Advances in Intelligent Systems Research, volume 147
International Conference on Network, Communication, Computer Engineering (NCCE 2018)

Research and Implementation of Blockchain Warehouse Receipt Trading Platform Based on BFT

Jianfeng Li[1, a)], Zhihong Zhang[1, b)] and Yuehan Zhang[2, c)]

[1]*School of Information Engineering, Zhengzhou University, Zhengzhou 450000, China.*
[2]*Zhengzhou Esunny Information Technology Co, Zhengzhou 450008, China.*

[a)] lijianfeng0108@qq.com
[b)] iezhzhang@zzu.edu.cn
[c)] zhangyuehan@esunny.cc

Abstract. For the problems of excessive centralization in information flowing, low performance in business extension, and being difficult to guard against the risk of the bill market in the traditional warehouse receipt trading system, this paper proposes a warehouse receipt trading platform based on block chain technology. By introducing the BFT (Byzantine Fault Tolerance) mechanism into block chain technology, the platform implements a distributed consensus algorithm based on the improved BFT algorithm. Other technologies such as smart contract of Ethereal are used to optimize the algorithm. Compared with the traditional centralized system, the platform has the characteristics of decentralization, distributed consensus, trustworthiness, traceability, irreversibility, and fault tolerance. The experimental result shows that the consensus efficiency of the platform has obvious advantages compared with current block chain systems like Bitcoin and Ethereal.

Key words: Warehouse Receipt Trading, Block chain, Byzantine Fault Tolerance, Distributed Consensus, Smart Contract.

INTRODUCTION

In the traditional centralized warehouse receipt trading model, due to the need for a trusted third party to process and verify the information transfer between the two parties 1, there are many shortcomings of this system. First, the centralized system faces many security risks from the outside, once a hacker attacks a central node (or a central cluster) successfully, it can destroy the whole system; besides, the third party itself may exist some problems which will influence the transaction security, once the center makes a multiple payment, forges counterfeit money, modifies or deletes the database casually, the interests of transaction participants will be damaged.

Aiming at the problems of single point fault, third party trust, low security and non-traceability of data in the traditional centralized system, this paper builds the warehouse receipt trading platform by block chain 23, 4 technology which has the characteristics of distributed ledger, polycentrism consensus, time series data, traceability, irreversibility and trustworthiness 5. As the consensus algorithm determines the consistency of distributed ledger and the performance of the platform, this paper introduces Byzantine Fault Tolerance (BFT) 6 mechanism to the platform and improves it. In detail, this paper models the BFT state machine which provides warehouse trading consensus service, uses the block synchronization and signature algorithm which ensures the consistency of the ledger, and formulates smart contract 7, 8 which manages node dynamically.

Compared with the traditional centralized warehouse system, the block chain warehouse receipt trading platform based on the improved BFT algorithm has significant advantages. First of all, the decentered chain structure ensures that the warehouse receipt transactions recorded in the block cannot be tampered; secondly, the process and validation of trading is decentralized, negotiated and trust, the information transfer is done in multi nodes; besides, the consensus algorithm has fault-tolerant and dynamically manageable characteristics while maintaining the

consistency of node states. What's more, the experimental results show that the platform has obvious advantages over the Bitcoin and ethereal in terms of transaction efficiency (Transaction per Second, TPS).

SYSTEM MODEL

Architecture and Modules

The warehouse receipt trading platform based on the improved BFT consensus algorithm can be built in different entities (i.e. physical nodes) such as exchanges, banks and member companies. The overall architecture of the platform is shown in Fig. 1.

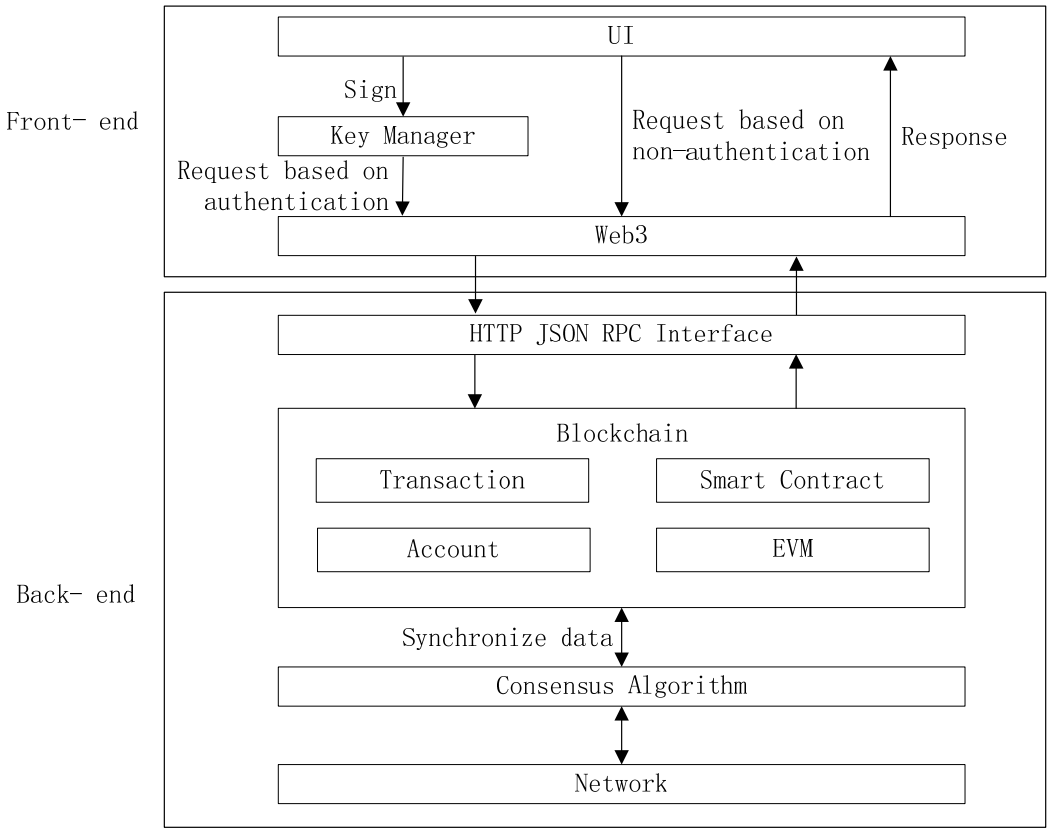

FIGURE 1. The overall architecture of the warehouse receipt trading platform.

The platform is divided into two modules: the back-end includes Network, Consensus Algorithm, Block chain and RPC; the front-end includes Web3, Key Manager and UI.

1) Network: it controls the connections of nodes, propagation of transactions, blocks, consensus and other data based on P2P network. In the network, node is divided into two types (i.e. consensus node and common node), only the first one runs the consensus algorithm.

2) Consensus Algorithm: it uses the BFT mechanism which is improved in the consensus algorithm to create a block containing ethereal transactions which is recognized and executed by EVM and negotiate the block. In the algorithm, when there are F illegal nodes, if the total number of consensus nodes (written N) is equal or greater than 3*F+1, and the problem of the Byzantine generals 9 can be solved, the fault tolerance is 33%, and the activity and correctness is proved strictly.

3) Block chain: account is divided into two types (i.e. external and internal account), they are used to sign the ethereal transactions including warehouse receipt trading relevant data and maintain the data executed by EVM virtual machine separately. Transaction is divided into two types (i.e. deploy and call transaction), they are used to

deploy and call contract (e.g. warehouse receipt trading and node management contracts) separately. Smart contract is used to define the logic of warehouse receipt trading business and other operations, its execution environment is EVM.

4) HTTP JSON RPC Interface: it defines the access interfaces of block chain, transmits JSON format messages like ethereal transactions based on HTTP protocol.

5) Web3: it is a middleware that encapsulates JSON RPC API to interact with the block chain by requesting a back-end RPC service.

6) Key Manager: it is an ethereal wallet called Metalmark, this module can store account keys and sign transactions when clients want to operate the warehouse receipts. This module is not needed when clients just retrieve the data of the back-end.

7) UI: it includes listing, delisting and other pages, the listing and delisting operations can trigger the signing operation of ethereal wallet.

Trading Process

The warehouse receipt listing and delisting operations are negotiated through the consensus algorithm, the process is as follows.

1) Sellers choose the warehouse receipts and input a price and a quantity, the front-end calls the back-end to retrieve the quantity of the corresponding warehouse receipts and judges whether the quantity is sufficient. If not sufficient, the operation is done; else, the next operation is followed.

2) The front-end creates an ethereal transaction which includes the encoded data of the corresponding warehouse receipts and broadcasts the transaction to the back-end nodes after signing.

3) The back-end packages ethereal transactions into a block. Then, the block is processed, verified and voted through nodes. In this process, consensus nodes can reject the block and vote against if they don't approve the transactions in the block. After that, a block including transactions which is approved by majority consensus nodes is committed and synchronized in consensus and common nodes, the transactions are executed correctly, and the warehouse receipts are listed.

4) Buyers choose the listed warehouse receipts and input a quantity. Then, the funds of the buyers will be judged. If not sufficient, the operation is done; else, the next operation is followed.

5) An Ethereal transaction is created and signed by the front-end. Then, the transaction is broadcasted to the back-end.

6) The back-end packages ethereal transactions into a block. After negotiating, the block including transactions which is approved by majority consensus nodes is committed and synchronized in consensus and common nodes, the delisting operation is done, the funds and the warehouse receipts recorded in contract are transferred.

CONSENSUS MODEL

Algorithm Description and Definition

In the process of warehouse receipt trading, the consensus and synchronization of block occurs in consensus nodes, and the synchronization of block occurs in common nodes, it means that common nodes only accept blocks negotiated consistently from consensus nodes. Besides, in this platform, the type of nodes is different, and the nodes can be changed and managed dynamically. In this section, we describe the BFT consensus algorithm and the formulas used in the consensus process, and we will define the state transition process of consensus and above improvement in following sections.

First of all, what is the BFT consensus algorithm? It is a state machine replication algorithm 1011, service of blockchain is modeled as a state machine, and the state machine replication is shared and synchronized in different nodes. A copy of each of the state machines have preserved the status of the service, and also realized the operation of the service.

The consensus consultation process consists mainly of 4 states: proposal, vote, ack and commit. In the proposal state, the proposal node which is selected from consensus nodes packages transactions into the block and send it to other consensus nodes; in the vote state, consensus nodes vote the block, and then broadcast the voting results; in the ack state, the consensus nodes vote again, and then broadcast the results to other nodes; in the final state, the nodes

make a decision according to the results of ack votes, that is committing the block (enter the commit state) or creating a new block (enter the proposal state). The full states form a special state called new height, it means that the nodes negotiate a new block of different height after committing previous block, and the first three states form a state called new round, it means that every new height state includes at least one process of proposing and voting.

Secondly, we define the key concepts and the messages transmitted in the consensus process. Every node must verify them while receiving them.

The state (written $\mathbb{S}$) of state machine is defined by three tuples:

$$\mathbb{S} = (H, R, S) \tag{1}$$

In the formula (1), H represents the block height, R represents the consensus round, and S represents the state type ({New_Height, New_Round, Proposal, Vote, Vote Wait, Ask, Ask Wait, and Commit}).

The calculation formula of proposal node index (written i) is defined as follows:

$$i = H + R \ mod \ n \tag{2}$$

In the formula (2), H represents the block height, R represents the consensus round, and n represents the number of consensus nodes.

The consensus messages are defined as MS Proposal, Mogen and MS Timeout:

$$MSG_{Proposal} = (H, R, B, i, Sig_{Proposal}) \tag{3}$$

In the formula (3), H represents the block height, R represents the consensus round, B represents the proposed block, i represents the index of consensus node, and SigProposal represents the proposal node's signature to the proposal data.

$$MSG_{Neg} = \begin{cases} (Type, H, R, V, i, Sig_B, Sig_{Neg}), & Type = VOTE, V = B \\ (Type, H, R, V, i, Sig_{Null}, Sig_{Neg}), & Type = VOTE, V = Null \\ (Type, H, R, V, i, Sig_{Neg}), & Type = ACK, V \in \{B, Null\}. \end{cases} \tag{4}$$

In the formula (4), Type represents the negotiation type (Type∈ {Vote, Ask}), H represents the block height, R represents the consensus round, V represents the voting result for the proposed block (V∈ {B, Null}), B represents the vote for "Yes", Null represents the vote for "No"), i represents the index of consensus node, Sib represents the consensus node's signature to the proposed block, Signal represents null signature, Signe represents the consensus node's signature to the negotiation data.

$$MSG_{Timeout} = (H, R, St) \tag{5}$$

In the formula (5), H represents the block height, R represents the consensus round, and St Represents the state of timeout ({St∈ Proposal St, Vote Wait St, Ask Wait St}).

State Transition

The process of consensus is defined in Fig. 2.

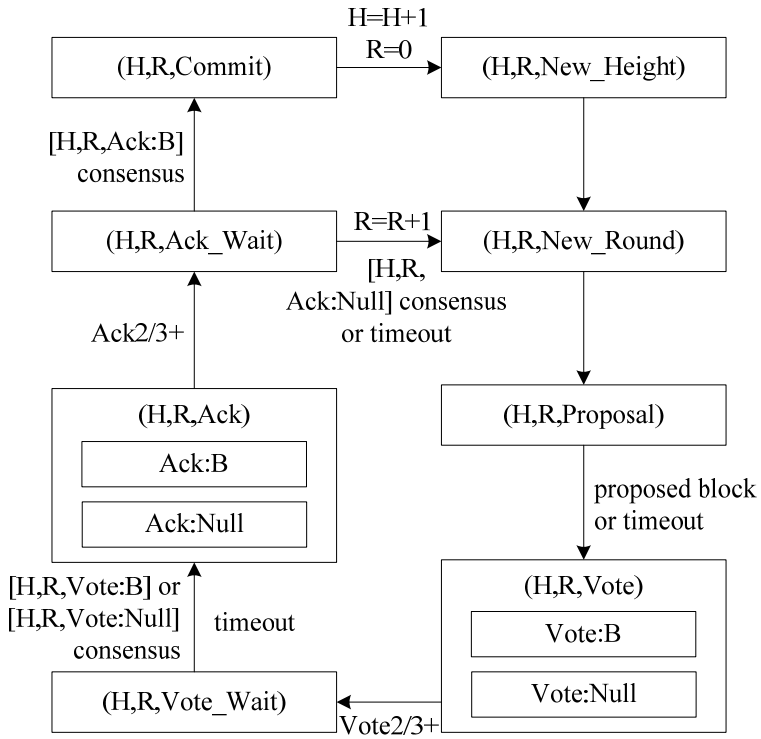

FIGURE 2. The process of state transition.

1) New Height state: update the height of H and enter the new Round state.
2) New Round: update the round R according to the previous state, update the proposal node index is, and enter the Proposal state.
3) Proposal state: if it is the proposal node, then create the proposal block B including transactions, and broadcast the MS Proposal; if not, then set the timeout through the MS Timeout; when getting the MS Proposal or timeout, enter the Vote state.
4) Vote state: if the proposed B is valid, then vote [Vote: B], and broadcast the Mogen; otherwise, vote [Vote: Null], and broadcast the Mogen; when receiving 2/3 [Vote: *] votes (i.e. 2/3 any Vote votes]), enter the Vote Wait state.
5) Vote Wait state: set timeout through the MS Timeout, when getting 2/3 [Vote: B/ Null] votes (i.e. consensus of B or Null) or timeout, enter the Ask state.
6) Ask state: if the number of [Vote: B] votes is not less than 2/3 the whole votes, then vote [Ask: B], and broadcast the Mogen; otherwise, vote [Ask: Null], and broadcast the Mogen; when receiving 2/3 [Ask: *] votes (i.e. 2/3 any Ask votes), enter the Ask Wait state.
7) Ask Wait state: set timeout through the MS Timeout, when getting 2/3 [Ask: B] votes (i.e., consensus of B), enter the Commit state; when getting 2/3 [Ask: Null] votes (i.e. consensus of Null) or timeout, enter New Round state.
8) Commit state: submit B and execute transactions in B in block chain, then enter the New Height state.

Algorithm Improvement

In this section, we describe the improvement of the BFT algorithm to satisfy the characteristics of block synchronization and node management.

The original BFT algorithm runs on the same block chain height, once the situation of downtime and reconnection happens, the consensus heights of different consensus nodes will become different. Besides, the common nodes cannot get blocks because consensus process happens in consensus nodes. To solve the problem, we

introduce the block synchronization mechanism, and we also introduce signatures of consensus nodes in order to ensure that the blocks come from real consensus nodes.

1) Synchronization: after finishing handshake, two nodes send their newest information of block hash, height and others. Every node compares the information with local block chain data, the node whose height is lower will request the blocks, and then, another node will response. Besides, after committing a new height block by consensus nodes in the Commit state of state machine, the block will be broadcasted to common nodes.

2) Signature: first of all, the key information (e.g. public key) of consensus nodes are recorded in block header, it ensures that the set of nodes participating in each round of consensus negotiation cannot be tampered. Secondly, the list of node index is and signature Sib is stored while committing the block in the Commit state of state machine, although the lists of different nodes can be different, the size of the list strictly satisfies the number of votes. Thirdly, the list of indexes and signatures is broadcasted while sending the block to common nodes, and they can verify the signatures.

As for node management, we use smart contract to manage node dynamically. Smart contract stipulates node information (e.g. public key, node type, IP address, port). The data in contract are agreed and cannot be tampered, once contract is executed by EVM while committing a block, the platform will update the information of consensus and common nodes. The process of deploy and call of node management contract is shown in Fig. 3.

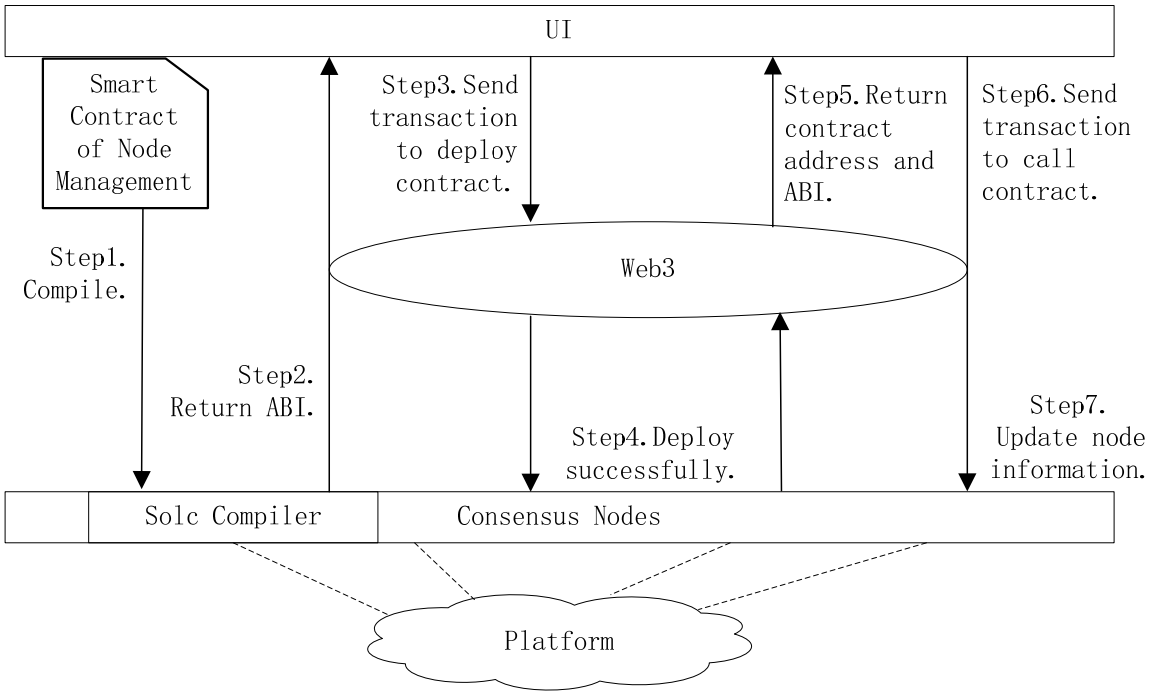

FIGURE 3. The process of node management.

As shown in Fig. 3, a smart contract of node management is compiled by Sold compiler firstly (step 1), after that, ABI (Application Binary Interface) returns (step 2), then we can deploy it to nodes by creating one ethereal transaction (step 3), after negotiating (step 4), the contract is associated with an account address (step 5), we can call it by creating another ethereal transaction and specifying input data satisfied with ABI (step 6). Once the latter transaction is recognized in the Commit state of state machine, the platform will update the information of nodes (step 7).

EXPERIMENT AND RESULT ANALYSIS

Experiment Environment

This paper tests the consensus efficiency (TPS) of the warehouse receipt trading platform in 4 virtual machines which play the roles of consensus nodes. The configuration of virtual machines is as follows: 4 core processor, Intel Xeon CPU E7-4830, 2.2GHz, 8G memory, Linux Red Hat 6.5 (Santiago) operating system.

Experiment Process

We firstly get the transaction data of blockchain height from 4420000 to 4434000 and from 480000 to 488000 in Ethereum and Bitcoin official browsers separately. The results are in Table 1.

TABLE 1. The distribution of block size in Bitcoin and ethereal.

System	Block Size	Number of Blocks	Proportion
ethereal	0-100	9732	69.51%
	100-200	3869	27.64%
	200-300	399	2.85%
	>300	0	0.00%
Bitcoin	0-1000	912	11.40%
	1000-2000	5042	63.00%
	2000-3000	2046	25.60%
	>3000	0	0.00%

The block size (number of transactions per block) is different in Ethereum and Bitcoin: the speed of block generation is controlled in 14 seconds per block in Ethereum, a single block contains a few transactions; the speed of block generation is controlled in 10 minutes per block in Bitcoin, a single block contains more transactions. Therefore, we test the TPS of the block size from 50 to 3000, and the final results are the mean values of the results of 20 experiments. The results are in Table 2.

TABLE 2. The result of consensus efficiency tested with different block size.

Block Size	TPS
50	85.4
100	128.2
200	157.5
300	173.9
500	194.9
1000	201.6
1500	201.9
2000	202.1
3000	203.2

As the size of most blocks is from 0 to 200 in ethereal and from 1000 to 2000 in Bitcoin, we test 50 and 100 times with corresponding block size and the results show that the TPS is from 60 to 160 and from 201 to 203 respectively. Besides, we calculate the TPS of ethereal and Bitcoin, they are from 0 to 20 and from 0 to 5. Compared with these data, this platform has a high performance. The results are shown in Fig. 4 and Fig. 5.

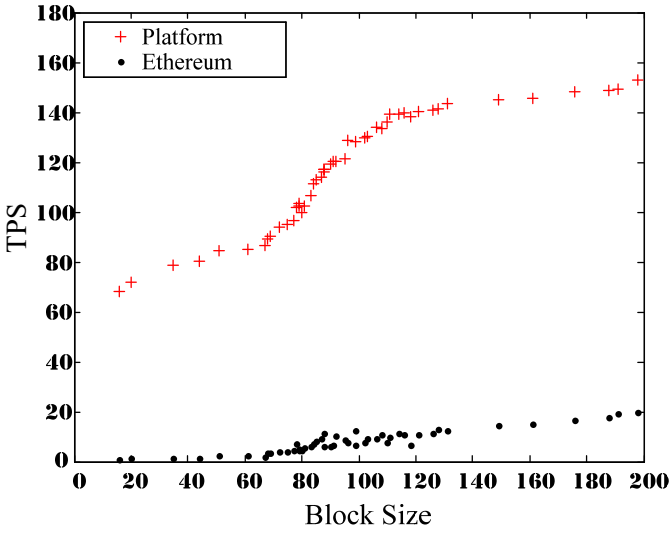

FIGURE 4. The consensus efficiency comparison between the platform and ethereal.

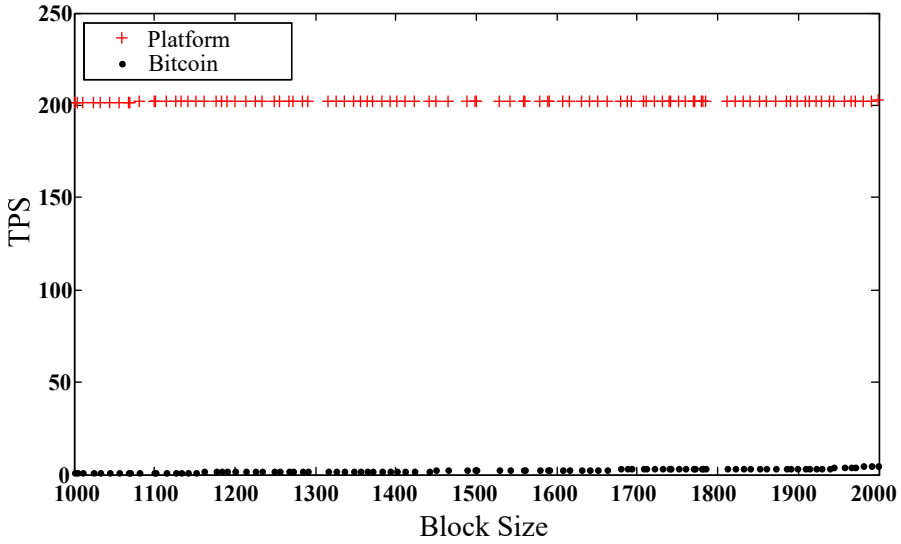

FIGURE 5. The consensus efficiency comparison between the platform and Bitcoin.

Experiment Result

The experimental results show that the consensus efficiency of the warehouse receipt trading platform become more efficient with increasing block size, when the block size is bigger than 1000, the efficiency tends to be steady. And compared with Ethereum and Bitcoin, it can be seen that the warehouse receipt trading platform built in this paper has obvious advantages for consensus efficiency.

SUMMARY

This paper constructs a warehouse receipt trading platform based on the improved BFT consensus algorithm which is mainly explained. The platform solves the problems of the traditional centralized warehouse receipt trading system, such as excessive centralization, single point fault, and third-party trust, transaction opaque and non-

traceability. By analyzing the consensus efficiency of the platform, we can conclude that the efficiency is significantly higher than that of mainstream block chain platforms like Bitcoin and ethereal.

Although the platform based on the improved BFT algorithm has a good efficiency, the problems of constraints of network bandwidth and smart contract execution efficiency still exist. In the follow-up study, we can further improve the efficiency by optimizing the smart contract and EVM virtual machine, and we can also introduce lightning network and partition technology into block chain.

REFERENCES

1. H T Mei, J Liu. Industry present situation, existing problems and strategy suggestion of blockchain. Telecommunications Science. Vol. 32 (2016) No. 11, p. 134-138.
2. Information on: bitcoin.org/bitcoin.pdf.
3. M D Pierro. What is the blockchain. Computing in science & engineering. Vol. 19 (2017) No. 5, p. 92-95.
4. Y Yuan, F Y Wang. Blockchain: The State of the Art and Future Trends. Act Automatic Sonica. Vol. 42 (2016) No. 4, p. 481-494.
5. P He, G Yu, Y F Zhang, et al. Survey on Blockchain Technology and Its Application Prospect. Computer Science. Vol. 44 (2017) No. 4, p. 1-7.
6. M Castro, B Liskov. Practical byzantine fault tolerance. Proceedings of the third symposium on operating systems design and implementation. New Orleans, 1999, p. 173-186.
7. Information on: github.com/ethereum/wiki/wiki/White-Paper.
8. L Luu, D H Chu, H Olickel, et al. Making Smart Contracts Smarter. Proceedings of the 2016 ACM Sigsac Conference on Computer and Communications Security. Vienna, 2016, p. 254-269.
9. L Lamport. The Byzantine Generals Problem. Acme Transactions on Programming Languages & Systems. Vol. 4 (2016) No. 3, p. 382-401.
10. L Lamport. Time, clocks, and the ordering of events in a distributed system. Communications of the Acme. Vol. 21 (1978) No. 7, p. 558-565.
11. F B Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. Acme Computing Surveys. Vol. 22 (1990) No. 4, p. 299-319.