武汉理工大学

硕士学位论文

基于XMPP的企业级即时通信系统的研究与实现

姓名：陈武

申请学位级别：硕士

专业：计算机应用技术

指导教师：熊盛武;杨世达

20090401

中文摘要

随着信息技术和网络通信技术的发展，即时通信系统的应用也得到了飞速的发展，比较典型的有 QQ、MSN，它们的出现使得在以网络为载体的新世纪出现了一种全新的交流方式，它极大地提高了人们交流的效率同时降低了交流的成本，但是现在的各个即时通信系统采用了各自不同的协议，使得不同系统用户之间无法交流，XMPP(Extensible Messaging and Presence Protocol)协议的出现解决了这一难题。

Jabber/XMPP 是一个由开源组织制定的、基于 XML 语言的协议，其开放特性使开发者之间可以互相竞争，促进了即时通信领域技术的进步与发展，同时改变了以往的即时通信系统之间无法互联的状况。由于 Jabber/XMPP 系统采用了分布式的结构和模块化的系统架构，因此很容易对系统的功能进行扩展。正是基于以上优点，基于 XMPP 协议的即时通信系统成为企业信息化过程中必不可少的一部分。

论文包括以下主要内容：

1．介绍了 XMPP 协议的相关知识，并列出该协议的详细流程。

2．详细分析了该企业级即时通信系统系统需求，根据系统的设计原则和功能目标进行总体架构，给出了系统的总体设计和功能模块的划分。

3．根据系统的设计原则，结合开源的协议包 smack，实现了该系统需求中要求实现的功能。

论文的重点之处在于通过开源协议的实现，构建出适合中国国情的企业级即时通信系统架构，并将其运用于实践中。实践表明，该整合框架功能强大、耦合度低，增强了系统的可维护性和可扩展性，达到了即时通信系统的设计目标。

关键字：XMPP，smack，企业级，XML，即时通信系统

Abstract

With the development of information and network communication technology, instant messaging systems have also developed rapidly, such as QQ, MSN, the emergence of instant messaging systems created a new exchange method in the new century which was on the carrier of network. The instant messaging systems improved the efficiency and reduced the cost greatly when people communicate with each other, but the various instant messaging systems were based on different protocols, so that the users could not communicate with each other by instant messaging systems, this problem was solved by XMPP (Extensible Messaging and Presence Protocol).

Jabber/XMPP which was based on XML was developed by an open source organization. The emergence of Jabber/XMPP leaded to the public competition between developers and promoted the development in the field of instant messaging technology. It also changed the situation that instant messaging systems can't communicate with each other. It was easy to extend the functions of Jabber/XMPP systems, because Jabber/XMPP systems were based on a distributed architecture and modular system architecture. Because of these advantages, the instant messaging system based on XMPP for enterprise became an essential part of the enterprises.

The thesis includes the following content:

1. A summary about the relevant knowledge of XMPP and the detail process of XMPP.

2. The thesis analyzed the requirements of the enterprise-class instant messaging system, gave the overall design of the system and the delineation of functional modules in accordance with the principles of system and function design.

3. Realized the functions that were demanded by the systems through combining with open-source package of protocol in accordance with the principles of system design.

The thesis focused on building enterprise-class instant messaging systems architecture which was suitable for China's national condition by realizing open

source protocol. Practices proved that the framework of the integration was powerful and low coupling. The framework of the integration enhanced the system's maintainability and scalability, which achieved the goals of instant messaging system.

Keywords: XMPP, smack, enterprise-class, XML, instant messaging systems

独 创 性 声 明

本人声明，所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得武汉理工大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名：[signature] 日期 2009.5

关于论文使用授权的说明

本人完全了解武汉理工大学有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权武汉理工大学可以将本学位论文的全部内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存或汇编本学位论文。同时授权经武汉理工大学认可的国家有关机构或论文数据库使用或收录本学位论文，并向社会公众提供信息服务。

（保密的论文在解密后遵守此规定）

研究生签名：[signature] 导师签名：[signature] 日期 2009.5

第1章 绪 论

1.1 选题背景

随着信息技术以及网络技术的发展，人们相互之间交流的手段越来越多，而其中最重要的转变应该是即时通信系统的出现，短短的二十年间即时通信系统的功能越来越强大，用户越来越多，极大地方便了人与人之间的交流。

即时通信系统（IM，Instant Information System）实现了利用现代计算机网络通信技术进行信息交互，使人与人之间、企业与企业之间的交互变得方便而高效。由于即时通信系统的众多优势，现在市面上出现了许多种即时通信系统，在这些 IM 软件给我们的生活带来方便的同时，也出现了新的问题，也就是这些 IM 各自遵循不同的协议，从而导致这些系统之间不能互联互通，Jabber/XMPP 协议的出现解决了这一难题[1][2]。

Jabber/XMPP 是一个由开源组织制定的、基于 XML 语言的协议，其开放特性使开发者之间可以互相竞争，促进了即时通信领域技术的进步与发展，同时改变了以往的即时通信系统之间无法互联的状况。由于 Jabber/XMPP 采用了分布式的结构，模块化可扩展的系统架构使得扩展它的功能变得简单[3]。

随着企业信息化建设的发展，即时通信系统逐步融入企业之中，因为相对于传统的通信手段(电话、邮件)来讲，即时通信系统以网络为载体，具有成本低，效率高等优点，使人们摆脱了复杂费时的电话和枯燥无味的邮件，将即时消息、文件传输、语音通信和视频通信等功能成功的集成于即时通信系统中，从而满足了不同工种和不同任务的需要，为现代企业提供了一种全新的办公平台[4][5]。

1.2 国内外发展现状

即时通信是一种基于互联网应用的实时交互方式。网络上的用户可以利用 IM 软件实现文字、音频和视频等信息的即时传送，以及点对点的数据交换。在信息时代的背景下，IM 一经推出迅速风靡全球，它极大地方便了人与人之间的沟通。

上世纪九十年代中期，四位以色列人开发出世界上第一个即时通信软件ICQ，随后便出现各种各样的即时通信软件，如雅虎公司与微软公司分别推出Yahoo! Messenger 和 MSN Messenger，其中 1999 年 2 月腾讯公司推出的 OICQ 获得成功(OICQ 在 2000 年正式改为 QQ)，是目前国内拥有用户数量最多的即时通信系统。较为流行的 IM 软件还有阿里旺旺、网易泡泡、GTalk 等[6][7]。

即时通信系统是计算机技术发展的产物，系统的开发涉及到多项技术的融合，因此即时通信系统是一个复杂的系统工程，即时通信的出现是基于互联网通信方式的一次重大变革，对即时通信的研究有着重要的应用价值。目前即时通信系统软件的开发没有统一的协议标准，大部分公司都是自己开发通信协议或者在开源的协议基础上进行修改，这就造成了腾讯公司一家独大的局面，不利于即时通信系统的发展，同时也给广大的用户使用带来不便，而 XMPP 协议的出现解决了这一难题，使得大量的 IM 有了一个可靠的高效的开放协议，有利于实现各种 IM 的资源整合，因此致力于基于 XMPP 协议的即时通信系统研究，开发出适合中国国情的中小型企业即时通信系统具有重要的意义[8][9]。

近年来，随着 XML 技术的日趋完善及广泛应用，基于 XML 语言的通信协议已经成为主流，强烈的冲击着传统的通信协议。因为采用基于 XML 语言的通信协议，使系统的开发摆脱了语言的束缚，使协议和程序开发语言完全脱离，同时也使客户端和服务器脱离，人们可以根据自己的喜好选择开发所用的语言。简化了应用程序的开发，降低了各个模块的耦合度，这些技术为即时通信系统的开发提供完善的框架，成功地解决了即时通信系统开发中出现的许多问题。

1.3 课题研究意义

随着 internet 的普及和网络技术的发展，通过互联网进行实时信息的传输成为当今研究的技术热点，也是各大企业新时期的价值增长点，因为同传统的交流方式相比，即时通讯具有价格低廉，用户体验更加舒服的优势，这将改变人们的行为方式、提高工作效率，具有里程碑的意义，因而这就决定了实时通信系统将拥有大量的用户群，并将带来巨大的经济利益。于是各大运营商开发了各种各样的软件，各种软件所遵守的通信协议各不相同，这就导致不同的实时通信软件之间无法通信，这就限制了实时通信软件的普及也给人们的使用带来不便，XMPP 协议的出现使即时通信领域有了统一的标准，该协议的出现将打

破个别软件的垄断，实现不同软件间的互联互通，使人们的使用更加方便[10][11]。

本文研究的主要内容有：1.Jabber/XMPP 技术是开源的和可扩展的并能使现有的即时通信应用系统之间实现互连互通。2.XMPP 提供了不同系统间通信 XML 数据流处理技术，使系统的开发摆脱了程序语言的束缚，能为多网络间连接提供安全和易于实现的编程语言环境。3.XMPP 提高了一套高效的数据路由机制。4.XMPP 提高实时信息处理的效率和文件传输的效率，同时实现了同 SOCKS5 协议的集成。

1.4 课题研究内容及论文组织结构

课题主要研究内容包括以下几个方面：

1. 对现有即时通信系统开发技术 JAVA EE、XMPP 协议、SASL 协议、XML 语言等进行分析和研究。

2. 详细分析即时通信系统的需求并设计该系统。

3. 提出基于 XMPP 协议的实现方案，通过 java 语言实现协议中规范的相关功能。

论文的组织结构如下：

第 1 章 绪论。绪论部分简要介绍了即时通信系统的发展现状，以及该系统的实现对企业信息化建设的重大实践意义。

第 2 章 基于 XMPP 协议的即时通信系统相关技术。本章从 C/S 模式的分布式体系结构分析开始，介绍了即时通信系统实现的各种协议，以及 XMPP 协议的实现的载体 XML 语言在这一协议中的应用。

第 3 章 系统的总体设计和服务器端的实现。本章具体介绍了 XMPP 协议在即时通信系统服务器端的实现，介绍了服务器端的工作原理，并对数据库做了详细设计，并对支持文件传输的 SOCKS5 协议进行了集成。

第 4 章 即时通信系统客户端的实现。本章具体介绍了 XMPP 协议在即时通信系统客户端的实现，介绍了客户端的工作原理。

第 5 章 系统的测试。从分层的角度对系统进行测试，以确保系统的可靠性和正确性。

第 6 章 总结与展望。本章对论文的主要工作进行总结，并指出了下一步研究工作的重点，展望了未来即时通信系统发展方向。

第2章 基于 XMPP 协议的即时通信系统相关技术

2.1 C/S 模式的体系结构

传统的应用系统开发普遍采用 C/S（client/server，客户端/服务器端）两层体系结构，其中客户端负责向用户提供操作界面，接受用户输入数据信息，通过请求（request）向服务器端发送数据，接受服务器端的响应（response），并将响应结果向用户显示。服务器端主要负责被动接受客户端的请求，接受数据并进行响应的逻辑处理，以及数据持久化工作，并发送响应，将处理结果返回给客户端。这种 C/S 两层体系结构是一种通用的，基于消息的模块化结构，其结构比较简单。

2.1.1 XML 的特点和优势

可扩展标记语言 (XML) 是 Web 上的数据通用语言，它使开发人员能够将结构化的数据从不同的应用程序传递到桌面，进行本地解析和封装，XML 允许为特定应用程序创建特定的数据格式，它是在服务器之间传输数据的理想格式[12]。XMPP 协议采用 XML 作为传输的数据格式，使得 XMPP 协议实现了跨平台，XML 主要具有以下几个特点：1、XML 是一个精简的 SGML，它将 SGML 的丰富功能与 HTML 的易用性结合到 Web 应用中，它保留了 SGML 的可扩展功能，这使得 XML 从根本上有区别于 HTML，并且 XML 种还包括可扩展格式语言 XSL(Extensible Style Language)和可扩展链接语言 XLL(Extensible Linking Language)使得 XML 的显示和解析更加方便快捷；2、XML 是 W3C 正式批准的，它完全可用于 Web 和工具的开发，XML 具有标准的域名说明方法，支持文档对象模型标准、可扩展类型语言标准、可扩展链接语言标准和 XML 指针语言标准。使用 XML 可以在不同的计算机系统间交换信息，而且还可以跨越国界和超越不同文化疆界交换信息；3、XML 支持复用文档片断，使用者可以发明和使用自己的标签，也可以与他人共享，可延伸性大，在 XML 中，可定义一组无限量的标准，可以有效地进行 XML 文件的扩充[13][14]。

2.2 XMPP 协议基础

2.2.1 XMPP 协议简介

XMPP 协议采用的是客户端-服务器架构，所有从一个客户端发到另一个客户端的消息和数据都必须经过 XMPP 服务器转发，而且还支持服务器间 DNS 的路由，这也就使构建服务器集群成为可能，使不同服务器下的客户端也可以通信，XMPP 的前身是 Jabber，一个开源组织制定的网络即时通信协议[14]。

对于 XMPP 协议来说，由于其核心是采用 XML 流传输协议定义的，从而使得 XMPP 协议能够应用在一个比其它网络通信协议更高效更规范的基础上。XMPP 协议的核心就是在网络上分片断发送 XML 流的协议。这个流协议是 XMPP 的即时通讯指令的传递手段，也是一个非常重要的可以被进一步利用的网络基础协议。所以可以说，XMPP 就是在 TCP 连接上传输 XML 流[15]。借助于 XML 易于解析和阅读的特性，使得 XMPP 的协议的实用性更强，适用面更广[16]，XML 流传输的是与即时通讯相关的信息。在此之前的各种通信协议中，这些信息要么用二进制的形式发送，要么用纯文本指令加空格加参数加换行符的方式发送。而 XMPP 传输的即时通讯信息和过去的类似，只是协议的形式变成了 XML 格式的形式[17]。这不但使得解析变得容易了，人们阅读起来也变得容易了，同时方便了开发和查错，但是同时由于大量的信息重复，虽然确保了信息的完整，但网络传输的效率变低，这是 XMPP 的不足之处。

2.2.2 XMPP 协议簇

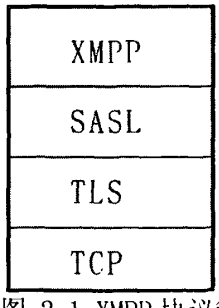

图 2-1 XMPP 协议簇

为了防止服务器间发送的数据被篡改或偷听，服务器通信引入了 TLS 机制，使用 TLS 能实现数据加密，从而保证了在传输过程中数据的安全，该协议由两

层组成：TLS 记录协议（TLS Record）和 TLS 握手协议（TLS Handshake）。较低的层为 TLS 记录协议，位于某个可靠的传输协议（例如 TCP）上面。其协议在应用中的顺序为：TCP--TLS--SASL—XMPP，TCP 是传输控制协议，TLS 是传输层安全协议，SASL 是简单认证和安全层协议，以上所列协议根据各自不同的功能，在系统实现中处于不同的层级[18][19]。

基本 IM 协议组：定义了基本的即时通信系统所应该支持的协议组，不同的协议代表了不同的功能，这些功能的扩展是基于 XMPP 协议良好的扩展性能的，任何一项功能的扩展都有其固定的规范，这些是在协议中规定好的，如表 2-1 所示[20][21]：

表 2-1 基本 IM 协议组

协议规范	需求级别
XMPP 核心协议	必需实现的
XMPP IM 协议	必需实现的
服务发现协议	必需实现的
带内注册协议	推荐实现的
非 SASL 认证协议	对服务器端推荐实现的 ； 对客户端不推荐实现
出错条件映射	推荐实现的
实体能力协议	对客户端必需实现的

中级 IM 协议组：定义了系统中级应用所应该支持的协议组，就是在基本的通信系统中添加了多用户聊天，文件传输等功能，如表 2-2 所示：

表 2-1 中级 IM 协议组

协议规范	需求级别
基本 IM 协议组协议	必需实现的
多用户聊天协议	必需实现的
XHTML-IM 协议	必需实现的
文件传输协议	必需实现的
实体能力协议	必需实现的

须注意的是，表 2-2 中的协议仅应用于客户端（也即，它们并对服务器引入新的要求）。另外，这些协议有各自依赖的协议[23][24]

2.2.3 地址空间

一个实体可以是任何一个被认为是一个网络端点，而且它是通过 XMPP 协议进行相互间通信的。这些实体都有一个具有唯一性的地址，并符合规范要求

的格式，一个 XMPP 实体的地址被称为 Jabber Identifier 或 JID ，其作用类似于 IP 地址。一个合法的 JID 包括一组排列好的元素，包括域名，节点名，和资源名，其格式为：jid = [node"@"]domain["/"resource][25][26]

域名（domain identifier）是一个主要的元素并且是 JID 中唯一必需的元素，一个纯粹的域名或 IP 地址也是一个合法的 JID，它通常代表网络的网关或者主服务器，其它实体通过连接它来实现 XML 转发和数据管理功能[26]。

节点名（node identifier）是一个可选的第二元素，放在域名之前并用符号"@"分开。它通常表示一个向服务器或网关请求和使用网络服务的实体(比如一个客户端)，当然它也能够表示其他的实体(比如在多用户聊天系统中的一个房间)。节点名所代表的实体，依赖于一个特定的域名[26]。

资源名（resource identifier）是一个可选的第三元素，它放在域名的后面并由符号"/"分开。资源名可以跟在 <node@domain>后面也可以跟在 <domain> 后面。它通常表示一个特定的会话，连接或者一个附属于某个实体的对象，比如多用户聊天室中的一个参加者。对于服务器和和其他客户端来说，资源名是不透明的[26]。

所有 JID 都是基于上述的结构。类似 <user@host/resource> 这种结构，最常用来标识一个即时消息用户，这个用户所连接的服务器，以及这个用户用于连接的资源（比如某一类型的客户端软件，QQ、MSN 等等）。

2.2.4 XMPP 协议的命名空间

jabber:iq:private—私有数据存储，用于本地用户私人设置信息，比如用户备注等
jabber:iq:conferenee——一般会议，用于多个用户之间的信息共享
jabber:x:encrypted—加密的消息，用于发送加密信息
jabber:x:expire—消息终止
jabber:iq:time—客户端时间
jabber:iq:auth—简单用户认证，一般用于服务器之间或者服务器和客户端的认证
jabber:x:roster—内部的花名册条目
jabber:x:signed—标记的在线状态
jabber:iq:search—用户数据库查询，用于向服务器端发送查询请求
jabber:iq:register—注册请求，用于用户注册相关信息。
jabber:iq:roster—花名册(好友名单)管理，用于企业的部门设置或者是用户组设置
jabber:x:conference—会议邀请，用于向参加会议用户发送开会通知
jabber:x:event—消息事件
vcard-temp—临时的 vCard，用于设置用户的头像以及昵称等信息[26]

2.2.5 XMPP 协议消息格式定义

即时通信系统的各个实体之间的通信是以XML节和XML流的形式出现的，不仅可以进行简单的文本交流，也可以携带各种复杂的数据和文件，这一特性决定了基于XMPP协议的即时通信系统的客户端和服务器端摆脱了语言的限制，只需要能对XML语言进行解析就可以了。

XML流的定义：一个XML流是一个类似于容器的流，其中包含了两个实体之间通过网络发送的XML元素。一个XML流是由一个XML打开标签<stream>开始，流的结尾则是以一个XML关闭标签</stream>结束 。在流的开始到结束之间，初始化它的实体可以通过流发送大量的XML元素，用于流的交互和通信，最终整个XML流是一个符合XML规范的XML文本[25]。

XML节的定义：一个XML节是一个XMPP实体通过XML流向另一个XMPP实体发送的XML报文中的一个节点。一个XML节存在于根元素<stream>的下层，任何XML节都是从一个XML流的下一级的一个打开标签开始，到对应的关闭标签，每个XML节代表了不同的信息实现了不同的功能。在这里定义的XML节仅限于<message/>，<presence/>和<iq/>元素。

XMPP协议包括3个顶层XML元素[26]

<presence>此元素用于确定用户的订阅状态，可以查询、询问好友的在线状态，同时也可以发布自己的在线状态

如下所示：

```
<presence>
from = 'chenwu@192.168.0.5/msn'
to = 'wangjun@192.168.0.5/ msn'
type = 'probe'>
</presence>
```

<presence>元素可以取下面几种值[27][28]：

probe：用于服务器向客户端查询订阅状态，相当于一个“探针”

subscribe：请求订阅别人，即请求加对方为好友

subscribed：同意被别人订阅，也就是确认被对方加为好友

unsubscribe：取消订阅别人，请求删除某好友

unsubscribed：拒绝被别人订阅，即拒绝对方的添加请求

<message>：用于在两个XMPP用户之间发送信息，其中包含的元素规定了消息的源节点、目的节点、发送的形式和信息的内容，如果用户在线服务器立即转

发；否则服务器就存储。

to：标识消息的接收方，一般用 JID 标示

from：指发送方的名字或标示(id) ，一般用 JID 标示。

text：此元素包含了要提交给目标用户的信息，是纯文本信息。

type：是发送的消息的类型(群发或单发)，如果是“groupchat”则“to”属性中只能添服务器的 JID(多用户聊天中用到这种类型)，意味着是群发给目的服务器下的所有用户。如果是“chat”则“to”属性可以填服务器也可以填客户端的 JID，表示只同某一个用户聊天。

结构如下所示：

```
<message to = 'wangjun@192.168.0.5/ msn'
type = 'chat'>
<body>你好，在忙吗</body>
</message>
```

<iq>此元素管理 XMPP 服务器上任何两个用户间的转换，允许他们通过相应的 XML 格式的查询和响应，比如：查询用户的用户名、密码等信息，设置自己的用户名和密码等

<iq>主要的属性是 type。包括：

get：获取当前域值。

set：设置或替换 get 查询的值。

result：说明成功的响应了先前的查询。

error：查询和响应中出现的错误。

结构如下所示：

```
<iq from = 'wangjun@192.168.0.5/ msn'
    id= '1364564666'type = 'result'>
```

2.3 XMPP 核心协议

2.3.1 用户注册流程

步骤 1：客户端向服务器发送注册请求，jabber:iq:register 命名空间标示了这一 XML 节的作用，是服务器端确定其功能模块的依据。

```
<iq type='get' id='register1'>
<query xmlns='jabber:iq:register'/>
</iq>
```

步骤 2：服务器向客户端返回需注册的字段：用户名、密码和邮箱地址，表

示服务器端需要了解用户的上述信息。

```
<iq type='result' id=' register1 '>
<query xmlns='jabber:iq:register'>
<instructions>
  Choose a username and password for use with this service.
   Please also provide your email address.
</instructions>
<username/>
<password/>
<email/>
</query>
</iq>
```

步骤 3：客户端向服务器发送注册字段对应的值，客户端输入相关信息后，确认发送往服务器端，服务器端接收后会做相应的处理。

```
<iq type='set' id=' register 2'>
<query xmlns='jabber:iq:register'>
<username>wangjun</username>
<password>123456</password>
<email>wangjun@163.com</email>
</query>
</iq>
```

步骤 4：服务器返回注册成功信息，注册成功后，服务器返回成功信息，以便用户知道注册成功[30][31]。

```
<iq type='result' id=' register 2'/>
```

2.3.2 用户登录流程

步骤 1：客户端初始化流给服务器，以便能得到服务器的应答，从而确认该服务器是否可用。

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        to='XMPPExample.com'
        version='1.0'>
```

步骤 2：服务器返回一个流标签给客户端作为应答，以表示目的服务器可用。

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        id='c2s_123'
        from=' XMPPExample.com'
        version='1.0'>
```

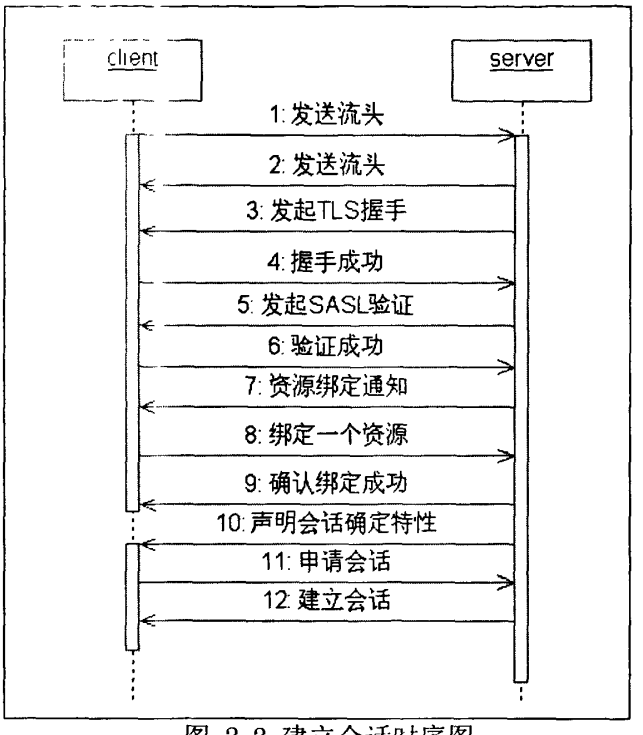

图 2-2 建立会话时序图

步骤 3：目的服务器发送 STARTTLS 节给客户端，该节中包括验证机制、加密机制和其他流特性。

```
<stream:features>
    <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
      <required/>
    </starttls>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>DIGEST-MD5</mechanism>
      <mechanism>PLAIN</mechanism>
    </mechanisms>
  </stream:features>
```

步骤 4：客户端发送 STARTTLS 节给目的服务器，表示客户端支持 TLS 协议。

```
<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

步骤 5：目的服务器通知客户端可以继续进行三次握手协议。

```
<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

步骤 5 (或者)：服务器通知客户端 TLS 握手失败并关闭流和 TCP 连接。

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

```
</stream:stream>
```

步骤 6：客户端和服务器尝试通过已有的 TCP 连接完成 TLS 握手。

步骤 7：如果 TLS 握手成功，客户端发送一个新的流给服务器。

```
<stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    to=' XMPPExample.com'
    version='1.0'>
```

步骤 7 (或者)：如果 TLS 握手不成功，服务器关闭 TCP 连接。

步骤 8：服务器发送一个流头信息应答客户端，表示开始另一协议，其中包括任何可用的流特性。

```
<stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    from=' XMPPExample.com'
    id='session_c2s'
    version='1.0'>
  <stream:features>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>DIGEST-MD5</mechanism>
      <mechanism>PLAIN</mechanism>
      <mechanism>EXTERNAL</mechanism>
    </mechanisms>
  </stream:features>
```

步骤 9：客户必须选择一种认证机制。如果服务器不支持该机制，将响应一个失败通知。

步骤 10：客户初始化一个新的流。

步骤 11：服务器回复一个被认证的流。

步骤 12：一旦客户认证成功，客户请求打开一个即时通信的会话。

```
<iq type='set' id='session1'>
<session xmlns='urn:ietf:parars:xml:ns:xmpp-session' >
</iq>
```

步骤 13：服务器通知客户创建会话成功或失败[33][34]。

```
<iq type='result' id='session1'>
<session xmlns='urn:ietf:parars:xml:ns:xmpp-session' >
<error type='wait'>
<internal-server-error xmlns='urn:ietf:parars:xml:ns:xmpp-stanzas' >
</error>
</iq>
```

2.3.3 获取好友在线信息流程

步骤 1、用户的本地服务器代替用户发送在线信息调查“探针”给，本地花名册中 subscription="to"和 subscription="both" 的联系人的可用资源。

```
<presence
        type='probe'
        from='server@192.168.0.5 '
        to='client@192.168.0.5'/ msn >
    <presence
        type='probe'
        from='server@192.168.0.5 '
        to='client@192.168.0.5/ msn '>
```

步骤 2、联系人的服务器在接收到调查“探针”后，代替所有可用的资源应答在线信息调查。

```
<presence
        from='juliet@192.168.0.15 '
        to=' server@192.168.0.5'
        xml:lang='en'>
      <show>away</show>
      <status>be right back</status>
      <priority>0</priority>
    </presence>
    <presence
        from='juliet@example.com/chamber'
        to= server @192.168.0.5/orchard'>
      <priority>1</priority>
    </presence>
    <presence
        from='benvolio@example.org/pda'
        to= server @192.168.0.5/orchard'
        xml:lang='en'>
      <show>dnd</show>
      <status>gallivanting</status>
    </presence>
```

2.3.4 添加好友的流程

步骤 1：客户端向本地服务器名册中添加新的条目，其中 item 中包含了好友的 JID，用户名和将其加入的群组。

```
<iq type='set' id='set1'from='user@192.168.0.5/balcony'>
      <query xmlns='jabber:iq:roster'>
        <item
```

```
            jid='contact@192.168.0.5'
            name='MyContact'>
          <group>MyBuddies</group>
        </item>
      </query>
    </iq>
```

步骤 2：服务器为这个新的名册条目初始化一个新的名册发送给这个用户的所有已经请求名册的可用资源，其'subscription'属性的值为 "none"； 并且以一个 IQ result 应答发送的资源表明名册设置成功了。

```
<iq type='set'to='user@192.168.0.5/balcony'>
    <query xmlns='jabber:iq:roster'>
      <item
          jid='contact@192.168.0.5'
          subscription='none'
          name='MyContact'>
        <group>MyBuddies</group>
      </item>
    </query>
  </iq>
<iq type='set'to='user@192.168.0.5/msn'>
    <query xmlns='jabber:iq:roster'>
      <item
          jid='contact@192.168.0.5'
          subscription='none'
          name='MyContact'>
        <group>MyBuddies</group>
      </item>
    </query>
  </iq>

  <iq type='result' id='set1' to='user@192.168.0.5/balcony'/>
```

步骤 3：用户向这个联系人请求在线信息的订阅，用户的客户端必须发送一个类型为'subscribe'的在线信息节给联系人。

```
<presence to='contact@192.168.0.5' type='subscribe'/>
```

步骤 4：用户的服务器初始化第二个新的名册发送给这个用户的所有已经请求名册的可用资源，把这个联系人设置成'none'订阅状态的未决的状态；这个未决的状态是由名册条目中包含的 ask='subscribe'属性所指示的。

```
<iq type='set' to='user@192.168.0.5/balcony'>
    <query xmlns='jabber:iq:roster'>
      <item
```

```
            jid='contact@192.168.0.5'
            subscription='none'
            ask='subscribe'
            name='MyContact'>
          <group>MyBuddies</group>
        </item>
      </query>
    </iq>
<iq type='set' to='user@192.168.0.5/msn'>
      <query xmlns='jabber:iq:roster'>
        <item
            jid='contact@192.168.0.5'
            subscription='none'
            ask='subscribe'
            name='MyContact'>
          <group>MyBuddies</group>
        </item>
      </query>
    </iq>
```

步骤 5：用户的服务器向联系人服务器转发订阅在线状态的请求。

```
<presence
        from='user@XMPPExample.com'
        to='contact@192.168.0.5'
        type='subscribe'/>
```

步骤 6：联系人的服务器向联系人的客户端转发订阅在线状态的请求。

```
<presence
        from='user@XMPPExample.com'
        to='contact@192.168.0.5'
        type='subscribe'/>
```

步骤 7：联系人的客户端添加一个条目并返回同意订阅在线状态的消息。

```
<iq type='set' id='set2'from='contact@192.168.0.5/msn'>
      <query xmlns='jabber:iq:roster'>
        <item
            jid='user@XMPPExample.com'
            name='SomeUser'>
          <group>SomeGroup</group>
        </item>
      </query>
    </iq>
    <presence to='user@XMPPExample.com' type='subscribed'/>
```

步骤 8：联系人的服务器初始化一个名册发送给所有联系人已请求名册的可用资源，包含一个关于那个用户的名册条目，并且其订阅状态为'from'(甚至联系

人不执行名册设置，服务器也必须发送它)； (2) 返回一个 IQ result 给发送的资源表示名册设置成功了； (3) 路由这个"subscribed"类型的在线信息节给用户，首先把'from'地址设为联系人的纯 JID；然后从所有联系人的可用资源向用户发送可用的在线信息。

```
<iq type='set' to='contact@192.168.0.5/resource'>
      <query xmlns='jabber:iq:roster'>
        <item
            jid='user@XMPPExample.com'
            subscription='from'
            name='SomeUser'>
          <group>SomeGroup</group>
        </item>
      </query>
    </iq>
    <iq type='result' to='contact@192.168.0.5/resource' id='set2'/>
    <presence
        from='contact@192.168.0.5'
        to='user@XMPPExample.com'
        type='subscribed'/>
    <presence
        from='contact@192.168.0.5/resource'
        to='user@XMPPExample.com'/>
```

步骤 9：向用户递送订阅成功的确认信息，向用户可用的资源广播新的名册，并递送可用的信息节。

```
<presence
        to='user@XMPPExample.com'
        from='contact@192.168.0.5'
        type='subscribed'/>
    <iq type='set' to='user@192.168.0.5/resource'>
      <query xmlns='jabber:iq:register'>
        <item
            jid='contact@192.168.0.5'
            subscription='to'
            name='MyContact'>
          <group>MyBuddies</group>
        </item>
      </query>
    </iq>
<iq type='set' to='user@192.168.0.5/msn'>
      <query xmlns='jabber:iq:roster'>
        <item
```

```
            jid='contact@192.168.0.5'
            subscription='to'
            name='MyContact'>
          <group>MyBuddies</group>
        </item>
      </query>
    </iq>
    <presence
        from='contact@192.168.0.5/resource'
        to='user@XMPPExample.com/resource'/>
```

步骤 10：用户客户端确认订阅在线状态成功

```
<presence
        from='user@XMPPExample.com'
        to='contact@192.168.0.5'
        type='subscribe'/>
```

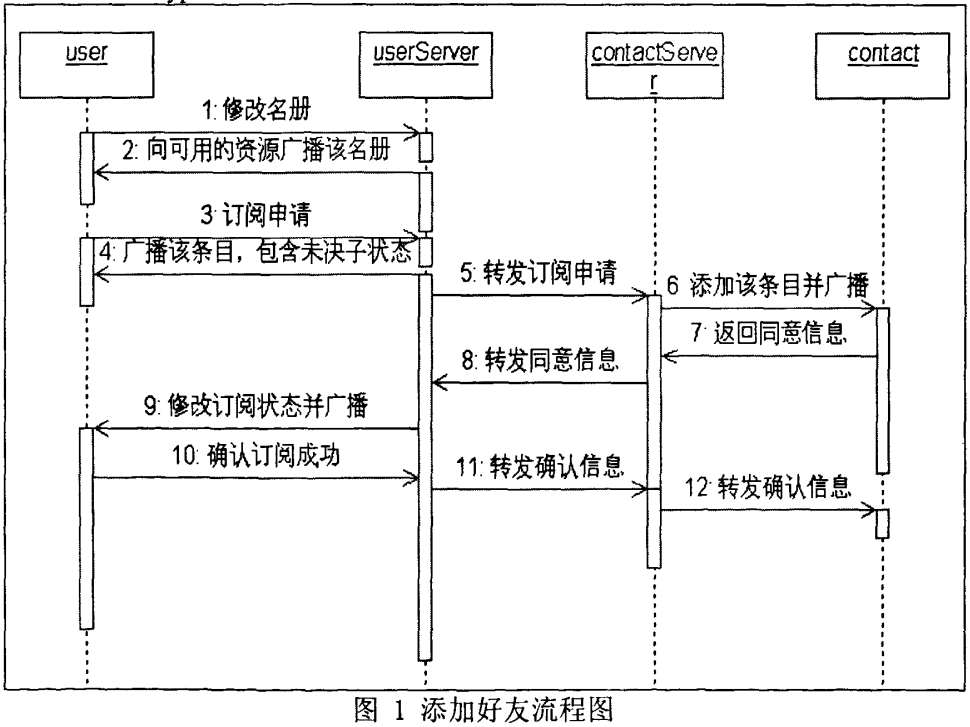

图 1 添加好友流程图

2.3.5 删除好友流程

该流程类似于删除好友的过程,如图 4 删除好友流程图所示。

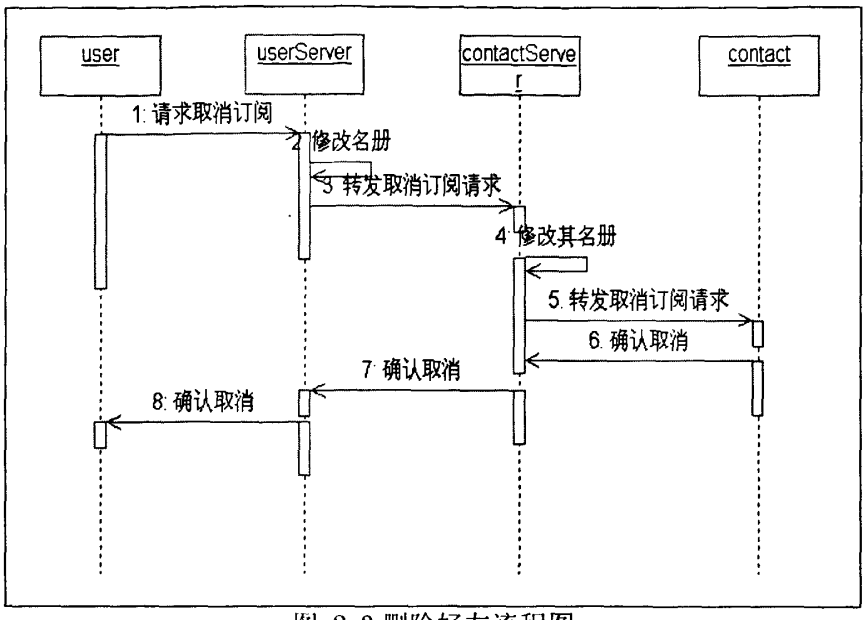

图 2-3 删除好友流程图

步骤 1：如果用户想取消对联系人的在线信息的订阅，用户必须发送一个"unsubscribe"类型的在线信息节给联系人。

```
<presence to='contact@192.168.0.5' type='unsubscribe'/>
```

步骤 2：作为一个结果，用户的服务器必须发送一个名册推送给这个用户的所有已请求名册的可用资源，包含一个关于这个联系人的更新名册条目，其'subscription'属性设为"none"； 并且必须路由这个"unsubscribe"类型的在线信息节给联系人(首先把'from'地址设为用户的纯 JID(<user@XMPPExample.com>))。

```
<iq type='set'>
    <query xmlns='jabber:iq:roster'>
      <item jid='contact@192.168.0.5'
            subscription='none'
            name='MyContact'>
        <group>MyBuddies</group>
      </item>
    </query>
  </iq>
  <presence
      from='user@XMPPExample.com'
      to='contact@192.168.0.5'
      type='unsubscribe'/>
```

步骤 3：接收到指向联系人的"unsubscribe"类型在线信息节之后，联系人的服务器必须初始化一个名册发送给这个联系人的所有已请求名册的可用资源，

包含一个关于这个用户的名册条目，其'subscription'属性值设为"none" (如果联系人不可用或未曾请求名册，联系人的服务器必须修改名册条目并在下次联系人请求名册时发送那个已修改的条目)；并且必须发送这个"unsubscribe"状态改变通知给联系人。

```
<iq type='set'>
    <query xmlns='jabber:iq:roster'>
      <item jid='user@ XMPPExample.com'
          subscription='none'
          name='SomeUser'>
        <group>SomeGroup</group>
      </item>
    </query>
  </iq>
  <presence
      from='user@ XMPPExample.com'
      to='contact@192.168.0.5'
      type='unsubscribe'/>
```

步骤 4：接收到"unsubscribe"类型的在线信息节之后，联系人应该承认收到那个订阅状态通知，要么发送一个"unsubscribed"类型的在线信息节给用户以证实它，要么发送一个"subscribed"类型的在线信息节给用户否认它； 这个步骤不影响订阅状态，但是让联系人的服务器知道它必须不再发送订阅状态变更通知给联系人。

步骤 5：联系人的服务器接着必须发送一个"unsubscribed"类型的在线信息节给用户；并且应该向用户发送从这个联系人的所有可用资源收到的不可用在线信息。

```
<presence
      from='contact@192.168.0.5'
      to='user@ XMPPExample.com'
      type='unsubscribed'/>
  <presence
      from='contact@192.168.0.5/resource'
      to='user@ XMPPExample.com'
      type='unavailable'/>
```

步骤 6：当用户的服务器收到类型为"unsubscribed" 和 "unavailable"的在线信息节，它必须递送它们给用户。

```
<presence
      from='contact@192.168.0.5'
      to='user@ XMPPExample.com'
      type='unsubscribed'/>
```

```
<presence
    from='contact@192.168.0.5/resource'
    to='user@ XMPPExample.com'
    type='unavailable'/>
```

步骤 7：接收到"unsubscribed"类型的在线信息节之后，用户应该承认收到那个订阅状态变更通知，要么向联系人发送一个"unsubscribe"类型的在线信息节以证实它，要么向联系人发送一个"subscribe"的在线信息节以否认它；这步骤不影响订阅状态，但是让用户的服务器知道它必须不在发送订阅状态变更通知给用户。

2.3.6 用户之间发送消息流程

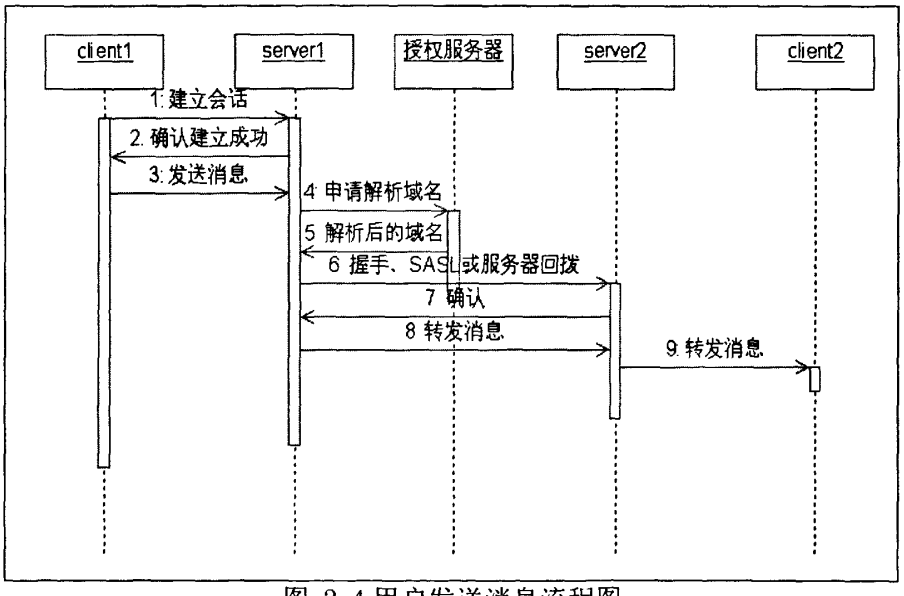

图 2-4 用户发送消息流程图

一个即时消息客户端应该通过提供一个 JID 或<message/>节中不同于发送者的'to'属性来指定一个消息的预定接收者。如果这个消息是在回复之前接收到的消息，而接收到的消息是从 JID 格式为<user@domain/resource>(例如，在一个聊天会话的上下文中)实体发来的，这个回复消息的'to'地址的值应该是<user@domain/resource>而不是<user@domain>，除非发送者知道(通过在线信息)预定的接收者的资源将不再可用。 如果消息是在任何现存的聊天会话或接收到的消息之外被发送的，'to'地址的值应该格式为<user@domain>而不是<user@domain/resource>

```
message
        to='romeo@example.net'
        from='juliet@XMPPExample.com/balcony'
        type='chat'
        xml:lang='en'>
      <body>Where are you from, Romeo?</body>
      <body xml:lang='cz'>proce, Romeo?</body>
    </message>
```

2.3.7 用户之间传送文件的流程

基于 XMPP 协议的文件传输必须经历三个步骤：SOCKS5 流协商、创建 socket 和传送文件字节流，在此期间 SOCKS5 流协商的建立一部分通过 XMPP XML 流，一部分通过一个独立的 socket，实际的文件传输发生在创建的 socket 上。为实现以上功能，XMPP 协议专门定义了一套完整的文件传输的元数据，如下：

file – 表示文件传输报文的根元素

size – 表示发送文件的大小，按字节计算

name – 表示发送者想要发送的文件名

date – 表示文件的最后修改日期，日期格式使用 XMPP Date and Time Profiles 指定的格式。

hash – 文件内容的 MD5 密文

offset – 指定了开始进行文件传输的起点位置，按字节计算。如果没有指定，默认值为 0。

length – 指定了从偏移量开始的要接收的字节数，默认值是从偏移量开始到文件末尾的字节数。

range – 元素中的两个元素都是可选的，发送无属性的元素等同于没有 range 元素，当在流初始化的结果中没有<range>元素时，发送者必须从偏移量为 0 处发送完整的文件，更通常的情况是，从偏移量位置开始根据指定长度通过字节流发送。

其具体的发送流程如下所示：

第一步：文件发送端，发送流协商包至目的端。

```
<iq type='set' id='angle8311f9ef' to='test@163.com/Exodus'>
    <si  xmlns='http://jabber.org/protocol/si'  id=' angle8311f9f0'
        profile='http://jabber.org/protocol/si/profile/file-transfer'>
```

```
<file xmlns='http://jabber.org/protocol/si/profile/file-transfer'
      name='test.txt'     size='4093'/>
 <feature xmlns='http://jabber.org/protocol/feature-neg'>
      <x xmlns='jabber:x:data' type='form'>
      <field var='stream-method' type='list-single'>
        <option><value>http://jabber.org/protocol/bytestreams</value>
        </option>
     </field>
     </x>
 </feature>
</si>
</iq>
```

第二步：文件接收端接收自发送端而来的流协商包，并发送流协商响应包至发送端。

```
<iq id=" angle8311f9ef" to="jjl@163.com/Home" type="result">
    <si id=" angle8311f9f0" xmlns="http://jabber.org/protocol/si">
      <feature xmlns="http://jabber.org/protocol/feature-neg">
        <x type="submit" xmlns="jabber:x:data">
            <field var="stream-method">
          <value>http://jabber.org/protocol/bytestreams</value>
            </field></x>
      </feature>
     </si>
     </iq>
```

第三步：文件发送端接收从文件接收端返回的流协商响应包，创建 socket，绑定一个地址，并监听(记下 IP 与端口号)，等待连接构造出如下的 bytestream 包，并发送该包。

```
<iq type='set' id=' angle8311f9f1' to='test@163.com/Exodus'>
     <query xmlns='http://jabber.org/protocol/bytestreams' sid=' angle8311f9f0'>
         <streamhost jid='jjl@163/Home' host='192.168.100.1' port='6642'/>
     </query>
</iq>
```

第四步：文件接收端接收 bytestream 包。创建一个 socket，并连接 bytestream 包中指定的 host 与 port，并开始 socket 5 协商（其流程如上节所述），等 socket 5 协商完毕，发送 bytestream 响应包。

```
<iq from='test@163.com/Exodus'
    type='result' to='jjl@163.com/Home' id=' angle8311f9f1'>
   <query xmlns='http://jabber.org/protocol/bytestreams'
       sid=' angle8311f9f0'>
        <streamhost-used jid='jjl@163.com/Home'/>
   </query>
```

```
</iq>
```

第五步：文件发送端接收包，并开始文件传输。

发送文件的 XML 流：

```
<message from='juliet@163.com/balcony' to='romeot@montague.net'>
  <sipub xmlns='http://jabber.org/protocol/si-pub'
        id='publish-0123'
        mime-type='text/plain'
        profile='http://jabber.org/protocol/si/profile/file-transfer'>
    <file xmlns='http://jabber.org/protocol/si/profile/file-transfer'
          name='missive.txt'
          size='1024'
          date='2005-11-29T11:21Z'/>
  </sipub>
</message>
```

接收文件的 XML 流：

```
<message from='romeot@montague.net' to='juliet@163.com/balcony'>
  <sipub xmlns='http://jabber.org/protocol/si-pub'
        id='pub234'
        mime-type='text/plain'
        profile='http://jabber.org/protocol/si/profile/file-transfer'>
    <file xmlns='http://jabber.org/protocol/si/profile/file-transfer'
          name='reply.txt'
          size='2048'/>
  </sipub>
</message>
```

2.4 SOCKS5 协议基础

在网络中防火墙的使用非常普遍，它有效的隔离了机构的内部网络和外部网络，这些防火墙系统大都充当着网络之间的应用层网关的角色，通常提供经过控制的 Telnet,FTP,和 SMTP 访问，随着全球信息的交流越来越频繁，更多的新的应用层协议的推出，这就有必要提供一个总的架构使这些协议能够更明显和更安全的穿过防火墙，也就有必要在实际上为它们穿过防火墙提供一个更强的认证机制，这种需要源于客户机-服务器联系在不同组织网络之间的实现，而这种联系需要被控制和是很大程度上被认证的，该协议被描述为用来提供在 TCP 和 UDP 域下为客户机-服务器应用程序便利和安全的穿过防火墙的一个架构。该协议在概念上被描述为一个介于应用层和传输层之间的"隔离层"，但是这类服务并不提供网络层网关服务，如 ICMP 报文的传输[26][41]。

在网络通信的过程中，对于数据传输实时性要求较高，因此大家都会选择UDP来作为数据传输协议，在TCP/IP协议族中UDP协议较TCP协议需要的网络系统资源更少，然而在企业应用中，由于网络安全原因等会导致除了特定端口以外的IP数据无法通过专用的路由或网关。为了支持这类应用，有人制定了专门的支持SOCKS连接的SOCKS5协议。SOCKS协议允许实现此类功能的代理软件可以允许防火墙以内的客户通过防火墙实现对外部的访问，甚至可以允许等待外部的连接。

2.5 本章小结

本章主要介绍了有关基于XMPP协议的即时通信系统的相关背景，包括XMPP协议的基本元素和结构，同时也阐述了XMPP协议在系统中的应用的形式，列举了：注册、登录、发消息等，是人们对XMPP协议的应用有了直观的了解，同时对即时通信系统的实现有了初步的认识，对基于XMPP协议系统的开发关键在于深刻理解协议的设计，只要理解了协议剩余的设计和开发工作将会很方便，因为协议的设计者已经将底层的XML节通信的规则和流程规定好，开发者只需要用面向对象的语言将其实现。

第3章 系统的总体设计和各功能模块的设计

3.1 系统结构

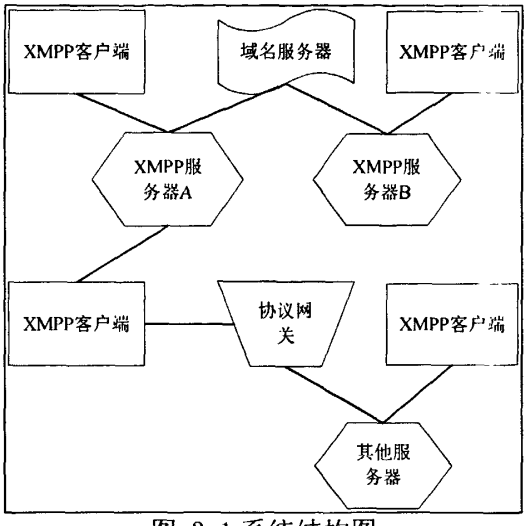

图 3-1 系统结构图

如图 3-1 所示，XMPP 协议采用的是客户端-服务器架构，其中包含了三种通信方式：客户端和服务器，服务器和服务器，客户端和非 XMPP 协议的客户端。

客户端和服务器：有客户端向服务器发送会话请求，服务器收到申请后会检查该客户端是否为本服务器域内的，如果是则开始会话，否则返回错误信息。

服务器和服务器：服务器 A 向服务器 B 发送会话请求时，域名服务器将会解析服务器 B 的地址，则该会话请求会被送到解析后的服务器 B 的地址处，

客户端和非 XMPP 协议的客户端：XMPP 客户端和非 XMPP 客户端之间的通信是通过协议网关进行的，协议网关的作用是进行协议之间的转换，以保证不同协议客户端之间的通信。

3.2 公共模块设计

由于该系统所有的功能实现都是基于网络间的 XML 流的通信，所以，需要有一个模块专门负责网络间通信和 XML 流的处理，主要功能包括服务器和客户

端之间通信时 TCP 套接字的处理，XML 流的解析、存储等功能。

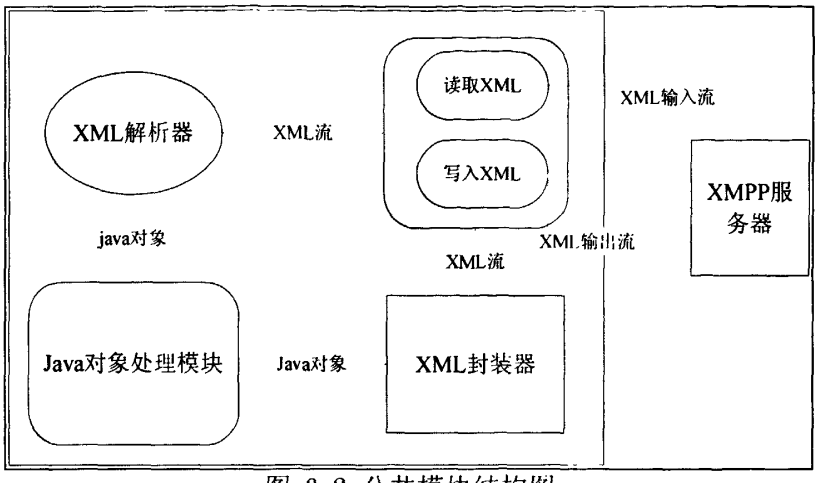

图 3-2 公共模块结构图

本模块可以按功能分为四个子模块：负责客户端与服务器之间连接的会话管理模块，主要功能是：建立连接，断开连接，接收 XML 流，发送 XML 流等；负责 XML 流的解析和封装的 XML 模块，主要功能是：将 XML 流解析成 java 对象，将 java 对象封装成 XML 流；负责对信息进行加密和解密的模块，主要功能是：对所发送的重要信息进行加密和解密；负责信息读取和转发的路由模块，主要功能是：从对应的端口截取数据，并根据其命名空间的不同进行转发。

其流程是 XMPP 服务器接收到 XML 流之后，会有读取器将其读取出来并将其作为入口参数传入 XML 解析器，XML 解析器通过对其命名空间的解析，从而确定将剩余的 XML 元素解析出来并传入相应的 java 对象中，从而最终将 XML 转换成 java 对象，然后将 java 对象传入应用程序模块中，实现其请求完成的功能并返回 java 对象，但是该 java 对象不能在网络中直接传输，必须先转换成 XML 节，于是，该 java 对象会被传入 XML 封装器中，被封装成 XML 节，通过 XMPP 服务器的发送端口发往目的节点。

java 对象处理模块处理流程如下：当该模块接收到 java 对象时，会先将该对象通过解密算法和解密密钥解密成 base64 码，然后再将 base64 码转换成二进制码，从而实现对 java 对象的解析。当完成业务逻辑处理后，该模块会将返回的 java 对象先由二进制码转换成 base64 码，然后用加密算法将其加密，这里的加密算法是由双方在建立会话时通过三次握手协议协商的。

当 XML 节被封装成 java 对象后，必须被转发至正确的模块中加以处理，这

就要求有一个路由转发模块，如图 3-3 所示。该模块的实现原理是：在系统启动时加载该路由模块，从而在内存中创建了一块路由表，记录了命名空间和功能模块之间的对应关系，当 java 对象被封装好之后，系统会读出其命名空间，再在路由表中查找其所对应的模块，从而动态地加载该模块，并将该 java 对象转发至该模块，从而实现路由转发的功能。

3.2.1 会话管理模块设计

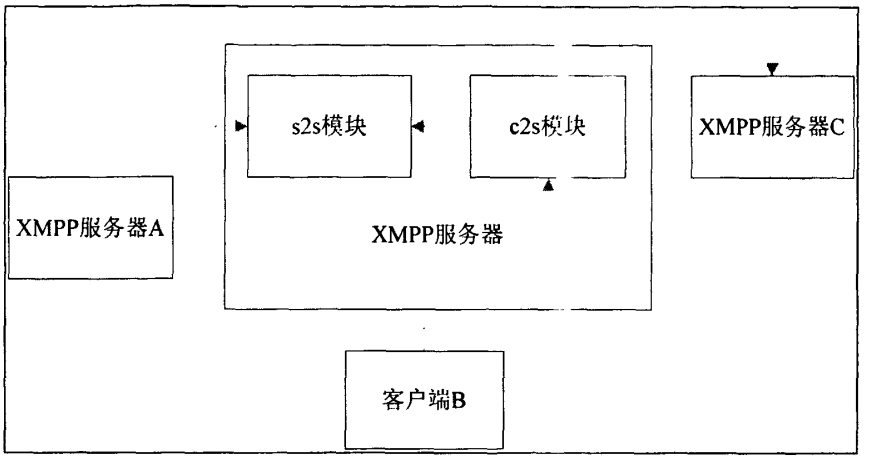

图 3-3 会话管理模块结构图

因为在 XMPP 系统中任何两个客户端之间的通信必须经过服务器，所以在本系统中会话分为两种：客户端对服务器(c2s)和服务器对服务器(s2s)。

在服务器端有两个会话组件，分别用于管理 c2s 和 s2s。其中 c2s 会话管理用于管理客户端和服务器之间的通信，该模块将会监听所指定的端口 5222，如果服务器收到数据包，则解析出命名空间并将数据包发送至该模块(c2s)，该模块收到后会与本地服务器中保存的本地用户信息进行对比，从而确定是否有权限处理该数据包。

s2s 模块用于管理服务器与服务器之间的通信，该模块将会监听服务器与服务器之间通信的端口，如果收到数据包(例如连接请求)，则会对发出数据包的服务器进行服务器回拨以确定该服务器的身份是否合法，并进行密钥的协商。

3.2.2 XML 处理模块设计

正如上一章对协议的描述，XMPP 协议采用的是 XML 格式并且都有固定的

格式，因此对该种 XML 文本的解析比较简单，可以通过定义 java 对象来存储 XML 报文中相关的信息。与解析 XML 报文类似，封装 XML 也很简单。这主要得益于协议对 XML 报文格式的规定，以及 XML 语言本身固有的可扩展特性。

XPP 与普遍使用的 XML 解析器两类技术：DOM(Document Object Model)和 SAX(Simple Api for Xml)不同。

(1)DOM 是基于对象树的方式，即将整个 XML 文件按照其层次结构封装成带有层级关系的对象，SAX 是基于事件流方式，而 XPP 是 streaming incremental 的方式。

(2)DOM 和 SAX 在解析 XML 时是不允许中断的，因为它们没有可靠的续传机制；而 XPP 是可以在任何时候中断，并且允许重新开始，这是因为该种方式提供了可靠的续传机制。

(3)SAX 与 DOM 都是基于“推模式”的解析，虽然“推模式”解析易于使用，但处理较大的 XML 文档时效率并不好。

3.2.3 路由模块设计

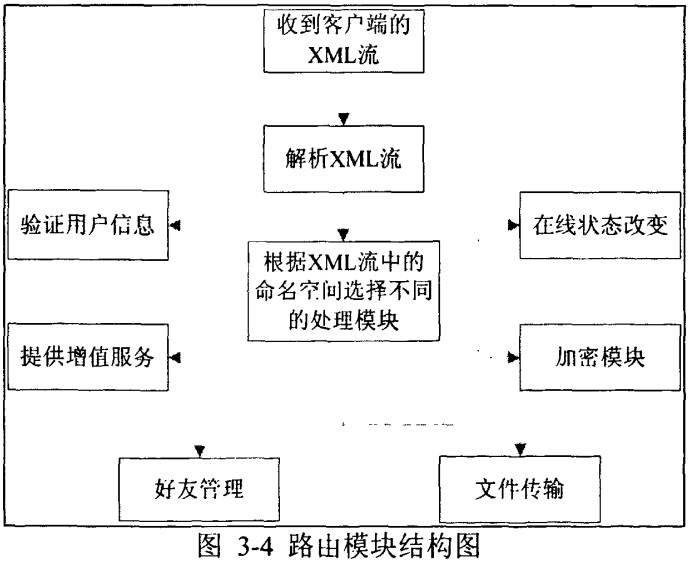

图 3-4 路由模块结构图

本系统服务器设计是基于模块化的思想，这样方便了系统的扩展减小了系统的开销，但同时也产生了一个问题，即不同模块的调用，为此 XMPP 协议引入了命名空间这一概念，命名空间往往是一个唯一确定的标示符(比如一个域名)，一个命名空间对应了唯一的一个系统模块。在网络中传输的报文都是以命

名空间标示的，因此在服务器端设计时就必须实现命名空间和不同的功能模块之间的路由。

3.2.4 加密模块设计

由于 XMPP 协议规定是以 XML 语言传输的，XML 语言本身非常易于解析和阅读的，所以一旦在传输的过程中 XML 报文被监听或截获，后果将不堪设想，因此，XMPP 系统引入了加密机制，该加密过程是在 XML 报文封装之前完成的，而且是对重要信息的加密，例如：用户名、密码等。

该加密机制是分两步完成的，首先将二进制码转换成 BASE64 码。

BASE64 是 MIME 邮件中常用的编码方式之一。它的主要思想是将输入的字符串或数据编码成只含有{'A'-'Z', 'a'-'z', '0'-'9', '+', '/'}这 64 个可打印字符的串，故称为"BASE 64"。 BASE 64 编码的方法是，将输入数据流每次取 6 bit，用此 6 bit 的值(0-63)作为索引去查表，输出相应字符。这样，每 3 个字节将编码为 4 个字符(3×8 → 4×6)，前两位用零代替。例如：10011011 11100010 01011010 转换后变成 00100110 00111110 00001001 00011010

在转换成 BASE64 码之后，再进行加密，加密算法由各网络实体建立会话时通过 TLS 协商，例如：MD5 等。

3.3 各功能模块的设计

本文所设计的即时通信系统服务器包含了传统即时通信系统的所有功能：用户注册，用户登录，好友名册，分组名册，添加好友，删除好友，获取好友在线信息，用户身份认证，向好友发送信息，文件传输等。同时由于 XMPP 服务器有模块化的特点，可以根据需要进行扩展。

3.3.1 用户注册功能设计

其流程图如下：

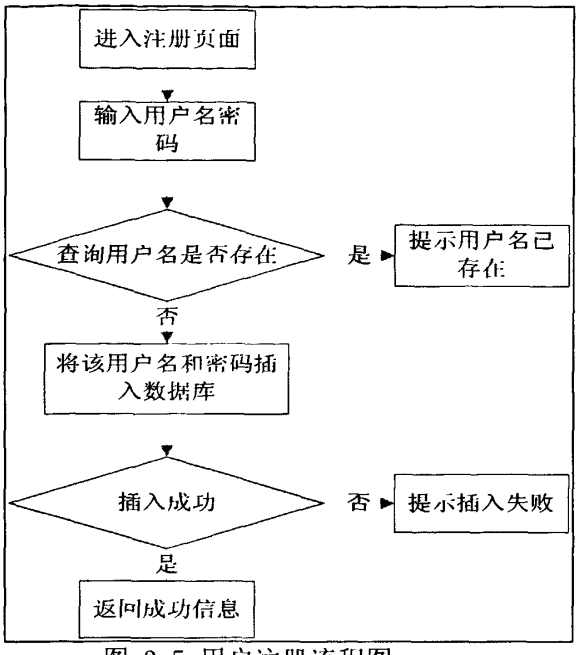

图 3-5 用户注册流程图

用户在使用该系统之前必须先注册用户名和密码，因此，必须有一个用户注册模块，其处理流程图如图 3-5 所示。在客户端用户注册窗口，由用户输入用户名、密码、邮箱地址等信息，并提交给服务器；服务器接收到以上信息后，会查询数据库中是否有该用户名，如果有，则提示该用户已经存在；如果没有，则将其插入数据库中，并向客户端返回注册成功的信息。

3.3.2 用户登录功能设计

其流程图如下：

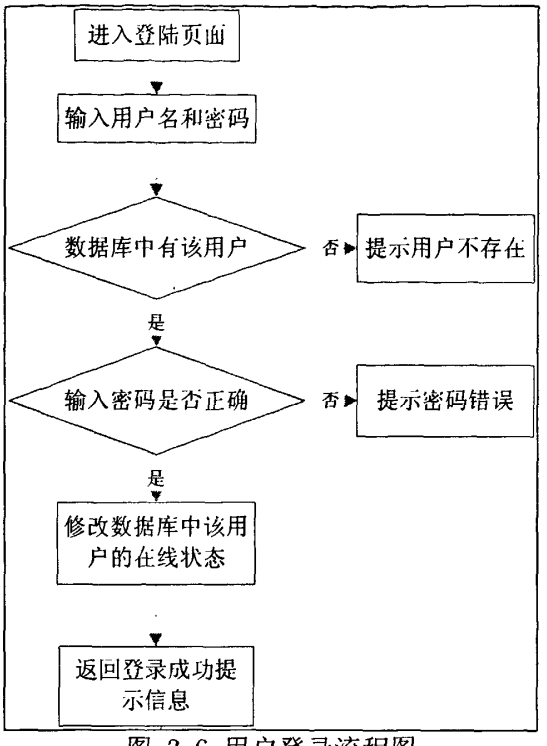

图 3-6 用户登录流程图

用户要使用该系统之功能必须先登录系统，在登录的过程中会对用户信息进行鉴别，以保证用户的使用权限，其流程图如图 3-6 所示。用户在客户端输入用户名和密码并提交至本地服务器，服务器在接收到以上信息后会解析出用户名和密码，并到数据库中查询，如果没有该用户名则返回错误并提示该用户不存在，否则表示该用户名存在，同时核对所输入的密码是否正确，如果错误则返回错误信息，否则，返回登录成功信息。登录的过程类似于前面协议中所讲的会话的建立，相当于某客户端和服务器间进行通信前建立连接的过程。

3.3.3 获取好友在线状态功能设计

其流程图如下：

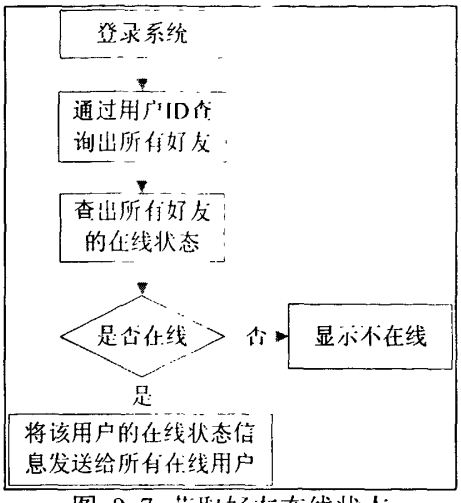

图 3-7 获取好友在线状态

该模块负责管理用户的好友列表以及获取好友的在线状态信息，该模块实现了整个系统最重要的功能，其它模块功能实现都是基于该模块，是 XMPP 协议的精髓，其流程如图 3-7 所示。用户登录系统后，服务器会根据该用户的 JID 查询出该用户的好友列表，如果好友列表为空则表示还没好友加入，否则将所有好友的信息返回给客户端，同时向所有好友发送该用户的在线状态信息，并将以上所有信息返回至客户端。

3.3.4 添加好友功能设计

添加好友功能是该系统非常重要的一项功能，其流程如图 3-8 所示。用户将想要添加的好友的 JID 发送至服务器端，服务器会在该用户的好友列表中查询该 JID，如果存在，则提示该用户已经是好友，否则在数据库中查询该用户的信息，如果不存在则返回错误信息提示该用户不存在，否则将该用户添加入该用户的好友列表中，并返回给客户端。当被添加的用户登录进入系统时，系统会自动弹出提示框，显示用户的加为好友的请求，然后询问是否同意加为好友，如果同意则添加好友成功，否则添加失败。

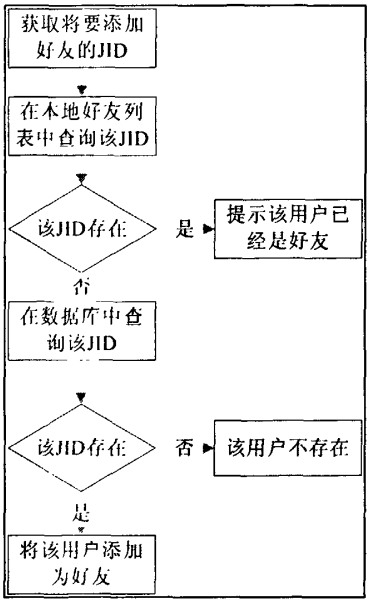

图 3-8 添加好友流程图

3.3.5 删除好友功能设计

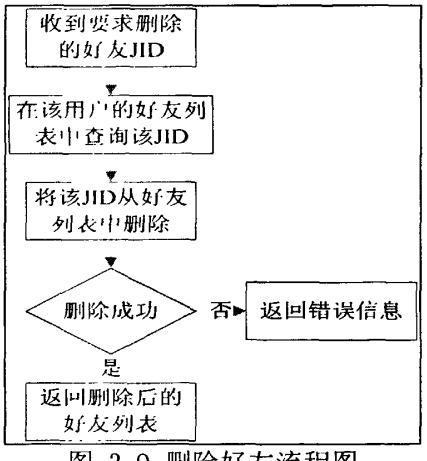

图 3-9 删除好友流程图

删除好友也是系统的一项重要功能，其作用是删除好友列表中的某一用户，其流程图如图 3-9 所示。用户从客户端向服务器发送要删除的用户的 JID，服务器收到该 JID 之后，会在该用户的好友列表中查询该用户，并将其删除，如果删除时出现异常，则返回错误信息给客户端，否则返回删除成功信息。同时应该注意的是，这里的删除只是将该用户从本地名册中删除，并不能将其从数据库

中删除。

3.3.6 给好友发送消息功能设计

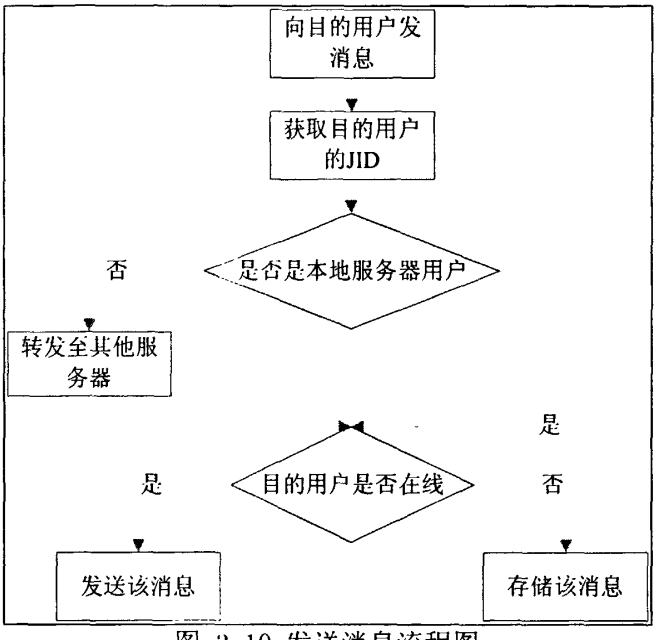

图 3-10 发送消息流程图

各个客户端之间发送文本信息是该系统的基本功能，其流程图如图 3-10 所示。源客户端经过服务器向目的客户端发送消息，服务器在收到源客户端发送来的信息时，从中解析出目的客户端的 JID 进而判断目的客户端是否属于本地域，如果不是则通过因特网转发至目的客户端所属的服务器，否则，判断目的客户端的在线状态，如果在线则发送消息，否则，存入数据库中，待目的客户端上线后再将该信息转发。

3.3.7 退出系统功能设计

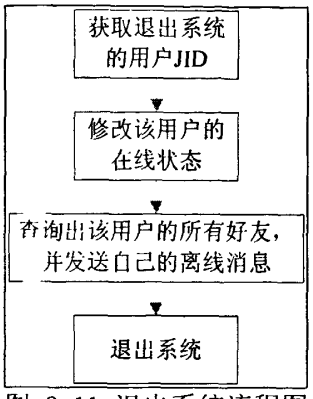

图 3-11 退出系统流程图

当用户退出系统时，由客户端向服务器端发出退出请求，服务器端接收到请求后，会修改数据库中该用户的在线状态，并向该用户的所有好友发送其在线状态改变信息，流程如图 3-11 所示。

3.4 文件传输模块设计

3.4.1 SOCKS5 协议的 XMPP 扩展

本系统除了支持文本信息通信之外，还扩展了文件传输功能，方便企业用户直接进行文本文件的传输，其中文件传输的代理服务器通常集成于 XMPP 服务器中。本系统的文件代理服务器采用的是 SOCKS5 协议，该协议是在 SOCKS4 协议基础上进行了改进，传统的 SOCKS4 协议只能支持 TCP 协议的代理，而经过改进后的 SOCKS5 协议则支持 TCP 和 UDP 的代理，本系统采用了 SOCKS5 协议，从而实现了对不同协议的支持。一般的代理软件都实现了两个版本的 SOCKS 协议—SOCKS4 以及 SOCKS5，其中 SOCKS5 协议支持 UDP 报文的传输以及多种验证方法，但是现存的 SOCKS5 协议并不能直接与 XMPP 协议集成，必须对其进行扩展以使其支持 XMPP 协议。其扩展元素如下：

<query/>元素

<query/>元素是所有带内元素相互之间交互的容器。这个元素必须是在"http://jabber.org/protocol/bytestreams"命名空间内。此元素有个表示流会话标识的属性，并包含了多个<streamhost/>元素，一个<streamhost-used/>元素或者一个

<activate/>元素。

"sid"指明了字节流会话的唯一标识符，该属性必须出现在报文中，属性值可以是任何字符。

"mode"属性指明了使用的传输模式，可以是'tcp'或者'udp'。如果没有属性，那么必须假设默认值是"tcp"。

<streamhost/>元素是负责传送网络连接的信息。在初始方发送给目标方的初始 IQ-Set 中，至少有一个元素的实例出现。如果有多个实例出现，每个必须是独立的主机/端口组成。

<streamhost-used/>元素指明正在使用的带外传输的流主机。它必须在从目标方发送给初始方的 IQ-Set 中出现，并且必须只有一个唯一的实例。

<activate/>元素是用来请求激活双向的或者单向的字节流。它必须在从初始方发送给流主机的 IQ-Set 中出现，而且是在从目标方接收到 IQ-result 之后，并且必须只有一个唯一实例。

<streamhost/>元素

<streamhost/>元素包含了字节流连接的信息，有表示流主机的 JID 属性，网络主机/地址，网路端口。该元素不能包含任何文本节点。

"jid"属性指明了流主机的 JID。该属性必须出现，并且必须在<iq/>中是合法的 JID。

"host"属性指明了要连接的主机。该属性必须出现。该属性的值必须是可分解的域名或者是"带点的十进制"IP 地址(比如 11.24.56.21)。

"port"属性指定了要连接的端口。该属性可以出现。该属性的值必须是十进制的合法的数字。

"zeroconf"属性指定了对于字节流可用的 zero-configuratio 服务。属性应该出现。该属性值应该是 '_jabber.bytestreams'。

当正在连接可用的主机时，初始方必须要么包含主机和端口要么包含 zeroconf 信息。

<streamhost-used/>元素

<streamhost-used/>元素指定了连接的流主机。该元素有个目的方连接的流主机的 JID。该元素不能包含任何文本节点。

"jid"属性指定了流主机的全 JID。该属性必须出现，并且必须在<iq/>中是合法的 JID。

<activate/>元素

<activate/>元素是一个用来触发代理完成连接的标志。

在传输文件的字节流之前必须进行流协商，流协商的过程就是发送端和接收端通过代理服务器协商连接端口、连接地址和建立会话的过程，其流程如下：

1．初始方发送 IQ-Set 节点给指定了包含全 JID 的目标方，以及流主机/初始方的网络和字节流的流 ID(SID)。

2．目标方打开一个 TCP 套接字连接到指定的网络地址。

3．目标方通过 SOCKS5 请求连接，请求参数设置为下面定义的 DST.ADDR 和 DST.PORT。

4．流主机/初始方发送成功通过 SOCKS5 连接的应答。

5．目标方发送 IQ-result 节点给初始方，该'id'与初始方发送的 IQ-Set 相同。

6．流主机/初始方激活字节流。

初始方请求激活流

```
<iq type='set'
      from='initiator@XMPPExample.com/foo'
      to='streamhostproxy.com'
      id='activate'>
   <query xmlns='http://jabber.org/protocol/bytestreams' sid='mySID'>
      <activate>target@example.org/bar</activate>
   </query>
</iq>
```

使用了数据包中的 SID 以及 from 地址这些信息后，代理才能根据 SID + Initiator JID + Target JID 的哈希值激活流。并提供一个合理的信任等级激活来自初始方的请求。

如果代理能完成请求，它必须使用 IQ-result 响应初始方

```
<iq type='result'
      from='streamhostproxy.com '
      to='initiator@XMPPExample.com/foo'
      id='activate'/>
```

3.4.2 文件传输功能设计

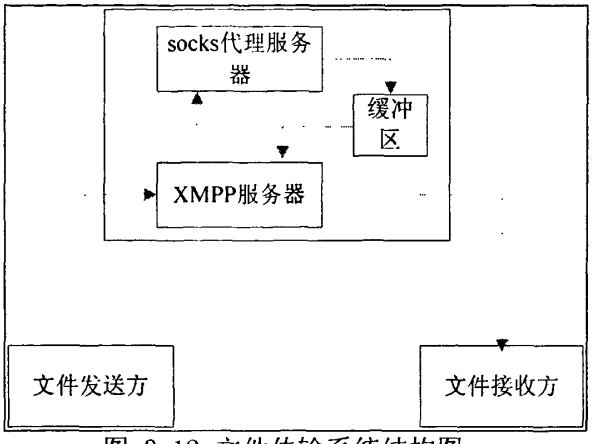

图 3-12 文件传输系统结构图

建立会话过程：

1. 会话发起者找到 SOCKS5 代理服务器地址。
2. 会话发起者在发送 IQ 节至目的端时，在 IQ 节中指定了代理服务器的完整 JID 和网络地址，代理服务器是支持字节流的 SID。
3. 目的端为选定的代理服务器打开一个 TCP socket 连接。
4. 代理服务器通过 SOCKS5 建立连接，设定目的端地址和目标端口参数的值。
5. 代理服务器通过 SOCKS5 发送连接成功确认至目的端。
6. 目的端发送 IQ 应答信息至发起者，并保留最初 IQ 中的 id 值。
7. 发起者在代理服务器打开一个 TCP socket 连接。
8. 发起者通过 SOCKS5 建立连接，设定目标地址和目标端口参数的值。
9. 代理服务器通过 SOCKS5 发送连接成功确认至发起者。
10. 发起者发送 IQ 节至代理服务器，请求代理服务器激活关联 StreamID 的字节流。
11. 代理服务器激活字节流，数据现在由代理在两个 SOCKS5 连接间转发。
12. 代理服务器发送 IQ 节至发起者，如果连接成功则答复字节流已经激活，否则指明一个错误。
13. 发起者和目的端可以开始使用字节流。

当 SOCKS5 代理服务器接收到发送方发过来的数据报时，接收方会将这些数据报按照接收的时序存放在一个缓冲区中，在接收的过程中会对每一个数据报进行鉴定，将那些不符合要求的数据报丢弃，并按照认证中规定的封装方式

进行封装，而同时该缓冲区必须从 SOCKS 代理服务器获得目的客户机的 IP 地址和端口。此时该缓冲区在接受数据报时必须丢掉那些与目的客户机 IP 和端口不符的数据报，从而实现对数据报的过滤和纠错。

在接受的报文中 FRAG 域指出了数据报是否为某一数据片中的一片。如果标明了 FRAG 域，则根据数据报所处的位序来确定其重组的顺序。值介于 1-127 之间的数据报片断位于数据片序列中间，每一个文件接收端都有一个和这些数据片相关的重组队列表和一个重组时间表。重组队列必须被再次初始化并且相关联的数据片必须被丢掉，而无论该重组时间表是否过期，或者一个新的携带 FRAG 域的数据报到达，并且 FRAG 域的值要小于正在进行的数据片序列中的 FRAG 域的最大值。

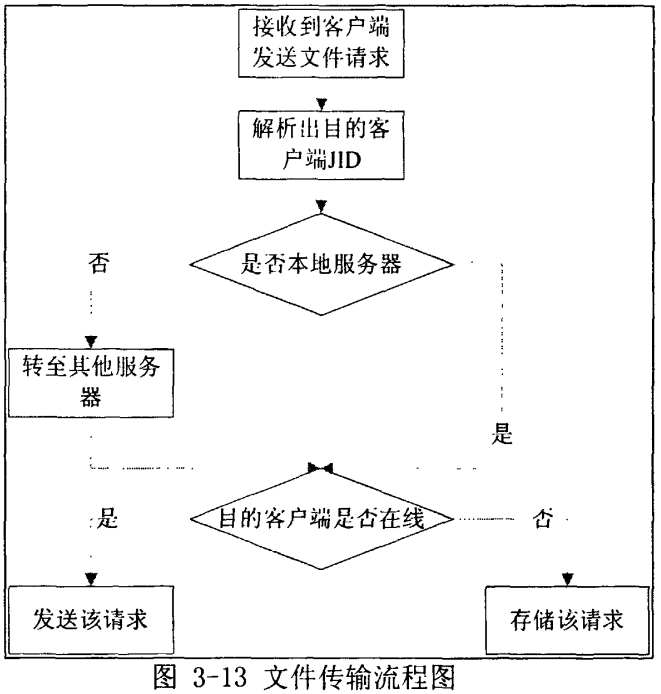

图 3-13 文件传输流程图

该系统实现了文件传输功能，允许一个客户端向另一个客户端传送文本文件，其流程图如图 3-13 所示，服务器在接收到一个客户端的请求后，将其 XML 节解析出来，并获取到目的客户端的 JID，并判断该 JID 是否属于本地服务器，如果属于本地服务器则直接查看该客户端的在线状态，如果在线，则转发该请求，否则存储该请求，知道目的客户端的在线状态为上线。

3.5 数据库设计

系统中的数据信息都是在数据库中存储的，主要包括：企业部门的信息，主要记录企业各个部门的人员组成，以及部门的功能等描述性信息；用户信息，主要记录各个用户的个人信息；服务器信息，主要记录服务器的相关配置信息。

数据包信息，主要记录数据包的相关信息。

3.5.1 企业部门信息

部门基本信息表：

用来存储企业部门的基本描述性息

表 3-1 部门基本信息表

departmentName	部门名字(主键)
departmentDescription	部门描述

部门用户表：

用来记录每个部门的用户成员

表 3-2 部门用户表

departmentName	部门名字(外键)
Username	用户名(外键)
isAdministrator	是否为该部门管理员

部门属性表

用来记录部门的属性，其属性可以由管理员添加。

表 3-3 部门属性表

departmentName	部门名字(外键)
propertyName	部门属性名
propertyValue	部门属性值

3.5.2 用户信息

用户基本信息表：

用来记录用户的基本信息。

表 3-4 用户基本信息表

username	用户名(主键)
password	密码
realname	真实姓名
email	邮箱地址
create_time	创建日期
last_modify_time	最近修改时间

用户名册表：

用来记录用户的好友的情况

表 3-5 用户名册表

userRosterId	名册 ID(主键)
Username	用户名
Jid	被订阅人的 ID
isSub	订阅关系
Nickname	被订阅的昵称

名册部门表：

用来记录用户自己的好友进行分组的信息

表 3-6 名册部门表

userRosterId	名册 Id(主键)
Rank	部门等级
groupName	显示的部门名称

用户属性表：

用来记录用户的属性，其属性可以自由添加

对用户的权限控制是通过该表实现

表 3-7 用户属性表

Username	用户名(主键)
propertyName	用户属性名
propertyValue	用户属性值

离线信息表：

用来存储用户在离线状态下收到的信息

表 3-8 离线信息表

Username	目的用户名(主键)
messageId	离线信息 Id
create_Date	离线信息创建时间
message_Size	离线信息大小
messageContent	离线信息内容

用户状态设置表：

用来记录用户当前接收信息的设定状态

表 3-9 用户状态设置表

Username	用户名(主键)
realName	真实姓名
isDefault	是否为默认设置
List	黑名单

用户名片表：

用来存储有关用户的显示信息

表 3-10 用户名片表

Username	用户名(主键)
vcard_value	名片信息

3.5.3 服务器信息

服务器属性表：

用来记录服务器的相关信息和设置

表 3-11 服务器属性表

propertyName	服务器属性名(主键)
propertyValue	服务器属性值

3.5.4 数据包信息

数据包表：

用来记录数据包的相关信息

表 3-12 数据包表

Type	访问类型(IQ、Message、Presence)
From	信息发送者
To	信息接收者
Detail	信息中真正要发送的数据

3.5.5 其他表项

系统中还有有关多人聊天、数据库版本控制等几个表，这里不再一一介绍。

3.6 系统客户端设计

本同客户端的设计相对比较简单，客户端只须负责提供用户操作窗口、发送请求、连接服务器端和相应客户端。其主要任务如下：管理与服务器的连接、提供用户操作的窗口、向服务器发送请求、解析来自服务器的数据流等。

3.6.1 客户端操作流程

1．向用户显示有注册和登录功能的窗口，其中还有找回密码，设置服务器地址等功能按钮。

2．用户设置服务器地址，并在登录框中输入用户名和密码，使客户端和服务器开启一个会话，通过一个 XMPPConnection 建立套接字连接与服务器通信，期间将进行三次握手协议以及 SASL 认证用于确认客户端的合法性，一旦通过以上验证程序客户端将与服务器建立连接，然后开始通过 XML 节实现通信。

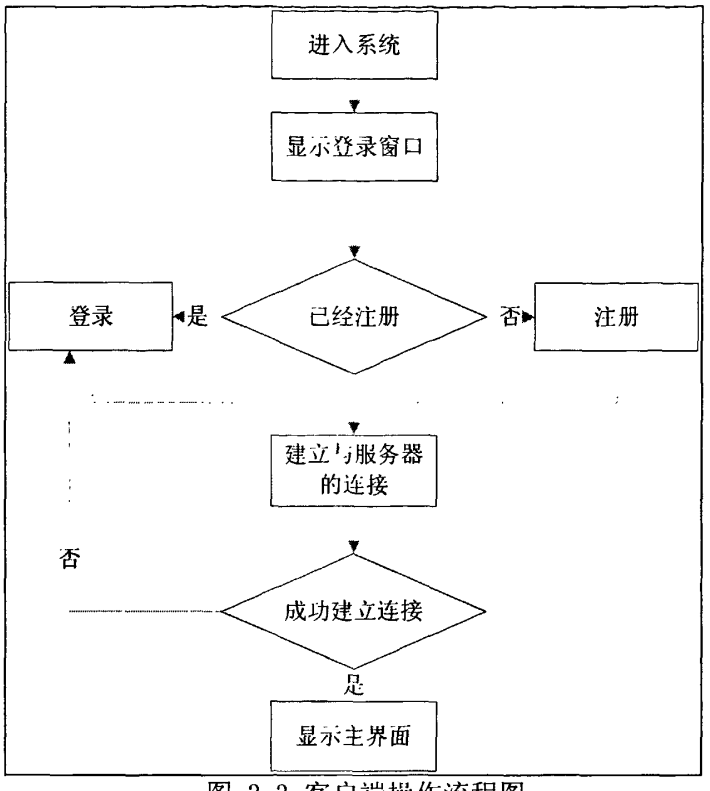

图 3-3 客户端操作流程图

3. 客户成功登陆/注册后，显示主窗口，其中包括用户的好友列表，以及不同的部门，显示同事的状态(在线、离线、外出等)，用户可以进行添加/删除联系人等操作。用户通过窗口操作时，系统会将用户发送的请求和数据通过 XML 处理模块封装成 XML 流并发往服务器。

4. 用户登录之后，客户端于服务器建立 TCP 套接字连接，服务器向客户端发送 XML 流以响应客户端的请求，客户端收到 XML 后，后通过 XML 解析模块将 XML 流解析成 java 对象。解析完成之后，系统根据解析得到的关键字和命名空间确定所要调用的模块，并作相应的处理，从而刷新客户端界面完成用户请求的功能。

3.7 本章小结

本章主要介绍了基于 XMPP 协议的即时通信系统服务器端和客户端的功能设计以及流程设计，由于许多功能的实现已经在 XMPP 协议中有所体现，所以本章介绍的重点偏重于用户体验的角度而不是底层的 XML 节的通信。同时鉴于

各个功能模块的相似性，所以在此只选择了其中几个具有代表性的模块分客户端和服务器端分别进行分析设计。此外还介绍了数据库表的设计，鉴于数据库表数量庞大，这里只选择了几个与前文介绍的模块相关的表。

第4章 即时通信系统的实现

通过上一章的介绍，我们对该系统主要功能在服务器端和客户端的工作原理有了初步的认识，在这一章将主要介绍系统的实现，包括公共模块的实现、各基本功能模块的实现、以及扩展功能模块的实现。前两章分别从底层通信和上层用户体验的角度介绍了该系统的设计，而本章所介绍的实现，实际上是将前两章所介绍的内容以面向对象的程序设计语言实现(本系统采用 java 语言)。

4.1 公共模块的实现

4.1.1 XML 节解析模块的实现

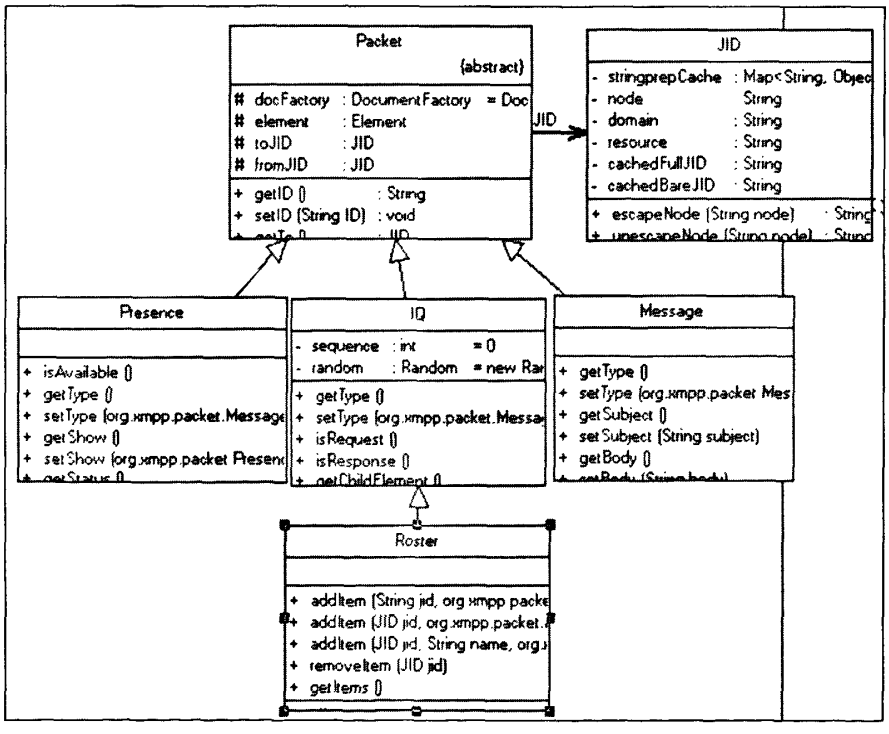

图 4-1 解析模块类图

图 4-1 是 XML 节解析模块的类图，从中可以看出其实现类的层次结构，Packet 类负责封装 XML 节的基本属性设置和获取 JID，设置和获取目的地址，最终通过这些方法设置的信息都会通过 toXML()方法转换成 XML 节。

Presence、IQ、Message 和 Roster 类都是在 Packet 类的基础上进行功能的扩展，从而实现对 XML 节中相关信息的设置和获取。

getID()和 setID()：用来获取和设置目的客户端的地址也就是 JID

getType()和 setType ()：用来获取和设置所有执行的操作的种类

getBody()和 setBody ()：用来获取和设置消息体

getSubject()和 setSubject ()：用来获取和设置消息的内容

4.1.2 数据库连接模块的实现

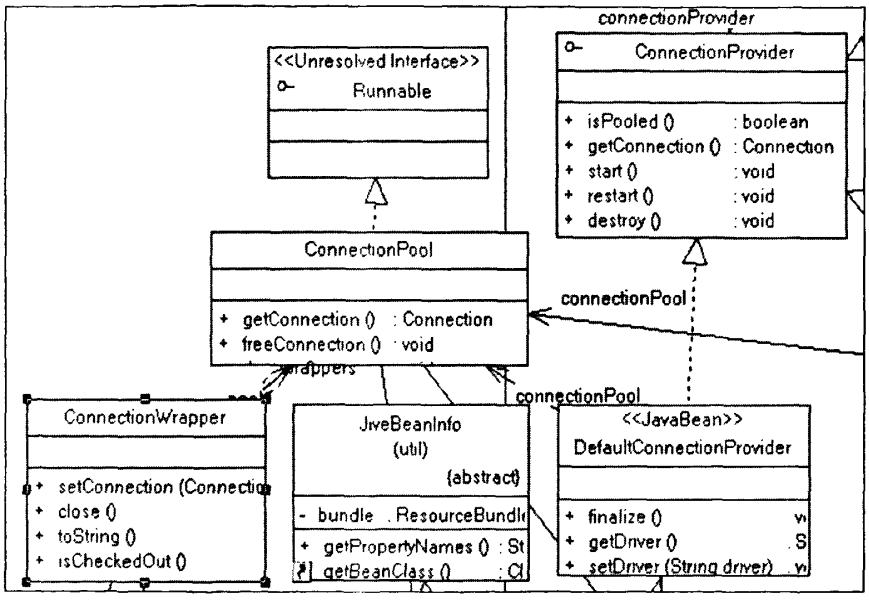

图 4-2 数据库连接类图

图 4-2 列出了数据库连接模块的类图，本系统提供了数据库连接池的功能，所有的数据库连接都由连接池来管理，负责数据库连接的创建，关闭和回收其中 ConnectionProvicer 类负责数据库连接的管理，其子类 DefaultConnectionProvicer 类中提供了数据库驱动、用户名、密码和数据库的 URl 等相关信息的设置和获取。

getDriver()和 setDriver()：用于获取和设置数据库驱动程序

getConnection()和 setConnection ()：用于获取和设置数据库连接

start()、restart()和 destroy()：用于管理数据库连接的生命周期

4.1.3 会话管理模块功能实现

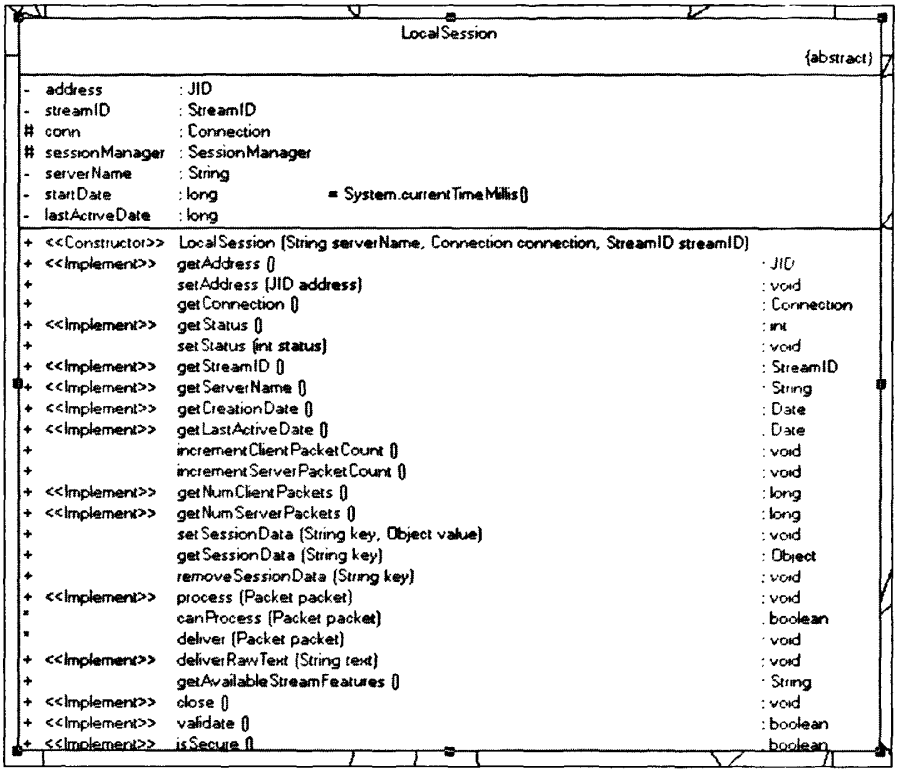

图 4-3 数据库连接类图

图 4-3 列出了数据库连接模块的类图，会话管理模块提供了一整套 XMPP 实体间会话所需的组件，包括监听端口、获取 IP 地址、统计待处理数据包、获取服务器名称、获取 session 信息、客户端和服务器间的认证和鉴权等功能。

public void incrementServerPacketCount()

该方法用于增加服务器端接收到的数据包的数量。

```
public LocalSession(String serverName, Connection connection, StreamID
streamID) {
        conn = connection;
        this.streamID = streamID;
        this.serverName = serverName;
        String id = streamID.getID();
        this.address = new JID(null, serverName, id, true);
        this.sessionManager = SessionManager.getInstance();
    }
```

该构造函数 定义了一个会话所具有的属性，包括采用的 XMPP 连接、服务

器名、XML 流的 ID 号以及返回的会话管理类(SessionManager)的实例。

4.1.4 路由模块功能实现

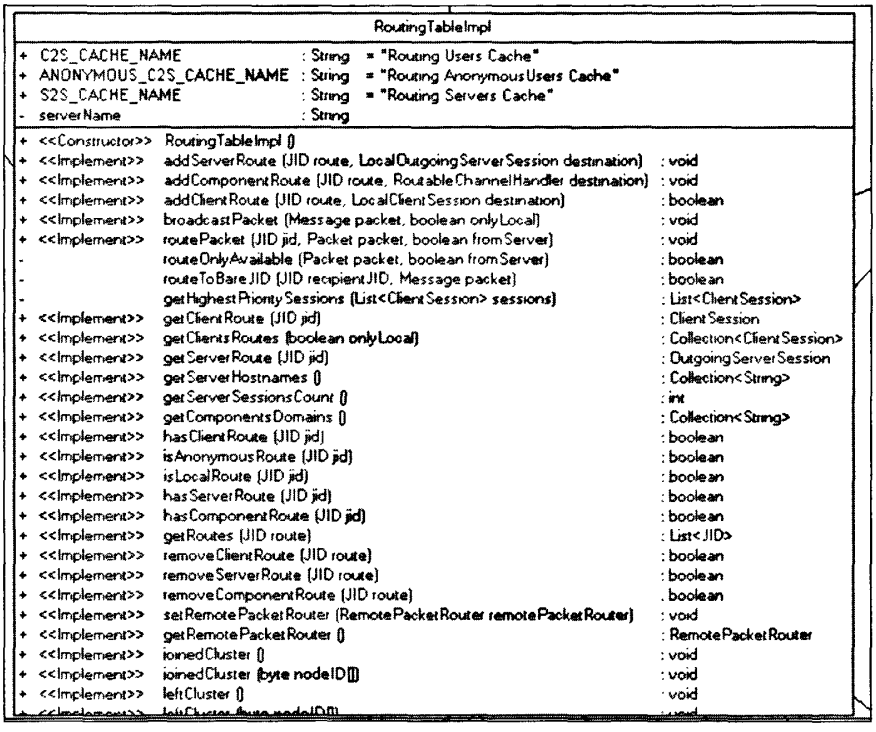

图 4-4 路由功能实现类图

系统路由功能的实现依赖于路由表的实现，路由表实际上是类似于哈希表的数据结构，用于存储命名空间和系统模块的一一对应关系，当系统服务器启动时，系统会在内存中开辟一片空间，用于将路由表加载入其中，当系统将接收到的 XML 报文中的命名空间解析出来之后，再在内存中查找对应的模块，找到对应的功能模块之后服务器会动态地加载该功能模块的代码，从而实现对不同模块的调用。同时，该路由模块还负责各种报文的转发。

上图中则列出了所有有关路由操作的实现方法，其中主要的方法介绍如下：

routePacket(JID jid, Packet packet, boolean fromServer)

该方法的功能是将从某服务器或客户端收到的报文(packet)路由至相应的服务器端的模块中。

routeToBareJID(JID recipientJID, Message packet)

该方法的功能是将从某服务器或客户端收到的消息(Message packet)路由至

一个纯 JID 中，一个纯 JID 表示一个服务器地址或者一个聊天室的地址。

broadcastPacket(Message packet, boolean onlyLocal)

该方法的功能是将从某个消息(Message packet)路由至每一个网络实体中，其中 onlyLocal 表示是只发往本地实体还是发往所有的网络实体中

4.2 服务器端实现

在本章将着重介绍，服务器端各个模块功能的实现，主要是其类图的介绍，和相关关键类的功能介绍。

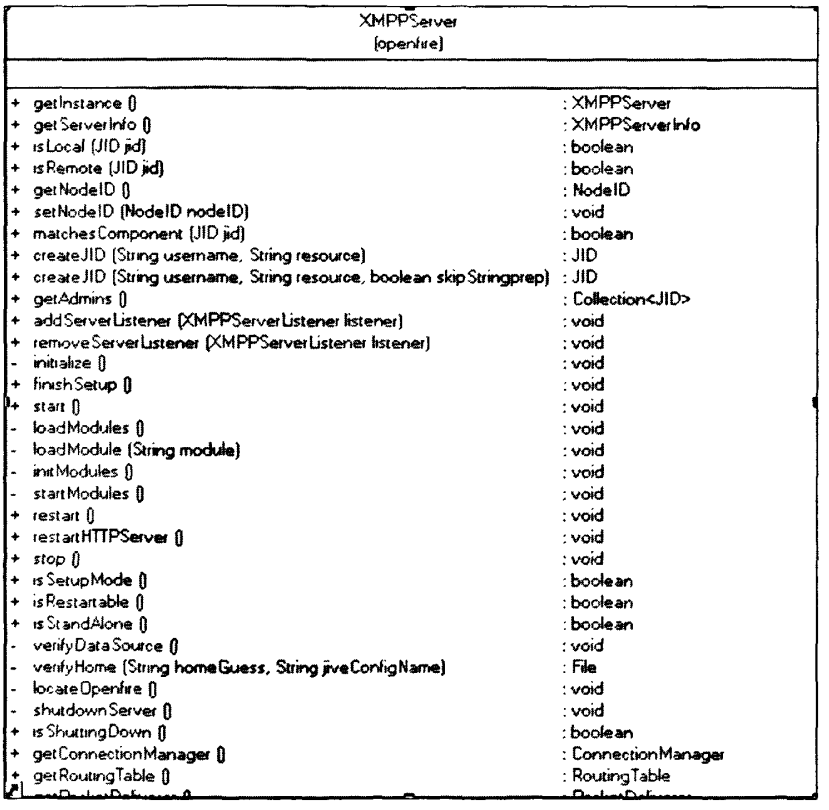

图 4-5 服务器启动类类图

图 4-5 列出的是服务器端的实现类，其中包含了实现本系统的最重要的方法，负责服务器平台的运行周期控制包括启动、重启和关闭，负责服务器端口的设置以及对端口的监听，负责对 JID 的判断以及 XML 节的转发等，是整个服务器端系统的实现类。

start()、restart()、stop()、shutdownServer()、restartHTTPServer()是用来管理

服务器的运行状态，实现对服务器的控制。

initModules()、loadModules()、startModules()是用于实现对服务器组件的动态加载。

4.2.1 用户注册功能实现

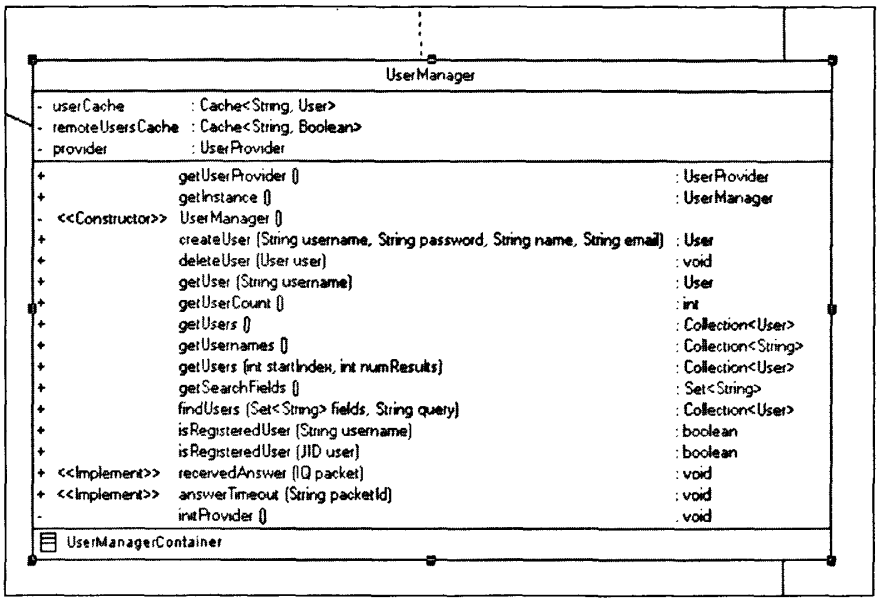

图 4-6 用户注册类图

图 4-6 给出了服务器端实现用户注册功能的实现，其中 createUser 方法是其功能实现的核心方法，通过获取 XML 节中传递的用户名、密码、昵称和 E_mail，并将其插入数据库中从而实现注册功能。

getUser(String username)、getUsernames()、isRegisteredUser(JID user)以及getUsers()方法用于实现对用户名的查询和验证功能。

4.2.2 用户登录功能实现

图 4-5 列出了用户登录模块实现的类图，为了保障系统的安全性系统的密码是经过两层加密后形成的，先是将普通密码转换成 Base64 格式的，然后再对其进行二次加密，其加密机制由客户端和服务器商定，一般情况下都是采用 MD5 加密。登录模块最核心的实现方法是 authenticate，通过传入用户名和密码实现其用户身份的鉴定。

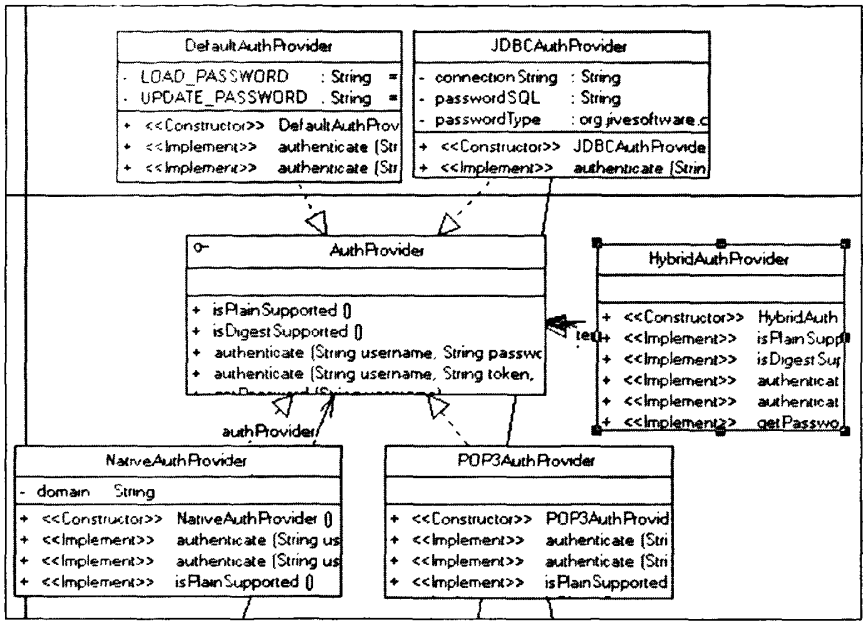

图 4-7 用户登录类图

4.2.3 名册管理功能实现

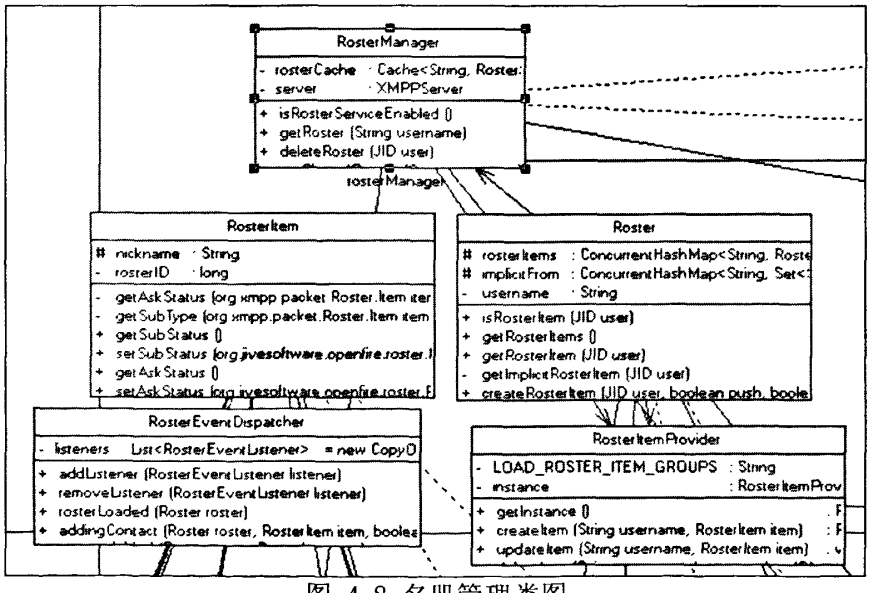

图 4-8 名册管理类图

图 4-8 列出了名册(Roster)管理模块，在系统通信中对名册的操作，包括增、删、查操作，在客户端的反映就是查找用户、添加好友、删除好友等功能，每

一个用户都对应一个名册，名册包含了经过分组的好友列表。

rosterLoaded(Roster roster)该方法用于导入指定的花名册；addContact(Roster roster, RosterItem item)用于向指定的花名册中添加一个条目，每个条目都对应了一个新用户；createItem(String username,RosterItem item)和 updateItem(String username,RosterItem item)用于更具用户名创建或更新条目。

4.2.4 文件传输模块功能实现

文件传输代理类图：

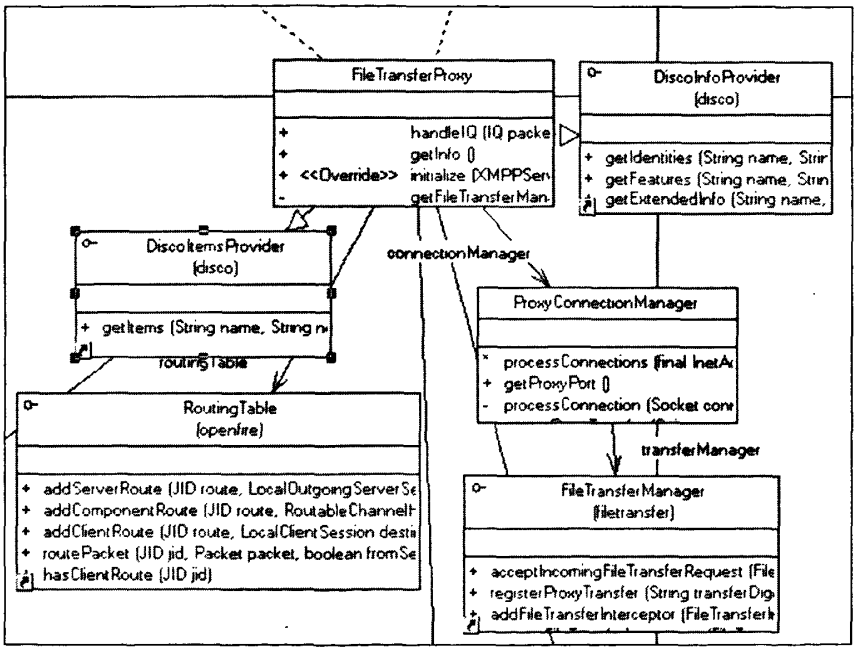

图 4-9 文件传输代理类图

本文件传输的代理服务器是集成于 XMPP 服务器中，它的作用是负责与客户端的会话以及所接受到的文件信息的过滤和路由转发。

processConnection(Socket connection)该方法用于处理客户端与代理服务器之间 socket 连接的建立。

routePacket(JID jid,Packet packet)该方法用于实现 packet 包的路由和转发。

addFileTransferInterceptor(FileTransferInterceptor interceptor)该方法用于添加文件传输的拦截器，实现对文件的过滤。

acceptIncommingFileTransferRequest(FileTransferRequest request)该方法用于

接收并处理客户端所发送的连接请求。

文件传输类图：

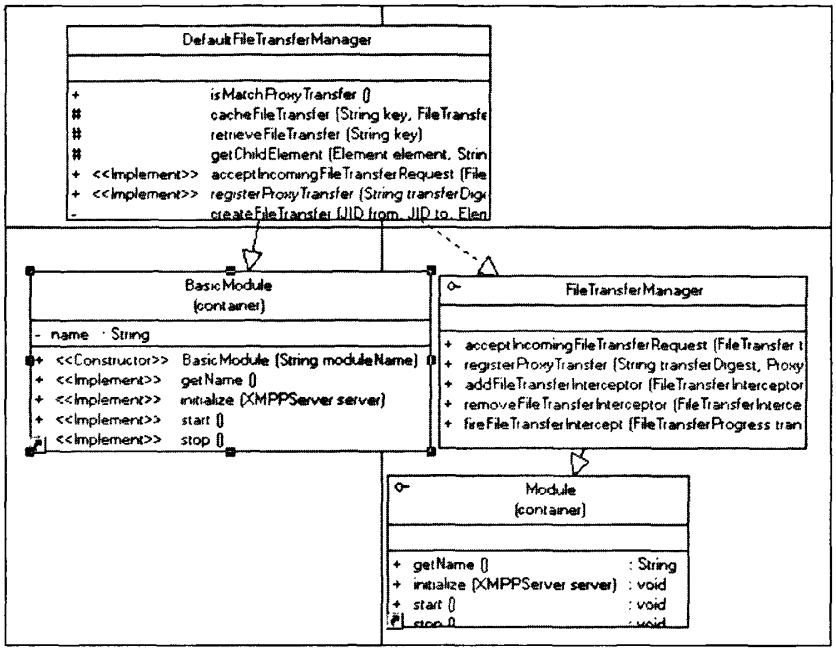

图 4-10 文件传输类图

文件传输模块实现相对简单，主要是文件传输请求的处理以及文件的过滤和路由。

isMatchProxyTransfer()该方法用于检测是否与代理服务器匹配。

cacheFileTransfer(String key, FileTransfer fileTransfer)该方法用于将接收到的文件信息存入缓存中，从而提高发送效率。

addFileTransferInterceptor(FileTransferInterceptor interceptor) 该方法用于添加文件传输的拦截器，实现对文件的过滤。

removeFileTransferInterceptor(FileTransferInterceptor interceptor) 该方法用于删除文件传输的拦截器，实现对文件的过滤。

4.3 客户端实现

本系统客户端的实现采用了开源的 XMPP 协议包(smack 包)，该包中包含了大量的有关 XMPP 协议的 API，对于客户端的开发非常方便。

4.3.1 客户端界面实现

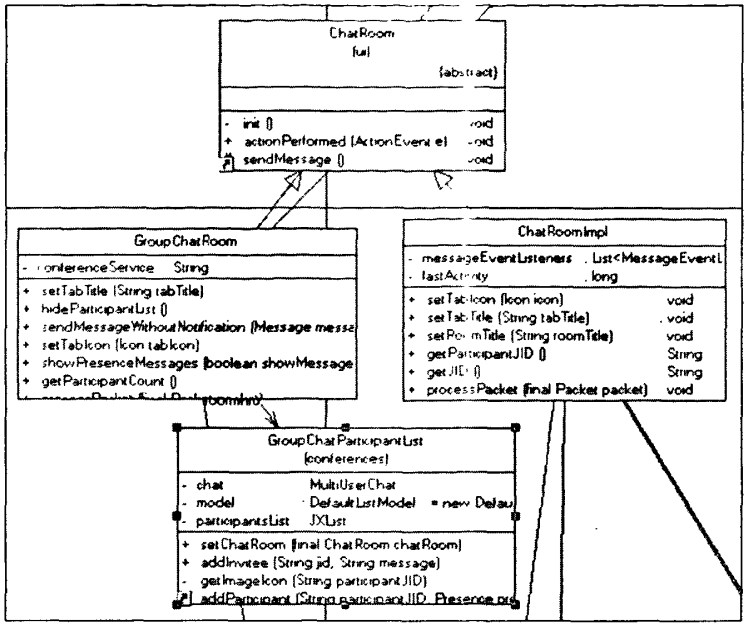

图 4-11 客户端界面类图

图 4-11 是本即时通信系统的聊天室的类图，客户端是采用 java 语言实现，采用 AWT 实现，其中给出了聊天室功能实现的方法。

processPacket(Packet packet)该方法用于处理接收到的 packet 包

setChatRoom(ChatRoom chatRoom)用于设置当前页面上的聊天室信息，本系统会为每一个联系人设一个聊天窗口。

4.3.2 客户端文件传输实现

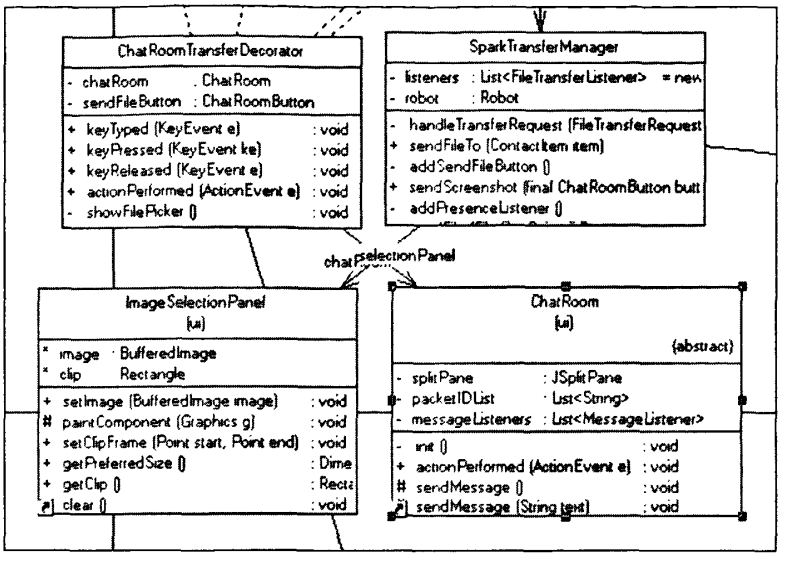

图 4-3 客户端文件传输类图

图 4-12 是客户端文件传输功能的实现类图，因为文件传输并没有专门的窗口而是在聊天室中实现，所以客户端的文件传输功能只需在聊天室中添加必要的出发事件就行了。

sendFileTo(ContactItem item)该方法用于向目标用户传送文件，其中传入参数是目标客户端的标示信息。

4.4 本章小结

本章主要介绍了公共模块、用户注册登录和文件传输功能的实现，由于本系统采用了开源的 XMPP 协议，在协议中已经将各项功能的实现的底层通信流程给于详细说明，因此，在实现其功能的工程中要做的工作主要是将协议中规定的通信流程和规范通过面向对象的计算机语言加以实现，同时由于鉴于其他模块功能都是基于这几个模块实现所以不做更详细的介绍。

第5章 系统测试

测试是项目开发过程中的一个重要环节，它是为了发现错误而执行程序的过程，它是软件质量保证的关键因素，因此本系统在实现的过程中进行了用例测试，以保证系统的可靠性及正确性。

5.1 系统测试设计

由于系统是基于 C/S 结构，分为客户端和服务端实现，为了保证系统的可靠性，我们也对系统进行了客户端和服务端测试。对本系统的测试是采用分模块的用例测试，针对每一个功能编写多个测试用例，以测试系统的功能实现以及容错性能，这里不再赘述。

5.2 系统测试实现

按系统所设计的测试框架，主要以系统管理模块中的用户管理子模块为例

5.2.1 系统测试结构设计

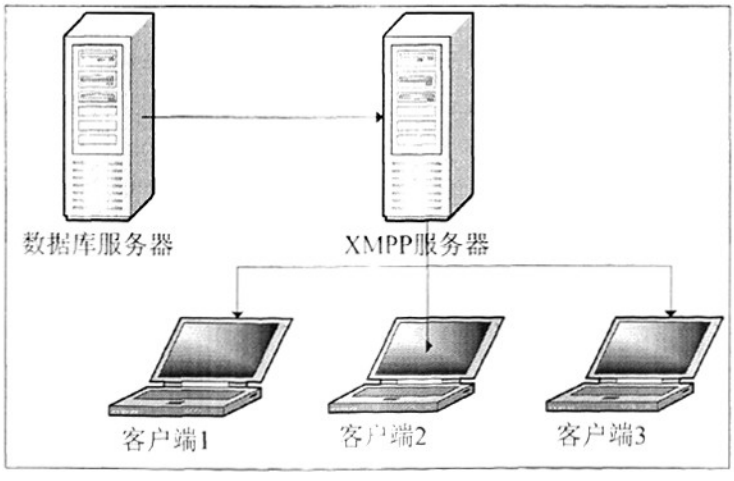

图 5-1 系统测试结构图

该系统的测试结构是基于企业内部办公环境的，因此，相对比较简单，只

需要一台数据库服务器用来存放数据，一台应用程序服务器也就是 XMPP 服务器，外加几个客户端，用来模拟实现企业内部用户。

5.2.2 用户注册功能测试

图 5-2 用户注册界面图

通过点击“注册”按钮进入注册用户页面，如图 5-2 所示。在输入框中输入不符合要求的信息，以测试系统的容错能力，系统提示“输入不合法，请重新输入”；输入相应的符合要求信息之后确认，系统提示“注册成功”；然后用注册成功的账户登录系统，发现能够进入系统，因此，本系统的用户注册功能完全实现，符合系统要求。

5.2.3 添加好友功能测试

首先，用户“chenwu”登录系统，请求将用户“wangjun”加为好友，其效果如图 5-3 添加好友界面图所示。

用户“wangjun”登录系统后，会弹出提示框，如图 5-4 添加好友确认界面图所示。用户“wangjun”单击“Accept”按钮后，系统显示加入好友后的用户列表，如图 5-5 添加好友成功界面图所示。

图 5-3 添加好友界面图

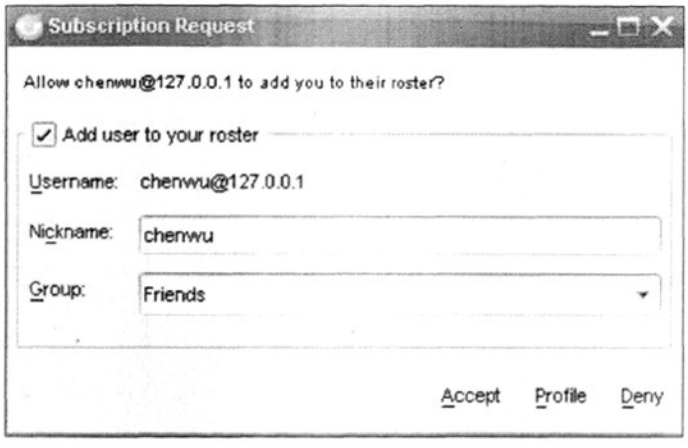

图 5-4 添加好友确认界面图

图 5-5 添加好友成功界面图

通过，以上测试可以看出，系统加入好友功能的实现流程与设计要求一致。

5.2.4 发送消息功能测试

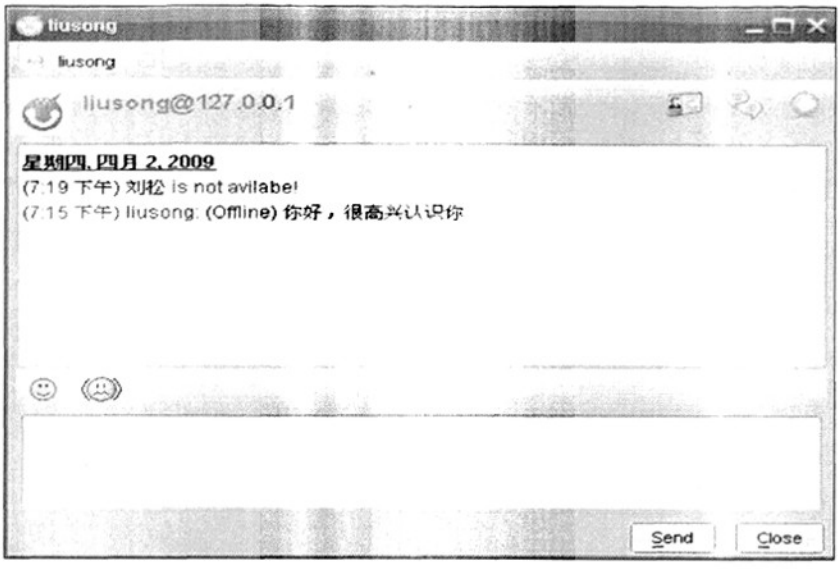

图 5-6 聊天窗口界面图

如图 5-6 所示，用户“chenwu”向用户“liusong”发送消息成功。

5.2.5 群发消息功能测试

图 5-7 群发消息界面图

进入群发消息页面后效果如图 5-7 所示，从中选择要发送的群组和用户，在左侧方框中编辑消息，单击“OK”，发送成功。

5.2.6 文件传输功能测试

图 5-8 文件传输界面图

用户“liusong”登陆后进入聊天室，直接发送文件，其效果如图 5-8 所示。

5.3 本章小结

鉴于本系统的操作性很强，因此我们采用用例测试的方法，经测试各项功能均已实现，并且该项目已经投入使用，均有很强的实用价值。

第6章 总结与展望

6.1 工作总结

本文分析了基于 XMPP 协议的即时通信系统，详细介绍了 XMPP 体系结构、XMPP 协议的相关概念和技术，实现了即时通信系统的功能。系统利用基于 java 的 JSF 技术加强页面的风格统一，并在系统的具体实现过程中采用用例测试确保系统的可靠性。主要工作包括：

1．介绍 XMPP 协议的概念，对比传统即时通讯系统分析了 XMPP 协议的优势，在基于 XMPP 协议的 XML 节传输技术上，结合流行的 mina 开源框架技术提出一种 C/S 模式的即时通信系统模型。

2．分析了即时通信系统需求和功能。在上述分析讨论基础上，实现了一个采用 C/S 体系结构、基于 XMPP 协议的即时通信系统。该系统符合 C/S 结构，采用 XML 语言作为网络传输语言，从而使客户端摆脱了语言的束缚，实现了客户端的多元化。

3．该系统的开发基于模块化的思想，使系统的逻辑结构十分清晰，同时也方便了功能的扩展和二次开发，在实际应用中，该系统达到了良好的可维护性和可扩展性等预期效果。

6.2 工作展望

在我国即时通信软件逐步发展的过程中，开发功能齐全的、反应快速的、基于 XMPP 协议，支持服务器集群的即时通信系统是我们努力的目标。

由于作者水平有限，加上时间关系，对 mina 框架的研究还不是很完善、成熟，有不少实际问题考虑不够周到，还有不少问题有待于解决。总结本文的研究工作还需要进行以下方面的深入：

1．本文作者只是通过简单的即时通信功能展示了 XMPP 协议在系统中的应用，由于即时通信系统的多样性和复杂性，服务器集群、协议网关、VOIP、ERP 集成等业务功能的实现还需进一步的完善。

2．由于 XMPP 协议在不断的改进和完善，相关文献也在不断地更新和升级，

这些都值得我们去进一步研究。同时要采用不断涌现的新方法和新技术，结合新的需求，随时对本方案进行可能的补充、完善性能。

参考文献

[1] 张彦,夏清国.Jabber/XMPP 技术的研究与应用[J],科学技术与工程,2007 年,第 06 期:12~14

[2] 段翰聪,卢显良,宋杰. 面向连接的 P2P 即时通信中继策略研究,计算机科学,2005 年第 6 期:22~25

[3] 吕喆,王树名. 基于 P2P 技术的企业即时通信平台开发研究,计算机与现代化,2007 年第 6 期

[4] 冯亚军,宋子林.基于 XMPP 协议的即时通信系统[J]. 军事通信技术.2005 33(12):57~59.

[5] 田李,彭晓明等.即时消息交换系统服务器设计与实现,计算机工程与设计,第 25 卷第 9 期,2004.09:35~36

[6] Dreamtech 软件研发组. 即时消息传递系统编程源代码解析. 北京:电子工业出版社,2002

[7] Milojicic D S,Kalogeraki V,Lukose R,et al.Peer-to-Peer Computing.Hp Laboratories Palo Alto HPL-2002-57 March 8,2002

[8] 翟朝阳,卢美莲,程时端. XMPP 的协议模型及应用前景[J]. 现代电信科技,2002,(3) : 26~31.

[9] Iain Shigeoka,Manning - Instant Messaging in Java - The Jabber Protocols,Manning Publications Company,2002

[10] 强磊. SIP 协议概述[EB /OL]. http: / /www. chinat2elecom. com. cn /20021227 /00004971. html,2002.12.27.

[11] 苗凯. XMPP 的安全机制分析,通信技术,2003 年 08 期

[12] 刘寿强,温子梅.企业即时通信系统_EIM_安全性初探,计算机安全,2004 年 08 期

[13] P. Saint-Andre,ed.,Extensible Messaging and Presence Protocol(XMPP): Core,IETF proposed standard,RFC 3920,Oct.2004

[14] 刘志治,李晓峰. 基于 SIP 协议的即时信息机制[J]. 北京邮电大学学报,2004,27 (3) : 137 ~142.

[15] Tom Myers,Alexander Nakhimovsky 著(王辉译). java xml 编程指南,电子工业出版社,2007.01

[16] P. Saint-Andre,ed.,Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence,IETF proposed standard,RFC 3921,Oct.2004

[17] P. Saint-Andre,ed.,Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM),IETF proposed standard,RFC

3922,Oct.2004

[18] Tim Bray,Jean Paoli.Extensible Markup Language（XML)1.0,W3C liability,1999

[19] 熊小敏,刘瑛,陈惠清. 基于 java 的网络即时通信系统的设计与实现,计算机与现代化,2005 年第 12 期:37~39

[20] 沈庆国,程时昕.移动计算机网络协议结构,计算机学报,1997.10:

[21] Jongbok Byun. Instant Messaging and Presence Technologies for College Campuses. IEEE Network,May/June 2005

[22] Jim Whitehead,Streaming XML with Jabber/XMPP,Published by the IEEE Computer Society,Sep. 2005

[23] 张云川,标准化的即时通信协议_SIMPLE 和 XMPP 的对比研究,武汉科技大学学报,第 28 卷第 4 期,2005.12

[24] RFC3921 Extensible Messaging And Presence Protocol(XMPP):Instant Messaging and Presenee,http://www.ietf.org/rfe/rfe3921.txt

[25] 黄勇,万琴,黄晓萍. 基于 XMPP 协议的即时消息系统及应用,江西科学,第 23 卷第 6 期,2005.12

[26] RFC3920 Extensible Messaging and Presence Protocol(XMPP)：Core, http://www.ietf.org/rfc/rfc3920.txt

[27] 刘培鹤,牛晓蕾等.企业即时通信系统安全性分析与设计,网络安全技术与应用 ,2007.02

[28] 张卫霞,刘瀛. 即时通信前景展望及中国电信的应对策略,现代通信,产业观察,2006.10

[29] 代印唐,张世永. 即时通信安全研究. 电信科学,2006,第 4 期

[30] Pin Nie,An open standard for instant messaging eXtensible Messaging and Presence Protocol (XMPP),TKK T-110.5190 Seminar on Internetworking,2006.05.04

[31] DJ Adams,Programming Jabber,O'Reilly Media,Inc. 2002

[32] 尹冀波.基于 SIP 的企业即时通信系统的研究,微机处理,2007.06,第 3 期

[33] 刘影,季波.企业级即时通信系统的应用研究,现代商贸工业,第 19 卷第 6 期,2007.06

[34] 刘彬,丛建刚等. 即时通信协议分析与监控技术研究,计算机应用研究,第 24 卷第 9 期,2007.09

[35] MDay,S Aggarwal,Gmohr,J.Vincent. RFC2779. http://www.faqs.org/rfcs/rfc2779.html

[36] MDay,J Rosenberg,H Sugano. RFC2778. http://www.faqs.org/rfcs/rfc2778.html

[37] 周建军. 企业级即时通信技术及其应用,电脑知识与技术,2006 年第 2 期

[38] 余其炯. 即时通信的现状和发展趋势,数字通信世界,2007 年第 6 期

[39] Stevens W R.Unix Network Programming Networking APIs Socket and XTI(Second

Edition)[M/CD].Prentince Hall International,Lnc.1998

[40] Banga G,Druschel P.Measuring the Capacity of a Web Server[Z].USENIX Symposium on Internet Technologies and Systems,1997

[41] 吴兆俊,解海涛,盛步云,罗丹. 一种基于 Agent 的即时通信软件互通解决方案,计算机应用于软件,第 23 卷第 9 期,2006.9

[42] 黄勇. 即时通信在差异和融合中前行,市场研究,2005 年第 11 期

致 谢

在完成这篇论文之际,我思绪万千，衷心地感谢在我读研期间里给予我关心帮助的所有老师、同学和朋友们。

首先我要感谢我的父母和大姐。在我读研期间，他们不仅在生活上也给予我无私的关怀和帮助，同时在我曾经迷茫和彷徨的时候给予我精神上的支持和鼓励，使我找到前进的方向。他们坚强不屈，不轻言放弃的精神，时时鼓励着我，在此谨向父母和大姐致以最崇高的敬意和衷心的感谢！

另外，我还要感谢熊盛武老师和杨世达老师。感谢两位老师在我三年的学习、生活以及论文写作过程中给予了我极大的关怀和帮助，两位老师严谨求实的治学态度，不辞辛劳的工作作风，勇于开拓创新的精神，将令我受益终生，在此向两位老师表示衷心的感谢！

最后，深深地感谢论文评阅和答辩委员会的各位教授和专家，是您们为我的硕士研究生学涯划上了圆满的句号。

附录：攻读硕士学位期间公开发表的论文及参与项目

1. 发表的学术论文

[1] 基于XMPP协议的即时通信系统服务器集群的研究与应用.软件导刊,2009.2

2. 参与研发的项目

[1] 电信增值业务发布平台

[2] 企业级即时通信系统