This is an exploratory proposal for a method of deterministically identifying sequencers on L2s to route transactions to, in order to facilitate universal synchronous composability between L2s. This approach uses the Ethereum base layer as a universal and credibly neutral sequencer selection mechanism, intended as a fallback to based sequencing, in cases where based sequencing may not be suitable for an L2 to adopt fully. This is very much an initial proposal intended for general feedback, discussion and debate.

Proposal

LimeChain's proposal on Vanilla Based Sequencing describes two methods for selecting the sequencer on L2: primary selection, when the current L1 proposer has opted-in to be an L2 sequencer for the rollup, and fallback selection, when the current L1 proposer has not

opted-in to be an L2 sequencer for the rollup, and hence some other method is required to select the L2 sequencer.

In the fallback selection mechanism, the L2 selects an L1 proposer at random from the other opted-in L1 proposers to be the L2 sequencer. This method will work perfectly well, but limits the possibility for synchronous composability between rollups. A single L1 proposer that is also the designated sequencer on multiple rollups is able to offer guarantees that bundles of transactions will be executed atomically on the respective rollups. In the fallback selection mechanism, there is no longer a single L1 proposer sequencing for two or more rollups at the same time, and instead we have separate and distinct L1 proposers, and guarantees of atomic execution of transactions are no longer possible.

To address this challenge, I propose to explore the idea of a deterministic sequencer selection function on L1 that can be used by any rollup. This would allow wallets or pre-confirmation gateways to identify which sequencers will be selected when, on which rollups, allowing them to route pre-confirmation requests accordingly.

The high level idea is that a pre-confirmation gateway can accept a bundle of transactions, typically a pair of transactions to be executed on two distinct rollups, and can route the requests to the relevant sequencers on the respective rollups, obtaining the preconf promises for the user.

Requirements

- Sequencers on L2s will need to be recorded via some central registry on L1 (perhaps an extension of the L1 pre-confirmation registry), and will need to be able to issue pre-confirmations.
- Pre-confirmations will probably need to be confirmed in 2 rounds instead of one. The first request is to place a lock on the preconfs (on a very short timeframe, in the order of 100s of milliseconds), and the second is to confirm. This will allow a pre-confirmation gateway (or wallet) to obtain a commitment that both sequencers on their respective rollups will issue a preconf promise, and when the sequencer has commitments from both, it can confirm the preconf requests and receive the preconf promises from both. The nomenclature gets a bit confusing here and will need to be thought about. Alternatively, another potentially less complicated but less secure approach is to allow the pre-confirmation gateway to cancel a pre-confirmation upon only receiving a preconf promise from one but not both L2 sequencers.
- There is a deterministic function in a smart contract on L1 which can be used to identify which L2 sequencer is going to be selected when on which rollup. This could leverage prevrandao

via block.difficulty

to provide a lookahead which L2s use to select sequencers on L2. This could be as simple as function that acts as a PRF which accepts a lookahead size n

measured in slots, and bounded range r

corresponding to the number of L2 sequencers, and which references block.difficulty

to return a list of random sequencers ids of size n

, within the range r

.

- May require some collateral to incentivize L2 sequencers to honor preconf requests, and to keep the L1 registry up-to-date, though this last point could prove difficult to coordinate among L2s and will require some more thought.

Rationale

The fallback sequencer selection will be employed by L2s more frequently in the beginning, until enough L1 proposers opt-in to based sequencing.

We need at least 20% of Ethereum's validator set to opt-in to based sequencing in order to have at least 1 proposer per epoch. Moreover, these proposers will need to opt in to being sequencers for a number of rollups (assuming enough rollups will provision for this in their sequencer designs - tbd). All of this is against a background of numerous AVSs that could offer more economically attractive alternatives for proposers to opt into.

Assuming we do reach the required threshold of 20% of the Ethereum validator set having opted-in to based sequencing (roughly 200K validators), depending on the design of decentralized L2s, they may not be able to support this many sequencers. For L2s that might seek to implement some form of consensus protocol using a validator set with a proposer selection mechanism, it may be impractical to support 200K validators. In this instance, having a credibly neutral sequencer selection mechanism would be helpful.

Furthermore, the fallback selection mechanism will likely to be continued to be employed by newer or smaller rollups in the future, before they can build a large enough share of L1 proposers that opt-in to being sequencers for their rollups. Again, in this instance, credibly neutral sequencer selection could be appealing.

The fallback selection by itself doesn't support universal synchronous composability, but synchronizing sequencer selection across rollups in a credibly neutral way (by using the L1), can facilitate USC.

Risks and Trade-offs

Users will need to trust the crypto-economic guarantees of the respective L2s that they are seeking preconfs of atomic execution of transactions from. Even if the L2 sequencers are also L1 validators (note that they don't need to be), users will still need to rely on the L2's security.

Under the primary selection method of the vanilla based sequencing approach, these security guarantees are improved somewhat, as we are dealing with a single L1 proposer for both rollups, and so their ability to offer guarantees of atomic transaction execution across L2s is improved.

Assuming that the L1 proposer posts the data from each rollup to the L1, then we also benefit from the subjective finality of those transactions, and so we know within one slot if the respective transactions have been included or not. This also a strong assumption however, as it depends on the volume of transactions on L2, with most rollups waiting until they have enough transactions to completely fill a blob with compressed data, which may not happen within one L1 slot. So again, it falls back to the user relying on the crypto-economic security of the L2s themselves.

This approach will of course require some buy-in from L2s that will need to implement the fallback sequencer selection, but this is also the case with based sequencing in general.

A major trade-off is the increased complexity from managing atomic execution guarantees, as described in the requirements section above. This will likely involve a lock-and-confirm mechanism between L2 sequencers and the preconf gateways (or a user's wallet), whereby the preconf gateway asks the sequencer to put a hold on a preconf for about a second, and then confirms that they want to go ahead with the preconf.

Open Questions

It's not clear to me exactly how to incentivize L2s or L2 sequencers to keep the registry up-to-date. If an L2 sequencer ceases to participate as an L2 sequencer, the registry will need to reflect this. This challenge is addressed in mteam's proposal for Credibly Neutral Preconfirmation Collateral, and potentially the same registry could be used for L2 sequencers, but this needs to be explored. Using a separate registry could bloat the ecosystem with the redundant overhead of managing multiple registrations for based-sequencing proposers.

There may be understandable apprehension from the ecosystem if the preconf gateways on L1 are used to route transactions to sequencers on L2s. This will cause even more centralization across the ecosystem, with potentially only 2 or 3 preconf gateways routing transactions across L1 and also a number of L2s. This could be improved through routing to individual sequencers via independent gateways on each L2, but this needs to be explored further.

This proposal as it stands will not withstand L1 re-orgs. It is an open question how the sequencer selection mechanism can be impervious to L1 re-orgs.

All questions, feedback, criticisms are very welcome.