TL;DR:

Clearly it is impossible to be certain that a chain's head can never change, unless that head is finalized. Both latency/network splits and minority attackers can cause short-term reorgs of non-finalized blocks in Casper FFG. But by observing some simple rules, we can still get very quick confirmation when network conditions are good, even for non-finalized blocks, under stronger assumptions (high network synchronicity).

Introduction

The Casper FFG fork choice rule is optimized for fast finality: Under good conditions, an epoch can usually be finalized within the two consecutive epochs, and a finalized epoch cannot be reverted unless (a) at least 1/3 of validators are dishonest and colluding and (b) 1/3 of the total stake is burned, causing a huge loss to the attacker.

This gives a very high degree of safety for any finalized epoch. But having to wait 64-96 blocks (between 12.8 and 19.2 minutes) for finalization is a very long time for many purposes. As an example, kraken currently requires 30 blocks confirmation for ETH deposits, or 7 minutes – so waiting for finality would actually be a regression in terms of UX.

It is not possible to make any absolute guarantees on any non-finalized chains. To get that property, you need byzantine fault tolerance which is what finalization in FFG does. And we have seen several possible attacks on the fork choice, that can be executed by attackers with <1/3 of the stake:

The bouncing attack

Full description: Analysis of bouncing attack on FFG

Short summary: Blocks that are "almost justified" but on a different chain than the latest justified block can suddenly disrupt the chain when they become justified.

Short-term reorgs (and finality delays)

Full description: https://econcs.pku.edu.cn/wine2020/wine2020/Workshop/GTiB20_paper_8.pdf

Short summary: Because the "head" votes of the current fork choice rule only vote for a block, and not also a slot, an attacker can withhold a number of blocks and while the other validators will keep voting for the last available block, these votes do not count toward the "empty" chain. Attackers can thus revert a number of blocks by withholding their blocks and their own attestations.

Current fixes

Some resistance to the bounce attack was addedhere.

There is a proposed fix for short term reorgs in the form of the(block, slot) fork choice rule. This means that whenever a block is missing, the current attesters vote for that specific slot being empty, as well as the previous block, and thus an empty chain also gets quick confirmation. This fix has a major downside: Under poor network conditions, many blocks (and even all blocks if latencies are generally >4s) can be confirmed as empty, and we would just be building an empty chain.

Suggested approach

Past approaches at fixing these problems (at least from the safety perspective) have focused on making the current head as sticky as possible. However, there is also another point of view: All of the above attacks are actually clearly visible on chain, if you pay more attention to it rather than just finding the head. The attacker has to withhold some attestations in order to be able to switch the fork choice to another head.

Let's make the following assumption: The network is currently fully synchronous:

- All the messages that we are seeing have been seen by all network participants
- An attacker may withhold messages, however, nobody (except the attacker) would have seen the withheld messages until they are released, when everyone sees them
- 2/3 of validators are honest and will keep voting according to the head they are seeing in the fork choice rule

We want to be assured that the current head (or some previous block on the current fork choice rule) will not be reverted by any of the above attacks. This is how we get this assurance:

1. Any finalized epoch clearly cannot be reverted, so start from the finalized epoch

2. Justified epochs can be reverted, but only if a later epoch on another fork gets justified. So we must protect against this possibility:

3. Let’s call the latest justified epoch e

. Then we verify that, for each epoch e'>e

, more than 1/3

of all validators voted for a descendant of the justified epoch e

as their FFG target. This ensures no epoch on an alternative fork can get 2/3

without validators equivocating.

- This will likely fail on the current epoch, which does not have enough attestations. Instead, verify that at at least 1/3

of the currently available votes (i.e. those with slot < current_slot) are for a descendant of e'

1. Let’s call the latest justified epoch e

. Then we verify that, for each epoch e'>e

, more than 1/3

of all validators voted for a descendant of the justified epoch e

as their FFG target. This ensures no epoch on an alternative fork can get 2/3

without validators equivocating.

1. This will likely fail on the current epoch, which does not have enough attestations. Instead, verify that at at least 1/3

of the currently available votes (i.e. those with slot < current_slot) are for a descendant of e'

1. Now, we turn to the head votes. We need to ensure that the fork choice cannot be reverted. We do this by once again starting from e

, checking that at every slot, the chain cannot be reverted by withheld votes. This means at every slot height s

that comes after epoch e

, we are going to check that at least 50% of all

the following attestations have voted for the current chain.

The tricky part of this is empty slots, which are exactly the crux of the short reorg attacks, because the current attestations do not vote for empty slots, only for their ancestor. When you evaluate the rule at an empty slot, you only count those attestations that vote for it as an empty slot

. This means that only attestations that vote for a filled slot that comes after the empty slot count; those that vote for its filled ancestor do not fix the empty slot as an empty slot, and should be counted as abstentions from the vote. Let’s illustrate this with some examples:

Example 1

Since Block C and its attestations are withheld by the attacker, we only see Block B and think that it is the head. However, when we try to apply our rule, it will fail when evaluated at the empty slot 1: The honest votes for slot 1 actually vote for Block A, because no new block was available yet.

Let’s say there are exactly 3200 validators so each slot gets a maximum of 100 attestations. Then it means that a total of 50 attestations (the honest attestations for slot 2) count towards the empty slot 1, but there are a total of 150 possible attestations: 50 at slot 1 (that is because 50 of the honest attestations already voted for slot 0, and thus can’t be withheld to vote on another chain, they are thus to be treated like abstentions) and 100 at slot 2.

We thus only have 33% of the possible attestations at slot 1 attesting for the empty slot, and the head at Block B is not safe according to our rule.

This can be seen from the red (withheld) attacker chain, that gets a total of 100 attestations at slot 1, and will thus beat the current head in the fork choice rule when it is released. If you want to be safe from reorgs you should wait for more confirmations and have to stick to an earlier head that can be considered safe (maybe block A in this example).

Example 2

In this alternative example, the attacker has only 25% stake, and when we evaluate the rule at the empty slot 1 of the visible chain, we get that 75 out of a total of 125 (25 at slot 1 and 100 at slot 2) attestations are voting for an empty slot at slot 1, for a total of 60%. The empty slot can thus be considered secure assuming synchronous networking conditions, and Block B is a safe head under these assumptions.

How to modify this under suggested changes of the fork choice rule

This proposal is for the current fork choice rule implemented in Eth2 at genesis. Currently several modifications to that fork choice are considered.

Going to the (block, slot) fork choice rule will make this problem easier, but it is still a wise choice to count the number of attestations of a head and its ancestors before trusting it. However, the (block, slot) fork choice rule, at least in its most naive form, imposes a 4s synchronicity condition in order to make any progress at all and is thus not great for liveness.

An interesting alternative that was suggested recently is to use the FFG head vote as another LMD ghost message, to remedy the problem of epoch boundary blocks reverting that was pointed out here. Under this suggestion, the full fork choice would be a hierarchy of 4:

1. Find latest finalized epoch
2. Find latest justified epoch
3. Find highest epoch boundary block according to LMD GHOST on FFG target votes
4. Find head block according to LMD GHOST on head votes

This rule has the effect of enforcing the (block, slot) fork choice rule only on epoch boundary blocks, for much better liveness (latencies of up to one epoch can be tolerated). If this is included, we have to change our rules here so that at least 50% have voted for each epoch, instead of just 1/3, in order to prevent chain reorgs on the basis of the LMD GHOST on FFG target rule (rule 3).

Conclusion

It may be tempting to try to make sure that reorgs never happen, and this is what the (block, slot) fork choice rule attempts to do. However, this is at the expense of liveness and a solution that provides good liveness is not yet known (and may indeed be impossible). However, we can see in this post that we can easily find out whether the current head is susceptible to a minority reorg attack. Under good networking conditions, this will almost never be the case, and thus the head can be trusted; otherwise, the “trustworthy” head may trail behind the current head by a few blocks. If exchanges and other high-value users of blockchains use this method for fast confirmations, they can get security vastly exceeding that of PoW chains without having to wait for finality.