Contributing to ns-3

Tom Henderson

NS-3 Annual Meeting Tutorials

June 22, 2021

Contributing vs. publishing your code

- It is good research practice to publish your code so that others may reproduce your results
  - Leaving behind enough artifacts and scripts so that your figures can be reproduced exactly
  - Code quality is sometimes a barrier to author's willingness to publish code
- Contributing to ns-3 is about making ns-3 better
  - Make it easier for both future users and maintainers

Mainline vs. app store

- In general, the project prefers to add new modules to the app store, unless they are considered to be of mainstream interest

- App store code does not have to undergo code review or meet the contribution guidelines of the mainline

Maintainers' view

Maintainers checklist

- Is the feature or patch potentially useful?
- Is the code clean and easy to read?
- Does it follow coding style guidelines and prevailing style?
- Are examples, tests, and documentation complete enough?
- Are there Python bindings issues?
- Are there any potential valgrind warnings?
- What version of compilers will be required?
- Are third-party libraries required?
- Does it slow or improve performance?
- Does it break existing users' code, out-of-tree code, or change behavior?
- Is the submitter responsive to make requested changes?

Administrative issues

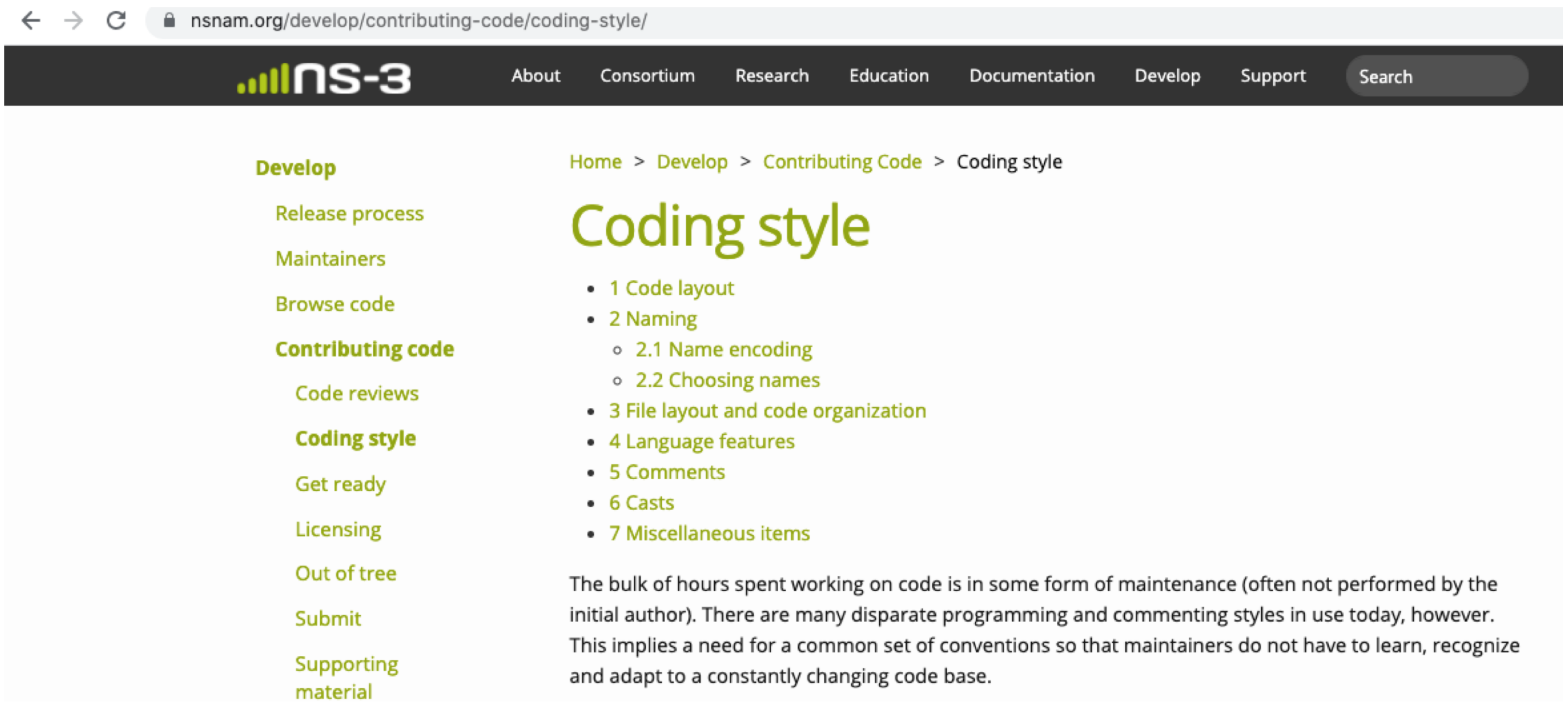
- Licensing:
  - GPLv2-compatible for code
  - CC-BY-SA 4.0 for documentation
- Copyright:
  - Copyright is not transferred to the ns-3 project
  - Copyrights should not be deleted when code is copied; copyright statements may be added
- Avoid author and funding agency attribution inline in the source code
- Adding to the Authors list of a file for a small change is generally avoided

Administrative issues (cont.)

- Please edit your .gitconfig.user when you are developing
- Commit strings usually have the following syntax:
  - First line fits in 80 columns
  - Module name followed by colon
  - Issue or MR # in parentheses (if applicable)
  - Terse comment (larger comments can be added as needed)

Coding style

- In general, make your new code look like existing code
- Avoid changing whitespace on parts of the code that do not pertain to your patch



Coding style

- 1 Code layout
- 2 Naming
  - 2.1 Name encoding
  - 2.2 Choosing names
- 3 File layout and code organization
- 4 Language features
- 5 Comments
- 6 Casts
- 7 Miscellaneous items

The bulk of hours spent working on code is in some form of maintenance (often not performed by the initial author). There are many disparate programming and commenting styles in use today, however. This implies a need for a common set of conventions so that maintainers do not have to learn, recognize and adapt to a constantly changing code base.

Git workflow

- Users should fork from GitLab.com and generate Merge Requests to the mainline
- Mainline keeps one ‘master’ branch with a nearly linear history
- Releases are small tagged branches
- Try to squash commits along feature and authorship boundaries
- We will rebase your commits if you do not do so, to avoid merge commits
  - Fast forward merge with rebase

App store considerations

- Module name selection
- Icon for main page
- Versioning and releases
- Default download
- Code review on ns-3-contrib-reviews