做一个安静的APP

- 2004~2015 专注移动工具APP开发，Symbian到Android
- 前阿里巴巴无线事业部·资深技术专家
- 致力于推动Android生态朝着更加开放和有序的方向发展
- 代表作：FontRouter、绿色守护（Greenify，全球用户数500万+）

- GitHub、Twitter、微博、XDA……
  - 江湖ID：oasisfeng

话说，从前……

- 封闭的监狱 与 开放的丛林 Jail v.s. Jungle
  - iOS 的监狱里，狱警用 App Store Review Guidelines 守护着秩序，震慑着恶念
  - Android 的丛林中，到处是带枪的猎人，用户不得不小心翼翼穿行于险恶的环境
  - 绿色守护是丛林生存的必备防身之物

敢问，路在何方？

- Android Marshmallow 赋予用户以强有力的武器
  - Runtime Permission
  - Doze Mode
  - App Standby
  - Improved Battery/Memory Graph

- 都已经无拘无束惯了，以后咋办？
  - 如何说服用户赋予权限？
  - 能否挣脱 Doze Mode & App Standby 的枷锁？
  - 还有机会逃脱用户的审视？

就让我做一个安静的APP吧

然而……

唉哟，这……我真不是有意的

- 一不小心就拖慢系统了
  - Broadcast receivers in AndroidManifest （可能远比你想象中慢）

```
<receiver ...>...<action name="android.net.conn.CONNECTIVITY_CHANGE"/>...
if (! network.isConnected()) return;
```

    - 进程创建和初始化（在低端机型上可能需要约 1 秒）
    - Application.onCreate()（在很多大型App中往往开销非常大，小心）

  - 『连环唤醒』—— 中低端手机的杀手
    - 你可能在无意识见就亲手创造了一个『连环杀手』！
    - 它为什么令资深用户如此深恶痛绝？

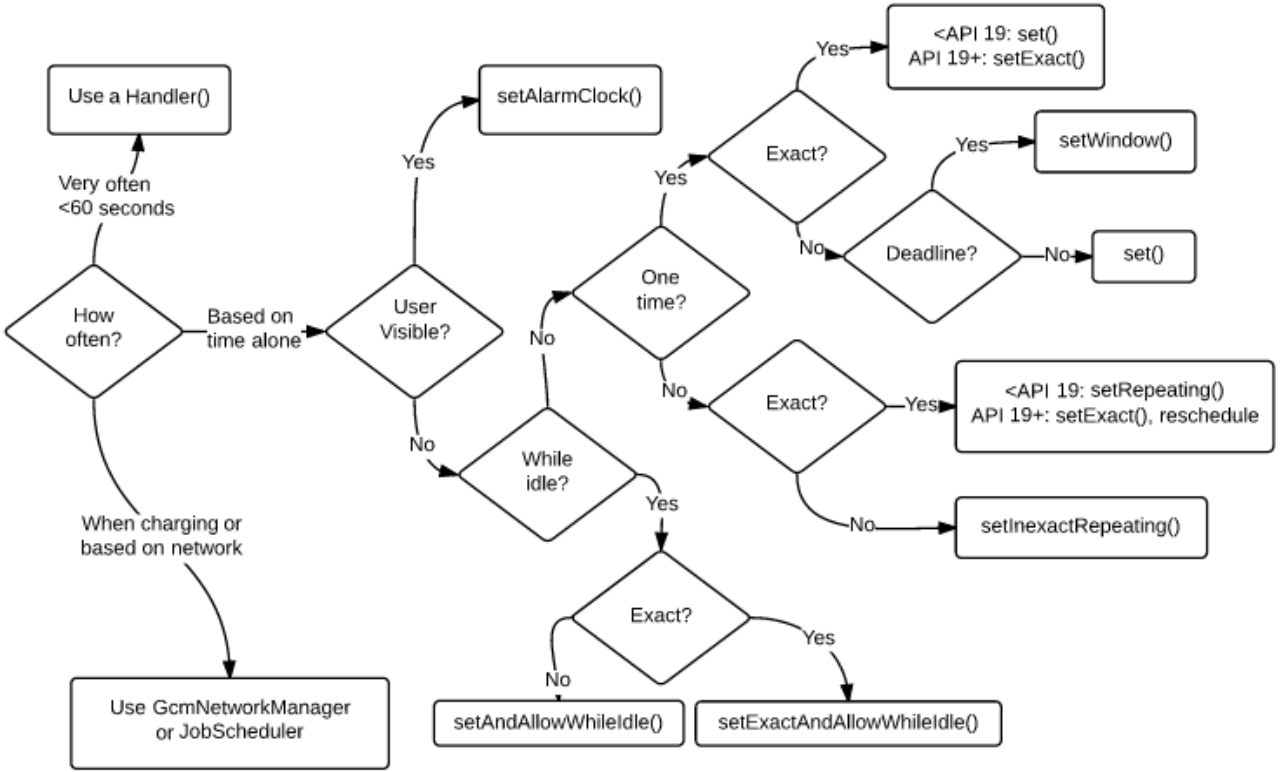
- 一不小心就拖慢系统了
  - Solution
    - 避免使用静态声明的 Broadcast Receiver，尽可能动态注册
    - 不再需要时禁用静态 Receiver PackageManager.setComponentEnabledSetting()
    - Lazy 初始化、分离的 UI 初始化（利用ActivityLifecycleCallbacks）
    - SyncAdapter、JobScheduler、GcmNetworkManager

- 一不小心就拖慢系统了
  - Background services
    - 内存消耗（为啥会不断增加？）
    - GC（Dalvik的最痛）
    - 频繁被重启
    - Exception spam in logcat
  - Solution
    - 独立于UI的后台进程
    - Object Pool
    - 改掉那些IDE自动生成的 `catch(…) { e.printStackTrace(); }`

- 一不小心就拖慢系统了
  - IO 密集型任务（Flash存储）
    - 小心剩余空间！
  - 非唤醒型 Alarm 埋下的地雷（呃？）
    - 造成屏幕解锁卡顿、开启相机卡顿

还有用户更敏感的……

- 一不小心又后台耗电了
  - 当心 Alarm！
    - 确保 Target SDK Version ≥ 19（Android 4.4）
    - 务必使用可对齐的周期：15分钟、30分钟……（INTERVAL_XXX 常量）
    - Wake up 与否，有讲究！（non-wakeup 并不总是最佳选择）
    - 尽可能不要使用 Exact Alarm！


Use a Handler()
Very often
<60 seconds
How
often?
Based on
time alone
User
Visible?
Yes
setAlarmClock()
No
While
idle?
Yes
Exact?
Yes
<API 19: set()
API 19+: setExact()
No
Deadline?
Yes
setWindow()
No
set()
No
One
time?
Yes
Exact?
Yes
<API 19: setRepeating()
API 19+: setExact(), reschedule
No
setInexactRepeating()
No
Exact?
Yes
setExactAndAllowWhileIdle()
No
setAndAllowWhileIdle()
When charging or
based on network
Use GcmNetworkManager
or JobScheduler

- 一不小心又后台耗电了
  - 当心 WakeLock！
    - PARTIAL_WAKE_LOCK 远比你想象中费电
    - 及早释放！
    - 确保得到释放（建议 try…finally）
    - 不要用WakeLock保持大文件下载，你需要的是 DownloadManager

- 一不小心又后台耗电了
  - 当心后台持久服务
    - 低端机型上系统内存回收造成的后台服务频繁重启，会大量积累耗电
    - 被忽视的偷跑线程，也会造成异常的后台耗电
    - 注册的各类回调，记得在 ACTION_SCREEN_OFF 后暂停

- 一不小心又后台耗电了
  - 当心 传感器 & 定位请求
    - 尽量使用一次性请求，避免持续侦听
    - 尽量使用网络定位，避免限定GPS定位；可能的话，考虑『Passive Provider』
    - 采集分析类传感器需求，尽可能使用批量采集机制（Android 4.4+）

- 第三方Push服务往往是耗电大户
  - 对比测试，慎重选择
  - 抱别人的大腿，省自己的电
  - 面向全球市场的App，请优先使用Google Cloud Messaging
  - 务必给用户关闭的机会

这么多坑，好累.....

简单，才是最好

- 你真的需要后台持久服务？
  - 很可能 SyncAdapter 已经够用
  - 考虑短生命周期的后台服务：Broadcast / Alarm + IntentService
  - 选用一个不需要启动后台持久服务的Push通道（如GCM）

- 你真的需要WakeLock？
  - 保持屏幕常亮并不需要WakeLock `(LayoutParams.FLAG_KEEP_SCREEN_ON)`
  - 小数据量下载/上传：别惦记 WakeLock，加上必要的重试机制。
  - 大数据量下载：DownloadManager
  - 大数据量上传：你需要的可能是 WifiLock

- 你真的需要Push？
  - 内容的实时性诉求
    - 你提供即时通讯服务、社交网络、协作类工具？
    - 考虑用 Sync Adapter 代替 Push
  - 内容的性质
    - 用户主动、迫切需要获得的内容？
    - 也许，你需要的只是一个社交媒体账号（比如 微信公众号）

当地球人已经无法阻止你的时候

请记得，用户终究可以选择『卸载』

谢谢大家！

冯森林 / oasisfeng

One more thing……

即将在我的GitHub上发布