A Fair and Resilient Decentralized Clock Network for Transaction Ordering

Andrei Constantinescu
ETH Zürich, Switzerland

Diana Ghinea
ETH Zürich, Switzerland

Lioba Heimbach
ETH Zürich, Switzerland

Zilin Wang
ETH Zürich, Switzerland

Roger Wattenhofer
ETH Zürich, Switzerland

Abstract

Traditional blockchain design gives miners or validators full control over transaction ordering, i.e., they can freely choose which transactions to include or exclude, as well as in which order. While not an issue initially, the emergence of decentralized finance has introduced new transaction order dependencies allowing parties in control of the ordering to make a profit by front-running others' transactions. In this work, we present the *Decentralized Clock Network*, a new approach for achieving fair transaction ordering. Users submit their transactions to the network's clocks, which run an agreement protocol that provides each transaction with a timestamp of receipt which is then used to define the transactions' order. By separating agreement from ordering, our protocol is efficient and has a simpler design compared to other available solutions. Moreover, our protocol brings to the blockchain world the paradigm of asynchronous fallback, where the algorithm operates with stronger fairness guarantees during periods of synchronous use, switching to an asynchronous mode only during times of increased network delay.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols

Keywords and phrases Median Validity, Blockchain, Fair Ordering, Front-running Prevention, Miner Extractable Value

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2023.5

Related Version *Full Version*: https://arxiv.org/abs/2305.05206

1 Introduction

The first blockchain, a decentralized distributed digital ledger that records transactions across a network of computers, was introduced in 2008 with Bitcoin by Nakamoto [33]. Blockchains offer a novel way of storing and transferring value in a trustless and secure manner, without the need for intermediaries. Despite their popularity, blockchain adoption was slow, as blockchains were, initially, mainly used to facilitate simple transfers of money between two individuals. However, this changed in 2015 with the introduction of smart contracts on Ethereum [44], allowing for complex digital agreements to be carried out on-chain. Nowadays, smart contracts are the backbone of a rapidly-growing complex ecosystem of decentralized financial applications known as *decentralized finance (DeFi)*. DeFi offers most traditional financial services, including decentralized exchanges, lending protocols, and stablecoins, without relying on financial intermediaries.

27th International Conference on Principles of Distributed Systems (OPODIS 2023).
Editors: Alysson Bessani, Xavier Défago, Yukiko Yamauchi, and Junya Nakamura; Article No. 5; pp. 5:1–5:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The smart contracts that govern DeFi are generally dependent on the transaction order. That is, the outcome of executing a set of transactions depends on their order. As most transactions were simple transfers in the early days, the original blockchain design did not need to pay much attention to transaction ordering. Instead, the power of transaction ordering is concentrated in miners or validators, which can freely choose which transactions to include and how to order them inside each block. Nowadays, block proposers (miners) extract profit from appropriately ordering, including, and excluding transactions during block production. This profit is known as miner (or maximal) extractable value (MEV). MEV accounts for a profit of at least US$ 650M [21] so far. In fact, Flashbots and other transaction relay protocols organized a whole market around ordering transactions.

Front-running Attacks

Most MEV relies on the ability of the attacker to *front-run* the victim's transaction tx. To be specific, the attacker observes a newly generated victim transaction tx in the mempool (the public waiting area for transactions). The attacker then introduces their own transaction tx'. If tx' executes before tx (front-running), the attacker profits at the expense of the victim. So, the attacker may simply bribe the block proposer with a high fee to execute tx' first, even though tx' was only created once tx was already publicly known.

Front-running can be broadly categorized into two types [35]: *tolerant* and *destructive*. Tolerant front-running involves the attacker placing their own transaction before the victim's transaction in the order of execution. This allows the attacker to gain an advantage, such as purchasing an asset at a lower price before the victim can. Such attacks are often seen on decentralized exchanges, where the attacker executes a trade before the victim, reaping the benefits of price changes. Destructive front-running, on the other hand, has the attacker taking out the victim's transaction altogether. Generally, the attacker copies the victim's presumably profitable transaction. If the attacker's transaction executes first, the victim's transaction would no longer execute, at least not as intended.

Our Contribution

We propose the Decentralized Clock Network (DCN), a novel solution for achieving fair transaction ordering. More concretely, our system ensures that, if a transaction tx was sent to the system long enough before transaction tx', then tx' cannot be ordered before tx, i.e., preventing tolerant front-running. In contrast to most previous solutions relying on the blockchain consensus algorithm to determine a relative ordering of the transactions, our approach employs a decentralized network of n nodes, equipped with clocks, resilient to $f < n/3$ byzantine failures to agree on a timestamp for each transaction. These timestamps are subsequently used to determine the order of the transactions inside each block and across blocks. Decoupling timestamping from ordering enables lower latency bounds whilst reducing the complexity of the consensus mechanism.

A blockchain system is synchronous if all messages arrive at the receiver within a known time-bound, and the nodes involved have local clocks that are (almost) perfectly synchronized. However, in times of turmoil, such as when participants are under attack, messages might experience longer delays, or clocks may no longer be aligned with real-time. Such failures are modeled by the asynchronous model. An important novelty in our work is that our protocol is designed to provide guarantees regardless of the network conditions, without knowing in advance which setup to expect. It is designed for the asynchronous model, however, if the network happens to satisfy some synchrony assumptions, which is often the case in real-world networks, it provides stronger guarantees reflecting in the order obtained. To quantify this effect, we propose a new notion of order fairness, called δ-Median Fairness. Roughly, transactions shall be ordered based on a value that is close to the median of the points in time when honest nodes in the DCN first learn about the transaction. Here, δ is an error parameter, determining the closeness of the estimated median to the true median of the honest timestamps in terms of quantiles. This definition is a stronger version of Honest-Range Fairness (or fair separability, as defined in [46]). When operating under asynchronous conditions, our algorithm achieves f-Median Fairness, which coincides with Honest-Range Fairness in the worst case $n = 3f + 1$, but is stronger otherwise. On the other hand, when the network is synchronous for a sufficient amount of time, our algorithm achieves the superior guarantee of $\lceil f/2 \rceil$-Median Fairness. In both cases, these guarantees are optimal. We add that our protocol sidesteps the attack where relative orders relying on the median can be manipulated by a single byzantine node presented in [27] by ensuring that (1) nodes always agree on some honest timestamp, and (2) with the help of cryptographic primitives, we do not allow nodes, or anyone else, to see the transaction contents before a timestamp is agreed upon.

Related Work

Fair Ordering. Blockchain front-running prevention techniques have been the subject of significant research in recent years. We point the reader to Baum et al. [5] and Heimbach et al. [25] for an overview of these approaches and only discuss the most relevant in the following.

Flashbots [20] and other private relay services, in which transactions are sent directly to a trusted third party for ordering and subsequent forwarding to validators for block inclusion, are widely adopted. While this approach is efficient, it centralizes the transaction ordering process, i.e., introduces a single point of failure, and is often used to front-run as opposed to protect against. In contrast, our approach distributes the transaction ordering responsibility.

In the field of fair transaction ordering, committee-based approaches have been widely studied. Generally, these approaches can be divided into two categories: those that can operate in asynchrony and those that assume partial synchrony, which is a model weaker than synchrony and stronger than asynchrony. To tackle fair ordering in partial synchrony, Pompe is proposed by Zhang et al. [46], Wendy is proposed by Kursawe [28] and Themis is proposed by Kelkar et al. [26]. As opposed to these protocols, the DCN we propose is equipped to handle asynchrony. In particular, Pompe and Themis rely on (partial) synchrony and Wendy assumes the clocks of the nodes are always synchronized.

Kelkar et al. [27] introduce Aequitas, which achieves state-of-the-art fairness properties, but has a significant communication complexity of $\mathcal{O}(n^4)$ in asynchrony. Our agreement protocol achieves in expectation $\mathcal{O}(n^3 \log \Delta)$ message complexity in asynchrony, where Δ denotes the *observed* network delay. We note that this delay does not have to be known a priori, as opposed to classical synchronous protocols.

Quick order fairness, introduced by Cachin et al. [14] achieves $\mathcal{O}(n^3)$ message complexity in asynchrony. While their protocol allows for a node to gain insider information before an ordering is agreed upon, our protocol adds further protection to users as the committee only sees the full transaction after the timestamp is agreed upon. Further, their approach, and the others, only target agreement amongst the permissioned committee, while our design extends to implementing the fair ordering on a permissionless blockchain after agreement has been reached in the permissioned committee.

Agreement Protocols. Achieving agreement on a value subject to some Validity condition, i.e., Byzantine Agreement (BA) [29], is an extensively studied problem in Distributed Computing. In real-world applications, hence also in our setting, it is desirable to expect the Validity condition to carry some meaning, while the classical BA definition only ensures that if honest nodes have the same input value v, they all output v. If this pre-agreement condition is not met, the honest nodes may output an adversarially chosen value. Recent works have focused on achieving more meaningful guarantees, such as ensuring that the honest output is *close* to the honest inputs' median [39], to the k-th lowest honest input [31], or somewhere in the range of honest inputs [42]. These works, however, only focus on the synchronous model. That is, they assume perfectly synchronized clocks and a publicly available upper bound on the network delay. A more realistic setting is the so-called asynchronous model, which drops this assumption, but showcases important limitations: in the asynchronous setting, BA cannot be achieved deterministically [19]. There is still hope, however: randomized asynchronous BA protocols exist [6, 11, 13, 15, 22, 32, 36, 41]; however, without meaningful Validity guarantees if the input space contains more than two values. Another relaxed variant of BA is Approximate Agreement (AA) [3, 18], which offers deterministic protocols that enable honest nodes to output values within the range of their inputs, with the caveat of weakening the Agreement guarantees: honest outputs are ε-close for any predefined $\varepsilon > 0$.

To implement our fair-ordering definition, we propose an asynchronous (randomized) BA protocol with optimal resilience, that achieves Median Validity [31, 40] with optimal-error guarantees, assuming that the inputs are integers. Our lower bound on this error implies that, when the network is asynchronous, and when aiming for optimal resilience, the best one can hope for is obtaining outputs within the range of the honest inputs. We circumvent this problem by designing a protocol whose Validity guarantees scale with the network conditions: if the synchrony assumptions are satisfied for a sufficient amount of time, our protocol will enable honest nodes to agree on a value satisfying the synchronous model's optimal guarantees on Median Validity. Otherwise, our protocol will at least provide Median Validity with optimal guarantees for the asynchronous model, hence the output agreed upon will be within the range of honest values. Designing protocols that achieve simultaneously optimal guarantees in both synchronous and asynchronous networks, has been a topic that attracted increased attention in the recent years in the Distributed Computing literature. There has been a line of works focusing on problems such as Byzantine Agreement [7], Approximate Agreement [23], State Machine Replication [8], and also Multi-Party Computation [4, 9, 17].

2 The Decentralized Clock Network

In this section, we describe the DCN, which consists of a network of nodes equipped with synchronized clocks operating with the objective of providing an accurate and decentralized timestamping service to blockchain transactions. The resulting timestamps are used to determine the ordering of the transactions inside each block, as well as across blocks. The intuition behind using a timestamping service is that, instead of relying on consensus to determine the ordering directly, like in FSS from ChainLink Labs [16], this way the order of the transactions is naturally induced by the timestamps, allowing the complexity of the agreement protocol to be reduced.

High-Level Design

To enable DCN support for ordering transactions on an existing blockchain, the blockchain requires only minor adaptations. In particular, with every submitted transaction, an additional timestamp computed by the DCN is expected. Validators should check for each block whether timestamps are authentic and whether the ordering induced by the timestamps is respected, rejecting the block otherwise. In order for this check to take place, timestamps computed by the DCN are accompanied by threshold signatures, cryptographic gadgets used to prove that each timestamp was agreed upon by at least one honest node. Nodes in the DCN must not only be trustworthy, but also have good network conditions and be able to handle a large volume of service requests. To ensure the precision and consistency of the nodes' clocks, as well as nodes' high availability, we implement the DCN as a permissioned system, where the identity and public keys of the nodes are known to the validators. Nodes are not intended to change frequently and, by keeping the set of nodes in the system fixed, we can ensure that the nodes are reliable and that the timestamping service is accurate.

Network Model and Assumptions

The DCN consists of n nodes in a fully-connected network, such that any two nodes in the network can communicate through authenticated channels. Nodes can moreover receive external inputs, e.g., transactions from users. Each node comes equipped with a clock. We assume that node clocks are periodically realigned with real-time, which can be achieved through the use of a common external reference, such as UTC time or GPS time.

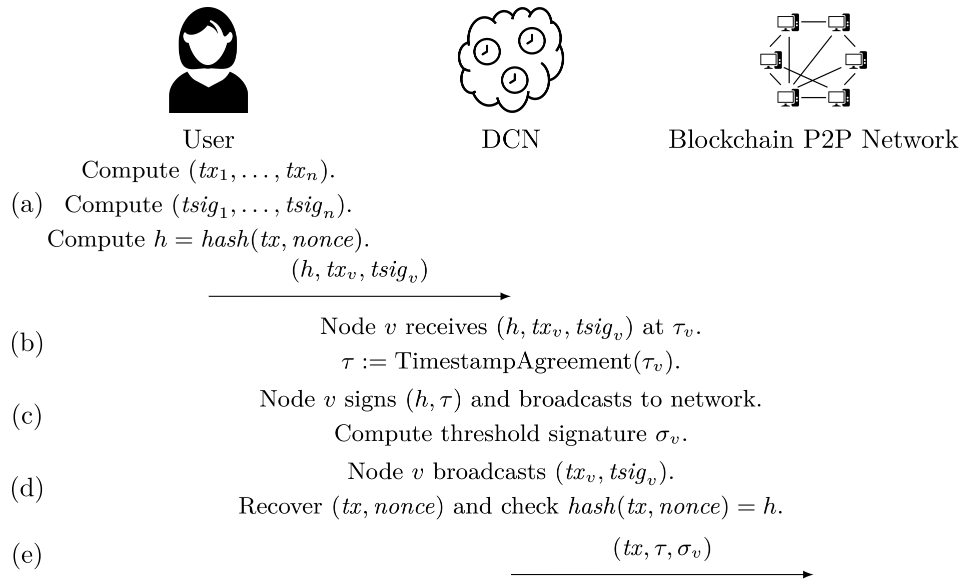
We consider an adaptive adversary that takes control during the protocol's execution of at most $f < n/3$ nodes, causing them to deviate arbitrarily (even maliciously) from the protocol; i.e., byzantine behavior.

We assume an estimation Δ_{DCN} representing an upper bound on the network delay within the DCN, i.e., messages sent between the nodes *should* be delivered within Δ_{DCN} time. Similarly, we assume an estimation Δ_{EXT} for the upper bound on the external network delay, i.e., for messages sent between users and the nodes in the DCN. We say that the network is synchronous if the message delays are *always* at most Δ_{DCN} and Δ_{EXT}, and the nodes' clocks are perfectly synchronized. If any of these conditions fail at any point, then the network is asynchronous. In our work, we will assume that the network is asynchronous. However, we take into account that an asynchronous assumption is often too pessimistic to model a real-life network. Hence, we aim to offer stronger guarantees during timespans when the network is synchronous, which should be the case most of the time if our estimations Δ_{DCN} and Δ_{EXT} are faithful. We also take into account that real clocks may fail the perfect synchrony assumption; i.e., they may have a small skew S, or their local rate may vary by a factor $\theta = 1 + o(1)$, as described in [30]. However, we assume perfect synchronization for simplicity of presentation, and we will briefly describe how our protocols can be modified to achieve the same synchronous guarantees under the weaker clock synchronization assumptions.

Cryptographic Primitives

As mentioned previously, we employ threshold signatures. In an (ℓ, n)-threshold signature scheme, a public key is known to the n nodes and also to all users and validators. Moreover, each node v knows a unique private key that enables the generation of a partial signature $\sigma_v(m)$ for any message m. The defining property of the scheme is that ℓ partial signatures from distinct nodes for the same message m can be combined into a single signature $\sigma(m)$ that can be verified using the public key. Formally, the scheme should satisfy robustness and non-forgeability (see the full version of [12, Section 2.3.2] for the definitions). For our purposes, we set the threshold $\ell = f + 1$ and choose the BLS scheme [10,38].

Furthermore, we require a secret sharing scheme. In a (k, n)-secret sharing scheme, a secret, such as a user transaction, is divided into n so-called *shares*, one known to each node,



Compute $(tx_1, \ldots, tx_n)$.
Compute $(tsig_1, \ldots, tsig_n)$.
Compute $h = hash(tx, nonce)$.
$(h, tx_v, tsig_v)$

Node v receives $(h, tx_v, tsig_v)$ at τ_v.
$\tau :=$ TimestampAgreement(τ_v).

Node v signs (h, τ) and broadcasts to network.
Compute threshold signature σ_v.

Node v broadcasts $(tx_v, tsig_v)$.
Recover $(tx, nonce)$ and check $hash(tx, nonce) = h$.

(tx, τ, σ_v)

Figure 1 Illustration of the transaction submission (i.e., main) protocol.

such that any k nodes can reconstruct the secret, while any coalition of at most $k - 1$ nodes cannot learn anything about it. Formally, the information-theoretic requirement is that any k shares uniquely determine the secret, while any $k - 1$ shares must be independent of the secret. Informally, given $k - 1$ shares, every possible transaction is equally likely to result in these shares. In our work, we choose $k = f + 1$ and use the Shamir scheme [37].

The Transaction Submission Protocol

In this section, we formally present the protocol used when users submit transactions to the blockchain (cf. Figure 1), which we also refer to as the *main* protocol. Assume a user wants to submit transaction tx, then the following steps are to be followed:

(a) The user generates a random nonce *nonce*. Then, the user splits the pair $(tx, nonce)$ into n shares $(tx_1, \ldots, tx_n)$ using the $(f+1, n)$-secret sharing scheme and signs the shares with their private key to get $(tsig_1, \ldots, tsig_n)$. Subsequently, the user hashes the transaction together with the nonce as $h = hash(tx, nonce)$. Finally, they send to each node v the tuple $(h, tx_v, tsig_v)$.

(b) Each node v receives $(h, tx_v, tsig_v)$ at some time τ_v. Together, the nodes run a *Timestamp Agreement* protocol to agree on a common timestamp τ for transaction tx. The agreement protocol is described in detail in Section 4.

(c) Upon reaching agreement, each node signs (h, τ) and broadcasts the signature to the other nodes. Each node v receives the signatures of (h, τ), verifies them, and uses the at least $f + 1$ valid ones to compute a threshold signature σ_v for the pair (h, τ).

(d) Afterwards, each node v broadcasts their signed share $(tx_v, tsig_v)$ to all other nodes. Each node receives the signed shares, verifies the signatures, and uses the at least $f + 1$ valid shares to recover the pair $(tx, nonce)$. Finally the node checks whether $hash(tx, nonce) = h$, aborting the protocol otherwise.

(e) Each node v now knows tx and submits it timestamped to the blockchain's peer-to-peer (P2P) network as the tuple (tx, τ, σ_v).

The blockchain now operates with tuples of the form (tx, τ, σ) instead of just transactions tx. For each transaction in a block, validators check the threshold signature σ using the public keys of the nodes. Moreover, they also check that transactions are ordered in non-decreasing order by τ inside the block and that the lowest timestamp in the block is no lower than the highest timestamp in the previous block.

We now provide additional intuition for the submission protocol and reasoning behind some of the design considerations. Step (a) describes the user-sided part, while steps (b)–(e) describe the DCN-sided part.

In step (a) the user hashes tx together with a random nonce and sends it to the nodes. The nonce is required to prevent malicious actors from inferring information about tx based on past transaction data; e.g. if a user submits similar transactions periodically, they can be identified by their hash and front-run, e.g., buying ETH every time they receive their paycheck. Moreover, the transaction-nonce pair is split into n shares which are distributed to the n nodes. This allows the DCN to recover the pair $(tx, nonce)$ after agreeing on timestamp τ, check its integrity against the hash h, and submit tx to the blockchain on the user's behalf, preventing users from submitting many timestamping requests without submitting matching transactions to the blockchain, which would be the source of attacks.

In step (b) the DCN agrees on a common timestamp τ for transaction tx using the agreement protocol described later on in Section 4, which is efficient, robust to at most $f < n/3$ byzantine failures in both synchronous and asynchronous settings, and achieves good fairness guarantees, whose definitions we postpone to the next section.

In step (c) the DCN computes threshold signatures for the pair (h, τ) consisting of the transaction hash together with timestamp τ. Any valid such signature can be used to prove that at least $f + 1$ nodes have agreed on it; i.e., at least one honest node.

In step (d) the nodes circulate their shares to recover the pair $(tx, nonce)$. Note that this has to be done after agreeing on the timestamp because otherwise a byzantine node could front-run tx by submitting its own transaction and having agreement happen for it faster than for tx. Moreover, checking the hash of the pair against h is required to prevent dishonest users from sending contradicting shares. Note that steps (c) and (d) can be implemented concurrently, but we chose not to do so for simplicity of exposition.

Finally, in step (e) each node v submits tx together with timestamp τ and threshold signature σ_v to the blockchain, which will handle checking the signatures and ensuring that transactions are ordered by timestamp inside each block and across blocks. Note that different nodes might compute different threshold signatures σ_v, even in the presence of no byzantine nodes, because of the choice of which individual signatures to include in σ_v, but any valid such signature is enough to certify the tuple (tx, τ, σ_v). We further note that validators will of course check that the transaction tx is only executed once.

We state our protocol's guarantees in the theorem below, which we prove in Section 5. We provide a formal definition for the term *fair timestamp* in Section 3.

▶ **Theorem 1.** *The transaction submission protocol achieves the following properties:*

- *(Honest-User Liveness) If a transaction is sent by an honest user, it gets processed and submitted to the mempool eventually. Moreover, if the honest user's messages reach the nodes within Δ_{EXT} time and the synchrony assumptions hold inside the DCN for an additional Δ_{DCN} time, the transaction gets submitted within expected $\mathcal{O}(\log \Delta_{EXT})$ communication rounds.*
- *(Integrity) If a transaction gets submitted to the mempool, the process was initiated by some user.*

- *(Unique Timestamp) If a transaction gets submitted to the mempool by two nodes, with timestamps τ and τ', then $\tau = \tau'$.*
- *(Fair Timestamp) If a transaction gets submitted to the mempool with timestamp τ, then τ is a* fair *timestamp.*

3 Timestamp Agreement and The Fairness Guarantees

Nodes in the DCN need to agree on a timestamp for each transaction. This problem reduces to achieving asynchronous Byzantine Agreement (aBA), with a special Validity condition, which will allow us to argue why transactions are ordered in a fair manner. We recall the classical definition of aBA, which requires the following properties: *(Weak Validity)* If all honest nodes have input τ, no honest node outputs $\tau' \neq \tau$; *(Agreement)* If two honest nodes output τ and τ', then $\tau = \tau'$; *(Termination)* Every honest node outputs with probability 1.

While aBA is an essential building block in distributed computing, it comes with many limitations. We first note the seminal result of [19], which proves that fault-tolerant aBA, even with binary inputs, cannot be solved deterministically. There is still hope however, as the distributed computing literature offers plenty of randomized aBA protocols [6, 11, 13, 15, 22, 32, 36, 41].

Unfortunately, there is another limitation that prevents us from directly applying existing aBA protocols to our setting, standing in its Weak Validity condition: this only ensures that honest nodes agree on an honest input if they joined aBA with the same input. This pre-agreement condition is a very strong requirement in our setting, and hence nodes may often end up agreeing on timestamps proposed by corrupted nodes. Such timestamps may be too low or too high, preventing us from ensuring any kind of fair ordering. We add that achieving a stronger condition that requires the honest nodes to always agree on some honest node's input is impossible, as one cannot distinguish between an honest node and a byzantine node following the protocol correctly, but with a corrupted input.

Meaningful Timestamps. Fortunately, there are still a few Validity variations we can consider. In the following definitions, we will make use of the timestamps that the nodes record when receiving messages from the user. We need to consider that, if the user is dishonest, some honest nodes might not hold such a timestamp. Note that there is at least one honest node who has received a message from this user (otherwise the user is essentially not sending a transaction). Then, let $\tau_{\max}$ denote the latest point in time recorded by an honest node when receiving this user's message. In the definitions, we assume that, if an honest node does not receive such a message, its input is $\tau_{\max}$. We stress that this assumption is strictly for simplicity of presentation and is not used in our protocols or their analysis.

With this convention in mind, we may provide stronger Validity definitions. In our setting, ensuring that honest nodes' outputs are in their inputs' range is already meaningful (*Honest-Range Validity*). This enables the order fairness definition below, discussed in [46].

▶ **Definition 2** (Honest-Range Fairness). *Let tx and tx′ denote two transactions. If all honest nodes receive the hash of tx before any honest node receives the hash of tx′, then tx will be ordered before tx′.*

Honest-Range Validity has been studied in the synchronous setting [42]. In the asynchronous setting, however, this condition has only been considered under much weaker Agreement requirements, which allow the honest outputs to be ε-close for some predefined $\varepsilon > 0$; see [3].

One could hope that a stronger order-fairness definition is possible. Our first attempt is as follows: if, at some time τ, most honest nodes have received the hash of some transaction

tx, while most honest nodes are yet to receive the hash of some transaction tx', then tx should be ordered before tx'. We express this condition with the help of the medians of the honest nodes' receipt timestamps:

▶ **Definition 3** (Median Fairness). *Suppose the hashes of transactions tx and tx' are received by the honest nodes at times $\tau_1 \leq \tau_2 \leq \ldots \leq \tau_{n-f}$ and resp. $\tau'_1 \leq \tau'_2 \leq \ldots \leq \tau'_{n-f}$. Then, if $\tau_\mu < \tau'_\mu$, where $\mu = \lceil (n-f)/2 \rceil$ denotes the index of the median, tx will be ordered before tx'.*

To achieve this order fairness definition, we need honest nodes to agree on the median of their timestamps. Consider the (δ-Median Validity) condition below, introduced by Stolz and Wattenhofer in [39], for $n > 3f$.

- (δ-Median Validity) Assume the honest inputs are arranged in non-decreasing order in an array T, and T_i is the i-th value in T. If an honest node outputs τ, then $\tau \in [T_{\mu-\delta}, T_{\mu+\delta}]$ (i.e., τ is δ-positions-close to T_μ), where $\mu = \lceil (n-f)/2 \rceil$.

Then, Median Fairness requires 0-Median Validity. This definition however cannot be achieved even in a synchronous network, as stated in Lemma 4, following directly from [31,39].

▶ **Lemma 4.** *If $n > 3f$ and $\delta < \lceil f/2 \rceil$, there is no synchronous protocol achieving Termination and δ-Median Validity.*

We therefore weaken our Median Fairness definition to allow some error.

▶ **Definition 5** (δ-Median Fairness). *Suppose the hashes of transactions tx and tx' are received by the honest nodes at times $\tau_1 \leq \tau_2 \leq \ldots \leq \tau_{n-f}$ and $\tau'_1 \leq \tau'_2 \leq \ldots \leq \tau'_{n-f}$ respectively. Let $\mu = \lceil (n-f)/2 \rceil$ denote the index of the median. Then, if $\tau_{\mu+\delta} < \tau'_{\mu-\delta}$, transaction tx will be ordered before transaction tx'.*

We note that δ-Median Validity has only been considered in the synchronous model [31,39], meaning that even the slightest increased network delay may cause the protocols of [31,39], which achieve δ-Median Validity, to completely fall apart. This motivates us to study δ-Median Validity in the asynchronous model. First, we show a lower bound on the δ achievable for the asynchronous case. Later on, we will further show this bound to be tight.

▶ **Lemma 6.** *If $n > 3f$ and $\delta < f$, there is no asynchronous protocol achieving Termination and δ-Median Validity.*

Proof. We assume that there is a protocol Π achieving δ-Median Validity and Termination. Let $\mu = \lceil (n-f)/2 \rceil$, and let v denote an honest node. The input value of node v will be $2f+1$. We define the following scenarios:

(a) The $n-f$ honest nodes have inputs $f+1, f+2, \ldots n$, and the corrupted parties do not participate in the protocol. Then, v must output a value in $[f+\mu-\delta, f+\mu+\delta]$.

(b) The $n-f$ honest nodes have inputs $1, 2, \ldots, n-f \geq 2f+1$. The f corrupted nodes follow the protocol correctly with inputs $n-f+1, n-f+2, \ldots, n$, while the messages of the honest nodes holding the f lowest inputs are delayed. Here, v should output a value in $[\mu-\delta, \mu+\delta]$. However, since from node v's perspective, this scenario is indistinguishable from Scenario (a), v must output a value in $[\mu-\delta, \mu+\delta] \cap [f+\mu-\delta, f+\mu+\delta] = [f+\mu-\delta, \mu+\delta]$.

(c) The $n-f$ honest nodes have inputs $2f+1, 2f+2, \ldots, n+f$. The f corrupted nodes follow the protocol correctly with inputs $f+1, f+2, \ldots, f$, while the messages of the f honest nodes holding the f highest inputs are delayed. Here, v should output a value in $[2f+\mu-\delta, 2f+\mu+\delta]$. Note that, for node v, this scenario is indistinguishable from Scenario (a) and Scenario (b). Therefore, node v must output a value in $[f+\mu-\delta, \mu+\delta] \cap [2f+\mu-\delta, 2f+\mu+\delta] = [2f+\mu-\delta, \mu+\delta]$.

Since $\delta < f$, we obtain that $\mu+\delta < \mu+f < 2f+\mu-\delta$, therefore the interval $[2f+\mu-\delta, \mu+\delta]$ containing node v's output is empty. This contradicts that Π achieves Termination. ◀

Lemma 6 showcases an important limitation, namely, in a purely asynchronous network, if $n = 3f+1$, one can only hope to achieve Honest-Range Validity, as in this case f-Median Validity degenerates to Honest-Range Validity. We note here that previous work has shown that a single byzantine node can manipulate the median [27]. However, as timestamps satisfying f-Median Validity are still in the honest range and as transactions are not visible during ordering, we do not see it as a threat.

Defining Timestamp Agreement. To mitigate the limitation posed by Lemma 6, we take into account that real-world networks are not as unreliable as the asynchronous model. Hence, we aim to provide better guarantees if the network *happens to be synchronous*. We investigate whether we can achieve best-of-both-worlds guarantees, in line with many recent works [4, 7, 9, 17, 23]. That is, we investigate whether there is an asynchronous protocol ensuring f-Median Validity that can additionally offer the stronger guarantee of $\lceil f/2 \rceil$-Median Validity if the network *happens to be synchronous* for sufficient time. Therefore, we introduce the following variant of aBA.

▶ **Definition 7** (Timestamp Agreement). *An n-nodes protocol, where each node may hold an integer timestamp as input, achieves Timestamp Agreement (TA) if, even when f of the nodes are corrupted, it achieves Agreement, f-Median Validity, and the following hold:*

- *if all honest nodes hold inputs, then all honest node obtain outputs with probability* 1;
- *if less than $f+1$ honest nodes hold inputs, then no honest node obtains output;*
- *if the synchrony assumptions hold for a sufficient amount of time and all honest parties receive their inputs accordingly, then $\lceil f/2 \rceil$-Median Validity is achieved.*

We note that we have proposed this definition taking into account that the user is not necessarily honest, and hence may not provide all honest nodes with inputs. If this is the case, our protocol still maintains f-Median Validity and Agreement. For the timestamp submission protocol, this implies that, if a dishonest user's transaction gets submitted to the chain, then the unique timestamp assigned to it still fits our f-Median Fairness definition. Hence, such adversarial behavior does not bring the dishonest user any real advantage.

We may now also define the term *fair* timestamp, used in Theorem 1: it is a timestamp satisfying f-Median Validity, and, if synchrony assumptions hold, $\lceil f/2 \rceil$-Median Validity.

4 The Timestamp Agreement Protocol

In this section, we present our protocol achieving Timestamp Agreement secure against $f < n/3$ byzantine corruptions. Formally, we obtain the result below. Recall once again that there is no deterministic protocol achieving asynchronous Timestamp Agreement, a fact following directly from FLP [19].

▶ **Theorem 8.** *There is an n-nodes randomized protocol Π_{TA} achieving TA resilient against $f < n/3$ byzantine corruptions. Π_{TA} has expected round complexity $\mathcal{O}(\log(\tau_{\max} - \tau_{\min}))$, where $\tau_{\min}$ and $\tau_{\max}$ denote the lowest and the highest honest inputs respectively. To achieve $\lceil f/2 \rceil$-Median Validity, the synchrony assumptions must to hold for $\Delta_{EXT} + \Delta_{DCN}$ time.*

Our protocol Π_{TA} consists of three steps. First, each node obtains a value satisfying δ-Median Validity. This is the only step where synchrony assumptions are required to achieve $\delta = \lceil f/2 \rceil$ instead of $\delta = f$. In the second step, nodes obtain very close values (they agree up to an error of $\varepsilon < 0.5$) within the range of values that honest nodes obtained in the first step. Agreement is then achieved in the last step, where each node decides whether to round its value obtained in the second step up or down. This will be done using aBA on the rounding option's parity. In the following, we describe each step of Π_{TA} in detail.

Step 1: δ-Median Validity. We first design a protocol Π_{init} that only focuses on achieving δ-Median Validity (while Agreement is covered by the subsequent steps).

Concretely, nodes send their input value to every party. To obtain a good estimation on the honest inputs' median, the nodes aim to receive as many honest inputs as possible. If the network is asynchronous, one may only expect to receive $n - f$ values. On the other hand, if the network is synchronous, and the user initiated the transaction at some time τ, all honest inputs are received by time $\tau + \Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$. Then, nodes wait until they have received timestamps from at least $n - f$ nodes, and, until at least $\Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$ time has passed since they have received the user's message. This way, if the synchrony assumptions hold, every honest timestamp is received.

Hence, each node v collects $n - f + k$ timestamps, where $0 \le k \le f$, and arranges them in an array R in non-decreasing order. If the network is synchronous, at most k of these values are corrupted. These may be lower than any honest input, hence shifting the honest median with at most k positions to the right, or higher than any honest input. Therefore, the honest median is in the subarray $R_\mu, R_{\mu+1}, \ldots, R_{\mu+k}$, where R_i denotes the i-th lowest value in R and $\mu = \lceil (n-f)/2 \rceil$. Then, to obtain a value that is $\lceil f/2 \rceil$-positions-close to the median, v outputs $\tau_\mu := R_{\mu + \lfloor k/2 \rfloor}$, i.e., the median of the subarray $R_\mu, R_{\mu+1}, \ldots, R_{\mu+k}$.

If the synchrony assumptions fail, however, the $n - f + k$ values from R might come from f corrupted nodes, and $n - 2f + k$ honest nodes. The $f - k$ missing honest timestamps provide the corrupted nodes with more power: shifting the honest median f positions to the right, or $f - k$ positions to the left. Regardless, the chosen output $\tau_\mu := R_{\mu + \lfloor k/2 \rfloor}$ still ensures that f-Median Validity holds.

We formally present the code of Π_{init} below. We note that this is the only step requiring synchrony for achieving $\lceil f/2 \rceil$-Median Validity. In order to achieve the same guarantee even if the nodes' clocks are not perfectly synchronized, we may replace the waiting time by $\theta \cdot (\Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}})$, to ensure that the fastest node waits long enough.

Protocol Π_{init}

Code for node v with input timestamp τ_{in}

```
1: Send your input $\tau_{\mathsf{in}}$ to all nodes.
2: After at least $\Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$ time, and when $n - f + k$ timestamps ($0 \le k \le f$) are received:
3:     $R$ := an array containing the timestamps received, ordered non-decreasingly.
4:     Output $\tau_\mu := R_{\lceil (n-f)/2 \rceil + \lfloor k/2 \rfloor}$.
```

We may now state and prove the properties of Π_{init}.

The next property enables us to ensure safety guarantees even when the user initiating the process is dishonest. It follows immediately from line 2 of Π_{init}.

▶ **Lemma 9.** *If less than $f + 1$ honest nodes provide inputs τ_{in}, then no honest node outputs. Otherwise, if each honest node provides an input τ_{in}, then all honest nodes output τ_μ.*

The following lemmas show that the nodes indeed obtain values satisfying the desired Validity guarantees.

▶ **Lemma 10.** *If an honest node outputs a timestamp τ_μ, then τ_μ satisfies f-Median Validity.*

Proof. If an honest node v has obtained a timestamp τ_μ, then it has received $n - f + k$ values, where $0 \le k \le f$. Out of these, at least $n - 2f + k$ values are honest.

Let T denote the array of honest inputs arranged in non-decreasing order. We show that $T_{\lceil (n-f)/2\rceil - f} \le \tau_\mu = R_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \le T_{\lceil (n-f)/2\rceil + f}$.

For the upper bound, note that R may miss $f - k$ out of the values in T, hence at most $f - k$ of the values T_i with $i \le \lceil (n-f)/2\rceil + \lfloor k/2 \rfloor$. This implies that $R_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \le T_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor + (f-k)} \le T_{\lceil (n-f)/2\rceil + f}$.

For the lower bound, note that R contains at most f corrupted values, hence at most f additional values that are lower than $T_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor}$. Then, we obtain that $R_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \ge T_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor - f} \ge T_{\lceil (n-f)/2\rceil - f}$. ◀

We now show that Π_{init} achieves $\lceil f/2 \rceil$-Median Validity if the synchrony assumptions hold, using a similar argument to the proof of Lemma 11. The key difference is that at least $n - f$ of the values received are honest (as opposed to $n - 2f + k$).

▶ **Lemma 11.** *If all honest nodes obtain inputs τ_{in} and join Π_{init} between time τ_{start} and time $\tau_{start} + \Delta_{EXT}$, and the synchrony assumptions hold until time $\tau_{start} + \Delta_{EXT} + \Delta_{DCN}$, then all honest nodes output timestamps τ_μ satisfying $\lceil f/2 \rceil$-Median Validity.*

Proof. We first show that all honest timestamps are received by time $\tau_{\mathsf{start}} + \Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$. Each honest node sends its input to all other nodes by time $\tau_{\mathsf{start}} + \Delta_{\mathsf{EXT}}$. Since the network is synchronous, these values are received within Δ time, hence by time $\tau_{\mathsf{start}} + \Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$. Then, since all honest nodes start the execution of the protocol at time at least τ_{start}, the protocol ensures that each honest node waits until time at least $\tau_{\mathsf{start}} + \Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$, and hence receives all honest timestamps.

Then, for every honest node, R contains all honest values, and $0 \le k \le f$ values from corrupted nodes. If T denotes the array of honest timestamps arranged in non-decreasing order, we need to show that $T_{\lceil (n-f)/2\rceil - \lceil f/2 \rceil} \le R_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \le T_{\lceil (n-f)/2\rceil + \lfloor f/2 \rfloor}$.

We first focus on the upper bound: since R contains all values T_i, and $k \le f$, the inequality $R_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \le T_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \le T_{\lceil (n-f)/2\rceil + \lfloor f/2 \rfloor}$ holds.

For the lower bound, we note that R contains at most $k + \mu + \lfloor k/2 \rfloor$ values lower than $T_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor}$. Out of these $k + \mu + \lfloor k/2 \rfloor$ values, at most k are corrupted. This means that $R_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor} \ge T_{\lceil (n-f)/2\rceil + \lfloor k/2 \rfloor - k} = T_{\lceil (n-f)/2\rceil - \lceil k/2 \rceil} \ge T_{\lceil (n-f)/2\rceil - \lceil f/2 \rceil}$, which concludes our proof. ◀

Step 2: Agreement up to a small error. Honest nodes have obtained timestamps τ_μ satisfying δ-Median Validity via Π_{init}. We now take a step towards achieving Agreement. We make use of an asynchronous protocol Π_{AA} achieving Approximate Agreement [3]. That is, Π_{AA} ensures that, for any given $\varepsilon > 0$, honest nodes obtain ε-close values τ_{AA} within the range of their values τ_μ (maintaining δ-Median Validity). Lemma 12 states the properties of Π_{AA}, and follows directly from [3]. In our case, any constant $\varepsilon < 0.5$ suffices.

▶ **Lemma 12.** *If an honest node outputs τ_{AA}, then τ_{AA} is within the range of timestamps τ_μ obtained by honest nodes in Π_{init}. If two honest nodes output τ_{AA} and τ'_{AA}, then $\left|\tau_{AA} - \tau'_{AA}\right| < \varepsilon < 0.5$. In addition, if less than $f + 1$ honest nodes hold timestamps τ_μ, then no honest node outputs; while if all honest nodes hold timestamps τ_μ, then all honest nodes output.*

The guarantee on obtaining outputs only when at least $f+1$ honest nodes participate follows from the fact that Π_{AA} requires nodes to wait for messages from $n-f$ distinct nodes. In addition, properties on honest nodes' outputs (if any) are ensured even if not all honest nodes participate since Π_{AA} is an asynchronous protocol. Concretely, this is because this setting is indistinguishable from a scenario where the non-participating honest nodes are simply delayed.

We add that Π_{AA} does not make any assumption on the range of honest values τ_μ to achieve these guarantees. It runs in iterations allowing the honest values to converge. If all honest nodes hold inputs τ_μ and range size of these inputs is Δ_μ ($\leq$ the difference between the times when the transaction hash is delivered to the honest nodes), then Π_{AA} runs for $\mathcal{O}(\log(\Delta_\mu/\varepsilon))$ iterations. Each iteration consists of a constant number of communication rounds, and incurs message complexity $\mathcal{O}(n^3)$. Therefore, the round complexity of Π_{AA} is $\mathcal{O}(\log \Delta_\mu)$, and the message complexity is $\mathcal{O}(n^3 \log \Delta_\mu)$.

Step 3: Rounding. Honest nodes have obtained ε-close values τ_{AA} satisfying δ-Median Validity. As depicted in Figure 2, since $\varepsilon < 0.5$, the range of honest values τ_{AA} either:

(a) contains an even integer α such that $\left|\alpha - \tau_{\mathsf{AA}}\right| < 0.5$ for all honest values τ_{AA};
(b) contains an odd integer α such that $\left|\alpha - \tau_{\mathsf{AA}}\right| < 0.5$ for all honest values τ_{AA};
(c) is between two integers: $\alpha \leq \tau_{\mathsf{AA}} \leq \alpha + 1$ for all honest values τ_{AA}.

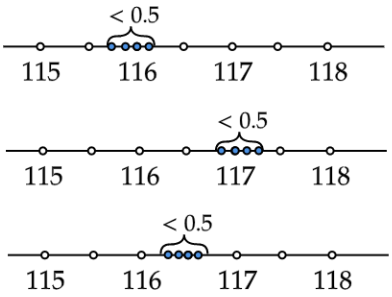


Figure 2 A few examples of possible outputs obtained in Π_{AA}. In the top and middle examples, representing cases (a) and (b) respectively, honest outputs are close to a single integer. In the bottom example, representing case (c), some honest outputs are closer to 116, while some are closer to 117.

Then, the problem of achieving Agreement comes down to enabling the honest nodes to choose between rounding down or rounding up their values τ_{AA}. Making this decision for cases (a) and (b) is trivial: honest nodes simply round their value τ_{AA} to the closest integer. Case (c), however, requires solving aBA. We therefore employ the randomized protocol Π_{aBA} of [32] that achieves aBA with binary inputs in expected round complexity $\mathcal{O}(1)$, with message complexity $\mathcal{O}(n^2)$. Note that we do not use aBA to decide on rounding either up or down, since this would break Agreement in cases (a) and (b). Instead, we use aBA to decide on the parity of the final rounding option. Once again, we note that if the user is dishonest and not all honest nodes were able to reach this stage, protocol Π_{aBA} still offers guarantees. Namely, if less than $f+1$ honest nodes have reached this stage, then no honest node obtains an output (as honest nodes are forced to wait for messages from $n-f$ distinct nodes). Otherwise, if honest nodes obtain outputs, these outputs still satisfy Weak Validity and Termination. This is the case even if not all honest nodes participate, since such a setting is indistinguishable from a scenario where the non-participating honest nodes' messages are simply delayed, and the guarantees of Π_{aBA} hold under asynchrony.

Each node v that has obtained a timestamp τ_{AA} picks two integers α and $\alpha+1$ such that $\alpha \leq \tau_{\mathsf{AA}} < \alpha+1$. Out of these two values, v picks the one that is closer to its τ_{AA} as an initial rounding option, denoted by β. Then, v joins Π_{aBA} with input b, representing the parity of β, and may obtain output b'. If $b' = b$, it outputs β, and otherwise it outputs its second rounding option. In cases (a) and (b), honest nodes pick the same value β. They join Π_{aBA} with the same input bit b, and Weak Validity ensures they output $b' = b$. Therefore, if sufficiently many honest nodes reached this stage, all participating honest nodes output β, which still satisfies δ-Median Validity. In case (c), all honest nodes that reach this stage pick the same value α. In this case, because the input timestamps τ_{in} are integers, both α and $\alpha+1$ satisfy δ-Median Validity. Even if honest nodes make a different choice for β, Π_{aBA} allows them to decide on the same bit b', hence they output the same rounding option.

We may now provide the formal code of our Timestamp Agreement protocol Π_{TA}. We define the constant $\varepsilon = 0.49$, but any choice of $\varepsilon < 0.5$ suffices.

Protocol Π_{TA}

Code for node v receiving a transaction at time τ_{in}

```
Join $\Pi_{\mathsf{init}}$ with input $\tau_{\mathsf{in}}$. Upon obtaining $\tau_\mu$ via $\Pi_{\mathsf{init}}$:
    Join $\Pi^{\varepsilon}_{\mathsf{AA}}$ with input $\tau_\mu$. Upon obtaining output $\tau_{\mathsf{AA}}$ in $\Pi^{\varepsilon}_{\mathsf{AA}}$:
        Let $\alpha$ be an integer such that $\alpha \leq \tau_{\mathsf{AA}} < \alpha + 1$.
        If $\tau_{\mathsf{AA}} - \alpha < \alpha + 1 - \tau_{\mathsf{AA}}$, set $\beta = \alpha$ and $\beta' = \alpha + 1$.
        Otherwise, set $\beta = \alpha + 1$ and $\beta' = \alpha$.
        Set $b = 0$ if $\beta$ is even, and $b = 1$ if $\beta$ is odd.
        Join $\Pi_{\mathsf{aBA}}$ with input $b$. Upon obtaining output $b'$ via $\Pi_{\mathsf{aBA}}$:
            If $b = b'$ , set $\tau_{\mathsf{out}} = \beta$. Otherwise, set $\tau_{\mathsf{out}} = \beta'$. Output $\tau_{\mathsf{out}}$ and terminate.
```

We now focus on proving Theorem 8. In Lemma 13, we show that Π_{TA} indeed achieves Timestamp Agreement. Then, Lemma 14 focuses on the round complexity. The requirement of synchrony assumptions holding only for $\Delta_{\mathsf{EXT}} + \Delta_{\mathsf{DCN}}$ in order to achieve $\lceil f/2 \rceil$-Median Validity is given by Π_{init}, since the subsequent steps of Π_{TA} are fully asynchronous.

▶ **Lemma 13.** *If less than $f+1$ honest nodes hold inputs τ_{in}, then no honest node outputs. Otherwise, honest nodes that output have obtained the same value τ_{out} satisfying δ-Median Validity, with $\delta = \lceil f/2 \rceil$ if the synchrony assumptions held for $\Delta_{EXT} + \Delta_{DCN}$ time at the beginning of the protocol's execution, and $\delta = f$ otherwise.*

In addition, if all honest nodes hold inputs τ_{in}, then all honest nodes output.

Proof. Lemma 9 ensures that $f+1$ honest nodes holding inputs τ_{in} are necessary in order to obtain outputs. In the following, we assume this was the case.

Lemma 12 ensures that honest nodes obtaining outputs (meaning all honest nodes if all of them had inputs τ_{in}) have obtained ε-close approximations τ_{AA} for $\varepsilon < 0.5$. These approximations are within the range of honest values τ_{in}, and hence satisfy δ-Median Validity, as ensured by Lemma 10 and Lemma 11.

Then, we need to consider two cases: when all obtained honest approximations are between two consecutive integers, and when some honest approximations are lower than an integer, while some are higher.

If there is some integer γ such that $\gamma \leq \tau_{\mathsf{AA}} < \gamma+1$ for all obtained honest approximations τ_{AA}, then honest nodes obtain $\alpha = \gamma$ and $\alpha + 1 = \gamma + 1$. Regardless of the chosen β and bit b, honest nodes obtain in Π_{aBA} the same bit b' which refers to the same value: either γ for all honest nodes that reached this stage, or $\gamma + 1$ for all honest nodes that reached this stage. Hence, these honest nodes output the same timestamp. It remains to show that the output timestamp is in the range of honest inputs. If all honest nodes that reached this stage have obtained $\tau_{\mathsf{AA}} = \gamma$, then they joined Π_{aBA} with the same input b representing γ's parity, and hence they output the same γ in the honest range according to Lemma 12. Otherwise, if at least one honest node has obtained $\gamma < \tau_{\mathsf{AA}} < \gamma + 1$, we take into account that the honest inputs are integers. Lemma 12 then implies that both γ and $\gamma + 1$ are in the honest inputs' range.

Otherwise, there is some integer γ such that $\gamma \leq \tau_{\mathsf{AA}} < \gamma{+}1$ for some honest approximation τ_{AA} and $\gamma + 1 \leq \tau'_{\mathsf{AA}} < \gamma + 2$ for some honest approximation τ_{AA}. Note that, in this case, Lemma 12 ensures that $\gamma + 1$ is in the range of the honest nodes' inputs. In addition, since Lemma 12 ensures $\tau'_{\mathsf{AA}} - \tau_{\mathsf{AA}} < 0.5$, both $\gamma + 1 - \tau_{\mathsf{AA}} < 0.5$ and $\tau'_{\mathsf{AA}} - (\gamma + 1) < 0.5$ hold. This applies to all honest nodes that reached this stage: namely, all these honest nodes choose the same $\beta = \gamma + 1$ and therefore join Π_{aBA} with the same bit b. Then, Π_{aBA} ensures all honest nodes that reached this stage output $b' = b$ and output $\gamma + 1$.

If all honest nodes had inputs τ_{in}, all honest nodes have obtained outputs in Π_{aBA}, and therefore all honest nodes output in Π_{TA}. ◀

The round complexity of Π_{TA} follows from the fact that Π_{AA} ensures termination within $\mathcal{O}(\log(\tau_{\max} - \tau_{\min}))$ rounds, if honest nodes' inputs are between $\tau_{\min}$ and $\tau_{\max}$, while Π_{aBA} ensures termination within expected constant time.

▶ **Lemma 14.** *If all honest nodes hold inputs τ_{in}, then honest nodes output within expected $\mathcal{O}(\log(\tau_{\max} - \tau_{\min}))$ rounds, where $\tau_{\min}$ and $\tau_{\max}$ denote the lowest and the highest honest inputs respectively (hence $\mathcal{O}(\log \Delta_{EXT})$ rounds if the synchrony assumptions are satisfied).*

5 Analysis of the Main Protocol

We now formally prove the properties of the transaction submission protocol. In particular, we prove Theorem 1.

▶ **Lemma 15** (Honest-User Liveness). *If a transaction tx is sent by an honest user, it gets processed and submitted to the mempool eventually, and, if the user's messages reach the nodes within Δ_{EXT} time and the synchrony assumptions hold inside the DCN for an additional Δ_{DCN} time, the transaction get submitted within expected $\mathcal{O}(\log \Delta_{EXT})$ communication rounds.*

Proof. Since tx was sent by an honest user, all honest nodes receive the necessary messages to join Π_{TA}, and hence they obtain a timestamp τ. Then all honest nodes obtain τ and send their shares to the other nodes. Since $f + 1$ shares are necessary to reconstruct tx and the shares are signed by the user (therefore the corrupted nodes cannot send corrupted shares), the honest nodes are able to reconstruct the transaction and submit it to the mempool. The round complexity follows from Lemma 14, and from the fact that the main protocol only adds a constant number of communication rounds over Π_{TA}. ◀

▶ **Lemma 16** (Integrity). *If a transaction tx gets submitted to the mempool, the process was initiated by some user.*

Proof. Submitting a transaction to the mempool requires signatures from $f + 1$ nodes, hence from at least one honest node. This honest node only signs if it has obtained output in the invocation of Π_{TA} corresponding to the hash of tx. This means that honest nodes have joined this execution of Π_{TA}, hence they have received input from some user. ◀

▶ **Lemma 17** (Unique Timestamp). *If a transaction tx gets submitted to the mempool with timestamps τ and τ', then $\tau = \tau'$.*

Proof. Assume that $\tau \neq \tau'$. First, note that timestamps are obtained via Π_{TA}, which assigns *tx* a unique timestamp by the Agreement property. Therefore, during an invocation of the main protocol for *tx*, all honest nodes obtain the same timestamp τ. Since $f+1$ signatures are required for the transaction to be submitted along with its timestamp, the corrupted parties are unable to submit *tx* to the mempool on their own. Hence, if $\tau \neq \tau'$, there must be an honest party that has signed τ', which happened through a different invocation of the main protocol, hence for a different transaction (ensured by the transaction's nonce). ◀

▶ **Lemma 18** (Fair Timestamp). *If a transaction tx gets submitted to the mempool with timestamp τ, then τ is a* fair *timestamp.*

Proof. Since *tx* was assigned a timestamp obtained via Π_{TA}, all honest parties have assigned a fair timestamp τ to *tx*, i.e., satisfying f-Median Validity or $\lceil f/2 \rceil$-Median Validity, depending on the network conditions and on the user's honesty. Then, since $f+1$ signatures are required for *tx* to be submitted, its timestamp was signed by an honest party, hence it is fair. ◀

6 Discussion

Front-running Resistance. The DCN effectively prevents tolerant front-running, i.e., the attacker's transaction executing before the victim's transaction. Transactions are ordered according to the timestamps returned by the DCN, which fulfill δ-Median Validity. Still, transactions submitted close to each other in time could receive the same timestamp, in which case the validator picks an order, or receives timestamps in the opposite order of the actual submission times. However, this is not an issue, as the transaction contents are hidden until the timestamp is agreed upon by the nodes. Thus, tolerant front-running, which, to be effective, requires the attacker to know the contents of the victim's transaction, is prevented.

The DCN does not address destructive front-running. To be more easily integrateable in the current blockchain infrastructure, the permissioned DCN only supplies the timestamp but does not interfere with the blockchain's consensus. Destructive front-running, thus, remains possible, as the block proposer (miner) could choose not to include the transaction.

Censorship Resistance. We note that by using timestamps as a decision factor when including transactions in a block, transactions become in some sense *block-bound*. Thus, a transaction can become temporarily censored if the block proposer does not include the transaction and the transaction's timestamp is too low to be included in future blocks. Further, under an asynchronous network, transactions may get lost solely due to messages getting delayed and the corresponding timestamp becoming obsolete by the time messages reach the validators. In a real-world implementation of our system, this issue can be circumvented by allowing users to resubmit their transactions with new nonces if they get lost. Importantly, the DCN does not decrease the censorship resilience any further than the block-bound model does in comparison to the classical non-block-bound blockchain model, while having the advantage of providing guarantees against front-running, which neither the classical nor the block-bound model can achieve alone. This follows both under synchrony and asynchrony by the Honest-User Liveness property.

Permissioned Network. The DCN is designed as a permissioned network consisting of specialized parties offering efficient and reliable transaction timestamping. Importantly, the DCN only supplies transactions with timestamps and is therefore designed to be used together with an existing permissionless blockchain. In particular, the responsibility of adding blocks to the ledger, validating blocks and storing the blockchain itself remains in the hand of the permissionless set of miners or validators. Essentially, the permissioned nature of the DCN does not reduce the robustness and decentralization of the network of validators that verify the blocks, i.e., proves that they are honest. The permissionless network, thus, retains control of the most fundamental task.

Similar permissioned setups are already common in practice today. For instance, Chainlink oracles bringing price data from the real world onto the blockchain usually operate in a similar fashion [1]. Moreover, since Ethereum's transition from Proof-of-Work to Proof-of-Stake, block building has become more concentrated [24,43,45], in that currently more than 90% of the blocks are built with *proposer-builder separation (PBS)* [2]. In the first six months since the merge, a mere 133 builders have built these PBS blocks that were included on the ledger [24]. With PBS, block building is no longer done by the validators themselves but is instead handled by highly sophisticated block builders [2], similarly in spirit to how the DCN is used for timestamping transactions. By shifting tasks requiring a high degree of complexity away from validators, such as building blocks, or timestamping transactions in the case of the DCN, the requirements to run a validator node decrease. Consequently, in the long run, the number of validators is expected to increase, leading to a higher overall degree of decentralization of the consensus layer [2,34], i.e., the core of the blockchain. In our case, parties participating in the DCN are now responsible for the non-trivial task of ordering transactions, while the complexity for the validators decreases. In particular, the task of block building becomes easier as validators must simply order transactions according to their timestamp.

Finally, we note that PBS and the DCN are incompatible. While the former optimizes for block value and thereby likely includes front-running transactions, the latter is designed to achieve a fair ordering that prevents front-running. If the DCN were integrated into a permissionless blockchain instead of PBS, the blockchain would protect users from front-running as opposed to maximizing block value on their behalf.

7 Conclusion and Future Work

We introduced the DCN, a novel and practical solution for fair transaction ordering in permissionless blockchains. Our approach differs from previous works by treating fair ordering as a Byzantine Agreement problem rather than a Byzantine State Machine Replication problem, leading to a simpler and faster algorithm while achieving good fairness guarantees. In particular, our new timestamp agreement protocol achieves $\lceil f/2 \rceil$-Median Fairness when the network is synchronous and falls back to a guarantee of f-Median Fairness during periods of asynchrony. These two bounds are the best that can be obtained in terms of δ-Median Fairness for the synchronous and asynchronous cases, respectively, as we have shown. The asynchronous fallback paradigm is a relatively unexplored, yet more robust notion than partial synchrony, so we find it natural to use it in designing other blockchain network protocols under realistic conditions.

As a next step, it would be valuable to consider the implementation of a dynamic set of nodes in the DCN, supporting the addition and removal of nodes in a controlled manner and the updating of related information. To do so, it will also be important to provide incentives for the nodes. One possible way to do so is to use rewards coming from transaction fees, similar to gas fees in other blockchain systems. Additionally, it would be of interest to develop a prototype of our proposed method and evaluate its performance on-chain. Finally, the DeFi scene would benefit from the development of approaches for combating destructive front-running, which our work does not address.

References

1 Oracles - defillama, 2023. URL: `https://defillama.com/oracles`.

2 Proposer-builder separation, 2023. URL: `https://ethereum.org/nl/roadmap/pbs/`.

3 Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In Teruo Higashino, editor, *Principles of Distributed Systems*, pages 229–239, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

4 Ananya Appan, Anirudh Chandramouli, and Ashish Choudhury. Perfectly-secure synchronous mpc with asynchronous fallback guarantees. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 92–102, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3519270.3538417`.

5 Carsten Baum, James Hsin-yu Chiang, Bernardo David, Tore Kasper Frederiksen, and Lorenzo Gentile. Sok: Mitigation of front-running in decentralized finance. *Cryptology ePrint Archive*, 2021.

6 P. Berman and J.A. Garay. Randomized distributed agreement revisited. In *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*, pages 412–419, 1993. `doi:10.1109/FTCS.1993.627344`.

7 Erica Blum, John Katz, and Julian Loss. Synchronous consensus with optimal asynchronous fallback guarantees. In *Theory of Cryptography Conference*, 2019.

8 Erica Blum, Jonathan Katz, and Julian Loss. Tardigrade: An atomic broadcast protocol for arbitrary network conditions. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27*, pages 547–572. Springer, 2021.

9 Erica Blum, Chen-Da Liu-Zhang, and Julian Loss. Always have a backup plan: Fully secure synchronous mpc with asynchronous fallback. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 707–731, Cham, 2020. Springer International Publishing.

10 Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 514–532. Springer, 2001.

11 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987. `doi:10.1016/0890-5401(87)90054-X`.

12 Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 524–541, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

13 Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 123–132, 2000.

14 Christian Cachin, Jovana Mićić, Nathalie Steinhauer, and Luca Zanolini. Quick order fairness. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 316–333. Springer, 2022.

15 Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 42–51, New York, NY, USA, 1993. Association for Computing Machinery. `doi:10.1145/167088.167105`.

16 ChainLink Labs. Fair Sequencing Service (FSS), 2020. URL: `https://blog.chain.link/chainlink-fair-sequencing-services-enabling-a-provably-fair-defi-ecosystem/`.

17 Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In *Theory of Cryptography Conference*, pages 623–653. Springer, 2021.

18 Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, May 1986. doi:10.1145/5925.5931.

19 Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

20 Flashbots. Flashbots. URL: https://docs.flashbots.net/.

21 Flashbots. Mev-explore. URL: https://explore.flashbots.net/.

22 R. Friedman, A. Mostefaoui, and M. Raynal. Simple and efficient oracle-based consensus protocols for asynchronous byzantine systems. *IEEE Transactions on Dependable and Secure Computing*, 2(1):46–56, 2005. doi:10.1109/TDSC.2005.13.

23 Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Optimal synchronous approximate agreement with asynchronous fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 70–80, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538442.

24 Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum's proposer-builder separation: Promises and realities. In *2023 ACM Internet Measurement Conference (IMC), Montreal, QC, Canada*, October 2023.

25 Lioba Heimbach and Roger Wattenhofer. SoK: Preventing Transaction Reordering Manipulations in Decentralized Finance. In *4th ACM Conference on Advances in Financial Technologies (AFT), Cambridge, Massachusetts, USA*, September 2022.

26 Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, strong order-fairness in byzantine consensus. *Cryptology ePrint Archive*, 2021.

27 Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III 40*, pages 451–480. Springer, 2020.

28 Klaus Kursawe. Wendy, the good little fairness widget: Achieving order fairness for blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 25–36, 2020.

29 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982. doi:10.1145/357172.357176.

30 Christoph Lenzen and Julian Loss. Optimal clock synchronization with signatures. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 440–449, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538444.

31 Darya Melnyk and Roger Wattenhofer. Byzantine agreement with interval validity. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 251–260. IEEE, 2018.

32 Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous binary byzantine consensus with $t < n/3$, $O(N^2)$ messages, and $O(1)$ expected time. *J. ACM*, 62(4), sep 2015. doi:10.1145/2785953.

33 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

34 Pablo Pettinari. Proof-of-stake vs proof-of-work, 2023. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/pos-vs-pow/.

35 Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214. IEEE, 2022.

36 Michael O. Rabin. Randomized byzantine generals. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, SFCS '83, page 403–409, USA, 1983. IEEE Computer Society. doi:10.1109/SFCS.1983.48.

37 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979. doi:10.1145/359168.359176.

38 Victor Shoup. Practical threshold signatures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 207–220. Springer, 2000.

39 David Stolz and Roger Wattenhofer. Byzantine agreement with median validity. In *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, volume 46, page 22. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.

40 David Stolz and Roger Wattenhofer. Byzantine Agreement with Median Validity. In Emmanuelle Anceaume, Christian Cachin, and Maria Potop-Butucaru, editors, *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, volume 46 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6591`, `doi:10.4230/LIPIcs.OPODIS.2015.22`.

41 Sam Toueg. Randomized byzantine agreements. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, PODC '84, page 163–178, New York, NY, USA, 1984. Association for Computing Machinery. `doi:10.1145/800222.806744`.

42 Nitin H Vaidya and Vijay K Garg. Byzantine vector consensus in complete graphs. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 65–73, 2013.

43 Anton Wahrstätter, Liyi Zhou, Kaihua Qin, Davor Svetinovic, and Arthur Gervais. Time to bribe: Measuring block construction market. *arXiv preprint arXiv:2305.16468*, 2023.

44 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

45 Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. Sok: Mev countermeasures: Theory and practice. *arXiv preprint arXiv:2212.05111*, 2022.

46 Yunhao Zhang, Srinath Setty, Qi Chen, Lidong Zhou, and Lorenzo Alvisi. Byzantine ordered consensus without byzantine oligarchy. *Cryptology ePrint Archive*, 2020.