Journal of Electronic Research and Application, 2026, Volume 10, Issue 2
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

RGS-PBFT: A Reputation-Based Grouping and Supervision PBFT

Xiangyuan Qi, Guanglei Qiang*, Qianxiang Zhang, Zhuokun Fan, Yu Du

School of Computer Science and Technology, Taiyuan Normal University, Jinzhong 030619, Shanxi, China.

**Corresponding author: Guanglei Qiang, 202325502004@stu.tynu.edu.cn*

Abstract: To address the problems of high communication overhead, insufficient robustness in master node election, and difficulty in timely management of faulty nodes in large-scale networks, the traditional PBFT consensus algorithm is proposed as a reputation-based grouping and supervision PBFT (RGS-PBFT). This method constructs a dynamic reputation score model based on the behavioral characteristics of nodes during the consensus process, such as verification correctness, voting participation, and operational stability. Based on this model, nodes are divided into consensus groups, candidate node groups, and ordinary node groups, enabling adaptive adjustment of node roles. During the master node election phase, the top 5% of nodes with the highest reputation scores from the consensus groups are selected as the candidate set. The ordinary node group and the candidate node group elect the master node through reputation-weighted voting, thus balancing the reliability of the master node with the decentralized nature of the election process. During the consensus process, only the consensus group participates in the core voting and confirmation process of PBFT to reduce communication complexity, while the remaining nodes mainly undertake verification and supervision responsibilities. Simultaneously, a monitoring and punishment mechanism for low-reputation nodes is introduced to limit, downgrade, and isolate faulty and abnormal nodes. Experimental results show that this mechanism, while maintaining Byzantine fault tolerance, can effectively reduce consensus communication overhead and improve master node election stability and overall system performance.

Keywords: PBFT; Blockchain; Reputation score; Consensus algorithm; Node classification

Online publication: March 31, 2026

1. Introduction

Byzantine Fault Tolerance (BFT) consensus is the core mechanism for achieving deterministic finality and highly reliable replication in consortium blockchains and permissioned blockchains [1]. Classic PBFT achieves secure consistency by using a "master node ordering + multi-round voting confirmation" method, with a maximum tolerance of $f < n/3$ Byzantine nodes. It has advantages such as no mining required, low computing power consumption, and deterministic confirmation, and is therefore widely used in permissioned blockchains and

distributed systems. However, the original PBFT still faces three prominent problems in scaling up and complex adversarial environments as follows:

(1) The communication complexity increases significantly with the node scale, and broadcast interaction will cause bandwidth and latency bottlenecks in large-scale networks [2];

(2) The master node plays a key ordering and driving role in the protocol. If the master node fails or adopts a "smart Byzantine strategy" to degrade performance, it may lead to a decrease in throughput or even frequent view switching [3];

(3) When there are differences in node reliability and participation behavior, if there is a lack of effective node evaluation and governance mechanisms, low-quality or abnormal nodes will increase invalid voting, timeouts, and message propagation noise, reducing the overall stability of the system [4].

To alleviate the communication bottleneck of PBFT in large-scale networks, a large number of studies in recent years have introduced the ideas of grouping or layering. Jiang *et al.* proposed an improved PBFT consensus algorithm, which scores nodes and divides them into different functional levels, allowing only some high-scoring nodes to participate in the consensus process, thereby effectively reducing system communication overhead and improving throughput performance [5]. On this basis, CG-PBFT further introduces a credit grouping mechanism, which dynamically divides consensus nodes and non-consensus nodes according to the node credit value, significantly improving the performance and stability of PBFT in dynamic network environments [6]. In addition, for specific application scenarios, P-PBFT and other works combine the grouping mechanism with business requirements, reducing the number of consensus participating nodes to reduce latency and improve system availability [7].

The above studies show that grouping and layering are effective means to improve the scalability of PBFT, but its master node election and node governance mechanisms still need to be further improved. Another type of research focuses on introducing reputation or credit models into PBFT to improve node governance capabilities. Li *et al.* combined game theory with reputation models to dynamically adjust the reputation value of nodes based on their behavior in the consensus process, and used a reward and punishment mechanism to incentivize nodes to participate in consensus honestly, thereby improving the overall performance and security of the system [8]. In 2025, Deng *et al.* proposed DRAC-PBFT, which introduced a hierarchical dynamic reputation control model to achieve adaptive adjustment of node roles through reputation assessment and control, thereby reducing communication complexity and enhancing the robustness of the system to abnormal behavior [9]. Moreover, some studies have incorporated direct interaction, indirect recommendation and historical behavior into the reputation assessment process from the perspective of reputation synchronization and multidimensional credibility modeling, further improving the stability and security of PBFT in complex adversarial environments [10,11]. The master node selection strategy directly affects the performance and stability of PBFT consensus. In recent years, more and more studies have tried to use reputation or credit information to guide the election of master nodes. Hussain *et al.* proposed a reputation-based leader selection and voting mechanism to improve the reliability of master node selection through weighted voting and suppress the influence of low-reputation nodes on the election process [12]. Meanwhile, some studies have combined historical behavior with voting weights to optimize the execution efficiency and Byzantine resistance of PBFT through a comprehensive scoring method [13]. Although the above methods have improved the quality of master node election to some extent, there is still room for further optimization in terms of candidate pool limitation, voter set design, and master election and dynamic grouping coordination.

In this context, this paper proposes a reputation-based improved PBFT (RGS-PBFT): nodes are dynamically

divided into consensus groups, candidate node groups, and ordinary node groups according to their reputation; master node candidates are generated from the top 5% of the consensus group's reputation and elected by weighted voting between the ordinary group and the candidate group; and the grouping is adaptively adjusted through periodic reputation updates, while monitoring and limiting the authority of faulty/low-reputation nodes are implemented, thereby reducing the critical communication burden, improving master election stability, and system availability while maintaining the Byzantine fault tolerance capability of PBFT.

2. PBFT consensus algorithm

PBFT is a permissioned Byzantine fault-tolerant consensus algorithm designed to achieve consistent replication of replica states in a distributed system even with the presence of Byzantine faulty nodes. PBFT assumes the system contains n replica nodes, of which at most f are Byzantine nodes, satisfying $n \geq 3f + 1$. Under this assumption, PBFT can guarantee the system's consistency (safety) and liveness in a partially synchronous network model. In PBFT, nodes are organized by view, with each view consisting of a primary node and several backup nodes. The primary node is responsible for ordering client requests and driving the execution of the consensus protocol, while the backup nodes are responsible for verifying the primary node's behavior and participating in voting confirmation. When the primary node fails or experiences performance issues, the system elects a new primary node through a view change mechanism to resume the consensus process. The three-stage PBFT protocol is shown in **Figure 1**.

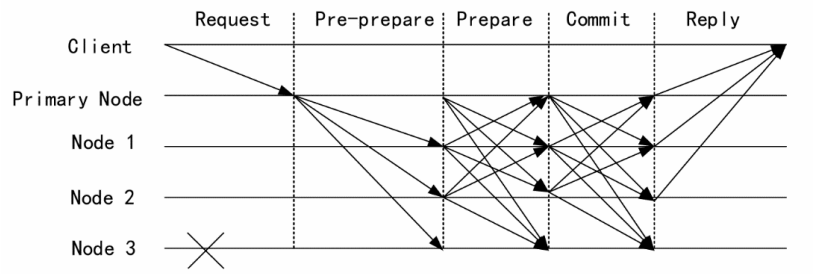

Figure 1. PBFT flowchart.

The consensus protocol of the PBFT consensus algorithm is mainly executed in the following stages sequentially:

2.1. Request stage

The client sends a request message containing specific operation content to the master node. The master node checks the validity of the received request, assigns a global sequence number, generates a message digest, and completes the signature. The processed request is then encapsulated as a pre-prepare message.

2.2. Pre-prepare stage

The master node broadcasts the pre-prepare message to all slave nodes. Upon receiving the message, the slave nodes verify the consistency and integrity of information such as the view number, sequence number, and message digest. If the verification is successful, the pre-prepare message is recorded in the local log.

2.3. Preparation stage

After completing the pre-prepare verification, the slave nodes broadcast a prepare message to other consensus nodes to indicate their acceptance of the request's ordering result. When a node collects $2f + 1$ prepare messages with the same sequence number from different nodes, it considers the request to have reached the prepared state under the current view and proceeds to the next stage.

2.4. Confirmation phase

Consensus nodes, including the master node, broadcast confirmation (Commit) messages to other nodes in the network. When a node receives $2f + 1$ confirmation messages with matching sequence numbers, it can determine that the execution order of the request has been finally confirmed by a majority of nodes in the system and can safely execute the operations contained in the request.

2.5. Response phase

After completing the request execution, the node returns the execution result to the client in the form of a response message. When the client collects at least $f + 1$ response messages with matching content, it can determine that the request has been successfully processed by the system.

3. RGS-PBFT

To enhance the scalability and robustness of PBFT in large-scale permissioned networks, this paper proposes an improved PBFT algorithm driven by reput ation scores (RGS-PBFT). This algorithm constructs a dynamic reputation score model based on node behavior data, and accordingly divides nodes into consensus groups, candidate groups, and ordinary groups. A closed-loop governance mechanism is formed through "layered consensus + weighted leader election + dynamic evolution + fault monitoring".

The system comprises N node sets $T = \{1,2,...,N\}$, tolerates a maximum of f Byzantine nodes, and satisfies the basic conditions of PBFT $N \geq 3f + 1$. The network uses a partially synchronous model, and message verifiability is ensured between nodes through digital signatures and hash digests. The algorithm operates on a view or fixed-batch (epoch) basis, denoted as $t = 0,1,2,\ldots$

3.1. Node reputation scoring model

3.1.1. Indicator definition

Within period t, the consensus participation and network behavior of each node i are statistically analyzed, and the following core indicators are constructed as follows:

(1) Verification of correctness

Used to reflect the quality of verification, its formula is shown in **Equation 1**.

$$A_i(t) = \frac{n_i^{correct}(t)}{n_i^{verify}(t)+\varepsilon} \tag{1}$$

Where n^{verify} is the number of times the node participated in verification, and $n^{correct}$ is the number of times it is consistent with "verifiable facts or majority consensus results".

(2) Voting participation rate

The formula used to reflect participation enthusiasm is shown in **Equation 2**.

$$V_i(t) = \frac{n_i^{vote}(t)}{n^{vote}{}_{total}(t)+\epsilon} \quad (2)$$

Where n^{vote} represents the number of times a node completes a vote as required by the protocol, and n^{vote_total} represents the total number of votes initiated by the system during this period.

(3) Online stability

The formula used to reflect availability is shown in **Equation 3**.

$$U_i(t) = \frac{T_i^{online}(t)}{T^{window}(t)} \quad (3)$$

(4) Response latency performance

It is used to reflect timeliness, and the smaller the value, the better. Its formula is shown in **Equation 4**.

$$L_i(t) = exp\left(-\frac{\bar{d}_i(t)}{d_0}\right) \quad (4)$$

Where d_i is the average response delay and d_0 is the scaling constant (which can be the network average or set empirically).

(5) Forwarding contribution

The formula used to measure the ability of the backbone of the transmission is shown in **Equation 5**.

$$F_i(t) = \frac{n_i^{forward}(t)}{max_{j\in N} n_j^{forward}(t)+\varepsilon} \quad (5)$$

3.1.2. Positive scores and penalties

The above indicators are linearly weighted to obtain the positive periodic score (normalized to 0–100), as shown in **Equation 6**.

$$S_i(t) = 100 \cdot (w_A A_i(t) + w_V V_i(t) + w_U U_i(t) + w_L L_i(t) + w_F F_i(t) \quad \text{in, } w_A + w_V + w_U + w_L + w_F = 1 \quad (6)$$

Meanwhile, tiered penalties are set for abnormal node behavior:

Provable malicious behavior: When a node engages in double signing, forgery, conflict voting, or digest tampering, a count of $m_i^{byz}(t)$ is applied, as shown in **Equation 7**.

$$P_i^{byz}(t) = \gamma_1 m_i^{byz}(t) \quad (7)$$

Timeout/Absence: When a node fails to send a message in a phase or times out, the count is multiplied by $m_i^{to}(t)$, as shown in **Equation 8**.

$$P_i^{to}(t) = \gamma_2 m_i^{to}(t) \quad (8)$$

Suspicious behavior: When a node exhibits abnormal frequency, deviates from the majority for an extended period, or exhibits abnormal forwarding behavior, the count is $m_i^{susp}(t)$, and the formula is shown in **Equation 9**.

$$P_i^{susp}(t) = \gamma_3 m_i^{susp}(t) \quad (9)$$

The total penalty is shown in **Equation 10**.

$$P_i(t) = P_i^{byz}(t) + P_i^{to}(t) + P_i^{susp}(t) \quad (10)$$

3.1.3. Reputation score update

To avoid relying on past high scores and to enhance sensitivity to new behaviors, a forgetting factor is introduced $\lambda \in [0,1)$.

$$R_i(t+1) = clip_{[0,100]}(\lambda R_i(t) + (1-\lambda)S_i(t) - P_i(t)) \quad (11)$$

Where $clip_{[0,100]}$ means to truncate the result to [0,100].

Where means to truncate the result to [0,100].

3.2. Dynamic grouping mechanism

3.2.1. Grouping principles

Nodes are managed hierarchically based on reputation scores, with a set ratio parameter α,β(α+β<1). Specifically: Consensus Group $G_C(t)$: Top-ranked nodes (α%); Candidate Group $G_K(t)$: Next-ranked nodes (β%); Ordinary Group $G_0(t)$: All other nodes.

$$G_C(t) = \{i | R_i(t) \geq Q_{1-\alpha}(t)\} \quad (12)$$

$$G_K(t) = \left\{i \middle| Q_{1-\alpha-\beta}(t) \leq R_i(t) < Q_{1-\alpha}(t)\right\} \quad (13)$$

$$G_O(t) = N, (G_C(t) \cup G_K(t)) \quad (14)$$

3.2.2. Hysteresis and steady-state strategies

To avoid frequent fluctuations in boundary nodes between groups, a hysteresis strategy can be adopted: a node needs to meet the group promotion condition for $h_{\uparrow}$ consecutive rounds; it needs to meet the group demotion condition for $h_{\downarrow}$ consecutive rounds, thereby improving system stability.

3.3. Master node election mechanism

3.3.1. Construction of candidate master node set

During the leader election phase of period t, the top 5% of the consensus groups by reputation score are selected to form a candidate set:

$$K = \max(1, \lceil 0.05 \cdot |G_C(t)| \rceil) \quad (15)$$

$$C(t) = TopK\ (G_C(t), R_i(t), K) \quad (16)$$

3.3.2. Voter aggregation and weighted voting

The voter pool consists of ordinary voters and candidates:

$$V(t) = G_O(t) \cup G_K(t) \quad (17)$$

To reduce the risk of low-reputation nodes manipulating elections, reputation-weighted voting is introduced. A weight is defined for voter $j \in V(t)$:

$$w_j(t) = \frac{R_j(t)}{\sum_{k \in V(t)} R_k(t) + \varepsilon} \tag{18}$$

The vote score for candidate node $i \in C(t)$ is:

$$Score\ (i,t) = \sum_{j \in V(t)} w_j(t) \cdot 1\{j \to i\} \tag{19}$$

The master node is determined as follows:

$$p(t) = argmax_{i \in C(t)} Score\ (i,t) \tag{20}$$

If a tie occurs, the following order will be used: higher $R_i(t)$ priority; if still tied, the node ID or the order from the previous round will be used for deterministic breaking.

3.4. RGS-PBFT process

After the master node $p(t)$ is elected, the consensus process executes the PBFT core phase within consensus group $G_C(t)$ to reduce critical communication complexity. Let $n_c = |G_C(t)|$ be the tolerable number of Byzantine nodes, $f_c = \left\lfloor \frac{n_c - 1}{3} \right\rfloor$, $Quorum = 2f_c + 1$ and the consensus protocol be as shown in **Figure 2**.

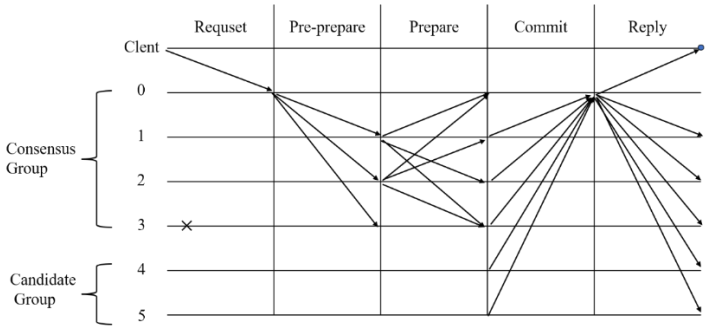

Figure 2. RGS-PBFT flowchart.

The consensus process is as follows:

3.4.1. Request phase

The client first sends a request message containing specific operation content to the master node selected in the current round. The master node is located within the consensus group and is responsible for driving the execution of the consensus protocol in this round. The client request is only sent to the master node and does not interact directly with other nodes.

3.4.2. Pre-prepare phase

After receiving the client request, the master node checks the validity of the request, assigns it a global sequence number, generates a request digest, and completes the signing operation. Subsequently, the master node broadcasts the encapsulated Pre-Prepare message to all nodes in the consensus group. After receiving the Pre-Prepare message, the consensus group nodes verify the consistency and integrity of the view number, sequence number, and message digest. Nodes that pass the verification record the pre-prepare message in their local logs, providing a consistent ordering basis for subsequent phases.

3.4.3. Preparation phase

After completing the pre-prepare verification, each normal node in the consensus group broadcasts a Prepare message to other consensus group nodes, indicating that it recognizes the execution sequence assigned by the master node for the request. As shown in the Prepare phase of **Figure 2**, Prepare messages are propagated only within the consensus group; candidate node groups do not participate in this phase. When a consensus node collects at least a certain number of Prepare messages with the same sequence number from different nodes, it considers the request to have reached the prepared state in the current view and proceeds to the next phase.

3.4.4. Confirmation phase

Consensus group nodes that have entered the prepared state continue to broadcast Commit messages to other consensus group nodes. As shown in **Figure 2**, messages in the Commit phase also generate intensive interactions only within the consensus group to complete the eventual consensus confirmation. When a node receives at least a certain number of consistent Commit messages, it can confirm that the execution order of the request has been ultimately recognized by a majority of consensus nodes in the system and safely execute the operations contained in the request. This design effectively limits the scale of critical communication while maintaining PBFT Byzantine fault tolerance.

3.4.5. Reply phase

After completing the request execution, the consensus group node returns the execution result to the client in the form of a Reply message. As shown in **Figure 2**, candidate node groups also receive execution results after the Commit phase for subsequent monitoring and reputation assessment, but their responses are not counted in the consensus quorum. A client can confirm that the request has been successfully executed by the system after receiving at least a certain number of consistent response messages.

4. Results and discussion

The improved PBFT consensus algorithm (RGS-PBFT) presented in this paper uses an Intel(R) Core(TM) i7-10875H processor, Ubuntu 18.04 operating system, and an RTX 2060 6G graphics card in its simulation device, and is written and implemented using Python 3.8. This section analyzes the time complexity, and then compares the algorithm with PBFT, RSPPBFT, and EPBFT, comparing the experimental differences in consensus latency and throughput to verify the effectiveness of the improvements presented in this paper [14–16].

4.1. Time complexity analysis

In the traditional PBFT algorithm, all nodes participate in the core consensus phases such as Pre-Prepare, Prepare, and Commit. The Prepare and Commit phases, in particular, require message broadcasting among all nodes, resulting in a single-round consensus time complexity of $O(N^2)$. This significantly limits system performance as the node size increases. In contrast, RGS-PBFT introduces a reputation-based grouping mechanism, strictly limiting the core consensus process of PBFT to consensus groups of size n_c. Candidate and ordinary node groups only assume supervisory and voting responsibilities and do not participate in message broadcasting during critical phases, thus reducing the dominant time complexity of the consensus protocol to $O(n_c^2)$. Since real-world systems typically have $n_c N$, RGS-PBFT effectively reduces the scale of message interaction and consensus latency while maintaining the original Byzantine fault tolerance, significantly improving the system's scalability and operational

efficiency in large-scale network environments.

4.2. Consensus delay

Consensus latency refers to the time elapsed from the submission of a request to its completion, and its expression is:

$$T_{delay} = T_{submit} - T_{authentication} \tag{21}$$

In the formula, T_{submit} is the consensus completion time and $T_{authentication}$ is the consensus start time. To ensure the integrity of the experiment, it was repeated 30 times and the average value was taken. The statistical consensus delay is shown in **Figure 3**.

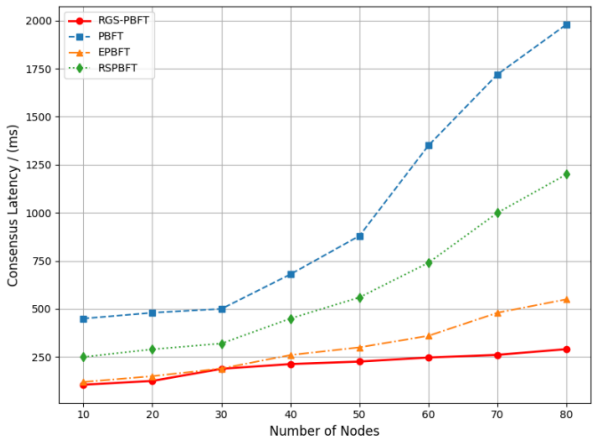

Figure 3. Consensus latency comparison.

Since each round of consensus requires nodes to send information to each other, the time complexity of RGS-PBFT proposed in this paper is $O(n_c^2)$ compared to the $O(N^2)$ time complexity of PBFT. Therefore, this paper divides consensus groups by reputation scores, reduces the number of participating consensus nodes, and isolates faulty nodes, which can effectively improve the efficiency of consensus.

4.3. Throughput

TPS stands for throughput. Higher throughput indicates a stronger transaction processing capacity of the system. Throughput is calculated as follows:

$$TPS = \frac{Transactions}{\Delta_t} \tag{22}$$

Where Δ_t represents the unit time, and *Transactions* represents the number of transactions completed within a given time.

To ensure experimental consistency, this experiment was repeated 30 times, and the average value was taken. The statistical throughput is shown in **Figure 4**.

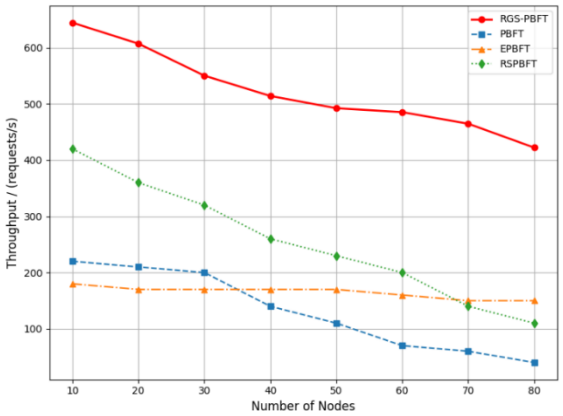

Figure 4. Throughput comparison.

Experimental results show that as the number of nodes increases, PBFT, EPBFT, RSPBFT, and RGS-PBFT all show a decreasing trend. However, the RGS-PBFT proposed in this paper optimizes the selection of master nodes and the consensus protocol, greatly improving efficiency. Furthermore, through reputation points, it can withstand more Byzantine nodes, preventing the system from being attacked and thus improving the system throughput.

5. Conclusion

This paper addresses the problems of high communication overhead, master node failure sensitivity, and difficulty in effectively managing node heterogeneity in large-scale networks caused by the traditional PBFT consensus algorithm. A new group consensus algorithm, RGS-PBFT, based on reputation score-driven consensus, is proposed. This algorithm dynamically groups nodes by constructing a multi-dimensional reputation score model, confining the core consensus process of PBFT to the consensus group. It also combines reputation-weighted master node election and node supervision mechanisms, effectively reducing the communication complexity of the consensus phase while maintaining the original Byzantine fault tolerance characteristics. Experimental results show that the proposed algorithm has better performance and scalability than the traditional PBFT in terms of consensus latency and system throughput.

Disclosure statement

The author declares no conflict of interest.

References

[1] Wu X, Wang Z, Li X, et al., 2025, DBPBFT: A Hierarchical PBFT Consensus algorithm with dual blockchain for IoT. Future Generation Computer Systems, 2025(162): 107429.

[2] Liu X, Fan X, Niu B, et al., 2025, 5G-Practical Byzantine Fault Tolerance: An Improved PBFT Consensus Algorithm for the 5G Network. Information, 16(3): 202.

[3] Li W, Feng C, Zhang L, et al., 2020, A Scalable Multi-Layer PBFT Consensus for Blockchain. IEEE Transactions on

Parallel and Distributed Systems, 32(5): 1146–1160.

[4] Zhao P, Qiang G, Fan Y, et al., 2025, YOLO11-FGA: Express Package Quality Detection Based on Improved YOLO11. Information, 16(12): 1021.

[5] Jiang W, Wu X, Song M, et al., 2023, Improved PBFT Algorithm Based on Comprehensive Evaluation Model. Applied Sciences, 13(2): 1117.

[6] Liu J, Deng X, Li W, et al., 2024, CG-PBFT: An Efficient PBFT Algorithm Based on Credit Grouping. Journal of Cloud Computing, 13(1): 74.

[7] Liu S, Zhang R, Liu C, et al., 2023, P-PBFT: An Improved Blockchain Algorithm to Support Large-Scale Pharmaceutical Traceability. Computers in Biology and Medicine, 2023(154): 106590.

[8] Li Z, Wang J, Li Y, 2025, An Improved PBFT Consensus Algorithm Based on Reputation and Gaming: An Improved PBFT Consensus Algorithm. Journal of Supercomputing, 81(1).

[9] Deng X, Zhang L, Zou Y, et al., 2025, DRAC-PBFT: PBFT Consensus Optimisation Algorithm Based on Dynamic Reputation and Adaptive Clustering. IET Blockchain, 5(1): e70025.

[10] Wang Y, Dou J, Yang P, et al., 2025, Improved PBFT Algorithm Based on Reputations Synchronization and Group Consensus. International Journal of High Speed Electronics and Systems, 34(3): 2440103.

[11] Ding J, Wu X, Tian J, et al., 2025, RE-BPFT: An Improved PBFT Consensus Algorithm for Consortium Blockchain Based on Node Credibility and ID3-Based Classification. Applied Sciences, 15(13): 7591.

[12] Hussain M, Mehmood A, Khan M, et al., 2025, Reputation-Based Leader Selection Consensus Algorithm with Rewards for Blockchain Technology. Computers, 14(1): 20.

[13] Wang Y, Qiao Y, Ma J, et al., 2025, An Improved PBFT Consensus Algorithm for Blockchain Systems, Proceedings of the 9th International Conference on Electronic Information Technology and Computer Engineering, 715–719.

[14] Zhang Q, Su J, Ma Z, et al., 2021, Blockchain Model Testing and Implementation Based on Improved PFBT Consensus. 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). IEEE, 2021(2): 1010–1015.

[15] Chen P, Chen Y, Wang X, et al., 2024, A High-Capacity Slicing PBFT Protocol Based on Reputation Evaluation Model. Wireless Networks, 30(9): 7469–7482.

[16] Li Y, Wang Z, Fan J, et al., 2019, An Extensible Consensus Algorithm Based on PBFT. 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). IEEE, 17–23.