Kernel hacking su Android

Better Embedded 2012

Better Embedded 2012 - Andrea Righi
andrea@betterlinux.com

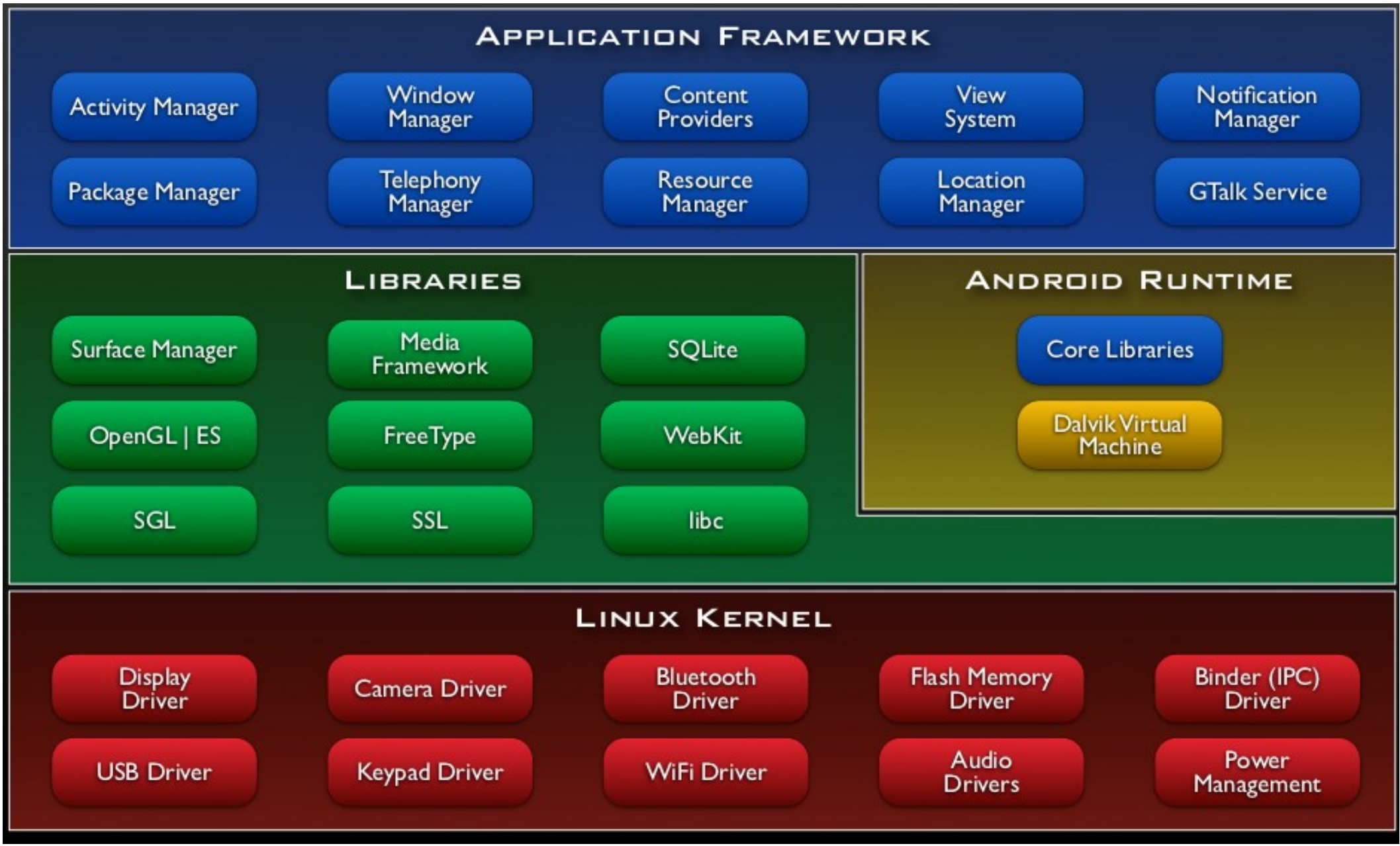
Agenda

- Overview
- Android Programming
- Android Power Management
- Q/A

Overview

What is Android OS?

- Linux kernel
- Android patches
- Bionic libc
- Dalvik VM (Java Virtual Machine)
- Application framework
- Apps


Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
GTalk Service
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Android: kernel modifications

- binder: IPC (i.e., communication between window manager and client)
- ashmem: shared memory (process cache)
- pmem: physically contiguous memory allocator
- logger: custom system logging facility
- wakelock: power management (hold machine awake on a per-event basis until wakelock is released)
- early suspend: suspend/resume hooks to be called when user-visible sleep state change
- lowmemorykiller (aka Viking Killer): custom OOM killer tunable from userspace (memory thresholds and oom_adjust thresholds)
- alarm timers: RTC-based wakeup
- timed gpio: driver that allows userspace programs to access and manipulate gpio pins and restore their state automatically after a specified timeout
- ram console: store kernel log to a persistent RAM area, so that after a kernel panic it can be viewed via /proc/last_kmsg
- USB gadget driver (ADB)
- YAFFS2 filesystem
- Lots of other small fixes and hacks...

Native libraries

- Bionic libc
  - Custom libc implementation, optimized for embedded use (optimized for size)
  - Don't support all the POSIX features
  - Not compatible with GNU Libc (glibc)
  - Native code must linked against bionc libc (or statically linked)

Android Programming

Better Embedded 2012 - Andrea Righi
andrea@betterlinux.com

Programming model

- Android applications are written in Java
  - Class libraries are similar to Java SE but not equal
  - Dalvik VM is a register-based architecture that dynamically translates from Java byte code into native architecture instructions
- However
  - It's still Linux, so we can build and run native Linux applications
  - Use the JNI bridge to bind C functions into the Java code

Android applications

- Applications are distributed as Android packages (.apk)
- Everything is needed to run an application is bundled inside the .apk
- Basically an “enhanced” ZIP file

Typical directory tree in Android

- / (initrd – ro)
  - /system (ro)
    - /system/bin
    - /system/etc
    - /system/lib
    - /system/usr
  - /data (applications data – rw)
  - /cache (applications cache – rw)
  - /mnt/sdcard (removable storage – rw)

Toolchain

- From http://developer.android.com
  - Android SDK
    http://developer.android.com/sdk/index.html
  - Android NDK
    http://developer.android.com/sdk/ndk/index.html
- From http://www.codesourcery.com
  - ARM toolchain to recompile the Linux kernel
    (get the one targeted at “EABI”)
    https://sourcery.mentor.com/sgpp/lite/arm/portal/subscription3053
  - ARM toolchain for native userspace applications
    (get the one targeted at "GNU/Linux")
    https://sourcery.mentor.com/sgpp/lite/arm/portal/subscription3057

Android emulator: goldfish

- The Android emulator runs a virtual CPU that Google calls Goldfish.
- Goldfish executes ARM926T instructions and has hooks for input and output -- such as reading key presses from or displaying video output in the emulator
- These interfaces are implemented in files specific tothe Goldfish emulator and will not be compiled into a kernel that runs on real devices.

Create an Android application

```
# create a new project
android create project  --package com.develer.hello --activity HelloAndroid \
            --target 2 --path ./helloandroid

# build in release mode
ant release

# sign a apk
jarsigner -verbose -keystore ~/certs/android.keystore \
            ./bin/HelloAndroid-unsigned.apk arighi

# align the apk
zipalign -v 4 ./bin/HelloAndroid-unsigned.apk ./bin/HelloAndroid.apk

# install
adb install ./bin/HelloAndroid.apk

# uninstall
adb uninstall HelloAndroid
```

Create a native ARM application

```
#include <stdio.h>

int main(int argc, char **argv)
{
	printf(“Hello, world!\n”);
	return 0;
}
```

Build & run:

```
$ arm-none-linux-gnueabi-gcc -static -o hello hello.c

$ adb push hello /data/local/tmp/hello

$ adb shell chmod 777 /data/local/tmp/hello

$ adb shell /data/local/tmp/hello
```

Run a custom kernel (Android emulator)

```
$ git clone https://android.googlesource.com/kernel/goldfish

$ git checkout android-goldfish-2.6.29

$ export PATH=/opt/toolchain/arm-eabi-2011.03-42/bin:$PATH

$ make ARCH=arm CROSS_COMPILE=arm-none-eabi- goldfish_defconfig

$ make ARCH=arm CROSS_COMPILE=arm-none-eabi-

$ emulator -kernel arch/arm/boot/zImage -avd <AVD_NAME>
```

Android Power Management

Better Embedded 2012 - Andrea Righi
andrea@betterlinux.com

Android typical use case

- Most of the time the device is in you pocket, completely in idle (suspend)
- You pull it ouf of your pocket maybe once, twice in a hour
  - Check the email, facebook, twitter, etc. for short “bursts” of time

Android Power Management

- Android will opportunistically enter a full system suspend state
- This forcibly stops processes from running
- Your battery last longer and your phone doesn't melt in your pocket

CPU State

- Running
- Idle
- Deep sleep
- Suspend
- Powered off

Android PM Features

- Early suspend
- Wakelocks
- Alarm Timer

Early suspend

- Suspend / resume kernel hooks called when user-visible sleep states change
- User-space framework writes the state to /sys/power/request_state

Early suspend

```
static struct early_suspend _powersave_early_suspend = {
        .suspend = powersave_early_suspend,
        .resume = powersave_late_resume,
};

register_early_suspend(&_powersave_early_suspend);

unregister_early_suspend(&_powersave_early_suspend);
```

Wake locks 1/2

- Used by applications to prevent the system from entering suspend or low-power state
- Driver API
  - struct wake_lock name
  - wake_lock_init()/wake_lock()/wake_unlock()
- Userspace API
  - Write lockname (and optionally lockname timeout) to /sys/power/wake_lock
  - Write lockname to /sys/power/wake_unlock

Wake locks 2/2

- Custom solution not integrated with the Linux Power Management (PM) subsystem
- Components make requests to keep the power on through “wake locks” (use wake locks carefully!!!)
- Support different types of wake locks:
  - FULL_WAKE_LOCK: cpu on, keyboard on, screen at full brightness
  - PARTIAL_WAKE_LOCK: cpu on
  - SCREEN_DIM_WAKE_LOCK: screen on (but may be dimmed), keyboard backlights allowed to go off
  - ...

Wake lock: Java interface

```
PowerManager pm = (PowerManager)mContext.getSystemService(
                                    Context.POWER_SERVICE);
PowerManager.WakeLock wl = pm.newWakeLock(
                                PowerManager.SCREEN_DIM_WAKE_LOCK |
                                PowerManager.ON_AFTER_RELEASE,
                                TAG);
wl.acquire();
 // ...
wl.release();
```

Android alarm

- Tell the kernel when it should wake-up
  - hrtimer: when the system is running
  - RTC: when the system is suspended

Android alarm

```
static struct alarm my_alarm;

alarm_init(&my_alarm, ANDROID_ALARM_RTC_WAKEUP, my_alarm_handler);

alarm_start_range(&my_alarm, expire_start, expire_end);

alarm_cancel(&my_alarm);
```

References

- Official Android developers website:
  - http://developer.android.com/
- Embedded Linux wiki (Android Portal):
  - http://elinux.org/Android_Portal
- The xda-developers forum:
  - http://www.xda-developers.com/
- mysuspend source code available at github.com:
  - https://github.com/arighi/mysuspend

Q/A

- You're very welcome!
- Twitter
  - @arighi
  - #bem2012