



360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



# Analysis and Improvement of NEO's dBFT Consensus Mechanism

09月19, 2018

Zhiniang Peng from Qihoo 360 Core Security

NEO uses the dBFT consensus mechanism to achieve "Byzantine Fault Tolerance" between nodes in consensus system. The NEO Whitepaper says that if the number of malicious bookkeeping nodes is less than 1/3, the dBFT consensus mechanism can guarantee the security and availability of the system. However, our research found that the NEO's current implementation of dBFT mechanism only guarantees a consensus between honest bookkeeping nodes. There is no fork between the bookkeeping nodes does not mean that there will be no forks in the whole network. NEO's current implementation of the dBFT consensus mechanism does not provide fault tolerance for  $f = \lfloor (n - 1)/3 \rfloor$  for a consensus system consists of  $n$  bookkeeping nodes.

## 文章目录

- [Introduction to NEO dBFT Consensus Mechanism](#)
- [Byzantine General's Problem and Blockchain](#)
- [Details of the NEO Consensus Mechanism](#)
- [The Fork in dBFT Consensus Mechanism](#)
- [Impact](#)

## Introduction to NEO dBFT Consensus Mechanism

The NEO blockchain is a distributed intelligent contract platform. NEO implements a Delegated Byzantine Fault-Tolerant consensus algorithm that draws on the characteristics of some PoS (NEO holders need to vote on bookkeeping nodes) to use the least resources to protect the network from Byzantine failures and to make up for some issues of the PoS. dBFT provides fault tolerance for  $f = \lfloor (n - 1)/3 \rfloor$  for a consensus system consists of  $n$  bookkeeping nodes. This fault tolerance



360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



also realizes security and availability and can be applied to any network environment.

The NEO dBFT consensus mechanism can be found in detail in the NEO official consensus mechanism Whitepaper: <http://docs.NEO.org/en-us/basic/consensus/whitepaper.html>

## Byzantine General's Problem and Blockchain

Byzantium was the capital of the Eastern Roman Empire and used to locate in today's Istanbul, Turkey. Because of the vast territory of the Roman Empire at that time, for the purpose of defense, each army was separated far apart, and the generals could only rely on messenger to deliver the news. At wartime, all the generals and adjutants in the Byzantine army must reach a consensus to attack the enemy's camp based on whether there is a chance to win. However, the problem is there might be traitors and spies in the army who may interfere with the generals' decisions and disrupt the order of the entire army. When consensus is reached, the results do not represent the opinions of the majority. If the rebellion is already known, the problem that how the remaining loyal generals can reach a unanimous agreement by eliminating the influence of the renegade then becomes the well-known "Byzantine General's Problem". The pBFT algorithm is a classic algorithm for solving the Problems.

A blockchain is a decentralized distributed ledger system. It could be used for registration and issuance of digitalized assets, property right certificates, credit points and so on. It enables transfer, payment, and transactions in a peer-to-peer way. The blockchain technology was originally proposed by Satoshi Nakamoto in a cryptography mailing list, i.e. the Bitcoin. Since then, numerous applications based on the blockchain emerged, such as e-cash systems, stock equity exchanges and Smart Contract systems. A blockchain system is advantageous over a traditional centralized ledger system for its full-openness, immutability and anti-multiple-spend characters, and it does not



360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



rely on any kind of trusted third-party. However, like any distributed system, blockchain systems are challenged with network latency, transmission errors, software errors, security breaches, hacking, etc.. In addition, the decentralization characteristics determine that any participant in the system cannot be trusted, there may be malicious nodes, and data differences due to inconsistent interests of all parties.

To counter these potential errors, blockchain systems require an efficient consensus mechanism to ensure that each node has a copy of a recognized version of the total ledger. The traditional fault-tolerant method for certain problems does not completely solve the fault tolerance problem of distributed systems and blockchain systems. A universal cure-to-all fault tolerance solution is in need. NEO's dBFT consensus mechanism is an alternative to the Byzantine problem of distributed nodes in blockchains.

## Details of the NEO Consensus Mechanism

Byzantine fault-tolerant delegation (dBFT) was adopted as a consensus mechanism (<http://docs.NEO.org/en-us/basic/consensus/consensus.html>). There are two types of NEO nodes in the whole network: one is the bookkeeping node, which is responsible for the consensus communication with other bookkeeping node to generate new blocks; the other is the ordinary node, which does not participate in the consensus, but can verify and accept new blocks. The bookkeeping nodes are generated through voting by the entire network users. The idea behind the NEO's adopting dBFT algorithm is that the PBFT algorithm can solve the consensus problem of distributed nodes very well, however, the more the node participate, the worse the PBFT consensus performs.



360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



In NEO, a relatively small number of bookkeeping nodes are selected through voting to perform PBFT consensus and generate new blocks, and then the new blocks are released to the whole network to reach a consensus across the network. The consensus process for generating new blocks between NEO bookkeeping nodes is as follows:

- 1: There are two types of nodes in the network: Ordinary nodes and bookkeeping nodes. Ordinary nodes do not participate in the consensus, and the bookkeeping nodes participate in the consensus process.
- 2: Select the Speaker. The NEO speaker's production mechanism is based on the MOD operations of the current block height and the number of bookkeepers. The speaker is actually elected in order.
- 3: Node initialization. The speaker is the primary node, and the other bookkeeping nodes are the backup nodes.
- 4: After the block publishing condition is met, the primary node broadcasts a PrepareRequest.
- 5: When the backup nodes receive the request, they will validate the request. If it's validated, the backup nodes will sign the block and propose a broadcast PrepareResponse.
- 6: After the bookkeeping nodes receive the PrepareResponse, they will save the signature information, and count if the block has signed by more than two-thirds bookkeeping nodes. If yes, then send the block.
- 7: The nodes receive the block, and the PersistCompleted is triggered. Then initialization is started again.

In order to prevent malicious bookkeeping nodes or dishonest primary node and to ensure the security and reliability of the system, NEO proposed changeview mechanism to further enhance the security of dBFT. When node $i$  does not reach a consensus after a time interval of  $2^{v+1} \cdot t$ , or receives a proposal containing an illegal transaction, it begins the view replacement process:



360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



1. Given  $k = 1$ ,  $v_k = v + k$
2. Node  $i$  sends out a view change request  $\langle \text{ChangeView}, h, v, i, v_k \rangle$ ;
3. Once any node receives at least  $n - f$  identical  $v_k$  from different  $i$ , the View Change is completed. Set  $v = v_k$  and consensus starts;
4. If the view change has not been completed after the interval of  $2^{v+1} \cdot t$ , then  $k$  will increase and returns to step 2;

The Flow Chart of the NEO consensus is as follows:

## The Fork in dBFT Consensus Mechanism

The core idea of dBFT is to reach a consensus through the pBFT protocol and the elected bookkeeping nodes, thus generating a consensus across the network. This seems to be a good idea, but the derivation is not true because of the difference between dBFT and pBFT. In pBFT, the non-bookkeeping node (client node) needs to receive the same execution result of at least  $f+1$  bookkeeping nodes to obtain the result. In dBFT, the non-bookkeeping node needs to obtain a block signed by at least  $2f+1$  bookkeeping nodes. It cannot be taken for granted that the Byzantine fault tolerance of dBFT can be derived from pBFT. Proving the security of the dBTF requires rigorous proof. An important premise in analyzing the security of network protocols is that the network is a complex environment, and it cannot be guaranteed that the packets sent first will arrive first. This is why the network protocol is so complicated. Below we present two counter-examples, to prove that NEO's current implementation of dBFT consensus mechanism does not provide fault tolerance for  $f = \lfloor (n - 1)/3 \rfloor$  for a consensus system consists of  $n$  bookkeeping nodes.

### Example 1:



360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



Assume that there are 7 bookkeeping nodes A1 A2 A3 A4 A5 A6 A7. Among them, A1 and A2 are malicious nodes (less than  $1/3$  of the number of nodes), and A1 is the current primary node. A1 generates block1 and send PrepareRequest(block1) to A2 A3 A4, and it also generates block2 and send PrepareRequest(block2) to A5 A6 A7. When everyone receives the PrepareRequest message, A2 A3 A4 will return the PrepareResponse (block1) message, and A5 A6 A7 will return the PrepareResponse (block2) message. At this point, Block1 and Block2 both have 4 signatures respectively which have been published on the network. The network has not yet reached a consensus, and further communication is needed. The consensus block is undecided. But at this time, the malicious node A2 can actually generate the signature of block2 (then he can have 5 signatures of block2 on his own), so A2 has the ability to generate a fork in the network. In this attack case, dBFT cannot achieve Byzantine fault tolerance (2 malicious nodes).

### Example 2:

Assume that there are 7 bookkeeping nodes A1 A2 A3 A4 A5 A6 A7. A2 is a malicious node (only one malicious node). A1 is assumed to be the primary node of the current round. Due to a large number of transactions being processed or other network reasons, A1 has a latency and send a PrepareRequest (block1) out just before timeout occur. A5 A6 A7 just received and verified the entire PrepareRequest (block1) within the limited time and replied with signatures. A2 A3 A4 have not received this block1 yet. According to the dBFT protocol, when time is up, everyone will request a change view. Suppose the changeview package arrived at A2 A3 A4 earlier than Block1. Everyone reached a consensus on the change view and proceed to the next view phase. If at this stage, A2 A3 A4 receives block1, but the view and the primary have changed, they will not accept block1. Then everyone will enter the next round and generate block2. If the consensus process of Block2 is good, then several honest nodes A3 A4 A5 A6 A7 have reached a consensus. But at this time, A2 has 5



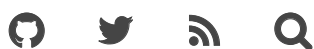
360 核心安全技术博客

🏠 主页 Home

🔒 归档 Archive

📁 分类 Category

👤 关于 About



signatures of Block1. It can construct valid Block1. Thus, A2 can create a fork in the network. In this attack case, dBFT cannot achieve Byzantine fault tolerance (1 malicious node).

The essential reason for the problem in this paper is that NEO's current implementation dBTF of attempts to replace the consensus of the whole network with the consensus between bookkeeping nodes; however, this assumption is not strictly established. The BTF protocol between the bookkeeping nodes can only guarantee the consensus among the honest bookkeeping nodes, but the consensus between the bookkeeping consensus nodes and the consensus between the non-bookkeeping nodes of the whole network is not strictly bound. With the current implementation of NEO, once forked, the fork node is not able to return to the NEO network.

## Impact

Upon the discovery of the problem, we sent an email to NEO founder Erik Zhang to elaborate the issue, Erik replied us in a few minutes that they had encountered the problem and that the actual fork had occurred in the network, and there is an announcement in NEO community about this problem. Now, a Pull Request is initiated on NEO github to handle the problem, but the stability is still being tested. The URL is: <https://github.com/NEO-project/NEO/pull/320>.

Through the analysis on the implementation and discussion of the PR, we find that the PR320 is actually a fix for the incomplete implementation of PBFT in NEO's current dBTF. The current understanding of the community is that the fork happened due to network issues between bookkeeping nodes because the lack of PBFT's commit phase in NEO. But in our research, even if the commit process is added, it can't prevent malicious bookkeeping nodes from constructing forked blocks. In response to this problem, the NEO team has proposed new improvements under the PR. Currently, the NEO community is refining the previous PR to completely solve the problem.





## 360 核心安全技术博客

 主页 Home

 归档 Archive

 分类 Category

 关于 About



In the process of solving this problem, we found that the NEO team handled security issues with great profession and efficiency, and the community responded actively in a timely manner. 360 security teams will continue to test, analyze and improve related issues with NEO to drive the benign development of NEO and blockchain technology.

本文链接: [http://blogs.360.cn/post/NEO\\_dBFT\\_en.html](http://blogs.360.cn/post/NEO_dBFT_en.html)

-- EOF --

作者 [admin001](#) 发表于 2018-09-19 09:01:07 , 添加在分类 [Blockchain](#) 下 , 最后修改于 2018-09-19 10:16:24

分享到: [新浪微博](#)[微信](#)[Twitter](#)[印象笔记](#)[QQ好友](#)[有道云笔记](#)

---

« [NEO dBFT共识机制分析与完善](#)

[毒云藤 \(APT-C-01\) 军政情报刺探者揭露](#) »

---

## Comments



© 2020 - 360 核心安全技术博客 - [blogs.360.cn](https://blogs.360.cn)

Powered by [ThinkJS](#) & [FireKylin 1.2.8](#)




360 核心安全技术博客

 主页 Home

 归档 Archive

 分类 Category

 关于 About

