Journal of Electronic Research and Application, 2026, Volume 10, Issue 1
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

NI-HotStuff: A Reputation-Driven Committee Framework for Efficient and Robust BFT Consensus

Long Zheng*

School of Computer Science and Technology, Taiyuan Normal University, No.319 Daxue Street, Yuci, 030619, China

**Corresponding author: Long Zheng, 1649635464@qq.com*

Abstract: Consensus mechanisms are fundamental to blockchain systems, ensuring that distributed nodes agree on the validity of transactions and data. However, performance bottlenecks, particularly those related to throughput, latency, and node selection, have increasingly constrained the scalability of modern blockchain deployments. To address these issues, this paper proposes NI-HotStuff, a reputation-driven committee-based BFT consensus framework built upon the HotStuff protocol. A CatBoost-based reputation model is introduced to learn and evaluate historical behavioral features of nodes, enabling quantitative reputation scoring. A hardware-aware bidding mechanism is further incorporated to dynamically compute each node's bid value and integrate it with its reputation score, thereby prioritizing stable and high-performance nodes for consensus participation. Moreover, a committee mechanism is established in which a set of committee nodes were selected from the candidate pool, and only committee members participate in the consensus process, reducing redundant communication and mitigating the performance drag caused by weak nodes. On top of that, a leader-selection strategy based on reputation values and inter-view time intervals is designed to prevent low-reputation or potentially malicious nodes from frequently becoming leaders. Experimental results demonstrate that NI-HotStuff significantly outperforms traditional PBFT and HotStuff in terms of communication overhead, consensus latency, and system throughput, with particularly notable improvements in small- and medium-scale node environments.

Keywords: Blockchain; Consensus algorithm; HotStuff; Reputation model; Bidding mechanism

Online publication: February 12, 2026

1. Introduction

In 2008, Satoshi Nakamoto launched the era of digital currencies driven by blockchain technology through the seminal work "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. As a decentralized and tamper-resistant distributed ledger based on chained data storage, blockchain integrates multiple fundamental technologies, including distributed storage, peer-to-peer communication, consensus mechanisms, and cryptography [2]. By continuously appending blocks that record transactions and information, blockchain ensures data security, transparency, and trustworthiness. Today, it has become an essential component of the modern digital economy,

with applications spanning finance, healthcare, logistics, supply chain management, and many other domains. A schematic diagram of blockchain-based traceability is shown in **Figure 1**. However, as blockchain is increasingly deployed in practical scenarios, improving system performance, particularly the efficiency of consensus, has emerged as one of the core challenges in its technological development [3].

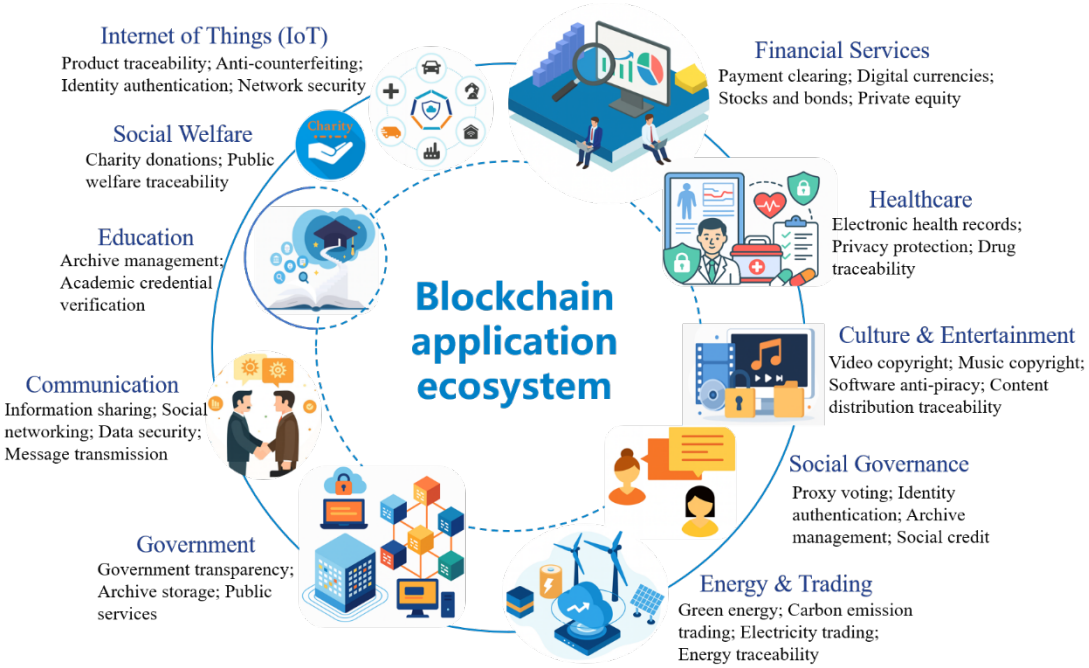

Figure 1. Diagram of blockchain-based traceability scenarios.

Consensus mechanisms constitute the core of blockchain systems, ensuring that all distributed nodes reach agreement on the validity of transactions and data. Commonly used consensus algorithms include Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT) [4]. PBFT has been widely adopted in private and consortium blockchains due to its strong fault tolerance and high security [5]. However, as the number of participating nodes increases, the communication overhead and latency of PBFT grow rapidly, limiting its scalability and applicability in large blockchain networks. To address these challenges, the HotStuff protocol introduces an optimized design. HotStuff employs pipelined execution and simplified signature aggregation to reduce communication rounds among nodes, thereby significantly improving consensus efficiency [6].

Nevertheless, HotStuff still faces bottlenecks in node selection, leader election, and its overall consistency protocol. To mitigate these issues, a variety of enhancements have been proposed in the literature [7]. Some work introduced a novel view-change mechanism that optimizes leader rotation, reducing both communication overhead and delay during leader transitions, and enabling rapid and effective leader replacement when failures occur [8]. This design improves consensus efficiency and enhances overall system performance. Another study proposed the Sync HotStuff protocol, which removes the traditional lock-step execution constraints in synchronous BFT systems and adopts a two-phase leader-driven structure that enables replicas to advance the next round of consensus without waiting for the previous commit [9]. This significantly reduces latency bottlenecks and improves throughput in synchronous BFT settings. Furthermore, it was introduced that weighted voting and an enhanced leader-rotation strategy, yielding substantial improvements from vote distribution to leader replacement within the HotStuff framework [10].

To address the limitations of HotStuff in node selection and leader election, this paper proposes NI-HotStuff, a reputation-driven committee-based BFT consensus framework. The main contributions are as follows:

(1) Construct a reputation evaluation and node classification module that generates dynamic reputation scores for all nodes based on their historical behavior features using the CatBoost model;

(2) A node bidding mechanism is introduced to dynamically compute each node's bid value by combining its hardware resources with its reputation score. This mechanism enables the system to select well-provisioned nodes for consensus participation, thereby reducing the involvement of low-performance nodes and improving both fault tolerance and overall consensus efficiency;

(3) A committee formation and leader-selection strategy is designed. From the candidate set with reputation scores above a minimum threshold, committee members are selected through weighted random sampling, and only committee nodes participate in the consensus process. Within the committee, the leader is dynamically chosen based on both node reputation and inter-view time intervals, preventing malicious or low-reputation nodes from frequently serving as leaders and enhancing the fairness, stability, and overall efficiency of the consensus protocol.

2. Related work

2.1. PBFT algorithm

The Byzantine Generals Problem was introduced by Lamport *et al.* to describe the consensus challenge faced by distributed systems in the presence of malicious nodes [11]. To address this issue, Castro *et al.* proposed the PBFT algorithm, which aims to ensure that all nodes can reach agreement even in Byzantine environments. The core objective of PBFT is to guarantee consistent consensus despite the presence of faulty or adversarial nodes. Specifically, in a blockchain network with n nodes, the system can tolerate up to f Byzantine nodes, where $f=(n-1)/3$. The PBFT algorithm proceeds through several phases, and the overall consensus workflow is illustrated in **Figure 2** and as follows:

(1) Request phase: The client sends a request to the primary node and waits for a response;

(2) Pre-prepare phase: Once the primary node receives the client request, it generates a unique identifier and creates a proposal for the request. The proposal is then broadcast to all replica nodes;

(3) Prepare phase: Upon receiving the pre-prepare message, each replica generates a prepare message and broadcasts it to all nodes. When a node receives more than $2f$ matching prepare messages that are consistent with the pre-prepare message, it proceeds to the commit phase;

(4) Commit phase: The node generates a commit message and broadcasts it to all other nodes. When it receives more than $2f$ matching commit messages that correspond to the same pre-prepare message, it sends a response to the client;

(5) Reply phase: When the client receives more than f matching responses, it concludes that the transaction has been successfully completed.

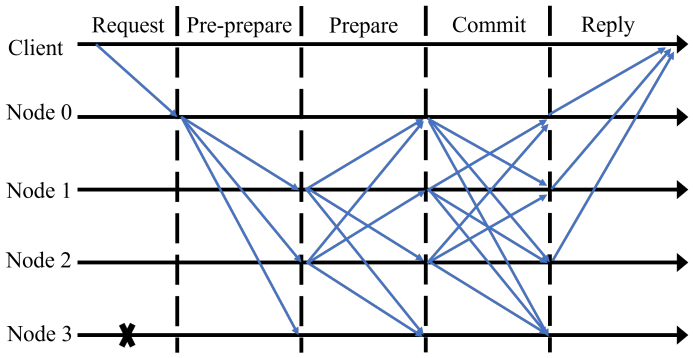

Figure 2. PBFT consensus process.

2.2. HotStuff algorithm

HotStuff is an improved Byzantine Fault Tolerant consensus protocol that operates in an environment similar to PBFT but maintains system state using a tree-structured architecture rather than a traditional log-based structure. Meanwhile, HotStuff extends the traditional two-phase commit procedure into a three-phase commit pipeline, reducing the communication complexity of view change from to and thereby significantly lowering the communication burden as the system scales. By adopting a one-to-many message dissemination pattern, HotStuff effectively compresses communication overhead and completes consensus within a single view [12]. Each view is associated with a unique, monotonically increasing view number and is coordinated by a designated leader. Upon completing a view, the system automatically progresses to the next one. If the current leader behaves maliciously or becomes faulty, a new leader takes over after a view timeout and continues the consensus process. Unlike PBFT, HotStuff embeds the view-change mechanism within every round of consensus, rotating the leader after each committed block. The overall workflow proceeds through three phases, prepare, pre-commit, and commit. In each phase, the leader must collect at least *n-f* replica votes to form a quorum certificate (QC), which serves as proof of intermediate agreement. The execution of the algorithm consists of the following stages, and the complete consensus process is illustrated in **Figure 3** and as follows:

(1) Prepare phase: The new leader first collects *n-f* NEW-VIEW messages from different replicas and extracts the prepareQC with the highest view number as the base branch (highQC). The leader then constructs a leaf node containing the client command at the end of the current branch and broadcasts a prepare message containing the proposal details to all replicas. Each replica verifies the validity of the proposal, and if it satisfies the branch extension rules, it sends a prepareVote to the leader;

(2) Pre-commit phase: When the leader successfully aggregates *n-f* prepareVote messages, it forms the prepareQC and broadcasts the pre-commit message. After receiving this message, each replica confirms that more than 2*f* replicas have approved the proposal, updates its local prepareQC bstatus, and sends back a commitVote to advance the protocol;

(3) Commit phase: The leader must collect *n-f* preCommitVote messages to generate the preCommitQC. After verifying this certificate, each replica marks the branch of the current proposal as a stable state (updating its lockedQC) and sends a commitVote to the leader, indicating its agreement to commit the proposal;

(4) Decide phase: After obtaining *n-f* commitVote messages, the leader generates the final commitQC and broadcasts the decide message. Replicas confirm that the proposal branch now satisfies commit rules and execute the client command stored in the leaf node. Once the command is executed, all replicas automatically trigger the view transition and proceed to the next consensus round.

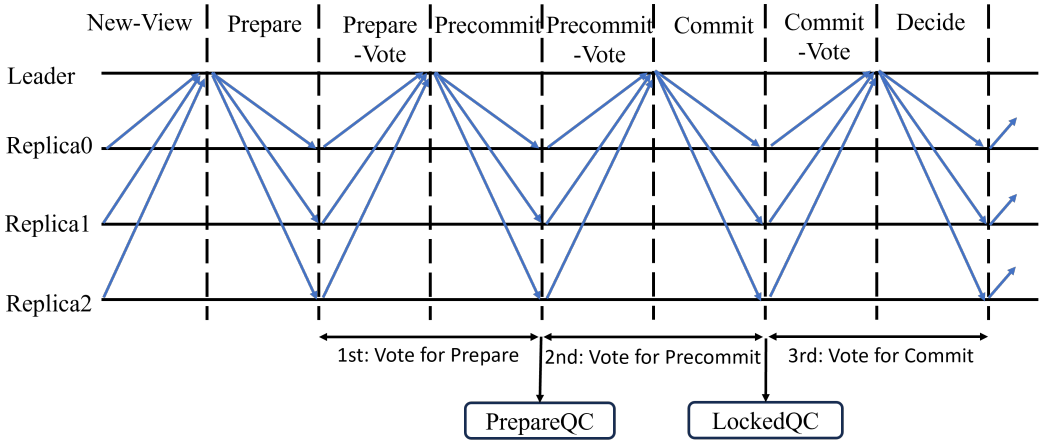

Figure 3. HotStuff consensus process.

2.3. CatBoost model

CatBoost is an ensemble model based on Gradient Boosting Decision Trees, with its most distinctive feature being its method of handling categorical features [13]. Traditional gradient boosting decision trees typically require categorical features to be converted into numerical features, such as one-hot encoding or label encoding [14,15]. In contrast, CatBoost adopts a specialized ordered-boosting algorithm that enables direct processing of categorical attributes without the need for explicit transformation, making it particularly suitable for solving classification problems [16]. By applying the CatBoost algorithm, a node classification model can be constructed to categorize nodes on the blockchain into different classes. The details are as follows:

Given a dataset with m features $\varphi=<x_1,x_2,...x_m>$ and label δ, a training set T with n samples can be formed as $T=\{[\varphi_1,\delta_1],[\varphi_2,\delta_2],...[\varphi_n,\delta_n]\}$. The CatBoost algorithm performs iterative optimization, where each iteration constructs a model that minimizes the loss function using gradient boosting decision trees. After k rounds of boosting, the optimal weak learner h_k for the final iteration is obtained, and all weak learners from each iteration are combined to form the final predictor F_k. As shown in **Equation (1)**:

$$\begin{cases} F_k = F_{k-1} + \alpha h_k \\ h_k = \arg min \sum_{i=1}^{m} \beta_i L(y_i, F_{k-1}(x_i) + h_k(x_i)) \end{cases} \quad (1)$$

Here, α is the learning rate, which controls the update step size of the model parameters in each boosting iteration. The operator *min* denotes the minimization of the objective function, β_i represents the sample weight, and $L()$ is the loss function used to measure the deviation between the true value and the predicted value of a sample. The term $F_{k-1}(x_i)+h_k(x_i)$ denotes the output of the k-th learner for the input x_i.

For the multi-class classification task in this study, the multinomial cross-entropy loss function is adopted, which is derived from the concept of categorical cross-entropy, defined as

$$L(y_i, p_i) = -\sum_{i=1}^{N} y_i \log(p_i) \quad (2)$$

Here, N denotes the number of classes, y_i represents the probability distribution of the true label, and p_i

represents the probability distribution output by the model.

2.4. Bidding mechanism

The bidding strategy refers to a class of methods in which participants acquire specific resource access rights or benefits by submitting competitive bids based on their own resource valuations and actual demands during the resource allocation and decision-making process [17]. In traditional bidding mechanisms, participants typically consider factors such as initial investment cost, risk preference, and the expected value fluctuations of target resources to determine a reasonable bid. At the same time, participants may engage in strategic interactions and competition, requiring them to assess and anticipate possible bidding behaviors of others and adjust their own strategies accordingly to improve their chances of obtaining the desired resources or services.

Bidding strategies have been widely applied across various domains [18–20]. In the energy management sector, power companies and users utilize bidding schemes to complete electricity trading and scheduling, thereby increasing energy utilization efficiency while reducing operational costs. In the transportation domain, vehicles can obtain preferential passage rights or facility usage rights through reliable bidding to optimize traffic flow and improve the allocation efficiency of transportation resources, easing congestion to some extent. In the field of data sharing and privacy protection, bidding allows participants to negotiate data access rights and usage permissions, enabling controlled and secure data exchange. Furthermore, in Internet-of-Things environments, intelligent devices can dynamically allocate bandwidth and storage resources through bidding, thereby enhancing system responsiveness and improving overall performance. Therefore, introducing bidding mechanisms into blockchain consensus provides new perspectives and support for node selection and resource allocation.

3. NI-HotStuff algorithm

To address the issues in the traditional HotStuff protocol, such as the randomness of node selection, the delayed handling of primary node failures, and the reduced consensus efficiency in the presence of malicious nodes, this paper proposes a reputation-driven committee-based BFT consensus framework, named NI-HotStuff. The proposed framework establishes a reputation evaluation and node classification module, which models historical node behaviors to generate dynamic reputation scores, forming a basis for subsequent committee selection and primary node determination. Building on this foundation, the algorithm incorporates hardware resource indicators and reputation scores to design a dynamic bidding mechanism that comprehensively assesses each node's bidding value. This mechanism ensures that only high-performance nodes participate in the consensus process, thereby reducing interference from low-capacity nodes and improving overall system efficiency. Furthermore, NI-HotStuff introduces a primary-node selection strategy based on reputation scores and time intervals. When the primary node fails or exhibits Byzantine behavior, the protocol can quickly determine a new primary node through reputation-based ranking and priority adjustment, ensuring the continuity and stability of consensus. Before consensus begins, NI-HotStuff performs preprocessing on all participating nodes, collecting and organizing behavioral data to train the reputation model. After consensus starts, the protocol follows the consistency procedure described in Section 3.1 to ensure the integrity and correctness of blockchain data.

3.1. Node identification mechanism

In the HotStuff protocol, the primary node is typically selected through a fixed round-robin mechanism based on the view number. This rotation strategy helps mitigate the risk of single-point failure and improves fairness

and robustness to some extent [21]. However, as the network scales, the communication overhead among nodes increases significantly, and fundamental factors such as hardware capability and network conditions can further influence the overall performance of HotStuff. Since the primary node is responsible not only for proposing and broadcasting blocks but also for performing additional bookkeeping operations, selecting an unreliable or Byzantine node as the primary may compromise the security of the consensus process. To address these challenges, NI-HotStuff integrates a CatBoost-based model along with a bidding mechanism to optimize primary node selection. By ranking candidate nodes through a comprehensive scoring scheme, the algorithm prioritizes stable and high-performance nodes for participation in consensus, thereby reducing the likelihood of Byzantine nodes being elected and effectively alleviating communication overhead within the network.

3.1.1. Node competitive value assessment

Under similar conditions, nodes equipped with superior computing and storage resources can process transactions, message broadcasting, and other consensus operations more efficiently, exhibiting faster execution speed and higher processing throughput. Meanwhile, their lower probability of hardware failure provides better stability and resilience against risks. Based on this observation, the NI-HotStuff algorithm introduces a node bidding mechanism into the reputation evaluation process, treating basic resource allocation as an important weighting factor. Before entering each consensus round, nodes submit a bidding value according to their computational and storage capabilities. A higher bidding value not only reflects a node's greater willingness to contribute resources to the consensus process but also increases the likelihood of obtaining a higher reputation score, thereby improving the potential of being selected as the primary node.

In this paper, considering node attributes and interaction characteristics, three key indicators, CPU core count, available storage capacity, and network bandwidth, are selected to measure a node's basic resource level. When a node first joins the network, its maximum available resources are recorded as the basis for subsequent reputation evaluation and the bidding process. Since these indicators differ in magnitude, we adopt normalization to ensure comparability. The normalized value is obtained using **Equation (3)**, where x denotes the original metric value and x' represents the normalized result:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3}$$

After normalization, the bidding value of each node is computed using **Equation (4)**. Specifically, B_i denotes the bidding value of node i; c_i, s_i, and b_i represent the number of CPU cores, available storage capacity, and network bandwidth of node i in the current consensus round, respectively. The parameters α, β, and γ are the weight coefficients corresponding to CPU cores, available storage, and bandwidth, and they satisfy α+β+γ=1. These weights can be dynamically adjusted according to practical system requirements.

$$B_i = \alpha \cdot c_i + \beta \cdot s_i + \gamma \cdot b_i \tag{4}$$

3.1.2. Node reputation score assessment

Before constructing the node identification model, a large number of predefined node samples are first introduced into the blockchain system in a balanced manner, during which the system extracts and transforms node features to obtain the corresponding characteristic values. To ensure the completeness of feature extraction, the system assigns an initial reputation value to each node and categorizes them into high-reputation, medium-reputation,

and low-reputation nodes according to their reputation levels. The correspondence between node categories and reputation values is shown in **Table 1**. During the identification phase, high-reputation nodes are generally capable of efficiently completing tasks such as block exchange and broadcasting, thus improving system verification and consensus performance. Medium-reputation nodes behave similarly to high-reputation nodes but may occasionally exhibit slight deviations that remain within an acceptable range. In contrast, low-reputation nodes often exhibit unfavorable behaviors, such as delayed responses or frequent timeout events, which may adversely affect system consensus efficiency and security.

Table 1. Reputation classification criteria for nodes

Node category	Reputation value
High-reputation node	$r_i > 0.7$
Medium-reputation node	$-0.5 < r_i \leq 0.7$
Low-reputation node	$r_i \leq -0.5$

Note: denotes the reputation value of node i.

The definitions of the four feature attributes are given as follows:

(1) Definition 1: Heartbeat detection, which measures whether a node is online and its responsiveness. It is defined as:

$$H(i) = \frac{r_i}{t_i} \tag{5}$$

where r_i denotes the number of successful heartbeat responses received from node during the observation period, and t_i represents the total number of heartbeat requests sent to node i.

(2) Definition 2: Node contribution, which reflects the node's activity level and assigns higher weights to recent participation through time decay. It is defined as:

$$C(i) = \frac{s_i}{t_i + \alpha} \tag{6}$$

where s_i refers to the number of successful consensus completions by node i during the observation period, t_i is the total number of consensus rounds that node i participated in, and α is a smoothing factor.

(3) Definition 3: Delay index, which reflects the node's delay performance during communication. It is defined as:

$$D(i) = \frac{1}{1 + \frac{\overline{l}_i}{L_{max}}} \tag{7}$$

where $\overline{l}_i$ is the average round-trip latency of node i, and L_{max} denotes the maximum allowable latency in the system.

(4) Definition 4: Network utilization, which reflects the usage condition of the node's network resources. It is defined as:

$$U(i) = \frac{\overline{b}_i}{B_i} \cdot (1 - p_i) \quad (8)$$

where $\overline{b}_i$ is the average effective bandwidth usage of node i, B_i is its available bandwidth capacity, and pi denotes the average packet loss rate of node i.

After training, the CatBoost model is deployed into the operational environment. At the end of each consensus round, behavioral and performance data are collected for each node, and corresponding feature values are computed and broadcast to all nodes. Based on the accumulated feature entries for each node, the CatBoost algorithm performs classification and identification. Furthermore, these features are not only used as classification inputs but also form the feature set for each consensus round. The updated reputation values from the previous round are incorporated as inputs, enabling correction of historical reputation scores. In this manner, the model reflects both long-term node stability and short-term dynamic changes, thereby providing more accurate references for the overall reputation scoring mechanism.

3.1.3. Node comprehensive scoring mechanism

During node selection, relying solely on performance-based metrics to compute the bidding value can reflect a node's computing power, bandwidth, and storage resources, but it fails to capture the node's overall reliability. Similarly, although reputation values derived from behavioral features can effectively identify anomalous behavior or malicious nodes, they overlook differences in hardware resources. Therefore, bidding values emphasize static computational capability, whereas reputation emphasizes dynamic operational behavior, and neither alone offers a complete evaluation. Thus, based on the two categories of metrics introduced earlier, this paper proposes a hybrid evaluation mechanism that integrates the bidding value with the reputation score, thereby preserving both security and performance to obtain a more representative node evaluation outcome.

To achieve this, we introduce a cross-round dynamic reputation update mechanism. The reputation value of node i in round t is defined as:

$$R_i^t = \delta \cdot R_i^{t-1} + (1 - \delta) \cdot f(X_i^t) \quad (9)$$

where R_i^t is the reputation value from the previous round, and $f(X_i^t)$ is the reputation score computed using this round's behavioral feature set (including heartbeat detection, node contribution, delay index, and network utilization). The parameter $\delta \in (0,1)$ is a smoothing factor that balances long-term and short-term behavior, preventing extreme oscillations caused by temporary performance fluctuations.

Based on this, the comprehensive evaluation score of node i is computed as:

$$S_i^t = \lambda \cdot R_i^t + (1 - \lambda) \cdot B_i \quad (10)$$

where B_i denotes the bidding value derived from the node's CPU cores, storage capacity, and available bandwidth, and λ is the weight factor.

Through this evaluation method, the final score not only accounts for the node's fundamental hardware resources but also incorporates its accumulated reputation across rounds, enabling a more holistic node assessment.

3.2. Consistency protocol optimization

Although the HotStuff protocol significantly reduces communication complexity compared with PBFT, in practical deployments it still requires all nodes to participate in consensus, which leads to additional overhead and increases the risk of communication failures. Therefore, NI-HotStuff introduces a committee-based mechanism at the consistency layer. By computing the comprehensive score of each node using the reputation evaluation model and setting a minimum threshold S_{min}, only nodes whose scores exceed this threshold are included in the candidate set C. To ensure the quality and controllable size of the candidate set, this paper adopts a Top-M selection rule: in the t-th round, nodes are sorted in descending order according to their comprehensive score S_i^t, and the top M nodes are selected to form C_t, where $M=K+m$ ($K=3f+1$, m is the redundancy, fixed as $m=max(2,[0.2K])$). The minimum threshold S_{min} corresponds to the M-th highest score. If system fluctuations or abnormal score distribution cause fewer than M nodes to satisfy the threshold, then all valid nodes are included in C_t. In case of ties, the ordering follows the previous-round ranking or the lexicographical order of node IDs. According to Byzantine fault tolerance theory, in a blockchain system with $n>=3f+1$ nodes, only when the committee size contains at least $3f+1$ nodes can it guarantee that, even in the presence of up to f malicious nodes, any two quorums (each larger than $2f+1$) still have non-empty intersection with honest nodes, thus preventing consensus divergence. Hence, the system adopts a weighted random sampling process to select a final committee of size $n'=3f'+1$, denoted as K, and only committee members participate in the consensus procedure.

The probability that node i is selected into the committee K is given by:

$$P_i(i \in K) = \frac{e^{\tau S_i^t}}{\sum_{j \in C} e^{\tau S_j^t}} \tag{11}$$

where S_i^t denotes the comprehensive score of node i in the t-th round, which considers both its reputation value and hardware resource capability. A higher score indicates that the node is more advantageous. The parameter τ is a tuning factor (temperature coefficient) used to control the preference intensity toward high-score nodes; as τ increases, the probability of selecting nodes with higher scores becomes larger. The denominator is the sum of the exponential scores of all nodes in the candidate set C, used to normalize the probability and ensure that the sum of the selection probabilities of all candidate nodes is 1.

Under this mechanism, nodes with higher comprehensive scores are more likely to be selected into the committee, while a certain degree of randomness is preserved to maintain fairness. Meanwhile, by adhering to the safety bound of BFT theory, the system guarantees security and liveness even under potential Byzantine behaviors. Therefore, this mechanism effectively reduces unnecessary communication and computation overhead, improving system reliability and consensus efficiency. The improved consistency protocol is illustrated in **Figure 4**.

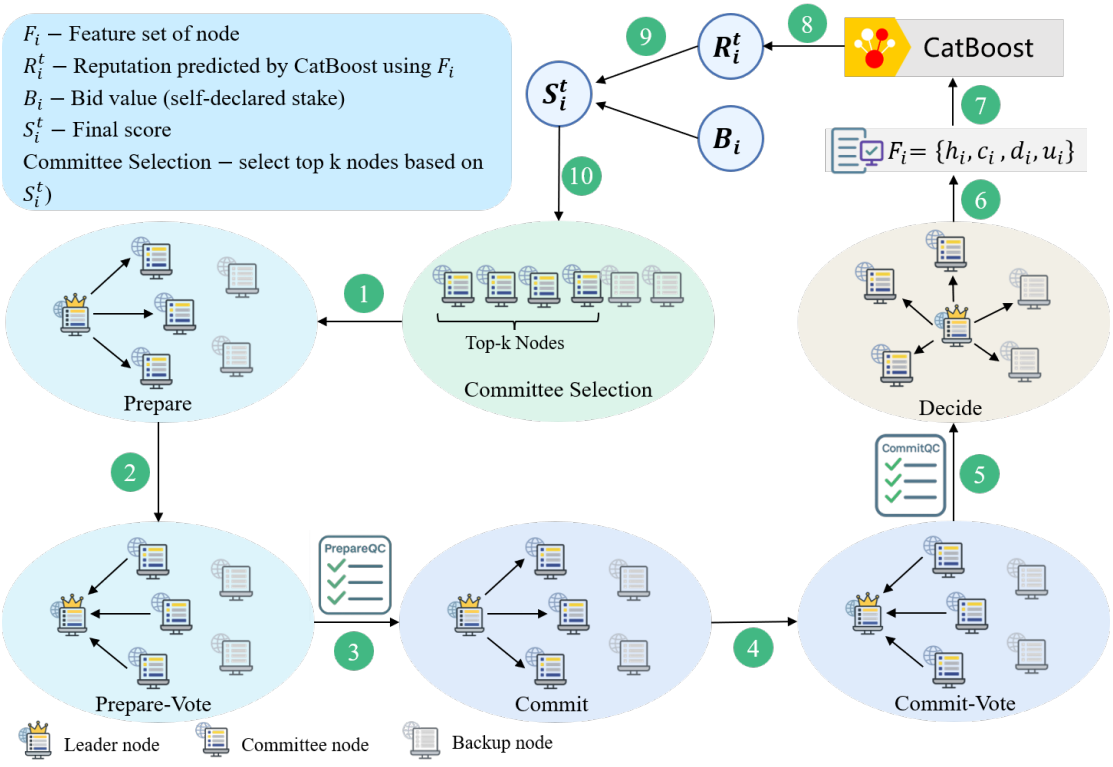

Figure 4. NI-HotStuff consensus process diagram.

3.3. Master node selection

In terms of leader selection, the algorithm recalculates and updates the comprehensive score of each node after every consensus round, thereby refreshing the candidate committee. The priority of a node being selected as the leader is determined by the committee based on its latest comprehensive score. This priority is correlated with the score defined earlier and is also adjusted according to the time interval since the node last served as leader. Nodes with higher recent scores and shorter leader intervals are more likely to obtain higher priority, which effectively reduces the risk that a long-idle node becomes the leader and enhances the fairness and stability of the system. The leader selection priority is defined as follows:

$$M_i = S_i^t \times \frac{\Delta T_i}{T_c} \tag{12}$$

where S_i^t denotes the comprehensive score of node i at round t, as defined in Equation 10. The term ΔT_i represents the time elapsed since node i last served as the leader, and T_c is a leader selection parameter set by the system to adjust the effect of time intervals on the leader selection process.

Before leader selection, all nodes in the blockchain update their priority values. That is, after the previous consensus round, each node's comprehensive score is recalculated according to the reputation model introduced earlier, and the new leader is selected based on the updated priority ranking.

4. Experiment and results analysis

This study develops a lightweight blockchain system using the Python programming language. The experimental

environment is configured as follows: the hardware platform is equipped with an Intel® CoreTM i7-14700HX CPU and 16 GB of RAM; the operating system is Windows 11 (64-bit); and the software environment uses PyCharm. The PBFT, HotStuff, and NI-HotStuff algorithms are simulated and evaluated within this setup.

4.1. Communication overhead analysis

The communication overhead refers to the total number of messages generated during the consensus process. NI-HotStuff adopts a committee-based two-phase pipeline (Prepare→Commit), where only a committee of size $M=3f+1$ participates in intra-phase interactions. According to the phase design, the leader broadcasts once to the committee in each phase, and each replica broadcasts one confirmation message. Therefore, the number of messages in a single phase is $2(M-1)$, and the two phases generate a total of $4M-4$ messages. When a view change occurs, $M-1$ NEW-VIEW messages are introduced in each switch. After v view changes, an additional $v(M-1)$ messages are produced. Hence, the total communication overhead can be approximated as: $4M-4+v(M-1)$. Compared with the original HotStuff, in which all N nodes participate in the three-phase process M, NI-HotStuff reduces the number of participating nodes from to committee size , and reduces the number of phases from three to two, thereby maintaining linear communication complexity $O(M)$ while significantly decreasing constant factors and practical communication overhead.

4.2. Latency analysis

In blockchain systems, consensus latency refers to the time interval from when a client initiates a request to when the request is finalized. Under the same experimental environment, we conducted one-round consensus delay tests for the NI-HotStuff algorithm and several other algorithms. The number of nodes was set to 80, 100, 120, 140, 160, and 180, respectively. Multiple experiments were repeated and averaged. The comparison results of PBFT, HotStuff, and NI-HotStuff are shown in **Figure 5**. It can be observed that as the network scale increases, the consensus latency of all three algorithms shows an upward trend, but to different extents. PBFT, due to its communication complexity of $O(n^2)$, experiences rapidly increasing latency as the number of nodes grows, and at 180 nodes, the delay exceeds 2500 ms, indicating poor scalability. HotStuff reduces communication complexity to $O(n)$ through its three-phase chain-locking and threshold signature optimization, significantly decreasing latency

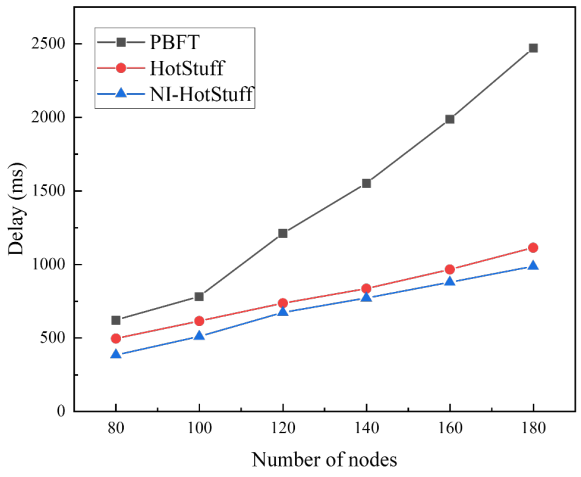

Figure 5. Comparison of consensus latency.

compared with PBFT and exhibiting a more gradual increasing trend overall. Building upon HotStuff, the NI-HotStuff algorithm proposed in this work further reduces the impact of low-performance nodes and frequent view changes through reputation evaluation and dynamic leader selection mechanisms. Therefore, NI-HotStuff maintains the lowest delay across all network sizes, achieving more than a 50% reduction compared with PBFT and a further 15–20% improvement compared with HotStuff, demonstrating superior performance and stability in complex network environments.

4.3. Throughput analysis

The throughput refers to the number of tasks or requests that the system can process per unit of time, which is generally used to measure the speed and transaction-handling capability of blockchain systems. It is usually expressed in TPS (Transactions Per Second), and its calculation formula is:

$$TPS = Transaction_{\Delta T}/\Delta T \tag{13}$$

where $Transaction_{\Delta T}$ denotes the number of transactions, and ΔT represents the time required to complete these transactions.

Under the same experimental settings with equal numbers of nodes and equal transaction loads, the throughput of the NI-HotStuff algorithm and several other algorithms is tested. The number of nodes is selected as 80, 100, 120, 140, 160, and 180, with the number of transactions set to 10. The throughput comparison results are shown in **Figure 6**. From the experimental results, it can be observed that with the increase in network size, the throughput of the three algorithms declines, but the rate of decline differs. PBFT suffers from communication complexity $O(n^2)$, causing throughput to drop sharply when the number of nodes exceeds 100 TPS, revealing a severe scalability bottleneck. HotStuff improves throughput by reducing communication complexity and outperforms PBFT across all scales. Meanwhile, the proposed NI-HotStuff algorithm achieves higher processing capability than HotStuff, with an improvement of over 25% in small-scale networks and potentially maintaining an advantage of about 10% in large-scale settings, demonstrating its efficiency and adaptability.

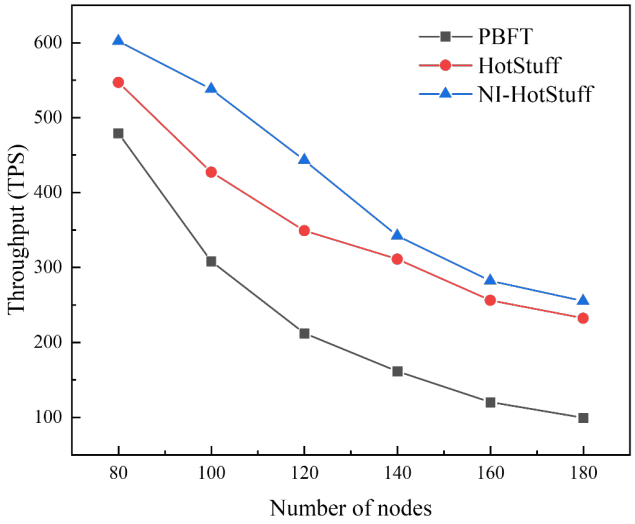

Figure 6. Throughput comparison.

5. Conclusion

To address the issues of arbitrary node selection, insufficient handling of faulty primary nodes, and performance degradation in the presence of Byzantine nodes in the traditional HotStuff consensus algorithm, this paper proposes a reputation-driven committee-based BFT consensus framework, termed NI-HotStuff. The proposed method first evaluates node reputation using a CatBoost model and incorporates competitiveness-based quantification of hardware resources to derive a weighted composite score for node selection. Meanwhile, a committee mechanism is introduced, where nodes whose reputation values exceed the minimum threshold form the candidate set, and a weighted random sampling strategy is employed to select the committee members. Only committee nodes participate in the consensus process, effectively reducing redundant communication and lowering system complexity. Furthermore, the algorithm designs a dynamic primary node selection strategy based on node reputation and time intervals, prioritizing committee members with high reputation and less frequent prior leadership, thereby significantly reducing the probability of low-performance or malicious nodes taking the primary role. Experimental results demonstrate that NI-HotStuff outperforms PBFT and HotStuff in terms of communication overhead, consensus latency, and system throughput, validating its higher efficiency and scalability while ensuring security and fairness.

Disclosure statement

The author declares no conflict of interest.

References

[1] Nakamoto S, Bit B, 2008, Bitcoin: A Peer-to-Peer Electronic Cash System. 2008, 2007.

[2] Yuan F, Zuo Z, Jiang Y, et al., 2015, AI-Driven Optimization of Blockchain Scalability, Security, and Privacy Protection. Algorithms, 18(5): 263.

[3] Amiri M, Wu C, Agrawal D, et al., 2024, The Bedrock of Byzantine Fault Tolerance: A Unified Platform for BFT Protocols Analysis, Implementation, and Experimentation. 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), 371–400.

[4] Yuan F, Huang X, Zheng L, et al., 2025, The Evolution and Optimization Strategies of a PBFT Consensus Algorithm for Consortium Blockchains. Information, 16(4): 268.

[5] Castro M, Liskov B, 1999, Practical Byzantine Fault Tolerance. OsDI, 99(1999): 173–186.

[6] Yin M, Malkhi D, Reiter M, et al., 2019, HotStuff: BFT Consensus with Linearity and Responsiveness. Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, 2019: 347–356.

[7] Malkhi D, Yin M, 2023, Lessons from HotStuff. Proceedings of the 5th Workshop on Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating Algorithms for Distributed Systems, 2023: 1–8.

[8] Jalalzai M, Niu J, Feng C, et al., 2023, Fast-Hotstuff: A Fast and Robust BFT Protocol for Blockchains. IEEE Transactions on Dependable and Secure Computing, 21(4): 2478–2493.

[9] Abraham I, Malkhi D, Nayak K, et al., 2020, Sync Hotstuff: Simple and Practical Synchronous State Machine Replication. 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020: 106–118.

[10] Liang B, Yuan F, Deng J, et al., 2025, Cs-pbft: A Comprehensive Scoring-Based Practical Byzantine Fault Tolerance Consensus Algorithm. The Journal of Supercomputing, 81(7): 859.

[11] Lamport L, Shostak R, Pease M, 2019, The Byzantine Generals Problem, Concurrency: The Works of Leslie

Lamport, 203–226.

[12] Zhang Z, Hu B, Tian L, et al., 2025, Efficient Dynamic-Committee BFT Consensus Based on HotStuff. Peer-to-Peer Networking and Applications, 18(3): 111.

[13] Prokhorenkova L, Gusev G, Vorobev A, et al., 2018, CatBoost: Unbiased Boosting with Categorical Features. Advances in Neural Information Processing Systems, 2018: 31.

[14] Breskuvienė D, Dzemyda G, 2023, Categorical Feature Encoding Techniques for Improved Classifier Performance When Dealing with Imbalanced Data of Fraudulent Transactions. International Journal of Computers Communications & Control, 18(3).

[15] Wang Z, Chen L, Wang F, 2023, Fuzzy Inference Attention Module for Unsupervised Domain Adaptation. IEEE Transactions on Fuzzy Systems, 32(4): 1706–1718.

[16] Wang Z, Wang X, Liu F, et al., 2021, Adaptative Balanced Distribution for Domain Adaptation with Strong Alignment. IEEE Access, 2021(9): 100665–100676.

[17] Ou W, Chen B, Dai X, et al., 2023, A Survey on Bid Optimization in Real-Time Bidding Display Advertising. ACM Transactions on Knowledge Discovery from Data, 18(3): 1–31.

[18] PankiRaj J, Yassine A, Choudhury S, 2019, An Auction Mechanism for Profit Maximization of Peer-to-Peer Energy Trading in Smart Grids. Procedia Computer Science, 2019(151): 361–368.

[19] Malik S, Thakur S, Duffy M, et al., 2023, Comparative Double Auction Approach for Peer-to-Peer Energy Trading on Multiple Microgrids. Smart Grids and Sustainable Energy, 8(4): 21.

[20] Dramitinos M, Stamoulis G, Courcoubetis C, 2007, An Auction Mechanism for Allocating the Bandwidth of Networks to their Users. Computer Networks, 51(18): 4979–4996.

[21] Yin M, Malkhi D, Reiter M, et al., 2018, HotStuff: BFT Consensus in the Lens of Blockchain, arXiv, https://doi.org/10.48550/arXiv.1803.05069