Two Attacks On Proof-of-Stake GHOST/Ethereum

Joachim Neu jneu@stanford.edu
Ertem Nusret Tas nusret@stanford.edu
David Tse dntse@stanford.edu

March 2, 2022

Abstract

We present two attacks targeting the Proof-of-Stake (PoS) Ethereum consensus protocol. The first attack suggests a fundamental conceptual incompatibility between PoS and the Greedy Heaviest-Observed Sub-Tree (GHOST) fork choice paradigm employed by PoS Ethereum. In a nutshell, PoS allows an adversary with a vanishing amount of stake to produce an unlimited number of equivocating blocks. While most equivocating blocks will be orphaned, such orphaned 'uncle blocks' still influence fork choice under the GHOST paradigm, bestowing upon the adversary devastating control over the canonical chain. While the Latest Message Driven (LMD) aspect of current PoS Ethereum prevents a straightforward application of this attack, our second attack shows how LMD specifically can be exploited to obtain a new variant of the balancing attack that overcomes a recent protocol addition that was intended to mitigate balancing-type attacks. Thus, in its current form, PoS Ethereum without and with LMD is vulnerable to our first and second attack, respectively.

1 Introduction

The currently proposed Proof-of-Stake (PoS) Ethereum consensus protocol [7, 5, 6] is constructed from an application of the finality gadget Casper FFG [2] on top of the fork choice rule LMD GHOST, a variant of the Greedy Heaviest-Observed Sub-Tree (GHOST) [20] rule which considers only each participant's most recent vote (Latest Message Driven, LMD). Subsequently, we refer as *validators* to participants with stake that allows them to vote as part of the protocol. A slightly simplified and analytically more tractable variant of the proposed PoS Ethereum protocol is given by the Gasper protocol [3]. The protocol is recapitulated on a high level in Section 2.

This report continues a sequence of earlier works [16, 12, 13, 19, 18, 4, 10, 11, 17] that highlighted security vulnerabilities in PoS Ethereum and the constituent protocols Casper FFG and LMD GHOST. More specifically, we report two new attacks.

The first attack, called *avalanche attack* and described in Section 3 (cf. [14]), suggests a fundamental conceptual incompatibility between PoS on the one hand and the GHOST fork

JN and ENT contributed equally and are listed alphabetically.

choice paradigm on the other hand. In a nutshell, 'orphaned' so called 'uncle blocks' off the canonical chain still influence the fork choice in GHOST. In Proof-of-Work (PoW), for which GHOST is secure [9], this does not create a problem because the number of uncle blocks which the adversary controls and by which it can influence the fork choice is bounded by the PoW mechanism. PoS, on the other hand, allows the adversary to equivocate, *i.e.*, for each block production opportunity the adversary can produce an unlimited amount of equivocating blocks extending different parent blocks of the block tree. As a result, GHOST's way of accounting for uncle blocks in fork choice gives the adversary much more influence over fork choice in PoS than in PoW, casting doubt over whether GHOST should be used with PoS at all.

The LMD aspect of current PoS Ethereum interferes with a naive application of the avalanche attack to PoS Ethereum. However, our second attack, described in Section 4 (cf. [15]), shows how the LMD feature specifically can be exploited to obtain a new variant of the balancing attack [12, 13]. This new attack also overcomes 'proposer boosting' [1], a recent protocol addition that was intended to mitigate balancing-type attacks.

Thus, in its current form, PoS Ethereum without LMD is vulnerable to our first attack (avalanche attack), and PoS Ethereum with LMD is vulnerable to our second attack (new LMD-specific variant of earlier balancing attack).

2 Proof-of-Stake Ethereum Consensus Protocol

Let Ledger_i^t denote the transaction ledger output by honest validator i at time t. *Security* of a consensus protocol such as PoS Ethereum is comprised of *safety* and *liveness*:

- **Safety:** For any given times t, t' and honest validators i, j, either Ledger_i^t is a prefix of $\mathsf{Ledger}_j^{t'}$, or vice versa. Less formally, the ledgers output by two honest validators at two points in time are consistent with each other.
- **Liveness:** If a transaction tx is received by all honest validators by some time t, then tx appears in $\mathsf{Ledger}_i^{t'}$ for any time $t' \geq t + T_{\mathsf{conf}}$ and for any honest validator i, where T_{conf} is the protocol's *confirmation time*. Less formally, transactions get confirmed in honest validators' ledgers with at most T_{conf} time delay.

We first describe a simplified version of PoS Ethereum without the LMD rule, reduced to the relevant fork choice mechanics. Although we present a self-contained description, familiarity with the protocols GHOST [20] and Gasper [3], as well as with the beacon chain's fork choice specification [7] and proposal weights[2] [1] are useful for a deeper understanding of this section.

In the consensus protocol (also called Committee-GHOST), time proceeds in synchronized slots of duration 2Δ, since it is assumed that message delay between honest validators is bounded by Δ. For each slot, one *proposer* and a *committee* of W validators are drawn independently and uniformly at random (without replacement for the committee members) from the N validators. We say a slot is honest or adversarial if the corresponding block proposer is honest or adversarial, respectively. The following fork-choice rule is used in the view of validator i at slot t to determine a canonical block and its prefix of blocks as the *canonical chain*:

[2] `https://github.com/ethereum/consensus-specs/pull/2730`

- Starting at the genesis block b_0, sum for each child block b the number of unique valid votes for that block and its descendents. Here, unique means counting only one vote per committee member in each slot $s < t$. Valid means that the vote from a committee member of slot $s < t$ was cast for a block produced in a slot $s' \leq s$. Ties are broken by the adversary.

  Add a proposal boost of W_p to the score of b if b or one of its descendants is a valid proposal from the current slot t. Here, valid means that the block is not from a future slot, was produced by the proposer of slot t, and that proposal time slots along block chains are strictly increasing.

- Pick the child block b^* with highest score (cf. GHOST rule [20]), breaking ties adversarially.

- Recurse ($b_0 \leftarrow b^*$) until reaching a leaf block. Output that block.

At the beginning of each slot, the slot's proposer determines a block using the fork-choice rule and extends it with a new proposal. Half way into each slot (*i.e.*, Δ time after the proposal and after the beginning of the slot), the slot's committee members determine a block using the fork-choice rule in their local view and vote for it. The unit of time is a time slot. A block from slot t and its prefix are confirmed and output as the ledger if and only if at time $t + T_{\mathsf{conf}} + \frac{1}{2}$ (*i.e.*, at the time of voting in slot $t + T_{\mathsf{conf}}$), the block is in the chain determined by the fork-choice rule in the view of the respective honest validator. Here, T_{conf} is the *confirmation time*.

For simplicity, we assume a fraction β such that (a) for any given slot the probability of the block proposer being adversarial is at most β, and (b) the fraction of adversarial validators in any committee is at most β throughout the protocol's execution.

The LMD rule modifies the above protocol as follows. Every validator keeps a table of 'latest votes' received from the other validators, in the following manner: When a valid vote from a validator is received, then the table entry for that validator is updated, if and only if the new vote is from a slot *strictly later than* the current entry. Hence, when a validator observes a slot-t vote from the committee member i of some slot t, it records this vote and ignores all subsequent slot-t' votes for $t' \leq t$ by i. Thus, if a validator receives two equivocating votes from the same validator for the *same* time slot, the validator counts only the vote received earlier in time.

3 Avalanche Attack on Proof-of-Stake GHOST

We describe a generic attack on PoS GHOST variants. This points to conceptual issues with the combination of PoS and GHOST, awareness of which might be of interest beyond PoS Ethereum fork choice design. PoS Ethereum, as it stands, is not susceptible to this attack (due to LMD, which comes with its own problems, see Section 4).

We assume basic familiarity with GHOST [20] and Gasper [3]. For details, we refer the interested reader to the beacon chain's fork choice specification [7] and earlier attacks [12, 13].

3.1 High Level Description

The *avalanche attack* on PoS GHOST combines *selfish mining* [8] with *equivocations*. The adversary uses withheld blocks to displace an honest chain once it catches up in sub-tree weight with the number of withheld adversarial blocks. The withheld blocks are released in a flat but

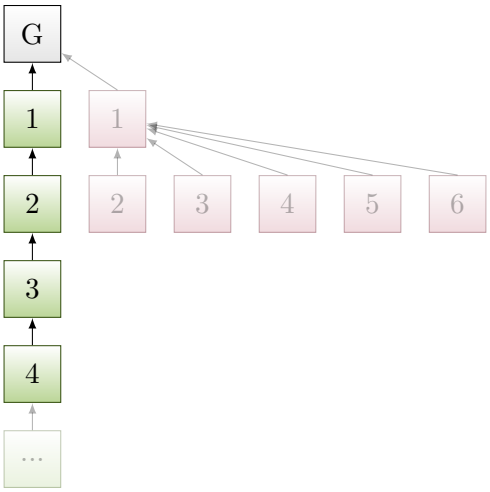

Figure 1: First, the adversary withholds its flat-but-wide sub-tree of $k = 6$ withheld blocks, while honest nodes produce a chain. (Green and red indicate honest and adversarial blocks respectively, and the numbers on blocks indicate which block production opportunity of honest/adversary they correspond to.)

wide sub-tree, exploiting the fact that under the GHOST rule such a sub-tree can displace a long chain. Only two withheld blocks enter the canonical chain permanently, while the other withheld blocks are subsequently reused through equivocations to build further sub-trees to displace even more honest blocks. The attack exploits a specific weakness of the GHOST rule in combination with equivocations, namely that an adversary can reuse 'uncle blocks' in GHOST, and thus such equivocations contribute to the weight of multiple ancestors. This casts doubt over whether GHOST should be used with PoS at all.

We also provide a proof-of-concept implementation for vanilla PoS GHOST and Committee-GHOST.[3] By 'vanilla PoS GHOST', we mean a one-to-one translation of GHOST [20] from proof-of-work lotteries to proof-of-stake lotteries. In that case, every block comes with unit weight. By 'Committee-GHOST' we mean a vote-based variant of GHOST as used in PoS Ethereum, where block weight is determined by votes and potentially a proposal boost [1]. Subsequently, we first illustrate the attack with an example, then provide a more detailed description, and finally show plots produced by the proof-of-concept implementation.

3.2 A Simple Attack Example

We illustrate the attack using a slightly simplified example where the adversary starts with $k = 6$ withheld blocks and does not gain any new blocks during the attack. In this case, the attack eventually runs out of steam and stops. (In reality, the larger the number of withheld blocks, the more likely the attack continues practically forever, and even for low k that probability is not negligible.) Still, the example illustrates that $k = 6$ blocks are enough for the adversary to displace 12 honest blocks—not a good sign.

First, the adversary withholds its flat-but-wide sub-tree of $k = 6$ withheld blocks, while honest

[3]Source code: `https://github.com/tse-group/pos-ghost-attack`

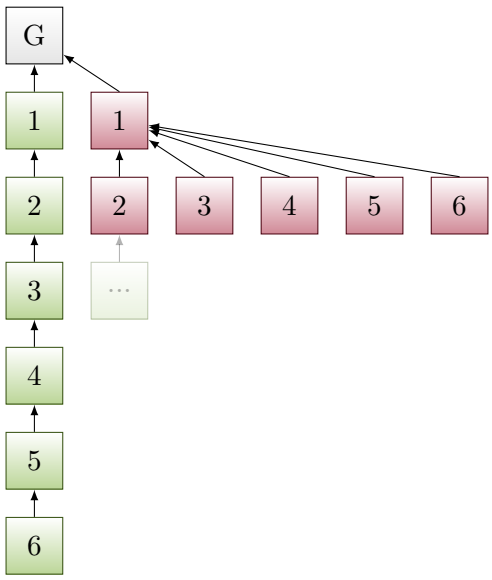

Figure 2: Once honest nodes build a chain of length $k = 6$, the adversary releases the withheld blocks, and displaces the honest chain.

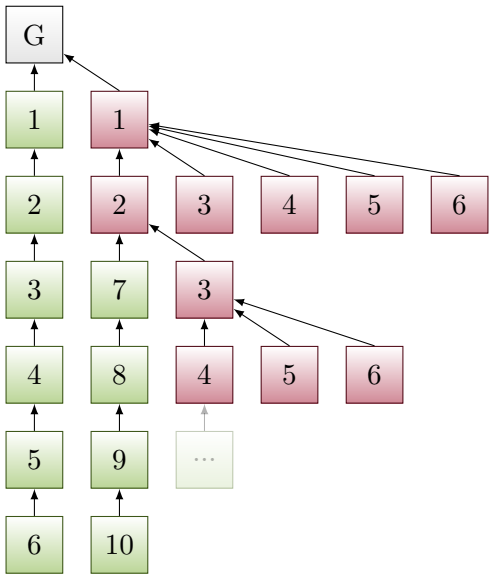

Figure 3: Note that the adversary can reuse its blocks $3, 4, 5, 6$. Honest nodes build a new chain on top of $2 \rightarrow 1 \rightarrow$ Genesis. Once that new chain reaches length 4, the adversary releases another displacing sub-tree.

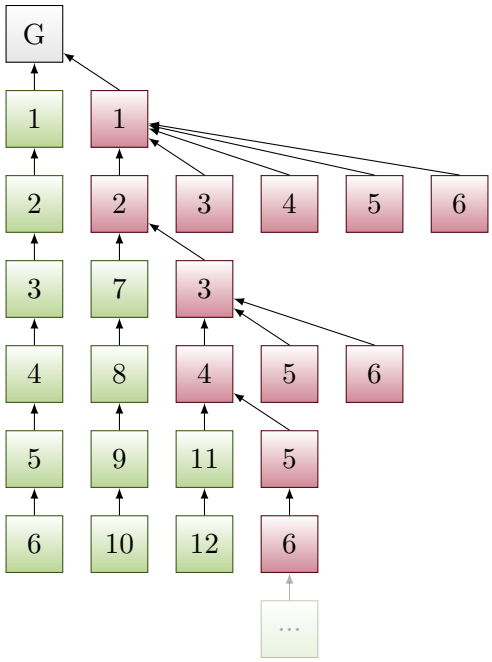

Figure 4: Finally, note that the adversary can reuse its blocks 5, 6. Honest nodes build a new chain on top of $4 \to 3 \to 2 \to 1 \to$ Genesis. Once the new chain reaches length 2, the adversary releases the last displacing sub-tree.

nodes produce a chain (cf. Figure 1). (Green and red indicate honest and adversarial blocks respectively, and the numbers on blocks indicate which block production opportunity of honest/adversary they correspond to.) Once honest nodes build a chain of length $k = 6$, the adversary releases the withheld blocks, and displaces the honest chain (cf. Figure 2). Note that the adversary can reuse its blocks $3, 4, 5, 6$. Honest nodes build a new chain on top of $2 \to 1 \to$ Genesis. Once that new chain reaches length 4, the adversary releases another displacing sub-tree (cf. Figure 3). Finally, note that the adversary can again reuse its blocks 5, 6. Honest nodes build a new chain on top of $4 \to 3 \to 2 \to 1 \to$ Genesis. Once the new chain reaches length 2, the adversary releases the last displacing sub-tree (cf. Figure 4). Honest nodes now build on $6 \to 5 \to 4 \to 3 \to 2 \to 1 \to$ Genesis. All honest blocks so far have been displaced. Overall, with this strategy, the adversary gets to displace $\Theta(k^2)$ honest blocks with k withheld adversarial blocks.

3.3 Attack Details

Selfish mining and equivocations can be used to attack PoS GHOST (using an 'avalanche of equivocating sub-trees rolling over honest chains'—hence the name of the attack). The following description is for vanilla PoS GHOST, but can be straightforwardly translated for Committee-GHOST. Variants of this attack work for Committee-GHOST with proposer boosting [1] as well.

Suppose an adversary gets k block production opportunities in a row, for modest k. This eventually happens with considerable probability. The adversary withholds these k blocks, as in *selfish mining* (cf. Figure 1 above). On average, more honest blocks are produced than

adversary blocks, so the developing honest chain eventually 'catches up' with the k withheld adversarial blocks. In that moment, the adversary releases the k withheld blocks. However, not on a competing adversarial chain (as in selfish mining for a Longest Chain protocol), but on a competing adversarial sub-tree of height 2, where all but the first withheld block are siblings, and children of the first withheld block. Due to the GHOST weight counting, this adversarial sub-tree is now of equal weight as the honest chain—so the honest chain is abandoned (cf. Figure 2 above). At the same time, ties between equal-weight sub-trees are broken such that honest nodes from now on build on what was the second withheld block. This allows the adversary to reuse in the form of *equivocations* the withheld blocks 3, 4, ..., k on top of the chain $2 \to 1 \to \mathsf{Genesis}$ formed by the first two withheld adversarial blocks, which is now the chain adopted by the honest nodes.

As an overall result of the attack so far, the adversary started with k withheld blocks, has used those to displace k honest blocks, and is now left with equivocating copies of $k-2$ adversarial withheld blocks that it can still reuse through equivocations (cf. Figure 3 above). In addition, while the k honest blocks were produced, the adversary likely had a few block production opportunities of its own, which get added to the pool of adversarial withheld blocks. Note that the attack has renewed in favor of the adversary if the adversary had two new block production opportunities, making up for the two adversarial withheld blocks lost because they cannot be reused. The process is now repeated (cf. Figure 4 above): The adversary has a bunch withheld blocks; whenever honest nodes have built a chain of weight equal to the withheld blocks, then the adversary releases a competing sub-tree of height 2; the chain made up from the first two released withheld blocks is adopted by the honest nodes, the other block production opportunities can still be reused in the future through equivocations on top of it and thus remain in the pool of withheld blocks of the adversary.

If the adversary starts out with enough withheld blocks k, and adversarial stake is not too small, then the adversary gains 2 block production opportunities during the production of the k honest blocks that will be displaced subsequently, and the process renews (or even drifts in favor of the adversary). No honest blocks enter the canonical chain permanently—a liveness failure. Also, all honest blocks are eventually removed from the canonical chain, after increasing amount of time—a safety failure for any fixed confirmation time.

3.4 Proof-of-Concept Implementation

For illustration purposes, we plot a snapshot of the block tree resulting after 100 time slots in our proof-of-concept implementation—see Figure 5 for PoS GHOST and Figure 6 for Committee-GHOST.[4] The attack is still ongoing thereafter, and as long as the attack is sustained, no honest blocks remain in the canonical chain permanently.

3.5 Applicability to PoS Ethereum

Section 2 gives a description of the Committee-GHOST protocol based on the consensus protocol of PoS Ethereum, albeit without the LMD aspect. The avalanche attack works on this protocol as well. In particular, an adversary controlling any positive fraction of validators can still replace $\Theta(k^2)$ honest blocks with k withheld blocks in this protocol.

[4]Source code: `https://github.com/tse-group/pos-ghost-attack`

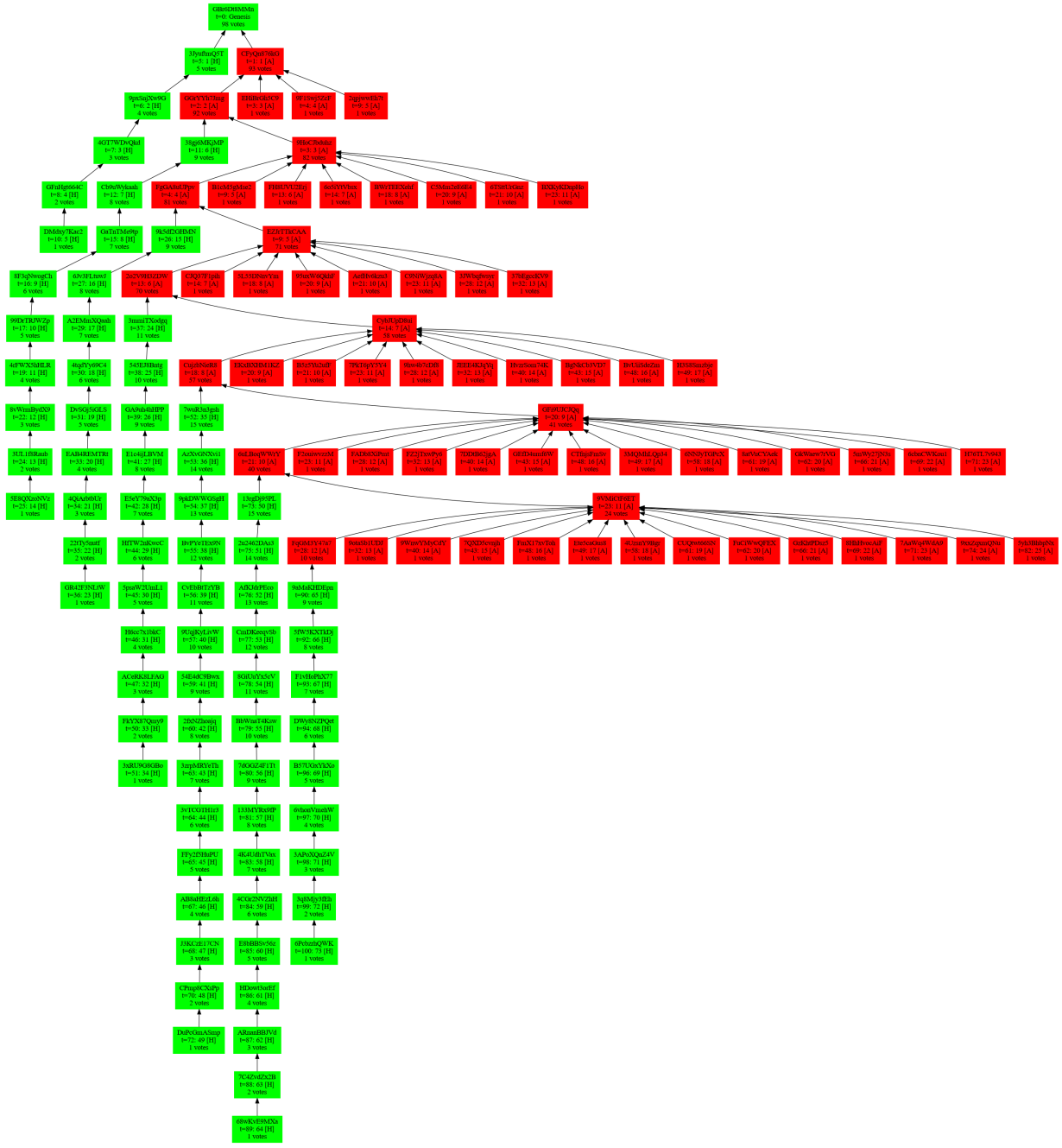

Figure 5: Snapshot of the block tree resulting after 100 time slots in our proof-of-concept implementation of the avalanche attack on PoS GHOST (adversarial blocks: red, honest blocks: green; adversarial stake: 30%, initially withheld adversarial blocks: 4)

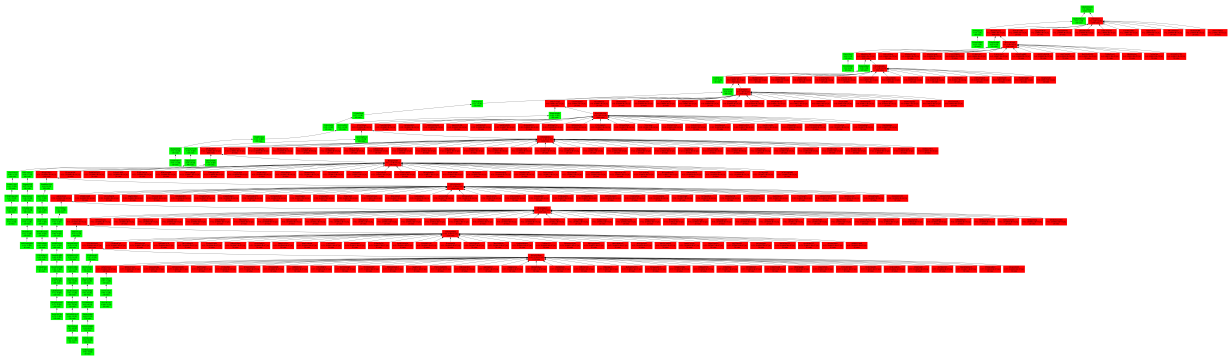

Figure 6: Snapshot of the block tree resulting after 100 time slots in our proof-of-concept implementation of the avalanche attack on Committee-GHOST (adversarial blocks: red, honest blocks: green; adversarial stake: 20%, initially withheld adversarial blocks: 12)

PoS Ethereum's LMD (Latest Message Driven) aspect interferes with this attack, but comes with its own challenges, as shown in Section 4. Nevertheless, the avalanche attack suggests a fundamental conceptual incompatibility between PoS (which enables equivocations) and GHOST (where equivocating uncle blocks disproportionately influence fork choice) and casts doubt over whether GHOST should be used with PoS at all.

4 LMD-Specific Balancing Attack on PoS Ethereum

Proposal weights (also called 'proposer boosting') were suggested [1] and implemented[5] to mitigate earlier balancing attacks [12, 13]. However, we show that the LMD aspect of PoS Ethereum's fork choice enables balancing attacks even with proposal weights. This is particularly dire because PoS GHOST without LMD is susceptible to the avalanche attack, as described in Section 3. A version of PoS Ethereum with the LMD rule is described in Section 2.

4.1 Preliminaries

Recall the following from earlier discussion of balancing-type attacks [12, 13]:

- On a high level, the balancing attack consists of two steps: First, adversarial block proposers initiate two competing chains—let us call them Left and Right. Then, a handful of adversarial votes per slot, released under particular circumstances, suffice to steer honest validators' votes so as to keep the system in a tie between the two chains and consequently stall consensus.
- It is quite feasible for an adversary to release two messages to the network in such a way that roughly half of the honest validators receive one message first and the other half of the honest validators receives the other message first. This certainly holds in the setting of adversarial network propagation delay [12] but also in the weaker setting of random network propagation delay [13].

[5] https://github.com/ethereum/consensus-specs/pull/2730

- The LMD rule deals with equivocating votes in the following way. Under LMD, every validator keeps a table of the 'latest message' (here, each message is a vote) received from each other validator, in the following manner:[6] When a valid vote from a validator is received, then the 'latest message' table entry for that validator is updated if and only if the new vote is from a time slot *strictly later than* the current 'latest message' table entry. Thus, if a validator observes two equivocating votes from the same validator for the *same* time slot, the validator considers the vote received earlier in time.

4.2 High Level Description

The LMD rule gives the adversary a remarkable power in a balancing attack: Once the adversary has set up two competing chains, it can equivocate on them. The release of these equivocating votes can be timed such that the vote for Left is received by half of honest validators first, and the vote for Right is received by the other half of honest validators first. Honest validators are split in their views concerning the 'latest messages' from adversarial validators. Even though all validators will soon have received both votes, the split view persists for a considerable time due to the LMD rule, and since the adversarial validators release no votes for later slots.

As a result, half of the honest validators will see Left as leading, and will vote for it; half will see Right as leading, and will vote for it. But since the honest validators are split roughly in half, their votes balance, and they continue to see their respective chain as leading. (The adversary might have to release a few votes every now and then to counteract any drift stemming from an imbalance on the chains different honest validators see as leading.) This effect is so stark, that it could only be overcome using proposer boosting if the proposal weight exceeds the adversary's equivocating votes (which is some fraction of the committee size) by more than a constant factor. Otherwise, if the adversary leads that constant factor number of slots, it can surpass the proposer boost again. In that case, the proposer effectively overpowers the committees by far, thus eliminating the purpose of committees.

4.3 A Simple Example

Let $W = 100$ denote the number of validators per slot. Suppose the proposal weight is $W_{\mathrm{p}} = 0.7W = 70$, and the fraction of adversarial validators is $\beta = 0.2$. Furthermore, for simplicity, assume that the attack starts when there are five consecutive slots with adversarial proposers.

During the first four slots, the adversary creates two parallel chains Left and Right of 4 blocks each, which are initially kept private from the honest validators. Each block is voted on by the 20 adversarial validators from its slot. Thus, there are equivocating votes for the conflicting blocks proposed at the same slot. For the fifth slot, the adversary includes all equivocating votes for the Left chain into a block and attaches it on the Left chain; and all the equivocating votes for the Right chain into an equivocating block and attaches it on the Right chain. With this, votes are 'batched' in the following sense. The adversary releases the two equivocating blocks from the fifth slot in such a way that roughly half of the honest validators see the Left block first (call H_{Left} that set of honest validators) and with it all the equivocating votes for the Left chain; and half of the honest validators see the Right block first (call H_{Right} that set of honest validators)

[6] https://github.com/ethereum/consensus-specs/blob/72d45971310a24f6e5ecfb149d23c9fac4c7878a/specs/phase0/fork-choice.md#update_latest_messages

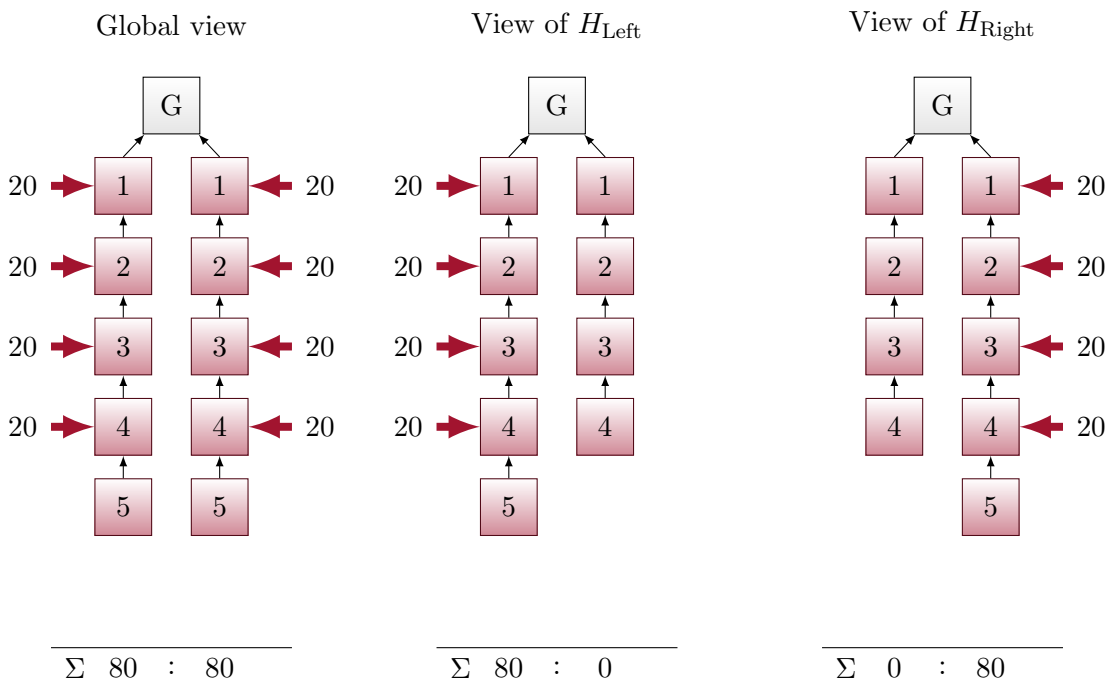

Figure 7: By the LMD rule, validators in H_{Left} and H_{Right} believe that Left and Right have 80 votes, respectively. They also believe that the respective other chain has 0 votes (as the later arriving votes are not considered due to LMD).

and with it all the equivocating votes for the Right chain. (Note that this trick is not needed in networks with adversarial delay, where the release of equivocating votes can be targeted such that each honest validator either sees all Left votes first or all Right votes first.) By the LMD rule, validators in H_{Left} and H_{Right} believe that Left and Right have 80 votes, respectively. They also believe that the respective other chain has 0 votes as the later arriving votes are not considered due to LMD (cf. Figure 7).

Now suppose the validator of slot 6 is honest and from set H_{Left}. Then, it proposes a block extending Left. Left gains a proposal boost equivalent to 70 votes (cf. Figures 8 and 9). Thus, validators in H_{Left} see Left as leading with 150 votes and vote for it. Validators in H_{Right} believe that Left has 70 votes while Right has 80 votes, so they vote for Right. As a result, their vote is tied—Left increases by roughly half of honest votes and Right increases by roughly half of honest votes (cf. Figure 10). At the end of the slot, the proposer boost disappears. In the view of each honest validator, both chains gained roughly the same amount of votes, namely half of the honest validators' votes. Assuming a perfect split of $|H_{\text{Left}}| = |H_{\text{Right}}| = 40$, Left:Right is now 120:40 in the view of H_{Left} and 40:120 in the view of H_{Right} (up from 80:0 and 0:80, respectively).

This pattern repeats in subsequent slots, with the honest validators in H_{Left} and H_{Right} solely voting for the chains Left and Right, respectively, thus maintaining a balance of weights the in global view and perpetuating the adversarially induced split view—In the LMD view of each validator, they keep voting for the chain they see leading, and 'cannot understand' why other honest validators keep voting for the other chain. (cf. Figure 11)

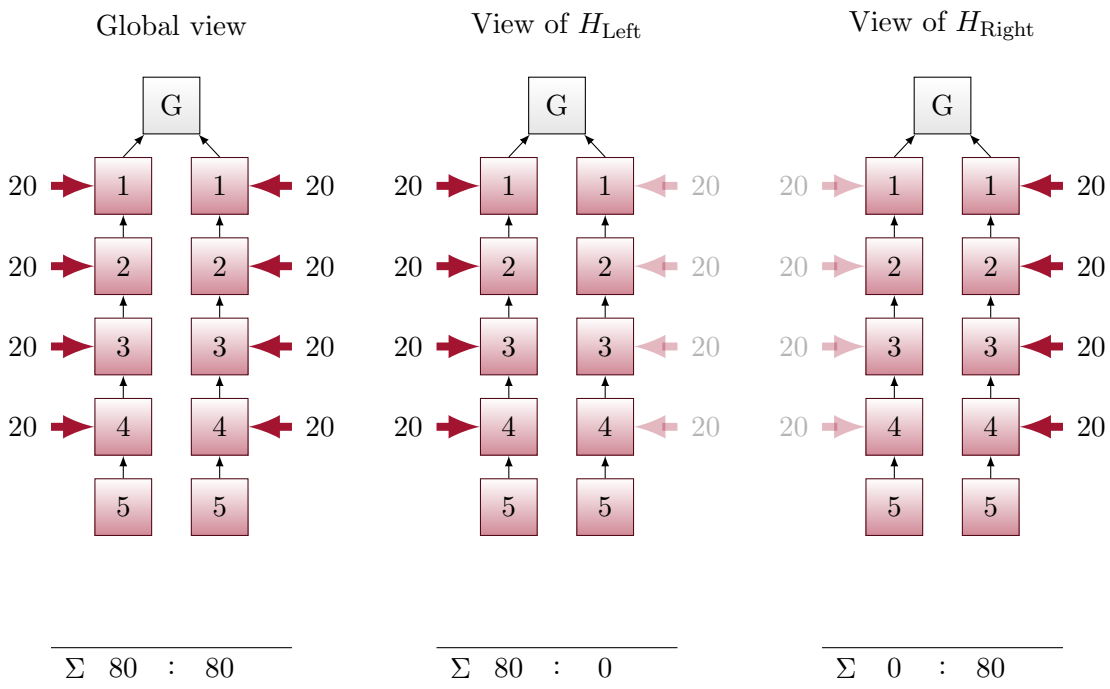

Figure 8: Suppose the validator of slot 6 is honest and from set H_{Left}. Then, it proposes a block extending Left, which gains a proposal boost equivalent to 70 votes.

Acknowledgment

We thank Danny Ryan, Aditya Asgaonkar, and Francesco D'Amato for feedback and fruitful discussions. JN, ENT and DT are supported by a gift from the Ethereum Foundation. JN is supported by the Reed-Hodgson Stanford Graduate Fellowship. ENT is supported by the Stanford Center for Blockchain Research.

References

[1] Vitalik Buterin. *Proposal for mitigation against balancing attacks to LMD GHOST*. 2020. URL: `https://notes.ethereum.org/@vbuterin/lmd_ghost_mitigation` (visited on 04/20/2021).

[2] Vitalik Buterin and Virgil Griffith. "Casper the Friendly Finality Gadget". In: *arXiv:1710.09437 [cs.CR]* (2019). URL: `https://arxiv.org/abs/1710.09437`.

[3] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. "Combining GHOST and Casper". In: *arXiv:2003.03052 [cs.CR]* (2020). URL: `https://arxiv.org/abs/2003.03052`.

[4] Vitalik Buterin and Alistair Stewart. *Beacon chain Casper mini-spec (comments #17, #19)*. Ethereum Research. Sept. 25, 2018. URL: `https://ethresear.ch/t/beacon-chain-casper-mini-spec/2760/17` (visited on 08/18/2020).

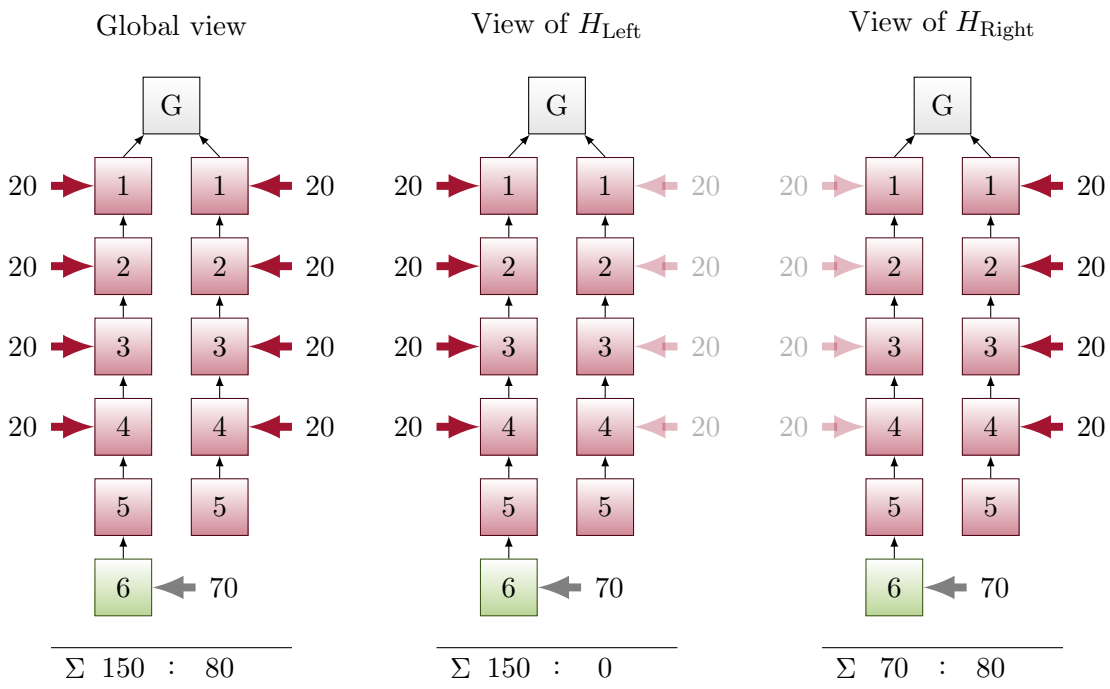

Figure 9: Due to the proposal boost, validators in H_{Left} see Left as leading with 150 votes and vote for it. Validators in H_{Right} see Left has 70 votes while Right has 80 votes, so they vote for Right.

[5] *Ethereum 2.0 Phase 0 – Beacon Chain Fork Choice*. 2020. URL: `https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/fork-choice.md` (visited on 04/22/2021).

[6] *Ethereum 2.0 Phase 0 – Honest Validator*. Apr. 5, 2021. URL: `https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/validator.md` (visited on 04/22/2021).

[7] *Ethereum 2.0 Phase 0 – The Beacon Chain*. Apr. 15, 2021. URL: `https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/beacon-chain.md` (visited on 04/22/2021).

[8] Ittay Eyal and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable". In: *Communications of the ACM* 61.7 (2018), pp. 95–102.

[9] Aggelos Kiayias and Giorgos Panagiotakos. "On Trees, Chains and Fast Transactions in the Blockchain". In: *LATINCRYPT*. Vol. 11368. Lecture Notes in Computer Science. Springer, 2017, pp. 327–351.

[10] Ryuya Nakamura. *Analysis of bouncing attack on FFG*. Ethereum Research. Sept. 8, 2019. URL: `https://ethresear.ch/t/analysis-of-bouncing-attack-on-ffg/6113` (visited on 08/18/2020).

[11] Ryuya Nakamura. *Prevention of bouncing attack on FFG*. Ethereum Research. Sept. 8, 2019. URL: `https://ethresear.ch/t/prevention-of-bouncing-attack-on-ffg/6114` (visited on 08/18/2020).

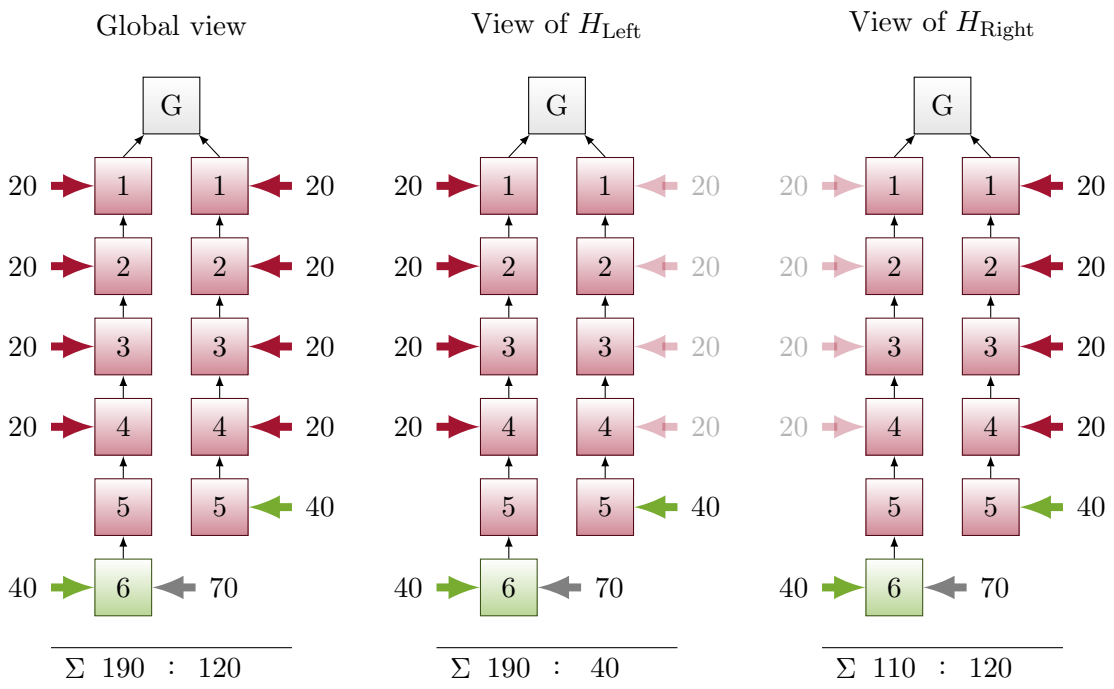

Figure 10: As a result of the split view, the vote of slot 6 is tied—Left increases by roughly half of honest votes (here 40) and Right increases by roughly half of honest votes (here 40).

[12] Joachim Neu, Ertem Nusret Tas, and David Tse. *A balancing attack on Gasper, the current candidate for Eth2's beacon chain*. Ethereum Research. Oct. 20, 2020. URL: `https://ethresear.ch/t/a-balancing-attack-on-gasper-the-current-candidate-for-eth2s-beacon-chain/8079` (visited on 04/22/2021).

[13] Joachim Neu, Ertem Nusret Tas, and David Tse. *Attacking Gasper without adversarial network delay*. Ethereum Research. July 25, 2021. URL: `https://ethresear.ch/t/attacking-gasper-without-adversarial-network-delay/10187` (visited on 09/02/2021).

[14] Joachim Neu, Ertem Nusret Tas, and David Tse. *Avalanche Attack on Proof-of-Stake GHOST*. Ethereum Research. Jan. 24, 2022. URL: `https://ethresear.ch/t/avalanche-attack-on-proof-of-stake-ghost/11854` (visited on 01/24/2022).

[15] Joachim Neu, Ertem Nusret Tas, and David Tse. *Balancing Attack: LMD Edition*. Ethereum Research. Jan. 24, 2022. URL: `https://ethresear.ch/t/balancing-attack-lmd-edition/11853` (visited on 01/24/2022).

[16] Joachim Neu, Ertem Nusret Tas, and David Tse. "Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma". In: *IEEE Symposium on Security and Privacy*. IEEE, 2021, pp. 446–465.

[17] Joachim Neu, Ertem Nusret Tas, and David Tse. "The Availability-Accountability Dilemma and its Resolution via Accountability Gadgets". In: *International Conference on Financial Cryptography and Data Security*. FC '22. Forthcoming. 2022. URL: `https://arxiv.org/abs/2105.06075`.

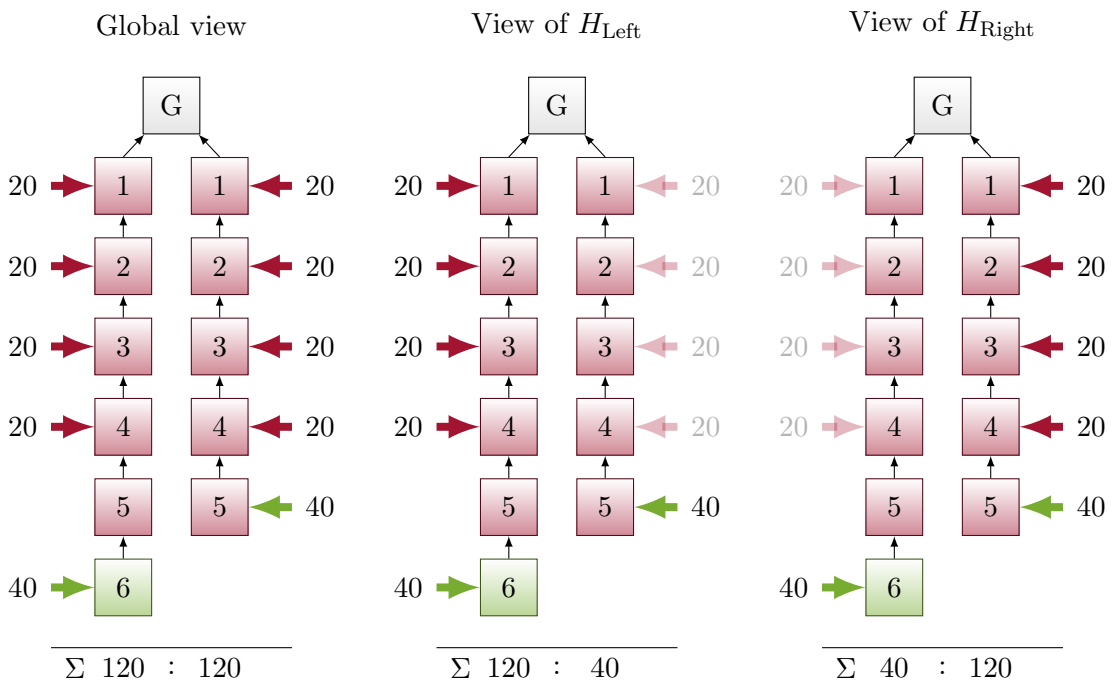

Figure 11: At the end of slot 6, the proposer boost disappears. In the view of each honest validator, both chains gained roughly the same amount of votes, namely half of the honest validators' votes. Assuming a perfect split of $|H_{\text{Left}}| = |H_{\text{Right}}| = 40$, Left:Right is now 120 : 40 in the view of H_{Left} and 40 : 120 in the view of H_{Right} (up from 80 : 0 and 0 : 80, respectively). The pattern of Figures 7–11 repeats in subsequent slots, with the honest validators in H_{Left} and H_{Right} solely voting for the chains Left and Right, respectively, thus maintaining a balance of weights (in global view–in the LMD view of each validator, they keep voting for the chain they see leading, and 'cannot understand' why other honest validators keep voting for the other chain) and perpetuating the adversarially induced split view.

[18] Michael Neuder, Daniel J. Moroz, Rithvik Rao, and David C. Parkes. "Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders". In: *Workshop for Game Theory in Blockchain (GTiB) at the 2020 Conference on Web and Internet Economics (WINE)* (2021). URL: https://arxiv.org/abs/2102.02247.

[19] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. "Three Attacks on Proof-of-Stake Ethereum". In: *International Conference on Financial Cryptography and Data Security*. FC '22. Forthcoming. 2022. URL: https://arxiv.org/abs/2110.10086.

[20] Yonatan Sompolinsky and Aviv Zohar. "Secure High-Rate Transaction Processing in Bitcoin". In: *Financial Cryptography*. Vol. 8975. Lecture Notes in Computer Science. Springer, 2015, pp. 507–527.