**OTC-RFC-14:**

**OriginTrail Club on TRAC Tokenomics**

Authors: OriginTrail Club  
Contributors: BRX, Dmitry, LuKu  
Date: Oct 14, 2022

## Introduction

OriginTrail Club is actively gathering all community feedback regarding all RFCs by the core developer team of OriginTrail and making them easily readable in one place.

Moving forward, we will continue to represent the core community’s interest and allow every community member to be heard, including those who do not wish to open a GitHub account to comment.

## Points of interest

1. 50k TRAC minimum required per full node
2. New stake slashing mechanism replaces litigations
3. Current default stake slashing values: 5% of TRAC locked for 2 years
4. The higher the stake, the higher the amount of tokens slashed, therefore bigger nodes are more likely to win a publish due to higher risk / reward
5. Retiring a node
6. Node backups, epochs lengths, lambda ?

## Feedback

1. **50k TRAC minimum required per full node**

The general consensus seems to be a positive one regarding the increased amount to run a node. Given the current market conditions, this is equivalent to 8,500$ USD as a minimum amount, and with the help of delegators, should be attainable by most users. The main concern would be whether the number of nodes would be enough to meet sufficient decentralisation across all neighborhoods (i.e. max amount of nodes would be probably around 5-6k).

1. **New stake slashing mechanism replaces litigations**

This is a welcome change as litigations prior to v5 would punish the node runner harder by completely removing the node runner’s stake. Slashing, or locking up a certain amount of tokens for a fixed duration, rendering the tokens useless in the meantime, serves as a good deterrent to node runners and promotes good behavior. In the community, it is almost an unanimous agreement that slashing instead of litigation is an improvement to the ecosystem.

The disagreement lies within the stake slashing default values.

1. **Current default stake slashing values: 5% of TRAC locked for 2 years**

These values seem to be quite high without knowing the specifics. Node runners need to know a clearly defined definition of stake slashing. In other words, how long can a node remain offline without triggering a slash ? How does a node know, despite being online, that it is functional and detected by the network ? Is slashing multiplicative or additive ? Is slashing possible by several assets I hold, several times a day, or is it capped to one slashing per 24 hours ? Will the slashed amount be affecting delegators as well or just the full node runner ?

A poorly designed slashing method could break the entire ecosystem. The benefits of running a node must always outweigh the risk of locking up funds. In real life, no centralized system is up 100% of the time. The stake slashing must have more metrics to distinguish bad actors from accidental outages.

In the event of a datacenter fire or any events causing a complete loss of data, how does a full node recover from that after rebooting using the same credentials, but without any datasets ? If a backup is available but is 1 week old, how likely is slashing if 1 week of assets have been lost ? These are all questions that need to be answered for node runners to better plan their node maintenance.

There is a clear agreement in the community that both metrics (5% of TRAC locked for 2 years) are way too high. For example, failing an epochs check for an assertion worth 0.01 TRAC and risking a lock for 2 years of 2500 TRAC (for a 50k full node) is way too high of a punishment. The stake slashing must be relative to the assertion amount and duration and be a lot more reasonable. 2 years in crypto time is a lot and way too big of a deterrent for node runners - something like 6 months should be considered instead. 5% of available TRAC for a 50k TRAC full node is 2500 TRAC. Depending on the token price, that also seems to be quite a lot if failing only 1 epochs check for 1 assertion can lock out that much TRAC.

A hold in the stake slashing execution should also be considered at the start of V6 mainnet to make sure the slashing mechanism operates as designed in production.

1. **The higher the stake, the higher the amount of tokens slashed, therefore bigger nodes are always selected as winners due to higher risk / reward**

This is a hot topic among the node runner community. This black-or-white system might push node runners to always one up their peers. For instance, node runner X with 50,000 TRAC will add 1 TRAC to beat node runner Y with 50,000 TRAC to win every single assertion. The difference in their staked amount does not necessarily represent their quality of service and slashing, if it happens, would almost represent an almost identical amount, making the risk similar but reward skewed towards node runner X. If this type of behavior continues, we will find ourselves with a huge amount of mega nodes and hurting decentralization in the end.

A great alternative would be to erase the all-or-nothing system and replace it with a probabilistic system.

Here is an example:  
  
The total R1 (set of nodes bidding on the assertion) has a total amount of 500,000 TRAC. Each node has a fixed probability of becoming R0 (winning the assertion).

R1:

Node A = 50,000 TRAC = 50/500 = 10%

Node B = 50,000 TRAC = 50/500 = 10%

Node C = 100,000 TRAC = 100/500 = 20%

Node D = 150,000 TRAC = 150/500 = 30%

Node E = 150,000 TRAC = 150/500 = 30%

The dices are thrown and the R0=3 group becomes:  
Node B,C,E

An alternative method would have the dice thrown 3 times (if R0=3) instead of once.

1. **Retiring a node**

More details are needed to understand the process of retiring a node without triggering a slash. There needs to be a way to remove our commitment in a reasonable amount of time and responsibly.

In order to balance supply and demand, both supply and demand must be flexible. As seen in versions prior to V6, node numbers did not go down despite lower demand, simply due to the fact that node runners were obliged to finish their jobs or risk being litigated. This caused a race to the bottom with nodes accepting jobs for a negative return which shouldn't happen.

Now, with the introduction of epochs checkpoints where node runners get paid a partial amount after proving they did their job, it is a perfect opportunity to allow the node to retire responsibly and let the asset holding back to R1 for a new bidding. That way, it will not cause bad behaviour (since the node still risks getting slashed if not able to provide proof of work), while promoting responsible continuation of the service through another bidding within the neighborhood and allowing the node to retire safely. This, in turn, will allow the supply of nodes to go down in a flexible manner to match demand.

1. Other feedbacks

Other points touched upon include node backup epochs length and lambda

Node backups also need to be discussed and how quickly we need to restore a node in case of an outage. The team must provide more guidance on the best ways possible to backup in order to prevent slashing.

Epochs, or checkpoints during service, should be lengthy enough to allow node runners some lee-way to fix their node in case of an outage. We need more information about the epochs' length, and whether they are fixed for all assertions or not.

Lambda has been a hot issue plaguing v5 node runners and hasn't been mentioned once in this RFC. Can we have a word about how lambda has been adapted to V6 ? Also there was an RFC 06 regarding the automated lambda, is that approach still relevant and would it be included sometime after v6 is released?

Neighborhoods - would be ideal if we get some clarity on what is the process of nodes assignments to these in both initial node install, and later when a new neighborhood is created. Also how would the system decide if more nodes are needed in a certain neighborhood, depreciation of neighborhoods, and how would the DKG ensure all nodes are having fair access to the data uploaded on the DKG.