华中科技大学文华学院

毕业设计（论文）

题目：多人会议系统的设计和实现

学 生 姓 名：[redacted] 学号：[redacted]

学 部（系）：信息学部

专 业 年 级：06 级软件工程

指 导 教 师：[redacted] 职称或学位：讲师

2010年 5 月 19 日

目 录

多人会议系统的设计和实现

摘 要

多人会议系统主要采用了微软的 MFC 框架，共同使用了 CSocket 与 CArchive 对象。这两个对象是最简单的套接字编程模型，其中 CArchive 对象将帮助程序员处理许多以前必须使用 API 或 CAsyncSocket 类来处理的通信问题，大大减少了编程的工作量。系统的实现采用了串行化技术。

系统主要实现了参加会议人员的聊天、电子白板功能。聊天主要是群聊，类似于QQ群聊，而且用户还能看到当前在线用户列表。当然服务器端也能看到当前用户列表和在线人员的聊天信息。电子白板模块主要实现了一个类似于绘图的程序，但它能够实现网络通信，即一个客户端所绘的图形，其它在线用户只要打开电子白板视图就能看到对方所绘制的图形。绘图的图形种类主要有 ：直线、连续直线、任意曲线、圆和圆形区域。而且可以选择绘制线条的粗细、颜色及填充区域的填充样式。

关键词：网络会议系统； Socket编程

Design and Realization of Multi-Participant Conference System

Abstract

Multi-Participant Conference System mainly uses of Microsoft MFC framework, and common use CSocket and CArchive object together. The two objects is the simplest model of socket programming, but CArchive will help programmers deal with the communication problems which must use the API or CAsyncSocket class before , and greatly reduced the programming effort. The System uses the serial technology.

The main function of System to achieve have the chat, whiteboard. The chat function can many person participate , similar to QQ, and the user can present a list of online users. Of course, the server can see the current list of users and online chat personnel information. Whiteboard module like a similar drawing program, but it can achieve network communication, that a client dwarfed graphics, other online users as long as you can see the other side open the Whiteboard view, drawn by the graphics. Drawing graphics types are: straight, continuous line, arbitrary curves, round and round the region. And can choose to draw the line thickness, fill color and fill area style.

Key words: Network Conference System; Socket Programming

前 言

信息化是当今世界发展的大局势，是对动经济社会变革的重要力量。大力推进信息化，是覆盖我国现代化建设全局的战略举措，是贯彻落实科学发展观、全面建设小康社会、构建社会主义和谐社会和建设创新型国家的迫切需求和必然选择。同时，随着网络的普及和通讯条件的改善，传统的通讯方式（如电话、传真）不能满足政府和企业“面对面”交流的需要。传统的会议方式不仅耽误时间，耗费了大量资源，而不能及时取用会议辅助资料，耽误了不少重大决策的产生，损失了时间与效率。许多政府机构和大型企业为了满足自身的远程音视频需求，通过采购硬件视频会议系统，满足了部分需求，使远程会议得以成为现实。但是这些大型硬件设备造价昂贵，使用条件苛刻，网络要求高，数据功能薄弱、无法活变更主会场、无法灵活加入会议，使绝大多数中小企业忘而却步，因此视频会议灵系统一直被认为是一种奢侈的技术。许多政府部门和企业希望能够得到一种更为便宜和便捷的视频会议产品或服务，能够提供多种功能，使他们的远程办公和远程全功能会议得以实现。系统要求视频会议图像的清晰度和图像的连续性。在实现视频交互的同时，要求与会人员利用该系统能实现手写输入电子白板、程序共享、文件传输等数据会议方面的各种功能，以满足多媒体视频会议的需求。

计算机、网络技术高速发展的今天人们对了解事物、交换信息的要求已经从纸、笔、书本、话音等发展到通过声光电信号等各种方式更准确、更快捷、更丰富地表达出来。企业要求能进行远程会议，开展商务交流；同时，在满足传统视频会议音 / 视频通信的基本要求同时，现代企业经营会议更希望能够提供更加丰富多样的会议讨论形式和功能，比如：企业远程开展季度 / 年度预算及工作计划的讨论，需要远程共享 PPT 等演示文档和 EXCEL 等表格，需要共同就某一网上信息开展讨论，需要彼此间如同身临其境地探讨问题。许多企业希望能够得到一种更为便宜和便捷的视频会议产品或服务，能够提多种功能，使他们的远程办公和远程全功能会议得以实现。

毫无疑问，几乎所有的企业都了解沟通的重要性，而且都在这方面做过相应的（力所能及的）拓展，而这其中视频会议系统更是扮演着重要的角色。

1 概 论

1.1 课题的来源及意义

在网络无所不在的今天，在 Internet 上，有 Netmeeting、腾讯 QQ、 MSN-Messager 等等网上聊天或者多人会议系统软件，极大程度上方便了处于在世界各地的友人之间的相互联系以及企业与企业，企业内部人员之间的信息沟通，也使世界好象一下子缩小了，不管你在哪里，只要你上了网，打开这些软件，就可以给你的朋友发信息，不管对方是否也同时在线，只要知道他有号码。

现在，企业、机关、学校都会有自己内部人员信息沟通的软件系统，有的功能非常多，极大的满足不同人的需求，其中多人会议系统比较流行的就数视频会议系统，自己做毕业设计时调研了很多关于这些系统的功能，常见的视频会议系统有：视维网络视频会议系统、 EbaiMeeting 视频会议系统、视高协同视频会议系统等等。它们的功能都大同小异，而且功能越来越多，越来越丰富。例如：视频沟通、音频沟通、协同办公、资料共享、会议决策等等一些功能。而且他们的每一种功能都是很丰富的。

多人会议系统软件，可以说是目前商业公司用户使用率最高的软件之一。

聊天，视频会议，文件共享等等功能一直是商业内部会议所用的网络软件之一，它是迄今为止对人类社会改变最为深刻的一种网络形态，没有极限的沟通将带来没有极限的生活。

多人会议系统软件的走向呈现出以下几大趋势：

首先 ，在应用上更加丰富，更加广泛；

其次，多人会议系统还将更加突显会议联系人信息的处理功能；

第三，随着无线技术的发展，将更加强调与无线互联网资源的整合；

第四，安全性和稳定性；

第五，与本地化应用的融合。

多人会议系统已成为企业内部人员沟通不缺少的手段，也是潜在的商机。

1.2 多人会议系统的特点

多人会议系统的特点主要包括以下几点。

1．即时通讯

加强内部沟通，提升管理的即时性，提高办公效率。

即时通讯平台是面向企业级应用的，在面向工作人员的客户端上可以统一呈现单位所用的人员信息（关键信息是可以隐藏的），这样就可以快速的找到你要找的同事，并能迅速的发起与他的对话，直接拖曳你想发送的超大文件，对方就能立刻看到你发送的消息，立刻就能提升对方接收你的文件。如果对方不在客户端上，对于文本信息，支持离线留言，还有可能提供短信的直接发送，同时支持文件传输，保证信息的即时性。

即时通信，解决内部沟通，解决了人与人之间的协作沟通，在人与人之间搭建了一座桥梁；消息提醒，在人与系统之间做了一个很好衔接，协同管理软件，提升管理软件的价值，进一步调高办公效率。

2．统一管理

可管可控的统一管理控制平台。

任何一个系统都有控制管理之说，对于面向企业级的多人会议系统来说更是如此，那如何体现系统的可靠性、可控性，又要体现出对领导的一些特殊性。

通过设置角色来控制人员的权限，员工能做什么，能做多少，都是可以通过权限角色来一一控制的。同时对于一些关键的特殊处理，比如，领导排在前面、一人可以多个职位、一些关键的信息资料（如手机号码）需要隐藏等，都可以通过特殊的权限处理以及设置来实现，这些都充分体现了可管可控性。

可管可控性，才是一个企业级办公系统真正的精髓。

3．组织结构的统一

方便快速地定位和查找人员，联系协作。

一个大型单位，分支机构太多，如何统一体现所有的人员组织信息，如何能方便地与各地方、各分支建立有效而快速的连接，进行及时沟通，构建一个虚拟却真实有效的沟通平台，提高整个工作单位的效率。

可以把单位所有的人员组织信息统一起来，并通过客户端统一展现出来，实时更新员工的通讯录，一目了然，方便员工之间的协作。也可以通过与其他系统的人员组织结构同步，实现人员组织结构的统一并同步更新。

统一的组织结构，实现了跨地球、跨分支、跨部门的人员统一。

4．会议沟通

提供多种形式的会议系统模式，满足不同会议的需求。

单位发展越来越快，部门也越来越多，人员也越来越多，大大小小的会议也就多了。显然，固定的会议室和传统的会议模式，无法满足日益增多的会议需求。比如：一个领导在北京，一个领导在上海，另外一个领导在深圳，如何实现他们之间的沟通？如何解决跨部门、跨地域的沟通？如何解决会议室沾满的问题？

提供专业级视频会议的支持，其主要功能有：强大的白板 / 文档协作、应用程序共享、网络文件管理、电子投票和表决、议程会议安排等强大的协同数据操作功能：高质量的音频效果，支持同时显示多路视频及会议过程的多种视频显示，支持会议录制功能，并具有全面的网络适应能力和安全管理机制，提供主席控制机制保证了沟通安全、有序、可控的进行，在会议中主持人与发言人等身份可在所用参与者之间任意转移。基于 WEB 的 B/S模式，自动下载并安装插件即可登陆系统，支持多服务器级联部署及服务器的整合和拆分，支持多会议室的结构。

多种会议形式，能为企业内部会议搭建一个实时通信、异地办公的协同办公的沟通平台。

1.3多人会议系统软件的现状

通过调研，分析了一些聊天软件如 QQ， MSM 等等聊天工具，他们都实现了一般的聊天功能，而且功能也不断的在完善，如聊天，支持一些特殊表情，图片粘贴等等，离线留言，还支持手机短信等等。当然视频，语音的功能也在不断的完善中。他们一般都是点对点的视频，语音，也许不久也会支持会议模式下的视频、语音聊天。

但作为企业内容会议所使用的聊天软件，它不能满足需要。必须还得有一些特殊的要求来满足企业内部会议的需要，如职员信息管理、白板共享、协同浏览等等企业内部不可缺少的会议交流手段。所以，有待改进他们，使他们更商业化，来满足企业内部的需求。

现在，一些视频会议系统逐渐走向了市场，成为商业内部会议沟通的重要手段，满足了他们内部人员沟通的要求。

基于上面的分析，开发一个商业化的视频会议系统，有很大的实用性，而且能够更好的满足他们的要求，技术也逐渐成熟，风险很小，利益不错，可以开发。虽然这仅是毕业设计，不可能开发出功能齐全的会议系统，但也能使自己对这一领域有了较深的了解。

多人会议系统软件在国内外有很多以及一些在网页上的即时通讯工具，像 Chinaren 网站上的 WebMaster 等等，都做得即美观，且功能强大，他们一般现在都拥有非常大的用户群。

在网上参加会议或聊天，都能够实现个人的需求如聊天，发文件，收 E-mail，视频，，音频交流，可以发送离线消息，不管用户当时是否在线，下次上线时，就可以看到这条消息了，可以保存用户的个人信息或介绍，供人查看等功能。总之，这种软件在网络上，还是有很大的用途的，为网络上通讯，带来极大的方便。

虽然说，现在这样软件已经有公司把它开发出来了，我再做也不一定有新意，也未必可以做得更好，但作为毕业设计，也算是对我能力的一个考验和这四年大学内我学习知识的一个检查。

1.4 面向对象方法与设计简介

传统的软件工程方法有生命周期方法和快速原型法。

面向对象方法学是一种全新的软件工程方法，其出发点和基本原则是尽可能模拟人类习惯的思维方式，把构成客观世界的实体抽象为对象。概括地说，面向对象方法学有四个要点：

- 认为客观世界是由各种对象组成的，复杂的对象可以由比较简单的对象以某种方式组合而成；
- 把所有对象都划分成各种对象类，每个对象类可以定义一组数据和方法；
- 按照子类和父类的关系，把若干对象类组成一个层次结构的系统；
- 对象彼此之间仅能通过传递消息互相联系。

用面向对象方法学开发的软件有以下优点：

- 与人类习惯的思维方法一致；
- 稳定性好；
- 可重用性好；
- 可维护性好。

2 网络通讯程序的设计原理和过程

2.1 TCP/IP 结构简介

TCP/IP 协议是一个四层协议，它的结构如图 2-1 所示。

应用层	各种应用层协议（Telnet、FTP 等）	
链路层	TCP	UDP
传输层	IP	
网络层	设备驱动程序及接口卡	

图 2-1 TCP/IP 协议族体系结构

每一层负责的功能如下。

- **链路层**：有时被称作数据链路层或网络接口层，通常包括操作系统中的设备驱动程机中对应的网络接口卡，它们一起处理与电缆（或其他任何传输媒介）的物理接口细节。该层包含的协议有：ARP(地址转换协议）和 RARP（反向地址转换协议）。
- **网络层**：有时也被称为互联网层，负责分组在网络中的活动，包括 IP 协议（网际协议）、ICMP 协议（Internet 互联网控制报文协议）以及 IGMP 协议（Internet 组管理协议）。
- **传输层**：该层主要为两台主机上的应用程序提供端到端的数据通信，它分为两个不同的协议:TCP(传输控制协议）和 UDP(用户数据报协议)。TCP 协议提供端到端的质量保证的数据传输，该层负责数据的分组、质量控制和超时重发等，对于应用层来说，就可以忽略这些工作。UDP 协议则只提供简单的把数据报从一端发送到另一端，至于数据是否到达或按时到达、数据是否损坏都必须由应用层来做。这两种协议各有各自的用途，前者一可用于面向连接的应用，而后者则在及时性服务中有着重要的用途，如网络多媒体通信等。
- **应用层**：该层负责处理实际的应用程序细节，包括大家都十分熟悉的 Telnet（电子公告板）、HTTP（World Wide Web 服务）、SMTP(简单邮件传输协议）、FTP(简单文件传输协议）和 SNMP(简单网络管理协议）

等著名协议。

图 2-2 表示路由器连接两个网络的方法。体现了 TCP/IP 的四层结构，还有各层之间的关系。

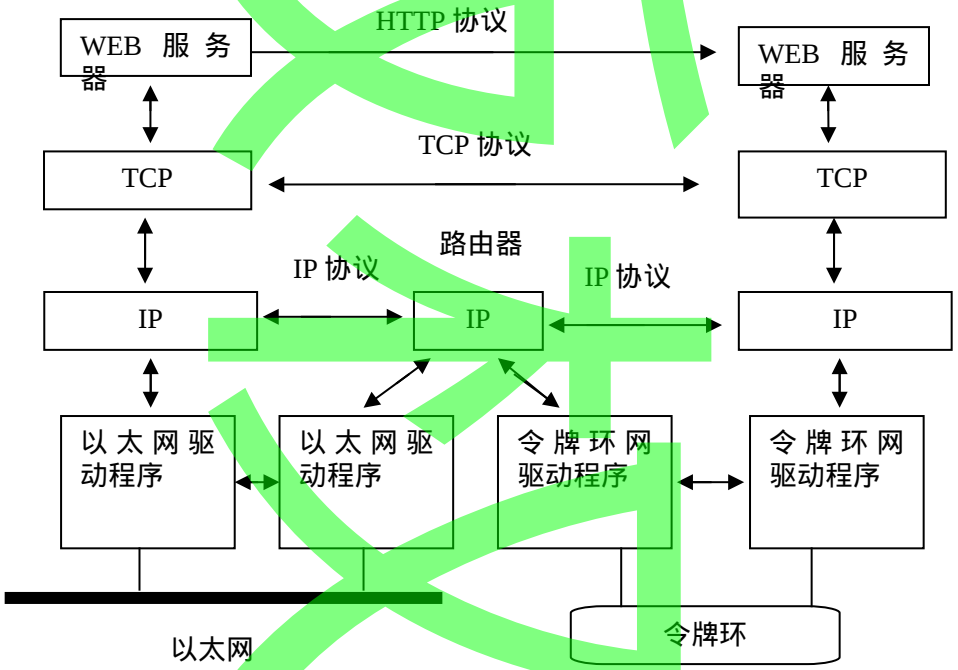


图 2-2 通过 TCP/IP 和路由器连接的两个网络

在图 2-2 所示的系统中，两个网络通过路由器互相连接。路由器可以用来把以太网、令牌环、点对点链接和 FDDI(光纤分布式数据接口) 等不同的网络连接到一起。

协议的各个层对上一层是透明的，也就是说，应用层只负责应用程序之间的通信规则完全不必理会数据是怎样在网络中传输，也不必理会它怎样接入网络。而对于链路层来改，它完全不需要知道正在传送的是什么数据，只需要负责传输就行了。

网桥、路由器和网关的差别。网桥是一种在链路层将不同类型的局域网连接成一个更大的局域网的网络设备，路由器则是在网络层实现该功能，而网关是指连接不同协议簇的进程（例如 TCP/IP 和 IBM 的 SNA)，它为某个特定的应用程序（常常

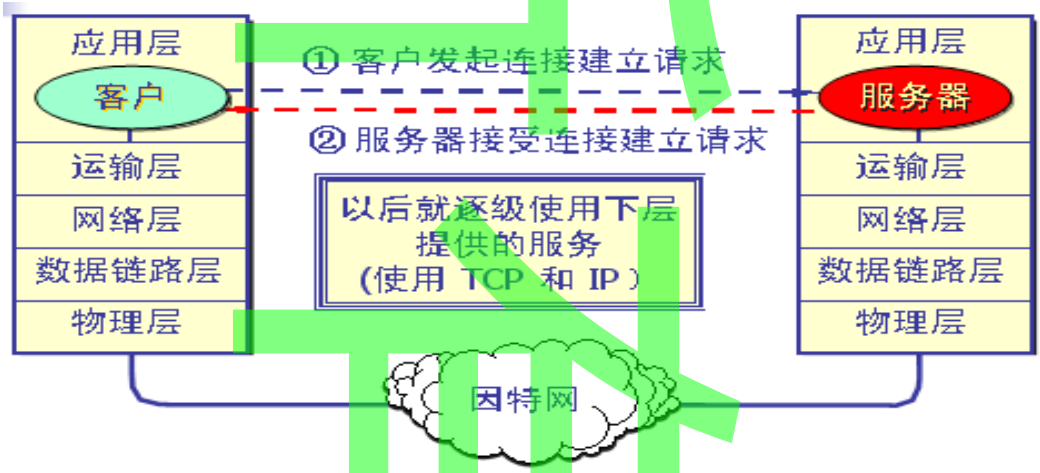


图 2-3 客户进程和服务器进程使用 TCP/IP 协议进行通信

是电子邮件或文件传输 ）服务。

图 2-3 表示客户进程和服务器进程使用 TCP/IP 协议进行通信的方法。

2.2 Winsock 网络编程接口

Winsock 是一套开放的、支持多种协议的 Windows 下网络编程接口，是 Windows 网络编程事实上的标准。应用程序通过调用 Winsock 的 API 实现相互之间的通信，而 Winsock 利用下层的网络通信协议功能和操作系统调用实现实际的通信工作。

随着编程技术的不断发展，一些技术，如面向对象编程、 ActiveX技术等都在某种程度上降低了 Winsock 网络编程的门槛，简化了编程的过程。但对 Winsock 的基本概念、编程原理的了解无论如何都是必要的，不仅因为有时程序员不得不在 API的层次上实现更灵活、更高级的编程，而且熟悉这些概念、原理对进行面向对象和 ActiveX 层次上的 Winsock 编程也大有裨益。

2.2.1 Winsock 概述

主要对 Winsock 的基本概念、编程模型等进行介绍，并给出一个简单但很能说明问题的例子。

1.**基本概念**

(1) 广播

数据报套接字可以用来向许多系统支持的网络发送广播数据包。要实现这种功能，网络本身必须支持广播功能，因为系统软件并不提供对广播功能的任何模拟。广播信息将会给网络造成极重的负担，因为它们要求网络上的每台主机都为它们服务，所以发送广播数据包的能力被限制于那些用显式标记了允许广播的套接字中。

(2) 字节顺序

不同的计算机有时使用不同的字节顺序存储数据。例如，基于 Intel 处理器的计算机和 Macintosh 计算机使用了相反的字节排序规则。 Intel 的字节排序被称为“ Little-Endian" ，它与网络的字节排序规则“ Big-Endian” 排序顺序相反。

任何从 Winsock 函数对 IP 地址和端口号的引用和传送给 Winsock 函数的 IP 地址和端口号均是按照网络顺序组织的，这也包括了 sockaddr in这一数据结构中的 IP 地址域和端口域 (但不包括 sines family 域 ）。

考虑到一个应用程序通常用与“时间”服务对应的端口来和服务器连接，而服务器提供某种机制来通知用户使用另一端口。因此getservbyname函数返回的端口号已经是网络顺序了，可以直接用来组成一个地址，而不需要进行转换。然而如果用户输入一个数，而且指定使用这一端口号，应用程序则必须在使用它建立地址以前，把它从主机顺序转换成网络顺序。相应地，如果应用程序希望显示包含于某一地址中的端口号，则这一端口号就必须在被显示前从网络顺序转换到主机顺序。

(3) 阻塞和非阻塞

套接字可以处于阻塞模式或非阻塞模式。调用任何一个阻塞模式的函数，都会产生相同的后果—耗费或长或短的时间等待操作的完成。而当套接字处于非阻塞模式时，API函数的调用立即返回，大多数情况下这些调用都会“失败”，并返回一个 WSAEWOULDBLOCK

错误，它意味着请求的操作在调用期间没有时间完成。 Winsock 的套接字 1/O 模型可以帮助应用程序判断一个套接字何时可供读写。

2. **套接字**

套接字 (Sockets)是通信的基石，是支持 TCP/IP 协议的网络通信的基本操作单元。可以将套接字看作不同主机间的进程进行双向通信的端点，它构成了在单个主机

内及整个网络间的编程界面。套接字存在于通信域中，通信域是为了处理一般的线程通过套接字通信而引进的一种抽象概念。套接字通常和同一个域中的套接字交换数据（数据交换也可能穿越域的界限，但这时一定要执行某种解释程序）。 Winsock 规范支持单一的通信域，即 Internet 域。各种进程使用这个域互相之间用 Internet 协议簇来进行通信 (Winsock 1.1 以上的版本支持其他的域，例如Winsock 2).

套接字可以根据通信性质分类，这种性质对于用户是可见的。应用程序一般仅在同一类的套接字间通信。不过只要底层的通信协议允许，不同类型的套接字间也照样可以通信。

套接字有两种不同的类型 ：流套接字和数据报套接字。

(1) 流套接字

流套接字提供双向的、有序的、无重复并且无记录边界的数据流服务，它适用于处理大量数据。网络传输层可以将数据分散或集中到合适尺寸的数据包中。

流套接字是面向连接的，通信双方进行数据交换之前，必须建立一条路径，这样既确定了它们之间存在的路由，又保证了双方都是活动的、可彼此响应的，但在通信双方之间建立一个通信信道需要很多开支。除此以外，大部分面向连接的协议为保证发送无误，可能会需要执行额外的计算来验证正确性，因此会进一步增加开支。

(2) 数据报套接字

数据报套接字支持双向的数据流，但并不保证数据传输的可靠性、有序性和无重复性。也就是说，一个从数据报套接字接收信息的进程有可能发现信息重复，或者和发出时的顺序不同的情况。此外，数据报套接字的一个重要特点是它保留了记录边界。数据报套接字是无连接的，它不保证接收端是否正在侦听，类似于邮政服务 ：发信人把信装入邮箱即可，至于收信人是否想收到这封信或邮局是否会因为暴风雨未能按时将信件投递到收信人处等等，发信人都不得而知。因此，数据报并不十分可靠，需有程序员负责管理数据报的排序和可靠性。

2.2.2 Winsock 编程技术

1. 简单客户机和服务器模型

进入 20 世纪 90 年代后，随着计算机和网络技术的发展，很多数据处理系统都采用开放系统结构的客户枷服务器 (Client/Server) 网络模型，即客户机向服务器提出请求，服务器对请求做相应的处理并执行被请求的任务，然后将结果返回给客户机。这种方式隐含了在建立客户枷服务器间通信时的非对称性。

客户机 / 服务器模型工作时要求有一套为客户机和服务器所共识的惯例来保证服务能够被提供（或被接受），这一套惯例包含了一套协议，它必须在通信的两头都被实现。根据不同的实际情况，协议可能是对称的或是非对称的。在对称的协议中，每一方都有可能扮演主从角色 ；在非对称协议中，一方被不可改变地认为是主机，而另一方则是从机。一个对称协议的例子是 Internet 中用于终端仿真的 Telnet ，而非对称协议的例子是 Internet 中的 HTTP。无论具体的协议是对称的或是非对称的，当服务被提供时必然存在客户进程和服务进程。

一个服务程序通常在一个众所周知的地址监听客户对服务的请求，也就是说，服务进程一直处于休眠状态，直到一个客户对这个服务的地址提出了连接请求。在这个时刻，服务程序被“惊醒”并且为客户提供服务—对客户的请求作出适当的反应。这一请求 / 响应的过程可以简单地用图 2-4 表示。虽然基于连接的服务是设计客户枷服务器应用程序时的标准，但有些服务也可以通过数据报套接字提供。

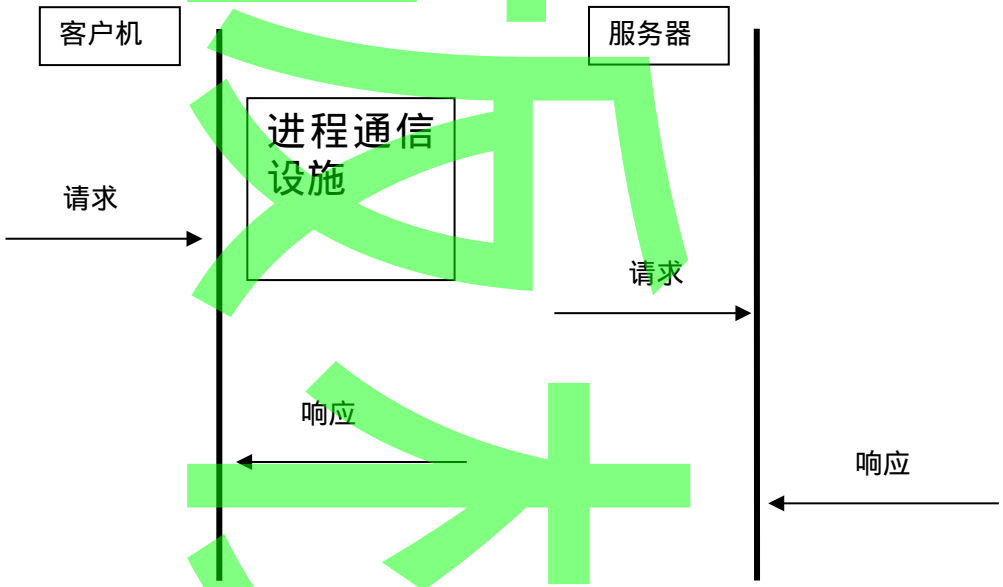


图 2-4 客户机与服务器模型

2. Winsock的启动和终止

由于 Winsock 的服务是以动态链接库 Winsock DLL 形式实现的，所以必须先调用 WSAStartup 函数对 Winsock DLL 进行初始化，协商 Winsock 的版本支持，并分配必要的资源。如果在调用 Winsock 函数之前，没有加载 Winsock 库，则会返回 SOCKETee ERROR 错误，错误信息是 WSANOTINITIALISED. WSAStartup 函数的原型如下 ：

int WSAStartup(WORD wVersionRequested, LPWSADATA 1pWSAData);

其中，参数 wVersionRequested 用于指定准备加载的 Winsock 库的版本 ：通常的做法是高位字节指定所需要的 Winsock 库的副版本，而低位字节则是主版本，然后，用宏 MAKEWORD(X,Y)(X 是高位字节， Y 是低位字节) 获得 wVersionRequested 的正确值。 1pWSAData参数是指向 LPWSADATA 结构的指针，该结构包含了加载的库版本有关的信息。

在应用程序关闭套接字后，还应调用 WSACleanup 函数终止对 Winsock DLL 的使用，并释放资源，以备下一次使用。 WSACleanup 函数的原型如下：

int WSACleanup (void);

该函数不带任何参数，若调用成功则返回 0，否则返回错误。

3. 错误的检查和控制

错误检查和控制对于编写成功的 Winsock 应用程序是至关重要的。事实上，对 Winsock API 函数来说，返回错误是非常常见的，但是多数情况下，这些错误都是无关紧要的，通信仍可在套接字上进行。尽管返回的值并非一成不变，但不成功的Winsock 调用返回的最常见的值是 SOCKET_ ERROR. SOCKETes ERROR 是值为一 1 的常量。如果错误情况发生了，就可用 W SAGetLastError函数来获得一段代码，这段代码明确地表明产生错误的原因。该函数的原型为：

int WSAGetLastError(void);

WSAGetLastError函数返回的错误都是预声明常量值，根据 Winsock 版本的不

同，这些值的声明不在 Winsock 1.h 中，就会在 Winsock 2.h 中。两个头字段的唯一差别是 Winsock 2.h 中包含的错误代码（针对 Winsock 2 中引入的一些新的 API 函数和性能）更多。为各种错误代码声明的常量一般都以 WSAE开头。

4. Winsock编程模型

不论是流套接字还是数据报套接字编程，一般都采用客户机 / 服务器方式，它们的运作过程基本类似，下面着重介绍流套接字的编程模型。

(1) 流套接字编程模型。流套接字的服务进程和客户进程在通信前必须创建各自的套接字并建立连接，然后才能对相应的套接字进行“读”、“写”操作，实现数据的传输。具体编程步骤如下：

① 服务器进程创建套接字。服务进程总是先于客户进程启动，服务进程首先调用 socket 函数创建一个流套接字。

socket 函数的原型如下：

SOCKET socket(int af, int type, int protocol);

其中，参数 of 用于指定网络地址类型，一般取AF INET，表示该套接字在 Internet 域中进行通信。参数 type 用于指定套接字类型，若取 SOCK STREAM 表示要创建的套接字是流套接字，而取 SOCK DGRAM 创建的是数据报套接字，这里取SOCK STREAM。参数 protocol 用于指定网络协议，一般取 0，表示默认为 TCP/IP 协议。若套接字创建成功则该函数返回所创建的套接字句柄 SOCKET，否则产生 INVALID_ SOCKET 错误。

② 将本地地址绑定到所创建的套接字上以使在网络上标识该套接字。这个过程是通过调用 bind 函数来完成的，该函数原型如下：

int bind(SOCKET s, const struct sockaddr* name, int namelen);

其中，第一个参数 s 标识一未捆绑套接字的句柄，它用来等待客户机的连接。第二个参数 name是赋予套接字的地址，它由 struct sockaddr结构表示，该结构的格式如下：

```
struct sockaddr
{
  u_short  sa_family;
  Char   sa_data[14];
  };
```

该地址结构随选择的协议的不同而变化，因此一般情况下另一个与该地址结构大小相同的 sockaddr_in 结构更为常用，sockaddr_in 结构用来标识 TCP/IP 协议下的地址，在 TCP/IP 协议的情况下，可以方便地通过强制类型转换把 sockaddr in 结构转换为 sockaddr 结构。其中，sin_family 字段必须设为AF INET，表示该 socket 处于 Internet 域。 sin-port 字段用于指定服务端口，应把端口号由主机字节顺序转换为网络字节顺序，如果端口号置为 0，则 Winsock 将为应用程序分配一个值在 1024 到 5000之间的唯一的端口。

一旦出错，bind 函数就会返回 SOCKETes ERROR。对 bind 来说，最常见的错误是 WSAEADDRINUSE。如果使用的是 TCP/IP，那么该错误表示另一个进程已经同本地 IP 接口和端口号绑定到了一起，或者那个 IP 接口和端口号处于 TIMEes WAIT 状态。假如对一个已经绑定的套接字调用 bind，便会返回 WSAEFFAULT 错误。

③ 将套接字置入监听模式并准备接受连接请求。 bind 函数的作用只是将一个套接字和一个指定的地址关联在一起，让一个套接字等候进入连接的 API函数则是 listen，其原型为：

int listen(SOCKET s, int backlog);

其中，参数 s 标识一个已捆绑未连接套接字的描述字。

如无错误发生，listen 函数返回 0，若失败则返回 SOCKET_ ERROR 错误，最常见的错误是 WSAEINVAL，该错误通常表示套接字在 listen 之前没有调用 bind.

进入监听状态之后，通过调用 accept 函数使套接字作好接受客户连接的准备。accept 函数的原型为：

SOCKET accept(SOCKET s, struct sockaddr* addr, int* addrlen);

其中，参数 s 是处于监听模式的套接字描述字。第一个参数应该是一个有效的 SOCKADDR IN 结构的地址，而addrlen是 SOCKADDR 一 IN 结构的长度。 accept 函数返回后， addr 参数变量中会包含发出连接请求的那个客户机的 IP 地址信息，并返回一个新的套接字描述字，它对应于已经接受的那个客户机连接。

④ 客户进程调用 socket 函数创建客户端套接字。

⑤ 客户向服务进程发出连接请求。通过调用 connect 函数可以建立一个端的连接。

connect 函数原型为：

int connect(SOCKET s, const struct sockaddr FAR *name, int namelen);

该函数的参数是相当清楚的 :s标识一个未连接的数据报或流类套接字的描述字。name是针对 TCP的套接字地址结构，它标识服务进程 IP 地址信息，若 name结构中的地址域为全零的话，则 connect 函数将返回 WSAEADDRNOTAVAIL 错误。namelen 则用于标识 name参数的长度。

⑥当连接请求到来后，被阻塞服务进程的 accpetQ 函数如③中所述生成一个新的套接字与客户套接字建立连接飞并向客户返回接收信号。

⑦一旦客户机的套接字收到来自服务器的接收信号，则表示客户机与服务器已实现连接，则可以进行数据传输了。 send, recv 函数是进行数据收发的函数。

send 函数的原型为：

int send(SOCKET s, const char* buf, int len, int flags);

其中， SOCKET参数是已建立连接的套接字描述字，表示发送数据操作将在这个套接字上进行。Send 函数返回发送数据的字节数，若发生错误，就返回 SOCKET_ ERROR 。常见的错误是WSAECONNABORTED，这一错误一般发生在虚拟回路由于超时或协议有错而中断的时候。

所有关系到收发数据的缓冲都属于简单的 char 类型，也就是说，这些函数没有“ Unicode” 版本。使用 Unicode 时需要把字符串当作 char*或把它转换为 char* 发送，在利用字符串长度函数告诉 Winsock API 函数收发的数据有多少字符时必须将这个值乘以 2，因为每个字符占用字串组的两个字节。

recv 函数的原型为：

int recv(SOCKET s, char* buf, int len, int flags);

其中，参数 s 是准备接收数据的套接字。第二个参数 buf是即将收到数据的字符缓冲区，而 len 则是准备接收的字节数或 buf缓冲的长度。

⑧ 关闭套接字。一旦任务完成，就必须关掉连接以释放套接字占用的所有资源。

图 2-5 列出了流套接字编程的流程图。

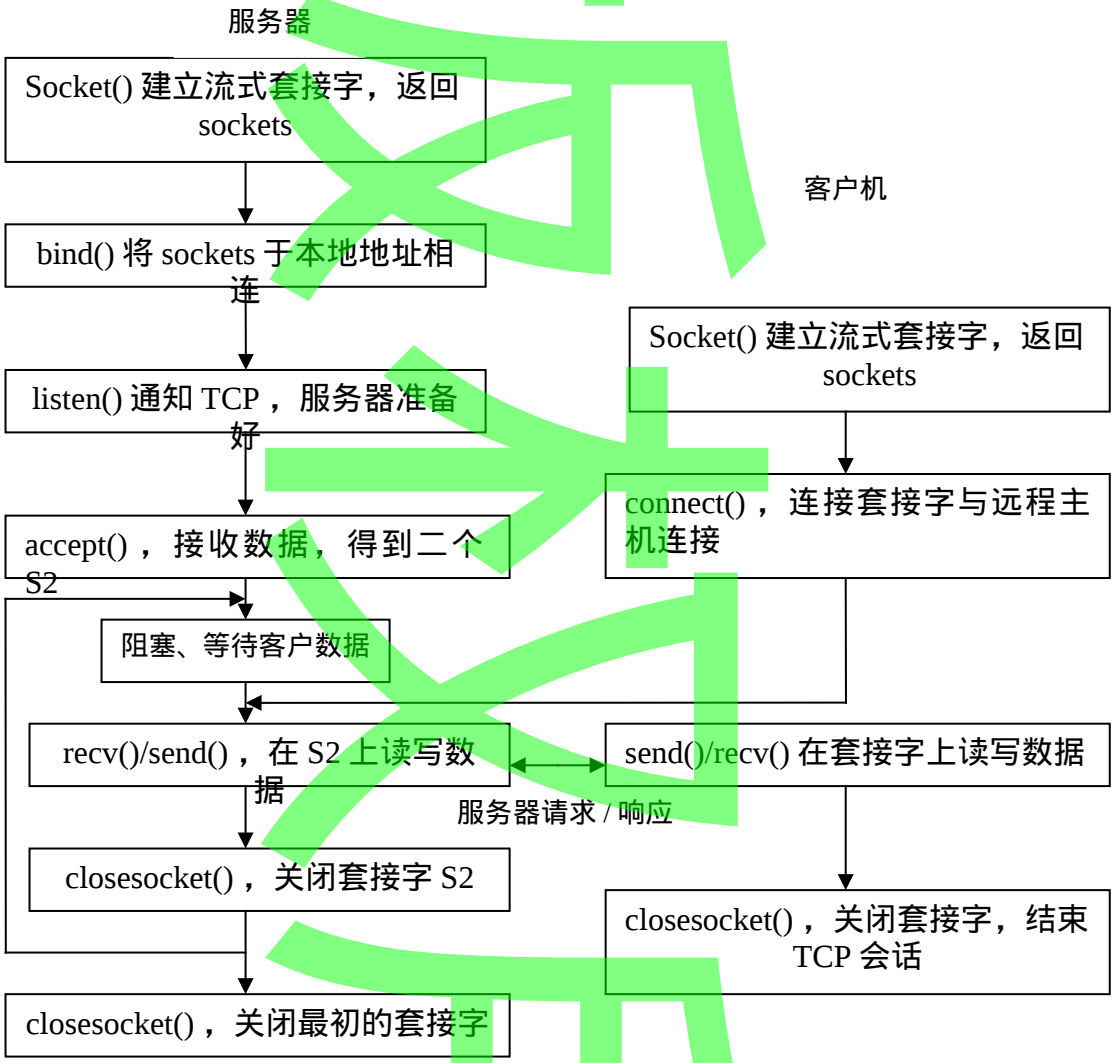


图 2-5 流套接字编程的流程图

(2) 数据报套接字编程模型

数据报套接字是无连接的，它的编程过程比流套接字要简单一些。

对于接收端（一般为服务端），先用 socket函数建立套接字，再通过 bind 函数把这个套接字和准备接收数据的 IP 地址信息绑定在一起，这和前面流套接字一样，但不同的是它不必调用 listen 和 accept函数，只需等待接收数据。并且由于它是无连接的，因此它可以接收网络上任何一台机器所发的数据报。常用的接收数据函数是recvfrom，它的原型为：

```
int recvfrom( SOCKET s, char* buf, int len, int flags,
    struct sockaddr* from, int* fronden);
```

其中，前面的四个参数和 recv 函数一样，而参数 from 是一个 SOCKADDR 结构指针， fromlen 参数是带有指向地址结构的长度的指针。当它返回数据时， SOCKADDR 结构内便填入发送数据端的的地址。

对于发送端的数据报套接字来说，可以有两种方法发送数据。第一种，也是最简单的一种，便是建立一个套接字，然后调用 sendto 函数。 sendto 函数的原型为：

```
int sendto( SOCKET s, const char* buf, int len, int flags,
    struct sockaddr* from, int* fronden)；
```

除了 buf 是发送数据的缓冲， len 指明发送的字节数外，其余参数和 recvfrom的参数一样， to参数是一个指向 SOCKADDR 结构指针，带有接收数据的套接字目标地址信息。数据报套接字编程流程如图 2-7 所示。

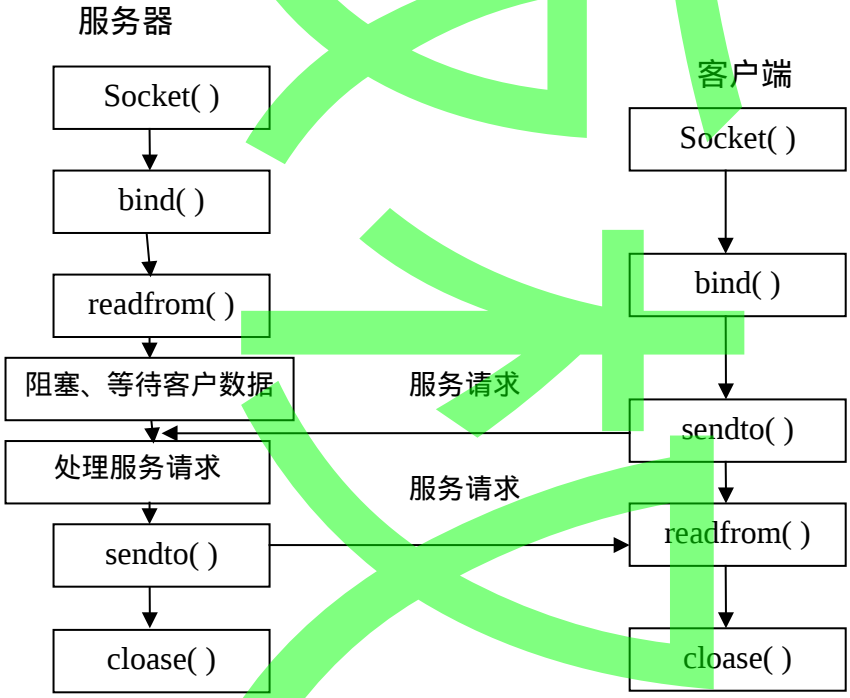


图 2-7 无连接套接字流程图

2.3 MFC Windows Sockets 类介绍

MFC 提供了两个 Windows Sockets封装类 ：CAsyncSocket 和 CSocket 类，它们使编程工作变得相对简单， CAsyncSocket 类是对 Winsock API的简单封装，其成员函数都带有 Winsock API 的影子，并且编程模型也大致相同。CSocket类则封装了大量实现细节，使得编程效率大大提高。

CSocket类是 CAsyncSocket 类的派生类，它提供了对通过 CArchive 对象使用套接字工作的更高级抽象。 CSocket类的使用比 CAsyncSocket 类更加容易，它继承了 CAsyncSocket 类的许多封装了API的成员函数，并且管理了通信的大多数方面，这使用户从原来不得不使用原始 API或者 CAsyncSocket 类的繁杂工作中解脱出来。更加重要的是 CSocket类提供了对于同步操作 CArchive 对象十分重要的阻塞功能，且 CSocket通过与类 CSocketFile和 CArchive 一起来管理对数据的发送和接收，使收发数据的操作变得简单明了。

1. CSocket对象与串行化技术

共同使用 CSocket与 CArchive 对象是最简单的套接字编程模型，其中 CArchive 对象将帮助程序员处理许多以前必须使用 API或 CAsyncSocket 类来处理的通信问题，大大减少了编程的工作量。如果与非 MFC 服务器进行通信，就不能使用 CArchive 对象，因为这种服务器并不能处理这种类型的对象，这时就不得不使用字节排序的套接字编程方法。本系统的实现采用串行化技术。

对于套接字来说，归档对象是与 CSocketFile 对象而不是与标准 CFile对象相关的，与连接到一个磁盘文件不同， CSocketFile 对象连接到一个 CSocket 对象。一个 CArchive 对象将负责管理一个缓冲区。当 storing(发送) 归档对象的缓冲

区被填满时，相关的 CSocketfile 对象将缓冲区的内容取出，清空与套接字相关的归档缓冲区。当 loading(接收) 归档对象的缓冲区被填满时， CSocketFile 对象停止读出直到缓冲区可用。

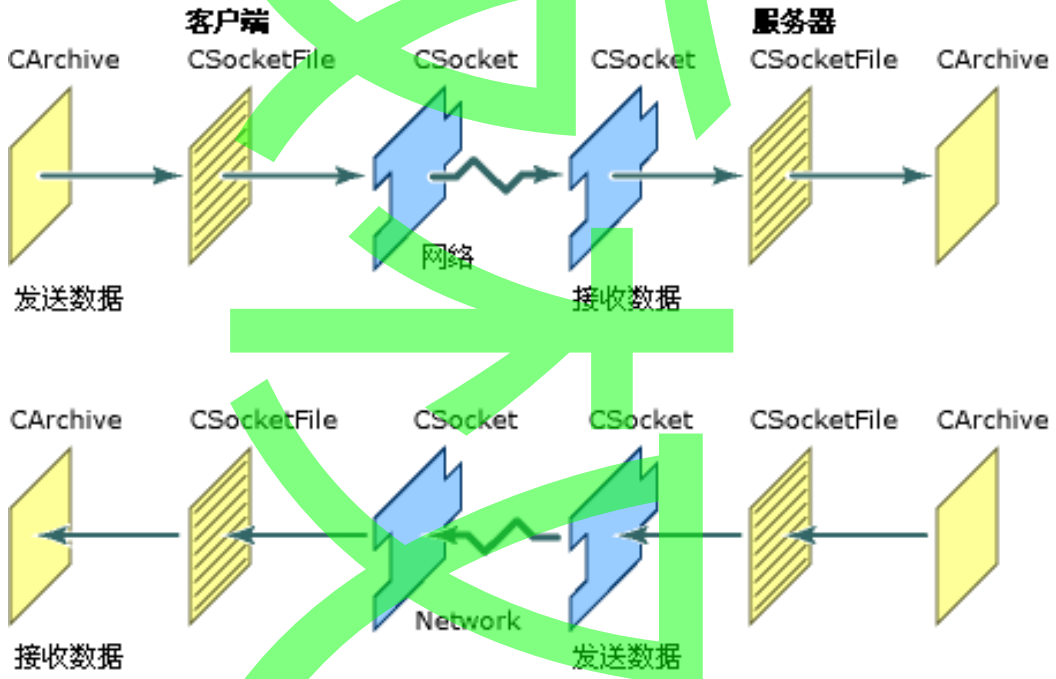


图 2-8 CSocket, CsocketFile和 Carchive对象之间的关系

尽管 CSocketFile 类是 CFile 的派生类，但它并不支持 CFile 的一些成员函数，例如 Seek, GetLengh, SetLengh, LockRange, UnlockRange 以及 GetPosition 等函数，这还是比较容易理解的，因为 CSocketFile 是从相关的 CSocket对象中读出或写入字节序列，与文件无关，诸如 Seek 或 GetPosition 等函数并无意义。为了禁止 CSocketFile 所不支持的 CFile 成员函数在 CSocketFile 类中被重载，在执行重载这些函数时，就会触发一个 CNotSupportedException 异常。 CSocket, CSocketFile 和 CArchive 对象之间的关系如图 2-8 所示。 CSocket对象实际上是一个两态对象，有时异步 (通常的状态) 有时同步。在异步态，套接字能够接收来自框架的异步通告。但在操作过程中，例如接收或发送数据时，套接字就切换到同步状态。这意味着，套接字将不能接收其他异步通信，直到同步操作完成。

2. CSocket类编程模型

下面是使用 CSocket对象进行客户和服务器之间通信的一般编程模型，它只适用于数据流套接字，因为数据报套接字不能使用 CArchive 。

- 分别构造服务器和客户套接字对象。
- 调用对象的 Create函数创建套接字，而 Create函数会调用 Bind 函数将此套接字绑定到指定的地址。需要注意的是为服务器创建套接字时需要为其指定端口号。
- 套接字创建完毕后，服务器调用 Listen 成员函数开始侦听客户的连接请

求，而客户可以调用 Connect成员函数向服务器请求连接。

- 当服务器监听到客户连接请求时，创建一个新的套接字，并将其传送给Accept成员函数以接收客户的连接请求，函数执行失败会返回特定的错误码。
- 为服务器和客户的套接字对象分别创建一个与之相联系的 CSocketFile对象。
- 为服务器和客户的套接字对象分别创建一个与 CSocketFile 相联系的 CArchive 对象以进行数据的发送和接收。
- 使用 CArchive 对象在客户和服务器套接字之间传送数据。
- 在任务执行完成后，销毁 CArchive, CSocketFile 和 CSocket 对象。

3 需求分析

3.1 概述

1. 网络会议系统的意义

随着计算机技术和通信技术的飞速发展，计算机支持的协同工作(CSCW)应运而生，并在近年来得到了巨大的发展。 CSCW 系统充分利用计算机络资源和通信技术，让分散于不同空间的用户在相对较短时间间隙中协同工作完成同一任务。在 CSCW 系统中，用需要运用多种会商工具进行语音、文字、图形和图像等的流，进行密切配合、协同工作，电子白板是一个十分重要协同工作工具。通过使用鼠标、键盘、手写笔和触摸屏硬件 I/O 设备，各协作用户可以在电子白板上面绘制图形入文本、注释、剪切、复制图片，并同步显示在其他用户白板界面上，从而达到资源共享、实时交流的目的。由于子白板交流具有灵活、方便、及时的应用特点，因此被广地集成在视频会议、远程教学、自动化办公、 ERP 软件中，有很高的实用价值。在当今流行的视频会议系统中也有电子白板的影子。

2．一般其主要应有的功能描述

- 信息共享。信息共享是电子白板系统的基本任务，它要求电子白板应用系统为各协作成员提供方便可靠的信息收发、修改和删除机制，提供运行在不同操作平台上的不同应用程序对数据的访问和交换。具体地说就是提供信息共享的不同访问方式，根据用户的身份，提供对数据的不同的访问权限等。

- 多媒体群组通信。电子白板系统提供了支持在协作成员之间互换多媒体信息的通信机制，这些媒体包括文本、图形、图像等。其次，提供群组通信支持，包括异步组通信和同步组通信，它使通信服务具备多种数据交换方式，即点到点、点到多点、多点到一点和多点到多点等。这意味着，协作的用户可作为数据的发方或收方，又可以同时具备收 / 发的功能。

- 个体活动管理。电子白板系统为各协作用户提供宽松的 WYSIWIS(What You See Is What I See, 你见即我见) 机制，允许参加者对同一事务的不同部分以不同形式进行观看和修改；同时提供安全机制，对公用操作 / 数据和私有操作 / 数据进行区分，为参加协调工作的用户保留一部分私有数据不为群体共享。

- 群体协作管理。电子白板系统支持多个用户参与同一工作，它提供给各协作用户一个公共平台，每一个协作用户在它的协调下完成一项共同的工作，它负责对活动的步骤加以协调，其中包括工作流支持系统、群组方法支持工具、群组工作程序协调系统和群组决策支持系统，也包括群体活动中成员间任务和责任的划分；同时在协调中采用协调控制策略，如令牌控制方式、并发控制和协商控制等，以避免个体之间的冲突。

3． 实现关键技术

在电子白板系统中，影响协同工作不能正常进行的主要因素有 2 个：一个是本地用户和远程协作者的操作数据在本地白板上执行产生的冲突；另一个是不同的远程协作者发出的操作数据在本地白板上执行产生的冲突。协同用户对白板资源的共享冲

突呼吁一种有效的并发策略来保证协同工作的可靠性。此外，对于协作者操作数据的传输需要设计一种应用层协议，使之能传输不同类型的数据，如文本、图形、图片等。因此，如何设计系统的并发策略和通信协议成为关键。

3.2 多人会议系统的性能要求

- 统处理的准确性和及时性
- 系统的开放性和系统的可扩充性
- 系统的易用性和易维护性
- 系统的标准性
- 系统的先进性
- 系统的响应速度

3.3 易用性需求

系统操作简单，界面人性化，能为普通用户迅速掌握使用方法，快速的使用本系统所提供简单明了的管理功能，从而使得系统能被广大用户接受，扩大使用人群面积。

3.4 多人会议系统的功能需求

此软件主要是模仿多人视频会议系统的实现，来实现企业内部多人会议的功能。主要功能是支持聊天，电子白板共享和在线用户列表的管理。由于时间和技术方面的原因，一个人也不可能实现商业化软件的很多功能。

- 多人聊天
  即大家所熟悉的 QQ 群聊天方式，一个人发的信息，会议中的所有人都可以看到信息。
- 私人聊天
  该功能是由客户端决定是否只接受要好和自己聊天对象的信息。和一般的私聊有区别。
- 在线用户列表显示
  服务器端和客户端都可以看到会议中在线人数列表。
-

简易电子白板功能

主要实现了类似于绘图程序的绘图功能，可以绘制直线、连续直线、任意曲线、圆和圆形区域。直线的线条可以改变粗细和颜色。圆形区域可以选择几种填充类型。

4 概要设计和详细设计

4.1 编写目的

在本系统项目的前一阶段，也就是需求分析阶段中，已经将系统用户对本系统的需求做了详细的阐述，这些用户需求已经在上一阶段中对实地调研中获得，并在需求规格说明书中得到详尽得叙述及阐明。

本阶段已在系统的需求分析的基础上，对系统做概要设计。主要解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计等。在以下的概要设计报告中将对在本阶段中对系统所做的所有概要设计进行详细的说明。

阶段的详细设计中，程序设计员可参考此概要设计报告，在概要设计对系统预定系统所做的模块结构设计的基础上，对系统进行详细设计。在以后的软件测试以及软件维护阶段也可参考此说明书，以便于了解在概要设计过程中所完成的各模块设计结构，或在修改时找出在本阶段设计的不足或错误。

4.2 总体设计

4.2.1 客户端

客户端的主要功能模块如图 4-1 所示。

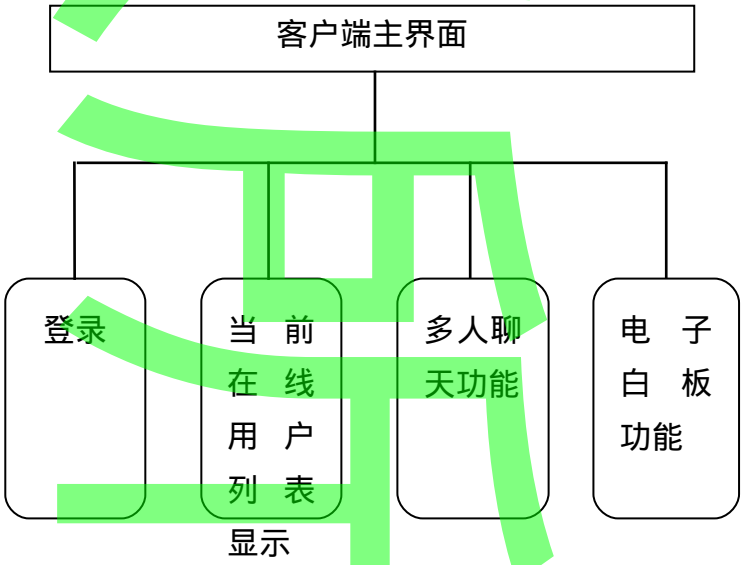
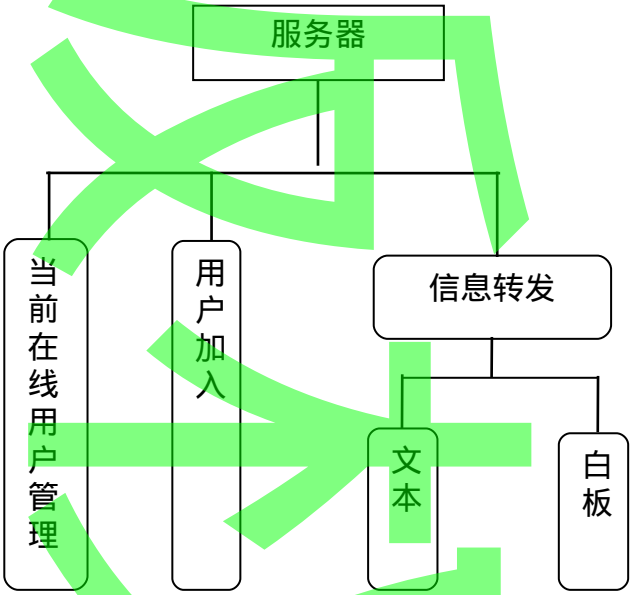


图 4-1 客户端的主要功能模块

4.2.2 服务器端

服务器的主要功能模块如图 4-2 所示。



4-2 服务器的主要功能模块

4.3 详细设计

4.3.1 客户端设计

1. 消息类的封装

由前面的知识可知， CSocket 类的串行化技术使得发送和接收网络数据就如普通的数据串行化一样简单，因此封装一个可以串行化的消息类是必要的，消息的发送和接收只需使用流操作符对缓冲区进行存取就可以了。并且这样做可以使程序具有很好的扩张性，例如如果应用程序要在原来的基础上增加一些消息，那么我们要做的只是把相应的消息项加入类中，再在服务器和客户端进行相应的解释即可。为了支持消息类的串行化，需继承 COject 类，重写相关串行化函数。消息类 CMessage的定义如图 4-3所示。

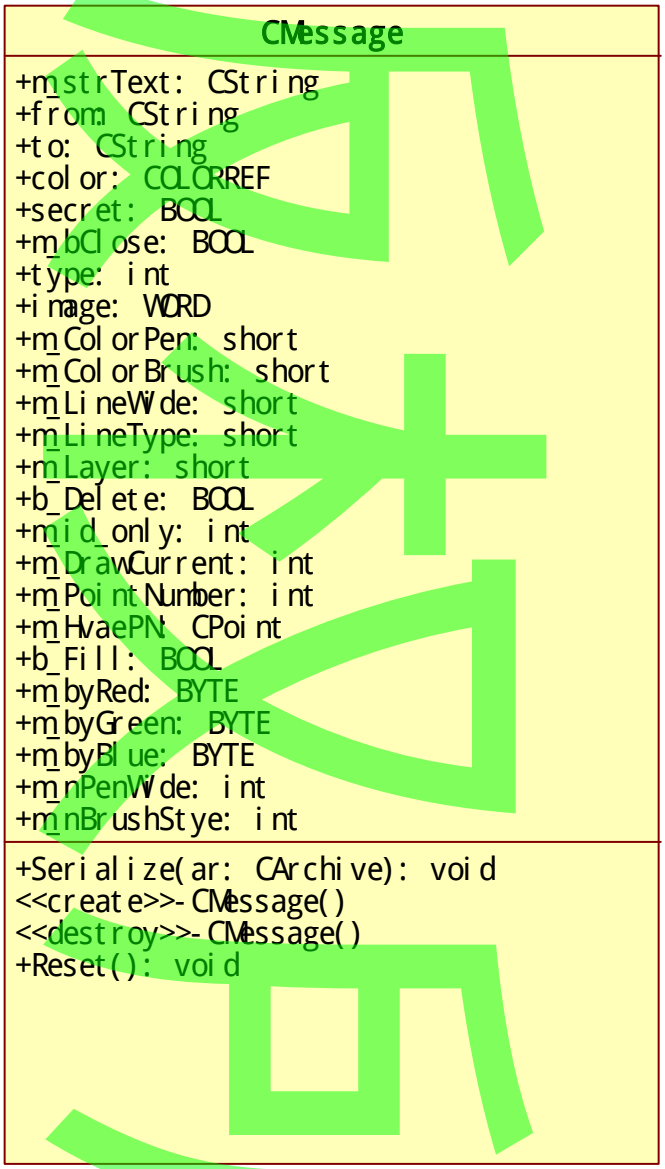


图 4-3 消息类 CMessage的定义

其中， m _strText 用于存放客户向服务器发送的信息 ;from 为消息发送方的名称； to为消息的接收方名称 ;color 为字体颜色 ;secert用于标识消息是否为“悄悄话”; m_bClose 用于表示客户是否结束运行 ;type 用于标识动作；而 Reset 成员函数用于使消息采用默认的设置 ；以上这些属性是文字交流的信息。下面介绍电子白板模块的一些属性： m_ColorPen 表示画笔的颜色； m_ColorBrush 表示画刷的填充颜色； m_LineWide表示线宽 （像素 ）； m_LineType 表示绘图的线型； m_DrawCurrent 表示用于区别传输的是哪类线条； m_PointNumber 表示一次传输几个点； m_HvaePN[100]表示假定一次传输的点不超过 100，用来保存传输的点的坐标； b_Fill 表示是否填充， 1-- 多边形区域， 0-- 连续直线或任意曲线条； m_byRed 表示当前绘图对象的颜色的一个分量 ，即画笔或画刷的颜色； m_nPenWide 表示画笔粗细； m_nBrushStye表示画刷的类型。

2． 客户端套接字类的设计

为了使客户端处于异步模式，需要从 CSocket 类派生一个自己的 WinSock 类 CChatSocket 类，它与基类唯一的不同点是重载了 OnReceive 函数，它用于当套接字接收到数据时对消息进行处理。该类的定义如图 4-4。

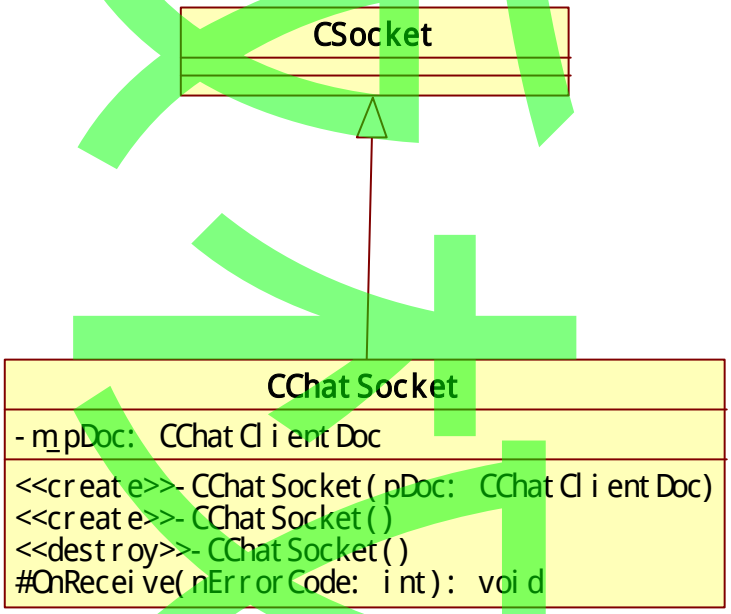


图 4-4 CChatSocket 类的定义

其中，由于主要的数据处理工作都在文档类中进行，因此类中的 m_pDoc 为 CChatClientDoc 对象指针，使套接字与文档联系起来，在后面你将看到它们是如何协调工作的。而聊天客户只需返回从服务器返回的聊天内容，因此只需重载 OnReceive 函数以通知程序“收到消息，请进行处理”，而发送消息的动作则直接通过套接字进行，并不需要用户界面响应。

3. 处理套接字通信的类

在 MFC 应用程序中，文档主要负责处理数据，而视图用于显示数据。对于客户端来说，它所处理的数据就是聊天信息，亦即服务器返回的数据和发送给服务器的数据。在本应用程序中，文档类CChatClientDoc负责数据的接收和发送，以及套接字通信的初始化。 CChatClientDoc类的定义中，成员变量 m strHandle 代表用户所用的聊天称呼，该变量由用户指定 ;m_pSocket 代表客户套接字对象 ;m_pFile 代表套接字文件对象 ;m_pArchiveIn(接收) 和 m_pArchiveOut(发送) 分别代表套接字归档对象 ;mee bFileter 用于标识是否屏蔽消息 ;m bConnected 用于标识套接字是否已经连接。

下面主要介绍一下与通信相关的几个函数。

ProcessPendingRead() 函数前面介绍了，先介绍此函数所调用的函数 ReceiveMsg() , 此函数主要是读取收到的通信信息，根据信息协议类型的不同，做出不同的处理。其实现代码如下：

```
//接收数据的工作
 void CChatClientDoc::ReceiveMsg()
{
   TRY
   {
```

```
            msg.Serialize(*m_pArchiveIn);
            DisplayRecMsg(msg.type, msg.from, msg.to, msg.secret,
                    msg.m_strText, msg.color);
            if(msg.type == -7 || msg.type == -2 || msg.type == -9)
                m_bConnected = FALSE;
            if(msg.type == -9)
                DisplayMsg(" 该用户名已经有人使用，请更名重新登录 !\n");
        }
        CATCH(CFileException, e)
        {
            msg.m_bClose = TRUE;
            m_pArchiveOut->Abort();

            CString strTemp;
            if (strTemp.LoadString(IDS_SERVERRESET))
                DisplayMsg(strTemp);
            if (strTemp.LoadString(IDS_CONNECTIONCLOSED))
                DisplayMsg(strTemp);
        }
        END_CATCH
          if(msg.m_bClose && (msg.from == m_strHandle)){
            SAFEDELETE(m_pArchiveIn);
            SAFEDELETE(m_pArchiveOut);
            SAFEDELETE(m_pFile);
            SAFEDELETE(m_pSocket);
            m_bConnected = FALSE;
        }
        if((msg.type == -1 && msg.from != m_strHandle) || (msg.type == -8))
        {
            GetView()->GetParent()->SendMessage(WM_ADDLIST,
                                                (LPARAM)&(msg.from), msg.image);
        }
        if((msg.type == -7) || (msg.type == -2)){
            GetView()->GetParent()->SendMessage(WM_ADDLIST + 1,
                                                (LPARAM)&(msg.from), msg.image);
        }
    }
```

从实现代码来看，消息的接收是非常的简单的。只需简单的调用 CMsg 对象的 Serialize函数即可， CMsg对象是之前提到的类 CMessage实现的。同样的，信息的发送也是非常的简单，原理是一样的，其实现代码如下：

```
    void CChatClientDoc::SendMsg()
    {
```

```
msg.from = m_strHandle;
if (m_pArchiveOut != NULL)
{
    TRY
    {
        msg.Serialize(*m_pArchiveOut);
        m_pArchiveOut->Flush();
    }
    CATCH(CFileException, e)
    {
        m_pArchiveOut->Abort();
        delete m_pArchiveOut;
        m_pArchiveOut = NULL;
        CString strTemp;
        if (strTemp.LoadString(IDS_SERVERRESET))
            DisplayMsg(strTemp);
    }
    END_CATCH
  }
}
```

从以上代码可看出，消息的发送也非常简单，只需简单地调用 CMsg 对象的 Serialize函数即可，聊天信息的显示代码请参考源代码，下面阐述电子白板的实现。

4. 简易电子白板设计

在 windows应用程序中，绘图是非常重要的操作。虽然我们可以用已经创建好了的位图来显示图形，但是在许多情况下仍然希望应用程序能够对视或控件绘图，这需要了解 windows的绘图机制。任何程序需要直接在屏幕或是打印机上绘图时，都需要使用图形设备接口。图形设备接口 (Graphics Device Interface) 是指这样一个可执行程序：它处理来自 windows应用程序的图形函数调用，然后把这些调用传递合适的设备驱动程序，由设备驱动程序来执行与硬件相关的函数，并产生最后的输出结果。经常同图形设备接口相提并论的另一个概念是设备上下文 (Device Contex ,DC)。设备上下文是一种 windows数据结构，它包括了与一个设备 (如显示器或打印机) 的绘制属性相关的信息。所有的绘制操作通过一个设备上下文对象进行，该对象封装了实现绘制线条、形状和文本的 Windows API 函数。设备上下文可以用来向屏幕、打印机和图元文件输出结果。

在本简易电子模块程序中，将实现一般绘图应用程序所具有的绝大多数功能，并提
供一些额外的辅助功能 (即网络通信方面)。具体而言，其实现的功能主要包括如下方面。

(1) 图元绘制

- 直线图元，或者称为线段图元，允许用户使用鼠标选锌直线顶点绘制。
- 曲线图元，允许用户从一点开始连续绘出所有点组成的曲线段。

●

矩形图元，使用鼠标控制矩形的位置和外形。
绘图操作都能够被撤销或恢复。
(2) 绘制条件设置
在绘制图元时，还能够改变绘制条件。

●

改变线条颜色，使用户可以绘制出不同颜色的图元。

●

改变线条类型，使用户可以绘制出不同线型的图元。
各种图形绘制的原理和技术阐述如下。
(1) 绘制直线

要在白板上绘制一条直线，并传输到对等方，需要建立消息数据结构，包含的数据项见表 4-1。

表 4-1 直线消息数据结构

m_DrawCurrent	m_nPenWide	m_clorDraw	mPointOrign	point
当前绘制直线	画笔的宽度	画笔的颜色	直线的起点	直线的终点

下面是实现鼠标交互绘制直线的功能。具体的操作步骤是 ：选中左侧工具栏中的“直线”项后，按下鼠标左健点中直线的起点，然后当鼠标在屏幕上移动时就拖动一条直线在屏幕上移动，如果再按下鼠标左键选中直线的终点，就完成了直线的交互绘制操作，如果在按下直线起点后按下的是鼠标右健，则擦除拖动的橡皮线，放弃本次直线绘制操作。以下是具体的实现过程。

➢ 修改消息处理函数 OnLButtonDown

为了完成鼠标交互绘制直线的操作功能，在电子白板实现模块文件中的消息处理函数 OnLButtonDown 中加入一些代码；在绘制直线状态下（ m_DrwaCurrent==1 ），当第一次按下鼠标左键时（ PushNumb==0 ），把鼠标按下点（ mPointOrign ）记录为直线的起点和鼠标移动时的上一个点（ mPointOld ），记录下按键次数并捕捉以后的鼠标操作。当第二次按下鼠标左键时（ PushNumb==1 ），就完成直线的绘制工作，然后，释放了捕捉到鼠标，并将鼠标左键次数设置为 0（ PushNumb=0 ），重新开始直线的交互绘制操作。

➢ 修改消息处理函数 OnMouseMove

为了在交互绘制直线时能实现拖到橡皮条线的功能，在此函数中加入一些代码。如果正在绘制一条直线，在已经第一次按下鼠标左键选中了直线起点（ m_DrwaCurrent==1 && PushNumb==1 ）的情况下，如果现在的鼠标点与上一个点不相同（ mPointOld!=point ），就要设置 R2-NOT 的绘制模式（ dc.SetROP2(R2_NOT)； ）。首先将原来的橡皮条线擦除，然后绘制直线起点到鼠标点的连线，并将当前鼠标点记录我上一个移动点 mPointOld. 当鼠标移到时，这个过程一直进行下去，就实现了拖动橡皮条线的功能。

在进行第一次移动操作时，没有上一条橡皮线供擦除，但因为这时鼠标点与直线起点相同，所以擦除的直线的起终点相同，不会在屏幕上留下一条多余的直线。

- 修改消息处理函数 OnRButtonDown

为了在交互绘制直线的操作过程中，能够放弃正在进行的绘制操作，需要在此消息处理函数中，加入一些代码。在交互绘制直线并且已经按中了直线起点〔正在进行直线拖动 ）的情况下，按中鼠标右键会将屏幕上拖动的橡皮线擦除，结束本次直线的绘制并释放捕捉的鼠标。

（ 2 ）绘制连续直线

要绘制连续直线，并传输到对等方，需要建立消息数据结构，包含的数据项见表 4-2。

表 4-2 连续直线消息数据结构

m_DrawCurrent	m_nPenWide	color	m_PointNumber	mPointOrign	point
当前正在绘制的线型	画笔宽度	画笔颜色	传输的点的个数	起点	终点

连续直线的鼠标交互绘制，采用与直线相似的方法 ：不同的是，连续直线是由多个顶点组成的，在纯鼠标操作状态下，需要加人更多的鼠标控制来完成操作。现在设定连续直线的绘制方法如下：运行绘制连续直线的操作菜单后，按下鼠标左键时，会在视图屏幕上连续绘制直线段，鼠标移动时，移动点和上一个顶点之间有拖动线，按下鼠标右键时，就会完成连续直线的绘制操作。

为了实现以上的交互绘制功能，同样需要对三个鼠标消息处理函数进行修改。实现过程与直线的实现类似。

不同的是消息处理函数 OnLButtonDown 。如果绘制的是连续直线且按下了二次以上鼠标左键 （一个顶点不能形成一条直线 ），在视图上绘出图型。最后，完成本次交互绘制操作并释放捕捉的鼠标。

（ 3 ）绘制圆和圆形区域

要绘制连续直线，并传输到对等方，需要建立消息数据结构，包含的数据项见表 4-3。

表 4-3 圆和样圆形区域消息数据结构

m_DrawCurrent	m_nBrushStye	b_Fill	m_nPenWide	color	mPointOrign	point
当前绘制类型	画刷类型	是否填充	画笔宽度	画笔颜色	起点	终点

在屏幕上直接用鼠标绘制画所用的方法很多，如圆心及半径法、三点法 （点中三点确定一个圆），两点法 （点中两点作为直径确定一个圆 ）等方法。此绘图以圆心半径法实现交互绘制圆和圆形区域的功能，用国心半径法画圆的操作步骤是：用鼠标左键点中一点作为圆心，然后当鼠标移动时，就会拖动一个以按下点为圆心，以第一个按下点和鼠标目前的移动点之间的距离为半径的圆移动。在这种情况下，按下鼠标左键会完成一个圆的绘制操作，而按下鼠标右键就会擦除拖动的圆而放弃绘制操作。

为了完成圆（或圆形区域 ）的鼠标交互绘制功能，同样需要对三个鼠标消息处理函数进行修改。在计算得到了圆心和半径的实际坐标和长度，完成本次交互绘制圆的操作，为了在鼠标交互绘制圆的过程中能够拖动橡皮圆，当鼠标移动时，如果当前鼠标移动点与上一个移动点不同，将经过原来移动点的圆，绘制经过当前点的圆。

为了在交互绘制圆和圆形区域的操作过程中，能够放弃正在进行的绘制操作，在交互绘制圆和圆形区域并且已经按中了圆心起点〔正在进行拖动）的情况下，按中鼠标右键会将屏幕上拖动的橡皮线擦除，结束本次圆的绘制并释放捕捉的鼠标。

（ 4 ）绘制任意曲线

要在白板上绘制任意曲线，并传输到对等方，需要建立消息数据结构，包含的数据项见表 4-4。

表 4-4 任意曲线消息数据结构

m_DrawCurrent	m_nPenWide	m_clorDraw	mPointOrign	point
当前绘制曲线	画笔的宽度	画笔的颜色	曲线每段直线的起点	曲线每段直线的终点

绘制任意曲线与绘制直线相同，任意曲线是有无数个很小的直线而组成的。下面是实现鼠标交互绘制任意曲线的功能。具体的操作步骤是 ：选中左侧工具栏中的“任意曲线”项后，按下鼠标左健点中任意曲线的起点，然后当鼠标在屏幕上移动时就拖动一条条很小的直线组成的曲线。如果在按下起点后按下的是鼠标右健完成本次任意曲线绘制操作。其具体实现过程和直线类似，同样修改那三个消息处理函数，不同的是在消息处理函数 OnMouseMove 中，每当鼠标移动一次，就绘制一条上一点到当前鼠标所在位置点的很小的直线，连续移动鼠标就会绘出任意曲线。

网络通信方面，要把电子白板的消息发送到服务器端，然后让服务器端来转发此类信息的内容，代码中都有一段消息的封装和发送函数，把绘制图形的属性都封装在一个消息类中，然后调用文档类的发送消息函数发送消息，接下来客户端要解析电子白板发送过来的消息，并据此在本地白板上显示出所绘制的图案。下面一段代码是绘制直线发送信息的代码：

```
if (m_DrwaCurrent==1)  // 如果正在绘制直线
{
        if (PushNumb==0)  // 如果是第一次按下鼠标左键
        {
            PushNumb++;   // 按下鼠标左键的次数加 1 次
            mPointOrign=point; // 直线的起点等于按中点
            mPointOld=point;   // 记录本次按中点
            SetCapture();     // 捕捉鼠标输入
        }
        else if (PushNumb==1)      // 第二次按下鼠标左键 （即按下直线的结束点 ）
        {

            CPen pen;
            pen.CreatePen(PS_SOLID,m_nPenWide,m_clorDraw);
            CPen *pOldPen=dc.SelectObject(&pen);
            dc.MoveTo(mPointOrign);
            dc.LineTo(point);
            dc.SelectObject(pOldPen);
            pen.DeleteObject();
```

```
                PushNumb=0;  // 鼠标左键按下次数为  0，重新进行直线的绘制
                ReleaseCapture();  // 释放捕捉的光标

               pDoc->msg.type=30;  //封装直线绘制的信息
               pDoc->msg.m_DrawCurrent=1;
                pDoc->msg.m_nPenWide=m_nPenWide;
               pDoc->msg.color=m_clorDraw;
               pDoc->msg.m_PointNumber=2;
                pDoc->msg.m_HvaePN[0]=mPointOrign;
               pDoc->msg.m_HvaePN[1]=point;
                pDoc->SendMsg();  // 发送消息

                mPointOrign=point;
            }
        }
```

其它线形的信息封装是类似的。

4.3.2 服务器端程序的设计

聊天服务器在界面上要比聊天客户简单许多，只有一个文本视图用于显示客户之间通信的信息，以及对用户在线列表的管理。同时，聊天程序服务器端也封装了消息类，该类与客户程序一致，客户和服务器正是在此基础上进行通信的。但服务程序分别封装了两个 Socket类　:CListeningSocket 和　CClientSocket类，前者用于管理侦听，后者用于管理连接。

1. CListeningSocket类

这个类主要是负责监听管理的套接字。

该类重载了 OnAccept 函数以使对来自客户的连接请求作出响应 ，当该套接字接收到客户的连接请求时，就调用文档对象的 ProcessPendingAccept函数进行处理。CListeningSocket类的定义如图 4-5 所示。

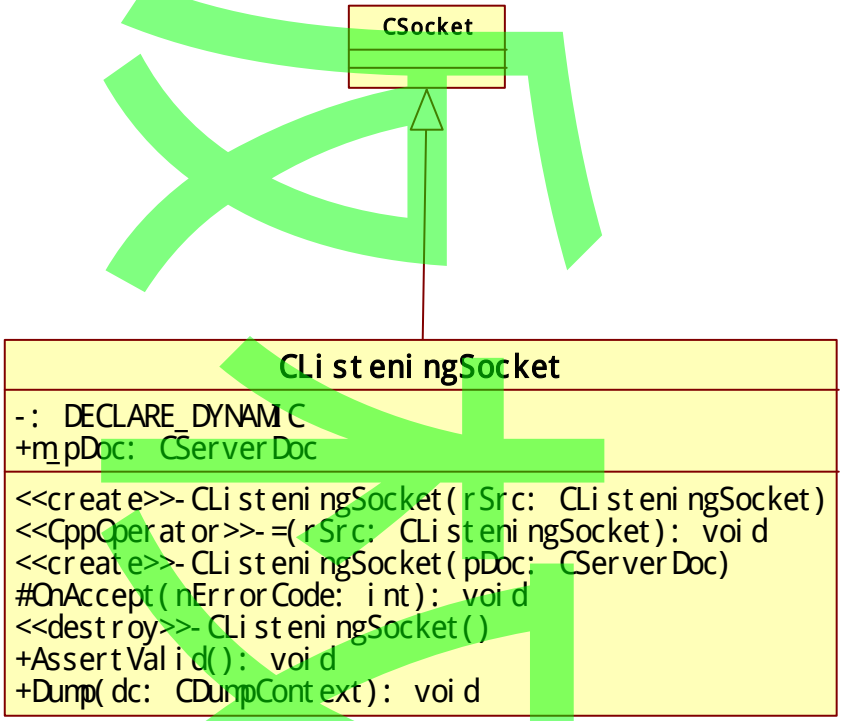


图 4-5 CListeningSocket类的定义

2. CClientSocket类

该类是负责连接管理的套接字类。该类重载了 OnAccept 函数以使对来自客户的连接请求作出响应。该类与前面的客户程序派生套接字类非常相似，唯一不同的是CClientSocket套接字类中封装了串行化功能，而在客户端应用程序中该串行化功能是由文档对象进行管理的。这也就是说，该类封装了客户端应用程序的大部分功能，因此可以认为该套接字类所管理的就是一个客户应用程序。

管理通信的类。在该服务端程序中，整个通信由CServerDoc来管理。

该类的定义如图4-6所示。

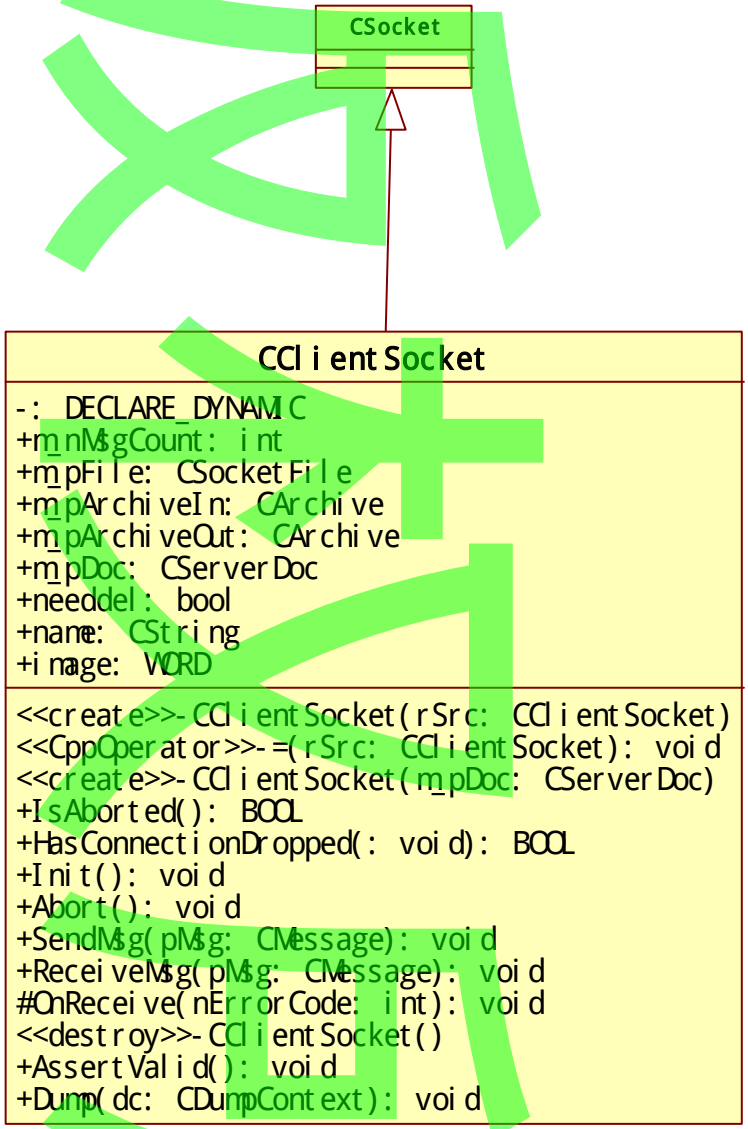


图 4-6 CClientSocket 类的定义

由于聊天服务程序具有与聊天客户程序相似的通信处理方式，因此下面只介绍几个有别于客户的方面。

对于客户应用程序来说，它只需管理一个连接套接字，而服务器需要管理若干个CClientSocket连接套接字对象，这就要在应用程序中有一个适当的组织形式，以便对套接字进行管理。这里在文档类中定义了一个指针链表对象：

CPtrList m connectionList;

每当服务器新建一个连接套接字对象时，就将其添加到 m_connectionList 链表中，这样就方便了对连接套接字对象的管理。

前面提到过，当服务器接收到客户的连接请求时，就会调用 ProcessPendingAccept 函数来完成连接，当服务器接收到客户发来的聊天信息时，就遍历连接套接字链表将聊天消息一一发送给所有客户，该功能由函数 UpdateClients 实现，当用户指定消息为“悄悄话”时，服务程序并没有进行过滤，该功能由客户程序实现。

(1) 服务器端的编程模型

主要利用了 CSocket 对象与串行化技术。

共同使用 CSocket 与 CArchive 对象是最简单的套接字编程模型，由于本系统是简单的多人会议系统，并没有采用多线程编程技术。由于 CArchive 对象将帮助程序员处理许多以前必须使用 API 或 CAsyncSocket 类来处理的通信问题，大大减少了编程的工作量，故本系统的实现正是采用了这种方法。如果与非 MFC 服务器进行通信，就不能使用 CArchive 对象，因为这种服务器并不能处理这种类型的对象，这时就不得不使用字节排序的套接字编程方法。

对于套接字来说，归档对象是与 CSocketFile 对象而不是与标准 CFile 对象相关的，与连接到一个磁盘文件不同， CSocketFile 对象连接到一个 CSocket 对象。一个 CArchive 对象将负责管理一个缓冲区。当 storing(发送) 归档对象的缓冲区被填满时，相关的 CSocketfile 对象将缓冲区的内容取出，清空与套接字相关的归档缓冲区。当 loading(接收) 归档对象的缓冲区被填满时， CSocketFile 对象停止读出直到缓冲区可用。

下面是使用 CSocket 对象进行客户和服务器之间通信的一般编程模型，它只适用于数据流套接字，因为数据报套接字不能使用 CArchive 。

①分别构造服务器和客户套接字对象。

②调用对象的 Create 函数创建套接字，而 Create 函数会调用 Bind 函数将此套接字绑定到指定的地址。需要注意的是为服务器创建套接字时需要为其指定端口号。

③套接字创建完毕后，服务器调用 Listen 成员函数开始侦听客户的连接请求，而客户可以调用 Connect 成员函数向服务器请求连接。

④当服务器监听到客户连接请求时，创建一个新的套接字，并将其传送给 Accept 成员函数以接收客户的连接请求，函数执行失败会返回特定的错误码。

⑤为服务器和客户的套接字对象分别创建一个与之相联系的 CSocketFile 对象。

⑥为服务器和客户的套接字对象分别创建一个与 CSocketFile 相联系的 CArchive 对象以进行数据的发送和接收。

⑦使用 CArchive 对象在客户和服务器套接字之间传送数据。

⑧在任务执行完成后，销毁 CArchive, CSocketFile 和 CSocket 对象。

CArchive 对象只能单向传递数据 ：载入 （接收 ）或存储 （发送 ）。在某些情况下，用户必须使用两个 CArchive 对象，一个进行数据发送，而另一个进行数据接收 . 并且 CArchive 类特别为 CSocket 类提供了 IsBufferEmpty 成员函数，如果缓冲区包含多个数据消息，例如，需要循环直到所有的数据都被读出，并且缓冲区被清空。否则，下一个将接收数据的通告可能被无限延迟。使用 IsBufferEmpty 函数，以确定接收到了所有数据。

(2) 通信的实现

服务端主要承担客户端发来的信息，并解析信息的类型，根据不同的类型来处理信息。服务端消息类的封装与客户端是一致的，这是必然的，不然消息的解析就会不成功，而达不到信息的交换。有关此类的详细设计客户端已经解释过了，此不再复述。再者服务端也封装了两个通信类，一个用于侦听，一个用于管理客户端套接字的管理。这两个类概要设计也介绍过了。

当服务器端启动的时候，是在文档类中创建一个用于管理通信的 CListeningSocket ，此后的客户端的连接都是由它来管理。然后一直侦听、等待客户的连接。

当有客户端连接时，就会自动调用 CListeningSocket 的 :OnAccept(int nErrorCode) 函数。由于文档是数据管理的核心，所以在此函数中调用了文档类的一个成员函数 ProcessPendingAccept(); ，来处理客户端的套接字信息，这个函数中，会把连接的客户端套接字保持到一个链表中。以后的查找工作都会和他打交道。以后就开

始服务端和客户端信息的收发工作了。

当客户端发送信息给服务器端时，就会调用管理通信类 CClientSocket 的函数 OnReceive ，在这个函数中，实际的处理工作还是由文档类来承担的，承担此任务的是文档类中 ProcessPendingRead 函数，这个函数中是对客户端发送到服务器端信息的处理，是登录时，就会更新用户列表，由函数 UpdateList(pSocket) 实现；是信息时，就会更新客户端的信息显示，由函数 UpdateClients() 来实现。下面看看这两个函数的的实现代码，这两个函数是服务器端信息处理最重要的，关系到客户端信息能不能接收到其他用户信息的关键。

先看看 UpdateList(pSocket) 函数是怎样通知客户端用户列表的更新显示的。

此函数要实现用户端登录时用户列表信息显示的更新，必须遍历之前提到的那个用于保存客户端套接字的链表，逐一给客户端发送信息来实现，这时候要修改信息的一些属性，如协议类型，来让客户端解析信息。从而更新用户列表。

下面看看 UpdateClients() 函数怎么来更新用户端聊天信息和电子白板信息的交互的

其实现原理与上一个函数是相似的。这里一会介绍一下这个函数中调用的 SendMsg(pSocket, pMsg)函数，该函数实现对客户端发送信息。

这两个函数是对信息的处理，但是处理之前先必须读取客户端发送过来的信息，这两个函数之前都有一个函数 ReadMsg ，正是这个函数来读取客户端发送过来的信息。

在对取数据时，根据信息的类型不同，在服务器端显示客户端的一些信息，如当前用户列表清单，聊天信息的显示等。如 msg.type == -1 表示用户登录的信息，msg.type == -2表示用户离开会议，其他的聊天信息会在服务器端显示。

4.4 运行结果

系统的开发环境为 Visual Studio 6.0，采用了 MFC 框架来实现，测试和运行环境为： Windows XP 系统， 1GB 内存的机器。系统运行结果简要阐述如下。

1. 主界面

客户端如图 4-7 。左边是当前在线用户列表，右边是聊天内容的显示区域。下面是发送信息区域。

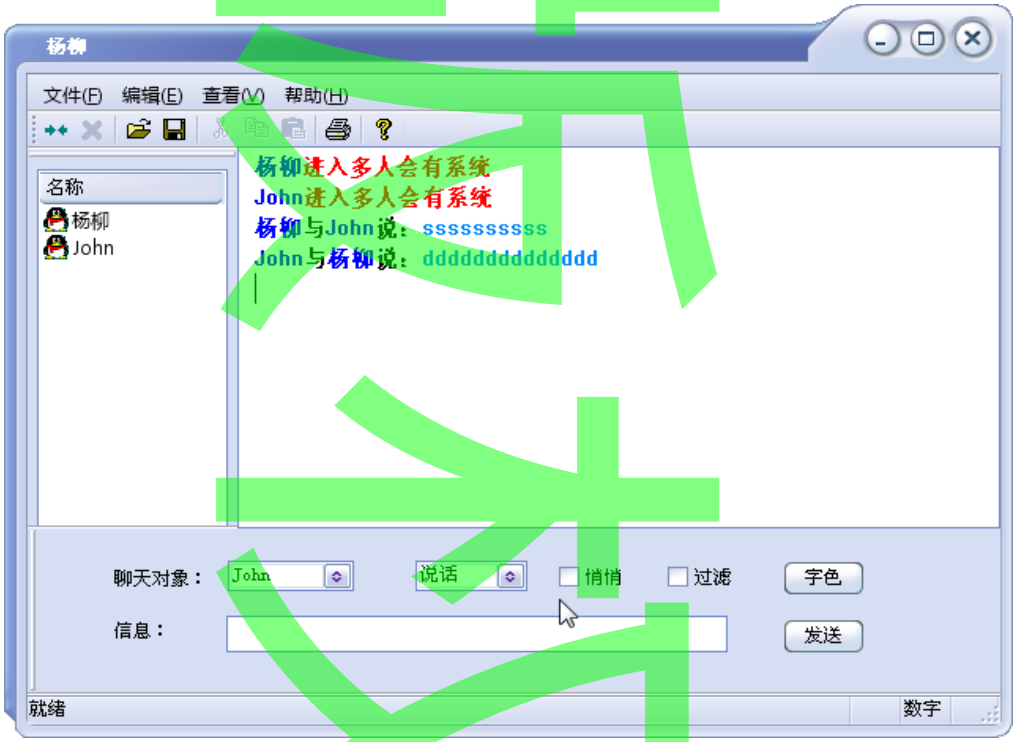


图 4-7 客户端主界面

服务器端如图 4-8 所示。左边和客户端相同，是当前在线用户列表，而且还有用户的 IP 地址信息。右侧为当前在线客户端用户聊天信息的显示区域。

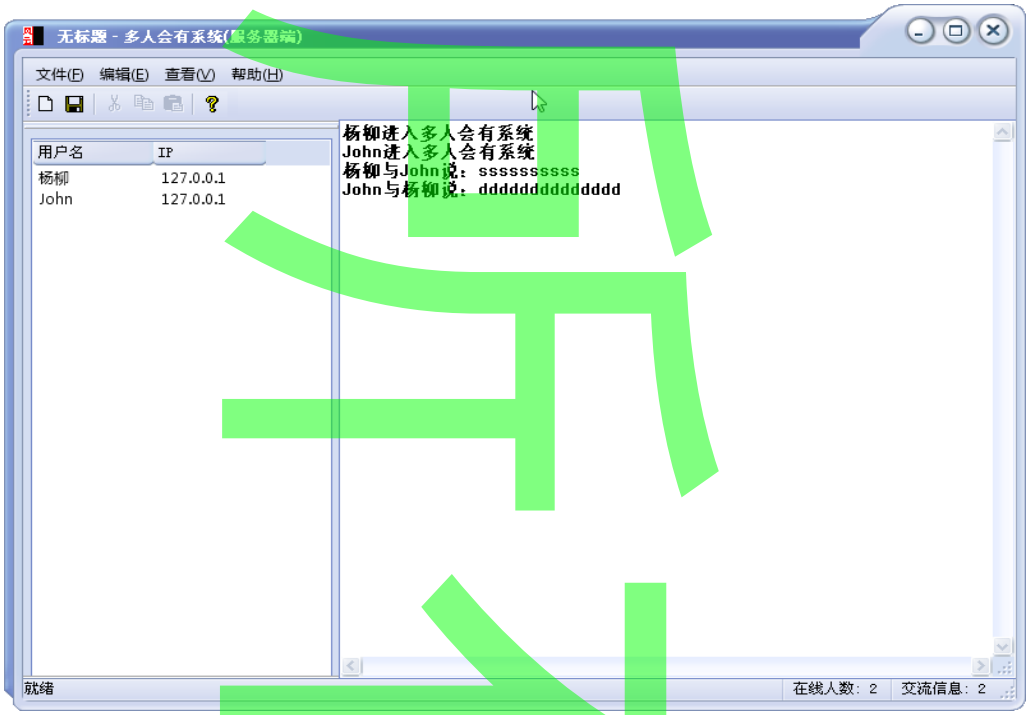


图 4-8 服务器端界面

2. 白板功能

直线绘制如图 4-9 所示。界面左边是工具箱，可以选择几种要绘制的线条类型，右

侧工具箱可以选择线条的粗细及填充区域的填充类型。左下方带颜色的那个方块是用来选择颜色，类似于调色板。

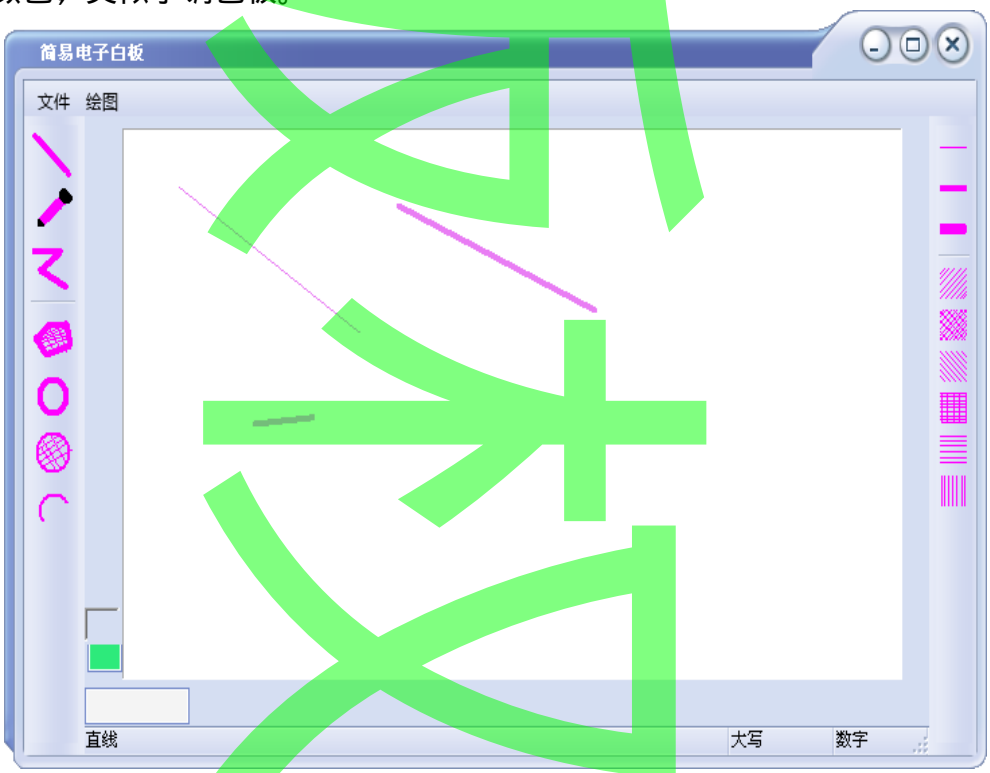


图 4-9 直线的绘制

连续直线的绘制如图 4-10 所示。

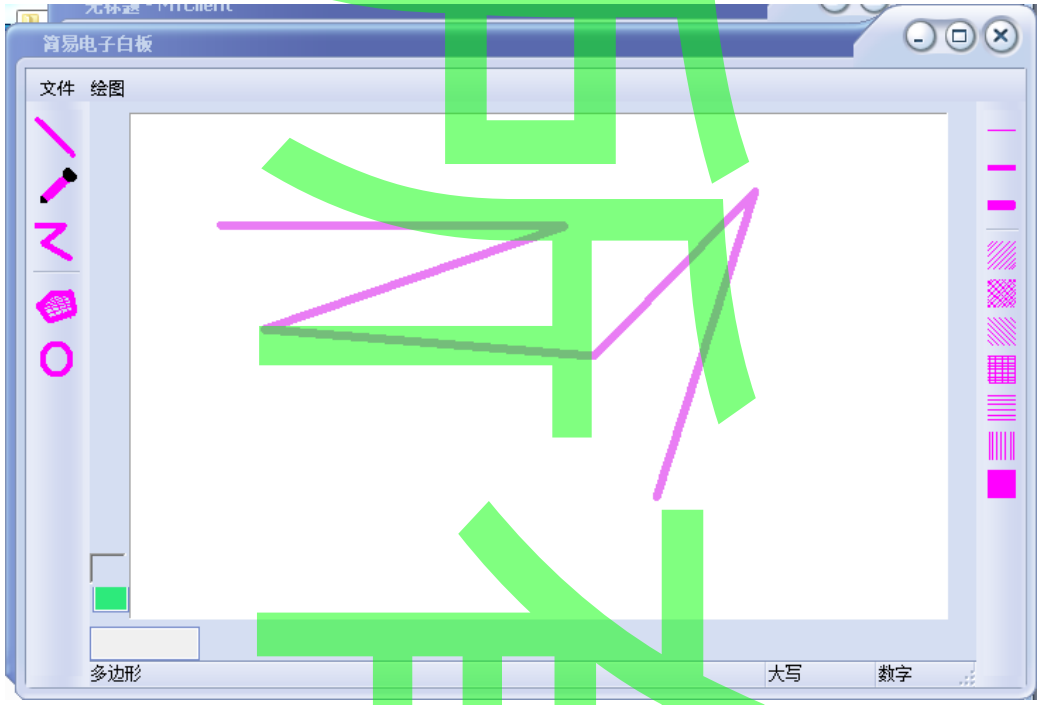


图 4-10 连续直线的绘制

圆和圆形区域的绘制如图 4-11 所示。

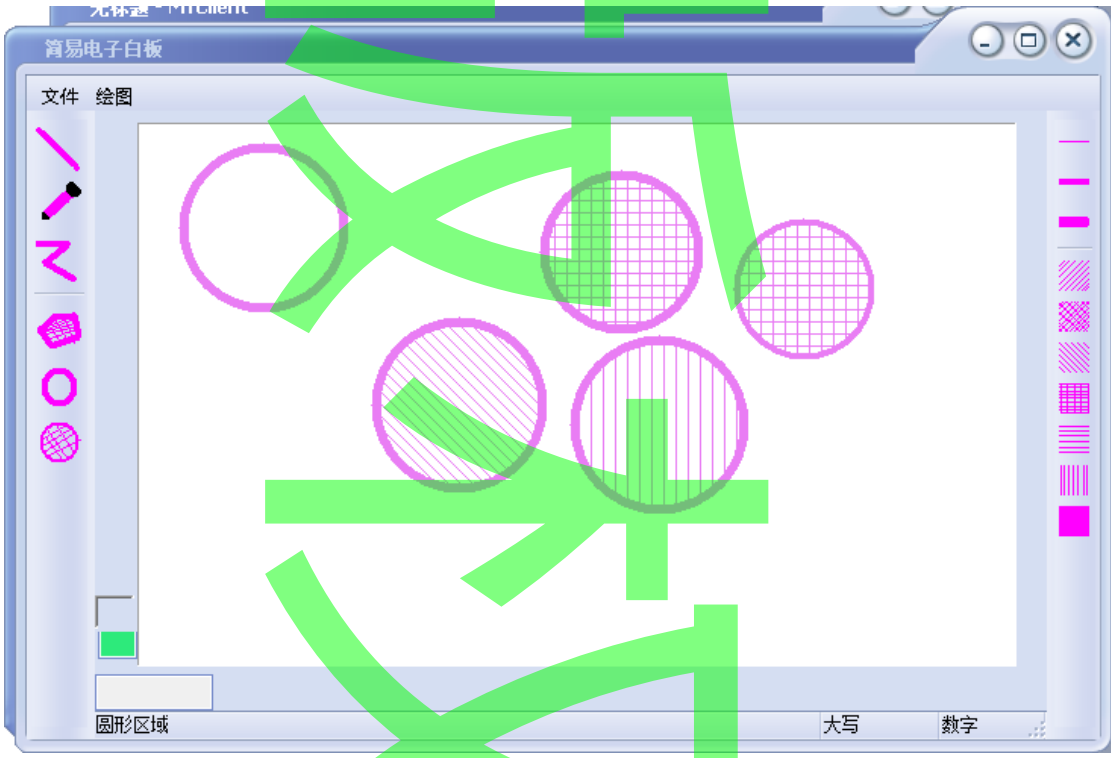


图 4-11 圆和圆形区域的绘制

任意曲线的绘制如图 4-12 所示。

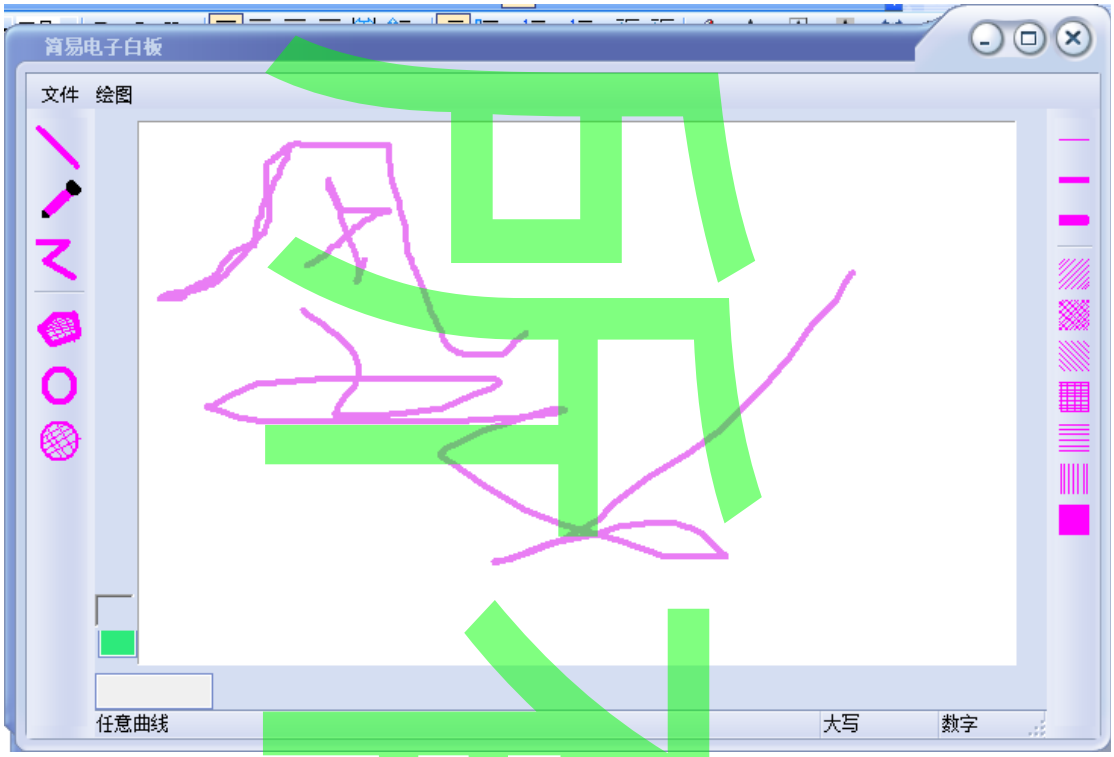


图 4-12 任意曲线的绘制

结束语

多人会议系统主要采用了微软的 MFC 框架，共同使用了 CSocket 与 CArchive 对象。这两个对象是最简单的套接字编程模型，其中 CArchive 对象将帮助程序员处理许多以前必须使用 API 或 CAsyncSocket 类来处理的通信问题，大大减少了编程的工作量。本系统的实现采用了串行化技术。

系统主要实现了参加会议人员的聊天、电子白板功能。聊天主要是群聊，类似于 QQ 群聊，而且用户还能看到当前在线用户列表。当然服务器端也能看到当前用户列表和在线人员的聊天信息。电子白板模块主要实现了一个类似于绘图的程序，但它能够实现网络通信，即一个客户端所绘的图形，其它在线用户只要打开电子白板视图就能看到对方所绘制的图形。绘图的图形种类主要有 ：直线、连续直线、任意曲线、圆和圆形区域。而且可以选择绘制线条的粗细、颜色及填充区域的填充样 .

系统实现了很简单的聊天功能及简单的电子白板功能，还有不足和需要改进的地方。聊天功能群聊是实现了，但应该实现私聊功能，完全类似于 QQ 那种聊天方式功能是系统应该实现的。客户端的电子白板只实现了简单的几个图形的绘制，还没有实现图形保存、删除及撤销当前绘制等等操作，作为能够商业化的系统是应该实现的功能不只这些，还有协调浏览，能够打开文档共享等等功能。服务器端只是简单的显示了当前用户列表、对应 IP 地址及客户端交流信息的显示，服务器端不足之处主要是没有控制权，应该实现强大的管理和控制客户信息，如注册用户信息的保存和更新，可以屏蔽某语言不好用户的发言权，甚至可以把当前某用户踢出会议室，这是一些服务器端需要改进的功能。

参考文献

[1] 孙鑫. VC++深入详解. 北京：电子工业出版社，2006

[2] John E.Swanke. Visual C++MFC 扩展编程实例. 北京：机械工业出版社，2000

[3] 张忠帅. VC++ 2008 专题应用程序开发实例精讲. 北京：电子工业出版社，2008

[4] 胡峪，刘静. VC++高级编程技巧与示例. 西安：西安电子科技大学出版社，2001

[5] 郭克新. Visual C++ 代码参考与技巧大全. 北京：电子工业出版社，2008

[6] 电脑编程技巧与维护杂志社. Visual C/C++ 系统开发典型实例解析. 北京：水利水电出版社，2007

[7] 李媛媛. Visual C++ 网络通信开发入门与编程实践. 北京：电子工业出版社，2008

[8] 胡鸣. Windows 网络编程技术. 北京：科学出版社，2008

[9] 李英. Visual C++ 编程与项目开发. 北京：华东理工大学出版社，2004

[10] 严华峰等. Visual C++ 课程设计案例精编. 北京：水利水电出版社，2004

[11] 青软实训组，钟岱晖. C++ 开发之路. 北京：电子工业出版社，2009

[12] 陈坚，陈伟等. Visual C++ 网络高级编程. 北京：人民邮电出版社，2001

[13] 严华峰等. Visual C++ 课程设计案例精编. 北京：中国水利水电出版社，2002

[14] 陈建春. Visual C++ 开发 GIS 系统--开发实例剖析. 北京：电子工业出版社，2009

致 谢

在本次毕业设计过程中 ，翁广安讲师对该论文从选题、构思、资料收集到最后定稿的各个环节给予细心指引与教导，使我对计算机网络通信有了深刻的认识，使我得以最终完成毕业设计，在此表示衷心感谢。翁老师严谨的治学态度、丰富渊博的知识、敏锐的学术思维、精益求精的工作态度、积极进取的科研精神以及诲人不倦的师者风范是我终生学习的楷模。导师的高深精湛的造诣与严谨求实的治学精神将永远激励着我。在四年的大学生涯里，感谢母校给我带来的学习环境，感谢系和班级里德同学在学习上和生活上的关怀，还得到众多老师的关心支持和帮助，在此，谨向老师们致以衷心的感谢和崇高的敬意！

感谢我的父母对我多年来在物质及精神上的帮助和支持，是他们的不断鼓励使得我能够顺利完成学业，。在我求学的生涯中，每一个前进的脚印都凝结了他们辛苦的汗水。

最后，我要向在百忙之中抽时间对本文进行审阅、评议和参加本人论文答辩的各位师长表示感谢！