Edit 2018.03.03: figured out the best way to do slashing conditions.

I propose a rule change to the Casper FFG contract: instead of an epoch’s length being a fixed 50 blocks, 50 is merely the base

epoch length, and an epoch’s actual length is calculated at epoch start time according to the following rule:

- If the last epoch finalized a checkpoint, the epoch length is 50 blocks.
- Otherwise, the epoch length is double the length of the previous epoch.

This has the advantage that it allows Casper to keep finalizing checkpoints even under highly adverse conditions, where latency is extremely high, as eventually the epoch length will catch up to the latency, allowing a checkpoint to be finalized.

Instead of the leak penalty being based on LFE ^ 2, the penalty will now be 4 ^ LFE, ensuring that it continues to be quadratic in the important variable, time

since finality.

We change slashing conditions as follows. We continue to define epoch number as floor(epoch_start_block_number / 50)

; when the epoch length is longer than 50, we simply skip epoch numbers. Every vote specifies not just the current epoch, but also the next epoch minus one (eg. if the current epoch is 108 and the next is expected to be 112, then the vote specifies (108, 111)). We refer to these two values as target start

(ts) and target end

(te).

We leave the NO_SURROUND condition unmodified, but we replace the NO_DBL_VOTE condition with NO_INTERSECT:

A validator cannot sign two votes whose target ranges intersect; that is, a validator cannot sign V1 and V2 where V1.te >= V2.te >= V1.ts >= V2.ts

.

The proof of safety is the same: if there are two finalized checkpoints on separate chains A and B, with chain B having a higher target end, then we can keep walking back through that chain and it is not possible to avoid either intersecting or surrounding at least one of the target ranges of A.