Celestia's data availability layer

Celestia is a data availability (DA) layer that provides a scalable solution to thedata availability problem . Due to the permissionless nature of the blockchain networks, a DA layer must provide a mechanism for the execution and settlement layers to check in a trust-minimized way whether transaction data is indeed available.

Two key features of Celestia's DA layer aredata availability sampling (DAS) andNamespaced Merkle trees (NMTs). Both features are novel blockchain scaling solutions: DAS enables light nodes to verify data availability without needing to download an entire block; NMTs enable execution and settlement layers on Celestia to download transactions that are only relevant to them.

Data availability sampling (DAS)

In general, light nodes download only block headers that contain commitments (i.e. , Merkle roots) of the block data (i.e. , the list of transactions).

To make DAS possible, Celestia uses a 2-dimensional Reed-Solomon encoding scheme to encode the block data: every block data is split intok × k shares, arranged in ak × k matrix, and extended with parity data into a2 k × 2 k extended matrix by applying multiple times Reed-Solomon encoding.

Then,4 k separate Merkle roots are computed for the rows and columns of the extended matrix; the Merkle root of these Merkle roots is used as the block data commitment in the block header.

To verify that the data is available, Celestia light nodes are sampling the2 k × 2 k data shares.

Every light node randomly chooses a set of unique coordinates in the extended matrix and queries full nodes for the data shares and the corresponding Merkle proofs at those coordinates. If light nodes receive a valid response for each sampling query, then there is ahigh probability guarantee that the whole block's data is available.

Additionally, every received data share with a correct Merkle proof is gossiped to the network. As a result, as long as the Celestia light nodes are sampling together enough data shares (i.e. , at leastk × k unique shares), the full block can be recovered by honest full nodes.

For more details on DAS, take a look at theoriginal paper .

Scalability

DAS enables Celestia to scale the DA layer. DAS can be performed by resource-limited light nodes since each light node only samples a small portion of the block data. The more light nodes there are in the network, the more data they can collectively download and store.

This means that increasing the number of light nodes performing DAS allows for larger blocks (i.e. , with more transactions), while still keeping DAS feasible for resource-limited light nodes. However, in order to validate block headers, Celestia light nodes need to download the4 k intermediate Merkle roots.

For a block data size ofn 2 bytes, this means that every light node must downloadO (n) bytes. Therefore, any improvement in the bandwidth capacity of Celestia light nodes has a quadratic effect on the throughput of Celestia's DA layer.

Fraud proofs of incorrectly extended data

The requirement of downloading the4 k intermediate Merkle roots is a consequence of using a 2-dimensional Reed-Solomon encoding scheme. Alternatively, DAS could be designed with a standard (i.e. , 1-dimensional) Reed-Solomon encoding, where the original data is split intok shares and extended withk additional shares of parity data. Since the block data commitment is the Merkle root of the2 k resulting data shares, light nodes no longer need to downloadO (n) bytes to validate block headers.

The downside of the standard Reed-Solomon encoding is dealing with malicious block producers that generate the extended data incorrectly.

This is possible asCelestia does not require a majority of the consensus (i.e. , block producers) to be honest to guarantee data availability. Thus, if the extended data is invalid, the original data might not be recoverable, even if the light nodes are sampling sufficient unique shares (i.e. , at leastk for a standard encoding andk × k for a 2-dimensional encoding).

As a solution,Fraud Proofs of Incorrectly Generated Extended Data enable light nodes to reject blocks with invalid extended data. Such proofs require reconstructing the encoding and verifying the mismatch. With standard Reed-Solomon encoding, this entails downloading the original data,i.e. ,n 2 bytes. Contrastingly, with 2-dimensional Reed-Solomon encoding, onlyO (n) bytes are required as it is sufficient to verify only one row or one column of the extended matrix.

Namespaced Merkle trees (NMTs)

Celestia partitions the block data into multiple namespaces, one for every application (e.g., rollup) using the DA layer. As a result, every application needs to download only its own data and can ignore the data of other applications.

For this to work, the DA layer must be able to prove that the provided data is complete,i.e. , all the data for a given namespace is returned. To this end, Celestia is using Namespaced Merkle trees (NMTs).

An NMT is a Merkle tree with the leafs ordered by the namespace identifiers and the hash function modified so that every node in the tree includes the range of namespaces of all its descendants. The following figure shows an example of an NMT with height three (i.e. , eight data shares). The data is partitioned into three namespaces.

When an application requests the data for namespace 2, the DA layer must provide the data sharesD3 ,D4 ,D5 , andD6 and the nodesN2 ,N8 andN7 as proof (note that the application already has the rootN14 from the block header).

As a result, the application is able to check that the provided data is part of the block data. Furthermore, the application can verify that all the data for namespace 2 was provided. If the DA layer provides for example only the data sharesD4 andD5 , it must also provide nodesN12 andN11 as proofs. However, the application can identify that the data is incomplete by checking the namespace range of the two nodes,i.e. , bothN12 andN11 have descendants part of namespace 2.

For more details on NMTs, refer to theoriginal paper .

Building a PoS blockchain for DA

Providing data availability

The Celestia DA layer consists of a PoS blockchain. Celestia is dubbing this blockchain as thecelestia-app , an application that provides transactions to facilitate the DA layer and is built usingCosmos SDK . The following figure shows the main components of celestia-app.

celestia-app is built on top ofcelestia-core , a modified version of theTendermint consensus algorithm . Among the more important changes to vanilla Tendermint, celestia-core:

- Enables the erasure coding of block data (using the 2-dimensional Reed-Solomon encoding scheme).
- Replaces the regular Merkle tree used by Tendermint to store block data with aNamespaced Merkle tree
- that enables the above layers (i.e.
- , execution and settlement) to only download the needed data (for more details, see the section below describing use cases).

For more details on the changes to Tendermint, take a look at theADRs . Notice that celestia-core nodes are still using the Tendermint p2p network.

Similarly to Tendermint, celestia-core is connected to the application layer (i.e. , the state machine) byABCI++ , a major evolution ofABCI (Application Blockchain Interface).

The celestia-app state machine is necessary to execute the PoS logic and to enable the governance of the DA layer.

However, the celestia-app is data-agnostic -- the state machine neither validates nor stores the data that is made available by the celestia-app. [][Edit this page on GitHub] Last updated: Previous page Monolithic vs. modular blockchains Next page The lifecycle of a celestia-app transaction []