# Consensus Algorithm for Bigger Blockchain Networks

a@sumus.team, k@sumus.team, rr@sumus.team

March 3, 2018

**Abstract**

A new consensus algorithm called stake-distributed Byzantine Fault Tolerant (sdBFT) is offered, allowing to increase the number of network nodes participating in the process of consensus-building by several times over the existing BFT family algorithms, and to substantially increase transaction rates.

## 1 Introduction

When creating Bitcoin and blockchain technology, Satoshi Nakamoto used what was called a Proof-of-Work (PoW) algorithm as a consensus algorithm [1]. With the increased popularity of blockchain technology, the peculiarities of the PoW algorithm, resulting in a low transaction rate and, therefore, a high cost, became obvious.

Many IT-area experts have created new consensus algorithms or modified already existing ones, eliminating weak points of the PoW algorithm. As a result, the PoS, DPoS, LPoS, PoE, PoIT, and pBFT consensus algorithms have appeared. In this article we have tried to make our contribution to the development of consensus algorithms.
It became necessary to create our own consensus algorithm while attempting to create a blockchain that met the following requirements:

1. It should not take more than 1 minute to create a new block.
2. Blockchain network type - a corporate one.
3. The total number of network nodes that may participate in consensus-building should vary from $10^3$ to $10^4$.
4. There should be a high transaction rate - no less than $10^3$ transactions per second.
5. Blockchain algorithm implementation should not require any substantial power consumption.
6. Blockchain algorithm implementation should not require any substantial computation power compared to PoW blockchains.

Previously suggested consensus algorithms were considered by the authors of this article to not meet the above mentioned requirements for the following reasons.

 

The PoW consensus is the first type of consensus implemented within Bitcoin currency blockchains. This type of consensus is characterized by a low block closing rate and a low transaction rate. According to the listed data [2], the rate of Bitcoin currency transactions is only 7 transactions per second. The process of blockchain block formation requires substantial computation power with the PoW consensus applied. At the same time, the more computation power a participant in the consensus has, the more chances he has to form a block, which leads to an unproductive computation race between consensus participants.

The Proof-of-Stake (PoS) consensus and its variations of DPoS, LPoS were suggested to solve the issues with the PoW consensus related to high computation power demand and low rates of transaction block closing [3]. Despite a higher rate of transaction block closing and a lower computation power demand, the PoS algorithm has its flaws. The issue of the PoS algorithm is in centralization of coins (system resources). A blockchain user who has a maximum amount of system resources is getting more coins for providing the service of block acknowledgment. Thus, the number of nodes that take part in a consensus will gradually decrease, leading to non-compliance with the blockchain requirements listed above.

The pBFT consensus is another alternative to the PoW consensus algorithm. A few implementations of the pBFT consensus were suggested, one of which is "Honey Badger" [4]. As it is shown in Fig. 1, the fewer nodes participating in the consensus, the better the operation of the pBFT consensus implementation.

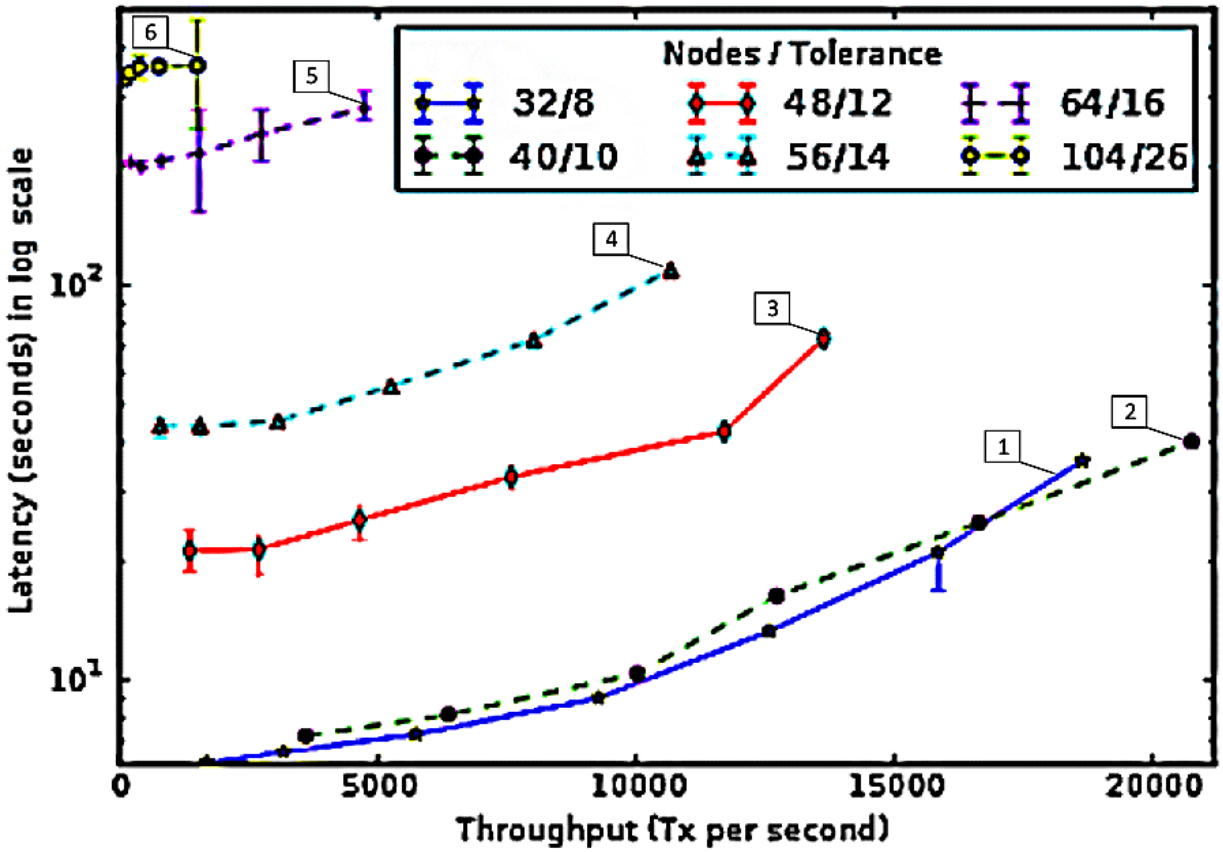


Figure 1. Diagram showing the dependency of block formation latency on the throughput of incoming transactions

 

Figure 1 illustrates the dependency diagram of block formation latency on the rate of incoming transactions where Nodes/ Tolerance is the relation between the total number of nodes reaching consensus and the total number of nodes not reaching consensus.

Curve 1 shows changes in block closing time depending on the throughput of incoming transactions for 32 nodes. Curve 2 shows the same type of changes for 40 nodes, while curves 3 through 6 show the same type of changes for 48, 56, 64, and 104 nodes, respectively.

The consensus is working with the highest level of efficiency when the number of nodes does not exceed 40.

The transaction throughput for such a number of nodes reaches $2 \cdot 10^4$ transactions per second, herewith block closing time does not exceed 40 seconds.

If the number of nodes exceeds 60, for curves 5 and 6, the transaction throughput for that number of nodes does not exceed $0.5 \cdot 10^3$ transactions per second. At the same time, block closing time will exceed 100 seconds. Consensus time approached 6 minutes for 104 nodes.

The authors of the article offer another kind of approach for the solution of productivity loss while using the pBFT algorithm, implemented within a stake-distributed Byzantine Fault Tolerant (sdBFT) consensus algorithm.

## 2 sdBFT Algorithm Essentials and Assumptions

We assume that the task of first-order consensus-building resolves itself to the reaching of general consensus by the participants of a distributed system while some participants haven't taken part in reaching the general consensus. This may occur due to the following reasons:

1. Error in the process of transferring a message about decision-making by one of the participants of the distributed system.
2. The process of transferring a message about decision-making by one of the participants of the distributed system is too slow.
3. Failure in the process of the participant's work in system.
4. Misrepresentation when making a decision in the system, whether intentional or unintentional.

If we consider reasons 1-3 for not taking part in the process of consensus-building, then it will be sufficient for a decision to be made that the following condition is fulfilled: $n > m + 1$ [5], where $m$ – is the number of participants who haven't taken part in consensus-building and $n$ – is the number of participants who have made a decision. In case when the participants not taking part in the process of consensus-building for the fourth reason appear in the system, then the task will resolve itself to the Byzantine Generals' Problem, which has a solution when:

$$n > 3 \cdot m \tag{1}$$

If we consider instead of $n$ generals $n$ blockchain nodes, taking part in the process of consensus-building, then it will be evident that this task is similar to the Byzantine Generals' Problem. In order to

 

solve the issue of the growth of time needed for consensus-building it is suggested from a finite set of nodes $B_n$, the cardinality of this set is $\overline{B}_n = n$, to extract a subset $\overline{B}_{n'}$ ($B_{n'} = n^{'}$) and assuming that node features are distributed along the entire blockchain, to solve the task not with $B_n$, but with $B_{n'}$ at $n \gg n^{'}$. The total number of nodes along the entire blockchain is set equal to $N$. Let us denote this set of nodes as $A_N$.

Let us specify the following function:

$$d : T \mapsto Y_{B_n},\ d = d(t),\ t \in T \tag{2}$$

where $t$ – is an independent variable corresponding to the current time. Let us assume that $d \in Y_{B_n}$ value of function (2) corresponds to the current state of $B_n$ in the moment of $t$.

Let us also specify the following function:

$$f : Y_{B_n} \mapsto \mathbb{N}_N,\ f = f(d),\ d \in Y_{B_n} \tag{3}$$

where $\mathbb{N}_N$ – is a finite subset of a set of natural numbers $\mathbb{N}$ of cardinality measure $N$.

Let us assume that the $B_{n'}$ subset is randomly selected from the $B_n$ set where $n'$ can be set. Then we'll get:

$$B_{n'} \subset B_n \subseteq A_N \tag{4}$$

Let us assume that $Y_{B_{n'}}$ – is a set of values of $d$, corresponding to all the current states of $B_{n'}$, $Y_{B_{n'}} \subseteq Y_{B_n}$ .

Let function $f$ denote $Y_{B_{n'}}$ for $\mathbb{N}_{n'}$ set, $\overline{\mathbb{N}}_{n'} = n'$. Let us assume that the numbers of nodes obtained by such display are $j_k,\ k = 1, \ldots, n'$. Let us assume that $j_{\hat{k}}$ is a number of assigned master nodes, $1 \leq \hat{k} \leq n'$. If $b$ is a formed block in relation to which a set of nodes $B_{n'}$ is attempting to reach the consensus at some point of time $t'$, then we will denote the SHA-3 hashing function [6] of this block as $H(b)$, and its values as $h$. Then the digital signature computation result based on, for instance, EdDSA algorithm [7] with edwards25519 elliptic curve [7] parameters, will be equal to $s = sig(h)$ .

# 3 Algorithm description

1. Let a node with $k$ ($1 \leq k \leq N$) number make a record $I$ into $B_n$ at $\hat{t} \in [t, t^{'})$ point of time.

2. Let us choose all $j_k$, including $j_{\hat{k}}$ by means of $f$ function. Consensus-building is executed with $[t, t^{'})$ half-interval.

3. If the master node admits the inclusion of record $I$ into block $b$ acceptable, it transfers this record to all nodes from $B_{n'}$ to check and include it into block $b$. Otherwise record $I$ will be rejected without notice.

4. A new record is included into a block prior to $t^{'}$ point of time. The master node is sending a message about block $b$ fixation to the same nodes. All the nodes from $B_{n'}$ are calculating the value hash-function $H(b)$ that is equal to $h$.

 

5. Each node is calculating a digital signature:

$$s_l = sig(h),\ \begin{cases} l = 1, \ldots, n' \\ l \neq \hat{k} \end{cases} \tag{5}$$

and sending it to node $\hat{k}$.

6. The master node is waiting for digital signatures in $\Delta t$ period of time following $t'$ moment. At $\Delta t + t'$ point of time the following tuple is formed at the master node

$$s_b = (s_1, \ldots, s_j),\ 1 \leq j < n' \tag{6}$$

The master node is checking each digital signature from (6) and calculating the number of correct signatures. Signatures of some nodes from $B_{n'}$ may turn out to be voting negatively or incorrect when in $B_{n'}$ there is a node with some number $j_{\tilde{k}},\ 1 \leq \tilde{k} \leq n'$, which
a) acknowledges record $I$ as incorrect;
b) has a blockchain state $\tilde{d}$ which is different from state $d$ for node $j_{\hat{k}}$ at $\tilde{t} \in [t, t')$ point of time;
c) will corrupt record $I$ in the process of blockchain block formation.

7. The master node is calculating the number of correct signatures $\mu$ and is checking the execution of the following inequality:

$$\mu > \left[\frac{2}{3}n'\right] \tag{7}$$

If (7) is not being executed, then the master node will make a conclusion that consensus hasn't been reached, otherwise the following number is made up for block $b$:

$$b \parallel s_{k_1} \parallel \ldots \parallel s_{k_\mu},\ 1 \leq l < n',\ l = 1, \ldots, \mu \tag{8}$$

for which $H(b \parallel s_1 \parallel \ldots \parallel s_\mu)$ and $sig$, which is a digital signature of a node with $j_{\hat{k}}$, number.

8. We will call the following number

$$d' = b \parallel s_1 \parallel \ldots \parallel s_\mu \parallel sig(H(b \parallel s_1 \parallel \ldots \parallel s_\mu)) \tag{9}$$

as a new closed block which we will consider to be a new blockchain state $d'$, corresponding to $t'$ point of time. The master node is sending (9) to all the nodes from $A_N$ set.

9. Let's assume that $s_b$ and $sig$ (9) are being checked at every node from $A_N$ set that has $1 \leq m \leq N$ number. If they haven't gone through the check, then the block $b$ is added to the blockchain of a node number $m$ and blockchain at that node passes into $d' = d(t')$ state. If that note hasn't received the above-mentioned digital signatures for the check at $[t' + \Delta t, t' + \Delta t + \lambda]$ period where $\lambda$ – is information transfer delay time, then the node that has number $m$ will consider the consensus hasn't been reached and choose a new $B_{n''}$ set based on the old state $d$ applying (3).

 

# 4 Specifying $f$ function

Let us calculate the double hash from (9), denoting the obtained number as $\nu$. Let us build a pseudorandom bit sequence of the following form:

$$\nu_1 = H(H(d)),\ \nu_2 = H(H(d+1)),\ \ldots \tag{10}$$

We'll obtain the following bit recording:

$$R = \nu_1 \parallel \nu_2 \parallel \ldots \tag{11}$$

We will sequentially split bit recording (11) into tuples without any skipping or overlapping with $r$ number of bits in each of them, which are the numbers of nodes forming the $B_{n'}$ set. These numbers are denoted by $j_k$, as indicated above, where $k = 1, \ldots, n'$ , $1 \leq j_k \leq N$ and $\hat{k} \triangleq 1$ because there will always be a node whose number will be formed first by the master node. If the already obtained $j_k$ number repeats, then the repeatedly obtained number will be skipped.

# 5 Conclusions

In the authors' opinion the offered sdBFT algorithm should have a higher processing speed compared to BFT algorithms. It will be possible to control the speed of creation of new blocks or, in other words, the speed of algorithm operation, by changing the power of set $B_{n'}$. A potentially large number of consensus participants makes a conspiracy difficult when a group of voting nodes is forming a new block, managing the block's content at their own discretion, because another $B_{n'}$ set of voting nodes will be chosen before the start of another consensus-building process. The authors of this article believe pseudorandom choice of $B_{n'}$ set of voting nodes won't allow a great impact on the process of choosing nodes during the next voting round. Going forward, the authors expect to obtain experimental evidence of the theoretical provisions set forth in this article, evaluate the speed of sdBFT algorithm performance, and study the possible blocking of blockchain networks.

# References


[1] Satoshi Nakamoto (2009). "Bitcoin: A Peer-to-Peer Electronic Cash System". www.bitcoin.org .

[2] (2018.01.10). "Transactions Speeds: How Do Cryptocurrencies Stack Up To Visa or PayPal?". https://howmuch.net/articles/crypto-transaction-speeds-compared .

[3] BitFury Group (2015.09.13). "Proof of Stake versus Proof of Work". http://bitfury.com/content/5-white-papers-research/pos-vs-pow-1.0.2.pdf .

[4] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, Dawn Song (2016). "The Honey Badger of BFT Protocols". https://eprint.iacr.org/2016/199.pdf .

[5] Leslie Lamport, Robert Shostak, Marshall Pease (1982). "The Byzantine Generals Problem". ACM Transactions on Programming Languages and Systems. T.4, 3: 382–401 .


 

[6] National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf .

[7] S.Josefsson, I.Liusvaara. “Edwards-Curve Digital Signature Algorithm (EdDSA)”. IETF RFC. https://tools.ietf.org/html/rfc8032 .

SUMUS
AB SUM AD SUMUS

DN: cn=k, o=sumus team, ou, email=k@sumus.team, c=<o

SUMUS
AB SUM AD SUMUS

DN: cn=RR, o=sumus team, ou, email=rr@sumus.team, c=<o

SUMUS
AB SUM AD SUMUS

DN: cn=A, o=sumus team, ou, email=a@sumus.team, c=<o

 