This is a major algorithm of a chain-based proof-of-stake consensus protocol using the process of cyclical voting. It applies a finalization period in about one round of voting with a relatively simple mechanism. I think it might be considered in optional schemes as a supplementary in the future. The following is a brief introduction of it and there is the link of the full paper at the bottom of this post if someone is interested.

Consensus processes

Consensus processes are divided into the following steps:

(1) Vote generation: Stakeholders generate votes using the probability based on the stakes held, gain the qualification for voting, and become miners to build blocks.

(2) Competition for building blocks: Miners generate blocks and gain awards using the probability based on the generated number of votes.

(3) Confirmation of canonical chain: Miners vote on blocks in cycles, and the branch winning the most votes is recognized as the canonical chain.

(4) Block finalization: When the count result meets certain conditions, this result cannot be changed under a given assumption (2/3 of the nodes are honest). Then, the situation is called block finalization, the branch finalized is the final canonical chain, and the data in finalized blocks are the final data.

Votes Generation

Stakeholders perform operations on random numbers from the current constants (block hash, timestamp, etc.) and compare them with the number of their stakes to meet a condition, in which the demander qualifies for voting. This condition is denoted as

VRF_VoteProof < Account_stakes

,

where Account_stakes

refer to the stakes of the account. The higher the number of stakes, the higher the possibility to meet this condition, generating votes according to stakes. The stakeholder who qualifies for voting generates a vote and publishes it on the network to be registered as a miner (or voter). The registration record is packed into the block as the transaction record.

Introducing qualification for voting decreases and flexibly controls the number of validators so that different projects strike a balance between safety and response speed.

Competition and generation of blocks

Stakeholders participate in the building and maintenance of the blockchain after becoming miners. To gain awards, miners would have to compete with each other for the right to build blocks:

(1) A random number operation is conducted based on constants (such as timestamp and block hash) by miners. When the expected results reach a certain target and meet the requirements to generate blocks, it is denoted as

Block_Proof < target * votes * efficiency

,

where target

refers to difficulty, which controls the speed of block generation; votes

refers to the votes owned by miners and is directly proportional to the possibility of block generation; and efficiency

refers to the efficiency of block generation, which is affected by various factors and which will be later introduced in detail.

(2) When the requirements for block generation are met, the miner will pack the voting records along with the transactions that he received into a block and will publish them on the network.

Confirmation of canonical chain

Starting with the Genesis block, miners in cycles (e.g., “10 blocks” or “120 s”) select a branch from the previous canonical chain, sign the address of the top block, and publish the result on the network, which is called the voting. Voting records are packed into the block like transactions. The process of confirming the current canonical chain compares the priority of branches from the voting records and determines the branch (also a chain) with the highest priority.

Priority of branches

The weight of a branch is the number of total votes gained by the root and all its descendants. From this calculation method, the comparison of two chains in priority begins from their common ancestors and the weight of the respective branches they belong to be calculated. The heavier one takes priority. To find the canonical chain, we should compare the existing competitive branches one by one and find the chain with the highest priority (as shown in Fig. 1).

[

q_001

724×513 17.4 KB

](https://ethresear.ch/uploads/default/original/2X/e/e7e3b6d19f88d07b44602ff2318845738311e60b.png)

Block finalization

The finalization process of chain-based protocols, different from BFT protocols, does not need multiple rounds of voting to keep the strictly synchronized state among nodes. When the first round of voting reaches a consensus, we only need to determine that this consensus is a supermajority decision rather than identifying whether all nodes have received the results. Therefore, finalization can be completed in one round of voting in an ideal condition. However, to guarantee that the finalized chains do not conflict (i.e., safety) and voters have opportunities to amend their votes to avoid situations where the chain cannot be finalized forever (i.e., liveness), the situation becomes complex, requiring new voting rules and resulting in delayed finalization.

New voting rules

From the mechanism of cycle voting, when dishonest users do not exceed 1/3, we determine that provided one branch wins more than 2/3 of total votes during one cycle, this branch should not have competitors and could be finalized. However, to guarantee that all finalized branches do not conflict with each other, we need a continuous finalization link starting from the Genesis block. This also requires that the votes are consistent. Thus, we modify the voting rule by adding the following rules on the first voting rule:

(1) The interval between the target blocks of two votes taken by a miner is larger than or equal to one voting cycle; otherwise, the miner will be punished.

The additional rules are:

(2) Each vote requires a backward connection to the voter’s previous vote called the “source.” The votes without connection to the source are called “sourceless votes” that do not weigh anything as the amendment to wrong votes. However, it can serve as the source of later votes.

(3) The Genesis block is the first finalized block and can be used as the source of later votes without violating other rules.

(4) Votes whose source is located at the finalized block are called “rooted votes.” Only rooted votes have weight, thus progressively producing subsequent finalized blocks. Meanwhile, the ancestor blocks of the finalized block are also regarded as finalized. The “rooted votes” can be transmitted forward. The details mentioned here are presented in Fig. 3 and Fig. 4.

*(Transmission means that if c is a rooted vote, the vote connecting c as a source is also rooted.)

[

q_003_004

877×794 43.6 KB

](https://ethresear.ch/uploads/default/original/2X/5/5807dbf32e623549df436a400bb16cd56c396e0f.png)

Safety and liveness

The principle to guarantee safety is simple: the voter only connects each vote to the next one, and thus, votes are unkept as an end-to-end link (as shown in Fig. 3) if there exist conflicting votes, so his voting link will not have conflict. Now that the voting links of single voters do not have conflicts, the finalization points generated based on the continuous voting links will not conflict either, thus guaranteeing safety (as shown in Fig. 5).

[

q_005

734×397 13.8 KB

](https://ethresear.ch/uploads/default/original/2X/6/631a53ef67cec4d1d239a7a68709bd9042aa0123.png)

However, this cannot meet the requirements for liveness because it is very likely that more than 1/3 of votes will be taken on wrong branches after a fork appears. Once such a case appears, the finalization link breaks without recovery. Therefore, to keep liveness, first, become aware of such case—we call it “key fork”; then, we allow the sources of the next votes to move back to the fork position from the current targets. This provides a hesitation space to the voters and allows them to correct their previous decisions and return to the canonical chain (as shown in Fig. 6). Lastly, we delay the finalization point to the fork position.

[

q_006

718×443 21.3 KB

](https://ethresear.ch/uploads/default/original/2X/4/48884d5bd2ca60b186e7d9f45c6dd821543360bc.png)

Awareness and positioning of the key fork

To become aware of the key fork, we adjust some finalization conditions. The previous condition in which “more than 2/3 of the total votes be gained during one voting cycle” is replaced with “more than 2/3 of the total votes be gained during one voting cycle and the distance between the targets and the sources of those votes is less than two cycles.” Therefore, only the branch selected by continuous votes (their intervals are not more than two cycles) can be finalized. This condition concentrates the sources of all votes that can reach finalization in the range of two cycles before the finalization point (as shown in Fig. 7). This way, we need only to inspect whether there are sufficient votes in this range.

[

q_007

830×403 29.9 KB

](https://ethresear.ch/uploads/default/original/2X/6/64a2fe9655ecc31b1b19efb567f11aaaa8e773cc.png)

When a key fork is detected, we find the fork point and the corresponding hesitation period using a backtracking algorithm.

When block a

at height h

is generated, we use a natural number n

to represent the temporary “hesitation period” at the position of a

, and the block at a height of h

– n

is called the temporary “retracement point,” which is generally equal to the fork point. The value of n

is based on the following calculation results:

(i) If the votes gained by the chain within two voting cycles before a

(including) exceed 2/3

of the total votes, n = 0

.

*(For the same voter, only the final vote is calculated.)

(ii) If the abovementioned votes are less than or equal to 2/3

of the total votes, all votes with the target at a height of h

are traced back to a height of h – 1

and traced back to h – 2

with those votes whose target are at a height of h – 1

. By parity of reasoning, when traced back to h – i

, if the condition in Clause (i) is satisfied, n = i

.

(iii) The height of the temporary retracement points at Position a

cannot be lower than that of the actual retracement point at the position two cycles before a

.

(iv) The actual retracement point at Position a

is taken from the earliest temporary retracement point during two cycles after Position a

(including), and the actual hesitation period N

also results from this.

*(Therefore, in the worst case, a block has to wait for two cycles after it is generated until it can be finalized.

However, provided that one chain after this block gains 2/3 of the votes during those two cycles, there will be no retracement point earlier than this block, and we can get the actual retracement point right then so that it can be finalized immediately.)

Voting retracement and delayed finalization

After the hesitation period is gained, the voting retracement and block finalization processes are shown as follows:

(1) Each vote of a miner only uses the block of his previous vote target or the block that traces back within N

steps (including) from it as the source. N

value is taken from the actual hesitation period at the height of the miner's previous voting target in the related chain, and miners who violate this rule will be punished (as shown in Fig. 9).

(2) Count from the root, when a branch obtains more than 2/3

of the total votes in one voting cycle and the distances between the sources and targets of those votes are less than two voting cycles, the finalization condition is satisfied. However, the finalized block is not the branch's root b

but the actual retracement point B

at Position b

(as shown in Fig. 9).

Conclusions

The above sections describe the primary algorithm of this protocol. Because it was designed using the chain consensus and PoS, the algorithm inherited some of their advantages, such as the low energy consumption of PoS, high fault tolerance of chain consensus for the network environment, more validators, and more loose requirements for them. While guaranteeing safety and activity, the algorithm provides the finalization ability with a delay of less than one voting cycle.

Here is the full paper and you could see it for more detail.https://github.com/yj1190590/Consensus