Parallel Protocol

MIMO Capital
mimo.capital

March 8th, 2021

Abstract

The Parallel Protocol is a decentralized stablecoin issuance protocol on the Ethereum blockchain. Parallel stablecoins are decentralized, collateral-backed synthetic assets pegged to a fiat currency. Parallel stablecoins are kept stable by collateral locked in smart contract Vaults. At launch, the Parallel Protocol offers a single stablecoin called PAR which is pegged to the Euro. Over time, the Parallel Protocol will progressively decentralize itself, handing over control to a diverse community of people holding the MIMO governance token.

Contents

1 Motivation

1.1 Why a new stablecoin?

Crypto has undergone multiple cycles of adoption ("bubbles") over the last 11 years since Bitcoin's inception in 2009. The early days were primarily dominated by technically sophisticated users ("Innovators") as the industry was immature and required strong tech skills. This culminated in the bubble of Jan 2014. Over the next 3 years, crypto companies improved usability and product offerings, which subsequently allowed a new group of users to enter our industry ("Early Adopters"). These early adopters were strong risk takers and were attracted by the early stage investment opportunities in the form of ICOs. The funding during the ICO bubble shaped the way for new companies to emerge and further improve on usability and products.

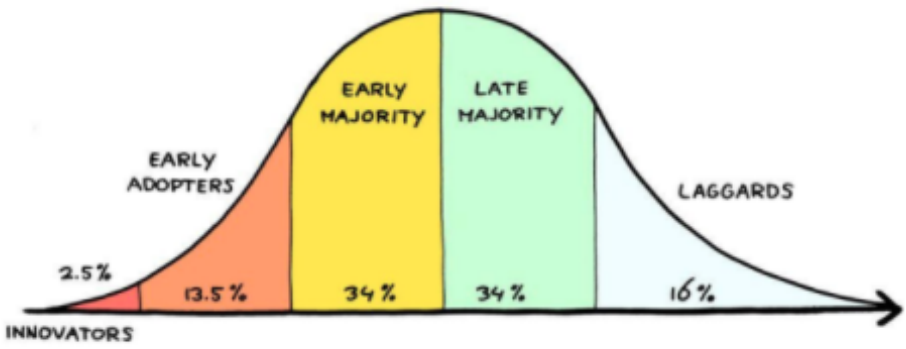

Figure 1: Technology adoption curve

Today we are at the brink of a new group of users coming into our industry ("Early Majority"). Their expectations are again different to the previous user group. Early adopters were very comfortable with high risk and wanted to "own" a piece of this new industry. These users established crypto-currencies and tokens as its own asset class over the last 4 years. The new users coming into our industry today have a lower risk tolerance, which can be seen by the increase of stablecoin usage for transfers compared to using volatile crypto-currencies for transfers. At the same time, these new users are not looking for high-risk high-return, but instead they are looking for returns denominated in their own currency. In a sense we are witnessing a change of mindset from 'owning crypto as asset class' for its high-risk-high-returns towards 'make crypto work for me' with returns denominated in fiat.

The crypto industry has matured to offer products to these new users in the form of savings & loans. While loans are still heavily used by the previous user group to facilitate leveraged trading and hedging in both the trusted world of centralized service providers, and increasingly by using decentralized finance ("DeFi") offerings; the new user group are very strongly

attracted to the savings rates of 5-15% APY and higher.

MIMO has a unique opportunity to bridge the existing chasm between the DeFi world and trusted world of regulated financial institutions by offering a fully decentralized stablecoin platform with savings & loans. This will allow us to both move fast and leverage the existing trends in DeFi and offer a highly competitive product to our crypto-curious ("Early Majority") user group.

1.2 How PAR is different

Here are the features that make the Parallel Protocol different from the competition.

1.2.1 The first decentralized EUR stablecoin

Today's stablecoin market is dominated exclusively by USD denominated stablecoins (Tether, Circle, Maker, Paxos, etc). Trusted stablecoin issuers are facing negative interest rates in Europe, which is undermining the viability of their business model of interest rate arbitrage. So far no decentralized EUR stablecoin exists, which opens up a big opportunity to win the entire EUR stablecoin market. PAR will be the first decentralized EUR stablecoin.

1.2.2 Growing liquidity

A proportion of the fees collected from borrowers are used to incentivize liquidity in a handful of PAR Automated Market Maker (AMM) pools. This incentive will attract more liquidity to PAR pools, facilitating low-slippage trades between PAR and other cryptoassets given the deeper market depth.

1.2.3 Optimizing yield

Liquidity providers can earn attractive rates by adding their liquidity (e.g. PAR and ETH) to an AMM pool. Liquidity providers will continuously receive the pool's trading fees. In addition, liquidity providers will also receive additional income collected through protocol fees such as loan origination and borrowing fees.

2 Protocol Specification

2.1 Overview

The core of the Parallel Protocol are **Vaults**. Users mint **PAR** by depositing **collateral** such as Ether (ETH) into the Vault smart contract. The steps involved to mint new PAR are as follows:

- A Borrower deposits collateral and a new Vault is automatically created.
- Based on the Vault's collateral balance, a Borrower can borrow up to a certain amount of PAR. The vault must be collateralized with more than a **Minimum Collateralization Ratio (MCR)** for borrowing. For example, an MCR of 150% means borrowers need 150% collateral deposited before they can borrow.
- A separate liquidation MCR (**Liquidation Ratio (LR)**) is used to calculate for liquidations. For example, an LR of 130% means vaults with an MCR below 130% can be liquidated. Both ratios for initial borrowing and liquidations are configured per collateral type.
- The PAR smart contracts mints the borrowed amount of PAR to the Borrower.
- An **Origination Fee** is applied for newly created debt.
- PAR are ERC20 tokens that you can transfer and use normally, pegged to the EUR fiat currency.
- A **Borrowing Fee** accrues over time on all active Vaults, which has to be fully repaid before the Borrower can withdraw their collateral.
- A **Health Factor** is the ratio between a vault's current and minimum MCR (or LR.) If a Vault's liquidation health factor goes below a minimum value due to market changes, profit-seeking Liquidators can liquidate the **undercollateralized** Vault to receive its collateral at a discount.
- Borrowers need to retain enough collateral in their Vaults to borrow additional funds and avoid being liquidated.
- Borrowers can close their vault and withdraw their collateral by fully repaying their debt and any accrued fees.
- Liquidators earn a liquidation bonus for liquidating underwater vaults.
- A **Liquidation Fee** is charged to the borrower during liquidation, which is added to the outstanding debt.

There are no counterparties involved in this exchange, as actors in the network transact directly with the PAR smart contracts. Vaults are non-custodial with each Borrower having full control over their collateral and borrow balances, provided it meets the minimum health factor.

2.2 System Guarantees

The following properties must be true to preserve the integrity of the Parallel Protocol.

2.2.1 All PAR are always backed by collateral

The current total supply of PAR must always be backed by a greater total value of collateral:

$$PAR.totalSupply() < value(totalCollateral) \tag{1}$$

2.2.2 PAR Circulating supply and fees equals total debt

In the Parallel protocol, borrowing and other fees collected are minted as PAR and distributed to different protocol actors. The sum of the supply of PAR and accrued fees must be equal to the total debt managed by the protocol:

$$PAR.totalSupply() + mintableFeeIncome = totalDebt() \tag{2}$$

2.3 Smart Contract Architecture

The Parallel Protocol is split to the following contracts:

- **Vaults Core**: The main contract to interact with the Parallel protocol. All calculations happen here.
- **Vaults Core State**: Owns the global state (cumulative rates last refresh) for each collateral type available to borrow against. All state updates & calculations happen here
- **PAR**: PAR are standard ERC20 tokens, whose value is pegged to the EUR fiat currency.
- **Rates Manager**: Stateless. Calculates debt and fees for each Vault.
- **Liquidation Manager**: Stateless. Calculates health factors and liquidation variables.
- **PriceFeed**: Responsible for retrieving collateral prices in fiat, for calculating Vault health factors.
- **FeeDistributor**: Responsible for collecting and distributing protocol fees.
- **AddressProvider**: Indexes all module addresses read by other modules.

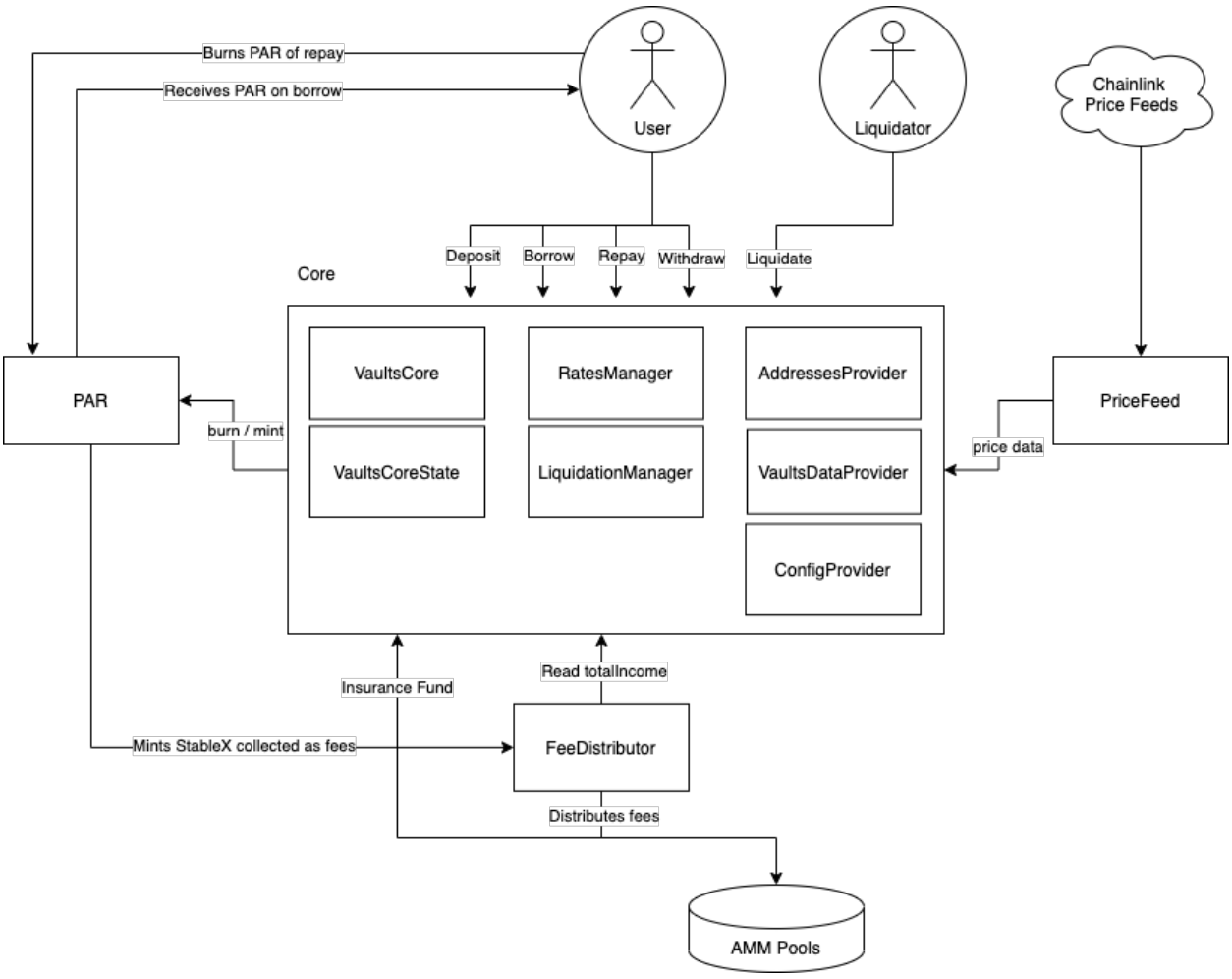

Figure 2: PAR architecture

- **ConfigProvider**: Responsible for updating protocol config parameters.
- **VaultsDataProvider**: Owns vault state, base debt, and vault related read functions.

2.4 Vaults Core

The VaultsCore contract is the main interface for the user to interact with the Parallel protocol such as depositing, borrowing, repaying, and liquidating. It stores the collateral, the insurance fund and handles all calculations.

The Vaults Core contract has the following interface:

```
interface IVaultsCore {
  function deposit(address _collateralType, uint256 _amount) external;
  function depositByVaultId(uint256 _vaultId, uint256 _amount) external;
  function depositAndBorrow(address _collateralType, uint256 _depositAmount,
      uint256 _borrowAmount) external;
  function withdraw(uint256 _vaultId, uint256 _amount) external;
```

```
  function withdrawAll(uint256 _vaultId) external;
  function borrow(uint256 _vaultId, uint256 _amount) external;
  function repayAll(uint256 _vaultId) external;
  function repay(uint256 _vaultId, uint256 _amount) external;
  function liquidate(uint256 _vaultId) external;

  ...
}
```

In addition to ERC20 collateral types, VaultsCore also supports ETH directly through functions such as depositETH, withdrawETH, and others.

2.4.1 Depositing

Depositing collateral increases the amount of PAR you can borrow.

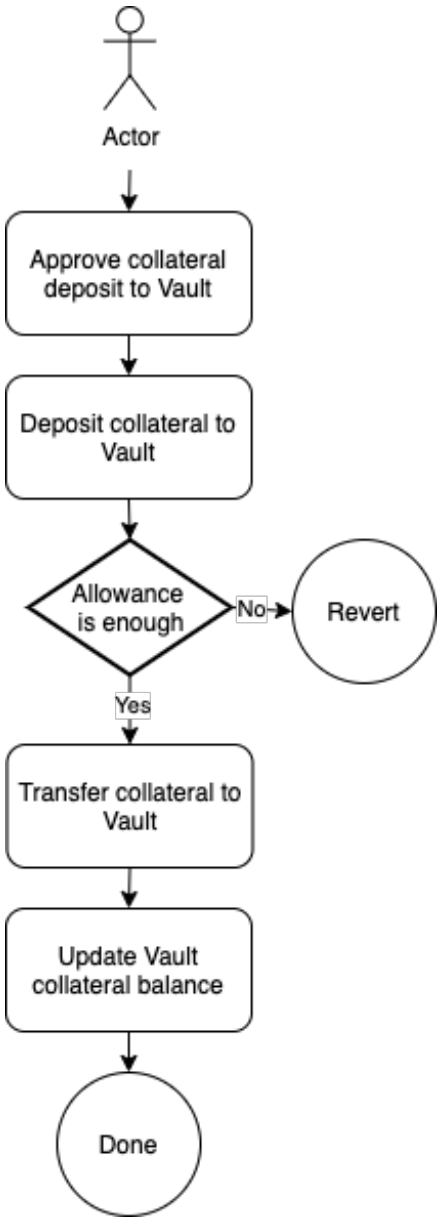

Figure 3: Opening a vault and Depositing collateral

2.4.2 Borrowing

Borrowing alters the total supply of outstanding PAR. When you borrow, the Vaults contract mints new PAR tokens for the Borrower.

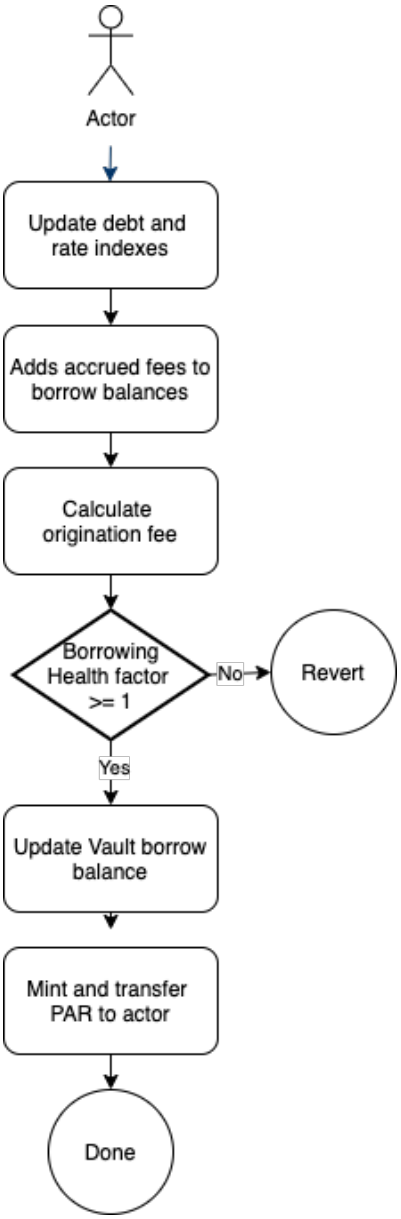

Figure 4: Borrowing PAR

Borrowers can continue to mint PAR as long as they deposit a greater value of collateral in their vault. This guarantees that all outstanding PAR are fully backed.

2.4.3 Withdrawing

Users can withdraw their collateral as long as it does not decrease the Vault's health factor below the minimum amount. This function is limited to the Vault's owner.

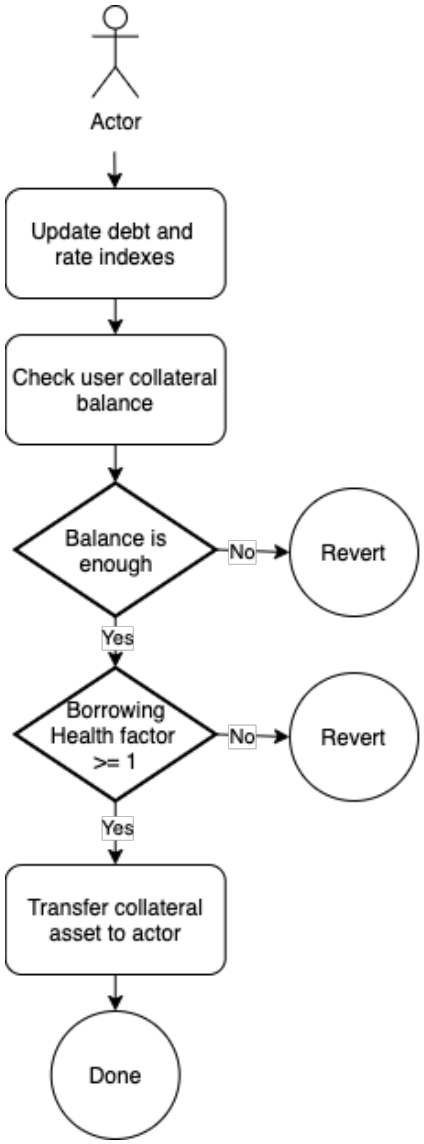

Figure 5: Withdrawing collateral

2.4.4 Repaying

Users can repay their debt partially or fully. PAR tokens are burned when a loan is repaid.

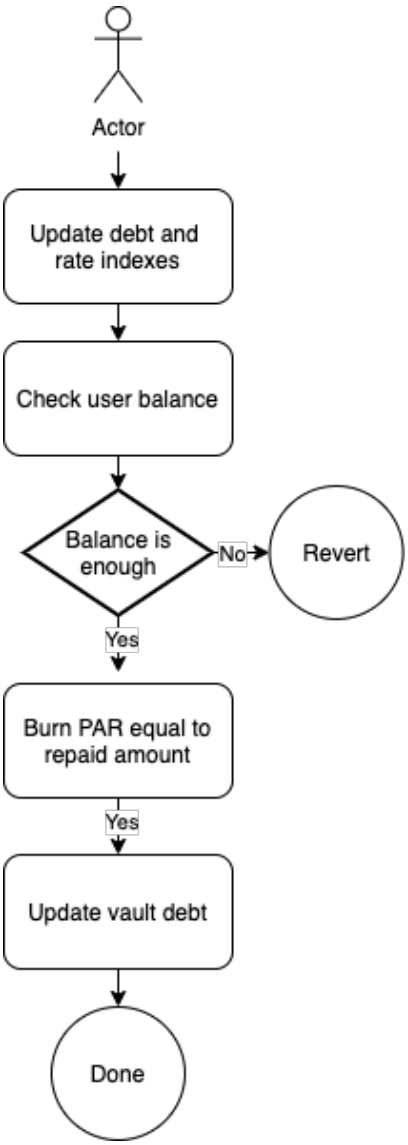

Figure 6: Repaying a vault

2.4.5 Liquidation

Liquidation ensures that there is always sufficient collateral to cover all PAR tokens. Vaults below a specified health factor is subject to liquidation by profit-seeking actors in the network. An insurance fund is funded by platform fees to guarantee liquidations even when the value of the collateral drops below the level of outstanding vault debt. The Liquidation Ratio (LR) acts as a buffer to avoid unnecessary use of the insurance fund.

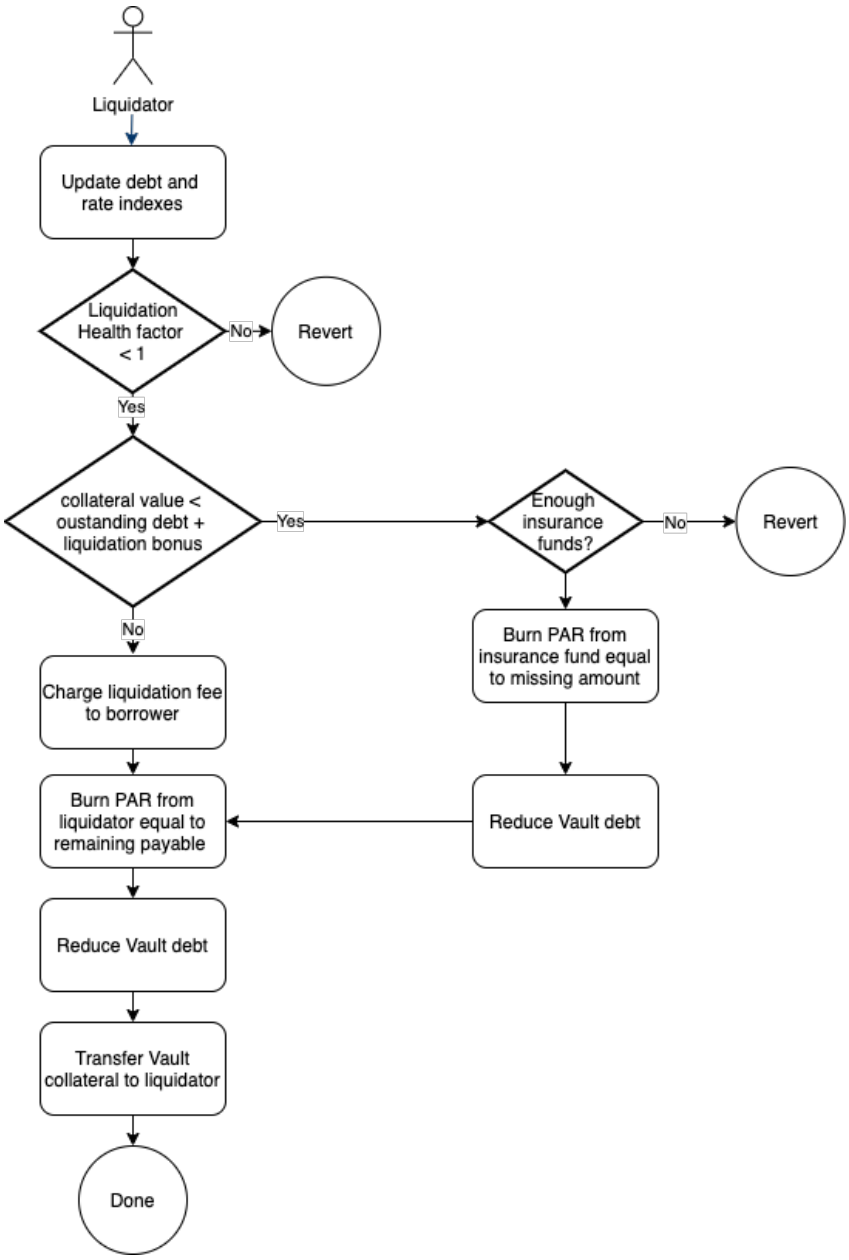

Figure 7: Liquidating a vault

A **liquidation fee** is charged to the borrower during liquidation, added to the outstanding debt. This liquidation fee is not charged in cases where the collateral value is less than the outstanding debt as the difference is covered by the insurance fund and any additional fee would be directly charged to the insurance fund and subsequently become income. In edge cases where the collateral is sufficient to cover the outstanding debt, but not the liquidation fee, the liquidation fee is reduced to avoid charging the insurance fund unnecessarily.

2.5 Rates Manager

The Rates module is responsible for calculating the outstanding debt and fees for each Vault. Vaults accrue fees over time. The Parallel Protocol uses the following formula to efficiently calculate the debt and accrued fees of all Vaults:

$$VaultTotalDebt = VaultBaseDebt * CumulativeRate \tag{3}$$

A vault's total debt is the amount that borrowers will have to fully repay before the vault's collateral can be fully withdrawn. This includes both the principal amount deposited and fees accrued over the length of the loan. This value is not stored anywhere, and is calculated dynamically based on **Vault Base Debt** and **Cumulative Rate**.

2.5.1 Vault Base Debt

Vault Base Debt is a variable unique to each Vault. It's a partial representation of a vault's debt. Vault Base Debt is updated whenever a vault's debt is added / removed, and is calculated as:

$$NewVaultBaseDebt = OldVaultBaseDebt \pm \frac{\Delta TotalVaultDebt}{CumulativeRate} \tag{4}$$

Note that the resulting change in Vault Base Debt does not equal the change in Total Vault Debt. The change in Vault Base Debt is a function of both Total Vault Debt and the Collateral Cumulative Rate.

2.5.2 Cumulative Rate

Collateral Cumulative Rate is a variable unique to each Collateral type (e.g. ETH.) It's a representation of the aggregate borrowing fees over time. It's calculated with a 1 sec precision based on the block-time.

Cumulative Rate is updated whenever the 'VaultsCore.refreshCollateral()' function is called, which will update the value to the current time. Contracts that rely on the Cumulative Rate will call this function at the start of their transaction.

Cumulative Rate is calculated as:

$$NewCumulativeRate = OldCumulativeRate * (1 + BorrowRate)^{Timesincelastupdate} \quad (5)$$

When a new collateral is added, its Cumulative Rate is set to 1. A positive borrowing fee will increase the Cumulative Rate over time.

2.6 Liquidation Manager

The LiquidationManager is responsible for calculating health factors used to determine if a Vault is open for liquidation. It also calculates liquidation bonuses and discounts.

2.7 Fee Distributor

Whenever PAR are borrowed and repaid, fees are collected at the protocol level. Fees include the **origination fee**, **borrowing fee** and the **liquidation fee**, all of which are collected from borrowers. Income payable is calculated using the same formulas as borrowing fees. Fees collected are distributed to a safety reserve and incentivized liquidity pools.

New income is always equal to the total debt oustanding minus the totalSupply of the stablecoin.

$$NewIncome = TotalDebt - par.totalSupply() \quad (6)$$

At launch, fees are distributed as follows:

- 10% of fees collected go to the safety reserve used to cover severely unhealthy vaults.
- 90% of fees collected are used to incentivized PAR AMM pools with the goal of encouraging liquidity for PAR tokens.

The **FeeDistributor** module is responsible for collecting and distributing fees.

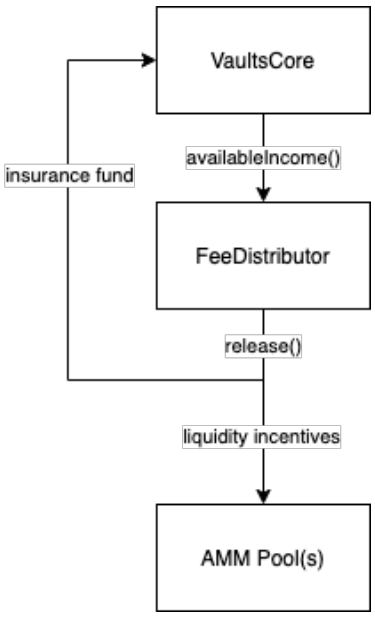

Figure 8: Fee Distributor

2.7.1 Safety Reserve

All vaults in the Parallel Protocol are covered by an insurance fund. The safety reserve comes into use when a vault that faces liquidation does not have enough collateral to pay for the entire outstanding debt. The safety reserve will cover the difference in those cases.

Liquidation will simply fail if the safety reserve can not cover it. Once enough fees have been collected liquidation can proceed. During this time, the safety reserve will take on the volatility risk of the collateral.

2.7.2 Incentivized Liquidity Pools

The Parallel Protocol incentivizes its users to act in the best interest of the network as it grows. The protocol distributes a percentage of all fees collected from borrowers to the liquidity providers of PAR AMM pools. This ensures that there's always sufficient liquidity for PAR tokens.

These AMM pools offer a savings product that earns optimized yield based on both swap fees and the vault fees collected from borrowers.

2.8 Price Feed

The Parallel Protocol requires up-to-date information about the price of the Vaults' collateral assets in order to calculate the health factors of vaults. Real-time price data lets you

find out if a vault is open for liquidation.

For this purpose, the Parallel Protocol uses Chainlink's Price Reference Data feeds to get prices of assets in fiat (USD) and fiat exchange rates (EUR / USD.)

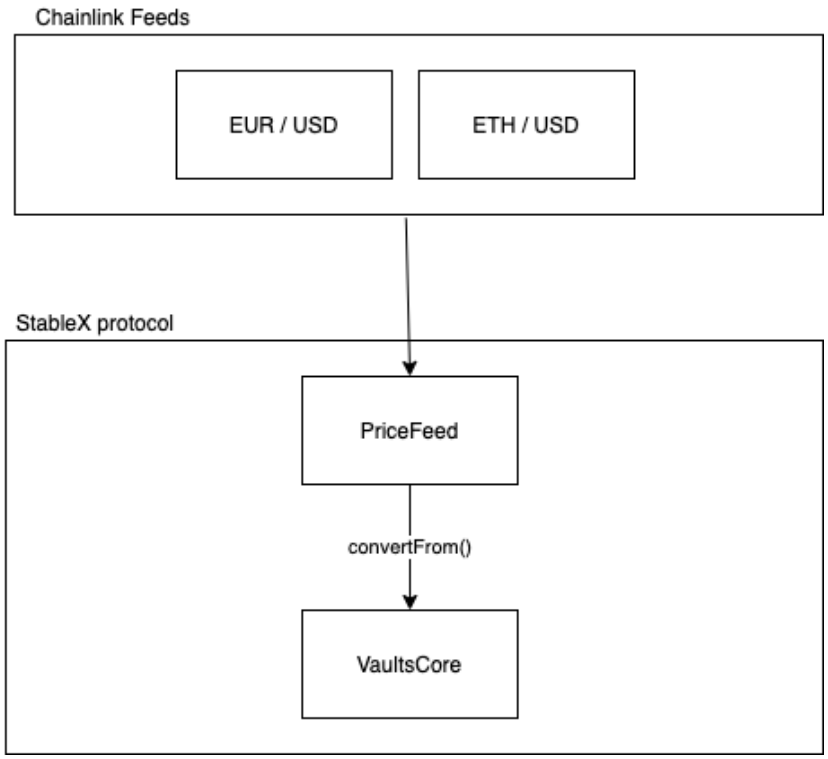

Figure 9: Price Feed

2.9 ConfigProvider

The Parallel protocol stores and reads system parameters from the **ConfigProvider** contract. Parameters that are set for each collateral type include:

- **minCollateralRatio**: The minimum MCR before a vault can be liquidated.
- **minOpenRatio**: The minimum MCR required for borrowing and withdrawing.
- **originationFee**: An optional origination fee for newly created debt. Can be 0.
- **liquidationBonus**: The liquidation bonus to be paid to liquidators.
- **liquidationFee**: An optional fee for liquidation debt. Can be 0.

These system parameters are designed to be adjusted through the Parallel protocol's decentralized governance process.

2.10 MIMO Token

MIMO governance tokens are distributed to users of the protocol for both creating PAR through borrowing and for providing liquidity in AMM pools. This token distribution guarantees that the protocol stewardship remains with the users and the community of the Parallel Protocol.

2.11 Liquidity Mining

MIMO distribution is managed by the **MIMODistributor** module, which mints MIMO tokens according to a supply curve that reduces the amount of new MIMO tokens by 5.55% every week.

2.11.1 Distribution parameters

In the 1st week, 55.5m MIMO will be issued. Then, the amount issued in subsequent week are calculated as:

$$WeeklyIssuance = 55.5m * 0.9445^{weeks} \tag{7}$$

This guarantees that there will never be more than 1,000,000,000 MIMO tokens.

MIMO is distributed across different liquidity pools, for example:

- SupplyMiner (WETH): 25%
- SupplyMiner (WBTC): 25%
- DemandMiner (PAR:WETH AMM pool): 50%

Liquidity mining rewards are used to incentivize users for creating PAR and supplying PAR liquidity.

2.11.2 Contract Architecture

The MIMODistributor follows the FeeDistributor model that is used for distributing fees in the main protocol. It maintains a list of smart contract "Payees" who receive a share of the newly minted MIMO tokens.

Two types of contracts exist to distribute MIMO: Supply Miners and Demand Miners.

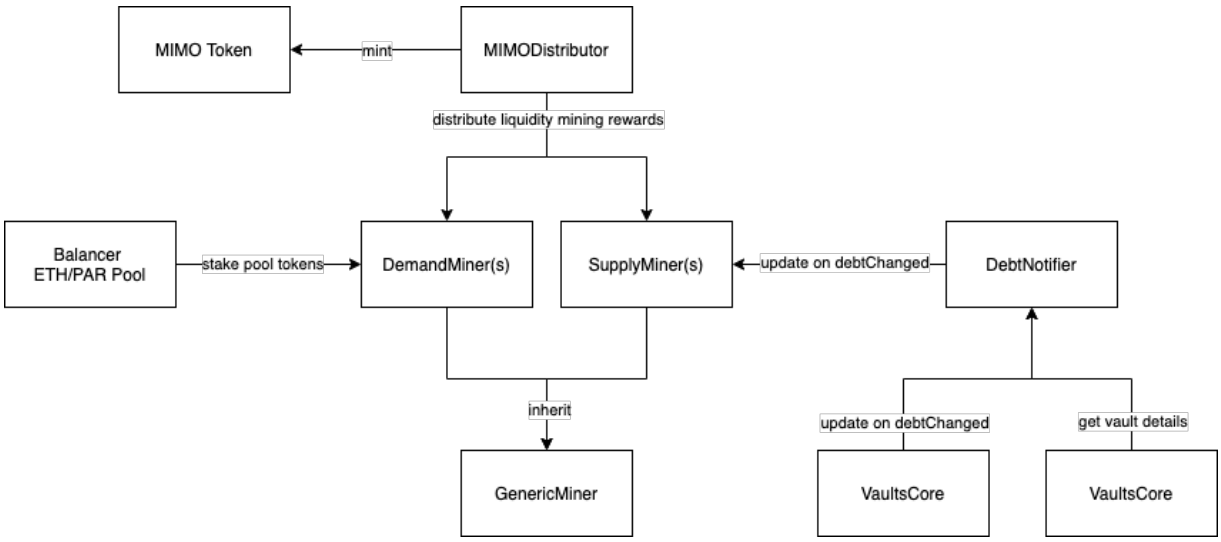

Figure 10: Liquidity Mining Architecture

- SupplyMiners are used to incentivize creating PAR by borrowing. Each 'SupplyMiner' is configured to incentivize borrowing against 1 specific collateral type (Eg: WETH).
- DemandMiners are built to incentivize providing PAR liquidity in AMM pools. Each DemandMiner manages the distribution of MIMO to 1 AMM pool via a pool token staking mechanism.
- Both the DemandMiner and the SupplyMiner use the same logic which they inherit from the GenericMiner contract. The difference between the two is the code for depositing withdrawing AMM pool tokens in the DemandMiner and the updating of staking power due to borrowing in the SupplyMiner.

The DebtNotifier contract mananages a mapping between collateral and the corresponding SupplyMiner.

2.11.3 Contract Summary

MIMODistributor Mints MIMO tokens. Owns the distribution formula (-5.55%/week).

SupplyMiner Incentivizes borrowing. BaseDebt is used to compare relative debt of all users. A different SupplyMiner is used for different collateralTypes, as baseDebt is not comparable across collateral types. Basedebt can be used as a standin for debt, because debt is calculated with the following formula:

$$debt = baseDebt * cumulativeRate \tag{8}$$

DemandMiner Incentivizes AMM pools through pool token staking. A different DemandMiner is deployed for each AMM pool that is intended to be incentivized.

GenericMiner Based on ERC-2917

DebtNotifier Gets notified by VaultsCore on borrow() and repay() that debt of a specific vault has changed. It manages a mapping for each collateral type to notify the correct SupplyMiner contract of the debt change.

GovernanceAddressProvider Manages all the addresses of the deployed contracts to find each other.

2.12 Governance

The Parallel Protocol governance system is a set of smart contracts that allows MIMO tokenholders to accrue voting power and vote on proposals that implement changes to the protocol in a decentralized manner.

2.12.1 Contract Architecture

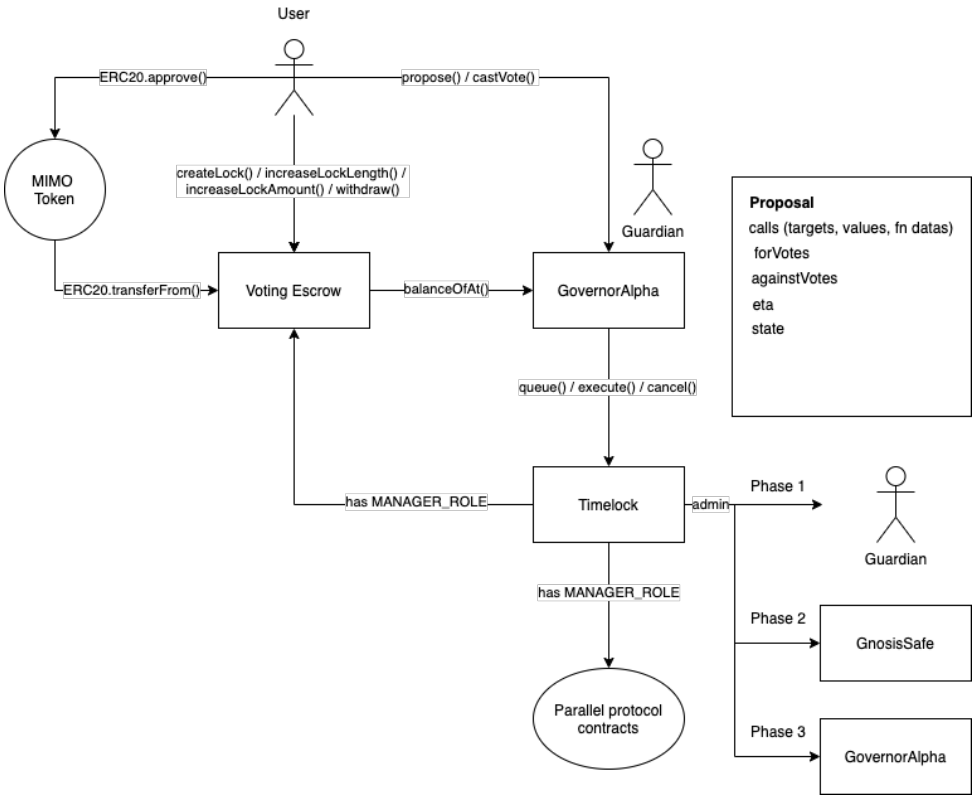

Figure 11: Governance architecture

The governance system consists of the following contracts:

- **VotingEscrow**: Turns staked governance tokens into voting power to align long-term incentives. Implements a balanceOfAt() function used in GovernorAlpha to determine voting power at a specific time.
- **GovernorAlpha**: Governance contract where proposals are created, voted, and executed. Works as the administrator of Timelock.
- **Timelock**: Each protocol contract is owned by a Timelock. The Timelock contract can modify system parameters, logic, and contracts in a time-delayed pattern.

2.12.2 Voting Power

In order to participate in Parallel Protocol governance, token holders must lock their MIMO in the VotingEscrow contract. Users receive non-transferable vMIMO in exchange, with 1 vMIMO token equivalent to locking 1 MIMO for 4 years (or 4 MIMO for 1 year). Users' voting power decays as their lock period comes closer to expiration, but can be extended at any time.

The voting power is calculated as follows:

$$votingWeight_i = stake_i * remainingLockup_i \tag{9}$$

The staking contract keeps track of remaining lockup in real-time without any user intervention. According to the formula, voting power is topped up at the moment of deposit, and starts decreasing linearly afterwards.

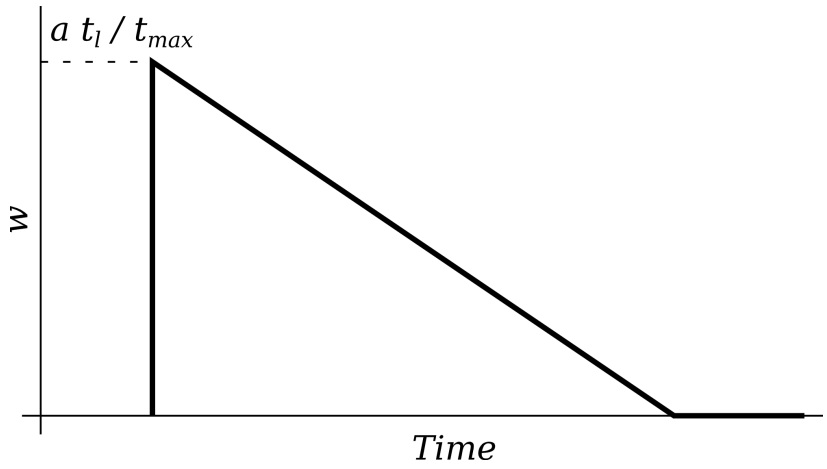

Figure 12: Voting power decreases linearly as time progresses, and vanishes completely at lockup expiration.

MIMO tokenholders can lock their MIMO tokens for a specified period of time with 'VotingEscrow.createLock()' and adjust their lock amounts / periods upwards:

```
interface IVotingEscrow {
  function createLock(uint256 _value, uint256 _unlockTime) external;
  function increaseLockAmount(uint256 _value) external;
  function increaseLockLength(uint256 _unlockTime) external;
  function withdraw() external;

  function balanceOf(address _owner) external view returns (uint256);
  function balanceOfAt(address _owner, uint256 _blockTime) external view returns (
      uint256);
}
```

You can increase your voting power by increasing the amount of tokens locked and increasing the length of the lock.

Note that locking tokens beyond the MAXTIME of 4 years will not earn additional voting power. Locking 100 tokens for 4 years now and locking 100 tokens for more than 4 years earns you the same amount of voting power.

2.12.3 Creating Proposals

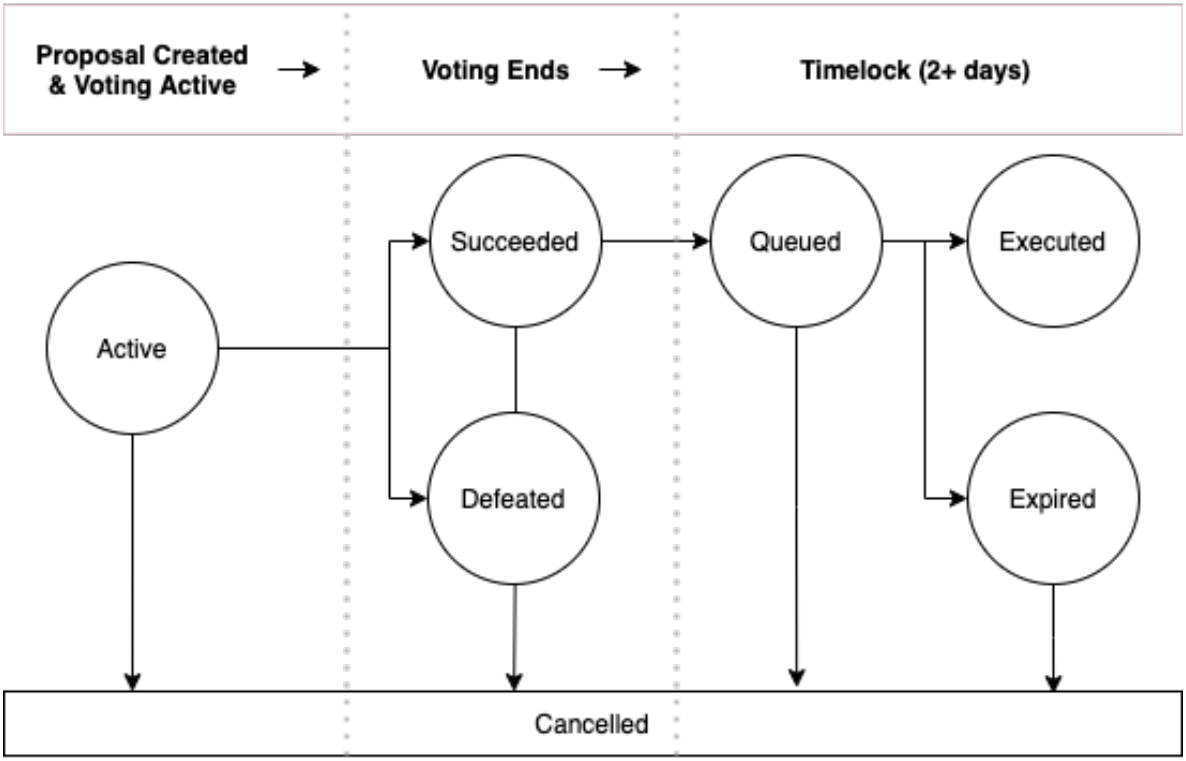

Figure 13: Voting and governance flow.

Creating a proposal requires the proposer to have 0.02% staked tokens staked at MAXTIME. Once a proposal has been submitted, voting is live for 3 days.

Proposals that gain majority support and meet the '1% of total staked MIMO tokens at MAXTIME quorum requirement are executed after a 2 day time-lock delay. These governance parameters are set within the GovernorAlpha contract as constants.

Creating a proposal with 'propose' requires a list of contract calls. Contract calls in a proposal can update system parameters and perform upgrades to the protocol.

```
interface IGovernorAlpha {
    function propose(
      address[] memory targets,
      uint256[] memory values,
      string[] memory signatures,
      bytes[] memory calldatas,
```

```
7         string memory description,
8         uint256 endTime
9     ) public returns (uint)
10 }
```

2.12.4 Voting on Proposals

The Parallel Protocol governance system uses an updated version of Compound's GovernorAlpha contract. A Timelock administered by GovernorAlpha has the access control permissions required to update protocol parameters and perform upgrades. Proposals are proposed, voted on, queued, and executed on-chain.

GovernorAlpha calls the VotingEscrow.balanceOfAt() function at a proposal's endTime to calculate the user's voting power for that proposal.

2.12.5 Queueing and Executing Proposals

After a proposal has successfully passed voting, any address can call the GovernorAlpha queue method to move the proposal into the Timelock queue. A proposal can only be queued if it has succeeded.

The Timelock has a minimum delay of X days, which is the least amount of notice possible for a governance action. Each proposed action will be published at a minimum of X days in the future from the time of announcement.

After the Timelock delay period, any account may invoke the execute method to apply the changes from the proposal to the target contracts. This will invoke each of the actions described in the proposal.

If a single call within the proposal fails, the whole batch will revert. Execution can be retried later. Alternatively, the proposal can also be cancelled by the guardian address. Proposals will expire automatically after a grace period of 14 days if they are not executed.

3 Summary

In this paper we've introduced the Parallel Protocol.

Following a mainnet launch, PAR will progressively decentralize to a community governance model. Holders of the PAR governance token will be able to vote on-chain on the ongoing operations and upgrades to the Parallel Protocol.

To get the latest project updates please visit our website at MIMO.capital.

4 Glossary

Automated Market Maker: Smart contract with a price-adjustment model in which an asset's spot price deterministically responds to market forces and market participants on either side of the market trade with the AMM rather than with each other.

Borrow rate: The interest rate paid over time by borrowers (e.g. 2%.) Each collateral has its own borrow rate.

Borrow fee: The amount of fees accrued in a vault. Calculated based on the cumulative borrow rate and the amount and length of the loan.

Cumulative rate: The time-adjusted cumulative borrow rate for a collateral. Accounts for rate changes.

ERC20: A technical standard used for smart contracts on the Ethereum blockchain for implementing tokens.

Health factor: A number that represents the collateralization level of a vault. Vaults with a health factor less than 1 is open for liquidation.

Liquidation fee: Fee charged to the borrower for getting his collateral liquidated.

Origination fee: Fee applied when borrowing / opening a loan.

Vault base debt: A number unique to each vault that partially represents a vault's debt.

Vault total debt: The amount that borrowers will have to fully repay before the vault's collateral can be fully withdrawn.

Stablecoins: Cryptocurrencies designed to minimize the effects of price volatility. They seek to function as a store of value and a unit of account.

References

[1] MakerDAO, *Dai*
https://makerdao.com/dai/