Prerequisite: https://vitalik.ca/general/2018/12/05/cbc_casper.html

I propose a version of the LMD GHOST fork choice rule and an algorithm that makes it easy to tell whether or not a given block is valid under the CBC validity condition of "a block B with parent P is valid if all the evidence included in the state of B points to P being the correct result of the LMD GHOST fork choice rule". This is a step toward making CBC practically implementable.

In general, GHOST-style fork choice rules follow the same pattern: start with some block H

, then let C_1... C_n

be the children of H

. If the list of children is empty, just return H

, if there's one child then set H

to that child and repeat, and if there is more than one child, choose the child that has the strongest support (this could be latest message count, proof of work, or one of many other metrics).

I propose a modification that makes forks always binary (that is, any H

has at most two children): if a given block H

has multiple children, then arrange the children into a virtual tree where we are deciding bit-by-bit on the hash of the child. For example, if H

has three children C_1, C_2, C_3

with hashes 010...

, 011...

and 101...

, then the tree looks as follows:

[

%5BH%5D%20--%3E%20%5BH%2B1%5D%2C%5BH%5D%20--%3E%20%5BH%2B0%5D%2C%5BH%2B1%5D%20--%3E%20%5BH%2B10%5D%2C%5BH%2B10%5D%20--%3E%20%5BH%2B101%5D%2C%5BH%2B101%5D%20--%3E%20%5B.....%5D%2C%5B.....%5D%20-%3E%20%5BC3%5D%2C%5BH%2B0%5D%20--%3E%20%5BH%2B01%5D%2C%5BH%2B01%5D%20--%3E%20%5BH%2B010%5D%2C%5BH%2B01%5D%20--%3E%20%5BH%2B011%5D%2C%5BH%2B010%5D%20--%3E%20%5B.....%20%5D%2C%5BH%2B011%5D%20---%3E%20%5B%20.....%20%20%5D%2C%5B.....%20%5D%20---%3E%20%5BC2%5D%2C%5B%20.....%20%20%5D%20----%3E%20%5BC1%5D

305×903

](https://yuml.me/diagram/scruffy/class/%5BH%5D%20--%3E%20%5BH%2B1%5D%2C%5BH%5D%20--%3E%20%5BH%2B0%5D%2C%5BH%2B1%5D%20--%3E%20%5BH%2B10%5D%2C%5BH%2B10%5D%20--%3E%20%5BH%2B101%5D%2C%5BH%2B101%5D%20--%3E%20%5B.....%5D%2C%5B.....%5D%20-%3E%20%5BC3%5D%2C%5BH%2B0%5D%20--%3E%20%5BH%2B01%5D%2C%5BH%2B01%5D%20--%3E%20%5BH%2B010%5D%2C%5BH%2B01%5D%20--%3E%20%5BH%2B011%5D%2C%5BH%2B010%5D%20--%3E%20%5B.....%20%5D%2C%5BH%2B011%5D%20---%3E%20%5B%20.....%20%20%5D%2C%5B.....%20%5D%20---%3E%20%5BC2%5D%2C%5B%20.....%20%20%5D%20----%3E%20%5BC1%5D)

Note that this means that if any of the three children has more support than the other two combined (ie. at least 51%), then it will win in the GHOST fork choice as before, but if none of the three strongly dominate then the result could be different, for example if C_1

has score 4, C_2

has score 3 and C_3

has score 5, then under simple LMD GHOST C_3

would win, but in modified GHOST C_1

would win.

Conjecture: this difference does not actually matter much in real life, and does not make any attacks easier, because any attack that takes advantage of the new structure where C_1

and C_2

support each other could have also succeeded in a three-way fork where H

had two regular children D

and C_3

, where C_1

and C_2

were children of D

.

Now, let's see what this enables. We assume that the blockchain state keeps track of the following data structure: for each validator, what is the height up to which the block that their latest attestation pointed to agrees with the current chain? We store this as a number: 256 * actual_shared_height + number_of_common_bits

, where number_of_common_bits

is the number of initial bits in common between the hash of the chain's actual next block and the hash of the ancestor of the message signed by the validator at the lowest height that is not shared with the chain. That is, if the actual chain has head B_1

and a validator's latest message is B_2

, and B_1

and B_2

have common ancestors up to height n

, and at height n+1

the ancestors of B_1

and B_2

agree in their first k

bits, then the "virtual agreement height" stored is n * 256 + k

.

We also maintain a mapping: virtual height -> how many validators whose latest message agrees up to this virtual height. We store it in a sum tree, so we can now determine in O(log(n))

time how many validators agree with the main block up to at least some given virtual height (we call this agreeing[h]

). To determine if a block is valid, we now need only verify one property: that there is no virtual height h

where agreeing[h+1] < (agreeing[h] - at[h]) * \frac{1}{2}

, where at[h]

is the set of validators that agree up to exactly h

and whose latest signed message is of a block at exactly that height. If it is the case that there is no such h

, then the current block's parent actually is the LMD GHOST evaluation of the messages the block knows about.

Because agreeing[h]

declines monotonically (as it represents how many validators agree up to at least that point), we can do this in O(log^2(n))

time: binary-search the max height h

where at least half of all validators agree, verify that agreeing[h+1] \ge (agreeing[h] - at[h]) * \frac{1}{2}

, then binary-search the max height h

where at least a quarter of all validators agree, and so forth until you reach the head.

Notice that the fact that each node in the virtual tree can only have at most two children is necessary for this technique to work, as it ensures that verifying that agreeing

does not “step down too quickly” is necessary and sufficient for verifying bitwise GHOST compliance.