Blockchain implementation approach with custom consensus mechanism, flexible block structure, and hybrid transaction mechanism v.1.09

Alexander Andreev (alexandr@silentnotary.com), Maxim Breus (max @silentnotary.com), Alexey Petrov (aleksey@silentnotary.com)

Legal Disclaimer: The information shared in this whitepaper is not all-encompassing or comprehensive and does not in any way intend to provide legal advice and should not be taken as an offer to buy any tokens. The UBIX Network's primary purpose of publishing this whitepaper is to provide prospective token holders with important and relevant information so they can thoroughly analyze the project, make well informed decisions, and to receive feedback or comments.

If at some point the UBIX Network decides to offer tokens for sale, then a separate document containing detailed disclosures and risk factors will be created for such a matter. The separate document may include an updated version of this whitepaper, which may be significantly different from the current one.

Abstract

This whitepaper describes an implementation approach for a network consisting of various blockchains with different consensus algorithms (including public and private types), also known as the net of networks or «blockchain of blockchains». The UBIX Network protocol allows users to create new blockchains while the network remains peer-to-peer and has no gateways. Adding blockchains does not require payment from users. It's built in such a way that improves overall network stability and does not impose significant restrictions on its performance. Moreso, user nodes are involved in the transfer of transactions, storing the mempool and storing all network data. This contribution to the stability of the network makes it possible to waive the UBIX Network usage fees for the users. This approach is substantially different from the one used by other platforms. Another key implementation feature is a configurable consensus mechanism that enables the integration of various blockchains into a single peer-to-peer network. The platform operates as a fully functional production-level solution.

Introduction

With the recent advancement of blockchain technology, there has been a problem with blockchain interoperability, which is – to some extent – similar to the one that was solved earlier for LAN and WAN integration [1]. In recent years, while many were trying to solve this problem, a new class of projects emerged, e.g., Polkadot or Cosmos [2]. In addition to the lack of commonly accepted standards and technological challenges, there is also a major legal obstacle. It is the so-called problem of "legal relations chain break" for public blockchains. The problem tends to result in the widening adoption of private blockchains in the corporate sector, which further increases the variety of protocols being used and integration complexity. Mention should also be made to the challenges of the further end-user's services development and their financing, as well as UBIX Network protocol long-term development financing system challenges. Further upcoming challenges should also be mentioned:

- The deployment of upcoming end-user services and their financing
- The long-term development and financing of the UBIX Network protocol

1. Network

The problem statement for network build-out was described as follows:

- Build a peer-to-peer network that supports the creation of other networks (user blockchains) within itself;
- The blockchains being added must maintain connection, without any gateways; and
- There should be an option to change the consensus type.

The network consists of nodes, and each node accepts the rules of one or more integrated blockchains (hereinafter referred to as "consiliums"), and thus, possesses the right to process transactions for these consiliums. Since one node can belong to various consiliums, the number of working blockchains on the platform can significantly exceed the number of physical nodes. For example, a set of five nodes might have thirty, or even fifty consiliums.

The consilium is activated once the specific conditions determined by the consilium rules are met. After the activation, the consilium processes the transactions accumulated in the mempool that have a corresponding label (i.e. are designated to be processed by this consilium). The processed transactions are recorded in blocks, which are ordered and directed according to rules described in section 1.3, forming a directed acyclic graph (DAG) that consists of such blocks.

Nodes in a network can operate in one of two modes: Regular and witness. In instances where the witness nodes aren't present in the network, it works in read-only mode (i.e. regular nodes accept blocks and process them, but the network can't create blocks). In the read-only mode, no new blocks can be created and recorded. All nodes

have the same local state base blockstate and chainstate. The witness nodes are grouped into consiliums and are responsible for block creation (transactions processing). More details can be found in section 1.2.

New node loading starts with the known peers' request from seed nodes in the DNS record (it is a standard bootstrap and is the same as the one used in Bitcoin). Next, the blocks download starts from the available network nodes. Similar to Bitcoin or Litecoin, the network is implemented using a UTXO-like model, where the movement of assets is recorded in the form of a directed acyclic graph ("DAG") between addresses. Thus, the inputs/outputs form their underlying DAG. The blocks form a DAG as they are created by multiple consiliums. Google protocol buffer is used for the data serialization to optimize the block size and adjust the structure to store additional fields.

Transactions

In Ethereum, transactions have no input section, meanwhile in Bitcoin they are UTXO (i.e. stateless). The UBIX Network protocol uses a hybrid of these two approaches. More specifically, the inputs are standard for UTXO, and the outputs could be either standard for UTXO, or non-standard where it is possible to specify a contract call execution code (refer to section 1.5 for more details). In addition, for transactions with contracts, the transaction specifies the ID of the consilium, where the system looks for this contract. Once found, the consilium forms it and creates a block from it.

Blocks

The block consists of a header and serialized transactions. The header is standard (i.e. the same as the one used in other blockchain implementations). For transactions inside the block, the Merkle Tree [3] is calculated and its root is written into the header. Another difference from other blockchain implementations is the consilium ID field. In other blockchains, the block contains a link to the hash of the parent block, meanwhile in the UBIX Network protocol, there may be several links like this.

Let's take a closer look at the concept of height in DAG. Let's assume there is a chain and a parent block (i.e. the genesis), and the network just started. There is a single witness in the network. The genesis has a height equal to 0, the first block has a height equal to 1, the second block has a height equal to 2, and so on. Then, another consilium is added to the blockchain, it creates a block, and attaches it to the third block and the genesis. Does this lead to uncertainty as to what block should be used to calculate the height parameter (the genesis or third block)? In this case, the logic implemented in the UBIX Network protocol chooses the longest path to the genesis (i.e. the block in question will have a height of 4, even though it has two connections to the genesis and the third block). This check is performed for each block and the height of the block will be the maximum height of the parents plus one.

1.1 General Architecture

The general architecture of the UBIX Network platform is shown in the below *Figure 1.* The top of the diagram shows exactly how the user interacts with the platform through the super app. In particular, the diagram shows one common interface, and below that there are microservices that communicate with each other through the internal bus, and then with the network itself. Next, the transactions ("TX") are shown belonging to different blockchains – the large circles – with the nodes being represented by small circles. The intersection of blockchains are shown, due to the fact that the nodes can belong to more than one consilium. There's a node shown at the intersection to illustrate the fact that it can belong to multiple consiliums (2 in this case). The squares correspond to the blocks that make up the DAG.

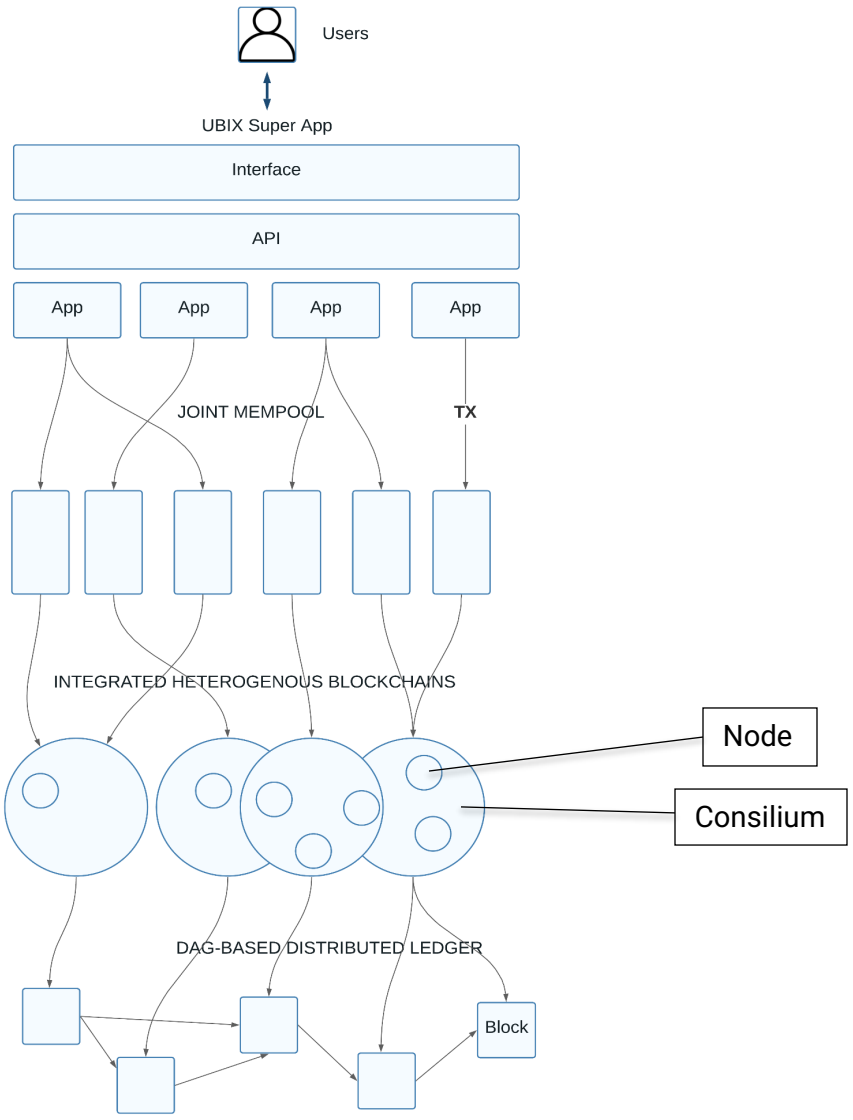

Figure 1. The general architecture of the UBIX Network

1.2 Blockchains

Every integrated blockchain ("consilium") is a composition of the nodes subset that accepted the same consilium rules. One node can accept several sets of rules (i.e. it can belong to more than one consilium). Currently, PoS & RoundRobin consiliums are implemented (i.e. the block's validity is confirmed by the presence of the required number of signatures in the block). There is a plan to expand the block header to accommodate other types of valid blocks (E.g., PoW, PoH [4], etc.) in the future.

The choice of who offers the block (the so-called proposer) depends on the consilium type. It could be revolving (i.e. one at a time) or PoS (i.e. depending on the amount blocked in the consilium contract and the linear congruential generator [5]).

The number of signatures sufficient to accept a block for the RR consilium is the quorum parameter of the consilium. If the number of signatures isn't sufficient, a qualified majority approach is used (⅔ of the total number of participants in the consilium). For the PoS, a simple majority of voting stacks is enough to accept a block. The standard Byzantine fault [6] method is used for voting.

The consilium can be open, or in other words, anyone can join it by depositing the amount, as per the rules (configuration). There are also closed consiliums that can be joined by the consilium invitation.

Users can leave the consilium only after a certain number of blocks have been created, the number is a configurable parameter. Aside from the type of consilium, the consilium rules (configuration) also include various fees. There's a plan to implement a feature that allows for a vote on fee and rule changes, as well as the members' exclusion.

Rules are a set of conditions that determine:

- The condition for the nodes to accept the rules (it defines, inter alia, the private/public type of the blockchain). These rules define a list of legitimate nodes.
- The blockchain activation conditions (periodically, based on the number of transactions accumulated in the mempool).
- Transaction processing conditions (e.g. the presence of a quorum).
- The transaction processing order (consensus conditions). These conditions determine the type of consensus – PoW, PoS, etc. The consensus algorithm is split into two parts. The former is the service one, where the DAG is used, and the latter is available for users to customize at the service level.
- The evolution conditions (the amendment of the rules).

Since the system is designed to handle a big number of consiliums and their contracts, the collision prevention mechanism was developed. Namely, the councils are assigned sequence numbers, and the collision prevention business logic will be implemented. The collision prevention logic will be activated if this happens: The same smart contract is called by more than one consilium. In this case, the collision prevention logic will decide the order for processing the smart contracts. For example, the value of parameter a is 3 for the consilium A, and the value of parameter a is 5 for the consilium B. What value is stored into the variable a is very important for the system. In the case of a single consilium, the process is fully determined, and the order of how blocks are processed is quite straightforward. For multiple consiliums, the collisions are resolved by the collision prevention logic. The code is written in such a way that the smart contract is executed by the consilium it belongs to. A contract execution from another consilium is prohibited. There are plans to further improve the code for smart contract execution so the smart contract cross-consilium execution is supported without collisions.

1.3. The DAG Build Out Principle (Best Parents Selection Rules)

The Merkle Tree mechanism is used for block integrity validation. This tree is built from blocks, according to the idea of block interconnection. The block must refer all blocks that have ancestors containing the outputs spent for the block in question. In this case, all pairs of inputs and outputs form a partially ordered set, built from a chain of blocks lying on the same branch of the Merkle Tree.

The parent choice:

- The maximum number of 'non-conflicting vertices' (i.e. the terminal blocks of branches that do not contain double spends). In case of a conflict, the branch with the most blocks containing different consiliums is selected (i.e. supported by the majority).
- Blocks with a height (the greatest chain length from a block to the genesis block) greater than the height of those blocks where the transaction inputs of this block appeared (it is impossible to spend the UTXO at a lower level than they appeared).

1.4. Finality of the Transactions

The finality of block transactions contains the following main features:

- Once more than half of the consiliums include a block into their chains, the block, and the blocks below, become stable (their content is executed and finalized). After that, one can be sure that the transaction from this block will not be returned to mempool.
- On the contrary, if one of the two conflicting branches (stored independently of one another until one of them wins) becomes stable, then the other will be completely discarded and transactions (except for the conflicting ones) will be returned to the mempool.
- In case of the node restart, the unstable blocks will be replayed and will wait again for the stabilization.

1.5. Smart Contracts

In the UBIX Network protocol, each blockchain supports smart contract deployment. A smart contract invocation and execution is allowed only in the blockchain where it is deployed. Nevertheless, there is a plan to upgrade smart contracts functionality, which would allow one smart contract to be invoked by another one located in a different blockchain. Contracts are written in JavaScript and their execution is performed by an interpreter. This approach differs from the one used by Ethereum, where smart contracts are compiled into the byte code and then executed by the EVM. From the payment for smart contracts execution standpoint, the smart contract execution cost is a parameter of the consilium (i.e. it can be configured both in terms of value and currency being used). Take any smart contract for example, where the execution cost could be 10,000 coins. This approach differs from the one used by Ethereum, where the cost of execution is proportional to the number of EVM instructions executed.

At present, users can't deploy smart contracts on their own. There's a plan to remove the preliminary smart contract moderation step. As a prerequisite to this, there must be a protection mechanism implemented to avoid perpetual loops, and a payment system to charge for every instruction execution.

1.6. Tokens

At the outset, the TRC10 token standard was chosen as a point of reference. There's a plan to implement other token standards, including NFT. In the current implementation, a token factory was built. Within the token factory, it is now possible to invoke a method of dedicated smart contract that belongs to the special consilium, and the platform will generate the number of tokens requested. The method's mandatory parameters define some key details for the tokens being issued (e.g. total supply).

If necessary, it is possible to deploy one's own smart contract, which will create tokens. At the moment, processing token transactions is only possible in those blockchains where they were issued.

2. Interface – Super Application

Having analyzed the success stories of super applications (WeChat, Yandex Go, etc.), the decision was made to implement a similar approach for the UBIX Network platform. This allowed a single-entry point for all services, as well as a means to reduce the technological barrier for end-users. The unified super application is integrated with all services via an internal API. The services use NATS for information sharing.

In terms of implementation, the UBIX Network ecosystem consists of integrated microservices (this includes user services) and a monolith (for the blockchain). The microservice architecture approach simplifies the development and new services roll-out. This enables the re-use of existing services or some of their methods (like

a wallet or payment gateway), thus reducing complexity, time-to-market, and errors. Ubikiri and SilentNotary services are built using the microservice architecture.

The unified authorization service called 'Identity' deserves a special mention. Other services access Identity via a non-public API as shown in *Figure 2*.

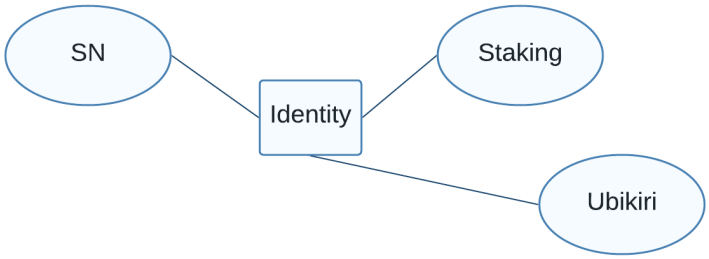

Figure 2. Interaction of the Identity service with other internal UBIX Network services

The blockchain is based on the implementation of the RPC protocol (see *Figure 3*) that provides special methods, such as 'get the last block' or 'send a transaction'.

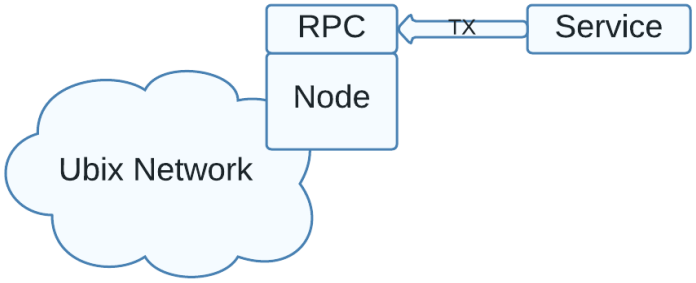

Figure 3. Interaction of services with nodes via RPC

Some services have an open API. Like Ethereum and Bitcoin, UBIX Network nodes use Level DB to store data in the key-value format. The authorization service uses PostgreSQL to store encrypted data.

Depending on the microservice and data types, the UBIX Network protocol uses different types of data storage. The blockchain is stored locally for each node.

3. Legal solutions

A key feature of the UBIX Network protocol is the existence of the consilium ID field in a transaction. The field defines what blockchain processes the transaction. An EDS is required for a user to set the consilium ID value and sign the transaction. From a legal standpoint, a signed transaction – where the consilium ID field is populated – means that the user acceded to the public contract (i.e. the user became one of its parties). Moreover, the consent of the user is explicitly expressed and confirmed by the EDS. The rules of a consilium consist of two parts: Technical and legal, with the public contract being added to the latter. The legal part is very important for people and organizations' legal relations.

For example, the consilium owners attached the following public contract to it: "A user accepts the contract by sending a transaction on this consilium." This example could be made more complex if contracts that bind parties are included. Then, those transactions will have an economic value and create legal relations. This way, people accrue rights and responsibilities once transactions like the ones above are sent. This was the key reason for introducing consiliums.

Aside from blockchain users who send transactions, some nodes can also accept the rules of a certain consilium and accede to the agreement. This way, a chain of legal relations is built.

Take note on the problem of the legal relations chain gap, as it is well known that there is no obligated party in any public blockchain, so no one must do anything. Thus, the blockchain is regulated by purely economic principles (i.e. the profitability of mining). As long as mining is profitable, the number of miners increases, and the blockchain is stable and growing. On the contrary, once mining becomes unprofitable – as this happened many times before (for instance, refer to the case of Ethereum Classic [7]) – the number of miners significantly decreases and the blockchain becomes vulnerable to an 'attacker' with high computing power. This 'attacker' could hijack the network, rewrite transactions, and tweak the data in the blockchain (e.g. amending the vote totals of the company's shareholders meeting). Thus, the lack of an obligated party is a rather significant problem for ordinary businesses. The proposed solution to this problem is for the users to create a new private blockchain (consilium) with the nodes residing in the UBIX Network. In this case, from the user data (business part) standpoint, the blockchain (consilium) is completely private. Meanwhile, from the UBIX Network protocol perspective, the new blockchain (consilium) is an integral part for the complete UBIX Network. Since the UBIX Network protocol supports any consensus mechanism, the approach could easily be expanded to existing blockchains. They can be connected to the network similarly, as demonstrated by maintaining privacy at the user data level and moving integration to the blockchain level (migration of nodes to the UBIX Network).

Another challenge companies often face is the requirement to pay public blockchain fees using cryptocurrencies. In many jurisdictions, the legal side of such payments is convoluted, and it is very hard, if not impossible, to legally

pay the fees with cryptocurrencies. There may also be other obstacles for legal entities to purchase cryptocurrencies and then pay with them for goods and services. That is often why companies are driven to use private blockchains. In the proposed approach, creators of the new consilium pay for the setup of new nodes like deployment costs, integration, and maintenance. In other words, this is a variation of the BYOD concept applied to nodes. The UBIX Network does not require users to pay the fees in cryptocurrencies. In other words, the proposed approach eliminates the need to use cryptocurrencies to cover the costs of the blockchain.

Network users participate in its development by the simple means of using it. All nodes in the network have a complete replica of all data from the entire network. Therefore, a private blockchain (consilium) that is integrated into the UBIX Network processes transactions, and as a result, maintains the stability of the UBIX Network. Its nodes are involved in the transfer of transactions, mempool, and all the network data storage. Accordingly, when user nodes sign a block and put it in the shared storage, they make it impossible to change previous blocks and transactions because they are signed. In the current implementation, the finality is not based on probability, rather it is based on facts. If 50% of the consiliums include a block in the graph, then it becomes final.

In this case, the integration will be done only at the technology level and no user business data will be stored in the UBIX Network. Currently, the functionality for transaction storage split by consiliums (sharding) is in the planning stage and is being prepared for development. Two ways of implementation are considered: (i) Coin issue (coins are used for network protection and fuel mechanism) and (ii) A completely free option (no coin).

4. The Financing System of the Ecosystem Development

A hybrid venture capital and ICO approach is used to finance the development of the UBIX Network and its ecosystem (i.e. each service is funded in small rounds, allocating as much resources as required to achieve the round goals). In parallel, each service issues its tokens. As a rule, these tokens are shares of the fees collected by the service. This approach (sprints) is time-tested and well-proven in the venture capital industry. Registration of a transaction through a token sale represents a participation opportunity by regular users and circulation in the secondary market.

There are plans to build an external API and a community of software developers using the crowdsourcing and crowdfunding model. It should be noted that our team has already had a positive experience of working with this model. In particular, the Ubistake staking service was developed in this approach. After several months of stable service operations, the distributed fee amount came out to 10 times the development costs.

5. The Long-Term Network Development

The long-term development of the network is very important. 10% of all transaction fees are credited to a special long-term development fund, and then the fund's liquidity is used to finance UBIX Improvement Proposals

(UIP). The difference from Ethereum Improvement Proposals (EIP) is that the UIP funding source is specified in advance, and financing comes from within the network.

The UIP procedure is in the preparation stage. So far, 3 development proposals have been prepared. One of them, for example, covers the implementation of the sharding mechanism. The current version of the UIP proposal process consists of the following steps: Initiators prepare proposals, a vote is held, and the winning proposal is selected. Then a team is built to implement the project. In cases of success, the team shares the allocated development reward for this UIP amongst themselves.

6. The Main Ecosystem Elements

Figure 4 shows that the UBIX Network is used as a foundation for all services on the platform:

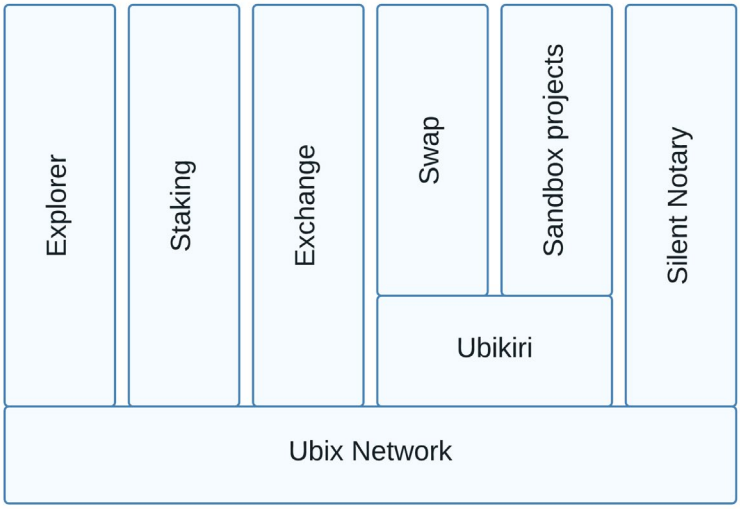

Figure 4. Location of some services on the UBIX Network platform

Explorer is a service that allows users to see the transactions, block interconnections, and the way the transaction tree is built. It is a mandatory part of any blockchain. The graph (DAG) could be seen in this too. It is currently a part of Ubikiri.

Ubikiri super application is a UI that integrates all UBIX Network protocol services that are available to the user after a single registration.

UBIX Exchange (Token Name: UBXE) – UBIX Exchange is used for internal project tokens so that they can be immediately traded in an automatic fashion (i.e. no separate listing procedure is required).

UBIX Depository (Token Name: UBCDR) – Depository, also known as Swap. The service provides token packaging/unpacking functionality. Users can package UBIX Network tokens or coins into a different format (e.g. ERC20). It is a system of economic connection with other blockchains.

Ubistake (Token Name: UBXM) – The service allows users to receive passive income through deposits. First, users deposit coins to Ubistake. Next, this money is used to purchase a witness node (i.e. one that mines blocks). The greater the volume of collected deposits, the more likely this particular node creates the next block, and therefore, receives a reward. Furthermore, the reward is divided among the depositors, thus demonstrating how they receive passive income.

SilentNotary (Token Name: UBSN) – The application allows users to notarize any files. For each file uploaded to the system, a hash is calculated and then recorded into the blockchain. The user is given a link to a certificate where it is possible to check the file for authenticity.

Launchpad (Sandbox) is a service designed to be used within the UBIX Network Community for start-ups. The service supports ICOs by issuing shares (tokens).

7. UBX Coin and Tokens

One of the UBIX Network special consiliums is a monetary one that processes transactions in the native UBX currency. This cryptocurrency is used to pay fees and is a means of economic stabilization of the entire network. The monetary consilium has a PoS consensus type and it contains master nodes that create blocks and receive rewards for this.

Upon the network launch, as a one-off, a certain number of coins were issued in this consilium. These coins were distributed in a certain way among several funds:

- The Main Fund (most of the coins are located in this fund and it is also used for payouts to users who have a stack in excess of 1 million).
- The Development Fund (10% of fees are accumulated in this fund).
- The Team Fund.
- The Witnesses Support Fund (if the number of transactions is relatively small, the transactions are generated and the monetary consilium witnesses collect the fee).
- The Partnership and Stability Fund (also known as the “icebreaker”) is used to fund some critical network upgrades.

A reserve fund was created as well and was supposed to provide liquidity in case it's needed. The decision was made to distribute the reserves as per a certain schedule amongst all coin holders. Each day, there are accruals of 0.1% that are distributed among all coin holders. This number is expected to gradually decrease and hit 0 in about 20 years.

The development system of the UBIX Network platform (i.e. the one for internal services) is different from the one used for user services development. The fact that the platform is an infrastructure makes it problematic to finance it through rounds. For more details see section 5 on the Long-term network development.

References

[1] White Paper: Routers/WAN access devices white paper

https://www.computerweekly.com/feature/White-Paper-Routers-WAN-access-devices-white-paper

[2] A Survey on Blockchain Interoperability: Past, Present, and Future Trends, section 5.2, p. 77

https://arxiv.org/pdf/2005.14282.pdf

[3] Merkle, R. (1998), A digital signature based on a conventional encryption function

https://people.eecs.berkeley.edu/~raluca/cs261-f15/readings/merkle.pdf

[4] Solana: A new architecture for a high performance blockchain v0.8.13

https://solana.com/solana-whitepaper.pdf

[5] Linear congruential generator

Gentle, James E., (2003). Random Number Generation and Monte Carlo Methods, 2nd edition, Springer, ISBN 0-387-00178-6.

[6] Byzantine fault

Kirrmann, Hubert (n.d.). "Fault Tolerant Computing in Industrial Automation" (PDF). Switzerland: ABB Research Center. p. 94.

[7] Deep Chain Reorganization Detected on Ethereum Classic (ETC)

https://blog.coinbase.com/ethereum-classic-etc-is-currently-being-51-attacked-33be13ce32de