Journal of Computers Vol. 32 No. 3, 2021, pp. 274-288
doi:10.3966/199115992021063203020

Business-Aligned Raft Algorithm in the Consortium with Multiple Permissioned Blockchains

Yansong Zhang[1*], Bo Shen[1], Yingsi Zhao[2]

[1] Beijing Key Laboratory of Communication and Systems, School of Electronic and Information Engineering, Beijing Jiaotong University, Haidian District, Beijing, China
20120169@bjtu.edu.cn, bshen@bjtu.edu.cn

[2] School of Economics and Management, Beijing Jiaotong University, Haidian District, Beijing, China
yszhao@bjtu.edu.cn

Received 17 April 2021; Revised 1 May 2021; Accepted 17 May 2021

Abstract. Raft is a well-known consensus algorithm with extensive application scenarios thanks to its comprehensibility and scalability in permissioned blockchain. But some issues about consensus efficiency still exist in Raft. For example, Raft detaches from business, and the random variable about leader identity has the largest information entropy in the leader election process. In this paper, we propose a Raft-like consensus algorithm, named Business-aligned Raft (BARaft) algorithm, which supports high transaction throughput and business adaptability. BARaft aims at optimizing the performance of the consensus algorithm in the consortium with multiple permissioned blockchains. BARaft combines actual application scenarios and specific characteristic of nodes to improve the adaptability and scalability of business. It enhances the countdown mechanism of leader election in Raft to reduce the randomness of leader election and build up business efficiency with fewer forwarded transactions. It also introduces new node states to reduce the scale of consensus cluster and improves the consensus efficiency. Experimental results show that BARaft provides a higher transaction throughput than Raft by 46.95% in a cluster of five consensus nodes.

Keywords: permissioned blockchain, raft consensus algorithm, Hyperledger Fabric, business adaptability, leader election analysis

1 Introduction

The concept of blockchain originated from the white paper of bitcoin [1]. After continuous evolution, blockchain was finally abstracted from bitcoin and used as a cutting-edge technology alone, which has now been widely used. According to openness and method of accessing-the-chain, blockchain is divided into three categories: public blockchain, consortium blockchain and private blockchain [2]. Public blockchain is a kind of completely open blockchain in which anyone can participate and become part of network. Consortium blockchain is a kind of multi-centralized blockchain in which the joining and withdrawal of nodes should be authenticated by every consortium organization.

Consensus algorithm is a key technology in the blockchain, which is designed to ensure the consistency of data on distributed nodes. Data interaction and calculations of consensus algorithm will restrict the efficiency of blockchain applications to some degree. Existing consensus algorithms can be grouped into two types [3]: the first is proof-based consensus algorithm normally used in public blockchain such as PoW and PoS [1, 4]; the second is voting-based consensus algorithm requiring a lot of data interaction between consensus nodes. Voting-based consensus algorithm is commonly used in consortium blockchain and private blockchain because every node should obtain relatively accurate information from others. It can also be divided into two categories [3]: one is able to deal with the Byzantine generals problem such as PBFT; the other can only tolerate crashed nodes such as Raft.

* Corresponding Author

Authentication mechanism of permission blockchain can manage the joining and exiting of all nodes. It will increase the difficulty for byzantine nodes to do evil. Consensus algorithm, for example Raft, can just consider the case that nodes only crash in permission blockchain. Raft has the advantages of simplicity and understand-ability. Only more than half of nodes need to work properly in a Raft consensus cluster to achieve data consistency. Because of high centralization and simplicity, Raft is more efficient than byzantine fault-tolerant algorithms [5]. However, aiming at enhancing consensus efficiency of Raft, we found the following problems: (1) Leader election process has high randomness and heartbeat mechanism is simple. Nodes have the same probability to become a leader in the leader election process and leader undertakes necessary algorithmic tasks in Raft. So if there is a performance-limited or network-status-poor node becoming the leader, it might cause multiple rounds of leader election process and make the whole consensus process slow. (2) The states transfer and data consistency of Raft are independent of the upper-level business. Consistency process is identical for different business scenarios. This may cause some problems, for example when client requires that one node can't elect as the leader. (3) Raft never considers the multi-chains demand in a consortium. If a node maintains multiple blockchains simultaneously in a consortium, it may cause the large memory pressure.

In response to the above problems, we proposed an improved Raft consensus algorithm, called business-aligned Raft algorithm, which can realize adaptive status adjustments according to different business requirements in a multi-chains consortium. The major contributions made in this paper are as follows: (1) Node status table is introduced to record the status of all nodes in a cluster. It contains business information to implement business adaptability of the consensus algorithm. (2) Heartbeat mechanism can be adjusted through node status table in order to reduce the randomness of leader election effectively. (3) By adding two kinds of node states, the scale of consensus cluster can be reduced, which can accelerate consensus process.

Section 2 introduces the existing research on Raft algorithm. Section 3 presents design and consensus process of BARaft. Section 4 theoretically analyzes the efficiency and availability of BARaft. Section 5 is the experimental part. Section 6 concludes the paper.

2 Related Work

Raft consensus algorithm can be divided into two key steps [6]: leader election and log replication. The states transfer figure of Raft is shown in Fig. 1. Every node will be in one of three states in Raft consensus cluster. A well-working leader broadcasts heartbeat messages to network at regular intervals. But if a follower fails to receive a heartbeat message or log append message from the leader, it will make the election-timeout timer countdown end resulting in the state transition from follower to candidate. A candidate elects the leader by sending vote messages to the other nodes. If a candidate receives votes from a majority of nodes in consensus cluster, it wins an election and transitions to leader state. Leader is the core node of a cluster dominating log replication and handling all transactions from clients.

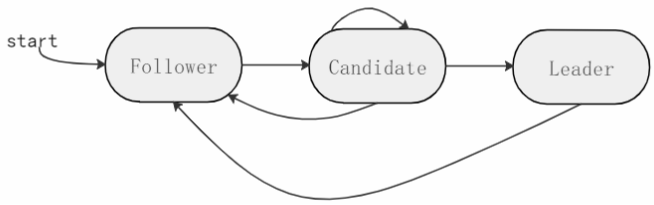


Fig. 1. The states transfer figure of Raft

Our empirical study shows that there are two strategies for the improvement of Raft: one is to enhance its safety, for example making Raft can tolerate a certain amount of byzantine nodes; the other is to improve the performance and stability. Christopher Copeland et al. [7] proposed a byzantine fault tolerant Raft algorithm keeping the simplicity and comprehensibility. They analyzed the affects of byzantine nodes on the process of leader election and log replication, and given some solutions by integrating message interaction pattern of PBFT [8] in Raft. Rihong Wang et al. [9] introduced Kademlia protocol to improve Raft, which optimized leader election and log replication by a kind of table, called K-Bucket. Yu Gao et al. [10] altered leader election to a leadership transfer between multiple controllers. Before leader

lost, the leadership was actively transferred to other controllers, which can effectively shorten the network unavailability time caused by leader shutdown.

Most of existing Raft-like consensus algorithms are aiming at improving leader election and log replication process technically, without considering actual application scenarios and specific characteristic of nodes. In this paper, by combining Raft with business layer information, the consensus algorithm can realize adaptive status adjustments when facing different application requirements.

3 Protocol Design

3.1 Node Design

The node roles of BARaft can be divided into five types: leader, Candidate, follower, outsider and initiator. The state transformation model is shown in Fig. 2. Leader is the core node of a consensus cluster responsible for transaction messages processing, log replication process maintenance and consensus process control. Follower participates in the consensus process of consensus cluster, who will receive log append messages from leader and save consensus log entries. When follower fails to receive a message from leader for a period of time, it will convert state to candidate which is a temporary state. Candidate launches leader election process and broadcasts the vote messages. After polling the votes of a majority of nodes, the candidate will be elected as the new-term leader. When the network is established firstly, outsiders are elected. They do not participate in the consensus process of a blockchain and only receive blocks passed by leader. Identity information of outsiders is bound to a blockchain by writing it into the block metadata of the blockchain. When a blockchain restarts, the identity information of outsiders will be quickly extracted from existing blocks. Initiator is also a transient state, used to ensure the timeliness and integrity of the node status table maintained by all nodes in the stage of network establishment. Initiator will broadcast its status data, and then the other nodes will update node status tables maintained by themselves when receive the status data of the initiator.

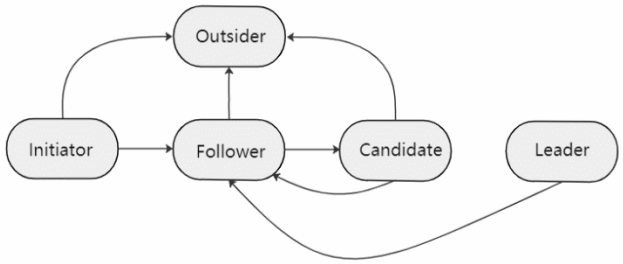


Fig. 2. The states transfer figure of BARaft

3.2 Node Status Table

Node status table is a core component of BARaft, which can reduce the randomness of leader election process, decrease the scale of consensus cluster, and increase business adaptability. There are five items for one node in a node status table by default, which are computing performance, the number of blockchains currently maintained by the node, business correspondence, business constraint, and log synchronization progress of the node. The first two items are attributes common to all blockchains; the latter three items are related to a blockchain. The contents of node status tables will be different between two blockchains on account of the different business requirements.

The functions of node status table are as follows: (1) Outsiders election. In the stage of blockchain establishment, BARaft will evaluate the performance of nodes and elect chain-based outsiders by the node status table. (2) Optimize leader election process. Node status table reduces the randomness of leader election process and increases the probability of a better-performance node becoming the leader by adjusting the countdown mechanism of leader election. (3) Business adaptability of consensus algorithm. BARaft can adjust itself according to business by adding business constraints to the node status table, such as node A must participate in consensus process and node B should try not to participate.

Modeling of Business Relationship. Nodes can be divided into consensus nodes and client nodes based on business. Client nodes can send transaction request messages to consensus cluster. It must establish connections with specific consensus nodes firstly and then pick one of them to communicate. Therefore, there is a graph structure correspondence between client nodes and consensus nodes in one consensus cluster. In a node status table, the graph structure is modeled as a matrix able to be processed by BARaft. Fig. 3 shows an example of business correspondence modeling. The columns of the matrix represent consensus nodes, the rows represent client nodes, and the matrix element whose sum by row is 1 represent the probability of connection between consensus node and client node. Through business matrix, BARaft can calculate the average number of clients served by each consensus node, which shows the business importance of consensus nodes.

	node1	node2	node3	node4	node5
client1	1	0	0	0	0
client2	1	0	0	0	0
client3	0.5	0	0.5	0	0
client4	0	1	0	0	0
client5	0	0	0	1	0
Sum	2.5	1	0.5	1	0
Score	0.5	0.2	0.1	0.2	0

Fig. 3. Business matrix

Business constraints are modeled as structured sentences that can be recognized by BARaft. The structured sentences contain three items: Subject, constraints including two types: “must” and “not”, and identity information. Fig. 4 shows that node 1 cannot be elected as outsider, that represents that node 1 must participate in consensus process.

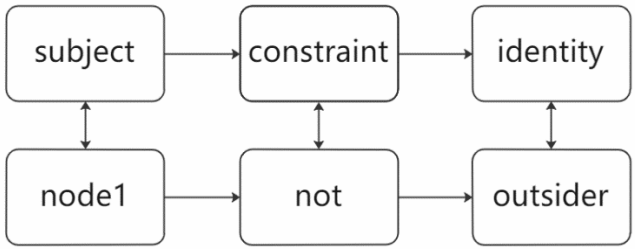


Fig. 4. Business constraint

Outsiders Election Algorithm Design. Outsiders election algorithm must satisfy two requirements. First, since outsiders do not participate in the consensus process of a blockchain and only passively received blocks, it is necessary to comprehensively consider the performance of nodes, consensus efficiency and business importance of nodes to select appropriate outsiders. The second is the number of outsiders. Due to the particularity of outsiders, large number of outsiders easily makes consensus cluster vulnerable, but small may cause the improvement of consensus efficiency to be insignificant. Therefore, outsider number should ensure maximize consensus efficiency while minimizing the impact on network stability.

In response to the above analysis, we proposed a measurement mechanism based on node status table and a scheme for confirming the number of outsiders, which together constitute the election algorithm of outsides.

Measurement Mechanism. Measurement mechanism is based on three items of computing performance, the number of blockchains currently maintained by the node and business correspondence. The specific formula of measurement mechanism is shown as Eq. (1).

$$Score = \frac{\alpha_1 \frac{S_p}{P_{max}} + \frac{\alpha_2}{2}\left[\tanh \frac{4}{C_{max}}\left(\frac{C_{max}}{2} - n_c\right) + 1\right] + \beta_1 S_b}{\alpha_1 + \alpha_2 + \beta_1}, \quad \textbf{(1)}$$

where S_p is the computing performance value of a consensus node and P_{max} is the maximum of it. n_c is the number of blockchains maintained by a consensus node and C_{max} is the maximum of it. S_b is the business value of a consensus node. α_1, α_2 and β_1 are weights. *Score* is a normalized value positively related to physical performance and business importance of a consensus node. So we can sort the scores to choose a certain number of outsiders.

n_c is mapped to the range of [0, 1] by nonlinear method whose function graph is shown in Fig. 5. The mapping function is a hyperbolic-tangent-like function enable to make the changing rate of function around $\frac{1}{2}C_{max}$ is greater than 0 and C_{max} because the number of blockchains maintained by each consensus node is likely to be distributed around half of the maximum number of blockchains under normal statistics.

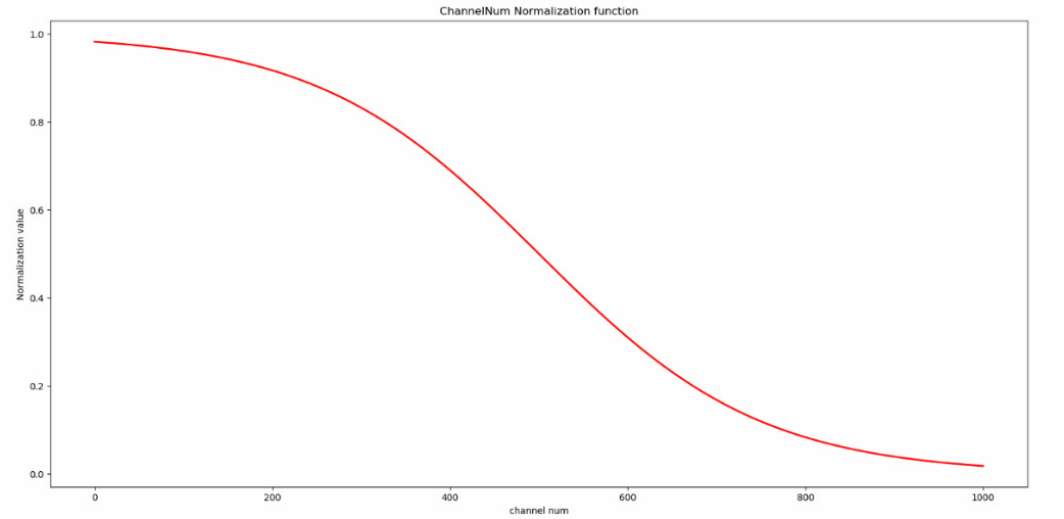


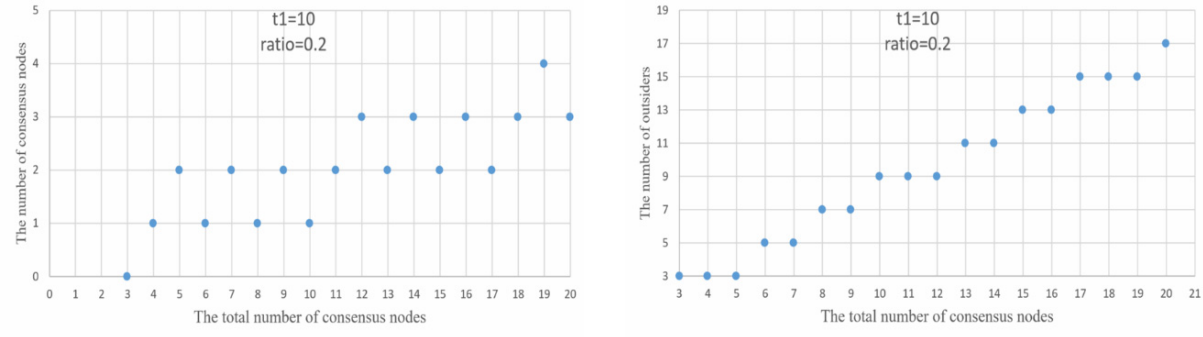
Fig. 5. Hyperbolic-tangent-like mapping function

Based on the business matrix, we can figure out S_b in the range from [0, 1] by summing the matrix elements by column and then dividing by the total number of clients. Consensus node has more important business and serves more clients if the S_b value of it is larger.

The Number of Outsiders. We proposed a piece-wise function to determine the number of outsiders including three intervals. The first interval is stable period in which the network stability significantly affects the number of consensus nodes. The number of outsiders should be a constant in this period. The second interval is transition period so that the number of outsiders presented a linear growth trend. The third interval is growth period. At this time, the network scale is larger, so the stability is less affected by a small number of nodes. The number of outsiders should be selected according to the proportion in this period. There are three parameters that need to be determined in the scheme: the threshold of transition period, the threshold of growth period and the ratio of growth period. Eq. (2) shows the constraint condition of three parameters to keep the number of consensus nodes monotonous, where t_1 is the transition period threshold, t_2 is the growth period threshold and r is the ratio of growth period.

$$\begin{cases} t_1 \bmod 2 = 0 \\ t_2 = \dfrac{t_1 - 1}{1 - r} \end{cases} \tag{2}$$

Fig. 6 is an implementation of the scheme for confirming the number of outsiders which ensures the number of consensus nodes is odd after removing outsiders. On the left side of Fig. 6, the abscissa is the total number of consensus nodes and the ordinate is the number of outsiders. On the right, the abscissa is also the total number of consensus nodes and the ordinate is the number of nodes participating in consensus process.



(a) left (b) right

Fig. 6. The number of outsiders

Leader Election Optimization. In the countdown mechanism of Raft, the timer's time is randomly generated between timeout and twice the timeout. We model the identity of leader as a random variable in leader election process. The random variable in the leader election model of Raft has the largest information entropy. For networks with large differences in node performance, too much randomness will cause greater risk of leader election. Therefore, the countdown mechanism is optimized through node status table.

Based on the measurement mechanism of outsider election algorithm, the variant equation in leader election process shown in Eq. (3) increases log synchronization progress of nodes. a is computing performance value and b is the score of blockchain number maintained by consensus nodes. a and b are the same as in the measurement mechanism. $\hat{S}_b$ is a fine-tuned score about business importance. e is a score about log synchronization. $u(e)$ is a step function of e. α_1, α_2, β_1 and β_2 are weights.

$$\begin{aligned} \widehat{Score} &= u(e) \times \frac{\alpha_1 a + \alpha_2 b + \beta_1 \hat{S}_b + \beta_2 e}{\alpha_1 + \alpha_2 + \beta_1 + \beta_2} \\ u(e) &= \begin{cases} 1, & e \geq 0 \\ 0, & e < 0 \end{cases} \\ a &= \frac{S_p}{P_{max}} \\ b &= \frac{1}{2}\left[\tan h \frac{4}{C_{max}} \left(\frac{C_{max}}{2} - n_c \right) + 1 \right] \end{aligned} \tag{3}$$

Because the probability of an outsider becoming the leader is zero, the process of calculating business importance scores of nodes has no need to consider outsiders. Adjusted business importance evaluation $\hat{S}_b$ is equal to the business score S_b over the total business score excluding outsiders. In Raft, leader completely controls the log synchronization process of consensus nodes. Therefore, the value of e is calculated by leader and is sent to the specific follower through log-append messages. The process for determining the value of e from log consensus progress is shown in Fig. 7. If log synchronization score of a node is equal to zero, it means that it is currently at an initial stage where consensus cluster has just started. There are no log entries that have reached consensus in this period. But if log synchronization score of a node is less than zero, it means that the node cannot be elected as the leader temporarily, because it does not contain the latest log entries that have been on blockchains. This function is implemented by the step function $u(e)$, which can set $\widehat{Score}$ of these nodes to zero.

We use Eq. (4) to apply the normalized $\widehat{Score}$ to timer generation process. It can reduce the information entropy of the random variable about leader identity, so that a better node has a greater probability of serving as the leader.

$$timer = Timeout + (1 - \widehat{Score}) \times Timeout \tag{4}$$

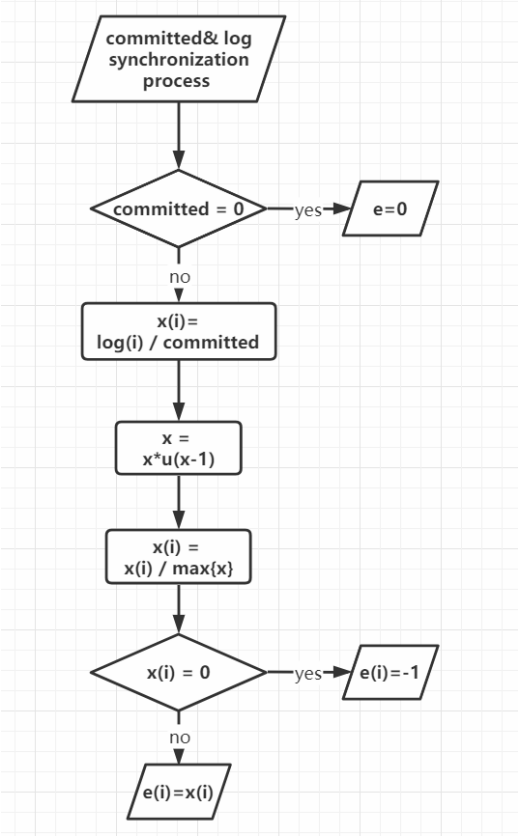


Fig. 7. Flow chart of determining the value of e

3.3 Consensus Process

In BARaft, consensus process includes four parts: node status broadcasting, leader election, outsider election and log replication.

Node Status Broadcasting. This process occurs in the initial stage that a blockchain has just been established. There are no blocks and outsider identity information on the chain. Therefore, all consensus nodes will become initiator state, which sends node status broadcast messages at regular intervals. The broadcast message contains node status information of the sender. After receiving a broadcast message, the node will save data of the message into its own node status table if there is no outsider identity information recorded locally. But if there is already outsider identity information, it will reply a message, which contains outsider identity information in the current blockchain. The initiator can extract outsider identities from the message, and then transit to outsider state or follower state. Otherwise, the initiator will automatically transit to follower state at the end of countdown.

Leader Election. The process of leader Election is similar to Raft with only two differences: first, an optimized countdown mechanism is used in leader election process; second, candidate election for a leader does not require votes from outsiders.

Outsider Election. After a leader is elected, if there is no outsider identity information on the blockchain, outsiders will be elected according to node status table. Through the outsider election algorithm mentioned in section 3.2, leader can evaluate the performance of all nodes and elect a certain number of nodes to participate in the consensus process of blockchain. After outsider election is completed, the leader will broadcast messages containing identity information of outsiders in the network. Other nodes will automatically transit their state to follower or outsider when receiving the messages.

Log Replication. Log replication process is managed by leader. The leader processes all transactions from clients in a consensus cluster. Different from Raft, leader does not rely on outsiders to update consensus progress in BARaft.

4 Analysis

In this section, we analyze the efficiency and availability of BARaft. All proofs here have a premise that the consensus efficiency of BARaft is higher than that of traditional Raft.

4.1 Performance Analysis

In this part, we analyze the relationship between the countdown mechanism with low information entropy and leader election, and then study its impact on the efficiency of Raft. For simplicity, suppose that network connections among nodes are in good condition, there are three nodes in consensus cluster including one leader and two followers respectively denoted by $node_1$ and $node_2$, the $\hat{S}_b$ value of $node_1$ is greater than that of $node_2$, and this value of the leader is zero. To control the number of variables, we assume that the values of *a*, *b*, and *e* of $node_1$ and $node_2$ in Eq. (3) are equal.

The measurement score of $node_1$ is

$$\widehat{Score_1} = \frac{\alpha_1 a + \alpha_2 b + \beta_1 \hat{S}_{b1} + \beta_2 e}{\alpha_1 + \alpha_2 + \beta_1 + \beta_2}, \tag{5}$$

and the score of $node_2$ is

$$\widehat{Score_2} = \frac{\alpha_1 a + \alpha_2 b + \beta_1 \hat{S}_{b2} + \beta_2 e}{\alpha_1 + \alpha_2 + \beta_1 + \beta_2}. \tag{6}$$

By combining Eq. (5) and Eq. (6), we obtain

$$\Delta\widehat{Score} = \widehat{Score_1} - \widehat{Score_2} = \frac{\beta_1 \times (\hat{S}_{b1} - \hat{S}_{b2})}{\alpha_1 + \alpha_2 + \beta_1 + \beta_2}. \tag{7}$$

We denote that

$$n = \frac{\beta_1}{\alpha_1 + \alpha_2 + \beta_1 + \beta_2} \tag{8}$$

And

$$\Delta\hat{S}_b = \hat{S}_{b1} - \hat{S}_{b2}, \tag{9}$$

so we have

$$\Delta\widehat{Score} = n \times \Delta\hat{S}_b. \tag{10}$$

By combining Eq. (4) and Eq. (10), we obtain

$$\Delta T = Timeout \times \Delta\widehat{Score} = T \times n \times \Delta\hat{S}_b, \tag{11}$$

where $T = Timeout$ and *t* is the countdown time during leader election.

After the leader crashes and stops sending heartbeat messages and log-append messages, two followers will start their countdown mechanism. Because of the different start time and duration of the timer, the probability of nodes elected as the leader will be different. The starting process of timer in the network can be modeled as a *poisson* process with parameter λ indicating the number of nodes whose countdown is started during unit time. λ will be affected by network conditions, and the overall network which is more stable has larger λ. We can assume that the time when leader sent the last batch of messages before crashing is the initial time in the *poisson* process.

Noted that i_1 is the starting time of the first timer after the leader crashes and i_2 is the starting time of the second timer. $X = i_2 - i_1$. Because time interval sequences of the *poisson* process obey the exponential distribution with parameter λ. So we have $X \sim \exp(\lambda)$. Only when X is greater than Δt and

the timer of $node_2$ starts at i_1, $node_2$ can become candidate first and be elected as the leader successfully. Therefore, we have

$$P(node_2 -> leader) = \frac{1}{2} \times P(X > \Delta t) = \frac{1}{2e^{\lambda Tn\Delta \hat{S}_b}}, \tag{12}$$

And

$$P(node_1 -> leader) = 1 - \frac{1}{2e^{\lambda Tn\Delta \hat{S}_b}}, \tag{13}$$

Since the duration of timer is randomly generated, the probability of nodes elected as the leader is the same in Raft as shown in Eq. (14).

$$P'(node_1 -> leader) = P'(node_2 -> leader) = \frac{1}{2} \tag{14}$$

If clients send transactions to followers, followers must forward the messages to the leader. Therefore, the business value of a node can represent the probability that consensus cluster does not need to forward transactions from clients when the node becomes the leader. According to these, we have the calculation formula of the forwarding probability when consensus cluster receives a transaction. Eq. (15) is the formula where P_i is the probability of $node_i$ elected as the leader.

$$P_1 \times (1 - \hat{S}_{b1}) + P_2 \times (1 - \hat{S}_{b2}) + \cdots + P_i \times (1 - \hat{S}_{bi}) + \cdots \tag{15}$$

Under the current assumptions, the probability that consensus cluster needs to forward one transaction in Raft is $\frac{1}{2}$, because the $\hat{S}_b$ of the leader is equal to zero. This probability in BARaft is

$$\hat{S}_{b2} + \frac{\Delta \hat{S}_b}{2e^{\lambda Tn\Delta \hat{S}_b}}. \tag{16}$$

Then we can obtain

$$ratio = \frac{P_{Raft}}{P_{BARaft}} = \frac{e^{\lambda Tn\Delta \hat{S}_b}}{2\hat{S}_{b2}e^{\lambda Tn\Delta \hat{S}_b} + \Delta \hat{S}_b}. \tag{17}$$

ratio-1 is the percentage of transaction forwarding probability reduced by BARaft compared to Raft, as shown in Eq. (18).

$$ratio - 1 = \frac{\Delta \hat{S}_b e^{\lambda Tn\Delta \hat{S}_b} - \Delta \hat{S}_b}{2\hat{S}_{b2}e^{\lambda Tn\Delta \hat{S}_b} + \Delta \hat{S}_b} \times 100\%. \tag{18}$$

Since λTn is greater than zero, *ratio*-1 must be greater than zero, which indicates that BARaft can reduce transaction forwarding in a consensus cluster. If $\hat{S}_{b2}$ tends to zero to enlarge the business score gap between $node_1$ and $node_2$, *ratio*-1 will tend to $e^{\lambda Tn} - 1$. Assuming that λT is equal to 1 and n is equal to $\frac{1}{4}$, it can be calculated that *ratio*-1 is 28.4%, which means that compared with Raft, the transaction forwarding can be effectively reduced by 28.4%. In addition, as network scale increases, the performance optimization will be more significant.

In addition to reducing the amount of forwarded transactions in the network, the optimized countdown mechanism can also increase the leader election probability of a node with better performance, which makes better node undertake higher computational pressure of the leader. It can alleviate the bottleneck of consensus efficiency. By setting the $\widehat{Score}$ of nodes whose log consensus progresses have not reached the common consensus progress to zero, the probability of these nodes elected as the leader is greatly reduced. It can reduce the occurrence of leader election redundancy causing the unavailability period of

consensus cluster to be longer and consensus efficiency to deteriorate.

4.2 Availability Analysis

The availability of BARaft depends on the stable work of outsiders. The following will analyze the availability through several specific scenarios.

Scenario 1. The initialization questions of node status table when restarting consensus network.

When node status table of a consensus node is first started, the internal data will be set to an initial value, which will be updated when initiators broadcast node status information. In addition, the node will make a local snapshot mapping for the data in node status table at certain points in the process of running. Therefore, if the network is restarted, the values in node status table will not be set as they were on the first startup. The previously saved node status data will be imported from local snapshot and nodes will convert to corresponding stable state. This setting can effectively use the historical data of node status table and reduce unnecessary data interaction in the network.

Scenario 2. Outsider becomes candidate.

After outsider election process finished, the scale of consensus network will become smaller because the blocks stored in outsiders are not be considered by other nodes in a consensus network. Therefore, the progress of block synchronization in outsiders may not reach the consensus progress in the network. When network is restarted, there might be a situation where a node who should be outsider is converted to follower or candidate because there is no block in this node. If an outsider becomes candidate by mistake, it will send vote messages to the other nodes. And at this time, more than half of nodes in the network have the identity information about outsiders. These nodes will check the data source when they receive voting messages. If sender is an outsider, they will send a rejection vote and send a message containing the identities of outsiders. So if the candidate receives the messages finding that it should be an outsider, it will change its state to outsider. Since more than half of nodes in the network must have the identity information of outsiders, it will be impossible for outsiders to become a leader. In addition, in order to prevent outsiders from becoming a candidate, a directional reply mechanism about identity information of outsiders is set up in initiator and follower state, which can basically ensure that outsiders cannot be converted to candidate state.

Scenario 3. The influence of configuration messages about deleting consensus nodes on outsiders.

Consensus cluster is a dynamic network and clients can adjust node configurations in the network by sending configuration messages. After outsider election process finished, the identity information of outsiders will be saved in every log entry, and then will be written into the metadata of every block. When a client sends the configuration message about deleting one consensus node, if it is an outsider that is removed, each node will adjust the locally stored identity information of outsiders. Therefore, the leader will use the new identity information of outsiders in the later log consensus process, and when generating a new block, the updated outsider identities will also be written. By binding outsider identity information to the blockchain, it can effectively solve the impact of node dynamic changes on identity information of outsiders.

5 Evaluation and Result

In this section, we set up a comparative experiment to study the impact of the transaction forwarding process on the overall algorithm performance in Raft. Then we analyze the transaction throughput of BARaft by comparing with Raft.

5.1 Experimental Setting

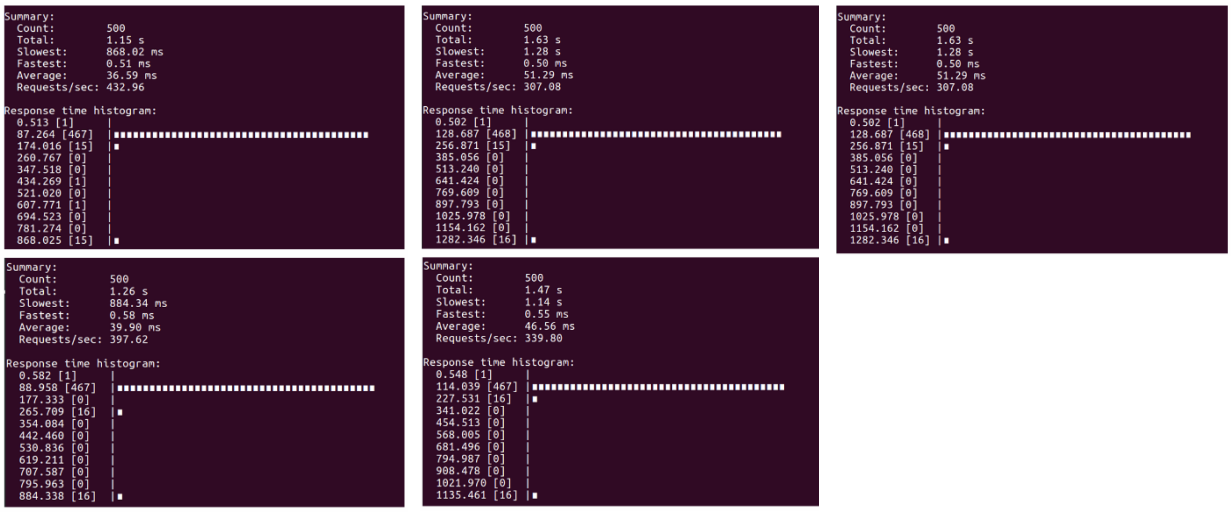
The server machines was 2.3GHz Intel(R) Core(TM) PC with 8GB RAM, and ran ubuntu 16.04 LTS operating system. The experiment was implemented on the Hyperledger Fabric which is an open source project [11], and used Raft in the Hyperledger Fabric 1.4.3 as a comparison. BARaft was written in the Golang language and implemented the call interface of Fabric enabling it to be successfully embedded in Hyperledger Fabric 1.4.3. Otherwise, we built a set of servers with consensus algorithm running on the bottom layer to test the transaction throughput by developing Fabric chaincode [11] and implementing Node interface and web backend. The experiment used Wrk to test the transaction throughout of the

entire server based on Http protocol and used Ghz to separately test the performance of consensus algorithm at the bottom of Fabric based on Grpc.

5.2 Transaction Forwarding Analysis

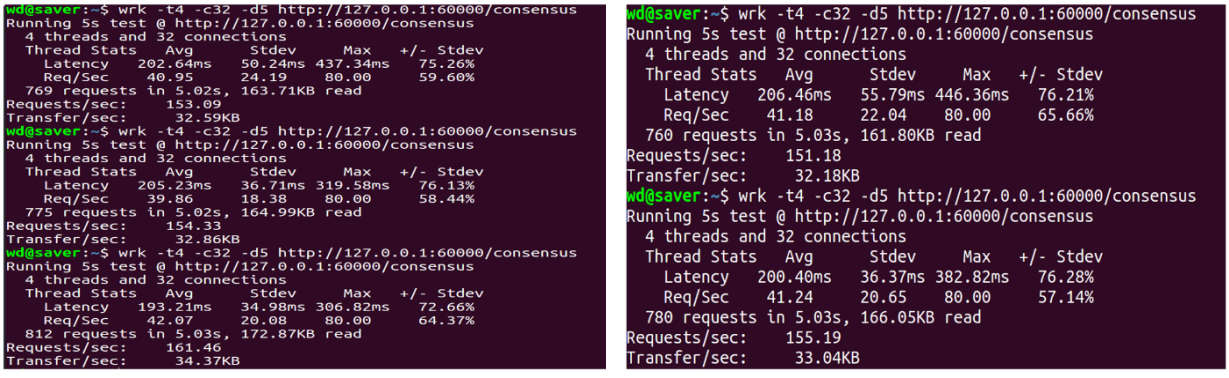
From section 4.1, we obtain that the improvement in business efficiency of BARaft is mainly thanks to the reduction in forwarded transactions, which is caused by optimizing the countdown mechanism. Therefore, a comparative experiment was designed to analyze the impact of transaction forwarding process on the performance of Raft. The experiment used Wrk and Ghz for performance evaluation respectively. By modifying client code, we can easily determine the destination of the transaction. In our comparative experiment, there were two sets of tests: in the first set of tests, clients sent all transaction messages to the leader, and in the second set of tests, transaction massages were sent to follower. Every test set used two methods for performance evaluation, respectively Wrk and Ghz, and five independent measurements were conducted based on each method. There were five nodes in the Raft consensus cluster.

Fig. 8 shows five performance evaluation results using Ghz and Fig. 9 shows results using Wrk in the first set of tests. For comparison, Fig. 10 shows five evaluation results using Ghz and Fig. 11 shows results using Wrk in the second set of tests.



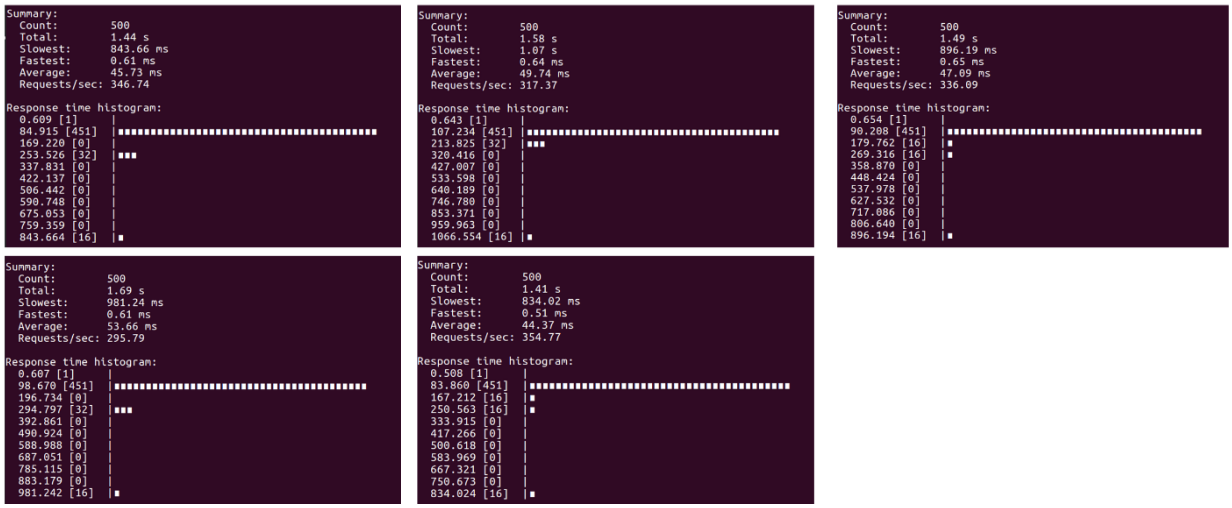
(a) rs1-rs2 (a) rs3-rs4 (c) rs5

Fig. 8. Results using Ghz in the first set of tests



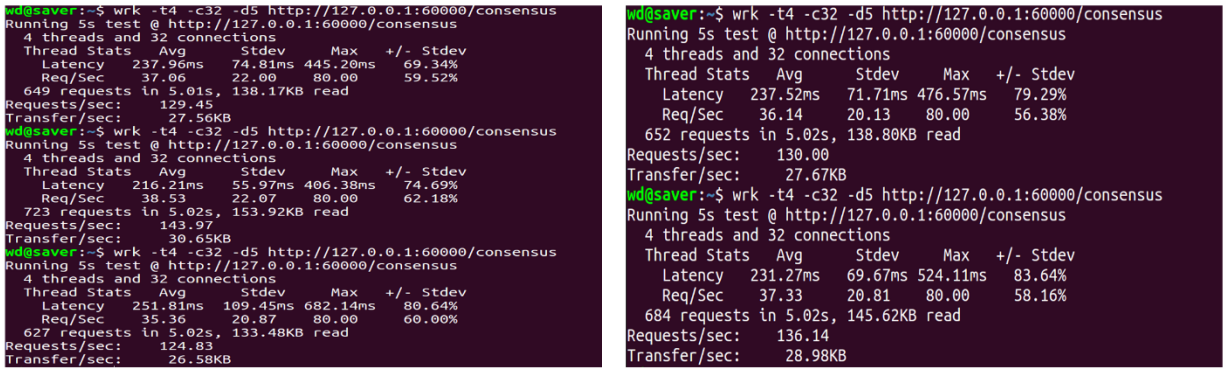
(a) rs1-rs2 (a) rs3-rs4

Fig. 9. Results using Wrk in the first set of tests



(a) rs1-rs2 (b) rs3-rs4 (c) rs5

Fig. 10. Results using Ghz in the second set of tests



(a) rs1-rs3 (a) rs4-rs5

Fig. 11. Results using Wrk in the second set of tests

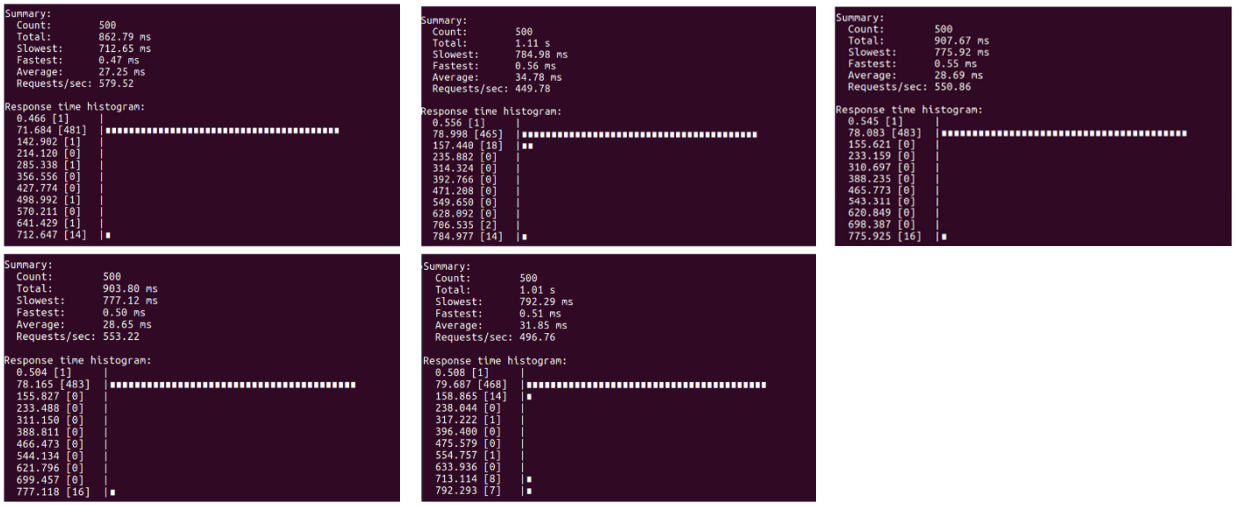
By calculating and comparing the average of the five measurement results in each figure, we can get Table. 1, which shows that the transaction throughput of no transaction forwarding is 10.56% higher than that of transaction forwarding. Therefore, by reducing the amount of forwarded transactions in a consensus cluster, the efficiency of consensus algorithm can be effectively increased.

Table 1. Comparisons on two sets of tests

	Ghz (tps)	Wrk (tps)	Mean (tps)
submit	330.15	132.88	231.52
not submit	356.91	155.05	255.98
increase	8.10%	16.68%	10.56%

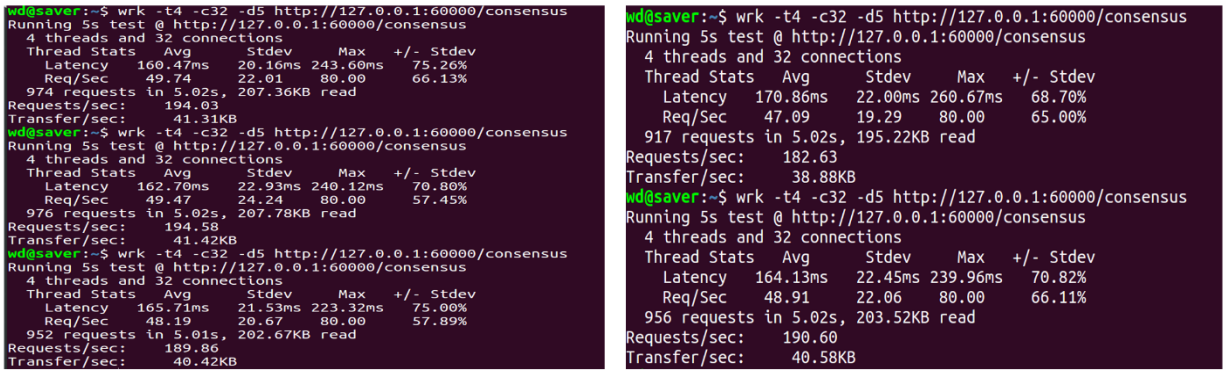
5.3 BARaft Algorithm Evaluation

In this section, we implemented BARaft based on the Hyperledger Fabric and evaluated the performance of the consensus algorithm using Wrk and Ghz respectively. There were the same five nodes as in section 5.2. Fig. 12 shows five performance evaluation results of BARaft using Ghz and Fig. 13 shows results using Wrk.



(a) rs1-rs2 (b) rs3-rs4 (c) rs5

Fig. 12. Evaluation results of BARaft using Ghz



(a) rs1-rs3 (b) rs4-rs5

Fig. 13. Evaluation results of BARaft using Wrk

By combining the evaluation results shown in Table. 1, Fig. 12 and Fig. 13, we obtain Table. 2, which shows transaction throughput of BARaft can be increased by 46.95% compared to Raft with five nodes in a consensus cluster. The throughput improvement is mainly thanks to the reduction in the scale of the consensus cluster and the account of forwarding transactions. The reduction in the scale of consensus cluster can fasten the consensus process to be completed and the business adaptability of BARaft can decrease the forwarding process of transactions.

Table 2. Comparisons between BARaft and Raft

	Ghz (tps)	Wrk (tps)	Mean (tps)
Raft/submit	330.15	132.88	231.52
Raft/not submit	356.91	155.05	255.98
Raft/mean	343.53	143.97	243.75
BARaft	526.03	190.34	358.19
Increase	53.12%	32.21%	46.95%

6 Conclusion

Aiming at improving the consensus efficiency of Raft, the paper investigated a variety of existing Raft improvements. We proposed a Raft-like consensus algorithm to achieve the business adaptability, which is named BARaft. It enhances the countdown mechanism to solve the problems of high randomness in leader election process and introduces node status table to eliminate the gap between consensus algorithm and business. By theoretical analysis, the improvement of countdown mechanism can reduce forwarded transactions, thereby improving the business efficiency of a consensus cluster.

BARaft can be used in the applications of permissioned blockchain. It can meet the needs of multiple blockchains in an alliance, because of the considerations about the attributes of consensus nodes. Through comparative experiments, Raft without submitting had a 10.56% improvement than Raft with submitting, and BARaft had a 46.95% increase in transaction throughput than Raft. BARaft can only tolerant Byzantine fault and sacrifices security to improve consensus efficiency because of the reduction in the scale of consensus cluster. In the future, we will explore how to reduce the parameters of node status table through adding some constraints, and how to optimize the update and maintenance of node status table.

Acknowledgements

This work is supported by National Key R&D Program of China (2018YFC0832300;2018YFC0832303).

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system. <https://bitcoin.org/bitcoin.pdf>, 2008 (accessed 29.03.20).

[2] Y. Yun, S. Xu, Overview of blockchain technology, Journal of Quality and Certification 5(2020) 56-58.

[3] G.-T. Nguyen, K. Kim, A survey about consensus algorithms used in blockchain, Journal of Information Processing Systems 14(1)(2018) 101-128.

[4] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, <https://decred.org/research/king2012.pdf>, 2012.

[5] M. Du, X. Ma, Z. Zhang, X. Wang, Q. Chen, A review on consensus algorithm of blockchain, in: Proc. 2017 IEEE International Conference on Systems, Man, and Cybernetics, 2017.

[6] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: Proc. 2014 USENIX Annual Technical Conference, 2014.

[7] C. Copeland, H. Zhong, Tangaroa: a byzantine fault tolerant raft, <https://www.scs.stanford.edu/14au-cs244b/labs/projects/copeland_zhong.pdf>, 2016.

[8] M. Castro, B. Liskov, Practical byzantine fault tolerance, in: Proc. the Proceedings of the Third Symposium on Operating Systems Design and Implementation, 1999.

[9] R. Wang, L. Zhang, Q. Xu, H. Zhou, K-bucket based raft-like consensus algorithm for permissioned blockchain, in: Proc. 2019 IEEE International Conference on Parallel and Distributed Systems, 2019.

[10] Y. Gao, H. Li, Y. Li, B. Liu, X. Wang, H. Ruan, Using tla+ to specify leader election of raft algorithm with consideration of leadership transfer in multiple controllers, in: Proc. 2019 IEEE International Conference on Software Quality, Reliability and Security Companion, 2019.

[11] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S.W. Cocco, J. Yellick, Hyperledger fabric: A distributed operating system for permissioned blockchains, in: Proc. the Thirteenth EuroSys Conference, 2018.

Appendix: Variable Table

Table. 3 shows the description of all variables appearing in this paper.

Table 3. Variable description

Variable	Description	Variable	Description
S_p	Value of computing performance	P_{max}	The maximum of S_p
n_c	The number of blockchains	C_{max}	The maximum of n_c
S_b	Business value	α_1	Weight of computing performance
α_2	Weight of blockchain number	β_1	Weight of business
$Score$	Measurement score	t_1	Transition period threshold
t_2	Growth period threshold	r	Ratio of growth period
a	Normalized value of computing performance	b	Normalized business value
$\hat{S}_b$	fine-tuned score about business value	e	Value about log synchronization
β_2	Weight of log synchronization	$u(e)$	Step function of e
$\widehat{Score}$	Enhanced measurement score	$Timeout$	A constant
Timer	Time of a timer	$\widehat{Score_1}$	The $\widehat{Score}$ of $node_1$
$\widehat{Score_2}$	The $\widehat{Score_2}$ of $node_2$	$\hat{S}_{b1}$	The $\hat{S}_b$ of $node_1$
$\hat{S}_{b2}$	The $\hat{S}_b$ of $node_2$	T	$Timeout$
$\Delta\widehat{Score}$	$\widehat{Score_1} - \widehat{Score_2}$	$\Delta\hat{S}_b$	$\hat{S}_{b1} - \hat{S}_{b2}$
n	$\beta_1/(\alpha_1+\alpha_2+\beta_1+\beta_2)$	Δt	Time difference between $node_1$ and $node_2$
i_1	Starting time of the first timer	i_2	Starting time of the second timer
X	$i_2 - i_1$	λ	The parameter of poisson process
$ratio$	P_{Raft} / P_{BARaft}	P_{Raft}	Transaction forwarding probability in Raft
P_{BARaft}	Transaction forwarding probability in BARaft		