|  |
| --- |
|  |
| 多人在线聊天系统 |
| 软件设计说明书 |

|  |
| --- |
| 殷康龙、张猛  2018-3-16 |

目录

[1任务概述 2](#_Toc508990311)

[1.1编写目的 2](#_Toc508990312)

[1.2项目背景 2](#_Toc508990313)

[1.3参考资料 2](#_Toc508990314)

[2概要设计 2](#_Toc508990315)

[2.1整体框架 2](#_Toc508990316)

[2.2需求概述 4](#_Toc508990317)

[2.2.1 服务器端功能需求 4](#_Toc508990318)

[2.2.2 客户端功能需求 4](#_Toc508990319)

[2.3运行环境 4](#_Toc508990320)

[2.3.1服务器端 4](#_Toc508990321)

[2.3.2客户端： 5](#_Toc508990322)

[2.4开发工具 5](#_Toc508990323)

[3详细设计 5](#_Toc508990324)

[3.1服务器功能模块 5](#_Toc508990325)

[3.2客户端功能模块 9](#_Toc508990326)

[3.3服务器与客户端通信（TCP） 14](#_Toc508990327)

[3.4客户端与客户端之间通信（UDP） 15](#_Toc508990328)

[3.4.1UDP服务器端过程 15](#_Toc508990329)

[3.4.2UDP客户端过程： 16](#_Toc508990330)

[3.5界面设计 17](#_Toc508990331)

# 1任务概述

## 1.1编写目的

此项目为西北工业大学网络编程的课程设计选题。目标是在VS下，使用Win32网络编程和MFC框架实现多人在线聊天的服务器端和客户端。用户能够选择网络中的聊天对象，通过UDP广播或者通过TCP对指定用户进行通信，方便用户的交流。最终，在windows操作系统下实现多人在线即时聊天功能。

## 1.2项目背景

现在，即时聊天系统是目前Internet 上最为流行的通讯方式，而各种各样的即时通讯软件也层出不穷；服务提供商也提供了越来越丰富的通讯服务功能。随着互联网的发展，即时通讯的运用将日益广泛。

广域网的聊天系统多重多样，知名的软件主要有Facebook、腾讯QQ等。局域网聊天通信软件也有很多，最著名的应该是飞秋。为了学习和应用Windows网络通信编程，我们学习了相关知识，为了应用实践，使用网络通信中的TCP和UDP编程，实现了多人在线聊天系统。

## 1.3参考资料

《Visual\_C++\_6.0程序设计从入门到精通》

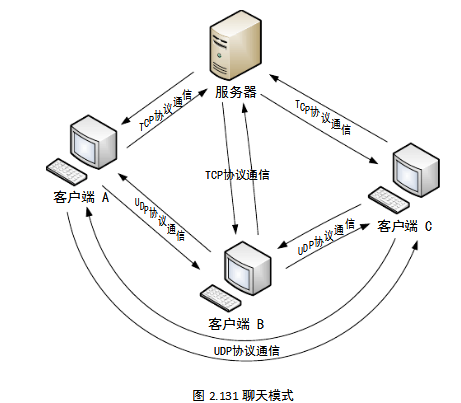
《VC++深入讲解》

《Windows程序设计（王艳平，第二版）》

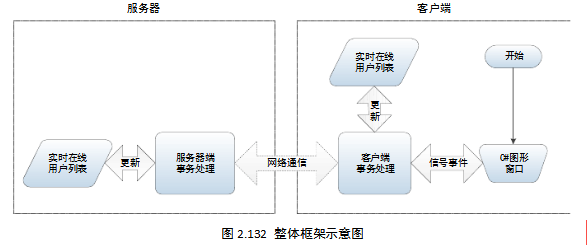
《GDI+SDK参考（翻译版本）》

# 2概要设计

## 2.1整体框架



如上图所示，服务器与客户端之间采用TCP/IP协议通信，客户端与客户端之间通过 UDP通信。服务器专门负责记录当前在线用户列表，并能进行广播。服务器持续保持对端口的侦听状态，每当有主机上线时，首先连接至服务器，服务器收到连接后，将该主机的进行UDP通信的IP和端口，发往其他在线主机。这样其他主机便知道该主机已上线，并知道其所在位置，从而可以进行连接和对话。在服务器进行了广播之后，因为各个主机已经知道了其他主机的位置，因此主机之间的对话可以直接进行连接。因此，使用这种模式时，各个主机依然需要保持对端口的侦听。在某台主机离线时，与登录时的模式类似，服务器会收到通知，然后转告给其他的主机。



## 2.2需求概述

### 2.2.1 服务器端功能需求

1. 服务器能够选择端口号并且启动服务器端的TCP服务，等待客户端的Socket连接。
2. 服务器能够接受多个客户端的请求，并且能够识别不同的客户端。为每个客户端建立唯一的身份表示。
3. 连接成功后，服务器能接收客户端发过来的文本信息并显示。
4. 服务器能够选择用户，并通过界面信息框发送消息。
5. 服务器能够清空消息缓冲区域的内容。
6. 服务器能够选择端口号并启动服务器端的UDP服务，并且等待客户端的数据包，接受数据并且显示。
7. 服务器端能够显示本机的地址和状态。

### 2.2.2 客户端功能需求

1. 某用户刚登录服务器时，服务器需对其发送实时在线用户列表。
2. 用户可以通过指定主机地址和端口号连接指定的服务器。用户在连接过程中可以选择TCP或者UDP协议进行连接。
3. 某用户登录或退出程序时都需要给服务器发送一个消息以通知其他用户。
4. 用户能够通过界面发送消息。
5. 用户界面能够显示历史消息，包括其他用户传递的消息和用户自身发送的消息。用户能够通过界面清空历史消息缓冲区
6. 状态栏能够显示用户当前连接的状态。

## 2.3运行环境

### 2.3.1服务器端

硬件：CPU推荐Pentium IV 1GHz以上；

内存：推荐256MB以上；

硬盘：数据库空间2GB以上。

### 2.3.2客户端：

操作系统：Windows7及以上；

硬件：CPU推荐Pentium IV 2GHz以上；

硬盘：系统安装需10G以上。

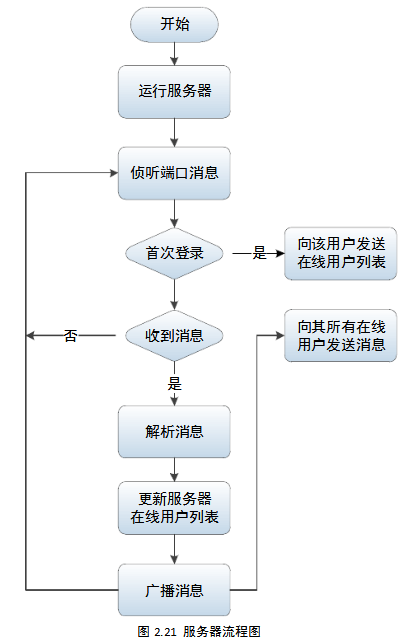
## 2.4开发工具

VC++6.0或以上版本；

操作系统：Windows7及以上。

# 3详细设计

## 3.1服务器功能模块



1. 首先服务器初始化MFC编程和Win32编程的基础环境，将界面窗口绑定到当前的APP中，并进行初始化，运行服务器端的界面程序。
2. 服务器端初始化界面控件，从界面获取TCP和UDP的本地地址和端口号，通过新建Socket，绑定到本地地址和端口号，运行Socket服务。
3. 服务器端采用MFC的消息响应机制，当客户端对界面的控件进行操作时，会触发相应的消息，对不同的消息建立不同的逻辑处理功能。
4. 服务器端采用异步消息框架对客户端的连接进行处理。当用户发送进行连接、发送消息、取消连接的时候都会出发相应的消息，然后通过对消息的处理，完成对客户端的相应。

// 设置socket为窗口通知消息类型,异步接受消息

::WSAAsyncSelect(m\_socket, m\_hWnd, WM\_SOCKET, FD\_ACCEPT|FD\_CLOSE);

// 进入监听模式

::listen(m\_socket, 5);

1. 当用户进行连接时，捕获连接请求，将用户的socket添加到用户列表当中。显示用户的相关信息。

case FD\_ACCEPT: // 监听中的套接字检测到有连接进入

{

if(m\_nClient < MAX\_SOCKET)

{

//定义接受者

sockaddr\_in sockAddr;

memset(&sockAddr, 0, sizeof(sockAddr));

int nSockAddrLen = sizeof(sockAddr);

// 接受连接请求，新的套节字client是新连接的套节字,包括客户端的地址和端口号

SOCKET client = ::accept(s,(SOCKADDR\*)&sockAddr,&nSockAddrLen);

// 设置新的套节字为窗口通知消息类型

int i = ::WSAAsyncSelect(client,

m\_hWnd, WM\_SOCKET, FD\_READ|FD\_WRITE|FD\_CLOSE);

SocketAddr sa={};

sa.sock=client;

sa.sockAddr=sockAddr;

AddClient(sa);

}

else

{

MessageBox("连接客户太多！");

}

1. 当用户发送消息时，捕获用户消息，将用户的消息显示到界面当中。

case FD\_READ: // 套接字接受到对方发送过来的数据包

{

// 取得对方的IP地址和端口号（使用getpeername函数）

// Peer对方的地址信息

sockaddr\_in sockAddr;

memset(&sockAddr, 0, sizeof(sockAddr));

int nSockAddrLen = sizeof(sockAddr);

//通过peer的方式得到对方的ip地址和端口号信息

::getpeername(s, (SOCKADDR\*)&sockAddr, &nSockAddrLen);

//得到端口号,并将端口号转化为字符串

u\_short port = sockAddr.sin\_port;

char cs\_port[10];

sprintf\_s(cs\_port,10,"%d",int(port));

// 转化为主机字节顺序

int nPeerPort = ::ntohs(sockAddr.sin\_port);

// 转化为字符串IP

CString sPeerIP = ::inet\_ntoa(sockAddr.sin\_addr);

// 取得对方的主机名称

// 取得网络字节顺序的IP值

DWORD dwIP = ::inet\_addr(sPeerIP);

// 获取主机名称，注意其中第一个参数的转化

hostent\* pHost = ::gethostbyaddr((LPSTR)&dwIP, 4, AF\_INET);

char szHostName[256];

strncpy\_s(szHostName,256, pHost->h\_name, 256);

// 接受真正的网络数据

char szText[1024] = { 0 };

::recv(s, szText, 1024, 0);

// 显示给用户

CString strItem = CString(szHostName) + "["+ sPeerIP+ "/"+CString(cs\_port)+ "]: " + CString(szText);

AddStringToList(strItem,TRUE);

//m\_listInfo.InsertString(-1,strItem);

}

1. 当用户切断连接时，捕获用户断开连接的请求。

case FD\_CLOSE: // 检测到套接字对应的连接被关闭。

{ //定义接受者

// 设置新的套节字为窗口通知消息类型

SocketAddr sa ={};

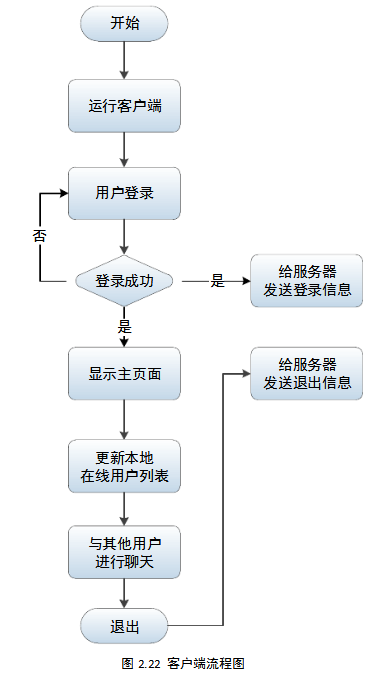
sa.sock=s;

RemoveClient(sa);

::closesocket(s);

}

## 3.2客户端功能模块



1. 运行客户端，初始客户端的应用，并启动客户端的推图形界面。

CDialog::OnInitDialog();

// 设置图标

SetIcon(theApp.LoadIcon(IDI\_MAIN), FALSE);

// 创建状态栏，设置它的属性

m\_bar.Create(WS\_CHILD|WS\_VISIBLE|SBS\_SIZEGRIP, CRect(0, 0, 0, 0), this, 101);

m\_bar.SetBkColor(RGB(0xa6, 0xca, 0xf0)); // 背景色

int arWidth[] = { 200, -1 };

m\_bar.SetParts(2, arWidth); // 分栏

m\_bar.SetText(" TCP and UDP Client by yinkanglong and zhangmeng ", 1, 0); // 第一个栏的文本

m\_bar.SetText(" 空闲", 0, 0); // 第二个栏的文本

// 初始化发送按钮和发送编辑框的状态

GetDlgItem(IDC\_SEND)->EnableWindow(FALSE);

GetDlgItem(IDC\_TEXT)->EnableWindow(TRUE);

// 初始化连接套节字

m\_socket = INVALID\_SOCKET;

m\_UDPsocket = INVALID\_SOCKET;

return TRUE;

1. 用户选择指定的服务器地址和端口号，进行UDP和TCP的连接。并能够显示连接状态。

case FD\_CONNECT: // 套节字正确的连接到服务器

{

// 设置用户界面

GetDlgItem(IDC\_CONNECT)->SetWindowText("断开连接");

GetDlgItem(IDC\_ADDR)->EnableWindow(FALSE);

GetDlgItem(IDC\_PORT)->EnableWindow(FALSE);

GetDlgItem(IDC\_TEXT)->EnableWindow(TRUE);

GetDlgItem(IDC\_SEND)->EnableWindow(TRUE);

m\_bar.SetText(" 已经连接到服务器", 0, 0);

}

1. 用户能够接受其他人发送的消息，通过异步消息处理框架对消息进行处理。

case FD\_READ: // 套接字接受到对方发送过来的数据包

{

// 从服务器接受数据

char szText[1024] = { 0 };

::recv(s, szText, 1024, 0);

// 显示给用户

AddStringToList(CString(szText) + "\r\n");

}

break;

1. 客户端能够启动新的线程，进行UDP的连接。

if(m\_UDPsocket == INVALID\_SOCKET) // 开启服务

{

// 取得目标地址

CString sAddr;

GetDlgItem(IDC\_ADDR)->GetWindowText(sAddr);

if(sAddr.IsEmpty())

{

MessageBox("请输入服务器地址！");

return;

}

// 取得UDP端口号

CString sPort;

GetDlgItem(IDC\_EDIT\_UDP\_PORT)->GetWindowText(sPort);

int nPort = atoi(sPort);

if(nPort < 1 || nPort > 65535)

{

MessageBox("端口号错误！");

return;

}

remoteUDPAddress.sin\_family=AF\_INET;

remoteUDPAddress.sin\_port = htons(nPort);

remoteUDPAddress.sin\_addr.S\_un.S\_addr = inet\_addr(sAddr);

// 创建监听套节字

m\_UDPsocket = socket(AF\_INET,SOCK\_DGRAM,IPPROTO\_UDP);

if (m\_UDPsocket == SOCKET\_ERROR)

{

MessageBox("启动服务出错！");

return;

}

//绑定随机地址信息

sockaddr\_in sin;

sin.sin\_family = AF\_INET;

sin.sin\_port = 0;

sin.sin\_addr.s\_addr = INADDR\_ANY;

// 绑定端口

if(::bind(m\_UDPsocket,(sockaddr\*)&sin,sizeof(sin)) == SOCKET\_ERROR)

{

MessageBox("端口号绑定失败");

return;

}

//开启多线程，用来接收数据包

//MessageBox("UDP线程启动前");

h\_udpThread = ::CreateThread(NULL,0,(LPTHREAD\_START\_ROUTINE)udpRec,this,0,NULL);

//udpRec();

//MessageBox("UDP线程启动后");

//if(!CreateAndListen(nPort))

//{

// MessageBox("启动服务出错！");

// return;

//}

// 设置相关子窗口控件状态

GetDlgItem(IDC\_BUTTON\_UDP\_START)->SetWindowText("停止UDP客户端");

//m\_bar.SetText(" 正在监听……", 0, 0);

//GetDlgItem(IDC\_EDIT\_UDP)->EnableWindow(FALSE);

}

else // 停止服务

{

MessageBox("UDP服务终止");

// 关闭所有UDP连接

CloseUDPSocket();

// 设置相关子窗口控件状态

GetDlgItem(IDC\_BUTTON\_UDP\_START)->SetWindowText("开启UDP客户端");

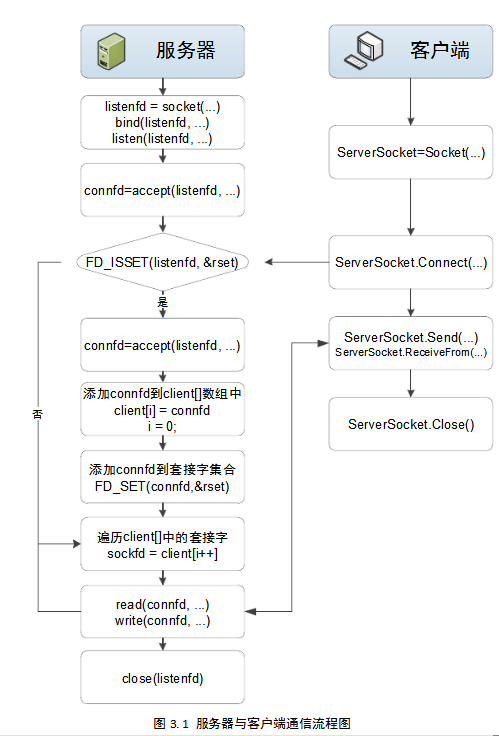
//m\_bar.SetText(" 空闲", 0, 0);

//GetDlgItem(IDC\_EDIT\_UDP)->EnableWindow(TRUE);

}

1. 用户能够将记录当前连接的其他用户。

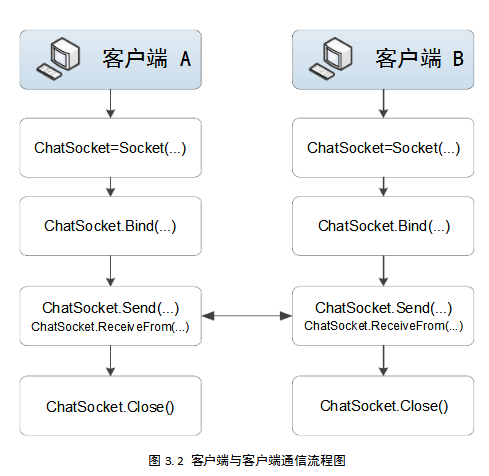
## 3.3服务器与客户端通信（TCP）



**tcp过程**

1. 首先初始化CWinAPP函数的一些东西，包括初始化唯一的窗口变量
2. 点击开始OnStart函数触发，取得端口号，设置了控件状态
3. OnStart调用CreateAndListen函数，创建了服务器Socket服务，并采用异步消息监听模式
4. OnSocket负责接收TCP异步消息进行处理。接受Socket请求，关闭Socket请求，读取Socket数据流
5. 其中由辅助函数AddClient RemoveClient UpdateClientList实现连接列表的控制。
6. 由辅助函数CloseAllSocket进行关闭连接，AddStringToList用来显示数据
7. 发送部分首先由OnBnClickedButtonSelect函数对发送对象列表进行控制
8. 由OnBnClickedButtonSend函数实现内容发送到发送对象列表

## 3.4客户端与客户端之间通信（UDP）



### 3.4.1UDP服务器端过程

1. 点击开始OnBnClickedButtonUdp函数触发，取得端口号，设置了控件状态。
2. OnBnClickedButtonUdp使用多线程调用了udpRec函数，实现了多线程启动udp接受服务。
3. udpRec函数负责接受同步的UDP消息，并对消息进行显示和处理。
4. 其中由辅助函数AddUDPClient RemoveUDPClient对UDP用户列表实现控制
5. 由副主函数CloseUDPSocket进行关闭连接
6. #发送部分首先OnBnClickedButtonUdpSedn对UDP用户列表遍历发送数据包

### 3.4.2UDP客户端过程：

1. 点击开始OnBnClickedButtonUdp函数触发，取得端口号，设置了控件状态。
2. OnBnClickedButtonUdp使用多线程调用了udpRec函数，实现了多线程启动udp接受服务。
3. udpRec函数负责接受同步的UDP消息，并对消息进行显示和处理。
4. 由副主函数CloseUDPSocket进行关闭连接

## 3.5界面设计

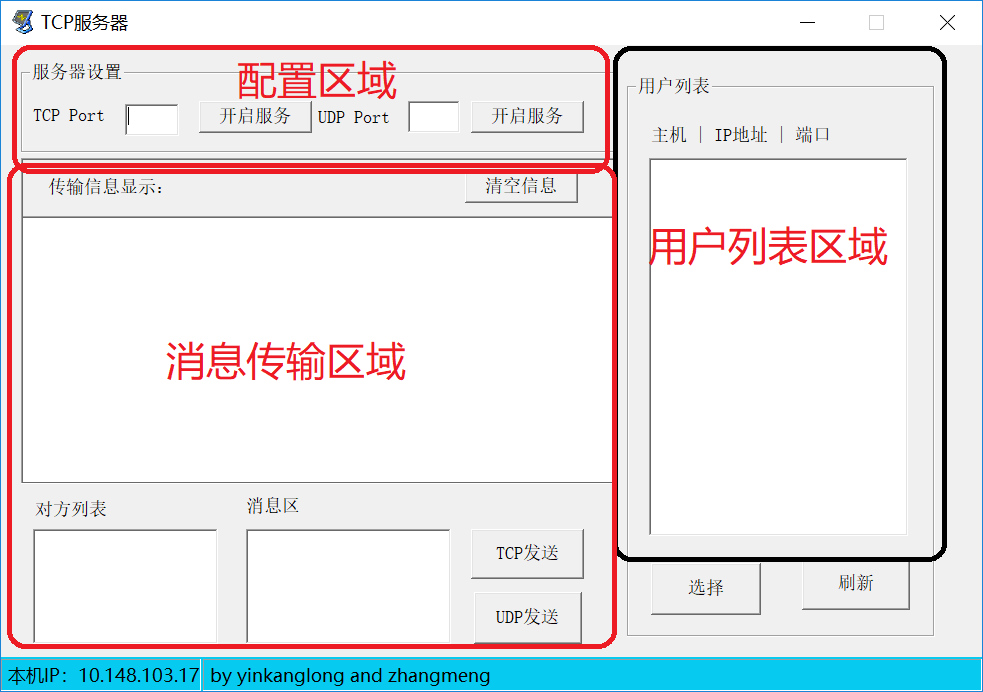


Figure 1 服务器界面设计

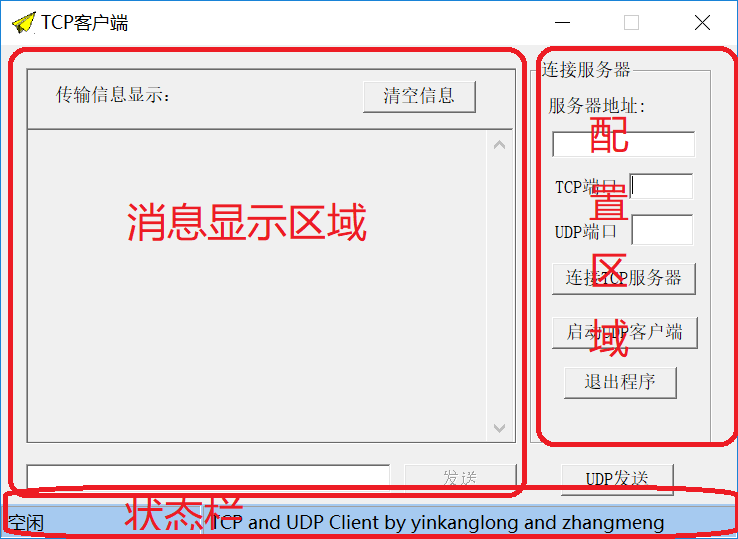


Figure 2 客户端界面设计