# [android 6.0 Doze/App Standby模式对push的影响调研](https://wiki.sankuai.com/pages/viewpage.action?pageId=559195133)

* 创建： 未知用户 (haitao.li)，最新修改于： [八月 10, 2016](https://wiki.sankuai.com/pages/diffpagesbyversion.action?pageId=559195133&selectedPageVersions=11&selectedPageVersions=12)

从Android 6.0 (API level 23)开始，Android引入了两种节省电量的新特性：Doze和App Standby。通过限制app的行为，来达到省电的目的。只要app运行在6.0及以上的系统上，都会受到这两个特性的限制，无论app编译时的target API是不是23。

 一、Doze

如果设备未连接电源，则当屏幕关闭一段时间后，系统就会进入Doze模式。在Doze模式下，app无法访问网络，app的jobs、syncs以及standard alarms会被推迟执行。  
然而，系统会周期性地退出Doze模式，进入短暂的活跃状态（maintenance window），来继续app的活动、执行被推迟的jobs、syncs以及standard alarms等任务，此时app可以连接网络。随后，系统会再次进入Doze模式，随着系统处于Doze模式时间增长，进入maintenance window的频率会越来越低，如下图。

然而，当用户大幅度移动设备、点亮屏幕或是接入电源时，系统会立即退出Doze模式，进入正常的活跃状态。

**Doze 模式下的限制**

* 无法访问网络
* 系统会忽略wake locks，即app无法持续占有电源
* 标准闹钟 AlarmManager（包括[setExact()](http://developer.android.com/reference/android/app/AlarmManager.html" \l "setExact(int,%20long,%20android.app.PendingIntent))和[setWindow()](http://developer.android.com/reference/android/app/AlarmManager.html" \l "setWindow(int,%20long,%20long,%20android.app.PendingIntent))）都会被延时到下一个maintenance window才激活  
  1 . 如果app仍需要在Doze时使闹钟生效，可以使用[setAndAllowWhileIdle()](http://developer.android.com/reference/android/app/AlarmManager.html" \l "setAndAllowWhileIdle(int,%20long,%20android.app.PendingIntent))或[setExactAndAllowWhileIdle()](http://developer.android.com/reference/android/app/AlarmManager.html" \l "setExactAndAllowWhileIdle(int,%20long,%20android.app.PendingIntent))  
  2 . 使用函数[setAlarmClock()](http://developer.android.com/reference/android/app/AlarmManager.html" \l "setAlarmClock(android.app.AlarmManager.AlarmClockInfo,%20android.app.PendingIntent))设置的闹钟在Doze时仍会生效，系统会在闹钟生效前退出Doze。
* 系统不会进行Wi-Fi扫描
* 系统不允许sync adapters运行
* 系统不允许JobScheduler运行

小结：在Doze模式下，网络访问被屏蔽，维持唤醒（Wake Lock）被忽略，定时任务（Alarm）被推迟（以指数递增的后延），而不会杀进程，但是push通道不可避免的会断掉。其实系统进入Doze模式的条件比较苛刻，首先有一个小时的等待期 —— 屏幕关闭后半小时开始进行『大幅度运动监测』，接下来半小时内无大幅度运动才会进入Doze Mode。所以正常情况下，系统很难进入Doze模式。当移动设备、点亮屏幕或者充电时系统就会恢复活跃，push可以立即恢复运行，并收到消息推送。另外，尽管Android 6.0引入了setAndAllowWhileIdle()和setExactAndAllowWhileIdle()两个新的函数来在Doze模式下执行定时任务，但是每个app每9min只能执行一次。

 二、App Standby

系统通过App Standby机制可以把用户没有使用的app置为空闲状态，当然前提是app没有命中以下条件：

* 用户明确的启动了该app
* app有前台进程（包括前台的activity、前台service，或者正在被其他activity及前台service使用）
* app当前没有在锁屏界面或者通知栏有通知

当设备连接到电源时，系统会把所有处于standby的app恢复到正常运行状态，如果设备长期处于空闲状态，系统会每天允许standby的app联网一次。

小结：在App Standby状态下，App很难得到运行，因此Push受到的影响会更大，当用户重新打开App或者充电时就会恢复。

三、官方解决方案

1. 使用GCM，但**在国内环境下是不现实的**；
2. Android提供了白名单机制，使app能在Doze和Standby时仍能够访问网络和使用partial wake locks。然而，其他的限制，如作业调度，异步操作，及闹钟仍不会执行。  
   开发者可以通过接口isIgnoringBatteryOptimizations()来判断当前app是否处于白名单中。对于白名单，有以下几点值得关注：

(1) 用户可以手动配置白名单，其位置在设置->电池→电池优化；

(2) 开发者可以通过发送action为REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS的Intent来调起电池优化配置页面；

(3) 开发者也可以通过添加权限REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS使其可以触发系统对话框，询问用户是否将该app添加到白名单中。触发添加白名单确认对话框的action为ACTION\_REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS；

(4) 当然，如果用户需要，他们可以随时在电池优化配置里把app从白名单中移除。

**这种方式类似与请求定位等权限，但是对用户不是特别友好，可以考虑。**

四、总结

其实Doze模式对Push的影响不大，因为很难进入Doze模式，且可以通过setAlarmClock()继续使用定时器。主要是Standby模式影响比较大，如果app长时间没被使用，就无法自启动运行。目前的解决方案主要是通过官方的REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS权限让用户加入白名单，其他暂时没有有效的解决方案。

五、实验测试

| **Doze测试** | **测试项** | **6.0原生系统** | | **7.0原生系统** | |
| --- | --- | --- | --- | --- | --- |
| **Doze测试** | **测试项** | **6.0原生系统** | | **7.0原生系统** | |
| 1 | Doze模式下能否保持长连、发送心跳、接收push | 否（定时任务被推迟执行，无法发送心跳，长连12min后中断，但长连中断之前可以接收push） | |  | |
| 2 | 退出Doze模式后是否可以立即恢复Push长连并接收到Push | 是（由于设置了定时器任务，退出Doze模式时，会立即执行定时任务，并重建长连） | |  | |
| 3 | Doze模式下使用setAndAllowWhileIdle()设置定时器是否有效 | 是（使用该函数可以暂时退出Doze模式发送心跳，但是9min只能发出一个心跳，无法维持长连，在长连断掉后，会自动重连）  20:08:39 第一个心跳  20:20:39 长连中断  20:23:39 重连 | |  | |
| **Standby测试** | **测试项** | **6.0原生系统** | **7.0原生系统** | |
| 1 | 系统多久后会强制app进入standby状态 |  |  | |
| 2 | 点评app将push设为了前台进程，是否会被强制进入standby状态 | 使用adb强制进入standby模式是可以进入的，但是不知道系统自身会不会使我们的应用进入standby模式。进入standby模式后，定时任务仍然可以执行，但是不能联网。 |  | |
| 3 | 若可以进入standby模式，接入电源后，push是否可以立即恢复并接收到push | 由于触发了重连定时器，恢复后可以重新建立长连接。 |  | |