Power_

Yellow paper

v.0.2 (draft)

Hybrid three-dimensional scalable blockchain
with instant transactions and real-time auto sharding.

info@thepower.io

Disclaimer

- This document is subject to change.
- Additional parts to be published hereafter.
- The purpose of this document is to acquaint you with the project technical aspects comprising the basis for solving such an ambitious task.
- Questions and comments may be sent to info@thepower.io or posted in our official Telegram group.

ATTENTION

In case you do not agree with following terms of the non-disclosure agreement, please close this Yellow Paper and remove it from your computer in such a way that it can not be restored. Continued reading of the text of this document means your full and unconditional acceptance of the confidentiality conditions. You fully understand and assume obligations to maintain confidentiality and are ready to bear full financial responsibility for any uncoordinated disclosure of the information contained in this Yellow Paper or a part of such information.

Terms of non-disclosure agreement is available in Appendix A.

Abstract

Since 2008, Blockchain technology has continually evolved, becoming more versatile to use and cost effective. Today, however, despite all the diversity found within the Blockchain space, most business problems remain unsolved or at best partially solved. Examples of such problems in modern organizations include:

- Network bandwidth (transaction rate);
- Network and data security;
- Scalability;
- Latency time;
- User experience;
- Flexible economy;
- Versatility and the ability to customize a Blockchain for different conditions and user requests.

There are projects achieving significant results in solving these problems, but other no less important problems remain unresolved because of it.

We believe that Blockchain must simultaneously solve all the problems listed above in order to become a mass working tool for companies and organizations.

The Power team has a goal to achieve it.

To learn more:

- Visit the project website (thepower.io) and read our White Paper.
- Gain access to the testnet or to learn how to run your local Power_testnet.
- Review our code on GitHub (github.com/thepower).

Preamble

Power_blockchain is a distributed ledger platform for high-speed processing of interactions within a decentralized single address space. In contrast to many classic platforms, Power_blockchain is a platform for general purpose interactions where processing is not limited to such applications as value transferring [1] or smart contracts [2].

Power_blockchain attains high-speed processing by means of horizontal scaling of internal distribution of interactions processing solutions to dedicated blockchain subsystems – shards. Shards process transactions in parallel. According to trilemma [3], scalable projects usually have security or decentralisation problems, and consequently are considered to be private (permissioned) platforms, for example Hyperledger, Fabric and BigchainDB. However, Power_blockchain platform is developed to utilize all conditions in the trilemma.

The basis of Power_blockchain is a three-layer architecture which allows it to be a public platform while also solving the scaling problem:

- *Management layer* is responsible for decentralization.
- *Sharding layer* solves the scalability problem.
- *Validators layer* provides a high degree of security.

Because of the use of multidimensional (multilayer) blockchain management system own algorithm of consensus Resonance has been developed. It’s utilized within shards (Sharding layer), while enabling nodes to interact in frame of two other layers.

Resonance consensus

Introduction

The problem of finding coordinated decisions by a group of peer electronic devices while some of them do not work properly or do not work at all was formulated in 1982 as the "Byzantine Generals Problem". Various algorithms solving this problem have been proposed and introduced in distributed systems since then. Now a distributed system running on these algorithms is called BFT-system (Byzantine Fault Tolerance).

However, most of these algorithms are generic and have serious disadvantages creating difficulties for their effective use in Blockchain platforms. The Power has developed a brand-new BFT consensus algorithm called Resonance. The algorithm allows us to solve the problem efficiently within a Blockchain with sharding.

All the modern consensus algorithms used in Blockchain systems can be generally divided into two types:

- Type I. The algorithm requires the identification of all the system participant nodes. Such algorithms are often called the BFT algorithms.
- Type II. The algorithm doesn't require the identification of some or all system participant nodes. Such an algorithm has been used in the Bitcoin - the first Blockchain platform - PoW, PoS, DPOS, etc.

Problems to be solved

The main problems related to the use of *Type II* algorithms for the building of blockchain platforms with sharding are:

- The algorithms of this type (to some extent) include block's chain branching feature. It does not cause serious problems for a single branching chain, as there is a simple rule for determining the validity of the blocks chain branch - the "the chain having the greatest length of blocks is right". In order to check compliance with this rule, you must have a node fully synchronized with other chain producing nodes. But, in the systems using several parallel chains (sidechains or sharding systems), the transactions transfer from one chain to another involves additional difficulties. The target chain nodes must somehow confirm that these transactions belong to the valid version of the original chain branches. And the complicated problem here is to generate and verify this confirmation.
- The idea of branching and the "the chain having the greatest length of blocks is right" rule has led to the interesting empirical rule: ensuring the transaction irreversibility requires waiting for 5 blocks in the chain over the block in which the transaction is to be done. No doubt that such a rule leads to a serious increase in latency - the user has to wait long for the result of its transaction completion.

The main problems related to the use of *Type I* algorithms for the building of blockchain platforms with sharding are:

- Despite the fact that the BFT algorithms usually do not involve the chains branching, their implementations such as pBFT have quite a number of steps in the algorithm, resulting in long transaction registration time - about 3-5 seconds.
- Most of the BFT algorithms has been developed based on the asynchronous interaction models. This leads to the fact that the system vitality can be described by the $n > 2f + 1$ formula. That means that if your system has more than ⅓ of idle or malfunctioning nodes, the system will pause its work until the missing unit will come back to operating state.

The problems inherent to most blockchain platforms are:

- Rejecting the requests from the Blockchain platform client.

  In most algorithms, a block forms a single node, and the other nodes only validate it. This results in the node creating a block based on its own purposes which may not include transactions do not comply with its targets. Some examples of such actions may include: selection of only the transaction with the highest commission, failure to include the winning transaction (if the included transaction could lead to win for the sending party), and so forth.

  In such algorithms, a constant change of blocks generator occurs in theory. It seems helpful in reducing the risk for the client of non-inclusion of its transactions in the next block. But, in practice, the number of block generators in real blockchain platforms is about two dozen or even less. As a result, the probability that the next block generator will not include the client transaction becomes high enough.

- Common transactions sequence.

  The transactions line generation system is not directly connected with the consensus algorithms. However, it seriously affects the interaction between the client and the blockchain platform. This system stores and delivers the transactions from the client to the block generator, and it also consists of decentralized nodes. Since the delivery system is asynchronous, the client has no guarantees of timely transaction delivery. Therefore, the interaction "user - transaction – block entry" is extended on the transactions delivery time via a common sequence, and the time is most often unknown.

System model

We assume that all the nodes forming a distributed system are tightly connected to each other in a network. The messages between all nodes are delivered with minimal delay and without losses. The messages loss is considered an unfortunate incident and further processed as an error.

Each node in the system has information about each other node in the system (the address in the communication system between nodes providing data communication between nodes), allowing each node to communicate via a data link with any other nodes within the system.

We use the sha-2, elliptic-curve cryptography cryptographic libraries for authentication of all interactions between the nodes. All the nodes know the each other's public keys for verifying the messages from other nodes.

In order to reduce the size of transmitted data during the confirmation, we apply the Merkle Tree Proof (MTP). When transmitting data to the MTP, we can delete unnecessary data from the structure leaving only their hashes, and the validity of the MTP general structure can be easily confirmed by calculations.

All the nodes are constantly synchronized with each other for all the steps of the algorithm to be started and ended simultaneously. To do this, the nodes transact additional synchronizing interactions.

The system clients can send their transactions (requests for changes to the system) to any node. The nodes jointly form the common sequence of transactions at each round.

During the development of the algorithm, we assumed the presence of the enemy who is able to coordinate the work of the captured nodes but cannot affect the connectivity of the normally operating nodes.

Resonance Algorithm

The Resonance algorithm is an algorithm for round-to-round state coordination of the system consisting of distributed nodes. The consensus algorithm is applied in order to match the newly found system states with the maximum possible number of nodes at each round. The minimum number of nodes required to achieve a coordinated system state in each round is 50% + 1 node coordinated in this state.

The basic algorithm repeated at each coordination round is described below:

Step 1. "Round Start"

Each system node sends a message to each system node including itself that the round has been started. This message includes a set of transactions proposed by the node from the transactions sent by users of it, a new round number, a valid MTP with the previous round number and a round repeat counter, if a round was not ended at least once. The nodes select messages with the maximum number of the previous round for processing the next step. If there are several messages with the same maximum round number but different valid MTP with the previous round number, the node selects messages with the maximum round repeat counter.

After the first step, if there are no nodes with a Byzantine behaviour, each working node receives the same number of messages about the round start with the round number. If there are nodes with the Byzantine behaviour, these nodes send messages with the lower round number than other nodes do (without Byzantine behaviour). In this case, the node received a message from another node with the round number less than its own, it sends valid MTP back to this node with the previous round numbers stored in the node database.

Step 2. "Round End"

Each node looks in the messages selected at the last step and extracts transactions recorded to them. Then each such node sorts the transactions using identical method (method of sorting does not matter, but it must be common for all nodes). After that, the nodes verify the resulting sequence of transactions and their implementability.

For example, if the initial condition says that there is a variable equal to 5, it is strictly greater than zero, and the operation is proposed to subtract 10 (usual debit operation), therefore it will be impossible to calculate a new value, and the operation will not be performed.

Division by zero could be another example when the calculation of a new value is impossible. Further, the transactions which can not be performed (repeated subset, obviously false subset, a subset of which can not be calculated based on the current value) are removed from the list.

The node then applies these transactions, and a new system state is sent to all nodes, including itself, in a message containing a hash of the new state, round number and the largest round repeating counter, as well as the digital signature of this node.

At the end of a step, the node accumulates messages from all system nodes containing the hash of the new state, round number and the round repeating counter. After the accumulation of messages, the node could be transferred to one of the states listed below depending on the number of idle nodes, nodes with Byzantine behaviour in the system and the communication system between the nodes.

- "Round Failed". The node has received less than 50% of hash signatures with the same round number and round repeat counter. If the node fails to this state, it increments the

round repeating counter and starts to coordinate values from the beginning.

- "Round Success". The node has received more than 50% of hash signatures with the same round number and round repeat counter and accumulated 50%+1 of identical hashes. The round coordination is considered successful, and a distributed system comes to a single decision. The node is ready to start a new round of decision-making. If necessary, the round counter value for the node is set to 0.
- "Round Failed, correction is required". The node has received more than 50% of hash signatures with the same round number and round repeat counter but did not accumulate 50%+1 of identical hashes. This situation indicates that the nodes with Byzantine behaviour appeared, and it is necessary to identify them and finish the round. To do this, additional correction steps described below are used.

If the system contains at least one node with a Byzantine behaviour, it may send the messages with different subsets of the new value to other nodes at the 1st stage. This will lead to the fact that no node in the system accumulates 50%+1 of identical hashes of the new state, which means that the round is not yet finished and no decision has been applied yet. The system can apply a coordinated value using a correction algorithm we have developed.

The correction algorithm:

- Step "Starting Correction". A node that has not received the 50% + 1 node of the same responses at the *Round End* creates a message containing the current round number, and messages received from other nodes at the *Round Start*. These messages contain a subset of a new system value proposed by the node, new round number, and valid MTP with the previous round number. Next, the node sends a message to each other node.
- If the system has one or more nodes that have collected 50%+1 of identical signatures at the *Round End*, these nodes send a response message to the nodes that did not receive 50% + 1 node of identical responses at the *Round End*, and these nodes start synchronizing with the new system value.
- If none of the system nodes collected 50%+1 node of identical hash signature at the *Round End*, each node collects common matrix of messages containing the information about all messages sent from each node to each other. Each node analyses the matrix and defines nodes sent messages with different subsets in this round (nodes with the Byzantine behaviour).
- Step “Correction End".

Each node excludes the *Round Start messages* received from the nodes identified as a Byzantine behaviour, and then repeats the *Round End* step.

The Reward model: par minting

An important difference between a distributed system and a decentralized blockchain system is that owners of the blockchain system nodes are economically interested in their nodes’ working condition. The reward for the work of such nodes in the system helps to achieve the system decentralization.

With lack of rewarding node’s owner does not have the motivation to protect it from attack or e.g. not to transfer the right to someone else. It leads in turns to a situation in which the process of nodes seizure comes to simple and cheap matter.

Thus, the network consist of unseized nodes can easily become a network with captured nodes, and such network seizure is economically quite cheap. The project goal is to develop a public network where the cost of seizure might be so much high as it would be unviable. So that, the cost of seizure might be

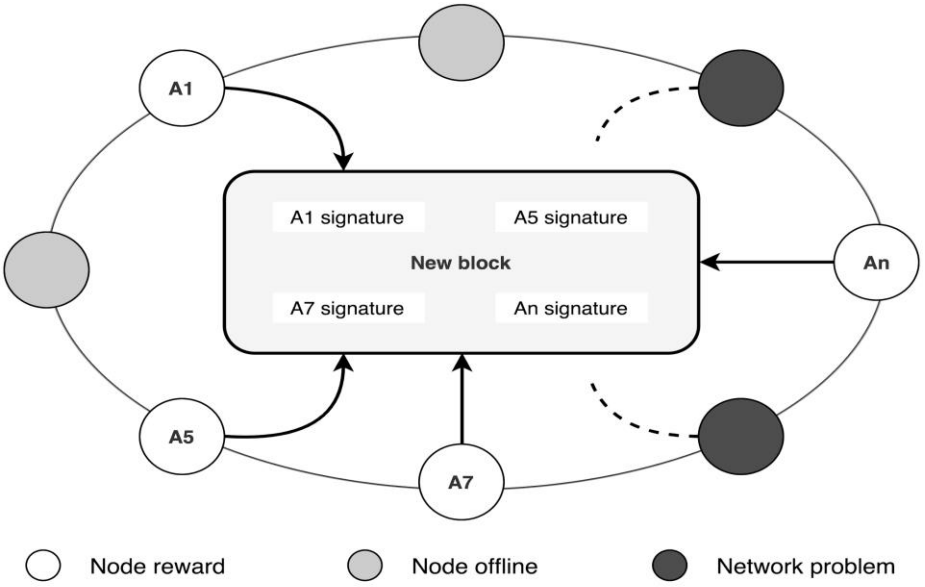

Image №1. Reward model

higher than the reward. It could be possible only in case nodes are interested to protect itself and the cost of protection directly correlates to actions reward. Thus, the higher nodes reward, the safer common network.

In the PoW, PoS, DPOS-based systems, only one block generator receives the reward for the block generation. The Resonance algorithm differs from most other ones in the fact that all system nodes are involved in the block generation. Therefore, in systems based on Resonance, all the participants must be rewarded. The normal operation of the system may be ensured only by encouraging those participants who actually got to the round coordinated state, and not encourage those who did not participate.

The algorithm's feature is that it is impossible to identify on the *Round Start* stage which nodes are involved in all steps of Round creation. So, the rewarding procedure for block K is executed by system smart contract as one of the coordination tasks for next rounds and must be carried out if the block number is n>K.

Summary

The presented scheme describes the algorithm of coordination of new blocks in the decentralized blockchain system. The implementation of the algorithm has the following advantages:

- Predictably low client transaction registration time;

  The clients can send requests on the transactions implementation directly to the node and receive an answer about the result. The transaction processing is time-limited, so the client can predict the time of registration of its transactions.

  The small number of steps in the basic algorithm and the correction algorithm helps to reduce the time of transaction registration by the system. We are going to achieve a registration time of 1 second or less.

- Low probability of rejecting of client's request by the node;

  The client may send its transaction to several or all nodes in the system. In such a case, its transaction will not be included only if 50% of nodes together made that decision.

- Simple check of the blocks validity for the participants of other chains (shards) and the client;

Checking the block sent to the client node or node from another chain just have to make sure that all of the signatures of this block are valid and really belong to the generating nodes of this block in the system. To do this, we need only a valid knowledge about the nodes and their cryptographic keys that compose a decentralized system.

- System efficiency if nodes are failed.

  In order for the system to work, it is necessary for the routine mode to have 50% + 1 node of all nodes registered in the system work normally. Thus, the system will continue to operate even if 50% -1 nodes will fail.

The attention should be given to the important technological features of the Resonance algorithm

- Algorithm are demanding for the network quality for all participants, more demanding than that of analogues.
- The high efficiency of the algorithm at a small number of nodes.

The efficiency decreases with the increase of the nodes number, as it is accompanied by an intense increase in the interaction between the nodes. This problem is solved using a scaling technology (sharding) which will be described below.

Scalable architecture

Introduction

The project architecture has been developed primarily to solve the trilemma (horizontal scaling while maintaining security and decentralization).

However, during the architecture design the great attention has been focused on following issues common for modern blockchain platforms:

1. Latency.

Latency is the time of system response to the client on an irreversible record of the transaction in the system.

Unfortunately, in most cases latency increments due to nodes amount growth in the system because of network outgoings. Herewith the increase of nodes is important for blockchain systems since it leads to increment of the budget for an attack to the system and works towards greater fault tolerance. Thus, most systems have to keep the balance between latency and decentralization. In an ideal system, latency might be constant or decrease.

2. Privacy.

Organizations and users of distributed public registers face the problem of data surveillance. In most cases, it takes just to identify the wallet owner to get the whole story of his transactions. The blockchain is a large database which may affect private life and commercial activities in the long term, working with big data. But the companies need to keep trade secrets and trust data from other participants in the commercial chains.

3. Expensive transaction.

The commercial services need an opportunity to predict and model transaction costs. The blockchain developers do not care about an economical efficiency of the developed systems compared to the centralized solutions with no compromise. Blockchain platforms tokens are volatile in the markets, but the speculation affects all the system users and exposes additional financial risks to the company.

4. Versatility.

Classic blockchain technologies are not intended for global use without any limitations. Specialized platforms are too specific, so they can not present a market

standard and compete for users in their respective segments. For the decade of blockchain technology development, nobody has developed universal approaches as it was with HTTP and TCP/IP development by making the Internet available for everyone.

Three-level architecture

The Power is a *Blockchain platform* with three-level architecture:

1. Shards layer.

Multiple mini-Blockchains called *shards*. All working transaction interactions and smart contracts are performed at this layer.

2. Validators layer.

This level is implemented to increase security of inter-chain (cross-shard) transactions. An inter-chain transaction not approved by the validators team will not be valid in a destination shard.

3. Management layer.

This level includes only one blockchain consisted of all system nodes. The purpose of this chain is key systems platform management, including all nodes, shards and accounts.

Classic blockchain is not divided by separate execution levels and has no interactions between the system levels. All the transactions had the same utility role.

In the platform, the transactions are divided into:

1. *utility transactions* intended for the interactions between the system users. These transactions account for the vast majority of all transactions;
2. *managing transactions* intended for the management of the system elements to ensure its effective operation. Security of these transactions is critical to the platform.

The multi-layer architecture allows us to separate different types of transactions to increase overall performance. The utility transactions are carried out in Shards layer, which provides their scalability and minimizes transaction costs since the transactions are performed in separate shards. The managing transactions are performed at the Management layer, which increases their performance safety since all the nodes are involved in the transactions processing.

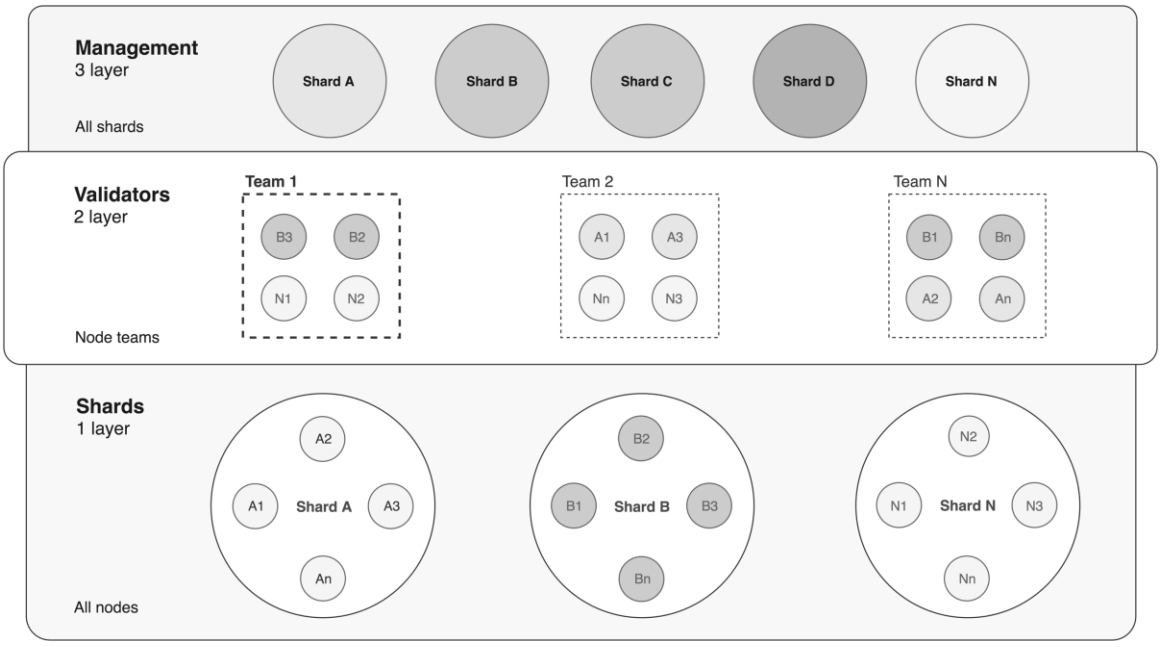

Image №2. Blockchain layers

As a result, this interaction structure allows us to perform the most important functions in the most secure level of the platform, and the ordinary transactions are performed in less secure but more productive mini-blockchains (shards).

Power_node

The most important part of the platform is a *network node*. The system of cross-node interactions creates all vital functions of transactions registration services and decentralized code execution. What the network node is?

In Power_blockchain, the node is a combination of the following:

- Software allowing to perform consensus-finding in order to create a new block in the shard (*Shards layer*).
- Software allowing to perform consensus-finding in order to create a new block in the *Management layer*.
- Software that allowing to perform verification of new shard block at the *Validators layer*.
- Cryptographic key pair, through which other nodes can identify the relation of all service packs on the network during the nodes interaction.
- The system of node management access.

Nodes ownership structure

In Power_blockchain, the right to manage the node is regulated by records in the blockchain. Technically, this right is given in the form of a special token (NT, Node Token). The account having this token is able to manage the node and owns it. Each token has an ID so that even in case of the owner change it does not affect a node identification.

All Node Token s are initially stored in the system smart contract. The user can get a Node Token by transferring SK tokens to the address of the smart contract. If the user has decided to refuse the node, he can transfer NT token to the system smart contract and get back SK tokens.

The Node Token entitles the account to which it is assigned to change the node public key. Thus, the node owner can not worry about the private key theft if a fraud gains physical access to the server with node software. In such a situation, the account owner can generate a new pair of cryptographic keys and register the public key in blockchain as the new key in a second. When a new key is registered, a node captured with the old key will be recognized as invalid and will not be taken into account in the course of the system operation.

Node roles

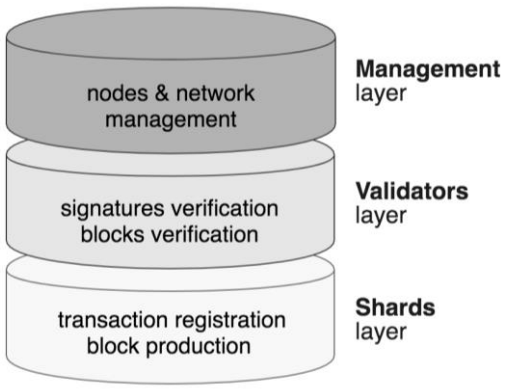

Image №3. Node roles & functions

In classical decentralized platforms, a system node is a combination of platform software and hardware. Typically, one node as a server unit performs one role, e.g. blocks producer or validator. However, a Power_blockchain node simultaneously performs three roles:

- Participates in the platform management as a part of Management layer;
- Handles user transaction as a part of a shard;
- Verifies the blocks generated in other shards as a validator at the Validators layer.

Thus, there is no need to require that all roles were performed by a single software on a single

server. As noted above, each role in Power_blockchain has its own software. And due to the fact that the system of node ownership is secure and there is no need for a high level of cryptographic keys storage security, each role can be performed both on the same physical or virtual server and on different ones.

Management layer (ML)

The Management layer is a large and relatively slow blockchain included all the system nodes. At any time, the platform node is simultaneously involved in a separate shard and a Management layer chain.

As a part of Management layer, the node performs the following:

- Adding, moving and deleting the nodes in the public shards;
- Creating, separation and removal of public shards;
- Registration of creating and reconfiguration of private shards.

Consensus in the Management layer (ML)

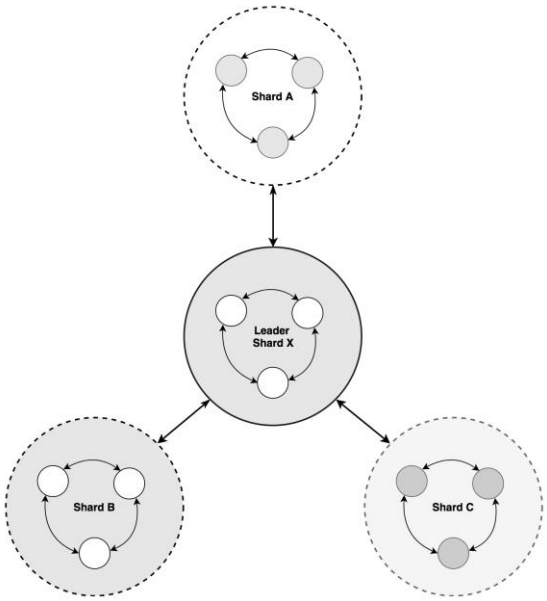

Image №4. ML consensus

All actions in the Management layer are important for the platform and performed by the nodes in this layer to improve the reliability of solutions. All nodes of the platform carry out the verification and approval of decisions made in the ML.

Since the number of platform nodes can be astronomic the conventional methods of approval solutions are inapplicable. Therefore, the project uses a specially designed two-level asynchronous consensus algorithm similar to pBFT [5].

Its first level includes the nodes from some shard. In the case of approval solutions in ML, each such approval first confirmed within the shard using shard protocol of the Resonance consensus. As a result, shard decides on each action as a second level consensus participant and has a confirmed decision by a number of nodes.

A coordination is performed at the second level of new ML block.

Round is a time period from the beginning of block generation until its completion.

The *shards* are the second level participants.

Each round passes the following stages of the block generation algorithm:

1. In each round, a new leader is appointed. The leader is a shard, which will be responsible for the block generation and approval collection. The leader is appointed alternately, according to the position in the shards list. Thus, the first block is formed by the 1st shard, the second - by 2nd shard in the list, and so on.
2. At the beginning of the round, each shard having registered transactions sent for consideration in Management layer sends them to the leader of this round.
3. Having received a list of transactions to be processed in Management layer from the shards within a specified time, the leader rejects the invalid transactions and generates a new block from valid

transactions. Once the block is formed, the leader sends it to all shards for approval.

4. After receiving a block, each shard holds its approval on the 1st level, receiving a coordinated solution among its nodes. The number of nodes approved a new ML block is sent back to the leader.
5. The leader receives a response from shards with approvals. Once the leader will collect approvals of the total number of nodes greater than 50%, the leader declares the block completed and sends it to all shards together with confirmations.

The new ML block cannot be created in this round in case of:

- at the beginning of the round, a shard which is to be the leader of this round is not functioning;
- the leader shard is captured by the attacker and includes an invalid transaction to the new ML block at the 3rd stage;
- the leader is unable to collect the required number of approvals from the total number of nodes greater than 50%;

Then, the creation of ML block might be initialized from the beginning. The next round must be started with a new leader – a shard next on the list (or the first one if the previous was the last one).

Shards Layer

All platform nodes, in accordance with the sharding idea, are divided into separate mini-blockchains called shards. The combined set of shards and their interaction in Power_blockhain performs transactions, just like in a classic blockchain.

Shard is a blockchain with a small number of nodes operating based on the Resonance consensus algorithm. We call it a mini-blockchain to emphasize that its size is small compared to the Management layer blockchain.

Assume that the system includes n shards, system users are distributed evenly across all shards. Then the number of users in every shard will *cshard = cusers_total / n*. Let the user may choose a shard to send a transaction with equal probability among the shards. Then the probability that the transaction will be sent to its own shard will be *1 / n*. Given *n = 3*, we get the probability that the transaction will not leave the sender shard = *33%,* and if *n = 100*, the probability of it becomes 1%. As can be seen for systems such as The Power (with a large number of parallel running Blockchains), a top priority is to ensure effective and safe inter-chain interaction.

For the purpose of all the shards together to perform as a single platform, the shards operate in a standardized way that allows for fast and reliable inter-chain interaction.

Single address space

In the classic "monolithic" blockchain platform all of its customers interact directly with each other using the addressing rules of a particular platform. The interaction between customers of different blockchain platforms even built on the same code (forks) is difficult. The difficulty is not only due to the fact that the addressing rules may differ, but even if the rules are the same (just as in forks). In the first case, it is not clear how to address from one blockchain to another, in the second case it is unclear to which of two blockchain platforms this address may apply.

Despite the fact that the platform is based on a set of shards, it must look for the client as the single-space blockchain platform in usage. This requires that all shards are working in a single address space. Enabling a single address space for customers in all shards we also solve the problem of addressing between clients connected to different shards. This becomes a

problem of finding customers in shards and routing between shards.

Therefore, the client sending a transaction doesn't need to worry in which shard is a recipient of the transaction, when the shard recipient will receive information on customer transactions. Everything is managed by shards themselves.

In contrast to platforms where customer's address is a cryptographically processed public key, The Power platform uses common accounts registry to ensure a single address space. This method provides great opportunities but doesn't make the system too complex:

- Address easily readable by a person and transferable to the other party using conventional methods.
- Versatility of cryptography type use. When adding new methods of asymmetric cryptography, there is no need to abandon old addresses.
- Simpler methods of the multi-signature use or even multi-signatures with hierarchy use.
- Smaller address size, especially in cases of multi-signatures and high bit depth cryptographic keys.

The system accounts addresses are divided into two types: used in private shards and in public shards. Any addresses stored requires 8 bytes from MSB to LSB. The highest three bits are used to indicate the address type (private or public). A value of 100 indicates a public address, 101 - private. Any other values of the three highest bits make an address invalid.

In general, the address has the following structure:

- for public address

  100GGGGG GGGGGGGG GGGBBBBB BBBBBBBB BBBBBBBB AAAAAAAA AAAAAAAA AAAAAAAA

- for private address

  101BBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB AAAAAAAA AAAAAAAA AAAAAAAA

Where:

101 and 100 - type of address (private or public);

G - public address group ID;

B – address unit ID;

A – wallet ID in a unit.

To be human-readable, the addresses can be presented as text or hexadecimal format. The hexadecimal addresses are converted according to the rules of conversion from binary to hexadecimal notation.

Converting to text is performed according to the following rules:

- public addresses have the following format

  AADD LLLL LLLL LLLL LLCC (20 symbols length)

Where:

AA - two letters of the Latin alphabet, encoding two MSBs of the decimal representation of the group identifier in a number system to base 26.

The values of bits are coded as follows: A-0, B-1, C-2, ..., Z- 25;

DD - two decimal LSBs of the group-identifier;

AADD – corresponds to G bits in binary representation.

L - decimal representation of block identifier and wallet in a block;

SS - checksum calculated by CRC32 algorithm.

Since the bit representation can not be accurately converted to decimal and 26-based, the conversion in AADD group may generate not more than 2^{16} options, and in the group LLLL LLLL LLLL LL - not more than 2^{45}.

format HHHHHHHHHHHHHHHH CC (18 symbols) is used for private addresses, wherein:

H - hexadecimal representation of block ID and wallet ID;

SS - checksum calculated by CRC32 algorithm.

Registering of the address occurs after sending specialized transaction in the shard. In this transaction, public key/keys might be specified, as well as the results of the PoW (proof-of-work) might be shown. The PoW task is needed to counter the massive addresses registering attack. In case of high-speed address registrations in the system, the complexity of the PoW task will be raised by the Management layer. If the speed is normal, the PoW complexity will be reduced.

To enable smooth registration of public address, the Management layer assigns to each shard a pool of addresses G and B bits in the address mask. In case of imminent exhaustion of the address pool, the shard requests a new pool from the Management layer.

Avoiding double spending

Due to the fact that most transactions are performed at the Shards Layer, and all the actions take place in separate shards, there are two options of "double spending" attacks.

The first option occurring in any classic blockchain platforms is an attack on the superiority on a particular time to enter invalid information in a new block, such as re-spending. In our platform, protection from attacks on the Shards layer is implemented through a Resonance consensus algorithm. At the platform level, the Validators layer is responsible for this problem. See detailed information in Validators layer section.

The second attack option abuses the feature described above - a single address space. The attacker may try to make expenses in several shards at the same time. Since the shards operate independently without receiving information on spending, they could have registered such transactions. To prevent these types of attacks, an additional rule is introduced in the platform:

The customer transaction is carried out only in the shard in which it is registered.

This rule is very similar to the behaviour of the phone in today's cellular networks. Despite the availability of several radio spots, the cell phone automatically selects one and registers there to receive and send calls to the network.

Just like with a cell phone, the platform customer is free to choose the transition between any shards.

Inter-chain interaction

The block validation mechanism is similar for all the shards. This allows you to check a transaction from one shard in another shard. All nodes contain information about the system structure.

Triple recording rule

In order to guarantee the delivery of the transaction between shards, the platform uses the following mechanism called a "triple recording rule":

- The client sends a transaction and a transaction signature to the shard node in which he/she is currently registered. The node launches a consensus algorithm and transmits the transactions and signatures received by it to other nodes of the shard in order to include all new transactions in the new shard block. When forming a new data block, all transactions, wherein the transaction recipient is not registered in the same shard as sender, allocated in a separate section of the new data block. Thus, the new data block includes two sections:

- section of the new data block including transactions which sender and recipient are registered in the same shard;
- section of the new data block including transactions which sender and recipient are not registered in the same shard.
- After adding a new block in blockchain of sender shard, the nodes extract from this block data on all transactions with their MTP (Merkle Tree Proof) from the 2nd section. Then the node generates outgoing transactions packages with their MTP for each recipient shard. Each transaction package with their MTP contains the packet number for the recipient shard, address of the recipient shard, the title of the new data block, as well as transactions with their MTP.
- Then, the nodes belonging to the sender shard transmitted via communication channels all generated packages to at least one of the recipient nodes.
- When the recipient shards receive packages with transactions from sender shards, the nodes of recipient nodes shards first check the number of each transaction package. If the package number from sender shard exceeds the number recorded in the previous recipient shard of transactions package into two units or more, the nodes of recipient shards send the received packets with the transactions to the queue. Upon receipt of missing transaction packages, they go to the next step from the queue.
- The package with a number next after previously registered in the shard of recipient of the package is validated by the receiving shard nodes through the validation of all transactions contained in the package.
- After validation, the transaction package is included in the new shard recipient block, and the transactions of this package are applied to the receivers.
- After including the block with the transactions package to the recipient shard blockchain, the recipient shard sends an information to a sender shard information on including a transaction package in a new recipient shard data with confirmation in the form of MTP.
- When a sender shard node receives an information on including transactions package into a new data block of recipient shard with MTP, a sender shard node validates this MTP. After validation by the sender node shard, the information on including transactions package with MTP in the new shard block data, the node sends this information to be included in the new sender shard block. After this, the transaction registration is completed.

Let us describe the above process simply: the information about the transactions route is recorded three times. *First*, the transaction is registered in the sender shard (where the client is registered). *Second*, in the recipient shard where the transaction also finds its recipient customer. The *third* record is performed in the sender shard - the fact that the transaction has been successfully transferred is recorded. While this record has not appeared in the sender shard, its nodes will try to bring the transaction to the sender shard.

Private shards

The private shards structure is somewhat different from the public shards. The main difference is that all of the nodes composing a private shard are controlled by private shard owner. The private shard has no validators, as the shard owner does not want to provide independent validators access to its data and there is no need for its own validators.

Although ensuring the information validity and smart contracts within the private shard is

performed by the owner, there is still a problem of ensuring the immutability of the data recorded in the shard. To do this, we use a well-known anchoring mechanism. As has already become a classic method, a hash of the next block is located within a public blockchain thus placing the block hash in the protected environment.

However, in this case, the classic method is inapplicable: in order to confirm the validity of the information in the block located in the chain between blocks with anchored hashes, all the block chains might be collected till the block with anchored hash, and then all the hashes to be counted to verify the validity. To simplify interactions with honest shards, we offer to produce anchoring of each private shard block in the public space - Shards Layer. This seriously simplifies the interaction between the private and the public shards in the platform.

Also, the anchoring mechanism for private shards has been expanded the system. The public level of Shards Layer, upon receiving specialized anchoring transaction, verifies it, specifically:

- block number – the shard will not accept hash of a block with number different from the number next to the last block of the registered shard;
- block signature - the transaction should include signatures of nodes created a block. It checks whether the nodes signed this block are legitimate and whether the consensus has been achieved.

The owner of the private shard may record in the blockchain an information on the public key of the legitimate nodes of its shards. This eliminates the possibility of attackers to create their blocks in a private shard by adding illegitimate nodes. When checking the transaction, it will not be considered valid.

Another difference between the private and public shard: the public blockchain considers itself as a common public account, in which all of its assets are stored. So it is impossible to generate an asset and give it to another public account within the private shard. Only those assets that had previously been transferred to the account of the private shard can be transferred to other customers with public accounts. All tokens generated within the private shard can be transferred only within it.

Validators layer

This system logic level also consists of all the system nodes, but its nodes perform different functions. Each shard node must be a validator of transactions in other shards. Thus, by grouping validators nodes, the platform creates a system that performs verification of actions in all shards.

The shards sizes are small enough compared to the total number of nodes in the system. This aspect theoretically leaves attackers an opportunity to negotiate among themselves to have control over 50% + 1 node. If the attackers can take control over such percentage of shards, they will be able to carry out an unsecured invalid inter-chain transaction. It turns out that the security of the entire system depends on the security level of the most vulnerable element (shard). In order to make this kind of attack impossible, we have implemented the second layer.

So, how the role of validator will help to solve the security issues? Each shard corresponds to a group of validators (nodes from other shards). This group verifies the blocks within the shard and issues verification signature of that the validator checked the block and considers it valid. Each block is considered valid only if more than half of validators issued their verification signature. Thus, to make shard able to send unsecured transaction, the attacker must control 50% + 1 node in shard and 50% + 1 validators.

Therefore, the next important step in improving the system security is a validators assignment scheme, which would minimize the chances of attackers to get simultaneous access to 50% + 1 node of attacked shard and validators. Unlike the shard nodes, its validators should change periodically. Since the permanent list of validators increases the risk of capture by the attacker, the risk of capture is increased with the time passed since the validator assignment. Also, lists of validators must be generated randomly, which allows the possibility of captured validators to be distributed more evenly.

These rules help to achieve the main goal - increase the cost of an attack on the system. Thanks to them, the attack on a weaker platform subsystem (shard) will not break its operation. Only taking control over a large number of nodes in the system, increases the probability of system operation fault proportionally to the percentage of the captured system nodes. Thus, the described system's security achieves almost the same level as that of a PoW consensus projects, where the probability of double spending attack with 30% of the units exists, but very small.

When operating validators within a shard, it is important to pay attention to validators scheduled rotation. When the validator changes a checked shard, the validator is no longer required to store data of the old shard blockchain. It must connect to a new shard and get a full copy of its data. Obtaining a copy of the shard is quite time-consuming uses significant network environment. If the validators change is carried out simultaneously, such action may not be less severe than the DDOS attack. Therefore, it is important that a change of validators for the whole system was carried out at regular intervals. The platform currently expects that the validators team every hour rotates 10% of nodes, so the validators group fully rotates within 10 hours.

The systems using validators have a predicted vulnerability [8] described in the document "Verier's dilemma and attacks". The attack principle is quite simple - an attacker creates a transaction with such input data that its validation takes much more time than any ordinary transaction. Thus, the attacker seeks to ensure that validator would be economically interested to stop testing and accept the results of an attacker "as is" to receive compensation for the current validation and move to the next block validation. As a result, the validators pass such a transaction, not even knowing if it contains incorrect data.

The Resonance Consensus Algorithm is synchronous, which means that each shard has set execution time for the block calculations. Thus, if the validator detects that the validation execution time exceeds the calculated value, it means the attack "Verier's dilemma and attacks". Despite the fact that the Power_blockchain validator easily detects such an attack, it could still choose from two strategies:

- Despite the attack detection, to approve the block and get a reward for it. The disadvantage is that a successful attack will damage the project's reputation, which may affect the profitability of the node's further work in all roles (and block generator and validator). In addition, the blockchain keeps information about validators confirmed an invalid block, which means the economic penalties for such validators. The results for the validator: less than $ 0.01 fee per block approval, the penalty for the approval of the invalid block (now the penalty time is expected in the amount of 100% pledge).
- Upon detection of an attack, to report about it to the Management layer for general inspection of shard and block containing an attack. After the settlement of the situation,

the validator receives reward for actions on protecting the system security. Results for the validator: reward (now it is 10% of the pledge amount).

Since the second option is an economically advantageous strategy, the majority of independent validators choose the second option, thus activating the overall shard verification. This means that the probability of such an attack is very low on the platform.

To prevent capturing of 50%+ 1 node of validators by an attacker it is important to the function a pseudo-random number generator (PRNG) of validators allocation into groups had the best possible quality so that the attacker could not influence and even predict the future distribution. However, the blockchain systems have problems with generation of truly random numbers based on the data already entered in the blockchain. Since these data are known to all participants, PRNG rules are known, and any participant can predict the result of the PRNG. To avoid such "random results", we must add additional sources of entropy within the generated block. The platform offers the following mechanism for obtaining sources of entropy: during the creation of each block, the nodes add to the block a mark of entropy - the hash signature of the previous block hash. Since only the owner of the node knows its private key, and it is economically interested to keep it confidential, the hash value of the previous block before publication is only known to the owner. Since the Resonance consensus algorithm is synchronous, the block includes at least 50% + 1 node independent sources of entropy.

The likelihood that an attacker can take control over a team of validators can be found using a binomial distribution§ For the team of N participants, if an attacker captured p percent of the total number of validators in the system, the probability that more than half of the team in this round will be controlled by an attacker will be equal to:

$$\sum_{k=\frac{N}{2}}^{N} p^k (1-p)^{N-k} \binom{N}{k}$$

As the standard for the system, we assume $N = 100$. In this case, the probability of capturing more than 50% of the attacker's units equal to:

Percentage of validators controlled by an attacker	40%	33%
Probability of taking control over the validators group operation	0.0271	0.0004

If the shard nodes coordinate their actions, they can increase the number of validators group members. This will reduce the likelihood of an attack, but also a little slows down a block approval. For N = 150 we get:

Percentage of validators controlled by an attacker	40%	0.0082
Probability of taking control over the validators group operation	0.0271	1.83 * 10-5

The main differences of validators from blocks generators in a shard are:

1. The validators do not need to constantly keep a copy of the validated shard. The validator synchronizes its copy of the shard data during the validation, and deletes data during the transition to another shard.
2. Validator is not involved in the block generation, but only validates it. Therefore, compared to blocks generators, it performs less computation.

Bibliography

1. Bitcoin

2. Ethereum

3. Trilemma of scalability

4. Leslie Lamport, Robert Shostak, Marshall Pease. The Byzantine Generals Problem

5. Miguel Castro, Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery

6. Loi Luu, Jason Teutsch, Raghav Kulkarni, Prateek Saxena. Demystifying Incentives in the Consensus Computer

7. Binomial distribution

8. Ittai Abraham, Srinivas Devadas, Kartik Nayak, Ling Ren. Efficient Synchronous Byzantine Consensus

9. Kevin Driscoll, Brendan Hall, Håkan Sivencrona, Phil Zumsteg. Byzantine Fault Tolerance, from Theory to Reality

Appendix A

ATTENTION

In case you do not agree with following terms of the non-disclosure agreement, please close this Yellow Paper and remove it from your computer in such a way that it can not be restored. Continued reading of the text of this document means your full and unconditional acceptance of the confidentiality conditions. You fully understand and assume obligations to maintain confidentiality and are ready to bear full financial responsibility for any uncoordinated disclosure of the information contained in this Yellow Paper or a part of such information.

NON-DISCLOSURE AGREEMENT

This Non-Disclosure Agreement (hereinafter Agreement) is entered into by and between:

Decentralized Technologies OÜ, registered under the laws of Estonia under registry number 14502188, having its registered office at Estonia, Tallinn, Lasnamäe 4c-11, 11412 (hereinafter Disclosing Party), represented by a legal representative Maxim Mikhaylenko

and any person, who read this Yellow Paper document (hereinafter Receiving Party)

(Disclosing Party and Receiving Party hereinafter separately also Party or collectively Parties).

NOW, THEREFORE, the Parties have agreed as follows:

5. For purposes of this Agreement, Confidential Information shall include any information, in whatever form, relating to the Project and/or business affairs of the Disclosing Party, which is delivered by the Disclosing Party, including but without limitation to, information of a technical, operational, administrative, economic, planning, business or financial nature, as well as data, software, trade secrets, copyright, intellectual property and know-how and any other information whatsoever of confidential nature, in whole or in part. All information in this Yellow Paper is confidential too.
6. The Receiving Party hereby warrants to keep confidential and not to use, disclose, enable or cause any third party, without prior written consent of the Disclosing Party, to become aware of any Confidential Information.
7. The Receiving Party shall carefully permit access to Confidential Information only to its representatives and employees to whom such access is reasonably necessary for execution of the Project and shall require those persons to sign non-disclosure restrictions at least as protective as those in this Agreement. The Receiving Party agrees that it shall remain liable for any breach by its representatives and employees of the obligations set out in this Agreement.
8. All Confidential Information disclosed under this Agreement shall be and remain the property of the Disclosing Party and nothing contained in this Agreement shall be construed as granting or conferring any rights to such Confidential Information on the other Party. Nothing in this Agreement shall be deemed to grant to the Receiving Party a license expressly or by implication under any patent, copyright or other intellectual property right.
9. The Receiving Party shall at its expense, at the written request of the Disclosing Party, cease to use and immediately destroy or return to the Disclosing Party any and all records, notes, and other written, printed, or tangible materials in its possession pertaining to the Confidential Information.
10. The Receiving Party's obligations under this Agreement do not extend to information that is: (i) publicly known at the time of disclosure or subsequently becomes publicly known through no fault of the Receiving Party; (ii) discovered or created by the Receiving Party before disclosure by the

Disclosing Party or learned by the Receiving Party through legitimate means other than from the Disclosing Party, as proven by its written records; or (iii) disclosed by Receiving Party with Disclosing Party's prior written approval.

11. Should the Receiving Party breach any of the obligations set forth in this Agreement, the Disclosing Party shall have the right to request the Receiving Party to (i) immediately terminate such breach; (ii) surrender to the Disclosing Party any revenues received in connection with such breach; (iii) pay to the Disclosing Party a contractual penalty for each breach in the amount of EUR 10,000; and (iv) compensate the Disclosing Party for damages caused by such breach (to the extent they exceed the above penalty).
12. This Agreement expresses the complete understanding of the Parties with respect to this subject matter and supersedes all prior proposals, agreements, representations, and understandings.
13. If a court finds any provision of this Agreement invalid or unenforceable, the remainder of this Agreement shall be interpreted so as best to effect the intent of the Parties.
14. This Agreement shall remain in force for an indefinite period of time.
15. This Agreement is governed by the laws of Estonia excluding the choice of law rules. Any dispute, controversy or claim arising out of or relating to this Agreement shall be resolved by Harju County Court, Tallinn, Estonia.

ATTENTION

In case you do not agree with these terms of the non-disclosure agreement, please close this Yellow Paper and remove it from your computer in such a way that it can not be restored. Continued reading of the text of this document means your full and unconditional acceptance of the confidentiality conditions. You fully understand and assume obligations to maintain confidentiality and are ready to bear full financial responsibility for any uncoordinated disclosure of the information contained in this Yellow Paper or a part of such information.