Foliage

In the previous diagrams, there is no place for farmers to specify their rewards, since all blocks are canonical. There is also no place to include execution layer data. Everything we have talked about so far is the trunk of the blockchain.

Farmers have no say in how their block is constructed in the trunk, since they must use the exact proof of space, VDFs, and signatures that are specified. In order to include farming rewards, as well as execution payload with transactions, in the system, we must introduce an additional component of beacon blocks called *foliage*.

Trunk: The component of beacon blocks and the beacon chain which includes VDFs, proofs of space, PoSpace signatures, challenges, and previous trunk blocks, and is completely canonical. The trunk never refers to the foliage chain.

Foliage: The component of beacon blocks and the beacon chain which includes specification of where rewards should go, execution chain payload with user transactions, and what the previous foliage block is. This is up to the farmer to decide and is grindable, so it can never be used as input to the challenges.

Re-org: A re-org (or reorganization) is when a node's view of the peak changes, such that the old view contains a block that is not included in the new view (some block has been reversed). Both trunk and foliage re-orgs are possible, but should be rare in practice, and low in depth.

In the diagram below we can see that the foliage is added to blocks to produce an additional chain. This foliage includes a hash of the previous foliage, a reward block hash, and a signature. These foliage pointers are separate from the trunk chain, and are not canonical. That is, farmers could theoretically create a foliage re-org where foliage is replaced, but the exact same trunk (proofs of space and time) are used.

To prevent a foliage re-org, honest farmers only create one foliage block per block. As soon as one honest farmer has added a foliage block, the foliage becomes impossible to re-org beyond that height with the same PoSpace, since that farmer will not sign again with the same PoSpace.

Furthermore, blocks like B3, which come parallel with another foliage block (B2), do not have to sign the previous foliage block, since they do not necessarily have enough time to see it.

> By "coming in parallel", we mean that the second block's signage point occurs before the first block's infusion point.

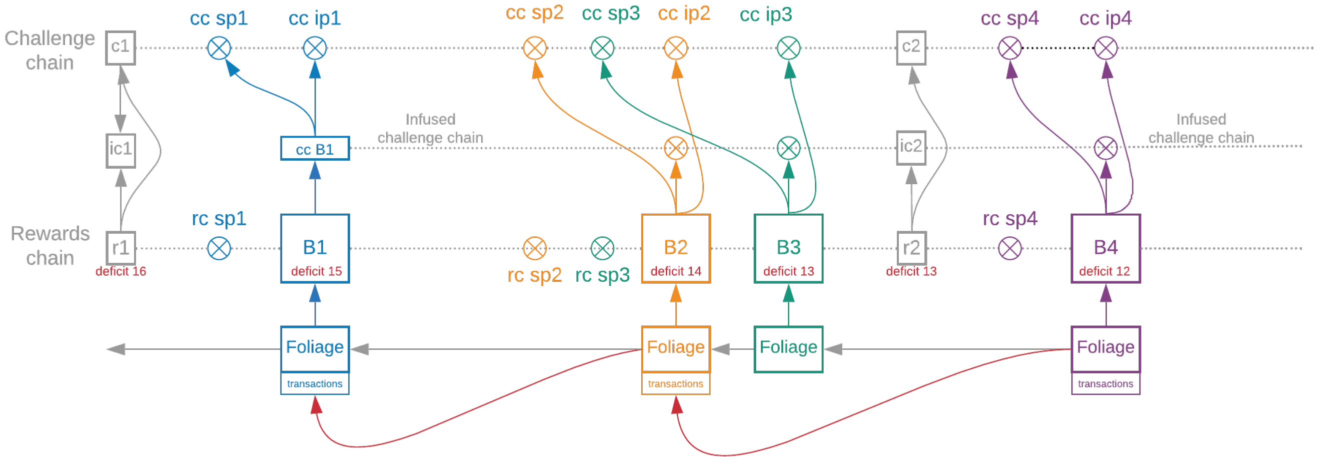
The red arrows in the diagram represent a foliage pointer that is signed by the plot key for the proof of space in that block. The gray arrows represent a hash pointer which is not signed by the plot key (therefore the gray arrow in B3 can be replaced if B2 changes or is withheld). This prevents attacks where B2 modifies their block and forces B3 to re-org.

Blocks which have red pointers are also eligible to create execution chain block, and are therefore called transaction blocks.

A block is a transaction block if and only if it is the first block whose signage point occurs after the infusion of the previous transaction block.

In the diagram, sp3 comes before B2, (a transaction block, and the previous block of B3), so B3 cannot be a transaction block.

The red arrows provide security by burying foliage blocks, but the gray arrows do not. The purpose of the gray arrows is to maintain a linked list in the foliage, and to reduce complexity in implementations. However, blocks with gray arrows pointing to them do get buried in the next-next block.



The block hash is a hash of the entire foliage and trunk block. Re-orgs work on block hashes. Even if we see a chain with the same proofs of space and time, as long as the foliages are different, the blocks will have different hashes.

Note that the farmers of blocks B2 and B3 might both have a chance to create the block, so they must both provide the signed pointer and execution payload. However, any transaction block can be included as a normal block as well, and since B2 and B3 are in parallel, only one of them can make a transaction block.

While all blocks still choose the account address of where their rewards go, those execution chain block do not get withdrawn into the execution chain until the next transaction block.

Transaction Block Time

The average time between transaction blocks is 52 seconds. Several values are required to calculate this average:

- Sub-slot time = 300 seconds
- Signage point time = 64 per sub-slot, or 300/64 = 4.69 seconds
- Average block time = 32 per sub-slot, or 300/32 = 9.38 seconds
- Minimum signage points from current signage point until infusion_iterations is reached = 3
- Minimum time for infusion_iterations to be reached (and therefore, minimum time between transaction blocks) = 3 * (300/64) = 14.06 seconds
- Average signage points until infusion_iterations is reached is slightly more than 3.5 (must wait 3 signage points, plus an average wait of about 50% of the next signage point), or around 3.5 * 4.69 = 16.42 seconds.
- To create a transaction block, infusion_iterations first must be met, and then the next block some seconds afterwards will be a transaction block. The total average time for this to happen is around 26 seconds.

The time between transaction blocks was deliberately chosen for a specific game-theoretic reason: If transaction blocks occurred at the same rate but there were no empty blocks between them, re-orgs and bribery attacks would be easier to pull off.

Additionally, the fact that there are empty blocks between transaction blocks provides several benefits:

- If blocks were created at the same rate and all of them contained transactions, low-power machines such as the Raspberry Pi wouldn't be able to keep up with the chain and therefore wouldn't be supported.
- Empty blocks can also help dampen the effect of the chain slowing down, for example during a dust storm.
- Finally, empty blocks help to smooth farmers' rewards.

Revision #5
Created 5 June 2023 14:43:14 by Admin
Updated 30 October 2024 18:14:15 by Admin