A Technical Whitepaper for Reflect Contracts

Abstract

The goal is to create a token that will distribute fees to holders when a user makes a transaction. One way is to individually add the reward to all user's balance. However, such transaction could easily fail due to network gas limit. A different method is to create a deflationary mechanism so that tokens one holds are worth more.

Introduction

Over the couple of month, auto-staking (auto-farming) tokens have become a new trend in the DeFi space. Some call this the DeFi 2.0 token.

In traditional farms, users have to stake their tokens manually. For some people, this feels like a two-step process right after supplying liquidity to pools or swapping to native token. Moreover, users have to risk their funds since the staked tokens end up in MasterChef smart contract.

Reflect contracts, or RFI contracts, solve this issue by implementing an auto-staking feature built in to the token. This way, users can safely store their tokens in their wallet while still earning rewards. However, one big difference is that rewards come from transaction fees, not from MasterChef minting new tokens.

Background

Before diving into the contract, a new concept must be introduced: t-space and r-space values, along with tTotal and rTotal. tTotal, which belongs to t-space, represents tokens in circulation or total supply of a token. On the other hand, rTotal, which belongs to r-space, is a reflected value of tTotal. The term "reflected" cannot be easily explained in words but one interpretation of rTotal is token supply in reserve. Furthermore, values in t-space can easily be converted to r-space form, and vice versa using formula (3).

Stakers are users who earn passive income by holding native token. In contrast, non-stakers do not earn rewards. Router contracts, pair contracts, dev wallets are usually excluding from staking in order to fully reward users.

Defining tTotal and rTotal

$$\text{tTotal} = \text{totalSupply} \tag{1}$$

$$\text{rTotal} = \text{MAX} - (\text{MAX} \bmod \text{tTotal}) \tag{2}$$

rTotal can be further broken down as follows:

$$\begin{aligned}\mathrm{rTotal} &= \mathrm{MAX} - (\mathrm{MAX} \bmod \mathrm{tTotal}) \\ &= \mathrm{MAX} - (\mathrm{MAX} - q \cdot \mathrm{tTotal}) \\ &= q \cdot \mathrm{tTotal} \quad (1 \le q \le \mathrm{MAX})\end{aligned}$$

From the property of remainders:

$$\begin{aligned}0 \le \mathrm{MAX} \bmod \mathrm{tTotal} < \mathrm{tTotal} \\ -\mathrm{MAX} \le -q \cdot \mathrm{tTotal} < \mathrm{tTotal} - \mathrm{MAX} \\ \mathrm{MAX} - \mathrm{tTotal} < \mathrm{rTotal} \le \mathrm{MAX}\end{aligned}$$

Therefore, rTotal is a multiple of tTotal and it is between $\mathrm{MAX} - \mathrm{tTotal}$ and MAX. The value MAX can be any number above tTotal, and ~uint256(0) $\approx 10^{77}$ is usually the chosen value.

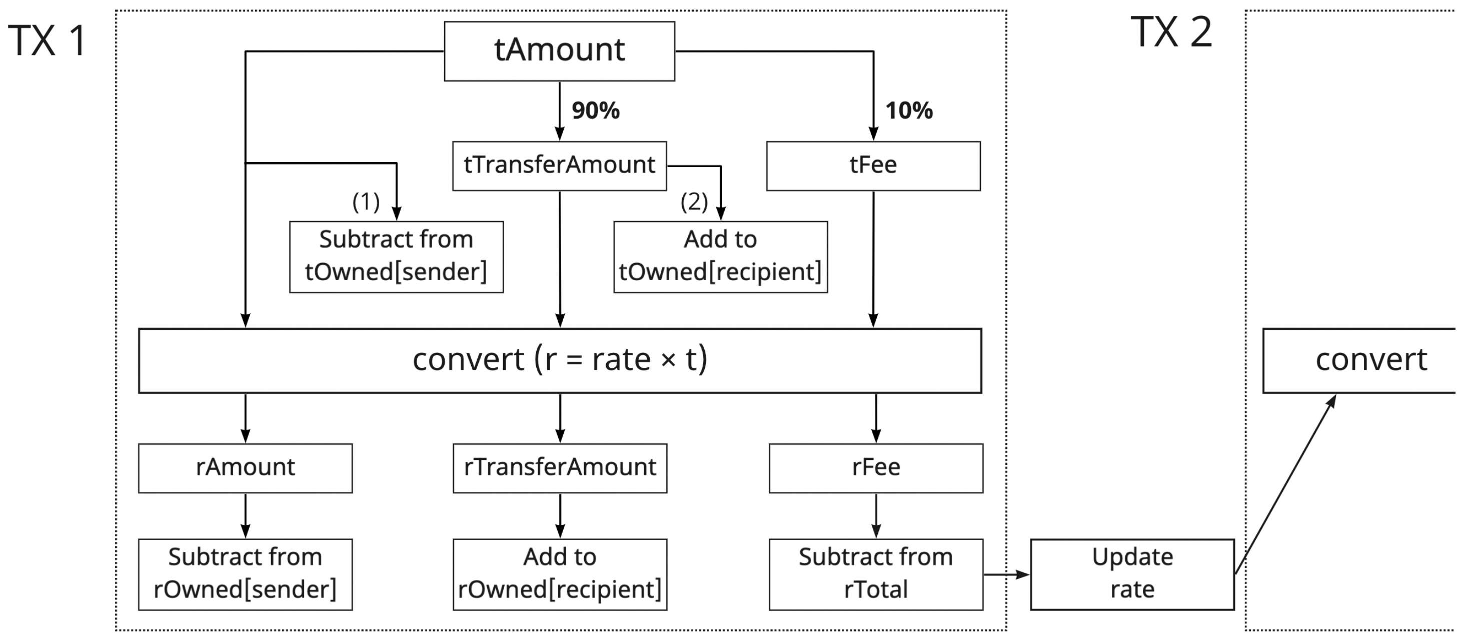
Conversion

Any t-space value (t) can be converted to r-space value (r) as follows:

$$r = q \cdot t = \frac{\mathrm{rTotal}}{\mathrm{tTotal}} \cdot t \tag{3}$$

Transaction

Transaction mechanism is illustrated in the figure below:



Note:

(1) If sender is excluded from staking

(2) If recipient is excluded from staking

Glossary:

- tAmount: token amount that sender pays/transfers including tFee
- tFee: transfer fee (note: 10% was chosen arbitrarily)
- tTransferAmount: tokens that will get transferred to recipient
- tOwned[user]: User's balance represented in t-space (only used by non-stakers)
- rOwned[user]: User's balance represented in r-space

Deflationary mechanism

As illustrated in the figure above, by decreasing rTotal, a deflationary mechanism is achieved. After the first transaction, a newly generated rate will be used for the next transaction or when viewing user balance. This makes tokens in r-space more valuable.

Relationship of rTotal in n+2, n+1, and n-th transaction:

$$\begin{aligned}
\mathrm{rTotal}(n+1) &= \mathrm{rTotal}(n) - \mathrm{rFee}(n) \\
&= \left(1 - \frac{\mathrm{tFee}(n)}{\mathrm{tTotal}}\right) \cdot \mathrm{rTotal}(n) \\
\mathrm{rTotal}(n+2) &= \mathrm{rTotal}(n+1) - \mathrm{rFee}(n+1) \\
&= \left(1 - \frac{\mathrm{tFee}(n+1)}{\mathrm{tTotal}}\right) \cdot \left(1 - \frac{\mathrm{tFee}(n)}{\mathrm{tTotal}}\right) \cdot \mathrm{rTotal}(n)
\end{aligned}$$

rTotal in n-th transaction can be calculated using the following formula:

$$\mathrm{rTotal}(n) = \prod_{i=1}^{n} \left(1 - \frac{\mathrm{tFee}(n-i)}{\mathrm{tTotal}}\right) \cdot \mathrm{rTotal}(0) \tag{4}$$

tFee is 10% of $\mathrm{tAmount}$, therefore $\forall \mathrm{tFee} \ll \mathrm{tTotal}$. This makes tokens in r-space deflationary while maintaining $\mathrm{rTotal}(\forall n) > 0$.

Avoiding non-stakers from equation

It is essential to keep non-stakers out of the equation when calculating rate. Instead of equation (3), which includes non-stakers, rate is calculated as follows:

$$\mathrm{rate} = \frac{\mathrm{rTotal} - \sum\limits_{\mathrm{user} \in \mathrm{non\text{-}stakers}} \mathrm{rOwned}[\mathrm{user}]}{\mathrm{tTotal} - \sum\limits_{\mathrm{user} \in \mathrm{non\text{-}stakers}} \mathrm{tOwned}[\mathrm{user}]} =: \frac{\mathrm{rSupply}}{\mathrm{tSupply}} \tag{5}$$

Dev note: Since this operation involves a loop, non-stakers should be no more than 10 and is recommended to store the summation part in storage variables.

Calculating balance

For stakers, balance in t-space is calculated as follows:

$$\text{balanceOf}[\text{user}] = \frac{\text{rOwned}[\text{user}]}{\text{rate}} = \frac{\text{tSupply}}{\text{rSupply}} \cdot \text{rOwned}[\text{user}] \tag{6}$$

After a transaction, a new rate will be applied:

$$\text{balanceOf}[\text{user}]' = \frac{\text{rOwned}[\text{user}]}{\text{rate}'} = \frac{\text{tSupply}}{\text{rSupply} - \text{rFee}} \cdot \text{rOwned}[\text{user}] \tag{7}$$

Therefore, the general formula for balance in n-th transaction is:

$$\text{balanceOf}[\text{user}](n) = \frac{\text{tSupply}}{\text{rSupply}(0) - \sum_{i=0}^{n-1} \text{rFee}(i)} \cdot \text{rOwned}[\text{user}] \tag{8}$$

By combining equations (4) and (8), one can prove that denominator will never go to 0.

$$\text{balanceOf}[\text{user}](n) = \frac{\text{tSupply}}{\prod_{i=1}^{n} \left(1 - \frac{\text{tFee}(n-i)}{\text{tSupply}}\right) \cdot \text{rSupply}(0)} \cdot \text{rOwned}[\text{user}] \tag{9}$$

For non-stakers, their balance is simply $\text{tOwned}[\text{user}]$.

Calculating using the first method

As mentioned in the introduction, one way to calculate user's balance is to simply add reward to their original balance:

$$\begin{aligned} \text{balanceOf}[\text{user}]' &= \text{balanceOf}[\text{user}] + \text{fee} \cdot \frac{\text{balanceOf}[\text{user}]}{\text{tSupply} - \text{fee}} \\ &= \frac{\text{tSupply}}{\text{tSupply} - \text{fee}} \cdot \text{balanceOf}[\text{user}] \end{aligned}$$

Interestingly, this equation looks similar to equation (7).
However, this method quickly becomes an issue after another transaction:

$$\begin{aligned} \text{balanceOf}[\text{user}]'' &= \frac{\text{tSupply}}{\text{tSupply} - \text{fee}''} \cdot \text{balanceOf}[\text{user}]' \\ &= \frac{\text{tSupply}}{\text{tSupply} - \text{fee}''} \cdot \frac{\text{tSupply}}{\text{tSupply} - \text{fee}'} \cdot \text{balanceOf}[\text{user}] \end{aligned}$$

One issue with RFI contracts

RFI contracts has one fundamental issue over including users into staking. The conversion rate between t-values and r-values is calculated using equation (5). Here, it is excluding non-stakers out of equation in order to fully reward the stakers. However, when a new user gets included into staking, the rate updates, thereby changing all stakers balance. Using this feature, it can be used to rug-pull rewards from stakers by including a whale.

Expanding the contract

Some projects have expanded the contract to not only reward native tokens but also reward other tokens such as BNB, BTCB, CAKE, etc. This is done by taking portion of the reward, then convert to the other token using the router contract.

However, this can lead the token price to drop since the native token is sold in every transaction.

About the author

Username: REGO350
GitHub: https://github.com/REGO350
Contact: rego350xwb02@gmail.com