Ordering Transactions with Bounded Unfairness: Definitions, Complexity and Constructions

Aggelos Kiayias[1], Nikos Leonardos[2], and Yu Shen[3](✉)

[1] University of Edinburgh and IOG, Edinburgh, UK
aggelos.kiayias@ed.ac.uk
[2] National and Kapodistrian University of Athens, Athens, Greece
[3] University of Edinburgh, Edinburgh, UK

Abstract. An important consideration in the context of distributed ledger protocols is fairness in terms of transaction ordering. Recent work [Crypto 2020] revealed a connection of (receiver) order fairness to social choice theory and related impossibility results arising from the Condorcet paradox. As a result of the impossibility, various relaxations of order fairness were proposed in prior works. Given that distributed ledger protocols, especially those processing smart contracts, must serialize the input transactions, a natural objective is to minimize the distance (in terms of number of transactions) between any pair of unfairly ordered transactions in the output ledger — a concept we call *bounded unfairness*. In state machine replication (SMR) parlance this asks for minimizing the number of unfair state updates occurring before the processing of any request. This unfairness minimization objective gives rise to a natural class of parametric order fairness definitions that has not been studied before. As we observe, previous realizable relaxations of order fairness do not yield good unfairness bounds.

Achieving optimal order fairness in the sense of bounded unfairness turns out to be connected to the graph theoretic properties of the underlying transaction dependency graph and specifically the *bandwidth* metric of strongly connected components in this graph. This gives rise to a specific instance of the definition that we call "directed bandwidth order-fairness" which we show that it captures the best possible that any ledger protocol can achieve in terms of bounding unfairness. We prove ordering transactions in this fashion is NP-hard and non-approximable for any constant ratio. Towards realizing the property, we put forth a new distributed ledger protocol called Taxis that achieves directed bandwidth order-fairness. We present two variations, one that matches the property perfectly but (necessarily) lacks in performance and liveness, and another that achieves liveness and better complexity while offering a slightly relaxed version of the property. Finally, we comment on applications of our work to social choice, a direction which we believe to be of independent interest.

M. Joye and G. Leander (Eds.): EUROCRYPT 2024, LNCS 14653, pp. 34–63, 2024.
https://doi.org/10.1007/978-3-031-58734-4_2

1 Introduction

The development of blockchain protocols, starting with the Bitcoin blockchain [30], lead to increased interest in a classic problem in distributed systems — the state-machine replication (SMR) problem, cf. [34]. In SMR, the task of executing a state machine is assigned to a set of processors, and in the Byzantine fault tolerant version of the problem, processing requests should proceed unhindered by the actions of faulty nodes, even if such nodes arbitrarily deviate from the protocol in a coordinated manner.

A special case of state machine replication is the problem of ledger consensus, cf. [18–20,32], that requires the joint maintenance of a ledger of transactions so that two fundamental properties are being met: (i) consistency, i.e., the ledger of settled transactions is growing monotonically, (ii) liveness, i.e., the ledger of transactions incorporates new transactions in a timely manner. A third property related to the order of transactions has received much less attention in analysis work. In the original SMR abstraction of [34], while proper ordering of transactions is required, the *fairness* of this order is not explored.

The formal investigation of fairness in the context of ordering transactions was initiated with the elegant results of [25], which introduced it formally as "order-fairness" and pointed to an inherent impossibility to attain it in the distributed setting that relates to the Condorcet paradox. In a nutshell, (receiver) order-fairness posits that whenever two transactions $\texttt{tx}$ and $\texttt{tx}'$ are received in this order by most nodes in the system, then they should not be ordered differently in the ledger they maintain. The Condorcet paradox kicks in when cycles in the receiving order of three or more transactions exist across the nodes. In such case, it turns out that there may be no output transaction order that satisfies order-fairness. This motivates relaxations of order-fairness that enable protocols to circumvent the impossibility and realize them in a distributed setting.

There are two principal approaches in relaxing fairness. The first one relies on a concept of time that can apply across all participants. In approximate order fairness [25], fair ordering in the output applies only to appropriately "spaced apart" transactions across all nodes. In timed-relative fairness, cf. [27,37], if a transaction $\texttt{tx}$ is received by *all* honest parties prior to $\texttt{tx}'$, then $\texttt{tx}$ must be sequenced before $\texttt{tx}'$. There are two obvious disadvantages of this approach to fairness: first, it requires reference to some shared notion of time. Second, it gives up on a lot of transactions whose propagation patterns somewhat overlap; in many applications however (e.g., front running mitigation) it is exactly for such transactions that fair ordering is needed.

The second principal approach, block order fairness [25], gives up on assigning a unique sequence number to all transactions in the output ledger. Transactions can be batched together in the same "block" in which case the system can be said to refrain from actually ordering them. Block order fairness has the advantage that it can be defined without referring to any shared notion of time and thus it can apply to transactions that are submitted concurrently and even apply to asynchronous execution environments. However, given that many distributed ledger applications require a total ordering of the output, it leaves open the question how far apart transactions may end up when finally serialized.

Motivated by the above, we set out to investigate the natural objective of minimizing the number of unfairly ordered transactions preceding for any transaction in the serialized output of a distributed ledger. This objective, similar to block order fairness, needs no shared notion of time to be meaningful, but it also translates to an eminently practical guarantee in the SMR setting that block order fairness lacks: the system will strive to minimize the number of (inevitable) unfair state updates that happen before the processing of any transaction.

To illustrate the importance of bounded unfairness with an example, consider a simple automated market maker (AMM) where it maintains a constant product $XY = C$ for swapping two tokens X and Y (Y is the native token) and one transaction can only sell a fixed number of tokens. When the market loses confidence in X, every stakeholder would like to sell X to minimize her loss. Consider an AMM state (X_0, Y_0) (i.e., $X_0 Y_0 = C$) and a number k of stakeholders $s_0, s_1, \dots, s_{k-1}$, each one selling an amount of $\Delta = \delta X_0$ tokens at times $t_0 < t_1 < \cdots < t_{k-1}$, respectively. At time t_i, the AMM state has slipped to the point $(X_0 + i\Delta, C/(X_0 + i\Delta)) = ((1+i\delta)X_0, Y_0/(1+i\delta))$ and the exchange rate for stakeholder i should be $(Y_0/X_0)/[(1+i\delta)(1+(i+1)\delta)]$. In particular, an unfair ordering $(s_1, s_2, \dots, s_{k-1}, s_0)$, will result in stakeholder s_0 transacting with exchange rate $(Y_0/X_0)/[(1+k\delta)(1+(k+1)\delta)]$ instead of $(Y_0/X_0)/(1+\delta)$. It follows the exchange rate becomes worse proportionally to $1/k^2$ when $k-1$ unfair state updates take place, thus keeping the number of unfair state updates at the minimum possible value (as bounded unfairness strives to do) is in the best interest of the users, protecting them from the effects of unfair slippage.

1.1 Our Results

We introduce a new class of (receiver) order fairness definitions – *bounded unfairness*. In SMR protocols all input transactions are eventually sequenced following an ordering σ which assigns a unique index to each transaction. Our definition is parameterized by a threshold φ and a bound B; each party that runs the protocol has an "input profile" which ranks all received transactions according to the order they were received. For two transactions $\texttt{tx}, \texttt{tx}'$, we say that $\texttt{tx} \prec^{\varphi} \texttt{tx}'$ if φ fraction of input profiles present $\texttt{tx}$ before $\texttt{tx}'$. Given any two such transactions, the output ordering σ should satisfy $\sigma(\texttt{tx}) - \sigma(\texttt{tx}') \le B$. I.e., $\texttt{tx}$ cannot be serialized more than B positions later compared to $\texttt{tx}'$.

Observe that (φ, B)-fairness matches standard order fairness for the case of $B = 0$ while for any choice $B > 0$ it relaxes it. The relaxation allows transactions $\texttt{tx} \prec^{\varphi} \texttt{tx}'$ to be "unfairly" sequenced as $(\texttt{tx}', \dots, \texttt{tx})$ with $\texttt{tx}'$ coming at most B positions earlier. Given the unrealizability of fairness for $B = 0$, the obvious question to ask here is for what values of B it is possible to realize the property and what is the smallest possible choice for it.

In order to minimize B, we allow it to be a function of the parties' input profiles and the given pair of transactions. The input profiles define a transaction dependency graph G which includes an edge $(\texttt{tx}, \texttt{tx}')$ if and only if $\texttt{tx} \prec^{\varphi} \texttt{tx}'$.

Given this, we observe that the problem of (φ, B) fairness relates to the concept of *graph bandwidth* over G, cf. [21]. The bandwidth problem asks for a vertex ordering $\sigma : V \to \mathbb{N}$ that minimizes the *maximum difference* $\sigma(u) - \sigma(v)$ across all edges $(u, v) \in E$. We call this the *directed bandwidth* as it aims at minimizing the length of the "backward edges" in G, i.e., those that violate fairness. We instantiate the bound B using the directed bandwidth of the strongly connected component (SCC) that contains the two transactions, or 0 if no such SCC exists.

Our first result regarding our new definition that bases fairness on directed bandwidth establishes that it is the best possible we can hope for in terms of minimizing unfair state updates preceding any transaction. Indeed, we prove that for any protocol that serializes the transactions there can be SCCs in the dependency graph within which two transactions *must* be ordered spaced apart by as many positions as the directed bandwidth of the SCC. Any function B that beats this bound is unrealizable!

Our second result regarding fairness based on directed bandwidth investigates the complexity of the problem. By adapting previous results for the bandwidth problem over undirected graphs, we prove that the directed bandwidth problem is NP-hard and non approximable for any constant ratio. Armed with this result, we prove that any protocol that realizes our order fairness property optimally also solves directed bandwidth, i.e., it solves an NP-hard problem. We also prove upper and lower bounds for the maximum directed bandwidth across all graphs; this result establishes the worst-case that is to be expected in terms of serializing the transactions with bounded unfairness. In particular we show that in the worst-case directed bandwidth equals $n - \Theta(\log n)$ where n is the number of transactions. Given the above, a natural question is how previous relaxations of order fairness fare w.r.t. bounding unfairness. We present explicit counterexamples illustrating how such previous definitions do not provide good bounds.

We then turn to investigate the inherent tension between liveness and our fairness definition. Similar to block order fairness, we first prove that it is impossible to satisfy liveness and directed-bandwidth order-fairness when the transaction delivery mechanism is asynchronous. Intuitively, the reason is that Condorcet cycles may extend indefinitely in a manner which is impossible to accommodate outputting any transaction in the cycle without breaking fairness. Given this impossibility one has to either settle for weak-liveness (transactions are included *eventually* [25]) or restrict fairness a bit more. Towards this latter target we consider a bounded delay transaction dissemination environment where each transaction is disseminated within a window of time Δ_{tx}. In this setting our core observation is that Condorcet cycles spanning a long period of time can be partitioned across the time domain in such a way that a bound on directed bandwidth of the graph can be derived. In such graphs we prove that directed bandwidth is bounded by at most 3 times the maximum number of transactions disseminated concurrently within a Δ_{tx} time window. This gives rise to a relaxed definition of our fairness notion that we call *timed directed bandwidth*.

The astute reader so far would have observed that we introduced our concept in a setting where participants are static — the relation $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ which gives rise to the transaction dependency graph is based on numbers of parties who witness a particular order between the two transactions. In a permissionless environment however, e.g., such as that of Bitcoin, participants may engage with a protocol in a transient manner hence making $\prec^{\varphi}$ ill defined. We address this issue by recasting the relation in the permissionless setting as follows: $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ means that a φ fraction of hashing power "stands behind" a particular ordering between two transactions for a minimum period of time which is specified by a security parameter.

Armed with the above definitional framework, we focus on realizing directed bandwidth fairness. We put forth Taxis, a permissionless protocol that operates in the same setting as Bitcoin. We present two variants. In $\mathsf{Taxis}_{\mathsf{WL}}$, miners continuously submit suffixes of their transaction input profile packaged within proofs-of-work using the 2-for-1 PoW technique of [20] that are included provided they are sufficiently recent using the recency condition of [33]. In this fashion it is possible to continuously compute and expand the transaction dependency graph G on-chain for the settled set of transactions. The ledger is then created by identifying SCCs of G and calculating directed bandwidth.

Our second variation of the Taxis protocol breaks long cycles when they occur and uses the median of timestamps to determine the transaction ordering within large SCCs. This enables us to achieve liveness and *timed directed bandwidth fairness*, the relaxation of our directed bandwidth fairness definition that relaxes fairness for particularly long Condorcet cycles.

We present a full analysis of our protocols in a permissionless dynamic participation setting using the analytical toolset from [17,20]. Notably, we enhance the "typical execution" concept by lower-bounding the difficulty that φ fraction of honest parties can acquire. This lower bound makes it possible for us to show that, for a specific transaction $\mathtt{tx}$, φ fraction of honest parties can accumulate more difficulty than others (including the adversary and the rest of honest parties) during K consecutive rounds, which guarantees that parties will agree on (i) the transactions that precede $\mathtt{tx}$; and (ii) a timestamp associated with $\mathtt{tx}$. Combining these properties with our dependency graph construction rules, we conclude consistency, liveness (for Taxis) and order-fairness according to the description above.

Regarding performance, we note that $\mathsf{Taxis}_{\mathsf{WL}}$ runs exponentially on the number of edges of the subgraph of the transaction graph that is defined by the (largest) Condorcet cycle. Recall that given the hardness and non-approximability of directed bandwidth we cannot expect a polynomial-time algorithm; furthermore, in practice, Condorcet cycles may be quite small or even not occurring at all (in non-adversarial settings), see [24], hence for practical purposes exponential dependency on their corresponding subgraph length may not be prohibitive. Furthermore, for Taxis, assuming a Δ_{tx} bound on transaction dissemination, we improve the complexity to be bounded by an exponential on the size of the largest Condorcet cycle (for constant throughput environments).

We conclude with a discussion on alternative ways to relaxing order-fairness and open questions regarding the structure of transaction dependency graphs. While our concept of directed bandwidth achieves an optimal ordering of transactions in terms of bounding unfairness, there can be orthogonal considerations that highlight the multi dimensionality of the fairness problem. We also discuss issues related to dynamic participation and how our results can also translate to the permissioned setting.

As a final contribution we would like to highlight how our work can have applications to social choice theory. Typically, in social choice, the input profiles of parties (e.g., rankings of the candidates) are assumed to be finite sequences. Given such rankings, it is sought to produce an agreeable ordering with good properties. In such case, fairness captures the natural property that if candidate A is preferable by a majority of participants compared to another candidate B, then A should be ranked higher in the final ranking. In the social choice context, our result can be seen as a way to answer the social choice problem when participants have an ever evolving sequence of preferences and it is desired to combine their preferences while minimizing the violations of their preferences as much as possible. For instance, consider an infinite sequence of news items, and a dynamic population of agent-editors with distinct preferences for each one, in terms of e.g., how interesting each one is. The task is to produce a single output news feed that respects the preferences of the agent-editors as much as possible. Our results readily translate to this setting enabling the agent-editors to produce a unified news feed with the minimum possible misplacement between news items: specifically if φ fraction of agents deems item n as more interesting than item n′, then n′ will be placed at most B positions before n in the unified news feed.

Organization of This Paper. The rest of the paper is organized as follows. In Sect. 2, we present preliminaries on the protocol exeuction model, transaction profiles and dependency graphs. We formally define and analyze transaction order fairness in Sect. 3. In Sect. 4, we present our fair-order protocol Taxis. We discuss some future research directions that might be of independent interest in Sect. 5. A survey of related works [2,5,6,9,23–25,27,28,36,37], and a detailed description of preliminaries, protocols and proofs can be found in the full version of this paper [26].

2 Preliminaries

In this section we first briefly describe our protocol execution model, transaction diffusion mechanism and the dynamic environment. Refer to [26] for more details. We then introduce transaction profiles and dependency graphs in Sect. 2.2. They are notations crucial to the discussion of order fairness.

2.1 Protocol Execution Model

Our model follows Canetti's formulation of "real world" notion of protocol execution [7,8] for multi-party protocols. To specify the "resources" that may be available to the instances running protocol—e.g., the diffuse channel—we will follow the approach of describing them as ideal functionalities in the terminology of [8].[1]

The protocol execution proceeds in "rounds". Parties are always aware of the current round (i.e., synchronous processors); this is captured by a global clock $\mathcal{G}_{\text{CLOCK}}$ [22]. Inputs are provided by an environment program $\mathcal{Z}$ to parties that execute the protocol Π. The adversary $\mathcal{A}$ is a single entity that takes control of corrupted parties, and is both "adaptive" (i.e., $\mathcal{A}$ can take control of parties on the fly) and "rushing" ($\mathcal{A}$ is allowed to observe honest parties' actions before deciding her reaction). The hash function $H(\cdot)$ is modeled as a random oracle $\mathcal{F}_{\mathsf{RO}}$ and abstracts parties attempting to solve "proof-of-work" [14]. Following the convention that different types of messages are diffused by their own network, we consider two diffusion functionalities — one for general messages ($\mathcal{F}_{\mathsf{Diffuse}}$) and another for transactions (see below). For all messages except transactions, the communication is bounded-delay (a.k.a., "partial synchronous" [13,32]). I.e., there is an upper bound Δ (measured in number of rounds) and the adversary may delay the delivery of messages for up to Δ rounds.

Transaction Diffusion Model. The environment program $\mathcal{Z}$ is responsible for generating new transactions and handing them to the diffusion functionality. We consider two types of transaction diffusion functionality — $\mathcal{F}_{\mathsf{Diffuse}}^{\mathsf{tx,async}}$ and $\mathcal{F}_{\mathsf{Diffuse}}^{\mathsf{tx},\Delta_{\mathrm{tx}}}$. The first functionality captures the asynchronous transaction diffusion, i.e., in $\mathcal{F}_{\mathsf{Diffuse}}^{\mathsf{tx,async}}$ the adversary can deliver a transaction `tx` at any time (after `tx` is generated by $\mathcal{Z}$). The only restriction is that $\mathcal{F}_{\mathsf{Diffuse}}^{\mathsf{tx,async}}$ should send all transactions to all parties eventually. The second one captures a Δ_{tx}-disseminated transaction diffusion network. Specifically, in $\mathcal{F}_{\mathsf{Diffuse}}^{\mathsf{tx},\Delta_{\mathrm{tx}}}$ the adversary is forced to deliver a transaction to all the honest parties within Δ_{tx} rounds after it is learnt by at least one honest participant.

Considering the physical limits on transaction throughput, we assume that the total number of transactions will be a polynomial function of the running time of protocol execution.

Dynamic Participation. In order to describe the protocol execution in a more realistic fashion, following the treatment in [3], we classify protocol participants into different types. Especially, *alert* parties—the core set of parties to carry out the protocol—are those who have access to all the resources (random oracle, network, clock) and are synchronized with each other. We also put some restriction on the environment's power to fluctuate the number of *alert* parties [17,20]; i.e., for any window of fixed length, the increase/decrease on the number of alert parties is bounded (see [26] for more details).

State Machine Replication. State machine replication [34] is a problem that asks a set of parties accepting input logs to maintain a public data structure

[1] Note that these notions are used for model description only, our security proof is property-based.

that serializes the logs. This public data structure is called *ledger* in the context of ledger consensus (cf. [18,19]). Conventionally, a public ledger should satisfy two properties (we adopt $\mathcal{L}$ as the settled part of the ledger in party's view, and $\widetilde{\mathcal{L}}$ the whole ledger held by the party).

- **Consistency**: For any two honest parties $\mathsf{P}_1, \mathsf{P}_2$ reporting $\mathcal{L}_1, \mathcal{L}_2$ at rounds $\mathtt{r}_1 \leq \mathtt{r}_2$, respectively, it holds that $\mathcal{L}_1$ is a prefix of $\widetilde{\mathcal{L}}_2$.
- **Liveness**: (parameterized by $u \in \mathbb{N}$, the "wait time" parameter): If a transaction $\mathtt{tx}$ is provided to all honest parties for u consecutive rounds, then it holds that for any player P, $\mathtt{tx}$ will be in $\mathcal{L}$.

2.2 Transaction Profiles and Dependency Graphs

Let $\mathbb{T}$ denote the (finite) set of all possible transactions with elements $\mathtt{tx}$. A *transaction profile* (or "profile" for short) is a bijection $\mathsf{R} : \mathbb{T} \to [m]$ where $m = |\mathbb{T}|$. For each (honest) party P_i, its receiving transaction log forms a profile which is denoted by R_i. Consider a set of n parties $\mathcal{P}$, we write $\mathcal{R} = \langle \mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_n \rangle$ as the list of all transaction profiles. Regarding order fairness, we are interested in a serialization function F that takes an indefinte number of transaction profiles $\mathcal{R}$ as input and outputs a new profile denoted by σ, namely $\sigma = F(\mathcal{R})$.

We adopt "$\prec$" to describe the "order before" relation on $\mathbb{T} \times \mathbb{T}$. Note that this relation is (i) irreflexive (not $\mathtt{tx} \prec \mathtt{tx}$); (ii) asymmetric ($\mathtt{tx} \prec \mathtt{tx}'$ implies not $\mathtt{tx}' \prec \mathtt{tx}$) and (iii) transitive (i.e., $\mathtt{tx} \prec \mathtt{tx}'$ and $\mathtt{tx}' \prec \mathtt{tx}''$ implies $\mathtt{tx} \prec \mathtt{tx}''$). We write $\mathtt{tx} \prec_i \mathtt{tx}'$ if $\mathsf{R}_i(\mathtt{tx}) < \mathsf{R}_i(\mathtt{tx}')$; i.e. $\mathtt{tx} \prec \mathtt{tx}'$ in party P_i's profile (in other words, P_i receives transaction $\mathtt{tx}$ before $\mathtt{tx}'$). For every pair of distinct transactions $\mathtt{tx}, \mathtt{tx}'$ in $\mathbb{T}$, they are ascribed either the relations $\mathtt{tx} \prec_i \mathtt{tx}'$ or $\mathtt{tx}' \prec_i \mathtt{tx}$ in profile R_i.

In order to achieve order fairness, we are interested in the pairs of transactions such that one is received by sufficiently many parties before the other. To measure what "sufficiently many" means, we adopt $\varphi \in \mathbb{R}^+$ as the order fairness parameter. We say $\mathtt{tx} \prec^{\varphi}_{\mathcal{R}} \mathtt{tx}'$ if, for profiles $\mathcal{R}$, $\mathtt{tx} \prec \mathtt{tx}'$ holds in at least φ fraction of these profiles (when the profile set is explicit in the context we drop the subscript and simply write $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$). Note that when $\varphi \leq 1/2$, it results in a logical contradiction as both $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ and $\mathtt{tx}' \prec^{\varphi} \mathtt{tx}$ hold. Hence, we only care about φ such that $1/2 < \varphi \leq 1$.

Transaction Timestamp Assignment. We next present a timestamp assignment function F_{ts} which is useful in the context of state machine replication problem. Note that different from the one-shot consensus where input profiles are given to parties as input in an instant, the transaction log that a party receives grows with time. Hence, we assign each transaction a timestamp to indicate when it is received. I.e., parties store transactions in pair $\langle \mathtt{tx}, \mathtt{t} \rangle$ where $\mathtt{t} \in \mathbb{N}^+$ is the time that they receive $\mathtt{tx}$. We denote the timestamp of $\mathtt{tx}$ in profile R as $\mathsf{TS}(\mathtt{tx}, \mathsf{R})$ and the list of all timestamps associated with

$\texttt{tx}$ in $\mathcal{R}$ as $\mathsf{TS}(\texttt{tx})$. We call the profiles $\mathcal{R}$ $\varDelta_{\text{tx}}$-disseminated if for all transactions, the timestamps associated with them are within a $\varDelta_{\text{tx}}$ time window (i.e. $\forall \texttt{tx} \in \mathbb{T}, \max \mathsf{TS}(\texttt{tx}) - \min \mathsf{TS}(\texttt{tx}) \leq \varDelta_{\text{tx}}$).

Consider an assignment of a timestamp to each transaction, which can be represented by a function $F_{\mathsf{ts}} : \mathbb{T} \to \mathbb{N}^+$. If for profiles $\mathcal{R}$ it holds that $\forall \texttt{tx} \in \mathbb{T}, F_{\mathsf{ts}}(\texttt{tx}) \in \mathsf{TS}(\texttt{tx})$, then we say F_{ts} is compliant with $\mathcal{R}$. Let $\mathbb{F}_{\mathsf{ts},\mathcal{R}}$ denote the set of all compliant F_{ts} with $\mathcal{R}$. Especially, we are interested in the mapping from each transaction to its earliest receiving time; and we denote this mapping by $F_{\mathsf{ts}}^{\mathsf{min}}$ (i.e., $\forall \texttt{tx} \in \mathbb{T}, F_{\mathsf{ts}}^{\mathsf{min}}(\texttt{tx}) = \min \mathsf{TS}(\texttt{tx})$).

Transaction Dependency Graphs. Consider a list of transaction profiles $\mathcal{R} = \langle \mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_n \rangle$. An $(\mathcal{R}, \varphi)$-dependency-graph is a directed graph $G_{\mathcal{R},\varphi}$ constructed as follows. For each transaction $\texttt{tx}_i$, add a vertex v_i to $G_{\mathcal{R},\varphi}$; then, for any pair of vertices $\texttt{tx}_i, \texttt{tx}_j$, add an edge (v_i, v_j) if $\texttt{tx}_i \prec^{\varphi} \texttt{tx}_j$. When $\prec^{\varphi}$ is the majority relation (i.e., $\varphi = 1/2 + 1/m$, where m is the total number of transactions), we write $G_{\mathcal{R}}$ and call the graph $\mathcal{R}$-dependency. Note that when $\varphi > 1/2$, at most one of (i, j) and (j, i) can be added — i.e., a dependency graph is oriented.

Graph Notations. A vertex ordering of a graph $G = (V, E)$ is a bijection $\sigma : V \to [n]$ where $n = |V|$. A null graph is the unique graph having no vertices. A subgraph S of G is another graph such that $V(S) \subseteq V(G) \wedge E(S) \subseteq E(G)$ ($V(S)$ must include all endpoints of the edges in $E(S)$). Conversely, a supergraph H of G is a graph formed by adding vertices, edges, or both to G. A spanning supergraph is a supergraph by merely adding edges to the original graph.

A directed graph is strongly connected if every vertex is reachable from every other vertex. The strongly connected components are maximal subgraphs of a directed graph that are themselves strongly connected.

For a dependency graph G, we slightly abuse the notation and use transaction $\texttt{tx}$ and its generated vertex v interchangeably. For instance, $(\texttt{tx}, \texttt{tx}')$ denotes the edge from vertex v generated by $\texttt{tx}$ to vertex v' generated by $\texttt{tx}'$. And $\sigma(\texttt{tx})$ is the same as $\sigma(v)$ where v is generated by $\texttt{tx}$.

3 Order Fairness

In this section we first give a definition of order fairness in the sense of bounded unfairness. We then connect order fairness to DirectedBandwidth and provide a fine-grained fair-order definition which is the best that one can expect in this setting. Next, we study this problem in the context of state machine replication and permissionless participation respectively. Due to the space limit, all proofs in this section are presented in the full version of this paper [26].

3.1 Bounded Unfairness and Serialization

An ideal fair order σ on profiles $\mathcal{R}$ follows all φ-preferences in $\mathcal{R}$. I.e., for all $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ it holds $\sigma(\mathtt{tx}) < \sigma(\mathtt{tx}')$. Unfortunately, this is impossible with the existence of Condorcet cycles — φ-preferences can be cyclic and hence no σ can satisfy all of them simultaneously. To see the simplest example, fix $\varphi = 2/3$ and consider three transactions $\mathtt{tx}_1, \mathtt{tx}_2, \mathtt{tx}_3$ and three profiles $\mathsf{R}_1 = \mathtt{tx}_1 \prec \mathtt{tx}_2 \prec \mathtt{tx}_3$, $\mathsf{R}_2 = \mathtt{tx}_2 \prec \mathtt{tx}_3 \prec \mathtt{tx}_1$ and $\mathsf{R}_3 = \mathtt{tx}_3 \prec \mathtt{tx}_1 \prec \mathtt{tx}_2$. We have $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$, $\mathtt{tx}_2 \prec^{\varphi} \mathtt{tx}_3$ and $\mathtt{tx}_3 \prec^{\varphi} \mathtt{tx}_1$.

Note that there is a hidden constant in $\sigma(\mathtt{tx}) < \sigma(\mathtt{tx}')$ (i.e., $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') < 0$) to indicate the position that $\mathtt{tx}'$ can be placed before $\mathtt{tx}$. A natural relaxation on standard order fairness would be to enlarge this distance to some realizable extent. I.e., an order is fair if for all preferences $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$, $\mathtt{tx}'$ is not ordered at a position that is too earlier compared with $\mathtt{tx}$. In order to acquire a fine-grained fairness notion, we are interested in upper-bounding this distance on every pair of transactions $\mathtt{tx}, \mathtt{tx}'$ in specific transaction profiles $\mathcal{R}$. Thus we define B as a function of $\mathcal{R}, \varphi, \mathtt{tx}$ and $\mathtt{tx}'$ and require $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') < B$. This gives us an intuitive and parametric definition of order fairness.

Definition 1 (**(φ, B)-fair-order**). *A profile σ is a (φ, B)-fair-order on $\mathcal{R}$ if for all $\mathtt{tx}, \mathtt{tx}'$ such that $\mathtt{tx} \prec^{\varphi}_{\mathcal{R}} \mathtt{tx}'$, it holds that $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') \leq B$ where B is a function of $\mathcal{R}, \varphi, \mathtt{tx}$ and $\mathtt{tx}'$.*

(φ, B)-fair-order is unrealizable when B is a function such that there exist $\mathcal{R}$ and it holds that $\forall \sigma, \exists (\mathtt{tx}, \mathtt{tx}'), \sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') > B(\mathcal{R}, \varphi, \mathtt{tx}, \mathtt{tx}')$. In other words, B is too "small" on some profiles thus no ordering could order $\mathtt{tx}, \mathtt{tx}'$ "close enough" as specified by B. On the other hand, Definition 1 is trivial when B is a function such that $\exists (\mathcal{R}, \mathtt{tx}, \mathtt{tx}'), B(\mathcal{R}, \varphi, \mathtt{tx}, \mathtt{tx}') \geq m - 1$ where m is the total number of transactions in $\mathcal{R}$ (i.e., $\mathtt{tx}, \mathtt{tx}'$ can be arranged apart for an arbitrary distance). The reason such a B is called trivial is that, intuitively, given a set of profiles, any protocol that realizes an (unfair) order with $B = m - 1$ on one transaction pair $\mathtt{tx}, \mathtt{tx}'$ can be converted into a new protocol with fair order $B' < m - 1$ on every pairs, by simply swapping $\mathtt{tx}, \mathtt{tx}'$ in the output profile[2]; moreover, as we show in Theorem 5, there exists a *practical* B which requires distance strictly less than $m - 1$ (more precisely, $m - \log m/2$) for every pair of transactions.

Serialization with Adversarial Profiles. We then consider order fairness in the presence of an adversary. Given a protocol execution, the set of honest parties $\mathcal{H}$ is well-defined and we let $h = |\mathcal{H}|$ denote the number of honest parties. We abstract the sequence of transactions received by an honest party P_i as R_i, and write $\mathcal{R}^{\mathcal{H}} = \langle \mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_h \rangle$ as the honest profiles. Regarding corrupted parties, note that they can deviate arbitrarily from the protocol thus the profile

[2] Notice that $B = m - 1$ on a transaction pair $\mathtt{tx}, \mathtt{tx}'$ only when $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ and $\mathtt{tx}$ is put at the last but $\mathtt{tx}'$ is put at the first of the output profile. By swapping $\mathtt{tx}, \mathtt{tx}'$ we get a new order with largest unfair distance strictly smaller than $m - 1$.

abstraction does not apply to them. Instead, we model the adversarial manipulation as follows. Suppose F is a serialization function that takes an indefinite number of profiles as input and outputs a new profile, for every honest party we require that they output $\sigma = F(\mathcal{R})$ where $\mathcal{R} = \langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle$ and $\mathcal{R}^{\mathcal{A}}$ is some arbitrary profiles (this models the adversarial behavior). Note that for different honest parties, $\mathcal{R}^{\mathcal{A}}$ can be different to them. Thus, the following definition does not ask for agreement — i.e., honest parties could output different profiles as long as they are all fair orderings on $\mathcal{R}^{\mathcal{H}}$; it implies agreement only in the all honest setting.

Definition 2 (implementing a fair-order serialization). *Given a protocol execution, an (F, φ, B)-consistent serialization event happens if and only if for any honest party P_i, there exist profiles $\mathcal{R} = \langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle$ such that*

(i) $\mathcal{R}^{\mathcal{H}}$ is defined by the sequence of transactions received by honest parties;
(ii) P_i outputs $\sigma = F(\mathcal{R})$ and σ is a (φ, B)-fair-order on honest profiles $\mathcal{R}^{\mathcal{H}}$.

A protocol serializes transactions according to F with (φ, B)-order-fairness, if the (F, φ, B)-consistent serialization event happens with overwhelming probability.

Notice that, in order to implement a non-trivial fair-order serialization, the adversary should not be too powerful with respect to the fairness parameter φ. To model this we consider upper-bounding profiles in $\mathcal{R}^{\mathcal{A}}$ and we write t as its maximum number of profiles. Then we consider the threshold on t with respect to the number of honest parties h and fair-order parameter φ. For instance, dishonest majority $(t > h)$ is infeasible with any φ. This is because if the adversary could select more profiles than the honest, then $\mathcal{A}$ can completely dominate the φ-preferences. In other words, the adversary can vanish any $\texttt{tx} \prec^{\varphi} \texttt{tx}'$ by simply inserting profiles with the opposite order.

To see how adversarial power should be restricted in terms of the fair order parameter, we prove that when $t \geq (2\varphi - 1)h$, it becomes impossible to implement non-trivial fair-order serialization.

Theorem 1. *When $t \geq (2\varphi - 1)h$, no protocol implements non-trivial fair-order serialization.*

We say an adversary $\mathcal{A}$ is admissible with fairness parameter φ if it holds that $t < (2\varphi - 1)h$ and in Definition 2 the number of profiles in $\mathcal{R}^{\mathcal{A}}$ are upper-bounded by t. All following discussions on order fairness and transaction serialization are with respect to an admissible adversary.

3.2 Transaction Dependency Graphs

Fix φ and n transaction profiles $\mathcal{R}$, there will be a unique $(\mathcal{R}, \varphi)$-dependency-graph $G_{\mathcal{R},\varphi}$. When $\varphi < 1/2 + 1/n$ and n is odd (i.e., the majority preference), the dependency graph will be a tournament (since all pairwise preference can be extracted). As φ increases, the graph becomes more and more sparse. While the specific edges to be removed are subject to the profiles, we show that the structure of dependency graphs depends on the fairness parameter φ, and a large φ implies graphs without cycles of small size. For instance, when $\varphi > 2/3$, no directed triangle can exist in the dependency graph; when $\varphi > 3/4$, no directed square can exist; etc. We formalize this property in Theorem 2.

Theorem 2. *For any $\varphi > 1/2$ and any profiles $\mathcal{R}$, the $(\mathcal{R}, \varphi)$-dependency-graph $G_{\mathcal{R},\varphi}$ does not contain cycles of size k for all $k < \lceil 1/(1-\varphi) \rceil$.*

Conversely, given an oriented graph G, there exist some profiles whose dependency graph is exactly G. McGarvey [29] provides an approach to construct these profiles (with majority preference). We briefly describe McGarvey's approach here. Suppose we would like to construct a profile set $\mathcal{R}$ from an oriented graph G with m vertices. For each edge $(v_i, v_j) \in G$, add two profiles $\mathsf{R}_1, \mathsf{R}_2$ to $\mathcal{R}$ with $\mathsf{R}_1(\texttt{tx}_i) = 1, \mathsf{R}_1(\texttt{tx}_j) = 2, \mathsf{R}_2(\texttt{tx}_i) = m-1, \mathsf{R}_2(\texttt{tx}_j) = m$ and $\mathsf{R}_1(\texttt{tx}_k) + \mathsf{R}_2(\texttt{tx}_k) = m+1$ for all $k \neq i, j$ — i.e., $\texttt{tx}_i, \texttt{tx}_j$ are put at the head and rear of the profile respectively and the rest are in an exactly reversed order. Notice for all edge (v_i, v_j), $\texttt{tx}_i, \texttt{tx}_j$ are in the same order only in the two profiles constructed from them.

Dependency Graph with Adversarial Profiles. Given a protocol execution, the $(\mathcal{R}^{\mathcal{H}}, \varphi)$-dependency-graph G is unique and well-defined. We are interested in the relationship between G and dependency graphs that are constructed with adversarial profiles.

Note that parties cannot distinguish which profile is corrupted, thus for $\mathcal{R} = \langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle$, it is infeasible to consider the dependency graph based on preferences held by φ fraction of profiles. For instance, when φh honest parties believe $\texttt{tx} \prec \texttt{tx}'$, the adversary can collude with the minority and vanish this preference in $\mathcal{R}$; similarly, when $(\varphi h - 1)$ honest parties receive $\texttt{tx} \prec \texttt{tx}'$, the adversary can join forces with them and make this preference account for φ fraction in $\mathcal{R}$.

Fix honest profiles $\mathcal{R}^{\mathcal{H}}$, we show that for any admissible adversary $\mathcal{A}$ and any adversarial profiles $\mathcal{R}^{\mathcal{A}}$ selected by $\mathcal{A}$, it yields a dependency graph G' on $\langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle$ with majority preference such that all edges in G remain the same orientation in G'.

Theorem 3. *Fix φ and honest profiles $\mathcal{R}^{\mathcal{H}}$ and denote the $(\mathcal{R}^{\mathcal{H}}, \varphi)$-dependency-graph by G. For any graph $G' \in \{G_{\mathcal{R}} \mid \mathcal{R} = \langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle$ and $\mathcal{R}^{\mathcal{A}}$ is chosen by an admissible adversary$\}$, it holds that G' is a spanning supergraph of G.*

Theorem 3 shows, with admissible adversarial manipulation, the φ-preferences are "robust" among all dependency graphs. We write the set of all possible dependency graphs on $\mathcal{R} = \langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle$ from majority preference as $\mathbb{G}_{\mathcal{R}^{\mathcal{H}},\varphi}$. Note that when given $\mathcal{R}^{\mathcal{H}}$ and φ, the set of all possible $\mathcal{R}^{\mathcal{A}}$ is well-defined with an admissible adversary by Theorem 1.

3.3 Bounded Unfairness from Directed Bandwidth

Given honest transaction profiles $\mathcal{R}$ (with Condorcet cycles), our goal is to find an ordering that does not put $\texttt{tx}'$ too early before $\texttt{tx}$ when $\texttt{tx} \prec^{\varphi} \texttt{tx}'$. Consider a dependency graph $G \in \mathbb{G}_{\mathcal{R},\varphi}$ and a vertex ordering σ on G. Theorem 3 implies that G contains cycles (as all edges forming the cycles in $G_{\mathcal{R},\varphi}$ preserve in G), i.e., there will be back edges $(\mathtt{tx}, \mathtt{tx}')$ such that $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ and $\sigma(\mathtt{tx}) > \sigma(\mathtt{tx}')$. The length of a back edge $(\mathtt{tx}, \mathtt{tx}')$ is the distance of its source and target in the ordering $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}')$. Ideally, a fair order comes with back edges of small lengths. In order to quantify how small the length of a back edge can be, we are interested in finding a vertex ordering on the dependency graph G that minimizes the maximum length of back edges. Following the similar treatment in [21] (where they consider the forward edges), we state this problem as DirectedBandwidth in Definition 3.

Definition 3 (Directed Bandwidth). *Given a directed graph $G = (V, E)$,* DirectedBandwidth *asks to find a vertex ordering σ^* such that $\mathtt{DBW}(\sigma^*, G) = \min_{\sigma} \mathtt{DBW}(\sigma, G)$ where*

$$\mathtt{DBW}(\sigma, G) = \max_{\substack{(u,v) \in E, \\ \sigma(u) > \sigma(v)}} \sigma(u) - \sigma(v).$$

The directed bandwidth of a graph G is $\mathtt{DBW}(G) = \mathtt{DBW}(\sigma^*, G)$.

Note that when G is acyclic, there exist σ which is a topological ordering on G such that no back edge exists; this has little to do with the fair-order serialization problem and $\mathtt{DBW}(G) = 0$ for an acyclic graph. We also note that $\mathtt{DBW}(G) = 0$ if G is the null graph.

Analogous to Definition 3, Bandwidth [10,11,16] is a well-known and extensively studied graph problem aiming at minimizing the quantity $\mathtt{BW}(G, \sigma) = \max_{(u,v) \in E} |\sigma(u) - \sigma(v)|$ among all vertex orderings on an undirected graph. Bandwidth has been proved to be both NP-hard [31] and NP-hard to approximate within any constant ratio [12] over general graphs. Further, Bandwidth remains NP-hard and NP-hard to approximate even on very restricted graphs like caterpillars of hair length at most 3 (a restricted tree).

Since an undirected graph can be converted to a digraph by replacing each edge with two symmetric directed edges, there is a simple reduction from Bandwidth to DirectedBandwidth and thus DirectedBandwidth is also NP-hard and NP-hard to approximate over general graphs. Notice that, in our context, dependency graphs are all oriented graphs. We prove that DirectedBandwidth remains NP-hard and NP-hard to approximate within any constant ratio over oriented graphs.

Theorem 4. DirectedBandwidth *is NP-hard and NP-hard to approximate within any constant ratio over oriented graphs.*

DirectedBandwidth can be solved trivially in factorial time ($\mathcal{O}^*(n!)$) by an exhaustive search on all possible orderings; and, unlike some vertex ordering problems that can be solved by dynamic programming or divide-and-conquer, so far there is no evidence that these techniques also applies on DirectedBandwidth. A recent work by Jain *et al.* [21] provides exponential algorithms to find the exact and approximate solutions to DirectedBandwidth. Specifically, the exact algorithm runs in $\mathcal{O}^*(3^{|V|} \cdot 2^{|E|})$ time; and in order to get an ordering with bandwidth at most $(1+\epsilon)$ times the optimal one, an approximation algorithm runs in $\mathcal{O}^*(4^{|V|} \cdot (4/\epsilon)^{|V|})$ time. We briefly describe the exact algorithm for DirectedBandwidth in [26].

Largest Possible Directed Bandwidth. Since all oriented graphs can be generated by profiles, we are interested in the largest possible bandwidth on graphs with a fixed number of vertices.

Note that, given n vertices, the worst bandwidth $n-1$ can always be avoided by finding an edge (i,j) and outputting σ such that $\sigma(i)=1$ and $\sigma(j)=n$. And, for a small constant k, we can check if a graph has bandwidth $n-k$ by checking $\mathcal{O}(n^{2k})$ vertex orderings — i.e., we select k vertices each at the head and rear of orderings and see if a back edge exists between the two sets. Unfortunately, the time complexity of this simple approach grows to factorial when $k=\Theta(n)$ hence it becomes impractical for large graphs. This raises the question whether it is possible to find a vertex ordering with directed bandwidth, e.g., $0.99n$, for any oriented graph with n vertices.

Here we give a negative answer to this question. We prove that, among all oriented graphs with n vertices there exist some tournaments with large directed bandwidth compared with n[3]. In Theorem 5 we show that the above simple approach to check bandwidth will soon terminate on some graphs by considering Zarankiewicz's problem.

Theorem 5. *Let $\mathbb{G}_n$ denote the set of all oriented graphs with n vertices. It holds that*

$$n - 4\log n < \max_{G\in\mathbb{G}_n} \mathtt{DBW}(G) < n - \log n/2.$$

$(\varphi, \mathtt{DBW})$-fair-order. After extracting the directed bandwidth of a graph in Definition 3, we are now ready to define fair order based on upper-bounding how much $\mathtt{tx}'$ can be ordered before $\mathtt{tx}$ when $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$.

Note that, given a transaction profile set $\mathcal{R}$ and its dependency graph G, we cannot simply define the upper bound as $\mathtt{DBW}(G)$. This is because G might contain several strongly connected components and their sizes might differ a lot. Actually, the bandwidth of a graph G is the maximum bandwidth among all strongly connected components in G.

$$\mathtt{DBW}(G) = \max\{\mathtt{DBW}(G') : G' \text{ is a strongly connected component of } G\}.$$

Suppose there is a SCC that contains thousands of transactions and $\mathtt{DBW}(G)$ is also in the thousands. Then, for other relatively small SCCs with, for instance, 10 transactions, an upper bound as $\mathtt{DBW}(G)$ does not set any limitation on how they should be ordered.

[3] This result implies that no algorithm can guarantee finding a vertex ordering of directed bandwidth $0.99n$.

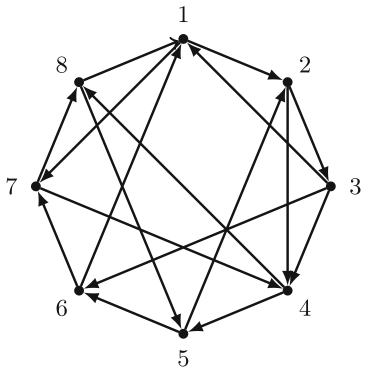
Additionally, note that when given $\mathcal{R}^{\mathcal{H}}$ and φ, a fair-order serialization should consider all possible dependency graphs $\mathbb{G}_{\mathcal{R}^{\mathcal{H}},\varphi}$ with admissible $\mathcal{R}^{\mathcal{A}}$. Theorem 3 shows that $\mathcal{A}$ may create new cycles or enlarge existing ones, but $\mathcal{A}$ cannot remove any edge that has already been there in $G_{\mathcal{R}^{\mathcal{H}},\varphi}$. Due to the above observations, we propose a fine-grained definition of order fairness (Definition 4) on top of Definition 1 by replacing the initial function with largest $\texttt{DBW}$ on all possible SCCs. Specifically, for a pair of transaction $\texttt{tx} \prec^{\varphi} \texttt{tx}'$, if among all possible dependency graphs $\mathbb{G}_{\mathcal{R}^{\mathcal{H}},\varphi}$ there is no graph with SCC that contains $\texttt{tx}, \texttt{tx}'$ simultaneously then the final output should follow $\texttt{tx} \prec \texttt{tx}'$. Otherwise, we will define the upper bound on their distance in the output by extracting all SCCs containing $\texttt{tx}, \texttt{tx}'$ over $\mathbb{G}_{\mathcal{R}^{\mathcal{H}},\varphi}$ and find the largest possible bandwidth.

Definition 4 (($\varphi, \texttt{DBW}$)-fair-order). *A profile σ is a $(\varphi, \texttt{DBW})$-fair-order on $\mathcal{R}$ if for all $\texttt{tx}, \texttt{tx}'$ such that $\texttt{tx} \prec^{\varphi}_{\mathcal{R}} \texttt{tx}'$, it holds that*

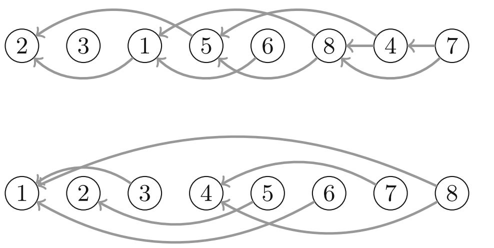
$$\sigma(\texttt{tx}) - \sigma(\texttt{tx}') \le \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \texttt{DBW}\big(\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}')\big),$$

where $\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}')$ is a function that outputs an SCC in G that contains both $\texttt{tx}, \texttt{tx}'$ if it exists, and a null graph otherwise.

Note that in an all honest setting, no $\mathcal{R}^{\mathcal{A}}$ exists, thus Definition 4 can be simplified as "$\texttt{tx} \prec^{\varphi} \texttt{tx}' \implies \sigma(\texttt{tx}) - \sigma(\texttt{tx}') \le \texttt{DBW}(\mathsf{SCC}(G_{\mathcal{R},\varphi}, \texttt{tx}, \texttt{tx}'))$". See below for an example where we have 8 transactions in $\mathcal{R}$ and the $(\mathcal{R}, \varphi)$-dependency-graph is illustrated in Fig. 1(a). Since $\texttt{DBW}(G_{\mathcal{R},\varphi}) = 3$, a $(\varphi, \texttt{DBW})$-fair-order on $\mathcal{R}$ should satisfy $\texttt{tx} \prec^{\varphi} \texttt{tx}' \implies \sigma(\texttt{tx}) - \sigma(\texttt{tx}') \le 3$. We provide a profile $\sigma = \texttt{tx}_2 \prec \texttt{tx}_3 \prec \texttt{tx}_1 \prec \texttt{tx}_5 \prec \texttt{tx}_6 \prec \texttt{tx}_8 \prec \texttt{tx}_4 \prec \texttt{tx}_7$ which is a fair order on $\mathcal{R}$ in Fig. 1(b). Note that only back edges are illustrated and the back edges $(5, 2)$, $(4, 5)$ and $(8, 1)$ are of maximum length 3. Also compare with the



(a) dependency graph $G_{\mathcal{R},\varphi}$



(b) $(\varphi, \texttt{DBW})$-fair-order (above) and lexicographic order (below)

Fig. 1. *Illustration of a dependency graph, a $(\varphi, \texttt{DBW})$-fair-order and a lexicographic order on $\mathcal{R}$. Only back edges are illustrated in (b).*

lexicographic order which has a back edge of length 7 (Aequitas and Themis may output this order, see below for comparison with existing protocols).

We highlight that Definition 4 is the most precise definition that we can make on top of Definition 1 and 2. For any new definition that tries to further reduce $\max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \mathtt{DBW}(\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}'))$ for transactions $\mathtt{tx}, \mathtt{tx}'$, there will exist some profiles $\mathcal{R}^{\mathcal{A}}$ leading to SCCs with large bandwidth which can invalidate the new definition. Refer to Sect. 5 for further discussions.

Theorem 6. *Suppose that a protocol implements (φ, B)-fairness for a function B. Then for all $\mathcal{R}$ there are $\mathtt{tx}, \mathtt{tx}'$ with $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$, such that B satisfies $B(\mathcal{R}, \varphi, \mathtt{tx}, \mathtt{tx}') \geq \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \mathtt{DBW}(\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}'))$.*

Comparison with Existing Protocols. We show that Aequitas [25], Themis [24], pompe [37] and wendy [27] fail to implement $(\varphi, \mathtt{DBW})$-fair-order serialization (Definition 2 and 4) even in the all honest setting. For Aequitas, the core observation here is that when an alphabetical order is adopted to order transactions within a Condorcet cycle, it is always feasible to simply manipulate the labels of transactions and produce any desired order. Next, Themis improves the transaction linearization in a Condorcet cycle to a Hamiltonian-cycle-based order. We point out that this treatment will always produce an order such that $\mathtt{tx}, \mathtt{tx}'$ are at the head and rear respectively but it holds $\mathtt{tx}' \prec^{\varphi} \mathtt{tx}$. Regarding pompe and wendy, note that in order to be resistant to possible adversarial manipulation, transactions are ordered by their median timestamp. Thus, we could get any desired output by constructing profiles with carefully selected timestamps.

The following two examples show how these protocols fail our fair-order serialization definition. In both examples we consider a Condorcet cycle of m transactions and denote its dependency graph as G.

Example 1 (Aequitas and Themis). Suppose $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$, we assign labels to transactions such that $label(\mathtt{tx}_2) < label(\mathtt{tx}_i) < label(\mathtt{tx}_1)$ for all $\mathtt{tx}_i$ other than $\mathtt{tx}_1, \mathtt{tx}_2$. Since an alphabetical order is adopted in a cycle, Aequitas will output $\sigma_{\mathsf{Aequitas}} = \mathtt{tx}_2 \prec \ldots \prec \mathtt{tx}_1$; i.e., $\sigma_{\mathsf{Aequitas}}(\mathtt{tx}_1) - \sigma_{\mathsf{Aequitas}}(\mathtt{tx}_2) = m - 1$. Note that Themis can also output $\sigma_{\mathsf{Aequitas}}$ if the transaction label is well-selected and the Hamiltonian cycle starts with $\mathtt{tx}_2$. Refer to [26] to see a detailed profile example $\mathcal{R}$ such that for all $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$, an output σ satisfying our definition yields $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') \leq 1$. However, Aequitas and Themis outputs an order $\sigma_{\mathsf{Aequitas}}$ and there exist some $\mathtt{tx} \prec \mathtt{tx}'$ such that $\sigma_{\mathsf{Aequitas}}(\mathtt{tx}) - \sigma_{\mathsf{Aequitas}}(\mathtt{tx}') = m - 1$.

Example 2 (pompe and wendy). Suppose $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$, we assign timestamps to transactions so that the median timestamps yield $\mathsf{med}(\mathtt{tx}_2) < \mathsf{med}(\mathtt{tx}_i) < \mathsf{med}(\mathtt{tx}_1)$ for all i such that $\mathtt{tx}_i$ is a transaction other than $\mathtt{tx}_1, \mathtt{tx}_2$. Since median timestamp decides the final order, pompe and wendy will output $\sigma_{\mathsf{pompe}} = \mathtt{tx}_2 \prec \ldots \prec \mathtt{tx}_1$; i.e., $\sigma_{\mathsf{pompe}}(\mathtt{tx}_1) - \sigma_{\mathsf{pompe}}(\mathtt{tx}_2) \leq m - 1$. Refer to [26] to see a detailed profile example $\mathcal{R}$ such that for all $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$, an output σ satisfying our definition yields $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') \leq \lceil m/3 \rceil$. However, pompe and wendy outputs an order σ_{pompe} on $\mathcal{R}$ and there exist $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ such that $\sigma_{\mathsf{pompe}}(\mathtt{tx}) - \sigma_{\mathsf{pompe}}(\mathtt{tx}') = m - 1$.

3.4 Fairness versus Liveness

We define our fair order notions based on the *complete* transaction profiles. However, during the protocol execution parties can only learn a prefix of their profiles. In this section we discuss the inherent tension between liveness and order fairness. Specifically, we prove that it is *impossible* to satisfy all desired properties when the transaction dissemination is asynchronous (even if in the non-corrupting setting); next, we show that, when there is an upper bound on transaction diffusion, it is *possible* to have liveness with relatively weak but still useful fairness.

Fairness in an Asynchronous Network. Suppose the transaction dissemination is asynchronous — i.e. a transaction can appear at any position of a (complete) transaction profile. In order to get a complete view of the transaction set that precedes a specific transaction $\mathtt{tx}$, parties may have to wait indefinitely from the first time they saw $\mathtt{tx}$. Note that standard liveness is still applicable with asynchronous transaction diffusion network. We have the following dilemma: if a Condorcet cycle spans for a long period of time and part of the transactions are delivered to all participants, then these transactions should appear in the (settled) output. In such scenario, parties have to decide the order with incomplete information.

We show below that the asynchronous dissemination will inevitably lead to the failure of $(\varphi, \mathtt{DBW})$-order-fairness. I.e. in order to satisfy consistency and liveness, the honest parties have to output an ordering σ on $\mathcal{R}$ such that $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ but $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') = n - 1$ where n is the total number of transactions in $\mathcal{R}$.

The general proof idea is to construct two executions that are indistinguishable up to some time $t + L$ (L is the liveness parameter) so that parties have to output transactions up to time t due to liveness. However, the transaction profiles are different after time $t + L$ such that in the first execution it forms a Condorcet cycle but there is no cycle in the second. We extract the possible outputs at the end of the first execution, by carefully considering consistency and liveness conditions in both executions. We conclude that the output in the first execution must be an ordering with the worst bandwidth, which implies the failure of order fairness.

Theorem 7. *Suppose the transaction dissemination is asynchronous, there is no protocol that can achieve consistency, liveness and* $(\varphi, \mathtt{DBW})$*-order-fairness.*

One approach to solve this dilemma is to relax liveness (a.k.a. weak-liveness, cf. [25]). I.e., standard liveness holds if there is no Condorcet cycle or a cycle does not span for long time; however, the system completely loses liveness during the ongoing of a large cycle.

Definition 5 (Weak-liveness, informal). *If a transaction* $\mathtt{tx}$ *is provided to all honest parties for sufficiently many consecutive rounds, then* $\mathtt{tx}$ *will be in the (settled) ledger eventually.*

Weak-liveness is not in-line with the standard BFT SMR problem; and since Condorcet cycles can chain together thus form a cycle of infinite length, it is also difficult to measure how "weak" this relaxation is compared with the standard definition (it is subject to the largest cycle in transaction profiles). Hence, we turn to another direction towards the reconciliation — we would like to achieve standard liveness as well as slightly weaker (but still non-trivial) fairness.

Fairness with Δ_{tx}-disseminated Transactions. Suppose there exists an upper bound Δ_{tx} on transaction dissemination, i.e., if t is the earliest timestamp associated with $\texttt{tx}$, then in all honest profiles it cannot be the case $\langle \texttt{tx}, t' \rangle$ for $t' \geq t$. We show that the results in Theorem 7 can be mitigated in this scenario.

The core observation is, if a Condorcet cycle spans for a long period of time, we can perform partition on the set of transactions in this cycle, and these partitions correspond to a good partition on the dependency graph such that we can figure out an upper bound on the DirectedBandwidth problem.

The partition rule on the dependency graph goes as follows. Let T_{SCC} denote the set of all transactions in the Condorcet cycle and G_{SCC} its corresponding generated graph. Consider a timestamp assignment F_{ts} on T_{SCC} and a constant $\Delta \in \mathbb{N}^+$ such that $\Delta \geq \Delta_{\text{tx}}$. An $(F_{\mathsf{ts}}, \Delta)$-partition P on T_{SCC} is a set of non-empty subsets $P_1, P_2, \ldots$ such that

$$P_i = \left\{ \texttt{tx} \mid \texttt{tx} \in T_{\mathsf{SCC}} \wedge M + (i-1)\Delta \leq F_{\mathsf{ts}}(\texttt{tx}) < M + i\Delta \right\}$$

where $M = \min\{F_{\mathsf{ts}}(\texttt{tx}) \mid \texttt{tx} \in T_{\mathsf{SCC}}\}$ (i.e. the earliest timestamp among all transactions in T_{SCC}). Note that the union of the parts of this partition is exactly the original transaction set and the intersection of two distinct parts is empty.

An $(F_{\mathsf{ts}}, \Delta)$-partition on G_{SCC}, the dependency graph of T_{SCC}, is a set of non-empty subsets $V_1, V_2, \ldots$ such that V_i is a set of vertices in G_{SCC} such that all corresponding transactions are in partition P_i. Especially, consider the earliest timestamp assignment $F_{\mathsf{ts}}^{\mathsf{min}}$, transaction dissemination Δ_{tx} and its corresponding $(F_{\mathsf{ts}}^{\mathsf{min}}, \Delta_{\text{tx}})$-partition on G_{SCC}. The bandwidth of G_{SCC} is at most twice of the maximum number of vertices in a partition.

Theorem 8. *Consider profiles $\mathcal{R}$, their dependency graph G and a strongly connected component $G_{\mathsf{SCC}} \in G$. Consider an $(F_{\mathsf{ts}}^{\mathsf{min}}, \Delta_{\text{tx}})$-partition on G_{SCC} that corresponds to the sets $V_1, V_2, \ldots,$. Then it holds that*

$$\texttt{DBW}(G_{\mathsf{SCC}}) \leq 2 \max \left| V_i \right|.$$

Note that it is a non-trivial task to design a protocol that allows parties to learn the earliest timestamp of each transaction without any trusted third party[4]. Nonetheless, a protocol can, for each transaction let parties agree on a timestamp that falls in its Δ_{tx} dissemination time window; and such protocol can be resistant to an adversary that controls up to half of the total resources, which is compliant with any admissible adversary (for technical details on synthesizing

[4] We note that so far there is no protocol that can complete this task.

a good timestamp, see Sect. 4). Thus, we consider dependency graphs with a compliant timestamp assignment $F_{\mathsf{ts}} \in \mathbb{F}_{\mathsf{ts},\mathcal{R}}$ and we allow that the specific assignment (as long as it is compliant with $\mathcal{R}$) can be chosen by the adversary.

We highlight that, in this context there exist a simple ordering trick that can provide us good bandwidth. Specifically, consider a dependency graph G and an $\mathcal{R}$-compliant timestamp assignment F_{ts}. By sorting vertices with a non-decreasing order on F_{ts} (i.e., we order u before v if $F_{\mathsf{ts}}(u) < F_{\mathsf{ts}}(v)$), it yields a vertex ordering with bandwidth upper bounded by three times the maximum total number of concurrent transactions in a Δ_{tx} time window (Theorem 9). We highlight that this ordering approach can be done without knowing the exact upper bound (Δ_{tx}) on transaction dissemination. Additionally, the bandwidth of this ordering is independent of the size of the Condorcet cycle — in other words, its performance is better on large cycles compared with small ones.

Theorem 9. *Consider profiles $\mathcal{R}$, its dependency graph G and a strongly connected component $G_{\mathsf{SCC}} \in G$. Suppose $F_{\mathsf{ts}} \in \mathbb{F}_{\mathsf{ts},\mathcal{R}}$ is a compliant timestamp assignment with respect to $\mathcal{R}$, and σ is a vertex ordering on G_{SCC} that orders vertices by a non-decreasing order on F_{ts}, then it holds that*

$$\mathtt{DBW}(\sigma, G_{\mathsf{SCC}}) \leq 3 \max \left|V_i\right|.$$

Timed Directed Bandwidth. Given that Definition 4 might conflict with liveness even if the transaction dissemination is Δ_{tx}-bounded, we shall define a feasible fair order based on our observations in Theorem 8 and 9. Our technique is to extend the bandwidth function DBW to a timed fashion — i.e., the input dependency graph G is now accompanied with the earliest time that a transaction appears in the (honest) profile. A timed directed bandwidth function TDBW on a (strongly connected) graph G with timestamp assignment $F_{\mathsf{ts}}^{\mathsf{min}}$ works as follows. If the earliest timestamp of two transactions are sufficiently apart from each other (i.e., the cycle is large and spans for a long time) then TDBW returns an upper bound as extracted in Theorem 9; otherwise it returns the directed bandwidth on graph G.

$$\mathtt{TDBW}(G) = \begin{cases} 3 \max \left|V_i(G)\right| & \textit{if } \exists(\mathtt{tx}, \mathtt{tx}')F_{\mathsf{ts}}^{\mathsf{min}}(\mathtt{tx}) \geq F_{\mathsf{ts}}^{\mathsf{min}}(\mathtt{tx}') + 3\Delta_{\mathrm{tx}}, \\ \mathtt{DBW}(G) & \textit{otherwise.} \end{cases}$$

We are now ready to extend the (φ, DBW)-order-fairness (Definition 4) by replacing the bandwidth function DBW with the timed bandwidth function TDBW. In this new definition, if two transactions are not within the same Condorcet cycle over all possible dependency graphs, their order in the output should follow parties' preference; if they are in the same cycle on some graphs, and all cycles are relatively small (i.e., it does not span for too long time) then their distance is upper-bounded by the largest possible bandwidth of the SCCs; and finally if some cycles do span for a long time, then we replace the upper-bound by three times the total number of concurrent transactions in a Δ_{tx} time window.

Definition 6 **((φ, $\mathtt{TDBW}$)-order-fairness).** *A profile σ is a (φ, $\mathtt{TDBW}$)-fair-order on $\mathcal{R}$ if for all $\mathtt{tx}, \mathtt{tx}'$ such that $\mathtt{tx} \prec^{\varphi}_{\mathcal{R}} \mathtt{tx}'$, it holds that*

$$\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') \leq \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \mathtt{TDBW}\big(\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}')\big),$$

where $\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}')$ is a function that outputs an SCC in G that contains both $\mathtt{tx}, \mathtt{tx}'$ if it exists, and a null graph otherwise.

3.5 Bounded Unfairness in a Permissionless Environment

In this section we show how to adapt our (φ, B)-order-fairness notion to a permissionless environment. We highlight that the only change we have to make in this new setting is to re-define the abstraction of profiles and the "order before by sufficiently many" notion ($\prec^{\varphi}$); all other definitions and arguments regarding order fairness could remain the same as above.

In a permissioned network, there is a one-to-one mapping from parties to profiles. This is because (honest) parties are online during the entire execution, thus profiles are exactly the abstract of their transaction logs at the end of the execution. Unfortunately, this is not the case in a permissionless environment in that parties can join and leave by their will (without notifying anyone else) and (possibly) no party can eventually hold a complete transaction profile.

Recall that in Sect. 2.1 we present a fine-grained classification on the type of participating parties. Especially, alert parties are the core participants that own all resources to run the protocol and have synchronized with each other. Under this dynamic participation model, we would like to use a profile to refer to the transaction log that an alert party holds at a specific round. In other words, we re-consider the mapping above in the permissionless setting as follows. Since there is no guarantee that an alert party P at round r will remain alert at any round other than r, we abstract the transaction log held by P at round r as a profile. Note that these profiles can be incomplete, i.e., it may only contain a few transactions $T \subsetneq \mathbb{T}$ and is a mapping $T \to [m]$ where $m = |T|$. We say a profile is a (P, r)-profile, if it corresponds to the transaction log of an alert party at round r. Also note that regarding Definition 2 with an admissible adversary, the number of profiles in $\mathcal{R}^{\mathcal{A}}$ should be bounded by a round-by-round fashion — i.e., at a round r, $\mathcal{R}^{\mathcal{A}}$ can report at most $t < (2\varphi - 1)h$ profiles where h is the number of (P, r)-profiles.

Then, we re-define the notion of "order before by sufficiently many". Let t be the earliest time that at least one of $\mathtt{tx}$ and $\mathtt{tx}'$ appears in φ fraction of the (P, t)-profiles. We say $\mathtt{tx}' \prec^{\varphi} \mathtt{tx}$, if during a sufficiently long period of time, say, K rounds, at least φ fraction of the (P, r)-profiles report $\mathtt{tx}' \prec \mathtt{tx}$ where $r \in [t, t+K)$ and P is an alert party at round $\mathtt{r}$.

4 Taxis Protocol

In this section we present a new protocol Taxis and its basic building blocks. The ultimate product of Taxis is a ledger $\mathcal{L}$ providing fair transaction order. Due to the space limit, all detailed protocol description and analysis are presented in the full version of this paper [26].

Before we introduce Taxis, we present its preliminary version $\mathsf{Taxis}_{\mathsf{WL}}$ as a direct comparison with $\mathsf{Aequitas}$. $\mathsf{Taxis}_{\mathsf{WL}}$ achieves consistency, weak-liveness and $(\varphi, \mathtt{DBW})$-order-fairness. Specifically, while the liveness is weak (same as $\mathsf{Aequitas}$), this protocol achieves the best transaction order fairness that we can expect. Next, by adding a few simple modifications on $\mathsf{Taxis}_{\mathsf{WL}}$, we present Taxis that reconciles the tension between liveness and fair order. The ledger of Taxis satisfies consistency, (standard) liveness and $(\varphi, \mathtt{TDBW})$-fair-order.

Taxis is a two-stage protocol that decouples the mining procedure of profiles and the final serialization of transactions. We will use blockchain as an intermediate information aggregator to collect profiles (i.e., transaction log) and build the ultimate ledger $\mathcal{L}$ on top of this blockchain. For simplicity, we present $\mathsf{Taxis}_{\mathsf{WL}}$ and Taxis assuming static number of participants and discuss how to adapt them to the dynamic participation in [26].

Blockchain Notations. A block with target $T \in \mathbb{N}$ is a quadruple of the form $\mathcal{B} = \langle ctr, r, h, x \rangle$ where $ctr, r \in \mathbb{N}$, $h \in \{0, 1\}$ and $x \in \{0, 1\}^*$. A blockchain $\mathcal{C}$ is a (possibly empty) sequence of blocks; the rightmost block by convention is denoted by head$(\mathcal{C})$ (note head$(\varepsilon) = \varepsilon$). These blocks are chained in the sense that if $\mathcal{B}_{i+1} = \langle ctr, r, h, x \rangle$, then $h = H(\mathcal{B}_i)$, where $H(\cdot)$ is cryptographic hash function with output in $\{0, 1\}^\kappa$. We adopt $\mathsf{TS}(\mathcal{B})$ to denote the timestamp of $\mathcal{B}$; and, slightly abusing the notations and omitting the current time $\mathtt{r}$, we will use $\mathcal{C}^{\lceil k}$ to denote the chain from pruning all blocks $\mathcal{B}$ such that $\mathsf{TS}(\mathcal{B}) \geq \mathtt{r} - k$.

2-for-1 Proof-of-Work. 2-for-1 PoW is a primitive that binds multiple PoW mining processes together by utilizing a single random oracle query. It was first proposed in [19] to improve the corruption threshold in ledger consensus. This primitive mitigates the possible attack with multiple independent mining processes, where the adversary can join forces to any one of the oracles and gain undesired advantage.

We will use 2-for-1 PoW to mine two types of blocks: ledger blocks and profile blocks. Ledger blocks form the Taxis blockchain and they will only include recent profile blocks (unlike regular blockchains, ledger blocks in Taxis will not include any transactions "directly"). Meanwhile, parties will use profile blocks to report their local profiles. We denote the mining target of ledger blocks and profile blocks by T_{LB} and T_{PB}, respectively. Taxis will maintain a constant ratio between them; for simplicity, in our presentation and analysis, we assume $T_{\mathsf{LB}} = T_{\mathsf{PB}}$.

The main goal of adopting 2-for-1 PoW to bind the mining process of these two types of blocks together, is to achieve better *chain quality*. Recall that chain quality is bad in the Bitcoin backbone protocol [19,20], where the adversary can contribute more blocks to the common prefix compared with her relative computational power. By adopting 2-for-1 PoW, Taxis guarantees that, for a sufficiently long time, φ fraction of parties mine φ fraction of the profiles (and they are all included by ledger blocks in the blockchain).

Freshness and Recency Parameter. For the sake of achieving better chain quality on profile blocks, certain changes should be made to the 2-for-1 mining procedure. Ideally, the adversary $\mathcal{A}$ should not be allowed to mine profile blocks timestamped in the very future; and, blocks should go stale as time passes by so that $\mathcal{A}$ cannot choose to withhold them to gain a sudden advantage. Analogous to the treatment in fruitchain [33], we introduce two mechanisms to help ensure the *freshness* of profile blocks. On one hand, the header of a profile block should point to the last block in the settled blockchain; this prevents the adversary from mining blocks in the very future, as an honest ledger block will introduce fresh randomness which is unpredictable for $\mathcal{A}$. On the other hand, we set a recency parameter R (in rounds) such that a profile block PB referring to a settled ledger block LB will only be valid before time $\mathsf{TS}(\mathsf{LB}) + R$ (in other words, it cannot be included by a ledger block with timestamp later than $\mathsf{TS}(\mathsf{LB}) + R$).

4.1 $\mathsf{Taxis_{WL}}$ Protocol

Mining Procedure. In every round, parties try to mine new blocks after they update their local chains according to the chain selection rule (see below for validation details). Two different block contents will be prepared: ledger block content `LBContent` which contains all (valid) newly seen profile blocks; and profile block content `PBContent` that includes the local profile of the miner. Parties then compute the Merkle root $\mathtt{st}_{\mathsf{LB}} = \mathsf{MerkleTree}(\mathtt{LBContent})$ and $\mathtt{st}_{\mathsf{PB}} = \mathsf{MerkleTree}(\mathtt{PBContent})$, respectively. Next, miners make a single random oracle query with the following input: ctr, a random nonce; h, the reference to previous block; h', the reference to the last block in the settled part; $\mathtt{r}$, the current timestamp; $\mathtt{st}_{\mathsf{LB}}$, the Merkle root of ledger block; and $\mathtt{st}_{\mathsf{PB}}$, the Merkle root of the profile block. They receive an ouput

$$u = H(ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}}).$$

If $u < T_{\mathsf{LB}}$, the party succeeds in mining a ledger block. A new block LB with content `LBContent` is generated and appended to the local chain. If the value of the reversed output string (which we denote by $[u]^{\mathsf{R}}$) satisfies $[u]^{\mathsf{R}} < T_{\mathsf{PB}}$, a new profile block PB is mined and will be diffused to the network.

Note that timestamp $\mathtt{r}$ is shared information in both blocks, so it is impossible to get two products with different timestamps. This prohibits the adversary from manipulating timestamp unless she completely drops from one mining procedure. For a ledger block, the reference to the settled block (h') and the Merkle root of profile blocks ($\mathtt{st}_{\mathsf{PB}}$) are dummy information and we do not care about their values, they are only useful when parties want to check their validity (see below). Similarly, for a profile block, the reference to the previous block (h) and the Merkle root of ledger blocks ($\mathtt{st}_{\mathsf{LB}}$) are dummy information.

We also highlight that there is no need for parties to include their entire transaction log in PB. A prefix of the profile can be pruned if all transactions in that prefix appear in the settled blockchain for more than K rounds (i.e., these transactions have been reported for sufficiently long time and parties agree on the set of transactions that precede them, see below for details). Note that with $\Delta_{\mathtt{tx}}$-disseminated transaction diffusion and liveness property of the blockchain, all transactions received by an honest party before time t is guaranteed to be in the settled blockchain within a constant time (see protocol analysis). Furthermore, if P notices that its local transaction log shares a common prefix with another profile block PB in the settled blockchain, then P can produce profile blocks with pointer to PB to indicate their common part and thus save space.

Validity Check of Chains. Recall that the Taxis blockchain is similar to that of Bitcoin's (except that Taxis includes additional 2-for-1 PoW information) and so we follow [20] regarding the validity of ledger blocks. In addition, we also need to check the validity of profile blocks. For a valid profile block PB, we require that its block header satisfies three properties: (i) PB correctly reports a reference to LB where LB is the last block after pruning the blockchain for k rounds; (ii) PB reports a timestamp that is earlier than the ledger block containing PB but no later than $\mathsf{TS}(\mathsf{LB}) + R$; and (iii) hash of PB block header is smaller than the profile block target T_{PB}. A chain $\mathcal{C}$ in Taxis is valid if $\mathcal{C}$ itself is valid and all the profile blocks included in $\mathcal{C}$ are valid.

Extracting Transaction Order. We detail how the ledger $\mathcal{L}$ is extracted in $\mathsf{Taxis}_{\mathsf{WL}}$. Generally speaking, parties will use profile blocks in the settled part of the blockchain to build a dependency graph; then, transaction order is determined by running graph condensation and (possibly) DirectedBandwidth algorithm (see [26]) on all SCCs. Note that all of these computations can be done locally based on the on-chain information.

As protocol execution proceeds, local chains held by honest parties will share a long common prefix (we write k as the common prefix parameter — i.e., the rounds that parties need to prune their local chain). Protocol participants will extract a transaction pool TXPool in their common prefix by selecting those transactions that have been reported for sufficiently long time. More specifically, in order for a transaction $\mathtt{tx}$ to be selected, there should exist a K-time-window of $\mathtt{tx}$, starting at time t such that (i) t is the timestamp of the earliest ledger block that includes a profile block PB reporting $\mathtt{tx}$; and (ii) this K-time-window should be fully included in the settled blockchain — i.e., at round $\mathtt{r}$ a party only considers time window that starts before round $\mathtt{r} - k - K$.

Transactions in TXPool are then added to a dependency graph G as vertices. Regarding rules to add edges, for each transaction $\mathtt{tx}$ we care about the profile blocks in its K-time-window: if the majority of these profile blocks report $\mathtt{tx}' \prec \mathtt{tx}$, then we add a *dotted* edge $(\mathtt{tx}', \mathtt{tx})$ to G (when $\mathtt{tx}'$ does not exist in G, a vertex of $\mathtt{tx}'$ is added). Note that a dotted edge $(\mathtt{tx}', \mathtt{tx})$ does not confirm the preference $\mathtt{tx}' \prec \mathtt{tx}$ in G. In order to count the edge in the subsequent computation, we need to wait for the K-time-window of $\mathtt{tx}'$ and see if the majority of those profile blocks report $\mathtt{tx} \prec \mathtt{tx}'$. When this holds, we update the dotted edge to *solid* (all the subsequent computations on G only consider solid edges). The reason for designing this two-phase edge adding rule is because, for those transaction pairs such that no φ-preference holds, the adversary might be able to report conflicting orders in the corresponding K-time-windows[5].

After constructing the dependency graph G, parties can linearize the transactions on top of G. Notice that G can be cyclic. Parties first compute the condensation graph G^* of G — i.e. each SCC is replaced by a vertex. Since G^* is acyclic, there exist source vertices (i.e., vertices without incoming edges) in G^*. Protocol participants do the following steps repeatedly. Let V_{source} denote the set of all source vertices in G^* such that for all $v \in V_{\mathsf{source}}$ all transactions in v are in TXPool (transactions that are waiting for some unconfirmed ones will never be selected in V_{source}). If V_{source} is empty then parties terminate and output the final ledger $\mathcal{L}$. Otherwise, they select $v \in V_{\mathsf{source}}$ such that the starting time of v's associated K-time-window is the earliest among V_{source} (if a vertex in G^* represents a SCC in G, we choose the earliest time window in that SCC). Then, if v represents a single vertex in G, parties append the corresponding transaction to $\mathcal{L}$ directly; otherwise, they run the DirectedBandwidth algoirthm to extract the bandwidth-optimal order l on v_{SCC} (i.e., the component in the original graph that condenses to v in G^*) and append l to $\mathcal{L}$. After processing v, we remove it from G^* and this yields a new source vertex set V_{source}.

$\mathsf{Taxis}_{\mathsf{WL}}$ **Ledger Properties.** With bounded dynamic participation and appropriate parameters, the ledger $\mathcal{L}$ of $\mathsf{Taxis}_{\mathsf{WL}}$ satisfies three properties — consistency, weak-liveness and $(\varphi, \mathtt{DBW})$-order-fairness. Note that for consistency, a suffix of $\mathcal{L}$ should be pruned to be resistant to adversarial manipulation. Refer to the protocol anaylsis in [26] to see the detailed consistency parameter.

Theorem 10. *Assuming bounded dynamic participation, bounded network delay and honest majority, there exist protocol parameterizations such that the ledger* $\mathcal{L}$ *of* $\mathsf{Taxis}_{\mathsf{WL}}$ *achieves consistency, weak-liveness and* $(\varphi, \mathtt{DBW})$*-order-fairness except with probability negligibly small in the security parameter.*

4.2 Taxis Protocol

We present the full Taxis protocol on top of $\mathsf{Taxis}_{\mathsf{WL}}$ in this section. Briefly speaking, we add a fallback mechanism to order transactions that remain unconfirmed for a long time based on the beginning point of their K-time-window. Note that we only make two simple changes in the mining and order-extraction stage.

Mining Procedure. In Taxis, parties book-keep the local receiving time of transactions; and, when mining profile blocks, they additionally attach these timestamps to each transaction. All the other steps in the mining procedure remain the same. Since parties will agree on the profiles of a transaction `tx` in a sufficiently long time window, they will agree on the timestamp vector associated with `tx` as well.

[5] When an edge from $\mathtt{tx}'$ to `tx` exist, `tx` will not get confirmed into the ledger. Also note that, with overwhelming probability, solid edges will appear on those transaction pairs with φ-preference. For details, see the protocol analysis.

Extracting Transaction Order. During the order-extraction stage, a fallback mechanism is provided to deal with cycles that span for a long time. Specifically, for all unconfirmed vertices $V_{\mathsf{unconfirmed}}$ in the condensation graph G^*, we check if there exist a vertex $v \in V_{\mathsf{unconfirmed}}$ such that its corresponding SCC (v_{SCC}) in G contain transactions whose K-time-window begins before $\mathtt{r} - (K + k + \Delta_{\mathsf{timeout}})$[6]. Note that $\Delta_{\mathsf{timeout}}$ is a timeout parameter that indicates the cycle spans for a long time (see protocol analysis for more details). If such v in G^* exists, we order all vertices in v_{SCC} in an increasing order based on their median timestamp. For a transaction $\mathtt{tx}$, its median timestamp $\mathsf{med}(\mathtt{tx})$ is computed on the timestamp vector associated with $\mathtt{tx}$ in its K-time-window. Note that since parties will agree on $\mathtt{tx}$'s timestamp vector, they will also agree on $\mathsf{med}(\mathtt{tx})$; and, taking the median guarantees that $\mathsf{med}(\mathtt{tx})$ falls in the Δ_{tx}-dissemination time window with $\mathtt{tx}$, thus the results in Theorem 9 applies.

In addition, when tracing the previous dependency graphs, Taxis will be able to detect those large cycles by carefully comparing the beginning point of K-time windows among all transactions in the cycle, so that it will process them using the same fallback mechanism (this guarantees consistency).

Taxis Ledger Properties. We provide a full analysis of the security of Taxis protocol with bounded dynamic participation in [26]. Specifically, we prove that the ledger $\mathcal{L}$ of Taxis satisfies three desired properties — consistency, (standard) liveness and (φ, $\mathtt{TDBW}$)-order-fairness.

Theorem 11. *Assuming bounded dynamic participation, bounded network delay and honest majority, there exist protocol parameterizations such that the ledger $\mathcal{L}$ of Taxis achieves consistency, liveness and* (φ, $\mathtt{TDBW}$)*-order-fairness except with probability negligibly small in the security parameter.*

Performance Analysis of Taxis. We detail the computation/communication complexity of the Taxis protocol. For the proof of work part and communication overhead, it requires a random oracle call per round and possibly (if a PoW is found) a message transmission with message size, worst case, linear in the security parameter plus the number of transactions that are disseminated within a sliding window of length polylogarithmic in the security parameter.

To maintain the local transaction dependency graph G, note that G can be built incrementally since all vertices and edges are extracted from the settled part of the blockchain; and, every time a vertex $\mathtt{tx}$ is added to G, the number of computational steps required (which will add the necessary edges between the vertices) is linear in the number of transactions that appear in $\mathtt{tx}$'s K-time-window, which is also of length polylogarithmic in the security parameter.

Regarding solving DirectedBandwidth on each SCC of the transaction dependency graph, note that while the exact algorithm from [21] consumes exponential time with respect to the number of concurrent transactions, we

[6] We note that two large cycles cannot run in parallel, and there is at most one vertex with multiple transactions that can pass the timeout check. Refer to protocol analysis for more details.

highlight that, in real execution, it runs in practical time for two reasons. First, a polynomial-time fallback will be triggered after a time slack of length $\Delta_{\mathsf{timeout}}$ has passed, where $\Delta_{\mathsf{timeout}}$ is a parameter that is of the same order of magnitude with respect to the common prefix parameter, the size of input (i.e., the number of vertices in a SCC) to DirectedBandwidth is therefore bounded by a polylogarithmic function of κ times the transaction throughput. On the other hand, the transaction dependency graph of a large Condorcet cycle is of good structure[7] such that we could improve the running time from $\mathcal{O}^*(3^n \cdot 2^{n^2})$ in [21] to $f(t) \cdot n^t \cdot 2^{nt}$ where t is the transaction throughput and $f(t)$ is a function that depends only on t, note that $t \ll n$. We present and analyze this algorithm in [26]. Also note that while this local computation is the most expensive step but it only needs to be performed once for each SCC throughout the entire protocol execution.

5 Discussion and Future Directions

Alternative Ways of Relaxing Order Fairness. In this paper we define transaction order fairness based on upper-bounding the positions that a transaction can be ordered before another when violating their preference. It is worth highlighting however that the graph theoretic model we put forth in Sect. 3 can accommodate a larger variety of order fairness relaxations.

For instance, one could consider the relaxation "an output profile σ should break the *least* number of φ-preferences." In the context of dependency graphs, this idea on fair order can be related to the FeedbackArcSet problem [4] which asks to remove a subset of edges to make the graph acyclic while keeping the subset as small as possible.

Another possible relaxation is to minimize the cumulative size of all violations. This means that instead of focusing on the maximum distance of back edges in a component, we care about their sum $\sum_{(u,v)\in E, \sigma(u)>\sigma(v)} \sigma(u) - \sigma(v)$. This definition, can be considered as the "global" variant of our order fairness notion; and, the corresponding graph problem — MinimumLinearArrangement [4] — is also well-studied.

Another flavor of fairness that can also be cast in the same context is studied in a recent work, Themis, [24], called consequent-transaction fairness, which can be viewed in our context as maximizing the number of consecutive forward edges.

Structure of Transaction Dependency Graphs. An interesting question arises with respect to $(\mathcal{R}, \varphi)$-dependency-graphs, as they were defined in Sect. 2.2. In Theorem 2 we show that, unless $\prec^{\varphi}$ is the majority relation, $G_{\mathcal{R},\varphi}$

[7] If a Condorcet cycle spans for a long time, and the time points that two transactions enter this system are sufficiently apart from each other, then the edge between these two transactions will never be selected as backward edge. For large Condorcet cycles, such type of edges account for the vast majority of all the edges. See a detailed explanation in [26].

cannot have arbitrarily small cycles. Is this also a sufficient condition? I.e., given a graph G without cycles of size less than $\lceil 1/(1-\varphi) \rceil$, do there exist profiles $\mathcal{R}$ such that $G = G_{\mathcal{R},\varphi}$? When $\prec^{\varphi}$ is the majority relation, this question was answered positively for any oriented graph by McGarvey in [29] (see Subsect. 3.2). The majority case has been studied further in other works. Stearns [35] and later Erdős and Moser [15] give bounds on the required size of $\mathcal{R}$. More recently, Alon [1] looked into a more refined property of $\mathcal{R}$. We suggest the study of similar questions with respect to $G_{\mathcal{R},\varphi}$, when $1/2 < \varphi \leq 1$ as an interesting direction.

(φ, DBW)-fair-order. Theorem 6 shows that (φ, DBW)-fair-order is the best that we can expect on a Condorcet cycle SCC in terms of bounding unfairness. We note here that it is possible for some transactions $\mathtt{tx}, \mathtt{tx}'$ to be put even closer than the bandwidth among all bandwidth-optimal orderings. Nonetheless, there is no need to push this definition a step further (e.g., to bound the distance of any two transactions by their maximum distance among all bandwidth-optimal orderings). The reason is that Definition 4 has already been restricted enough such that only a bandwidth-optimal ordering on this SCC will satisfy it. Even if we might be able to bound the distance on some transaction pairs further it does not change the set of orderings that satisfy this definition.

Securing Order Fairness with Transient Joining. Astute readers may notice that, in Sect. 3.5, it becomes impossible to achieve order fairness in the permissionless environment if the joining pattern of alert parties is transient — i.e., no party stays alert for a long time hence no transaction order preference can persist in the network. While this problem stems from the nature of permissionless settings and is thus intrinsically impossible to solve, we provide two alternative ways to model the execution environment that can offer different trade-offs.

One route is to restrict the adversarial power on registering / de-registering parties. I.e., $\mathcal{A}$ is allowed to de-register at most τ fraction of honest parties during any time window of length K, but τ should remain sufficiently small with respect to K so that when a transaction is received by sufficiently many parties earlier than another, both could be continuously reported.

Alternatively, we could extend our dynamic participation model (Sect. 2.1) to let parties "bootstrap" to collect transactions before they become alert. Specifically, we introduce a profile as a new resource that an alert party needs in order to run the protocol. If a party P has passively listened to the protocol and obtained a sufficiently long transaction log, then P is "profile-ready." Alert parties should be those that are also profile-ready. Given this and that the environment (which controls how the population of parties fluctuates) is restricted to offer a sufficient number of alert parties, in this new setting we guarantee that all alert parties can keep reporting transaction order preference; and, this mechanism is robust against the adversarial registration and de-registration on alert parties.

Order Fairness in the Permissioned Setting. We note that $(\varphi, \texttt{DBW})$-fair-order serialization can also be achieved with a PKI. Specifically, parties could make use of the broadcast and set consensus module in Aequitas [25] to let parties agree on a dependency graph; then, instead of alphabetically linearizing transactions in the same "block", parties use the directed bandwidth algorithm (refer to our full version [26]) to get the bandwidth-optimal order. With this additional treatment, we can adapt Taxis as a permissioned protocol that achieves consistency, weak-liveness and $(\varphi, \texttt{DBW})$-order-fairness.

Acknowledgements. We thank the anonymous reviewers for their valuable suggestions and feedback. Yu Shen's research has been supported by Input Output (iohk.io) through their funding of the University of Edinburgh Blockchain Technology Lab.

References

1. Alon, N.: Voting paradoxes and digraphs realizations. Adv. Appl. Math. **29**(1), 126–135 (2002). https://doi.org/10.1016/S0196-8858(02)00007-6
2. Asayag, A., et al.: A fair consensus protocol for transaction ordering. In: 2018 IEEE 26th International Conference on Network Protocols, ICNP 2018, Cambridge, UK, September 25-27, 2018, pp. 55–65. IEEE Computer Society (2018). https://doi.org/10.1109/ICNP.2018.00016
3. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: composable proof-of-stake blockchains with dynamic availability. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018: 25th Conference on Computer and Communications Security, pp. 913–930. ACM Press, Toronto, ON, Canada (2018). https://doi.org/10.1145/3243734.3243848
4. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: A note on exact algorithms for vertex ordering problems on graphs. Theory Comput. Syst. **50**(3), 420–432 (2012). https://doi.org/10.1007/s00224-011-9312-0
5. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 524–541. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_31
6. Cachin, C., Micic, J., Steinhauer, N., Zanolini, L.: Quick order fairness. In: Eyal, I., Garay, J.A. (eds.) FC 2022: 26th International Conference on Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 13411, pp. 316–333. Springer, Heidelberg, Germany, Grenada (2022). https://doi.org/10.1007/978-3-031-18283-9_15
7. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000). https://doi.org/10.1007/s001459910006
8. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000). https://eprint.iacr.org/2000/067
9. Chiang, J.H.y., David, B., Eyal, I., Gong, T.: FairPoS: input fairness in permissionless consensus. In: Bonneau, J., Weinberg, S.M. (eds.) 5th Conference on Advances in Financial Technologies (AFT 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 282, pp. 10:1–10:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.10

10. Chinn, P.Z., Chvátalová, J., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices—a survey. J. Graph Theory **6**(3), 223–254 (1982). https://doi.org/10.1002/JGT.3190060302
11. Cygan, M., Pilipczuk, M.: Faster exact bandwidth. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) Graph-Theoretic Concepts in Computer Science, 34th International Workshop, WG 2008, Durham, UK, June 30 - July 2, 2008. Revised Papers. Lecture Notes in Computer Science, vol. 5344, pp. 101–109 (2008). https://doi.org/10.1007/978-3-540-92248-3_10
12. Dubey, C.K., Feige, U., Unger, W.: Hardness results for approximating the bandwidth. J. Comput. Syst. Sci. **77**(1), 62–90 (2011). https://doi.org/10.1016/J.JCSS.2010.06.006
13. Dwork, C., Lynch, N.A., Stockmeyer, L.J.: Consensus in the presence of partial synchrony. J. ACM **35**(2), 288–323 (1988). https://doi.org/10.1145/42282.42283
14. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_10
15. Erdős, P., Moser, L.: On the representation of directed graphs as unions of orderings. Math. Inst. Hung. Acad. Sci **9**, 125–132 (1964)
16. Feige, U.: Coping with the NP-hardness of the graph bandwidth problem. In: SWAT 2000. LNCS, vol. 1851, pp. 10–19. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44985-X_2
17. Garay, J., Kiayias, A., Leonardos, N.: Full analysis of nakamoto consensus in bounded-delay networks. Cryptology ePrint Archive, Report 2020/277 (2020). https://eprint.iacr.org/2020/277
18. Garay, J., Kiayias, A.: SoK: a consensus taxonomy in the blockchain era. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 284–318. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_13
19. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
20. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 291–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_10
21. Jain, P., Kanesh, L., Lochet, W., Saurabh, S., Sharma, R.: Exact and approximate digraph bandwidth. In: Chattopadhyay, A., Gastin, P. (eds.) 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 150, pp. 18:1–18:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2019). https://doi.org/10.4230/LIPIcs.FSTTCS.2019.18
22. Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 477–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_27
23. Kelkar, M., Deb, S., Kannan, S.: Order-fair consensus in the permissionless setting. In: Cruz, J.P., Yanai, N. (eds.) APKC 2022: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop, APKC@AsiaCCS 2022, Nagasaki, Japan, 30 May 2022, pp. 3–14. ACM (2022). https://doi.org/10.1145/3494105.3526239

24. Kelkar, M., Deb, S., Long, S., Juels, A., Kannan, S.: Themis: fast, strong order-fairness in byzantine consensus. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023, pp. 475–489. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3616658
25. Kelkar, M., Zhang, F., Goldfeder, S., Juels, A.: Order-Fairness for byzantine consensus. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12172, pp. 451–480. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_16
26. Kiayias, A., Leonardos, N., Shen, Y.: Ordering transactions with bounded unfairness: definitions, complexity and constructions. Cryptology ePrint Archive, Report 2023/1253 (2023). https://eprint.iacr.org/2023/1253
27. Kursawe, K.: Wendy, the good little fairness widget: achieving order fairness for blockchains. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, pp. 25–36. AFT 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3419614.3423263
28. Malkhi, D., Szalachowski, P.: Maximal extractable value (MEV) protection on a DAG. In: Amoussou-Guenou, Y., Kiayias, A., Verdier, M. (eds.) 4th International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2022). Open Access Series in Informatics (OASIcs), vol. 110, pp. 6:1–6:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/OASIcs.Tokenomics.2022.6
29. McGarvey, D.C.: A theorem on the construction of voting paradoxes. Econometrica **21**(4), 608–610 (1953). https://doi.org/10.2307/1907926
30. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
31. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. Computing **16**(3), 263–270 (1976). https://doi.org/10.1007/BF02280884
32. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22
33. Pass, R., Shi, E.: FruitChains: a fair blockchain. In: Schiller, E.M., Schwarzmann, A.A. (eds.) 36th ACM Symposium Annual on Principles of Distributed Computing, pp. 315–324. Association for Computing Machinery, Washington, DC, USA (2017). https://doi.org/10.1145/3087801.3087809
34. Schneider, F.B.: Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Comput. Surv. **22**(4), 299–319 (1990). https://doi.org/10.1145/98163.98167
35. Stearns, R.: The voting problem. Am. Math. Mon. **66**(9), 761–763 (1959). https://doi.org/10.1080/00029890.1959.11989405
36. Vafadar, M.A., Khabbazian, M.: Condorcet attack against fair transaction ordering. In: Bonneau, J., Weinberg, S.M. (eds.) 5th Conference on Advances in Financial Technologies (AFT 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 282, pp. 15:1–15:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.15
37. Zhang, Y., Setty, S., Chen, Q., Zhou, L., Alvisi, L.: Byzantine ordered consensus without byzantine oligarchy. In: Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation. USENIX Association, USA (2020)