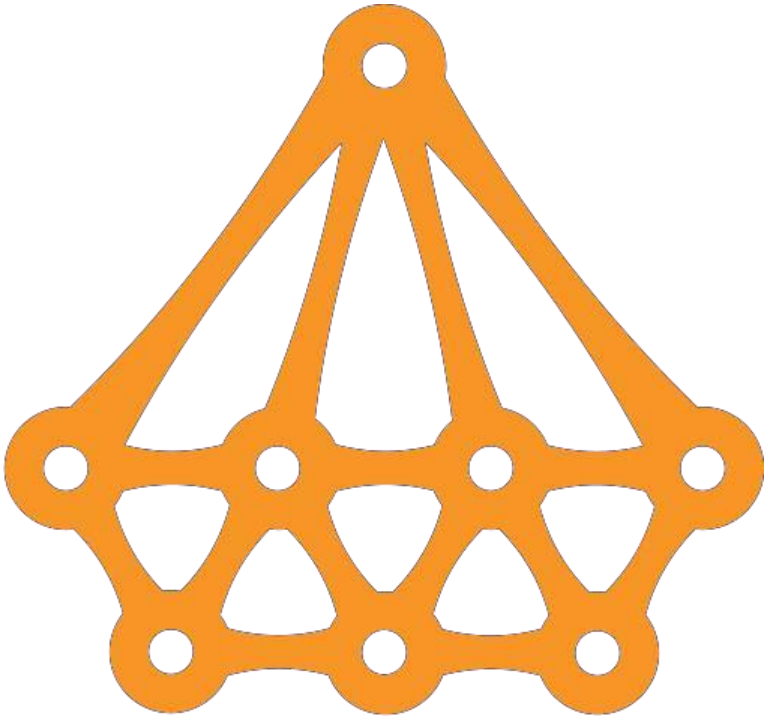


NERTHUS White Paper

A Universal DAG-based Blockchain Programming Platform

Abstract

Like Ethereum, Nerthus was created to build a universal smart contract programming platform and blockchain operating system. It has witstone, its own Turing-complete programming language, and OVM, its operating environment. But the difference with Ethereum is that we use the DAG technology for the data structure as it can solve the inherent paradox of the block + chain structure in traditional blockchain systems, i.e., low throughput, latency in confirmation of transaction, and block bloat.

Gong Jianfeng

Table of Contents

What is Nerthus?

Nerthus is a universal DAG-based blockchain programming platform and a decentralized and distributed blockchain operating system. It has its built-in Turing-complete programming language, with which users can build and define their own features, develop their own applications and blockchain systems, or issue their own tokens.

Why We Need Nerthus While Ethereum Already Exsist?

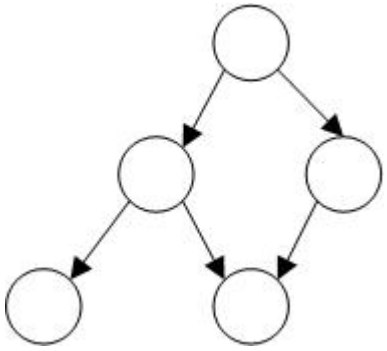
Traditional blockchain systems like bitcoin and Ethereum apply the block + chain structure, which carries with it the inherent defects of block data bloat, latency in transactions, and low throughput. Take the well-known debate over the scaling debate of bitcoin for example. Some people argue that without increasing the block size, many transactions will not be timely packed into blocks and thus suffer long latency as the transaction volume keeps increasing, while others against it believe that if the block size is too big, the block data will bloat so rapidly that they cannot be easily stored in ordinary personal computers. This is a typical paradox to reflect the inherent defects of traditional blockchains. By contrast, Nerthus applies the DAG (Directed Acyclic Graph) structure. There is no need for miners to pack transactions as they can be self-created and self-issued. It avoids the problem of transaction latency and is free of any performance limitation from computers of packing miners. Therefore the problem with throughput no longer exists. In the Nerthus system, each user maintains the data of all parents of its own units and stores the data of all parents of its counterpartied, instead of storing all data of the network. So even a huge amount of data in the network will have little impact on an individual user.

	Throughput	Latency	Data bloat
Nerthus	Theoretically with no limit	Only with network lag	Little impact on individual users even with a huge amount of network data
Ethereum	Theoretically 2000TPS	block produced every 15s	Expected to be 1T by the end of 2017

Technical Part

Background Information

DAG stands for Directed Acyclic Graph. In graph theory, if there is no way to start at any vertex of a directed graph and travel along several edges that eventually loops back to that vertex again, the graph is a directed acyclic graph. The following figure shows a typical directed acyclic graph, in which the circles represent vertexes while lines are edges representing the relations between vertexes.



The DAG structure was introduced to the blockchain by IOTA and was later applied and improved by Byteball. IOTA system, when a new transaction arrives, it must approve two previous transactions and in that way indirectly approves all transactions previously approved by those two transactions. Vertexes in DAG are transactions and lines with arrows represent the approval relation between transactions. IOTA has this concept of accumulative weight, meaning the own weight of a transaction plus the sum of own weights of all transactions that are approved by the transaction. Since in DAG structure transactions are always created and issued by themselves, theoretically an attacker can always build a transaction with a higher weight than the one he/she wishes to overturn for the purpose of double spending. Based on the efforts made by IOTA, Byteball introduced the concepts of main chain and witness to the structure and encourages the approval of several parent transactions. However, it has a witness list for each transaction unit, which not only increases unit data but also causes chaos if malicious attackers attempting double-spending deliberately issues double-spending units with different witness lists. Meanwhile, the confirmation mechanism of Byteball follows a main chain, with

encounter with many witnesses along the chain, raising the complexity and uncertainty of transaction confirmation. Though transactions will finally reach a stable determinacy, the time required for transaction confirmation is uncertain. Nerthus carries further improvements based on Byteball to maintain a witness list for each user. And as a new feature inspired by DPOS, Nerthus also allows a transaction unit to be deterministically confirmed once it is validated by a witnessing unit cosigned by all witnesses. Details in this regard will be illustrated later.

Unit

Nerthus does not apply the traditional chain structure as is used by bitcoin and Ethereum. Instead, it applies the DAG structure.

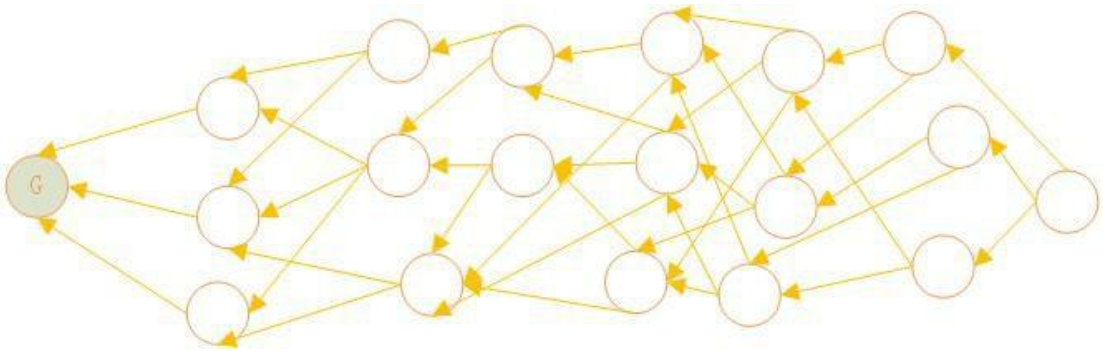


Figure 1

The above figure shows a DAG structure. The cycles are vertexes and lines represent the relations between vertexes. Arrows are directed from child units to parent units. G is the genesis unit. Each unit can travel back to its parents, parents of parents… until eventually reaching the genesis. As for Nerthus, a circle represents a unit. These units reference one and more previous units (parents) to build the order of units. A unit may contain several different types of data, like payment, text messages, and smart contracts, etc..

In DAG each new unit approves and confirms its parents, the parents of that parents.....up until the genesis; and includes the hash of its parents to its own unit. If someone tries to alter the data, then he/she must change the hash of the unit, making it inconsistent with the hash of this unit referenced by child units that directly or indirectly confirm it. To succeed requires the cooperation of all its children: a child alters the hash it references, changing its own hash. Then this child unit has to cooperate with all its children, children of children, until the last children. Once a unit

is broadcast into the network and validated by other units that are built on top of it, the number of users to be coordinated to alter the data grows like a snowball. In traditional single chain structure, to succeed in altering data requires consistency with only several later blocks theoretically (51% attack where a large computational power would enable to create a few blocks rapidly. As these blocks are self-controlled, the cooperation among them will be enough to alter data). As for DAG, it is more complex and more difficult to revise data.

Double spending and Serial Unit Series Chain of an Address

In a decentralized system, preventing double spending is a must and a foundation, failing which the whole system can no longer stand. DAG solves the double spending through the following protocol rules.

1. A unit cannot reference parents that are directly or indirectly referenced by its other parents.
2. If the same address issues more than one unit, a later unit must reference (directly or indirectly) all its previous units, to form the serial unit series of the same address.
3. If someone breaks Rule 2 and issues one or more nonserial units or unit series, they will be deemed as double spending no matter if there is actual double spending or not.
4. In case of double spending though in accordance with Rule 2, then in the serial unit series, the earlier one is deemed valid, while the later one is not. If someone breaks Rule 2 and issues several nonserial units or unit series, then according to the best serial unit series algorithm, only one unit or serial unit series is valid, while all others are deemed invalid.
5. If a unit of an address directly or indirectly references two or more nonserial units issued by this address, then this unit is invalid no matter there is actual double spending or not.

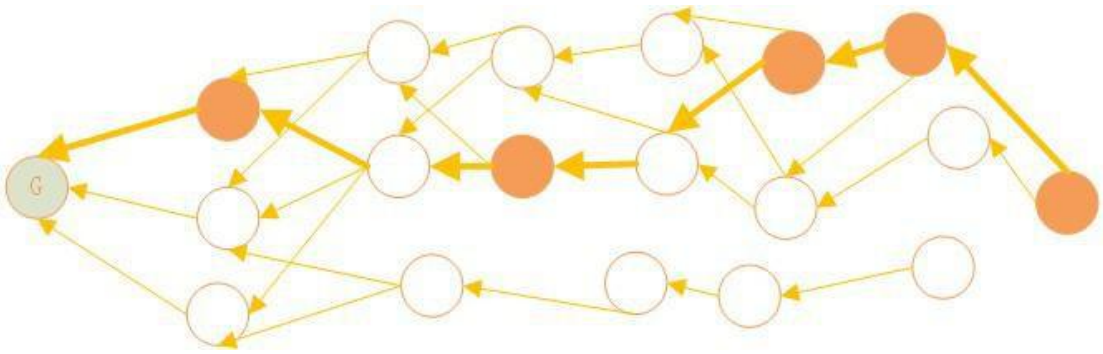


Figure 2

In Figure 2, the orange solid circles are all units issued by the same address. Later units directly or indirectly include earlier units, forming a serial unit series.

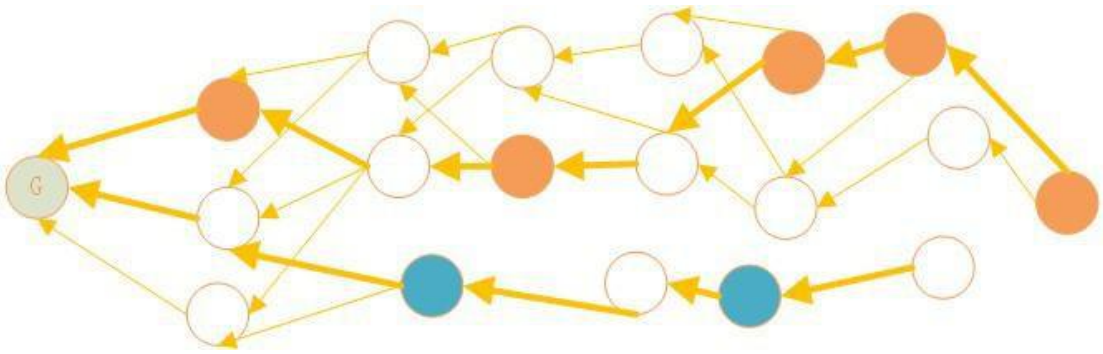


Figure 3

In Figure 3, both orange and blue solid circles are units issued by the same address. We can see that there is no serial reference relations between blue solid circles and orange solid circles. In this case only one serial unit series will be acknowledged. All transactions of other serial units will be deemed as invalid.

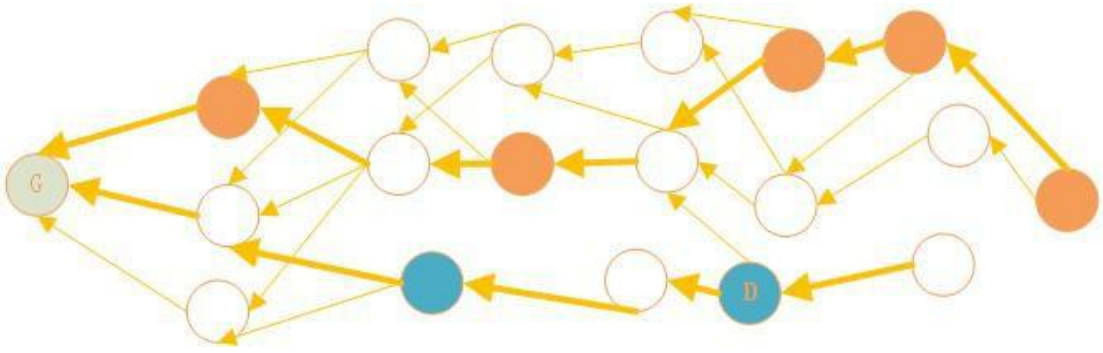


Figure 4

In Figure 4, both orange and blue solid circles are units issued by the same address. The blue unit D has indirectly included nonserial orange units and blue units. According to Rule 4, unit D will not be recognized and is thus invalid. Since D has referenced several nonserial units issued by the same address, subsequent units built

on top of it issued by the same address reference them indirectly and are thus invalid as well.

We can see from Figure 2, 3, and 4 that in the right-to-left direction, with a unit in the right directed with an arrow to its parent, then the parent directed to the parent of the parent …… up until the genesis unit, a chain that we call a serial unit series chain of an address is thus formed.

This serial unit series chain starts from a child unit who chooses its best parent, who will then choose its own best parent, etc. up until the genesis unit. The best parent is determined by calculating a weight based on the height of the unit, number of included witnessing units, timestamp, number of serial units of the child unit, and paths.

Based on Rule 5, we simplify the unnecessary complexity caused by two serial unit series of an address which start independently but intersect with each other later in a certain point. Therefore, if two or more serial unit series chains occur (to simplify the illustration, we consider independent nonserial units issued by the same address as serial unit series chains of this address even if there is only one independent unit that cannot form a serial unit series), only one of these series will be deemed as valid as per Rule 4. The following rules are used to choose one serial unit series chain of an address as the valid one, rendering all others invalid and unrecognized.

1. If the starting point is invalid, then all subsequent units are invalid. Among several different serial unit series issued by an address, if the first address unit of a chain is invalid, then the whole serial unit series chain is invalid.
2. According to Rule 1, we can make a simplification. If the same address issues several different serial unit series, we only need to compare the first address units of these series and choose the best address unit. Then the serial unit series chain of this best address unit is valid while all others are not.

Witness

Since there is always an order of serial units of an address, the problem of double spending can be easily solved when earlier units are valid while later units are invalid. But it gets complicated if an attacker deliberately issues several nonserial units or unit

series, because in DAG units are created and issued by themselves, so each unit can choose its own parents and height, and forge timestamps. He/she may forge units with higher weight than the unit he/she wishes to overturn within the context of existing rules or create more units to confirm the double-spending unit. The attacker might also create a hidden unit series chain and propagate it at a certain moment to overturn all previous units. And what makes this worse is that you don't know when he/she will propagate it. That's why we introduce the witness mechanism to solve this problem.

The witness mechanism was inspired by the delegated proof of stake (DPOS) of traditional blockchains. Witnesses are chosen by elections. Users may apply for the election and pay a certain amount of security to become witness candidates. The system will calculate an index according to the votes obtained, security, reputation and real-world identities, and general behaviors of a witness candidate and then assign users to them based on the index. As the reward, a witness will share the transaction fees of all units of his/her witnessed user with all other witnesses of this user. However, this reward will be frozen for 3 months before being drawn out. If a witness fails to perform his/her duties in a long time or to issue any witnessing unit, he/she will be deprived of the witness qualification and cannot run for witness in a specific period. If the witness fails to obey witnessing rules and issues invalid and malicious witnessing blocks, he/she will have his/her security forfeited and never be able to become witness again.

Each user has his/her own witness list composed of odd number of witnesses. Before a user creates the first unit, the system will assign a witness list for him/her broadcast the list to the network, which will maintain the witness list of each user. To prevent collusion between users and witnesses, users cannot choose their own witnesses and neither can witnesses choose their users. Besides, the witness lists will be updated periodically. All these updates will be recorded by the network, as well as the starting time and ending time when a witness serves a certain user.

After a user issues a unit, the witnesses in the witness list of the user will witness and confirm the unit in the following steps:

1. The user issues the unit to the network (and directly to his/her witnesses).
2. The witnesses receive the unit and communicate with all other witnesses

of the user to confirm that all witnesses receive the same unit. This is a necessary step. If a malicious user wants to attack the network, he/she may send different nonserial units to different witnesses. Without communication among the witnesses, each witness will confirm a different nonserial unit. As a result, the system will fall into chaos. Take an extreme scenario for example. If a user has a witness list of 21 witnesses. Before forming a unit series, he issues 21 different nonserial units at the same time and sends one to each witness. If the witnesses don't communicate with each other, they will each confirm the unit received thereby. As a result, 21 independent units are all valid, which is surely unacceptable.

3. If all online witnesses receive the same unit, they will check that there is no mistake, cosign the witnessing unit, and issue it to the network. Online witnesses must account for over 50% of all witnesses in this user's witness list. If the user has 21 witnesses, then at least 11 online witnesses are required. The reason why we introduce the concept of online witnesses is that there may be network problems, or witnesses may go offline sometimes, or some witnesses have malicious intent. Leaving a 49% redundancy can ensure the robustness of the system.
4. If all online witnesses receive different units from the same user, they will exchange these different units to check if there are any valid inclusion relations. If yes, then the last child unit will be validated. In case of malicious inclusion among these units (meaning that a unit directly or indirectly references two nonserial unit series), the unit will be discarded. If several units are nonserial and independent from each other, then all online witnesses will choose the unit with the smallest hash as the valid unit to cosign a witnessing unit and issue it to the network.
5. Once the witnessing unit is issued, the validated unit of the user and all ancestors referenced by it are confirmed. And this confirmation is final.

Witnessing Rules:

1. At the same intersection, the same witness can only validate one unit or one unit series thereof.
2. If a user's unit or unit series already has a validating unit, then all

subsequent validating units must be built on top of this serial unit series chain already with the validating unit and an order must be established among these validating units as well.

3. If a witness breaks the above Rule 1 and 2, his/her security and the transaction fee are frozen for three months and will all be forfeited and he/she can never become witness again.

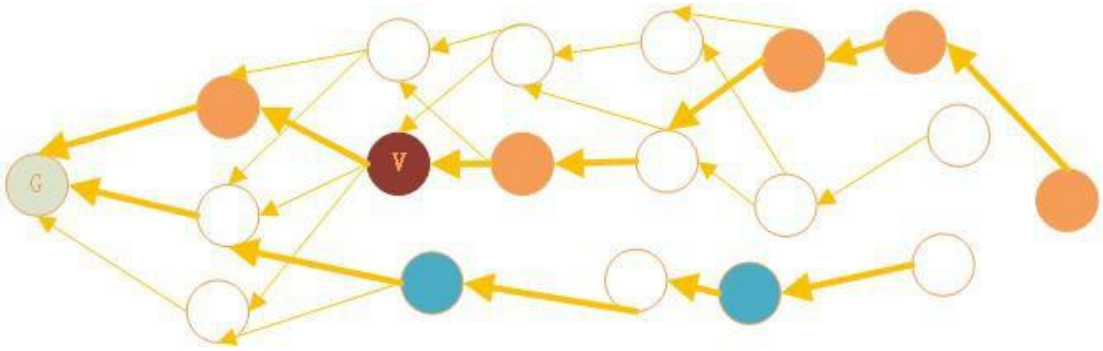


Figure 5

In Figure 5, the orange chain and the blue chain are two independent serial unit series issued by the same address, neither of which has any units before the first unit thereof. We call such unit not referencing any units as unit 0. And the intersection here is called intersection 0. The purple unit V is the validating unit signed by witnesses which confirms the orange serial unit series.

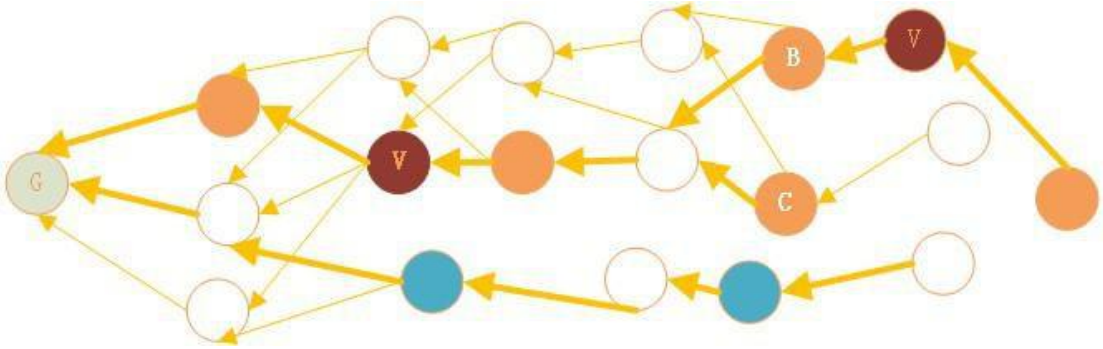


Figure 6

In Figure 6, two forks appear in the orange serial unit series after the first validating unit: unit B and unit C. Since the validating unit chooses B, then B is valid while C is not. Starting from the first orange unit, if excluding nodes not related to the address, then B and C are in the fourth generation. So we call them intersection 4.

On intersection 0, the validating unit confirms the orange one and on intersection 4 the validating unit confirms B. According to Witnessing Rule 1, all other serial unit

series or units which are now there or later forged on intersection 0 and intersection 4 are invalid. And according to Witnessing Rule 2, if a user's unit or unit series already has a validating unit, then all subsequent validating units must be built on top of this serial unit series chain already with the validating unit. And an order must be established among these validating units as well. This approach can make sure any user address has only one clear and recognized serial unit series.

Nerthus's Five Big Breakthroughs

1. More thorough decentralization

Traditional block + chain structure need to have a kind of centralized operation, which means a bookkeeper, to verify and process all current transactions, and then pack them into a block and then publish to the network.However, the Nerthus system, as described above, adopts a unit +DAG structure, without the concept of blocks. All units are created and published by the users themselves. Its verification and confirmation are assumed by its descendant unit,who refer it to as ancestor unit. Nerthus does not need a bookkeeper like the traditional block + chain structure, and the centralized operation of packaging all current transactions to blocks, therefore, it is a more thorough decentralization system.

2. No throughput bottleneck

Because the traditional block + chain structure has a central operation process, that is, the bookkeeper needs to package the transactions to the block.Therefore, the transaction processing capacity of the blockchain system must be subject to the following three points: 1) The performance of the bookkeeper node machine;2) The network bandwidth of the bookkeeper node;3) The block size. Because there is such a central operation, no matter how to optimize, there will always be a bottleneck point of processing capacity. As mentioned above, Nerthus system adopts the unit +DAG structure, and there is no central operation of bookkeeper packaging blocks, so there is

no concept of blocks. The unit is published by the users and validated by other units, so there is no throughput bottleneck.

3. No paradoxical dilemma between blockchain scalability and data expansion

Traditional block + chain structures require all transactions to be packaged into blocks, then those transactions would be valid.If the block capacity is set small, when trading volumes are high, many transactions cannot be packaged into blocks in time.If the block capacity is set large, it will make blockchain data expand rapidly, and ordinary PC cannot run full nodes, and only a few people could run full nodes, which will result in centralization. This is also the fundamental contradiction of the bitcoin scalability debate. As mentioned above, Nerthus does not have the concept of blocks, so there is no inherent paradox dilemma of traditional block+chain structures.

4. Definitively predictable finality

The traditional block+chain structure does not preclude the possibility of two or more blocks being produced at the same time, which lead to a fork. In the case of a fork, the traditional blockchain will take the longest chain as the effective chain. The mechanism would in theory be unable to determine finality, because no one can guarantee that wheather there is a longer chain being hidden. However, In Nerthus system, through the witness mechanism, it can not be overridden as long as it is verified and confirmed by the witness unit published by the witness, which is the finality.

5. Optional transaction confirmation speed

Witness can release witness block at five speed levels: urgent, very fast, fast, ordinary and slow. Users can select the transaction confirmation speed according to their own requirements.

Nerthus Ecosystem and Three-tier Architecture

Nerthus is a universal smart contract platform and blockchain operating system. It is also working to build an ecosystem based on Nerthus.

Nerthus uses the unit +DAG structure at the bottom layer, creating a more thorough decentralized blockchain system without bookkeeper packaging. A blockchain system without a throughput bottleneck is the core part of Nerthus ,and is also the infrastructure it provides. In the whole Nerthus system, it's at the bottom, it's the base layer.

On top of the Nerthus base layer, we also introduced the service layer for developers developing applications quickly based on Nerthus. In the service layer, in addition to encapsulating various APIs of the core layer, we provide blockchain translation system and sidechain system. The blockchain translation system mainly refers to making two independent blockchains able to understand each other and communicate with each other without obstacles.Sidechain system refers to enterprise users, which can quickly generate private chains and alliance chains based on Nerthus and link them to the mainchain of Nerthus, it also can use the Nerthus translation system communicate and make transactions with other chains.In addition to advanced technologies,a good ecosystem should be able to a variety of resources.The main purpose of the blockchain translation system is to integrate the existing various blockchain resources, so that existing block chain projects can communicate with and interact with each other through Nerthus, then become a part of Nerthus ecosystem. Application developers based on Nerthus can develop applications that across multiple chains.The sidechain system provides a fast and low-cost way for enterprises and institutions to establish alliance chain and private chain. Although these enterprises and institutions can only establish alliance chain or private chain for a variety of reasons, they still have communication and transaction needs with other public chain users, and the sidechain system of Nerthus just provides such functional interfaces for them.

Above the service layer is the application layer of Nerthus.The Nerthus application layer refers to various blockchain applications developed on Nerthus, which are mainly developed by third-party developers. Currently, blockchain applications, especially, blockchain wallets, its user experience is not very good. Blockchain developers are more focused on the implementation of functionality, not very high attention to the user experience. A technology, a system, an application, and if you want everyone to use it, the user experience is very important. Nerthus pays great attention to user experience, learning from Apple's IOS, Nerthus will build a set of specifications and standards for Nerthus application layer.

Witstone and OVM

Smart contract programming language and running environment are the infrastructure to implement the programmable blockchain operating system.

OVM is a virtual machine specially developed for Nerthus blockchain system, and provides a safe, reliable and high-performance running environment for Nerthus smart contracts.

We also provide a programming language for writing Nerthus smart contracts——Witstone.

Witstone is a scripting language similar to javascript. It adds support for some of the features of nerthus at the language level. It is an easy to use smart contract programming language, and it does not take developers much time to learn, they can write Nerthus smart contracts in a short time.

Finality

A big problem of bitcoin and Ethereum is that there is no certain and

unchangeable final state. Theoretically, with enough computational power and faster block producing speed to create a longer hidden chain, the previous blocks can be reversed. In Nerthus, units become final once the witnesses issue witnessing blocks, and cannot be reserved.

Confirmation Time

Confirmation time depends on the time spent by witnesses to issue witnessing blocks. We classify this time as five levels: super urgent, urgent, fast, ordinary, and slow. Each level requires a different amount of validation fee to balance the load of witnesses and the time demand of users being witnessed. Super urgent units can be confirmed within 1s if not considering network conditions.

NERTHUS Tokens

Nerthus will provide Nerthus tokens, the fuel to maintain the Nerthus system. Users need to pay witnessing fees to witnesses in order to create and issue new units and pay step counting fees to operate smart contracts. Nerthus tokens total 10 billion.

Conclusions

Like Ethereum, Nerthus was created in an attempt to build a universal smart contract programming platform and blockchain operating system. It has witstone, its own Turing-complete programming language, and OVM, its operating environment. But the difference with Ethereum is that we use the DAG technology for the data structure as it can solve the inherent paradox of the block + chain structure in traditional blockchain systems, i.e., low throughput, latency in confirmation of transaction, and block bloat. To have more practical smart contracts, Nerthus system is designed with an off-chain data validation consensus mechanism, which allows previously impossible scenarios to be realized by smart contracts. Nerthus will promote the adoption and application of smart contracts in the real world.

References

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Vitalik Buterin. Ethereum White Paper : A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper.

[4] Serguei Popov for Jinn Labs. The tangle.https://iota.org/IOTA_Whitepaper.pdf, April 2016.

[5] Anton Churyumov. Byteball: A Decentralized System for Storage and Transfer of Value.

[6] SergioDemianLerner DagCoin, https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf, 2015.

[7] Delegated Proof of Stake. http://docs.bitshares.org/bitshares/dpos.html

[8] Nxt, 2013, http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt

[9] Smart contracts: https://en.bitcoin.it/wiki/Contracts