The Kowala Protocol: A Family of Distributed, Self-Regulating, Asset-Tracking Cryptocurrencies

Eiland Glover and John W. Reitano

Abstract

Cryptocurrencies such as Bitcoin, Ether, and Dash exhibit significant volatility. Consumers, merchants, traders, investors, miners and developers have a need for a cryptocurrency whose value can be counted on to remain roughly stable from one day to the next and whose operation does not depend on potentially unreliable third parties such as banks.

Cryptocurrencies with automated value-pegging mechanisms, such as NuBits and BitUSD, have suffered repeated failures due to malfunction and lack of adoption[1], while companies that hold centralized, one-to-one reserves in fiat, such as Tether, are vulnerable to fraud or a government shutdown[2].

The *Kowala Protocol* is our proposed method for creating a new family of cryptocurrencies which maintain stable values while retaining other benefits of cryptocurrencies, such as decentralization, security, privacy, speed of transfer, and low transaction costs.

[1] See https://www.cato.org/blog/dollar-denominated-cryptocurrencies-flops-tethered-success
[2] See https://medium.com/crypto-punks/tether-misconceptions-2984430e7e48

Why We Believe A Stable Cryptocurrency Is Needed

Beyond the considerable benefits that many cryptocurrencies can bring, a stable cryptocurrency has the potential to provide unique benefits to many stakeholders, such as consumers, merchants, investors, traders, miners and developers. Assuming theoretical universal adoption of such a stable cryptocurrency is realized, potential benefits to these stakeholders include:

Consumers:

- avoiding traditional cryptocurrency volatility when purchasing real-world products and services;
- avoiding volatility of less stable fiat currencies (such as of those of Venezuela, Zimbabwe, Nigeria, etc)
- providing a stable store of value; and
- gaining access to bank-like services offered by third parties (for consumers without easy access to bank accounts denominated in stable fiat currencies).

Merchants:

- avoiding volatility when selling products and services for cryptocurrency; and
- gaining access to bank-like services offered by third parties (for merchants without easy access to bank accounts denominated in stable fiat currencies).

Investors:

- parking funds in a decentralized, stable asset.

Traders:

- pursuing arbitrage opportunities in new cryptocurrency markets;
- parking funds in a decentralized asset that has stable value; and
- trade in and out of stable asset at low cost and high speed.

Developers:

- incorporating payment functionality into apps and websites without the need to establish a merchant account; and
- incorporating such payment functionality while avoiding traditional cryptocurrency volatility.

Important notes:

1. Any forward-looking discussion of kCoins and mTokens, currency specific or otherwise, mining, trading, contracts, and/or rewards is theoretical in nature and should be understood as such. At this time, no mTokens have been issued, and thus no kCoins have been created. Further, there can be no assurance that any coin or token will be widely adopted or that any market or exchange will develop in which any coin or token may be bought, sold, or otherwise transacted.

2. There are inherent limitations to back-tested data, also described as “modeled”, “theoretical”, or “hypothetical”. The results of such testing are not representative of those obtained in a live market scenario, and actual results may differ significantly from those presented here.

What is the Kowala Protocol?

The Kowala Protocol defines a method for constructing a family of distributed, self-regulating, asset-tracking cryptocurrencies called *kCoins*.

Each kCoin is designed to be traded on open exchanges and to maintain a close to one-to-one value relative to any widely traded asset such as a currency (USD, EUR, JPY, etc.) or other asset. Each kCoin will be identified by a symbol consisting of the letter *k* followed by the symbol of the underlying asset. For example, the kCoin of USD is kUSD, that of EUR is kEUR, and so forth.

kCoins will constantly gather market information from publicly available sources and regulate their value through three core mechanisms: minting variable amounts of coins, applying variable fees, and obtaining feedback from an active and well-informed trading market. In time, these mechanisms should always return each kCoin to parity with its underlying, tracked asset. Anticipation of a kCoin's eventual return to parity, in turn, has the potential to create pure arbitrage opportunities for traders seeking to profit from slight fluctuations in kCoin market prices.

We chose the codebase of the Go Ethereum client[3] as the starting point for the development of kCoins in order to access both Ethereum's strong feature set (especially its sophisticated smart contract facilities) and the collective abilities and ongoing efforts of its development team. On top of this foundation, each kCoin is designed to include the value stabilization and market observation features described above. Finally, to achieve ultra-fast transaction processing performance, we have replaced the Ethereum consensus protocol with Konsensus, an implementation of PBFT derived from Tendermint. Because every kCoin will need a robust exchange market to function properly, each kCoin will be implemented as a distinct, independent blockchain with its own tokens, smart contracts, mining community, etc.

Note: The discussion below applies to any kCoin, but, for ease of explanation, we will use the specific example of kUSD—the kCoin pegged to the U.S. dollar—in most of the remainder of this white paper.

[3] See https://github.com/ethereum/go-ethereum

Key Components of the Kowala Protocol

At the center of the Kowala Protocol are three stability-related mechanisms that are designed to keep the market price of kUSD at or very near $1.

The section *Stability Mechanism 1: Minting Algorithm* below describes how a variable amount of newly minted coins will be used to push the market price toward the target of $1 when necessary.

The section *Stability Mechanism 2: Stability Fee* describes how, in the scenario in which the market price is below $1 and a low minted amount is not sufficient to bring it back to $1, a special, variable fee will be applied to each subsequent transaction until the market price begins to rise.

The section *Stability Mechanism 3: Trading Activity* describes how traders, informed by the previous two mechanisms and motivated by a desire for profit, are incentivized to engage in trading activity that should accelerate the return of the market price of kUSD to $1.

Beyond these stability mechanisms, the other principal components of the Kowala Protocol are a unique consensus protocol (see section *Konsensus*), a system for reporting prices via oracles (see section *Price Oracles*), and rewards for users who hold balances in a way that benefits the network (see section *Stability Rewards and Stability Contracts*.

Note that all aspects of the Kowala Protocol described in this white paper are subject to change.

Stability Mechanism 1: Minting Algorithm

To calculate the amount of new coins to be minted, we will need to introduce four new concepts.

The first concept is a *dead-end address*, which is an address that is precluded from having outbound transactions and thus whose balance can increase but not decrease.

The second concept is the *available coin supply*, which is the total value of all coins that are available for sending via transactions. Formally this is defined with the following function:

$$availableCoinSupply(b) := totalCoinSupply(b) - \sum_{d \in D(b)} balance(d, b)$$

where *totalCoinSupply(b)* refers to the total number of coins issued as of block number *b*, *D(b)* is the set of all dead-end addresses as of block *b*, and *balance(d, b)* is the balance of dead-end address *d* as of block *b*. In other words, the available coin supply is the total number of sendable coins.

The third concept is the *minted amount cap*, *cap(b)*, which is defined as:

$$cap(b) := \begin{cases} 0.0001 \times availableCoins(b-1) & b > 1, availableCoins(b-1) \geq 1000000 \\ C & otherwise \end{cases}$$

where C is a fixed initial cap that applies when the available coin supply is very low (under 1 million coins). Initially, we set C to 82, since this achieved good results in our testing, but this constant can be changed in future updates to the protocol.
The minted amount cap is expected to ensure that, after an initial growth period, the newly minted amount will never be larger than one basis point of the entire available coin supply.

The fourth concept is the *market price of* kUSD, *p(b)*, which is determined by an oracle-based system designed to come to a trustworthy and accurate consensus on the value of kUSD as reported publicly on exchanges (see the section *Price Oracles* below).

With all these concepts in place, we are ready to define the amount of new coins to be minted, *mintedAmount(b)*, as:

$$mintedAmount(b) := \begin{cases} 42 & b = 1 & (initial) \\ mintedAmount(b-1) & b \bmod n \neq 1 & (no-change) \\ min(UP \times mintedAmount(b-1), cap(b)) & b > 1, b \bmod n = 1, p(b) \geq p(b-1) \geq 1 & (ratchet-up) \\ max(\frac{mintedAmount(b-1)}{DOWN}, 0.000001) & otherwise & (ratchet-down) \end{cases}$$

where *UP* and *DOWN* are adjustment factors. Initially, we set *UP* to 1.050 and *DOWN* to 1.777, since these achieved good results in our testing, but these factors may be changed in future updates to the protocol.

The calculation of *mintedAmount(b)* is split into four scenarios: *initial*, no-change, *ratchet-up, and ratchet-down.*

During the *initial scenario*, which occurs only for the first block, the minted amount is set to the arbitrary value of 42.

During the *no-change scenario*, which occurs whenever the current block is not the first block of an n-block interval, we simply leave the minted amount at the value it had for the

prior block. Currently, we use an interval size of 900 blocks, or approximately 15 minutes, though this may be adjusted in the future.

Next, we consider the *ratchet-up scenario*, which occurs when, at the start of an n-block interval the price of kUSD is at or over $1 and is rising or flat, when compared to the prior block. In this scenario, the minted amount is set to a small increase (currently 5%) over the previous interval's per-block minted amount. This increased amount is limited by the minted amount cap.

We initially hypothesized that, when large numbers of newly minted coins are earned by miners, a large portion of such coins will reach exchanges as market sell orders and drive down the price of kUSD. A detailed agent-based behavior model with multiple market scenarios supports this hypothesis (see *Agent-Based Modeling* below). For this reason, we posit that for the ratchet-up scenario, no further mechanism is needed to reduce the price to $1.

Next, we consider the remaining logical possibility, which we label the *ratchet-down scenario*. This scenario occurs in when, at the start of an n-block interval the price of kUSD is below $1 and, compared to the prior block, is falling or flat. In this scenario, we do not want to mint additional kUSD since this would likely reduce the price, which is at best unnecessary and at worst harmful. For this reason, the ratchet-down portion of the function states that we should gradually reduce the minted amount, dividing the prior minted amount by our adjustment factor (currently 1.777), subject to a minimum of 0.000001 kUSD.

Repeated applications of the formula in the ratchet-down scenario during a prolonged drop in price can quickly lower the minted amount to nearly zero. For example, in this scenario, reducing the minted amount from 1 kUSD to under 0.0.0001 kUSD takes only 16 intervals (approximately 4 hours). However, even a near-zero minted amount may not be sufficient to raise the price of the coin if there is a large drop in coin demand during the same period. In the section *Stability Mechanism 2: Stability Fee* below, we will address this insufficiency by introducing a way to materially reduce the total coin supply.

Stability Mechanism 2: Stability Fee

Like Ethereum. the Kowala Protocol stipulates that every transaction sender be charged a fee, called the *transaction fee*, part of which is used to compensate miners for their computational effort, since this effort is essential for maintaining the network. However, the transaction fee in kUSD does more than compensate miners for this effort. It also includes an additional amount, called the *stability fee*, which ranges between 0 and 2% of the transaction amount. The primary purpose of the stability fee is to raise the price of kUSD by reducing coin supply

when the price of kUSD is below $1. The following formula shows the two parts that comprise the transaction fee:

$$transactionFee(t) := computeFee(t) + stabilityFee(t)$$

where *computeFee(t)* refers to the compute fee (see section *Compute Fee* below) and *stabilityFee(t)* refers to the stability fee, which will be defined later in this section.

The stability fee should conform to the following constraints:

- under normal conditions, the stability fee is zero; and
- in the abnormal condition of a prolonged drop in demand for kUSD, the stability fee should still represent no more than 2% of the transaction amount and should be lowered back down to zero once normal conditions return.

In real-time uses of a payment system, we will also need to define a formula for an *estimated stability fee,* which should conform to the following constraints:

- it is easy to calculate; and
- it is either equal to or just slightly above the actual fee, assuming the actual transaction takes place within 15 minutes of calculating the estimate.

The stability fee is initially set to the same value as the compute fee. This ensures that transactions that impose more computational costs on the network will bear more of the costs of maintaining stability. Then we will periodically increase it by a small amount (9%) until it rises to its maximum value (2% of the transaction amount).

For the purpose of defining the stability fee and estimated stability fee formally, we will introduce the function *r(b)*, which tells us how many times the stability has been increased after being set to the amount of the compute fee:

$$r(b) := \begin{cases} -1 & b = 1 & (initial) \\ r(b-1) & b \bmod n \neq 1 & (no-change) \\ r(b-1)+1 & b > 1, b \bmod n = 1, p(b) \leq p(b-1) < 1 & (ratchet-up) \\ max(r(b-1)-1, -1) & otherwise & (ratchet-down) \end{cases}$$

The definition of *r(b)* implies that its value changes only at the first block of each interval, if at all. As we will see below, a value of less than 0 for r(b) will used to signify that that the stability fee should be set to the trivial value of 0.

We next define the base *stability fee* for transaction *t*.

$$baseStabilityFee(t) := \begin{cases} 1.09^{r(block(t))} \times computeFee(t)) & r(block(t)) \geq 0 \\ 0 & otherwise \end{cases}$$

where *block(t)* is the block number associated with transaction *t*. See *Compute Fee*, below, for a definition of *computeFee(t)*. We are now ready to define the *stability fee* and the *estimated stability fee* for transaction *t*:

$$stabilityFee(t) := min(baseStabilityFee(t), 0.02 \times amount(t))$$

$$estimatedStabilityFee(t) := min(1.09 \times stabilityFee(t), 0.02 \times amount(t))$$

where *amount(t)* is the amount associated with transaction *t*.
Once received from the transaction sender, the stability fee will be split into two equal parts; one half will be burned (i.e., transferred to a dead-end address), and the other half will be distributed to certain holders of kUSD in order to encourage them to hold increased balances at key times. See the sections *Coin Burning* and *Stability Rewards and Stability Contracts* below for details on how these quantities are handled.

Our modeling indicated that a non-zero stability fee was applied infrequently in anticipated market conditions and was effective in raising the price of kUSD when applied (see *Agent-Based Modeling* below).

Stability Mechanism 3: Trading Activity

Given sufficient time, the first two mechanisms above should cause the price of kUSD to revert to parity (i.e., 1 USD per kUSD). We label this tendency to revert to parity the *first-order effect* of the Kowala Protocol. Once the first-order effect has become recognized by market participants, we expect three *second-order effects* to occur.

First, since the parity price is the only particular price to which kUSD should have a natural inclination to return, this price constitutes a game-theoretic focal (or Schelling) point[4]. Based on the focal point theory, we believe that the absence of perfect communication and trust among disparate market participants along with the status of the parity price as a focal point increase the likelihood that the price of kUSD will return to parity.

[4] See https://en.wikipedia.org/wiki/Focal_point_(game_theory)

Second, whenever deviations of the price from parity do occur, they will give rise to short-term profit opportunities for professional arbitrageurs. The source of these opportunities is the difference between the time horizon of relatively patient arbitrageurs and that of other market participants for whom the short-term need to move into or out of kUSD exceeds concern over these price deviations. Although arbitrageurs may exploit these profit opportunities purely for self-interest, their trading activity should have the positive effect of accelerating the return to parity.

Third, confidence in the reversion to parity should strengthen further as participants observe a history of such reversion in the marketplace.

All of these second-order effects depend on the development of a well-functioning USD/kUSD (or equivalent BTC/kUSD or ETH/kUSD) exchange market. Since the existence of such exchange markets are in the interest of exchanges, miners, and users of kUSD, we believe it is highly likely that they will arise through the self-interested activities of these groups.

For its part, Kowala plans to participate in independent, open-market, profit-seeking trading activities based on the same information publicly available to all.

Compute Fee

The *compute fee* is the portion of the transaction fee which is transferred to the miners of kUSD and is determined by the following formula:

$$computeFee(t) := gasUsed(t)gasPrice(t)$$

where *gasUsed(t)* is the gas used in processing transaction *t*, and *gasPrice(t)* is the price of one unit of gas at the time of transaction t. Both gasUsed(t) and gasPrice(t) are defined similarly to their analogs in Ethereum, and the compute fee itself is similar Ethereum's "transaction fee". For simple send transactions, gasUsed(t) is set to 21,000, and, for smart contract transactions, its value will depend on the operations triggered by the associated smart contract. The default value of *gasPrice(t)* is currently set to 2×10^{-8}.

The sum of compute fees for all transactions in the current block will be awarded to the elected proposer of the block:

$$computeReward(b) := \sum_{t \in T(b)} computeFee(b, t)$$

Coin Burning

The burn amount is defined as half of the sum of all the block's stability fees:

$$burnAmount(b) := \frac{\sum_{t \in T(b)} stabilityFee(t)}{2}$$

where T(b) is the set of all transactions that form block b.

The burn amount is sent to a dead-end address, which has the effect of permanently reducing the available supply of kUSD. Over time, the burn amount should allow kUSD to respond effectively to a decrease in demand. For example, if just 15% per day (on average) of the kUSD coin supply is transacted via on-chain transactions, the burn amount will reduce the coin supply by over 8% in 30 days and over 41% in 180 days.

Stability Rewards and Stability Contracts

The purpose of stability rewards is twofold: to allow network participation by an expanding circle of people and to incentivize behavior that supports network utility (by reducing the quantity of kUSD available for sale when the price is below $1).

The distribution of stability rewards to kUSD balance holders will be managed via a special-purpose smart contract called the *stability contract*. A participant who wishes to receive stability rewards for locking up funds will subscribe to the stability contract by registering an address, called the source address, and depositing the associated funds into the stability contract. To reduce the load on the system, participants will be required to deposit a minimum of 100 kUSD.

To prepare to define the stability reward amount, we first define the stability amount as the half of the total stability fees for the block. Note that this amount is equal to the burn amount (see *Coin Burning* above). In other words, half of the total stability fees are directed to the burn amount and the other half to the stability amount:

$$stabilityAmount(b) := \frac{\sum_{t \in T(b)} stabilityFee(t)}{2}$$

The stability amount is then partitioned into individual *stability rewards*, one for each stability contract, using the following formula:

$$stabilityReward(b,a) := \left(\frac{balance(b,a)}{\sum_{x \in A(b)} balance(b,x)}\right) stabilityAmount(b)$$

where *b* is a block number, *a* is a source address with an associated stability contract, *A(b)* is the set of all source addresses associated with stability contracts as of block b, and *balance(b,x)* denotes the balance as of block b of the source address x.

The pending stability rewards for each registered source address will be set aside within the stability contract. While a participant may unregister his source address and withdraw the associated funds at any time, if he does so before the price of kUSD returns to $1 or higher, he will forfeit any associated stability rewards.

mUSD and Other mTokens

The Kowala Protocol is a two-token system. The primary token in a Kowala Protocol blockchain will be a kCoin, that is, a stable-value payment token such as kUSD. In addition, associated with each kCoin will be a secondary, mining token or *mToken*; for example, the mToken associated with kUSD is called *mUSD*. Proof of ownership of a minimum number of these mTokens (currently 30,000) will be a requirement for mining the associated kCoin.

Each mToken will be associated with cryptographic addresses and other meta-information designed to securely manage a) tracking of performance of mining responsibilities, b) the recipient address for mining rewards, c) mToken ownership, and b) delegation of mining responsibilities to third parties. A specialized smart contract will be created to manage this information.

There will be 2^{30} (1,073,741,824) mTokens minted for each Kowala Protocol-based blockchain. Once the initial mTokens have been minted, a smart contract feature will prevent additional mTokens from being created in the future.

Konsensus: The Kowala Consensus Protocol

The consensus protocol of Kowala, *Konsensus*, is a formal set of rules that allows a decentralized group of participants to agree on the advancement of the blockchain, including the handling of transactions and the distribution of incentivising rewards.

Konsensus is derived from Tendermint[5], a mostly asynchronous consensus protocol which is itself based on Byzantine Fault Tolerance (BFT)[6]. We chose to base our work on Tendermint because it has achieved transaction performance that is superior to other widely used, proof-of-work-based blockchain consensus protocols[7], and because the resistance of its underlying protocol, BFT, to attack has been carefully studied[8]. Like Tendermint, Konsensus requires its participants to be in constant contact with each other.

In Konsensus, a network is comprised of two or more nodes connected by a well-defined communications protocol. A node, in this context, is a computer running a Kowala Protocol-compatible mining client. Communication will be achieved using devp2p, Ethereum's P2P protocol layer.

Like other cryptocurrencies, kUSD will be mined. For every block, all qualifying miners will have the chance to earn a reward of kUSD by contributing to the advancement and security of the blockchain. A miner is an active node on the network that controls a minimum number (currently set at 30,000) mTokens. In order to advance the blockchain, the network must include at least one miner, though we would typically expect there to be many more.

Mining in Konsensus will proceed one block at a time. For every block, a leader, called the *proposer*, is deterministically elected from the current miners. All other miners are known as *validators* for the duration of the block. The function of the proposer is to assemble and propose the next block in the chain, and the function of the validators is to vote on whether to accept the proposed block or reject it. The proposer first assembles a viable block that includes pending transactions and then distributes the block to validators for formal verification. Each validator inspects the block for suitability and then casts a cryptographically-signed vote to accept or reject the block. If two-thirds or more of the validators vote to accept the block, the proposer must commit it to the blockchain and signal to the entire network that the block is finalized. If the two-thirds threshold is not reached, the block is rejected. The process then begins again, with a new proposer elected to propose another block.

Election Based on mToken Ownership

Miners will not have an even chance of being elected as the proposer; rather, their likelihood of being chosen will be proportional to the number of mTokens associated with the node. The

[5] See https://tendermint.readthedocs.io/projects/tools/en/master/introduction.html
[6] See http://pmg.csail.mit.edu/papers/osdi99.pdf
[7] See https://blog.cosmos.network/a-tour-of-cosmos-for-developers-7517ba1b4045
[8] See https://www.usenix.org/legacy/event/nsdi08/tech/full_papers/singh/singh_html/index.html

likelihood of the election of a given miner is given by:

$$electionChance(m, b) = \frac{tokensHeld(m, b)}{\sum_{x \in M(b)} tokensHeld(x, b)}$$

where *m and M(b)* are a particular miner and the set of all miners, respectively, who are mining during the current block b. Over time, a miner will be elected proposer with a frequency proportional to the ratio of the number of tokens he or she holds to the total tokens held by all active miners. After being elected, a proposer must correctly perform the actual work of proposing and will be then rewarded for this work.

Price Oracles

The determination of the price of kUSD will be achieved through its *price oracle system*, which includes individual *price oracles*, associated rules, and a supporting smart contract. The price oracles are independent, highly-motivated miners who determine the data used to calculate the price of kUSD. Only eligible miners can participate as oracles. To be eligible, a miner must maintain a minimum stake of mUSD (currently 6 million) and not be banned by a majority of other oracles.

Each oracle will be incentivized by special rewards, called oracle rewards, to identify exchange trade data and to report that data by sending it to a special smart contract, called the *oracle smart contract*. This data must be reported at least once during each n-block interval. Currently, n is set to 900 blocks (approximately 15 minutes), but this interval may change. At the first block of each interval, the oracle smart contract uses an algorithm to transform the trade data reported during the prior interval into a single price. The core of the algorithm is to include only those recent transactions that are reported by a simple majority of oracles. This algorithm--which is still under development--will also weigh recent transactions more heavily than older ones, and may make other adjustments, such as discarding trades from exchanges that deviate excessively in reported trade prices from other exchanges.

The amount of the oracle reward will be a small percentage bonus on top of the portion of the newly minted amount the oracle would normally receive in exchange for proposing a block. This approach incentivizes oracles to discover and report trustworthy and accurate trade data, and piggybacks on the game-theoretic incentives and disincentives that apply to all miners.

To pay for oracle rewards, a small amount of the newly minted amount for each block will be deducted and placed into a system account called the *oracle account*. The amount of this *oracle deduction* is determined by the following formula:

$$oracleDeduction(b) := oracleDeductionFraction(b) mintedAmount(b)$$

The function *oracleDeductionFraction(b)* is not defined here but will return values between 0 and 0.01 in such a way that, in normal circumstances, the oracle account will have a balance sufficient to pay out anticipated oracle rewards.

We use the term *minted reward* to describe the portion of the minted amount that remains after the oracle deduction to be given to proposers. The amount of the minted reward is simply:

$$mintedReward(b) := mintedAmount(b) - oracleDeduction(b)$$

The base amount for the oracle reward is set to 4% of the minted reward for block b, subject to the limit of the actual balance in the oracle account:

$$baseOracleReward(b) := min(0.04\, mintedReward(b), fundBalance(b))$$

Beyond being eligible, an oracle must also be *qualified* to receive a reward (called the oracle reward) for doing the work of reporting trade data. To be qualified, an oracle must 1) consistently have reported trade data that has been substantially in consensus with data reported by other participating oracles during the n-block intervals covering the prior 6 hours, and 2) must be the proposer of the current block.

The oracle reward is then defined as the base oracle reward when a proposer is a an oracle that is qualified to receive such rewards and zero at all other times:

$$oracleReward(m, b) := \begin{cases} baseOracleReward(b) & m\, is\, a\, qualified\, oracle \\ 0 & otherwise \end{cases}$$

Mining Reward

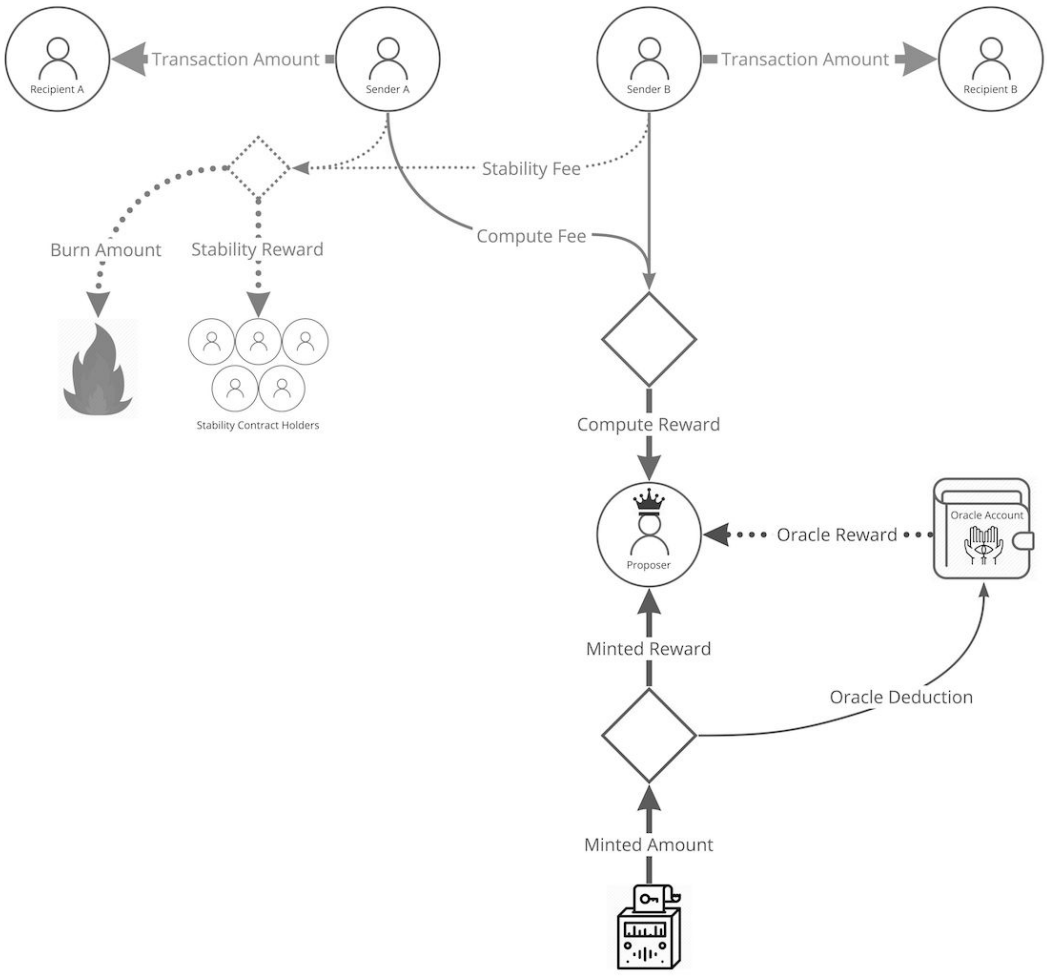
In Konsensus, the crucial work will be done by proposers. The proposer's role will require significantly more resources and energy expenditure than the validators'. The proposer must interrogate the blockchain to determine the amount to be minted (see *Stability Mechanism 1: Minting Algorithm*, above), assemble the block and ensure it is distributed properly—all in a short timeframe—which is non-trivial. Indeed, the processing of blocks would halt without a proposer, but, with a proposer, can continue even if any validators (or even all of them) leave

the network.

The total reward earned by the proposer for his or her work is called the *mining reward*, and is equal to the sum of three components: 1) the minted reward , 2) any applicable oracle reward, and 3) the compute reward:

$$miningReward(m, b) := \begin{array}{l} mintedReward(b) + \\ oracleReward(m, b) + \\ computeReward(b) \end{array}$$

where m is a miner and b is a block number. The following diagram shows these three components, and the source of their respective values:



Proposer Eligibility, Proposal Stake

Konsensus aims to punish dishonest or free-loading miners by making them ineligible to propose blocks (and thus earn rewards) for a game-theoretically sufficient number of blocks. For example, a would-be proposer must have participated in the prior 10,000 blocks as a validator in order to be eligible to propose the current block.

In order to receive mining rewards, proposers will be required to stake a minimum number of mUSD, initially to be set at 30,000 mUSD. Furthermore, for performance reasons, the roster of actively mining nodes will have a maximum size. Once the roster is full, in order to mine, a prospective mining node will need to offer a mining stake that is higher than the lowest stake of all nodes in the roster. A proposed stake that is greater than this lowest stake will be accepted and will bump the node with the lowest stake off the roster. This maximum size is expected to be 500 nodes when the kUSD blockchain launches. We expect to increase the maximum roster size to 5000 in the future by adjusting the consensus protocol or related mechanisms. Independent of roster size, mUSD owners may decide to combine their tokens in order to stake a sufficient number of mUSD to mine under these scenarios.

A special-purpose smart contract will be set to track mining activity and stakes. Using this contract, the mUSD owner will specify three things: the staked amount of mUSD, the node that is authorized to mine, and the address that will receive the kUSD mining rewards. Note that the reward receipt address is independent of the node performing the associated mining.

The amount staked is labeled a *proposal stake*. Normally, the proposal stake is reclaimed after the proposal is accepted or rejected, but if the proposal is determined to be dishonest, the stake is forfeited.

Transaction Speed

In Konsensus, there is no concept of post-creation confirmation. Blocks are irreversible once created. Compare this to Ethereum, where 6 or more 15-second block cycles may need to pass before a block is considered non-reversible.

We are actively investigating optimization of Konsensus, with the goal of processing 7,000+ transactions per second with a typical per-transaction processing time of one second. This anticipated performance would compete favorably with real-world commercial payment systems such as Visa.

Reduced Energy Usage

Because the Konsensus is designed to avoid the need for the expensive search for hash solutions, there is no incentive for miners to use powerful, energy-wasting mining hardware to outperform their fellow miners. This approach provides security by incentivizing the distribution of blockchain validation-by-consensus across many independent parties (because only active miners receive validation rewards) but should use very little electricity compared to typical cryptocurrency consensus mechanisms. For example, one analyst estimates that there are between 5,000 and 100,000 Bitcoin miners who collectively consume approximately 774 megawatts of electricity.[9] By comparison, we estimate that 100,000 miners could run kUSD client software on low-power machines and consume less than 5 megawatts of electricity.

Continuous Agent-Based Modeling

We have created a sophisticated agent-based software model to test the simultaneous use of all three mechanisms that form the Kowala Protocol. We have run a significant number of simulations against various permutations of the model, including:

- variations in constants used by the Minting Algorithm
- variations in constants used to define the Stability Fee
- variations in starting conditions
- market demand fluctuations, including mass panics
- rapidly increasing and decreasing numbers of participants
- excessive optimism and pessimism of arbitrageurs and prospectors

The algorithms and constants described in this white paper reflect the results of our testing, and the Kowala Protocol incorporates only the behaviors which consistently and reproducibly yielded the best results in our models.

Future agent-based modeling will be based on actual historical market data. Kowala will regularly create large-scale simulations of established marketplace behavior and conduct predictive research into further refinements to the Kowala Protocol. These enhanced simulations will feature agents whose behavior is derived from genetic algorithms informed by actual, historic market decisions. Kowala will also model potential attacks by malicious actors in order to preempt them and to uncover other unforeseen vulnerabilities.

[9] See https://bravenewcoin.com/news/number-of-bitcoin-miners-far-higher-than-popular-estimates

Conclusion

This white paper has identified the problem of volatility in cryptocurrency and proposed the Kowala Protocol as a robust solution to this problem. Although we have established through extensive modelling that the protocol worked in many anticipated scenarios, more work is needed to demonstrate with higher certainty that the specific mechanisms described here will work in a real-world market. We invite others to contribute to improving the Kowala Protocol by visiting https://kowala.tech and to participate in the development of the kCoin client software located at https://github.com/kowala-tech/kcoin.