G53MDP
Mobile Device Programming

Lecture 17 – Power (1)

Batteries

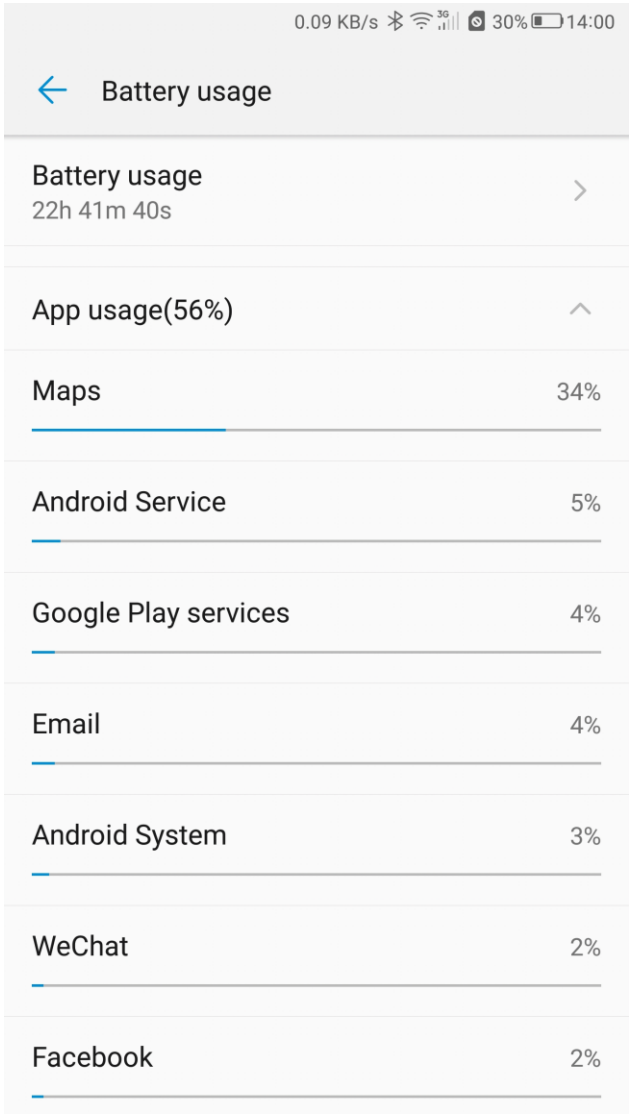
- Battery technology evolving relatively slow compare with the power consuming parts (e.g. larger screens, faster CPUs, faster network communications)
- Batteries have a limited power capacity
  - Power is the rate at which energy is used
  - Capacity measured in milliamp hours (mAh)
  - The amount of current that the battery can provide for one hour, before running out of charge
- 1000mAh battery can provide 1000mA (or 1A) for one hour
  - iPhone 7 has a 2900mAh battery
  - Laptop may have a 5800mAh battery

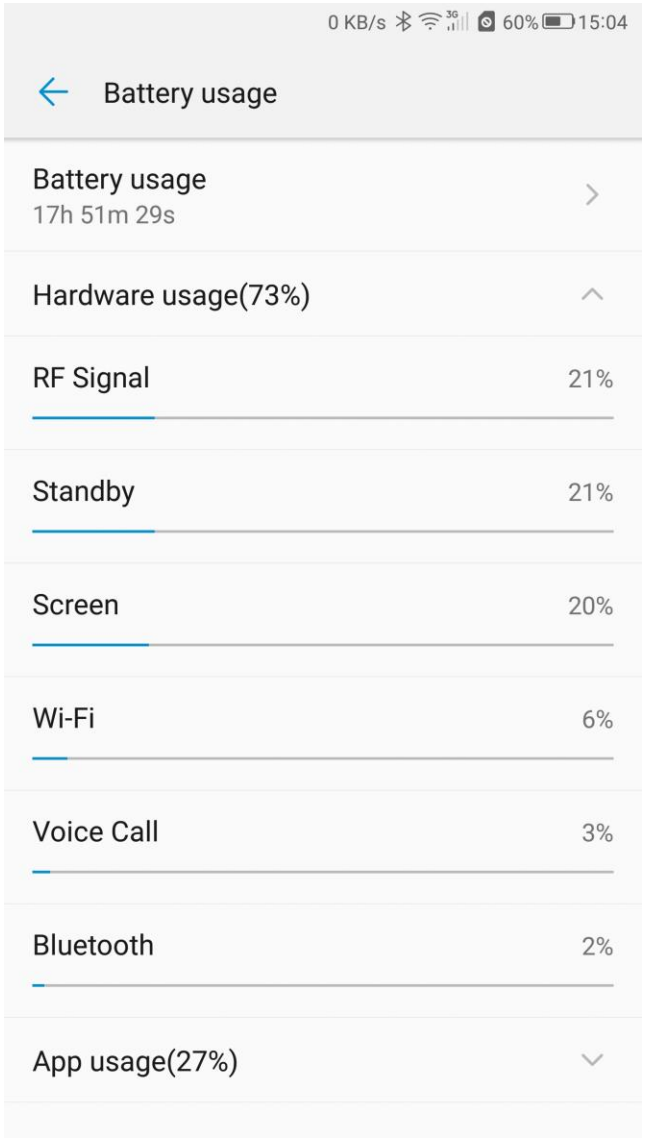
How can we save power for mobiles?

Learning Outcomes

- Battery and power consumption calculation
- Android Power Management
- Knowledge about different modes for battery life enhancement in Android (App Standby & Doze)
- Knowledge about firebase cloud messaging

Power Usage Profiles





Battery Consumption Statistics

- Framework tacks 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 times at specific speeds

Component Power Profile

- Manufacture Defined in: /frameworks/base/core/res/res/xml/power_profile.xml

```
<item name="none">0</item>
<item name="screen.on">0.1</item>  <!-- ~200mA -->
<item name="screen.full">0.1</item>  <!-- ~300mA -->
<item name="bluetooth.active">0.1</item> <!-- Bluetooth data transfer, ~10mA -->
<item name="bluetooth.on">0.1</item>  <!-- Bluetooth on & connectable, but not connected, ~0.1mA -->
<item name="wifi.on">0.1</item>  <!-- ~3mA -->
<item name="wifi.active">0.1</item>  <!-- WIFI data transfer, ~200mA -->
<item name="wifi.scan">0.1</item>  <!-- WIFI network scanning, ~100mA -->
```

- Power cost=system track time x power consumption values

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

Power Management Design

- A wrapper to Linux Power Management
- Added to the Kernel
  - Early Suspend framework
  - Partial 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 system's power
  - Overall system power consumption decreases
  - "Better" use of battery capacity

Early Suspend / Late Resume

- More modifications to the Linux kernel
  - Suspend as much as possible even if the kernel is still operating
- Early suspend
  - 200 levels of suspension **between** working and sleeping
    - EARLY_SUSPEND_LEVEL_BLANK_SCREEN = 50
    - EARLY_SUSPEND_LEVEL_STOP_INPUT = 75
    - EARLY_SUSPEND_LEVEL_STOP_DRAWING = 100
  - Hook into the ordering of the suspension process (smaller number is called first)
  - Attempts to suspend as much as possible
- Late resume
  - Kernel devices that were early_suspended are subsequently late_resumed
  - Can wake the kernel without waking up all peripherals
  - Resume suspended devices once the kernel is awake and working

Android PowerManager

- **“Device battery life will be significantly affected by the use of this API.**

  Do not acquire PowerManager.WakeLocks unless you really need them, use the minimum levels possible, and be sure to release them as soon as possible.”
- Apps and services must request CPU resource in order to keep power on, otherwise auto sleep (shut down CPU, suspend operational RAM)
- Use PowerManager.WakeLock object to control the power state of the device.

WAKE_LOCK

- Request the WAKE_LOCK permission in the manifest.
  - <uses-permission android:name="android.permission.WAKE_LOCK"/>
- Use newWakeLock() to create a PowerManager.WakeLock object

```
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
 PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK, "My Tag");
 wl.acquire();
   ..screen will stay on during this section..
 wl.release();
```

Flag Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On* (don't sleep even press power button)	Off	Off
SCREEN_DIM_WAKE_LOCK	On (sleep when press power button)	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On (sleep when press power button)	Bright	Off
FULL_WAKE_LOCK	On (sleep when press power button)	Bright	Bright

WakefulBroadcastReceiver

- Broadcast receiver that handles PARTIAL_WAKE_LOCK in a Service
  - To ensure the device is not in sleep while transitioning the work to a service.
  - Add the receiver in manifest: <receiver android:name=“.MyWakefulReceiver”></receiver>
  - Use startWakefulService() to start an intent service

Acquiring / Releasing WAKE_LOCK

- Request sent to PowerManager to acquire a wake lock
  - PowerManagerService notified to take a wake lock
  - Add wake lock to an internal list
  - Set the requested power state
  - If this is the first partial wake lock, take a kernel partial wake lock
  - For subsequent wake locks simply add to the list
- Request sent to PowerManager to release the wake lock
  - Wake lock removed from the internal list
  - If the wake lock is the last partial wake lock in the list, release the kernel wake lock
  - If kernel main wake lock is the only wake lock, release main kernel wake lock and device moves to suspend

Power Manager State

- adb shell dumpsys power

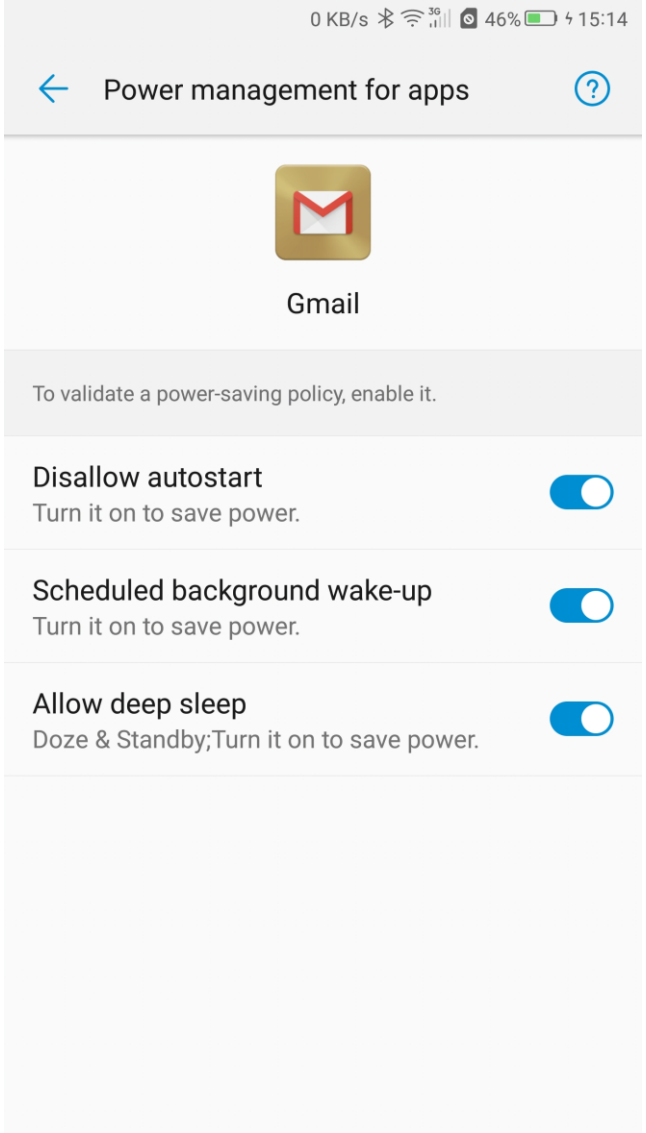
```
Wake Locks: size=4
  PARTIAL_WAKE_LOCK                'TaskSchedulerService_wakelock' ACQ=
  PARTIAL_WAKE_LOCK                'wake:com.google.android.gms/.auth.a
ms (uid=10014 pid=2388)
  PARTIAL_WAKE_LOCK                'wake:com.google.android.calendar/co
=2740)
  PARTIAL_WAKE_LOCK                'ScheduleNextAlarmWakeLock' ACQ=-136
```

Simple Method to Keep Screen On

- Alternative to WAKE_LOCK, a convenient screen on function: FLAG_KEEP_SCREEN_ON
  - Only in activity not in service or other component
- Implemented either in activity (flexible to turn off):
  - getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
- Or in Manifest:
  - Android:keepScreenOn="true";

Battery Life Enhancement in Android (Android 6.0 onwards)

- Designed for devices that have a default off behaviour
- **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

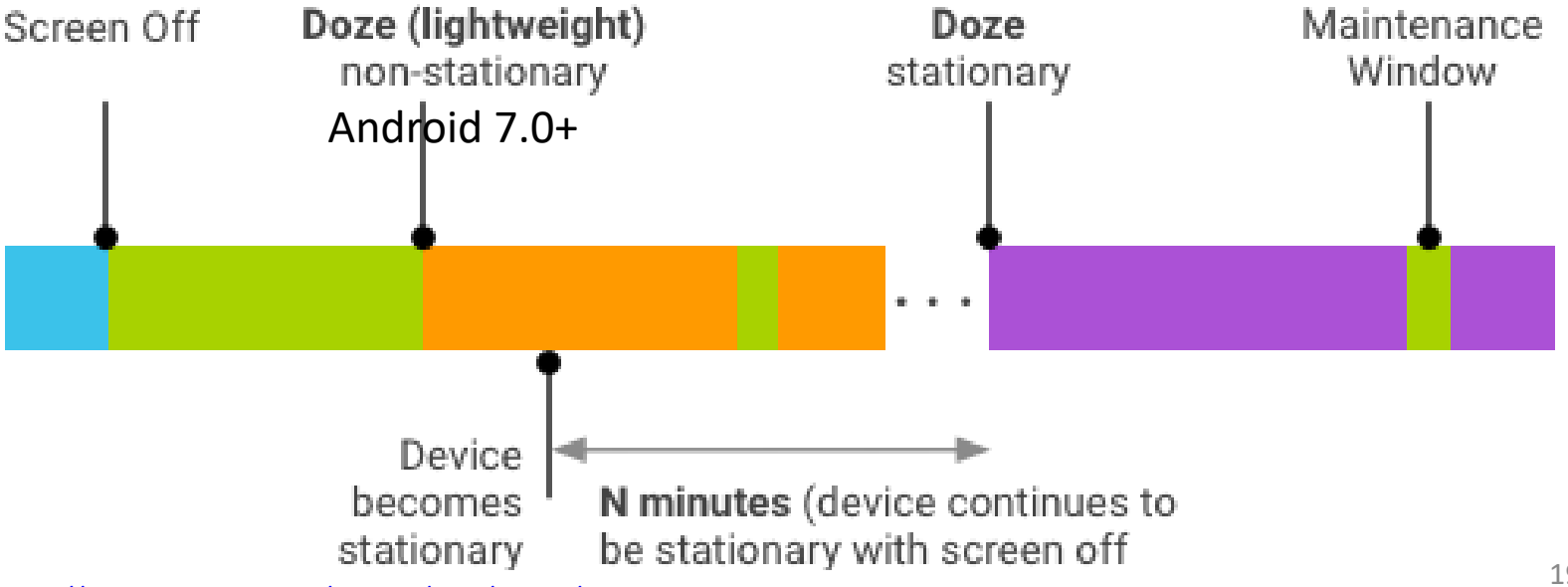


App Standby

- **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)
  - Hasn't 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

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



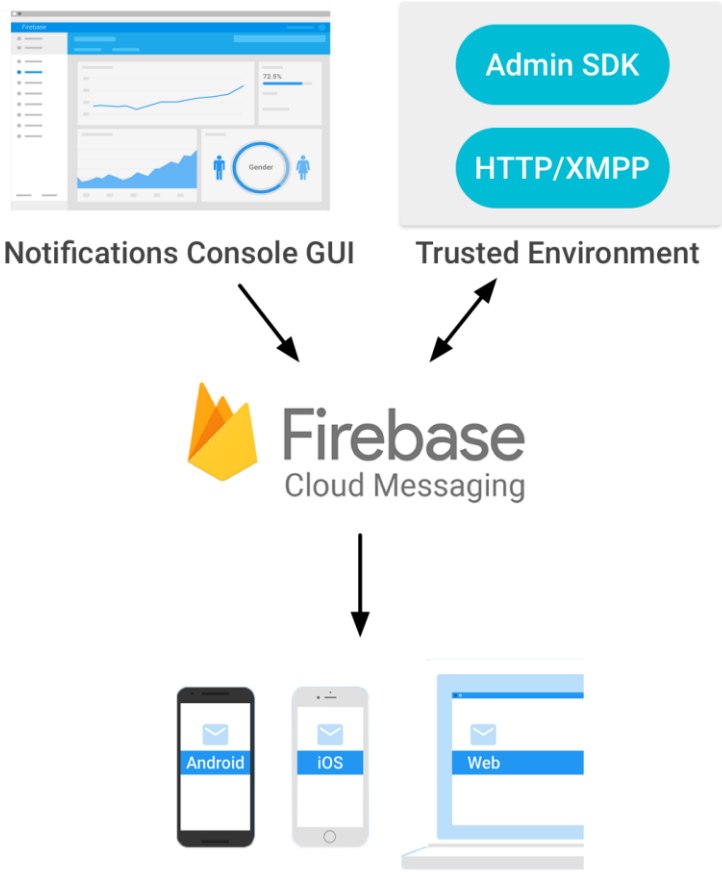
https://source.android.com/devices/tech/power/mgmt

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 can't use FCM high-priority messages or App's core function is affected

Firebase Cloud Messaging (FCM) (Replaced Google Cloud Messaging)

- A cross-platform reliable battery-efficient messaging solution that allows deliver message between server and client
- Provide single and persistent connection to the server
- Can notify a client app that new email or other data is available to sync.
- Can deliver to single device, groups of devices or devices subscribed to topics
- Send messages from client apps to server (e.g. chats, messages, etc.)



https://firebase.google.com/docs/cloud-messaging/

Whitelisted from Doze?

- Does the app need to run when screen off, stationary & on battery?
- Does the app can use FCM/FCM high priority messaging?
- Does the core functionality affected by Doze/Standby?
- Does the core functions need persistently requires a peripheral device for internet access?

- *WhatsApp*?
- The *activity tracker* for CW2?

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

Summary

- Battery and power consumption calculation
  - Calculate power cost by tacking time & use of power profile
- Android Power Management
  - Wake locks
- App Standby & Doze
  - Start/exit conditions & mode behaviours
  - What apps should be exemption from them?
- Firebase cloud messaging
  - Reliable battery-efficient messaging solution that allows deliver message between server and client

References

- https://developer.android.com/training/scheduling/wakelock.html
- https://developer.android.com/training/scheduling/alarms.html
- https://developer.android.com/reference/android/app/AlarmManager.html
- https://developer.android.com/topic/performance/scheduling.html