Nimses Blockchain: System of Electronic Assets

Nimses Inc

nimses.com

Abstract. An economic system that provides participants with an unconditional, basic income for each minute of their lives requires both reliance on cryptography and provision of acceptable throughput. Typical solutions relying either on trust or cryptography do not meet these requirements. The hybrid system of Nimses does meet the requirements for such a global solution. A centralized server timestamps transactions, linking them into a chain, where hashes of the blocks in the chain are announced in the generally recognized, immutable ledger. The transaction model of the above-mentioned system allows operations to be processed simultaneously and independently, where throughput is limited only by the speed of reaching an internal consensus about transaction order.

1. INTRODUCTION

Modern centralized systems of electronic assets, which solve the double-spending problem by means of a trusted third party, basically rely on the trust of their users. As far as trust is concerned, such systems can not compete with ones which stem from cryptography and, as a rule, have a peer-to-peer structure [1]. Usually, there is a mechanism that incentivizes users to maintain the viability of the system and requires huge computational resources. As a result, elements such as a trusted third party and central emitter become obsolete in such modern systems. Another important disadvantage of this kind of systems is the low transaction processing speed, which makes them unsuitable for global use.

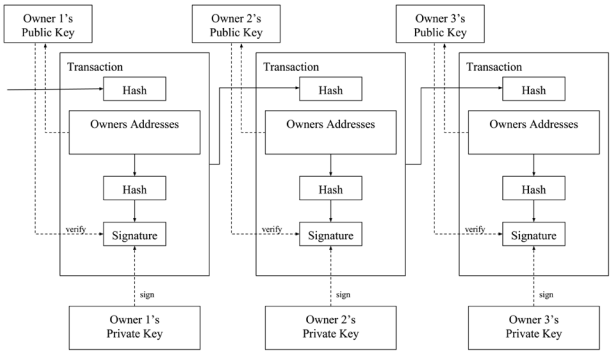
A new system reliant on cryptography, but not on trust, is needed. Such a system must provide sufficient throughput for large-scale everyday use.

Nimses proposes a solution for the above problem. The solution is based on a centralized timestamp server and submission of an electronic asset as a chain of digital signatures. The server acts as a trusted third-party to solve the problem of double-spending, verifying the chronological order of transactions. Trust is guaranteed due to the anchoring of the operation history.

2. TRANSACTIONS

An electronic asset is defined as a public chain of digital signatures and an account as an ordered pair (a unique identifier hereinafter referred to as an address and an asset).

Initiating a regular transfer, the account owner creates and signs a transaction; the hash[1] of the previous transaction is attached to the following one. This information is attached to the asset. A payee can verify each digital signature to ensure the correctness of the entire chain.



[1] Here and below, a hash stands for the result of a one-way function.

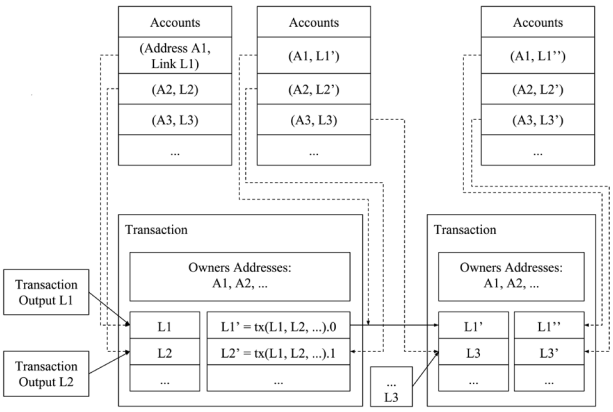
There is a problem when a payee can't verify how many times the current asset has been spent by a previous owner. There are two common solutions to this problem. The first is to shift these concerns to a trusted central authority, which makes the entire monetary system dependant on itself. The second is to create a peer-to-peer network in which participants need to publicly announce transactions, as well as be able to agree on the only order of the transactions. This causes a low network throughput and huge computational outlays.

To address the above shortcomings of both approaches, Nimses use a hybrid solution based on a centralized timestamp server that provides the correctness and chronological order of transactions. This is similar to a solution with a trusted central authority, but has a mechanism of regularly anchoring the entire transaction history in a public, immutable ledger.

3. TIMESTAMP SERVER

The system rules are defined as a set of generally accepted statements, established within it. For example transaction commissions, tax levies, etc.

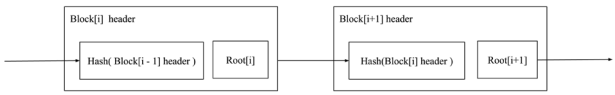
Having received a transaction, the server adds it to the processing queue. The processing is organized as follows: first, the server checks whether a transaction corresponds to the correctness of the current system state. If the check is positive, the server attaches the latest links of assets named "input links" and some result named "output links", generated according to the system rules, to the transaction. The output links are connected with the input ones by means of the transaction, which then become the new input links in the following transactions. As a result, an accurate, chronological order of the procedure is guaranteed. Following the correct processing, the server attaches a timestamp to the transaction and publicly announces it. The timestamp indicates that specific data existed at that moment, and thus entered the chain.



The problem, of course, is that providing a genuine version of chain history by server relies on trust. We need a way for the user to be assured that the previous links of the assets haven't been deleted. For that, the server regularly anchors the state of the chain into a generally recognized immutable ledger. This is how the chain is built, and the next link strengthens all the previous ones.

4. BLOCKS

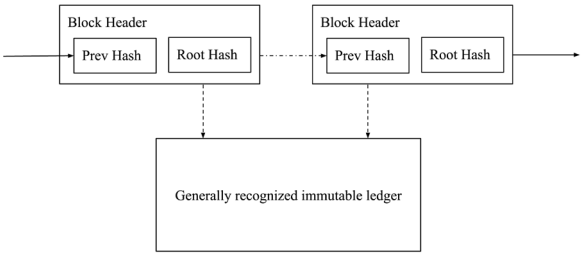
To simplify the anchoring of the data history, transactions are organized into blocks, which consist of a Merkle tree from the transaction hashes and the block header. These trees are built from an ordered selection of transactions in the chain for a predefined period. A block header is defined as an ordered pair (the hash of the previous block header and Merkle tree root of the current one). Therefore, the blocks are linked in an ordered chain, wherein an attempt to change the information of any block requires the recomputation of all the subsequent ones. Thus, it's enough to anchor just the header hash instead of the entire chain of transactions.



The usage of blockchain simplifies both the anchoring of the data history and the verification of a transaction's existence. To check whether a transaction is included into a block, it is not necessary for a user to download the entire transaction history, which is unacceptable due to its size and growth speed. A user can request a link to the block, that contains the transaction and Merkle path leading to it, to make sure that the corresponding transaction belongs to the block, and all subsequent blocks are accepted and verified by the server.

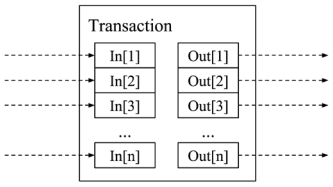
5. ANCHORING TIMESTAMPS

The anchoring of a transaction's history is established through a public announcement of data in an immutable ledger, as in a newspaper or Usenet-post, that guarantees the unambiguous origin of the data. In 2017, it is reasonable to use public peer-to-peer decentralized systems as a ledger for anchoring, such as Bitcoin and Etherium, as these have a sufficient level of reliability.



6. COMBINING AND SPLITTING VALUE

To provide transfers between more than two participants, transaction input and output links represent themselves, not as pairs, but as ordered sets. For such a transaction to be validated by the system, it is required to be signed by every payment sender.



7. NIMSES IMPLEMENTATION

7.0. Technical Requirements

- Potential users volume: $3x10^9$ users
- Transaction latency: ~ 1 second
- Transaction throughput: up to 10^6 TPS

7.1. Assets

In the current implementation, there is a single asset type corresponding to an account, . Initially, there are two types of assets: **nim** and **infinim**.

Nims are emitted via an internal emission by the system, which is a function of astronomical time. Infinims appear due to the consumption of a certain number of nims according to the system rules.

The asset type is encoded with 32-bits.

7.2. Accounts

A unique account address consists of a 16-byte numerical sequence and a 4-byte asset type.

An account is described as follows: account type, asset type, current balance, number of transactions in the chain of digital signatures, set of associated public keys that verifies the account holders, emission type, and other service data.

Account state is defined as: timestamp, transaction number in the corresponding chain of digital signatures, and the final balance of the last link, which is calculated for this timestamp regarding the previous one, with consideration of the emission and transaction receipt.

Account type can be either a service or a user. User account types fall under nim-token emission.

Accounts are separate entities in the system, representing the current state of user assets. They are created by a special transaction that associates a unique address, initial state, and a set of public keys. Then, the registered address is ready to participate in transactions.

A subset of some transactions is signed, not with the public keys of corresponding accounts, but with special service keys. These service keys are authorized with genesis root keys. For example, a tax institute is allowed to withdraw certain amounts of tax deductions from user accounts.

```
Address  = {16 byte} + {asset_id: 4 byte}

Account  = (Address, Asset, Type, Emission, {Public
Key}, State)

State    = (Time Point, Balance, Nonce, ...)
Account Type = {
    COMMON,
    HUMAN,
    GENESIS,
    TAX,
    REMOVED,
    ...
}
```

7.3. Emission

One nim is emitted for every minute since a user has registered.

In order to determine the number of emitted nims while a transaction is processed, the timestamp server additionally timestamps a transaction being recorded in the chain with an astronomical timestamp and computes output links of the chain, taking nim emission into account.

Infinim emission is a special case, wherein there is a separate account of service type that accumulates nims up to a certain amount, in order to generate an infininim.

7.4. Genesis

The chain begins with an initial state named Genesis. Genesis is defined by a service account and a set of root keys. The Genesis account determines the initial nim emission as the system starts.

7.5. Transactions

A typical transaction consists of three parts: header, body and sender witnesses.

```
Tx = (Header, Body, Witnesses)
```

A transaction header is a tuple of protocol version, transaction type and time window, that shows duration while one can be recorded in the chain.

```
TxTypes  = {
    CREATE_ACCOUNT,
    SPEND,
    TAX_SPEND,
    GENESIS_SPEND,
    REG_KEY,
    REVOKE_KEY,
    ...
}

TxHeader = (Version, Type, Time Window)
```

The account address and number of either sent or received assets are defined as value reference.

```
ValueRef = (Account Address, Value)
```

A transaction body is an ordered pair of sets that contains value references of senders and receivers correspondingly.

```
Tx.Body  = (From: {ValueRef}, To: {ValueRef})
```

Senders' witness data is an ordered set of witness data of every sender. A sender witness data is an ordered pair made up of the sender's signature and hash of the public key that belongs to the signing pair. SHA3-256 is used as a hash function [4] (FIPS-202).

```
Witnesses = ({Witness})

Witness  = (Signature, Public Key Hash)
```

A digital signature that is derived by means of signing both the header and body of a transaction with any key pair from the set which is attached to a sender account is a sender signature. There is ECDSA [2] (NIST.FIPS.186-4) on SECP256R1 [3] (RFC 5480) curve chosen as a digital signature algorithm.

```
Signature = ECDSA(Public Key, Tx.SigId[i], Private Key)
```

Having a transaction added in the chain is supplemented by hashes of the input links and output links. Every link is identified with a 256-bit hash. Their uniqueness is assured by the system and is verified when a transaction is added to the chain. Transaction uniqueness is provided in the same way.

Transactions may be signed either by private user accounts keys or by special service keys.

7.6. Service Keys

The system introduces service keys to provide some necessary functions of Nimses' economic system. Tax withdrawals or penalties for a violation of the rules stand as examples of these functions. These service keys can be used instead of user keys and are registered in the system by means of root key signature. Their secret parts are stored on special devices known as "Hardware Security Modules" in secure networks and can not be extracted from there.

```
Key Types = {
    KEY_USER,
    KEY_ROOT,
    KEY_MASTER,
    KEY_IDENTITY,
    KEY_TAX,
    KEY_FAMILYPAYMENT,
    ...
}
```

Rules of transaction validation are determined by a type of transaction and type of participating accounts.

7.7. Consensus

Participants of peer-to-peer networks must come to an agreement on the only version of the transaction history to prevent the double-spending problem. There are some general algorithms that are used for this approach: proof-of-work, proof-of-authority, proof-of-stake and others, which provide a uniqueness-of-history version. Centralized systems address this shortcoming and can use more efficient methods that prevent double-spending.

Since there is no need in replicating the entire history all over a huge number of participants to come to an agreement, it allows processing transactions in parallel, thus high throughput is provided.

Timestamp server is defined as a private distributed network of separate nodes, which use a globally distributed storage of NewSQL class. This architecture provides linear scalability of both space and transaction throughput. The internal nodes reach consensus by means of the Paxos algorithm.

Current implementation of Nimses takes about one second to process a transaction and is able to provide simplified proof of whether a transaction is included in the chain immediately after the block is written. Therefore, throughput is directly proportional to the speed of reaching the internal consensus.

7.8. Blocks

Blocks amalgamate transactions in ordered sets and serve as tools that provide publicity, immutability, and chronological order.

A block consists of a set of transactions and a block header. A block header contains the hash of the previous block and the roots of three Merkle trees.

```
Block Header = (Version, Height, CommitAt, Gen, Prev Block Hash, Tx Root, Witness Root, Receipt Root, Witness)
```

Having a block formed and written, the block hash is recorded in a third-party trusted ledger. Blocks can be also complemented with signatures of extra validators in the system, that confirm the correctness of the chain and transformations.

7.9. Merkle Tree

The computation of the Merkle tree for transaction identifiers is similar to the algorithm described in RFC6962[5] section 2.1.2. SHA3-256[4] (FIPS-202) is used as a hash function. The last node is duplicated on levels with an odd number of vertices.

---

8. CONCLUSION

The hybrid system of Nimses meets the specified technical requirements. User trust stems from the infeasibility of changing chain history that has been anchored in a third-party ledger. The described custom model of transactions that allows parallel processing, dynamic size, and time period of a block, provides sufficient throughput that meets the requirements for a planetary system of asset exchange.

9. REFERENCES

[1] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf

[2] Federal Information Processing Standards Publication, Digital Signature Standard, July 2013, https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf

[3] S. Turner, D. Brown, K. Yiu, R. Housley, T. Polk, Elliptic Curve Cryptography Subject Public Key Information, RFC 5480, March 2009, https://tools.ietf.org/html/rfc5480

[4] Federal Information Processing Standards Publication, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015, https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf

[5] B. Laurie, A. Langley, E. Kasper, Certificate Transparency, RFC 6962, June 2013, https://tools.ietf.org/html/rfc6962