Linux collaboration summit
April, 2012
San Francisco, CA

Let's kill all proprietary drivers for good

Luis R. Rodriguez
Adrian Chadd
Qualcomm Atheros

How to kill proprietary drivers

- Prioritize clean kernel APIs / code
- Permissively license Linux kernel drivers
- Localize usage of the GPL
- Pick a BSD, call for unification of BSDs
- Intellectual property strategies
- Replace "internal codebases" in favor of working with two upstreams:
  - Linux
  - BSD
- If driver code unification is desired, make driver unification a community problem
- WANALs, proactive engineering strategies

Where did this come from?

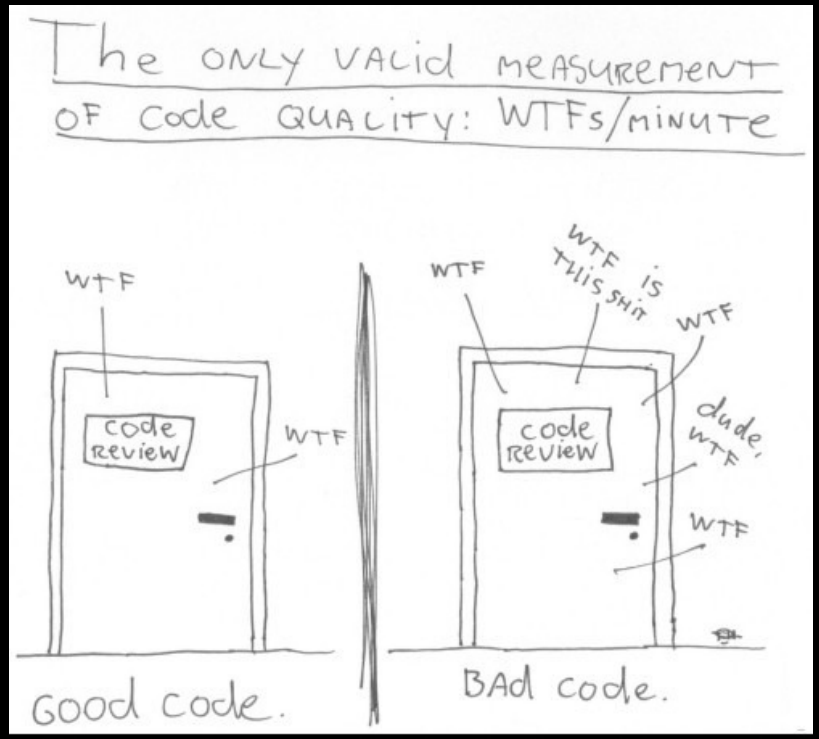
- Atheros / Sam Leffler maintains MadWifi uses net80211
- BSDs picks up net80211
- Linux rejects net80211 - oh well
- Sam Leffler, FreeBSD, moves on
- Linux gets mac80211, cfg80211
- Community upstreams ath5k with **SFLC**'s help to share HAL code with OpenBSD
- Linux community abandons MadWifi
- Luis joins Atheros - Linux hacker
- Upstreaming of the Linux ath9k driver by Atheros under **ISC**
- Broadcom follows **ISC license strategy**
- Adrian Chadd picks up Sam Leffler's work
- Adrian Chadd joins Atheros - FreeBSD hacker
- Review of internal driver infrastructure
- Proposals for how to help internal driver infrastructure

A pattern on proprietary drivers

WTF metrics off the charts

for all non-upstream

Linux vendor drivers



Why? How did we get here?

- BSD, Linux, Microsoft, Apple, Solaris, QNX, etc
- With the PC, explosion of new hardware components, hundreds of drivers for different OSes
- With mobile, explosion of other new hardware
- OS priorities based on type of user
- Linux only started becoming a priority on the "desktop" recently
- Microsoft, Apple drivers: good **run time tests**, validation, nice shiny certification logos
- Not all FOSS projects have good software.. but
- Linux, BSDs: software architect assholes

What else is involved in delivering a driver?

It's not just about writing code! That may even work!

- There **may** be compliance testing and certifications needed before a product can be sold
- There **may** also be (a lot, I hope) a lot of internal standards and regression testing
- There **may** be some very specific customer extensions that they've paid for, but not allowed for public consumption
- There **may** be cross-licencing of patents..
- .. and some commercial agreements.

These **may** make it difficult to open up a commercially developed codebase.

What about multi-OS drivers?

- Because there's more to writing a driver than just writing the code..
- .. there **may** be pressure to share as much code as possible between platforms
- The Atheros softmac driver targets:
  - Windows (XP, Vista, 7, 8)
  - Linux (various 2.6, 3.x kernels)
  - Legacy - vxWorks, *BSD
- Compare the complexity of a simple 10/100 Ethernet NIC driver to what is required for 802.11 support
  - Do you really think it's feasible to reimplement, from scratch, the entirety of an 802.11 stack for each operating system?
  - Will each OS / stack require its own full tests?

Multi-OS drivers - the problem

- But how much can you really share?
  - What is actually, fundamentally different between OSes?
    - device model, threading/locking model, network model, bus model..
- What do you end up having to do?
  - Find the "middle ground" between all OS platforms
  - This may not be the most { optimal, efficient, concise } way to solve the problem
- Every company likely does it completely differently
  - Tailored to their specific device needs
  - Can these even co-exist between companies?
  - Companies reinventing the wheel, year after year
  - **Why**?!

What is a crap driver?

- Don't use the word "**crap**" -- banned -- leads to crap
- Accepting crap means you realize you can improve software
- Crap can address **run time** functionality, really really well
  - Doesn't mean its good
- **Does not** prioritize **code readability** and **long term maintenance**
- **Does not** consider an **ecosystem** for reinventing the wheel on fixing issues
- **Branching hell**, customers fork all your work
- In Linux, anything under **drivers/staging TAINT_CRAP**
- **Even good drivers can become crap**, once you evolve, you must accept its crap
- Examples even in the Linux kernel:
  - Linux wireless regulatory, evolution
    - Big vendor crap --> upstream --> crap --> re-architect
    - Regulatory simulator in userspace
  - Staging drivers, compat-wireless crap/ for patches
- Move on, adapt fast, evolve, accept criticism

Improve internal driver codebase

- Linux porting issues:
  - Always have to clean code up
  - Community collaborations not being merged back
  - QCA developer program
- Code modularization / component owners
- Atomicity of changes
- Annotation of fixes (Cc: stable)
- Distributed development model
- Clean kernel APIs / code
- Benefits of collaborative development

Strategic localization of the GPL

- Historically one license for one single project
- BSD license first, GPL evolution
- Linux embraces GPLv2
- FSF engineers GPLv3
- Linux stays on GPLv2
- Linux and other projects use "Dual BSD/GPL" for some things, ambiguity created
- SFLC: **GPL-Compatible OK**!
  - ath5k: HAL uses simple **ISC**, 2 Clause BSD license
    - Changes-licensed-under
    - **Signed-off-by** - Educate on usage, spread it
  - ath9k: first fully permissively licensed driver
  - Solaris ports ath9k to OpenSolaris, OpenSolaris dies
  - Broadcom follows ISC license approach
- Call for **drivers** to be **permissively licensed**
- **Core kernel functionality: GPL**
- Arguments against: **only** GPL enforcement will ensure giving back
  - Alternative: proactive engineering to address corporate investments

Are non-Linux GPL drivers possible?

- Yes but its a pain in the ass
- Andy Grover:
  - GPLv2 - paravirt Windows driver
- Writing GPLv2 Windows drivers means you become a mingw developer
- http://groveronline.com/2008/07/mingw-cross-compilation-adventure/
- So possible but painful at present
- Intel drivers: some permissively licensed files, mostly headers
- Better use a fully permissive licensed Linux drivers, share with BSDs

BSD usage and call for unification

- BSDs split off **years ago** for many reasons
- **Motivate BSD contributions** by vendors:
  - Guidelines for using permissively licensed drivers on Linux
  - If you can address permissively licensed Linux drivers you should be able to address BSD as well
  - **Would be nice** if BSDs fragmentation was addressed
- If you want **proprietary bells / whistles** in kernel:
  - Use BSD
  - But you still need to address patent grants
    - **Arguably** permissive licenses have implicit patent grant
    - Better **avoid patents in kernel** for both BSD and Linux
- **Is some BSD unification possible**?
  - Not the place to ask -- but we are asking
  - Lets start wiith 802.11 unification ideas
- **BSD vendor summits**
  - Run a few times a year, both standalone and at conferences
- **BSD conventions**
  - BSDCan, EuroBSDCon, AsiaBSDCon, others here and there..

Intellectual Property

- Patent **trolling**:
  - Industry is patent trigger happy
  - Software patents in FOSS: **can of worms**
  - This issue sucks, let us, engineers **simplify the problem!**
- Kernel:
  - Linux kernel **GPLv2**: deal with it!
  - Help sharing: **localize GPL**
    - New drivers: permissively licensed
  - **Prioritize** simple and **clean kernel** API / **drivers** over adding IP in kernel. Early architectural review to address IP
  - **Punt IP to userspace**, otherwise:
    - Explicit patent grant: RCU -- GPLv2 + explicit patent grant Documentation/RCU/RTFP.txt
  - BSD kernel: permissively license patent grant **theory**
  - **ClearBSD license**: not accepted, we tried, although GPLv2 compatible
- Userspace:
  - WebM VP8 lesson: **FOSS license** + **separate patent grant**
  - Open solution + explicit patent grant for **hardware vendor?**
  - Other ideas? Innovate strategies
- We are not attorneys!
  - Attorneys can address this at the Legal tracks
  - We are engineers

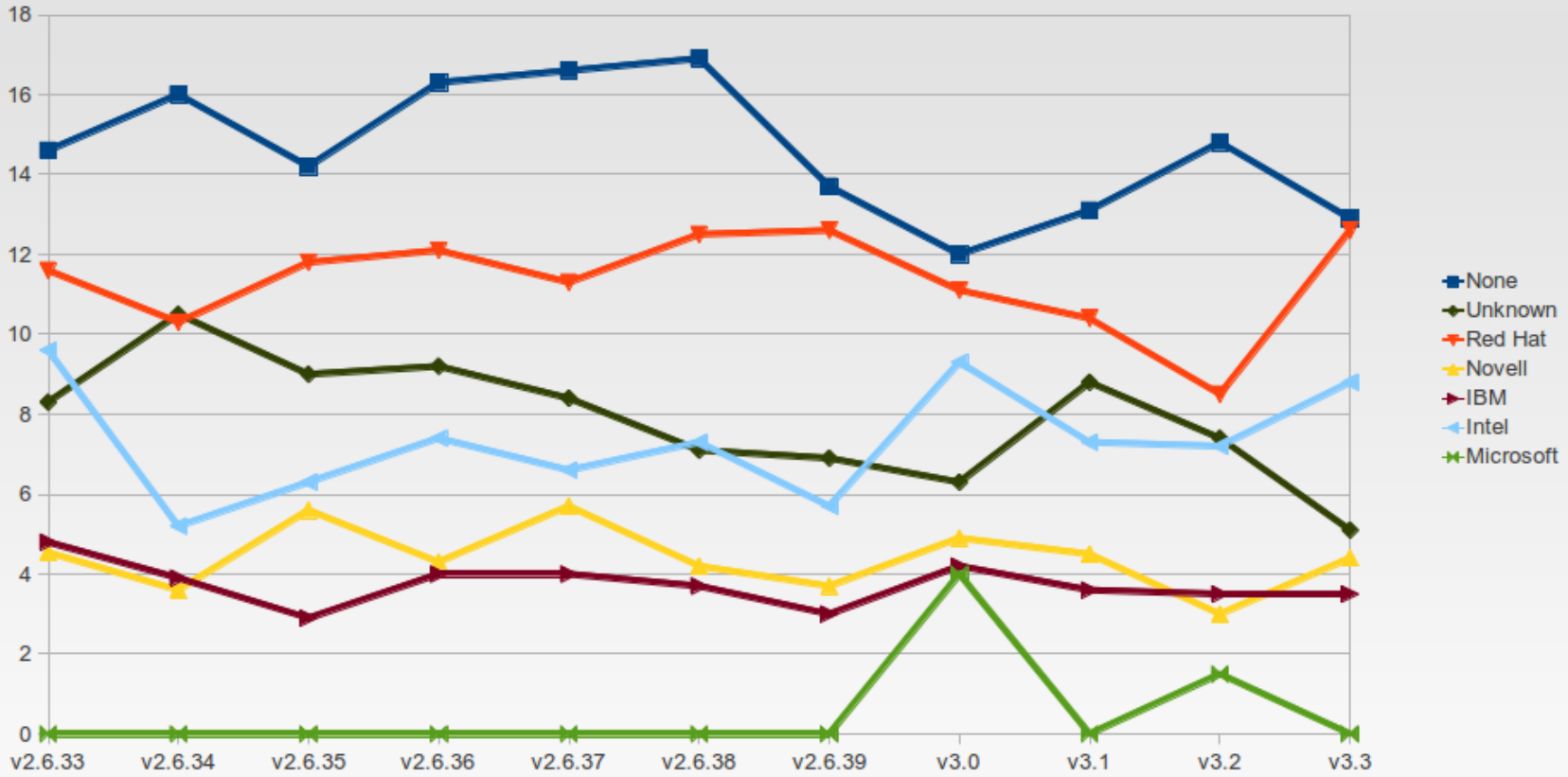
Business justifications

- Streamline / lead Operating System ecosystems
- Bring benefit of collaborative development to non-Linux Operating Systems
- Can the community be relied on?
  - Motivation - understand it
  - Innovation - can leverage off of this work
  - Geographically spread
  - Talent scouting
- Can the community be accounted for?
  - Hobbyists - top contributor to the Linux kernel
  - GSoC

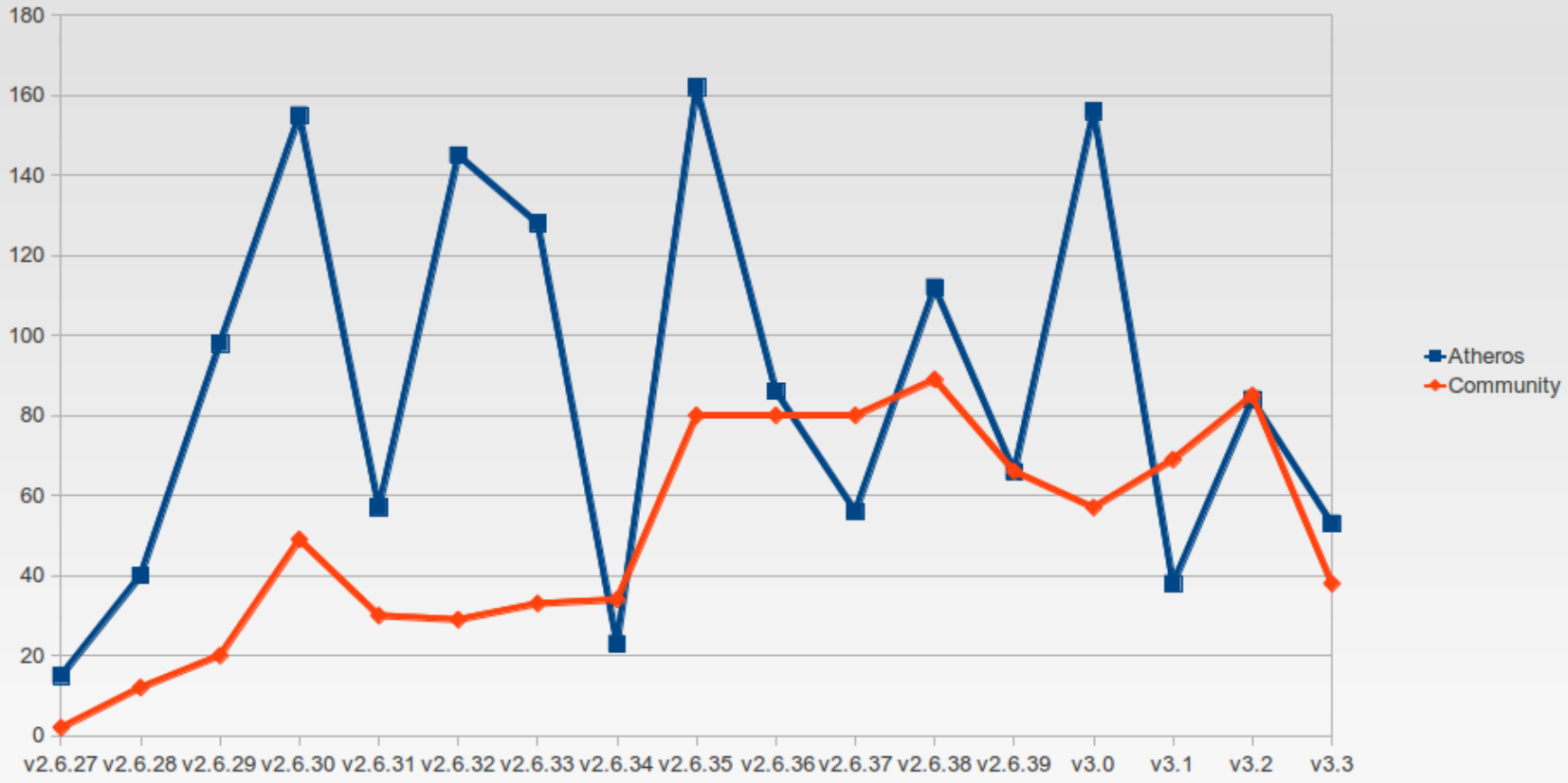
Collaborative development

- Hobbyists, top contributors
  - Linux
  - BSD
  - Wikipedia
- Ideal situation: a good balance
  - Educate: NDA, public specs, wikis
  - Stimulate motivation
  - Stimulate innovation
- Can you rely on this? Let's review history

Contributions to Linux



ath9k contributions upstream



What is Qualcomm Atheros doing?

- Lets start off simple
- Address an Ethernet driver first
- Linux alx driver:
  - Rejected on good technical grounds
  - Permissive license considerations hopefully being reviewed
  - Example of possible issues in driver unification utopia
- If not Ethernet, lets work on the next new driver, whatever that is, the simpler the better
- Bluetooth: too simple, what's next?

Can we do better?

- Is driver unification a pipedream?
- Can we unify drivers on at least
  - Linux
  - BSD
- Two options:
  - Help with unification - what's in it for
    - Better coordination, testing, do we care?
  - Go away: punt problem back to companies
    - Only two upstreams possible: BSD, Linux

If unification is possible

- Lesson to be learned:
  - Corporations reinventing the wheels, horribly
  - Can the community help?
  - If we want to help kill proprietary drivers, lets talk about it
- Live with at the very least two upstreams:
  - Linux
  - BSD
- coccinelle spatch
  - SMPL - language of what patches should look like
  - Used in the Linux kernel for code sanity checks
  - Used to help clean drivers out of drivers/staging
  - Used to help address large kernel API changes
- spdiff
  - Shiny new tool
  - Lets you skip writing SMPL
  - Automatic inference of high level specifications of changes to structured data
  - Give it two patches: it gives you SMPL

The end

Objectives to consider for killing proprietary drivers in the long run as engineers. Each one may take time to achieve.

- Architecturally punt IP "problem" to userspace moving forward
- Dual BSD / Linux strategy
- Driver unification, a community problem

Attorney's homework: FOSS friendly IP strategies in userspace