

同济大学软件学院

硕士学位论文

即时通讯技术系统的研究与实现

姓名：张鑫焱

申请学位级别：硕士

专业：软件工程

指导教师：金伟祖;杭咏新

20090501

摘要

即时通信 IM (Instant Messaging) 是目前 Internet 上最为流行的通讯方式, 而各种各样的即时通信软件也层出不穷; 服务提供商也提供了越来越丰富的通信服务功能。随着互联网的发展, 即时通信的运用日益广泛, 即时通信软件业方兴未艾。

点对点网络已经被越来越多的用户所需要并且作为一种标准的分发信息的方式登堂入室, 因为它的结构使得网络富有延展性, 相较普通网络有更高的效率和更好的表现。P2P(peer-to-peer)网络是非中心化, 自组织和动态的网络, 并且为传统的客户端-服务器计算模型提供另外一种选择。客户端-服务器(C/S)结构允许用户链接到某一个服务器, 尽管服务器是可扩展的, 但这总有限制。而 P2P 网络却拥有几乎无限的扩展可能。

本文首先描述了 P2P 技术的发展和现状, 然后初步探讨了即时通信所涉及的一些技术: 如通讯协议、服务模式等; 最后对现有的技术进行了对比和分析, 提出了一些可能的改进方案, 并且设计了一个基于 P2P 的即时通信系统。

在系统设计与建模过程中, 使用了 UML 和面向对象的分析、设计方法, 并使用 Rational Rose 作为建模工具; 本系统基于 C++, 使用 VC 作为开发工具。最终实现了多个用户可以点对点的进行即时通信。

关键词: P2P, 即时通信, TCP, 客户端, 服务器

Abstract

Instant Messaging is currently the most popular way to communicate on the Internet, by the way various Instant Messaging software have been continuously appearing; service provider offers more and more communication services nowadays. As the development of the Internet, Instant Messaging will perform a wider use and is growing up day after day.

Peer-to-peer network is needed by a growing number of users as a standard distribution of a standard information arrival method, because it makes the structure of the network more scalability, which has higher efficiency and better performance than ordinary networks. P2P network has no center, self-organized, from simple sense, it is a dynamic (network) and provides another option to the traditional client-server model. client-server structure allows users to link to a certain server, while the server can be extended, it is always limited. but peer-to-peer network has almost unlimited expansion potential.

This article first describes the development and the current status of the Instant Messaging. Then it initially discusses the technologies of the Instant Messaging such as communication protocol, service mode and friend-making mode. It makes a comparison and analysis with respect to the current technology and also provides some possible solution for improvement. At last it designs and develops an Instant Messaging System.

During the process for system design and modeling, I use UML and object-oriented analysis & design method, use Rose as the tool for modeling. This system is based on C++ and uses VC as the development tool.

Key words: P2P, Instant Messaging, TCP, Client, Server

同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

签名：

张金林

2009年6月13日

学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：张鑫炎

2009年6月13日

经指导教师同意，本学位论文属于保密，在 年解密后适用本授权书。

指导教师签名：

学位论文作者签名：

年 月 日

年 月 日

第1章 引言

1.1 课题起源

1.1.1 即时通信的产生

随着电脑和互联网的普及,尤其是移动互联网的发展,标志着以“联网、易用、时尚”为特征的后PC时代的到来。传统的3C被赋予了新的含义,人们利用电脑和互联网的时间越来越多,以“社区(Community)”、“内容(Content)”、“商务(Commerce)”为主要特征的网络即时通信IM(Instant Messenger),最大程度的体现了网络给人们生活带来的变化,使得人们的沟通更加方便、快捷,使人们真正有了天涯若比邻的“地球村”的感觉。网络即时通信是一种在后PC时代兴起的,以Internet网络及其他有线、无线网络为基础的,在交互双方之间实时地传送语音、文字、图像等信息的通信方式。1996年7月,四个以色列的年轻人在特拉维夫成立了一家名为Mirabilis的小公司,公司的名称取自拉丁语,意为神奇。四个月之后,世界第一个即时通信软ICQ在他们手中诞生,自此拉开了神奇的序幕。他们没想到,当初仅仅是为了使连接在同一个服务器上的用户能相互交流而开发的ICQ,在后来的日子里能如此风光无限;他们更没想到,即时通信软件迅速席卷了全球网民。即时通信软件的最大特点是在网上进行信息的实时交流,它的产生有着深刻的社会和技术原因。大凡人们都有渴望社交、获得社会尊重、实现自我的需求,这正是网络即时通信软件风行的驱动力。而物质文明的日益发达所带来的副作用,又使得人们习惯地与周围的人保持距离,以致人们更愿意对陌生人敞开心扉。与传统的通信方式相比,即时通信具备快捷、廉价、隐蔽性等特点,在网络上可以跨年龄、身份、行业、地域的限制,达到人与人、人与信息之间的零距离交流。从这点上讲,网络即时通信的出现改变了人们的沟通方式和交友文化,大大拓展了个人生活交流的空间。

1.1.2 即时通信的定义

即时通信是指能够即时发送和接收互联网消息等的业务。利用“即时通信”工具,网民间可以实现异地文字、语音、视频的实时互通交流;同时,借助即时通信工具,还可以帮助企业提高业务协同性及反馈的敏感度和快捷度。作为使用频率最高的网络软件,即时通信已经突破了技术上的种种局限,加强了系统的稳

定性，被认为是现代交流方式的新象征。

即时通信大部分的模式依然采用CS(Client/Server, 客户端/服务器)结构, 如图1.1, 但它不同于传统的客户端/服务器结构。用户首先从即时消息IM服务器上获取好友列表, 以建立点对点的联系, 然后用户(Client1)和其好友(Client2)之间采用点对点方式发送信息; 在无法直接点对点联系时, 则用服务器中转的方式完成。

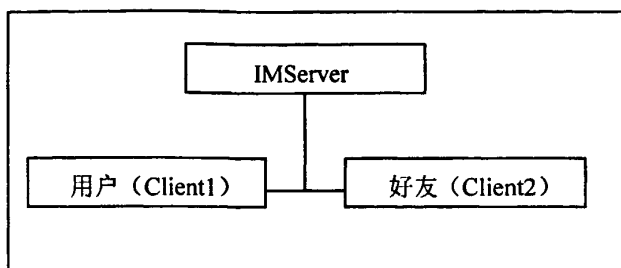


图 1.1 典型的即时通信体系结构

1.1.3 即时通信的优势

即时通信系统的互动性高于传统的BBS(Bulletin Board System, 电子公告板系统)和E-mail。传统的邮件投递方式现在被称为“蜗牛邮件”，因为收件人至少要等一天才能从邮局收到信件。而E-mail虽然在发送后的几秒钟内即可抵达收件人的电子信箱，不过E-mail的致命弱点是发送者很难知道收信人是否即时上线接收邮件。而即时通信系统可以使信息的收发双方在线上进行互动交流。利用它可以实时传送文字信息、语音信息和发送文件。另外，它还可以作为单位内部联络的一种方式。可以用来召开网络会议，比起实地的会议来，不仅快捷，而且节省开支。

近年来, 随着无线通信的快速发展, 移动互联网为传统的Internet注入了新的活力, 同时也为即时通信带来了一场革命, 交互双方已经不再局限于以网络终端设备为唯一的接入手段, PC机在传统的网络即时通信中所起的作用正在被手机、电话所取代。这使得网络即时通信更加大众化和人性化, 即使不会使用电脑的人现在也可以享受到即时通信给人们生活带来的变化。另外, 即时通信软件被加入了越来越多的新功能, 使它正从毫无商业应用价值的聊天、游戏转变出来, 成为能为商业企业带来商机和实惠的领域。所有的这一切都让人们人们对网络即时通信赋予了越来越多的注意力。可以说, 即时通信的发展把后PC时代的特征体现得淋漓尽致, 人们从来没有像今天这样感受到天涯咫尺。

1. 1. 4 即时通信的主要应用

目前,大多数用户选择即时通信软件主要用作与好友沟通交流,部分用户则以“交更多的朋友”作为主要目的。由此可见,即时通信的最主要功能是作为大家彼此交流沟通的工具。随着即时通信的普及,人们也越来越多地在工作环境中享受到这些软件所带来的便利。根据AOL调查结果显示:

(1)27%的即时通信用户都在工作时使用即时通信服务,比去年增长了71%;

(2)在工作时使用即时通信软件的用户中,70%的人主要利用即时通信软件与同事进行联络;

(3)62%的用户偶尔会在工作时间使用即时通信系统与家人和朋友联络,34%的用户利用它与客户进行沟通,仅1%的办公室即时通信用户使用即时通信来避免与别人面对面沟通可能带来的尴尬。即时通信之所以拥有如此大规模的用户群体,“免费”是其很重要的一个因素。

艾瑞市场咨询公司通过多家网站联机和 E-mail 问卷调查获得的数据显示,目前国内即时通信用户仍然以低收入为主,其中月收入低于 1500 元的用户比例高达 64.5%,而月收入超过 5000 元以上的用户仅占 3.83%的比例。另一个主要原因便是高速互联网连接的普及,大约 71%的互联网用户都是在家里使用高速互联网连接。这些用户中 1/3 以上的人认为,高速的互联网接入使他们把更多的时间用于即时通信服务,进行共享图片、文件等。因此,不难预见,即时通信在提高工作效率和降低通信成本方面发挥着其他通信方式不可替代的作用,从而必将成为主流通信工具之一。

1. 2 课题目的和意义

随着信息化的普及和发展,现在网络不仅在各企事业单位中的应用越来越广泛而且也已经逐渐融入了每个人的生活当中,逐渐的成为人们生活中不可或缺的一部分。

为了更好的充实人们的生活,我特做此系统满足人们在日常生活的需要和需求,也为了满足人们在信息流通方面的方便,使得人们更能分享互联网上的资源,使得的网络的意义更能充分的体现。

交流是系统的关键也是目的,它能让你在简单的对话中了解复杂的社会,多彩的世界,使得人与人之间的距离彻底拉近,就像是面对面的交谈,彼此之间没有界限,没有差别,只有心与心的共振。

它也是你心灵抒发的对象,把你心中的快乐,郁闷,论点,观点肆无忌惮的

阐发出来,没有压制,只有争论。它就是你阐发心灵的平台,是你交友的另一重要方式。缺少了它你会感觉没什么,但是一旦有了它你会发现它是那么的重要。

1.3 课题的发展情况

1.3.1 学术机构研发

1.北京大学—Maze

Maze 是北京大学网络实验室开发的一个中心控制与对等连接相融合的对等计算文件共享系统,在结构上类似 Napster,对等计算搜索方法类似于 Gnutella。网络上的一台计算机,不论是在内网还是外网,可以通过安装运行 Maze 的客户端软件自由加入和退出 Maze 系统。每个节点可以将自己的一个或多个目录下的文件共享给系统的其他成员,也可以分享其他成员的资源。Maze 支持基于关键字的资源检索,也可以通过好友关系直接获得。

2.清华大学—Granary

Granary 是清华大学自主开发的对等计算存储服务系统。它以对象格式存储数据。另外,Granary 设计了专门的结点信息收集算法 PeerWindow 的结构化覆盖网络路由协议 Tourist。

3.华中科技大学—AnySee

AnySee 是华中科大设计研发的视频直播系统。它采用了一对多的服务模式,支持部分 NAT 和防火墙的穿越,提高了视频直播系统的可扩展性;同时,它利用近播原则、分域调度的思想,使用 Landmark 路标算法直接建树的方式构建应用层上的组播树,克服了 ESM 等一对多模式系统由联接图的构造和维护带来的负载影响。

1.3.2 企业研发产品

1.广州数联软件技术有限公司-Poco

POCO 是中国最大的 P2P 用户分享平台,是有安全、流量控制力的,无中心服务器的第三代 P2P 资源交换平台,也是世界范围内少有的盈利的 P2P 平台。目前已经形成了 2600 万海量用户,平均在线 58.5 万,在线峰值突破 71 万,并且全部是宽带用户的用户群。成为中国地区第一的 P2P 分享平台。

2.深圳市点石软件有限公司-OP

OP-又称为 Openext Media Desktop,一个网络娱乐内容平台, Napster 的后继者,它可以最直接的方式找到您想要的音乐、影视、软件、游戏、图片、书籍

以及各种文档，随时在线共享文件容量数以亿计“十万影视、百万音乐、千万图片”。OP整合了Internet Explorer、Windows Media Player、RealOne Player和ACDSee，是国内的网络娱乐内容平台。

3.基于 P2P 的在线电视直播-PPLive

PPLive 是一款用于互联网上大规模视频直播的共享软件。它使用网状模型，有效解决了当前网络视频点播服务的带宽和负载有限问题，实现用户越多，播放越流畅的特性，整体服务质量大大提高。

1. 4 课题要求

通过本课题的最终设计，要求设计者实现一个基于 P2P 的即时通信方案，能搜索并记录双方网络 IP 地址，基于 TCP 协议进行文本内容的传输，并实现一对多发送文本信息的功能；要求设计者对基于 P2P 的即时通信技术有一个系统地、全面地了解，为基于 P2P 相关领域的软件开发打下一定的编程基础。

第2章 相关理论与技术

2.1 P2P 模型简介

最近几年,P2P迅速成为计算机界关注的热门话题之一,财富杂志更将P2P列为影响Internet未来的四项科技之一。P2P打破了传统的Client/Server (C/S)模式,在网络中的每个结点的地位都是对等的,既充当服务器,为其他结点提供服务,同时也享用其他结点提供的服务。P2P技术的特点体现在以下几个方面:

1.非中心化

网络中的资源和服务分散在所有结点上,信息的传输和服务的实现都直接在结点之间进行,可以无需中间环节和服务器的介入,避免了可能的瓶颈。P2P的非中心化基本特点,带来了其在可扩展性、健壮性等方面的优势。

2.可扩展性

在P2P网络中,随着用户的加入,不仅服务的需求增加了,系统整体的资源和服务能力也在同步地扩充,始终能较容易地满足用户的需要。整个体系是全分布的,不存在瓶颈。理论上其可扩展性几乎可以认为是无限的。

3.健壮性

P2P架构天生具有耐攻击、高容错的优点。由于服务是分散在各个结点之间进行的,部分结点或网络遭到破坏对其它部分的影响很小。P2P网络一般在部分结点失效时能够自动调整整体拓扑,保持其它结点的连通性。P2P网络通常都是以组织的方式建立起来的,并允许结点自由地加入和离开。P2P网络还能够根据网络带宽、结点数、负载等变化不断地做自适应式的调整。

4.高性能/价格比

性能优势是P2P被广泛关注的一个重要原因。随着硬件技术发展,个人计算机的计算和存储能力以及网络带宽等性能依照摩尔定理高速增长。采用P2P架构可以有效地利用互联网中散布的大量普通结点,将计算任务或存储资料分布到所有结点上。利用其中闲置的计算能力或存储空间,达到高性能计算和海量存储的目的。通过利用网络中的大量空闲资源,可以用更低成本提供更高的计算和存储能力。

5.隐私保护

在P2P网络中,由于信息的传输分散在各结点之间进行而无需经过某个集中环节,用户的隐私信息被窃听和泄漏的可能性大大缩小。此外,目前解决Internet隐私问题主要采用中继转发的技术方法,从而将通信的参与者隐藏在众多的网络

实体之中。在传统的一些匿名通信系统中,实现这一机制依赖于某些中继服务器结点。而在P2P中,所有参与者都可以提供中继转发的功能,因而大大提高了匿名通讯的灵活性和可靠性,能够为用户提供更好的隐私保护。

6. 负载均衡

P2P网络环境下由于每个结点既是客户端又是服务器,减少了对传统 C/S 结构服务器计算能力、存储能力的依赖,同时因为资源分布在多个结点,更好的实现了整个网络的负载均衡。

2.2 基于 P2P 即时通信模型选择

2.2.1 即时通信系统的一般模型

即时通信服务有两个实体:发送者和即时收信箱。即时通信协议定义了即时通信服务、发信者和即时收信箱之间的交互作用。即时通信服务与其他邮件服务不同之处在于,即时消息本身足够的小,便于快速交付到即时收信箱。个体,用户代理,即时通信系统的模型用图 2.1 表示:

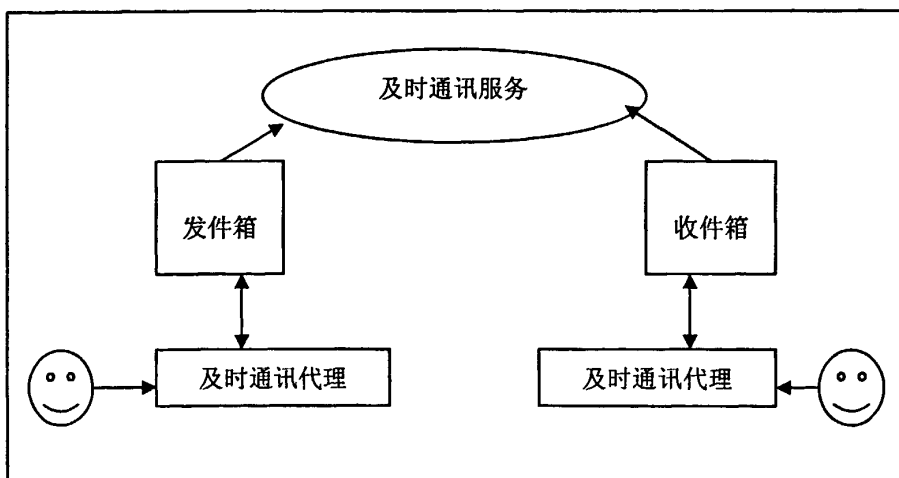


图 2.1 即时通信系统的一般模型

即时通信系统模型发信者或者即时收信箱的身份鉴别。

- 1.不同的收信箱有不同的身份鉴别需求。
- 2.不同的旁观者有不同的身份鉴别的需求。

3.服务内部可能存在多服务器或者多代理服务器。也就是说一个即时通信服务的逻辑的连接,内部可能有复杂的重定向和代理结构。代理服务器即时收信箱的身份与其他服务器交换即时消息。

4.服务并不需要一个服务器,可以通过发信者和即时收信箱之间的直接通讯实现。

5.涉及与其他即时通信服务的情况,可能会有一定的内部机制。

个体还可以通过收件箱用户代理设置交付规则(Delivery Rules)限制即时通信服务把即时信息交付到即时收信箱。

2. 2. 2 即时通信系统拓扑模型

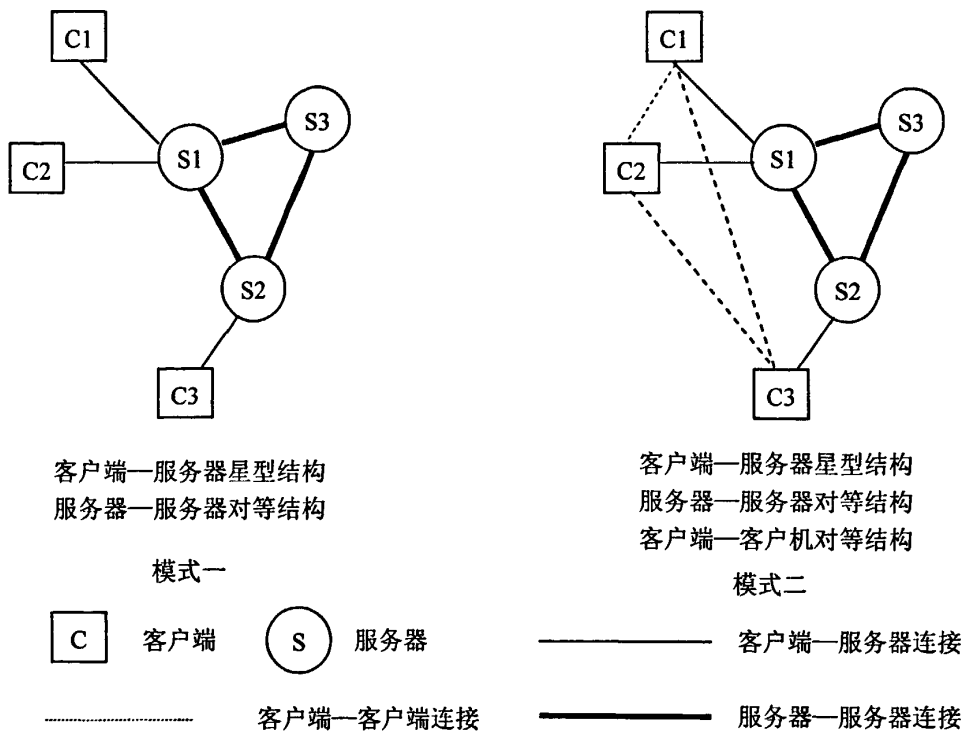


图 2.2 服务的内部拓扑结构

现在不少即时通信系统服务的内部拓扑中采用模式一;也有极其个别系统在即时通信服务上面采用模式二(例如,深圳腾讯的 QQ 服务中的即时通信服务,发信者首先考虑直接与即时收信箱进行通讯,如果发现直接通讯超时,会转为使用服务器中转完成通讯)。

模式一是一种非常成熟的模式,大多数 TCP/IP 应用系统,例如 SMTP 服务、DNS 服务、WWW 服务等都是使用这种模式。模式二,对于模式一是一个技术上的修正,虽然这个修正带来了新的问题,但是其实用性是不可质疑的;本系统正是实用模式二作为服务模式。

本系统的服务是由服务器构成的,本系统中每个实体都由一个寄主服务器管理。个体必须向服务器注册,注册后就有可以控制的发信者和即时收件箱。发信

者是独立于即时通信服务外的实体，而即时收信箱则存在于寄主服务器中。本系统的通讯系统中只有存在发送者和接收者两类实体。注册后的用户必须在服务器上建立登记表，才能知道其他用户的存在，与其他用户进行交流。

2.3 通信协议

2.3.1 TCP 协议

TCP 和 UDP 是 TCP/IP 协议族的两个主要的传输协议，TCP 是面向连接的，UDP 是面向无连接的，而实际上他们最大的区别在于为应用协议提供不同的服务，TCP 协议为应用程序提供点到点的通信：建立可靠的连接。如果有故障发生，阻碍分组到达远程系统，或者服务器不接受连接，客户都会得到通知：数据可靠的交付，故障而不能交付，发送方会得到通知；具有流控的传输；双工的传输，单个 TCP 连接允许同时双向传送数据流模式，TCP 从发送方到接收方发送没有报文边界的字节。

2.3.2 UDP 协议

UDP 为应用程序提供多对多的通信，UDP 在进行通讯的应用的数量上面，具有更大的灵活性。多个应用可以向一个接收方发送报文，一个发送方向也可以向多个接收方发送报文。UDP 还可以使用底层网络的广播和组播设施交付报文：UDP 提供的是不可靠交付语义，报文可能丢失、重复或者失序，而发送方是得不到通知的；缺乏流控制；报文模式，当有数据交付的时候，必须制定报文边界。

1. 传输层协议的选定

由于 TCP 和 UDP 的语义及其不同，如果要考虑应用协议所要求的语义，设计者就不能不在面向连接和无连接的传输协议间作出选择。利用面向连接的 TCP 协议，可以大大简化应用协议的设计工作。“由于 UDP 不提供可靠的交付，无连接传输要求应用协议提供可靠性，并在必要时，使用一种称为自适应重传的复杂技术。为现有的应用程序增加自适应重传比较困难，它需要程序员具有相当地专业知识。”

选择 TCP 一个很大的缺点就是，建立和中止连接的三次握手使 TCP 比起 UDP 开销大。“在考虑是用 UDP 还是用 TCP 作为域名服务系统的运输层协议是，设计者往往陷入两难的境地。一个理想的解决方案应该既能提供可靠的数据传输，又不需要专门的建立和释放连接，不需要报文的反段和重组，同时还能是两端的空闲状态所处的时间最短。TCP 什么都好，只可惜他需要建立和释放连接。”^[2]

还有一个严重的问题就是 TCP 在空闲的连接上根本不发送任何分组。假设

客户与某个服务器建立了连接，并与之交换请求和响应，接着便崩溃了。因为客户已经崩溃了，它就不会再发送任何请求了，然而，服务器到目前为止对它收到的所有请求都进行了响应，它便不会再向客户发送更多的数据了。在这种情况下，服务器拥有分配给该连接的数据结构(包括缓存空间)，并且这些资源不能被重新分配。服务器是设计成始终运行的。如果不断有客户崩溃，服务器就会耗尽资源(比如，套接字、缓存空间、TCP 连接)从而中止运行。

2. 3. 3 Socket 原理

在服务器端程序中，Socket 首先通过 socket 函数建立服务器端的 socket，并通过 bind 设置 socket 所使用的服务器端 IP 地址及通信端口。带服务器端 socket 建立之后，则通过侦听表示服务器端应用程序开始侦听客户端的连接。当收到来自客户端的连接请求时，便通过 accept 建立与客户端的连接。

建立服务器 Socket 应用程序步骤如下：

1. 建立服务器端的 Socket，并且以此侦听来自客户端的连接请求。
2. 当服务器段侦测到来自客户端的连接请求时，则接收此请求并建立客户端的 Socket，该 Socket 将作为客户端连接及后续处理发送及接收数据的依据，至此则完成服务器与客户端的 Socket 通信连接，
3. 处理根据客户端的信息，一般称为请求，可视为客户端的指令需求。例如 HTTP 通信协议的 URL 请求，或 FTP 通信协议的 FTP 命令(如 get、put)等。
4. 根据客户端传来的请求，服务器端需经过程序逻辑处理之后，发送回相对应的执行结果或错误信息至客户端，例如 HTTP 服务器须发送回 HTML 网页内容，而 FTP 服务器则发送回 FTP 指令的结果。
5. 当程序完成数据或命令的处理之后，便关闭 Socket 通信链接。

在传递数据上，服务器与客户端之间可通过 read、recv 及 send、write 进行数据的接收和发送。

当连接中断时，程序利用 closesocket 关闭 Socket 并释放系统资源。

2. 3. 4 分析 Socket 在客户端的开发流程

客户端 Socket 应用程序与服务器端 Socket 应用程序的流程很相似，最大的差别在于：

1. 服务器端 Socket 应用程序主要用于侦听及接收客户端的连接，而客户端 Socket 应用程序则用于尝试与服务器端建立连接。
2. 端 Socket 应用程序发送信息指令至服务器端，并接收服务器端所返回的结果；而服务器端 Socket 应用程序则处理指令逻辑，并将结果或错误信息发送

至客户端。

客户端应用程序首先通过 socket 函数建立客户端的 socket，其主要目的是与指令的服务器端(指定主机的 IP 地址及通信端口号)建立连接，此操作由 connect 来完成。当服务器端收到 connect 的连接请求时，则由服务器端的 accept 建立此连接。

建立客户端 Socket 应用程序的步骤大致如下：

1. 客户端的 Socket，在建立时需指定欲连接服务器端的主机名称（或 IP 地址）与 Internet 服务的通信端口。
2. 特定信息或指令至服务器端。
3. 服务器端返回的执行结果或错误信息，并以特定格式显示。例如 HTTP 通信协议会通过 HTML 内容显示。
4. 客户端不需要服务器端的处理时，便关闭 Socket 通信连接。

在数据传递上，可利用 read、recv 及 send、write 接收及发送数据。最后并通过 closesocket 关闭 Socket 连接。

2. 4 J2EE 技术

目前，Java 2 平台有 3 个版本，它们是适用于小型设备和智能卡的 Java 2 平台 Micro 版（Java 2 Platform Micro Edition, J2ME）、适用于桌面系统的 Java 2 平台标准版（Java 2 Platform Standard Edition, J2SE）、适用于创建服务器应用程序和服务的 Java 2 平台企业版（Java 2 Platform Enterprise Edition, J2EE）。

J2EE 是一种利用 Java 2 平台来简化企业解决方案的开发、部署和管理相关的复杂问题的体系结构。J2EE 技术的基础就是核心 Java 平台或 Java 2 平台的标准版，J2EE 不仅巩固了标准版中的许多优点，例如“编写一次、随处运行”的特性、方便存取数据库的 JDBC API、CORBA 技术以及能够在 Internet 应用中保护数据的安全模式等等，同时还提供了对 EJB（Enterprise JavaBeans）、Java Servlets API、JSP（Java Server Pages）以及 XML 技术的全面支持。其最终目的就是成为一个能够使企业开发者大幅缩短投放市场时间的体系结构。

J2EE 体系结构提供中间层集成框架用来满足无需太多费用而又需要高可用性、高可靠性以及可扩展性的应用的需求。通过提供统一的开发平台，J2EE 降低了开发多层应用的费用和复杂性，同时提供对现有应用程序集成强有力支持，完全支持 Enterprise JavaBeans，有良好的向导支持打包和部署应用，添加目录支持，增强了安全机制，提高了性能。

J2EE 规范定义了一个完善的应用组件技术框架，作为企业级应用系统基本

构造模块的组件就是建立在这个框架之上。从简单的 Web 应用到复杂的分布式企业级应用，几乎所有的业务应用，都可以在此技术框架基础上构造。

(1) Servlet

Servlet 是一些用来扩展 Web 服务器功能的 Web 组件，它基于请求/响应机制。Servlet 从客户端(例如 Web 浏览器)获得请求，然后，将响应结果返回客户端。Servlet 的这种特点使它非常适合于 Web 应用。Servlet 和 EJB 组件的区别在于 EJB 组件所提供的服务器端组件特性并不能全部适用于 Servlet,Servlet 更适合于处理简单的请求/响应任务，而且它不需要应用服务器所提供的复杂服务的支持。

(2) JSP

JSP (Java Server Pages)是另一种类型的 Web 组件，它是从 Servlet 技术发展而来的。事实上，JSP 脚本可以编译成 Servlet。两者最大的不同是 JSP 并不是纯 Java 代码，它容许将复杂的 Java 代码嵌入到 HTML 或者 XML 文档中，注重解决客户端的外表和显示问题，JSP 技术可以提供与 Servlet 相同的功能。为了保持 JSP 的简洁性和核心业务逻辑代码的隐蔽性，JSP 技术提供了特殊的标记符，用于调用 JavaBean。组件和标记符库。JavaBean 组件和标记符库提供了一种非常有价值的服务，它使 Web 应用开发人员能够简化 JSP 页面中的 Java 代码，减少了在 JSP 页面的编辑过程中因为偶尔的疏忽而造成整个页面的破坏，容许独立于 JSP 页面而修改表示逻辑。

(3) EJB

EJB (Enterprise JavaBeans)是 J2EE 平台的核心，也是 J2EE 得到业界广泛关注和支 持的主要原因。我们知道，J2EE 的一个主要目标就是简化企业级多层应用系统的开发，使得程序员将主要精力放在业务逻辑的开发上。EJB 正是基于这种思想的服务器端技术，它本身也是一种规范，该规范定义了一个可重用的组件框架来实现分布式的、面向对象的业务逻辑。EJB 的核心思想是将业务逻辑与底层的系统逻辑分开，使开发者只需关心业务逻辑，而由 EJB 容器实现目录服务、事务处理、持久性、安全性等底层系统逻辑。

根据功能的不同，EJB2.0 规范中定义了三种 Enterprise JavaBean：会话 Bean(Session Bean)、实体 Bean(Entity Bean)和消息驱动 Bean(Message-Driven Bean)。

会话 Bean 分为无状态和有状态两种。一般无状态的会话 Bean 模拟商业逻辑，比如计算价格等。有状态的会话 Bean 通常模拟一个客户会话，它会临时保存客户信息，根据客户要求调用其它 Bean 来存取数据。两种会话 Bean 都不保存状态信息或者数据，当客户断开连接或者服务器关闭时，会话 Bean 也就随之消失。一个会话 Bean 的典型例子是网站上的购物车。

实体 Bean 模拟业务数据, 它表示一个数据存储, 可以是状态信息或者数据库中的一条记录。实体 Bean 在客户断开连接或者服务器关闭后, 仍有服务保证其数据得以保存。实体 Bean 按照操作类型分为 Bean 管理的持久实体 Bean(Bean-Manager Persistence, BMP)和容器管理的持久实体 Bean(Container-Manager Persistence, CMP), 两者主要区别是前者由开发人员提供数据访问逻辑的实现方法, 以便在实体 Bean 实例和存储空间之间建立映射, 后者不需要开发者提供任何持久逻辑, 而是由 EJB 容器来处理数据访问。一个实体 Bean 的典型例子就是客户帐号信息。

消息驱动 Bean 在行为上很象会话 Bean。不同的就是仅在需要向这些 Bean 发送消息时才调用消息驱动 Bean, 比如在需要的时候发送用户确认信息等。

一个可部署的 EJB 组件包含三各部分:

Remote 接口 Remote 接口定义 EJB 组件中提供的可供用户调用的方法, 也就是通常所说的实现业务逻辑的函数(比如计算商品价格的函数), 以供远程客户端调用。在 EJB 组件部署到容器的时候, 容器会自动生成 Remote 接口相应的实现, 即 EJB 对象, 它负责代理用户的调用请求。

Home 接口 Home 接口定义一组方法来创建新的 EJB 对象, 查找、定位和清除已有的 EJB 对象。在 EJB 组件部署时容器也会自动生成相应的 Home 对象, 该对象负责查找和创建 EJB 对象, 返回 EJB 对象的引用给客户端, 用户利用该引用调用 EJB 组件的方法, 从而得到结果, 最后, Home 对象清除 EJB 对象。我们可以形象地称 Home 接口为 EJB 对象的工厂。

Bean 类是业务逻辑的具体实现类。其可供用户调用的方法在 Remote 接口中定义。

另外, 在提交和部署 EJB 组件时, 还需要 EJB 部署描述文件, EJB 部署描述文件是标准的 XML 文件。部署描述文件至少由 ejb 描述文件和 ejb 部署文件这两个文件组成, 前者用来描述 EJB 自身的一些特征, 比如名称、结构、类型和环境参数等, 后者是和部署相关的描述。

EJB 容器非常复杂, 一般由专业的 J2EE 应用服务器开发商提供, 比较流行的 EJB 容器有由 BEA 公司的 Weblogic Server, IBM 公司的 WebSphere, Sun 公司的 Planet 等。EJB 容器除了为 EJB 提供事务处理、目录服务、持久性管理和安全服务外, 还负责 EJB 的部署、发布和生命周期管理。

(4)Java 数据库连接(JDBC)

JDBC 是访问关系数据库的 API, JDBC 的价值在于允许你访问任何关系型数据库都使用相同的 API。利用 JDBC API 和不同的数据库驱动程序就可以访问不同的数据库。JDBC 驱动程序共有四种类型:Type1, Type2, Type3, Type4。

Type1: JDBC-ODBC 桥, 在 JDBC 刚产生时, JDBC-ODBC 桥是非常有用的。通过它, 开发者可以使用 JDBC 来访问一个 ODBC 数据源。这种方式的缺点是, 它需要在客户机器上安装有一个 ODBC 驱动, 这样就失去了 JDBC 平台无关的好处。此外, ODBC 驱动器需要客户端的管理。

Type2: JDBC-native 驱动桥, JDBC-native 驱动桥提供了一个建筑在本地数据库驱动上的 JDBC 接口, 而没有使用 ODBC。JDBC 驱动将标准的 JDBC 调用转变为对数据库 API 的本地调用。使用类型 2 的驱动也会失去 JDBC 平台无关的好处, 并且, 需要在客户端安装数据库的本地驱动程序。

Type3: JDBC-network 桥, JDBC-network 桥不需要客户端的数据库驱动, 而是通过网络协议访问中间服务器, 由中间服务器访问数据库。这会引起诸如负载均衡、连接池等技术, 数据缓冲也是可能的。由于类型 3 的驱动通常可以带来相对小的下载时间, 它是平台无关的, 并且不需要客户端的安装和管理, 因此很适合作 Internet 应用。

Type4: 纯 Java 驱动, Type4 使用纯 Java 数据库驱动来提供直接的数据库访问。用来代替类型 2 驱动程序, 类型 4 驱动程序使用厂商专用的网络协议把 JDBC API 调用转换成直接网络调用, 它们这样做是通过与数据库建立直接的套接字连接。类型 4 提供的性能一般要优于类型 1 和类型 2 驱动程序。类型 4 驱动程序也是应用中最简单的驱动程序, 因为不需要安装其它库或者中间件。所有的主要数据库厂商都为他们的数据库提供了类型 4 JDBC 驱动程序, 也可以从第三方厂商获得这些驱动程序。

(5) Java 命名目录服务(JNDI)

JNDI 设计是为了简化在开发高级网络程序中对目录基础设施的访问。目录是一种特殊的数据库, 提供了对其数据存储的快速访问, 目录数据库是以一种读取优化(read-optimized)的层次结构来存储信息。传统上, 我们需要使用不同的 API 来访问不同的目录服务, 如 Sun 公司的 Networ Information Service(网络信息服务, NIS)或者 Lightweight Directory Access Protocol(轻量级目录访问协议, LDAP)。但是 JNDI 提供了一个标准的 API 来访问任何类型的目录。JNDI 还为我们提供了在网络上存储和检索 Java 对象的能力。

(6) Java 事务处理 API(JTA)和事务处理服务(JTS)

JTA (Java Transaction Architecture)定义了一组标准的接口, 为应用系统提供可靠的事务处理支持。JTS (Java Transaction Service)是 CORBA OTS 事务监控的 Java 实现。JTS 规定了事务管理器的实现方式, 该事务管理器在高层支持 JTA 标准, 在底层实现了 OMG OTS 规范的 Java 映射。

(7) Java 消息服务(JMS)

JMS (Java Message Service)是一组用于和面向消息的中间件相互通信的 API。它既支持点对点的消息,也支持发布/订阅式的消息通信。

(8)JavaMail

JavaMail API 允许在应用程序中以独立于平台、独立于协议的方式收发电子邮件。JavaMail 依赖于 JavaBeans 触发框架(JavaBeans ActivationFramework, JAF)来负责处理 MIME 编码,JavaMail 利用 JAF 来处理 MIME 编码的邮件附件。

(9)Java 验证和授权服务(JAAS)

JAAS (Java Authentication and Authorization Service)用两个步骤实现安全性。认证,即由用户提供认证信息(如用户名和密码)来取得系统认证,这一过程又称为登陆:授权,在被确认为合法用户后,系统根据用户的角色授予其相应的权限。J2EE 的授权是基于安全角色的概念,一个安全角色是一个拥有相同权限的逻辑组。J2EE 的安全角色由应用组件提供商来定义。

(10)J2EE 连接器架构(JCA)

JCA (Java Connector Architecture)API 使用户可以从一个 J2EE 部署访问现有企业信息系统。这些现有系统包括运行着高端事务处理的大型机系统,企业资源计划系统(ERP)或者用户自己的专有系统。

(11)JAXP

JAXP (The Java API for XML Parsing)是分析 XML 文档的 API,而且它也是一个 XML 分析器的中性实现接口。通常用户都是从内部的 Servlet, JSP 或者 EJB 组件中使用 JAXP API。

(12)Java 远程方法调用(Java Remote Method Invocation, RMI)和 RMI-IIOP

远程方法调用是分布式对象应用程序中的一种主要机制。它允许我们使用接口来定义远程对象。我们可以接着调用这些远程对象上的方法,就象它们在本地一样。真正的线级(wire-level)传输机制是由具体的实现确定。例如, Sun 公司是在 TCP/IP 的基础上使用 Java Remote Method Protocol (Java 远程方法协议, JRMP),而其它实现形式如 BEA Weblogic,则有它们自己的协议。

RMI-IIOP 是 RMI 的扩充,但它是在 HOP(Inter-ORB Protocol)上实现,可提供我们定义一个通向任何远程对象的远程接口,这些对象可以用任何支持 OMG 映射和 ORB 的语言实现。

(13)JavaIDL

JavaIDL 是 Sun 公司基于 Java 的对 CORBA 规范的实现,JavaIDL 允许与其它语言集成,而且它能够让分布式对象利用 CORBA 提供的全面服务。

虽然 J2EE 还不能够解决 P2P 中所遗留得所有问题,但它已经为 P2P 的开发提供了很有吸引力的平台。

除了以上列举的,与 P2P 相关技术还有很多,它们有时也被称作 P2P,或者

与 P2P 相关，或者可以被当作 P2P 使用，主要是以下几方面内容：

- 1、软件代理(Agent)技术；
- 2、JXTA ；
- 3、Web Services 技术；

第3章 系统需求

3.1 系统目标

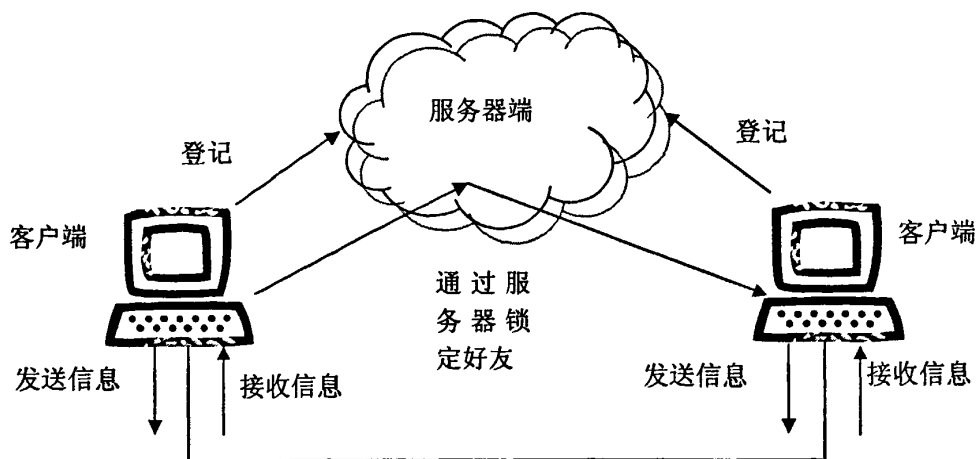


图 3.1 简单的即时通信

系统要实现的即时通信系统是一个简单、方便的通信系统，如图 3.1，用户通过网上进行交互。此即时通信系统要可以方便用户之间进行交流，信息即时发送，即时回复。这里的信息可以是文字，图像，视频和文件。

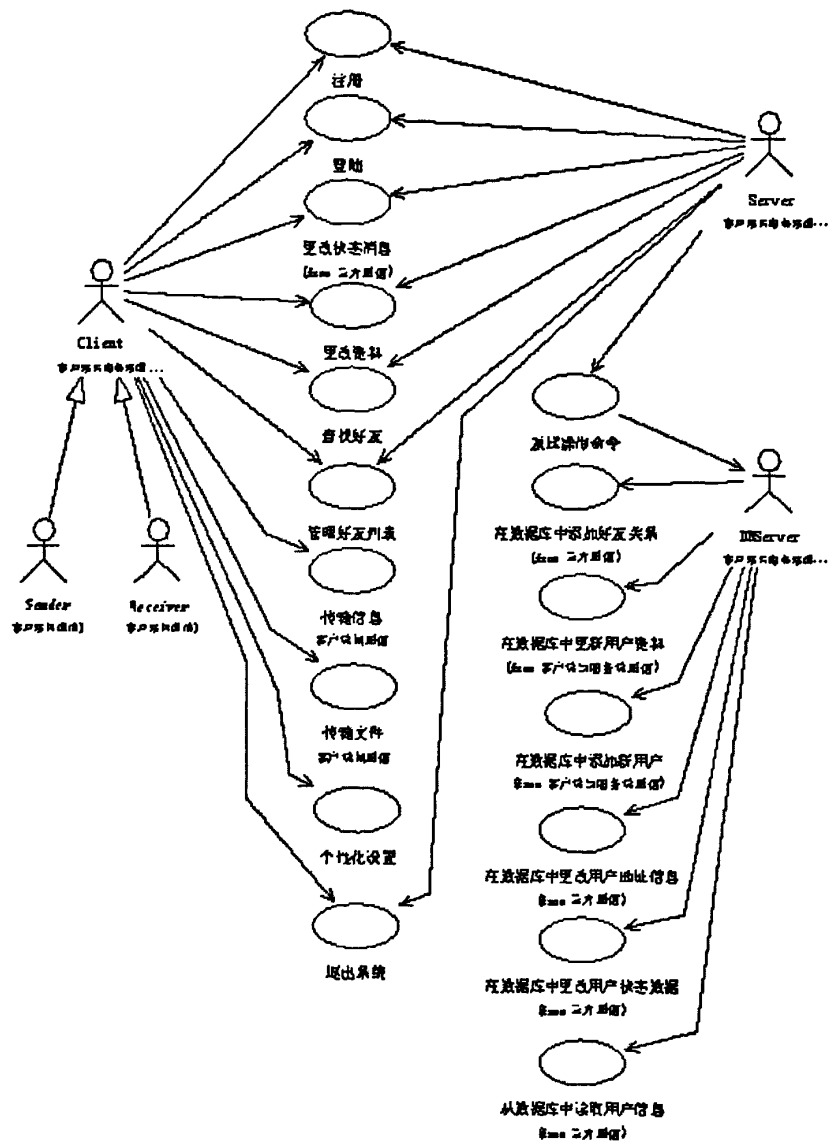
3.2 系统角色

- 1) Client: 客户端，使用进行即时通信的人
 - 2) MessageSender: 端对端传输消息或发送文件时的发送端
 - 3) MessageReceiver: 端对端传输消息或者文件时的接收端
- 其中，MessageSender 和 MessageReceiver 从属于 Client

3.3 系统用例概况

根据需求分析，开始系统的图形化建模。用例技术通过用例、执行者和用例以及用例之间的关系来描绘系统外在可见的需求情况，它是用户和开发者共同剖析系统功能的起点。本文的用例图均使用 UML 统一建模语言和 Rational

Rose2003 建模工具设计，图形如下：



3.2 系统用例图

即时通信系统用例由以下组成。

- 1) 注册用例是使用者通过注册成为用户的。
- 2) 登录：连接服务器获得服务；
- 3) 更改状态：更改自己在服务器的状态，状态分为：在线、隐身、离线三种；
- 4) 更改资料：更改自己的资料并保存在数据库服务器；
- 5) 查找好友：从服务器获得好友信息；

- 6) 管理好友列表：对好友进行添加、删除、分组等操作；
- 7) 个性化配置：对进行本地化配置；
- 8) 传输信息：传送文本信息或者富文本信息给好友；
- 9) 退出系统：从服务器退出，取消服务；

3.4 用户注册用例

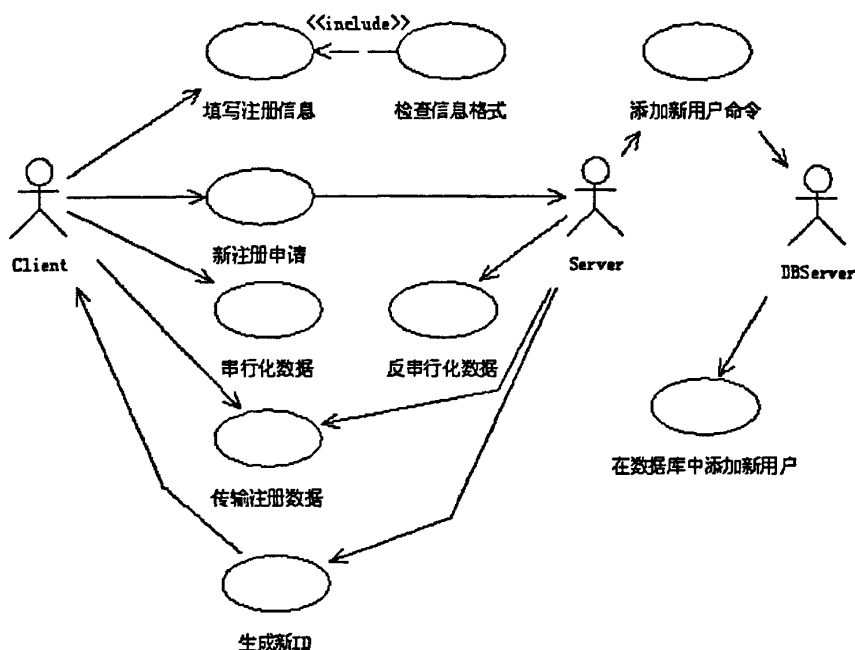


图 3.3 注册用例图

注册用例图用例说明：

- 1) 填写注册信息
- 2) 检查信息格式
- 3) 新注册申请
- 4) 串行化数据
- 5) 传输注册数据
- 6) 生成新 ID
- 7) 添加新用户命令
- 8) 在数据库中添加新用户

3.5 用户登录用例

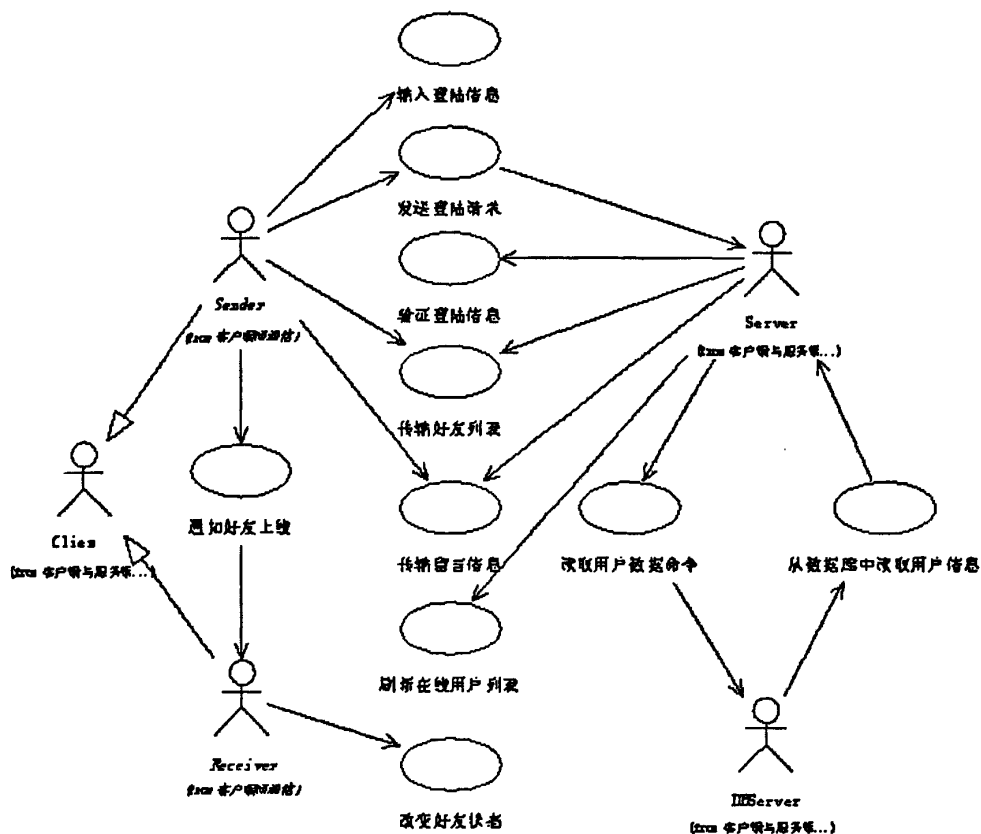


图 3.4 登录用例图

登录用例图用例说明：

- 1) 输入登录信息
- 2) 发送登录请求
- 3) 验证登录信息
- 4) 传输好友列表
- 5) 传输留言信息
- 6) 刷新在线用户列表
- 7) 通知好友上线
- 8) 改变好友状态
- 9) 读取用户数据命令
- 10) 从数据库中读取用户信息

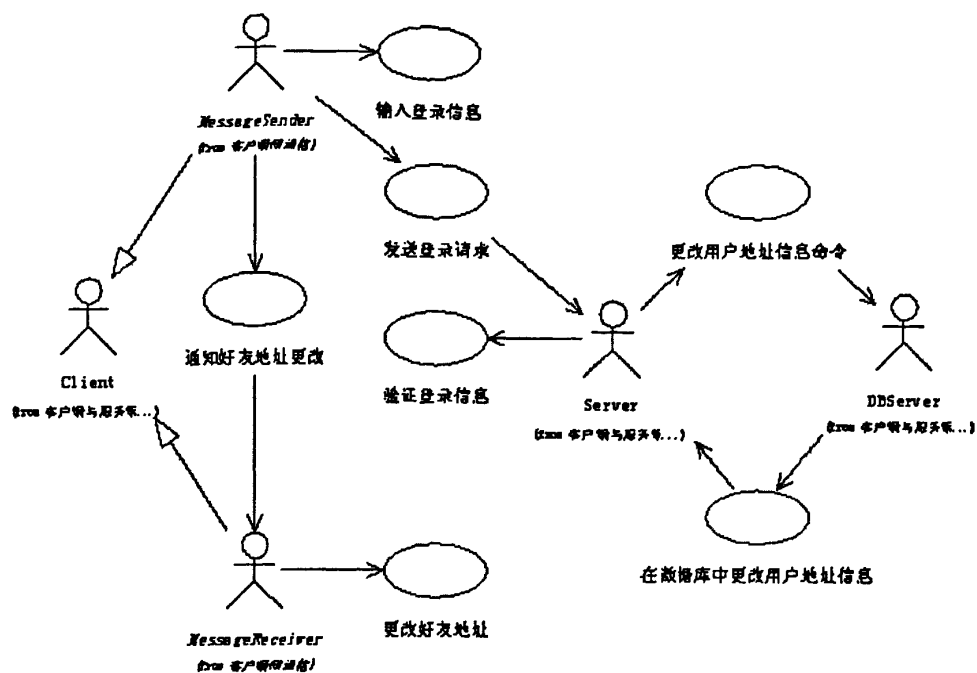


图 3.5 用户重复登陆用例图

重复登录用例图用例说明：

- 1) 输入登录信息
- 2) 发送登录请求
- 3) 验证登录信息
- 4) 通知好友地址更改
- 5) 更改好友地址
- 6) 更改用户地址信息命令
- 7) 在数据库中更改用户地址信息

更改状态用例图

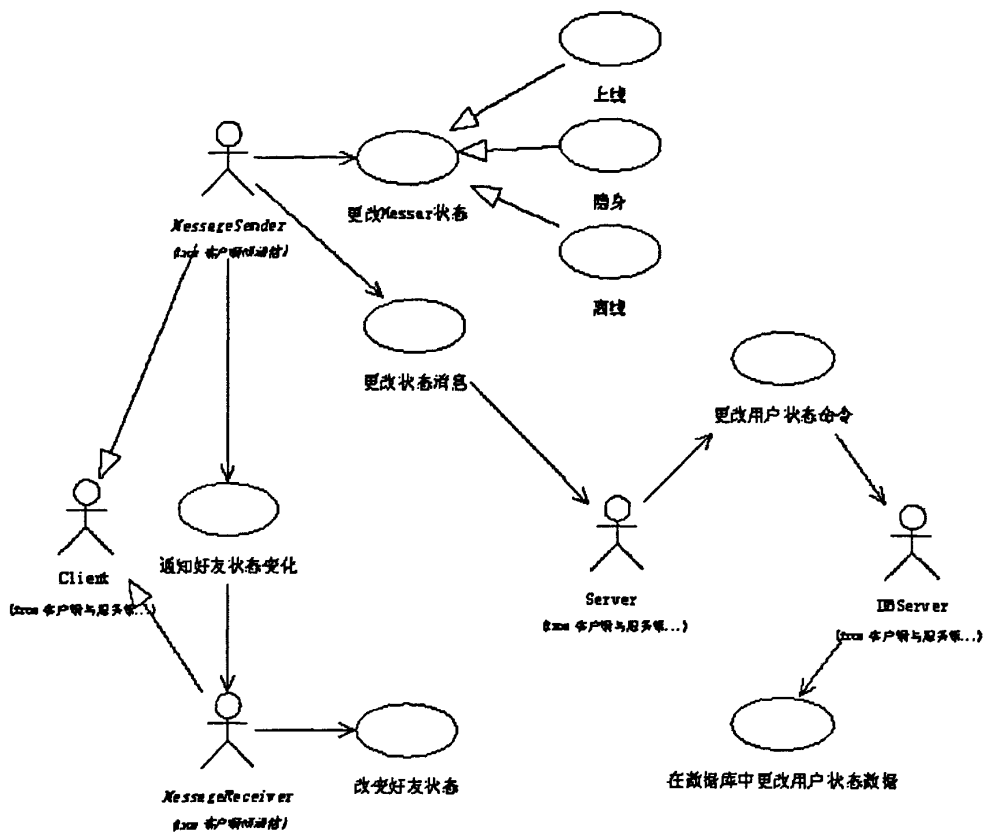


图 3.6 更改状态用例图

更改状态用例图用例说明:

- 1) 更改状态
- 2) 上线
- 3) 隐身
- 4) 离线
- 5) 更改状态消息
- 6) 通知好友状态变化
- 7) 更改好友状态
- 8) 更改用户状态命令
- 9) 在数据库中更改用户状态数据

更改资料用例图

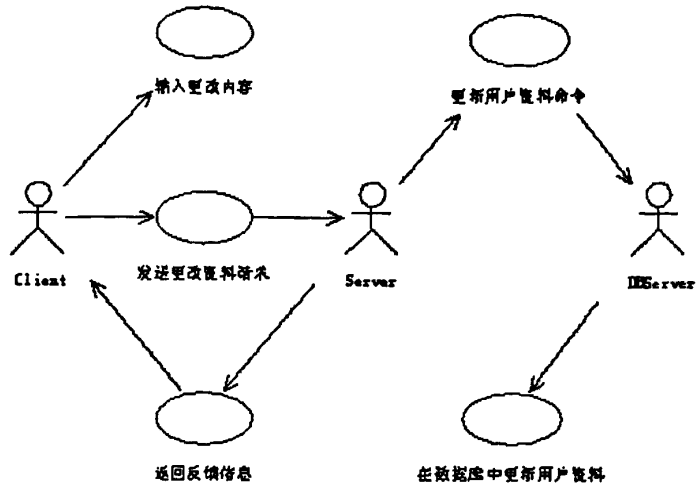


图 3.7 更改资料用例图

更改资料用例图用例说明

- 1) 输入更改内容
- 2) 发送更改资料请求
- 3) 返回反馈信息
- 4) 更新用户资料命令
- 5) 在数据库中更新用户资料

3.6 建立客户通信用例

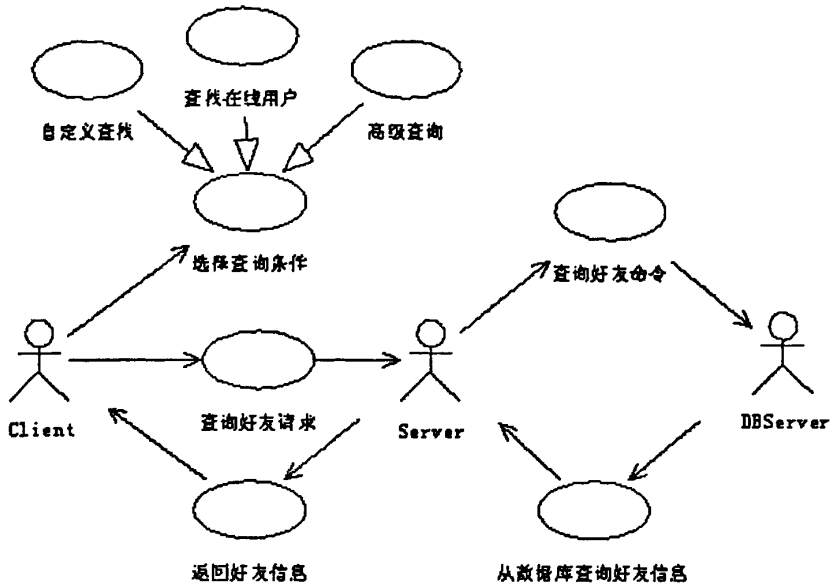


图 3.8 建立客户通信用例图

建立客户通信用例用例图用例说明：

- 1) 自定义查找
- 2) 查找在线用户
- 3) 高级查询
- 4) 选择查询条件
- 5) 查询好友请求
- 6) 返回好友信息
- 7) 查询好友命令
- 8) 从数据库查询好友信息

3.7 客户信息传输用例

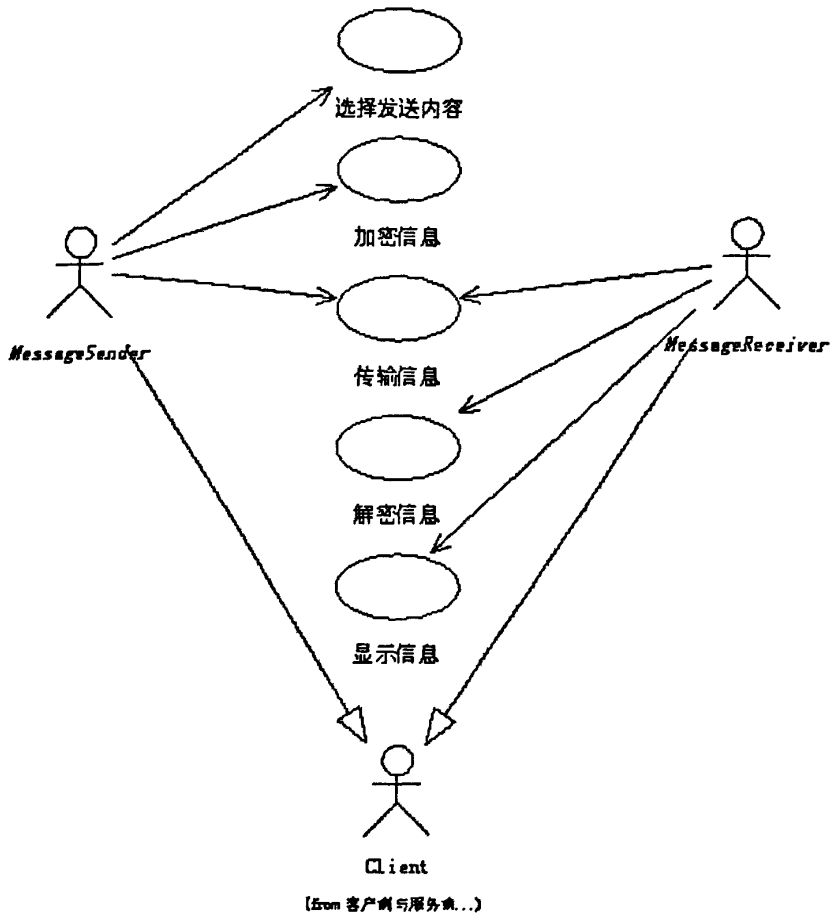


图 3.9 客户信息传输用例图

客户信息传输用例说明：

- 1) 选择发送内容
- 2) 加密信息
- 3) 传输信息
- 4) 解密信息
- 5) 显示信息

退出系统用例图

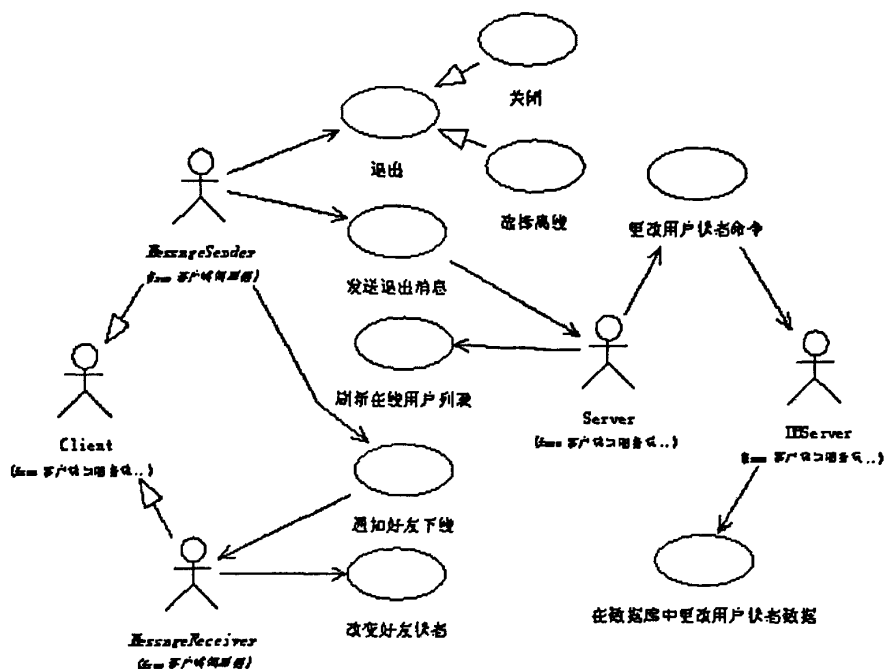


图 3.10 退出系统用例图

退出系统用例

- 1) 退出
- 2) 发送退出消息
- 3) 刷新在线用户列表
- 4) 通知好友下线
- 5) 改变好友状态
- 6) 选择离线
- 7) 关闭
- 8) 更改用户状态命令
- 9) 在数据库中更改用户状态数据

第4章 系统设计

4.1 系统体系构架

本系统的体系构架如图 4.1 所示。有三部分组成：通信服务器、通信客户端、数据库。

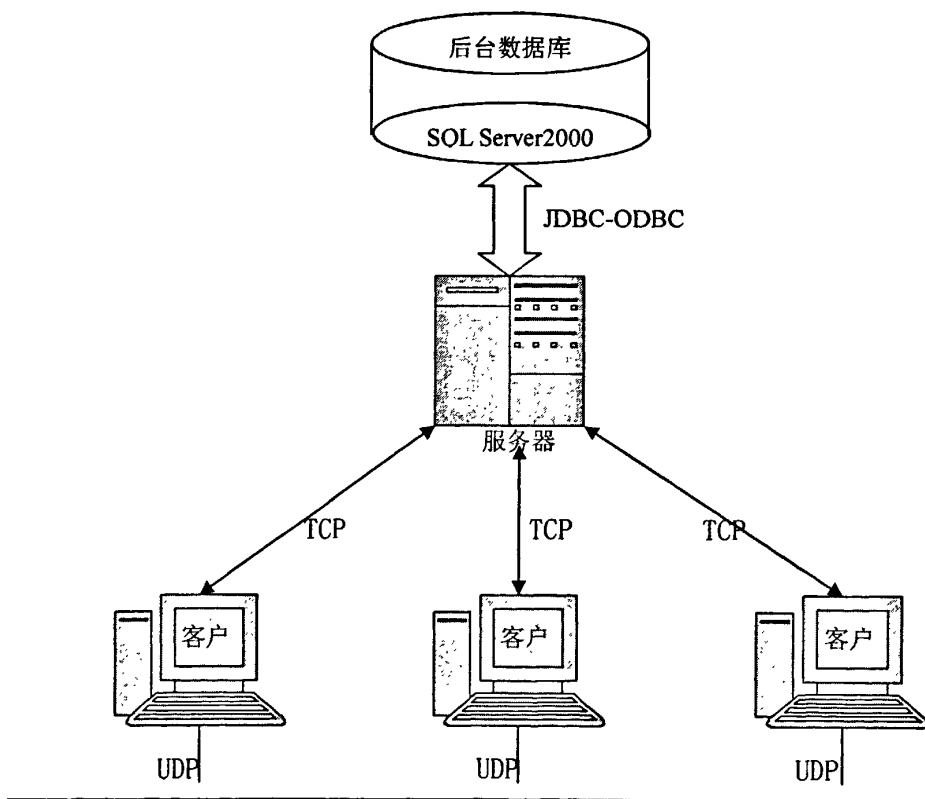


图 4.1 系统体系构架

通信服务器：其主要功能是响应客户端的请求，与后端数据库相联结完成用户的注册，登录等请求；

数据库：记录所有用户的登录，注册等信息。

客户端：与通讯服务器建立关联，找到 P2P 的对等点，进行数据的传送。

本系统模式采用的是 P2P 模式，客户与服务器之间用 TCP 协议连接。

客户端同服务器端的连接采用 TCP 协议，见图 4.2 所示，目的是保证保持通信的持续性。

由于客户端与服务器的连接要与数据库相连，必须确保传输准确无误，而且它们之间还要进行互相通信，如验证用户信息和读取好友资料，所以必须要用传输准确的 TCP 协议。客户端与客户端的连接就比较简单，强调快速，只要把信息发送过去就可以，所以就不须先建立连接的 TCP 协议，这样节省了 TCP 建立连接所需的时间。

其中 TCP 在传输数据之前，会先在主机间(例如服务器端和客户端)建立通信链接，通过此通信链接，数据可在计算机间传输。

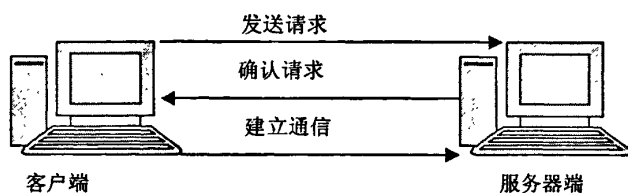


图 4.2 客户端同服务器端通信协议

客户与客户之间用 UDP 协议连接，UDP 不提供数据错误的侦测以及数据重送等功能，因此不能确保数据能完整地发送。

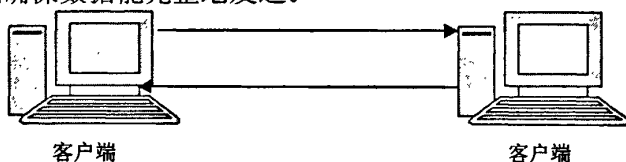


图 4.3 客户端与客户端通信协议

4.2 消息设计

4.2.1 服务器与客户消息

消息主要有侦听消息，请求消息：

为了使能及时响应用户的请求，当用户很多时，仍然能够适应要求，我把侦听与发送数据的 Socket 分开，并分别都建立了多个实例，也就是说，支持多个端口的侦听，发送数据使用的是多个端口，我只对侦听端口感兴趣，对发送数据的端口不感兴趣，因为，发送端口是多少都无所谓。

CRecvSocket 和 CSendSocket 都是从 CAsyncSocket 类里继承而来，分别处理侦听请求各发送数据，在 CServerSocket 类里，定义了几个 CRecvSocket 和 CSendSocket 对象的实例，

通过 CServerSocket 类对内部进行组织和管理, 提供给上层的接口是 CServerSocket, 它隐藏了服务器底层通讯的细节及多线程发送数据的问题, 提供给上一层一个统一的接口, CServerSocket 类的使用, 是先建立一个它的实例, 再调用成员函数 Create(), 传入必要的参数, 发送数据时, 就调用其成员函数 SendData, 处理接受数据, 在 CRecvSocket 类的 OnReceive 里处理, 调用了名为 ProcessRecvData 的线程函数, 用户在这个线程函数里写上具体的处理代码。

消息数据格式和对象

1. 侦听类结构

```
class CRecvSocket : public CAsyncSocket
{
private:
    char m_szResponseMsg[MaxResponseMsgLength];    // 确认消息串
    int m_szrLength;                                // 确认消息串的长度
public:
    CRecvSocket();
    virtual ~CRecvSocket();
public:
    BOOL Create(int nPort);                          // 指定端口号, 建立一个侦听 Socket
    virtual void OnReceive(int nErrorCode);           // 处理接受数据
};
```

2. 发送数据的 socket 类结构

```
class CSendSocket : public CAsyncSocket
{
private:
    BYTE* pBuf;
    // 指向 CServerSocket 的 m_arBuf 缓冲区, 处理发送后是否收到确认信息的缓冲区
    char m_szResponseMsg[20];
    int m_szrLength;
public:
    CSendSocket(BYTE*buf);
    virtual ~CSendSocket();
    BOOL Create();                                    // 建立一个发送的 socket
```

```
virtual void OnReceive(int nErrorCode); // 处理发送数据后接受到的
确认消息
};
```

3. 服务器的 socket 类

这个通讯类的 SendData, 当发送数据失败时, 可以重发几次, 次数可由用户来确定

```
class CServerSocket
{
public:
    void Close();
    BOOL Create(int SendSockNum, CArray<DWORD, DWORD>&aPort, int TimeOut=
TimeWaitForRes);
//建立侦听 Socket 和发送 Socket, aPort 是侦听端口数组, TimeOut 是设置发
送超时时间
    BOOL SendData(CData* pData, int FailReDoTimes=3);
//数据函数, 这个函数适合在线程里发送数据函数, 而服务器的响应都是在工人
线程里完成的, 所以, 很适合服务器的发送数据。成功返回真, 不成功返回假
    CServerSocket();
    virtual ~CServerSocket();
private:
    CRecvSocket* m_apRecvSocket[ListenSocketNum]; //侦听 Socket 的指
针数组
    CSendSocket* m_apSendSocket[SendSocketNum]; // 发送 Socket 的指针
数组
    int m_nRecvSocket; //侦听 Socket 的个数
    int m_nSendSocket; // 发送 Socket 的个数
    BOOL m_abBusy[SendSocketNum]; // 标志每个发送 Socket 是否处
于忙的状态
    BYTE m_arBuf[CheckBufLength]; //检查发送数据发回的确认的缓冲数组
    DWORD m_nTotalSend; //保存从 CserverSocket 建立后, 发送了多
少次数据
    DWORD m_nTimeOut; // 发送数据的超时时间
};
```

4.2.2 客户与客户消息

主要有两种消息发送数据，和接收数据，详述如下：

与客户端上层的接口是 CClientSocket 类，它隐藏了服务器底层通讯的细节及多线程发送数据的问题，提供给上一层一个统一的接口，CClientSocket 类的使用，是先建立一个它的实例，再调用成员函数 Create() 传入必要的参数，发送数据时，就调用其成员函数 SendData，或 SendDataInThread 处理发送数据，在 CRecvSocket 类的 OnReceive 里处理，向父窗口发送一个 WM_RECIEVE_MSG 消息，并把接受到的数据作为参数传递给父窗口。两个函数的适应情况，SendData 函数，适用于需要直接发送数据的场合，不需要回应。如果在线程里执行，则可由其返回值确定发送成功与否。而 SendDataInThread 是建立一个线程，在线程里调用 SendData 函数进行发送数据，通过向指定接受窗口发送消息来确定是否成功。

消息数据格式和对象

```
class CClientSocket : public CAsyncSocket
{
public:
    CClientSocket();
    virtual ~CClientSocket();
    void SendDataInThread(CData* pData, CWnd*pWnd);
    // 在线程里发送数据，成功，或失败都会向指定窗口类发送一个
    WM_SENDINTHREAD_RES 的消息，参数 WPARAM 为发送数据的指针，参数 LPARAM
    为 1，则表示发送成功，0 则表示发送失败
    BOOL Create(int TimeOut=TimeWaitForRes); //建立 SOCKET
    BOOL SendData(CData* pData, int FailReDoTimes=3);
    //发送数据成员函数，失败重试指定的 FailReDoTimes 次，成功返回真
    CWnd* GetOwner() {return m_pWnd;};
    //得到接受发来数据的处理窗口指针
    接受发来数据的处理窗口指针
    virtual void OnReceive(int nErrorCode);处理 FD_READ 网络事件
private:
    CWnd* m_pWnd; //接受发来数据的处理窗口指针
    char m_szResponseMsg[MaxResponseMsgLength];
```

```

int m_szsLength;
BOOL m_bBusy; //是否处理忙状态
BYTE m_arBuf[CheckBufLength]; //检查确认信息的缓冲数组
DWORD m_nTotalSend; //从启动到现在为此，发送的数据总数
DWORD m_nTimeOut; //发送数据的超时时间
};

```

4.3 系统功能模块设计

本系统 P2P 协议的实现服务器端和客户端的实现完成的，任何即时通信系统不外乎由服务器端和客户端两部分组成，其中服务器端得实现主要是通过完成对数据库的操作、监听客户端、和建立连接三个部分既可，客户端的实现则要完成用户的新建、用户的登陆、查找好友和即时通信功能，这样就基本完成了系统功能。

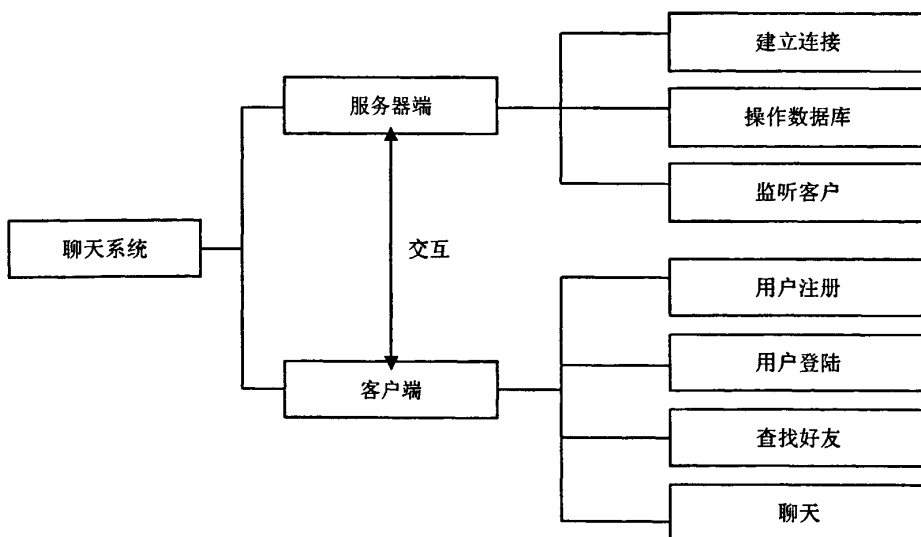


图 4.4 系统的功能模块

4.3.1 服务器端模块设计

4.3.1.1 TCP 协议层

CClientSocket : 实现建立连接，绑定等功能

CServerSocket : 负责监听客户端连接，实现并发处理






CSendSocket: 实现客户端发送数据
CRecvSocket : 实现客户端接受数据

程序中, Socket 首先通过 socket 函数建立服务期端的 socket, 并通过 bind 设置 socket 所使用的服务器 IP 地址和通信端接口。待服务器端 socket 建立之后, 则通过侦听表示服务器应用程序开始侦听客户端的连接。当收到来自客户端的连接请求时, 便通过 accept 建立与客户端的连接。

4.3.1.2 数据持久层设计

在本即时通信系统中数据库是采用 SQL2000, 服务器数据库设计的要求是要能够满足客户端的需求, 保存用户信息和用户好友信息, 提供离线消息的服务, 和发广播消息的服务等。总共有五个表: 用户信息表 (Users) 好友信息表 (Friends) 广播消息表 (Broadcast) 离线广播表 (OffBroadcast) 离线消息表 (OffMsg)。

表 4.1 基本表

	broadcast
	Friends
	OffBroadcast
	OffMsg
	Users

1、users 用户信息表

用户信息表用于记录用户基本信息。

表 4.2 User 用户信息表

	字段名称	数据类型
①	UserId	自动编号
	Id	数字
	PhotoId	数字
	Password	文本
	Name	文本
	Sex	数字
	Age	文本
	CanbeAdd	数字
	Email	文本
	HomePage	文本
	Address	文本
	Phone	文本
	Fax	文本
	Department	文本
	Description	文本

- 1) UserId: 自动编号; 4 字节长整形 (主键)
- 2) Id: 用户帐号; 4 字节长整形
- 3) Photoid: 用户的图象编号; 4 字节长整形
- 4) Password: 用户登陆的密码; 字符串
- 5) Name: 用户的姓名; 字符串
- 6) Sex: 用户的性别; 单字节整形 0 男 1 女 2 未知
- 7) Age: 用户的年龄; 字符串 (为了适应不愿填写此项的人)
- 8) Canbeadd: 能否被人加为好友;
- 9) Email: 电子信箱; 字符串
- 10) Homepage: 个人主页; 字符串
- 11) Address: 地址; 字符串
- 12) Phone: 电话; 字符串
- 13) Fax: 传真; 字符串
- 14) Department: 部门; 字符串
- 15) Description: 个人简介; 字符串

2、 friends 好友信息表

表 4.3 friends 好友信息表

	字段名称	数据类型
①	Num	自动编号
	MyId	数字
	FriendId	数字

- 1) Num: 自动编号 4 字节长整形 (主键)
- 2) MyId: 自己帐号 4 字节长整形
- 3) FriendId: 朋友帐号 4 字节长整形

3、 broadcast 广播消息表

表 4.4 broadcast 广播消息表

	字段名称	数据类型
▽	MsgId	自动编号
	SendTime	数字
	SenderId	数字
	Msg	备注

- 1) MsgId 自动编号 4 字节长整形 （主键）
 - 2) SendTime 发送时间 4 字节长整形
 - 3) SenderId 发送者的帐号 4 字节长整形
 - 4) Msg 发送的消息 备注类型
- 4、 OffBroadcast 离线广播表

表 4.5 OffBroadcast 离线广播表

	字段名称	数据类型
▽	Num	自动编号
	RecvId	数字
	MsgId	数字

- 1) Num 自动编号 4 字节长整形 （主键）
 - 2) RecvId 接受者的帐号 4 字节长整形
 - 3) MsgId 广播消息号 4 字节长整形 （对应广播消息表的 MsgId）
- 5、 OffMsg 离线消息表

表 4.6 OffMsg 离线消息表

	字段名称	数据类型
▽	MsgId	自动编号
	SenderId	数字
	RecvId	数字
	RecvTime	数字
	nIndex	数字
	Msg	备注

- 1) MsgId 自动编号 4 字节长整形 （主键）
- 2) RecvId 接受者的帐号 4 字节长整形
- 3) SenderId 发送者的帐号 4 字节长整形
- 4) RecvTime 接受的时间 4 字节长整形
- 5) nIndex 发送消息的类型 4 字节长整形
- 6) Msg 发送的消息 备注类型

4.3.1.3 服务器功能层设计

服务器端主要完成建立连接、操作数据库、监听客户的功能，其具体含义是，首先服务器端建立一个 ServerSocket 的连接，不断侦听是否有客户端要和服务器端连接或者断开连接，当有客户断要与服务器建立连接服务器立即创建一个新的线程与客户端建立连接，然后再对数据库做出相应的操作最后把结果返回给客户端，服务器端对数据库的操作包括录入用户信息、修改用户信息、查找好友数据库的资料以及添加好友到数据库等等，总之对数据库的操作也就是对数据库进行增、删、改、查操作。

(1) 用户注册

服务器收到用户的注册请求，便开始接受客户传递的信息，诸如客户的昵称，性别，籍贯，个人资料等，接受完毕后，便通过 Jdbc-Odbc 与后台数据库连接，然后向数据库添加记录，如果成功，便向客户返回其号码，并在数据库中注册用户的 IP 地址，然后更新其 Status 为 1 即用户在线。客户收到服务器返回的信息后，便打开主程序窗口，并同时开始创建 UDP 以便在用户之间建立联系。

(2) 用户登录

在客户端，用户输入其号码和密码，然后建立与服务器的连接，告诉服务器我要登录，服务器收到后，开始通过 JdbcOdbc 读取数据库，然后与用户输入的信息比较，如果相同就向客户返回成功消息并将其 Status 字段设为 1 表示上线了以及注册其 IP 地址，否则返回错误，如果客户收到成功信息就打开主窗口，否则提示出错。如果成功，便打开主程序窗口，并同时开始创建 UDP 以便在用户之间建立联系。然后客户向服务器请求读取好友名单，服务器收到该请求，开始读取数据库中的 friend 表，得到好友的号码后，再在 icq 表中读取好友资料，然后向客户端发送这些信息，客户收到后就在主窗口显示好友，并且建立几个矢量 (Vector) 用以存储好友的昵称，号码，ip 地址等信息。

4.3.2 客户端 (P2P 端) 模块设计

4.3.2.1 客户端功能层设计

客户端主要完成四大功能：新建用户、用户登陆（应由服务端实现，尽管客户端提供界面）、查找好友和即时通信功能。客户端申请与服务器端建立连接，当客户断与服务器端建立连接通道后就可以向服务器端发送新建用户信息和登陆信息，还有客户端还支持查找好友和完成信息的编辑、发送和

接受功能。

1、好友管理功能

用户登录后,按查找按钮后,开始向服务器发出查找请求,服务器读取数据库表 icq 并向客户返回其结果,客户收到后在查找窗口中显示,如果用户选择了一个好友,就向服务器发送添加好友请求,服务器收到后就向数据库表 friend 中添加自己的号码以及好友的号码,并从 icq 表中读取其基本信息返回给客户端,然后客户收到并在主窗口显示该好友。并且通过 UDP 通知该客户,对方收到该消息后,可以选择添加该用户为好友或者不。

用户在其好友列表中选择要删除的好友并按删除,然后向服务器发送删除请求,服务器收到该请求后,连接数据库表 friend 删除用户及该好友的记录,如果成功就向客户返回成功消息,客户收到后在其好友列表中删除该好友。

2、收发信息功能

收发信息是本即时通信系统的核心部分,也是本系统网络传输的体现所在。

两个客户端可以通过 UDP 协议直接通信,不管你的好友在不在线都可以通过 UDP 协议给他(她)发送消息,同时也可以接受好友发给你的消息。

4.3.2.2 客户端 P2P 协议设计

客户端的应用程序与服务器端 Socket 应用程序的流程很相似,最大的区别在于:

1. 服务器端 Socket 应用程序主要用于侦听及接收客户端的连接,而客户端 Socket 应用程序则用于尝试与服务器端建立连接。

2. 客户端 Socket 应用程序发送信息指令至服务器端,并接收服务器端返回的结果;而服务器端 Socket 应用程序则处理指令逻辑,并将结果或错误信息发送至客户端。而 UDP 为无连接的通信协议,其主要目的在于处理传输少量的数据。与 TCP 不同的是,UDP 在传输数据之前不需要建立通信链接。仅须设置计算机间的 IP 及使用相同的端口,即可相互传输信息,因此 UDP 只提供单向的数据传输。

由于本系统最主要的部分在网络传输上,所以网络传输协议就成了本设计的重中之重,设计好网络传输也就显的很重要,一定要弄清楚网络协议在此程序中原理。

其中用到了 Socket,它是介于应用程序与硬件之间,并可以提供标准的函数以符合不同的网络硬件规格。Socket 也是他们相互传输的数据结构。

4.3.2.3 P2P 模块算法描述

发送一个数据,需要等待响应信息的回来,如果在规定时间内,还没有收到确认信息,则认为发送数据丢失,将重试 FailReDoTime 次,如果还是没有确认信息发送回来,则返回发送失败,否则,返回发送成功。

因为客户端可以多线程的发送数据,有一个请求,就建立一个线程进行处理。我为每个发送 socket 设置一个是否忙的标志 busy,当需要发送数据时,就选择一个空闲的 sendsocket,设置为忙,然后发送数据,再设置回空闲,然后等待确认信息的回来。

SendData 函数的实现算法,与服务器端是基本一样的原理,通过设置一个缓冲区长度为 N,然后,为每一次数据进行统计,发一次,就加 1,然后把发送数据中的 $This = count \% N$,且把缓冲区中第 This 个成员设为 0,在发送端,就要数组中的第 This 个成员是否为 1 即可,在接受到的确认信息中,取出 This 项,再为缓冲区中第 This 项设为 1,这样就可以快速且可靠的判断发送数据是否得到响应回来了。

因为在客户端,大部分数据,是在某个消息处理函数中执行的,所以,适用于服务器的 SendData 发送数据函数,在客户端,若是在消息处理函数中发送,函数返回值,将永远为 FALSE,于是就添加了一个 SendDataInThread 函数,把发送数据的过程放在线程里执行,通过发消息的手段,来返回结果。在 SendDataInThread 函数中,调用 SendData 函数进行发送数据。

第5章 系统功能实现

5.1 服务端功能模块实现

5.1.1 用户注册

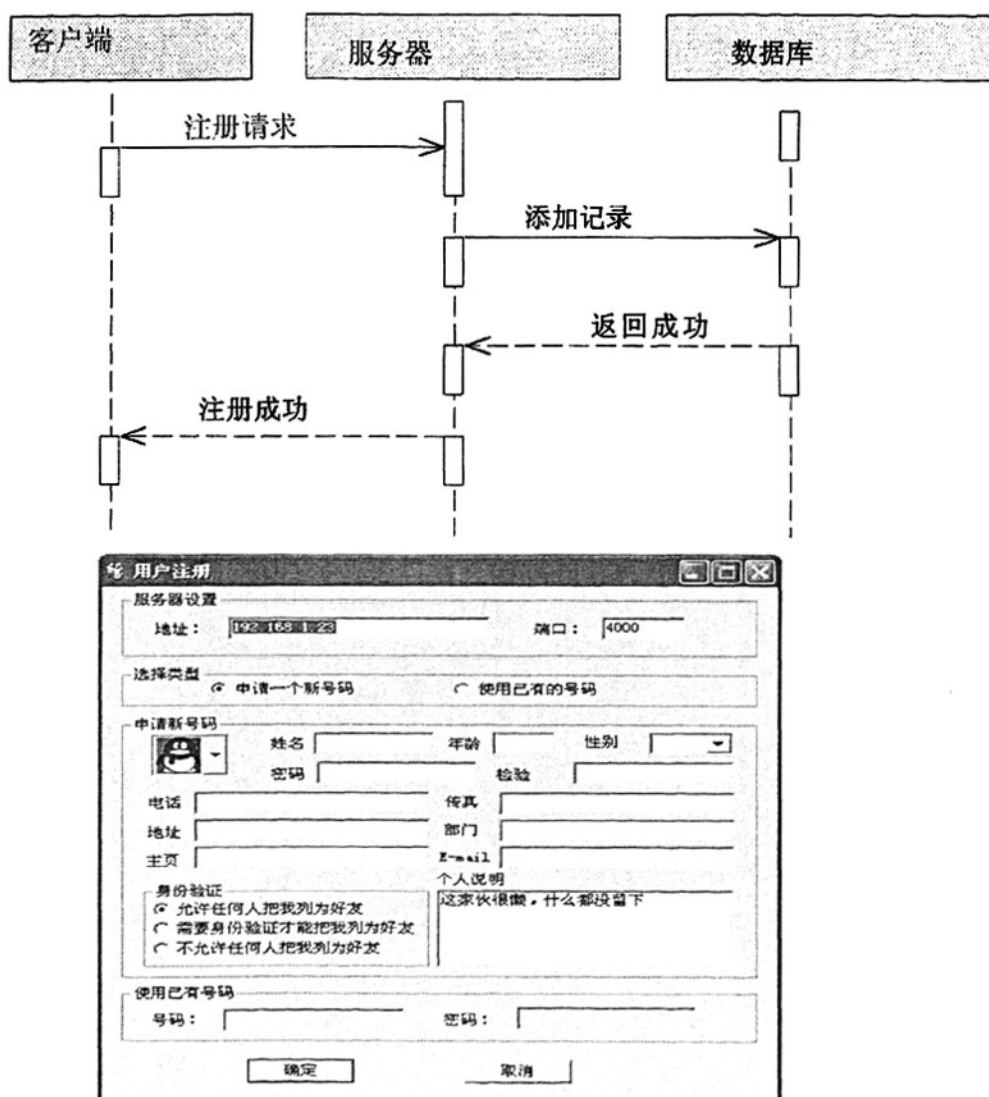
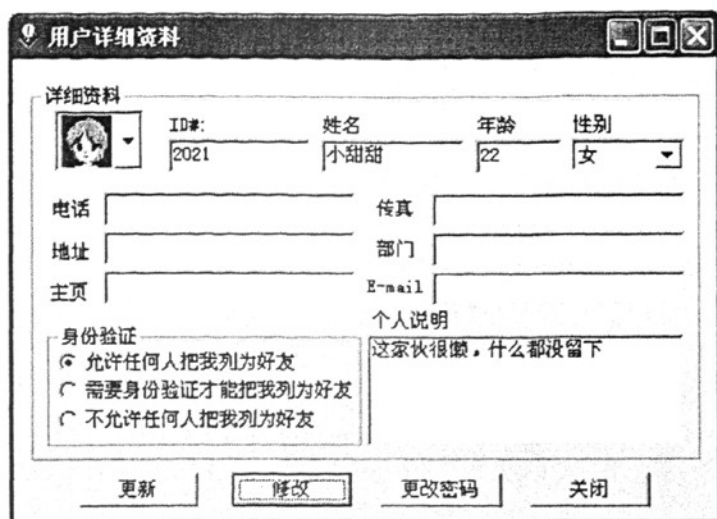


图 5.1 用户注册界面

及修改:



用户详细资料

详细资料

ID#
 姓名
 年龄
 性别

电话
 传真

地址
 部门

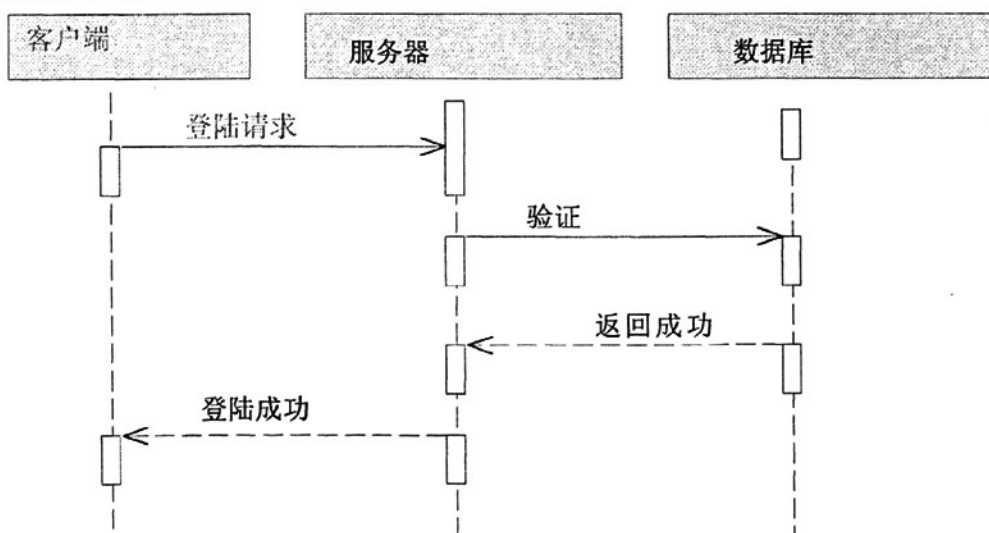
主页
 E-mail

身份验证
☒ 允许任何人把我列为好友
☐ 需要身份验证才能把我列为好友
☐ 不允许任何人把我列为好友

个人说明
 这家伙很懒，什么都没留下

图 5.2 用户详细资料

5.1.2 用户登录



a) 登录界面

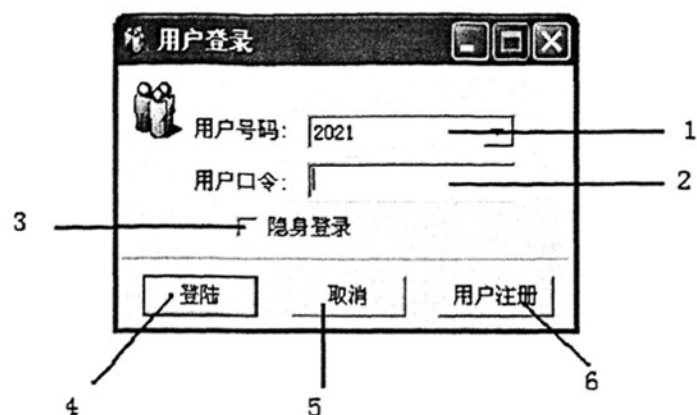
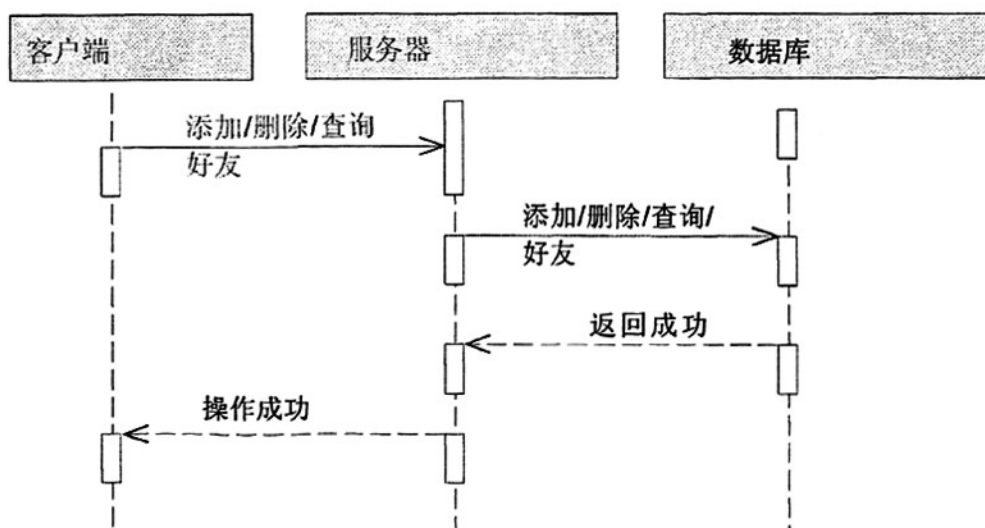
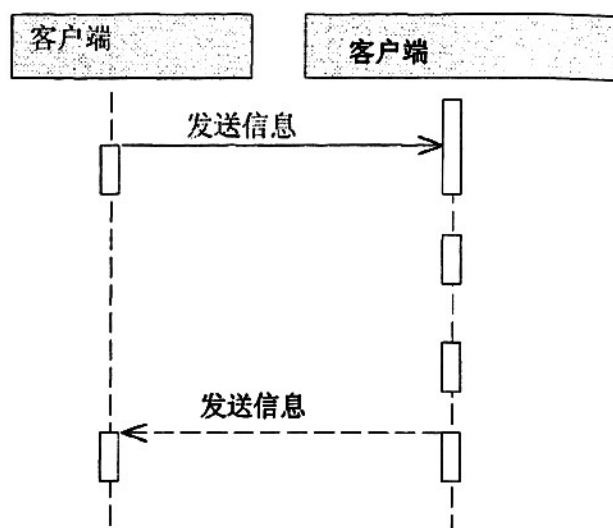


图 5.3 登录界面

1. 输入用户号码
2. 输入用户口令
3. 隐身登录选项
4. 登陆按键
5. 取消登陆
6. 用户注册选项

5.1.3 用户管理





5.1.4 在线管理

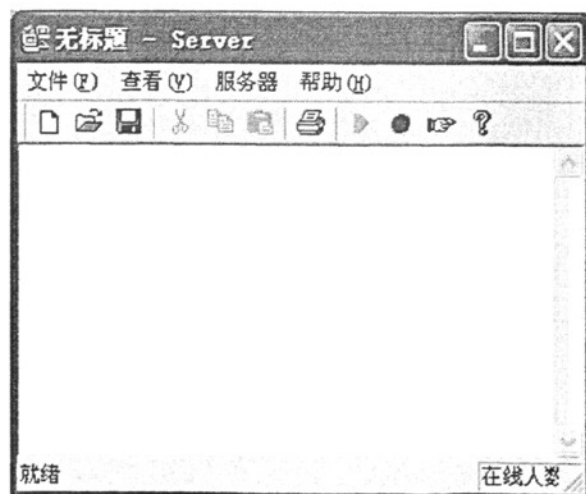


图 5.4 服务器主界面

由于开发占据了我大部分的精力和时间，平时所作的测试也就是一些手工的调试和边界的黑盒测试，在这里只是列举我测试的一些范围，不做详细描述。

1. 服务器最大登陆人数测试
2. 消息中转测试

3. 添加删除好友功能测试
4. 注册功能测试
5. 登陆或隐身或者下线后头像颜色测试等等

5.2 客户端功能模块实现

b) 主界面

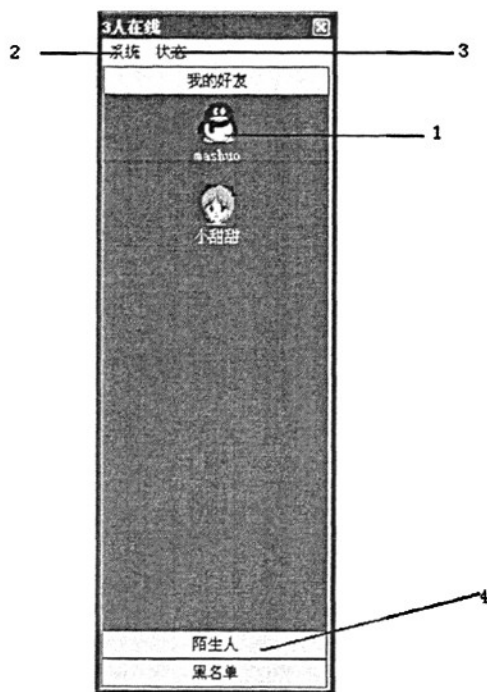


图 5.5 主界面

1. 用户头像及昵称
 2. 系统参数设定
 3. 登录状态选项
 4. 自定义分组
- c) 主界面参数选项

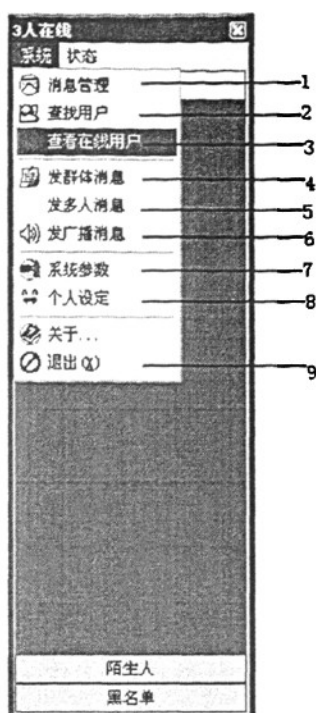


图 5.6 主界面参数选项

1. 对消息的发送进行基本的管理
 2. 可以按 ID 和用户名对用户进行查找
 3. 查看目前在线的用户
 4. 对所有人发送消息
 5. 按姓名列表和 ID 列表发送多人消息
 6. 向所有人发送广播消息
 7. 对服务器的 IP 地址和端口进行设置以正确登录
 8. 可以修改个人信息
 9. 推出系统
- d) 聊天模式对话框

5.2.1 发送消息

e)

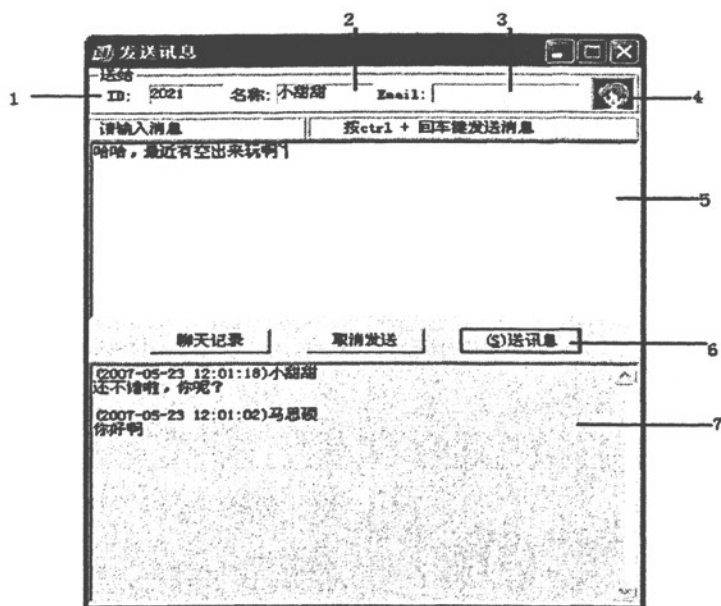


图 5.7 聊天窗口

1. 对方用户 ID
2. 对方用户名称
3. 对方用户 Email
4. 对方用户头像
5. 发送信息主窗口
6. 发讯息按钮 (可用 ctrl+回车键代替)
7. 查看聊天记录
- f) 其他界面

5.2.2 查看在线用户

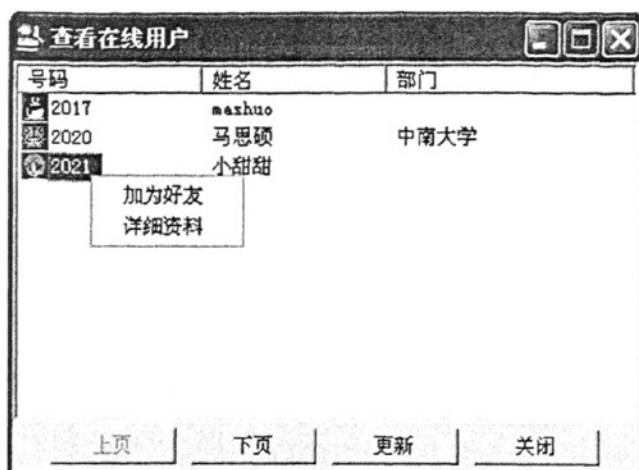


图 5.8 查看在线用户界面

5.2.3 发送广播信息

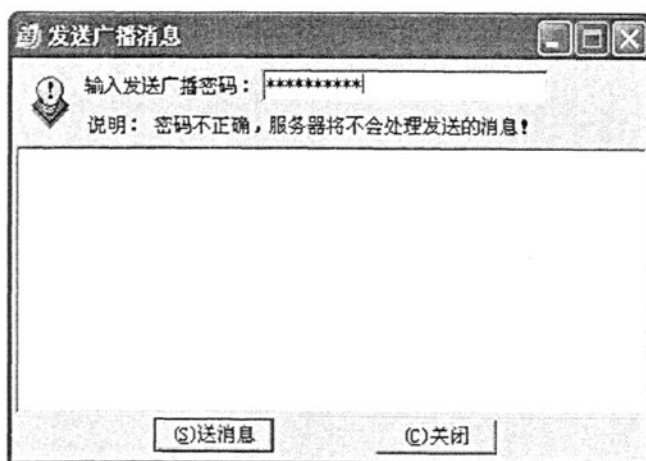


图 5.9 发送广播消息

5.2.4 查找用户

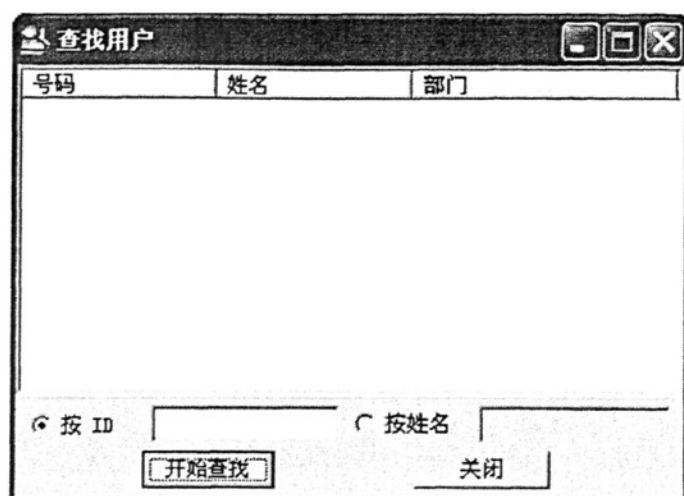


图 5.10 查找用户



图 5.11 卡通头像选择

第6章 核心算法

6.1 服务器端检查用户是否仍然在线的算法

6.1.1 设计思想

客户端的用户，可能因为某种异常情况，或死机，或执行非法操作，或忽然掉线，网络发生故障等，可能使客户端程序意外退出。没有或无法发送回下线消息，也有可能发送的下线消息丢失。所以，必须以一定的时间间隔去检查每一个在线的用户，看其是否仍然在线。因为我需要时常发送回当前在线的总人数，所有，可以把发送这条消息作为检查。若发送无响应，则表示用户已经离线了。

6.1.2 算法简述

以每 TimerSpanServer 的时间为间隔，向每个在线(或隐身)的用户发送当前有多少人在线的消息，成功就继续。若发送不成功（客户端无响应），就判断此用户不在线上，并从 Friends 表中查出所有把此人当做朋友的人，向他们发送此人下线的消息（不处理此时发送不成功的情况）。

6.2 服务器端处理用户请求的算法

6.2.1 算法及描述

侦听到用户的请求，就会新建一个线程进行处理。根据调用线程的函数传来的参数，根据接受到的数据，取出其类型，根据不同的类型的数据，根据规定的流程进行分类处理用户请求。

此工人线程函数的结构模型为：

```
UINT ProcessRecvData(LPVOID param)
```

```
{  
    CData* pData=(CData*)param;  
    UINT index=pData->index;  
    Switch(index)  
    {  
        case ONHIDE::
```

```

.....
break;
case ONLINE:
.....
break;
.....
}
}

```

6.2.2 分析每个处理流程

case ONLINE: 为用户请求上线的消息, 相应处理为:

- 1) 判断是否为合法用户
- 2) 若此时, 这个人已经在线, 就不处理
- 3) 发送上线成功消息给此用户
- 4) 如果此用户原状态为隐身, 则转到 (9)
- 5) 从数据库中查出此用户的所有好友帐号, 并发给此用户
- 6) 发送所有在线的好友的 ID,IP,Port,OnlineState,给此用户
- 7) 从 offbroadcast 表中查出所有离线广播, 依次发给此用户, 并删除在表上相应的数据
- 8) 从 offmsg 表中查出所有离线消息, 依次发给此用户, 并删除在表中相应的数据
- 9) 查找出把此用户当做朋友的所有用户 ID, 并向这些用户中在线的或隐身的发送此用户上线消息

case ONHIDE: 为用户请求隐身登陆的消息, 相应处理流程为:

- 1) 判断是否为合法用户, 否就不处理
- 2) 若此时, 这个人已经在隐身登陆, 就不处理
- 3) 发送隐身登陆成功消息给此用户
- 4) 如果此用户原状态为在线, 则转到 (9)
- 5) 从数据库中查出此用户的所有好友帐号, 并发给此用户
- 6) 发送所有在线的好友的 ID,IP,Port,OnlineState,给此用户
- 7) 从 offbroadcast 表中查出所有离线广播, 依次发给此用户, 并删除在表上相应的数据
- 8) 从 offmsg 表中查出所有离线消息, 依次发给此用户, 并删除在表中相应的数据
- 9) 查找出把此用户当做朋友的所有用户 ID, 并向这些用户中在线的或隐身

的发送此用户隐身登陆消息

Case OFFLINE 用户发来的离线消息

- 1) 判断此用户是否为合法用户，否就不处理
- 2) 如果此用户已经下线，就不处理
- 3) 找出把此用户当作好友的人,给其中在线或隐向的发此用户的 OFFLINE 消息

case SEND_MSG_TO_FRIEND: 发给好友的离线消息

case REFUSE_AS_FRIEND: 别人拒绝某人把它加为好友

case ACCEPT_AS_FRIEND: 同意某人加为好友

case FRIEND_IDENTITY_VALIDATE: 需要身份验证

这四个类型的信息，处理过程相同：

- 1) 判断此用户 ID 和 FriendId 是否为合法用户，否就不处理
- 2) 判断此接受方的用户是否在线（或隐身），是则发送此消息给此用户，否则存入 offmsg 表中

Case MULTI_SEND_MSG: 用户给多个好友发的离线消息

- 1) 对每个接受方用户 ID，执行以下两步操作
- 2) 判断此用户 ID 和 FriendId 是否为合法用户，否就不处理
- 3) 判断此接受方的用户是否在线（或隐身），是则发送此消息给此用户，否则存入 offmsg 表中

Case TEST_BROADCAST_PWD: 检查发广播的密码

- 1) 判断此用户是否为合法用户，否则就不处理
- 2) 判断是否是正确的密码，发回一个 CMsg3 类的一个反馈消息

Case SEND_BROADCAST: 发送的广播消息

- 1) 判断此用户是否为合法用户，否则就不处理
- 2) 判断密码是否正确，不正确则不进行处理
- 3) 把这条广播消息插入到 broadcast 表中
- 4) 对所有用户，执行以下操作：
- 5) 如果在线，则发广播消息给用户，否则就存在 offbroadcast 表中

Case FRIEND_DETAIL: 用户请求查看某人详细信息

- 1) 判断此用户 ID 和 FriendId 是否为合法用户，否就不处理
- 2) 从 Users 表中查出此 FriendId 的用户的详细信息
- 3) 把此详细信息发送给请求用户

case ADD_AS_FRIEND: 用户要求把某人加为好友的请求

- 1) 判断此用户 ID 和 FriendId 是否为合法用户，否就不处理

- 2) 如果 FriendId 拒绝任何人加它或需要身份验证, 则发回 RE_ADD_AS_FRIEND 响应, 结束。
- 3) 在 Friends 表中查, 看此 FriendId 用户是否已经是 ID 用户的好友, 是则返回相应错误信息, 结束。
- 4) 把这个好友信息插入 Friends 表中
- 5) 发送成功加入消息
- 6) 看被加入的人是否在线, 是, 则发送一个通知消息, 否则把这条消息插入 offmsg 表中。

case APPLY_SHOW_ONLINE: 查看在线的人的请求

- 1) 判断此用户是否为合法用户, 否则就不处理
- 2) 按信息中的 Value 值, 进行定位
- 3) 如果超出范围, 则不处理 (Value 错误)
- 4) 把在指定位置开始的, 在线的人不超过 PersonNumEveryTime 个人的信息, 发送给用户

case FIND_FRIEND_BY_ID: 用 ID 号进行查找用户

- 1) 判断此用户 ID 和 FriendId 是否为合法用户, 否就不处理
- 2) 在数据库表 Users 中查找此用户
- 3) 如果找不到此用户则返回 ID_NOT_FOUND_BY_ID 消息, 结束
- 4) 如果找到了, 则返回这个用户的详细信息 (FOUND_FRIEND_BY_ID)

case FIND_FRIEND_BY_NAME: 用姓名查找朋友

- 1) 判断此用户是否为合法用户, 否则就不处理
- 2) 在 m_pUsers 数组中查找与此姓名相同的用户
- 3) 如果没有找到, 则返回一个 NAME_NOT_FOUND_BY_NAME 消息, 结束
- 4) 如果找到了, 则把这此人的相关信息存入 CshowOnlinePeople 的一个实例中, 并发送给请求的用户。

case DELETE_A_FRIEND: 删除一个好友

- 1) 判断此用户 ID 和 FriendId 是否为合法用户, 否就不处理
- 2) 在 Friends 表中删除 Friends.myid=msg.myid and Friends.friendid=msg.friendid 的那项记录

case DELETE_SELF_IN_FRIEND: 选择在某人的好友中删除自己

- 1) 判断此用户 ID 和 FriendId 是否为合法用户, 否就不处理。在 Friends 表中删除 Friends.myid=msg.friendid and Friends.friendid=msg.myid 的那项记录

case CHANGE_PERSONAL_INFO: 修改个人信息

- 1) 判断此用户是否为合法用户，否则就不处理
- 2) 按 Mask 的值来构造修改的 SQL 语句
- 3) 执行修改操作

case CHANGE_PASSWORD: 修改个人的密码

- 1) 判断此用户是否为合法用户，否则就不处理
- 2) 从 Users 中查出此用户的 password
- 3) 检验旧密码是否正确，否，则不处理，结束
- 4) 正确则修改此用户的密码

Case APPLY_ID_LOGIN: 申请一个新帐号

- 1) 把当前用户的详细信息插入 Users 表中
- 2) 更新相应数据结构中的数据 (m_nMaxUserId, m_nNumberOnline, m_pUsers)
- 3) 得到用户的 ID 号，发送给用户 APPLY_ID_OK 消息

case HAVE_ID_LOGIN: 使用已有号码进行登陆

- 1) 判断此用户是否为合法用户，否则就发送帐号不存在的信息，结束
- 2) 判断用户发出的密码是否正确，否则就发送密码错误的信息。结束
- 3) 密码正确，发送登陆成功消息

6.3 客户端处理服务器发过来数据的算法

在客户端接受到数据后，若不是确认消息，则向父窗口发送一个 WM_RECIEVE_MSG，将把接受到的数据做为参数传递过来。

在消息处理函数 ProcRecv 进行处理接受到的数据。对接受到的不同数据类型，分别进行相应处理。

现在，对其中一部分数据进行分析：

case SEND_MSG_TO_FRIEND: 接受好友发来的信息

首先判断消息的发送者是不是在自己的好友列表中，若是，则保存下当前用户的信息指针(pInfo)和状态指针(pState)，若不是，则在自己的陌生人列表中查找是否有这个人，找到，则保存下当前用户的信息指针(pInfo)和状态指针(pState)，没有找到，则说明是一个新的陌生人，新建其数据结构，添加到陌生人列表中，并保存其用户的信息指针(pInfo)和状态指针(pState)。然后如果 pState->pRecv 指向的对话框类实例没有建立，则建立实例。最后，向 pState->pRecv 指向的对话框发送 WM_RECVMSG 消息，并把 pInfo 和 pState 作为参数传递过去，让其显

示接受到的信息。

1. case ONLINE: 某人上线的消息

寻找好友中是否有这个人，有则保存其在线状态、IP 和 Port，然后让其加亮显示。

2. Case ONHIDE 某人隐身

寻找好友中是否有这个人，有则保存其隐身状态、IP 和 Port，然后让其变灰显示。

3. case OFFLINE 某人下线

寻找好友中是否有这个人，有则保存其离线状态，然后让其变灰显示。

4. case ONLINE_OK 上线成功

设置任务栏图标为上线状态，启动请求每用户详细资料的线程

5. case ONHIDE_OK 隐身登陆成功

设置任务栏图标为隐身状态，启动请求每用户详细资料的线程。

第 7 章 结论

近年来 P2P 正成为应用和研究的热点,以 Napster 和 Gnutella 为代表的 P2P 软件受到用户的热烈欢迎,导致这些 P2P 系统的规模急剧增加。

本软件即是基于 P2P 的即时通信软件,其以 TCP 协议为文本信息传输的基本协议,基本达到了任务书对本次课题的要求。

本文主要对本软件服务器端的设计作了比较详细的介绍,并对服务器端进行了初步的开发。而客户端的设计有很多需要完善的地方。如果要使本系统成为一个实际应用系统并在局域网上实际应用,除了完成客户端的功能外,笔者认为还应做以下的完善和开发工作:

1. 进一步完善底层通讯协议,使能够更好的处理数据的发送和接受。
2. 多线程下的对临界数据访问的问题。
3. 对发送的数据,进行一定的加密措施,使之更加安全可靠。
4. 对本地用户的密码的加密问题,在本程序中,没有对本地密码进行加密。
5. 在客户端系统,功能的设置有待进一步丰富,功能的实现还有待完善和改进。
6. 在客户端处理中,响应从服务器发来的数据,是在主线程里执行的,所以,当接受数据忙时,就会出现没有响应的情况。进一步考虑,如何能改善这种情况。

致 谢

首先,最感激的是导师金伟祖教授,本论文是在金老师的悉心指导下完成的,没有金老师的指导与鼓励,论文是不可能得以顺利完成的。从课题的研究、论文选题到实证调研、论文撰写,都饱含了金老师的亲切关怀和悉心指导,在论文遇到瓶颈时金老师给予我无微不至的指导与帮助。导师严谨的治学态度、深厚的学术功底使我获益良多,也深深的影响和激励着我;金老师热情向上的生活态度、宽宏豁达的处事风格,更在人生观上给予我很多启发。师从金老师是我一生的幸运与自豪。在此,学生对恩师的谆谆教导、鼓励与关心致以最诚挚的敬意!

时间过的很快,转眼间就要毕业了,在研究生的这段时间里我努力的学习,同济大学也给了我们很多的支持,在此我感谢学校的领导和老师,对我的关心和照顾。

最后,感谢父母亲在我求学过程中所给予的支持与鼓励,他们的关爱和殷切期望是我学习进步的不竭动力,愿亲爱的爸爸妈妈永远健康快乐!

毕业是人身中一个阶段的完成,也是另一个阶段的开始,愿以本文与我最敬爱的以及所有关心、帮助过我的人们分享。

参考文献

- [1] Dana Moore, John Hebel.对等网[M].清华大学出版社.2003年2月.
- [2] 张泊平.P2P 网络的应用与发展[J].科学技术与工程.2005年9月.第18期.pp.1271-1275.
- [3] Dreamtech Software Team.Peer to Peer Application Development [M].Hungry Minds, November.2001.
- [4] Douglas E. Comer.David L. Stevens.TCP/IP 网络互连 (第3卷): 客户/服务器编程及应用[M].人民邮电出版社.2002年1月.
- [5] 马丁著 胡琛 沙东键译. 网络革命: P2P [M], 2000年.
- [6] 胡峪 刘静. VC++编程技巧与示例[M]. 西安电子科技大学出版社, 2000年
- [7] Dreamtech 软件研发组 吴文辉 陈建荣 肖月国尊 等译. 对等网络编程源代码解析. 电子工业出版社, 2002年8月.
- [8] 夏云庆 编著 Visual C++ 6.0 数据库高级编程 北京希望电子出版社
- [9] 冯登国. 计算机通信网络安全. 北京: 清华大学出版社, 2001
- [10] 单国栋, 戴英侠, 王航. 计算机漏洞分类研究. 计算机工程, 2002, 28(10):3-6
- [11] 段兴等, 《visual basic 数据库实用程序设计 100 例》人民邮电出版社
- [12] 方美琪, 《软件开发工具》, 经济科学出版社
- [13] 李建中, 王珊. 《数据库系统原理 (第2版)》电子工业出版社, 2004.9
- [14] 李昭原, 刘又诚《数据库系统原理与技术》北京航空航天大学出版社
- [15] Applied Microsoft.NET Framework Programming (美) Jeffrey Richter 著 清华大学出版社
- [16] Computer Emergency Response Team/Coordination Center. CERT/CC Vulnerability Disclosure Policy. 2000 October 09.
- [17] Duc. A. Tran, Kien, A. Hua, et al, A Peer-to-Peer Architecture for Media Streaming, IEEE

JSAC Special Issue on Advances in Overlay Networks[J], pp.171-172, 2003.

[18] Zhao,B.Y., Kubiawicz,J. and Joseph, A., Tapestry: An infrastructure for fault-tolerant wide-area location and routing[J], IEEE JSAC Special Issue on Advances in Overlay Networks, pp.1-28, 2001.

[19] Stephen E.D., Deborah E., Dino. Fetal, An Architecture for Wide-area Multicast Routing [J], IEEE/ACM Transaction on Networking, 4(2), April 1996, pp.126-135.

[20] Castro.M, Druschel.P, Kermarrec. A-M et al, A large-scale and decentralized application-level multicast infrastructure [J], IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications), pp.100-110, 2002.

[21] Kien. A. Hua, Mounir A. Tantaoui, Wallapak Tavanapong, Video Delivery Technologies for Large-Scale Deployment of Multimedia Applications [J], Proceedings of The IEEE, Vol. 92, No. 9,pp.1-13, September 2004.

[22] Duc. A. Tran, Kien, A. Hua, ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming [J], IEEE JSAC Special Issue on Advances in Overlay Networks, pp.162-173, 2003.

个人简历 在读期间发表的学术论文与研究成果

个人简历:

张鑫淼, 男, 1984 年 6 月生。

2006 年毕业于中国矿业大学 艺术与设计学院 艺术设计专业 获学士学位。

2006 年入同济大学 软件学院 软件工程专业 读硕士研究生。

已发表论文:

[1]张鑫淼, 企业应用集成, 《九州学林》2007.6