直播间重构设计文档

|  |  |  |  |
| --- | --- | --- | --- |
| 当前版本： | V 1.0 | 归属部门/项目： | 比邻项目组 |
| 编写人： | **李斌** | 创建日期： | 2018-06-13 |

# 概述

## 1.1系统简介

直播间服务主要用于比邻用户的直播功能，主要涉及用户开播，上下麦管理，用户列表管理，公屏聊天等功能。

## 1.2项目背景

现有的直播间服务并不是单独闭环的服务，基本上每个请求都会通过java提供的thrift接口去查询，上报数据。当前这种模型对外部服务有较强的依赖，并且直播间内的数据信息也较为分散，为了满足后续的各种需求，我们需要把直播间服务做成一个比较内聚的服务，尽量减少和外部服务的耦合。

## 1.3目的

重构完成的直播间能独立处理用户开播，上下麦，push通知等功能，并能将直播间内的数据提供出来供其他服务获取。

## 1.4设计原则

本次重构并不改变现有的功能，也不影响当前版本客户端的业务流程，一切在后台修改。

# 架构设计

## 2.1整体架构

### 2.1.1现有的服务器分层



现有的直播间接入层有三个，其中CppProtocol是老版本的长连接服务器，为后面的C++服务接入用户消息，bilinServer是一个http的接入层，供用户开播时查询信息，开播鉴权等处理。Mars是新版本的接入层，为后面的Go服务接入用户消息。

现有的服务层有Go BcServer，C++的bcServer，java服务，其中前两个是用于频道内用户的各种消息处理，java服务用于控制频道内的用户鉴权，上下麦控制，信息查询等。

当前的架构下，频道内每个直播间内的操作都需要经过java，导致耦合非常严重，如果java服务暂时不可用，会直接影响整个直播间的服务。

### 2.1.2重构后的服务分层



如上图所示，重构后的直播间服务完全成为一个内聚的服务，所有直播间的消息都发到该服务上，BcServer用于处理频道内的各种逻辑，RoomService用于存储频道内的业务数据，以及提供对外数据查询。

由于当前客户端开播鉴权还是走的http接口，所以java鉴权服务继续保留，用于开播鉴权审核，当审核通过时，把消息流转给直播间服务。

### 2.1.3直播间服务架构



上图所示是重构后的直播间服务架构，重构后的直播间服务是一个高内聚的服务，业务逻辑放在一个“新直播间服务”进程内，数据抽象出一个Room Service服务，新直播间服务直接负责客户端的所有直播间相关的消息接入，其中：

* 老版本App（version<=4.5）的用户继续原来的逻辑，通过java进行开播鉴权，然后从C++ CppProtocol接入直播间服务。
* 当前版本（version 4.6 ~ 5.1）的用户一样，通过java进行开播鉴权，然后通过mars协议接入直播间服务。
* 重构过渡期间，可以继续使用java的http通道进行鉴权，鉴权之后的消息发给新直播间服务即可。
* 重构完成之后，老版本的客户端遗留的一些使用java直播控制服务的接口，比如joinhotline等thrift接口，继续保持，java从RoomService后去查询直播间数据。
* 重构之后的版本，将把用户开播鉴权和直播间内的控制流程全部包含进来，APP客户端做一下对应修改，开播请求走Mars通道即可。

## 2.2直播间服务流程设计

### 2.2.1初始化流程



新的直播间服务初始化流程如图，仅仅依赖于RoomService，进程启动之后即开始服务。

### 2.2.2用户进频道流程

1. 老版本（version<=4.5）用户进频道



用户先通过http请求鉴权处理，鉴权完成后进直播间，通过查询java的thrift接口获取信息。

当前版本（version = 4.6 ~ 5.1）跟老版本逻辑一直，只是C++直播间改为Go的服务器。

1. 重构后的版本用户进频道流程



老的客户端还是走Http的鉴权流程，鉴权成功后再进入直播间，统一不走java thrift接口，改为直播间内部服务的tars接口。

1. 新版本客户端进频道流程



由上图可以看出，新的版本发布后，直播间所有相关操作均通过bcserver完成。

### 2.2.3用户上下麦操作



新版本？是否沿用该流程？？？

### 2.2.4主持人退出房间



老版本主持人退出房间需要通知java，因为java需要显示主持人直播状态。

新版本主持人退房间统一在bcserver处理。



## 2.3官频流程设计

### 2.3.1初始化流程

### 2.3.2后台配置官频

### 2.3.3服务器处理官频逻辑

1. 老版本官频逻辑



老版本的官频处理如图所示，java有个定时器，去扫描全局官频数据，查找排挡时间到了的房间

1．如果当前主播在麦上，则不做处理

2．如果当前麦上有人，则需要给当前麦的用户发push消息，并且给切档的主播也发push

3. 等待1分钟之后会调用java thrift接口发送切档消息，当C++成功切档后会返回状态给java，java再更新db数据。

老版本的逻辑中，一次切档的流程会有多次thrift调用和回调处理，目前已知的问题是，有时间到了之后java的切档消息没有发，用户无法正常换挡。并且java的逻辑比较依赖C++提供的主播状态，如果某个主播状态不对（调用thrift失败），则会影响官频的后续操作。

1. 重构后官频逻辑



重构后的官频与java解耦，直接从db中遍历数据，拿到排挡主播数据之后开始发push，通知用户上下麦。

### 2.3.4数据一致性问题



Bcserver有多个，并且是没有优先级的，从db中取数据，需要保持数据的一致性，避免发送重复的消息。

## Room Service

提供一下房间信息服务查询，包含房间基本信息，当前麦上嘉宾，用户列表等信息。

由于直播间业务均用tars这套，所以直播间控制相关的信息也存在这里，包含但不限于踢人消息，被禁言，禁音，直播间用户特权信息等。

# 3协议设计