATION);return;}

    //向服务器发送数据报 UDP方式
    int flag=m_mysocket.SendToEx(m_command.GetBuffer(),(m_command.GetLength()+1)
    * sizeof(WCHAR),m_ServerPort,m_ServerIp);
    if (flag == SOCKET_ERROR) {
        MessageBox(_T("向服务器发送请求失败!"),_T("错误!"),MB_OK|MB_ICONSTOP); return;
    }
}

```

3) 响应服务器返回数据时的操作:

重载 MySocket 类里的 OnReceive() 函数, 添加接收到服务器返回数据的操作代码。

```

void MySocket::OnReceive(int nErrorCode)
{
    // TODO: 在此添加专用代码和/或调用基类

    TCHAR buff[4096]; //用于缓冲区接收主机返回的数据报
    ReceiveFrom(buff, sizeof(buff), GetIp, GetPort); //从主机接收数据
    CString data(buff);
    GetBuff=data;

    CUDPClientDlg* dig = (CUDPClientDlg*)theApp.m_pMainWnd;
    dig->m_ServerReceive=GetBuff;
    dig->UpdateData(false);

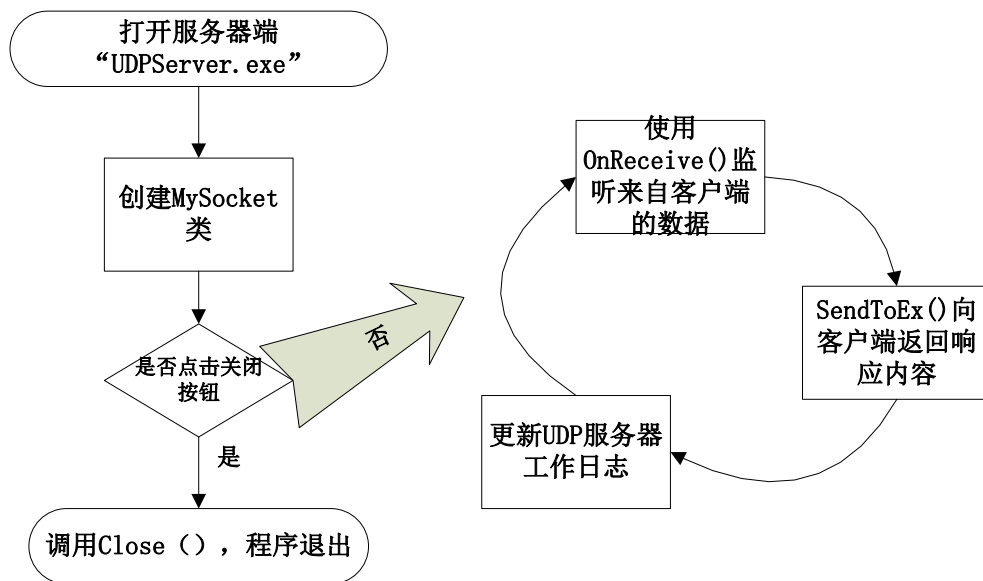
    CAsyncSocket::OnReceive(nErrorCode);
}

```

2. 服务器端:

在服务器端主要的流程是当服务器端程序初始化创建的时候同样会创建一个继承于 CAsyncSocket 类的自定义类 MySocket, 此时会查找本机可用的 IP 地址列表, 并选择第一个 IP 和端口号创建一个 MySocket 类, 然后重载 MySocket 类里面的 OnReceive() 函数执行监听操作, 当有客户端发送数据报时, 使用 SendToEx() 函数以 UDP 数据报方式向客户端返回响应内容。

其消息结构图如下:



1) 程序初始化操作:

1. 新建类 MySocket, 继承自 CAsyncSocket 类
2. 对话框类的构造函数内对变量初始化如下:

```

CUDPServerDlg::CUDPServerDlg(CWnd* pParent /*=NULL*/)
: CDialogEx(CUDPServerDlg::IDD, pParent)
, m_time(_T(""))
, m_date(_T(""))
, m_port(2000)
, m_ip(_T(""))
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
  
```

3. 在对话框类初始化 OnInitDialog()内添加如下代码:

获取主机名:

```

//获取主机名和可用IP地址|
char hostname[20] = "";
int errorcode;
if ((errorcode = gethostname(hostname, sizeof(hostname))) != 0) {
    char char_error[100];
    _itoa_s(errorcode, char_error, 10);
    MessageBoxA(this->GetSafeHwnd(), char_error, "errorcode", MB_OK);
    AfxGetMainWnd()->SendMessage(WM_CLOSE);
};

CString hostname_cstring(hostname);
CString record;
record.Format(L"当前获取的主机名是: 【 %s 】", hostname_cstring);
update_date_and_time(m_date, m_time); //更新日期和时间
m_con_List.AddString(m_date+_T(" ") + m_time+_T(":")+record);
  
```

获取可用 IP 列表, 并显示在 ListBox 控件上:

```

hostent *hn;
hn = gethostbyname(hostname);
int i = 0;
while (hn->h_addr_list[i] != 0) {
    char *p = inet_ntoa(*(in_addr *)hn->h_addr_list[i++]);
    wchar_t pw[20];
    SHAnsiToUnicode(p, pw, 20);
    CString str;
    str.Format(L"%s", pw);
    m_con_ip.AddString(str);
    record.Format(L"当前主机第 %d 个可用IP地址是： 【 %s 】", i, pw);
    update_date_and_time(m_date, m_time); //更新日期和时间
    m_con_List.AddString(m_date+_T(" ") + m_time+_T(":")+record);
}
if (i == 0) {
    AfxGetMainWnd()->SendMessage(WM_CLOSE);
}
UpdateData(false);
m_con_ip.SetCurSel(0);
UpdateData(true);

```

获取本地时间和日期函数：

```

//更新当前时刻的日期和时刻值
void CUDPServerDlg::update_date_and_time(CString& date, CString& time)
{
    date=CTime::GetCurrentTime().Format(L"%Y/%m/%d");
    time=CTime::GetCurrentTime().Format("%H:%M:%S");
    UpdateData(false);
}

```

创建 MySocket 类：

```

//启动服务器操作

if(m_ip.IsEmpty())
{MessageBox(_T("无服务器Ip地址！"),_T("错误！"),MB_OK|MB_ICONEXCLAMATION); return TRUE;}
if(m_port <=0 || m_port> 65535)
{MessageBox(_T("端口值设置错误！"),_T("错误！"),MB_OK|MB_ICONEXCLAMATION); return TRUE;}

UpdateData(true);
if (!m_mysocket.Create(m_port, SOCK_DGRAM, FD_READ,m_ip)) { //以数据报方式创建socket
    MessageBox(L"Socket套接字创建失败", L"错误", MB_OK|MB_ICONSTOP);return TRUE;
};
record.Format(L"启动服务器成功！当前主机IP是： 【 %s 】，端口号是： 【 %u 】！", m_ip,m_port);
update_date_and_time(m_date,m_time); //更新日期和时间
m_con_List.AddString(m_date+_T(" ") + m_time+_T(":")+record);

```

2) 监听客户端数据的函数 OnReceive ()

```

void MySocket::OnReceive(int nErrorCode)
{
    // TODO: 在此添加专用代码和/或调用基类
    CUDPServerDlg* dig = (CUDPServerDlg*)theApp.m_pMainWnd; //获取对话框句柄
    TCHAR buff[4096];
    CString GetIP;
    UINT GetPort;
    ReceiveFrom(buff, 4096, GetIP, GetPort);
    CString data(buff);
    CString temp=data;

    dig->update_date_and_time(dig->m_date, dig->m_time);

    CString Answer=_T("错误请求");
    data.MakeLower();

    if(data=="time")
        Answer=dig->m_time;
    if(data=="date")
        Answer=dig->m_date;

    //响应客户端程序
    SendToEx(Answer.GetBuffer(), (Answer.GetLength() + 1) * sizeof(WCHAR), GetPort, GetIP);
    CString record;
    record.Format(L":收到IP=%s Port=%u请求【%s】,响应【%s】", GetIP, GetPort, temp, Answer);
    dig->m_con_List.AddString(dig->m_date+_T(" ") + dig->m_time + record);

    CAsyncSocket::OnReceive(nErrorCode);
}

```

六、 实验总结

通过本次实验学习了如何使用 CAsyncSocket 类编写一个简单的客户/服务器程序，加深了对客户/服务器模型相互交互的理解，明白了 IP 地址和端口的作用，同时对 UDP 数据报传送方式有了更加深刻的认识。

在此次实验中也我遇到过一些问题，如一开始没有使用 Close（）函数，导致端口号被重复使用使得 Socket 创建失败；又如在一开始客户端的 Socket 创建过程中，使用了指定的端口号，后来发现在这种情况下同一台机器上只能打开一个客户端程序与服务器端相连，后来使用了随机生成客户端端口号的办法解决了这个问题，使得服务器端可以同时与多个客户端程序相互交互！

总之，通过这次实验加深了我对计算机网络协议，传输等方面的理解，感觉知识水平和能力都有所提升。