Compound:
The Money Market Protocol

Version 1.0
February 2019

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 exchange the time value of 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 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 establishes money markets, which are pools of assets with algorithmically derived interest rates, based on the supply and demand for the asset. Suppliers (and borrowers) of an asset interact directly with the protocol, earning (and paying) a floating interest rate, without having to negotiate terms such as maturity, interest rate, or collateral with a peer or counterparty.

Each money market is unique to an Ethereum asset (such as Ether, an ERC-20 stablecoin such as Dai, or an ERC-20 utility token such as Augur), and contains a transparent and publicly-inspectable ledger, with a record of all transactions and historical interest rates.

2.1 Supplying Assets

Unlike an exchange or peer-to-peer platform, where a user's assets are matched and lent to another user, the Compound protocol aggregates the supply of each user; when a user supplies an asset, it becomes a fungible resource. This approach offers significantly more liquidity than direct lending; unless *every* asset in a market is borrowed (see below: the protocol incentivizes liquidity), users can withdraw their assets at any time, without waiting for a specific loan to mature.

Assets supplied to a market are represented by an ERC-20 token balance ("cToken"), which entitles the owner to an increasing quantity of the underlying asset. As the money market accrues interest, which is a function of borrowing demand, cTokens become convertible into an increasing amount of the underlying asset. In this way, earning interest is as simple as holding a ERC-20 cToken.

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 Augur can supply their tokens to the Compound protocol, and earn interest (denominated in Augur) 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 Assets

Compound allows users to frictionlessly borrow from the protocol, using cTokens as collateral, for use anywhere in the Ethereum ecosystem. 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.

2.2.1 Collateral Value

Assets held by the protocol (represented by ownership of a cToken) are used as collateral to borrow from the protocol. Each market has a collateral factor, ranging from 0 to 1, that represents the portion of the underlying asset value that can be borrowed. Illiquid, small-cap assets have low collateral factors; they do not make good collateral, while liquid, high-cap assets have high collateral

factors. The sum of the value of an accounts underlying token balances, multiplied by the collateral factors, equals a user's ***borrowing capacity***.

Users are able to borrow up to, but not exceeding, their borrowing capacity, and an account can take no action (e.g. borrow, transfer cToken collateral, or redeem cToken collateral) that would raise the total value of borrowed assets above their borrowing capacity; this protects the protocol from default risk.

2.2.2 Risk & Liquidation

If the value of an account's borrowing outstanding exceeds their borrowing capacity, a portion of the outstanding borrowing may be repaid in exchange for the user's cToken collateral, at the current market price minus a ***liquidation discount***; this incentives an ecosystem of arbitrageurs to quickly step in to reduce the borrower's exposure, and eliminate the protocol's risk.

The proportion eligible to be closed, a ***close factor***, is the portion of the borrowed asset that can be repaid, and ranges from 0 to 1, such as 25%. The liquidation process may continue to be called until the user's borrowing is less than their borrowing capacity.

Any Ethereum address that possesses the borrowed asset may invoke the liquidation function, exchanging their asset for the borrower's cToken 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 order-books.

2.2.3 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 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 interest rate equilibrium, in each money market, based on supply and demand. 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 utilization ratio $\boldsymbol{U}$ for each market $\boldsymbol{a}$ unifies supply and demand into a single variable:

$$U_a = Borrows_a \,/\, (Cash_a + Borrows_a)$$

The demand curve is codified through governance and is expressed as a function of utilization. As an example, borrowing interest rates may resemble the following:

$$Borrowing\ Interest\ Rate_a \;=\; 2.5\% + U_a * 20\%$$

The interest rate earned by suppliers is *implicit*, and is equal to the borrowing interest rate, multiplied by the utilization rate.

2.3.1 Liquidity Incentive Structure

The protocol does not guarantee liquidity; instead, it relies on the interest rate model to incentivize it. In periods of extreme demand for an asset, the liquidity of the protocol (the tokens available to withdraw or borrow) will decline; when this occur, interest rates rise, incentivizing supply, and disincentivizing borrowing.

3 Implementation & Architecture

At its core, a Compound money market is a ledger that allows Ethereum accounts to supply or borrow assets, 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 cToken Contracts

Each money market is structured as a smart contract that implements the ERC-20 token specification. User's balances are represented as cToken balances; users can `mint(uint amountUnderlying)` cTokens by supplying assets to the market, or `redeem(uint amount)` cTokens for the underlying asset. The price (exchange rate) between cTokens and the underlying asset increases over time, as interest is accrued by borrowers of the asset, and is equal to:

$$exchangeRate \;=\; \frac{underlyingBalance + totalBorrowBalance_a - reserves_a}{cTokenSupply_a}$$

As the market's total borrowing balance increases (as a function of borrower interest accruing), the exchange rate between cTokens and the underlying asset increases.

Function ABI	Description
`mint(uint256 amountUnderlying)`	Transfers an underlying asset into the market, updates msg.sender's cToken balance.

`redeem(uint256 amount)` `redeemUnderlying(uint256 amountUnderlying)`	Transfers an underlying asset out of the market, updates msg.sender's cToken balance.
`borrow(uint amount)`	Checks msg.sender collateral value, and if sufficient, transfers the underlying asset out of the market to msg.sender, and updates msg.sender's borrow balance.
`repayBorrow(uint amount)` `repayBorrowBehalf(address account, uint amount)`	Transfers the underlying asset into the market, updates the borrower's borrow balance.
`liquidate(address borrower, address collateralAsset, uint closeAmount)`	Transfers the underlying asset into the market, updates the borrower's borrow balance, then transfers cToken collateral from the borrower to msg.sender

Table 2. ABI and summary of primary cToken smart contract functions

3.2 Interest Rate Mechanics

Compound money markets are defined by an interest rate, applied to all borrowers uniformly, which adjust over time as the relationship between supply and demand changes.

The history of each interest rate, for each money market, is captured by an *Interest Rate Index*, which is calculated each time an interest rate changes, resulting from a user minting, redeeming, borrowing, repaying or liquidating the asset.

3.2.1 Market Dynamics

Each time a transaction occurs, the Interest Rate Index for the asset is updated to compound the interest since the prior index, using the interest for the period, denominated by r * t, calculated using a per-block interest rate:

$$Index_{a,n} = Index_{a,(n-1)} * (1 + r * t)$$

The market's total borrowing outstanding is updated to include interest accrued since the last index:

$$totalBorrowBalance_{a,n} = totalBorrowBalance_{a,(n-1)} * (1 + r * t)$$

And a portion of the accrued interest is retained (set aside) as reserves, determined by a ***reserveFactor***, ranging from 0 to 1:

$$reserves_a = reserves_{a,(n-1)} + totalBorrowBalance_{a,(n-1)} * (r * t * reserveFactor)$$

3.2.2 Borrower Dynamics

A borrower's balance, including accrued interest, is simply the ratio of the current index divided by the index when the user's balance was last checkpointed.

The balance for each borrower address in the cToken is stored as an *account checkpoint*. An account checkpoint is a Solidity tuple *<uint256 balance, uint256 interestIndex>*. This tuple describes the balance at the time interest was last applied to that account.

3.3 Borrowing

A user who wishes to borrow and who has sufficient balances stored in Compound may call `borrow(uint amount)` on the relevant cToken contract. This function call checks the user's account value, and given sufficient collateral, will update the user's borrow balance, transfer the tokens to the user's Ethereum address, and update the money market's floating interest rate.

Borrows accrue interest in the exact same fashion as balance interest was calculated in section 3.2; a borrower has the right to repay an outstanding loan at any time, by calling `repayBorrow(uint amount)` which repays the outstanding balance.

3.4 Liquidation

If a user's borrowing balance exceeds their total collateral value (borrowing capacity) due to the value of collateral falling, or borrowed assets increasing in value, the public function `liquidate(address target, address collateralAsset, address borrowAsset, uint closeAmount)` can be called, which exchanges the invoking user's asset for the borrower's collateral, at a slightly better than market price.

3.5 Price Feeds

A *Price Oracle* maintains the current exchange rate of each supported asset; the Compound protocol delegates the ability to set the value of assets to a committee which pools prices from the top 10 exchanges. 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.

3.6 Comptroller

The Compound protocol does not support specific tokens by default; instead, markets must be whitelisted. This is accomplished with an admin function, `supportMarket(address market, address interest rate model)` that allows users to begin interacting with the asset. In order to borrow an asset, there must be a valid price from the Price Oracle; in order to use an asset as collateral, there must be a valid price and a collateralFactor.

Each function call is validated through a policy layer, referred to as the *Comptroller*; this contract validates collateral and liquidity, before allowing a user action to proceed.

3.7 Governance

Compound will begin with centralized control of the protocol (such as choosing the interest rate model per asset), and over time, will transition to complete community and stakeholder control. The following rights in the protocol are controlled by the admin:

- The ability to list a new cToken market
- The ability to update the interest rate model per market
- The ability to update the oracle address
- The ability to withdraw the reserve of a cToken
- The ability to choose a new admin, such as a DAO controlled by the community; because this DAO can itself choose a new admin, the administration has the ability to evolve over time, based on the decisions of the stakeholders

4 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; when demand to borrow an asset grows, or when supply is removed, interest rates increase, incentivizing additional liquidity
- 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

---

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/