[

Powerful Intents: Part 2

1080×608 4.6 KB

](https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/f6b8fd2e3c322b755617c38aecab0a2d5b3bea86.png)

This is the second part of Brink's three part "Powerful Intents" series. We recommend readingPowerful Intents: Part 1 first.

Solver Incentives

Solvers are critical to all intent-centric systems. They handle the technical details of transaction issuance, navigate the complexities of swap routing and other state modifying solutions, and ultimately ensure that user intents are settled quickly and efficiently. However, solvers are not benevolent entities. In a purely decentralized and trustless environment, solvers are in fierce competition with each other. They're incentivized only by the MEV profit they can extract from each intent.

In order to protect users against malicious actors, intent-centric protocols only allow solvers to operate within the bounds of what a user signs and approves. Ideally, an intent-centric protocol aligns the goals of users with the incentives of solvers. A protocol with well designed solver incentives has network effects at scale. In other words, user outcomes improve as solver competition increases

.

In this post we will explore the following

- Mechanisms for Explicit and Implicit Incentives
- Dynamics of Solver Competition
- Reducing costs by Forcing Solver Coordination
- The concept of Fee Escalators
- Incentives for Conditional Intents
- Outcome optimization trade-off of Speed vs. Price
- Incentives around Solver Reputation
- The need for Protocol Standardization

Explicit vs. Implicit Incentives

In order for solvers to settle an intent, they need to be able to cover their costs and potentially earn profit. Solver revenue could be explicitly paid by the intent signer, or implicitly extracted by the solver from the assets within the intent. Intents that don't involve swaps need to incentivize solvers with an explicit payment or fee. Intents that involve swaps can incentivize solvers implicitly with a spread-based profit. Spread comes from the difference between the user's signed intent price and the price that a solver is able to source.

Let's look at an intent to transfer USDC as an example. If a user creates an intent to transfer 1,000 USDC from wallet A to wallet B, there is no implicit incentive for a solver to execute the transfer on behalf of the user. If the user includes a payment of 50 USDC to the solver, there is an explicit incentive to settle the transfer. The 50 USDC payment would ideally be equivalent to the solver's cost of execution plus some additional profit.

Intents to swap don't require explicit incentives, because there are implicit incentives for solvers to fill swaps on behalf of users. Solvers can cover costs and make profit by taking advantage of the spread between the user's signed intent price and the best available market price.

Consider a user who signs an intent for a limit swap of 1 ETH to 1,600 USDC. If the best ETH to USDC route a solver can find is exactly 1 ETH to 1,600 USDC, then the solver has no incentive to execute the intent. However, if the ETH/USDC market price increases and the solver finds a route for 1 ETH to 1,700, the solver can pay the user 1,600 USDC and take the extra 100 USDC for themselves. If 100 USDC is enough to cover the solver's gas cost for the transaction and provide the solver with sufficient profit, then it makes sense for the solver to execute.

[

1673×1315 179 KB

](https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/dbdbd27e9c2535dbdf2aab77085e7b87ee7f5170.png)

You might ask why the solver doesn't wait until the ETH/USDC market price increases even further, in order to take more than 100 USDC. The answer is, if a solver decides to hold out and wait for greater profit they risk losing the spread to another solver.

Solver Competition

Solving is generally a winner-takes-all game, where only one solver will successfully settle a given intent. This winning solver takes all of the profit associated with the intent. The first solver to settle the intent will succeed, and all subsequent attempts will fail.

When solvers openly compete with each other, they are incentivized to provide the best routes they can find and to execute them as fast as possible. As new solvers enter, they compete for smaller profit margins, user fees move toward zero, and the market for intent based MEV moves toward perfect competition. This is an example of user and solver incentive alignment, where competition among untrusted solvers results in better pricing and faster settlement for users.

In a fully centralized system, a single permissioned solver could choose to take advantage of their position in order to extract more profit. A solver with no competition does not have to factor in any risk of failed transaction cost or potential for front-running. In a completely open and permissionless system where profit margins are pushed toward zero, users will benefit from more efficient route-finding, but they also have to indirectly pay for any costs associated with collision and front-running risk that solvers need to account for.

Since all solver costs are ultimately passed on to the user, intent-centric protocols should ideally help solvers reduce their costs. Two of the most costly risks for solvers are transaction collisions

and front-running.

Intent-centric protocols can mitigate both of these risks by forcing solver coordination.

Transaction Collisions

If any EOA can participate as a solver at any time, and there is no protocol mechanism in place to force solvers to coordinate, this results in competition for block inclusion and for favorable transaction ordering. When competition is high, solvers will inevitably submit transactions that collide. Collisions happen when more than one solver transaction is included in the same block.

Since solvers are competing for MEV, a solver might share some of that MEV with a block builder to make sure their solution is included in the block first. A payment to the block builder eats into a solver's profit margin making it more costly for them to execute the solution, but is necessary to avoid a revert. A reverting solution will always result in a loss for the solver. For every intent that a solver solves, a cost-benefit analysis needs to be done to compare block inclusion cost vs. the risk of paying for a reverted transaction. Solver costs associated with mitigation of revert risk are indirectly passed on to users signing intents.

Intent Front-Running

Front-running in the dark forest of permissionless mempools has been around since the early days of Ethereum. Competition for MEV is constantly increasing as more value flows through EVM transactions, leading to the rise of generalized front-running bots that monitor public mempools for any transactions involving permissionless transfer of funds.

These bots can easily replace a profit-taking address with their own and ensure that their transaction is included first by incentivizing block builders. Advanced generalized front-running bots have even been involved in multimillion dollar exploits.

Front-running is a huge risk to solvers. Any transaction that a solver submits to a public mempool runs the risk of being front-run. Generalized front-running bots are extremely advanced and have taken MEV from intent solvers. An advanced front-running bot doesn't need to do any solving, and doesn't even need to know about the specific intent or protocol that the solver is interacting with. In other words, solvers do the hard work of finding an efficient route, and generalized front-running bots copy the route and execute it first to take profit from the solver.

If the intents protocol and the signer of the intent don't include any built in front-running protection mechanisms, then the solver has to mitigate front-running themselves. They really only have two options, and each option has serious drawbacks:

1. Submit their solution to a private mempool:

Front-runners won't be able to see the solution transaction, but this will result in slower execution and potential loss of the solution opportunity due to market movement.

1. Pay block builders for more favorable ordering:

Front-runners can still see the solution transaction and will do the same until ultimately the solver's profits are driven close to zero.

Ideally, intents protocols are set up to help mitigate these risks for solvers by implementing mechanisms that force solvers to coordinate.

Forcing Solver Coordination

If the potential for solver collisions and front-running are eliminated, solvers don't need to worry about revert risk or block ordering cost. If these costs are lower for solvers, that will result in better pricing for users.

Protocols have taken a few different approaches to force solver coordination, such as round-robin

solver selection, whitelists

, and Order Flow Auctions (OFA's)

.

Round-robin Selection

In round-robin selection, solvers register and are randomly given exclusive permission to solve intents. Although this prevents front-running and collisions, it's problematic because solvers can extract additional value from an intent when they are given exclusive permissions to solve that intent. This mechanism is good at coordinating solvers, but removes the need for solvers to compete with each other to provide better prices.

Solver Whitelists

Restricting intent permissions to a whitelist of solvers is a very common mechanism used by intent-centric protocols. This helps mitigate the risk of generalized front-running while still maintaining some level of solver competition. However, it relies on solvers within the whitelist to be trusted. At scale with a very large whitelist of anonymous solvers, one of those solvers will inevitably start running their own generalized front-runner. Whitelists also require governance, which usually adds some degree of centralization.

OFA's

Recently designed Order Flow Auction (OFA) mechanisms help to mitigate these risks by moving solver competition off-chain. With OFA's, solvers are required to bid on intents before they are allowed to solve them. The solver with the best solution wins the right to settle the intent and gets to extract the associated MEV. Protocols such as CoW Swap and UniswapX use OFA's to select solvers who provide the best price routes. Because the selected solver has this temporary permission to settle on-chain, it is impossible for generalized bots to front-run their solution, and removes the need for the

solver to compete for transaction ordering.

In general, any system that forces solver coordination does so by granting some limited permissions to solvers. Protocols that use OFA's force solver coordination by moving competition between solvers off-chain, and pre-determining which solver is allowed to settle on-chain.

In the event that a permissioned solver does not follow through with their solution, a fallback mechanism needs to be in place. In the case of UniswapX, a dutch auction for the intent will start, where the price that the user is willing to accept for the swap will decrease every block. Decreasing price means increasing spread-based fees for solvers. When fees increase, eventually they will be high enough for a solver to settle the intent.

Fee Escalators

The concept of increasing fees over time until a solver is sufficiently incentivized to settle can generally be described as a fee escalator (originally defined in this post). Fee escalators allow for price discovery and can provide a suitable level of guaranteed settlement for intents.

In the case of UniswapX's fee escalator, swap prices decrease every block which causes solver incentives (in the form of spread fees) to increase every block. This escalating fee mechanism has the benefit of making settlement time more predictable, and ensuring that the swap doesn't get stuck in an un-fillable state where solvers won't settle because the swap MEV can't cover their costs.

The UniswapX dutch auction fee escalator approximates the behavior of amarket order, but technically it is a dynamically priced limit order. For a user who wants their swap to settle quickly for a price that is as close to market price as possible, this mechanism makes sense. The user in this case has an expectation that the swap will be settled within a few blocks. A user who is more concerned about price and less concerned about timeliness or guaranteed settlement might opt not to use UniswapX. They could instead sign a fixed price limit swap intent and hope that market conditions become favorable enough for solvers to settle the intent at some point in the future.

Consider a fixed price limit swap intent signed for exactly 1 ETH input to 1,800 DAI output. With this intent, the amount of MEV a solver can extract changes based on market price and gas price fluctuations relative to the swap asset ratio. In other words, solver incentives change only when market conditions change. Since market conditions are not predictable, the speed at which the limit swap will be settled is also not predictable. There is also no guarantee of settlement: it is entirely possible that any given limit swap signed at a price more favorable than current market price will never settle, because the market could move in an unfavorable direction forever.

A class of intents known as Conditional Intents

(read more in Part 1) have no guarantee for settlement because the arbitrary state conditions they require may or may not ever occur. Fixed price limit swaps are a very simple type of conditional intent, where the condition required is an increase in the value of the input asset relative to the output asset. While the incentives for solver settlement of limit swaps are straightforward, other types of conditional intents can have more complicated solver incentives and trade-offs for users.

Conditional Intent Incentives

When a signed intent includes a swap that requires a state condition, the timeframe for settlement of the intent is not predictable. This presents some incentive challenges that don't exist for intents that are expected to settle immediately.

Consider a continuous intent to swap 1 WETH for USDC every 300 blocks at market price. A user signs an intent so that the first swap cannot be settled until 300 blocks from now, as long as the price is above 1,700 WETH/USDC. The required state conditions here are that blockNumber must be currentBlock + 300 or greater, and that the price of WETH/USDC is greater than 1,700. No one knows what the WETH/USDC price will be or what a solver's costs will be 300 blocks from now. Variability in solver costs usually comes from the fluctuation in gas prices. It's feasible that settlement cost could drastically increase or decrease within 300 blocks due to a volatile gas market.

For this type of intent it would be difficult to implement a system where users accept the best solver bids, because the user is pre-signing the intent 300 blocks before settlement is expected to happen. An on-chain bidding system could be implemented, but this would be gas-intensive and add more cost for settlement: a cost that ultimately gets passed on to the user.

A better alternative in this case would be to use a fee escalator in order to provide a predictable timeframe for settlement after each 300 block interval has passed. The intent would need some starting point for the price, which could come from a reliable and difficult to manipulate on-chain pricing source (like a Uniswap TWAP Oracle). The price could decrease from this starting point until a solver is sufficiently incentivized to settle the swap. This is a “dutch auction” fee escalator, where the starting price is based dynamically on state (TWAP and blockNumber).

[

2396×1480 170 KB

](https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/c61ab4929b0d1f3e0bc03d63493bd24c7bbeb8b5.jpeg)

Speed vs. Price Trade-off

Intent outcome optimization usually involves some trade-off between speed and price, and users should be able to choose which they want to optimize for. A user optimizing for price might choose to sign a fixed price limit swap and wait for settlement, whereas a user optimizing for speed might choose to use a system like UniswapX to get close to immediate and guaranteed settlement for whatever price the current market is offering.

Users will generally want to optimize somewhere in the middle of the price vs. speed curve. Consider these two scenarios:

- Speed over price:

A user signs an intent for a stop-loss on a potentially rapidly declining market. Optimizing for speed of settlement over price makes sense here because slow settlement could actually result in a significantly worse price. This optimization could be accomplished by signing a more aggressive fee escalation curve as part of the intent, meaning a greater fee incentive increase for solvers in each block.

- Price over speed:

A user holding a long ETH position signs an intent to average out to USDC over the course of months. Optimizing for price over speed makes sense here because this intent involves a longer timeframe and less urgency to settle. A fee escalator here might not be necessary, if the user wants to be extremely price sensitive and does not prioritize guaranteed settlement.

The same trade-off exists for intents that don’t involve a swap, such as transfer, bridging, or deposit/withdraw. Offering higher fees to solvers results in faster execution, but costs more for users. Different scenarios require different priorities for signers of intents, and users should be able to make this choice.

Reputation Based Incentives

Protocols that rely on solver reputation are potentially dangerous for a few reasons. Relying no reputation implies that solvers will be rewarded for good behavior and punished for bad behavior. They will receive more exclusive order flow if they have a better reputation. This presents significant risk of centralization of OFA’s and intents. If any permissionless protocol relies on reputation alone, and solvers are psuedo-anonymous entities, the amount of profit they could take from exploiting the system needs to constantly outweigh the cost of damaging their reputation as a solver. In a permissionless system, there is never any guarantee that this is the case because there is no limit to the amount of value allowed to flow through the protocol.

A protocol that uses formal reputation scoring for solvers and allows/disallows solvers based on DAO governance would only be as efficient as the governance process allows. It could be detrimental to allow a bad-acting solver to continue to

operate even for a few blocks. Governance processes generally operate on a longer timescale and would not respond quickly enough.

It's possible that informal reputation for solvers will evolve in the form of users limiting their intents to specific solvers or sets of solvers. This could be done with or without the user having a full understanding of what permissions they are signing. Applications could require that users provide permissions to a set of application specific solvers, in order to gain some exclusive MEV from the order flow. As with anything in Web3, the risks associated with exclusivity and centralization have to be balanced against the complexities of building a decentralized and trustless system.

Protocol Standardization

The solver ecosystem today is quite new and is rapidly evolving. There are only a handful of intent-centric apps for solvers to integrate with. Each of these apps deals with market/limit swaps only and each has built their own unique app-specific protocol. Solvers that have integrated with these apps/protocols are already faced with very high competition for small profit margins. Protocols often add new functionality, and solvers have to spend a lot of time and effort keeping up in order to remain competitive.

In the future, thousands of apps will provide a web2-like user experience through account abstraction and intents. It's unlikely that each app will build a new intent protocol or that solvers will integrate with thousands of app-specific protocols separately. Having an open standard for intents that solvers can integrate with will allow new apps to tap into an existing solver network rather than having to build their own. EIP-7521 is an emerging standard for generalized intents using smart wallets that aims to accomplish this. Brink is building a protocol for composable intents that allows for the creation of advanced intent workflows and order flow auctions that automatically plug into an existing solver network.

Solvers that integrate with standardized protocols will get access to future order flow from many apps without having to build new integrations for each individual app.

Conclusion

The intents space is growing rapidly and solver incentives are one of the most critical pieces of the puzzle. There is a very clear network effect in intent-centric systems: more solver competition = better user outcomes

. Unlike smart contract code, solver networks are extremely hard to fork. Solvers themselves are closed source, privately operating bots, created by some of the most skilled devs in crypto. They operate in a fiercely competitive dark forest.

The intents layer harnesses the power of the dark forest for the benefit of users. As solver networks scale up, the need for users to interact directly with blockchain networks will eventually disappear completely.

At Brink we are building toward a future where all Web3 apps will run on user signed intents settled by solvers.

Resources

Powerful Intents: Part 1

Ethereum is a Dark Forest

Order Flow, Auctions and Centralisation

Thread on Solver Front-Running

Our favorite generalized front-runner: c0ffeebabe.eth

Acknowledgements

Thanks to Connor Howe @ Enso and Stephen Monn @ Essential for your help and feedback on this post.