视频会议系统架构设计

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： |  |
| 作 者： |  |
| 完成日期： |  |

# 目录

[目录 1](#_Toc403982089)

[1. 简介 2](#_Toc403982090)

[1.1. 目的 2](#_Toc403982091)

[1.2. 术语 2](#_Toc403982092)

[1.3. 导读 2](#_Toc403982093)

[2. 软件架构 2](#_Toc403982094)

[2.1. 需求规定 2](#_Toc403982095)

[2.2. 整体结构 3](#_Toc403982096)

[2.3. 会议开始时序图 4](#_Toc403982097)

[2.4. 语音接入时序图 5](#_Toc403982098)

[2.5. 转发服务与会议服务之间的同步 6](#_Toc403982099)

[2.6. 多会议服务之间的同步 8](#_Toc403982100)

[2.7. 会议服务处理加入会议的缓存的流程 10](#_Toc403982101)

[2.8. 登陆会议服务的授权过程 11](#_Toc403982102)

[3. 服务端缓存定义 11](#_Toc403982103)

[3.1. 服务信息缓存 12](#_Toc403982104)

[3.2. 房间分配信息缓存 12](#_Toc403982105)

[3.3. 会议信息缓存 12](#_Toc403982106)

[4. 部署视图 13](#_Toc403982107)

[4.1. 服务进程部署 13](#_Toc403982108)

[4.2. 网关部署 14](#_Toc403982109)

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的视图来描述软件系统的各个方面，记录并表述已对系统的构架方面作出的重要决策。

## 术语

|  |  |
| --- | --- |
| 服务 | 表示一个提供服务的进程 |
| 会议服务 | 负责会议室创建与销毁、会议成员管理的服务进程 |
| 转发服务 | 负责多会议室音频转发的服务进程 |
| 服务器 | 表示容纳服务的机器，一个服务器中会有一到多个服务 |
| NodeID | 每个服务都有的属性，用作服务器的唯一标识，int32型 |

## 导读

1. 多会议室转发设计（**2.4节**、**2.5节**）
2. 服务器单点故障处理（**2.5节**、**2.6节**、**2.7节**）
3. 授权（**2.8节**）
4. 部署（**4.1节**、**4.2节**）

# 软件架构

## 需求规定

见《网络会议开发需求.xlsx》

1、99U中集成试用demo

2、多会议室音频UDP转发传输开发

3、服务单点故障处理

## 整体结构



如上图所示，整个会议系统由会议服务、白板服务、转发服务、Redis缓存系统、会议邀请系统和会议客户端等6个部分构成。各个部分的功能如下：

1. 会议服务，负责会议室的创建与销毁、与会人员的管理、合理分配服务器资源等。
2. 白板服务，负责在会议过程中透传同会议室的白板共享数据。
3. 转发服务，负责在会议过程中透传同会议室的音频数据。
4. Redis缓存系统，所有的服务器都高度依赖的缓存系统，用于服务器之间的信息同步。
5. 会议邀请系统，负责会议室的创建和会议通知的发送，现有91U的V2会议邀请系统。与91U集成时，可复用这一部分进行改造。
6. 会议客户端，负责会议客户端的所有功能，包括各协议的发送和解析、会议信息的展示、白板的展示、语音的采集、发送和播放，同时要考虑断线重连等异常情况的处理。客户端将会**对外提供SDK**，使整个会议系统的界面可以随意改造。

需要**特别说明**的是，以上各部分中，1~4都可以有多个实例来共同承载大量客户端的并发访问。即可以有多个会议服务、多个转发服务和多个白板服务，而Redis作为一个成熟的缓存系统也已有成熟的集群部署方案。下面我们以时序图的方式，描述各个服务之间是如何协调进行服务的。

## 会议开始时序图

描述会议开始的过程中，会议邀请系统、客户端和会议服务之间的主要通信过程。消息的整体时序图如下：



## 语音接入时序图

描述了客户端如何通过会议服务器与转发服务取得通信。

在一个会议创建的时，会议服务器会分配一台转发服务器给当前会议（具体过程见**2.5节**）。当一个客户端加入会议时，会议服务器返回RoomID（会议室ID，**uint32型**）、UserID（用户ID，**uint32型**），以及转发服务器的“IP地址和端口”给客户端。客户端得到这些信息后，即可通过转发服务器的“IP地址和端口”与转发服务器取得通信联系，同时，客户端会将自己的RoomID和UserID转给转发服务器。

转发服务器会通过客户端传上来的RoomID和UserID建立一个转发表，实时转发相同RoomID下的音频。例如RoomIDA下有UserIDA、UserIDB和UserIDC的音频发送上来，对于UserIDA，转发服务器会将UserIDB和UserIDC和音频转发给它；对于UserIDB，转发服务器会将UserIDA和UserIDC的音频转发给它；对于UserIDC，转发服务器会将UserIDA和UserIDB 的音频转发给它。

整体时序图如下：



## 转发服务与会议服务之间的同步

本节描述了会议管理服务如何与多个转发服务之间进行必要信息的同步，以及某个转发服务故障后的处理过程。

正常情况下，转发服务每1秒钟向Redis中写入本服务的信息，信息包括NodeID(服务器编号)、IP、端口、在线人数、当前时间等。与此同时，会议服务运行后，会定时从Redis中获取可用的转发服务信息保存在本地内存中，当一个会议创建后，分配其中一个转发服务给此会议室使用。

当会议室服务发现已知的转发服务中，有某一个服务超过10秒（可配置）未更新它的信息，则会认为该转发服务不可用，不再将它分配给新建的服务。同时，对于已经分配了该服务的会议室，它会重新分配一个可用的转发服务给它们，并即时通知相关的客户端进行服务的切换。

整体时序图如下：



## 多会议服务之间的同步

本节描述了多个会议服务之间的同步工作过程 ，以及某个会议服务故障后的会议恢复处理过程。

所有会议服务都会每1秒向Redis中写入本服务的信息（NodeID、IP、端口、在线人数、当前时间），如果不能写入Redis，则会议服务会将自己退出。同时每1秒(暂定，为了保证及时性)从Redis中获取所有会议服务的信息。这样，所有的会议服务都拥有其它会议服务的信息。

客户端在收到会议邀请后，在进入会议室之前，需要向任意一台（这些服务的地址使用域名解析或者保存在客户端本地配置）会议服务请求需要连接的会议服务（的IP和端口），请求时需要带上会议室的标识（RoomUrl）。

会议服务在收到请求后，根据RoomUrl从Redis缓存中查询是否已经存在此会议的分配信息（**见3.2节**），如果有则检查分配信息中的会议服务是否是可用的，如果没有，则根据各会议服的负载，分配一台会议服务器给当前会议，并将这个分配信息写到Redis缓存中。特别说明，这个分配过程使用Redis事务保证整个过程的是原子的。

当某个服务超过5秒（暂定）未向Redis中写入信息时，所有的会议服务都会检测到，并认为这个会议服务不可用了。这时，所有的连接在这个会议服务上的客户端都会掉线，掉线后，客户端会重新向任意一台会议服务请求需要重新连接的会议服务，并回到(加入)当前正在进行的会议中。**2.7节**会提到会议服务是如何保证在这种异常情况下会议的正常进行的。



## 会议服务处理加入会议的缓存的流程

描述了加入会议过程中，会议服务对Redis缓存所做的相关工作。从中可以看出单台会议服务故障后是如何保证会议过程不中断的。

在会议服务中，我们除了将会议信息保存在本地内存中，同时也会保存在Redis缓存中，这主要用于灾难恢复。当服务发生故障后，本地内存中的会议信息会丢失，待服务恢复正常后，可以从Redis缓存中重新恢复会议的信息，以此保证会议的正常进行。

例如服务A故障，我们的系统经过**2.6节**中所描述的过程进行处理后，原先连接在服务A的用户全部重新连接到B服务。此时B服务是没有A服务中的会议信息的，这时B服务便会从Redis中去查找相应的会议信息并保存到本地内存中，这样会议的一些基本信息就不会丢失，会议任然可以正常进行。



## 登陆会议服务的授权过程

我们的系统将会以单点登陆“101帐号中心”的方式进行授权。整体架构如下：



如图所示，整个登陆授权过程由以下几个部分构成：

1、101帐号中心，公司的帐号认证中心，对外接供了帐号认证接口。

2、授权接口，授权接口封装了授权所需要的方法，供会议客户端和会议服务器调用。客户端和服务器可以使用任何一个实现了此接口的授权模块。客户端在向会议服务器请求授权时，会在协议中指明使用哪种授权方式（如101授权、不授权或其它授权方式），会议服务器会根据协议使用不同的授权模块进行验证。本次我们将会采用101授权模块。

3、101授权模块，101授权模块通过调用101帐号中心的提供的接口，实现了授权接口中的方法。

# 服务端缓存定义

本节描述了**第2节**中所提到的几个缓存结构。

## 服务信息缓存

缓存结构为Redis: Hash

|  |  |  |
| --- | --- | --- |
| **Keys** | **HKeys** | **HValue** |
| whiteboard\_servers |  |  |
| transfer\_servers |  |  |
| meetings\_servers |  |  |
|  | Node1 | {"ip":"192.168.181.11",//服务地址  "port":6666, //服务端口  "usernum":30, //当前人数  "cpu":70,//cpu占用率  "priority":1,//分配优先级  "lastactive":1300000000 //上次写入时间，**规定各服务每1秒将信息写入缓存**，如果超过**一定时间**没有写缓存，则认为此服务不可用  } |
|  | Node2 | … |
|  | Node3 | … |
|  | Node4 | … |

## 房间分配信息缓存

结构为Redis:String

|  |  |
| --- | --- |
| **Keys** | **HValue** |
| meeting\_map\_roomurl1 | {"msaddr":{"nodeid":30,"ip":"192.168.57.119","port":6060}} |
| meeting\_map\_roomurl2 | {"msaddr":{"nodeid":30,"ip":"192.168.57.119","port":6060}} |
|  | … |

## 会议信息缓存

结构为Redis:Hash

|  |  |  |
| --- | --- | --- |
| **Keys** | **HKeys** | **HValue** |
| mi\_roomurl1 | roomid | 100 |
| manager | sip:sip:manager@127.0.0.1 |
| meeting\_server | {"nodeid":30,"ip":"192.168.57.119","port":6060} |
| transfer\_server | {"nodeid":30,"ip":"192.168.57.119","port":6060} |
| wb\_server | {"nodeid":30,"ip":"192.168.57.119","port":6060} |
| mi\_roomurl1 | roomid | 100 |
| manager | sip:sip:manager@127.0.0.1 |
| meeting\_server | {"nodeid":30,"ip":"192.168.57.119","port":6060} |
| transfer\_server | {"nodeid":30,"ip":"192.168.57.119","port":6060} |
| wb\_server | {"nodeid":30,"ip":"192.168.57.119","port":6060} |
| … | … | … |

## 会议成员缓存

结构为Redis:Hash

|  |  |  |
| --- | --- | --- |
| **Keys** | **HKeys** | **HValue** |
| members\_roomurl1 |  |  |
|  | memurl1 | {"mute":"yes/no","memuid":101} |
|  | memurl2 | … |

# 部署视图

## 服务进程部署

暂定使用3台机器，假设为机器A、机器B和机器C。如果有条件，可将机器A和机器B置于一条线路，将机器C置于另一条线路。

1. 会议服务：分别在机器Ａ和机器Ｃ上部署一个会议服务，从**2.6节**可看出，会议服务本身具有负载均衡的策略。
2. 转发服务：在机器B上部署两个转发服务，在机器C上部署一个转发服务，会议服务在进行分配时，会优先分配机器B上的转发服务，机器C上的转发服务只作为备用，以免影响位于机器C上的其它服务。
3. 白板服务：分别于机器A和机器C部署一个白板服务，会议服务会将会议平均分配到不用的白板服务上。
4. Redis：在机器A和机器C上部署Redis主从复制结构，并部署Redis Sentinel对主从结构进行监控，当主服务器出问题时，Redis Sentinel会自动进行主从切换，当主服务器恢复后自动成为从服务器。
5. Nginx：我们会在不用的线路上部署多个Nginx进行端口转发，不同网段的用户会被路由到相应线路的Nginx上，见**4.2节**。每个Nginx都与所有的后端的服务相连，不同的服务对应Nginx上不同的端口，客户端根据不同的端口可以路由到对应的服务上去。例：假设我们的域名为mcu.99.com， mcu.99.com:6080对应会议管理服务A，mcu.99.com:6081对应会议管理服务B，mcu.99.com:6091对应白板服务A。

整体部署视图如下：



## 网关部署

