NEXTY: A CONSENSUS TO GET ZERO TRANSFER FEE AND INSTANT TRANSFER BLOCKCHAIN

THANH DAO & HA DANG
CO-FOUNDER/CTO @ NEXTY PLATFORM
THANHDAO@NEXTY.IO / HADANG@NEXTY.IO

Abstract. This is the detail technical paper of Nexty Platform, focusing on describing the operation of a consensus protocol called Proof of Foundation. Proof of Foundation was inspired by the Proof of Authority introduced by Szilágyi [2017] with improvement from Nexty Platform by introducing a new confirmation system named **DCCS** - **D**ual **C**ryptocurrency **C**onfirmation **S**ystem to achieve a decentralized system and bring a highly incentive system for blockchain maintainers.

1. Introduction

In addition to NTY coin-base, DCCS has the secondary token named NTF. NTF is used for authorizing an account to become the maintainer/sealer of the confirmation system described detail in section 2, as well as to calculate reward for block sealer. NTF token has total supply of $10,000,000$ and will be distributed for the first $100,000,000,000$ NTY holders that joined in "Smart Staking" program described detail in the white paper of Nexty [2017]. In the other words, NTF will be rewarded to the pioneers having a clear vision and a strong believe in Nexty Platform in the future. That's the reason why we call the new consensus protocol as *"Proof of Foundation"*. The power of Nexty Platform, however, is not belong to NTF holders because they could be voted down by Nexty community if community found that NTF holders did any bad behavior of cheating, malicious or hand-shaking to make the network become centralize chain as well as using out-of-date source code. As a result, NTF holders have the only role as block sealer for Nexty blockchain and governed by NTY community via decentralize voting system on a smart contract.

2. How to authorize an account to become a block sealer

The system will set up a configuration parameter to determine the minimum value, $min - ntf$, that an account need to have to become block sealer. Nexty will build and develop a smart contract, which allow NTF token holder, having enough token, to grant another account called "executing-account" to become the block sealer by setting "authorized-sealer" state in the smart contract with the value equal to address of NTF holder. If an address already has an "authorized-sealer" value, it can not receive authorization from another NTF holder until the NTF holder has made the withdrawal it from "authorized-sealer". The "authorized-sealer" list will be determined at the check point block of each sealing round by reading state from the smart contract at that point of time. If in the current sealing round, any "authorized-sealer" does not perform at least one sealing activity, the "authorized-sealer" value will be withdrawn to its NTF holder by updating the state of the smart contract at the checkpoint block number of the next sealing round and of course that "authorized-sealer" will not be involved in the next sealing rounds until it will be authorized to become "authorized-sealer" again.

3. Block sealing mechanism

The DCCS block sealing mechanism will be implemented as in following steps. Firstly, the sealers will be numbered from 0 to $n-1$ (called "sealing-id") randomly at the beginning of each sealing round; in which, n is the number of registered "authorized-sealer" and is determined by the state of the smart contract at the *checkpoint block number* of the corresponding sealing round. To ensure the randomness of "sealing-id", the numbering is calculated by a hash, $\boldsymbol{\xi_k}$, as following formula when starting a new sealing round.

$$\xi_{\mathrm{k}} \equiv \mathtt{KEC}(\mathbf{block}, \boldsymbol{\Lambda}_{\mathrm{k}}) \tag{1}$$

block: is the first block number, a.k.a *checkpoint block number* of the sealing round.

Λ_{k}**:** is the address that the NTF holder has set as "authorized-sealer" in the smart contract.

`KEC`**:** is the **SHA-3 Keccak-512** hash function of any input.

After that, the "sealing-id" of each authorized-sealer will be taken by the position of *sealing hash*, $\boldsymbol{\xi_k}$, in the array $(\xi_0, \xi_1, ..., \xi_{\mathrm{n}-1})$, ascending ordered by the *sealing hash*.

To ensure performance of the system, the "sealing-id" of the all authorized sealers will be snapshot only once from the smart contract state at the *checkpoint block number* of each sealing round and stored in the local database as well as in **lru cache** of each node.

3.1. **Scenario 1.** If sealing node is not in recent sealers.

The node will determine whether it's the in-turn sealer for the next block or not, according to the following formula:

$$\sigma_{\mathrm{k}} \equiv (\nu - \mathbf{block}) \quad \mathbf{mod} \quad \Pi \tag{2}$$

ν**:** is the current block number that is being sealed.

block: is the first block number, a.k.a *checkpoint block number*, of the current sealing round.

Π**:** total number of authorized sealer reading from smart contract state at the *checkpoint block* of the current sealing round.

If the remainder σ_{k} equal to "sealing-id" of the node, then that node has the right to seal block immediately. Otherwise, at each of the next block, sealing nodes have to wait a course of time, ψ_{k}, which is calculated by the formula:

$$\alpha = 001.387978000 \tag{3}$$
$$\beta = 000.002313279 \tag{4}$$
$$\gamma = 000.004626590 \tag{5}$$
$$\delta = 199.999400000 \tag{6}$$

$$\psi_{\mathrm{k}} \equiv \sum_{i=1}^{\zeta_{\mathrm{k}}} \mathbf{floor}\left(\frac{\alpha}{\beta * i + \gamma} + \delta\right) \tag{7}$$

3.2. **Scenario 2.** If sealing node is in recent sealers

In this scenario, sealers still can continue to seal block, but they have to wait a period of time longer than all other sealers, not in recent sealers. This is procedure to ensure that the system can still generate blocks, even when there is only one sealer, but the chain be suspended for a very long time. The waiting time ranges from 3 - 30 minutes, if there are $1,000$ - $10,000$ authorized sealers and only one or few active sealers.

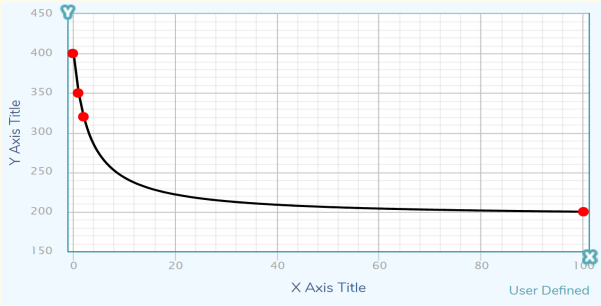
Waiting time, ψ_{k}, is caculated by the following formula:

$$\alpha = 001.387978000 \tag{8}$$
$$\beta = 000.002313279 \tag{9}$$
$$\gamma = 000.004626590 \tag{10}$$
$$\delta = 199.999400000 \tag{11}$$

$$\psi_{\mathrm{k}} \equiv \sum_{i=1}^{n} \mathbf{floor}\left(\frac{\alpha}{\beta * i + \gamma} + \delta\right) + \sum_{i=1}^{\zeta_{\mathrm{k}}} \mathbf{floor}\left(\frac{\alpha}{\beta * i + \gamma} + \delta\right) \tag{12}$$

The coefficients of the above formula are assumed as following. The sealing node having "sealing-id" equal to σ_{k}, calculated by formula (2), will have the priority to seal block before 400ms, the next sealing node will have the priority to seal block before 350ms, then 320ms,... till the final sealing node with approximately to 200ms. Its graph corresponds to the function $\mathtt{y} = \frac{a}{(b*x+c)} + \mathtt{d}$

To estimate values for the parameters, we use curve fitting method to find relatives coefficient in the website My Curve Fit with the variance [1] ≈ 1.



A sealing round of all authorized sealer is called an epoch. Epoch will have various length, depends on the number of sealers in a chain of each sealing round.

A sealer who is worried about being withdrawn authorized sealer to NTF holder might not participate in sealing activity in a sealing round, but still occupies the right to seal block at some point earlier than his/her sealing turn; causes welter and racing for sealing block. By calculating the fixed waiting time for each sealer, using formula (7) or (12), consensus are able to keep track of sealer. Nexty blockchain will verify a block, sealed by an "authorized-sealer", by figuring out if the generating time of "authorized-sealer"'s block is smaller than it's waiting time.

4. Block sealer reward calculation

After each sealing round, reward value $\Re$ of each sealer will be calculated as the minimum value of the total reward divided by the active sealers according to the proportion of NTF they have on the total NTF of the sealers who participated in the seal.

$$\theta = \frac{\Omega}{N} \tag{13}$$

$$\phi = \frac{\Upsilon_{NTF}}{\Xi_{NTF}} \tag{14}$$

$$\Re \equiv \min(\theta, \phi) \tag{15}$$

Ω: total reward
N: total number of active sealer of the current sealing round
Υ_{NTF}: amount of NTF token that the owner of active sealer is holding
Ξ_{NTF}: total amount of NTF token that all active sealer are holding

According to formula (15), people who own NTF in a natural way will divide his/her NTF into many wallets in order to equip multiple sealers, instead of building only one sealer and spend too much NTF on it. This ensures the Nexty chain will have sufficiently large number of nodes to make Nexty chain become highly decentralization, stability and robust.

Total reward Ω is the amount of NTF was produced after each sealing round, calculated as *[reward generated in one block] * [number of blocks generated in that sealing round]*.

The number of rewards generated for a block is equal to the number of rewards per year divided by $\mathbf{15,768,000}$, the expected number of block within one year (initial configuration of block time is 2 seconds).

[1] R^2 is 1 minus the ratio of sum of the squares of the residuals divided by the sum of the squares of the differences between Y fit and the mean Y value

The amount of rewards in a year is calculated in the following proportions:

Year	% of circulating supply
1	10
2	5
3	2
4	1
5	1
...	...
...	...
...	...
n	1

5. Anti-SPAM

Since Nexty's transaction fee is zero, if there is no proper anti-spam protection, it can easily be attacked by sending multiple transactions simultaneously, causing system blockage.

We have the following anti-spam methods:

The general rule is that the system will charge the first transaction with a fee of *0.2 per account*, which will be added to the reward value of the sealing round. From subsequent transactions, the system will not charge transaction fees, but requires those accounts to run a continuous wallet application with the number of blocks equal to *[0.0015 * gas-used]*, after that, they will be able to send next transactions.

Mobile wallet app is a light client version, as known as a block verifier (based on light client protocol standard) which has the ability to save transactions of the it's own account and contiguous accounts in *Merkle Tree* and block headers. The responsibility of a light client is to verify all transactional information related to it, as well as validate the blocks generated afterward. There should be a comparison if the blockchain is interfered with more than 15 blocks backwards, then notify to the owner of the wallet that the system might be manipulated. In some case, community must be accessed to see if any errors occurred, and vote down on the "authorized-sealers" that caused the malicious data. Under this rule, a spam action that spends more than 56.7 times the amount of resources on the chain to confirm such a transaction. This results in a loss in resources for spammers and therefore, ruins their intention of spamming.

Users still have the option of creating a charged transaction in case they do not want to maintain block verifier.

For **dApp**, it can provide its own system of nodes to withstand the user's gas and then there may or may not be a charge on the user's gas depending on the development needs of the **dApp**.

References

Nexty. Nexty platform white paper, Dec. 2017.

Péter Szilágyi. Clique poa protocol and rinkeby poa testnet, Mar. 2017.