#metahash
CONCORDIA RES PARVAE CRESCUNT

Yellow Paper

Technical documentation

MetaHash AG,
Gartenstrasse 6, 6300 Zug
Switzerland

Version: 1.0 (220718)

Abstract

2017 was the year of decentralized technology boom. Decentralization paradigm addresses several key issues: the issues of trust in the system and controlling authorities, systems scalability and reliability, and the need for third-party involvement. The decentralization platform and app market, however, is limited by two major factors. First, geographic distribution of nodes over insufficient connection infrastructure causes slow data synchronization and validation. Second, the vulnerability of distributed networks to 51% attack[1] leaves much to be desired in terms of security and trust.

#MetaHash overcomes these obstacles by introducing dynamic node roles and implementing a fragmented cryptographic proof. While simultaneously decreasing the volume of transmitted data, this approach optimizes network synchronization, and solves issues of transcontinental data transmission and archival. Furthermore, this approach has lead to the creation of #MetaPoS, a multi-layered consensus system, enhancing the network resilience to 51% attack up to 90%.

Utilizing the proprietary #MetaSync and #MetaPoS technologies, the #MetaHash team created #TraceChain, the fastest blockchain protocol as of today. Load test results show throughput above 50,000 transactions per second[2]. Methods and results can be found within this document.

#MetaHash is the next-generation high-performance and low-fee blockchain. High confirmation speed and low data processing fees enable support for micro-transactions, and allow operating decentralized applications in near real-time.

1. https://www.investopedia.com/terms/1/51-attack.asp
2. In a broad bandwidth (up to 600,000 transactions per second)

Project Goals

Short-Term Milestones

1. High-speed multi-blockchain based on #TraceChain protocol:
    a) Design geo-distributed network architecture with minimum synchronization time while maintaining maximum data transfer capacity.
    b) Select a consensus model allowing for minimum transaction confirmation time while maintaining transaction immutability.
    c) Keep the cryptographic proof of asset ownership while decreasing amount of stored and transmitted data.
    d) Select an optimal balance model.
    e) Mitigate excessive data storage issues within the time-proven blockchain network structure.
    f) Allow nonlinear validation of blockchain data.
    g) Avoid hard forks while deploying major software updates.
    h) Solve the issue of blockchain merges after network split.
2. Decentralised storage
3. #MetaApps: virtual app hosting
4. #MetaDNS: distributed name service
5. #MetaGate: multi-currency wallet and DApp browser

Mid-Term Milestones

1. Mobile resource sharing (cellular and satellite communications, data transmission). Consensus for mobile devices
2. DApp Marketplace
3. Digital identification with biometric data
4. IoT capabilities and compatibility
5. Universal reputation system

#MetaHash Technical Features

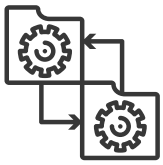
The exceptional technical characteristics of the #MetaHash network result from two innovative technologies: #MetaSync and #MetaPoS. #MetaSync data synchronization subsystem allows complete and integral data updating around the world in under 3 seconds, while #MetaPoS multi-consensus subsystem allows generating and validating the blocks in parallel with the synchronization process. Each subsystem is thoroughly described in the following sections. The test methods and results are provided below.

Experimentally proven features of #MetaHash include:

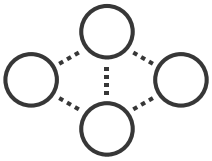
Transaction capacity: over 50,000 transactions per second

Storage required per 10,000,000 financial transactions: 2.5 GB

Block generation time: 1 second



Network synchronization time: up to 3 seconds



Minimum number of nodes: 4

Testing Methodology

The tests involved 244 nodes across 5 continents. A wide range of node parameters were tested: from low-performance (1 vCpu, 256 MB RAM) to high-performance (40 vCpu, 512 GB RAM). Tests were conducted for 3 months, during which time more than 500 billion transactions were processed.

Test 1. Building the network

This test evaluated the process of network building. Prior to network build, #TraceChain evaluated the effective performance of each node and speed of data transmission between nodes. In mainnet, such tests are randomly conducted for each node in the background. Following the test results, #TraceChain AI algorithm, active on all nodes, computed an optimal network map and assigned either of the following roles to each node: Peer, Verification, Core, or Torrent. *Fig.1* shows a graphic image of the resulting network map. The signal travels from the outer layer towards the center (Peer node to Verification node and to Core node).

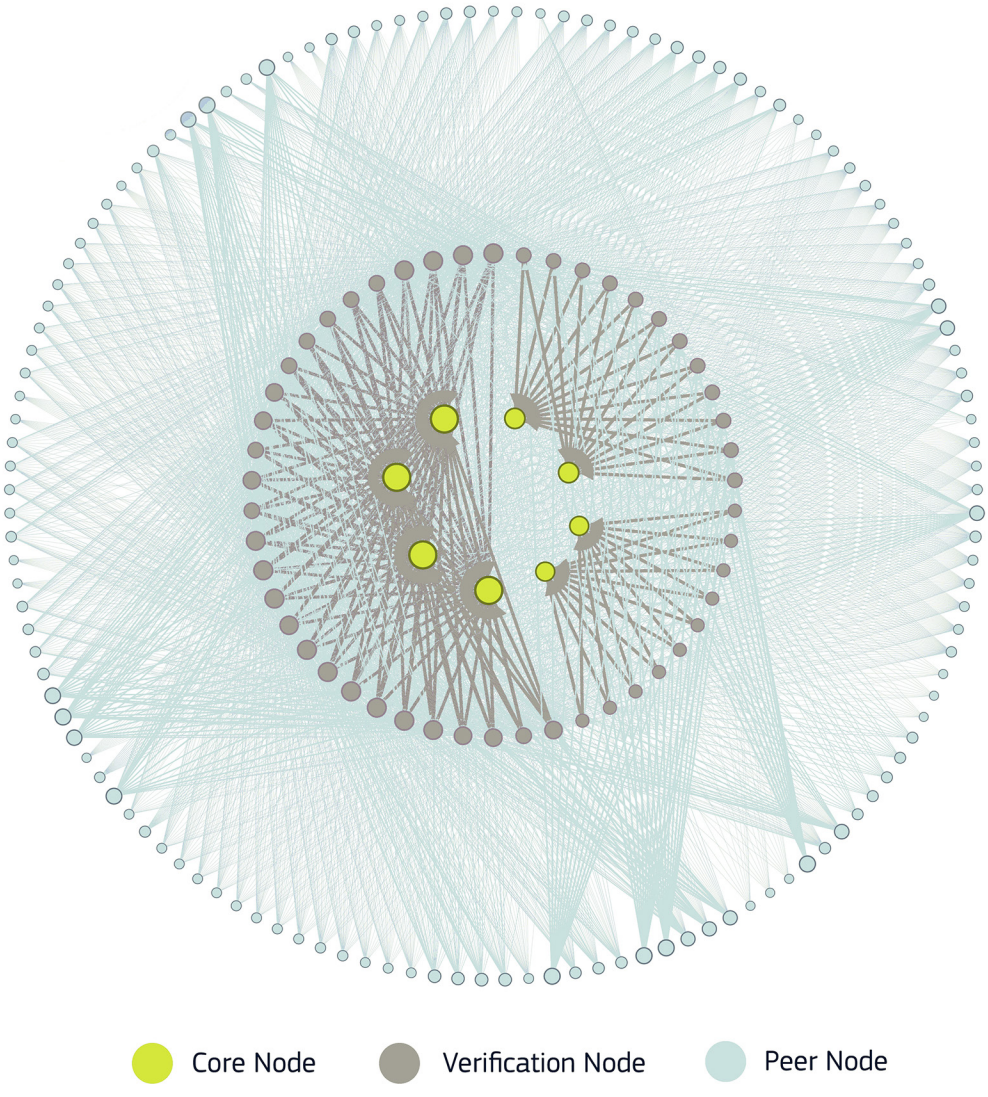


Fig.1. Network map

Test 2. Bulk transactions verification latency

This test studied the dependence of transaction time on the number of transactions. The transaction time is defined as the total time from the transaction creation until the complete propagation of the processed transaction across the entire network. An example of a transaction propagation path across the network is shown in *Fig. 2*. The number of transactions sent varied from 1 to 600,000 per second.

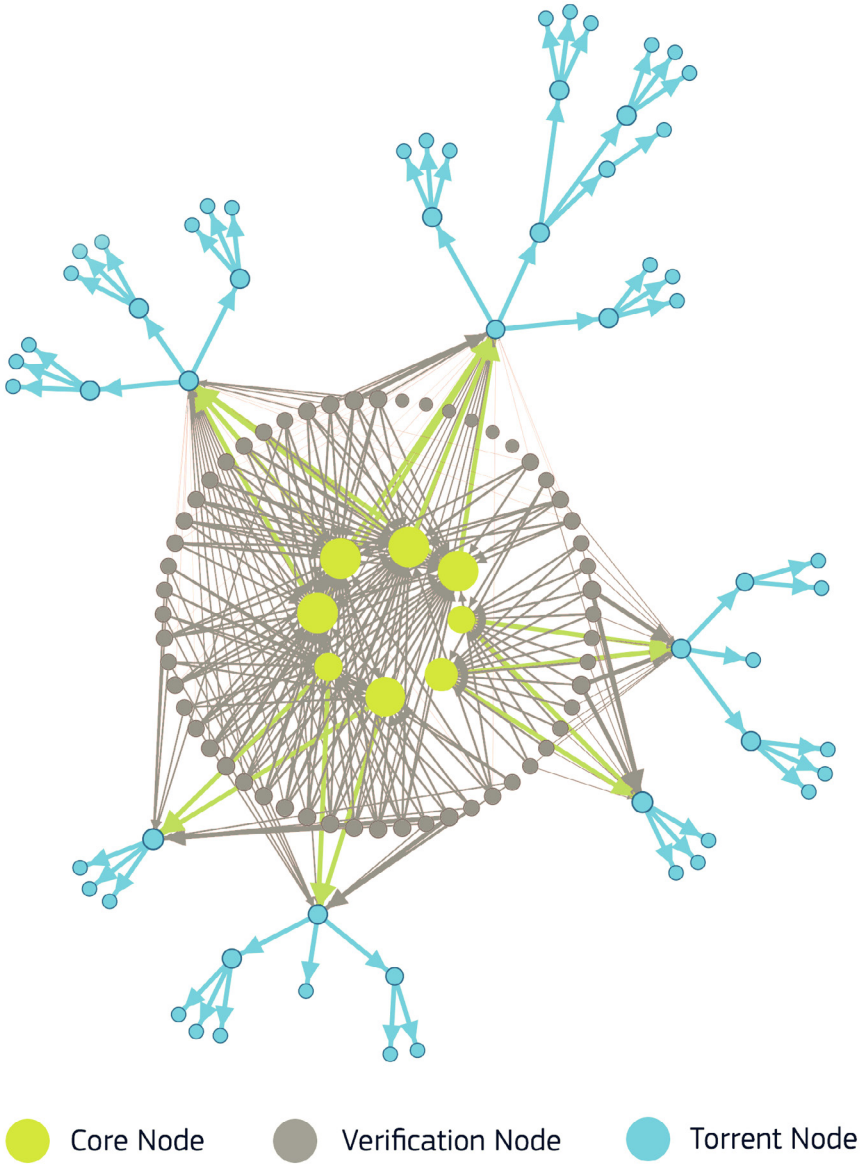


Fig. 2. Transactions propagation from Core to Torrent nodes

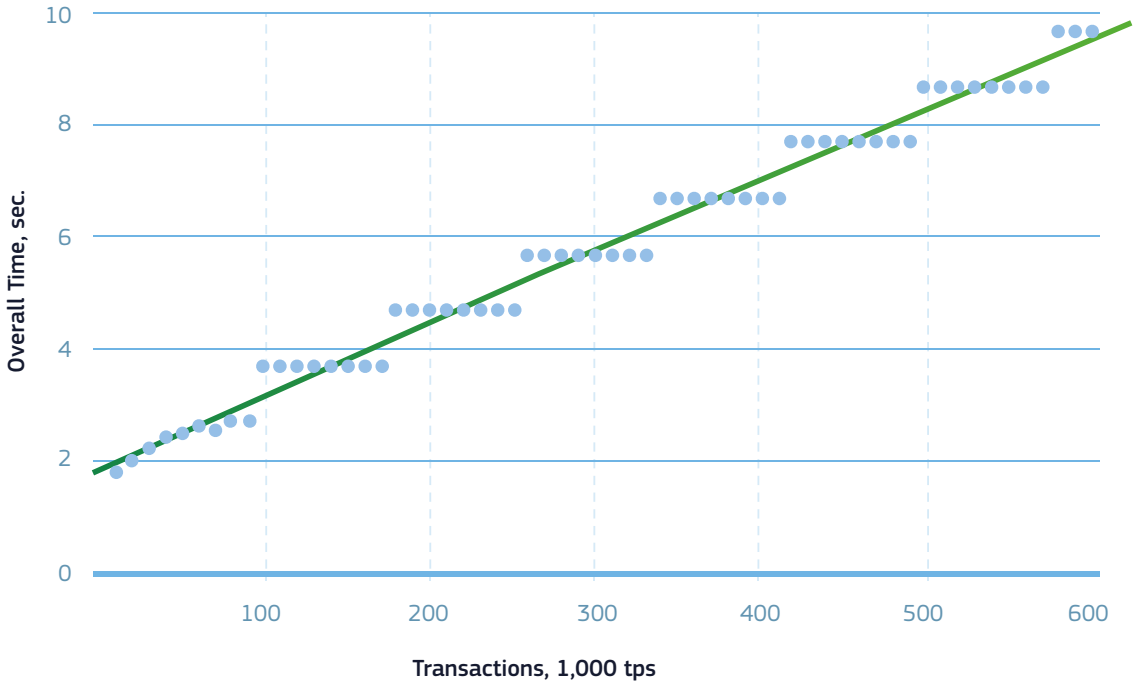


Fig 3. Verification time over the number of transactions

The blue dots in *Fig. 3* denote the acquired experimental data. 80,000x transactions marks show the recorded threshold of network bandwidth per 1 second. Transactions that are not yet processed by Core nodes are queued and remain pending until they are processed in the next block. The interpolation polynomial (yellow line on the graph) appears linear up to the mark of 600,000 transactions per second, which proves an excellent scalability of #MetaHash network within the studied range.

Test 3. Network failure

The test measured the network performance depending on the number of lost nodes. A certain percentage of nodes was randomly turned off (10%, 20%, 30%, 40%, 50%, 60%, 70%). *Fig. 4* illustrates the shutoff scenario. The results show that even with as low as 30% nodes active the network is still fully capable of functioning, even keeping the claimed performance capacity, network availability, and transaction verification and distribution speed. Similar results were achieved after a simulation of failure in cluster server collocations (e.g. data center failure, global ISP issues).

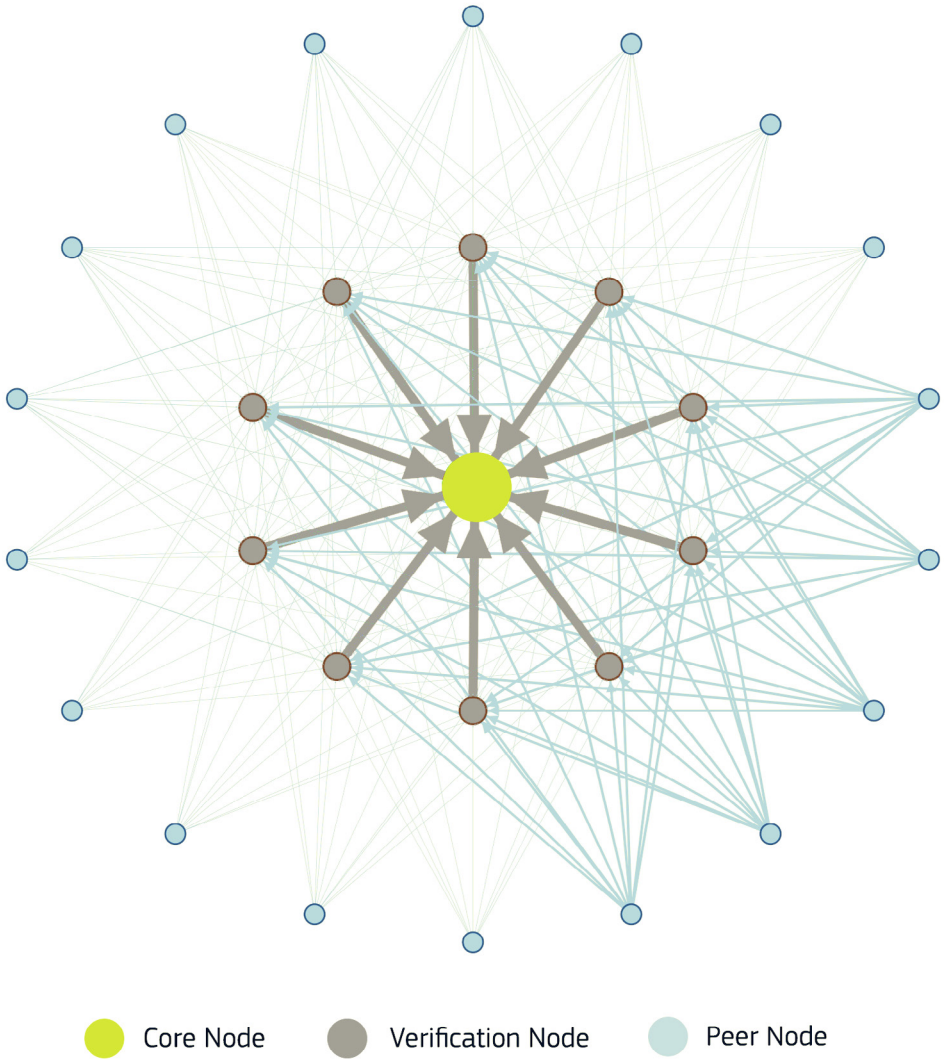


Fig. 4. Partial shutoff of the network

Test 4. Continental blockage and network recovery

This test simulated a continental blockage, which lead to the network being split into two and a subsequent rebuild of each, followed later by a merge between the two. The purpose was to test the speed of consensus reorganization after the split and the synchronization speed after network recovery.

In this scenario, after rebuilding, each network has a complete structure, but with fewer nodes. Each network forms its own chain of transaction blocks, which later will be merged with replacement of conflicting transactions. When the connection between networks is restored, and cut-off nodes are reconnected to the network or new Core and Torrent nodes are added, the effective performance slows down to 30,000 transactions per second during the period of data synchronization. Therefore, the network effectively slows down 2.5-3 times for the duration of the merge.

Testing Results

#MetaHash throughput allows handling of 60,000 to 80,000 transactions per second in a broad range of environmental bandwidth loads (up to 600,000 transactions per second), outpacing the claimed performance of 50,000 transactions per second. Exceeding the incoming transactions rate over the claimed performance level, the processing speed is observed to slow down. Network resilience is maintained for up to 70% of network failure, while further levels show performance deterioration. In the event of a continental network blockage, the networks retain local functionality and automatically rebuild as soon as connectivity is restored. During rebuilding, networks function, however, in a slower mode.

Network Description

#MetaHash is a single rank peer-to-peer network. Every member of the network runs the same software and automatically gets its role assigned. Roles are assigned by the #TraceChain AI algorithm executed on all nodes and described in detail in sections below.

#MetaHash network has 5 key roles:

- #MetaGate
- Peer node
- Verification node
- Core node
  - Master core node
  - Slave core node
- Torrent node

Each role provides specific services in both synchronization and consensus subsystems. To make the #MetaHash operation principles more comprehensible, the subsystems are explained individually below.

#MetaSync Data Synchronization Subsystem

This section describes the #MetaHash network from the perspective of the #MetaSync data synchronization subsystem and network design only.

Network Structure

The role assignment to network members in the data synchronization subsystem is maintained by #TraceChain AI. Role assignment is primarily based on the nodes' physical properties, such as memory, CPU performance and network connection quality. However, performance is not the sole criterion which determines node role. A similarly important characteristic is the node's geographic location.

The data synchronization subsystem roles functionality is described in **Tab. 1**.

Tab. 1. Data synchronization subsystem roles and functions

Role	**Functions**
#MetaGate	Light user-client with an option to use part of the hard drive as an archive storage when forging mode is live.
Peer node	Accepts incoming connections from #MetaGate light clients; has a direct connection to Verification nodes; does not have sufficient processing capacity for synchronization tasks; ensures fast network connectivity with clients and protects against invalid transactions.
Verification node	Accepts incoming connections from Peers only; maintains long-distance or cross-continental connections; maintains a direct connection to Core nodes; has sufficient processing capacity to be used for synchronization; serves as the network communication layer for the Cores.
Core node	Accepts incoming connections from Verification nodes only; has a good connection to Core nodes in other regions; has sufficient processing capacity for fast block generation; controls synchronization processes.
Slave core node	Duplicates Core node operation; will automatically replace the Core node in case of unavailability or failure.
Torrent node	Accepts connections from all roles; has sufficient storage capacity to keep blockchain data; transmits data between #MetaGate and other nodes; may be used to work with the raw data archive; may report balances; may carry out specific tasks to support application functioning.

The network is schematically shown on the *Fig.5*.

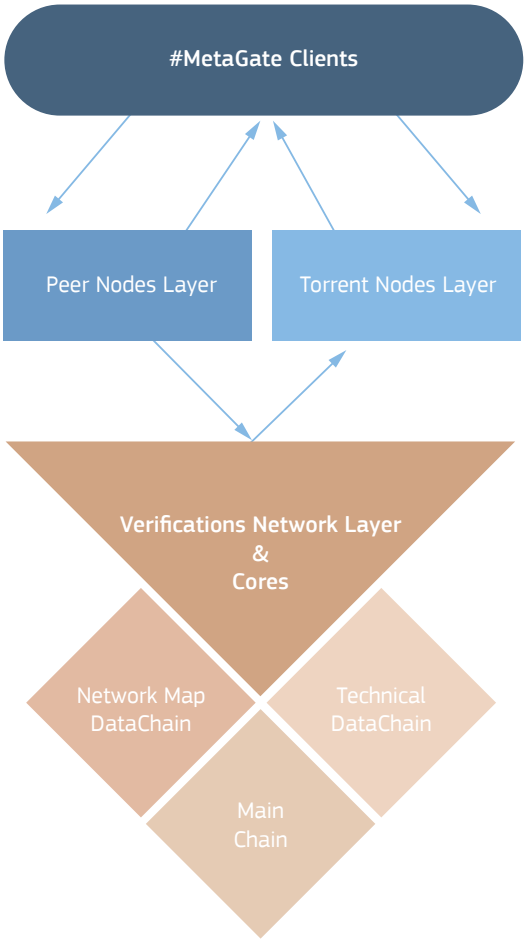


Fig. 5. Network diagram

Network member's role is not static. Members may change their roles based on the criteria listed above and #TraceChain AI-based behavioral analysis.

Trust

#MetaHash network implements the node's confidence factor defined as Trust. Trust is used as weighting factor in role assignment and voting influence. Its value depends on the duration of node's operation in network, node's performance, absence of network failures and errors occurred due to this node. Trust is described in detail in "#MetaHash Multiple Proof of Stake" section.

Network Topology

Data routing is the most crucial task for optimizing synchronization, therefore, #TraceChain AI keeps the network map dynamically updated. Each member generates its own map of the entire network, and updates it continuously, to speed up data delivery. This map is used for choosing the optimal operator of a node's own tasks as well as those of its clients.

Network maps generated by nodes are signed with account for their Trust and Stake (the voting process is described in chapter "#MetaHash Multiple Proof of Stake") and sent to Network Map DataChain for public use.

The client generates its own network map while interacting with the various members of the network. Thus, each client may choose the best operator for its request - that is the operator with the highest level of Trust and best connection speed for this specific client.

In the event of a node failure automatic voting for roles redistribution takes place, based on the network maps and nodes' Trust.

To achieve an optimal network map #TraceChain AI queries every node for performance and intra-node transfer speed data. Thus, #TraceChain AI randomly puts some nodes into a testing mode to get this information. Tests are specifically designed in a way that a compromised node will be unable to spoof the network by providing higher performance and speed values, since it will be unable to understand that the simulation is on. This is achieved by providing the information on running tests only at the final stages of interaction with the node.

Upon first connection to the network #TraceChain uses a list of node addresses received via DNS. After the network map is finalized any further calls are carried out in accordance with this map.

Network Map

The network map state is defined in a block of a separate blockchain. Each block describes the state of all members in the network, relevant topology, performance test results and Trust.

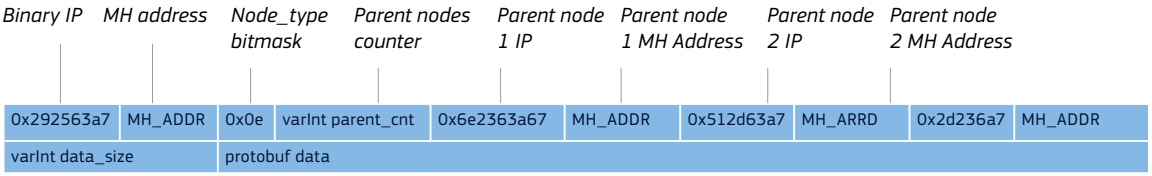


Fig.6. Object scheme

Tab. 2. Object data structure

Field	Description
Binary IP	Node IP in binary format
MH address	Node's wallet address with some coins, or the one the coins have been delegated to
Node_type bitmask	Bitmask for the current node's role: • master • slave • verif • proxy • torrent-full • torrent-tx • torrent-balance • torrent-bkp
Parent nodes counter	The number of parent nodes [**type: varInt**][3]
Parent node 1 IP, Parent node 1 MH Addr, Parent node 2 IP, Parent node 2 MH Addr, etc.	The list of parent nodes corresponding to the number of nodes in **parent nodes counter** and sorted by priority. It is required to specify the following parameters for a parent node: • Node IP • Node's wallet address with some coins, or having coins delegated to it.

3. varInt - proprietary type of integer variable length data developed by #MetaHash similar to another example

The type of parent node depends on the current node role of a network member according to *Tab. 3*

Tab. 3. Roles and parent nodes compliance

Role	Parent nodes
Core node	Slave core nodes
Slave core node	Torrent nodes
Verification node	Nodes the Verification node is giving the information to
Torrent node	Nodes the Torrent node receives the data from

Following the node description there is a protobuf-serialized[4] binary list of reports on the test results of other nodes, signed by the private key of this node.

For every node tested the results are saved in a format described in *Fig. 7* and *Tab. 4*.

Fig.7. Tests section schema

Tab. 4. Data structure of Tests section

Field	Description
MH_ADDR	Node IP in binary format
Ping ms	Ping in milliseconds
tps	Transactions per second
mbs	Network channel bandwidth
sign/sec	Signatures per second
ram	RAM available
rate	Node's Trust

The signature and public key are indicated at the block's end to allow checking the validity of test record.

4. protobuf - method of data packing into binary code proposed by Google: https://developers.google.com/protocol-buffers/

#MetaPoS Multi-Consensus Subsystem

In any blockchain system, it is critical to define a state of consensus - a state wherein all members of a network agree to a proposed change of, or addition to, existing blockchain records. In the section below the network roles within #MetaHash are described exclusively from the viewport of the #MetaPoS consensus subsystem in the context of existing blockchain technologies. The section details the MultiPoS consensus model, cryptographic protection, the blockchain structure and content, transactions and the #TraceChain protocol capabilities.

Two key requirements have been outlined when writing an algorithm for a consensus model: reaching consensus at the highest speed possible, and providing improved security from corruption within the network. At the time of writing, the Delegated Proof of Stake (DPoS)[5] was the fastest consensus model, tested by many popular platforms. DPoS proposes that each member in a network holds voting power (Stake), equivalent to the value owned by the member. The member may then grant their Stake to a trusted delegate. This speeds up the voting process, as the number of delegates is generally small. Unfortunately, in practice it often appears that large amounts of Stakes are collected by a limited group of people, thus, neutralizing the network decentralization.

The #MetaHash team has improved the DPoS model and developed a new algorithm - the Multiple Proof of Stake (MultiPoS) consensus model. In this model, multi-layered validation provides the basis for protecting a network against corruption. In cases when the central network entities, which generate blocks, appear to be corrupted, the rest of the network may vote to rebuild the network and to re-distribute the roles, thus, neutralizing the threat. Furthermore, the block-creating cores, while managing synchronization, are not static. They vary according to dynamic network parameters, and are not granted final validation rights. Also, this model allows to run validation and block distribution processes in a parallel decreasing consensus time. This innovative method of validation and rights management is at the core of #MetaPoS technology.

5. https://bitshares.org/technology/delegated-proof-of-stake-consensus/

#MetaHash Multiple Proof of Stake (MultiPoS)

Roles and their functions in #MetaPoS consensus subsystem

1. Peer node. Receives transactions from clients, checks transaction validity, sends transactions to verification nodes. Does not keep the blockchain state in memory, therefore, does not require significant computing resources.
2. Verification node. Checks the validity and economic feasibility of transactions received from peer nodes. Keeps records of all transactions processed. Requires computing resources.
3. Core node. Accepts transactions from verification nodes, queues transactions for block generation. Generates blocks. Used for consistent information sharding.
4. Slave core node. Provides post-verification of signed blocks. May substitute for a core node, in case of core node failure.
5. Torrent node. Distributes blockchain information.

Reaching consensus

Core nodes are key players in the consensus. They compile transactions into blocks. The following section addresses node synchronization processes, block generation, the network consensus process, and the distribution of blocks to torrent nodes.

Key definitions

Tr - A transaction within the network (see next section for detailed description). Transactions may undergo a validation process. Let the symbol Θ denote a signature validity check. Thus, we may denote a transaction validity check as $\Theta(Tr)$. This function returns a $True$ value when a transaction is valid, and $False$ otherwise. Let us also define methods to obtain elements of a transaction. Specifically, the value of a transaction is defined as Tr_{value} , and a transaction fee as Tr_{fee} .

Vr - Verification node. During formation of a block, the amount of verification nodes may vary, and is represented in any given time as cur_verif. This value is, therefore, a function of time, which can be computed as $cur\_verif(t)$, where t represents time. We may define: max_verif as $max\_verif=max(cur\_verif(t))$ and min_verif as $min\_verif=min(cur\_verif(t))$. We may also define access to a specific verification node within a network as Vr_i , where $i\in[1;cur\_verif]$, evidently limited by $cur\_verif\in[min\_verif;max\_verif]$.

Cr - Core node. At the time of a block's creation, the number of core nodes is fixed. This constant is denoted as cur_cores. Access to a specific core node within the network is thus denoted as Cr_i, where $i\in[1;cur\_cores]$.

Adr - An address within the system. Defined by a pair of private and public keys, and an account balance. Methods to access properties of an address can be declared, and in particular, access to the balance at an address is denoted by $Adr_{balance}$.

Synchronization Phase 1

Each verification node processes and calculates a multitude of transactions, that arrived at a verification node and are deemed valid. λ is the set of transactions that match the condition $\Theta(Tr)$, and $Adr_{balance} \geq Tr_{value} + Tr_{fee}$. Thus, for verification node Vr_i , a set of valid transactions is λ_i . As λ is a set of transactions, this set inherits the properties of a transaction, including hash transactions. A set of valid transaction hashes for a specific verification node is defined as $\lambda_{h,i}$, where the first index (h) denotes the object's properties, and the second (i) - the verification node index.

Synchronization Phase 2

At any given moment, the system is defined with multiple core nodes. In the second synchronization phase, a temporary attribute role $Active\ Core$ describes a core node that is active for the duration of the creation of a block. As the set of cores is well-ordered, each verification node at any point in time is aware of each active core. The current core identifier can be obtained as $cur\_core = cur\_block \bmod cur\_cores$. This formula is simplified by fixing the number of cores in the system, in order to reduce the complexity of calculating the index. At this stage, all sets λ_i will be transferred to Cr_{cur_core} , as follows from the role of $Active\ Core$. For simplicity, $Cr_{cur\_core} = ACr$ is a valid notation. The result of this operation is set η , which is formed from the union of all sets λ_i . Thus:

$$\eta = \bigcup_{i \in [1;cur\_verif]} \cdot \lambda_i$$

Synchronization Phase 3

After forming a raw_block (A block of valid transactions according to the $Active\ Core$, comprised of), the $Active\ Core$ sends it to a random number of verification nodes. This number is denoted as $cnxt\_verif$. Verification nodes synchronize between themselves such that each has a raw_block. Each verification node then forms the difference of sets $v = \lambda \setminus \eta$, so that each has $v_i = \lambda_i \setminus \eta$. In normal system operation, $v_i = \emptyset$, and thus:

$$\bigcup_{i \in [1;cur\_verif]} \cdot v_i = \emptyset$$

When this equation is false, each verification node, for which $v_i \neq \emptyset$ sends v_i to a system-wide predefined amount $cnxt\_cores$ of core nodes according to a list. In this manner, the next $Active\ Cores$ will have additional transactions:

$$\omega k = \bigcup_{i \in [1;cur\_verif]}$$

where k - indices of previous blocks, for which $v_i \neq \emptyset$.

Synchronization Phase 4

While v_i is sent to each core node, verification nodes simultaneously send the raw_block to the all other $non\text{-}Active$ core nodes. Once all core nodes have received a block, the set $\eta_{h,j}$ is formed, in a manner analogous to the set $\lambda_{h,i}$. This set contains the hash of the raw_block transactions, as received by each node. The transaction hashes are much more compact than the transactions themselves, thus reducing the resource cost of the synchronization operation. As at this stage only the hashes of the transactions are used, it follows that only the attribute η_h is required. Further, consensus can be reached if and only if the set of hashes on the $Active\ Core$ is equal and equivalent to the set of intersections of transaction hashes on all other cores:

$$\eta_h \equiv \bigcap_{j\in[1;cur\_cores]} \cdot \ \eta_{h,j}$$

Or, as in transaction notation:

$$\bigcup_{i\in[1;cur\_verif]} \cdot \ \boldsymbol{\lambda}_{h,i} \equiv \bigcap_{j\in[1;cur\_cores]} \cdot \left(\bigcup_{i\in[1;cur\_verif]} \cdot \ \boldsymbol{\lambda}_{h,i} \right)$$

Synchronization Phase 5

If the condition of the fourth synchronization phase is achieved, then the core nodes form the pos_block, which contains the signatures of all core nodes that have verified the block. This block is then sent down to the torrent nodes.

The system involves up to 64 core nodes. Block creation proceeds by order from one core to the next. In the event of failure of a core node, a slave core steps in as a replacement. A block is only validated if checked and confirmed by more than 67% of the other cores and verification nodes. Only after that will the next core in line commence formation of the next block, while the current validated block is sent to 3 torrent nodes, for dissemination to clients.

Torrent nodes once again confirm a block, and if it is found valid, initiate dissemination of the block to users and other torrent nodes. If, however, the block is found to be invalid, the torrent nodes raise a signal, alerting the system that core nodes have been compromised, initiating a role redistribution process.

Transaction flowchart and consensus algorithm visualization

Fig. 8 shows a flowchart describing the process of reaching consensus., *Tab. 5* provides descriptions and time intervals for each stage of the process. Mo = Móde designation is used, denoting the most frequent occurring observation in a data set.

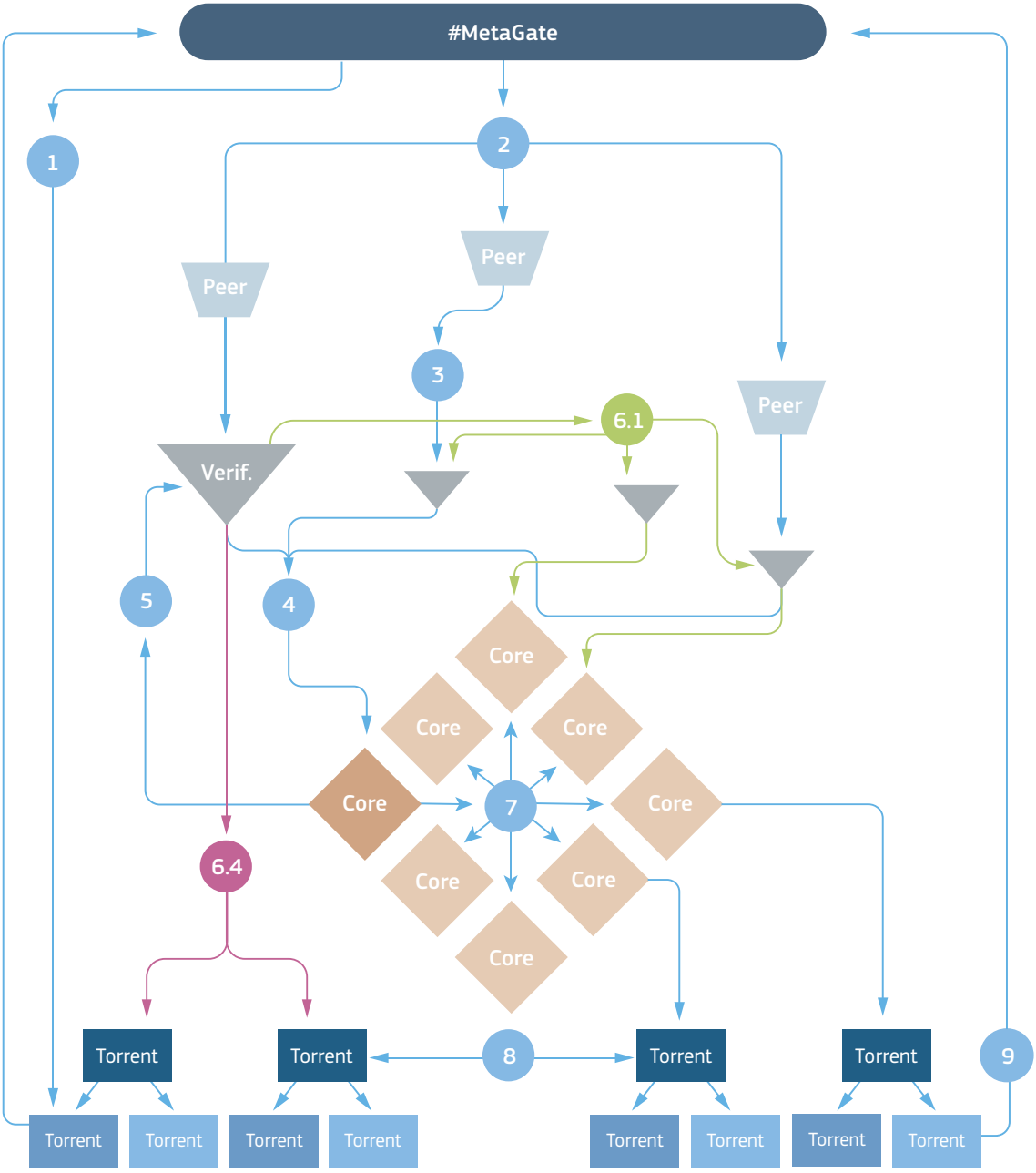


Fig. 8. Consensus reaching diagram

Tab. 5. Consensus reaching stages

#	Process	Execution time
1	#MetaGate requests service data from torrent nodes.	Mo=0.27s 0.05 - 1.5s
2	#MetaGate creates a transaction, signs it and sends it to Peer Nodes.	Mo=0.27s 0.05 - 1.5s
3	Peer node checks the sufficiency and accuracy of data, signature of the transaction and sends it to verification nodes.	Mo=0.2s 0.002 - 0.8s
4	Verification node checks data accuracy and transaction's economic feasibility. Possessing the information on which core node is currently active (which is currently creating a block), the verification node sends a transaction to the latter.	Mo=0.01s 0.002 - 0.8s
5	The core node generates a block from the incoming transactions and gives it to the nearest verification nodes.	Mo=0.47s 0.001 - 1s
6	**Processes occurring in parallel:**	
6.1	Verification nodes distribute block among the other verification nodes and the nearest Core nodes.	Mo=0.27s 0.002 - 0.8s
6.2	Verification nodes check the transactions in a block. Having found any lost transactions, verification nodes deliver these to the following Core nodes, participating in a block's creation, to include the transaction in the next block or to discard it completely.	
6.3	Discarded transactions are delivered to Torrent nodes so that the user could learn the reasons of discarding and the status.	
6.4	The block that was not yet verified by voting of all cores is seeded to the root layer of torrent nodes to speed up synchronization.	
7	Core nodes verify if the block is correct, sign it and deliver it to the root layer of torrent nodes via verification nodes. The information about the block delivered to the root layer of torrent node is already verified by the core nodes. Every torrent node on a way to a client may check the core nodes voting result and add his own vote.	Mo=0.27s 0.002 - 0.8s

#	Process	Execution time
8	The block is synchronized with the torrent node tree from the root layer of torrent nodes in line with the network map. In the event a torrent node does not validate that the block is correct, it votes for the network rebuilding. If 67% + 1 vote of torrent nodes are in favor of network rebuilding, torrent nodes stop delivering new blocks until network rebuilding is complete.	Mo=0.47s 0.002 - 1.3s
9	#MetaGate requests the status of transaction from torrent nodes. In case torrent node believes that the block is incorrect, the former informs the client thereof. Hence, in the event one of the torrent nodes has replied to a client that the current block might be compromised, transaction cannot be deemed as irreversible and the probability of network's reconfiguration is high. However, if all torrent nodes inform the client that the network functions normally and there are no compromised core nodes, the probability of transaction rollback tends to zero.	

According to test results described in Testing section, transaction verification and block distribution by torrent nodes, starting from the transaction arrival to a peer node takes 1.69 sec on average, and 5.5 sec as a maximum. Thus, it takes only 1.69 sec to generate an irreversible block across global network of 244 nodes.

Trust

Each node obtains a Trust factor between 0.01 and 1. Trust is increased by 0.005 units every 24 hours during reward calculation when all transactions are validated correctly, and decreased by 0.05 in case of network degradation or by 0.5 in case of incorrect validation of a transaction immediately after detection. Achieving a Trust value of 1 requires 198 days.

Voting

The system employs a vote-based decision mechanism, this includes validation of blocks, changes to network topology, or software updates.

Any address can delegate its coins to another address. The delegation process and its constraints are described in detail in the Forging section. The total amount of coins, own and delegated, held by an address, is called the Stake, and denotes the absolute value of the voting power of that address.

$$Stake = Holder\ Coins + Delegated\ Coins$$

For voting weighted votes are used - superposition of node's Stake and Trust.

$$Effective\ Stake = Stake \cdot Trust$$

Cryptography

For network security the digital signature algorithms of ECDSA (Elliptic Curve Digital Signature Algorithm) family are used. SHA-256[6] is used as a main hash function. These algorithms are considered attack-resistant[7] and have repeatedly proven their applicability in the sphere of digital security. The results of the testing undertaken by COMPUTER SECURITY RESOURCE CENTER are presented in Cryptographic Algorithm Validation Program report[8].

As quantum computers advance, #MetaHash considers using post-quantum cryptography algorithms[9].

Block Structure

To achieve the optimal volume of data storage, archiving and blockchain multi-level high-speed validation, #MetaHash uses 3 types of blocks:

Genesis block is a primary block, containing information on the initial release of coins and their distribution[10].

State block is a snapshot of a blockchain state. A State block is generated once per billion of transactions (250 GB) and requires a vote by all verification nodes. State blocks make it possible for verification nodes not to store older blocks, freeing space for new data, leaving old data in the data storage archive.

Micro block is generated every second based on Genesis or State blocks. Micro blocks are validated by a number of nodes that are available to run validation during the cycle, and checked by all validators upon their generation.

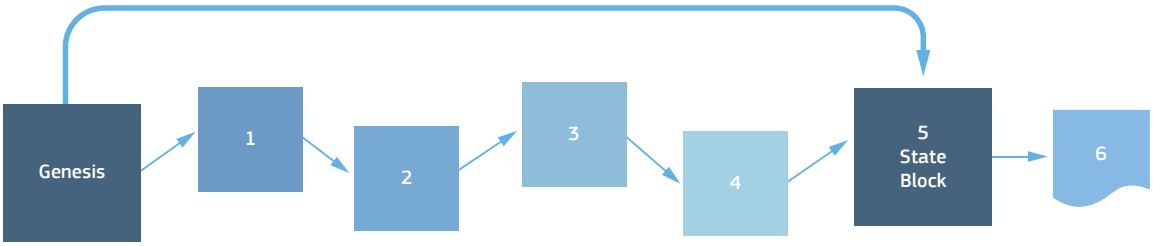


Fig. 9.

6. TBA

7. https://csrc.nist.gov/projects/hash-functions

8. https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Secure-Hashing

9. https://en.wikipedia.org/wiki/Post-quantum_cryptography

10. https://metahash.org/docs/MetaHash_WhitePaper_EN.pdf

State blocks as an alternative to hard fork

Since a State block contains a complete network snapshot, a hard fork is not necessary in cases of software updates. After network converts to new software, operations continue according to the same rules, until the next State block is sent. Activation of the new State block also activates the changes in the rules as required by the new software, allowing a simultaneous and smooth update process.

Blocks discarding procedure

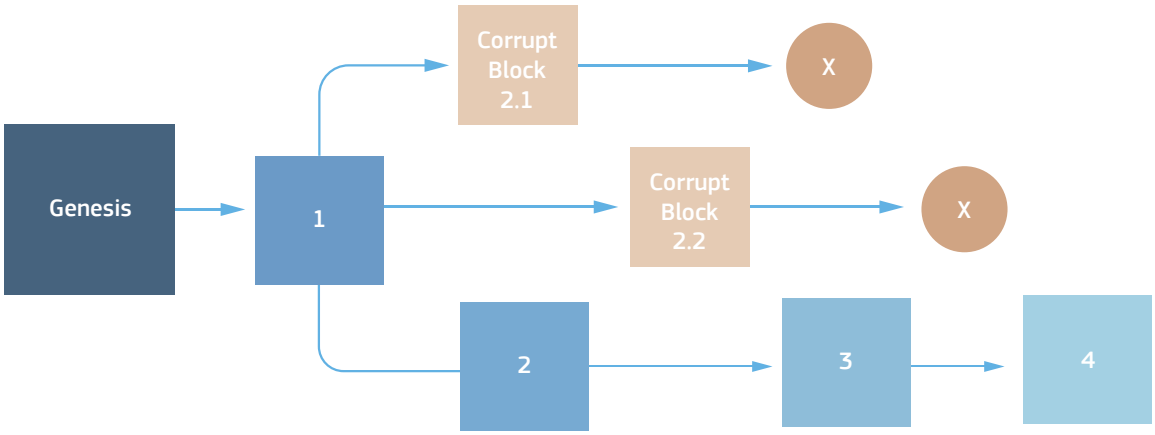


Fig. 10. Block discarding process

In #TraceChain all validators vote simultaneously, rather than one by one. Therefore, if the core nodes create an invalid block and if the following validators reject it for some reason, the corruption check takes place and, where required, the network is rebuilt until consensus can be reached. At the same time, every new transaction is buffered at the verification node layer, so that processing of the buffer starts when the next valid block becomes available.

Network split and merge

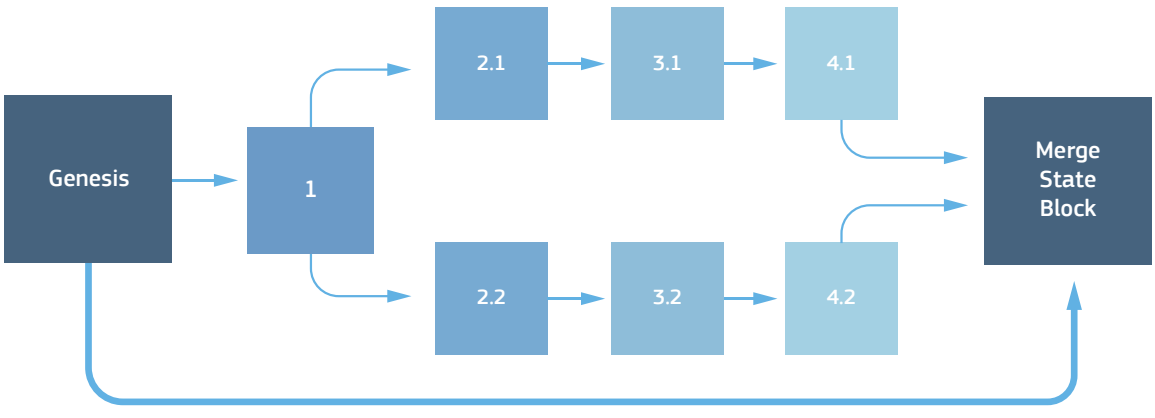


Fig. 11. Network split and merge

Under extreme conditions, for example, during cross-continental connection failure or loss of connection with any particular country, a network may split into two or more separate networks. In this case each network will retain the original structure, but with the fewer nodes. After the merge and synchronization of lost blocks, the incoming transactions will be accumulated in a pool, and an unscheduled Merge State block creation will commence. All Micro blocks will be saved and merged into one State block, then the network will rebuild itself, continuing operations from the new State block.

To eliminate double spend motivation, those transactions which involve one address and exceed balance will be deemed as valid for the network with the maximum votes. Recipients in the network with fewer votes will be informed that the network has been split. In theory, the recipient may initiate double spend in any other network, and the transaction may be cancelled. Thus, the recipient will have to wait until the networks merge, while standard transactions will be processed as normal. Though such a system has its flaws, it allows rolling back transactions which have been made on purpose to achieve double spend (as opposed to the scheme, where the longest blockchain wins and all transactions in it are subject to a rollback).

Balance Model

The basis of any financial system is the asset management method. It is impossible to work with assets without a clear understanding of available balance. Double-entry bookkeeping, as applied in classic accounting, or triple-entry bookkeeping, favored in decentralized systems accounting, would not be applicable to blockchain platforms, if platforms did not allow tracking the user balance.

Currently, two popular balance models are: Unspent Transaction Output (UTXO) and the Account Model. The first model is a directed graph of assets moving between users, the second is a database with the current network state.

Both models have their advantages and disadvantages, as shown in *Tab. 6*. For more details about issues with the models, please refer to the article prepared by HashEx[11] consultancy.

Tab. 6. Pros and cons of UTXO and Account balance models

UTXO model	**Account model**
Higher degree of privacy for new addresses, the coin does not have an owner	Need to store all accounts state
More easily scaled through sharding	More efficient usage of a repository
Hard to work with smart contract states	Intuitively clear approach
Allows using multi-threading for computations	Light clients can analyse states more easily
Complete transparency of assets movement	High degree of fungibility; Harder to track assets
	Every transaction must have a nonce
	To prevent the state machine from storing nonce, unused addresses are deleted
	Inconvenient tracking of internal transactions in a public blockchain

11. TBA

A key advantage of the first model is the uniqueness of every coin, while the second model provides a simpler smart contracts implementation. As both aspects are important for a modern system, it is not reasonable to choose one model above the other. This is confirmed by solutions applied for popular products. *Tab. 7* presents the models used by the most popular blockchains.

Tab. 7. Popular blockchain model comparison

Blockchain	Model	Consensus	Smart contracts
Bitcoin	UTXO	PoW	-
Ethereum	Account	PoW/PoS	+
EOS	Account	DPoS	+
Tron	UTXO+Account	DPoS	+
Qtum	UTXO+AAL	PoS	+
ICON	Account+?[12]	LFT (BFT)	+

It should be noted, that nearly all new systems have a hybrid balance model, while Ethereum and EOS use a pure account model.

At the initial stage it was decided to use account model to speed up implementation. We have also resolved an issue of data growth associated with nonce accounts storage by starting a new nonce fork after each State block. Thus, the information of unused or extremely old accounts is automatically erased from node memory.

Later a hybrid scheme will be considered: UTXO (for balances) and Accounts (for contracts). Since #TraceChain features a lot of transactions, a standard Merkle tree in UTXO model may take too much time and considerable disc space. Therefore, an AVL+ tree[13] will be introduced, to optimize the proof model and reduce resources required for data storage.

To put that in context, *Fig. 12*, *13* and *14* present the graphs of dependency of proof size (in bytes) on tree size for three types of trees (Descartes' tree (Treap), skip list and AVL), of proof size per modification on tree size (Ethereum and AVL+) and of block processing time on blocks total.

12. No public information available on additional model.

13. Improving Authenticated Dynamic Dictionaries, with Applications to Cryptocurrencies https://eprint.iacr.org/2016/994.pdf

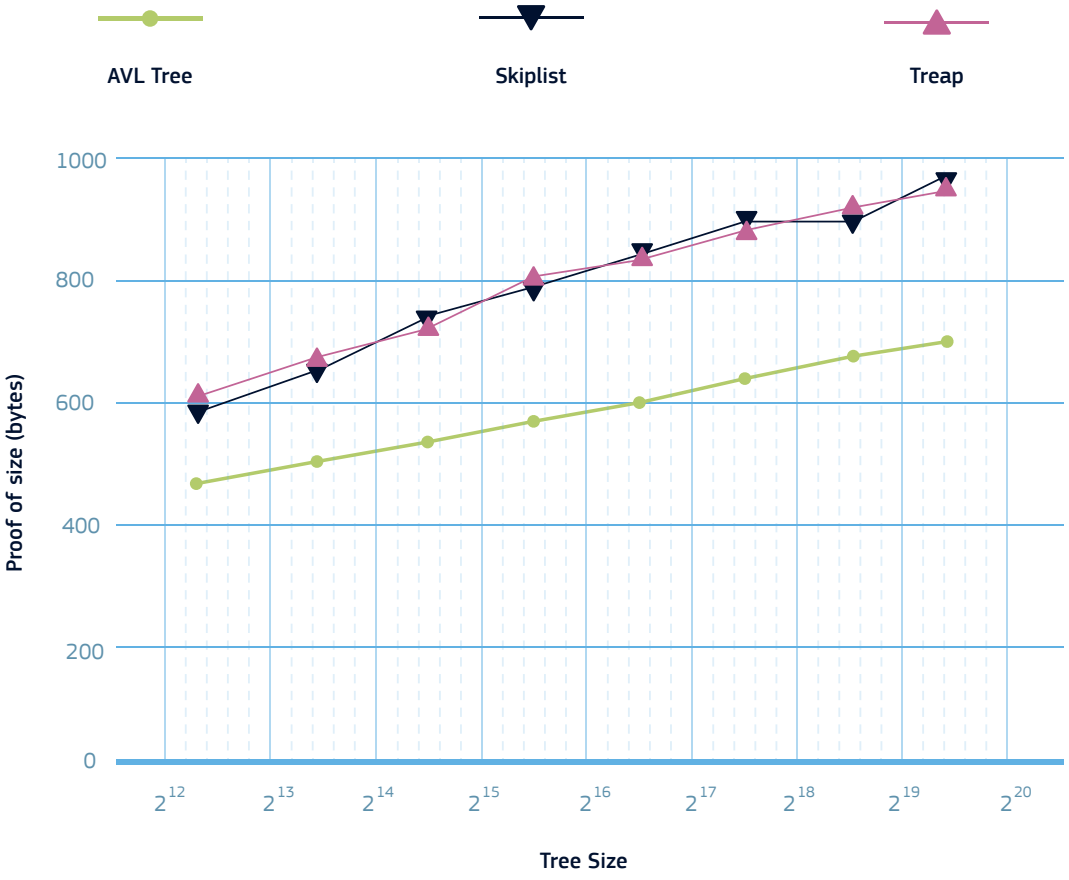


Fig. 12. Proof size on tree size dependency

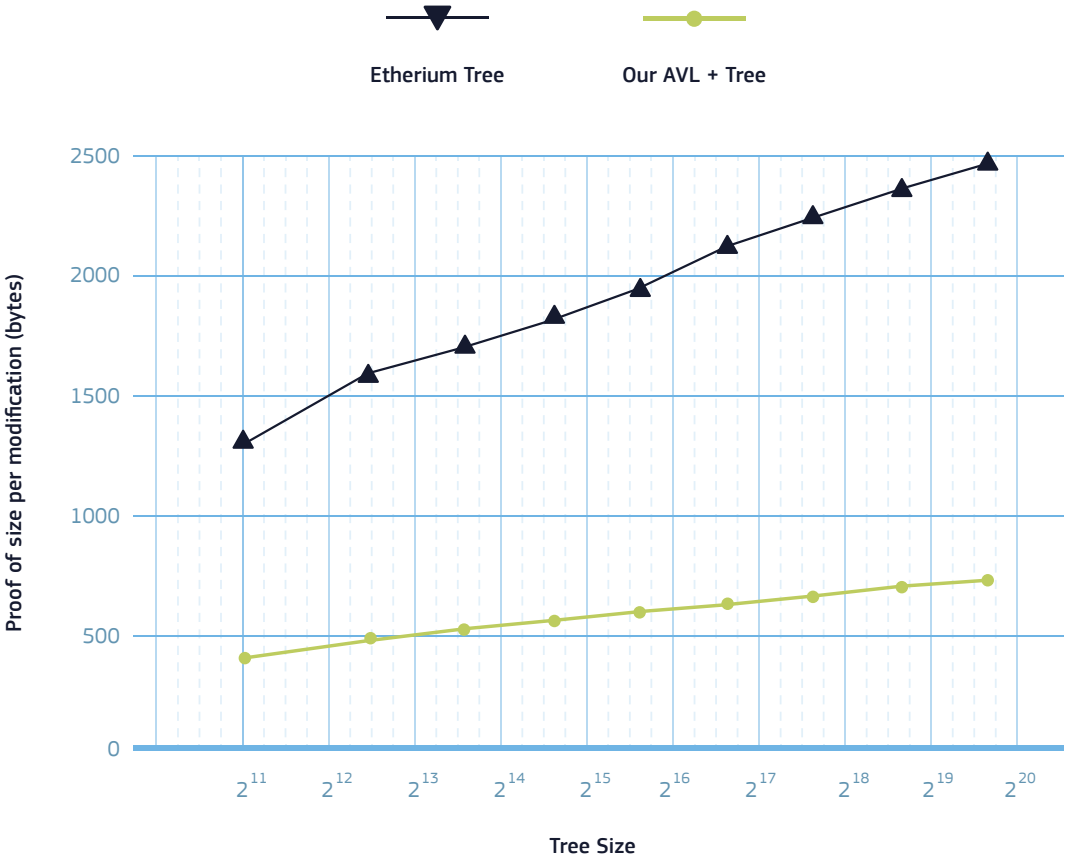


Fig. 13. Proof size per modification on tree size dependency

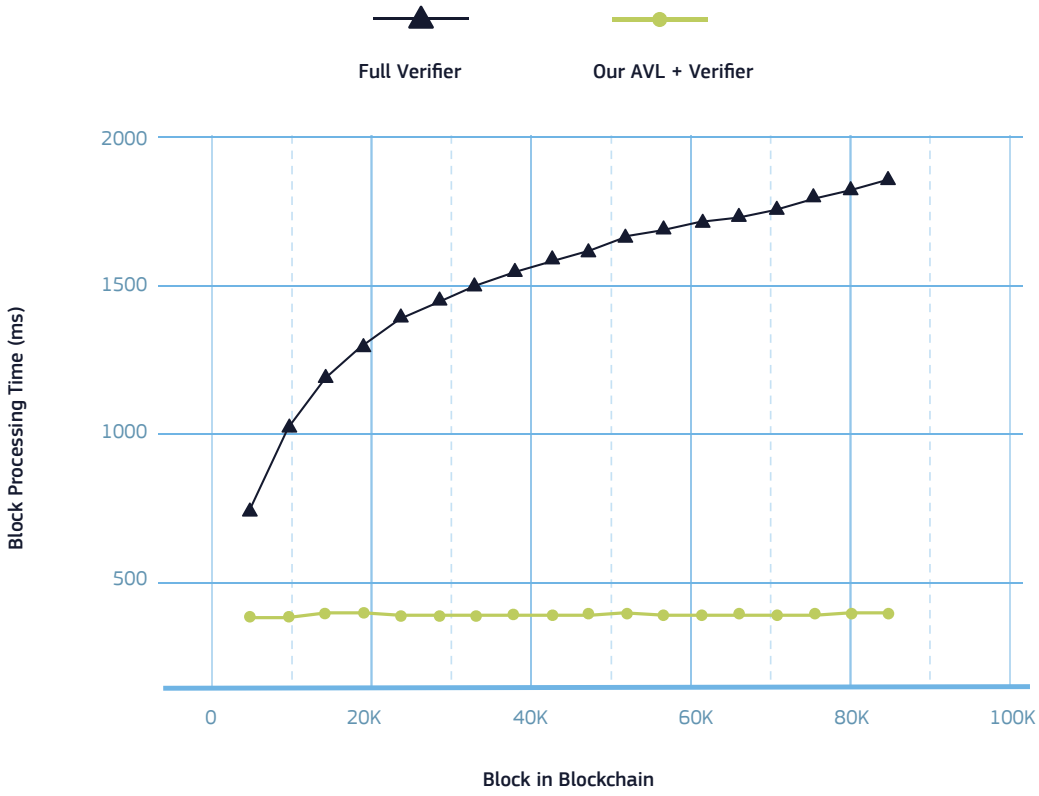


Fig. 14. Block processing time on blocks in blockchain dependency

Transactions

Transaction is a structure that describes a change in the blockchain state. For example, a transaction can describe the following: “perform a cryptocurrency transfer operation between two addresses”. Transactions are transmitted in binary form and confirmed with signature and the public key[14]. Transactions in the system are presented in the form of a POST request. *Fig. 15* provides a graphic representation of a transaction, while *Tab. 8* provides a data structure inside the transaction.

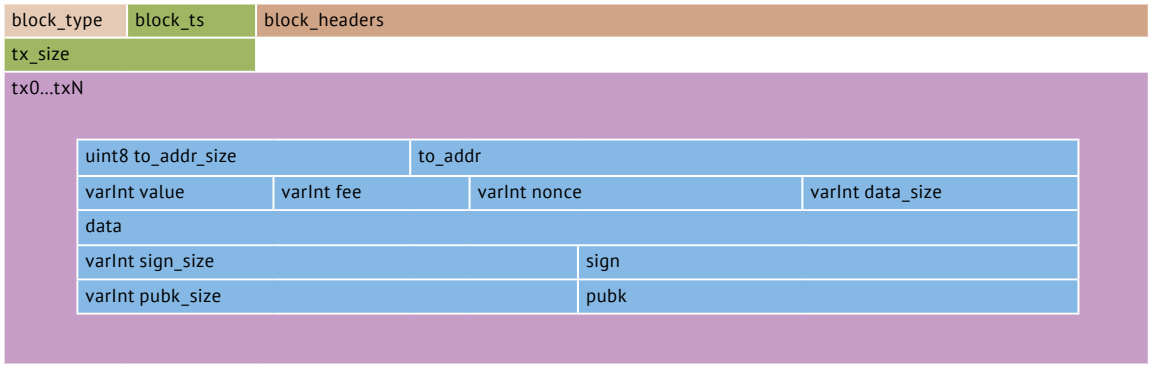


Fig. 15. Graphic representation of transaction

Tab. 8. Transaction structure

Field	Description
to	recipient address
value	amount of assets
fee	amount of fee
none	blockchain secure number
data	blockchain records
pubkey	public key
sign	signature

14. TBA

Protocol Features

#TraceChain is a binary protocol supporting financial transactions, data transactions and technical transactions. **Technical transactions** provide an opportunity for vote delegation, cooperative delegation for combination into forging pools, voting and validation, control of child addresses and special-use addresses. **Data transactions** allow data storage and execute such applications as messenger, gambling and other. **Financial transactions** allow transfer of money between the addresses. All types of transactions are stored in different chains in order to optimize the size of the stored data and disallow excessive amplification of financial chain due to the technical data.

Special-use addresses

1. Main address. Used in most cases. No advanced features
2. Smart contract address. Assigned exclusively to smart contracts
3. Developer address. Used for smart contract administration
4. Hypervisor address. Used for virtualization system
5. Frozen address
6. Child address with withdrawal to parent or frozen address only
7. Child address with withdrawal to frozen address only
8. Multisig address

Address generation from the public key can be done in several steps:

1. Take a part of a 65-byte Public Key where first byte is 0x04, the next 32 bytes correspond to X coordinate, and the last 32 bytes - to Y coordinate,
2. Perform SHA-256 hashing on the public key,
3. Perform RIPEMD-160 hashing on the result of the previous step,
4. Add a marker byte in front of the result based on the address type:
    - 0x00 for main address,
    - 0x08 for smart contract address,
    - 0x09 for developer address,
    - 0x10 for hypervisor address,
    - 0x01 for frozen address,
    - 0x02..0x06 for child address with withdrawal to parent or frozen address only
    - 0x07 for child address with withdrawal to frozen address only
    - 0x11..0x15 for multisig address,

5. Perform SHA-256 hash on the result,
6. After another SHA-256 hash on the last result take only the first 4 bytes of the resulting hash,
7. Add these 4 bytes to the end of RIPEMD-160 hash from item 3.

Multisignature address

Multisignatures (multisigs) are addresses requiring multiple users' signatures to perform a transaction. Two different approaches are used in #MetaHash, one for situations requiring anonymity and the other for non-anonymous situations.

Anonymous multisig

When an anonymous approach is required, the private key of the multisig is split into parts and distributed between owners via Fiat-Shamir Secret Sharing method[15].

Non-anonymous multisig

In this approach multisig is a smart contract and requires that transaction is signed with several private keys.

Frozen address

Freeze is performed by means of a special transaction which the address sends back to itself. Together with the transaction a timestamp value is sent, until the end of which any transfer operations from this address are disallowed. Incoming transfers are allowed, however. The freeze time can be changed with a similar transaction, but only for extending it, not shortening. Transactions with later timestamp events are prioritized over others. Such addresses are used as security that the coins will remain unspent for a certain timeframe.

Child address

Special-use child address allows transactions only to a frozen address with a common parent or to a frozen parent address. Such addresses are needed for online wallets requiring offline wallet signatures for operations.

15. https://en.wikipedia.org/wiki/Feige%E2%80%93Fiat%E2%80%93Shamir_identification_scheme

Forging

Forging is a process of block generation and validation which supports the operation of entire #MetaHash network. Every node participates in the forging process. It is a high-performance alternative to mining (used in PoW networks).

Forging is rewarded. The reward for forging is granted from the forging pool and from collecting fees for transactions. Total size of forging pool is 4,600,000,000 #MetaHashCoins (#MHC) unevenly distributed within the first 10 years of network operation[16]. Rewards are paid every 24 hours.

The periodicity of recalculation and distribution may vary based on technical needs of the network. Unlike in PoW, nodes for #MHC forging never get obsolete and their efficiency does not deteriorate, as their number is limited to the minimum number of coins required for a specific role. The faster the network nodes are, the higher capacity the network will have.

As all nodes are essential to network operation, the rewards are distributed between all members, not only between Core and Torrent nodes creating and validating new blocks.

#MHC Delegation

Address holders unwilling to upkeep their own node may delegate their coins to their trusted nodes and still receive forging rewards. During delegation, a technical transaction occurs. Coins will be frozen for the delegation period and cannot be transmitted to another address or delegated to another node until the delegation is canceled.

Delegation is canceled once every 24 hours during the next role reassignment operation. Coins can be delegated once again only after cancellation of the delegation.

If coins were delegated to a compromised node that has been detected as performing malicious activity, the delegation is canceled automatically, and all coins are frozen for 10 days to exclude the possibility of any transfers or redelegation. Coins remain in the ownership of their holder, but are unavailable for the next 10 days.

16. https://static.metahash.org/docs/MetaHash_WhitePaper_EN.pdf, chapter Financial Model

Reward Distribution

Tab.9. Reward distribution among the members of the network

Reward Share	Reward Type
50%	Coin Holder Reward. Reward to the holders of #MHC
40%	Node Reward
10%	#MetaGate Reward. Reward of 1000 active #MetaGate holders

Coin Holder Reward

Coin Holder Reward is calculated according to the following formula:

$$Coin\ Holder\ Reward = 0.5 \cdot \frac{Holder\ Coins\ At\ Stake}{All\ Coins\ At\ Stake} \cdot (ForgingPool + Commissions)$$

$Holder\ Coins\ At\ Stake$ - Amount of coins delegated by the address holder

$All\ Coins\ At\ Stake$ - Amount of coins delegated by all the network addresses

$ForgingPool$ - Forging stimulation pool issued once every 24 hours for 10 years

$Commissions$ - All transactions or storage of data commissions received by the network during the past 24 hours

Node Reward

The financial stimulation system is aimed at adding nodes with the maximum throughput capacity, uniformly among all roles and latency zones (normally these are geographical regions). To maintain maximum performance of the system, it is beneficial to have a limited number of high-performance broad bandwidth computers over the large number of computers per se, while lower performance level computers may act as peer nodes and protect the network core. Decentralization is achieved through an abundance of nodes in the pool that are able to assume any role, and automatic decentralized change of node roles. The more saturated the network is with high-capacity nodes, the higher amount of transactions the network is capable of processing per unit of time.

Node rewards are calculated based on the number of coins which are delegated to them, taking into account their value for the network. The main advantage of the fastest nodes is that they can gather more delegate votes and, therefore, increase the share of the received commissions, while lower performing nodes have limitations on the amount of delegated coins.

Node reward calculation

Role value Crole

$$Crole \cdot \frac{64}{real\ number\ of\ nodes\ suitable\ for\ role}$$

The network requires at least 64 computers matching each role.
During the node assessment the largest value is chosen out of all possible roles.
Only $Crole \geq 1$ is taken into account during the calculation.

$Crole$ stimulates adding such nodes and roles to the network, that are in deficit for the balanced operation of the network.

Performance coefficient Cperf

To calculate $Cperf$ the nodes are rated by their speed from 1 to 100% where 0-20% are the slowest nodes, and 61-100% are the fastest according to the tests. Coefficient calculation rules are presented in *Tab. 10*

*Tab. 10. **Cperf** calculation*

Coefficient	Speed rating
1/3	0-20%
2/3	21-40%
1	41-60%
4/3	61-80%
5/3	81-100%

$Cperf$ stimulates adding nodes with the fastest channels and CPU for signature verification.

Geolocation coefficient Cgeo

$$Cgeo = \frac{nodes\ in\ latency\ cluster \cdot latency\ clusters\ amount}{all\ nodes\ count\ in\ all\ latency\ clusters}$$

$Cgeo$ stimulates adding nodes to the regions with fewer nodes (normally regions with costly operation).

Effective coefficient C

$$C = Crole \cdot Cperf \cdot Cgeo$$

$$effective\ delegated\ coins\ for\ Node = C \cdot Trust \cdot delegated\ \#MHC$$

$$Node\ reward = 0.4 \cdot \frac{effective\ delegated\ coins\ for\ Node}{all\ effective\ delegated\ coins\ for\ all\ Nodes} \cdot (Forging\ Pool + Commissions)$$

Stakes by roles

$$Effective\ delegated\ coins\ (ed\#MHC) = C\ я \cdot \#MetaHash\ Coins\ at\ stake$$

With few nodes in the network, C multiplies the effect of the delegated coins and allows the node to take a role owning much less delegated #MHC than the required value. With overabundance of nodes in a certain role and region, the required #MHC stake may be higher than specified.

The required stake of $ed\#MHC$ and reward limits for each role are specified in *Tab. 11*:

Tab. 11. Required ed#MHC stake and reward limits for each role

Role	ed#MHC stake	#MHC reward limit
Core node	1.000.000	none
Verification node Torrent node Peer node	100.000	999.999
#MetaGate	100	99.999

Upon activation the node undergoes its primary assessment for the role it may fit according to its technical characteristics. When 1,000,000 #MHC is reached with the adequate technical properties, the node can be assigned to any role in the network, while receiving rewards for the highest role. And vice versa, if the node is underperforming compared to the fastest network nodes, it can get the rewards only within the limits for other roles.

#MetaGate Reward

#MetaGate reward is distributed among 1,000 random active members. If the number of active #MetaGate nodes is below 1,000, reward for the empty slots will be allocated to the next scoring member. #MetaGate is considered active if the client has been online at least 4 hours within the past 24 hours.

5% — top reward

1% — second

0.5% — third

0.4% — fourth

0.35% — fifth

0.95% — from 6 to 100 per 0.01%

1.8% — from 101 to 1.000 per 0.002%

Recommended hardware

Tab. 12 shows hardware characteristics allowing to maintain the claimed network performance, as well as their estimated cost of purchase or lease. With sufficient number of nodes with characteristics superior to the recommended ones, the network will automatically reassign the underperforming nodes to roles such as Verification, Peer or Torrent.

Tab. 12. Recommended hardware and its cost.

Role	Sample HW config/example VM specification	Purchase	Lease
Core	2 x Intel® Xeon Gold 5120 CPU, 512GB RAM, 4TB HDD	$16.000	$600
Torrent	Intel® Xeon® Silver 4110, 32GB RAM, 4TB HDD	$2.000	$200
Verification	Intel® Xeon® E3-1240, 8GB RAM, 1TB HHD	$1.000	$180
Peer	Intel Core i3 7100, 4G RAM, 500GB HDD Digital Ocean droplet / Linode VM	$850 -	$170 $50

Decentralization Strategy

As of now, all network nodes are controlled by the #MetaHash team, therefore the system is deemed centralized. Decentralization of the new technology should go through a number of public tests, troubleshooting and real-time changes. The #MetaHash team plans to support the network until it is fully decentralized and capable of self-development.

Phased decentralization will allow testing of each component one at a time with virtually no risk for the network. There are 3 steps proposed for each component:

1. Testing in developers' network
2. Public testing of updates via testnet
3. Update moved to mainnet

Decentralisation steps include:

1. Developers' network launch. Load tests and verification of processing correctness with 500 billion transactions.
2. Public testnet launch.
3. Public mainnet launch.
4. Anchoring to popular blockchains.
   The mainnet being already launched, multiple backups and archiving of data are required for the increased security of the members. In order to avoid changes when a critical vulnerability is detected and as a mitigation of the misuse of authority by the team, it is planned to use anchoring of #TraceChain checksum hashes to public blockchains. Each block's hash containing calculation of the forging reward will be recorded to Ethereum and Bitcoin blockchains. Therefore, each member will be able to check the validity of the entire chain having static control points in two public blockchains.
5. Launch of the peer node forging. All incoming transactions will be processed by the network of public decentralized nodes which allows to make sure no transactions appear in the blockchain by any other means. Also, at this stage the mechanism of forging rewarding of the first available role will be tested.
6. Torrent node launch. At this stage the network reaches the goal of being a public decentralized storage of all transactions and blocks in a public repository and a torrent verification of the cores' performance correctness. Clients start receiving data from decentralized nodes.
7. Switching on the automated role assignment by #TraceChain AI for the #MetaHash team nodes. User nodes remain with the static roles for now;
8. Switching on the automated role assignment by #TraceChain AI for user nodes for Peer and Torrent roles. This stage verifies the correctness of network map design through machine learning and automated network rebuilding.
9. Start security testing with selected vendors against attacks on consensus and network failures.

10. Software update and start of bounty campaign focused on debugging and security issues. At this stage main architectural vulnerabilities to network attacks are already eliminated and the public bounty helps engage the public to eliminate rare types of attacks.
11. Full decentralization of cores, slave cores and verification nodes.
12. Software update ensuring a completely autonomous distribution of roles by the network.

Phased decentralization helps provide a continuous operation of the network as it is being implemented and risk control in terms of various attacks on the network.

Risks

Risk assessment was done by HashEx[17] based on FERMA[18] standard. Only risks with negative outcomes for the project have been studied. Three-dimensional assessment method has been used for the risk description. Risk assessment was based on risk probability, consequences (threats) and the organization's exposure.

1. Node availability, DDoS attack on nodes

Risk type: Operational.

Probability: Medium (Possible). Decentralized networks continuously experience attacks of such type. The attack may not last too long as it is quite costly for the attacker (estimated to $10,000 per day for a static-role system).

Consequences (Threats): Low.

Risk Control/Mitigation: Network automatically starts rebuilding if any node becomes unavailable.

2. Network bandwidth, DDoS attack via multiple transaction generations

Risk type: Operational.

Probability: Low (Remote). Unlike the DDoS type in previous risk, this attack requires much larger computational resources from the attacker, hence makes it an even more costly operation in comparison to the previous risk.

Consequences (Threats): Low.

Risk Control/Mitigation: During high network load transaction fees start growing nonlinearly, thus, attack quickly becomes impractical. With trying to maintain the 20% network load such attack would cost $2,000 each second. With 80% load one transaction would cost $3.91 at ICO price and the attack would cost $200,000 per second. Thus, such attack can increase transaction price in the network only for a short span of time.

3. Attack through the change of system local time

Risk type: Operational.

Probability: Medium (Possible). Attacker may modify the system local time to disrupt the node synchronization.

Consequences (Threats): Low.

Risk Control/Mitigation: All key time parameters are measured in ticks by the nodes without any reference to the server local time.

17. https://hashex.org/

18. https://www.theirm.org/knowledge-and-resources/risk-management-standards/irms-risk-management-standard/

4. Unlimited growth of blockchain size

Risk type: Operational.

Probability: High (Probable). With the course of time the blockchain size may lead to the impossibility to store the blockchain on a regular hardware equipment. At the moment of this writing, the size of Ethereum exceeds 600 GB, while the size of Bitcoin is close to 200 GB. Given the industry growth rate, in several years Ethereum size may exceed 10TB resulting in utter impossibility to store it on personal computers.

Consequences (Threats): High.

Risk Control/Mitigation: One of #MetaHash features is storage of archived data in a distributed repository with no negative side-effects on main network performance at the expense of State blocks. According to #MetaHash calculations [*See Appendix 1*], with continuous peak load on the network the storage of archived data will be financially efficient at least for 10 years. For this scenario it is proposed to implement a cheap archive repository or deploy a purging mechanism of the 10+ year old data. This decision should be taken by voting. Should #MetaHash ever face any challenges related to storage of the historical data, it will be possible to perform a fork approved through general voting and to create a new Genesis block in order to reduce the costs related to historical data storage.

5. Stake majority in single possession. 51% attack

Risk type: Strategic.

Probability: Low (Remote).

Consequences (Threats): High.

Risk Control/Mitigation: In #MetaPoS consensus subsystem, solving the BFT problem of getting the 67%+1 Stake simultaneously on several layers requires having control of over 90% of Stake. Also, such attack is extremely hard to synchronize since as soon as the network node signals that it's compromised, it loses its Trust value and can only be supported by other compromised nodes. Even with the unlikely emergence of the network with 90% of single-handedly owned Stake, #MetaGate clients will start retroactively reject the blocks and stop trusting the compromised torrents, effectively splitting the network in twos. Since clients using the network are its most valuable part, possessing 90% of control over the Stake only gives control over the 'dead' network with no clients, while the remaining 10% will become the 100% of coin holders in the new network.

6. Developers authority misuse during the system launch

Risk type: Strategic.

Probability: High (Probable). At the initial stage numerous critical nodes are under control of developers. The network does not have the required level of decentralization yet and, thus, is vulnerable to the malevolent changes.

Consequences (Threats): High.

Risk Control/Mitigation: Anchoring of #MetaHash to two popular blockchains: Bitcoin and Ethereum on a regular basis. Launch of forging on peer nodes and torrent nodes to ensure that all chain logs are stored in the public decentralized repository so that any illegitimate alteration is easily noticeable.

The risks described above are depicted on the risk map (*Fig. 16*).

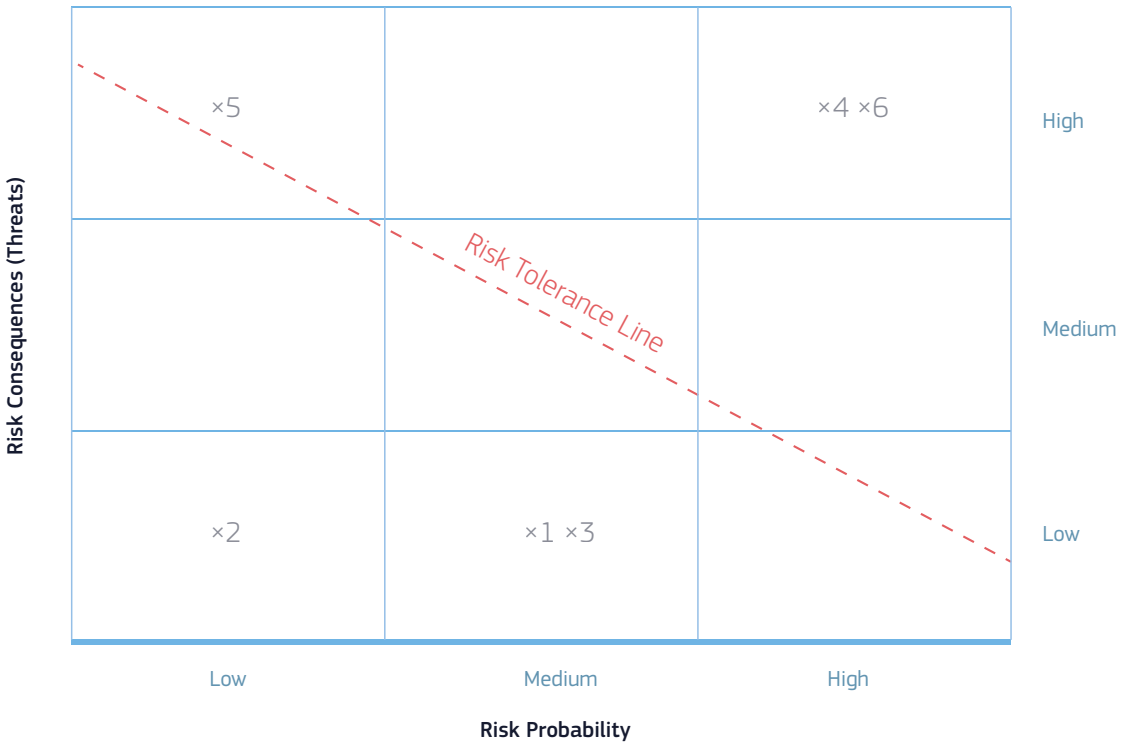


Fig. 16. Risk map

Strategic risks above the tolerance line require a more detailed study as they may result in ineluctable catastrophic consequences for the project as a whole.

Conclusion

From the project start in early 2017, the #MetaHash team have conducted a thorough research of IT technologies and studied possible solutions of the project goals.

The following results were achieved:

1. Optimal synchronization structure of the geographically distributed network designed with capacity of 18 Megabytes per second (>50,000 financial transactions) within 3 seconds. This includes validation of the sender signature and check for balance availability to make the transaction.
2. Decentralized storage of the archived data and state-block mechanism in place.
3. Various consensus implementations of new type, MultiPoS, were tested, where multiple verification nodes vote for the block correctness while generating the next one. This allows getting an irreversible transaction significantly faster and minimize a probable blockchain fork and rejection of the branches.
4. Optimal balance model with all the benefits of blockchain network structure was chosen, reducing the size of stored and transmitted data by 60% in comparison to average performance characteristics of other blockchain platforms, as well as adding the capability of nonlinear data validation inside the blockchain, resulting in a tenfold speed increase in blockchain validation.
5. Successfully resolved network merge after a split caused by access restriction for a group of members or a cross-continental blockage.
6. The seamless vote-based software update introduction mechanism was developed, allowing to avoid network hard forks.

A complete production-grade release version of the platform was prepared, allowing fast transactions at minimum cost by reducing infrastructure costs and increasing capacity. This enables colossal opportunities for blockchain technology in real-time applications, Internet of Things paradigms, micro-transactional applications, and others.

The #MetaHash Team
www.metahash.org

#metahash

Appendix 1. Data storage

The more transactions performed in the network, the more data must be stored. Any project focused on a broad bandwidth faces the challenge of data storage and financial expenses related to such storage.

In #MetaHash, with 50,000 transactions per second, the size of the data compressed in binary form is shown in *Tab. 13*.

Tab. 13. Data size at 50,000 transactions per second

per hour	per 24 hours	per month	per year
45 GB	1,080 GB	32,400 GB	385 TB

The most cost-efficient drives in 2018 are those with 4 terabyte capacity. The cost of 1 GB in these products is $0.025, or $25.6 for 1 terabyte. One needs 96 drives a year or 1920 drives for redundancy. Overhead costs to maintain such a storage system, including replacement of damaged drives will amount to $48,000 per year. Data storage at peak load without duplication will cost $9,878 per unit annually.

As the storage system is decentralized, in order to secure seed data for the network, significant storage capacity will be required. Allowing for redundancy, the costs are expected to rise 20 times more than the unit price to ensure an adequate number of nodes and backups.

The cost of data storage drives at peak load, providing for 20x redundancy is equal to $118,546 + $48,000 for overheads. In total, storage cost at peak load with more than 20x reserve is below $200,000 annually.

Transaction volume at estimated peak load is shown in *Tab.14*.

Tab. 14. Volume of transactions at the estimated peak load

per second	per minute	per hour	per 24 hours	per year
50,000	3,000,000	180,000,000	4,320,000,000	1,559,520,000,000

In total, $1 is more than enough to cover the data storage expenses for 7 million transactions per year, and 500,000 transactions with 10-year history records. The cost of storage for one transaction is $0.00014 for 10 years at the estimated average peak load.

If the network reaches high costs for historical data storage, it may be subjected to forking, confirmed by a general vote, creating a new Genesis block, to reduce data storage expenses.