MAINTAINING AN OUT-OF-TREE DRIVER AND AN UPSTREAM DRIVER SIMULTANEOUSLY

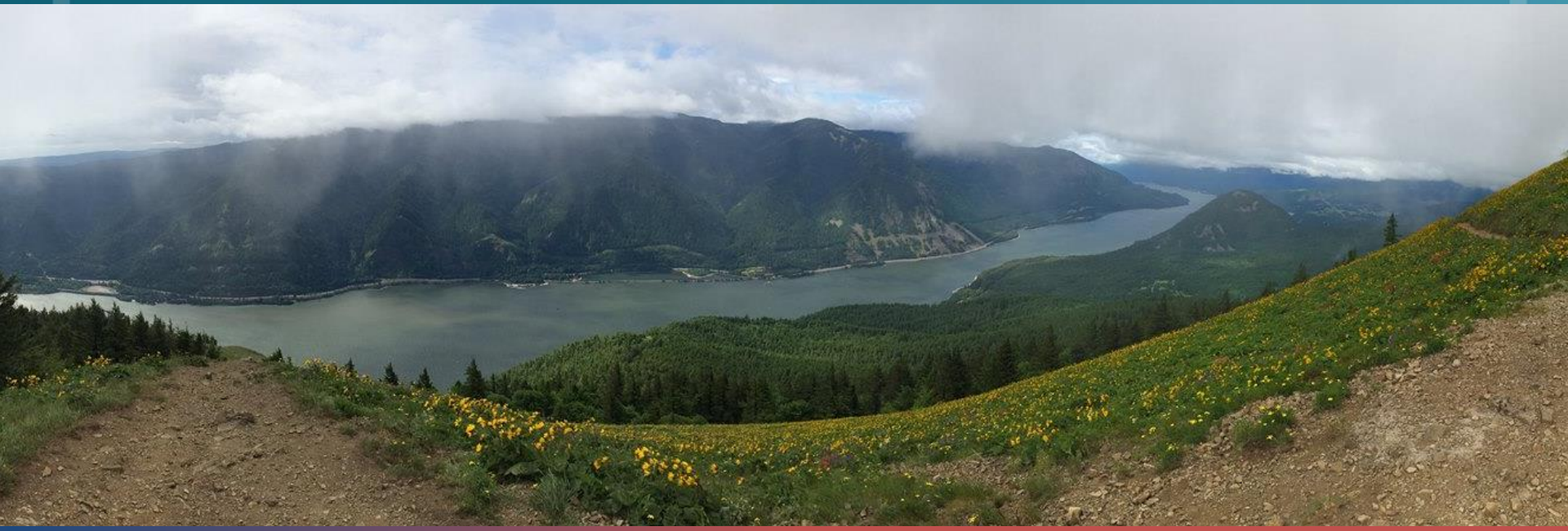
(WITH MINIMAL PAIN)

Catherine Sullivan
Intel

LinuxCon 2015

ME

- Intel ND Linux Ethernet drivers
- 40G product line
- A little 10G



OVERVIEW

- Why do we need two drivers?
- How ixgbe and igb maintain two drivers
- The problems we had with that method
- How i40e maintains two drivers
- The advantages to the new method
- What our continuing problems are
- Future steps

WHY DO WE NEED TWO DRIVERS?

OUT-OF-TREE

- OEMs need a driver for the CD they ship with the hardware
- Backwards kernel compatibility
- Sandbox to play
  - Silicon validation
  - Emulation platforms
  - A0 hardware
- Stand-alone driver

UPSTREAM

- Distros pull driver patches from the kernel
- Easier for customers using the latest kernel to have the driver built in

HOW ND SUBMITS PATCHES UPSTREAM

- We submit patches to intel-wired-lan list which are tracked by patchwork
- Jeff Kirsher applies them to his queue on kernel.org
- Our validation engineer tests the patch
- Jeff submits the patches to net(-next)

HOW IXGBE AND IGB WORK

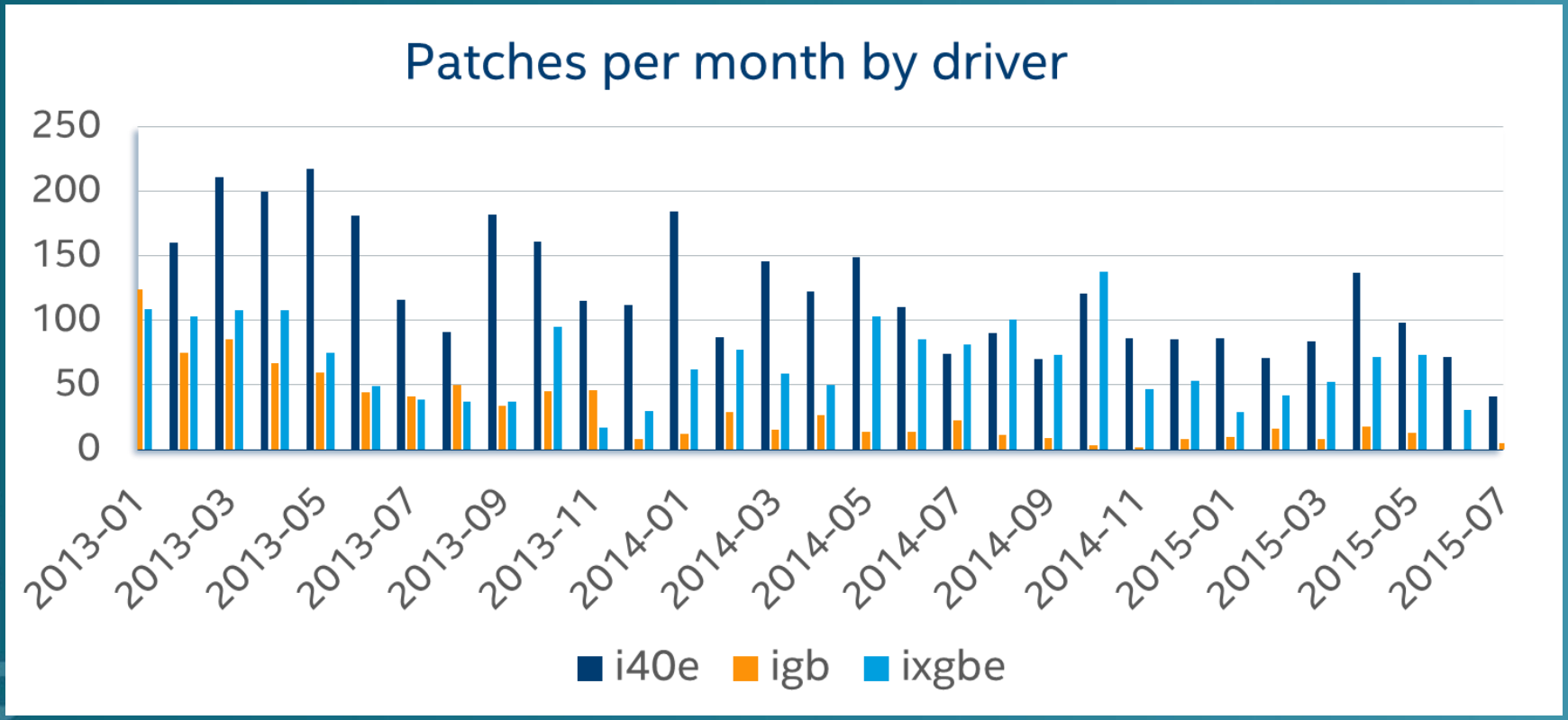
- The developer is responsible
  - The developer has to generate two patches, one for each driver and submit them
- Community patches get pulled into the Out-Of-Tree driver by the development team

THE PROBLEMS WITH THIS METHOD

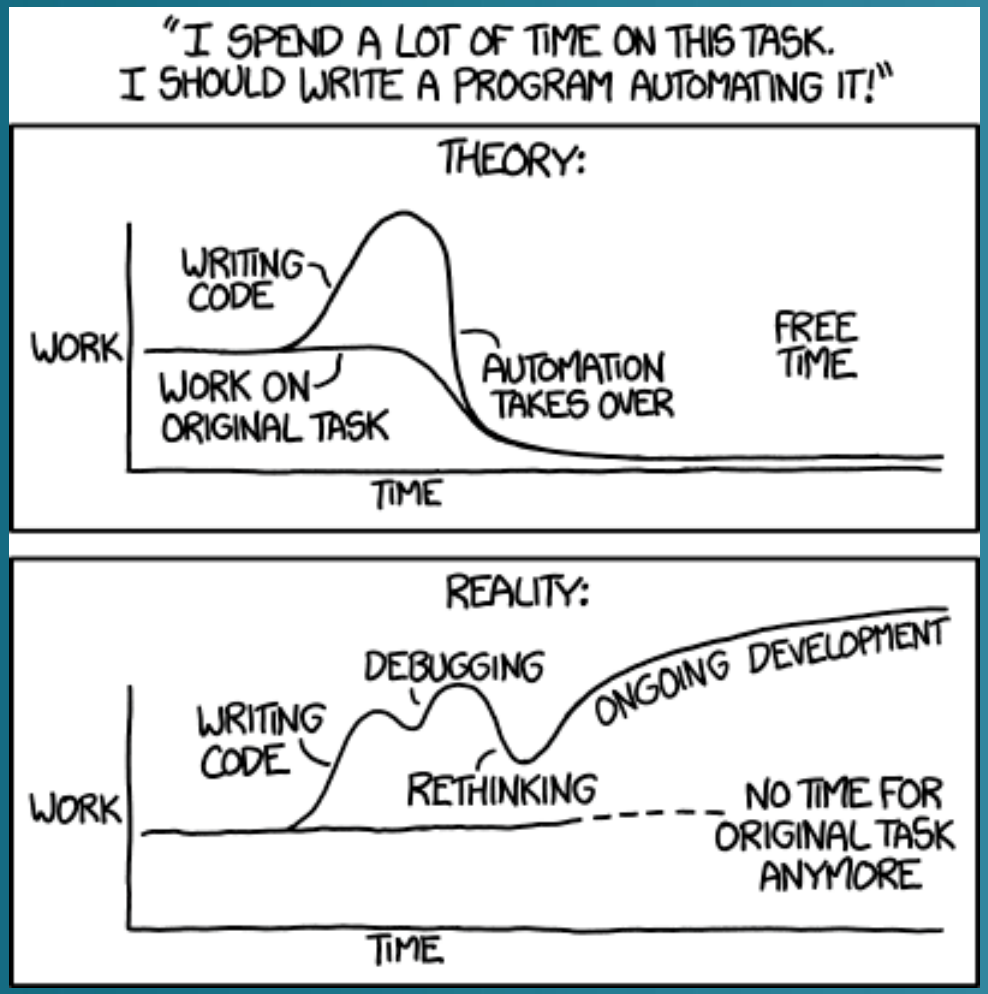
- Easy to miss patches – no one is checking that everything is upstream
- No easy way to compare drivers
- Community patches often get missed because no one person is responsible for them

THE BIG PROBLEM

Duplicate work becomes much worse when there is more work to duplicate



THE NEW METHOD USED FOR I40E



Automation - https://xkcd.com/1319/

THE NEW METHOD USED FOR I40E

- Kernel stripped build of Out-Of-Tree driver
  - Build flags to strip/transform code
  - ixgbe and igb actually have this but don't use it this way
- kernelpatch.sh
  - Script to generate upstream patches between two tags
- Human
  - Apply and fix-up patches

KERNEL STRIPPED BUILD OF OUT-OF-TREE DRIVER

- Strip Flags
- Keep Flags
- unifdef
- BUILD=KERNEL flag
  - Legacy Interrupts
  - VXLAN

```
# build.mk
# Makefile for generating stripped source for various build types
#
# Strip flags:  if preceded by a -D,    removes #ifdefs but leaves code
#                                       removes #ifndefs and code
#               if preceded by a -U,    removes #ifdefs and code
#                                       removes #ifndefs but leaves code
# Keep flags:   if not specified        removes all #ifdefs and code
#                                       removes all #ifndefs but leaves
code
#               if specified            leaves #ifdef and code
#                                       leaves #ifndefs and code
#               if prefixed with -      removes #ifdefs but leaves code
#                                       removes #ifndefs and code
# Note that strip flags always apply first!
```

KERNEL STRIPPED BUILD OF OUT-OF-TREE DRIVER

- Coccinelle patches
  - Semantic patches to make code more upstream conformant
- Script to fix duplicate definitions in i40e and i40evf

```
// put parens around the '-' match in order to make the parser not leave
// extraneous () around ((ret == I40E_SUCCESS) && foo) transforms
@@
expression E;
@@
- (E == I40E_SUCCESS)
+ !E

@@
expression E;
@@
- (E !=I40E_SUCCESS)
+ E
```

KERNELPATCH.SH

- Takes one or two tags as arguments
- Check the tags exist in the repos
- Reset to first tag
- Save i40e(vf)-a
- Iterate over patches
  - Checkout next patch
  - Save i40e(vf)-b
  - Check parent
    - git rev-parse --verify "$SHA^"
  - Save diff of A and B, the patch description, and sha
  - Save the title into list
  - Save B into A
  - Clean the directory

KERNELPATCH.SH

- Remove reverted patches
- Remove empty patches
- Create list of ALL patch titles
- Additional options
  - Email patches to author & maintainer
  - Generate a single patch

```
function usage() {
        echo "Usage: $0 <prevtag> [<stoptag>] \
[email <name> [test]] [single <core|shared>]"
}
```

HUMAN

- Apply patch
  - Resolve conflicts
- Squash patches if necessary
  - Patches in the same series that touch the same code is frowned upon
- test.sh
  - Compiles i40e & i40evf
  - Sparse check
  - checkpatch.pl
  - Check patch description

HUMAN

- testall.sh <n>
  - Git rebase interactive, execute test.sh on each patch
- Format and email patches to intel-wired-lan

TOOLS THAT HELP

- git rebase –i
  - Reorder, squish, change description, edit patch, execute a command
  - Edit and cleanup the stack of patches
- git notes
  - Add notes to a commit without changing the commit
  - Helpful noting that a future patch has to be squashed with this patch

TOOLS THAT HELP

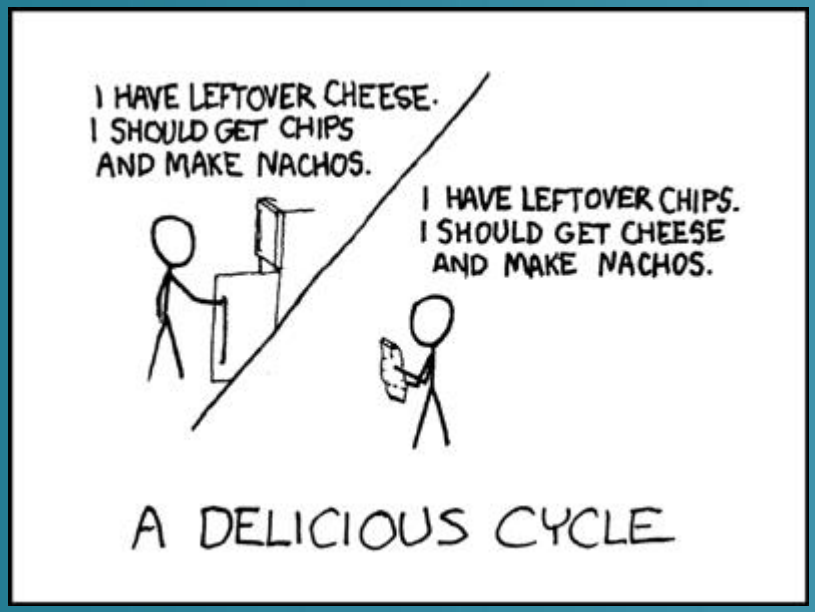
- git cola
  - GUI Git tool
  - Helpful for splitting code into multiple patches
- Meld
  - Visual diff
  - Great for comparing BUILD=KERNEL driver to upstream driver

ADVANTAGES

- Still extra work but not double - more is automated
- Harder to miss a patch because every patch is generated for you
- Can diff BUILD=KERNEL and upstream to see difference (hopefully none)
- Have a good starting point to automate more going forward

PROBLEMS THAT STILL EXIST

- Hard to catch up if we fall behind
  - Becomes vicious circle
- Currently still need a human
  - Merge conflicts still regularly occur
  - Not everyone runs upstream checks



Delicious - https://xkcd.com/140/

PROBLEMS THAT STILL EXIST

- Pulling patches down in the other direction
  - Currently we still rely on developers to pull a patch down from upstream into the Out-Of-Tree driver
  - Need to automate this because patches get missed
- Still need validation resources for every patch

FUTURE WORK – AUTOMATE!

- Unit testing on every patch submitted to the Out-Of-Tree driver
  - Checkpatch.pl
  - Sparse check
  - Upstream compile check
- Add a squash flag that can be recognized by the script and automatically squash anything necessary
- During nightly builds for the Out-Of-Tree driver, run kernelpatch.sh with the email option

FUTURE WORK – AUTOMATE!

- Eventually, get to the point where we can automatically apply the patches
  - send the maintainer an email if there is a conflict
  - send the patches out for review if there are no conflicts
- Extend the new process to other drivers

HELPFUL LINKS

- Intel-wired-lan mailing list

https://lists.osuosl.org/mailman/listinfo/intel-wired-lan

- Intel-wired-lan patchwork

https://patchwork.ozlabs.org/project/intel-wired-lan/list/

- Jeff Kirsher's net tree

https://git.kernel.org/cgit/linux/kernel/git/jkirsher/net-queue.git/log/?h=dev-queue

- Jeff Kirsher's next tree

https://git.kernel.org/cgit/linux/kernel/git/jkirsher/next-queue.git/log/?h=dev-queue

Maintaining two drivers simultaneously doesn't have to mean doubling the workload – automation is our friend.