\DeclareMathOperator{\Hash}{Hash} \DeclareMathOperator{\True}{True} \DeclareMathOperator{\False}{False} \DeclareMathOperator{\b}{b} \DeclareMathOperator{\r}{r} \DeclareMathOperator{\d}{d} \DeclareMathOperator{\DiracDelta}{DiracDelta} \DeclareMathOperator{\pow}{pow}

Disclaimer

The project is proof of concept, built for ETHParis. It is not production ready yet. This message is draft and I hope for your discussions and criticism.

Abstract

There are a lot of dead ICO tokens, and their price will be never zero because the tokens have all properties of bitcoin. But the values and liquidity of these tokens are small.

What we do: burn ICO tokens on the smart contract and mint PYRO token.

This way help to free a lot of memory slots in ethereum from unused tokens and try to melt the trash into something useful.

No Oracles here

We need to unbind the contracts for any centralized sources. There are two problems that we need to solve:

- how many PYRO tokens we need to mint for each burning
- how we can safely add new addresses of tokens to the burning list

The first problem is solved via our minting math model.

Pyromania math model

Let’s determine the burn rate for a token as follows:

r(t) = \int\limits_{-\infty}^{t} \gamma b(t) e^{\gamma (\tau-t)} \d \tau

.

Substituting b(t) = \sum\limits_{t_j<t} b_j \delta(t-t_j)

into this equation, we got

r_i=\sum_{j \leq i} b_i e^{\gamma (t_j - t_i)}

.

Believing t_i

is block number and rewriting the expression as recurrence relation, we got

r_i = r_{i-1} \alpha^{\Delta t_i} + b_i

where \Delta t_i

is the number of blocks between burning events, b_i

is the amount of tokens burned by the user at the ith event and \alpha \lt 1

is a decay parameter. We set it in such a way that the rate roughly halved every day.

Let’s determine price of minted tokens to burned tokens as \frac{a}{2 \sqrt{r}}

, where a

is scaling parameter. Such a system is elastic by burn supply: if we want to burn more tokens with a higher burning rate, the cost of burned tokens will be lesser.

Then amount of minted tokens is caluclated by following formula:

c_i = \int\limits_{0}^{b_i} \frac{a}{2\sqrt{r_{i-1} \alpha^{\Delta t_i} + \beta}}\d \beta = a (\sqrt{r_i} - \sqrt{r_i - b_i})

If there is a situation where there is only one token-holder, they can emulate minting by spending little amounts of a token. To protect from this we add a new variable \kappa

– a price cap so that c_i/b_i

can not be more than \kappa

.

If c_i/b_i > \kappa

, we redefine c_i

as c_i = b_i\kappa

.

At each step we determine \kappa = (\kappa_0 \omega + \sum_{i-1} c_i)/(\omega + \sum_{i-1} b_i)

, where

\kappa_0

and \omega

are initial parameters.

Under discussion: emission limiting

if we add new coins to burn or more people become to burn coins, total emission rate grows. It may be not good for the cryptocurrency. We can limit the emission by the following way:

Define q_i

as old c_i

.

Total unnormalized minted value for all coins at ith event:

Q_i = \sum\limits_{coins} q_i

The same for all history:

P_i = \sum_{j\leq i} Q_j

Unnormalized mint rate:

U_i = U_{i-1} a^{\Delta t_i} + Q_i

Now we can determine c_i

as

c_i = f(P_i, U_i) a (\sqrt{r_i} - \sqrt{r_i - b_i}),

where f(x,y)

is emission scaling function.

If we need to get bitcoin-like behavior, set f(x,y) = e^{-\epsilon x}

. For ethereum-like behavior set f(x,y) = 1/y

.

I tend to something like ethereum, because constant per time reward is a good tradeoff, allowing us to burn all future deployed dead ICO's, but save the coin from unlimited emission.

Addition of new tokens to burn

The burning procedure is working only for listed at burner smart contract tokens because some kinds of tokens (mintable) must not be allowed to burn, also different tokens must be added with different initial parameters.

That’s why the addition of new tokens is an important procedure comparable to the soft fork of cryptocurrency.

The final model is discussed, but the current draft state is building DAO with two parts:

- developers team (need to setup the DAO)
- representatives of PYRO token holders (selected by voting for a fixed period of time)

Members can vote for adding and removing tokens from the burning list. They may be motivated to keep the PYRO token alive with bonds, locked for some years. Also, they may be motivated to vote for the addition of new tokens and get some PYRO tokens obtained by burning these new tokens.

Also, developers can burn their shares and bring true democracy to the DAO.

This is draft, more detailed model of DAO with all details of design and motivations is under discussion.

Thanks

To Roman Semenov, Alexey Levin and Alexander Khlebuschev for math model discussion

Author

Igor Gulamov

Links

You may see the live demo at[pyromania.io](pyromania.io) and source code at https://github.com/zdai-io/Pyromania-ETHParis.