Block Validation

Block validation in BPX is composed of two parts: header validation and body validation.

The header validation performs consensus algorithm-related checks, such as proof of space and time, signage points and infusion points, previous block hashes, foliage hashes, and timestamps. Execution payload headers are also checked, if the block contains one. Notably, it does not validate the execution payload body.

Body validation entails sending execution payload to execution client to execute all transactions in EVM.

Validating a beacon block will require access to some blocks in the past, up to a maximum theoretical value of three times the max number of blocks in a slot (3x128=384), but usually only a few are needed. Also, information regarding previous sub-epochs and epochs is needed for validation, as well as the current system timestamp. Implementations can cache only some recent blocks instead of storing all blocks in memory. Beacon client maintains a database of BlockRecords, which contain only the important pieces of block information required for validating future blocks.

Full Sync vs Normal Operation

There are two cases when a beacon client might verify blocks.

1. During a full sync, where the beacon client is trying to catch up to the most recent block, starting from an old block height. In this case, the beacon client is able to download many blocks at once.
2. During normal operation, where the beacon client is caught up to the most recent block, and is only downloading one block every few seconds.

We'll cover both of these cases below.

Full Sync

Full sync is the process by which a beacon client downloads and validates all of the blocks in the blockchain and catches up to the most recent block. Full sync is important, because it allows new nodes to validate that a blockchain is the heaviest - and thus, the currently valid - beacon chain. It allows everyone to come to consensus on the current state, regardless of when they come online, or for how long they go offline.

The method of full sync can vary between implementations, but the high level algorithm is the following:

1. Connect to other peers on the network, by querying the DNS introducer, and crawling the network.
2. Check the current weight of the peak of the peers, and select a few peers to sync from.
3. Download and validate a weight proof, to ensure that the given peak has real work behind it.
4. Download and validate all blocks in the beacon chain, in batches.

Weight proofs are important, because they prevent other peers from lying to us about what the heaviest peak is. They also prevent us from downloading potentially useless data. Once the beacon client is caught up to the beacon chain, it can properly farm, access the execution payloads, etc.

Normal Operation

Normal operation is the process by which a beacon client continuously gossips and receives blocks with other peers, always following the heaviest peak. If our beacon client is at weight 2000, and we see that a peer has a peak at weight 2100, then we fetch that block from the peer. Usually, this is done in two phases:

1. The unfinished block is propagated across the network, along with all information up to the signage point, execution payload, etc.
2. The finished block, which includes infusion point VDFs, is also propagated.

Normal operation is much less CPU-intensive than full sync. Low-power machines like the Raspberry PI 4 should be able to easily continue normal operation.

Block Validation Steps

1. Beacon client receives a beacon block (beacon clients p2p network)
2. Beacon client performs the header validation
3. If the block is transaction block, the transactions in the block are sent to execution client as an execution payload (local RPC connection)
4. Execution client executes the transactions in EVM and validates the state in the execution block header (i.e. checks hashes match)
5. Execution client passes validation data back to beacon client, block now considered to be validated (local RPC connection)
6. Beacon client adds beacon block to head of its own blockchain

---

Revision #3
Created 5 June 2023 14:44:12 by Admin
Updated 27 October 2024 10:33:00 by Admin