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, aggelos.kiayias@ed.ac.uk
[2]National and Kapodistrian University of Athens, nikos.leonardos@gmail.com
[3]University of Edinburgh, yu.shen@ed.ac.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 deep 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 investigated 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 injected 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 transaction. 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 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 in the permissionless setting. We present two variants of our protocol, one that matches the property perfectly but (necessarily) lacks in performance and liveness, and a second variant that achieves liveness and better complexity while offering a slightly relaxed version of the directed bandwidth definition. Finally, we comment on applications of our work to social choice theory, a direction which we believe to be of independent interest.

[*]This work was supported by Input Output (iohk.io) through their funding of the Edinburgh Blockchain Technology Lab.

Contents

1 Introduction **3**
- 1.1 Our Results . . . 4

2 Preliminaries **6**
- 2.1 Protocol Execution Model . . . 6
- 2.2 Transaction Profiles and Dependency Graphs . . . 7

3 Order Fairness **9**
- 3.1 Bounded Unfairness and Serialization . . . 9
- 3.2 Transaction Dependency Graphs . . . 10
- 3.3 Bounded Unfairness from Directed Bandwidth . . . 11
- 3.4 Fairness versus Liveness . . . 15
- 3.5 Bounded Unfairness in a Permissionless Environment . . . 17

4 Taxis Protocol **18**
- 4.1 $\mathsf{Taxis}_{\mathsf{WL}}$ Protocol . . . 19
- 4.2 Taxis Protocol . . . 22
- 4.3 Taxis with Dynamic Participation . . . 24

5 Discussion and Future Directions **25**

A Further Related Works **29**
- A.1 Blind Order Fairness . . . 29
- A.2 Block Order Fairness . . . 30
- A.3 Timed Order Fairness. . . . 31

B Preliminaries (Cont'd) **31**

C Proofs, Algorithms and Examples Omitted in the Main Body **34**
- C.1 Proofs Omitted in Section 3 . . . 34
- C.2 Hardness of DirectedBandwidth over Oriented Graphs . . . 36
- C.3 An Exact Algorithm for DirectedBandwidth . . . 36
- C.4 A Faster Algorithm for DirectedBandwidth over Dependency Graphs . . . 37
- C.5 Upper Bound on Worst Case Directed Bandwidth . . . 39
- C.6 Examples of Comparison with Existing Protocols . . . 40
- C.7 Taxis protocols . . . 41

D Security Analysis **43**
- D.1 Notations and Preliminary Results . . . 43
- D.2 Properties of Profile Blocks and Median Timestamp . . . 48
- D.3 Consistency, Liveness and Order-Fairness . . . 51

1 Introduction

The development of blockchain protocols, starting with the Bitcoin blockchain [Nak08], lead to increased interest in a classic problem in distributed systems — the state-machine replication (SMR) problem [Sch90]. 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. [GKL15, GKL17, PSS17, GK20], 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 [Sch90], 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 [KZGJ20], 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 $\mathtt{tx}$ and $\mathtt{tx}'$ 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 protocol designers 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 [KZGJ20], fair ordering in the output applies only to appropriately "spaced apart" transactions across all nodes. In timed-relative fairness, cf. [Kur20], if a transaction $\mathtt{tx}$ is received by *all* honest parties prior to $\mathtt{tx}'$, then $\mathtt{tx}$ must be sequenced before $\mathtt{tx}'$. 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, 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. Further, by giving up serialization of all output transactions, one also runs the risk of trivializing the challenge of fair ordering: the trivial protocol that issues all outputs in a single uninterrupted batch also satisfies block-order fairness.

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.

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 $\mathtt{tx}, \mathtt{tx}'$, we say that $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ if φ fraction of input profiles present $\mathtt{tx}$ before $\mathtt{tx}'$. Given any two such transactions, the output ordering σ should satisfy $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') \leq B.$, i.e., $\mathtt{tx}$ cannot be serialized more than B positions later compared to $\mathtt{tx}'$.

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 $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ to be "unfairly" sequenced as $(\mathtt{tx}', \ldots, \mathtt{tx})$ with $\mathtt{tx}'$ 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. Observe that minimizing B is of practical relevance in the context of an SMR system: the smallest B is, the smallest number of unfair state updates will occur between the processing of any two input transactions.

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 $(\mathtt{tx}, \mathtt{tx}')$ if and only if $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$. Given this, we observe that the problem of (φ, B) fairness relates to the concept of *graph bandwidth* over G, cf. [JKL$^+$19]. 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 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 impos-

sibility one has to either settle for weak-liveness (transactions are included *eventually* [KZGJ20]) 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 $\Delta_{\mathtt{tx}}$. 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 $\Delta_{\mathtt{tx}}$ 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 [GKL17] that are included provided they are sufficiently recent using the recency condition of [PS17]. 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 [GKL17, GKL20]. 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 [KDL+21], hence for practical purposes exponential dependency on their corresponding subgraph length may not be prohibitive. Furthermore, for Taxis, assuming a $\Delta_{\mathtt{tx}}$ 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 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 Section 2, we present preliminaries on the protocol exeuction model, transaction profiles and dependency graphs. We formally define and analyze transaction order fairness in Section 3. In Section 4, we present our fair-order protocol Taxis. We discuss some future research directions that might be of independent interest in Section 5. Detailed description of related works, preliminaries, protocols, and proofs can be found in the appendix.

2 Preliminaries

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

2.1 Protocol Execution Model

We present our model in light of [GKL20]. Generally speaking, we build on Canetti's formulation of "real world" notion of protocol execution [Can00a, Can00b] for multi-party protocols and adapt it to the bounded delays, dynamic participation and respecting environment.

The protocol execution proceeds in "rounds". 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 $\mathcal{F}_{\text{RO}}$ and abstracts parties attempting to solve "proof-of-work" [DN92]. Following the convention that different types of messages are diffused by their own network, we consider two diffusion functionalities — $\mathcal{F}^{\Delta}_{\mathsf{Diffuse}}$ for general messages and $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$ for transactions. For all messages except transactions, the communication is bounded-delay (a.k.a., "partial synchronous" [DLS88, PSS17]). 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. We denote the network functionality disseminating transactions by $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$. Parties will communicate with $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$ to receive new transactions. Note that different from $\mathcal{F}^{\Delta}_{\mathsf{Diffuse}}$ where the adversary $\mathcal{A}$ can manipulate messages by adding adversarial delays or re-ordering them, $\mathcal{A}$ can no longer launch the same attack on transactions in $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$.

The environment program $\mathcal{Z}$ is responsible for generating new transactions and handing them to $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$. We consider two types of transaction diffusion functionality — $\mathcal{F}^{\mathsf{async}}_{\mathsf{Diffuse},\mathtt{tx}}$ and $\mathcal{F}^{\Delta_{\mathrm{tx}}}_{\mathsf{Diffuse},\mathtt{tx}}$. The first functionality captures the asynchronous transaction diffusion, i.e., $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$ can deliver a transaction $\mathtt{tx}$ at any time (after $\mathtt{tx}$ is generated by $\mathcal{Z}$). The only restriction is that $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$ should send all transactions to all parties eventually. The second one captures a Δ_{tx}-disseminated transaction diffusion network. Specifically, in $\mathcal{F}^{\Delta_{\mathrm{tx}}}_{\mathsf{Diffuse},\mathtt{tx}}$ forces transactions to be delivered to all the honest parties within Δ_{tx} rounds after they are 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 and respecting environment. In order to describe the protocol execution in a more realistic fashion, following the treatment in [BGK+18], we classify protocol participants into different types (detailed in Table 1). 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 put some restriction on the environment's power to fluctuate the number of *alert* parties. Suppose $\mathcal{Z}$ with fixed coins produces a sequence of parties h_r where r ranges over all rounds of the execution. We define the following property (cf. [GKL20]):

Definition 1. *For $\gamma \in \mathbb{R}^+$ we call $(h_r)_{r\in[0,B)}$, where $B \in \mathbb{N}$, (γ, s)-respecting if for any set $S \subseteq [0, B)$ of at most s consecutive integers, $\max_{r\in S} h_r \leq \gamma \cdot \min_{r\in S} h_r$.*

We say that $\mathcal{Z}$ is (γ, s)-respecting if for all $\mathcal{A}$ and coins for $\mathcal{Z}$ and $\mathcal{A}$ the sequence of *alert* parties h_r is (γ, s)-respecting. Note that this respecting environment now ties closer to the dynamic participation and applies Definition 1 on the set of alert parties (which is slightly different from [GKL17, GKL20]).

State machine replication. State machine replication [Sch90] is a problem that asks a set of parties accepting input logs to maintain a public data structure that serializes the logs. This public data structure is called *ledger* in the context of ledger consensus (cf. [GKL15, GK20]). 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 replica 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 delivered. 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 $\mathtt{tx}$ in $\mathcal{R}$ as $\mathsf{TS}(\mathtt{tx})$. We call the profiles $\mathcal{R}$ Δ_{tx}-disseminated if for all transactions, the timestamps associated with them are within a Δ_{tx} time window (i.e. $\forall \mathtt{tx} \in \mathbb{T}, \max \mathsf{TS}(\mathtt{tx}) - \min \mathsf{TS}(\mathtt{tx}) \leq \Delta_{\text{tx}}$).

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 $\forall \mathtt{tx} \in \mathbb{T}, F_{\mathsf{ts}}(\mathtt{tx}) \in \mathsf{TS}(\mathtt{tx})$, 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 $F^{\mathsf{min}}_{\mathsf{ts}}$ (i.e., $\forall \mathtt{tx} \in \mathbb{T}, F^{\mathsf{min}}_{\mathsf{ts}}(\mathtt{tx}) = \min \mathsf{TS}(\mathtt{tx})$).

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 $\mathtt{tx}_i$, add a vertex v_i to $G_{\mathcal{R},\varphi}$; then, for any pair of vertices $\mathtt{tx}_i, \mathtt{tx}_j$, add an edge (v_i, v_j) if $\mathtt{tx}_i \prec^{\varphi} \mathtt{tx}_j$. 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 $\mathtt{tx}$ and its generated vertex v interchangeably. For instance, $(\mathtt{tx}, \mathtt{tx}')$ denotes the edge from vertex v generated by $\mathtt{tx}$ to vertex v' generated by $\mathtt{tx}'$. And $\sigma(\mathtt{tx})$ is the same as $\sigma(v)$ where v is generated by $\mathtt{tx}$.

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 Appendix C.

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 2 ((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 2 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[1]; 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 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

[1]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 $m - 1$.

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 3 (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 $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ 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 3 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 [McG53] 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 $\mathsf{R}_1(\mathtt{tx}_i) = 1, \mathsf{R}_1(\mathtt{tx}_j) = 2, \mathsf{R}_2(\mathtt{tx}_i) = m-1, \mathsf{R}_2(\mathtt{tx}_j) = m$ and $\mathsf{R}_1(\mathtt{tx}_k) + \mathsf{R}_2(\mathtt{tx}_k) = m + 1$ for all $k \neq i, j$ — i.e., $\mathtt{tx}_i, \mathtt{tx}_j$ 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), $\mathtt{tx}_i, \mathtt{tx}_j$ 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 $\mathtt{tx} \prec \mathtt{tx}'$, the adversary can collude with the minority and vanish this preference in $\mathcal{R}$; similarly, when $(\varphi h - 1)$ honest parties receive $\mathtt{tx} \prec \mathtt{tx}'$, 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 anadmissible 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 $\mathtt{tx}'$ too early before $\mathtt{tx}$ when $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$. 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 [JKL+19] (where they consider the forward edges), we state this problem as DirectedBandwidth in Definition 4.

Definition 4 (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 4, Bandwidth [CCDG82, Fei00, CP08] 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 [Pap76] and NP-hard to approximate within any constant ratio [DFU11] 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. Refer to Appendix C.2 for a detailed proof.

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.* [JKL+19] 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 Appendix C.3.

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[2]. In Theorem 5 we show that the above simple approach to check bandwidth will soon terminate on some graphs by considering Zarankiewicz's problem. We refer to Appendix C.5 for a detailed proof.

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

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

(φ, DBW)-fair-order. After extracting the directed bandwidth of a graph in Definition 4, we are now ready to define fair order based on upper-bounding how much $\texttt{tx}'$ can be ordered before $\texttt{tx}$ when $\texttt{tx} \prec^{\varphi} \texttt{tx}'$.

Note that, given a transaction profile set $\mathcal{R}$ and its dependency graph G, we cannot simply define the upper bound as $\texttt{DBW}(G)$. This is because G might contain several strongly connected

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

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.

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

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

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 5) on top of Definition 2 by replacing the initial function with largest 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 5 ($(\varphi, \texttt{DBW})$-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*

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

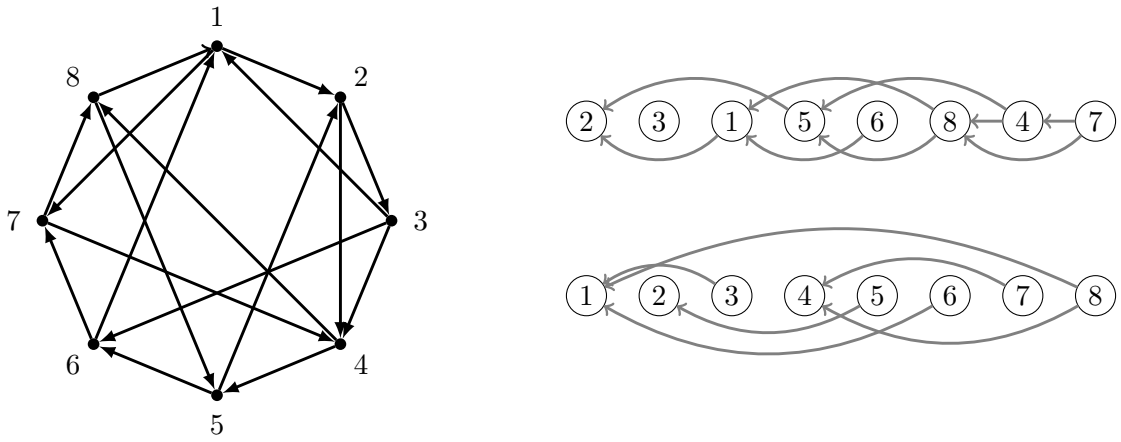
where $\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}')$ is a function that outputs an SCC in G that contains both $\texttt{tx}, \texttt{tx}'$ if it exists, and a null graph otherwise.

Note that in an all honest setting, no $\mathcal{R}^{\mathcal{A}}$ exists, thus Definition 5 can be simplified as "$\texttt{tx} \prec^{\varphi} \texttt{tx}' \implies \sigma(\texttt{tx}) - \sigma(\texttt{tx}') \leq \texttt{DBW}(\mathsf{SCC}(G_{\mathcal{R},\varphi}, \texttt{tx}, \texttt{tx}'))$". See below for an example where we have 8 transactions in $\mathcal{R}$ and the $(\mathcal{R}, \varphi)$-dependency-graph is illustrated in Figure 1(a). Since $\texttt{DBW}(G_{\mathcal{R},\varphi}) = 3$, a $(\varphi, \texttt{DBW})$-fair-order on $\mathcal{R}$ should satisfy $\texttt{tx} \prec^{\varphi} \texttt{tx}' \implies \sigma(\texttt{tx}) - \sigma(\texttt{tx}') \leq 3$. 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 Figure 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 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 5 is the most precise definition that we can make on top of Definition 2 and 3. For any new definition that tries to further reduce $\max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \texttt{DBW}(\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}'))$ for transactions $\texttt{tx}, \texttt{tx}'$, there will exist some profiles $\mathcal{R}^{\mathcal{A}}$ leading to SCCs with large bandwidth which can invalidate the new definition. Refer to Section 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 $\texttt{tx}, \texttt{tx}'$ with $\texttt{tx} \prec^{\varphi} \texttt{tx}'$, such that B satisfies $B(\mathcal{R}, \varphi, \texttt{tx}, \texttt{tx}') \geq \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \texttt{DBW}(\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}'))$.*

Proof. Fix an order fairness parameter φ. Towards a contradiction, suppose there exist a function B such that $\exists \mathcal{R}, \forall (\texttt{tx}, \texttt{tx}'), B(\mathcal{R}, \varphi, \texttt{tx}, \texttt{tx}') < \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \texttt{DBW}(\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}'))$. We consider a protocol Π that implements (φ, B)-fair-order serialization and an execution E of Π. Suppose in E an honest party outputs $\sigma = F(\langle \mathcal{R}, \mathcal{R}^{\mathcal{A}} \rangle)$. Let $\mathcal{R}^{\mathcal{A}}$ be profiles such that the $(\langle \mathcal{R}, \mathcal{R}^{\mathcal{A}} \rangle)$-dependency-graph G has a SCC G_{SCC} containing $\texttt{tx}, \texttt{tx}'$ such that $\texttt{DBW}(G_{\mathsf{SCC}}) = \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \texttt{DBW}(\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}'))$. For any vertex ordering σ on G, we have $\sigma(\texttt{tx}) - \sigma(\texttt{tx}') \geq \texttt{DBW}(G_{\mathsf{SCC}}) = \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \texttt{DBW}(\mathsf{SCC}(G, \texttt{tx}, \texttt{tx}')) > B(\mathcal{R}, \varphi, \texttt{tx}, \texttt{tx}')$. This contradicts the fact that Π implements (φ, B)-fair-order serialization. □



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

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

Figure 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).

Comparison with existing protocols. We show that Aequitas [KZGJ20], Themis [KDL+21], pompe [ZSC+20] and wendy [Kur20] fail to implement $(\varphi, \texttt{DBW})$-fair-order serialization (Definition 3 and 5) 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 $\texttt{tx}, \texttt{tx}'$ are at the head and rear respectively but it holds $\texttt{tx}' \prec^{\varphi} \texttt{tx}$. 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 $\texttt{tx}_1 \prec^{\varphi} \texttt{tx}_2$, we assign labels to transactions such that $\mathit{label}(\texttt{tx}_2) < \mathit{label}(\texttt{tx}_i) < \mathit{label}(\texttt{tx}_1)$ for all $\texttt{tx}_i$ other than $\texttt{tx}_1, \texttt{tx}_2$. Since an alphabetical order is adopted in a cycle, Aequitas will output $\sigma_{\mathsf{Aequitas}} = \texttt{tx}_2 \prec \ldots \prec \texttt{tx}_1$; i.e., $\sigma_{\mathsf{Aequitas}}(\texttt{tx}_1) - \sigma_{\mathsf{Aequitas}}(\texttt{tx}_2) = m - 1$. Note that Themis can also output $\sigma_{\mathsf{Aequitas}}$ if the transaction label is well-selected and the Hamiltonian cycle starts with $\texttt{tx}_2$. Refer to Appendix C.6 to see a detailed profile example $\mathcal{R}$ such that for all $\texttt{tx} \prec^{\varphi} \texttt{tx}'$, an output σ satisfying our definition yields $\sigma(\texttt{tx}) - \sigma(\texttt{tx}') \leq 1$. However, Aequitas and Themis outputs an order $\sigma_{\mathsf{Aequitas}}$ and there exist some $\texttt{tx} \prec \texttt{tx}'$ such that $\sigma_{\mathsf{Aequitas}}(\texttt{tx}) - \sigma_{\mathsf{Aequitas}}(\texttt{tx}') = m - 1$.

Example 2 (pompe and wendy). Suppose $\texttt{tx}_1 \prec^{\varphi} \texttt{tx}_2$, we assign timestamps to transactions so that the median timestamps yield $\mathsf{med}(\texttt{tx}_2) < \mathsf{med}(\texttt{tx}_i) < \mathsf{med}(\texttt{tx}_1)$ for all i such that $\texttt{tx}_i$ is a transaction other than $\texttt{tx}_1, \texttt{tx}_2$. Since median timestamp decides the final order, pompe and wendy will output $\sigma_{\mathsf{pompe}} = \texttt{tx}_2 \prec \ldots \prec \texttt{tx}_1$; i.e., $\sigma_{\mathsf{pompe}}(\texttt{tx}_1) - \sigma_{\mathsf{pompe}}(\texttt{tx}_2) \leq m - 1$. Refer to Appendix C.6 to see a detailed profile example $\mathcal{R}$ such that for all $\texttt{tx} \prec^{\varphi} \texttt{tx}'$, an output σ satisfying our definition yields $\sigma(\texttt{tx}) - \sigma(\texttt{tx}') \leq \lceil m/3 \rceil$. However, pompe and wendy outputs an order σ_{pompe} on $\mathcal{R}$ and there exist $\texttt{tx} \prec^{\varphi} \texttt{tx}'$ such that $\sigma_{\mathsf{pompe}}(\texttt{tx}) - \sigma_{\mathsf{pompe}}(\texttt{tx}') = m - 1$.

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. [KZGJ20]). 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 6 (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* $\mathcal{L}$ *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 $\mathtt{tx}$, then in all honest profiles it cannot be the case $\langle \mathtt{tx}, t' \rangle$ 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

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

where $M = \min\{F_{\mathsf{ts}}(\mathtt{tx}) \mid \mathtt{tx} \in T_{\mathsf{SCC}}\}$ (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 $F^{\mathsf{min}}_{\mathsf{ts}}$, transaction dissemination Δ_{tx} and its corresponding $(F^{\mathsf{min}}_{\mathsf{ts}}, \Delta_{\text{tx}})$-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 $(F^{\mathsf{min}}_{\mathsf{ts}}, \Delta_{\text{tx}})$-partition on G_{SCC} that corresponds to the sets $V_1, V_2, \ldots,$. Then it holds that*

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

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[3]. 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 a good timestamp, see Section 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*

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

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

Timed directed bandwidth. Given that Definition 5 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 $\texttt{DBW}$ 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 $\texttt{TDBW}$ 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 $\texttt{TDBW}$ returns an upper bound as extracted in Theorem 9; otherwise it returns the directed bandwidth on graph G.

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

We are now ready to extend the $(\varphi, \texttt{DBW})$-order-fairness (Definition 5) by replacing the bandwidth function $\texttt{DBW}$ with the timed bandwidth function $\texttt{TDBW}$. 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 7 ($(\varphi, \texttt{TDBW})$-order-fairness). *A profile σ is a $(\varphi, \texttt{TDBW})$-fair-order on $\mathcal{R}$ if for all $\texttt{tx}, \texttt{tx}'$ such that $\texttt{tx} \prec_{\mathcal{R}}^{\varphi} \texttt{tx}'$, it holds that*

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

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.

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 Section 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 $T \subseteq \mathbb{T}$ 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 3 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.

Before we introduce Taxis, we present its preliminary version $\mathsf{Taxis}_{\mathsf{WL}}$ as a direct comparison with Aequitas. $\mathsf{Taxis}_{\mathsf{WL}}$ achieves consistency, weak-liveness and $(\varphi, \mathtt{DBW})$-order-fairness. Specifically, while the liveness is weak (same as 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 Section 4.3.

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 $\mathrm{head}(\mathcal{C})$ (note $\mathrm{head}(\varepsilon) = \varepsilon$). 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 process together by utilizing a single random oracle query. It was first proposed in [GKL15] 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 blockchain, ledger blocks in Taxis will not include any transactions). 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 [GKL15, GKL17], 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 [PS17], 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 $\mathsf{Taxis}_{\mathsf{WL}}$ 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 Δ_{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.

Protocol 1 $\mathsf{Taxis_{WL}}$-MiningProcedure($\mathtt{r}$)

Fetch information from $\mathcal{F}^{\Delta}_{\mathsf{Diffuse}}$ and $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$ and get new chains $(\mathcal{C}_1, \ldots, \mathcal{C}_M)$, new transactions $(\mathtt{tx}_1, \ldots, \mathtt{tx}_i)$ and new profile blocks $(\mathsf{PB}_1, \ldots, \mathsf{PB}_j)$
Set $\mathtt{buffer}_{\mathsf{PB}} \leftarrow \mathtt{buffer}_{\mathsf{PB}} \,\|\, (\mathsf{PB}_1, \ldots, \mathsf{PB}_j)$
Set $\mathtt{localProfile} \leftarrow \mathtt{localProfile} \,\|\, (\mathtt{tx}_1, \ldots, \mathtt{tx}_i)$
$\mathcal{C}_{\mathrm{loc}} \leftarrow \mathsf{maxvalid}(\mathcal{C}_{\mathrm{loc}}, \mathcal{C}_1, \ldots, \mathcal{C}_M)$
$\mathtt{LBContent} \leftarrow$ (valid) profile blocks in $\mathtt{buffer}_{\mathsf{PB}}$ that are not mined in $\mathcal{C}_{\mathrm{loc}}$
$\mathtt{PBContent} \leftarrow \mathtt{localProfile}$ ◁ Remove a prefix to save space.
$h' \leftarrow$ hash value of block head$(\mathcal{C}^{\lceil k}_{\mathrm{loc}})$
$h \leftarrow$ hash value of block head$(\mathcal{C}_{\mathrm{loc}})$
$u \leftarrow \mathsf{H}(ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}})$
if $u < T_{\mathsf{LB}}$ **then**
    Set $\mathsf{LB} \leftarrow \langle ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}} \rangle$ and $\mathcal{C}_{\mathrm{loc}} \leftarrow \mathcal{C}_{\mathrm{loc}} \,\|\, \mathsf{LB}$
    Diffuse $\mathcal{C}_{\mathrm{loc}}, \mathtt{LBContent}$
end if
if $[u]^{\mathsf{R}} < T_{\mathsf{PB}}$ **then**
    Set $\mathsf{PB} \leftarrow \langle ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}} \rangle$
    Diffuse $\mathsf{PB}, \mathtt{PBContent}$
end if
$ctr \leftarrow ctr + 1$

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 [GKL17] 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. See Appendix C.7 for the complete description of IsValidChain in $\mathsf{Taxis_{WL}}$.

Extracting transaction order. We detail how the ledger $\mathcal{L}$ is extracted in $\mathsf{Taxis_{WL}}$. 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 Algorithm 1) 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 becasue, 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[4].

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 Algorithm 1 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}.

We present the full serialization code in Protocol 2. Note that we slightly abuse the notation of block timestamp and we write $\mathsf{TS}(\mathtt{tx})$ as the beginning time of K-time-window associated with $\mathtt{tx}$— i.e., it is the timestamp of the first ledger block LB that includes a profile block with $\mathtt{tx}$.

Protocol 2 $\mathsf{Taxis_{WL}}$-$\mathsf{ExtractTransactionOrder}(\mathcal{C}_{\mathrm{loc}}, \mathtt{r})$

Initialize TXPool and graph G to be empty.
for $\mathtt{tx} \in \mathsf{PB} \in \mathsf{LB} \in \mathcal{C}_{\mathrm{loc}}^{\lceil K+k}$ **do**
    Add $\mathtt{tx}$ to TXPool and $v_{\mathtt{tx}}$ to G
end for
▷ Build graph
for $\mathtt{tx} \in \mathsf{TXPool}$ **do**
    $\mathsf{PB}_{\mathtt{tx}} \leftarrow \{\mathsf{PB} \mid \mathsf{PB} \in \mathsf{LB}$ s.t. $\mathsf{TS}(\mathtt{tx}) \le \mathsf{TS}(\mathsf{LB}) < \mathsf{TS}(\mathtt{tx}) + K\}$
    for $\mathtt{tx}' \in \mathsf{PB} \in \mathsf{PB}_{\mathtt{tx}}$ **do**
        if majority of $\mathsf{PB}_{\mathtt{tx}}$ report $\mathtt{tx}' \prec \mathtt{tx}$ **then**
            if $(\mathtt{tx}', \mathtt{tx}) \in G$ **then**
                Mark $(\mathtt{tx}', \mathtt{tx})$ as solid
            else
                Add edge $(\mathtt{tx}', \mathtt{tx})$ to G and mark as dotted
            end if
        else
            if $(\mathtt{tx}, \mathtt{tx}') \in G$ **then**
                Mark $(\mathtt{tx}, \mathtt{tx}')$ as solid
            else
                Add edge $(\mathtt{tx}, \mathtt{tx}')$ to G and mark as dotted

[4]When an edge from $\mathtt{tx}'$ to $\mathtt{tx}$ exist, $\mathtt{tx}$ 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.

19: **end if**
20: **end if**
21: **end for**
22: **end for**
▷ Compute order.
23: Compute the condensation graph G^* of G. ◁ Using solid edges only
24: Let V_{source} be the set of vertices in G s.t. v is a source vertex in G^* and all transactions in v are in TXPool.
25: **while** $V_{\mathsf{source}} \neq \emptyset$ **do**
26: Let v be the vertex in V_{source} with earliest K-time-window
27: $v_{\mathsf{SCC}} \leftarrow$ component in G that corresponds to v in G^*
28: Run Algorithm 1 on v_{SCC} and get transaction order l
29: $\mathcal{L} = \mathcal{L} \parallel l$
30: Remove v from G^* ◁ This yields a new V_{source}
31: **end while**

OUTPUT: ledger $\mathcal{L}$ (a list of transactions with strict total order).

$\mathsf{Taxis}_{\mathsf{WL}}$ 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 Appendix D and Theorem 27 to see the detailed consistency parameter.

Theorem 10 (informal). *In a (γ, s)-respecting environment, if Conditions* (C1)*,* (C2) *and* (C3) *are satisfied, 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. I.e., we replace Line 3 in Protocol 1 with "Set $\mathtt{localProfile} \leftarrow \mathtt{localProfile} \parallel (\langle \mathtt{tx}_1, \mathtt{r} \rangle, \ldots, \langle \mathtt{tx}_i, \mathtt{r} \rangle)$" where $\mathtt{r}$ is party's current local time. All the other steps in Protocol 1 remain the same. Since parties will agree on the profiles of a transaction $\mathtt{tx}$ in a sufficiently long time window, they will agree on the timestamp vector associated with $\mathtt{tx}$ as well. We present the complete mining procedure in Appendix C.7.

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 $\mathtt{r} - (K + k + \Delta_{\mathsf{timeout}})$[5]. Note that $\Delta_{\mathsf{timeout}}$ is a timeout parameter that indicates the cycle spans

[5]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.

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 the 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).

Protocol 3 Taxis-ExtractTransactionOrder($\mathcal{C}_{\mathrm{loc}}, \mathtt{r}$)

Extract TXPool and build graph G ◁ Same as Protocol 2 Line 1-22
▷ Compute order
Compute the condensation graph G^* of G ◁ Using solid edges only
Let V_{source} be the set of vertices in G s.t. v is a source vertex in G^* and all transactions in v are in TXPool
while $V_{\mathsf{source}} \neq \emptyset$ **do**
    Let v be the vertex in V_{source} with earliest K-time-window
    $v_{\mathsf{SCC}} \leftarrow$ component in G that corresponds to v in G^*
    if $\exists v_1, v_2 \in v_{\mathsf{SCC}}$ s.t. $\mathsf{TS}(v_1) + \Delta_{\mathsf{timeout}} \leq \mathsf{TS}(v_2)$ **then**
        Order transactions in v_{SCC} in an increasing order based on their median timestamps and get l.
    else
        Run Algorithm 1 on v_{SCC} and get transaction order l
    end if
    $\mathcal{L} = \mathcal{L} \parallel l$
    Remove v from G^* ◁ This yields a new V_{source}
end while
▷ Run fallback if a cycle lasts for long
Let $V_{\mathsf{unconfirmed}}$ denote all vertices that remains in the graph
for $v \in V_{\mathsf{unconfirmed}}$ **do**
    $v_{\mathsf{SCC}} \leftarrow$ component in G that corresponds to v in G^*
    if $\exists v' \in v_{\mathsf{SCC}}$ s.t. $\mathsf{TS}(v') \leq \mathtt{r} - (K + k + \Delta_{\mathsf{timeout}})$ **then**
        Order vertices in v_{SCC} in an increasing order based on their median timestamps and get a prefix l up to v'
        $\mathcal{L} = \mathcal{L} \parallel l$
    end if
end for

OUTPUT: a list of transactions $\mathcal{L}$ with strict total order.

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

Theorem 11. *In a (γ, s)-respecting environment, if Conditions* (C1), (C2) *and* (C3) *are satisfied, the ledger $\mathcal{L}$ of Taxis achieves consistency, liveness and $(\varphi, \mathtt{TDBW})$-order-fairness with parameters as specified in Theorem 27 and 28 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 `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 `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 (Algorithm 1) from [JKL+19] consumes exponential time with respect to the number of concurrent transactions, we highlight that, in real execution, it runs in practical time for two reasons. First, a polynomial-time fallback (Line 18-21 in Protocol 3) 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 (cf. Equation (1) and (2)), 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[6] such that we could improve the running time from $\mathcal{O}^*(3^n \cdot 2^{n^2})$ in [JKL+19] 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 Appendix C.4. 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.

4.3 Taxis with Dynamic Participation

We detail how to adapt the above two protocols $\mathsf{Taxis}_{\mathsf{WL}}$ and Taxis from static number of parties to the dynamic participation.

Difficulty recalculation function. In a permissionless environment, mining difficulties T_{LB} and T_{PB} should be adjusted as the participant population fluctuates. We adopt the same difficulty adjustment function in Bitcoin[7] to recalculate T_{LB} (so T_{PB} is adjusted as well). More specifically, this function adjusts mining difficulty at the end of epochs (an epoch consists of m blocks), and the target for the next epoch T'_{LB} is computed base on current target T_{LB} and the time elapsed to mine m blocks — i.e., $T'_{\mathsf{LB}} = \frac{\Lambda}{m/f} \cdot T_{\mathsf{LB}}$ where f is the ideal block generation rate, and Λ is the difference between the timestamps of the last block in the previous and current epoch. In addition, the relative amount of difficulty that can be adjusted each time is bounded by a constant τ (which is similar to that in Bitcoin where $\tau = 4$ is in use).

Counting majority as accumulated difficulty. Note that after profile blocks are mined under different difficulties, we should count "majority" in a K-time-window in a new sense. Precisely, we say some blocks account for the majority if they accumulate more than $d/2$ difficulty where d is the total difficulty of all profile blocks in the time window. This new majority rule applies to building

[6] 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 Appendix C.4.

[7] We refer to [GKL17] for more details on analysis of Bitcoin with dynamic participation.

graph stage in Protocol 2 and Protocol 3. It also applies to the selection of median timestamps. I.e., a timestamp t is selected for $\texttt{tx}$ because t is the earliest timestamp such that all profiles that report a timestamp no later than t yields more than half of the accumulated difficulties in $\texttt{tx}$'s K-time-window.

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 Section 3 can accommodate a larger variety of order fairness relaxations.

For instance, one could consider the relaxation "an output profile σ should breaks the *least* number of φ-preferences." In the context of dependency graphs, this idea on fair order can be related to the FeedbackArcSet problem [BFK+12] 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 [BFK+12] — is also well-studied.

Another flavor of fairness that can also be cast in the same context is studied in a recent work, Themis, [KDL+21], 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 Sec 2.2. In Theorem 2 we show that, unless $\prec^{\varphi}$ is the majority relation, $G_{\mathcal{R},\varphi}$ 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 [McG53] (see subsection 3.2). The majority case has been studied further in other works. Stearns [Ste59] and later Erdős and Moser [EM64] give bounds on the required size of $\mathcal{R}$. More recently, Alon [Alo02] 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.

$(\varphi,\texttt{DBW})$-fair-order. Theorem 6 shows that $(\varphi,\texttt{DBW})$-fair-order is the best that we can expect on a Condorcet cycle SCC in terms of packing the transactions as tightly as possible. We note here that it is possible for some transactions $\texttt{tx},\texttt{tx}'$ 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 5 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. Alert readers may notice that, in Section 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 kill 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 (Section 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 [KZGJ20] to let parties agree on a dependency graph; then, instead of alphabetically linearizing transactions in the same "block", parties use Algorithm 1 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.

References

[ACG+18] Avi Asayag, Gad Cohen, Ido Grayevsky, Maya Leshkowitz, Ori Rottenstreich, Ronen Tamari, and David Yakira. A fair consensus protocol for transaction ordering. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 55–65, 2018.

[Alo02] Noga Alon. Voting paradoxes and digraphs realizations. *Advances in Applied Mathematics*, 29(1):126–135, 2002.

[BFK+12] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. A note on exact algorithms for vertex ordering problems on graphs. *Theory of Computing Systems*, 50(3):420–432, Apr 2012.

[BGK+18] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 913–930, New York, NY, USA, 2018. Association for Computing Machinery.

[Bol78] Béela. Bollobás. *Extremal Graph Theory*. L.M.S. monographs. Academic Press, 1978.

[Can00a] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, Jan 2000.

[Can00b] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://ia.cr/2000/067`.

[CCDG82] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs. The bandwidth problem for graphs and matrices—a survey. *Journal of Graph Theory*, 6(3):223–254, 1982.

[CMS21] Christian Cachin, Jovana Micic, and Nathalie Steinhauer. Quick order fairness. *CoRR*, abs/2112.06615, 2021.

[CP08] Marek Cygan and Marcin Pilipczuk. Faster exact bandwidth. In Hajo Broersma, Thomas Erlebach, Tom Friedetzky, and Daniel Paulusma, editors, *Graph-Theoretic Concepts in Computer Science*, pages 101–109, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[DFU11] Chandan Dubey, Uriel Feige, and Walter Unger. Hardness results for approximating the bandwidth. *Journal of Computer and System Sciences*, 77(1):62–90, 2011. Celebrating Karp's Kyoto Prize.

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, Apr 1988.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 13–147, Berlin, Heidelberg, 1992. Springer-Verlag.

[EM64] Paul Erdős and Leo Moser. On the representation of directed graphs as unions of orderings. *Math. Inst. Hung. Acad. Sci*, 9:125–132, 1964.

[Fei00] Uriel Feige. Coping with the np-hardness of the graph bandwidth problem. In *Algorithm Theory - SWAT 2000*, pages 10–19, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[FG03] Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, PODC '03, pages 211–220, New York, NY, USA, 2003. Association for Computing Machinery.

[GK20] Juan Garay and Aggelos Kiayias. Sok: A consensus taxonomy in the blockchain era. In Stanislaw Jarecki, editor, *Topics in Cryptology – CT-RSA 2020*, pages 284–318, Cham, 2020. Springer International Publishing.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 291–323, Cham, 2017. Springer International Publishing.

[GKL20] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. Full analysis of nakamoto consensus in bounded-delay networks. Cryptology ePrint Archive, Report 2020/277, 2020. `https://ia.cr/2020/277`.

[HT94] Vassos Hadzilacos and Sam Toueg. A modular approach to fault-tolerant broadcasts and related problems, 1994.

[JKL+19] Pallavi Jain, Lawqueen Kanesh, William Lochet, Saket Saurabh, and Roohani Sharma. Exact and Approximate Digraph Bandwidth. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019)*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[KDK22] Mahimna Kelkar, Soubhik Deb, and Sreeram Kannan. Order-fair consensus in the permissionless setting. In *Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, APKC '22, pages 3–14, New York, NY, USA, 2022. Association for Computing Machinery.

[KDL+21] Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, strong order-fairness in byzantine consensus. Cryptology ePrint Archive, Paper 2021/1465, 2021. `https://eprint.iacr.org/2021/1465`.

[Kur20] Klaus Kursawe. Wendy, the good little fairness widget: Achieving order fairness for blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, pages 25–36, New York, NY, USA, 2020. Association for Computing Machinery.

[KZGJ20] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 451–480, Cham, 2020. Springer International Publishing.

[McG53] David C. McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, 21(4):608–610, 1953.

[MS] Dahlia Malkhi and Pawel Szalachowski. Maximal extractable value (mev) protection on a dag.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

[Pap76] Ch. H. Papadimitriou. The np-completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, Sep 1976.

[PS17] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 315–324, New York, NY, USA, 2017. Association for Computing Machinery.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 643–673, Cham, 2017. Springer International Publishing.

[Sax80] James B. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM Journal on Algebraic Discrete Methods*, 1(4):363–369, 1980.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, dec 1990.

[SR20] Yaakov Sokolik and Ori Rottenstreich. Age-aware fairness in blockchain transaction ordering. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pages 1–9, 2020.

[Ste59] Richard Stearns. The voting problem. *The American Mathematical Monthly*, 66(9):761–763, 1959.

[ZSC+20] Yunhao Zhang, Srinath Setty, Qi Chen, Lidong Zhou, and Lorenzo Alvisi. Byzantine ordered consensus without byzantine oligarchy. In *Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation*, USA, 2020. USENIX Association.

A Further Related Works

In a long line of research on understanding order fairness in the state machine replica (SMR) problem, Schneider [Sch90] first proposes *order* — "Every non-faulty state machine replica processes the requests it receives in the same relative order" — as the third property (other than consistency and liveness) that an ideal BFT SMR protocol should satisfy. This is later formalized by Garay and Kiayias [GK20] as "serializability" in ledger consensus. Serializability requires that if $\mathtt{tx}$ enters all honest parties' mempool before $\mathtt{tx}'$, honest parties should reject the order $\mathtt{tx}', \mathtt{tx}$ (in their settled ledger).

We provide an overview of all existing works on defining order fairness and implementing fair-order protocols. Roughly speaking, we classify them into three directions — (i) blind-order-fairness, which orders transactions while hiding their content; (ii) block-order-fairness, solving the Condorcet paradox by claiming "blocks"; and (iii) timed-order-fairness, defining a time-based fair-order without cyclic preferences. We highlight that, compared with our contributions, none of these existing fair order notions provides bounded unfairness — i.e., there is no bound in the definition on how "unfairly" a transaction could be put before another.

A.1 Blind Order Fairness

Blind order fairness [ACG+18, SR20, MS] considers the fair-order problem in a practical point of view — i.e., since transaction content is the main information that an adversary will use to manipulate transaction order, we hide the content before their order is finalized. In other words, all protocol participants (including the adversary) are "blind" to the transactions; and, when parties learn the transaction content, it has already been too late to re-order them. For the sake of hiding content, cryptographic primitives are employed in blind-order-fairness protocols, e.g., commit-and-reveal [ACG+18, SR20], verifiable secret sharing and threshold encryption [MS].

Aside from blindness, some works provide additional "fair" (random) pending transaction selection rules. Helix [ACG+18] is a protocol that applies a verifiable random sampling on the public memory pool, thus selecting transactions with equal probability. Analogous to this design, Sokolik and Rottenstreich [SR20] present a random transaction inclusion scheme with weight measured by the time that transactions stay in the mempool.

While blind-order-fairness achieves the minimum desideratum that the adversary cannot completely dominate the transaction order, we highlight a few flaws with the above-mentioned schemes. First off, employing commit-and-reveal scheme to hide transaction content would incur the "selective open" issue — i.e., the adversary can commit a large number of transactions and only open the one that gains the most profit. Moreover, blind-order-fairness is not resistant to a network adversary. For instance, the adversary can still conduct censorship and learn the transaction source

or infer part of its metadata. And, most importantly, blind-order-fairness does not reflect the real transaction diffusion pattern, which is contrary to our definition as well as block-order-fairness and timed order fairness (see below).

A.2 Block Order Fairness

Kelkar *et al.* [KZGJ20] identifies the existence of Condorcet paradox (which is first observed in the social choice theory) in defining fair-order based on aggregating φ-fraction of parties' individual preference. Condorcet paradox shows that the collective preferences can be cyclic. This non-transitivity directly leads to the impossibility of achieving receive-order-fairness where processors decide a final order following φ fraction of honest parties' choice exactly (i.e., $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$ indicates $\mathtt{tx}_1$ before $\mathtt{tx}_2$ in the output).

Towards the goal of mitigation, Kelkar *et al.* work on a relatively weak definition, which blockifies all transactions in a Condorcet cycle and claims that they are processed simultaneously. They call this φ-block-order-fairness — "If $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$, then the final output reports $\mathtt{tx}_1$ *no later than* $\mathtt{tx}_2$.". We point out that the output of φ-block-order-fairness is actually a partial order — i.e., it does not indicate the order inside a block. Therefore in Definition 8, we adopt σ' to denote a surjection from $\mathbb{T}$ to $\{1, 2, \ldots, m\}$ where $m \leq |\mathbb{T}|$.

Definition 8 (φ-block-order-fairness [KZGJ20], restated). *A function F satisfies* φ-block-order-fairness *if for all input $\mathcal{R} = \mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_n$ and $\sigma' = F(\mathcal{R})$,*

$$\mathtt{tx}_i \prec^{\varphi} \mathtt{tx}_j \Longrightarrow \sigma'(\mathtt{tx}_i) \leq \sigma'(\mathtt{tx}_j).$$

Aequitas protocol family. Kelkar *et al.* [KZGJ20] also present a protocol family Aequitas that achieves φ-block-order-fairness and can tolerate the adversarial nodes for up to $1/2$ in a synchronous network and up to $1/4$ assuming asynchronous transaction dissemination. Notably, φ-block-order-fairness in Aequitas is achieved by sacrificing (standard) liveness of the protocol. An intuitive explanation is that when the protocol execution encounters Condorcet cycles, parties have to wait indefinitely until the end of these cycles (which can last forever in the worst case), thus blocking all the transactions in the cycle from getting settled.

Following the same definition in [KZGJ20], Kelkar *et al.* [KDK22] proposes a variant of Aequitas that extends the participation model to a "permissionless" setting, where (a small fraction of) parties can join and leave but the total number of parties running the protocol remains *static* during the execution. The general idea is to let the transaction order of a modular chain simulate the receiving order of a server in the previous model. Then, parties will run the permissioned Aequitas protocol based on the on-chain data. Note that this protocol only works when no Condorcet cycles exist (this is achieved by constraining the network delay).

Remark 1. Kelkar *et al.* [KZGJ20, KDK22] claim that Aequitas can achieve both standard liveness and φ-block-order-fairness when the transaction diffusion network is fully synchronous — i.e., $\mathtt{tx}$ is delivered to all honest parties within two consecutive rounds. However, we point out that this is not true in our model where the input profiles are strict total orders on transactions. In other words, even if a bunch of transactions are received in the same "round", they may still form Condorcet cycles. Note that this is not a peculiarity of our model: any fine grain timing model would result on a strict total ordering for the input profiles.

In order to solve the weak-liveness issue in Aequitas, Kelkar *et al.* propose a leader-based permissioned protocol Themis [KDL+21] that achieves Definition 8 with standard liveness. This protocol is built on top of Hotstuff and is resistant to up to $1/4$ corrupted nodes. Themis allows

leaders to split transactions in a Condorcet cycle into batches and output transactions in each batch following the Hamiltonian-cycle order. We discuss the downside of this ordering policy in the context of Definition 5 in Appendix C.6.

It is also worth noting that Cachin *et al.* [CMS21] elaborate on a variant of φ-block-order-fairness, defining a differential order fairness property based on the standard validity notions for consensus protocols [FG03]. They also present an efficient atomic broadcast protocol that guarantees message delivery in a differential fair order.

A.3 Timed Order Fairness.

Block-order-fairness suffers from potential liveness failure unless assuming a non-corrupting adversary and order fairness parameter $\varphi = 1$. However this would tremendously restrict the usage scenarios of the fair-order protocol. Thus, some follow-up works [Kur20, ZSC+20] focus on further weakening the order fairness notion to circumvent Condorcet cycles.

Since every protocol participants will conventionally maintain a local clock, it is reasonable to let them assign a local timestamp to each transaction. Now, consider two (local) timestamp vectors of transaction $\mathtt{tx}, \mathtt{tx}'$ respectively. We can decide the their order as $\mathtt{tx}, \mathtt{tx}'$ if these two vectors are "separated" by a timestamp τ. More precisely, "separate" means that all timestamps assigned to $\mathtt{tx}$ are earlier than τ and all timestamps assigned to $\mathtt{tx}'$ are later than τ.

Definition 9 (timed-order-fairness [Kur20, ZSC+20], restated). *A function F satisfies* timed-order-fairness *if for all input $\mathcal{R} = \mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_n$ and $\sigma = F(\mathcal{R})$,*

$$\max F_{\mathsf{ts}}(\mathtt{tx}_i) < \min F_{\mathsf{ts}}(\mathtt{tx}_j) \Longrightarrow \sigma(\mathtt{tx}_i) < \sigma(\mathtt{tx}_j)$$

where F_{ts} is an admissible timestamp assignment with $\mathcal{R}$.

While working on the same direction, [Kur20, ZSC+20] look at different problems. [Kur20] designs a widget which consists of a fixed number of reputable validators to help any blockchain protocols ensure fair ordering; and [ZSC+20] presents a state machine replica BFT protocol to achieve order fairness.

Timed-order-fairness has been criticized for strongly relying on synchronized clocks. For instance, the definition becomes meaningless when local clocks are apart from each other for an hour. Nonetheless, we highlight that "timestamps" in Definition 9 does not necessarily need to be the real world time. In fact, timed-order-fairness works as long as the protocol itself can maintain a "protocol time", which let all the time-aware parties' local clocks stays in a narrow interval. This can be easily achieved with some clock synchronization protocols.

B Preliminaries (Cont'd)

Round structure and protocol execution. As in [GKL15, PSS17], the protocol execution proceeds in "rounds" with inputs provided by an environment program denoted by $\mathcal{Z}$ to parties that execute the protocol Π. The adversary $\mathcal{A}$ is "adaptive," and allowed to take control of parties on the fly, as well as "rushing," meaning that in any given round the adversary gets to observe honest parties' actions before deciding how to react. The diffusion functionality is similar to those in [GKL15, PSS17]; it allows order of messages to be controlled by $\mathcal{A}$, i.e., there is no atomicity guarantees in message broadcast [HT94], and, furthermore, the adversary is allowed to spoof the source information on every message (i.e., communication is not authenticated). $\mathcal{A}$ can inject messages for selective delivery but cannot change the contents of the honest parties' messages nor

prevent them from being delivered beyond Δ rounds of delay — a functionality parameter. The precise value of Δ will be unknown to the protocol and hence protocol participants will not use Δ as a parameter to select or filter the messages.

The environment program $\mathcal{Z}$ determines the protocol execution; it creates and interacts with other instances of programs at the discretion of a control program C. Following [Can00b], $(\mathcal{Z}, C)$ forms of *a system of interactive Turing machines* (ITM's). The only instances allowed by C are those of the protocol program Π, an adversary $\mathcal{A}$. These are called ITI's (interactive Turing Machines Instances). We refer to [Can00b] for further details on the mechanics of the model. The only additional feature that is relevant to our setting is that we assume each instance is initialized with a special Boolean flag denoted as `isAlert` which is set to false upon initialisation.

Hash function as a random oracle. We model the hash function $H(\cdot)$ as a random oracle $\mathcal{F}_{\mathrm{RO}}$. It accepts queries of the form $(\mathsf{compute}, x)$ and (verify, x, y). For the first type of query, assuming x was never queried before, a value y is sampled from $\{0,1\}^{\kappa}$ and it is entered to a table T_H. If x was queried before the pair (x, y) is recovered from T_H. The value y is provided as an answer. For the second type of query, a membership test is performed on the table. Alert parties are allowed to ask one query per round of the type compute and unlimited queries of the type verify. The adversary $\mathcal{A}$ is given a bounded number of compute queries per round and no verify queries (the adversary can easily simulate those locally). The bound for the adversary is determined as follows. Whenever a corrupted party is activated the bound is increased by 1; whenever a query is asked the bound is decreased by 1 (it is not necessary that the specific corrupted party makes the query).

The diffusion functionality. Message passing and round bookkeeping is maintained by this functionality (note that transaction diffusion is maintained by an independent functionality which prohibits the adversarial manipulation). A round variable `r` is initialized to 0. For each party a string denoted by Receive() is maintained and the party is allowed to fetch the contents of its corresponding Receive() at any time. The functionality records all messages of the form $(\mathsf{Diffuse}, m)$ it receives from the parties. Completion of a round for a party is indicated by sending a special message ($\mathsf{RoundComplete}$). The adversary $\mathcal{A}$ is allowed to receive all the currently recorded $\mathsf{Diffuse}$ messages at any time and messages to the Receive() strings as desired. The round is completed when the adversary submits its ($\mathsf{RoundComplete}$) message. In such case, the functionality inspects the contents of all Receive() strings and includes any messages m that were diffused by the parties Δ rounds ago but not contributed by the adversary to the Receive() tapes (in this way guaranteeing message delivery up to Δ rounds). It also flushes any diffuse records that are placed in the Receive() string of all parties. The variable `r` is then incremented and a new round begins.

Dynamic participation. In order to describe the protocol execution in a more realistic fashion, following the treatment in [BGK+18], we classify protocol participants into different types in Table 1.

	Basic types of *honest* parties	
Resource	**Resource unavailable**	**Resource available**
random oracle	*stalled*	*operational*
network	*offline*	*online*
clock	*time-unaware*	*time-aware*
synchronized state	*desynchronized*	*synchronized*

Table 1: A classification of protocol participants.

Consider a party P at a given point of the protocol execution, we say (i) P is *operational* if P is registered with the random oracle, and *stalled* otherwise; (ii) P is *online* if P is registered with the network, and *offline* otherwise; (iii) P is *time-aware* if P is registered with the global clock, and *time-unaware* otherwise; and (iv) P is *synchronized* if P has been participanted in the protocol for sufficiently long time and it holds a chain that shares a common prefix with other *synchronized* parties, and *desynchronized* otherwise.

We define *alert* parties based on the classification above. In short, alert parties are those who have access to all the resources and are synchronized. They are the core set of parties to carry out the protocol.

$$\textit{alert} \stackrel{\text{def}}{=} \textit{operational} \wedge \textit{online} \wedge \textit{time-aware} \wedge \textit{synchronized}.$$

In addition, we define *active* parties to depict parties including all that are alert, adversarial or time-unaware.

$$\textit{active} \stackrel{\text{def}}{=} \textit{alert} \vee \textit{adversarial} \vee \textit{time-unaware}$$

The dynamic bounded-delay setting. Given the functionalities as described above observe that contrary to prior formalizations, the adversary can choose the termination of the round thus deciding on the spot how many honest parties were activated adaptively. In each round, the number of alert parties (cf. Section 2.1) that are active in the protocol is denoted by h_r and is equal to the total number of parties that have submitted the (RoundComplete) indicator to the diffusion functionality and have their internal flag `isAlert` set to true. Determining $h_{\mathtt{r}}$ can only be done by examining the view of all alert parties and is not a quantity that is accessible to any of the honest parties individually. The number of "corrupt" parties controlled by $\mathcal{A}$ in a round `r` is similarly denoted by $t_{\mathtt{r}}$.

Parties, when activated, are able to read their input tape Input() and communication tape Receive() from the diffusion functionality. If a party finds that its `isAlert` flag is false, it enters a "bootstrapping" mode where it will diffuse a discovery message and synchronize. When the synchronization phase terminates, the party will set its `isAlert` flag to true and after this point it will be counted among the alert parties. An honest party goes "offline" when it misses a round, i.e., the adversary issues a (RoundComplete) but that party misses the opportunity to complete its computation. To record this action, whenever this happens we assume that the party's `isAlert` flag is set to false (in particular this means that a party is aware that it went offline; note, however, that the party does not need to report it to anyone). Also observe that parties are unaware of the set of activated parties. As in previous works (e.g., [GKL15]), we assume, without loss of generality, that each honest party has the same computational power.

Properties of protocols We consider a "standalone" execution without any auxiliary information. Hence we use $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$ to denote the random variable ensemble that is a concatenation of the views of all parties ever activated (party P's view after the execution is denoted by $\text{VIEW}^{\mathsf{P}}_{\Pi,\mathcal{A},\mathcal{Z}}$). In our theorems we will be concerned with *properties* of protocols Π. Such properties will be defined as predicates over $\text{VIEW}_{\Pi,\mathcal{A},\mathcal{Z}}$ with a small probability of error in κ as well as in a parameter k that is selected from $\{1,\ldots,\kappa\}$.

C Proofs, Algorithms and Examples Omitted in the Main Body

C.1 Proofs Omitted in Section 3

Proof of Theorem 1. Suppose a protocol Π implements non-trivial (φ, B)-fair-order serialization with $t \geq (2\varphi-1)h$. Fix φ, h and $t \geq (2\varphi-1)h$ and consider two executions E_1, E_2 on Π with honest profiles $\mathcal{R}_1^{\mathcal{H}}$ and $\mathcal{R}_2^{\mathcal{H}}$ respectively. Suppose there are m transactions. Profile $\mathcal{R}_1^{\mathcal{H}} = \mathsf{R}_{11}, \mathsf{R}_{12}, \ldots, \mathsf{R}_{1h}$ reports $\mathsf{R}_{1j}(\mathtt{tx}_i) = i$ for all $i \in [m]$ when $j \leq \lceil \varphi h \rceil$ and $\mathsf{R}_{1j}(\mathtt{tx}_i) = m + 1 - i$ for all $i \in [m]$ when $j \geq \lceil \varphi h \rceil + 1$. I.e., φ fraction of $\mathcal{R}_1^{\mathcal{H}}$ share exactly the same order while the rest share exactly the same reversed order. Similarly, in profile $\mathcal{R}_2^{\mathcal{H}}$, $\mathsf{R}_{2j}(\mathtt{tx}_i) = i$ for all $i \in [m]$ when $j \leq \lfloor (1-\varphi)h \rfloor$ and $\mathsf{R}_{2j}(\mathtt{tx}_i) = m + 1 - i$ for all $j \geq \lfloor (1-\varphi)h \rfloor + 1$.

Since $t \geq (2\varphi - 1)h$, we have two identical profiles $\mathcal{R}_1 = \langle \mathcal{R}_1^{\mathcal{H}}, \mathcal{R}_1^{\mathcal{A}} \rangle$ and $\mathcal{R}_2 = \langle \mathcal{R}_2^{\mathcal{H}}, \mathcal{R}_2^{\mathcal{A}} \rangle$. These profiles can be constructed by, e.g., letting $(2\varphi - 1)h$ profiles in $\mathcal{R}_1^{\mathcal{A}}$ report an order the same as R_{1h}, letting $(2\varphi-1)h$ profiles in $\mathcal{R}_2^{\mathcal{A}}$ report an order the same as R_{2h} and letting the rest of $\mathcal{R}_1^{\mathcal{A}}, \mathcal{R}_2^{\mathcal{A}}$ report the same order.

Notice that in $\mathcal{R}_1^{\mathcal{H}}$ it holds that $\forall i \in [m], \forall j > i, \mathtt{tx}_i \prec^{\varphi} \mathtt{tx}_j$; and in $\mathcal{R}_2^{\mathcal{H}}$ we have $\forall i \in [m], \forall j < i, \mathtt{tx}_i \prec^{\varphi} \mathtt{tx}_j$. And since $\mathcal{R}_1$ and $\mathcal{R}_2$ are identical we have $F(\mathcal{R}_1) = F(\mathcal{R}_2)$.

Now, consider an order σ with $\sigma(\mathtt{tx}_i) = 1$ and $\sigma(\mathtt{tx}_j) = m$. If $i > j$ then it implies for input $\mathcal{R}_1^{\mathcal{H}}$ in E_1, $\mathtt{tx}_j \prec^{\varphi} \mathtt{tx}_i \implies \sigma(tx_j) - \sigma(\mathtt{tx}_i) = m - 1$. Conversely, when $i < j$ it implies for input $\mathcal{R}_2^{\mathcal{H}}$ in E_2, $\mathtt{tx}_j \prec^{\varphi} \mathtt{tx}_i \implies \sigma(tx_j) - \sigma(\mathtt{tx}_i) = m - 1$. Since the similar argument works for all possible orderings, it implies that either $B(\mathcal{R}_1^{\mathcal{H}}, \varphi, \mathtt{tx}_i, \mathtt{tx}_j) \geq m - 1$ or $B(\mathcal{R}_2^{\mathcal{H}}, \varphi, \mathtt{tx}_i, \mathtt{tx}_j) \geq m - 1$, which contradicts the fact that Π implements non-trivial fair-order serialization. □

Proof of Theorem 2. Consider a cycle $v_1 v_2 \cdots v_k v_1$ of size k. By transitivity of $\prec$, for any party P_i such that $v_k \prec_i v_1$, there exists a j such that $v_j \not\prec_i v_{j+1}$. Since $v_k v_1 \in E(G_{\mathcal{R},\varphi})$, we have $v_k \prec^{\varphi} v_1$. Thus, there exists a j such that $v_j \not\prec_i v_{j+1}$ for at least $\varphi/(k-1)$ fraction of parties. But because $v_j v_{j+1} \in E(G_{\mathcal{R},\varphi})$, it holds

$$\frac{\varphi}{k-1} \leq 1 - \varphi \implies k \geq \frac{1}{1-\varphi}.$$

□

Proof of Theorem 3. Since the transaction space on R to construct G and G' are the same, $V(G) = V(G')$. It suffices to show that $E(G) \subseteq E(G')$. Suppose towards a contradiction, there is an edge $(\mathtt{tx}, \mathtt{tx}') \in E(G)$ but $(\mathtt{tx}, \mathtt{tx}') \notin E(G')$. This implies $\mathtt{tx} \prec^{\varphi}_{\mathcal{R}^{\mathcal{H}}} \mathtt{tx}'$; and in the profile set $\mathcal{R}$ that generates G', less than half of the profiles report $\mathtt{tx} \prec \mathtt{tx}'$. However since the adversary can append at most $(2\varphi-1)h - 1$ profiles, we get the following contradiction: $\varphi h < (1-\varphi)h + (2\varphi - 1)h - 1 = \varphi h - 1$. □

Proof of Theorem 6. Fix an order fairness parameter φ. Towards a contradiction, suppose there exist a function B such that $\exists \mathcal{R}, \forall (\mathtt{tx}, \mathtt{tx}'), B(\mathcal{R}, \varphi, \mathtt{tx}, \mathtt{tx}') < \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \mathtt{DBW}(\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}'))$. We consider a protocol Π that implements (φ, B)-fair-order serialization and an execution E of Π. Suppose in E an honest party outputs $\sigma = F(\langle \mathcal{R}, \mathcal{R}^{\mathcal{A}} \rangle)$. Let $\mathcal{R}^{\mathcal{A}}$ be profiles such that the $(\langle \mathcal{R}, \mathcal{R}^{\mathcal{A}} \rangle)$-dependency-graph G has a SCC G_{SCC} containing $\mathtt{tx}, \mathtt{tx}'$ such that $\mathtt{DBW}(G_{\mathsf{SCC}}) = \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \mathtt{DBW}(\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}'))$. For any vertex ordering σ on G, we have $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') \geq \mathtt{DBW}(G_{\mathsf{SCC}}) = \max_{G \in \mathbb{G}_{\mathcal{R},\varphi}} \mathtt{DBW}(\mathsf{SCC}(G, \mathtt{tx}, \mathtt{tx}')) > B(\mathcal{R}, \varphi, \mathtt{tx}, \mathtt{tx}')$. This contradicts the fact that Π implements (φ, B)-fair-order serialization. □

Proof of Theorem 7. We first prove this theorem for 3 transactions and $1/2 < \varphi \leq 2/3$, then we show how to extend this to an arbitrary number of transactions and φ.

Consider two executions E_1, E_2 with transaction profiles illustrated as in Figure 2. Note that they are different only in the view of party P_3 after time $t + L$. First, consider execution E_2. We have $\mathtt{tx}_2 \prec^{\varphi} \mathtt{tx}_3$, $\mathtt{tx}_3 \prec^{\varphi} \mathtt{tx}_1$ and $\mathtt{tx}_2 \prec^{\varphi} \mathtt{tx}_1$. Hence at the end of the execution it should output $\mathtt{tx}_2 \prec \mathtt{tx}_3 \prec \mathtt{tx}_1$. Now, consider this execution up to time $t + L$. Since $\mathtt{tx}_3$ appears in all parties' transaction profiles and L is the liveness parameter, $\mathtt{tx}_3$ should appear in the output at time $t + L$. In addition, due to consistency parties should output $\mathtt{tx}_2 \prec \mathtt{tx}_3$ at time $t + L$ in E_2. Since the two executions are indistinguishable up to time $t + L$, in E_1 it should also output $\mathtt{tx}_2 \prec \mathtt{tx}_3$ at time $t + L$. I.e., eventually E_1 will output $\mathtt{tx}_2 \prec \mathtt{tx}_3 \prec \mathtt{tx}_1$ by consistency. However, since $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$ in E_1, this order gives the worst bandwidth 2.

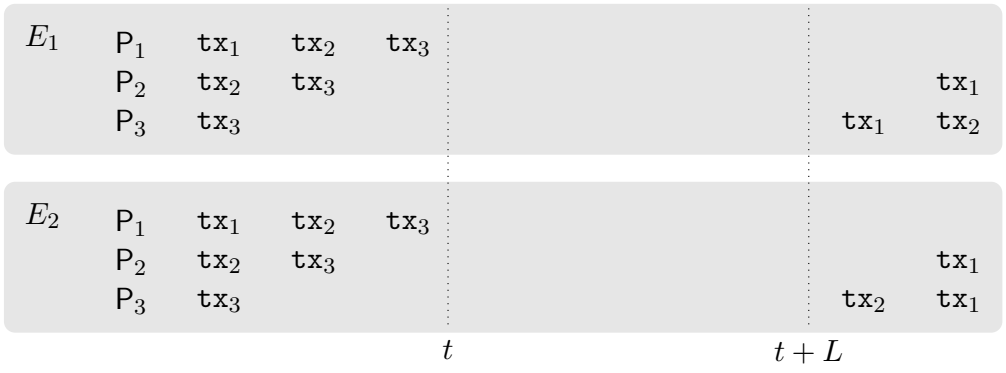


Figure 2: Transaction profile illustration.

By replacing $\mathtt{tx}_2, \mathtt{tx}_3$ with a set of transactions, this proof can be extended to profiles with arbitrary n transactions and output with the worst bandwidth $n - 1$. And, for $\varphi > 2/3$ it is still feasible to construct similar profiles — in the first execution the dependency graph is a directed cycle and by swapping the order of the last two transactions in the last profile, the second dependency graph is acyclic. □

Proof of Theorem 8. We first show that the subscripts of P_i are consecutive. Suppose — towards a contradiction — there is no partition V_i $(1 < i < n)$. Since all transactions are Δ_{tx}-disseminated, and the partition is a $(F_{\mathsf{ts}}^{\mathsf{min}}, \Delta_{\text{tx}})$-partition, for any two vertices $u \in V_j$ $(j \leq i - 1)$ and $v \in V_k$ $(k \geq i + 1)$, it holds that $F_{\mathsf{ts}}^{\mathsf{min}}(u) + \Delta_{\text{tx}} < F_{\mathsf{ts}}^{\mathsf{min}}(v)$. I.e., in all profiles, transaction $\mathtt{tx}_u$ precedes $\mathtt{tx}_v$; hence edge $(u, v) \notin E_{\mathsf{SCC}}$; this contradicts the fact that G_{SCC} is strongly connected.

Consider a vertex ordering σ on G_{SCC} that orders vertices by a non-decreasing order on $F_{\mathsf{ts}}^{\mathsf{min}}$. It holds that $(\forall u \in V_j, v \in V_k)\ j < k \implies \sigma(u) < \sigma(v)$. Since no back edge exists between two non-adjacent partitions, we have

$$\mathtt{DBW}(G_{\mathsf{SCC}}) \leq \mathtt{DBW}(G_{\mathsf{SCC}}, \sigma) \leq \max_{i \in [n-1]} \left|V_i\right| + \left|V_{i+1}\right| \leq 2 \max \left|V_i\right|.$$

□

Proof of Theorem 9. Suppose a $(F_{\mathsf{ts}}^{\mathsf{min}}, \Delta_{\text{tx}})$-partition produces $n \geq 4$ partitions (otherwise it is trivial) and consider a back edge (u, v). The proof of theorem 8 shows that either u, v are in the same partition, or u is in the next partition of v.

If u, v are in the same partition (i.e., $u, v \in V_i$), then $F_{\mathsf{ts}}(u), F_{\mathsf{ts}}(v)$ are either in partition V_i or V_{i+1}. Note that for all vertices $v \in V_j$, $F_{\mathsf{ts}}(v)$ is either in V_j or V_{j+1}. We have $\sigma(v) - \sigma(u) \leq |V_{i-1}| + |V_i| + |V_{i+1}| \leq 3 \max |V_i|$.

If u, v are in the adjacent partition (i.e., $u \in V_i$ and $v \in V_{i+1}$), since $\sigma(v) \geq \sigma(u)$, $F_{\mathsf{ts}}(u), F_{\mathsf{ts}}(v) \in V_{i+1}$; otherwise if $F_{\mathsf{ts}}(v) \in V_i$ or $F_{\mathsf{ts}}(u) \in V_{i+2}$ it cannot form the order. Similarly, we have $\sigma(v) - \sigma(u) \leq |V_{i+1}| + |V_{i+2}| \leq 2 \max |V_i|$. □

C.2 Hardness of DirectedBandwidth over Oriented Graphs

In this section we consider the hardness of DirectedBandwidth over oriented graphs. With respect to the Bandwidth problem the following result of Dubey, Feige, and Unger is of interest.

Theorem 12 ([DFU11]). *It is* NP*-hard to approximate* Bandwidth *within a ratio of c, for any constant $c > 0$.*

We provide the same result for DirectedBandwidth over oriented graphs.

Theorem 13. *It is* NP*-hard to approximate* DirectedBandwidth *over oriented graphs within a ratio of c, for any constant $c > 0$.*

Proof. It suffices to show how to approximate Bandwidth within a constant, assuming we can approximate DirectedBandwidth over any oriented graph. Let $G = (V, E)$ be a simple connected graph with bandwidth b. We define an oriented graph $G' = (V \cup V', E')$, where $V' = \{u' : u \in V\}$ and $E' = \{uu' : u \in V\} \cup \{u'v : uv \in E\}$. We call each $u \in V$ the parent of u', V the parent nodes and V' the child nodes. Note that since G is undirected, for each $uv \in E$, both $u'v$ and $v'u$ are in E'. We claim that the bandwidth of G at most twice the bandwidth of G'.

We first observe that the bandwidth of G' is at most b. Indeed, let $\pi = (u_1, u_2, \ldots, u_n)$ be a permutation of V with bandwidth b in G. We construct a permutation π' of $V \cup V'$ with bandwidth b in G'. Set $q = \lfloor n/b \rfloor$ and $r = n \mod b$. For $k = 0, 1, \ldots, q-1$, let

$$\sigma_k = (u'_{qb+1}, \ldots, u'_{qb+b}, u_{qb+1}, \ldots, u_{qb+b}) \text{ and } \pi' = \sigma_0 \cdots \sigma_q,$$

where $\sigma_q = (u'_{qb+1}, \ldots, u'_{qb+r}, u_{qb+1}, \ldots, u_{qb+r})$. To verify that $\mathtt{DBW}(\pi', G') \leq b$, consider any edge $uv \in E$; edge uu' has length either b or $r < b$ and edge $u'v$ is either a forward edge or has the same length as uv under π.

Now let σ be a permutation of $V \cup V'$ with bandwidth b'. We can construct a permutation π for V by simply removing the child nodes. The bandwidth of π is at most $2b'$. It follows that if $b' \leq cb$, then we obtain a permutation of V with bandwidth $2cb$. In other words, if we can approximate DirectedBandwidth within ratio c, then we can approximate Bandwidth withing ratio $2c$. □

C.3 An Exact Algorithm for DirectedBandwidth

We briefly describe the exact algorithm for DirectedBandwidth in Algorithm 1. For more technical details, refer to [JKL+19].

The general idea of this algorithm follows the strategy of "kill what cause you trouble" — i.e., since only the back edges show effect in deciding directed bandwidth of a graph G, we could first guess the set of back edges R, then remove all forward edges (i.e., we get a directed acyclic graph $G' = (V_G, R)$) and convert it to a DAG-Bandwidth problem on G'. A DAG-Bandwidth problem asks to output a topological sort on a DAG D with the minimum bandwidth (note that this is the bandwidth of the undirected graph D' constructed from D, by connecting vertices u, v when at least one (u, v) or (v, u) exists in D). Jain *et al.* [JKL+19, Section 3] presents an algorithm that solves DAG-Bandwidth, following the idea from Cygan and Pilipczuk [CP08]. Roughly speaking, this algorithm consists of two phases Bucketing and Ordering. The Bucketing phase allocates a

set of vertices to a range of consecutive positions of a targeted size; the outcome of this step is a collection of bucketings containing a bucketing that is "consistent" with the final ordering. Then, the Ordering phase extracts a topological ordering from the buckets acquired in the previous phase.

Algorithm 1 DirectedBandwidth

Input: Graph $G = (V, E)$.

1: Set $\texttt{DBW}(G) = -\infty$
2: **for** $R \subset E$ **do**
3: Set $G' = (V, R)$
4: Solve DAG-Bandwidth and get $\texttt{DAG-BW}\ (G'), \sigma_{\texttt{DAG}}$. ◁ Run Bucketing and Ordering
5: **if** $\texttt{DAG-BW}\ (G') > \texttt{DBW}(G)$ **then**
6: $\texttt{DBW}(G) \leftarrow \texttt{DAG-BW}\ (G'), \sigma \leftarrow \sigma_{\texttt{DAG}}$
7: **end if**
8: **end for**

Output: $\texttt{DBW}(G), \sigma$.

DAG-Bandwidth can be solved in $\mathcal{O}^*(3^{|V|})$ time [CP08, JKL$^+$19]; since the total number of possible back edge set is $2^{|E|}$, Algorithm 1 runs in $\mathcal{O}^*(3^{|V|} \cdot 2^{|E|})$ time.

C.4 A Faster Algorithm for DirectedBandwidth over Dependency Graphs

Algorithm 1 solves DirectedBandwidth over general graphs. The most time-consuming factor lies at the stage that exhaustively iterates over subset of edges and sets them as forward ones; on oriented graphs, this incurs an exponent that is quadratic in terms of the number of edges.

While this blow-up arises from the hardness of the underlying problem, thus becomes impossible to improve in the worst case[8], we present here that for a long-lasting Condorcet cycle, its dependency graph is of good structure when considered with the (bounded) transaction throughput and its directed bandwidth is relatively small compared with the number of vertices (see Theorem 8), hence leads to a new algorithm which, while still runs in exponential time, reduces the exponent of the most time-consuming factor from quadratic to linear in the number of vertices.

As a high-level intuition of the structure of such type of graphs, consider a graph G of long-lasting Condorcet cycles and a vertex $v \in G$. Regarding all edges between v and v', if v' is a vertex that enters the system earlier (later resp.) than v and their diffusion window does not overlap, then due to the security of Taxis protocols, it is guaranteed that an edge (v', v) ((v, v') resp.) appears in the dependency graph. And we shall never set these edges as backward edges when applying a vertex ordering over the graph, as it incurs a directed bandwidth that is larger than the number of transactions diffused between v and v' (which is bounded by the transaction throughput) which contradicts Theorem 8.

The main ideal of this algorithm follows that in [JKL$^+$19] with two extra steps: (i) instead of iterating all possible subset of edges, we pre-select some edges and never consider setting them as backward edges in the final ordering; and (ii) regarding the subroutine of solving DAG-Bandwidth by calling the bucketing and ordering algorithm in [CP08, JKL$^+$19], we replace it with Saxe's algorithm [Sax80], an algorithm that decides if DAG-Bandwidth is smaller than a (fixed) parameter

[8]For instance, the network is congested and a large number of transactions enter the system at roughly the same time and consequently all edges in the dependency graph can have either orientation.

in polynomial time. We provide a detailed description in Algorithm 2.

Compared with Algorithm 1, this new algorithm requires two extra inputs. First, the median timestamp (in the K-time-window) of each transaction. This is represented as the a mapping F_{ts} from each vertex (transaction) to an integer (round number). And second, a parameter $\Delta^*_{\text{tx}} \in \mathbb{N}^+$ as the threshold to select edges. We require that $\Delta^*_{\text{tx}} \geq 3\Delta_{\text{tx}}$. Note that while in our model Δ_{tx} is a parameter that is unknown to parties, a rough estimation on Δ_{tx} can be made by observing the transaction diffusion pattern; and, overestimation does not hurt the correctness of this algorithm but only incurs some penalty on its running time.

Algorithm 2 DirectedBandwidth over $G_{\mathcal{R},\varphi}$

Input: $G = (V, E)$ the graph, $F_{\mathsf{ts}} : V \to \mathbb{N}^+$ a mapping and $\Delta^*_{\text{tx}} \in \mathbb{N}^+$.

Set $\texttt{DBW}(G) = -\infty$ and $E_{\mathsf{backward}} = E$
for $(u, v) \in E$ **do**
  if $F_{\mathsf{ts}}(v) - F_{\mathsf{ts}}(u) \geq \Delta^*_{\text{tx}}$ **then**
    $E_{\mathsf{backward}} \leftarrow E_{\mathsf{backward}} \backslash (u, v)$
  end if
end for
for $R \subset E_{\mathsf{backward}}$ **do**
  Set $G' = (V, R)$
  Solve DAG-Bandwidth and get $\texttt{DAG-BW}\ (G'), \sigma_{\texttt{DAG}}$. ◁ Call Saxe's algorithm.
  if $\texttt{DAG-BW}\ (G') > \texttt{DBW}(G)$ **then**
    $\texttt{DBW}(G) \leftarrow \texttt{DAG-BW}\ (G'), \sigma \leftarrow \sigma_{\texttt{DAG}}$
  end if
end for

Output: $\texttt{DBW}(G), \sigma$.

Correctness of Algorithm 2. It is obvious that for any graph G, as long as there exist bandwidth-optimal orderings such that all backward edges are in E_{backward}, Algorithm 2 will output one of them due to the correctness of Saxe's algorithm. Thus, correctness of Algorithm 2 follows Lemma 14.

Lemma 14. *Let $G = (V, E)$ be a strongly connected component in the transaction dependency graph and F_{ts} be a mapping from each $v \in V$ to the median timestamp in its K-time-window. There exists an ordering σ on G such that (i) $\texttt{DBW}(\sigma, G) = \texttt{DBW}(G)$; and (ii) $\forall u, v \in V$, $\sigma(u) < \sigma(v) \implies F_{\mathsf{ts}}(u) \leq F_{\mathsf{ts}}(v) + \Delta^*_{\text{tx}}$.*

Proof. Let σ' be a vertex ordering on G such that (i) $\texttt{DBW}(\sigma', G) = \texttt{DBW}(G)$; and (ii) $\exists u, v \in V$ such that $\sigma(u) < \sigma(v)$ and $F_{\mathsf{ts}}(u) > F_{\mathsf{ts}}(v) + \Delta^*_{\text{tx}}$. If no such σ' exists then the lemma holds. Otherwise we show that by swapping vertex pairs we can convert σ' to σ such that both items (i) (ii) in the lemma holds on σ.

Let E'_σ denote the set of backward edges in σ that violates item (ii) in the lemma. I.e.,

$$E'_\sigma = \{(v, u) \mid (v, u) \in E \wedge \sigma(u) < \sigma(v) \wedge F_{\mathsf{ts}}(u) > F_{\mathsf{ts}}(v) + \Delta^*_{\text{tx}}\}.$$

Let $d(E'_\sigma)$ denote the maximum distance among all edges in E' (i.e., $d(E'_\sigma) = \max_{(u,v)\in E'_\sigma}\{\sigma(v) - \sigma(u)\}$ or 0 if E'_σ is empty) and $m(E'_\sigma, d^*)$ denote the number of edges with distance d^*. Note that $d(E'_\sigma) \leq \texttt{DBW}(E'_\sigma, G)$. Obviously when there exist σ such that $\texttt{DBW}(\sigma, G) = \texttt{DBW}(G)$ and $d(E'_\sigma) = 0$ then the lemma holds.

Fix a bandwidth optimal ordering σ' such that $d(E'_{\sigma'}) = d^* > 0$. Let u, v be two vertices such that $(v, u) \in E'_{\sigma'}$ and $\sigma'(v) - \sigma'(u) = d(E'_{\sigma'})$. Consider a new ordering σ'' that differs from σ' by swapping the position of u, v. We show that $\texttt{DBW}(G, \sigma'') = \texttt{DBW}(G)$ and, either $d(E'_{\sigma''}) < d(E'_{\sigma'})$, or $d(E'_{\sigma''}) = d(E'_{\sigma'}) = d^*$ but $m(E'_{\sigma''}, d^*) \leq m(E'_{\sigma'}, d^*) - 1$.

Since only the position of u, v is swapped, we consider edges with endpoints either in u or v. Consider v first. Let v' be a vertex other than u, v and consider the following three cases.

- When $\sigma''(v') < \sigma''(v)$, the distance of backward edge (v, v') in σ'' is less than that in σ'.
- When $\sigma''(v) < \sigma''(v') < \sigma'(v)$, if (v', v) is a backward edge in σ'' then its distance is strictly less than d^*.
- When $\sigma'(v) < \sigma''(v')$, note that it holds $(v, v') \in E$; otherwise, since $\Delta^*_{\text{tx}} \leq 3\Delta_{\text{tx}}$, a transaction diffusion time window can last for at most Δ_{tx} rounds, and in a typical execution the adversary can only manipulate the median timestamp to some time within the diffusion window, we have: when $\texttt{tx}_{v'} \prec \texttt{tx}_v$ it implies that $\texttt{tx}_{v'} \prec \texttt{tx}_u$ (i.e., (v', u) is a backward edge in σ'), contradicting the fact that (v, u) is the backward edge of largest distance. Thus, (v, v') remains as forward edges and has no effect on $E'(\sigma'')$.

A similar argument can be made for u due to symmetry. Since all newly introduced backward edges have distance $d' < d^* \leq \texttt{DBW}(G)$, we get $\texttt{DBW}(\sigma'', G) = \texttt{DBW}(G)$ and if $m(E'_{\sigma'}, d^*) = 1$ then no backward edge of distance no less than d^* exists in σ'' hence $d(E'_{\sigma''}) < d(E'_{\sigma'})$; if $m(E'_{\sigma'}, d^*) \geq 1$ then since (v, u) is no longer a backward edge in σ'' we have $d(E'_{\sigma''}) = d(E'_{\sigma'})$ and $m(E'_{\sigma''}, d^*) \leq m(E'_{\sigma'}, d^*) - 1$.

By iteratively applying the above swapping strategy, for any bandwidth-optimal ordering σ' we shall convert it into another bandwidth-optimal ordering σ that $d(E'_{\sigma}) = 0$ hence it satisfies both item (i) (ii) in the lemma. □

Time complexity of Algorithm 2. We define throughput t as the number of concurrent transactions entered into the system within a Δ^*_{tx} time window, thus in the pre-processing stage in Algorithm 2, for each vertex v at most t edges with target in v will be preserved in E_{backward}. I.e., after this stage we have $|E_{\mathsf{backward}}| \leq |V|t$. In addition, due to Theorem 8 the we have $\texttt{DBW}(G) < t$, thus Saxe's algorithm runs in $f(t) \cdot |V|^t$ time where $f(t)$ depends only on t (For more details, refer to [Sax80]). Combining these two parts together we get $f(t) \cdot |V|^t \cdot 2^{|V|t}$ (note that $t \ll |V|$).

Remark 2. The size of E_{backward} still grows exponentially. However, note that the input to Saxe's algorithm should be a DAG. I.e., when iterating over the subset of E_{backward}, we shall not call Saxe's algorithm in many scenarios. For instance, let C be a cycle in an SCC. If none of its edges are in the subset of E_{backward} then we can skip to the next iteration without calling Saxe's algorithm (because C preserves in the result graph thus it must not be a DAG). While this has no effect on the asymptotic result, we remark it here for some intuition on the practicality of Algorithm 2.

C.5 Upper Bound on Worst Case Directed Bandwidth

Define $\texttt{DBW}(n) = \max_G B(G)$, where G over the set of oriented graphs on n nodes. The following theorem concerns Zarankiewicz's problem.

Theorem 15 ([Bol78]; Theorem 2.5, p. 311). *For all natural numbers $2 \leq a \leq n$, any bipartite graph with parts of size n and more than $(a-1)^{1/a}n^{2-1/a} + (a-1)n/2$ edges contains at least one $a \times a$ clique.*

We call a pair (S, T) of two disjoint sets of nodes of a digraph a *one-way* pair, if there are no edges from a node in T to a node in S. The relation to this problem is captured in the following observation.

Proposition 16. *Consider a digraph $G = (V, E)$ on n nodes. If there is a permutation on the nodes of G with no back edges of length more than B, then there is a one-way pair (S, T) with $|S| = |T| = \lfloor (n - B)/2 \rfloor$. Conversely, if there is a one-way pair (S, T) with $|S| = |T| = n - B$, then there is a permutation on the nodes of G with no back edges of length at least B.*

Proof. Let S be the set of the first $\lfloor (n - B)/2 \rfloor$ nodes of π and T the set of the last so many nodes. □

Theorem 17. *Let $\mathbb{G}_n$ denote the set of all oriented graphs with n vertices. It holds that*

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

Proof. Note that for both bounds we may restrict attention to tournament graphs. To prove the lower bound, it suffices to show that there exists a tournament on n nodes such that for any permutation there are back edges of length at least $B = n - 4\log(n)$. We use a probabilistic argument, drawing intuition from Zarankiewicz's problem.

Set $k = \lfloor (n - B)/2 \rfloor = \lfloor 2 \log(n) \rfloor$. For any two nodes u, v, choose (u, v) or (v, u) with probability $1/2$ each. There are $\binom{n}{k}\binom{n-k}{k}$ pairs of disjoint sets of size k. Let X be the number of such pairs of sets that have no edges from T to S. Then, using $\binom{n}{k} \leq (en/k)^k$,

$$\mathbb{E}[X] = \binom{n}{k}\binom{n-k}{k} 2^{-k^2} \leq \Big(\frac{en}{k}\Big)^k \Big(\frac{e(n-k)}{k}\Big)^k 2^{-k^2} < 1.$$

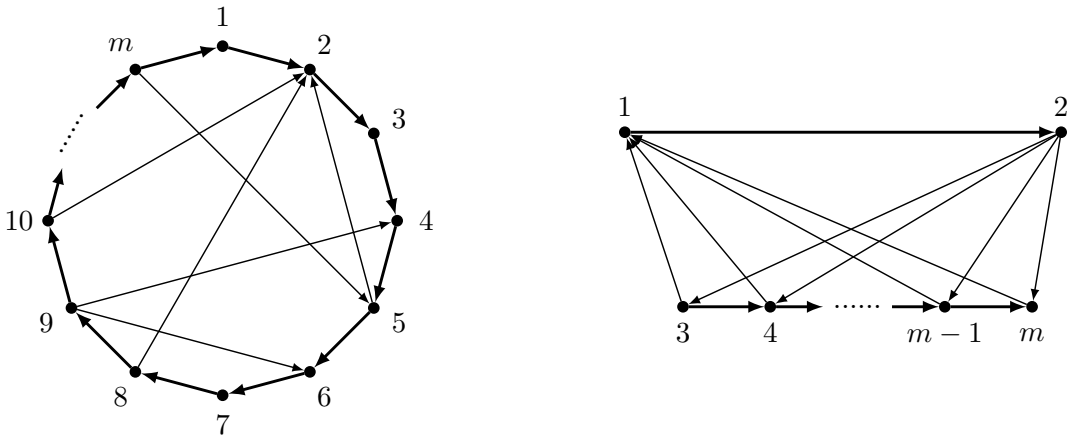
The last inequality holds because for the chosen value of k and $n > 16$, $en < k2^{k/2}$ and so $(en/k)^k / 2^{k^2/2} < 1$. It follows that there is a tournament with $X = 0$. By the first part of the proposition above, a permutation with $B \leq n - 4\log(n)$ does not exist for this tournament.

For the upper bound, by the second part of the proposition it suffices to show that for any tournament $G = ([n], E)$ there is a one-way pair (S, T) with $|S| = |T| = a$, where $a = \lceil \log(n)/2 \rceil$. To that end, construct a bipartite graph $G = (U, V, D)$ with $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$ and $E = \{(u_i, v_j) : (i, j) \in E\}$. Note that if $S \times T$ is a clique of this graph, then (S, T) is a one-way pair. By the second part of the proposition and Theorem 15, it suffices to show that $(a-1)^{1/a} n^{2-1/a} + (a-1)n \leq \binom{n}{2}$. This indeed holds for the chosen value of a. □

C.6 Examples of Comparison with Existing Protocols

We first show that the output of Aequitas and Themis yields $\sigma(\mathtt{tx}) - \sigma(\mathtt{tx}') = m - 1$ for $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ on some input profiles with m transactions. Consider a dependency graph $G = (V, E)$ where $V = 1, 2, \ldots, m$. Its edge set $E = E_1 \cup E_2$ contains two subsets, a cycle $E_1 = \{(1, 2), (2, 3), \ldots, (m - 1, m), (m, 1)\}$, and a set of some edges $E_2 = \{(i, j)\}$ such that $2 \leq j \leq i - 2$ (E_2 can contain an arbitrary number of edges). Due to the profile construction rules proposed by McGarvey [McG53] (see Section 3.2), we can always find profiles whose dependency graph is exactly G . Also see Figure 3(a) for an illustration of G. Now we assign the labels to each transaction such that $\mathit{label}(\mathtt{tx}_2) < \mathit{label}(\mathtt{tx}_i) < \mathit{label}(\mathtt{tx}_1)$ for all $\mathtt{tx}_i$ other than $\mathtt{tx}_1, \mathtt{tx}_2$, Aequitas will output $\sigma_{\mathsf{Aequitas}} = \mathtt{tx}_2 \prec \ldots \prec \mathtt{tx}_1$ which yields $\sigma_{\mathsf{Aequitas}}(\mathtt{tx}_1) - \sigma_{\mathsf{Aequitas}}(\mathtt{tx}_2) = m - 1$ for $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$. Note that following the Hamiltonian-cycle-based rule, Themis will output the same order as Aequitas (note that the rotation on $\sigma_{\mathsf{Aequitas}}$ does not matter since there is always a back edge from the last transaction to the first). Next, consider an output $\sigma = \mathtt{tx}_1 \prec \mathtt{tx}_n \prec \mathtt{tx}_{n-1} \prec \ldots \prec \mathtt{tx}_2$ that satisfies Definition 5, it holds that $\sigma(\mathtt{tx}_i) - \sigma(\mathtt{tx}_j) \leq 1$ for all $\mathtt{tx}_i \prec^{\varphi} \mathtt{tx}_j$.

Regarding pompe, we consider three profiles $\mathsf{R}_1 = \mathtt{tx}_1, \mathtt{tx}_2, \mathtt{tx}_3, \mathtt{tx}_4, \ldots, \mathtt{tx}_m$, $\mathsf{R}_2 = \mathtt{tx}_2, \mathtt{tx}_3, \mathtt{tx}_4, \ldots, \mathtt{tx}_m, \mathtt{tx}_1$ and $\mathsf{R}_3 = \mathtt{tx}_3, \mathtt{tx}_4, \ldots, \mathtt{tx}_m, \mathtt{tx}_1, \mathtt{tx}_2$. I.e., we replace $\mathtt{tx}_3$ in the 3-transaction



(a) Aequitas (b) pompe

Figure 3: Illustration of dependency graphs.

Condorcet cycle with $m-2$ transactions, following the same order $\mathtt{tx}_3, \mathtt{tx}_4, \ldots, \mathtt{tx}_m$ in all profiles. See Figure 3(b) for an illustration of its dependency graph G. Suppose σ is a profile that satisfies Definition 5 and

$$\sigma = \mathtt{tx}_3 \prec \ldots \prec \mathtt{tx}_{\lceil n/3 \rceil} \prec \mathtt{tx}_1 \prec \mathtt{tx}_{\lceil n/3 \rceil + 1} \prec \ldots \prec \mathtt{tx}_{\lceil 2n/3 \rceil} \prec \mathtt{tx}_2 \prec \mathtt{tx}_{\lceil 2n/3 \rceil + 1} \prec \ldots \prec \mathtt{tx}_m.$$

We have $\sigma(\mathtt{tx}_i) - \sigma(\mathtt{tx}_j) \leq \lceil m/3 \rceil$ for all $\mathtt{tx}_i \prec^{\varphi} \mathtt{tx}_j$. However, we consider a timestamp assignment F_{ts} on R_1, R_2 and R_3 such that pompe would output σ_{pompe} under F_{ts} and $\sigma_{\mathsf{pompe}}(\mathtt{tx}_1) - \sigma_{\mathsf{pompe}}(\mathtt{tx}_2) = m-1$ for $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$ Specifically, let $F_{\mathsf{ts}}(\mathtt{tx}_i, \mathsf{R}) = t$, i.e., in all profiles it reports $\mathtt{tx}_i$ is received at time t for all $3 \leq i \leq m$. Then, let $F_{\mathsf{ts}}(\mathtt{tx}_1, \mathsf{R}_1) = t-2$, $F_{\mathsf{ts}}(\mathtt{tx}_1, \mathsf{R}_2) = t+1$, $F_{\mathsf{ts}}(\mathtt{tx}_1, \mathsf{R}_3) = t+1$ and $F_{\mathsf{ts}}(\mathtt{tx}_2, \mathsf{R}_1) = t-1$, $F_{\mathsf{ts}}(\mathtt{tx}_2, \mathsf{R}_2) = t-1$, $F_{\mathsf{ts}}(\mathtt{tx}_2, \mathsf{R}_3) = t+2$, we have $\mathsf{med}(\mathtt{tx}_1) = t+1$ and $\mathsf{med}(\mathtt{tx}_2) = t-1$, which yields an order with $\sigma_{\mathsf{pompe}}(tx_1) = m$ and $\sigma_{\mathsf{pompe}}(tx_2) = 1$.

C.7 Taxis protocols

Validity check of chains. Note that this part is the same for Taxis and $\mathsf{Taxis}_{\mathsf{WL}}$. A ledger block $\mathsf{LB} = \langle ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}} \rangle$ is valid iff. it satisfies predicate $\mathsf{validBlock}^{T_{\mathsf{PB}}}(\mathsf{LB}) = H(\mathsf{LB}) < T_{\mathsf{LB}} \wedge ctr < 2^{32}$. A ledger block chain $\mathcal{C}$ is valid iff. $\forall \mathsf{LB} \in \mathcal{C}$ it holds $\mathsf{validBlock}^{T_{\mathsf{PB}}}(\mathsf{LB}) \wedge (h^* = H(\mathsf{LB})) \wedge (\mathtt{r} < \mathtt{r}^*)$ where $h^*, \mathtt{r}^*$ are the corresponding block header information reported by LB's next block.

Protocol 4 $\mathsf{IsValidChain}(\mathcal{C}, \mathtt{r})$

$r' \leftarrow \mathtt{r}, \mathsf{LB} \leftarrow \mathrm{head}(\mathcal{C})$
$\langle ctr, h, h', r, st_{\mathsf{LB}}, st_{\mathsf{PB}} \rangle \leftarrow \mathsf{LB}$
$h^* \leftarrow H(ctr, h, h', r, st_{\mathsf{LB}}, st_{\mathsf{PB}})$
while $\mathcal{C} \neq \varepsilon$ **do**
    $\langle ctr, h, h', r, st_{\mathsf{LB}}, st_{\mathsf{PB}} \rangle \leftarrow \mathsf{LB}$
    $validLB, validPB \leftarrow \mathsf{true}$
    ▷ Check validity of ledger block
    if $(h^* \neq H(\mathsf{LB})) \vee (h^* \geq T) \wedge (ctr \geq 2^{32}) \vee (r \geq r')$ **then**

$validLB \leftarrow \mathsf{false}$
end if
▷ Check validity of profile blocks in LB
for $\mathsf{PB} \in \mathsf{LB}$ **do**
$\mathsf{LB}' \leftarrow$ the last ledger block s.t. $\mathsf{TS}(()\mathsf{LB}) \leq \mathsf{TS}(()\mathsf{PB}) - k$
if $\mathsf{TS}(\mathsf{PB}) \geq \mathsf{TS}(\mathsf{LB}) \vee h' \neq H(\mathsf{LB}') \vee \mathsf{TS}(\mathsf{PB}) \geq \mathsf{TS}(\mathsf{LB}') + R \vee H(\mathsf{PB}) \geq T_{\mathsf{PB}}$ **then**
$validPB \leftarrow \mathsf{false}$
end if
end for
if $validLB \wedge validPB = \mathsf{true}$ **then**
$r' \leftarrow r, h^* \leftarrow h$
Remove the rightmost block in $\mathcal{C}$
$\mathsf{LB} \leftarrow \mathrm{head}(\mathcal{C})$
else
return false
end if
end while
return true

The mining procedure of Taxis. This mining protocol is generally the same as Protocol 1. The only difference is that parties will append the local timestamp for each transaction.

Protocol 5 $\mathsf{Taxis}_{\mathsf{WL}}$-$\mathsf{MiningProcedure}(\mathtt{r}, \kappa)$

Fetch information from $\mathcal{F}_{\mathsf{Diffuse}}^{\Delta}$ and $\mathcal{F}_{\mathsf{Diffuse},\mathtt{tx}}$ and get new chains $(\mathcal{C}_1, \ldots, \mathcal{C}_M)$, new transactions $(\mathtt{tx}_1, \ldots, \mathtt{tx}_i)$ and new profile blocks $(\mathsf{PB}_1, \ldots, \mathsf{PB}_j)$
Set $\mathtt{buffer}_{\mathsf{PB}} \leftarrow \mathtt{buffer}_{\mathsf{PB}} \| (\mathsf{PB}_1, \ldots, \mathsf{PB}_j)$
Set $\mathtt{localProfile} \leftarrow \mathtt{localProfile} \| (\langle \mathtt{tx}_1, \mathtt{r} \rangle, \ldots, \langle \mathtt{tx}_i, \mathtt{r} \rangle)$
$\mathcal{C}_{\mathrm{loc}} \leftarrow \mathsf{maxvalid}(\mathcal{C}_{\mathrm{loc}}, \mathcal{C}_1, \ldots, \mathcal{C}_M)$
$\mathtt{LBContent} \leftarrow$ (valid) profile blocks in $\mathtt{buffer}_{\mathsf{PB}}$ that are not mined in $\mathcal{C}_{\mathrm{loc}}$
$\mathtt{PBContent} \leftarrow \mathtt{localProfile}$ ◁ After removing a prefix in settled part
$h' \leftarrow$ hash value of block head$(\mathcal{C}_{\mathrm{loc}}^{\lceil k})$ ◁ Find rightmost blocks in the settled ledger
$h \leftarrow$ hash value of block head$(\mathcal{C}_{\mathrm{loc}})$
$u \leftarrow \mathsf{H}(ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}})$
if $u < T_{\mathsf{LB}}$ **then**
Set $\mathsf{LB} \leftarrow \langle ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}} \rangle$
$\mathcal{C}_{\mathrm{loc}} \leftarrow \mathcal{C}_{\mathrm{loc}} \| \mathsf{LB}$
Diffuse $\mathcal{C}_{\mathrm{loc}}, \mathtt{LBContent}$
end if
if $[u]^{\mathsf{R}} < T_{\mathsf{PB}}$ **then**
Set $\mathsf{PB} \leftarrow \langle ctr, h, h', \mathtt{r}, \mathtt{st}_{\mathsf{LB}}, \mathtt{st}_{\mathsf{PB}} \rangle$
Diffuse $\mathsf{PB}, \mathtt{PBContent}$
end if
$ctr \leftarrow ctr + 1$

D Security Analysis

In this section we provide a security analysis of the Taxis protocol. We show that, in a (γ, s)-respecting environment, when appropriately parameterized, the ledger $\mathcal{L}$ in Taxis satisfies three security properties—consistency, liveness and order fairness.

The first two properties have been introduced in Section 2.1; regarding the order fairness property, it requires that the protocol implements $(\varphi, \texttt{TDBW})$-fair-order serialization — i.e., in the settled part of the final ledger, all the transactions will follow the order defined by Definition 7.

D.1 Notations and Preliminary Results

We adopt "Bitcoin backbone protocol" analytical toolset (cf. [GKL17, GKL20]) as the main framework to carry out the security analysis of Taxis. In this section we revisit some notations and results in the bounded-delay network from [GKL20].

Notably, we extend the notation of $D_{\mathtt{r}}$ that models the total difficulty that honest parties can acquire during round $\mathtt{r}$ to $D_{\mathtt{r}}^{\mathtt{tx}}$ ($D_{\mathtt{r}}^{\mathtt{tx},\mathtt{tx}'}$, resp.), which captures the amount of difficulty that φ fraction of honest parties associated with $\mathtt{tx}$ ($\mathtt{tx}, \mathtt{tx}'$, resp.) can get. Specifically, $D_{\mathtt{r}}^{\mathtt{tx}}$ argues for the median timestamp of $\mathtt{tx}$ and $D_{\mathtt{r}}^{\mathtt{tx},\mathtt{tx}'}$ is for the fraction of profiles that report $\mathtt{tx}' \prec \mathtt{tx}$ in the time window of $\mathtt{tx}$ (note that $\mathtt{tx}, \mathtt{tx}'$ and $\mathtt{tx}', \mathtt{tx}$ are two different pairs, details see analysis below). We equip the typical execution (see Definition 14(b)) with lower bounds on $D_{\mathtt{r}}^{\varphi,\mathtt{tx}}$ ($D_{\mathtt{r}}^{\mathtt{tx}\prec^{\varphi}\mathtt{tx}'}$, resp.) as well. On the other hand, we quantify the length of a K-time-window (cf. Equation (2)) and introduce a new restriction (cf. Condition (C3)) that lower-bounds φ so as to get desired properties on profile blocks.

Our probability space is over all executions of length at most some polynomial in κ and λ and we denote by $\mathbf{Pr}$ the probability measure of this space. Furthermore, let $\mathcal{E}$ be a random variable taking values on this space and with a distribution induced by the random coins of all entities (adversary, environment, parties) and the random oracle.

If at round $\mathtt{r}$ exactly h alert parties query the oracle with target T, the probability of at least one of them will succeed is $f(T,h) = 1-(1-pT)^h \leq pTh$, where $p = 1/2^{\kappa}$. During round $\mathtt{r}$, alert parties might be querying the random oracle for various targets. We denote by $T_{\mathtt{r}}^{\min}$ and $T_{\mathtt{r}}^{\max}$ the minimum and maximum of those targets. Moreover, the initial target T_0 implies in our model an initial estimate of the number of alert parties h_0. For convenience, we denote $f_0 = f(T_0, h_0)$ and simply refer to it as f.

Definition 10. – *Round* $\mathtt{r}$ *is* good *if* $f/2\gamma^2 \leq ph_{\mathtt{r}}T_{\mathtt{r}}^{\min}$ *and* $ph_rT_{\mathtt{r}}^{\max} \leq (1+\delta)\gamma^2 f$.
- *Round* $\mathtt{r}$ *is a* target recalculation point *of a chain* $\mathcal{C}$*, if* $\mathcal{C}$ *has a block with timestamp* $\mathtt{r}$ *and height a multiple of* m.
- *A target recalculation point* $\mathtt{r}$ *is* good *if the target* T *for the next block satisfies* $f/2\gamma \leq ph_{\mathtt{r}}T \leq (1+\delta)\gamma f$.
- *A chain is* good *if all its target-recalculation points are good.*
- *A chain is stale if for some* $u \geq \ell + 2\Delta$ *it does not contain an honest block with timestamp* $v \geq u - \ell - 2\Delta$.
- *The blocks between two consecutive target recalculation points* u *and* v *on a chain* $\mathcal{C}$ *are an epoch of* $\mathcal{C}$*. The duration of the epoch is* $u - v$.

Definition 11. *For a round* $\mathtt{r}$*, let:*
- GoodChains($\mathtt{r}$) $\triangleq$ *"For all* $u \leq \mathtt{r}$*, every chain in* $\mathcal{S}_u$ *is good."*
- GoodRound($\mathtt{r}$) $\triangleq$ *"All rounds* $u \leq \mathtt{r}$ *are good."*
- NoStaleChains($\mathtt{r}$) $\triangleq$ *"For all* $u \leq \mathtt{r}$*, there are no stale chains in* $\mathcal{S}_u$*."*

- $\textsc{Accurate}(\mathtt{r}) \triangleq$ *"For all $u \leq \mathtt{r}$, all chains in $\mathcal{S}_u$ are accurate."*
- $\textsc{Duration}(\mathtt{r}) \triangleq$ *"For all $u < \mathtt{r}$ and $\mathcal{C} \in \mathcal{S}_u$, the duration Λ of any epoch in $\mathcal{C}$ satisfies $\frac{1}{2(1+\delta)\gamma^2} \cdot \frac{m}{f} \leq \Lambda \leq 2(1+\delta)\gamma^2 \cdot \frac{m}{f}$."*
- $\textsc{CommonPrefix}(\mathtt{r}) \triangleq$ *"For all $u < \mathtt{r}$ and $\mathcal{C}, \mathcal{C}' \in \mathcal{S}_u$, head($\mathcal{C} \cap \mathcal{C}'$) was created after round $u - \ell - 2\Delta$."*

Random Variables. We are interested in quantifying the difficulty acquired by honest parties at round $\mathtt{r}$. Thus, following [GKL20], we define three random variables $D_{\mathtt{r}}$, $Y_{\mathtt{r}}$ and $Q_{\mathtt{r}}$ with respect to round $\mathtt{r}$ to analyze the total difficulties that are acquired by *all* alert parties. Especially, for a specific transaction $\mathtt{tx}$, we associate it with φ fraction of alert parties and wish to extract the lower bound on total difficulties that these alert parties [9] can acquire during a round $\mathtt{r}$.

- $D_{\mathtt{r}}$ is the sum of the difficulties of all blocks computed by honest parties at round $\mathtt{r}$. Additionally, fix φ fraction of honest parties with respect to transaction $\mathtt{tx}$ (transaction pair $\mathtt{tx}, \mathtt{tx}'$ resp.); let $D_{\mathtt{r}}^{\mathtt{tx}}$ ($D_{\mathtt{r}}^{\mathtt{tx},\mathtt{tx}'}$ resp.) denote the sum of the difficulties of all blocks computed by these parties at round $\mathtt{r}$.
- $Y_{\mathtt{r}}$ is the maximum difficulty among all blocks computed by honest parties at round $\mathtt{r}$.
- $Q_{\mathtt{r}}$ is equal to $Y_{\mathtt{r}}$ when $D_u = 0$ for all $r < u < r + \Delta$ and 0 otherwise.

A round $\mathtt{r}$ is called *successful* if $D_{\mathtt{r}} > 0$ and *isolated successful* when $Q_{\mathtt{r}} > 0$. For a set of rounds S we write $h(S) = \sum_{\mathtt{r} \in S} h_{\mathtt{r}}$ and similarly $t(S)$, $D(S)$, $D^{\varphi,\mathtt{tx}}(S)$, $Y(S)$, $Q(S)$.

Regarding the adversary, while he may query the random oracle for arbitrarily low target and obtain blocks with arbitrarily high difficulty, we wish to upper-bound the difficulty he can during a set of J queries. Consider a set of consecutive adversarial queries J and associate it with the target of the first query (this target is denoted by $T(J)$). We define $A(J)$ and $B(J)$ to be equal to the sum of the difficulties of all blocks computed by the adversary during queries in J for target at least $T(J)/\tau$ and $T(J)$, respectively.

Let $\mathcal{E}_{\mathtt{r}-1}$ fix the execution just before round $\mathtt{r}$. In particular, a value $E_{\mathtt{r}-1}$ of $\mathcal{E}_{\mathtt{r}-1}$ determines the adversarial strategy and so determines the targets against which every party will query the oracle at round $\mathtt{r}$ and the number of parties $h_{\mathtt{r}}$ and $t_{\mathtt{r}}$, but it does not determine $D_{\mathtt{r}}$ or $Q_{\mathtt{r}}$. For an adversarial query j we will write $\mathcal{E}_{j-1}$ for the execution just before this query.

Mathematical facts. Fact 1 captures some facts within a (γ, s)-respecting environment.

Fact 1. Let U be a set of at most s consecutive rounds in a (γ, s)-respecting environment and $S \subseteq U$.

(a) For any $h \in \{h_r : r \in U\}$, $\frac{h}{\gamma} < \frac{h(S)}{|S|} \leq \gamma h$.

(b) $h(U) \leq (1 + \frac{\gamma|U \setminus S|}{|S|} h(S))$.

(c) $|S| \sum_{r \in S} (ph_r)^2 \leq \gamma (ph(S))^2$.

We are also interested in some martingale inequalities (cf. [GKL17]).

Definition 12. *A sequence of random variables $(X_0, X_1, \ldots)$ is a martingale with respect to the sequence $(Y_0, Y_1, \ldots)$, if, for all $n \geq 0$, X_n is determined by $Y_0, \ldots, Y_n$ and $\mathbb{E}[X_{n+1} \mid Y_0, \ldots, Y_n] = X_n$.*

[9] Note that $1/2 < \varphi \leq 1$, thus in the worst case there exist exponentially many φ fraction subsets of alert parties. Nonetheless, since the number of transactions are polynomially bounded with respect to the total running time, we show that the bad event—difficulties acquired are below the lower bound—is still negligibly small (see the proof of Theorem 20 for details).

Theorem 18. *Let $(X_0, X_1, \ldots)$ be a martingale with respect to the sequence $(Y_0, Y_1, \ldots)$. Suppose an event G implies*

$$X_k - X_{k-1} \leq b \text{ (for all } k\text{) and } V = \sum_k \mathbf{Var}[X_k - X_{k-1} \mid Y_1, \ldots, Y_{k-1}] \leq v.$$

Then, for non-negative n and t,

$$\mathbf{Pr}[X_n \geq X_0 + t \wedge G] \leq \exp\Big\{ - \frac{t^2}{2v + 2bt/3} \Big\}.$$

Protocol parameters and conditions. We summaries all the protocol parameters in Table 2.

Parameter	Description
h_r	The number of alert parties mining in round r.
t_r	The number of corrupted parties mining in round r.
δ	Advantage of honest parties ($t_r < (1-\delta)h_r$ for all r).
f	The probability at least one alert party out of n_0 computes a block for target T_0.
m	The length of an epoch in number of blocks.
Δ	Network delay in rounds.
κ	Security parameter; length of the hash function output.
(γ, s)	Respecting environment parameter.
ϵ	Quality of concentration of random variables.
λ	Related to the properties of the protocol.
φ	Order-fairness parameter (cf. Definition 7).
R	The recency parameter.
K	The length of a profile block window in number of rounds.
$\Delta_{\mathsf{timeout}}$	The timeout parameter to process cycles that span for a long time.

Table 2: Summary of Taxis parameters.

In order to get desired convergence, we consider a sufficiently long consecutive sequence of at least

$$\ell = \frac{4(1+3\epsilon)}{\epsilon^2 f[1-(1+\delta)\gamma^2 f]^{\Delta+1}} \cdot \max\{\Delta, \tau\} \cdot \gamma^3 \cdot \lambda \tag{1}$$

rounds. Moreover, with the purpose of get desired properties on profile blocks, we would be considering the length of a K-time-window (measured in number of rounds), the recency parameter and the fallback timeout parameterto be at least

$$K = (\frac{\gamma}{\epsilon} + 1)(\ell + 3\Delta), R = 3\ell + 6\Delta \text{ and } \Delta_{\mathsf{timeout}} = 3\ell + 7\Delta + 5\Delta_{\mathrm{tx}}. \tag{2}$$

The protocol parameters in Table 2 should satisfy certain conditions in order to make our analysis viable. First, we require that the length of an epoch is lower-bounded w.r.t. ℓ.

$$2\ell + 6\Delta \leq \frac{\epsilon m}{2(1+\delta)\gamma^3 f} \tag{C1}$$

Then, we also require that network delay Δ and party fluctuation ratio γ is well upper-bounded.

$$[1-(1+\delta)\gamma^2 f]^{\Delta} \geq 1 - \epsilon \text{ and } \epsilon \leq \delta/8 \leq 1/8 \tag{C2}$$

Finally, we require that given ϵ, δ, the order fairness parameter φ should be large enough to ensure that φ fraction of honest parties still substitute more than half of total number of parties (and the advantage is large enough to aborb the error bounded by 5ϵ).

$$(1-7\epsilon)\varphi > 1 - \delta/2 \tag{C3}$$

Typical executions. We define the notion of *typical* executions following [GKL17, GKL20]. The idea here is that given a certain execution E, we compare the actual progress and the expected progress that parties will make under the success probabilities. If the difference and variance are reasonably small, and no bad events (see Definition 13) about the underlying hash function happen, we declare E *typical*.

Definition 13. *An insertion occurs when, given a chain $\mathcal{C}$ with two consecutive blocks $\mathcal{B}$ and $\mathcal{B}'$, a block $\mathcal{B}^*$ created after $\mathcal{B}'$ is such that $\mathcal{B}, \mathcal{B}^*, \mathcal{B}'$ form three consecutive blocks of a valid chain. A copy occurs if the same block exists in two different positions. A prediction occurs when a block extends one with later creation time.*

Definition 14 (Typical execution). *An execution E is typical if the following hold.*

(a) For any set S of at least ℓ consecutive good rounds,

$$(1-\epsilon)[1-(1+\delta)\gamma^2 f]^{\Delta} ph(S) < Q(S) \leq D(S) < (1+\epsilon)ph(S)$$

(b) Fix any transaction $\mathtt{tx}$ and preference pair $\mathtt{tx}, \mathtt{tx}'$ and any set S of at least 2ℓ consecutive good rounds,

$$D^{\mathtt{tx}}(S) > (1-\epsilon)\varphi ph(S) \text{ and } D^{\mathtt{tx},\mathtt{tx}'}(S) > (1-\epsilon)\varphi ph(S)$$

(c) For any set J of consecutive adversarial queries and $\alpha(J) = 2(\frac{1}{\epsilon} + \frac{1}{3})\lambda/T(J)$,

$$A(J) < p|J| + \max\{\epsilon p|J|, \tau\alpha(J)\} \text{ and } B(J) < p|J| + \max\{\epsilon p|J|, \alpha(J)\}$$

(d) No insertions, no copies, and no predictions occurred in E.

Following [GKL20], we extract the quantitative relations between honest and adversarial hashing power during some consecutive rounds with length at least ℓ in Lemma 19. Additionally, in order to extract the accumulated difficulty of profile blocks, we conclude the relationship between total difficulty acquired by all parties ($D(S) + A(J)$) and their hashing power in Lemma 19(d).

Lemma 19. *Consider a typical execution in a (γ, s)-respecting environment. Let $S = \{r : u \leq r \leq v\}$ be a set of at least ℓ consecutive good rounds and J the set of adversarial queries in $U = \{r : u - \Delta \leq r \leq v + \Delta\}$.*

(a) $(1+\epsilon)p|J| \leq Q(S) \leq D(U) < (1+5\epsilon)Q(S)$.

(b) $T(J)A(J) < \epsilon m/4(1+\delta)$ or $A(J) < (1+\epsilon)p|J|$; $\tau T(J)B(J) < \epsilon m/4(1+\delta)$ or $B(J) < (1+\epsilon)p|J|$.

(c) If w is a good round such that $|w - r| \leq s$ for any $r \in S$, then $Q(S) > (1-\epsilon)[1-(1+\delta)\gamma^2 f]^{\Delta}|S|pn_w/\gamma$. If in addition $T(J) \geq T_w^{\min}$, then $A(J) < (1-\delta+3\epsilon)Q(S)$.

(d) If w is a good round such that $|w-r| \leq s$ for any $r \in S$ and $T(J) \geq T_w^{\min}$, then $D(S)+A(J') < (1+\epsilon)p(h(S)+|J'|)$ where J' denotes the set of adversarial queries in S.

Proof. For the proof of items (a)(b)(c), we refer to [GKL20]. Regarding Lemma 19(d), note that we are under the same condition as that in Lemma 19(c), we have

$$A(J') < A(J) < (1-\delta+3\epsilon)Q(S) \leq (1-\delta+3\epsilon)(1+\epsilon)ph(S) < (1-\delta+3\epsilon)(1+\epsilon)(1-\delta)p|J'| < (1+\epsilon)p|J'|.$$

The second inequality follows Lemma 19(c); the third one comes from Definition 14(a); the last inequality is a consequence of Condition (C2).

By combining the upper bound on $A(J')$ with $D(S) < (1+\epsilon)ph(S)$ (which comes from the definitions) we get the desired inequality. □

We claim that almost all executions that are polynomially bounded in κ and λ are typical.

Theorem 20. *Assuming the ITM system $(\mathcal{Z}, C)$ runs for L steps, the probability of the event "$\mathcal{E}$ is not typical" is bounded by* $\mathsf{poly}(L)(e^{-\lambda} + 2^{-\kappa})$.

Proof. Definition 14 extends the definition of typical executions in [GKL20] by adding item (b). Hence regarding the proof of item (a)(c)(d) in Definition 14, we refer to [GKL20]. In this proof we only consider Definition 14(b) and the eventual error probabilities.

Fix a transaction $\mathtt{tx}$ and its associated φ fraction alert parties. Fix a set of $R \geq 2\ell$ consecutive rounds $S = s_1, s_2 \ldots, s_R$ and let $h_j, j \in [R]$ denote the number of honest queries make during round s_j. We work on per query that φ fraction of honest parties make during S. Let J denote the queries (made by φ fraction of honest parties) in S, $\nu = |J| \geq \varphi h(S)$, and Z_i the difficulty of any block obtained from query i. Consider the sequence of random variables

$$X_0 = 0; X_k = \sum_{i\in[k]} Z_i - \sum_{i\in[k]} \mathbb{E}[Z_i|\mathcal{E}_{i-1}], k \in [\nu].$$

This is a martingale sequence with respect to sequence $(\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_\nu)$ in that

$$\mathbb{E}[X_k|\mathcal{E}_{k-1}] = \mathbb{E}[Z_k - \mathbb{E}[Z_k|\mathcal{E}_{k-1}]|\mathcal{E}_{k-1}] + \mathbb{E}[X_{k-1}|\mathcal{E}_{k-1}] = X_{k-1}.$$

The last equation follows the linearity of conditional expectation and the fact that X_{k-1} is a deterministic function with respect to $\mathcal{E}_{k-1}$.

Regarding the details relevant to Theorem 18, we pick t as

$$\epsilon \sum_{i\in[\nu]} \mathbb{E}[Z_i|\mathcal{E}_{i-1} = E_{i-1}] \geq \epsilon\varphi \sum_{j\in[R]} ph_j = \epsilon\varphi ph(S) \stackrel{\text{def}}{=} t$$

and consider an execution satisfying G_t. Fix $i \in [\nu]$, let j be the round s_j that query i belongs to. Let

$$Z_i - \mathbb{E}[Z_i|\mathcal{E}_{i-1} = E_{i-1}] \leq \frac{1}{T_j^{\min}} = \frac{ph_j}{ph_j T_j^{\min}} \leq \frac{\gamma ph(S)}{ph_j T_j^{\min} R} \leq \frac{\gamma ph(S)}{fR/(2\gamma^2)} = \frac{2\gamma^3 t}{\epsilon\varphi fR} \stackrel{\text{def}}{=} b$$

and we see that the event G implies $X_k - X_{k-1} \leq b$. To get the bound on V, note that

$$\mathbf{Var}(X_k - X_{k-1}|\mathcal{E}_{k-1}) = \mathbb{E}[(Z_i - \mathbb{E}[Z_i|\mathcal{E}_{i-1}])^2|\mathcal{E}_{k-1}] \leq \mathbb{E}[Z_i^2|\mathcal{E}_{k-1}].$$

Hence, based on the independence of random variables as well as Fact 1(c), we pick v as

$$\begin{aligned}\sum_{k\in[\nu]} \mathbb{E}[Z_i^2|\mathcal{E}_{k-1} = E_{k-1}] &\leq \varphi \sum_{j\in[R]} \sum_{i\in[h_j]} \frac{1}{T_i^2} \cdot pT_i = \varphi \sum_{j\in[R]} \frac{ph_j}{T_j^{\min}} \\ &= \varphi \sum_{j\in[R]} \frac{(ph_j)^2}{ph_j T_j^{\min}} \leq \frac{2\varphi\gamma^2}{f} \sum_{j\in[R]} (ph_j)^2 \leq \frac{2\varphi\gamma^3}{fR} \cdot (ph(S))^2 \leq \frac{2\gamma^3 t^2}{\epsilon^2\varphi fR} \stackrel{\text{def}}{=} v.\end{aligned}$$

In view of these bounds (note that $bt = \epsilon v$), by Theorem 18, we have

$$\mathbf{Pr}[-X_\nu \geq t \wedge G_t] \leq \exp\Big\{-\frac{t^2}{2v(1+\frac{\epsilon}{3})}\Big\} \leq \exp\Big\{-\frac{\epsilon^2\varphi f R}{4\gamma^3(1+\frac{\epsilon}{3})}\Big\} \leq e^{-\lambda}.$$

Note that the last inequality comes from Condition (C1) and φ vanishes since Condition Condition (C3) implies that $\varphi > 1/2$.

The above arguments also applies, when we fix a transaction preference pair $\mathtt{tx}, \mathtt{tx}'$. Now consider the execution that runs for L steps. The total number of transactions is bounded by $\mathsf{poly}(L)$; and the total number of preference pairs is bounded by $\mathsf{poly}(L)$ as well. Regarding the preference pairs notice that they grow quadratically with the number of transactions hence remain a polynomial function of the running time. We get

$$\mathbf{Pr}[\text{failure of Definition 14(b)}] = 1 - (1 - e^{-\lambda})^{\mathsf{poly}(L)} \leq \mathsf{poly}(L)e^{-\lambda}. \tag{3}$$

Combining Equation (3) and those error probabilities in [GKL20], we get asymptotically the same result. I.e., the probability that "$\mathcal{E}$ is not typical" is bounded by $\mathsf{poly}(L)(e^{-\lambda} + 2^{-\kappa})$. □

Security properties of the Taxis blockchain from [GKL20]. We briefly describe the blockchain security properties common prefix and chain quality, which applies to the blockchain (i.e., chain of ledger blocks) in Taxis. Notably, common prefix is defined by pruning blocks according to their timestamps.

- **Common Prefix** (CP) with parameter $k \in \mathbb{N}$. For any two alert parties $\mathsf{P}_1, \mathsf{P}_2$ holding chains $\mathcal{C}_1, \mathcal{C}_2$ at rounds r_1, r_2, with $r_1 \leq r_2$, it holds that $\mathcal{C}_1^{\lceil k} \preccurlyeq \mathcal{C}_2$.
- **Chain Quality** (CQ) with parameter $k \in \mathbb{N}$ and $\mu \in [0, 1]$. For any party P with chain $\mathcal{C}$ and any segment of that chain of difficulty d such that the first block of the segment was computed at least k rounds earlier than the last block, the blocks the honest parties have contributed in the segment have total difficulty at least $\mu \cdot d$.

While we introduce 2-for-1 PoW to produce profile blocks and collect them later, the Taxis blockchain is still of the same structure as Bitcoin backbone protocol [GKL15, GKL17], thus achieving blockchain properties remains the same as the analysis in [GKL20]. Hence, for these two properties, we refer to [GKL20]. When appropriately parameterized, blockchain in Taxis satisfies these two security properties except with probability negligibly small in κ. In addition, the same argument stands for those predicates defined in Definition 11.

Theorem 21 (Security of Taxis blockchain). *In a typical execution and (γ, s)-respecting environment, if Conditions* (C1), (C2) *are satisfied, then the Taxis blockchain satisfies common prefix with parameter $k = \ell + 2\Delta$ and chain quality with parameters $\ell + 2\Delta$ and $\delta - 3\epsilon$. Meantime, all predicates of Definition 11 hold.*

D.2 Properties of Profile Blocks and Median Timestamp

In this section we focus on the properties of profile blocks collected in the Taxis blockchain. We first show that thanks to the freshness introduced by block hash in the settled blockchain and the recency parameter R, the adversary cannot start to mine profile blocks containing a specific transaction $\mathtt{tx}$ much earlier than the beginning of $\mathtt{tx}$'s K-time-window. Note that Lemma 22 holds also for those adversarial transactions.

Lemma 22. *In a typical execution and (γ, s)-respecting environment, consider a transaction* $\mathtt{tx}$*. Suppose r is the timestamp of the earliest ledger block that contains a profile block with* $\mathtt{tx}$*. Then, the adversary mined profile blocks containing* $\mathtt{tx}$ *no earlier than $r - 4(\ell + 2\Delta)$.*

Proof. For the sake of a contradiction, suppose the adversary mined such a block at round $r' < r - 4(\ell + 2\Delta)$. We show that this implies the happening of prediction (cf. Definition 13) or the violation of either CommonPrefix or Accurate, contradicting the execution being typical.

Let PB denote the first profile block with $\mathtt{tx}$. Since PB is considered "recent" with respect to the ledger block with timestamp r, PB should point to a ledger block LB with timestamp $\mathsf{TS}(\mathsf{LB}) \geq r - 3\ell + 6\Delta$. We write the creation time of LB as r_{LB}. We have $r_{\mathsf{LB}} \geq r - 4(\ell + 2\Delta)$; otherwise, it violates Accurate in that LB's timestamp deviates from its creation time for more than $\ell + 2\Delta$ rounds. However, since the creation time of LB is $r'_{\mathsf{LB}} \geq r - 4(\ell + 2\Delta)$, $\mathcal{A}$ start to mine PB at round $r' < r - 4(\ell + 2\Delta)$ using LB's block hash implies a happening of prediction because $\mathcal{A}$ successfully extends a block before its creation time. This concludes our proof. □

Next, we show that, in a typical execution, the profile blocks (weighted in terms of their difficulty) that φ fraction of honest parties (φ satisfies Condition (C3)) can produce is proportional to their relative hashing power, except with some well-bounded disadvantage 5ϵ. Note that this result directly implies φ fraction of honest parties will accumulate more than half of the total difficulty (i.e., more difficulty than the coalition of the rest of honest parties and the adversary).

Lemma 23. *In a typical execution and (γ, s)-respecting environment, assuming all the conditions are satisfied, consider a transaction* $\mathtt{tx}$*. Let r be the timestamp of the earliest ledger block containing* $\mathtt{tx}$*, and d the total difficulty of profile blocks containing* $\mathtt{tx}$ *which are included in a ledger block with timestamp r' such that $r \leq r' \leq r + K$. We have (i) φ-fraction of honest parties mined profile blocks with total difficulty at least $d/2$; and (ii) for any transaction* $\mathtt{tx}'$*, φ-fraction of honest parties mined profile blocks of* $\mathtt{tx}', \mathtt{tx}$ *with total difficulty at least $d/2$.*

Proof. Let $\mathcal{C}$ denote a chain held by an honest party at round $r + K + (\ell + 2\Delta)$. Let $S_0 = \{u : r \leq u \leq r + K\}$ denote the valid timestamp set for ledger blocks that include timestamp blocks with $\mathtt{tx}$. Let LB be the last block produced by honest parties before round $r + K$ and denote its timestamp by r_{LB}. Since $\mathcal{C}$ will become stale if there is no honest block from $r_{\mathsf{LB}} + K - (\ell + 2\Delta)$ for $\ell + 2\Delta$ rounds, we get that $r + K - (\ell + 2\Delta) < r_{\mathsf{LB}} \leq r + K$.

Let $S_1 = \{u : r + (\ell + 2\Delta) \leq u \leq r_{\mathsf{LB}} - \Delta\}$ and $S_2 = \{u : r - 4(\ell + 2\Delta) \leq u \leq r_{\mathsf{LB}} + \ell + 2\Delta\}$. S_1 is the time interval for all honest parties to mine profile blocks with $\mathtt{tx}$; and S_2 is for the adversary. The lower bound of S_1 is derived from the fact that every honest party will learn the earliest ledger block after $(\ell + 2\Delta)$ rounds after its timestamp, and the upper bound is because all timestamp blocks will take up to Δ rounds to diffuse to all honest parties. The lower bound of S_2 is acquired due to the unpredictability discussed in Lemma 22. Regarding the upper bound of S_2, it is achieved by considering the first honest block LB' after $r_{\mathsf{LB}} + K$, which will have timestamp $r_{\mathsf{LB}'} < r_{\mathsf{LB}} + \ell + 2\Delta$ (otherwise it violates NoStaleChains). The adversary cannot produce blocks in S_0 after $r_{\mathsf{LB}'}$ as it can no longer revert LB', so all the subsequent timestamp blocks produced after $r_{\mathsf{LB}'}$ are invalid w.r.t. the current chain.

Let J denote the adversarial queries associated with S_2. We first consider item (i). In order to prove that φ fraction of honest parties can produce at least half of timestamp block difficulty, it suffices to show that $D^{\mathtt{tx}}(S_1) > d/2$.

We show that the number of alert parties in S_2 is at most 4ϵ more than that in S_1.

$$h(S_2) \leq (1 + \frac{\gamma|S_2 \backslash S_1|}{|S_1|})h(S_1) \leq \left(1 + \frac{\gamma(5\ell + 10\Delta)}{K - (\ell + 3\Delta)}\right)h(S_1) < (1 + 5\epsilon)h(S_1).$$

The first inequality comes from Fact 1(b); the second one follows the discussion above; the last one holds since by Equation (2) we get $|S_1| \geq K - (\ell + 3\Delta) = \gamma(\ell + 3\Delta)/\epsilon$.

Next, note that by combining Condition (C1) with Equation 2 we get $|S_2| \leq K + 5(\ell + 2\Delta) \leq (\gamma/\epsilon + 6)(\ell + 3\Delta) < 2\gamma(\ell + 3\Delta)/\epsilon \leq \frac{1}{2(1+\delta)\gamma^2} \cdot \frac{m}{f}$. This implies that during S_2, the blockchain will evolve at most two epochs, which satisfies the pre-condition of Lemma 19(d). We get

$$\begin{aligned} D^{\mathtt{tx}} &\geq (1-\epsilon)\varphi ph(S_1) > (1-6\epsilon)\varphi ph(S_2) > (1-6\epsilon)\frac{\varphi}{2-\delta}p[h(S_2)+|J|] \\ &> (1-7\epsilon)\frac{\varphi}{2-\delta}[D(S_2)+A(J)] > (1-7\epsilon)\frac{\varphi}{2-\delta}[D(S_1)+A(J)] > \frac{1}{2}[D(S_1)+A(J)] = \frac{d}{2}. \end{aligned}$$

The first inequality comes from the typical execution (cf. Definition 14); the second one is achieved by substituting $h(S_1)$ with $h(S_2)$; the next inequality follows the honest majority setting; and the forth one is by applying Lemma 19(d); the last inequality holds due to Condition (C3).

Regarding item (ii), we associate a transaction pair $\mathtt{tx}', \mathtt{tx}$ with a set of φ fraction honest parties and get the same result. □

Lemma 23 implies that, first, the median timestamp $\mathsf{med}(\mathtt{tx})$ of $\mathtt{tx}$ in the K-time-window is within the Δ_{tx} rounds that $\mathtt{tx}$ is disseminated to all honest parties; and second, if $\mathtt{tx}' \prec^{\varphi} \mathtt{tx}$, then in the dependency graph G in an honest party's view after both $\mathtt{tx}, \mathtt{tx}'$ are in TXPool, there will be an edge $(\mathtt{tx}', \mathtt{tx})$ in G.

Lemma 24. *In a typical execution and (γ, s)-respecting environment, consider a Δ_{tx}-disseminated transaction $\mathtt{tx}$. Let r denote the least round that $\mathtt{tx}$ is received by at least one honest party. The weighted median timestamp associated with $\mathtt{tx}$ in its K-time-window satisfies $r \leq \mathsf{med}(\mathtt{tx}) < r + \Delta_{\mathrm{tx}}$.*

Proof. Suppose $\mathsf{med}(\mathtt{tx}) < r$ or $\mathsf{med}(\mathtt{tx}) \geq r + \Delta_{\mathrm{tx}}$, which means that there exist profile blocks with more than half of the total difficulty that report a timestamp earlier than r or no earlier than $r + \Delta_{\mathrm{tx}}$. This contradicts the fact that the difficulty of profile blocks containing $\mathtt{tx}$ and reporting a timestamp t of $\mathtt{tx}$ such that $r \leq t < r + \Delta_{\mathrm{tx}}$ constitutes at least half of the total difficulty, according to Lemma 23. □

Lemma 25. *In a typical execution and (γ, s)-respecting environment, consider two transactions $\mathtt{tx}, \mathtt{tx}'$ such that $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$. Let r be the time such that both the K-time-window of $\mathtt{tx}$ and $\mathtt{tx}'$ are in TXPool, and let $G = (V, E)$ denote the dependency graph at round r in an honest party's view. It holds that $(\mathtt{tx}, \mathtt{tx}') \in E$.*

Proof. Lemma 23 implies that in the K-time-window of both $\mathtt{tx}$ and $\mathtt{tx}'$, majority of the profile blocks (counted by their accumulated difficulty) will report $\mathtt{tx} \prec \mathtt{tx}'$. At a round that $\mathtt{tx}' \in \mathsf{TXPool}$, an edge $(\mathtt{tx}, \mathtt{tx}')$ will be added to G. Moreover, at a round that $\mathtt{tx} \in \mathsf{TXPool}$, it will not add edge $(\mathtt{tx}', \mathtt{tx})$ to G or remove $(\mathtt{tx}, \mathtt{tx}')$ if it exists. This concludes our proof. □

Lemma 26. *In a typical execution and (γ, s)-respecting environment, if a transaction $\mathtt{tx}$ is associated with a K-time-window starting at time r, then for every transaction $\mathtt{tx}'$ such that $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ is false (either $\mathtt{tx}' \prec^{\varphi} \mathtt{tx}$ or incomparable), $\mathtt{tx}'$ is associated with a K-time-window starting at time $r' \leq r + 3\ell + 7\Delta + 2\Delta_{\mathrm{tx}}$.*

Proof. Suppose t is the earliest round such that $\mathtt{tx}$ is received by at least one honest party. Since all transactions are Δ_{tx}-disseminated, for every $\mathtt{tx}'$ such that $\mathtt{tx} \prec^{\varphi} \mathtt{tx}'$ is false, $\mathtt{tx}'$ is received by at least one honest party no later than $t + \Delta_{\mathrm{tx}}$ (otherwise $\mathtt{tx}'$ must be at a later position in all profiles).

We have $r \geq t - (\ell + 2\Delta)$, since a ledger block with timestamp r should be learnt by all honest party before round $r + \ell + 2\Delta$. Next, we consider a $\mathtt{tx}'$ such that $\mathtt{tx}' \prec^{\varphi} \mathtt{tx}$ and its K-time-window starting at time r'. $\mathtt{tx}'$ will be received by all honest parties no later than $r + 2\Delta_{\text{tx}}$, since it is Δ_{tx}-disseminated. Moreover, due to chain quality, there will be at least one profile block PB containing $\mathtt{tx}$ produced at time before $r + 2\Delta_{\text{tx}} + \ell + 2\Delta$; and PB will be included in a ledger block with time $r' \leq r + 2\Delta_{\text{tx}} + 2\ell + 5\Delta$ (again because of chain quality and it takes up to Δ rounds to diffuse PB to all honest parties). This concludes our proof. □

D.3 Consistency, Liveness and Order-Fairness

We prove that the ledger $\mathcal{L}$ of Taxis satisfies three desired properties. For consistency, parties have to remove a suffix of transactions from their local ledger $\mathcal{L}$ according to the starting point of their K-time-window. More specifically, suppose the consistency parameter is $k_{\mathcal{L}}$. At round r parties remove transactions in $\mathcal{L}$ such that the settled part does not contain transactions with K-time-window starting after $r - k_{\mathcal{L}}$. Note that the beginning time of K-time-window might not be monotonically increasing in $\mathcal{L}$, hence a transaction $\mathtt{tx}$ whose time window starts before $r - k_{\mathcal{L}}$ might also be removed if a transaction with later starting point precedes $\mathtt{tx}$ in $\mathcal{L}$ (but parties should let the settled part contain as many transaction as possible).

We say that a vertex v in the condensed dependency graph G^* is confirmed if all the transactions in v have their K-time-window in the settled part of the Taxis blockchain and all their ancestors in G^* have confirmed.

Theorem 27 (Consistency). *In a typical execution and (γ, s)-respecting environment, Consistency is satisfied by setting the settled transactions to be those reported more than $K + 4\ell + 9\Delta + 2\Delta_{\text{tx}}$ rounds deep.*

Proof. Consider two honest parties $\mathsf{P}_1, \mathsf{P}_2$ reporting $\mathcal{L}_1, \mathcal{L}_2$ at rounds $r_1 \leq r_2$, respectively. For the sake of a contradiction, suppose after removing all unsettled transactions from $\mathcal{L}_1, \mathcal{L}_2$ it holds that $\mathcal{L}_1 \not\preceq \mathcal{L}_2$, and let $\mathtt{tx}_1, \mathtt{tx}_2$ denote the transaction in $\mathcal{L}_1$ and $\mathcal{L}_2$ after the fork respectively. Let G_r denote the dependency graph at round r and G_r^* its condensation graph, and v_1, v_2 the vertices in G_r^* that contains $\mathtt{tx}_1, \mathtt{tx}_2$ correspondingly.

Since transactions are more than $K + 4\ell + 9\Delta + 2\Delta_{\text{tx}}$ rounds deep and TXPool contains transactions that are $K + \ell + 2\Delta$ round deep, by considering Lemma 26, it holds that for a transaction $\mathtt{tx}_1$ ($\mathtt{tx}_2$ resp.) in the settled part of $\mathcal{L}$, if $\mathtt{tx}_2 \prec^{\varphi} \mathtt{tx}_1$ ($\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$ resp.) then $\mathtt{tx}_2$ ($\mathtt{tx}_1$ resp.) will be in TXPool.

If $\mathtt{tx}_1$ and $\mathtt{tx}_2$ are not in the same cycle, we have $v_1 \neq v_2$ and consider three scenarios. (i) Suppose $\mathtt{tx}_1 \prec^{\varphi} \mathtt{tx}_2$, at round r_2 since $\mathtt{tx}_2 \in \mathcal{L}$ we have $\mathtt{tx}_1 \in \mathsf{TXPool}$. And, an edge (v_1, v_2) must exist in G because of Lemma 25. This contradicts the fact that $\mathcal{L}_2$ report $\mathtt{tx}_2$ before $\mathtt{tx}_1$, as (v_1, v_2) implies that $\mathtt{tx}_1$ must be ordered before $\mathtt{tx}_2$. (ii) Suppose $\mathtt{tx}_2 \prec^{\varphi} \mathtt{tx}_1$, at round r_1 since $\mathtt{tx}_1 \in \mathcal{L}$ we have $\mathtt{tx}_2 \in \mathsf{TXPool}$. And, an edge (v_2, v_1) must exist in G because of Lemma 25. This contradicts the fact that $\mathcal{L}_1$ report $\mathtt{tx}_1$ before $\mathtt{tx}_2$. (iii) We suppose $\mathtt{tx}_1, \mathtt{tx}_2$ are incomparable. Consider G_{r_1} it holds that both $\mathtt{tx}_1, \mathtt{tx}_2$ are in TXPool. For all round $r \geq r_1$, the existence and orientation of edge between $\mathtt{tx}_1, \mathtt{tx}_2$ are the same as that at round r_1. If $(\mathtt{tx}_1, \mathtt{tx}_2) \in G_{r_1}$, then it contradicts the fact that $\mathcal{L}_2$ reports $\mathtt{tx}_2$ before $\mathtt{tx}_1$ at round r_2; if $(\mathtt{tx}_2, \mathtt{tx}_1) \in G_{r_1}$, then it contradicts the fact that $\mathcal{L}_1$ reports $\mathtt{tx}_1$ before $\mathtt{tx}_2$ at round r_1; and if no edge exists, $\mathcal{L}_1$ reports $\mathtt{tx}_1$ before $\mathtt{tx}_2$ implies that the beginning time of the K-time-window of $\mathtt{tx}_1$ is earlier than that of $\mathtt{tx}_2$, again this contradicts the fact that $\mathcal{L}_2$ reports $\mathtt{tx}_2$ before $\mathtt{tx}_1$.

If $\mathtt{tx}_1$ and $\mathtt{tx}_2$ are in the same cycle, we have $v_1 = v_2$. Note that after the completion of an SCC (completion means that all transactions in SCC are confirmed), no vertices can be added or

removed. This is because if a vertex is added, it implies that some vertices in the SCC are waiting for other transactions, contradicting that SCC is complete; if a vertex is removed, then it implies some of the vertices in the SCC contains the transaction that are not in TXPool. Suppose the cycle is small, we directly get the contradiction in that Algorithm 1 is deterministic.

Now, suppose that the cycle is large and it runs the fallback mechanism in Protocol 3. And consider a transaction $\texttt{tx}$ in this cycle with K-time-window starting at round r. Lemma 26 guarantees that all transactions that might have a median timestamp earlier than $\texttt{tx}$ will start their K-time-window no later than $3\ell + 7\Delta + 2\Delta_{\text{tx}}$, since Lemma 24 implies that the median timestamp will fall in the Δ_{tx}-dissemination window. Hence, when $\texttt{tx}$ is still in $V_{\mathsf{unconfirmed}}$ at round $r + K + 4\ell + 9\Delta + 7\Delta_{\text{tx}}$, it implies that a cycle spans for more than $3\Delta_{\text{tx}}$ rounds exist. And by outputing transactions with an increasing order on their median timestamps up to $\texttt{tx}$, we get that the output at the ongoing of this long cycle is consistent. When tracing back at the future rounds, note that for $\texttt{tx}$ to be in $V_{\mathsf{unconfirmed}}$ at round $r + K + 4\ell + 9\Delta + 7\Delta_{\text{tx}}$, there must exist a $\texttt{tx}'$ such that its K-time-window starts no earlier than $r+3\ell+7\Delta+7\Delta_{\text{tx}}$ which will pass the check in Line 7 in Protocol 3 and always output the same order due to the median timestamp rule. □

Theorem 28 (Liveness). *In a typical execution and (γ, s)-respecting environment, Liveness is satisfied for wait time $K + 6\ell + 14\Delta + 7\Delta_{\text{tx}}$ rounds.*

Proof. Let r denot the least round such that $\texttt{tx}$ is learnt by all alert parties. Similar to the argument in Lemma 26 we get that $\texttt{tx}$ and its K-time-window will start with a timestamp no later than $r+2\ell+5\Delta$. Since the fallback in Protocol 3 guarantees that $\texttt{tx}$ is forcibly inserted into $\mathcal{L}$ after remaining in the dependency graph for more than $\Delta_{\mathsf{timeout}}$ rounds, by considering Equation (2) we get $K + 6\ell + 14\Delta + 7\Delta_{\text{tx}}$, this completes the proof of liveness. □

Theorem 29 (Order-Fairness). *In a typical execution and (γ, s)-respecting environment, Taxis implements $(\varphi, \texttt{TDBW})$-fair-order serialization.*

Proof. Fix an execution E and honest profiles $\mathcal{R}^{\mathcal{H}}$. Consider the dependency graph G in an honest party's view at the end of the execution. Lemma 25 shows that all edges in $G_{\mathcal{R}^{\mathcal{H}},\varphi}$ will be in G. I.e., G is a spanning supergraph of $G_{\mathcal{R}^{\mathcal{H}},\varphi}$. It suffices to prove that there exist $\mathcal{R}^{\mathcal{A}}$ such that the $(\langle \mathcal{R}^{\mathcal{H}}, \mathcal{R}^{\mathcal{A}} \rangle)$-dependency graph is exactly G. Note that $\mathcal{R}^{\mathcal{H}}$ in K rounds are proportional to the profile blocks mined with timestamps fall in these K rounds. Their exist some $\mathcal{R}^{\mathcal{A}}$ that is also proportional to the adversarial profile blocks mined in this time period.

Note that by considering Lemma 26 and $\Delta_{\mathsf{timeout}}$ in Equation (2), we get that any cycle that passes the fallback check has vertices with timestamp difference more than $5\Delta_{\text{tx}}$ rounds. Since the adversarial manipulation on the transaction is up to Δ_{tx} rounds, this implies that there exist at least two transactions in the cycle such that the first time that they are delivered to an honest party is at least $3\Delta_{\text{tx}}$ apart. Combining this result with Theorem 8 and 9 we learn that their order follows Definition 7. This concludes the proof. □