High-Confidence Computing 2 (2022) 100070

Contents lists available at ScienceDirect

High-Confidence Computing

homepage: www.elsevier.com/locate/hcc

Workload-based randomization byzantine fault tolerance consensus protocol

Baohua Huang*, Li Peng, Weihong Zhao, Ningjiang Chen

School of Computer and Electronic Information, Guangxi University, Nanning, China

ARTICLE INFO

Keywords:
Blockchain
Consensus algorithm
BFT
PBFT
Verifiable random function

ABSTRACT

This paper introduces a new Byzantine fault tolerance protocol called workload-based randomization Byzantine fault tolerance protocol (WRBFT). Improvements are made to the Practical Byzantine Fault Tolerance (PBFT), which has an important position in the Byzantine Fault consensus algorithm. Although PBFT has numerous advantages, its primary node selection mechanism is overly fixed, the communication overhead of the consensus process is also high, and nodes cannot join and exit dynamically. To solve these problems, the WRBFT proposed in this paper combines node consensus workload and verifiable random function (VRF) to randomly select the more reliable primary node that dominates the consensus. The selection of the nodes involved in the consensus is based on the node workload, and the optimization of the agreement protocol of the PBFT is also based on this. Simulation results show that the WRBFT has higher throughput, lower consensus latency, and higher algorithmic efficiency compared to the PBFT.

1. Introduction

Blockchain is a distributed ledger system that allows for reliable data storage. Data is stored in the form of blocks, which are linked into a verifiable, tamper-evident chain by recording the hash value of the previous block. Since the block data exists in more than one node in the blockchain, it is possible to trust the data recorded in the blockchain network without trusting any node. Therefore, how to ensure the consistency of the block data and the linkage of the blocks stored in each node is the core of the blockchain, and the consensus algorithm is the solution used to solve this problem in the blockchain.

In 1980, Pease et al. proposed the consistency problem in distributed systems [1], i.e., when there are crash nodes in a distributed system that cannot respond to messages in time due to network congestion or hardware failure, how can the nodes agree on a decision or a state change. In 1982, Lamport proposed the Byzantine Generals Problem [2], which asks how to ensure consistency when there are traitor generals, i.e., Byzantine nodes, in a distributed system. Unlike the previously proposed crash nodes, Byzantine nodes are malicious nodes that can take any action to interfere with the consensus of the system, such as deliberately not responding to messages or responding to wrong messages.

Consensus algorithms have been classified into two categories based on their ability to tolerate Byzantine nodes: Byzantine Fault Tolerance (BFT) consensus algorithms and non-BFT consensus algorithms. The classical non-BFT consensus algorithms include the Paxos [3] consensus algorithm proposed by Lamport in 1989 and a simplified Paxos algorithm called Raft [4] proposed by Ongaro et al. Since then, most of the research on non-BFT consensus algorithms has been carried out based on these two works.

The first practical and effective BFT consensus algorithm that can solve Byzantine faults in a weak synchronous environment was the Practical Byzantine Fault Tolerance (PBFT) [5] proposed by Castro et al. in 1999. Later, in 2008, Satoshi Nakamoto proposed the Proof-of-Work (PoW) [6] based on hash collisions. In PoW, all nodes that want to be the next block proposer need to keep searching for a random number that satisfies the system requirements. This process wastes a lot of hash power. To reduce this waste, Proof-of-Stake (PoS) [7] was proposed. Instead of using hash power to generate random numbers, each node in PoS uses the equity it owns to find random numbers, which reduces the waste of hash power to some extent. BitShare improves the PoS consensus algorithm by combining the delegation voting system and proposes the Delegation Proof-of-Stake (DPoS) [8], which further improves the consensus efficiency on the basis of PoS and no longer needs to consume hash power, but it is easy to be centralized.

Compared with PoW, PoS, and DPoS, PBFT is a deterministic consensus algorithm without forks and tolerant of Byzantine nodes, thus it can be considered as a more ideal consensus algorithm. However, PBFT also has certain drawbacks, including a fixed way of primary node selection, the inability to work in dynamic network environments, the necessity

* Corresponding author.
E-mail address: bhhuang66@gxu.edu.cn (B. Huang).

https://doi.org/10.1016/j.hcc.2022.100070
Received 14 March 2022; Received in revised form 27 May 2022; Accepted 12 June 2022

to spend a significant amount of communication overhead to reach an agreement, et.

Many studies proposed improvements revolve around the selection of PBFT nodes, including the selection of consensus node sets and primary node. For example, the introduction of node credit values to select high credit value nodes as primary node enhances security while improving consensus efficiency. However, such consensus algorithms incorporating trust models become more likely to have its node selection predicted as the system runs, which can generate other security issues. Some subsequent studies have combined verifiable random functions for the selection of nodes and primary nodes involved in consensus. Because of the smaller consensus scale, transaction latency is reduced. And makes the selected nodes unpredictable and has higher security. However, at this stage, the majority of the improved consensus algorithms with random selectivity require tokens.

Therefore, an improved Byzantine consensus algorithm SBFT is proposed in this paper, and the contributions of this paper are as follows.

1) It is proposed to calculate the workload of the nodes as the credit credentials of the nodes. Because of their different roles in the consensus process, nodes must take different workloads, and this paper proposes to evaluate the workload value of nodes based on the workload they do and their positive contribution to consensus reaching, which is used to select more reliable nodes to participate and lead the consensus.
2) Optimize the selection of consensus nodes and primary node by combining node workload values and VRF. This enables SBFT to better resist malicious attacks and reduce the probability of consensus failure, thus increasing system throughput and reducing transaction latency.
3) Based on the selection of more trusted and reliable primary node, we also simplify the consistency protocol of PBFT to reduce the transaction latency and communication overhead required for consensus.

2. Related work

In response to some of the problems with PBFT, domestic and foreign scholars have improved PBFT mainly in terms of optimization mechanism and architecture. The papers [9–11] optimize PBFT in terms of hierarchical structure to improve scalability and consensus efficiency. The literature [12,13] proposes a speculative Byzantine fault-tolerant consensus SBFT that uses a collector to aggregate messages to turn PBFT into linear PBFT, significantly reducing the amount of communication. In the study of improved PBFT node selection based on credit values, Gao et al. proposed the T-PBFT [14] based on the EigenTrust trust model, which replaces a single primary node with a primary cluster, evaluates node trust with a credit model, and selects high-quality nodes in the network to establish a consensus group, with the aim of reducing the probability of view changes and improving the fault tolerance of the system. Tong et al. proposed the Trust-PBFT [15] consensus algorithm by integrating the PeerTrust trust computing model and PBFT, which improves the fault tolerance, system throughput, and system scalability of the consensus mechanism by selecting nodes with high trust to participate in transactions.

Among the studies on improved PBFT node selection incorporating randomness, Algorand [16,17] consensus algorithm combines PoS and verifiable random function (VRF) [18] to randomly select verification nodes and proposal nodes with high security, high throughput, and resistance to malicious attacks, however, its communication overhead is also high. Ontology [19] project introduced the VBFT consensus algorithm based on PoS and VRF. VBFT classifies nodes into consensus nodes and alternate nodes, where consensus nodes are divided into proposal nodes, verification nodes, and confirmation nodes, and it uses VRF to randomly select the set of consensus nodes. It has high security, high throughput, and low latency, but users need to pledge tokens to participate in the consensus network. Wu et al. proposed an optimized hybrid consensus algorithm based on PoS and PBFT [20], which uses verifiable random ordering to reduce the number of nodes involved in consensus to a constant, and its throughput, latency, and scalability are better than the PBFT algorithm. Zhan et al. proposed the delegated randomized Byzantine fault-tolerant protocol DRBFT [21] based on PBFT and developed a random selection algorithm called RS to cooperate with the voting mechanism to reduce the number of nodes involved in the consensus process, and this scheme is unpredictable, randomized and fair.

After comparing and analyzing some existing consensus algorithms, we proposed a workload-based randomization Byzantine fault tolerance consensus protocol named WRBFT, which optimizes the selection of consensus nodes and primary nodes by combining node workload and verifiable random functions.

3. Overview of PBFT consensus algorithm

3.1. Basic ideas of PBFT

PBFT is a form of state machine replication and mainly includes agreement protocol, view change protocol, and checkpoint protocol. The agreement protocol is used to ensure that all honest nodes process client requests in at least the same order and can return correct and consistent results to the client. The view change protocol is used to keep the consensus algorithm active and ensure that it continues to work properly when the consensus fails. The checkpoint protocol is used to set checkpoints, according to which nodes can safely discard information from the logs to reduce the memory overhead of the nodes, and can also be used to help to crash nodes synchronize to the latest state.

PBFT is a view-based consensus protocol, where the consensus process consists of a succession of views. In a view, there exists a determined primary node to dominate the consensus process, and the other nodes are replicas participating in the consensus. Suppose there are N nodes, and the maximum number of Byzantine nodes that the PBFT consensus algorithm can tolerate is f. PBFT algorithm can perform consensus properly if and only if $N \geq 3f+1$.

In PBFT, views are numbered sequentially, by view changes in an accumulative form, and nodes are numbered sequentially from 0 to N-1. The number p of the primary node in each view is determined by the view number v and the number of consensus nodes N calculated according to $p = v \bmod N$. When a view times out and fails to reach consensus or the primary node fails, the view change protocol will be executed and change to the next view to continue the consensus.

3.2. Normal-Case Operation

Under normal circumstances, the agreement protocol of PBFT belongs to a typical three-stage process, namely, *pre-prepare, prepare*, and *commit*. The process is shown in Figure 1, and the detailed steps are as follows:

1) *Request*: client c sends a request message <REQUEST,o,t,c> to the consensus network. Where o denotes the specific operation of the request and t denotes the timestamp of the request.
2) *Pre-prepare*: when the primary node receives an authenticated *request* message, it assigns a sequence number *seq* to this request in the current v and broadcasts a *pre-prepare* message <<PRE-PREPARE,v,*seq*,d>,m> to all backup nodes, where m is the request message and d is the digest of m.
3) *Prepare*: replicas i broadcasts *prepare* message <PREPARE,v,*seq*,d, i> to the other backup nodes after receiving the validated *pre-prepare* message.
4) *Commit*: the replica i records the verification-passed *prepare* message to the log after receiving it. If cumulatively $2f+1$ verification-passed *prepare* messages are received from different nodes (in-

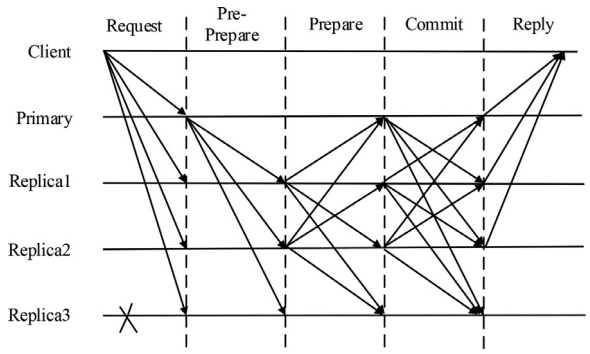

Fig. 1. Execution flow of PBFT consensus algorithm.

cluding itself), a *commit* message <COMMIT,*v,seq,d,i*> is broadcast to all backup nodes.

5) *Reply*: After receiving 2*f*+1 (including its own) validated commit messages, the replica *i* executes the specific operation of the request and sends a *reply* message <REPLY,*v,t,c,i,r*> to the client, with *r* being the result of the request execution. The client considers the request executed when it receives a correct reply from *f*+1 different replicas.

3.3. Deficiency analysis of PBFT algorithm

Based on a detailed analysis of the PBFT process, it can be concluded that the PBFT algorithm has the following inadequacy:

1) Fixed primary node selection scheme: The primary node is selected by rotation, which does not guarantee its reliability and is easily manipulated by adversaries. The known malicious nodes or faulty nodes in the previous view will still dominate the consensus as to the primary node with the view change, making the reliability and security of the system significantly reduced.
2) High communication overhead: PBFT's agreement protocol includes two multi-to-multiple network-wide broadcasts, which are quite bandwidth-intensive. The network is highly susceptible to congestion when the number of requests is too large.
3) Poor system scalability: There is no flexible consensus node join and exit mechanism to meet the demand of system node change, and it makes known malicious nodes or faulty nodes exist and participate in consensus all the time, so the system has poor security, availability, and scalability.

4. Improved consensus algorithm

The WRBFT algorithm proposed in this paper dynamically calculates the workload values of nodes by evaluating their participation after each consensus round and selects nodes to participate in consensus based on their workload values. Then the node workload value and verifiable random function are combined for primary node selection, which makes the selected primary nodes unpredictable while reducing the probability of non-honest nodes becoming primary nodes. Moreover, the agreement protocol of PBFT is improved based on the selection of more reliable primary nodes, which reduces the time and communication overhead required for consensus.

4.1. Node workload value

4.1.1. Node workload value calculation

The workload value of nodes proposed in this paper is calculated based on the workload made by nodes cumulatively in each consensus round, and the workload value of nodes is dynamically updated after each consensus process. The initial workload value of a node is WV_{init}, and the workload value $TotalWV_v^i$ of node i after participating in the consensus of view v is calculated according to equation (4.1).

$$TotalWV_v^i = TotalWV_{v'}^i * \left(1 + Succ_v * Pri_v^i * \delta\right) \tag{4.1}$$

In the above equation, $TotalWV_{v'}^i$ is the workload value of node i before participating in view v, and v' is the view number of the last consensus participation of node i before view v. $Succ_v$ is a parameter related to whether consensus is completed in view v. If consensus is reached in view v, $Succ_v$ takes a positive value, and in this case, the value of $TotalWV_v^i$ increases; conversely, if consensus is not completed, $Succ_v$ takes a negative value and the value of $TotalWV_v^i$ decreases. The value of Pri_v^i is determined by whether node i participates in the consensus of view v as the primary node, and is used to distinguish the workload of the primary node from the consensus node to ensure that after a successful consensus, the workload value of the primary node increases more than the consensus node and decreases more than the consensus node in case of failure. δ is the base increment for calculating the workload value. When a node is detected to have more than two instances of dishonest behavior, its workload value will drop below the minimum threshold WV_{low}. dummy citation Table 1.

4.1.2. Workload value recovery mechanism

To prevent nodes from inert behavior or tend to centralize due to high workload values, we design a node workload value recovery mechanism. By maintaining a workload value threshold interval [WV_{low},WV_{high}], when executing the node workload value recovery mechanism, the workload values of nodes with workload values greater than WV_{high} are recovered to a random value in the interval [WV_{init}, WV_{high}], and the workload values of nodes in the interval [WV_{low} +(WV_{init} - WV_{low})/2, WV_{init}) that is restored to WV_{init}. The node workload value recovery mechanism can ensure the motivation of low workload value nodes and also reduce the possibility of inert behavior or malicious behavior of nodes due to high workload value. The flow of the node workload value recovery mechanism is shown in Algorithm 1.

Algorithm 1
Node workload value recovery algorithm

Input:$(W, WV_{init}, WV_{low}, WV_{high}, N)$
Output:(W)

```
1./* W is an array of node workload values and N is the number of nodes */
2. t ←WV_low + (WV_init - WV_low )/2;
3.for i←0 to N-1 do
4. if W [i] > WV_high
5. W [i] ← Random(WV_init, WV_high);
6. else if W [i] >= t and W[i] < WV_init
7. W [i] ← WV_init;
8. end if
9. end for
10. return W;
```

4.2. Node Selection

In WRBFT, nodes are classified into the primary node, consensus nodes, and candidate nodes. The primary node in WRBFT are selected by all nodes in the consensus network with workload values greater than WV_{low} +(WV_{init} - WV_{low})/2 using a cryptographic sortition algorithm, based on the workload values as node weights. Assume there are N nodes, and the top n nodes are selected as consensus nodes in each round in order of workload value to participate in the consensus, and the other candidate consensus nodes are kept in sync with the consensus network.

4.2.1. Selection of the primary node

The selection of the primary node in view v is determined by all nodes in the consensus network with $TotalWV_v^i$ greater than WV_{low} +(WV_{init} - WV_{low})/2 using a cryptographic sortition algorithm based on

verifiable random function (VRF) implementation. VRF has three prime properties: verifiability, uniqueness, and randomness. Each node holds a key pair (public key pk_i, private key sk_i), the VRF computation function *VRF_Evaluate* produces only unique outputs *vrf_hash* and *proof* with the same input sk_i and message *msg*. The *vrf_hash* is a pseudo-random string of hash length uniquely determined by sk_i and *msg*. The *proof* is available to the person who has pk_i to verify whether *vrf_hash* indeed corresponds to *msg* using the *VRF_Verify* function. In WRBFT, primary node selection is based on the VRF, which makes the selection of the primary node unpredictable, but still not resistant to Sybil attack. Therefore, this paper combines the node workload to better defend against malicious attacks.

Cryptographic sortition as shown in Algorithm 2. Nodes i uses VRF to calculate the hash value *vrf_hash*, and it is uniformly distributed in the interval $[0,2^{hashlen}]$, let e=*vrf_hash* $/2^{hashlen}$, even though e is uniformly distributed between [0,1]. Set an expectation value σ and the total workload value of participating consensus nodes as *WV_Sum*, and calculate the probability that node i succeeds k times in WV_i trips of Bernoulli experiments with probability μ, denoted as $B(k; WV_i, \mu)$, based on the probability $\mu=\sigma/WV\_Sum$ and the workload value of nodes WV_i. The sum of the values of k taken from 0 to j is summed to find the value of j when e falls within interval I, which is shown in equation (4.2)

$$I = \left[\sum_{k=0}^{j} B(k; WV_i, \mu), \sum_{k=0}^{j+1} B(k; WV_i, \mu)\right] \tag{4.2}$$

Algorithm 2
Cryptographic sortition algorithm

Input:$(sk_i, seed, \sigma, WV_i, WV\_Sum)$
Output:$(vrf\_hash, proof, j)$

1./*seed is a safety parameter */
2. *vrf_hash, proof* ← *VRF_Evaluate*(sk_i, *seed*);
3. $e \leftarrow vrf\_hash/2^{hashlen}$;
4. $\mu \leftarrow \sigma/WV\_Sum$;
5. $j \leftarrow 0$;
6. **while** $e < \sum_{k=0}^{j} B(k; WV_i, \mu)$
7. j++;
8. **end while**
9. **return** *vrf_hash, proof, j*;

The j that satisfies the condition is the sort result of node i. When j greater than 0 means that node i success and is allowed to be the primary node. When multiple nodes are successful, the one with the largest value of j becomes the primary node, and the one with the smallest node number becomes the primary node when there is the same j.

We need a continuously updated, randomly selected, publicly available security parameter *seed* as input to the function *VRF_Evaluate* in each consensus round, which is used to avoid selecting the same consensus node each time so that each node can generate a different random number at each sortition. If the last generated block is *Blk*, we update the security parameter *seed* in view v using the hash of the block Blk_v, i.e., $seed_v = Hash(Blk_v)$.

Consensus nodes verify the sortition result of node i based on the *vrf_hash, proof,* and sortition result j sent by node i, using the verification key pk_i maintained by this node and *seed* to verify the sortition result. The verification process is shown in algorithm 3.

4.3. Consensus process

Compared to the typical three-stage consensus process of PBFT, WRBFT replaces multi-to-multiple communication with one-to-multiple communication based on the selection of a more trusted primary node. Furthermore, WRBFT incorporates the view change process into the normal consensus process. In PBFT, the view change protocol is executed only when an abnormal situation occurs and enough nodes expect to replace the view. In WRBFT, on the other hand, after each consensus,

Algorithm 3
Verify algorithm

Input:$(pk_i, seed, \sigma, WV_i, WV\_Sum, vrf\_hash, proof, j)$
Output:$(true/false)$

1. **if** ¬*VRF_Verify*(pk_i, *seed, vrf_hash, proof*)
2. **return** *false*;
3. **end if**
4. $e \leftarrow vrf\_hash/2^{hashlen}$;
5. $\mu \leftarrow \sigma/WV\_Sum$;
6. $j' \leftarrow 0$;
7. **while** $e < \sum_{k=0}^{j'} B(k; WV_i, \mu)$
8. j'++;
9. **end while**
10. **if** j != j'
11. **return** *false*;
12. **end if**
13. **return** *true*;

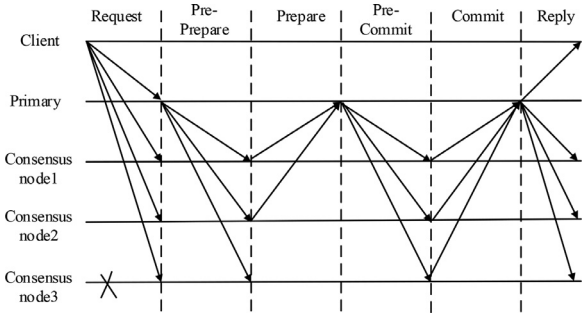

Fig. 2. Execution flow of WRBFT consensus algorithm.

the consensus node is reselected to change to the next view to continue the consensus, regardless of whether it is successful or not. To keep the liveliness of the algorithm, we use a timeout mechanism to guarantee the view change.

WRBFT extends the two-stage confirmation of PBFT into a three-stage confirmation, i.e., a four-stage consensus process. The process is shown in Figure 2, and the specific consensus process is as follows:

1) *Request*: client c sends a request message <REQUEST,o,t,c> to the consensus network. Where o denotes the specific operation of the request and t denotes the timestamp of the request.
2) *Pre-prepare*: when the primary node receives an authenticated *request* message, it assigns a sequence number *seq* to this request in view v, and broadcasts a *pre-prepare* message <<PRE-PREPARE,v,*seq*,d,s>,m> to all consensus nodes. Where m is the request message and d is the digest of message m, and s is the sortition result, including *vrf_hash, proof*, and j.
3) *Prepare*: consensus node i sends *prepare* message <PREPARE,v,*seq*, d,i> to the primary node after receiving the validated *pre-prepare* message.
4) *Pre-commit*: the primary node writes all the received validated *prepare* messages to the log, and when it receives $2f$+1 validated *prepare* messages from different consensus nodes (including itself), it broadcasts a *pre-commit* message <PRE-COMMIT,v,*seq*,d,i> to all consensus nodes.
5) *Commit*: consensus node i executes the specific operation of the request after receiving the validated *pre-commit* message. Then, sends *commit* message <COMMIT,v,*seq*,d,i> to the primary node.
6) *Reply*: the primary node executes the specific operation of the request after receiving $2f$+1 validated *commit* messages and sends a *reply* message <REPLY,v,t,c,i,r> to the client and other consensus nodes, with r being the result of the request execution. If the client receives a correct *reply* from the primary node, the request is considered as having been executed.

Table 1
Experimental parameters table

Parameter	Setting
Node workload value initial value WV_{init}	50
Node workload value threshold WV_{low}, WV_{high}	20, 100
Marking the success of consensus: $Succ_v$	1, -1
Parameters to flag whether to participate in consensus as a primary node: Pri_v^i	1, 2
Base increment for node workload value calculation δ	1.0
A round of consensus process contains the number of transactions	2000
Timeout Threshold	5s

5. Experiments

The experiments in this paper are conducted on a Windows 10 system configured with Intel i7-9700 3.00 GHz processor and 16 GB RAM. WRBFT and PBFT are implemented through Go language, using threads to listen to different ports instead of nodes and opening multiple threads to simulate the communication process of consensus nodes. The experiments are conducted by analyzing the comparison of communication volume, throughput, transaction latency and fault tolerance with PBFT to verify the effectiveness of WRBFT. The parameter settings in the experiments are shown in Table 1.

5.1. Communication overhead analysis

5.1.1. Communication volume analysis

Assume that the total number of nodes in the system is N. In the consensus process of PBFT, the number of message interactions in the *Pre-prepare* is N-1, and the number of message interactions in both the *Prepare* and *Commit* is $N(N\text{-}1)$. And the amount of communication required for view change is $N(N\text{-}1)+(N\text{-}1)$. Assuming that g is the probability of occurring a view change, the total number of communications in the PBFT consensus process is $(2+g)N^2\text{-}N\text{-}g\text{-}1$.

Let the nodes selected to participate in the consensus process of WRBFT be n, and $n = hN$, $0< h\leq 1$.The amount of message interactions in all its four consensus phases is hN-1. At the end of each consensus, WRBFT will make a multi-to-multi communication into a new view, so the total interaction message volume of the WRBFT consensus process is $h^2N^2+3hN\text{-}4$.

5.1.2. Communication volume comparison

Let the ratio of communication overhead of PBFT algorithm to WRBFT algorithm be Y. Then we have:

$$Y = \left((2+g)N^2 - N - g - 1\right)/\left(h^2N^2 + 3hN - 4\right) \tag{5.1}$$

The experiments are conducted by taking g=0, h=2/3; g=0, h=1; g=1/2, h=2/3; g=1/2, h=1, respectively, and the comparison plots are shown in Fig 3. It can be seen that even in the case of g=0, h=1,i.e., no view change occurs in PBFT and all nodes in WRBFT participate in consensus, the communication volume of PBFT is about 1.5 times higher than WRBFT.

5.2. Throughput Analysis

To test and compare the throughput of the PBFT algorithm and WRBFT algorithm, the experiments were conducted several times with the different number of nodes in both cases of existence and non-existence of Byzantine nodes, and the average number of transactions per second reached consensus by nodes was counted.

The comparison results are shown in Fig 4. It can be seen that the throughput of the WRBFT algorithm is greater than that of the PBFT for different numbers of nodes, regardless of the presence of Byzantine nodes, due to the small size of the WRBFT consensus and the fact that the nodes involved in the consensus are more trustworthy and reliable with the advantage of being able to reach consensus quickly.

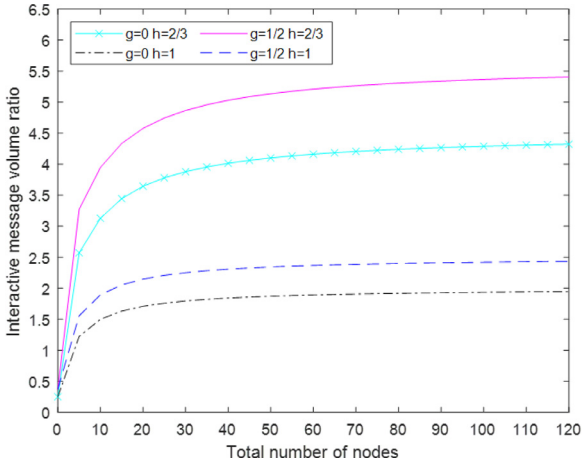

Fig. 3. Consensus interaction message volume comparison diagram.

5.3. Transaction Latency Analysis

The experiments took several values of the number of nodes and tested the delay required for a transaction to be executed from its generation to its execution in two cases with and without the presence of Byzantine nodes, and the results of the comparison of the two algorithms were based on the average of several experiments.

The comparison plots are shown in Figure 5. When the number of nodes increases, the latency of both algorithms grows. However, the latency of the WRBFT algorithm is always smaller than that of the PBFT and its latency grows slower. Comparing a) and b) in Figure 5, it can be found that the latency growth of WRBFT with Byzantine nodes is not significant compared to the case without Byzantine nodes, while in the case with Byzantine nodes, the latency growth of PBFT is faster due to the small possibility of Byzantine nodes participating in consensus in WRBFT, while Byzantine nodes are always present in PBFT, which leads to more view change.

5.4. Fault Tolerance Analysis

When the total number of nodes is N, the maximum number of Byzantine nodes that the PBFT can tolerate is $f=\lfloor (N-1)/3 \rfloor$. And for WRBFT, let $n=hN$, $0<h\leq 1$. When the system selects the nodes with the top n workload values as consensus nodes to participate in consensus, in the consensus process, to ensure the correctness of consensus, the primary node needs to receive at least $hN\text{-}f$ messages from different nodes, among which, the correct messages received should be more than f. So it is necessary to let $hN\text{-}f\text{-}f\geq f$,i.e.,$f\leq(hN\text{-}1)/3$. And among the remaining $(1\text{-}h)N$ nodes that do not participate in the consensus, all can be Byzantine nodes in the extreme case. At this point, the maximum Byzantine number that WRBFT can tolerate is $f'=f+(1\text{-}h)N$, and its fault tolerance drops to the same as PBFT only when h=1, i.e., all nodes in WRBFT participate in consensus.

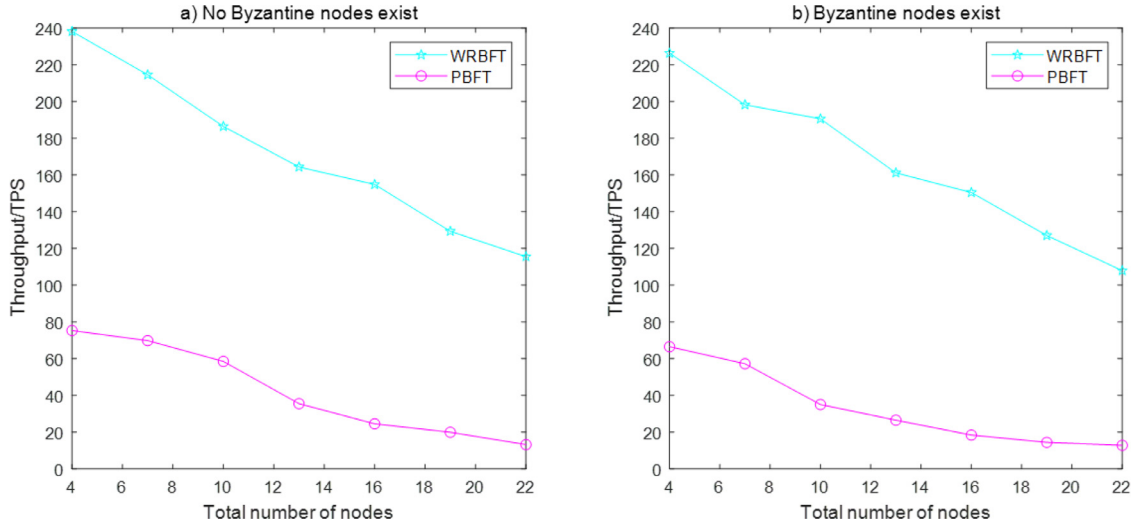

Fig. 4. Throughput comparison chart.

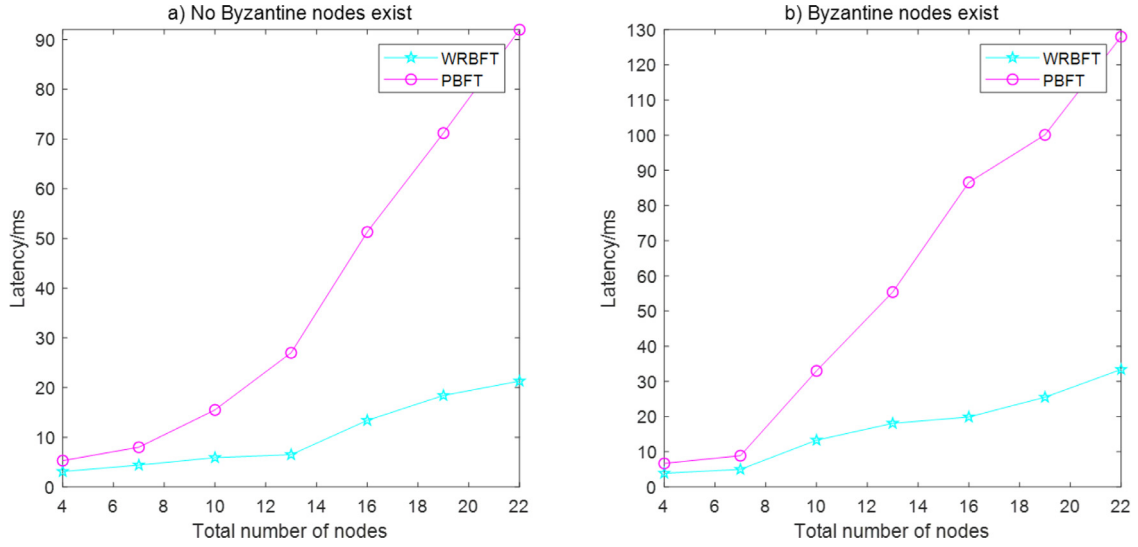

Fig. 5. Latency comparison chart.

Table 2
Comparisons of our SBFT with other BFT-type consensus algorithms

Consensus protocol	Whether to rely on tokens	Primary node selection method	Communication Complexity	Security	Computational complexity	Fault tolerance
PBFT [5]	No	$p = v \bmod N$	O(N^2)	Medium	Medium	(N-1)/3
SBFT [13]	No	$p = v \bmod N$	O(N)	Medium	Medium	(N-1)/3
T-PBFT [14]	No	Selected based on credit value	O(N^2)	Medium	Medium	$(1-\frac{2}{3}h)\cdot N-\frac{1}{3}$
VBFT	Yes	Combined with PoS random selection	O(N^2)	High	High	(N-1)/3
WRBFT	No	Combined with the workload value randomly selected	O(N)	High	High	$(1-\frac{2}{3}h)\cdot N-\frac{1}{3}$

5.5. Comparison with related works

We compare WRBFT with classical PBFT, an excellent scheme of BFT consensus protocols SBFT, T-PBFT, an improved scheme of credit-based PBFT, and VBFT, a scheme of BFT class combining PoS and VRF. In the compared schemes, both PBFT and the consensus algorithm based on PBFT algorithm are applicable to both federated and private chains. SBFT has the same primary node configuration and selection as PBFT, and both perform primary node selection according to the rotation when the view is replaced. This rotation approach cannot avoid Byzantine nodes from participating in consensus as primary nodes, which will lead to a higher probability of consensus timeout and failure, resulting in frequent view replacement and system performance degradation.

T-PBFT improves node selection based on PBFT by combining credit value, by selecting nodes with high credit value as primary nodes, compared to PBFT, it relies on more trusted primary nodes to reduce the probability of view replacement in the consensus process and effectively improves the consensus efficiency, but during the system operation, the primary node and consensus nodes in each view have the risk of being corrupted and breached, and it is difficult to resist malicious attacks

such as bribery attacks. VBFT combines PoS and VRF to improve the security effectively, based on PoS to ensure the trustworthiness of the nodes and VRF to make the selected nodes unpredictable, which can effectively resist various malicious attacks, but it relies on tokens in this mechanism and has high computational complexity, which cannot meet a wider range of application scenarios.

WRBFT, on the other hand, ensures the trustworthiness of the nodes involved in consensus by combining node workload values and VRF, reduces the size of the nodes involved in consensus in the network, and can reach consensus faster; it ensures the unpredictability of primary node selection and does not need to rely on pledged tokens, which can meet more application scenarios. WRBFT improves the consistency of PBFT based on selecting a more reliable primary node, which reduces the communication complexity in the consistency protocol by allowing the master node to take on more workload. In summary, a comparison between WRBFT and other BFT-type consensus algorithms is shown in Table 2.

6. Conclusion

In this paper, an optimized consensus algorithm WRBFT is proposed based on the PBFT algorithm. In this algorithm, consensus nodes are selected by calculating the node workload values, and primary nodes are selected randomly using a verifiable random function to make primary nodes and consensus nodes more reliable, which can effectively resist malicious attacks as well as reduce the possibility of a consensus failure. It also improves the agreement protocol of PBFT based on the selection of more trusted and reliable nodes, which reduces the time and communication overhead required for consensus. Experimental results show that the WRBFT algorithm proposed in this paper effectively reduces the communication overhead, improves the throughput, and reduces the transaction latency in the network when compared with the PBFT algorithm under the same conditions.

In future works, the calculation of node workload value in the algorithm can be improved to increase the dynamics and adaptability of the algorithm, and we can consider the combination of group signatures to speed up the consensus reaching.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61962005) and National Key Research and Development Program of China (No. 2018YFB1404404).

References

[1] M Pease, R Shostak, L. Lamport, Reaching agreement in the presence of faults[J], Journal of the ACM (JACM) 27 (2) (1980) 228–234.
[2] L Lamport, R Shostak, M Pease, The Byzantine generals problem[M], Concurrency: the Works of Leslie Lamport (2019) 203–226.
[3] L. Lamport, The Part-Time Parliament[J], Acm Transactions on Computer Systems 16 (2) (1998) 133–169.
[4] D Ongaro, J. Ousterhout, in: Search of an Understandable Consensus Algorithm[C]//2014 USENIX Annual Technical Conference (Usenix ATC 14), 2014, pp. 305–319.
[5] M Castro, B. Liskov, Practical Byzantine Fault, Tolerance[C]//OsDI. 1999, 99 (1999) 173–186.
[6] Nakamoto S. Bitcoin, A peer-to-peer electronic cash system[J], Decentralized Business Review (2008) 21260.
[7] S King, Nadal S. Ppcoin, Peer-to-peer crypto-currency with proof-of-stake[J], self--published paper 19 (1) (August, 2012).
[8] Schuh F, Larimer D. Bitshares 2.0: Financial smart contract platform[J]. Accessed: Jan, 2015, 15: 2017.
[9] Y Li, L Qiao, Z. Lv, An optimized byzantine fault tolerance algorithm for consortium blockchain[J], Peer-to-Peer Networking and Applications (2021) 1–14.
[10] W Li, C Feng, L Zhang, et al., A Scalable Multi-Layer PBFT Consensus for Blockchain[J], IEEE Transactions on Parallel and Distributed Systems 32 (5) (2021) 1146–1160.
[11] Z H CHEN, Q Li, Improved PBFT Consensus Mechanism based on K-medoids[J], Computer Science 46 (12) (2019) 101–107 in Chinese.
[12] R Kotla, L Alvisi, M Dahlin, et al., Zyzzyva: Speculative Byzantine Fault Tolerance[J], Acm Transactions on Computer Systems 27 (4) (2009).
[13] G G Gueta, I Abraham, S Grossman, et al., in: SBFT: A Scalable and Decentralized Trust Infrastructure[C]//2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2019, pp. 568–580.
[14] S Gao, T Yu, J Zhu, et al., T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm[J], China Communications 16 (12) (2019) 111–123.
[15] W Tong, X Dong, J. Zheng, in: Trust-PBFT: A PeerTrust-Based Practical Byzantine Consensus Algorithm[C]//2019 International Conference on Networking and Network Applications (NaNA), 2019, pp. 344–349.
[16] Leung D, Suhl A, Gilad Y, et al. Vault: Fast Bootstrapping for the Algorand Cryptocurrency[J]. 2019.
[17] Y Gilad, R Hemo, S Micali, et al., in: Algorand: Scaling Byzantine Agreements for Cryptocurrencies[C]//Proceedings of the Twenty-Sixth Acm Symposium on Operating Systems Principles, 2017, pp. 51–68.
[18] Micali S, Rabin M, Vadhan S. Verifiable random functions[J]. 1999: 120-130.
[19] Ontology Project.Consenses mechanism[EB/OL]. [2020 -06-14]. https://docs.ont.io/ontology-elements/consensus-mechanism.
[20] Y Wu, P Song, F. Wang, Hybrid Consensus Algorithm Optimization: A Mathematical Method Based on POS and PBFT and Its Application in Blockchain[J], Mathematical Problems in Engineering 2020 (2020) 7270624.
[21] Y Zhan, B Wang, R Lu, et al., DRBFT: Delegated randomization Byzantine fault tolerance consensus protocol for blockchains[J], Information Sciences 559 (2021) 8–21.