Tokenized Deflationary Mutual Fund Based on Synthetic Reserves - Vessel Protocol V1.0

Amir Kashapov, Phillip Zlatkovic, Elmore Shaw, Grady Tucker

December 24, 2021

Abstract

Vessel Protocol aims to bring one of the most proven utilities on traditional markets, mutual funds, onto the decentralized web. We present a novel approach to how we have tackled this challenge in this paper - a DeFi utility token akin to an ETF for which the value is coupled to the assets which are encapsulated within the decentralized mutual fund our protocol provides. The fund itself is a trustless DAO, whereby user votes from one epoch to the next shape how constituent assets within it evolve using a voting algorithm we elaborate upon in this paper. Most ETF tokens work by ”wrapping” a set of reserve assets, however, our protocol emulates the price action of the assets with the help of token reserves and token burns, thereby rendering the ”reserve assets” in our prior comparison synthetic. In order to incentivize decentralization, there is an established bounty scheme within the protocol to ensure fairness and to encourage timely protocol re-stablization. A guaranteed token burn establishes a locked deflationary mechanic, and over time the token supply is guaranteed to diminish with volume. Thus, overall, the aforementioned combines the established advantages of diversification in mutual funds with the benefits of decentralization and deflationary mechanics and removes the element of ”opaque” management over how ETFs are shaped or how they evolve over time.

Contents

1 Introduction

Mutual funds, index funds, and ETFs are all traditional finance (tradFi) instruments for investors to allocate their money into a diversified basket of goods, bonds, securities, etc., in a cost-effective, sensible way that may be difficult to individually recreate asset by asset. They proliferate, and are advantageous, because they diversify and balance out risk, are a way of passively investing, and because, on average, an aggregate or an index is likely to outperform the market in comparison to an individual asset - a good example would be the S&P500. The turbulent evolution of the cryptocurrency world, accelerated in part thanks to powerful Turing-Complete languages such as Ethereum's Solidity, brought about thousands of new coins and new functionalities such as DAO, lending, yield farming, staking, NFTs, with many myriad potential applications in the world of DeFi still yet to be discovered. This makes considerations of the advantages offered by mutual funds, index funds, and ETFs sensible in the context of the evolving cryptocurrency market. Given that thousands of coins are now created per year, it could be daunting to keep track of the market, and thus, a single token with the functionality of a well-managed mutual fund would fulfill the purpose of a low-risk diversified investment into a pool of assets that can be thought of as a passive investment into the market or an asset class as a whole. Of course, however, we would not be the first to make this consideration or suggestion. Our proposed token differs in a number of key ways from alternative solutions currently available. First and foremost, our token serves to create a truly decentralized mutual fund with governance votes and DAO over how the fund evolves over epochs as opposed to maintaining reserves controlled by a centralized party in order to influence price. Another point of differentiation is that our token leverages the power of strong deflationary mechanics due to how we have implemented our solution without foregoing the advantages of a traditional mutual fund. Lastly, in contrast with how traditional mutual funds function, we leverage utilities known as oracles to give us price feeds and create a synthetic reserve pool of assets to hedge the risk of the mutual fund and to "cushion" a market down-trend in addition to accelerating the growth of the mutual fund during a market uptrend in a manner we will elaborate upon shortly.

2 A Brief Description of Our Solution

Our implementation of the Vessel Protocol in its current state to create a decentralized mutual fund that evolves with the help of a DAO is a system that couples the evolution of the price of the Vessel token with the evolution of the price of the mutual fund. This coupling is achieved through a less traditional route. Whilst traditional implementations of stabilized assets rely on collateralization in order to work, usually with the help of some set of reserve assets, those implementations rely on altering demand in some manner to achieve a desired change in the price of an asset. We have implemented an alternative implementation with equally valid economics from a theory standpoint by instead altering the supply of the token as opposed to the demand of the token. It is therefore the case that should the value of our tokenized mutual fund be desired to increase, the circulating supply of tokens must decrease and conversely should the value of our tokenized mutual fund be desired to decrease the circulating supply of tokens must increase. In order to achieve this, we have implemented the Vessel protocol to operate under the assumption that any time, price elasticity of supply is unitary elastic, meaning to say that an alteration in the supply yields an equivalent alteration in price of the token. Unitary elastic supply is an ideal economic scenario, and thus it stands to reason to implement our system with it as our core.

Our token aims to mirror an underlying mutual fund's price action, however the underlying mutual fund itself is synthetic and does not exist, as we merely rely on the price action of the fund's sub-components and do not need to hold reserves of said sub-components. As we have mentioned, we alter the supply of our tokenized synthetic mutual fund, and we do so by segregating the supply of Vessel tokens into a reserve vault of Vessel tokens, a burn vault of Vessel tokens, and the user supply of Vessel tokens. At the end of a given epoch, reserve vault tokens are either minted into the circulating supply available to users or burned, however, we set thresholds for the maximum and minimum size of each of the above three components. As a result of these enforced limits, our system can only be considered as loosely coupled, however it allows us to put a strong deflationary mechanic in place and the thresholds guarantee that the protocol is operational within a set of bounds at any given time so as to make fluctuations more stable. In the upcoming sections, we will be much more specific and provide mathematical backing for our reasoning.

3 Leveraging Oracles As a Primary Utility

Prior to discussing the rebase protocol in the next sections, one thing to note is that in order for it to operate and make decisions, it would require to know the individual prices of the underlying assets in a decentralized fashion at separate instances in time. Since this is contrasted with traditional mutual funds where hedge fund managers are able to fulfill this role or from a standard computer program accessing an index or aggregate index on the internet, an alternative is required to provide a trustless, secure, unbiased, and reliable price feed at any point in time. Since this is the primary purpose of an oracle, it is the utility we leverage. It is easiest to illustrate how this works with a generalized example demonstrated in Figure 1.

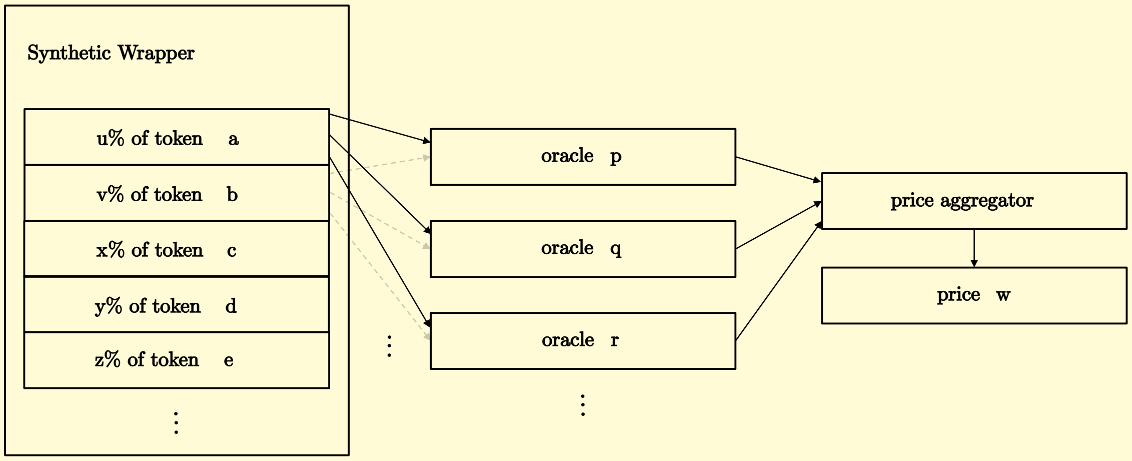

Figure 1: Calculating wrapper value w

Assume our synthetic mutual fund wrapper artificially encapsulates 5 tokens a, b, c, d, and e, present in percentages of u%, v%, x%, y%, and z%, respectively at the start of an epoch. At the end of an epoch, the percentages become u'%, v'%, x'%, y'%, and z'%, respectively. As the figure above illustrates, at any given point in time, the price w of a wrapper can be derived by individual queries passed for each token to a set of oracles, followed by an aggregation and averaging of prices dependent on the percentages in which the tokens are present in the original wrapper. The above description serves to provide some intuition, as we next provide a formal formula for the change in the value of the token and the value of the mutual fund, which we term the wrapper, prices next. The change in the token value and in the wrapper value will be used by the protocol to make a decision with regards to how to alter the supply at the end of any given epoch.

4 Inter-Epoch Changes in Token & Wrapper Prices

Definitions:

t := the price of the Vessel token
w := the price of the Vessel wrapper
w_i := the i'th coin inside the Vessel wrapper
' := a designation of the current epoch as opposed to the prior
address($this$) := the Vessel token address queried for price
coinAddres[[i] := the i'th coin's address inside the wrapper queried for price
balancedRatio[i] := the percentage weight of the i'th coin in the wrapper

Formulas:

$$\Delta t = \frac{tokenPrice'(address(this)) - tokenPrice(address(this))}{tokenPrice(address(this))}$$

$$\Delta w_i = balancedRatio[i] \times \frac{tokenPrice'(coinAddress[i]) - tokenPrice(coinAddress[i])}{tokenPrice(coinAddress[i])}$$

$$\Delta w = \sum_{i=0}^{n-1} \Delta w_i$$

5 Different Protocol Scenarios & Economics

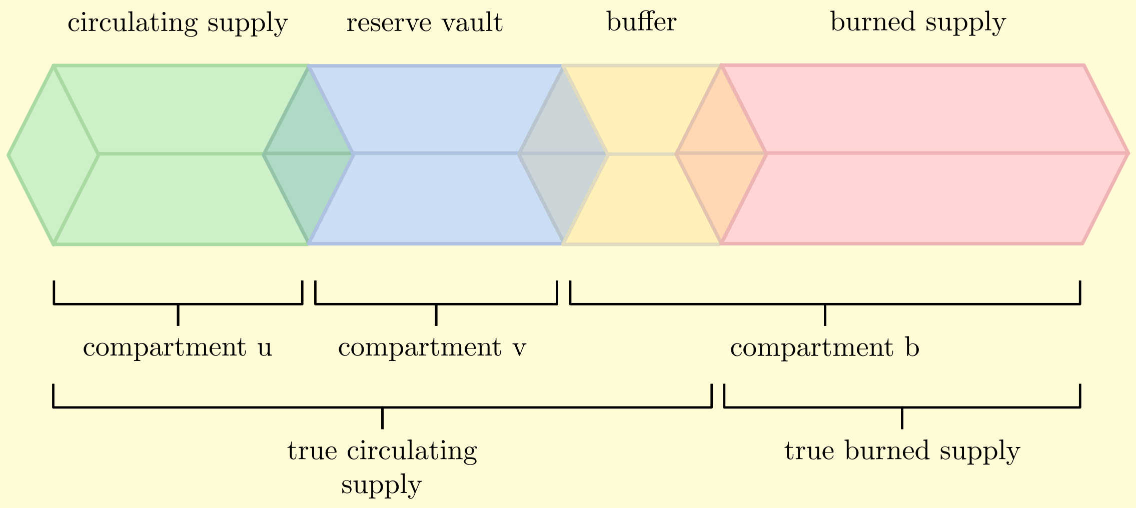

Figure 2: Vessel Supply

Before proceeding, we discuss the individual portions of the vessel supply in more detail. The circulating supply available to users, u, and the size of the reserve vault of Vessel tokens, v, are initially at a ratio of 1:1. This is the maximum ratio between the two, and it ranges from 1:0 to 1:1, representative of an empty reserves vault and a full reserves vault respectively. Those thresholds are selected to maintain protocol stability, as we have priorly mentioned. We would also like to specify that our burn wallet has a forever inaccessible and accessible portion, the latter of which we can consider is "buffering" but from which tokens can be recovered in the case tokens need to be minted into the circulating supply.

Another thing we should note is that every transaction that takes place is taxed, this provides a deflationary pressure by discouraging sellers, however, it primarily serves to reward holders through reflections (elaborated upon in the next section) and to automatically provide locked liquidity to a decentralized exchange.

Definitions:

∂ := the ideal number of tokens to alter the supply by at the conclusion of an epoch (without factoring guards [thresholds] in)

θ := the threshold ratio of the vault, utilized to affect the change in supply in order to dilute or diminish it - the threshold ratio of the vault is the maximum allowable ratio of the vault that can be affected

u := the circulating supply available to users

Scenarios on epoch end:

Case 1: $(\Delta t - \Delta w) < 0 \implies$ token supply is ideally to be diminished by ∂
where $\partial = \text{min}(\theta u, (\Delta w - \Delta t)u)$

Case 2: $(\Delta t - \Delta w) > 0 \implies$ token supply is ideally to be diluted by ∂
where $\partial = \text{min}(\theta u, (\Delta t - \Delta w)u)$

Case 3: $\Delta t = \Delta w \implies$ no change in token supply

More Definitions:

v := the reserve supply inside the vault
b := the supply of the burn wallet
t := the total supply
f := the forever burnt (inaccessible) portion of the burn wallet
x := theoretical maximum supply dilution
y := maximum supply dilution to ensure an ever-increasing vault size

Invariants & deduction to arrive at thresholds for ∂ in cases 1 and 2:

$$u + v + b = t \text{ [constant]} \tag{1}$$

$$u \geq v^{*} \tag{2}$$

$$b \geq u - v \tag{3}$$

$$b = t - u - v \text{ [from (1)]} \tag{4}$$

$$t - u - v \geq u - v \text{ [from (3) and (4)]} \tag{5}$$

$$u \leq t/2 \text{ [from (5)]} \tag{6}$$

$$x = t/2 - u \text{ [to maintain (6)]} \tag{7}$$

$$x = \frac{b + v - u}{2} \tag{8}$$

$$f = \frac{1}{4}\Sigma \text{ (transaction taxes)} \tag{9}$$

$$y = x - f \tag{10}$$

We now revisit the protocol scenarios with stricter guards (thresholds) for the value of ∂ in cases 1 and 2 to maintain the above invariants.

Scenarios with guards (thresholds) in place:

Case 1: supply is to be diminished

- token supply is ideally to be diminished by ∂
- $v = v - min(\partial, v)$ [set a guard]
- $b = b + min(\partial, v)$
- thus, conduct a transfer of $min(\partial, v)$ tokens from the v compartment to the b compartment

Case 2: supply is to be diluted - token supply to be ideally diluted by ∂ tokens

Case 2a: $\partial \leq u - v$

- set $v = v + \partial$
- set $b = b - \partial$
- thus, conduct a transfer of ∂ tokens from the b compartment to the v compartment

Case 2b: $\partial > u - v$

- $\partial = \partial_1 + \partial_2$

[*] max ratio of u:v is 1:1; if not, cut off excess in v to b so that $u = v$

- $\partial_1 = u - v$
- $\partial_2 = \partial - (u - v) = \partial + v - u$
- set $v = v + \partial_1$
- set $b = b - \partial_1$
- thus, conduct a transfer of ∂_1 tokens from the b compartment to the v compartment
- iff $y > 0$
  - set $b = b - min(\partial_2, y)$
  - set $u = u + min(\partial_2, y)$
  - thus, conduct a release of $min(\partial_2, y)$ tokens to the circulating supply [through reflections, elaborated on soon]

6 Background on Reflection Contracts

In reflection contracts, the balance of a wallet is dynamic, whereby the balance is the number of tokens a user holds. This is done in order to allow users to automatically stake their tokens simply by holding and to earn "reflected" rewards.

More definitions:

t_{supply} := the unchanging, total, traditional supply of the coin
r_{supply} := the diminishing token supply in [†]r-space

Formulas relating to r-space supply and wallet balance:

$$r_{supply} = r_{supply_0} - \sum (\text{taxes incurred by all user transactions}) \tag{11}$$

$$balanceOf(wallet) = \frac{r_{owned}[wallet] \times t_{supply}}{r_{supply}} \tag{12}$$

We can see in the above that, since the balance of a wallet at any given time is computed dynamically using equation 12, and since r_{supply} is decreasing with each user transaction as per equation 11 whilst t_{supply} is constant by definition, the balance of a wallet that does not transact itself is non-decreasing, since the user balance is stored in r-space for staking which then gets converted to the traditional t-space.

7 How Vessel Protocol "Mints" Supply to Users

Let us consider how the rebalance protocol can release $min(\partial_2, y)$ [in Case 2b iff $y > 0$] tokens into the circulating supply (in t-space) to users from the "buffering" (accessible) portion of the burn wallet. In such a scenario, we intend to reflect/siphon tokens from the burn wallet strictly to the user wallets, thus avoiding reflections to the reserves vault, to the contract, and "self-reflections" to the burn wallet.

[†]The purpose of the r-space will soon be clear. All user balances are actually stored in the r-space in the r_{owned} array and are then converted to the t-space

Let ' designate the current epoch as opposed to the prior; in particular, we are interested in how to vary particular r-space values so that $reflections$ in the t-space are transferred from the burn wallet proportionately to users whilst the t-space values of the vault and contract wallets are unchanged and the burn wallet's t-space value is diminished by $reflections$.

Given:

$r_{supply} = r_{contract} + r_{users} + r_{vault} + r_{burn}$
$r'_{supply} = r'_{contract} + r_{users} + r'_{vault} + r'_{burn}$ [r_{users} does not change post reflection]
$r_{users} = r_{supply} - r_{contract} - r_{vault} - r_{burn}$
$r_{users} = r'_{supply} - r'_{contract} - r'_{vault} - r'_{burn}$
$r_{contract}/r_{supply} = r'_{contract}/r'_{supply}$
$r_{vault}/r_{supply} = r'_{vault}/r'_{supply}$

[from the balanceOf formula, the t value inside the vault should not change]

$$r_{burn} \times \frac{t_{supply}}{r_{supply}} = r'_{burn} \times \frac{t_{supply}}{r'_{supply}} + reflections$$

[the amount reflected is a t value, which when added to the new burn wallet's t value should equate to the old burn wallet's t value]

$$\frac{r_{burn}}{r_{supply}} = \frac{r'_{burn}}{r'_{supply}} + \frac{reflections}{t_{supply}}$$

[divide the above by t_{supply}]

Note that obtaining r'_{supply} would allow us to easily calculate $r'_{contract}$, r'_{value}, and r_{burn}'.

Deduction:

Reducing the number of unknowns to obtain r'_{supply}:

$$r_{supply} - r_{contract} - r_{vault} - r_{burn} = r'_{supply} - r'_{contract} - r'_{vault} - r'_{burn}$$

[from eqs for r_{supply} and r'_{supply}, r_{users} is unaltered so both sides are equal to r_{users} and to each other]

$$r_{supply} - r_{contract} - r_{vault} - r_{burn} = r'_{supply} - \left(\frac{r_{contract}}{r_{supply}}\right) r'_{supply} - r'_{vault} - r'_{burn}$$

[substitute for $r'_{contract}$]

$$r_{supply} - r_{contract} - r_{vault} - r_{burn} = \left(1 - \frac{r_{contract}}{r_{supply}}\right) r'_{supply} - r'_{vault} - r'_{burn}$$

[take common r'_{supply} factor out]

$$r_{supply} - r_{contract} - r_{vault} - r_{burn} = \left(1 - \frac{r_{contract}}{r_{supply}}\right) r'_{supply} - \left(\frac{r_{vault}}{r_{supply}}\right) r'_{supply} - r'_{burn}$$

[substitute for r'_{vault}]

$$\frac{r'_{burn}}{r'_{supply}} = \frac{r_{burn}}{r_{supply}} - \frac{reflections}{t_{supply}}$$

[from the given]

$$r'_{burn} = \left(\frac{r_{burn}}{r_{supply}} - \frac{reflections}{t_{supply}}\right) r'_{supply}$$

$$r_{supply} - r_{contract} - r_{vault} - r_{burn} = (1 - \frac{r_{contract}}{r_{supply}} - \frac{r_{vault}}{r_{supply}}) r'_{supply} - \left(\frac{r_{burn}}{r_{supply}} - \frac{reflections}{t_{supply}}\right) r'_{supply}$$

$$r_{supply} - r_{contract} - r_{vault} - r_{burn} = (1 - \frac{r_{contract}}{r_{supply}} - \frac{r_{vault}}{r_{supply}} - \frac{r_{burn}}{r_{supply}} + \frac{reflections}{t_{supply}}) r'_{supply}$$

Results:

$$r'_{supply} = \frac{r_{supply} - r_{contract} - r_{vault} - r_{burn}}{1 - r_{contract}/r_{supply} - r_{vault}/r_{supply} - r_{burn}/r_{supply} + reflections/t_{supply}}$$

$$r'_{contract} = \left(\frac{r_{contract}}{r_{supply}}\right) r'_{supply}$$

$$r'_{vault} = \left(\frac{r_{vault}}{r_{supply}}\right) r'_{supply}$$

$$r'_{burn} = \left(\frac{r_{burn}}{r_{supply}} - \frac{reflections}{t_{supply}}\right) r'_{supply}$$

Thus, by altering the above four variables specified in "results", $reflections$ are transferred, as we have originally desired, from the burn wallet strictly to users, in proportion to how many tokens they hold.

8 DAO Voting System

We now turn our attention towards the DAO that governs the evolution of the synthetic fund. The fundamental components of the DAO include the individual vote, the vote tally at the end of an epoch, and a careful update of appropriate parameters. Within the synthetic fund, there will, at first, be 20 coins users will be able to choose from at any time. As such, there will be an associated $coinAddresses[20]$ variable which houses the contract addresses of these tokens. The reason they are stored in a variable and not hardcoded is because the Vessel team reserves the right to change them at any time, a necessary element of centralized control in the event that one of them gets changed or a token gets exploited or the community requests a change. They will initially house "blue chip" coins. Another variable is $coinVotes[20]$, which specifies how many votes have been cast during an epoch for each of the twenty coins in a given epoch up to that point, whereby votes get annulled at the end of each epoch. In a given epoch, we thus have that the total numbers of votes cast is given by:

$$totalVotesCast = \sum_{i=0}^{n-1} coinVotes[i] \quad \text{[n=20]}$$

With regards to an individual vote, we have that individual voting must always assign ratios to each of the 20 tokens in the synthetic fund with regards to how many votes are cast towards them. An individual voting will have the last epoch they voted in recorded, whereby the next time they vote the current epoch must be greater than the last epoch in which they voted. We thus set a condition whereby:

$$\forall i \mid 0 \leq userRatio[i] \leq 1 \text{ and } \sum_{i=0}^{n-1} userRatio[i] = 1.0000000000$$

(zeros to demonstrate precision)

The maximum number of votes that can be cast by any given wallet at any given time is calculated in the prior epoch as maxVotes, so that an individual vote associated with one wallet is capped at 1/1000th of the total voting power. To enforce the above, we thus have a necessary condition when we update the number of votes allocated to all tokens after any user casts their votes as:

$$\forall i \mid coinVotes[i] = coinVotes[i] + userRatio[i] \times min(maxVotes, balanceOf(msg.sender)$$

where $userRatio[i]$ is the ratio of user votes allocated to coin i, and $balanceOf(msg.sender)$ is how many Vessel tokens the individual voting wallet holds.

We now turn our attention to how votes are tallied. We have chosen to ignore coins that have had less than 5% of all votes cast towards them. $totalVotesCast$ is saved upon epoch end into a $lastVotesCast$ variable, because we want to have a smooth transition from one epoch to the next, whereby we factor in how many votes were cast last epoch and towards which coins in addition to how many votes were cast this epoch and towards which coins. We thus set up an $imbalancedRatios[20]$ array, which will hold the ratios of all coins within the synthetic wrapper prior to the discarding of those that do not surpass the 5% minimum threshold. We will also thus have a $balancedRatios[20]$ array, with the aforementioned discardings taken into account. On the next page, we describe the vote tally algorithm sequence, which also performs some setup necessary for the next epoch.

Vote tally algorithm:

- $\forall i \mid imbalancedRatio[i] = \frac{lastVotesCast}{lastVotesCast+totalVotesCast} \times balancedRatio[i] + \frac{totalVotesCast}{lastVotesCast+totalVotesCast} \times \frac{coinVotes[i]}{totalVotesCast}$
- $\forall i \mid imbalancedRatio[i] = imbalancedRatio[i] < 0.05 \; ? \; 0 : imbalancedRatio[i]$
- $ratioSum = \sum_{i=0}^{n-1} imbalancedRatio[i]$
- $\forall i \mid balancedRatio[i] = \frac{imbalancedRatio[i]}{ratioSum}$

For the next epoch also perform the following:

- compute $maxVotes$ as $\frac{r_{users} \times t_{supply}}{1000 \times r_{supply}}$ where $r_{users} = r_{supply} - r_{contract} - r_{vault} - r_{burn}$
- update $lastVotesCast = totalVotesCast$
- $\forall i \mid coinVotes[i] = 0$
- update $totalVotesCast$ as 0

9 The Decentralized Rebalancing Bounty Incentivization Scheme

A prominent problem that commonly arises with smart contracts that need to operate or trigger some function calls at specific times is that these functions need to be called externally, as one cannot, for instance, create an EVM-compatible contract such that a function is programmed to automatically be called 7 days in advance. There are two popular solutions to this problem on the Ethereum network - Chainlink's Keepers and the Ethereum Alarm Clock. Our contract is EVM-compatible, but in order to avoid relying on an external utility, Vessel has created a decentralized rebalancing bounty incentivization scheme in the form of Vessel rewards that are transferred to the first wallet that successfully triggers an epoch update first after a given epoch expires.

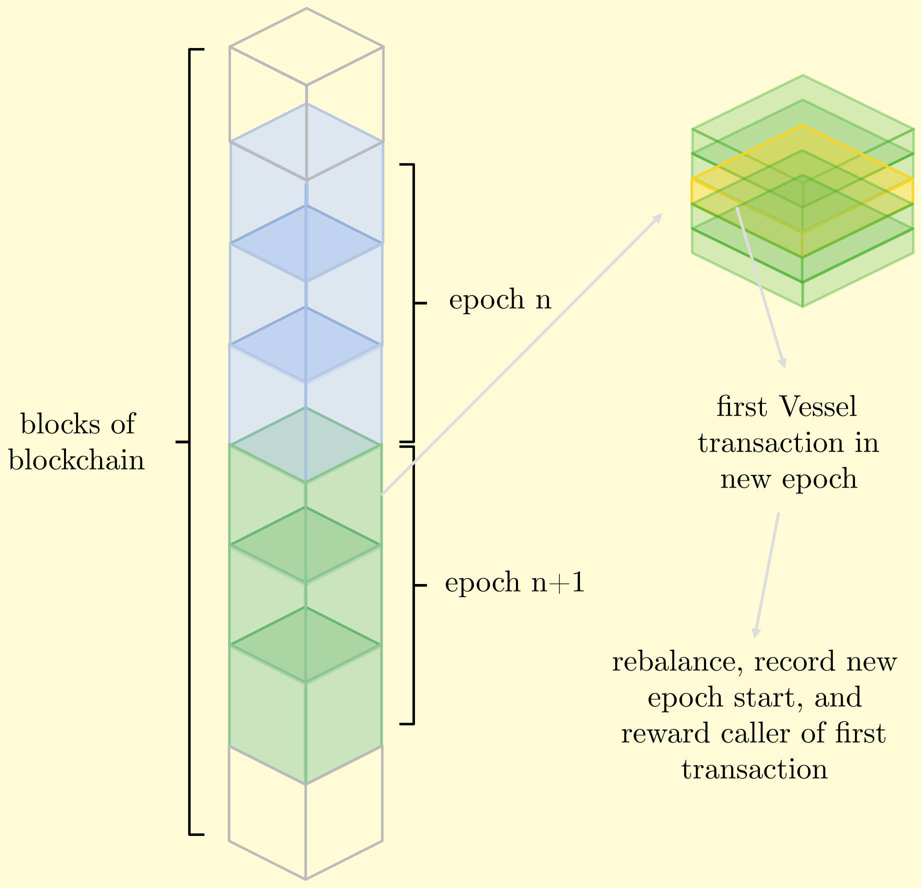
blocks of blockchain
epoch n
epoch n+1
first Vessel transaction in new epoch
rebalance, record new epoch start, and reward caller of first transaction

Figure 3: Bounty Scheme

In Figure 3, cubes represent blocks on the blockchain. Different colors represent blocks belonging to different epochs. When it is time to activate the rebalance function and move onto the next epoch, an automated call by Vessel will be triggered with an event listener. However, any individual wallet which first calls the rebalance function will be rewarded with 0.01% of Vessel supply to incentivize users activating the rebalancing mechanism as early as possible when it is time to rebalance the next epoch. We once again emphasize that this is done in lieu of the absent time node oracle mechanisms on certain chains to ensure the decentralization of the rebalancing mechanism and to prevent abuse and manipulation via centralized control of the rebalancing function by the Vessel developers, in addition to preventing an instance in which an outage, say, would disallow Vessel developers to trigger the method call themselves. Initially, 2.5% of the total supply will be placed into a bounty wallet to award all rebalancers, however it is subject to reflections and as such will grow with protocol use. In the event it is depleted the Vessel team itself will replenish it with their own funds. This is because Vessel cannot guarantee the scalability of the approach albeit it should last for an absolute minimum of 5 years - due to the fact that epochs are currently set to be 7 days in length and the one-time bounty is set to 0.01% of the Vessel supply. The alternative to the above wallet being refilled by the Vessel developers is to utilize a solution such as migrating to Chainlink keepers when they become available on the network on which Vessel is deployed, a consideration Vessel developers will always keep in mind.

10 The Necessary Elements of Centralized Control Reserved by Vessel

There are certain elements of centralized control that the Vessel team reserves. This is an unavoidable aspect of the protocol that needs to exist due to security, scalability, and flexibility reasons. One advantage of this is that it will allow the Vessel developers to be constantly engaged and involved with the community. The ability to change the coin addresses available for users to select for the synthetic vault of assets, as we have mentioned formerly, is under the Vessel team's control for security and scalability reasons whereby we will be able to update them down the line upon requests from our users or if a coin is exploited. Vessel developers will also be able to change the epoch length as well as reset the current epoch and votes in conjunction with the above. Lastly, Vessel also maintains control over the θ threshold of the vault of reserves and the ability to reset θ itself to maintain vault stability.

11 Plans for Vessel Protocol V2.0

We have already started planning and developing Vessel Protocol V2.0. The most apposite extension we would like to implement first and foremost is stronger coupling. This is because whilst our protocol is highly deflationary and we have established a pegged system between the fund and the value of the Vessel token, protocols such as tomb.finance on the Fantom network have inspired us to consider an implementation making use of Seigniorage. Seigniorage would allow the value of the Vessel token to more naturally react to the synthetic fund of reserves in a fashion far less discrete than our current implementation in addition to having the Vessel token become very strongly coupled to the value of the underlying fund at any time, and we are thus very adamant about implementing it.

Another crucial extension we would like to address, ideally in our V2.0 release, is the ability to stake funds in individual asset classes yielding variable rewards based on the performance of the asset class. This is a highly complicated issue due to the synthetic nature of the fund, and thus, whilst we have formulas and ideas on hand, a great deal of work remains. We have every confidence the above two will be achieved, and, together will allow implementing an API-like "Mutual Fund/ETF as-a-service" model, which is a concise summary of what our end goal for the V2.0 release will be.

12 Plans for Vessel Protocol V3.0

Due to the synthetic nature of the mutual fund, we are considering a release of, in conjunction with the above features, the Vessel DEX, whereby the Vessel token would serve as a utility and governance token akin to Uniswap's. The inherent utility of the Vessel token would remain, however, in addition to the ability of users voting on the governance of DEX policies and on the evolution of the synthetic fund, the liquidity of the DEX itself will additionally be utilized to mould the constituent components and ratios of the synthetic fund. This would thus make the synthetic fund representative of the health of the most prevalent tokens on the DEX in addition to the user demand over how the fund should be shaped.