TL;DR

- To obtain faster finality, we describe a hierarchy of checkpoints attesting to the validity of StarkNet's state: Full Node checkpoints every "minute", and public L1 checkpoints every "hour".
- Both types of checkpoints rely on validity proofs, and as such are final.
- Both types of checkpoints are equally secure: the "minute" checkpoints can be made public on L1 by any StarkNet Full Node.
- The "hour" checkpoint can be a recursive validity proof, proving its composite "minutes".

1. Motivation

Large blocks and fast finality are two desired properties in a ZK-Rollup system. Why?

Large Blocks:

For ZK-Rollups and specifically for StarkNet to scale, they need to generate the largest blocks possible. The cost of verifying a STARK proof is poly-logarithmic in the number of transactions in the batch, so the cost of performing this verification on-chain [1] does not change much with the batch size. Therefore, the more transactions in one proof, the merrier.

Fast Finality:

If Alice has just received funds from Bob, she would like to know that the operation is finalized and that she has these funds in her account.

These two desired properties present a trade-off: the first one requires large blocks in order to improve scalability, which leads to long delays, while the second one requires short block intervals in order to gain faster finality.

In this proposal, we offer a way to satisfy both properties, using checkpoints.

2. Related work

Our proposal shares the motivation and general idea with a previously published proposal related to ZK-Rollups based on SNARK proofs, called A pre-consensus mechanism to secure instant finality and long interval in zkRollup by Leohio. Leohio suggested using checkpoint commitments as well. In his scheme, the next state is sent to L1, along with the ZK-proof and other needed data to prove its integrity as calldata. The new state and the hash of this data are stored as a commitment on L1. Anyone can verify the validity of the commitment using the data available in the calldata, or publish a fraud-proof to refute it.

This solution is suitable for ZK-SNARKs, as their proof size is small. Unfortunately, it is unsuitable for ZK-STARKs as these proofs are larger and they scale poly-logarithmically.

Consequently, because of the STARK proof size, it is more expensive to include it in the L1 event, and therefore cannot use this mechanism for fast finality with checkpoints. We propose below a different mechanism.

3. Background: How does StarkNet's State-Update work?

In StarkNet there is a Sequencer that decides what sequence of transactions will be included in the next block. The Sequencer sends the block to a Prover, which computes a proof and submits it to the L1 Verifier contract. Once the Verifier verifies a block's proof, it stores on-chain the associated Fact - the result of the off-chain computation. This Fact, which attests to the validity of the block, can later be queried by anyone. After the Fact is registered on L1, a State-Update L1 transaction is called to update the global state of StarkNet according to the newly generated - now also STARK-proven and verified - L2 block of transactions.

Importantly: The State-Update L1 operation is executed if and only if a corresponding Fact exists on-chain.

4. The algorithm

Roles

There are two types of operators in StarkNet: Sequencers and Full Nodes. Sequencers aggregate and order transactions into blocks, and provide a block's validity proof. Full Nodes store the full state of the network.

The algorithm assumes there is only one Sequencer per epoch/block height, as will be the case for StarkNet [2].

Basic Flow

We define two types of checkpoints: “minute” checkpoints and “hour” checkpoints. A checkpoint includes a new state and a proof, attesting to the validity of the state transition since the previous checkpoint, and other supporting data (e.g. data availability).

In order to reduce gas costs, for the “minute” checkpoints, only the state itself is committed on-chain; the rest of the data - validity proof, data availability, etc. - is sent over the StarkNet gossip network and is stored and verified by StarkNet Full Nodes, while “Hour” checkpoints are verified publicly on L1. Both types of checkpoints has the same finality: the “hour” checkpoints are verified on L1, and the “minute” checkpoints validity proofs are verified by StarkNet Full Nodes and can be posted on L1 by any one of those nodes, thus giving that “minute” checkpoint L1 finality.

Full Nodes who wish to check the finality of a specific transaction will be able to do so by checking the following conditions:

1. An on-chain commitment of the new state was published on L1.
2. They have a valid validity proof for this state, along with other required validity data.
3. The specific transaction is a part of the new state.

Optimizations

Recursive Proofs: The generation of small interval validity proofs requires computing power from the network, however, this computation is put to good use: to generate a large time interval validity proof, all of its small interval proofs are aggregated into a recursive STARK proof that attests to the validity of the large interval’s state transition. The recursive proof will then be submitted to L1.

5. Summary

We described a protocol for using a two-layered checkpoint system on StarkNet in order to achieve fast finality while still allowing for large blocks. In this protocol StarkNet Full Nodes track checkpoints every “minute” (thus allowing them to get faster finalization for txs), whereas L1 maintains a record of “hour” checkpoints. Both types of checkpoints are equally secure: the “minute” checkpoints can be made public on L1 by any StarkNet Full Node. Recursive proofs can be used in order to construct a validity proof for an “hour” checkpoint from its constituent “minute” checkpoints.

[1] roughly 5M gas currently.

[2] In StarkNet’s second development phase (Constellations), there is only one Sequencer, and in the final development phase (Universe), a leader election mechanism is used to choose a Sequencer for a certain time period.