# U°OS blockchain framework

May 28, 2019

# Contents

**Abstract**

The concept of a cryptographically protected and distributed transaction ledger has demonstrated its efficiency in a series of projects. Decentralized frameworks based on blockchain technology allow communities to build transparent and reliable peer-to-peer systems that implement economic relationships between the users of the network. Over the recent years a series of consensus protocols were created and utilized in existing blockchain systems [4, 13, 14, 16, 17]. In this paper we introduce the U°OS blockchain framework and a novel DPoI (Delegated Proof of Importance) consensus algorithm that takes into consideration the value of the social and economic interactions between the members of the network and motivates users to actively contribute to the network growth. Notably, our DPoI metric can be modified to account for a variety of interactions, that arise in a particular network system. DPoI is a high-performance, resource efficient, network-growth inducing algorithm that rewards network participants for the economy enhancing operations in the system. Delegated Proof of Importance is an upgrade to the existing blockchain solutions, that integrates the concepts of the Delegated Proof of Stake (DPoS) and properties of networks. The system protocol is designed in accordance with business and end-user requirements such as privacy, transparency, and smooth volatility, which is achieved due to adaptive emissions proportionate to both network activity growth and the volume of the network itself, according to Metcalfe's law [1]. Our U°OS blockchain framework with the DPoI consensus metric represents a flexible architecture that can be deployed to set up a number of decentralized blockchain-based applications from social networks to service platforms with the direct growing economic value for all users. In this sense, U°OS is a unique framework that unlike any centralized system can be employed to create highly transparent network-based economies.

# 1 Introduction

An unprecedented increase in interest towards blockchain technologies has been observed in the recent years. At this time these technologies were primarily implemented in the form of distributed payment networks [4, 18, 19, 20] and distributed infrastructure systems [21, 22, 23, 24]. These payment networks are decentralized and enable fast low-cost peer-to-peer financial transactions between the users of the system, while infrastructure systems implement smart contracts and decentralized applications (DApps).

The basis of any blockchain system that also determines its technical characteristics is the network consensus algorithm. Namely, consensus algorithm is the mechanism that allows the network nodes to reach the agreement about the contents of the distributed ledger. In this paper we would like to review existing consensus protocol solutions and introduce a novel U°OS blockchain framework with a corresponding DPoI consensus metric.

## 1.1 Previous work

The problem of distributed consensus for networks with potentially fraudulent participants known as the Byzantine Generals' Problem was stated in 1982, long before the creation of blockchain. [2] Since then an array of different solutions has been developed [3]. However, the first solution that did not rely on a trusted third party, was the Proof-of-Work (PoW) algorithm [4]. Despite its advantages, PoW inherently has a number of shortcomings, namely, security [6], scalability, performance[5], the problem of progressive centralization of the networks around the largest mining pools [8] and, most importantly, the need to use vast volumes of physical resources, such as electricity and computing power to generate blocks. [9].

Computing resources needed for block hashing in the current blockchain frameworks, that implement PoW, are tremendous and far exceed the computing power of the world's greatest supercomputers. Energy use for the block mining is comparable to the power consumption of some countries and it continues to grow [7]. In 2012, in order to mitigate these shortcomings, the PPCoin, currently known as PeerCoin, cryptocurrency became the first to utilize an alternative consensus algorithm, Proof-of-Stake (PoS) [10]. In PoS consensus networks, the probability of creating a new block depends on the volume of tokens in a participant's account. Despite significant reduction in resource utilization, PoS turns out to have several drawbacks and in its current state, according to a number of experts, cannot serve as an adequate replacement to PoW [11], [12]. One of the major weaknesses of PoS is that

it additionally motivates users to concentrate all funds in one place or with one user, which leads to centralization of the network.

The next iteration of PoS was introduced as the Delegated Proof-of-Stake (DPoS) [13]. Here network members are divided into two groups: members, who delegate the authority to create blocks and validate transactions, and validators (block producers). This partition provides better scalability and efficiency. Nevertheless, DPoS still has the problem of motivation for a participant to use their assets actively instead of accumulating them, which has a negative impact on the growth and induces network centralization.

Yet another consensus metric, the Proof-of-Importance consensus algorithm (PoI), was first introduced in the NEM cryptocurrency [14]. PoI incentivizes network participants' activity. The major departure from PoS is that block generation probability and reward distribution depends not only on the volume of a user's deposits, but also on the participant's activity rate and reputation. Thus, the algorithm motivates users to be more active by participating in more transactions and contributing to the network development. Despite all its merits, PoI has some shortcomings in efficiency.

## 1.2 Motivation

As a matter of fact, social and economic interactions are integral components of any network system. Therefore, facilitation of social and economic activities contributes to the network development. We propose here a novel DPoI consensus algorithm that takes into account users' social and financial transactions in order to encourage participation in the network growth and prevent centralization. U°OS framework with DPoI consensus algorithm allows users to build virtually any network economy system on top of the U°OS blockchain and assign flexible financial and social activity scores, that reflect the nature of the socio-economic relationships in that particular system.

DPoI metric provides a mechanism for network growth via emission of tokens for social and economic transactions and also a mechanism of decentralization by accounting for socio-economic activity in the overall importance score. In a summary, DPoI metric supplies three main mechanisms for:

- network development facilitation via emission allocation for social and economic transactions
- decentralization via socio-economic importance score calculation
- deployment of DApps with a wide range of social and economic relationships

# 2 U°OS consensus algorithm

The major goal of U°OS project is to design a consensus algorithm with an individual influence score metric, that facilitates efficient score redistribution, motivates users to participate actively in the network development and prevents centralization. Modern blockchain solutions have problems with scalability, security and efficiency, and to solve those problems, U°OS protocol introduces the DPoI (Delegated Proof of Importance) Consensus Algorithm. This consensus algorithm combines the advantages of DPoS and PoI, and delegates validation rights to a limited number of accounts, based not only on the stake value of the protocol members, but also on the their transactional activity, in order to achieve high levels of efficiency and scalability within the network.

The U°OS consensus algorithm (Delegated Proof-of-Importance, DPoI) is based on the DPoS consensus algorithm [13]. In addition to the individual stake amount, our algorithm also considers incoming financial and social transactions of the user. In the U°OS Protocol participants have the option of delegating the right to validate blocks to a limited number of accounts through voting, using their personal importance scores, analogous to DPoS. Unlike DPoS, however, DPoI importance score formula is calculated from three components, namely, the stake amount, financial transfer activity and social activity. This framework is highly flexible since the network can collectively choose not only the weight of contribution of each term in the final importance score, but also decide on how to calculate the transfer and social activity scores, given the structure of economic and social interactions in the system. In general, the working principle of the U°OS Protocol Consensus Algorithm can be explained as follows.

## 2.1 Importance score calculation

DPoI importance score can be interpreted as an importance rating of an account $i$ in the network. It is calculated using the formula below:

$$r_i = (1 - \omega_a - \omega_s)v_i + \omega_a\pi_i + \omega_s\sigma_i \tag{1}$$

where $v_i$ is the stake volume index, $\pi_i$ is the financial activity index, $\sigma_i$ is the social network activity index, and $\omega_a$ and $\omega_s$ are the weight coefficients, that determine the relative significance of each component of the user activity.

Stake volume index is based on the amount of tokens owned and allocated by the account for the usage of the physical resources (CPU and bandwidth), and represents a balance proportion of the total amount of stake in the sys-

tem. Thus, an account with non-zero stake balance has non-zero importance score.

Social and financial activity indices depend on the transactional activity of the account in the most recent time window period, determined by the U°OS protocol. These indices are calculated using NCDAwareRank algorithm, described in the details in the sections below. We will characterize the main principles of the algorithm, based on the calculation of the financial index $\pi_i$. Later in the separate section we will explain how the algorithm is used to calculate the social activity component $\sigma_i$. NCDAwareRank gives more preference to the accounts that are tightly integrated in the general network, which helps to make the network resistant to the *splitting account attacks** performed by the botnets with a small number of accounts. Another important feature of the algorithm is the utilization of incoming activity only. This is a PageRank-based paradigm, that ensures a user can obtain the score only from the accounts that refer to it. Such a score is a direct representation of the user's utility for the entire network.

Financial activity index $\pi_i$ is calculated only for the accounts with the stake balance exceeding the $A_0$ threshold. This value is determined by the network. When calculating the account financial activity index only transactions with the amount of tokens higher than $T_0$ are taken into account. That value is also determined by the network. Transactions that do not meet these thresholds are not included. Financial activity index depends on transactions, creation time of which lays in some pre-determined time interval. The duration of this interval is $W$ blocks. Contribution of every transaction decreases exponentially.

Noticeably, each network application based on U°OS framework can define their unique set of financial and social transactions that are included in the calculation of the corresponding financial and social index scores. These applications can be deployed as DApps (distributed applications) and may implement a wide range of economic and/or social systems, such as service platforms, knowledge networks, digital content copyright systems, libraries, public records, online markets etc. Unlike centralized versions of such services, U°OS DApps would preserve the important properties of blockchain systems, particularly, decentralization, record immutability, and transparency.

*See Appendix 1 Glossary for definition

### 2.1.1 Financial activity score calculation

The vector of financial indices $\boldsymbol{\pi}^{(j)}$, where $j$ is the indicator of the iteration in the algorithm, is calculated according to the NCDAwareRank, following the recurrent relation:

$$\boldsymbol{\pi}^{(j+1)} = (\alpha \boldsymbol{O} + \beta \boldsymbol{M} + (1 - \alpha - \beta)\boldsymbol{E})\boldsymbol{\pi}^{(j)} \tag{2}$$

Here $\boldsymbol{\pi}^{(j)}$ is a vector of the financial activity score index values. The vector is standardized, i.e. the sum of its elements is 1. $\boldsymbol{O}$ is the outlink matrix, $\boldsymbol{M}$ is the interlevel proximity matrix. See the definitions of the matrices below. Coefficients $\alpha$ and $\beta$ are the weights that determine contributions of the matrices $\boldsymbol{O}$ and $\boldsymbol{M}$. Their sum must be less than 1. $\boldsymbol{E}$ is the teleportation matrix, added to ensure that the series is convergent. This matrix is defined as follows:

$$\boldsymbol{E} = \frac{1}{N}\hat{\boldsymbol{E}}$$

where $N$ is the number of the accounts and $\hat{\boldsymbol{E}}$ is the matrix in which all the elements are equal to 1, the calculation continues until for some $j$ the following condition is fulfilled:

$$\|\boldsymbol{\pi}^{(j+1)} - \boldsymbol{\pi}^{(j)}\| < \delta$$

Here $\|\cdot\|$ is the vector norm defined as the sum of its elements, $\delta$ is the predetermined calculation accuracy. As an initial approximation, a vector $\boldsymbol{\pi}^{(\boldsymbol{0})}$ with all the elements equal to $\frac{1}{N}$ can be used.

**Outlink matrix calculation:** The outlink matrix $\boldsymbol{O}$ is calculated as follows. First, the weight matrix is calculated:

$$w_{ij} = \sum_{k|j\to i, h_k \geq H_0 \wedge h_k \leq H_0 + W} \theta(a_k - T_0)\theta(s_i - A_0)\theta(s_j - A_0) a_k \exp\left(lnK[\frac{h_k}{D}]\right)$$

where $a_k$ is the token sum of transaction $k$, $h_k$ is the $k$-th transaction depth, i.e. the block order number from the current point, also known as a block height, $K$ and $D$ are transaction contribution decrease parameters, that define how much the contribution of each transaction decreases over time, $\theta$ is the standard Heaviside step function, and $[\cdot]$ is a standard floor function. According to the formula, transaction contribution decreases with the rate $K$ over every $D$ number of blocks, created after this given transaction. The sum is taken over all the transactions of a deposit from the account $i$ to the

account $j$, depth of which lays between $H_0$ and $H_0 + W$. $H_0$ and $W$ are the parameters, which values are currently equal to $H_0 = 0$ and $W = 17280000$ blocks. In this setting only, the transactions from the time gap, duration of which is $W$ blocks, contribute to the financial activity index calculation. Thus, we obtain the coefficient of the incoming financial activity from the user $j$ to the user $i$ as follows:

$$\hat{o}_{ij} = \begin{cases} w_{ji} - w_{ij} & \text{if } w_{ji} - w_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

After that the matrix, that was obtained, is standardized so that the sum of the elements in every column is equal to 1.

$$o_{ij} = \begin{cases} \frac{\hat{o}_{ij}}{\sum\limits_k \hat{o}_{kj}} & \text{if } \sum\limits_k \hat{o}_{kj} > 0, \\ 0 & \text{otherwise} \end{cases}$$

Outlink matrix $\boldsymbol{O}$ captures the structure of incoming financial transactions in the network graph.

**Interlevel proximity matrix calculation:** Let $\Omega$ be the set of all the accounts used for the financial activity index calculation. $\Omega$ is further divided into disjoint subsets $A_i$, called NCD (Nearly-Completely Departed) blocks. These blocks are obtained using the SCAN algorithm, described below. For the given account $u$, $G_u$ is the set of all the accounts, for which the according member of the outlink matrix $o_{vu}$ is greater than zero. Then the set $\chi_u$ of proximal accounts of $u$ is obtained as:

$$\chi_u = \bigcup_{v \in \{u\} \cup G_u} A_{(v)}$$

The interlevel proximity matrix is defined as follows:

$$M_{vu} = \begin{cases} \frac{1}{N_u |A_{(v)}|} & \text{if } v \in \chi_u \ , \\ 0 & \text{otherwise.} \end{cases}$$

where $N_u$ is the number of NCD-blocks in $\chi_u$. Interlevel proximity matrix $M$ represents the structure of connectedness within the network.

**SCAN-based network partitioning:** SCAN algorithm is used to partition network graph into clusters and to prevent activity imitation or fraud between several affiliated accounts [15]. An indirected graph $G = \{V, E\}$ has every vertex, representing a user, and every edge, representing a non-zero

element of the outlink matrix. The structure of the vertex $v$ is the set of all the adjacent vertices:

$$\Gamma(v) = \{w \in V | (v, w) \in E\} \cup \{v\}$$

The structural similarity of two vertices can be defined as follows:

$$\sigma(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

The vertex $\varepsilon$-neighborhood is a set of vertices for which

$$N_\varepsilon(v) = \{w \in \Gamma(v) | \sigma(v, w) \geq \varepsilon\}$$

The *CORE* is a vertex for which the number of elements in the $\varepsilon$-neighborhood is more than $\mu$.

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$$

Vertex $w$ is directly structurally reachable from vertex $v$ if

$$DirREACH(v, w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \vee w \in N_\varepsilon(v)$$

Vertex $w$ is structurally reachable from vertex $v$ if

$$REACH(v, w) \Leftrightarrow \exists v_1, ..., v_n \in V\colon \forall i \in \{1, ...n-1\}\colon DirREACH(v_i, v_{i+1})$$

Vertex $v$ is structurally connected with vertex $w$ if

$$CONNECT(v, w) \Leftrightarrow \exists u \in V\colon REACH(u, v) \vee REACH(u, w)$$

A cluster is a subset of vertices structurally connected to each other. It is possible to show that every vertex can only belong to one cluster. A vertex can also belong to no cluster; in this case it can either be a hub if there are vertices belonging to two different clusters in its environment, otherwise it is an independent vertex.

### 2.1.2 Social activity score calculation

U°OS blockchain can interact with a variety of DApps and register a defined set of social transactions in the network. Each account issues outgoing activity and receives input from other users. Together these connections form a graph with directed edges between the account nodes. Social rates, obtained

by the accounts, depend on the amount of the incoming social activity, which constitutes the utility of the user to the whole network. We denote this rate as the social network activity index. It contributes to the general importance rate, defined by the formula (1). At the moment algorithm collects *upvote* and *repost* transactions for the social activity rate computation.

Thus, social index calculation is based on several pieces of data. First, we define the matrix $\boldsymbol{V}$, that contains the information about the upvote transactions, with elements:

$$V_{ik} = \begin{cases} e^{[h_\alpha/D]lnK} & \text{if the account } k \text{ upvoted the content } i \text{ at the height } h_\alpha, \\ 0 & \text{otherwise.} \end{cases}$$

where $h_\alpha$ is the $\alpha$-th transactional height within the sliding window of transaction blocks with a fixed length $W$, i.e. the block height from the current time point. Only the transactions, that fit within this window, are taken into account during the calculation of the social rate and overall DPoI importance rate during every computation cycle. Weight $V_{ik}$ of each transaction decays in $K$ times every $D$ blocks within the window $W$. Therefore, if a user does not receive new incoming upvote transactions, their social rate decays with time.

Second, the matrix $\boldsymbol{P}$ contains the information about the content owners:

$$P_{ik} = \begin{cases} 1 & \text{if the content } k \text{ belongs to the account } i, \\ 0 & \text{otherwise.} \end{cases}$$

Third, the matrix $\boldsymbol{R}$ contains the information about the reposts:

$$R_{ik} = \begin{cases} 1 & \text{if the content } i \text{ is a repost of the account } k, \\ -1 & \text{if } i = k \text{ and the content } i \text{ is a repost,} \\ 0 & \text{otherwise.} \end{cases}$$

The purpose of the matrix $\boldsymbol{R}$ is to redirect the contribution of the upvote transactions from reposts to the original authors.

Given these matrices, we attain the matrix $\boldsymbol{V'}$:

$$\boldsymbol{V'} = (\boldsymbol{I} + \boldsymbol{R})\boldsymbol{V}$$

where $\boldsymbol{I}$ is a unit matrix.

And the intermediate matrix $\hat{\boldsymbol{U}}$, with elements:

$$\hat{U}_{ik} = max_n(P_{in}V'_{nk})$$

This matrix $\hat{\boldsymbol{U}}$ defines the relationship between the owner of the content and the user that sends the incoming upvote transactions. As you can see, the weight, that a voter transmits to the content owner, is the maximum of the weights of all pieces of content, that they upvote. It is based on the matrix $\boldsymbol{V}$, that represents the relationship between the voter and the content. Maximum value is used instead of sum to protect the algorithm from rate inflation as a result of automatic content generation and upvotes.

As it is characterized by the matrix $\boldsymbol{V}$, weights of the individual transactions $V_{ik}$ decay with time. Some accounts, however, might not have any outgoing activity for a period of time, longer than the window $W$. This results in the columns with zero elements in the matrix $\boldsymbol{V}$. Such condition violates the matrix stochasticity, that is necessary for the convergence of the iterative PageRank procedure.

There are several ways to resolve the issue. First solution is to fill the empty columns with equal values, while second option is to add a value of 1's to the diagonal elements of the matrix. Nevertheless, these solutions have notable drawbacks. In a former case, we would diffuse the account rates, such that the difference between the rates of active and inactive accounts becomes insignificant. In a later case, own activity of the account would result in the user rate decrease, contrary to their interest.

In order to provide effective decaying weights and preserve the matrix stochasticity, we add a special hidden account $h$, which accepts upvotes from every account. Existence of this account ensures that there will be at least one outgoing transaction for each account in the network at every point in time. For this account:

$$\forall i \neq h\colon \hat{U}_{hi} = 1, \forall i\colon \hat{U}_{ih} = 0$$

And after standardization:

$$U_{ik} = \begin{cases} \frac{\hat{U}_{ik}}{\sum_n \hat{U}_{nk}} & \text{if } \sum_n \hat{U}_{nk} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Unlike solutions, outlined previously, our answer does not have the mentioned drawbacks. Conversely, it presents several advantages. First of them is that we can now measure the activity of the network with regards to a fixed standard. We can monitor what proportion of the overall network rate is received by the hidden account, which indicates the rate of activity in the network — the higher is the rate of this account, the lower is the network activity. Second, now we have a standard, relative to which we can observe the weights decay. Since transactions towards the hidden account happen

regularly, the relative balance between the social index scores of the hidden account and other nodes demonstrates the level of score decay.

Besides above mentioned matrices, calculations of social index scores are based on the stack vector $\boldsymbol{s}$ and a validity weight vector $\boldsymbol{v}$, containing the validity values for every account. These values originate from the trust relationships on the network, and are described in details in the section 2.1.3 below.

Based on these data structures, we can calculate social activity vectors for the accounts $\boldsymbol{\sigma^a}$ and for the content $\boldsymbol{\sigma^c}$. These vectors are standardized to 1:

$$\sum_i \sigma_i^a = 1, \sum_i \sigma_i^c = 1$$

We calculate the values $\boldsymbol{\sigma^a}$ with a modified PageRank routine:

$$\hat{\boldsymbol{\sigma}}^{\boldsymbol{a}(n+1)} = (\alpha \boldsymbol{U} + (1-\alpha)\boldsymbol{T})\hat{\boldsymbol{\sigma}}^{\boldsymbol{a}(n)} \tag{3}$$

where the teleportation matrix $\boldsymbol{T}$ contains the information about the initial weights of the accounts:

$$\boldsymbol{T} = ((1-\beta-\gamma)\boldsymbol{e} + \beta\boldsymbol{v} + \gamma\boldsymbol{s})\boldsymbol{e}^T \tag{4}$$

Here $\beta$ and $\gamma$ are the coefficients with values between 0 and 1, $\boldsymbol{e}$ is a unit vector, $\boldsymbol{s}$ is the stack vector, and $\boldsymbol{v}$ is the validity vector. Utilization of a validity vector serves as the modification of a classical PageRank algorithm. Coefficients $\beta$ and $\gamma$ in the matrix $\boldsymbol{T}$ determine what proportion of the overall priority is allocated to the equal values, defined by the unity vector $\boldsymbol{e}$, and what proportion belongs to the other two components.

After the computation converges, we should subtract the contribution of a stack from the rating vector, because social index of an account should solely depend on the incoming social transactions:

$$\dot{\boldsymbol{\sigma}}^{\boldsymbol{a}} = \frac{1}{1-(1-\alpha)\gamma}(\hat{\boldsymbol{\sigma}}^{\boldsymbol{a}} - (1-\alpha)\gamma\boldsymbol{s})$$

In such a case, when stake of the user increases, it does not effect their own social activity rate, but only influences the social rates of other users.

Finally, we must remove the rating vector of the hidden account. This operation changes norm of the rating vector, so we must standardize it to 1:

$$\sigma_i^a = \frac{\dot{\sigma}_i^a}{\sum_{j \neq h} \dot{\sigma}_j^a}, i \neq h$$

Based on the final social rates of the network participants $\boldsymbol{\sigma^a}$, we can obtain social rates of the content, represented by the vector $\boldsymbol{\sigma^c}$:

$$\boldsymbol{\sigma^c} = \boldsymbol{V'\sigma^a}$$

### 2.1.3 Validity trust network

Normally, PageRank starts the calculation with the vector of equaly distributed weights, but in our algorithm for social index rate calculation, we modify this vector to integrate the system of trusts in our calculations. This is our innovation, that helps further protect the network from the botnets. Trusts are assigned by the members of the network to other members in a manual fashion and serve as a validation of their accounts. The user transmits trust proportionally to their stack amount. Such a condition ensures that a high level of trust is provided only by the meaningful users in the network. The higher is the level of overall trust that account obtains the higher is potentially the social index rate, they might receive.

The validity vector $\boldsymbol{v}$ is used in the formula (4). As mentioned before, it is invented in order to implement additional botnet attack protection and is calculated using the PageRank algorithm.

$$\hat{\boldsymbol{v}}^{n+1} = (\alpha O_T + (1-\alpha)T_T)\hat{\boldsymbol{v}}^n \tag{5}$$

Here $O_T$ is outlink matrix, that is based on the *trust* relations, $T_T$ is the teleportation matrix:

$$\boldsymbol{T_T} = ((1-\lambda)\boldsymbol{e} + \lambda\boldsymbol{s})\boldsymbol{e}^T$$

Validity trust scores, obtained with the PageRank algorithm in (5), are turned into binary values, using the Heaviside step function $\theta$:

$$\dot{v}_i = \theta(\hat{v}_i - 1/N)$$

and standardized towards a sum of 1:

$$v_i = \frac{\dot{v}_i}{\sum_k \dot{v}_k}$$

These scores are later used in (3).

## 2.2 Importance score usage: network governance and emission

Importance index $r_i$ in the framework is used for two major purposes:

- network governance through voting for validators (block producers), *calculator nodes*[†] and change of the algorithm parameters
- emission calculation for each user $i$

First, it defines the amount of new tokens received by the account in case when the emission is positive. See section 3 for details.

Second, importance index defines the relative weight of the account during the voting for producers and for the *calculator nodes*[‡]. A single user can offer their candidacy for both groups. Voting allows the delegation of certain powers within the system to a limited number of the validation accounts. Producers, that own nodes which produce and verify blocks, as well as the *calculator node*[§] owners, are selected by voting by other members of the network. Candidates nominate themselves for a desired position and voting can be performed by other users at any time. Changes are recorded within a short period, according to the protocol routine.

Change of the blockchain algorithm parameters and the block validation procedure is performed using voting as well. Here users delegate these decisions to the *calculator nodes*[¶].

## 2.3 DPoI vulnerability modeling and resistance to attacks

The main aim of modeling the behaviour of the algorithm is to find the vulnerabilities, accepted in the system, and to propose suitable improvements to the method. Here we research the weak spots of the DPoI consensus algorithm, that was described in details in the sections above.

According to the algorithm, financial and social activity indices are calculated for every account. These indices measure the useful input of each account in the network. Together these two indices contribute to the total importance index of the network participant, according to which the active accounts are rewarded for their network utility.

Thus, there is an incentive to manipulate the DPoI algorithm in order to receive a large activity indices in absence of an actual fruitful exertion.

There are several possible vectors of attacks:

- Creation of a subnetwork of isolated accounts and imitation of the useful activity between them, defined here as an *account splitting attack*

[†]See Appendix 1 Glossary for definition
[‡]See Appendix 1 Glossary for definition
[§]See Appendix 1 Glossary for definition
[¶]See Appendix 1 Glossary for definition

- Expansion of activity index of an existing active account, using additional set of accounts, known here as a *rate expansion attack*
- Security of the activity index from the alien accounts, that have a large activity index, denoted here as a *rate stealing attack*
- Increase of the rate of a single account in the attacking subnetwork, that contains a large number of accounts with a high or minimal rates, named here as a *rate concentration attack*

We perform analysis of each type of these attacks below.

### 2.3.1 Test network

### 2.3.2 Account splitting attack

**Financial index DPoI:** We consider an account splitting attack in a network, consisting of a base and an attacking subnetworks. Let us assume that the attacking subnetwork contains $N$ individual accounts. These users transfer some amount of assets between each other in a cyclical manner. The objective of the attack here is to obtain the maximum fraction of the activity index of the entire network.

Financial activity index is calculated with the NCDAwareRank algorithm by setting the parameter $\mu$ to a value greater than 0. In case when $\mu = 0$, NCDAwareRank procedure is reduced to a basic PageRank method.

Based on the parameter $\mu$ and the size of the attacking subnetwork $N$, the modeling of the attack produces the following results:

| | Activity index fraction | |
|---|---|---|
| N | $\mu = 0$ | $\mu = 0.1$ |
| 10% | 0.12973 | 0.12711 |
| 20% | 0.22997 | 0.22583 |
| 30% | 0.30965 | 0.30465 |
| 40% | 0.37295 | 0.36741 |
| 50% | 0.42668 | 0.42089 |

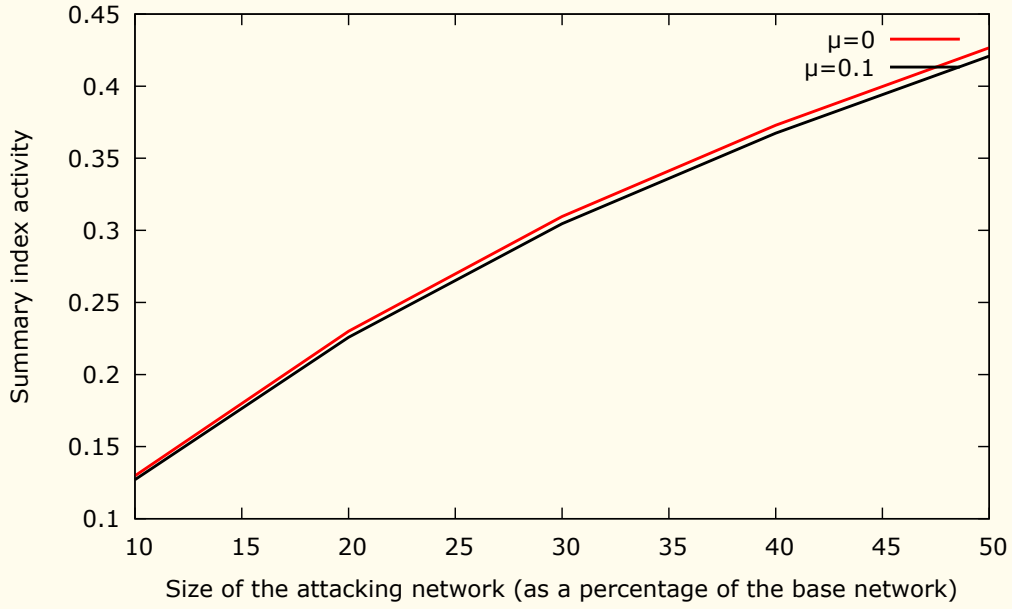


Figure 1: Attack results

From the visual representation on figure (1), we conclude, that the total activity index increases with the number of accounts. Therefore, additional steps are necessary to protect the base subnetwork from this type of attack. Two of possible provisions here are the nonzero stack threshold for accounts, that participate in the rating calculation, or a nonzero cost for the initiation of a new account.

We will discuss this problem in more details.

**Social index DPoI:**

### 2.3.3 Rate expansion attack

**Financial index DPoI:** Let us consider the most active account in the base subnetwork. Financial activity index of this account is equal to 2.36e-03, i.e. 0.236% of the overall network activity.

We also examine the attacking subnetwork which consists of 100 accounts, that performs rate expansion attack. The aim of this attack is to obtain the maximal possible total activity index. All of the 100 accounts cyclically transfer the assets through the subnetwork, starting and finishing with the largest account of the attacking subnetwork. The result is on figure (2).

You can see that with the growth of the exchange volume with the largest account, the total activity index of the attacking network grows as well. This

growth, however, slows down when reaching a threshold, which makes it ineffective to keep increasing the exchange volume. You can see also that the NCDAwareRank algorithm ($\mu = 0.2$) is more resistant to this kind of attack, than the PageRank algorithm ($\mu = 0$).

Generally, it can be shown that there is a limit for the total activity index, that one can get with this kind of the attack. This limit is equal to:

$$\frac{\mu + \eta}{1 - \mu - \eta} R$$

$R$ is the activity index of the large account. Meaning of coefficients $\mu$ and $\eta$ are described in

This kind of an attack can be used to obtain the rating using one's own large account as well as an alien account.

Obtaining a significant activity index requires a high exchange volume, which must be comparable to the total outgoing volume of the large account. Providing the exchange volume that is high enough is not an easy objective in case, when one is using an alien large account.

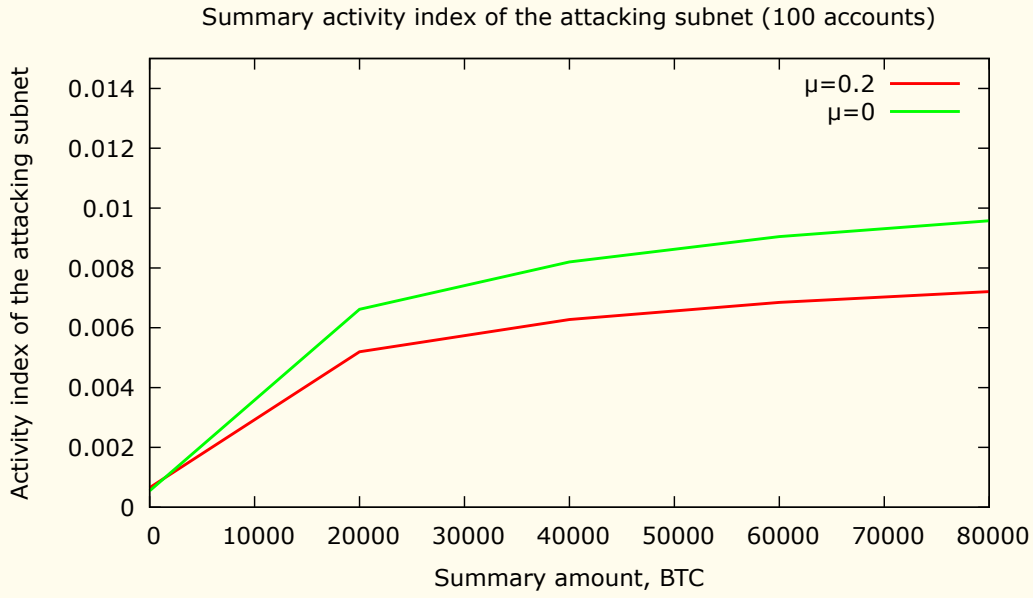


Figure 2: Results of the rate expansion attack

**Social index DPoI:**

### 2.3.4 Rate stealing attack

### 2.3.5 Rate concentration attack

### 2.3.6 Conclusions

Following mitigations are used in the project in order to reduce the risk of attacks:

- Only incoming transfers contribute to the importance index, which makes obtaining score from the existing accounts more difficult.
- There are stake thresholds for accounts, participating in the importance index calculation, and also the amount thresholds for transfers, which makes attacks more expensive for the attackers.
- Coefficients $\omega_a$ and $\omega_s$ have relatively small values, this makes contribution of activity index and social index to total importance index significantly smaller than the stake contribution.
- Usage of the decaying weights makes effect of any fraud activity temporary.

# 3 Emission

## 3.1 U°OS token utilization

U°OS tokens constitute the core of our crypto-economy. They are used in the system in several ways:

- to allocate CPU and bandwidth resources, using staked token amounts. Only core U°OS tokens can be used for CPU and bandwidth resource allocation.
- to purchase other resources, such as RAM, and perform other forms of financial transfers via smart contracts using unstaked tokens. Potentially, many types of tokens can be used for these activities.
- to vote for block producers and *calculator nodes*[‖]. The amount of staked tokens, owned by the account, contributes to the user's importance during the voting process.
- to increase the importance score and receive the dynamic emission. Amount of tokens staked by the account directly influences the importance received by the user and the amount of dynamic emission.

Thus, staked core tokens are used for the resource allocation and play an important part in emission and importance calculation, while unstaked tokens can be used in direct transfers.

Emission amount at launch constitutes one billion of protocol tokens, distributed to the original network accounts to start the protocol. The U°OS project implements adaptive emission. The emission volume is calculated regularly, in a certain time interval, $t_0, t_1, ...t_i$, where $t_{i+1} = t_i + T$. The volume of emission depends on the network activity growth in the preceding time period $T$.

## 3.2 Network activity calculation for a period of time

In order to calculate the emission, we first need to calculate the network activity for the unit time period $T$, defined by the U°OS protocol. To begin, we calculate the matrix of weights, according to the formula:

$$w_{ij}(t_n) = \sum_{k|j \to i, t_k \in [t_{n-1}, t_n]} a_k$$

[‖]See Appendix 1 Glossary for definition

Here $a_k$ is the sum amount in k-th transaction, $t_k$ is the time at which k-th transaction was created. Summation is performed for all the transactions transferring any amount from account $j$ to account $i$ and created at the time frame from $t_{n-1}$ to $t_n$.

In fact, each matrix element $w_{ij}$ represents a weight of connection between account $i$ and account $j$ in a given time frame. Next we need to calculate the matrix of connections $l$:

$$l_{ij}(t_n) = \begin{cases} 1 & \text{if } w_{ji}(t_n) - w_{ij}(t_n) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

We calculate the activity in a given time frame as:

$$A(t_n) = \sum_{i,j} l_{ij}(t_n)$$

In this way, activity is calculated as a number of connections between active accounts in a set timeframe.

## 3.3 Emission value calculation

Target emission value $E_T$ depends on the network activity growth. It defines the upper bound of the aggregate amount of the emission, that is achievable with the following activity value $A$:

$$\Delta A(t_n) = A(t_n) - A_{max}(t_{n-1})$$

$$E_T(t_n) = \begin{cases} E_T(t_{n-1}) + K_E \Delta A(t_n), & \text{when } \Delta A(t_n) > 0, \\ E_T(t_{n-1}) & \text{otherwise} \end{cases}$$

Here $K_E$ is a coefficient that defines the maximum value of the emission with the activity increased by 1. $A_{max}(t_{n-1})$ is the previous maximum value since the system launch:

$$A_{max}(t_{n-1}) = max\Big(A(t_i), t_i \in [t_0, t_{n-1}]\Big)$$

Emission value, which is issued at a certain time $t_n$, is defined by the formula:

$$E(t_n) = \lambda S(t_{n-1}) f\Big(\kappa \frac{E_T(t_n) - E_S(t_{n-1})}{\lambda S(t_{n-1})}\Big)$$

Here $\lambda$ is the marginal growth of the token amount in the system S per one emission. It is defined through $L$, which specifies the marginal growth $S$ in a year, expressed as a percentage:

$$\lambda = (1 + \frac{L}{100})^{1/N} - 1$$

Here $N$ is the number of emission issues per year.

$f(x)$ - is a sigmoidal function. In the present implementation of the algorithm a hyperbolic tangent is used as this function.

$\kappa$ is a coefficient between 0 and 1 and it defines the speed at which a full emission approaches the target emission $E_T$ if the activity level remains the same over the long term.

Initial values of both $E_T$ and $E_S$ are zero:

$$E_T(t_0) = 0, E_S(t_0) = 0$$

Dynamic emission allocated to a user for their social and financial activities motivates the user to participate in the network development, thus, helping to achieve the overall network growth.

# 4 U°OS framework architecture

In this section we outline the general architecture of the U°OS blockchain.**We have several types of nodes in the system, connected via the peer-to-peer protocol, namely, *api*, *block producer*, *calculator*, and *seed* nodes. Nodes that have a connection between each other are called *peer* nodes. Nodes can be accessed by the owner alone or by any user via *private* or *public* APIs, respectively. Every node, irregardless of their type, performs the following activities:

- synchronizes and distributes pending transactions into new blocks and new blocks between the peer nodes via peer-to-peer connection
- communicates with the other programs and receives the information about the state of the blockchain (accounts, token sums, list of block producers, voting results, smart contracts, data tables, transaction blocks, and others) via private and public APIs

The node types are briefly defined as follows:

- *Block producer node* – owned by the block producer, issues blocks according to the schedule and distributes them among the peer nodes
- *Calculator node* – calculates user importance scores, based on social and financial transactions from irreversible blocks, and saves the results to the blockchain
- *Seed node* – a node, that has a public peer-to-peer connection, it is used by the new network participants, when they join the network
- *API node* – a node with a public API

**Detailed documentation is located at the U°OS GitHub and at the EOS.io website: https://developers.eos.io

# Appendix 1 Glossary

**U°OS consensus algorithm (Delegated Proof-of-Importance, DPoI)**

The consensus algorithm, which is based on a calculation of the importance index of an account, which in turn depends on Stake Volume Index and Activity index

**Importance index**

The importance index of an account is calculated as a function of Stake Volume Index and Activity index

**Account**

An entity represented by a tuple of pairs of keys (public + private), which is registered in a blockchain by an individual.

**Block Producers**

Accounts with the right to verify blocks. They must have a node.

**Node**

A P2P network node, which performs all the calculations in blockchain. A node belonging to a producer account (producer node), produces blocks.

**Calculator node**

A node that performs importance score calculation and distributes calculated scores back to the network

**Splitting account attack**

An attack performed by a network of many accounts. The goal of the attacker is to receive a maximal total importance rate while utilizing minimal resources

# References


[1] Metcalfe, B. (2013). Metcalfe's law after 40 years of ethernet. Computer, 46(12), 26-31. URL: http://ieeexplore.ieee.org/abstract/document/6636305/

[2] Lamport, L., Shostak, R., Pease, M. (1982). The Byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS), 4(3), 382-401. URL: https://www.microsoft.com/en-us/research/uploads/prod/2016/12/The-Byzantine-Generals-Problem.pdf

[3] Castro, M., Liskov, B. (2002). Practical Byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems (TOCS), 20(4), 398-461. URL: https://dl.acm.org/citation.cfm?doid=571637.571640

[4] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. URL: https://bitcoin.org/bitcoin.pdf

[5] Croman, K. et al. (2016). On scaling decentralized blockchains. In International Conference on Financial Cryptography and Data Security (pp. 106-125). Springer, Berlin, Heidelberg. URL: http://www.comp.nus.edu.sg/ prateeks/papers/Bitcoin-scaling.pdf

[6] Eyal, I., Sirer, E. G. (2014). Majority is not enough: Bitcoin mining is vulnerable. In International conference on financial cryptography and data security (pp. 436-454). Springer, Berlin, Heidelberg. URL: https://arxiv.org/pdf/1311.0243.pdf

[7] Bitcoin Energy Consumption Index. digiconomist.net. URL: https://digiconomist.net/bitcoin-energy-consumption

[8] Buterin, V. (2014). Mining Pool Centralization at Crisis Levels. URL: https://bitcoinmagazine.com/articles/mining-pool-centralization-crisis-levels-1389302892/

[9] Bentov, I., Gabizon, A., Mizrahi, A. (2016). Cryptocurrencies without proof of work. In International Conference on Financial Cryptography and Data Security (pp. 142-157). Springer, Berlin, Heidelberg. URL: https://link.springer.com/chapter/10.1007/978-3-662-53357-4_10/

[10] King, S., Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. URL: https://peercoin.net/assets/paper/peercoin-paper.pdf

[11] Demeester, T. (2017). Critique of Buterin's A Proof of Stake Design Philosophy. URL: https://medium.com/@tuurdemeester/critique-of-buterins-a-proof-of-stake-design-philosophy-49fc9ebb36c6

[12] Poelstra, A. (2014). Distributed consensus from proof of stake is impossible. URL: https://download.wpsoftware.net/bitcoin/old-pos.pdf

[13] Dantheman. (2017). DPOS Consensus Algorithm - The Missing White Paper. URL: https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[14] NEM Technical Reference. Version 1.2.1. February 23, 2018 URL: https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf

[15] Xiaowei Xu et al. (2007). SCAN: A Structural Clustering Algorithm for Networks. URL: http://www1.se.cuhk.edu.hk/ hcheng/seg5010/slides/p824-xu.pdf

[16] Zhang E., A Byzantine Fault Tolerance Algorithm for Blockchain. URL: https://docs.neo.org/en-us/basic/consensus/whitepaper.html

[17] VIVA White paper (2017). URL: https://s3.amazonaws.com/vivacoin/viva-white-paper-v-2-0.pdf

[18] David Mazieres, (2016), The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus. URL: https://www.stellar.org/papers/stellar-consensus-protocol.pdf

[19] Evan Duffield and Daniel Diaz, (2018), Dash: A Payments-Focused Cryptocurrency. URL: https://github.com/dashpay/dash/wiki/Whitepaper

[20] Colin LeMahieu, (2015), Nano: A Feeless Distributed Cryptocurrency Network. URL: https://nano.org/en/whitepaper

[21] A Next-Generation Smart Contract and Decentralized Application Platform (2018). URL: https://github.com/ethereum/wiki/wiki/White-Paper

[22] EOS.IO Technical White Paper v2 (2018). URL: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[23] An Introduction to Hyperledger, (2017), URL: https://github.com/hyperledger/hyperledgerwp/blob/master/paper.pdf

[24] NEO White Paper: Smart Economy, (2018). URL: https://docs.neo.org/en-us/whitepaper.html