I Still Think We Have a Scaling Problem

Wolfram Sang

Consultant / Renesas Upstream Kernel Team

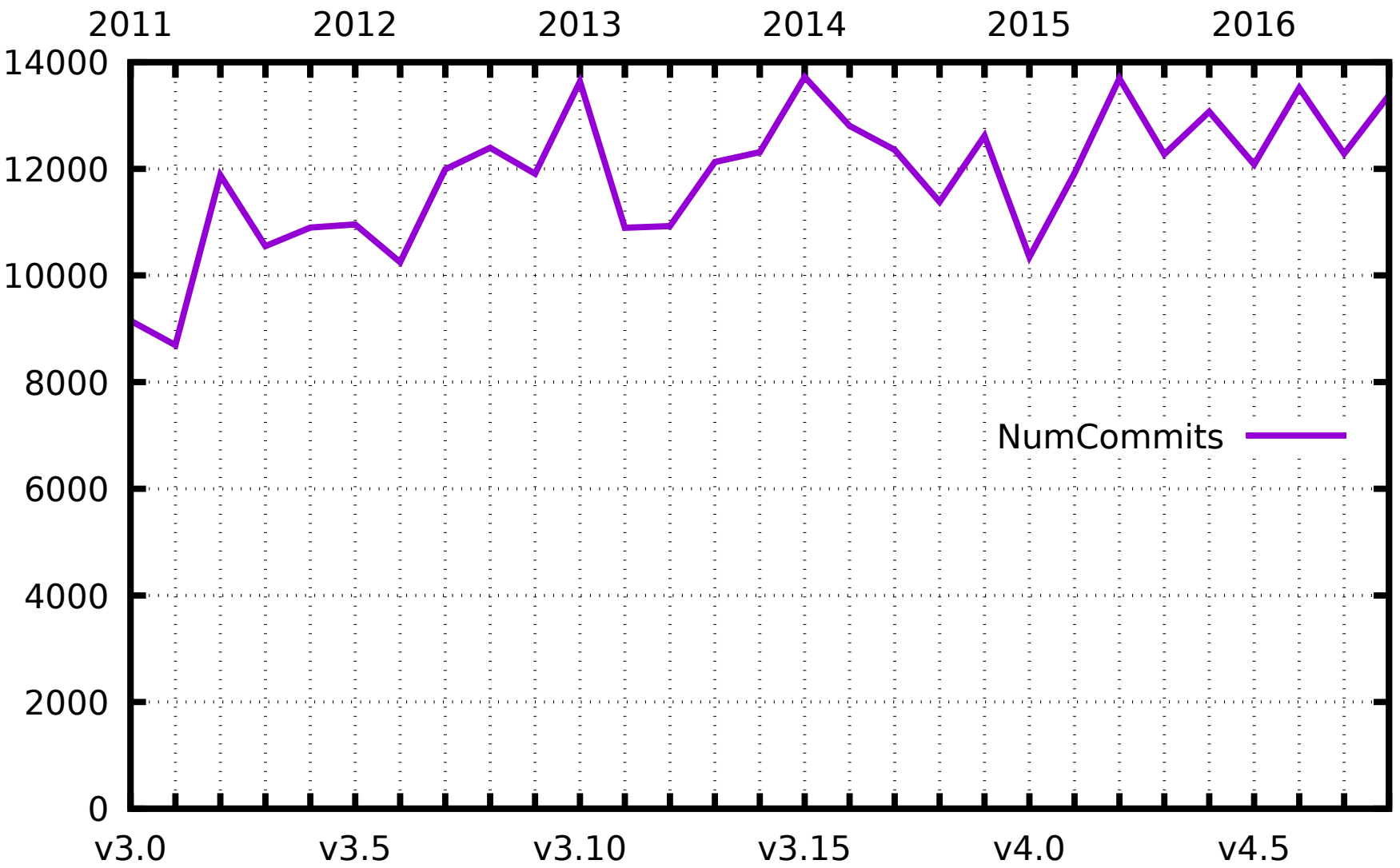
5.10.2016, LinuxCon Europe 2016

I dare to disagree

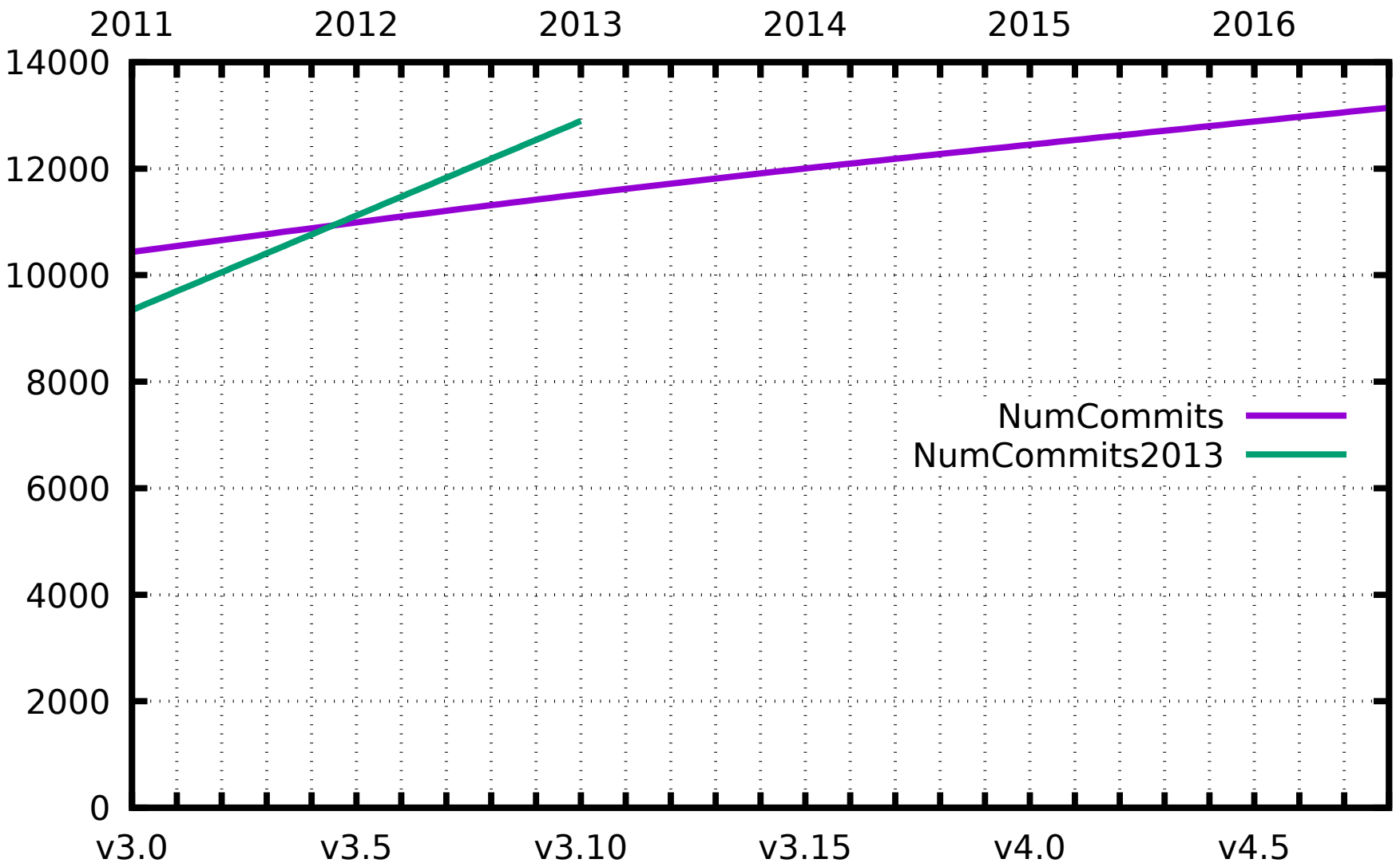
LWN on May 11th, 2016

Quote:
"The overall picture ... is one of a development process that continues to function like a relatively well-tuned machine. The number of contributors continues to increase, the patch flow is steady, and there do not appear to be many process-scalability issues in sight."

Statistics: # of patches



Statistics: # of patches (linearized)



Additional info

- Merges not counted

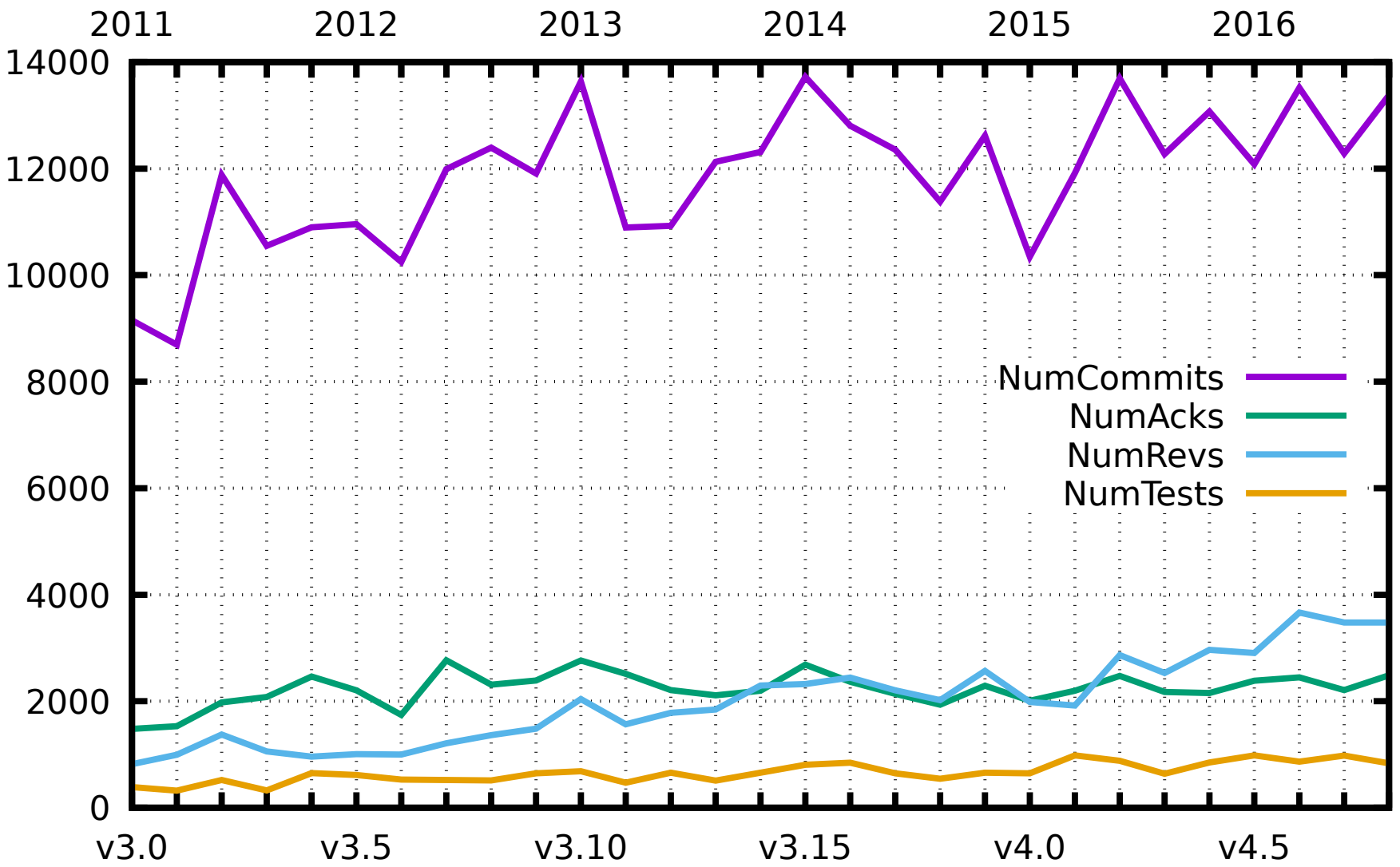
  But they are work, too

- Stats only based on accepted patches

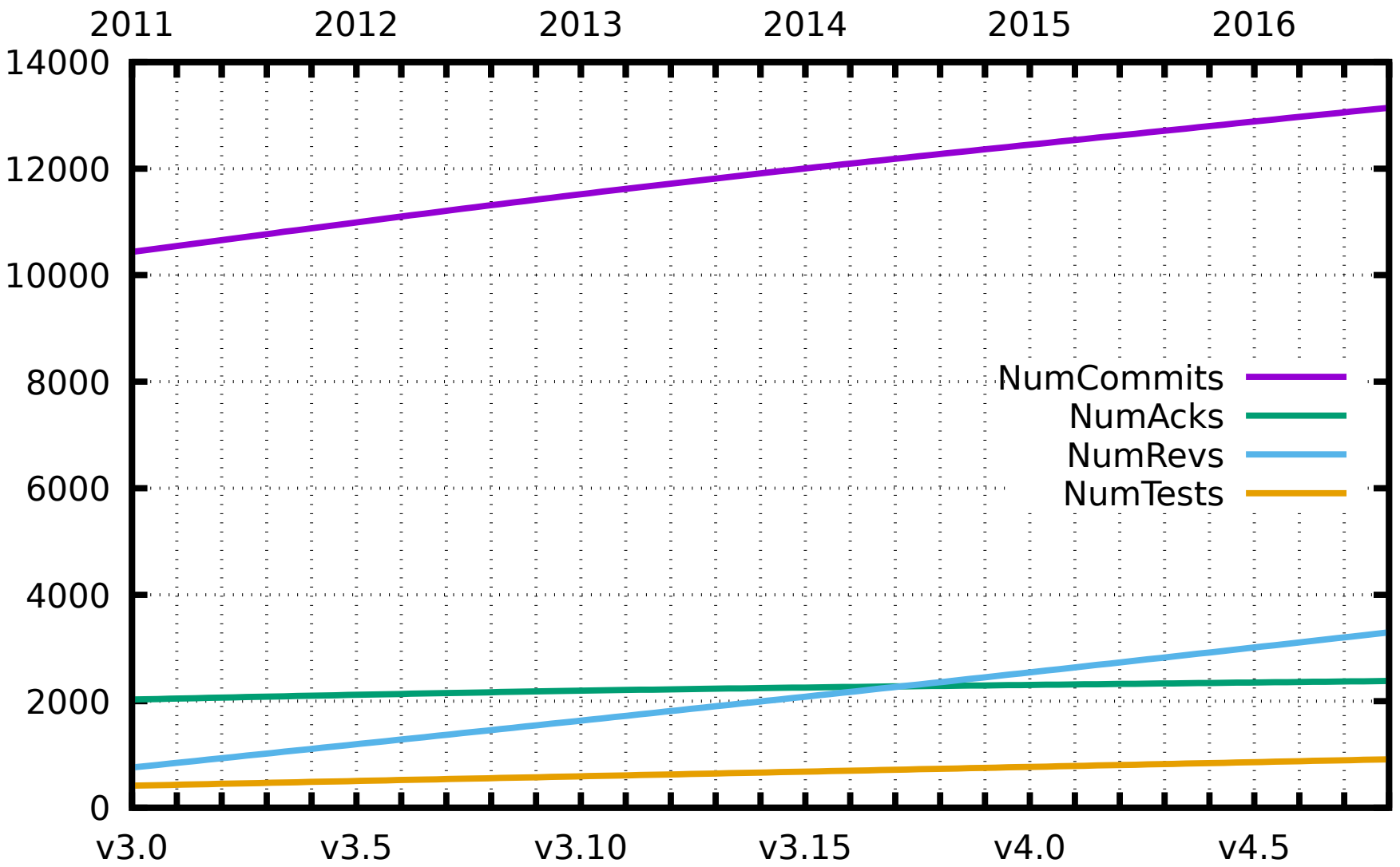
  There are also superseded and rejected patches, teaching new authors...

- Situation at v3.0 was already far from ideal

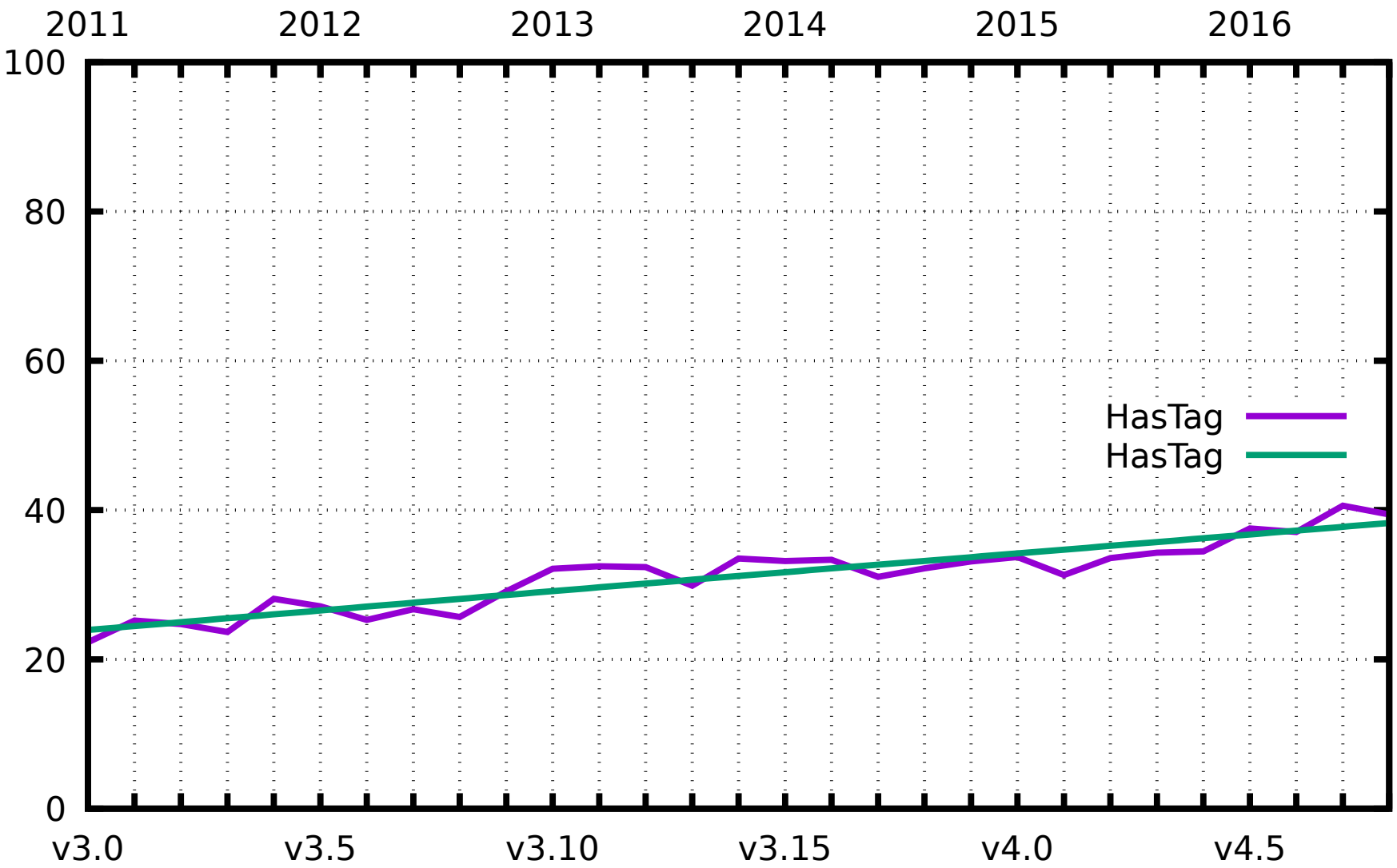
Statistics: # of tags



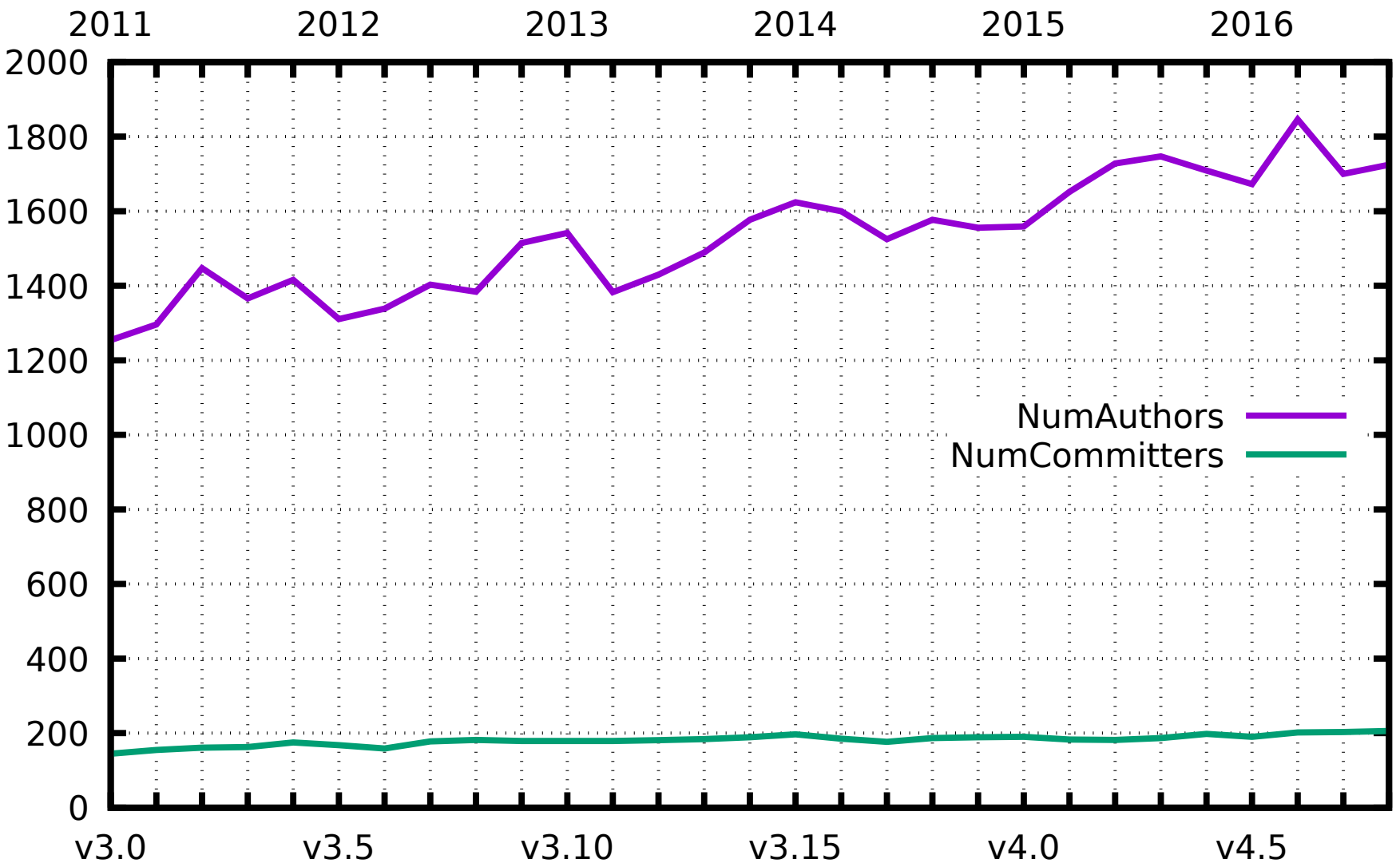
Statistics: # of tags (linearized)



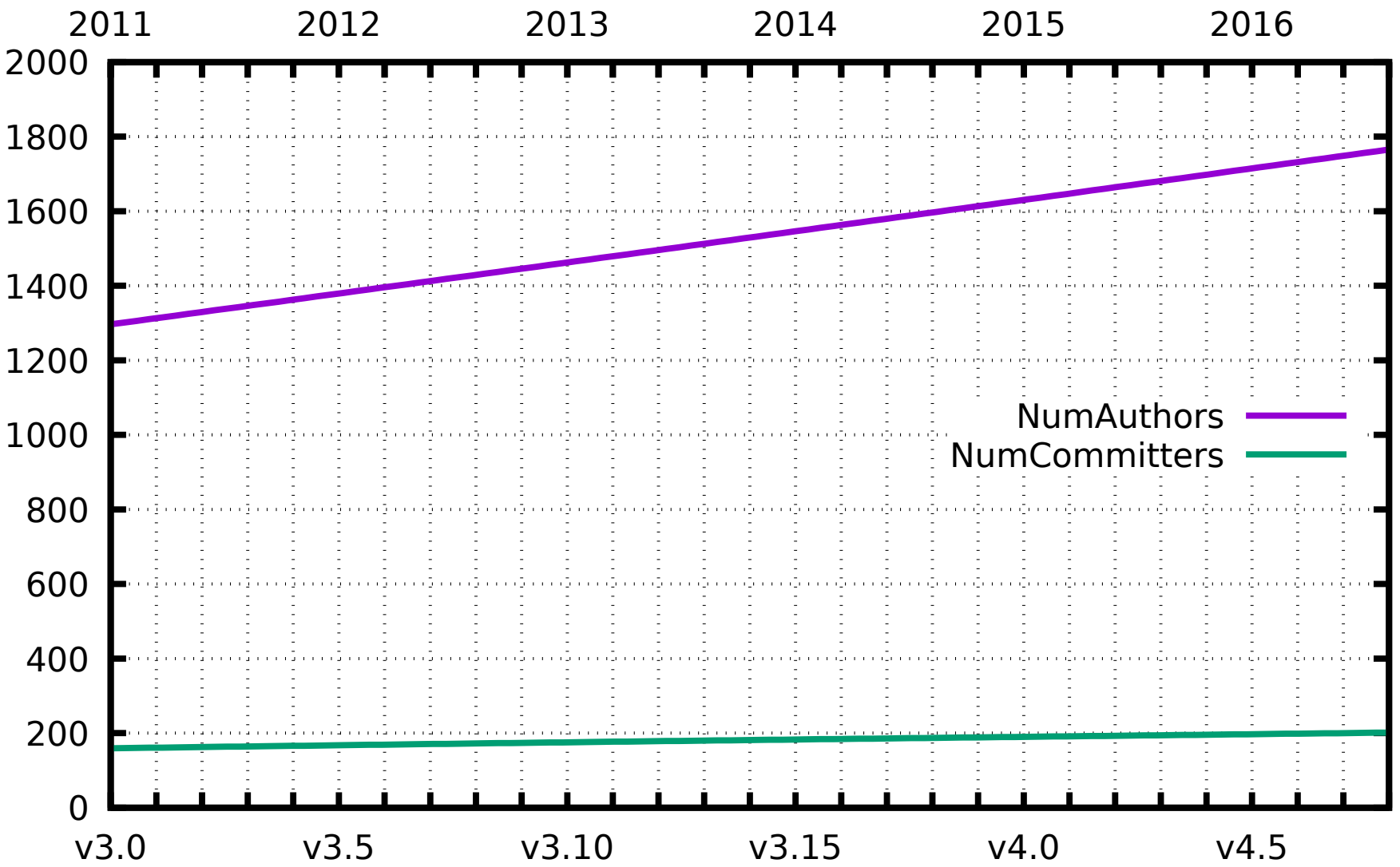
Statistics: commits with tags (percentage)



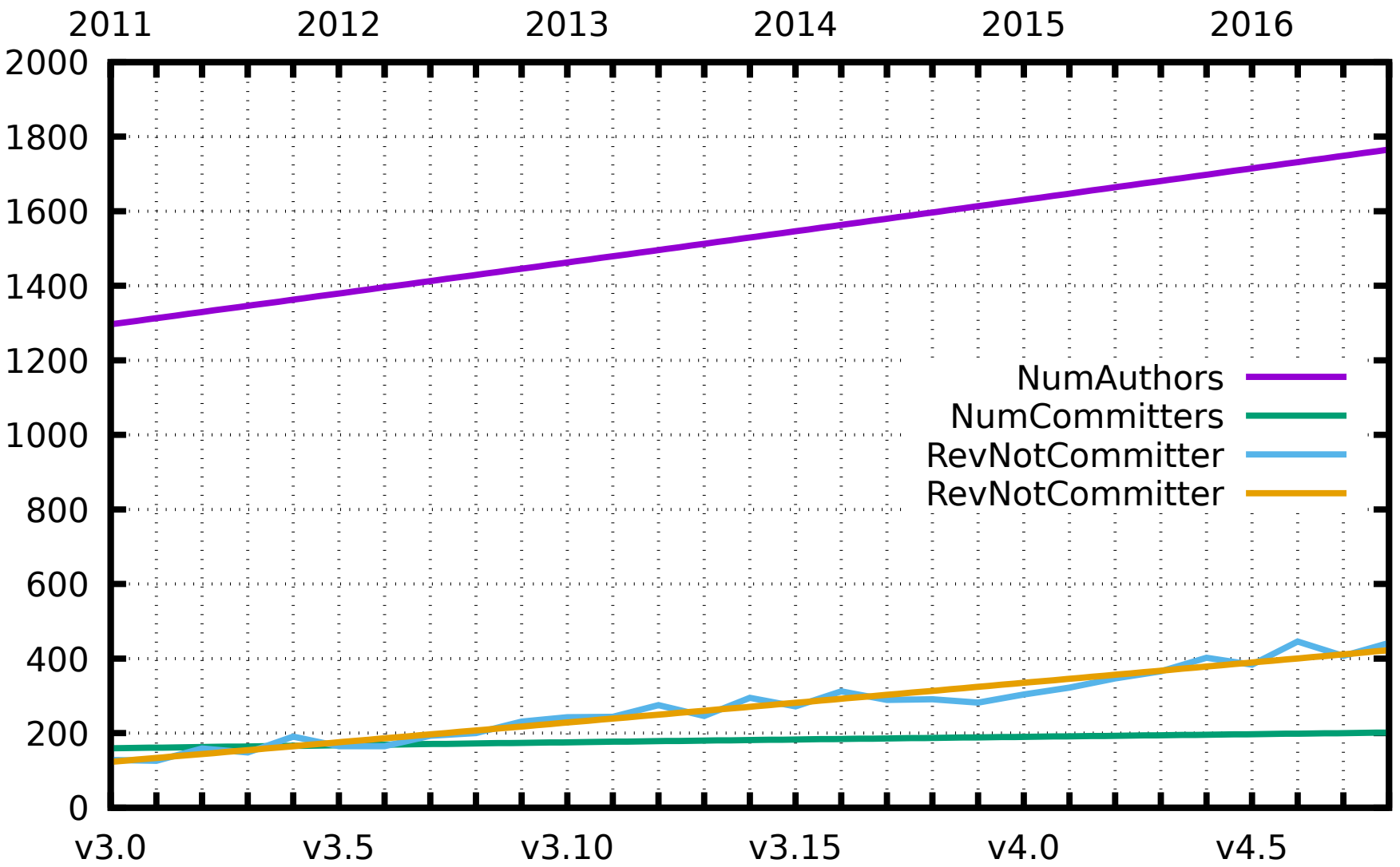
Statistics: # of people



Statistics: # of people (linearized)



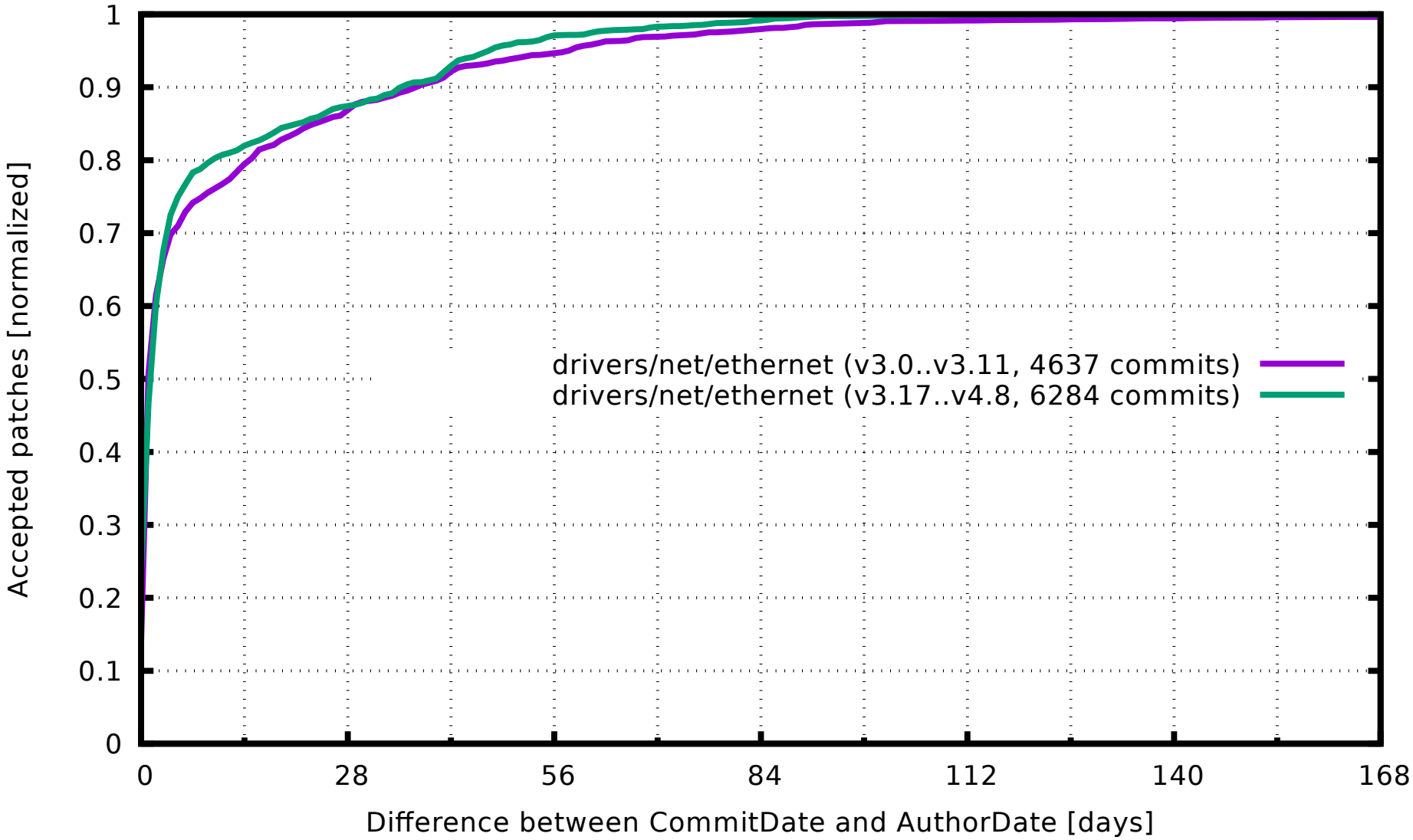
Statistics: more # of people (mostly linearized)



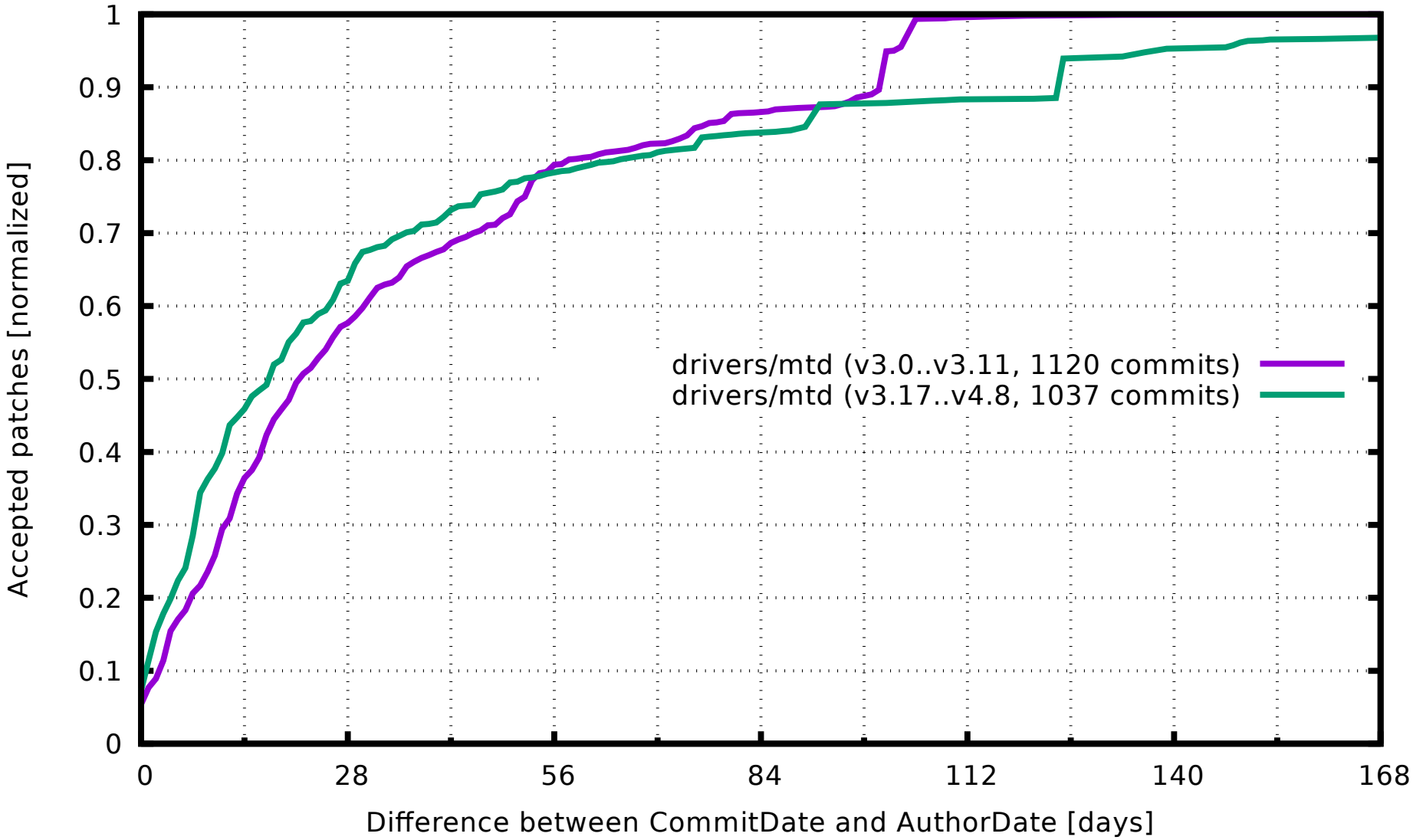
Other metrics

So, how did the latency go?

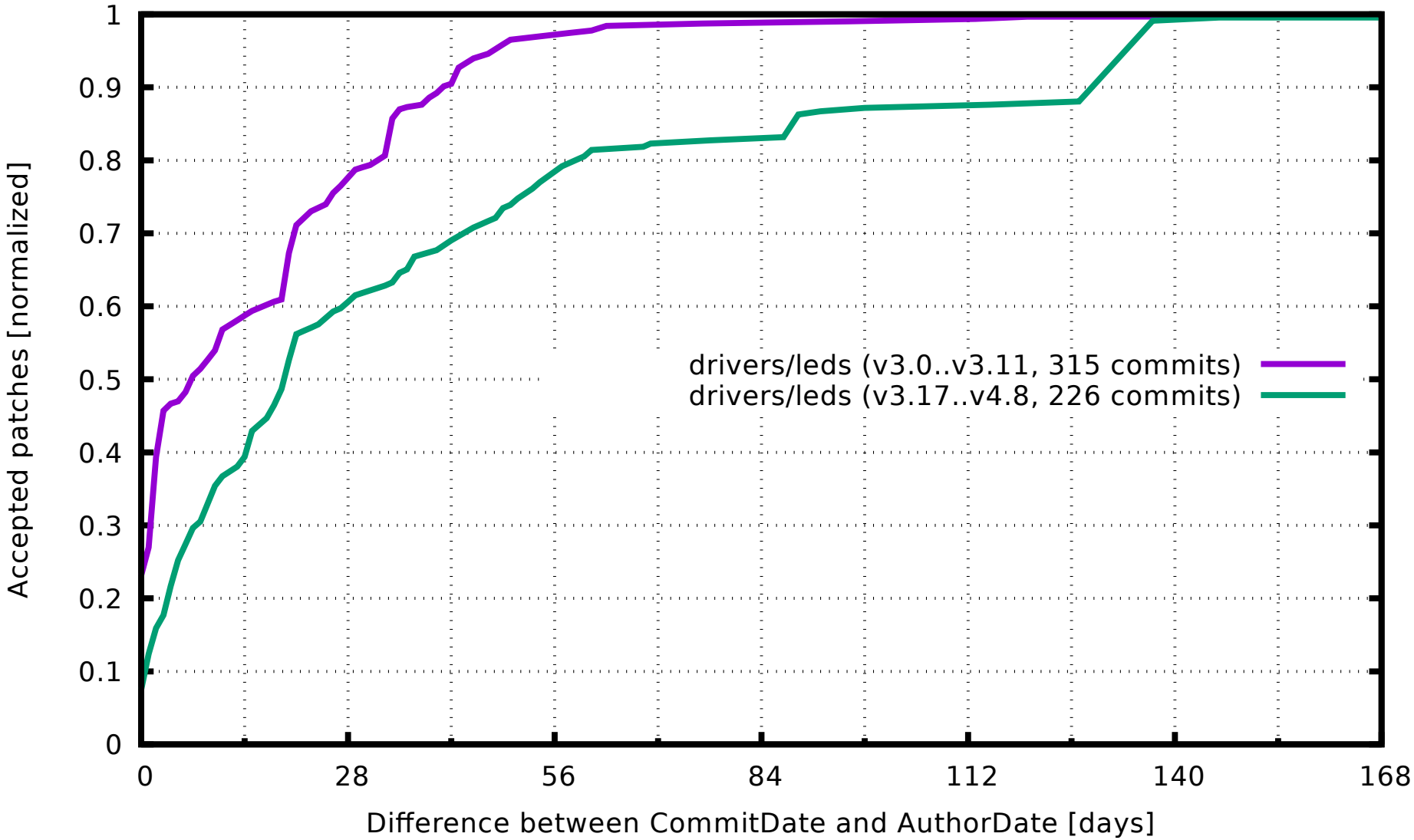
Subsystem Latency: Heroes!



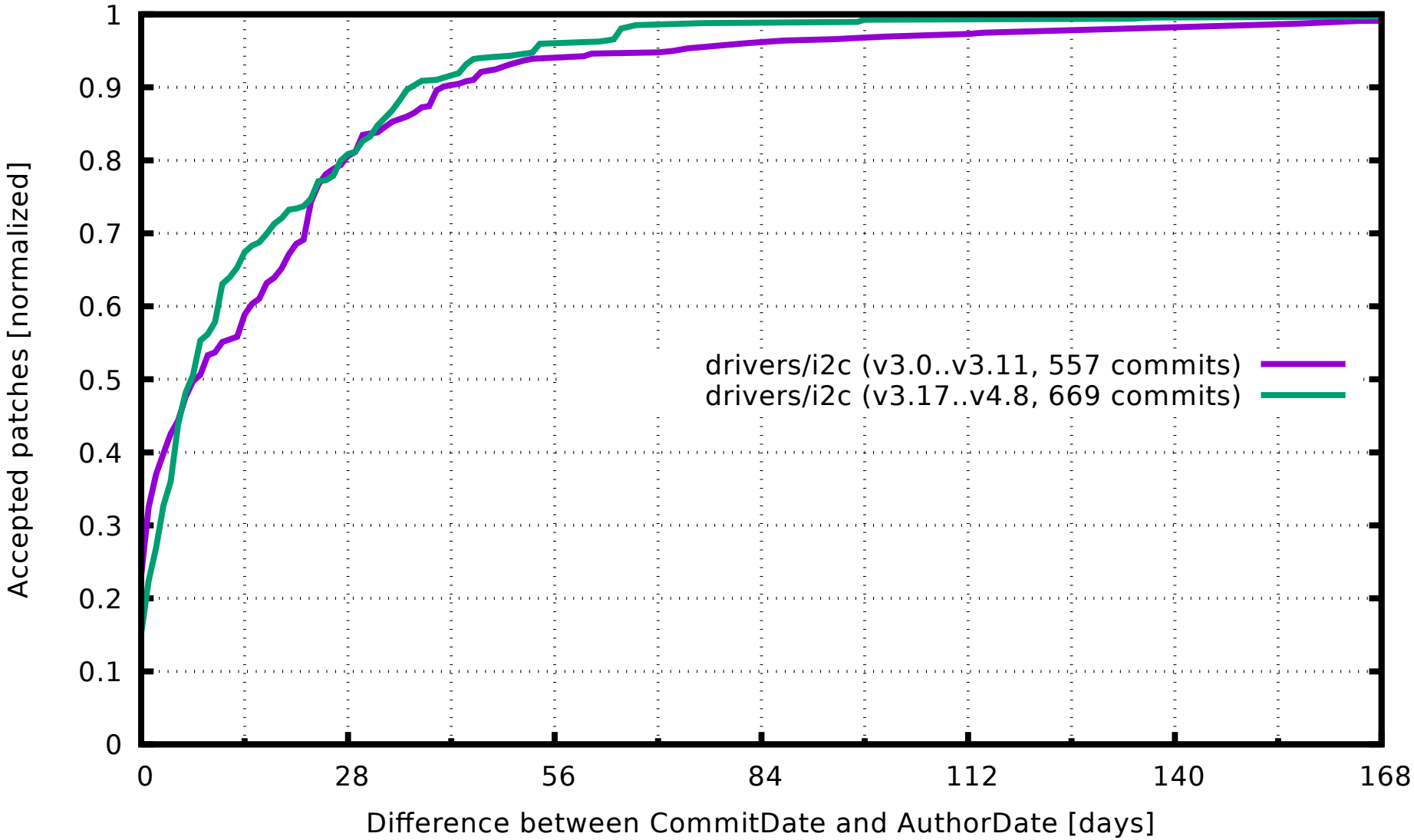
Subsystem Latency: Gains...



Subsystem Latency: ...and losses



Subsystem Latency: My turn...Wut?



Updating the slide from 2013

- Again: Stats only based on accepted patches
- There are also totally valid patches which got lost

  Couldn't find a way to measure that, though
- Weather forecast: Situation will get worse

  Expect either increased latency or questionable patches coming in

Updating the slide from 2013

- Again: Stats only based on accepted patches

- There are also totally valid patches which got lost

  Couldn't find a way to measure that, though

Updating the slide from 2013

- Doesn't take versioning into account

  V1 may have waited for months before a review, then V2 is applied within days

- Again: Stats only based on accepted patches

- There are also totally valid patches which got lost

  Couldn't find a way to measure that, though

Updating the slide from 2013

- Doesn't take versioning into account

  V1 may have waited for months before a review, then V2 is applied within days

- Again: Stats only based on accepted patches

  Idea: extend Patchwork to provde data for all processed patches

- There are also totally valid patches which got lost

  Couldn't find a way to measure that, though

Updating the slide from 2013

- Doesn't take versioning into account

  V1 may have waited for months before a review, then V2 is applied within days

- Again: Stats only based on accepted patches

  Idea: extend Patchwork to provde data for all processed patches

- There are also totally valid patches which got lost

  ~~Couldn't find a way to measure that, though~~

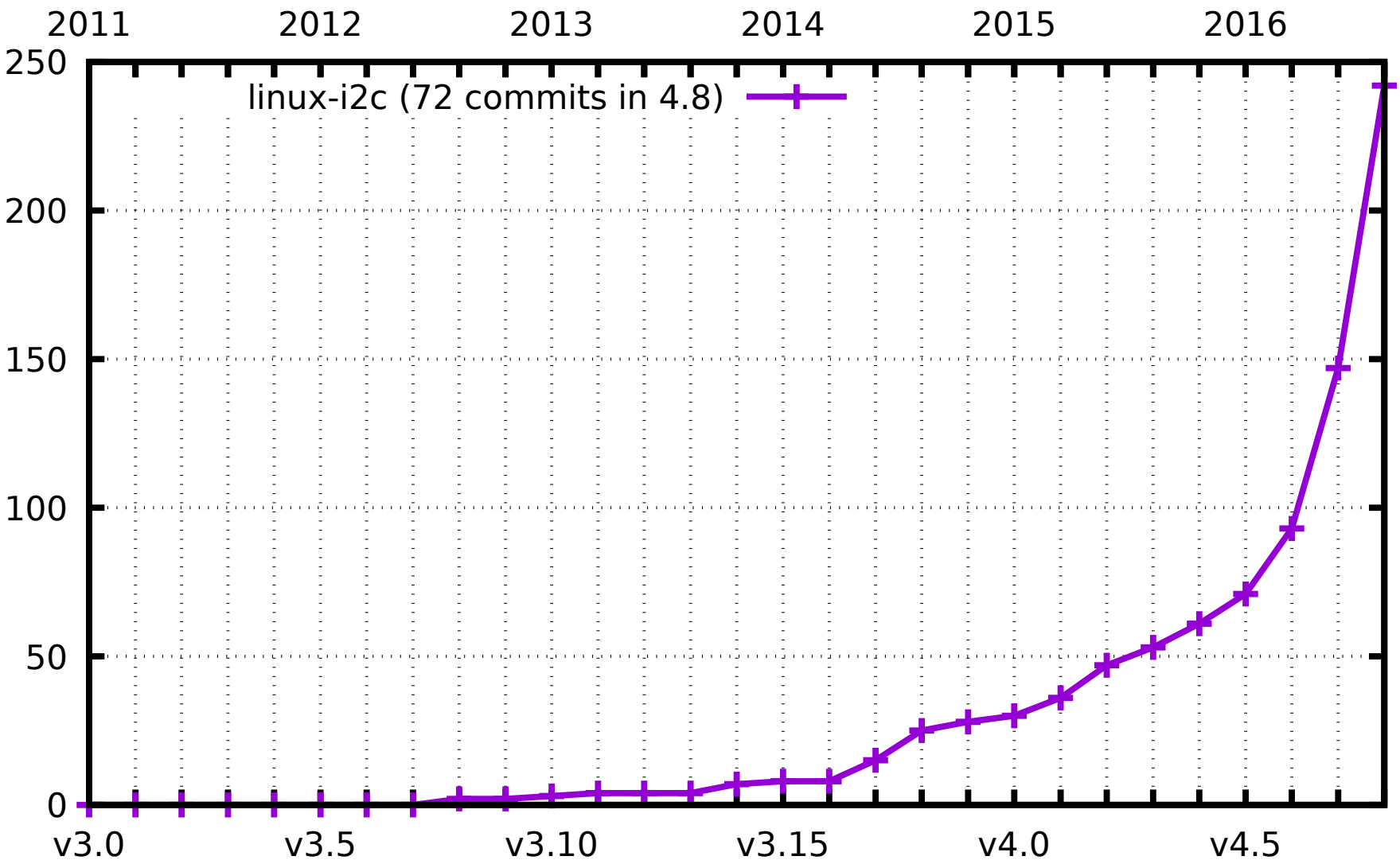
Trying a new metric

Unprocessed patches

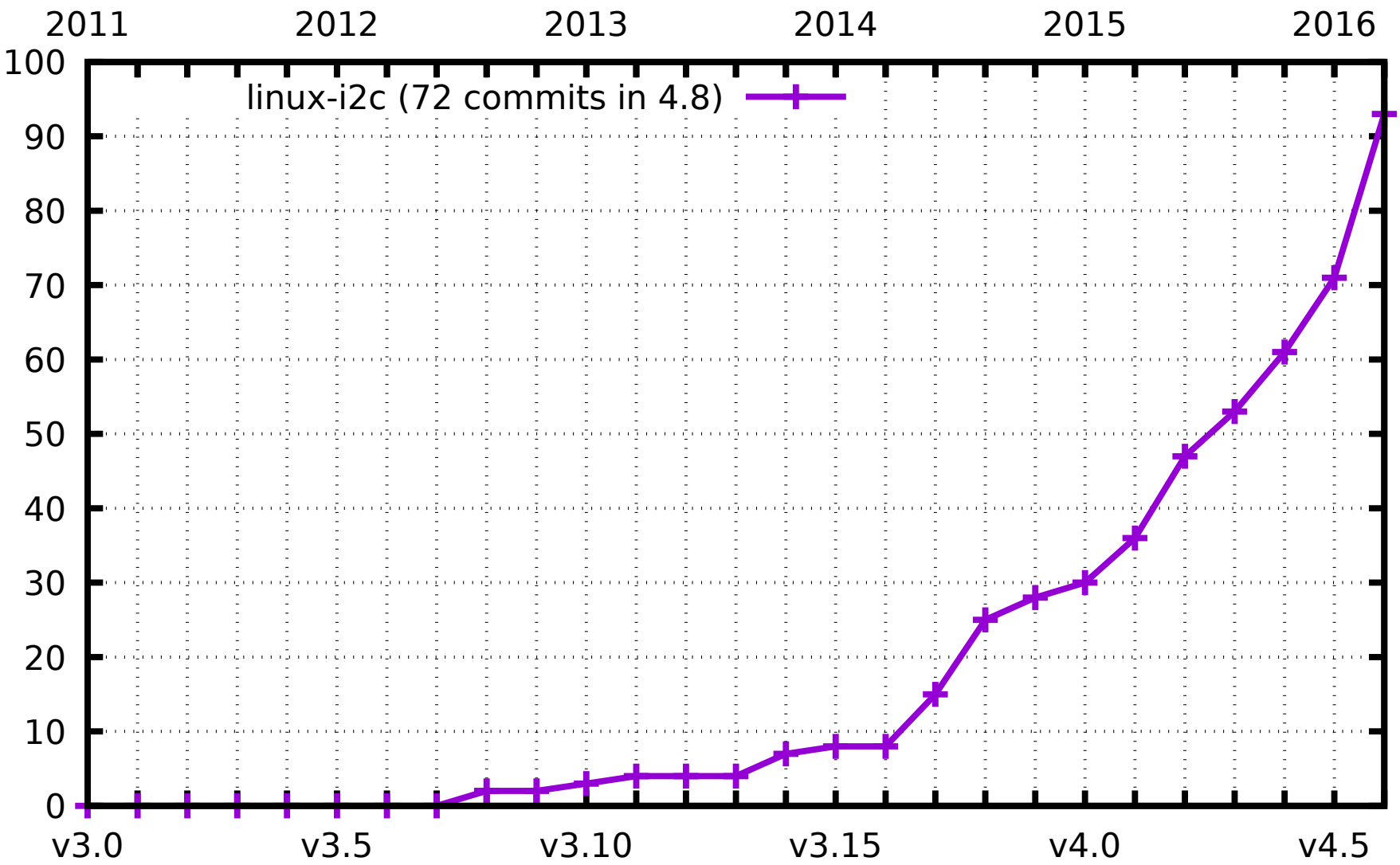
- check subsystems which use Patchwork actively

  using all tags like Rejected, Superseded, etc regularly
- retrieve all dates of patches marked as New
- map them to Kernel versions

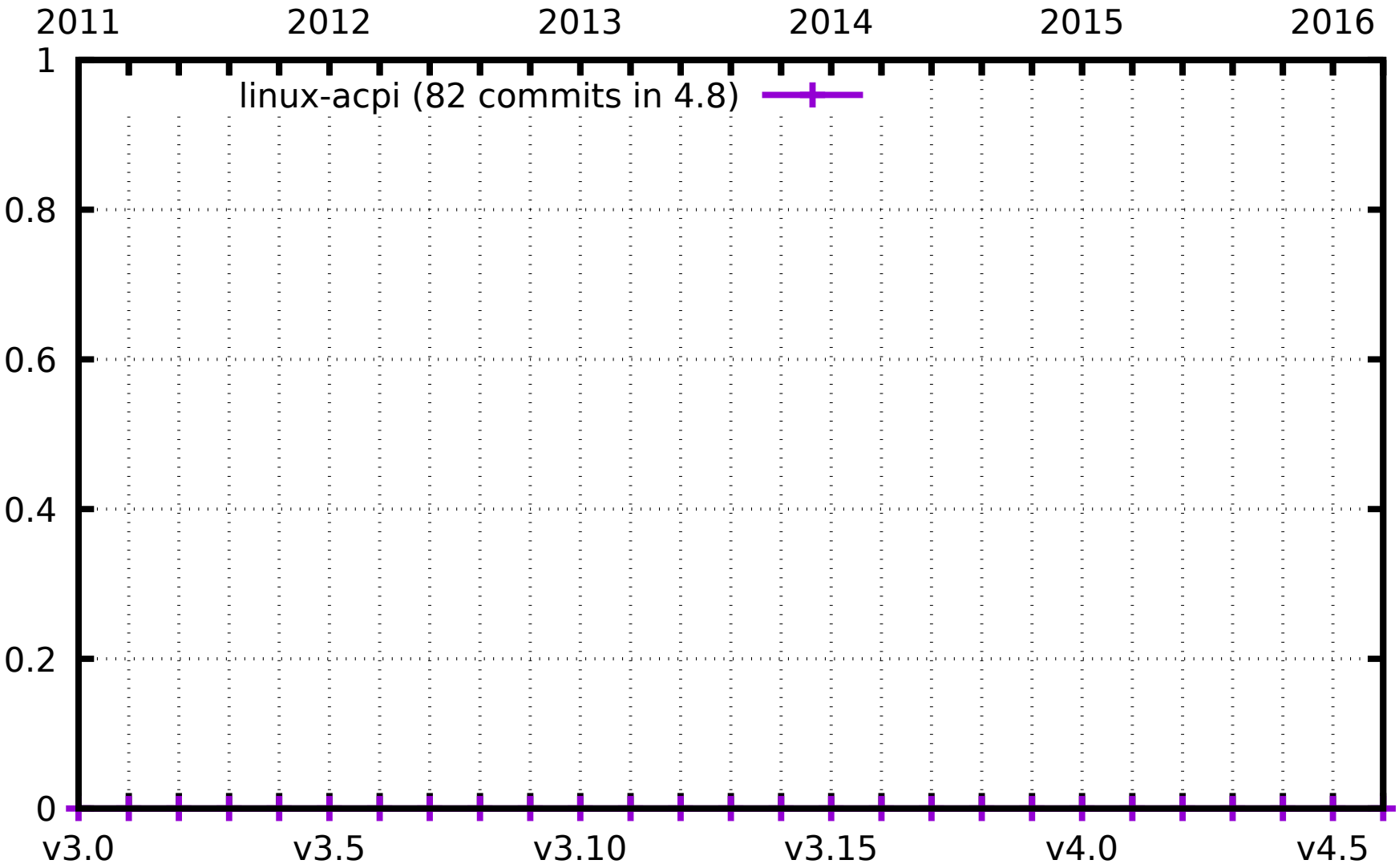
Unprocessed patches (up to now)



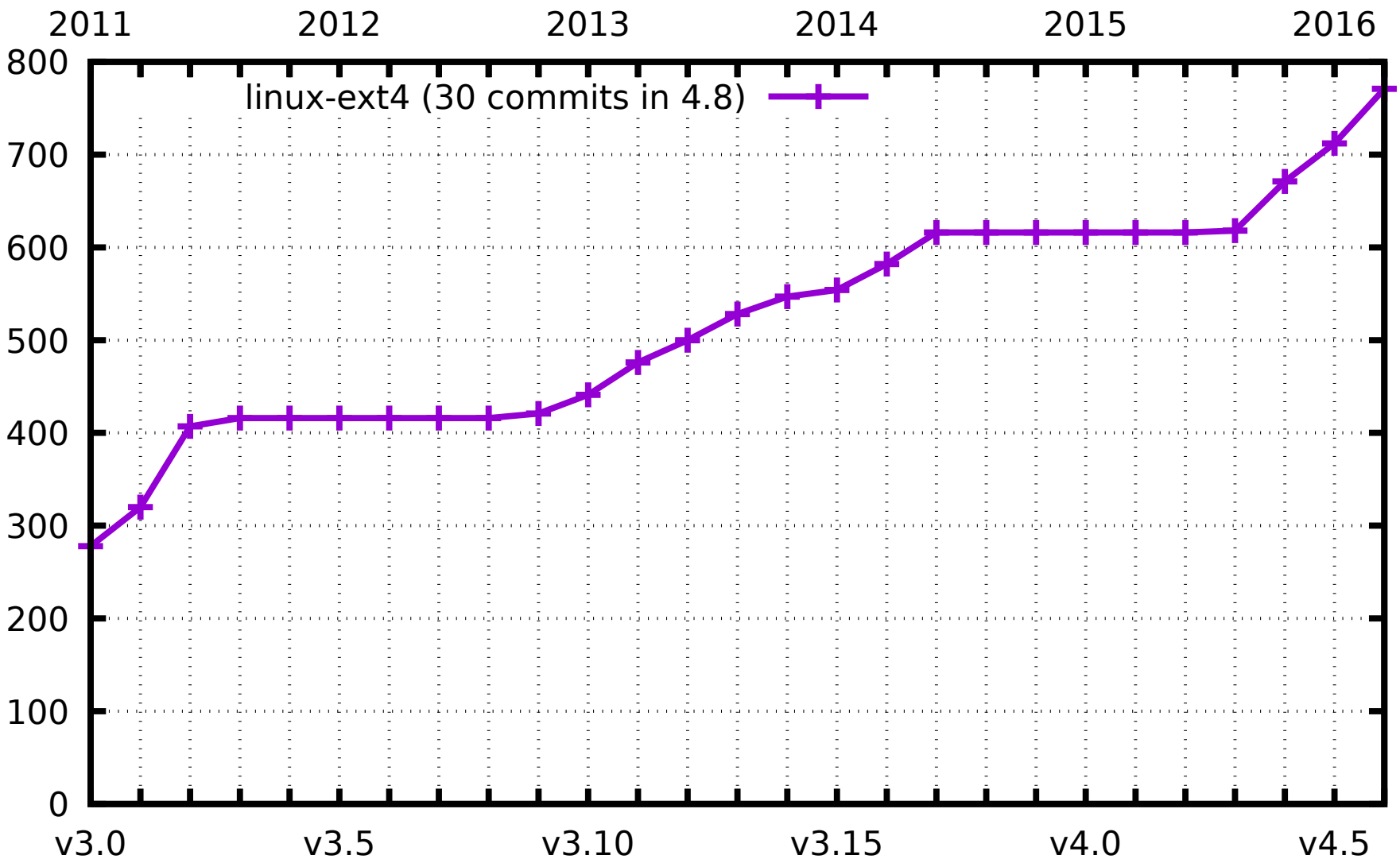
Unprocessed patches (skipping last cycles)



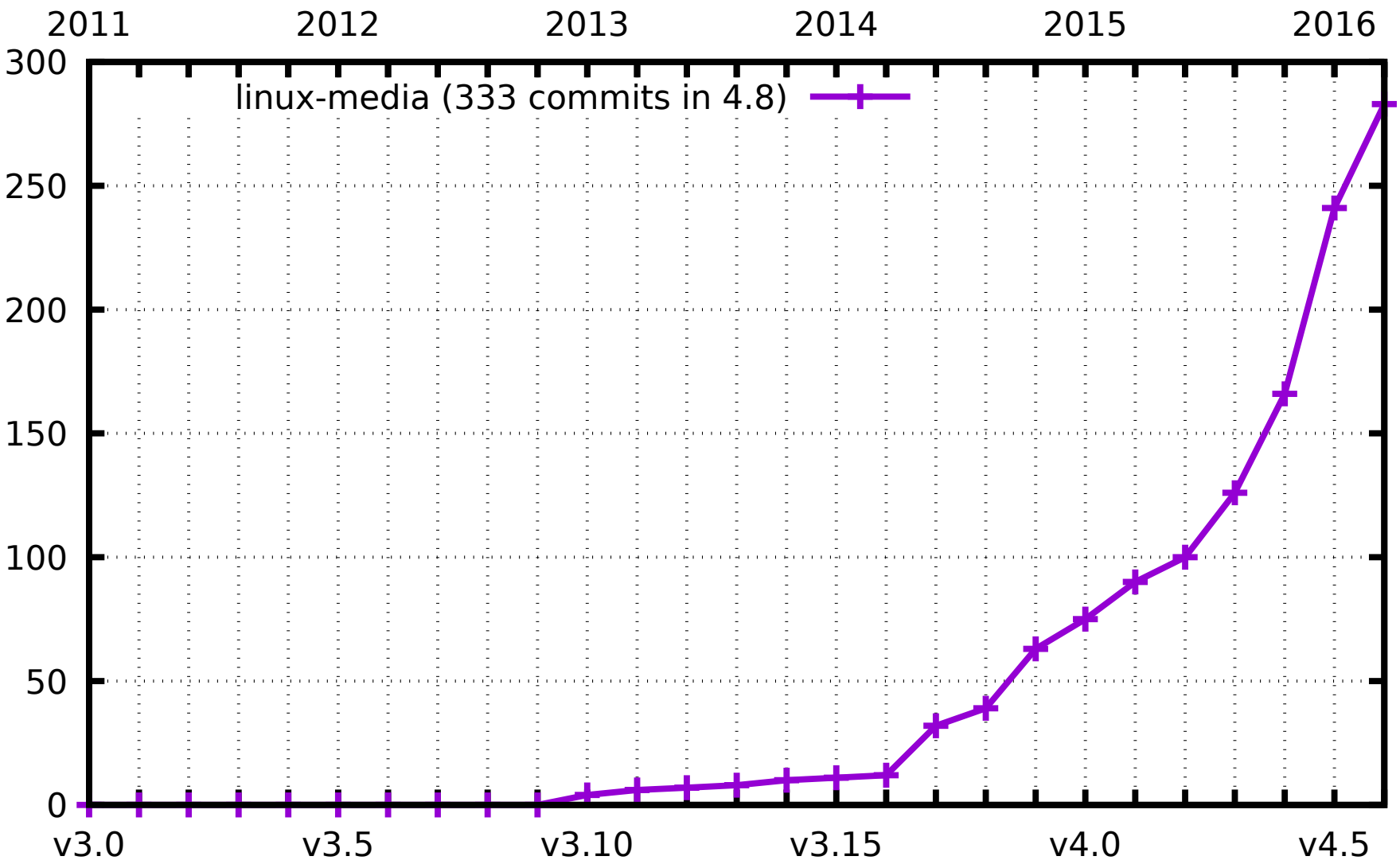
Unprocessed patches: Heroes!



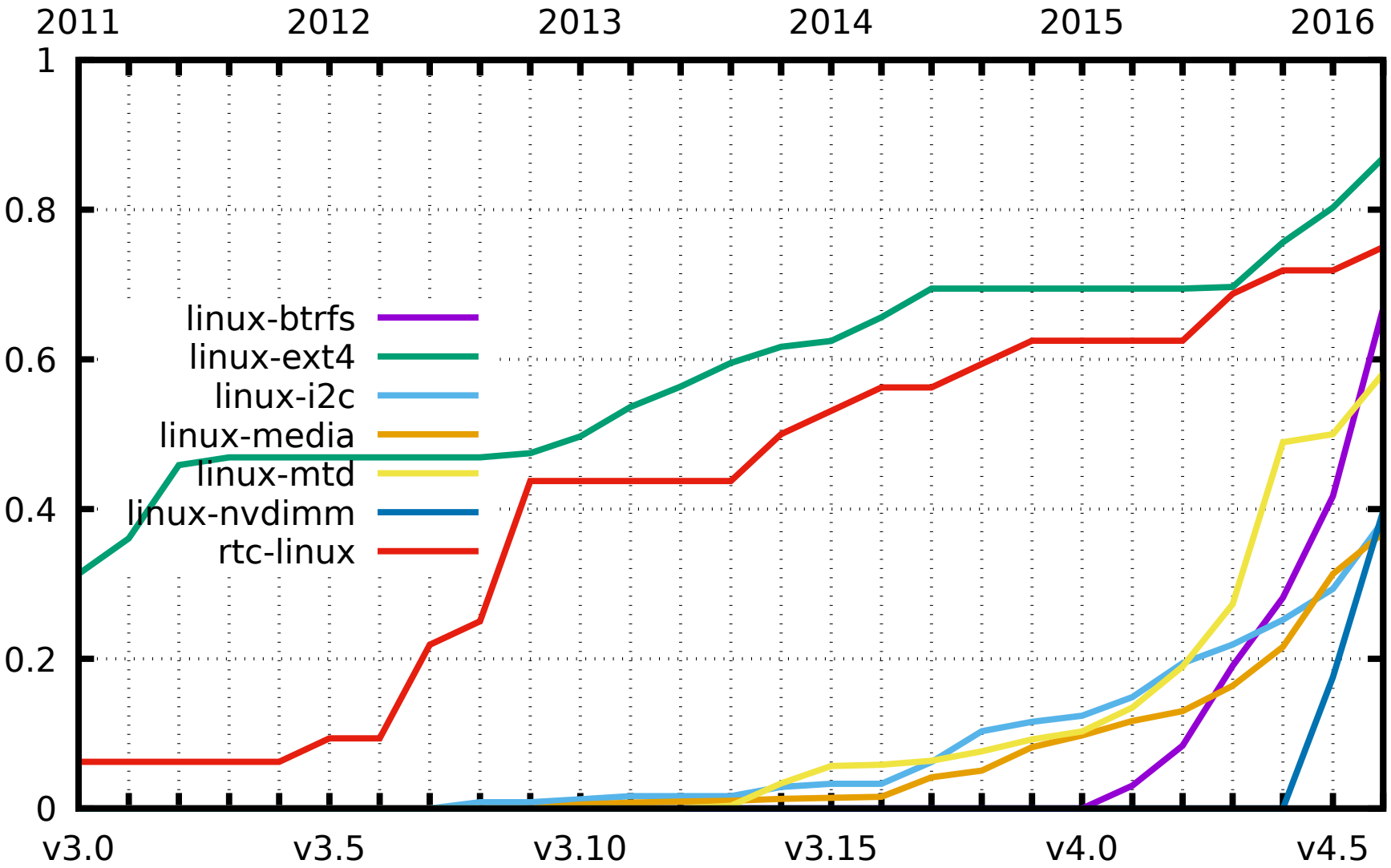
Unprocessed patches: my expectation of "normal"



Unprocessed patches: seems to be the pattern



Unprocessed patches (normalized): a trend?



So, we do have problems, or?

First conclusions

- Number of committers doesn't scale at all
- Number of reviewers doesn't scale enough
- Good news: Latency of accepted patches did not increase generally
- Bad news: This is only half of the truth. number of unprocessed patches significantly increased
- extend Patchwork to provide more data

What you can do: Users[1]

Give feedback

- give comments about patches

  show interest, tell about issues, ...

- give tags

  Tested-by! Very important one, no need to be a coder for that

[1] as in "users of patches"

What you can do: Developers

Always give your best shot

- missing experience is no problem
- sloppiness *is* a problem
- be honest, give reasons if you did suboptimal solutions

take part in reviewing

- review
- review your own patches!
- if you are interested, become a maintainer

What you can do: Maintainers

Work harder...*not!*

- we don't need burnout
- be aware of the fast-forward button

have your tools ready

- yeah, workflow is highly subjective...
- ...still check with other maintainers what is out there
- My top three recommendations:
  keyboard shortcuts, git hooks, patchwork
- automize tasks

What you can do: Organizations

allow developers to take part in reviewing

- some really want to if allowed
- those will improve their skills a lot
- next talk will point out the advantages of having maintainers in-house

educate internally

- OK to ask maintainer for subsystem specific help
- not OK to let maintainers teach basic stuff

  I'll push back immediately if I know you have people inhouse which can do the same

I don't want a KII

Remember: It was not an action, it was a *reaction*.

Changes to I2C

What is a maintainer? Trond Myklebust said it nicely[2]:

"Currently, the Linux maintainer appears to be responsible for filling all of the traditional roles of:

- software architect
- software developer
- patch reviewer
- patch committer
- and software maintainer

."

[2] link here

Changes to I2C

What is a maintainer[3] in the future?

- ~~software architect~~
  one of the software architects
- ~~software developer~~
  one of the software developers
- ~~patch reviewer~~
  one of the patch reviewers
- patch committer
- software maintainer
- (new focus!) advertiser for distributed community efforts

[3] well, at least the I2C maintainer

Changes to I2C

Disadvantages

- expect bigger latencies

Advantages

- I keep sane

  ...because I am neither a machine nor part of a machine

- can spend more time fixing this issue on higer levels

Thank you for your attention!

Let's fix that together!

pretty please?

Questions?

- Right here, right now...
- Later at the conference
- wsa@the-dreams.de