FUJI

Protocol Whitepaper

V1.0

August 2021

Abstract

Fuji is an infrastructure protocol that aggregates lending-borrowing crypto markets within blockchain networks. The protocol value-add proposition is to optimize interest rates to its users (both borrowers and lenders) by automating routing and movement of funds across lending-borrowing protocols and blockchain networks in search of the best APR. This document describes the definitions, architecture and theory behind the Fuji Protocol.

Author: Daigaro Cota
Co-authors: Boyan Barakov, Edgar Moreau
Reviewers:
Editors:

Contents

1. Introduction

1.1. Decentralized Finance

Blockchain technologies have been evolving since the inception of Bitcoin in 2009. Through the evolution of blockchain technologies and the emergence of turing-complete programming languages within blockchain (e.g Ethereum Virtual Machine), a series of applications have been developed that replicate the traditional finance industry's core business. These financial applications are capable of performing transactions of any size, almost instantaneously, through trustless and permissionless software code called *smart contracts*. These financial applications allow anybody with cryptocurrency to access basic and sophisticated financial tools in decentralized networks. The collection and network of all these financial applications within blockchain has been commonly referred to as *decentralized finance* or for short *DeFi*.

1.2. Lending-Borrowing Protocols

Lending-Borrowing protocols are a type of financial application within DeFi. They allow anybody with a crypto asset to lend and borrow the same or different crypto assets. The first lending-borrowing protocols for crypto assets functioned as peer-to-peer systems and facilitated collateralized and uncollateralized loans between market participants directly. However, these peer-to-peer lending protocols turned out limited as users had to individually manage the various term aspects of each loan, and to the eyes of critics, it was a tedious task for the average user to handle.

The *Compound Protocol* was deployed in 2019 and it introduced an algorithm that automatically sets floating interest rates for lenders and borrowers based on supply and demand. Compound Protocol's algorithm allows users to earn interest by supplying their crypto assets and borrow them "without having to negotiate terms such as maturity, interest rate, or collateral with a peer or counterparty"[1].

[1] Compound: The Money Market Protocol. https://compound.finance/documents/Compound.Whitepaper.pdf

The introduction of the Compound Protocol, and its simplified method of defining interest rates, propelled greater adoption among users in the lending-borrowing space.

1.3. Aggregating Lending Borrowing Protocols

After Compound’s introduction, many similar lending-borrowing protocols followed suit. Among them is *Aave Protocol*, which evolved rebranded from a peer-to-peer lending system previously known as *EthLend*. At the time of writing Compound and Aave represent the two biggest lending-borrowing markets with over 25-billion-dollar worth in Total Value Locked (*TVL*) in crypto assets. However, more than half a dozen lending-borrowing protocols exist with large enough pools. All these protocols handle a portfolio of similar crypto assets, and their interest rate models differ slightly in set-up parameters but similarly adapt to supply and demand changes.

The existence of various lending-borrowing protocols for similar crypto assets within the same or different blockchains and the constant fluctuation of interest rates due to supply and demand creates arbitrage opportunities. On the other side, users would be required to constantly monitor the markets and to move funds from one protocol to another if they want to take advantage of the difference in interest rates. Such a task could be seen as inconvenient and costly for most of them, considering the time and network transaction fees involved. This creates favorable conditions for development of optimization strategies for lend yields and borrowing cost.

Aiming to facilitate this to DeFi users, the Fuji Protocol was designed to aggregate the lending-borrowing markets, with an initial focus on minimizing the cost of borrowing. The protocol adds value to the user by creating a system that monitors interest rates, routes transactions to the lowest rate and rebalances debt across the various protocols in or out different blockchains in constant search of a lower interest rate. A similar mechanism that may be implemented in the future can be used to maximize lending rates by rebalancing liquidity across the lending-borrowing markets.

2. Fuji Protocol

2.1. Fuji Protocol Core System Diagram

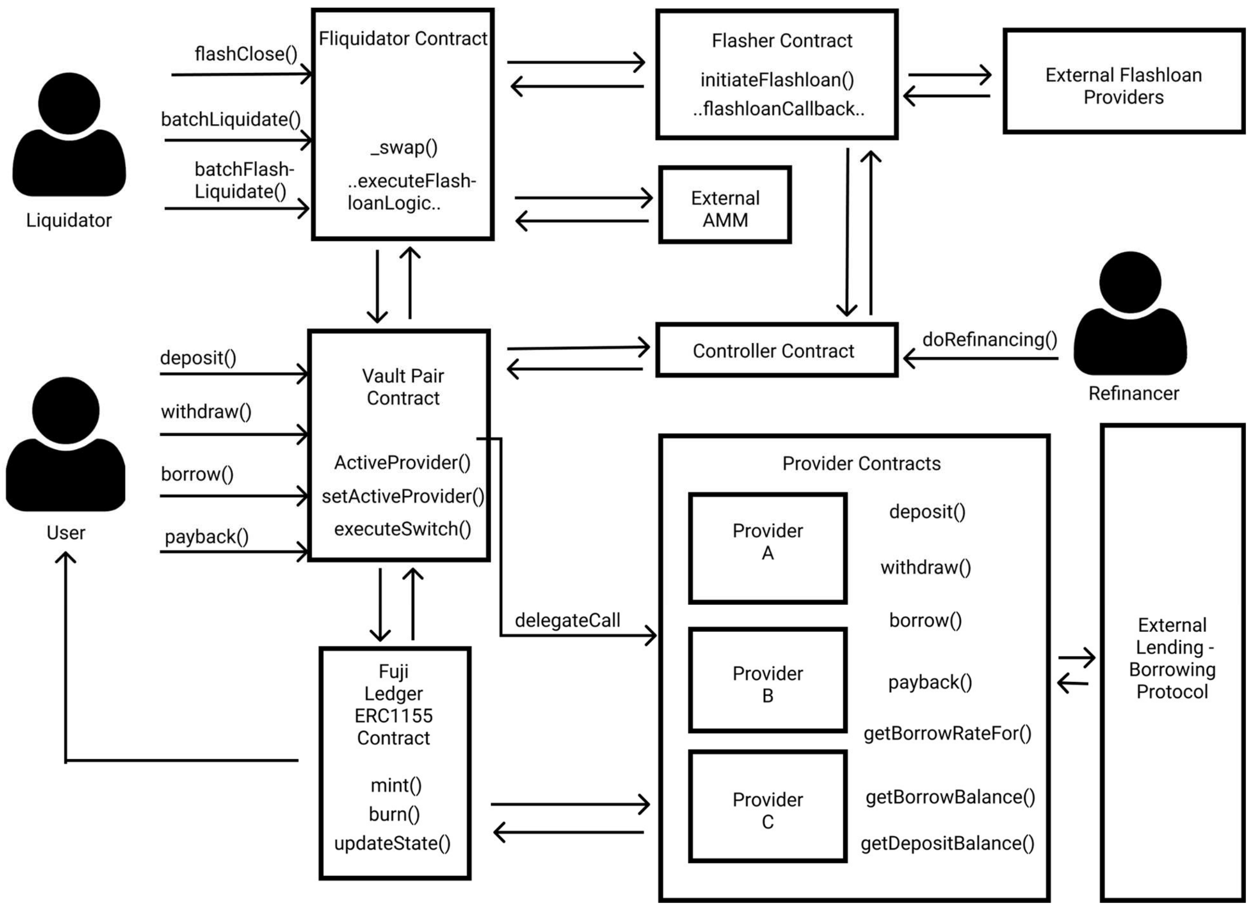

Figure 1 - Fuji's Core System Diagram

2.2. Provider Contract

Fuji defines a *Provider* as any protocol with a lending-borrowing market that implements the following main 4 functions: deposit, withdraw, borrow and payback.

Since the required arguments for each of these functions differs across lending-borrowing protocols, it is necessary to define a common interface for all. Fuji creates a *Provider* smart contract to wrap the unique logic and arguments for each lending-borrowing protocol into a common one. *Provider* smart contracts are stateless contracts. They are delegate-called by the vault-pair contracts. The concept of a common interface is inspired from Instadapp[2], where these interfaces are called Connectors.

2.3. Vault Contract

The *Vault* contract is the center of user interaction and holder of the collateral and debt position at the underlying lending-borrowing protocols. Users can only: deposit, withdraw, borrow, and payback through the vault contract. Each Vault contract is designed to handle specific collateral and borrow asset types, also called pairs. Thus, isolated pairs allow for better management of risk while aggregating the lending-borrowing markets. A key parameter for aggregation of the same pair across different lending-borrowing protocols is the collateralization ratio in each of them. They need to be similar or close to one another. Second to the collateralization ratio is the size of the liquidity pool.

PAIR-TYPES		COLLATERAL-IZATION RATIO	Market Liquidity	AGGREGATED LENDING-BORROWING PROTOCOL			
COLLATERAL	BORROW			PROTOCOL A	PROTOCOL B	PROTOCOL C	....
ETH	DAI	>75%	>$500M	X		X	
ETH	DAI	>66%	>$10M	X	X	X	
WBTC	ETH	>75%	>$100M	X	X	X	
USDC	ETH	>90%	>$1M		X		
....	....						....

Table 1 - Pair-type markets Aggregation Example

The Vault contract contains a pointer for the Provider contract address offering the best rate. At the moment of user interaction all transactions are routed through the active provider. The active provider is set/changed at the moment of a rebalancing event, or directly called by an authorized account (during the setup for example) when it is strategic according to details described in Section 3.1.

The bookkeeping of every user's deposit and borrowed amounts, within a *Vault* contract, is tracked via interest accruing tokens in the FujiERC1155 contract (for

[2] Instadapp: Middleware that aggregates multiple DeFi protocols into one upgradable smart contract layer. https://docs.instadapp.io/

short called “F1155” hereafter and further explained in Section 2.4). Each vault-pair is assigned a specific token Id for the collateral and borrow assets in the F1155 contract. By design the F1155 tokens are non-transferable, although transferability could be enacted for future applications. The vault checks the collateralization ratio of each user at the time of any withdraw and borrow operation using the F1155 tokens. This ensures that no user falls undercollateralized per factor ratio defined in each Vault contract. This factor ratio in each *Vault* allows the protocol to define unique collateralization requirements for every pair crypto market.

2.4. FujiERC1155 or “F1155” Contract

The *F1155* contract is the bookkeeping ledger for all vaults in the Fuji protocol. It follows the ERC1155 token standard and it is used to keep track of all deposits and borrow positions opened by all users. Mint and Burn functionality are restricted to the Vault and Fliquidator (explained in Section 2.7) contracts only.

The *F1155* contract keeps track of the interest accumulated by users using an *index* for each token Id. The *index* concept and its mechanics were introduced in Aave V2.0 Whitepaper[3] and are referred as *interest accruing tokens*. Within Fuji’s context, the index for each token Id, represents the accumulated interest (either yield or borrowing) since the moment of deployment for a specific vault. All token Ids have an initial index value equal to 1. The index value, at any time, for a specific token Id can be computed as:

Equation 1: Index Computation

$$i_{t_x} = i_{t_{x-1}} + \left(B_{t_{x-1}} - B_{t_x}\right)/B_{t_{x-1}}$$

$$Where:$$

$$i_{t_x} = index\ value\ at\ time\ x$$

$$B_{t_x} = Vault\ balance\ at\ time\ x$$

As it can be noted from Equation *1*, the vaults require to be bootstrapped with a reserve liquidity to avoid previous balance $B_{t_{x-1}}$ to equal zero. Index values must be computed before execution of any withdraw, borrow and payback operations

[3] Aave: Protocol Whitepaper V2.0, https://github.com/aave/protocol-v2/blob/master/aave-v2-whitepaper.pdf

at the base lending-borrowing protocols. This avoids creating a negative delta parameter $\left(B_{t_{x-1}} - B_{t_x}\right)$, which would result in reducing the index.

The balance of any user and its accrued interest at any time, for a specific token Id, can be computed as:

Equation 2: User Balance Computation

$$b_{t_x} = i_{t_x} \times Sb_{t_0}$$

$$Where:$$

$$b_{t_x} = user's\ balance\ at\ time\ x$$

$$i_{t_x} = index\ value\ at\ time\ x$$

$$Sb_{t_0} = user's\ scaled\ balance$$

The scaled balance is an abstract number that facilitates the computation of principal + accumulated interest for every user in the pool. It is a stored value for each user that interacts with the vaults, for each token Id, for a zero-balance-user, in the *F1155* contract. Scaled balance is computed as follows:

Equation 3 - User Scale Balance Computation

$$Sb_{t_0} = \$A/i_{t_x}$$

$$Where:$$

$$Sb_{t_0} = user's\ scaled\ balance$$

$$\$A = transaction\ amount$$

$$i_{t_x} = index\ value\ at\ time\ x$$

For the initial release of the Fuji protocol the transfer functionality of ERC1155 tokens was deactivated. The obvious reasoning behind this decision was to avoid the transfer of debt from a user address to another. However, the possibility to allow guarded transfer could be enabled in the future.

2.5. Flasher Contract

The *Flasher* contract facilitates the transfer of funds from a lending-borrowing protocol to another. “A flash loan is a smart contract transaction in which a lender smart contract lends assets to a borrower smart contract with the condition that the assets are returned, plus an optional fee, before the end of the transaction"[4]. Flash loans allow for efficient capital transfers that are not possible and have no equivalent in traditional finance. They can be described as a transaction in which someone will borrow as much as a bank reserve has, perform operations with the funds and return the amount back plus a fee, all done within a short block of time (the average block time in the Ethereum Mainnet is ~13 seconds for instance). If the return of the funds plus fee is not guaranteed before the end of this short time lapse all transactions are reverted as if they never happened.

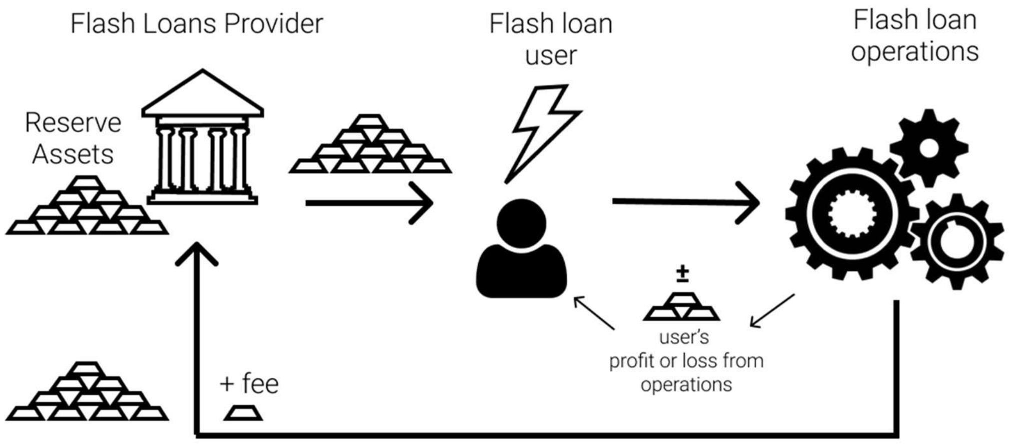

Figure 2 - Simple Flash loan Transaction Flow

Flash loans are the main tool for rebalancing, and liquidation operations within the Fuji protocol. These key operations are further described in Section 3. Execution of the flash loan calls within the Flasher Contract is restricted to the *Controller* and *Fliquidator* contract addresses.

2.6. Controller Contract

The *Controller* contract initiates the execution of rebalancing operations. In Fuji's context, a refinancing transaction is that which moves debt and collateral from a lending-borrowing protocol to another, while a rebalance is simply the transfer of assets. The function calls for refinancing or rebalancing are capable of taking arguments to decide a partial or full transfer of funds of a specific vault from a lending-

[4] Extracted from EIP-3156: Flash Loans, https://eips.ethereum.org/EIPS/eip-3156

borrowing protocol to another. These function calls are restricted to a list of executors that belong to the decentralized network of Fuji.

Technical description of the refinancing operation job is described in Section 3.1

2.7. Fliquidator Contract

The *Fliquidator* contract facilitates the liquidation of users across the vaults. Liquidation, in the context of Fuji, is the process of selling a portion of a user's collateral to payback their undercollateralized debt. This occurs when the value of their collateral asset falls below a predefined threshold. Each vault defines this threshold value depending on the external requirements set by the aggregated lending-borrowing protocols. Due to the price volatility of crypto assets, liquidations play an important maintenance job for the protection of the pooled funds in vault contract. Users taking a debt position through Fuji are free to decide their risk exposure to price volatility, by deciding the amounts of collateral and debt they overtake. This free choice implies putting at risk other users' funds within the vault. To address this issue, Fuji performs internal liquidations, with requirements slightly above the underlying lending-borrowing protocols. The liquidation job is critical to avoid losing funds from users with healthy loan-to-value ratios.

Due to the usage of the ERC1155 standard for bookkeeping of user's positions, it is possible to perform batch liquidations: liquidating many positions in a single transaction and thus, saving on transactional costs. The *Fliquidator* contract allows two types of batch liquidation; one requires the liquidator to have the underlying debt asset, and the other uses a flash loan to borrow it. Liquidation functions are restricted to a list of executors that belong to the decentralized network of Fuji. A penalty fee is applied to liquidated users to deter very risky behavior when taking debt positions.

Technical description of the liquidation operation job is described in Section 3.2

3. Key Operations

3.1. Refinancing

Refinancing opportunities surge when the users demand to lend and borrow for pairs in the market changes across the lending-borrowing protocols. The theoretical rationale to perform a refinancing operation has three components: refinancing cost, refinancing opportunity cost, and amount to be refinanced. We first look into the refinancing cost.

Refinancing is a function of the transaction cost and the flash loan percentage fee. The transaction cost is paid to the blockchain network. All transactions in the blockchain are computational operations that store/change memory state. The fee is proportional to the size of the computation/storage inputs of the transaction and is typically paid in the native crypto asset of the blockchain network. Refinancing cost can then be expressed as shown in *Equation 4*. This equation utilizes the term *gas* which is particular to the Ethereum network. Gas represents computational units, and it has a free-market price valued on fractions of ether (Ethereum's native cryptocurrency, ETH). All other turing-complete blockchains operate similarly, and therefore, *Equation 4* still applies. On other blockchain only the native cryptocurrency price is replaced.

Equation 4 - Cost of Refinancing

$$R_{cost} = \sum \quad T_{x_{gas}} \cdot G_{price} \cdot ETH_{price} + B_{debt} + FL_{fee}$$

$$Where:$$

$R_{cost} = Refinancing\ cost$

$\sum \quad T_{x_{gas}} = sum\ of\ all\ the\ transactional\ gas, see\ \text{Table 2}.$

$G_{price} = gas\ price, typ.\ expressed\ in\ gwei\ (1e^{-9}\ ETH)$

$ETH_{price} = open\ market\ price\ for\ ETH$

$B_{debt} = amount\ of\ debt\ being\ refinanced$

$FL_{fee} = flash\ loan\ fee$

Flash loan fee could either be free (zero), a fixed small cost, or a small fraction of B_{debt}, depending on the flash loan provider. At the moment of writing: Cream Finance flash loan fee is 0.03% of B_{debt}, while Aave offers them at 0.09%. There is a tendency for flash loan cost to be lower, as there is no incentive to use them as arbitrage profit margins get tighter.

Transaction Type	**Avg Gas Units per Tx**[5]
Deposit	*~250,000*
Withdraw	*~300,000*
Borrow	*~450,000*
Payback	*~400,000*
Trade	*~225,000*

Table 2 - Avg. Gas Amounts for Transactions in Lending-Borrowing Protocols of Ethereum

Moving into the second component of the theoretical rationale behind refinancing, we explore the refinancing opportunity cost. This concept can be described as the interest accumulated someone could be saving for not switching to a platform with a better rate in a given timeframe. Since popular lending-borrowing protocols execute at least one operation per block, the compounding of interest gets updated approximately in continuous fashion. Therefore, the refinancing opportunity cost can be described with the following equation:

Equation 5 - Refinancing Opportunity Cost

$$R_{oc} = B_{debt} \cdot (e^{\Delta r \cdot t} - 1)$$

$$Where:$$

$$R_{oc} = refinancing\ opportunity\ cost$$

$$B_{debt} = amount\ of\ debt\ being\ refinanced$$

$$\Delta r = difference\ in\ rates, where\ 0 < \Delta r < 1$$

$$t = time\ elapsed\ since\ last\ refinance\ operation$$

[5] Gas units per transaction vary across Ethereum's lending-borrowing protocols.

Graph shown in *Figure 3*, illustrates the variance in APRs between a pair-type of two lending-borrowing protocols. A market event triggers the difference which results in the opportunity to refinance debt from provider A to provider B between times t_0 and t_1 to save cost on interest rates.

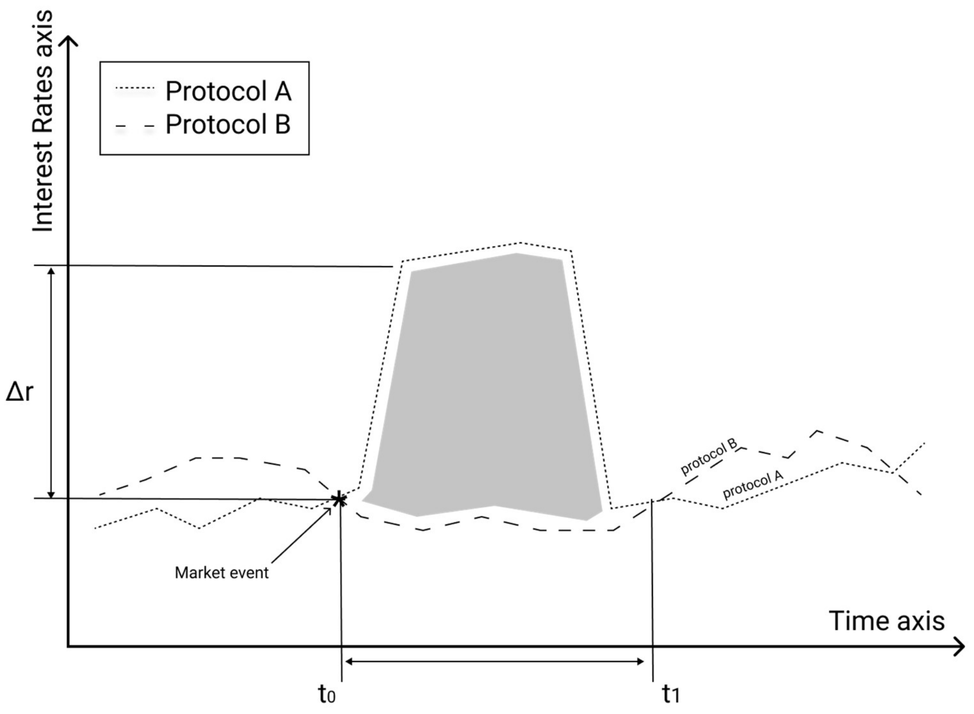

Figure 3- Refinancing Opportunity Cost Plot

Since t_1 is unknown during present market conditions, algorithms can be developed to predict what is the expected length time, and what should be the minimum interest rate difference to consider refinancing. Overall, a refinancing operation makes economic sense only when time scale and rate difference are significant. On the other hand, as an economy of scale builds up in a vault, users will benefit from the collected pool funds, such that the amount of debt is significantly large and outweighs the other factors in *Equation 5*. This will justify more frequent refinancing, and drive further cost optimization.

Lastly, we discuss the third component of the rationale behind refinancing operations: amount to refinance. As mentioned in the last point, Fuji debt undertakers benefit from an economy of scale. However, as vault-pairs become largely liquid, the movement of significant funds across lending-borrowing platforms will create slippage in interest rates. For this reason, it is critical to understand the mechanics of the interest rate model described by the following sets of equations:

Equation 6 - Borrow and Supply Interest Rates (Interest Rate Model)

$$U_r = \frac{Borrows_a}{Supply_a}$$

$$r_b = MU_r + Base_{rate}$$

$$r_s = U_r \cdot r_b$$

$$Where:$$

$U_r = utilization\ ratio$

$Borrows_a = \$\ amount\ of\ assets\ borrowed, where\ 0 < Borrows_a < Supply_a$

$Supply_a = \$amount\ of\ assets\ supplied, where\ Supply_a > 0$

$r_b = borrowing\ rate$

$$M = Change\ slope\ for\ borrow\ rate, where\ M > 0$$

$$Base_{rate} = Initial\ r_b\ when\ U_r = 0$$

$r_s = supply\ earning\ rate$

The equations above are shown in *Figure 4*; values were substituted to generate the plot: M= 0.4, U_r= 0.54, $Base_{rate}$= 0.025.

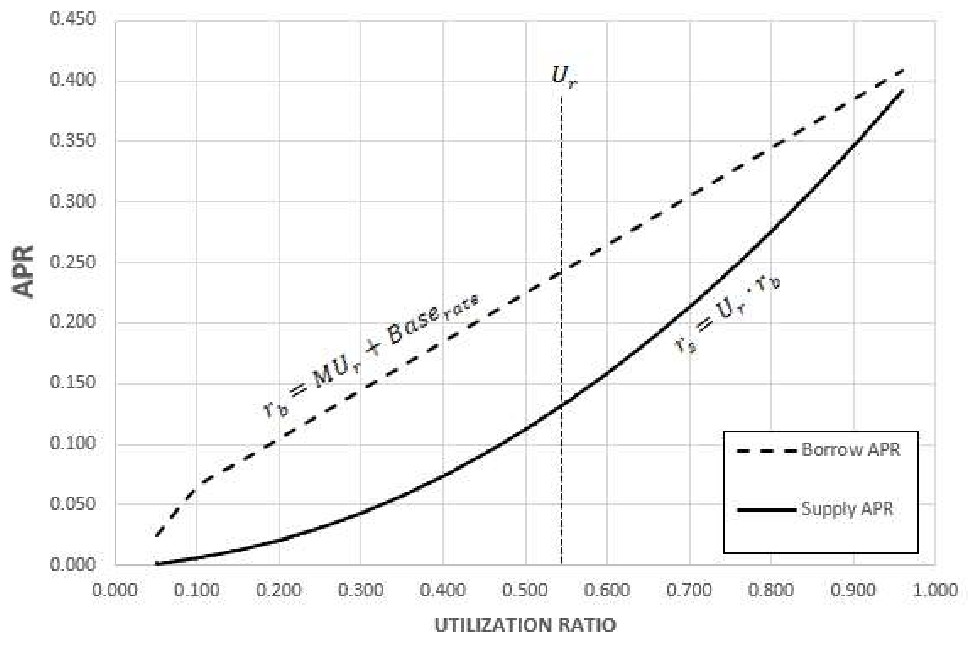

Figure 4- Interest Rate Model Graph Example

Slippage can be defined as the change in interest rate induced by the change in either supplied or borrowed amount in the money market. For purposes of a refinance operation, we can assume that the supplied amount of crypto asset in the pair-type money market remains constant. Then, the slippage or change in interest rate can be described as:

Equation 7 - Slippage Equation for Interest Rate Model

$$\Delta r_b = S_{\%} = M \cdot \Delta U_r$$

$$\Delta U_r = \frac{\delta Borrows_a}{Supply_a}$$

$$Where:$$

$$S_{\%} = slippage\ percentage$$

$$\Delta U_r = change\ in\ utilization\ ratio$$

$$\delta Borrows_a = amount\ of\ debt\ to\ be\ refinanced$$

Since the basic interest rate model for the borrow rate is a linear equation, slippage is directly proportional to the ratio of debt being refinanced. The slope M value then plays a crucial role in defining the amount that can be transferred from a protocol to another for a pair-type money market. This must be considered in the context of all other lending-borrowing platforms. Figure 5 provides a visualization to estimate the refinancing amount given some market conditions. The limit that can be refinanced on a specific lending borrowing protocol can be estimated by using

Equation 7, along with the current market difference in rates. Is also important to keep in mind that the interest rate model can also vary as utilization rate changes.

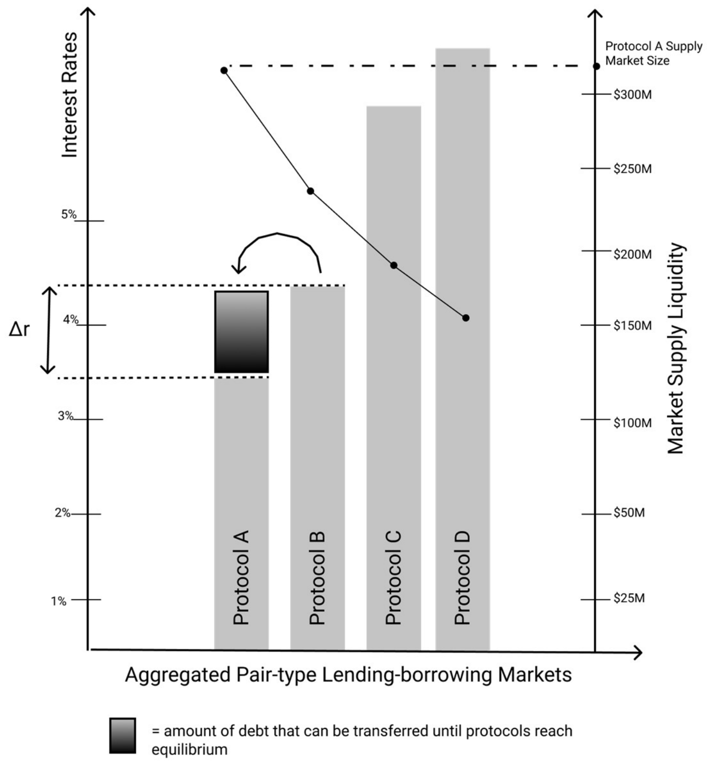

Figure 5 -Estimating Transfer Amount Limit

Refinancing operations are crucial for the value-add proposition of Fuji. The task of collecting the information and executing the refinancing jobs, should be performed by a network of users who have a vested interest in the protocol. This point is further discussed in Section 4.

3.2. Liquidations

The permissionless adoption of lending and borrowing in DeFi has been traditionally overcollateralized[6]. That means borrowers put higher value as collateral in crypto assets than they take in the form of a loan. This mechanism incentivizes the borrower to pay back without the lender needing to know the personal, or legal information of the borrower. However, the volatile nature of crypto assets leads to situations in which the value of the collateral becomes close to the value of the loan. At that moment, a liquidation event must take place. Liquidation is the operation of selling a portion or the entirety of a user's collateral to pay back an open debt

[6] At the time of writing undercollateralized loans are still experimental but they will become increasingly more important.

position. The ratio of loan debt amount to the collateral is known as *Loan-to-value (LTV)*. The ratio must be computed using a common currency unit. The inverse of such a ratio is known as the *collateralization ratio*. Since the different lending-borrowing protocols use both ratios, both are introduced for Fuji's context.

Equation 8- Loan-to-Value (LTV) and Collateralization Ratio

$$LTV = \frac{Borrow_{\$}}{Collateral_{\$}}$$

$$Collateralization\ Ratio = \frac{Collateral_{\$}}{Borrow_{\$}}$$

A *Vault* contract represents the collective position of all users in a particular pair. Therefore, it is important that risky users in the vault do not compromise the funds of lenders and healthy borrowers. For this reason, the Fuji protocol makes liquidations an internal mechanism.

As indicated in Section 2.4, the Fuji protocol has an internal accounting system in the F1155 contract. This accounting facilitates the job of monitoring the individual users positions with a recurrent periodicity, and also helps identify if any user is close to liquidation.

The vault contracts independently define the threshold the LTV at which a liquidation should occur. Liquidators are Fuji users that sign up to monitor position and to constantly check for liquidations. The service provided by Liquidators must be economically incentivized in order to maintain the protocol safe during price volatility.

4. Decentralized Governance

Fuji protocol consists of decentralized network of users that facilitate aggregation and optimization of interest rates in lending- borrowing crypto asset markets across blockchains in a peer-to-peer model. The approach to decentralization of the system requires governance decisions to be made, and to maintain the key operations described in Section 3. For this reason, the Fuji protocol plans to introduce a token.

4.1. Fuji Token

The Fuji token is the tool that allows users of the Fuji protocol to participate in refinancing and liquidations jobs. It is also used to partake in the governance decisions of the protocol. The Fuji token design follows the ERC20 standard as defined in the Ethereum blockchain documentation. The Fuji token has no intrinsic value and it is distributed to the users of the protocol. As users benefit from the protocol's optimization of lending-borrowing rates, they also become Fuji holders. This promotes their interest to ensure the maintenance of the key operations tasks and future governance decisions.

4.2. Governance Role

The changing conditions in DeFi, and constant evolution of blockchain technology do not guarantee the architectural design of Fuji today will be able to address the needs for lending-borrowing aggregators of the future. For this reason, the protocol is designed to allow smart contract updates through the decentralized governance body.

The governance body will subject all changes to vote, which will be agreed and verified in the blockchain. All critical changes to the protocol are subject to time-lock delays. The following list includes but is not limited to the decision power of the governance body:

- Incentives to Refinancing job operators
- Incentives to Liquidation job operators
- Deploying new vaults for new crypto assets.
- Changing LTV thresholds in the vaults given market conditions.
- Deploying refinancing strategies.

- Adjusting refinancing\rebalancing amounts.
- Adding a lending-borrowing provider to an existing vault-pair type.

5. Conclusion

The Fuji protocol is a proposal for an infrastructure project that aggregates lending-borrowing crypto asset markets. This infrastructure facilitates users to benefit from the best APRs in the lending and borrowing markets. The complexity and optionality of blockchains and lending-borrowing protocols will make Fuji the place to go for new users seeking to engage in lending and borrowing with crypto assets. The architecture design of the protocol is explained along with the theoretical rationale behind some of the design features. A decentralized network of users is built around the project to ensure distributed maintenance of the key operational tasks that make the infrastructure functional. In addition, the Fuji protocol is not meant to have a single entity in control of the funds involved in the aggregation of the lending-borrowing pairs.