Headless Android Strikes Back!

ABS2014

04/29/2014

Gary Bisson
Embedded Software Engineer
• Embedded Software Engineer at Adeneo Embedded (Bellevue, WA)
  ▶ BSP Adaptation
  ▶ Driver Development
  ▶ System Integration
• Linux/Android enthusiast
SESSION OVERVIEW

1. Introduction

2. Headless Architecture

3. Headless Applications

4. Demonstration

5. Conclusion
Introduction
WHY ARE WE HERE?

• Android without UI?
• Use cases?
• Set the expectations of such system

Warning
Not about Embedded Linux vs. Headless Android...
WHAT'S THE INTEREST?

- Same OS/application across product line
- Standardized development environment
- Android API & tools:
  - SDK/NDK
  - ADB/Fastboot
  - systrace
WHAT'S DIFFERENT NOW?

- Update:
  - What has changed since first introduced?
  - Tips & tricks from past experience
- Come to the dark side of Android...
Headless Architecture
ANDROID ARCHITECTURE

Applications
- Home
- Contacts
- Phone
- Browser
- ...

Application Framework
- Activity Manager
- Window Manager
- Content Providers
- View System
- Notification Manager
- Package Manager
- Telephony Manager
- Resource Manager
- Location Manager
- XMPP Service

Libraries
- Surface Manager
- Media Framework
- SQLite
- OpenGL|ES
- FreeType
- WebKit
- SGL
- SSL
- Ilibc

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
The full-blown stack without:
  ▶ SurfaceFlinger
  ▶ WindowManager
  ▶ WallpaperService
  ▶ InputMethodManager
  ▶ SystemUI

Some tricks: fake values from SF Client
CYBORGSTACK SOLUTION

- Integration into source tree:
  - From Cyborgstack's GitHub:
    - headless branch
  - Change for generic-eng target
  - Directly into AOSP internals
GOING FURTHER

Some went further:

- Remove stock apps
  - Browser
  - HTMLViewer
  - ...

- Remove unnecessary *preloaded-classes*
  - View
  - Graphics
  - ...

- Remove few other System Services
SOME FIGURES

- Vanilla Gingerbread `generic-eng` build:
  - `system` size: 64MB
  - Free memory: 122/256MB

- Cyborgstack Headless build:
  - `system` size: 64MB
  - Free memory: 172/256MB

- Enhanced Headless build:
  - `system` size: 47M
  - Free memory: 202/256MB
As stated by Cyborgstack:

- Very much a proof of concept
- Not easily portable
  - Change of frameworks, system...
- Good starting point
- Gingerbread now getting old
AOSP INTEGRATION

- `ro.config.headless` property
- Alongside Jelly Bean 4.1 release
- Hasn't really evolved since though
NEW ARCHITECTURE

- **SurfaceControl**: tells user the device is Headless
- **WallpaperService**: not started
- **SystemUI**: not started
- **DisplayManager**: returns `HeadlessDisplayAdapter`
- **PhoneWindowManager**: skips action to user
- **ActivityManager**: skips Home app + activity creation
WHAT'S DIFFERENT

Advantages:

• Same tree for both headless and regular builds
• Easy to tweak:
  SystemProperties.get("ro.config.headless", "0")

Drawbacks:

• Not as thorough as it could be
• System Server crashes... needs modifications
Quick fixes:

- Patch `SurfaceControl` not to throw an exception
- Patch `SurfaceFlinger` not to start `bootanim`
- Remove `SystemUI` + some stock apps
- `config.disable_noncore`
- `config.disable_systemui`
GOING FURTHER

Same work needs to be done:

- Remove `WindowManager`
- Remove other UI-specific app/libraries
- Reduce preload libraries
- ...
SOME FIGURES

- Vanilla KitKat `aosp_arm-eng` build:
  - `system` size: 303MB
  - Free memory: 284/512MB

- Generic `armv7-a-neon` mini build:
  - `system` size: 128MB
  - Free memory: 356/512MB

- Generic "Headless" mini build:
  - `system` size: 128MB
  - Free memory: 356/512MB

- Optimized "Headless" mini build:
  - `system` size: 124MB
  - Free memory: 394/512MB
Headless Applications
NO ACTIVITY SO WHAT?

• App components:
  ▶ Service
  ▶ ContentProvider
  ▶ BroadcastReceiver

• Android Framework
Headless Android

Headless Applications

NO ACTIVITY SO WHAT?

Activities

Broadcast Receivers

Services

Content Providers

System Services
HOW TO?

- **AndroidManifest.xml** trick
- **am** commands
- **persistent** for System apps only
- **BOOT_COMPLETED** Intent otherwise

**System Services:**
- `onSensorChanged()`
- `onKeyDown()`
- Custom System Service!
By default, debugging only works for Activity-based application.

Need to start the application manually with `am`

Either attach manually or specify it in code: `android.os.Debug.waitForDebugger()`
USE CASES

- Barcode scanner
- Home automation remote
  - Button vs. Touchscreen
  - LED vs. Display
  - IP stays the same
Demonstration
HARDWARE SELECTION

- Android emulators
  - Gingerbread 2.3.7_r1
  - Kit Kat 4.4_r1
- Low-end ARM device:
  - Atmel sam9g20-ek (64M of RAM)
  - No graphics
Conclusion
CONCLUSION

- Good intentions
- Industry demand
- Activity limitation
- Source code: https://github.com/gibssson/headless-android
QUESTIONS?
REFERENCES

- Karim Yaghmour: Embedded Android
  [O'Reilly Shop](#)
- Opersys/Cyborgstack: Headless Android
  [ABS2012 - Headless Android](#)
  [Opersys Headless Blog Post](#)
- Headless Android Blogspot (Casey Anderson): Gingerbread Patches