2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS) | 978-1-6654-7177-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDCS54860.2022.00031

Themis: An Equal, Unpredictable, and Scalable Consensus for Consortium Blockchain

Linpeng Jia[1,2], Keyuan Wang[1,2], Xin Wang[1,2], Lei Yu[1,2], Zhongcheng Li[1,2], and Yi Sun[*1,2]
[1]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*
[2]*School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China*
jialinpeng@ict.ac.cn, wangkeyuan@ict.ac.cn, wangxin18z@ict.ac.cn
yulei2008@ict.ac.cn, zcli@ict.ac.cn, sunyi@ict.ac.cn
*Corresponding author: Yi Sun

***Abstract*—Consensus algorithm is the core component of consortium blockchains. *Equality*, *Unpredictability* and *Scalability* are three important demands for the consensus algorithms of consortium blockchain. Existing deterministic consensus algorithms (e.g. PBFT) can ensure *Equality*, but cannot meanwhile meet *Unpredictability* and *Scalability*; probabilistic consensus algorithms (e.g. PoW) can achieve *Scalability* and guarantee a decent *Unpredictability*, but cannot meet the *Equality* requirement. In this paper, we propose a new consensus algorithm, namely *Themis*, which takes the three properties into account. *Themis* independently adjusts the block-producing difficulty of each node through a self-adaptive node election mechanism, effectively reducing the correlation between the block-producing frequency and the invested computing power of each node. Besides, a GEOST main chain consensus rule is proposed to handle forks and further improve the performance of the algorithm. If a fork occurs, consensus nodes will choose the sub-chain with the highest *Equality* to join the main chain. Evaluations show that *Themis* achieves outstanding performance in *Equality* and *Unpredictability* while ensuring *Scalability*, compared with the existing algorithms.**

***Index Terms*—Consensus algorithm, Blockchain, Equality, Unpredictability, Scalability**

I. Introduction

Consortium blockchains have become an important part of blockchain ecosystem. At present, thousands of blockchain applications based on some consortium blockchain frameworks such as HyperLedger Fabric [1] and FISCO BCOS [2], have emerged in various countries, covering finance [3], healthcare [5], public welfare [6], transportation [7] and other industries. Consensus algorithm is the core component of consortium blockchain. The pros and cons of the consensus algorithm will directly affect the application range and comprehensive performance of consortium blockchain.

Existing consortium blockchain systems mainly adopt BFT-based deterministic consensus algorithms [8]–[15], represented by the Practical Byzantine Fault Tolerance (PBFT) [8]. Consensus nodes in these consensus algorithms often determine the leader node based on a round-robin protocol or a uniform random function so that each consensus node has equal opportunity to produce a new block (referred as ***Equality***). **Equality is a basic requirement for the consensus algorithm of consortium blockchain**. Different consensus nodes in a consortium blockchain may have a certain preference for the order of transaction execution, because they are representatives of different user sets.

However, with the application scale and fields expanding, two new trends have emerged in the development of the current consortium blockchain, and new requirements have also been put forward for the design of its consensus algorithm:

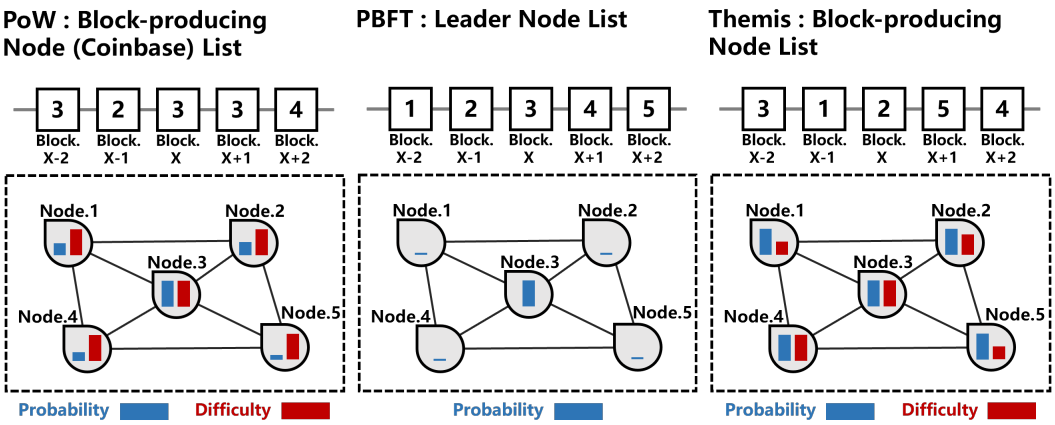
- **Trend-1. System from closed to open: consortium blockchains are requiring block producers (block-producing node) being unpredictable during the consensus process.**

Previously, consortium blockchain systems were mainly closed systems, which only provided blockchain services for users within the consortium. But now, they are required to be more and more open. The consensus nodes of consortium blockchain also need to provide services such as data query for users outside the consortium. However, the openness brings more security attack problems, especially when the leader node of each round is exposed in advance. If the leader node is determined in advance, it would be easier for malicious nodes to launch single point attacks or disrupt the system's normal operation in other ways. Therefore, a new requirement is put forward for consortium blockchain consensus algorithms: the election of block producers should be random and uncertain, before any valid block is generated in each consensus round (referred as ***Unpredictability***).

- **Trend-2. Scale from small to large: consortium blockchains are requiring consensus algorithms to achieve high scalability.**

The consensus node sets of the existing consortium blockchains are mostly small in scale (several to dozens of nodes). As the application fields of consortium blockchains keep expanding and the integration of related industries keeps accelerating, a large-scale consortium blockchain platform (supporting hundreds or thousands of nodes) is in increasingly high demand. Therefore, in order to meet the deployment requirements of various application scenarios in different scales, consortium blockchain consensus algorithms need to make sure that the transaction throughput will not significantly decrease when the number of consensus nodes grows larger (referred as high ***Scalability***).

2575-8411/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDCS54860.2022.00031



(a) PoW [16] (b) PBFT [8] (c) *Themis*

Fig. 1. Illustration for different consensus algorithms. Taking the consensus round of Block.X as an example, (a) PoW [16]: each node has the same difficulty, but its probability and frequency are positively correlated to its invested computing power; (b) PBFT [8]: only the leader node has the right to produce block, and each node has the same frequency; (c) *Themis*: each node's probability and frequency are basically the same, and its difficulty matches its computing power.

However, existing algorithms cannot simultaneously achieve the above three key demands. For most existing BFT-based consensus algorithms, the leader node can be known soon after the current consensus round starts. This cannot meet the *Unpredictability* requirements. In addition, BFT-based consensus algorithms need to be completed through several rounds of communication among nodes, which induces significant communication overhead and severely limits the achievable scale of consensus nodes. Most of the existing probabilistic consensus algorithms [16]–[22], such as PoW [16], support virtually any number of nodes to participate in the consensus, and the block producer cannot be known before the valid block is generated in the current round, which can basically meet the *Scalability* and *Unpredictability* requirements. However, taking PoW [16] as an example[1], the block-producing probability of a certain node is positively correlated to its invested computing power, and the computing power cannot be limited. This cannot meet the *Equality* requirement. Moreover, the concentration of computing resources will also lead to the block-producing probability being predictable[2].

We propose a novel consortium blockchain consensus algorithm, namely *Themis*, to adapt to the new development trends of consortium blockchains. *Themis* is based on the Proof-of-Work (PoW) mechanism, and requires no extra communication among nodes during the consensus process, which ensures high *Scalability*. We design a self-adaptive node election mechanism to reduce the correlation between the block-producing frequency and the invested computing power. This mechanism works by dynamically adjusting the block-producing difficulty of each node based on their historical block-producing frequency. It makes the block-producing probability of each node approximately equal, thereby effectively improving the *Unpredictability*, solving the problems of the existing deterministic (completely predictable) and probabilistic (probability is predictable) consensus algorithms. In addition, the self-adaptive node election mechanism and the GEOST main chain consensus rule ensure the *Equality* of *Themis* consensus algorithm.

Based on the analyses and evaluations, we demonstrate that *Themis* achieves the convergence of history and resilience to 51% computing power attack. Experiments show that, compared to PoW [16], *Themis* reduces the variance of block-producing frequency by 89.20% (reflecting *Equality*) and variance of block-producing probability by 97.18% (reflecting *Unpredictability*), and guarantees the same *Scalability*. Compared to PBFT [8], *Themis* achieves the goals of *Unpredictability* and *Scalability*, and guarantees the same level of *Equality* after the algorithm converges in a few consensus rounds. Fig.1 illustrates the comparison of PoW [16], PBFT [8] and *Themis* in terms of block-producing probability, block-producing frequency and block-producing difficulty.

The rest of the paper is structured as follows: We present the design goals of consortium blockchains' consensus algorithm in Section II. We provide the overview of *Themis* in Section III and introduce the node election mechanism and main chain consensus rule in Section IV and V. The security and overhead analyses are given in Section VI, and evaluations of our algorithm are provided in Section VII. Section VIII introduces related work. And Section IX summarizes the contributions of this paper.

II. Design Goals

Under the development trend of consortium blockchain, we first generalize three basic goals of designing a consensus

[1]Other probabilistic consensus algorithms, such as PoS [19], have similar mechanisms.

[2]In Bitcoin system [16], from Jan 06 to Jan 12 in 2022 [26], the sum of the computing power of the top 4 mining pools has exceeded 59.17% of the computing power of the entire network, while a large number of unknown nodes that independently participate in the consensus only generate about 1.68% of the valid blocks.

algorithm of consortium blockchain:

- **Equality:** *Equality* refers to the frequency that the blocks generated by each consensus node are finalized in the main chain (namely, block-producing frequency) is the same. *Equality* is a statistic that can be measured by calculating the variance of block-producing frequency (Equation.1) over a period of time. The smaller its variance, the higher *Equality* the algorithm achieves, which indicates each consensus node enjoys a more equal opportunity to produce blocks.

$$\sigma_f^2 = Var(\{f_i | i \in [0, n)\}) \tag{1}$$

In Equation.1, $f_i = q_i/\Delta$ represents the block-producing frequency of consensus node N_i, where Δ represents the number of blocks in a counting epoch, q_i represents the number of blocks produced by the consensus node N_i in one epoch.

- **Unpredictability:** *Unpredictability* means the election process of the block producer should be random and unpredictable before a valid block is generated in each consensus round. *Unpredictability* can be measured by the variance (Equation.2) of the probability of a node producing a valid block during each consensus round (namely, block-producing probability). The smaller its variance, the higher *Unpredictability* the algorithm achieves[3].

$$\sigma_p^2 = Var(\{p_i | i \in [0, n)\}) \tag{2}$$

In Equation.2, p_i represents the block-producing probability of consensus node N_i.

- **Scalability:** The algorithm can support consensus nodes of different scales to participate in the consensus process. And there should be no significant decrease in transaction throughput (i.e. transactions per second, *TPS*) and no drastical increase in delay due to the expansion of the consensus node set. Under the requirement for a certain *TPS*, the more nodes participating in the consensus the algorithm supports, the higher *Scalability* it achieves.

III. Overview

Each consensus round of *Themis* consists of two phases: node election and main chain consensus. In the node election phase, all nodes set their own difficulty complying with the difficulty adjustment mechanism (Section IV) to independently solve the *Proof-of-Work puzzle*. A node broadcasts the valid block it generates to other nodes, and also monitors the blocks generated by other nodes. In the main chain consensus phase, the sub-chain with the highest *Equality* is chosen to join the main chain according to the GEOST main chain consensus rule (Section V). Each time a main chain consensus round finishes, the local state of each consensus node will migrate to a new state. The specific process of each phase is as follows:

- **Node Election**: In each consensus round, a node firstly decides its block-producing difficulty of the current round according to the difficulty adjustment mechanism. Then the node selects some transactions from the transaction pool upon its preferences, and stores them into block body in order, and initializes the block header. After the block is constructed, the node tries to solve the puzzle by modifying fields such as $nonce$ in the block header based on its current difficulty. If in a calculation, the hash value of the block header is less than the puzzle's target value corresponding to its difficulty, the node will successfully generate a valid block for the current round. Next, the node signs the block header with its private key and broadcasts the block together with its signature to other consensus nodes.
  During each node election, there may exist several valid blocks generated across the entire network. After a block generated by other nodes is received, the node firstly verifies whether the block header signature belongs to the node in the consensus node set of consortium blockchain; then checks whether the difficulty and the hash value of the block header are correct according to the latest difficulty table in its local storage; and finally checks the validity of the transactions in the block. Valid blocks will be added to the local block tree and invalid ones will be discarded.
- **Main Chain Consensus**: Since solving the Proof-of-Work puzzle is a random process, there may exist several valid blocks during the broadcast. These valid blocks coexist and are at the same height in the local block tree of the node, that is, a fork occurs. In order to solve the fork, the GEOST rule specifies that the root block of the sub-tree with the highest *Equality* is added to the main chain when the block tree contains several blocks at the same height; if sub-trees have the same *Equality*, the root block of the sub-tree first received by the node will be selected to join the main chain. These above processes loop until the selected block no longer has child blocks (i.e. the latest block of the main chain). Till now, a consensus round has been completed, and the nodes will start a new round of node election based on the latest block.

Security Assumptions. *Themis*' security assumptions are the same as related work [14], [21]–[24]: 1) An honest node is always behaving honestly[4]; 2) the data sent by an honest node will eventually reach all honest nodes within a known maximum delay δ [27].

IV. Node Election

In Section III, we notice that in the node election phase, the following important questions need to be solved to let *Themis*

[3]*Equality* requires the consensus nodes have approximately equal status, which means that the number of blocks produced by each node in a period of time is approximately equal. *Unpredictability* requires that it is difficult to predict which node will produce the next block during a consensus round. *Equality* and *Unpredictability* are different goals. Taking the PBFT [8] algorithm as an example, the round-robin block-producing rule achieves a very good *Equality*, but there is no *Unpredictability* at all.

[4]Honest nodes always obey the protocol.

work properly:

1. How to adaptively adjust the block-producing difficulty of each node to ensure good *Equality* and *Unpredictability*? And how to synchronize the current difficulty values of each node for verifying the validity of blocks? (IV-A)
2. How to set the block-producing difficulty to prevent each node from not willing to invest computing power and threatening the system security? [5] (IV-B)
3. How to maintain and update the consensus node set to ensure that new honest nodes can easily join the consensus and malicious nodes can be excluded from the consensus in time? (IV-C)

This section will provide specific solutions to the above questions.

A. Self-adaptive Block-producing Difficulty Adjustment Mechanism

The core idea of the self-adaptive block-producing difficulty adjustment mechanism in *Themis* consensus is to periodically adjust each node's difficulty based on the historical block-producing frequency, thereby to make the block-producing probability of each node as equal as possible. The smaller the variance of block-producing frequency is, the better the algorithm performs in terms of *Equality* and *Unpredictability*.

The *Themis* algorithm generates current round's candidate blocks based on the Proof-of-Work mechanism. The block-producing difficulty of each consensus node is set independently. For example, the block-producing probability of a consensus node N_i in a consensus round is its effective computing power ratio. As shown in Equation.3, $h_i^{'}$, h_i and m_i respectively represent the effective computing power, actual computing power and the multiple of basic block-producing difficulty (IV-B) of the consensus node N_i; n represents the quantity of consensus nodes.

$$p_i = \frac{h_i^{'}}{\sum_{i=0}^{n-1} h_i^{'}} = \frac{h_i/m_i}{\sum_{i=0}^{n-1} h_i/m_i} \tag{3}$$

However, the actual invested computing power of consensus nodes cannot be obtained directly. In order to reduce the variance of block-producing probability of nodes, we need to first estimate the distribution of nodes' effective computing power. Based on Equation.3, it can be known that the number of blocks produced by consensus nodes in a difficulty adjustment epoch obeys the binomial distribution. According to the number of blocks q_i^e produced by each node in an epoch[6], an approximate effective computing power distribution can be reflected with maximum likelihood estimate (MLE) method. As shown in Equation.4 and Equation.5, Δ represents the number of blocks in each difficulty adjustment epoch; e represents the epoch index.

$$P(p_i|q_i^e) = \binom{\Delta}{q_i^e} p_i^{q_i^e} (1-p_i)^{\Delta - q_i^e} \tag{4}$$

Solve Equation.4 and get the MLE for p_i:

$$\frac{\partial P(p_i|q_i^e)}{\partial p_i} = 0 \Rightarrow p_i = \frac{q_i^e}{\Delta} \tag{5}$$

Since the MLE of the binomial distribution is unbiased, so the effective computing power ratio estimated above is unbiased, and satisfies $E(q_i^e/\Delta) = p_i$. Therefore, we can use the ratio of the block-producing frequency $f_i^e = q_i^e/\Delta$ in the previous epoch to the expected block-producing frequency[7] $F_0 = 1/n$, and the previous multiple of difficulty m_i^e, to calculate the multiple of difficulty in the next epoch (Equation.6).

$$m_i^{e+1} = \max\left(\frac{f_i^e}{F_0} \times m_i^e, 1\right) = \max\left(\frac{n \times q_i^e}{\Delta} \times m_i^e, 1\right) \tag{6}$$

And $m_i^0 = 1$. If some nodes do not participate in the consensus for a long period of time, the block-producing difficulty may continuously decrease, which threatens the system security. To avoid this, the difficulty for each consensus node should be at least set to basic block-producing difficulty in the corresponding epoch.

As *Themis* algorithm runs, every Δ blocks, based on the block-producing frequency of each node in the current epoch, each consensus node will update the multiple of difficulty of all nodes in the next epoch. During the calculation, each node can calculate the current block-producing difficulty of all nodes according to the same blockchain information and the same rules. Therefore, each node can verify the validity of blocks without extra communication among nodes.

Epoch length Δ setting will affect the estimation of the block-producing probability. The impact of the epoch length Δ will be specified in Section VII-D in detail.

B. Block-producing Difficulty Settings

Basic Difficulty Settings. *Themis* reduces the correlation between the block-producing frequency and the invested computing power of consensus nodes through a self-adaptive difficulty adjustment mechanism, so that nodes only need to invest a fixed and small amount of computing power to participate in the consensus in an equal manner. However, in *Themis*, the total effective computing power is directly related to the computational costs of successful attacks from malicious nodes. Therefore, in order to prevent computing power attacks to some extent, the system running *Themis* needs to set a basic block-producing difficulty D_{base} ($D_{base} \geq 1$), whose expectation meets Equation.7.

$$\frac{T_0}{E(D_{base})} \times \frac{1}{T_{max}} = \frac{1}{I_0 \times nH_0} \tag{7}$$

Let the expected block interval be I_0 (seconds). H_0 represents the minimum times of calculating puzzle per second

[5]The decrease of the overall computing power will reduce the costs of malicious nodes' attacks.

[6]Consensus nodes adopt GEOST (Section V) to deal with forks and determine the sole main chain. q_i^e refers to the number of blocks produced by each node in its local main chain during the epoch e.

[7]In ideal case, the variance of block-producing frequency of each node is 0, that is, the expected block-producing frequency of each node is $1/n$.

for each consensus node; T_0 is the target value of the puzzle when $D_{base} = 1$; T_{max} refers to the maximum hash value of the SHA-256 function.

In Equation.7, the left expression represents the probability that a consensus node successfully generates a valid block every time it tries to solve the puzzle; the denominator in the right expression represents the expected number of times that the entire network tries to solve the puzzle in order to generate a valid block. While the algorithm is running, the effective computing power of the whole network will tend to be stabilized and fluctuate within a certain range through the dynamic difficulty adjustment. In order to guarantee a relatively stable block interval in each epoch, *Themis* adjusts the basic block-producing difficulty based on the ratio of the block interval to its expected value in the previous epoch.

Node Difficulty Settings. Take a consensus node N_i for example, in epoch e, its block-producing difficulty in each consensus round is $D_i^e = m_i^e \times D_{base}^e$; The target value for solving the puzzle is $t_i^e = T_0/D_i^e$. Once the hash value of the block header the node calculates is less than t_i^e, the node can successfully produce a valid block in this round.

C. Consensus Node Set Update

In *Themis*, each consensus node has the right to raise proposals to add or remove a consensus node. A new node can contact any node in the current consensus node set to construct a node joining proposal (Add), which contains its address, proof of identity and other information. During the *Themis* consensus, each node needs to monitor other nodes' behavior. Once malicious behavior is detected, node should construct a node removing proposal ($Remove$), which contains the proof of node removing, such as: packing invalid transactions, launching double-spending attacks etc.

After consensus node has locally constructed the proposal, it sends a transaction to call the consensus node set management contract $NodeSetContract$, waiting for other nodes to vote for a node joining or removing proposal (one node one vote). If the supporting nodes exceed half of the consensus node set, the proposal will take effect at the beginning of the next consensus round.

The update of the consensus node set will cause changes in the effective computing power distribution. In order to let the effective computing power of the entire network be matched with the basic difficulty setting, the basic block-producing difficulty also needs to be adjusted. Regardless of whether a node joins or leaves, after a period of adjusting effective computing power, the effective computing power of each node is about H_0. Therefore, the basic difficulty can be adjusted proportionally according to the ratio of consensus nodes' quantity in the next epoch n^{e+1} to consensus nodes' quantity in the current epoch n^e. Meanwhile, the calculation of the block-producing frequency also needs to be adjusted accordingly.

V. Main Chain Consensus

After introducing the self-adaptive block-producing difficulty adjustment mechanism, the variance of block-producing probability is reduced, therefore more forks (multiple blocks produced at the same block height) may occur. Then, consensus nodes need to choose a block to join the main chain according to the main chain consensus rule. A new round of consensus should be based on that block. In *Themis*, we propose a new rule GEOST to handle forks.

Algorithm 1 The Greedy the most Equal-Observed Sub-Tree (GEOST)

Input: Block tree $Tree$, Height of the block tree a
Output: Main chain B

```
B[0] = GenesisBlock.
for i = 1; i < a; i++ do
    K = Children_Tree(B[i − 1])
    B[i] = K[0]
    if K.length > 1 then
        for j = 1; j < K.length; j++ do
            B[i] = Equal(B[i]) > Equal(K[j]) ? B[i] : K[j]
        end for
    end if
end for
return B.
```

A. Fork Processing Requirement

During broadcasting the first valid block of each round in *Proof-of-X* [16]–[22] consensus, other consensus nodes are still solving the puzzle before receiving that block. So it's possible to generate more than one valid block at the same height, namely fork. Under the longest chain rule [16] or the heaviest chain rule (GHOST) [28] adopted by most consensus algorithms, when the length or weight of the sub-trees is the same, each consensus node adopts the first block it receives to continue the next consensus round. If the block-producing rate of each sub-tree is the same or very close, the fork may last for several rounds until a certain sub-tree wins.

Therefore, in order to avoid any changes in the state of upper-level applications caused by forks, the block confirmation time of *Proof-of-X* [16]–[22] algorithms is relatively long (e.g. 1h in Bitcoin [16]). However, in consortium blockchains, long block confirmation time will severely affect the timeliness of applications. We found that introducing *Equality* into the main chain consensus can bring about two benefits: 1) Determine the main chain faster when multiple valid sub-trees coexist. 2) Ensure that the main chain is accepted by the majority of nodes (or effective computing power) in terms of the highest *Equality*, which improves the *Equality* of our algorithm.

B. The Greedy the most Equal-Observed Sub-Tree (GEOST)

Before giving the specific process of the GEOST rule, we first define ***the most equal chain*: the chain with the highest *Equality*[8] selected by the majority of nodes (or effective computing power) in the entire network** (referred as, the

[8]Defined in Section II.

main chain in *Themis*). Compared with other blocks of the same height in the block tree, the blocks in the main chain are located in the sub-tree with the largest number of blocks (firstly received by most nodes), and the lowest variance of block-producing frequency σ_f^2.

The pseudo code of GEOST is shown in Algorithm.1, where $GenesisBlock$ is the genesis block when the algorithm is launched, a constant shared by all consensus nodes; the function $Children_{Tree}(B[i-1])$ returns the child blocks array of the block B_{i-1} in the block tree $Tree$; the function $Equal(B[i])$ returns the *Equality* priority of the sub-tree, whose root block is B_i. Specifically, trace back to the $GenesisBlock$. When there are several blocks at the same height in the block tree $Tree$, GEOST will first adopt the root block of the sub-tree which contains the largest number of blocks to the main chain. If distinct sub-trees contain the same number of blocks, then GEOST will choose the block with the lowest variance of block-producing frequency σ_f^2 to join the main chain. When it comes to the case that the variance σ_f^2 of distinct sub-trees is still the same, the node will choose the leaf block of the first received sub-tree as the previous block to start the next consensus round.

Fig.2 illustrates the comparison of the three main chain consensus rules under selfish mining [29] attacks[9]. In the second consensus round, the honest consensus node set produced three blocks, namely $2A$, $2B$, and $2C$. The weight of the sub-tree of these three blocks is the same, and the variance of block-producing frequency is also the same. Therefore, at the beginning of the third consensus round, the three blocks $2A$, $2B$, and $2C$ should exist simultaneously. After the fourth consensus round, the number of blocks in the sub-tree of blocks $3B$ and $3C$ is the same, but the variance of block-producing frequency σ_f^2 of the sub-tree which follows the block $3C$ is lower, so block $4C$ is adopted as the previous block to solve the next puzzle. Compared with the longest chain rule [16], GEOST and GHOST [28] both can alleviate the selfish mining [29] problem and accelerate confirming the main chain. Compared with GHOST [28], GEOST considers the variance of block-producing frequency of the nodes during the main chain consensus. Therefore, several sub-trees coexisting situation can be finalized faster.

VI. Analysis

Before giving the experiment results, in this section, we analyze the convergence of *Themis* algorithm (VI-A) and the properties of resisting 51% effective computing power attacks (VI-B) from the mechanism aspect, as well as the storage and communication overheads (VI-C), fork rate and fork duration (VI-D), and discuss the possibility of replacing the Proof-of-Work mechanism with other *Proof-of-X* mechanisms (VI-E).

A. The Convergence of History

We first prove that GEOST can guarantee the consistency of the local data among consensus nodes, namely the convergence

[9]The variance of block-producing frequency σ_f^2 at different block is just for indicative purpose.

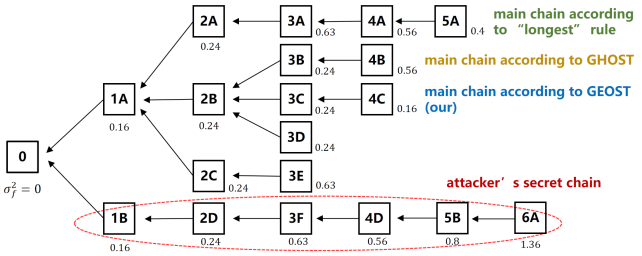


Fig. 2. A block tree in which the longest chain, the chain selected by GHOST, and the chain selected by GEOST differ. An attacker's chain is only able to switch the main chain under the longest chain rule.

of history. Let ψ_{B_j} be the moment when all consensus nodes reach a consensus on the block B_j, where j represents the block *id*.

Proposition 1 (The Convergence of History). $E(\psi_{B_j}) < \infty$, ***which means, the block*** B_j ***will either be adopted to the main chain or be treated as a fork and abandoned by all nodes over a certain period of time.***

Proof. Given the maximum network delay (transmission diameter) δ [27], assuming that the block B_j is neither adopted nor abandoned by all nodes at the moment $\tau > time(B_j)$, where $time(B_j)$ is the moment when the block B_j is produced.

Event ε_τ^0 means a new block is generated in the network during τ to $\tau+\delta$ and there is no new block-producing event from $\tau+\delta$ to $\tau+2\delta$. The sub-tree where this block is located has a lower variance of block-producing frequency, though it has the same number of blocks as the other sub-trees who share the same ancestor block. When the event ε_τ^0 occurs for the first time, all nodes will have received this block at $\tau+2\delta$, which leads the variance of block-producing frequency of a certain sub-tree lower than that of other sub-trees. Then the state of the block B_j is also determined right away. Since the block interval in *Themis* complies exponential distribution, $Pr(\varepsilon_\tau^0)$, the probability of event ε_τ^0, has nothing to do with τ. Hence the expected waiting time for the event ε_τ^0 is finite [30]. ψ_{B_j} has a finite upper bound, $E(\psi_{B_j}) < \infty$.

ε_τ^1 means that a block is generated in the network at a new height during $\tau+\delta$ to $\tau+2\delta$ and there is no new block-producing event from $\tau+2\delta$ to $\tau+3\delta$. All nodes have received all valid blocks within the time period from τ to $\tau+\delta$, but the status of block B_j has not been confirmed. This is because at that moment the sub-tree where block B_j is located has the same number of blocks and variance of block-producing frequency as the other sub-trees who share the same ancestor block. When the event ε_τ^1 occurs for the first time, all nodes will have received the new block at $\tau+3\delta$, which leads the number of blocks of a certain sub-tree larger than that of other sub-trees. Then, the state of the block B_j is also confirmed immediately. $Pr(\varepsilon_\tau^1)$, the probability of event ε_τ^1, has nothing to do with τ as well [30]. As above, ψ_{B_j} has a finite upper bound, $E(\psi_{B_j}) < \infty$.

B. Resilience to 51% Attacks

In *Themis*, the block-producing rate of consensus node set is positively related to the effective computing power of consensus nodes. Let λ_{honest} be the block-producing rate of the honest node set.

Proposition 2 (Resilience to 51% Attacks). Suppose the attackers' block-producing rate is $q\lambda_{honest}$***, where*** $q \in [0, 1)$***. Once the block*** B_j ***was adopted to the main chain by the honest node set at the moment*** $time(B_j) + \tau$***, as*** τ ***grows, the probability that the block*** B_j ***will be moved out of the main chain is gradually down to*** 0***.***

Proof. The block B_j is adopted to the main chain by the honest node set at $time(B_j) + \tau$, that is, $\psi_{B_j} < time(B_j) + \tau$. Meanwhile, the number of blocks in the sub-tree where the block B_j is located is growing at the rate of λ_{honest}, and it is quicker than that in the attackers' chain. Therefore, as τ grows, the difference between the number of blocks in the sub-tree of the block B_j and that of the attacker's chain will gradually increase. The probability of the block B_j being forked (moved out of the main chain) tends to down to 0 (The Law of Large Numbers).

C. Storage and Communication Overheads

Storage Overhead. Compared with the existing consortium blockchain consensus algorithms, each consensus node in *Themis* also needs to store the multiple m_i^e of difficulty in each epoch of each node and the quantity q_i^e of blocks added to main chain in each epoch, apart from the complete blockchain data and consensus nodes' identity authentication information. During the implementation of *Themis*, $float$ type (4 Bytes) is chosen to store m_i^e and int type (4 Bytes) for q_i^e. In each epoch, the data storage size of the entire network will increase by $8n$ Bytes (far smaller than average block size[10]). Therefore, we believe that the impact of the additional storage overhead on system performance could be ignored.

Communication Overhead. In *Themis*, compared with the traditional PoW [16] algorithm, after a valid block is produced, the consensus node needs to sign the block header with its private key, introducing a small size increase of a signature data (about 128 Bytes far smaller than average block size [32], [33]) to each block. Under the same conditions (data structure and broadcast algorithm, etc.), compared with the traditional PoW [16] algorithm, *Themis* does not increase much communication overhead and can still guarantee *Scalability* (VII-D).

D. Fork Rate and Fork Duration

Y. Shahsavari et al. [31] established a model analyzing fork in Bitcoin network and concluded that the fork rate of PoW [16] is $1 - e^{-\frac{\delta}{T_0}}$. Furthermore, their experimental results show that the fork rate of PoW gradually decreases, as the average out-degree of nodes increases. Compared with *Themis*, the distribution of computing power in PoW [16] is more concentrated [26] (the variance of block-producing probability is higher). Meanwhile, the existence of mining pools provides a higher out-degree, mining pools can cover more nodes after producing blocks. Therefore, PoW [16] achieves slightly lower fork rate and fork duration (VII-D).

[10]In the second half of 2021, average block size of Bitcoin network is 1.06 MB [32]; and 68.4 KBytes in Ethereum network [33].

E. Other Proof-of-X Mechanisms

When implementing the system, some other *Proof-of-X* mechanisms can replace the Proof-of-Work mechanism of *Themis* algorithm after some modifications. Taking Proof-of-Stake (PoS) [19] and Proof-of-Reputation (PoR) [25] as examples to give a hint: 1) In the PoS [19] mechanism, the *coinDay* of a node is public information, and the larger *coinDay*, the larger the target value of the puzzle to solve. To avoid the problem of inequality and predictability caused by the different *coinDay*, the way to calculate *coinDay* needs to be modified; 2) In the PoR [25] mechanism, the leader of each round is uniquely determined according to the node's reputation. So it's recommended to combine committee establishment and leader election mechanism similar to those in Algorand [11] to determine the block-producing node, enhancing the *Unpredictability* of PoR [25] mechanism.

VII. Evaluation

A. Experiment Settings

To evaluate the performance of *Themis*, we design and implement the *Themis* algorithm and related modules including consensus node initialization, node election, main chain consensus, ledger processing etc. and carried out a comparative experiment with classic algorithms. The main environment configurations of the evaluation part are as follows:

Basic Settings. Similar to most blockchain testbeds, in our settings, each transaction size is 512 Bytes, the bandwidth of all connections between nodes are set to 20 Mbps, data transmission between nodes adopts basic Gossip protocol, and the minimum transmission delay between nodes is 100 ms. The delay varies with the amount of transmitted data.

Initial Distribution of Computing Power. To match reality, we initialize the actual computing power of consensus nodes based on BTC.com's mining pool ranking on Jan 06-12, 2022 [26]. As shown in Fig.3, for a mining pool node with produced

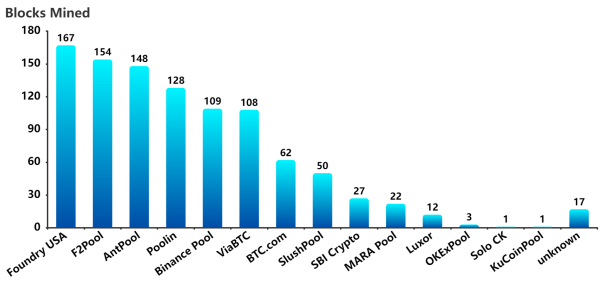


Fig. 3. An Estimation of Blocks Mined by Different Nodes from Jan 06, 2022 to Jan 12, 2022 [26].

block number b_i, its actual computing power is $b_i \times H_0$. Those unknown blocks are considered as being generated by independent nodes, so each node's computing power is assigned H_0.

Proportion of Vulnerable Nodes. Vulnerable nodes mean the nodes that are easily conquered by malicious nodes through single-point attacks etc., and prevented from putting the produced blocks into the main chain after they are determined to be the producer in a certain round. By adjusting the proportion of vulnerable nodes $R_{vul} \in [0, 32\%]$, we can get the comparison of the algorithms' performance under different attack circumstances.

Epoch Length. The epoch length Δ will affect the adjustment rate and effect of the variance of block-producing frequency. We will compare the changes in the stable value of variance of block-producing frequency under different epoch lengths, to determine the optimal range of epoch length[11].

B. Comparison Algorithms

The *Themis* algorithm contains two versions: *Themis* (with GEOST) and *Themis-Lite* (with GHOST [28]). We compare our algorithms with the widely used PoW [16] algorithm (*PoW-H* [12]) and the most adopted consensus algorithm (PBFT [8]) in consortium blockchains.

C. Metrics

The related evaluation metrics are as follows:

- **Variance of Block-producing Frequency** (σ_f^2): Block-producing frequency is defined as the ratio of the number of blocks produced by consensus node to the total number of blocks in the main chain within a period of time. The smaller the variance of block-producing frequency, the higher *Equality* the algorithm achieves.
- **Variance of Block-producing Probability** (σ_p^2): Block-producing probability is defined as the ratio of the effective computing power of consensus node to the total effective computing power in the network during each consensus round. The smaller the variance of block-producing probability, the higher *Unpredictability* the algorithm achieves.
- **Transactions Per Second (TPS)**: *TPS* is to evaluate the average transaction processing rate. Under the premise of a certain transaction throughput, as the consensus nodes' scale increases, the algorithm with smaller change in *TPS* achieves higher *Scalability*.
- **Fork Duration**: From the start to the end block height during a fork in the block tree.

D. Evaluation Results

Equality : As shown in Fig.4, in PBFT, consensus nodes produce blocks in a round-robin manner, and its *Equality* is the best. The *Themis* algorithm greatly improves the *Equality* compared to *PoW-H* through the self-adaptive difficulty adjustment node election mechanism and the GEOST main chain consensus rule. When the algorithm converges, the variance of block-producing frequency of *Themis* and *Themis-Lite* is only 10.80% and 12.16% of that of *PoW-H*. Moreover, *Themis* performs better than *Themis-Lite*, indicating that the GEOST rule helps further improve the *Equality* by choosing the most equal chain when a fork occurs.

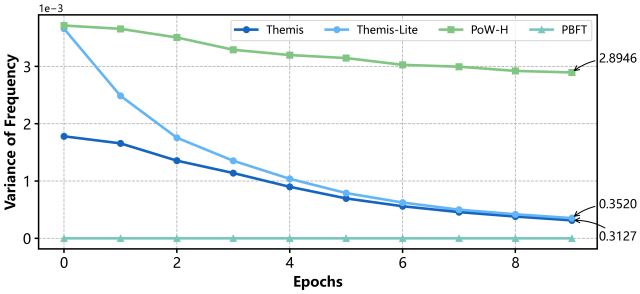


Fig. 4. *Equality* (lower the better): variance of block-producing frequency against epochs.

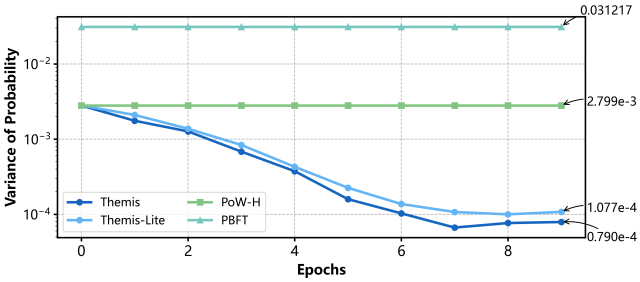


Fig. 5. *Unpredictability* (lower the better): variance of block-producing probability against epochs.

Unpredictability : As shown in Fig.5, *Themis* achieves the best *Unpredictability*. When the algorithm converges, the variance of block-producing probability is only 2.82% of that of *PoW-H*. The *Themis-Lite* algorithm is slightly less unpredictable. Its variance of block-producing probability increases by 2.87×10^{-5} compared to *Themis*, and is 3.85% of that of *PoW-H*. Although PBFT has the best *Equality*, its each consensus round is completely predictable. Thus the *Unpredictability* differs by orders of magnitude from other algorithms, the variance of block-producing probability is 395 times that of *Themis* and 11 times that of *PoW-H*. In addition, the actual block-producing frequency of consensus nodes in *Themis* algorithm obeys a binomial distribution. After the algorithm converges, the variance of block-producing probability of *Themis* and *Themis-Lite* fluctuates to a certain extent.

Scalability : As shown in Fig.6, *PoW-H*, *Themis* and *Themis-Lite* algorithms perform basically the same (*TPS* varies within 20), and are significantly better than the PBFT algorithm. When the scale of nodes is small, each algorithm has a good performance (> 1000). But as the number of consensus nodes

[11]The following evaluations are carried out under the configuration of $\Delta = 8n$ (in the optimal range).

[12]*PoW-H* improves the Bitcoin PoW [16] algorithm, with GHOST [28] as its main chain consensus rule.

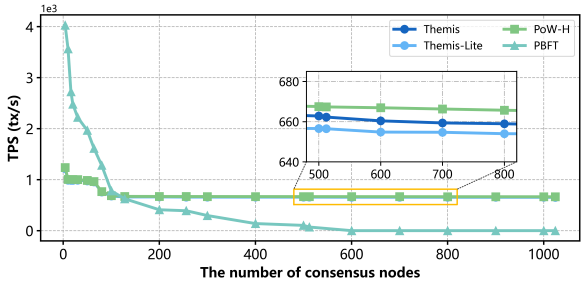


Fig. 6. *Scalability* (higher the better): TPS against the number of consensus nodes.

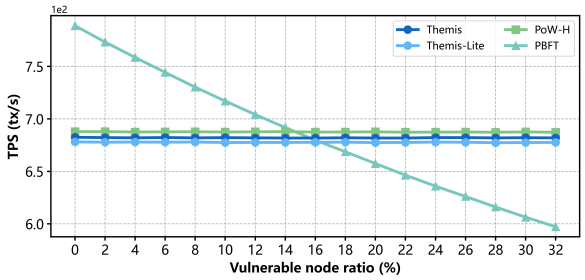


Fig. 7. *Attack Scenarios* (higher the better): TPS against vulnerable node ratio.

increases, the *TPS* of PBFT algorithm drops rapidly. When the number of nodes is over 200, the *TPS* of PBFT rapidly decreases to below 500. And when the number of nodes reaches 600, the *TPS* of PBFT almost hits 0, while the *TPS* of the remaining three algorithms still remains around 650.

Attack Scenarios : Fig.7 illustrates the performance comparison of 4 algorithms under the malicious attacks on block producers [13]. As the proportion of vulnerable nodes increases, *PoW-H*, *Themis* and *Themis-Lite* algorithms can maintain a relatively stable *TPS*, while the *TPS* of PBFT drastically reduces. On the one hand, this is because the three algorithms have better *Unpredictability* than PBFT. On the other hand, in these three algorithms, even if the attacks successfully restrict a node from producing blocks, other nodes can still continue the consensus on schedule, with a little increase on the block interval in that round. But in PBFT, a timeout mechanism will be triggered once a successful attack launched, and the block interval will greatly increase.

Fork Duration : Both *Themis* and *PoW-H* have forks[14]. In order to measure the overhead caused by forks, we carry out 6 experiments respectively under the same block-producing difficulty and block interval settings, and calculate the longest fork duration and the average fork rate. As shown in Fig.8, *PoW-H* has the lowest overhead, its fork rate is 4.36%. Generally, it takes 1-2 blocks to converge, while *Themis* and *Themis-Lite*

[13]For the sake of fairness, the number of consensus nodes for each algorithm is 100.

[14]Since PBFT is insensitive to nodes' computing power and has no forks, there is no need to experiment on the its fork duration.

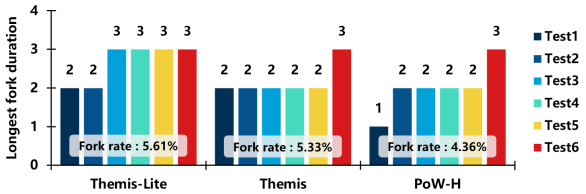


Fig. 8. *Fork Duration* (lower the better): fork duration among three algorithms.

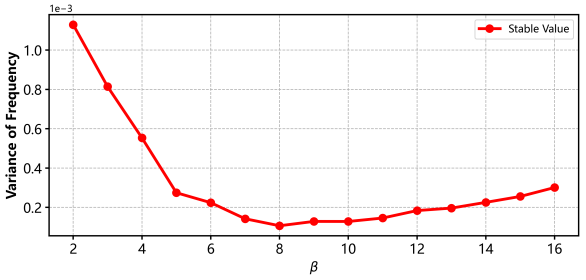


Fig. 9. Variance of block-producing frequency against β.

have a lower variance of block-producing probability. So under the same settings, the fork duration (requiring 2-3 blocks to converge) and the fork rate (5.33% and 5.61%, respectively) both increased a little. By comparing *Themis* and *Themis-Lite*, we find that, compared to GHOST [28], GEOST can effectively reduce the longest fork duration and fork rate.

Epoch Length : The performance of *Themis* algorithm is affected by the length of the difficulty adjustment epoch. To measure this impact, we compare the relationship between the stable value[15] of the variance of block-producing frequency and different epoch lengths ($\Delta = \beta n$, $\beta \in [2, 16]$) at the same block height. As shown in Fig.9, we find that as β increases, the stable value of the variance of block-producing frequency shows a trend of first decreasing and then increasing. This is because when β is small, the block-producing frequency fluctuates sharply and there exists a large gap with the expected block-producing frequency; when β is large, high computing power nodes have already produced many blocks in the counting epoch, which weakens *Equality*. Therefore, we recommend setting $\beta \in [7, 11]$ in *Themis*' actual deployment.

VIII. Related Work

This section mainly introduces some recent representative work on blockchain performance optimization. We start by categorizing probabilistic and deterministic consensus algorithms' major features and state-of-the-art algorithms. Then, we introduce two types of blockchain scalability solutions (Sharding and Layer-2) orthogonal to *Themis* consensus algorithm.

A. Consensus

According to the consistency of blockchain data, consensus algorithms can be divided into two categories: probabilistic

[15]Average variance of block-producing frequency for the last 5 epochs.

consensus and deterministic consensus. Table.I shows the comparison among the *Themis* algorithm and five representative consensus algorithms in terms of *Equality*, *Unpredictability* and *Scalability*. Only the *Themis* algorithm meets the three requirements simultaneously.

Probabilistic Consensus. Consensus nodes reach an agreement on the block data with a probability that gradually increases as time passes. In general, probabilistic consensus algorithms [16]–[22], represented by PoW [16] and PoS [19], provide weak consistency. A probabilistic consensus is mainly adopted in public blockchains. Consensus nodes compete for generating new blocks by solving Proof-of-Work puzzle etc., and determine the main chain through the longest chain rule [16] etc. Nodes don't need additional communication so the network scales. However, the block-producing probability is positively correlated to the invested resources (e.g. computing power), and the resources cannot be limited. Nodes tend to invest their resources in mining pools to obtain stable income. The centralization issue weakens the algorithm's *Equality* and *Unpredictability*.

Deterministic Consensus. The block data cannot be changed once an agreement is reached, which provides a strong consistency. The representative algorithms include PBFT [8], Algorand [11], HoneyBadgerBFT [12], and $Pompē$ [13]. A deterministic consensus is mainly adopted in consortium blockchains where identity authentication is required and finalizes the current round result by "multi-phase communication + node signatures". Traditional BFT-based consensus algorithms such as PBFT [8] can provide a good *Equality*, however the large communication overhead and the block producer being predictable severely limit *Scalability* and *Unpredictability* of such algorithms. To address the requirements of different scenarios, some of the latest algorithms [11] combine the economic incentive mechanism in public blockchains to improve *Scalability*, but weaken *Equality* and *Unpredictability*; some [12], [13], [15] remove the node election mechanism, but the extra communication overhead limits *Scalability*.

B. Sharding

Sharding [34]–[39] is a technology that increases the scalability of blockchain. By modifying the blockchain architecture, sharding divides the consensus node set into multiple small groups, namely shards. Each shard can process transactions in parallel, so as to increase the overall TPS. Elastico [34] is the first proposed sharding solution. It passes transactions to different shards for parallel processing and realizes "transaction sharding". However, consensus nodes in Elastico [34] need to store all shards' information. Solutions such as OmniLedger [35], RapidChain [36], and Monoxide [37] have proposed architectures which realize "state sharding". In these solutions, each node only needs to store the information of its own shard and part of the global information used for processing the cross-shard transactions. OptChain [38], based on the $PageRank$ algorithm, passes transactions to different shards according to the correlation between the new coming transactions and the historical transactions of each shard, which greatly reduces the proportion of cross-shard transactions. Pyramid [39] proposes a layered sharding architecture, which enables a cross-shard transaction to be submitted in multiple related shards within a consensus round through bridge-shards.

TABLE I
COMPARISON OF CONSENSUS ALGORITHMS

	Equality	Unpredictability	Scalability
PoW [16]	△	△	◯
PBFT [8]	◯	×	×
Algorand [11]	△	△	◯
HoneyB. [12]	−	−	×
$Pompē$ [13]	−	−	×
Themis	◯	◯	◯

Note: "◯" indicates that the algorithm can achieve the goal; "×" indicates that the algorithm cannot meet the goal; "△" indicates that the algorithm can meet the goal, but still needs improvement; "−" indicates that the algorithm's design does not consider the goal.

C. Layer-2

The Layer-2 solutions [40]–[44] realize the high-efficiency and low-cost circulation of the on-chain state through the design of novel mechanisms like on-chain state locking, off-chain state circulation, and on-chain state update. These make the expansion of the overall system's transaction throughput come true. Off-chain two-party channel and off-chain channel network [40] are early Layer-2 solutions. Users who directly build a channel can process any number of transactions off chain. Users who do not directly build a channel can complete off-chain transactions through a routing algorithm [41]. In recent years, in order to adapt to different scenarios, more channel solutions such as virtual channel [42] and multi-party channel [43] have also been proposed. To achieve more complex operation like off-chain contract execution, researchers proposed Roll-Up [44], which periodically uploads the compressed information of off-chain transactions and states on chain so that users can experience the same smart contract functions as those on-chain.

IX. CONCLUSION

In this paper, we firstly analyze the new trends in the development of consortium blockchain, and specify three basic goals of consortium blockchain consensus algorithm design: *Equality*, *Unpredictability* and *Scalability*. With regards to this, we propose a consensus algorithm, *Themis*, for consortium blockchains. The experimental results show that, compared with PoW, *Themis* reduces the variance of block-producing frequency by 89.20% (reflecting *Equality*) and variance of block-producing probability by 97.18% (reflecting *Unpredictability*), and guarantees the same *Scalability*. Compared to PBFT, *Themis* avoids the disadvantages of the PBFT algorithm in terms of *Unpredictability* and *Scalability*, and achieves the same level of *Equality* after the algorithm converges in a few consensus rounds.

ACKNOWLEDGEMENTS

The work described in this paper was supported by the National Key R&D Program of China (2019YFB1404903), and the National Natural Science Foundation of China (61972382).

REFERENCES

[1] Hyperledger Fabric. https://github.com/hyperledger/fabric, 2022.
[2] FISCO BCOS. https://github.com/FISCO-BCOS/FISCO-BCOS, 2022.
[3] Y. Chen, C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," in *Journal of Business Venturing Insights*. 2020, 13: e00151.
[4] H. Sun, X. Wang and X. Wang, "Application of Blockchain Technology in Online Education," in *International Journal of Emerging Technologies in Learning (iJET)*. 2018, 13(10), pp. 252-259.
[5] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *Proceedings of IEEE 18th International Conference on R-health Networking, Applications and Services (Healthcom)*. 2016, pp. 1-3.
[6] J. Li, F. Qu, X. Tu, T. Fu, J. Guo and J. Zhu, "Public philanthropy logistics platform based on blockchain technology for social welfare maximization," in *Proceedings of the 8th International Conference on Logistics, Informatics and Service Sciences (LISS)*. 2018, pp. 1-9.
[7] Y. Yuan, F. Wang, "Towards blockchain-based intelligent transportation systems," in *Proceedings of IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 2663-2668.
[8] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1999, pp. 173–186.
[9] B. Guo, Z. Lu, Q. Tang, J. Xu and Z. Zhang, "Dumbo: Faster asynchronous bft protocols," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020, pp. 803-818.
[10] M. Yin, D. Malkhi, M. Reiter, G. Gueta and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*. 2019, pp. 347-356.
[11] Y. Gilad, R. Hemo, S. Micali, G. Vlachos and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*. 2017, pp. 51-68.
[12] A. Miller, Y. Xia, K. Croman, E. Shi and D. Song, "The Honey Badger of BFT Protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2016, pp. 31–42.
[13] Y. Zhang, S. Setty, Q. Chen, L. Zhou and L. Alvisi, "Byzantine ordered consensus without Byzantine oligarchy," in *Proceedings of the 14th Symposium on Operating Systems Design and Implementation (OSDI)*. 2020, pp. 633-649.
[14] Q. Wang and R. Li, "A Weak Consensus Algorithm and Its Application to High-Performance Blockchain," *IEEE Conference on Computer Communications (INFOCOM)*. 2021.
[15] M. Kelkar, F. Zhang, S. Goldfeder and A. Juels, "Order-fairness for Byzantine Consensus," in *Annual International Cryptology Conference (CRYPTO)*. Springer, 2020, pp. 451-480.
[16] S. Nakamoto, "A peer-to-peer electronic cash system," Bitcoin.org, 2008.
[17] I. Eyal, A. Gencer, E. Sirer and R. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proceedings of the 13th Symposium on Networked Systems Design and Implementation (NSDI)*. 2016, pp. 45-59.
[18] E. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*. 2016: 279-296.
[19] S. King, S. Nadal, "Ppcoin: Peer-to-peer Crypto-currency with Proof-of-stake," peercoin.net, 2012.
[20] I. Bentov, A. Gabizon, A. Mizrahi, "Cryptocurrencies without proof of work," in *Proceedings of International Conference on Financial Cryptography and Data Security (FC)*. 2016, pp. 142-157.
[21] A. Kiayias, A. Russell, B. David and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference (CRYPTO)*. Springer, 2017, pp. 357–388.
[22] C. Badertscher, P. Gaži, A. Kiayias, A. Russell and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2018, pp. 913-930.
[23] R. Pass, E. Shi, "Hybrid Consensus: Efficient Consensus in the Permissionless Model," in *Cryptology ePrint Archive*. 2016, https://eprint.iacr.org/2016/917.
[24] R. Pass, E. Shi, "FruitChains: A Fair Blockchain," in *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC)*. 2017, pp. 315–324.
[25] F. Gai, B. Wang, W. Deng and P. Wei, "Proof of reputation: A reputation-based consensus protocol for peer-to-peer network," in *International Conference on Database Systems for Advanced Applications (DASFAA)*. Springer, 2018, pp. 666-681.
[26] BTC.com, "BTC Pool Ranking," btc.com, 2022.
[27] R. Pass, L. Seeman and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2017, pp. 643–673.
[28] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Proceedings of International Conference on Financial Cryptography and Data Security (FC)*. 2015, pp. 507-527.
[29] I. Eyal, E. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436-454.
[30] D. Williams. "Probability with martingales," *Cambridge University Press*. 1991.
[31] Y. Shahsavari, K. Zhang and C. Talhi, "A Theoretical Model for Fork Analysis in the Bitcoin Network," in *2019 IEEE International Conference on Blockchain (Blockchain)*. 2019, pp. 237-244.
[32] Blockchair, "Bitcoin Average Block Size Chart," blockchair.com, 2022.
[33] Etherscan, "Ethereum Average Block Size Chart," etherscan.io, 2022.
[34] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2016, pp. 17-30.
[35] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 583-598.
[36] M. Zamani, M. Movahedi and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2018, pp. 931-948.
[37] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proceedings of the 16th Symposium on Networked Systems Design and Implementation (NSDI)*. 2019, pp. 95-112.
[38] L. Nguyen, T. Nguyen, T. Dinh and M. Thai, "Optchain: optimal transactions placement for scalable blockchain sharding," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 2019, pp. 525-535.
[39] Z. Hong, S. Guo, P. Li and W. Chen, "Pyramid: A Layered Sharding Blockchain System," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. 2021.
[40] J. Poon, T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," https://lightning.network/lightning-network-paper.pdf, 2016.
[41] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti and M. Alizadeh, "High Throughput Cryptocurrency Routing in Payment Channel Networks," in *Proceedings of the 17th Symposium on Networked Systems Design and Implementation (NSDI)*. 2020, pp. 777-796.
[42] S. Dziembowski, L. Eckey, S. Faust and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 106-123.
[43] Y. Ye, Z. Ren, X. Luo, J. Zhang and W. Wu, "Garou: An efficient and secure off-blockchain multi-party payment hub," *IEEE Transactions on Network and Service Management*. 2021, 18(4), pp. 4450-4461.
[44] H. Kalodner, S. Goldfeder, X. Chen, S. Weinberg and E. Felten, "Arbitrum: Scalable, private smart contracts," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security)*. 2018, pp. 1353-1370.