Tectonic: All-in-one DeFi Protocol

Erik Lie

January 20, 2021
v1.0.0

Abstract

Decentralized Finance (DeFi) emerging on Ethereum is showing promising results toward the creation of a disintermediated financial system. In this paper, we describe a new DeFi system design which is able to integrate token swapping, money market and stablecoins into one single, unified system. We believe that the new design increases capital efficiency compared to other DeFi systems exist today.

1 Introduction

Decentralized Finance (DeFi) has become a trending topic in the cryptocurrency and blockchain space recently. DeFi enables the creation of an open financial system that has limited or no involvement from centralized institutions. By doing so, they intend to reduce transaction costs, broaden financial inclusion, empower open access, encourage permissionless innovation, and create new business opportunities. [7]

Major DeFi systems can be classified per below [1]:

- Token swapper: Uniswap [3], Balancer [16], Curve [9]
- Money market: Compound [14], Aave [1]
- Stablecoin: MakerDAO [2]

In this article, we propose the use of balance sheet and accounting ratios as tools to devise a DeFi system. The accounting ratios will be used to monitor the financial healthiness of the system and determine proper incentives for actors who may bring the system back to equilibrium. The new design enables capital to be utilized for different purposes, hence improving capital efficiency compared to other DeFi systems exist today.

Note: For this version, only details of token swapper (AMM) and governance token will be discussed.

[1] A similar but slightly different classification can be found in [21].

2 System Overview

The system can be described as *Actors* performing actions against *Pools*.

2.1 Actors

Actors can be classified into 5 groups:

Liquidity Providers. They provide capital to the system, expecting rewards (i.e.: dividend) to be paid upon withdrawal.

Casual Users. They perform actions (e.g.: swap, borrow) on the system. They can access the system services with fees.

Arbitrageurs. They are paid for helping the system to restore equilibrium via swapping.

Stakers. They are governance token holders who stake their governance tokens, expecting rewards (i.e.: system surplus) to be paid upon unstake.

Keepers. They are paid for helping the system to liquidate bad loans.

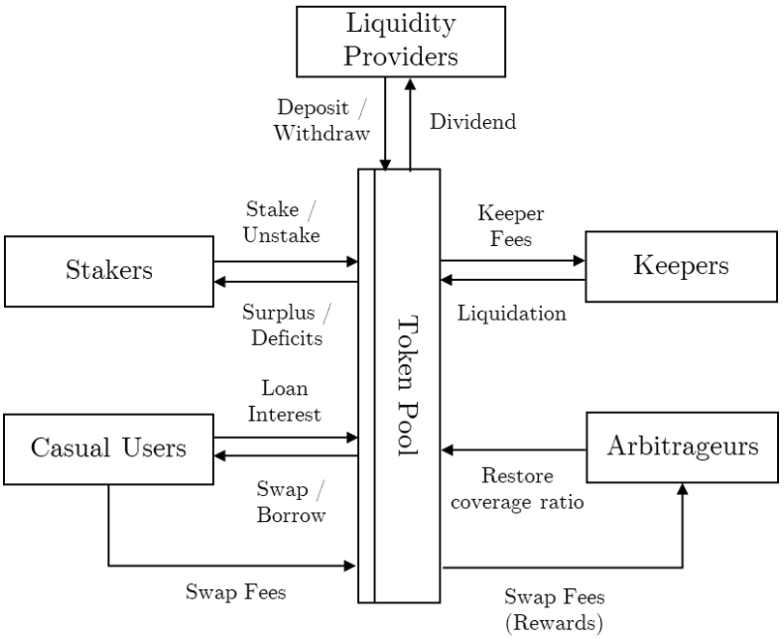


Figure 1: Actors and Pools.

2.2 Pools

Token Pools. They hold the capital provided by liquidity providers. Each token pool represents the balance sheet for a particular token.

Token Pool (Token 1)

Assets	Liabilities
Cash: 95 Loan: 10	Deposit: 100

Actions that can be performed against token pools and their balance sheet impact will be discussed in Section 3. Accounting ratios and associated financial risks will be mentioned in Section 4.

Token pool is comprised of *State Token* and *System Balance*, to be discussed in Section 7.

3 Actions

Assuming two tokens exist in the token pools. The exchange rate of Token 1:USD is 100:1 and Token 2:USD is 1:1. Numbers are displayed in tokens' native unit.

3.1 Deposit and Withdraw

Everything starts from deposits. The system collects capital from liquidity providers when they deposit and utilizes them to provide services to casual users. Deposits are akin to lend money to the system.

Definition 3.1.1. (Deposit). When a liquidity provider deposits l_i amount of token i into token pool i.

Example. Assume the liquidity provider holds 10,000 units of token 1 in his wallet and deposit $l_1 = 100$ into the system.

User Wallet (Token 1)

Assets	Liabilities
Cash: 10,000	

Token Pool (Token 1)

Assets	Liabilities

Base State

User Wallet (Token 1)

Assets	Liabilities
Cash: 9,900 Deposit: 100	

Token Pool (Token 1)

Assets	Liabilities
Cash: 100	Deposit: 100

After Deposit

Definition 3.1.2. (Withdraw). When a liquidity provider withdraws l_i amount of token i from token pool i.

Example. Following the setup for the previous deposit case. The liquidity provider subsequently withdraws $l_1 = 100$ from the system.

User Wallet (Token 1)

Assets	Liabilities
Cash: 9,900 Deposit: 100	

Token Pool (Token 1)

Assets	Liabilities
Cash: 100	Deposit: 100

After Deposit

User Wallet (Token 1)

Assets	Liabilities
Cash: 10,000	

Token Pool (Token 1)

Assets	Liabilities

After Withdraw

Withdrawal is subjected to *withdrawal fees*, to be defined in Section 5.2.

3.2 Token Swapping

Token swapping means exchanging one token for another. Upon token swapping, liquidity for one token (the from-token) increases but for another one (the to-token) decreases.

Definition 3.2.1. (External Swap). When l_i amount of token i is injected into the system, in exchange for l_j amount of token j from the system.

Example. Assume $l_1 = 1$ for token 1. Using the given exchange rate, the corresponding value is $l_2 = 100$ for token 2.

Token Pool (Token 1)

Assets	Liabilities
Cash: 100	Deposit: 100

Token Pool (Token 2)

Assets	Liabilities
Cash: 10,000	Deposit: 10,000

Base State

Token Pool (Token 1)

Assets	Liabilities
Cash: 101	Deposit: 100

Token Pool (Token 2)

Assets	Liabilities
Cash: 9,900	Deposit: 10,000

After External Swap

Definition 3.2.2. (Internal Swap). When l_i amount of token i already deposited in the system is used for exchanging l_j amount of token j in the system.

Example. Assume $l_1 = 1$ for token 1 is already deposited in the system. Using the given exchange rate, the corresponding value is $l_2 = 100$ for token 2.

Token Pool (Token 1)

Assets	Liabilities
Cash: 100	Deposit: 100

Token Pool (Token 2)

Assets	Liabilities
Cash: 10,000	Deposit: 10,000

Base State

Token Pool (Token 1)

Assets	Liabilities
Cash: 100	Deposit: 99

Token Pool (Token 2)

Assets	Liabilities
Cash: 10,000	Deposit: 10,100

After Internal Swap

Token swapping is subjected to *price slippage*, to be defined in Section 5.1.

Note: Only external swap will be implemented in the first version.

4 Financial Risk

The fundamental financial risks are *Solvency Risk* and *Liquidity Risk*. By monitoring these two risks and apply appropriate strategies, we can ensure system's financial healthiness.

4.1 Solvency Risk

Insolvency occurs when the system does not have enough assets to cover its liabilities. There are two ways for DeFi protocols to tackle insolvency today:

- Token swappers (e.g.: Uniswap [3], Balancer [16]) require liquidity providers to inject liquidity with at least 2 token pairs, and all exchange rate risks are transferred to liquidity providers; hence, the system itself doesn't take on any solvency risk.
- Money markets (e.g.: Compound [14], Aave [1]) and Stablecoins (e.g.: MakerDAO [2]) rely on over-collateralization and upon token-level insolvency they will sell the collateral at discount via auction or liquidation to unwind the loan position.

These systems have either shifted the exchange rate risk to liquidity providers, or use over-collateralization with auction or liquidation to protect the system. Worst still, many systems cannot tolerate token-level insolvency, which limits the flexibility for the system to allocate capital towards better uses. One of our system's contribution is, by clarifying solvency risk, token-level insolvency can be relieved for better capital efficiency.

Definition 4.1.1. (Token-level insolvency). When the assets A_i of a token i (where $i \in N$) become less than its liabilities L_i.

$$A_i < L_i$$

Definition 4.1.2. (System-level insolvency). When the sum of assets A_i for all token i ($\forall\, i \in N$) become less than the sum of liabilities L_i, measured in a common unit by some exchange rate fx_i [2].

$$\sum_{i=1}^{N} A_i \times fx_i < \sum_{i=1}^{N} L_i \times fx_i$$

[2] The common unit is usually USD.

Temporary insolvency does not imply permanent insolvency. For token-level insolvency, the solvency position can be restored when users perform token swapping from the insolvent token to other tokens. For system-level insolvency, the solvency position can be restored when the future income of the system is sufficient to recover any shortfall existed, or through governance token insurance (Section 6.3).

Definition 4.1.3. (Coverage Ratio). Coverage ratio $r_{c,i}$ is defined by assets A_i divided by liabilities L_i for some token i where $i \in N$. The lower the ratio, the higher the solvency risk of the token.

$$r_{c,i} = \frac{A_i}{L_i}$$

Example. An illustration with the external swapping case.

Token Pool (Token 1)

Assets	Liabilities	Coverage Ratio
Cash: 101	Deposit: 100	$r_{c,1}$: 1.01

Token Pool (Token 2)

Assets	Liabilities	Coverage Ratio
Cash: 9,900	Deposit: 10,000	$r_{c,2}$: 0.99

We will use coverage ratio as an input for the *price slippage curve* in our automated market maker, to be discussed in Section 5.1.

4.2 Exchange Rate Risk

Exchange rate risk means that the system may incur losses due to fluctuations in the exchange rate. It is the main reason for why the system may fall into insolvency.

Lemma 4.2.1. For a zero-surplus system (i.e.: assets equal liabilities), either

$$r_{c,i} = 1 \; \forall i \; \in N, \text{or}$$
$$\exists \, i, j \in N \text{ s.t. } r_{c,i} < 1 \text{ and } r_{c,j} > 1$$

Proof. First, we note that a zero-surplus system means:

$$\sum_{i=1}^{N} (A_i - L_i) \times fx_i = 0$$

Since $fx_i \geq 0 \; \forall \; i \in N$, either:

- $A_i - L_i = 0 \; \forall : i \in N \rightarrow r_{c,i} = 1 \; \forall \; i \in N, \text{or}$
- $\exists \, i, j \in N$ s.t. $r_{c,i} < 1$ and $r_{c,j} > 1$

Proposition 4.2.2. Exchange rate risk may lead to system-level insolvency for a zero-surplus system if $\exists \; i \in N$ s.t. $r_{c,i} \neq 1$

Proof. If $\exists \; i \in N$ s.t. $r_{c,i} \neq 1$ for a zero surplus system, from Lemma 4.2.1. we know that $\exists \, i, j \in N$ s.t. $r_{c,i} < 1$ and $r_{c,j} > 1$.

We only need to name one example to prove the statement is true. Assume $\exists \; k \in i$ s.t. $fx'_k > fx_k$ and $fx' = fx$ otherwise.

$$\begin{aligned}
&\sum_{i=1}^{N} (A_i - L_i) \times fx'_i \\
&= \sum_{i=1}^{N} (A_i - L_i) \times fx_i + (A_k - L_k) \times (fx'_k - fx_k) \\
&= 0 + (A_k - L_k) \times (fx'_k - fx_k)
\end{aligned}$$

By assumption, $r_{c,k} < 1$,hence $A_k - L_k < 0$. Therefore $(A_k - L_k) \times (fx^{'}_k - fx_k) < 0$ and hence the system is insolvent.

Exchange rate risk is unavoidable if we are allowing token-level insolvency. With some coverage ratio smaller than 1 (and others larger than 1), the system is always taking side in a bet.

The system will collect fees from all user activities described in Section 3. These fees will gradually build up the equity (i.e.: surplus) position as capital buffer to defense against insolvency.

5 Fees and Slippages

User actions, including token swapping and borrowing, are subjected to fees and slippage. Fees collected will help system to build up surplus to combat insolvency.

5.1 Swapping Fees

In traditional exchange models (i.e.: order book), liquidity is provided by market makers who place limit orders in the order book. When a large buy order is being executed, the average price will increase as the order needs to sweep through higher part of the order book to source for sufficient liquidity. This phenomenon is called *price slippage*.

In a token swapper model, the order book no longer exists, and the liquidity provider has given up their right to set a price. It becomes the system's sole responsibility to determine a proper algorithm for setting the exchange rate and slippage. As a result, we need to determine an algorithm that has a similar effect as price slippage to penalize large orders. This algorithm is known as an *automated market maker*.

Automated market makers (AMM) have been studied in [11], [12], [17], [18] but they are largely focused on prediction market. Recently, DeFi systems like Uniswap [3] and Balancer [16] have extended the concept of AMM with constant function market maker ([4], [23]) to digital asset exchanges, but they require liquidity providers to inject liquidity in pairs. Curve [9] allows single-sided liquidity provision but is focused only on stablecoins. Moreover, all these systems are transferring exchange rate risk (called impermanent loss [19]) to liquidity providers, which is not ideal as lender normally would not want to expose to exchange rate risk.

Our proposed AMM will be used for digital asset exchange. To obtain the price information, price oracles such as Chainlink [10] will be used and the AMM determines the price slippage. Upon token swapping, liquidity for one token is improved while it is reduced for the other token. This increases the solvency risk for the system since the under-covered position is exposed to exchange rate risk. From this perspective, it is reasonable to establish price slippage as a function of coverage ratio.

Definition 5.1.1. (Swapping Exchange Rate). The exchange rate used in token swapping is defined as:

$$fx_{i\to j} = \frac{fx_i}{fx_j} \times (1 - S_{i\to j} - h)$$

where fx_k is the token k:USD ($k \in \{i, j\}$) exchange rate obtained from price oracle and $S_{i\to j}$ is the swapping slippage to be defined in Definition 5.1.3 and h is haircut to be defined in Definition 5.1.7 respectively.

Definition 5.1.2. (Price Slippage Curve). Price slippage curve f maps coverage ratio r_c to $f(r_c)$ such that its slope will become the price slippage. The price slippage curve should satisfy the following properties:

1. f is continuous
2. f is decreasing
3. f is convex
4. $f'(r_c) < \epsilon$ for $r_c > \theta$

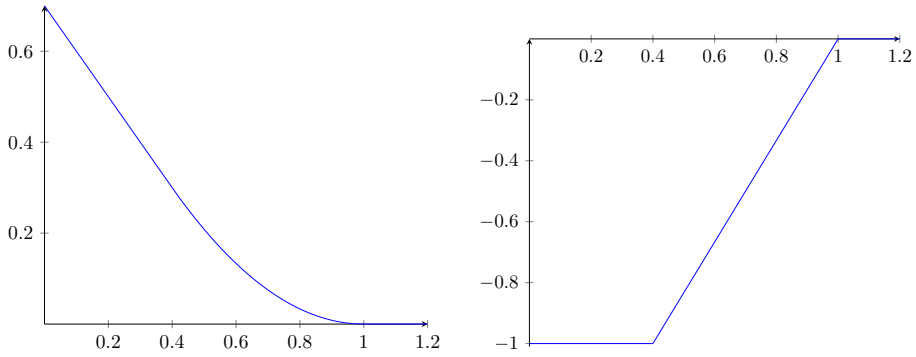
The continuous requirement ensures that there are no sudden jumps in the slippage. The decreasing requirement ensures that the slope is always negative as slippage is a fee. The convexity requirement is an important requirement for the slippage curve as we will use this property later. Condition 4 is more art than science, as slippage should not be so expensive such that the token swapper becomes unattractive even when the system's solvency risk is low. For example, if $\theta = 1$ we may only want to charge slippage no larger than $\epsilon = 0.1\%$.

There are many choices for f. The one that our system will use:

$$f(x) = \begin{cases} -x + \dfrac{a+b}{2} & \text{if } 0 \leq x < a \\ \dfrac{(x-b)^2}{2(b-a)} & \text{if } a \leq x < b \\ 0 & \text{if } x \geq b \end{cases}$$

where a and b are adjustable parameters.

Figure 2: Plot of price slippage curve f and f' ($a = 0.4$ and $b = 1$)



The price slippage curve in 5.1.2 takes only one input (the coverage ratio) from a token. But there are two tokens involved in token swapping, where one token is expected to have an improving coverage ratio while the other worsening. To determine the final slippage that is chargeable, we will simply add up the slippage for both tokens.

Definition 5.1.3. (Swapping Slippage). Assume token swapping of l_i token i to l_j token j. The corresponding coverage ratios before and after swapping are therefore $r_{c,i}$, $r_{c,j}$ and $r'_{c,i}$, $r'_{c,j}$. The swapping slippage $S_{i\to j}$ is therefore:

$$S_i = \frac{f(r'_{c,i}) - f(r_{c,i})}{r'_{c,i} - r_{c,i}}$$
$$S_j = \frac{f(r'_{c,j}) - f(r_{c,j})}{r'_{c,j} - r_{c,j}}$$
$$S_{i\to j} = S_i - S_j$$

Proposition 5.1.4. Swapping slippage penalizes actions that moves $r_{c,i}$ and $r_{c,j}$ apart, but rewards actions that bring them closer.

Proof. Assume $r_{c,i} > r_{c,j}$. After swapping, the coverage ratio is moved such that $r'_{c,i} > r_{c,i}$ and $r'_{c,j} < r_{c,j}$. Hence, the swapping action is moving $r_{c,i}$ and $r_{c,j}$ apart.

By slope inequality [8]:

$$\frac{f(r_{c,i}) - f(r_{c,j})}{r_{c,i} - r_{c,j}} \leq \frac{f(r'_{c,i}) - f(r_{c,j})}{r'_{c,i} - r_{c,j}} \leq \frac{f(r'_{c,i}) - f(r_{c,i})}{r'_{c,i} - r_{c,i}}$$

$$\frac{f(r_{c,j}) - f(r'_{c,j})}{r_{c,j} - r'_{c,j}} \leq \frac{f(r_{c,i}) - f(r'_{c,j})}{r_{c,i} - r'_{c,j}} \leq \frac{f(r_{c,i}) - f(r_{c,j})}{r_{c,i} - r_{c,j}}$$

This immediately follows that:

$$\frac{f(r'_{c,j}) - f(r_{c,j})}{r'_{c,j} - r_{c,j}} \leq \frac{f(r'_{c,i}) - f(r_{c,i})}{r'_{c,i} - r_{c,i}}$$

$$S_i - S_j \geq 0$$

i.e.: The system is penalizing actions that moves $r_{c,i}$ and $r_{c,j}$ apart.

Proof. Assume $r_{c,i} < r_{c,j}$. After swapping, the coverage ratio is moved such that $r_{c,i} < r'_{c,i} < r_{c,j}$ and $r_{c,i} < r'_{c,j} < r_{c,j}$. Hence, the swapping action is moving $r_{c,i}$ and $r_{c,j}$ closer.

By slope inequality [8]:

$$\frac{f(r_{c,i}) - f(r'_{c,i})}{r_{c,i} - r'_{c,i}} \leq \frac{f(r_{c,i}) - f(r_{c,j})}{r_{c,i} - r_{c,j}} \leq \frac{f(r'_{c,i}) - f(r_{c,j})}{r'_{c,i} - r_{c,j}}$$

$$\frac{f(r_{c,i}) - f(r'_{c,j})}{r_{c,i} - r'_{c,j}} \leq \frac{f(r_{c,i}) - f(r_{c,j})}{r_{c,i} - r_{c,j}} \leq \frac{f(r'_{c,j}) - f(r_{c,j})}{r'_{c,j} - r_{c,j}}$$

This immediately follows that:

$$\frac{f(r'_{c,i}) - f(r_{c,i})}{r'_{c,i} - r_{c,i}} \leq \frac{f(r'_{c,j}) - f(r_{c,j})}{r'_{c,j} - r_{c,j}}$$

$$S_i - S_j \leq 0$$

i.e.: The system is rewarding actions that moves $r_{c,i}$ and $r_{c,j}$ closer.

This is an ideal property for swapping slippage. We would like to make the token-level coverage ratio less extreme to reduce exchange rate risk (thereby solvency risk) of the system.

Definition 5.1.5. (Swapping Fees). Swapping fees will be charged based on swapping slippage on the to-token j.

$$Sf_{i\to j} = \Delta y \times S_{i\to j}$$

Othman et.al. [17] showed that no market maker can satisfy three desirable properties: path independence, no-arbitrage and liquidity sensitivity.

Proposition 5.1.6. Swapping fees is path independent and liquidity sensitive for given exchange rate.

Proof. Swapping fees is path independent for given exchange rate.

$$\begin{aligned}
Sf_{i\to j} &= \Delta y \times S_{i\to j} \\
&= \Delta y \times (S_i - S_j) \\
&= \Delta y \times \left(\frac{L_i \times \big(f(r'_{c,i}) - f(r_{c,i})\big)}{\Delta x} - \frac{L_j \times \big(f(r'_{c,j}) - f(r_{c,j})\big)}{\Delta y} \right) \\
&= \frac{\Delta y}{\Delta x} \times L_i \times \big(f(r'_{c,i}) - f(r_{c,i})\big) - L_j \times \big(f(r'_{c,j}) - f(r_{c,j})\big) \\
&= fx \times L_i \times \big(f(r'_{c,i}) - f(r_{c,i})\big) - L_j \times \big(f(r'_{c,j}) - f(r_{c,j})\big) \\
&= \big(fx \times L_i \times f(r'_{c,i}) - L_j \times f(r'_{c,j})\big) - (fx \times L_i \times f(r_{c,i}) - L_j \times f(r_{c,j}))
\end{aligned}$$

Setting $\mathbf{F}(r_{c,i}, r_{c,j}) = (fx \times L_i \times f(r_{c,i}) - L_j \times f(r_{c,j}))$:
$Sf_{i\to j} = \mathbf{F}(r'_{c,i}, r'_{c,j}) - \mathbf{F}(r_{c,i}, r_{c,j})$

Hence $Sf_{i\to j}$ is path independent.

Proof. Swapping fees is liquidity sensitive.

Compare $Sf_{i\to j}$ against $Sf^*_{i\to j}$ where $r_{c,j} = r^*_{c,j}$ but $L_j < L^*_j$.

For the same swapping amount Δy, this implies $r'_{c,j} < r'^*_{c,j}$.

Since f is decreasing and convex, $S_{i\to j} \leq S^*_{i\to j}$, this follows that slippage fees $Sf_{i\to j} \leq Sf^*_{i\to j}$ and therefore is liquidity sensitive.

This described a desirable system property: The higher the liquidity, the lower the slippage.

Since the swapping fee is path independent, fees collected from actions that brings coverage ratios apart will be offset by the reverse of those actions. This makes the system profit neutral, which is undesirable because the system needs to generate profit to pay dividends to liquidity providers, and also to build up a capital buffer to defend against insolvency.

Definition 5.1.7. (Haircut). A haircut h will be charged upon swapping on the to-token j.

$$hf_j = \Delta y \times h$$

The haircut will make both side of the trade subject to a small fee (e.g.: 0.1%), making the trades asymmetric such that the system will have positive expected fees being collected.

One more point to note is that the swapping fees mentioned in Definition 5.1.5. will only be allocated into the system surplus but not the as dividend (hence not obliged to the Profit Sharing rule). The reason is that the distortion of the coverage ratios is assumed temporary and the fees collected will later be transferred to arbitrageurs. Hence they should not become dividends to liquidity providers.

Instead, the fees collected from haircut in Definition 5.1.7. are permanent and should follow the Profit Sharing rule to be described in Section 6.1.

5.2 Withdrawal Fees

Withdraws are subject to fees to defend against *Withdraw Arbitrage*, to be discussed in Section 8.5.

Definition 5.2.1. (Withdrawal Fees). Withdrawal fees w is defined as:

$$w = \frac{(1 - r_c') \times f(r_c) - (1 - r_c) \times f(r_c')}{r_c' - r_c}$$

Proof. First, we consider the swapping gains from a Δx order:

$$-\Delta x \times \frac{f(r_c^*) - f(r_c)}{r_c^* - r_c}$$

Note that $r_c^* = \frac{A+\Delta x}{L}$ and $r_c = \frac{A}{L}$. Therefore:

$$-\Delta x \times \frac{f(r_c^*) - f(r_c)}{\frac{A+\Delta x}{L} - \frac{A}{L}} = L \times (f(r_c) - f(r_c^*))$$

The maximum swapping gain occurs when $\Delta x \to \infty$, or $f(r_c^*) \to 0$:

$$L \times (f(r_c) - f(r_c^*)) \to L \times f(r_c)$$

Now assume someone withdraws y amount of the token, such that the coverage ratio changes from $r_c = \frac{A}{L}$ to $r_c' = \frac{A-y}{L-y}$. Following the same argument, the maximum dollar gain from token swapping after withdrawal is $(L - y) \times f(r_c')$. The change in maximum dollar gain from token swapping is therefore:

$$(L - y) \times f(r_c') - L \times f(r_c)$$

Hence the fees per withdraw y needed to eliminate the gain above:

$$\frac{(L - y) \times f(r_c') - L \times f(r_c)}{y}$$

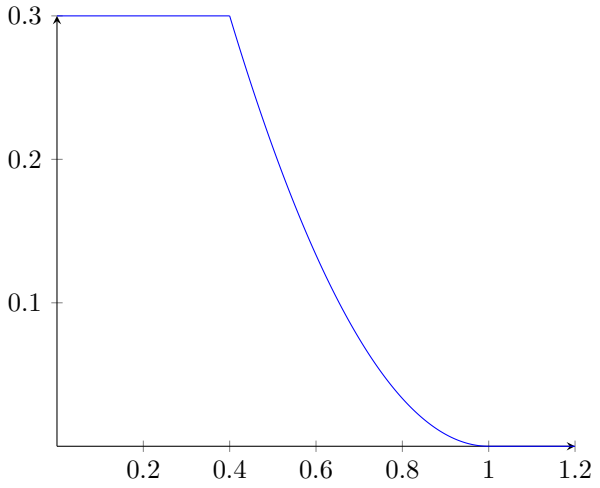
After some algebra we will have the result w specified in Definition 5.2.1

We can derive the derivatives for w by taking limit on $\Delta r \to 0$.

By setting $r_c' = r_c + \Delta r$:

$$\begin{aligned} w'(r_c) &= \lim_{\Delta r \to 0} \frac{(1 - r_c - \Delta r) \times f(r_c) - (1 - r_c) \times f(r_c + \Delta r)}{r_c + \Delta r - r_c} \\ &= \lim_{\Delta r \to 0} \frac{(1 - r_c) \times (f(r_c) - f(r_c + \Delta r))}{\Delta r} - f(r_c) \\ &= -(1 - r_c) \times \lim_{\Delta r \to 0} \frac{f(r_c + \Delta r) - f(r_c)}{\Delta r} - f(r_c) \\ &= -\left((1 - r_c) \times f'(r_c) + f(r_c)\right) \end{aligned}$$

Figure 3: Plot of withdrawal fees w ($a = 0.4$ and $b = 1$)



6 Dividend

We have mentioned how fees will be collected in the previous section. A natural subsequent question is: how do we allocate and distribute the collected fees between liquidity providers and the system?

6.1 Profit Sharing

Profit sharing is controlled by a parameter: the retention ratio.

Definition 6.1.1. (Retention Ratio). The retention ratio r determines the portion of fees that will be remained to the system. Hence, the remaining portion $1 - r$ will be paid to liquidity providers. Practically, this can be done by debiting the fees to cash in full and crediting liabilities with $1 - r$ portion.

Example. An illustration below where fees is collected after borrowing. Assume the user borrowed 1 unit of token 1. The system is charging 0.1 unit of token 1 as interest. We set $r = 0.2$.

Token Pool (Token 1)

Assets	Liabilities
Cash : 99 Loan: 1.1	Deposit: 100

After Borrow

Token Pool (Token 1)

Assets	Liabilities
Cash: 100.1	Deposit: 100.08

After Repay

While the ratio may seem low, note that liquidity providers are only eligible for the fees collected for the same token they have provided liquidity for, while governance token holders may eventually redeem surplus from any tokens in token pools.

6.2 System Surplus

As fees are gradually collected in the token pools after profit sharing (the retained portion), the system's equity position will build up and become token surplus and system surplus.

Definition 6.2.1. (Token Surplus). Token surplus E_i (i.e.: token equity) for token i is defined by its asset A_i minus its liabilities L_i.

$$E_i = A_i - L_i$$

Definition 6.2.2. (System Surplus). System surplus E (i.e.: system equity) is defined by sum of token equities for all tokens $i \in N$, measured in a common unit by some exchange rate fx_i [3].

$$E = \sum_{i=1}^{N} E_i \times fx_i$$

When E becomes negative, it is called *System Deficits*. Negative E is also equivalent to system-level insolvency as described in Definition 4.1.2.

Token surplus and system surplus are important because of the following:

- *Insurance Fund*. As token surplus represents excess equity in token pools, they naturally become an insurance fund to protect against insolvency under adverse scenarios. The beauty of the system is that system surplus becomes an integrated part of the system and requires no further engineering effort to implement.
- *Lower Slippage*. The slippage curves become insensitive when the coverage ratio / quick ratio becomes large as proven in Proposition 5.1.6. Hence, maintaining a surplus may reduce slippage and enhance the system's overall competitiveness. Another way to interpret is that, since the system has lower insolvency risk, the costs charged should also be lower.

[3] The common unit is usually USD.

7 State Tokens

Ethereum is a transaction-based *state machine*. The following states need to be kept for respective system actors:

- *Liquidity Providers*. Liabilities owe to each liquidity providers;
- *Borrowers*. Loans (plus interest) the borrowers owe to the system;
- *Minters*. Stablecoins that minters have minted in the system;
- *Stakers*. Governance tokens that the system owe to each stakers.

Many actions may affect the aforementioned states. For example, upon swapping the haircuts collected will increase the system liabilities as dividend (and proportionally shared among liquidity providers). For gas optimization, we prefer not to update each individual liquidity providers' balance, since the gas cost will become proportional to the total number of liquidity providers $O(N)$. Instead, we can use a trick called *State Token* to reduce the gas cost to $O(1)$.

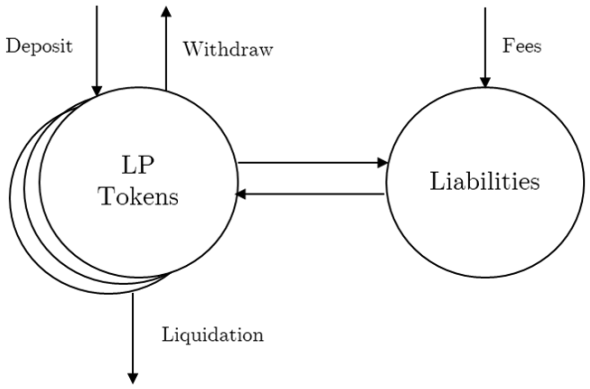


Figure 4: LP Token.

The rough idea is to separated the individual balance and system balance. If the action is triggered by individual actions (e.g. deposit / withdraw), the individual balance will be updated via state token. If the action is triggered by system changes (e.g. fees), only the system balance will be affected. Since the system balance is a number to be recorded on the blockchain, this prevents the need of updating individual balance for each actor.

The states required to be tracked are:

Actors	Individual	System
Liquidity Providers	LP Tokens	Liabilities
Borrowers	Debt Tokens	Debts
Minters	Mint Tokens	Mints
Stakers	Stake Tokens	Stakes

7.1 LP Token

LP Token represents the proportional share that a liquidity providers possess against total *Liabilities*. It can be summarized as below:

Event	LP Tokens	Liabilities
Fees		+
Deposit	+	+
Withdraw	-	-
Liquidation	-	-

Besides liabilities, the system may also record the initial deposit amount (per liquidity provider) in order to calculate each LP's rate of return.

Definition 7.1.1. (Fees). Fees f affects only the *Liabilities* L. Formula defined by $L' = L + f$.

Example. An illustration below for adding 2 unit of fees into Liabilities.

Token Pool (Token 1)

LP Tokens	Liabilities
User 1: 50 User 2: 50	Balance: 100

Base State

Token Pool (Token 1)

LP Tokens	Liabilities
User 1: 50 User 2: 50	Balance: 102

After Fees

It can be seen that now the system owes each user 51 units of token, even the LP Token owned by each of them is only 50 units.

Definition 7.1.2. (Deposit). Deposit d affects both *LP Tokens* l and *Liabilities* L. Formula defined by: $L' = L + d$ and $l' = l + d \times \frac{\Sigma l}{L}$.

Example. Following the setup for the previous case. Assume User 1 further deposits 50 unit of token to the system.

Token Pool (Token 1)

LP Tokens	Liabilities
User 1: 50 User 2: 50	Balance: 102

After Fees

Token Pool (Token 1)

LP Tokens	Liabilities
User 1: 99.02 User 2: 50	Balance: 152

After Deposit

It can be seen that now the system owes User 1 101 units of token and User 2 51 units of token. The new LP tokens didn't dilute User 2's share.

Definition 7.1.3. (Withdraw). Withdraw w affects both *LP Tokens* l and *Liabilities* L. Formula defined by: $L' = L - w$ and $l' = l - w \times \frac{\Sigma l}{L}$.

Example. Following the setup for the previous case. Assume User 2 withdraw half of his holdings from the system.

Token Pool (Token 1)

LP Tokens	Liabilities
User 1: 99.02 User 2: 50	Balance: 152

After Deposit

Token Pool (Token 1)

LP Tokens	Liabilities
User 1: 99.02 User 2: 25	Balance: 126.5

After Withdraw

It can be seen that now the system owes User 1 101 units of token and User 2 25.5 units of token. User 2 withdraws 25.5 units of token.

8 Possible Attacks

We will analyze some possible attacks for the system, and discuss methods to mitigate them. Summary below:

Risk	Cause	Solution
Reentrancy	Implementation flaw	Security Audit
Front-Running	Inherited from design	Max Threshold
Oracle Attack	Oracle Price can be manipulated (e.g.: flash loan)	Whitelist tokens + Multiple oracles
Split Orders	Inherited from design	Path Independence
Withdrawal Arbitrage	Inherited from design	Withdrawal fees

8.1 Reentrancy

Reentrancy is a well-known attack as seen in the DAO incident [6]. In general, this attack can be prevented by finishing all internal work (i.e.: state changes) first, and only then calling external functions [13].

8.2 Front-Running

Since all transactions are visible in the mempool for a short while before being executed, observers of the network can see and react to an action before it is included in a block. An example of how this can be exploited is with a decentralized exchange where a buy order transaction can be seen, and second order can be broadcast and executed before the first transaction is included.

Our solution is similar to the one Uniswap [3] adopted, where each user needs to specify a "maximum slippage". If the final slippage is larger than this threshold, then the user action will be canceled.

8.3 Oracle Attack

Oracle attacks may happen if the external market that the oracle is relying on is manipuatable [20]. This may happen if the liquidity or transaction volume in the external market is too small. To address the possibility of an oracle attack, the system should only list reputable, liquid tokens in the beginning. Multiple oracles should be sourced (e.g.: Uniswap [3], Chainlink [10]) and time-weighted average price [3] can be used to protect against flash-loan [22].

8.4 Split Orders

Split orders mean that an order is split into multiple orders with same total amount to achieve a different aggregate result from the single, larger order. For example, a split order attack is possible if the slippage from swapping 1 token is different from swapping 0.5 tokens twice. This attack can be mitigated by ensuring that the slippage curve is path independent:

Proof. Assuming the swap brings $r_{c,i}$ to $r'_{c,i}$ and $r_{c,j}$ to $r'_{c,j}$. By the result in Proposition 5.1.6:

$$Sf_{i,j} = \mathbf{F}(r'_{c,i}, r'_{c,j}) - \mathbf{F}(r_{c,i}, r_{c,j})$$

Now assume the swap is breakdown into 2 transactions, such that the swap brings $r_{c,i}$ first to $r^*_{c,i}$ then $r'_{c,i}$ and $r_{c,j}$ first to $r^*_{c,j}$ then $r'_{c,j}$:

$$Sf^*_{i,j} = \mathbf{F}(r^*_{c,i}, r^*_{c,j}) - \mathbf{F}(r_{c,i}, r_{c,j})$$
$$Sf'_{i,j} = \mathbf{F}(r'_{c,i}, r'_{c,j}) - \mathbf{F}(r^*_{c,i}, r^*_{c,j})$$

Hence we have $Sf_{i,j} = Sf^*_{i,j} + Sf'_{i,j}$ and completes the proof.

8.5 Withdrawal Arbitrage

Withdrawal arbitrage is a possible attack in the absence of withdrawal fees. Assume $r \leq 1$, a user can first swap to remove Δx amount of the token to push its coverage ratio from r to r^*, withdraw y liquidity to further push down the coverage ratio to r^{**}, then swap the same amount Δx back (coverage ratio to r') to earn the difference in slippage fees.

In other words:

$$\frac{f(r) - f(r^*)}{r - r^*} > \frac{f(r^{**}) - f(r')}{r^{**} - r'}$$

Proof. Note that if $r_{c,i} \leq 1$, we have $r > r^* > r'$ and $r > r^{**} > r'$:

By slope inequality [8]:

$$\frac{f(r^*) - f(r')}{r^* - r'} < \frac{f(r) - f(r')}{r - r'} < \frac{f(r) - f(r^*)}{r - r^*}$$

$$\frac{f(r^{**}) - f(r')}{r^{**} - r'} < \frac{f(r) - f(r')}{r - r'} < \frac{f(r) - f(r^{**})}{r - r^{**}}$$

This immediately follows that:

$$\frac{f(r) - f(r^*)}{r - r^*} > \frac{f(r^{**}) - f(r')}{r^{**} - r'}$$

Withdrawal arbitrage can be prevented by introducing withdrawal fees specified in Section 5.2.

9 Economics Analysis

We will analyze the economics aspects of the protocol in this section.

9.1 Value at Risk

The system risk can be modeled exactly by the Markovwitz model [15].

Definition 9.1.1. The Value at Risk (VaR) of the system is at quantile q is:

$$\text{VaR}(q) = \boldsymbol{E'\mu} + \sqrt{\boldsymbol{E'\Sigma E}} \times \Phi^{-1}(q)$$

Proof. Definition 6.2.2. has defined the system surplus (deficit). Notice that it is also the system's portfolio that is exposed to exchange rate risk.

$$E = \sum_{i=1}^{m} E_i \times fx_i$$

By concerning about the return r of individual fx_i between some time interval, we can rewrite the above equation:

$$P = \sum_{i=1}^{m} E_i \times r_i$$

where $r_i \sim N(\mu_i, \sigma_i^2)$ and $Corr(r_i, r_j) = \rho_{ij}$.

Hence, the mean and variance of the portfolio is given by:

$$\text{E}(P) = \boldsymbol{E'\mu}$$

$$\text{Var}(P) = \boldsymbol{E'\Sigma E}$$

where $\boldsymbol{\Sigma}$ is the variance-covariance matrix.

Hence, the result in definition 9.1.1. follows.

9.2 Arbitrageur's Profit

We will analyze arbitrageur's optimal strategy and the associated profit in this section.

Proposition 9.2.1. Arbitrageur's optimal strategy is to restore all coverage ratios' r_i to the same value r^*, where:

$$r^* = \frac{\sum_{i=1}^{n} L_i \times fx_i \times r_i}{\sum_{i=1}^{n} L_i \times fx_i}$$

and the associated profit is:

$$\boldsymbol{F^*} = \sum_{i=1}^{N} L_i \times fx_i \times (f(r^*) - f(r_i))$$

Proof. The fees (or rewards) for the bringing the coverage ratios from r to r' are:

$$\Delta x \times \frac{f(r') - f(r)}{r' - r} = L \times (f(r') - f(r))$$

Assuming there are n tokens, and token follows exchange rate fx_i. The profit or loss after swapping is:

$$\sum_{i=1}^{N} L_i \times fx_i \times (f(r_i') - f(r_i))$$

subjected to:

$$\sum_{i=1}^{n} fx_i \Delta x_i = 0$$

The constraint equation can be rewritten as:

$$\sum_{i=1}^{n} L_i \times fx_i \times (r_i' - r_i) = 0$$

The maximum profit obtainable from the system is equivalent to the following optimization problem:

$$\begin{aligned} \min_{\vec{r'}} \quad & \sum_{i=1}^{N} L_i \times fx_i \times (f(r_i') - f(r_i)) \\ \text{s.t.} \quad & \sum_{i=1}^{N} L_i \times fx_i \times (r_i' - r_i) = 0 \end{aligned}$$

Denote $\boldsymbol{F}$ to be the optimization equation and $\boldsymbol{G}$ to be the constraint equation. The Lagrangian function is:

$$\mathcal{L} = \boldsymbol{F} - \lambda \boldsymbol{G}$$

Differentiate the above equation with respect to r_i and λ gives us the system of equations:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial r_i'} = L_i \times fx_i \times (f'(r_i') - \lambda) = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = \displaystyle\sum_{i=1}^{N} L_i \times fx_i \times (r_i' - r_i) = 0 \end{cases}$$

The first line implies that the extrema is reached if:

$$f'(r_1') = f'(r_2') = ... = f'(r_n') = \lambda \tag{1}$$

Next, we want to show that $\boldsymbol{F}$ follows the convexity of f.

$$\begin{aligned} \boldsymbol{F}(\alpha \vec{R'} + (1-\alpha)\vec{R^*}) &= \sum_{i=1}^{n} L_i \times fx_i \times (f(\alpha r_i' + (1-\alpha) r_i^*) - f(r_i)) \\ &\leq \sum_{i=1}^{n} L_i \times fx_i \times (\alpha f(r_i') + (1-\alpha) f(r_i^*) - f(r_i)) \\ &= \alpha \sum_{i=1}^{n} L_i \times fx_i \times f(r_i') + (1-\alpha) \sum_{i=1}^{n} L_i \times fx_i \times f(r_i^*) \\ &= \alpha \boldsymbol{F}(R') + (1-\alpha)\boldsymbol{F}(R^*) \end{aligned}$$

Hence $\boldsymbol{F}$ is convex.

Now, obviously $r_1' = r_2' = ... = r_n' = r^*$ satisfies equation (1). Substituting this into $\boldsymbol{G}$ gives us:

$$r^* = \frac{\sum_{i=1}^{n} L_i \times fx_i \times r_i}{\sum_{i=1}^{n} L_i \times fx_i}$$

Hence the maximum profit that an arbitrageur can gain at any snapshot is:

$$\boldsymbol{F^*} = \sum_{i=1}^{N} L_i \times fx_i \times (f(r^*) - f(r_i))$$

Hence, arbitrageurs are incentivized to help to restore the coverage ratios whenever $\boldsymbol{F^*}$ is larger than the costs.

Now let's try to analyze the optimal strategy (and profit) for arbitragers when the oracle price is different from the real price.

Proposition 9.2.2. If the "actual" exchange rate fx^* is different the oracle exchange rate fx used in the system, arbitrageur's optimal strategy is to optimize the following equation:

$$\begin{aligned} \min_{\vec{r'}} \quad & \sum_{i=1}^{N} L_i \times fx_i^* \times ((f(r_i') - f(r_i)) - (r_i - r_i')) \\ \text{s.t.} \quad & \sum_{i=1}^{N} L_i \times fx_i \times (r_i' - r_i) = 0 \end{aligned}$$

Denote $\boldsymbol{F}$ to be the optimization equation and $\boldsymbol{G}$ to be the constraint equation. The Lagrangian function is:

$$\mathcal{L} = \boldsymbol{F} - \lambda \boldsymbol{G}$$

Differentiation the above equation:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial r_i'} = L_i \times fx_i^* \times (f'(r_i') + 1) - L_i \times fx_i \times \lambda = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = \displaystyle\sum_{i=1}^{N} L_i \times fx_i \times (r_i' - r_i) = 0 \end{cases}$$

This is equivalent to the system of equations:

$$\begin{cases} f'(r_i') = \dfrac{fx_i}{fx_i^*} \times \lambda - 1 \\ \displaystyle\sum_{i=1}^{N} L_i \times fx_i \times r_i' = \sum_{i=1}^{N} L_i \times fx_i \times r_i \end{cases}$$

The equation can be solved by complex optimization [5] or other numerical techniques.

The optimization result for proposition 9.2.2. indicates that the optimal strategy for the arbitrageur, unlike that of proposition 9.2.1, may not bring all coverage ratios equal. This in turn, may increase the exchange risk of the system. Hence, quoting a correct exchange rate from the oracle is paramount to reduce system risk.

References

[1] Aave protocol whitepaper v1.0. 2020.

[2] The maker protocol: Makerdao's multi-collateral dai (mcd) system. 2020.

[3] H. Adams, N. Zinsmeister, and D. Robinson. Uniswap v2 core. 2020.

[4] G. Angeris and T. Chitra. Improved price oracles: Constant function market makers. *arXiv preprint arXiv:2003.10001*, 2020.

[5] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] V. Buterin. Critical update re: Dao vulnerability, 2016.

[7] Y. Chen and C. Bellavitis. Blockchain disruption and decentralized finance: The rise of decentralized business models. *Journal of Business Venturing Insights*, 13:e00151, 2020.

[8] K.-S. Chou. Lecture notes in mathematical analysis: Convex functions, 2018.

[9] M. Egorov. Stableswap - efficient mechanism for stablecoin liquidity. 2019.

[10] S. Ellis, A. Juels, and S. Nazarov. Chainlink: A decentralized oracle network. 2017.

[11] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.

[12] R. Hanson. Logarithmic markets coring rules for modular combinatorial information aggregation. *The Journal of Prediction Markets*, 1(1):3–15, 2007.

[13] B. Jiang, Y. Liu, and W. Chan. Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 259–269. IEEE, 2018.

[14] R. Leshner and G. Hayes. Compound: The money market protocol. 2019.

[15] H. Markowitz. Portfolio selection*. *The Journal of Finance*, 7(1):77–91, 1952.

[16] F. Martinelli and N. Mushegian. A non-custodial portfolio manager, liquidity provider, and price sensor. 2019.

[17] A. Othman, D. M. Pennock, D. M. Reeves, and T. Sandholm. A practical liquidity-sensitive automated market maker. *ACM Transactions on Economics and Computation (TEAC)*, 1(3):1–25, 2013.

[18] A. Othman and T. Sandholm. Liquidity-sensitive automated market makers via homogeneous risk measures. In *International Workshop on Internet and Network Economics*, pages 314–325. Springer, 2011.

[19] Pintail. Understanding uniswap returns, 2019.

[20] K. Qin, L. Zhou, B. Livshits, and A. Gervais. Attacking the defi ecosystem with flash loans for fun and profit. *arXiv preprint arXiv:2003.03810*, 2020.

[21] F. Schär. Decentralized finance: On blockchain-and smart contract-based financial markets. *Available at SSRN 3571335*, 2020.

[22] D. Wang, S. Wu, Z. Lin, L. Wu, X. Yuan, Y. Zhou, H. Wang, and K. Ren. Towards understanding flash loan and its applications in defi ecosystem. *arXiv preprint arXiv:2010.12252*, 2020.

[23] Y. Wang. Automated market makers for decentralized finance (defi). *arXiv preprint arXiv:2009.01676*, 2020.