Journal of Computer Science and Digital Technology, 2026, 2(1), 21-36
https://doi.org/10.71204/pyppdf95

RF-PBFT is an Improved PBFT Consensus Algorithm Based on the Random Forest Algorithm

Yingchen Xu [1,*], Ye Tian [1], Ying Jing [1], Fujiang Yuan [1]

[1] School of Computer Science and Technology, Taiyuan Normal University, Shanxi 030619, China

*** Correspondence:**

Yingchen Xu

274513634@qq.com

Received: 12 February 2026 / Accepted: 11 March 2026 / Published online: 12 March 2026

Abstract

The Practical Byzantine Fault-Tolerant (PBFT) consensus algorithm faces challenges such as random master node selection, high communication overhead, and a lack of incentive and penalty mechanisms, which undermine its efficiency and security in large-scale network environments. To address these issues, this paper proposes an improved PBFT consensus algorithm, RF-PBFT, which integrates a random forest-based node grouping mechanism with a dynamic reputation system. The algorithm leverages key behavioral data of nodes during the consensus process to train a random forest model, partitioning nodes into a consensus set and a candidate set. Nodes with superior predictive performance are selected to participate in the consensus process, thereby reducing redundant communication overhead. Furthermore, a differentiated reputation mechanism is established for the consensus group and the candidate group to encourage positive behavior and deter malicious actions. The master node is elected through a combination of prediction results and voting, enhancing system reliability and security. Simulation results demonstrate that, compared with traditional PBFT and other state-of-the-art variants, RF-PBFT significantly reduces communication overhead, decreases consensus latency, and improves throughput, validating its effectiveness in multi-node blockchain systems.

Keywords: Blockchain; PBFT; Random Forest Algorithm; Reputation Mechanism

1. Introduction

With the continuous iteration and innovation of digital technologies, cutting-edge technologies such as artificial intelligence, big data, and the Internet of Things are profoundly changing the industrial landscape and social structure. Against this backdrop, blockchain technology, due to its core characteristics such as decentralization, traceability, and tamper-proof nature, is gradually becoming an important pillar supporting the next generation of information infrastructure. Blockchain not only shows broad application prospects in fields such as finance, supply chain

management, healthcare, and intelligent manufacturing, but its underlying consensus algorithm is also considered crucial for ensuring system security and reliability (Veronese et al., 2009).

Consensus algorithms play a crucial role in coordinating distributed nodes to achieve a unified ledger state, directly impacting the performance, security, and scalability of a blockchain system. Currently, common consensus algorithms include Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Raft, and Practical Byzantine Fault Tolerance (PBFT) (Vedant et al., 2022; Castro et al., 1999). Different algorithms present varying trade-offs in efficiency, security, and decentralization: PoW ensures security through hash calculations but suffers from high energy consumption and slow transaction confirmation speeds; PoS improves energy efficiency through stake allocation but carries the risk of stake concentration; DPoS increases transaction processing speed but is prone to centralization; Raft boasts low communication complexity and high efficiency but is highly dependent on network reliability and lacks fault tolerance (Li & Li, 2025; Madigan et al., 2012). In contrast, the PBFT algorithm achieves a relative balance between decentralization and performance. It can ensure consensus in a system with a certain number of Byzantine nodes without requiring massive computing resources, and is therefore widely used in consortium blockchains. PBFT's advantages lie primarily in its efficiency, low energy consumption, and strong Byzantine fault tolerance, making it particularly suitable for consortium and private blockchain environments with a limited number of nodes and relatively clear trust relationships. However, the PBFT algorithm faces several challenges, specifically in the following aspects:

(1) In large-scale network environments, the PBFT algorithm suffers a sharp decline in system throughput as the number of nodes increases dramatically, making it difficult to achieve efficient consensus processes in multi-node environments.

(2) The master node selection method is simplistic, typically determined by view number. This increases the probability of a malicious node being elected, severely impacting consensus efficiency.

(3) The lack of reward and punishment mechanisms reduces node participation in consensus, making it difficult to curb malicious behavior and compromising system stability and security.

To address the high complexity, random master node selection, and lack of reward and punishment mechanisms inherent in the PBFT consensus algorithm in large-scale blockchain networks, this paper proposes a PBFT consensus algorithm based on random forest grouping (FR-PBFT). First, to address the high communication overhead of PBFT in large-scale network scenarios, a random forest grouping algorithm is used to predict future node behavior. Based on the prediction results, nodes are divided into consensus and candidate node sets, selecting a subset of reliable nodes to participate in consensus and reducing unnecessary communication overhead. Secondly, to address the lack of a reward and punishment mechanism in PBFT, different reward and punishment mechanisms were set for the consensus node set and the candidate node set to ensure the enthusiasm of nodes to participate in consensus and voting. Finally, to address the issue of random selection of master nodes, a method of prediction results plus voting was adopted to elect reliable master nodes.

2. Related research

In recent years, the PBFT consensus algorithm has been extensively studied. Regarding scalability, many studies have proposed layered architectures and communication optimization schemes. Gueta et al.(2018)proposed SBFT (Scalable Byzantine Fault Tolerance) technology, which, with the help of collectors, threshold signatures and optimistic fast paths, achieves double the throughput and improves latency compared to traditional PBFT. In addition, Sukhwani et al. (2017)]analyzed the consensus process in permissioned blockchains through stochastic modeling, and confirmed the communication bottleneck that may occur when the node scale increases, emphasizing the necessity of PBFT optimization in large networks. HotStuff achieves linear communication complexity and responsiveness during network synchronization (Wu et al., 2025; Yin er al., 2019), making the consensus speed match the actual network latency; BFT-SMART improves system efficiency by improving communication strategies (Bessani et al., 2014). Li et al. (2020) designed a multi-layer PBFT model, which compresses the communication volume to a linear level through tree-like hierarchical grouping, but needs to balance cross-layer latency and fault tolerance performance. Na et al. (2022) proposed the DPNPBFT algorithm, which adopts a dual-master node power-sharing mechanism to improve the fault tolerance of the algorithm and is suitable for large-scale low-power scenarios such as the Internet of Things. Zhong et al. (2023) proposed an improved PBFT algorithm for secure knowledge transactions, ST-PBFT, which uses a grouping method based on consistent hashing to improve communication efficiency.

3. Overview of PBFT Consensus Algorithm and Random Forest Algorithm

3.1. PBFT Consensus Algorithm

The PBFT algorithm requires no staked equity or consumes competitive resources, resulting in low costs for malicious nodes. Its three-phase voting mechanism effectively eliminates interference from malicious nodes in the consensus process, possessing fault tolerance capabilities up to one-third of the total number of nodes. However, the current PBFT algorithm still has many areas requiring improvement, such as high communication overhead and strong randomness in master node selection, leading to low consensus efficiency. Therefore, most researchers have conducted in-depth research on master node selection methods and consensus consistency, striving to improve the consensus efficiency of the PBFT algorithm. In practical applications, such as transaction settlement in the financial sector and information sharing in supply chain management, there is an urgent need for efficient and reliable consensus algorithms. The PBFT algorithm has certain application potential in these scenarios due to its inherent characteristics, but existing problems severely restrict its effectiveness. Therefore, in-depth analysis of the PBFT algorithm and the proposal of practical and effective improvement strategies are of significant practical importance for further exploring the potential of blockchain technology and expanding its application boundaries. This is precisely the starting point and core direction of this paper. The consensus protocol execution process of the PBFT consensus algorithm is as follows:

The consensus protocol is the core of the PBFT (Practical Byzantine Fault Tolerance)

algorithm. In a traditional PBFT consensus network, nodes are mainly divided into three categories: client nodes, master nodes, and replica nodes (slave nodes). Client nodes are responsible for initiating consensus requests and receiving the final consensus results; master nodes receive client requests, number and sort them, and distribute the processed messages to replica nodes after signing; replica nodes are responsible for verifying the received consensus messages and determining the consistency of the consensus results. When a master node fails or malfunctions, the system initiates a view switching mechanism to elect a new master node within a specified time to continue the subsequent consensus process.

The execution process of the PBFT algorithm can be divided into five main stages: Request, Pre-prepare, Prepare, Commit, and Reply. The overall process is shown in Figure 1, and is detailed below:

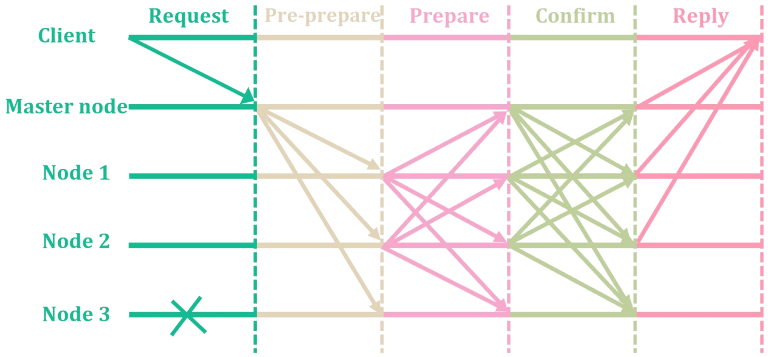

Figure 1. PBFT Algorithm Consensus Process

(1) Request Phase: The client sends a consensus request message to the master node, in the format <Request, m, t, c,σ_c>, where m represents the request content, t is the timestamp, c is the client identifier, and σ_c is the client signature.

(2) Pre-prepare Phase: After receiving the request, the master node numbers and sorts it, then generates a pre-prepare message <<Pre-prepare, v, n, D(m),σ_p>, m>, and sends it to all replica nodes. Here, v is the current view number, n is the message sequence number, D(m) is the message digest, and σ_p is the master node signature.

(3) Prepare Phase: After receiving the pre-prepare message, the replica nodes verify its validity and generate a prepare message <Prepare, v, n, D(m), b_i, σ_b> for broadcast, where b_i is the replica node identifier, and σ_b is the node's signature. In this phase, each replica node compares its generated preparation message with those sent by other replica nodes. When a node receives at least 2f+1 consistent preparation messages, it can proceed to the next phase. This phase helps screen for Byzantine malicious nodes in the network.

(4) Commit Phase: After the preparation conditions are met, each node generates a commit message <Commit, v, n, D(m), b_i, σ_b> and broadcasts it. When a node receives at least 2f+1 commit messages consistent with its own, it indicates that consensus has been largely reached, and it can proceed to the response phase.

(5) Reply (Response Phase): After confirmation in the commit phase, the node sends a response

message <Reply, v, n, σ_b, r> to the client, where r represents the result after executing the request. The client considers its request to have been successfully completed when it receives at least f+1 consistent response messages.

3.2. Random Forest Algorithm

Random Forest is a machine learning algorithm based on the Bagging ensemble strategy. Its core principle is to rely on multiple decorrelated decision trees for collaborative prediction. By leveraging a dual randomness mechanism, it overcomes the limitations of single decision trees and is now widely used in tasks such as classification and regression. The training process of Random Forest is shown in Figure 2.

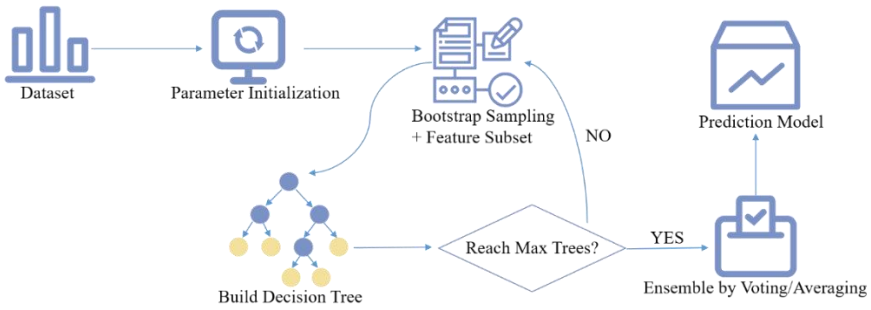

Figure 2. Random Forest Training Process

The training process of the Random Forest algorithm, and the specific implementation of each stage, are as follows:

(1) Input Dataset and Parameter Initialization

Random forest training first imports a dataset D containing sample feature vectors xi and labels yi. Then, configure the core parameters: number of decision trees T to balance complexity (few trees easily lead to underfitting, more trees increase cost); tree structure parameters (such as maximum depth d, minimum number of samples per leaf node) to control tree growth and avoid overfitting of a single tree; the number of features per tree m is set to $\lfloor \sqrt{d_{\text{total}}} \rfloor$ (d_{total} is the total number of features), and feature randomization enhances the independence between trees, allowing each tree to learn different feature subsets and adapt to multi-dimensional feature learning.

(2) Bootstrap Sample Sampling (Constructing Training Sets for Multiple Trees)

For the t-th tree (t = 1,...,T), n samples are drawn with replacement from D to generate Dt. Sampling characteristics: data diversity (different Dts have different characteristics, approximately 36.8% are out-of-bag samples used for OOB validation), and unbiasedness (sampling with replacement ensures that the training set distribution is consistent with the original data). This step provides differentiated data for each tree, constructs diverse base learners, supports subsequent ensemble learning (voting/averaging), and improves the model's generalization and fitting capabilities.

(3) Building a Single Decision Tree

In the building a single decision tree stage, a single decision tree is trained using a "sample subset Dt + feature subset Ft". The core logic revolves around greedy splitting and recursive termination. When performing greedy splitting, starting from the root node, the features in the feature subset Ft and the possible splitting thresholds are traversed, and the optimal splitting method is determined based on the task type. For classification tasks, the node purity is evaluated by calculating the Gini index, using the following formula:

$$\text{Gini}(S) = 1 - \sum_{c=1}^{C} \left(\frac{|S_c|}{|S|} \right)^2$$

(Where S is the current node sample set, C is the total number of classes, and S_c is the subset of samples belonging to class c in S) Select the split that minimizes the sum of the Gini coefficients of the child nodes to reduce sample uncertainty; for regression tasks, calculate the Mean Squared Error (MSE) to measure the dispersion of node samples, using the formula:

$$\text{MSE}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y})^2$$

(where is the mean of the sample labels in S) Select the split that minimizes the MSE of the child nodes to improve the consistency of sample prediction. When the tree depth reaches the preset maximum depth max_depth, or the number of leaf node samples is less than min_samples_leaf, the recursion is terminated. In the classification task, the mode of the sample labels output by the leaf node (the category that appears most frequently) is used as the predicted value of the node.

(4) Determine if the maximum number of trees has been reached (Reach Max Trees).

Check if the number of decision trees built has reached the T set in the parameter initialization phase: if the number has not been reached, return to the "Bootstrap Sampling + Feature Subset Selection" step to continue building the next decision tree; if the number has been reached, enter the "Multi-Tree Integration" step to improve model performance by utilizing the diversity of multiple trees.

(5) Multi-tree ensemble (Ensemble by Voting/Averaging)

In the multi-tree ensemble stage, the prediction results of T decision trees need to be summarized and integrated according to the task type: For a regression task, let the predicted class of sample x by the t-th tree be: , The prediction results of all trees are summarized, and the class with the most votes is taken as the prediction result, which can be described by the formula:

$$H(x) = \arg\max_{c \in \{1,\ldots,C\}} \sum_{t=1}^{T} \text{ú}(h_t(x) = c)$$

(Where I is an indicator function, taking the value 1 when the condition is met and 0 otherwise), multi-tree results are fused through majority voting to reduce the impact of single-tree misjudgment; for regression tasks, let the predicted value of the t-th tree for sample x be

$h_t(x) \in R$,The arithmetic mean of all tree predictions is taken as the final result, and the consensus is:

$$H(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

Mean aggregation is used to smooth the differences in multi-tree predictions and improve the stability of the results.

(6) Output Prediction Model

After completing the above steps, a random forest model is obtained, which is an ensemble of T decision trees through a "voting/averaging" process. When the model is applied, a new sample is input, each decision tree makes an independent prediction, and finally, the ensemble strategy outputs the final classification label or regression value, achieving effective prediction of unknown data.

4. RF-PBFT Algorithm Design

4.1. System Model

To address the shortcomings of the traditional PBFT consensus algorithm in node management and consensus efficiency, this system constructs a closed-loop architecture of "node grouping - master node election - consensus execution - reputation update". The core is to optimize the node grouping strategy through the random forest algorithm and combine it with the reputation mechanism to improve consensus security and efficiency. The specific system model structure is shown in figure 3.

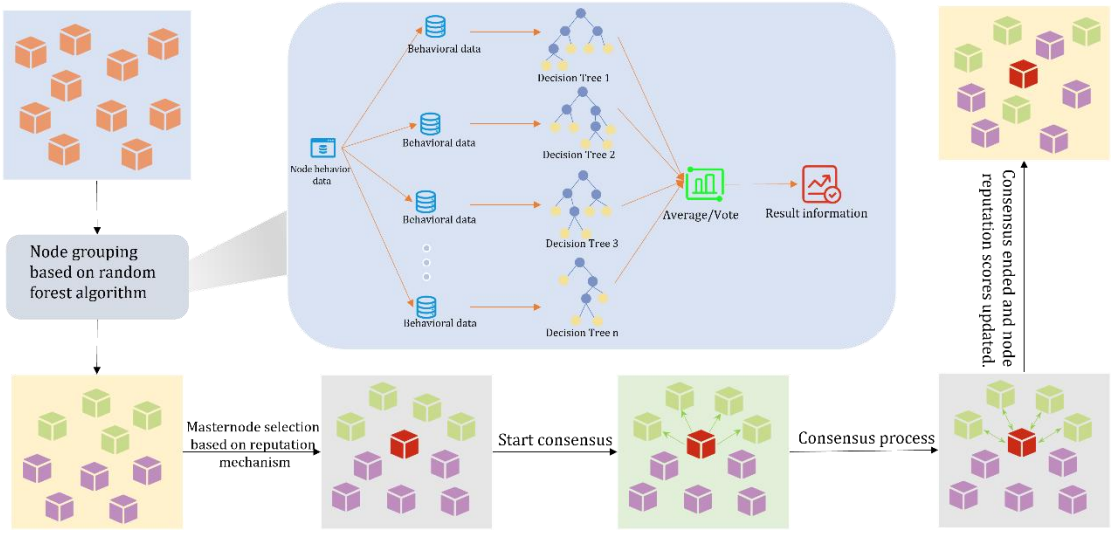

Figure 3. RF-PBFT system model diagram

In the RF-PBFT system model, the system takes historical node behavior data (including reputation records, interaction features, etc.) as input, deploys a random forest algorithm to construct an ensemble decision tree model, generates multiple training subsets through bootstrap sampling, trains decision trees independently in each subset, and integrates them by voting,

outputting a highly reliable consensus node group (participating in the core consensus) and a candidate node group for backup observation. On this basis, the master node election introduces a dynamic reputation mechanism, which integrates the grouping results (random forest predicted labels) with real-time behavior data (such as consensus response efficiency and consensus completion rate), selects nodes with high reputation values and stable behavior to be elected as the master node for this round of consensus. After consensus is completed, the system quantifies the node behavior performance (such as voting enthusiasm and consensus completion rate) and dynamically updates the reputation value (deduction for malicious behavior and reward for compliant behavior), and feeds the updated data back to the node grouping module as new training samples, forming a closed-loop iteration of "node grouping - master node election - consensus execution - reputation update", enabling the system to adapt to the dynamic changes in node behavior and continuously optimize the accuracy of the grouping strategy.

4.2. Reputation Mechanism

The reputation mechanism is a key indicator for measuring node behavior and a necessary condition for the grouping algorithm. This scheme distinguishes between the reputation mechanisms of consensus nodes and candidate nodes. Before the formal consensus process begins, all nodes have a default reputation score of 0. First, a three-round pre-consensus process is performed on the nodes. Based on the pre-consensus process, the node behavior data is obtained. After the RF random forest grouping model obtains the node behavior data, it predicts the future behavior of each node and assigns a corresponding group label. For example, the top 50% of nodes in terms of predicted behavior are labeled as "Consensus Group." The remaining nodes are in the candidate group. After grouping, the master node for this round is selected by voting, and the pre-consensus work is completed.

At the start of the consensus process, the performance of all nodes is recorded by the monitoring nodes. After each round of consensus, the node is assigned a reputation score based on its group's reputation mechanism. If a consensus node's reputation score is in the bottom half, it is designated as a candidate node and added to the candidate node set. At this point, candidate nodes are sorted according to their reputation scores, and the corresponding number of nodes are converted into consensus nodes. When a candidate node becomes a consensus node, its score is halved.

4.2.1. Consensus Node Reputation Mechanism

In this scheme, nodes are divided into consensus nodes and candidate nodes. These two types of nodes play different roles in the consensus process, hence their reputation score formulas also differ. The formula for measuring the reputation score of consensus node i is as follows:

$$\mathrm{Sc}_i = \alpha \cdot CE_i + \beta \cdot CR_i - \gamma \cdot E_{punish}^{-1} \cdot P_i(x)$$

Where Sc_i represents the overall evaluation score of consensus node i, with higher values indicating better performance. CE_i represents the node's consensus efficiency, reflecting its relative performance per unit time. CR_i represents the consensus completion rate, i.e., the proportion of nodes successfully participating in consensus. $\mathrm{E}_{\mathrm{punish}}$ is the anomaly penalty

coefficient, reflecting the negative impact of abnormal node behavior on consensus and imposing different levels of penalty based on the magnitude of the impact. The components of each formula will be introduced below.

(1) Consensus efficiency: Represents the relative rate at which a node reaches consensus per unit of time.

$$CE_i = \frac{C_i / T_i}{\frac{1}{N}\sum_{j=1}^{N}(C_j / T_j)}$$

Where C_i is the number of times node i successfully participates in consensus. T_i is the node's online time. N is the total number of consensus nodes in the system. The normalized ratio is expressed as: >1 for values above average, <1 for values below average.

(2) Consensus Completion Rate: Represents the success rate of a node in initiating or participating in consensus.

$$CR_{\mathrm{i}} = \frac{C_{\mathrm{i}}}{P_{\mathrm{i}}}$$

Where P_i is the total number of consensus rounds participated in by node i, and C_i is the number of consensus rounds successfully completed by node i.

(3) Abnormal penalty coefficient

$$E_{punish} = 1 - \frac{K_i}{P_i}$$

Here, E_{punish} is the anomaly penalty coefficient, K_i is the number of times node i triggers anomalies (such as duplicate proposals or data errors), and P_i represents the total number of consensus rounds participated in by node i. The logic of this coefficient is: the more anomalies, the heavier the penalty; if there are no anomalies, $E_{punish} = 1$, and the more anomalies, the closer it gets to 0.

(4) Consensus Penalty Scoring

$$P_i(x) = P_0 + \alpha \cdot \sqrt[3]{\frac{x}{\max_x}}$$

The core concept of this formula is that the penalty increases with the number of failures. P_0 is the base penalty, which is set to 0 in this scheme to avoid deducting points even when there are no failures. α is the penalty coefficient, which can be adjusted according to the scenario. X represents the cumulative number of times node i has failed to participate in consensus. Max_x represents the maximum cumulative number of times all nodes in the network have failed to successfully participate in consensus; the formula has undergone dynamic normalization to avoid extreme values.

4.2.2. Candidate Node Reputation Mechanism

(1) Response rate: The willingness and stability of a node to respond when requested for assistance by the system, representing the node's enthusiasm for participating in consensus.

$$R_{\mathrm{i}} = \frac{R_{\mathrm{i}}^{\mathrm{response}}}{R_{\mathrm{i}}^{\mathrm{request}}}$$

Where R_i is the node's response rate.

(2) Voting enthusiasm

When a candidate node casts its vote in support of the master node in each round of selection, it is considered to have successfully voted and will be awarded a certain reputation score. Otherwise, a certain reputation score will be deducted as a penalty.

$$CD_{i,k} = CD_{i,k-1} + UD; UD = \begin{cases} d, Vote\ successful \\ -2d, Vote\ failed \end{cases}$$

Where $CD_{i,k}$ and $CD_{i,k-1}$ are the reputation scores updated by node i after the consensus of the current round and the previous round, respectively. UD is the candidate node reputation score update factor. When a node successfully participates in voting in this round, it is given a certain reputation score d as a reward. When a node does not vote in this round, it is penalized by deducting 2d reputation scores. The default value of d in this scheme is 0.5.

4.3. Node Grouping and Master Node Selection

In the consensus process of the RF-PBFT algorithm, a detection node is first set up. This node does not participate in any consensus process; it only detects the behavior of other nodes. After the client sends a request to the master node, the behavior detection node in the system tracks the interaction behavior of nodes in the consensus process in real time (such as voting consistency, message response timeliness, etc.), and inputs the detected behavior data into the random forest prediction model after each round of consensus. The model has a high accuracy in predicting the future behavior of nodes after previous training. Based on the model prediction results, the top 50% of nodes can be included in the consensus node set. At this time, a node needs to be selected from theθconsensus nodes to serve as the master node for this round of consensus. However, relying solely on reputation score to select the master node is too simplistic and can easily lead to issues such as centralization and reduced system security. Therefore, this scheme improves the selection of the master node. After the node grouping is completed, all nodes vote for the consensus node, and the master node is elected according to the number of votes. The number of votes for node i is calculated as follows:

$$V_i = \sum_{y=1}^{N} S_y; S_y = \begin{cases} \theta, if\ node\ y\ supports\ node\ i \\ \beta, if\ node\ y\ does\ not\ support\ node\ i \end{cases}$$

In this paper, θ and β are set to 1 and -1, respectively.

4.4. Consistency Process

This paper's solution groups nodes based on behavioral data predictions, resulting in different permissions for each group. Nodes in the consensus node group participate in the consensus process, while nodes in the candidate node group passively receive the consensus message. This transforms the traditional PBFT's "all nodes participate in consensus" into "only reliable nodes participate, with the remaining nodes only responsible for message propagation," while simplifying the commit phase and avoiding unnecessary communication overhead. The improved consensus flowchart is shown in Figure 4.

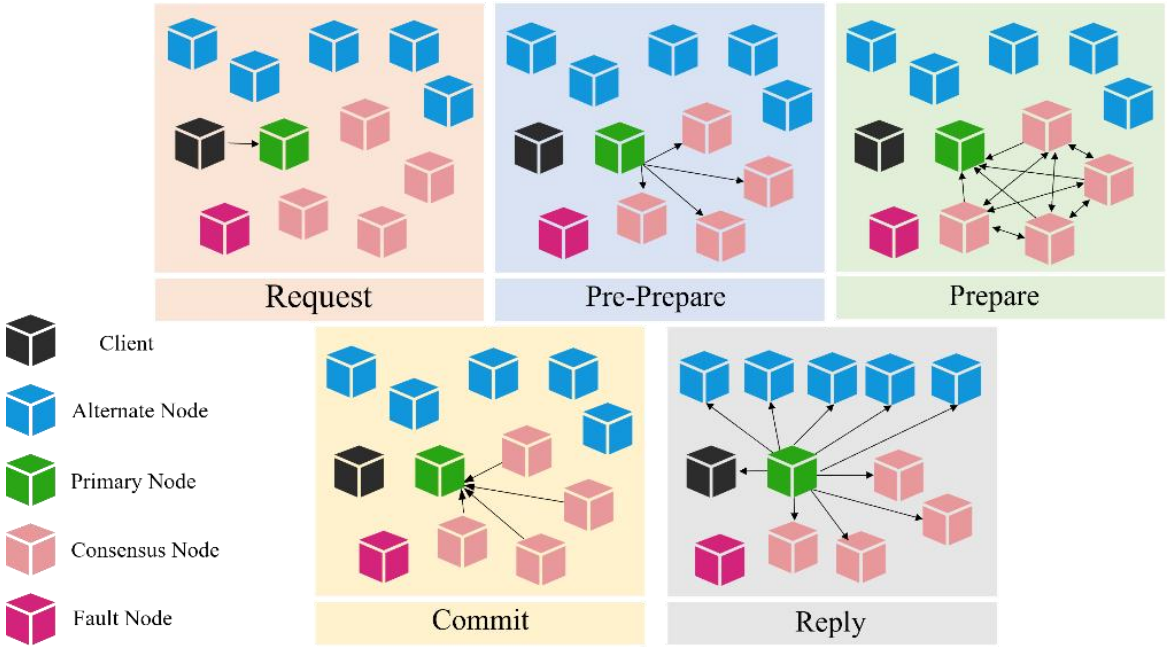

Figure 4. Consensus Process Improvement

5. Experimental Results and Analysis

The simulation experiment of the improved algorithm proposed in this paper used a 13th Gen Intel® Core™ i9-13900K 3.00GHz processor, Windows 11 operating system, 128GB of RAM, and Python 3.8 for code writing and implementation.

5.1. Classification Aaccuracy

This experiment compares the classification performance of three classification algorithms—Random Forest, Decision Tree, and Naive Bayes—based on a sample dataset of 1000 records. Accuracy and F1 score were used as evaluation metrics, and the results are shown in Table 1. The data shows that the Random Forest algorithm performs best in both metrics (Accuracy = 0.955, F1 Score = 0.957), with significantly higher classification accuracy than Decision Tree and Naive Bayes. The experiment demonstrates that the proposed grouping algorithm can better reduce the negative impact of malicious nodes on system consensus, thus improving system consensus efficiency and security.

Table 1. Performance Comparison Experiment of Classification Algorithms

Algorithm	Accuracy	F1 Score
Decision Tree	0.931	0.936
Naive Bayes	0.852	0.855
Random Forest Algorithm	0.955	0.957

5.2. Communication Overhead

Communication overhead refers to the total number of messages exchanged between nodes during the consensus process, and is an important indicator for judging the efficiency of a consensus algorithm.

Figure 5 compares the communication overhead of RF-PBFT, APBFT, and traditional PBFT. As shown in the figure, the amount of interactive messages increases with the number of nodes, but PBFT's increase is significantly greater than the other two, while RF-PBFT and APBFT's increases are relatively gradual. Furthermore, RF-PBFT's communication overhead is consistently significantly lower than that of similar algorithms. Therefore, RF-PBFT demonstrates a significant advantage in reducing network communication burden and is more suitable for blockchain networks composed of large numbers of nodes.

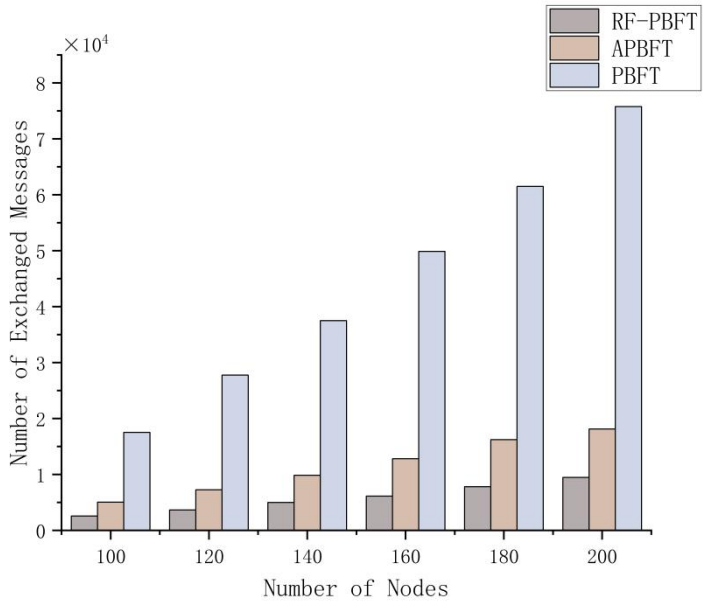

Figure 5. Comparison of communication overhead

5.3. Consensus Delay

Consensus latency is a crucial metric for measuring system performance, directly impacting system efficiency and user experience. Consensus latency refers to the time elapsed from when a client sends a request to when that request is fulfilled. Lower consensus latency among nodes in the system results in faster consensus being reached. The calculation of consensus latency is

shown in the following formula.

$$T = T_F - T_R$$

Where: T_F represents the time when the client receives f+1 identical reply messages, at which point the client's request has been successfully executed. T_R represents the time when the client sends the request to the master node.

With 100, 120, 140, 160, 180, and 200 nodes, 50 repeated experiments were conducted for each, and the average value was calculated. The final results are shown in Figure 6. When the number of nodes for the three algorithms is the same, the consensus latency of RF-PBFT and APBFT is significantly lower than that of traditional PBFT, and the advantage of this scheme is even more obvious compared to APBFT. It reduces some unnecessary communication overhead and demonstrates better consensus efficiency (figure 6).

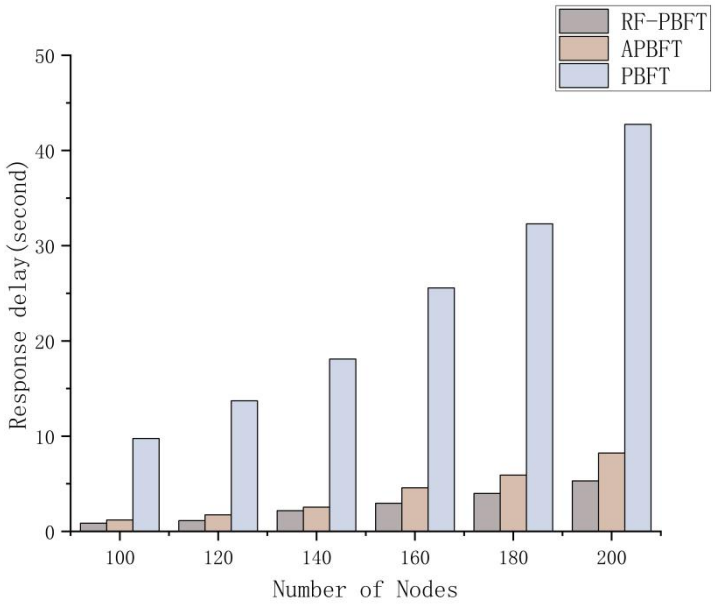

Figure 6. Consensus latency comparison

5.4. Throughput

Throughput (TPS), as a core indicator for measuring blockchain transaction processing capacity, reflects the number of transactions that complete consensus per unit of time and directly reflects the system's efficiency. The results of this latency test are shown in Figure 7: As the number of nodes increases, the TPS of RF-PBFT, APBFT, and traditional PBFT all show a downward trend. At the same node scale, RF-PBFT's transaction processing capacity advantage is particularly prominent: with 100 nodes, its TPS far exceeds that of PBFT; even when the number of nodes increases to 200, RF-PBFT still maintains a high throughput level.

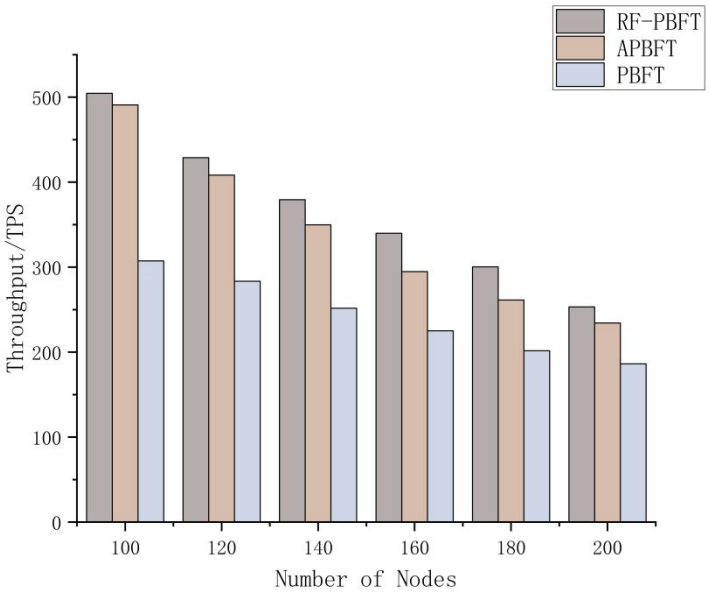

Figure 7. Throughput Comparison

5.5. Experiment Summary

This study focuses on the RF-PBFT consensus algorithm, conducting experiments on classification accuracy, communication overhead, consensus latency, and throughput. Comparison of grouping accuracy with random forest, decision tree, and Naive Bayes algorithms shows that the proposed RF-PBFT algorithm improves the reliability and security of system grouping and master node selection. In a multi-node environment of 100-200 nodes, this scheme groups nodes based on their behavioral data and introduces a reputation mechanism, ensuring reliable nodes participate in consensus, thus avoiding unnecessary communication overhead and improving system consensus efficiency. Simulation experiments demonstrate that the RF-PBFT scheme has significant advantages over PBFT and APBFT in multi-node scenarios in terms of communication overhead, consensus latency, and throughput.

6. Conclusion

This study proposes an improved PBFT consensus algorithm, FR-PBFT, which integrates random forest-based node grouping and a dual reputation mechanism to enhance consensus efficiency and security. The algorithm utilizes machine learning to predict node behavior and dynamically group nodes, effectively reducing communication complexity and mitigating the impact of malicious nodes. The introduction of the reputation system further incentivizes active node participation and effectively curbs malicious behavior. Experimental evaluations at different node scales demonstrate that FR-PBFT outperforms traditional PBFT and APBFT in terms of communication overhead, consensus latency, and throughput. This method provides a scalable and adaptive solution for consortium blockchains and private blockchains, making a theoretical and practical contribution to the field of distributed consensus. Future work will focus on further optimizing the real-time performance of this model and exploring its applicability in more dynamic and heterogeneous network environments.

Author Contributions:

All authors have read and agreed to the published version of the manuscript.

Funding:

This research received no external funding.

Institutional Review Board Statement:

Not applicable.

Informed Consent Statement:

Not applicable.

Data Availability Statement:

Not applicable.

Conflict of Interest:

The authors declare no conflict of interest.

References

Bessani, A. N., Sousa, J., & Alchieri, E. A. P. (2014). State machine replication for the masses with BFT-SMART. In Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 355 – 362). IEEE.

Castro, M., & Liskov, B. (1999). Practical Byzantine fault tolerance. In Proceedings of the 3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI) (pp. 173 – 186).

Gueta, G. G., Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M. K., Seredinschi, D.-A., Tamir, O., & Tomescu, A. (2018). SBFT: A scalable and decentralized trust infrastructure. arXiv preprint arXiv:1804.01626.

Li, W., Feng, C., Zhang, L., Xu, H., Cao, B., & Imran, M. (2020). A scalable multi-layer PBFT consensus for blockchain. IEEE Transactions on Parallel and Distributed Systems, 32(5), 1146 – 1160. https://doi.org/10.1109/TPDS.2020.3038142

Li, Z., Wang, J., & Li, Y. (2025). An improved PBFT consensus algorithm based on reputation and gaming. The Journal of Supercomputing, 81(1), 323.

Madigan, D. J., Litvin, S. Y., Popp, B. N., Carlisle, A. B., Farwell, C. J., & Block, B. A. (2012). Tissue turnover rates and isotopic trophic discrimination factors in the endothermic teleost, Pacific bluefin tuna (Thunnus orientalis). PLOS ONE, 7(11), e49220. https://doi.org/10.1371/journal.pone.0049220

Na, Y., Wen, Z., Fang, J., Zhang, X., & Wang, J. (2022). A derivative PBFT blockchain consensus algorithm with dual primary nodes based on separation of powers (DPNPBFT). IEEE Access, 10, 76114 – 76124.

Sukhwani, H., Martínez, J. M., Chang, X., Trivedi, K. S., & Rindos, A. (2017). Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger

Fabric). In Proceedings of the 36th IEEE Symposium on Reliable Distributed Systems (SRDS) (pp. 253–255). IEEE.

Vedant, A., Yadav, A., Sharma, S., Thite, O., & Sheikh, A. (2022). A practical Byzantine fault tolerance blockchain for securing vehicle-to-grid energy trading. In Proceedings of the Global Energy Conference (GEC) (pp. 1–6). IEEE.

Veronese, G. S., Correia, M., Bessani, A. N., & Lung, L. C. (2009). Spin one's wheels? Byzantine fault tolerance with a spinning primary. In Proceedings of the 28th IEEE International Symposium on Reliable Distributed Systems (SRDS) (pp. 135–144). IEEE.

Wu, X., Wang, Z., Li, X., Ding, J., Tian, J., & Liu, Y. (2025). DBPBFT: A hierarchical PBFT consensus algorithm with dual blockchain for IoT. Future Generation Computer Systems, 162, 107429.

Yin, M., Malkhi, D., Reiter, M. K., Gueta, G. G., & Abraham, I. (2019). HotStuff: BFT consensus in the lens of blockchain. In Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC) (pp. 347–356).

Zhong, W., Feng, W., Huang, M., Du, D., & Li, Z. (2023). ST-PBFT: An optimized PBFT consensus algorithm for intellectual property transaction scenarios. Electronics, 12(2), 325.