Android on a non-mobile embedded system

a war story

Arnout Vandecappelle
Christophe Cap

arnout@mind.be
christophe.cap@niko.be

© 2013 Essensium N.V.
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License

http://www.mind.be/content/Presentation_Android-non-mobile.odp
Does this look like an Android device?
Does this look like an Android device?
Android on a non-mobile embedded system

a war story

Arnout Vandecappelle
Christophe Cap

© 2013 Essensium N.V.
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License

http://www.mind.be/content/Presentation_Android-non-mobile.odp
Android on a non-mobile embedded system

- **Benefits and limitations**
  - Reduces time-to-market
  - Is somewhat familiar
  - Is sometimes supported on your device
  - But future is uncertain

- **Lessons learned**
  - Porting to your device is relatively easy
  - Size is a problem
  - You're on your own for updates
Background

• We come from a Linux world
  – Earlier generation based on Linux/Qt
  – Several products with Linux, various “distros”
  – Kernel customization for custom board

• We do a lot of outsourcing
  – To get things done quickly
  – Source code is always transferred to Niko
  – Incoming inspection of source code
Android benefits and limitations
Android reduces time to market

- Smartphone/tablet app can be reused
- UI work can easily be outsourced
  - Many app developers out there
  - UI designers know Android
- App has shorter release cycle
  - Reuse testing of smartphone/tablet app
    Just add another device to the pool
  - Good test frameworks exist
  - App is a smaller part ⇒ less risk
Android is (somewhat) familiar

- It's still a Linux kernel – but with many adaptations
- It still uses many traditional libs and daemons – but with many adaptations
- Source is based on `make` and `git` – but used in an unfamiliar way (lunch, repo)
- It doesn't follow Filesystem Hierarchy Standard
- It doesn't use glibc, but bionic – not POSIX!
Android is (somewhat) familiar

- **System Apps**
  - Device: /system/app/
  - Src: packages/

- **User Apps**
  - Device: /data/app/
  - Src: device/marakana/alpha/app/

- **Android Framework Libraries**
  - Device: /system/framework/ (android.*)
  - Src: frameworks/base/core/

- **Java Libraries**
  - Device: /system/framework/
  - Src: libcore/
  - (java.* and javax.*)

- **System Services**
  - Device: /system/app/
  - Src: frameworks/base/cmds/system_server
  frameworks/base/core/

- **Dalvik Runtime**
  - Device: /system/bin/dalvikvm and /system/bin/app_process
  - Src: dalvik/ and frameworks/base/core/

- **JNI**
  - **Init/Toolbox**
    - Device: /init
    - /system/bin
    - Src: system/core/
  - **Native Daemons**
    - Device: /system/bin
    - Src: system/external/frameworks/base/cmds/
  - **Native Libs**
    - Device: /system/lib/
    - Src: bionic/external/frameworks/base
  - **HAL**
    - Device: /system/lib/hw
    - Src: hardware/

- **Linux Kernel**
  - Not part of Android source (AOSP)
Custom programs can be reused if...

- You can port them to bionic or you install glibc on the system
- You can adapt paths to /system/* or you chroot into an FHS tree
- You can package it in lunch/repo (yet another build/package system)
- Hopefully you don't need to run the same code on stock Linux...
Many chip and board vendors provide Android

- Branches off AOSP (https://android.googlesource.com/platform)
- AOSP itself contains some boards
  Actually only one at the moment: Pandaboard
- SoC vendor forks
  TI OMAP, Freescale i.MX, …
- Board vendor forks
  e.g. Phytec, Variscite, Inforce, Adeneo, …
- Community-driven forks
  Android-x86, rowboat (Gumstix), …
Is Android future-proof? Kernel lags behind

• Freescale i.MX Android kernel:
  – is at 3.0.15, not 3.0.99
  – > 2000 patches between v3.0.15 and imx_3.0.15-android

• TI DM37x Android kernel:
  – TI provides 2.6.32-based kernel
  – android.googlesource.com is at 3.0.58, not 3.0.99
  – > 3000 patches between v3.0.58 and android-omap-3.0
Is Android future-proof? Board vendors don't provide updates

• Board vendors typically provide only a few versions e.g. Gingerbread, Jellybean

• Usually no more updates after move to new Android version
  – Theoretically not much of a problem since older apps mostly run on newer Android
  – However, porting customizations can be a lot of effort

⇒ you're on your own if you want long-term support
Is Android future proof? Community forks vary

- Tracking AOSP is not trivial!
- E.g. android-x86: only 4.0-r1 considered stable
  - Not yet supported by AMD
  - Graphics and codec acceleration may not be available
Lessons learned
Adding BSP is easy

• Just edit arch/arm/mach-xxx/board-yyy as usual; same for display

• Exporting to middleware (app space) is more difficult
  – especially sensors, HW acceleration for multimedia

• See e.g.
  – Porting Android presentation at ELC-E 2012
    http://elinux.org/images/f/ff/Porting_Android_4.0_to_a_Custom_Board.pdf
  – Embedded Android training
    http://www.opersys.com/training/embedded-android

• But all this is typically already provided by the vendor
Basic customizations are very easy

- Custom bootsplash is foreseen in *bootable*
- Define required services in *device_name.mk*
- Replace *Launcher* with custom app
- Remove SystemUI, lock screen, unneeded apps and services
  But it's easy to break dependencies, so test carefully
- Customizing standard apps is more work
The Ethernet problem

See Adding Ethernet Connectivity at ELC-E 2012

- Ethernet works fine on the Linux side
- But it's not visible in Connectivity Manager so apps don't detect that it's available
- Android-x86 has Ethernet Connectivity Manager but it doesn't work perfectly
Size matters

- AOSP is provided as git trees, not tarballs
- Full clone takes about a day
- Impossible to integrate in our svn-based workflow
- Build from scratch takes too long for a nightly build
- In the end, we shifted this responsibility to the board vendor
App store doesn't come for free

- AOSP doesn't contain Google Play or other Google products
  Agreement with Google is required
- So you cannot use Google Play to
  - push updates
  - sell features by installing apps
- Google Play server-side isn't open source
  But FDroid (from Replicant) is open source
  git://gitorious.org/f-droid/fdroidserver.git
- Relatively easy for an app to update itself
Update daemon app

- Service with permission to install packages: `android.permission.INSTALL_PACKAGES`
- Polls your company URL for presence of new `.apk`
- Downloads `.apk`
- Installs it with local package manager: `pm install -r foo.apk`
OS updates: roll your own

- AOSP has *bootable*, but chip vendors usually ship U-Boot
- Update is normally initiated by the user
  FailSafe remote update is not foreseen at all
- On the other hand, it's U-Boot + Linux userspace,
  so not so different from other embedded systems
Conclusion:
Android can save a lot of time

- If you can reuse a smartphone app (or vice versa)
- If it is already ported to your platform but doesn't cost more time than porting Linux
- If you don't need standard Linux tools/daemons
- Once you have the platform you can quickly respond to market and functional changes.