Authors: Bo Du (Polymer Labs)

There are three core issues that could be solved using a cosmos sdk (PoS) based shared or single rollup sequencer for Ethereum rollups. The solution gets us both native IBC interoperability and decentralized sequencing at the same time!

The first issue is not well defined and incompatible transport layers across rollups and chains. An interoperability protocol has three layers - application (e.g. HTTP, gRPC), transport (e.g. TCP/UDP) and state (e.g. physical network links). The transport layer produces the commitment of chain A to some messages intended to be sent to chain B. This commitment is generally in the form of some merkle root.

Inter-Blockchain Communication Protocol or IBC has an extremely well defined transport layer that is analogous to TCP/UDP in the OSI model. IBC transport layer commitments would be understandable by any IBC enabled chain as well. This would effectively merge Ethereum and the Cosmos (and more because the IBC network is growing)

.

Seamless interoperability between chains requires a standardized transport layer otherwise packet commitments made by one chain would not be understood by another chain. IBC is already the standard transport protocol in the Cosmos and can be the standard transport protocol in Ethereum as well.

The second issue is the problem of centralized canonical bridges in the L1 → L2 direction. Currently these bridges are both centralized and maintained by the rollup teams themselves. Using cosmos sdk based sequencers would allow rollups to communicate with one another via light clients and IBC instead of using a centralized canonical bridge.

The third issue is the problem of centralized sequencers. Currently L2 protocols like Arbitrum and Optimism run their own sequencer internally. While this is a great business for them, this is not great for decentralization. A PoS sequencer means that anyone can acquire tokens to join the network as a block proposer. It becomes a free and dynamic market with an easy MEV integration (it's easy to add MEV logic to a cosmos sdk chain).

In summary, there are three core issues a cosmos sdk based sequencer would solve:

- Incompatible and not well defined transport layers for interoperability
- Centralized canonical bridges in the L1 → L2 direction
- Centralized sequencers

This solution solves for the centralized sequencer, canonical bridge and interoperability problems at the same time.

Implementation wise a few things would need to happen for L2 <> L2 IBC communication. Let's take a high level look at this.

On the sending side rollup:

- Bindings between the rollup VM → IBC transport logic on the sequencer would need to be made
- IBC transport logic on the sequencer would commit to all of the IBC related messages produced by the rollup
- The IBC transport commitment needs to be posted on chain on Ethereum (in addition to the rollup state commitment)

On the receiving side rollup:

- Run an ethereum light client on chain or

run a tendermint light client (use the PoS validators of the sequencer of the L2 as an attester to an Ethereum block)

- Relayers prove the transport commitment made by the sending L2 exists in the Ethereum state commitment
- Relayers can query for and supply merkle proofs for packets within the transport commitment

There's some additional exploration that can be done here around the sequencer sharing security from Ethereum re-stakers on a solution like EigenLayer. This would provide additional security for receiving L2s that opt to use the PoS validators of the sequencer of the L2 as an attester to an Ethereum block.

For those coming from the web2 space, the solution w.r.t. interoperability can be thought of as introducing a networking sidecar

. Think Sidecar by Istio or Envoy by Lyft.

Let's bring all ecosystems together through IBC and decentralized sequencers!