Network
Internet-Draft
Intended status: Best Current Practice
Expires: August 19, 2019

M. Thomson
Mozilla

February 15, 2019

Using GitHub at the IETF
draft-thomson-git-using-github-00

Abstract

This document describes best practices for working groups that use GitHub for their work.

Note to Readers

Discussion of this document takes place on the GitHub@ietf mailing list (ietf-and-github@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search?email_list=ietf-and-github [1].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 19, 2019.

Table of Contents

1. Introduction 3
1.1. Distributed Version Control Systems 3
1.2. GitHub 3
1.3. Other Services 4
1.4. Document Goals 4
1.5. Notational Conventions 5
2. Administrative Policies 5
2.1. Organizations 5
2.2. Backup and Archive Requirements 6
2.3. Communicating Policies 6
2.3.1. Contribution Policies on Repositories 6
3. Deciding to Use GitHub 7
3.1. What to Use GitHub For 7
3.2. Working Group Policies 7
3.3. Repositories 8
3.4. Editors and Contributors 8
3.5. Document Formats 9
4. Contribution Methods 9
4.1. Issues 9
4.1.1. Issue Labelling 10
4.1.2. Closing Issues 10
4.2. Pull Requests 10
4.2.1. Discussion on Pull Requests 11
4.2.2. Merging Pull Requests 11
4.3. Monitoring Activity 11
5. Internet-Draft Publication 12
6. Assessing Consensus 12
7. Continuous Integration 13
8. Advice to Editors 13
9. GitHub Limitations 14
10. Security Considerations 14
11. IANA Considerations 15
12. References 15
12.1. Normative References 15
12.2. Informative References 15
12.3. URIs 16
Appendix A. Experiences from Working Groups 16
A.1. CORE 16
A.2. QUIC 17
Appendix B. Acknowledgments 19
Author's Address 19

1. Introduction

The IETF has an open and transparent process for developing standards. The use of GitHub or similar tools, when used as part of this process, can have several objectives. GitHub provides tools that can be helpful in editing documents. Use of this service has proven to reduce the time that working groups need to produce documents and to improve the quality of the final result.

The use of source control improves traceability and visibility of changes. Issue tracking can be used to manage open issues and provide a record of their resolution. Pull requests allow for better engagement on technical and edditorial changes, and encourage contributions from a larger set of contributors. Using GitHub can also broaden the community of contributors for a specification.

This document describes how the IETF uses GitHub through the development of Internet-Drafts. This concentrates on the work that occurs within IETF working groups. Recommendations for working groups and their chairs are made for integrating these tools with their processes.

This document is meant as an enhancement to RFC 2418 [RFC2418]. It provides guidance to working group chairs and participants on how they can best use GitHub. The small number of rules in this document are there to ensure common usage patterns between working groups and to avoid issues that have been encountered in the past.

A companion document, [GH-CONFIG], describes administrative processes that supports the practices described in this document.

1.1. Distributed Version Control Systems

Different version control systems are a critical component of software engineering and are quite useful also for document editing.

Git is a distributed version control system . Each instance of a repository contains a number of revisions. Each revision stores the complete state of a set of files. Users are able to create new revisions in their copy of a repository and share revisions between copies of repositories.

1.2. GitHub

GitHub is a service operated at https://github.com/ [2]. GitHub provides a centralized store for git repositories. GitHub is freely accessible on the open Internet (see Section 9), albeit currently only via IPv4.

GitHub provides a simplified and integrated interface to not only git, but also provides basic user management, an issue tracker, associated wiki, project hosting, and other features.

There are a large number of projects at GitHub and a very large community of contributors. One way in which some IETF Working Groups have seen benefit is in the increased reviews and associated issues and improvements that come from broader participation by facilitating those in this community to participate.

1.3. Other Services

Git is not the only version control system available, nor is GitHub the only possible choice for hosting. There are other services that host revision control repositories and provide similar additional features to GitHub. For instance, BitBucket [3], or GitLab [4] provide a similar feature set. In additional to a hosted service, software for custom installations exists.

This document concentrates primarily on GitHub as it has a large and active community of contributors. As a result, some content might not be applicable to other similar services. A working group that decides to adopt an alternative tool or service can still benefit from the general guidance in this document.

1.4. Document Goals

This document aims to describe how a working group might best apply GitHub to their work. The intent is to allow each working group considerable flexibility in how they use GitHub.

This document does require that policies for use of GitHub are agreed and clearly communicated within the working group (see Section 2). The remainder of the document contains guidelines and advice on how to construct a workable policy.

The requirements here apply to the case where working groups decide to use GitHub as a primary means of interaction. Individuals can set their own policies when using GitHub for managing their own drafts, or for managing drafts that they edit on behalf of a working group that has not explicitly adopted GitHub.

For both sets of users, this document aims to provide some amount of advice on practices that have proven to be effective.

1.5. Notational Conventions

The words "MUST", "MUST NOT", "SHOULD", and "MAY" are used in this document. It's not shouting; when they are capitalized, they have the special meaning defined in [RFC2119].

2. Administrative Policies

The following administrative rules provide the necessary oversight and transparency.

2.1. Organizations

Organizations are a way of forming groups of contributors on GitHub. Each Working Group SHOULD create a new organization for the working group.

A working group organization SHOULD be named consistently so that it can be found. For instance, the name could be ietf-<wgname> or ietf-<wgname>-wg.

A single organization SHOULD NOT be used for all IETF activity, or all activity within an area. Large organizations create too much overhead for general management tasks, particularly when there is a need to maintain membership.

Each organization requires owners. The owner team for a working group repository MUST include responsible Area Directors. Area Directors MAY also designate a delegate that becomes an owner and working group chairs MAY also be owners.

A team with administrator access SHOULD be created for the Working Group Chairs and any Working Group Secretary. Administrator access is preferable, since this does not also include the ability to push to all repositories and ownership does not grant any other significant privileges.

When Area Directors or Working Group Chairs change, teams MUST be updated to reflect the new membership status.

When a Working Group is closed, the responsible Area Director is responsible for removing existing members from teams in the organization. Repositories MUST be updated along to indicate that they are no longer under development.

2.2. Backup and Archive Requirements

When an IETF Working Group is closed or when associated mailing lists are closed, mail archives and datatracker information from that work is backed up and accessible. The same applies to GitHub repositories.

Any repositories including issues and discussion SHOULD be backed up on IETF resources. It is desirable for those to be accessible via the Working Group's datatracker page. For example, this might be via URLs listed in the More Info section on the Working Group Charter page.

The IETF MAY decide to backup information associated with a Working Group's organization periodically. This decision can be made differently per Working Group in consultation with the responsible Area Director.

2.3. Communicating Policies

Each Working Group MAY set its own policy as to whether and how it uses GitHub or GitLab. It is important that occasional participants in the WG and others accustomed to IETF tools be able to determine this and easily find the policy and GitHub or GitLab organization.

A simple example of how to do this is to include a link to the GitHub organization on the WG Charter page in the datatracker under More Info. Similarly, if there are multiple mailing list options, links to those mailing lists should be given. An example of this is at https://datatracker.ietf.org/wg/quic/charter/.

2.3.1. Contribution Policies on Repositories

One important policy is the IETF IPR policy (see [RFC5378], [RFC3979], and [RFC4879]). Part of this policy requires making contributors aware of the policy.

The IETF Trust license file for open source repositories [5] MUST be included prominently in any document repository.

Including this information in the CONTRIBUTING file is sufficient.

In addition to the boilerplate text there can be a benefit to including pointers to other working group materials, the IETF datatracker, specific drafts, or websites. Adding such text is at the discretion of the Working Group Chairs.

3. Deciding to Use GitHub

A Working Group Chairs are responsible for determining how to best accomplish the Charter in an open and transparent fashion. The Working Group Chairs are responsible for determining if there is interest in using GitHub and making a consensus call to determine if a the proposed policy and use is acceptable.

Chairs SHOULD involve Area Directors in this decision if they intend to use GitHub for anything more than managing of drafts.

While a document editor can still use GitHub independently for documents that they edit, even if the working group does not expressly choose to use GitHub, any such public repository MUST follow the guidelines in Section 2.3.1. This recognizes that editors have traditionally chosen their own methods for managing the documents they edit but preserves the need for transparent contributions with awareness of IPR considerations.

3.1. What to Use GitHub For

Working Group Chairs have to decide what GitHub features the working group will rely upon. Section 4 contains a more thorough discussion on the different features that can be used.

Once a document is published in a repository on GitHub, many features like pull requests, issue tracking or the wiki can be individually disabled. If specific features are not used by the working group in the development of the document, disabling those features avoids creating confusion in the wider community about what can be used.

3.2. Working Group Policies

Working Group Chairs that decide to use GitHub MUST inform their working groups of their decision on the working group mailing list. An email detailing how the working group intends to use GitHub is sufficient, though it might be helpful to occasionally remind new contributors of these guidelines.

Working Group Chairs are responsible for ensuring that any policy they adopt is enforced and maintained.

Updating the README or CONTRIBUTING file in the repository with details of the process ensures that the process is recorded in a stable location other than the mailing list archive. This also makes any working group policies available to casual contributors who might only interact with the GitHub repository.

GitHub prominently links to the CONTRIBUTING on certain pages. This file SHOULD be used in preference to the README for information that new contributors need. A link to the CONTRIBUTING file from the README is advised.

3.3. Repositories

New repositories can be created within the working group organization at the discretion of the chairs. Chairs could decide to only create new repositories for adopted working group items, or they might create repositories for individual documents on request.

All repositories for working group documents MUST be public. Repositories for private documents MAY be kept private, but only where there is a specific reason for doing so. For instance, a document that details a security vulnerability might be kept private prior to its initial publication as an Internet-Draft. Once an Internet-Draft is published, repositories SHOULD be made public.

The adoption status of any document MUST be clear from the contents of the repository. This can be achieved by having the name of the document reflect status (that is, draft-ietf-<wg>-... indicates that the document was adopted), or through a prominent notice (such as in the README).

Experience has shown that maintaining separate repositories for independent documents is most manageable. This allows the work in that repository to be focused on a single item.

Closely related documents, such as those that together address a single milestone, might be placed in a single repository. This allows editors to more easily manage changes and issues that affect multiple documents.

Maintaining multiple documents in the same repository can add overheads that negatively affect individual documents. For instance, issues might require additional markings to identify the document that they affect. Also, because editors all have write access to the repository, managing the set of people with write access to a larger repository is more difficult.

3.4. Editors and Contributors

Working group chairs MUST give document editors write access to document repositories. This can be done by creating teams with write access and allocating editors to those teams, or by making editors collaborators on the repository.

Working group chairs MAY also grant other individuals write access for other reasons, such as maintaining supporting code or build configurations. Working group chairs, as administrators or owners of the organization might also have write access to repositories. Users other than document editors, including chairs, SHOULD NOT write to working group documents unless with prior coordination with document editors.

Working groups MAY create a team for regular contributors that is only given read access to a repository. This does not confer additional privileges on these contributors, it instead allows for issues and pull requests to be assigned to those people. This can be used to manage the assignment of editorial or review tasks to individuals outside of the editor team.

3.5. Document Formats

In addition to the canonical XML format [RFC7991], document editors might choose to use a different input form for editing documents, such as markdown. The choice of input format is left to document editors.

4. Contribution Methods

Contributions to documents come in many forms. GitHub provides a range of options in addition to email. Input on GitHub can take the form of new issues and pull requests, comments on issues and pull requests, and comments on commits.

4.1. Issues

The GitHub issue tracker can be an effective way of managing the set of open issues on a document. The record of issues - both open and closed - can be a useful way of recording decisions made by a working group.

Issues can be given arbitrary labels, assigned to contributors, and assembled into milestones. The issue tracker is integrated into the repository; an issue can be closed using a special marker in a commit message.

When deciding to use GitHub, Working Group Chairs MUST decide how the GitHub issue tracker are used. Use of the issue tracker could be limited to recording the existence of issues, or it might be used as the venue for substantial technical discussion between contributors.

4.1.1. Issue Labelling

A system of labeling issues can be effective in managing issues. For instance, marking substantive issues separately from editorial can be helpful at guiding discussion. Using labels can also be helpful in identifying issues for which consensus has been achieved, but that require editors to integrate the changes into a document.

Labels can be used to identify particular categories of issues or to mark specific issues for discussion at an upcoming session.

If labels are a core part of working group process, chairs MUST communicate any process to the working group. This includes the semantics of labels, and who can apply and remove these labels.

4.1.2. Closing Issues

Editors have write access to repositories, which also allows them to close issues. The user that opens an issue is also able to close the issue. Chairs MUST provide guidance on who is permitted to close an issue and under what conditions.

4.2. Pull Requests

Pull requests are the GitHub feature that allow users to request changes to a repository. A user does not need to have write access to a repository to create a pull request. A user can create a "fork", or copy, of any public repository. The user has write access to their own fork, allowing them to make local changes. A pull request asks the owner of a repository to merge a specific set of changes from a fork (or any branch) into their copy.

Editors SHOULD make pull requests for all substantial changes rather than committing directly to the "master" branch of the repository.

Pull requests have many of the same properties as issues, including the ability to host discussion and bear labels. Critically, using pull requests creates a record of actions taken.

For significant changes, leaving a pull request open until discussion of the issue within the working group concludes allows the pull request to track the discussion and properly capture the outcome of discussions.

Groups of editors could adopt a practice of having one editor create a pull request and another merge it. This ensures that changes are reviewed by editors. Editors are given discretion in how they manage changes.

4.2.1. Discussion on Pull Requests

In addition to the features that pull requests share with issues, users can also review the changes in a pull request. This is a valuable feature, but it has some issues.

Comments in a review other than a summary are attached to specific lines of the proposed change. Such comments can be hard or impossible to find if changes are subsequently made to the pull request. This is problematic for contributors who don't track discussion closely.

For this reason, working group chairs SHOULD discourage the use of inline comments for substantial technical discussion of issues.

4.2.2. Merging Pull Requests

Working groups MUST determine who is permitted to merge pull requests. Document editors SHOULD be permitted to merge pull requests at their discretion. This requires that editors exercise some judgment. Working group chairs MAY occasionally identify a pull request and request that editors withhold merging until working group consensus has been assessed.

Note that the copy of a document that is maintained on GitHub does not need to be a perfect reflection of working group consensus at every point in time. Document editors need some flexibility in how they manage a document.

4.3. Monitoring Activity

Several working groups have created read-only mailing lists that subscribe to activity notifications on repositories. The volume of information on these lists can be too high to monitor actively, but access to an archive of actions can be useful.

An alternative is to rely on periodic email summaries of activity, such as those produced by a notification tool like github-notify-ml [6]. This tool has been used effectively in several working groups, though it requires server infrastructure.

A working group that uses GitHub MAY provide either facility at the request of the chairs.

5. Internet-Draft Publication

During the development of a document, individual revisions of a document can be built and formally submitted as an Internet-Draft. This creates a stable snapshot and makes the content of the in-progress document available to a wider audience.

Editors SHOULD create a new Internet-Draft submission two weeks prior to every session (see Section 7.1 of [RFC2418]). Participants in a session can't be expected to monitor changes to documents in real-time; an Internet-Draft ensures that there is a common, stable state that is known to all participants.

Working group chairs MAY request the creation of an Internet-Draft at any time, in consultation with document editors.

6. Assessing Consensus

The work that occurs on GitHub could be part of the consensus process, but the ultimate decision on consensus regarding a document is made by the chairs [RFC2026].

Monitoring activity on GitHub can require a greater time commitment than following a mailing list. This is because there is an increased volume of activity to follow. Participants who wish to limit this time commitment might follow GitHub activity selectively, either by following only specific issues or by occasionally reviewing the state of the document. Chairs are reminded that assessing consensus based on GitHub content alone cannot be assumed to reach all interested participants.

A working group chair SHOULD consult the working group mailing list for any issue that is potentially contentious. Relying on input provided through GitHub alone might result in gaining input from a narrower set of participants. This includes important milestones like Working Group Last-Call, where review from the widest possible audience ensures a higher quality document. Managing input from multiple sources in assessing consensus is similar to what is needed when balancing mailing list discussion versus in-person meeting discussion.

The use of issues and labels has proven to be effective in managing contentious issues. Explicitly labeling closed issues so that those with formal consensus means that there is no confusion about the status of issues.

7. Continuous Integration

Various third-party services offer the ability to run tests and other work when changes are made to a document.

One common practice is to use these continuous integration services to build a text or HTML version of a document. This is then published to GitHub Pages, which allows users to view a version of the most recent revision of a document. Including prominent link to this version of the document (such as in the README) makes it easier for new contributors to find a readable copy of the most recent version of a draft.

Continuous integration can also validate pull requests and other changes for errors. The most basic check is whether the source file can be transformed successful into a valid Internet-Draft. For example, this might include checking that XML source is syntactically correct.

For documents that use formal languages a part of specifications, such as schema or source code, a continuous integration system might also be used to validate any formal language that the document contains. Tests for any source code that the document contains might be run, or examples might be checked for correctness.

8. Advice to Editors

Document editors are primarily responsible for maintaining documents. Taking on a few additional tasks can greatly improve the process for the working group.

Using GitHub means that it is more likely that a contribution is made by users who aren't very familiar with the work. If a duplicate issue is raised, point the user to the existing issue before closing the issue. If a contributor seems rude in a comment, be courteous in response.

Pull requests from new contributors can contain errors or omissions. Some contributors won't natively speak English, so changes might have grammatical errors. If a change is generally sound, rather than rejecting the pull request or requesting changes, accept the change and then make any minor corrections yourself.

Never close a pull request or issue without first understanding why it was made and then explaining why you aren't accepting it. If you are uncertain, ask a chair for guidance.

If a contributor makes a comment that raises what you believe to be a new issue, create an issue for them. If the issue has an obvious solution, consider creating a pull request. It doesn't matter what venue the issue was raised in, email, issue discussion, a pull request review, capturing issues quickly ensures that problems become visible and can be tracked.

This takes a little more effort, but these simple steps can help encourage contributions, which will ultimately improve the quality of your document.

9. GitHub Limitations

At the time of writing, github.com is not reachable using IPv6. This is an affront to all that the IETF stands for and a slap in the face to all the people who worked so hard to design and deploy the latest version of the Internet Protocol. While we can collectively be ashamed and disappointed that this is the situation, that doesn't necessarily make the service any less useful.

10. Security Considerations

Continuity of operations is always a consideration when taking a dependency on an external service. If GitHub were to fail in some way, anyone relying upon its services would be seriously affected.

Widespread use of git reduces the exposure to a system failure because the primary repository is replicated in multiple locations. This includes hosted web pages; the content of web pages is maintained as a branch in the main repository. Maintaining a mirror of a repository that is hosted on GitHub is relatively simple and might be considered as a way to provide a backup for the primary repository.

However, other information maintained on GitHub is more vulnerable to loss. This includes issues and discussion on those issues, discussion and reviews of commits and pull requests, and any content hosted on the wiki. Tools exist for extracting this information for backup.

The potential for malicious actions by compromised or malcontent editors, chairs and area directors is relevant in maintaining the integrity of the content that GitHub hosts. Backups allow for recovery of content, and regular submissions as Internet-Drafts ensure that work is not lost completely.

11. IANA Considerations

This document has no IANA actions.

12. References

12.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3979] Bradner, S., Ed., "Intellectual Property Rights in IETF Technology", RFC 3979, DOI 10.17487/RFC3979, March 2005, <https://www.rfc-editor.org/info/rfc3979>.

[RFC4879] Narten, T., "Clarification of the Third Party Disclosure Procedure in RFC 3979", RFC 4879, DOI 10.17487/RFC4879, April 2007, <https://www.rfc-editor.org/info/rfc4879>.

[RFC5378] Bradner, S., Ed. and J. Contreras, Ed., "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, DOI 10.17487/RFC5378, November 2008, <https://www.rfc-editor.org/info/rfc5378>.

12.2. Informative References

[GH-CONFIG] Cooper, A. and P. Hoffman, "GitHub Configuration for IETF Working Groups", draft-ietf-git-github-wg-configuration-00 (work in progress), February 2019.

[ID-TEMPLATE] Thomson, M., "martinthomson/i-d-template", n.d., <https://github.com/martinthomson/i-d-template>.

[RFC2418] Bradner, S., "IETF Working Group Guidelines and Procedures", BCP 25, RFC 2418, DOI 10.17487/RFC2418, September 1998, <https://www.rfc-editor.org/info/rfc2418>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/info/rfc7991>.

12.3. URIs

[1] https://mailarchive.ietf.org/arch/search?email_list=ietf-and-github

[2] https://github.com/

[3] https://bitbucket.org/

[4] https://about.gitlab.com/

[5] https://trustee.ietf.org/license-for-open-source-repositories.html

[6] https://github.com/dontcallmedom/github-notify-ml

[7] https://github.com/martinthomson/i-d-template

[8] https://trac.ietf.org/trac/core/wiki

[9] http://httpwg.org/

[10] https://datatracker.ietf.org/wg/quic/charter/

[11] https://github.com/quicwg

[12] https://github.com/orgs/quicwg/teams/editors

[13] https://github.com/orgs/quicwg/teams/contributors

[14] https://github.com/quicwg/wg-materials

[15] https://github.com/quicwg/base-drafts

[16] https://www.ietf.org/mailman/listinfo/quic-issues

[17] https://github.com/orgs/quicwg/people/quic-issues

Appendix A. Experiences from Working Groups

A.1. CORE

The CoRE WG (Constrained RESTful Environments) has been actively using the Trac/SVN combination offered by the Tools Team for its older drafts.

Some newer drafts (including some drafts that are not yet WG drafts but could be considered candidates for that) are now being worked on in the "core-wg" GitHub organization.

These drafts generally use Martin Thomson's template [7], except where the build process (examples, grammars) is much more complicated than can easily be supported by this template.

For most repos, a CI (continuous integration) process is set up that generates a readable editor's copy (in HTML form) as well as a diff from the most recent submitted version (tools TXT diff), linked from the README; both have turned out to be very valuable. (Unfortunately, the travis-based CI process is somewhat brittle, so there is an appreciable failure rate.)

We try to keep discussion on the mailing list (as opposed to getting them entirely in the GitHub issues), but may not have been very successful in that; it definitely requires constant vigilance.

The WG Wiki [8] says:

> With respect to the mode of operation of the repository, the CoRE WG follows the lead of the HTTPBIS WG [9]. Specifically that means that GitHub issues are welcome to record editorial issues as well as technical ones; as are "pull requests" (forks of the repository with fixes for an issue). However, technical discussion should not happen in the forums implicitly created by the issues, but on the WG mailing list.

We currently do not have an active backup regime.

A.2. QUIC

The QUIC WG [10] was chartered in October 2016, and has been using GitHub very intensively.

We created a GitHub organization called "quicwg" [11], which the WG chairs administer. Under than organization, we set up two teams, one for WG document editors [12] and one for regular contributors [13]. Membership in the former team is contingent on being chosen as an editor for a WG deliverable. The latter team is more open, and consists of people that the chairs and editors want to assign reviews or issues to. Obviously, anyone can raise issues, comment on them, submit pull requests, etc. The benefit of the "contributors" team really lies in allowing the assignment of tasks to individuals, which is otherwise not possible.

Underneath the "quicwg" organization, we created two repositories, one for WG materials [14] and one for our base WG drafts [15]. Only the chairs have commit permissions to the WG materials repo, which is mostly used to hold presentations and other materials from our various meetings. This repo is configured to not allow issues to be raised or have a wiki (we instead store Markdown files inside the repo.)

Our second repo, for "base drafts", is where most of the work occurs. The decision to use a common repo for several drafts was deliberate. QUIC is a complex protocol with a complex specification, text moves between different documents and issues can affect several. Maintaining each draft in a separate repo, while "cleaner" on first impression, actually complicates this workflow. When the WG adopts additional drafts, we will decide on a case-by-case basis whether they will be made part of the "base drafts" or if we create a new repo underneath the organization. Since Martin Thomson is an editor, we use his setup template [ID-TEMPLATE] to rapidly publish HTML editor copies of the specs.

The "base drafts" repo is configured to allow issues to be raised, and its wiki is enabled (but rarely used.) Editors (and chairs) have commit rights to this repo.

We use sets of labels to tag issues that are raised. One set simply indicates which draft(s) an issue applies to, or whether it is potentially of broad "design" impact, or "editorial" in nature so that an editor can use his or her own discretion to resolve it without WG consensus. A second set is used to track the WG consensus on each issue (with states that currently include "needs-discussion", "confirm-consensus", "notify-consensus" and "editor-ready"). Issues progress from "needs-discussion" to either "confirm-consensus" or "notify-consensus". The former is entered when consensus amongst the participants in the discussion has emerged, and the WG needs to confirm this consensus on the list. The latter is entered when a consensus call happened at a WG meeting, and the mailing list needs to confirm this consensus. (It is not clear if two separate labels actually make all that much sense here.) Once WG consensus has been established, an issue is labeled "editor-ready".

Although the QUIC WG has only been chartered for a few months, we have already had ~250 issues raised, many of which have attracted dozens of comments. Good issue topics and actively searching for prior issues before opening new ones is essential to manage the workflow.

In order to allow WG participants to follow the activity on GitHub without needing to check the GitHub web site, we have set up a

separate "quic-issues" [16] mailing list at the IETF. It was a deliberate decision to use a list other than the regular WG mailing list. First, because we are intensively using GitHub, a lot of notifications get generated (dozens per day), which would drown out other list traffic, Second, the issues list is configured as a read-only list, where all incoming email is rejected, except for some whitelisted senders. The intent is to keep all discussion on the regular WG mailing list, or on GitHub tickets. (While GitHub will correctly reflect email replies to issue notifications, they seem to loose sender information, which is useless.)

Getting GitHub notifications to go to this list was mildly painful, and involved creating a dummy "IETF QUIC WG" GitHub user account [17], whose subscription email address is the quic-issues list address. The dummy user was made a member of the QUIC GitHub organization, and will therefore by default "track" all repo activity. This will cause GitHub to create the desired stream of notification emails to an IETF list. One caveat here is that GitHub uses the email address associated with the user who is interacting with the web site as the sender address of notification emails, which requires regular whitelisting in mailman. It also means that these users are allowed to otherwise email the issues list; we trust they don't. This email integration is rather dissatisfyingly complex; we'd be interested to learn of a better way.

Appendix B. Acknowledgments

This work wouldn't have been possible without the hard work of those people who have trialled use of GitHub at the IETF. Alia Atlas contributed significant text to an earlier version of this document.

The experiences of the CORE WG in Appendix A.1 were contributed by Carsten Bormann. The experiences of the QUIC WG in Appendix A.2 were contributed by Lars Eggert.

Author's Address

Martin Thomson
Mozilla

Email: mt@lowentropy.net