



QQ18334373taikongyi的专栏

目录视图

摘要视图

订阅

个人资料



QQ18334373taikongyi



访问：21366次

积分：621

等级： 3

排名：千里之外

原创：32篇 转载：23篇

译文：7篇 评论：1条

文章搜索

文章分类

- 移动开发 (10)
- C++ (12)
- Android (16)

文章存档

- 2017年01月 (4)
- 2016年12月 (2)
- 2016年11月 (3)
- 2016年04月 (19)
- 2015年07月 (1)

展开

阅读排行

- Android6.0状态栏图标原 (2186)
- vc++中printf的一个实现 (1372)
- Android Doze模式 (1065)
- secs协议 (734)
- MFC的CDialog完全展开 (668)
- vmware tools+ubuntu+ai (609)
- C++ Primer读书笔记（二 (555)
- 若干时间后忘记的问题--- (474)
- adb、fastboot常用命令 (474)
- java面向对象总结 (464)

赠书 | AI专栏（AI圣经！《深度学习》中文版）

每周荐书：分布式、深度学习算法、iOS（评论送书）

【获奖公布】征文 | 你会为AI转型么？

Android Doze模式

2016-11-30 10:33

1073人阅读

评论(0)

收藏

举报

分类： Android (15)

概论

从android6.0开始，**Android**引入了两种省电技术以延长电池的使用寿命，分别是低电耗模式（Doze）和应用待机模式（App standby）模式。当设备屏幕关闭，不充电，Doze模式会通过限制app访问网络，推迟后台作业，同步来减少了对于电池电量的消耗。对于一些不是经常使用的app，Appstandby会禁止app后台的网络活动。

只要App在android6.0或更高版本的系统上运行，都会受到Doze和App Standby 模式的约束。（前提是需要编译ROM之前通过配置xml文件开启Doze功能）。

根据第三方测试显示，两台同样的Nexus 5，开启的Doze的一台待机能达到533小时。

1 Doze模式要点分析

1.1 Android 6.0 的Doze模式：

假如设备不插电充电，保持静止，且关闭屏幕一段时间，设备就会进入Doze模式，在Doze模式期间，系统会禁止app访问网络，延迟app的后台作业(JobScheduler)，同步(SyncAdapter)，alarm。

在每一个maintenance 窗口结束后，系统又会重新进入Doze模式，挂起网络连接活动，推迟任务，同步，alarm。随着时间的推移，系统调度maintenance 窗口的频率会越来越少。

1.2 Android 7.0增强Doze模式：

Android N 则通过在设备未插接电源且屏幕关闭状态下、但不一定要处于静止状态，就会进入Doze模式，分为两个阶段，根据对App行为的限制分为浅度doze（light idle）和深度doze（deep idle），相比android6.0，进入doze模式的门槛更低。

Android7.0 进入Doze模式的两个阶段：
screen off on battery stationary not/yes

当设备进入Doze模式后，系统会周期性的唤醒设备以提供简短的维护时间窗口（maintenance）（上图橙色），在此窗口期间，应用程序可以访问网络并执行任何被推迟的作业和同步，然后又进入Doze模式中的IDLE状态（上图绿色），周期性循环，并且从First level到进入Second level的过程中进入maintenance的周期会逐渐变大。总结表格如下：

级别	IDLE程度	进入条件	对App的行为限制	退出条件	设备硬件要求
第一级别限制	浅度IDLE (Light IDLE)	设备不充电，屏幕关闭	1不能访问网络； 2推迟作业和同步	激活屏幕，设备充电	无
第二级别限制	深度IDLE	设备不充电，屏幕关闭,设备保持静止一	1 不能访问网络；	激活屏幕，设备充电，有移动动作，Alarm 到时	要求具有SMD（Significant motion Dector），一种用于检测设备是否处于静止状态传感器

评论排行

- vc++中printf的一个实现 (1)
- java、C++、C#、Objec (0)
- virtual、inline、interface (0)
- c语言中常用宏定义 (0)
- 预处理器标识 (0)
- JAVA中抽象类，接口 总! (0)
- http://blog.csdn.net/zhan (0)
- 若干时间后忘记的问题-- (0)
- 数列题快速求解方法 (0)
- 重入函数和非重入函数 (0)

推荐文章

- * CSDN日报20170725——《新的开始，从研究生到入职亚马逊》
- * 深入剖析基于并发AQS的重入锁(ReentrantLock)及其Condition实现原理
- * Android版本的"Wannacry"文件加密病毒样本分析(附带锁机)
- * 工作与生活真的可以平衡吗？
- * 《Real-Time Rendering 3rd》提炼总结——高级着色：BRDF及相关技术
- * 《三体》读后思考-泰勒展开/维度打击/黑暗森林

最新评论

- vc++中printf的一个实现
- yiping1: 这本来就是系统函数。

			2 wake lock失效；		
	(Deep IDLE)	段时间	3 禁止GPS/WIFI 扫描；		
			4 Alarms推迟		
			5 作业，同步推迟。		

注：Notifications的到来不会导致退出doze模式

Significant motion sensor 介绍:
用来检测用户当前的移动状态,例如走路,骑着自行车,正在开车等
传感器选用规则：

Doze 模式启动之前会对当前设备传感器进行检查，以决定doze模式的深度：

- 步骤1：检查com.android.internal.R.integer.config_autoPowerModeAnyMotionSensor，如果有设置ID，根据ID获取传感器实例，没有的话跳到步骤2。
- 步骤2：检查com.android.internal.R.bool.config_autoPowerModePreferWristTilt，如果有设置，则获取传感器ID，根据ID获取传感器实例，没有的话跳到步骤3。
- 步骤3：获取TYPE_SIGNIFICANT_MOTION传感器，获取成功则有条件可以进入Deep Doze模式，没有则只能进入Light Doze模式。

1.3 设备开启Doze功能

设备开启Doze功能后，根据是否具有SMD传感器决定是否进入深度Doze模式，如果没有SMD传感器，Doze模式下只能进入浅度睡眠Doze模式。
开启Doze功能步骤：
（1）确定设备已经安装了GCM服务（可选）；
（2）在源代码将 overlay/frameworks/base/core/res/res/values/config.xml中的config_enableAutoPowerModes 参数数值设置为 true，
即bool name="config_enableAutoPowerModes">true</bool>，这个参数在AOSP的源码默认是关闭的。
（3）确定预装的app和服务在Doze下进行过适配。
（4）将一些重要的服务加入白名单，如果不能使用GCM服务，app又要求具有即使推送消息功能，那么最好把这类app加入白名单。
Doze控制service在android源码framework里面的路径：
DeviceIdleController.java (frameworks\base\services\core\java\com\android\server)

1.4 Doze模式下的各种状态

Doze模式控制下的7种状态解释：
🔇 ACTIVE: 设备在使用中，或者连接着电源充电。
🔇 INACTIVE: 使用者关闭了屏幕，不充电。
🔇 IDLE_PENDING: 设备已经过了不活动的状态，准备进入第一级别的idle模式，这里就会开始检测设备是否移动。
🔇 SENSING: 正在检测设备是否静止。
🔇 LOCATING: 获取位置信息。
🔇 IDLE: 设备进入了第二级别的idle模式，处于完全doze模式。
🔇 IDLE_MAINTENANCE: 短暂的推出idle，进入常规的处理模式。

1.5 设备进入Doze模式流程分析

ACTIVE
INACTIVE （30 min）(First level) 设备保持不活动状态
IDLE_PENDING （30min）判断用户的运动状态：骑自行车，开车，走路
sensing （4min）检测设备是否处于静止状态
locating （30s）在进入IDLE之前更新位置信息

关闭

IDLE (60min × m)(Second level)进入深度IDLE,m为倍数

IDLE_MAINTENANCE (5min)

设备不充电，关闭屏幕后，会进入了inactive状态，系统会设置一个5min的alarm和30min的alarm，5min的alarm到时后设备会进入light doze模式，light doze模式会持续60min（假设设备没有被唤醒），接着就进入deep doze模式。

在处于light doze模式期间，idle的持续时间达5min，这个持续时间会随着idle次数的增加而增加（5min<=5min*m<=15min），每一次idle结束后紧接着会有maintenance窗口出现，这个maintenance窗口的时间是60s。以上时间参数都可以进行预先设置，根据实际情况进行调整，更详细流程可参考文档：android 7.0 Doze模式状态图.pdf

总结：设备在不充电，关闭屏幕大约5min后会进入light idle模式，大约经过1h后会进入deep idle模式，进入deep idle后则意味着完全DOZE模式--最大力度省电。

2 在Doze模式下适配App

2.1 Doze模式下对alarm的优化

android6.0 引入了setAndAllowWhileIdle()和 setExactAndAllowWhileIdle()

通过这两个API设置的alarm，即使在Doze模式下alarm也能运行，但根据Google官方文档，使用这应用每9分钟只能唤醒一次alarm。

如果App有重要的定时通知功能，例如日历App的代办事项提醒功能，则需要采用最新的api优化appl能。

🔪 setAndAllowWhileIdle

设置可在idle模式下执行的alarm

🔪 setExactAndAllowWhileIdle

设置可以在idle模式下执行的精确Alarm

2.2 消息推送

Google建议App使用Google Cloud Messaging(GCM)进行消息推送，因为当系统处于Doze模式，或者App处于standby模式时，GCM GCM high-priority messages会唤醒app并允许其进行网络访问，App 处理完推送消息后又回到待机模式。所以GCM不会影响系统的Doze状态，也不会影响其他处于待机状态的App。

因为国内基本上不能使用GCM，但鉴于针对国内开发的应用基本不会使用GCM，使用为了适配7.0而让app使用GCM服务基本不可行。

2.3 白名单的运用

在国内，android设备基本上不能使用Google服务，但App又需要网络连接来接收实时消息推送，针对这个问题，系统提供了可以配置白名单让app免于被Doze模式和App standby模式优化。

一个被加入白名单的App 可以在Doze模式期间和App待机模式期间使用网络和持有partial_wake_lock（屏幕处于关闭状态，但CPU依然运作，直到释放锁），但是像作业，同步，alarm这些任务还是会被推迟运行，依然受Doze模式和App待机模式约束，也就是说，并不是加入白名单就完全不受Doze模式和App待机模式约束。

2.3.1 用户动态添加App进入白名单

一个App可以通过调用isIgnoringBatteryOptimizations()函数来检查自己的是否在白名单列表里面。如果不在，可以通过以下几种方式将自己加入白名单：

🔪 用户可以通过设置>电池>电池优化来手动配置白名单

🔪 App 可以通过发送Intent：ACTION_IGNORE_BATTERY_OPTIMIZATIONS_SETTINGS 直接导航用户到电池优化界面

🔪 在App加入 REQUEST_IGNORE_BATTERY_OPTIMIZATIONS权限，然后app通过发送Intent：ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS去触发弹出一个请求加入白名单的对话框，用户在对话框中选择是或否在决定是否加入白名单(推荐这种方式)

🔪 用户通过“设置”app进行设置：设置->电池->电池优化。

最终这些被加入白名单的app都会被保存在：/data/system/deviceidle.xml文件里面。

2.3.2 预先添加System App进入白名单

对于一些系统应用，需要在第一次开机的时候就自动加入白名单，只需在编译ROM之前预先先把package包名写入frameworks\base\data\etc\platform.xml

格式为：

<allow-in-power-save package="com.android.providers.downloads" />

系统编译过程中会自动把platform.xml 安装到system镜像的/system/etc/permissions目录下。

测试案例：

关闭

导出platform.xml，将platform.xml配置成如下：

```
<allow-in-power-save package="com.android.providers.downloads" />
<allow-in-power-save package="com.yulong.android.coolmart" />
<allow-in-power-save package="com.yulong.android.coolyou" />
```

然后替换platform.xml，恢复工厂设置后“电池优化”界面，选择“未优化”，可以看见对于app被配置成未优化。

2.4 后台优化

Android N 删除了三项隐式广播，以帮助优化内存使用和电量消耗，分别是：

- ✂ CONNECTIVITY_ACTION
- ✂ ACTION_NEW_PICTURE
- ✂ ACTION_NEW_VIDEO

应用不再接收静态注册的CONNECTIVITY_ACTION广播，但是应用在前台时仍然能够监听到动态注册CONNECTIVITY_CHANGE广播。

应用程序不能发送或者接收ACTION_NEW_PICTURE 和ACTION_NEW_VIDEO广播,这个优化会影响

应用适配方案1：

将之前静态注册的CONNECTIVITY_ACTION改成成动态注册。

应用适配方案2：

使用JobScheduler API执行网络任务，使用JobInfo.Builder()创建JobInfo对象时,调用setRequiredNetworkType()设置当检测到有网络连接时开始网络任务。

3 App Standby模式

如果用户在一段时间内没有使用某个app，那么系统就会让这个app处于空闲模式，限制app访问网络，推迟app的作业和同步任务。

当设备插上电源时，或者app被用户启动，系统就会让app退出待机模式，恢复app到正常运行状态，让app自由访问网络和执行它们之前延迟的作业和同步。

App处于活动状态下的表现：

1. 一个进程正在前台运行（进程可以是一个activity或者一个前台服务，或者当前进程正在被其他activity或者服务调用），例如通知监听器，可以外界供访问的服务，动画壁纸等；
2. 一个可以被用户看见的通知，例如在锁屏界面的通知，在通知栏托盘里面的通知。
3. 一个app明确的被用户直接启动，例如直接点击app启动。

如果app保持一段时间不处于上述的三种状态中，就会标记为不活动，系统就会调度App进入App 待机模式。

监控	处于standby模式对app的限制	退出standby模式的条件
设备没有充电， app在一段时间内都没有被启动过（几小时以上）， App会被标记为standby 模式	每天只能使用网络一次，推迟app的同步和作业	1 设备充电 2 App被启动

关闭

用户控制界面：

打开设置->开发者选项->未启用的应用，可以手动设置app是否启用。

4 使用GCM与App进行信息交互

Google Cloud Messaging(GCM) 是谷歌提供的一套用于服务端实时推送消息到设备终端的服务，所有需要实时消息推送的app都可以共享使用这个连接。这个服务是可以共享使用的，所有app可以共享使用GCM服务。

当系统处于Doze模式，或者App处于standby模式时，GCM high-priority messages会唤醒app并允许其进行网络访问，App 处理完推送消息后又回到原先的模式。所以GCM不会影响系统的Doze状态。

5 在Doze模式和App Standby模式测试App状态

5.1 在Doze模式测试app

通过如下步骤测试你的app：

- 1. 一台搭载android6.0（或更高版本）的真实设备或虚拟机
- 2. 开启debug 模式，并安装需要测试的app
- 3. 运行app，保持app活动状态
- 4. 关闭设备屏幕
- 5. 通过如下命令强制系统进入Doze模式：

\$ adb shell dumpsys battery unplug

\$ adb shell dumpsys deviceidle step

第二条命令需要多跑几次，直至设备状态进入idle状态。可以根据命令返回的信息来判断是否已经另

式：

当命令窗口返回IDLE时，则完全进入了Doze低电耗模式，再运行一次adb shell dumpsys deviceidle --step=，设备就会进入maintenance 维护窗口状态。

其他常用调试命令：

- 1. 列出白名单命令：adb shell dumpsys deviceidle whitelist
- 2. adb shell dumpsys deviceidle -h查看命令帮助：

5.2 在App 待机模式下测试App

步骤如下：

- 1. 一台搭载android6.0（或更高版本）的真实设备或虚拟机
- 2. 打开debug 模式，安装app
- 3. 打开app，保持app 处于活动状态
- 4. 运行如下命令强制app进入App待机模式:

\$ adb shell dumpsys battery unplug

\$ adb shell am set-inactive <packageName> true

通过如下命令模仿唤醒app

\$ adb shell am set-inactive <packageName> false

\$ adb shell am get-inactive <packageName>

6 传统省电VS Doze省电

传统省电：通过调节CPU和联网频率，或者只保留通话，信息重要功能，直接关闭GPS,WIFI,移动数据网络。

Doze省电:更加智能化的省电

- 1. 智能检测当前用户使用手机场景；
- 2. 根据用户使用手机场景调节省电深度级别；
- 3. 不需要手动关闭app，Doze会根据用户使用场景自动限制app行为；
- 4. 用户不需要手动开启doze省电，doze会自动运转后台，随时随地根据

Google 官方Doze介绍链接：

<https://developer.android.com/about/versions/nougat/android-7.0-changes.html#doze>

<https://developer.android.com/training/monitoring-device-state/doze-standby.html>

7 应用开发测试案例

7.1 在Doze模式下使用alarm

案例描述：使用android6.0引入的新API setAndAllowWhileIdle()创建一个alarm，启动alarm后，通过adb命令让设备进入Doze模式，确定Doze模式下alarm是否起作用，确定alarm起作用的时长。

关键代码：

关闭

```
在AlarmService定义一个alarm , 设定10s

publicclass AlarmService extends Service {

    publicstatic String TAG = "AlarmService-Test";

    @Override
    public IBinder onBind(Intent arg0) {
        // TODO Auto-generated method stub
        returnnull;
    }

    @Override
    publicvoid onCreate() {
        // TODO Auto-generated method stub
        Log.d(TAG, "OnCreat");
        super.onCreate();
    }

    @Override
    publicvoid onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
    }

    @Override
    publicint onStartCommand(Intent intent, int flags, int startId) {
        // TODO Auto-generated method stub
        AlarmManager alarmManager =(AlarmManager)getSystemService(Context.ALARM_SERVICE);
        int offset = 10*1000;
        long trigTime = SystemClock.elapsedRealtime()+offset;
        Intent alarmIntent = new Intent(this, AlarmReceiver.class);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, alarmIntent, 0);
        Log.d(TAG, "alarm set at "+new Date().toString());
        alarmManager.setAndAllowWhileIdle(AlarmManager.ELAPSED_REALTIME_WAKEUP, trigTime, pendingIntent);
        returnsuper.onStartCommand(intent, flags, startId);
    }
}

10s 到时后AlarmReceiver接收到广播 , 并重新启动AlarmService , 重新设备一个10s 的alarm。

publicclass AlarmReceiver extends BroadcastReceiver {

    @Override
    publicvoid onReceive(Context arg0, Intent arg1) {
        // TODO Auto-generated method stub
        Log.d(AlarmService.TAG, "alarm bell at "+new Date().toString());
        Intent alarmIntent = new Intent(arg0, AlarmService.class);
        arg0.startService(alarmIntent);
    }
}

测试步骤：
1 启动app ;
```

关闭

- 2 点击start alarm button，代码设备10s后alarm到时，发出广播
- 3 关闭屏幕
- 4 通过adb 命令强制设备进入Doze模式

从日志分析得出结论：

- 1 在未进入Doze模式下，可以在10s 后处理alarm发出的广播；
- 2 在进入Doze模式后，则至少需要3分钟后才处理发出的广播，而且下一次处理广播后需要19分钟后才处理。

7.2 App请求加入白名单

案例描述：在一个app中通过ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS 请求用户将自己加入白名单。

关键代码：

- 1 在android manifest 文件里面加入权限：

<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>

- 2 请求加入白名单代码：

```
String packageName = mContext.getPackageName();
mContext.getSystemService(Context.POWER_SERVICE);
Intent intent = new Intent(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
intent.setData(Uri.parse("package:"+packageName));
startActivity(intent);
```

7.3 使用JobScheduler执行后台网络任务

案例描述：使用JobService来执行网络变化监听任务

- (1) 定义一个MyJobService

```
publicclass MyJobService extends JobService {

    privatestaticfinal String TAG = "MyJobService";

    @Override
    publicboolean onStartJob(JobParameters arg0) {
        // TODO Auto-generated method stub
        Log.d(TAG,"onStartJob "+ arg0.getJobId()+" "+new Date().toString());
        if(isNetworkConnected()){

            Log.d(TAG,"NetworkConnected++++"+ arg0.getJobId());
            //do something
            returntrue;
        }
        returnfalse;
    }

    privateboolean isNetworkConnected(){
        ConnectivityManager connManager=
        (ConnectivityManager)this.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo=connManager.getActiveNetworkInfo();
        return (networkInfo!=null&& networkInfo.isConnected());
    }

    @Override
    publicboolean onStopJob(JobParameters arg0) {
        // TODO Auto-generated method stub
        Log.d(TAG,"onStartJob "+ arg0.getJobId());
        returnfalse;
    }
}
```

关闭

（2）调度任务方法实现

```
public static void scheduleJob(Context context){

    JobScheduler jobScheduler = (JobScheduler) context.getSystemService(Context.JOB_SCHEDULER_SERVICE);
    JobInfo jobinfo = new JobInfo.Builder(
        252,
        new ComponentName(context, MyJobService.class))
        .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)
        .build();

    jobScheduler.schedule(jobinfo);
    Log.d("MyJobService", "scheduleJob at "+new Date().toString());

}
```

setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)设置任务运行条件，当运行条件为发现执行任务。

（3）manifest文件中生命service

```
<service android:name="com.example.jobscheduledemo.MyJobService" android:permission="android.permission.BIND_JOB_SERVICE" />
</service>
```

scheduleJob开始后大约经过十几秒后，手动打开wifi网络开关，MyJobService检测到网络可以用，立即运行任务。

7.4 设置App 为Standby模式

UsageStatsManager管理类提供了一个隐藏的直接设置某个app为standby模式的api：

```
/**
 * @hide
 */
public void setAppInactive(String packageName, boolean inactive) {
    try {
        mService.setAppInactive(packageName, inactive, UserHandle.myUserId());
    } catch (RemoteException e) {
        // fall through
    }
}
```

（1）声明权限

```
<uses-permission android:name="android.permission.CHANGE_APP_IDLE_STATE"/>
```

（2）设置app为inactive或active：

```
setAppInactiveByUser("com.android.calculator2", arg1);
```

关闭

顶

0

踩

0

上一篇

AndroidN新特性

下一篇

Android 事件捕捉和处理流程

相关文章推荐

- Android 6.0新特性之Doze模式
- Android Doze模式分析
- 深入Android 'M' Doze
- Android6.0中Doze模式实现原理的源码分析
- Android 6.0的省电技术Doze作用影响以及避免方法

- Android6.0新特性之DozeMode
- Android M新特性Doze and App Standby模式详解
- Android Doze模式调试
- 关于Doze 模式下对AlarmManager 的影响
- Android 7.0 Doze模式分析



猜你在找

- 【直播】机器学习&数据挖掘7周实训--韦玮

【直播】3小时掌握Docker最佳实战-徐西宁

【直播】计算机视觉原理及实战--屈教授

【直播】机器学习之矩阵--黄博士

【直播】机器学习之凸优化--马博士
- 【套餐】系统集成项目管理工程师顺利通关--徐朋

【套餐】机器学习系列套餐（算法+实战）--唐宇

【套餐】微信订阅号+服务号Java版 v2.0--翟东平

【套餐】微信订阅号+服务号Java版 v2.0--翟东平

【套餐】Javascript 设计模式实战--曾亮

查看评论

暂无评论

发表评论

用户名：

haijunz

评论内容：



提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved



关闭