南京邮电大学

毕 业 设 计（论 文）

|  |  |
| --- | --- |
| 题 目 | 移动互联网智能工具云平台设计与实现 |
| 专 业 | 软件工程（嵌入式培养，NIIT） |
| 学生姓名 | 陈兵 |
| 班级学号 | B140411 B14041126 |
| 指导教师 | 李超 |
| 指导单位 | 计算机、软件学院、网络空间安全学院 |

日期：2017年12月15日至2018年6月8日

毕业设计（论文）原创性声明

本人郑重声明：所提交的毕业设计（论文），是本人在导师指导下，独立进行研究工作所取得的成果。除文中已注明引用的内容外，本毕业设计（论文）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本研究做出过重要贡献的个人和集体，均已在文中以明确方式标明并表示了谢意。

论文作者签名：陈兵

日期：2018年5月31日

摘 要

随着“中国智造”和“工业4.0”发展战略的提出，如何对传统工具进行改造，提升制造业智能化水平成为重要的课题。

本文针对传统工具改造，提出移动互联网智能工具系统这一概念，并着眼于移动互联网智能工具系统中的服务端，设计和实现了一个可兼容和接入多样化设备的移动互联网智能工具云平台。移动互联网智能工具云平台包含：后台管理系统、App HTTP服务端、App Socket服务端三个模块。后台管理系统基于ThinkPHP框架，采用MVC模式实现。App HTTP服务端使用PHP实现。App Socket服务端基于Mina框架，通过自定义协议实现。

最后本文进行了云平台的部署和测试，通过分析日志，验证了本文设计和实现的云平台具有可靠性、健壮性和可扩展性。

目前，本系统已经被公司采用，投入到市场中，提供商业化服务。

**关键词：**移动互联网；智能工具；云平台；Socket；IO模型；MINA框架；

**ABSTRACT**

With the development strategy of "China's Intellectual Creation" and "Industry 4.0", how to transform traditional tools and upgrade the level of intelligent manufacturing has become an important issue.

This paper aims at the transformation of traditional tools, puts forward the concept of mobile internet intelligent tool system, and focuses on the service end in the mobile internet smart tool system. It designs and implements a mobile internet smart tool cloud platform that can be compatible and access diverse devices. Mobile Internet smart tool cloud platform includes three modules: background management system, App HTTP server, and App Socket server. The back-end management system is based on the ThinkPHP framework, implemented using the MVC pattern. The App HTTP server is implemented using PHP. The App Socket server is based on the Mina framework and is implemented through a custom protocol.

In the end, the paper deploys and tests the cloud platform, and analyzes the logs to verify that the cloud platform designed and implemented in this paper is reliable, robust, and extensible.

At present, this system has been adopted by the company and put into the market to provide commercial services.

**Key words:** Mobile Internet; Smart tools; Cloud platform; Socket; IO model;

MINA framework

目 录

[第一章 绪论 1](#_Toc515566781)

[1.1 研究背景 1](#_Toc515566782)

[1.2 研究内容 2](#_Toc515566783)

[1.3 研究意义 3](#_Toc515566784)

[第二章 相关技术介绍 4](#_Toc515566785)

[2.1 ThinkPHP框架 4](#_Toc515566786)

[2.2 IO模型和MINA框架 4](#_Toc515566787)

[2.2.1 高并发IO模型 5](#_Toc515566788)

[2.2.2 基于Java NIO的MINA框架 7](#_Toc515566789)

[第三章 移动互联网智能工具云平台设计 10](#_Toc515566790)

[3.1 系统框架 10](#_Toc515566791)

[3.2 CLOUD数据库设计 11](#_Toc515566792)

[3.3 后台管理系统设计 17](#_Toc515566793)

[3.3.1 整体架构 17](#_Toc515566794)

[3.3.2 系统功能结构设计 18](#_Toc515566795)

[3.4 App HTTP服务端设计 20](#_Toc515566796)

[3.4.1 系统技术架构 20](#_Toc515566797)

[3.4.2 用户接口设计 21](#_Toc515566798)

[3.5 App Socket服务端设计 23](#_Toc515566799)

[3.5.1 通信模块的设计 23](#_Toc515566800)

[3.5.2 设备注册模块的设计 24](#_Toc515566801)

[3.5.3 消息处理模块的设计 25](#_Toc515566802)

[3.5.4 其他模块设计 27](#_Toc515566803)

[第四章 移动互联网智能工具云平台实现 29](#_Toc515566804)

[4.1 后台管理系统实现 29](#_Toc515566805)

[4.1.1 系统登录模块实现 29](#_Toc515566806)

[4.1.2 产品信息管理模块实现 30](#_Toc515566807)

[4.1.3 更新反馈管理模块实现 30](#_Toc515566808)

[4.2 App HTTP服务端实现 32](#_Toc515566809)

[4.2.1 注册模块的实现 32](#_Toc515566810)

[4.2.2 登录模块的实现 33](#_Toc515566811)

[4.2.3 设备绑定模块的实现 33](#_Toc515566812)

[4.3 App Socket服务端实现 34](#_Toc515566813)

[4.3.1 通信模块的实现 34](#_Toc515566814)

[4.3.2 设备注册模块的设计 36](#_Toc515566815)

[4.3.3 消息处理模块 38](#_Toc515566816)

[4.3.4 其他模块实现 40](#_Toc515566817)

[第五章 系统测试 42](#_Toc515566818)

[5.1 功能测试 42](#_Toc515566819)

[5.1.1 测试环境搭建 42](#_Toc515566820)

[5.1.2 测试方案 44](#_Toc515566821)

[5.1.3 测试结果及分析 44](#_Toc515566822)

[结束语 47](#_Toc515566823)

[致 谢 48](#_Toc515566824)

[参考文献 49](#_Toc515566825)

# 第一章 绪论

## 研究背景

随着“中国智造”和“工业4.0”发展战略的提出，如何对传统工具进行改造，使其成为智能工具，提升制造业智能化水平成为重要的课题。

所谓智能工具，指包含了传感器+微处理器+数据通信模块的传感工具，其中传感器负责环境信息收集，微处理器进行信息记忆和处理，传输器进行数据传输，将信息发送到网络。智能传感器相比普通传感器最大的区别在于智能化，智能传感器能把传感器检测信的号和处理器的信号处理功能结合在一起，并把数据实时发送到网络，从而智能化。

在传统PC互联网时代。智能工具大多还是以直接集成传感器+微处理器+数据通信模块这几个功能模块实现。但这带来了以下几个问题：

1. 物料成本高。智能工具的基础是传感器，但是为了实现智能化，往往需要集成微处理器和数据传输模块等其他模块，这大幅度提升了智能工具的物料成本。
2. 开发成本高。从硬件角度，智能工具的微处理器和传输器要与传感器配套开发，硬件的开发成本大幅度提高。从软件角度，为了使传感器、微处理器、传输器之间能够相互通信、相互协作，需要开发配套的软件系统，由于嵌入式硬件系统开发面向底层，开发难度大，这大幅度提高了软件的开发成本。
3. 移动性低。传统的智能工具往往通过固定网络传输数据，比如有限连接，或特定区域的无线连接，这限制了智能工具的应用场景，大幅度降低了智能工具的移动性。
4. 维护代价大。智能工具各模块的集成化，导致了每个模块可维护性降低，可能一个模块的损坏就导致整个工具报废。

基于以上问题，我们必须重新考虑智能工具的设计问题。如今移动互联网的迅猛发展，为我们带了新思路和新方案。移动终端完全可以充当智能工具的大脑和传输媒介，以取代以往智能工具中所必须集成的微处理器和数据传输模块，这使得智能工具的物料成本降低，可维护性提高，同时移动应用的开发往往比嵌入式开发更加容易，这使得软件开发成本和开发周期大幅度降低。

## 研究内容

为了响应“中国智造”和“工业4.0”发展战略，解决智能工具的成本高、移动性低、维护代价大等问题，本文提出移动互联网智能工具系统这一概念。所谓移动互联网智能工具系统，就是由传感设备、移动终端、云平台三部分组成的物联网系统。其中传感设备集成了传感器和数据传输模块，但此传输器只负责与移动终端进行通信，与PC互联网时代智能工具中集成的数据传输模块相比，功能更加简单，实现更加容易，成本相对也较低。

本文只简要介绍整个系统架构，而着重对移动互联网智能工具系统中的云平台进行研究、设计、实现和测试：

1. 移动互联网智能工具云平台往往要承载大量的设备连接和访问，因此如何实现高并发的服务器是一个重要的课题，而高并发服务器的实现经常需要构造高性能的IO模型，所以本文首先研究了同步非阻塞IO[2] 、IO多路复用[3] 等模型，并在此基础上研究了Java NIO[4] 和基于Java NIO的MINA网络通讯框架。
2. 移动互联网智能工具云平台需要为移动终端提供访问接口来实现用户注册登录、好友功能、设备登记绑定等功能。从移动终端和云平台软件开发的角度，通过HTTP[5] 协议提供这类服务是最常见和最便捷的方式。所以需要设计App HTTP服务端。
3. 此外需要考虑数据透传、设备的远程控制问题，由于HTTP是无状态协议，且在HTTP1.0协议中是不面向连接的，虽然在后续的HTTP1.1和HTTP2.0得到了改造，使其面向连接，但相对于直接利用TCP[6] 协议，HTTP协议的代价仍旧过高。考虑到终端数量及数据量，TCP协议是明智之选。因此需要设计App Socket服务端。App Socket服务端的设计包括通信协议的设计和程序设计。
4. 设备相关部分信息需要有专门的平台录入，因此需要设计一个后台管理系统，以提供用户、设备等相关信息的管理功能。
5. 移动互联网智能工具云平台三大模块的实现。其中最核心的是App Socket服务端的实现，涉及到TCP粘包断包、消息透传、消息群发、单备登录持等诸多具体实现问题。此外还需要考虑如何实现各模块间的协作。
6. 最后搭建测试环境，设计测试方案，对移动互联网智能工具云平台进行功能测试和验证。

## 研究意义

本文响应了“中国智造”和“工业4.0”发展战略，从移动互联网为切入点，利用移动终端作为传感设备的“大脑”和传输媒介来取代微处理器和数据通信模块这两个模块，从而降低智能工具成本和维护代价，提高智能工具的可移动性，使其不再因网络而限制其使用场景。

本文分析了移动互联网智能工具的使用场景和逻辑，考虑了多样性设备所要求的可扩展性，和设备数量带来的并发要求，从而设计实现了一套完整的云平台方案，这是整个移动互联网智能工具系统中必不可少的关键一环。

# 第二章 相关技术介绍

## ThinkPHP框架

ThinkPHP是一个简单、快速、兼容的PHP开发框架。如图 2.1所示，ThinkPHP包含了核心层、扩展层和应用层，核心层的核心类库提供了底层架构、兼容处理、基类库、模板引擎、缓存机制等常用组件，扩展层提供数据库访问，以及其他运行时资源的封装。应用层TinkPHP提供了基于MVC模型目录结构及相关配置文件，具体的实现则由开发者完成。

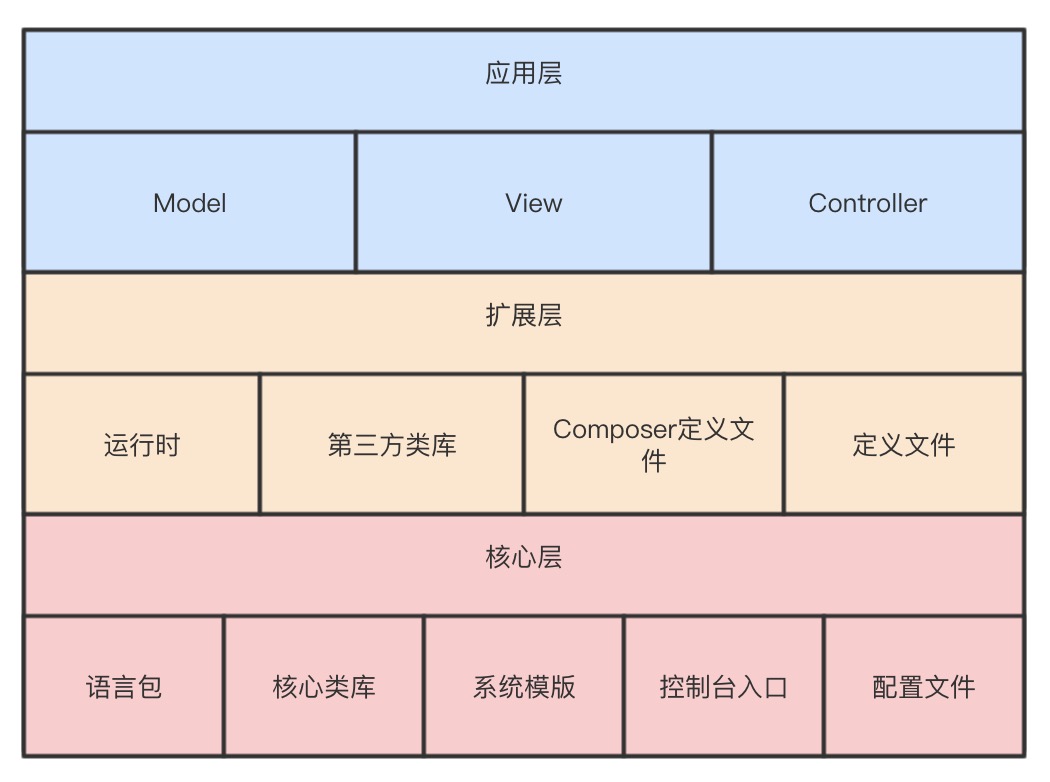


图 2.1 ThinkPHP框架架构图

## IO模型和MINA框架

IO模型主要有五种：同步IO和异步IO，阻塞IO和非阻塞IO，以及IO多路复用，本节主要介绍同步非阻塞IO和IO多路复用这两种高并发IO模型。

网络编程模型主要涉及accept、 read、write等操作或方法。

### 高并发IO模型

1. 同步非阻塞IO模型

传统的同步阻塞IO（即BIO）编程模型中，accept、reade、write方法都是同步阻塞的。以read方法为例，传统的BIO编程模型中，如果socket的接收缓冲区里没有数据，read方法会一直阻塞，直到收到数据，返回读到的数据。

阻塞机制导致了服务器无法并发处理多个socket连接，普遍的做法是使用多线程[7] 技术来处理多个socket连接。但过度的依赖于线程有以下缺点：

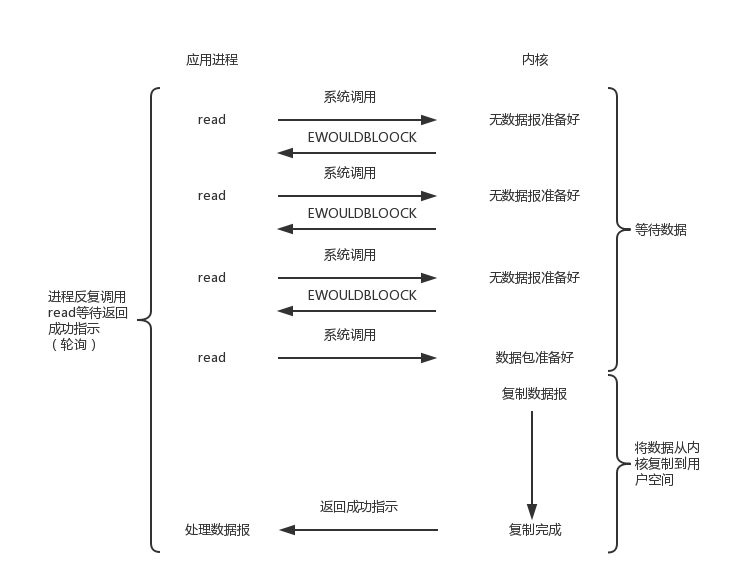
* 1. 线程的创建和销毁代价很高。
  2. 线程本身比较占用内存，特别对于Java编程，内存资源的考虑尤为重要。
  3. 线程的切换成本很高。

为了克服以上缺陷，同步非阻塞IO和IO多路复用模型应运而生。

同步是指内核态[7] 中，数据在设备空间（网络、磁盘等）和内核空间之间的交换是同步的。

非阻塞IO，在用户进程中调用IO操作，无论成功与否，该IO操作会立即返回，之后进程可以进行其他操作，即如果socket的接收缓冲区有数据，就把数据从网卡读到内存，并且返回给用户；反之则直接返回无数据信息，永远不会阻塞。所以数据是否读取成功将交给调用者判断。图 2.2展示了非阻塞IO模型是如何读取数据的。

图 2.2非阻塞IO模型



1. IO多路复用模型

在多路复用IO模型中，会有一个线程不断去轮询多个socket的状态，只有当socket真正有读写事件时，才真正调用实际的IO读写操作。因为在多路复用IO模型中，只需要使用一个线程就可以管理多个socket，系统不需要建立新的进程或者线程，也不必维护这些线程和进程，并且仅当真正有socket读写事件进行时，才会使用IO资源，所以它大大减少了资源占用。其工作流程如图 2.3所示。

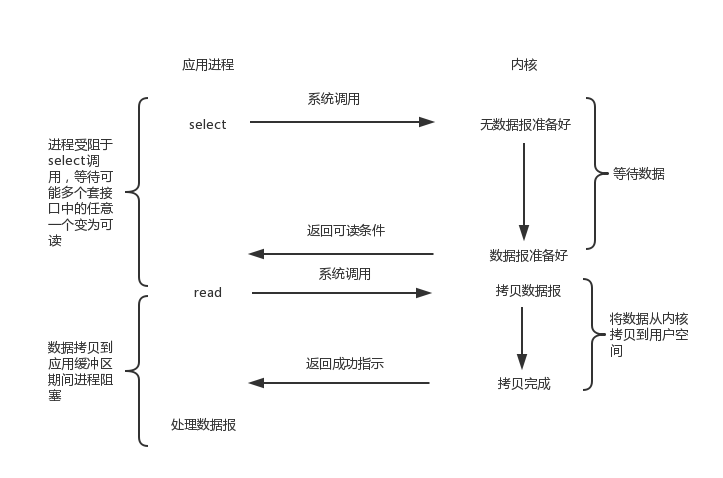


图 2.3 IO多路复用模型图

### 基于Java NIO的MINA框架

1. Java NIO概述

IO多路复用往往搭配非阻塞IO使用。这是因为IO多路复用模型中select只通知数据是否可读，而不会告诉可读的数据量，且无法保证数据一定可读，因为网络环境中有各种不确定因素导致数据的丢失，如果使用阻塞IO模型，在上述情况下，很难保证整个线程不被堵塞，即使可以实现，效率也大大折扣，而IO多路复用的核心就是单个线程通过记录跟踪每一个socket的状态来同时管理多个I/O流。

Java NIO正是IO多路复用模型和非阻塞IO模型结合的产物。Java NIO全称Java Non-blocking IO，由Channel、Buffer、Selector几个核心部分组成，具有以下特性：

1. 为所有的原始类型提供缓存（Buffer）支持；
2. 提供字符集编码解决方案；
3. 提供一个新的原始IO抽象：Channel；
4. 支持锁和内存映射文件的文件访问接口；
5. 提供多路非阻塞式的高伸缩性网络IO 。

其工作模式如图 2.4所示。

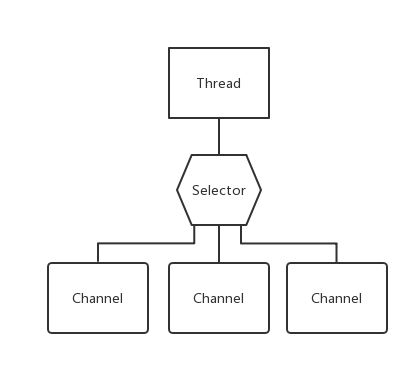


图 2.4 Selector工作模式

1. MINA框架概述

MINA是一个网络通信应用框架， MINA 可以帮助我们快速开发高性能、高扩展性的网络通信应用，它通过Java NIO技术基于TCP/IP和UDP/IP协议提供了抽象的、事件驱动的、异步的API。值得注意的是MINA框架中所提的异步并非异步IO，而是指MINA框架提供异步的API。

其整体架构如图 2.5所示，其中IoService执行实际的IO操作。IoFilter Chain,将数据进行过滤或者转换为期望的数据结构，反之亦然。IoHandler执行实际的业务。

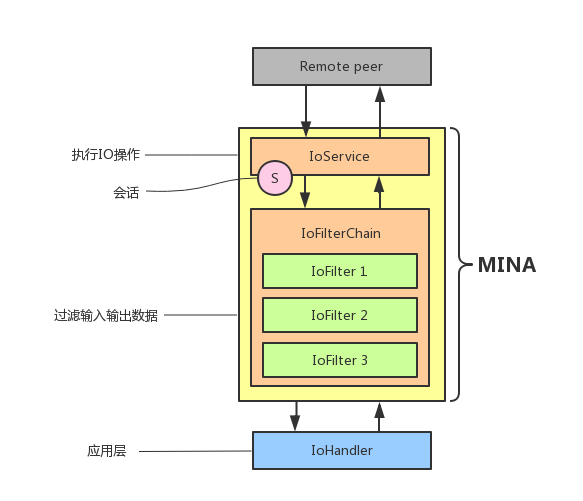


图 2.5 MINA框架架构

如图 2.6所示，MINA框架服务器端工作流程可描述为：

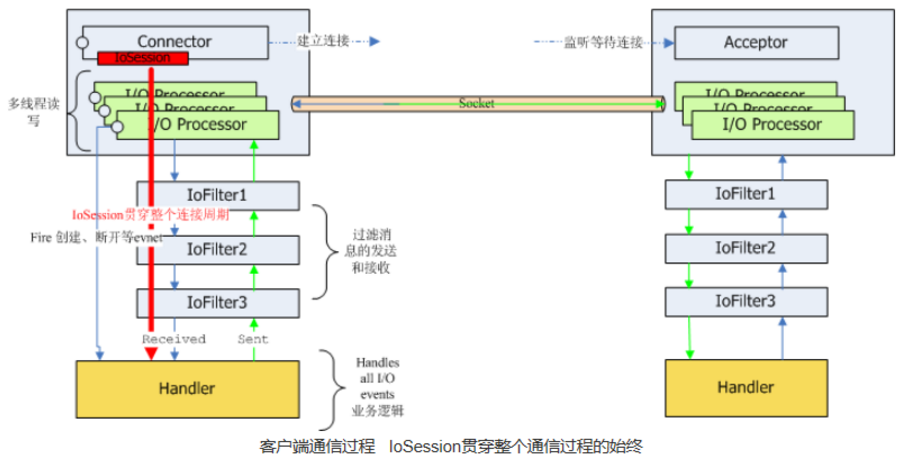


图 2.6 MINA框架工作流程

1. 通过SocketAcceptor与客户端建立socket连接；
2. 连接建立之后 将IO的读写交给了IO Processor线程，IO Processor是多线程的；
3. 通过IO Processor 读取的数据经过IoFilterChain里所有配置的IoFilter，IoFilter进行消息的过滤，格式的转换，在这个层面可以制定一些自定义的协议；
4. 最后IoFilter把数据交给 Handler进行业务逻辑处理，整个读取过程便完成了；
5. 写入过程反之。

# 第三章 移动互联网智能工具云平台设计

## 系统框架

以数据驱动程序设计的角度，云平台可划分为CLOUD数据库模块和程序模块。程序模块再按照功能和实现划分为三个子模块：后台管理系统、App HTTP服务端和App Socket服务端，如图 3.1 所示。

* 1. CLOUD数据库模块：用户信息、设备信息及设备采集的数据的存取；
  2. 后台管理系统：顾名思义就是提供智能工具管理功能的网站，主要提供厂家、产品、用户、设备管理以及更新反馈功能。
  3. App HTTP服务端：为客户端提供用户注册登录、好友系统、设备绑定分享等、云数据管理等接口。
  4. App Socket服务端：用于与客户端建立长连接，提供设备注册、设备状态维护、单设备登录、设备数据远程查看、设备远程操控等接口。

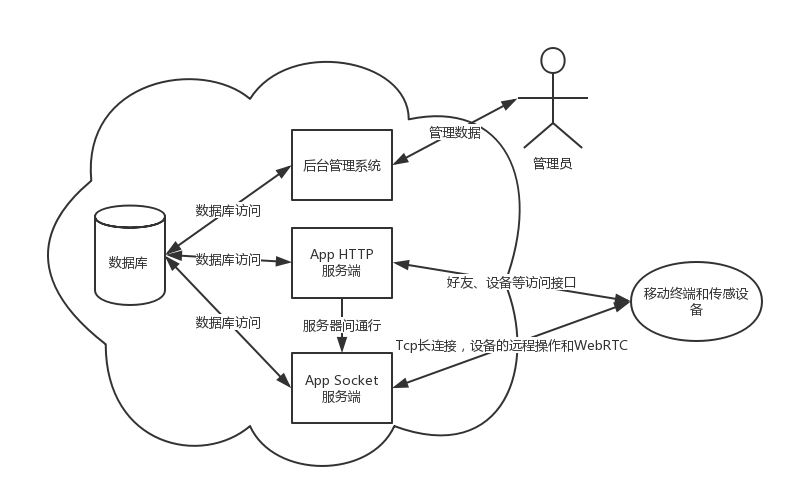


图 3.1 移动互联网智能工具云平台架构

## CLOUD数据库设计

通过分析云平台功能需求，建立以下12个表：

* ADMIN表 存储管理员相关信息；
* ALERTMODE表 存储传感设备的工作模式；
* APP\_UPDATE表 存储ANDROID APP版本信息；
* APP\_UPDATE\_CONFIG表 存储配置更新信息；
* COUNTRY表 存储国家编码信息；
* DEVICE表 存储设备相关信息；
* FACTORY表 存储厂家相关信息；
* FRIEND表 存储好友关系信息及状态信息；
* MESSAGE表 存储反馈信息和好友设备间相关消息；
* PRODUCT表存储产品相关信息；
* SHAREDEVICE表 存储分享设备信息
* USER 存储用户信息。
  1. 管理员信息（ADMIN表）：

表 3.1

| Name | Type | Null | Default | Comments | Extra |
| --- | --- | --- | --- | --- | --- |
| admin | varchar(16) | No | None | 管理员账号 | Primary key |
| pass | varchar(16) | No | None | 管理员密码 |  |
| type | int(1) | No | None | 账号类型 |  |
| createTime | varchar(30) | No | None | 创建时间 |  |
| mode | bit(8) | No | 0 | 管理员权限 |  |

* 1. 工作模式信息（ALERTMODE表）

表 3.2

| Name | Type | Null | Default | Comments | Extra |
| --- | --- | --- | --- | --- | --- |
| name | varchar(100) | No | None | 报警模式名 |  |
| value | char(4) | No | None | 指令值 |  |
| id | int(4) | No | None | 模式id |  |

* 1. ANDROID APP版本信息（APP\_UPDATE表）

表 3.3

| Name | Type | Null | Default | Comments | Extra |
| --- | --- | --- | --- | --- | --- |
| name | varchar(20) | Yes | NULL | app |  |
| code | varchar(30) | Yes | NULL | 版本号 |  |

* 1. ANDROID APP配置信息更新（APP\_UPDAT\_CONFIG表）

表 3.4

| Name | Type | Null | Default | Comments | Extra |
| --- | --- | --- | --- | --- | --- |
| content | varchar(16) | No | None | app更新内容 | Primary key |
| updateTime | varchar(20) | No | None | 上次更新时间 |  |

窗体底端

* 1. 国家编码信息（COUNTRY表）

表 3.5

| Name | Type | Null | Default | Comments | Extra |
| --- | --- | --- | --- | --- | --- |
| code | char(4) | No | None | 国家编码 | Primary key |
| text | char(60) | No | None | 对应国家 |  |

* 1. 设备信息（DEVICE表）

表 3.6

| Name | Type | Null | Default | Comments | Extra |
| --- | --- | --- | --- | --- | --- |
| physicalId | varchar(16) | No | None | 用户物理id | Primary key |
| deviceName | varchar(20) | No | None | 设备名字 |  |
| ownerId | int(15) | No | None | 设备主人id |  |
| factoryNumber | varchar(20) | No | None | 厂商编号 |  |
| productTypeNumber | varchar(20) | No | None | 产品类型编号 |  |
| productNumber | varchar(20) | No | None | 产品编号 |  |
| safeCode | varchar(20) | No | None | 设备安全码 |  |
| isOnline | int(1) | No | None | 设备是否在线 |  |
| ssid | varchar(30) | No | None | 设备连接的终端路由器或者手机 |  |
| iosuuid | varchar(50) | No | None | ios的uuid |  |
| isRemoveLink | int(1) | No | 1 | 设备锁定状态 |  |
| createTime | varchar(20) | No | None | 设备创建时间 |  |
| updateTime | varchar(20) | No | None | 设备更新时间 |  |

* 1. 厂家相关信息（FACROTY表）

表 3.7

| NAME | TYPE | NULL | DEFAULT | COMMENTS | EXTRA |
| --- | --- | --- | --- | --- | --- |
| FACTORYNUMBERrimary | CHAR(6) | NO | NONE | 厂家编号 | Primay key |
| TEXT | VARCHAR(30) | NO | NONE | 编号对应文本 |  |
| IMGURL | VARCHAR(100) | NO | NONE |  |  |
| CREATETIME | VARCHAR(20) | NO | NONE | 创建时间 |  |

* 1. 好友信息（FRIEND表）

表 3.8

| NAME | TYPE | NULL | DEFAULT | COMMENTS | EXTRA |
| --- | --- | --- | --- | --- | --- |
| FRIENDID | INT(16) | NO | NONE | 好友关系序号 | AUTO  INCREMENT,  Primary key |
| FRIENDA | INT(30) | NO | NONE | 朋友A逻辑ID |  |
| FRIENDB | INT(30) | NO | NONE | 朋友B逻辑ID |  |
| ANICKNAME | VARCHAR(20) | NO | NONE | B对A昵称 |  |
| BNICKNAME | VARCHAR(20) | NO | NONE | A对B昵称 |  |
| ISAGREE | INT(1) | NO | 0 | 是否同意 |  |
| CREATETIME | VARCHAR(20) | NO | NONE | 数据表创建时间 |  |
| UPDATETIME | VARCHAR(20) | NO | NONE | 更新时间 |  |

* 1. 消息信息（MESSAGE表）

表 3.9

| NAME | TYPE | NULL | DEFAULT | COMMENTS | EXTRA |
| --- | --- | --- | --- | --- | --- |
| ID | INT(16) | NO | NONE | 消息ID | AUTO\_ NCREMENT  Primary key |
| UUID | INT(16) | NO | NONE | 用户ID | FOREIGN KEY |
| CONTENT | VARCHAR(5000) | NO | NONE | 消息内容 |  |
| CREATETIME | VARCHAR(20) | NO | NONE | 发送时间 |  |
| DIRECTION | INT(1) | NO | NONE | 消息发送方向 |  |

* 1. 产品信息（PRODUCT表）

表 3.10

| NAME | TYPE | NULL | DEFAULT | COMMENTS | EXTRA |
| --- | --- | --- | --- | --- | --- |
| FACTORYNUMBER | CHAR(6) | NO | NONE | 厂家编号 | * FOREIGN KEY |
| FACTORYTEXT | VARCHAR(30) | NO | NONE | 厂家编号对应文本 |  |
| PRODUCTTYPENUMBER | CHAR(6) | NO | NONE | 产品类型 | FOREIGN KEY |
| PRODUCTTYPETEXT | VARCHAR(30) | NO | NONE | 产品类型编号对应文本 |  |
| PRODUCTNUMBER | CHAR(6) | NO | NONE | 产品编号 | FOREIGN KEY |
| PRODUCTTEXT | VARCHAR(100) | NO | NONE | 产品编号对应文本 |  |
| MINTEMPERATURE | FLOAT | NO | -40 |  |  |
| MAXTEMPERATURE | FLOAT | NO | 650 |  |  |
| NEWVERSION | VARCHAR(10) | NO | NONE | 设备最新版本 |  |
| CREATETIME | VARCHAR(20) | NO | NONE | 创建时间 |  |
| UPDATETIME | VARCHAR(20) | NO | NONE | 更新时间 |  |
| ALERT | VARCHAR(1000) | NO | NONE | 数据测量上下限及警报上下限 |  |
| FREQUENCY | VARCHAR(50) | NO | NONE | 发射频率 |  |

* 1. 分享设备信息（SHAREDEVICE表）

表 3.11

| NAME | TYPE | NULL | DEFAULT | COMMENTS | EXTRA |
| --- | --- | --- | --- | --- | --- |
| SHAREID | INT(16) | NO | NONE | 分享序号 | AUTO\_INCREMENT,  Primary key |
| OWNERID | INT(15) | NO | NONE | 设备主人ID |  |
| SHAREDEVICEUSERID | INT(15) | YES | NULL | 被分享者ID |  |
| DEVICEID | VARCHAR(16) | NO | NONE | 设备ID |  |
| SHARETYPE | INT(1) | NO | NONE | 分享类型 |  |
| ISMOVERIGHT | INT(1) | NO | 0 | 是否属于授权设备 |  |
| ISAGREE | INT(1) | NO | 0 | 是否同意 |  |
| CREATETIME | VARCHAR(20) | NO | NONE | 分享创建时间 |  |
| UPDATETIME | VARCHAR(20) | NO | NONE | 分享修改时间 |  |

* 1. 用户信息（USER表）

表 3.12

| NAME | TYPE | NULL | DEFAULT | COMMENTS | EXTRA |
| --- | --- | --- | --- | --- | --- |
| UUID | INT(16) | NO | NONE | 用户逻辑ID | AUTO\_INCREMENT,  Primary key |
| SSID | VARCHAR(30) | NO | NONE | 用户SSID |  |
| PHONE | VARCHAR(30) | NO | NONE | 用户手机号 |  |
| EMAIL | VARCHAR(30) | NO | NONE | 用户EMAIL |  |
| PASS | VARCHAR(30) | NO | NONE | 用户密码 |  |
| TOKEN | VARCHAR(70) | NO | NONE | 用户TOKEN值 |  |
| LASTLOGINIP | VARCHAR(20) | NO | NONE | 上次登陆IP |  |
| VERIFYCODE | VARCHAR(20) | NO | NONE | 验证码 |  |
| COUNTRYCODE | CHAR(4) | NO | 0086 |  |  |
| CREATETIME | VARCHAR(20) | NO | NONE | 创建时间 |  |
| UPDATETIME | VARCHAR(20) | NO | NONE | 更新时间 |  |

## 后台管理系统设计

### 整体架构

1. 系统技术架构

后台管理系统技术架构如图 3.2所示，系统采用ThinkPHP框架，基于MVC[8] 的软件架构模式设计，数据库采用MySQL实现，在具体功能模块开发中利用了极光推送实现消息的推送服务，后端对外提供HTTP访问接口，前端系统采用Bootstrap[10] 框架，并使用jQuery[11] 库，快速搭建一个美观易用的前端系统。

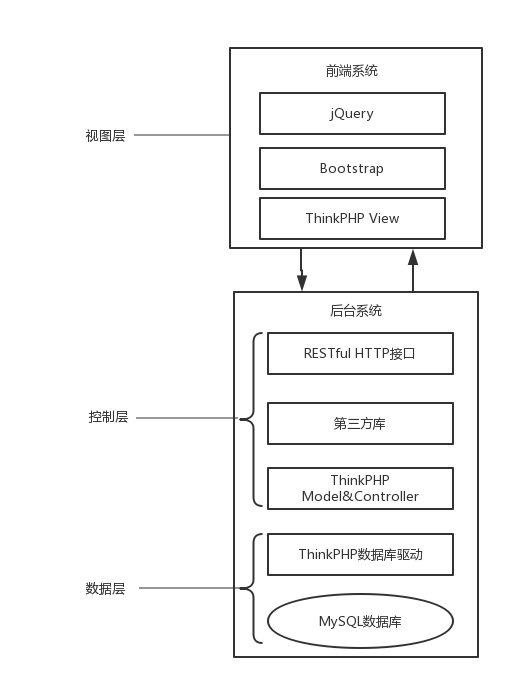


图 3.2 后台管理系统技术架构

视图层，处于软件架构的最上层，负责具体业务和视图展示，是与用户直接进行交互的界面接口，提供用户管理、设备管理、更新反馈等功能的操作界面。

控制层，本系统中的控制层主要用于接收前台页面发送的请求，并根据请求调用业务逻辑层相应的操作进行请求处理。请求包括：页面导航请求、数据库读写请求和第三方API调用请求处理。

数据层，接收业务逻辑层的调用，将对数据库访问请求进行封装。

### 系统功能结构设计

后台管理系统功能架构如图 3.3所示。其中实线表示功能的从属关系，虚线表示功能的关联关系。

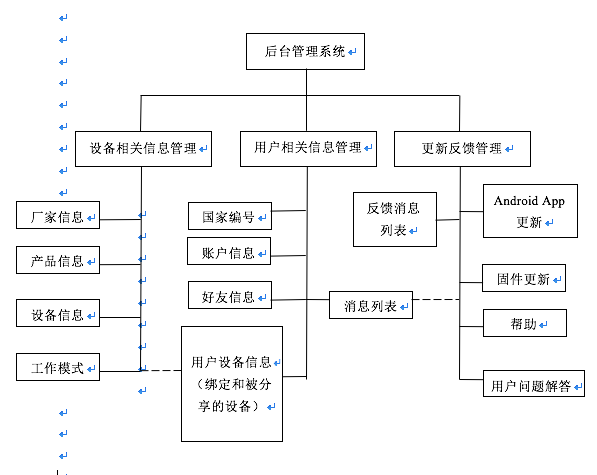


图 3.3 后台管理系统功能架构

1. 设备相关信息管理

主要分为厂家信息管理、产品信息管理、设备信息管理和工作模式管理。传感设备来自很多不同的厂商，且传感设备有很多不同的种类，如温度、湿度、风速传感器，同一厂商的同一类型设备也会有不同的型号。这三个方面的不同分别对应以上三个功能模块。厂家信息管理包括：厂家编号、厂家名称、创建时间、厂家图片。产品信息管理包括：厂家编号、厂家名称、产品类型编号、产品类型名称、产品编号、产品名称、数据上下限及报警值上下限，蓝牙发射频率，固件版本。设备信息管理包括：设备物理地址、设备厂家、设备类型、设备名称、设备主人、设备在线状态、固件版本。工作模式管理包括：模式索引ID、模式命令字、模式名称。

1. 用户相关信息管理

主要分为国家编号管理、和用户信息管理两个部分。国家编号用于手机号码注册，管理员可对其增删改查。用户信息管理提供两个功能，一提供账户基本信息的查看修改，包括：用户账号、密码、国家、ip、注册时间、登录状态的查看，用户的删除，二是提供用户相关的好友、绑定设备、被分享设备、反馈简要等信息的查看、删除操作。

1. 更新反馈

更新反馈功能涉及反馈消息列表、app更新、固件更新、帮助、用户问题解答几个部分。反馈消息列表提供反馈消息的查看和回复功能，app和固件更新提供软件版本更新服务，帮助提供app使用说明书的编辑上传，用户问题解答则提供常见问题录入功能。

## App HTTP服务端设计

### 系统技术架构

App HTTP服务端技术架构如图 3.4，系统采用RESTFul[9] 架构，使用PHP语言开发，使用单一文件接口模式设计，一个PHP文件对应一个接口以降低耦合度，数据库采用SQLite和MySQL，使用云之讯实现短信验证功能，使用极光推送完成好友信息、设备信息的推送功能，使用phpmailer实现邮件功能。

内部接口设计如图 3.4所示，分为数据库接口、工具类接口和通信接口

* 1. 数据库接口封装数据库初始化和连接过程，隐藏数据操作过程，为应用提供数据提供增删改查操作接口；
  2. 工具类接口封装一些常用的功能调用，包括数据排序、用户token检测、用户ID提取、字符串处理、字节处理、消息打包等功能；
  3. 通信接口，App HTTP服务端和App Socket服务端的通信接口，封装tcp请求、数据打包发送等操作。



图 3.4 App HTTP服务端技术架构

### 用户接口设计

在考虑用户接口的详细设计前，必须要考虑一下几个基础方面的设计：

* 1. 身份验证的设计

用户在调用敏感资源时需要验证用户身份[12] ，用户身份验证通常使用登录的方式来实现，然而服务端采用RESTFul架构，RESTFul最重要的一个设计原则就是，客户端与服务器的交互在请求之间是无状态的，所以如何维护登录状态是需要考虑的。

普遍的做法是当涉及到用户状态时，每次请求都要带上身份验证信息。实现上，大部分都采用token的认证方式，流程为：

* 1. 用户用账户密码登录成功后，服务器生成一个用户token，并返回给客户端；
  2. 客户端将token保存在本地，后续发起其他请求时，将token作为参数携带；
  3. 服务器检查token的有效性，有效则返回数据，若无效，分两种情况：token过期和token错误，分别返回不同的错误信息：
  4. 消息格式的设计

接口数据采用JSON[13] 格式进行传输，因为JSON格式有一下优点：

1. 与XML相比，JSON更加的简洁，易于阅读；
2. JSON更加轻量级，不管是编写，传输，还是解析都更加高效；
3. JSON在传输过程中采用了压缩技术，更加的节省宽带；
4. 众多的语言支持，如javascript，python，C，C++等主流语言都支持；
5. 特别的，在PHP世界, 已经有PHP-JSON和JSON-PHP出现了, 便于PHP序列化后的程序直接调用. PHP服务器端的对象、数组等能够直接生JSON格式, 便于客户端的访问提取。

不过，需要注意的是，JSON的值只有六种数据类型：

* 1. Number：整数或浮点数
  2. String：字符串
  3. Boolean：true 或 false
  4. Array：数组包含着方括号[]中
  5. Object：对象包含在大括号{}中
  6. Null：空类型

所以，传输的数据类型不能超过这六种数据类型。特别的，对于Date类型JSON会将其转换为特定格式的字符串，但不同的解析库解析方式可能不同，可能导致异常。为了根除这种问题，设计方案中采用毫秒数表示日期。

基于JSON设计出如下消息格式：

{

**"status"**:"",

**"info"**: ""|{}|[]

}

status: 状态信息，成功时为”success”，失败时为“error”;

info：结果信息，可以为失败时的错误信息，格式为字符串，成功时的字符串、对象或数组信息。

上述身份验证和消息格式设计为每个接口提供了基础支持，下面叙述用户接口的设计：

1. 验证码接口：获取注册登录或密码找回时所需的短信或邮件验证码；
2. 添加用户接口：提交用户注册信息和验证码完成用户的注册；
3. 登录接口：提交用户信息，返回token用于后续访问认证；
4. 添加好友接口：通过手机号或邮箱请求添加另一个用户为好友；
5. 确认好友请求接口：确认其他用户发起的好友请求；
6. 修改好友昵称接口：修改数据库好友信息中的昵称字段；
7. 删除好友接口：解除好友关系，删除好友信息；
8. 添加设备接口：已登录用户对设备进行绑定操作；
9. 分享设备接口：用户登录绑定设备后，分享此设备给已建立好友关系的用户；
10. 获取设备列表接口：获取用户已绑定设备、被分享设备列表；
11. 修改设备名称接口：自定义设备昵称；
12. 移交设备操作权限接口：移交已分享设备的权限给被分享好友；
13. 确认移交操作权限接口：权限移交目标用户确认接受权限移交；
14. 删除设备接口：删除用户绑定的设备，同时会解除此设备相关的分享和权限移交；
15. 锁定或解锁设备接口：设备被绑定后可通过此接口锁定或解锁设备，被锁定设备不可被其他用户绑定，未锁定设备可被其他用户绑定，被其他用户绑定后，原有绑定、分享、权限移交关系解除。
16. 获取用户消息接口：获取未经同意的好友请求、未经同意的设备分享请求、未经同意的权限移交请求、消息反馈、历史消息记录；
17. 删除消息接口：删除消息反馈和历史消息列表的消息；
18. 移除分享接口：用户解除设备的所有分享关系；
19. 数据备份接口：备份用户设备采集的数据到云端；
20. 数据恢复接口：下载云端备份数据到本地；
21. 反馈接口：提交反馈信息。

## App Socket服务端设计

### 通信模块的设计

本文使用自定义的通信协议。基于数据传输方向的不同，本文把系统通信协议中的消息分为五种类型：移动端到服务器的消息、服务器到移动端的消息、服务器到服务器内部消息和移动端到移动端以服务器作为中转的消息[15] [16] 。五种消息都采用相同的协议格式，如图 3.5所示。

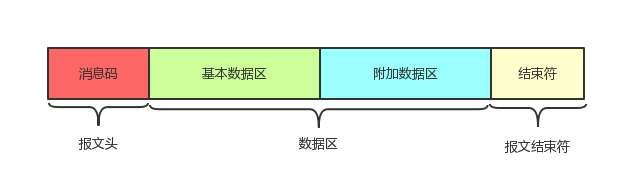


图 3.5 消息格式

报文头的消息码表示该消息的类型。数据区包括基本数据区和附加数据区两个部分，基本数据区存储协议的一些必要信息，如源session id和目标session id，附加数据区存储需要服务器处理或者转发的数据。结束符标识报文结束。

1. 解码字节流

TCP协议是、面向字节流的，服务器系统接收到的数据均为字节流数据。当服务器接收到数据后，首先根据数据的第一个字节（消息码）判断消息的类型。然后将字节解码为对应的消息对象。其工作流程如图 3.6所示。

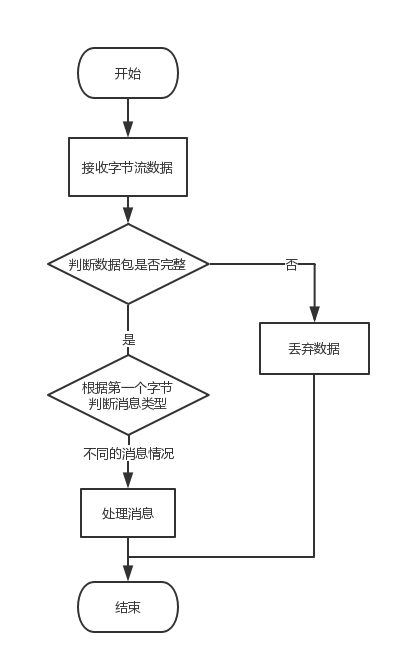


图 3.6 解码字节流流程图

1. 编码消息对象

当服务器系统需要向移动端发送数据时，需要将对应的消息对象编码为字节流数据。由于所有协议中消息的格式相同，所以两种消息的编码过程时一致的。首先获取一个字节的消息码，再获取数据区数据，讲这些数据一次写入一个字节数组，最后将这个数据发送出去。

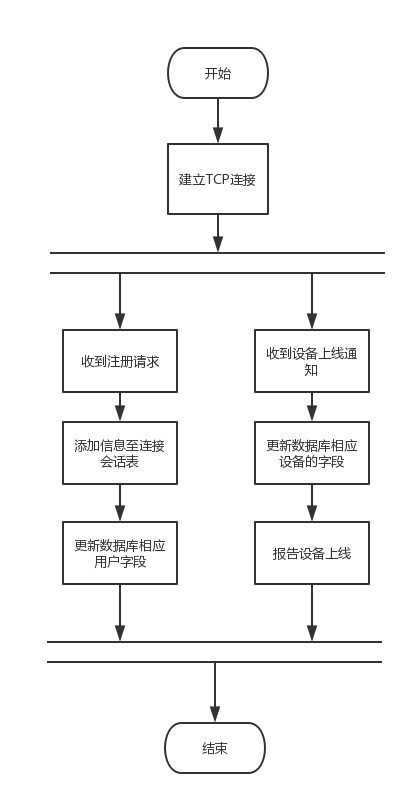
### 设备注册模块的设计

客户端被用户打开并登录后，会向服务器系统发起TCP连接，建立连接后客户端会发起注册请求。服务器系统接收到请求后，会将客户端的信息注册到会话信息表中，并更新状态至数据库。

会话信息表是一个哈希类型的数据结构，整个服务器系统维护一张该表。会话信息表存储的是一个键值对，这个键值对应的键为客户端的ID，值为对应的连接会话对象。

除了记录客户端的ID和连接会话外，服务器系统还要知道当前客户端的设备连接状态，无连接则不记录任何信息，有连接则需要记录连接设备的物理ID，设备连接状态的保存在数据库中完成。当一个已经建立socket连接的客户端连接设备时，会向服务器系统发送设备上线消息，这个消息包含客户端ID和所连接设备的物理ID，接收到消息后服务器系统解析出客户端ID和设备ID，然后将这些数据更新到数据库中，并将当前所连接的客户端ID群发给所有共享者，这样在远程设备操作时就可以找到对应设备所连接设备。其流程如图 3.7所示。

图 3.7 设备注册模块工作流程图



### 消息处理模块的设计

消息处理模块的主要功能是对接收到客户端消息进行处理。包括透传消息、群发消息和心跳包[14] 处理。心跳包由客户端定时发送一个短消息用于维护连接，其实现较为简单，所以本文不单独进行叙述。

1. 透传消息的处理

所谓透传消息，指消息从一个客户端发送消息通过服务器转发给另一个客户端，在这个过程中服务器不对消息内容做任何处理或记录，在本文中指不对扩展数据区的内容做处理或记录。

为实现透传功能，服务器建立了会话信息表来，会话信息表的作用是将客户端的ID和连接会话对象绑定，通过客户端ID可以查找到与之对应的的唯一一个连接会话对象。

在处理透传消息时，服务器首先从需要转发的消息中获取数据接收端的客户端id，然后查找会话信息表，获取数据接收端的连接会话对象。再根据通信协议格式将数据重新封装成对应的消息对象。最后通过数据接收端的连接会话对象将消息发送出去。服务器透传消息的流程如图 3.8所示。

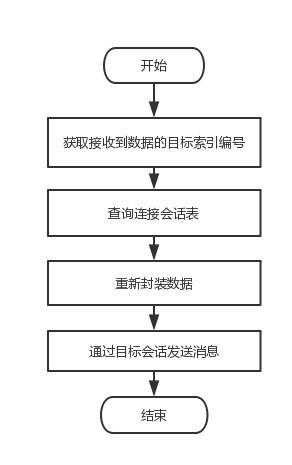


图 3.8 透传消息的工作流程

1. 群发消息的处理

群发消息主要存在于共享设备状态处理场景，这些场景包括：共享设备上下线， 设备主人删除共享设备。在以上场景中需要群发消息给所有共享者，通知他们刷新设备列表中设备的状态。

服务器首先从数据库查询共享设备的所有共享者的客户端ID，然后查询会话信息表获取对应的客户端连接会话对象。再将数据封装，通过该连接会话对象发送出去。重复上述步骤，直至所有共享者客户端ID都被遍历。群发消息的处理流程如图 3.9所示。

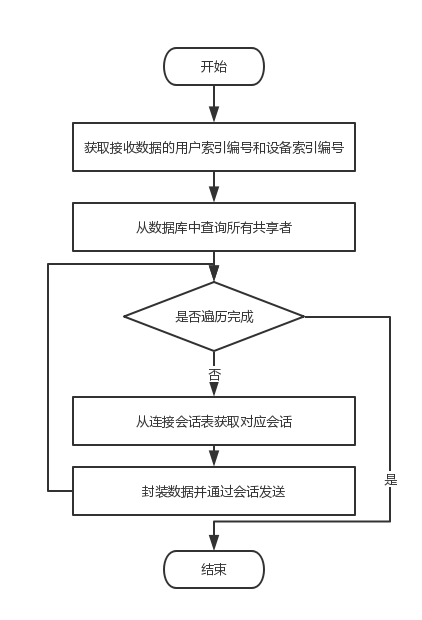


图 3.9 群发消息工作流程图

### 其他模块设计

1. 单设备登录模块

用户账号同时只能在一个移动终端上登录，这样设计的原因是一个账号多台设备登录的需求并不大，另外要维持多设备登录服务器代价太大，所以限制一个账号同一时刻只能在一台设备上登录。

从整个系统来看实现单设备登录有两种方式，一个是主动自检，另一个是被动下线。

1. 主动自检

客户端主动向服务器系统发送查询消息，查询当前用户的登录状态，然后与本地保存的状态进行对比，如果发生变化则说明当前账户已经在其他设备上登录，此时客户端主动下线。

主动自检需要定时检测登录状态，这显然比较耗费设备、网络和服务器资源，所以比较好的做法是把检测和心跳包结合在一起来降低整体的资源消耗。

1. 被动下线

被动下线发生在设备注册过程中，服务器系统在记录客户端ID前会首先查询当前设备所对应连接的客户端ID，经过比对，如果新旧ID不同则说明登录状态发生变化，此时使用旧客户端ID作为索引编号查询获取对应的连接会话，然后向此连接会话发送下线通知以达到被动下线的目的。

1. 系统退出模块的设计

由于客户端与服务器系统建立的都是稳定的长连接，因此当客户端连接不可用时，需要一种机制来断开连接，释放资源，以此避免服务器资源的占用。

服务器系统首先需要找到数据库设备表中，对应的客户端记录，将其删除，然后从连接会话表中删除键值对，最后断开连接。

1. 日志记录模块的设计

日志系统是一种不可或缺的跟踪调试工具，特别是在无人职守的后台程序和哪些没有跟踪调试环境的系统中有着广泛的应用。长期以来，日志系统作为一种应用程序服务，对于跟踪调试、程序状态记录、崩溃数据恢复都有非常现实的意义。

日志记录模块主要有两个功能。一是设备服务器系统的日志记录方式，如日志保存的位置、日志的名称、日志记录的格式等。二是记录员服务器系统执行的操作，如建立连接、消息的发送、发生的异常处理信息等，用于服务器系统的后期维护。

服务器系统的日志采用文本的方式存储在服务器系统程序的根目录下的Logs文件夹中，采用每天产生一个日志文件的方式更新记录。日志文件主要记录服务器系统的通信操作，如当有客户端发起注册时，日志文件记录客户端的ID、设备ID、客户端操作、操作发生的时间等信息。

# 第四章 移动互联网智能工具云平台实现

## 后台管理系统实现

后台管理系统分为系统登录、用户信息管理、厂家信息管理、产品信息管理、设备状态管理、国家编号管理、更新反馈管理、工作模式管理八个模块。其中用户信息管理、厂家信息管理、国家编号管理、工作模式管理、设备状态管理实现较为简单，不做详述。

### 系统登录模块实现

后台管理系统中用户必须通过登录后，才能够使用系统，用户登录后，可以进行权限范围内的操作。超级用户登录后，可以进行所有操作，同时管理普通管理员账号。

系统登录模块中最重要的部分就是访问控制权限[17] ，需要根据用户的身份的不同来展示不同的可见菜单、可访问功能。系统登录模块的流程如图 4.1所示

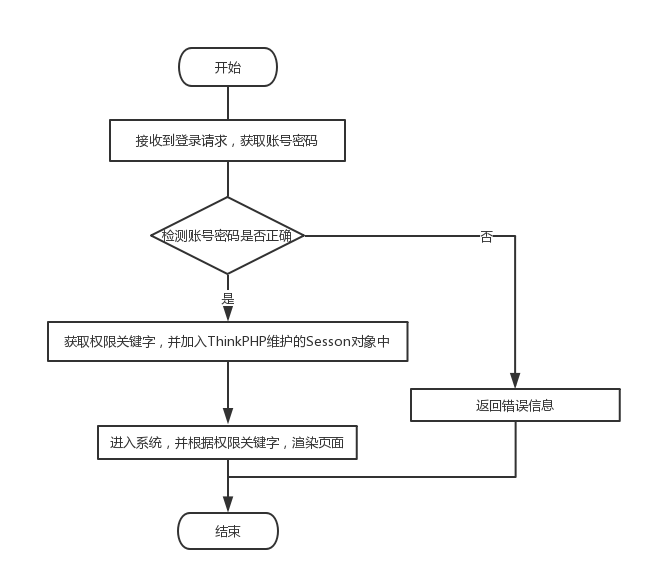


图 4.1 后台管理系统登录模块工作流程图

系统提供了Session管理和操作的完善支持，因此可以对一次登录维护一个权限关键字，本文权限关键字存储于数据库的admin表中，它为一个8bit的数据，每一位表示系统模块的一个权限，置为1则表示有对应模块的权限。在用户登录初始化时，系统从数据库读取权限关键字，然后存储到Session中。在视图层利用ThinkTemplate模版引擎获取权限关键字再按位计算显示对应的菜单，实现可见菜单的限制。在控制层系统接收到页面HTTP请求时，会首先从Session中获取权限关键字，计算当前用户是否具有该请求权限，有则处理，无则返回错误，实现可访问功能的限制。

### 产品信息管理模块实现

产品信息管理模块主要实现产品登记登记功能。一个产品需要包含一下信息：厂家编号、厂家名称、产品类型编号、产品类型名称、产品编号、产品名称、测量上下限和警报值、发射频率、固件版本。

厂家编号与名称在厂家信息管理模块实现，其他信息由管理员在此模块录入。其中测量上下限和警报值需要记录产品各个工作模式下的测量上下限和警报上下限，其记录格式如下：模式1:测量下限/测量上限;报警下限/报警上限#模式2...。发射频率指智能工具的蓝牙发射频率，分为两种类型：固定档位、连续档位。固定档位形如：0.75#0.85#0.95，客户端即可选择0.75、0.85、0.95三个发射频率。连续档位形如0.10/1.00，客户端即可选择0.10至1.00间的任意值作为发射频率。如图 4.2所示。



图 4.2 产品信息管理

### 更新反馈管理模块实现

更新反馈管理主要实现反馈消息、App更新登记、固件更新登记、帮助文档上传、常见问题登记。

管理员可查看用户反馈的消息，并做出回复。后台接收到前台的回复请求后需要更新数据库中feedback表，将对应记录的isRes字段设置为1，表示已经回复，同时需要将回复的消息插入到数据库的message表中。为了保证操作的原子性、一致性，在实现时需要使用MySQL的事物[18] 机制。在ThinkPHP中，SQL事物的使用十分简单：

开始事物：$model->startTrans();

提交事物：$model->commit();

回滚事物：$model->rollback();

以上将feedback表的更新和message表数据的插入步骤放置在$model->startTrans();之后，当且仅当feedback表的更新和message表数据的插入都成功后才调用$model->commit()，反之调用$model->rollback()。

App更新登记，管理员需将APK命名为:名称#版本号.apk的格式，然后上传apk，系统解析app的版本号并将其更新至app\_update表，后续查询更新只需要查询此表即可。

固件更新原理与App更细类似，所以不再详述。

帮助文档上传，文档要求如下：

表 4.1 语言代号映射表

|  |  |
| --- | --- |
| 语言名称 | 语言代号 |
| 简体中文 | zh\_CN |
| 英文 | en |
| 法语 | fr |
| 西班牙语 | es |
| 葡萄牙语 | pt |
| 俄语 | ru |
| 日语 | ja |
| 德语 | de |

1. 上传文件格式zip,命名规则help\_语言代号.zip，语言代号如表 4.1所示。
2. 目录格式以中文为例

help\_zh\_CN.zip/　　　　 文件名

├── index.html 帮助文档页面

├── img/ 帮助页面所用的图片目录

　　　├── xxx 图片（命名无规则只需与html中图片名称对应）

系统首先检测上传是否发生以下错误：文件大小超出了服务器的空间大小、要上传的文件大小超出浏览器限制、文件仅部分被上传 、没有找到要上传的文件、服务器临时文件夹丢失、文件写入到临时文件夹出错。检查通过后，系统将其解压至指定文件夹，并根据语言代号命名文件夹，这样指定url访问前缀和国家代码就可访问到对应的帮助文件。

用户问题解答则集成了一个富文本编辑器，管理员可根据实际经营过程上传一些解答性的问题和解答。

## App HTTP服务端实现

App HTTP服务端接口众多，但实现起来大同小异，所以本节只介绍注册模块、登录模块、设备绑定模块这三个模块的实现。

### 注册模块的实现

为了防止恶意注册，需要检测注册操作来自用户本人，这通常利用验证码机制来实现[19] 。App HTTP服务端设计时允许两种注册方式，手机号码和邮箱，因此需要针对手机号码和邮箱实现不同的验证模块。短信验证码通过接入云之讯服务来实现。邮件验证码通过使用第三方库phpmailer实现。

以短信验证方式为例，流程如下：

* 1. 客户端首先向服务器系统请求国家编码信息，服务器系统返回相应信息。
  2. 客户端以phone(手机号码)，country(国家编码)为参数向服务器系统发起发送验证码的请求。
  3. 服务器系统接收到请求后，首先检查手机号码格式判断是否有效，再检查用户是否已经注册过，以上检测都通过，则随机生成6为数字验证码，调用云之讯接口，请求发送验证码。反之则返回错误信息，注册结束。
  4. 用户收到短信验证码后，完善表单，发起注册请求。
  5. 服务器系统接收到请求，再此重复步骤(3)中的检查过程，通过则返回成功信息，失败返回错误信息。
  6. 注册结束。

注册操作的时序图如图 4.3所示

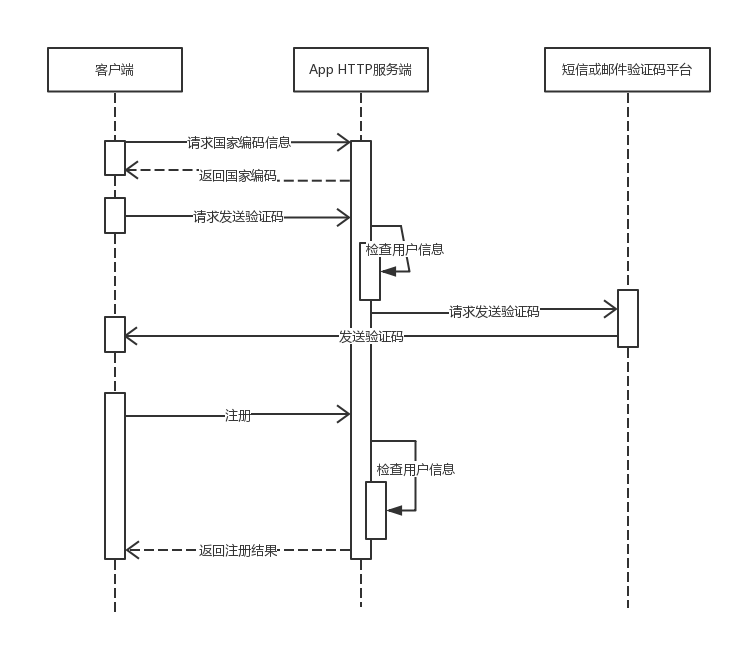


图 4.3 App HTTP注册接口时序图

### 登录模块的实现

服务器系统处理客户端请求时，往往需要验证用户身份，用户身份可以通过账号密码来验证。但是如果每次请求都携带账户密码信息，无疑增加系统的被攻击点，只要有一个接口的安全性未得到保障，用户的账号密码就会泄漏。

实际开发过程中，往往通过token验证用户的身份，token在用户登录时分配给用户，并更新到数据库对应用户的token字段和token更新时间字段中，用户在之后的访问中只要携带token即可。服务器系统在接收到个请求后都会验证验证码的正确性，并检测token是否超时。验证通过则执行相应操作，验证失败返回错误信息。

### 设备绑定模块的实现

一个设备只能被一个账户绑定，绑定类型有两种：弱绑定和强绑定。弱绑定表示用户账户与设备间只是申明绑定关系，但是其他用户依旧可以绑定该设备，只是在绑定时有所提示。强绑定，表示绑定设备被用户锁定不运行其他用户强制绑定。设备绑定模块涉及两个接口：请求添加设备，和请求锁定设备。请求添加设备成功后，用户账户和设备的弱绑定关系就建立了。请求锁定设备必须以账户和设备已经是弱绑定关系为前提，成功后，则强绑定关系建立。

## App Socket服务端实现

### 通信模块的实现

在MINA框架中，专门提供了一个过滤器ProtocolCodecFilter来处理协议的编解码问题。过滤器ProtocolCodeFilter需要通过一个编解码工厂ProtocolCodeFactory来构造。在编解码工厂中，添加具体的编解码器。

其中编码器的实现很简单，只需调用接口即可。而解码器需要处理数据包断包和粘包问题。TCP协议保证了数据包在传输过程中的正确有序，但并不能保证发送端的一个写操作触发接收端一个读事件。MINA框架是基于事件驱动的，简单的TCP协议不能满足要求，在程序中需要一种机制是的发送端的一次写操作能出发一个服务器端的读事件。另外，在程序中，需要一种方法来找出当前消息的结尾和下一条消息的开始在哪里。根据第三章 移动互联网智能工具云平台设计中通信协议设计可知，通过识别数据报结束符可以解决断包和粘包的问题。其核心的方法doDecode伪代码实现如下：

1 boolean doDecode(session,buffer,callback){

2 start = 获取buffer中当前数据索引位置;

3 标记buffer当前数据位置;

4 **while** (buffer中仍有数据) {

5 la = 从buffer中读取一字节数据;

6 **if** (la为报文结束字符'#' && buffer中无剩余数据) {

7 callback(buffer中所有数据);

8 **return** **false**;

9 }

10 **if** (la为报文结束字符'#' && buffer中有剩余数据) {

11 callback(buffer中从start开始到当前位置的所有数据);

12 **return** **true**;

13 }

14 **if** (la不是报文结束字符'#' && buffer中无剩余数据) {

15 将buffer位置指针重置到标记的位置;

16 **return** **false**;

17 }

18 }

19 **return** **false**;

20 }

session：当前连接会话。buffer：数据缓冲对象。callback：消费数据的回调，回调后，数据将交给handler处理。方法返回true时，解码器会检查buffer中数据是否被消费，如果没有数据被消费MINA框架直接抛出异常，如果有数据被消费，则检查是否还有剩余未被消费数据，如果有，则在此执行doDecode()方法,没有则等待，直到有新的数据被缓冲。返回false时，解码器停止执行doDecode()方法会，并将buffer中未被消费数据暂存至session,然后等待接收新的数据，新的数据到来时，解码器将session中暂存的数据与新数据合并缓冲，然后调用doDecode()方法。

当服务器系统接收到数据时，会使用编解码过滤器进行数据解码。过滤器首先根据报文结束符’#’对数据包进行处理，形成完整数据包后回调给Handler进行处理，通讯模块的时序图如图 4.4所示。

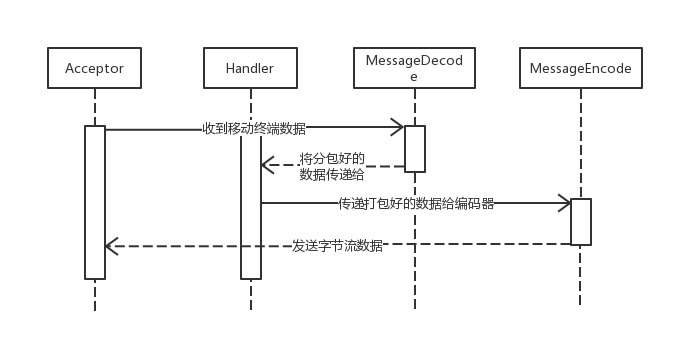


图 4.4 App Socket服务端通讯模块时序图

### 设备注册模块的设计

服务器系统接收到注册请求后，对当前连接进行注册。服务器系统设备注册的类图如图 4.5所示。

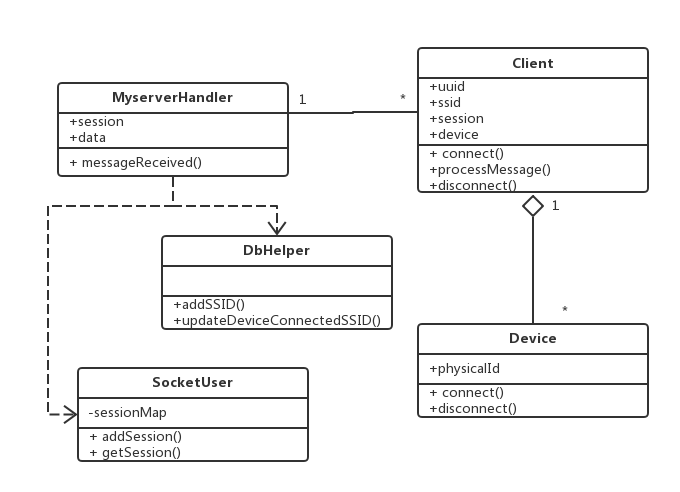


图 4.5 App Socket设备注册类图

其中，MyServerHandler类为系统消息处理类，经过过滤器ProtocolCodecFilter过滤的数据由该对象进行处理，属性值session代表了当前网络连接，属性值data代表接收到的数据包。

Client类是对客户端的抽象，属性值uuid，表示客户端当前登录账户的索引编号，属性值ssid表示注册连接会话的索引编号，属性值device表示Device类的实例。

Device类是对传感设备的抽象，属性值phsicalId表示设备的物理ID，通常为mac地址。

SocketUser类为连接会话表的维护类，静态属性值sessionMap是一个全局的哈希表，用于存储Client对象中ssid属性值和session对象组成的键值对。

DbHelper类是数据库处理类。

服务器系统收到注册请求后，首先解析出Client的uuid和ssid，并获取对应连接的session对象，然后使用SocketUser类的静态方法addSession()把ssid作为键，session对象作为值添加到sessionMap表中，同时通过DbHelper对象的addSSID()方法更新user表中对应uuid的用户的ssid字段。

服务器收到设备上线消息后，首先解析出Client的ssid和physicalId，然后通过DbHelper对象的updateDeviceConnectedSSID()方法更新device表中对应physicalId的记录的ssid字段

设备注册模块的活动图如图 4.6所示。

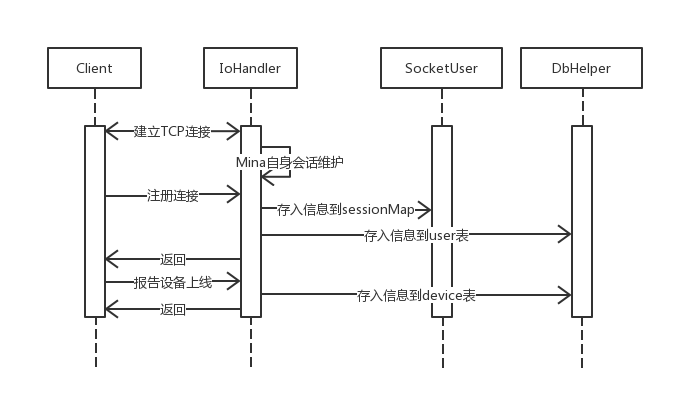


图 4.6 App Socket设备注册时序图

### 消息处理模块

服务器系统收到消息后，对消息数据进行解析。按照实现功能的不同，消息处理可以分为心跳包消息的处理、透传消息的处理和群发消息的处理。心跳包由客户端定时发送一个短消息用于维护连接，其实现较为简单，所以本文不单独进行叙述。消息处理模块的类图如图 4.7所示。

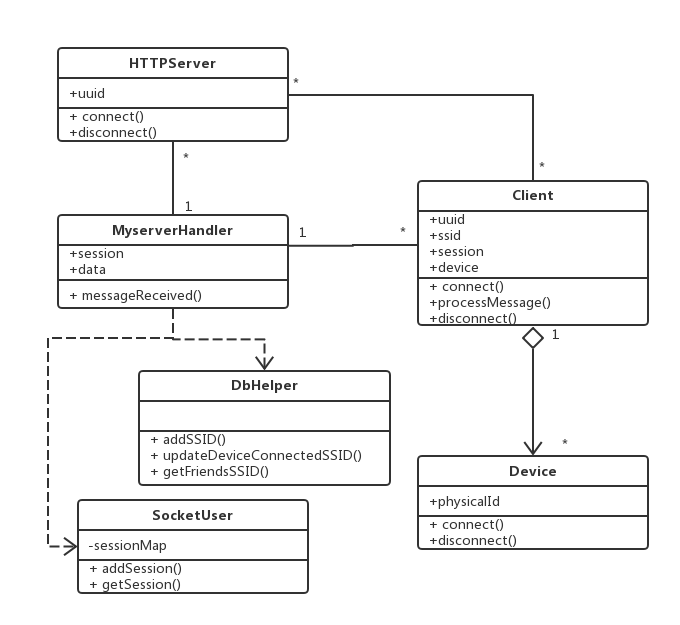


图 4.7 App Socket消息处理模块类图

HTTPServer类是App HTTP服务端的抽象，uuid表示发起HTTP请求用户的索引编号。

1. 透传消息的实现

Client对象向服务器系统发送需要转发的数据，服务器系统接收到后，解析出消息类型如果是转发消息则解析基本数据区中的数据，即源ssid和目标ssid，然后使用SocketUser类的静态方法getSeesion()获取对应目标ssid的session，如果获取不到则表示目标ssid未注册连接，直接丢弃消息，反之将源ssid和目标ssid的位置交换重新打包，通过目标连接会话发送给客户端。整个过程的系统活动图如图 4.8所示。

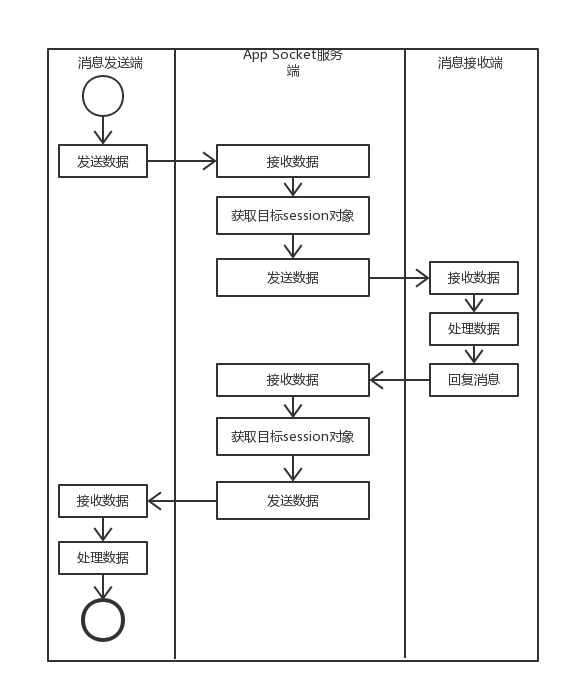


图 4.8 透传消息活动图

1. 群发消息的实现

群发消息有两个场景，一是共享设备主人删除设备绑定，二是共享设备上下线。设备删除接口是由App HTTP服务端提供的，所以这里涉及到两个HTTP服务端和Socket 服务端的通信问题，本文所采用的解决办法是让HTTP服务端向socket服务端发送TCP请求。

当Client对象向HTTP服务端请求删除设备绑定后，HTTP服务端先执行相应的解绑操作，然后通知socket服务端，socket 服务端接收到设备绑定删除消息后，解析消息得到uuid，然后通过DbHelper类的getFriendsSSID()方法查询对应uuid的所有好友当前的ssid，构成一个好友的ssid集合，再查找出对应的session集合，遍历发送更新设备列表消息。整个系统活动图如图 4.9

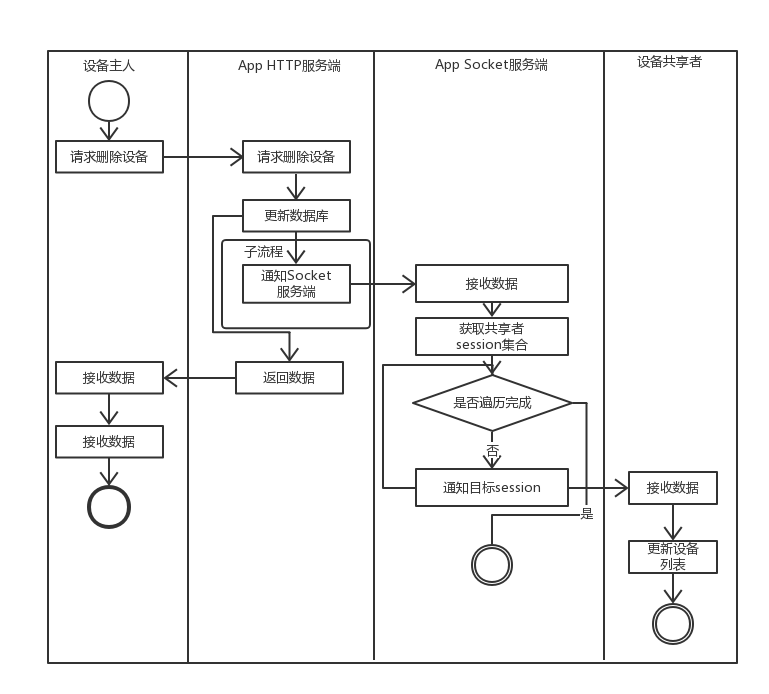


图 4.9 设备删除消息群发活动图

设备上下线，服务器系统解析出ssid和phsicalId，然后通过DbHelper类的getRemoteSSID()查询所有共享者的ssid集合，再获取到session集合，然后通知刷新设备的状态。

### 其他模块实现

1. 单设备登录模块的实现

用户账号同时只能在一个移动终端上登录，主要通过两个方式实现：一个是主动自检，另一个是被动下线。

1. 主动自检

客户端向服务器系统发送心跳包中携带uuid，服务器系统解析出uuid通过DbHelper的getToken()方法获取对应uuid用户的token，并打包返回给客户端，客户端与本地保存的token进行比较，如果不一致则表明账户在其他设备上登录，或者登录超时。

1. 被动下线

被动下线处理流程发生在设备注册连接的过程中，服务器系统解析出ssid、uuid，在通过DbHelper的getSSID()方法获取对应uuid索引的用户的ssid，称为ossid，若两者不同，则表示连接发生了改变，通过ossid获取session对象，再通过session发送下线消息。

1. 系统退出模块的实现

App Socket服务端连接退出的策略是：当某个会话连接没有在规定时间内与服务器系统进行通信，那么服务器系统就认为此连接会话已经失效，主动断开该连接。为了维持客户端与服务器系统的长连接，客户端需要定时向服务器系统发送心跳包消息。

当通信超时后，MINA框架会产生一个Idle事件，这个事件对应于IoHandler中的sessionIdle()方法。在sessionIdle()方法中，调用sessionClosed()方法进行退出处理。处理过程如下：

查询SocketUser中的sessionMap表，找到对应的session对象，关闭session对象连接，并从sessionMap表中删除记录。

1. 日志记录模块的实现

MINA框架提供了一个日志过滤器LoggingFilter实现日志记录功能。App socket服务端使用Log4j包来设计实现日志记录功能。

设置日志过滤器：

LoggingFilter loggingFilter = new LoggingFilter();

chain.addLast(“loggin”,loggingFilter);

App socket服务端在消息处理模块的所有处理中都添加了详细的日志记录，便于程序状态的记录和后期系统的维护。

# 第五章 系统测试

## 功能测试

### 测试环境搭建

硬件环境：

* 1. 云平台程序运行于阿里云的云服务器上，服务器操作系统为Ubuntu 4.8.2，CPU为 Intel(R) Xeon(R) CPU E5-2650 v2 ，主频2.60GHz，内存2G。
  2. 两台Android手机作为客户端，分别称为终端A终端B。
  3. 温度传感器作为作为测试设备。

软件环境搭建包括： Apache服务安装配置、Tomcat服务安装配置、MySQL数据库服务安装配置、PHP运行环境搭建、Java运行环境搭建及phpMyAdmin安装。

以Ubuntu[19] 系统环境为例，搭建过程如下：

1. 安装apache
   1. 安装命令：sudo apt-get install apache2。
   2. 配置：执行sudo /etc/init.d/apache2 restart重启apache服务。
   3. 检测方式：在浏览器里输入[http://localhost](http://localhost/)或者[http://127.0.0.1](http://127.0.0.1/)，如果能正常访问到Apache主页，说明Apache就成功的安装了，反之安装失败。
   4. 注意事项：Apache的默认安装，会在/var下建立一个名为www的目录，这个目录为Web根目录。
2. 安装PHP
   1. 安装命令：sudo apt-get install libapache2-mod-php5 php5 php5-mysql
   2. 配置：修改Apache的配置文件http.conf，添加index.php至DirectoryIndex变量后，添加AddType application/x-httpd-php-source .phps和AddType application/x-httpd-php .php。执行sudo /etc/init.d/apache2 restart重新启动Apache，使其加载PHP模块。
   3. 检测方式：在Web目录下面新建一个test.php文件，然后输入:<?phpinfo();?>至test.php文件中，保存文件，在浏览器里输入http://127.0.0.1/test.php，如果在网页中显示PHP模块信息，则说明PHP运行环境已经搭建成功。
   4. 注意事项：无。
3. 安装JDK
   1. 安装命令：sudo apt-get install oracle-java8-installer
   2. 配置：无。
   3. 检测方式：执行java –version，如正确显示Java版本信息，则表明安装成功。
   4. 注意事项：无。
4. 安装Tomcat

安装方式：手动安装，安装过程如下：

* 1. 从官网下载Tomcat安装包，解压至指定目录；
  2. 切换至Tomcat目录，执行sudo chmod +x ./\*和sudo chmod +x ./bin/\*.sh为执行文件添加执行权限；
  3. 添加Java的环境变量至Tomcat配置文件setclasspath.sh，写入内容如下：

*#---------------------*

*# java path*

*#---------------------*

JAVA\_HOME=/usr/lib/jvm/java-8-oracle

JRE\_HOME=$JAVA\_HOME/jre

JAVA\_BIN=$JAVA\_HOME/bin

CLASSPATH=$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib

PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin

export JAVA\_HOME JRE\_HOME PATH CLASSPATH

* 1. 执行 sudo sh configtest.sh检查配置是否正确；
  2. 执行 sudo ./startup.sh启动Tomcat服务，在浏览器中[http://localhost:8080](http://localhost:8080/)，如果显示Tomcat主页，则表示安装配置成功。

1. 安装MySQL数据库:
   1. 安装命令：sudo apt-get install mysql-server mysql-client。
   2. 配置：安装过程中按照提示设置账户和密码。
   3. 检查方式：执行mysql –u 账号 -p命令，查看MySQL是否安装成功和账户设置是否正确。
   4. 注意事项：无。
2. 安装phpMyAdmin数据库管理
   1. 安装命令：sudo apt-get install phpMyAdmin
   2. 配置：安装过程中会要求选择apache2或lighttpd作为Web Server，选择apache2。将phpMyAdmin与apache2建立连接，以本文为例：www目录在/var/www，phpMyAdmin在/usr/share/phpMyAdmin目录，使用命令sudo ln -s /usr/share/phpMyAdmin /var/www/html建立链接。
   3. 检查方式：在浏览器地址栏中打开[http://localhost/phpMyAdmin](http://localhost/phpmyadmin)，如果成功显示登录界面则表示安装成功。
   4. 注意事项：无。

### 测试方案

测试步骤如下：

1. 通过管理系统录入测试厂家、产品、设备等信息；
2. 两个Android手机各自注册一个账号并登录，称为账号A和账号B，之后的步骤中终端A、B都是表示已经分别登录账号A、B的Android手机，终端的相关操作即对账户的操作；
3. 终端A向终端B，终端B同意好友请求或拒绝好友请求；
4. 建立好友关系后，终端A绑定一个传感设备，并将此设备分享给终端B；
5. 终端A连接共享的传感设备进入设备主界面，观察终端B设备列表对应此设备的状态；
6. 终端A控制传感器采集数据，并观察数据测量上下限和警报值；
7. 终端B也进入设备主界面，观察采集到的数据
8. 终端A把设备权限移交给终端B；
9. 同样执行第6步，但终端B主动操作传感器采集数据；
10. 终端A删除设备，观察终端B设备状态；
11. 终端A执行反馈，检查更新等操作；
12. 在后台管理系统中，查看用户、设备等信息是否正常；
13. 查看App Socket服务器日志是否正常。

### 测试结果及分析

测试结果主要从两个方面判断，一是通过终端的与服务器的交互直观的判断，另一个则是通过App Socket的日志记录判断。由于前者不易量化，所以下面通过几个重要的日志记录来展现测试结果并分析。

每天保存的日志文件如图 5.1所示

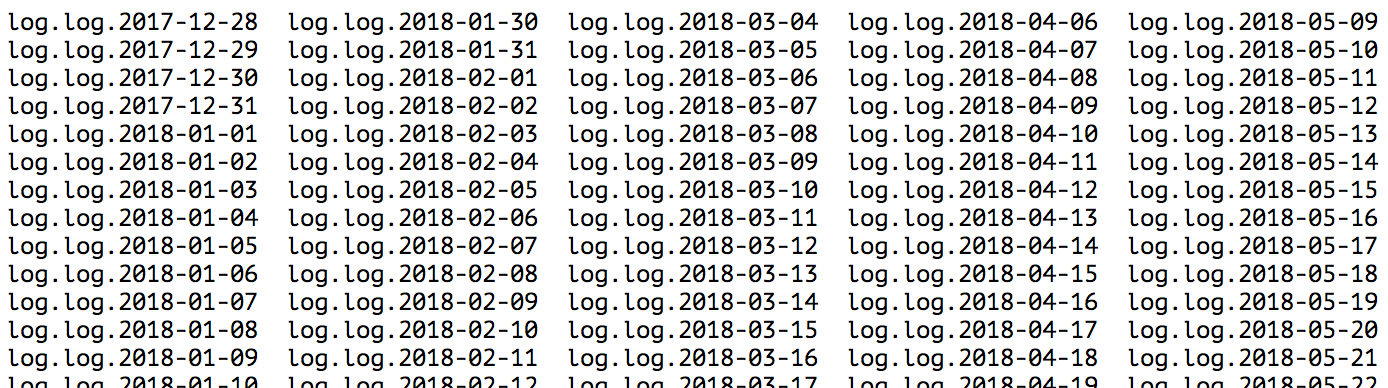


图 5.1 日志文件图

1. 设备注册连接

如图 5.2 所示，ssid为a47174002b4a7343的终端注册连接。

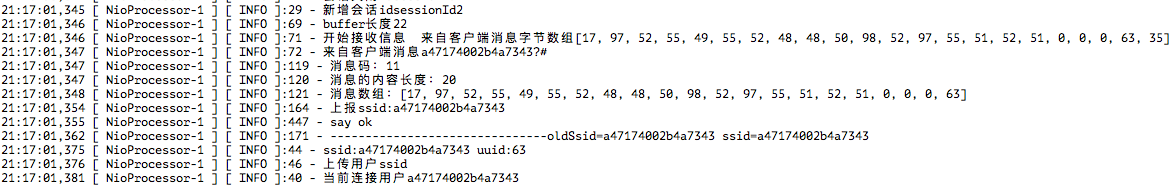


图 5.2 终端注册连接日志图

如图 5.3所示，physicalId为209148B494FE的设备上线。

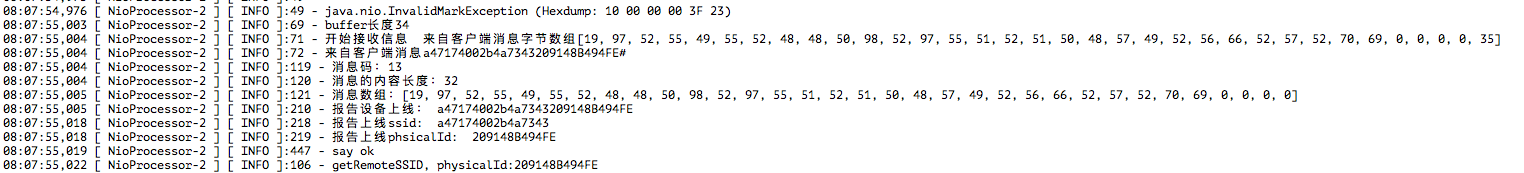


图 5.3 传感设备上线日志图

1. 设备主人删除设备，App Socket服务端群发通知。



图 5.4设备删除群发消息日志图

1. 两个终端间通过App Socket服务端透传消息。

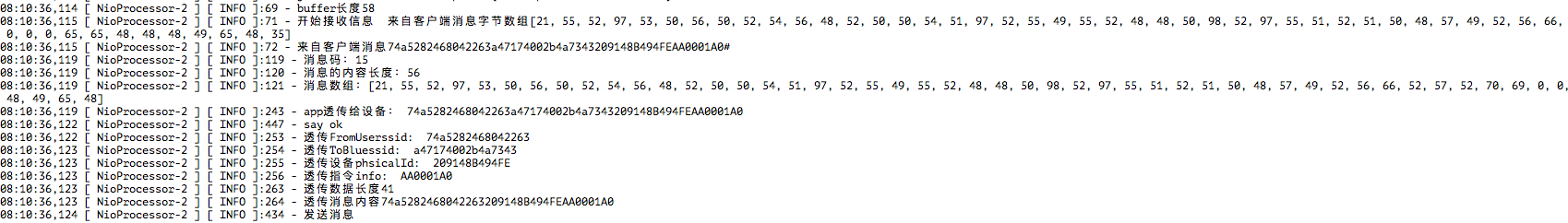


图 5.5消息透传日志图

1. 账户在另一个终端上登录，App Socket服务端通知前一个登录设备下线。

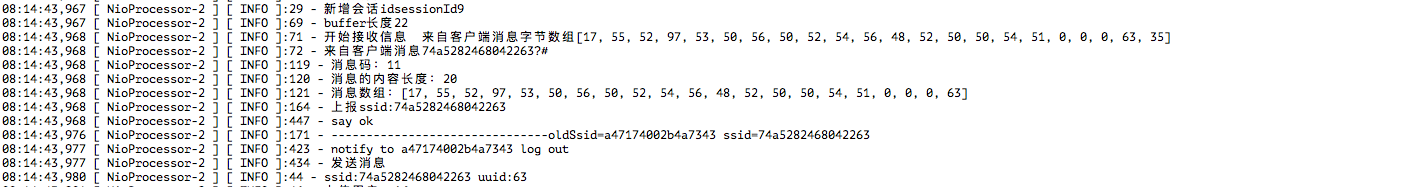


图 5.6单设备登录日志图

以上从App Socket服务端日志的可以分析出，系统整体运行良好稳定，因此可以证明本文所设计实现的移动互联网智能工具云平台可具体运用于实际生产生活。

# 结束语

本文响应“中国智造”和“工业4.0”发展战略，针对传统PC互联网时代智能工具成本高、移动性低、可维护性差等问题，结合移动互联网所带来的优势，提出移动互联网智能工具系统这一概念。移动互联网智能工具系统中使用移动终端作为传感器的“大脑”和传输媒介，降低了智能工具的成本，提高移动性和可维护性。然后聚焦于移动互联网智能工具系统中云平台的设计和实现。

本论文的主要工作如下：

第一，介绍了智能工具概念、特点和发展现状等。重点介绍了智能工具在传统PC互联网中的不足，以及移动互联网浪潮下的解决方案的构思和优点。

第二，介绍了ThinkPHP框架和几种IO模型原理，分析采用同步非阻塞IO和IO多路复用的优势。最后对基于Java NIO的MINA框架做了简要的概述和工作流程分析。

第三，对云平台的三个模块：后台管理系统、App HTTP端、App Socket端的设计进行了介绍。

第四，对云平台的三个模块：后台管理系统、App HTTP端、App Socket端的实现做了详细的阐述。

第五，从功能上对云平台各个模块进行测试，分析测试结果。

总之，本文完成了基于移动互联网的智能工具云平台的设计与实现。

# 致 谢

时光荏苒，回首大学四年，一路走来，实属不易，在此我要感谢学校老师对我的培养，感谢同学在学习和生活上的陪伴，有了他们我才在学业上有所进步，在生活上有所成长。

从题目初定，到项目实现，再到论文定稿的完成，这其中有挫折也有收获。我要感谢我的导师李超老师，在他的指导下我才能顺利的完成这次毕业设计，此外也要感谢董艾华老师在论文撰写过程中对我辅导。

另外要感谢共同参与项目的伙伴，整个移动互联网智能工具系统，如果没有他们的奋斗，便无法实现。

最后，我要特别感谢我的父母与姐姐，是他们一直以来的关心与支持让我有了前进的动力。再次向所有帮助过我的人致谢！

# 参考文献

1. Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions[D]. Australia:Department of Electrical and Electronic Engineering, 2012. 10-23
2. Kento Sato, Naoya Maruyama, Kathryn Mohror. Design and modeling of a non-blocking checkpointing system[D]. USA:IEEE, 2012. 13-27
3. Andy Oram, Maria Gulick. Linux System Programming: Talking Directly to the Kernel and C Libray[M]. US:O'Reilly Media, 2013. 51-214
4. William PughJaime, Spacco. MPJava: High-Performance Message Passing in Java Using Java.nio[M]. US:Springer, 2004. 4-50
5. R Fielding, J Gettys, J Mogul, H Frystyk, L Masinter, P Leach, T Berners-Lee. Hypertext Transfer Protocol -- HTTP/1.1[EB/OL]. https://www.rfc-editor.org/rfc/pdfrfc/rfc2616.txt.pdf.
6. Craig Hunt. TCP/IP Network Administration[M]. US:O'Reily, 2002. 423-460
7. Daniel, Bovet, Marco Cesati. Understanding the Linux Kernel: From I/O Ports to Process Management[M]. US:O'Reily, 2005. 35-92
8. Dengshan Tian,Jiwen Wen,Yin Liu,Ning Ma,Huiping Wei School of Economics and Management,Beijing Forestry University,100083,Beijing,China;[A Layered WEB Framework Model](http://cpfd.cnki.com.cn/Article/CPFDTOTAL-TJGY201012002028.htm)[A];[C];2010
9. Cesare Pautasso, Olaf Zimmerman, Frank Leymann. Restful web services vs. "big"' web services: making the right architectural decision[J/OL].

https://dl.acm.org/purchase.cfm?id=1367606.2008

1. David Cochran. Twitter Bootstrap Web Development How-To[M]. UK:Packt Publishing Ltd, 2012. 32-77
2. Addy Osmani. Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide[M]. US:O'Reilly, 2012. 234-312
3. Chi-KwongChan, L.M. Cheng. Cryptanalysis of a Timestamp-Based Password Authentication Scheme[J]. Computers & Security, 2001, 21(1): 74-76
4. D Crockford. The application/json Media Type for JavaScript Object Notation (JSON)[EB/OL]. https://tools.ietf.org/html/rfc4627.
5. Zhanjie, Wang Xiao, Li. A New Real-Time Heartbeat Failure Detector[M]. China:IEEE, 2008.
6. 向运. 基于Apache Mina的智能家居服务器设计与实现[D].陕西：长安大学,2013.
7. 郭伟. 基于MINA和SSH的智能家居管理平台的设计与实现[D].广东：华南理工大学,2015
8. 暴志刚, 胡艳军, 顾新建. 基于Web的系统权限管理实现方法[J]. 计算机工程, 2006, (1): 169-170
9. 徐进辉, 徐明. 数据库事务处理模型研究[J]. 计算机工程与科学, 2004, (4): 62-65
10. 林永成. 社交网络机器用户甄别技术研究与应用[D]. 中国:吉林大学, 2013.
11. Kyle Rankin, Benjamin, Mako Hill. The Official Ubuntu Server Book[M]. US:Creative Commons Attribution-ShareAlike, 2013.