netdev FAQ

What is netdev?

It is a mailing list for all network-related Linux stuff. This includes anything found under net/ (i.e. core code like IPv6) and drivers/net (i.e. hardware specific drivers) in the Linux source tree.

Note that some subsystems (e.g. wireless drivers) which have a high volume of traffic have their own specific mailing lists.

The netdev list is managed (like many other Linux mailing lists) through VGER (http://vger.kernel.org/) with archives available at https://lore.kernel.org/netdev/

Aside from subsystems like those mentioned above, all network-related Linux development (i.e. RFC, review, comments, etc.) takes place on netdev.

How do the changes posted to netdev make their way into Linux?

There are always two trees (git repositories) in play. Both are driven by David Miller, the main network maintainer. There is the `net` tree, and the `net-next` tree. As you can probably guess from the names, the `net` tree is for fixes to existing code already in the mainline tree from Linus, and `net-next` is where the new code goes for the future release. You can find the trees here:

- https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net.git
- https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git

How do I indicate which tree (net vs. net-next) my patch should be in?

To help maintainers and CI bots you should explicitly mark which tree your patch is targeting. Assuming that you use git, use the prefix flag:

```
git format-patch --subject-prefix='PATCH net-next' start..finish
```

Use `net` instead of `net-next` (always lower case) in the above for bug-fix `net` content.

How often do changes from these trees make it to the mainline Linus tree?

To understand this, you need to know a bit of background information on the cadence of Linux development. Each new release starts off with a two week "merge window" where the main maintainers feed their new stuff to Linus for merging into the mainline tree. After the two weeks, the merge window is closed, and it is called/tagged `-rc1`. No new features get mainlined after this -- only fixes to the rc1 content are expected. After roughly a week of collecting fixes to the rc1 content, rc2 is released. This repeats on a roughly weekly basis until rc7 (typically; sometimes rc6 if things are quiet, or rc8 if things are in a state of churn), and a week after the last vX.Y-rcN was done, the official vX.Y is released.

Relating that to netdev: At the beginning of the 2-week merge window, the `net-next` tree will be closed - no new changes/features. The accumulated new content of the past ~10 weeks will be passed onto mainline/Linus via a pull request for vX.Y -- at the same time, the `net` tree will start accumulating fixes for this pulled content relating to vX.Y

An announcement indicating when `net-next` has been closed is usually sent to netdev, but knowing the above, you can predict that in advance.

Warning

Do not send new `net-next` content to netdev during the period during which `net-next` tree is closed.

RFC patches sent for review only are obviously welcome at any time (use `--subject-prefix='RFC net-next'` with `git format-patch`).

Shortly after the two weeks have passed (and vX.Y-rc1 is released), the tree for `net-next` reopens to collect content for the next (vX.Y+1) release.

If you aren't subscribed to netdev and/or are simply unsure if `net-next` has re-opened yet, simply check the `net-next` git repository link above for any new networking-related commits. You may also check the following website for the current status:

> http://vger.kernel.org/~davem/net-next.html

The `net` tree continues to collect fixes for the vX.Y content, and is fed back to Linus at regular (~weekly) intervals. Meaning that the focus for `net` is on stabilization and bug fixes.

Finally, the vX.Y gets released, and the whole cycle starts over.

So where are we now in this cycle?

So where are we now in this cycle?

Load the mainline (Linus) page here:

> https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git

and note the top of the "tags" section. If it is rc1, it is early in the dev cycle. If it was tagged rc7 a week ago, then a release is probably imminent. If the most recent tag is a final release tag (without an `-rcN` suffix) - we are most likely in a merge window and `net-next` is closed.

How can I tell the status of a patch I've sent?

Start by looking at the main patchworks queue for netdev:

> https://patchwork.kernel.org/project/netdevbpf/list/

The "State" field will tell you exactly where things are at with your patch. Patches are indexed by the `Message-ID` header of the emails which carried them so if you have trouble finding your patch append the value of `Message-ID` to the URL above.

How long before my patch is accepted?

Generally speaking, the patches get triaged quickly (in less than 48h). But be patient, if your patch is active in patchwork (i.e. it's listed on the project's patch list) the chances it was missed are close to zero. Asking the maintainer for status updates on your patch is a good way to ensure your patch is ignored or pushed to the bottom of the priority list.

Should I directly update patchwork state of my own patches?

It may be tempting to help the maintainers and update the state of your own patches when you post a new version or spot a bug. Please do not do that. Interfering with the patch status on patchwork will only cause confusion. Leave it to the maintainer to figure out what is the most recent and current version that should be applied. If there is any doubt, the maintainer will reply and ask what should be done.

I made changes to only a few patches in a patch series should I resend only those changed?

No, please resend the entire patch series and make sure you do number your patches such that it is clear this is the latest and greatest set of patches that can be applied.

I have received review feedback, when should I post a revised version of the patches?

Allow at least 24 hours to pass between postings. This will ensure reviewers from all geographical locations have a chance to chime in. Do not wait too long (weeks) between postings either as it will make it harder for reviewers to recall all the context.

Make sure you address all the feedback in your new posting. Do not post a new version of the code if the discussion about the previous version is still ongoing, unless directly instructed by a reviewer.

I submitted multiple versions of a patch series and it looks like a version other than the last one has been accepted, what should I do?

There is no revert possible, once it is pushed out, it stays like that. Please send incremental versions on top of what has been merged in order to fix the patches the way they would look like if your latest patch series was to be merged.

Are there special rules regarding stable submissions on netdev?

While it used to be the case that netdev submissions were not supposed to carry explicit `CC: stable@vger.kernel.org` tags that is no longer the case today. Please follow the standard stable rules in :ref:`Documentation/process/stable-kernel-rules.rst <stable_kernel_rules>`, and make sure you include appropriate Fixes tags!

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\process\(linux-master) (Documentation) (process)maintainer-netdev.rst`, line 165); *backlink*

Unknown interpreted text role "ref".

Is the comment style convention different for the networking content?

Yes, in a largely trivial way. Instead of this:

```
/*
 * foobar blah blah blah
 * another line of text
 */
```

it is requested that you make it look like this:

```
/* foobar blah blah blah
 * another line of text
 */
```

I am working in existing code which uses non-standard formatting. Which formatting should I use?

Make your code follow the most recent guidelines, so that eventually all code in the domain of netdev is in the preferred format.

I found a bug that might have possible security implications or similar. Should I mail the main netdev maintainer off-list?

No. The current netdev maintainer has consistently requested that people use the mailing lists and not reach out directly. If you aren't OK with that, then perhaps consider mailing security@kernel.org or reading about http://oss-security.openwall.org/wiki/mailing-lists/distros as possible alternative mechanisms.

What level of testing is expected before I submit my change?

At the very minimum your changes must survive an `allyesconfig` and an `allmodconfig` build with `W=1` set without new warnings or failures.

Ideally you will have done run-time testing specific to your change, and the patch series contains a set of kernel selftest for `tools/testing/selftests/net` or using the KUnit framework.

You are expected to test your changes on top of the relevant networking tree (`net` or `net-next`) and not e.g. a stable tree or `linux-next`.

How do I post corresponding changes to user space components?

User space code exercising kernel features should be posted alongside kernel patches. This gives reviewers a chance to see how any new interface is used and how well it works.

When user space tools reside in the kernel repo itself all changes should generally come as one series. If series becomes too large or the user space project is not reviewed on netdev include a link to a public repo where user space patches can be seen.

In case user space tooling lives in a separate repository but is reviewed on netdev (e.g. patches to `iproute2` tools) kernel and user space patches should form separate series (threads) when posted to the mailing list, e.g.:

```
[PATCH net-next 0/3] net: some feature cover letter
 â””â”€ [PATCH net-next 1/3] net: some feature prep
 â””â”€ [PATCH net-next 2/3] net: some feature do it
 â””â”€ [PATCH net-next 3/3] selftest: net: some feature

[PATCH iproute2-next] ip: add support for some feature
```

Posting as one thread is discouraged because it confuses patchwork (as of patchwork 2.2.2).

Can I reproduce the checks from patchwork on my local machine?

Checks in patchwork are mostly simple wrappers around existing kernel scripts, the sources are available at:

https://github.com/kuba-moo/nipa/tree/master/tests

Running all the builds and checks locally is a pain, can I post my patches and have the patchwork bot validate them?

No, you must ensure that your patches are ready by testing them locally before posting to the mailing list. The patchwork build bot instance gets overloaded very easily and netdev@vger really doesn't need more traffic if we can help it.

netdevsim is great, can I extend it for my out-of-tree tests?

No, `netdevsim` is a test vehicle solely for upstream tests. (Please add your tests under `tools/testing/selftests/`.)

We also give no guarantees that `netdevsim` won't change in the future in a way which would break what would normally be considered uAPI.

Is netdevsim considered a "user" of an API?

Linux kernel has a long standing rule that no API should be added unless it has a real, in-tree user. Mock-ups and tests based on `netdevsim` are strongly encouraged when adding new APIs, but `netdevsim` in itself is **not** considered a use case/user.

Any other tips to help ensure my net/net-next patch gets OK'd?

Attention to detail. Re-read your own work as if you were the reviewer. You can start with using `checkpatch.pl`, perhaps even with the `--strict` flag. But do not be mindlessly robotic in doing so. If your change is a bug fix, make sure your commit log indicates the end-user visible symptom, the underlying reason as to why it happens, and then if necessary, explain why the fix proposed is the best way to get things done. Don't mangle whitespace, and as is common, don't mis-indent function arguments that span multiple lines. If it is your first patch, mail it to yourself so you can test apply it to an unpatched tree to confirm infrastructure didn't mangle it.

Finally, go back and read :ref:`Documentation/process/submitting-patches.rst <submittingpatches>` to be sure you are not repeating some common mistake documented there.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\process\(linux-master) (Documentation) (process)maintainer-netdev.rst`, line 283); *backlink*

Unknown interpreted text role "ref".