Introduction to Android

Simone Gasparini

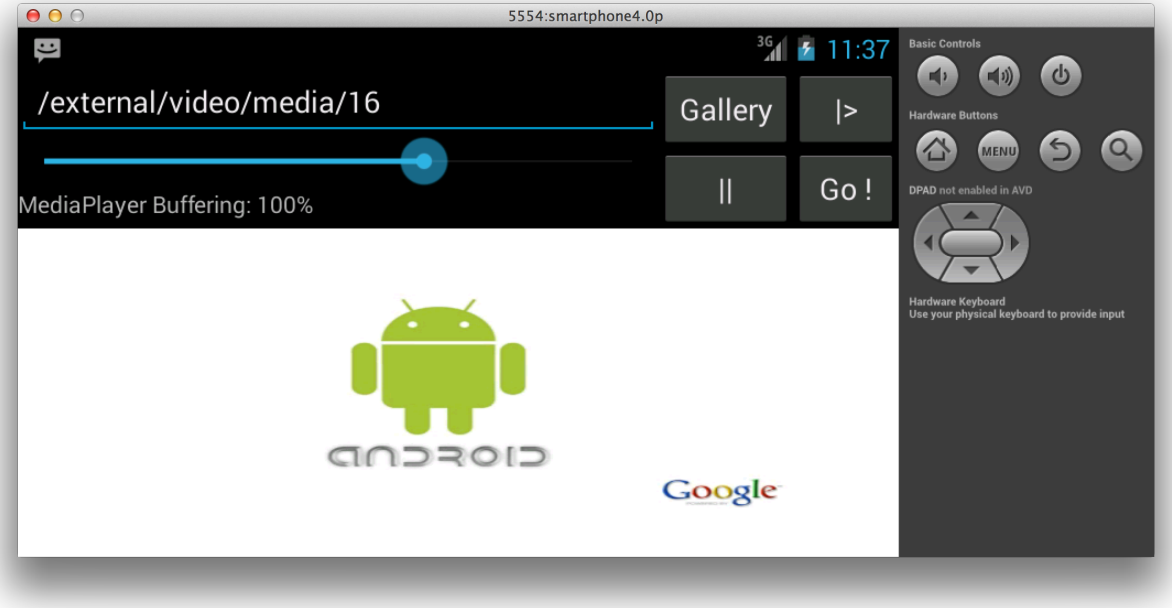
simone.gasparini@enseeiht.fr

Module Organization

- 8 Classes on introduction to Android programming
  - Introduction to Android
  - Android tools
  - Android programming
    - Intents
    - UI widgets
  - Android Camera
  - Native programming with JNI
    - Computer vision / Image processing on Android with OpenCV
  - High-Performance computing with RenderScript

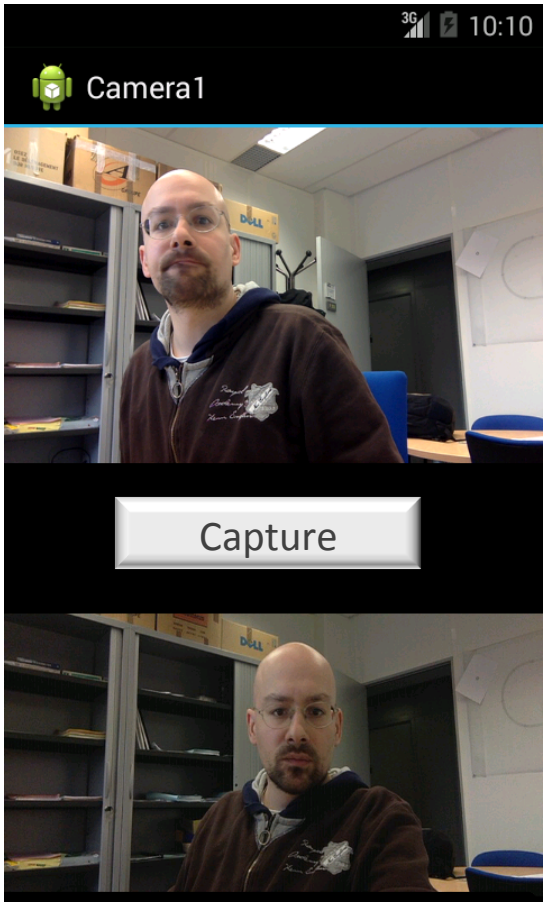
Module Organization

- 6 TP/labs on programming Multimedia applications with android
  - Video player



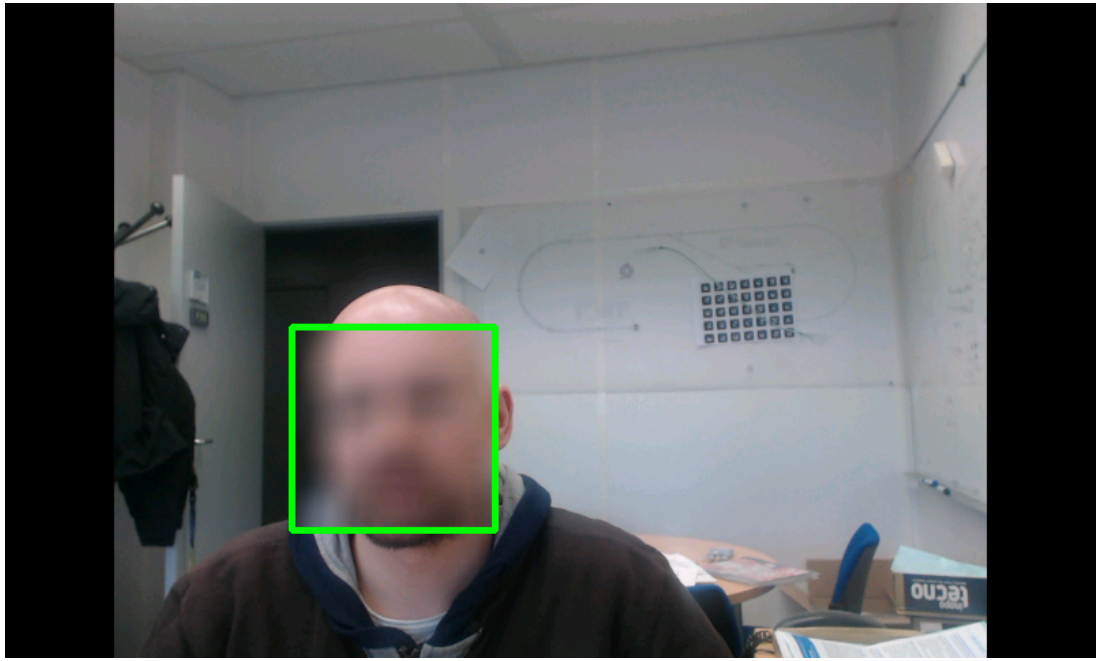
Module Organization

- 6 TP/labs on programming Multimedia applications with android
  - Video player
  - Camera

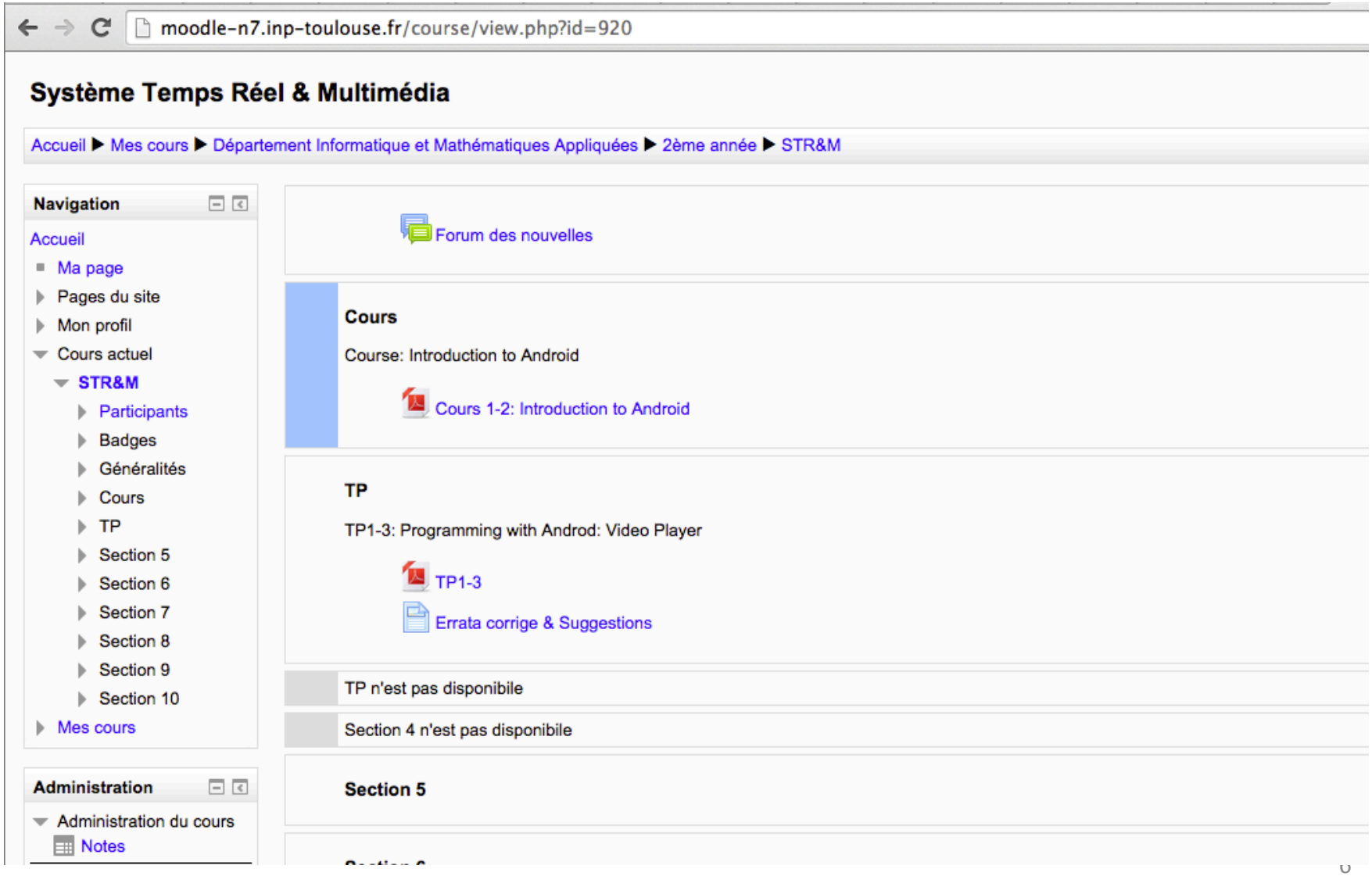


Module Organization

- 6 TP/labs on programming Multimedia applications with android
  - Video player
  - Camera
  - Defacer (with OpenCV)



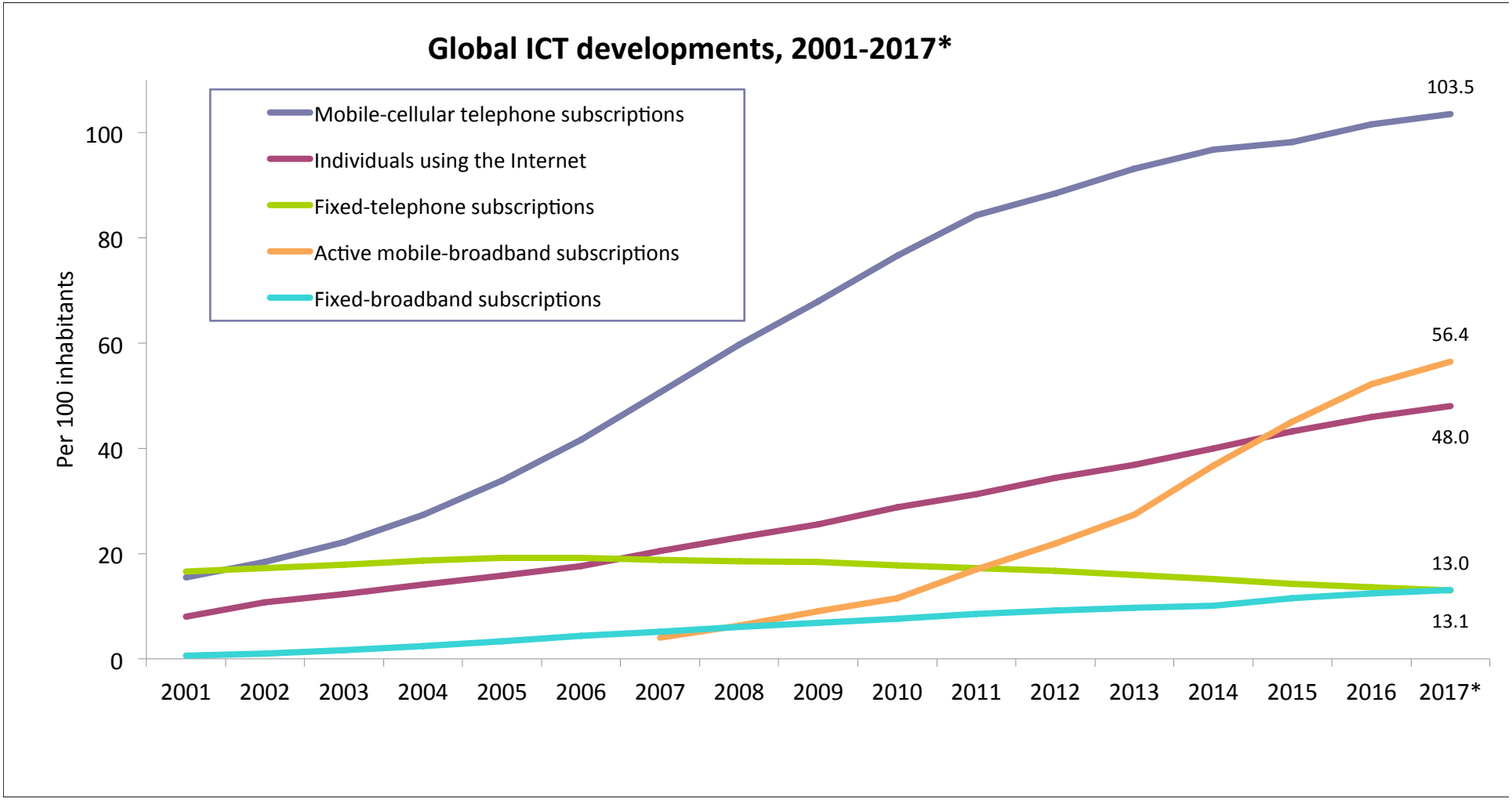
Module Organization



Plan

- **Mobile devices**
- Android overview
  - History
  - Architecture
  - Applications
- The tools
- Android Applications

Mobile device growth



Mobile devices history

- First generation (1G) – 1980s
  - **Analog** mobile telephony,
  - voice-centric,
  - largely frequency modulation (FM)-based, operating in several analog standards
  - employing in-vehicle and **eventually portable**, single-band handsets
  - weighting up to 9kg…

AT&T's Radio telephone ~1970s

Motorola DynaTAC 8000X , 0.8kg, ~4000$

Mobile devices history

- Second generation (2G) – early 1990s
  - Early digital technology,
  - providing voice and narrowband **data (SMS)**
  - small, portable, multiband handsets over GSM
  - Data over GPRS (2.5G @40kbs) and EDGE (2.75G @150kbs)

Mobile devices history

- Third generation (3G) – from early 2000s
  - Enhanced digital technology,
  - providing voice and high-speed data (UMTS @>200 kbit/s)
  - multiband handsets
  - mobile TV, streaming video, video on demand, GPS…

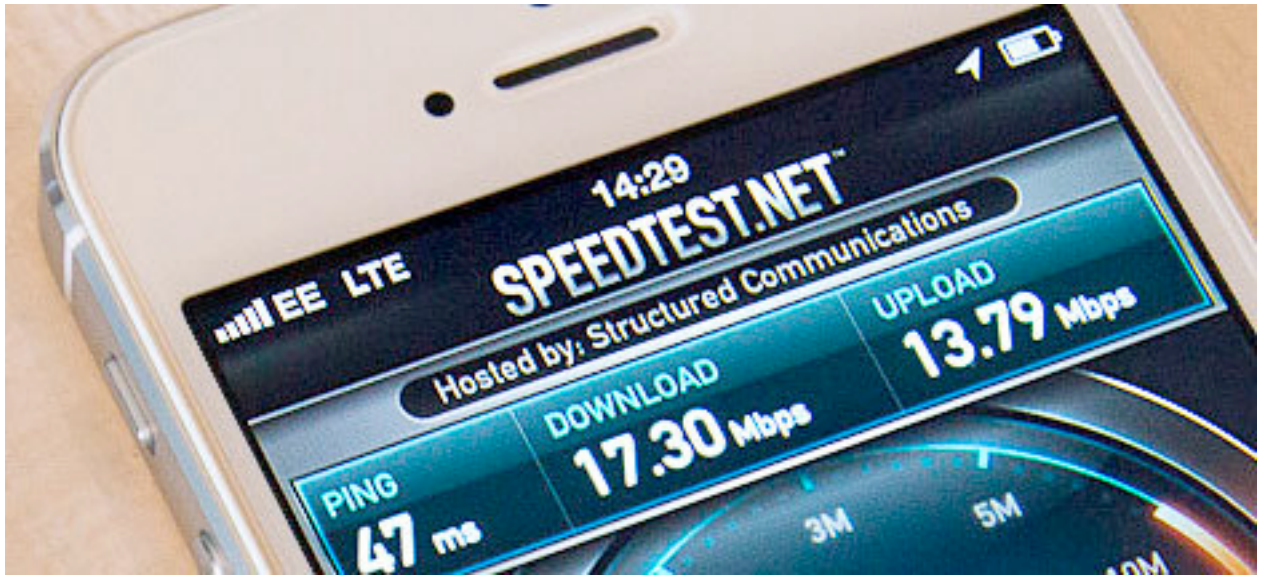
smartphone

tablet

phablet

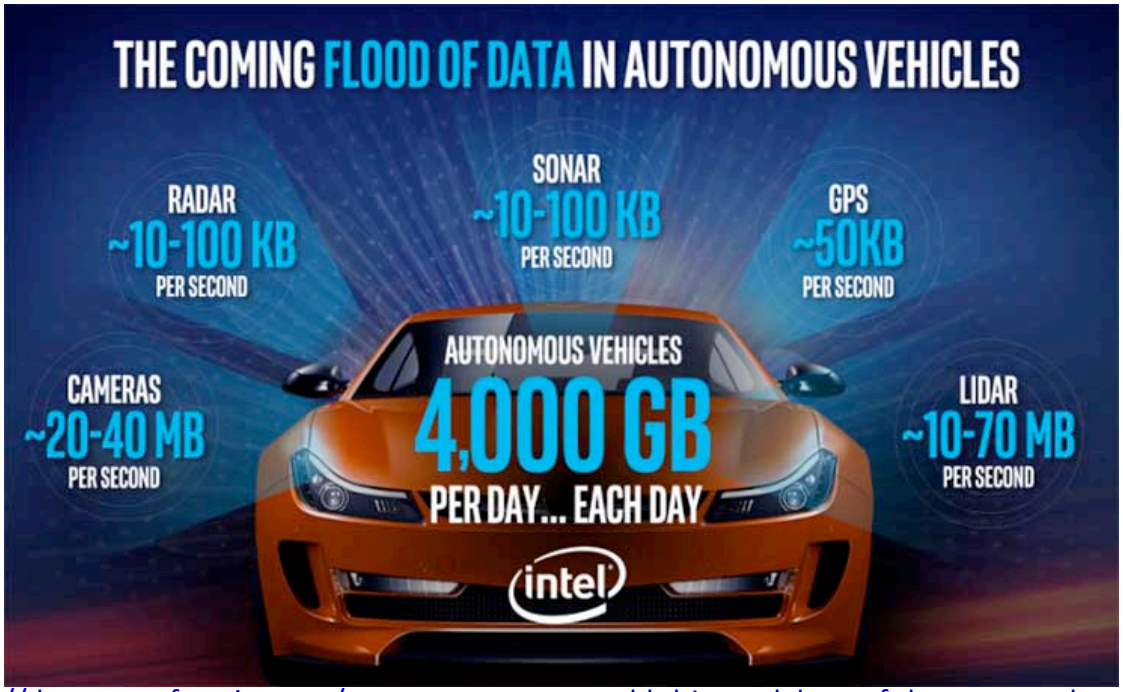
Mobile devices history

- Fourth generation (4G) – now! (it was "coming soon" when this EU started in 2014!)
  - Next-generation technology (WiMAX),
  - multimedia, broadband, highly spectrum-efficient transmission over adaptive-mode and -band handsets at comparatively much higher transmission rates.



Mobile devices history

- Fifth generation (5G) – is coming!
  - Up to 20 gigabits per second (10x faster than 4G)
  - Networks of connected vehicles that would relay relevant traffic information, and automatically maintain safe driving distances.
  - Telemedicine
  - 3D mapping etc.



https://datacenterfrontier.com/autonomous-cars-could-drive-a-deluge-of-data-center-demand/

Challenges & issues

- **Insufficient bandwidth**
  - Mobile Internet access is generally slower than direct cable connections, using technologies such as GPRS and EDGE, and more recently HSDPA and HSUPA 3G networks.
  - These networks are usually available within range of commercial cell phone towers.
  - Higher speed wireless LANs are inexpensive but have very limited range.
- **Transmission interferences**:
  - Weather, terrain, and the range from the nearest signal point can all interfere with signal reception.
  - Reception in tunnels, some buildings, and rural areas is often poor.

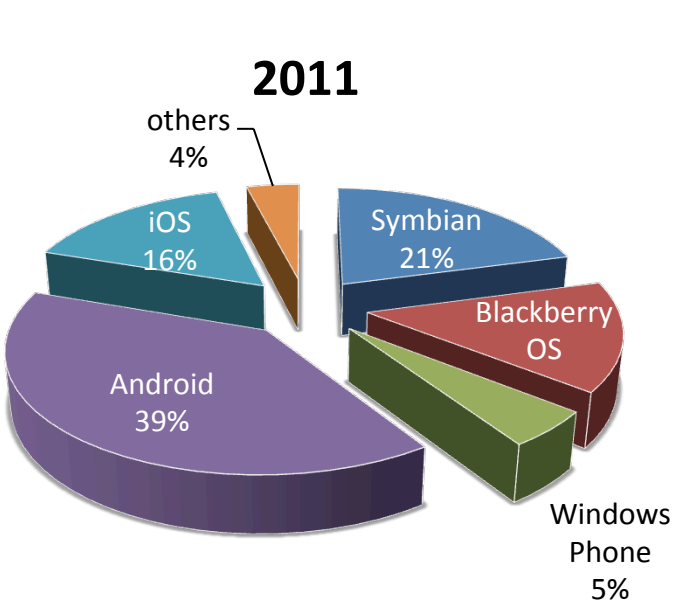
Challenges & issues

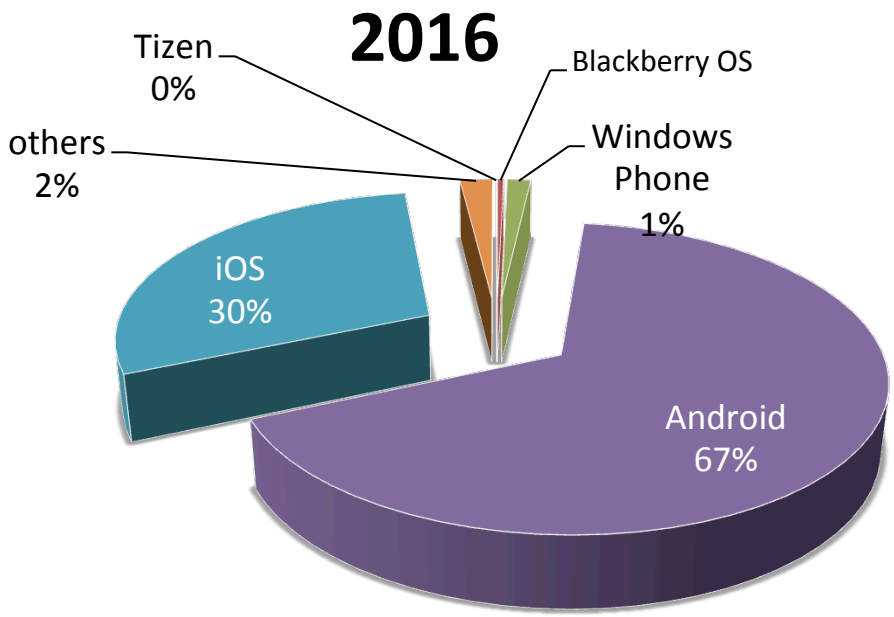
- **Power consumption**:
  - mobile devices rely entirely on battery power.
  - compact size of mobile devices requires expensive batteries and computing optimizations
- **Human interface with device**:
  - Screens and keyboards tend to be small, which makes them hard to use.
  - Alternate input methods such as speech or handwriting recognition
- **Potential health hazards (safety)**:
  - Use mobile devices while driving…
  - Interference with sensitive medical devices.
  - Phone radiation affecting health???

Challenges & issues

- **Security standards**
  - When working mobile, one is dependent on public networks, requiring careful use of VPN.
  - Security standards must be implemented on the devices to protect sensible data.
  - Privacy concerns
    - Sharing data
    - Tracking devices with GPS

Today's platforms





iOS android BlackBerry bada Firefox OS

Windows Phone palm webOS ~~symbian~~ ~~MeeGo~~ TIZEN

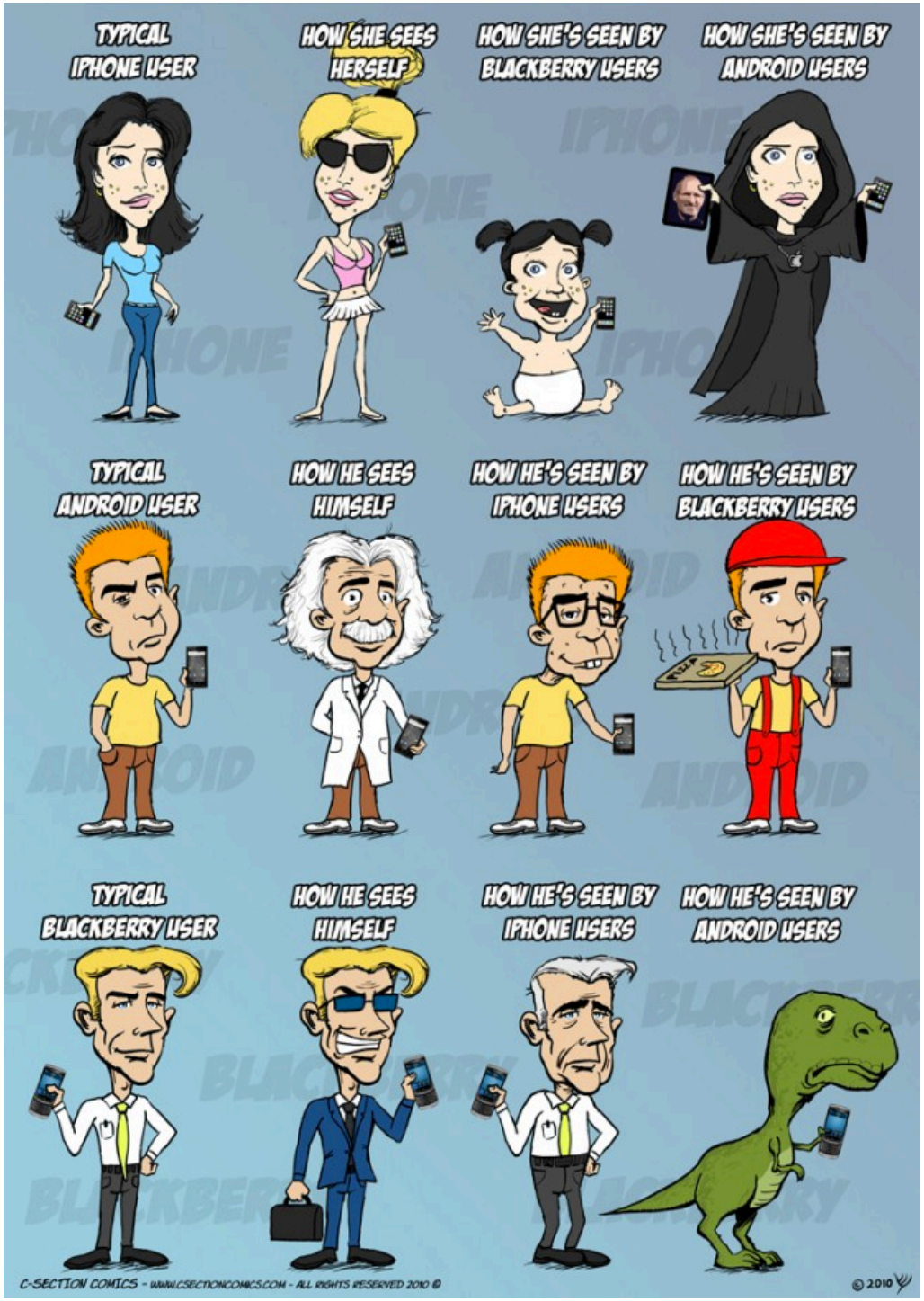
Mobile devices – Main Platforms

Platform	Applicantion Language	Undelying OS
Windows Phone	VB.Net, C#	Windows NT
iOS	Objective-C	Darwin (unix-like)
Blackberry OS	Java	Blackberry
Android	Java, native C++	Linux
Palm WebOS	JavaScript, C/C++	WebOS
Symbian OS	C++	SymbianOS
Firefox OS	HTML5/CSS3/JavaScript	Linux
Ubuntu Touch	C/C++, JavaScript	Linux
Tizen	HTML5	Linux

The trends

- HTML5 as a standard for cross-platform applications
  - The same application can run on very different devices natively (no browser)
  - Firefox OS and Tizen

- Convergence to all-devices OSs
  - The same OS can run on any device
  - Apple with iOS and OSX
  - Ubuntu touch
  - Windows (Phone) 10


TYPICAL IPHONE USER
HOW SHE SEES HERSELF
HOW SHE'S SEEN BY BLACKBERRY USERS
HOW SHE'S SEEN BY ANDROID USERS
TYPICAL ANDROID USER
HOW HE SEES HIMSELF
HOW HE'S SEEN BY IPHONE USERS
HOW HE'S SEEN BY BLACKBERRY USERS
TYPICAL BLACKBERRY USER
HOW HE SEES HIMSELF
HOW HE'S SEEN BY IPHONE USERS
HOW HE'S SEEN BY ANDROID USERS
C-SECTION COMICS - WWW.CSECTIONCOMICS.COM - ALL RIGHTS RESERVED 2010 ©
© 2010

Plan

- Mobile devices
- **Android overview**
  - History
  - Architecture
  - Applications
- Android Applications
- The tools

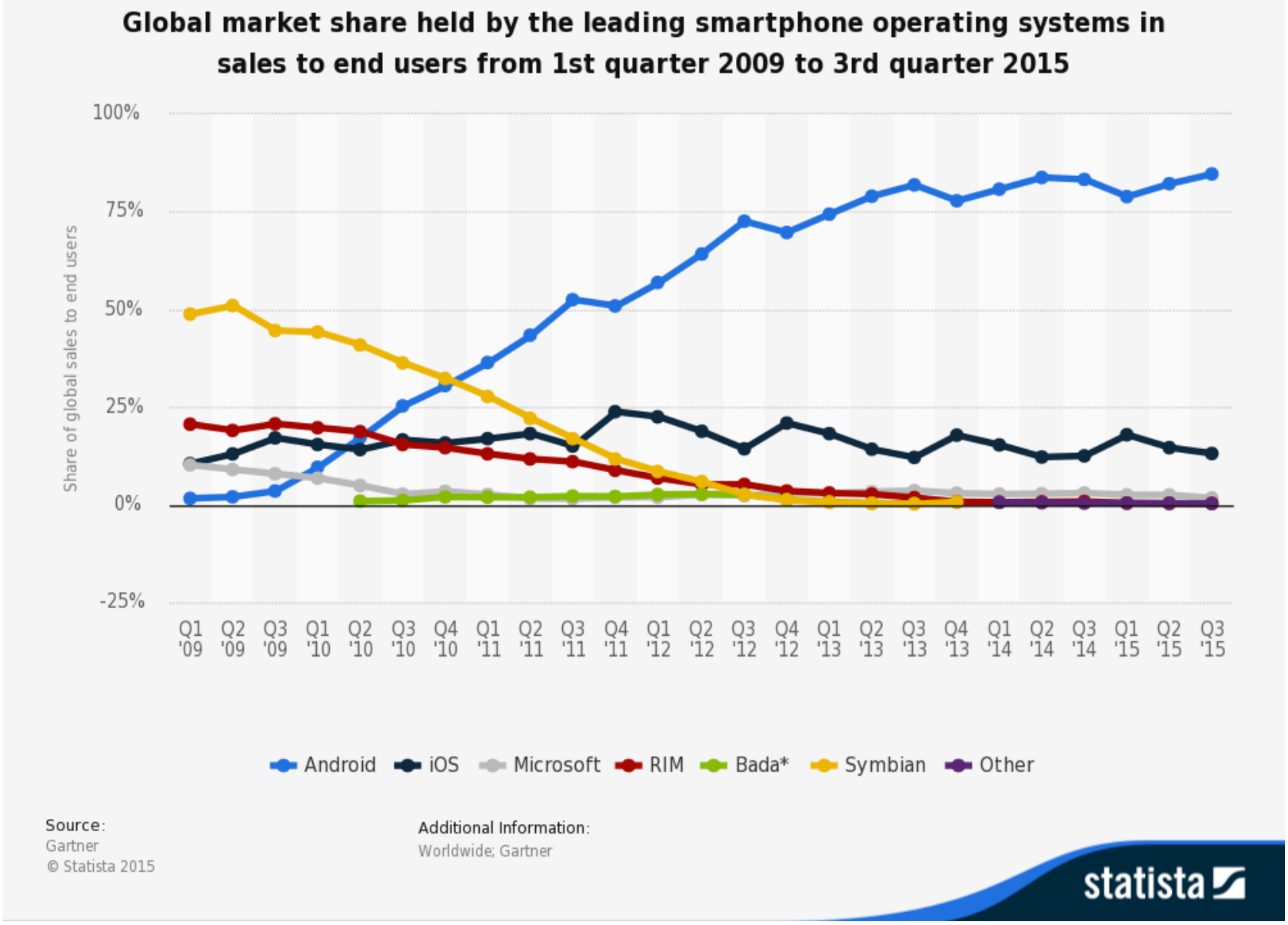
Why Android?

- Operating system thought for mobile devices
  - Collection of software to interface the hardware and applications
- Open source!
  - Licence Apache
  - Code maintained by Google publicly available
    Android Open Source Project http://source.android.com/
  - Huge community of developers
- Linux-based
- Free development platform (Vs Apple…)
  - Programming language:
    - Java
    - C/C++ using JNI
  - SDK is free

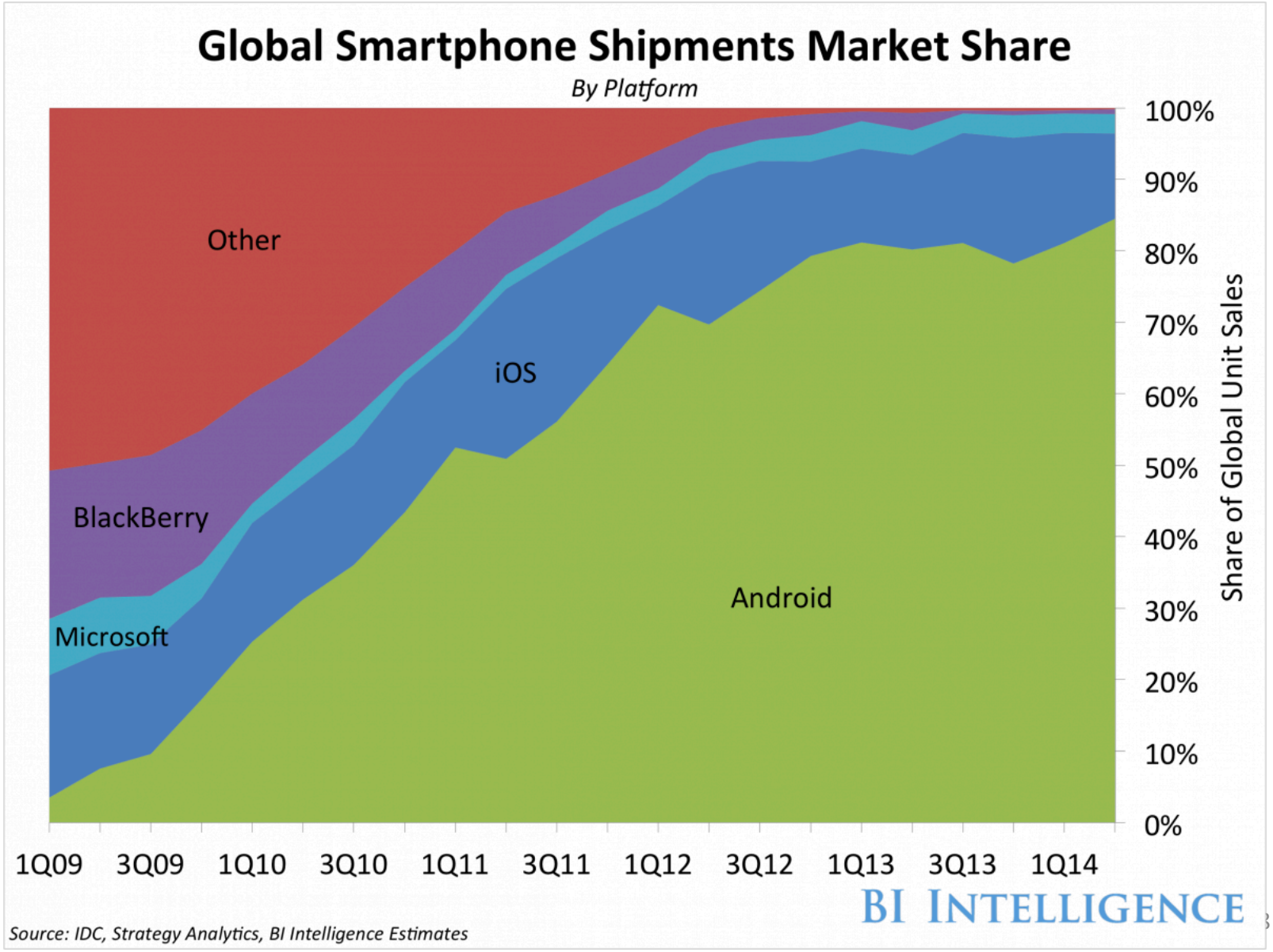
Android History

- Oct 2003 – Android Inc start-up founded to develop sw for digital cameras
  - Then they moved the focus on smartphone OS
- Aug. 2005 – Google buy Android Inc.
  - Google intention to enter mobile market?
  - Developing a flexible, upgradable system based on linux kernel
- Jan. 2007 – iPhone unveiled with touchscreen feature
  - Google was still working on keyboard based device
  - Change of plan!
- Nov. 2007 – Open Handset Alliance announced
  - HW makers, carriers and Google to develop open standards for mobile devices
- Oct. 2008 – First Android device on the market
  - HTC + T-Mobile using SDK 1.0
- 2009 – Proliferation of Android mobile devices
- Since end of 2010 Android is the top-selling smartphone platform

Android market share evolution



Android market share evolution



Open Handset Alliance (OHA)

- Founded on 6 November 2007 by 34 members
- Consortium of 84 firms led by Google
- Aim at developing open standards for mobile devices
- Promote Android platform
- Members are forbidden from producing devices based on incompatible forks of Android

open handset alliance

Google

Google's Angle…

- Google is a media company
- Main business from **ads selling**
- Push the diffusion of a common **"stable" playing field**
- Provide additional services on top of the common ground
  - Ads, Applications, Music/movie streaming…
  - … but mostly ADS!!
  - Not really interested/convenient to produce the HW too
    Nuclear stand-down: Google, Samsung, and the sale of Motorola Mobility

- Critiques:
  - Open source?
  - Strict control and constraints on adopting "undesired" fork (i.e. Amazon Kindle)
  - A "look but don't touch" kind of open
    Google's iron grip on Android: Controlling open source by any means necessary

Android main features

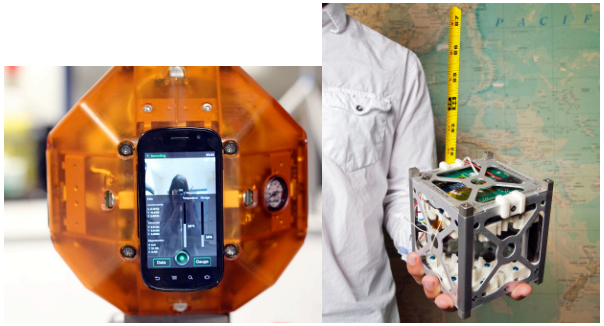
- Reusable, modular and highly configurable application framework
- Energy efficient
- DVM : Dalvik Virtual Machine
  - Java Virtual Machine optimized for embedded devices
- Integrated web browser (Chrome)
- Integrated 2D libraries
- 3D graphic support with OpenGL SE 3.0
- Support for databases with SQL-Lite
- Support for multi-touch screen

Android main features

- Multimedia support:
  - Image JPG, PNG, GIF, WebP
  - Video MPEG4, H.264, VP8
  - Audio MP3, AAC, AMR, MIDI, VORBIS
- Hardware support:
  - GSM, EDGE, 3G, 4G and Wi-Fi
  - Bluetooth, NFC
  - Sensors:
    - Camera, GPS, compass, accelerometer barometer, light, temperature, humidity
    - What Google can really do with Nest, or really, Nest's data
- Software development environment
  - ~~ADT plugin for Eclipse or~~ Android Studio
  - Graphic Emulator
  - Utilities for debugging and profiling

Not just smartphones

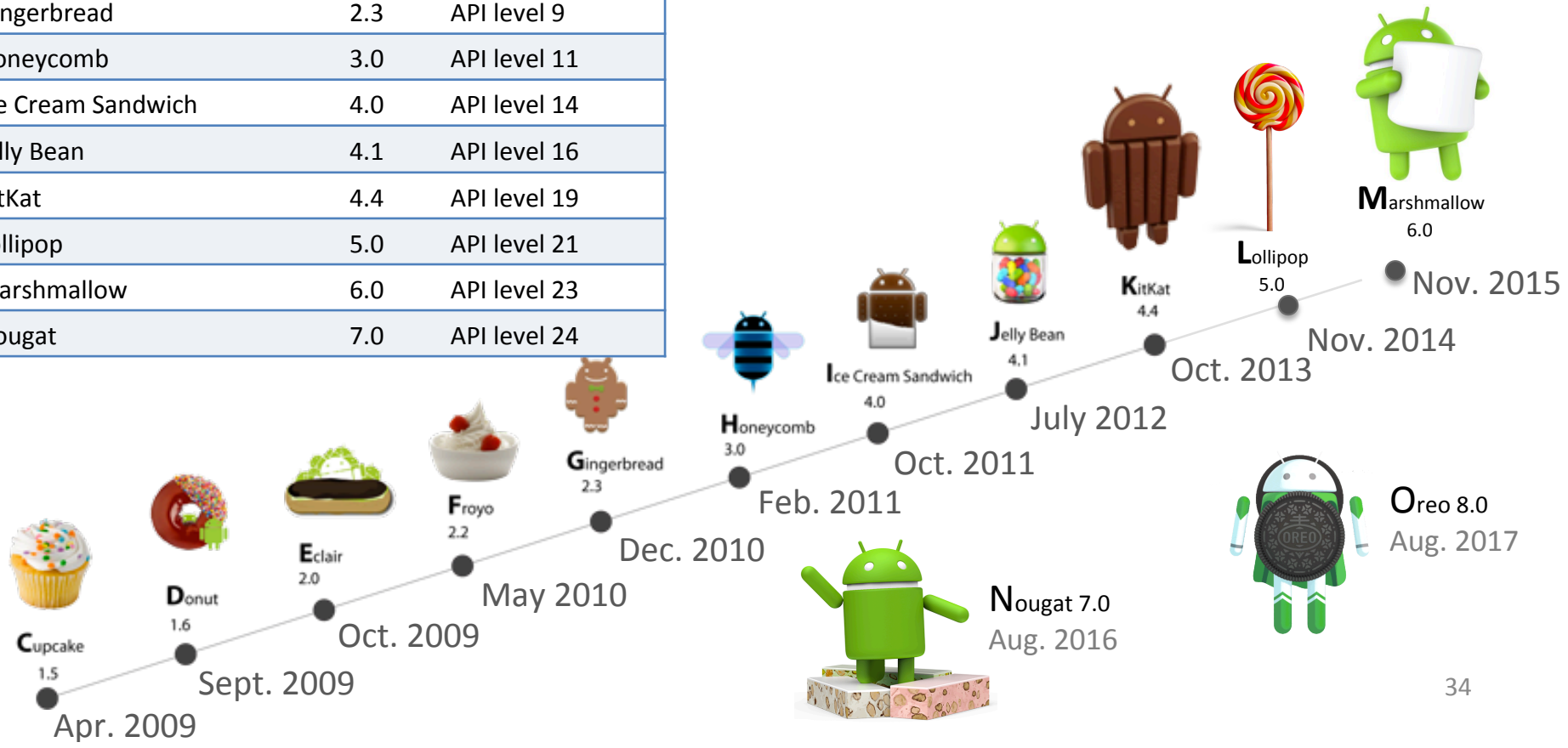
- Smartphones, Tablets, Phablets…
- Netbook / Smartbook
- Smart devices: raspberry pi 
- Cars (Google, Audi, GM, Honda, Hyundai, and Nvidia) (Lollipop)
  - Open Automotive Alliance aims to bring Android inside the car
  - Safety / security issues?
- GPS, laundry machines, TV, refrigerators (domotics, Internet of the Things…)
  - Smart TVs, smart fridges, smart washing machines? Disaster waiting to happen
- Mini-Satellites!
  - How NASA got an Android handset ready to go into space



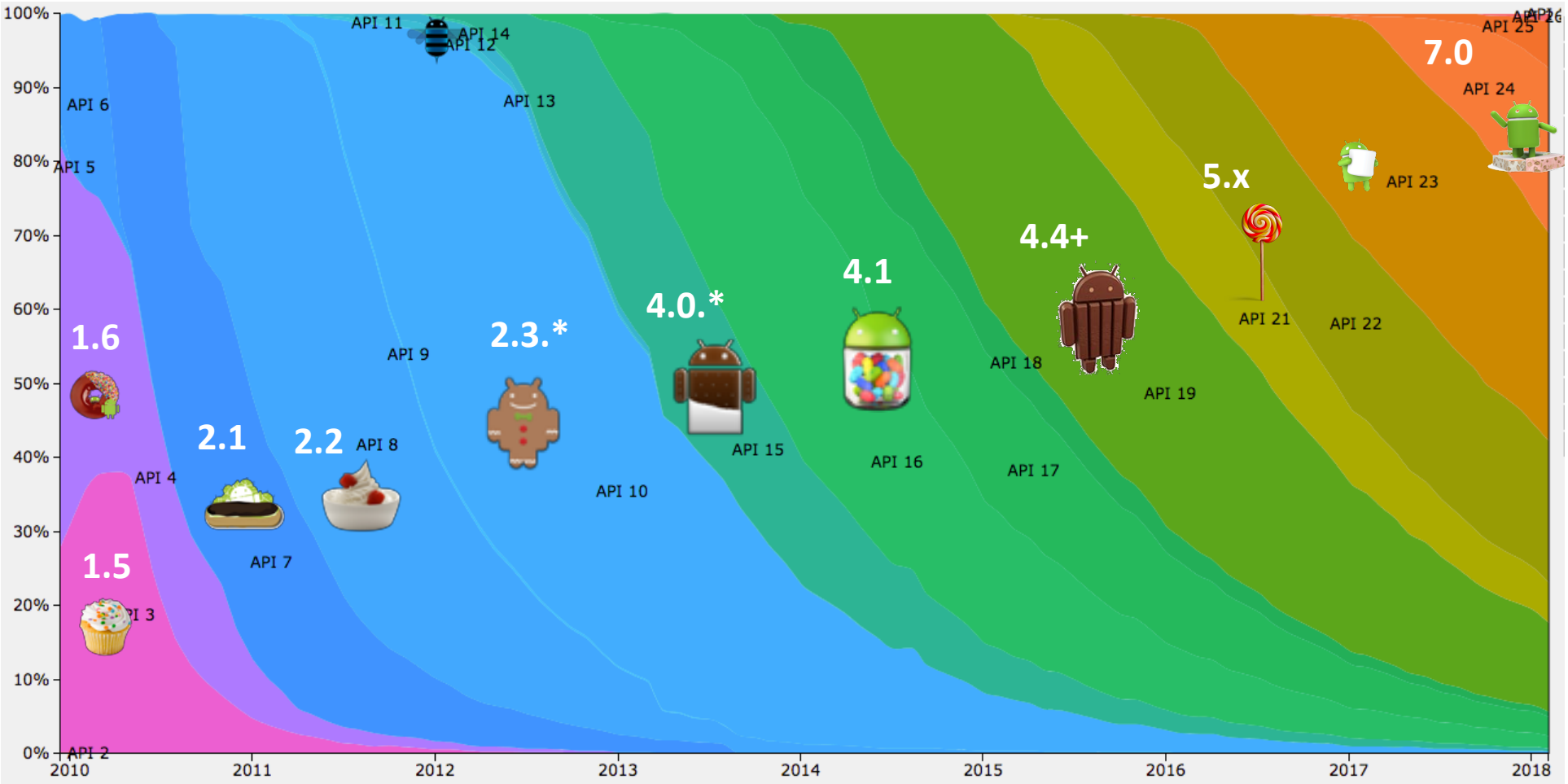
Android's versions over time

Code name	Version	API level
(no code name)	1.0	API level 1
Cupcake	1.5	API level 3
Donut	1.6	API level 4
Eclair	2.0	API level 5
Froyo	2.2	API level 8
Gingerbread	2.3	API level 9
Honeycomb	3.0	API level 11
Ice Cream Sandwich	4.0	API level 14
Jelly Bean	4.1	API level 16
KitKat	4.4	API level 19
Lollipop	5.0	API level 21
Marshmallow	6.0	API level 23
Nougat	7.0	API level 24

[Balky carriers and slow OEMs step aside: Google is defragging Android]

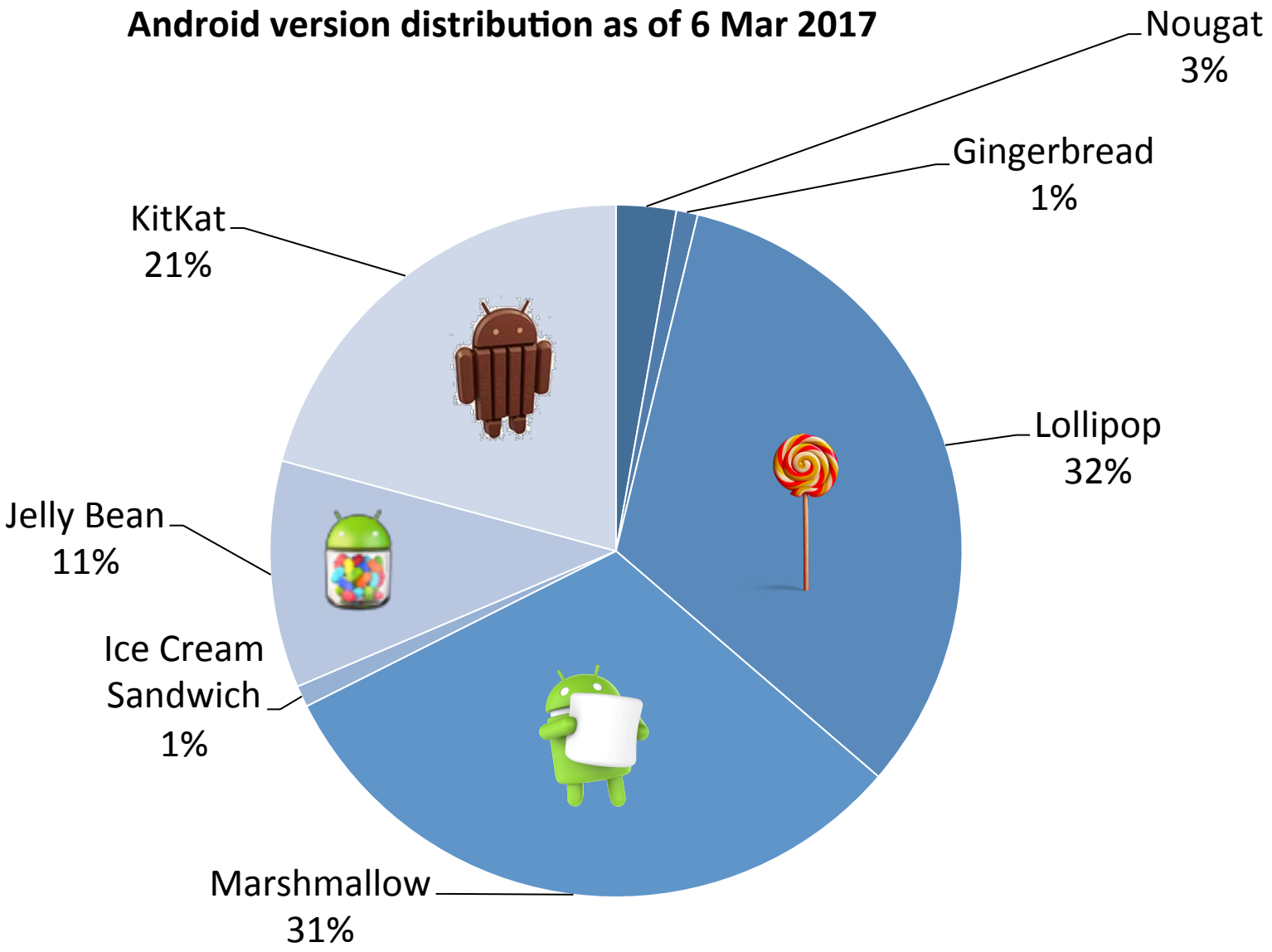


Android's versions distribution over time



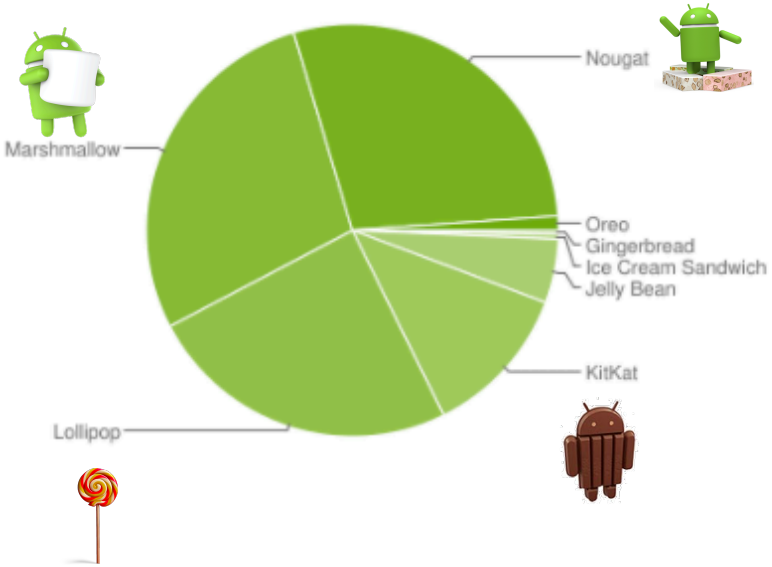
2009 – 2019

Android version distribution



Android version distribution

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%

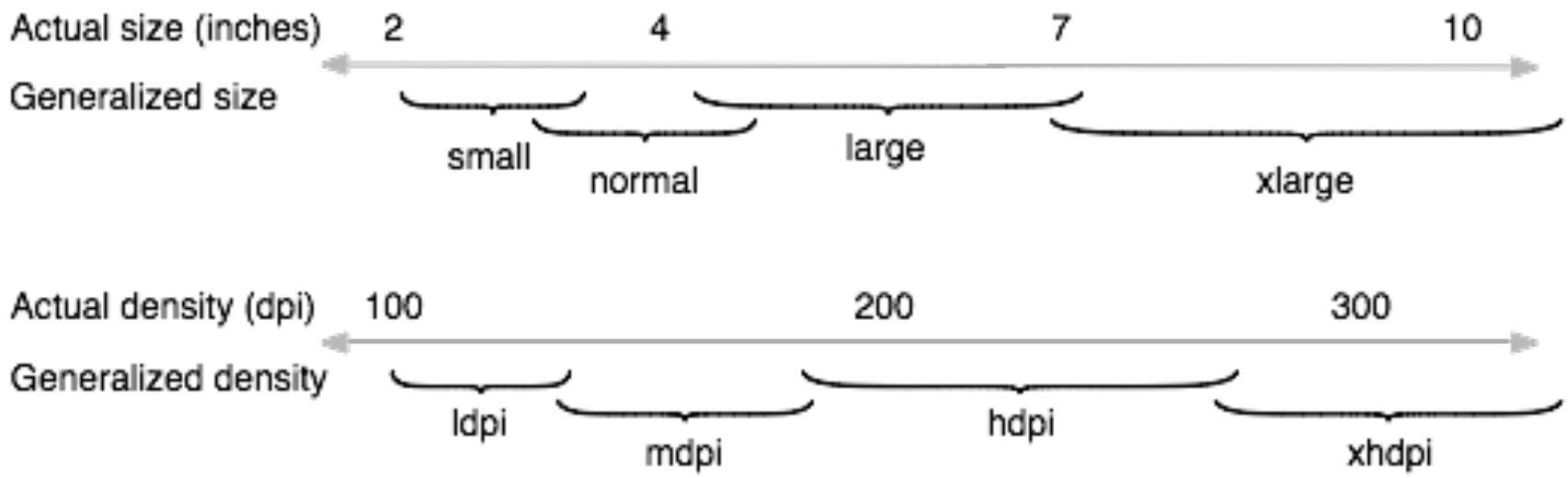


Data collected during a 7-day period ending on February 5, 2018.

Any versions with less than 0.1% distribution are not shown.

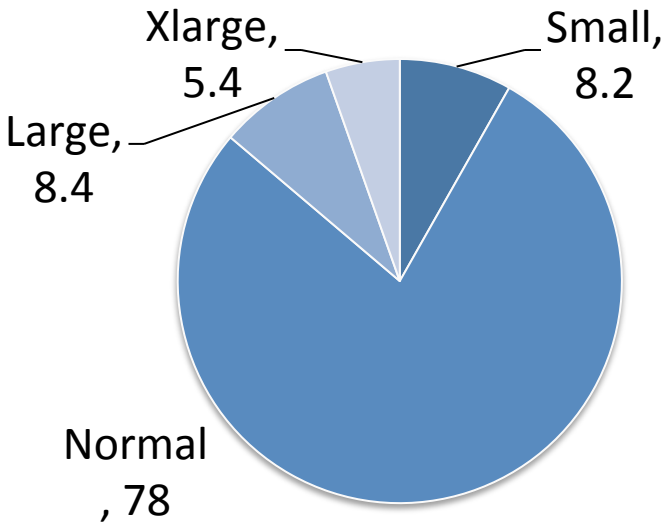
Screen size and density distribution

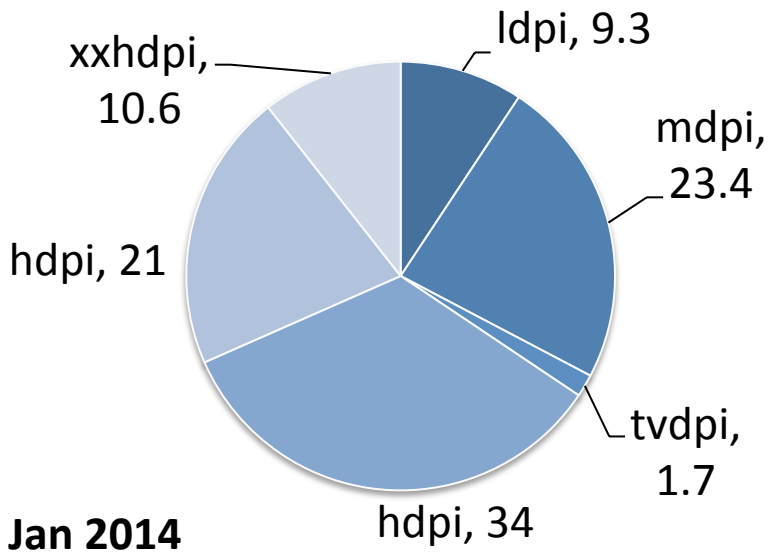
	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	8.2%						8.2%
Normal	0.1%	14.2%		33.0%	20.1%	10.6%	78.0%
Large	0.9%	4.5%	1.7%	0.6%	0.7%		8.4%
Xlarge	0.1%	4.7%		0.4%	0.2%		5.4%
Total	9.3%	23.4%	1.7%	34.0%	21.0%	10.6%	



Screen size and density distribution

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	8.2%						8.2%
Normal	0.1%	14.2%		33.0%	20.1%	10.6%	78.0%
Large	0.9%	4.5%	1.7%	0.6%	0.7%		8.4%
Xlarge	0.1%	4.7%		0.4%	0.2%		5.4%
Total	9.3%	23.4%	1.7%	34.0%	21.0%	10.6%	

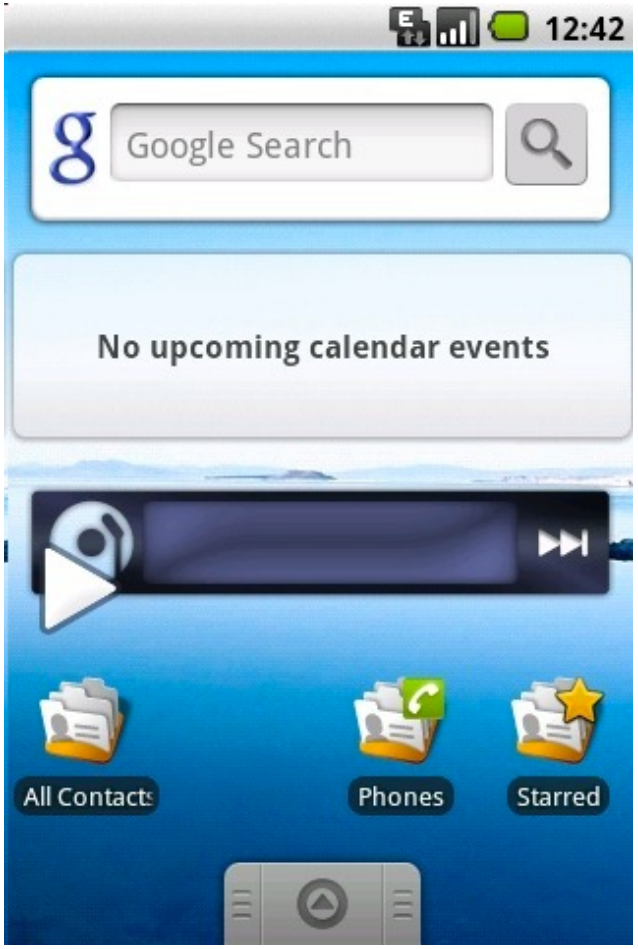




As of 8 Jan 2014

Android 1.0 (API level 1)

- 23 September 2008 only on HTC Dream

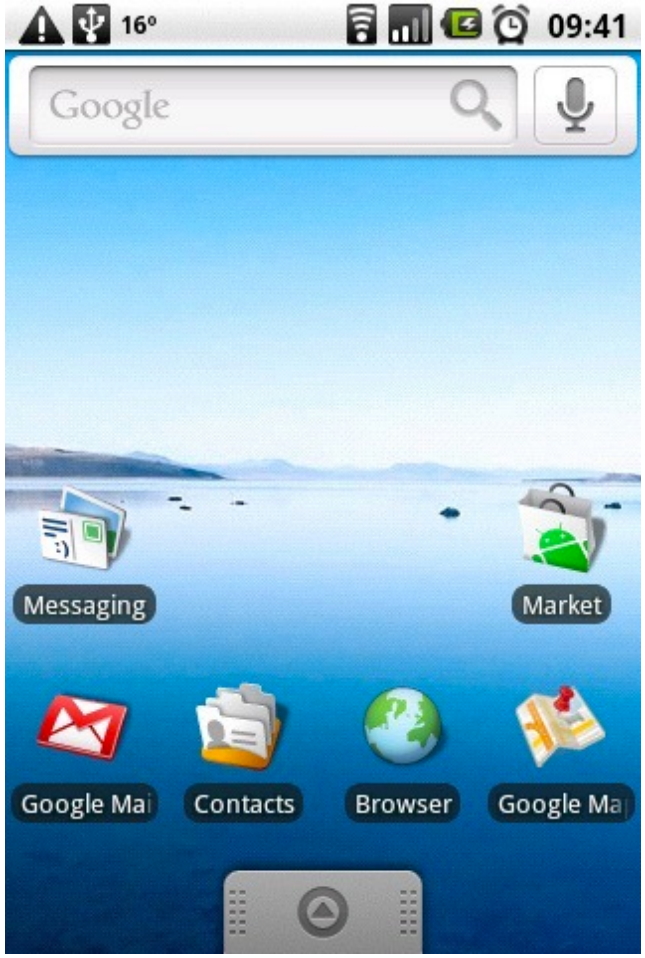


Android 1.0 (API level 1)

- 23 September 2008 only on HTC Dream
- Android Market to install and update apps
- Web browser
  - multiple pages show as windows ("cards")
- Camera support with limited features (no resolution change, quality, wb…)
- Access to web email servers, supporting POP3, IMAP4, and SMTP
- Gmail, Contacts and Calendar synchronization
- Google Maps with Street View to view maps and satellite imagery
- Google Search to search the Internet and the phone
- Google Talk instant messaging, SMS and MMS
- Media Player, enabling management, importing, and playback of media files
- Voice Dialer to dial without typing
- YouTube player
- Wi-Fi and limited Bluetooth support

Cupcake 1.5 (API level 3)

- 30 April 2009
- Linux Kernel 2.6.27

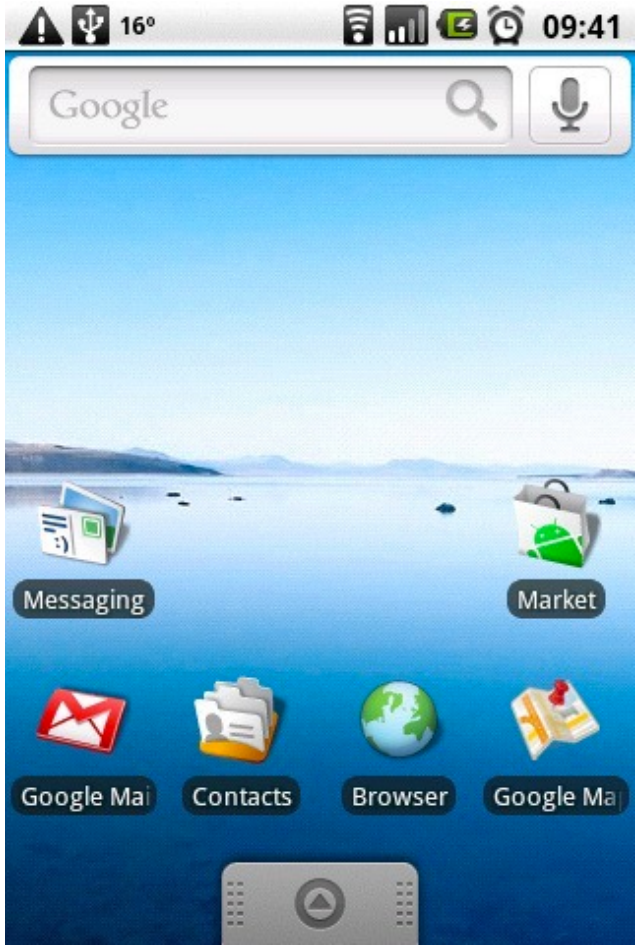


Cupcake 1.5 (API level 3)

- 30 April 2009
- Linux Kernel 2.6.27
- First release codenamed after desserts
- Video recording and playing in **MPEG-4 and 3GP**
- Video and pictures upload to YouTube and Picasa
- **Virtual keyboards** with text prediction and user dictionary for custom words
- Auto-pairing and stereo support for Bluetooth
- Support for Widgets
  - miniature application views that can be embedded in other applications
- Animated screen transitions
- Auto-rotation option

Donut 1.5 (API level 4)

- 15 Septembre 2009
- Linux Kernel 2.6.29

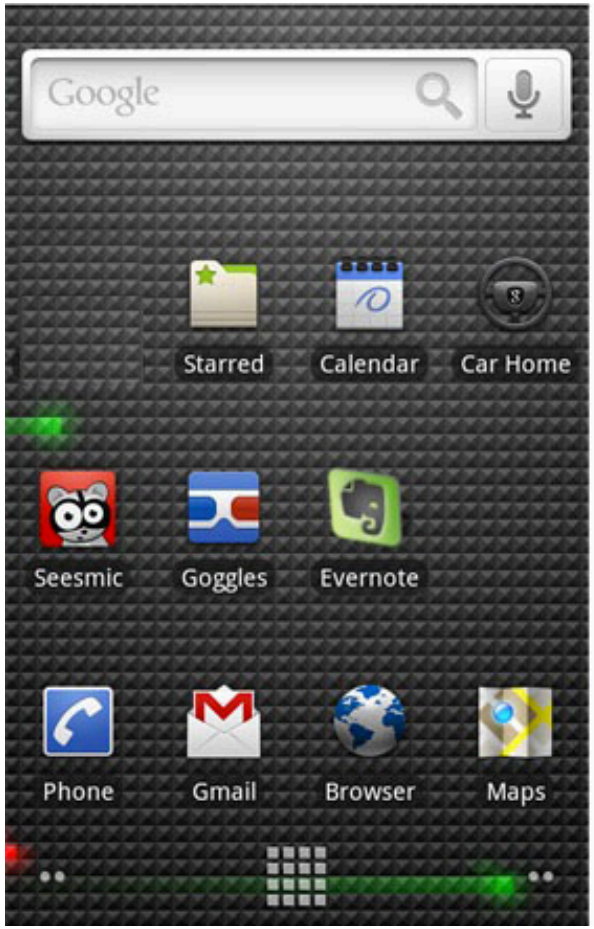


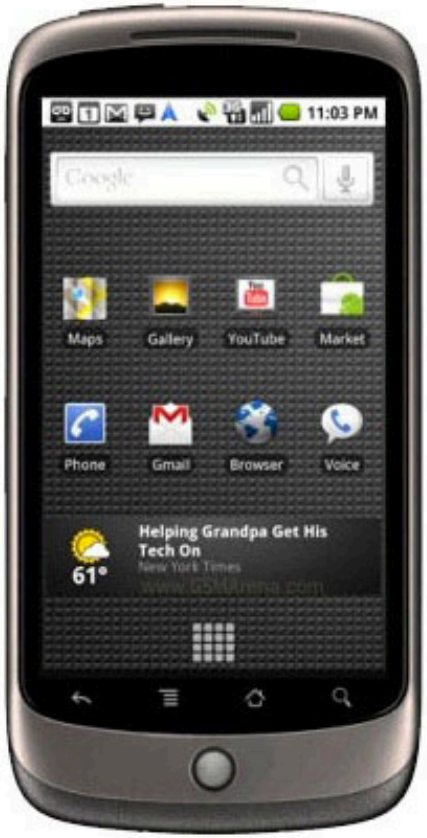
Donut 1.5 (API level 4)

- 15 September 2009
- Linux Kernel 2.6.29
- Voice and text entry search enhanced to include bookmark history, contacts, and the web
- Multi-lingual speech synthesis engine to "speak" a string of text
- Gallery, camera more fully integrated, with faster camera access
  - Select multiple photos for deletion
- Updated technology support for CDMA/EVDO, 802.1x, VPNs, and a text-to-speech engine
- Support for WVGA screen resolutions
- Speed improvements in searching and camera applications
- Expanded Gesture framework and new GestureBuilder development tool

Eclair 2.1 (API level 5)

- 26 October 2009
- Linux Kernel 2.6.29





Eclair 2.1 (API level 5)

- 26 October 2009
- Linux Kernel 2.6.29
- Expanded Account sync, multiple accounts synchronization per device
- New camera features
  - flash support, digital zoom, scene mode, white balance, color effect and macro focus
- Improved virtual keyboard
  - smarter dictionary that learns from word usage and includes contact names as suggestions
- Refreshed browser UI with **support for HTML5**
- Optimized hardware speed and revamped UI
  - Support for more screen sizes and resolutions, with better contrast ratio
- MotionEvent class enhanced to **track multi-touch events**

Froyo 2.2 (API level 8)

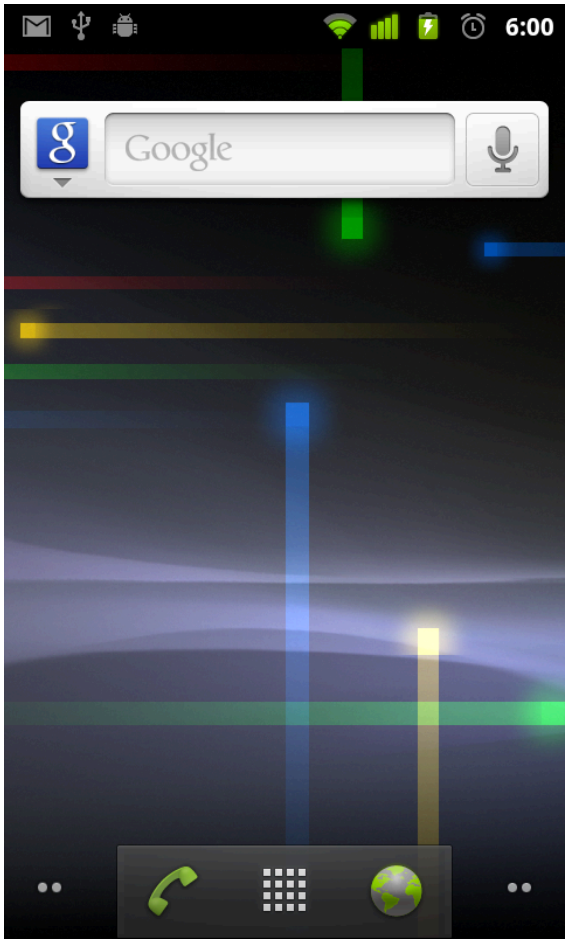
- 20 May 2010
- Linux Kernel 2.6.32

Froyo 2.2 (API level 8)

- 20 May 2010
- Linux Kernel 2.6.32
- Speed, memory, and performance optimizations through JIT compilation
- Integration of Chrome's V8 JavaScript engine into the Browser app
- USB tethering and Wi-Fi hotspot functionality
- Option to disable data access over mobile network
- Updated Market application with batch and automatic update features
- Quick switching between multiple keyboard languages and their dictionaries
- Adobe **Flash support**
- Support for high-PPI displays (up to 320 ppi), such as 4" 720p screens

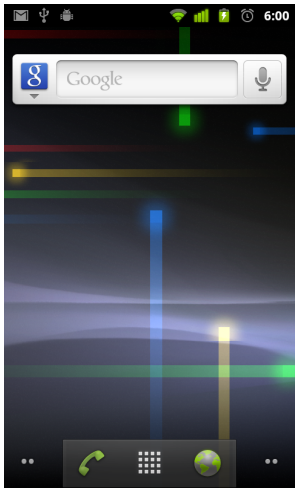
Gingerbread 2.3 (API level 9)

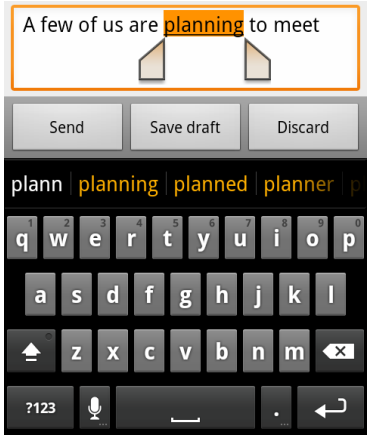
- 6 December 2010
- Linux Kernel 2.6.35



Gingerbread 2.3 (API level 9)

- 6 December 2010
- Linux Kernel 2.6.35
- Updated UI design with increased speed
- Support for XL screen sizes and resolutions
- Native support for VoIP
- Enhanced copy/paste functionality
- Support for Near Field Communication (**NFC**)
- Support for **multiple cameras** on the device
  - WebM/VP8 video playback, and AAC audio encoding
  - New audio effects (reverb, equalization…)
- Improved power management
- Native support for **more sensors** (such as gyroscopes and barometers)

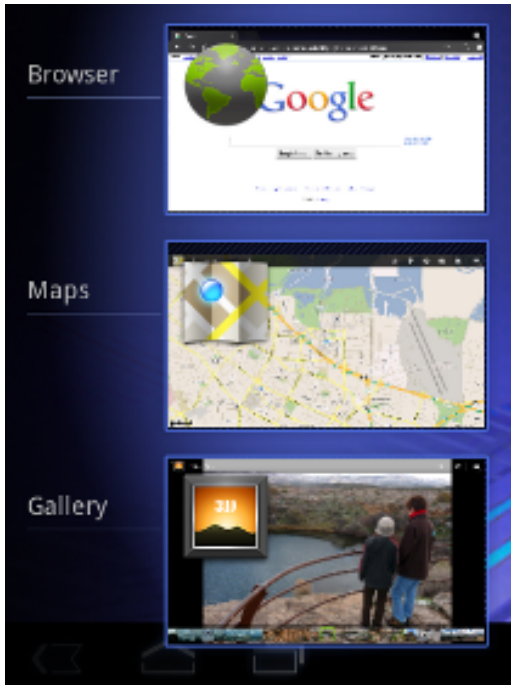




HoneyComb 3.0 (API level 11)

- 22 February 2011
- Linux Kernel 2.6.36

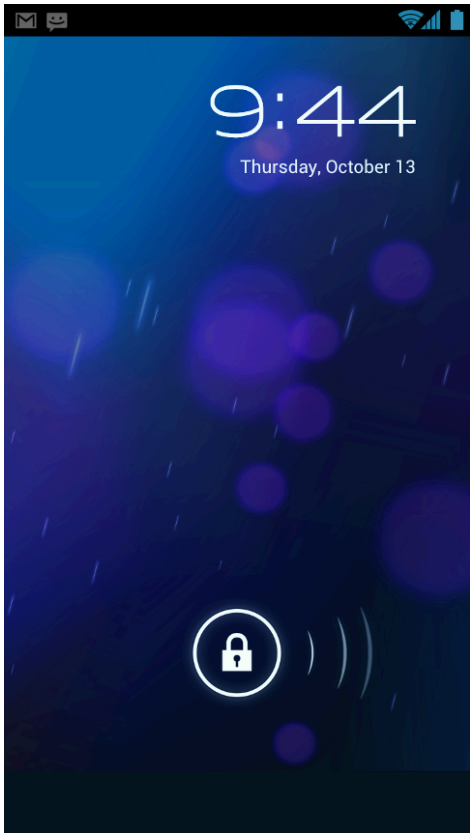
HoneyComb 3.0 (API level 11)

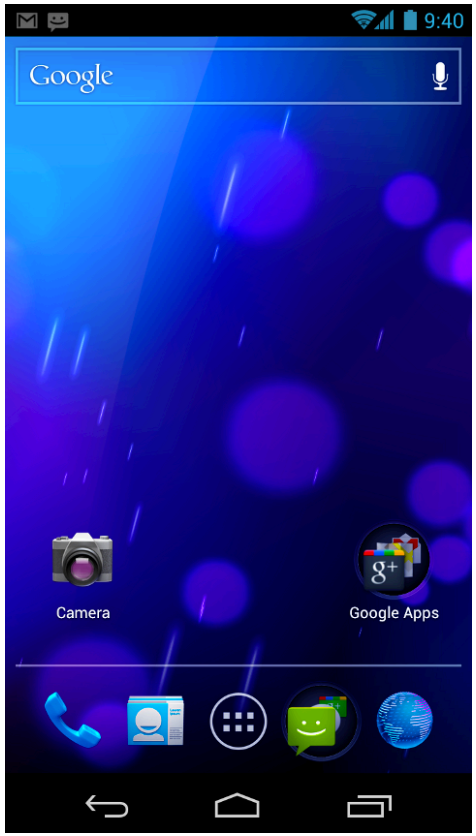


- 22 February 2011
- Linux Kernel 2.6.36
- First **tablet-only update**
- Hardware acceleration
- Support for **multi-core processors**
- Simplified multitasking
- Improved camera interface
- Support for video chat using Google Talk
- Ability to encrypt all user data
- Multi-tab browser replacing browser windows, plus “incognito” mode
- Optimized tablet support with a new “holographic” user interface
  - System Bar (notifications, status, and soft navigation buttons)
  - Action Bar (contextual options, navigation, widgets)

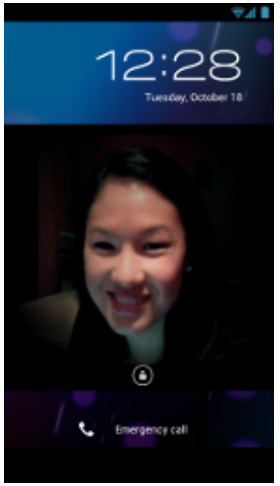
IceCream Sandwich 4.0 (API level 14)

- 19 October 2011
- Linux Kernel 3.0.1





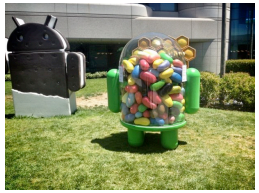
IceCream Sandwich 4.0 (API level 14)



- 19 October 2011
- Linux Kernel 3.0.1
- Soft buttons
- Integrated screenshot capture
- **Face Unlock**
- Automatic syncing of Chrome bookmarks
- **Android Beam** (NFC app to exchange in short range)
- Hardware acceleration of the UI
- **Wi-Fi Direct**
- 1080p video recording for stock Android devices

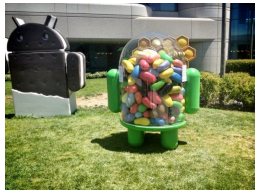
- Improved camera application with zero shutter lag, time lapse settings, panorama mode, and the ability to zoom while recording

Jelly Bean 4.3 (API level 16)



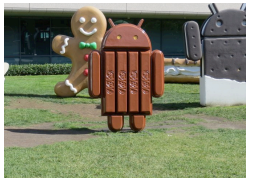
- 9 July 2012
- Linux Kernel 3.0.31

Jelly Bean 4.3 (API level 16)



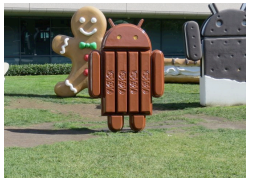
- 9 July 2012
- Linux Kernel 3.0.31
- Smoother user interface:
  - Triple buffering in the graphics pipeline
  - OpenGL ES 3.0 for High-Performance Graphics
- Offline voice dictation
- **Google Play** replaces Android market
- Improved camera application
- Ability to turn off notifications on an application-specific basis
- Google Wallet (for the Nexus 7)
- **Google Now** voice assistant and search application
- Preinstalled mobile version of **Google Chrome**

KitKat 4.4 (API level 19)



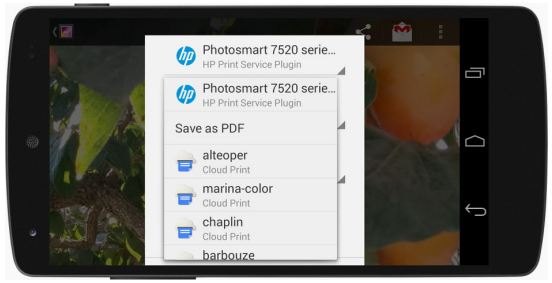
- 31 October 2013

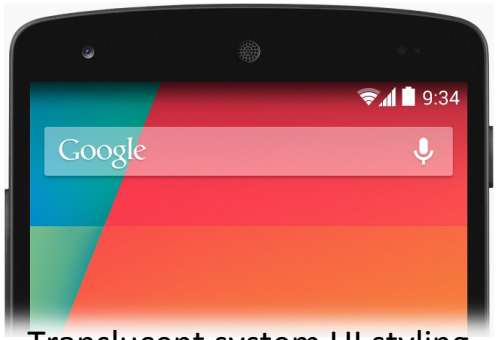
KitKat 4.4 (API level 19)



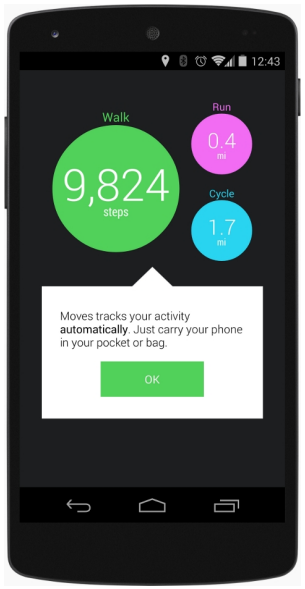
- 31 October 2013
- Applications can now use "**immersive mode**"
- Optimizations for performance on devices with lower specifications, including zRAM support and "low RAM" device API
- Wireless printing capability
- Sensor batching, step detector and counter APIs

WiFi Printing

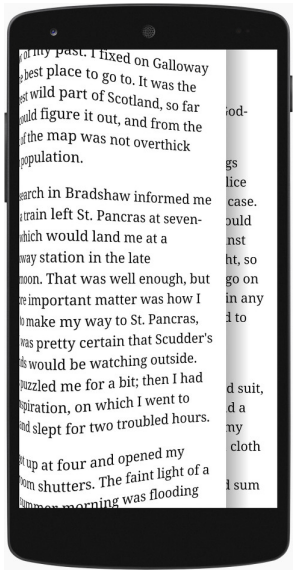




Translucent system UI styling



Step detector



Immersive mode

Lollipop 5.0 (API level 22)

- November 2014

Lollipop 5.0 (API level 22)

- Further extension to phones, tablets, and **wearables**, to TVs and **cars**.
- **Material design** for the GUI
- Improved performances with:
  - new **ART runtime**
  - support for **64-bit architectures**
  - New **job scheduling** API
  - Support for Khronos OpenGL ES 3.1
- New types of sensors:
  - **tilt detector** sensor helps improve activity recognition on supported devices,
  - **heart rate sensor** reports the heart rate of the person touching the device.

Marshmallow 6.0 (API level 23)

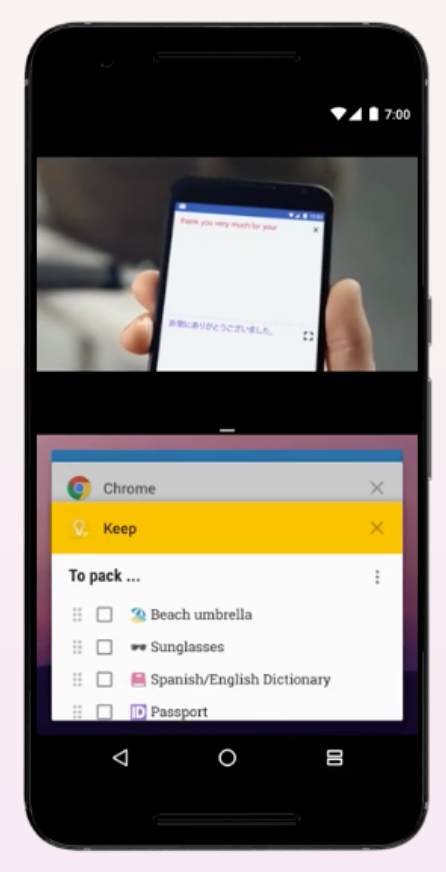
- October 2014

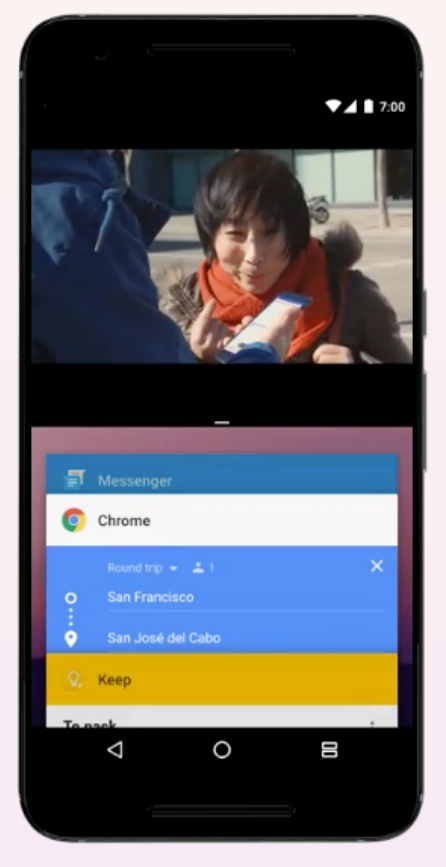
Marshmallow 6.0 (API level 23)

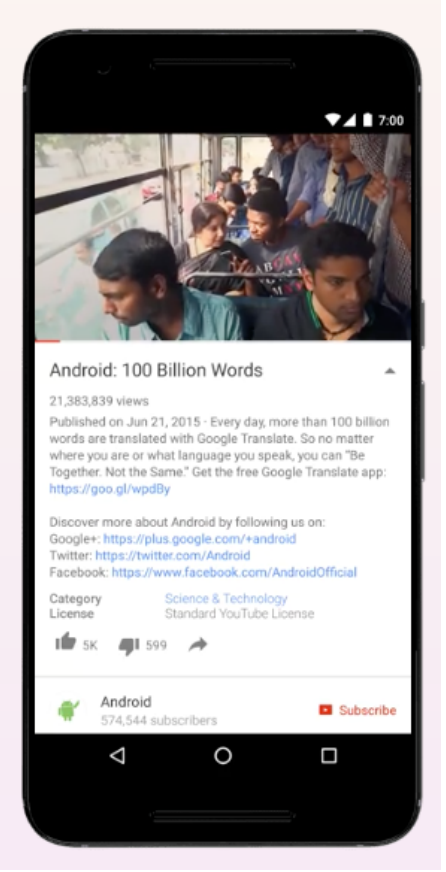
- "Google Now on Tap", searches within the context of information currently being.
- Redesigned application permission model:
  - 8 permission categories,
  - Permissions granted when needed for the first time and revokable by the user at any time
- Improved power management:
  - Device enters a low-power state if it is inactive and not being physically handled
- Fingerprint support
- USB Type-C support

Nougat 7.0 (API level 24-25)

- August 2016





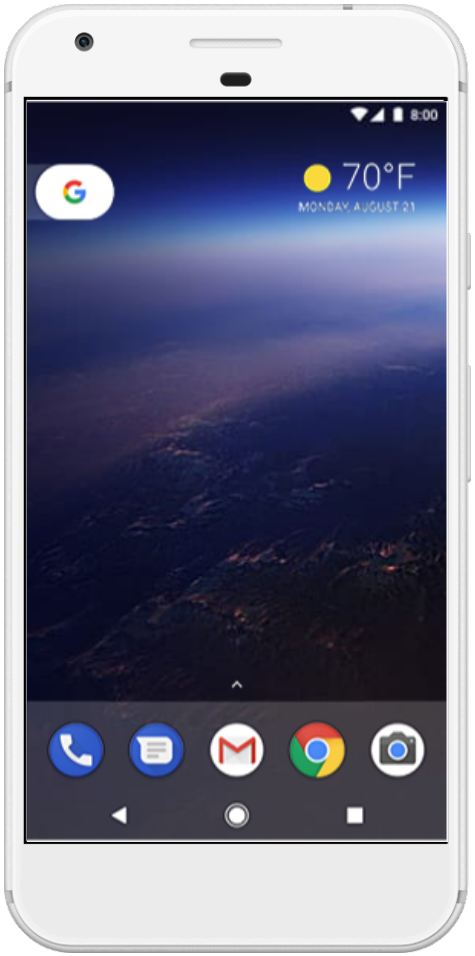


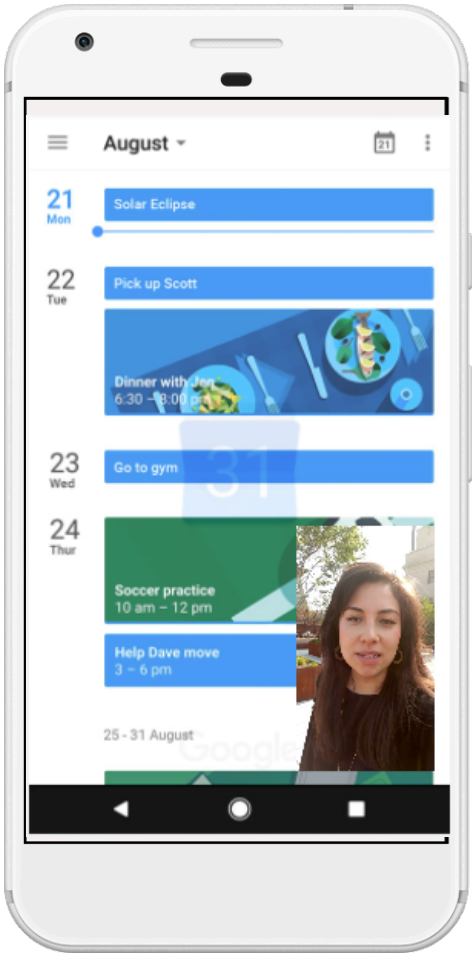
Nougat 7.0 (API level 24-25)

- Multiple apps on-screen at once in a split-screen view
- Improved power management
- support for Vulkan, the new low-level 3D-rendering API to augment OpenGL ES but with higher graphics performance
- support for the Google Daydream virtual reality platform.
  - specific "VR mode", with advanced technology for reduced graphics latency
- Security improvements

Oreo 8.0 (API level 26)

- August 2017





Oreo 8.0 (API level 26)

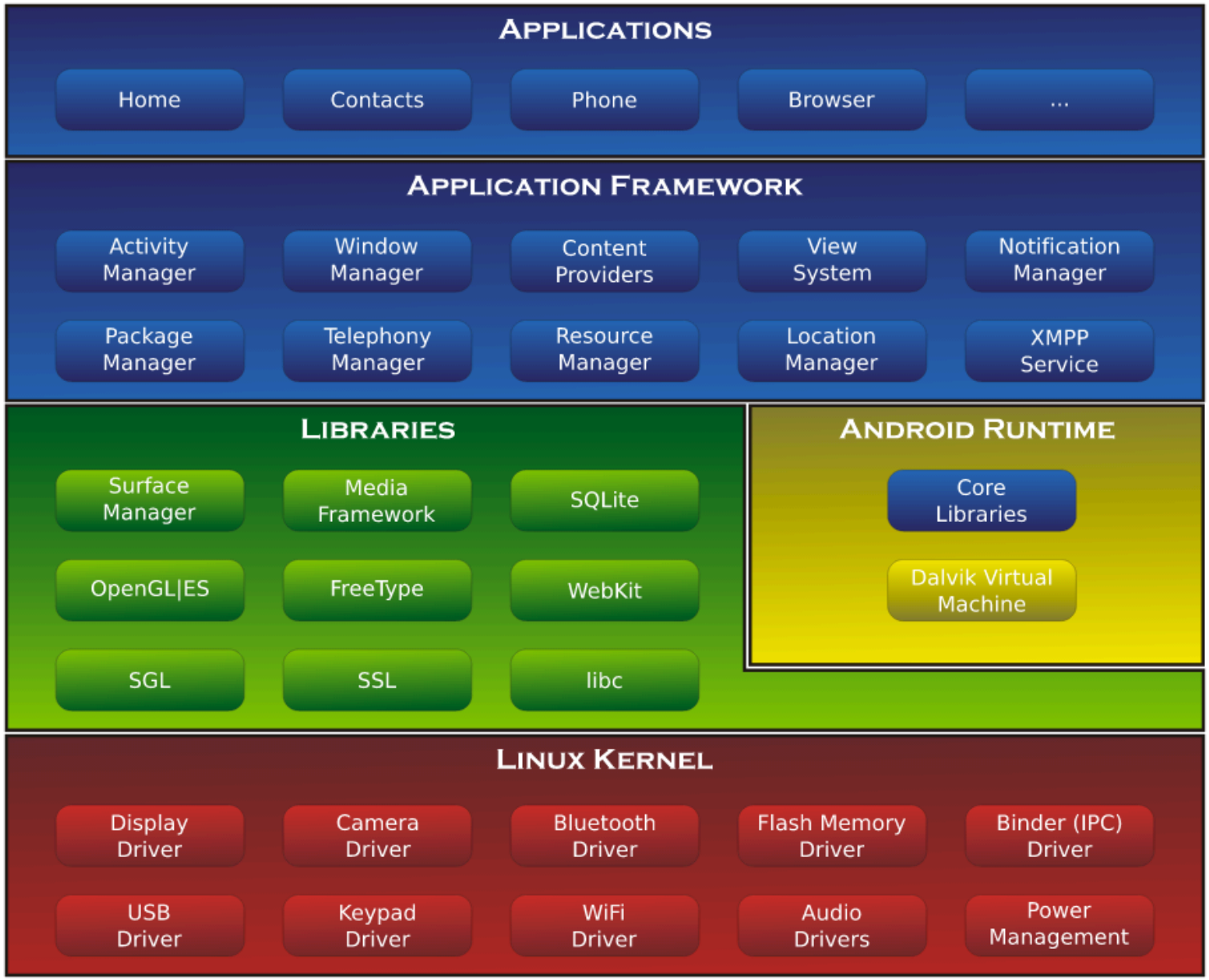
Major features:

- picture-in-picture support for video,
- performance improvements and battery usage optimization
- support for autofillers when writing text
- Bluetooth 5 support
- Wi-Fi Aware: when in the range of a particular access point or another compatible the device can receive notifications of applications or services available in the proximity
- Android Go – a software distribution of the operating system for low-end devices with optimizations designed to reduce mobile data usage

Plan

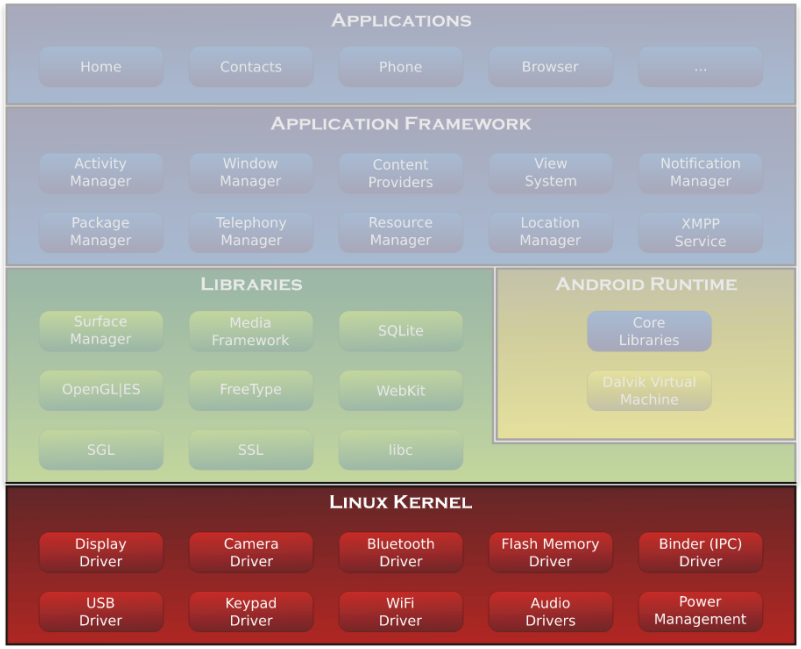
- Mobile devices
- Android overview
  - History
  - **Architecture**
  - Applications
- The tools
- Android Applications

Android Architecture



The Linux kernel

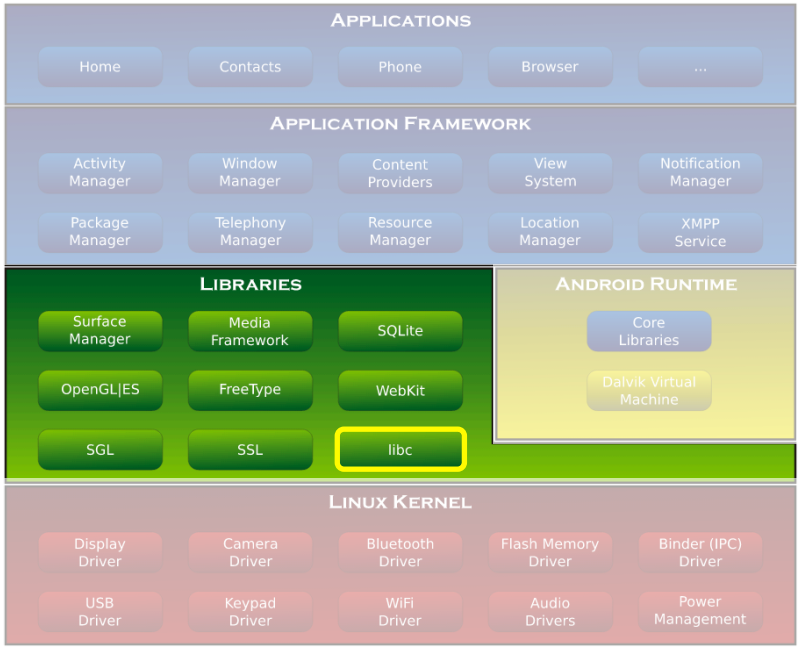
- Based on Linux kernel 2.6
  - No X-window system
  - Custom/simplified **Bionic libc**
- Why Linux?
  - Open source
  - Memory and process management highly stable and performing
  - Permission-based security system since ’70
- **Patches** to Linux kernel
  - Power Management
    - Eg alarms to wake up when suspended
  - Shared memory among processes
  - Memory/process management
  - Binder
    - Inter-process communication for sharing data



Kernel features unique to Android

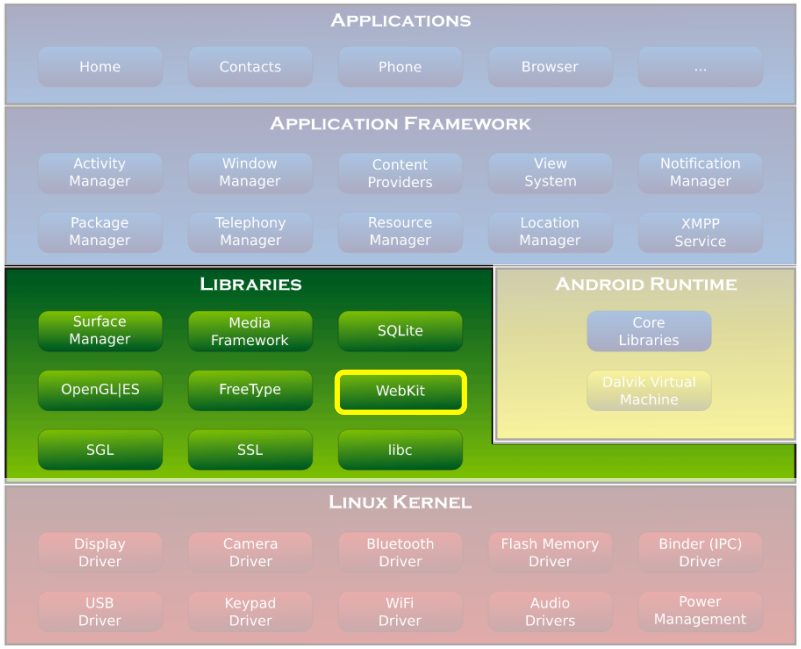
The libraries

- C/C++ libraries for low-level features
- **Bionic libc**
  - Open Source (BSD)
  - Optimized for mobile devices
    - Light (~200Kb)
    - Fast
  - Support for ARM and x86 CPUs
  - Limited support for STL
  - Brand new thread management



The libraries

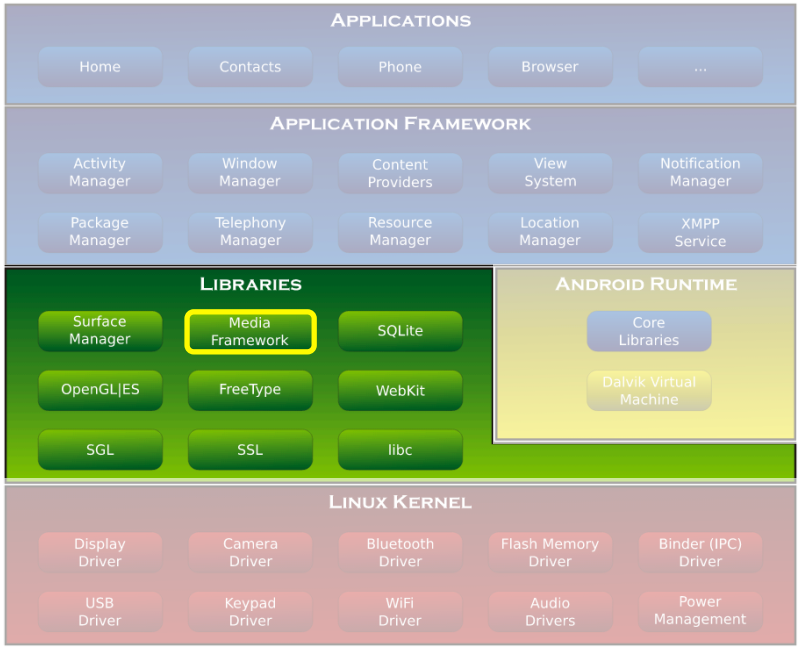
- C/C++ libraries for low-level features
- **Bionic libc**
  - Open Source (BSD)
  - Optimized for mobile devices
    - Light (~200Kb)
    - Fast
  - Support for ARM and x86 CPUs
  - Limited support for STL
  - Brand new thread management
- **Webkit (→ Blink)**
  - CSS, Javascript, DOM, AJAX
  - Fully compatible with Internet standards
  - Minor modification to adapt to mobile

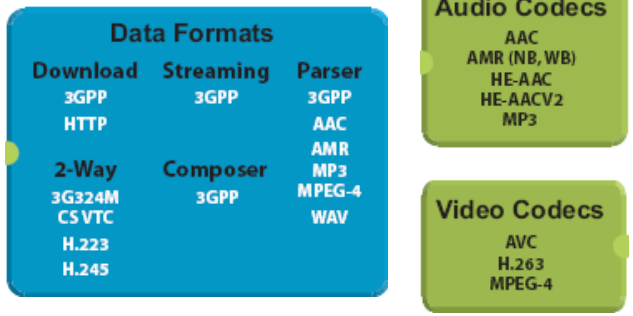


[Google going its own way, forking WebKit rendering engine]

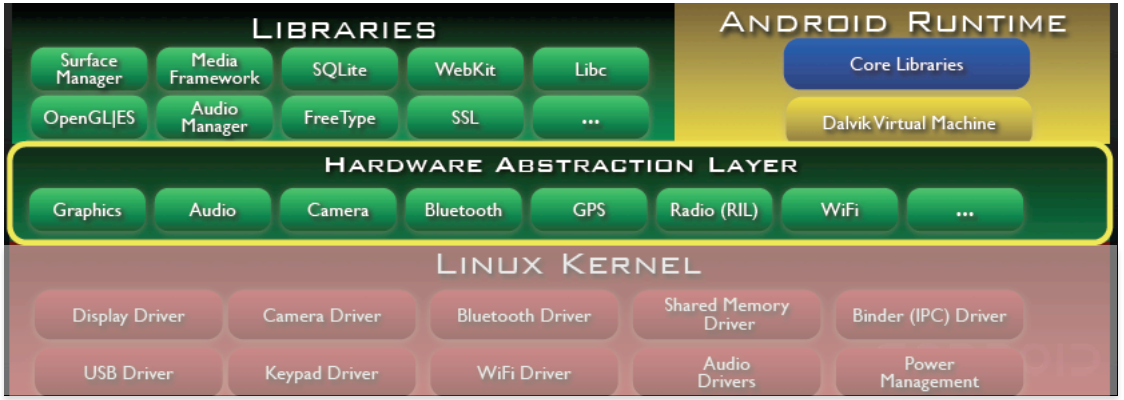
The libraries

- C/C++ libraries for low-level features
- **Bionic libc**
  - Open Source (BSD)
  - Optimized for mobile devices
    - Light (~200Kb)
    - Fast
  - Support for ARM and x86 CPUs
  - Limited support for STL
  - Brand new thread management
- **Webkit**
  - CSS, Javascript, DOM, AJAX
  - Fully compatible with Internet standards
  - Minor modification to adapt to mobile
- **Media Framework**
  - Based on OpenCORE, support for most formats

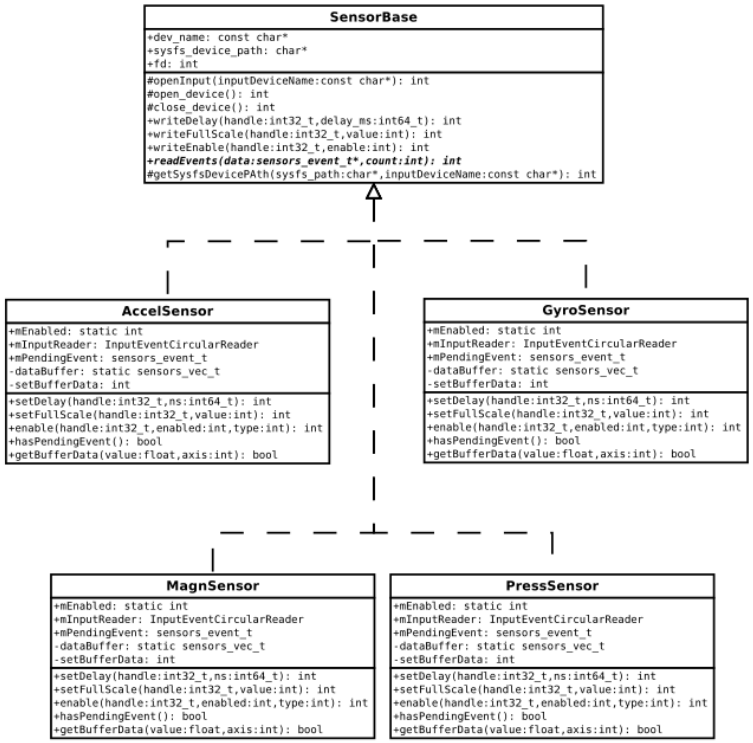




Hardware abstraction layer

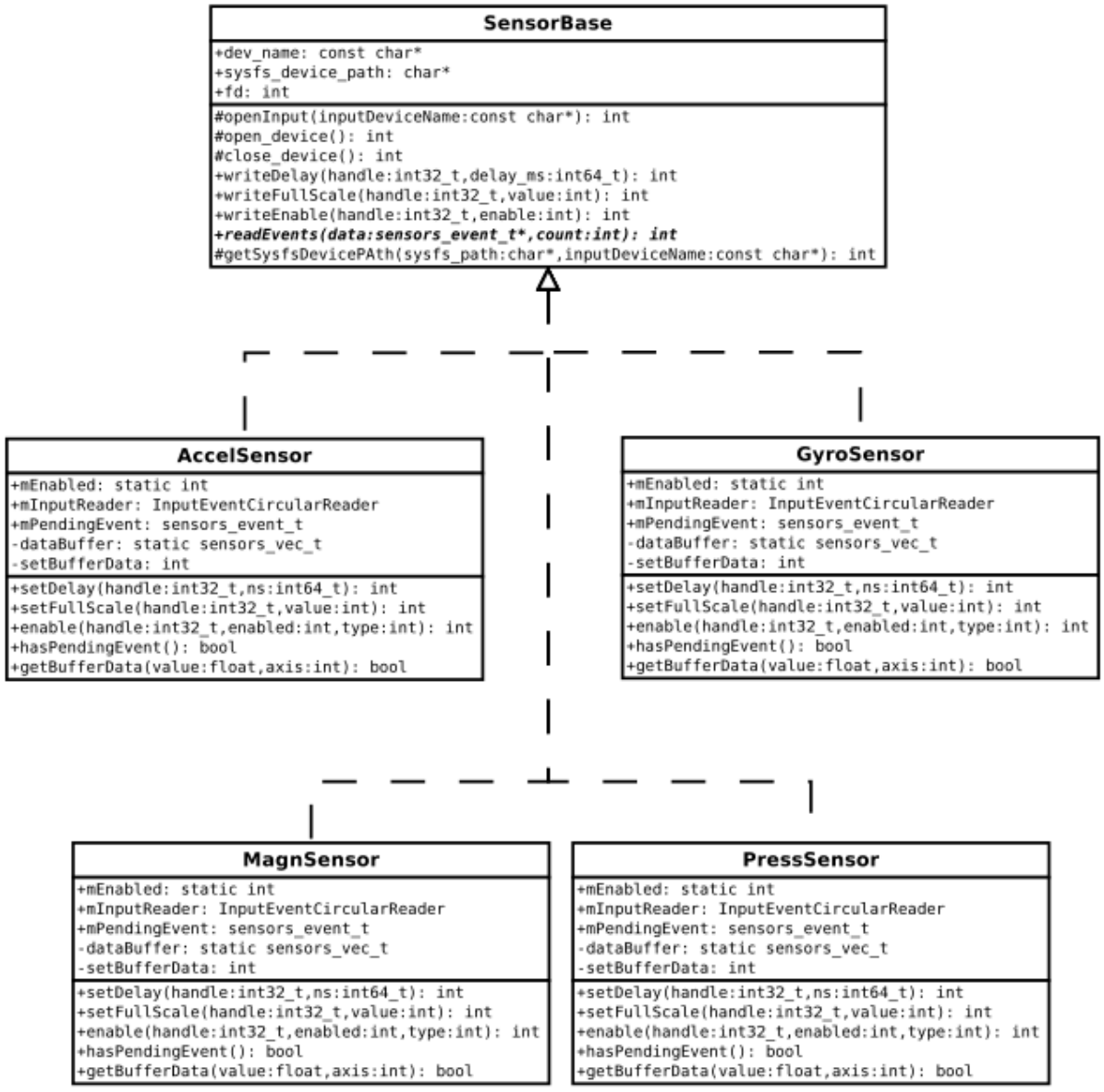


- Between the libraries and Linux kernel
- Separation between **logical platform and hardware interfaces**
- It provides the interfaces API of the drivers
  - Better **portability** of the platform over different hardware



Hardware abstraction layer

- Betweer
- Separati
  hardwar
- It provid
  - Better
    differe



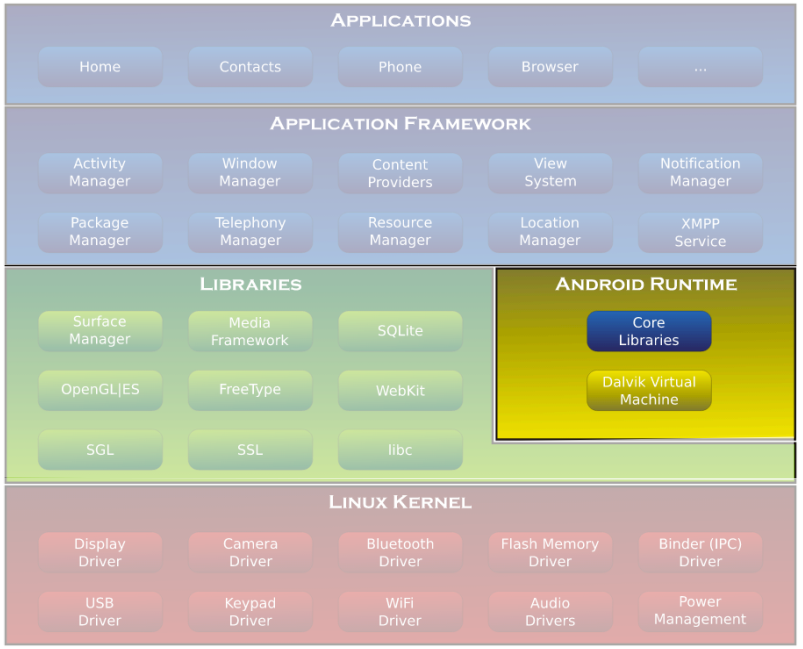
Android Runtime

Java Libraries

- Application are written in Java

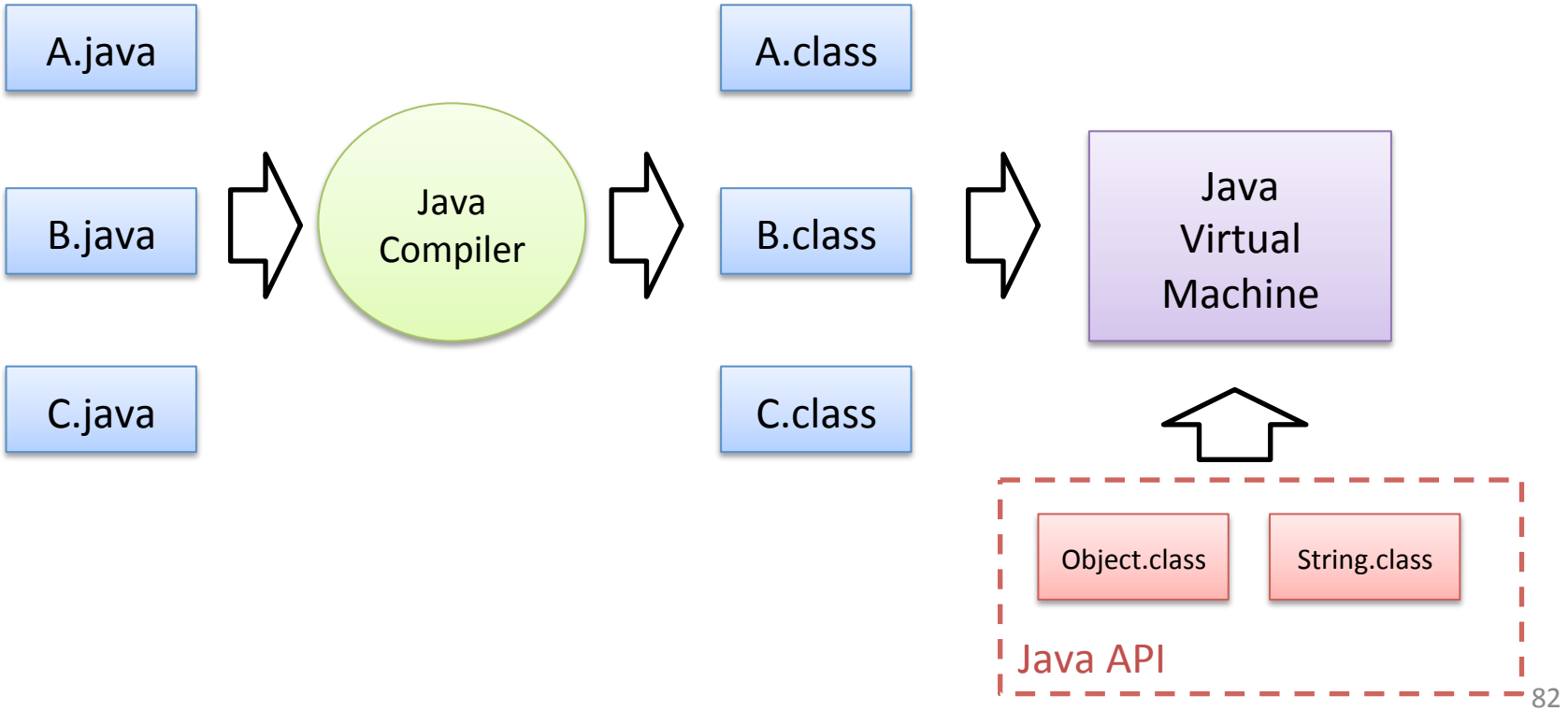
Dalvik virtual machine

- Written by Dan Bornstein
- Named after the fishing village of Dalvík in Iceland
- **A "kind" of** Java Virtual Machine for mobile systems
  - low computational power
  - limited memory
  - Energy efficient
- Use a **specific bytecode** not the Java one
  - .dex applications compiled with "**dx**" utility
- Optimized to be "multi-instance"



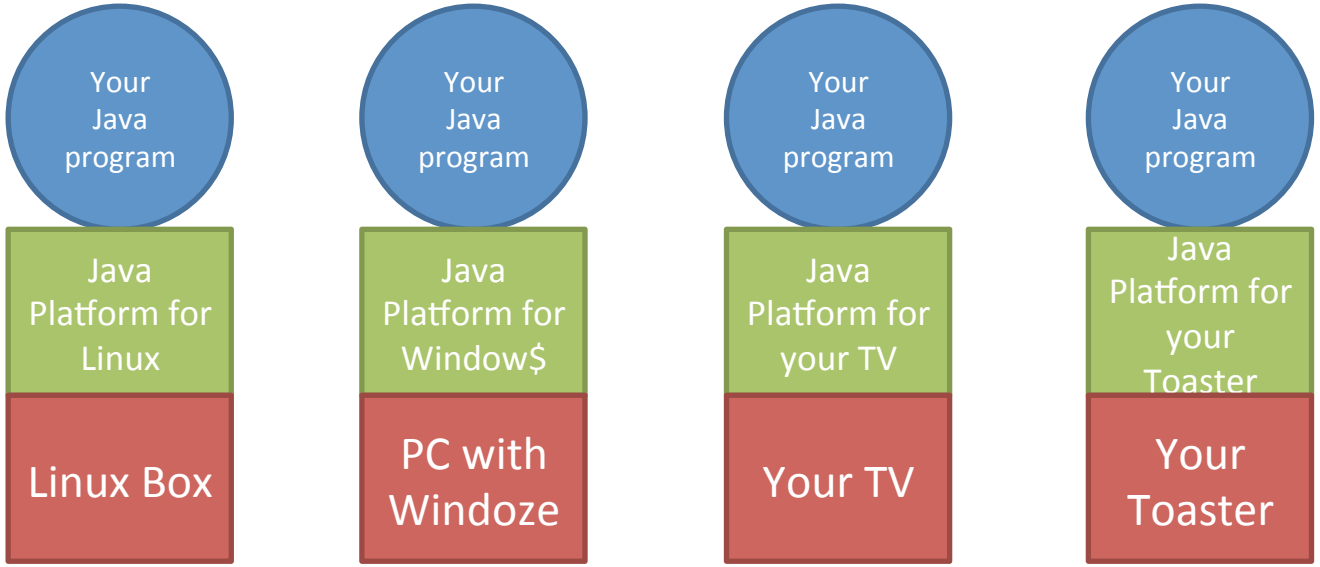
Back to the basics – Java Architecture

- Java is an interpreted language
- The code is not directly compiled into machine-language instructions
- The code is compiled into .class file that are run on the virtual machine
- Access to the system resources using the .class of the Java API



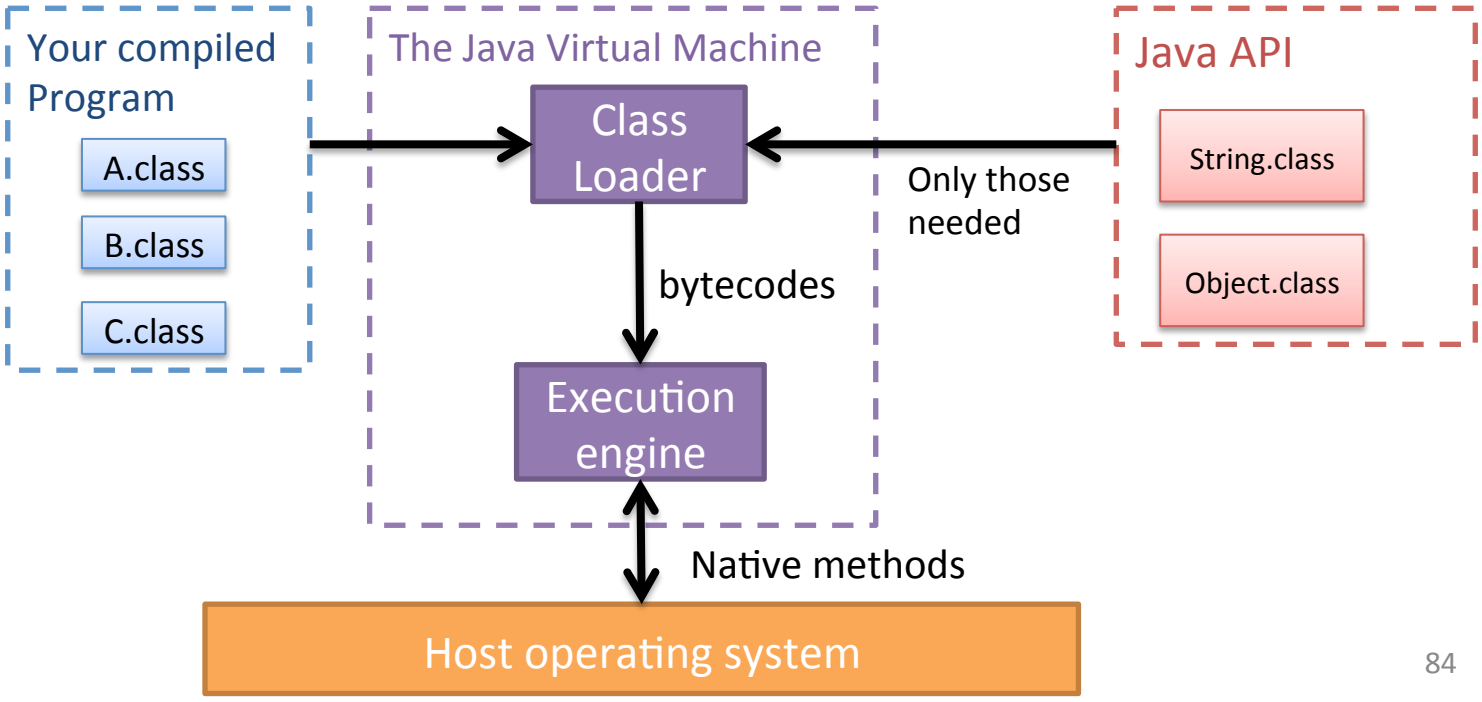
Back to the basics – Java Architecture

- Java Virtual Machine + Java API = **Java Platform** for which they are compiled
- The same java program can run on any device as long as they have an implementation of the Java platform



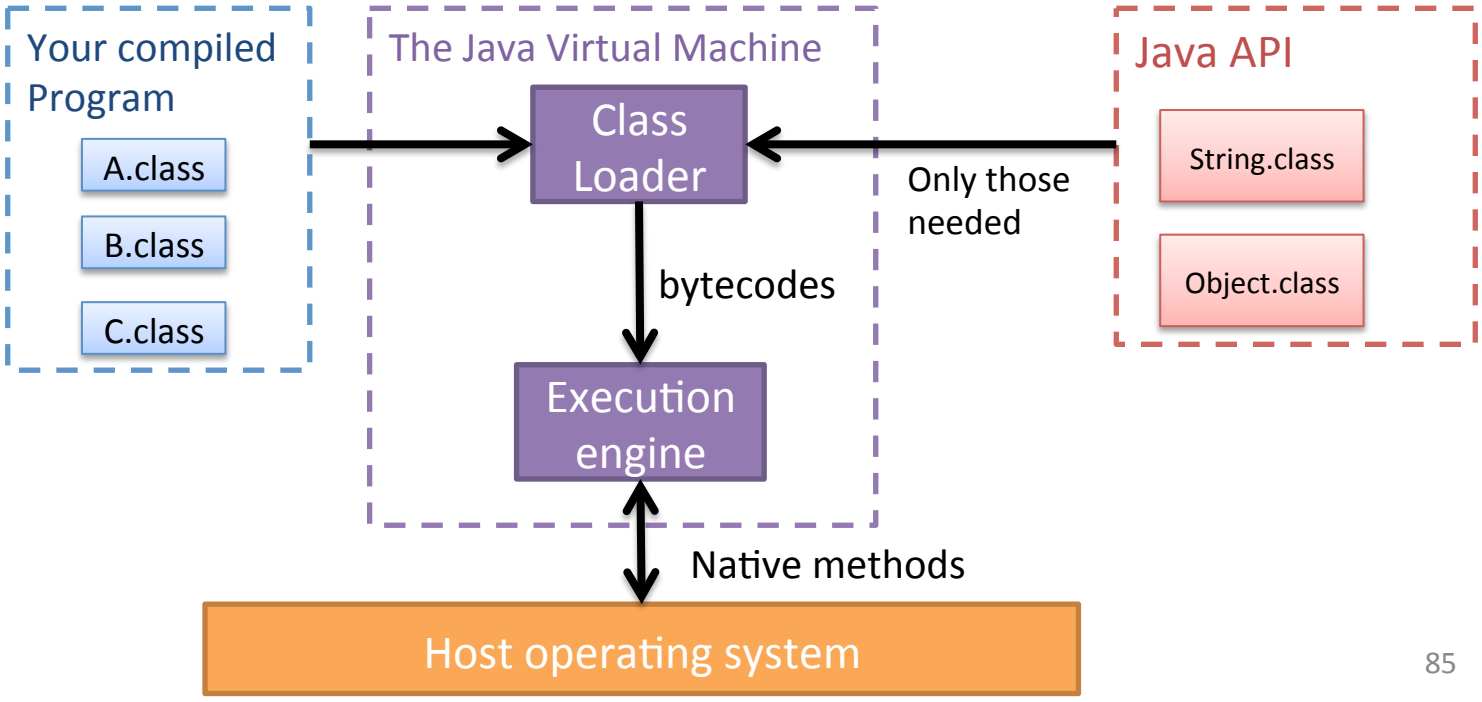
Back to the basics – Java Virtual machine

- **Abstract computer**: the specification define features that every JVM must have
- Implementation can be SW or HW
- It loads the .class files (class loader) and execute the bytecode (execution engine)

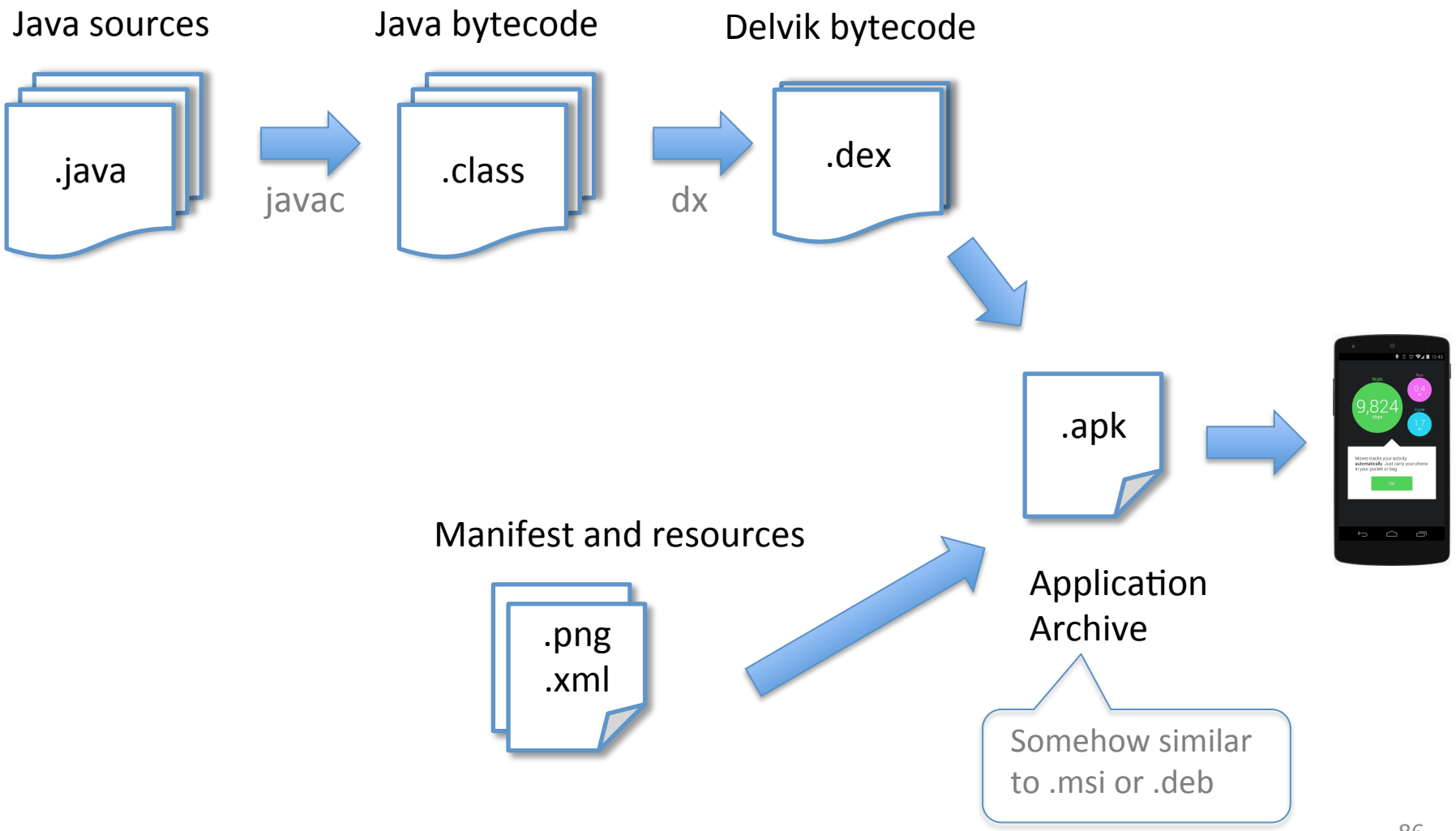


Back to the basics – Java Virtual machine

- **Class loader**: it loads the classes and their bytecodes
- **Execution engine**: it executes the bytecodes
  - Interpretation: direct translation in machine language
  - Just-in-time (JIT): translation and caching for later use
- Overheads…



Android compilation



Android Runtime

- Applications are written in Java using a slightly reduced set of Java packages
  - Based on J2SE 1.5
  - No SWING, printing…
- `java.*, javax.*`
  - Java language, I/O, concurrency
- `android.*`
  - Android Core libraries for application lifecycle, GUI etc.
- `org.`
  - Internet and web services for all internet operations
- `junit.*`
  - Unit testing

Android Runtime

java.io
java.lang
~~java.lang.management~~
java.math
java.net
java.nio
java.security
java.sql
java.text
java.util
javax.crypto
javax.net
javax.security

~~javax.security.auth.spi~~
~~javax.security.auth.kerberos~~
~~javax.security.sasl~~
javax.sound
javax.sql
javax.sql.rowset
javax.xml.parsers
org.w3c.dom
org.xml.sax
~~java.applet~~
~~java.awt~~
~~java.beans~~
~~java.lang.management~~

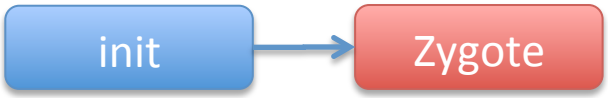
~~java.rmi~~
~~javax.accessibility~~
~~javax.activity~~
~~javax.imageio~~
~~javax.management~~
~~javax.naming~~
~~javax.print~~
~~avax.rmi~~
~~javax.swing~~
~~javax.transaction~~
~~javax.xml~~
javax.xml.parsers
~~org.ietf.*~~
~~org.omg.*~~
~~org.w3c.dom.*~~

Android Runtime

- **Every application run on its own instance of a Delvik Virtual machine**
  - No interaction problems among different applications
  - Every app has its own private memory space (sandboxing)
  - No general crash!
    - If an application crashes along with its (copy of) DVM the others can go on

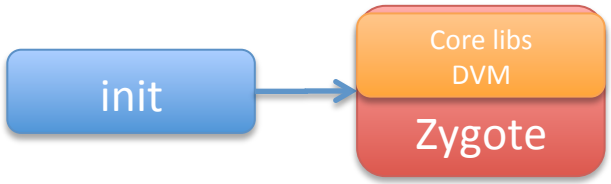
Android Runtime

- **Every application run on its own instance of a Delvik Virtual machine**
  - No interaction problems among different applications
  - Every app has its own private memory space (sandboxing)
  - No general crash!
    - If an application crashes along with its DVM the others can go on
- At the beginning a process **Zygote** is created



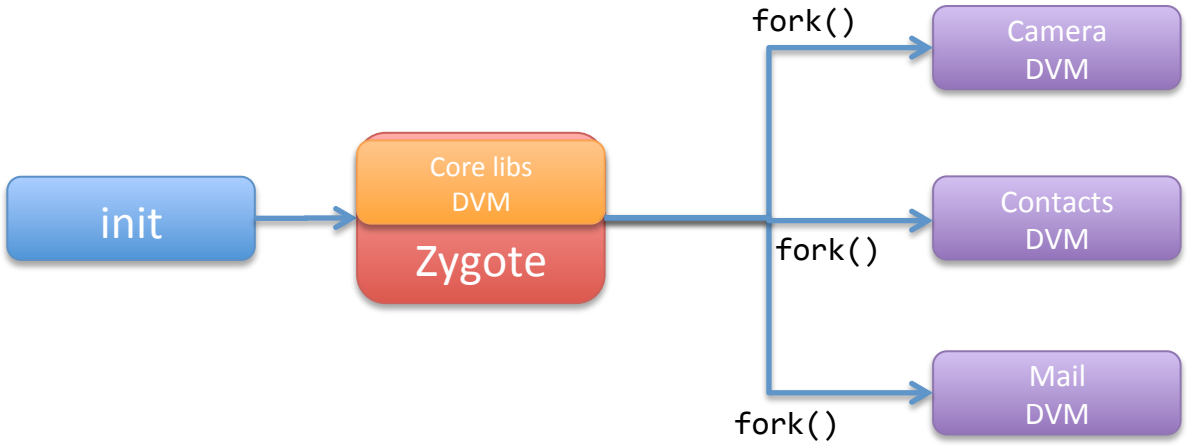
Android Runtime

- **Every application run on its own instance of a Delvik Virtual machine**
  - No interaction problems among different applications
  - Every app has its own private memory space (sandboxing)
  - No general crash!
    - If an application crashes along with its DVM the others can go on
- At the beginning a process **Zygote** is created
  - It creates an instance of a DVM
  - It loads the lib core classes



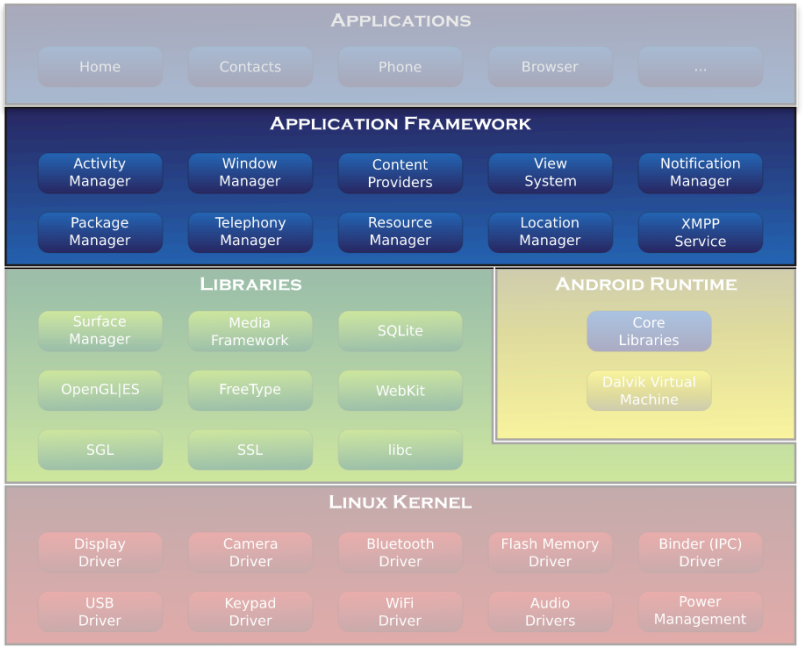
Android Runtime

- **Every application run on its own instance of a Delvik Virtual machine**
  - No interaction problems among different applications
  - Every app has its own private memory space (sandboxing)
  - No general crash!
    - If an application crashes along with its DVM the others can go on
- At the beginning a process **Zygote** is created
  - It creates an instance of a DVM
  - It loads the lib core classes
- Every new application that is launched is a **fork of Zygote**
  - They all share the same copy of the libs (unless the app changes them)



Application Framework

- It provides the API to create the apps
- Based on **Services**
  - Applications always running in background without interacting with the user
  - Provide essential features that can be used by other apps
- **Modular** structure that allows to
  - Get access to the hardware
  - Launch other background services
  - Set alarms and notifications
- General idea is to **re-use components**
  - Each service and application publish its functionality so that can be used by others



Core Services

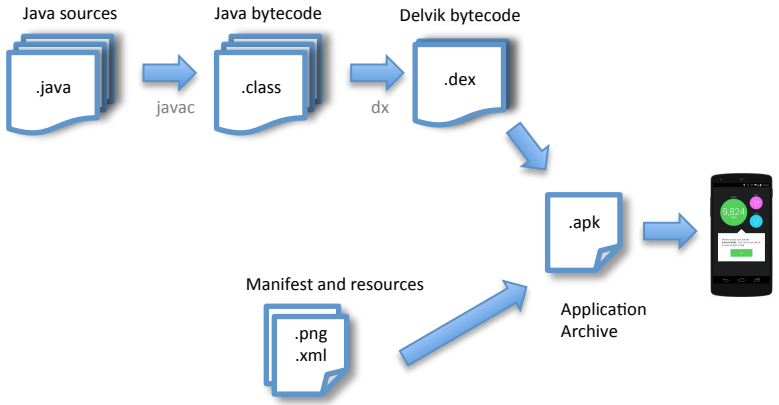
- Activity Manager
  - Maintains the stack of active apps and allows to move back and forth among them
- Package Manager
  - Loads the .apk
- Window Manager
  - Manages the application windows
- Content Provider
  - Provide the data to other apps (eg. contacts, access to multimedia files)
- View System
  - Provides all the graphical GUI elements

Hardware Services

- Telephony service
  - Access to the interfaces related to the phone service (GSM, 3G…)
- Location Service
  - Access to the GPS for localization
- Bluetooth Service
  - Access to the bluetooth interface
- Wi-Fi Service
  - Access to the WiFI interface
- Sensor service
  - Access to the various sensors available on the device
- USB Service …

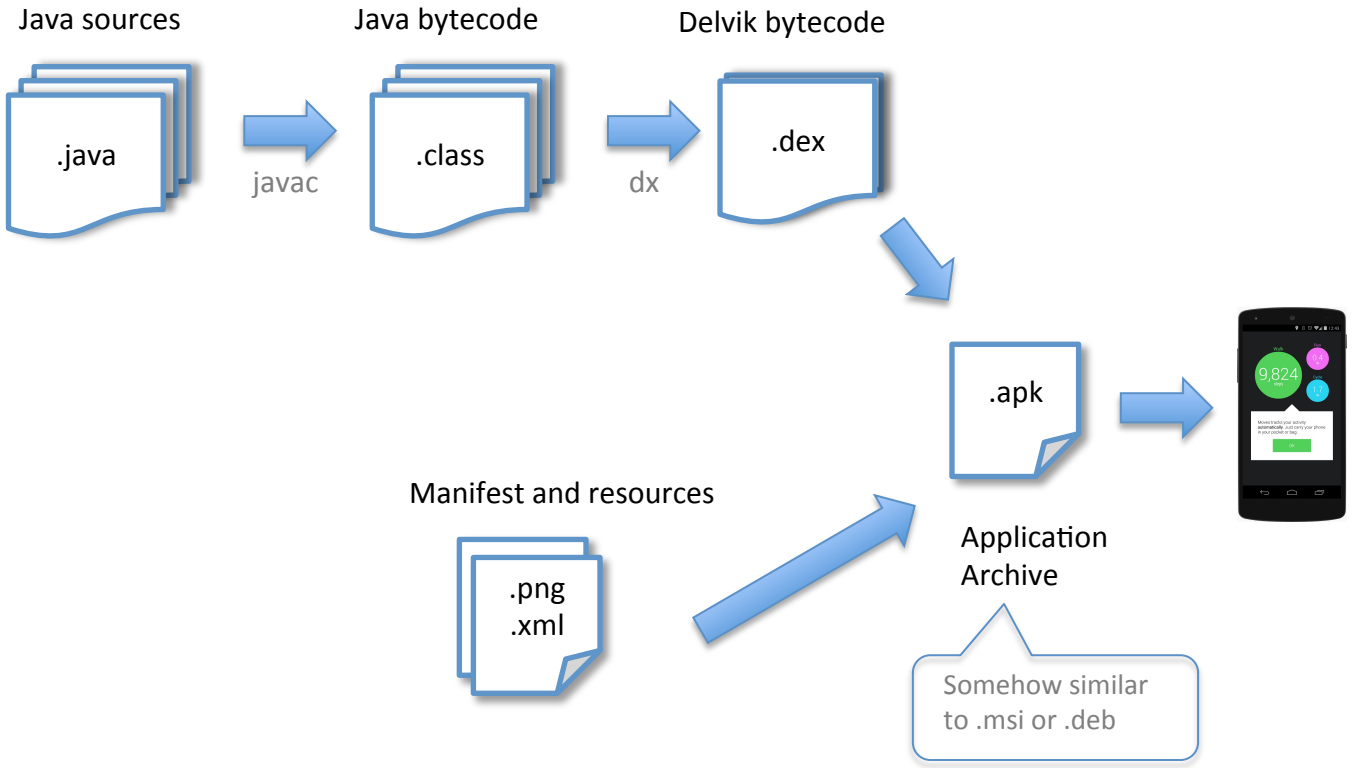
Applications

- Main default applications
  - Mail, calendar, contacts, SMS etc
- Can be replaced with custom applications
  - That's what OEM and carrier company does
- Applications are in general meant to interact with others providing reusable features
- All written in Java using the Android SDK
- Each application run on a fork instance of a DVM
- Sandboxing makes the whole system robust to application crash



Recap

- Android apps are written in **Java**
- Android SDK tools compile your code into an **APK**
  - .dex (Dalvik VM bytecode)
  - Other resources (GUI description files, icons, images…)



Recap

- Once installed, each Android app lives in a **security sandbox**:
  - The Android is a **multi-user Linux system** in which each app is a different user.
  - Each app has a unique Linux user ID
    - only that user ID can access app's files.
  - Each process has **its own virtual machine** (DVM), so an app's code runs in isolation from other apps.
  - By default, every app runs in **its own Linux process**.
    - process starts when app's components need to be executed,
    - Shut down when no longer needed or system need memory for other apps.

Plan

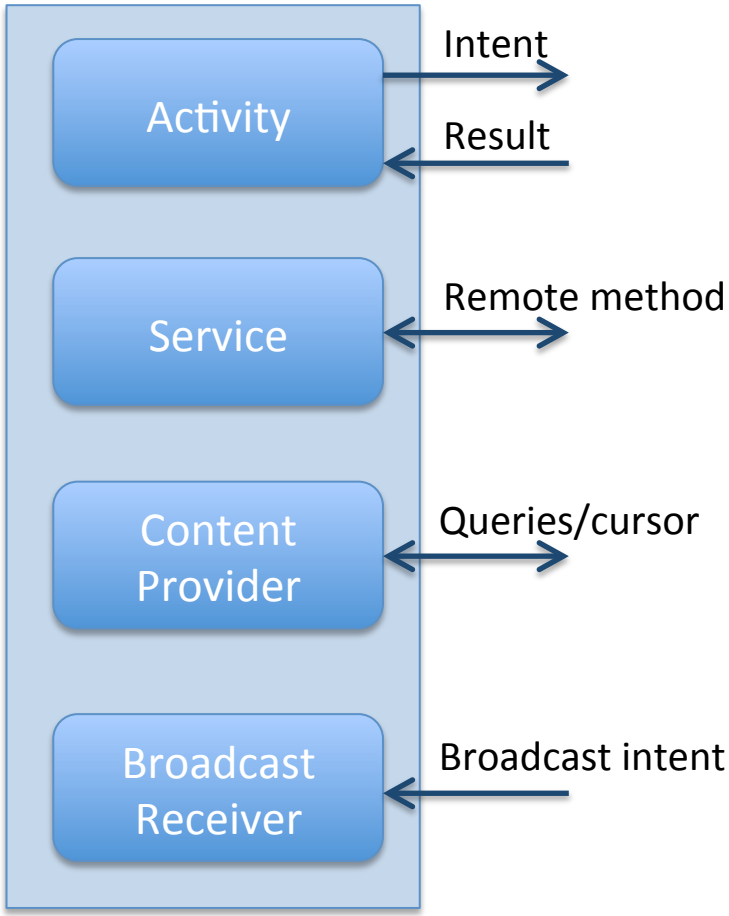
- Mobile devices
- Android overview
  - History
  - Architecture
  - **Applications**
- The tools
- Android Applications

Plan

- Mobile devices
- Android overview
  - History
  - Architecture
  - **Applications**
- The tools
- Android Applications

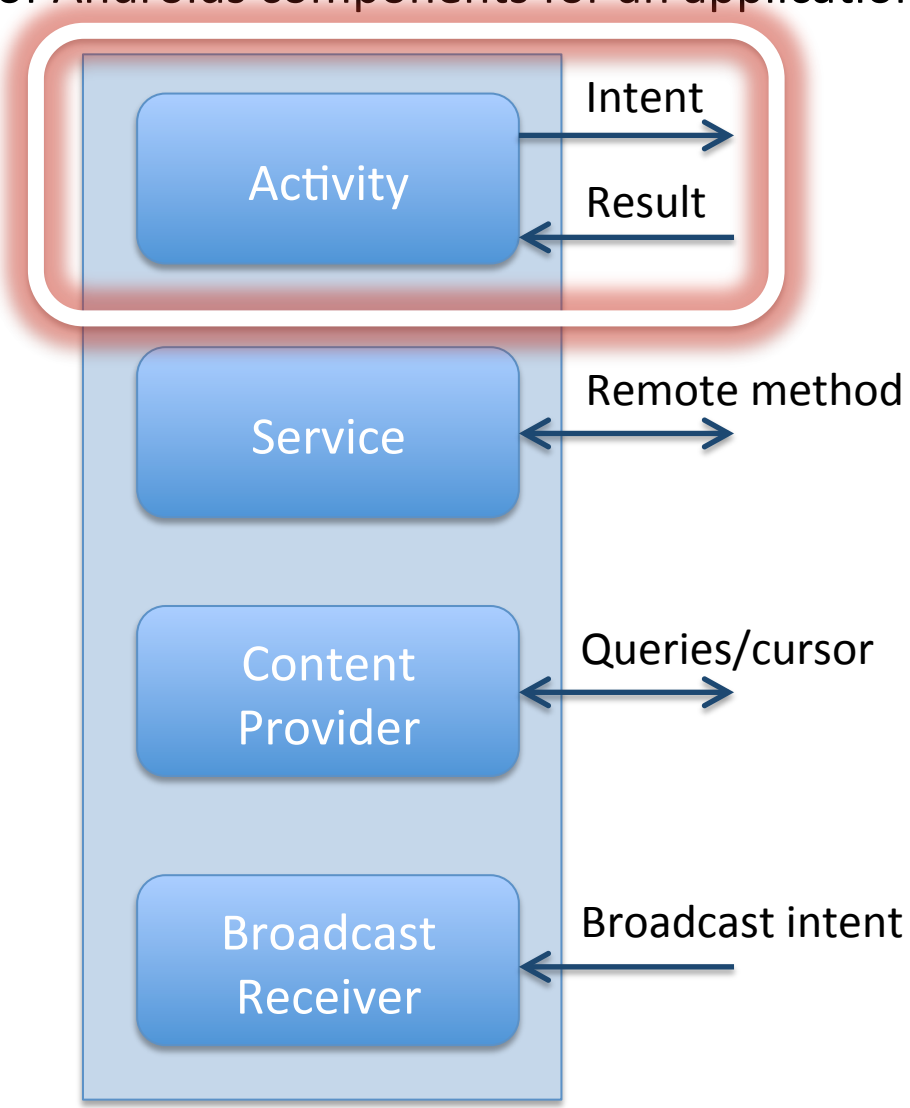
Android applications

4 kinds of Androids **components** for an application



Android applications

4 kinds of Androids components for an application

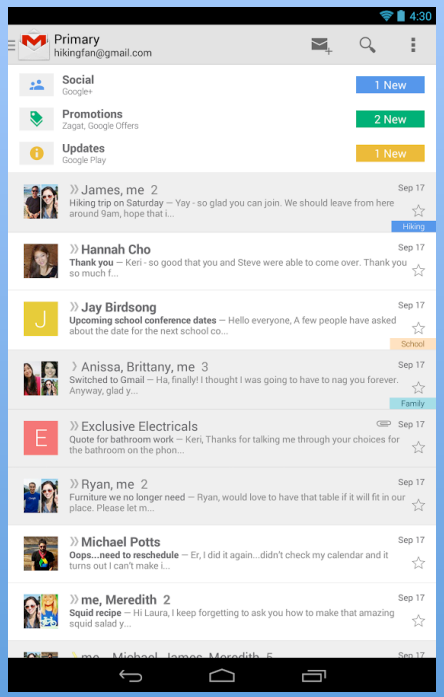


Activity

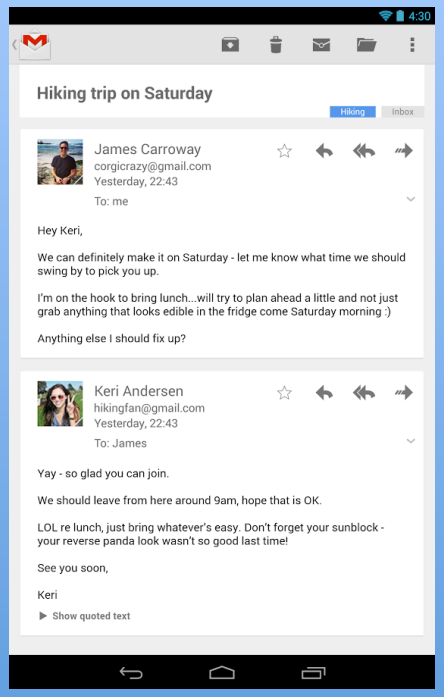
- An Activity represents a **single screen** with **user interface**
- An application is composed of **one or more** activities
- Each one independent from one to another

Activity

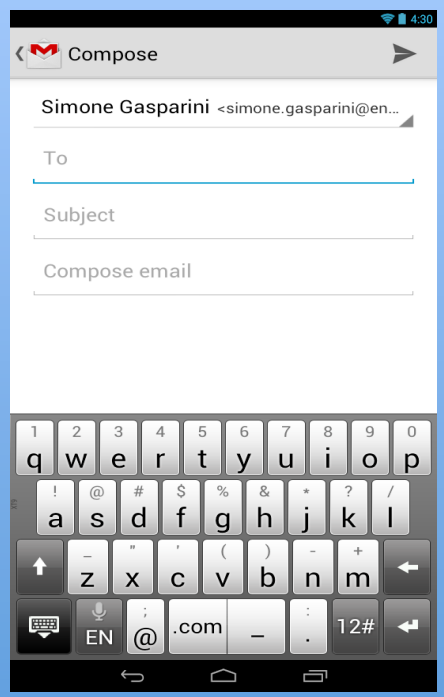
Gmail Application



Activity 1
List of mails



Activity 2
Read mails



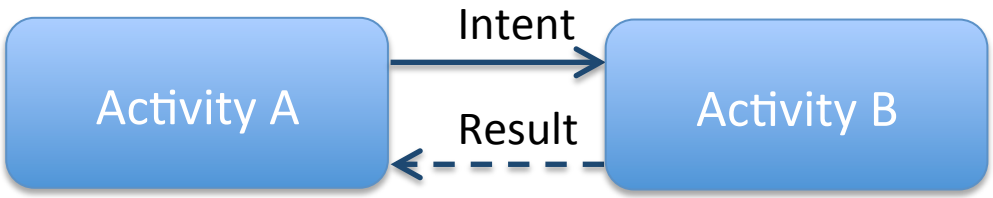
Activity 3
Write mails

Activity

- An Activity represents a **single screen** with **user interface**
- An application is composed of **one or more** activities
- Each one independent from one to another
- Each activity is a **"view"** of the app underlying data
  - As in the **Model-View-Controller** design pattern
- Each activity is Java object of the class `Activity`

Activity

- An application as set of activities
- Application is a **continuous flow** from one activity to another
- Activities are **reusable**, interchangeable parts of the application
- **Communication** between activities
  - Not method calling (API)
  - Use of **intent**



Activities

SMS messaging application

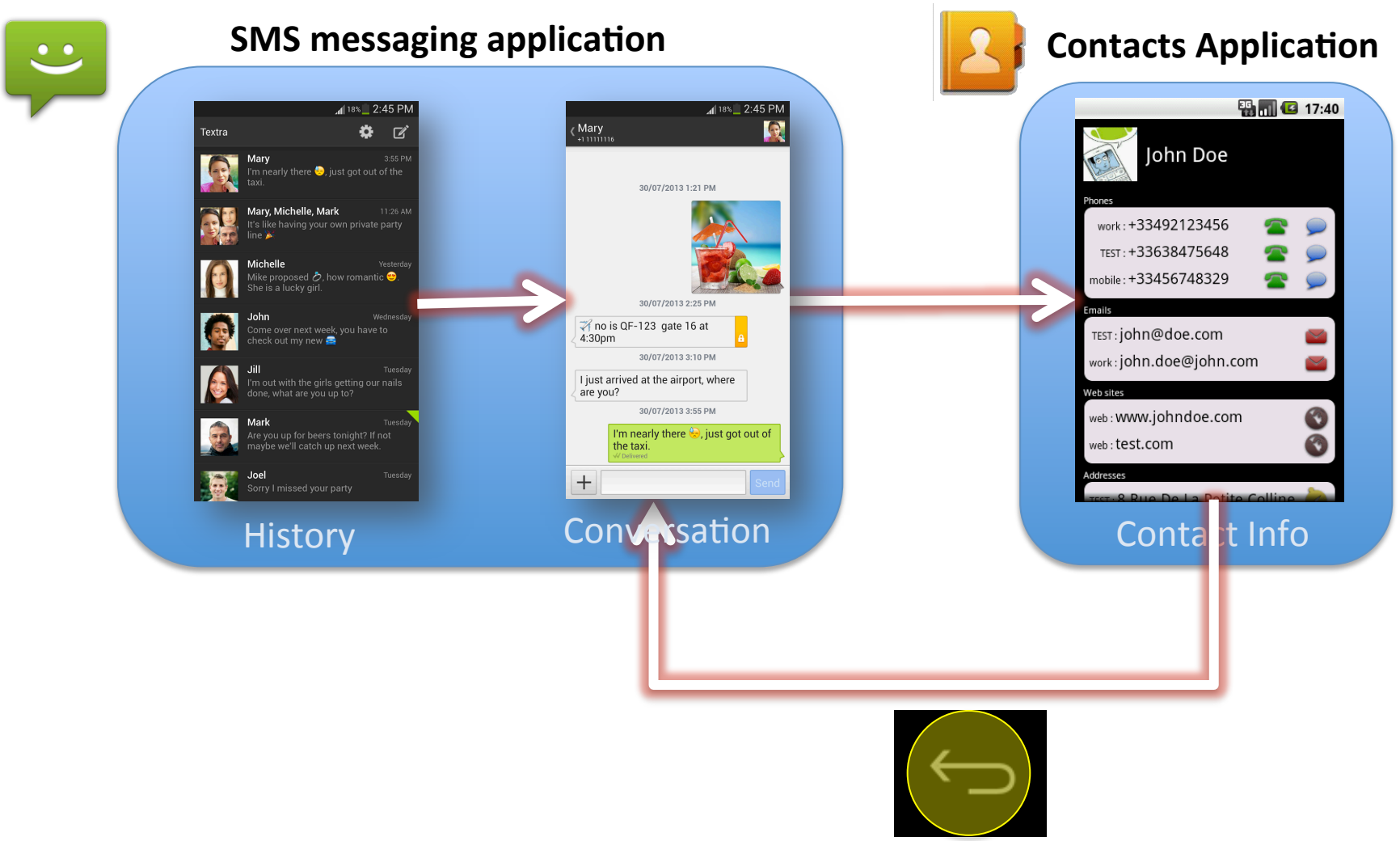
Activities

- Activities are independent objects
- They **can invoke one each other** according to the user actions

Activities

- An application can start an **activity of another application**

Activities

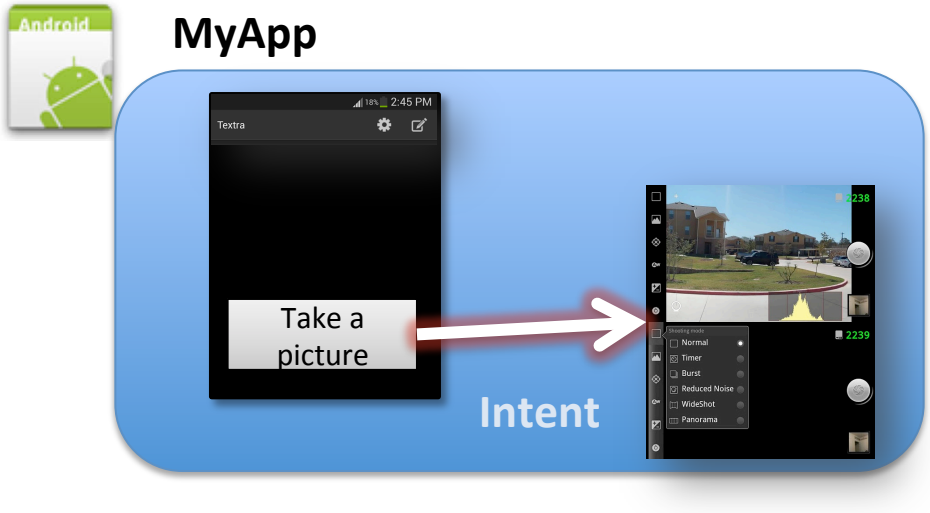

SMS messaging application
Contacts Application
Textra
Mary
I'm nearly there just got out of the taxi.
Mary, Michelle, Mark
It's like having your own private party line
Michelle
Mike proposed how romantic She is a lucky girl.
John
Come over next week, you have to check out my new
Jill
I'm out with the girls getting our nails done, what are you up to?
Mark
Are you up for beers tonight? If not maybe we'll catch up next week.
Joel
Sorry I missed your party
History
Mary
30/07/2013 1:21 PM
30/07/2013 2:25 PM
no is QF-123 gate 16 at 4:30pm
30/07/2013 3:10 PM
I just arrived at the airport, where are you?
30/07/2013 3:55 PM
I'm nearly there just got out of the taxi.
Send
Conversation
John Doe
Phones
work : +33492123456
TEST : +33638475648
mobile : +33456748329
Emails
TEST : john@doe.com
work : john.doe@john.com
Web sites
web : www.johndoe.com
web : test.com
Addresses
Contact Info

Intents

- Asynchronous messages which allow application components to **request functionality** from other Android components
- Interact with components from the own and other applications

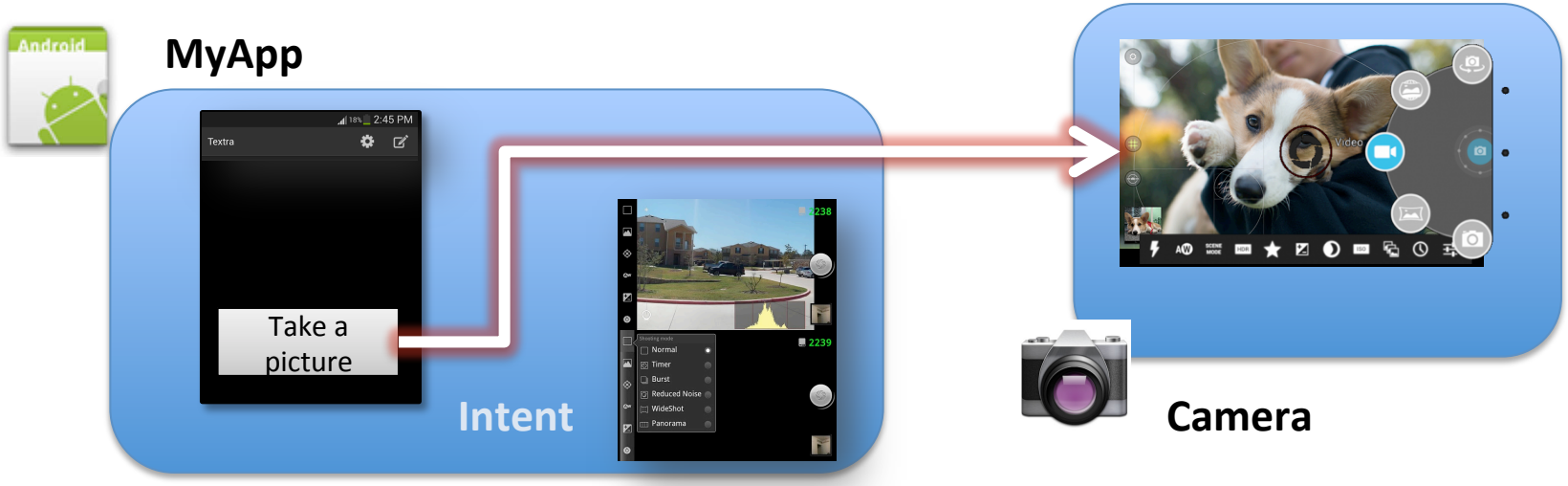
Intents

- Asynchronous messages which allow application components to **request functionality** from other Android components
- Interact with components from the own and other applications
- Example: my application needs to take a picture
  - Start an intent to “take picture”
  - Either implement an activity **on your own (EXPLICIT INTENT)**

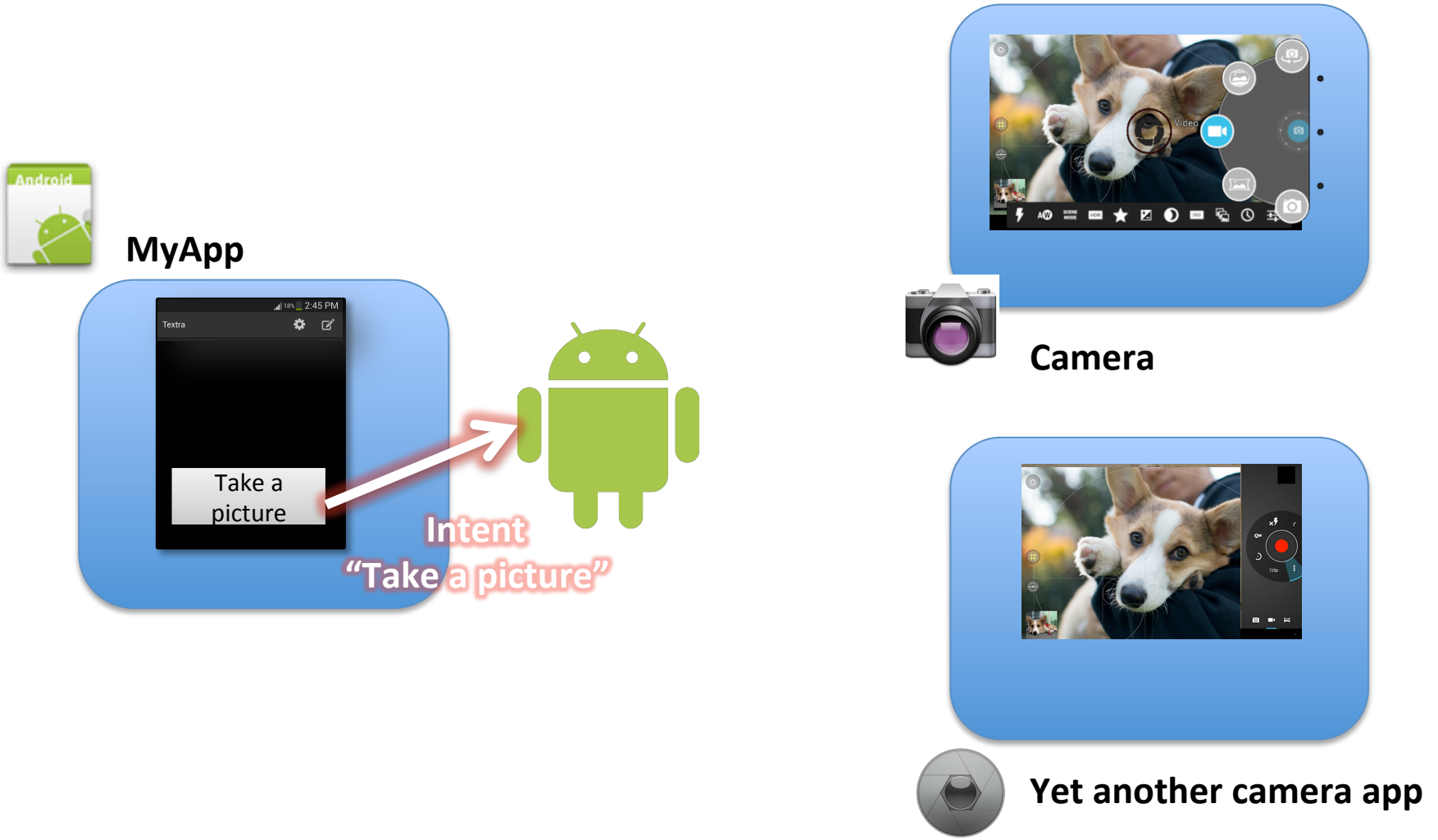


Intents

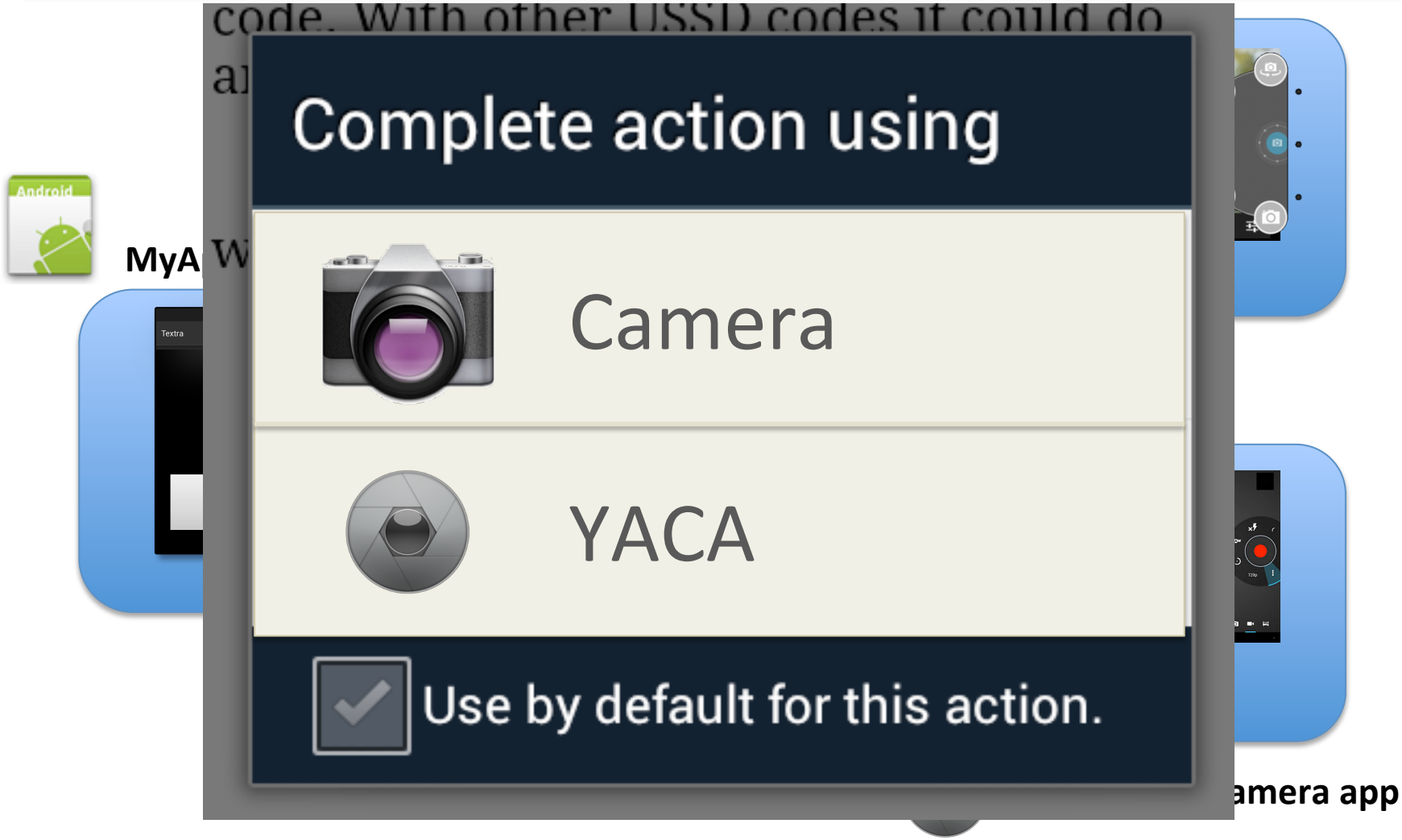
- Asynchronous messages which allow application components to **request functionality** from other Android components
- Interact with components from the own and other applications
- Example: my application needs to take a picture
  - Start an intent to “take picture”
  - Either implement an activity on your own
  - Or ask **available activity(ies)** to take care of it (**IMPLICIT INTENT**)



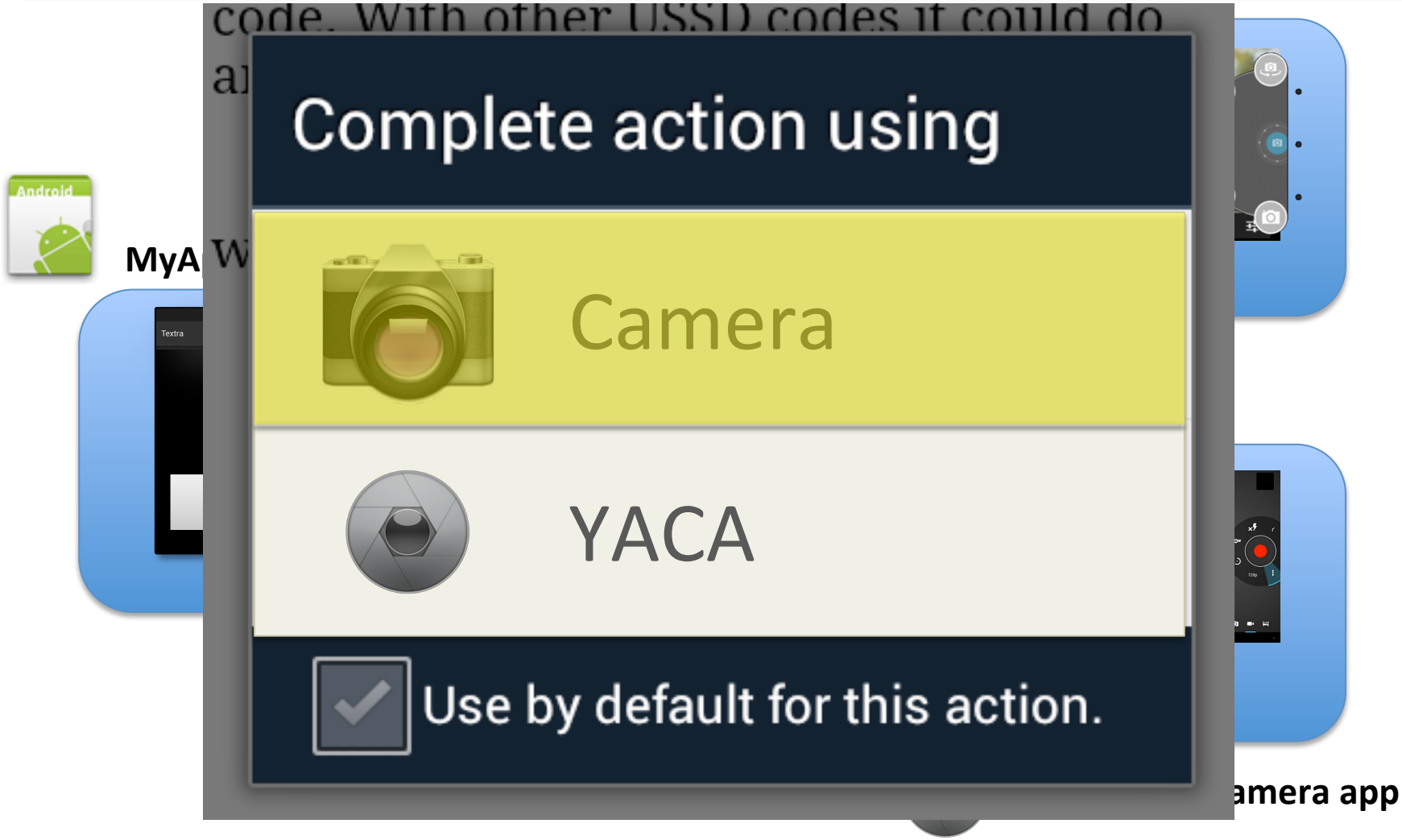
Implicit Intents



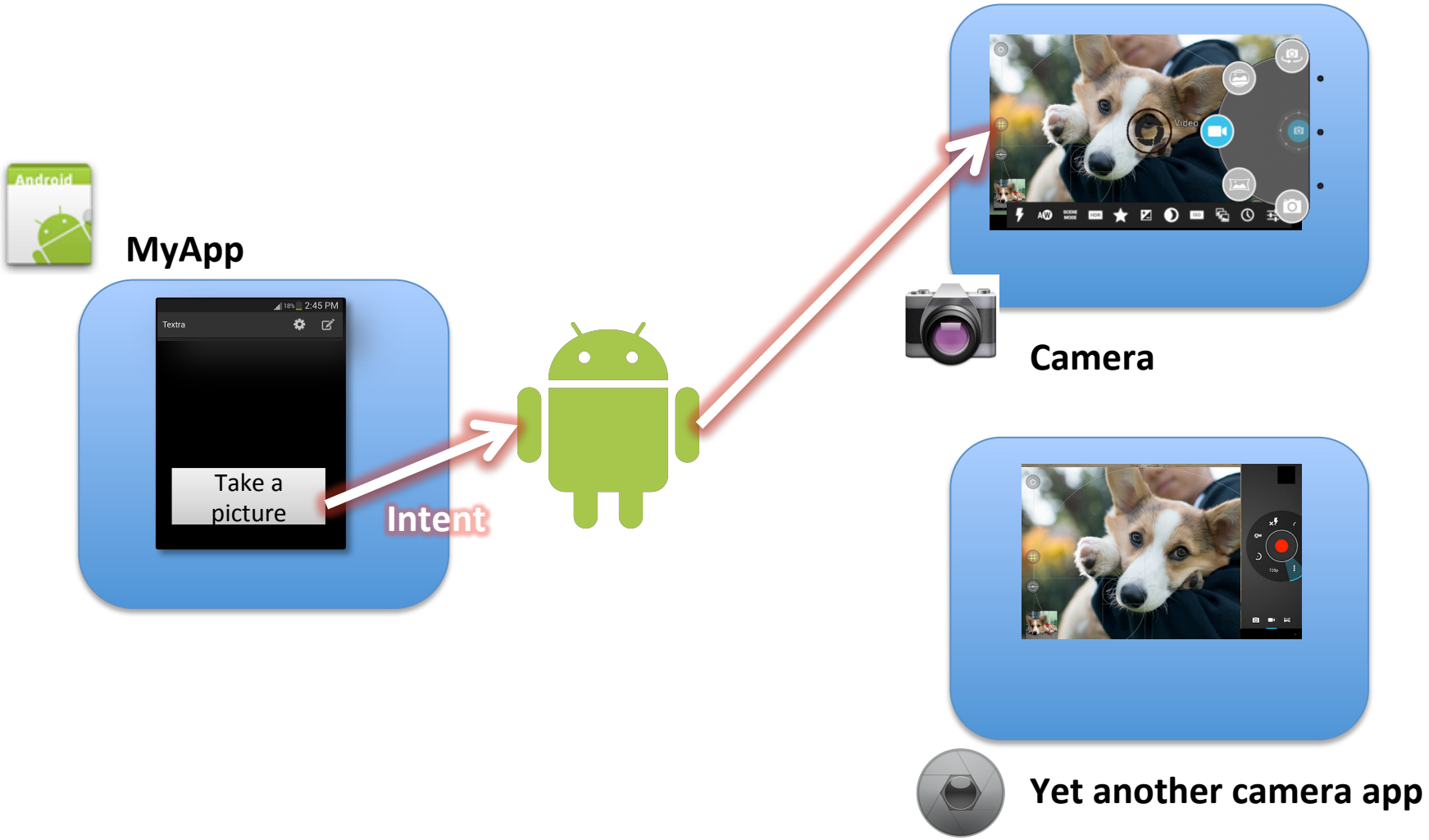
Implicit Intent



Implicit Intent

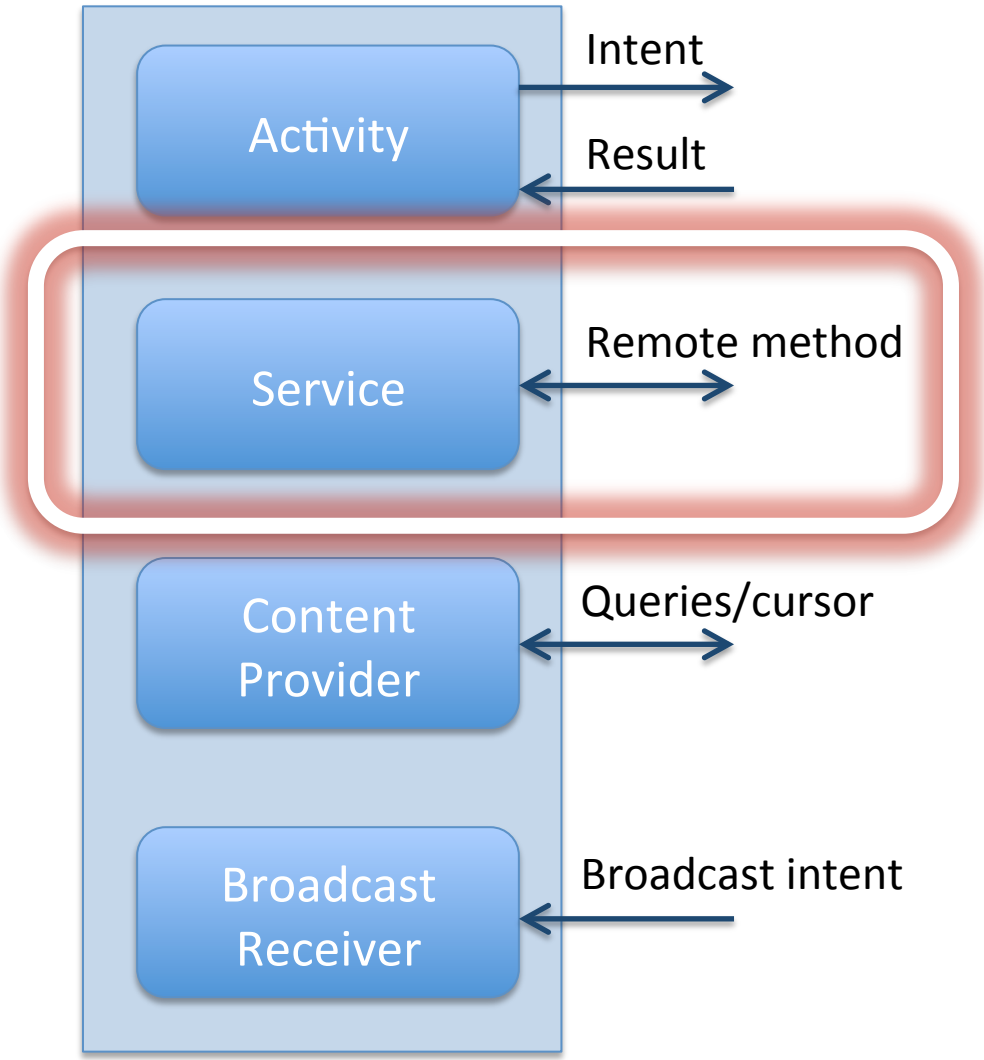


Implicit Intent



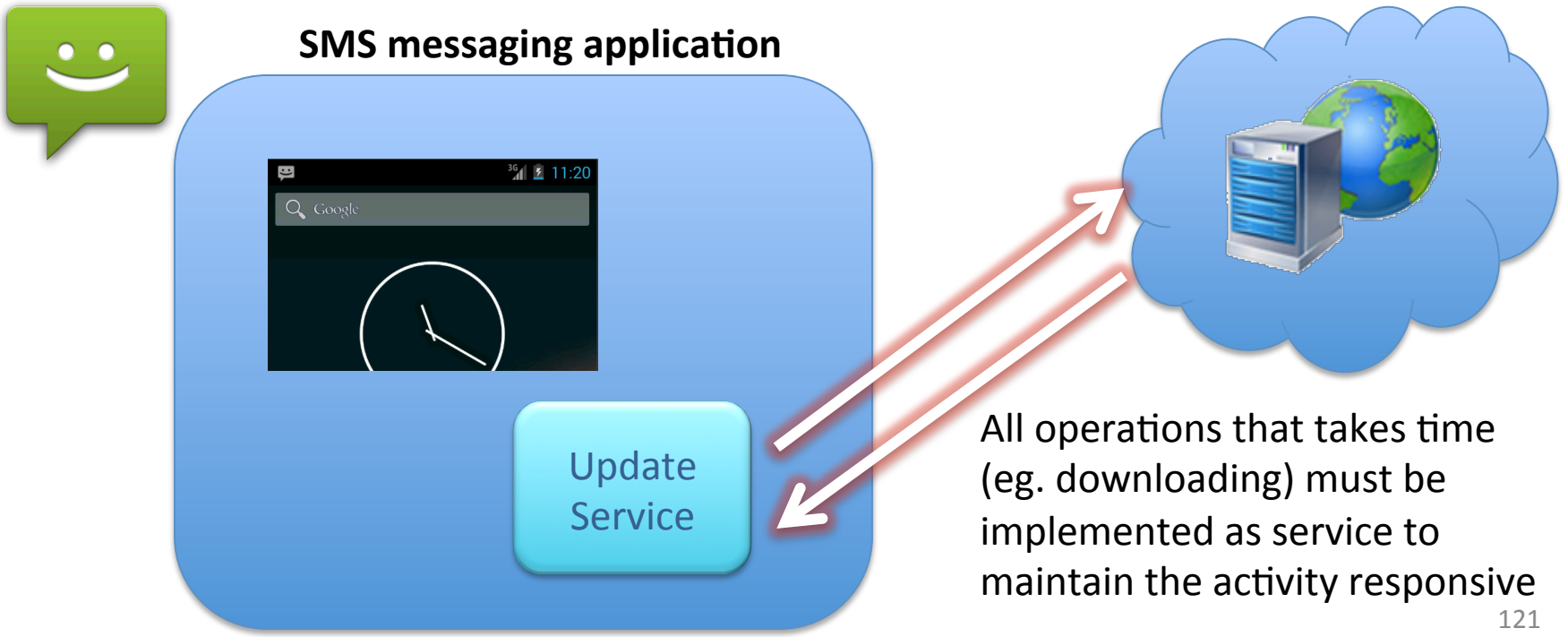
Android applications

4 kinds of Androids components for an application



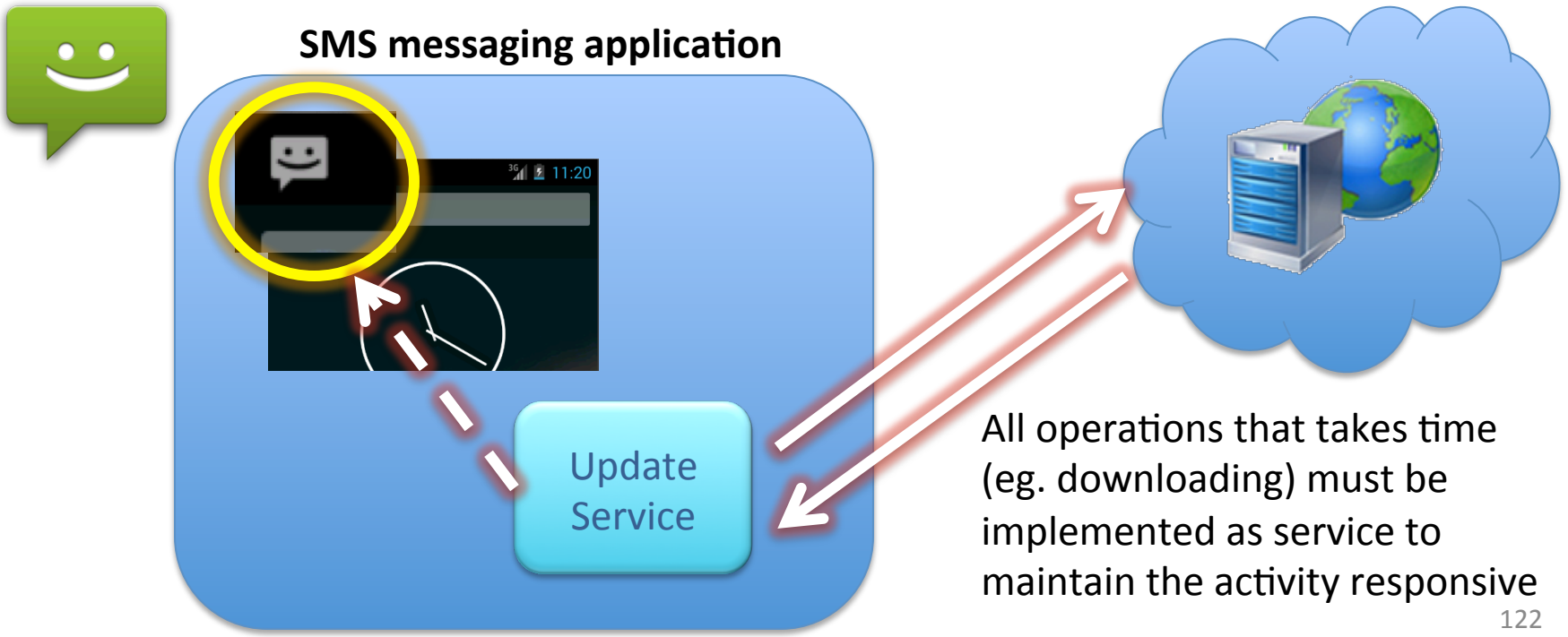
Services

- A service does **not have a GUI**
- It always run in **background**
  - Music player, GPS tracker, SMS listener, Incoming Phone calls…
- It cannot be killed to free resources



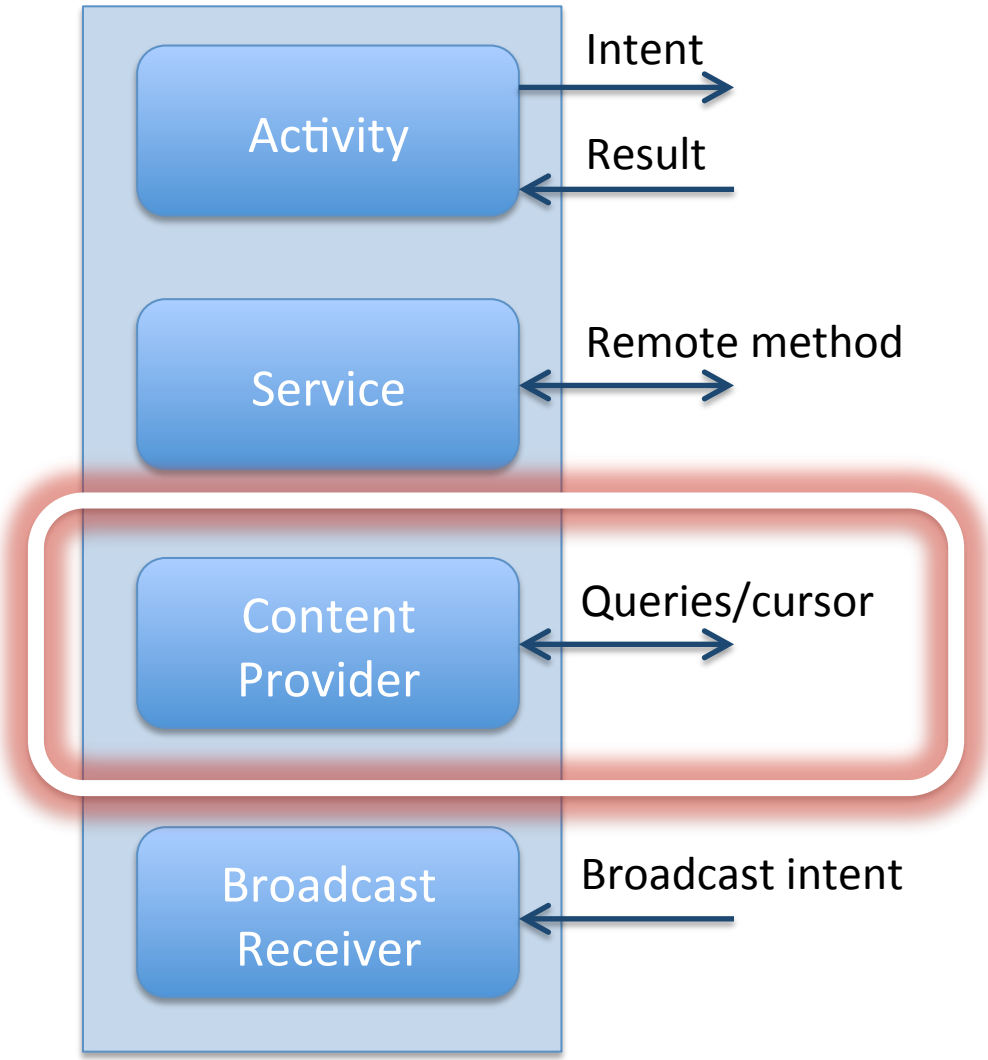
Services

- A service does **not have a GUI**
- It always run in **background**
  - Music player, GPS tracker, SMS listener, Incoming Phone calls…
- It cannot be killed to free resources



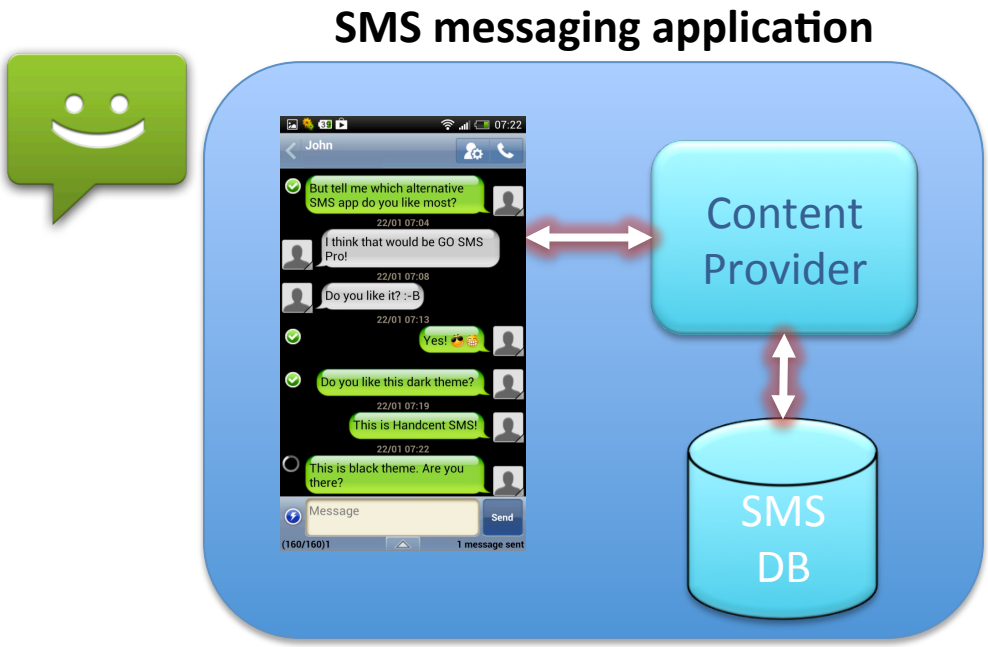
Android applications

4 kinds of Androids components for an application



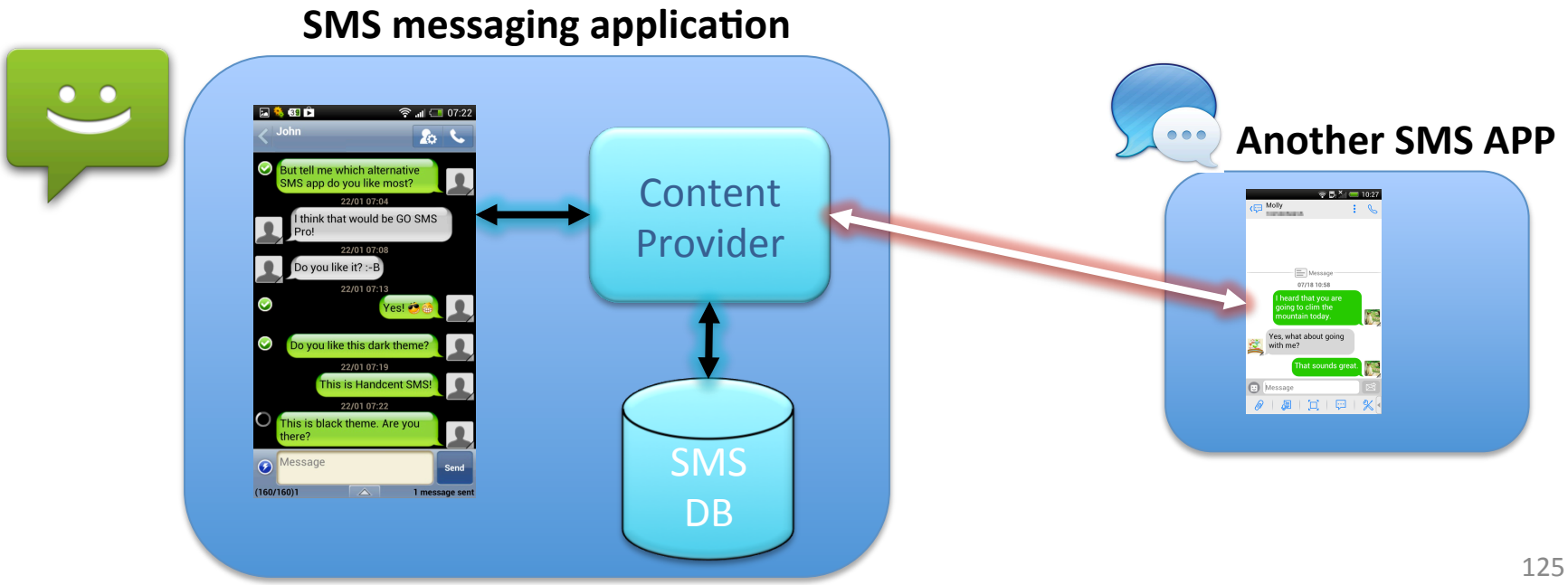
Content provider

- A content provider allows applications to **access** and **share data**
- It can be used:
  - within an application to access **its data**
  - to share private data with other application.



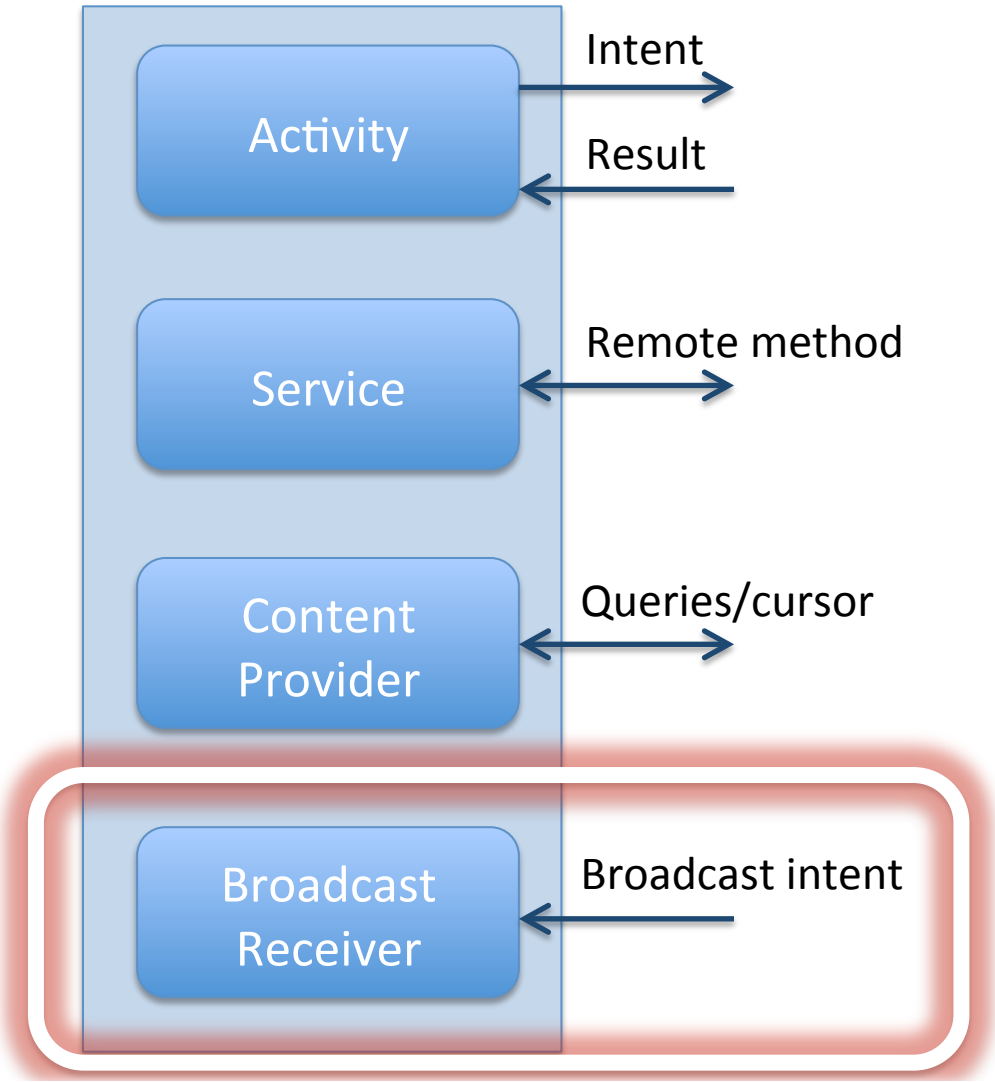
Content provider

- A content provider allows applications to **access data**
- It can be used:
  - within an application to access its data
  - to share private data with **other applications**.
- An interface to the application private DB



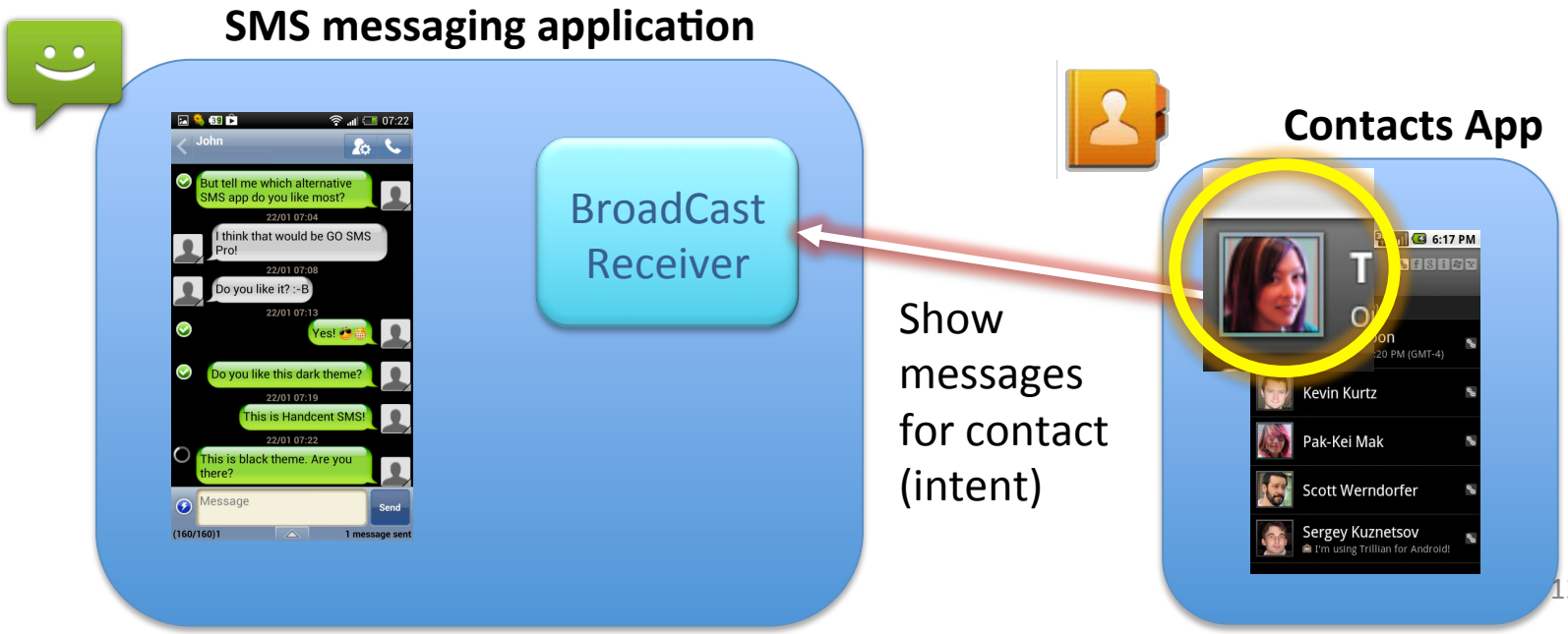
Android applications

4 kinds of Androids components for an application



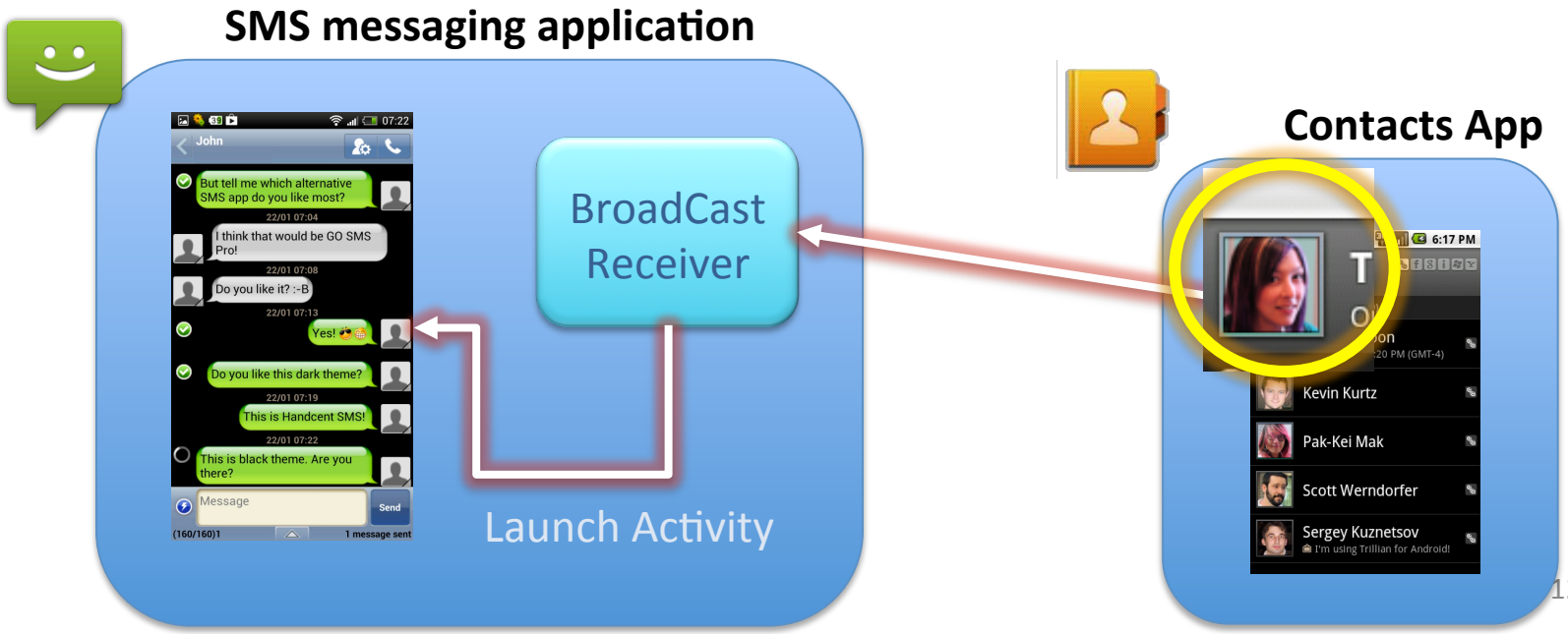
Broadcast receiver

- Allows the application to register for system or application events or **intents**
- All registered receivers for an event will be notified by the Android runtime once this event happens.
- Not associated to a GUI
- It can launch activities if needed



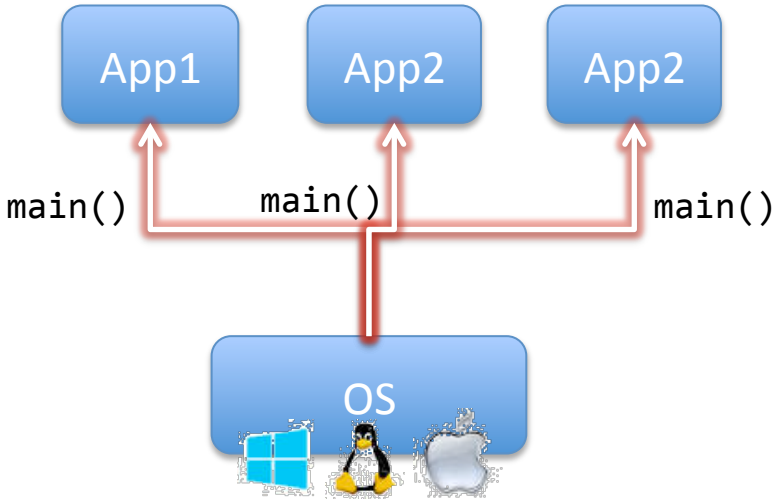
Broadcast receiver

- Allows the application to register for system or application events or **intents**
- All registered receivers for an event will be notified by the Android runtime once this event happens.
- Not associated to a GUI
- It can launch activities if needed

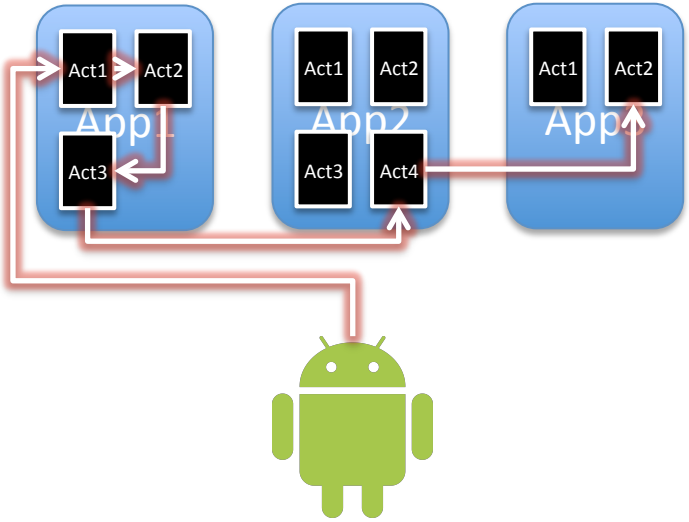


Traditional Vs Android applications

- OS call "`main`", the only "**entry point**" of the application
- Communication among applications is limited

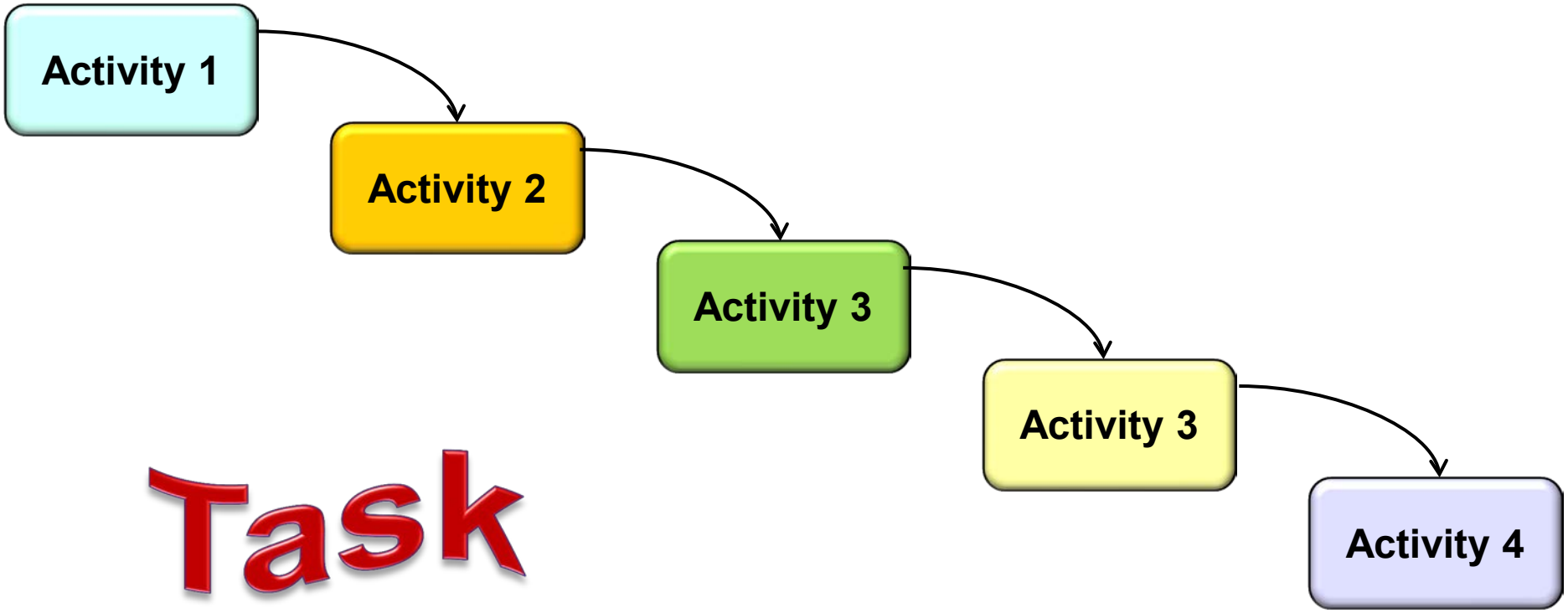


- Application may have multiple entry point
- Applications can start at different places
- It depends on the **user flow** and **behaviour**



Task

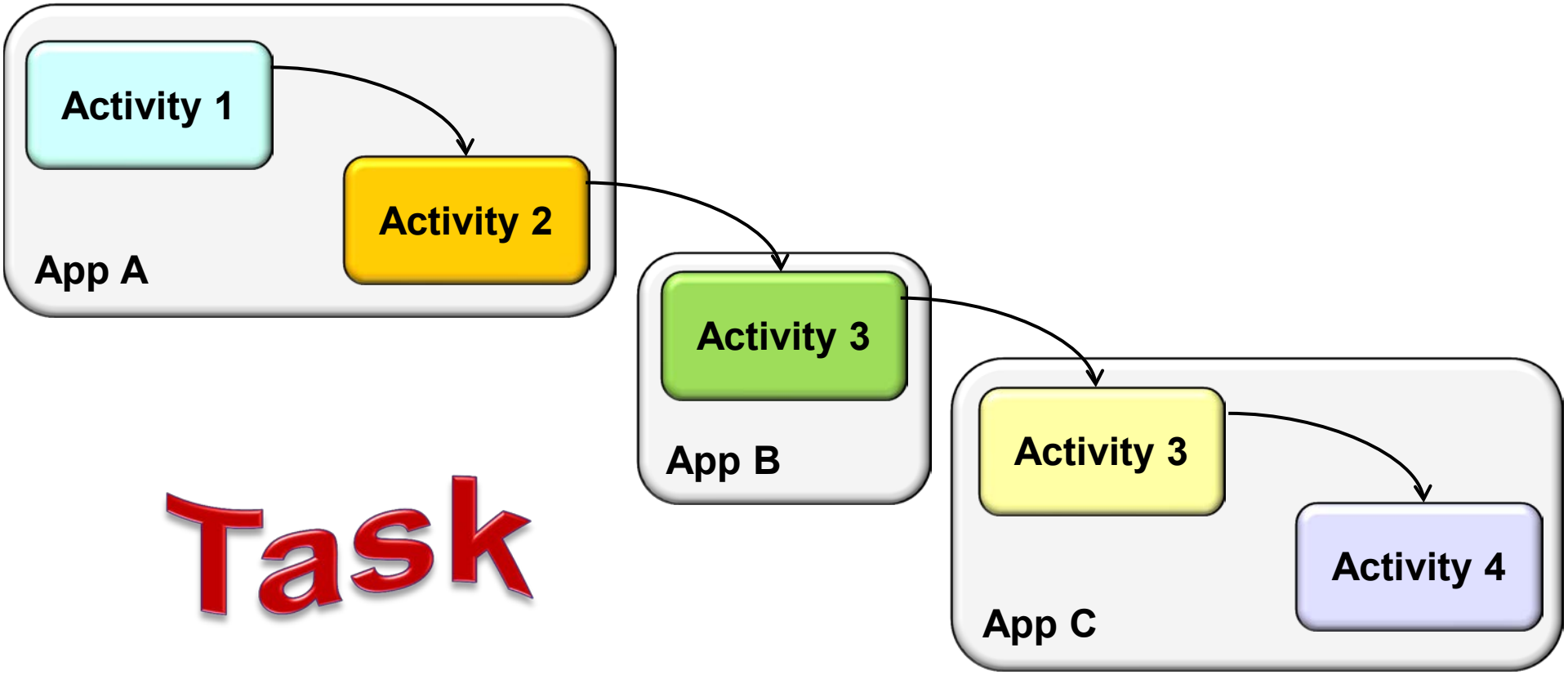
- A Task is a chain of related Activities



http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Task

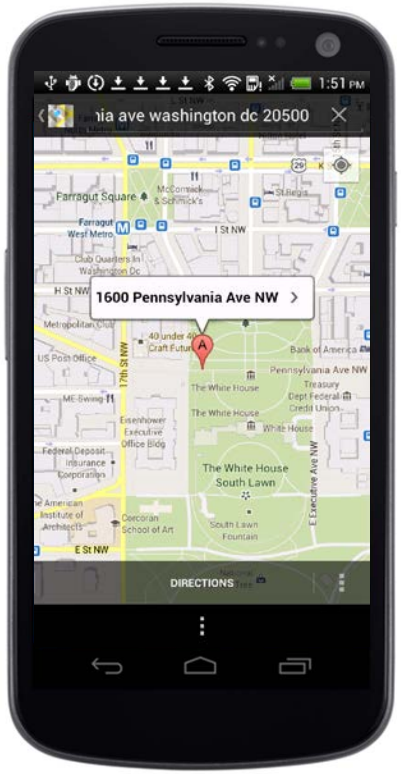
- A Task is a chain of related Activities
- Task are not necessarily provided by a single app



http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Task

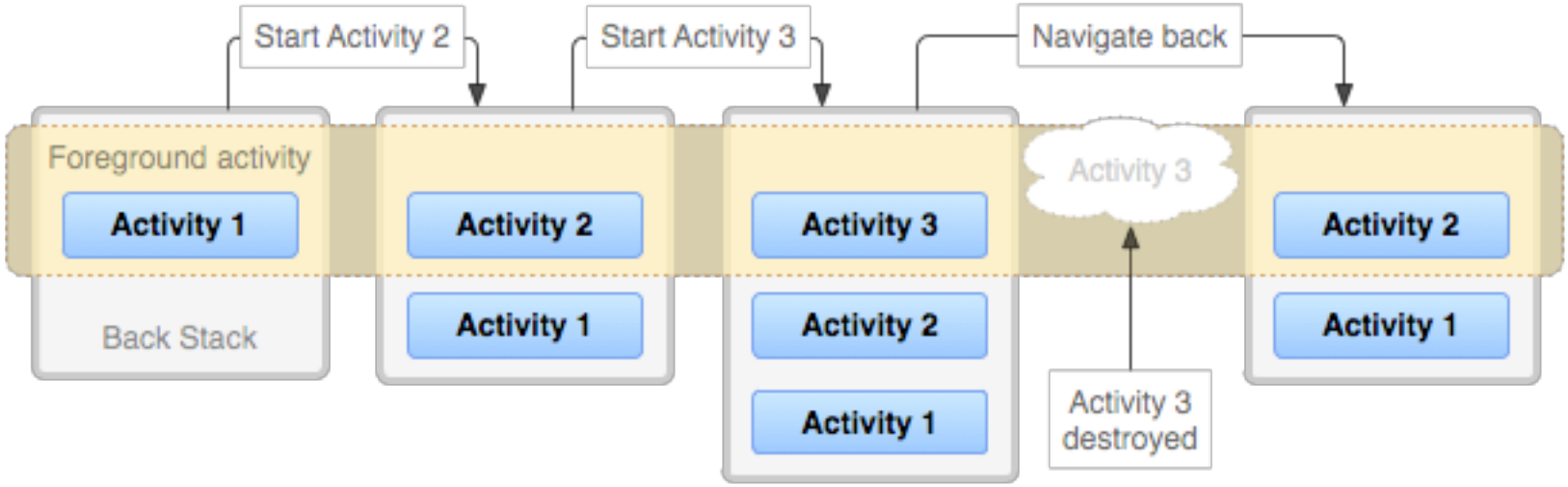
- A Task is a **chain of related activities**
- Task are not necessarily provided by a single app
- Tasks give the illusion that multiple (often unrelated) Activities were developed as part of the same app



http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Task

- The task's Activity objects are stored on a "**back stack**" with the currently running Activity at the top



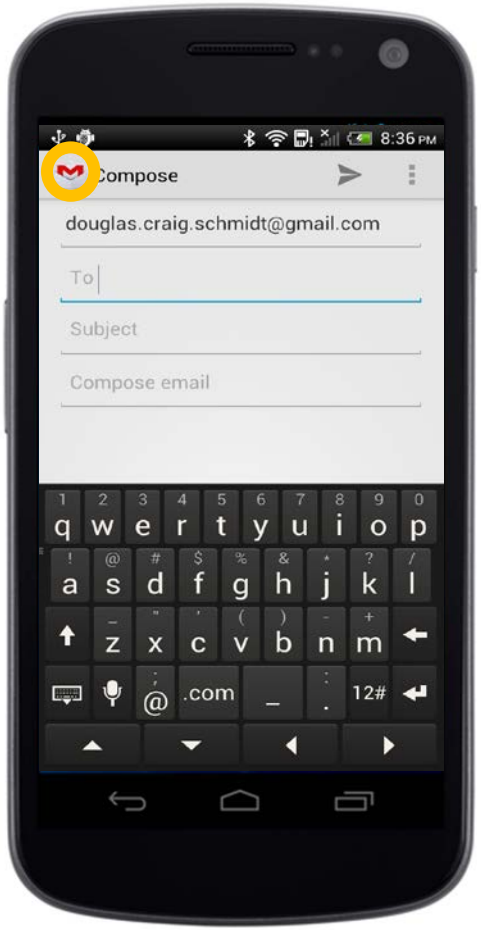
http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

Task

- The task's Activity objects are stored on a "**back stack**" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack

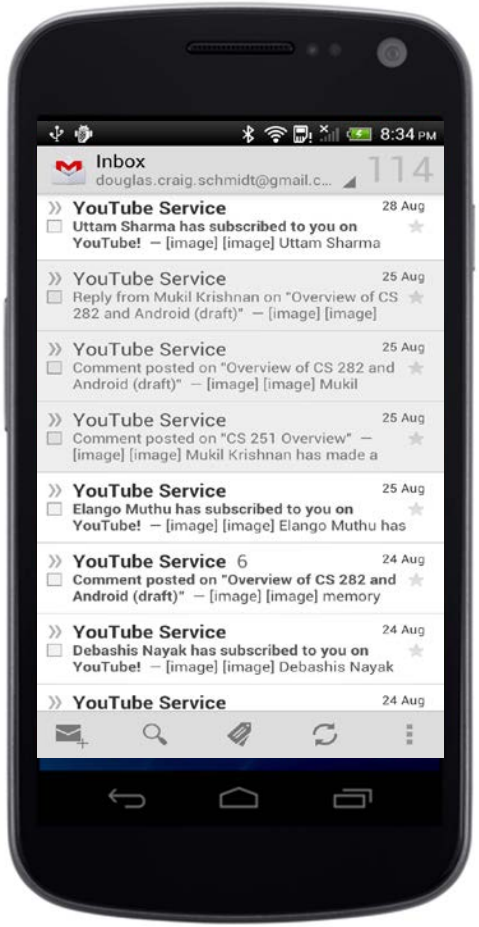
Task

- The task's Activity objects are stored on a "**back stack**" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack
  - Finishing an Activity pops it off the stack...



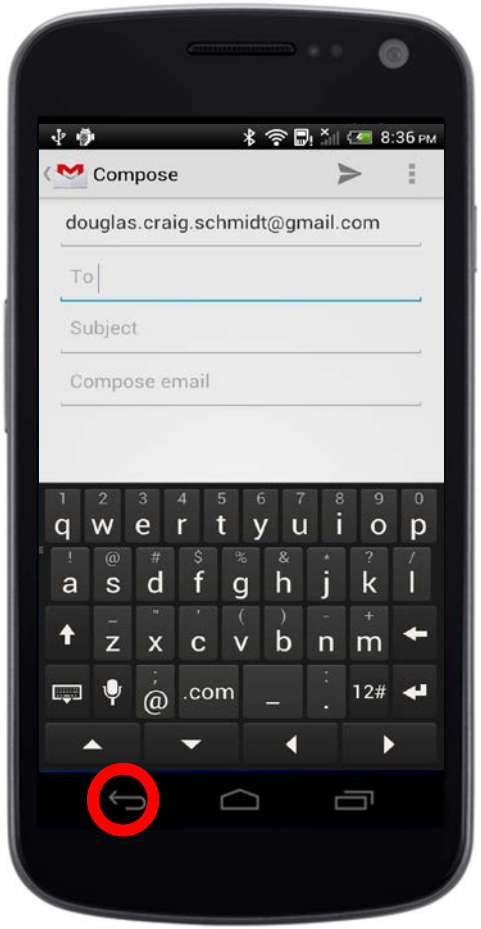
Task

- The task's Activity objects are stored on a "**back stack**" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack
  - Finishing an Activity pops it off the stack...
  - ... and returns to the previous Activity



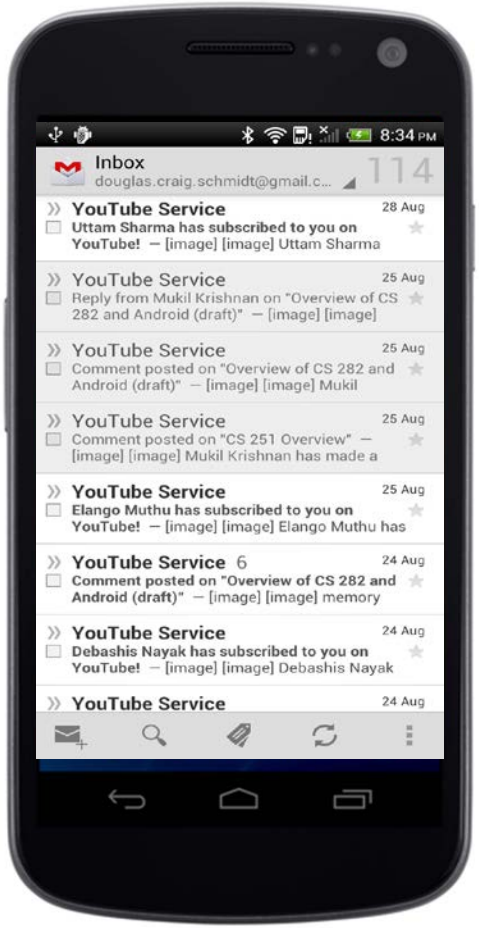
Task

- The task's Activity objects are stored on a "**back stack**" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack
  - Hitting the BACK button pops current activity off the stack...

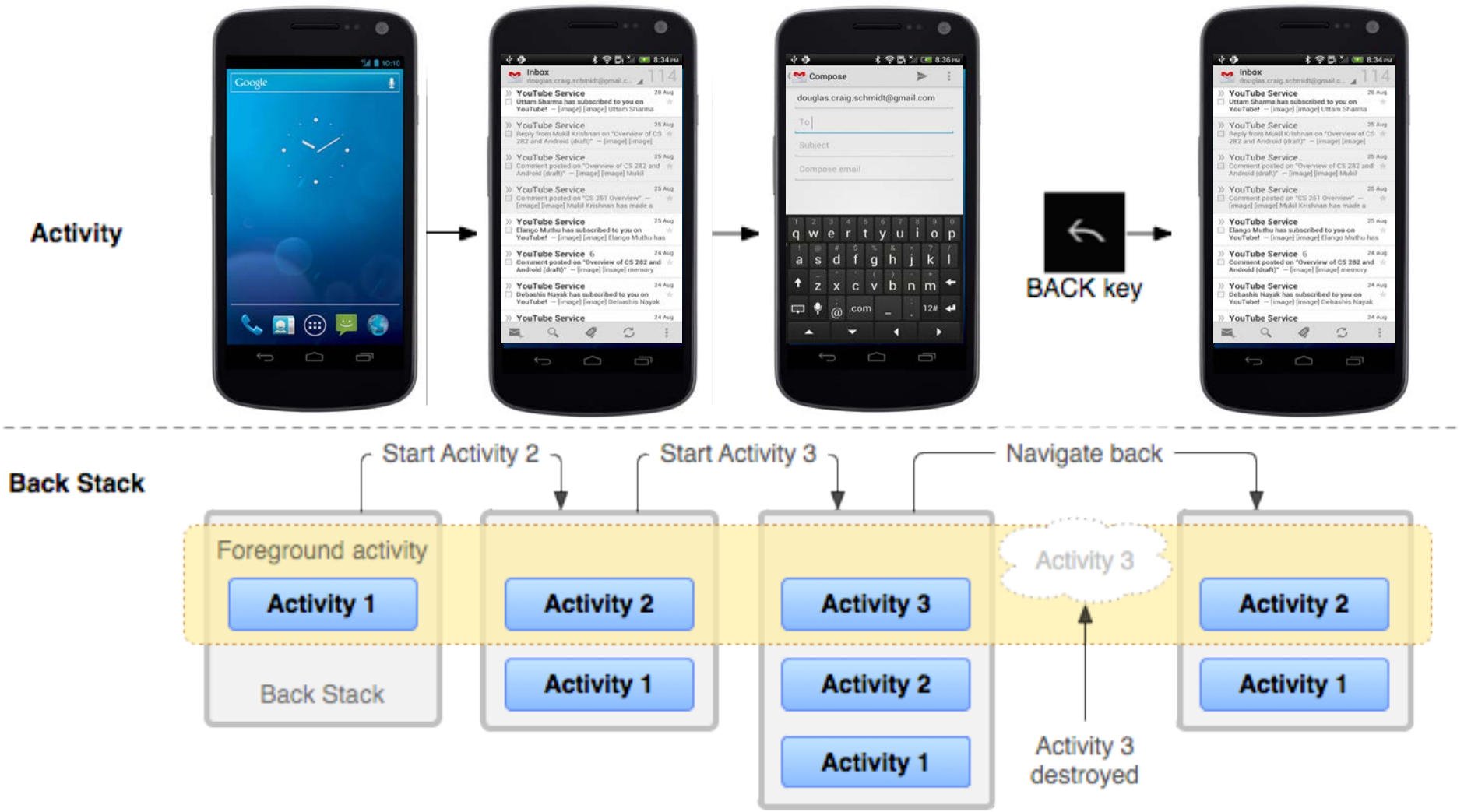


Task

- The task's Activity objects are stored on a "**back stack**" with the currently running Activity at the top
- At runtime
  - Launching an Activity places it on top of the stack
  - Hitting the BACK button pops current activity off the stack...
  - ... and returns to the previous Activity



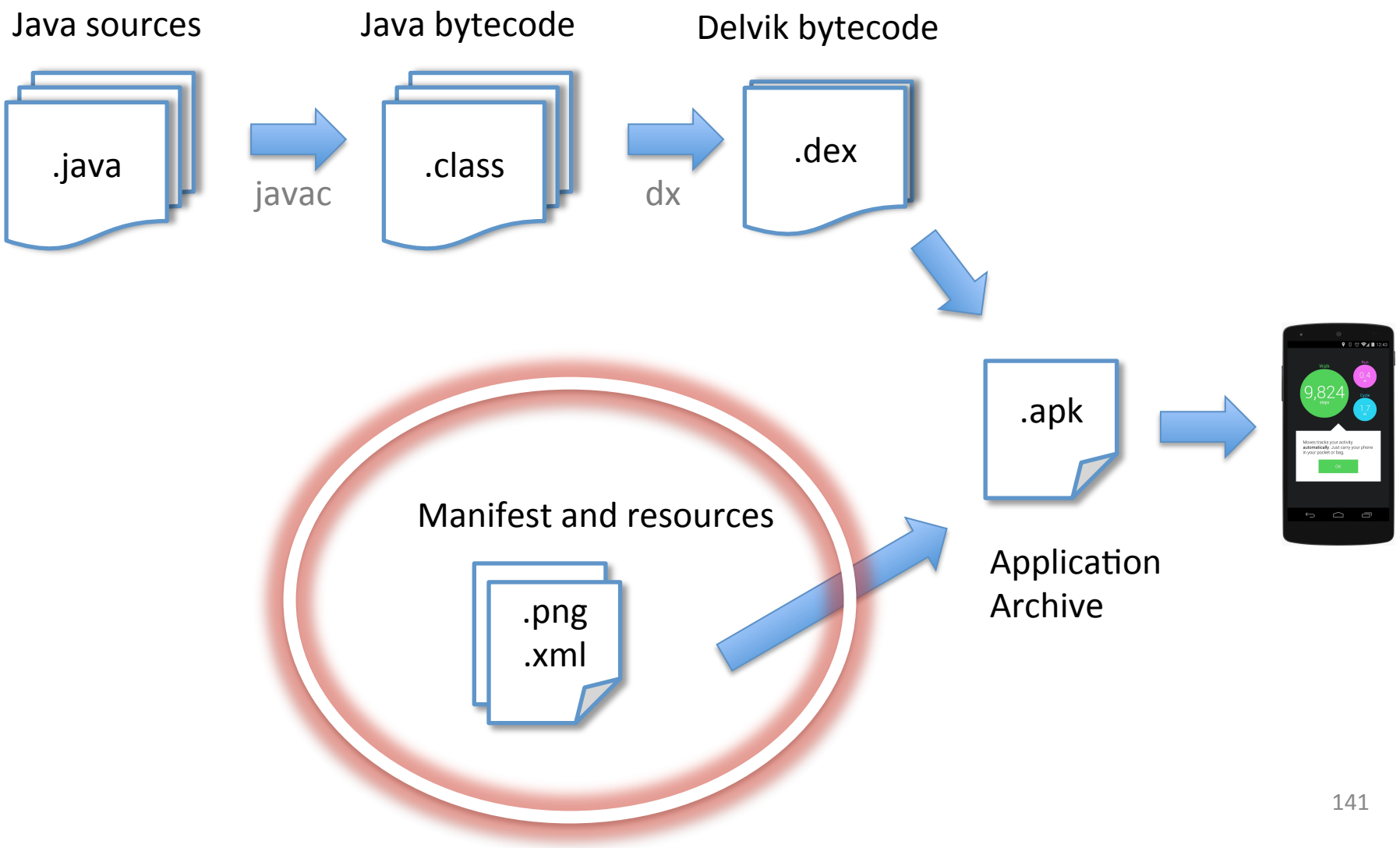
The Back-task



To resume – Android Applications

- 4 main components for android applications
  - Activity
    - The interface with the user, a "view" of the data
  - Service
    - No GUI and interaction, running in backgroud
  - Content Provider
    - Share and access application private data
  - Broadcast receivers
    - Listener for system-wide events and intents
- Intents are the messages used to propagate and accept requests
- Chains of activities from different applications form a Task

Applications



Application Manifest File

XML file containing the description of the application in terms of

- General Info (app name, version etc..)
- List of components (activities, services, Content providers…)
- Minimum API level required by the app (compatibility)
- The hardware required by the app (camera, wifi, bluetooth sensors…)
- The **user permissions** the app requires (Internet connection, access to SD card…)
- External libraries needed (Google API, Facebook API…)
- …

Application resources

- All the resources that are not code
  - Images, icons, audio, video…
- Menus, styles, GUI layout are define as XML files
  - Update many app features without coding
  - Optimize **for different configurations** of
    - Screen sizes
    - Device orientations
    - Languages and localizations
- Each resource is then accessible from the app code
  - See R.java file later…

To resume – Android Applications

- 4 main components for android applications
  - Activity
    - The interface with the user, a "view" of the data
  - Service
    - No GUI and interaction, running in backgroud
  - Content Provider
    - Share and access application private data
  - Broadcast receivers
    - Listener for system-wide events and intents
- Intents are the messages used to propagate and accept requests
- Chains of activities from different applications form a Task

To resume – Android Applications

- 4 main components for android applications
  - Activity
    - The interface with the user, a "view" of the data
  - Service
    - No GUI and interaction, running in backgroud
  - Content Provider
    - Share and access application private data
  - Broadcast receivers
    - Listener for system-wide events and intents
- Intents are the messages used to propagate and accept requests
- Chains of activities from different applications form a Task