This post is co-authored by Carl, Caspar and Francesco. Special thanks to Danny and Barnabé for input and discussions.

This proposal involves understanding two attack vectors (balancing attacks and ex ante reorgs). A recent talk provides the quickest intro to both of these.

Why change?

Before jumping into the proposed change to the fork choice rule let us briefly outline why this may be desirable in the first place.

There is the balancing attack

, which is discussed here, here and here. There already is an open PR that mitigates this attack vector via proposer weight boosting. You can find a high-level write up of the idea here.

However, we recently found a new attack vector: ex ante reorgs

. It is written up properly in a paper here and there is a recent talk on the topic here. The proposer boosting mentioned above fixes most of the ex ante reorging scenarios already, but not all.

It's worth noting that the ex ante reorg attack can make use of strategies from the balancing attack to improve its effectiveness. This layered attack is also of relevance for this post.

Boosting does not mitigate all ex ante reorgs

Despite boosting or in fact because of boosting there is a way to execute a variant of an ex ante reorg: The analysis of the authors of the balancing attack, says that the proposer weight W_p

should be ~80\%

of committee weight W

for strongest security (liveness + safety; liveness being attacked and the limiting factor here). Assuming this, a 7\%

-adversary can do the following: Propose a hidden block in n+1

, vote for it with 7\%

adversarial attestations from that slot. Thus, the honest block n+2

builds on block n

(n+2

sibling to n+1

); 93\%

honest vote for it (because of proposer boost), adversary again uses their 7\%

attestations from slot n+2

to vote for n+1

. Block n+3

is adversarial, builds on n+1

. Now the chain of block n+1

has 7\%

from slot n+1

, 7\%

from slot n+2

, and 80\%

from the n+3

proposer boost which equals 94\%

, which in turn is more than the 93\%

from the honest committee of slot n+2

(2*7\% + 80\% = 94\% > 93\%

). As a result honest validators switch over and vote for n+3

, and n+2

is forked out.

In short, proposer weight boosting mitigates cheap reorgs, but cannot prevent attacks by adversaries with very large stake.

Essence of the problem

For more context please watch the talk and/or read this paper on ex ante reorgs

.

To understand why the proposal helps to prevent the aforementioned attacks it may be helpful to understand on a high level how the adversary times his blocks and attestations to exectue these attacks.

When balance attacking

, the adversary tries to split honest validators into different views on what the current head of the chain is. Ideally, the adversary can convince half of the committee to vote for their block and the other half of the committee to vote for the honest block. Why? The honest committee's votes cancel each other out. Therefore with very few adverserial votes the adversary can keep an ex ante reorg going and eventually tip the chain into the direction of their liking, or just prevent finality by making it impossible for either branch to get the required 2/3 majority of FFG votes. Adversary can split committees in half by catching honest proposal early and then targeting p2p network in coordinated fashion: Release 'sway votes' (attestations) such that half of the committee hears them before they hear the latest block and run their fork choice including the adversarial sway vote (and hence vote for adversarial block); the other half of the committee hears the sway vote after they have heard the honest block already (and hence run their fork choice already and vote for honest block).

When concluding an ex ante reorg

, the adversary tries to convince the majority of a committee to vote for their block such that the next honest proposer sees the adversarial block as the current head of the chain and thus builds on it. This then concludes the reorg.

Proposed change

Time thresholds that can be used to split nodes into different views are the key to the previously discussed attacks. Accordingly, the proposed fix eliminates the single point in time that the adversary utilizes to target their release of a block/sway vote(s). This is done by giving proposers and attesters different deadlines for considering attestations, but allowing honest attesters to synchronize with the proposer's view (attestations included on-chain).

The goal of this proposal is that byzantine behaviour alone, in the absence of latency, should not be sufficient to force honest validators into conflicting views when exercising their proposal and attestation duties. The attacks mentioned are all a symptom of our current fork choice rule not having this property:

- Attesters all having the same view at attestation time prevents the continuation of a balancing attack
- Attesters having the same view as the proposer prevents ex ante reorgs

The fix works as follows:

[

1320×756 30.8 KB

](https://ethresear.ch/uploads/default/original/2X/0/0d58916498ba5daa22589f21805ff4c01859f363.png)

We introduce a message deadline D = 10

seconds through slot n

before which everyone considers consensus messages (attestations and blocks: all that is relevant to the fork-choice; from here on just called messages) as they normally do. Let A

be a validator's local message view. After D

, validator for block n+1

store their messages in a cache, A'

.

The proposer in slot n+1

continues to consider messages right until the point they produce their block B_{n+1}

.

The committee in n+1

runs their fork choice based on the attestations before D plus all attestations the proposer included in B_{n+1}

. Only after they have attested, validators merge in their cached messages (A = A \cup A'

) and reset the cache (A' = \emptyset

).

In the event of a missed block B_{n+1}

(no block seen after 4s into n+1

), attesters run their fork choice based on all the messages they've seen so far, including those after D

. In other words, if no block has been heard 4s into a slot attesters merge all cached messages before running fork choice to attest.

What is happening here

Validators pause their fork choice at the message deadline and the proposer helps synchronize their view of the chain after this point. This way, committee members still develop their own view of the network, but the proposer has the ability to include additional attestations in the last 2s of the slot. These additional attestations are also considered by the attesters, allowing everyone to synchronize (and thus all agree on sway votes!).

Why it works

Assuming max 2s latency, a timely and honest proposal would be attested to by all honest attesters. A little proof: max 2s latency means that all messages sent before D

are seen by the proposer before running the fork choice, and added to the proposal. Any message sent after the deadline would by definition be received after the deadline, and thus ignored by all attesters unless they see it in the proposal. Since a timely proposal is guaranteed to be seen in time by attesters, also due to the 2s max latency, honest attesters utilize a message in their fork choice if and only if the proposal includes it. Therefore, all of the fork choices run have the same output as the proposer's.

Previously the adversary would try to catch an honest block early on the p2p and then either release sway votes to make a majority of the honest committee vote for the adversarial block (to conclude a reorg), or split the honest committee into different views to keep the reorg going (roughly half vote for adversarial block and the other half for honest block).

Basically the attacks require the adversary to hear a block early and only then release sway votes. With the new fork choice releasing sway votes after observing the honest block will not work, because they will only ever be considered until after they attested already.

On a high level, there are two things that the new fork choice rule prevents: balancing attack (splitting validators into different views) and concluding reorg attacks (winning over majority of honest validators by releasing block/attestations after observing honest block).

This proposal works against this by enabling the proposer to (mostly) unify the views of the attesters (with their own view) should an attacker try sway a committee.

Split votes at time D

If an attacker releases (previously unseen) attestations (A_{split}

) around time D

in an attempt to split the attestation views A

of committee members, they will temporarily succeed. (Some attesters will have their A

include A_{split}

while others will not.) However, the proposer for B_{n+1}

has an additional 2 seconds before they are due to propose in which time they should see A_{split}

and include these attestations into B_{n+1}

. (Assuming the proposer's network latency is bounded by 2 seconds, see more on this in latency section)

An honest proposer builds their block on what they perceive to be the head of the chain which, by definition, is the result of running the fork choice over all the attestations they have seen so far.

Therefore all committee members who follow the fork choice described above will include A_{split}

in their fork choice too and the attack has failed.

Split votes when B_{n+1}

is released

Should an attacker try release A_{split}

around the time B_{n+1}

is being proposed, whether the proposer sees the votes or not, honest committee memebers will agree with the proposer as they base their opinion on A \cup B_{n+1}

and not A'

.

Preventing ex-ante re-orgs

Ex-ante re-orgs work by having a malicious proposer release block B_{n}

(along with attestation(s) for B_n

) late enough that the proposer in B_{n + 1}

doesn't see B_n

, but committee members in B_{n + 1}

see B_n

(with the attacker's attestation(s)) which they then attest to as the winner of their fork choice.

This style of attack is not possible under this new fork choice proposal as the attacker would have to either release B_n

before D

and the proposer of B_{n+1}

would see it in time to build on top of it or it is in the time interval between D

and end of slot n

in which case only messages received before D

or included in B_{n+1}

are considered.

No more power than necessary

While the proposer now has additional sway over the views of the committee members, it is highly bounded as they can only add extra information that the attesters might not have seen, as opposed to gaining extra voting power in the fork choice. This greatly minimizes the attack surface for malicious proposers; especially when compared to something like proposer boosting.

Appendix

Musings on latency

First up, yes that's right, we're assuming good latency conditions.

So what if latency is >2s and the adversary's sway vote(s) that split honest validators into different views is not heard by the next proposer before proposing? This is bad, because attesters would not be able to synchronize on a canonical view and hence be split into different views. But importantly, for an adversary to continuously achieve this the honest proposer should never hear the sway vote before the attestation deadline D

either. Point being that if the proposer isn't specifically targeted, then with ~50% probability the proposer should hear the sway votes at cutoff time D

.

Alternatively the attacker could try to target proposers specifically and DoS attack them such that the sway votes don't find their way on-chain immediately and hence views are split. While this does work, this is a separate problem in itself that is actively being worked on too. It is worth noting though that this DoS'ing would need to succeed on each respective proposer slot after slot… Additionally, a single slot with "normal" latency is sufficient to break the attack again.

It's a soft fork

update

One of the upsides of this proposal is that it is backwards & forwards compatable with nodes who don't follow this new fork choice rule.

No changes for proposers

The behaviour for proposers in the current fork choice and this proposal are the same.