第一章 项目介绍

1.1 项目背景

JXChat2 项目为本人毕业设计题目，意在开发一个 p2p 的应用程序，能够实现点对点互传消息和文件功能。

目前国内外的即时通讯软件基本上全部基于 C/S 模式，使得服务器端的压力很大造成一定的性能瓶颈。C/S 模式的应用对于无法链接广域网的客户端来说，根本无法实现任何互通互动功能。

当前的 P2P 应用大部分需要一个总的服务器如 MSN，ICQ，QQ 等。虽然这些应用在 P2P 传输时可以建立链接，但其前提条件是能够通过集中式的总服务器得到对方 IP 地址或 NAT 内的映射。一旦客户端无法和总服务器建立链接则客户端将陷入瘫痪。即使在同一 NAT 内的两个端点也无法建立通信。另外大部分防火墙在处于安全考虑的情况下有可能阻截部分 P2P 的传输，限制不安全协议，导致 P2P 的传输速率下降甚至受阻。

所以当今互联网时代急切需要一个能够实现从集中到分布的全功能 P2P 应用软件。

1.2 项目目标

首先研究当前世界 p2p 应用现状，确定具体的 P2P 开发方案，包括开发语言和框架。然后在适合的平台上开发一个支持文件传输和即时消息的应用。项目计划支持如下功能：

1. p2p 应用程序
2. 支持点对点消息互传
3. 支持点对点文件互传
4. 自适应国际化功能，内嵌多种语言
5. 多系统支持

第二章 JXChat2 架构及开发平台简介

JXChat2 项目基于 Java 语言实现，主要核心架构基于 JXTA 技术，开发平台选用 Eclipse with MyEclipse plugin 和 Netbeans。

2.1 JXTA 简介

Sun 公司自从 1995 年向世界推出了 Java 语言以来，每年都会在 Java 领域里推出新的技术，从 JavaCard、J2ME 到 J2EE、JINI、JavaTV，推动了 Java 技术的发展和应用。JXTA 是该公司 2000 年向业界推出的新技术。该技术的目的是为 P2P 的网络应用开发提供一个统一的平台，而且为了鼓励和支持该技术的发展，JXTA 项目采用了开放源码的方式，因此吸引了大量业界人士参与到 JXTA 技术的研究与应用当中，JXTA Community(www.jxta.org)就是人气很旺的一个 Java 技术研究开发的网站。

JXTA 最早起源于 2000 年的夏天，现在大家把 JXTA 看成是 P2P 的平台，JXTA 的目标是要解决几个技术与商业上的难题。第一是解决众多 P2P 系统互不相通的问题。2000 年，是 P2P 突飞猛进的高潮年，但高潮背后却是许多小公司用自己的封闭系统试图在 Internet 上圈一块地。Sun 认为，只有互通才能真正发挥出 P2P 的优势，就好像 IM(Instant Messaging)，能互连的人越多，越有价值。所以 Sun 决定出面发布一个平台，使所有 P2P 系统都能连接起来，只有 Sun 这样位置中立、但在技术上有雄存实力被大家认可的公司才有希望做成这一平台。

JXTA 的另外一个目的就是找寻一套数量最少、概念最简单的系统构成的“积木”。如果成功，这几块积木就会是今后大家构架信息系统的基本模块，从而帮助人们摆脱像 Windows 或 TCP/IP 这样的传统软件带来的包袱。Java、Jini 和 JXTA 像是 J 的三部曲，Java 取自著名咖啡产地名，Jini 是 genie(精灵) 的谐音，而 JXTA 则是 Juxtapose 的缩写。当时 Bill Joy 用 grep 把所有 J 打头的英文词找出来，juxtapose 跃然纸上，很是巧妙。既表现了 P2P 或肩并肩的意义，又说明 JXTA 不局限于 P2P。但 JXTA 与众不同，它是由一系列网络协议构成的，用任何语言都可以实现，并不只限于 Java，只有彻底独立于操作系统、网络传输技术以及程序设计语言，才真正达到了跨平台，而这样的技术，最容易受到业界的认同。

JXTA 是项目创始人、Sun 首席科学家 Bill Joy 二十多年酝酿的结晶，“JXTA 技术是网络编程和计算的平台，用以解决现代分布计算尤其是点对点(P2P)计算中出现的问题。”JXTA 研究项目，将提供使用户更便捷地访问连接在互联

网上的个人电脑资源的新框架，从而进一步拓展互联网的空间。同时 JXTA 也是 Sun 的 ONE 互联网战略的延续，并且将更积极的姿态与 Microsoft 的.net 战略和 Hailstorm 计划一争高低。Joy 指出，JXTA 可能是 Sun One 平台最简单的一部分，而不是打算将它变成像微软的.Net 那样复杂的东西，而且 JXTA 也将是开放源代码的团体；有别于而微软的 Passport 和 Hailstorm 技术都是申请专利的专有技术，Sun 希望通过公开 JXTA 源代码的方式，成为微软最有力的竞争对手，在公开源代码的领域中，Linux 和 Apache 是卓越的成功典范，Sun 也希望 JXTA 能铸造新的成功。

JXTA 技术提供了基础性的机制解决当前分布计算应用中面临的问题，实现新一代统一、安全、互操作以及异构的应用。目前它支持基于 Java 技术的平台和系统。而将来 JATX 技术将不受到内存的限制而支持更多小型移动设备。JXTA 通过 Java 技术和 XML 数据表达的结合，提供了强大的功能使得垂直应用得以交互，并且可以克服目前 P2P 软件中的限制。同时，通过小型、简单、便于开发的构造模块，JXTA 将使开发者从建立各自框架的复杂工作得以解放，可以潜心关注于建设各类新颖、创造性的、分布式计算应用。

2.1.1 JXTA 的设计目标

首先，JXTA 是为了构建 P2P 网络而制订的一组协议，是处理构建 P2P 网络所碰到的问题的解决方法，JXTA 标准协议规范介绍如下：

“JXTA 由六个协议组成，这些协议是专为特定的、分布式的、对等的网络计算而设计的。使用这些协议，Peer 可以互相合作来建立自我组织、自我管理的对等组，而不必关心它们在网络中所处的位置(在网络边缘或者防火墙的后面)，并且也不需要集中的管理机构。”

因此 JXTA 的核心是六个协议，其次，JXTA 是 P2P 应用程序开发的运行平台；目前 JXTA 首先推出了基于 Java 的参考实现，提供了支持六个协议的 Java API，JXTA 还将推出包括 C 语言在内的其他编程语言的 API，JXTA 在设计时有如下几个目标：

1. 操作系统无关
2. 语言无关
3. 为 P2P 应用提供服务和基础

从本质上讲，JXTA 的目标是希望在任何设备，从台式机到 PDA、汽车、洗衣机等设备都可以支持 P2P 编程。这里有几个概念上的目标，它们包括：

1. 使用组来组织 Peer 并且在组内提供服务和应用的环境。
2. 组可以使用认证和验证方式来控制组内的访问权限。
3. 通过网络来发布关于 Peer 和网络资源的信息。
4. 通过系统来发布各种请求。
5. 提供一个基础平台，供 Peer 之间做路由和通信。在防火墙或者其他障碍后面的 Peer 之间的通信也是这个目标中很关键的一部分。
6. 提供一种机制允许 Peer 之间可以彼此监视状态和资源。

除此之外还有一些其他目标，例如加密、支持不同的通信协议、易用性、稳定性和性能等，所有这些目标在设计 JXTA 协议和最初的 Java API 时，都被考虑到，另外，开发人员和 Sun 公司的管理者还考虑了以下目标：

1. 系统应该允许任何设备直接加入到 JXTA 网络中去。
2. 系统应该允许 ISP 对网络上的 Peer 进行集中管理。
3. 系统应该支持数字产品版权的管理，例如购买的软件、音乐 CD、电影等。
4. 封装和抽象一些特定的核心功能，以便产生出商业方面的应用。

从上面列出的目标可以看出两点，首先要让企业觉得使用 JXTA 可以使自己对系统进行控制，原因在于大部分 P2P 系统没有集中式的管理，所以在应用中不受企业的欢迎；其次，对于硬件或者软件提供商来说，JXTA 系统需要能够创造出利润。

根据以上这些目标，JXTA 被设计成企业可以接受的、容易维护的、健壮的，并且能够满足任何 P2P 应用的概念。

2.1.2 JXTA 的层次结构

JXTA 由三层组成，如图 1 所示。第一层是 JXTA 核心层，它包含了服务所需要的核心功能；第二层是服务层，它提供了访问 JXTA 协议的接口；第三层是应用层，它使用服务来访问 JXTA 网络和 JXTA 提供的功能。这样的设计和一个标准的操作系统比较相似，标准的操作系统包括核心操作系统、服务和应用程序。

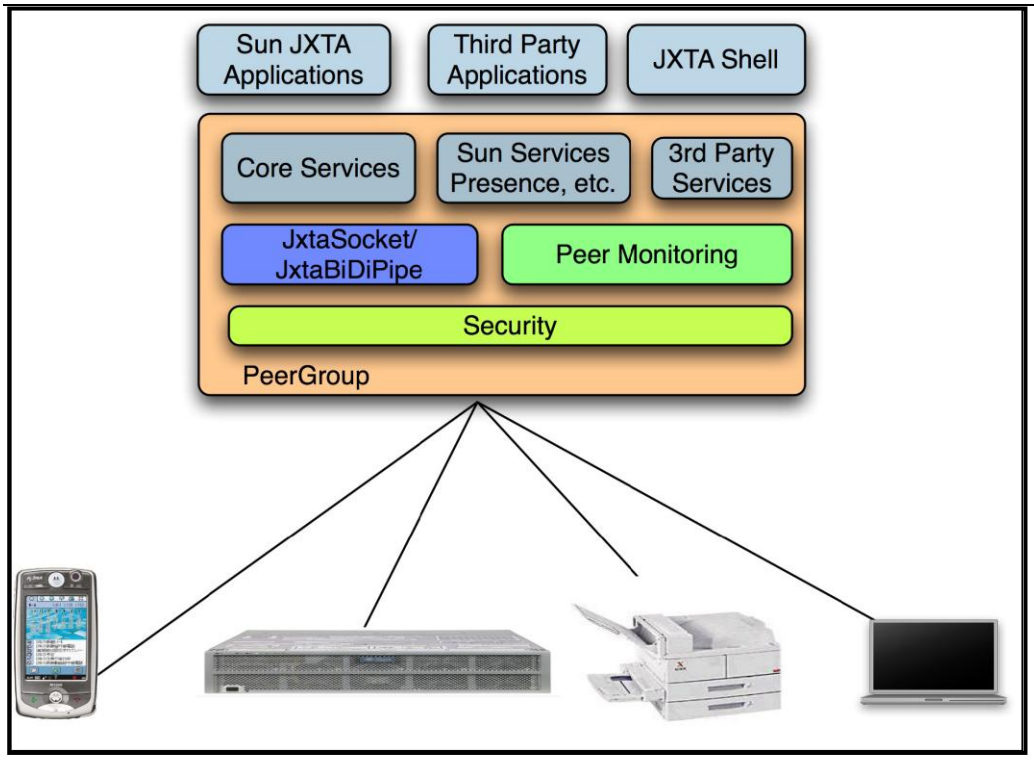

图 1 JXTA 的三层结构

各层的说明如下所示：

1. 核心层(JXTA Core)：这一层封装了最根本的东西，包括 Peer、对等组、Peer 发现、Peer 通信、Peer 监视和相关的安全原语。
2. 服务层(JXTA Services)：这一层包括对于 P2P 网络不是必需的、但很通用的功能，如查找、共享、索引、代码缓存和内容缓存的机制。
3. 应用层(JXTA Application)：这一层包括了应用 JXTA 服务开发出来的完整的 P2P 应用程序，例如 myJXTA，JXTA-CAD 等应用程序。

2.2 Eclipse 简介

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。幸运的是，Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools，JDT）。 Eclipse 的工作截图如图 2 所示

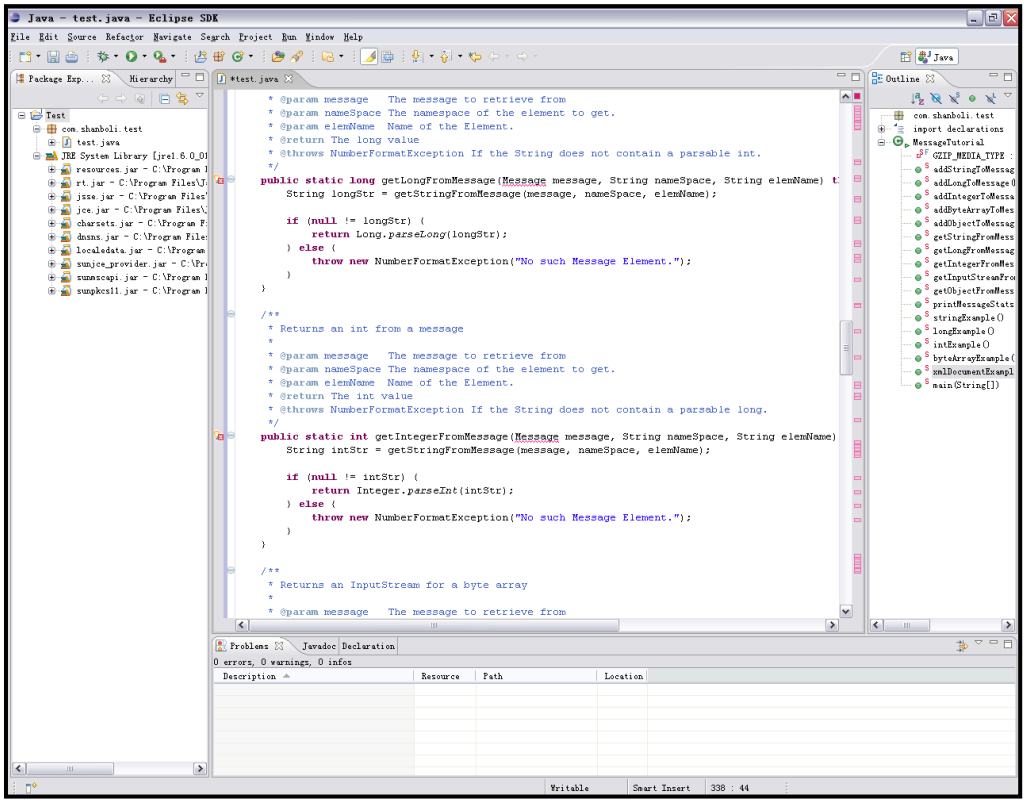

图 2 Eclipse 的工作截图

虽然大多数用户很乐于将 Eclipse 当作 Java IDE 来使用，但 Eclipse 的目标不仅限于此。Eclipse 还包括插件开发环境（Plug-in Development Environment，PDE），这个组件主要针对希望扩展 Eclipse 的软件开发人员，因为它允许他们构建与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的每样东西都是插件，对于给 Eclipse 提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥场所。

这种平等和一致性并不仅限于 Java 开发工具。尽管 Eclipse 是使用 Java 语言开发的，但它的用途并不限于 Java 语言；例如，支持诸如 C/C++、COBOL 和 Eiffel 等编程语言的插件已经可用，或预计会推出。Eclipse 框架还可用来作为与软件开发无关的其他应用程序类型的基础，比如内容管理系统。

基于 Eclipse 的应用程序的突出例子是 IBM 的 WebSphere Studio Workbench，它构成了 IBM Java 开发工具系列的基础。例如，WebSphere Studio Application Developer 添加了对 JSP、servlet、EJB、XML、Web 服务和数据库访问的支持。

Eclipse 最初由 OTI 和 IBM 两家公司的 IDE 产品开发组创建，起始于 1999 年 4 月。IBM 提供了最初的 Eclipse 代码基础，包括 Platform、JDT 和 PDE。

目前由 IBM 牵头，围绕着 Eclipse 项目已经发展成为了一个庞大的 Eclipse 联盟，有 150 多家软件公司参与到 Eclipse 项目中，其中包括 Borland、Rational Software、Red Hat 及 Sybase 等。

Eclipse 是一个开发源码项目，它其实是 Visual Age for Java 的替代品，其界面跟先前的 Visual Age for Java 差不多，但由于其开放源码，任何人都可以免费得到，并可以在此基础上开发各自的插件，因此越来越受人们关注。近期还有包括 Oracle 在内的许多大公司也纷纷加入了该项目，并宣称 Eclipse 将来能成为可进行任何语言开发的 IDE 集大成者，使用者只需下载各种语言的插件即可。

2.3 MyEclipse 简介

MyEclipse 是一个专门为 Eclipse 设计的商业插件和开源插件的完美集合。MyEclipse 为 Eclipse 提供了一个大量私有和开源的 Java 工具的集合，很大程度上解决了各种开源工具的不一致和缺点问题，并大大提高了 Java 和 JSP 应用开发的效率。图 3 所示为 MyEclipse 工作截图。

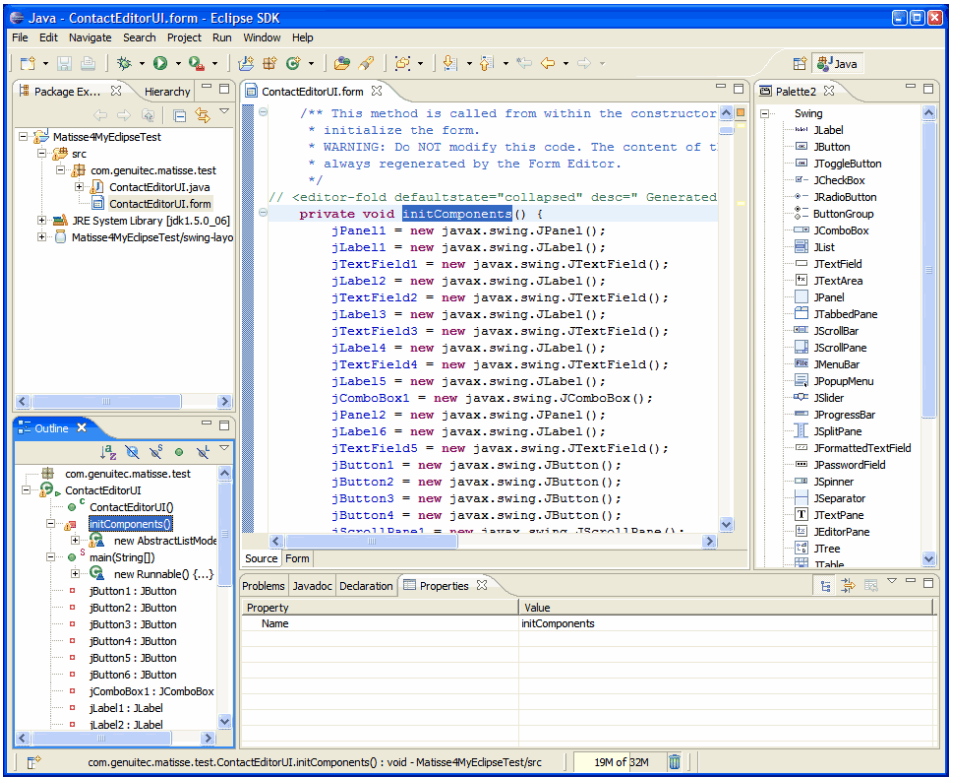

图 3 MyEclipse 工作截图

MyEclipse 的实际价值来自其发布的大量的可视化开发工具和实用组件。

如 CCS/JS/HTML/XML 的编辑器，帮助创建 EJB 和 Struts 项目的向导并产生项目的所有主要的组件如 Action/Session Bean/Form 等，此外还包含编辑 Hibernate 配置文件和执行 SQL 语句的工具。

MyEclipse 的开发者是 Genuitec。MyEclipse 的官方网站为 http://www.myeclipseide.com。目前最新的成熟版本是 MyEclipse5.5 。

2.4 Netbeans 简介

NetBeans.org 由 Sun 公司在 2000 年创立，它是开放源运动以及开发人员和客户社区的家园，旨在构建世界级的 Java IDE。NetBeans.org 全球 IDE 下载次数已超过 160 万，拥有 2 万多个注册成员，并且还在不断发展壮大。其最新研发的 NetBeans 5.5 中文版将帮助中国的开发人员创建更迅速、更稳定和更灵活的开发环境。NetBeans 当前可以在 Solaris、Windows、Linux 和 Macintosh OS X 平台上进行开发，并在 SPL(Sun 公用许可)范围内使用。 NetBeans 5.5 和开放源码网站 http://www.netbeans.org 已经获得业界广泛认可，并支持 NetBeans 扩展 IDE 模块目录中上百个个模块。

图 4 所示为 NetBeans 的工作区截图

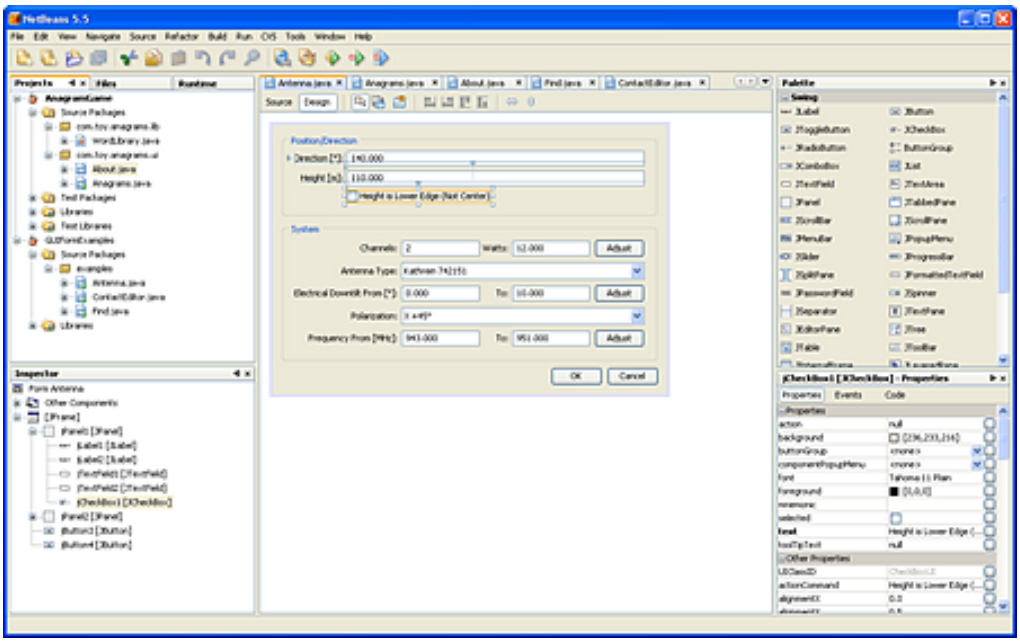

图 4 NetBeans 的工作区截图

NetBeans 是一个全功能的开放源码 Java IDE，可以帮助开发人员编写、编译、调试和部署 Java 应用，并将版本控制和 XML 编辑融入其众多功能之中。NetBeans 已经放出最新的测试版 NetBeans 6， Netbeans6 可支持 Java SE6 平

台标准版应用的创建、采用 JSP 和 Servlet 的 2 层 Web 应用的创建，以及用于 2 层 Web 应用的 API 及软件的核心组的创建。此外，NetBeans 5.5 还预装了一个 Web 服务器，即 Tomcat，从而免除了繁琐的配置和安装过程。所有这些都为 Java 开发人员创造了一个可扩展的开放源多平台的 Java IDE，以支持他们在各自所选择的环境中从事开发工作。

2.5 Matisse4MyEclipse 简介

Matisse 扩展了 NetBeans IDE 4.1 Form Editor 的功能，提供简单而直接的 GUI 布局功能，不再需要用户理解复杂的 Swing 布局管理器。当您往窗体中拖放组件时，IDE 自动建议组件的对齐方式、间距和大小调整约束。组件的位置是由程序所运行的平台的外观来确定的，这与 GridBagLayout 不同。无论调整组件大小、改变组件位置或者在不同平台上运行，Matisse 制作的 GUI 看起来都会很棒。

Matisse4Myeclipse 是 matisse 在 MyEclipse 的完整实现，以满足客户日益增长的需求。图 5 和图 6 所示为 Matisse4MyEclipse 的截图。

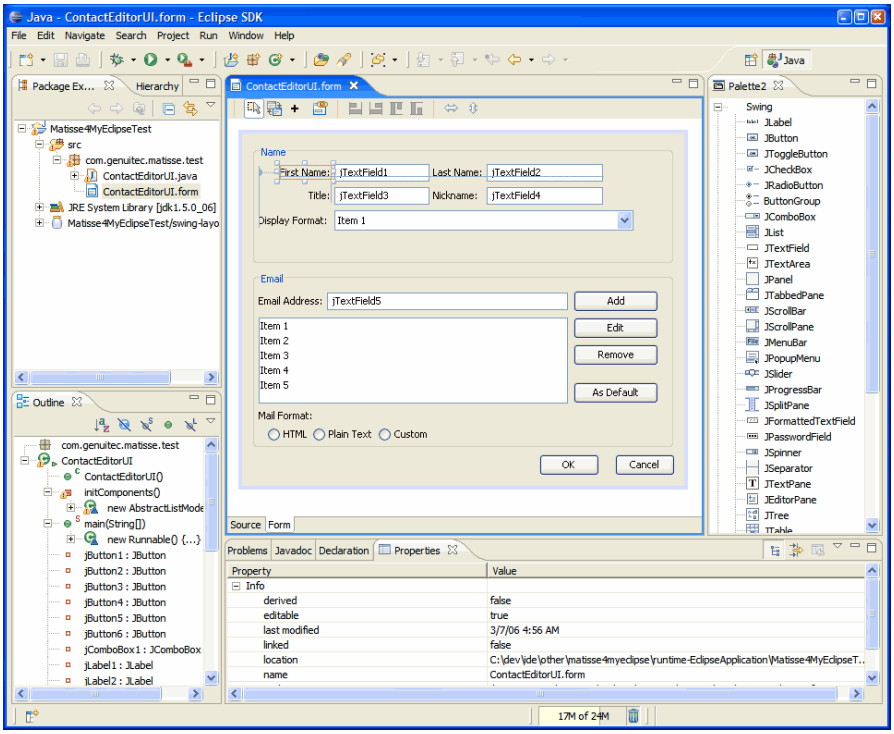

图 5 Matisse4MyEclipse 的截图

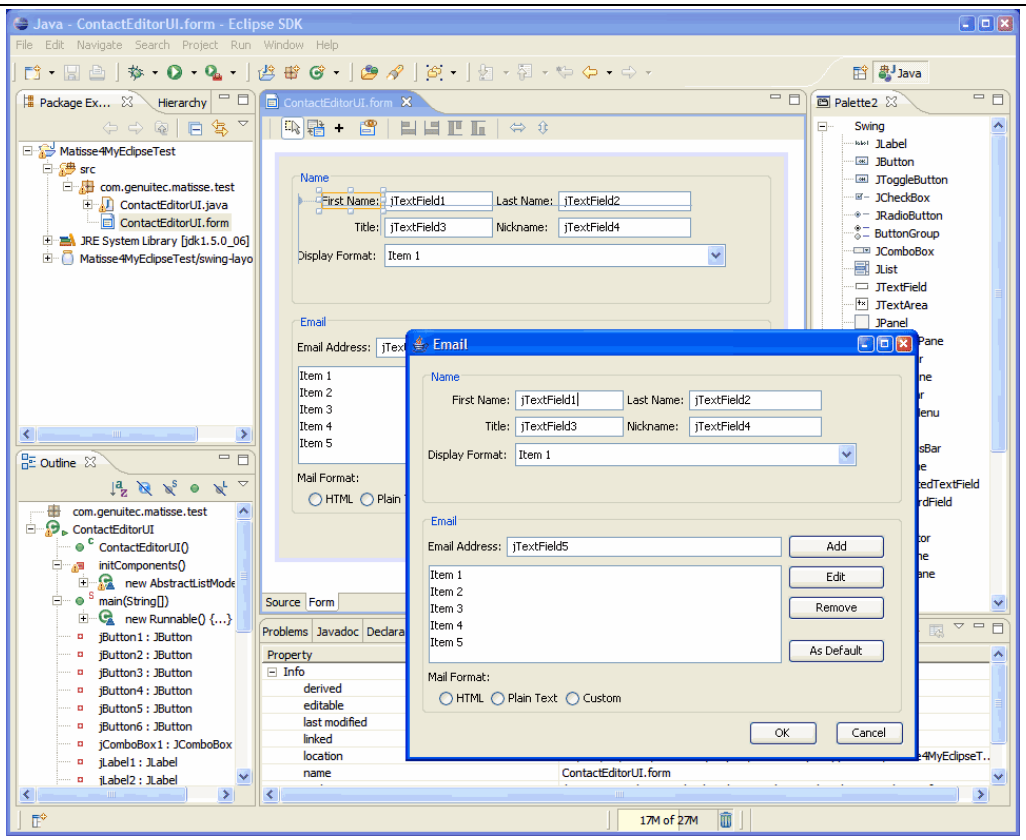

图 6 Matisse4MyEclipse 的截图

第三章 JXChat2 架构

JXChat2 和当前市场上的产品相比，有着显著的优势。本章将从当前市场上的产品的核心问题和 JXChat2 的优势两方面来论述。

3.1 当前 IM 和 P2P 应用的核心问题

下面几节将简要介绍 P2P 可能存在的问题。这些问题是一般性的，可能在许多情况下发生，JXChat2 试图使用核心 JXTA 协议克服许多问题，但是这些问题仍然有可能发生，这要取决于应用程序和网络拓扑结构。

3.1.1 集中式服务器

当前大部分的 IM 的运行方式都是集中式服务器架构。尽管计算机的成本有所降低，但是 Internet 服务器仍然很昂贵。原因在于：随着 Internet 的发展，服务器的能力也需要提高。由于 CPU 的速度每两年只能翻一倍，所以服务器必须使用多个 CPU 或者群集服务器，以便能够服务于成千上万的用户。考虑到所有这些硬件，运行服务器的成本很高，并且要求非常昂贵的软件。由于用户数量众多，为了聚集 Web 服务器和应用程序服务器，技术在过去的几年内发生了很大的飞跃。数据库也进入了太拉（T）字节的范围。

P2P 使用户（点）能够控制其数据的使用和访问。许多情况下，在提供相同类型的服务时，P2P 要比在服务器环境中复制数据有效得多。P2P 应用程序具有灵活性和容错能力。它们可以在需要时复制数据并把数据广播到多台计算机。而在服务器系统中，存在许多故障点，许多安装都因为追求成本和可靠性之间的平衡而存在多个故障点。

现在到了推行 P2P 应用程序的时候了，这些应用程序利用了可用计算资源和用户桌面上的数据。桌面当前位于网络的边缘，但也是网络应该开始的地方。桌面是未开发的负荷(untapped workhorse)，是用户需要与应用程序交互的地方(而不是使用最初仅仅为了共享科学论文开发的浏览器技术的遍布全球的中继)。现在应该在如何对待和使用资源方面进行一场革命。为了让大家初步了解服务器技术的区别，我们将比较 AOL 和 Morpheus(Morpheus 是一种使用 FastTrac P2P 协议的音乐共享应用程序)。尽管 AOL 有 3000 多万用户，但是由于硬件的限制，只有 200 万用户可以同时在线。相比之下，Morpheus 可以在不需要新增任何硬件的情况下支撑 180 多万用户。虽然这可能是苹果与桔子之

间的不平等比较，但是可以考虑一下特定于 AOL 的应用程序。AOL 必须支持客户端与服务应用程序的连通性，为此需要几百万美元的设备；相反，Morpheus 使用用户的计算机和用户的带宽。Morpheus 的 Web 页面的惟一开销就是运转成本。

3.1.2 均衡带宽

许多 P2P 应用程序由于同时提供应用程序的客户端和服务器部分，所以利用的进入和外出带宽一样多。另一方面，传统的 Web 应用程序使用非常少的客户端外出带宽和更多的进入带宽。例如，Web 页面请求的数据量很少，而页面本身的数据量很多。P2P 应用程序更为均衡，出为如果它们提供 Web 页面，那么它们既要读取来自其他点的页面，又要向其他点提供页面。大多数 ISP 都不是以这种方式提供服务。例如，有些电缆调制解调器和 DSLISP 提供 1．5Mbps 的进入带宽和 128Kbps 的外出带宽。如果进入速度等于外出速度，那么性能会更好一些。即使有更多的宽带 ISP 提供均衡访问，ISP 的基础结构可能仍然仅支持面向进入的均衡负载，而不是面向外出流量的均衡负载。

这个问题没有简单的解决方案。然而，如果 ISP 和公司直接支持 P2P，那么 P2P 可以给它们带来很大好处。例如，通过提供类似于网络路由器的专用 P2P 中继和集合点，那么系统可以应付更大的流量，甚至能够在 ISP 或公司的网络内促进点的合作。

这暗示着硬件和软件生产商有机会生产专门设计的 P2P 中继和集合点。这类 P2P 工具在 P2P 中的作用相当于路由器和防火墙在公司环境中的作用。

3.1.3 SOP 与 P2P

基于 Web 服务器和应用程序的网络基础结构、技术和协议得到了充分的理解。ISP 和公司内部都有标准操作程序（standard operating procedures，SOP）。这两种实体期望用户使用 Web 浏览器访问服务。然而，P2P 网络与 Web 服务不完全相同。例如，P2P 不再使用端口 80 与其他点通信。这与正常 Web 服务中应该采取的做法非常类似，但是端口 80 上的流量可能要比预期的大。一般来讲，网络中通信的类型存在差别。

P2P 设备可能缓解部分问题。如果公司支持 P2P 并且能够正确控制流量，那么不需要通过隧道或其他技术回避端口 80。使用能够由公司控制的标准化中继(例如公司的防火墙和路由器)，它们可以优化流量并过滤内容。

对于那些试图控制用户或阻止使用非公司应用程序的公司，过滤内容可能是 P2P 中继的最主要用途．ISP 也可能需要这么做，以便向有小孩的家庭提供过滤后的内容。

P2P 领域中的许多人可能发现控制和过滤与 P2P 的原则不一致。事实上，Web 最初也是开放和自由的。公司不能失去控制，它们将寻求一些方式来消除自己不希望看到的结果，要增加安全性考虑，不对公司的 Intranet 以及 Intranet 与广阔的 Internet 之间的边界进行控制的 P2P 网络是无法创建的。由于理解公司的行为，对公司友好的 P2P 基础结构将胜过那些对公司不友好的基础结构。

对当今世界进行建模的另一方面就是增值服务。拥有专用 P2P 设备的 ISP 可以向顾客提供与 P2P 应用程序之间的增强连接。这与提供 Web、电子邮件和 Web 托管服务类似。

3.1.4 名称空间

目前，寻址 Web 服务器的工作由 DNS 完成。在 P2P 领域，目前还没有应对的机制。拥有 DNS 名称的计算机也是静态的，而 P2P 点要更为动态一些，它们不够稳定。现在还没有创建满足 P2P 网络需要的统一名称空间。探索仍在进行，人们还没有为黄金阶段的到来做好任何准备。如果没有正确、惟一的方式可用于创建与硬件对象相关的名称，而是由每个开发者选择不同方法来命名点之间的通信，那么部分 JXTA 网络可能存在问题。

Ipv6 这种扩展 IP 寻址协议对各种设备的影响如何？是的，这种协议确实可能进行惟一寻址，但是它几乎不可能导致防火墙、NAT 设备和代理服务器消失。这些设备仍然有助于管理局域网以及提供足够的安全性。在近几年 Ipv6 还无法广泛地使用。由于 ISP 要对惟一 IP 寻址收费，所以用户仍要为拥有指派的 IP 地址花钱，对于 Ipv6 也一样。虚拟寻址仍然仅是一种游戏名称。

3.1.5 知识产权事项

因为内置的控制少了，所以知以产权问题在 P2P 网络中尤为突出。在服务器环境中，对访问数据的用户进行验证和授权是一件简单的事情。数据、验证服务和权限管理都集中在服务器上。P2P 分散了内容发布，看起来是不受控制的。

失去控制的观点是不确切的。实际上，失去控制的惟一原因是大多数开发

者都没有实现权限管理。极大多数情况下，充分相信用户是非常方便的。有几种不同的方式可以用来跟踪文档和它们的用户。虽然这些技术并非都是十分互全的，但是许多技术都非常合算。多数情况下，人们都是诚实的。你需要在易用性和针对因为滥用而造成损失的可能性之间进行权衡。访问权限和版权在公司环境中同样重要。在 P2P 网络中，公司实际上可以进行更多的控制。P2P 系统可以使用与 Internet 应用程序相同的验证、数字签名机制。

3.1.6 用户与滥用者

P2P 不是尽善尽美的。比较难办的事情之一就是控制用户的行为。问题在于对等网络中没有存储有关用户行为信息的集中点。文件共享系统就是一个简单的例子。没有中心授权机构，确保所有用户都行为规范是一件困难的事情。下面是用户可能执行的一些不规范的操作：

1. 不共享任何文件
2. 共享无效文件
3. 共享感染病毒的文件
4. 不在线为其他点提供公平的文件搜索共享
5. 不允许其他点完成文件上载
6. 共享对其他用户没有价值的内容

在服务器模型中，对用户的控制易于实现。有关用户的信息和资源都位于一个地方。例如，对数据的访问可以方便地与用户的权限或参与情况匹配起来。在 P2P 网络中，识别点是否有权访问数据或服务是一件困难的事情。问题在于没有集中的授权机构。这类问题的解决方案通常是使用数字签名。虽然可以使用数字签名验证点或用户，但是在记录其行为方面仍然存在问题。很明显，行为可以记录在每个交互的点上，但是，只有通过协作，组才能明确成员是否违犯了规则。

当然，应用程序可以控制该点上的用户。许多应用程序现在都这么做。关键是确保应用程序涵盖了所有的不良行为类型，并且能够抵抗攻击。

某些情况下，不需要防止不良的点。大多数 P2P 网络都非常大，有问题的点很少破发现。根据 Eytan Adar 和 Bernardo A.Huberman 写的一篇名为《Free Riding on Gnutella》论文，70%的 Gnutella 用户都不向系统提供文件和资源，1%的用户提供网络上可用内容的一半。换句话说，有时我们需要预计到最坏的情况并规划阻止它，或者决定自己是否可以容忍最坏的情况。

3.1.7 用户信任的网络

不仅应该有值得信任的数据和用户，应用程序和网络也应该得到用户的信任。成功的关键还包括避免滥用用户的数据、带宽和 CPU 资源。如果某个应用程序可能导致用户接收到病毒或者会接管他们的计算机，那么用户将转向其他地方。

3.2 JXChat2 的优势

通过上一节的论述，我们了解到，当前的 P2P 还有很多无法令用户满意的地方。JXChat2 用自己优秀的架构解决了上述大部分核心问题。

3.2.1 非集中式服务器依赖

QQ 为了让用户能够随时登录，架设了基于各种协议的共 12 个服务器（目前这个数字仍然在增长），每一个服务器都是价格昂贵的机群。单单为了维护这些机群，腾讯需要每月花费上百万元的各种费用。

微软公司的 msn 也是服务器依赖型应用。去年众所周知的持续了数天的国内用户无法登录 MSN 事件正是由于总服务器疏于管理，长时间的运转和超大的链接数量导致服务器最终瘫痪。这一事故不但给 msn 的客户们造成了巨大损失更是微软的声誉收到影响，失去大量用户。

图 7 和图 8 说明了依赖集中服务器的应用程序工作方式：

图 7 显示为当客户端可以与服务器建立连接时，同一区域内的两个用户通过服务器发现对方并进行互通。

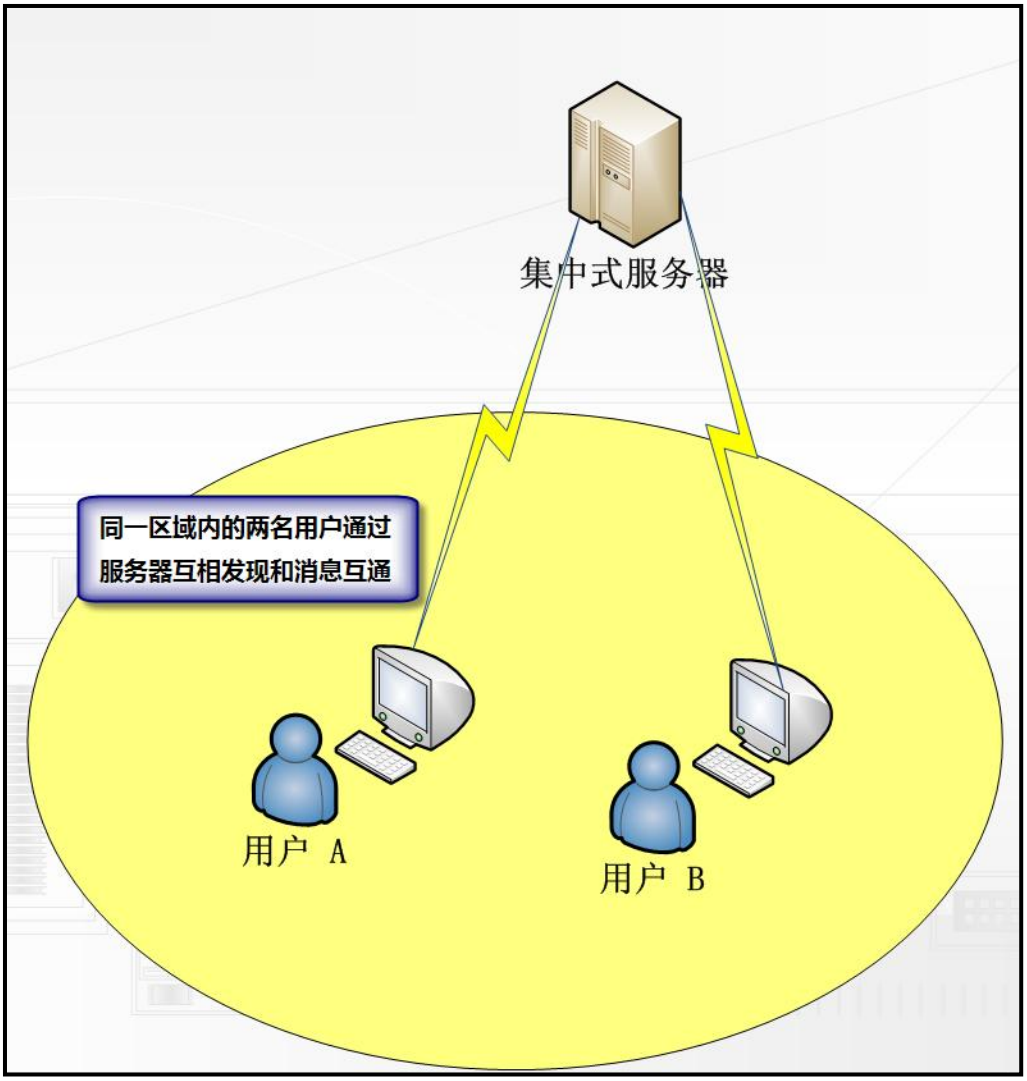

图 7 当客户端可以与服务器建立连接时，
同一区域内的两个用户通过服务器发现对方并进行互通

图 8 显示的是当服务器瘫痪，用户无法和服务器联通时，同一区域内的两

个用户无法通信的效果图。

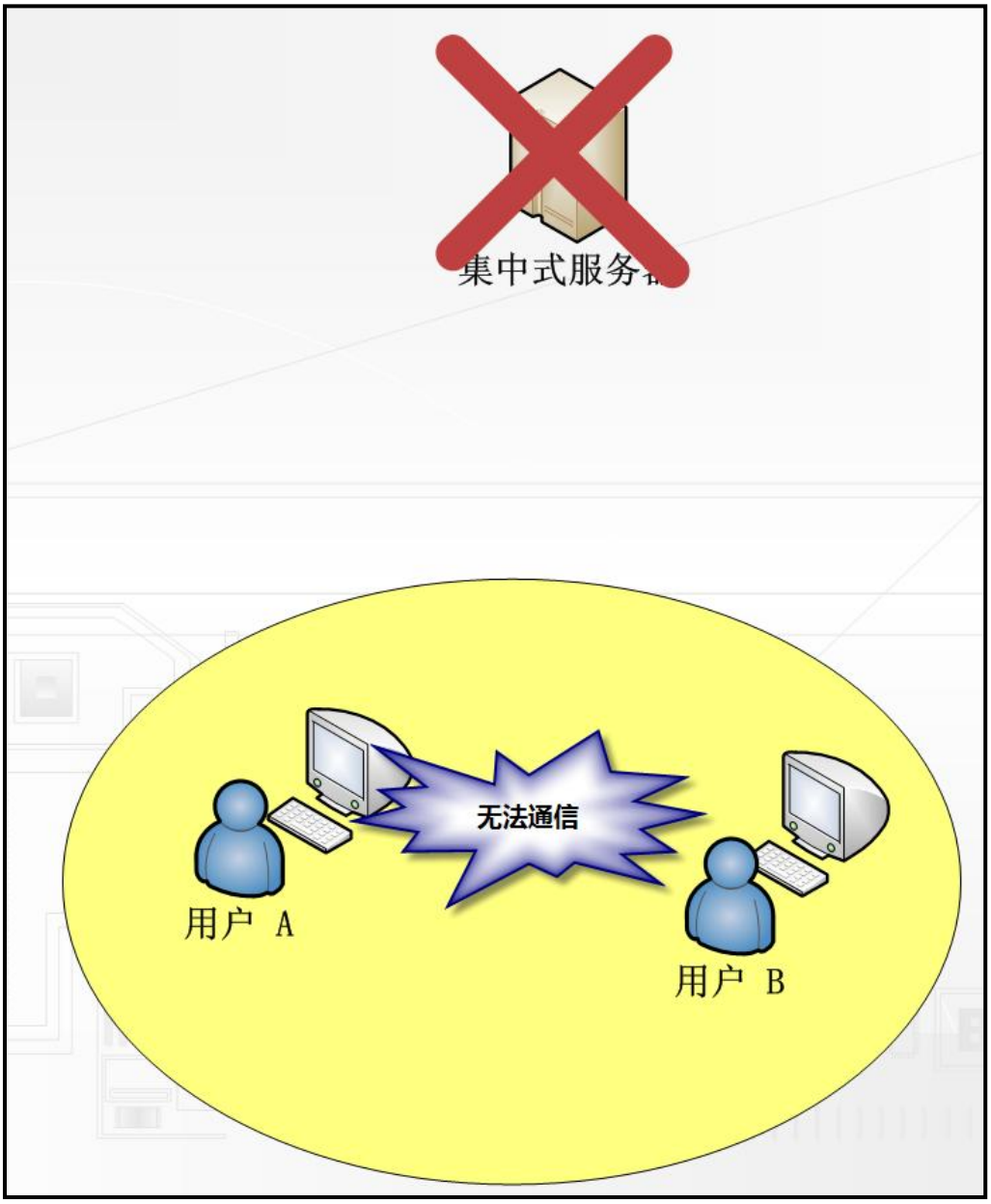

图 8 显示的是当服务器瘫痪，用户无法和服务器联通时，同一区域内的两个用户无法通信的效果图

JXChat2 不需要成本昂贵又难以维护的集中式服务器。

由于 JXChat2 基于优秀的 JXTA 架构，使得 JXChat2 可以有多种发现机制。如动态发现机制，静态发现机制和通过代理查找发现机制。

动态发现机制可以使处于同一 NAT 或 Router 下的一 JXChat2 通过一个多播信息快速找到另一个 JXChat2。而 QQ 和 MSN 却无法做到这一点。如果某公司或学校链接外网的服务器坏掉，那么直接导致所有 QQ 和 MSN 用户掉线，并失去所有消息互通或文件互传的功能。这时就可以看到 JXChat2 的优势了，因为 JXChat2 不需要集中式服务器就可以发现其他客户端。

JXChat2 的工作机制如图 9 所示同一 Router 或 NAT 下的用户可通过局域网内的多播方便地进行查找。

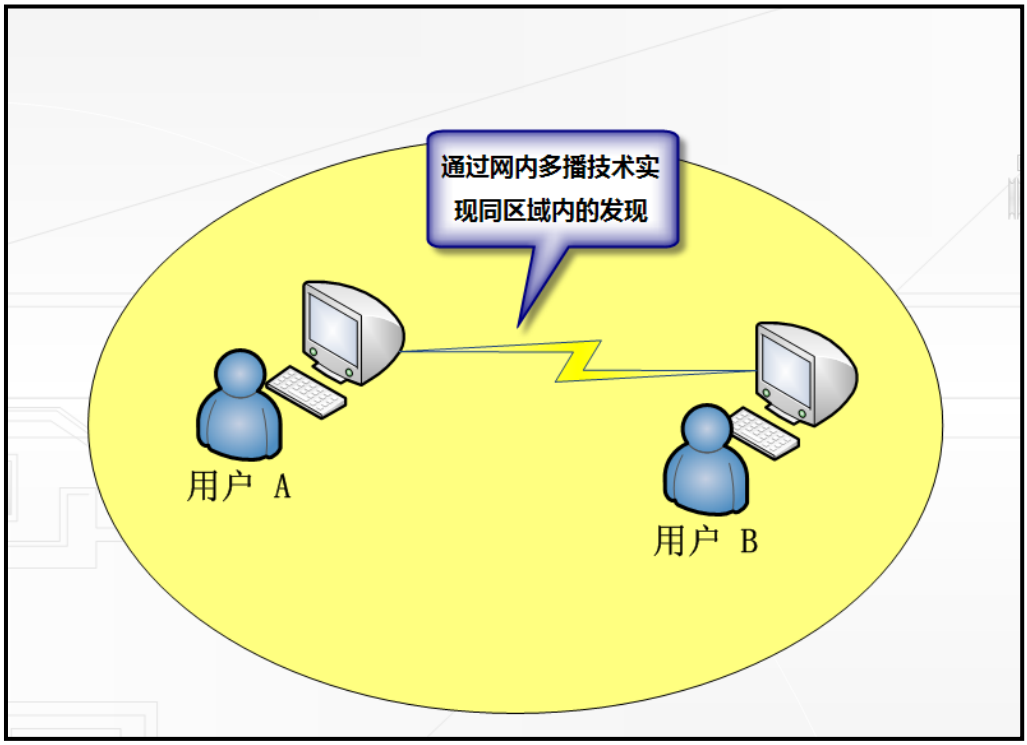

图 9 同一 Router 或 NAT 下的用户可通过局域网内的多播进行查找

3.2.2 跨 Firewall

跨防火墙功能是得到白领级用户重要基础功能。大多数公司为了防止员工在上班事件料 QQ，都通过阻止 tcp 和 udp 的 8000 端口和 8001 端口，并阻止对 QQ 的 12 个服务器的通信来达到阻止员工使用 QQ 的目的。

如图 10 所示，通过集中服务器进行发现的应用一旦被防火墙阻止将无法使用。

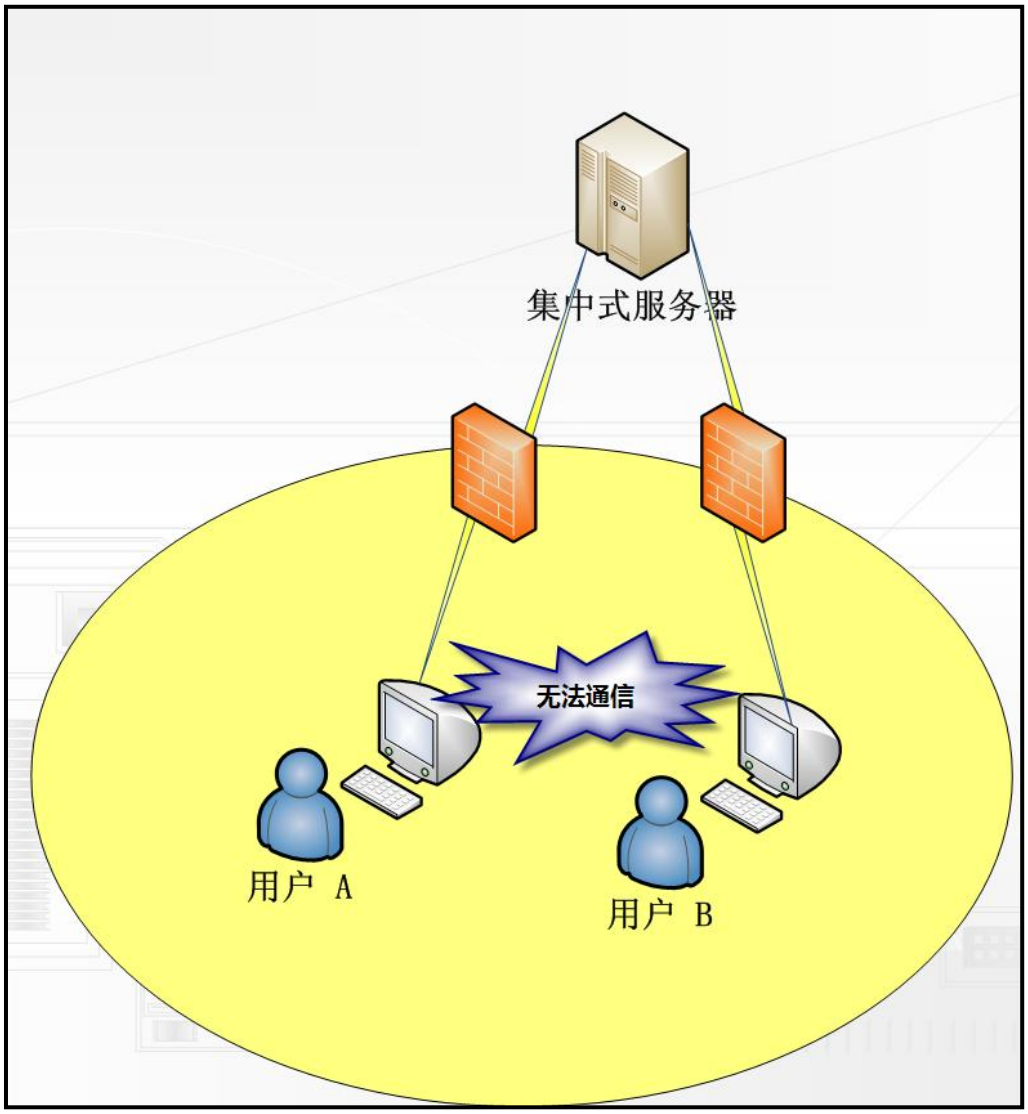

图 10 通过集中服务器进行发现的应用一旦被防火墙阻止将无法使用

JXChat2 有自己的发现机制，可以配置通过代理向一个 JXChat2 应用发送请求。当需要通过代理发送请求时 JXChat2 把寻找对等点的消息打包发送给代理服器的一个 HTTP 请求中；然后代理服务器就像对待其他消息一样转发该请求。当防火墙外的 JXChat2 应用收到请求后它也会将回复信息放在 HTTP 消息中，然后将之发送给代理服务器，再由代理服务器转发给寻找发起端的 JXChat2 应用。整个过程全部通过 HTTP 协议完成，并实现了两个 JXChat2 的发现。HTTP 协议对任何防火墙和过滤机制来说都是无条件允许通过的。

图 11 显示为 JXChat2 的跨 Firewall 工作原理：

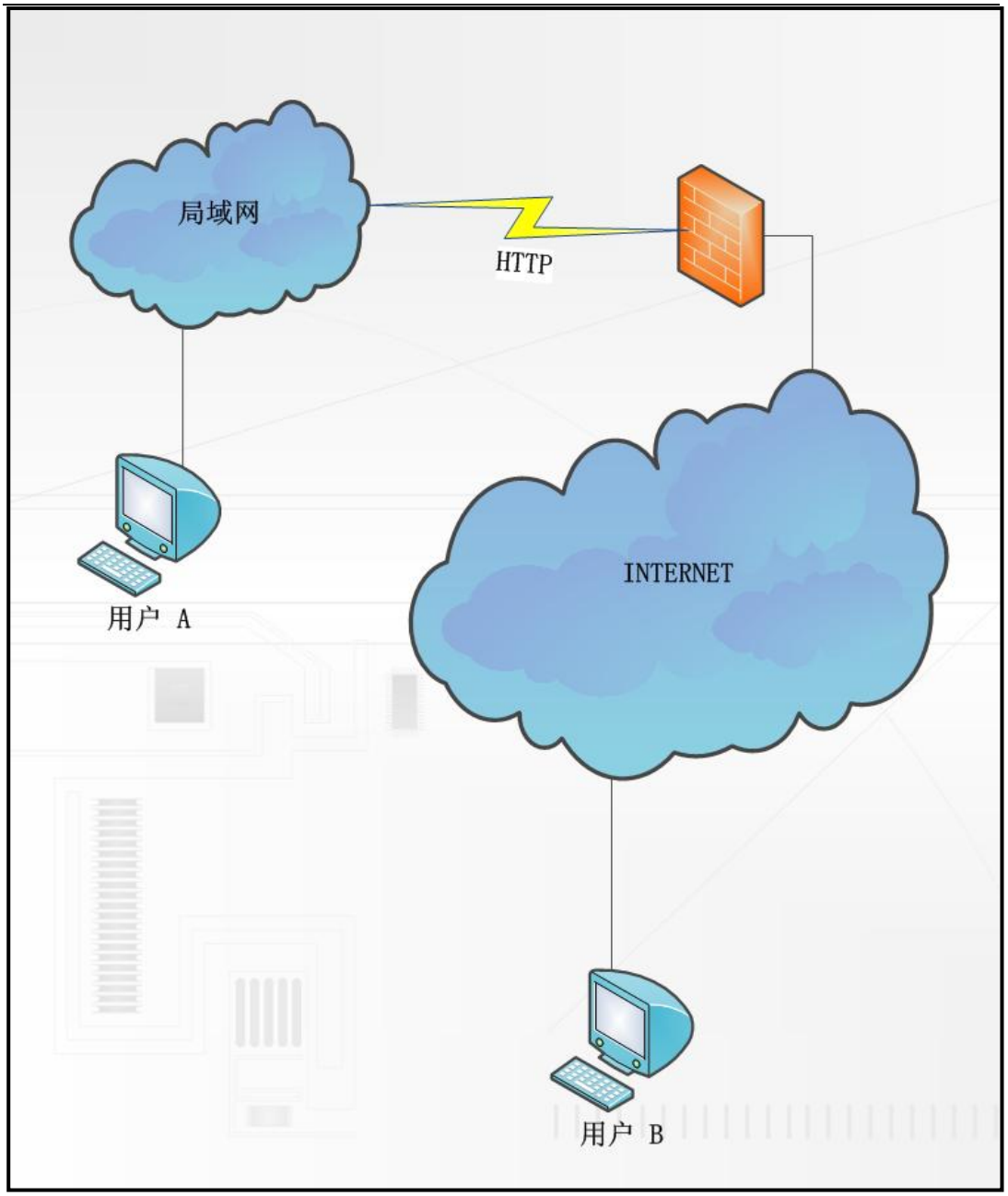

图 11 JXChat2 的跨 Firewall 工作原理

3.2.3 跨 NAT

跨 NAT 问题也是 P2P 应用中涉及到的重要问题。像某些完全基于 IP 互通的 P2P 应用在处理 NAT 内和 NAT 外的机器时就会遇到真实 IP 和 NAT 内 IP 不同的问题。可能一台 NAT 外的机器 ip 为 202.112.144.108，一台 NAT 内的机器 IP 为 192.168.0.9。如果纯从 IP 角度来讲这两台机器的 ip 无法互相链接。

JXChat2 就此问题应用可作为汇聚对等体的 JXChat2 客户端解决了跨 NAT 的应用。图 12 所示为 JXChat2 的跨 NAT 工作原理。

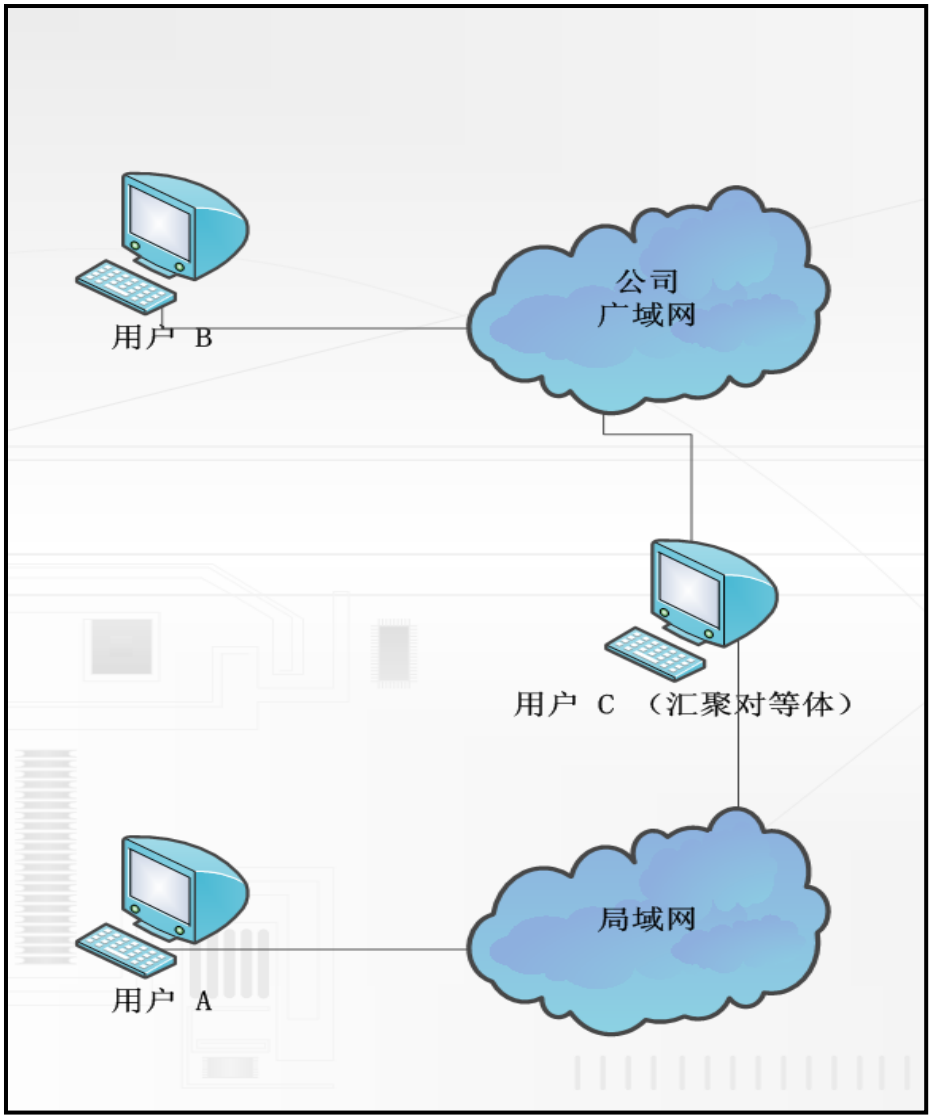

图 12 JXChat2 的跨 NAT 工作原理

作为汇聚对等体的 JXChat2 用户 C，同时处在局域网和公司网络中。用户 A 通过搜索找到了用户 C，用户 B 通过相似的搜索找到用户 C，由于用户 C 可以作为汇聚对等体（在其内部保留一份已知客户端列表）用户 B 和用户 A 就可以发现对方了。

第四章 JXChat2 的实现

JXChat2 目前已经实现的功能有：消息的互传，文件的互传，国际化的支持等

4.1 消息的互传

消息互传即聊天应用被广泛认为是 P2P 程序，这是因为点节点直接与每个其他点节点交互的缘故。聊天应用程序的实现方式多种多样，与它们关联的业务模型也是如此。最流行的例子包括：Tencent QQ，AOL Instant Messenger、Yahoo! Messenger、MSN Messenger 和 ICQ。目前，聊天程序被消费者和专业人员在家庭和办公室里广泛使用，它提供了一种实时的通信媒介，通常比备忘录、电子邮件、蜂窝电话和会议更适合人们的需要。

最流行的商业聊天解决方案使用集中式实现。简而言之，所有交互都要通过一个或多个中心服务器。中心服务器提供连接成员的准确目录并路由全部消息。这种实观不具有很好的伸缩性，需要拥有庞大系统的大型数据中心，以便支持大量的用户。另一方画，集中式解决方案往往在其指定数量用户的操作上和跟踪用户的能力上是确定的。聊天经营者把他们的服务作为一种工具，为其服务争取新的成员，大多数经营者都把这种方式当成一种在其聊天应用程序上销售陈列广告的方式。

基于 JXTA 技术的 JXChat2 非常适合于聊天实现。这在许多 JXTA 的聊天实观中都得到了证明，包括完全集中的、代理的和完全分散的实现。值得注意的发现是，使用 JXTA，集中实现是最困难的（要求在服务器上为所有会话维护状态和消息队列），而完全分散的解决方案则是最简单的，因为 JXTA 平台关注发现以及与其他点节点通信的所有底层事务。

图 13 显示为 JXChat2 的聊天截图：

图 13 JXChat2 的聊天截图

4.2 文件的互传

文件互传或文件共享应用程序仍然是 P2P 技术中最有争议的应用程序。诸如 Napster 以及目前的 Morpheus 和 KaZaA 之类的程序被广泛用来共享媒体文件，它们在某些情况下与现有的版权有冲突。不管大家如何看待当前的版权法，都不能忽视 P2P 在文件共享方面的威力。这种威力也可以直接使企业受益，在企业中仍需对它进行全面的开发。

P2P 的特征之一是信息往往以有机的方式进行分布，这使得控制非常困难。对于受版权保护的内容，这一点使问题进一步恶化，因为内容可以被迅速复制。另一方面，这种特征也有有利的一面，即内容可以更快地被使用，消失的可能性变小。有些工程正是利用这些特征来创建非常可靠的、持久数据存储网络，

例如 OceanStore（UCB）和 LOCKSS(Lots of Copies Keeps Stuff Safe-Stanford)。

就某些消费者文件共享实现（如 Napster）进行的争论可能会使真正的发布受到影响。考虑内容发布（文件共享）中 P2P 的替代方案时．要注意使内容易于被自己和自己的点访问；这里有两种做法。一种是把信息保存在组服务器上，另一种就是创建 Web 站点。把文件保存在服务器的优点是文件位置已知的，它可能由专业人员进行备份和维护。然而，在实践中，这些服务器往往不是按照有利于组中的其他人可靠地使用共享文件的原则来使用的。另外，服务器经常是完全无法或者难以从公司防火墙之外进行访问。因此，用户往往以附件的形式邮寄文件，以便处理引入版本问题的各种事务。

另一种解决方案就是创建自己的 Web 服务器。这是大多数用户的主要障碍。典型情况下，创建 Web 服务器需要获得静态 IP 地址、申请域名、创建 DNS 表项（假设可以访问 DNS 服务器)、安装和维护 Web 服务器以及管理对系统进行的访问。可以比较一下 Web 和 Napster，在 Napster 中，所需要做的就是复制文件，热后发布到共享目录，这样就可以了。有些 Napster 用户常常重命名他们需要发布和共享的非音乐文件。

目前流行的文件共享应用程序在实现上各不相同。两种极端的实现方式是 Gnutella 和 Freenet。这两种方式都是完全分散的。

那些基于 FastTrack 技术的解决方案也是分散的。然而，FastTrack 使用了超级节点或集合点的概念，它们有助于消息路由和内容发现。

Napster 被认为是最集中的实现，因为它依靠集中的服务器对点节点的内容进行索引。在 Napster 中，所有搜索都是对服务器上的集中进行索引完成的，尽管实际文件共享使用直接的 P2P 完成。

JXChat2 的文件互传机制建立在管道的基础上，大大提高了传输速度，并通过缩小单个包的大小来有效减小出现丢包后恢复的时间。

JXChat2 传输文件的截图如图 14 所示

图 14 JXChat2 文件传输截图

4.3 国际化支持

支持国际化功能是 JXChat2 的一大亮点。项目开发之初就考虑到了国际化应用的需求。尽可能的将文字显示都放在界面文件中，项目开发完成后通过 NetBeans 的国际化向导实现国际化的功能。

目前已支持的语言有中文，英文，德文，西班牙文，法文，意大利文，瑞典文，日文等。默认语言是英文。

图 15 显示为在 NetBeans 下进行国际化向导的界面

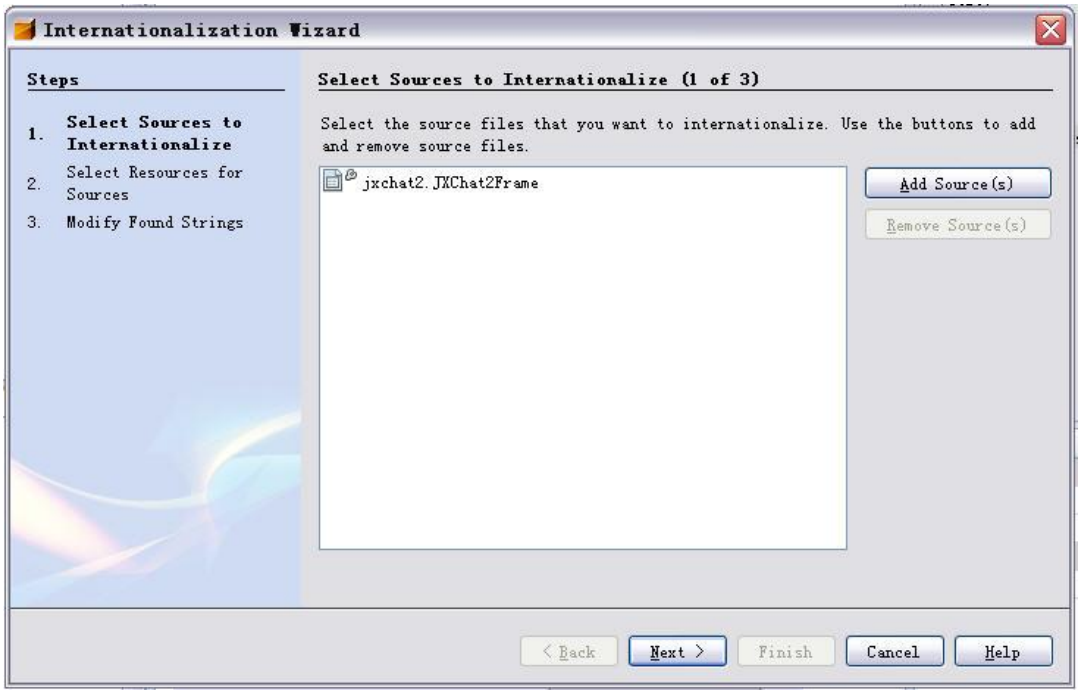

图 15 NetBeans 国际化向导界面

图 16 显示为加入国际化属性文件的工程截图：

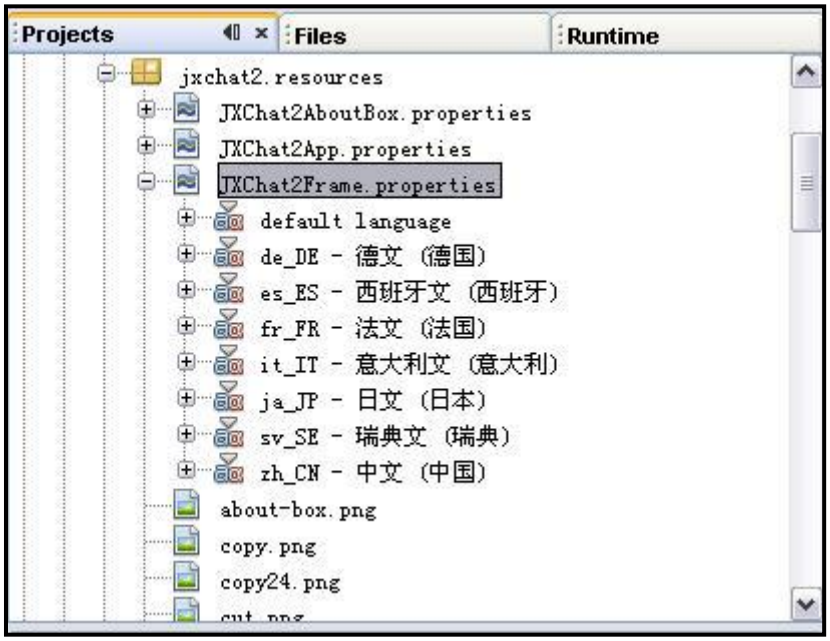

图 16 国际化属性文件的工程截图

图 17 显示为设置语言为日文的运行效果截图：

图 17 JXChat2 的日文的运行效果截图

第五章 主要功能实现方法

5.1 界面结构

JXChat2 的界面十分简洁，图 18 所示为 JXChat2 的截图

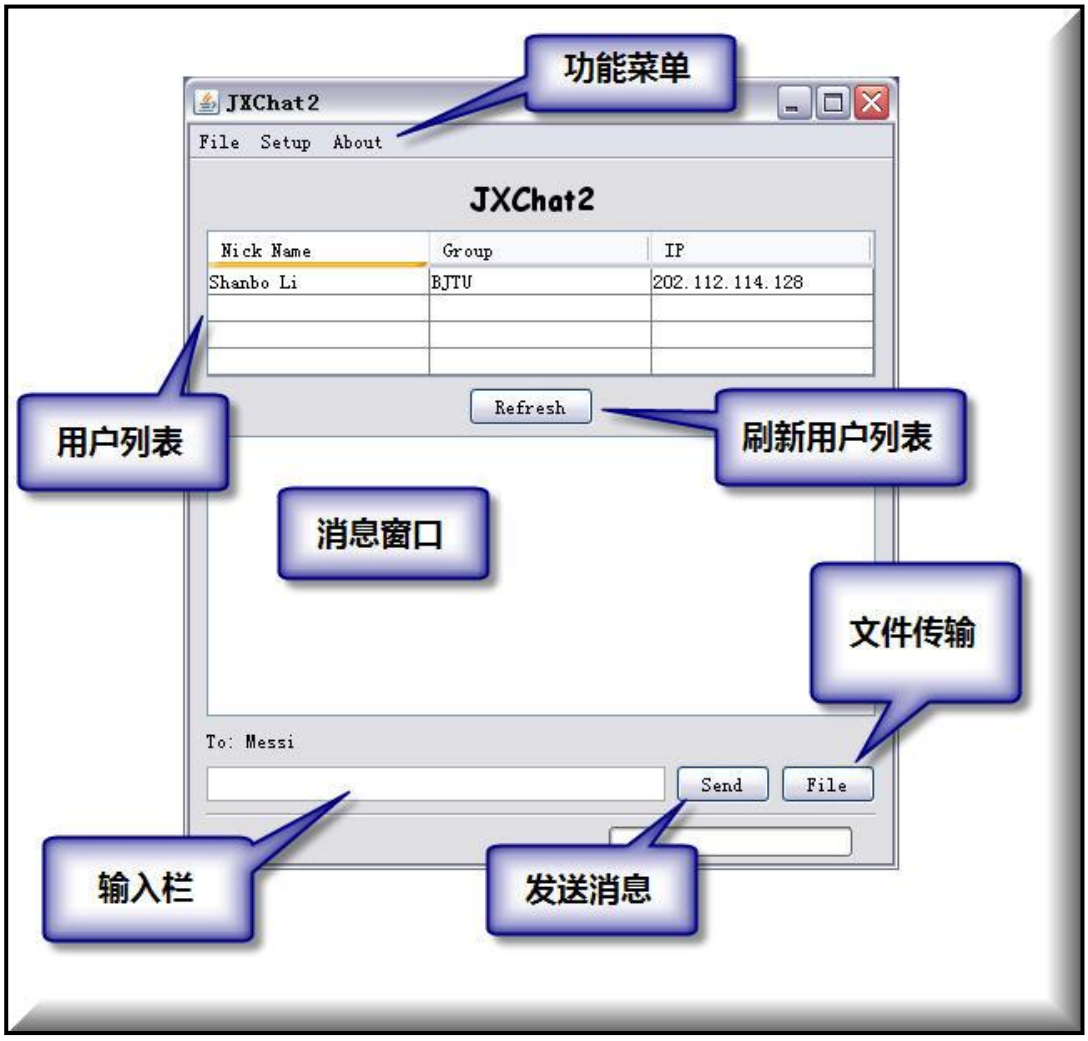

图 18 JXChat2 的截图及功能说明

主要按钮及功能区如图所示分别为功能菜单，用户列表，刷新用户列表按钮，消息窗口，输入栏，发送消息，文件传输按钮等。各按钮和功能区的主要

内容如下：

1. 功能菜单：退出系统，这时网名，版权信息。
2. 用户列表：显示当前在线用户。
3. 刷新用户列表按钮：手动刷新用户信息。
4. 消息窗口：显示聊天信息，文件传输信息。
5. 输入栏：输入发送的消息。
6. 发送按钮：发送消息。
7. 文件传输按钮：传送文件。

全部界面用 Matisse 开发，不存在传统 Java 应用的布局问题。各控件均由 Drag & Drop 方式生成，故不再详细论述。图 19 所示为 NetBeans 的 Matisse 开发环境：

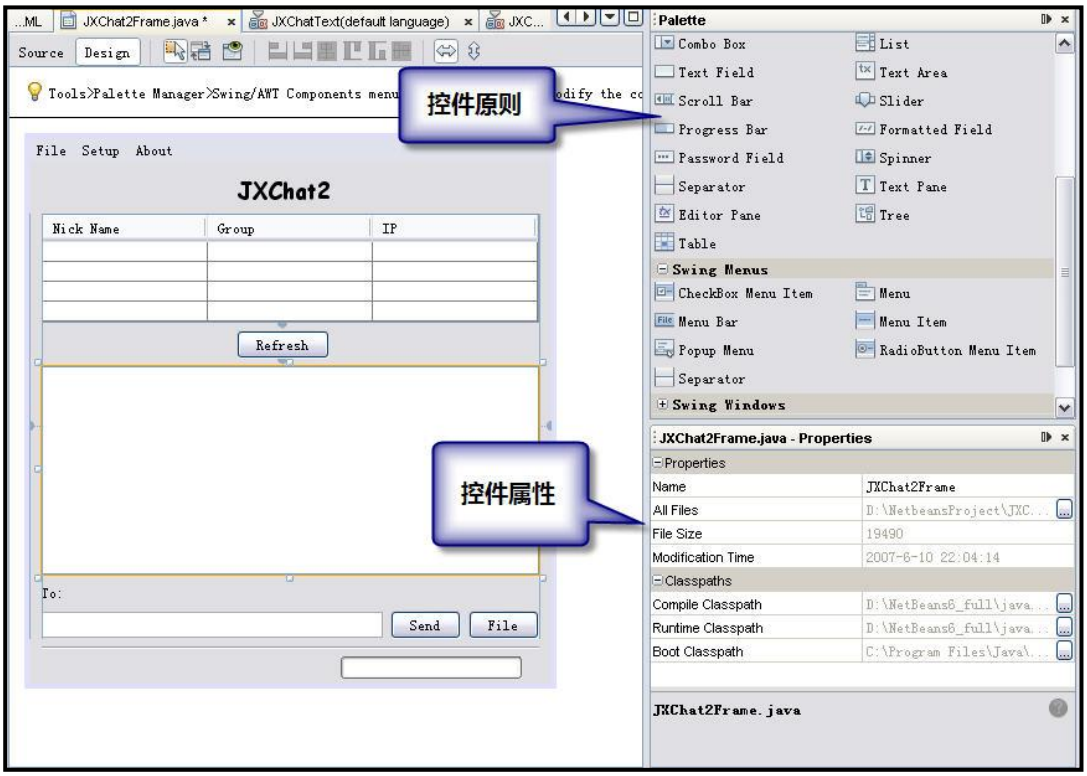

图 19 Matisse 开发环境

5.2 内核部分

JXChat2 项目的文件目录树如图 20 所示：

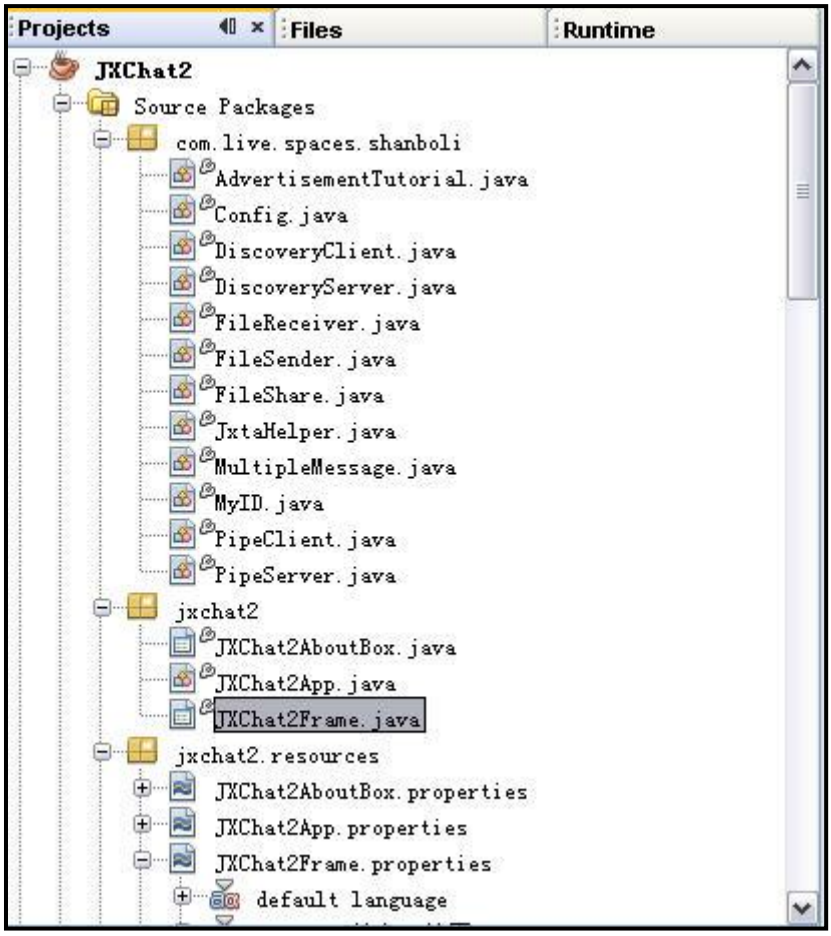

图 20 文件结构树

5.2.1 对等点的发现

对等点即 JXChat2 的客户端的发现是一切活动的前提。发现由 DiscoverClient 和 DiscoverServer 两个类完成。两个类的类结构图分别如图 21 和图 22 所示：

图 21 DiscoveryClient 结构

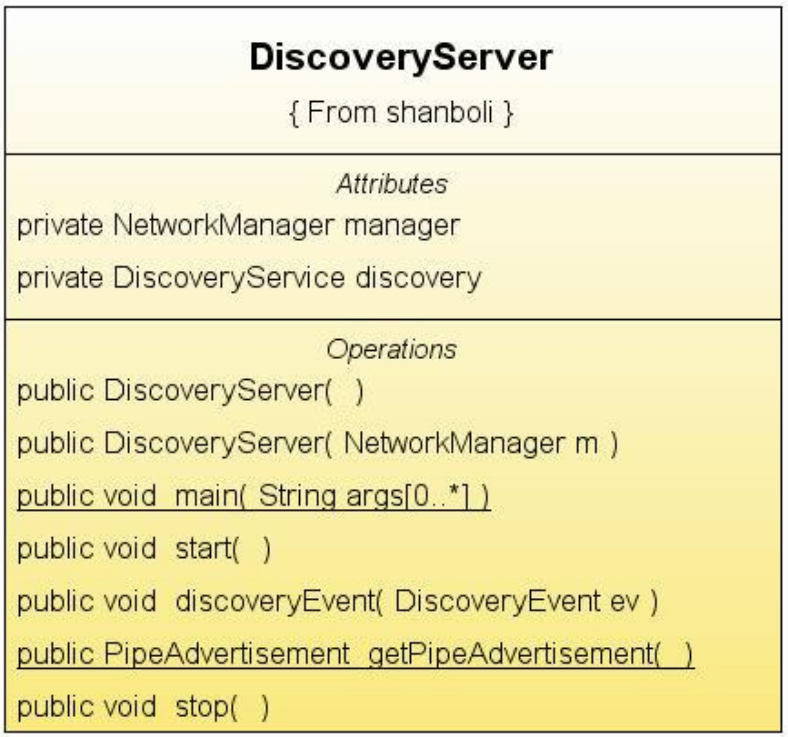

图 22 DiscoveryServer 结构

下面介绍 JXChat2 的发现过程。JXChat2 首先初始化 JXTA 平台，然后发送查找消息到默认的组即 NetPeerGroup 寻找其他 JXChat2 客户端。每当收到一个反馈值，JXChat2 就将新发现的端写入自己的可见用户列表中，以表示发现新用户。

JXTA 的 Discovery Service 类为发现端点，组，管道和服务提供了一套异步的接口。JXChat2 的发现和反馈都实现了 DiscoveryService 的接口。

在这里 DiscoveryClient 类用 DiscoveryListener 接口接收异步的发现消息。其中主要方法的功能如下：

start() 方法:

start()方法首先加入一个接受发现请求（DiscoverRequest）的发现接收器（DiscoverListener）

代码如下：

```
discovery.AddDiscoveryListener(this);
```

无论何时，只要接收器发现了一个发现请求，就会自动调用 discoverEvent()方法。这个方法实现了异步接受发现请求的功能。

在加入接收器后 start()方法通过 getRemoteAdvertisements() 方法始终循环发送寻找请求消息（DiscoverRequest）。

discoveryEvent() 方法：

因为有上面的加入 DiscoveryListener，所以必须实现 discoveryEvent()方法。发现服务会在收到发现请求时自动调用本方法。新的发现的端点会通过 DiscoveryService 自动加入到本地的缓存中(.jxta/cm/group_name)。discoveryEvent()方法会传送一个 DiscoveryEvent 类型的参数。在本方法内的 getResponse()方法可以反回一个事件。在这里，返回 DiscoveryResponseMsg 类的一个实例，代码如下：

```
DiscoveryResponseMsg res = ev.getResponse();
```

每一个 DiscoveryResponseMsg 类都包含一个 peer 的广告。下面代码显示的到广告的过程：

```
PeerAdvertisement peerAdv = res.getPeerAdvertisement();
```

通过 getName()方法就可以得到对方用户的名字了。

```
name = peerAdv.getName();
```

至此，发现过程完成，整个过程如图 23 所示：

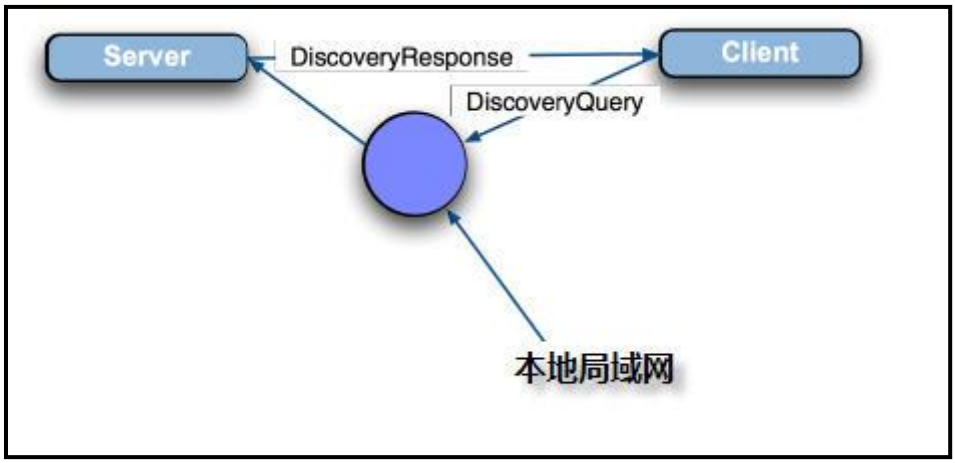

图 23 发现过程截图

5.2.2 消息传输

消息的发送和接收由 PipeServer 类和 PipeClient 类完成。他们的功能如下：

PipeServer：

建立一个有专有 ID 的接收管道（input pipe），监听发送到此管道的消息。

PipeClient：

建立一个有专有 ID 的发送管道（output pipe），并从这个管道发送消息

他们的类图如图 24 和图 25 所示：

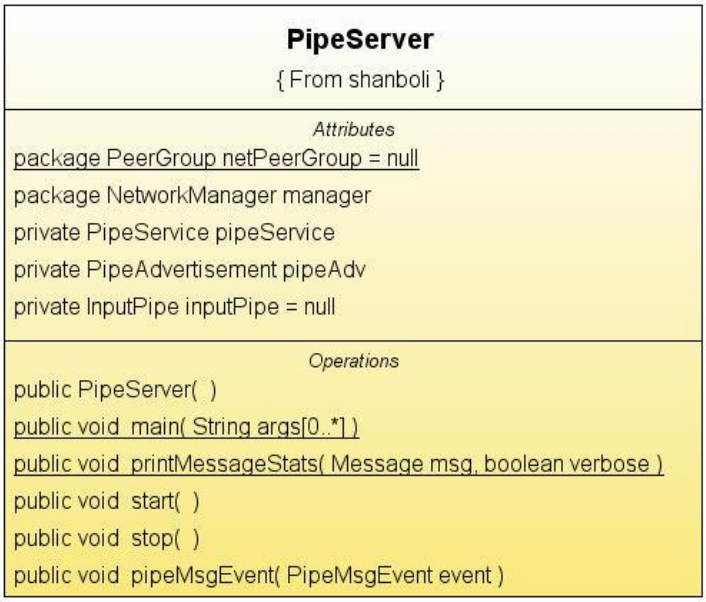

图 24 PipeServer 结构

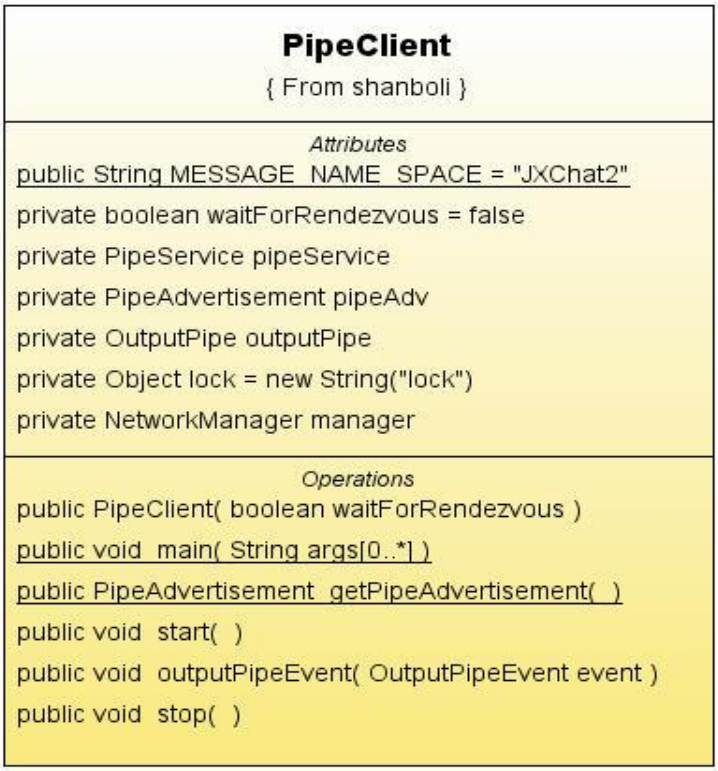

图 25 PipeClient 结构

在介绍消息传输机制之前需要把管道的概念阐述清楚。在 JXTA 架构中管道是一个十分重要的概念。管道为两个管道端点〔即输入管道（接收端）和输出管道（发送端）〕之间提供了一种单向的、虚拟的连接。管道连接的建立独立于管道端点所位于的对等体。例如，输入管道端点可以位于防火墙和 NAT 后，而输出管道的端点则可以位于 internet 上面的一个对等体上。端点甚至可以处于不同的物理网络上。只要两个端点之间存在 JXTA 中继对等体，就可以在他们之间定义一个逻辑管道。

因此，管道可以在不考虑连通性的情况下建立连接。两个对等体可能需要中间的一个路由对等体来实现互相通信。管道把对等体连接虚拟化为性质相同的连接，并且为 JXTA 网络提供了一种完全联通的概念。

图 26 所示为对等体上的管道连接示意图：

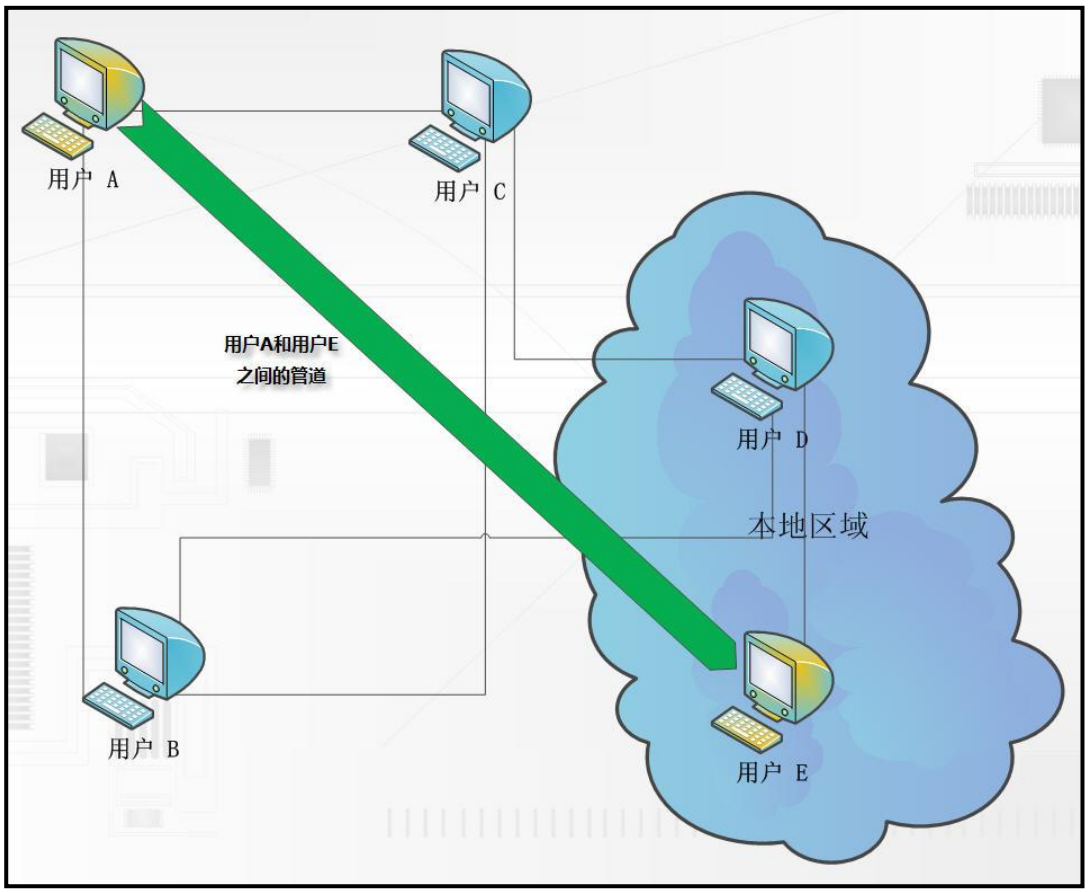

图 26 管道传输示意图

消息传递主要方法功能如下：

5.2.2.1 PipeClient 端

start() 方法：

start 方法首先初始化 JXTA 平台并建立一个默认的组（net peer group），代码如下：

```
netPeerGroup = PeerGroupFactory.newNetPeerGroup();
```

然后从默认的组中得到一个管道服务用于之后要建立的输入管道。

```
pipeSvc = netPeerGroup.getPipeService();
```

之后用特定的 ID 建立一个用于装载信息的管道广告。

```
pipeAdv = (PipeAdvertisement)
    AdvertisementFactory.newAdvertisement
        (MimeMediaType.XML_DEFAULTENCODING, is);
```

通过 getPipeService()可以得到一个用于发送和接收的管道服务。管道服务的 createOutputPipe()方法用于建立一个发送消息的管道。

```
pipeService = manager.getNetPeerGroup().getPipeService();
pipeService.createOutputPipe(pipeAdv, this);
```

outputPipeEvent(OutputPipeEvent event) 方法：

当输出管道建立时 outputPipeEvent(OutputPipeEvent event)方法会被自动调用。下面代码为 **outputPipeEvent 方法的核心代码。**

```
System.out.println("Sending message");
//create the message
msg = new Message();
//add a string message element
StringMessageElement sme =
```

```
new StringMessageElement(MESSAGE_NAME_SPACE,
                                        messageToBeSent, null);
msg.addMessageElement(null, sme);
//send the message
outputPipe.send(msg);
System.out.println("message sent");
```

其中第二句新建一个 Message 类的实例 msg，用于存放信息。后面通过新建一个 StringMessageElement 类的实例 sme 存放用户要发送的消息，并把它加入到 msg 中用于发送。最后的 outputPipe.send(msg)就是通过管道将信息发出。

5.2.2.2 PipeServer 端

PipeServer 端的动作和 PipeClient 端类似，只是在接收到管道消息后要调用 pipeMsgEvent 方法将接收到的消息解码并提取需要的信息。

pipeMsgEvent 的核心代码如下所示

```
public void pipeMsgEvent(PipeMsgEvent event) {
        ……
        Message msg;
        ……
        msg = event.getMessage();
        // get all the message elements
        ……
        MessageElement msgElement = msg.getMessageElement(null,
                                PipeClient.MESSAGE_NAME_SPACE);
        ……
}
```

其中 msg = event.getMessage();用于得到管道消息。另一句中的 getMessageElement 用于得到 msg 中的消息。消息存储于 msgElement 中，可以用 Object.toString()直接得到上一小节中存储于 sme 中的消息。

5.2.3 文件传输

JXChat2 的文件传输借助于 JXTA Socket 实现，考虑到文件的传输要求不能有错误，所以整个发送和接收的过程都加入了 md5 验证机制。控制文件发

送和接收的类分别是 FileSender 和 FileReceiver。

发送方（FileSender）开始时会发送文件名和文件大小给接收方，然后每次从文件中读取定量大小的内容，然后计算 MD5 摘要验证码，接着将部分文件内容发给接收方（FileReceiver），接收方收到文件内容后用 MD5 算法提取接收到的信息，并将 MD5 摘要发送给发送方，并告知希望继续接收消息。发送方接收到接收方的 MD5 信息后会和自己发送的 MD5 进行比对，只有确认发送正确才会继续发送。

5.2.3.1 FileSender 类

FileSender 类是 JXChat2 的一个负责文件发送的类。FileSender 类的结构如图 27 所示：

图 27 FileSender 结构

发送文件内容核心代码如下：

```
if (requestedOffset != -1) {
    byte[] out_buf = new byte[PAYLOADSIZE];
    byte[] in_buf = new byte[16];
    MessageDigest md5 = MessageDigest.getInstance("MD5");
    md5.reset();

    long offset = requestedOffset;
    int bytesRead = readFile(0, offset, out_buf);
    md5.update(out_buf, 0, bytesRead);
    while (bytesRead != -1) {
        dos.writeLong(offset);
        dos.writeInt(bytesRead);
        out.write(out_buf, 0, bytesRead);
        bytesSend += bytesRead;
        out.flush();
        dis.readFully(in_buf);
        requestedOffset = dis.readLong();
        bytesSend += in_buf.length;
        if (Arrays.equals(in_buf, md5.digest())) {
            md5.reset();
            offset = offset + bytesRead;
            updateProgress(offset);
            if (requestedOffset != -1) {
                bytesRead = readFile(offset, requestedOffset, out_buf);

                if (bytesRead != -1) {
                    offset = requestedOffset;
                    md5.update(out_buf, 0, bytesRead);
                }
            }
        }
    }
    dos.writeLong(offset); //should be the overall filesize
    dos.writeInt(0); //end of transfer
}
```

从代码中可以清楚地看到读取文件时是分块进行并验证。特别从bytesRead = readFile(offset, requestedOffset, out_buf);一句可以看出 JXChat2 在文件传输过程中并不是一次性将文件读入内存，而是一次次读取，这样有效防止了在读取超大文件时内存益处可能导致的系统崩溃。

5.2.3.2 FileReceiver 类

FileReceiver 类是 JXChat2 的一个负责文件接收的类。FileReceiver 类的结构如图 28 所示：

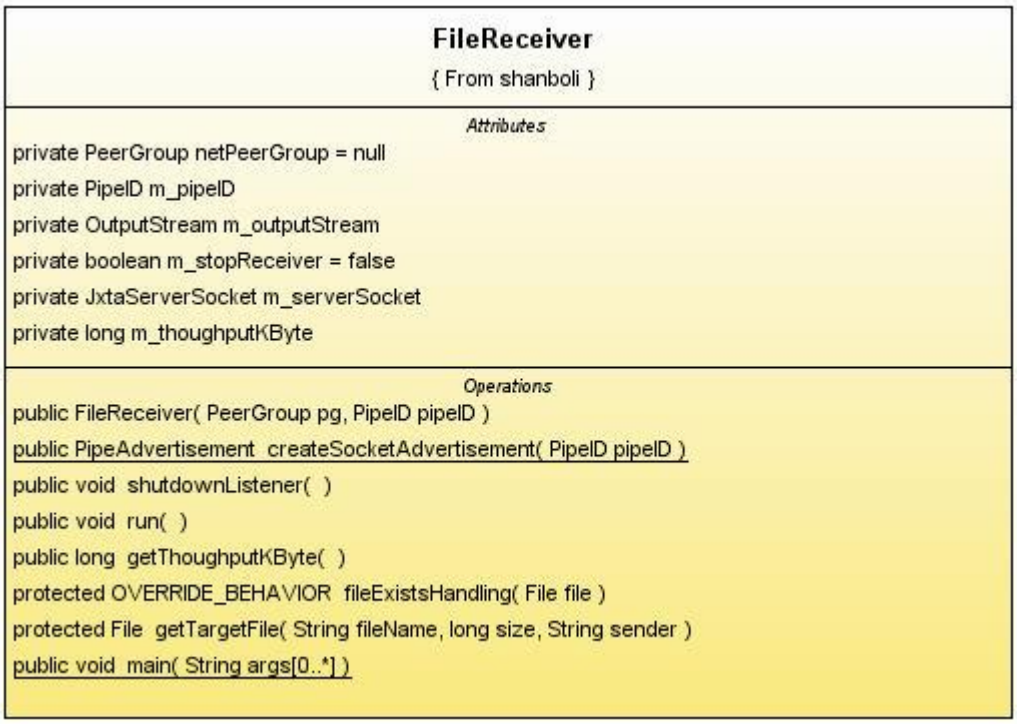

图 28 FileReceiver 结构

FileReceiver 类的接收文件内容的核心代码如下所示：

```
while (size != 0) {
    byte[] buf = new byte[size];
```

```
        md5.reset();
        dis.readFully(buf);
        md5.update(buf);
        byte[] checkSum = md5.digest();
        out.write(checkSum); //发送接收到内容的 MD5 摘要
        wantedOffset = receivedOffset + size;
        out.writeLong(wantedOffset);
        out.flush();
        long nextOffset = dis.readLong();
        int nextSize = dis.readInt();
        if (nextOffset == wantedOffset) { //  是否是期望接收的内容
            m_outputStream.write(buf); //接收到的内容写入文件
            receivedOffset = nextOffset;
            size = nextSize;
            total = total + buf.length;
        }
}
```

从上段代码的生成 MD5 摘要部分可以看到系统采用了验证的机制，防止文件传输过程中出现错误，而最后的 if 条件判断是否是我们期望接收的内容，如果是我们期望接收的内容就写入文件。

第六章 项目前景

JXChat2 有着广阔而又美好的前景。首先 JXChat2 基于下一代 p2p 技术 JXTA，使他有无可比拟的扩展性。而 JXChat2 在项目之初就设计为一个开源的项目，在 Goode Code 拥有自己的项目空间并用 SVN 进行版本控制。

6.1 强大的可扩展性

JXChat2 严格遵守 JXTA 协议，稍加修改即可以方便的和其他基于 JXTA 技术的应用程序进行互通，更可以通过相似原理开发出应用于手机上的 JXChat2。又因为 JXChat2 可以利用 JXTA 的多种形式的消息传播机制，可以根据用户需要开发出提供音频和视频功能的应用。

6.1.1 多平台

由于 JXTA 本身的优越性，可以开发出运行于移动设备上的 JXChat2 并和运行于电脑上的 JXChat2 进行互通。

图 29 展示了 JXTA 的虚拟网络功能

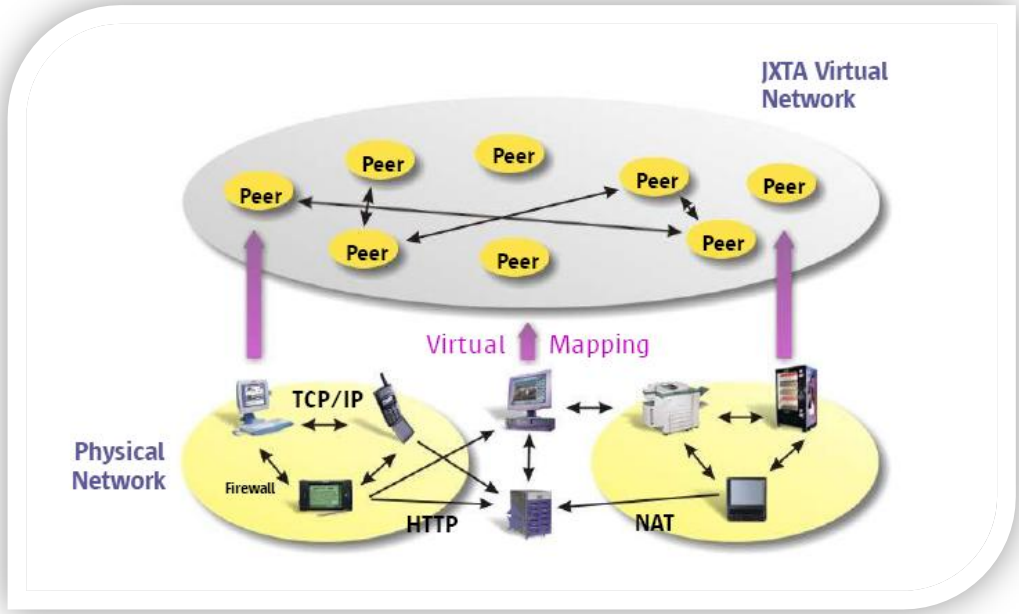

图 29 JXTA 虚拟网络

从上图可见 JXChat2 的扩展空间十分广阔。

6.1.2 多功能

JXChat2 的消息传输机制是把需要用户需要传输的消息以要素（element）的形式加在消息中。这里的要素是 String 类的字符串。从 JXTA 的文档中可以看到，加入 element 的成分除了 String，也可以是整型，长整型，byte 类型，甚至是输入输出流。这就给建立音频和视频传播建立了基础。

6.2 开源和社区

开源是软件的革命。

JXChat2 在开发之初就计划作为开源软件，并在 Google Code 申请了项目空间。项目主页如图 30 所示：

图 30 JXChat2 在 Google Code 的项目主页

任何人可以在任何地方通过项目 svn checkout 项目的最新代码。

项目 svn 主页如下：
http://code.google.com/p/jxchat2/source

匿名用户可以通过如下 svn 连接 checkout 最新代码：
svn checkout http://jxchat2.googlecode.com/svn/trunk/jxchat2

虽然项目最初由我一人开发，但欢迎有兴趣的朋友加入到 JXChat2 中，共同开发。

相信 JXChat2 会有美好的未来。