浙江科技学院

2021 届本科毕业设计

题　　目　即时通讯系统的设计与实现

学　　院　信息与电子工程学院

专　　业　软件工程

班　　级　172 班

学　　号　1170290064

学生姓名　孙浩

指导教师　庄儿

教师职称　讲师

完成日期　2021 年 5 月 23 日

浙江科技学院毕业设计（论文）、学位论文
版权使用授权书

本人 孙浩 学号 1170290064 声明所呈交的毕业设计（论文）、学位论文《 即时通讯系统的设计与实现 》，是在导师指导下进行的研究工作及取得的研究成果。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，与我一同工作的人员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谢意。

本毕业设计（论文）、学位论文作者愿意遵守**浙江科技学院** 关于保留、使用学位论文的管理办法及规定，允许毕业设计（论文）、学位论文被查阅。本人授权**浙江科技学院**可以将毕业设计（论文）、学位论文的全部或部分内容编入有关数据库在校园网内传播，可以采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文）、学位论文。

（保密的学位论文在解密后适用本授权书）

论文作者签名：孙浩　　导师签名：

签字日期：2021年05月30日　　签字日期：2021年5月31日

摘要

在越发高度信息化的移动互联网时代，即时通信在互联网中有着举足轻重的地位。在一些在线游戏、社交软件等应用中，IM 是其中不可或缺的功能模块。即时通信其高效迅捷的交流方式打破了空间的束缚，使得人类的生活更加丰富多彩。

在众多的场景中都需要一套底层的 IM 系统以支持，本文调研了业内常用的分布式 IM 系统相关的解决方案，针对要支持大规模并发 IM 这样的场景，本文设计和实现了一套可行的 IM 架构方案，以保证 IM 系统的高可靠性、高可用性、实时性以及有序性。传统的 IM 系统喜欢将所有的服务统筹在一起，这样就会造成服务代码复杂，难以持续开发和运维等问题，而本文则尽可能地追求微服务化和模块化，希望能够实现 IM 系统的可扩展性以及服务之间相对的隔离性，减少服务失败造成对其它服务的影响。

关键词 IM 系统；实时通信；消息推送

ABSTRACT

In the era of more and more highly information-based mobile internet, instant communication plays an important role in the internet. In some online games, social software and other applications, IM is an indispensable function module. Instant messaging its efficient and rapid communication way to break the constraints of space, so that human life is more colorful.

In many scenarios, a low-level IM system is needed to support. This paper investigates the solutions of distributed IM systems commonly used in the industry, in this paper, a feasible IM architecture is designed and implemented to ensure the high reliability, high availability, real-time and order of the IM system. The traditional IM system likes to integrate all the services together, which makes the service code complex, difficult to develop and maintain continuously, and so on, we hope to achieve the scalability of IM system and the relative isolation between services, to reduce the impact of service failure on other services.

Keywords IM system; real-time communication; message push

目录

第 1 章 引言

1.1 项目背景

即时通讯一般用于需要即时发送和接收互联网信息等的业务场景。自即时通讯的模式面世以来，它就不停的飞速发展，并在发展期间迅速丰富其功能，逐渐形成了音乐、电视、游戏、电子邮件等功能。可以说，即时通讯不再是单单的聊天工具，它已经发展为集娱乐、资讯、办公协作等为一体的综合化平台了。

而经过十多年的本土化发展后，即时通讯在国内大局已定，腾讯公司是毋庸置疑的龙头老大，凡是与即时通讯相关的领域腾讯无处不在。但从市场上来看，如果想要打破这种垄断状态是需要在即时通信领域上做出革命性的技术突破的。

1.2 研究意义

即时通信从宏观角度上划分可以简单分为两个大类，第一类是面向个人的即时通信类商业软件产品；第二类是企业内部使用的即时通信。其中面向个人的商业软件产品毫无疑问是目前市场的主流，因此，如何成功地构造出一个更优的面向个人的即时通信产品是一个重要的研究方向。

首先笔者要提出的一点是，市面上已经有非常多且非常优秀的开源的即时通信类的项目，我们尽量选择站在巨人的肩膀上，对于这些造好的轮子，取其精华，去其糟粕，分析其中好的架构模式，找寻让即时通信效率更高的技术手段。

一款优秀的即时通信软件，是必须要保证实时性、可靠性、一致性和安全性的。实时性保证的是消息的实时触达，可靠性保证消息的不丢失和不重复，而一致性则是保证同一个设备的时间顺序、不同设备的漫游同步，安全性保证数据传输安全、数据存储安全、消息内容安全。实时性、可靠性、一致性和安全性是通信系统的重要性能指标。

本文提出的分布式通信架构和一些技术方案可以满足实时性、可靠性、一致性和安全性的特性，可以构建出更可靠、更优秀的通信服务器。这样的通信框架如果被更多即时通信行业中的中小企业、创业公司采用，会减小它们的成本，促进这些企业更加平稳积极地发展。

1.3 国内即时通信技术应用场景

(1)微信、qq、钉钉等主流 IM 应用：这是 IM 技术的典型应用场景；

(2)微博、知乎等社区应用：它们利用 IM 技术实现了用户私信等点对点聊天；

(3)抖音、快手等直播/短视频应用：它们利用 IM 技术实现了与主播的实时互动；

(4)米家等智能家居物联网应用：利用 IM 技术实现实时控制、远程监控等；

(5)滴滴、Uber 等共享家通类应用：利用 IM 技术实现位置共享；

(6)在线教育类应用：利用 IM 技术实现在线白板。

第 2 章 即时通信系统设计

2.1 系统架构设计

该平台采用 B/S 模式，前端也就是客户端采用 vue 作为主要服务，后端主要采用 SpringBoot+Tio 进行开发,采用简洁的消息格式实现客户端到服务端的消息发送。数据库使用 redis 作为缓存，mysql 用于持久化数据。

2.2 功能模块设计

客户端模块设计：用户可以注册账号并登录进入系统，首先会进入到聊天界面，通过侧部导航可以进入到通讯录显示已经添加的好友和群组，也可以添加好友，点击侧部导航栏最上方可以查看和修改自己的个人信息和退出登录。

图 2-1 系统功能模块图

2.3 数据库设计

2.3.1 系统实体设计

如表 2-1 所示，本系统使用 MySQL 作为后台数据库，其中数据库名为 mim, 包括以下几个表，分别为：

表 2-1 实体说明表

user	用户
group	群组
chat_type	聊天类型
chat_person_record	私聊聊天记录
command	命令请求
friends	好友关系表
user_group	用户和群组的关系表

chat_group_record	群聊聊天记录
char_offine_msgs	群离线消息表

2.3.2 数据表设计

用户表的数据库表如表 2-2 所示：

表 2-2 用户表 user

字段名	类型	描述
id	int	主键 id
user_id	varchar	用户 id
nick	varchar	用户昵称
avatar	varchar	用户头像
create_time	timestamp	创建时间
last_login_time	timestamp	最后一次登录时间
sign	varchar	个性签名
username	varchar	用户名
password	varchar	密码

聊天类型表的数据表如表 2-3 所示：

表 2-3 聊天类型表 chat_type

字段名	类型	描述
id	int	主键 id
chat_type	int	聊天类型，分为私聊、群聊和未知，其中私聊为 2，群聊为 1，未知为 0

个人聊天记录表的数据表如表 2-4 所示：

表 2-4 个人聊天记录表 chat_person_record

字段名	类型	描述
id	bigint	主键 id
from	varchar	发送用户 id
to	varchar	目标用户 id
msg_type	int	消息类型，如(0:text、1:image 、 2:voice 、3:vedio 、 4:music 、5:news)
content	varchar	消息内容
send_time	timestamp	发送时间
receive_time	timestamp	接收到消息的时间

群聊天记录表的数据库表如表 2-5 所示：

表 2-5 群聊天记录表 chat_person_record

字段名	类型	描述
id	bigint	主键 id
from	varchar	发送用户 id
group_id	varchar	目标群组 id
msg_type	int	消息类型，如(0:text、1:image 、 2:voice 、3:vedio 、 4:music 、5:news)
content	varchar	消息内容
send_time	timestamp	发送时间
receive_time	timestamp	接收到消息的时间

群组表的数据库表如表 2-6 所示：

表 2-6 群组 group

字段名	类型	描述
id	int	主键 id
group_id	varchar	群组 id
avatar	varchar	群组头像
member_count	int	群组人数

命令表的数据表如表 2-7 所示：

表 2-7 命令表 command

字段名	类型	描述
id	int	主键 id
command_type	int	命令请求类型

friends 的数据库表如表 2-8 所示：

表 2-8 好友关系表 friends

字段名	类型	描述
id	int	主键 id
user_id	varchar	用户 id
friend_id	varchar	好友 Id

用户群组关系表的数据库表如表 2-9 所示：

表 2-9 用户群组关系表

字段名	类型	描述
id	int	主键 id
user_id	varchar	用户 id
group_id	varchar	群组 id

群离线消息表的数据库表如表 2-10 所示：

表 2-10 群离线消息表 chat_offline_record

字段名	类型	描述
id	int	主键 id
user_id	varchar	用户 id
group_id	varchar	群组 id
msg_id	bigint	消息 id
msg_content	varchar	消息内容

2.4 消息 ID 设计

如果消息 ID 采用字符串方式，那么会造成存储空间的大量占用，而且不能利用存储引擎的特性使得相邻的消息存储在一起，会影响消息的写入和读取性能。因此对于消息，我们同采使用数字类型，但是如果数字是随机的，那么我们也无法利用存储引擎的特性让相邻的消息存储在一起，会加大随机 IO，降低消息的写入和读取性能，而且随机的 ID 也很难保证全局唯一性，因此我们一般都是使得消息 ID 递增的。

2.4.1 消息 ID 递增性

如果消息 ID 采用字符串方式，那么会造成存储空间的大量占用，而且不能利用存储引擎的特性使得相邻的消息存储在一起，会影响消息的写入和读取性能。因此对于消息，我们同采使用数字类型，但是如果数字是随机的，那么我们也无法利用存储引擎的特性让相邻的消息存储在一起，会加大随机 IO，降低消息的写入和读取性能，而且随机的 ID 也很难保证全局唯一性，因此我们一般都是使得消息 ID 递增的。

2.4.2 递增的三种级别

全局递增：指消息 ID 在整个 IM 系统随着时间的推移是递增的。全局递增的话一般可以使用 Snowflake（当然，Snowflake 也只是 worker 级别的递增）。此时，如果你的系统是读扩散的话为了防止消息丢失，那每一条消息就只能带上上一条消息的 ID，前端根据上一条消息判断是否有丢失消息，有消息丢失的话需要重新拉一次。

用户级别递增：指消息 ID 只保证在单个用户中是递增的，不同用户之间不影响并且可能重复。典型代表：微信。如果是写扩散系统的话信箱时间线 ID 跟消息 ID 需要分开设计，信箱时间线 ID 用户级别递增，消息 ID 全局递增。如果是读扩

散系统的话感觉使用用户级别递增必要性不是很大。

会话级别递增：指消息 ID 只保证在单个会话中是递增的，不同会话之间不影响并且可能重复。典型代表：QQ。

2.4.3 递增的两种方式

连续递增是指 ID 按 1,2,3...n 的方式生成；而单调递增是指只要保证后面生成的 ID 比前面生成的 ID 大就可以了，不需要连续。

据我所知，QQ 的消息 ID 就是在会话级别使用的连续递增，这样的好处是，如果丢失了消息，当下一条消息来的时候发现 ID 不连续就会去请求服务器，避免丢失消息。此时，可能有人会想，我不能用定时拉的方式看有没有消息丢失吗？当然不能，因为消息 ID 只在会话级别连续递增的话那如果一个人有上千个会话，那得拉多少次啊，服务器肯定是抗不住的。

对于读扩散来说，消息 ID 使用连续递增就是一种不错的方式了。如果使用单调递增的话当前消息需要带上前一条消息的 ID（即聊天消息组成一个链表），这样，才能判断消息是否丢失。

2.5 消息格式设计

在设计消息格式时要考虑到许多的点：

(1)网络数据大小，消息所要占用的带宽和传输效率：

虽然对单个用户来说，数据量传输很小，但是对于服务器端要承受众多的高并发数据传输（尤其现时高并发、大用户量的 IM 聊天应用和实时推送服务端等场景），必须要考虑到数据占用带宽，尽量不要有冗余数据，这样才能够少占用带宽，少占用资源，少网络 IO，提高传输效率。

(2)网络数据安全性：敏感数据的网络安全：

对于相关业务的部分数据传输都是敏感数据，所以必须考虑对部分传输数据进行加密。这通常出现在银行等数据安全性要求很高的应用行业和场景里，当然传统的即时通讯应用里基于用户隐私考虑，数据加密也是同样是个必须考虑的问题。安全性是应用的基础条件，需求是一样的，只是加密程度、安全性级别要求有不同而已。

(3)编码复杂度：

编码复杂度包括序列化和反序列化复杂度、效率、数据结构的可扩展性和可

维护性。

对于平台相关业务的代码实现也需要考虑到数据发送方和数据接收方数据处理的复杂度和数据结构的可扩展性，可维护性，人力成本和实施复杂度也必须考虑在内。通常情况下，即时通讯应用（比如 IM 聊天应用）在开发的前期，为了方便调试，很多团队会用简单的文本协议、JSON 等能直观查看的方式，但后期生产部署后，为了流量等考虑，可能会转用 Protobuf 等更省流量的协议。但总之，协议的定义不可能永远一成不变，但如果在实现的时候就有这些预见性，相性会大大减轻未来的运营风险。

(4)协议通用性：

数据类型必须是跨平台，数据格式是通用的，大家普遍能接受上手的。当然，现在已经迈入移动互联网时代，多端、多平台、异构平台的数据通讯是先决条件，而协议的选择，通用性也最多只是应用层有区别。当然，无论如何，异构平台的一致性，是毫无争议的必备条件。

第 3 章 即时通讯系统的优化方案

3.1 消息送达机制实现

3.1.1 消息投递过程

消息的可靠性是即时通讯系统的重要特性。

其中，IM 的客户端和服务端通过发送报文，也就是请求包来完成消息的传递。而报文又分为以下三种：

请求报文(request)：客户端主动发送给服务器的报文

应答报文(ack)：服务器被动应答客户端的报文

通知报文(notify)：服务器主动发送给客户端的报文

一般来说，消息的投递过程是这样的：(1)客户端 A 向服务端发送一个请求报文；(2)服务端在成功处理后，回应给客户端 A 一个响应包；(3)如果此时客户端 B 在线，那么服务端就会主动向客户端 B 发送一个消息通知包。

3.1.2 消息投递过程中易出现的问题

在客户端 A 收到响应包后，只能说明服务端成功接收到了消息，但并不能说明客户端 B 接收到了消息，如果在这个期间出现消息的丢失，那么客户端 A 是完全不知情的。

3.1.3 应用层确认机制

了解过 TCP 和 UDP 的人都知道，UDP 是一种不可靠协议，而 TCP 是一种可靠的传输层协议。TCP 做到可靠的方式就是通过超时、重传、确认。那么，如果想要做到应用层的消息可靠投递，我们就要类似地加入应用层的确认机制，要想让发送方 A 确认接收方 B 收到了消息，就必须让接收方 B 给出一个消息的确认，而这个消息的确认过程和消息的发送流程相类似：(1)客户端 B 给服务端发送一个响应包；(2)服务端在成功处理后回复给客户端 B 一个响应包；(3)服务端主动向客户端 A 发送一个通知包。这样，我们就完成了一个消息投递和消息确认的闭环流程，就能够保证消息未送达时客户端 A 能够成功地发现并重新发送消息了[1]。

3.2 保证消息有序性

3.2.1 消息乱序场景

消息的有序性是分布式 IM 系统中的一个重要的特性。因为在分布式系统中，客户端的始终和服务器的时钟很可能是不同步的，如果只是参考单方的时钟，就很可能会出现大量的消息乱序、错序[2]。

例如，如果我们只依赖客户端的时钟，A 比 B 时间晚 30min，A 给 B 发消息，然后 B 给 A 回复。

发送顺序是：

客户端 A：”XXX”

客户端 B：”YYY”

接收方的排序是：

客户端 B：”YYY”

客户端 A：”XXX”

因为 A 的时钟完了 30min，所以 A 的消息在客户端 B 会被排在后面。如果只依赖于服务器的时钟，也会出现类似的问题，因为两个服务器的时钟也可能不一致。

3.2.2 保证消息有序性机制

为了解决这种问题，我的思路是通过可以做这样一系列的操作来实现。

(1) 服务器时间对齐：

这部分就是后端运维的锅了，由系统管理员来尽量保障，没有别的招儿。

(2) 客户端通过时间调校对齐服务器时间：

比如：客户端登录以后，拿客户端时间和服务器时间做差值，发送消息的时候考虑这部分差值。

在我的 im 架构里，这个能把时间对齐到 100ms 这个级，差值再小的话就很困难了，因为协议在客户端和服务器之间传递速度 RTT 也是不稳定的[3]。

消息同时带上本地时间和服务器时间：

消息具体可以这样的处理：排序的时候，对于同一个人的消息，按照消息本地时间来排；对于不同人的消息，按照服务器时间来排，这是插值排序算法[4]。

3.3 群聊消息去重机制

为了防止消息丢失我们在应用层加入了重传和确认机制，但是这又会带来新的问题，那就是一条消息可能会被重复发送。

例如：假设客户端成功收到了服务端推送的消息，但其后续发送的确认包丢失了，那么服务端将会在超时后再次推送该消息，如果业务层不对重复消息进行处理，那么用户就会看到两条完全一样的消息[5]。

消息去重的方式其实非常简单，一般是根据消息的唯一标志(id)进行过滤。

具体过程在服务端和客户端可能有所不同：

(1) 客户端：我们可以通过构造一个 map 来维护已接收消息的 id，当收到 id 重复的消息时直接丢弃；

(2) 服务端 ：收到消息时根据 id 去数据库查询，若库中已存在则不进行处理，但仍然需要向客户端回复 Ack（因为这条消息很可能来自用户的手动重发）。

3.4 网络心跳包机制

3.4.1 为什么需要心跳包机制

(1) 问题一：一个客户端连接服务器以后，如果长期没有和服务器有数据来往，可能会被防火墙程序关闭连接，有时候我们并不想要被关闭连接。例如，对于一个即时通讯软件来说，如果服务器没有消息时，我们确实不会和服务器有任何数据交换，但是如果连接被关闭了，有新消息来时，我们再也没法收到了，这就违背了“即时通讯”的设计要求[6]。

(2) 问题二：通常情况下，服务器与某个客户端一般不是位于同一个网络，其之间可能经过数个路由器和交换机，如果其中某个必经路由器或者交换器出现了故障，并且一段时间内没有恢复，导致这之间的链路不再畅通，而此时服务器与客户端之间也没有数据进行交换，由于 TCP 连接是状态机，对于这种情况，无论是客户端或者服务器都无法感知与对方的连接是否正常，这类连接我们一般称之为“死链”。

3.4.2 心跳包如何解决问题

(1) 心跳包解决问题一：此应用场景要求必须保持客户端与服务器之间的连接正常，就是我们通常所说的“保活“。如上所述，当服务器与客户端一定时间内没有有效业务数据来往时，我们只需要给对端发送心跳包即可实现保活。

(2) 心跳包解决问题二：要解决死链问题，只要我们此时任意一端给对端发送一个数据包即可检测链路是否正常，这类数据包我们也称之为”心跳包”，这种操作我们称之为“心跳检测”。顾名思义，如果一个人没有心跳了，可能已经死亡了；一个连接长时间没有正常数据来往，也没有心跳包来往，就可以认为这个连接已经不存在，为了节约服务器连接资源，我们可以通过关闭 socket，回收连接资源[7]。

3.4.3 应用层的心跳包机制设计

由于 keepalive 选项需要为每个连接中的 socket 开启，这不一定是必须的，可能会产生大量无意义的带宽浪费，且 keepalive 选项不能与应用层很好地交互，因此一般实际的服务开发中，还是建议读者在应用层设计自己的心跳包机制。

从技术来讲：心跳包其实就是一个预先规定好格式的数据包，在程序中启动一个定时器，定时发送即可，这是最简单的实现思路。

但是，如果通信的两端有频繁的数据来往，此时到了下一个发心跳包的时间点了，此时发送一个心跳包。这其实是一个流量的浪费，既然通信双方不断有正常的业务数据包来往，这些数据包本身就可以起到保活作用，为什么还要浪费流量去发送这些心跳包呢？

所以，对于用于保活的心跳包，我们最佳做法是：设置一个上次包时间，每次收数据和发数据时，都更新一下这个包时间，而心跳检测计时器每次检测时，将这个包时间与当前系统时间做一个对比，如果时间间隔大于允许的最大时间间隔（实际开发中根据需求设置成 15 ~ 45 秒不等），则发送一次心跳包。总而言之，就是在与对端之间，没有数据来往达到一定时间间隔时才发送一次心跳包[8]。

3.5 保证 IM 系统的数据安全

3.5.1 IM 系统的数据安全简介

IM 系统架构中的数据安全比一般系统要复杂一些，从通信的角度来说，它涉及到 socket 长连接通信的安全性和 http 短连接的两重安全性。而随着 IM 在移动端的流行，又要在安全性、性能、数据流量、用户体验这几个维度上做权衡，所以想要实现一套完善的 IM 安全架构，要面临的挑战是很多的。

IM 系统架构中，所谓的数据安全，主要是通信安全和内容安全。

3.5.2 保证通信安全

所谓的通信安全，这就要理解 IM 通信的服务组成。目前来说，一个典型的 im 系统，主要由两种通信服务组成：

(1) socket 长连接服务：技术上也就是多数人耳熟能详的网络通信这一块，再细化一点也就是 tcp、udp 协议这一块；

(2) http 短连接服务：也就是最常用的 http rest 接口那些[9]。

3.5.3 保证内容安全

密码学三大作用：加密（ Encryption）、认证（Authentication），鉴定（Identification） 。

详细来说就是：

(1) 加密：防止坏人获取你的数据。

(2) 认证：防止坏人修改了你的数据而你却并没有发现。

(3) 鉴权：防止坏人假冒你的身份。

恶意攻击者如果在通信环节绕开或突破了“鉴权”、“认证”，那么依赖于“鉴权”、“认证”的“加密”，实际上也有可有被破解。

针对上述问题，那么我们需要对内容进行更加安全独立的加密处理，就这是所谓的“端到端加密”（E2E）。

比如，那个号称无法被破解的 IM——Telegram，实际上就是使用了端到端加密技术[10]。

3.6 网络短连接的优化方案

通常我们开发一个移动端应用，会直接调用系统提供的网络请求接口去服务端请求数据，再针对返回的数据进行一些处理，或者使用 iOS 中的开源 AFNetworking/OKHttp 这样的网络库(Android 中可以用 HttpURLConnection 或者开源的 okhttp 库)，管理好请求线程和队列，再自动做一些数据解析，就结束了[11]。

但对于追求用户体验的应用来说，还会针对移动网络的特性做进一步优化，包括：

(1)速度优化：网络请求的速度怎样能进一步提升？

(2)弱网适应：移动端网络环境随时变化，经常出现网络连接很不稳定可用性差的情况，怎样在这种情况下最大限度最快地成功请求？

(3)安全保障：怎样防止被第三方窃听/篡改或冒充，防止运营商劫持，同时又不影响性能？

3.6.1 速度请求的优化

正常一条网络请求需要经过的流程是这样：

(1)DNS 解析，请求 DNS 服务器，获取域名对应的 IP 地址；

(2)与服务端建立连接，包括 tcp 三次握手，安全协议同步流程；

(3)连接建立完成，发送和接收数据，解码数据。

这里有明显的三个优化点：

(1)直接使用 IP 地址，去除 DNS 解析步骤；

(2)不要每次请求都重新建立连接，复用连接或一直使用同一条连接(长连接)；

(3)压缩数据，减小传输的数据大小。

3.6.2 DNS 优化

DNS 完整的解析流程很长，会先从本地系统缓存取，若没有就到最近的 DNS 服务器取，若没有再到主域名服务器取，每一层都有缓存，但为了域名解析的实时性，每一层缓存都有过期时间。上面这种 DNS 解析机制有几个缺点[12]：

(1)缓存时间设置得长，域名更新不及时，设置得短，大量 DNS 解析请求影响请求速度；

(2)域名劫持，容易被中间人攻击，或被运营商劫持，把域名解析到第三方 IP 地址，据统计劫持率会达到 7%；

(3)DNS 解析过程不受控制，无法保证解析到最快的 IP；

(4)一次请求只能解析一个域名。

为了解决这些问题，就有了 HTTPDNS，原理很简单，就是自己做域名解析的工作，通过 HTTP 请求后台去拿到域名对应的 IP 地址，直接解决上述所有问题。自己实现 HTTPDNS 的好处总结就是：

(1)域名解析与请求分离，所有请求都直接用 IP 地址，无需 DNS 解析，APP 定时请求 HTTPDNS 服务器更新 IP 地址即可；

(2)通过签名等方式，保证 HTTPDNS 请求的安全，避免被劫持；

(3)DNS 解析由自己控制，可以确保根据用户所在地返回就近的 IP 地址，或根据客户端测速结果使用速度最快的 IP[13]；

(4)一次请求可以解析多个域名。

第 4 章 即时通讯系统功能设计与实现

本章节主要目的是对系统的功能进行设计与实现的过程，主要是从系统功能的实现的流程和系统功能模块的界面的实现开始展开，探讨的主要是即时通讯系统的大致功能设计和实现，包含注册模块、登录模块、个人信息模块、通讯录模块、聊天模块和发现模块的设计和实现。

4.1 关键技术介绍

即时通讯系统主要包含客户端和服务端，其中客户端使用 vue+socketio-client 实现，服务端使用 SpringBoot+socketIo 实现，数据库使用 mysql。

4.1.1 什么是实时通信

“实时通信”指的是：

(1)客户端可以随时主动地将数据发送给服务端；(2)当客户端关注的内容在发生改变时，服务器可以实时地通知客户端。比较于传统的 C/S 请求模型，“实时通信”时客户端不需要主观地发送请求去获取自己关心的内容，而是由服务器端将消息”推送”给客户端。

注意：上面的“推送”二字打了引号，实际上现有的几种技术实现方式中，并不是服务器端真正主动地推送，实际上，它们只是通过一定的技术手段制造了一种“实时通信”的虚幻假象。

就目前现有的几种技术而言，主要有以下几类：

(1)客户端轮询：传统意义上的短轮询（Short Polling）；

(2)服务器端轮询：长轮询（Long Polling）；

(3)单向服务器推送：Server-Sent Events（SSE）；

(4)全双工通信：WebSocket。

4.1.2 websocket 和 socket.io 区别

websocket

(1) 一种让客户端和服务器之间能进行双向实时通信的技术

(2) 使用时，虽然主流浏览器都已经支持，但仍然可能有不兼容的情况

(3) 适合用于 client 和基于 node 搭建的服务端使用

socket.io

(1) 将 WebSocket、AJAX 和其它的通信方式全部封装成了统一的通信接口

(2) 使用时，不用担心兼容问题，底层会自动选用最佳的通信方式

(3) 适合进行服务端和客户端双向数据通信

4.1.3 socketIo 特点

(1)实时分析：将数据推送到客户端，这些客户端会被表示为实时计数器，图标或者是日志客户；

(2)实时通信和聊天：只需几行代码便可写成一个 Socket.IO 的简单聊天应用；

(3)二进制流传输：从 1.0 版本开始，Socket.IO 支持任何形式的二进制文件传输，例如：图片，视频，音频等；

(4)文档合并：允许多个用户同时编辑一个文档，并能看到每个用户做出的修改

4.2 注册模块

如图 4-1 所示，点击注册即可注册即时通讯系统的账号，注册完毕后会直接跳转到系统中的主界面。

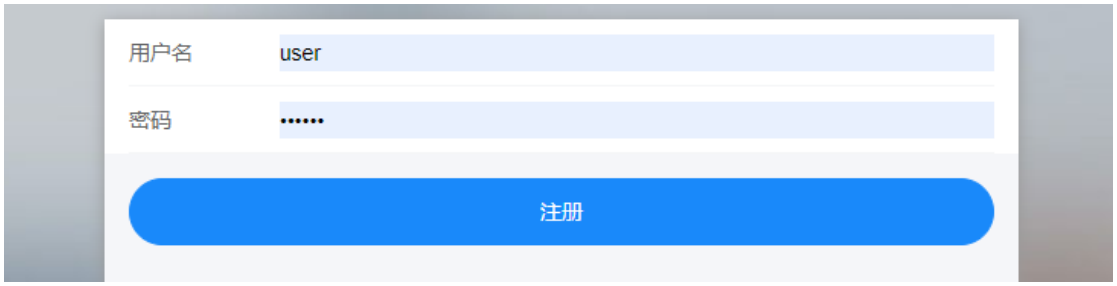

图 4-1 注册模块

4.3 登录模块

用户的登录是即时通讯系统的基本功能。用户登录的流程分为验证用户的身份信息、获取上次会话信息、完成消息的主题订阅，向好友广播通知自己的上线状态通知。

(1)在客户端上填写用户名和密码，然后请求登录；

(2)服务器端会调用相应的服务验证用户身份和密码正确性,验证通过后登录

(3)服务端会保存用户的基本信息到 session 中,同时客户端也会存储用户信息，在访问其他页面时会检查用户信息状态

(4)服务端向客户端返回信息，其中信息包括用户的基本信息等

(5)客户端会跳转到主页面，并向服务端请求获取需要初始化的数据，包括最近的会话列表，消息推送服务器的地址

如图 4-2 所示，在注册界面成功注册后可以在登录界面中输入相应的账号和密码登录。

图 4-2 登录模块

4.4 个人信息模块

在下方的 TAB 栏中可以切换到“我的”一栏，然后便可看到上述界面，其中，个人信息为昵称，签名。

如图 4-3、图 4-4、图 4-5、图 4-6、图 4-7 和图 4-8 所示，可以点击进入对个人信息进行修改。

图 4-3 个人信息模块

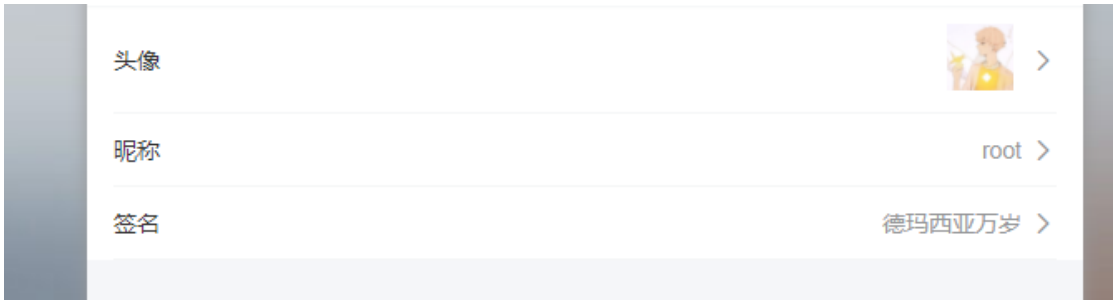

图 4-4 个人信息更改项

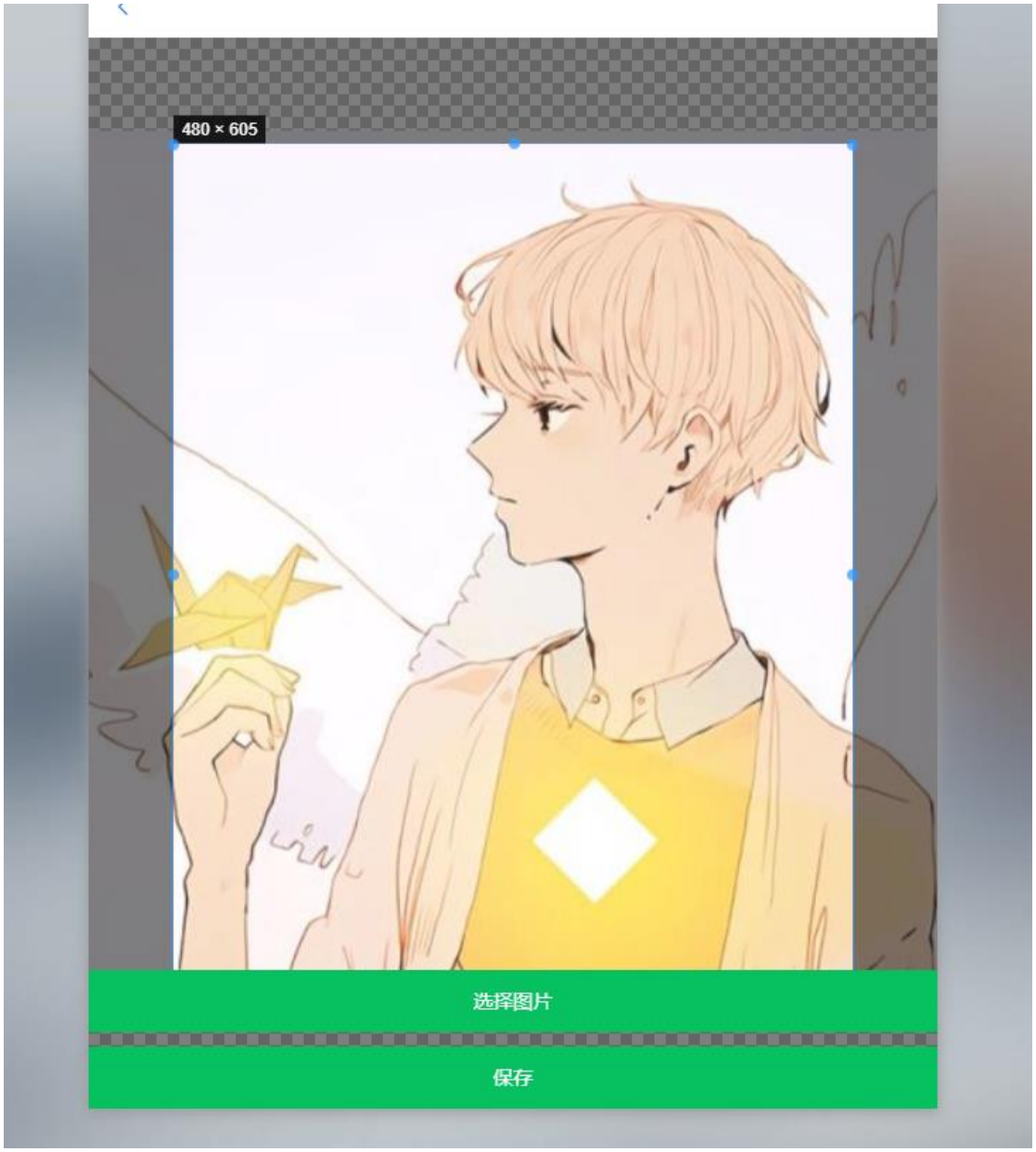

图 4-5 选择头像

图 4-6 修改昵称

图 4-7 修改签名

图 4-8 注销

4.5 通讯录模块

如图 4-9 所示，进入通讯录，可以看到好友列表和群组，可以通过搜索功能快速定位到想要查找的好友，可以在新的申请中查看好友申请和群组申请。

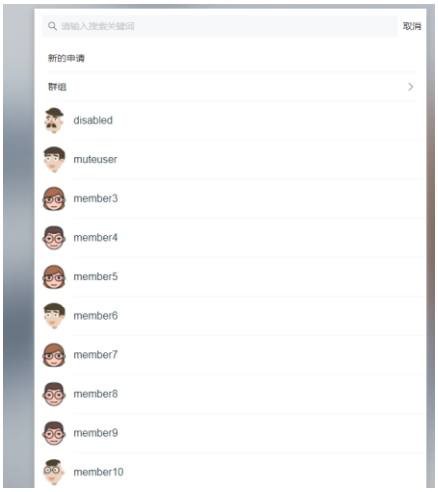

图 4-9 通讯录模块

4.6 聊天模块

如图 4-10 所示，点击会话或者联系人可以进入到相应的聊天界面，可以在其中查看到历史聊天数据，也可以发送最新的消息给目标用户，其中消息类型可以为文字、文件、图片、表情等多个类型。

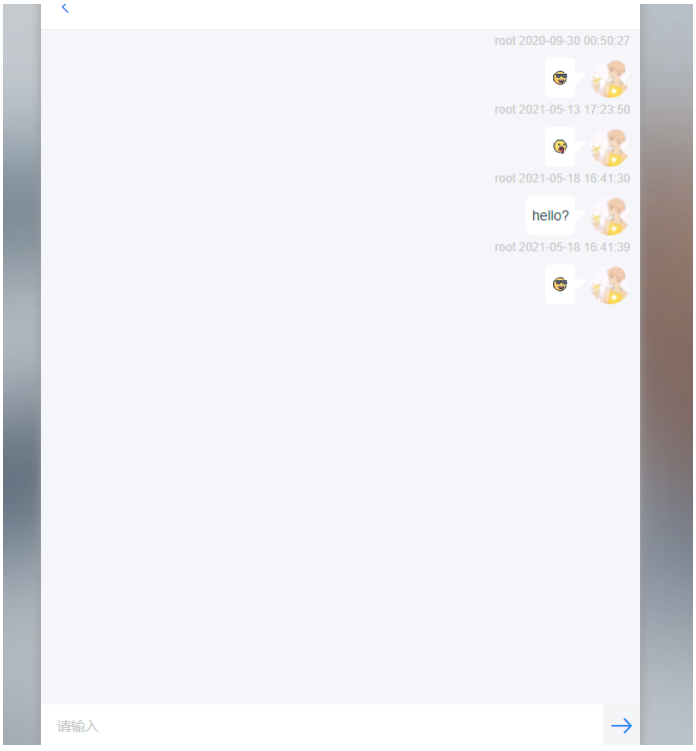

图 4-10 聊天框

第 5 章 系统测试与运行

系统测试与运行的章节是通过编写相应的测试用例和测试工具来进行手工测试和自动化的测试，确保能够相对全面地达到我们预期地测试目的，有效地去验证即时通讯系统有没有满足用户地需求或者在需求分析和详细设计阶段达到的目标，比较出目标系统和当前系统存在的差距，方便我们去根据这些差距有效调整当前系统，最终完成一个用户预期的目标系统的构建。

5.1 测试方案

在测试的原则上我们要求测试方法涉及设计上所有的模块以及运行编写的所有代码模块所实现的功能并和最初的需求以及设计方案进行对比，修改所发现的程序错误和运行中的不期望出现的因素并最大程度的向需求和设计靠拢。

5.2 测试项目说明

5.2.1 登录部分测试

系统需求：安装 JVM 并且支持 java8 的操作系统，登录测试表如表 5-1 所示：

表 5-1 登录测试表

	输入示例	预计输出
Id 不存在	ID：kdnhqw PSW：weqjnkqwe	界面提示显示错误信息
Id 存在但密码不正确	ID：kdnhqw PSW：wdcswhwqd	界面提示显示错误信息
Id 不存在但密码被使用	ID：kdnhqw PSW：abcdfe	界面提示显示错误信息
Id 与密码均正确	ID：kdnhqw PSW：wdhdcsjj	进入主界面
点击注册链接	单击”注册界面”	打开注册界面

5.2.2 注册模块测试

系统需求：安装 JVM 并且支持 java8 的操作系统。注册测试表如图 5-2 所示：

表 5-2 注册测试表

	输入示例	预计输出
注册已存在的 ID	ID：abcdwed	提示用户已存在
两次密码输入不一致	PSW1：wgebdhnj PSW2：fghwbjnd	提示两次密码输入不一致
有未填写项	ID：空白或 PSW 空白	提示有未输入项

5.2.3 好友管理模块

系统需求：安装 JVM 并且支持 java8 的操作系统。好友管理表如表 5-3 所示：

表 5-3 好友管理表

		输入示例	预计输出
添加好友	对方不在好友列表	输入对方 id,点击添加	提示添加成功，并返回好友列表
	对方在好友列表	输入对方 id,点击添加	提示好友存在
	对方 ID 不存在	输入一个不存在的 ID	提示用户不存在
删除好友		选中删除的好友，点击删除	提示删除并从数据库中以及好友列表中删除

5.2.4 私聊模块

系统需求：安装 JVM 并且支持 java8 的操作系统。私聊测试表如表 5-4 所示：

表 5-4 私聊测试表

		输入示例	预计输出
开启私聊		选中私聊好友后点击私聊按钮	双方建立私聊页面
	建立超过限制数目的私聊窗口		提示窗口过多
	发送信息	各个私聊界面分别发送信息	信息在相应的界面显示
关闭私聊		点击关闭按钮，关闭当前会话页面	关闭私聊窗口
	中途一方下线		关闭相应窗口

5.3 测试结果说明

以上测试用例均通过，说明即时通讯系统各个功能基本正确。性能测试指标有待在实际运行环境中进一步确定。

第 6 章 结束语

本文结合了许多市面上成熟的 IM 解决方案，深入地去探讨这些方案的可行性以及各自的优缺点，在实际的业务场景对这些方案的解决逻辑进行分析，并根据这些方案自己完成了一套可用的即时通讯系统的设计和架构，并根据相应功能点实现了即时通讯软件。本文主要完成了以下几个方面的内容：

考察调研了当前国内的即时通讯软件的市场背景和发展现状，在此基础上分析出一款更优的即时通讯系统的必要性；

分析了即时通讯系统的研究意义，提出一款高可用的即时通讯系统必须满足实时性、可靠性、一致性和安全性；提出了该即时通讯系统的架构方案、数据库设计方案以及系统功能的设计；针对于即时通讯系统中常见的问题进行讨论，并提出相应的解决方案；在系统功能设计方面，针对每一块功能，给出相应的设计与实现方案。对每一块功能进行测试，并给出相应的测试结果，测试结果明确地验证了系统能够合理地运行，完成了制定的各个指标。

致谢

在学位论文即将完成之时，有非常多的人想要感谢。万分感谢在这期间陪伴我经历过这些事的人，万分感谢指导老师的耐心指导和莫大的鼓励，万分感谢这次论文给了我充实自己的机会，万分感谢伴我走过这一切的老师、同学和朋友。最后，衷心地感谢百忙中抽出时间审阅我的论文的各位专家评委们。

参考文献

[1] 叶为正,林声肯,黄立轩,许志明,李晶.即时通讯系统的设计与实现[J].计算机技术与发展,2020,30(02):216-220.

[2] 余春贵. 企业级即时通讯系统设计与实现[D].华南理工大学,2018.

[3] 陈亮. 即时通讯系统的设计与实现[D].南京邮电大学,2017.

[4] 车永光.Web 的即时通讯系统设计研究[J].信息通信,2018(06):178-180.

[5] 王真. 移动互联网即时通讯系统密钥管理技术研究及应用[D].北京邮电大学,2020.

[6] 宋希香.第三代基于 Java 的校园即时通讯工具的设计与实现[J].科技视界,2020(18):133-135.

[7] 李勇.基于麒麟系统的即时通讯系统设计与实现[J].自动化技术与应用,2020,39(03):51-55.

[8] 张昭理,高俊茹,孙建文.在线学习系统中的即时通讯工具应用研究[J].中国教育信息化,2019(17):13-19.

[9] 杨志永. 军工企业即时通讯安全体系研究[D].电子科技大学,2020.

[10] 杨熙乾.计算机电子信息技术在即时通讯上的应用[J].数字技术与应用,2018,36(09):15-16+18.

[11] 吴迪.一种在线多人多事务即时通讯系统的设计与实现[J].太原师范学院学报(自然科学版),2018,17(02):55-58.

[12] Kim Giyoon et al. Forensic analysis of instant messaging apps: Decrypting Wickr and private text messaging data[J]. Forensic Science International: Digital Investigation, 2021, 37

[13] Ángel Manuel Guerrero-Higueras et al. Facilitating the learning process in parallel computing by using instant messaging[J]. The Journal of Supercomputing, 2021, 77(4) : 3899-3913.