Name in progress...

Michael Frank
michael.frank@stud.hs-flensburg.de
Hochschule Flensburg
Flensburg, Schleswig-Holstein, Deutschland

Nico Hohm
nico.hohm@stud.hs-flensburg.de
Hochschule Flensburg
Flensburg, Schleswig-Holstein, Deutschland

ABSTRACT

Abstract in progress ...

KEYWORDS

Open source development, Open source, Software development, Development initiative, Browser extension, Chrome extension, GitHub, Blockchain, Ethereum, Smart contracts

1 INTRODUCTION

1.1 Open source development on GitHub

Open source development is a type of software development in which a decentralised and collaborative community develops software publicly and transparently. Open source development enables the creation of innovative and free software through the collaboration of many people, and it also provides free access to the software for everyone [11, 12]. Some examples for such significant projects are the Linux Kernel [13] and the Mozilla Firefox browser [8], which are used by many people every day. A particular difficulty in open source development is the coordination of the many developers who contribute to the development of the projects with their own extensions or improvements. Changes have to be tracked, traced and, if necessary, reversed [12]. These problems occur not only in open source development, but also in normal software development. As in normal software development, version control is used to solve these problems [12, 14].

In particular, the version control protocol Git is ideally for open source development, because with Git it is possible to have several distributed remotes that can access and manage the same source code [2, 14]. Projects that are coordinated via git are called repositories. The developers have a local copy of the repository on their systems and can push their changes to or pull the current status from the main repository. These actions are coordinated via a so-called git server, which has to be hosted somewhere so that the developers can work with it [2]. It is important that this server is permanently accessible, otherwise the actions mentioned will no longer work. and coordination of the repository will be interrupted [14]. Hosting services such as GitHub were created so that such problems can be prevented and not everyone has to set up their own git server if they want to start an open source project [3, 14]. GitHub is the largest git hosting provider today, with over 56 million registered developers and over 100 million repositories. [3] The service is also used by large IT corporations such as Microsoft, Facebook and Google and hosts a large number of the largest and most important open source repositories [5].

Normally, an open source project on GitHub starts with a user creating a repository for it. The user who created the repository is the owner of it. This user has full control over the repository and can push changes, decide which changes are accepted (merged) and even delete the repository. In addition, he can add so-called collaborators to the repository, who have read and write rights in the repository [7]. Normal users who do not have collaboration rights can contribute to the open source project by creating a pull request with their change. Another user with the necessary rights can then decide whether the change is useful or not. Depending on his decision, he can merge (accept) or reject (reject) the changes. The described process is the typical approach to how the community develops for an open source project.

1.2 Problem

The described workflow for managing repositories on GitHub but also on other hosting services such as GitLab or Bitbucket has some significant disadvantages. Firstly this approach is not necessarily decentralised or democratic. Very few people usually have the necessary rights to merge pull requests, and they can decide over the head of the general community whether to merge or reject a pull request. So it doesn't matter what the general community thinks, as long as the administrators have a different opinion. Rejecting good or useful pull requests is bad, but not a direct threat to the project. The opposite is to merge a critical bug into the main repository, which can cause enormous damage, as in the example of the Heartbleed bug in the Open SSL repository [9]. This danger exists mainly because it only takes one person with the necessary rights to overlook the bug and decide to merge the flawed pull request. Less critical problems, which nevertheless complicate the work in the communities of open source projects, are the lack of initiative for developers to review pull requests or to create pull requests themselves. If an open source repository is not financially supported by a company or a large community, they usually live on developers who work on these projects in their spare time, which means that further development sometimes takes a very long time.

1.3 Solution

To solve these problems, this paper presents the development of a chrome browser extension that uses the GitHub API [4] and smart contracts [6] on the Ethereum blockchain [1] to enable decentralised management of pull requests in GitHub repositories with a financial

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

incentive for the community. Using the Ethereum Blockchain, the community can vote on which pull requests should be merged or rejected, with decisions to merge good requests and reject bad ones being rewarded. In addition, the community can use a crowdfunding mechanism to pool Ether, the native currency of the Ethereum blockchain, to pay developers for solving problems or bugs. The functioning of the protocol on which the browser extension is based is explained in detail in *3. Protocol*. Our goal with this browser extension and this paper is to solve the problems mentioned and to improve the way open source development is done.

2 RELATED WORK

Related work falls into three areas: *repository governance*, *decentralised voting* and *developer initiative*. Previous work has already looked at decentralised approaches to manage GitHub repositories. One work pursues the idea that a repository always belongs to the ownerless protocol and that pull requests can only be merged or rejected by the community via this protocol. This should prevent the danger that users with writing rights misuse them [14].

Furthermore trustless and decentralised voting on the blockchain is a topic that has been addressed in several papers and which also plays a critical role in this paper [10, 14].

In addition, the lack of financial incentive to implement problems or changes in a repository is an issue that has been investigated. The research looked at what is important to the developer who solves the bounty and what is important to the financial backer of the bounty. [15]

3 PROTOCOL

The described browser extension basically consists of two parts. A frontend that allows users to vote on pull requests or to back a bounty financially, and a kind of protocol that runs locally in the browser extension but uses the GitHub API and the Ethereum blockchain to take over actions such as distributing stakes or merging and rejecting pull requests by the vote outcome. The protocol is the actual solution how the control of the pull request management is decentralised, the frontend on the other hand provides the possibility to interact with the protocol. In the process of development and research, a total of two protocols were designed, with the second protocol being an extension of the first. Adding further functionalities that we subsequently deemed as important. This section introduces both protocols and their differences.

3.1 First protocol

The first protocol starts with a developer deciding to develop a change and creating a pull request so that it can be merged into the main branch of the repository. After the pull request has been created, any community member, i.e. anyone who follows the repository, can start the voting phase for the pull request. In this step, a smart contract is created which is used for the later voting and the distribution of the stakes.The community can then vote for or against the pull request for a certain period of time. When someone submits a vote, they must weight it with a stake. Ether is used as a stake and the more Ether is staked on a vote, the higher its weight of it. After the voting period has expired, the protocol adds up the stakes for and against the pull request.

If the majority of the stakes, i.e. more than 50 percent, have staked for the pull request, it is merged, otherwise it is rejected. It is important to note that stakes votes cannot be changed subsequently, once the stake has been sent to the smart contract, it is held until the vote is resolved. After that, the majority stakers divide the stakes of the minority stakers among themselves. They get a percentage share of the minority stakes in relation to their stakes in the winning pool. A staker whose decision has won and who represents 50 percent of the majority stakes receives half of the minority stakes. The complete process is graphically illustrated in *Figure 1*.

3.2 Second protocol

As already described, the second protocol is an extension of the first protocol. The decision why we have expanded the first protocol is explained in the *5. Discussion*. In general, however, it can be said that we have considered the extension as an improvement of the first protocol. The protocol can be basically divided into four phases, which are as follows:

(1) Initialisation and bounty funding
(2) Issue claiming and solution
(3) Pull request voting
(4) Evaluation and distribution

During the first phase, the smart contract is created after the initialisation of the bounty process. With the help of this smart contract, all further processes such as crowdfunding the bounty, claiming the issue, voting on the pull request and distributing the rewards and stakes are orgranised persistent, decentralised and trustless via the Ethereum blockchain. The details of the mentioned processes and the individual phases are explained in this section.

Phase 1: Initialisation and bounty funding
The protocol workflow first starts independently when a community member creates an issue, in the GitHub repository, related to a problem or a new feature. If the owner or an authorized user of the open source repository thinks that the issue is reasonable, they can initiate a bounty process for the issue. The community can then fund the bounty with their own Ether to motivate a developer to solve the issue. A developer can potentially receive this bounty as a reward if he solves the issue in a pull request and the community accepts it. The Ether paid into the bounty is held in a smart contract for a certain period of time, so that the bounty cannot be negatively manipulated in the short term. If no developer wants to solve this issue and the period described above has expired, the the community members that funded the bounty receive their shares back.

Phase 2: Issue claiming and solution
While the community collects the bounty, a developer can always decide for himself whether the bounty is high enough for him to solve the issue. As soon as the reward is high enough that a developer would solve the issue for this amount, he can claim the issue

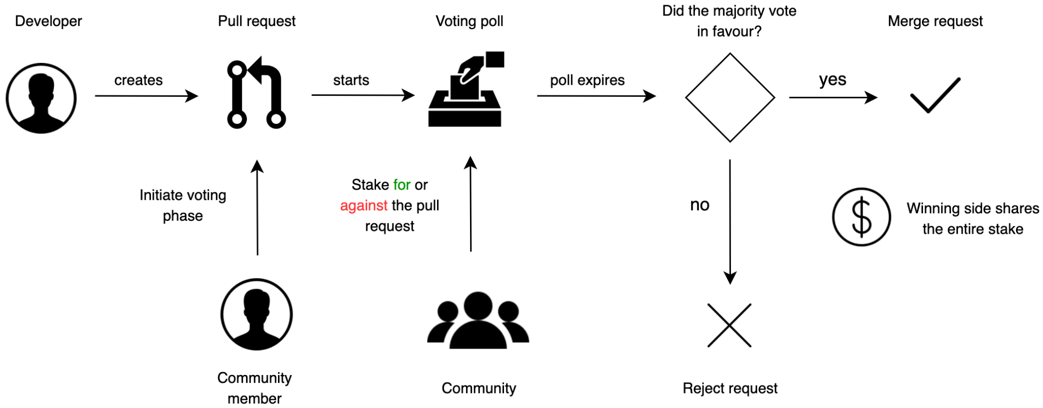

Figure 1: First version of the extension protocol

for him. But the following must be given. No other developer has already reserved the issue, an issue can only be processed by one developer. The developer has to pay a collateral to claim the issue, which he can lose if he either does not submit a solution within a given period of time or if it is rejected by the community. Once the issue is reserved by a developer, he has a certain period of time in which he has to program a solution and provide it as a pull request. If the developer does not provide a solution or it is rejected by the community, he loses his stake, which is then sent to the bounty. The issue is then set back to the claiming phase.

Phase 3: Pull request voting
This process is the same as the first variant of the protocol, the only differences are that the voting process starts, when the developer submits his pull requests and that the staker must include a comment on his vote as to why he voted the way he did. This comment is then posted in the comment section of the pull request.

Phase 4: Evaluation and distribution
As with the first protocol, the minority stakes are transferred to the majority stakers after the voting phase. The difference here is that if the pull request is accepted by the community, the developer receives a fixed percentage share of the minority stakes and the remaining share goes to the majority staker. He also receives the collected bounty. However, if the pull request is rejected,the developer loses his however, which gets allocated to the bounty and the protocol puts the issue back into the claiming phase.

4 IMPLEMENTATION

5 DISCUSSION

Protocol change
- Was sehe ich für Probleme - Was machen wir gut? - Wo sehe ich noch Probleme?

6 CONCLUSION AND FUTURE WORK

- Anbindung an Torbens

REFERENCES

[1] [n.d.]. Ethereum developer resources. https://ethereum.org/en/developers/ [Online; Retrieved 20.02.2021].
[2] [n.d.]. Git scm. https://git-scm.com/ [Online; Retrieved 20.02.2021].
[3] [n.d.]. GitHub about. https://github.com [Online; Retrieved 20.02.2021].
[4] [n.d.]. GitHub REST API. https://docs.github.com/en/rest [Online; Retrieved 20.02.2021].
[5] [n.d.]. Gitstar Ranking. https://gitstar-ranking.com/ [Online; Retrieved 20.02.2021].
[6] [n.d.]. Introduction to smart contracts. https://ethereum.org/en/developers/docs/smart-contracts/ [Online; Retrieved 20.02.2021].
[7] [n.d.]. Permission levels for a user account repository. https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/permission-levels-for-a-user-account-repository [Online; Retrieved 20.02.2021].
[8] MDN Web Docs. [n.d.]. Contributing to the Mozilla code base. https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Introduction [Online; Retrieved 20.02.2021].
[9] Melissa Iori. [n.d.]. Heartbleed Keeps Flowing-Open Source Security. ([n. d.]).
[10] David Khoury, Elie F Kfoury, Ali Kassem, and Hamza Harb. 2018. Decentralized voting platform based on ethereum blockchain. In *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*. IEEE, 1–6.
[11] Redhat. [n.d.]. What is open source? https://www.redhat.com/en/topics/open-source/what-is-open-source [Online; Retrieved 20.02.2021].
[12] Maha Shaikh and Ola Henfridsson. 2017. Governing open source software through coordination processes. *Information and Organization* 27, 2 (2017), 116–135.
[13] torvalds. [n.d.]. GitHub repository of the linux kernel. https://github.com/torvalds/linux [Online; Retrieved 20.02.2021].
[14] Torben Ulrich. [n.d.]. Development of a blockchain based access control protocol for GitHub repositories as Open-Source project. ([n. d.]).
[15] Jiayuan Zhou, Shaowei Wang, Cor-Paul Bezemer, Ying Zou, and Ahmed E Hassan. 2019. Bounties in open source development on github: A case study of bountysource bounties. *arXiv preprint arXiv:1904.02724* (2019).