(Kernel) Upstreaming 101: Social and Technical Lessons

Oct 29, 2012

Deepak Saxena
dsaxena@plexity.net

Linaro CONNECT

The Problem

- Ever increasing number of Linux developers!
  - Specially in the ARM world
- Lots of good technical documentation!
  - *Linux Device Drivers* - http://lwn.net/Kernel/LDD3/
  - *Understanding the Linux Kernel* - http://goo.gl/p4pM4
  - *Linux Kernel Development* - http://goo.gl/0YY27
- Organizations beginning to understand technical and business benefits of getting code upstream!
- More developers interested in working upstream!
- This is a problem?

The Issues...

- Using documented APIs and infrastructure is easy
  - Creating new APIs and infrastructure is hard
    - Creating good APIs and infrastructure is harder
      - Creating good APIs and infrastructure for upstream is even harder
    - Many SOC problems require creating new subsystems
      - Or modifying existing ones to support new features
- Books tend to be x86 oriented
  - Do not cover drivers common to other arches: i2c, spi, etc
  - Make assumptions about underlying HW
  - (See Upcoming *Porting Linux* by Jon Masters, ETA 2013)
- Working with the community is not documented
  - Much tribal knowledge on the process
  - Same mistakes are often made by new participants

Today's Goals

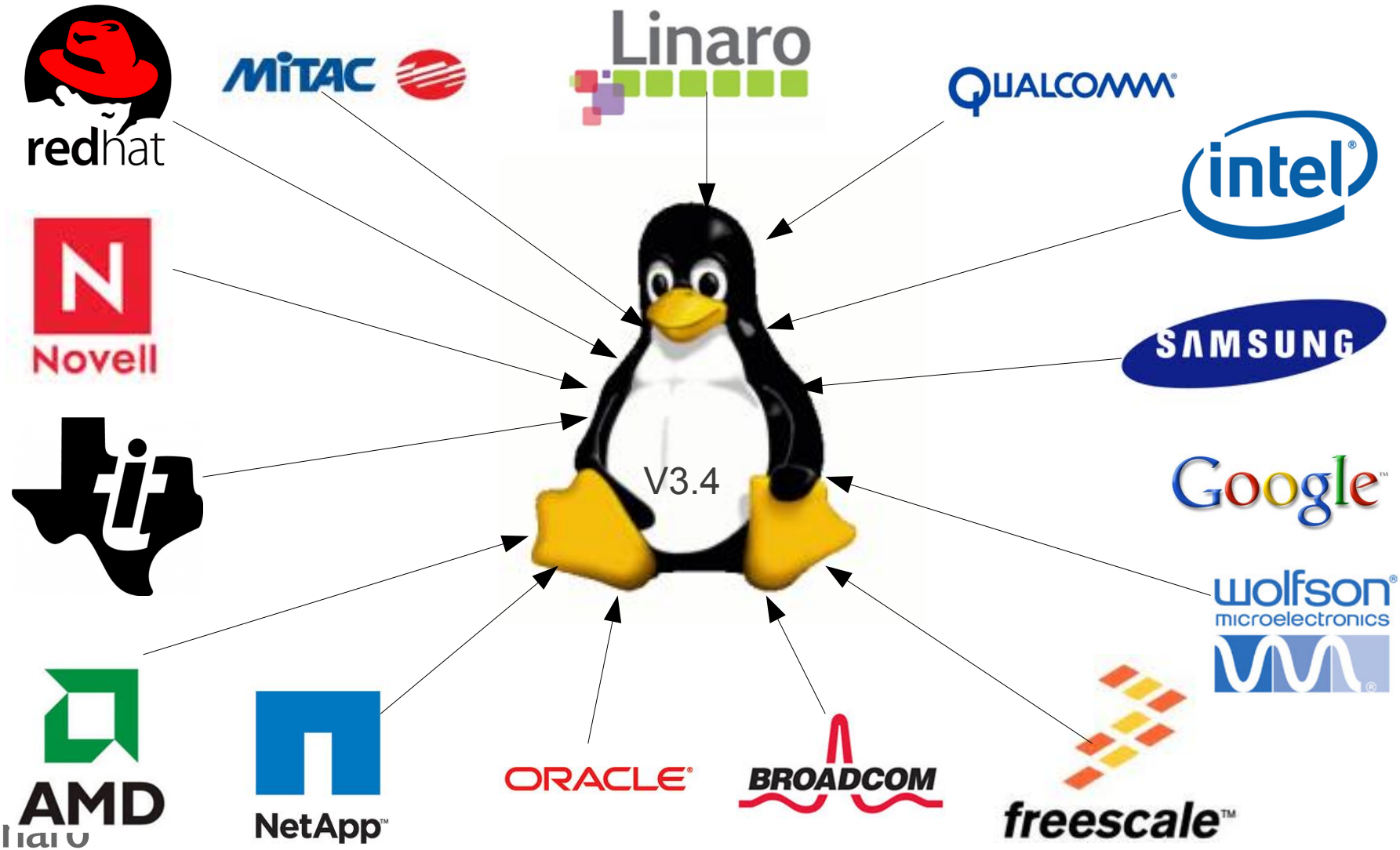
- Provide an overview of upstreaming code
  - Social and Technical
    - From Design to Submitting Patches
      - How does the process differ from closed development?
      - Some general guidelines on design and coding.
      - Some low level details on submitting patches.
  - Geared at those who are completely new to this world
    - For both Managers and Engineers

Who Am I? Why Listen To Me?

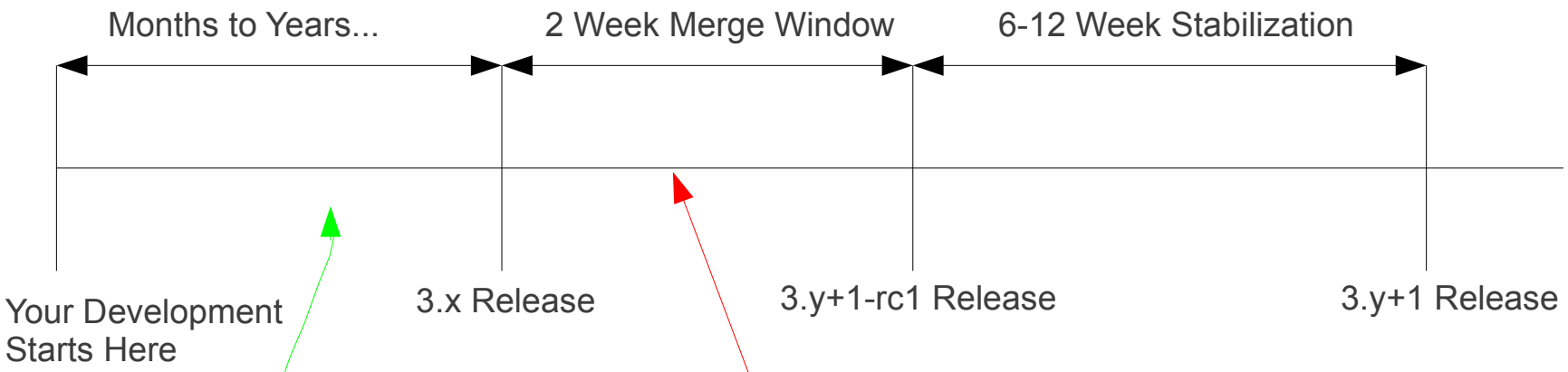
- Kernel Working Group TL
  - I make spreadsheets and run meetings
- Before that:
  - 12 years kernel experience
    - One of first folks at Intel to work on Linux drivers
      - (though a little under the radar...)
    - Developed and maintained IXP* Xscale NPU ports
    - Kernel maintainer at MontaVista for several years
      - Reviewed all patches for distro kernel
      - Reviewed thousands of lines of vendor BSP code
  - OLPC kernel maintainer for about 1.5 years
  - Was active in reviewing code upstream
  - Have given similar talk at various conferences:
    - Linux Bangalore, ELC, LinuxConf.au, Fosdem

Who Exactly Is The Community?

(From list of top 3.4 contributors: http://lwn.net/Articles/496193/)

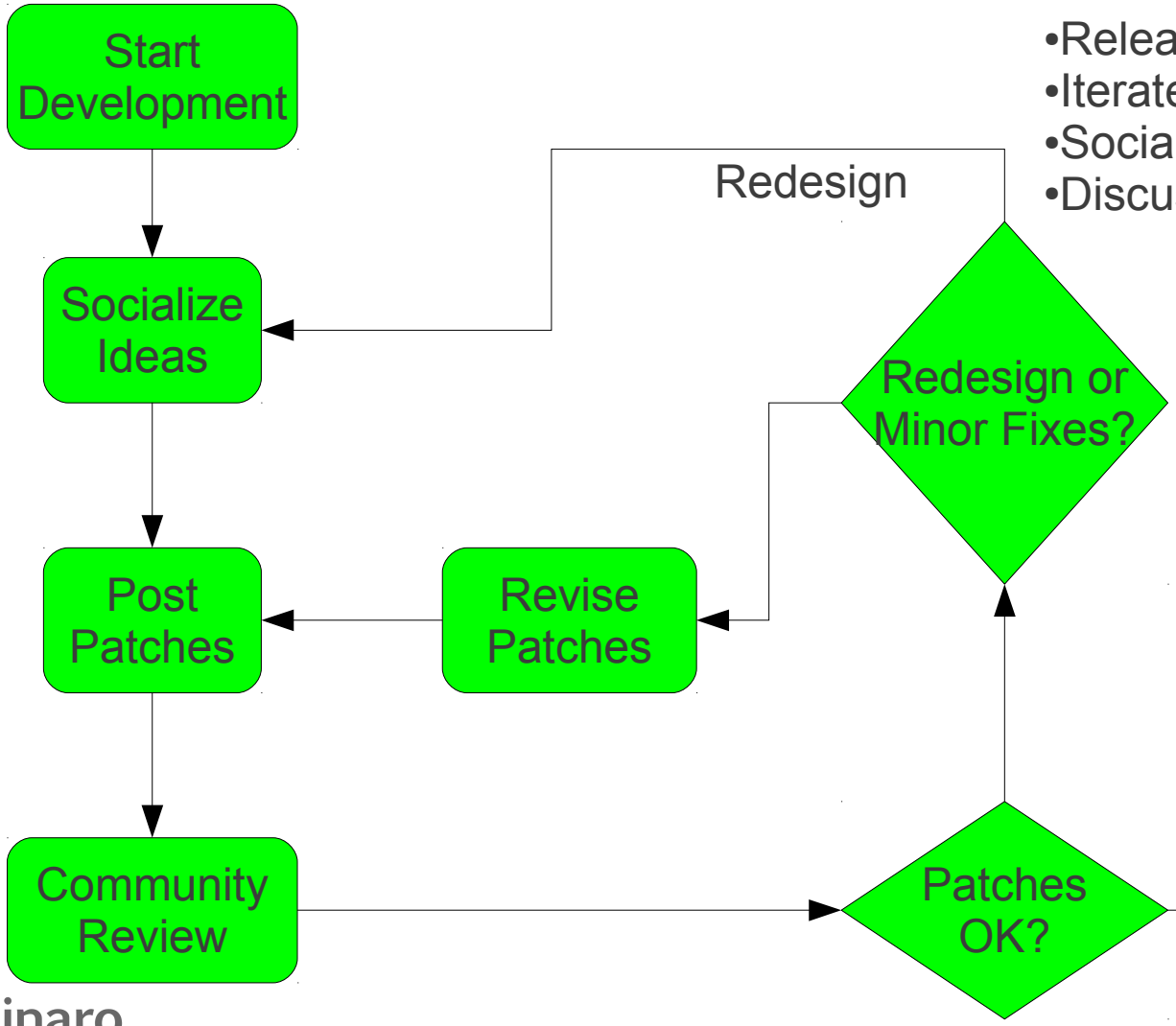


Upstream Development: The Right and Wrong Way

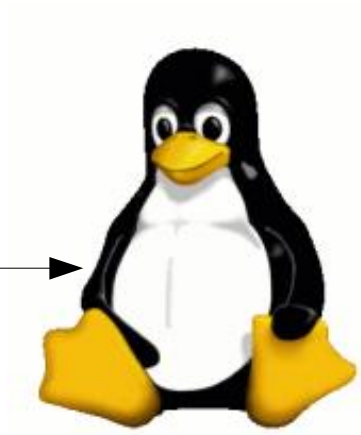


If you wait until this window to release your code, expecting it to just go in, it will not (there have been a few exceptions, but very rare). You need to **release your code early and often** during your development phase to get feedback and fix issues as they come up.

Upstream Based Development



- Release Early, Release Often
- Iterate Design/Code Cycles
- Social component very important
- Discussion Time >> Coding Time



Avoiding Issues: Don't Re-invent the Wheel

- Do Your Homework
  - Many problems already have known solutions in Linux
    - Driver APIs for specific functionality
    - User/Kernel interfaces and system level tools
    - It may just not be obvious at first.
      - May not be documented in one single place
      - May just be tribal knowledge
    - Google is your friend here
    - Read the code
    - Ask the experts
  - Release Early, Release Often to save your time!

Avoiding Issues: Abstraction

- Don't Abstract Unnecessarily
  - NO cross-OS HALs (unless extremely well designed)
  - Don't add wrappers around existing in-kernel APIs
  - Too much abstraction makes code harder to maintain
    - Code is harder to read/debug
    - Abstraction layer might have bugs
    - Kernel API changes are hidden
    - Difficult for someone new to get involved
  - Release Early, Release Often to find these issues early!

Avoiding Issues: Solving Common Problems

- Your HW is Not Unique
  - Your HW may seem unique at first, however...
    - Others are close by or will soon follow
    - Most HW comes from similar research paths
    - They will need same kernel support as you do
  - Don't blindly add hooks to kernel for these features
    - We don't want multiple implementations of same idea
    - Need to add APIs at proper points in kernel
      - Work with external developers to develop these
    - Work with other HW vendors
      - !THIS IS WHAT LINARO IS ALL ABOUT!
  - Release Early, Release Often to find common solutions!

Avoiding Issues: Good API Design

- Good APIs are:
  - Easy to use & Hard to misuse
    - Follow KISS Principle: Keep it Simple Stupid
      - Should be very clear from just name, parameters
      - Should be obvious when you use it wrongly
        - Build time errors or obvious error values
  - Just read Rusty's Blog Post:
    - http://ozlabs.org/~rusty/index.cgi/tech/2008-03-30.html
  - Release Early, Release often so you don't built a whole stack of software on top of poorly designed APIs!

Practicalities: How Do I Post Patches?

- Read Documentation/SubmittingPatches
- Send email to maintainers and list
  - One cover email with:
    - Summary of WHAT you are solving
    - Explaining WHY you used your approach
    - Talk about any dependent patchsets
    - Don't assume knowledge about your technology area
    - git log --diff (which includes patch headers)
  - Always post patch against latest tip
  - One message per patch
    - Break up your changes into small units that build on each other
      - If you are adding 100's of lines to one function...you're doing it wrong
      - Refactor your patches using “git-rebase -i”
  - Go to today's git session

Practicalities: Who do I send my patches to?

- Use git-log and git-blame commands
  - git-log to find out the last few people to make changes
  - git-blame to find out if
- Read MAINTAINERS file in top level directory
- get-maintainers.pl script
  - Will give you mailing list and maintainer address

```
~/src/linux$ ./scripts/get_maintainer.pl spi_message_queue.patch
Grant Likely <grant.likely@secretlab.ca> (maintainer:SPI SUBSYSTEM)
Randy Dunlap <rdunlap@xenotime.net> (maintainer:DOCUMENTATION)
spi-devel-general@lists.sourceforge.net (open list:SPI SUBSYSTEM)
linux-doc@vger.kernel.org (open list:DOCUMENTATION)
linux-kernel@vger.kernel.org (open list)

~/src/linux$ ./scripts/get_maintainer.pl -f drivers/mfd/max8925-core.c
Samuel Ortiz <sameo@linux.intel.com> (supporter:MULTIFUNCTION DEV...)
linux-kernel@vger.kernel.org (open list)

:~/src/linux$ ./scripts/get_maintainer.pl -f drivers/i2c
"Jean Delvare (PC drivers, core)" <khali@linux-fr.org> (maintainer:I2C SUBSYSTEM)
"Ben Dooks (embedded platforms)" <ben-linux@fluff.org> (maintainer:I2C SUBSYSTEM)
"Wolfram Sang (embedded platforms)" <w.sang@pengutronix.de> (maintainer:I2C SUBSYSTEM)
linux-i2c@vger.kernel.org (open list:I2C SUBSYSTEM)
linux-kernel@vger.kernel.org (open list)
```

Practicalities: Using Signed-Off By

- Signed-off-by *is legally binding!*
- Make sure you have approval!

```
Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

Practicalities: Nobody Replied to My Patches!

- Be patient, don't expect an immediate response.
- Check mailing list archives:
  - What's the typical response rate?
  - Did the maintainer announce a vacation?
- Is this a merge window?
  - Everyone is super busy, try again when it closes
- If it really seems like patches got dropped, email the maintainer privately with a pointer to thread.
  - If still no response:
    - Ping one level up in tree.
    - For drivers: Ping Greg KH
    - Sometimes you just have to email patches to Linus directly
      - Shames the maintainer, gets his/her attention

Practicalities: Responses From Other Developers

- Acked-by: This person is OK with the changes. Usually a maintainer of a subsystem affected by patch.
- Reviewed-by: This person has done a thorough technical review of the patch.
- Tested-by: This person did some level of testing. Allows maintainer to know that it has been validated on an environment other than original author's.

Practicalities: When Someone Attacks You :(

- It unfortunately happens :(
- Do NOT Escalate
  - Take a step back
    - Go work on something else
    - Grab a beer (BUT NOT WHILE REPLYING!)
- Pick out the technical issues
    - Get some help in reading the negative email
    - Respond to just these
- Hint: Attending conferences and meeting people face to face reduces the likelihood of this happening.

Practicalities: I *can't* Release Early and Often

- Ask for Private Reviews
  - NDAs with upstream maintainer(s) and experts
    - Linaro can possibly help here
      - Several of these people are part of our organization
      - We have good relationships with many upstream maintainers
- Can you release the concepts if not the code?
  - Fine art of providing enough details w/o giving away the secret sauce
  - What are the high level problems you want to solve?

Summary

- The community is just a bunch of developers
  - Like you, just with more experience of the process.
  - Most of them want to help!
- Good kernel code:
  - Doesn't re-invent what's already there
  - Doesn't add abstraction for the sake of abstraction
  - Has well-designed APIs
  - Solves common problems
- Read the documentation and the code
- Participate openly and respectfully
- Ask us for help
- **Release Early, Release Often :)**

Questions or Comments?