设备端连接存活状态上报功能方案设计

概述

设备端连接存活状态上报功能的场景是，当APP端的播放请求到达时，业务端先判断一APP所请求的设备是否已经连接到视频服务器上，如果设备已经连接到服务器上，业务端直接下发播放链接给APP；如果设备没有连接到服务器，业务端按完整的流程让设备端先连接到视频服务器上，再下发播放链接给APP；基于这种场景，业务端需要知道设备端的连接存活状态。

基本方案

设备端的连接存活状态按以下格式写到redis服务里。所有写到redis里的值都带上120秒的超时时间，如果在120秒内这条记录没有更新，记录将自动失效。

> *feed_id=<公网IP>*

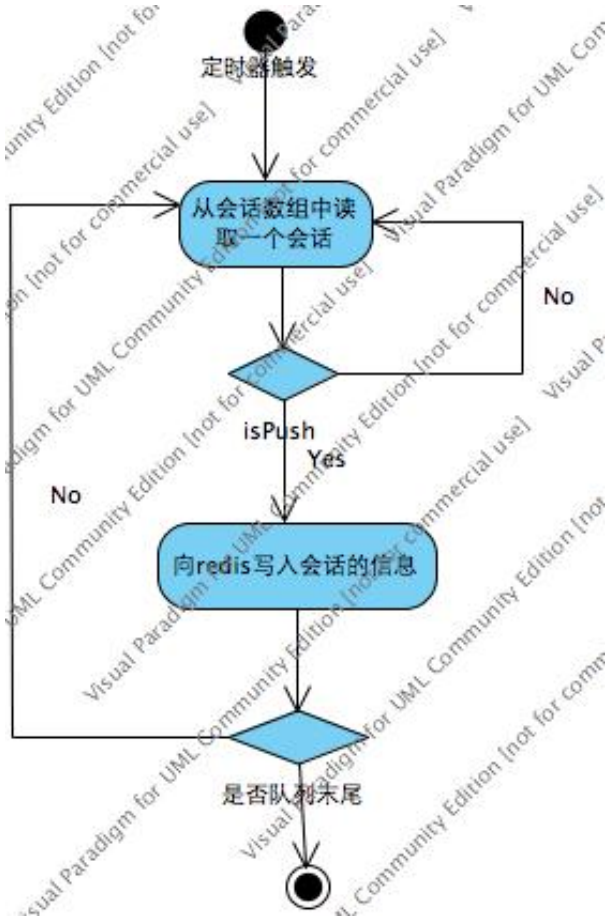
1. feed_id作为key；
2. 值是所在服务器的VIP分配的IP地址；

方案一

目前已经知道darwin服务器有保存当前会话的数组，只是没有标识哪个会话是推流，哪个会话是拉流，方案一是在数组里保存的会话上增加会话的feedid和标识会话是推流的标志，通过一个心跳线程轮询的方式把推流的会话的存活状态定时发送到redis。

当会话接收到断开连接的事件时主动删除redis里的值。

业务流程图：



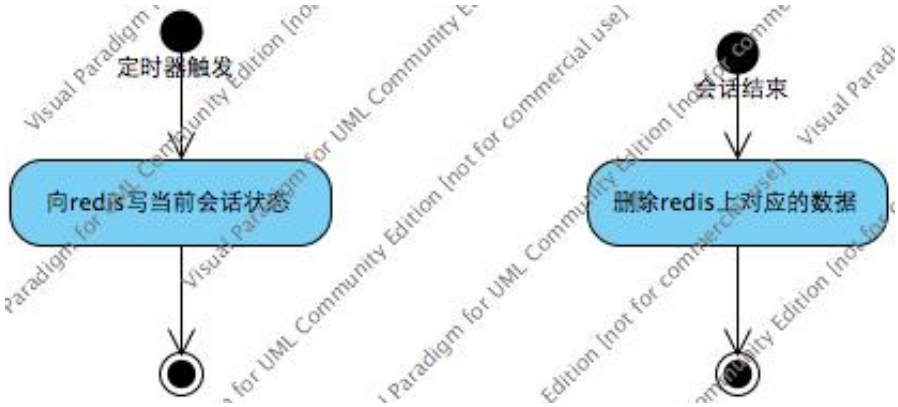
以上方式的优点是利用已经有的会话的数组，而且只需要一个心跳线程，总体的资源开销小。

缺点是，这个数组的更新需要时间，不能确保数组里的会话的状态是实时的，可能导致业务端下发的url是无效的。

方案二

把feed_id和服务器的IP地址都保存到会话的上下文中，在设备端的会话建立的时候上报存活状态到redis，在会话的存活期间定时更新这条数据，当会话结束的时候（不论是正常结束还是异常结束，但是不包括服务程序崩溃）改变状态。

业务流程图：



这个方式的好处是对单个会话的状态进行实时监控，状态可以实时的上报，实时性很好。

缺点是当服务器异常崩溃的时候，系统必须等待redis中的数据过期，在过期之前业务端会得到错误的状态，导致APP不能正常的连接，并且需要在会话内建立心跳线程，所以当并发数较大时资源的开销会比较大。

结论

内部讨论后，以darwin的资源消耗为优先考虑，首先选择方案一，需要在darwin里插入一个心跳线程，以轮询会话数组的方式来写redis，达到上报会话状态的目的，业务端也需要针对会话状态的上报作相应的修改