郑州大学毕业设计（论文）

题目：移动端实时聊天系统的设计与实现

指导教师（校内）：郑志蕴　职称：教授
（校外）：王　磊　职称：高级工程师

学生姓名：任玉琢　学号：20122480227
专　　业：软件工程
院（系）：信息工程学院
完成时间：2016 年 5 月 28 日

2016 年 5 月 28 日

移动端实时聊天系统的设计与实现

摘要：随着移动端智能设备的普及和 GPRS、WIFI 高速无线网络的接入，移动互联网用户呈现出了爆发式的增长。即时通讯是应用程序的的基础功能，可以使不同设备间的用户进行沟通和交流，在中国使用移动终端的庞大用户群体中，使用即时通讯功能的用户越来越多。但是由于各即时通讯软件都过于封闭，很难与其他平台进行整合，应用为了提供即时通讯功能，开发者不得不重复做工作，在缺少即时通讯框架的情况下，各应用的质量良莠不齐，给用户造成众多不便，因此一个好的即时通讯框架的出现就迫在眉睫。

本文首先给出系统的研究背景，分析了即时通讯系统框架的需求。进而对系统进行了设计，提出了系统架构，并根据服务端与客户端所要解决的问题不同，提出了不同的解决方案。对服务端和客户端的实现分别进行了技术选型：服务端需要解决支持大量用户并发访问和主动向客户端推送消息，因此选用 Node.js 和 socket.io 搭建高性能服务端程序，服务端程序支持集群部署，可主动向客户端推送消息；选用 Android 作为开发平台，使用经典的 MVC 架构搭建客户端程序。根据设计，系统实现了好友管理、消息发送和接收等功能。最后，对系统进行了部署并进行功能、性能和安全性测试。

关键字：即时通讯；socket.io；Android；负载均衡；

The Design And Implementation of Real-Time Communication System

Abstract：With the popularity of Mobile Smart Devices and the evolution of high-speed wireless, users of Mobile Internet increase tremendously. IM(Instant Messaging)is the basic function of most of Mobile Internet applications. Without the communication framework, the application is hard to use, so the unified communication framework is urgent.

In this paper, we give the background and the demands analysis of the mobile applications communication framework firstly. Then, we design the client/server system architecture, based on the architecture , we build the server side by Node.js and socket.io and build the client side by Android. From the functions view, The server side can deal with lots of requests and forward messages, the client side is friendly to use. From the design pattern view, the server side is designed by cluster and the client side is designed by the MVC pattern. Finally, the system is deployed and tested, including functional test, performance evaluation and safety test.

Key words: IM; socket.io; Android; load balance;

目录

1 绪论

本章对选题的背景和意义进行了分析，提出了系统需要满足的功能，确定了本文需要做的工作和本文结构。

1.1 选题背景和研究现状

随着智能手机的普及，人们生活、购物、消费和交流等方式都与以前发生了巨大的变化[1]，其中即时通讯是很多应用都不可或缺的组成部分[2]。即时通讯功能提供了服务者和被服务者之间简便快捷的沟通方式[3]；提供了服务者和服务者、被服务者和被服务者之间分享和交流感受心得的简便手段。纵观国内外即时通讯工具，包括微信、QQ、阿里旺旺、Skype[4]、Google Allo 等，都因其过于封闭[5]，无法在自己应用中根据业务进行扩展、精简或更改。这迫使众多应用需要自己实现即时通讯功能，众多开发者不得不重复做工作，但又由于没有统一标准的原因，在众多应用中的即时通讯功能的实现良莠不齐，同时许多应用功能不够完善，使用习惯不一致，通信效率不够高，给用户也造成了众多不便。

1.2 选题意义

为了解决在 1.1 中提出的原因，一款功能完善、安全可靠和易于集成，易于修改的即时通讯开源框架就显得极其必要了。

对于即时通讯系统，需要满足以下功能：

1. 平台方面：服务端支持为多种客户端平台提供透明服务，任何客户端间可以相互通信；客户端支持移动设备主流平台。

2. 安全性方面：服务端和客户端关键数据加密存储，服务端对关键请求进行身份验证。

3. 多媒体数据访问方面：服务端支持多媒体数据分布式存储，对于不同地域用户请求，负载到不同服务节点，对客户端请求有良好响应速度。

4. 并发控制方面：服务端可集群部署，对于大量用户请求负载到不同服务节点，同时需要有良好扩展性。

1.3 本论文的主要工作

本论文通过研究已有即时通讯技术与协议，结合本系统特点，提出以下解决方案：以 Android+Node.js 为开发平台[6]，REST 风格+JSON 为通信协议[7]，WebSocket 为通信基础[8]，并以实际应用为开发背景，进行研究、设计并实现了本系统。根据 1.1 中提

出的问题，提出了以下解决措施：

1. 平台方面：系统使用 C/S 架构模式，服务端使用 node.js+socket.io 搭建，使用标准协议进行通信，服务端发送和接收标准协议数据，对于客户端平台间的差异，交由服务端统一接口，分别进行处理，客户端无需关心；由于开发环境限制，客户端基于 Android 平台实现。

2. 安全性方面：客户端第一次使用用户名和密码进行身份验证，验证过程与服务端通信数据加密传输，服务端验证通过后，服务端返回客户端动态生成的唯一随机 token[9]，客户端以后使用该 token 进行身份确认。同时服务端使用单点登陆技术，同一时间同一账号只可在一台设备登陆，保证用户账号安全。用户注册使用动态短信验证码确认身份，保证安全，防止恶意攻击。

3. 多媒体数据访问方面：系统接入第三方七牛云多媒体数据对象存储平台，支持域名配置，实现图片和语音多媒体数据发送、存贮及接收，使用 CDN 数据访问加速[10]，同时将普通聊天消息数据和多媒体数据在客户端进行本地缓存，减少不必要数据传输流量，加快应用响应速度，提高应用可用性。

4. 并发控制方面：应用服务器分布式部署，使用 Nginx 进行负载均衡[11]，在多服务器间均分服务器负担；使用 Node.js 和 Socket.io 事件机制，提高并发访问处理速度；使用 Redis 内存级缓存，存储用户身份确认信息，并在多服务器间进行数据同步，解决数据一致性问题。

1.4 论文的主要架构

第 2 章：对应用系统进行需求分析，概述系统结构，分析服务端和客户端用例、功能性需求和非功能性需求，介绍系统使用的关键技术。

第 3 章：对应用系统进行详细设计，提出服务端系统需要解决的问题，针对这些问题进行架构设计，对服务端功能模块和数据库进行设计；针对客户端，进行模块设计、界面设计、数据库设计，对数据缓存等其他问题提出解决方案。

第 4 章：对系统服务端及客户端实现功能进行介绍。主要包括对注册登录模块、好友管理模块、消息发送接收模块和历史消息模块的实现进行详细介绍。

第 5 章：对系统进行部署及测试，测试包括功能性测试和非功能性测试两部分。

第 6 章：总结项目所做工作，对项目发展前景进行展望。

2 系统需求分析

本章先整体分析了系统需要满足的功能和特点，又分别从服务端和客户端的用例、功能性需求和非功能性需求三个方面进行进一步的阐述。

2.1 系统概述

本系统作为即时通讯框架，需要满足以下几个特点：

1. 平台无关性：服务端可以同时服务多种平台，对于不同的客户端，包括不同浏览器、不同平台：Android、iOS 等都需要提供统一接口及解决方案，服务端不应依赖于客户端的具体实现，透明提供服务。

2. 可扩展性：系统需要易于修改可扩展，服务端和客户端需要将模块解耦和，可配置，在不同应用场景中尽量减少模块间依赖，对于需要集成系统，则只需要将相关模块集成到项目中而不必全部集成。

3. 实时性：即时通讯对服务端请求处理效率要求很高，则系统应该有一定并发请求处理能力，同时对于过大规模用户量情况下，应易于服务端分布式部署。

4. 安全性：系统应在一定程度保证用户数据安全性，包括身份确认，非法访问拒绝服务，异地多点登陆强制下线等。

基于这些要求，对服务端及和客户端进行需求分析。

2.2 服务端需求分析

从服务端用例，功能性需求和非功能性需求进行需求分析。

2.2.1 服务端用例分析

作为即时通讯系统的服务端，要为客户端的操作提供基本的接口，以满足客户端应用需求，其中主要包括发送验证码、登陆、转发消息、推送消息、个人信息管理、好友管理等。

如图 2-1，对服务端用例图进行了描述：

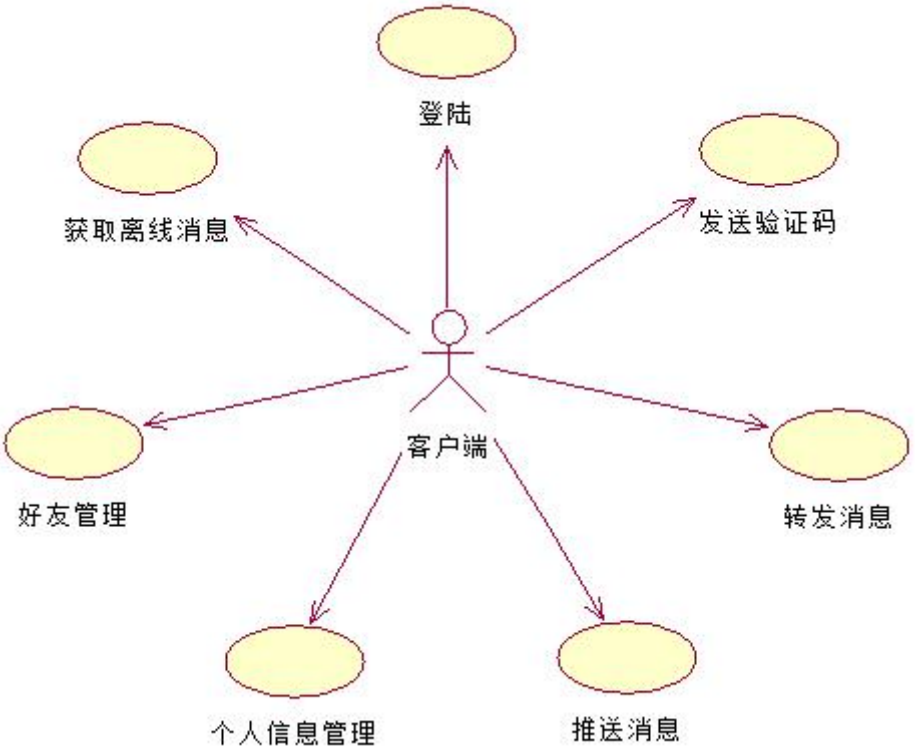

图 2-1 服务端用例图

2.2.2 服务端功能性需求

对服务端系统的功能性需求描述如下：

1. 客户端可以请求服务端发送随机短信验证码，用户注册时需要验证用户身份，防止恶意攻击；

2. 客户端可以通过用户名密码进行登录，服务端经过处理，验证用户身份，如果身份验证通过，给客户端下发身份确认密钥 token，后续关键操作客户端必须在请求中使用 token 进行身份确认；

3. 客户端发送消息给服务端，服务端会将消息转发给接受者，如果接受者为个人，将消息转发给个人，如果接受者为一个群组，将消息转发给所有此群组内用户；

4. 客户端可以接收服务端推送的消息，推送指给全部用户推送消息，在应用更新，服务端维护，节假日营销时可以使用推送功能；

5. 在客户端连接到服务端时，服务端可以向客户端推送客户端退出登录或没有连接网络时，服务端接收到的消息，用于保证客户端不会错过任何消息；

6. 客户端可以对个人信息进行管理，包括修改昵称，修改头像等；

7. 客户端可以对好友进行管理，包括增加、删除好友，修改好友备注等；

2.2.3 服务端非功能性需求

作为一个应用的服务端，为了提高服务端可用性、安全性和可靠性，服务端在满足功能性需求时，还应满足一些非功能性需求，才能更好的为客户端提供服务，主要

包括以下几个方面：

1. 请求并发处理能力：面对大量用户访问，每一用户节点支持每秒 1024 并发即时通讯；

2. 请求相应速度：对于处理的响应速度要够快，在网络延迟可靠下，延迟小于 1s，即对于客户端请求处理时间小于 1s；

3. 客户端平台无关性：对于不同客户端应提供透明服务而不应依赖于客户端系统，支持客户端至少包括市场上主流 Android 系统和 iPhone；

4. 良好的扩展性：在系统功能需要扩展，需要添加新功能时，通过分层的结构设计，将模块间低耦合封装，可以保证程序具有良好扩展性。

2.3 客户端需求分析

从客户端用例，功能性需求和非功能性需求进行需求分析。

2.3.1 客户端用例分析

作为一个即时通讯框架，客户端可能有多平台，这里考虑到开发环境限制，以 Android 为基础平台进行分析。一个可用的即时通讯框架，需要包含基础的登陆，发送消息，好友管理等功能，同时可附加相关基础功能，包括应用反馈、应用分享和群组管理等功能。

如图 2-2，对客户端系统核心用例进行了描述：

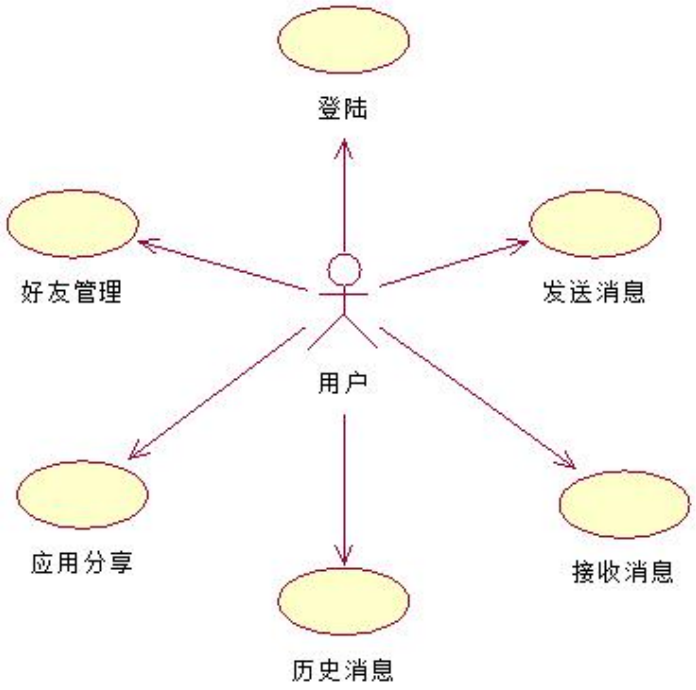

图 2-2 客户端用例图

2.3.2 客户端功能性需求

对客户端系统的功能性需求描述如下：

1. 新用户可以通过应用简便快捷注册，老用户可以通过用户名密码进行登录；

2. 客户端可以在一次登录成功后，下次进入 APP 免密码登录，本地应用加密持久化确认信息；

3. 用户可以对好友进行管理，添加好友和删除好友，添加好友至少应该包含面对面通过扫描二维码添加好友和搜索手机号添加好友；

4. 用户可以给好友发送消息，消息类型包括文本消息、语音消息和图片消息；

5. 用户可以获取历史消息，在用户连接到服务端时，可以主动获取离线消息；

6. 用户可以接收消息，并在联系人处显示未读消息条数。

2.3.3 客户端非功能性需求

作为即时通讯框架，客户端需要满足一定条件，包括以下几方面：

1. 可配置：在集成项目中，需要可以通过很简单的配置就屏蔽掉不需要的功能；同时客户端需要对各功能模块进行解耦，方便扩展和提高已有部分可重用性；

2. 低电量消耗：Android 的性能瓶颈就是电池使用效率不高，需要在保证效率情况下，减少电量消耗；

3. 低流量消耗：在现有流量通讯资费情况下，需要尽可能减少通信数据量以减少资费使用。

2.4 本章小结

本章通过对服务端和客户端用例情况出发，分析了系统的功能性需求和非功能性需求，确定了系统需要实现的功能和要达到的性能指标，同时也确定了系统的范围，在此后的系统设计与实现，都主要以本章需求分析的结论为依据。

3 系统总体方案设计

针对系统服务端和客户端不同的特点，对服务端和客户端采用了不同的设计方案。对于服务端，从服务端需要解决的问题出发，提出了针对服务端服务不同平台、大量用户并发访问、需要保证数据安全和解决数据不一致等问题，分别提出了解决方案，对服务端系统架构和数据库进行了设计，并对服务端功能模块进行了划分；对于客户端，需要解决客户端界面友好和不变信息缓存等问题，本章针对这些问题，对客户端数据库进行设计，功能模块进行划分，设定了数据缓存策略。

3.1 服务端设计

3.1.1 服务端需要解决问题

服务端的可靠、稳定和高效是保证整个系统可用的基础，要达到这样的目的，需要解决以下几个问题：

1. 由于以 Android 及 iPhone 为代表的智能机的普及，一个好的应用的用户量往往呈现爆发式指数级数量的增长，此时一台服务器往往很难为所有用户提供服务，需要多个服务器进行集群部署，将用户请求负载到不同服务器上，但与此同时，也需要解决数据不同步不一致的问题；

2. 由于移动客户端与服务端往往使用的是无连接 REST 协议，因此对于用户的每一次请求都需要进行身份确认，服务端存储的确认信息需要大量多次访问，如果将这些数据都放到数据库中，就会大大降低访问效率，也会对数据库服务器造成极大压力；

3. 应用客户端登陆一次，身份验证通过后，若客户端关闭或服务器重启，往往会导致确认信息失效，用户需要重复登陆，操作麻烦，体验不好；

4. 对于即时通讯系统，当服务端接收到消息时，需及时将消息推送给接收者，延迟要尽可能小；

5. 对于用户数量庞大的系统，可能有大量用户并发访问，用户并发访问服务器响应速度和最大支持并发数也是考验服务器性能的关键因素和重要指标。

3.1.2 服务端架构设计

针对在 3.1.1 中提出的这些问题，进行了服务端端架构设计，如图 3-1，并提出以下解决方案：

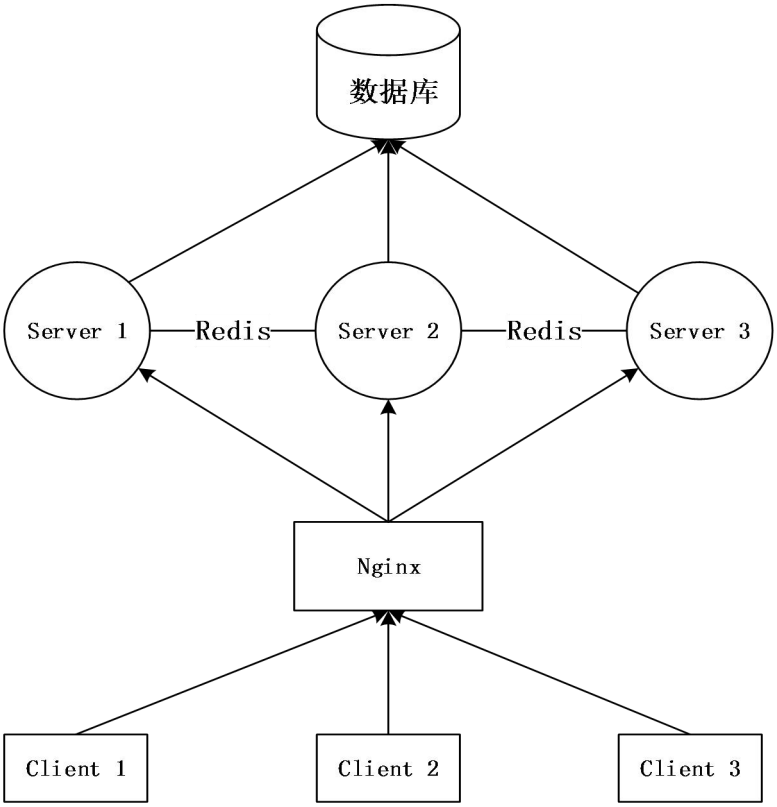

图 3-1 服务端系统架构图

1. 使用 Nginx 服务器作为负载均衡服务器：Nginx 是由是一个发布于 2004 年的一个高性能的 HTTP 和反向代理的开源服务器，其因具有很好的稳定性、丰富的功能集、和较低系统资源消耗的特点，是一个在各方面表现都良好的轻量级负载均衡服务器。对于使用多台应用服务器的情况，可以在 Nginx 配置文件中进行配置，当客户端发起请求给 Nginx 服务器，Nginx 服务器可以根据服务器忙碌状态将客户端发送的请求，转发到相对处理任务少的服务器上，再由应用服务器进行处理并返回[12]。

2. 使用 Redis 缓存数据库：Redis 遵从 BSD 开源许可，以 Key-Value 形式存储数据的内存数据库，支持多种数据类型的数据存储，包括字符串（Strings）、列表（Lists）、有序集合（Sorted sets）和散列（Hashes）[13]等，根据官方统计，其读速度 110000 次/s，写速度 81000 次/s[14]，相比读取数据库数据，读取 Redis 中的数据要快得多，这对于读取存储需要经常访问的数据的效率有极大的提高。由于每次客户端发送接收消息都需要确认用户身份，对客户端提供的确认信息的确认需要大量读写操作，因此将用户身份确认信息保存在 Redis 中，这可以极大提高服务器处理效率。用户确认信息包括两部分：tokenMap 和 userMap，其中是用户 ID 及身份确认 token 的键值对。每次用户发送消息及其他敏感操作要携带 token，如果用户提供的 token 与 ID 对应，则可保证用户身份合法，否则认为用户登录过期或者是冒用身份，服务器不予提供服务，强制客户端下线。

3. 身份确认信息持久化：使用 Redis 进行数据持久化[15]，如果服务器重新启动，但 Redis 数据并没有清空，可以保证登录过的用户的身份确认信息依旧有效，避免用

户因为服务器重启而导致的确认 token 不可用而导致登陆失效；在客户端登陆成功经过身份验证后，需要将服务器返回的身份确认 token 在客户端持久化，下一次启动应用，可使用已经经过确认的 token 与服务器通信，保证客户端无需重复登陆。

4. 使用 Socket.io[16]进行即时通讯：Socket.io 是一款即时通讯框架，包括 Node.js 实现服务端，JavaScript 实现浏览器端，Java 实现 Android 端，Swift 实现 iOS 端，这保证了其原生跨平台的特性。服务端的 socket.io 支持的通讯方式有多种，如表 3-1，其中通讯效率从上至下依次降级。WebSocket 是 HTML5[17]一种新协议，支持浏览器全双工通信，效率最高，如果服务端通过 WebSocket 与客户端建立连接，服务端与客户端相互沟通的 Header 大小大概只有 2 字节，同时，长连接情况下，服务端可以主动给客户端推送消息。现在主流浏览器已经支持 HTML5 协议，支持 WebSocket 连接，Android 客户端使用 Java8 编译，也支持 WebSocket。对于不支持 WebSocket 的浏览器，可以依次透明降级使用支持的最好的方式，这一降级过程对程序完全透明，这可以保证服务端的跨平台兼容性问题。

表 3-1 Socket.io 支持同性协议

支持协议	介绍
WebSocket	是 HTML5 开始支持的一种客户端与服务端通信的协议，在 2011 年被定为标准。全双工同性可以提高传输效率。在 Java7 中也实现了 WebSocket 协议。
Adobe Flash Socket	可以使用网页或程序嵌入的 Flash 来与服务端进行通信。
AJAX long polling	长轮训，服务端与客户端一直保持连接，当服务端有事件发生，响应给客户端。客户端不断向服务端发送数据包，请求获取最新数据。
AJAX multipart streaming	AJAX 流数据传输技术。使用 JavaScript 异步方式在客户端和服务端创建长连接。
Forever Iframe	流方式，浏览器通过内嵌 iframe，在客户端与服务端之间创建长连接。
JSONP Polling	跨域传输，用户可以使用服务端传送的 callback 参数进行相应处理。

5. 采用事件驱动和函数回调机制[18]：根据即时通讯特点——并发量大，消息内容简短频繁，使用 Node.js 和 socket.io 的事件驱动及函数回调的特性。用户发送一个请求，则触发服务器上的一个函数，在多不同任务需要处理时，会触发服务器多函数并发执行。同时，在处理过程中，使用函数回调机制，程序可以在等待回调过程中继续处理其他请求，而不必等函数完全处理完成。这可以提高程序并发性，充分利用系统资源。在多服务器情况下，可以使用 Redis 可以对数据进行同步，同时可过

socket.io-redis 在不同服务器间相互转发消息，保证消息转发可靠性。

3.1.3 服务端功能模块设计

系统按照功能模块，可以划分为八个模块，如图 3-2，各模块介绍如下：

1. 登陆模块：负责客户端使用用户名密码登陆，生成 token；

2. 注册模块：负责短信验证码发送，验证码验证，验证用户是否已经注册，用户注册成功信息保存；

3. Redis 操作模块：Redis 数据读写操作模块；

4. 数据库操作模块：对数据库进行操作；

5. 推送模块：负责给客户端推送信息，包括推送给多用户的应用更新信息，服务器维护信息，节日促销信息等；

6. 好友管理模块：好友申请、通过、删除等操作；

7. 实时消息转发模块：对客户端发送的实时消息进行转发，在接受者是某人时，将信息转发给接受者，当接受者是一个群组时，将消息转发给所有该群组内成员。服务端接收到消息后对客户端进行确认，监听客户端连接，并在监听到客户端连接时把用户离线时收到的消息推送给客户端。是系统核心功能模块。

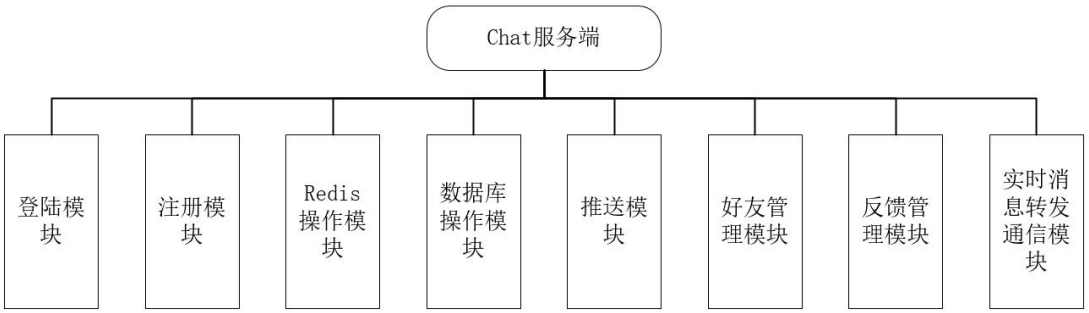

图 3-2 服务端功能模块图

3.1.4 服务端数据库设计

根据需求，服务端主要需要存储用户信息，用户间关系信息，用户间消息信息、群组信息和群组与用户关系信息等。据此，抽象出用户表，用户关系表，群组表，群组与用于关系表，消息表等，进行数据库设计，如图 3-3：

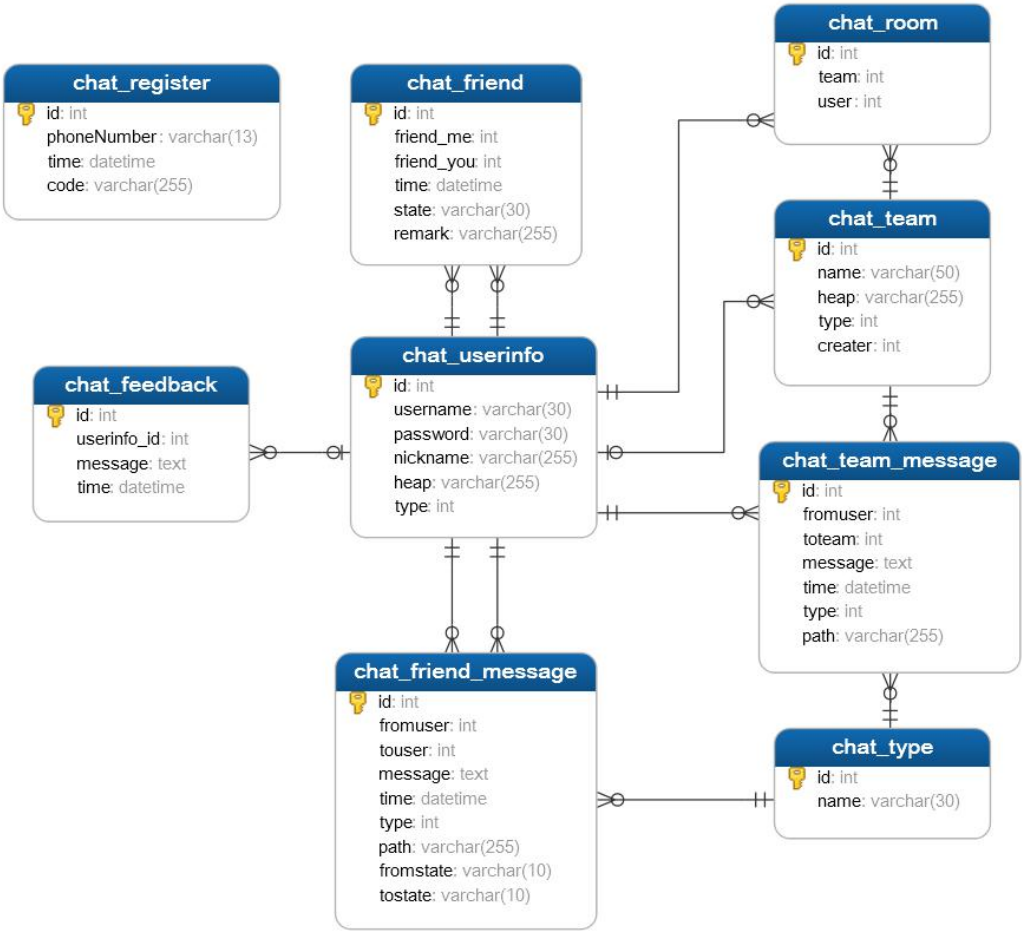

图 3-3 数据库模型 E-R 图

对数据库表解释如表 3-1：

表 3-1 服务端数据库表说明

编号	表名	描述
1	chat_register	注册短信验证码记录表
2	chat_userinfo	用户信息表
3	chat_friend	用户朋友关系记录表
4	chat_friend_message	用户朋友消息内容表
5	chat_type	消息类型表
6	chat_feedback	用户反馈内容记录表

3.2 客户端设计

对客户端进行功能模块、界面、缓存及协议等方面设计。

3.2.1 客户端功能模块设计

客户端使用 MVC 架构，并将网络访问，数据库操作，设置等分别用不同的包管理，方便扩展维护及修改。客户端系统功能模块图如图 3-4：

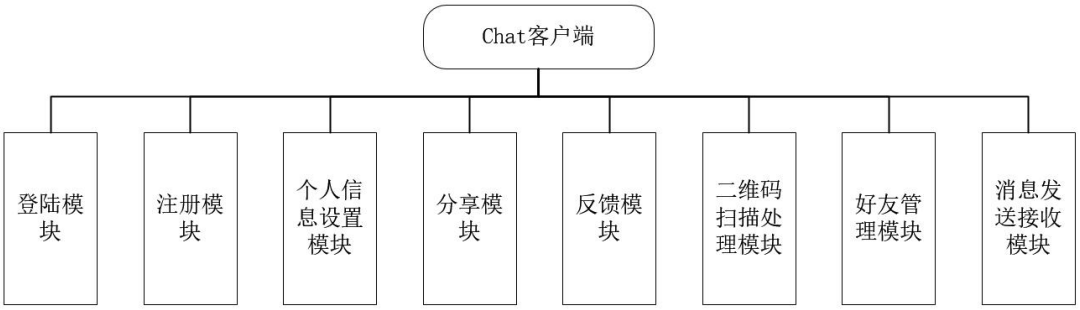

图 3-4 客户端功能模块图

3.2.2 客户端界面设计

客户端平台为 Android，在 Android 中布局是友 XML 文件确定或者是在 JAVA 代码中动态生成。本着简单易用的原则，登陆界面提供用户名、密码输入框及注册入口按钮，如图 3-5；应用主界面本着内存资源占用尽可能低，操作尽量简单的原则，将应用主界面设计为一个 Activity 中包含四个 Fragment 形式，使用 Tab 切换形式，如图 3-6；其他功能设置在菜单选项中，如图 3-7；聊天详细界面包含有有输入框和历史消息等，如图 3-8。

图 3-5 登陆界面

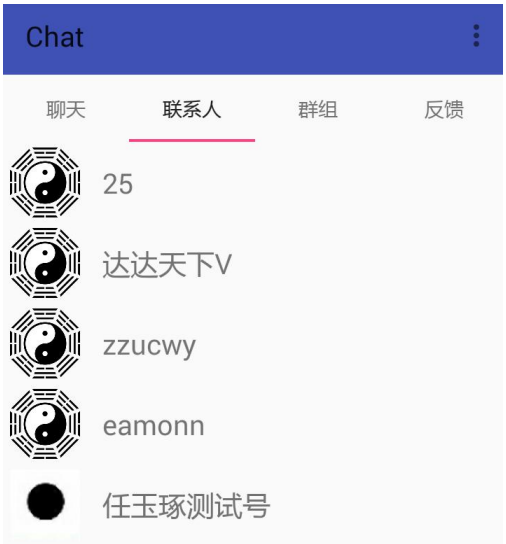

图 3-6 主界面

图 3-7 菜单界面

图 3-8 聊天详细界面

3.2.3 客户端数据库设计

数据库根据登陆用户不同，用户所使用的数据库表也不同，用“+用户 id”表示替换用户 id 后的表名，客户端数据库主要存储不易轻易更改且常用数据，包括历史消息，朋友列表等，如表 3-2：

表 3-2 客户端数据库表说明

编号	表名	描述
1	feedback	反馈
2	message+用户 id	朋友消息表
3	chat_friend+用户 id	朋友关系表
4	messagenumtoread+用户 id	未读消息条数记录表
5	userbrief+用户 id	聊天消息用户信息

3.2.4 客户端缓存处理

考虑即时通讯应用的独特性，历史消息不会轻易更改，历史记录查看频繁的特点，客户端应该缓存历史消息，对于多媒体文件，只要本地存在语音或图片文件，包括自己发送或接收到的文件都不应重复获取，减少网络流量，而且这样可以在客户端不联网情况下，仍可进行查看历史消息、查看用户信息等操作。

对于图片缓存采用 Picasso 图片缓存加载框架[19]，修改默认使用方式，增加文件级缓存，设置内存和文件两级缓存，达到提高加载速度减少网络数据流量的目的。

对于语音文件采用与服务端比对 MD5 值的方式，若 MD5 值相同，命名为同一文件不上传不下载。

历史消息缓存在数据库中。

3.2.5 客户端其他功能设计

客户端其他功能设计包括：

1. 二维码相关：二维码生成扫描使用 Google 开源框架 ZXing[20]，可以根据用户信息本地生成二维码供其他用户扫描。同时扫描二维码提供直接扫描二维码和图片文件扫描两种方式，在黑暗条件下提供闪光灯打开方式；

2. 应用分享：基于 ShareSDK 多平台分享解决方案，对 ShareSDK 进行了修改，在自由申请其他平台分享权限后，进行配置可进行应用分享，提供微信、QQ、微博、人人等分享方式；

3. 一对多推送：使用 JPush 极光推送，JPush 是一被广泛使用的可在低流量消耗情况下进行及时推送的第三方框架，其具有 20 万条/s 的推送速度。

4. 错误日志收集：服务端日志信息保存在 myLog.log 文件中，客户端异常错误日志信息保存在/chat/logs/errlog.log 文件中，客户端出现异常时会自动重启，并将错误日志信息上传服务器。

3.3 通信协议设计

服务端与客户端通信使用 REST+JSON 通信协议。REST 是一种万维网的软件架构风格，它是基于 HTTP，URI，XML，HTML 这些协议和标准的设计风格，它适用于客户端与服务端之间无状态连接情况，相比 HTML 连接可以降低资源消耗，充分利用缓存，提高效应速度。JSON 是轻量级数据交换格式，易于人阅读和编写，易于机器解析，采用键值对方式，支持对象、字符串、数值和数组等，易于通信。本系统核心业务包括消息发送及接收，且具有偶发性大，数据量小的特点，REST+JSON 的配合使用，可以很好的满足需求。

3.4 本章小结

本章通过对服务端问题分析，提出解决方案，对服务端进行架构设计并给出了系统架构图，对服务端功能模块进行了分析，对数据库表和主外键约束进行了设计；

本章也确定了客户端系统范围，以简洁高效的原则设计了主界面风格，对客户端数据库进行了设计，考虑客户端应用的独特性，针对缓存、推送和分享等方面提出了解决方案。

4 移动端实时聊天系统实现

本章针对设计，对各功能模块进行了编码实现，并对数据格式、业务逻辑和关键代码等进行了说明和介绍。

4.1 注册登录功能的实现

1. 登陆功能

用户第一次安装客户端，进入应用后，会跳转到登录界面，用户需要输入用户名和密码，客户端将用户名和密码加密后提交给服务端，服务端根据请求，查询是数据库是否有该用户信息，如果数据库包含用户信息，则认为此次登录请求成功，服务端动态生成唯一动态令牌 token，并将该 token 与登录用户绑定起来，唯一对应，将 token 与用户对应关系缓存到 Redis 中，作为缓存使用，并将用户 token 与用户其他个人信息一并返还给客户端，客户端收到登录成功返回信息后，将 token 及用户个人信息持久化到本地，作为缓存使用，同时，也可以在重新进入应用后不需要重新从服务端获取信息直接可以进行操作，以后所有关键操作用户需要在 HTTP 请求头或者是消息体里携带用户 token 信息，服务端通过 token 获取找到对应用户，并与客户端提交的操作执行者身份进行比对，如果一致，认为此次请求合法，继续提供服务，否则认为请求非法，不予提供服务，返还客户端 403 请求状态码，客户端收到 403 状态码后，清空登陆 token 及个人缓存信息，退出登录，迫使非法用户重新登录，登陆流程如图 4-1：

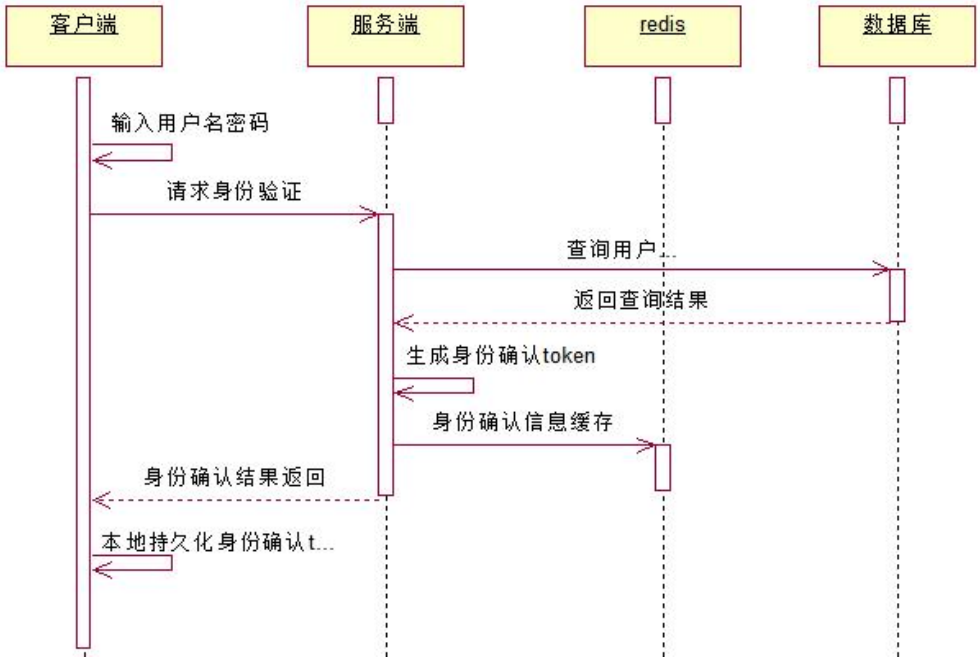

图 4-1 用户登录流程时序图

2. 注册功能

如果用户没有账号，可以通过登陆界面直接进入注册界面，提供了两种注册通道，两种注册通道的区别为后台发送验证码方式不一致，分别使用阿里大鱼（付费可靠）和 SMS For Android（免费不稳定），用户需要输入合法手机号，手机接收动态短信验证码，用户提交验证码与手机号进行比对，如果一致，认为该手机号合法，否则不允许进一步操作。通多手机号验证的用户可以继续选择输入个人相关信息进行注册，注册成功即可登录。

4.2 好友管理功能的实现

客户端发送 HTTP 请求，服务端通过 Express 接收客户端请求，/friends 请求获取所有好友列表，/friend 获取某好友详细消息，/userinfo 获取非好友用户公开信息，/addFriend 添加好友，/updateRemark 修改备注信息，/invite 发出好友申请，/submitFriend 同意用户请求，/deleteFriend 删除好友。不同请求对应不同数据库操作，将数据组织后，结果返回给客户端。

4.2.1 添加好友

1. 扫描二维码：用户可以通过扫描好友二维码进行添加，二维码生成及解析都由客户端动态生成，不消耗网络流量，二维码直接包含用户个人基本信息，不需要从网络获取。获取用户信息后可以直接点击添加好友发出好友请求。使用 Google 的开源框架 ZXing 实现。

2. 手机号模糊查找：用户可以通过手机号进行好友添加，有模糊查找和精确查找两类，服务端可根据具体应用进行配置，查找到用户信息后可以选择添加好友。

4.2.2 同意好友申请

用户在发出申请后，需要等待对方同意申请才能成为好友关系。考虑好友申请并不是十分频繁，好友申请数据在每次进入应用会从服务端获取一次，在进入好友查找界面不进行主动刷新，减少请求次数，可以手动刷新、下次进入应用再次获取或服务端推送好友请求时再次获取。

4.2.3 删除好友

好友关系是双向关系，但是在一方进行删除后，系统认为两用户不希望再次联系，会将好友关系双向删除，如果只是想单项删除，可以使用黑名单实现。

4.3 消息发送和接收功能的实现

4.3.1 普通文本消息发送和接收

通过 socket.io 连接，将消息内容转换成 JSON 格式，通过 private 事件发送给服

务端，服务端接收到消息后，进行身份确认后将消息转发给接收者，同时给发送者以ACK 事件形式发送确认回执，发送者把消息从正在发送状态修改为已经发送状态，修改发送时间为服务端时间，消息 id 为服务端数据库 id。接收者接收到消息后，判断时间是否显示，修改用户未读消息数目，更新界面 UI 显示。

4.3.2 多媒体消息发送和接收

需要发送多媒体数据时，首先将图片或语音消息进行预处理，在本地缓存，生成七牛云 token，然后将图片或语音存储到七牛云，根据七牛云返回结果，解析出对象访问 URL，将 URL 设置到消息 path 字段，并将消息类型设置为图片或语音，然后将消息以普通文本消息形式发送。接受者接收到图片或语音消息，首先将图片或语音下载到本地进行缓存，修改界面显示情况，以显示语音或图片形式显示。如果消息第二次加载，在获取多媒体数据时先判断本地是够已经有缓存文件，如果已经存在，不重复获取。直接使用本地缓存减少数据流量，提高访问速度。如图 4-2。

对于语音消息，采用了 zip 格式[21]进行加密压缩，在用户录音结束后，客户端对于语音消息进行压缩，并设置密码，然后将消息发送给服务端；接收方接收到消息后，在需要播放语音时进行预加载，将语音消息解压并解密，当退出应用时，删除解密后文件保证安全。同时由于 zip 的压缩，文件大小也有相应减小，经多次测试，文件大小大约可以减小 12%，相比未压缩情况下，这极大的减少了网络通信流量，节省服务端存储空间。

对于图片消息，在文件发送时，将图片进行剪切并压缩，一般情况下，图片大小控制在数十到数百 KB 内不等，对于较大图片，基本每张图片都控制在 350KB 内，这在移动端分辨率情况下，显示效果基本不变。这在保证了用户体验情况下，也减少了网络通信流量，提高传输效率，同时避免了客户端由于加载大图片而造成的内存占用过多而导致的程序崩溃。

对于多媒体消息的接收，支持使用了七牛云 CDN（Content Delivery Network，内容分发网络）加速技术。CDN 技术是在网络节点中设置虚拟网络，CDN 系统根据客户端请求，根据网络流量和各节点的连接情况、负载情况，以及服务节点到客户端距离等综合信息，将用户请求转发到离用户最近网络状况最好的节点上，客户端可以就近获取多媒体资源，解决网络拥挤的情况，加速客户端获取资源的速度。

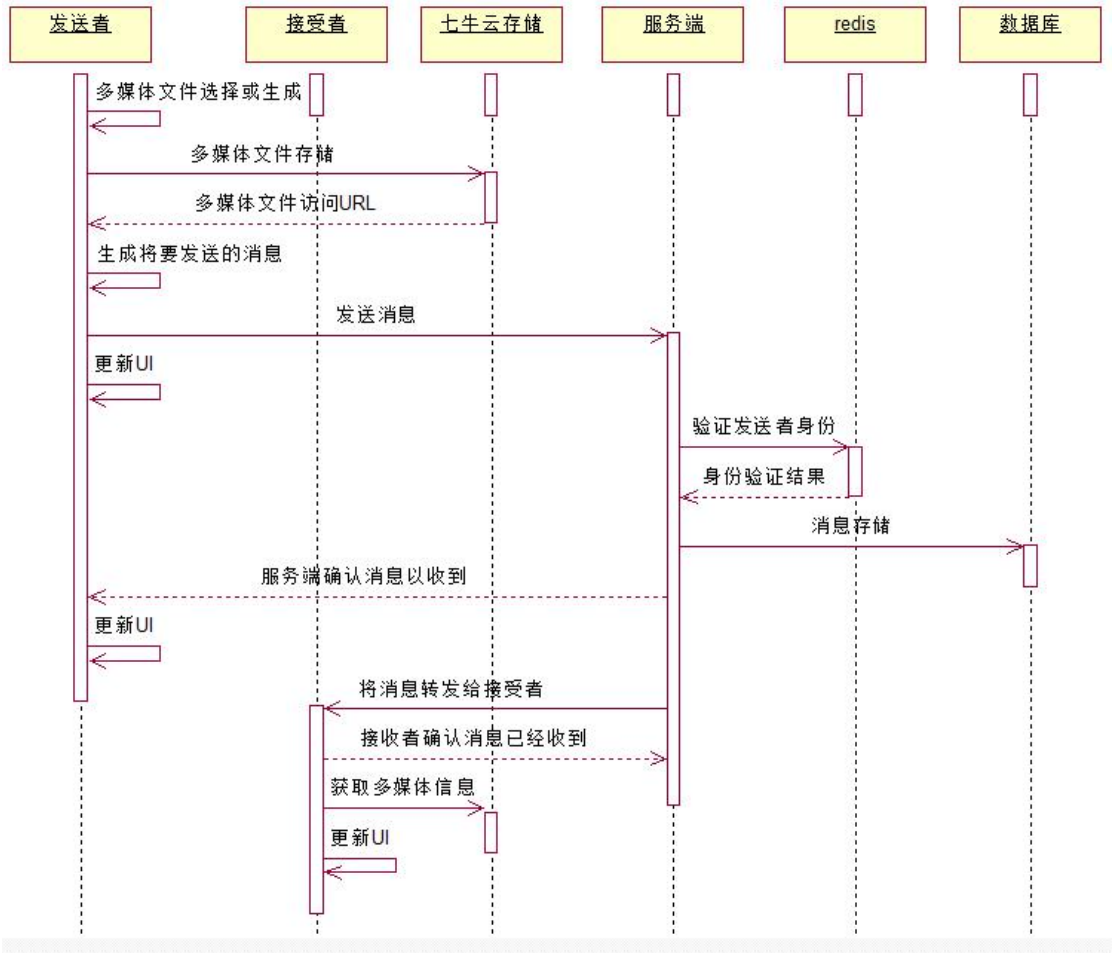

图 4-2 多媒体消息发送时序图

4.4 扫描二维码功能的实现

随着智能机的普及，加之二维码简便易用的特点，越来越多应用需要二维码扫描功能。在本应用中，集成 Google 的开源二维码处理框架 ZXing，客户端可以根据个人信息生成不同分辨率的二维码，其他用户可以直接使用 ZXing 的二维码扫描功能进行扫描添加好友的操作，二维码数据通过 JSON 传递，有两种形式，数据格式如下：

```
//手机相机像素高，包含更多信息
{"heap":"http://7xswvn.com1.z0.glb.clouddn.com/FjFQFrSVT2EDaqrVgUzyYALVYSwB","nickname":"玉琢","username":"18337146328","userId":24}
//手机相机像素低，只包含基本信息
{"nickname":"玉琢","username":"18337146328","userId":24}
```

4.5 历史消息的实现

对于好友消息，服务端标记用户是否已读的状态，在用户连接到服务端时，服务端将此用户所有未读消息推送给客户端，客户端接收到后给服务端返回回执，这可以保证用户能接收到所有用户消息。

对于群组消息，由于群组消息一般较多，不适合一次将所有消息都发送给客户端

啦，客户端记录所有群组已读消息情况，在上线时分别获取不同群组未读消息，可以保证客户端接收到所有用户消息。给出注册回调函数及获取历史消息关键代码客户端实现：

```
//注册重连监听事件，注册重连回调函数
mSocket.on("connect", connect);
//重连时获取历史消息
Emitter.Listener connect = new Emitter.Listener() {
    public void call(final Object... args) {
        new Runnable() {
            public void run() {
                //将自己用户信息报告给服务器
                UserInfo userInfo = new UserInfo();
                userInfo.setId(ChatApplication.getUserId());
                userInfo.setToken(ChatApplication.getToken());
                //向服务器发出获取历史消息请求
                mSocket.emit("history",JsonUtils.
                    transFormObjectToJsonObject(userInfo));
                ChatMainActivity.getAllTeamUnreadMessage();
                LogUtil.log("reconnect");
            }
        }.run();
    }
};
```

4.6 免密码登陆的实现

应用安装后，在本地生成一配置文件，其中保存用户基本信息、身份确认 token 和崩溃处理标志，应用在启动时从文件中加载各种信息，如果获取不到，认为用户没有登录过，需要用户进行登录，并且在用户登录成功后，将获取到的各种信息进行本地持久化，在下次应用启动时就不再需要登录。用户在退出登录会删除所有配置文件中个人信息及确认 token 信息。服务端在登陆成功时生成 token 使用 md5 加密算法，随机生成，生成的 token 与用户 id 唯一对应，服务端关键代码如下：

```
//通过用户名查询密码
database.login(username, password, function(res){
    // 用户名不存在时返回 401 状态码
    if(res.toString() === ''){
        response.writeHead(401);
        response.end('{"res":"err"}');
    }else{
       //登录成功，生成确认身份 token
        var md5 = crypto.createHash('md5');
        md5.update(((Math.random() * Math.random())).toString());
```

```
        var token = md5.digest('hex');
       //从返回值中获取用户 id
       var userId = res[0]['id'];
       redisOption.setToken(token, userId, function(id){
           if(id === undefined){
               response.writeHead(403);
               response.end('{"res":"fail"}');
           }else{
               response.writeHead(200);
               responseResult='{"res":"success","token":"'+token+'",'
+'"id":'+res[0]['id']+',"nickname":"'+res[0]['nickname']+'","heap":"'+res[0]['h
eap']+'",'+'"type":'+res[0]['type']+'}';
               response.end(responseResult);
            }
        });
    }
});
```

4.7 本章小结

本章根据功能模块划分，包括注册模块，好友管理模块，群组管理模块，消息发送接收模块，分享功能模块，扫描二维码模块，获取历史消息模块，免密码登陆等几大主要功能模块的实现。详细阐述了各功能模块的实现，给出了关键模块的处理流程图和时序图。

5 系统部署及测试

本章对系统进行了部署，设计了测试用例，对系统功能和性能分别进行了测试，给出了功能测试结果和性能测试性能指标。

5.1 系统部署

将服务端系统部署到腾讯云服务器，由于设备的有限性，将客户端部署到联想K30-T 真机和一模拟器进行测试，硬件和软件环境如表 5-1：

表 5-1 部署环境

序号	设备	数量	硬件配置	软件配置
1	云服务器	1	1、CPU：1CPU 2、内存：1GB 3、硬盘：20GB 4、上网宽带：1Mbps	1、Ubuntu Server 14.04.1 2、LTS 32 位 3、Nginx1.9.9 4、Node.js v5.2.0 5、Mysql5.7 6、redis-server3.0
2	Lenovo K30-T	1	1、处理器：4 核 1.2GHZ 2、运行内存：1GB	1、Android 4.4.4 2、浏览器：系统自带浏览器 3、应用客户端：本系统客户端
3	Android 模拟器	1	1、处理器：4 核 1.2GHZ 2、运行内存：1GB	1、Android 5.0 2、浏览器：系统自带浏览器 3、应用客户端：本系统客户端
4	无线路由器	1	1、TP-LINKTL-WR886N 2、速度 450MB/S 3、协议 802.11n	无

5.2 系统测试

5.2.1 功能性测试

功能性测试是要测试在输入必要测试数据时是否可以产生预期输出，生成预期结果。在测试过程中，要注意用例的代表性和对边界进行测试，避免因为输入的不当而导致测试的不全面，找不到系统隐藏错误。

下面是对登陆、注册，发送信息和好友管理等业务进行测试，见表 5-2、表 5-3、表 5-4：

表 5-2 用户注册及登陆功能的测试

用例编号	T0001	类型	功能测试
测试目的	测试用户注册及登陆功能		
前提条件	系统没有待注册用户信息		
测试步骤	1. 进入系统注册界面； 2. 填写合法手机号，点击发送验证码； 3. 用输入手机接收发送过来的验证码，在系统验证码输入窗口中输入收到的验证码； 4. 点击注册验证按钮； 5. 提示验证成功，请输入用户名等信息，手机号不可更改； 6. 填写其他必要信息点击注册； 7. 提示注册成功，关闭注册窗口，跳转到登陆窗口； 8. 输入刚刚注册的用户名和密码，点击登陆按钮； 9. 登录成功。		
测试结果	1. 成功发送验证码； 2. 验证码与手机号匹配验证成功，否则提示验证失败； 3. 输入必要数据，提示注册成功，关闭注册窗口跳转到登陆界面； 4. 使用注册信息登陆，提示登陆成功，跳转到系统主界面。		

表 5-3 发送消息功能的测试

用例编号	T0002	类型	功能测试
测试目的	测试发送消息功能		
前提条件	用户登录成功		
测试步骤	1. 用户点击好友进入聊天窗口界面； 2. 在输入框内输入文本信息，点击发送； 3. 在输入框输入表情，点击发送； 4. 点击多媒体消息发送按钮，点击语音发送按钮； 5. 按住录音按键，语音输入后松开录音按键； 6. 点击多媒体消息发送按钮，点击图片发送按钮； 7. 选择图片发送。		
测试结果	1. 文本消息发送成功，对于消息为电话号码的点击直接跳转到系统拨号窗口，为邮件的点击直接跳转到邮件发送界面，为网址的直接调用系统浏览器； 2. Emoji 表情可以正常发送及显示； 3. 语音消息可以正常发送，接受者可以收听语音； 4. 可以发送并接收显示图片。		

表 5-4 好友管理功能测试

用例编号	T0003	类型	功能测试
测试目的	测试好友管理功能		
前提条件	用户登录成功		
测试步骤	1. 用户点击菜单添加好友按钮； 2. 在好友查找选择框内输入要查找好友手机号，点击查找按钮； 3. 选择某一用户，点击进入用户详情界面； 4. 点击添加好友，发出好友申请。 5. 被邀请用户进入好友添加界面； 6. 被邀请用户点击同意按钮； 7. 刷新好友列表； 8. 向左侧滑动要删除好友； 9. 点击删除好友。		
测试结果	1. 用户发送好友申请成功； 2. 被申请者可以收到好友申请，并可以同意好友申请； 3. 删除好友后朋友界面看不到好友信息。		

5.2.2 非功能性测试

并发性测试：由于本系统暂时并没有大量用户，不能同时并发发送大量数据，因此只能由一个客户端循环发送多条消息。经测试，在服务端打印日志信息时，同时发送 1500 条消息时，统计一秒内消息增长情况，数据库一秒内消息 id 从 1383 增长到 2481（其他信息不在此一秒内不做统计），服务端一秒内共处理 1099 条消息，达到目标一秒内 1024 并发处理请求，在更少日志打印情况下和多用户并发发送情况下效率会更高。

可靠性测试：关闭数据库 MySql 服务和 Redis 服务，服务端程序会主动尝试重连，当服务恢复时，重连成功，恢复服务。虽然服务端没有经过长时间压力测试，但是连续运行 30 天，服务几十个用户没有发现任何异常崩溃情况。

安全性测试：客户端和服务端关键数据经过加密，在不知加密算法时，无法破解；七牛云 token 和 JPush 推送公钥和私钥，包含公钥和私钥，在不知道公钥和私钥时无法破解。

6 总结与展望

6.1 总结

随着移动互联网的高速发展，智能机和平板电脑移动终端的普及，人与人沟通方式正发生着巨大的变化，即时通讯框架的出现和发展对避免重复造轮子，简化应用开发和使用者工作具有重要意义和价值。

本文主要研究的工作和创新性设计主要包括以下几个方面：

1. 本文以实际应用为背景，以打造行业规范框架为目标，对即时通讯系统进行了分析、设计和实现，具有良好的安全性，实时性和可扩展性，可以方便集成到其他项目及平台使用。采用经典 MVC 架构的设计，提高应用可扩展性。

2. 系统跨平台框架及应用平台无关性，大大提高了平台可用性，减少了不同平台维护成本。服务端与客户端交互，抛弃使用传统 HTTP 请求应答的方式，改用 socket.io，在减少资源占用情况下效率更高，实时性更强。同时服务端采用 Node.js，使用语言特性，以事件驱动，大大提高了服务端有限资源情况下，数据处理吞吐能力。

3. 系统架构设计，原生支持集群部署；三级缓存设计，优化系统访问速度；NoSql 数据库与关系数据库结合使用，发挥各自优势；负载均衡分流，降低单台服务器压力。这些设计都为在用户大规模增长情况系统升级扩展打下了良好基础。

4. 其他辅助功能完善，包括应用分享，二维码扫描，错误日志收集等，可以简便扩展成以即时通讯为核心的其他应用。

6.2 展望

本文虽然对实时性通信系统进行了研究和设计，实现了即时通讯功能，根据实际业务需求，仍有很多内容有待深入研究：

1. 数据是企业的命脉，在系统中，暂时还缺少数据备份，双机热备和容灾处理等相关考虑；

2. 地理位置信息也是企业提供服务的重要依据，针对用户所处位置，可以提供更有针对性的服务，但是本暂时系统没有针对自动定位等相关功能，例如摇一摇，附近的人等功能，给用户更多选择更好体验；

3. 在网络飞速发展的今天，网速问题也逐渐解决，不支持视频交流，不支持语音通话，界面设计不够美观，用户使用体验还有待提高。

因此本系统还有许多不足，还需要自己不断学习，积累经验。我相信随着自己专业学习的深入，技术能力的提高，对本系统的不断改进、扩展和完善，会将本系统扩展为跟具实用性的即时通讯框架。

致谢

在本论文完成之际，我要向所有关心过我的老师和同学们表示我衷心的感谢。

首先，感谢我的指导老师郑志蕴老师。郑老师专业的严谨、渊博的知识、敬业的态度以及和蔼的性格，都让我获益匪浅，是我终身学习的榜样。在本论文选题和撰写过程中，郑老师给出了很多宝贵的意见和建议，本文的顺利完成与郑老师的悉心指导是分不开的，在这里向郑老师的付出表示最衷心的感谢。同时，感谢我实习过程中的给予我巨大帮助的王磊老师，王老师精湛高深的专业水平，平易近人的优秀品质，精益求精的探索精神，都为我树立了良好的榜样。

其次，感谢郑州大学和信息工程学院对我的培养，感谢在本科四年中帮助过我的老师卢红星老师、宋玉老师、庄雷老师、姬波老师、谢琦老师以及所有帮助过我的老师，感谢你们的无私教导和帮助。

同时，感谢我的室友和朋友，感谢一路上有你们陪伴，你们的友情是我一生的财富。

最后，感谢我亲爱的家人，你们在我成长过程中一直默默的付出、理解和支持，你们让我感受到温暖，你们是我最坚强的后盾，我无以回报。

祝你们永远开心幸福！

参考文献

[1] The growth of China's mobile use is mind-blowing[EB/OL]. http://www.businessinsider.com.au/china-mobile-growth-2015-7.

[2] 李锐. 浅谈即时通讯工具现状及其发展趋势[J]. 中国科技信息. 2013(16): 86-87.

[3] 熊刚, 余智华, 郭莉. 即时通讯的现状、未来走向 和研究挑战[J]. 中国计算机学会通讯. 2016, 2(1): 54-59

[4] 罗伟. 基于 Android 平台的即时通讯系统的研究与实现[D]. 湖南师范大学. 2009

[5] 郭鑫杰. 即时通讯系统的设计与实现[D]. 南京大学. 2012

[6] 黄经赢. 基于 Socket.io+Node.js+Redis 构建高效即时通讯系统[J]. 现代计算机(专业版). 2014(19): 62-64+69

[7] JOSH, POLLOCK. INTRODUCTION TO THE JSON REST API[EB/OL]. http://torquemag.io/2014/08/introduction-wordpress-new-universal-connector-json-rest-api/.

[8] 李兴华. 基于 WebSocket 的移动即时通信系统[D]. 重庆大学. 2013

[9] 韦广剑. 基于 Android 令牌的动态密码认证系统的研究与实现[D]. 武汉理工大学. 2012

[10] 王薇薇,李子木. 基于 CDN 的流媒体分发技术研究综述[J]. 计算机工程与应用. 2004(08): 121-125

[11] Using NGINX and NGINX Plus with Node.js and Socket.IO, the WebSocket API[EB/OL]. https://www.nginx.com/blog/nginx-nodejs-websockets-socketio/.

[12] 张俊杰. 基于用户兴趣模型的推荐算法及系统实现研究[D]. 上海大学. 2014

[13] 曾泉匀. 基于 Redis 的分布式消息服务的设计与实现[D]. 北京邮电大学. 2014

[14] How fast is Redis?[EB/OL]. http://redis.io/topics/benchmarks.

[15] 白鑫. 基于 Redis 的信息存储优化技术研究与应用[D]. 北方工业大学. 2014

[16] nuclearace. Socket.io[EB/OL]. https://github.com/socketio.

[17] World Wide Web Consortium. HTML5.[EB/OL]. https://www.w3.org/TR/2014/REC-html5-20141028/.

[18] 崔康. NodeJS 的异步编程风格[EB/OL]. http://www.infoq.com/cn/news/2011/09/nodejs-async-code.

[19] square. Picasso.[EB/OL]. http://square.github.io/picasso/.

[20] 李璐. 基于 ZXing 的条码技术研究 [J]. 福建电脑. 2014(05): 17-18+156

[21] Zip(file format).[EB/OL]. https://en.wikipedia.org/wiki/Zip_(file_format).