Top 10 Tips For Developing Android Handsets

Linux World 2008

Wayne Lee

QUALCOMM CDMA Technologies

Agenda

- Intro
- Android Overview
- Top 10 Tips

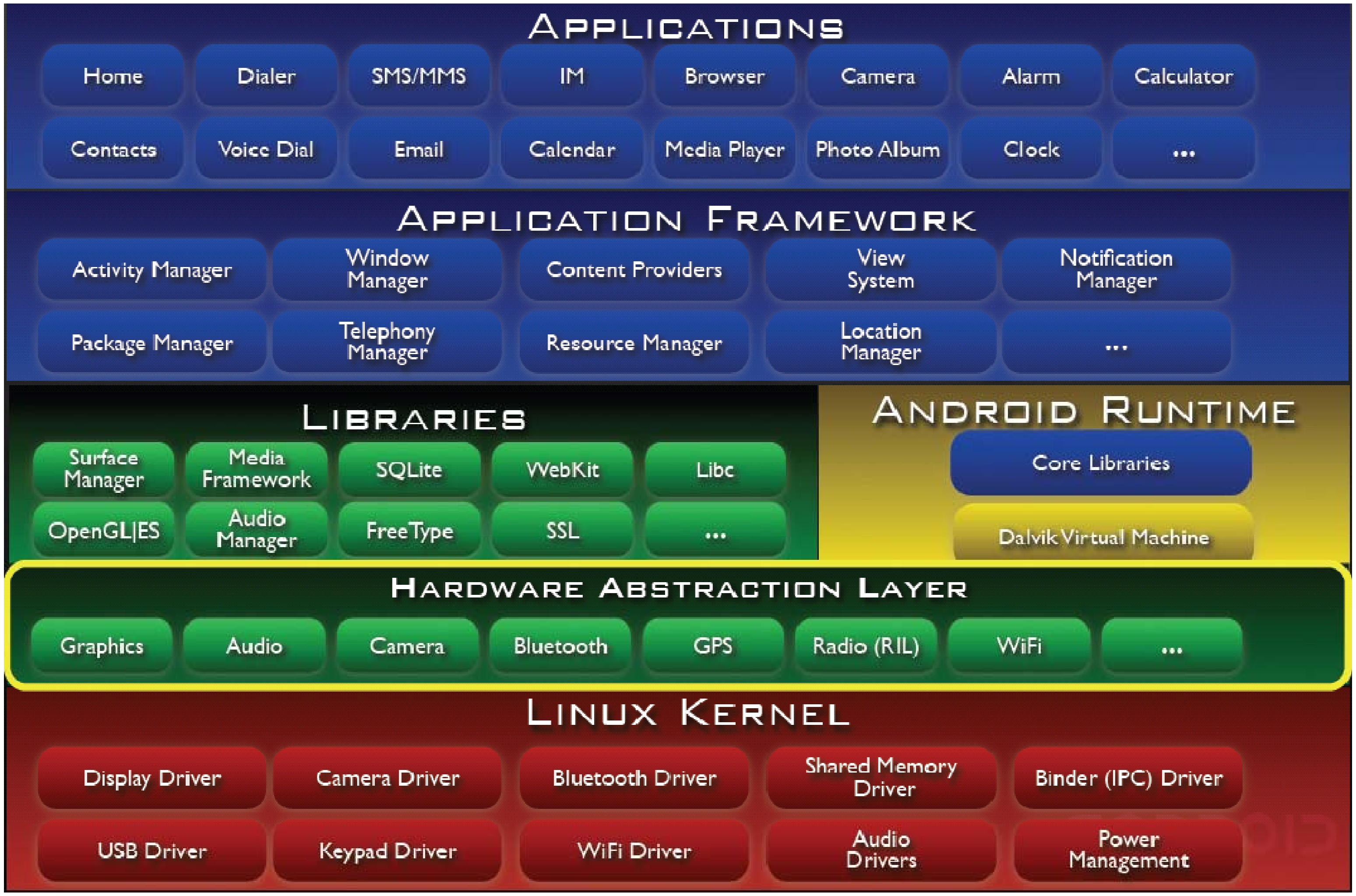
Qualcomm and Android

- Qualcomm is a founding member of the Open Handset Alliance (OHA)
- Qualcomm sells integrated chips that power cell phones:
  - 3G modem/GPS engine
  - Application processor (that runs Linux/Android)
  - Peripherals and multimedia acceleration cores
- Qualcomm is working with Google to integrate and optimize Android on select Qualcomm multiprocessor chipsets (MSM7k, QSD8K)

Scope of Talk

- Tips and lessons learned for handset vendors wishing to develop handsets that support Android
- Focus on kernel/drivers and Android HAL (Hardware Abstraction Layer)
- Will not cover Android in detail
  - Will only give very brief overview of Android
  - Will not cover upper layers of Android framework and app development
  - Google IO has lots of good presentations ☺

Android Architecture



Source: http://sites.google.com/site/io/anatomy--physiology-of-an-android

Android: Kernel and HAL

- Android uses Android/Linux, not GNU/Linux

- Kernel:
  - Linux kernel 2.6.25 + patches
  - Patches: ashmem, binder, power manager, low memory killer, kernel debugger, logger, usb function driver
  - Source: http://git.android.com

- Hardware Abstraction Library
  - Android interface to hardware
  - Allows for kernel drivers or proprietary user-space drivers
  - Supports drivers without standard APIs (e.g. vibrator)

Android: Display & Multimedia

- SurfaceFlinger
  - Server to compose 2D/3D surfaces from multiple apps onto the framebuffer
  - Can use 2D/3D HW accelerators to blit, rotate, scale, blend, etc.
- OpenGL ES library
  - Either software or hardware accelerated implementations can be used
- Media Framework based on PacketVideo OpenCORE
  - Supports audio, video, imaging codecs
  - Pluggable architecture for hardware/software codecs
    - Uses OpenMAX IL interface
- AudioFlinger
  - Server to mix audio channels from multiple apps to audio devices

Good Android Links

- http://code.google.com/android
- http://sites.google.com/site/io/anatomy--physiology-of-an-android
- http://sites.google.com/site/io/inside-the-android-application-framework
- http://sites.google.com/site/io/dalvik-vm-internals
- http://www.android-internals.org

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

Usual legal disclaimers from an engineer

- IANAL (I am not a lawyer)
- TINLA (This is not legal advice)

Spend Time to Understand Open Source Software (OSS) licenses

- Linux Kernel is licensed GPL
- Most of Android will be licensed as Apache v2

- GPL and Apache:
  - “ASL [Apache Software License] … is a permissive license that is conducive to commercial development and proprietary redistribution. Code that is distributed under the ASL and other permissive licenses can be integrated into closed-source proprietary products and redistributed under a broad variety of other terms. Unlike permissive open-source licenses, "copyleft" licenses (such as the GPL) generally impose restrictions on redistribution of code in order to ensure that modifications and derivatives are kept open and distributed under similar terms.”
  - Source: http://arstechnica.com/news.ars/post/20071106-why-google-chose-the-apache-software-license-over-gplv2.html

- OSS license can affect software design

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer
6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

Booting Linux/Android

1. Boot Linux from RAM
   - Start with Linux kernel with Android patches (android-2.6.25 on http://git.android.com)
   - Basic arch/board files (io, irq, timers)
   - Use JTAG to init RAM, create ATAGS, cmdline args, r0/r1/r2
   - Load kernel to RAM and run.

2. Add a simple root ramdisk filesystem and console
   - Add UART or DCC (JTAG on ARM) drivers for console
   - Use busybox for UNIX tools

Booting Linux/Android

3. Get critical drivers working (needed for basic Android booting)
   - USB (needed for ADB and flashing)
   - MTD (NAND) / filesystem

4. Boot from Flash
   - Need mechanism to flash (bootloader/USB/Ethernet or JTAG)
   - Update bootloader to load kernel/ramdisk from flash
   - Tip: Put the kernel/ramdisk in a normal partition, not in a filesystem (simplifies bootloader)
   - Tip: Boot from network (TFTP) (saves flash time)

Booting Linux/Android

5. Add basic UI drivers
   - Framebuffer
   - Keypad/Trackball

6. Build and boot Android
   - Build Android from source
   - Flash Android system image

7. Add remaining drivers
   - Modem, Data, GPS
   - Audio, Camera, Video, Graphics
   - WiFi, SD, Bluetooth
   - Touchpad, Sensors, Battery

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

ANDROID DEBUG BRIDGE

Get ADB (and USB) working early

- Android Debug Bridge (ADB) is essential for Android development/debugging
  - Requires USB

- ADB is used for:
  - Moving files to and from the device (adb push/pull/sync)
  - Logger (adb logcat)
    - Get logs on a per subsystem/verbose basis. Very, very useful
  - Running shell commands (mv, dmesg, reboot, ps, top, etc)
    - Tip: % adb shell <cmd>
  - Stop and restart services (adb shell start/stop)
  - TCP port forwarding (adb forward)
    - Needed for remote gdb debugging
  - Plus more!

- http://code.google.com/android/reference/adb.html

Fastboot and other tools

- Get bootloader working with the fastboot protocol
  - Requires USB + fastboot protocol
  - Allows for flashing of bootloader, boot (kernel/ramdisk), and system images
  - Could also be used for flashing other firmware (e.g. modem)

- More tools
  - Stack – dumps call stack of all processes from logcat crash output
  - Remote GDB to debug System Servers
  - Eclipse with Android plugin to debug Android apps

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

Drivers: Kernel or User-space

- Kernel
  - Must be GPL
  - Kernel manages multiple clients, security based on file permissions
  - Excellent Linux driver model
  - Leverages existing kernel drivers
  - E.g: BlueZ, SDIO

- User-space library (part of the HAL)
  - Can be proprietary (some exceptions)
  - Use UIO or lightweight driver to map hardware memory and proxy interrupts
  - May perform better (no need to trap into kernel, sometimes no copying)
  - More difficult to support multiple clients
  - Security is important: kernel must ensure user-space does not program HW to scribble over system memory
  - E.g. 3D graphics driver

Porting Drivers

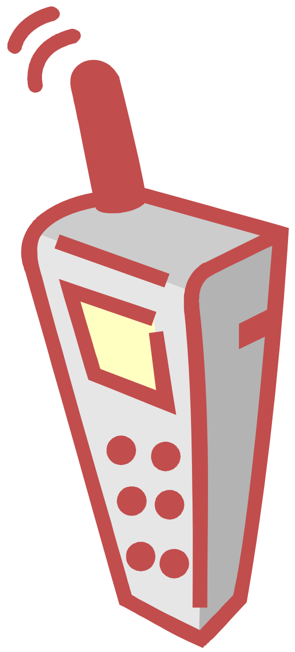
- Porting legacy drivers to kernel may not be best option
  - Fundamental different driver architectures:
    - Legacy RTOS drivers are task/event loop driven.
    - Linux drivers are blocking
  - Pure Linux driver are often easier to read/debug and better performing
  - May be tough (impossible?) to push upstream to kernel.org

- Porting layers of layers of software may not be best option
  - Results in slower performance, it adds up.
  - Consider the size of cache
  - Sometimes better to rewrite

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution



The RIL Shim Layer

- Radio Interface Layer (RIL)
  - HAL interface between the Android TelephonyManager and the baseband modem
  - Voice, Data, SMS, SIM, SIMToolkit
  - Android includes a reference AT command based RIL

- RIL API
  - Currently 75 requests (e.g. Dial) and 17 unsolicited responses (e.g. signal strength)
  - Android RIL inspired by GSM TS 27.007
  - Can implement as AT command based or proprietary protocol (e.g. RPCs via shared memory)

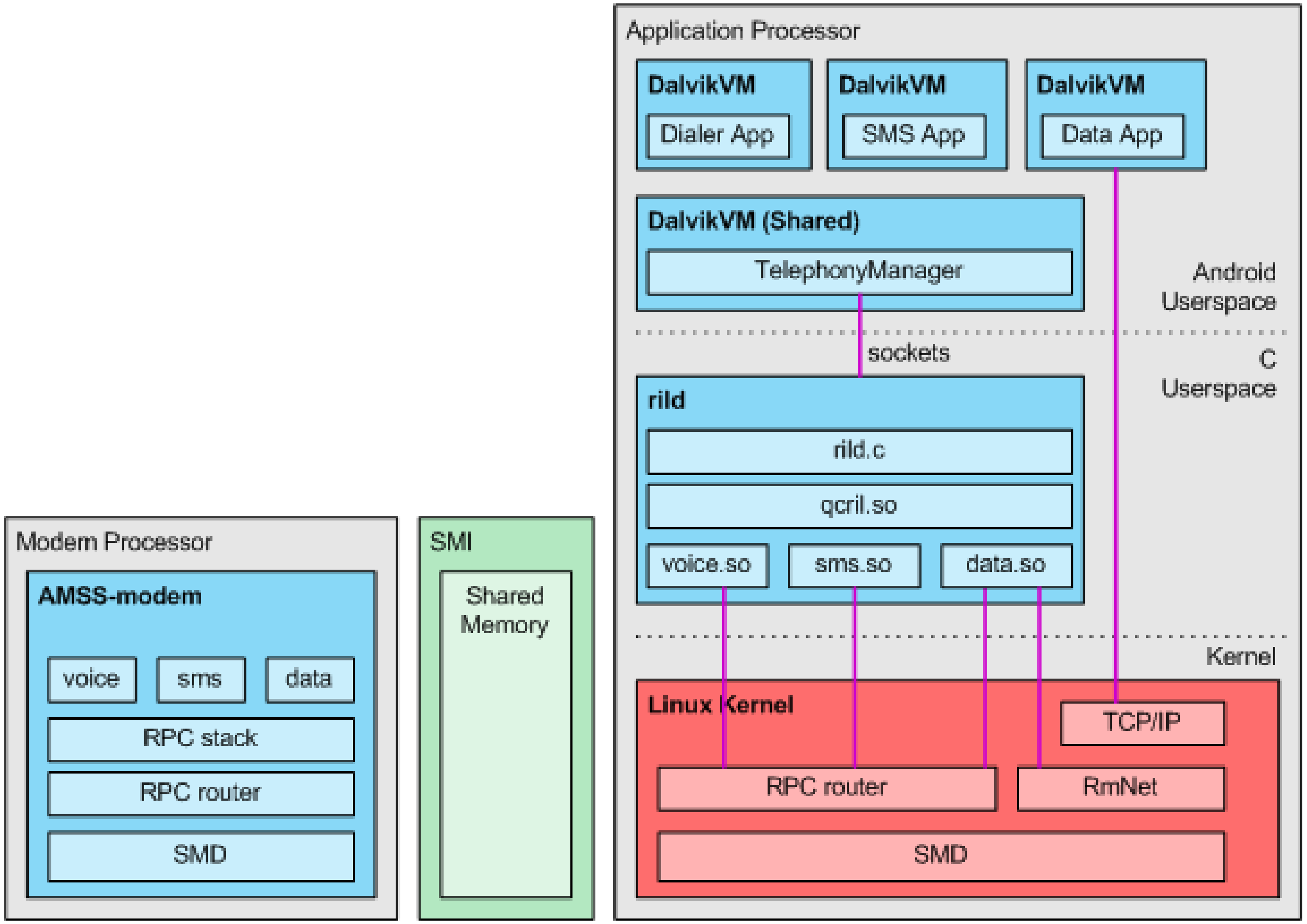
- Enhancing the RIL
  - Carriers may require specific features (e.g. JCDMA, O2 Homezone)
  - Qualcomm working on adding CDMA and Multi-mode support

The RIL Shim Layer

- Certification
  - GCF
  - CDG

- Tips
  - SIM and SIM Toolkit is complex, get an expert
  - RIL needs to provide neighboring cell id (non-standard). Used by Google Maps.
  - Data: suggest packet interface, no DHCP, no PPP.
  - Depending on modem interface, RIL may get complicated with lots of logic/state.
  - Android data is always-on, modem should support dormancy

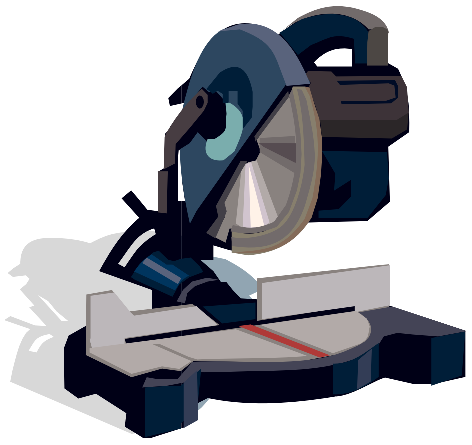
The RIL Shim Layer



Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. **Accelerate Multimedia in HW**
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution



Accelerating Multimedia in Hardware

- Software codecs requires lots of CPU
  - May not meet performance metrics (fps, bitrate, resolution, triangles/sec)
  - May consume too much power (CPU running, high clock rate)
  - Prevents other apps from running well.

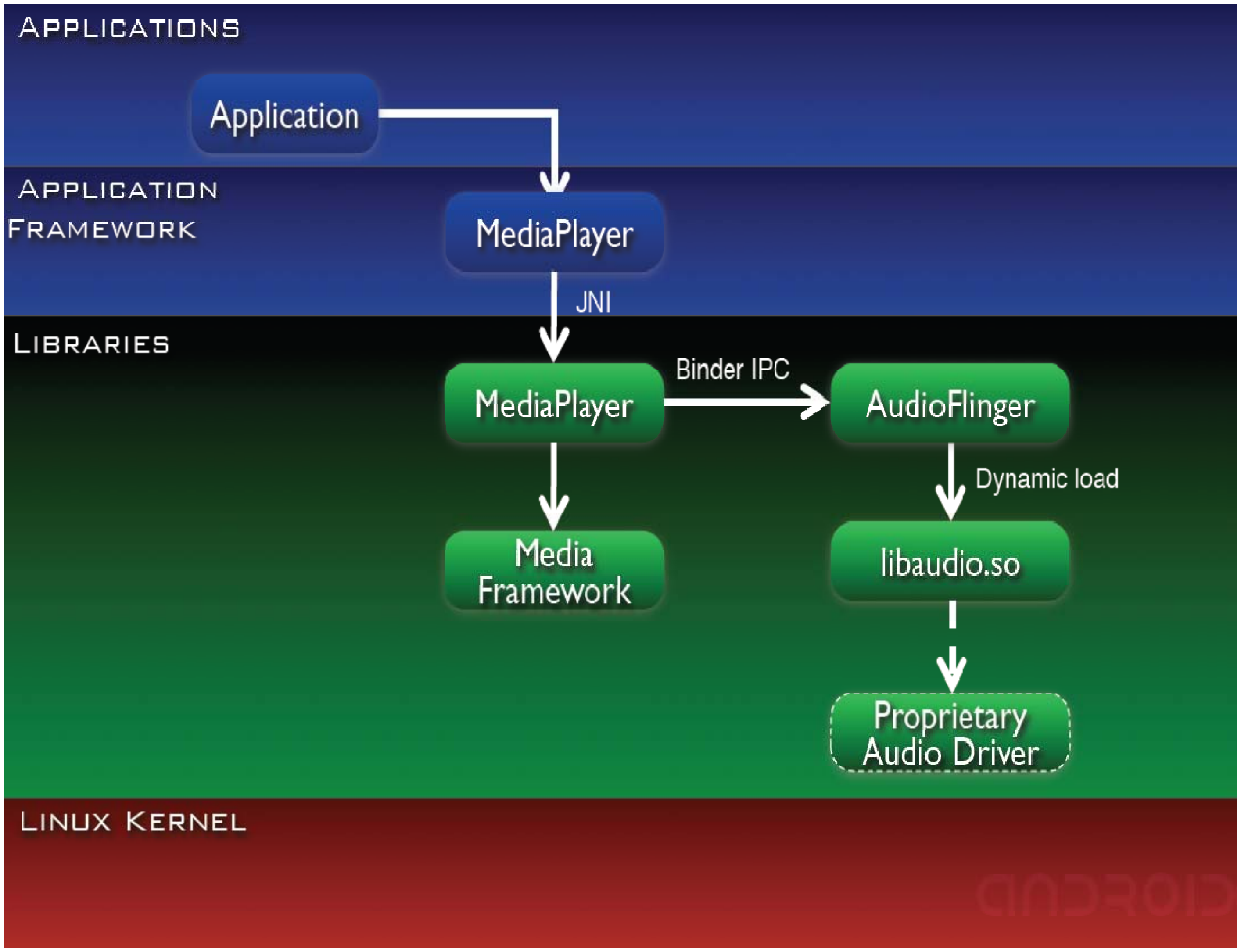
- H.264 decode example
  - Software codecs maxed CPU out at QVGA (320x240)
  - Hardware codecs able to achieve HVGA (480x360) at 500kbps, 30fps, using very little ARM11 CPU @ 528MHz for high level video processing

- Android easily supports acceleration:
  - Audio via the audio HAL
    - Can sit under OpenMAX IL or ALSA or OSS or native
  - Video and imaging via OpenMAX IL interface
  - 3D via OpenGL ES lib
  - Surface composition via HAL (2D or 3D)
    - Can accelerate blt, rotating, scaling, blending, etc

Accelerating Multimedia in Hardware

- Tips
  - Reduce memcpy
  - Excessive layers and threads increases context switches, blows cache, and reduces performance
    - Consider rewriting instead of pulling in large frameworks as is
  - Queue large buffers, then sleep

Accelerating Multimedia in Hardware

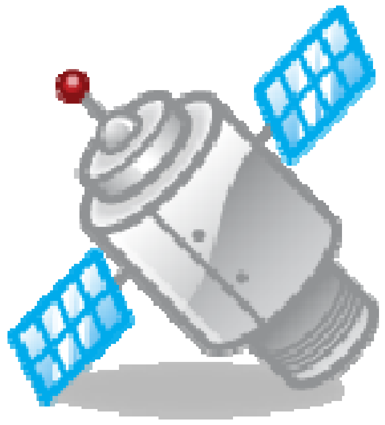


Source: http://sites.google.com/site/io/anatomy--physiology-of-an-android

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution



Add a good GPS solution

- Today's mobile devices/services demand location aware devices

- Location technologies
  - GPS (unassisted)
  - AGPS (network assisted)
    - Get sat info from network, compute on device/server
  - Other tricks
    - WiFi location
    - Cell tower triangulation
    - XTRA

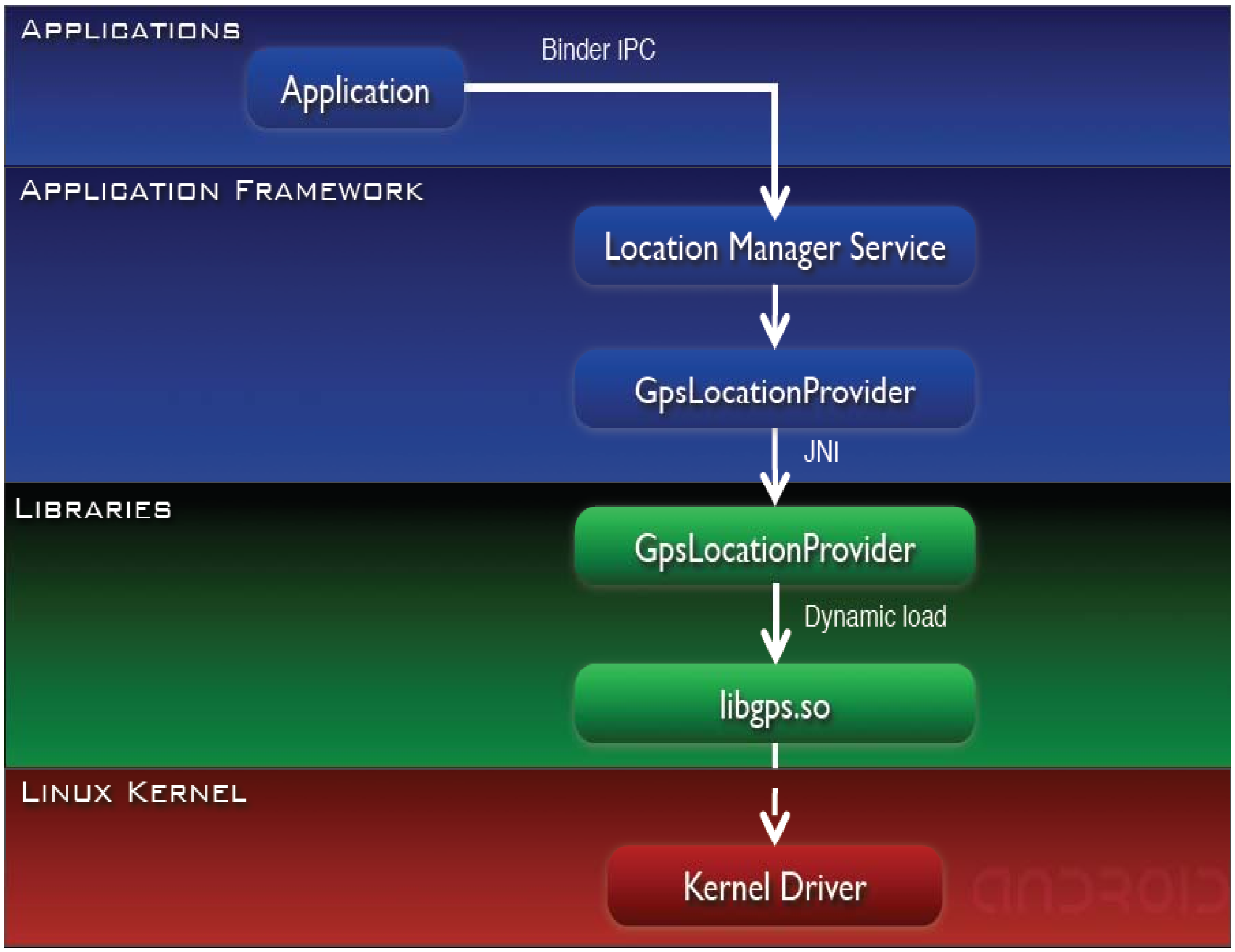
- Location metrics:
  - Time to first fix (cold, warm)
  - Accuracy (under different environments)

Add a good GPS solution

- Location Manager runs in System Server process
  - Links with GPS HAL lib
    - Can use standard NMEA, or
    - Interface specific to hardware (e.g. RPC)

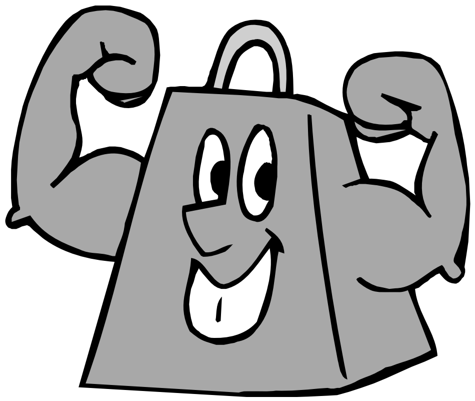
- Android may not support all GPS features
  - Will be open source for community to augment

Add a good GPS solution



Source: http://sites.google.com/site/io/anatomy--physiology-of-an-android

Top 10 Tips



1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

Maximize Performance, Minimize Power

- Performance and power are often at odds
  - Plan for this process to take some time
- Suspend/Idle
  - Android aggressively enter suspend.
    - Carefully use Android suspend locks (wake locks)
    - Can’t take too long to enter/exit suspend
  - Shut down as much as possible in idle
    - May incur additional interrupt latency (pm_qos?)
  - Synchronize wakeup events (BT, WiFi, 3G sleep, System)
- Clocks
  - Design frequency plan to reduce active oscillators/PLLs per “user-case” (standby, talk, audio, video)
  - Scale CPU/bus speed depending on load
    - Use cpufreq and optimize governor algorithm
- Devices
  - Place device and GPIO pads in low power mode when in suspend/idle and when disabled
  - Disable clocks and voltage regulators when in low power mode
  - Disable LCD/backlight when not in use (e.g. no user activity, audio playback)

Maximize Performance, Minimize Power

- Tips:
  - Use tickless kernel and deferred timers
  - Use lower power dedicated hardware (e.g. multimedia) instead of power hungry CPU
  - Shift to low power memory

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

Misc things to not forget about

- Factory provisioning
- Software upgrade
  - OTA
- Operator specific customizations
  - VFLive, Orange Homescreen, etc.
- Screen size customizations
  - Android 1.0 currently optimized for HVGA
  - May require adjustments to apps, fonts, icons for different screen size
- OEM customizations
  - Differentiate your Android solution from competitors
  - Themes, custom applications, etc.

Misc things to not forget about

- Support restarting of crashed modules/devices
  - RIL, modem
    - System must allow for modem restart
  - Servers/HAL (GPS, Audio, Video, 3D, Imaging, GPS, Bluetooth, WiFi)
    - System must allow for Server restarts

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

A vertically integrated and optimized Android solution?

- Porting Android to target platform and integrating HW acceleration takes time:
  - Kernel/Driver (BSP)
  - RIL with modem (with certification)
  - Integrated Multimedia with HW accelerators (audio, video, imaging, camera, 2D, 3D)
  - GPS, Bluetooth, WiFi
  - Maximize performance, minimize power
- Consider starting with pre-integrated, optimized components from OHA silicon vendor and software commercialization members.
- Allows OEMs to focus on differentiating Android features (apps, UIs, novel designs)

Top 10 Tips

1. Understand Open Source licenses
2. Booting Linux and Android
3. I want my ADB!
4. Drivers: kernel or user-space?
5. The RIL Shim Layer

6. Accelerate Multimedia in HW
7. Add a good GPS solution
8. Maximize performance, minimize power
9. Misc
10. An Integrated Android solution

QUALCOMM®
CDMA Technologies

Thank you!