Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Android Development

More on API

Phone services

GSM/CDMA data

- To retreive the cell information the following classes can be used
  - `android.telephony.gsm.GsmCellLocation`
  - `android.telephony.cdma.CdmaCellLocation`
- The inquiry is made with using the `android.telephony.TelephonyManager`
  - `Context.getSystemService(Context.TELEPHONY_SERVICE)`
  - Additional useful information can be retrieved:
    - Status and type of the cellular network
    - Data traffic, name of operator, SIM status, roaming status, …
  - `getCellLocation()` – cell data
  - `List<NeighboringCellInfo> getNeighboringCellInfo()`

Specific cell data

- Information
  - RSSI, CID, LAC, PSC, NetworkType
    - Received Signal Strength
    - GSM cell ID
    - GSM LAC
    - PSC – The primary scrambling code in the UMTS network
    - NetworkType:
      - GPRS, EDGE, UMTS, HSDPA, HSUPA, HSPA

WiFi and Net

Wifi management

- `android.net.wifi` package
- Most important classes
  - `WifiManager`
  - `WifiConfiguration`
  - `WifiInfo`
- Additional classes
  - `WifiManager.WifiLock`
  - `WpsInfo`
  - `WifiManager.MulticastLock`
  - `WifiP2pManager`

WifiManager

- `Context.getSystemService(Context.WIFI_SERVICE)`
- Possibilities
  - Using the class the stored WiFi networks can be retreived
    - `List<WifiConfiguration> getConfiguredNetworks()`
    - `int addNetwork(WifiConfiguration config)`
    - `int updateNetwork(WifiConfiguration config)`
    - `boolean saveConfiguration()`
  - Available WiFi networks can be scanned
    - `boolean startScan()`
    - `List<ScanResult> getScanResults()`
  - Status
    - `boolean isWifiEnabled()`
    - `boolean setWifiEnabled(boolean enabled)`
    - `int getWifiState()`
  - Connecting
    - `boolean disconnect()`
    - `boolean reconnect()`
    - `boolean reassociate()`
    - `DhcpInfo getDhcpInfo()`

WifiLock

- The WiFi radio can be forced to be powered on while the device is sleeping (and the WiFi might be powered off otherwise)
  - It can not override the user settings or airplane mode, of course
  - Permission required
    - `android.permission.WAKE_LOCK`
  - Functions of the class
    - `acquire()`
    - `release()`
    - `isHeld()`
    - `setReferenceCounted(boolean refCounted)`
      - The `acquire` and `release` calls are counted and the lock is in effect while the number of `acquire` calls is greater than the number of `release` calls.

WifiConfiguration and WifiInfo

- WifiConfiguration
  - This class represents a configured WiFi network including the security information as well
    - Priority
    - NetworkID
    - SSID, BSSID
    - Keys
- WifiInfo
  - Represents active and connecting WiFi connections
    - MAC address
    - SSID, BSSID, RSSI
    - Link speed
    - IP address
    - Detailed network information

Managing the networks

- android.net package
  - More specific `ConnectivityManager`
  - Using that class the actual connection parameters can be retrieved

ConnectivityManager

- Taks
  - To monitor the connections
    - GPRS, WiFi, WiMax, UMTS
  - As the circumstances are changing Intents are sent to the receivers
  - A fail-over connection can be utilized
    - In case the WiFi area is not accessible an UMTS connection can be opened
- Most important functions
  - `getActiveNetworkInfo(), getAllNetworkInfo(),`
  - `getNetworkPreference(), …`
- NetworkInfo class
  - `getDetailedState(), getType(), isAvailable(), isConnected(), isFailover(), isRoaming(), …`
  - Type
    - Bluetooth, Ethernet, WiFi, WiMax, Mobile_*
- DHCPInfo

Additional classes

- `URI`, and `URI.Builder`
  - The URI class is abstract and immutable
  - A Builder is capable of creating URI-s
  - All paramteres can be set from the Builder
- `MailTo`
  - mailto:// protocol
    - Header, CC, BCC, Subject, To
- Querying the data transmitted/received
  - For different connection types
  - `TrafficStats`
- `LocalSocket`, `LocalServerSocket`, `LocalSocketAddress`
  - UNIX-domain Socket

Accessing network

- Previous API elements augment the traditional Java classes (java.net package)
  - `Socket`
  - `ServerSocket`
  - `URI`, `URL`, `Proxy`, …
- Furthermore `javax.net.ssl`
  - `SSLSocket`
  - `SSLServerSocket`

Bluetooth

Bluetooth

- android.bluetooth package
- Important classes
  - BluetoothAdapter
  - BluetoothDevice
  - BluetoothSocket
  - BluetoothServerSocket
- Further classes - profiles
  - BluetoothClass
  - BluetoothProfile
  - BluetoothHeadset
  - BluetoothA2dp
  - BluetoothHealth

Bluetooth profiles supported by Android

- Headset
  - Hands-Free Profile (HFP)
  - Headset Profile (HSP)
- Advanced Audio Distribution Profile (A2DP)
- Health Device Profile (HDP)
- Human Interface Device Profile (HID)
- Message Access Profile (MAP)
- Personal Area Networking Profile (PAN)
- Audio/Video Remote Control Profile (AVRCP)
- Generic Attribute (Gatt, HOGP, BTLE): Bluetooth 4.0
- Generic Attribute Server (Gatt Server, BTLE): Bluetooth 4.0

BluetoothAdapter

- To represent a local Bluetooth adapter
  - Required permissions
    - BLUETOOTH
    - BLUETOOTH_ADMIN

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    // There is no Bluetooth in the phone
}
```

BluetoothAdapter

- Turning on and off
  - `enable()` and `disable()` function
  - However instead of that functions:

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

  - In that case the permission is requested from te user
    - If he/she grants it, the system enables the Bluetooth
- The name of the local device
  - `getName()` and `setName(String s)`

BluetoothAdapter

- Discovering remote devices
  - `cancelDiscovery()`, `startDiscovery()`, `isDiscovering()`
  - `getScanMode()`
- Following actions are sent about the progress of the process:
  - `ACTION_DISCOVERY_STARTED`
  - `ACTION_DISCOVERY_FINISHED`
  - `ACTION_FOUND`

BluetoothAdapter

- List of paired devices
  - `Set<BluetoothDevice> getBondedDevices()`
- Direct connection using hardware address
  - `BluetoothDevice getRemoteDevice(String address)`
- Checking adress
  - `checkBluetoothAddress(String address)`
- Connection
  - `BluetoothServerSocket mBluetoothAdapter.listenUsingRfcommWithServiceRecord(NAME, MY_UUID);`
  - Socket as server

BluetoothDevice

- Represents a remote device
  - Creating a connection
  - Retrieving information
  - `getBondState(), getAddress(),`
  - `BluetoothClass getBluetoothClass()`
    - Can describe the Bluetooth capabilities
- Connection
  - `BluetoothSocket createRfcommSocketToServiceRecord(UUID uuid)`
  - Creating a Socket as a client to connect to another device

Using a Bluetooth Profile

- Required steps
  1. An instance of the default adapter
  2. Proxy to the appropriate profile object `getProfileProxy()`
  3. Setting up a `ServiceListener`
  4. Waiting for the `onServiceConnected()` call, to access the proxy
  5. From this point, the Bluetooth state can be monitored and the incoming information can be processed

Using a Bluetooth Profile

- Example – for a headset

```
BluetoothHeadset mBH;
BluetoothAdapter mBA = BluetoothAdapter.getDefaultAdapter();
private BluetoothProfile.ServiceListener mPL= new
BluetoothProfile.ServiceListener() {
  public void onServiceConnected(int profile, BluetoothProfile proxy) {
    if (profile == BluetoothProfile.HEADSET) {
      mBH = (BluetoothHeadset) proxy;
    }
  }
  public void onServiceDisconnected(int profile) {
    if (profile == BluetoothProfile.HEADSET) { mBH = null; }
  }
};
mBA.getProfileProxy(context, mPL, BluetoothProfile.HEADSET);
mBA.closeProfileProxy(mBH);
```

PowerManager
BatteryManager

PowerManager

- Using the PowerManager it can be prevented that the device starts to „sleep"
  - You have to have a serious reason to do so, as the battery lifetime is affected
    - To measure continuously
- Following table summarizes the operation modes
  - * denotes cases, when use cannot override the lock

Flag Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On*	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

PowerManager

- How to use:

```
PowerManager pm =
     (PowerManager) getSystemService(Context.POWER_SERVICE);
PowerManager.WakeLock wl = pm.newWakeLock(
     PowerManager.SCREEN_DIM_WAKE_LOCK, "My Tag");
wl.acquire();
// Taks, when the device cannot sleep
wl.release();
```

- Further possibilities
  - Querying the status
  - Specifying what should happen when a lock is released

BatteryManager

- Contains contants to describe the battery status
  - The infromation can be retrieving by receiving broadcast messages
  - Actions:
    - ACTION_BATTERY_CHANGED
    - ACTION_BATTERY_LOW
    - ACTION_BATTERY_OKAY
    - ACTION_POWER_CONNECTED
    - ACTION_POWER_DISCONNECTED
  - States of ACTION_BATTERY_CHANGED

BATTERY_HEALTH_COLD	BATTERY_PROPERTY_CAPACITY
BATTERY_HEALTH_DEAD	BATTERY_PROPERTY_CHARGE_COUNTER
BATTERY_HEALTH_GOOD	BATTERY_PROPERTY_CURRENT_AVERAGE
BATTERY_HEALTH_OVERHEAT	BATTERY_PROPERTY_CURRENT_NOW
BATTERY_HEALTH_OVER_VOLTAGE	BATTERY_PROPERTY_ENERGY_COUNTER
BATTERY_HEALTH_UNKNOWN	BATTERY_STATUS_CHARGING
BATTERY_HEALTH_UNSPECIFIED_FAILURE	BATTERY_STATUS_DISCHARGING
BATTERY_PLUGGED_AC	BATTERY_STATUS_FULL
BATTERY_PLUGGED_USB	BATTERY_STATUS_NOT_CHARGING
BATTERY_PLUGGED_WIRELESS	BATTERY_STATUS_UNKNOWN

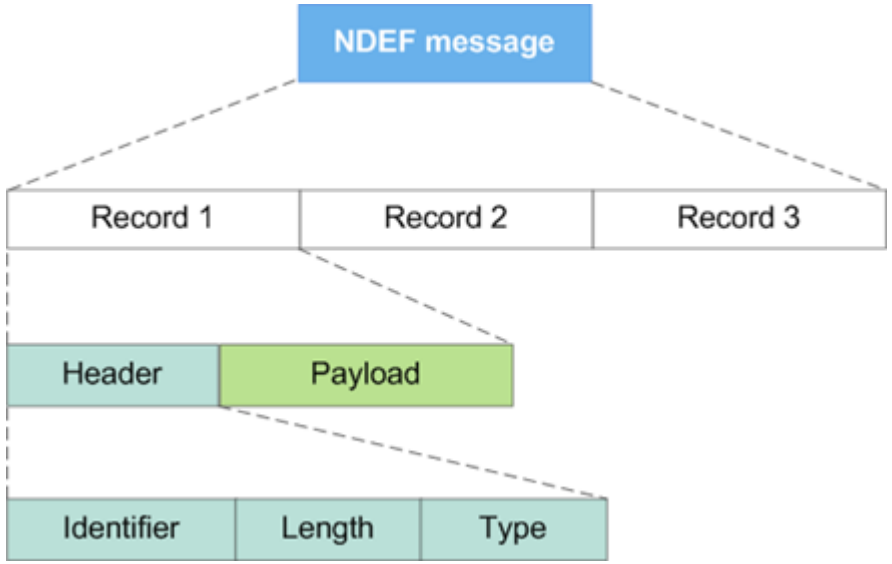
Vibration

- `android.os.Vibrator` class
- Corresponding function calls
  - `cancel()`
  - `hasVibrator()`
  - `vibrate(long millisec)`
  - `vibrate(long[] pattern, int repeat)`
- As the application process finished all the vibration will be cancelled.

NFC

NFC data structure

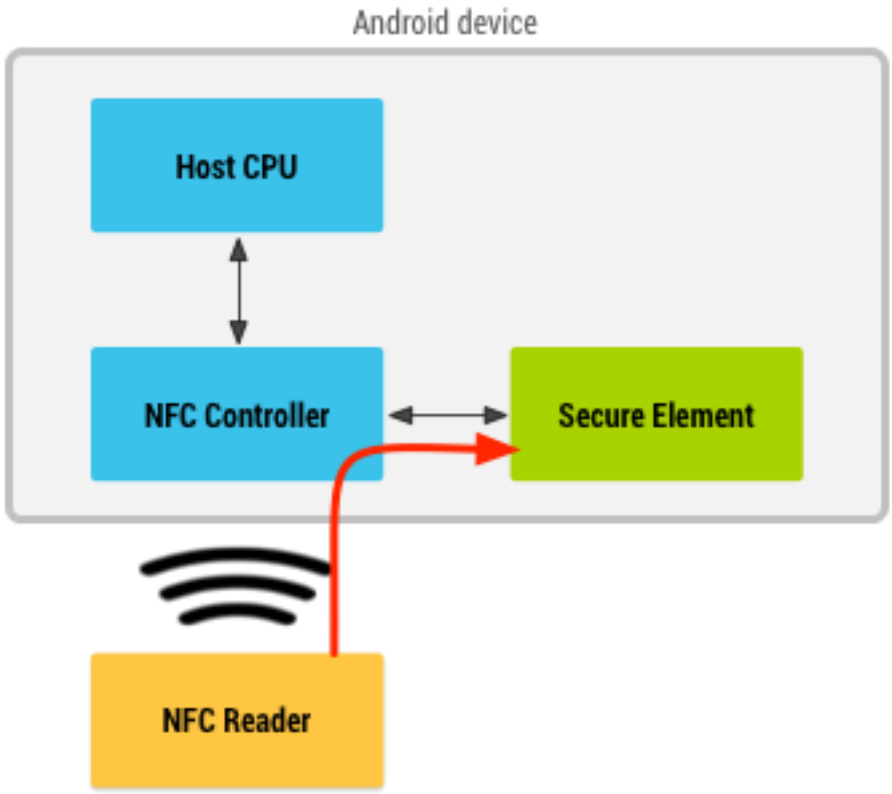
- NDEF message
- ID = TNF (3 bit)
  - URI: 0x0003
  - Empty: 0x0000
  - External: 0x0004
  - MIME: 0x0002
  - Well Known RTD 0x0001
  - Unknown 0x0005
  - Unchanged 0x0006
- RTD
  - URI, TEXT, Smart Poster, …

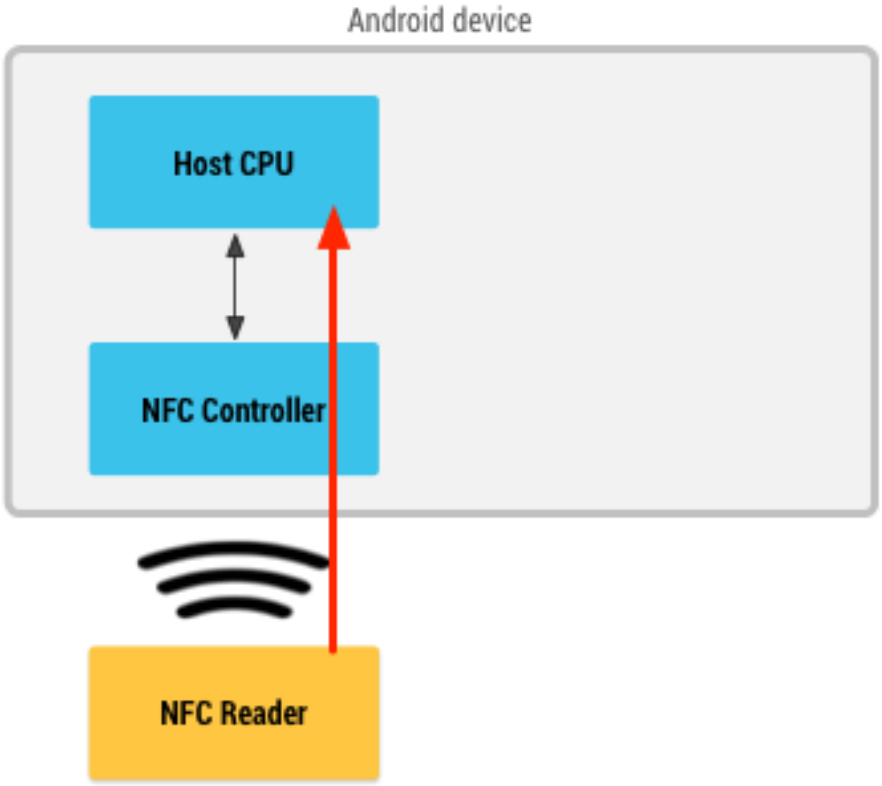


NFC and Android

- Three different operation mode
  - Write/read: passive NFC tags
  - P2P: between two Android devices
  - Card emulation: the phone can emulate tags, which can be read by other devices

Card Emulation

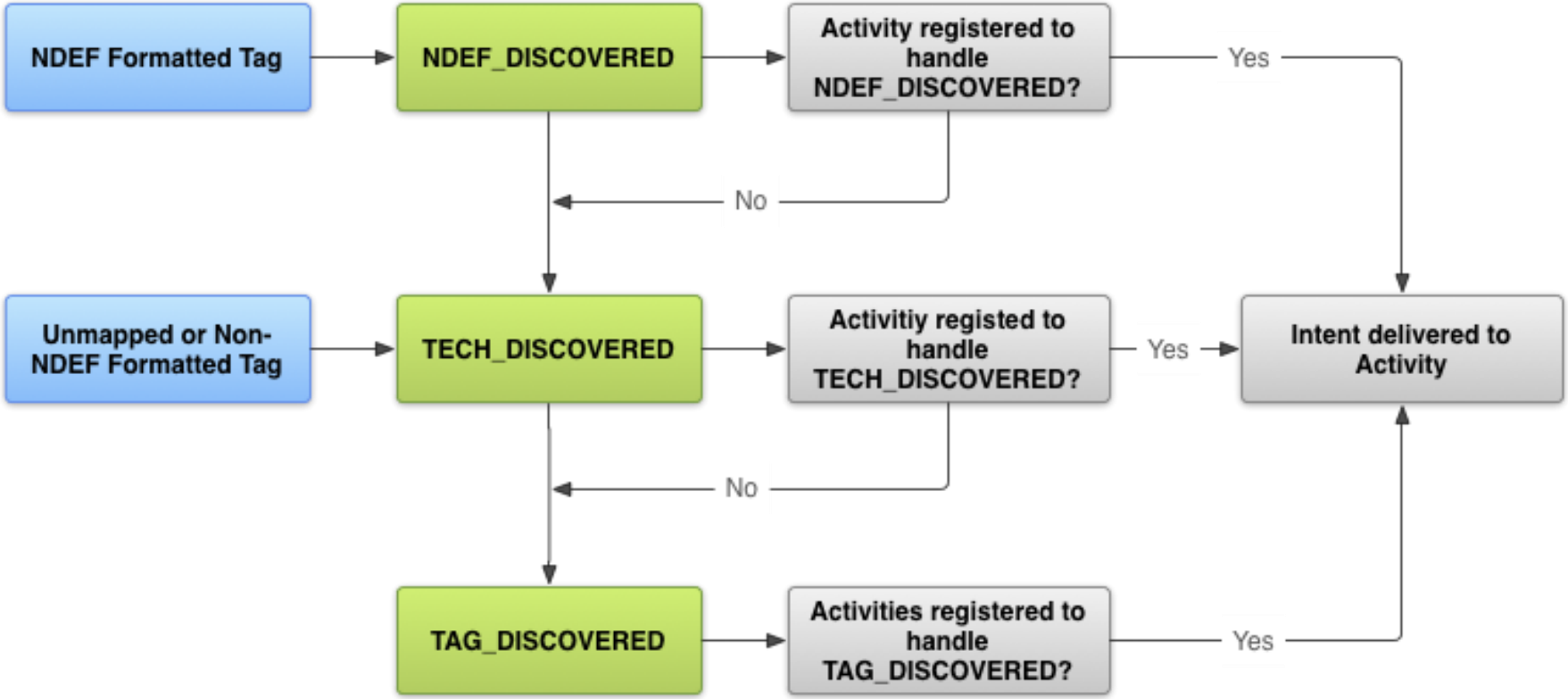




Reading Tags

- As the range of NFC communication is short and the time while the NFC is in range is also short the corresponding Activity is automatically found by the Android system
  - Reading NFC tag
  - The MIME type is processed
  - The data is put into an `Intent`
- The most widely the NDEF tags are supported
  - In `NdefMessage` there is one or more `NdefRecord`
- Furthermore it is possible to read other data

Handling Intents



Defining Intent filters

-
```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain" />
</intent-filter>
```

-
```
<activity>
...
<intent-filter>
    <action android:name="android.nfc.action.TECH_DISCOVERED"/>
</intent-filter>

<meta-data android:name="android.nfc.action.TECH_DISCOVERED"
    android:resource="@xml/nfc_tech_filter" />
...
</activity>
```

Android Beam

- Peer-to-peer communication between two Android devices
  - The sending application should be in foreground
  - The recipient device should be unlocked
  - AAR: Android Application Record
- Receiving:

```
<intent-filter>
  <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <data android:mimeType="application/vnd.com.example.android.beam"/>
</intent-filter>
```

API Evolution

Lollipop – Project Volta

- `JobScheduler`
  - Performing low priority tasks later, when circumstances are different
    - While charging
    - When Wi-Fi is available
  - The task should be put into a `JobInfo` object
    - Parameters can be set
  - Example

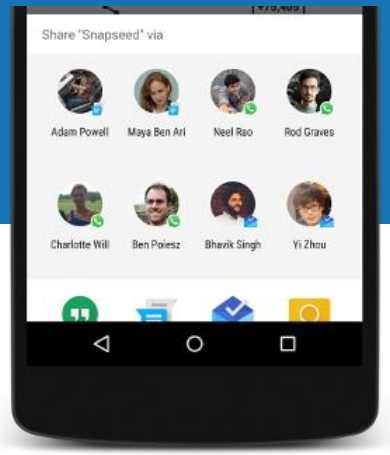
```
val uploadTask: JobInfo = JobInfo.Builder(
        jobId,
        serviceComponent /* JobService component */
).run {
    setRequiredNetworkType(JobInfo.NETWORK_TYPE_UNMETERED)
    build()
}
val jobScheduler =
context.getSystemService(Context.JOB_SCHEDULER_SERVICE) as
JobScheduler
jobScheduler.schedule(uploadTask)
```

Lollipop – Usage Log

- `android.app.usage` API
- Permission required:
  `android.permission.PACKAGE_USAGE_STATS`
- The user should enable it as well
- Aggregated data can be accessed (daily, weekly, stb.)
- Data collected
  - When was it used last
  - How much time in foreground
  - Changes of the configuration (pl. screen rotating)

Marshmallow

- Doze and App Standby
  - If a user unplugs a device and leaves it stationary, with its screen off, for a period of time, the device goes into **Doze** mode, where it attempts to keep the system in a sleep state
  - **App Standby** allows the system to determine that an app is idle when the user is not actively using it
- Fingerprint Authentication
- App Linking
  - This feature allows to associate an app with a web domain
- Auto Backup for Apps
  - Automatic full data backup and restore for apps

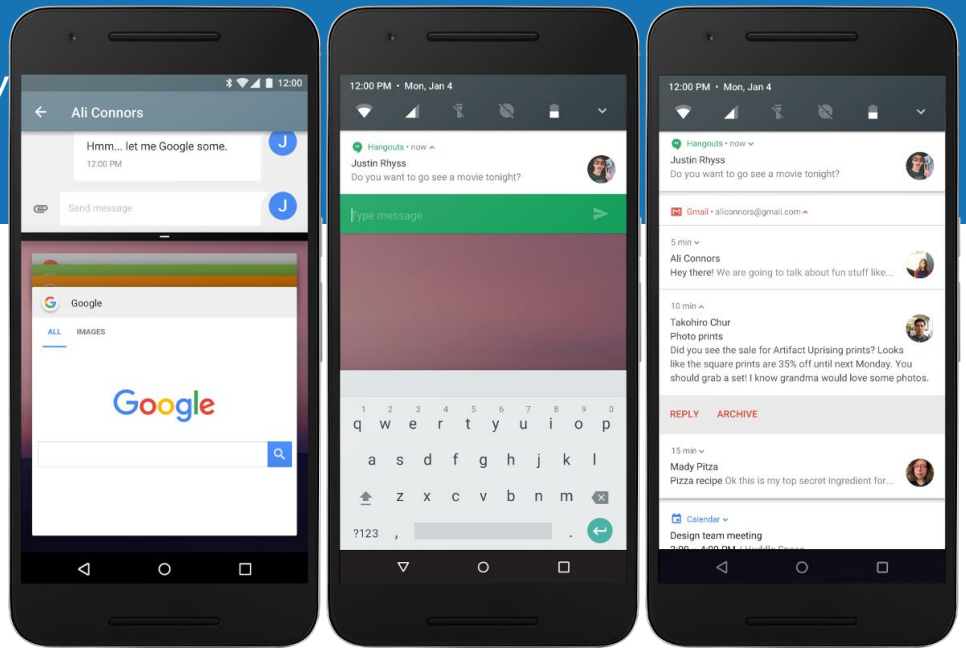


Marshmallow

- Direct Share
  - API to make sharing intuitive and quick for users
- Voice Interactions
  - New voice interaction API which, together with Voice Actions
- Confirm Credential
  - To authenticate users based on how recently they last unlocked their device
- Adoptable Storage Devices
  - Users can adopt external storage devices such as SD cards
  - Adopting an external storage device encrypts and formats the device to behave like internal storage
- Notifications
  - Some improvements

Marshmallow

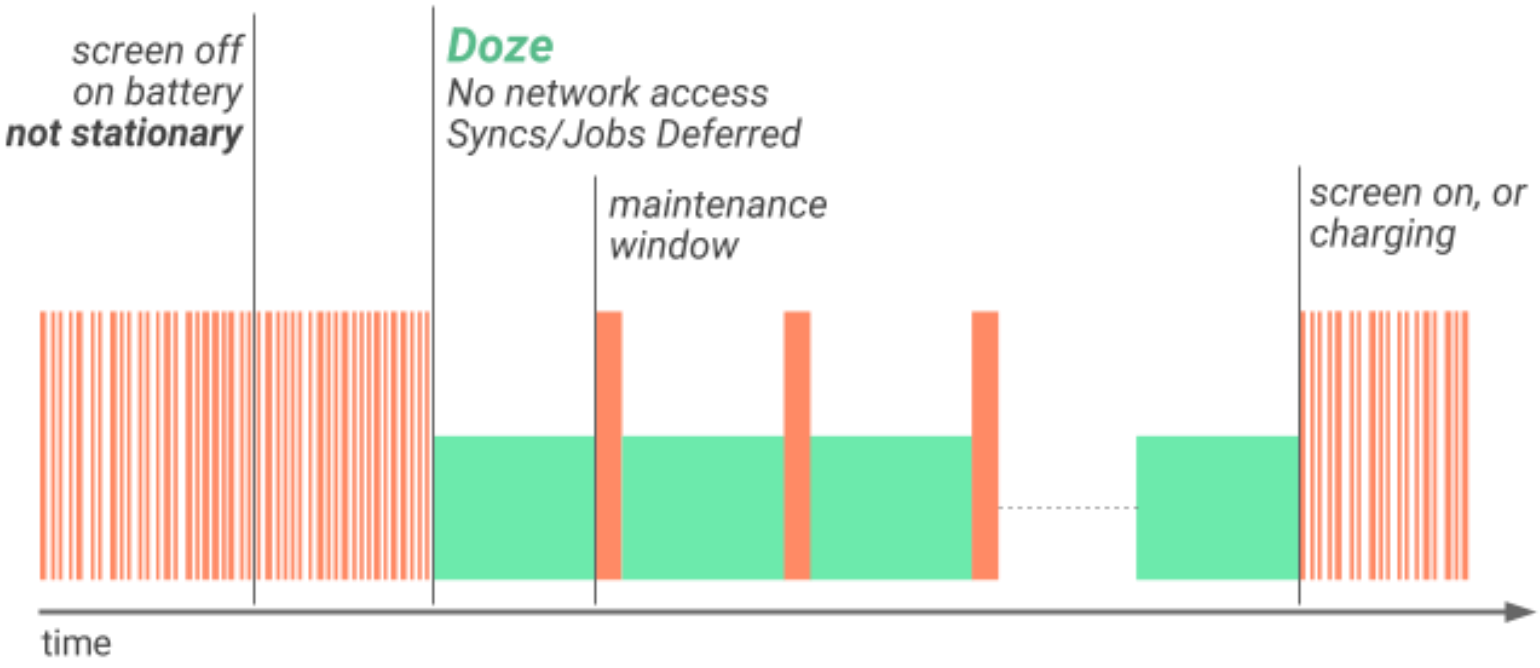
- Bluetooth Stylus Support
- Improved Bluetooth Low Energy Scanning
- Hotspot 2.0
- 4K Display Mode
  - Allows apps to request that the display resolution be upgraded to 4K rendering on compatible hardware
- Audio Features
  - Support for the MIDI protocol
- Video Features
  - Adds new capabilities to the video processing APIs
- Camera Features
  - Flashlight API
  - Reprocessing API



Nougat

- Multi-window Support
- Notification Enhancements
  - Template updates
  - Messaging style customization
  - Bundled notifications
  - Direct reply
  - Custom views
- Profile-guided JIT/AOT Compilation
  - Just in Time (JIT) compiler with code profiling to ART
    - Lets it constantly improve the performance of Android apps as they run
- Doze on the Go...
  - a step further and saves battery while on the go

Doze


screen off
on battery
not stationary
Doze
No network access
Syncs/Jobs Deferred
maintenance
window
screen on, or
charging
time

Nougat

- Project Svelte: Background Optimization
  - To minimize RAM use by system and apps across the range of Android devices in the ecosystem
- SurfaceView
  - Provides better battery performance than TextureView
- Data Saver
  - Helps reduce cellular data use by apps, whether roaming, near the end of the billing cycle, or on a small prepaid data pack
- Vulkan API
  - New 3D rendering API
  - Vulkan development tools and libraries are rolled into the Android 7.0DK. They include:
    - Headers
    - Validation layers (debug libraries)
    - SPIR-V shader compiler
    - SPIR-V runtime shader compilation library

Nougat

- Quick Settings Tile API
- Number Blocking
  - Android provides a consistent way for apps to support number blocking across a wide range of devices
  - Among the other benefits that apps can take advantage of are:
    - Numbers blocked on calls are also blocked on texts
    - Blocked numbers can persist across resets and devices through the Backup & Restore feature
    - Multiple apps can use the same blocked numbers list
- Call Screening
  - A number of actions based on an incoming call's Call.Details
    - Reject the incoming call
    - Do not allow the call to the call log
    - Do not show the user a notification for the call
- Multi-locale Support, More Languages
- New Emojis

Nougat

- WebView
  - Chrome + WebView, Together
  - Multiprocess
  - Javascript run before page load
  - Geolocation on insecure origins
- OpenGL ES 3.2 API
- Android TV Recording
- Accessibility Enhancements
- Direct Boot
  - Improves device startup times and lets registered apps have limited functionality even after an unexpected reboot
    - If an encrypted device reboots while the user is sleeping, registered alarms, messages and incoming calls can now continue to notify the user as normal
- APK Signature Scheme v2

Nougat

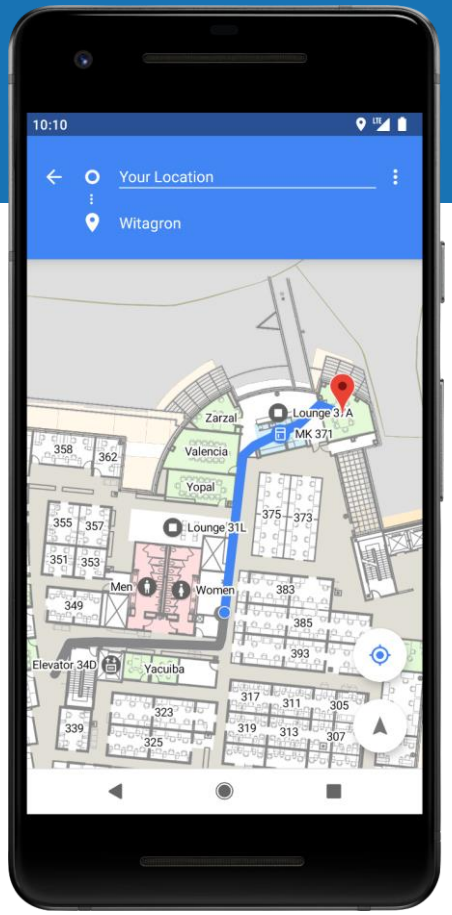
- VR Support
  - platform support and optimizations for a new VR Mode
  - a number of performance enhancements, including access to an exclusive CPU core for VR apps
- Print Service Enhancement
- Frame Metrics API
  - to monitor its UI rendering performance
- Virtual Files
- Sustained Performance API
  - To enable OEMs to provide hints about device-performance capabilities for long-running apps
- Custom Pointer API
- Keyboard Shortcuts Helper

Oreo

- New functions
  - Notification Channels
    - to provide a unified system to help users manage notifications
  - Picture in picture
    - PIP is a special type of multi-window mode mostly used for video playback
  - Autofill
    - Android O makes filling forms, such as account and credit card forms, easier with the introduction of the Autofill Framework
  - Adaptive icons
    - adaptive launcher icons, which can display a variety of shapes across different device models
- Several API Changes

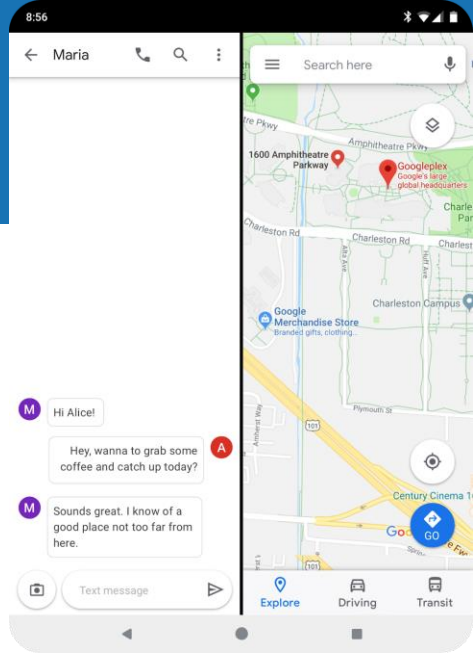
Pie

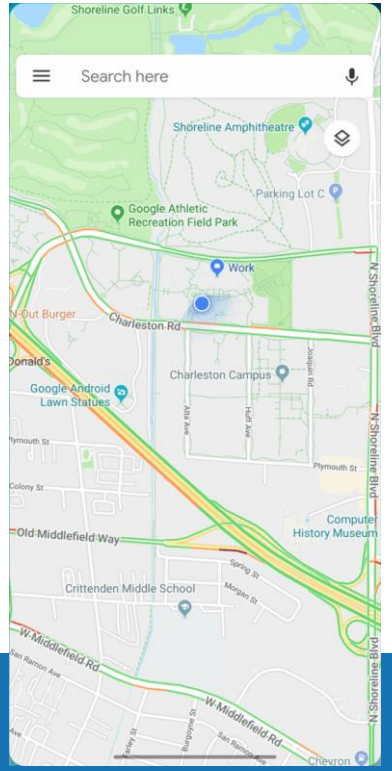
- New features
  - Indoor positioning with Wi-Fi RTT
  - Display cutout support
  - Enhancement of notifications
  - Multi-camera support
  - HDR VP9 Video, HEIF image compression
  - Data cost sensitivity in JobScheduler
  - Neural Networks API 1.1
  - Security enhancements



Q – Android 10

- New functions
  - Foldables
    - Robus multi-window support.
  - 5G networks
    - Platform support for 5G
  - Smart Reply in notifications
  - Dark Theme
  - Gesture navigation
  - Sharing shortcuts



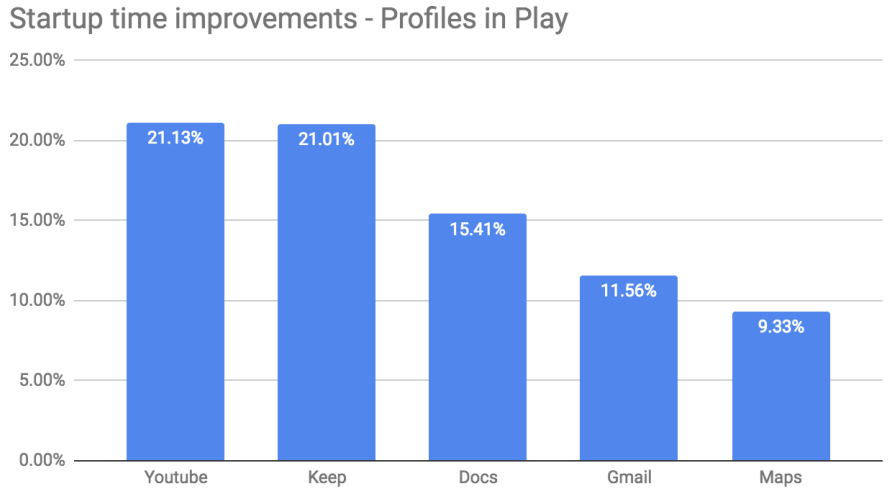


Q – Android 10

- New functions
  - Security improvements
    - Storage encryption
    - TLS 1.3 by default
    - Platform hardening
    - Improved Biometrics
  - Camera and media
    - Dynamic depth for photos
    - Directional, zoomable microphones
    - …

Q – Android 10

- Android foundations
  - ART optimizations
  - Neural Networks API 1.2
  - Thermal API
    - CPU, GPU thermal queries



Homework

- Second checkpoint!
  - Deadline – next week! (05/06)
  - I'll assess your progress and provide feedback.

Google Play

Next week