TLDR

- Orderflow auctions (OFA) promise to return users the value they create via their orders. We believe this will be the year of OFAs and expect significant competition happening between different OFA designs and their implementations in the next 6 months.
- We surveyed 12 independent orderflow auction implementations against a core framework to identify their key design decisions and tradeoffs.
- We predict OFA implementations will cluster around four distinct designs which serve independent needs around price discovery and execution, as well as find efficiencies from specialization towards swap use cases or arbitrary transactions.

Introduction

In crypto today, when a user generates a transaction, MEV searchers quickly extract value from it. Most of this extracted value ends up with the validators. Order Flow Auctions (OFAs) let users capture the value they create.

OFAs auction off the right to execute an order to bidders and give users the winning bid amount. OFA aka Payment For Order Flow (PFOF) is an old practice in traditional finance (TradFi) dating back at least 40 years. In recent times, Robinhood popularised PFOF by selling users Orderflow (OF) and guaranteeing zero fees trading.

OFAs are poised to be one of the major narratives of 2023. In recent months, several companies have raised risk capital to build OFAs promising to help return MEV to users. For this article, we surveyed the solutions proposed by 12 independent teams to highlight their key design decisions and surface the tradeoffs they are making.

The Orderflow Auction Framework

All flavors of orderflow auctions we've surveyed has the same framework at its core. The difference between the solutions resides in how the four components are implemented. We describe the framework below.

Orderflow auction design decisions

In the next section, we explain each design decision in detail with the key concept ( ) needed to make that decision.

Order types

An OFA implementer needs to decide if it will support arbitrary order types or aim for an application-specific order type. While the generic route is attractive from a market size perspective, it will inevitably incur a loss of efficiency relative to an application-specific system. A specialized OFA designed for a narrow use case like swaps can optimize their system for user experience while targeting a large part of the market.

The orderflow auction promise is to return value which is currently leaked by users to validators. We can look at the distribution of validator revenues to reason about where this MEV comes from. As shown in the image below, we found that in the last 6 months, ~71% of all validator payments (78K eth vs 32k eth) are from transactions containing at least one swap (code).

Remember, the blockspace fee market on Ethereum is designed to resolve contention over state transition requests - like who gets to execute their trade first. The biggest customers of contentious blockspace are MEV searchers who use swaps to extract value by sandwich transactions, atomic back-runs or even CeFi-DeFi arbs.

0xAPI, Hashflow, DFlow, and CowSwap are examples of swap orderflow auctions. These services optimize for providing price improvement relative to interacting directly with on-chain liquidity by recapturing value leaked to validators.

Systems with less natural methods for compensating users must contend with returning value through reducing gas fees or providing cashback. The key insight is that these mechanisms are all equivalent in terms of economic value returned, but have different levels of efficiency and user experience. An example of this friction is the gas overhead of an additional payment or the user experience of receiving a non-native token.

The most generic value return mechanism would be a new transaction type which allows for negative gas price and gas payment in any token.

Information sharing

Once orderflow auctions have their orders, they must send information to bidders. Bidders then calculate how much value is extractable (EVEVEV) from the order based on this information. But before going forward let's learn how bidders estimate

the EVEVEV of a swap transaction.

Suppose the blockchain is in an initial state as described in the image below, i.e. there are 3 exchanges: Binance (CEX), Uniswap (DEX), and Sushiswap (DEX). At the initial state, the prices on all three exchanges are the same and price discovery happens on Binance. A swap transaction comes into the mempool which can change the price on Uniswap.

Extractable Value (EV): What value should the Bidder bid for this swap transaction? One way to estimate this is by measuring how much value can the bidder extract from this transaction. A swap transaction has two types of Extractable Value (EVEVEV):

- They can combine several swap transactions and predict the future direction of prices.
- They can sandwich the hedge performed by searcher A on Binance.
- They can use it to rebalance their inventory and save on uniswap trading fees.

We can loosely define an optimal price as the EVorderingEV_{ordering}EVordering + EVsignalEV_{signal}EVsignal of the best informed bidder with regards to an incoming order. An auctioneer wants all bidders to bid their true EVEVEV in order to perform efficient price discovery.

Optimum price:

An auctioneer may decide to limit the amount of value it exposes to the auction by limiting the amount of information that is revealed to the bidders. This leads to a direct loss of price optimality and can cause other adverse issues like bid spamming.

A bidder in a competitive single transaction auction with partial information on the incoming orders will attempt to approximate the missing information by either: 1) making a bid that is conditional on obfuscated data, or 2) spamming bids. Both of these approaches are prone to concerns regarding latency race leading to centralization and scalability issues due to excessive simulation resource consumption.

In the next section, we explore how auctions can attempt to efficiently capture information value.

Permissioning bidder list

The orderflow auction designer needs to decide how many and which bidders are allowed to participate in the auction.

Broadly, the options are the following:

- Open access: Barriers to entry for the ability to participate are as low as possible.
- Gated access: There is some gatekeeping on the ability to execute an order, either through a whitelist, a reputation system, a fee, or a seat auction.
- Exclusive access: A single party has the ability to execute an order.

The designer's first intuition is likely that more bidders provides better price discovery and that bidding on individual orders provides the lowest barriers to participation, thus leading to optimal price discovery.

As it turns out, it's not quite that simple.

Common value auctions (like auctioning a single swap opportunity) suffers from value leakage due to something called the winner's curse. The dominant strategy in common value auctions is to collude with other bidders to discount bids from the optimal price due to the high joint probability of a bid being higher than the true value. Common value auctions lead to bidder collusion.

Another source of value leakage is adverse selection from orderflow toxicity. Bidders want to make sure that originators and OFAs send ALL their orders via the auction, and not participate in the auctions themselves. For swap transactions, bidders will make a loss from toxic orders and make a profit from non-toxic traders. Since the originators have more information about pending orders relative to bidders, they can fill the good orders themselves thus leading to adverse selection for the bidders.

When selecting a gating mechanism for the bidder list, the orderflow auction runs into lots of issues. Open access leaks information value to the public, gated access runs into bid shading issues, and exclusive access runs into value attribution issues.

Lets look at two auction designs and compare their price optimality:

In the first auction - you have an open access auction. Everyone can see the content of the incoming orders. Unfortunately, this means the information value of an order is also leaked to counterparties meaning bidders will not include EVsignalEV_{signal}EVsignal in their bids.

In the second auction - you instead auction off the exclusive right to execute all orderflow for a period. This auction is a private value auction were the bids are based on predictions about the future. You therefore avoid leakage through winners curse. You also avoid leaking EVsignalEV_{signal}EVsignal as the bidder has exclusive access to the flow and must not worry about adversaries taking the information value for themselves. Unfortunately, this exclusive auction suffers from an

attribution problem. The auction efficiently prices the aggregate orderflow, but is unable to attribute the value to individual orders.

Bid selection function

Once all bidders have estimated the value (EVEVEV) of the order and placed their bids, the auction needs to decide how to select the winning bid - they need to define a bid selection function. This is made tricky due to the double auction problem.

Once bidders have submitted their bid value EVEVEV for an order, the auction systems needs to

EV=BidAmount+InclusionFeesEV = BidAmount + InclusionFeesEV=BidAmount+InclusionFees

The EVEVEV can be split into two parts BidAmountBidAmountBidAmount is the amount Bidders are willing to give to the OFA and InclusionFeesInclusionFeesInclusionFees is the amount Bidders spend to ensure reliable inclusion of their Order.

There are effectively 2 auctions happening:

If BidAmountBidAmountBidAmount is low then the bidder might lose the auction, whereas if InclusionFeesInclusionFeesInclusionFees is low the order might never be included On-Chain resulting in a bad UX.

There are 2 possible bid selection functions:

Adding complex ways to return value back to users will also result in increased InclusionFeesInclusionFeesInclusionFees reducing the overall user cashback. In our opinion, an OFA which can ensure maximum cashback to the user without deteriorated inclusion will win!

Execution guarantees

Once the OFAs decide on the winning bid, the order + winning bid is ready to be included on-chain. OFAs need to ensure reliable inclusion without increasing information leakage.

OFAs need to ensure that the order + winning bid gets reliably included on-chain. In this step, OFAs need to decide how many block-builders they should send the order+ winning bid. Once a winning bid is sent to the block-builder the following steps need to happen for the “order + winning bid” to be included.

Inclusion reliability is significantly degraded when certain actors have a “last look” ability to cancel a bid at the inclusion stage. This last look cancelation could be performed by a block builder who is also a bidder in the auction if they notice that their bid is no longer profitable due to price movement on a CEX.

The decision of who carries non-execution risk is often locked in place due to other design decisions. If the bid selection function is Max inclusion fees, the user carries the risk. If the OFA is building a system using signed messages then there is reduced blockspace contention i.e. increased inclusion reliability, and the non-execution risk can be put on the bidder. If the order + winning bid is sent via bundles then OFAs are handcuffed and need to use trusted builders, potentially leading to builder centralization.

An auction which sends the order + winning bid to several builders (trustless setup) will leak informational value to multiple builders. Whereas, if the auction sends the order + winning bid to trusted block-builders (to reduce info leak) they cannot ensure reliable inclusion. The most trivial implementation involves OFAs giving the responsibility of order inclusion to the winning bidders. If the winning bidder cannot include the transaction in the given time, they pay some penalty and the order is released by the auction to the public mempool.

The Four Orderflow Auctions

In the previous section, we looked at a core framework for thinking about all orderflow auctions and their key design decisions. In this section, we discuss 4 example OFA implementations which we believe have inherent efficiencies and product market fit. The 4 auctions have the potential to compose with each other if built right.

RFQ price auction

An RFQ price auction is a price discovery system for swaps. It uses signed messages combined with an application-specific contract code to execute swaps. Gated market makers provide liquidity in RFQ auctions and the best price wins the bid. Note, there is no on-chain price for RFQ auctions which reduces EVorderingEV_{ordering}EVordering and bidders mostly extract value via EVsignalEV_{signal}EVsignal.

In TradFi RFQ systems are used to place big-volume orders which need liquidity to be sourced and market makers bid based on how much liquidity they can access. More access to liquidity ensures lesser slippage and more price improvement for the user.

We believe swap specialized OFAs (exclusive batch auctions & RFQ price auctions) will be more efficient in returning swap

value to originators and will dominate their category for the following reasons:

Exclusive batch auctions

An exclusive batch auction holds an auction for the exclusive rights to execute all the order flow in a time window to a single winner. Since bidders cannot see the orders, they place the bid based on the predicted market volatility and their average execution quality. Exclusive batch auctions depend on a reference price in order to assure good user prices and therefore can't be used for price discovery. One weakness of this approach is that batch auctions is the difficulty of attributing the proceeds of the auction to individual orders.

Sending all the order flow to a single bidder eliminates information leakage and the winner's curse. Due to the winner's curse a live auction OFA has never been successful in TradFi yet.

Transaction execution service

A transaction execution service shares order information with all bidders. Bidders primarily extract EVorderingEV_{ordering}EVordering and the OFA decides the winning bidder based on maximum user cashback. The seeks to "internalize" the MEV generated by the transaction and return it to users. This type of system carries non-execution risk and needs extra mechanisms to ensure the reliable inclusion of orders in this design.

70% of surveyed OFA implementations adhere to this design. This design works well with all transaction types and is the easiest to integrate for wallets and (bidders) searchers.

One of the main value add of transaction execution service is providing fee estimation - that is, having some sophisticated model to determine what price users should pay in order to win a contested state transition.

Blockspace aggregator

A blockspace aggregator enables users to express granular conditions on the state transition they are requesting, including the cashback or fee they are willing to pay. A blockspace aggregator does not help a user with price discovery, but rather provides it with a channel for reliable inclusion of a complex transaction. The block-builders receive all the bids and picks which to include based on the highest InclusionFeesInclusionFeesInclusionFees. Since preference expression is not inherently supported by Ethereum, implementation involves a new transaction type or a centralized service.

This design promises a high inclusion rate since it is maximizing for the MEV captured by validators, at the cost of builder centralization as it relies on trusted block builders for execution and is incompatible with FIFO ordering systems which are popular on L2s.

Conclusion

Orderflow auctions (OFAs) promise to give users cashback for the value they create via their orders. We believe this will be the year of OFAs and expect significant competition happening between different OFA designs and their implementations in the next 6 months. All these solutions will be based on the same core framework. The resulting MEV supply chain will end up looking like the image below.

Swap transaction will be routed through a combination of RFQs for price discovery and PFOF style auction for execution. Regular transactions will be routed to a combination of execution services which will aim to minimize the cost to users, and to blockspace aggregators who will maximize the amount of extracted MEV before sending a bundle to a builder for inclusion.

We believe implementations will cluster around these four distinct orderflow auction designs as they each serve independent needs and benefit from efficiencies from existing in different corners of the tradeoff matrix.

Acknowledgments

Thank you to the following teams for their contribution to open research on orderflow auctions: Blink, Bloxroute, Cowswap, Dflow, Flashbots, Kolibrio, Merkle, Nectar, Rook, Wallchain

Thank you to Simon Brown from Consensys Research and Alex Nezlobin from London School of Economics.

References