Contents

1 How can we save power for mobiles

- Knowledge about the power consumption of each component and app
- Screen /network/CPU off or release other device resources as soon as not needed
- Limit resources for less frequently used apps
- Set device to sleep as soon as no user interactions
- Global management of running jobs across all apps (i.e. perform syncs, upload/download together at fixed time window)

1.1 Battery Consumption Statistics

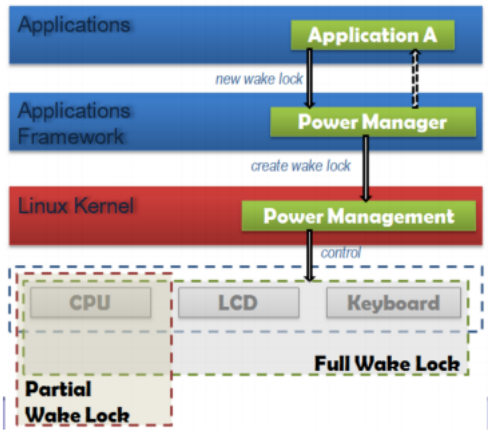
- Framework tracks the time that devices spend in different states (e.g. WiFi chipset:on/off, Display :low/high brightness)
- Controlling service pushes state changes to BatteryStats service. Framework pulls the data at these transition points
- App consumption power is calculated based on CPU run time at specific speeds

2 Android Power Management Concept

- Designed for mobile devices
- Goal is to prolong battery life
- Based on Linux Power Management (**not suitable for a mobile device**)
  - G0 (working)
  - G1 (sleeping)
    - S1 (CPU stops executing instructions, power to CPU and RAM maintained)
    - S2 (CPU powered off, cache is flushed)
    - S3 (Standby / sleep / suspend to powered RAM)
    - S4 (Hibernate / suspend to disk, RAM powered off)
  - G2 (S5, soft off)
  - G3 (mechanical off)
- Mobile phones have a default off behaviour

2.1 Power Management Design

- A wrapper to Linux Power Management
- Added to the Kernel Wake Lock mechanism
- Apps need to request CPU & Screen to be on with WakeLocks, otherwise Android will shut down the CPU
- Wake locks and timeouts constantly switch the state of the systems power
  - Overall system power consumption decreases
  - Better use of battery capacity



3 Battery Life Enhancement in Android (Android 6.0 onwards)

- **App Standby**: defer background activity for apps with no recent user interaction.
- **Doze**: deep sleep if user has not actively used the device for extended periods of time
- **Exemptions**: system apps and cloud messaging services preloaded on phone are exempted from App Standby & Doze.
- Test apps in App standby & Doze mode for the desired performance

4 App Standby

- **Start conditions**: an app is not actively used for a certain time will be placed in an idle mode
  - Not doing foreground work (activities/service, pending notifications)
  - Hasnt been explicitly launched for certain time (days).
- **Actions**: No network access, no background jobs, can set Alarms, can use wake locks, allow network access once per day
- **Exists conditions**: plug in the device, do some foreground tasks

4.1 App Standby Buckets (Android 9.0)

Prioritize apps based on how recently and how frequently the apps are used

- Active: currently or recently used
- Working Set: regular use
- Frequent: often used not everyday
- Rare: not frequently
- Never: never

5 Doze

- **Starts Doze**: when device is in idle - screen off, on battery & stationary.
- **In Doze**: no network access; no CPU-intensive work; wakelocks ignored; no wifi scan; deferred alarm manager alarms; only high priority notification received; no job scheduler; no sync adapters.
- **Exits Doze**: user interaction, device motion, screen on, or AlarmClock alarm. Notification do not cause Doze exits.
- **Maintenance window**: complete pending activities (syncs, jobs, etc). MMS/SMS/Telephony services are excluded from Doze

6 Exemptions

- System apps and cloud messaging services preloaded on phone are exempted from App Standby & Doze.
- Use whitelist for apps to be partially exempt from Doze & App Standby
  - `isIgnoringBatteryOptimiztions()` to check if in the whitelist;
  - `ACTION_IGNORE_BATTERY_OPTIMIZATION_SETTINGS` intent to direct user to battery optimization options
  - `REQUEST_IGNORE_BATTERY_OPTIMIZATIONS` allow user to add app to whitelist directly
- Whitelisted apps can use network and hold partial wake locks during Doze & App Standby, but jobs & syncs are still deferred.
- App should not be on the whitelist unless cant use FCM high-priority messages or Apps core function is affected

7 Sustained Performance Mode (new API in Android 7.0)

- *Thermal throttling prevents* a long running apps maintain its performance (e.g. game, camera, virtual reality application etc.)
- App can request the platform to enter a SPM, and keep a consistent level of performance
  - Only tests on Android 7.0 on Nexus 6P devices
  - Set using `Window.setSustainedPerformanceMode()`
  - System automatically disables the mode when no longer in focus

8 Schedule Jobs in Android

For any timing operations that occur only during the lifetime of your application, it is best to use **application resources** (i.e. handler class) rather than system resources (e.g. Alarm manager/job Scheduler) to schedule tasks.

- AlarmManager (< API 21) Based on time. Only use when need to execute for specifc time.
- JobScheduler (≥API 21) Based on conditions
- Firebase Job Dispatcher (>API 14) Similar to JobScheduler but can be used for lower APIs and requires GooglePlay service
- Work Manager (New) Requires GooglePlay service Automatic use of alarmManager and jobScheduler

8.1 AlarmManager

- Schedule a task run at a specific time point/ time interval
- AlarmManager holds a CPU wake lock, when alarm receivers onReceive() is executing
- Alarm delivery is inexact (Android 5.0+); use setWindow and setExact for exact delivery.

8.1.1 AlarmManager Usage

- Specify an Intent to be broadcast at some time
  - setRepeating(int type, long triggerAtMillis, long intervalMillis, PendingIntent operation)
  - setExact(int type, long triggerAtMillis, PendingIntent operation)
- Extend Broadcast Receiver
  - onReceive method is called when the alarm goes off
  - Start a service to actually do the work

8.2 Efficient Scheduling

- Multiple scheduled alarms waking the phone: Suspending and waking the phone takes power
- More efficient to schedule multiple alarms at the same time

8.3 JobScheduler

- Many apps perform tasks asynchronously outside the main activity. (e.g download files, sync database to cloud etc.)
- Job Scheduler collects pending jobs across all apps, and schedule them to **run at about the same time** → sleep for longer → save power
- Specify requirements for network and timing for each job, then the JS robustly optimise the execution time.
- May defer jobs that comply with Doze & App Standby
- JobService will run on the main thread. Need to manage any asynchronous tasks yourself (e.g. Threads).

Reference section

placeholder