|  |
| --- |
|  |
| Power Management in Android |

|  |
| --- |
| Menna Mohamed  Macrina Ashraf |

**Android** is a [mobile operating system](https://en.wikipedia.org/wiki/Mobile_operating_system) developed by [Google](https://en.wikipedia.org/wiki/Google), based on a modified version of the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) and other [open source](https://en.wikipedia.org/wiki/Open_source) software.

**What is Power Management?**

It’s a feature that allows the users to control the amount of power consumed by a device they are using with minimal impact on performance. It enables devices to switch between various power modes, each device with different usage characteristics related to each device’s performance.

**Power Management in Linux:**

The need for power management is not very powerful in PCs as they have unlimited Power Source, but the need for power management appeared with the ever increasing power demand of mobile devices.

There are 2 main techniques used in power management in Linux:

* Advanced Power Management (APM).
* Advanced Configuration and Power Interface (ACPI).

**Advanced Power Management (APM):**

It’s found in older systems. Power management is done at Basic Input/Output System (BIOS), It deﬁnes the hardware independent software interface between hardware-speciﬁc power management software and an operating system power management policy driver. It allows higher-level software to use APM without any knowledge of the hardware interface. APM can extend the life of system batteries and thereby increases productivity and system availability.

**Advanced Configuration and Power Interface (ACPI):**

It is supported by modern PCs. ACPI allows control of power management from within the operating system unlike APM were power management is done at BIOS level. With ACPI the user can specify at what time a device, such as a display monitor, is to turn oﬀ or on. If the user chooses they can specify a lower level of power consumption when the battery starts running low so that essential applications can still be used while other less important applications are allowed to become inactive. In short ACPI gives both the user and the operating system the power to customize the system to suit their personal power needs.

**Power Management in Android:**

* Over 2 billion active devices on Android, so android continually adds new feature to help the platform optimize the off-charger behavior of applications and devices to extend battery life.

Android features in battery life enhancement:

* **App Restrictions:** The Platform can offer apps that effect negatively effect on battery life, so user can restrict those apps from resources consumption.
* **App Standby:** allows the system to determine that an app is idle when the user is not actively using it. The system makes this determination when the user does not touch the app for a certain period of time and none of the following conditions applies:
* The user explicitly launches the app.
* The app has a process currently in the foreground.
* **Doze:** The platform can enter in a state of deep sleep if users have not actively used their device for extended periods of time, periodically, the system exits Doze for a brief time to let apps complete their deferred activities, the system runs all pending syncs, jobs, and alarms, and lets apps access the network.
* Doze Restriction:
* Network access is suspended.
* The system ignores [wake locks](https://developer.android.com/reference/android/os/PowerManager.WakeLock.html) (wake lock is a mechanism to indicate that your application needs to have the device stay on).
* The system does not perform Wi-Fi scans.
* The system does not allow [sync adapters](https://developer.android.com/reference/android/content/AbstractThreadedSyncAdapter.html) to run.
* The system does not allow [JobScheduler](https://developer.android.com/reference/android/app/job/JobScheduler.html) to run.
* **Exemptions:** Preloaded system apps and cloud messaging services are typically exempted from App Standby and Doze by default, developers can use exemptions to apply these settings to their apps, users can exempt apps from App Standby and Doze power saving modes.
* Exemptions may be needed in:
* Device manufacturers using a Cloud Messaging platform other than Firebase Cloud Messaging (FCM).
* Carrier using non-FCM Cloud Messaging platform.
* Third-party application using non-FCM Cloud Messaging platform.
* Unnecessary exemptions:
* Undermine the benefits of Doze and App Standby.
* Allowing applications to overcome beneficial controls the platform has over power use and that can lead users to be unhappy about the power consumptions.
* Using Exemptions recommended in cloud messaging services or apps with similar functions.

**Power Management Advices for Android Developers:**

* Three Important things to keeping app power-thrifty:
  + - Makes app lazy first.
    - Take advantages of platform features in power management.
    - Using tools that help identify battery-draining reasons.
* Lazy first means looking for ways to reduce and optimize operations that are battery-intensive, the core questions support Lazy First design:
* Reduce: Are there redundant operations your app can cut out? , For example, can it cache downloaded data instead of repeatedly waking up the radio to re-download the data?
* Defer: Does an app need to perform an action right away?  For example, can it wait until the device is charging before it backs data up to the cloud?
* Coalesce: Can work be batched, instead of putting the device into an active state many times? For example, is it really necessary for several dozen apps to each turn on the radio at separate times to send their messages? Can the messages instead be transmitted during a single awakening of the radio?
* Platform features provides 2 categories of help to optimize app’s battery use:
* Platform provides several APIs that can be used in app implementation.
* Internal mechanisms in the platform that help in save battery life:
* Doze can affect apps differently, depending on the capabilities they offer and the services they use. Many apps function normally across Doze cycles without modification. In some cases, you must optimize the way that your app manages network, alarms, jobs, and syncs. Apps should be able to efficiently manage activities during each maintenance window.
* App Standby.
* App Standby buckets: System limits apps' access to device resources based on the user’s usage patterns.
* Background restrictions: System promote to user to restrict app that have bad behavior from access system resources.
* Power management restrictions: List of power restrictions that can be imposed on apps under certain conditions.
* Testing and troubleshooting.
* Tooling: Using Tools available (like Profile GPU Rendering, Battery historian) to identify which parts of your apps consume the most power.

**Improvement in power management in Android Pie:**

Android 9 introduced new features to ensure device power management improvement:

**App standby buckets**:

It prioritizes apps' requests for resources based on how recently and how frequently the apps are used. Based on the app usage patterns, each app is placed in one of five priority buckets. The system limits the device resources available to each app based on which bucket the app is in.

* **Four priority buckets:**
* **Active:** An app is in the active bucket if the user is currently using the app
* If an app is in the active bucket, the system does not place any restrictions on the app's jobs, alarms, or FCM messages.
* Example:
* The app has launched an activity
* The app is running a foreground service
* The user clicks on a notification from the app.
  + **Working set:** App is in the working set bucket if it runs often but it is not currently active,
* If an app is in the working set, the system imposes mild restrictions on its ability to run jobs and trigger alarm
* For example, a social media app that the user launches most days.
* **Frequent:**  app is in the frequent bucket if it is used regularly, but not necessarily every day.
* If an app is in the frequent bucket, the system imposes stronger restrictions on its ability to run jobs and trigger alarms, and also imposes a cap on high-priority FCM messages.
* For example, a workout-tracking app that the user runs at the gym.
* **Rare:** app is in the rare bucket if it is not often used.
* If an app is in the rare bucket, the system imposes strict restrictions on its ability to run jobs, trigger alarms, and receive high-priority FCM messages. The system also limits the app's ability to connect to the internet.
* For example, a hotel app that the user only runs while they're staying at that hotel..

**Battery Saver Improvement:**

When battery saver is turned on, the system places restrictions on all apps. This is an existing feature that is improved with Android 9.

* Device manufacturer determines the precise restrictions imposed.
* For example, AOSP builds the system applies certain restrictions:
* The system puts apps in app standby mode more aggressively, instead of waiting for the app to be idle.
* Background execution limits apply to all apps, regardless of their target API level.
* Location services may be disabled when the screen is off.
* Background apps do not have network access.