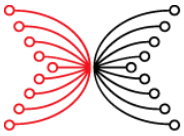

INTERMEDIATE RESEARCH REPORT | OCTOBER 31ST 2016

A REVIEW OF THE DASH GOVERNANCE SYSTEM

ANALYSIS AND SUGGESTIONS FOR IMPROVEMENT

The Veritas Team

Dmytro Kaidalov, Andrii Nastenko, Oleksiy Shevtsov,
Mariia Rodinko, Lyudmila Kovalchuk, Roman Oliynykov

IOHK.io

Abstract

This report contains the intermediate results of an analysis of the Dash Governance System, which is a blockchain-based means for self-funding the development and advancement of the cryptocurrency Dash. During this review, weaknesses, both from an architectural and implementation point of view, were discovered. Presented are possible improvements as well as development proposals, which include descriptive sketches as a preliminary step towards full formal analysis. The report also analyzes the ties between the funding in the Dash Governance System and Dash's founders/developers, as well as future funding directions. The analysis of the Dash Governance System is presented as an initial overview of DAO's and other self-governing systems for cryptocurrencies, in the spirit of both improving Dash as well as all cryptocurrencies. While the report provides a basic outline of Dash, some knowledge of cryptocurrencies and Dash is necessary for a complete understanding of the findings of this report (see, for example, the Dash whitepaper). The research was performed using the source code v0.12.1, which is currently deployed in TestNet (the current version in production is v0.12.0).

Table of Contents

1. Introduction

***Note** that all information gathered in this document relates to the v0.12.1 version of the Dash source code. This version is planned for the upcoming release and has already been deployed to the TestNet.*

The current version in the MainNet is v0.12.0. The workings of the Governance model in the new version are essentially the same as the previous version, although the code itself has been significantly refactored.

The primary purpose of the Dash Governance System (DGS) is to provide a self-sustaining mechanism for the advancement of the cryptocurrency. To the best of our knowledge, Dash is one of the first cryptocurrencies which can pay for its continued existence (development, marketing, etc.) from its blockchain. Other existing cryptocurrencies (such as Bitcoin and other altcoins) rely mostly on off-site budgeting, with several ways of obtaining funding: pre-sale of coins, donations, venture funding, etc.

The means by which Dash receives funding differs from these other cryptocurrencies: while other cryptocurrencies devote their entire block reward to miners (or to other actors who support blockchain maintenance, such as Masternodes), Dash reserves a portion of its block reward for its own future use.

These reserved coins then fund various projects related to the cryptocurrency. Anyone can suggest a project for funding (for example, a salary for the core development team). Once a project is proposed, it is voted on by the network (more precisely, by the network of Masternodes) to determine if it should be funded.

In general, a project proposal lifecycle can be described in the following diagram (Fig. 1):

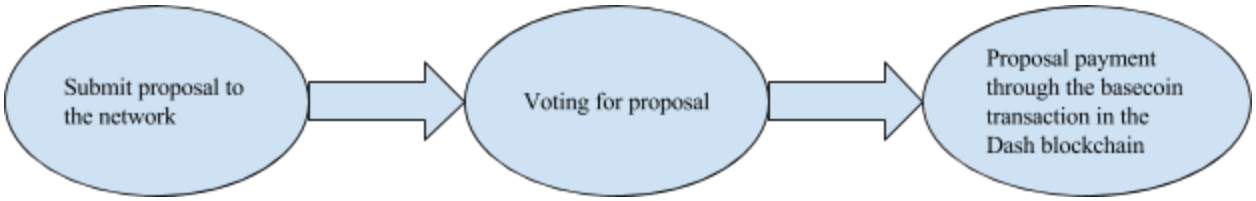

Fig.1 - High-level Scheme of Project Proposal Lifecycle

The payment itself happens approximately once a month (more precisely, once every 16616 blocks, which is roughly one month as blocks are generated every ~2.6 min). This payment occurs through a special block, called a "superblock." This superblock is essentially a regular block with a coinbase transaction added which pays money to the proposals selected for budgeting in the current voting cycle (see Fig. 2).

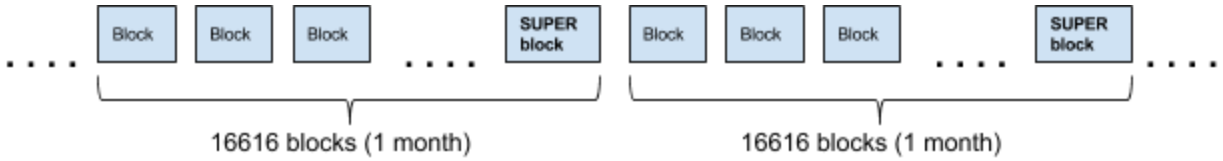

Fig. 2 - Superblock Cycles

The next three section will describe in detail how the DGS works. There are essentially three steps: the proposal (Section 2), voting (Section 3), and payment (Section 4). Section 5 will review how funds have been distributed thus far in the DGS. Following this will be an analysis of the weaknesses of the DGS (Section 6), and concrete suggestions to improve it (Sections 7 & 8).

1. Submission of Funding Proposals

The first step of obtaining funding from the DGS is to submit a project proposal to the network. Using the Dash core wallet, anyone can submit a proposal with a specially formatted proposal command. A person who submits a proposal first creates a voting transaction of 0.33 Dash (5 Dash in v0.12.0), which is the cost of setting up a proposal; this payment helps to prevent DoS attacks on the DGS and is not received to any address, but instead is burned.

1.1. Proposal Ballot Formation

As previously mentioned, a unique format for the proposal submission is necessary and includes six mandatory fields. They are represented in Table 1.

Note that if one or more of these fields are invalid the Masternodes will automatically vote negatively for the proposal, and it is deleted from the network.

Field name	Description
1. Proposal name	Should contain not more than 20 symbols. The symbols can be [*a-zA-Z0-9-_*]. It should not coincide with any existing proposal name.
2. URL to the proposal description	A char string with not less than five symbols.
3. Start Epoch	The date when payment should commence. ISO8601 date format (*YYYY-MM-DD*) should be used. Internally it is converted to the number of seconds since the start of the epoch.
4. End Epoch	The date when payment should cease. ISO8601 date format should be used. End Epoch should be greater than or equal to Start Epoch (*end_time >= start_time*). Further, End Epoch should be greater than the current time (*end_time >= current_time*).
5. Payment address	Should be a valid base58, non-multisig dash address. A valid address is a RIPEMD-160 hash which contains 20 bytes. Prior to base58 encoding 1 version byte is prepended and 4 checksum bytes are appended so the total number of base58 encoded bytes should be 25. This means the number of characters in the encoding should be about 34 (25 * log2(256) / log2(58)).
6. Payment amount	Should be a non-negative value, not more than the max budget for the next superblock (*0 < payment_amount <= next_superblock_max_budget*)

Table 1 - Proposal Data

The final proposal ballot which goes to the network consists of proposal data, current time, protocol version, and a hash of the voting transaction.

1.2. Submission Rules and Cost

Prior to submitting a proposal, a special voting transaction must be created. This transaction burns 0.33 Dash with a simple output script "OP_RETURN Hash(proposal_ballot_info)". As it can be seen, a hash of the proposal information is added to the transaction so it can be used only for one proposal. This hash serves as a unique identifier of the proposal and guarantees that the proposal data can't be changed in the future.

Hash(proposal_ballot_info) = SHA-256(nHashParent | nRevision | nTime | proposalData)

nHashParent - for proposals always equals '0';

nRevision - current protocol version. For now equals '1';

nTime - current UTC time;

proposalData - Base58 encoded data (in JSON) from Table 1.

The submission process is done with the help of the Dash core wallet command line tool and Sentinel scripts. Sentinel, a set of Python scripts, is a subsystem for managing the Governance System. One of those scripts can be used to generate a proper command for the Dash CLI (command line interface). This helps to encode proposal data into the right format.

The final proposal ballot which goes to the network has the following structure:

```
CGovernanceObject {
    Type = GOVERNANCE_OBJECT_PROPOSAL
    hashParent = 0
    nRevision            // protocol version
    nTime                // creation time
    txidFee              // voting transaction id
    proposalData         // encdoed with Base58 proposal data
```

}

Note there are no restrictions on who can submit a proposal. It can be any user who pays the 0.33 Dash collateral (i.e., it is not restricted to Masternodes).

1.3. Submission Propagation and Logging

After submission, the proposal is relayed through the p2p network to all nodes like a typical transaction.

It is important to note that the proposal is not stored in the blockchain; in fact, neither proposals nor votes are stored in the blockchain. The proposal ballots are only stored in the internal pools of each node. The nodes periodically synchronize these pools between each other.

Each node verifies the proposal by checking that the voting transaction is valid. If the voting transaction is valid, the proposal is added to the internal pool.

Some time after submission a proposal can be seen on the web (for example, at the websites Dash Central (www.dashcentral.org) or Dash Budget Proposal Vote Tracker (www.dashvotetracker.com)). These third-party tools are more convenient ways to review existing proposals. Further, each user has access to the list of current proposals through the Dash core wallet. This allows a user to see which proposals are in the internal pool of his wallet.

After the proposal has been properly submitted, the proposal submitter will promote the project to the Masternode owners and campaign for votes.

2. Proposal Voting

In the DGS only Masternodes have the ability to vote on proposals. There are several types of votes which are described in Table 2.

Type	Description
VOTE_SIGNAL_FUNDING	A vote for a proposal to be funded. This type of vote is carried out manually by Masternode operators. These votes are counted to make funding decisions about a proposal.
VOTE_SIGNAL_VALID	A vote for a proposal validness. A Masternode periodically checks all proposals and votes for their validity. This is done automatically by the Sentinel subsystem.
VOTE_SIGNAL_DELETE	A vote for a proposal to be deleted from the system. This functionality is not yet fully implemented.
VOTE_SIGNAL_ENDORSED	A vote which is used for delegation purposes. This functionality is not yet implemented.

Table 2 - Types of Votes

The most important vote type is VOTE_SIGNAL_FUNDING, for it is where the funding determination happens. The other vote types are primarily used to prevent voting for invalid proposals (i.e., an invalidly-structured proposal).

2.1. Vote Structure

A proposal is voted on using the Dash core wallet using a specific CLI command (see Table 3). There are also third-party tools which allow Masternode operators to vote, but these are simply a layer built on top of the CLI command. The vote itself can be one of three values: “Yes,” “No,” “Abstain.”

Field name	Description
TxOut	Collateral transaction + output index - the information about the Masternode collateral transaction (i.e., the 1000 Dash necessary to run a Masternode). It is used to verify that the sender of the vote is a valid Masternode.
Hash(proposal_ballot_info)	A hash of the project proposal so that the vote can be connected to the specific proposal.
Signal	The type of vote. For example, VOTE_SIGNAL_FUNDING.
Outcome	The vote for the proposal itself: “Yes,” “No,” “Abstain.”
Time	Creation time of the vote.

Table 3 - Vote structure

All this information is signed with the Masternode private key and relayed to the network.

2.2. Vote Propagation and Logging

A vote is transmitted through the p2p network like any other transaction. Each node claims a vote as valid and saves it if the following checks are passed:

- The node recognizes the proposal for which the vote was issued.
- The data of the vote is correct, together with a proper signature.
- A valid Masternode signed this vote.
- A previous vote for the given Masternode for this proposal was more than one hour ago.
- The timestamp of the vote is no more than one hour in the future (compared to current time).

Again, it is important to note that a vote is stored only in the internal storage. It is not stored in the blockchain.

A Masternode can vote multiple times for a specific proposal; however, only the last vote will be counted, and any previous votes are overwritten by the new vote. This gives the Masternode the ability to change its vote. The only rule is that the time difference between two votes on the same proposal must be more than one hour apart, in order to prevent DoS attacks.

A vote is valid as long as its corresponding proposal is valid, and there is no newer vote of the same type.

If a Masternode drops off the payment list and/or network (for example, due to not being online or due to the movement of its collateral funds), it loses its voting power and also its voting history.

2.3. Voting from the Masternode Point of View

The only actors of the Dash network who participate in voting are Masternodes. Although there are tools to allow an operator of multiple Masternodes to vote all his Masternodes simultaneously, each proposal must still be individually reviewed to determine one's voting decision. Information about a proposal comes from open discussion on the web or personally. A proposal ballot itself holds only a web link to a description, the requested amount of money, and other metadata. So an operator usually will spend some time reviewing all proposals and making voting decisions.

Note that a Masternode operator cannot modify a submitted proposal. Consider, for example, a Masternode operator who is generally supportive of a particular proposal. However, he believes the total amount of Dash requested in the proposal should be different than what was actually requested. In that case, the operator can only vote "Yes," "No," or "Abstain," for the proposal; he cannot make any suggested changes to it. However, since the discussion of the project is an open procedure, Masternode operators can argue for what they want in a proposal. But that discussion is done outside of the p2p network, usually on various Dash-related social media sites. A new proposal would have to be submitted to the network to take into consideration the suggested changes.

Taking into account that payments for proposals happen approximately once a month (at a specific block height), voting decisions regarding specific proposals can be made until this date. The actual amount of time for voting on a proposal depends on when a proposal is submitted. If it is submitted one week before the requested payment date, for example, then there is only a week for Masternodes to vote on that proposal. If a proposal doesn’t gather the required number of votes it won't be funded. So it is the responsibility of proposal creators to allocate an adequate amount of time for Masternodes to analyze and vote on their proposals.

Currently, there are typically less than two dozen proposals per payment period, so the total time required for Masternodes operators to review and vote on all proposals is minimal. Of course, as the system grows and more proposals are issued, more time from operators will be needed to review all of them. Hypothetically, it is possible that at some point in time the number of proposals will grow so that it will be impractical for a Masternode operator to have enough time to properly review and vote on them all. Because of this, the Dash community is actively looking into voting delegation schemes. These are not implemented in the current system, but Dash is working in this direction. More detailed analysis of possible delegation schemes will be done in later sections.

3. Budget Finalization and Payment Procedure

One of the most important parts of the DGS is the payment procedure. In general, this process consists of three stages: (1) submitting the finalized budget; (2) voting for the finalized budget; and (3) issuing a superblock with payments. All three stages are performed during the "budget maturity period" of 1662 blocks (approximately three days) before the superblock itself.

Note that in v0.12.1 this procedure is done automatically without the need for human interaction.

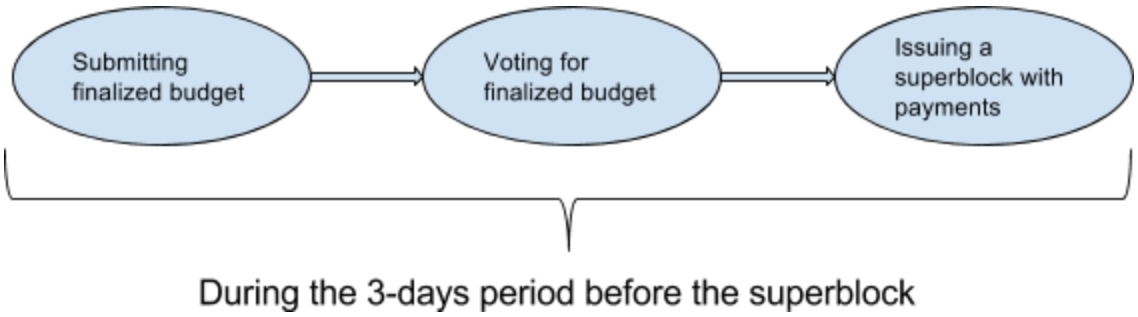

Fig. 3 - Finalizing a Budget

3.1. Adding Approved Proposals to Finalized Budget

For a set of proposals to be paid the Masternodes must agree on this set; this is the purpose of the budget finalizing procedure. It helps to reach a consensus between Masternodes on a finalized list of proposals.

The first step is for a finalized budget to be submitted. For this purpose each Masternode builds an internal finalized budget with the following algorithm:

1. All proposals are filtered so only approved proposals are considered. An approved proposal is one which is valid and has at least V amount of votes:

$$V = (V_{Yes} - V_{No}) \geq 0.1 * M,$$

where M - total amount of registered masternodes;

V_{Yes} - amount of "Yes" votes;

V_{No} - amount of "No" votes.

At least 10% of the masternodes must vote positively for a proposal to be approved.

2. All proposals are sorted in descending order based on their V value.

3. The maximum payment budget is calculated for this budgeting cycle with the following formula:

B = ((nSubsidy/100)*10)*16616,
where nSubsidy - a fixed block reward (5 Dash) decreased by 7.1% per year.

Technically this is a reconstruction of the 10% blocks rewards during the period of 16616 blocks (1 month).

4. Approved proposals are taken from the sorted list one by one. Then each is checked to confirm if the requested amount of coins is not greater than the remaining budget funds available. If the funds are available for the proposal, it is added to the finalized budget. The total remaining budget is then reduced by the requested amount of money, and the procedure is repeated for the next proposal until the total budget is exhausted or there are no approved proposals remaining that fit under the remaining budget amount.

Proposal selection procedure pseudocode:

```
var totalBudget = getTotalBudget()
Forall (proposal <- approved_and_ranked_proposals) {
        If (proposal.amount <= totalBudget) {
                finalized_budget.add(proposal)
                totalBudget = totalBudget - proposal.amount
        }
}
```

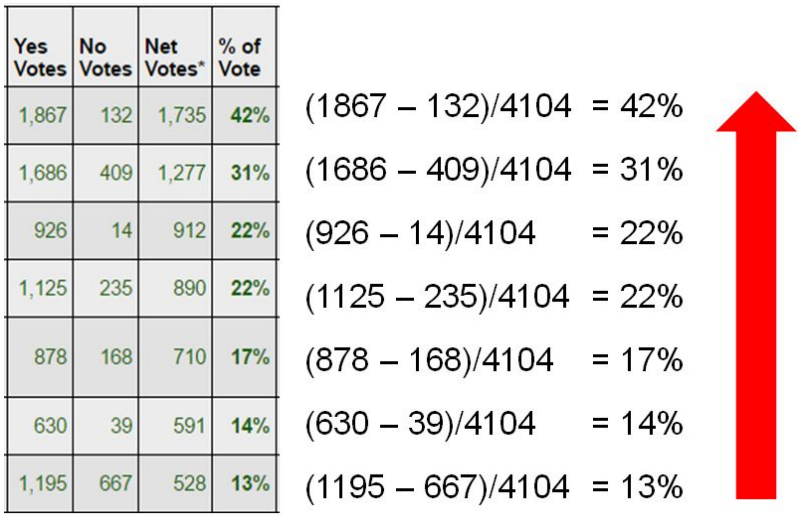

Fig. 4 - Example of Proposal Selection for Funding

3.2. Submitting Finalized Budget

The next step in the process is to submit the finalized budget to the network. Not all Masternodes will submit their versions of the budget; only a subset which is randomly selected.

The procedure for selecting the subset of Masternodes is the following: during the "finalization period" (1662 blocks prior to the superblock) each Masternode checks the hash of each issued block in the blockchain. If the hashed value of the Masternode ID is closer to the block hash than the hashed IDs of other masternodes, then this Masternode issues a finalized budget.

So for each block during the "finalization period," there will be one Masternode chosen to submit the finalized budget. This Masternode will sign the finalized budget and relay it through the p2p network to other nodes.

If a finalized budget with the same list of proposals has been already issued a Masternode will skip issuing its own finalized budget and simply vote positively for the existing one.

Note that there is no real incentive to participate in the finalization procedure. Masternodes don't receive any rewards or fees for the issued finalized budgets.

3.3. Voting for Finalized Budget

Nodes will save all valid (in terms of structure and signature) finalized budgets in local storage. Each node will give a vote for the finalized budget if and only if it is consistent with the internally-built budget (meaning that the list of proposals is the same in both; a Masternode checks the hashes of the two lists). It is possible that there are several finalized budgets (from different Masternodes) which have the same list of proposals. In this case, the vote will be given for the finalized budget with the biggest hash value.

The structure of the vote is the same as for a proposal vote. Votes are spread through the p2p network to all nodes.

Note that this procedure is done automatically.

3.4. Issuing a Superblock

Finalized budgets and votes are spread across the network with all nodes seeing and saving them. When it is time to issue a superblock, a miner will pick a finalized budget with the biggest absolute amount of "Yes" votes (biggest V value) and pay all proposals from the budget list.

Note that a superblock is valid only if it has at least V votes:

$$V = (V_{Yes} - V_{No}) \geq 0.1 * M,$$

where M - total amount of registered masternodes;

V_{Yes} - amount of "Yes" votes;

V_{No} - amount of "No" votes.

For validating a superblock, each node should have a corresponding finalized budget in the internal database.

5. Review of Previously Funded Proposals

An initial analysis of previously funded proposals includes the identity of the applicants themselves (Fig. 5), as well as the subject areas of the proposals themselves (Table 4).

As of October 2016, 65% of the entire treasury allocated to proposal funding was paid to projects initiated by Evan Duffield, the Dash creator. The second biggest applicant is Ryan Taylor (AKA "babygiraffe"), a core team member. Taylor has received 8% of project funding. Therefore, 73% of all funding from the DGS have been allocated to insiders of the system.

Such an allocation should not be surprising considering the early life-cycle of Dash, but it should be noted that such an allocation would likely be unhealthy if a permanent feature.

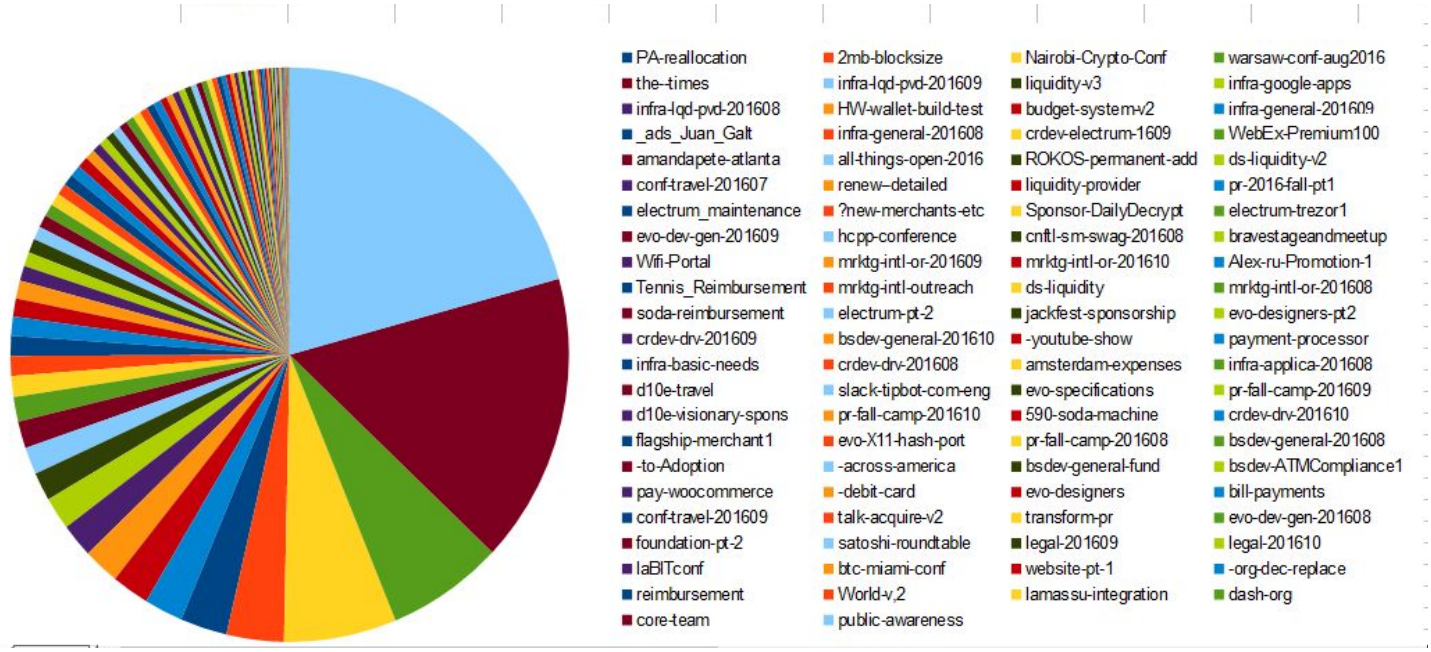

Fig. 5 - Distribution of Project Funding in Total Treasury Amount

The allocation of funds thus far has been for multiple types of proposals, with almost 60% for Marketing/PR and software development.

Funds for	Percent
Marketing and PR	35.3%
Software development	22.8 %
Domain name purchase	10.1 %
Tech development, ATM integration	7.5 %
Legal support	4.2%

Table 4 - Funding Trends Statistics

All information regarding projects payments was obtained from public web sources (dashninja.pl) and then verified by information from the Dash blockchain itself. For this purpose a special parser was developed for the Dash blockchain.

4. Weaknesses

The Dash Governance System is one of the most robust decentralized governance systems for cryptocurrencies today. However, like most first-in-kind technologies, it is not without weaknesses. Many of its weaknesses derive from the fact that the governance procedure happens primarily outside of the blockchain. Such a setup opens the system up to multiple vectors of attack. Further, there are weaknesses in the incentive structure for Masternodes, due to the significant resources required to participate. In this section, we will outline these weaknesses briefly, and in Sections 7 and 8 suggest possibilities for improvement.

4.1. Superblock Validation Attack

As has been mentioned previously, votes and finalized budgets are not fixed in the blockchain. This opens the possibility of manipulating this data to create forks in the blockchain. A situation thus exists in which a superblock (which is still essentially a coinbase transaction) must be validated with the help of data which is not stored in the blockchain.

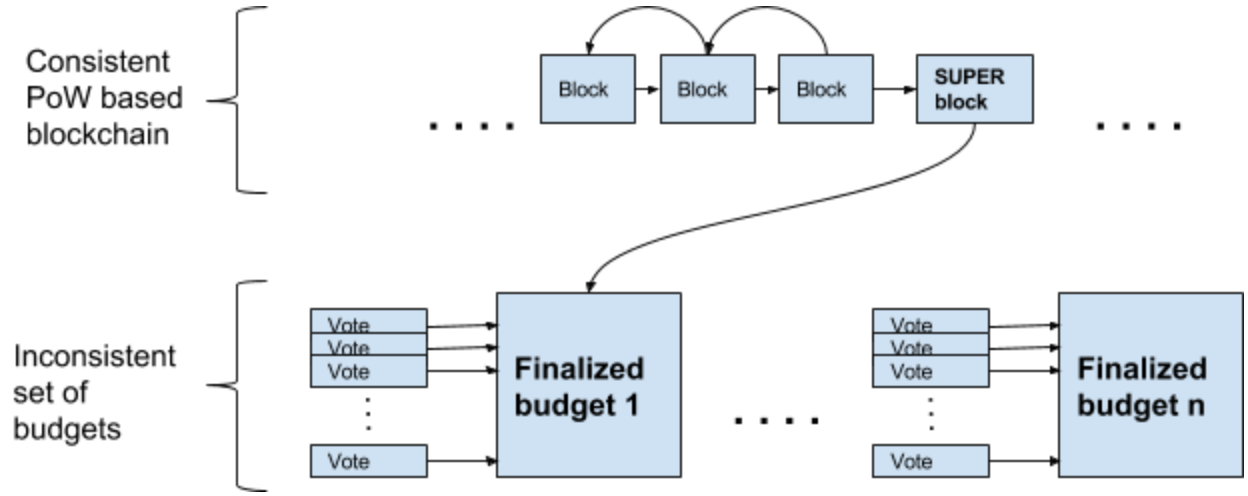

Fig. 6 - Block validation dependencies

As it can be seen from Fig. 6, validating a regular block requires information from the previous block. Particularly, a proof hash is used for selecting a random Masternode to be paid in coinbase transactions. However, in the case of a superblock, further external information is needed to validate the coinbase transaction. The superblock contains payment for the proposals, but how are those payments validated? For this, a finalized budget is required. Each node must have a set of all finalized budgets, and for validating a superblock, it will choose the finalized budget with the largest number of votes.

Modeling an attack:

1. Consider a point of time when superblock (S1) is to be created. A miner who creates this superblock will pick a finalized budget (F1) with the largest number of votes (VF1). At this point of time VF1 > VFi where Fi is any other finalized budget. So all nodes on the network will use F1 to validate the superblock.

2. Further, assume that the blockchain proceeds to some number of blocks since S1. At this point in time, an adversary issues additional votes for the finalized budget F2 so the absolute value of votes VF2 > VF1. At this moment the finalized budget F1 is no longer valid for the superblock S1. All honest nodes will therefore reject S1 and all further blocks.

Thus, manipulation of finalized budgets and votes can lead to forks in the blockchain. The manipulation itself is a result of the fact that there is no strong consensus (via the blockchain) regarding finalized budgets.

4.2. Multiple Voting

In the current implementation of the DGS, votes are issued by the Masternodes and relayed through the p2p network to other nodes. A vote is checked for validness only when it is seen the first time. This means that once a vote is accepted and added to the internal pool, it will be valid as long as the corresponding proposal is valid. A vote will not be deleted even if the Masternode which signed this vote is no longer a Masternode (for example, the Masternode operator transfers coins away from the Masternode's collateral transaction output).

This leads to a situation in which a single person with 1000 Dash can successively establish several Masternodes, and then issue multiple votes for the same proposal. Alternatively, an operator of multiple Masternodes could vote for a project and then take his Masternodes offline. This would reduce the total number of Masternodes used in determining a proposal’s approval. Either situation, of course, can lead to a manipulation of the final voting result for that proposal.

Note that it is possible that this problem is a bug in the code. By design, if a Masternode drops of the payment list/network due to not being active or due to movement of collateral, it loses its voting power and also its voting history (this information was obtained on the Dash forum [2]).

4.3. Discarded Votes

As was mentioned previously, if a Masternode moves its collateral funds it is no longer a Masternode and loses its voting power as well as its voting history. However, if votes from this particular Masternode are no longer valid, then it is possible that its corresponding finalized budget is also invalid because it is no longer the most voted.

4.4. Masternode Voting Incentives

Another weakness of the DGS is its incentive structure. In the current model there is no reward for Masternodes to analyze proposals and to vote. In practice, proposal analysis is probably the most time-consuming operation for a Masternode operator, for it requires direct human interaction.

Simply maintaining a Masternode requires only the 1000 Dash collateral, the initial bootstrapping, and any costs associated with maintaining the hardware and updating the software. The Masternode itself operates autonomously; however, participation in the DGS needs direct human involvement: analyzing the information from each proposal requires research and often additional expertise.

Currently, there are only a few proposals to decide upon each month, making proposal analysis possible by each Masternode operator. As the system grows, however, and more people start to use Dash, it is likely there will be exponentially more proposals, and proposal analysis would no longer be a trivial exercise for the Masternode operator.

Consider the following. Assume that maintaining a Masternode leads to a 10% return in Dash coins per year. Currently, the price for 1 Dash is near $10. The cost of one Masternode is 1000 Dash, leading to approximately 100 Dash in return per year, which is approx $1000 at the current price of Dash. Thus, the return in fiat is around $85 per month. (You also must subtract from that any cost of maintaining the server - typically between $5-$20/month). Such a return is not commensurate with spending many hours analyzing dozens of proposals each month.

One could argue that there would only be more and more proposals because the price of Dash is rising. Thus, the return for the Masternode in spending power is much greater than 10%, and therefore worth the time for proposal analysis. This is true for those who have invested in Masternodes prior to Dash's price rise. Once the price stabilizes, however, the incentive again diminishes for new Masternode operators to be involved in proposal analysis and voting, for their actual return remains 10%.

Further, proposals run a broad gamut of areas: software development, marketing, third-party integrations, etc. The expertise required to analyze all these proposals properly is beyond most people, and so a Masternode operator might need to consult or hire an expert, further lowering the return on his investment.

Thus, for these reasons a rational Masternode operator might choose to analyze and vote on only a handful of proposals he or she finds of interest, or no proposals at all.

4.5. Voting Representation

In every cryptocurrency, the question of who is involved in governance looms large. In most cryptocurrencies, miners are the primary "voters," as they, in essence, control the network. The DGS is an attempt to move decision-making to other actors, specifically, Masternode operators.

At first glance, this appears to be a representative democracy: a selected group of individuals who represent the whole community. However, Masternode operators themselves are not selected via a voting process; they are simply investors in Dash.

In theory, anyone can operate a Masternode and participate in voting. However, there is a limited number of Masternodes based on the total money supply of Dash at the time. So, theoretically, this limit could be reached, and others would not be able to participate in the DGS.

More practically, the cost of operating a Masternode can be too high for many people, particularly due to the 1000 Dash collateral required. As the price of Dash increases, the investment needed for a Masternode - and thus a place in the DGS - increases.

Currently, there are third-party alternatives which allow users to purchase a "share" of a Masternode, but these solutions are not part of the Dash protocol itself, nor do they allow for direct voting in the DGS.

As a result, Dash has a system where all decisions are made by a limited set of specific actors. The opinion of the rest of the Dash citizens (miners, regular users) is not considered.

4.6. Voting Delegation

Currently, the Masternodes in the DGS engage in direct democracy. This means that each Masternode votes for each project proposal and finalized budget. There is no build-in delegation mechanism.

Of course a Masternode operator can directly delegate his Masternode private key to some other party who will sign votes. It is likely inevitable that Masternode operators will voluntarily delegate their votes to third parties as the DGS grows and more proposals need to be analyzed. But, of course, such a delegation opens the Masternode operator - and possibly the network itself - to possible vectors of attack.

4.7. Decreasing Treasury Fund

The portion of funds for proposed projects is 10% of each block reward during a budgeting cycle. The block reward itself is decreased at a rate of 7% per year. Therefore, payments for proposals are also decreased by 7% per year. Eventually, at some point in the future, there will be no funds to pay out.

4.8. Flexibility of Treasury Monetary Policy

The allocation of funds in the DGS is not flexible. Unspent money from a previous cycle can’t be transferred to the next cycle, and any funds unspent are simply never created (which has a similar effect to burning funds).

Further, there is no mechanism to regulate the amount of funds issued for proposals. It is bounded by a hard-coded 10% of the block reward during budgeting cycle and no more or less. If the price of Dash drops significantly it is probable that very few projects would be able to be funded because the total pay-off in fiat currency would be too low.

At the current time the treasury budget is always exhausted in superblocks, i.e., almost all funds are spent each budgeting cycle. What if the Dash community grows and more funding is needed beyond what the current 10% can provide? There would need to be a change in the source code to increase the total budget allocation.

4.9. Code Review

During investigation of the source code several bugs were found and reported to the core team. Some of these were quite critical.

For example, there was a bug in which a Masternode could issue an indefinite amount of votes for the same proposal, with all these votes valid and counted. It was reported on the Dash forum [4] with detailed information and links to the source code. After discussion with developers the issue was accepted and fixed by a Dash team member.

One bug was committed directly to the Dash repository through the pull request [5]. There was a synchronization issue with project proposals (if a node gets a vote for which it doesn't have the corresponding proposal then it should synchronize proposal, but this functionality didn't work correctly). The pull request was successfully reviewed and merged to the Dash repository.

The attack discussed in Section 6.1 was also reported and discussed on the forum [6]. Most likely, however, this is not a bug in code but an architectural weakness.

There is also an issue with discarded Masternodes as mentioned previously. Once a vote has been considered as valid, it remains in the internal pool of a node even if the corresponding Masternode isn't actually a Masternode anymore.

Additional bugs were discovered during the investigation process; however, they were not reported due to time constraints.

Furthermore, a good deal of functionality isn't finished yet. The current version of the code being analyzed hasn't been deployed to the MainNet yet, so it is possible that additional functionality will be added before release.

5. Directions for Improvement

Currently, the Dash cryptocurrency is in the early stages of its development. The Dash core team is continuously integrating new features into the cryptocurrency.

In our opinion, there are several areas where the DGS can be significantly improved. In this section, we will describe some of these improvements.

5.1. Voting Delegation

The current voting procedure in the DGS presupposes direct voting by Masternodes for all proposals (Fig. 7).

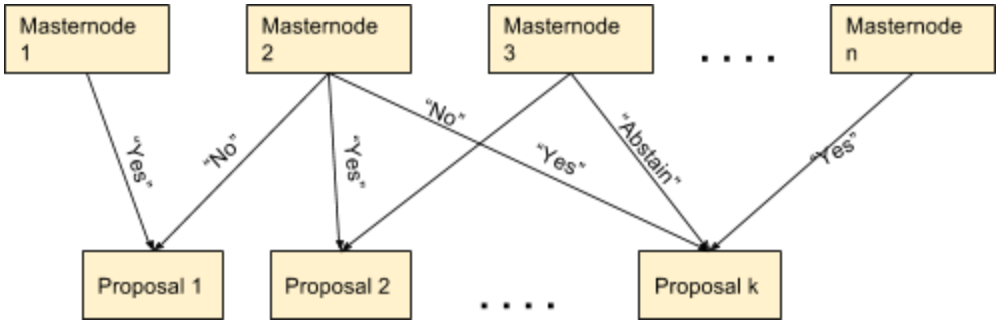

Fig. 7 - Direct Voting by Masternodes

As discussed in Section 6.4 ("Masternode Voting Incentives"), the current DGS setup can lead to low Masternode voting participation, due to the resources and expertise needed to participate. This setup can lead to nonoptimal decisions.

An improvement of this system would be to integrate a delegation scheme in which Masternodes can delegate their voting power to a third party which has the incentive and expertise to vote optimally.

5.1.1. Voting Committees

One possible solution to the participation issue has already been suggested by the Dash community[3]. (Note that it is a very early discussion on the forum, not an announced feature, so it can't be treated as an official statement). Based on the notion of committees, the suggestion is that Masternodes delegate their voting power to committees which are small groups of Masternodes willing to do the research necessary for each proposal. These committees would be elected for a set period of time, and once a committee has been elected, it has the power to decide on proposals internally and then signal their decisions to the network.

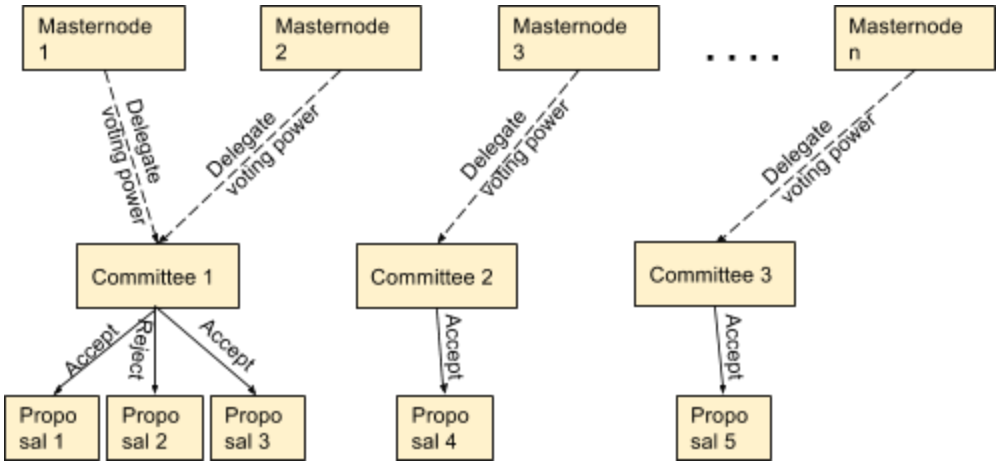

Fig. 8 - Delegation of Voting Power to Committees

If a Masternode doesn't want to delegate its voting power, it would still be able to vote on proposals directly.

There are still many open questions in such a system. How should the proposals be divided between different committees? Should there be one or more committees? Should each committee have different specializations (i.e., marketing, development, infrastructure, etc.)? Regardless, the general direction of the delegation of voting power to entities with greater expertise is a positive one.

5.1.2. Liquid Democracy

Another possible solution is to implement a system based on liquid democracy. The voting power of a Masternode can be delegated to some third party (it can be a Masternode or not). The more Masternodes delegate their votes to some other party the more voting power this party will have. Then these delegates will vote for the proposals.

In any time a Masternode can revoke its delegation, so there is a direct incentive for the delegate to act honestly. Of course, for such a system to work efficiently, some monetary incentive should be offered to the delegates.

In this system a Masternode would not be required to delegate its vote; the Masternode could still directly vote for proposals.

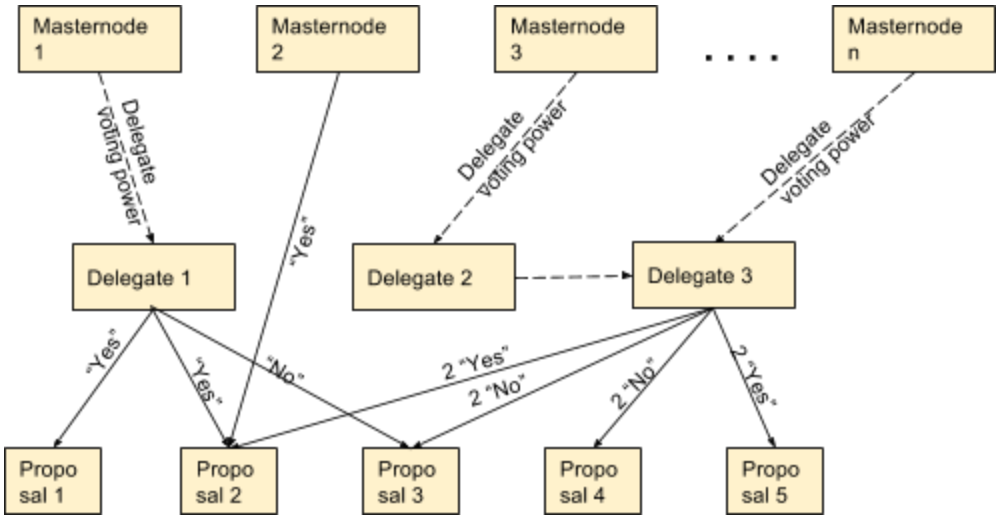

Fig. 9 - Liquid Democracy

Such a system could have many benefits: it is scalable, can lead to better decisions, and has a low barrier to entry to be a delegate.

In a liquid democracy, voting power is proportionally divided among delegates according to the willingness expressed by Masternodes.

Technically this can be implemented via proxy signatures or a special delegation transaction.

5.2. Participation of All Actors

Voting systems are most ideal when they best represent the desires of all actors in the system. In the DGS, however, only Masternodes can vote for proposals. Thus, Masternodes will naturally vote according to their preferences; however, this might not necessarily be the best choice for the entire system. Thus it would be better to give some level of voting power to all participants in the system *according to their stake in that system*.

5.3. Flexible Project Funding

Flexible project funding would involve adjusting the amount of funds generated for projects in each budgeting cycle. Currently, the amount of funds generated in the DGS is upper-bounded by the 10% of each block reward during the cycle. It would be more optimal to not set such bounds, and instead give voters a chance to decide the amount of funds that would be created.

It is reasonable in this case to provide a proportional system whereby more funds would require more votes. That would prevent the possibility of uncontrolled issuing of funds.

6. Development Proposals

6.1. Adoption of Self-Sufficient Blockchain

As it was noted earlier (see Section 6.1 "Superblock Validation Attack"), it is possible to invalidate the blockchain by updating information outside it, such as the list of votes for projects.

To prevent this vulnerability, the blockchain must contain all the information needed for blockchain validation. Thus, all information related to a finalized budget, as well as a list of all votes for it (pro and contra) should be included in the blockchain (i.e., hashes of both).

Purpose: to prevent attacks that change data not included in the blockchain itself to invalidate the blockchain.

6.2. Flexible Participation Requirements and Funding

6.2.1. Flexible Participation Requirements

Currently, the collateral amount of Dash needed to run a Masternode is fixed at 1000 Dash. For a more flexible and, correspondingly, more stable system in a changing environment, it is preferable to have a dynamic value for voting participants' deposits. This would allow more actors to participate, regardless of the current price of Dash.

6.2.2. Flexible Project Funding

Currently, if all the funds in a funding period are not allocated to projects, the remaining funds are lost (i.e., never generated). However, a better approach would be the possibility of transferring unissued funds to future funding epochs. Alternatively, the system could have the possibility of issuing extra coins for project support, with strict requirements for voters supporting such coin emission.

For both flexible participation requirements and flexible project funding, quorum requirements for significant shifts must be strict to give assurance of current system stability.

Purpose: to have a flexible self-controlled procedure for adjusting system parameters when the external environment changes.

6.3. Implementation of Delegation with Liquid Democracy

As the number of proposals increases along with the difficulty of understanding them, it becomes more difficult to recognize the best decision on each funding proposal. In the future, there may very well be too many projects that require professional expertise to be properly evaluated and vote upon.

Thus, participants with voting rights should have the possibility of voting directly or delegating their votes to experts in specific categories (described in Section 8.4 below).

Proposed delegation format: [participant, category, delegate, up to term/up to revoke]. E.g., participant votes directly for development and independent expertise selection, but delegates votes for legal support, marketing, etc.

If a Masternode operator is not satisfied with the vote given by delegated expert, he can issue a vote which supersedes the vote of the delegate. Further, if Masternode operators believe delegated experts are corrupted, they can always override their votes.

Purpose: *automatically* forming expert groups on specific categories, allowing *paid* professionals to direct cryptocurrency development.

6.4. Assigning Categories to Each Funding Proposal

Delegation opens up the possibility of including experts in the proposal approval process. Nevertheless, there are no universal experts, as each expert has his particular area or areas of professional expertise.

Thus, specific categories for proposals need to be defined. E.g.:

- PR/marketing;
- development;
- independent expertise;
- legal support;
- infrastructure;
- charity;
- other.

Each ballot upon submission would be assigned a specific category, which would allow for a more efficient delegation process.

Purpose: simplify selection of experts for specific categories.

6.5. Introduction of Paid Professional Expert Role

Every successful innovation requires several stages on its path from new idea to widespread technology. Among them is switching from early adopters with expert level knowledge (“techno-geeks”) to the average users who have no deep understanding of the technical details of the system they utilize. However, during this process, the complexity level of the technology often increases, which requires involving professional experts for continued development. So we see various actors involved with competing needs and divergent bases of knowledge.

Consequently, the implementation of direct democracy can be sufficient during early stages, but often delegation is required in future stages. Moreover, the complexity of proposed projects can lead to the impossibility of restricting decisions to volunteers. Thus, the introduction of paid professional experts is required at some point in the technology's lifecycle.

One possible solution for this issue in the DGS would be as follow:

If a delegate is selected by some pre-determined number of Masternodes (e.g., greater than 0.5% of all voting power in a specific category), he would be included in the list of paid experts for this category.

Payment, proportional to delegated votes, is provided for:

- voting “yes” for projects which are accepted in the finalized budgets;

- voting “no” for projects which are rejected in the finalized budgets.

Payment to experts is issued only after project completion or some predetermined period after project completion. The longer the delay for payment is, the higher the increasing coefficient used (higher than bank interest, etc.)

Information on expert activity can be kept outside the blockchain (i.e., a separate database shown at a site as is done in the current DGS, or even a separate “professional expert” blockchain). Only the rewards for experts are included in the blockchain (with format “to be available since ...”).

Purpose: having automatically-formed groups of professional experts, and creating incentives for them to select high-quality projects accepted by the community.

6.6. Expanding List of Voting Entity Types

A successful cryptocurrency requires several types of participants. At the very minimum, every cryptocurrency needs the following:

- users who do asset exchanges based on technology provided by cryptocurrency;
- entities who verify transactions and produce new blocks (“miners,” “slot winners,” etc.);
- network nodes to support functions of the peer-to-peer network (“full nodes/Masternodes”).

Only users obtain resources from outside the system and are ready to pay (directly or indirectly) for cryptocurrency utilization. The remaining entities (miners and full nodes) have an incentive to take part and serve the system. As was mentioned previously (section 8.5), during early stages of system development, techno-geeks may provide support on a volunteer basis, but as the project matures, there must be an economic incentive for continued contributions.

Bitcoin and most other cryptocurrencies provide direct compensation for miners, while Dash (and a few other solutions, see appendixes) also provide compensation for full nodes (Masternodes).

However, for governance decisions, Dash provides voting rights to Masternodes only. When this system comes to a more mature development level, there may be conflicts of interest among the various participants:

- Masternodes (the only voting entity) are interested in increasing their income, which may lead to:
  - increasing their portion of the block reward, at the expense of miners or the funding system;
  - increasing transaction fees or block rewards (direct or indirect payments from users), which may lead to users leaving the cryptocurrency due the high cost of transactions and/or high cryptocurrency inflation;
- miners are interested in increasing their part of reward (at the expense of the Masternodes or the funding system), as well as increasing payments from users, with corresponding consequences;
- users are interested in decreasing their payments, which leads to a decreasing of funds available to miners and Masternodes, which may lead to a cessation of their activity.

As was already mentioned, Dash currently provides voting rights to Masternodes only, which could lead to a

usurpation of cryptocurrency control. This, in turn, can lead to the loss of other participants' interests, leading them to leave the system and destroying the cryptocurrency itself.

To keep a proper balance, some level of voting rights are needed to all essential participants.

In our opinion, there should be some reserved quotas for each participant type (e.g., 20% for users, 30% for miners and 50% for Masternodes; specific values should be decided by further analysis). This would be determined by some level of deposit, which could be defined by an auction-like procedure.

Existing incentives for Dash Masternodes (full nodes with deposit), as well as for block issuers (miners), are kept.

All coins from the auction-determined voting deposit are returned after their expiration (when not used for a next term deposit).

Purpose: to keep a proper balance of all participants' interests, as well as to keep the cryptocurrency being developed instead of being exhausted by the unique incentives of one participant subtype.

7. Conclusions

1. Dash is the first cryptocurrency with a developed and practically-tested self-governing system which allows funding of cryptocurrency development projects.

2. Dash utilizes a significant amount of Bitcoin source code (with a different hash function to prevent Goldfinger attacks from Bitcoin miners). Nevertheless, the new version of Dash provides a large amount of own source code, not only in C++, but also in Python, for its Governance System.

3. Dash has three main type of actors in its system: regular users, miners, and Masternodes (full nodes with a collateral deposit). Masternodes are tasked with supporting the cryptocurrency's backbone p2p network, and they receive block rewards for these activities. Masternodes are currently the only actors who have the right to vote for cryptocurrency development proposals.

4. Any network participant can submit a funding proposal. As of v0.12.1, the cost of a proposal is 0.33 Dash (to prevent DoS attacks). A hash of the submission is included in the blockchain with a transaction that burns 0.33 Dash. Votes from Masternodes are spread over the p2p network like transactions, but votes are not included in the blockchain (they are only kept in each Masternode/full node local database).

5. Each reward cycle for Dash is 16616 blocks, which is approximately one month for a typical production frequency of new blocks. 90% of this period is devoted to manual human (Masternode operator) voting for proposals, 10% (the last 1662 blocks, or approximately three days) is given to automatic finalization of the list of funded projects (finalized budget) for the current period. One reward period can provide up to 8308 Dash = 16616 (blocks) * 0.5 (block reward allocated to the treasury). This amount is decreased annually due to the decreasing of the block reward. The necessary condition for a proposal to become funded is to receive at least +10% of the total possible votes between the votes given pro and votes given contra. The most popular projects (i.e., those receiving the maximum difference pro votes) are funded up to exhausting the complete treasury reward.

6. For the period of September 2015 until October 2016, the total amount of Dash received by funded proposals through the DGS reached approximately 94,000 Dash. At the current exchange rate (Oct 2016), this equals approximately 850,000 USD. An initial analysis of funded projects was made which showed that approximately 73% of the total funding went to project proposals from the cryptocurrency founders/developers. The main uses of project funding were marketing & PR, software development, and a domain name purchase.

7. While being the first significant cryptocurrency with self-governance and self-funding, the DGS has certain weaknesses, both from an architectural and implementation point of view. For example, it is possible to invalidate the blockchain by changing information not included in it (changing voting results), and to have participants engage in multiple voting, as well as other issues. Further, we found that the source code needs quality improvement. During our research, several bugs, including some critical for system security, were found and reported to the core team. We must again note that the code we analyzed was currently in testing and not in production.

8. This report presented various directions for improvement as well as development proposals targeted towards security, efficiency, and stability of the DGS. In addition, to avoid hard-coded "magic numbers" (e.g., 1000 Dash for Masternode deposit), further research is needed to determine specific parameters for the System.

8. References

[1] https://github.com/dashpay/dash/tree/v0.12.1.x

[2] https://www.dash.org/forum/threads/12-1-testnet-testing-phase-two-ignition.10818/

[3] https://www.dash.org/forum/threads/12-1-testnet-testing-phase-two-ignition.10818/#post-105009

[4] https://www.dash.org/forum/threads/12-1-testnet-testing-phase-two-ignition.10818/page-3#post-105703

[5] https://github.com/dashpay/dash/pull/1061

[6] https://www.dash.org/forum/threads/12-1-testnet-testing-phase-two-ignition.10818/page-3#post-105995

This report is based on information primarily obtained from the Dash source code. Some additional information was obtained from discussions with developers on the Dash online forum.

Note that all information gathered in this document relates to the v0.12.1 version of the Dash source code. This version is planned as the next production release and has already been deployed to the TestNet.

The current version in the MainNet is v0.12.0. The essential model of the DGS in v0.12.1 is the same as v0.12.0; however, the code itself has been significantly refactored.

9. Appendix A. Cryptocurrency Governance Formal Descriptions (for further game-theoretic analysis)

9.1. A.1 Formal Description of Dash Election Procedure (sketch)

Definition 1. A proposal is a bunch of information $p = \{name, descr, start, end, addr, amount\}$ where: *name* - proposal name, *descr* - web url to the detailed description of the project, *start/end* - a period of time when this proposal participates in the election procedure, *addr* - address for payment, *amount* - amount of payment. A proposal is an entity which suggests to fund some project. A set of voters should decide if they want to pay for this project or not.

Let $B = \{B^{(1)}, B^{(2)}, ...\}$ be a blockchain where $B^{(i)} = \{b_1^{(i)}, b_2^{(i)}, ..., b_{16616}^{(i)}\}$ represents an $i-th$ budgeting cycle. Each $b_{16616}^{(i)}$ block in the $i-th$ budgeting cycle is considered a superblock where elected proposals are paid. A proposal can be paid in a certain superblock only if this superblock is issued in a period between *start/end* dates of of the given proposal. Before a superblock is issued all voters are supposed to vote on proposals with next options: "Yes", "No", "Abstain", so that they form a preference profile Pr.

Definition 2. An election $E = (P, V, Pr)$ is given by a set $P = \{p_1, p_2, ..., p_m\}$ of project proposals, a set $V = \{v_1, v_2, ..., v_n\}$ of voters and a preference profile $Pr = \{pr_1, pr_2, ..., pr_n\}$ where each pr_i represents preferences of the voter $v_i \in V$. A preference profile of $i-th$ voter $pr_i = (pr_i^Y, pr_i^N)$ consists of 2 sets:

- $pr_i^Y \subseteq P$ (a set of proposals $pr_i^Y = \{p_{\mu^i(1)}, p_{\mu^i(2)}, \dots, p_{\mu^i(\zeta(i))}\}$ which were voted positively by the voter $v_i \in V$);

- $pr_i^N \subseteq P$ (a set of proposals $pr_i^N = \{p_{\eta^i(1)}, p_{\eta^i(2)}, \dots, p_{\eta^i(\xi(i))}\}$ which were voted negatively by the voter $v_i \in V$);

Any voter $v_i \in V$ can't vote positively and negatively simultaneously: $pr_i^Y \cap pr_i^N = \varnothing$. It follows that $pr_i \subseteq P$ (a set of proposals which were voted by the voter $v_i \in V$ is always a subset of P).

An *election rule* R is a mapping that given an election $E = (P, V, Pr)$ outputs a set of winners $W = \{p_{\varpi(1)}, p_{\varpi(2)}, ..., p_{\varpi(k)}\}$ that are selected to be payed in the current budgeting cycle. A set of winners is eventually a subset of all proposals $W \subseteq P$.

A election rule R is parametrized by the function $amount(p_i)$ which is given a proposal p_i returns an amount of money requested by this proposal and by the function $limit(B^{(j)})$ which is given a budgeting cycle $B^{(j)}$ returns a reward limit for this cycle (that is maximal amount of money which can be payed to all proposals).

An election rule $R(E)$ returns a set of winners $W \subseteq P$ according to the next rules:
1. Let $\nu(p_i)$ be a function given a certain proposal $p_i \in P$ returns an absolute count of votes for this proposal $\nu(p_i) = \sum_{j=1}^{n} \pi(p_i, pr_j)$, where

$\pi(p_i, pr_j) = 1$ *if* $p_i \in pr_j^Y$,
$\pi(p_i, pr_j) = -1$ *if* $p_i \in pr_j^N$,
$\pi(p_i, pr_j) = 0$ *if* $p_i \notin pr_j$.

2. Let $P_{acc} \subseteq P = \{p_{\chi(1)}, p_{\chi(2)}, ..., p_{\chi(\varrho(z))}\}$ be a sorted list of proposals so that $\nu(p_{\chi(1)}) \geq \nu(p_{\chi(2)}) \geq .... \geq \nu(p_{\chi(\varrho(z))})$ and for each p_i, $i \in \{\chi(1), ..., \chi(\varrho(z))\}$ absolute amount of votes $\nu(p_i) \geq 0.1 * |V|$ (proposals which have $\nu(p) < 0.1 * |V|$ are considered invalid and not included the set P_{acc}).

3. A set of winners $W = \{p_{\varpi(1)}, p_{\varpi(2)}, ..., p_{\varpi(k)}\}$ consists of proposals from P_{acc} where $\varpi(1) > \varpi(2) > ... > \varpi(k)$ so that $\sum_{i=1}^{k} amount(p_{\varpi(i)}) \leq limit(B^{(j)})$. Reward limit of the budgeting cycle $B^{(j)}$ is defined as follows $limit(B^{(j)}) = \sum_{t=1}^{16616} \sigma(b_t^{(j)})$, where $\sigma(b_t^{(j)}) = 0.1 * blockReward$ (so that 10% of block reward is accumulated for the budgeting purposes).

9.2. A.2 Formal Description of Cryptocurrency Treasury (sketch)

Let $B = (B^{(1)}, B^{(2)}, \ldots)$ be a blockchain of cryptocurrency with treasury, where $B^{(i)} = (b_1^{(i)}, b_2^{(i)}, \ldots, b_{\tau(i)}^{(i)})$ represents a i th reward epoch consisting of $\tau(i)$ blocks.

Blocks are bind into a sequence by a cryptographic hash function using a block state $\omega(.)$ function:
$\omega(b_j^{(i)}) = H(b_{j-1}^{(i)}), j \in \{2, 3, \ldots, \tau(i)\}$,
$\omega(b_1^{(i)}) = H(b_{\tau(i-1)}^{(i-1)})$ and
$\omega(b_1^{(1)}) = 0$.

Reward *limit* for the epoch i is defined as

$$\sigma(B^{(i)}) = \sum_{j=1}^{\tau(i)} \sigma(b_j^{(i)}),$$

where $\sigma(b_j^{(i)}) = \sigma'(b_j^{(i)}) + \sigma''(b_j^{(i)})$,
$\sigma'(b_j^{(i)})$ is a block reward and
$\sigma''(b_j^{(i)})$ is collected transaction fees for the block $b_j^{(i)}$.

Note that in Dash v0.12.1 for the current year within a cryptocurrency normal network mode some parameters are constant, e.g. $\tau(i) = 16616$, $\sigma'(b_j^{(i)}) = 0.5$ and $\sigma''(b_j^{(i)}) = 0$.

There is a list of projects (winners) funded at the i th reward epoch: $(\pi_1^{(i)}, \pi_2^{(i)}, \ldots, \pi_{\rho(i)}^{(i)})$. Each project $\pi_j^{(i)}$ got $\upsilon(\pi_j^{(i)})$ votes during the voting stage, where $\upsilon(\pi_1^{(i)}) \geq \upsilon(\pi_2^{(i)}) \geq \ldots \geq \upsilon(\pi_{\rho(i)}^{(i)})$.

Funding amount $\eta(\pi_j^{(i)})$ for the winning project is provided according to the project proposal, and

$$\sum_{j=1}^{\rho(i)} \eta(\pi_j^{(i)}) = \eta(B^{(i)}) \leq \sigma(B^{(i)}).$$

Treasury for the i th reward epoch is divided among honest and malicious projects: $\eta(B^{(i)}) = \eta^H(B^{(i)}) + \eta^M(B^{(i)})$.

The attacker's aim is to maximize his funding amount: $\eta^M(B^{(i)}) \to max$, up to $\eta^M(B^{(i)}) = \eta(B^{(i)})$, obtained on his funded projects $(\pi^{(i)}_{\mu(1)}, \pi^{(i)}_{\mu(2)}, \ldots \pi^{(i)}_{\mu(\zeta(i))})$.

It can be provided via keeping $\mu(j) \le \rho(i)$, $1 \le j \le \zeta(i)$, that leads to the requirement $\upsilon(\pi^{(i)}_{\mu(\zeta(i))}) \ge \upsilon(\pi^{(i)}_{\rho(i)})$, assuming $\upsilon(\pi^{(i)}_{\mu(1)}) \ge \upsilon(\pi^{(i)}_{\mu(2)}) \ge \ldots \ge \upsilon(\pi^{(i)}_{\mu(\zeta(i))})$ and having $\eta(\pi^{(i)}_{\mu(j)}) \to max$ within a limitation stated above.

Votes $\upsilon(\pi^{(i)}_{\mu(j)})$ obtained for a malicious project $\pi^{(i)}_{\mu(j)}$ are given both from maliciously controlled and honest voting entities, i.e. $\upsilon(\pi^{(i)}_{\mu(j)}) = \upsilon^M(\pi^{(i)}_{\mu(j)}) + \upsilon^H(\pi^{(i)}_{\mu(j)})$.

The cost of obtaining necessary amount of votes from voting entities, controlled by the attacker, is calculated as $\varphi(\upsilon^M(\pi^{(i)}_{\mu(j)})) = D \times \upsilon^M(\pi^{(i)}_{\mu(j)})$, where D is the market available deposit amount for running the voting entity (for the Dash cryptocurrency $D = 1000$; now a masternode can be bought cheaper if paid by a fiat currency due liquidity limits of the current Dash trade amount) .

The cost of advertising a malicious project $\pi^{(i)}_{\mu(j)}$ among honest users to get $\upsilon^H(\pi^{(i)}_{\mu(j)})$ votes from them is defined as $\phi(\upsilon^H(\pi^{(i)}_{\mu(j)}))$.

The current model considers no governmental agencies like NSA, mounting attacks to the treasury system (i.e., in the absence of "goldfinger" attacks).

The cryptocurrency treasury security condition is

$$\varphi(\upsilon^M(\pi^{(k)}_{\mu(j)})) + \sum_{k \in \mathfrak{I}} (\phi(\upsilon^H(\pi^{(k)}_{\mu(j)}))) > \Xi(\varphi(\upsilon^M(\pi^{(k)}_{\mu(j)}))) + \sum_{k \in \mathfrak{I}} \eta^M(B^{(i)}),$$

where $\mathfrak{I}$ is a set of all reward epochs during the attack,

and $\Xi(\varphi(\upsilon^M(\pi^{(k)}_{\mu(j)})))$ gives the residual value of maliciously controlled voting entities cost after the attack (which may change the cryptocurrency exchange rate, thus changing the residual value).

10. Appendix B. Difference in Funding Payments Between Blockchain and Web

The table below shows the differences between funding payments as published on the web and those stored in the blockchain. The data is consistent except several payments (they are marked as cancelled on the web while in the blockchain they are accepted and paid).

from website dashninja.pl		from *. dat files of Dash core wallet		The difference between website data and blockchain data
time stamp	Budget Payment	time stamp	Budget Payment	
1475657128	145,000	1475657128	145,000	0
1475657051	250,39	1475657051	250,390	0
1475656963	83,64	1475656963	83,640	0
1475656908	28,92	1475656908	28,920	0
1475656825	231,260	1475656825	231,260	0
1475656143	361,770	1475656143	361,770	0
1475656059	26,000	1475656059	26,000	0
1475656022	100,000	1475656022	100,000	0
1475655743	527,150	1475655743	527,150	0
1475655731	94,030	1475655731	94,030	0
1475655606	1733,460	1475655606	1733,460	0
1475655191	600,520	1475655191	600,520	0
1475654992	333,330	1475654992	333,330	0
1475654907	1000,000	1475654907	1000,000	0
1475654631	610,260	1475654631	610,260	0
1475654492	123,000	1475654492	123,000	0

1475654008	1176,000	1475654008	1176,000	0
1473041502	225,000	1473041502	225,000	0
1473041382	100,000	1473041382	100,000	0
1473041266	80,000	1473041266	80,000	0
1473041056	159,450	1473041056	159,450	0
1473040482	464,660	1473040482	464,660	0
1473040459	10,000	1473040459	10,000	0
1473039333	748,990	1473039333	748,990	0
1473038534	91,000	1473038534	91,000	0
1473038416	333,330	1473038416	333,330	0
1473038335	1497,510	1473038335	1497,510	0
1473038019	27,980	1473038019	27,980	0
1473037866	211,870	1473037866	211,870	0
1473037625	1044,100	1473037625	1044,100	0
1473036742	23,000	1473036742	23,000	0
1473036325	51,730	1473036325	51,730	0
1473036298	355,740	1473036298	355,740	0
1473036091	66,170	1473036091	66,170	0
1473035390	610,260	1473035390	610,260	0
1473034929	123,000	1473034929	123,000	0
1473034918	1176,000	1473034918	1176,000	0
1470423613	43,670	1470423613	43,670	0
1470423139	225,000	1470423139	225,000	0
1470422914	350,000	1470422914	350,000	0

1470422823	700,000	1470422823	700,000	0
1470422511	80,000	1470422511	80,000	0
1470422489	651,410	1470422489	651,410	0
1470422222	444,330	1470422222	444,330	0
1470421776	288,580	1470421776	288,580	0
1470421483	57,850	1470421483	57,850	0
1470421108	430,370	1470421108	430,370	0
1470420642	1293,990	1470420642	1293,990	0
1470420180	651,410	1470420180	651,410	0
1470419238	199,750	1470419238	199,750	0
1470419101	123,000	1470419101	123,000	0
1470418783	610,260	1470418783	610,260	0
1470418768	1176,000	1470418768	1176,000	0
1467805514	775,000	1467805514	775,000	0
1467805378	133,980	1467805378	133,980	0
1467805120	40,590	1467805120	40,590	0
1467805096	352,970	1467805096	352,970	0
1467804963	105,000	1467804963	105,000	0
1467804334	955,170	1467804334	955,170	0
1467804261	416,020	1467804261	416,020	0
1467804147	41,900	1467804147	41,900	0
1467803490	270,570	1467803490	270,570	0
1467803459	742,920	1467803459	742,920	0
1467803370	43,680	1467803370	43,680	0

1467802729	609,450	1467802729	609,450	0
1467802389	603,590	1467802389	603,590	0
1467802007	449,690	1467802007	449,690	0
1467801998	123,000	1467801998	123,000	0
1467801513	1176,000	1467801513	1176,000	0
1467801483	610,260	1467801483	610,260	0
1465188572	775,000	1465188572	775,000	0
1465188540	78,680	1465188540	78,680	0
1465188469	490,490	1465188469	490,490	0
1465188209		1465188209	104,010	-104,01
1465188113		1465188113	42,480	-42,48
1465188058	701,670	1465188058	701,670	0
1465187486	950,000	1465187486	950,000	0
1465187475	45,800	1465187475	45,800	0
1465187435	150,000	1465187435	150,000	0
1465187423	48,870	1465187423	48,870	0
1465187412	2156,000	1465187412	2156,000	0
1465187368	610,260	1465187368	610,260	0
1465187309	1176,000	1465187309	1176,000	0
1462575966	775,000	1462575966	775,000	0
1462575740	701,670	1462575740	701,670	0
1462575452	45,800	1462575452	45,800	0
1462575409	48,870	1462575409	48,870	0
1462574977	158,370	1462574977	158,370	0

1462574677	150,000	1462574677	150,000	0
1462574327	1053,070	1462574327	1053,070	0
1462574217	2156,000	1462574217	2156,000	0
1462574188	610,260	1462574188	610,260	0
1462574053	1176,000	1462574053	1176,000	0
1459960939	775,000	1459960939	775	0
1459960912	250,000	1459960912	250,000	0
1465188209	5,000	1465188209	5,000	0
1459959832	701,670	1459959832	701,670	0
1459959453	45,800	1459959453	45,800	0
1459958881	150,000	1459958881	150,000	0
1459958600	788,000	1459958600	788,000	0
1459958584	95,000	1459958584	95,000	0
1459958123	2156,000	1459958123	2156,000	0
1459957397	100,000	1459957397	100,000	0
1459956695	610,260	1459956695	610,260	0
1459955596	1176,000	1459955596	1176,000	0
1457338702	55,000	1457338702	55,000	0
1457338628	400,000	1457338628	400,000	0
1457338184	50,000	1457338184	50,000	0
1457338128	717,700	1457338128	717,700	0
1457337669	2156,000	1457337669	2156,000	0
1457337191	2100,000	1457337191	2100,000	0
1457336759	100,000	1457336759	100,000	0

1457336379	1176,000	1457336379	1176,000	0
1457335992	610,260	1457335992	610,260	0
1454717937	140,000	1454717937	140,000	0
1454717771	200,000	1454717771	200,000	0
1454717520	717,700	1454717520	717,700	0
1454717435	2156,000	1454717435	2156,000	0
1454717197	100,000	1454717197	100,000	0
1454717026	2100,000	1454717026	2100,000	0
1454716900	1176,000	1454716900	1176,000	0
1454716538	610,260	1454716538	610,260	0
1454716236	5,000	1454716236	5,000	0
1452098440	1100,000	1452098440	1100	0
1452098332	140,000	1452098332	140,000	0
1452098215	150,000	1452098215	150,000	0
1452098176	2100,000	1452098176	2100,000	0
1452097886	590,000	1452097886	590,000	0
1452097712	2156,000	1452097712	2156,000	0
1452096838	1176,000	1452096838	1176,000	0
1452096736	610,260	1452096736	610,260	0
1446858126	1787,790	1446858126	1787,790	0
1446857957	438,500	1446857957	438,500	0
1446857704	1952,230	1446857704	1952,230	0
1446857567	2156,000	1446857567	2156,000	0
1446857324	1176,000	1446857324	1176,000	0

1444240907	127,000	1444240907	127,000	0
1444240810	1417,000	1444240810	1417,000	0
1444240791	257,682	1444240791	257,682	0,00033911
1444240386	2024,000	1444240386	2024,000	0
1444240319	308,650	1444240319	308,650	0
1444239949	1176,000	1444239949	1176,000	0
1444239665	2156,000	1444239665	2156,000	0
1441622648	2156,000	1441622648	2156,000	0
1441622598	2529,000	1441622598	2529,000	0
1441622431	1176,000	1441622431	1176,000	0
TOTAL			94086,992	

Table B.1 Verification of Budget Payments

This Data was additionally verified with an outside (non-Dash) blockchain explorer (https://bitinfocharts.com/darkcoin/block/), with the results being the same as found in the .dat files of the Dash core wallet (as shown in the Table B.1). Our evaluation engaged three separate sources of blockchain data (Dash site data, Bitinfocharts, and raw binary file data of the blockchain itself).

11. Appendix C. The DAO and Other Self-Governing Cryptocurrency Systems

11.1. C.1. The DAO System

11.1.1. C.1.1. Introduction

Decentralized Autonomous Organization (DAO) allows its participants to automate control of their funds and formalize governance rules. It is achieved by using blockchain technology and smart contracts functionality.

DAO code is written in the "Solidity" programming language (it is a Turing complete language). A DAO is activated by deployment on the Ethereum blockchain.

All information below were taken from whitepaper and source code [1,2].

The DAO lifecycle can be presented in the following stages:

1) **Creation Phase**: creation of the minimum predetermined number of tokens (it is called a minimum DAO Creation Goal). During this time period holders buy tokens for ether. The token grants its holder voting and ownership rights. The number of tokens created is proportional to the amount of ether transferred. If the minimum DAO Creation goal is not reached at the close of the Creation Phase, all ether is returned.
2) **"Proposal" Phase**. After the Creation Phase any DAO Token Holder may become a Contractor by submitting proposals to use a DAO's ether. Members of a DAO cast votes weighted by the amount of tokens they control.

A lifecycle of proposal can be described in the following way (see Figure C.1).

Figure C.1. A lifecycle of proposal

All DAO's operations (creation of tokens, creation of proposals, voting etc.) are performed by using ***smart contracts***. Each contract has member variables and functions which can be externally called by sending a transaction to the Ethereum network with the DAO contract address as the recipient and the method ID (optional with parameters) as data.

11.1.2. C.1.2. Creation Phase

11.1.2.1. C.1.2.1. Token Creation

There are two smart contracts concerning tokens:

- TokenCreationInterface that defines how the DAO token is created;
- TokenInterface that defines inner workings of the DAO token.

TokenCreationInterface has variables and functions described in Table C.1.

Parameter	Description
closingTime	Unix time at which the Creation Phase ends
minTokensToCreate	The number of wei equivalent tokens which are needed to be created by the DAO in order to be considered fueled
isFueled	The boolean that is true if DAO has reached its minimum fueling goal, false otherwise
privateCreation	The address is used for DAO splits - if it is set to 0, then it is a public Creation, otherwise, only the address stored in privateCreation is allowed to create tokens
extraBalance	The managed account is used to hold the excess ether which is received after the Creation rate is decreased during the Creation Phase
weiGiven	The amount of wei given by each contributor during the Creation Phase and is only used to refund the contributors if the Creation Phase does not reach its fueling goal
TokenCreation	The constructor initiates the Creation Phase with the arguments minTokensToCreate, closingtime and privateCreation, which will be set in the constructor of the DAO contract which is only executed once, when the DAO is deployed

Table C.1

In order to interact with the contract the following functions can be used:

- createTokenProxy. This function creates one unit of the DAO tokens minimum denomination for every wei sent.
- refund. This function can be called by any contributor to recieve their wei back if the Creation Phase failed to meet its fueling goal.
- divisor. This function is used to calculate the price of the token during the Creation Phase in the function createTokenProxy.

The events FuelingToDate, CreatedToken and Refund are used to inform lightweight clients of the status of the Creation Phase.

11.1.2.2. C.1.2.2. Token Price

An amount of ether is represented in its base unit ***wei***. This is defined as 1 Wei = 10^{-18} Ether. Similarly, DAO tokens are denoted with T and always represent the amount of DAO tokens in its base unit, dened as 10^{-16} DAO token.

DAO Token Creation rate decreases over time. This reflects an assumption that early acts of DAO Creation have greater risk, as they may have less information about the potential success of the DAO and do not know whether what contribution will lead to full fueling of the DAO. The DAO has the Creation schedule defined in Table C.2.

The number of tokens (in its base unit) each person Creates is calculated as: $P(t) \cdot \Xi_c$. Here Ξ_c stands for the amount of ether sent to fuel a DAO, denoted in wei.

$P(t)$	**Time period**
1	From the beginning of the Creation Phase and till 2 last weeks
$1 + 0.05 \cdot m(t),\ m(t) = (t - (t_c - 2 \cdot w))/d,$ t is the unix time in seconds; t_c is the closing time of the fueling period (end of token creation, in Unix time); w is a week in seconds; d is a day in seconds.	From the beginning of 2 last weeks till 4 last days (the amount of ether required to Create DAO tokens increases daily by $0.05\Xi_c$ per base unit of DAO token)
1.5	From the beginning of 4 last days till the end of the Creation Phase

Table C.2

11.1.3. C.1.3. “Proposal” Phase

A token holder who submits a proposal called Contractor.

There is a smart contract called DAOInterface that allows to execute such operations as creation of new proposal (to spend ether or elect new Curator), voting for proposals, split DAO, execute proposals and others.

11.1.3.1. C.1.3.1. Types of Proposals. DAO split

A token holder can submit three types of proposals:

- a proposal how to spend ether;
- a proposal to split DAO;
- a proposal for Curator.

To prevent a situation when an attacker with 51% of the tokens creates a proposal to spend all money, the minority can split a DAO into two. If an individual, or a group of token holders, disagree with a proposal and want to retrieve their portion of the ether before the proposal gets executed, they can submit and approve a special type of proposal to form a new DAO. The token holders that voted for this proposal can then split the DAO moving their portion of the ether to this new DAO, leaving the rest alone only able to spend their own ether.

There is a problem: some token holders might not be actively involved in their DAO and might not follow proposals closely and an attacker could use this to their advantage. The proposed solution is implemented by limiting each individual DAO to a single **Curator**. This Curator controls the list of addresses that can receive ether from the DAO, across all proposals. This gives the Curator of a DAO considerable power. To prevent the abuse of this power, it is possible for a DAO to vote for a new Curator, which may result in a split of the DAO as described above.

11.1.3.2. C.1.3.2. Creation of Proposal

11.1.3.3. A single proposal is defined by struct Proposal that has parameters described in Table C.3.

Parameter	Description
recipient	The address where the amount of wei will go to if the proposal is accepted
amount	The amount of wei to transfer to recipient if the proposal is accepted
description	A plain text description of the proposal
votingDeadline	A unix timestamp, denoting the end of the voting period
open	A boolean which is false if the votes have already been counted, true otherwise
proposalPassed	A boolean which is true if a quorum has been achieved with the majority approving the proposal
proposalHash	A hash to check validity of a proposal. Equal to sha3(_recipient, _amount, _transactionData)
proposalDeposit	The deposit (in wei) the creator of a proposal has send to submit a proposal
newCurator	A boolean which is true if this proposal assigns a new Curator
splitData	The data used to split the DAO. This data is gathered from proposals when they require a new Curator
yea	Number of tokens in favor of the proposal
nay	Number of tokens opposed to the proposal
votedYes	Simple mapping to check if a token holder has voted for it
votedNo	Simple mapping to check if a token holder has voted against it
creator	The address of the token holder that created the proposal

Table C.3 (previous page)

The purpose of a **proposal deposit** is to prevent spam. Its minimal value is set when the contract is deployed as constructor parameter. But the creator of the proposal can send any amount above this for the deposit.

The proposals will be sorted by the proposal deposit in the GUI, so in the case that a proposal is considered important, the creator of the proposal can deposit extra ether to advertise their proposal. The creator of the proposal will be refunded the entire deposit if quorum is reached, if it is not reached the deposit remains with the DAO.

A creation of proposal is made by newProposal call. The arguments of the function are described in Table C.4.

Argument	Description
recipient	The address of the recipient of the ether in the proposal (has to be the DAO address itself, the current Curator or an address on the whitelist allowedRecipients)
amount	The amount of wei to be sent in the proposed transaction
description	A string describing the proposal
transactionData	The data of the proposed transaction
debatingPeriod	The amount of time to debate the proposal, at least 2 weeks for a normal proposal and at least 1 week for a new Curator proposal
newCurator	A boolean defining whether this proposal is for a new Curator or not

Table C.4

Any token holder can make a proposal to vote for a new Curator. The process of choosing a new Curator is as follows:

- Any token holder can submit a proposal for a new Curator. In this case, no proposal deposit is required, because an attacker could vote for an extremely high deposit, preventing any splits.
- The debating period for this proposal is 7 days. This is 7 days less than the minimum required for regular proposals, allowing anyone to retrieve their funds before a potentially malicious proposal goes through. There is no quorum requirement, so that every token holder has the ability to split into their own DAO. The debating period is used to discuss the new Curator and conduct a rst vote that's non-binding. After this rst vote, token holders can confirm its results or not. In the case a majority opts to keep the original Curator, the minority can either keep the original Curator in order to avoid a split, or inversely they can confirm their vote for a new Curator and move their portion of the ether to a new DAO.

11.1.3.4. C.1.3.3. Debating and Voting

During **2 weeks** token holders can debate and vote for/against proposals. There is a special function in DAOInterface contract that called vote and has two arguments:

- _proposalID;
- _supportsProposal (true - if token holder supports proposal, false - otherwise).

To be accepted a proposal needs to get a minimum quorum of token votes which is calculated using the next formula:

$$q_{min} = \frac{T_{TOTAL}}{d} + \frac{\Xi_{transfer} \cdot T_{TOTAL}}{3 \cdot (\Xi_{DAO} + R_{DAO})},$$

where T_{TOTAL} - total amount of tokens created;

d - minQuorumDivisor (default value is 5, but it will double in the case the quorum has not been met for over 25 weeks);

$\Xi_{transfer}$ - the amount of ether Contractor propose to spend;

Ξ_{DAO} - the amount of ether owned by the DAO;

R_{DAO} - the amount of reward tokens (tokens that DAO receives when ether are spent (see subsection C.1.3.4)).

A quorum of **20%** of all tokens is required for any proposal to pass. In the event transfer equals the amount of ether a DAO has ever received ($\Xi_{transfer} = \Xi_{DAO} + R_{DAO}$), then a quorum of **53.33%** is required.

In order to prevent "proposal spam", a **minimal deposit** (should be > 0) can be required to be paid when creating a proposal, which gets refunded if quorum is achieved. The minimum deposit (in wei) required to submit any proposal that is not requesting a new Curator (no deposit is required for splits).

The **maximum proposal deposit** that can be given. is a fraction of total ether spent plus balance of the DAO (maxDepositDivisor = 100).

If quorum is not achieved, the DAO keeps the proposal deposit. The value of the proposal deposit can be changed from the default value by the DAO through another proposal.

The quorum for proposal is calculated as sum of positive and negative votes.

$$quorum = yea + nay.$$

To be accepted a proposal needs $quorum > q_{min}$ and $yea > nay$.

11.1.3.5. C.1.3.4. Reward Tokens

Reward tokens are generated when the DAO makes any transaction spending ether. When the DAOs products send ether back to the DAO, the ether is held within DAOrewardAccount. The DAO can use these rewards to fund new proposals or to fairly distribute the rewards to the reward token holders (using a proposal which gets voted on by the DAO token holders).

Reward tokens are used to divide the ether sent to DAOrewardAccount amongst the various DAOs that own reward tokens. Reward tokens are only transferred in the event of a DAO split or an update of the contract, they can never be owned by anything other than the original DAO or a fork of the original DAO that generated the reward tokens.

11.1.4. C.1.4. Conclusions

The main **advantages** of the DAO system can be considered the following:

- All information about proposals, voting and funding are saved in the blockchain by usage of smart contracts mechanizm.
- Each token holder can “save” his money by DAO splitting in case he is against some funding solutions, malicious activity etc.
- Ether return to the DAO, the DAO in general and token holders get rewards from funded projects.
- The votes number minimum depends on the amount of money is proposed to fund to project.
- The weight of vote (as well as the number of the reward tokens) is proportional to the amount of tokens a token holder possesses, so it is some kind of “Proof-of-Stake”.

The DAO system has some particular qualities compared with ordinary treasury systems. For example, the Dash problems connected with participation incentive are not presented in the DAO because token holders invest their own money so they have a very high incentive to examine and vote for the proposals. At the same time, it seems some features of the DAO (for example, getting rewards) can be implemented in the DGS. The possibilities of such implementation is still under analysis.

11.2. C.2. Decred

11.2.1. C.2.1. General

Decred is a hybrid PoW and PoS cryptocurrency which is self-funding. There are two entities in Decred: PoW miners who issue new blocks and PoS miners (or voters) who vote for these blocks.

The block reward is divided among different parties in the next way [3,4]:

- 60% of block reward goes to PoW miner;
- 30% of block reward goes to PoS miners (5 stakeholders are randomly selected to vote for a block, so 1 vote gets 6% of block reward);
- 10% of block reward goes to treasury (development subsidy).

Due to the date block reward is 27.96 DCR (it is decreased during the time) so the development subsidy is 2.7961 DCR per block [5].

For today total amount of money is near 3 634 000 (from the total 21 000 000 that are planned to issue), and developers have already got more than 195 000 DCR as a subsidy [5].

11.2.2. C.2.2. Self-funded development and proposals

All coins of development subsidy go to development organization called Decred Holdings Group, LLC (DHG) that follows the next principles [4]:

- DHG shall issue public financial statements every six months, starting March 8th, 2016. The frequency of financial statements may increase with activity, but it shall not occur more often than quarterly.
- DHG shall put forth a budget proposal each year on March 8th, after the corresponding public financial statement has been issued.
- The Funding Council shall review, propose changes, make changes, and ultimately approve the proposal by April 8th, one month from the initial budget proposal.
- Final approval of the budget via **proof-of-assembly** ("PoA") vote shall occur after Funding *Council* approval by April 18th, two months from the initial proposal.
- DHG shall make public requests for proposals ("RFPs") for projects that are to be completed by parties on a contractual basis. RFPs shall include a scope and an explanation of how the work shall benefit the project. Parties that submit proposals shall be required to include:
    1) a detailed description of the work to be performed;
    2) a series of milestones that can be verified as work is completed;
    3) a quote for the work, itemized by milestone, in U.S. Dollars ("USD").
- All proposals, both submitted and accepted, shall be made public one week after a proposal has been selected. Once the selection occurs, the associated RFP shall be removed. Contracted parties shall be paid exclusively in *decred* ("DCR") at the current effective DCR/USD rate at the time of payment, unless specifically noted otherwise.
- In the future, the development organization may need to change from DHG to another entity that serves an identical function. If and when this occurs, DHG shall transfer all assets to the new entity and the development subsidy shall be directed to the new entity.

11.2.3. C.2.3. About Proof-of-Assembly

Off-chain decision-making shall be used to resolve disputes related to development and voted on by the *Decred Assembly* as they arise, as an effective PoA, until such time PoA is integrated into the blockchain [4].

The *Decred Assembly* shall be composed of diverse Assembly members who are selected for membership by the *Admission Council* from the project ecosystem for representation.

Councils that are composed of Assembly members shall be formed to address ongoing and episodic matters. The initial Councils shall serve the separate functions of admission (*Admission Council*), creation (*Creation Council*), and attrition (*Attrition Council*).

The *Admission Council* shall vote on the inclusion of new members into the Assembly. All additional Councils shall be created by the *Creation Council*. The *Attrition Council* shall be responsible for deactivating both Councils and Assembly members as necessary.

Membership of the *Decred Assembly* shall consist of Assembly members who have been confirmed by a 60% or greater affirmative vote by the *Admission Council*. There is no restriction on the age or nationality of Assembly members, the only requirement is that of merit as judged by the *Admission Council*. Merit is judged on the basis of two characteristics:

1) the amount of time over which one has been involved with the project;

2) one's body of work and its impact in the context of the project.

Attrition is embraced by temporarily deactivating or actively expelling Assembly members by a 60% or greater affirmative vote by the *Attrition Council* on the basis of:

1) substantial non-fulfillment of duties for one or more Councils or the Assembly;

2) counterproductive behaviour that goes against the framework set forth in the Constitution without constructive action toward solutions.

All matters formally presented to a Council shall be resolved by a vote in 365 days or less.

The members of Assembly are the most meritorious members of the Decred community (but who are these people actually?). While the voting process isn't implemented using blockchain, perhaps it can't be considered democratic.

11.2.4. C.2.4. A bit more about requests for proposals

Some important points are below [6].

- DHG posts a new RFP on the Decred GitHub repository: https://github.com/decred/rfps. This content will be syndicated on the forum (https://forum.decred.org/forums/requests-for-proposals/) by the developer(s) who are the most informed about that particular RFP.
- Interested parties can review these RFPs and submit a proposal to rfp2016<at>decred.org. A proposal will include a detailed description of the work, a series of milestones and a quote, itemized by milestone.
- As the work is completed and milestones demonstrated, contractors shall be paid in Decred for each milestone.
- DHG is open to having either individuals or corporate entities submit proposals. Some RFPs may have a scope that requires multiple people working together, either as individuals or as a corporate entity.
- Development work will always be paid for in arrears and correspond with the completion of tangible deliverables.
- Proposals are selected based on a number of criteria that includes price. The lowest price is not guaranteed to get a proposal accepted. We aim to balance quality, timeliness, complexity and price.

11.3. C.3. Litecred

Litecred [7] is a new cryptocurrency with the characteristic of self-funded development, via block subsidy of 10% for each minted block. That 10% will be not directly usable at the beginning of the development stage but it will be achieved at PoW end, this means we are going to receive only around 0.8% every month. This will guarantee a small and continuous flow to the developers, in order to maintain the project active. There is no developer premine at launch. Funds collected from mining will be held in a public address and will be used for future development and/or given to users with proven skills that will help us with improvements/marketing and promotion.

Litecred wants to be intended as Decred side-project. Considering Decred as Bitcoin 2.0, Litecred wants to be Litecoin 2.0. However, it seems that Litecred has failed to develop (no access to website, it is absent on Crypto-Currency Market Capitalizations).

11.4. C.4. References

[1] Whitepaper: DECENTRALIZED AUTONOMOUS ORGANIZATION TO AUTOMATE GOVERNANCE.
[2] https://github.com/slockit/DAO
[3] https://wiki.decred.org/Introduction
[4] https://wiki.decred.org/Decred_Constitution
[5] https://dcrstats.com/
[6] https://forum.decred.org/threads/requests-for-proposals.683/
[7] https://bitcointalk.org/index.php?topic=1317122.0