[cross-posted from Medium]

tl; dr: We're proposing a modified transaction ordering policy for the Arbitrum sequencer, adding a "time boost" to the current first-come, first-served policy, where a transaction could pay a priority fee to get a small advantage or "time boost" in the ordering. This shouldn't affect most users, but can provide a better way to manage "latency racing" behavior.

The Arbitrum sequencer receives transactions from users and publishes an ordered sequence, which serves as input to the execution stage of Arbitrum. Currently, the sequencer follows a first-come, first-served (FCFS) sequencing policy.

There's a lot to like about FCFS. It's simple and easy to explain. It seems intuitively fair. It minimizes latency because the policy allows each transaction to be appended to the sequence immediately on arrival.

Other rollup protocols use FCFS as well. Optimism does, and based on descriptions of other systems they seem to do so as well.

But FCFS has some disadvantages, mainly that it can induce "latency racing" behavior, where sophisticated actors spend money and effort in a wasteful arms race to get closer to the sequencer, so they can get their transactions in slightly ahead of their competitors.

We think there is a way to do better, by adopting a modified version of FCFS, which we'll describe below.

Goals

We want a transaction ordering policy to have these properties:

- secret mempool:

Submitted transactions are not visible to anyone other than the sequencer, until they are included in the published sequence. This prevents parties from front-running or sandwiching others' transactions. (The sequencer is trusted not to engage in such tactics, although this trust requirement would be removed with the move to decentralized sequencing.)

- low latency:

Every transaction that arrives at the sequencer is included in the sequence within a short time limit, perhaps 0.5 seconds.

- short-term bidding:

Over short time intervals, transactions can gain an advantage in the ordering by bidding for position. This is meant to induce parties who are racing for early placement to do so by bidding, rather than spending resources (e.g., for latency reduction) to deliver their transactions to the sequencer faster.

- compatible with decentralization:

The policy can be adapted to work with a decentralized sequencer, so we don't have to abandon the policy when the sequencer is decentralized.

Time boost ordering policy

The new policy is a modified first-come, first-served. The mempool is still secret, to prevent front-running.

Every transaction is timestamped when it arrives at the sequencer. A transaction can choose to pay an extra priority fee, which will give it a time boost — making its timestamp slightly earlier, by up to 0.5 seconds.

Why 0.5 seconds? This parameter reflects a tradeoff. We want transaction senders to be able to buy a large enough boost that they have incentive to buy a boost rather than trying to engineer latency — which makes us want to have a larger maximum boost. But also we want to minimize the impact on the latency of non-boosted transactions, which will potentially have to wait for this period in case a boosted transaction might arrive after them — which makes us want a smaller minimum boost. We think 0.5 seconds balances the tradeoff reasonably, although others might have good arguments for a slightly smaller or larger value.

We expect that most transactions won't buy a time boost, so no changes are needed in wallets or application UX. Sophisticated parties who are already constructing Arbitrum transactions programmatically can decide whether to buy a time boost (and how much to buy), based on whether they're competing with others for early position and how much they value being first.

If a transaction's priority fee is F

, it will get a time boost (i.e., a subtraction from its timestamp) computed by this formula:

where is the g

maximum time boost available (planned for 0.5 seconds in production), and c

is a constant to be determined. In exchange for this time boost, the L2 gas price paid by the transaction will be increased by F

.

[

1400×833 69.8 KB

](https://global.discourse-cdn.com/standard17/uploads/arbitrum/original/2X/f/ffe8eaf827b4bb46252b5e994f1a6f78b8cfb80d.png)

Here’s a graph of the formula, assuming the maximum boost g

is 0.5 seconds. By design, a lower fee buys a good chunk of the available boost, with diminishing returns as the fee goes up. The boost approaches 0.5 seconds if the fee gets very large.

Implementation

Here’s an approach to implementing the policy. Legacy transaction types should continue to work as before, so that no changes are necessary for most users and developers. As always, Arbitrum would charge gas fees on L2 but would ignore the “priority fee” fields in legacy transaction formats. (Many existing transactions send a non-zero priority fee to Arbitrum. Arbitrum does not collect that priority fee, and that shouldn’t change.)

Transactions that want a time boost would use a new L2 transaction type, which would be the same format as a legacy transaction (other than having a different transaction type label). For the new transaction type, the Ethereum priority fee field would be interpreted as a time boost fee, and would be collected by the Arbitrum chain.

The Arbitrum sequencer would apply the time boost formula, adjust timestamps accordingly, and sequence transactions in increasing order of the adjusted timestamp.

One consequence of the policy would be that transactions that didn’t pay for time boost would experience an extra 0.5 seconds of latency, because the sequencer would need to wait to see if any transactions coming soon after had a boost. But no transaction would ever need to be held for longer than 0.5 seconds.