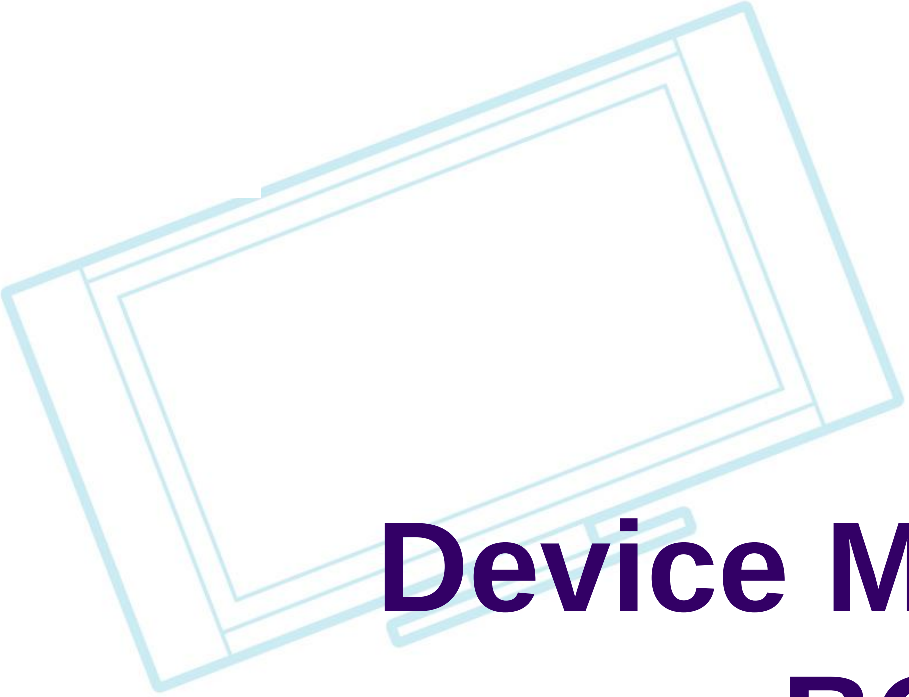
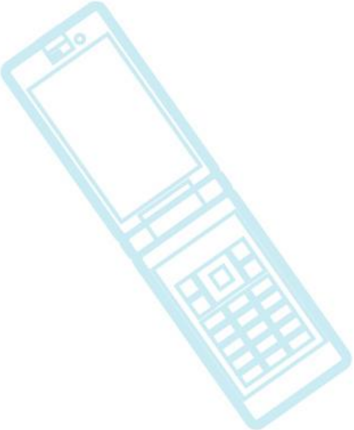
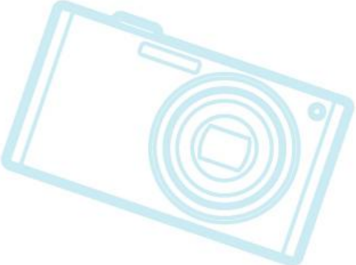
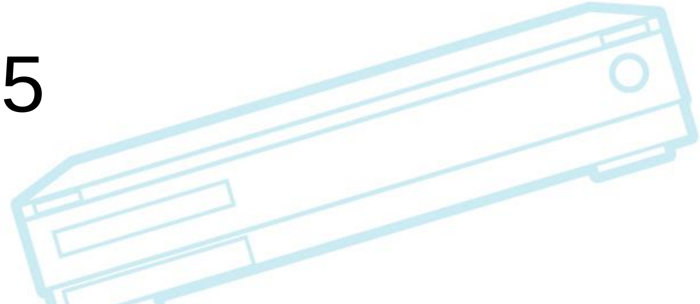



Device Mainlining BOF

October, 2015

Tim Bird

Senior Software Engineer, Sony Mobile

Linux Foundation CE Workgroup

Agenda

- Device mainlining project
- SoC status overview
- Project activities
- Ideas for improvements
  - Incentives education
    - Qualcomm/Sony case study
  - Best Practices
- Resources
- Discussion

Device mainlining project

- Joint project of Linux Foundation and Linaro
- Goal is to increase the amount of code in mainline for mobile and embedded SoCs
  - Measure out of tree code
  - Identify technical areas to work on
- Make it easier for developers to upstream code
  - Determine obstacles to mainlining
  - Reduce or eliminate those obstacles

Project activities

- Mainline obstacles identification and education
  - Obstacles survey
  - Talk and white paper
  - Mainline training (collect resources)
- Technical
  - Out-of-tree code analysis
  - Projects to address specific technical issues
  - Tools for new contributors (want to do this)
- SIGs/BOFs to discuss issues

Mobile SoC code out-of-tree

Company	Phone	SOC	Insertions
LG	G3	Msm	2.6 M
Motorola	Moto X	Msm	1.8 M
Samsung	Galaxy 4	Exynos	1.1 M
Samsung	Galaxy S5	Msm	3.1 M
Sony	Xperia Z2	Msm	1.8 M
Sony	Xperia C	Mediatek	1.9 M
Acer	Liquid E2	Mediatek	1.4 M
Asus	Zenfone 6	Atom	2.2 M
Huawei	Ascend P7	Hisilicon	2.7 M

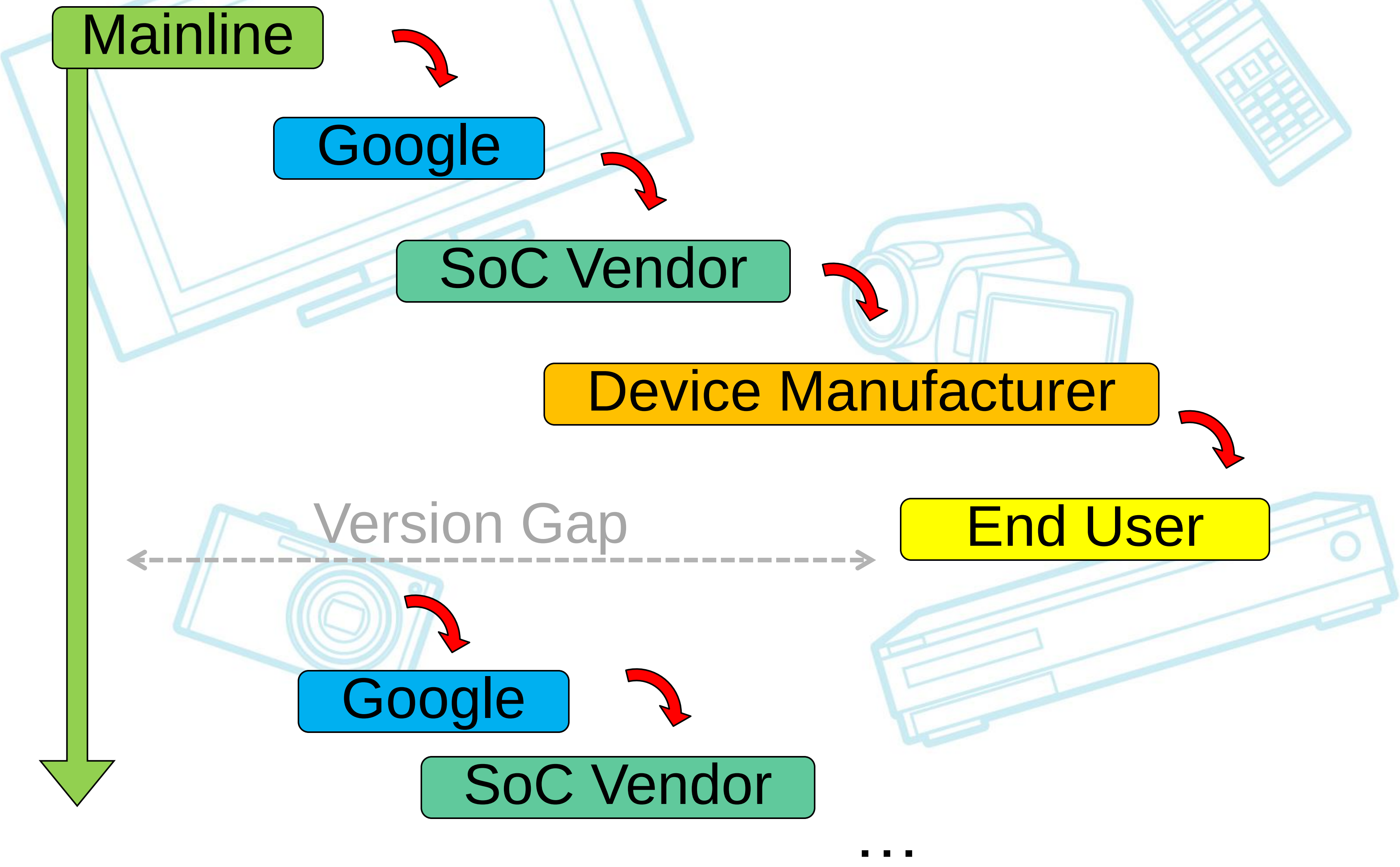
SoC out-of-tree status

- Most mobile devices have between 1 and 3 million lines of code out-of-tree
  - Code analysis for shipped products
    - Phones from 2014, v3.4 era
  - About 3 years and 20 versions behind mainline
    - Referred to as “Version Gap”
- Phones shipping now have 3.10
- 3.18 is in the pipeline
  - No phones based on 3.18 this year, that I know of

What’s the big deal?

- For manufacturers
  - Device manufacturers don’t participate in open source
    - Version gap
    - Ghetto-ization of patches for mobile devices
- For users
  - Devices are abandoned
  - No long-term support path for their hardware
  - Breaks open source model

Source of version gap



Mainline

Google

SoC Vendor

Device Manufacturer

End User

Version Gap

Google

SoC Vendor

...

Example of version gap

- Delta between Sony Mobile and mainline kernel
  - Sony mobile dependent on upstream supplier for Linux version (3.4 in this case)

Commiter e-mail	Commits	Authors
Google/Android commits	963	61
Other	2677	828
Qualcomm	20395	635
Sony Mobile	1799	203
Between our tree and mainline base (3.4)	25843	1757

...

Obstacles survey, talk, paper

- Determining obstacles
- Survey of corporate developers who don't contribute
  - "Obstacles" Talk and White Paper
  - White paper at: http://elinux.org/images/e/ed/Overcoming-Obstacles-to-Mainlining-White-Paper-version-0.9.pdf
  - LWN.net article at: https://lwn.net/Articles/647524/

Survey

- Conducted survey in September 2014
- Top obstacles (from survey):

Obstacle	How many agreed?
Older kernel version	54%
Depends on other code not upstream	50%
It’s too hard to contribute	45%
Could not test	41%
Employer does not provide time	40%
Patch not good enough	35%
Afraid of rejection	33%

Technical Analysis

- SoC out-of-tree code analysis
  - Find technical areas to work on
  - Upstream-analysis-tools
    - Set of tools to categorize diffs between production source trees and mainline
    - http://elinux.org/Phones_Processors_and_Download_Sites
    - https://github.com/tbird20d/upstream-analysis-tools
- Mainline technical areas of focus
  - http://elinux.org/Kernel_areas_of_focus_for_mainlining
    - Has notes for major areas of out-of-tree code, and ideas for projects to work on

Big problem areas

Area	Insertions range
Mach-xxx	347K – 417K
Media	120K – 360K
Video	37K – 346K
Wireless	80K – 250K
Sound	74K – 240K
Input	51K – 238K
Camera	50K – 210K
GPU	36K – 172K
Power	44K – 94K

Specific technical projects

- Wireless drivers
  - Help mature the mainline broadcom wireless driver
    - CEWG project to test brcm80211 on form-factor hardware
    - See http://elinux.org/Support_mainline_Broadcom_ wireless_driver_on_an_Android_platform
- USB
  - Integration with charger
  - Extcon for USB pins not connected to controller hardware

Other technical areas

- Sensors – promote the use of IIO
- Charging – need kernel framework for this
  - Lots of vendor charging code is in userspace now
- NFC/GPS/Bluetooth (and other things with UART-based drivers and weird sideband channels)
  - UART slave

Incentives

- Incentives
  - Google has little incentive to mainline code
    - Like VxWorks for Wind River in 1990s, BSP and drivers done by 3rd parties
    - It's good to be popular
  - SoC Vendors are on a treadmill
  - Way to get them to see benefit is to have a competitor get a financial advantage
- Sony/Qualcomm case study
  - Have worked over 2 years to get mainline support for a form-factor product
  - Now have a standard distro (Arch Linux) running bluetooth, wifi, display with Linux 4.3-rcx and about 15 patches
  - Can we convert that to cost reductions?

Best Practices

- Measure software costs
- Select hardware with upstream drivers
- Have a small team, off product treadmill, dedicated to upstreaming
- Train developers in open source methods
- Use open source methods internally
  - Mail lists, review, git, etc.
- Periodically review patches and categorize to: upstream, forward port, drop
- Post patches to get community feedback
  - Don’t make them perfect before sharing

Device Mainlining project resources

- Main project web links:
  - http://elinux.org/CE_Workgroup_Device_Mainlining_Project
  - Technical projects:
    - http://elinux.org/Kernel_areas_of_focus_for_mainlining
  - Other projects:
    - http://elinux.org/Mainlining_improvement_ideas
    - http://elinux.org/Kernel_Mainlining
- Mailing list:
  - http://lists.linuxfoundation.org/mailman/listinfo/device-mainlining

DISCUSSION

(Switch to elinux page now)

Possible discussion points

- Technical issues:
  - What other areas are deficient?
  - Are there solutions being worked on?
- Non-technical issues:
  - Convincing management to contribute
  - Big problem seems to be multi-OS code
  - Examples of code reduction between out-of-tree and mainline drivers
  - Examples of maintenance reduction between out-of-tree and mainline code
    - Reduced patch count?

- Slides after this one are here for possible discussion if specific points come up:
  - samsung/qualcomm comparison, kernel contribution stats
  - where companies get stuck, DT review bottleneck
  - Ideas from march meeting

Samsung Highlights

Area/directory	# of lines insertions (msm)	# of lines insertions (exynos)
Mach-xxxx	347K	89K
Media	364K	163K
Video	346K	176K
Sound	239K	86K
Wireless	251K	80K
Firmware	242K	101K
Input	238K	51K
Camera	121K	1K
USB	117K	35K
DTS	99K	0K
Gpu	53K	172K
Total	3105K	1100K

Kernel contribution notes

- Contributions by different companies

Author email domain	commits	Commiters (since 3.4)
Sony[me] *(sonymobile)*	53	14
Lge.com	565	11
Huawei	1220	71
Qualcomm|Codeaurora	1349	46
Moto	1035	15
Free-electrons	2333	9
Samsung	7031	160
Intel	17374	469

Results from: git log v3.4.. –author=<expr> --format=%ae | sort | uniq | wc -l

Where companies get stuck

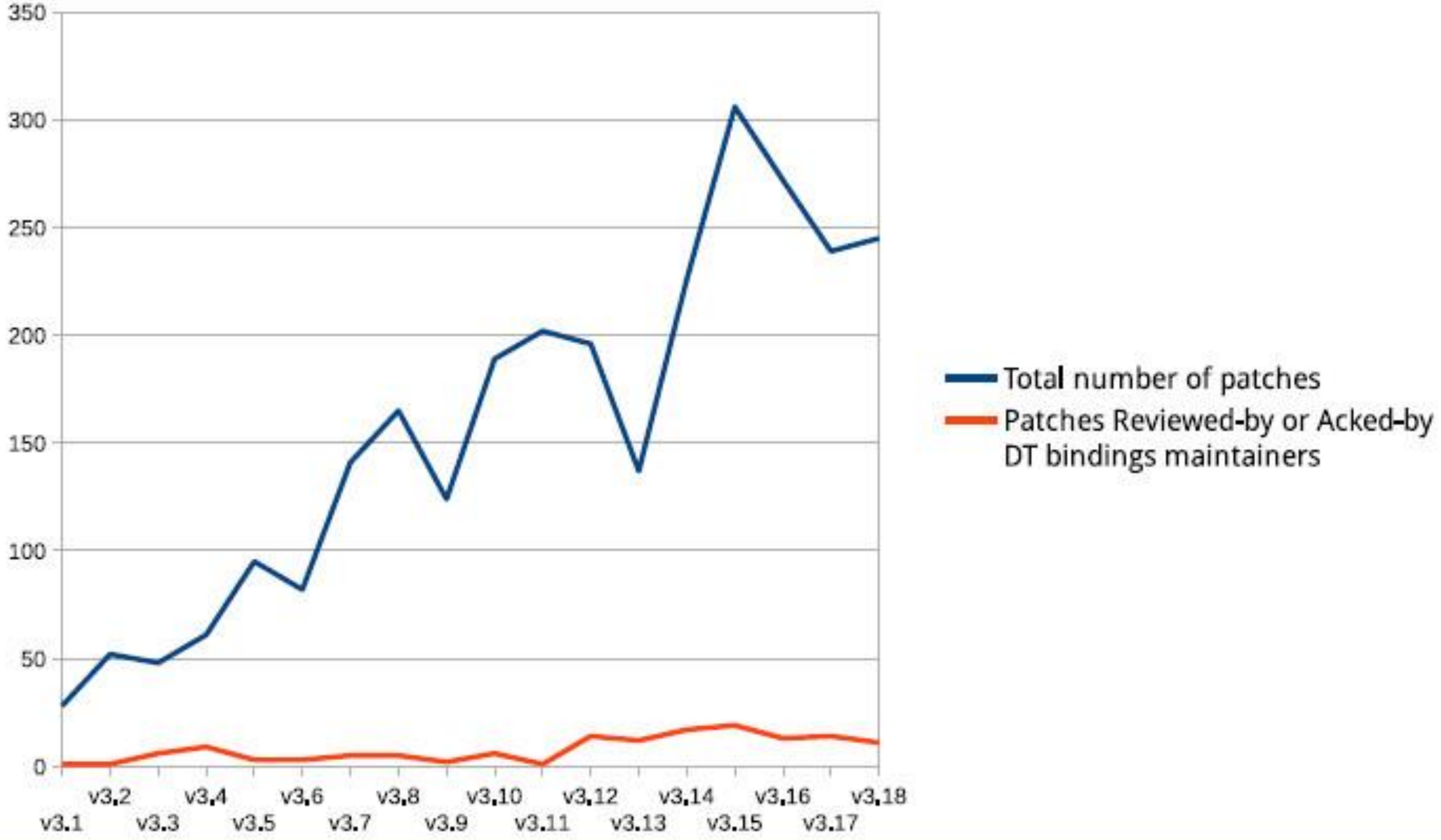
- Discussion from SIG meeting in March
  - Incentives
  - Product treadmill mismatch with mainlining
    - Product teams are too busy to learn OSS methods and contribute
  - Technical issues

Technical/Community issues

- Devicetree binding approval bottleneck
  - Thomas Petazzoni’s slides (next page)
    - From “Device Tree Stable ABI – a Fairy Tale”, presented at ELC
- Slow (non-responsive) maintainers
  - Example: hwspinlock, rpmsg, rtc
- Framework issues
  - Example: upstream USB - state machine doesn’t know about charging. Obviously can’t be used for real products.

Enough review?

- Stability of the system call ABI is achieved by careful review of the proposed changes.
- What amount of review do we have for DT bindings?



Ideas

- DT staging / SoC support in staging?
- Maintainer assistance
  - Help overloaded or slow maintainers
- Specific frameworks or sub-systems
  - Wireless, USB gadget (already discussed)
  - Media, video, sound, input
  - NFC, bluetooth
  - Low-level SoC support (mach-xxxx)
    - Regulators, clocks, resets, gpio, pinctrl, inter-processor communication, power management