Tapio V1.5

Nuts Finance

October 2023

Abstract

Tapio is an innovative DeFi protocol that proposes stableSwap pools for Liquid Staking Derivatives (LSDs) like stETH, cbETH, rETH, and many others. Liduidity providers can stake their LSDs in different pools and receive a common token tapETH maintaining a stable peg against regular ETH. They collect swap fee and continue receiving LSD staking rewards. Moreover, they can use their tapETH token in other DeFi protocols to increase their yield.

Tapio V1.5 aims at:

- Improving the tapETH token model to be more compatible with external DeFi protocols;
- Making smart contracts more flexible for future upgrades;
- Optimizing gas fee;

Contents

1 tapETH as a rebase token

1.1 Overview

In the current version Tapio V1, fee and LSD staking rewards are collected, converted to tapETH and distributed over tapETH holders. A tapETH holder has to execute a claim transaction to get his rewards. In Tapio V1.5, we want to define a rebase token model for tapETH. For this purpose, a similar stETH rebasing token model is considered.

1.2 Model

tapETH rebases happen when the user interacts with Tapio and it is always positive. Indeed, it consists in increasing the totalSupply of tapETH by the staking rewards, mint, swap and redeem fee from the different pools.
The mechanism of tapETH rebase model is based on shares. The share is a basic unit representing the tapETH holder's share in the total amount of tapETH controlled by the protocol. Shares aren't normalized, so the contract should store the sum of all shares to calculate each account's tapETH balance :

$$balances[account] = \frac{shares[account] * totalSupply}{totalShares} \tag{1}$$

where $totalSupply$ is the total Supply of tapETH that reflects the sum of amount of tapETH minted/burned from user deposits/withdrawals and collected staking rewards and fee in tapETH

For each pool i, the amount of collected staking rewards and fee in tapETH between t and $t^{'}$ is calculated as:

$$\Delta Fee_i^{t,t^{'}} = max(D_i^{t^{'}} - D_i^t, 0) \tag{2}$$

Then, the total supply of tapETH S is updated at time $t^{'}$ as follows (we assume that tapETH is not minted between t and $t^{'}$):

$$S_t^{'} = S_t + \sum_i max(D_i^{t^{'}} - D_i^t, 0) \tag{3}$$

When a new deposit/withdrawal happens, the shares get minted/burned to reflect what share of tapETH has been added/removed based on this formula:

$$shares[account] = \frac{balances[account] * totalShares}{totalSupply} \tag{4}$$

1.3 Technical specifications

The smart contract "StableAssetToken" that the represents the Token tapETH should be modified to include the share mechanism through the private mapping "shares": $mapping(address => uint256)$ and the following methods:

- getTotalShares(): returns the total amount of shares in existence.
- sharesOf(account): returns the number of shares owned by account.
- getSharesByPooledEth(tapEthAmount): returns the number of shares that corresponds to tapEthAmount of the protocol-controlled tapEth.
- getPooledEthByShares(sharesAmount): returns the amount of tapETH that corresponds to sharesAmount token shares.
- transferShares(recipient, sharesAmount): moves token shares from the caller's account to the provided recipient account.
- transferSharesFrom(sender, recipient, sharesAmount): moves sharesAmount token shares from the sender account to the recipient using the allowance mechanism.
- mintShares(recipient, sharesAmount): mint sharesAmount token shares for recipient.
- burnShares(account, sharesAmount): burn sharesAmount token shares from account.

The smart contract "StableAsset" should call the functions "mintShares" and "burnShares" inside each "mint" and "redeem functions, respectively.

2 WtapEth

2.1 Overview

Now, due to the rebasing nature of tapETH, the tapETH balance on the holder's address is not constant, it changes dynamically as staking and fee comes in. Although rebasable tokens are becoming a common thing in DeFi recently, many dApps do not support rebasing such as Maker, UniSwap, and SushiSwap are not designed for rebasable tokens. Listing tapETH on these dApps can result in holders not receiving their daily staking rewards which effectively defeats the benefits of tapETH. To integrate with such dApps, WtapETH (wrapped tapETH) is introduced. WtapETH is mainly used as a layer of compatibility to integrate WtapETH into other DeFi protocols, that do not support rebasable tokens, especially bridges to L2s and other chains, as rebases don't work for bridged assets by default. WtapETH balance is constant, while the token increases in value eventually (denominated in tapETH). At any time, any amount of tapETH can be converted to WtapETH via a trustless wrapper and vice versa

2.2 Model

WtapETH is an ERC20 token that represents the account's share of the tapETH total supply (tapETH token wrapper with static balances: 1 wei in shares equals to 1 wei in balance). WtapETH balance does not rebase, it can only be changed upon transfers, minting, and burning. At any time, anyone holding WtapETH can convert any amount of it to tapETH at a fixed rate, and vice versa. The rate is the same for everyone at any given moment. The rate gets updated once a day, when tapETH

undergoes a rebase. When wrapping tapETH to WtapETH, the desired amount of tapETH is locked on the WtapETH contract balance, and the WtapETH is minted according to the share formula (2). When unwrapping, WtapETH gets burnt and the corresponding amount of tapETH gets unlocked. Thus, the amount of tapETH unlocked when unwrapping is different from what has been initially wrapped (given a rebase happened between wrapping and unwrapping WtapETH).

2.3 Technical specifications

We need to implement the smart contract WtapETH as ERC-20 value-accruing token wrapper for tapETH. Its balance does not change with LSD staking rewards, but its value in tapETH does. Internally, it represents the user's share of the total supply of tapETH tokens.

The contract WtapETH implements the following Ethereum standards:

- ERC-20: Token Standard.
- ERC-2612: Permit Extension for ERC-20 Signed Approvals.
- EIP-712: Typed structured data hashing and signing.

The contract WtapETH has the following main functions:

- wrap(tapETHAmount): exchanges tapETH to WtapETH. tapETHAmount of tapETH token should be transferred to the contract and the corresponding shares amount (using the function "getSharesByPooledEth(tapEthAmount)") should be minted in WtapETH.
- unwrap(wtapETHAmount): exchanges WtapETH to tapETH. wtapETHAmount of WtapETH token should be burned and the corresponding tapETH amount (using the function "getPooledEthByShares(wtapETHAmount)") should be sent to the caller.
- receive(): Shortcut to stake ETH and auto-wrap returned tapETH. Indeed, The user can send ETH with regular transfer to the address of the contract and get WtapETH in return. The contract will send ETH to the most unbalanced pool, staking it and wrapping the received tapETH seamlessly under the hood (this feature was not implemented)

The contract WtapETH proposes the following View methods:

- getWtapETHByTapETH(): returns amount of WtapETH for a given amount of tapETH.
- getTapETHBytWtapET(): returns amount of tapETH for a given amount of WtapETH.
- tapETHPerToken(): returns the amount of tapETH tokens corresponding to one wtapETH.
- tokensPertapETH(): returns the amount of WtapETH tokens corresponding to one tapETH.

3 Smart contracts Design

From Tapio V1, we keep the smart contracts "StableAsset" and "StableAssetApplication" optimizing the gas fee as suggested in the internal review document. Besides, the smart contract StableAsset should integrate the share mechanism as proposed in (1.3).

The smart contract "AssetStableToken" ("TapEth") should implement the share mechanism as explained in (1.3). The new smart contract "WtapEth" should be implemented as explained in (2.3) and connected to the smart contract "TapEth".

All smart contracts are upgradeable.

4 tapETH depeg mitigation

let's X be a discount on the staking rewards and swap fee to be set by the governance in order to reduce tapETH/ETH depeg.

let's S_t be the total supply at time t, $\Delta Fee_{t,t'}$ is the collected staking rewards and swap fee between t and $t^{'}$.

The rebasing function consists in increasing the total supply of tapETH by the staking rewards and swap fee (in tapETH):

$$S_{t'} = S_t + \Delta Fee_{t,t'} \tag{5}$$

Considering the discount, the rebasing function becomes:

$$S_{t'} = S_t + (1 - X)\Delta Fee_{t,t'} \tag{6}$$

We assume that the last rebasing was done at time t.

A second version aims at periodically controlling and adjusting the tapETH depeg defined as:

$$depeg_t = max(S_t - \sum_i D_i(t), 0) \tag{7}$$

let's $depeg_t$ be the depeg tapETH/ETH at time t. At the rebasing time $t^{'}$, the discount $DISC_{t'}$ in tapETH is calculated as:

$$DISC_{t'} = min(depeg_{t'}, \Delta Fee_{t,t'}) \tag{8}$$

and the rebasing function is given by:

$$S_{t'} = S_t + \Delta Fee_{t,t'} - DISC_{t'} \tag{9}$$