Blockchain Space Tokenization

Aggelos Kiayias
University of Edinburgh, UK
IOG, London, UK

Elias Koutsoupias
University of Oxford, UK

Philip Lazos
Jump Trading, London, UK

Giorgos Panagiotakos
IOG, Athens, Greece

Abstract

Handling congestion in blockchain systems is a fundamental problem given that the security and decentralization objectives of such systems lead to designs that compromise on (horizontal) scalability (what sometimes is referred to as the "blockchain trilemma"). Motivated by this, we focus on the question whether it is possible to design a transaction inclusion policy for block producers that facilitates *fee* and *delay predictability* while being *incentive compatible* at the same time.

Reconciling these three properties is seemingly paradoxical given that the dominant approach to transaction processing is based on first-price auctions (e.g., as in Bitcoin) or dynamic adjustment of the minimum admissible fee (e.g. as in Ethereum EIP-1559) something that breaks fee predictability. At the same time, in fixed fee mechanisms (e.g., as in Cardano), fees are trivially predictable but are subject to relatively inexpensive bribing or denial of service attacks where transactions may be delayed indefinitely by a well funded attacker, hence breaking delay predictability.

In this work, we set out to address this problem by putting forward *blockchain space tokenization* (BST), namely a new capability of a blockchain system to tokenize its capacity for transactions and allocate it to interested users who are willing to pay ahead of time for the ability to post transactions regularly for a period of time. We analyze our system in the face of worst-case transaction-processing attacks by introducing a security game played between the mempool mechanism and an adversary. Leveraging this framework, we prove that BST offers predictable and asymptotically optimal delays, predictable fees, and is incentive compatible, thus answering the question posed in the affirmative.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Blockchain protocols, Predictable Service, Transaction Fees

Digital Object Identifier 10.4230/LIPIcs.AFT.2024.9

Related Version *Full Version*: `https://eprint.iacr.org/2024/1154.pdf`

Acknowledgements We would like to thank the anonymous reviewers for their valuable comments.

1 Introduction

Blockchain systems have bounded throughput and as a result at times of congestion they can process only a portion of the transactions submitted. Combining this with the need to unambiguously serialize transactions in order to determine the state of the underlying ledger, it is imperative that a policy of transaction inclusion must be applied by the protocol.

There are three main approaches for such policy that have been implemented in popular blockchain systems: (i) prioritize based on transaction fees submitted (e.g. as in Bitcoin), (ii) introduce a fee threshold for transaction inclusion that is dynamically adjusted at times of congestion (e.g., as in Ethereum currently[1]), (iii) fix fees deterministically as a function of the transaction itself and prioritize strictly based on a FIFO policy, (e.g., as in Cardano).

Observe that the above three approaches suggest also three different service models. In terms of pricing, the first one is auction based, the second is posted price but with a dynamically adjusted price based on level of congestion and the last one is fixed price irrespective of demand. Regarding inclusion delay, the first two approaches offer (subject to consensus) instant inclusion, as long as the user is willing to pay a high enough fee, while in the third approach the inclusion delay can grow unboundedly depending on congestion[2]. It follows that in the first two approaches the inclusion delay is *predictable*, in the sense that it is known ahead of time (say 1 day earlier), while prices are unpredictable as they can go up in an unbounded manner at times of congestion and are only known one block before inclusion. On the converse, in the third approach the price is fixed and thus predictable (in the native currency of the system), while the inclusion delay is unpredictable heavily depending on congestion.

It is self-evident that service predictability is key to many applications, e.g, a company using a blockchain system would like to know ahead of time the inclusion delay expected as well as its cost to properly plan its operations. As a further example, "layer 2" protocols like lightning [22], set time bounds for the participants to challenge protocol states and failing to predict the transaction delay for participants to respond has dire security repercussions.

Furthermore, given that blockchains operate in a decentralized setting, providing *adequate incentives* for the system operators to actually follow the prescribed inclusion policy is key for its successful deployment. The first two approaches have been shown to be largely immune to collusion through off-chain agreements (aka off-chain proof [24]), i.e., the user trying to bribe his way into the system not being a profitable endeavor for both the user and the operator. However, the third one does not fare well in the face of bribes: A user can simply pay off-chain a higher fee to bypass the FIFO order of inclusion and guarantee shorter inclusion delay, with the system operator also increasing his revenue by accepting such a deal.

Predictable fees, predictable delay, and off-chain proofness, three seemingly contradicting goals, motivate the work of this paper:

> Is it possible to design a transaction inclusion mechanism that is off-chain proof and offers fee and delay predictability at the same time?

Interestingly, prior work has often sidestepped this question. Some approaches assume blocks with unlimited size (or, equivalently, a limited number of submitted transactions) [10, 21], effectively making them immune to congestion. Others prioritize features like bidding, off-chain proofness, and incentive compatibility [24] forfeiting any concrete inclusion guarantees for transactions.

---

[1] Note that due to the presence of tips, the current approach of Ethereum also combines elements of approach (i).

[2] In fact, there is a specific price tag that an attacker has to pay in order to occupy the totality of blockchain processing capacity hence denying access to other users. In the case of Cardano this is in the order of $<$ \$100 per minute, see `https://forum.cardano.org/t/cardano-network-vulnerable-to-20-minute-spam-attack/86422`.

Our results. In this work we focus on off-chain proof transaction fee mechanisms that offer predictable service guarantees. Our contributions can be summarized as follows.

- We fill a gap in the worst-case modeling of transaction waiting time by introducing a general *inclusion-delay game*. As mentioned, previous work [21, 10, 24] has focused on other challenges and has not dealt with this issue on a satisfactory level. In this *security game* the attacker interacts with a number of block producers. Following a cryptographic approach to modeling, our attacker drives the submission of transactions as well as the creation of adversarial and honest blocks. Block producers choose only what transactions to include in a block according to some policy. Given this strong adversarial setup, we are interested in the worst-case delay that an adversary can induce against a transaction measured in number of issued blocks. The advantage of our approach is that it obviates the need to explicitly model the consensus part of the system, while capturing all the pertinent elements needed for analysis, namely the *mempool mechanism* used by block producers to select transactions.

  We consider a class of mechanisms that are associated with an abstract supply of "blockchain-access" tokens. This enables to describe mechanisms with *different* delay guarantees based on the value of the tokens used by a transaction (e.g., in the case of Bitcoin transactions that pay more fees will be given priority by the mempool mechanism).
- Armed with our model, we set out now to design a mempool mechanism with the desired properties. Blockchain space tokenization (BST) issues a number of "space tokens" that each one gives certain rights to a holder to post a transaction. *The policy of the BST mechanism is to include transactions based on a priority calculated by multiplying the token value by its age (measured in blocks since its last use). To reduce congestion, only transactions exceeding a dynamically adjusted priority threshold are eligible for inclusion in a block.* We prove that BST offers asymptotically optimal and predictable delays, predictable fees, and is also off-chain proof, thus answering our main research question in the *affirmative*; optimality here refers to the worst-case delay guaranteed by the mechanisms as a function of the relative amount of token value of the holder.
- We further substantiate the real-world applicability of the BST mechanism by (i) demonstrating through a set of simulations that the inclusion delays for a variety of token distributions and user activity levels match the optimal bounds, (ii) presenting a token allocation based on a sealed bid auction that enables interested users to bid and obtain the necessary tokens, while the whole blockchain system splits the space available for transaction in two separate, fenced parts: the tokenized space and the "spot space" that accepts transactions based on a posted price mechanism as in EIP-1559, and is also used to accommodate the auction, (iii) discussing how the mechanism can be easily instantiated in both UTXO (e.g., Bitcoin) and account based (e.g. Ethereum) ledgers.

Related work. In addition to the previously cited works, we are also drawing from transaction fee mechanism design. The closest connection (due to the threshold used) is the original EIP-1559 mechanism proposed in [4] and [24]. More broadly, [13] studied the effects of delays on user utilities and prices under Bitcoin, showing that individual miners cannot profitably affect the level of fees. Significant recent results include [6], which show (among other results) that in general, no transaction fee mechanism can satisfy user incentive compatibility, miner incentive compatibility and be off-chain agreement proof all at once. The stability of EIP-1559 has been studied through the lens of dynamical systems in a string of papers [23, 15, 16], which show that even though the prices can show chaotic behavior, the block sizes on average are indeed very close to the target. There have been many interesting proposals for updating

transaction fee mechanisms such as [14] with the Monopolistic Price, and Random Sampling Optimal Price (RSOP), the latter of which was initially proposed for digital goods in [11]. The Monopolistic Price mechanism is not always truthful from the users' side, but, as they show experimentally and [27] rigorously proves, it is approximately incentive compatible when demand is high. [3, 2] propose variants of "pay-forward" mechanisms, where fees do not directly go to the miner responsible for the next block but are distributed to others as well and [8] offer another variant of EIP-1559, which is proven to be more stable, by showing that the prices exhibit a martingale property for unchanging stochastic demand. Finally, [9] considers an online lens with non-myopic users that can specify a transaction deadline and [5] studies the problem from Bayesian mechanism design perspective.

A related line of work is that on scheduling computations in multi-threaded systems or data-centers [12]. There, scarce computational resources must be efficiently allocated to match demand of variable importance. A well-known mechanism in this area is lottery scheduling [26], where processes are each assigned a number of lottery tickets, and the scheduler samples the next process proportionally to its tickets. However, such an approach is not sufficient to guarantee off-chain proofness in our setting, as the scheduler can be "bribed" to include transactions arbitrarily. This should not come as a surprise, since off-chain proofness is not a target property for multi-threaded systems or data-centers, i.e., the scheduler is always trusted, thus disallowing direct use of the mechanisms developed in this area to the blockchain setting.

Finally, financial derivatives, such as options or futures, on transaction fees offer an alternative method to ensuring predictable costs in a system with variable prices, e.g., see [25]. These approaches are not directly comparable with our solution, since they additionally require the formation of a suitable market around the derivatives. On the other hand, our solution requires a hard fork to be implemented in major architectures, unlike derivatives which can be implemented in the form of a smart contract [19].

Organization. In Section 2, we introduce the inclusion-delay security game and define what it means for a blockchain to offer predictable service. The description and the theoretical guarantees provided by the blockchain space tokenization (BST) mechanism are described in Section 3. Section 4 discusses on how tokens in BST can be instantiated and distributed, as well as the benefits of operating BST together with a traditional spot-market mechanism such as EIP-1559. Section 5 focuses on BST deployment considerations, while the performance of the mechanism is evaluated through simulation in Section 6.

2 Predictable Service

Intuitively, a user is offered predictable service if for a cost payed ahead of time, the user is certain that a transaction produced at a later time will make it to the blockchain within some predetermined delay. From a security perspective, formalizing this notion requires an adequate security model. Previous modeling attempts have sidestepped this issue, mainly by making the assumption that blocks have infinite size [10, 20]. In this section, we fill this gap by providing a adequate model for analyzing service predictability.

2.1 Predictable delay

We start, by describing a simple cryptographic (worst-case) game to analyze the inclusion-delay of transactions under different mempool mechanisms. Worst-case delay is key to ensuring predictability as it provides a *known* upper bound on blockchain inclusion.

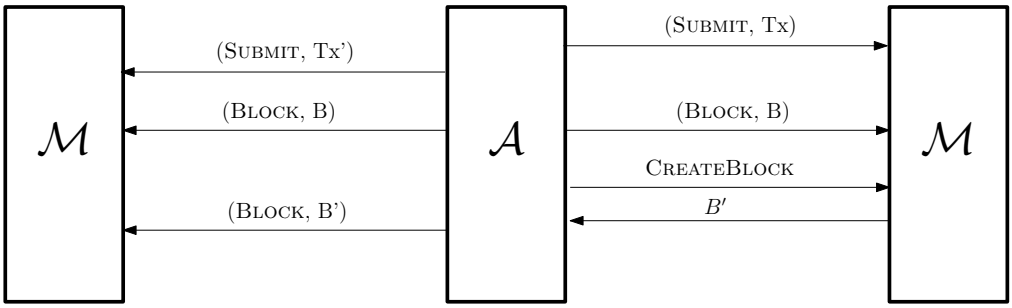

Figure 1 An overview of the messages exchanged between different parties in the inclusion-delay game. Note, that mempool instances always interact through $\mathcal{A}$.

The *inclusion-delay* game $G^{k,\mu}_{\mathcal{A},\mathcal{M}}$ is played between an adversary $\mathcal{A}$ and a number[3] of mempools running mechanism $\mathcal{M}$. The game centers around adversary $\mathcal{A}$ submitting transactions to mempools and instructing them to create blocks. $\mathcal{A}$'s objective is to maximize the time it takes for a specific transaction to be included in a block.

In detail, $\mathcal{A}$ has the following actions available:

- Submit a transaction to a mempool by sending a $(\textsc{Submit}, \text{tx})$ message.
- Issue a new block of transactions (size k) [4] by sending a $(\textsc{Block}, B)$ message to *all* mempools
- Instruct a mempool to create a new block by sending a CreateBlock message. The mempool will then notify $\mathcal{A}$ of its selection by responding with the new block B. $\mathcal{A}$ is expected to share the new block with all other mempools instances by sending a $(\textsc{Block}, B)$ message. [5]

We point to Figure 1 for an overview of the interactions in the game.

Our goal will be to design mechanisms that under suitable assumptions ensure that valid transactions appear in a block within a bounded number of new blocks created, counting from the time the transaction was submitted, no matter what $\mathcal{A}$ may do. Note, that $\mathcal{A}$ in our game is quite strong, as it fully controls transaction issuance, the delivery of messages, as well as the timing of block production. The only thing that is controlled by the mempools are the contents of the blocks they create.

Given that $\mathcal{A}$ can always create empty blocks in the inclusion-delay game, we are going to bound the adversarial block production rate, to ensure that at least some of the blocks produced are honest. Namely, we will assume that in any sequence of blocks ρ created, the adversary issues at most $\mu \cdot \rho$ of them, for $\rho \in \mathbb{N}$ and some $\mu \in [0, 1)$; μ a parameter of our model. This property holds for most state-of-the-art blockchains, and has been extensively analyzed under the name of *chain-quality* [10].

Now, we turn our attention to a subtle issue that has to do with what kind of worst-case delay guarantees can be achieved by mempool mechanisms in our game. In general, worst-case delay in bounded throughput systems (as in our game) is lower-bounded by the rate of

[3] W.l.o.g., we assume the existence of 2 mempool instances. If there was only a single instance of $\mathcal{M}$, $\mathcal{A}$ would be able to distinguish which blocks are honest and which adversarial, information that in permissionless blockchains is unavailable.

[4] Note, that if we let blocks have unbounded size the game becomes trivial; $\mathcal{M}$ includes all transactions it receives in the new block produced. This is also our main difference with previous works, e.g., [10, 20], where blocks have unbounded size, thus not capturing the transaction-level Denial-of-Service attacks we address here.

[5] Having $\mathcal{A}$ notify mempools about the creation of new blocks ensures that whether the creator of the block was a mempool or $\mathcal{A}$ is not leaked.

incoming traffic to throughput. If our game does not provide any way to limit the amount of incoming traffic, then the adversary can launch a "sybil"-attack at the transaction level to significantly delay some of the transactions produced. Concretely, if $\mathcal{A}$ wants to incur a delay of m blocks, it only has to submit $k \cdot m + 1$ valid transactions to the mempools, and then request the creation of m blocks. Since at most $k \cdot m$ transactions fit in m blocks, one of the transactions submitted will be delayed by m blocks.

To better reflect this situation within our model, and allow for shorter delays under certain preconditions, we introduce an abstract limited-supply "access token" [6] in our game; transaction creation will now be dependent/limited by token availability. In more detail, at the start of the execution $\mathcal{A}$ is expected to initialize the token distribution any way it chooses, while retaining the ability to dynamically change it at any point. Both initialization and update of the token distribution is performed by sending to *all* [7] mempools a message of the form $(\textsc{Token-Distribution}, (s_i, v_i)_i)$, where s_i is the unique identifier of the i-th token in the list that has value v_i. [8] Issued transactions may use any number of tokens, with the more token value used linked to lower delay. We point to Figure 2 for the augmented game $\tilde{G}^{k,\mu}_{\mathcal{A},\mathcal{M}}$.

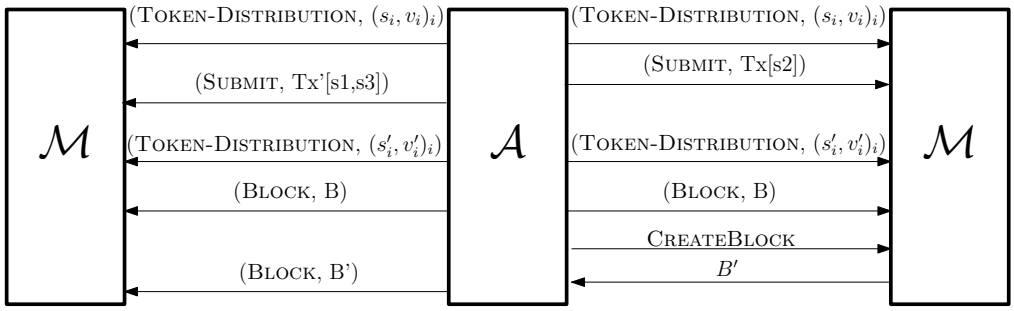

Figure 2 An overview of the augmented delay-inclusion game $\tilde{G}^{k,\mu}_{\mathcal{A},\mathcal{M}}$. Transactions reference tokens (depicted in brackets here) to ensure lower worst-case delay.

Next, we turn our attention to formalizing transactions, blocks, and chains.

▶ **Definition 1** (Transactions). *A transaction* tx *specifies (i) the list of tokens* $(s_1, \ldots, s_m)$ *it uses, and (ii) its size denoted by* $\text{size}(\text{tx})$*. The value of each token in the list is denoted by* $\text{val}(s_i)$*. The total token value of a transaction is defined to be* $\text{val}(\text{tx}) := \sum_{i\in[m]} \text{val}(s_i)$.

▶ **Definition 2** (Blocks, Chains). *A block* $B := ((\text{tx}_i)_{i\in[m]})$ *consists of a sequence of transactions. A chain* $\mathcal{C}$ *consists of a sequence of blocks.*

Typically transactions contain much more information than what we capture here. We choose to abstract away most of it as it is irrelevant for the problem at hand. Nevertheless, our model can be easily extended to describe complex transaction descriptions such as Ethereum's account model or Cardano's EUTxO.

[6] The (crypto-)currency stake distribution is a natural candidate token distribution in real-world blockchains. As we discuss later this is not the only option available.

[7] For simplicity, in our game we avoid explicitly modeling disagreement on the token distribution. Standard techniques can be used to address this issue, e.g., the token distribution is only updated after the relevant information are confirmed by the underlying blockchain.

[8] In reality, some kind of authentication mechanism, e.g., digital signatures, will be available to prove token-ownership. Such a mechanism is not needed in our model, since we assume that all transactions are produced by the adversary.

We next focus on defining what it means for a blockchain to be valid in our setting.

▶ **Definition 3** (Validity). *A block is valid if it contains transactions with size at most k and no two transactions reference the same token. A chain is valid if it contains only valid blocks.*

As before, our validity definition can be straightforwardly extended to account for the complex validity definitions found in real-world blockchains.

To guarantee provable inclusion guarantees we will require a transaction to (i) be well-distributed in the network (cf. the liveness condition in [10]), and (ii) not use the same tokens as some other concurrently submitted transaction. Obviously, we should not expect transactions not meeting these conditions to have provable inclusion guarantees.

▶ **Definition 4** (Good transactions). *A transaction* tx *submitted in the inclusion-delay game $\tilde{G}^k_{\mathcal{A},\mathcal{M}}$ is* good *iff*

- *no other transaction* tx′ *with overlapping token references is submitted until* tx *is included in a block;*
- tx *is submitted to all mempool instances.*

Our definition of security, which we introduce next, is concerned exactly with the worst-case delay of such "good" transactions.

▶ **Definition 5** (Worst-case delay). *We say that a mempool mechanism $\mathcal{M}$ has* worst-case delay *d iff for any adversary $\mathcal{A}$, it holds that any good transaction* tx *submitted in the inclusion-delay game $\tilde{G}^k_{\mathcal{A},\mathcal{M}}$ appears in a block by the time d blocks [9] are generated, counting from the time* tx *was submitted to the last honest party.*

We note d in our analysis will be a function of certain transaction attributes, such as the transactions size and token-value. Moreover, w.l.o.g, in the rest of this work we assume that at any point of the inclusion delay game the *total token-value* is 1.

2.2 Predictable cost

Knowing that a transaction will make it to the blockchain within a predetermined amount of time is insufficient to claim predictable service, as the cost of the transaction may be unpredictable until the time it is included in the blockchain. Instead, to ensure full-fledged predictability, a user submitting a transaction to the mempool mechanism must know ahead of time what the total cost for this transaction will be. Note, that the cost does not have to be fixed (as in Cardano), it only has to be determined and possibly paid ahead of time compared to when the transaction is going to be submitted. [10] This leads us to the following natural definition of service predictability.

▶ **Definition 6** (Predictable service). *A mempool mechanism $\mathcal{M}$ offers* predictable service *to some user with predictability parameter t, if the users knows that after time t it can issue a transaction with known delay and cost.*

[9] For simplicity, here we count delay in blocks to avoid introducing time explicitly. In principle, our definition can be adapted to count delay in time units, by introducing an adequate liveness condition in the inclusion-delay game. Such conditions are known to hold for blockchain protocols, e.g, see [10, 20].

[10] Similar guarantees in finance are provided by a forward market where the price of a commodity is locked-in a lot earlier that when the commodity is going to be delivered.

As argued earlier, Bitcoin and Ethereum fail to offer predictable service to *any user*, as a surge in demand just after submitting a transaction does not allow for predicting the cost of the transaction–the price may keep increasing until the transaction is no longer a valid candidate for inclusion.

In the case of Cardano things are a bit different. Transaction costs are fixed, and thus predictable. Moreover, due to the mempools processing transactions in a FIFO order, and the fact that stake is limited and thus only a limited number of valid transactions can be generated at any given point in time, it follows that delay is bounded in the worst-case and thus also predictable. However, although Cardano provides predictable service, albeit with a very large delay, it does not provide good incentives for block producers to follow the transaction inclusion policy.

Specifically, an urgent user attempting to "jump the queue" and include its high value transaction in the blockchain fast, can simply bribe operators to include its transaction first, potentially making the delay other users experience arbitrarily large. This is possible since the mempool operator has full control of the contents of the block. Resistance to such attacks, known as off-chain proofness (OCP), is explored in [24]. OCP is concerned with collusion agreements between users and operators trying to maximize their joint utility. A mechanism is OCP if for every set of off-chain agreements, there is an equally good "on-chain" scenario. Thus, even if a user pays off-chain a mempool producer for guaranteed inclusion, neither party should gain additional utility.

Given the shortcomings these mechanisms face, in the next section we present a new mechanism that manages at the same time to achieve service predictability and be off-chain proof.

3 Blockchain Space Tokenization

In this section, we describe and analyze the *block space tokenization* (BST) mechanism $\mathcal{M}_{\texttt{bst}}$, a deterministic mempool mechanism that offers predictable service while properly incetivizing correct behavior of both users and mempool operators. The mechanism centers around the concept of *blockchain space tokens* that give their owners the right to post transactions in the blockchain at a certain rate. The main idea is to prioritize inclusion of transactions using these tokens based only on *publicly verifiable information*, such as:

- the value of the tokens;
- the age of the tokens, i.e, the block height at which a token used by the target transaction was last used;
- the size of the transaction,

and thus make it easier to achieve incentive compatibility by enforcing and *checking* correct behavior.

We proceed to first give a detailed description of the core mechanism as well as analyze its worst-case delay in the honest/adversarial model. Then, we provide a modification of the algorithm that makes it off-chain proof while at the same time arguing that the resulting mechanism retains similar worst-case delay guarantees. Finally, we argue that if tokens are obtained by users ahead of time, the mechanism indeed offers predictable service.

3.1 Mechanism description

The core component of our mechanism is the way the priority value of a transaction is calculated. Key to understanding this component is understanding the role of token age, denoted by $age(s_i)$ for token s_i, which is equal to the number of blocks generated from the last time the token was used. Token age grows when a token is not used, while it is reset to zero every time it is used.

The first idea is to make the priority of a transaction proportional to the value of a token multiplied by its age. The intuition is that, for the same token age, a transaction with more token value should be prioritized over a transaction with less value; higher value should generally mean lower delays. For the same token value, a transaction consuming an "old" token should be prioritized over a transaction consuming a "young" token. This prevents the same tokens from monopolizing the usage of the chain. In general, we could use any monotone function of token value and age, but the product of value by age is the most natural one. When a transaction uses multiple tokens, its priority is proportional to the sum of the priorities of these tokens.

The second idea is to prioritize transactions at a rate inversely proportional to their size. This ensures that, for a fixed token value, a transaction twice the size is included with half the frequency. This maintains a constant throughput consumption rate per token value, regardless of transaction size, thus ensuring that obtaining a certain amount of space token value implies a certain space consumption rate.

Finally, we set an upper bound on priority that depends on token value density, the chain quality parameter, and block size. This is necessary to prevent attacks where an adversary stockpiles low-value tokens for an extended period before releasing them all at once, monopolizing space usage with high-priority transactions. The related transactions using these tokens would by then have maximum priority. Our carefully calibrated upper bound ensures high-value transactions remain prioritized, even if low-value tokens haven't been used for a long period.

Concretely, mechanism $\mathcal{M}_{\texttt{bst}}$ assigns *priority* values to transactions as follows:

$$\text{priority}_{\mathcal{C}}(\text{tx}) := \min\left\{\frac{\sum_{s_i\in\text{tx}}\text{val}(s_i)\cdot\text{age}(s_i)}{\text{size}(\text{tx})}, \frac{4}{(1-\mu)k}\cdot\left(1+\frac{\text{val}(\text{tx})}{\text{size}(\text{tx})}\right)\right\},$$

where $\mathcal{C}$ is the chain defined by the blocks created in the inclusion-delay game up to this moment, s_i is the i-th token used by the transaction, $\text{age}(s_i)$ is the number of blocks in $\mathcal{C}$ since s_i was last used by a transaction. Observe that for a transaction tx and chain $\mathcal{C}$, $\text{priority}_{\mathcal{C}}(\text{tx})$ is completely determined, making the priority value publicly verifiable.

Having defined a priority score, $\mathcal{M}_{\texttt{bst}}$ simply fills new blocks with the transactions with the highest priority. Next, we focus on analyzing the worst-case delay guarantees of $\mathcal{M}_{\texttt{bst}}$.

3.2 Security analysis

Next, we show that $\mathcal{M}_{\texttt{bst}}$ has optimal worst-case delay up to some constant terms, i.e. in the order of $O(\frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx})\cdot k})$. [11] The optimality claim is based on the fact that there can be at most $1/\text{val}(\text{tx})$ transactions with token value $\text{val}(\text{tx})$ and size $\text{size}(\text{tx})$, and thus it takes at least $\frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx})k}$ blocks to absorb them.

The main idea of the proof is the following: Firstly, tx will reach maximum priority after a sufficient number of new blocks is generated. This implies that in order for tx to not be included in the chain after this point in time, the adversary must fill any of the subsequent honest blocks with transactions of priority greater or equal to that of tx. Given that an $(1-\mu)$ fraction of the blocks is going to be honestly generated due to our assumption, we

[11] While the constants provided by the theoretical analysis are not tight, later in Section 6 we show through simulation that our mechanism indeed achieves tightly optimal delays under normal operation. Nevertheless, the theoretical analysis is important as it establishes that the worst-case guarantees of the mechanism in an environment almost entirely controlled by the adversary remain on the same order as the optimal ones.

show that it is impossible for the adversary to fill all of them with transactions other than tx that also have a matching priority score, and thus tx is necessarily included in a block in the predicted time.

▶ **Theorem 7.** *The worst-case delay* $d(\text{tx})$ *of mechanism* $\mathcal{M}_{\texttt{bst}}$ *is upper bounded by* $16 \cdot \frac{\text{size(tx)}}{(1-\mu)\text{val(tx)}k} + 2$*, when* $\text{size}(tx) < k/2$ *and* $\text{val(tx)} < 1/2$*.*

Proof. Let tx be some good transaction in an execution of $\tilde{G}^{k,\mu}_{\mathcal{A},\mathcal{M}}$, and let $\epsilon := \text{val(tx)}$, $c := \text{size(tx)}$, and $P := \frac{4}{(1-\mu)k}$. Moreover, let $u := d(\text{tx})/2$, and note that if tx is not included in the blockchain after u new (honest or adversarial) blocks have been generated, then it has maximum priority, i.e., $\epsilon \cdot u/c \geq P(1+\epsilon/c)$. For the sake of contradiction assume that the theorem does not hold, and tx does not enter the chain after $d(\text{tx})$ blocks are generated. We are going to show that such a scenario is impossible.

Let S' denote the set of (honest or adversarial) blocks generated starting $u+1$ blocks after the submission of tx, and up to the generation of $2u$ blocks. Let $S \subseteq S'$ denote the subset of honest blocks of S'. As argued earlier, during the generation of blocks in S, tx has maximum priority equal to $P \cdot f(\epsilon/c)$, where $f(x) := 1 + x$, is a monotonically increasing function in x, and $f(x) > 1$, for any $x > 0$. We have assumed that tx is not included in these blocks, it thus follows that any block in S should contain transactions with priority greater than $Pf(\epsilon/c)$. Due to the monotonicity of f, this implies that the token-value density of any such transaction is at least ϵ/c. Moreover, any block B in S must be at least $k' := (k - c + 1)$ full, otherwise tx would be included. If follows, that the total token-value referenced in B is at least $k' \cdot \epsilon/c$, and that

$$\begin{aligned}\sum_{\text{tx}_i \in B} \text{priority}_C(\text{tx}_i) \cdot \text{size}(\text{tx}_i) &\geq \sum_{\text{tx}_i \in B} Pf(\epsilon/c) \cdot \text{size}(\text{tx}_i) \\ &\geq Pf(\epsilon/c) \cdot \sum_{\text{tx}_i \in B} \text{size}(\text{tx}_i) \\ &\geq Pf(\epsilon/c)k' \qquad (1)\end{aligned}$$

The above inequality will be useful to determine the amount of priority adversarial transactions have to generate to fill all blocks in S.

Next, we focus on upper bounding the number of blocks in S the adversary can fill with transactions other than tx. W.l.o.g., we can assume that all adversarial blocks generated in S' are empty, and that $\mathcal{A}$ includes transactions that reference all available tokens (except those that are referenced by tx) in the first m blocks of S which are honest, for some optimally selected m. Note that it is optimal for $\mathcal{A}$ to first reference *all* available tokens, as in this way it can maximize the amount of priority of transactions used to fill any remaining blocks; w.l.o.g, we assume that tokens are initially of infinite age. As a sanity check, note that just creating transactions referencing tokens once, is not sufficient to cover all blocks in S, as each block requires referencing $\epsilon k'/c$ token-value, and thus a total of $u(1-\mu)$ blocks require referencing $u(1-\mu) \cdot \epsilon k'/c \geq 8k'/k > 4$ token-value, i.e., more than 1 which is the total amount of token-value.

Next, we provide an upper bound T on the sum of value-age products of the tokens in the first m honest blocks of S at the time the last honest block in S is generated. This will be important to argue that $\mathcal{A}$ will not be able to create enough transactions with high enough priority to fill all honest blocks. We thus have:

$$T \leq \sum_{i=1}^{m} s_i(u-i) = u\sum_{i=1}^{m} s_i - \sum_{i=1}^{m} s_i i \leq u(1-\epsilon) - \sum_{i=1}^{m} s_i i \ ,$$

where s_i is the total token-value in the i-th block of S. The quantity $\sum_{i=1}^{m} s_i i$ is minimized when s_1 gets its maximal value. As we have argued earlier, each honest block references at least $\epsilon k'/c$ of token-value. Thus, s_1 is at most $1-\epsilon-(m-2+\delta)\epsilon k'/c$, where the m-th block references $\delta\epsilon k'/c$ token-value for the first time in S', for some $\delta \in (0,1]$–the rest of the token-value necessary may come from previously used tokens. It follows that:

$$\begin{aligned}\sum_{i=1}^{m} s_i i &\geq (1-\epsilon-(m-2+\delta)\epsilon k'/c) + \sum_{i=2}^{m-1} \epsilon k'/c i + m\delta\epsilon k'/c \\ &\geq 1-\epsilon-\epsilon m k'/c + \epsilon k'/c \sum_{i=1}^{m} i - (1-\delta)(m-1)\epsilon k'/c \\ &\geq 1-\epsilon-\epsilon m k'/c + \epsilon \frac{m(m+1)}{2} k'/c - (1-\delta)(m-1)\epsilon k'/c\end{aligned}$$

Putting everything together, we have that:

$$T \leq u(1-\epsilon) - (1-\epsilon-\frac{\epsilon k'}{2c}(2m - m(m+1) + 2(1-\delta)(m-1))) \tag{2}$$

By the chain quality assumption there are at least $(1-\mu)u$ honest blocks in S. We have already argued about how the first m blocks are filed. Due to Inequality 1, the sum of value-age products required to fill the rest of the blocks in S must be greater or equal than

$$((1-\mu)u - m) \cdot k' Pf(\epsilon/c) + \delta k' Pf(\epsilon/c),$$

where the second term comes from the amount of priority required to fill the half empty m-th honest block. Moreover, our initial assumption about the behavior of $\mathcal{A}$ implies that this quantity must be smaller than T. Hence, it must hold that:

$$\begin{aligned}&((1-\mu)u - m) \cdot k' Pf(\epsilon/c) + \delta k' Pf(\epsilon/c) \\ &\leq u(1-\epsilon) - (1-\epsilon-\frac{\epsilon k'}{2c}(2m - m(m+1) + 2(1-\delta)(m-1))) \Rightarrow \\ &u \leq \frac{Pf(\frac{\epsilon}{c})k'(m-\delta) - (1-\epsilon-\frac{\epsilon k'}{2c}(2m - m(m+1) + 2(1-\delta)(m-1)))}{(1-\mu)Pf(\frac{\epsilon}{c})k' - 1 + \epsilon}\end{aligned}$$

It is easy to see that the derivative over m of the r.h.s. of the above inequality is equal to 0 when

$$m = cPf(\epsilon/c)/\epsilon + 3/2 - \delta \geq cPf(\epsilon/c)/\epsilon \geq c/(k'\epsilon),$$

where the last inequality follows from the facts that $k' > k/2, f(\epsilon/c) > 1$. On the other hand, m must be less than $1/\frac{\epsilon k'}{c} = c/(k'\epsilon)$. Since the r.h.s. is a quadratic function of m, it follows that we can upper bound it (and thus upper bound u) by setting $m := c/(k'\epsilon)$. Hence, we get:

$$u \leq \frac{cPf(\epsilon/c)/\epsilon + \epsilon + 1/2 - \delta - c/(2k'\epsilon) - (1-\delta)\epsilon k'/c - \delta k' Pf(\epsilon/c)}{(1-\mu)Pf(\epsilon/c)k' - 1 + \epsilon}$$

Replacing P by $\frac{4}{(1-\mu)k}$, for the denominator we get that:

$$(1-\mu)Pf(\epsilon/c)k' \geq (1-\mu)\frac{4}{(1-\mu)k} f(\epsilon/c)k' \geq 2$$

which implies that:

$$\begin{aligned} u &\leq \frac{cPf(\epsilon/c)/\epsilon + \epsilon + 1/2 - \delta - c/(2k'\epsilon) - (1-\delta)\epsilon k'/c - \delta k' P f(\epsilon/c)}{2 - 1 + \epsilon} \\ &\leq \frac{4cf(\epsilon/c)}{(1-\mu)k\epsilon} + \epsilon + 1/2 - \delta - \frac{c}{2k'\epsilon} - (1-\delta)\epsilon k'/c - \frac{4\delta k' f(\epsilon/c)}{(1-\mu)k} \\ &\leq \frac{8c}{(1-\mu)k\epsilon} + \epsilon + 1/2 - \delta - \frac{c}{2k'\epsilon} - (1-\delta)\epsilon k'/c - \frac{4\delta k' f(\epsilon/c)}{(1-\mu)k} \\ &\leq \frac{8c}{(1-\mu)k\epsilon} + 1 < u \end{aligned}$$

where the last inequality follows from the definition of u. Obviously, this is a contradiction and the theorem follows. ◀

Next, we turn our attention to the off-chain proofness of the mechanism.

3.3 Making mechanism $\mathcal{M}_{\mathtt{bst}}$ off-chain proof

Next, we provide a modification of $\mathcal{M}_{\mathtt{bst}}$ that ensures Off-Chain Proofness. Taking a leaf from Ethereum's EIP-1559 pricing mechanism [24], we employ variable-sized blocks. Namely, we allow block size to exceed our target size (up to some amount) and use this information as a signal of increased or decreased demand, i.e, the relation of the size observed to the target size. The mechanism makes use of this information by proportionally increasing or decreasing a dynamic threshold that transaction-priority must exceed to be included in a block, in an effort to make demand equal to the target size. This change essentially limits the power mempool operators have in choosing the contents of blocks in a way that *cannot* be manipulated. Concretely, bribing a mempool operator to include your low-priority transaction will not help, since including the transaction into the block will make it invalid due to the threshold limitation.

In more detail, let α be the target percentage we want blocks to be filled. We set the threshold τ' of the next block after a chain $\mathcal{C}$ to be:

$$\tau' = \tau \cdot \exp\left(\beta \cdot \frac{\sum_{\mathrm{tx}\in S} \mathrm{size}(\mathrm{tx}) - \alpha \cdot L}{\alpha \cdot L}\right) \tag{3}$$

where L is the maximum size of a block, $\alpha \cdot L$ is equal to k, $\beta > 0$ is a scaling factor, and τ is the old threshold. This is similar to the Ethereum threshold update: there is some leeway to measure if blocks are too empty or too full. We are going to use a slightly different definition of priority than that of the previous section, namely:

$$\widehat{\mathrm{priority}}_{\mathcal{C}}(\mathrm{tx}) := \min\{\frac{\sum_{s_i \in \mathrm{tx}} \mathrm{val}(s_i) \cdot \mathrm{age}(s_i)}{\mathrm{size}(\mathrm{tx})}, \\ \frac{4}{(1-\mu)k} \cdot \left(1 + \frac{\mathrm{val}(\mathrm{tx})}{\mathrm{size}(\mathrm{tx})}\right) \cdot (1+\rho)^{\frac{\phi - \ln(\mathrm{size}(\mathrm{tx})/\mathrm{val}(\mathrm{tx}))}{\ln(2)}}\}$$

where ρ and ϕ are constants that will be defined later. Essentially, we have disproportionally increased the maximum priority value transactions can reach. By doing this we avoid attacks where the attacker by using maximum priority low-value transactions disproportionally increases the threshold value and "cheaply" excludes high priority transactions from entering the blockchain in the next block. We extend Definition 3 (Validity) to require that all transactions included in a block should have priority larger that τ', and denote the modified protocol by $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$.

Following the discussion about the incentive issues of Cardano in Section 2.2, note that $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$ actually *is* off-chain proof.

▶ **Corollary 8.** *$\mathcal{M}_{\mathtt{bst}}$ is off-chain proof iff the threshold is high enough so that the eligible pending transactions can fit into a single block. Formally, if the set E contains all transactions such that $\widehat{\text{priority}}_{\mathcal{C}}(\text{tx}) \geq \tau'$, $\mathcal{M}_{\mathtt{bst}}$ is off-chain proof iff $\sum_{\text{tx}\in E} \text{size}(\text{tx}) \leq L$.*

The main idea of the proof [12] is that an appropriate threshold implies that the block producer can add all pending transactions to her block. Under normal circumstances the eligible transactions would be about $a \cdot L$ in size. Any other transaction would be ineligible and cannot be added through an off-chain deal, no matter how valuable.

Although not formally studied, in the non-myopic case if the current block producer requested additional payment, there is enough slack so that the next block producer could include the previous transactions as well. However, during a sudden increase in demand the threshold might need a few blocks to adjust, leading to an excess of eligible, valuable transactions that could collude with block producers. This situation is similar to the "tipless" mechanism from [24], or to standard EIP-1559 but with off-chain proofness replaced by user incentive compatibility.

3.4 Worst-case delay of mechanism $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$

The modifications we employed in $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$ puts the worst-case delay guarantees proved earlier for $\mathcal{M}_{\mathtt{bst}}$ at risk. Next, we argue that Theorem 7 also holds for $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$ and its worst-case delay is asymptotically optimal, i.e, in the order of $O(\text{size}(\text{tx})/((1-\mu)\text{val}(\text{tx})k))$, albeit with a small overhead that has to do with the time it takes for the threshold to catch up to maliciously changing traffic conditions. As before, experimental results show tightly optimal delays under normal operation conditions. Notably, our result does not make any assumptions about the number of eligible transaction at each round, i.e., it is independent of traffic spikes.

Our analysis requires that the target transaction has token value at least $2^{-\phi}$; parameters can be appropriately tuned to make ϕ rather large for realistic applications. For simplicity, here we assume that $\alpha := 1/2$, $\phi = 20$, $\rho := 0.1$, $\beta := \ln(1+\rho)$.

▶ **Theorem 9.** *Setting $\alpha := 1/2$, $\phi = 20$, $\rho := 0.1$, $\beta := \ln(1+\rho)$, $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$ has worst-case delay $d(\text{tx})$ at most*

$$80\frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx})k} + \ln(\frac{11\cdot\text{size}(\text{tx})}{\text{val}(\text{tx})})/\beta + 10$$

when $\text{size}(tx) < k/2$ and $2^{-\phi} \leq \text{val}(\text{tx}) < 1/2$.

Proof. The main rationale of the proof of Theorem 7 is that as long as the target transaction tx is not included in a block, it will eventually attain maximum priority, say

$$T := P(1+\epsilon/c)(1+\rho)^{\frac{\phi-\ln(c/\epsilon)}{\ln(2)}}$$

where $\epsilon := \text{val}(\text{tx})$, $c := \text{size}(\text{tx})$ and $P := \frac{4}{(1-\mu)k}$, and from this point on the adversary will have to fill honest blocks with high priority transactions other than tx, which it cannot do for long due to the limited rate at which priority is generated. We are going to apply the same

[12] We omit the formal proof of this result as it is rather similar to the analysis in [24].

logic to bound the worst-case delay of $\mathcal{M}_{\mathtt{bst}}^{\mathtt{th}}$, with the only difference that the adversary now may skip filling some honest blocks due to the threshold value being higher than T at that point of the game. This implies that for a number of blocks, at least as high as the number of blocks honest parties would leave empty, the threshold must be higher than T. We argue next, that in order for this to happen the adversary has to fill an amount of space with high priority transactions proportional to that in the original mechanism $\mathcal{M}_{\mathtt{bst}}$, and thus does not gain much in terms of worsening the delay.

For the sake of contradiction, assume that $\mathcal{M}_{\mathtt{bst}}^{\mathtt{th}}$ does not satisfy the theorem statement, and thus there exists a tx that has greater delay that $d(\text{tx})$. Note, that the term $(1+\rho)^{\phi-\ln(c/\epsilon)}$ is upper-bounded by 7 for our parameters. Let $u := 8 \cdot \frac{c}{(1-\mu)\epsilon k} + 1$, as in Theorem 7. Similarly to the argument there, after $7u$ blocks, tx will have obtained maximum priority equal to T. Now, let $B_1, \ldots, B_u$ be the sequence of blocks starting after tx has attained its maximum priority, and denote by T_i the threshold of block B_i.

Assume for the moment, that $T_1 < T$, and let $i_1, i'_1, \ldots, i_m, i'_m$ be a subsequence of indices of $1, \ldots, m$ such that

$$T_{i_j}, T_{i'_j} \leq T \text{ and } T_l > T \text{ for } l \in (i_j, i'_j), j \in [m];$$

i_j, i'_j mark a sequence of threshold values that are greater than T.

First, we argue that any B_{i_j} for $j \in [m]$ should contain transactions with token-value density at least $\epsilon/(2c)$, i.e, that

$$T_{i_j} > \hat{T} := P(1+\epsilon/(2c))(1+\rho)^{\frac{\phi-\ln(2c/\epsilon)}{\ln(2)}}$$

For two subsequent thresholds T', T'', where $T'' > T$, it holds that:

$$\begin{aligned} T'' > T &\Rightarrow T' e^{\beta(L-L/2)/(L/2)} > T \\ &\Rightarrow T' e^{\ln(1+\rho)} > P(1+\epsilon/c)(1+\rho)^{\frac{\phi-\ln(c/\epsilon)}{\ln(2)}} \\ &\Rightarrow T' > P(1+\epsilon/c)(1+\rho)^{\frac{\phi-\ln(c/\epsilon)}{\ln(2)}-1} \\ &\Rightarrow T' > P(1+\epsilon/(2c))(1+\rho)^{\frac{\phi-\ln(2c/\epsilon)}{\ln(2)}} = \hat{T} \end{aligned}$$

where w.l.o.g., we have assumed that B_{i_j} is full. It follows that B_{i_j} contains only transactions with priority greater than $\hat{T}$, which can only be attained if the transactions have toke-value density at least $\epsilon/(2c)$. Moreover, in case B_{i_j} is an honest block it should contain transactions with priority at least T and be at least aL full.

Furthermore, we argue that for the threshold to be larger than T in a sequence of blocks, as in $T_{i_j}, \ldots, T_{i'_j-1}$, it must be the case that blocks on average contain an amount of data proportional to their number. First, for subsequent threshold values it should hold that:

$$T_{i+1} = T_i e^{\beta(x_i - aL)/(aL)} \Leftrightarrow x_i = aL(1 + \ln(T_{i+1}/T_i)/\beta)$$

where x_i is the fullness level of block i. Now, for any j and $w := i_j, v := i'_j - 2$, we get that:

$$\begin{aligned} \sum_{i=w}^{v} x_i &= aL(\sum_{i=w}^{v}(\ln(T_{i+1}) - \ln(T_i))/\beta + 1)) \\ &= aL((\ln(T_{v+1}) - \ln(T_w))/\beta + v - w + 1) \\ &= aL(\ln(T_{v+1}/T_w)/\beta + v - w + 1) \\ &\geq aL(v - w + 1) = aL(i'_j - i_j - 1) \end{aligned}$$

and since the threshold in all these blocks is above $\hat{T}$, as we argued before, it must be the case that these blocks contain $aL(v-w+1)$ amount of transactions with that much priority each. Summing over all j, we get that the respective blocks should contain an amount of $aL\rho$ transactions with priority at least $\hat{T}$ each, where $\rho = |\{T_i | T_i > T, i \in [u]\}|$.

Finally, honest blocks with threshold lower than T must be covered with transactions of density at least ϵ/c and priority at least T, as otherwise tx is going to be included.

Putting it all together, we have the following:

- blocks in $S = \{B_i | B_i \text{ is honest}, T_i < T, i \neq i_j, i \in [u], j \in [m]\}$ should contain transactions with total size at least aL and priority at least T each;
- blocks in $H = \{B_i | T_j > T, i \in [u], i \neq i'_j - 1, j \in [m]\}$ should contain transactions with priority at least T;
- blocks in $W = \{B_{i_j} | j \in [m]\}$ should contain transactions with total size at least aL and priority at least $\hat{T}$ each;
- blocks in $W_H = \{B_{i_j} | B_{i_j} \text{ is honest }, j \in [m]\}$ should contain transactions with total size at least aL and priority at least T each.

Thus, the adversary has to generate transactions whose total normalized priority times size is at least:

$$\begin{aligned} & aL\hat{T} \cdot (2|S| + (|H| + |W \setminus W_H| + 2|W_H|)) \\ & \geq aL\hat{T} \cdot (|S| + |H| + m + |W_H|) \\ & \geq aL\hat{T} \cdot (|\{B_i| \text{ is honest block }, i \in [u]\}|) \\ & \geq aL\hat{T} \cdot (1-\mu)u/(2c) \end{aligned}$$

where we have used the fact that $|W| = m$ and blocks in W_H contain transactions with priority T. Hence, the adversary has to fill as many blocks as in the proof of Theorem 7 when the token-value density of the target transaction is $\epsilon/2c$. By our previous analysis this is not possible for $u_2 := 16c/((1-\mu)\epsilon k) + 1$.

Finally, it remains to argue about our assumption that T_1 is less than T. Assume that we are at a round where tx has maximum priority T and the threshold remains above or equal to T for $u' + 1$ rounds. By our earlier argument, to maintain the threshold above T the adversary must fill produced blocks with high priority ($> T$) transactions of total size:

$$\begin{aligned} aL(\ln(T_{u'+1}/T_1)/\beta + u') &\geq aL(\ln((P\epsilon/c)/(7P(1+1/2)))/\beta + u') \\ &\geq aL(u' - \ln(11c/\epsilon)/\beta) \end{aligned}$$

where $7P(1+1/2)$ is an upper bound on the threshold value. We want to choose a u' such that it is impossible for $\mathcal{A}$ to generate that many high priority transactions within u'.

By Theorem 7, we know that it is impossible to fill $\tilde{u}(1-\tilde{\mu})$ honest blocks with transactions of priority T in less than $\tilde{u} := 8c/((1-\tilde{\mu})\epsilon k)$ rounds. Take now $\tilde{\mu} > 1/(1 + \frac{8c}{\epsilon k \gamma})$, where $\gamma := \ln(11c/\epsilon)/\beta$. It holds that $\tilde{u} > \gamma + 8c/(\epsilon k)$. Setting $u' := \tilde{u}$, we see that

$$u' - \ln(11c/\epsilon)/\beta > u' - \gamma = 8c/(\epsilon k) \geq u'(1-\tilde{\mu})$$

which implies by our previous observation that $\mathcal{A}$ will not be able to fill the required blocks to retain the threshold larger than T for this selection of u'.

Concluding, tx must be included in a block after a total of

$$72c/((1-\mu)\epsilon k) + \ln(11c/\epsilon)/\beta + 8c/(\epsilon k) + 10$$

rounds. The theorem follows. ◀

3.5 Putting everything together

We have shown that given a transaction and its token-value, it is possible to bound (and thus predict) its worst-case delay. Moreover, that this delay is asymptotically optimal and that mechanism $\mathcal{M}^{\texttt{th}}_{\texttt{bst}}$ is off-chain proof. Note now, that if there was a way of distributing the blockchain space tokens ahead of time, mechanism $\mathcal{M}^{\texttt{th}}_{\texttt{bst}}$ would satisfy our initial goals. We formalize this idea in the next theorem.

▶ **Theorem 10.** *Given tokens are obtained and available in advance before use by a window of time t to a user, and setting the mechanism parameters as in Theorem 9, $\mathcal{M}^{\texttt{th}}_{\texttt{bst}}$ offers predictable service to that user with parameter t, it has asymptotically optimal worst-case delay, and is Off-Chain Proof.*

Proof. Theorem 9 and Corollary 8 imply asymptotically optimal worst-case delay and Off-Chain Proofness. To argue about predictable service, not that since tokens are obtained t time in advance, a user knows ahead of time both the cost and the worst-case delay of its transaction; the worst-case delay can be calculated based on the amount of token value obtained by the user using Theorem 9. The theorem follows. ◀

Note that as in [24], off-chain proofness is shown unless in the midst of a demand spike. To complete our proposal, in the next section we describe a way of distributing tokens ahead-of-time.

4 Allocation of Tokenized Space

In this section, we discuss a specific way to apply the ideas and results from the previous section. Let's first consider two potential options for token instantiation and distribution. The two extremes are to use either the existing stake (in Proof-of-Stake systems) or to create a *new space token specifically designed for this purpose*.

Using dedicated space tokens offers significant flexibility because they are independent of any restrictions related to other uses of stake, such as consensus or smart contracts. However, there is a risk of Denial-of-Service (DoS) attacks if a malicious party aquires almost all space tokens. We propose two methods to mitigate this risk, both of which offer additional advantages.

The first idea is to *partition the blockchain space into two fixed parts:* the "spot space" and the "tokenized space." In the *spot space*, transactions are included using the usual mechanism (e.g., first-price auctions in Bitcoin, EIP-1559 in Ethereum). In the *tokenized space*, transactions are included using the proposed BST mechanism of the previous section with the space tokens. The fraction allocated to the tokenized space is fixed permanently or adjusted very slowly by blockchain governance to meet demand. We anticipate power users who issue many transactions will utilize the tokenized space, while regular users will primarily use the spot space. DeFi users may leverage both spaces depending on their needs for speed, cost, and predictability.

The second idea is to *limit the lifetime of each access token.* A lifespan of a few months makes it challenging and expensive for a malicious actor to control all tokens for an extended period. Of course, preventing a well-funded attacker from acquiring all tokens is impossible, but this risk exists in any system that allocates blockchain space through a transaction fee system.

A simple and effective way to distribute the space tokens is to sell them in an *auction conducted within the spot space.* This approach avoids bootstrapping difficulties. For example, the system can run a sealed-bid auction every T blocks for tokens expiring after L blocks,

where L can be a small multiple of T (e.g., a monthly auction for tokens with a 3-month lifespan). While sealed-bid auctions typically require significant space for registering bids and recording outcomes on the blockchain, we anticipate that only a few hundred power users will participate, making on-chain execution feasible. Otherwise, the auction can run off-chain, with only the results recorded on the blockchain. To prevent incentive issues, *bidders should submit encrypted bids along with collateral*. During the auction, bids are decrypted and the winners are determined. The collateral ensures bidders don't withdraw after the auction and should be set high enough to deter this behavior.

With sealed bids, there are several options for a *truthful multi-unit auction*, including the VCG auction with no reserve price, the VCG with a reserve price [18], or even more exotic auctions (such as Triage auctions [7]). Taking into account limitations in communication and blockchain space, the VCG auction with reserve price emerges as the most suitable choice. This reserve price can be calculated based on historical values of the tokenized space, transaction fees of the spot space, or set as a fixed value with adjustments by blockchain governance. An additional advantage of the VCG auction with reserve price is that it is also the optimal auction for maximizing revenue in Myerson's settings [17]. The revenue of the auction could be equally distributed to the block producers at the end of the period to eliminate any strategic considerations by block producers. While technically this is a repeated auction, the analysis of the myopic (single-shot) setting captures the key aspects given the relatively long intervals between auctions.

Once the auction concludes, the tokens become tradable like any other token until they expire. Power users can estimate their service needs at the auction and then buy or sell tokens to adjust their requirements throughout the period.

5 Deployment Considerations

Next, we focus on deployment considerations of mechanism $\mathcal{M}^{\mathtt{th}}_{\mathtt{bst}}$. We argue that the main component of the system, i.e., the procedure that computes the priorities of different transactions, can be efficiently and compactly implemented in the major blockchain architectures.

Firstly, our scheme does no require support of *any specialized cryptographic primitives*, such as VRFs, VDFs, ZK-SNARKs, etc. [13] Typically, implementing new cryptographic primitives is one of the major obstacles in quickly releasing new technology in the blockchain landscape. The main operation of the mechanism revolves around being able to efficiently determine the priority of different transactions and pick the ones that have the maximum priority, which basically amounts to suitable book-keeping.

In more detail, the mempool operator should maintain a data-structure DS containing the block that each token was last referenced by some transaction. To determine the priority of a transaction tx, it suffices to query DS about the age of the tokens referenced by tx, and then simply compute the priority value following the equation in Section 3.3. Using some kind of self-balancing binary search tree to implement DS, e.g., an AVL tree, allows retrieving and updating the token-age related information in $O(\log(n))$ time in the worst-case, where n is the total number of tokens. Therefore, computing the priority of a transaction takes $O(m \cdot \log(n))$ time, where m is the number of tokens referenced by the transaction, while updating the token related information takes $O(l \cdot \log(n))$, where l is the number of token values to be updated. Space-wise, the AVL tree takes about $O(n)$ space.

[13] We note that this is not necessarily the case for the token-distribution part, where an auction has to be deployed.

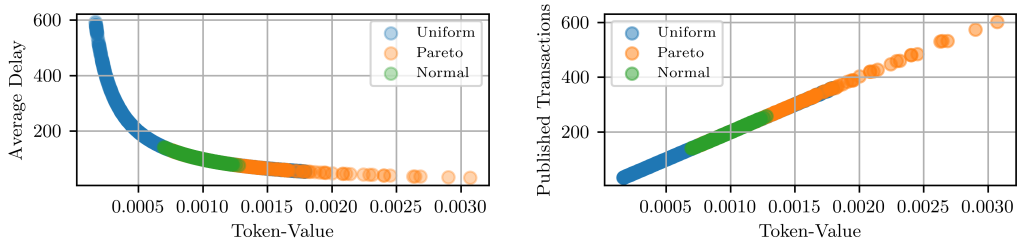

Figure 3 1000 users for 20000 steps, where every user becomes active with $p_{i,t} = 0.8$.

To more efficiently use their tokens, users may be tempted to split them in smaller chunks. This way they can better control the priority injected into a transaction by including more or less of these small tokens. Such token-splitting would result in more load to the system, and is generally unwanted. To avoid such an issue, a transaction can explicitly state how much of the priority generated by each token referenced should be used, with the rest retained for later use. Such a change does not affect the security analysis presented in previous sections, and can be easily implemented: DS, in addition to the last-use information stored per token, also stores any remaining priority left from a previous use of the token. Again, transaction priority can be efficiently computed.

Finally, we describe two possible instantiations of the mechanism in the UTxO (used by Bitcoin) and the account model (used by Ethereum), respectively.

In the UTxO-based case, say in Bitcoin, we can introduce our space tokens using the ordinals mechanism [1] each satoshi (Bitcoin's smallest denomination) can receive an inscription and afterwards it can be transferred as an NFT. The initial inscription can specify the value of the token in terms of priority, and subsequently, it is possible to consume only a fraction of the priority of a token by prescribing a value in $[0, 1]$ and using the reinscription mechanism of ordinals – this deals with the token-splitting issue described earlier. To issue a transaction utilizing such a token, it is sufficient to post a transaction transferring the corresponding NFT to the change address of the posting user, while setting the transaction fee to 0. It is easy to see that this mechanism can be facilitated as a soft fork in the Bitcoin network (note that transaction relaying with 0-fees would need to be amended accordingly). On the other hand, in the account-based case, say in Ethereum, a smart contract can mint the tokens with their corresponding values and subsequently the priority consumed can be specified in each transaction. Note finally that a hard-fork would be required to allow transactions posted with zero fees that utilize a space token instead to become admissible into the ledger (as due to EIP-1559 it is imperative that a valid transaction comes with a minimum fee).

6 Simulations

We validate our theoretical results using experiments on synthetic blockchain traffic. Specifically, we are assuming that there are n independent transaction issuers, each of which has tokens of value $v_i > 0$, sampled independently for the same distribution F (and subsequently normalized so the total amount is 1). At every step, each user might be *active* or *inactive*. If they are inactive at time $t - 1$, they flip a coin and become active at time t with probability $p_{i,t}$. Once they are active, they submit a transaction and remain active until that transaction is published. Then, they become inactive again and the cycle continues. For simplicity, the users do not trade their tokens and there are no adversarially produced blocks. We use the BST mechanism with the parameters of Theorem 9. Each transaction has size 1 and a block can hold up to 20 transactions, with 10 being the target size for the threshold update rule.

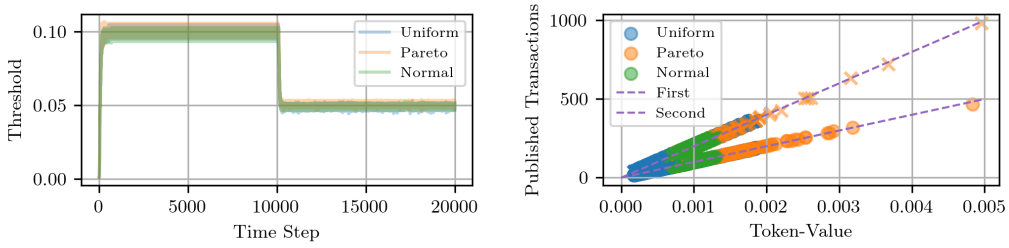

Figure 4 1000 users for 20000 steps, where a random subset of half the users stops issuing new transactions halfway. Notice that the threshold drops to half its value and the users that remain active publish twice as many transactions given their token value. The user transactions published in the second half are denoted using the cross symbol, while the circles refer to the first half.

We simulate two scenarios, both of which consist of four runs with different token distributions. Specifically, we have:

- Uniform in $[0, 1]$.
- Pareto Type II with parameter 2.
- Truncated Normal with $\mu = 100$ and $\sigma^2 = 10$.

In the first scenario, depicted in Figure 3, we have 1000 users for 20000 blocks. The activation probability of all users stays the same throughout. The average delays follow the worst case result from Theorem 9. The relation is much better depicted in the graph at the right, where the number of published transactions (which is the inverse of the delay multiplied by the number of blocks) is shown to be linear in the amount of tokens.

In the second scenario, depicted in Figure 4, we show that this mechanism has the ability to *adapt* to changes in demand. We vary the activation probability as follows: all users have $p_{i,t} = 1$ for the first 10000 steps and then half of the users switch to 0. Notice how the threshold decreases, and also that in both cases we match the optimal worst-case delay (which is easier seen as the number of published transactions), thus showcasing that under normal operation conditions our mechanism indeed achieves optimal delays.

References

1 Ordinal Theory Authors. Ordinal theory handbook. `https://docs.ordinals.com/`, 2024.

2 Soumya Basu, David Easley, Maureen O'Hara, and Emin Gün Sirer. Towards a functional fee market for cryptocurrencies. *arXiv preprint*, 2019. `arXiv:1901.06830`.

3 Vitali Buterin. On inflation, transaction fees and cryptocurrency monetary policy, 2016. URL: `https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy`.

4 Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, and Ian Norden. Ethereum improvement proposal 1559: Fee market change for eth 1.0 chain, 2019. URL: `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md`.

5 Xi Chen, David Simchi-Levi, Zishuo Zhao, and Yuan Zhou. Bayesian mechanism design for blockchain transaction fee allocation. *Available at SSRN 4413816*, 2023.

6 Hao Chung and Elaine Shi. Foundations of transaction fee mechanism design. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3856–3899. SIAM, 2023.

7 Shahar Dobzinski and Noam Nisan. Multi-unit auctions: Beyond roberts. *J. Econ. Theory*, 156:14–44, 2015.

8 Matheus VX Ferreira, Daniel J Moroz, David C Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 86–99, 2021.

9 Yotam Gafni and Aviv Yaish. Greedy transaction fee mechanisms for (non-) myopic miners. *arXiv preprint*, 2022. arXiv:2210.07793.

10 Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 2015.

11 Andrew V Goldberg, Jason D Hartline, Anna R Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.

12 Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for {Fine-Grained} resource sharing in the data center. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

13 Gur Huberman, Jacob D Leshno, and Ciamac Moallemi. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *The Review of Economic Studies*, 88(6):3011–3040, 2021.

14 Ron Lavi, Or Sattath, and Aviv Zohar. Redesigning bitcoin's fee market. *ACM Transactions on Economics and Computation*, 10(1):1–31, 2022.

15 Stefanos Leonardos, Barnabé Monnot, Daniël Reijsbergen, Efstratios Skoulakis, and Georgios Piliouras. Dynamical analysis of the eip-1559 ethereum fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 114–126, 2021.

16 Stefanos Leonardos, Daniël Reijsbergen, Daniël Reijsbergen, Barnabé Monnot, and Georgios Piliouras. Optimality despite chaos in fee markets. *arXiv preprint*, 2022. arXiv:2212.07175.

17 Roger B. Myerson. Optimal auction design. *Math. Oper. Res.*, 6(1):58–73, 1981.

18 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

19 Oiler. The oiler network. https://www.oiler.network/. Accessed: 2024-10-16.

20 Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. Cryptology ePrint Archive, Report 2016/454, 2016. URL: http://eprint.iacr.org/2016/454.

21 Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017 – 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017. doi:10.1007/978-3-319-56614-6_22.

22 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, January 2016.

23 Daniël Reijsbergen, Shyam Sridhar, Barnabé Monnot, Stefanos Leonardos, Stratis Skoulakis, and Georgios Piliouras. Transaction fees on a honeymoon: Ethereum's eip-1559 one month later. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 196–204. IEEE, 2021.

24 Tim Roughgarden. Transaction fee mechanism design. In Péter Biró, Shuchi Chawla, and Federico Echenique, editors, *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, page 792. ACM, 2021. doi:10.1145/3465456.3467591.

25 Itay Tsabary, Alex Manuskin, Roi Bar-Zur, and Ittay Eyal. Ledgerhedger: Gas reservation for smart-contract security. *Cryptology ePrint Archive*, 2022.

26 Carl A Waldspurger and William E Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, pages 1–es, 1994.

27 Andrew Chi-Chih Yao. An incentive analysis of some bitcoin fee designs. *arXiv preprint*, 2018. arXiv:1811.02351.