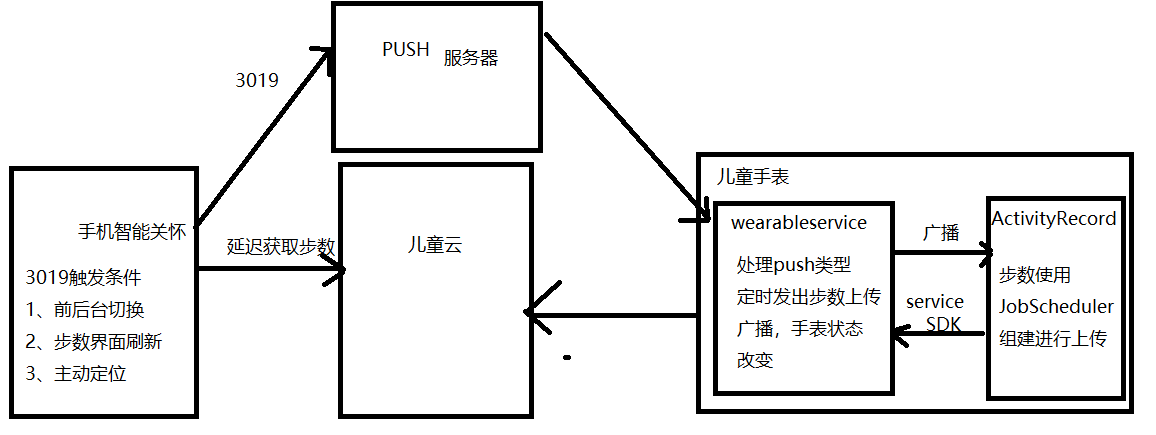
### 问题描述

在运动关怀界面显示小孩处于运动状态，但是在只能关怀步数界面刷新多次不见步数改变

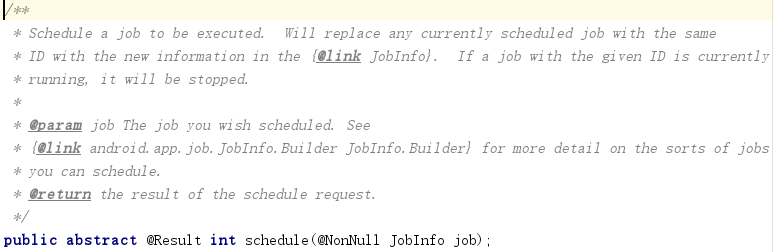
### 问题分析

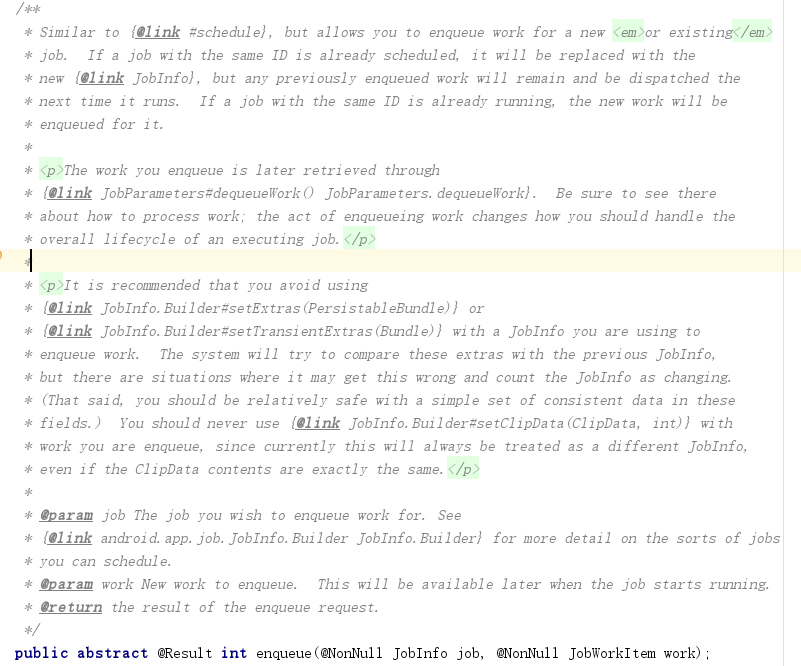
1. 首先和智能关怀同事沟通，最终得出步数上传的流程如下：

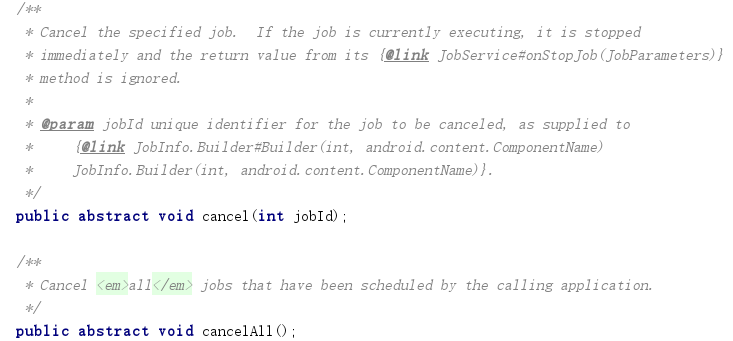


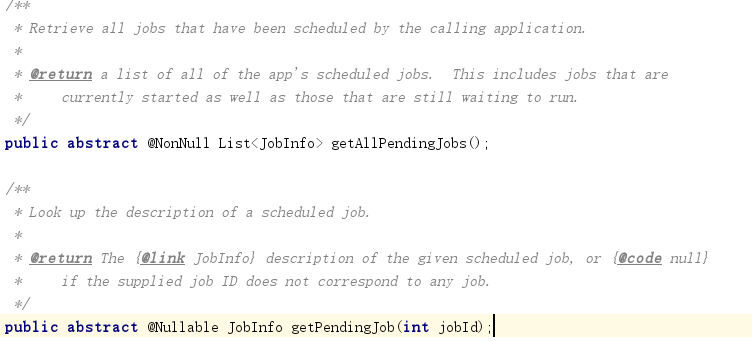
从上图可以看出，上传步数的原理为：从手机端的智能关怀通过主动定位、步数界面刷新、应用前后台切换会触发Push心跳，类型为3019，通过推送的方式通知手表中的wearableservice组件，然后通过广播跨进程通知运动模块发起上传步数的任务，然后通过wearableservice统一将数据发送给儿童云。最后手机侧延迟几秒钟从儿童云获取最新的儿童步数。从而达到步数刷新的操作。

1. 从上面的原理我们看出在该系统中存在一定的延迟是可以接受的，但是用户提供的视屏和log发现他进行了多次刷新，都没有看到步数改变，直到后面步数获取成功经历了2分钟左右。
2. 通过用户提供的儿童表日志，搜索3019以及wearableservice发出的广播action，可以看出用户发起了很多次，并且在ActivityRecord当中也受到了广播启动了JobScheduler，但是没有成功的上传步数。
3. 因此可以断定ActivityRecord中使用JobScheduler存在问题或者JobScheduler不适合在该系统中使用。
4. 与用户沟通和用户提供的bug等情况，综合分析用户期望的是他看到手表处于运动状态，那么刷新步数就会有少许步数增加，而多次刷新是为了看出动态变化。因此发送多个广播和push是合理情况。
5. 通过研究JobScheduler源码如下：



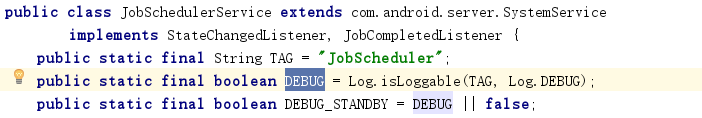






从上面源码可以看出JobScheduler为我们提供了两个执行的方法、两个取消的方法、两个获取任务的方法。

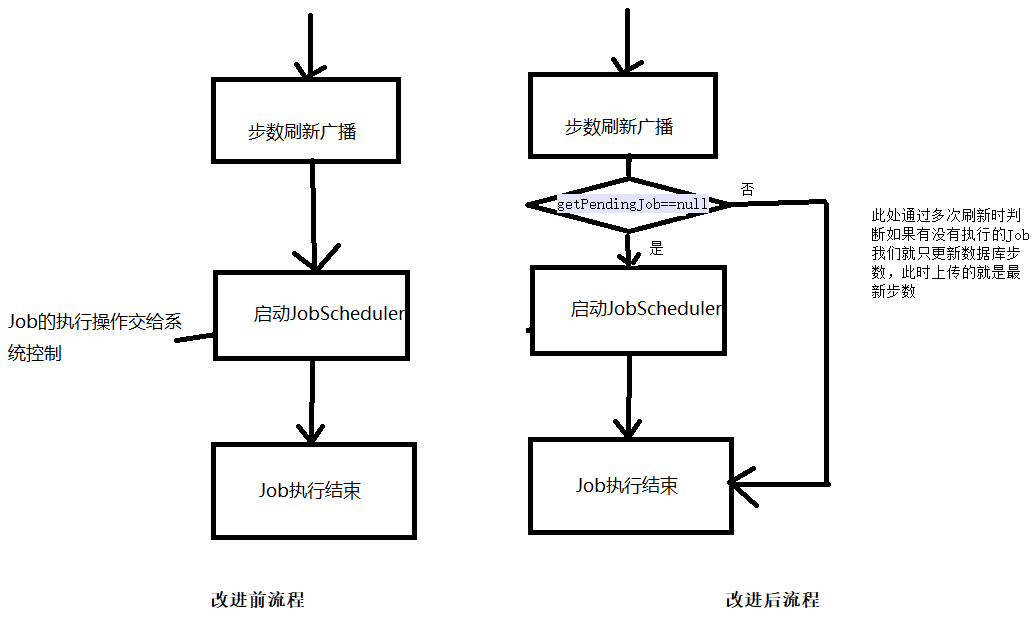
1. 再来看下我们自己的实现，发现是使用了schedule方法，根据此方法的源码和解释我们可以知道该方法会先根据jobId判断上一个job的执行状态情况进行逻辑，如果上一个job没有执行完 ，那么就会Cancel掉上一个job然后重新创建一个Job。
2. 根据用户的操作非常频繁的刷新步数，我们就可以大胆猜测，是因为上传步数需要时间，用户频繁刷新导致上一个Job没有完成的情况下被新的Job替换了，但是新的Job也没有执行成功。
3. 有了上述猜测，本人修改了JobSchedulerService中的日志打印，将Service的日志都打印、并编译services.jar通过adb push到系统的system/framework目录下，并重启手表，重新抓了分日志



10、通过最新日志，在相同操作情况下，的确存在Job被不停移除的情况，因此此上传系统中的问题已经明了。就是因为使用JobScheduler不当导致用户在频繁操作的时候数据一致无法上传。

### 优化方案

1. 根据上述的分析结果，JobScheduler提供的schedule和enqueue方法最适合该场景的是enqueue方法，他能够将所有Job都一一执行。
2. 但是呢，在该系统中用户期望看到的是最新的步数，也就是说没必要将所有Job都上传一边，如果每一个都上传对服务器、用户流量、手表功耗都有一定影响。
3. 因此我们还要采用schedule，但是我们还不能让我们的Job被Cancel，因此继续分析源码我们发现还提供了方法getPendingJob，该方法可以通过JobId获取到当前队列中是否存在同一个还没有执行成功的Job，那这个有什么用呢？看下图：



综上图：该流程优化后对功耗、用户流量、体验有很大的提升，此处的思想是步数更新单独更新，上传Job在执行过程中去实时获取最新数据库步数，此方法并且保证了至少每次刷新都会有最新数据被上传，而不是像原有系统一样为了上传最新步数导致Job不停被取消。