Australian National University

Android

Fundamentals

Gaurav Mitra

Scope

Topics to be covered, and assessed:

- Android Fundamentals
- Android Studio Overview
- App Components (Activity, Service, Content Provider, Broadcast Receiver, Intent etc)
- User Interface or UI (View, Layouts, Input Controls, Input Events, Event Handlers, Menus, Notifications etc)
- Persistent Data Storage

Scope

Topics beyond our scope (NOT to be covered):

- UI (Accessibility, Custom Components, Styles & Themes)
- Animation & Graphics
- Computation with Renderscript
- Media & Camera
- Connectivity (Bluetooth, NFC, Wifi P2P, USB, SIP)
- Synchronization via Cloud resources
- Web Apps

Why Android?

- Smartphones are ubiquitous
- Four major smartphone operating systems - Android, iOS, Windows Phone, Blackberry OS
- Android dominates the smartphone OS market share[1]
  - Q2, 2015: Android (**82.8%**); iOS (13.9%); Windows Phone (2.6%); Blackberry OS (0.3%); Other (0.4%)
  - Q2, 2014: Android (**84.8%**); iOS (11.6%); Windows Phone (2.5%); Blackberry OS (0.5%); Other (0.7%)
- Free and Open source[2]
- Supports multiple instruction-set architectures (ARM, x86 etc) and a variety of hardware platforms ranging from smartphones, tablets, laptops, set-top-boxes to watches, and soon.. cars!
- Built to run on top of the Linux kernel
- Focus on *touch-based* input i.e. UI is a major design priority

[1] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[2] https://source.android.com/.

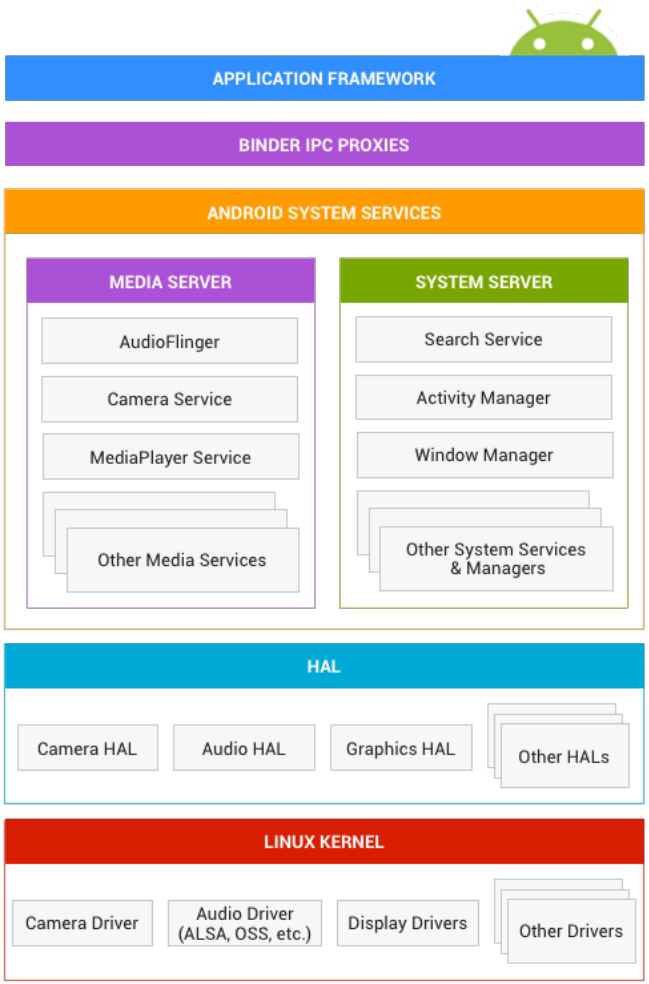
Software Construction Perspective

- Android incorporates the use of multiple software design patterns
- Creating an android app requires knowledge of
  - Java programming
  - Adaptable software design - A single app could target many different devices
  - Responsive UI design
  - Security and permissions
  - Mark-up languages such as XML
  - Persistent storage solutions using databases
  - Inter-process communication - Often between apps
  - Network communication
- Modularity is key in Android's *layered* architecture

Layered System Architecture

Loosely coupled layers built on top of each other[3]

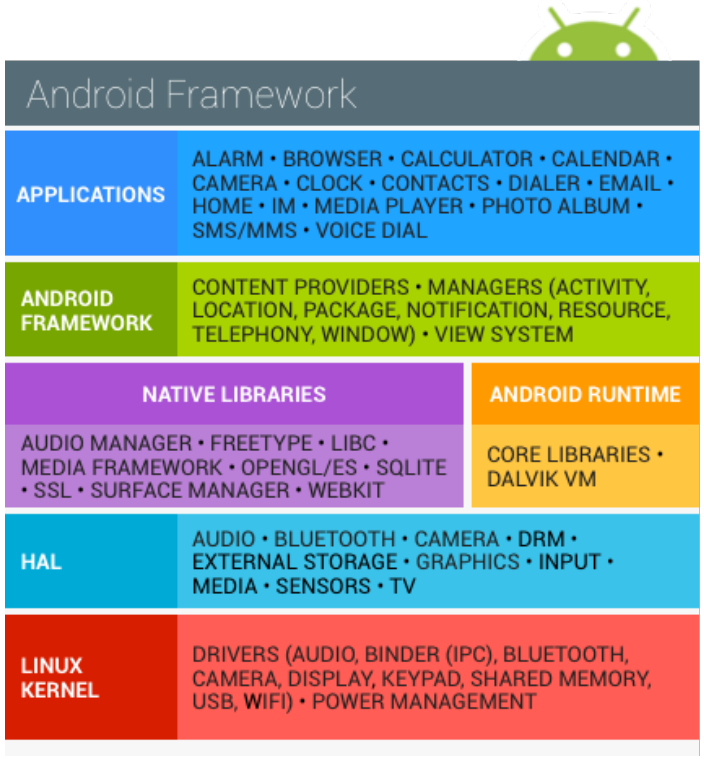
- *Application Framework*: For app developers. Many developer APIs map directly to the underlying HAL interfaces
- *Binder IPC Proxies*: Allows apps to make system calls to the System Server. This communication is hidden from the app developer
- *System Services*: Modular system and media services
- *Hardware Abstraction Layer (HAL)*: Defines standard interface for the hardware vendors to implement. Allows upper android layers to be device agnostic
- *Linux Kernel*: Special build using additions such as *wake locks*



[3] https://source.android.com/devices/.

Layered Software Stack

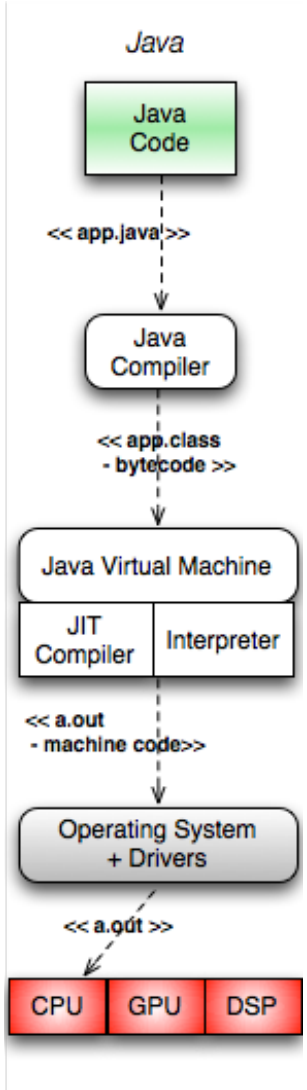
- *Loosely coupled* layers built on top of each other[a]
- *Android Framework*: For app developers. Our scope is limited to this layer only.
- *Android Runtime*: Dalvik Java virtual machine (VM). Most apps run on this VM
- *Native Libraries*: Some apps are run natively such as core services



[a] https://source.android.com/security/index.html.

Java Virtual Machine

- Java programs are device/processor agnostic
- Source compiled to Java *bytecode* intermediate representation (IR)
- Bytecode is executed by a JVM
- JVM compiles bytecode to machine code *Just-in-time* (JIT) i.e. during the execution of the program
- JIT compilation is a combination of *Ahead-of-time* (AOT) compilation and *interpretation*
- Machine code executed by OS on CPU



Android's Dalvik JVM

Design Requirements:

- Must work on resource constrained hardware - smartphones a few years back had limited amounts of RAM and processing power
- Application Sandbox - to ensure security, performance and reliability
- Lack of swap space
- Battery powered

Android's Dalvik JVM

Dalvik Design Features[4]:

- Every Android app runs in its own process, with its own instance of Dalvik VM
- Multiple Dalvik VM's can run together efficiently
- Optimized intermediate representation: *Dalvik Executable* (.dex) - Minimal memory footprint
- *dex* tool compiles .class files to .dex files
- Since each app runs an instance of Dalvik, both creation and execution of new Dalvik instances must be efficient
- Dalvik relies on the Linux kernel to provide threading and low-level memory management

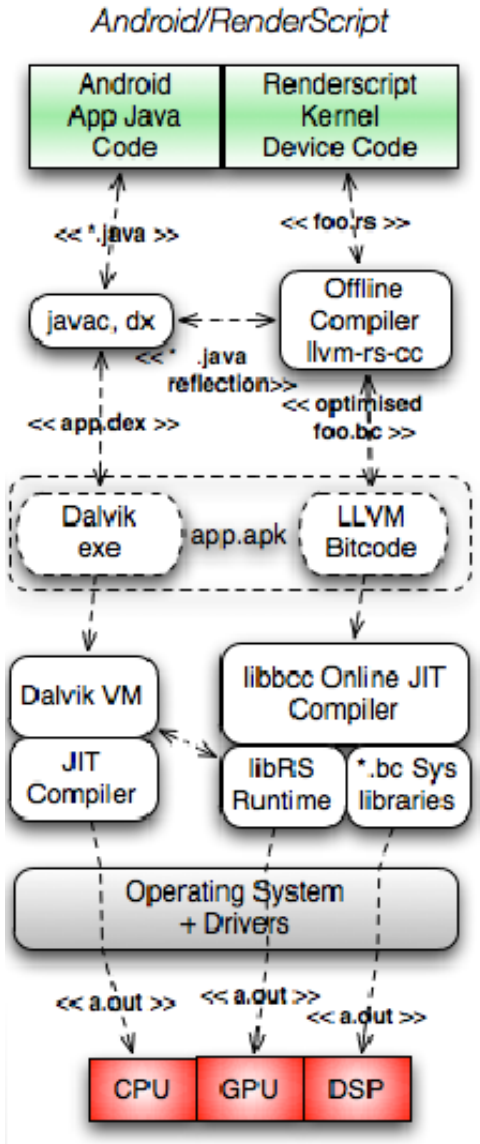
Dalvik has now been replaced with the improved ART (Android Runtime). However, ART remains compatible with Dalvik and runs Dex bytecode[5]

[4] http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf.

[5] https://source.android.com/devices/tech/dalvik/.

Android Code Path

- A typical smartphone has multiple prcessing elements such as a CPU, GPU and DSP
- Java code runs in the Dalvik or ART JVM which runs on the CPU
- How are the GPU and DSP cores used? *OpenGL* and *Renderscript*
- Renderscript code is compiled to *LLVM IR bitcode*
- LLVM Bitcode is JIT compiled and executed on the GPU or DSP



Security Sandbox

No app, by default, has permission to perform any operations that would adversely impact other apps, the OS or the user[6]

- Android is a *privilege-separated* OS
- Each application runs in isolation, in a *process sandbox*, with its own system identity (Linux UID, GID)
- Additional security provided via *permissions*
- Permissions enforce restrictions on specific operations that particular processes can perform
- Apps prevented from reading/writing user's private data, other app's files, unauthorized network access, keeping device awake etc
- The Dalvik VM is not a hard security boundary though - apps may still execute native code using the *Android NDK*

[6] http://developer.android.com/guide/topics/security/permissions.html.

Android Learning Resources

- Google developer docs:
  `http://developer.android.com/guide/index.html`
- API reference:
  `http://developer.android.com/reference/packages.html`
- Video training:
  `http://developer.android.com/training/index.html`
- Multiple books available

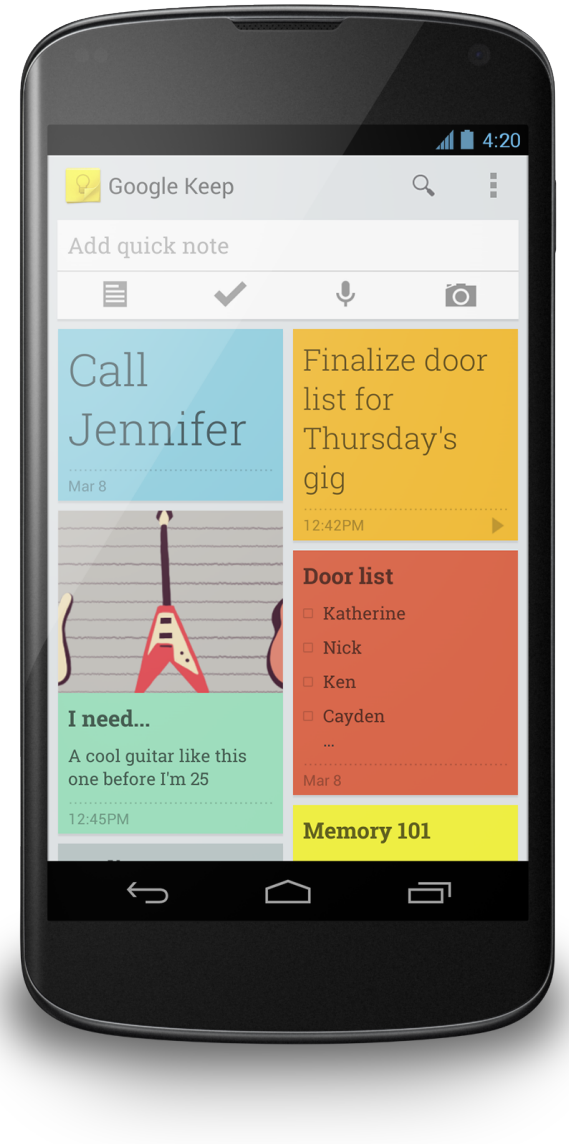
Android Version History and API Levels[7]

- *Android Alpha*: First release - November 2007
- *Android 1.0*: First commercial release - API Level 1.0 - September 2008
- Release schedule stabilized to two major releases every year 2010
- ...
- *Ice Cream Sandwich*: Version 4.0-4.0.4 - API level 15 - October 2011 - 97% of Android devices today have an API level equal or higher than ICS
- ...
- *Lollipop*: Version 5.0-5.1.1 - API level 22 - November 2014
- *Marshmallow*: Most recent version 6.0-6.0.1 - API level 23 - October 2015

[7] https://en.wikipedia.org/wiki/Android_version_history.

Assignment 1 - Overview

- Objective: Build an Android app that can be used to *create*, *store*, *edit* and *delete* simple notes
- A good example is Google's own app - Keep[a b]
- Assignment1 Specification is up
- Use minimum SDK version 15
- Labs 3 and 4 will focus on Android development
- Source code to be committed via git to CECS gitlab repository
- Week 7 due date: ~~28/03/2016 23:59 AEDT~~ 29/03/2016 10:00 AEDT
- No commits can be pushed after this time
- Final demo due in Week 7 labs



[a] https://www.google.com/keep/.

[b] https://googleblog.blogspot.com.au.