PoAS: A Stake-Based Blockchain Voting Consensus Protocol

Yj1190590[*1]

Abstract. This article introduces a stake-based blockchain voting consensus protocol. In this protocol, a voting process is used to accumulate stakes and then compensate for the main defects of Proof of Stake. The voting process, competition mechanism and the strategy of selecting branches are generally described to introduce the main structure of a low-cost, simple and objective proof-of-stake system with deterministic finality. In addition, other advantages, such as the decentration of development and scalability solution are included.

KEY WORDS

1. Proof of Accumulated Stakes (PoAS). 2. Vote. 3. Finality. 4. Reward distribution.

1. Introduction

Till date, the continuous development of the blockchain consensus protocol is broadly divided into three systems: Proof of Work (PoW), Proof of Stake (PoS), and Byzantine Fault Tolerant (BFT). This article primarily discusses the problems in the PoS system, since for blockchain's most original and important application area—cryptocurrencies––although the PoS system does not have some of the BFT's characteristics like high throughput and low commissions, has more advantages in terms of code complexity, the degree of decentralization, and objectivity. It is also more suitable for providing currency functions that require extremely high security and fault-tolerance. Compared to PoW, which consumes a large amount of energy and has more and more centralized mining pools, PoS has great advantages and it will inevitably replace PoW as long as it solves the two biggest problems: Nothing at Stake (NaS) and centralized wealth distribution.

The NaS problem has existed since the introduction of PoS, and is a considered unavoidable by many. As compared to the PoW model, PoS miners do not need to pay high electricity costs when they are competing; thus, attackers can construct as many branched competitions as possible. As a result, the best strategy for miners is to simultaneously compete with each other in all branches that include security. However, as a matter of fact, the PoS's competitive resources—the total amount of stake––is fixed, which is an attribute that we can use to not only determine the one and only main branch but also have finality, a characteristic that PoW does not have, in a very convenient way. For miners, all votes can be made public using certain methods to prevent multiple votes.

The centralizing tendency of wealth distribution is also an inherent problem of PoS. According to the logic of PoS, Whenever Miners spend the same amount of time, richer Miners earn more; thus, they tend to spend more time working. Thus, rich miners continue to become richer; this process will continue until only the richest users remain in the system. Further, the wealth distribution obeys the Pareto's law. This means most wealth is acquired by few users; therefore, the extent of robbing the poor to feed the rich would be more significant. We solve this through a certain method called "Accumulating stakes" to reduce the working

[1]Yj1190590 (3171228@qq.com)

difficulty of the large number of users with small stakes and ensure that more users can participate in the distribution of wealth without reducing the overall competitive intensity.

"Accumulating stakes" means that the stakeholders assign their stakes to a group of people, in an active or passive way. The ones who receive those stakes will work for the stakeholders to active the network. There are several ways to accumulate stakes, one of them is by using the ability of "network dispersity" which could be the primary method under certain conditions (plenty of transactions and active users) because it is objective and easier to users. About the definition of "network dispersity", consider the following fact: As a greater number of dispersed online terminals work together, the faster they can acquire randomly distributed information in the network due to network latency. This type of "ability" is referred to as "network dispersity". Utilizing network dispersity to accumulate stakes can provide a non-simulatable competitive mechanism.

The consensus using the capabilities of stake and the accumulating processes is named as PoAS (Proof of Accumulated Stakes).

2. Scenario and Characters

The entire network can be considered as a type of canvassing scenario that involves three types of characters according to a node's functions.

Stakeholder. As the owner of the currency, stakeholders are the subject of every transaction in the network. Therefore, the broadcast source node of every transaction is initiated by a stakeholder. Stakeholders are primarily responsible for responding to "canvass" requests from gatherers and publishing transactions along with their votes. A stake held by a stakeholder is considered "votes" in this scenario, and the number of "votes" determines mining competition.

Miner. Miners usually have gatherers that canvass for "votes" throughout the network, and they use these "votes" to compete in block generation. Miners have the responsibility of determining the main branch.

Gatherer(optional). Gatherers are affiliated to the miners. They detect nearby stakeholders and send a "canvass" request as quickly as possible. Gatherers primarily run on network terminals by being embedded in client apps or web sources.

The network scheme (with Gatherers) is shown in Figure 1.

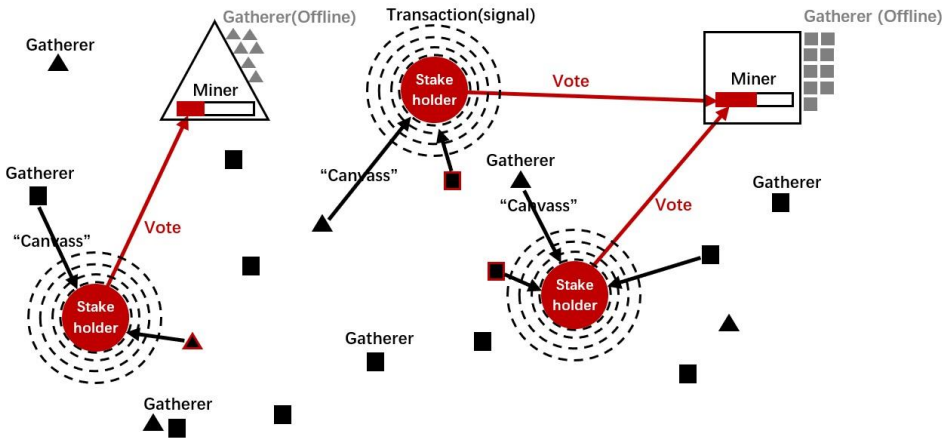

Fig. 1. Network scheme

Miners with more dispersed online gatherers have a greater likelihood of winning votes (i.e., stakes), which helps miners win mining competitions later.[1]

3. Consensus Process

We rely on two types of votes, namely the vote from stakeholders for miners and the vote from the miner node for the current main branch, to reach a consensus that will be introduced below.

3.1.1 The Vote For Miners

The purpose of voting for miners is to accumulate stakes, especially those that are dispersed at the hands of stakeholders with lower stakes. The specific steps of the voting phase are as follows:

(1) (optional) Before each transaction is released, the stakeholder sends a broadcast signal and the gatherer nodes nearby send a "canvass" request to the stakeholder offering the miner(the gatherer' owner)'s account after receiving the broadcast signal.

(2) After receiving the first request (if there is step (1)), the stakeholder writes the miner's account into the transaction structure and broadcasts the transaction.

Voting stakes are used for two purposes: competition for the rights to generate blocks and confirmation of the main branch. Therefore, during counting, the stakes of the two purposes are counted separately and recorded as x stake (the rights to generate blocks) and y stake (main branch). The two types of stakes are equal to the currency amount of each account in the initial status. The specific steps of the vote-counting phase are as follows:

(1) Distribute the x and y stakes of every stakeholders within the last voting cycle (e.g., 6000 blocks) to the miners they voted for. If a stakeholder has multiple votes on one block, then they will be equally divided on each vote;

(2) Record the set of all transactions in the last voting cycle as T, find the relevant transactions of each miner from set T and discard those before his last time of block generation, collect the rest of them recording as set T', $T' \subseteq T$. [2]

(3) In set T', collect the x stakes acquired by each miner in Step (1) and record them as set X.

(4) In set T, collect the y stakes acquired by each miner in Step (1) and record them as set Y.

According to the steps listed above, the stakes are concentrated in the hands of the miners through voting. The results are set X and set Y. For the counting of any time point, such as the block at position *a*, it refers to the statistical results one voting cycle ahead of block *a*, and are recorded as X_a and Y_a.

To control the voting frequency, the voting ability owned by the account (x stake and y stake) must be adjusted separately. For x stake, voting interval of each account is considered as the adjustment coefficient. The stake starts at zero, and a certain proportion is increased every time the interval adds a time unit *t* (e.g., 10 blocks). The stake reaches its maximum value when the interval increases to a certain voting cycle (e.g., ~100 h for 6000 blocks). As for y stake, *t* is equal to the voting cycle, which means that the y stake comes into effect only once in an interval of the voting cycle. Similarly, each UTXO is added with x and y stakes and the transfer interval is used as the adjustment coefficient to adjust the number of voting stakes. The method and parameters are similar to those used for adjusting the corresponding account stakes.

To increase the flexibility of voting, we should use transactions only for voting, and this does not require additional trading information. However, such a voting transaction does not have a transaction fee to reward the miners. Therefore, we need to provide a solution that can act as an incentive. We consider the total valid stakes of all voting transactions in the block as a parameter, which will affect the next block generation interval. As the stake increases, the calculation period will be reduced, e.g., if the calculation period is 0.9–1.1 s, this would

directly affect the generation speed of the next block. In this case, it would be better to pack maximum votes. This parameter should be occasionally adjusted as per the average value .

3.1.2 Block Competition

The purpose of the competition is to determine the miner that generates the next block; the miner operates as follows:

(1) The miner performs mathematical operations based on constants such as timestamp and personal signature. If the expected result meets goals and requirements of block generation, it is recorded as hashProof()<target*d*effective(x), where target refers to the goal; d refers to the difficulty adjustment parameter [3]; x refers to the number of x stakes ($x \in X$) obtained by the current miner. The effective() function calculates the valid part of x, which will be introduced in detail in section 3.2.2.

(2) After the requirements of block generation are met, the miner packs the transactions received during this time period, generates the block, and posts it online. The earnings of each node and other calculation parameters are also packed simultaneously. The earnings obtained during mining are distributed pro rata to the miner and all the stakeholders involved.

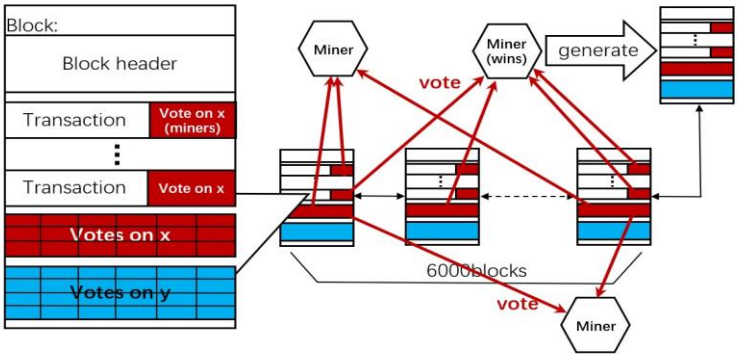

Fig. 2. Process of x-votes taking effect

The system counts votes in existing blocks, and the miner with the maximum number of votes has the greatest chance to win the competition.

3.2.1 The Vote For Main Branch

In PoAS, the priority of branches does not depend on their length, but instead on their weight, which is not equal to the number of blocks but equals the votes obtained by the block. Refering to the description in the GHOST protocol, we defines the weight of a PoAS branch as "the sum of votes obtained by all the descendant of the root of the branch."

To determine the main branch, the miners cyclically sign the top block of the current main branch and publishes the data to the network (for example, 60 blocks is one cycle. Recorded as "cycle2" to distinguish from the other voting cycle.). This data is the vote for the main branch, which is packaged into the block like a transaction. For these votes, the miners who generated the blocks will collect as many votes as possible into the blocks. Miners collecting voting information aligns with their own interests, as collecting the votes on the current branch can increase the branch's weight; in addition, collecting votes on other branches can reduce the competitive power of the corresponding miners in the current branch (this will be elaborated upon later in the next section). For the blocks generated in other branches, the miners will also try to collect as many references (block headers) as possible—which aligns with their own interests too—because doing so will significantly reduce the competitive power of the miners who generate these blocks (this will be elaborated upon later in the next

section). As a result, the blocks of each branch essentially contain the nodes and voting status of all branches, so each chain only need to count them once from the perspective of itself.

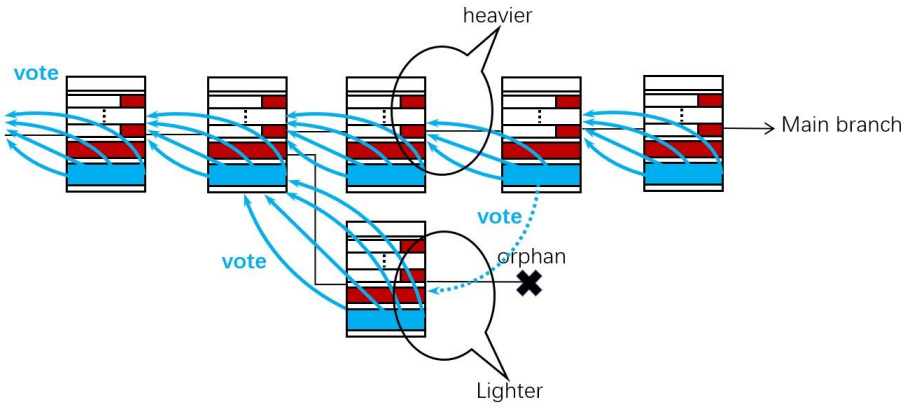

Fig. 3. Process of y-votes taking effect

When a fork occurs, the miners vote between the branches and the votes are recorded in the next block. This determines the number of votes (stakes) for both branches. This also determines the weight of branch. The heaviest branch is the main branch.

Suppose that the root of a branch to be calculated is c, d is a leaf node, and the length of c→d is shorter than one voting cycle2. We need to calculate the weight of this branch from the perspective of c→d. The calculation method is as follows：

(1) First, count Y_d, and then distribute the y stake of every miner in Y_d to each vote that recorded in c→d equally;

(2) Count the y stake of the vote that points to the descendants of c, then add up all the results, which is the weight of this branch.

If the length of c→d is longer than one voting cycle2, it needs to be separated into small segments using one voting cycle2 as the unit and then calculated by segments. According to the calculation method, each miner can only vote once for any branch shorter than one voting cycle2. Thus, if the branch obtains more than half of the total stakes of the whole system within a voting cycle2, it is impossible to have a competitive branch. We denoted the branch as "Finalized," and the block at the root of the branch as a "save point." All the blocks before the save point are irreplaceable, and all the blocks after the save point must be its descendants. The genesis block is the first save point, and the remaining save points are established based on the previous save point. The method is as follows:

(1) Assuming p is the latest save point and b_i is the newly generated block that is p's descendant, compare the priorities of b_i to determine the current main branch;[4]

(2) Assuming the new block on the head of the main branch is b, p_j is its ancestor and the length of p_j→b is shorter than one voting cycle2. Calculate the weight of branch with p_j as the root successively; when it exceeds 2/3 (ensure an appropriate fault-tolerance) of the total stakes, p_j becomes the new save point. Set $p = p_j$ and return to Step (1).

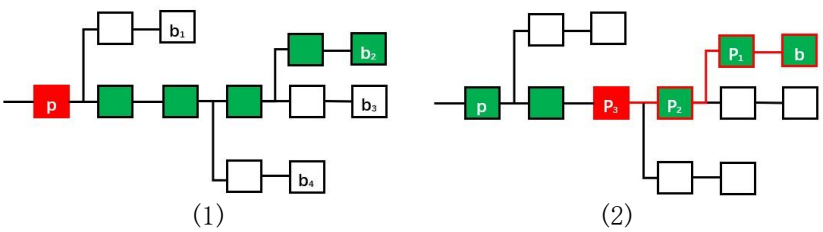

Fig. 4 Process of determining the main branch (1) and save point generation

3.2.2 Incentives and penalties

To ensure perspective consistency across all chains, it is necessary to motivate the miners to actively record votes and blocks from all branches. To achieve this, we consider the following incentives and penalties:

(1) If a miner generates a block in other branches and is referenced by a branch, then the miner is also considered to have generated a block in the branch, and his x stake's statistics are immediately zeroed out. In other words, as long as a miner generates blocks, he will be temporarily unable to participate in the competition for generating blocks in all branches, which is an incentive for other miners to ensure that all blocks can be referred to as much as possible. This will prevent miners from participating in competitions of multiple branches at the same time.

(2) The effective() function used in section 3.1.2 is actually intended to motivate the miners to vote, especially for the right branches. It works as follows: the current miner's x stake is distributed evenly to each of his votes within the last voting cycle2 of the current chain. Next, count the stakes that were voted to the current chain, which is effective (x). Simply put, the effective() function makes the miners' ability to generate blocks directly proportional to their correct votes. As a result, recording the wrong votes of other miners will reduce their ability to generate blocks, which is an incentive for miners to ensure that all wrong votes, as many as possible, are packed into the block. And for the correct votes, as mentioned in section 3.2.1, they can increase the competitiveness of the current branch and ensure to be recorded as many times as possible.

(3) If miners aim all of their votes for blocks that are relatively mature to ensure the correctness of the votes, this will create an impact on the identification of our current main branch. In this case, we stipulate that any vote that points to the current chain must point to the top block, otherwise it cannot be a valid record.

3.3 Summary

According to the abovementioned consensus process, stakeholders can hand over their work to miners via the voting process. Users with lower stakes only need to vote once in a voting cycle to ensure their stakes do not miss the voting activities. This can essentially avoid the unequal wealth distribution issue. Users with higher stakes do not need to remain online, which also ensures higher account security. Since the cost of stakeholders participating in network maintenance has been lowered, we can reduce the mining reward and avoid serious inflation.

Regarding the issue of miners' multiple voting, since the votes are already saved in the blocks before counting, the choices that anyone has ever made will be shown publicly; thus, there will be no hidden competition branches and miners cannot make multiple votes. For the historical attacks that construct new branches, we provide two layers of protection. One is the save point. Any historical attacks that occur before the save point will be rejected; the other one is the branch priority decision strategy. The branch of PoAS determines the priority based on the number of votes obtained. The branches with higher online stakes will definitely be heavier. Therefore, unless the historical stakes that an attacker possesses exceed all the online stakes of the current main branch, it will not succeed. Thus, the NaS problems known at present are solved.

Meanwhile, because the miner–gatherer form can be embedded in apps and websites, a large number of developers could be potential miners, which would reduce the threshold for mining, thereby improving the sustainability of chains.

4. Compete for Stakeholders

Miners will attempt to accumulate maximum stakes to achieve fast block generation. Rather than using the ability of network dispersity, some of them may cooperate with the stakeholders by other ways. The protocol provides three types of voting style for stakeholders in transaction structure. Type A: accept vote canvassing from gatherers in the broadcast-request form; type B: designate one miner and make him win the vote; type C: designate one miner that belongs to the stakeholder and work on mining individually. Three types of miners essentially compete using the stakes accumulated with three different abilities. Dynamic adjustment mechanisms are introduced via the reward distribution process to balance the mining power of these abilities. The risk that attackers will control the network by dominating one type of ability will be reduced effectively by this dynamic balance.

4.1 Wallet application

The wallet suppliers could profit from the system in different ways because the wallets participate closely in the process of consensus. This encourages developers to make better wallet applications, which is a step forward for client terminal development towards decentralization; or, more importantly, it provides an incentive to build sidechain projects and that helps to resolve the scalability problems.

However, from their own interests, wallet applications would attempt to make stakeholders vote for themselves with type-B transactions, which would eliminate the chances for the miners to compete with network dispersity. Therefore, a mechanism is required to encourage wallet applications to select type A other than type B. In this case, the "wallet account" field is added to type-A transactions and some profits are distributed to the suppliers of wallet applications; however, types-B transactions do not have such rewards. Due to the principle of maximizing profits, type-B users have no reason to choose type-A tags to broadcast their own transactions, thus ensuring the fairness of type-A competition.

5. Reward distribution

Users may engage in different selections of mining types, which will affect the network structure if the distribution proportion of mining rewards is different. We can balance the distribution of users by dynamically adjusting the allocation proportion. The total rewards are divided into transaction fees and block rewards.

5.1 Transaction fees

Users need to pay a certain service fee for every transaction to compensate for the resources consumed when miners record or execute the transactions. The transactions' service fees in PoAS are payed to the miners voted by them and awarded when those miners generate blocks, unlike Bitcoin where the service fees for all transactions in the blocks generated are obtained directly by block generation miners. However, for the purpose of ensuring security and encouraging higher service fees, we use a small portion of the service fee, e.g., 10%, as a reward to the miners who generated the current block.

When the service fee is considered as an income for distribution, miners and wallets are distributed at a fixed proportion, for example, miners account for 29%, wallets account for 31%[5], and the remaining part is shared by all the stakeholders of type-A and type-B transactions participating in the block generation. Considering the case wherein the number of stakeholders may be large when mining through network dispersity, we divide the earning into several parts and raffle among the stakeholders several times. The proportion of stakes of the stakeholders is the same as the probability of winning the lottery.

5.2 Block reward

To encourage more users with high stakes to participate in the maintenance, the system will issue another small amount of currency as the mining reward in addition to service fees. The amount of the mining reward affects the enthusiasm of stakeholders to participate in the type-C mining because type-C mining can't obtain any transaction fees. Since type-C mining is an important power of balance representing the competitors mining directly with stakes, the value of block reward should be calculated dynamically based on the current participation rate. If the participation level of type-C mining is lower than expected, then the mining reward can be increased and vice versa. For example, in the case wherein the miners and wallet accounts share 30% of the block rewards and 60% of the transaction fees and the participation is balanced, the rewards of mining should be about 1.2 times of the service fees.[6]

Mining reward value is an important balance parameter. It adjusts the number of participants in type-C mining activities and weakens the dominance of users' aggregation ability to prevent the centralization trend.

Block reward are distributed to the miners and wallets at a fixed proportion, for example, miners account for 14%, wallets account for 16%[5], and the remaining part is shared by all the stakeholders participating in the block generation according to the proportion of stakes. To the stakeholders of type-C transactions, all income is allocated to the miners; to the stakeholders of type-A and type-B transactions, it is the same as that of service fees, which is to raffle among stakeholders according to the ratio of stakes.

6. Frauds and attacks

(1) Simulate gatherers (i.e., sybil attack). Creating a large number of gatherer nodes via simulation and adding those simulate nodes to the P2P network will increase the success probability of canvassing.

To deal with this situation, we can control the process of creating P2P links, e.g., each node only needs to build connections with a certain number of nodes with the fastest response speeds.

(2) Super gatherers: If a gatherer is located near the backbone network, it will receive more votes because of the smaller network latency. If a large number of miners have opened up their super gatherers, the miners who compete relying on the number of network terminals will lose competitiveness.

Although a variety of countermeasures can be added in the voting network and also the rate of return on investment of this kind of attacks is not good, but the possibility will always exist, which can also be seen as a threat to the fairness of network dispersity. Even so, this does not cause a great impact on the consensus mechanism itself as the network latency and bandwidth as an ability can also bring fair competition. If countermeasures are not taken, and in the end, if the miners want to win more votes, they will need to arrange more super gatherer nodes globally, which will not consume many social resources and will not violate the original intention of our design.

(3) 51% attack.

The system will be vulnerable if a user controls >50% of the stakes in the network, which is the same as the common PoS protocol. However, the PoS system cannot maintain a high participation rate because users must run full nodes and keep them online to join the competition. Therefore, a lower online stake proportion reduces the level of security. In the PoAS system, stakeholders can join network maintenance with only a single vote during a voting cycle. With a lower participation requirement, we

can maintain a higher online proportion of the stake, which improves the system's security.

7. Conclusion

Compared to the PoW,PoS and BFT protocols, our model has the following benefits:

(1) No hashpower competition and no high-energy-consumption
(2) No such defects as multiple voting and history attack caused by the NaS problem
(3) Motivates sufficient competitive strength to maintain the security of network without falling into the problem of wealth centralization and inflation
(4) It has deterministic finality, objectivity, and complete decentralization but does not require any special node or extra expenses
(5) Provides incentive to the wallet developers, which ensures the sustainability and extendibility of the system

Conflict of Interest

Patent NO. : CN201811633256.0

Notes

[1] The theoretical winning probability is directly proportional to the number of online gatherers.
[2] After a miner generates a block, the x stakes obtained will be emptied, which will prevent attacks like stake grinding; the y stakes always maintain the counting interval of one voting cycle.
[3] Since this protocol may have a precise and objective counting on the active stakes of the current branch, it is not necessary to rely on the recent speed of block generation to adjust the difficulty, and parameter d can be set directly according to the sum of online stakes.
[4] Comparing the priorities of two leaf nodes begins with their forked position, which is the last common ancestor, and count the weights of the two branches after being forked in their respective perspectives. The heavier branch has a higher priority.
[5] In order to encourage the wallet application not to participate in mining competition, it is necessary to ensure that the wallet's revenue is slightly higher than the mining revenue.
[6] Users will not be shared in revenues by miners and wallets when they are mining alone with type-C transactions but will lose the opportunity to get service fees. The higher the mining reward is, the more advantages there are for users to select the type-C mining, so the ratio of users selecting type-C mining can be adjusted by controlling the amount of block rewards. Assuming that the miner and wallet share 30% of the block rewards and 60% of the transaction fees, the service fee is f, and the mining reward is F, for a user to have the same earnings in type-A and type-C mining, it must meet $f \times 90\% \times (1 - 60\%) + F \times (1 - 30\%) = F$; therefore, $F / f = 1.2$.

References

1. Yj1190590 (/yj1190590). "PoND (Proof of Network Dispersity) BlockChain Project." Github (accessed 29 April 2018) https://github.com/yj1190590/PoND/blob/master/README.md
2. Paul Firth. "Proof that Proof of Stake is either extremely vulnerable or totally centralised." BitcoinTalk.org (accessed 1 March 2016) https://bitcointalk.org/index.php?topic=1382241.0
3. Vitalik Buterin. "Long-Range Attacks: The Serious Problem With Adaptive Proof of Work." blog.ethereum.org (accessed 15 May 2014) https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/
4. Yonatan Sompolinsky and Aviv Zohar. "Secure High-Rate Transaction Processing in Bitcoin" No Publisher (2013) https://eprint.iacr.org/2013/881.pdf
5. Husam Ibrahim."A Next-Generation Smart Contract and Decentralized Application Platform" Github (2018) https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation