I wanted to sketch a possible tweak to Casper FFG, related to mypost last year about the possibility of removing the "source" param from the votes ("attestations") validators cast. (See also four old Twitter threads: Casper walkthrough, Casper in one tweet, earlier walkthrough, @danrobinson clarifying.) I was spurred to revisit the topic by @djrtwo's talk at Devcon Osaka, where the definition of finalization seemed messy to me.

The main motivations for this work are simplicity/clarity, especially of finalization, and robustness under network lag. See the final section below for six potential benefits.

If there's a better forum for discussing this stuff, please point me at it! I'm also happy to chat…

Sketch of proposal

1. As a validator, you're always casting two types of votes: the block B you're "voting for", and the earlier block F you're "voting-to-finalize".
2. Time is broken down into 6-second slots. In each slot, you can update your B and/or F, or (by default) leave either or both unchanged from the previous slot.
3. You can only vote-to-finalize a block F in slot s if:

a) block_height(F) ≥ block_height(F'), your previous vote-to-finalize; and

b) F received 2/3 of the votes in some slot sF

< s, which you specify in your vote-to-finalize; and

c) you've voted for F or descendants of F in every slot s', for sF

< s' ≤ s.

1. You can only vote for a block B in slot s if B is a descendant of either:

d) your current F, or

e) some other block J, which received 2/3 of votes in some slot sJ

, where sF

< sJ

< s.

Slashing rules

These correspond to rules a)-e) above. You get slashed if any of these apply:

- a): block_height(F) < block_height(F').
- b): >1/3 of votes in slot sF

were for blocks other

than F.

- c): at some slot s' (sF

< s' ≤ s), you voted for a block B that is not

F or a descendant of F.

- d) and e) combined:
- B is not a descendant of F, and
- for every slot sJ

(sF

< sJ

< s), >1/3 of votes were for blocks that aren't

ancestors of B.

- B is not a descendant of F, and
- for every slot sJ

(sF

< sJ

< s), >1/3 of votes were for blocks that aren't

ancestors of B.

Definition of finalization

A block F is finalized

as of some slot s, if 2/3 of votes-to-finalize in s are for F or descendants of F.

Safety and liveness

I believe safety and liveness follow from the rules and definition above. I can try to sketch proofs if anyone's interested…

Why do it this way?

Some possible advantages over existing FFG (as I understand it!): (see also the related list inlast year's post)

1. The definition of finalization above is simpler than the definition I understood from@djrtwo's talk, with its "k=1, k=2, k=3" cases.
2. Having each validator keep track of two things - the B it's voting for, and the F it's voting-to-finalize - is to me more intuitive than FFG's "source" and "checkpoint edge".
3. Time (slot number) and block height are distinguished. You can vote repeatedly for the same block (at the same height) in successive slots: by default your B and F stay unchanged from slot to slot. (It might make sense to require votes to refer to the previous vote - a "votechain"! - to prevent validators from filling in skipped votes later.)
4. A block can be finalized many slots after it was justified (got 2/3 of votes), whereas I believe in regular FFG it must be finalized the immediate block after. This could be useful, eg, in situations where a laggy network means validators are having trouble updating their votes every 6 seconds.
5. The slashing rules above are tighter than FFG's: FFG slashes more cases than are required to guarantee safety and liveness. (See also the @JustinDrake and @dlubarov notes in the post linked above.)
6. Each validator's "vote-to-finalize" blocks increase monotonically in height: you're prohibited from linking your current vote to an arbitrarily old justified block. This might simplify some things.