Android-IA Scalability Features To Support A Single Build Target

Andrew Boie
Jianxun “Chang” Zhang
Daniel Leung
Charlie Johnson
Matt Gumbel
Andy Ross

Agenda

- 01.org – Who We Are
- Problem statement
- Automatic module loading with ueventd
- Flexible installer
- Ethernet connectivity
- Scalable HALs
- EFI & secure boot

Android-IA on 01.org - Who We Are

- https://01.org/android-ia/
- Part of the larger 01.org website maintained by Intel Open Source Technology Center
- Independent Android community site dedicated to driving Android support and innovation on Intel Architecture
- Binaries and source code available
  - Code for all the topics covered today is/will be available online
- Ultimate objective of most of our IA enabling and innovation is upstream inclusion in Android Open Source Project (AOSP)
- Join our mailing list!

Single Build Target

Objective

- Run Android at some baseline level of functionality on multiple devices with a single binary installation image
- Ongoing process – every bit helps even if we can't do it all

Advantages

- Reduce the number of hard-coded parameters in the Android board configuration files
- Support many off-the-shelf devices, including ones we don't know about
- Reduce bring-up time on new platforms
- Target a class of devices instead of a specific device
- Lowers expertise required to bring up Android on a system

Scalability May Not Be For Everyone

- A single image may make it more difficult to optimize for a specific device without breaking something else
- Requires testing on all devices with any change
  - As opposed to just the specific device being targeted

Three Classes of Parameters

- Build-time configuration
  - Much of Android config is currently done here
  - Image is highly tuned to specific destination hardware
- Install-time configuration
  - Decisions made when software is installed
  - Permanent
    - Stored outside scope of software updates
    - Immutable
    - /factory/factory.prop
  - Scope limited to properties that are not auto-detectable or runtime immutable
    - Camera physical orientation
    - Graphics driver
    - LCD density (EDID-based config in future)
    - Disk partition layout
  - For auto-detectable properties
    - Run detection logic in the installer
    - Otherwise just interactively query the user
- Runtime configuration
  - Automatically detected or runtime mutable parameters
  - Manual selection, i.e. Settings app
  - Android PackageManager imposes some constraints on what is mutable

Automatic Kernel Module Loading

- Modprobe-like library functions
  - insmod_by_dep() and rmmod_by_dep() added to libcutils
  - Traverse modules.dep dependency hierarchy to insert all needed dependencies
  - System-wide and local blacklists can be used to skip loading particular modules
  - rmmod_by_dep() won't remove a dependency if used by something else
  - Uses modules.alias to map uevent modalias to the module name
- Enhance ueventd
  - Many uevents may come in before /system is mounted, queue them
  - Deferred processing until /system is available
    - Checks every time there is an 'add' event
- Additional init.rc commands
  - coldboot – trigger ueventd deferred module loading by triggering 'add' events in sysfs
  - probemod – improved 'insmod'; inserts required dependencies
- /sbin/modprobe
  - Drivers in kernel can request modules by launching a program
  - Default to /sbin/modprobe; thin wrapper around insmod_by_dep()
  - Not actually kernel.org GPL Modprobe

Automatic Module Loading (cont.)

- Loading appropriate WiFi Drivers
- Audio codecs
- USB peripherals
- Camera Hardware, uvcvideo
- Not everything can be auto-inserted yet
  - Currently building-in USB Ethernet and USB Serial drivers for alternate ramdisk targets
    - No /system available in Recovery Console
  - Sensor Hub drivers currently don’t probe available hardware
  - Modules that require parameters must be inserted via init.rc
    - No modules.conf (yet)
- You need security too
  - MODSIGN in Linux 3.7 – more on this later
- Plan is to upstream to AOSP soon
- https://01.org/android-ia/blogs/jzhang80/2012/increasing-android-device-scalability-automatic-kernel-module-loading

Flexible Disk Installer “Iago”

- Not really applicable to handset/low-end tablet products
- Replaces old bootable/diskinstaller
  - Buggy, not flexible, MBR with GRUB only
- Use-cases
  - Install on Android on commodity hardware
    - Including devices not previously known
    - Intended for devices that boot from removable media
  - Dual/Multi boot with other Oses
  - Three boot modes
    - Automatic installer
      - Uses predefined configuration
    - Interactive installer
      - Installation questions to customize to user’s needs
    - Live Android session directly from the USB stick

Flexible Disk Installer “Iago”

- Design goals:
  - Lightweight integration into Android tree
    - Pulls in parted, ntfs-3g, efibootmgr
    - Parted eventually going away in favor of custom GPT library
  - Support for platform-specific plug-ins similar to Recovery/OTA system
  - Interactive disk partitioning
  - Dual/Multi Boot support
  - GPT/UEFI support (Legacy BIOS/MBR support dropped)

Flexible Disk Installer “Iago”

- Query user for configuration parameters
  - Install-time configuration parameters established here
  - Auto-detectable but immutable props have detection logic run in installer environment
  - Selections written to /factory/factory.prop, never touched by OTA or Factory Data Reset
- Eventual support for Multi-Boot
  - Currently support dual boot with Windows 8
  - Ubuntu, Fedora, Tizen, multiple Android installs
- Eventual support for a GUI
  - Installation media boots into Live Android image
  - Installer frontend a special app that only exists in Live image

SMBIOS Properties

- Special case of install-time parameters for known devices
- System Management BIOS (SMBIOS) specification
- Microsoft requires OEMs to support this for certification, all Intel devices that can run Windows should have it
- DMI sysfs
  - /sys/device/virtual/dmi/id
  - Unique modalias per device
- Search for substrings in modalias for manufacturer and model information
- /system/etc/dmi-machine.conf
  - Individual system property files in /system/etc/machine-props/
  - Parameters must be known a priori, but can be updated OTA
- Devices that aren't supported instead configured by Installer questions

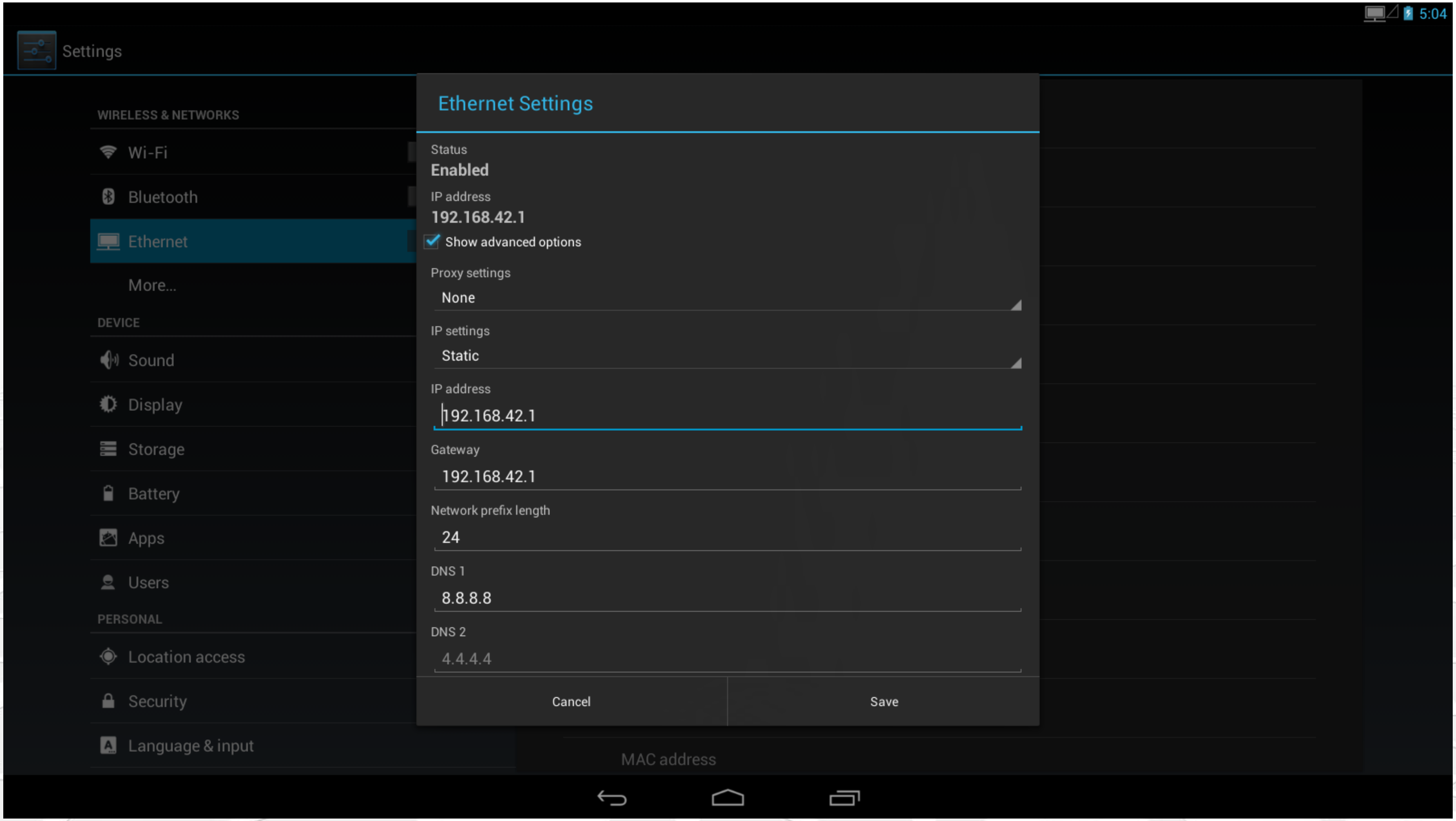
Disk Layout Scalability

- Disk information hardcoded in lots of places
  - recovery.fstab, vold.conf, init.rc or mountall fstab, OTA scripts, others…
- Establish /dev/block/by-name symlinks so files are static
  - As opposed to /dev/block/sda5 (example)
  - /dev/block/by-name/system
- Iago installer places partition names in GPT entries
  - Prefixed with randomly generated "install id"
  - Prevents issues with multiple Android installations on same device (Live image)
  - Modification to ueventd to create symlinks based on names passed in via block device uevents
- Many shipping Android devices do something similar
  - Partition name stored in the GPT
  - Include hard-coded controller name in path for security reasons
  - parse_platform_block_device() in ueventd
  - Otherwise, possible to spoof partitions using specially crafted GPT in removable media
- Advantages
  - Hardcoded files in build written once and never touched again
  - Physical disk configuration completely flexible, even span multiple disks
    - No Installer support yet, but could conceivably support things like LVM, SW RAID, etc.
  - Can install Android on removable media
  - But if security (user is enemy) is a concern don't do this!

Ethernet Connectivity

- Desirable for a few reasons
  - Devices without WiFi
  - ADB/GDB over Ethernet for devices without USB OTG
  - Performance throughput
- Configuration
  - Extended the Android Settings app
    - DHCP or Static IP configuration
    - Proxies
  - Status bar icon similar to WiFi
- Integrated with Android ConnectivityManager
  - Switches lower priority networks off when higher priority connections are available
  - EthernetManager not exposed directly to apps
    - Apps just see it as a generic network connection like WiFi or 3G
- Utility configuration
  - Use Ethernet as secondary network interface for debug
  - Allows Ethernet connectivity in alternate ramdisks
  - Also during bringup when UI isn't yet working
- https://01.org/android-ia/blogs/mkgumbel/2013/ethernet-support-android-ia

Ethernet Screenshot



Ethernet Settings

Status
Enabled

IP address
192.168.42.1

☑ Show advanced options

Proxy settings: None

IP settings: Static

IP address: 192.168.42.1

Gateway: 192.168.42.1

Network prefix length: 24

DNS 1: 8.8.8.8

DNS 2: 4.4.4.4

Device Triggers

- Sometimes need for more complicated processing on device insertion
- Ueventd only has limited functionality
  - Creation of device nodes
  - Permissions on device nodes based on ueventd.rc
  - Automatic insertion of modules and their dependencies based on modalias/modules.dep
- Extend ueventd.rc syntax to allow wildcards within the path (not just at end)
- Extend init.rc syntax
  - Perform additional actions when a device is added or removed
  - Example: bring up network interface when USB Ethernet adapter is connected
- Working with Google on acceptable upstream implementation
  - https://android-review.googlesource.com/#/c/40143/

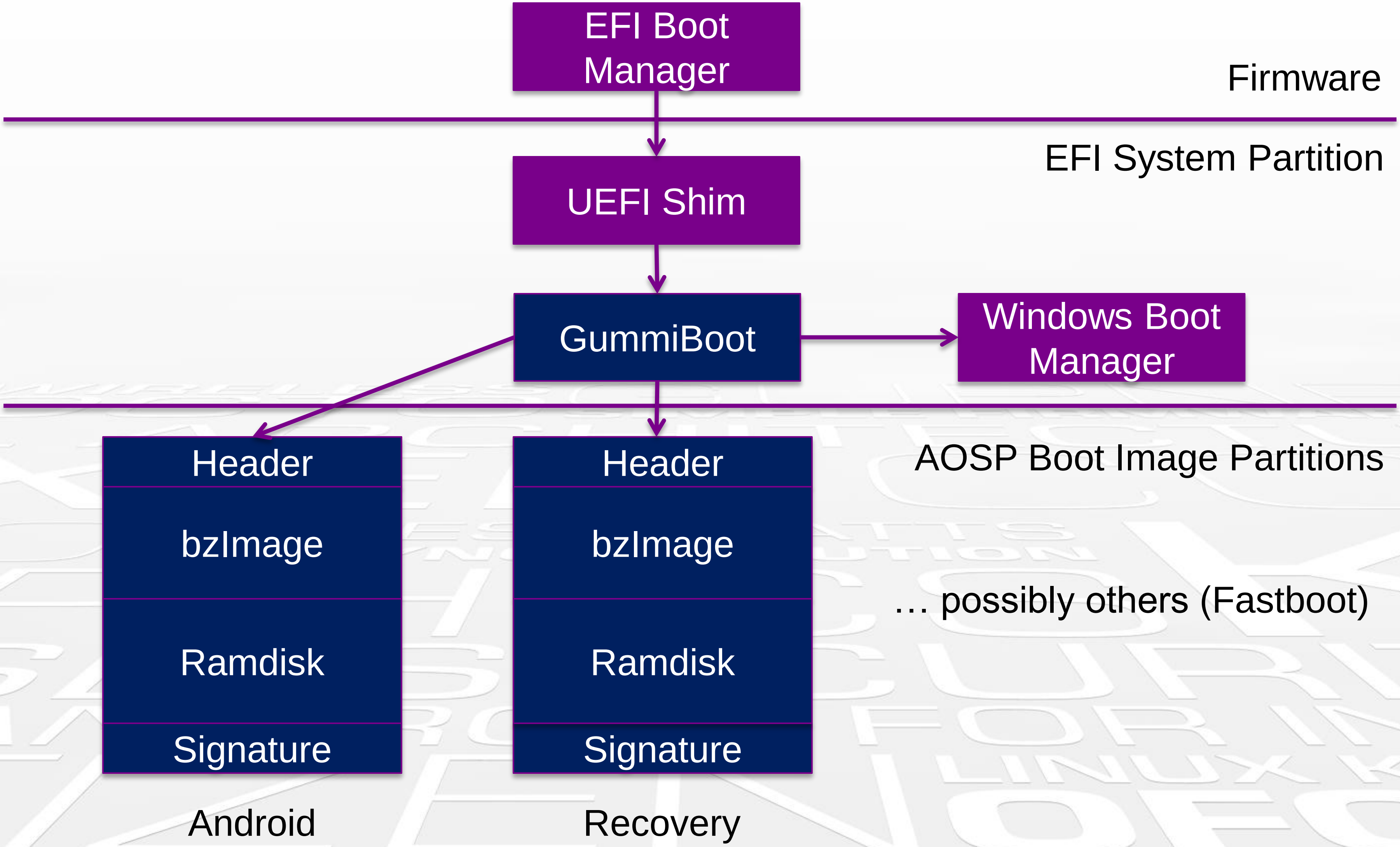
Scalable HALs

- Audio HAL
  - Extension of Nexus 7 Audio HAL
  - At boot time, probe attached audio codecs
  - Configure mixer controls appropriately
    - No standard set of names for mixer controls
    - Set of XML files for each codec vendor family
    - So far Realtek & Cirrus Logic (most common)
    - Can add new ones without modifying HAL code
- Sensors
  - Check for industry-standard IIO Sensor Hub at first boot
  - Slightly time-consuming, cache the result for later boots
  - We expect most Win-8 class slates to have this hub
- Camera HAL
  - Support various USB cameras using Video4Linux interfaces
  - Physical layout specified in ro.camera.* properties

UEFI Secure Boot

- Single secure boot solution for UEFI platforms
  - Some elements here still WIP and not on 01.org
- Need to trust bootloader stages, kernel, ramdisk, modules, and all inputs
- Linux kernel modules on /system signed with new modsign feature in Linux 3.7
  - Use static key checked into the build instead of throwaway key
    - Reduced size of OTA incremental images
    - Out-of-tree modules can be delivered as binaries
    - Fastboot won’t have to flash both /system and kernel
    - Development team doesn’t have any access to production key
  - All keys in repo are development test keys
  - sign_target_files_apks extended to additionally re-sign modules and replace public key in kernel with production key

UEFI Secure Boot Diagram



UEFI Shim

- Modified Red Hat UEFI Shim
  - Signed with the key stored in firmware, typically Microsoft key
  - Contains its own signature and key management logic from OpenSSL
  - Verifies next stage image is signed
  - Exports security services for use by later EFI stages
  - Loads next UEFI stage using PE/COFF link-loading to bypass FW security policy
  - Open source version has key onloading for adding own keys
  - Modification is to verify arbitrary blobs PE/COFF executables

GummiBoot

- Modified Gummiboot
  - Signed with key in UEFI Shim (not FW key!)
  - Supports loading standard Android boot image format
    - system/core/mkbootimg
  - AOSP boot image format slightly extended to include optional signature
  - Uses UEFI Shim security services to verify boot image and config files
  - Starts kernel directly using some efilinux code
  - Alternate boot target support
    - Interactive menu for eng builds
    - Check for ‘magic’ keys to load alternate targets like Recovery Console
    - Android Bootloader Control Block support for recovery console persistence
      - Re-launch Recovery Console with same parameters if power interrupted
    - LoaderEntryOneShot EFI variable set by kernel driver
      - For “adb reboot recovery”
    - Windows Boot Manager for Dual Boot installations

Userspace Fastboot

- Traditionally, Fastboot implemented in bootloader
  - Reference implementation in LK Bootloader
  - Need to re-implement with every bootloader change
- Implemented as a tertiary boot target
  - Additional boot image with special ramdisk
  - Similar to Recovery Console
- Plug-in architecture
  - Similar to Recovery Console plug-ins
  - Add platform-specific flashing commands
    - Update device firmware, baseband, BIOS, etc
- Uses recovery.fstab to map device nodes
- Full Android userspace is nice
  - Shell commands, libz, available
  - On-the-fly gzip decompression
  - Ethernet connectivity
- However, with migration to UEFI, plan is to re-implement as UEFI application which can be baked into firmware
  - Google likes this better because it will be always available

Future Work

- Framework overlay scalability
  - config.xml, overlays, etc.
  - Cyanogenmod has some work in this space
- Fastboot as EFI application
- More Installer plug-ins
- Integration of Sony DASH Dynamic Sensor HAL
  - https://github.com/sonyxperiadev/DASH
- Multiple graphics driver support
  - Multiple hwcomposer, gralloc, EGL driver libs
- Install-time App specification
- We’re hiring!