Timing Games: Implications and Possible Mitigations

by caspar and mike – based on extensive discussions and reviews from barnabé and francesco.

Acknowledgements

Additional thanks to Thomas, stokes, Toni, Julian, & Anders for discussions and comments!

Framing

This post aims to provide context about timing games, highlight their implications, and outline different paths forward. The goal is to initiate a constructive discussion, contributing to an informed decision by the community.

Context

Intro

Relying on honest instead of rational behavior in incentivized systems such as blockchain protocols is not sustainable. However, we are in a situation where timing games are individually rational to play and cause negative externalities for the network as a whole.

The equilibrium where everyone maximally plays timing games is not more favorable to any one validator than if everyone follows the protocol specifications honestly.

However, there is money to be made on the path to the equilibrium by block proposers playing timing games more aggressively than others; thus the honest strategy is not an equilibrium. While more research on sustainable and incentive-compatible mitigation schemes is conducted, it might be helpful to explore temporary, but less sustainable approaches to coordinate honest protocol participation.

In this post, we try to lay out the consequences of timing games and the options both in the short and long term. Ideally, this serves as a starting point for a genuine discussion regarding timing games within the broader community.

Time in Ethereum

In Ethereum, time is measured in 12-second slots. During each slot, one validator is selected as the block proposer. According to the honest validator specifications, the rules for protocol participation, a [block should be released at the beginning of the slot (0 seconds into the slot, t=0

)](https://github.com/ethereum/consensus-specs/blob/b2f2102dad0cd8b28a657244e645e0df1c0d246a/specs/phase0/validator.md#block-proposal). Furthermore, the protocol selects a committee of attesters from the validator set to vote on the canonical block in their local view. The specification dictates that the attestation is made as soon as they hear a valid block for their assigned slot, or 4 seconds into the slot (t=4

), whichever comes first. We refer to t=4

as the attestation deadline.

More on this here and here.

What are timing games?

Timing games are a strategy where block proposers intentionally delay the publication of their block for as long as possible to maximize MEV capture. Given the attestation deadline at t=4

, a rational proposer needs to ensure a sufficient share of the attesting committee, 40%, votes for their block to avoid being reorged by the subsequent block proposer.

A validator who intentionally

delays their block proposal to capture more MEV is playing timing games. We define honest behavior as:

- Using mev-boost

: Requesting a block header at t=0

.

- Not using mev-boost

: Requesting a block from the execution engine at t=0

.

Deliberate deviation from either strategy by modifying the client software is considered playing timing games under our definition. This is in contrast to “organic” latency in which a block proposal is later (e.g., from low-resource home staking).

Timing games are deeply rooted in how time in (partially) synchronous Proof-of-Stake protocols works. It’s not clear whether there exists an alternative design that prevents them entirely. Still, the status quo can be improved through short-term mitigations, as well as longer-term resolutions including protocol changes aimed at directly improving incentive compatibility. We explore both below.

More on this here, here, here, and here.

Impact of timing games

Zero-sum nature

By playing timing games, the proposer at slot N

reduces the duration of the slot N+1

auction. Any additional MEV earned by the slot N

proposer is, by definition, being taken away from the following validator. There is no free lunch; no new MEV is being created.

Consensus degradation

Delayed block proposals may lead to more missed slots, reorgs, and incorrect attestations. The protocol is designed around the network latency of propagating blocks between peers, thus any change in the timing of the initial release of the block has downstream effects on the attesting committee and subsequent proposer. Note that the degradation of the network is only rational insofar as proposers continue to benefit from playing more aggressive timing games. Any missed slots have a large negative impact on the yield of the pool, so optimizing for the inclusion rate should help with the consensus stability.

Attester timing games

In response to late proposals, rational attesters may delay their attestation deadline to vote accurately. This in turn allows block proposers to further delay their blocks. At the limit, a rational proposer knows their block needs to receive only 40% of the committee’s attestation votes (proposer boost). If they maximally delay their proposal they could target a split in the committee such that 40% of the committee hears the block before the attestation deadline (and vote for it), and 60% do not hear the block before the attestation deadline (and vote for the parent). An attester wants to get their head vote correct and so wants to make sure to be part of the accurate 40%. They can achieve this by delaying their attestation slightly (while making sure they propagate it in time for aggregation - a second round of sub-slot timing games). Attestation committees are large enough that targeting splits is feasible. Arguably the timing game would still be contained within the slot boundaries because otherwise, the subsequent block proposer could reorg the very late block.

This risks degrading into a consensus protocol that is hard for validators to reason about both theoretically and practically, potentially greatly undermining the stability and reliability of the Ethereum network overall.

Impact on blob inclusion (h/t Dankrad for mentioning this)

Blobs offered by EIP-4844 increase the size of the blocks and thus slow down their propagation to the rest of the network. If the attesting committee does not hear about a block and the accompanying blobs in time, there is a risk for the block to be reorged. Let t_1

denote the latest release time for a blobby block to reach sufficient attesters; let t_2

represent the same time but for a non-blobby block. Then t_1 < t_2

because of the increased block size of the blobby block. If the expected revenue of a block without blobs (but more MEV) released at t_2

is greater than that of a block with blobs at t_1

, a rational proposer will not include blobs. Blob creators may be required to pay higher blob priority fees to compensate for the opportunity cost faced by the rational block proposer.

The extra-protocol PBS market established by mev-boost

changes the game dynamics. Proposers do not know whether they can capture more MEV by including blobs and releasing the block earlier or excluding blobs. The underlying incentives exist for builders and proposers to overcome this information asymmetry. The most straightforward way is for them to enter a trusted relationship in the form of builder-relays (such as Flashbots, bloXroute, Eden, & Manifold). At the very least a builder-relay can delay their response to the getHeader

request up until the 950ms timeout.

Some more on this here.

Genuine latency or timing games?

It is not obvious if a validator is intentionally playing timing games or unintentionally proposing a block late.

mev-boost

detour

Note that this subsection is directly taken from “[Time is Money: Strategic Timing Games in PoS Protocols

](https://arxiv.org/pdf/2305.09032.pdf)”.

Searchers look for MEV opportunities (e.g., arbitrages), and submit bundles of transactions alongside bids to express their order preference to block builders. Block builders, in turn, specialize in packing maximally profitable blocks using searcher bundles, internally generated bundles, and other available user transactions before submitting their blocks with bids to relays. Relays act as trusted auctioneers between block proposers and block builders, validating blocks received by block builders and forwarding only valid headers to validators. This ensures validators cannot steal the content of a block builder’s block, but can still commit to proposing this block by signing the respective block header.

[

1712×1022 81.9 KB

](https://ethresear.ch/uploads/default/original/2X/4/47b1fb321cf475cbf77eee57e859476aa78c7e65.png)

When the proposer chooses to propose a block, the proposer requests getHeader

to receive the highest bidding, eligible block header from the relay. Upon receiving the header associated with the winning bid, the proposer signs it and thereby commits to proposing this block built by the respective builder in slot n. The signed block header is sent to the relay, along with a request to get the full block content from the relay (getPayload

). Finally, the relay receives the signed block header (signedAt

) and publishes the full block contents to the peer-to-peer network and proposer. As soon as peers see the new block, validators assigned to the slot can attest to it. This cycle completes one round of consensus repeating every slot.

Role of latency in mev-boost

Honest validator clients request a block from mev-boost

at t=0

. mev-boost

then pings all relays that it is connected to for their maximum bid and waits up to 950ms for the response. As a consequence, a block header might only be returned to the validator client 950ms after it was requested (note that this is not the norm). Stakers that behave honestly become indistinguishable from someone playing timing games and intentionally delaying their block.

Consider three hypothetical proposers, Alice, Bob, and Charlie. They each use mev-boost

to outsource their block production, but have slightly different setups. As a result, their p2p footprint is significantly different:

Alice:

As a solo staker in the eastern US, Alice sends her request to a US-based relay right at the beginning of her slot. With a short latency of 20ms and a quick signing on her local machine, the block is published at 60ms into the slot (three round trips are necessary). Because she made her request at t=0

, she unintentionally captured an extra 20ms worth of MEV that was generated in that interval.

Bob:

As a solo staker in Australia, Bob sends his request to the same US-based relay at the start of his slot. He has a worse internet connection and thus has a latency of 200ms to the relay. His block is published at 600ms into the slot, despite his request also being sent at t=0

. Bob benefits from being further from the relay because the 200ms it takes for his request to land on the relay are additional milliseconds of MEV captured by the winning bid.

Charlie:

Charlie is part of a staking pool also running out of the US. He intentionally waits 500ms into his slot before making the call to the relay. From his perspective, he can still get his block published well before the t=1

, so why not collect a little extra MEV for his trouble?

Tldr; honest protocol participation can be indistinguishable from rational validators playing timing games.

It requires active monitoring to understand if delays are due to bad latency or timing games and intentions are not necessarily distinguishable (bid time stamps provided by relays are fully trusted).

The path forward: To play or not to play timing games?

Time is money

The simplest path forward is to let things play out naturally and accept the reality of the protocol incentives as it is designed currently. It could be that timing games just shift the block release times back a bit and after a while, everyone adjusts to the new equilibrium, which is almost identical in payoffs compared to everyone following the (not rational) honest protocol specifications. We would likely observe more missed slots than before, but for the expected value of timing games to be positive, validators cannot be too aggressive with their block release strategy because they run the risk of their blocks being orphaned. In such a setting, one needs to consider accelerating timing games to allow stakers who cannot or do not want to actively engage in tuning latency parameters to partake in the arena of timing games. One way to enable this easily is the idea of “timing games as a service”, see further below.

The flip side to this is that this is uncharted territory with potential failure modes that are not well understood. Most importantly, why should only proposers play timing games, when it is rational to delay attestations in such an environment? It is unclear how this would play out and if the chain would be able to reliably produce blocks. Given this, such an accelerationist position toward timing games is potentially a higher risk.

Time is money but only temporarily

The honest protocol specification is not incentive-compatible because playing timing games is a more profitable strategy if a validator plays it more successfully than others. The equilibrium of timing games is identical in payoffs to everyone playing honestly (assuming relay enforcement via option 2, see below, or a relatively simple change to mev-boost

that is already discussed). Ultimately, all validators face a deadline, be it t=0

or t=4

. Facing either deadline leaves validators with equal payoffs relative to other validators. Playing honestly is merely not an equilibrium because there is money to be made on the path to the equilibrium, which is everyone playing timing games maximally. In this equilibrium, an optimized staking operator can gain a competitive advantage over another staker equivalent to the reduction in one-way latency between the validator and the relay. But this same competitive advantage already exists if everyone plays honestly.

Given the negative externalities of timing games and the short-lived room for increased profits, it’s worth exploring whether it’s possible to keep the honest strategy a Schelling point (in the colloquial meaning).

Possible short-term mitigations

Timeliness enforced by relays

Timing games present a typical prisoner’s dilemma where each validator has the option to defect by engaging in these games. This situation is inherently unstable, as any single validator’s decision to defect can disrupt the stability of everyone playing honestly. Relays can help to coordinate around not playing timing games

by enforcing some timeliness on validators, thereby reducing the scale of the coordination challenge to the more manageable count of relays.

Relays may reject builder bids that arrive after t=0

, or reject getHeader

or getPayload

requests after t=0

. In the status quo, this removes the ability for proposers to take advantage of timing games.

Assumptions

- No relay defects from enforcing these rules. Relays are already trusted entities so extending that trust further is not ideal, but not unrealistic.
- Validators do not enter trusted relationships with builders directly, circumventing the necessity for relays (e.g. builder-relays). Validators need a reputation to be trusted by builders, as they could steal the MEV. Hence, it is largely about trusting that large staking pools and whales do not defect from using relays.

Benefits

The coordination problem is reduced by orders of magnitude. It is still a prisoner's dilemma, but socially it is much easier to coordinate and sustain.

- Defection is more easily detectable, see the section on monitoring. Importantly, only a handful of relays need to be monitored to not release blocks late (further assuming no trusted validator-builder relationships).
- Relays are already trusted not to steal MEV, so adding a new trust assumption is more feasible than trusting the validators themselves.
- Relays already have some cutoff times implemented, which could easily be changed.

Drawbacks

- It is still rational for validators to defect if they find a way to do it.
- It only takes a single relay to "defect alongside" with a validator for defection to be possible.
- A validator and builder entering a trusted relationship can defect together. In other words, it incentivizes vertical integration.
- Social coordination is unsustainable and messy.

Monitoring

- Block release times. If consistently late, likely due to timing games.
- Cross-referencing bids on different relays to check when they were received. This can help give an idea of the winning bid likely arrived after the start of the slot.
- Transactions included in blocks that were not in mempool before t=0

indicate timing games are being played. There is still the possibility that the prior point is due to private orderflow, rather than the inclusion of late transactions. But together with late block releasing, this becomes a strong indicator for timing games.

- Correlating prices of on-chain trading venues with off-chain prices (potentially super noisy).

Overall, relay enforcement is a tool to make social coordination on following the honest protocol specifications easier. While it is not sustainable, it could help in the short term. The figure below shows the three calls that define the block production flow in mev-boost

. Each dotted line represents a point at which a relay enforcement could take place. Options 1-3 are discussed in detail below.

[

photo_2023-12-05 15.00.55

1139×616 50.3 KB

](https://ethresear.ch/uploads/default/original/2X/a/ae9f63a3af0c8394fabf71a04d390e91228e6116.jpeg)

mev-boost

calls

The [mev-boost

](https://github.com/flashbots/mev-boost/#api) flow contains three critical events, each of which could be enforced at the slot boundary of t=0

: (i) submit bid

, (ii) getHeader

, (iii) getPayload

. Right now, there are alreadytwo timestamps enforced by the relay:

getHeaderRequestCutoffMs = cli.GetEnvInt("GETHEADER_REQUEST_CUTOFF_MS", 3000) getPayloadRequestCutoffMs = cli.GetEnvInt("GETPAYLOAD_REQUEST_CUTOFF_MS", 4000)

In other words, getHeader

must be called by t=3

, and getPayload

must be called by t=4

. The following options explore changing these bounds to be more strict.

Option 1: Relay rejecting new bids after t=0

Description: The relay can reject any bids submitted by builders once the slot has begun (at t=0

). This removes any incentive for the proposer to delay their call to getHeader

because the value of the bid will no longer increase.

Option 2: Relay rejecting getHeader

requests after t=0

Description: Alternatively, a relay could reject getHeader

requests once the slot has begun (at t=0

). This has the effect of ending the auction since no new bids are served to the proposer and also enforces that the proposer at least receives their bid by the beginning of the slot.

Option 3: Relay rejecting returning getPayload

requests after t=0

Description: A relay could reject any calls to getPayload

after the slot has begun (at t=0

). This is an even stronger requirement because it enforces that the proposer completes the signing process by the beginning of their slot.

Timing Games as a Service by relays (h/t Justin)

Description: Another idea is for relays to offer “Timing Games as a Service” (TGaaS). This approach aims to democratize access to timing game rewards rather than eliminate them – the same design principle that inspired mev-boost

in the first place. TGaaS builds on the following facts:

- relays are well-peered in the p2p layer, and
- relays know the ping latency to the proposer.

With this, the relay can take on an additional role of advising the proposer on when they should sign a builder’s bid. This is easiest understood with a toy example; assume the relay has calculated that

1. the ping latency of the slot N

proposer is 100ms,

1. the proposer expects to take 100ms to sign a header (this could be a new field in the validator registration),
2. the block takes 400ms to propagate given the relay peering.

Then the relay can push a header to the proposer at t=3.3

, expecting the proposer to receive it at t=3.4

, complete the signing at t=3.5

, return the signed header to the relay at t=3.6

, leaving 400ms for the relay to circulate the block to the attesters. In reality, it would be up to the relay to tune these parameters to minimize the number of missed slots they cause. Most likely, a few hundred additional milliseconds would be added as a safety buffer to account for the tail latencies of networking.

In theory, a proposer could tune the getHeader

request time such that they receive the block header by the relay at the same time as the relay offering TGaaS would push the header to the proposer. Practically however, it is likely that outsourcing this complexity allows e.g. home stakers to play timing games more aggressively.

However, it is unclear how to not introduce complexity for a proposer if they sign up for multiple TGaaS relays. Say there are two relays A and B, with A fast, and at the “final time” for B (the time at which the proposer has to either sign B’s header or give up on B) the bid from B is higher. Now the proposer has to determine whether the extra time bought by A’s latency advantage is worth gambling on.

Pros:

- It democratizes access to timing games, reducing the benefit of sophistication.
- Relays would have extensive access to data and could tune their parameters to minimize the liveness impact of TGaaS.

Cons:

- It increases the probability of attester timing games being played.
- It is unclear how a proposer avoids complex latency parameter tuning without relying on a single point of failure.
- Proposers lose agency over block release time to relays.
- Relays might not maximize over the same objective function. The opportunity cost of a relay missing a slot is significantly less, e.g. they do not stand to lose CL rewards.
- This proposal furthers the dependency on relays and their performance directly corresponds to the Ethereum network health. Relays absorb more responsibility and risk, while presently they serve more as a simple pipe connecting proposers to builders.

Possible long-term mitigations

This section highlights some of the design space (by no means complete) that exists to alter incentives to improve protocol resilience to timing games. Each idea will only be briefly discussed as the details are beyond the scope of this post.

Retroactive proposer rewards

The goal here is to explicitly incentivize timeliness. The trouble is to find a good on-chain heuristic for what “timely” means. One heuristic could be to use the share of “same-slot attestations” that are included in the subsequent block. The intuition is that, if almost all attesters (from the same slot) saw the block in time to vote for it, then the block must have been somewhat timely.

Pros:

- Explicitly incentivizes timeliness.
- Arguably rewards intended validator behavior regardless of timing games.

Cons:

- MEV rewards can dominate timeliness rewards.
- Hard to discriminate between different degrees of timeliness, as within a few milliseconds the share with which the block is seen by most of the attesting committee jumps from a very low to high value.

Here is an old high-level write-up of this idea.

Missed slot penalties

Currently, there is no penalty for a proposer missing a block. The “only” cost is the opportunity cost of not receiving any consensus and execution-related rewards. Introducing a penalty for missing a slot would decrease the expected value of playing timing games because the increased probability of a missed slot (by playing timing games) now carries negative weight.

Pros:

- Simple to implement and reason about.
- Reduces the expected value of playing timing games.

Cons:

- Penalizes honest proposers who happen to organically miss a slot (solo stakers likely miss more slots relative to professional staking operators).
- Big node operators can better manage associated penalty risks for they propose many blocks a day.
- Requires selection of a “magic number” for how large the penalty is (all consensus rewards are already arbitrary, but due to the high variance of MEV, pricing penalties in that domain is particularly tricky).

Earlier attestation deadline with some backoff scheme

The attestation deadline being at t=4

makes the timing games highly profitable. If instead, the deadline was earlier, there would be less value to capture relative to everyone participating honestly. In other words, it reduces the difference in the value of playing honestly and playing timing games. This comes at the cost of worse liveness properties of the protocol because of the higher probability of organically missed slots due to network latency. Thus a backoff mechanism that dynamically adjusts the attestation deadline based on chain performance is necessary.

Pros:

- Reduces the absolute value capturable by timing games: lower opportunitiy cost to honest protocol participation.
- A backoff scheme is also a prerequisite for (block, slot)-voting, which in turn is useful for several things: a cleaner implementation of honest reorgs of late blocks and a prerequisite for ePBS designs.

Cons:

- Increases protocol complexity.
- Unlikely that the attestation deadline would be shifted significantly. Blobs will naturally shorten the possible timing game window. Even rebalancing the attestation deadline to something later is discussed.
- Only effective in reducing the opportunity cost to honest protocol participation.
- It does not perevent attester timing games, but could accelerate them.

Probabilistic sub-slot proposing (h/t Francesco & Vitalik)

Say a slot is 10s long and split up further into 100ms sub-slots. Each sub-slot has a 1% probability of having a proposer such that in expectation you have one proposer every 10s.

Then a proposer is only guaranteed their monopoly for the duration of their sub-slot. It might be that a proposer is elected for the subsequent sub-slot already. As a result, timing games are practically only possible in expectation.

Pros:

- Fundamentally changes the game dynamics and challenges timing games at its ‘core’: Delaying block proposals immediately risks being reorged.
- Prevents attester timing games.

Cons:

- Timing games are playable in expectation, but importantly it avoids attester games.
- Deep protocol change.
- PoW-style block times as opposed to a block proposer every 12s.

This is an exciting direction to explore further as it tackles the problem closer to the root. But it is also an involved protocol update with lots of elements yet to be understood. Currently, this is only a promising idea.

Conclusion

In the end, the path to choose largely depends on how dangerous attester games are considered to be. This post aims to provide context about timing games, highlight their implications, and outline different paths forward. The authors do not favor any mitigation specifically. Instead, the goal is to initiate a constructive discussion, contributing to an informed decision by the community.