# 基于UDP的客户端-服务器程序

计算机科学与技术专业 1711436 皮春莹

## 实验目的

利用CAsyncSocket类编写简单的客户-服务器程序，使用数据报方式传递信息。服务器收到客户发来的“Time”或“Date”请求后，利用本地时间和日期分别进行响应。

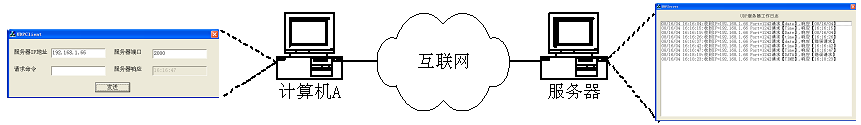


图 1 服务器-客户端示意图示意图

## 运行环境

系统环境是：Windows 10（64 位）。

本程序编译环境是：Visual Studio 2017，使用了 MFC 图形化界面框架。

## 编程思路

网络编程接口Socket把网络连接看成一个流，可以向流写字节，也可以从流读取字节。Socket通过绑定机制与驱动程序建立关系。数据由Socket交给驱动程序向网络发送。驱动程序收到与Socket绑定的数据后，由驱动程序交给Socket，应用进程从Socket中读取数据。Socket常用的操作有：连接远程机器、绑定端口、接收远程连接请求、监听到达的数据、发送数据、接收数据、关闭连接。Socket屏蔽了网络的底层细节（媒体类型、信息包的大小、网络地址、信息重发等），本次编程中使用Microsoft提供的CAysncSocket类。

数据报套接字（datagram sockets）使用UDP协议。UDP提供的是面向非连接、不可靠的服务，数据直接封装在IP数据包中投递，没有确认和排序的过程。UDP端口使用16位二进制数表示，利用端口对给定主机上的多个目标进行区分。

将本次编程分为客户端和服务器两部分，需要完成以下任务：

* 客户端

客户端需要编写一个继承CAysncSocket的类，重载OnReceive函数。在初始化对话框函数OnInitDialog()中，随机产生一个端口号，并创建Socket对象；当点击“发送”按钮时，触发按钮绑定的函数OnSend()，通过Socket对象向指定的IP地址和端口号发送数据报，请求日期（date）或时间（time），命令不区分大小写；Socket对象接收到的服务器数据后，触发OnReceive函数，将数据显示在应用进程的对话框中。流程图如下：

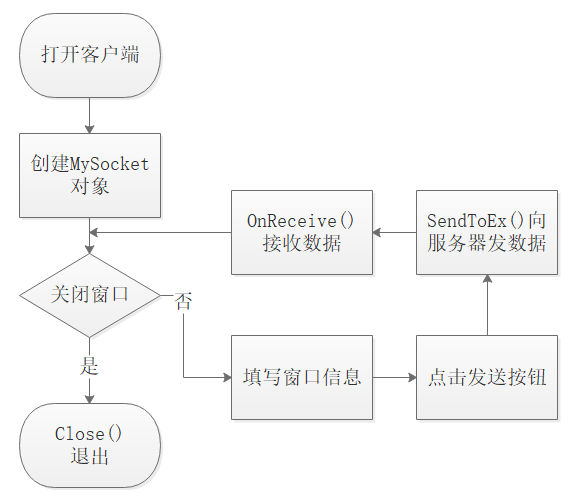


图 2 客户端流程图

* 服务器

服务器也需要编写一个继承CAysncSocket的类，并重载OnReceive函数。在初始化对话框函数OnInitDialog()中，查找当前可用的IP，并把端口号设置为2000，创建Socket对象，并在对话框中打印相关信息。当服务器的Socket接收到来自客户端的数据时，会触发OnReceive函数，OnReceive()判断请求命令是否合法，若合法，相应地返回当前日期，格式为“xx/xx/xx”，或返回当前时间，格式为“xx:xx:xx”，若为错误请求，则返回“错误请求”，在对话框中打印相关信息。流程图如下：

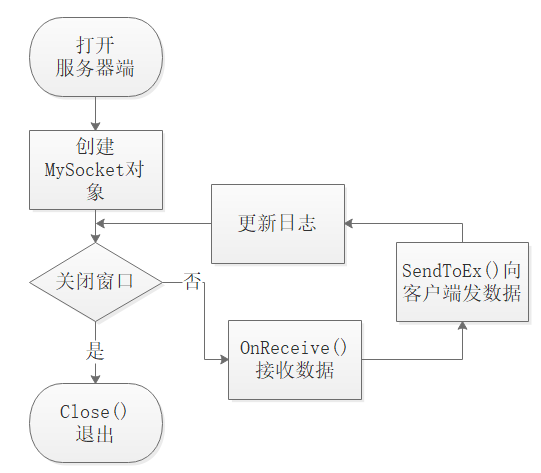


图 3 服务器流程图

另外，在实现基本收发功能的基础上，增加了对客户端信息发送失败的情况的处理，当客户端向服务器端发送请求失败时，会弹出消息框提醒用户，便于重发机制的实现。并且考虑了服务器与多个客户端通信的情况，对多个用户的信息进行保存和显示。

## 程序实现

### 客户端

1. MySocket类

MySocket类定义如下，增加了recIp和recPort两个变量，用于记录接受的IP地址和端口号。

1. **class** MySocket : **public** CAsyncSocket
2. {
3. **public**:
4. MySocket();
5. **virtual** ~MySocket();
6. **virtual** **void** OnReceive(**int** nErrorCode);
8. **public**:
9. CString  recIp;     //存放从主机接受的IP地址
10. **UINT**     recPort;   //存放从主机接受的端口号
11. };

重载OnReceive()，接收到数据时，会触发ReceiveFrom函数，将接收的数据报存入dataBuff，进行数据类型转换后，传递给对话框：

1. **void** MySocket::OnReceive(**int** nErrorCode)
2. {
3. **TCHAR** dataBuff[4096];  //缓冲区接收主机返回的数据报
4. ReceiveFrom(dataBuff, **sizeof**(dataBuff), recIp, recPort);    //从主机接收数据
5. CString data(dataBuff);
7. CUDPClientDlg\* clientDlg = (CUDPClientDlg\*)theApp.m\_pMainWnd;
8. clientDlg->serverReceive=data;
9. clientDlg->UpdateData(**false**);
11. CAsyncSocket::OnReceive(nErrorCode);
12. }
13. CUDPClientDlg类

CUDPClientDlg类中增加的成员变量及函数：

1. **public**:
2. CString serverIp;   //服务器的IP地址
3. **int** serverPort;     //服务器的端口值
4. CString serverReceive;  //服务器响应的数据报
5. CString command;    //向服务器请求的命令
6. MySocket mysocket;  //套接字
7. afx\_msg **void** OnSend();  //点击“发送”
8. afx\_msg **void** OnClose();

在OnInitDialog()中产生端口号，创建套接字的代码如下。最开始的AfxSocketInit()包装了函数WSAStartup()，作用是对Csocket进行初始化。UDP端口使用4位十六进制数表示，其中知名端口：0 – 1023，动态端口：1024 – 65535，由此产生合理的端口号，以数据报方式创建socket：

1. **if** (!AfxSocketInit()) {//初始化
2. MessageBox(L"初始化失败", L"提示", MB\_OK|MB\_ICONSTOP);
3. }
5. //随机产生一个客户端口号,知名端口：0 - 1023,动态端口：1024 - 65535
6. srand((unsigned)time(NULL));
7. **int** client\_port=(**int**)(rand()%8000+1024);
9. **if** (!mysocket.Create(client\_port, SOCK\_DGRAM, FD\_READ)) {//以数据报方式创建socket
10. MessageBox(L"Socket套接字创建失败", L"错误", MB\_OK|MB\_ICONSTOP);
11. };

OnSend()函数代码如下。在客户端窗口中的“发送”按钮绑定了OnSend()函数，用户在客户端窗口输入服务器IP和端口以及请求命令，OnSend更新控件对应的变量值后，进行判断，若输入正确，则使用Socket的SendToEX()方法，向服务端发送数据报。这里使用SendToEx()是为了支持Ipv6。此外，为了处理发送失败的情况，需要判断SendToEx()的返回值是否是SOCKET\_ERROR，如果是的话，需要弹出窗口告诉用户发送失败。

1. **void** CUDPClientDlg::OnSend()
2. {
3. UpdateData(**true**);   //更新控件对应的变量值
4. **if**(serverIp.IsEmpty())
5. {MessageBox(\_T("无服务器Ip地址！"),\_T("错误！"),MB\_OK|MB\_ICONEXCLAMATION); **return** ;}
6. **if**(command.IsEmpty())
7. {MessageBox(\_T("无请求命令！"),\_T("错误！"),MB\_OK|MB\_ICONEXCLAMATION); **return** ;}
8. **if**(serverPort <=0 || serverPort> 65535)
9. {MessageBox(\_T("端口值不正确！"),\_T("错误！"),MB\_OK|MB\_ICONEXCLAMATION);**return**;}
11. //向服务器发送数据报  UDP方式
12. **int** flag=mysocket.SendToEx(command.GetBuffer(),(command.GetLength() + 1) \* **sizeof**(**WCHAR**),serverPort,serverIp);
13. **if** (flag == SOCKET\_ERROR) {
14. MessageBox(\_T("向服务器发送请求失败！"),\_T("错误！"),MB\_OK|MB\_ICONSTOP); **return**;
15. }
16. }

Onclose()函数负责关闭套接字窗口和对话框，这里需要关闭套接字，否则可能出现端口号被重复使用，导致套接字创建失败的情况。

1. **void** CUDPClientDlg::OnClose()
2. {
3. mysocket.Close();       //关闭套接口
4. CDialogEx::OnClose();
5. }

### 服务器

1. MySocket类

重载OnReceive()，获取服务器对话框的句柄，调用ReceiveFrom()函数获取客户端IP和port以及发来的数据，将请求命令转为字符串后存入require中，更新当前的日期和时间，根据请求命令设置result的值为相应内容或“错误请求”。调用SendToEx()函数向客户端发送数据，在窗口上输出记录。

1. **void** MySocket::OnReceive(**int** nErrorCode)
2. {
3. //获取对话框句柄
4. CUDPServerDlg\* serverDlg = (CUDPServerDlg\*)theApp.m\_pMainWnd;
5. **TCHAR** ipBuff[4096];
6. CString revIP;
7. **UINT** revPort;
9. //分析请求内容
10. ReceiveFrom(ipBuff, 4096, revIP, revPort);
11. CString require(ipBuff);
12. serverDlg->update\_dt(serverDlg->mydate,serverDlg->mytime);
13. CString result=\_T("错误请求");
14. require.MakeLower();
15. **if**(require=="time")
16. result=serverDlg->mytime;
17. **if**(require=="date")
18. result=serverDlg->mydate;
20. //响应客户端程序
21. SendToEx(result.GetBuffer(), (result.GetLength() + 1) \* **sizeof**(**WCHAR**), revPort, revIP);
22. CString reply;
23. reply.Format(L": 收到IP=%s Port=%u 请求【 %s 】,响应【 %s 】", revIP, revPort,require,result);
24. serverDlg->mylist.AddString(serverDlg->mydate+\_T(" ")+serverDlg->mytime+reply);
26. CAsyncSocket::OnReceive(nErrorCode);
27. }

OnReceive()函数的控制流图如下：

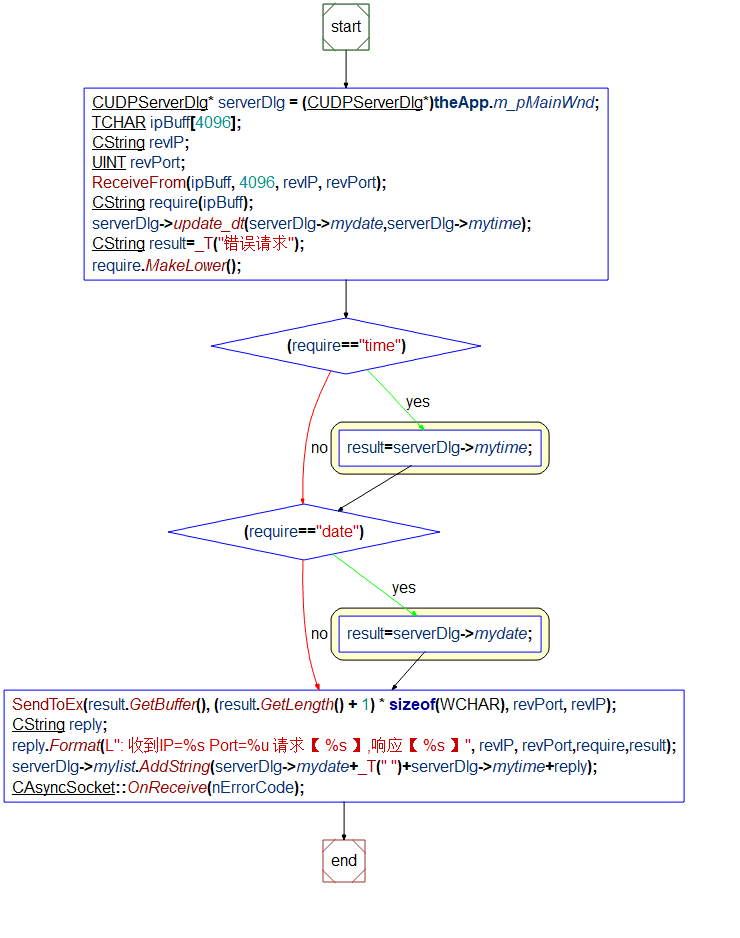


图 4 Server\_ClusterControlFlow-OnReceive

1. CUDPServerDlg类

CUDPServerDlg类中增加的成员变量及函数：

1. **public**:
2. CString mytime;     //当前服务器的时间
3. CString mydate;     //当前服务器的日期
4. **int** myport;         //当前服务器的端口
5. CString myip;       //当前服务器IP地址
6. CListBox mylist;    //记录框
7. MySocket mysocket;
8. **void** update\_dt(CString& date, CString& time);//更新时间和日期
9. afx\_msg **void** OnClose();

OnInitDialog()中对CSocket进行初始化，获取主机名：

1. update\_dt(mydate,mytime);//更新日期和时间
3. **if** (!AfxSocketInit()) {//初始化
4. MessageBox(L"初始化失败", L"提示", MB\_OK|MB\_ICONSTOP);
5. }
6. update\_dt(mydate,mytime);//更新日期和时间
7. mylist.AddString(mydate+\_T(" ")+mytime+\_T(":")+\_T("Socket初始化成功！"));
9. //获取主机名
11. **char** name[20] = "";
12. **int** errorcode;
13. **if** ((errorcode = gethostname(name, **sizeof**(name))) != 0) {
14. **char** char\_error[100];
15. \_itoa\_s(errorcode, char\_error, 10);
16. MessageBoxA(**this**->GetSafeHwnd(), char\_error, "errorcode", MB\_OK);
17. AfxGetMainWnd()->SendMessage(WM\_CLOSE);
18. };

获取可用 IP用到了gethostbyname()函数，hostent是host entry的缩写，该结构记录主机的信息，包括主机名、别名、地址类型、地址长度和地址列表。 主机的地址是一个列表的形式，因为当一个主机有多个网络接口时，自然有多个地址，我在这里只取了第一个作为服务器的IP地址。

这里需要做一些类型转换，主机的ip地址是以网络字节序存储的，需要打印的话，需要调用inet\_ntoa()，SHAnsiToUnicode()函数将ANSI编码转为Unicode，再用Format()函数将char\*转换成Cstring，赋值给记录当前服务器IP地址的成员变量。

1. //获取可用 IP 列表，这里可能有多个可用 IP，暂时只取第一个
2. hostent \*hostlist;
3. hostlist = gethostbyname(name);
4. **int** i = 0;
5. **while** (hostlist->h\_addr\_list[i] != 0) {
6. //主机的ip地址是以网络字节序存储的，需要打印的话，需要调用inet\_ntoa()
7. **char** \*p = inet\_ntoa(\*(in\_addr \*)hostlist->h\_addr\_list[i++]);
8. **wchar\_t** pw[20];
9. //converts as much as possible of an ANSI string to a Unicode string
10. SHAnsiToUnicode(p, pw, 20);
11. CString str;
12. str.Format(L"%s", pw);//将char\*转换成CString
13. myip = str;//默认只取第一个ip值
14. **break**;
15. }
16. **if** (i == 0) {
17. AfxGetMainWnd()->SendMessage(WM\_CLOSE);
18. }

创建套接字，判断IP和端口的初始化情况，若出现错误，则给出提示，否则调用Create()函数开启服务器，更新日期和时间并在窗口输出日志。

1. //启动服务器操作
2. **if**(myip.IsEmpty())
3. {MessageBox(\_T("无服务器Ip地址！"),\_T("错误！"),MB\_OK|MB\_ICONEXCLAMATION); **return** TRUE;}
4. **if**(myport <=0 || myport> 65535)
5. {MessageBox(\_T("端口值设置错误！"),\_T("错误！"),MB\_OK|MB\_ICONEXCLAMATION); **return** TRUE;}
7. UpdateData(**true**);
8. **if** (!mysocket.Create(myport, SOCK\_DGRAM, FD\_READ,myip)) {//以数据报方式创建socket
9. MessageBox(L"Socket套接字创建失败", L"错误", MB\_OK|MB\_ICONSTOP);**return** TRUE;
10. };
11. CString record;
12. record.Format(L"启动服务器成功！当前主机IP是：【 %s 】，端口号是：【 %u 】", myip,myport);
13. update\_dt(mydate,mytime);//更新日期和时间
14. mylist.AddString(mydate+\_T(" ")+mytime+\_T(":")+record);

update\_dt()函数按照固定格式，更新当前时刻的日期和时刻值：

1. **void** CUDPServerDlg::update\_dt(CString& date, CString& time)
2. {
3. date=CTime::GetCurrentTime().Format(L"%Y/%m/%d");
4. time=CTime::GetCurrentTime().Format("%H:%M:%S");
5. UpdateData(**false**);
6. }

OnClose()函数负责关闭套接字窗口和对话框，若未关闭套接字，可能出现端口号被重复使用，套接字创建失败。

1. **void** CUDPServerDlg::OnClose()
2. {
3. mysocket.Close();
4. CDialogEx::OnClose();
5. }

## 效果演示

UDPClient窗口中，ServerIP控件获取用户输入的服务器IP地址；ServerPort控件获取用户输入的服务器端口；Request控件获取用户输入的请求命令，输入“date”表示请求服务器当前日期，输入“time”表示请求服务器当前时间，其他请求为错误请求；Response控件显示服务器返回的值，根据请求命令的内容，这里会显示时间或日期或“错误请求”。

UDP服务器工作日志记录了服务器的一系列动作和动作发生的时间。在开始阶段会显示初始化是否成功，若Socket对象创建成功，接着会显示当前主机的IP地址和端口号。然后显示的是与客户端进行数据收发的记录，格式为：“xx/xx/xx xx:xx:xx：收到IP=xxx.xxx.xxx.x Port=xxxx请求【xxx】,响应【xxx】”。

客户端与服务器的交互效果如下：

1. 客户端输入错误的IP地址或端口号，弹出窗口信息“向服务器发送请求失败！”，如下图所示：

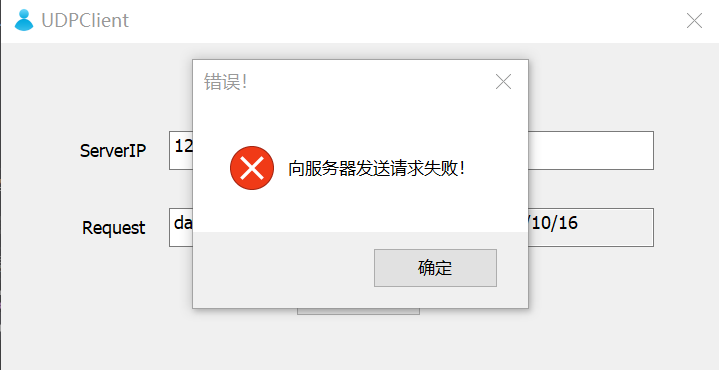


图 5 错误的IP地址或端口号

1. 请求日期，输入当前主机IP“192.168.163.1”以及端口号“2000”，请求命令为“date”（不区分大小写）,返回结果为当时的日期“2019/10/16”,并且服务器日志会增加一条信息：

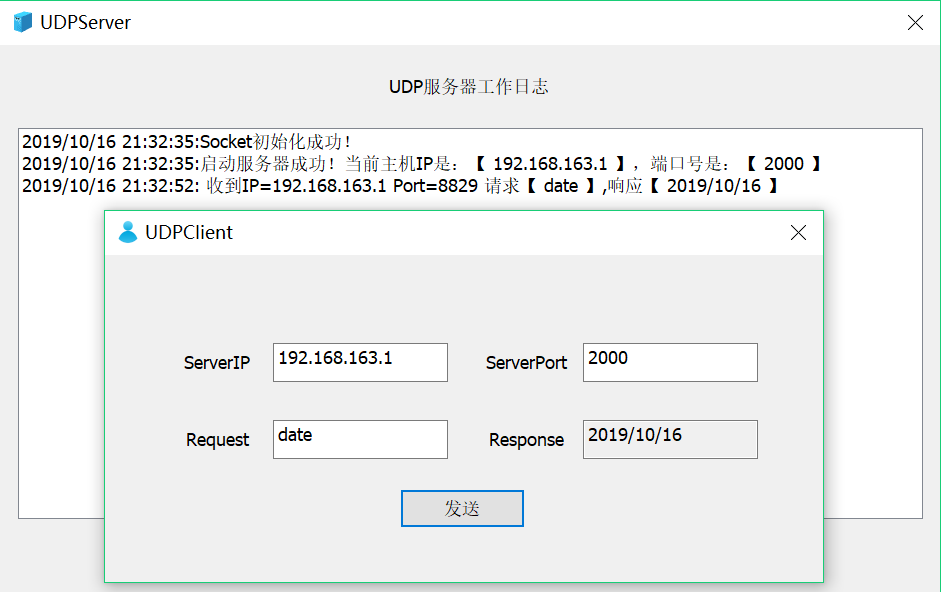


图 6 请求日期

1. 请求时间，输入当前主机IP“192.168.163.1”以及端口号“2000”，请求命令为“TIme”（不区分大小写）,返回结果为当时的日期“21:33:31”,并且服务器日志会增加一条信息：

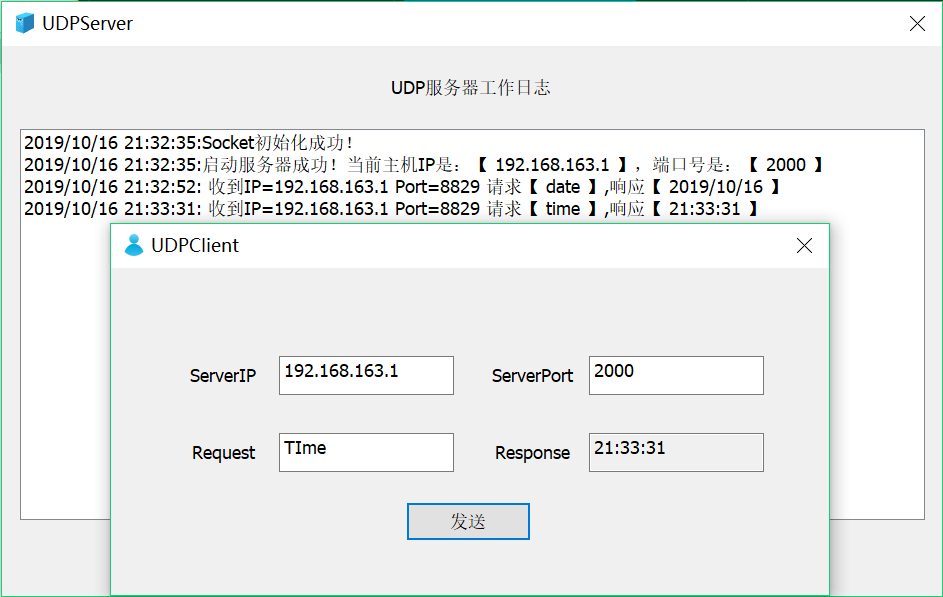


图 7 请求时间

1. 错误命令，当输入的IP和端口号都没有问题，但请求命令错误时，会得到响应“错误请求”，如下图所示：

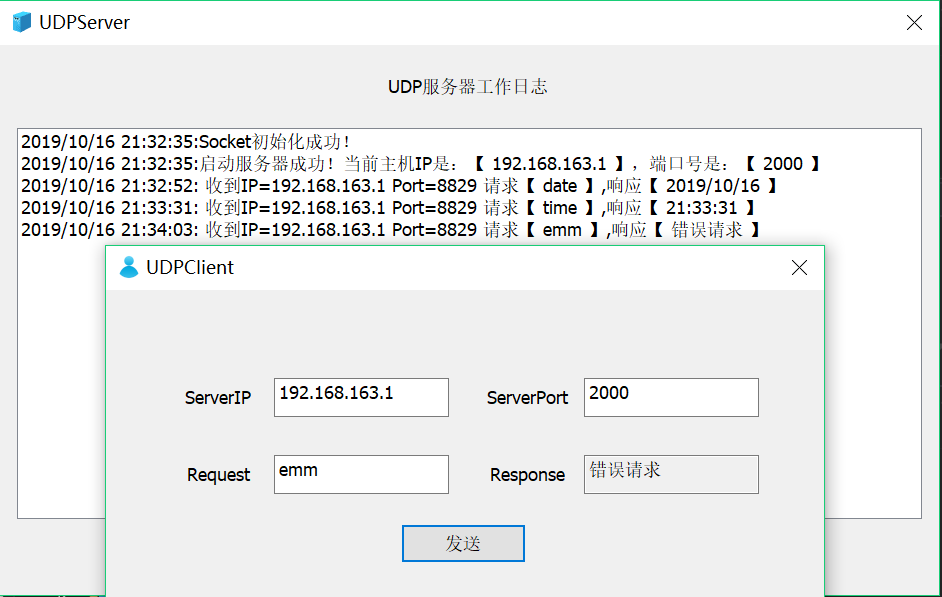


图 8 错误命令

1. UDP服务器与端口号为8643、8950、1390、1207等多个客户端进行响应，工作日志如下图：

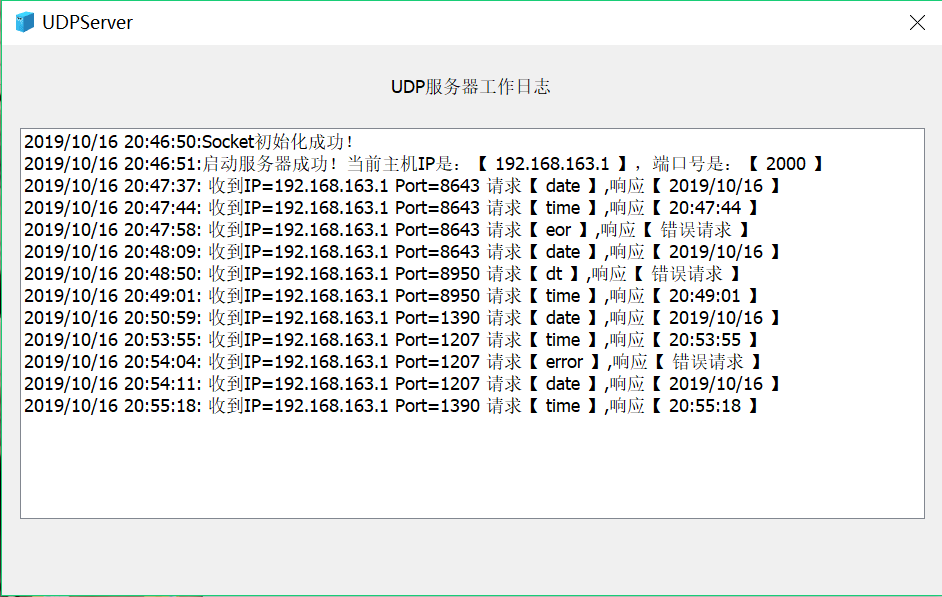


图 9 服务器工作日志

## 其他

关于UDP实验的部分在上面已经全部叙述。由于之前没有使用过MFC，所以这次做实验的过程中，接触了许多关于MFC的内容，新学到了一些变量类型，还有关于Socket的一些函数，在这里做一些笔记。

1. 在MFC程序中，CString既可以处理Unicode标准的字符串，也可以处理ANSI标准的字符串。
2. GetBuffer()主要作用是将字符串的缓冲区长度锁定。
3. 字符串前面加L表示该字符串是Unicode字符串。
4. \_T是一个宏，如果项目使用了Unicode字符集（定义了UNICODE宏），则自动在字符串前面加上L，否则字符串不变。因此，Visual C++里边定义字符串的时候，用\_T来保证兼容性。VC支持ascii和unicode两种字符类型，用\_T可以保证从ascii编码类型转换到unicode编码类型的时候，程序不需要修改。
5. wchar\_t是C/C++的字符类型，是一种扩展的存储方式。unicode编码的字符一般以wchar\_t类型存储。
6. afx\_msg为消息标志，它向系统声明：有消息映射到函数实现体；
7. DDX\_Text函数管理着对话框、表格视或控件视对象中的编辑控件与对话框、表格视或控件视对象的CString型数据成员之间的int，UINT，long，DWORD，CString，float或double型数据交换。
8. void AFXAPI DDX\_Control(CDataExchange\* pDX, int nIDC, CWnd& rControl)将ID为nIDC的控件与类型为CWnd的变量rControl绑定
9. void AFXAPI DDX\_CBString( CDataExchange\* pDX, int nIDC, CString& value ) 管理着在对话框、表格视或控件视中的组合框中的编辑控件与对话框、表格视或控件视的CString数据成员之间的CString数据交换。
10. BOOL Create( UINT nSocketPort = 0, int nSocketType = SOCK\_STREAM, long lEvent = FD\_READ |FD\_WRITE|FD\_OOB|FD\_ACCEPT|FD\_CONNECT| FD\_CLOSE，LPCTSTR lpszSocketAddress = NULL ) 该函数用来建立Socket。 其中，nSocketPort 为所选择的Socket 端口，一般要大于 1023， 如果该参数为0，则由系统选定一端口，默认值为0 ；nSocketType 为套接字类型：SOCK\_STREAM 表示为流套接字，SOCK\_DGRAM 表示为数据报套接字，默认值为SOCK\_STREAM ；lEvent 标识该Socket 要完成哪种工作，默认值为:

FD\_READ|FD\_WRITE|FD\_OOB| FD\_ACCEPT|FD\_CONNECT|FD\_CLOSE ；lpszSockAddress 为网络地址信息结构指针，包含网络地址， 默认值为NULL 。

1. 关于Socket的send()、sendTo()、sendToEx()的区别：

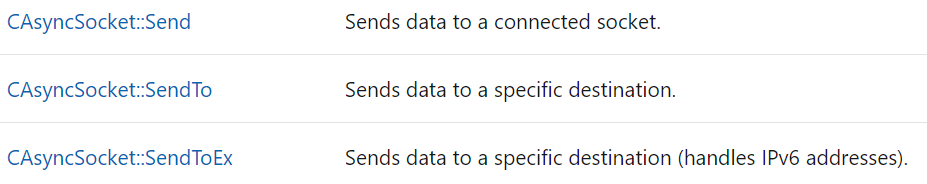


图 10 send()等的区别