Compound:
The Money Market Protocol

Version 0.2
February 2018

Authors
Robert Leshner, Geoffrey Hayes
https://compound.finance

Abstract
In this paper we introduce a decentralized protocol which establishes money markets with algorithmically set interest rates based on supply and demand, allowing users to frictionlessly borrow Ethereum assets.

Contents

1 Introduction

The market for cryptocurrencies and digital blockchain assets has developed into a vibrant ecosystem of investors, speculators, and traders, exchanging thousands [1] of blockchain assets. Unfortunately, the sophistication of financial markets hasn't followed: participants have little capability of trading the *time value* of assets.

Interest rates fill the gap between people with surplus assets they can't use, and people without assets (that have a productive or investment use); trading the time value of assets benefits both parties, and creates non-zero-sum wealth. For blockchain assets, two major flaws exist today:

- Borrowing mechanisms are extremely limited, which contributes to mispriced assets (e.g. "scamcoins" with unfathomable valuations, because there's no way to short them).
- Blockchain assets have negative yield, resulting from significant storage costs and risks (both on-exchange and off-exchange), without natural interest rates to offset those costs. This contributes to volatility, as holding is disincentivized.

Centralized exchanges (including Bitfinex, Poloniex...) allow customers to trade blockchain assets on margin, with "borrowing markets" built into the exchange. These are trust-based systems (you have to trust that the exchange won't get hacked, abscond with your assets, or incorrectly close out your position), are limited to certain customer groups, and limited to a small number of (the most mainstream) assets. Finally, balances and positions are virtual; you can't move a position on-chain, for example to use borrowed Ether or tokens in a smart contract or ICO, making these facilities inaccessible to dApps [2].

Peer to peer protocols (including ETHLend, Ripio, Lendroid, dYdX...) facilitate collateralized and uncollateralized loans between market participants directly. Unfortunately, decentralization forces significant costs and frictions onto users; in every protocol reviewed, lenders are required to post, manage, and (in the event of collateralized loans) supervise loan offers and active loans, and loan fulfillment is often slow & asynchronous (loans have to be funded, which takes time) [3-6].

In this paper, we introduce a decentralized system for the frictionless borrowing of Ethereum tokens without the flaws of existing approaches, enabling proper money markets to function, and creating a safe positive-yield approach to storing assets.

2 The Compound Protocol

Compound is a protocol on the Ethereum blockchain that enables the borrowing and lending of Ethereum blockchain assets by creating ***money markets***, which are collections of tokens made available to borrowers. Each money market has a transparent and publicly-inspectable balance sheet, which is used to algorithmically determine a floating interest rate based on the supply and demand for the asset, allowing suppliers of capital and borrowers to exchange the time value of an asset without having to negotiate terms (including maturity, interest rates, or collateral).

Each money market generates an economic profit, resulting from a spread between supply and borrowing interest rates, which is used to secure the protocol.

2.1 Supplying Tokens

When a user sends a token to a money market, it becomes a fungible resource; unlike an exchange or peer-to-peer platform, the user's tokens are not matched to a specific loan; instead, the user owns a share of the collective tokens, which fund *all* loans. This allows users to have significantly more liquidity than alternate funding models; they can withdraw their tokens, at any time, without waiting for a specific loan to mature.

Balances in a money market accrue interest based on the supply interest rate unique to the asset. Users can view their balance (including accrued interest) in real-time; when the user makes a transaction that updates his or her balance (supplying, transferring, or withdrawing a token), accrued interest is converted into principal and paid to the user.

2.1.1 Primary Use Cases

Individuals with long-term investments in Ether and tokens ("HODLers") can use a Compound money market as a source of additional returns on their investment. For example, a user that owns OmiseGo can supply their tokens to the Compound protocol, and earn interest (denominated in OmiseGo) without having to manage their asset, fulfill loan requests or take speculative risks.

dApps, machines, and exchanges with token balances can use the Compound protocol as a source of monetization and incremental returns by "sweeping" balances; this has the potential to unlock entirely new business models for the Ethereum ecosystem.

2.2 Borrowing Tokens

Compound allows users to frictionlessly borrow from the protocol, using collateralized lines of credit, and withdraw borrowed tokens outside of the protocol to a wallet, exchange, or another address. Unlike peer-to-peer protocols, borrowing from Compound simply requires a user to specify a desired asset; there are no terms to negotiate, maturity dates, or funding periods; borrowing is instant and predictable. Similar to supplying an asset, each money market has a floating interest rate, set by market forces, which determines the borrowing cost for each asset.

We enforce a rule that each account must have a balance that more than covers the outstanding borrowed amount; we call this a ***collateral ratio***, and an account can take no action (e.g. borrow or withdraw) that would bring its value below our desired ratio; to increase (or reset) the collateral ratio, users are free to repay a borrowed asset in whole or in part, at any time. Balances held in Compound, even while being used as collateral, continue to accrue interest normally.

2.2.1 Risk & Liquidation

If the value of a borrower's collateral declines below a ***liquidation ratio*** (which is smaller than the collateral ratio used to initially borrow an asset) the collateral becomes available to purchase (with the borrowed asset) at the current market price minus a ***liquidation discount***; this incentivizes an ecosystem of arbitrageurs to quickly step in to reduce the borrower's exposure, and eliminate the protocol's risk.

Any Compound user who has supplied the borrowed asset may invoke the liquidation function, in whole or in part, exchanging their asset for the borrower's collateral. As both users, both assets, and prices are all contained within the Compound protocol, liquidation is frictionless and does not rely on any outside systems or markets.

2.2.2 Primary Use Cases

The ability to seamlessly hold new assets (without selling or rearranging a portfolio) gives new superpowers to dApp consumers, traders and developers:

- Without having to wait for an order to fill, or requiring off-chain behavior, dApps can borrow tokens to use in the Ethereum ecosystem, such as to purchase computing power on the Golem network
- Traders can finance new ICO investments by borrowing Ether, using their existing portfolio as collateral
- Traders looking to short a token can borrow it, send it to an exchange and sell the token, profiting from declines in overvalued tokens

2.3 Ledger System

Compound maintains a complete and auditable balance sheet and ledger, comprised of all transactions; for each money market ***a***:

$$Cash_a + Borrows_a = Supply_a + Equity_a$$

Each transaction generates two accounting entries (a debit and credit) using international accounting standards:

Event	Debit	Credit
Supply token	*Cash*	*Supply*
Withdraw token	*Supply*	*Cash*
Borrow token	*Borrows*	*Supply*

Repay borrowed token (customer)	*Supply*	*Borrows*
Repay borrowed token (default event)	$Supply_{Collateral}$	$Borrows_{Asset}$
Accrue Interest (Supply)	*Equity*	*Supply*
Accrue Interest (Loan)	*Borrows*	*Equity*

2.4 Interest Rate Model

Rather than individual suppliers or borrowers having to negotiate over terms and rates, the Compound protocol utilizes an interest rate model that achieves an efficient interest rate equilibrium, in each money market, based on the supply and demand of individual assets. The utilization ratio ***U*** for each money market ***a*** unifies supply and demand into a single variable:

$$U_a = Borrows_a \, / \, Supply_a$$

Following economic theory, interest rates (the "price" of money) should increase as a function of demand; when demand is low, interest rates should be low, and vise versa when demand is high. The demand curve is codified through governance (and can be updated by the ***Chief Economist***) and is expressed as a function of utilization. As an example, borrowing interest rates may resemble the following:

$$Borrowing\ Interest\ Rate_a \ = \ 10\% + U_a * 30\%$$

For the protocol to be sustainable, and to withstand economic attacks (by both supplying and borrowing in the same money market), the total amount of interest earned by suppliers must be less than the total interest product by borrowers. The supply interest rate is a function of the borrowing interest rate, and includes a spread, ***S*** (such as 0.15), which represents the economic profit of the protocol:

$$Supply\ Interest\ Rate_a \ = Borrowing\ Interest\ Rate_a * U_a * (1 - S)$$

3 Architecture

At its core, a Compound money market is a ledger that allows Ethereum accounts to supply or borrow tokens, while computing interest, a function of time. The protocol's smart contracts will be publicly accessible and completely free to use for machines, dApps and humans.

3.1 MoneyMarket Contract

A *MoneyMarket* contract stores each accounts current balance. In its simplest form, the MoneyMarket acts similar to an ERC-20 token holding a balance value for each account asset. Balances in the MoneyMarket accrue interest over time, and we describe a low-gas method to achieve these balance updates below.

As clients of the protocol supply or borrow assets, the MoneyMarket updates the balance sheet entries for the appropriate asset by emitting two Ledger events (a credit and a debit). Additionally, we use this balance sheet to calculate the current interest rates for supplying and borrowing tokens described in section 2.4 above.

3.2 Account Checkpoints

The balance for each personal account or wallet address in the MoneyMarket is stored as an *account checkpoint*. An account checkpoint is a Solidity tuple *<uint256 balance, uint256 blockNumber>*. This tuple describes the balance at the time interest was last applied to that account. For example, if a customer supplies 20 ZRX tokens on block 15,000,000, we would store *<20, 15000000>*. Whenever any external event would require a customer's balance to update (e.g. when a user supplies or withdraws assets), we true up the customer's balance using a memoized lookup table. For example, if the customer above later initiates a withdrawal, that user sees an updated account checkpoint of *<20.5, 15200000>* immediately prior to processing that withdraw operation. This updated checkpoint indicates that the user has a balance of 20.5 ZRX tokens after interest was applied up to block 15,200,000. In this way, we allow accounts to effectively grow interest without paying excess gas fees.

3.3 Calculating Interest

Each money market contains two real-time, floating interest rates. The *InterestRateStorage* contract records this in the following way. Whenever any event would change the interest rate of an asset (which is any time an asset is supplied, borrowed or repaid), we calculate a new interest rate for that asset. We store a snapshot of both the interest rate and the interest applied to that asset since the beginning of time. That is, given an interest rate *i* applied at block *t*, if we take a snapshot *n* blocks later, we can calculate and store:

$$interest_{asset}(t + n, i) = interest_{asset}(t) + i \cdot n$$

Recursive calculation of interest

The snapshots of this *interest* function are sparse. That is, we only calculate or store the value when we know the interest rate has changed for block *t*. The interest rate, *i*, is calculated by the *InterestModel* contract which implements the formulas in section 2.4.

We can use the interest rate snapshots to calculate the balance of an account based on snapshots. Given interest rate snapshots at block t and block $t + n$, for an account with an account snapshot of <b, t> where b and t are the balance and timestamp of the account snapshot respectively:

$$balance_{account,\, asset}(t+n) = b \cdot (\ 1\ +\ interest_{asset}(t+n) - interest_{asset}(t)\ )$$

Calculating an account balance with interest from snapshots

That is, given an account which had a snapshot of balance b at block t, the balance at block $t+n$ can be calculated by multiplying the balance by the difference in interest snapshots between t and $t+n$. We know that both of the interest snapshots exist since we always snapshot when an account balance changes (since account balances affect the balance sheet and thus the interest rate).

When the balance sheet of a money market is updated, the prevailing interest rates in the market update, to immediately reflect changes in the relationship between supply and demand.

3.4 Borrowing

A user who wishes to borrow and who has sufficient balances stored in Compound may call *customerBorrow(address asset, uint amount)* on the MoneyMarket contract. This function call will update both the user's positive balance and loan balance to reflect the newly borrowed assets. The user is free to withdraw the borrowed assets so long as he or she maintains a sufficient collateral ratio.

Loans accrue interest in the exact same fashion as balance interest was calculated in section 3.3; a borrower has the right to repay an outstanding loan at any time, by calling `*customerPayLoan(address asset, uint amount)*` which repays the loan's principal and accrued interest.

If a user ever falls below the *liquidation ratio* due to the value of his or her collateral changing (e.g. the user is holding ZRX as collateral to borrow Ether and ZRX significantly falls in value), then we have a public function `*liquidateCollateral(address customer, address collateralAsset, address borrowAsset, uint collateralAmount)*` which exchanges the invoking user's asset for the borrower's collateral, at a slightly better than market price.

A *PriceOracle* maintains the current exchange rate of each supported asset; the Compound token-holders are responsible (and incentivized) to frequently update these prices. These exchange rates are used to determine borrowing capacity and collateral requirements, and for all functions which require calculating the value equivalent of an account.

3.5 Contract Interface

Contract	Function ABI	Action
MoneyMarket	`customerSupply(address asset,`	Transfers an ERC-20 token into

	`uint256 amount, address from)`	MoneyMarket and marks customer's updated balance.
MoneyMarket	`customerWithdraw(address asset, uint256 amount, address to)`	Transfers an ERC-20 to withdrawal address if funds available.
MoneyMarket	`customerBorrow(address asset, uint amount)`	Augments customer's balance in asset by amount if customer holds sufficient collateral.
MoneyMarket	`customerPayBorrow(address asset, uint amount)`	Pays down principal and interest on a borrow, either in original asset or from collateral if customer lacks sufficient cover.
MoneyMarket	`liquidateCollateral(address customer, address collateralAsset, address borrowAsset, uint collateralAmount)`	Pays down principal and interest on a borrow by exchanging the borrowed asset for the borrower's collateral. This function can only be called when a borrower breaches the liquidation ratio.
MoneyMarket	`accrueBalanceInterest(address customer, address asset)`	Immediately trues up account by adding interest to balance. Also called internally.
MoneyMarket	`accrueBorrowInterest(address asset, uint amount)`	Immediately trues up borrow by adding due interest. Also called internally.
MoneyMarket	`snapshotCurrentRate(address asset)`	Creates a new snapshot for the current block unit \`floor(block.number / 10000)\`. Public and can only be called once per block unit
PriceOracle	`setAssetPrice(address asset, uint256 value)`	Sets the PriceOracle price of a given asset. Called by a representative of token holders.

Table 1. ABI and Explanation of select MoneyMarket and PriceOracle functions on the Blockchain

3.6 Alternate Approach

The Compound protocol is effective, but is limited by any issues in the Ethereum blockchain itself (e.g. fluctuations in gas prices, delays in transaction processing, etc). Additionally, the proposed protocol is specified tied to Ethereum and ERC-20 tokens, but does not allow users to supply Bitcoin or other cryptocurrencies.

A Compound protocol sidechain could house the interest rate model and governance systems, and store customer assets; this would shift the most complicated computations (asset pricing, and interest rate calculations) off-chain where they can be accomplished inexpensively.

The sidechain can be bonded to the Ethereum blockchain through a bridge contract; users would be able to verifiably transfer assets into the protocol by transferring Ether and tokens into the bridge contract, and getting a merkle proof when the funds have been delivered to the sidechain. The Compound protocol sidechain would run independently, and every so often, post its merkle root to the Ethereum blockchain. At any point, a user would withdraw funds from the sidechain back to the bridge contract.

4 Protocol Token

Blockchain protocols create financial incentives in order to align rational economic agents to coordinate behavior towards a common goal [7]. The Compound protocol aligns two stakeholders with opposite goals (one seeking to maximize borrowing costs, the other to minimize them) by introducing a third stakeholder, the "owner", represented by the protocol token, which is incentivized to maintain healthy money markets.

Compound will be deployed to the Ethereum blockchain with a fixed supply of protocol tokens that will be issued to the project sponsor, members of the community, developers, partners, and investors. The token-holders exercise governance over the protocol by:

- Proposing and implementing updates to the interest rate model
- Proposing and implementing updates to the collateral ratio, liquidation ratio, and liquidation discount
- Posting asset prices to the price oracle

Initially, Compound Labs, Inc., the protocol sponsor, will manage the protocol until a more sophisticated voting mechanism is developed and governance is shifted entirely to the token-holders.

4.1 Economic Implications for Protocol Token

Economic profit (the spread between lending and borrowing rates) is native to the Compound protocol and each money market; in exchange for responsibly managing the variables and enforcing the rules of the protocol, the token-holders will be entitled to 100% of the residual profit.

5 Summary

- Compound creates properly functioning money markets for Ethereum assets

- Each money market has interest rates that are determined by the supply and demand of the underlying asset
- Users can supply tokens to a money market to earn interest, without trusting a central party
- Users can borrow a token (to use, sell, or re-lend) by using their balances in the protocol as collateral
- Compound token-holders maintain the protocol, and earn the economics of the system

---

References

[1] Cryptocurrency Market Capitalizations. https://coinmarketcap.com/
[2] Bitfixex Margin Funding Guide. https://support.bitfinex.com/
[3] ETHLend White Paper. https://github.com/ETHLend
[4] Ripio White Paper. https://ripiocredit.network/
[5] Lendroid White Paper. https://lendroid.com/
[6] dYdX White Paper. https://whitepaper.dydx.exchange/
[7] Fred Ehrsam: The Decentralized Business Model. https://blog.coinbase.com/