# Latency is Money: Timing Games /acc

Data Always - February 23, 2024

**tl;dr**

> In an effort to allow decentralized consensus participation, the Ethereum network inadvertently rewards latency to the benefit of less efficient block proposers. High-quality validators are intrinsically penalized by the system and struggle to compete on yield unless they simulate latency by playing timing games.
>
> Although these games are frequently vilified for degrading the health of the network, proposer sets (ex-CoinSpot) that intentionally delay their `getHeader()` requests empirically see few of their blocks reorged. By contrast, large entities like Coinbase often have extremely high-variance setups and see their blocks frequently reorged despite not playing timing games.

**Assumed Audience:** Ethereum researchers and stakers. You understand the MEV supply chain, have a basic understanding of the phases of a slot, and are comfortable with the roles played by builders, relays and proposers. You know what timing games are and are versed in the negative externalities that they cause for the network.

**Data Sources:**

- Bid time data from dataalways/mevboost-data on GitHub
- Top-level entities from Hildobby's proposer pubkey set on Dune
- Lido node operator pubkeys from the pubkey_mapping dataset on mevboost.pics

**Caveat**: This analysis does not take into account potential shifts in timing game dynamics that may occur once EIP-4844 is live on the network.

**Conflicts of Interest**: This research was entirely unfunded. The authors have exposure to spot ether and Coinbase staking products but declare no other meaningful conflicts of interest.

---

Timing games have always been inevitable. Now that multiple proposer sets have established that they can generate excess yield by delaying their block proposals, the initial unstable economic equilibrium has been shattered and the long-term expectation is that all proposers will shift to playing timing games. Unfortunately, smaller proposer sets and solo stakers do not have the resources to run extensive pilot projects to optimize their yield profiles.

We hope to create a foundation on which proposer sets can build to better understand whether or not, and to what level, they should begin delaying their block proposals. We begin by presenting network-wide data and then examine subsets of proposers with similar patterns of behaviour. We end by discussing advanced strategies that proposers may use to either optimize or obscure the timing games they play.


## Yield Optimization

Optimizing the timing of a block proposal is a balance between expanding the size of the transaction set in the mempool against the increasing probability of the block being insufficiently propagated through the network to avoid being forked (i.e., reorged). To calculate the expected yield of a block we take the product of the value of the block and the likelihood of the block being sufficiently propagated:

$$\mathbb{E}(k, t) = V(t) \cdot (1 - P(k,t)), $$

where $k$ is a subset of similar proposers, $t$ is the time in a slot, $\mathbb{E}(k,t)$ is the expectation value of a block proposal $V(t)$ is the value of the block, and $P(k,t)$ is the probability of a block being forked. The block value is assumed to be only depend on slot time (i.e., we don't factor in market or network conditions), whereas the probability of a block being forked depends both on the time in the slot and the

efficiency of the proposer set. We do not formalize efficiency, but we use the *k* parameter to distinguish between subsets.

**Block Value**

The value of time in a block is a foundational economic topic in the Ethereum research community, yet at its core is still not well understood. Visible in Figure 1, it is clear that bid values trend up in time, but the constant evolution of the ecosystem and the maturation of time-sensitive cross-domain MEV strategies (i.e., CeFi--DeFi arbitrage) threatens to warp the time dependence of block values.

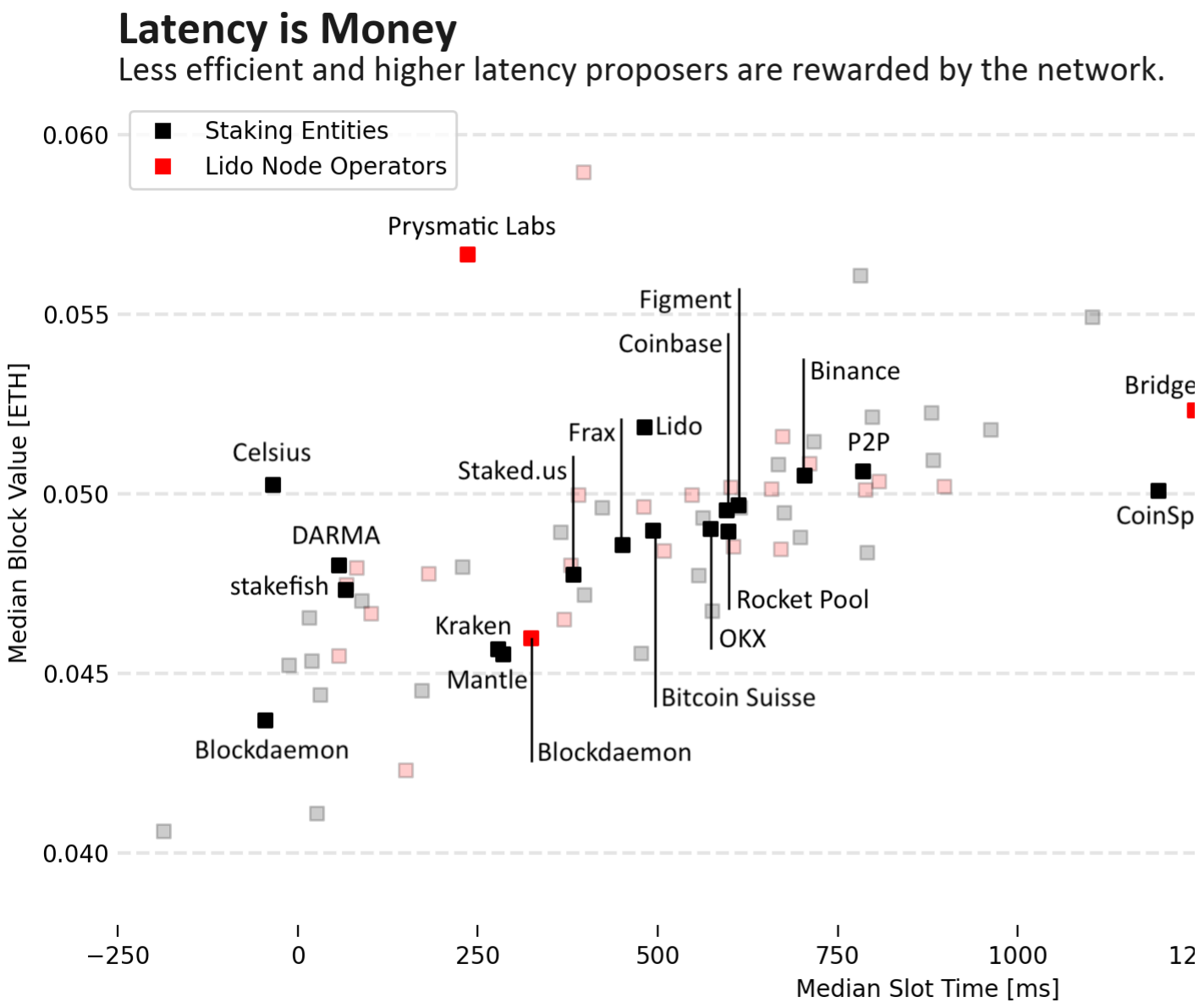
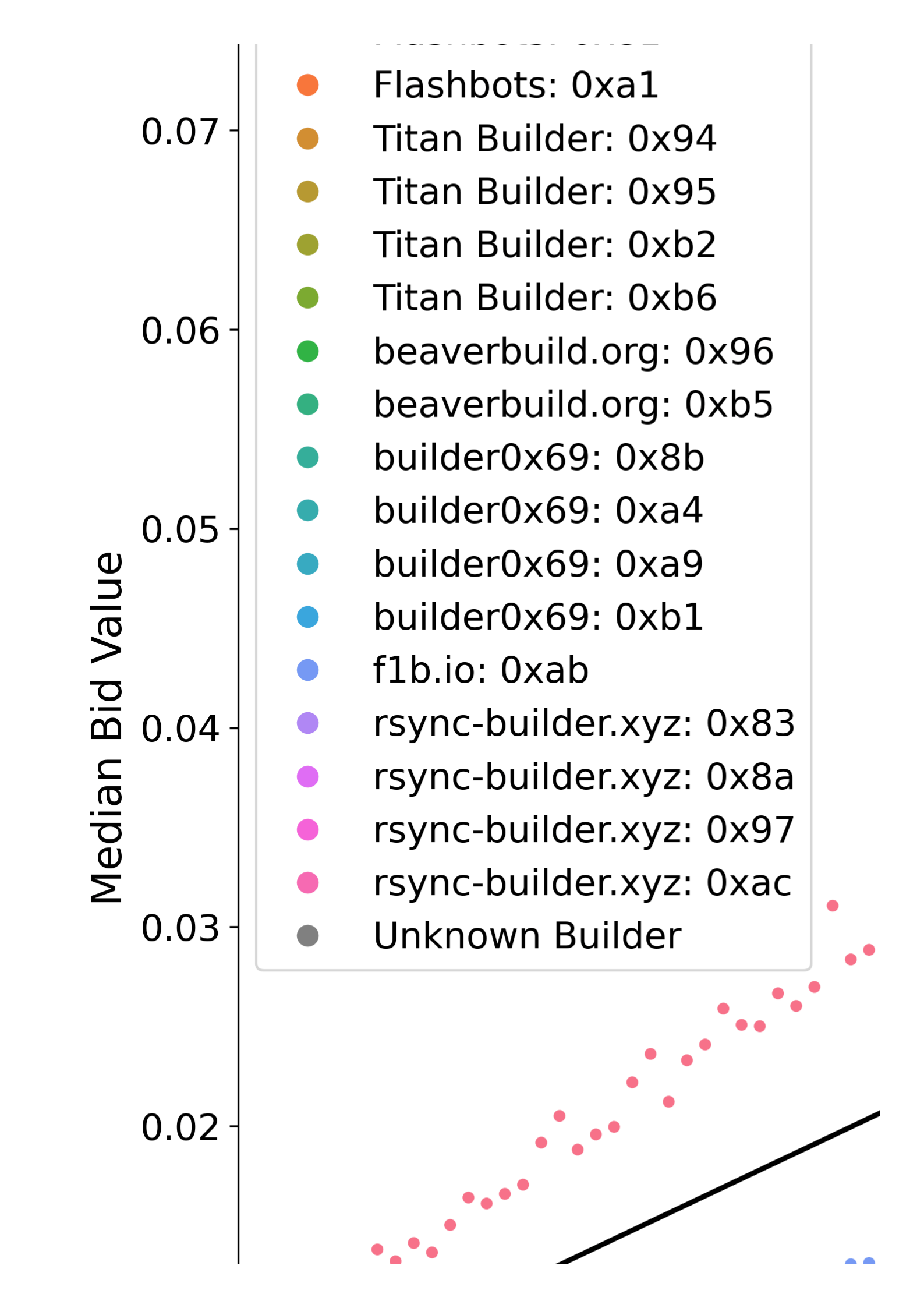



Figure 1: Proposers with inefficient or high-latency setups, as well as proposers who simulate these conditions, tend to earn higher execution layer rewards than honest high-quality proposers.

Despite our reservations, this analysis adopts the work of Thomas Thiery, seen in Figure 2, which showed that the median shape of *V(t)* in June 2023 was close to linear. Stemming from linearity, Thiery found intrablock transaction velocity to be constant, i.e., the value of every equally sized slice of time was the same. For the layman, this indicates that the median block did not contain measurable non-atomic MEV. As integrated builders continue to mature, we'll encounter more edge cases, but for now, we embrace the linear model as a simplistic yet mostly accurate approximation.


Flashbots: 0xa1
Titan Builder: 0x94
Titan Builder: 0x95
Titan Builder: 0xb2
Titan Builder: 0xb6
beaverbuild.org: 0x96
beaverbuild.org: 0xb5
builder0x69: 0x8b
builder0x69: 0xa4
builder0x69: 0xa9
builder0x69: 0xb1
f1b.io: 0xab
rsync-builder.xyz: 0x83
rsync-builder.xyz: 0x8a
rsync-builder.xyz: 0x97
rsync-builder.xyz: 0xac
Unknown Builder
Median Bid Value
0.07
0.06
0.05
0.04
0.03
0.02

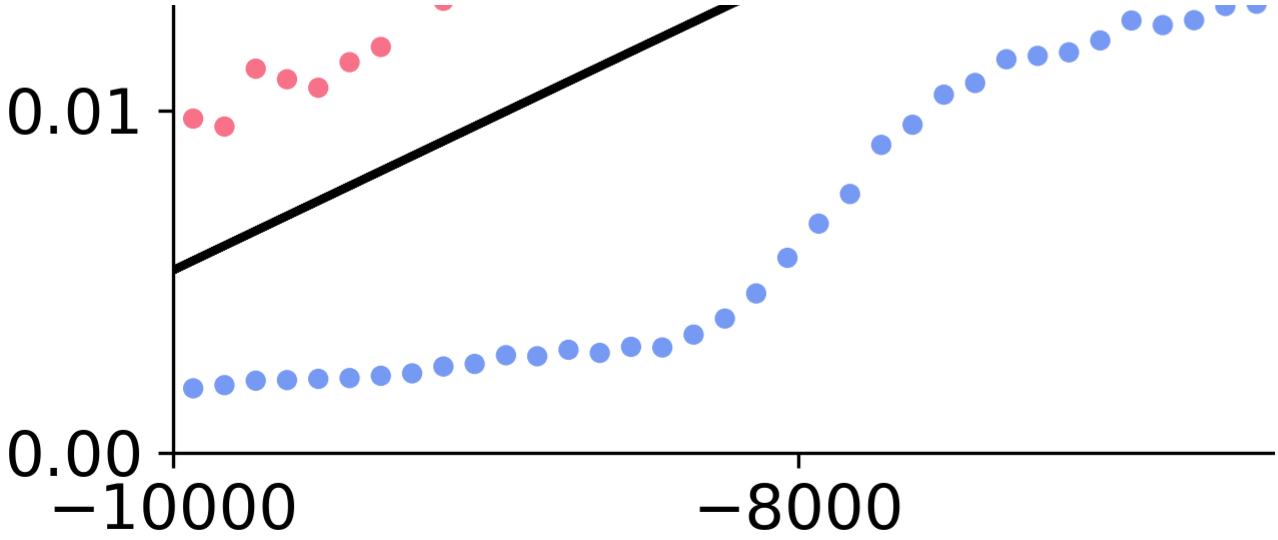



Figure 2: Analysis by Thomas Thiery demonstrated that the value of the median MEV bid profile received by the ultrasound relay in June 2023 trended up linearly in time.

Source: Empirical analysis of Builders' Behavioral Profiles

We chose not to adopt the terminal shape of the curve because we believe that the maximum is an artifact resulting from sparse data. The maximum is far outside the IQR of the bid submission data (Figure 2B), the analysis does treat blocks where these bids are noise versus meaningful independently, and builder behaviour has likely changed over the past nine months. One of the fundamental assumptions underlying timing games is that auction values trend up in time, so the idea that having more transactions to choose from could make your block less valuable is incompatible with theory.

### Forked Block Rates

There are no comprehensive studies of forked block rates and causes for us to rely upon. As a result, we approximate the temporal relationship empirically by calculating the comparative fraction of forked-to-successful blocks for every winning bid detection time by a relay. This methodology introduces error, but because there is no public data on `getHeader()` call times by proposers or reception times by relays, it is the best we can do.

The only characterization of this measurement error comes from Kiln's empirical analysis of the impact of block delays. Figure 3, a reproduction of Kiln's analysis, demonstrates that for high-quality proposers there is a strong correlation between intentional delays and the resulting winning bid timestamps, but that it clearly introduces noise. We note that Kiln has one of the best optimized block proposal setups, so although the correlation is strong in this dataset, it is likely weaker and temporally shifted for other proposers.

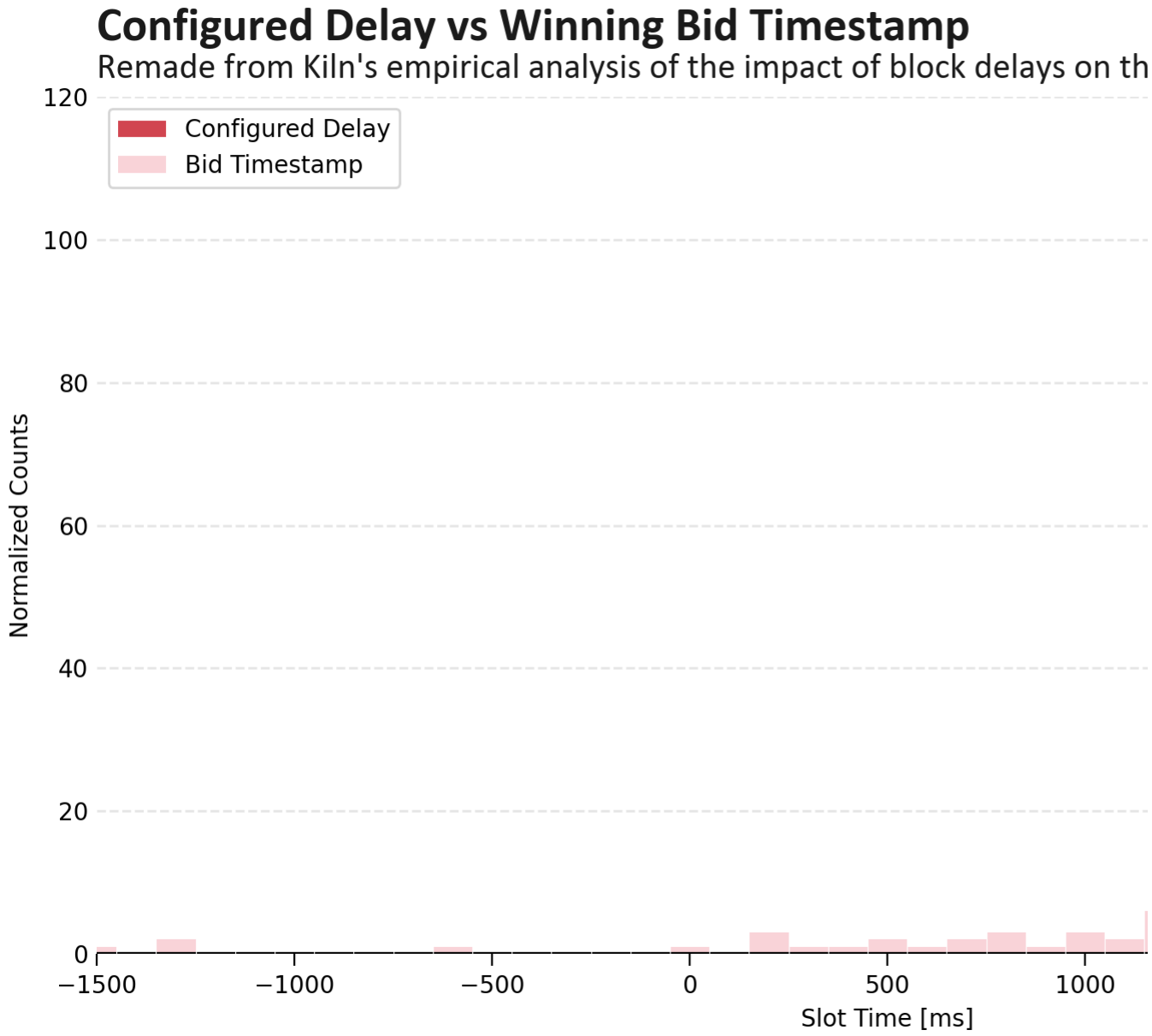



Figure 3: Recreation of Kiln's comparative analysis of the impact of configured delays and winning bid timestamps. Kiln found strong correlation but not perfect overlap between the two profiles. After discussing with the author, we chose to normalize the counts of the configured delay to adjust for missing bid data from the relay APIs.

Source: Empirical analysis of the impact of block delays on the consensus layer

In Figure 4, we compare the density profile of forked blocks against successful blocks. From the diverging shapes of the two profiles we see that the peak of the forked blocks profile is much later. This skew suggests that late arriving bids are less likely to be built on-top of.

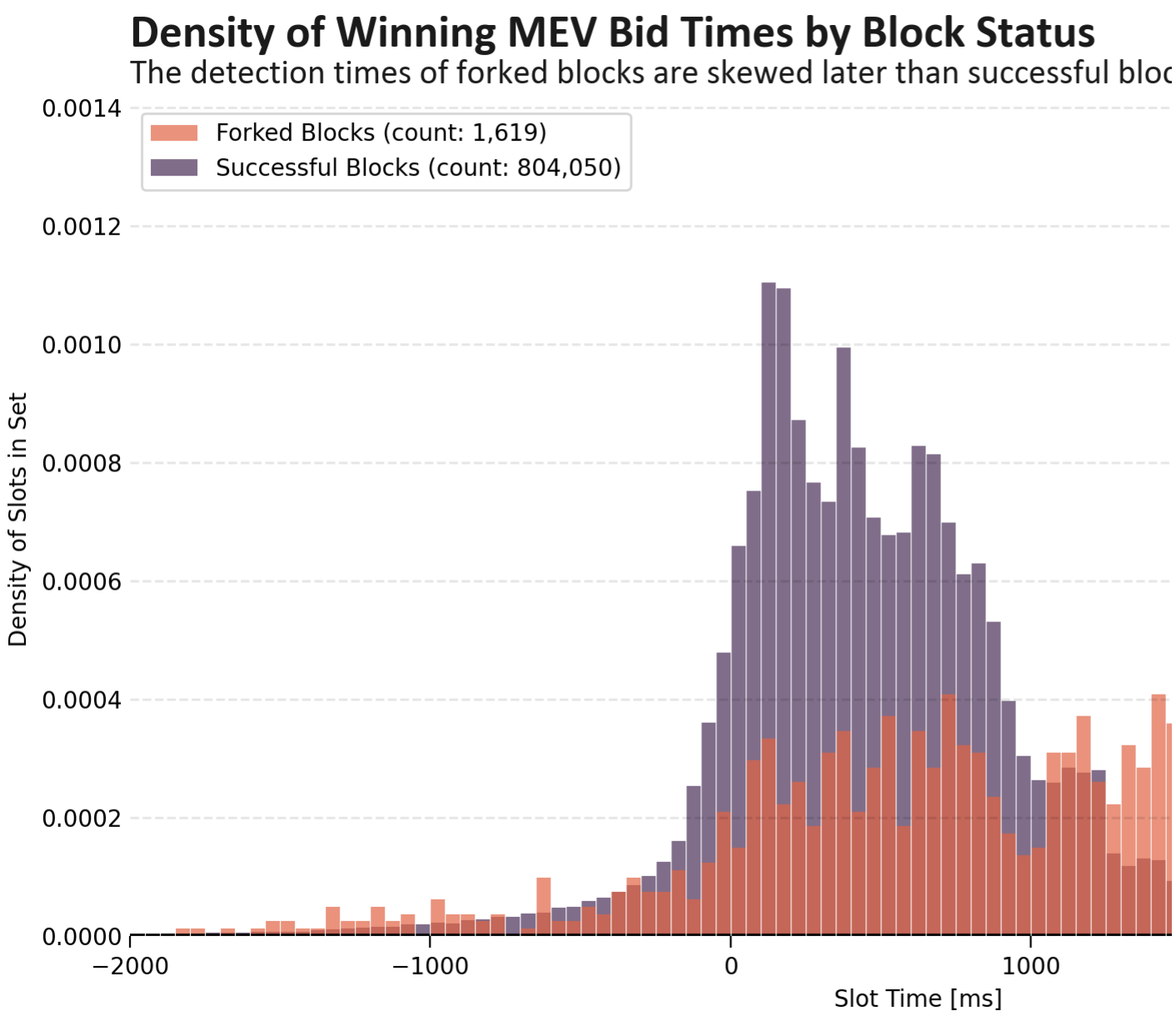



Figure 4: The distribution of winning bid times for forked blocks is skewed later than for successful blocks.

Combining the two datasets, in Figure 5, we generate an empirical relation between slot time and the odds of a block being forked. We find that winning bids that arrive before 1500 ms are almost never reorged, but as proposers push the envelope further and further, winning bids arriving after 2500 ms are often insufficiently propagated resulting in no rewards for the proposer.

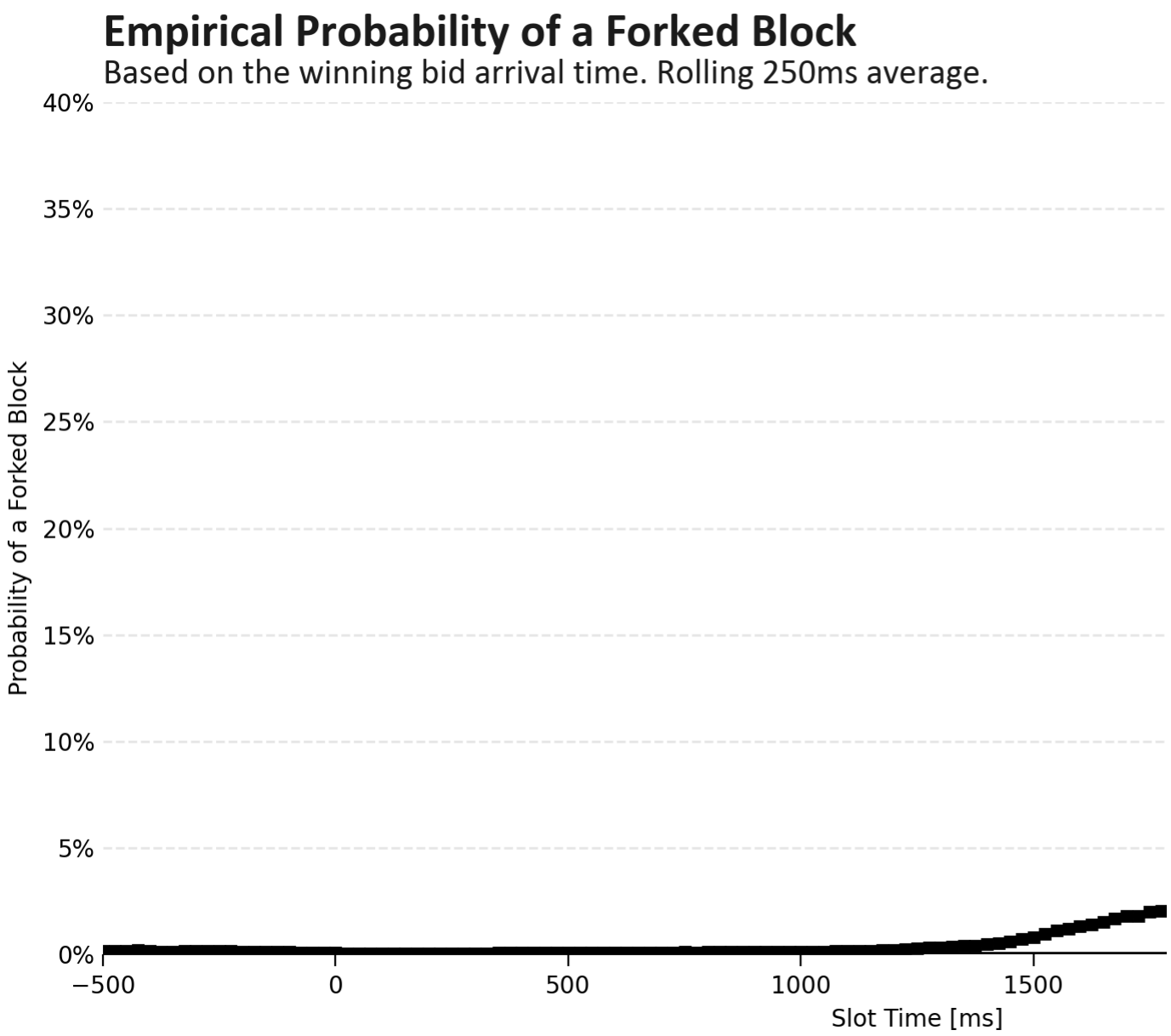



Figure 5: The probability of a forked block is heavily time-dependent once bids exceed the 1500 ms range. The network-wide profile has a minimum around 2200 ms stemming from sophisticated node operators playing timing games and targeting blocks in that time region.

**Expected Value of a Block Proposal**

Plotting the time dependence of our expected value equation for the superset of all $k$, we find that the profile peaks at a winning bid slot time of 2250 ms. After this slot time, the rate of forked blocks accelerates quickly and proposers see diminishing returns from further delaying proposals.

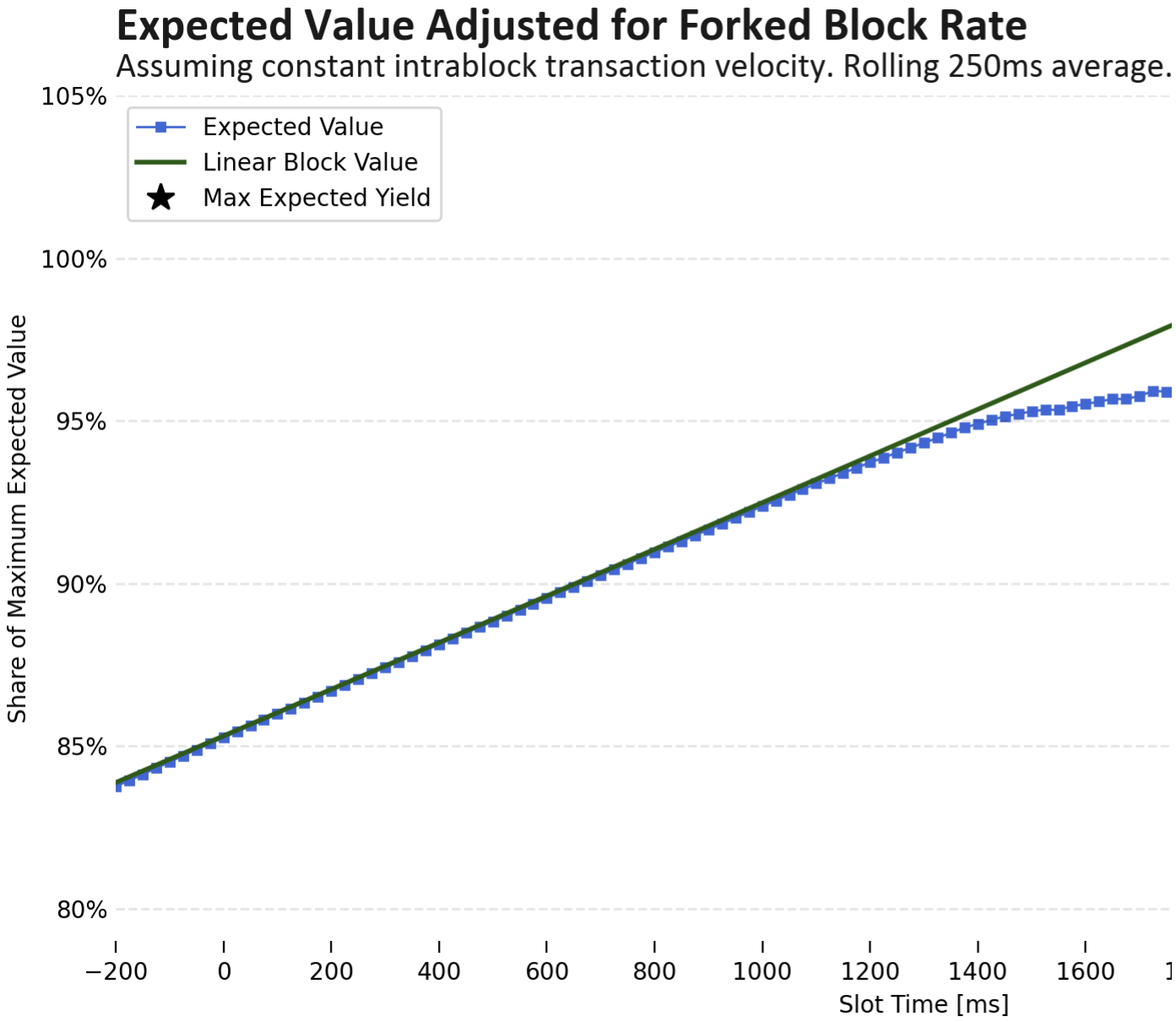



Figure 6: The expected value of a block begins to deviate from the block value once the forked block rate becomes nonzero, attaining a maximum at 2250 ms. The terminal behaviour of this curve resembles Thiery's block value fit that we rejected, but we note that Thiery's fit represents the block value not the expected value.

The median winning bid arrival time is approximately 350 ms, representing 88% of the maximum capturable value generated when the proposer controls which transactions are written to the blockchain. The theoretical median proposer maximizes their yield by delaying their `getHeader()` requests by 1900 ms.

However, winning bid arrival profiles are not delta functions—they are, in fact, very wide. Delaying all requests by 1900 ms would transform the entire long tail into forked blocks and drive missed slot rates to extreme levels. To properly optimize for expected value, we must shift every slot in the profile, calculate the new yield and adjust for the increased chance of a fork. The result is an optimal shift of 1400 ms and an increase in yield of 8.5%.

### Post MEV-Burn

We warned in September 2023 that MEV-Burn could push more proposers to play timing games. Our modelling suggests that vanilla MEV-Burn, which burns the first 10 s of value in a block, would shift the peak of the expected value profile from 2250 ms to 2475 ms (Figure 7). The decreased risk of missing a block will encourage proposers to delay as long as possible and increase the rate of forked blocks significantly. Modifications to vanilla MEV-Burn are currently being studied; we believe that adding aggressive penalties for missed slots is the cleanest way to modify the risk--reward of these games.

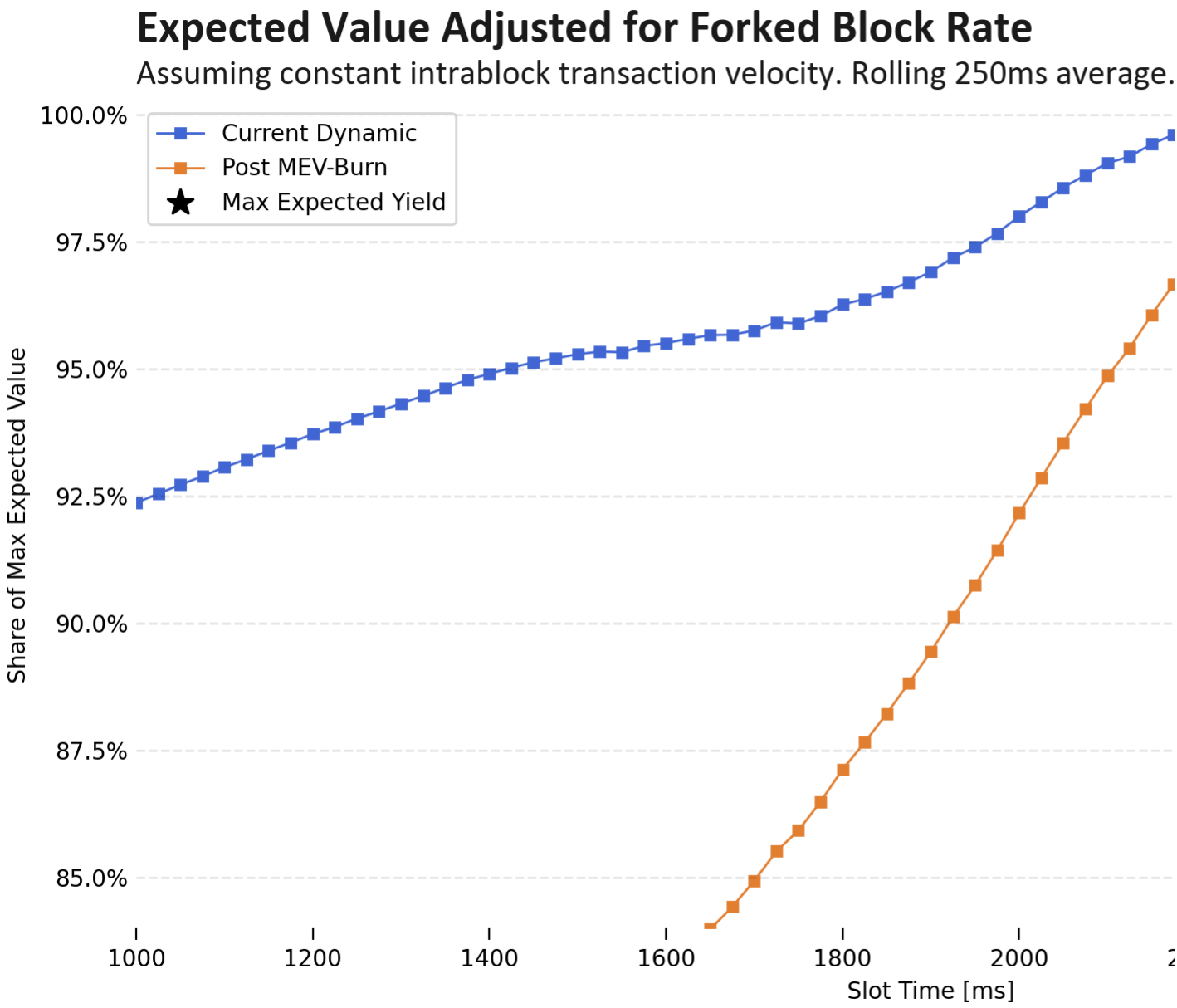



Figure 7: Vanilla MEV-Burn changes the shape of the expected block value risk--reward profile and shifts the maximum later into the block. Timing games will become worse under MEV-Burn unless the selected proposal is crafted carefully.

### Timing Games in the Wild

The inherent latency of peer-to-peer networks makes it challenging to determine which proposer sets are playing timing games. Rather than plainly enumerating which proposers submit late blocks, better approaches should account for the variance of slot times to separate between poor optimization and intentional delays. This analysis attempts to capture the long-tail behaviour of proposers by defining a timing spread as the difference between the 95th percentile of slot times versus the median for the proposer set. We plot this dynamic in Figure 8 to show which proposers submit late but consistent blocks, compared to which proposers are simply inconsistent.

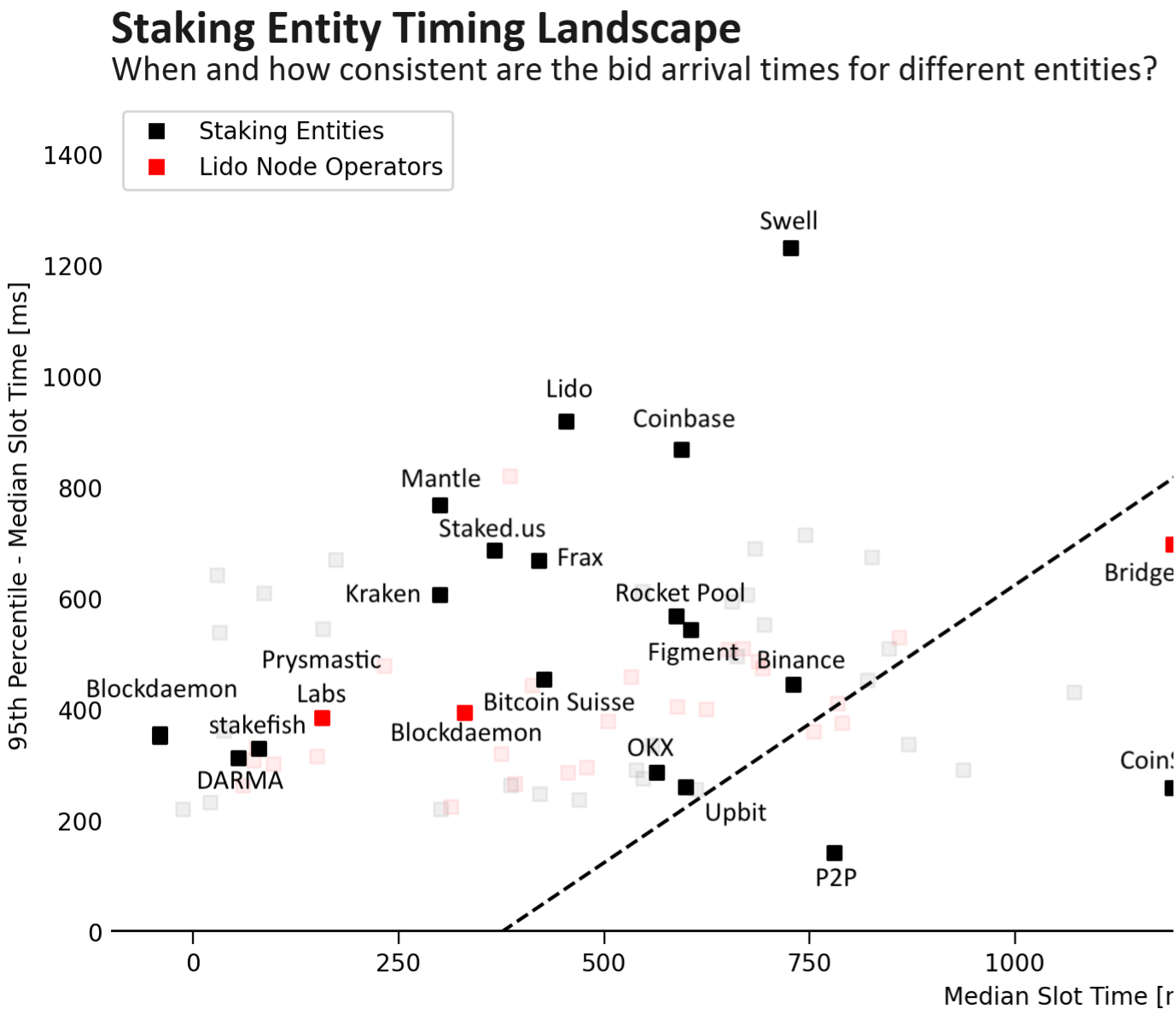



Figure 8: We can naïvely detect which proposers are playing timing games by measuring the difference between the long tail of their block proposals compared to their median block times. A larger long tail suggests inefficiencies in the setup of a proposer set, whereas a sharp drop-off suggests that entities have optimized setups and are intentional in the timing of their blocks.

Empirically proposer sets with low spread but high median slot times, are the most likely to be playing timing games; this group includes Kiln, Attestant, CoinSpot, and P2P. By contrast, the proposers who we expect to benefit the most from starting to play timing games are Blockdaemon, DARMA Capital, and stakefish. We note that although P2P has announced that they are intentionally delaying their blocks, they are far from the most aggressive actor in the space, and that proposer sets that are not playing timing games but have high variance frequently select later bids than even the most delayed P2P blocks.

**Relays**

Analyzing winning bid times can only provide insight into one part of the picture; the public does not have access to the timestamps of when relays receive `getHeader()` requests or when proposers send the requests. Without this metadata, we implicitly assume that the behaviour of relays is static when, in practice, this is evidently not the case. Timing games are also being played at the relay level.

The definitive examples of timing games at the relay level are the recent infrastructure upgrades made by bloXroute and the launch of its validator gateway in December 2023. The bloXroute validator gateway is a geographically distributed mesh of servers to which the latest bid values are constantly pushed. This network allows for validators to connect to a virtual relay closer to home at significantly reduced latency. bloXroute is then able to intentionally delay their `getHeader()` responses and deliver the data just-in-time (JIT) resulting in later and more valuable blocks for proposers.

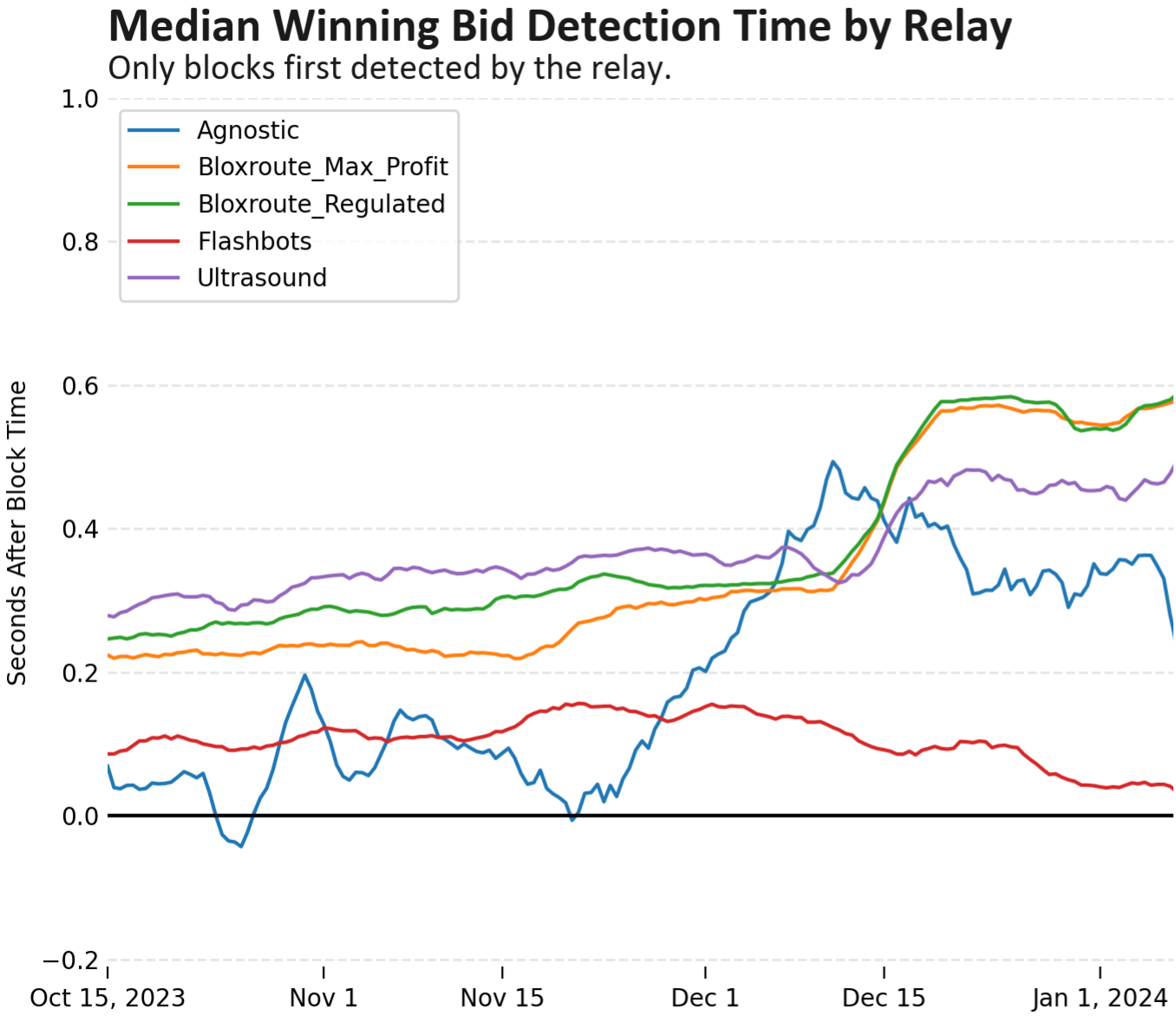



Figure 9: Most timing game analysis only considers the raw winning bid times and ignores changes at the relay level. With bloXroute recently moving to a JIT delivery model we can no longer rely on only proposer set bid timing to identify shifts in proposer behaviour.

If we fail to account for the evolving behaviour of relays, then we are left with inaccurate results that imply that all proposers started playing timing games in December. The drastic change in bloXroute's profile is apparent in Figure 10.

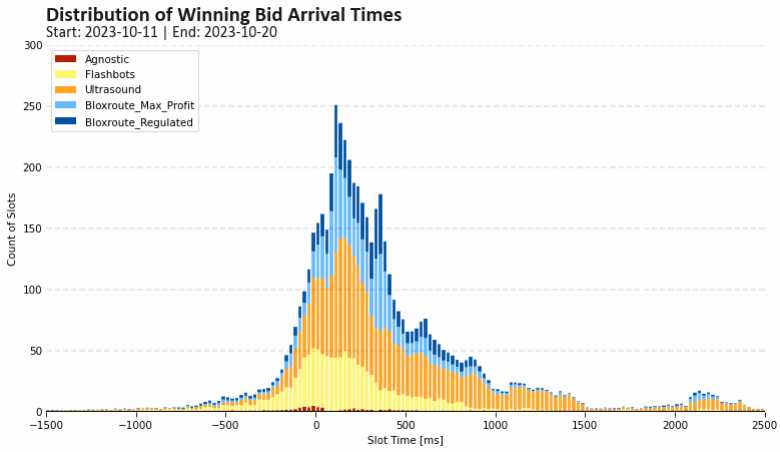



Figure 10: Despite the behaviour of proposers being mostly unchanged, the shift to JIT delivery by bloXroute relays makes it seem as if the majority of the network suddenly began playing timing games.

We suggest that analysts begin looking at deltas between the winning bid time for a slot and the median winning bid time at the relay during the 12-hour window. This methodology adds complexity and noise to individual measurements, but allows us to continue comparing results on longer timescales.

**Proposers**

P2P was the first proposer set to[explicitly advertise] that they were going to begin playing timing games. Despite maintaining a very low forked block rate, they have faced immense backlash from the community. Lost in the vitriol was the reality that, before P2P began delaying their block proposals, they consistently published blocks earlier than the majority of the network. The efficiency of their setup hurt their yield generation and made them uncompetitive. Figure 11 serves as a case study to show the very visible change in both raw and normalized bid arrival times.

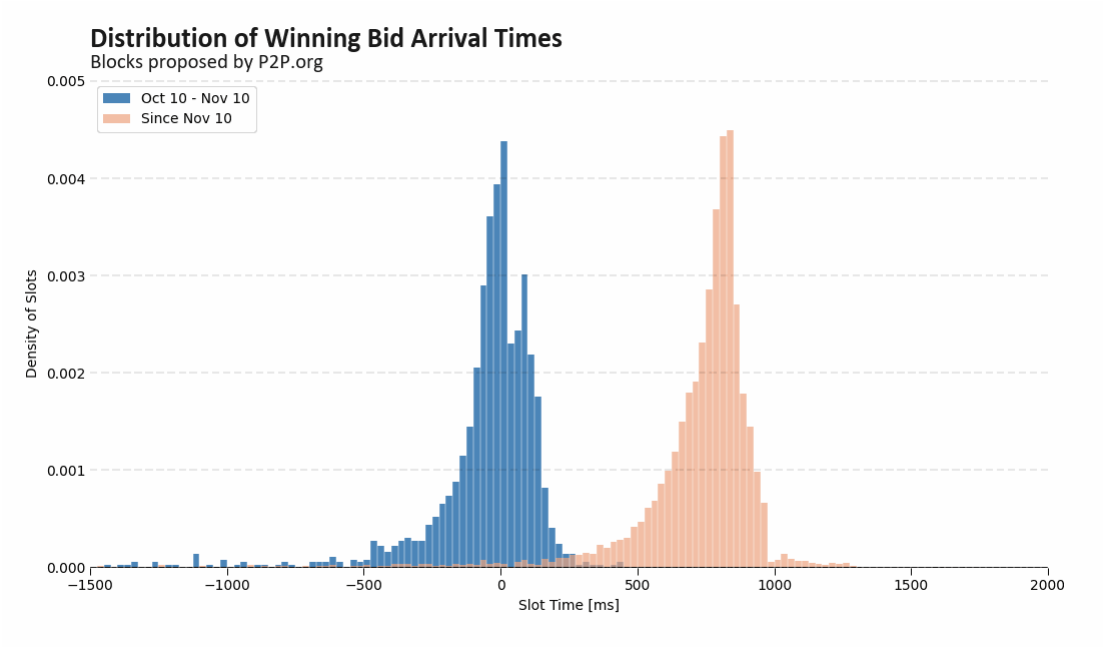



Figure 11: Normalizing P2P's winning bid arrival times we see that after accounting for changes at the relay level they are still playing timing games.

To demonstrate the importance of using normalized times, we use Rocket Pool as a control sample in Figure 12. Taking the raw winning bid times at face value, it appears to be time for the community to sharpen their pitchforks, yet once we adjust for the changes made by bloXroute we see that the normalized times are unchanged.

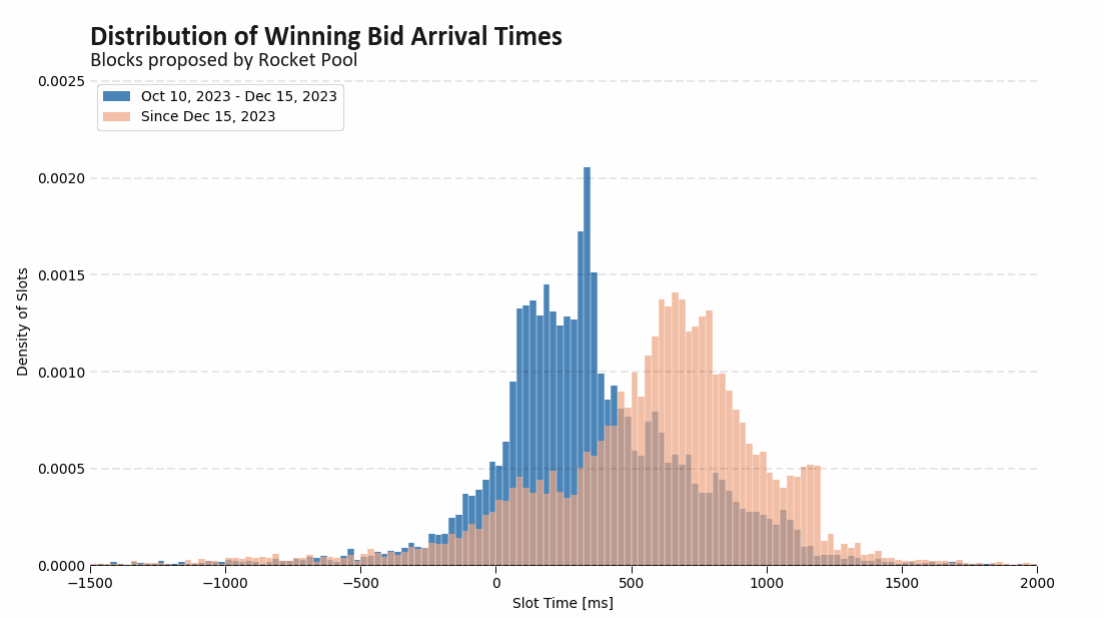



Figure 12: We use Rocket Pool as a control to show that if we do not account for changes at the relay level, even the most Ethereum-Aligned of proposers appear to have begun playing timing games.

Chorus One serves as another cautionary tale. Last year they ran a pilot project, dubbed*Adagio*, to examine the[artificial cost of latency in PBS]. Following the study, they[adopted the client modifications] for all their validators but did not adopt the changes on the nodes they operate for Lido. This decision creates confusion because Chorus One's Lido pubkeys are known but their other pubkeys are not well tagged. Analysts looking at the Lido dataset will see a change in the raw values; normalization reveals that the shift is explained by changes at the relay level.

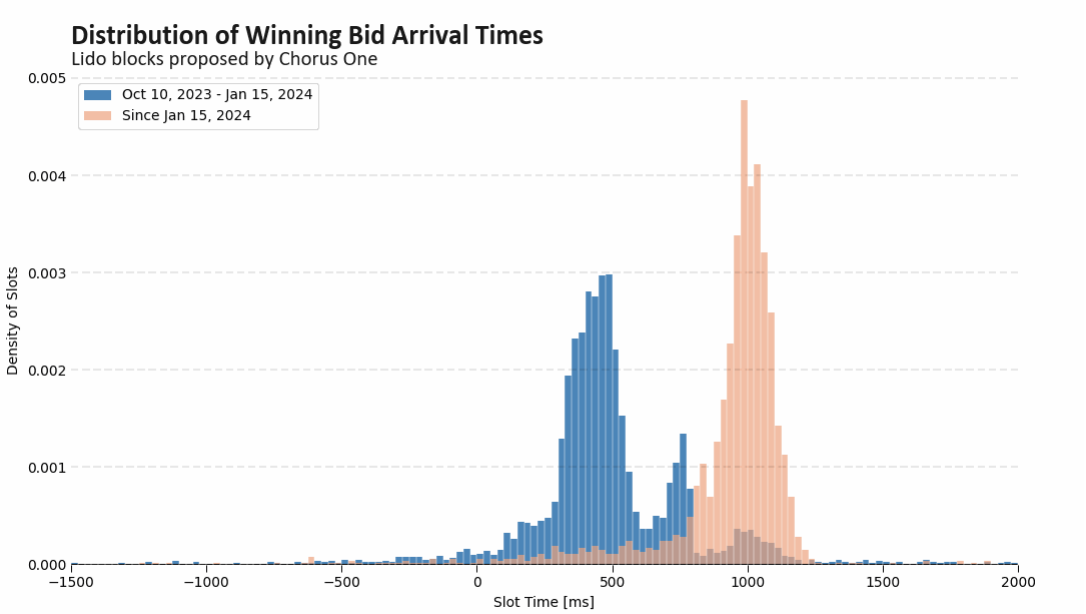



Figure 13: The scarcity of proposer pubkey mappings and lack of public data on relay$_{\text{getHeader()}}$ request reception times leads to false positives when identifying proposers who are playing timing games. Chorus One shined a light on themselves, but the only data available for their proposers is for a subset that is not engaging in timing games.

### Network Stability

The greatest fear around timing games is that they may cause degradation of consensus. Despite the clear logical inference, i.e., delayed blocks having a lower margin for error, we see that the majority of entities suspected of playing timing games are doing it responsibly. Looking at comparative median slot times and forked block rates in Figure 14, the two main outliers are CoinSpot and Coinbase, both of whom are centralized exchanges that treat staking as a secondary business. If the goal of the social layer is to reduce the total number of missed slots, then it should apply pressure to Coinbase, who is currently responsible for approximately half of all forked blocks.

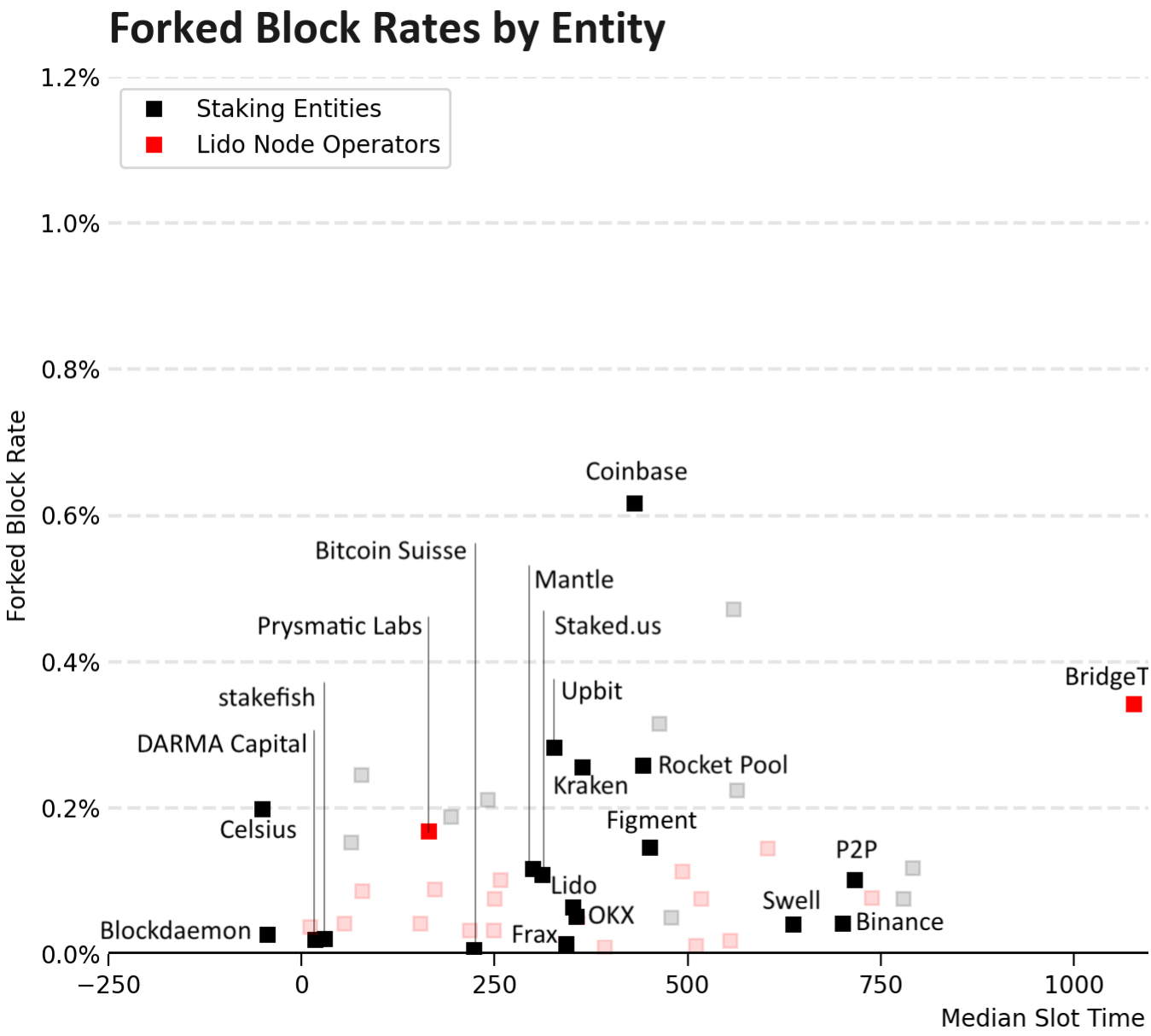



Figure 14: Despite having a much later median winning bid time, Kiln and Attestant do not see elevated forked block rates. CoinSpot and Coinbase stand out as the two major outliers.

## Optimization by Entity

### Coinbase

As the second largest proposer set, and the largest entity that is not solely focused on staking, it is no surprise that Coinbase accounts for a significant share the of forked blocks on the network. More notably, seen in Figure 15, the distribution of Coinbase's forked blocks is skewed later than the rest of the network suggesting that they may be less able to handle additional latency.

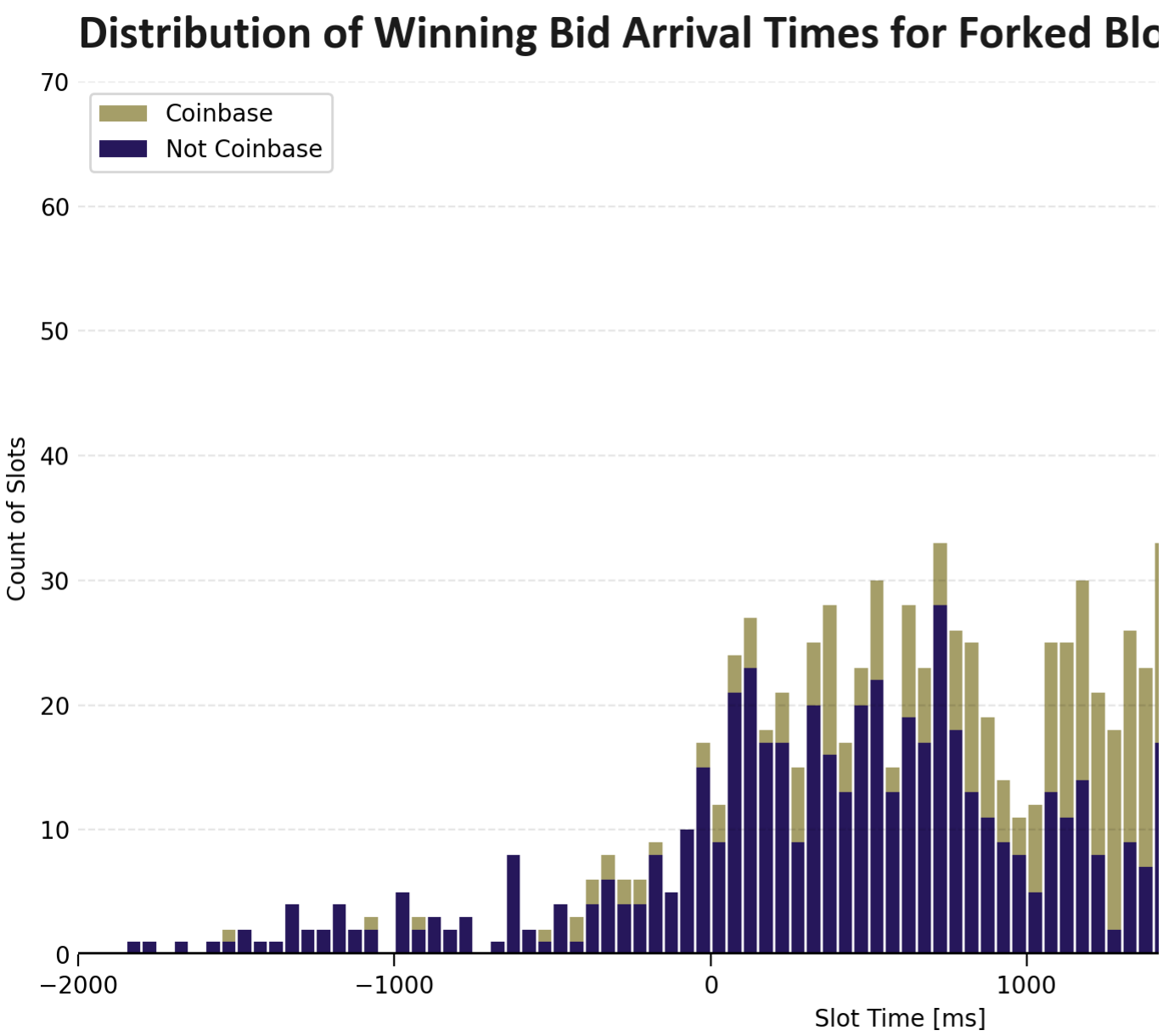



Figure 15: The distribution of Coinbase's forked block times is significantly skewed compared to the rest of the network. This suggests that latency is a more important factor for Coinbase than the rest of the network and that they should work on optimizing their setups or begin calling getHeader() earlier.

The existence of a large independent dataset from a single proposer allows us to cleanly characterize the time dependence of forked blocks for a given setup in Figure 16. The disappearance of the second maximum, originally seen in Figure 4, demonstrates that the global distribution is actually the superposition of two or more independent distributions; the minimum near 2200 ms was an artifact from professional node operators that are already playing timing games.

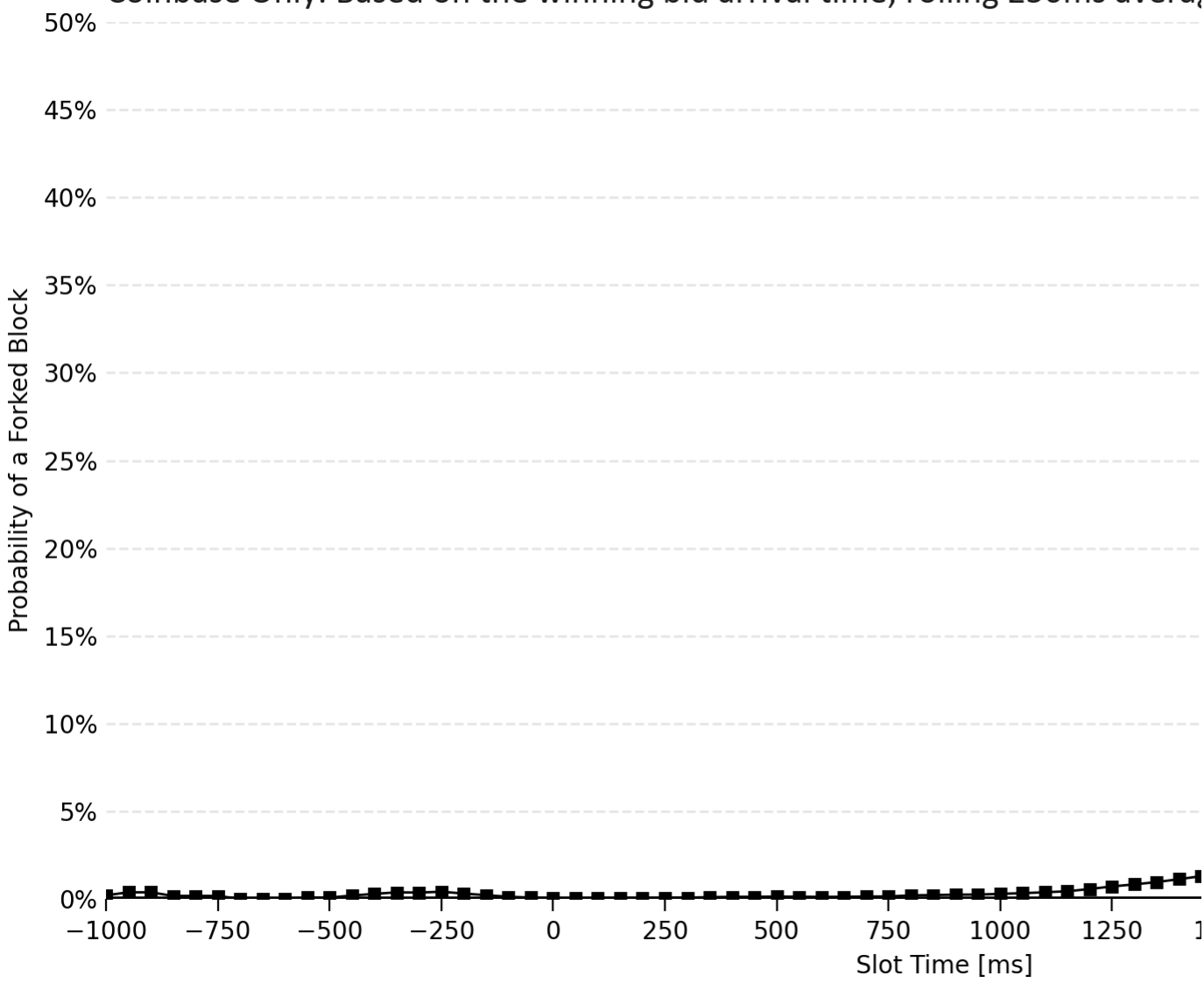



Figure 16: Coinbase begins to see an increased forked block rate for winning bids as early as 1500 ms. While the rest of the network is largely able to handle blocks that arrive significantly later than this, Coinbase's proposer setup is too inefficient handle later blocks.

Coinbase has the highest variance of any proposer set that doesn't outsource their validators to multiple node operators. The skew of this variance supplements their yield generation, but also results in high forked block rates, which will make it challenging for Coinbase to play additional timing games.

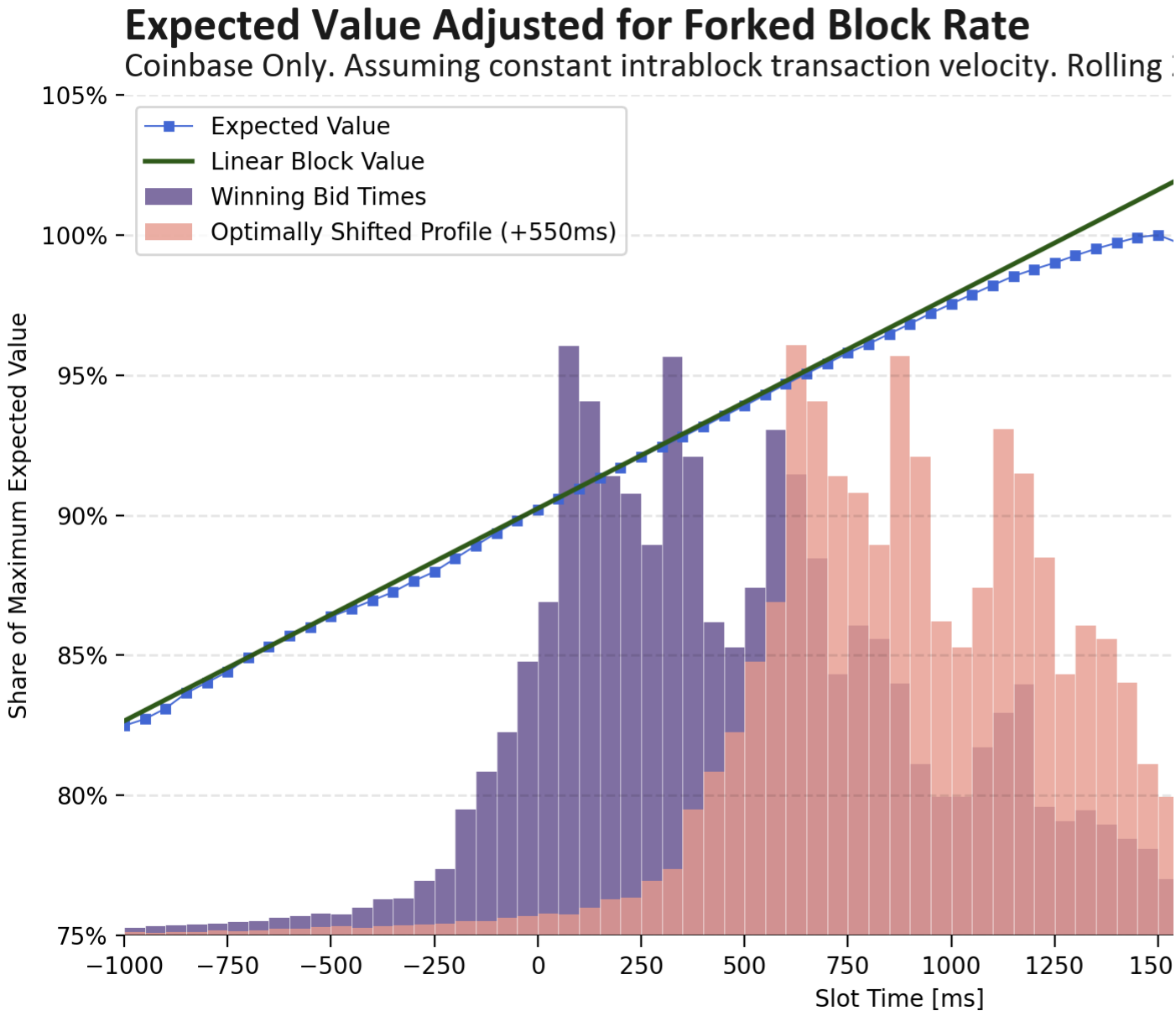



Figure 17: Coinbase could optimize the yield of their current proposer setup by delaying their getHeader() requests by 550 ms. Because Coinbase is the least efficient major proposer set and because they have such a large long tail of late blocks, this would result in a large increase in their forked block rate and only a marginal increase in execution layer yield.

In order for Coinbase to maximize their expected yield without improving the efficiency of their validator setup, they could delay their getHeader() requests by 550 ms, seen in Figure 17, but this would only increase their APR by approximately 2% and would further increase their count of missed slots. The priority for Coinbase should be to re-evaluate and streamline their proposer setup.

**Live Timing Games**

The initial backlash directed at P2P accused the node operator of "trading off chain stability against more MEV profits", but the data don't support this. When we isolate high-quality proposer sets who are known or suspected of playing timing games (P2P, Kiln, and Attestant) it's clear that they are playing the games responsibly; even though these entities are submitting late getHeader() requests, they are not being delayed long enough to be forked. In Figure 18 we see that the forked block rates for these operators does not increase later in the slot, suggesting that root cause of these forks is likely not time-dependent.

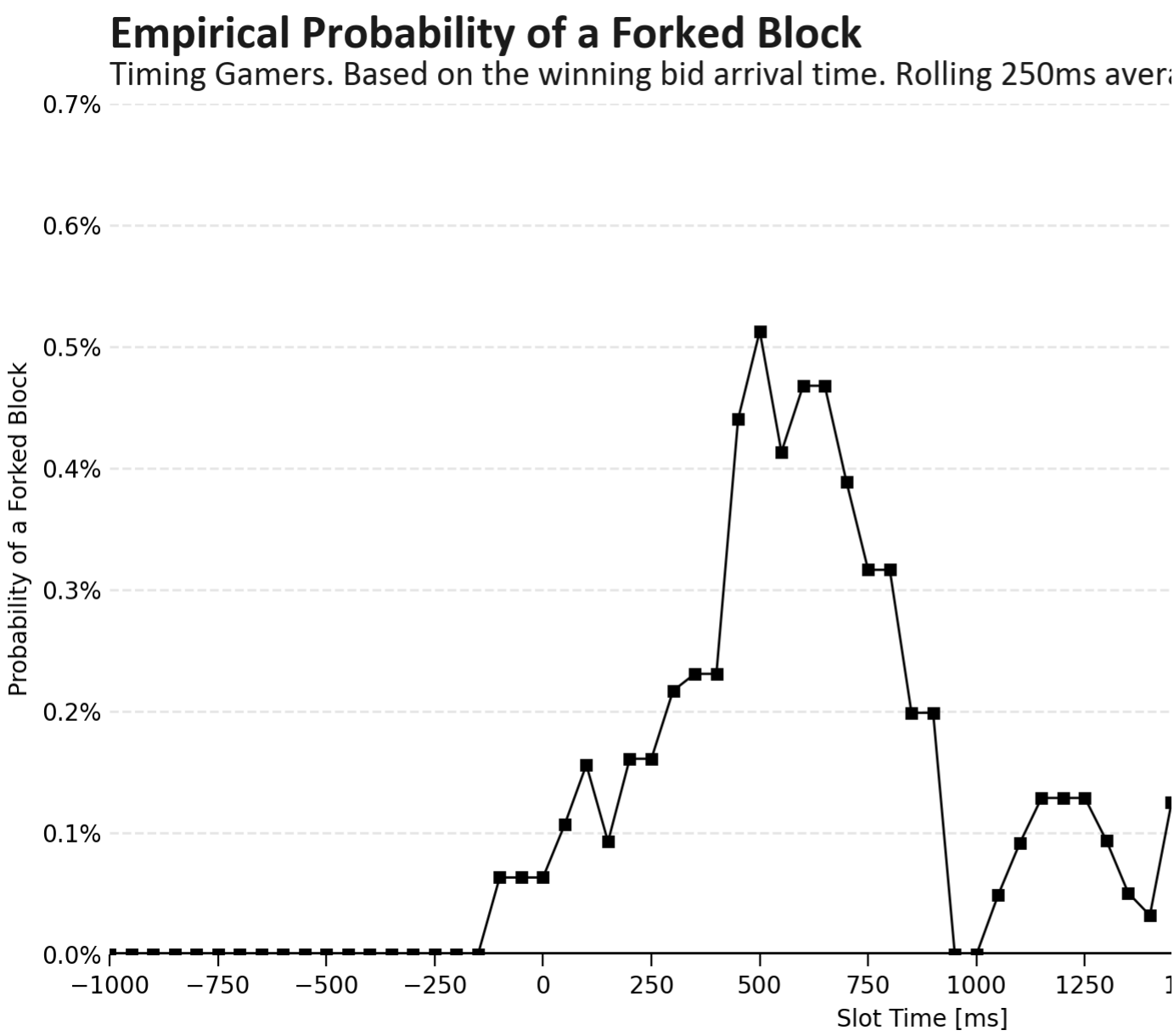



Figure 18: High-quality node operators that are playing timing games do not see an increase in forked block rates as winning bid times increase. This suggests that these operators are not operating at their limit and could further delay blocks to optimize their yield.

In Figure 19 we see multiple peaks because this is a superposition of the three operators. The late peak represents blocks from Kiln and the earlier peaks are from Attestant and P2P who both could likely handle more aggressive timing games. Because this group of proposers hasn't pushed the limit, we cannot model an optimal profile for them.

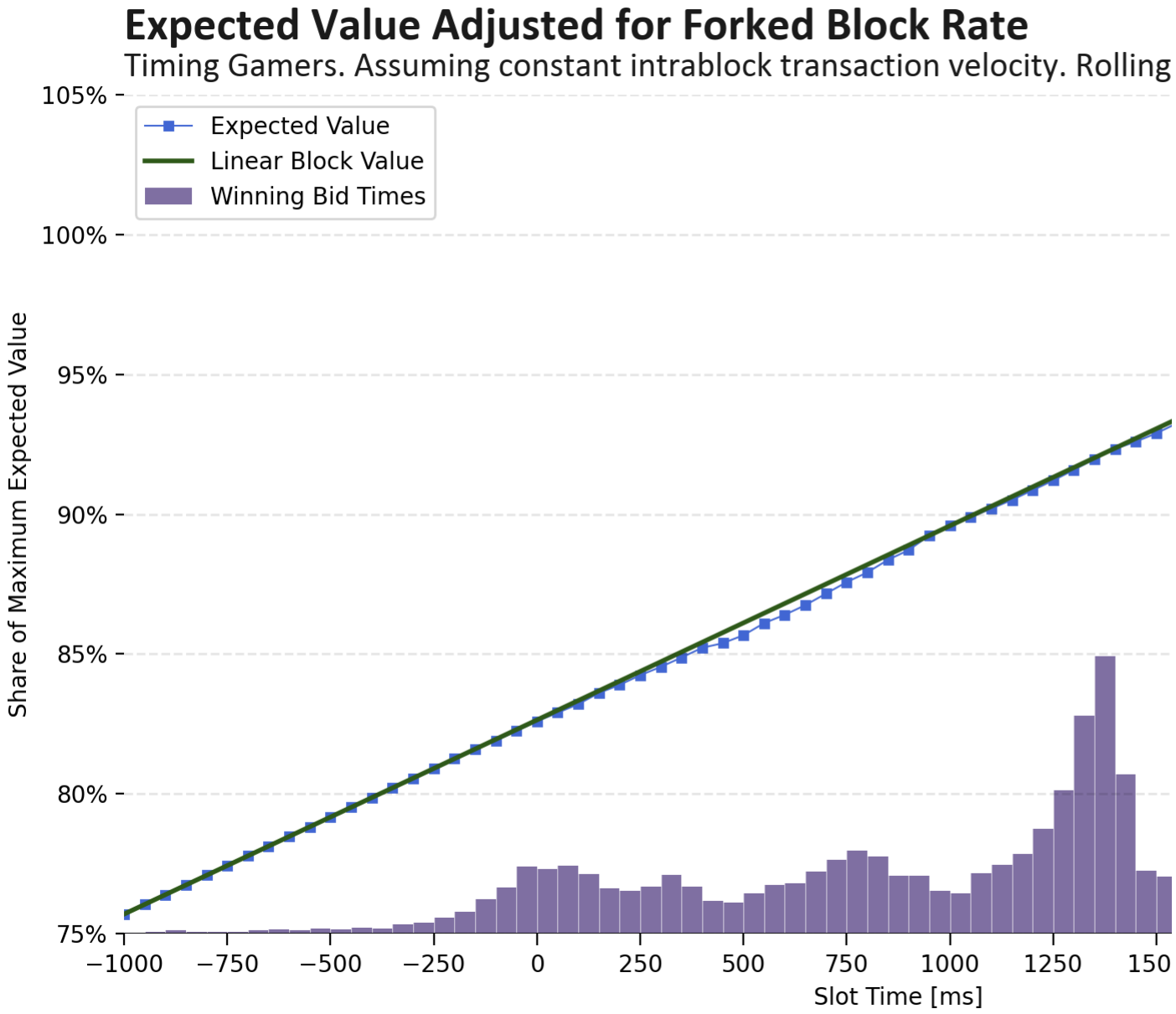



Figure 19: Due to their consistently low forked block rates, we can't optimize the bid timing of this group. Since they're already playing timing games, they should have more sophisticated in-house tools to optimize their bid timing anyway.

**Optimization by Group**

One of the biggest challenges this analysis faces is the temptation to treat dissimilar groups alike; because of data scarcity, there are many entities we cannot model on their own. We decided to break the remainder of proposers into three groups (Figure 20): efficient node operators with low variance in their bid profiles who submit blocks near the beginning of the slot, moderately efficient operators who submit blocks later but still have low variance, and noisy operators whose profiles have higher variance but that still submit blocks early in the slot.

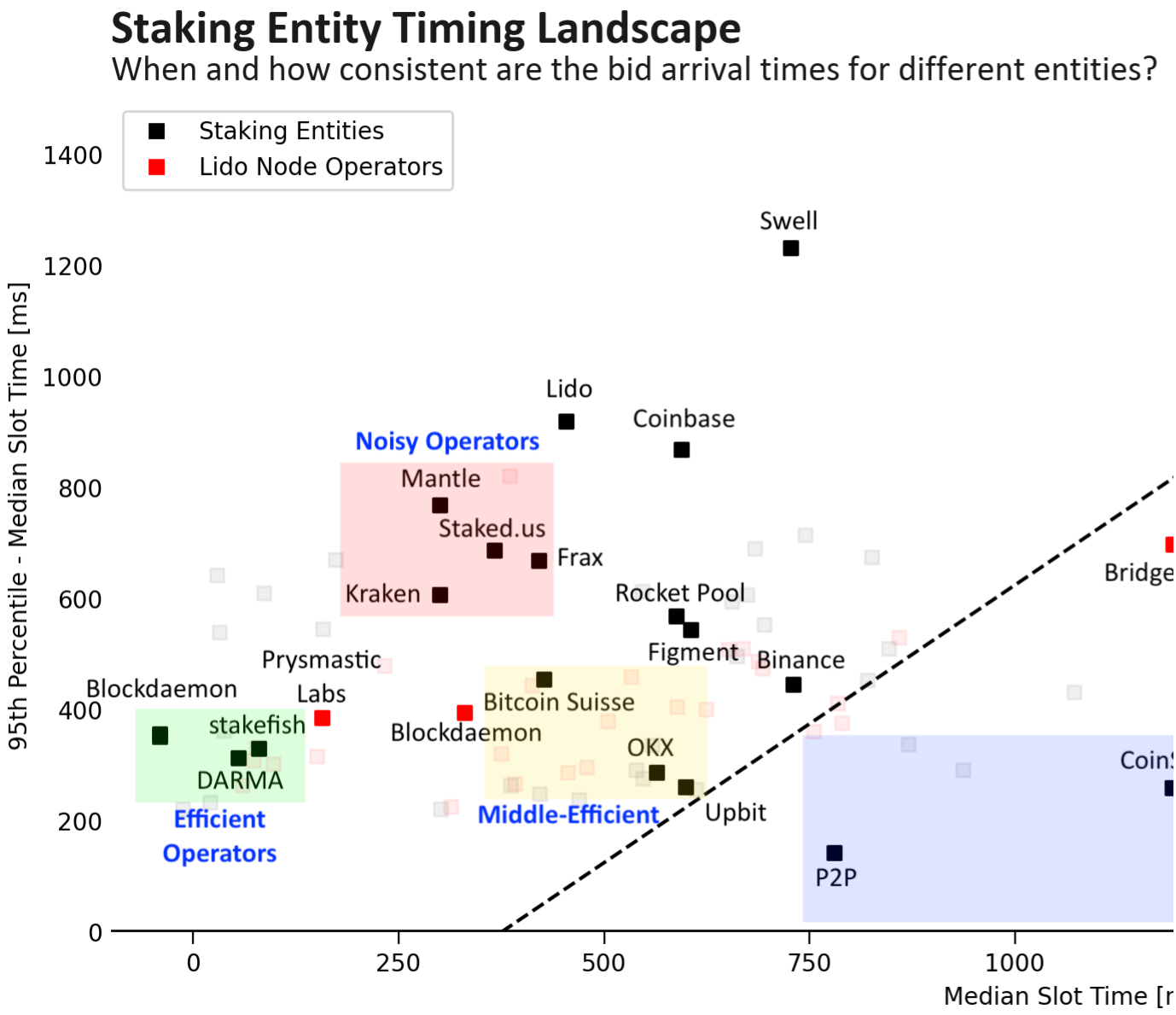



Figure 20: Proposer sets fall into a few distinct categories allowing us to treat small subsets together. Entities that outsource the operation of their validators (Lido and Swell) have outsized variance because their individual operators do not necessarily have similar profiles. The entities that would benefit the most from playing timing games are those in the efficient operator subset; they could shift their profiles near Kiln and likely keep low forked block rates.

Unfortunately, we still have some data scarcity issues. As a result, we use the forked block probability of the three groups together. This does a disservice to the hyper-efficient subset of operators, but at this point it's the best we can do. When we look at this combined group of proposers, we don't find empirical evidence of timing games. In Figure 21, winning bids received before 1200 ms are at very low risk of being forked, bids between 1200 ms and 2500 ms see heightened but relatively time-insensitive fork rates, and bids after 2500 ms see a large spike.

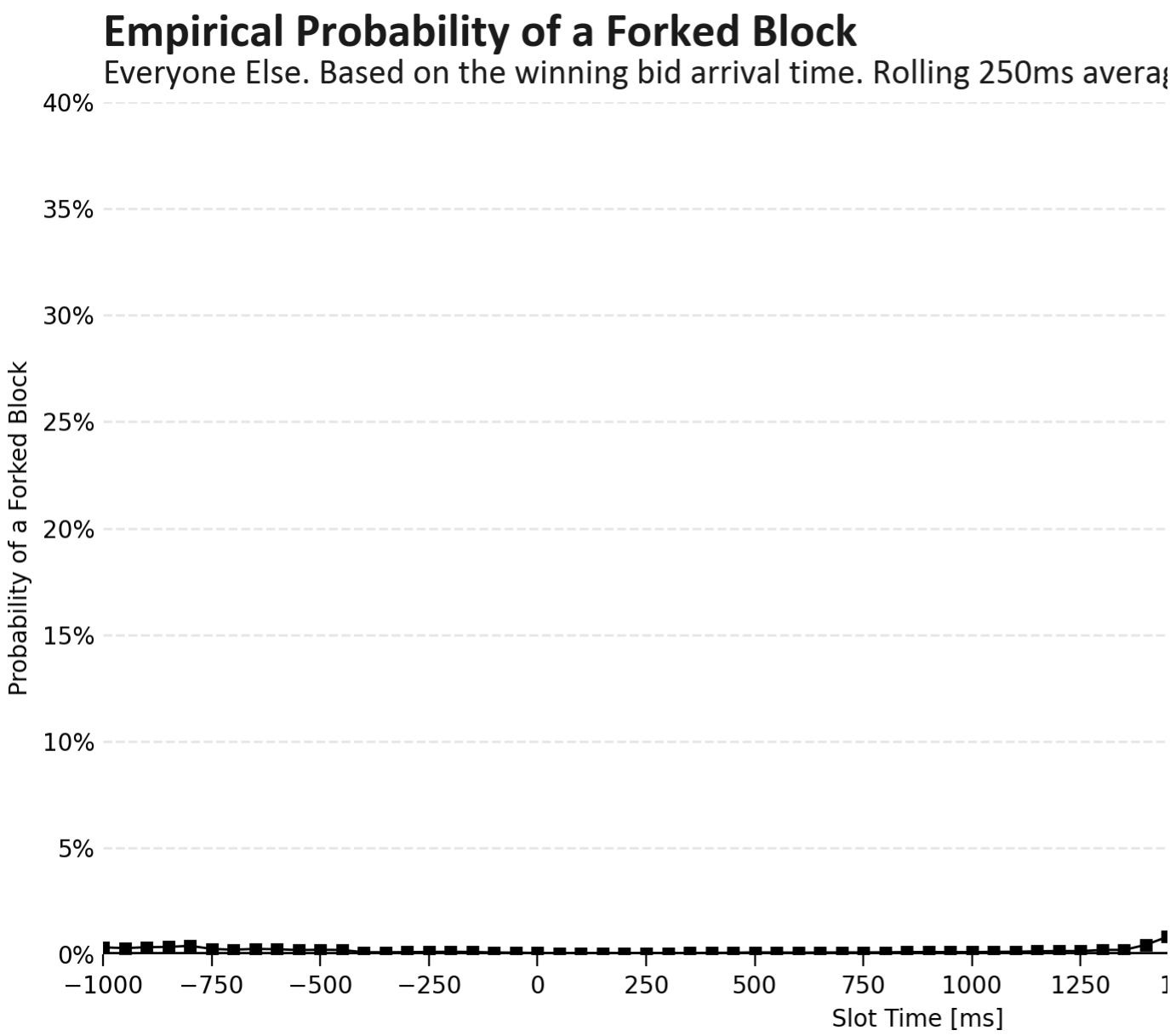



Figure 21: Eliminating Coinbase as well as proposers who are already playing timing games, we see that bids prior to 1250 ms are almost never forked and that fork rates remain low up to bid times of approximately 2500 ms.

Starting with the efficient node operator subset, this low variance group has a median winning bid arrival time of 10 ms. This group should be capable of delaying their getHeader() requests significantly to optimize their yield. Our estimate of their optimal shift is 2200 ms, seen in Figure 22, which has the potential to increase their median execution yield payments by approximately 15%.

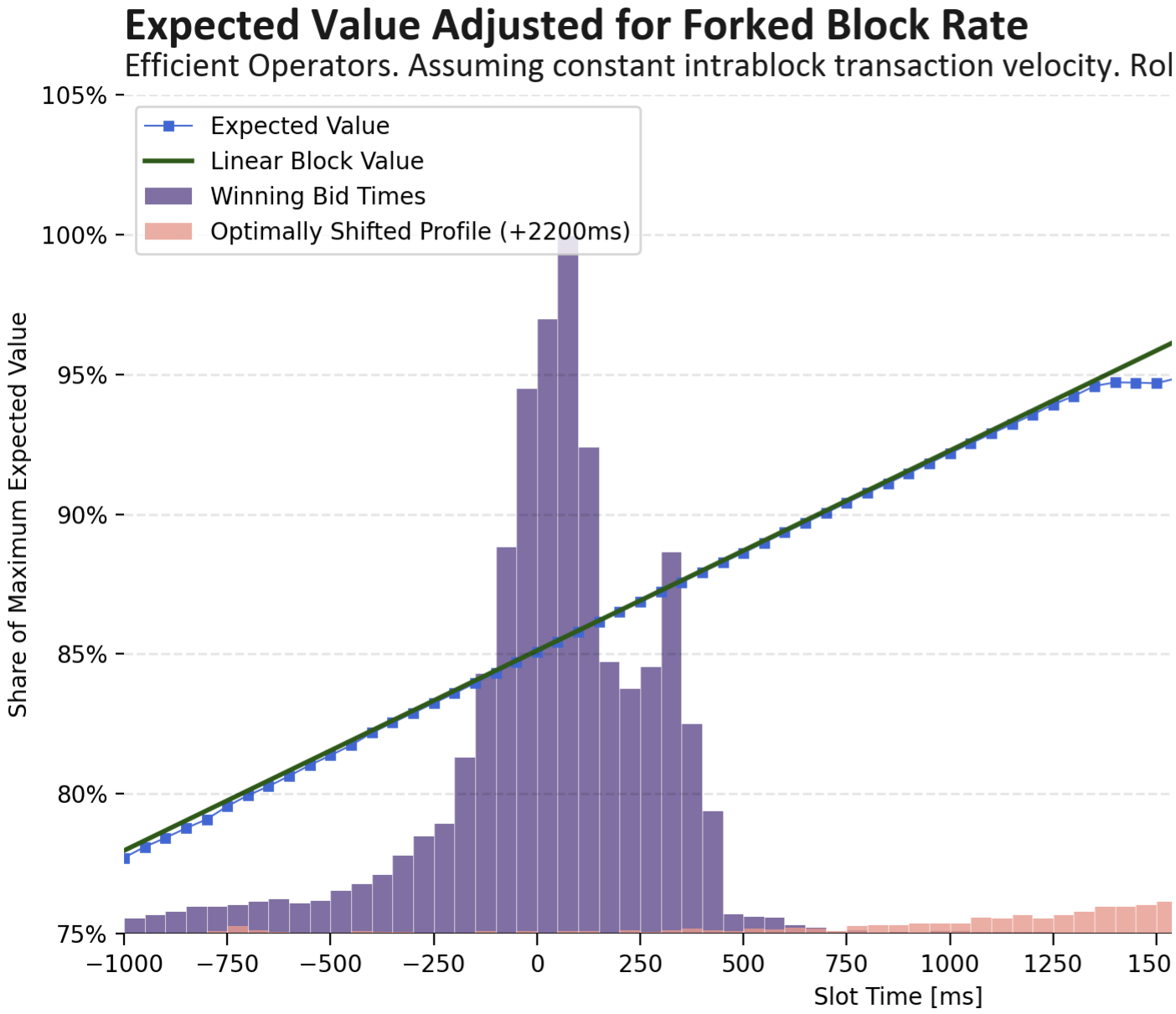



Figure 22: The efficient subset of node operators are the best positioned to begin playing timing games. Because of their high reliability and low variance in bid profiles, this group should be able to increase their yield by delaying their getHeader() requests by up to 2200 ms and significantly deepening the mempool.

The group of moderately efficient operators could optimize their yield by delaying theirgetHeader() requests by up to 1600 ms.

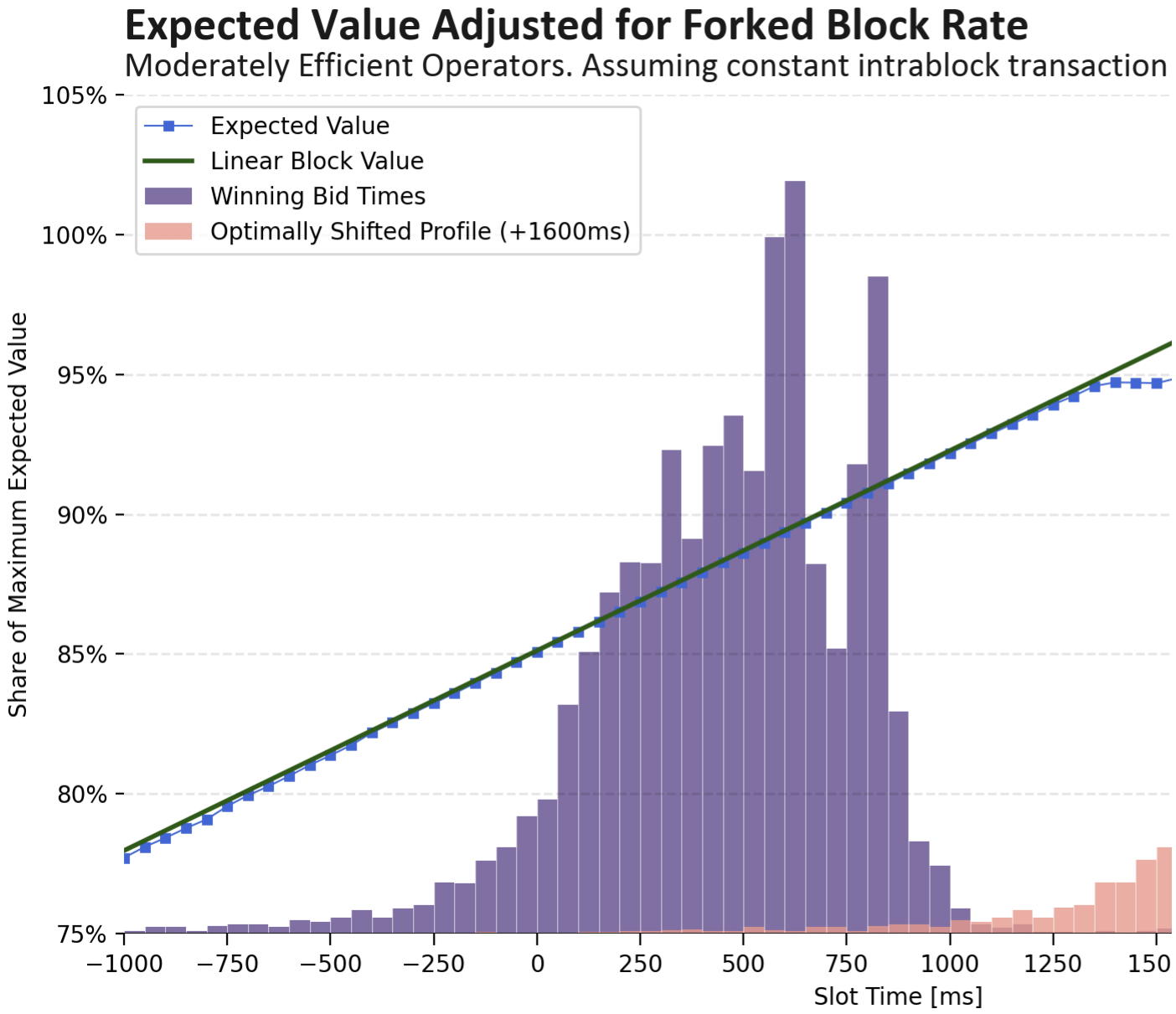



Figure 23: Moderately efficient operators are well positioned to play timing games. They can optimize their yield by delaying their `getHeader()` requests by up to 1600 ms.

The noisy operators group has nearly the same optimal time shift as the group of moderately efficient operators, but the shape of the profile is different. The wider distribution in Figure 24 for the noisy operator set pushes their optimal median winning bid time 400 ms earlier than the group of moderately efficient operators.

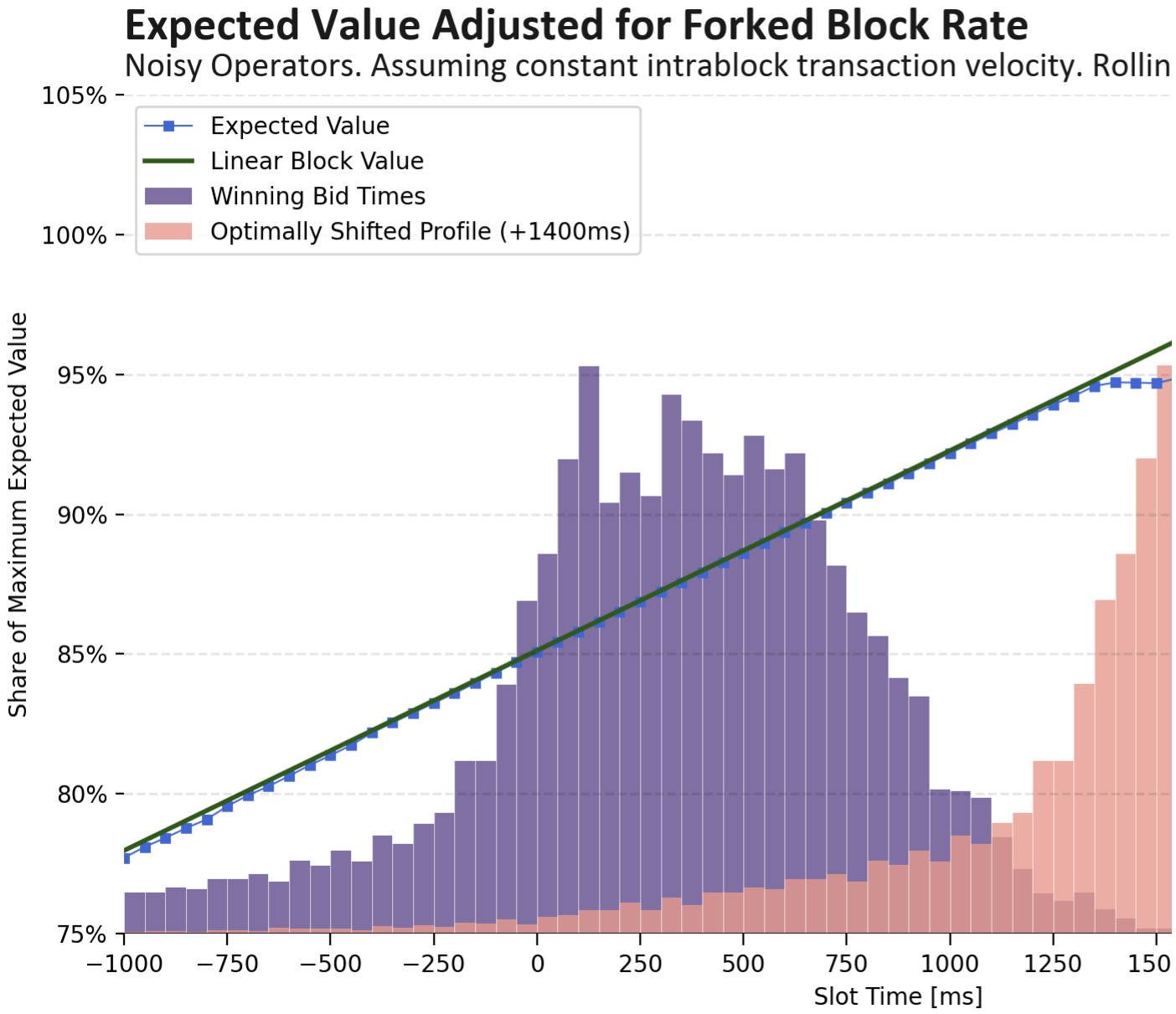



Figure 24: The noisy operator subset still has significant yield to gain by playing timing games, but the high variance of their winning bid profiles keeps the median lower than more consistent entities.

## Advanced Strategies

Naïve actors may choose to simply delay their blocks, but other optimizations exist that should be further studied.

**Proposer colocation:** One of the reasons why Kiln is able to be such an effective proposer while requesting headers so late into the block is likely because of their geographic location. Kiln is headquartered in Paris, France, while the ultrasound relay is located inOVH Roubaix, France about 200 km away. It is likely that Kiln's servers are located even closer than this.

**Relay staggering:** One of the biggest risks with pushing timing games to their limit is not getting a response in time from the relay that you're connecting to. To alleviate this, we would suggest modifying one's client to make calls to different relays at different times. For example, a proposer would request a header from Flashbots at +1000 ms and another from ultrasound at +2000 ms. If the header from ultrasound is not returned in time, then the proposer can fall back to the earlier Flashbots header and decrease their rate of forked blocks without reverting to local block building.

**Timing game obfuscation:** Due to the social backlash against proposers playing timing games it is likely that some proposer sets will desire to hide the magnitude of the games they choose to play. By implementing relay staggering and adding a bid-delta parameter, proposers can choose to play timing games on only the most valuable of blocks and revert to their normal behaviour when the added value is small.

**Value-dependent timing games:** The median block has linearly increasing value, but frequently blocks are proposed that derive the majority of their value from a small subset of transactions. In these cases the bid profiles are closer to step functions than constant slope lines. By monitoring bid values earlier in the block, proposers can watch for these step function blocks and choose not to play timing games in order to not risk high value blocks being forked.

**Volatility-dependent timing games:** Non-atomic MEV (mostly CeFi--DeFi arbitrage) becomes more valuable during periods of market volatility. Value in these blocks is heavily weighted towards the end of the slot, suggesting that timing games in these blocks should be more valuable to play than in ordinary blocks. In a further attempt to obfuscate the magnitude of a proposer set's timing games, proposers could choose to play timing games only during periods of price volatility.

## Discussion

edit: Caspar was kind enough to respond to the discussion and add more context to his quote. His comments are available inAppendix A.

When examining the implications of timing games, Caspar Schwarz-Schilling recently claimed:

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

We believe that statement is well intentioned but inaccurate. In practice, the protocol specification is not a level playing field. The status quo prioritizes poorly configured entities like Coinbase, while honest professional operators like Blockdaemon and stakefish are hurt by their prioritization of quality.

Maximal timing games do not result in the same equilibrium. Instead, they rebalance rewards to favour desirable traits present in high-reliability proposers. It is these same high-reliability proposers that have the power to accelerate timing games. We should therefore expect them to follow their incentives.

## Open Questions

- How can we reliably determine which proposers are playing timing games? Is there additional data that we need to identify these proposers?
- Will the relay API spec be changed to provide more data on `getHeader()` call and reception times?
- As relays continue optimizing JIT header delivery, how can we best model `getHeader()` request times?
- How can we quantitatively evaluate the negative externalities of timing games? Are there cases (e.g., Coinbase) where the network would be better off if proposers did optimize their setups and play timing games?
- Are there other MEV-Burn models that don't encourage proposers to play timing games? How does predictive MEV-Burn change the incentives?
- Are missed slot penalties a powerful enough mechanic to limit the timing games played by proposers?
- Should proposer sets with poorly optimized setups get to benefit from unintentional timing games?
- Is Attestant actually playing timing games or is their `min-bid` control flow resulting in high latency? Does it matter if the delays are intentional?
- How can we keep decentralized proposer sets competitive on yield as centralized entities begin playing more timing games?
- How much of a role does block size play in forked block rates? Does it vary by proposer? What other causes of forked blocks are there?
- If minimum viable issuance is adopted and execution layer rewards become a more important factor in validator yield, how much worse should we expect timing games to become?
- Does the identity of the subsequent proposer affect forked block rates?
- How often do payload requests time-out and revert to local block building? Does this inflate the apparent yield of some entities?
- How can we encourage better mapping of proposer pubkeys to analyze poorly tagged proposer sets?

## Appendix A: Caspar's Response to Discussion

Source thread

> Great analysis and cool to see more research on timing games!
>
> However, you conclude your writing with a discussion of a quote of our writing with Mike Neuder in a way that I think deserves a bit more context. While indeed a quote from our intro, in a later section discussing this comparison, we qualify our statement precisely:
>
> > "The equilibrium of timing games is identical in payoffs to everyone playing honestly (assuming relay enforcement via option 2, see below, or a simple change to mev-boost that is already discussed".
>
> -Source
>
> For reasons very specific to MEV-Boost it is true that high latency players outperform low latency players in the "everyone-plays-honestly" case.
>
> Intuition: honest players request a header at t=0, the request of high latency players will reach the relay later than that of a low latency player. As a result, high latency player signs a later bid (more mev).
>
> We have a section dedicated to comparing low/high latency players and honest/rational players in mev-boost and how honest protocol participation can be indistinguishable from rational validators playing timing games.
>
> High latency players outperforming low latency players in the "everyone-plays-honestly" case is *not* a fundamentally true property. Instead it is a quirk of mev-boost.
>
> We mention two possible ways of making our statement precise: (1) bid streaming: proposers stream bids continuously and sign the latest bid when they decide it is time to propose a block (2) Relays rejecting getHeader requests after t=0. Both of these options restore the relative latency advantage of a low latency player in the "everyone-plays-honestly" case.
>
> The reason we made the statement so loosely is that (1) I believe it offers a very important intuition: timing games don't magically create value. The main benefit of someone playing timing games comes from not everyone playing them. But of course being precise matters, and we should have alluded to the qualification alrdy in intro.