RASMART

YELLOW PAPER

Ver 1.0

Table of contents:

Rasmart is a multifunctional blockchain platform focused on the real business segment. The main advantages are the number of transactions per second (so-called system speed), a versatile api, that makes it possible to create a set of tools (services) for interaction with the outside world, and the decentralized registry and decreased fees to 0.001%

Intraduction

DISCLAIMER

This RASMART Technical yellow paper v1 is for information purposes only.
RASMART does not guarantee the accuracy of or the conclusions reached in this yellow paper, and this yellow paper is provided “as is”.
RASMART does not make and expressly disclaims all representations and warranties, express, implied, statutory or otherwise, whatsoever, including, but not limited to:
(i) warranties of merchantability, fitness for a particular purpose, suitability, usage, title or noninfringement;
(ii) that the contents of this yellow paper are free from error; and (iii) that such contents will not infringe third-party rights. RASMART and its affiliates shall have no liability for damages of any kind arising out of the use, reference to, or reliance on this yellow paper or any of the content contained herein, even if advised of the possibility of such damages.

In no event will RASMART or its affiliates be liable to any person or entity for any damages, losses, liabilities, costs or expenses of any kind, whether direct or indirect, consequential, compensatory, incidental, actual, exemplary, punitive or special for the use of, reference to, or reliance on this yellow paper or any of the content contained herein, including, without limitation, any loss of business, revenues, profits, data, use, goodwill or other intangible losses.

System advantages

Nowadays blockchain technology is rapidly developing, which has led to a series of requirements that the blockchain-based Rasmart meets, even providing its own improvements and extensions.

1. Versatility of use

Rasmart is a versatile and adaptive api that may be integrated into external services. This API uses Apache Thrift technology, which supports many industrial programming languages utilized to develop blockchain-interacting apps easily, quickly and efficiently. Many industrial middleware-class systems may as well work with Thrift ‘out of the box’, making it possible to seamlessly integrate heterogenic systems into a single complex.

2. Speed of Transactions

Speed of transactions is defined by the number of confirmed transactions put in the blockchain per second.

3. Safety

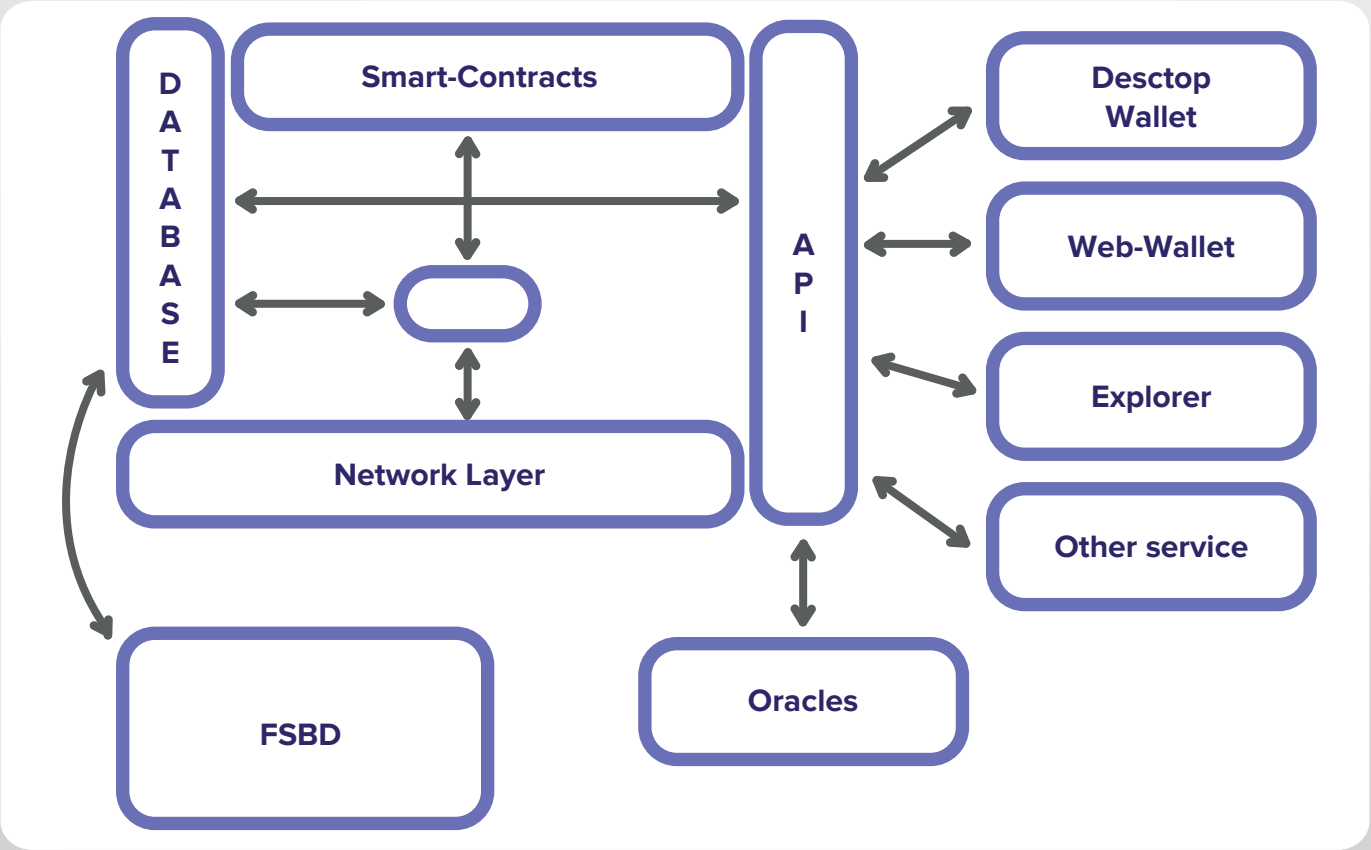
Nodes involved in block writing are changed every 0.2 seconds. The consensus materialized on the basis of Byzantine failure solution, gives us a chance to make right decisions even when there are compromised nodes. The entropic nature of trusted nodes selection provides the possibility of protection from both the so-called Sybil attack (when compromised nodes can ‘surrond’ the creative node and change it for the rest of the network) and the ‘51% attack’ (when the intruder gets the majority when a decision is made).

4. Smart contracts

Smart contracts of the Rasmart system are fast through the time-honored programming language ‘Lua’, whose advantages are quick execution and small interpreter (which does not require a huge computing power), totaling to a large transactional capacity and fast execution + a constantly updated sdk with various features.

Besides, the use of easy-to-learn Lua language will allow many programmers to write smart contracts, since it takes minimum time to master it.

Anatomy of Rasmart



Conceptually, Rasmart is made up of nodes (the main element of the platform) and a set of secondary modules created for end users (wallets, monitors, oracles) working under api represented by the nodes.

Node:

A node is a basic system module performing data transfers, data storage, smart contracts management and provision of api for external services
Each node has its own copy of the whole blockchain database, which makes the network absolutely fail-safe. If there is at least one node with an intact database, the whole network will be able to automatically restore from any destructive impact.

Api-based services

Wallet:

A wallet is an external app that communicated with the node through API and allows it to send and receive transactions as well as work with smart contracts.

Web-wallet:

A web wallet provides the same opportunities, though in this case it is a thin client, while the web-server connects to the node. Web-wallet users neither participate in the decision-making process nor receive any commission for that.

Explorer:

A monitor is also a thin client providing a summary on network performance. It shows the dynamics of new blocks creation, allows us to see transactions, smart contracts and the results of their execution, as well as cumulative information on the number of blocks, transactions and flow of funds per day, week or year...

Oracles:

To make a decision, smart contracts require information from the 'outside world'. Oracles are external apps which receive information from various sources through a secured channel and use the node API to put it into the blockchain so that smart contracts could access to the informatino and make decisions basing on that data.

Technical description of the project

Node

Depending on the context, a node is software or a computer with the Internet connection (a server or working station) running that software.

Network structure

A network consists of nodes able to exchange message via UDP protocol (layer 4 of the OSI model). Topologically a network is an overlay network (on top of Ethernet), where each node is given a unique node identifier, and the routing is via an ordered directed graph made up of node identifiers.

Routing and mapping the overlay network onto a real network infrastructure is fully performed by the transport layer of the node software.

Identification and addressing of nodes in the network

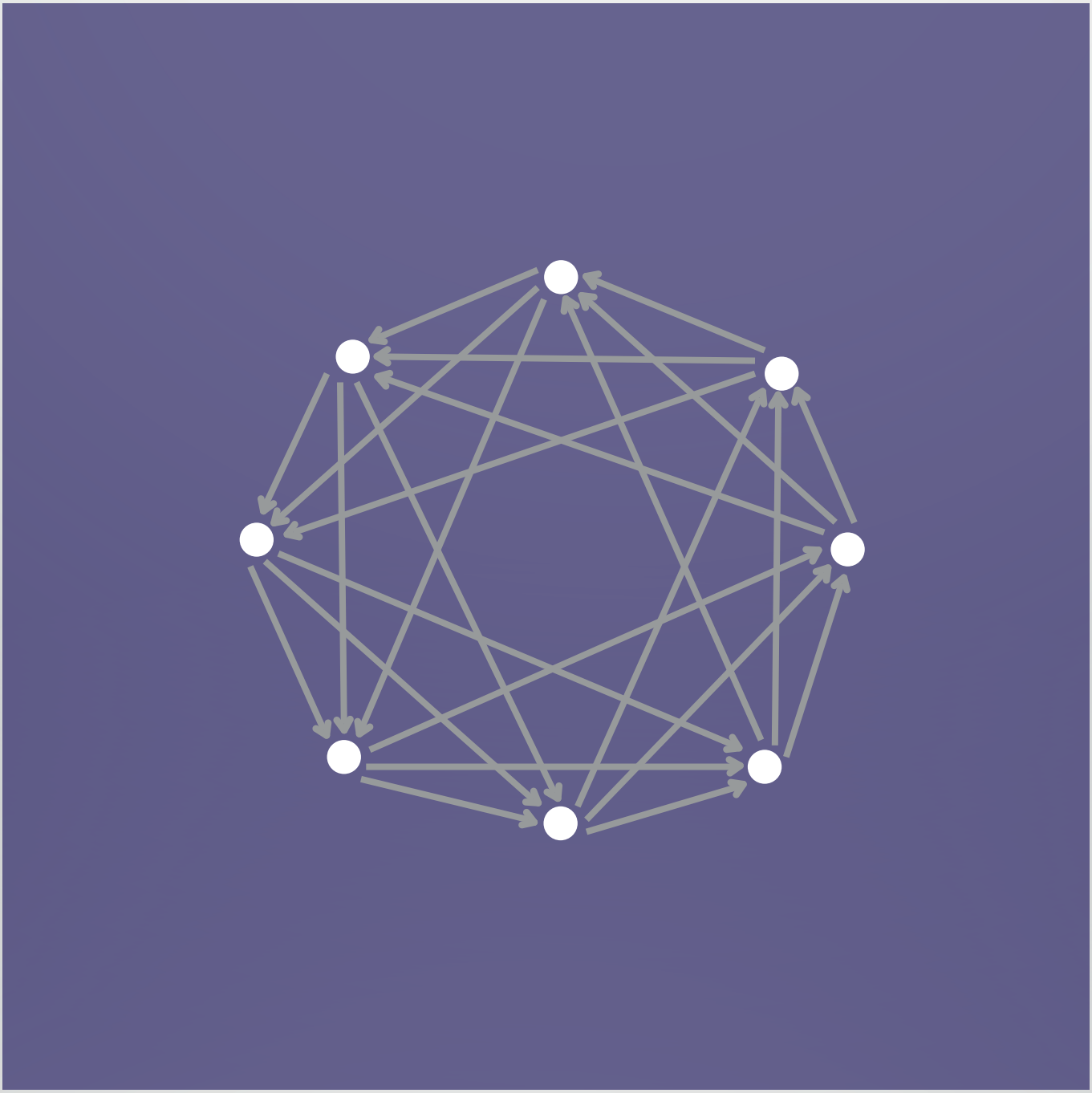
Each net member has their own pair of keys of asymmetric encryption both for buyer's public key encryption and sender's private key signature.
Identification of a node and its traffic is performed basing on the public key hash function.
The RSA (Rivest, Shamir and Adleman's) algorithm may be used for encryption and digital signatures with Diffie Hellman algorithm in a field of prime number with key length of 1024 bits or ECDHE (a type of algorithm having an elliptic-curve ephemeral key pair).
For the latter, key length of 256 bits is enough for a relatively similay cryptographic security.

Network organization

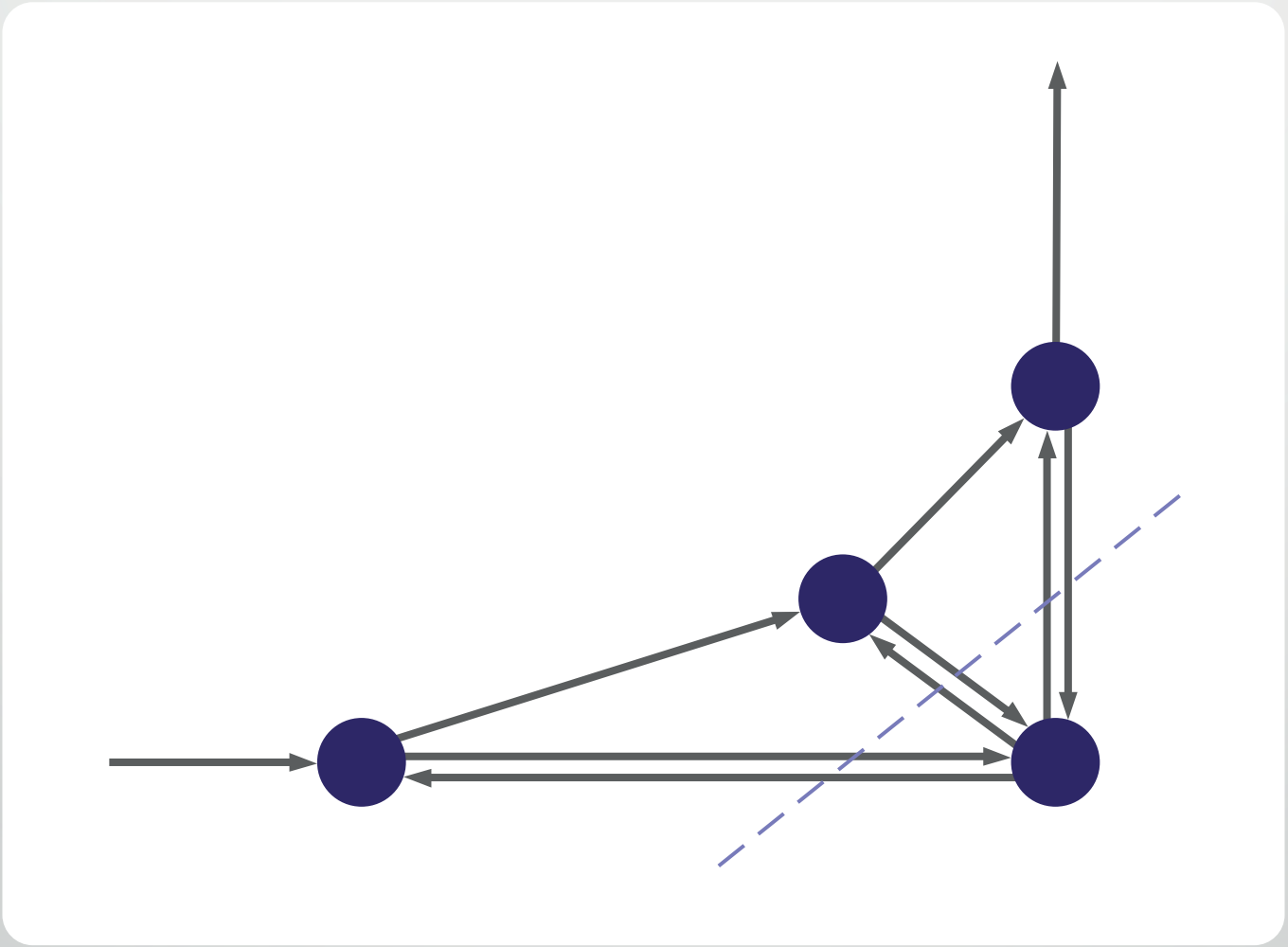
The network is a bidirected cyclic graph. Node identifiers are arranged and form a circle so that each node has right and left neighbors.

Creating a message, the node sends the package to all its neighbors. They look for the package recipient among the neighbors.

If they manage to find it, they send the message to it directly. If not, the left-receivers of the package send it to their right neighbors, and right-receivers send it to their left ones.



Excessive traffic provides protection from UDP losses, nodes shutdown on the path of the message, and attempts to counterfeit packages.
This scheme works only when all nodes can accept UDP packages at any time. The main requirement is a flat network with no routers and hierarchically nested private networks with so-called ‘private addresses’.
Nodes behind NAT can receive UDP traffic only in response to their request, only from the individual that request was sent to and only within the time interval that is the lifetime of ARP table entries of the routers.
If the nodes meet those conditions and adapt their behavior, they are allowed to participate in the network performance. By analyzing incoming and outgoing traffic, the nodes are able to understand the behavioral model they have to follow.



The criterion of the analysis is whether the node receives messages from those it did not send messages to. If not, the node arranges its performance so that it receives information within the interval of ‘open window’, which is to be supported.

DHT

The ‘circular’ organization uses equal distribution of hashes, which allows finding ‘neighbors’ in the ordered list of addresses and dynamically wedge newly created nodes between them. The DHT algorithm used in torrent networks works on ‘metrics’, used to complement the mechanism of selecting neighbors in a way that in reality they are located in different data centers on different continents. This is a protection from the Sybil attack, which allows for capturing the whole solution making block and take control of the network.

Or, by contrast, considering the node’s response rate, the metrics can improve the interaction speed that will increase the network speed due to really close nodes. Actually, a golden mean has to be found, that would be reliable enough in terms of safety, and fast enough in terms of transactional capacity.

PEX

The node exchange mechanism (peer exchange) allows rebuilding the network when new nodes are connected. By building into an ordered graph, the node finds its place, while its neighbors redistribute the subsets of their neighbors in a way that the UDP wave-length remains approximately the same.

Traffic

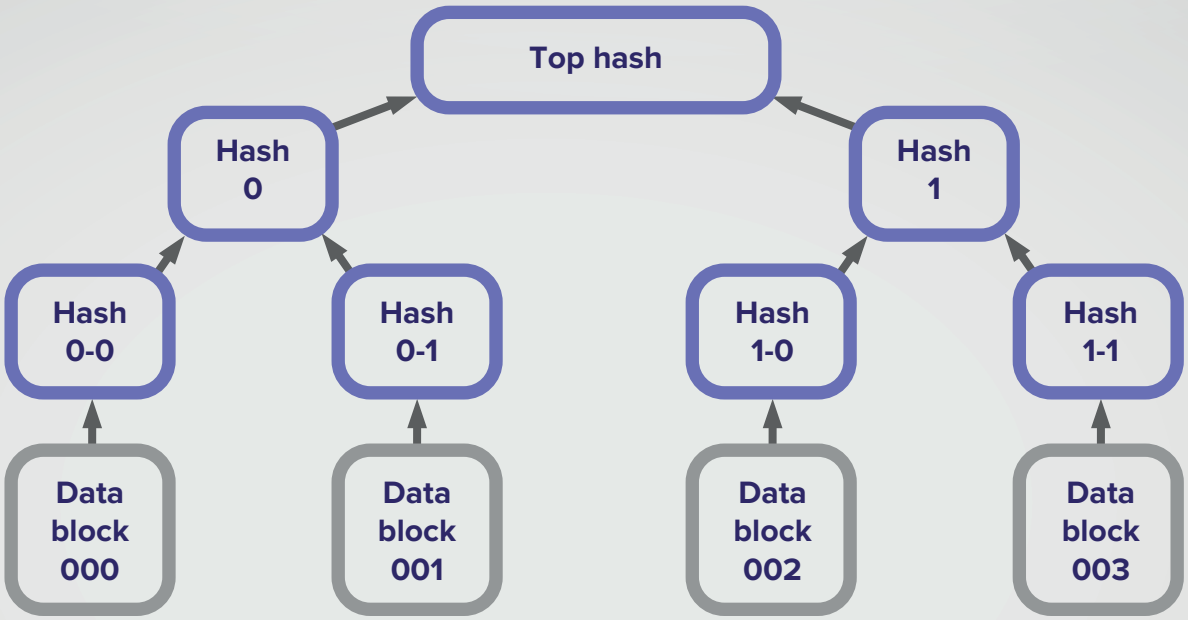
Networks work at the UDP layer. Packages produced by the node have their sender, receiver, command and a random payload of the command. To create a broadband message you just need to choose a non-existent address of the receiver (0, for instance).
Many commands allow nodes to register in the network, request missing information for the register synchronization, create transactions, send requests for participation in the solution making block, as well as commands needed to participate in the solution making block.
The node that produces a message signs it with a short-lived signature. The unique features of the signature are its time to live (TTL) and its low profile, which means it does not take much space in the UDP package. Given the short TTL, it does not represent a threat to security, but speed up checking and processing of such packages, making an unauthorized penetration or sending an 'alien' package impossible.
Verification of the short-lived signature is made at the network level; rejected packages will not move further that the first redirection and will not be factored at higher levels of the node software.
Big packages are split into several small ones which are sent as separate messages and aggregated only at the receiver's node. This is an additional measure for anti-interception protection.

REGISTER STRUCTURE

Register is a base of random data; its copies are stored and replicated at each node. Additional structures for integrity, credibility and applicability of the information are built over this data. Technically the database is arranged as a key-value storage, where key for a random value is its hash. The local storage for the database is the built-in Google's Level DB database.

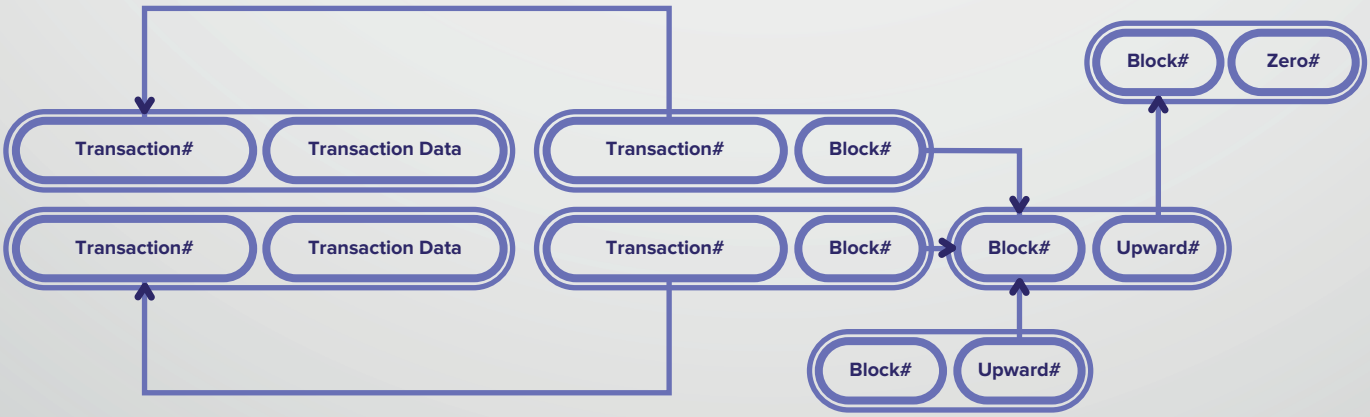
Storage and Merkle tree

Over the data entries, there's a binary tree and each node equals a hash created with pairwise combination of lower-level hashes. Thus, the entire data tree is being fold into one root hash that indicates the overall state of the database. The whole tree is checked so in case of some unapproved changes that hash would be changed and stop coincide with similar hashes if different nodes which shows that the database integrity has been violated. It is a modified algorithm of a Merkle tree.



Transaction Chain and Blockchain

A transaction chain is a list of entries that describe all the changes made to the database. It refers to the current database state (a root hash of the Merkle tree). These entries are united into blocks signed with e-signature of the block's creator and include hash of a previous block making a singly linked list - so called blockchain. This mechanism protects previously entered data from changes and authenticate those who make these changes legally.



Synchronization

As a database is a set of key-value pairs, databases synchronization comes down to unit-records exchange between nodes. Every time a transactional chain or a Merkle hash stop coinciding, a place of changed or/and absent entries is determined and they are acquired from the net. It enables not only to actualize latest changes but also ensures an initial synchronization and database restoring after a long absence of a nod in the net or an irreversible data loss in a local database.

Integrity Control

An integrity control is conducted every time the node starts working as well as evaluated local copy state is inconsistent with the last register state received from the net. Any breaches launch a synchronization cycle.

DATABASE ENTITIES

Wallets

A wallet is a personal account of a net member. To send and get money, each member needs a running node and a couple of e-signature keys. The wallet's identifier as well as the node's identifier is a public key hash.

Transactions

Carrying out a transferring, a net member creates a transaction, signs it with his/her private key and sends to the net. A financial transaction is an entity containing information about a sender, a receiver, a transaction sum, currency and optional additional data. These transactions make blockchain and are included in the database validation mechanism.

Random Entities

Apart from transaction information, a register can contain any other information of free formats. However, you can't save it directly. Each entity has an owner who is a smart-contract and can create, change or delete data within the register.

Smart Conracts

Smart contracts are kept in databases as stored procedures. It's a code which might be addressed according to its identifier. A smart contract's identifier is a hash of its source code.

TRANSACTIONS

Transactions created by net members are not part of the blockchain yet and are not included in the transaction chain. They must be validated first. This procedure comes down to the following two issues: a sender must have more money than he/she wants to transfer and the transaction must be signed with a sender's private key. Transactions that haven't passed this check are rejected while confirmed make a block which is signed with a validator and built into the blockchain.

"Fork" Problem's Solution

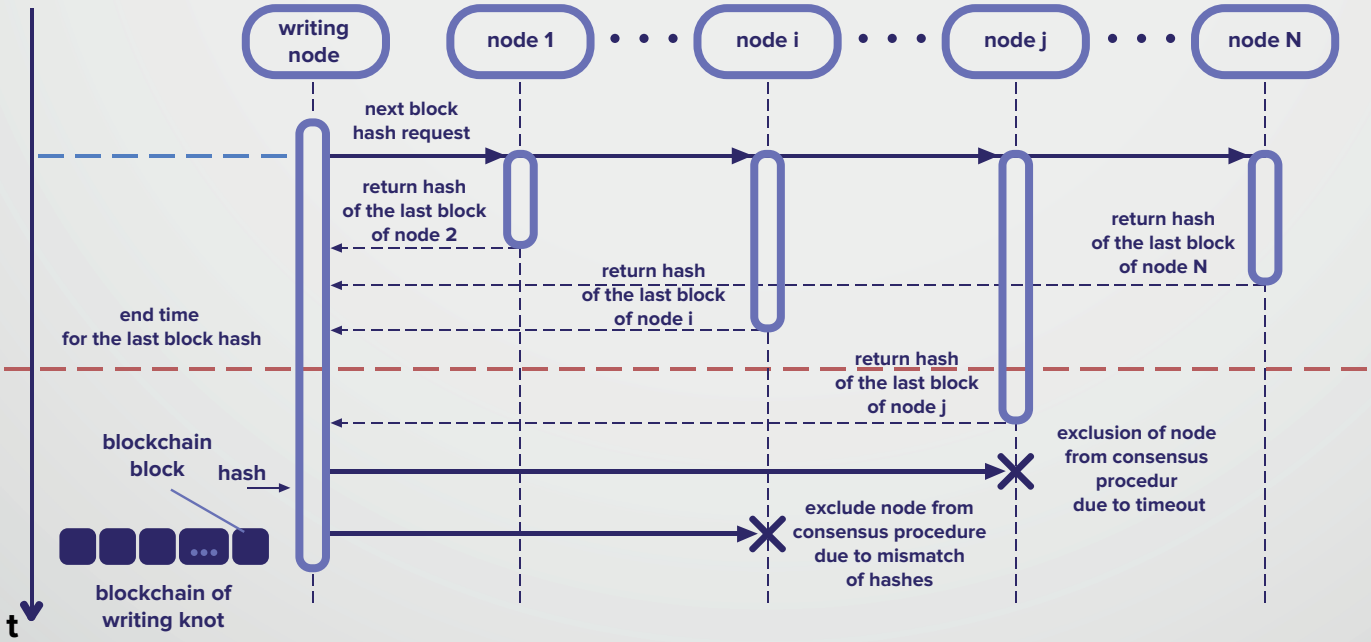
Blockchain is vulnerable to two forks. Where there're singly linked lists, there might be trees as well – nothing prevents several blocks from referring to one parental block. It can't be closed by a next block link as we do it for double-linked list because after being entered in the register, the blocks can't be changed.
As some other blockchains, we solve this problem by allowing only one entering block. Which validator will do it is decided during a verification round.

Decision Block

A decision block validates transactions and saves new blocks. It's a subset of nodes, which independently check data and exchange the results of decisions they make. In an untrusted environment (a decentralized net with nodes controlled by end users is initially untrusted), such mechanism helps to identify compromised nodes and get the correct result despite them. This method is based on solvation of Byzantine Generals' Problem, which helps to make right decisions based on partly reliable information.

Consensus Implementation

Consensus is a decision concerning validity of a transaction, which is made after all nodes have exchanged information on the validity of each particular transaction. After decisions concerning all the ready-to-enter transactions have been made, the system choses a node, which would enter the block. After this, the decision block is formed all over again with new nodes and the operation is repeated. By doing this we protect the system from forking – there's only one node that adds a new block.



Parallelization of Decision Block.

There might be more than one decision block. If we divide all the collected transactions into subsets, all the preparatory work might be done simultaneously and from cycle to cycle the writing right in decision-making process will be passed from one writing node to another. In this scheme, writing will be almost continuous and transactional capacity will multiply.

Transaction Graph Division

Presenting multiple transactions as a connected graph where nodes are senders and receivers while a connection between them equals sum of the transaction, helps to divide the graph into subsets optimally.
For this purpose, each subset must form a graph in which an overall balance of each node would be nonnegative. This task comes down to the Knapsack Problem solvation. The goal of this functional is to complete a list of prepared transactions so that ignore a minimum number of transactions that don't meet a general non-negativity graph condition. Transactions ignored at this stage will be passed to the next round for the repeated validation and completion.

FEES

Fee is a method to motivate net members on spending their computing power on the net functioning support. We need the nodes to work on productive equipment and have a high quality internet channel. Nodes that can respond faster are more likely to get into a decision block and have their fees.

Fee computation method

Before writing a block, the writing node completes transactions with the fee of each participant of the solution making block.Fees diving in 2 categories: with extra binary or not. Transactions without data has constant fee equals to 0.001% from amount. Transactions with extra data will have dynamically a calculated fee dependent to size of data and network load.

SMART-CONTRACTS

A smart contract is code which is executed during the creation of each transaction (when an identifier is pointed as a receiver). The identifier of the smart contract is its source code hash. Publication of a smart contract is a creation of a transaction containing its source code. The smart contract is put into a database and is executed when there's a transaction having its identifier as a receiver in the net. While being executed, the smart contract creates, modifies or deletes some entities in the database, which enables to keep the state between calls.
Smart contracts can also read information from blockchains and create transactions. As an example, we can mention a smart contract, which collects bets on a sports event and after getting the information about this event's results in the database, distributes winnings in proportion to combined distribution of the existing "bank". Another smart contract puts information that influences decisions into the database. Therefore, any facts from the outer world like quotations for currency, results of sports competitions, notarized documents can be smart contracts' entities.
Smart contracts are created with Lua programming language and have all its capacities inside an isolated virtual machine. Standard libraries are turned off, so smart contracts won't interact with the outer world.
Smart contracts get all the functions needed from a specially written SDK.
When receiving a transaction (when the receiver is an identifier of the smart contract), decisions block's members execute its code and after completion exchange hash state of the virtual machine including all the variables and entities. If all the decision block members have the same results of the smart contract execution, the transaction is saved in the blockchain and all the changed entities are synchronized in the net.
Smart contracts have no algorithmic limitations and it brings the risk of the system fail due to wrong or parasitical code that causes an infinite loop. To avoid this, smart contracts has limited time to execute a transaction, after that it stops and the transaction is considered invalid. However, the fee is charged and distributed among decision block members.
Entities that a smart contract can change must be initialized by it first. All the other entities can be read, but can't be written.
Smart contracts have neither input nor output, they can't send any network requests. All they can do is to read any entities from the register's local copy and create, modify or delete their entities.

API

Each node provides own binary protocol to interacte with other services. These requests might include transaction creating, smart contract publication, transactions list reading and retrieving values from the register.
API receives requests only locally so each software that is going to interact with the net and blockchain must have its own local node.
Wallet, which is a desktop application for making transactions, is an example of a third-party software that interacts with the net via API of the node set right at the same workstation.

INTEGRATION INTO EXTERNAL SERVICES

Widespread technology-based API, able to connect via many industrial programming languages and automatic systems, makes it possible to easily and almost seamlessly integrate the blockchain into an information system of any complexity.

IoT

Telemetry data from small autonomous systems may be stored in the blockchain, making it possible to save all telemetric record in a distributed ledger where it will be accessible and protected from changes.

Logistics

Due to the distributed storage in the blockchain the information on the relocated objects may be accessed from any place with the Internet connection. And all participants of the process, from the manufacturer to the customer, will get real-time information on the delivery progress.

Payment system

Use of RAS cryptocurrency for calculations. The business can use the RAS cryptocurrency as a payment unit between the parties of the transactions.
This will enable automatic exchange of RAS for fiat and other currencies through the exchange's API. Using other cryptocurrencies for transfers.

Gaming

Blockchain technologies will increase trust in game services due to openness and guarantee of preventing fraud by the organizators.
Blockchain provides the ability to perform transactions not from bank cards and accounts, but using Rasmart coins (RAS), that is, the crypto currency is used as a settlement facility inside the game.

Copyright protection

Our Technology allows to register information about a work in an unchanged and decentralized network of detachments. The service user downloads a unique number (hash) of a work created using a special program.

The rightsholder places this number in the distributed registry of the Rasmart blockchain After the data on authorship are recorded in the blockchain, it is no longer possible to change them. Data specified in the blockchain can not be changed Availability of the service - the record about authorship can be made by any participant of the system, reading information is free. Secure storage and updating of data on digital objects of intellectual property. Protection of both the author and the user from violations of their rights.

Exchange

Automation of payment systems of the banking industry with the possibility of scaling at the level of a separate unit and inter-branch exchange. Opportunities A unified protocol for the issue, storage, transfer of financial assets Distributed network with a public registry system of accounts with public and private keys. Security systems. Built-in programming language for creating services.

Loyalty

Traditional loyalty programs are difficult to maintain and inconvenient for users. Blockchain Rasmart, being an operating platform of a new type, is able to change the situation Blockchain Rasmart can act as an operating platform where all interaction occurs, and tokens released on the platform are scores accrued for actions.

Streaming Services

For such services can work as transactions(network module) with powerfull of smart-contracts and adding not only smart-contracts to transactions