Introduction

Recently, discussion around “intents” and their application have been a dominant topic in the Ethereum community.

If a transaction explicitly refers to “how” an action should be performed, an intent refers to “what” the desired outcome of that action should be. If a transaction says “do A then B, pay exactly C to get X back”

, an intent says “I want X and I’m willing to pay up to C”

.

This declarative paradigm unlocks exciting improvements in UX and efficiency. With intents, users are able to simply express a desired outcome while outsourcing the task of best achieving that outcome to sophisticated third parties. The notion of intents contrasts with today’s imperative paradigm of transacting where every parameter is explicitly specified by the user.

While the promise of these improvements presents a much-needed step for the ecosystem, intent-based designs on Ethereum can also have significant ramifications for off-chain infrastructure. In particular, there are important connections to MEV-related activities and market control. This post seeks to provide a brief definition of intents and their benefits, an exploration of the risks involved with their implementation and some discussion of potential mitigations.

What Are Intents?

The current standard method through which users interact with Ethereum is to craft and sign transactions

, messages in a specific format that provide all of the necessary information for the Ethereum Virtual Machine (EVM) to execute a state transition. However, creating transactions can be a complicated affair. Creating transactions requires reasoning about a vast web of smart contracts and details like nonce management while holding a specific asset to pay gas fees. This complexity leads to suboptimal UX and lost efficiency due to users being forced to make decisions without sufficient access to information or sophisticated execution strategies.

Intents arose to alleviate the user of these burdens. Informally, an intent is signed a set of declarative constraints which allow a user to outsource transaction creation to a third party without relinquishing full control to the transacting party.

In the standard transaction-based flow, a transaction signature permits the validator to follow exactly one computational path against a certain state while a tip incentivises the validator to do so. On the other hand, an intent does not specify exactly what computational path must be taken, but rather allows for any which satisfy certain constraints. By signing and sharing an intent, a user is effectively granting permission to recipients to choose a computational path on their behalf

(see figure below). This distinction allows a slightly more rigorous definition of intents as signed messages which allow for a set of state transitions from a given starting state, a special case of which is a transaction which allows for a unique transition

. That being said, we will continue to refer to “intents” as distinct from transactions.

Importantly, many intents can be included in a single transaction, allowing for matching overlapping intents, increasing gas and economic efficiency, e.g. in a builder-maintained orderbook where two orders can be netted against each other before going to a market. Other applications include cross-domain intents - signing one message instead of multiple transactions on different domains - using different replay resistance schemes, and more flexible user gas payments like allowing 3rd parties to sponsor gas or for payments in different tokens.

The Past and Future is Intents

Intents have been created for the outsourcing of the complexities of interacting with the blockchain while permitting users to maintain custody of their assets and cryptographic identity.

You may notice that many of these ideas correspond to systems that have been in operation for several years:

1. Limit Orders

: 100 X may be taken from my account if I receive at least 200 Y.

1. CowSwap-style Auctions

: same as above, but rely on a third-party or mechanism to match many orders to maximise execution quality.

1. Gas Sponsorship

: Pay gas in USDC instead of ETH. The intent can only be fulfilled with a matching intent which pays ETH in fees.

1. Delegation

: Only allow interacting with certain accounts in certain pre-authorized ways. The intent can only be fulfilled if the final

transaction respects the access control list specified in the intent.

1. Transaction Batching

: Allow batching of intents for gas efficiencies.

1. Aggregators

: Only use “best” price/yield for an action. The intent can be fulfilled by showing a proof that an aggregation over multiple venues was executed and the optimal path was taken.

Looking forward, intents are seeing renewed excitement in the context of cross-chain MEV (e.g. SUAVE), ERC4337-style account abstraction, or even Seaport Orders! While ERC4337 is moving full steam ahead, other novel applications like cross-domain intents still require further research. Further discussion of intents and their applications can be found in this talk.

Critically, across all old and new intent-based applications there needs to be at least one other party who is aware of the intent, incentivised to execute the intent and able to do so in a timely manner

. The questions of who

these parties are, how

execution takes place and what their incentives

are, must be asked to determine the efficacy, trust assumptions and broader-impact of an intent-driven system.

The Middlemen & Their Mempools

The most obvious channel for an intent to find its way into the hands of a willing middleman is the Ethereum mempool. Unfortunately, the current design does not support the propagation of intents. Concerns around DoS attacks may mean that generic support for fully general intents in the Ethereum mempool is out of the question even in the long term. As we will see below, the open and permissionless nature of the Ethereum mempool poses additional obstacles to adoption for intents.

In the absence of the Ethereum mempool, intent system designers are now faced with some design questions. One high-level decision is whether intents will be propagated to a permissioned set or be available in a permissionless manner so that any party can execute the intent.

Permissionless Mempools

One design one might strive for is a decentralised API that allows for the gossipping of intents across various nodes in a system, providing permissionless access to executors. This has been done before. For example, in 0x protocol relayers gossip limit orders among each other and put them on-chain when there is a match. The idea is also being explored in the context of a shared ERC4337 mempool to combat centralization and censorship risk. However, the design of such permissionless “intentpools” faces some significant challenges:

- DoS resistance

: one might have to limit the functionality of intents to avoid attack vectors (see the ERC4337 proposal for further discussion)

- Propagation incentives

: for many applications, executing an intent is a profitable activity. Thus, nodes operating the intentpool have an incentive not to propagate, to reduce competition in executing the intent.

- MEV

: intents, which rely on good behaviour of off-chain actors for execution quality, may face difficulty using a public, permissionless intentpool. If poor execution is profitable, permissionless intentpools are likely to lead to this outcome. This is similar to sandwiching in the Ethereum mempool today and is anticipated to be a widespread problem for DeFi-related intents. A possible path forward here might be permissionless, but encrypted intentpools.

Permissioned “Mempools”

A trusted centralised API is much more DoS resistant and does not need to propagate intents. The trusted model also provides some foothold against MEV concerns. As long as the trust assumptions hold, execution quality should be as guaranteed. Trusted middlemen may also have reputations associated with them, providing some incentive to provide good execution. Because of this, permissioned intentpools are attractive to intent-based application developers in the short term. However, as we are all well aware, strong trust assumptions have shortcomings and are somewhat antithetical to much of the blockchain ethos. These issues will be touched on below.

Hybrid Solutions

There are solutions which are mixtures of the above. For example, one could have permissioned propagation, but permissionless execution (assuming trust assumptions hold) or vice versa. A common example of a hybrid solution is order flow auctions.

The high-level idea of these designs is that users who need a counterparty may need to differentiate between better and worse counterparties (e.g. to take the other side of a trade at a favourable price). The design flow usually includes a trusted party who receives the intents (or transactions) from the user and facilitates the auction on their behalf. Participation in the auction is (sometimes) permissionless.

These kinds of designs have their own shortcomings and are likely subject to many of the concerns around permissioned intentpools, but there are some important differences that will become apparent later.

Bottom line: Intent-based applications involve more than just a new message format for interacting with smart contracts, they also involve propagation and counterparty discovery mechanisms in the form of alternative mempools. It is not trivial to design a mechanism for intent discovery and matching which is incentive-compatible and not centralizing at the same time.

What Can Go Wrong?

While intents are an exciting new paradigm for transacting, their widespread adoption may imply an acceleration of a larger trend of user activity shifting to alternative mempools. If improperly managed, this shift risks centralisation and entrenchment of rent-seeking middlemen.

Order Flow

If intent execution is permissioned and the permissioned set is not chosen with care, the migration out of the public mempool threatens to centralise block production on Ethereum.

The vast majority of block production on Ethereum currently happens via MEV-Boost, an out-of-protocol implementation of proposer-builder separation (PBS) and current roadmaps show no indication that this interface will change any time soon. PBS relies on the presence of a competitive market of block builders to channel MEV to the validator set. One major concern in PBS is that a block builder is able to acquire exclusive access to the raw materials required to produce valuable blocks - transactions and intents, AKA “order flow”. In the language of PBS, permissioned access to intents would be known as “exclusive order flow” (EOF). As discussed in this article, EOF in the hands of the wrong party threatens the market structure upon which PBS hinges as the exclusivity of order flow implies a moat against competitive forces.

A block builder (or a partnered entity) with control over a large share of Ethereum’s order flow would be in a position to produce the majority of mainnet blocks, opening a vector for censorship. As the network relies on competition between builders to channel value to validators (or to be burned in future), the dominance of a single builder would constitute a shift of value from Ethereum to the builder. Rent-seeking and censorship are certainly important threats to the protocol.

Trust

As many solutions require trust in intermediaries, development of new intent-based architectures is hampered by high barriers to entry, implying lower rates of innovation and competition to ensure execution quality.

In the worst case, a user finds themself in the position in which there is a single party executing intents, such as the monopolist block builder from the section above. In such a world, the monopolist block builder would be able to extract rents

and any new proposal for how intents should be processed would be denied traction

if not adopted by the builder. In the face of a monopolist, individual users lose negotiating power - an effect which is exacerbated when users use intents to give additional degrees of freedom to middlemen.

Unfortunately, market stagnation due to centralised infrastructure is not contained to concerns over the builder market. Even for non-block building operations, high barriers to entry can place middlemen in a powerful position as they face little competition. Consider for example, the state of the current order flow auction market. Several entities like Flashbots and CoWswap receive the majority of the order flow destined for OFAs. The distribution of order flow is in no small part due to the fact that these entities have existed for years or are associated with reputable entities, meaning they have successfully captured a degree of public trust. If a new OFA design were to attempt to enter the market, whoever was operating the new OFA would have to spend a lot of time convincing users and wallets that they are reputable and would not abuse their position. The necessity of such a campaign to win trust certainly constitutes a substantial barrier to entry.

The order flow auction market has only recently begun to pick up attention and it remains to see how competition will develop, but the market does provide an illustrative example of a setting in which permissioned, trusted mempools could enshrine a handful of powerful actors, undermining users’ best interests.

The EIP4337 intent format provides another example of where we are at risk of enshrining a certain kind of mechanism. Consider a world in which trusted architecture has been put in place to support 4337 intents. If another intent format - perhaps serving additional use cases like cross-domain functionality - is proposed, but the established trusted middlemen do not adopt this new format (after all, it doesn't have much adoption and competes with their business model), the implementation of the new format would require establishing trust in new entities. Again, we find ourselves in the position where innovating and challenging the status quo is met with trust-based barriers to entry.

Opacity

As many intent architectures entail the user surrendering some control over their on-chain assets and permissioned mempools imply a degree of impenetrability from the outside, we risk building an opaque system in which it is unclear how or whether users' expectations are met and threats to the ecosystem remain undetected.

The sections above pertain to the risk that power imbalances in the order flow market pose to users and the protocol. A related concern is that the ecosystem of middlewares and mempools that is developing between the user and the blockchain becomes opaque even to astute observers. This concern is particularly pertinent for intent-based applications which seek to allow the user to outsource important decisions like order routing.

The occasions on which MEV negatively impacts user execution generally arise due to high degrees of freedom that transactions give up to their executors (e.g. slippage limits). It is thus no great leap in logic to assert that intent-based applications which surrender greater degrees of freedom should design their systems for execution with greater caution. The worst-case outcome in this regard is a world in which using an intent-based application entails signing an intent that disappears (into the dark forest, if you will) and then somehow materialises as a transaction(s) with no clarity on how or by whom the transaction was created. Of course, the ability to monitor such an ecosystem relates to concerns around EOF and trust-based entrenchment as well. How is the Ethereum community supposed to monitor for threats to the health of its block production ecosystem if this ecosystem is obscure even to the most astute observers?

Mitigating Risks

The Ethereum mempool is limited. For some applications, this is due to its lack of privacy (sandwiching) and for others, it is due to its inability to support broader message formats. This leaves wallets and application developers in a difficult position as they must find some way to connect users to the blockchain while avoiding the dangers highlighted above.

In examining the issues above, we can extrapolate some properties of an ideal system. Such a system should be permissionless

so that anyone can match and execute intents while not trading off much execution quality

; general

so that deploying new applications doesn't require standing up new mempools; and transparent

so that the process by which intents are executed is reported publicly and data for execution quality auditing is made available, when privacy guarantees allow.

While teams like Flashbots and Anoma are working towards general-purpose solutions that satisfy the requirements above by marrying privacy and permissionlessness, the ideal system will likely not be ready in the near term. As such, different solutions making their own tradeoffs may serve different applications best. Although mechanisms like crlists that came as a response to many of the same concerns around transaction-based applications may not be available for intents, small tools, like allowing users to fall back to transactions where possible, may do well to improve worst-case scenarios. In a similar vein, applications looking to launch intentpools, would do well to seek generality if permissionless and select middlemen cautiously if permissioned.

Speaking broadly, we ask that intent-based application designers consider thoroughly the off-chain implications of their applications as these may touch the broader community, not just their user base, and we ask that the broader community maintains a watchful eye over the developments of the off-chain ecosystem surrounding Ethereum.

Conclusion

The adoption of intents represents a shift from an imperative to a declarative paradigm, which promises to improve UX and efficiency loss due to MEV leakage significantly. The demand for these applications is clear and many intent-based applications have been used widely for several years.

A growing adoption of intents, partly driven by ERC4337, is likely accelerating a move from the Ethereum mempool to new venues. Whereas this move is justified and inevitable, intent-based application designers have strong reason to be cautious in designing the off-chain components of their systems while robust infrastructure is in development.

There is still plenty of research and engineering to do in this nascent transacting paradigm and areas we didn't cover in this post such as designing an intent-expression language that allows for privacy. If you find this or other intent-related research subjects enticing, please reach out to 0xquintus georgios@paradigm.xyz.

Many thanks to Dan Robinson, Charlie Noyes, Matt Huang, John Guibas, Xinyuan Sun and Elijah Fox for their feedback on this article and Achal Srinivasan for designing the figures.