A New Era of DeFi with App-Specific Sequencing

Special thanks to @0xtaetaehoho, @ThogardPvP, @dex_chen_V, @Jskybowen, @visavishesh, @TrustlessState, @RyanSAdams, @FrankieIsLost, @0xnagu for the feedback, and credit to @tarunchitra and @matheusVXF for being years ahead of the game once again.

Introduction

Addressing MEV (Maximal Extractable Value) has been an ongoing challenge for Ethereum; the value supply chain incentivizes constant activity from arbitrageurs with diverse strategies of varying levels of sophistication, often at the expense of retail users. While many researchers have tried to address MEV through protocol-level changes, these efforts have not yet provided a satisfactory solution. The canonical infrastructure and auction mechanisms currently in use are able to competitively capture the lump-sum MEV in a block, but capture without fair redistribution is inadequate: why should MEV value accrue to the network validators when it can more effectively be captured and internalized on an application-by-application basis?

Enter Application-Specific Sequencing (ASS). Rather than attempting to rewrite the rules at the protocol level, ASS gives individual apps the power to take control of how their transactions are sequenced. By doing so, ASS allows onchain applications to protect their users and liquidity from the harmful effects of MEV while also giving them the opportunity to capture value that would otherwise be lost to Ethereum validators.

Imagine the potential: instead of high-frequency traders competing to maximally arbitrage each user (with nearly all of the arbitraged value leaking to the validators and hence underlying chains), each app could define its own rules for transaction ordering, creating a more tailored, efficient, and fair system for its own users. This marks a shift from trying to solve MEV at the network level to addressing it where it matters most—the application itself.

Background

The concept behind Application-Specific Sequencing (ASS) originated from Matheus's work on Verifiable Sequencing Rules (VSR) for decentralized exchanges (DEXes), which demonstrated that VSRs could improve trade execution and mitigate MEV by reducing miners' influence over transaction ordering. Tarun later expanded on this idea by showing how app-specific sequencing rules could significantly affect the payoff functions for protocol participants, such as users, validators, and sequencers.

Here, the payoff function represents the economic value of a particular transaction ordering. This value reflects the profit or utility gained by protocol participants, showing how transaction ordering impacts their financial outcomes. There are two critical characteristics of the payoff functions:

1. Non-smooth payoffs: Small changes in ordering can cause big changes in MEV.
2. Non-monotone payoffs: Small changes in ordering can either increase or decrease MEV, but not consistently in one direction.

When payoff functions exhibit both of these characteristics, optimizing the sequencing strategy becomes highly complex. In such cases, more sophisticated and bespoke approaches are necessary, at the application level, to ensure equitable outcomes for users and a sustainable DeFi ecosystem.

How does ASS work?

To understand ASS, let's first review the existing transaction supply chain.

In the current system:

1. Transactions are sent to public or private mempools.
2. Builders collect these transactions and package them into blocks.
3. Builders then compete in a block auction.
4. The winning builder's block is included in the blockchain and the value they bid is paid to the chosen proposer for the given block.

The figure below illustrates this process, showing how transactions flow from mempools to the blockchain via builders and trusted relays.

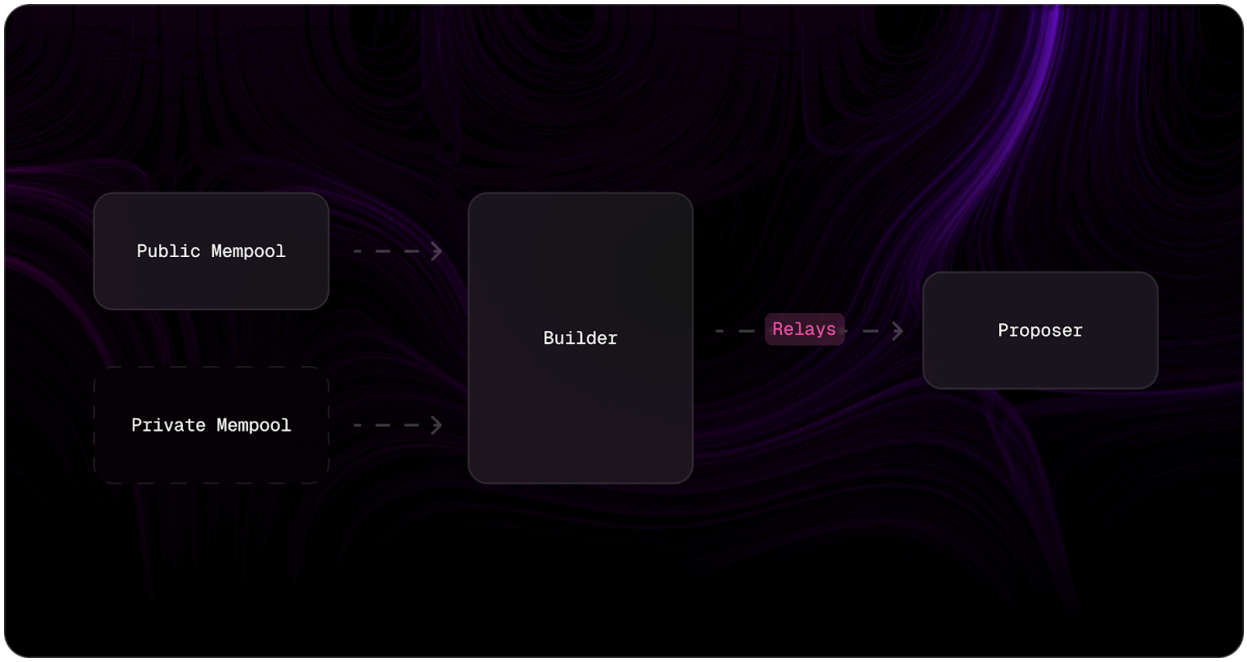


Diagram of the current transaction supply chain

Applications enabled by ASS, on the other hand, have the following properties:

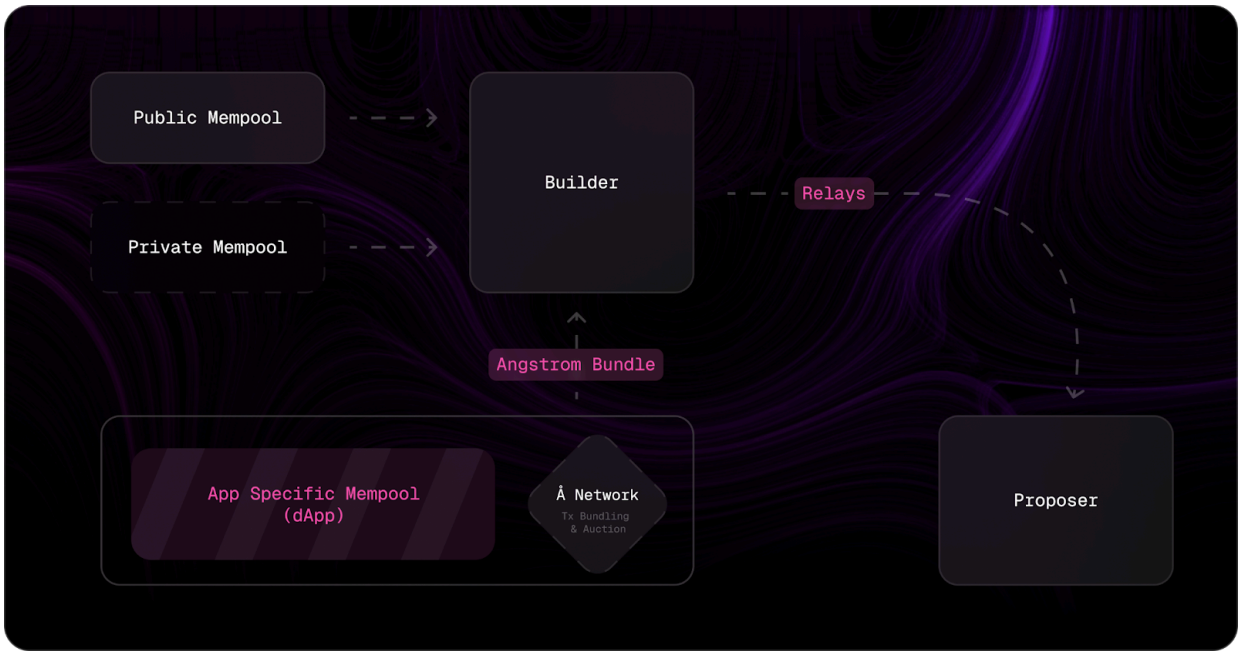
1. **Restricted Sequencing Rights**: This restriction ensures that only a designated sequencer or staked validators can interact with the application's contract on the chain it settles to, preventing nefarious bypassing of the application's logic for internal value redistribution.
1. **App-Specific Mempools**: Instead of submitting transactions to a public mempool, users send signed messages expressing their intent to an app-specific mempool. These intents are then collected and processed by the app-specific sequencer.
2. **Order-Agnostic Outcomes**: To enforce the sequencing rule and deliver the optimal economic payoffs to the target users, ASS transactions need to be order-agnostic to the builders' transaction ordering for the rest of the block. This is achieved by ensuring the state of the application is gated by its consensus mechanism. ASS orders are then consolidated into a single bundle which is sent to builders for inclusion. Given that this bundle isn't contentious with state accessed by other applications, it is agnostic to its position in the block.

ASS allows applications on any chain to regain sovereignty over its execution and state, hence enabling **sovereign applications**.

Given these fundamental tenets, let's use Angstrom as a practical example of a sovereign application. Angstrom is a UniswapV4 hook that protects its liquidity providers from adverse selection by CEX-DEX arbitrageurs, while also protecting swappers from sandwich attacks. A network of Angstrom nodes come to consensus, in parallel to Ethereum, on the set of transactions to be executed in the next block. The general flow is as follows:

1. CEX-DEX arbitrageurs bid for the right to be the first transaction to swap through the AMM (without a fee). Meanwhile, users send their intended swaps as signed limit orders to the Angstrom mempool.
2. The Angstrom Network runs its consensus protocol and forms a bundle where the first swap is the arbitrageur transaction with the highest bid. The bid amount is subsequently distributed, pro rata, to the underlying LPs in the range of the swap. All other valid limit orders, along with the underlying AMM liquidity, are executed at the same uniform clearing price.
3. This bundle is then sent to Ethereum builders and the public mempool by the proposing Angstrom node for the slot.
4. Builders include the Angstrom bundle at any position in the block. The bundle just needs to pay the base fee for inclusion due to its order-agnostic construction.

The following diagram illustrates the sovereign application in action.



The transaction supply chain in Angstrom

Liveness and Trust Assumptions

At its core, ASS is a form of partial block building where a sovereign application delegates the sequencing rights to a decentralized network of operators following a prescribed sequencing rule. This inevitably involves external parties who introduce additional liveness and trust assumptions.

Liveness Assumption

Sovereign applications depend on app-specific sequencers to follow the protocol correctly and provide timely state updates. In the event of a liveness breach, such as a network partition, users may be unable to

interact with parts of the application until valid consensus is restored.

Sovereign applications can also limit the scope of its functionality that requires external liveness assumptions. This can help ensure that critical states, like deposited liquidity, can remain accessible even in the event of a sequencer failure.

Trust Assumption

To ensure sequencers adhere to the prescribed sequencing rules, sovereign applications can leverage cryptoeconomic solutions (such as PoS) or cryptographic methods (like TEE or MPC). The specific approach may vary significantly depending on the needs of the application; some may require consensus on execution optimality, while others may focus on ensuring pre-execution privacy through cryptographic mechanisms. There are numerous tools available to reduce the trust overhead of sequencers and meet the unique objectives of each sovereign application.

Censorship Resistance

There are various types of censorship plaguing the Ethereum ecosystem:

1. Regulatory censorship: Builders and relays censor transactions based on the OFAC sanction list. This is one of the most prominent forms of censorship present on Ethereum today, predominantly enforced by relays.
2. Economic censorship: A motivated attacker can bribe the block proposer to censor victim transactions.
3. Node-level censorship: Nodes in the P2P network may refuse to propagate incoming transactions. This can be a major problem if the protocol operates optimally under the assumption that the majority of the nodes share the same view of the incoming transactions. Additionally, in such protocols, an adversary may be incentivized to partition the local views of the honest nodes (by sending a transaction to only half the nodes at the very end of a slot) and halt the protocol as a result.

Many researchers have voiced out the need for a better censorship resistance mechanism on Ethereum. Some proposals, like Multiple Concurrent Proposer (MCP) and Fork-Choice enforced Inclusion List (FOCIL), have surfaced and became the center of an ongoing discussion.

Censorship resistance is also a major concern for sovereign applications. The app-specific sequencers are likely external entities with various interests in receiving additional private transactions and orderflow. For example, an app-specific validator who is a market maker has an incentive to censor transactions sent by competing market makers. The sovereign application on top can thus suffer local censorship even if the base protocol is non-censoring.

Angstrom provides us with an example for a sovereign application with censorship resistance baked into its ASS design. To ensure that all valid orders are included in the upcoming slot, Angstrom nodes must broadcast any verified incoming orders and come to consensus on their inclusion within the proposed

transaction bundle. If the bundle is missing orders observed by the majority of the network, the proposer will be penalized. Here is an illustration of the censorship resistance mechanism for Angstrom.

Censorship resistance in a decentralized sovereign application

The Composability Dilemma

One of the major challenges that sovereign applications face is ensuring composability with transactions that interact with external contract states. Simply bundling app-specific transactions with arbitrary external ones undermines the order-agnostic property that is necessary for protecting the sovereign application and its users. A single invalid non-ASS transaction, when composed with an app-specific one, can have the second-order effect of reverting the entire bundle. When this happens, the sovereign application cannot execute its users' orders during the allocated slot (despite valid consensus being reached), thus harming user experience and overall welfare.

There are, however, potential solutions to the issue of composability, several of which are being explored by various teams. These include ideas like inclusion preconfirmations, shared app-specific sequencers, and builder commitments, each offering tradeoffs between degree of composability and trust overhead.

Inclusion Preconfirmations

To explain inclusion preconfirmations, it's important to first understand how based preconfirmations work. Based preconfirmations leverage cryptoeconomic security by ensuring proposers have put forth

staked collateral to guarantee the inclusion of a specific set of transactions ahead of a slot within the current epoch. This guarantee is limited by the size of the bond posted by the participating proposers.

Inclusion preconfirmations are a specialized form of based preconfirmations, where transaction inclusion is independent of any contract state. Transactions requesting inclusion preconfirmations must be state-agnostic and non-contentious, meaning their execution is unaffected by their position within the block. By utilizing inclusion preconfirmations, proposers can commit to including a non-ASS transaction only if the ASS bundle is included in the same block. This approach provides cryptoeconomically enforced composability between non-contentious transactions and ASS bundles.

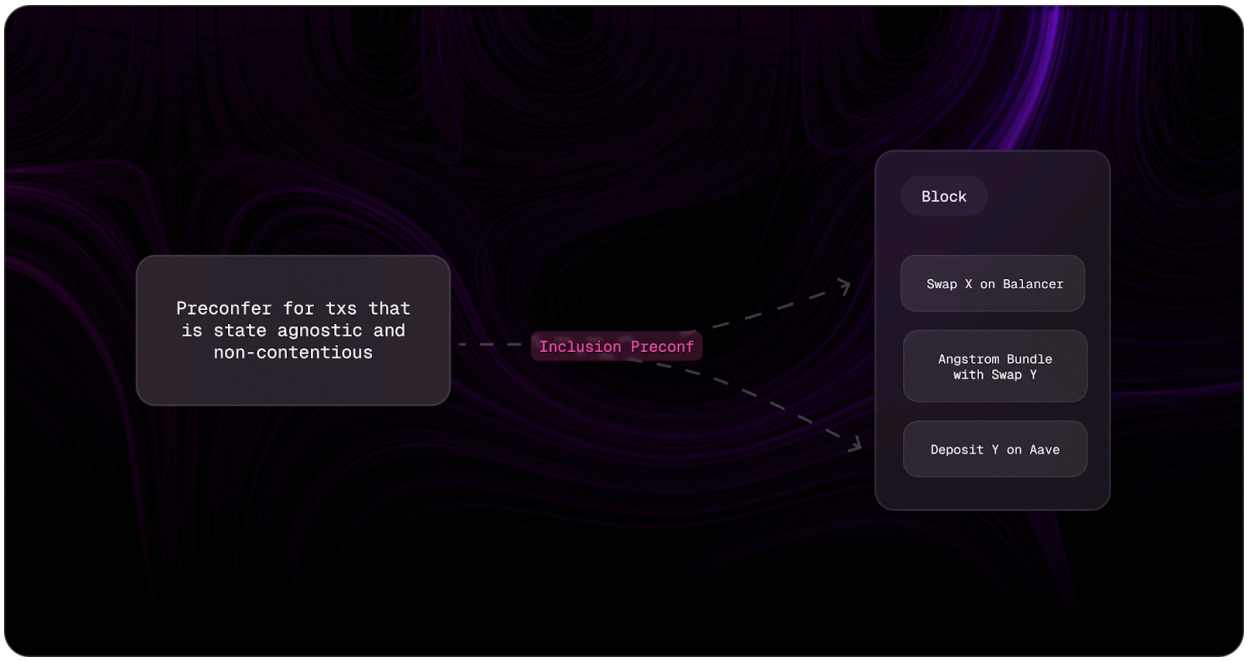


Illustration of inclusion preconf with ASS

However, given the limited composability provided by this solution, the added complexity and trust overhead may outweigh its benefits for certain sovereign applications. As a result, it's important to explore alternative approaches that could offer a more effective balance between simplicity and functionality.

Shared Application Specific Sequencers & Builder Commitments

Instead of relying on proposer commitments, sovereign applications can use app-specific sequencers to manage transaction ordering across multiple applications. For example, a sequencer handling transactions for several sovereign applications can facilitate atomic composability between them, as long as it follows the sequencing rules of each. This shared app-specific sequencer approach enables seamless composability and coordination across sovereign applications.

However, for non-sovereign applications, a different solution is required. Transaction inclusion commitments from block builders who participate in sequencing for sovereign applications can create atomic composability between non-sovereign and sovereign applications. Builders ensure that the specified transaction order across both types of applications is upheld – such builder commitment can bridge the composability gap for sovereign applications.

Illustration of builder commitment for atomic composability between ASS and non-ASS dApps (right) and shared app-specific sequencer for atomic composability between ASS dApps (left)

While there remain open questions regarding the economic dynamics of builder commitments, the feasibility of inclusion preconfirmations, and potential second-order effects of both, we are confident that the composability challenges that sovereign applications face will be resolved over time.

Sovereign Apps vs App-specific L2s and L1s

Currently, onchain apps have to build app-specific chains if they want to take control of the sequencing of their transactions. Concepts like Protocol Owned Builder (PoB) enables Cosmos L1s to have more expressive sequencing rules that help capture and redistribute MEV to their application. Similarly, an L2 sequencer with a VSR can have similar functionality. While both solutions allow for more expressive sequencing and MEV-capture by applications, sovereign apps are unique for the following reasons.

1. **No settlement overhead** – ASS does not execute or settle the sequenced transaction. Only the sequencing is delegated. The baseline trust assumption extends from the native execution environment, such as Ethereum or other L2s.
2. **Access to network effects** – Users do not need to bridge. Apps can directly leverage the users and liquidity on the native chain.
3. **Assets don't require locking** – Unlike L2s, most sovereign apps don't require users to lock their funds in bridging contracts. This design choice offers better security: if an app-specific sequencer fails, the potential damage is limited since the sequencer can only control transactions within the boundaries set by the smart contract. While some L2 solutions do implement safety features – such as escape hatches and forced inclusion – these measures are often difficult to use in practice. Users might need to wait several days before they can activate an escape hatch after losing connection to L2 updates. Similarly, forced inclusion via L1 typically involves at least a day's delay. Perhaps most importantly, these safety measures usually require technical expertise that most users don't have, making them impractical for the average person.
4. **Strong Liveness** – Liveness of the L2 usually depends on the centralized rollup sequencer. Liveness of the L1 depends on the honest majority of the nodes re-executing corresponding state transition functions. Liveness of a sovereign application mostly depends on the underlying execution environment (smart contracts can specify parts that need to rely on app-specific sequencers), thus allowing sovereign apps to inherent liveness from the most decentralized layer for critical functionality.

Properties	Execution Overhead	Liveness Overhead	Asset Control	Liquidity Access
Sovereign dApp	None	Dependent on the state update that the contract designates to sequencers	Cannot freeze asset	Based on the deployed execution environment (No bridging required)
L2	Secured via ZKP or fraud proof or TEE	Fully dependent on sequencer	Can freeze Require Forced inclusion/ escape hatch	Require bridging to bootstrap the liquidity and flow
Based L2	Secured via ZKP or fraud proof or TEE	Fully dependent on execution node	Can freeze Require Forced inclusion/ escape hatch	Require bridging to bootstrap the liquidity and flow (Can atomically compose with ETH L1)
L1	Honest Majority	Honest Majority	Honest Majority	Require bridging to bootstrap the liquidity and flow

Table comparing ASS applications, L2, Based L2, and L1

Conclusion

ASS empowers applications with full sovereignty over transaction sequencing, enabling them to define custom rules without the complexity of managing execution. This sovereignty allows applications to control its execution to optimize outcomes for their users. For instance, on Angstrom, LPs and swappers are treated as first-class citizens, with their economic payoff directly enhanced through custom sequencing rules.

Additionally, ASS can leverage a range of cryptoeconomic and cryptographic tools to enforce the optimality of user payoffs and implement robust censorship resistance mechanisms. Cryptoeconomic solutions, such as staking and slashing, can incentivize honest behavior among sequencers, while cryptographic methods like TEE and MPC enhance privacy and security. With these tools, the design potential of ASS is vast, allowing for the creation of more secure, efficient, and user-centered sovereign applications.

Despite the opportunities ASS offers, challenges such as the lack of native composability still exist. However, solutions like inclusion preconfirmation, shared ASS, and builder commitment present promising ways to overcome these hurdles. While some questions remain, we are committed to refining these approaches to deliver a smoother, more composable ASS experience.

We are here to make DeFi more sustainable, one ASS at a time.