By Lin Oshitani (Nethermind Research). Thanks to Conor and Aikaterini for the detailed discussions and review. Thanks also to Ahmad and Brecht for their review and comments. This work was partly funded by Taiko. The views expressed are my own and do not necessarily reflect those of the reviewers or Taiko.

Introduction

Based sequencing provides a credibly neutral shared sequencer layer that enables composability among rollups and between rollups and L1. Additionally, based preconfirmations provide fast preconfirmation services on top of based sequencing, significantly enhancing the user experience as a result. However, compared to non-preconfirming based sequencing, naive implementations of based preconfirmations introduce negative externalities that require thoughtful consideration. Although issues have been highlighted in works such as this Bell Curve episode (mainly in the context of non-based sequencers) and this write-up from Chainbound, we believe the topic remains largely underexplored.

In this post, we will analyze a simple “strawman” preconfirmation setup, identify its shortcomings, and shed light on the challenges that future solutions must address.

The Strawman

The strawman based preconfirmation setup is as follows:

- The L1 proposer may or may not delegate the preconf right to some external entity.
- We use the term preconfer

to describe the entity providing preconfirmations, which can either be the L1 proposer itself or an entity delegated from the L1 proposer.

- We use the term preconfer

to describe the entity providing preconfirmations, which can either be the L1 proposer itself or an entity delegated from the L1 proposer.

- The preconfer handles preconfirmations by providing two endpoints:

- Request endpoint: For users and searchers to request preconfirmations.

- Promise endpoint: For streaming the preconf results to the public. It enables the preconfirmation requester to promptly receive the result while allowing other users to stay updated on the latest preconfirmed state before initiating their own preconfirmations.

- Request endpoint: For users and searchers to request preconfirmations.

- Promise endpoint: For streaming the preconf results to the public. It enables the preconfirmation requester to promptly receive the result while allowing other users to stay updated on the latest preconfirmed state before initiating their own preconfirmations.

- Users will include a “preconf tip” to the requests to incentivize preconfers to provide preconfirmations.

- The preconfer preconfirms transactions primarily on a first-come-first-serve basis.

Furthermore, because they are significantly more complex to design and implement, we focus solely on execution promises. Execution promises guarantee the exact sequence and state of a transaction. In contrast, inclusion promises only ensure that a transaction will be included without specifying the conditions of its inclusion.

The Problems

We will cover six problems with the strawman based preconfirmation design:

- Problem 1: Latency races

- Problem 2: Congestion

- Problem 3: Tip pricing

- Problem 4: Fair exchange

- Problem 5: Liveness

- Problem 6: Early auctions

Problem 1: Latency races

Whoever has the lowest latency to the preconfer gains all the MEVback-running profit. This is because they can:

1. Be the first to obtain the latest state of the chain through the promise endpoint and
2. be the first to insert their back-run transaction via the request endpoint.

This structure has historically [incentivized latency races

](https://academic.oup.com/qje/article/130/4/1547/1916146), where network participants strive to minimize latency to the limit. Eventually, this would lead to searchers choosing to colocate or vertically integrate with preconfers, which significantly risks the network's geographical decentralization.

Such latency races have been a long-lasting concern for existing centralized sequencers. For example, the Arbitrum team has explored the idea of implementing Proof of Work (PoW) where they grant fast connections to participants who succeed in PoW while imposing artificial delays on participants who do not. However, this proposal encountered backlash from the community due to the substantial economic waste introduced.

Problem 2: Congestion

Given that L2 transaction fees are typically low, searchers may choose to avoid latency races altogether and instead flood the rollup with probabilistic arbitrage attempts. This can be done by spamming an arbitrage contract that attempts an arbitrage and rollbacks if it fails. In Solana, where the fee is extremely low, it has been reported that validators waste ~58% of their time processing such failed arbitrage transactions.

This would result in a situation resembling pre-Flashbotspriority gas auctions, where the competition among searchers congests the block space with failed arbitrage transactions, ultimately driving up gas fees for regular users.

Problem 3: Tip pricing

The preconfer must solve an online MEV problem, where they decide whether to preconf a transaction with no/limited visibility to other transactions that compete for the same position. For example, suppose the preconfer receives a preconf request with 1 ETH tip. How would the preconfer know that the tip is appropriately priced? Should they accept the tip and preconf immediately or wait for a while in case there is another request with a higher tip?

Problem 4: Fair exchange

The preconfer can withhold preconf promises and not return them to the user in a timely manner. Note that preconfers are incentivized to withhold preconf promises as much as possible to maximize their opportunity to reorder and insert transactions, thereby increasing their MEV.

As an extreme example, the preconfer could withhold all promises during its window (12 sec or more), reorder and inject txs as it wishes, and only publish the promises when the final tx batch is submitted to L1.

Problem 5: Liveness

For the case when the proposer delegates the preconfirming rights to an external preconfer, the liveness and censorship resistance of the preconfirmations will rely solely on this single external entity for the duration of the preconfer's slot(s).

Problem 6: Early auctions

Any system with L1 composable preconfirmations (i.e., preconfirmation of L1 transactions) will likely result in preconfer-builder integration,

where preconfer and builder become the same entity. This is for two reasons:

- With L1 preconfirmations, most cross-domain MEV, including CEX-DEX arbitrage, will be captured through preconfirmed transactions.
- Considering the bulk of MEV revenue comes from CEX-DEX arbitrage, this means that most MEV revenue will be secured through preconfirmations. Consequently, the revenue from building the non-preconfirmed portion of the block will be greatly reduced.
- Considering the bulk of MEV revenue comes from CEX-DEX arbitrage, this means that most MEV revenue will be secured through preconfirmations. Consequently, the revenue from building the non-preconfirmed portion of the block will be greatly reduced.

- Preconfed L1 transactions must be included at the top of the current block.
- Inserting any transactions before a preconfirmed transaction could alter the state anticipated by the preconfirmation, potentially invalidating the preconfirmation guarantee. This means builders must constantly incorporate the latest preconfed transactions into their blocks, which would be extremely difficult, if not unfeasible.

- Inserting any transactions before a preconfirmed transaction could alter the state anticipated by the preconfirmation, potentially invalidating the preconfirmation guarantee. This means builders must constantly incorporate the latest preconfed transactions into their blocks, which would be extremely difficult, if not unfeasible.

Combined with preconf delegation happening ahead of the proposer's slot, preconfer-builder integration leads us to a world where L1 proposers delegate their preconfirmation rights and block-building rights

to the same external entity ahead of time

for their slot.

Selecting the block builder in advance, known as early auctions

, contrasts sharply with the current MEV-Boost PBS pipeline, where block builders are dynamically chosen just-in-time

(JIT

) within the slot through block auctions. More details comparing JIT auctions and early auctions can be foundhere.

The goal of based sequencing is to inherit the security of L1. However, with based preconfirmations, we risk altering the security landscape of the underlying L1 itself. Although early auctions might not be entirely detrimental (further research and experimentation are needed), they represent a fundamental shift from the current MEV-Boost builder market. Therefore, they should be introduced with great care, especially when introduced off-protocol, where control over centralization tendencies is limited.

Conclusion

We observed that naive implementations of preconfirmations can lead to various negative externalities. As with all things in blockchains, trade-offs are inevitable. However, such negative effects should be mitigated as much as possible and, when needed, introduced as a deliberate choice, not an accident.

At Nethermind, along with our collaborators, we are actively researching solutions that address the issues outlined in this document. Stay tuned for more updates!