Rai: A Low Volatility, Trust Minimized Collateral for the DeFi Ecosystem

Stefan C. Ionescu, Ameen Soleimani

May 2020

Abstract. We present a governance minimized, decentralized protocol that automatically reacts to market forces in order to modify the target value of its native collateralized asset. The protocol allows anyone to leverage their crypto assets and issue a “reflex index” which is a dampened version of its underlying collateral. We outline how indexes can be useful as universal, low volatility collateral which can protect its holders, as well as other decentralized finance protocols, from sudden market shifts. We present our plans to help other teams launch their own synthetics by leveraging our infrastructure. Finally, we offer alternatives to current oracle and governance structures that are often found in many DeFi protocols.

Contents

Introduction

Money is one of the most powerful coordination mechanisms humanity leverages in order to thrive. The privilege of managing the money supply has historically been kept in the hands of sovereign leadership and the financial elite while being imposed upon an unwitting general public. Where Bitcoin has demonstrated the potential for a grassroots protest to manifest a store-of-value commodity asset, Ethereum gives us a platform to build asset-backed synthetic instruments that can be protected from volatility and used as collateral, or pegged to a reference price and used as a medium-of-exchange for daily transactions, all enforced by the same principles of decentralized consensus.

Permissionless access to Bitcoin for storing wealth and properly decentralized synthetic instruments on Ethereum will lay the foundation for the upcoming financial revolution, providing those at the fringes of the modern financial system the means to coordinate around building the new one.

In this paper, we introduce a framework for building reflex indexes, a new asset type which will help other synthetics flourish and will establish a key building block for the entire decentralized finance industry.

Overview of Reflex Indexes

A reflex index's purpose is not to maintain a specific peg, but to dampen the volatility of its collateral. Indexes allow anyone to gain exposure to the cryptocurrency market without the same scale of risk as holding actual crypto assets. We believe RAI, our first reflex index, will have immediate utility for other teams issuing synthetics on Ethereum (e.g MakerDAO's Multi-Collateral DAI [1], UMA [2], Synthetix [3]) because it gives their systems a lower exposure to volatile assets such as ETH and offers users more time to exit their positions in case of a significant market shift.

In order to understand reflex indexes, we can compare the behaviour of their redemption price to that of a stablecoin's price.

The redemption price is the value of one debt unit (or coin) in the system. It is meant to be used only as an internal accounting tool and it is different from the market price (the value that the market is trading the coin at). In the case of fiat-backed stablecoins such as USDC, the system operators declare that anyone can redeem one coin for one US dollar and thus the redemption price for these coins is always one. There are also cases of crypto-backed stablecoins such as MakerDAO's Multi Collateral DAI (MCD) where the system targets a fixed peg of one US dollar and thus the redemption price is also fixed at one.

In most cases, there will be a difference between the market price of a stablecoin and its redemption price. These scenarios create arbitrage opportunities where traders will create more coins if the market price is higher than redemption and they will redeem their stablecoins for collateral (e.g US dollars in the case of USDC) in case the market price is lower than the redemption price.

Reflex indexes are similar to stablecoins because they also have a redemption price that the system targets. The main difference in their case is that their redemption will not remain fixed, but is designed to change while being influenced by market forces. In Section 4 we explain how an index's redemption price floats and creates new arbitrage opportunities for its users.

Design Philosophy and Go-to-market Strategy

Our design philosophy is to prioritize security, stability and speed of delivery.

Multi-Collateral DAI was the natural place to start iterating on RAI's design. The system has been heavily audited and formally verified, it has minimal external dependencies and it gathered an active community of experts. To minimize development and communications effort, we want to make only the simplest changes to the original MCD codebase in order to achieve our implementation.

Our most important modifications include the addition of an autonomous rate setter, an Oracle Network Medianizer which is integrated with many independent price feeds and a governance minimization layer meant to isolate the system as much as possible from human intervention.

The very first version of the protocol (Stage 1) will only include the rate setter and other minor improvements in the core architecture. Once we prove that the setter works as expected, we can more safely add the oracle medianizer (Stage 2) and the governance minimization layer (Stage 3).

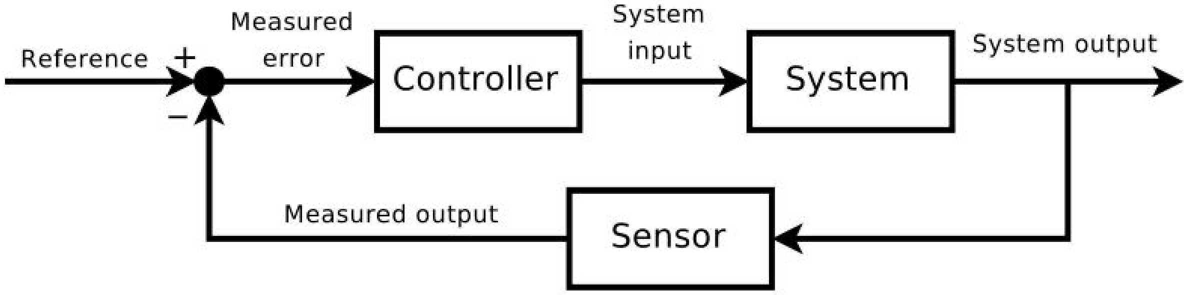
Monetary Policy Mechanisms

Introduction to Control Theory

One common control system that most people are familiar with is the shower. When someone starts a shower, they have a desired water temperature in mind which, in control theory, is called the *reference set point*. The person, acting as the *controller*, continuously measures the water flow temperature (which is called the system *output*) and modifies the speed at which they turn the shower's knob based on the *deviation* (or *error*) between the desired and the current temperature. The speed at which the knob is turned is called the system *input.* The objective is to turn the knob fast enough as to reach the reference set point quickly, but not so fast that the temperature *overshoots*. If there are system *shocks* where the water flow temperature suddenly changes, the person should be able to maintain the current temperature by knowing how fast to turn the knob in response to the disturbance.

The scientific discipline of maintaining stability in dynamic systems is called control theory and it has found broad application in cruise control for cars, flight navigation, chemical reactors, robotic arms, and industrial processes of all kinds. The Bitcoin difficulty adjustment algorithm which maintains the ten minute average block time, despite a variable hashrate, is an example of a mission critical control system.

In most modern control systems an *algorithmic controller* is typically embedded in the process and it is given control over a system input (e.g. a car's gas pedal) in order to automatically update it based on deviations between the system output (e.g. a car's speed) and the setpoint (e.g. the cruise control speed).



The most common type of algorithmic controller is the *PID controller*. Over 95% of industrial applications and a wide range of biological systems employ elements of PID control [4]. A PID controller uses a mathematical formula with three parts to determine its output:

Controller Output = Proportional Term + Integral Term + Derivative Term

The Proportional Term is the part of the controller which is directly *proportional* to the deviation. If the deviation is large and positive (e.g. the cruise control speed setpoint is far higher than the car's current speed) the proportional response will be large and positive (e.g. floor the gas pedal).

The Integral Term is the part of the controller which takes into account how long a deviation has persisted. It is determined by taking the *integral* of the deviation over time and it is primarily used to eliminate *steady state error*. It accumulates in order to respond to small, albeit persistent deviations from the setpoint (e.g. the cruise control setpoint has been 1 mph higher than the car's speed for a few minutes).

The Derivative Term is the part of the controller which takes into account how fast the deviation is growing or shrinking. It is determined by taking the *derivative* of the deviation and serves to accelerate the controller response when the deviation is growing (e.g. speed up if the cruise control setpoint is higher than the car's speed and the car starts to slow down). It also helps reduce overshoot by decelerating the controller response when the deviation is shrinking (e.g. ease up on the gas as the car's speed starts to approach the cruise control setpoint).

The combination of these three parts, each of which can be independently tuned, gives PID controllers great flexibility at managing a wide variety of control system applications.

PID controllers work best in systems that allow some degree of lag in the response time as well as the possibility of overshoot and oscillation around the setpoint as the system attempts to stabilize itself. Reflex index systems like RAI are well suited for this type of scenario where their redemption prices can be changed by PID controllers.

More generally, it has recently been discovered that many of the current central bank monetary policy rules (e.g. the Taylor Rule) are actually approximations of PID controllers [5].

Redemption Rate Feedback Mechanism

The Redemption Rate Feedback Mechanism is the system component in charge of changing a reflex index's redemption price. In order to understand how it works, we first need to describe why the system needs a feedback mechanism as opposed to using manual control and what the mechanism's output is.

Feedback Mechanism Components

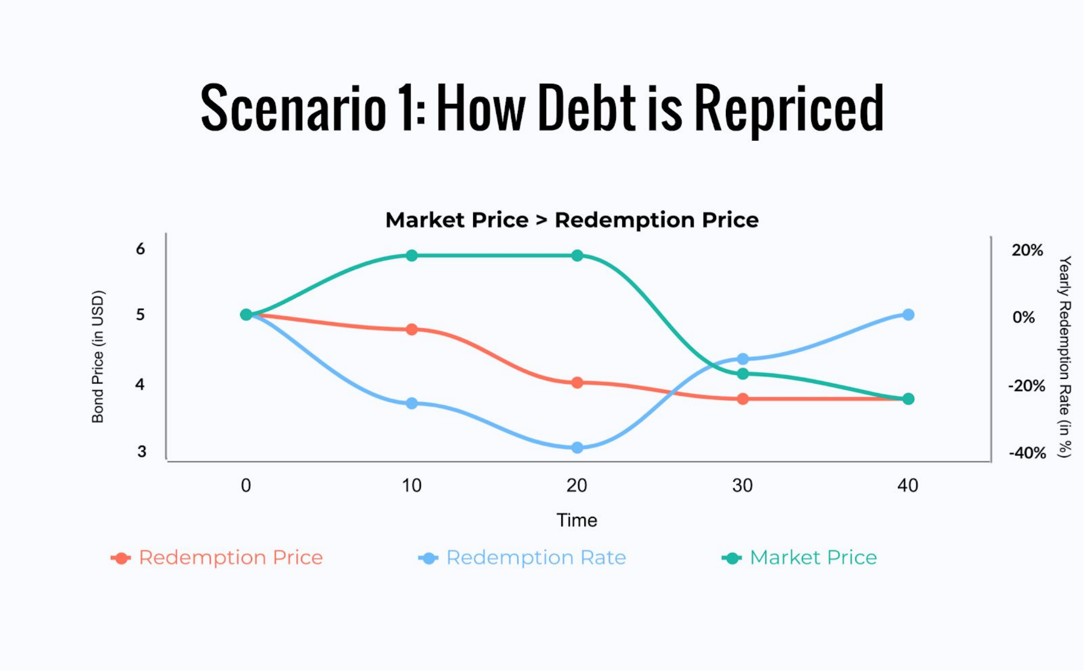
In theory, it would be possible to directly manipulate the reflex index's redemption price (described in Section 2) in order to influence index users and ultimately change the index's market price. In practice, this method would not have the desired effect on system participants. From the perspective of a SAFE holder, if the redemption price is increased only once, they might accept a higher price per debt unit, absorb the loss from a decreased collateralization ratio and maintain their position. If, however, they expect the redemption price to continue to increase over time, they would likely be more inclined to avoid expected future loss and thus choose to pay back their debt and close their positions.

We expect reflex index system participants not to respond directly to changes in the redemption price, but instead respond to the *rate of change of the redemption price* which we call the *redemption rate*. The redemption rate is set by a *feedback mechanism* that governance can fine-tune or allow to be fully automated.

Feedback Mechanism Scenarios

Recall that the feedback mechanism aims to maintain equilibrium between the redemption price and the market price by using the redemption rate to counter shifts in market forces. To achieve this, the redemption rate is calculated so that it opposes the deviation between market and redemption prices.

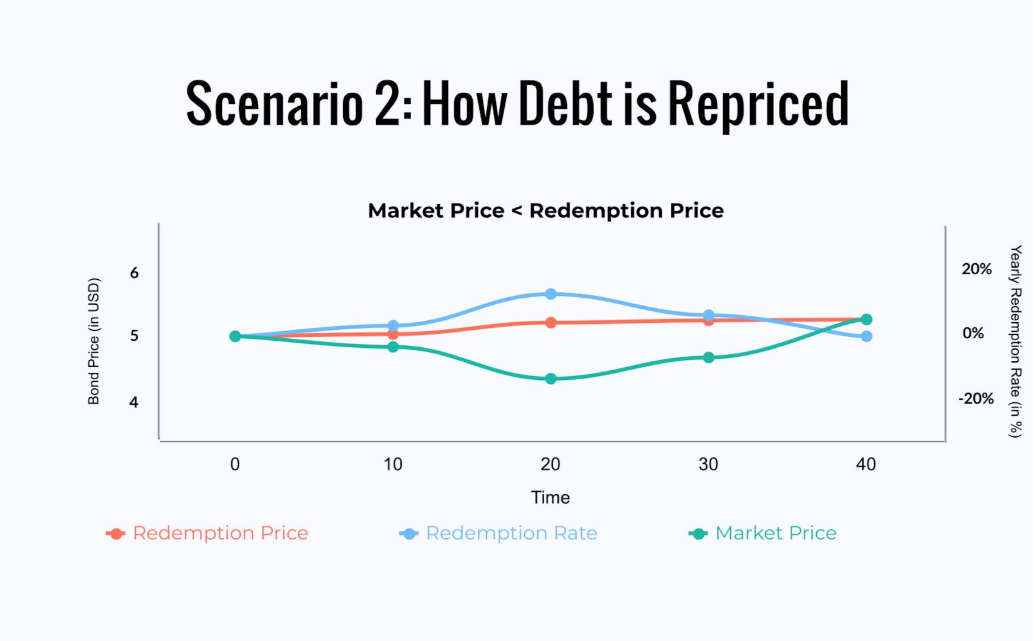
In the first scenario below, if the index's market price is higher than its redemption price, the mechanism will calculate a negative rate which will start to decrease the redemption price, thus making the system's debt cheaper.



The expectation of a decreasing redemption price will likely discourage people from holding indexes and encourage SAFE holders to generate more debt (even if the collateral price does not change) which is then sold on the market, thus balancing out supply and demand. Note that this is the ideal scenario where index holders react quickly in response to the feedback mechanism. In practice (and especially in the early days post launch) we expect a lag between the mechanism's kickoff and actual results seen in the amount of debt issued and subsequently in the market price.

On the other hand, in scenario two, if the index's market price is lower than the redemption price, the rate becomes positive and starts to reprice all the debt so that it becomes more expensive.

As debt becomes more expensive, the collateralization ratios of all SAFEs go down (thus SAFE creators are incentivized to pay back their debt) and users start to hoard indexes with the expectation that they will increase in value.



Feedback Mechanism Algorithm

In the following scenario, we assume that the protocol uses a proportional-integral controller to calculate the redemption rate:

- The reflex index is launched with an arbitrary redemption price ‘rand’
- At some point, the index’s market price rises from ‘rand’ to ’rand’ + x. After the feedback mechanism reads the new market price, it calculates a proportional term *p*, which in this case is -1 * ((‘rand’ + x) / ‘rand’). The proportional is negative in order to decrease the redemption price and in turn reprice the indexes so that they become cheaper
- After calculating the proportional, the mechanism will determine the integral term *i* by adding all the past deviations from the last *deviationInterval* seconds
- The mechanism sums the proportional and the integral and calculates a per-second redemption rate *r* that slowly starts to decrease the redemption price. As SAFE creators realize they can generate more debt, they will flood the market with more indexes

- After *n* seconds, the mechanism detects that the deviation between the market and redemption prices is negligible (under a specified parameter *noise*). At this point, the algorithm sets r to zero and keeps the redemption price where it is

In practice, the algorithm will be more robust and we will either make some variables immutable (e.g the *noise* parameter, *deviationInterval*) or there will be strict bounds over what governance can change.

Feedback Mechanism Tuning

Of the utmost importance to the proper functioning of the reflex index system is the tuning of the algorithmic controller parameters. Improper parameterization could result in the system being too slow to achieve stability, massively overshooting, or being generally unstable in the face of external shocks.

The tuning process for a PID controller typically involves running the live system, tweaking the tuning parameters, and observing the system's response, often purposefully introducing shocks along the way. Given the difficulty and financial risk of tweaking the parameters of a live reflex index system, we plan to leverage computer modeling and simulation as much as possible to set the initial parameters, but will also allow governance to update the tuning parameters if additional data from production shows them to be sub-optimal.

Money Market Setter

In RAI, we plan to keep the borrowing rate (interest rate applied when generating indexes) fixed or capped and only modify the redemption price, thus minimizing the complexity involved in modelling the feedback mechanism. The borrowing rate in our case is equal to the spread between the stability fee and DSR in Multi-Collateral DAI.

Even though we plan to keep the borrowing rate fixed, it is possible to change it alongside the redemption price using a money market setter. The money market changes the borrowing rate and the redemption price in a way that incentivizes SAFE creators to generate more or less debt. If an index's market price is above redemption, both rates will start to decrease, whereas if it is below redemption, the rates will increase.

Global Settlement

Global settlement is a method of last resort used to guarantee the redemption price to all reflex index holders. It is meant to allow both reflex index holders and SAFE creators to redeem system collateral at its net value (amount of indexes per each collateral type, according to the latest redemption price). Anyone can trigger settlement after burning a certain amount of protocol tokens.

Settlement has three main phases:

- **Trigger**: settlement is triggered, users cannot create SAFEs anymore, all collateral price feeds and the redemption price are frozen and recorded
- **Process**: process all outstanding auctions
- **Claim**: every reflex index holder and SAFE creator can claim a fixed amount of any system collateral based on the index's last recorded redemption price

Governance

The vast majority of parameters will be immutable and the inner smart contract mechanics will not be upgradeable unless governance token holders deploy an entirely new system. We chose this strategy because we can eliminate the meta-game where people try to influence the governance process for their own benefit, thus damaging trust in the system. We establish the proper operation of the protocol without putting too much faith in humans (the “bitcoin effect”) so that we maximize social scalability and minimize the risks for other developers who will want to use RAI as core infrastructure in their own projects.

For the few parameters that can be changed, we propose the addition of a Restricted Governance Module meant to delay or bound all possible system modifications. Moreover, we present Governance Ice Age, a permissions registry that can lock some parts of the system from outside control after certain deadlines have passed.

Time Bounded Governance

Time Bounded Governance is the first component of the Restricted Governance Module. It imposes time delays between changes applied to the same parameter. An example is the possibility to change the addresses of the oracles used in the Oracle Network Medianizer (Section 6.2) after at least *T* seconds have passed since the last oracle modification.

Action Bounded Governance

The second component in the Restricted Governance Module is Action Bounded Governance. Every governable parameter has limits on what values it can be set to and how much it can change over a certain period of time. Notable examples are the initial versions of the Redemption Rate Feedback Mechanism (Section 4.2) which governance token holders will be able fine-tune.

Governance Ice Age

The Ice Age is an immutable smart contract that imposes deadlines on changing specific system parameters and on upgrading the protocol. It can be used in the case where governance wants to make sure they can fix bugs before the protocol locks itself and denies outside intervention. Ice Age will verify if a change is permitted by checking the parameter's name and the affected contract's address against a registry of deadlines. If the deadline has passed, the call will revert.

Governance may be able to delay Ice Age a fixed number of times if bugs are found close to the date when the protocol should start to lock itself. For example, Ice Age can only be delayed three times, each time for one month, so that the newly implemented bug fixes are tested properly.

Core Areas Where Governance Is Needed

We envision four areas where governance might be needed, especially in the early versions of this framework:

- **Adding new collateral types**: RAI will be backed only by ETH, but other indexes will be backed by multiple collateral types and governance will be able to diversify risk over time

- **Changing external dependencies**: oracles and DEXs that the system depends on can be upgraded. Governance can point the system to newer dependencies in order for it to continue functioning properly

- **Fine-tuning rate setters**: early monetary policy controllers will have parameters that can be changed within reasonable bounds (as described by Action and Time Bounded Governance)

- **Migrating between system versions**: in some cases, governance can deploy a new system, give it permission to print protocol tokens and withdraw this permission from an old system. This migration is performed with the help of the Restricted Migration Module outlined below

Restricted Migration Module

The following is a simple mechanism for migrating between system versions:

- There is a migration registry that keeps track of how many different systems the same protocol token covers and which systems can be denied the permission to print protocol tokens in a debt auction

- Every time governance deploys a new system version, they submit the address of the system's debt auction contract in the migration registry. Governance also needs to specify if they will ever be able to stop the system from printing protocol tokens. Also, governance can, at any time, say that one system will always be able to print tokens and thus it will never be migrated from

- There is a cooldown period between proposing a new system and withdrawing permissions from an old one

- An optional contract can be set up so that it automatically shuts down an old system after it is denied printing permissions

The migration module can be combined with an Ice Age that automatically gives specific systems the permission to always be able to print tokens.

Automatic System Shutdown

There are cases that the system can automatically detect and as a result trigger settlement by itself, without the need to burn protocol tokens:

- **Severe Price Feed Delays**: the system detects that one or more of the collateral or index price feeds have not been updated in a long time

- **System Migration**: this is an optional contract that can shut down the protocol after a cooldown period passes from the moment when governance withdraws the ability of the debt auction mechanism to print protocol tokens (Restricted Migration Module, Section 5.4.1)

- **Consistent Market Price Deviation**: the system detects that the index's market price has been *x%* deviated for a long time compared to the redemption price

Governance will be able to upgrade these autonomous shutdown modules while still being bounded or until the Ice Age starts to lock some parts of the system.

Oracles

There are three main asset types that the system needs to read price feeds for: the index, the protocol token and every whitelisted collateral type. The price feeds can be provided by governance led oracles or by already established oracle networks.

Governance Led Oracles

Governance token holders or the core team that launched the protocol can partner with other entities who gather multiple price feeds off-chain and then submit a single transaction to a smart contract that medianizes all data points.

This approach allows for more flexibility on upgrading and changing the oracle infrastructure although it comes at the expense of trustlessness.

Oracle Network Medianizer

An oracle network medianizer is a smart contract that reads prices from multiple sources which are not directly controlled by governance (e.g Uniswap V2 pool between an index collateral type and other stablecoins) and then medianizes all the results. ONM works as follows:

- Our contract keeps track of whitelisted oracle networks it can call in order to request collateral prices. The contract is funded by part of the surplus the system accrues (using the Surplus Treasury, Section 11). Each oracle network accepts specific tokens as payment so our contract also keeps track of the minimum amount and the type of tokens needed for each request

- In order to push a new price feed in the system, all the oracles need to be called beforehand. When calling an oracle, the contract first swaps some stability fees with one of the oracle's accepted tokens. After an oracle is called, the contract tags the call as "valid" or "invalid". If a call is invalid, the specific faulty oracle cannot be called again until all the other ones are called and the contract checks if there is a valid majority. A valid oracle call must not revert and it must retrieve a price that has been posted on-chain sometime in the last *m* seconds. "Retrieve" means different things depending on each oracle type:

    - For pull based oracles, from which we can get a result right away, our contract needs to pay a fee and directly fetch the price

    - For push based oracles, our contract pays the fee, calls the oracle and needs to wait a specific period of time *n* before calling the oracle again in order to get the requested price

- Every oracle result is saved in an array. After every whitelisted oracle is called and if the array has enough valid data points to form a majority (e.g the contract received valid data from 3/5 oracles), the results are sorted and the contract chooses the median

- Whether the contract finds a majority or not, the array with oracle results is cleared and the contract will need to wait *p* seconds before starting the entire process all over again

Oracle Network Backup

Governance can add a backup oracle option that starts to push prices in the system if the medianizer cannot find a majority of valid oracle networks several times in a row.

The backup option must be set when the medianizer is deployed as it cannot be changed afterwards. Furthermore, a separate contract can monitor if the backup has been replacing the medianization mechanism for too long and automatically shut down the protocol.

Safes

In order to generate indexes, anyone can deposit and leverage their crypto collateral inside Safes. While a SAFE is opened, it will continue accruing debt according to the deposited collateral's borrowing rate. As the SAFE creator pays back their debt, they will be able to withdraw more and more of their locked collateral.

SAFE Lifecycle

There are four main steps needed for creating reflex indexes and subsequently paying back a SAFE's debt:

- Deposit collateral in the SAFE

  The user first needs to create a new SAFE and deposit collateral in it.

- Generate indexes backed by the SAFE's collateral

  The user specifies how many indexes they want to generate. The system creates an equal amount of debt that starts to accrue according to the collateral's borrowing rate.

- Pay back the SAFE debt

  When the SAFE creator wants to withdraw their collateral, they have to pay back their initial debt plus the accrued interest.

- Withdraw collateral

After the user pays back some or all of their debt, they are allowed to withdraw their collateral.

SAFE Liquidation

In order to keep the system solvent and cover the value of the entire outstanding debt, each SAFE can be liquidated in case its collateralization ratio falls under a certain threshold. Anyone can trigger a liquidation, in which case the system will confiscate the SAFE's collateral and sell it off in a *collateral auction*.

Liquidation Insurance

In one version of the system, SAFE creators can have the option to choose a *trigger* for when their SAFEs get liquidated. Triggers are smart contracts that automatically add more collateral in a SAFE and potentially save it from liquidation. Examples of triggers are contracts that sell short positions or contracts that communicate with insurance protocols such as Nexus Mutual [6].

Another method to protect SAFEs is the addition of two different collateralization thresholds: *safe* and *risk*. SAFE users can generate debt until they hit the safe threshold (which is higher than risk) and they only get liquidated when the SAFE's collateralization goes below the risk threshold.

Collateral Auctions

To start a collateral auction, the system needs to use a variable called *liquidationQuantity* in order to determine the amount of debt to be covered by every auction and the corresponding amount of collateral to be sold. A *liquidation penalty* will be applied to every auctioned SAFE.

Collateral Auction Parameters

Parameter Name	Description
minimumBid	Minimum amount of coins that need to be offered in one bid
discount	Discount at which collateral is being sold

lowerCollateralMedianDeviation	Max lower bound deviation that the collateral median can have compared to the oracle price
upperCollateralMedianDeviation	Max upper bound deviation that the collateral median can have compared to the oracle price
lowerSystemCoinMedianDeviation	Max lower bound deviation that the system coin oracle price feed can have compared to the system coin oracle price
upperSystemCoinMedianDeviation	Max upper bound deviation that the collateral median can have compared to the system coin oracle price
minSystemCoinMedianDeviation	Min deviation for the system coin median result compared to the redemption price in order to take the median into account

Collateral Auction Mechanism

The fixed discount auction is a straightforward way (compared to English auctions) to put collateral up for sale in exchange for system coins used to settle bad debt. Bidders are only required to allow the auction house to transfer their safeEngine.coinBalance and can then call buyCollateral in order to exchange their system coins for collateral which is sold at a discount compared to its latest recorded market price.

Bidders can also review the amount of collateral they can get from a specific auction by calling getCollateralBought or getApproximateCollateralBought. Note that getCollateralBought is not marked as view because it reads (and also updates) the redemptionPrice from the oracle relayer whereas getApproximateCollateralBought uses the lastReadRedemptionPrice.

Debt Auctions

In the scenario where a collateral auction cannot cover all the bad debt in a SAFE and if the system does not have any surplus reserves, anyone can trigger a debt auction.

Debt auctions are meant to mint more protocol tokens (Section 10) and sell them for indexes that can nullify the system's remaining bad debt.

In order to start a debt auction, the system needs to use two parameters:

- initialDebtAuctionAmount: the initial amount of protocol tokens to mint post-auction

- debtAuctionBidSize: the initial bid size (how many indexes must be offered in exchange for *initialDebtAuctionAmount* protocol tokens)

Autonomous Debt Auction Parameter Setting

The initial amount of protocol tokens minted in a debt auction can either be set through a governance vote or it can be automatically adjusted by the system. An automated version would need to be integrated with oracles (Section 6) from which the system would read the protocol token and reflex index market prices. The system would then set the initial amount of protocol tokens (*initialDebtAuctionAmount*) that will be minted for *debtAuctionBidSize* indexes. *initialDebtAuctionAmount* can be set at a discount compared to the actual PROTOCOL/INDEX market price in order to incentivize bidding.

Debt Auction Parameters

Parameter Name	Description
amountSoldIncrease	Increase in the amount of protocol tokens to be minted for the same amount of indexes
bidDecrease	Next bid's minimum decrease in the accepted amount of protocol tokens for the same amount of indexes
bidDuration	How long the bidding lasts after a new bid is submitted (in seconds)
totalAuctionLength	Total length of the auction (in seconds)
auctionsStarted	How many auctions have started until now

Debt Auction Mechanism

As opposed to collateral auctions, debt auctions only have one stage:

`decreaseSoldAmount(uint id, uint amountToBuy, uint bid)`: decrease the amount of protocol tokens accepted in exchange for a fixed amount of indexes.

The auction will be restarted if it has no bids placed. Every time it restarts, the system will offer more protocol tokens for the same amount of indexes. The new protocol token amount is calculated as *lastTokenAmount* * *amountSoldIncrease* / 100. After the auction settles, the system will mint tokens for the highest bidder.

Protocol Tokens

As described in earlier sections, each protocol will need to be protected by a token that is minted through debt auctions. Apart from protection, the token will be used to govern a few system components. Also, the protocol token supply will gradually be reduced with the use of surplus auctions. The amount of surplus that needs to accrue in the system before extra funds are auctioned is called the *surplusBuffer* and it is automatically adjusted as a percentage of the total debt issued.

Insurance Fund

Apart from the protocol token, governance can create an insurance fund that holds a wide array of uncorrelated assets and which can be used as a backstop for debt auctions.

Surplus Auctions

Surplus auctions sell stability fees accrued in the system for protocol tokens that are then burned.

Surplus Auction Parameters

Parameter Name	Description
bidIncrease	Minimum increase in the next bid
bidDuration	How long the auction lasts after a new bid is submitted (in seconds)
totalAuctionLength	Total length of the auction (in seconds)
auctionsStarted	How many auctions have started until now

Surplus Auction Mechanism

Surplus auctions have a single stage:

increaseBidSize(uint id, uint amountToBuy, uint bid): anyone can bid a higher amount of protocol tokens for the same amount of indexes (surplus). Every new bid needs to be higher than or equal to *lastBid* * *bidIncrease* / 100. The auction will end after maximum *totalAuctionLength* seconds or after *bidDuration* seconds have passed since the latest bid and no new bids have been submitted in the meantime.

An auction will restart if it has no bids. On the other hand, if the auction has at least one bid, the system will offer the surplus to the highest bidder and will then burn all the gathered protocol tokens.

Surplus Indexes Management

Every time a user generates indexes and implicitly creates debt, the system starts applying a borrowing rate to the user's SAFE. The accrued interest is pooled in two different smart contracts:

- The *accounting engine* used to trigger debt (Section 9.2) and surplus (Section 10.1) auctions
- The *surplus treasury* used to fund core infrastructure components and incentivize external actors to maintain the system

The surplus treasury is in charge of funding three core system components:

- Oracle module (Section 6). Depending on how an oracle is structured, the treasury either pays governance whitelisted, off-chain oracles or it pays for calls toward oracle networks. The treasury can also be set up to pay the addresses that spent gas to call an oracle and update it

- In some cases, independent teams that maintain the system. Examples are teams who whitelist new collateral types or fine tune the system's rate setter (Section 4.2)

The treasury can be set up so that some surplus recipients will automatically be denied funding in the future and others can take their place.

External Actors

The system depends on external actors in order to function properly. These actors are economically incentivized to participate in areas such as auctions, global settlement processing, market making and updating price feeds in order to maintain the system's health.

We will provide initial user interfaces and automated scripts to enable as many people as possible to keep the protocol secure.

Addressable Market

We see RAI as being useful in two main areas:

- **Portfolio diversification**: investors use RAI to get dampened exposure to an asset like ETH without the whole risk of actually holding ether

- **Collateral for synthetic assets**: RAI can offer protocols such as UMA, MakerDAO and Synthetix a lower exposure to the crypto market and give users more time to exit their positions in the case of scenarios such as Black Thursday from March 2020 when millions of dollars worth of crypto assets were liquidated

Future Research

To push the boundaries of decentralized money and bring further innovation in decentralized finance, we will continue to look for alternatives in core areas such as governance minimization and liquidation mechanisms.

We first want to lay the groundwork for future standards around protocols that lock themselves from outside control and for true “money robots” which adapt in response to market forces. Afterwards, we invite the Ethereum community to debate and design improvements around our proposals with a specific focus on collateral and debt auctions.

Risks and Mitigation

There are several risks involved in developing and launching a reflex index, as well as subsequent systems that are built on top:

- **Smart contract bugs**: the greatest risk posed to the system is the possibility of a bug that allows anyone to extract all the collateral or locks the protocol in a state it cannot recover from. We plan to have our code reviewed by multiple security researchers and launch the system on a testnet before we commit to deploying it in production

- **Oracle failure**: we will aggregate feeds from multiple oracle networks and there will be strict rules in place for upgrading only one oracle at a time so that malicious governance cannot easily introduce false prices

- **Collateral black swan events**: there is the risk of a black swan event in the underlying collateral which can result in a high amount of liquidated SAFEs. Liquidations may not be able to cover the entire outstanding bad debt and so the system will continuously change its surplus buffer in order to cover a decent amount of issued debt and withstand market shocks

- **Improper rate setter parameters**: autonomous feedback mechanisms are highly experimental and may not behave exactly like we predict during simulations. We plan to allow governance to fine-tune this component (while still being bounded) in order to avoid unexpected scenarios

- **Failure to bootstrap a healthy liquidator market**: liquidators are vital actors that make sure all issued debt is covered by collateral. We plan to create interfaces and automated scripts so that as many people as possible can participate in keeping the system secure.

Summary

We have proposed a protocol that progressively locks itself from human control and issues a low volatility, collateralized asset called a reflex index. We first presented the autonomous mechanism meant to influence the index's market price and then described how several smart contracts can limit the power that token holders have over the system. We outlined a self-sustaining scheme for medianizing price feeds from multiple independent oracle networks and then finished by presenting the general mechanism for minting indexes and liquidating SAFEs.

References

[1] “The Maker Protocol: MakerDAO’s Multi Collateral Dai (MCD) System”, https://bit.ly/2YL5S6j

[2] “UMA: A Decentralized Financial Contract Platform”, https://bit.ly/2Wgx7E1

[3] Synthetix Litepaper, https://bit.ly/2SNHxZO

[4] K.J. Åström, R.M. Murray, “Feedback Systems: An Introduction for Scientists and Engineers”, https://bit.ly/3bHwnMC

[5] R.J. Hawkins, J.K. Speakes, D.E. Hamilton, “Monetary Policy and PID Control”, https://bit.ly/2TeQZFO

[6] H. Karp, R. Melbardis, “A peer-to-peer discretionary mutual on the Ethereum blockchain”, https://bit.ly/3du8TMy

[7] H. Adams, N. Zinsmeister, D. Robinson, “Uniswap V2 Core”, https://bit.ly/3dqzNEU

Glossary

Reflex index: a collateralized asset that dampens the volatility of its underlying

RAI: our first reflex index

Redemption Price: the price that the system wants the index to have. It changes, influenced by a redemption rate (computed by RRFM), in case the market price is not close to it. Meant to influence SAFE creators to generate more or pay back some of their debt

Borrowing Rate: annual interest rate applied to all SAFEs that have outstanding debt

Redemption Rate Feedback Mechanism (RRFM): an autonomous mechanism which compares the market and redemption prices of a reflex index and then computes a redemption rate that slowly influences SAFE creators to generate more or less debt (and implicitly tries to minimize the market/redemption price deviation)

Money Market Setter (MMS): a mechanism similar to RRFM which pulls multiple monetary levers at once. In the case of reflex indexes, it modifies both the borrowing rate and the redemption price

Oracle Network Medianizer (ONM): a smart contract that pulls prices from multiple oracle networks (which are not controlled by governance) and medianizes them if a majority (e.g 3 out of 5) returned a result without throwing

Restricted Governance Module (RGM): a set of smart contracts that bound the power that governance tokens holders have over the system. It either enforces time delays or limits the possibilities that governance has to set certain parameters

Governance Ice Age: immutable contract that locks most components of a protocol from outside intervention after a certain deadline has passed

Accounting Engine: system component which triggers debt and surplus auctions. It also keeps track of the amount of currently auctioned debt, unactioned bad debt and the surplus buffer

Surplus Buffer: amount of interest to accrue and keep in the system. Any interest accrued above this threshold gets sold in surplus auctions that burn protocol tokens

Surplus Treasury: contract that gives permission to different system modules to withdraw accrued interest (e.g ONM for oracle calls)