**Introduction**

The execution price of an order on CoW should be the global onchain best possible attainable price for an order at a given timestamp (the batch auction expiry timestamp/the solver's time_limit

).

This price is in reality an abstraction of all types of liquidities and prices available for execution on-chain in the current Ethereum block. Solvers evaluate the best source, or combination of sources into a single solution, not only from available on-chain execution paths but also utilizing private sources of liquidity which could be their own liquidity or could be an integration with RFQ solution that is offchain, but holds liquidity onchain.

Liquidity provided into the CoW ecosystem, can be put on a spectrum that goes from least active (passive) to the most active. For example, permissionless on-chain pools in univ2-style AMMs represent the most passive type of LP, whereas univ3 pools represent a slightly more active LP and institutional market makers represent the most active type of LP (found on CEXs and fast CLOBs).

To picture what the "most active" liquidity looks like; the LP reacts to movements in global prices and can react by adjusting prices in lower double digit or even single milliseconds to events they monitor (order book orders, macro conditions, ETH price, etc). In contrast, the lack of activity of the univ2 AMM LP (by being confined to the Ethereum block times) exposes the LP to stale pricing occasionally when global prices trend fast in some direction (see LVR).

The key ideas of this post are:

1. Fast liquidity might on many occasions have the best universal price and hence it is desirable to have in CoW as one of abstracted away price points under the hood.

2. The current infra to have these fast liquidity actors in CoW is only through the solver path. Where either a solver uses a maker price in background, or the maker becomes a solver themselvs.

3. Becoming a solver requires a high stake, and therefore restricts the providers of this type of liquidity to be a large actor. The solver stake is there to protect the user in case of hiccups during settlement, and not to restrict sources of liquidity. The design space should allow for any user to participate in providing liquidity with 0 frictions as is the case with other major Defi protocols.

4. We have a sketch of a solution that we need the community's help in finalizing a spec for to make sure all stakeholders are aligned and everyone benefits. Where a truly optimal design might be an enshrined endpoint in the driver, we will focus as a starting point on a design that requires no current changes to any CoW service and can be deployed immediately.

**Character of a Fast LP**

Fast liquidity (the most active) differs to public on-chain sources of liquidity in two things:

1. Off-chain source of pricing: Fast LPs are not confined to the on-chain world. A lot more volume is traded off-chain than on-chain through centralized exchanges and otc desks. Also, off-chain trading is currently offering the lowest latency trading experience. It is therefore natural to have a lot more price discovery happening off-chain. These LPs are the masters of liquidity provision in that environment.

2. Timing matters in milliseconds. On-chain pricing is not time sensitive, time almost stops in an Ethereum block, and MEV related strategies on the searcher/builder infra is where ordering is determined. But with fast liquidity, time matters in milliseconds. If market makers are able to execute fast they can offer their best pricing. If they need to account for some holding timing risk, then they risk having a stale price, and hence add a margin (proportional to vol perhaps) to account for a 3s or 15s holding risk.

**Timing in CoW Protocol**

Given the time element is sensitive with respect to fast LPs, it is worth pondering how time works in CoW protocol if we want to integrate this type of liquidity in the most optimized way possible. Let us consider the order of events.

1. Receive order/batch auction ~30s

2. Solvers find best solutions ~13s

3. Driver determines winning solution ~<1s

4. Solution executed on-chain ~next block

[

877×354 18.5 KB

](https://europe1.discourse-cdn.com/business20/uploads/cow/original/1X/70b15452c0d14fa9f4804feee5ef7c3e4564333a.png)

The important time segment in the flow is the 13s solver time_limit. That is the time the solvers have in order to send back a solution. Because immediately after, the driver ranks the solutions, determines a winner and the settlement plan is executed on-chain. Hence, an active LP would want to react very near the end of the solver time_limit period. Fortunately, the driver can also send a notification back to inform them their solution was chosen.

Between a maker sending a quote, and receiving notification of chosen solution can in theory be less than 1s with optimizations (but 3s should be achievable today). Having short confirmation times for bids encourages tighter spreads from the market makers as holding risks are reduced and hedging can be done immediately.

### Design Goals

- Abstract all fast liquidity in a single endpoint to be used by solvers as if they choose any other UniV2 pool in solution, where this liquidity is represented in a fill-or-kill snapshot orderbook.
- Quick confirmation times for market makers
- Firm (guaranteed) maker bids for solvers to use in solutions with almost zero chance of execution failure.
- Unified Gateway; unified framerwork to eliminate redundnant interactions and api integrations.

### Sketch Solution: MooMaker

###### Requirements:

- The proposed solution is to have an offchain service called “MooMaker” that collects bids from makers and orders them like an orderbook and expose the final version of the orderbook some x time (perhaps 1s) before the time_limit for solvers to choose from in their solutions. Solvers can choose to totally ignore it or to utilize the price if it improves existing solution they are calculating.
- The orders in the CoW batch are publically available at the CoW orderbook endpoint. So no need to relay the orders in the CoW batch, however, what is missing is the time_limit

which is usually sent to the solver in the POST request. This time limit resembles the cow batch auction expiry. So we would like to relay a time limit to the makers to have a cutoff time for sending bids with regards to a given CoW batch.

- Importantly, the CoW driver notifies the solver if solution was chosen immediatley (200-300ms), but we also need to relay this information to the makers whether their solution is chosen.

###### Solution Components:

1. Off-chain MooMaker Service: Makers receive info about time_limit

, are able to send signed bids to (EIP-712), and receive notification about chosen bid as confirmation.

1. Smart Contract: Validates bids and executes transfers. Provides security for MM bids by validating.

### Open questions

- Bidding rules: Private bids? Public bids? Allow makers to cancel bids?
- In current workflow between solvers and RFQ systems, there is back and forth communication around, 1)indicative pricing 2)required qty 3)price levels. Does the current proposed flow and orderbook structure cover all desired communication? For example, no need relay the required qty/tokens from solver to maker, since all CoW orders are public. But perhaps the solver wants a price for a specific leg which represents different tokens than the CoW batch. How important is this?
- The orderbook structure covers the price levels requirement. Since the maker can submit their bids as price levels where each price level has a quantity associated with it. This is most frictionless experience. Is this true? And are makers comfortable sending signed bids like that? Makers in general want to have a final say before signing an order.
- Security concerns: Design a secure EIP-712 signed bids that protects market makers but also enforces the bid. What changes do we need to make to our PoC smart contract implementation.

### Resources

- PoC implementation: MooMaker · GitHub

- Test transaction (Goerli): Goerli Transaction Hash (Txhash) Details | Etherscan