# EMPERA Protocol

ver 0.4 (06.04.2022)
progr76@gmail.com

**Abstract** **2**

**Introduction** **2**

Choice between POW and POS 2

Weak points of modern blockchains 3

Poor performance. 3

DApps and centralization 3

Isolation of blockchains 4

Unpopularity. Development challenges. 4

**Architecture** **5**

Consensus 5

Network time 6

Synchronous state of blocks 7

Network 7

Network traffic 9

Runtime environment 10

Miners’ motivation 10

Pipeline processing 10

Processing layers 11

Proof of Memory (PoM) 12

**Security** **14**

Protection from double spending 14

Protection from replay attacks 14

Protection from DDoS attacks 15

Protection from Sybil attacks and Eclipse attacks 15

**Sharding** **16**

Introduction to sharding 17

Sharding in EMPERA 17

Sharding terms 18

Shard coins 18

Node 19

Horizontal scaling in merged mining 19

Shard miner’s motivation 20

Messaging pattern 20

Cross-sharding transactions (messages) 21

Channel security 22

Sending coins from the wallet between shards 23

Conclusions 23

## Abstract

By the beginning of 2022, blockchain platforms faced new challenges - centralization of validators and low popularity among ordinary users. The reasons are as follows: subjectivity of consensus, low blockchain compatibility (interoperability), complexity of development and use of DApps.

The following document describes how to reuse POW consensus by introducing a more rigorous definition of it. What if the selection of block chains in the Nakamoto consensus is based not on an artificial complexity function, but solely on the amount of work done:

- Will this lead to an increase in the network performance?
- Is it possible to build a network with unlimited scalability?

It also demonstrated how to use technologies from the web in a new way and build an original, truly decentralized web3. What if the most popular programming language JavaScript is applied:

- Will it allow processing transactions up to 1000 tps?
- Will it make the creation of DApps easier?
- Will it attract a large number of new developers?

## Introduction

### Choice between POW and POS

Existing cryptocurrencies can be conventionally divided into two types of consensus:

- POW (Proof-of-work): Bitcoin, Ethereum, Litecoin, etc.
- POS (Proof-of-Stake): Decred, Novacoin, Cardano, etc.

When choosing between POW and POS consensus, we were guided by the following:
Despite high energy costs, POW has a low entry threshold - an ordinary computer or laptop is enough to launch a personal node, which will participate in the creation (mining) of blocks. Moreover, as it will be shown below, to generate unlimited scalability.
POS is characterized by a “Nothing at Stake” vulnerability and demonstrates a high risk of centralization, while the network scalability is limited by the capabilities of the hardware used by validator nodes.
Other POS based protocols possess similar weak points:

> DPOS is a type of POS with a limited number of validators;
> POA (Proof of Authority), by nature, is also a type of POS, in which the stake is the node authority with a weight of 1. This is equivalent to distributing voting coins to certain trusted people.

In consensus protocols, the decision is made by the algorithm. The point is that for the algorithm the whole world is limited by a black box and it cannot identify the validity of a particular event in the physical world. But there are two things that it is still able to check:

1. By means of cryptography: digital signature if the algorithm contains valid information about the public key (for example, from the genesis block)
2. By means of POW: capacity of the equipment that was used for the block calculation.

The advantage of POW lies in mathematics - the capacity of the work performed can be transferred to a huge number of devices and shards (other connected parts of the network), since it is:

1. cost effective regarding the amount of information (only the block hash and the nonce number are to be transferred);
2. fast in time (transfer is performed according to the logarithmic dependence on the number of nodes);
3. valid from the point of view of proof (the number of leading hash zeros is considered, the probability of obtaining which depends only on the equipment capacity).

POW-based proofs can be scaled to the maximum and implemented across all nodes around the globe to maximize transaction security and network no-failure operation.

At the same time, information in POS is valid (i.e., can be proven) only within one network, since it relies on verification of a digital signature. Nodes are able to physically check the chain of transfer of the coin ownership (stake) in only one network, and while other networks do not have such an opportunity by design (for them, the coins of other chains are just numbers), it is impossible to create a protocol with a reasonably large and, particularly, unlimited scalability.

## Weak points of modern blockchains

### Poor performance.

The main problem of POW-based blockchains is the low speed. There are two important disadvantages:

- Nodes are not able to quickly synchronize blocks with each other. The connection of nodes to each other is random, there is no protocol for optimal networking;
- Time of the new block creation is random. The Nakamoto consensus uses the calculation of the hash of the required target complexity as a sign of block formation. Hash calculation is a random value, so the time is also random.

A comprehensive solution to these issues allows achieving blockchain performance at the level of 1000-2000 tpx per processor core.

### DApps and centralization

DApp (Decentralized Application) is a decentralized application based on blockchain technology along with the mechanism of the necessary instructions distribution. However, at the moment the term is being used incorrectly. It refers to a program that interacts with smart contracts in the Blockchain, but is actually located on a centralized server. The part that is located on the centralized server is critical, and functioning is impossible without it. A decentralized application of such a type can only guarantee the safety of user funds, as they are located on the Blockchain. Experience has shown that failures are possible by a mass influx of users to DEX/Swap, for example, with sharp changes in the rate.

Such a situation has arisen due to the fact that the existing Blockchains on the market do not actually provide hosting services on their platform. There is practically no such notion as a "user interface" in any of the projects.

### Isolation of blockchains

Large-scale projects often require high performance and their own algorithms for emission, reward distribution, inflation, etc. Therefore, they launch their own blockchains. But these blockchains are isolated from all others, which leads to a liquidity problem and inconvenience for users. To solve these problems, they create cross-chain bridges, which demonstrate new disadvantages in turn: centralization, low speed and security issues.

EMPERA is not just a blockchain - it is a blockchain platform. There are no forks, since each fork is a shard that can have miners in common with all other shards. Thus, it becomes possible to conduct cross-sharding transactions between them.
Creating a new shard (blockchain) is a simple and free process independent of third parties.

### Unpopularity. Development challenges.

Blockchain applications are extremely uncommon among users.
Two reasons can be distinguished:

- Complex entry for common users. Getting started with such an application requires additional steps and knowledge, which can be an impassable barrier for most. For example, installing a plugin or initializing a private key.
- Application development requires knowledge of specific programming languages, which are of high complexity and low popularity among programmers. In addition, some blockchains possess a controversial architecture:
    - Slow virtual machine (stack-based approach)
    - One runtime environment data type (uint256)
    - Incomplete programming language which cannot function in full without low-level (assembly language) insertions

Wide use of blockchain technologies can be boosted by the simplicity of developing smart contracts and dApps. To achieve that, the programming language should be as simple and clear as possible.
For example, it is necessary to use the most popular, modern and widespread technologies (JavaScript and HTML), adapted for the blockchain.
Web programmers represent the largest army in the IT industry. They are the closest to users, they will be able to create popular applications on the Blockchain at the earliest and make them as user-friendly as can be.

All this will attract hundreds of millions of new users to the blockchain industry and erase the boundaries between dApps and ordinary applications.

## Architecture

**EMPERA** is a platform for creating blockchains, as well as dApps within them. It consists of a program store, a data store and a decentralized communication protocol. The mechanism for publishing programs and data is free from censorship. Using peer-to-peer sharding allows to create an unlimited number of peer-to-peer subnets (shards) and thus increase the overall network performance almost endlessly.

### Consensus

Consensus algorithm is an automatic mechanism by which a blockchain network achieves a synchronous state of all its nodes.
Let's consider how this is implemented in the EMPERA blockchain:

The user signs the transaction and sends it to the network.
The network forms a new block of received transactions every three seconds. Each block contains the hash of the previous block, which allows them to be connected into sequential chains. The information written in such chains is protected by the PoW consensus from being overwritten.

The block hash is a SHA-3 one-way cryptographic function possessing the capacity determined by the number of its leading zeros.
Finding a hash with a larger number of zeros is a random process and depends on the number of searches for nonce numbers and the miner's account number.
Nodes are aimed at calculating the hash with highest capacity in order to find the leading hash and receive a reward.

> Difference from Bitcoin
> In EMPERA, miners compete to find the leading hash. If they manage, they distribute the information only about the found nonce and their account number over the network to receive the reward. The traffic is minimal. In Bitcoin, miners look for the leading block and, if found, they distribute the whole block over the network, which leads to excessive traffic and, therefore, delays.

In EMPERA, the chain length depends only on the network global time. If the current time is known, then it is possible automatically calculate the number of the current block, and therefore the length of the entire chain.
The main goal in consensus is the rule (program code), following which any independent third party can unambiguously determine which chain is the main one (leading). Thus, the consensus rule in EMPERA is as follows: **the leading chain is the one with a larger sum of capacities of all blocks.**

In order to create a leading chain, each node adds a link to the previous block with the maximum chain power to the new block.

> Difference from Bitcoin
> In Bitcoin, a new block is formed only when the target value of the block computational complexity is reached (the number of leading zero bits in the hash), and the leading chain is determined only by its length. The time of its creation is recorded in each block, after a specified amount of time (epoch) a new target complexity is recalculated.
> This algorithm shows disadvantages, since an attacker can create a chain of greater length, spending a low computational power by writing a fictitious creation time to each block, so that the complexity recalculation algorithm will not increase the target complexity. Thus, the third party will not be able to determine the correct chain by length alone. Bitcoin implies other mechanisms of protection against such actions, such as checkpoints (written hash codes in the code itself). In EMPERA, this is not required, since the mechanism for determining the leader is built-in the architecture itself.

## Network time

To reach consensus, nodes shall synchronize with each other. Synchronization speed directly affects the throughput of the blockchain.

In order for the network to work synchronously, the time shall be approximately the same.

The time in EMPERA is used to determine the current block number. Depending on it, each node decides which blocks to process - i.e. to receive from other nodes, search for the maximum hash of the leader and create new blocks.

EMPERA allows a one-unit deviation in block processing. Thus, the previous, current and subsequent (block from the future) blocks are processed. If the block number does not coincide with the current one according to the node, it only transmits (translates) information about it over the network.

The following protocol is used for time synchronization:
- The median of the obtained selection is calculated
- A drift constant is added to the absolute time according to UTC ("wall clock")

More details:

https://docs.google.com/document/d/1MWmqTvO9Zi5pKZzpnCGogQGBSwbEmMmLXSrVQOaPv1E/edit?usp=sharing

## Synchronous state of blocks

Each block consists of transactions sent by users. In order for the composition of blocks to be the same in all nodes at once, the following rules are applied:

- The number of transactions is limited
- Sorting of transactions by priority is applied
- Extra transactions are cut and sent in the following blocks
- The priority depends on the transaction size and the complexity of execution (the number of ticks of the smart contract), the number of coins on the sender's account and the number of previously sent transactions

Accordingly, if each node contains the same set of transactions (this is achieved by the network protocol of guaranteed delivery time), then they will have an identical composition of blocks.

## Network

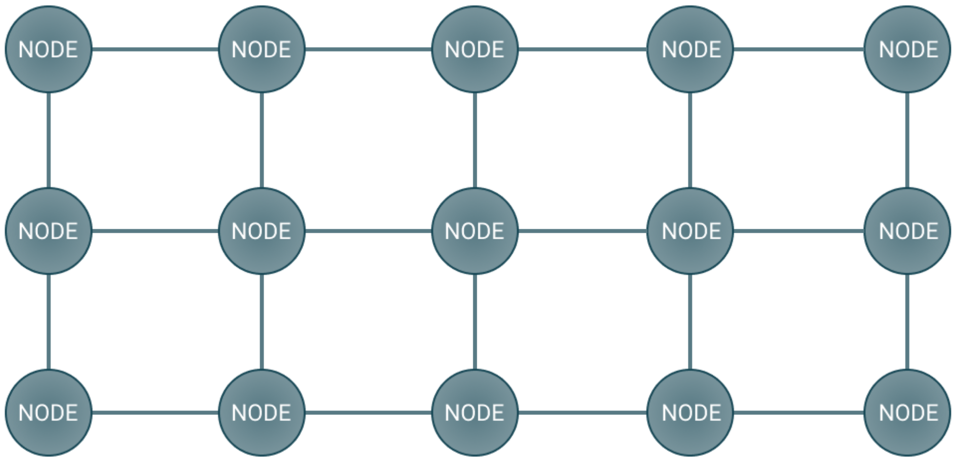
We use our proper **Crystal DHT** protocol, which provides:

- High node connectivity
- Constancy of transaction delivery time throughout the network

Such an approach allows for guaranteed delivery of transactions to all nodes of the network.

**Protocol description**

All nodes are self-organized into consecutive links, forming a multidimensional regular grid. The grid is shown below in its two-dimensional representation, but in real terms it has a multidimensional (8-dimensional and above) structure.



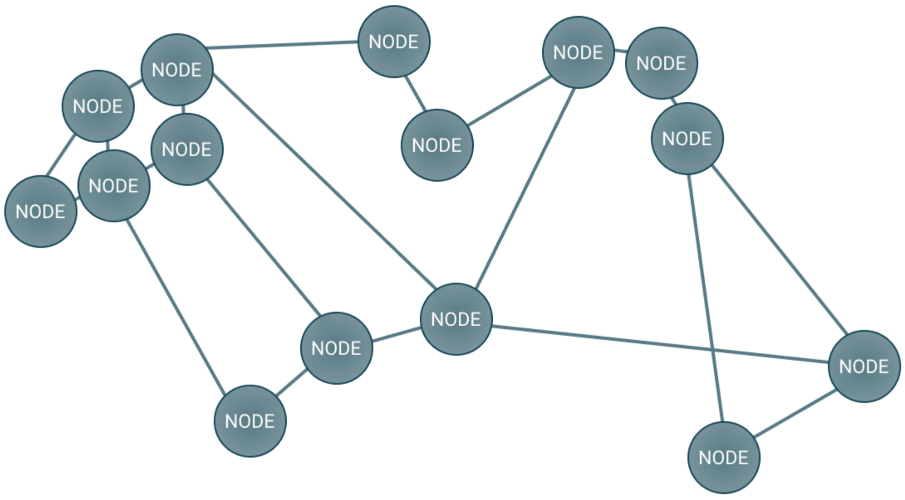
Nodes form links based on the similarity of their network address hash. Such a hash is a random value with a length of 32 bytes which does not change during the node performance. The hash randomness is used to divide the nodes into levels of connections with an exponentially decreasing number. At the first level, 1/2 nodes are available, at the second - 1/4, then 1/8, etc. Within each level there is a priority in the form of work with a more "responsive" node. Since the nodes are unevenly distributed through the levels, the faster nodes dominate at the beginning of the levels, followed by the more rare (i.e. random) ones. The lower levels link the nodes into local groups (i.e. the nodes that are in the same regions), the upper levels form the links between the local groups. Note that one such link is sufficient for a local group. The larger the group, the more likely it is to find at least one link.
In other words, this algorithm blocks the creation of closed local groups not linked with each other.

<u>Example in figures:</u>
The maximum distance of the regular node network depends logarithmically on the number of nodes in the network, which provides a constant transaction delivery time. So, if there are 1,000,000,000 nodes in the network and the average delivery time between nodes does not exceed 100 milliseconds, then the maximum time will be: 10 * 100 ms = 1 sec. 10 is the logarithm to base 8 (the number of active links) of 1 billion nodes. Simultaneously, the time of transaction delivery delay is less than 100 ms within a local group, and no more than 500 ms for connection with another continent.
Thus, it takes no more than 3 seconds for the blockchain to deliver data from the 1st node to the last one.

In order to “cement” the successful links, each node records for itself the statistics of successful content exchanges with another node. This statistic affects the priority of connection and provides protection against eclipse attacks, in which malicious nodes try to surround a single node and thus distort the information about the network and censor blocks and transactions.

<u>Differences from other networks</u>

Traditional blockchains do not use the ordering of nodes with each other, they do not have network-forming protocols - instead they use a gossip approach, according to which information is distributed randomly, and the diameter of the network is not constant. The general pattern of the network is the following:



This random organization of links does not guarantee fast delivery of blocks between all nodes. Therefore, the block formation time in other POW blockchains is in the range of minutes.

## Network traffic

To reduce traffic we use two approaches:

The first is protocol (which can be metaphorically compared to nodes blowing a light wind across each other before sending out a powerful storm of data)

- Firstly, nodes send each other short hashes from transactions (light wind)
- Then they send the data that was not captured in the first step

Such a protocol allows to favorably (i.e. perfectly) configure the transaction delivery traffic.

The second approach consists in sending only leader hash information, instead of sending leading blocks like in other blockchains. This allows to reduce technical traffic (i.e. when there are no user transactions) to approximately 4 Kbyte/sec.

Blockchain network simulation can be performed at the following link: https://terafoundation.org/JINN/model/model.htm
The peculiarity is that the code (jinn library) is the same as in the mainnet EMPERA.

## Runtime environment

JavaScript smart contract execution language runs in the V8 environment developed by Google. It compiles the source code directly into its own machine instructions, bypassing the bytecode stages. This provides high performance and allows processing thousands of transactions per second on a single core.

JavaScript is Turing complete, so the term Tick is used to limit the smart contract runtime, which is equal to the execution of one code line. The maximum size is limited by a constant. Tick is an equivalent of GAS in Ethereum, but no commission is paid for it.

For the sake of security, the smart contract code is pre-translated/pre-compiled and executed in a special protected virtual environment.

## Miners' motivation

Each node, in addition to strictly following the rules for creating a leading chain, can participate in the race to calculate the maximum block hash in order to receive a reward in case of success. In this regard, it goes through the nonce numbers and sends the maximum capacity values to the network (in this case, only the proof is sent: a link to the current block, nonce numbers and the account number for the reward).

The hash with the maximum number is added to the block, and the miner who found it during the run of the block receives a reward in the form of a system coinbase transaction, which is implicitly added to each block.

The reward size is determined through DAO by each shard separately in accordance with its own tokenomics policy.
*DAO is a Decentralized Autonomous Organization operating on the blockchain and controlled by the software code.*
For example, 1-2 coins each block and an inflation target of 2% per year.

## Pipeline processing

To ensure high performance of the blockchain, we use pipeline processing, which consists of the following steps:

1. "*STEP_ADDTX*" - adding a transaction to a new block
2. "*STEP_TICKET*" - sending information (ticket in the form of a short hash) about the transaction to neighboring nodes
3. "*STEP_TX*" - sending the transaction itself (if there is no such one in the neighboring node)

4. "*STEP_NEW_BLOCK*" - mining a new block and distributing information about the leading hash
5. "*STEP_SAVE*" - writing a new block to the database on the disk
6. "*STEP_USE_TX*" - changing account and KeyValue storage states
7. "*STEP_RESEND*" - re-sending transactions not included into the block

As can be noted, there is no block sending step in the pipeline, since due to the protocol features, the blocks are the same in all nodes already at STEP_NEW_BLOCK.
This is achieved by the fact that transactions are guaranteed to be delivered to all nodes, are equally prioritized within the block and a coinbase transaction is added automatically at STEP_USE_TX.

## Processing layers

We have divided processing into two main layers:

1. Layer of node synchronization and block delivery
2. Layer of transaction execution (application)

Let us recall what a blockchain is: it is a computer network in which each node is equal, the number of nodes is not limited, communication between the nodes is carried out by means of organizing an integrated data chain, in which information is written block-by-block in the form of commands (transactions).
In Bitcoin-like blockchains, only payment transactions are written in blocks, and invalid transactions are not allowed to be written (for example, a transaction which does not have a correct digital signature or enough money on the input address, or contains double spending, etc.).

In EMPERA, as well as in other blockchains such as Ethereum, this is possible. This is done for the sake of processing speed, and also due to the fact that it is impossible to know the result of processing a separate transaction outside the composition and order in the block in advance.

EMPERA blockchain allows writing any information in the block, and the blockchain is used as a transport. Each record (hereinafter referred to as transactions) has its own strict numbering and is divided into blocks, the blocks are linked to each other by means of a one-way cryptographic hash function. At the first layer, the task of the blockchain is to ensure the same information in each node of the network. This task is accomplished through the classic **POW** consensus.

The interpretation of the information correctness lies on the second layer. The second layer implies the cryptocurrency support - the built-in **EMPERA** coin and the accrual of block rewards to miners. Smart contracts are implemented on the same layer.
As the first layer guarantees the uniformity of data, and since the program code on the second layer is the same in all nodes of the network, it is obvious that performing the same

actions, all nodes will show the same result: the same balances on user accounts, the same statees of smart contracts. Thus, if these blocks contain invalid transactions such as double spending, then the validating layer will reject them equally in all network nodes.

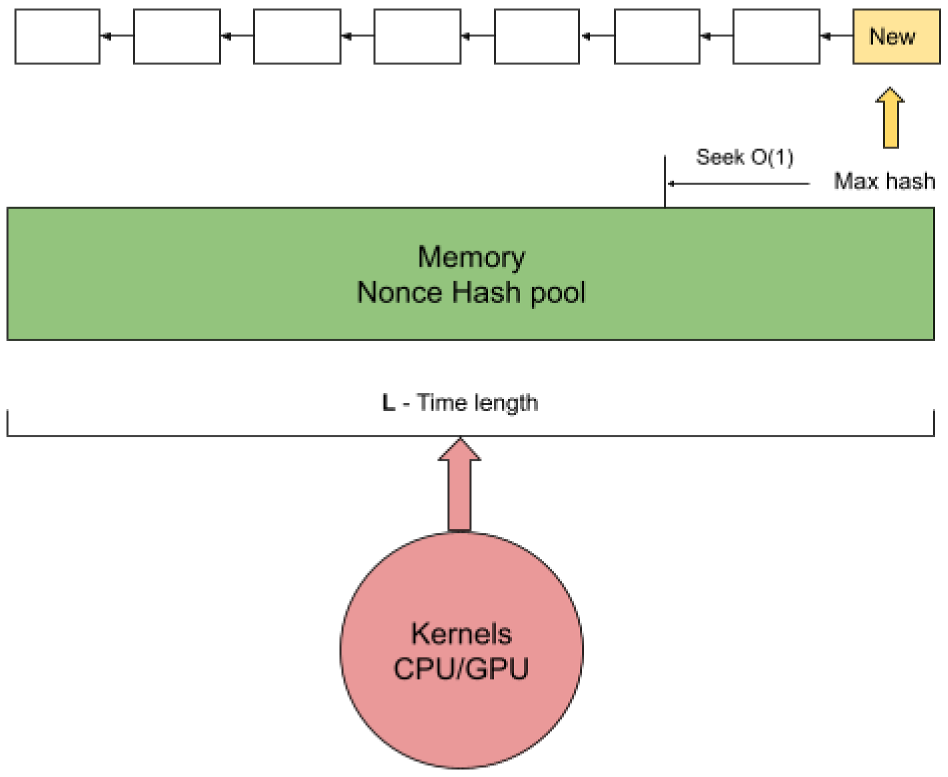
Validation can be performed at a different time and in a different process without interfering with the blockchain. Moreover, this can be done much faster by means of the so-called batch processing (mass validation) - it is possible to group operations and speed up the process due to fewer Database queries.

## Proof of Memory (PoM)

The goal of this algorithm is to equalize between each other mining equipment based on CPU, GPU or ASIC. For this purpose, we propose to use memory, but unlike other similar algorithms (such as Ethash), in our algorithm memory does not limit, but improves the performance of the target equipment. This can be accomplished by not using an integer nonce, but rather a specific computationally-intensive value - such as the one computed by the SHA-3 algorithm - and allowing that value to be reused over time to find the best block hash. Thus, it will be more advantageous to store these values in memory than to recalculate them.

In this algorithm, we apply the time parameter **L**, which indicates the validity of previously computed hashes in the pool. Thus, it sets the amount of memory precedence over computations. The hashes calculated during this time and written to memory are then quickly retrieved from memory whenever a block hash is calculated, while the purely computational hardware (ASIC) requires a new calculation.

The scheme of creating new hashes and using memory to select the best hash:



After filling all available memory in the pool, the computing cores can be stopped, which will lead to power savings.

NonceHash calculation formula for pumping pools in memory:

*NonceHash* = *sha3(Time, PubKey , Nonce)*

where:

*Time* - time of nance hash calculation from the valid range

*PubKey* - the public key of the miner

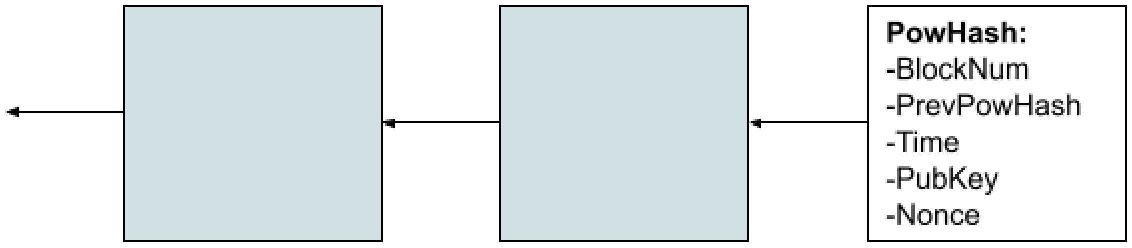
*Nonce* - random number

Parameters are sent to the network: *BlockNum,PrevPowHash, Time, PubKey, Nonce* which are the proofs of the work done. Each node checks the validity of time and recovers the power hash using the formula:

*NonceHash* = *sha3(Time, PubKey, Nonce)*

*PowHash* = *XOR(sha3(BlockNum,PrevPowHash), NonceHash)*

Blockchain Diagram:



## Security

From the point of view of security, the EMPERA blockchain has a number of advantages over other solutions due to its technological features:

Transactions are delivered to the most recent block. This block is called current. After the next 3 seconds a new block is created, then another block, and so on. Moreover, each block is linked to the previous one, forming a chain. The number of such blocks that are created after the block with the sent transaction is called the number of transaction confirmations.

The protocol has a probabilistic finality of the transaction - the more confirmations, the higher the impossibility of rewriting.

As long as the number of honest miners is more than 50%, the attack is considered impossible, since the probability of rewriting a block is less than **0.5**, and the product probability tends to zero with an increase in the number of confirmations.

### Protection from double spending

A transaction is protected from double spending if it is practically impossible to overwrite the blocks (i.e. probability tends to zero).

Double spending is the process of conducting two payments within the same network and the same coin. As a rule, the purpose of such actions is fraud.

To be protected from double spending, Bitcoin users usually wait from 10 minutes to an hour for the required number of confirmations.

In the EMPERA network, blocks are generated faster, but if you want to reach the same degree of reliability as in the Bitcoin network, you need to wait a comparable time. Time of waiting is the degree of reliability. There is no magic - in any POW algorithm you trade time for reliability. In EMPERA, we have made the choice more flexible. You can wait 1 minute or even 1 hour (if the amounts of transfers are significant).

### Protection from replay attacks

EMPERA provides protection against another type of attack that is often used by cybercriminals - Replay Attack.

Replay Attack (also known as a repeat attack or playback attack) is a cyberattack in which valid data transmission is maliciously or fraudulently repeated or delayed.

EMPERA is based on the Blockchain with POW consensus. This allows arranging all transactions sequentially one after the other. Transactions are also executed in series. When debiting from the account, the availability of the required amount is checked. At the moment of charge-off, the counter of the operation number ("OperationID") increases. Each next payment transaction shall have the subsequent or higher "OperationID" to prevent the same transaction from being applied multiple times.

## Protection from DDoS attacks

For protection against DDoS attacks, the block size in EMPERA is limited by a constant (for example, 350Kb), and the number of transactions in a block is also limited (for example, 3000).

DDoS (Distributed Denial of Service) is a type of hacker attack based on distributed denial of service caused by server overload.

Transactions in blocks are sorted by priority, and excess transactions are cut off and limitedly redirected to other blocks.
The priority depends on the size of the transaction and the complexity of execution (the number smart contract ticks), the number of coins on the sender's account and the number of previously sent transactions.
Thus, the more transactions are sent from a particular account, the lower the priority and the less likely it is to fall into the block.

## Protection from Sybil attacks and Eclipse attacks

In the network, all nodes are equal and anonymous, which enables Sibyl attacks, when an attacker creates a large number of nodes and tries to conduct malicious actions.

Sibyl Attack is a type of peer-to-peer attack in which the victim connects only to the nodes controlled by the attacker.

A “good” node in contrast with a “bad” node strictly follows the protocol. A “bad” node:

- does not transmit information when needed
- transmits information when it is not needed
- transmits false information

The goal of each node in the network is to correctly transmit information. The result of a correct transmission is a successfully synchronized block and the absence of orphan chains. This indicator is visible a few seconds after the exchange and is objective, since it is protected by the POW consensus, so it cannot be faked without 51% of the capacity (but in our reasoning we always assume that 51% of the capacity in the network are honest).

In summary, we have a fairly good mechanism for determining the correctness of following the protocol of those nodes with which we exchange. Therefore, in order to “cement” the successful links, each node records the statistics of successful exchanges with another node. This statistic affects the priority of the link formation when creating a network in the form of a multidimensional regular grid.

## Sharding

The concept of blockchain as a philosophy of decentralized transfer of values has become deeply embedded in society and raised a need to create such a global network that is completely decentralized and has no scalability problems.

Based on the new sharding protocol, it is possible to create a network with millions of nodes and overall performance measured in millions of transactions per second.

Sharding is a method of splitting and storing a single logical dataset in multiple databases.

However, in EMPERA, sharding refers to the division of a common network into components with a single security mechanism (by the same miners), working as a single network due to cross-sharding transactions.

The basic principle: an unlimited number of peer-to-peer blockchains connected to each other through a common network hash are created. Blockchains communicate with each other directly through cross-sharding transactions. Since the network does not have a main chain through which such transactions are carried out, there is no bottleneck, which allows for almost unlimited scalability. The security of the network is provided by the outer loop of the lightweight blockchain - it consists only of the headers of common hashes. The network miners perform the work on the selection of such a common network hash and validation of the child blockchains.

## Introduction to sharding

It is possible to ensure maximum decentralization in the form of an almost unlimited number of block producers by POW consensus. But a simple division of network nodes into shards in the form of separate blockchains, although it leads to a multiple increase in overall performance, also leads to a multiple loss of security. It becomes easier to carry out a 51% attack on each blockchain separately. Indeed, if the entire network is divided into 100 shards of equal mining power, then 0.51% of the mining equipment capacity of the entire network is enough to attack a separate blockchain.

To prevent such a threat, merged mining can be used. In this case, miners only calculate the total network hash, the complexity of the hash remains the same, and thus the security of the network is not compromised. But there arises a problem with the correct validation of shards, since the resources of one node are limited (to guarantee decentralized mining, we will assume that one node can validate no more than one shard).

The main idea for solving this problem is to use the social aspect of miners - "six degrees of separation". Miners can have multiple nodes, miners can have relations (trusted friends).

## Sharding in EMPERA

In practice, a simple question arises: what should do an ordinary user who has downloaded a mining program and possesses only one computer and resources enough for validation of only one shard?

POW consensus comprises a noteworthy relationship between the network security and time. The more time it takes to generate a hash - the more secure the network is, and vice versa. This is a kind of law that cannot be circumvented, but is inevitable. Hereof it follows: if there are not enough miners supporting the validation of the shard, then this shard will have a confirmation block less often in time. Shards that have a fairly rare number of confirmations will be less preferable for users, which will affect their market capitalization (since each shard is an independent blockchain with its own cryptocurrency).

It will be like this: the miner indicates the connection between his nodes and the shards that these nodes validate. During mining, **only** hashes of validated shards are included in the hash tree (Merkle tree) of the winner's block. Users will consider a shard block confirmed only if it is included in the common network hash.

Thus, a miner with only one node will have only one shard validated, and a miner with multiple nodes will have almost all shards validated. Since the miner with a larger number of resources (nodes) will gain the probability of finding blocks, then:

- If the network is dominated by the nodes of professional miners with a large number of nodes, then most blocks will contain all validated shards.
- If the number of nodes (with conditionally equal capacities) is equal, then averagely all shards will be validated through one block.
- If there are more individual miners, then the probability of shard validation will be in direct proportion to its popularity (and inversely to the total number of shards).

Accordingly:

- A new miner with a single node can easily participate in mining, while he is motivated to validate other shards at any time he wishes.
- To introduce a new shard, it is enough for at least one miner to validate it. The confirmation time for shard blocks depends on the popularity and market performance of the shard.

Taken together, all this provides a smooth and organic mechanism for the formation and maintenance of shards, which will be regulated by market methods - by changing the rate of the built-in shard cryptocurrency.

## Sharding terms

**Network:** this term is understood as a virtual (relay) network in the form of connection of nodes (computers) with each other in a certain sequence, grouped by different shards (blockchains) and operating under the same data transmission protocol.

**Shard:** in this version, a shard is understood as a separate blockchain validated by the miners of the common network. Shard is a part of the network.

## Shard coins

Each shard has its own embedded coin (cryptocurrency). Cross sharding transactions are actually cross sharding swaps (exchanges). There is no global cryptocurrency. The exchange of cryptocurrencies from different shards is possible only when smart contracts are created in the form of decentralized exchanges, in which the exchange rate depends on supply and demand.

To circulate a single cryptocurrency, for example EMPERA, each shard can implement payment channels in the form of smart contracts, which change one EMPERA token from one shard to another EMPERA token in another shard (the token ID depends on the smart contact itself and on what it considers EMPERA). The overall balance of coins and tokens inside the shard remains constant.

This approach provides financial security for shard users. If we assume that one shard is compromised, then the rest of the shards will not be harmed.

### Node

Node is the main unit of the network. All nodes are equal. They combine with each other according to certain rules and form a so-called relay network, which allows optimal information exchange. Nodes are not directly connected to each other, each node has a limited number of connections - it is proportional to the logarithm of the number of nodes in the whole network.

Functions of a node as a network participant:

1. Validation and exchange of shard (blockchain) data. According to the standard, a node can only validate one shard.
2. Validation and exchange of cross-sharding transaction buffers data (maximum number is limited)

### Horizontal scaling in merged mining

By a large number of shards (for example, thousands and hundreds of thousands), validation of all shards by one ordinary miner is almost impossible - one node can validate only one shard, and the number of nodes for an ordinary miner is limited.

Horizontal scaling provides means to increase these capabilities as follows:

**The first way** is to create a cluster from the miner’s own nodes. For this, in the settings of each node, the belonging to one cluster is specified by entering a common secret password. In this option, the number of supported shards is limited by the number of the miner's own nodes.

**The second way** is to include an external cluster of nodes that the miner trusts (for example, nodes of his/her friend) into the tree of trusted clusters. This is achieved by adding the public key of the external cluster and specifying the level of the trust hierarchy (whether its child nodes can be trusted, in turn). With this option, the number of supported shards can be many times greater than the number of the miner's own nodes. Moreover, it is possible to specify the degree. The miner can combine both options to achieve maximum reliability and profit.

*User strategy version:*

1. Add his/her own nodes (5 pieces, one shard)

2. Add a cluster of friends' nodes (2 clusters with one hierarchy level, i.e. only friends' nodes)
3. Add a cluster of 100 shards of a well-known company such as VISA/MasterCard

### Shard miner's motivation

The miner is motivated to receive rewards in each shard, thus he is incentivized to validate and include at least one shard in the header.

In order to maximize profits, the miner should strive to validate as many shards as possible, as rewards are received independently in each shard. Since a node can validate only one shard, the miner can launch several nodes with settings for validating different shards, and in order to include information on several shards in the block, he can combine his nodes into a trusted cluster.

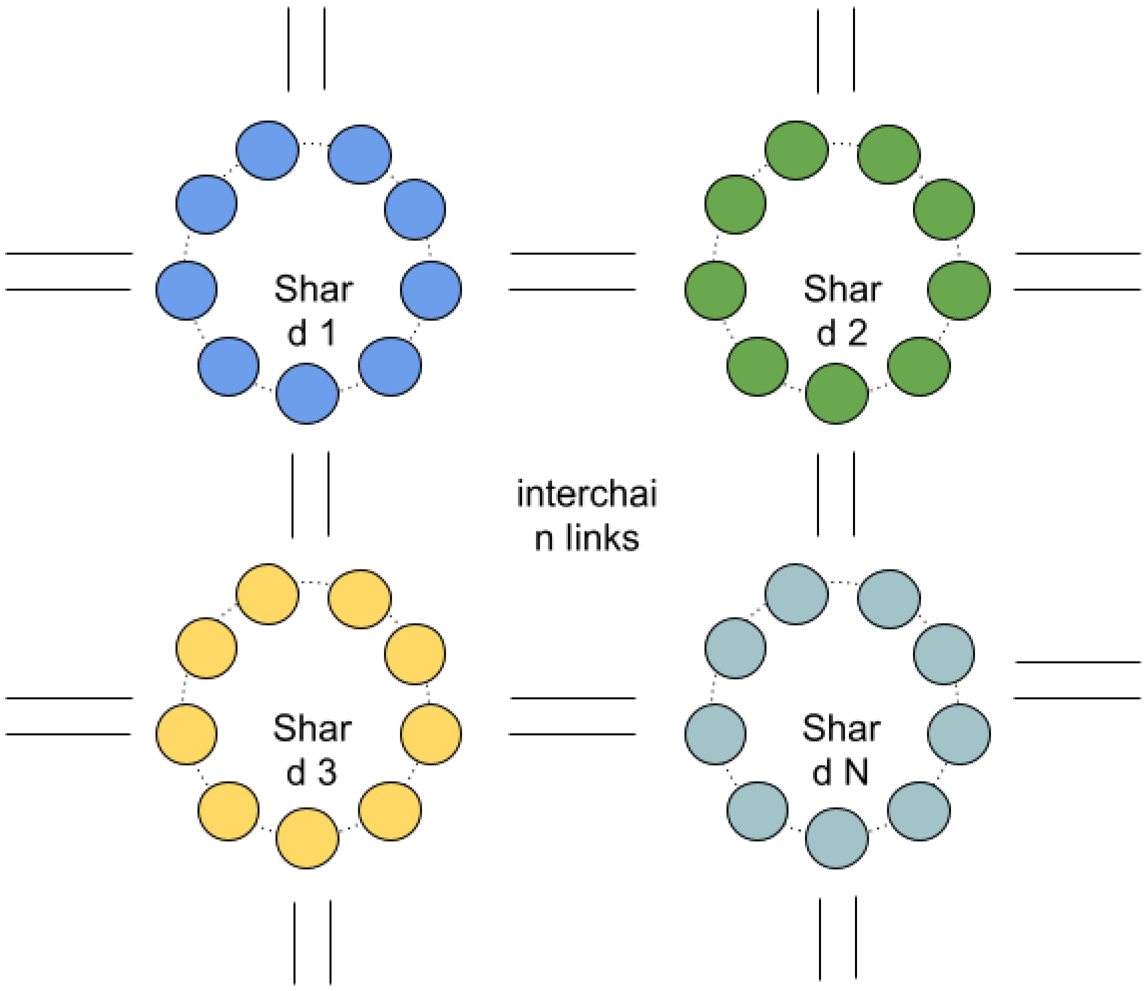
The best strategy will be to search clusters that can be trusted to include them in his mining list. In an ideal scenario, he will strive for 100% coverage of the network shards.

## Messaging pattern

Smart contracts can exchange messages with each other, as well as between different shards.

When creating a smart contract, the height in blocks is set, which determines the finality of the message transfer from one shard to another. When this value is reached, the message is sent for further processing to the smart contract. The minimum height is the actual value of security, if the smart contract is a channel for the transfer of values or other important information. The larger it is, the more secure the channel is, but on the other hand, the longer it takes to wait for the operation to be completed.

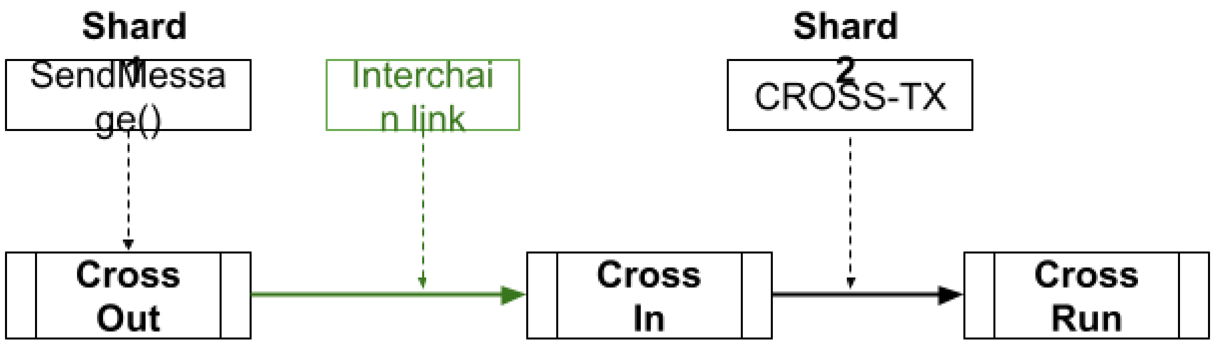
The process of transferring messages from one shard to another is carried out by those miners who simultaneously mine these two shards. If there are no such miners, the message will be postponed until they appear.



Cross-sharding transactions (messages)

If we have a node that mines several shards at the same time, then when creating a block, it can add special transactions containing information about the presence of messages intended for our blockchain in other shards. Such transactions do not independently participate in the exchange between nodes before the block is created. They can be transferred to other nodes only inseparably from the block.

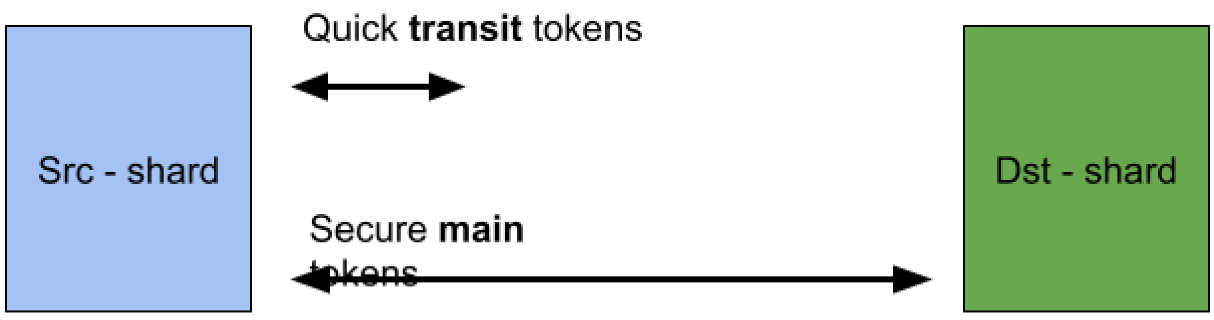
Miners vote for a particular composition of messages in blocks by creating virtual chains through special voting transactions. The more blocks a chain contains, the higher the authority of the message is. When a certain threshold, set initially, is reached, the message is committed in the receiver shard. In fact, miners act as oracles - witnesses to the presence of any information in the source shard.



## Channel security

The security of the channels can be determined through the cost of a 51% attack on the time it takes to successfully transfer a message to the receiver shard. The faster the channel, the cheaper the attack, or in other words, the greater the risk. By managing risks, we can build a scheme that delivers both high security and high performance. To achieve this, we will create two channels for the transfer of values:

1) Main channel - slow but reliable. The commit time is always indicated big enough, for example, 1 million blocks. This channel has its own token in the receiver shard, which equates the source shard's coin.
2) The fast channel is used to quickly transfer a small size of value (compared to the cost of the attack). For example, committing an operation after 100 blocks. The channel also has its own token, which we will call **transit** - it is needed only for short-term use.



The scheme functions as follows:

1. The user sends the amount of values via the fast channel (the sum of values is always less than the cost of the attack)
2. Having received them in the receiver shard in the form of transit tokens, it exchanges them for main tokens through the embedded DEX. Main tokens are transferred via a slow channel, so they are safe.
3. The arbitration specialist is responsible for ensuring the liquidity of transit tokens. It handles the transfer of transit tokens back to the source shard, and the main tokens -

to the receiver shard. To motivate him to place orders on the DEX, the rate will not be 1 to 1, but with a certain margin with the pledged cost of freezing capital and its profit. The particular rate will regulate the market.

### Sending coins from the wallet between shards

It is possible to send coins directly from the wallet interface between the accounts of neighboring shards, for this, the recipient's account address must contain the name of the shard and the account number, separated by a colon: *“SHARD: AccNumber”*
In this case, the name of the shard is resolved into the account number with the smart contract, which is a gateway (as an option, this list of matches can be threaded into the wallet interface itself).

The reason for using an intermediate smart contract is to isolate responsibility. If one shard is compromised, the values of the other shards will not be affected. By cross-transfers, the coins remain on the smart contract account within the blockchain; the other blockchain receives only a kind of note, on the basis of which the token traffic is performed.

This model allows creation of shards for a very wide range of purposes, for example, temporary ones, but with high performance.
Example: *shard1* is created, tokens are sold for it, users start working with dApps inside it, performing 1000 tps. After a while, when the size of the base exceeds a reasonable limit, for example 1000 GB, *shard2* is created, tokens are changed to *shard1* tokens, everyone goes to work in *shard2*, and miners disable support for *shard1*.

For details:
https://docs.google.com/document/d/1eglQs3uRiHj7R_mTciF3rKtHRytLmmsO-ordGgzVv2M/edit?usp=sharing

## Conclusions

Unlimited scalability can be achieved by using optional trust when validating shards. Unlike other blockchains, such a trust system lacks problems with decentralization as it is disabled by default. If a miner establishes a trust group and makes a mistake, the rest of the network participants reject the error, since the miners’ trust groups are different.