# Not All Fair Probabilistic Schedulers are Equivalent*

Ioannis Chatzigiannakis[1], Shlomi Dolev[2], Sándor P. Fekete[3], Othon Michail[1], and Paul G. Spirakis[1]

[1] Research Academic Computer Technology Institute (RACTI), and Computer Engineering and Informatics Department (CEID), University of Patras, 26500, Patras, Greece
[2] Department of Computer Science, Ben-Gurion University of the Negev, Israel 84105
[3] Department of Computer Science, Braunschweig University of Technology, Braunschweig, Germany
Email: {ichatz, michailo, spirakis}@cti.gr, dolev@cs.bgu.ac.il, s.fekete@tu-bs.de

**Abstract.** We propose a novel, generic definition of *probabilistic schedulers* for population protocols. We then identify the *consistent* probabilistic schedulers, and prove that any consistent scheduler that assigns a non-zero probability to any transition $i \rightarrow j$, where $i$ and $j$ are configurations satisfying $i \neq j$, is fair with probability 1. This is a new theoretical framework that aims to simplify proving specific probabilistic schedulers fair. In this paper we propose two new schedulers, the *State Scheduler* and the *Transition Function Scheduler*. Both possess the significant capability of being *protocol-aware*, i.e. they can assign transition probabilities based on information concerning the underlying protocol. By using our framework we prove that the proposed schedulers, and also the *Random Scheduler* that was defined by Angluin et al. [2], are all fair with probability 1. Finally, we define and study *equivalence* between schedulers w.r.t. *performance* and *correctness* and prove that there exist fair probabilistic schedulers that are not equivalent w.r.t. to performance and others that are not equivalent w.r.t. correctness.

**Keywords.** population protocol, probabilistic scheduler, fair scheduler, fairness, communicating automata, sensor network.

## 1 Introduction

Recently, Angluin et al. [2, 3] introduced the notion of a computation by a *Population Protocol* to model distributed systems in which individual agents are extremely limited and can be represented as finite-state machines. In their model, complex behavior of the system as a whole emerges from the rules governing

* This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

pairwise interaction of the agents. The computation is carried out by a collection of agents, each of which receives a piece of the input. These agents move around and information can be exchanged between two agents whenever they come into contact with (or sufficiently close to) each other. The goal is to ensure that every agent can eventually output the value that is to be computed.

An execution of a protocol proceeds from the initial configuration by interactions between pairs of agents. In a real distributed execution, interactions could take place simultaneously, but, when writing down an execution, simultaneous interactions can be ordered arbitrarily. Angluin et al. think of the order in which pairs of agents come into contact and interact as being chosen by an adversary. From a particular system configuration, the adversary decides which of the possible different interactions will be selected; essentially, it decides the computation sequence (i.e. schedule of interactions). So, the designer's goal is to make protocols work correctly under any schedule the adversary may choose.

In such models there may exist diverging (infinite) schedules of interactions such that during their execution some *event becomes possible infinitely often* but it has not an infinite number of occurrences. If the adversary selects such a sequence, it will lead the system to unfair situations, where although an event is realizable infinitely often, it never occurs because conflicts are resolved in a non equitable manner. To deal with these issues, a **fairness** restriction is imposed on the adversarial scheduler: the scheduler is not allowed to avoid a possible step forever. The fairness constraint allows the scheduler to behave arbitrarily for an arbitrarily long period of time, but does require that it behave nicely eventually. Therefore correctness is a property that can be satisfied eventually.

Fairness relative to a set of states is important since most of the "interesting" system properties express reachability relations of some set of states [16]. In other words, fairness becomes crucial when a property is to be proven in formal systems based on non-deterministic models. In this work we try to apprehend the concept of fairness in the *basic population protocol model*. To do so, we focus on the class of *probabilistic schedulers* proposed in [2, 3], in which the scheduler selects randomly the next pair to interact. We define two new adversaries that are bound by the fairness constraint of [2]. The "reasonable" scheduling policies that they introduce lead to significantly different performance characterizations for some protocols well studied in the relevant literature. We show that the current notion of fairness gives rise to many difficulties in studying not only performance but also protocol correctness.

In the area initiated by the proposal of the *Population Protocol* (PP) model [2] much work has been devoted to the, now, well-known fact that the set of computable predicates of the basic (complete interaction graph) PP model and most of its variants is exactly equal or closely related to the set of *semilinear predicates*. Moreover, in [2, 3], the *Probabilistic Population Protocol* model was proposed, in which the scheduler selects randomly and uniformly the next pair to interact. More recent work has concentrated on performance, supported by this random scheduling assumption. Additionally, several extensions of the basic model have been proposed in order to more accurately reflect the requirements

of practical systems. In [1], Angluin et al. studied what properties of restricted communication graphs are stably computable, gave protocols for some of them, and proposed a model extension with *stabilizing inputs*. In [11] the *Mediated Population Protocol* (MPP) model was proposed that extends the PP model with communication links that are able to store states. The MPP model was proved to be computationally stronger than the PP model and it was observed that it is especially capable of deciding graph properties, concerning the communication graph on which the protocol runs. In [9] the decidable graph properties by MPPs where studied for the first time and it was proven that connectivity cannot be decided by the new model. Unfortunatelly, the class of decidable graph languages by MPPs remains open. Finally, some works incorporated agent failures and gave to the agents slightly increased memory capacity. For the interested reader, [7] and [12] constitute nice introductions to the subject.

In Section 2 we provide a brief introduction to the PP model. In Section 3.1 we give a novel generic definition of *probabilistic schedulers*. We then study separately those that are *consistent*, i.e. those that never change the one-step transition probabilities between configurations, and state and prove a theorem that constitutes a useful tool for proving that a specific probabilistic scheduler is fair. In 3.2 we present the *protocol-oblivious Random Scheduler* as was proposed in [2] and define two new schedulers that are *protocol-aware*, namely the *State Scheduler* and *Transition Function Scheduler*. We then use our tool and prove that all these schedulers are fair with probability 1. In Section 3.3 we define *time equivalence* and *computational equivalence* of two probabilistic fair schedulers w.r.t. some population protocol $\mathcal{A}$. In Section 4 we study the performance of the OR Protocol (based on an example of [7]) when faced with our schedulers. This makes it evident that the fairness condition alone, as has been defined by Angluin et al. in [2], is not sufficient to guarantee the construction of protocols that perform well under all kinds of allowed schedulers. It seems that a protocol may perform optimally under some fair scheduler but at the same time reach its worst-case performance under some other, also provably fair, scheduler. In other words, we show that there exists a protocol for which two fair probabilistic schedulers are not time equivalent. Thus, either some stronger definition of fairness needs to be proposed or there needs to be some other way to formally exclude protocol-aware schedulers and other (yet unknown) types of schedulers that can be adjusted to lead to divergent performance scenarios. In Section 5 we show that, due to the weakness characterizing the selected notion of fairness, not only performance but also protocol correctness depends greatly on the underlying scheduler. To do so, we consider the Majority Protocol that was proven correct with high probability in [6] (if certain rational assumptions are satisfied), and study its behavior under the Transition Function Scheduler. We show that the Majority Protocol has a great probability of failure if the underlying scheduler is assumed to be the protocol-aware Transition Function Scheduler, which in turn implies that the Transition Function Scheduler is not computationally equivalent to the Random Scheduler w.r.t. the Majority Protocol. Finally, in Section 6 we discuss some promising future research directions.

## 2 Population Protocols

A *population protocol* (PP) is a 6-tuple $(X, Y, Q, I, O, \delta)$, where $X$, $Y$, and $Q$ are all finite sets, and $X$ is the *input alphabet*, $Y$ is the *output alphabet*, $Q$ is the set of *states*, $I : X \to Q$ is the *input function*, $O : Q \to Y$ is the *output function*, and $\delta : Q \times Q \to Q \times Q$ is the *transition function*. If $\delta(a, b) = (a', b')$, we call $(a, b) \to (a', b')$ a *transition* and we define $\delta_1(a, b) = a'$ and $\delta_2(a, b) = b'$.

A population protocol $\mathcal{A} = (X, Y, Q, I, O, \delta)$ runs on a *communication graph* (also known as *interaction graph*) $G = (V, E)$ ($G$ is here assumed to be directed and without multiple edges or self-loops). From now on we will use the letter $n$ to denote the cardinality of $V$ (size of the population). Initially, all agents (i.e. the elements of $V$) receive a global start signal, sense their environment and each one receives an input symbol from $X$. After receiving their input symbol, all agents apply the input function to it and go to their initial state (e.g. all agents that received $\sigma \in X$ begin with initial state $I(\sigma) \in Q$). An adversary scheduler selects in each step a directed pair of agents $(u, v) \in E$, where $u, v \in V$ and $u \neq v$, to interact. Assume that the scheduler selects the pair $(u, v)$, that the current states of $u$ and $v$ are $a, b \in Q$, respectively, and that $\delta(a, b) = (a', b')$. Agent $u$ plays the role of the *initiator* in the interaction $(u, v)$ and $v$ that of the *responder*. When interacting, $u$ and $v$ apply the transition function to their directed pair of states, and, as a result, $u$ goes to $a'$ and $v$ to $b'$ (both update their states according to $\delta$, and specifically, the initiator applies $\delta_1$ while the responder $\delta_2$).

A *configuration* is a snapshot of the population states. Formally, a configuration is a mapping $C : V \to Q$ specifying the state of each agent in the population. $C_0$ is the initial configuration (for simplicity we assume that all agents apply the input function at the same time) and, for all $u \in V$, $C_0(u) = I(x(u))$, where $x(u)$ is the input symbol sensed by agent $u$. Let $C$ and $C'$ be configurations, and let $u$, $v$ be distinct agents. We say that $C$ goes to $C'$ via encounter $e = (u, v)$, denoted $C \xrightarrow{e} C'$, if $C'(u) = \delta_1(C(u), C(v))$, $C'(v) = \delta_2(C(u), C(v))$, and $C'(w) = C(w)$ for all $w \in V - \{u, v\}$, that is, $C'$ is the result of the interaction of the pair $(u, v)$ under configuration $C$ and is the same as $C$ except for the fact that the states of $u$, $v$ have been updated according to $\delta_1$ and $\delta_2$, respectively. We say that $C$ can go to $C'$ in one step, denoted $C \to C'$, if $C \xrightarrow{e} C'$ for some encounter $e \in E$. We write $C \xrightarrow{*} C'$ if there is a sequence of configurations $C = C_0, C_1, \ldots, C_t = C'$, such that $C_i \to C_{i+1}$ for all $i$, $0 \leq i < t$, in which case we say that $C'$ is *reachable* from $C$.

## 3 Schedulers

### 3.1 Fair Probabilistic Schedulers

As defined in [3], the *transition graph* $T(\mathcal{A}, G)$ of a protocol $\mathcal{A}$ running on a communication graph $G$ (or just $T$ when no confusion arises) is a directed graph whose nodes are all possible configurations and whose edges are all possible one-step transitions between those configurations.

**Definition 1.** *A* probabilistic scheduler*, w.r.t. a transition graph $T(\mathcal{A}, G)$, defines for each configuration $C \in V(T)$ an infinite sequence of probability distributions of the form $(d_1^C, d_2^C, \ldots)$, over the set $\Gamma^+(C) = \{C' \mid C \to C'\}$ (the possibly closed out-neighbourhood of $C$), where $d_t^C : \Gamma^+(C) \to [0,1]$ and such that $\sum_{C' \in \Gamma^+(C)} d_t^C(C') = 1$ holds, for all $t$ and $C$.*

The initial configuration $C_0$ depends only on the values sensed by the population and, in particular, it is formed by their images under the input function. So, for the time being, we can assume that $C_0$ is selected in a deterministic manner. Let $C_t$ denote the configuration selected by the scheduler at step $t$ (the configuration of the system after $t$ selections of the scheduler and applications of the transition function). Assume that it is the $l$th time that $C_t$ is encountered during the execution so far; then a probabilistic scheduler selects $C_{t+1}$ randomly, according to the distribution $d_l^{C_t}$. In other words, $d_l^C$ denotes the probability distribution over $\Gamma^+(C)$ when $C$ is encountered for the $l$th time.

**Definition 2.** *We call a probabilistic scheduler* consistent*, w.r.t. a transition graph $T(\mathcal{A}, G)$, if for all configurations $C \in V(T)$, it holds that $d^C = d_1^C = d_2^C = \ldots$, which, in words, means that any time the scheduler encounters configuration $C$ it chooses the next configuration with the same probability distribution $d^C$ over $\Gamma^+(C)$, and this holds for all $C$ (each with its own distribution).*

From now on, and when no confusion arises, we shall use the letters $i$ and $j$ not only to denote configuration indices but also to denote configurations themselves. Note that a consistent probabilistic scheduler for $T(\mathcal{A}, G)$ is simply a labeling $P : E(T) \to [0,1]$ on the arcs of $T$, such that for any $i \in V(T)$, $\sum_{j \in \Gamma^+(i)} P(i,j) = 1$. So, any time a consistent scheduler encounters a configuration $i$, it selects the next configuration j according to the probability distribution defined by the labels of the arcs leaving from $i$. Note that in the latter case, if we remove from $T$ all $e \in E(T)$ where $P(e) = 0$ then the resulting graph $D$ is the underlying graph of a finite Markov chain where the state space is $\mathcal{C} = Q^V$ (all possible configurations) and for all $i, j \in \mathcal{C}$, if $i \to j$ then $\mathbb{P}_{ij} = P(i,j)$, i.e. equal to the label of arc $(i,j)$, otherwise $\mathbb{P}_{ij} = 0$.

A strongly connected component of a directed graph is *final* iff no arc leads from a node in the component to a node outside. A *configuration* is final iff it belongs to a final strongly connected component of the transition graph.

An *execution* is a finite or infinite sequence of configurations $C_0, C_1, C_2, \ldots$, where $C_0$ is an initial configuration and $C_i \to C_{i+1}$, for all $i \geq 0$. An infinite execution is *fair* if for every possible transition $C \to C'$, if $C$ occurs infinitely often in the execution then $C'$ also occurs infinitely often. A *computation* is an infinite fair execution.

Let $y_C$, where $y_C(u) = O(C(u))$ for all $u \in V$, denote the output (assignment) of configuration $C$. We say that a computation of a population protocol $\mathcal{A}$ *stabilizes* to output $y_C$ if it contains a configuration $C$ such that for all $C'$ reachable from $C$ we have that $y_{C'} = y_C$.

**Theorem 1.** *Let $\Xi = C_0, C_1, \ldots$ be an infinite execution of $\mathcal{A}$ on $G$, $\mathcal{F}_\Xi$ be the set of configurations that occur infinitely often in $\Xi$, and $T_{\mathcal{F}_\Xi}$ be the subgraph of $T(\mathcal{A}, G)$ induced by $\mathcal{F}_\Xi$. $\Xi$ is a computation (i.e. it is additionally fair) iff $T_{\mathcal{F}_\Xi}$ is a final strongly connected component of $T(\mathcal{A}, G)$.*

*Proof.* The "only if" part was proven in [3]. We prove here the "if" part. Assume that $T_{\mathcal{F}_\Xi}$ is a final strongly connected component of $T(\mathcal{A}, G)$ and that $\Xi$ is not fair (i.e. that the statement of the "if" part does not hold). Then there exists some configuration $C \in \mathcal{F}_\Xi$ (i.e. appearing infinitely often) and a $C' \notin \mathcal{F}_\Xi$ such that $C \to C'$. But this contradicts the fact that $T_{\mathcal{F}_\Xi}$ is final. □

We now keep the preceding definitions of $\Xi$, $T(\mathcal{A}, G)$, $\mathcal{F}_\Xi$, $T_{\mathcal{F}_\Xi}$, but additionally assume a consistent scheduler.

**Theorem 2.** *If for all $i \in \mathcal{F}_\Xi$ and all configurations $j$ s.t. $i \to j$ it holds that $\mathbb{P}_{ij} > 0$, then $\Xi$ is a computation with probability 1.*

*Proof.* Because $i$ is persistent and all its successor configurations $j$ may occur in one step from $i$ with non-zero probability it follows that those $j$ are also persistent with probability 1, i.e. they also occur infinitely often in $\Xi$, thus $\Xi$ is a computation with probability 1, by definition. □

**Definition 3.** *A scheduler $S$ is fair if for any protocol $\mathcal{A}$, any communication graph $G$, and any infinite execution $\Xi$ of $\mathcal{A}$ on $G$ caused by $S$, $\Xi$ is also a computation (i.e. additionally fair).*

Intuitively a scheduler is fair if it always leads to computations.

**Theorem 3.** *Any consistent scheduler, for which it holds that $\mathbb{P}_{ij} > 0$, for any protocol $\mathcal{A}$, any communication graph $G$, and all configurations $i, j \in V(T(\mathcal{A}, G))$ where $i \to j$ and $i \neq j$, is fair with probability 1.*

*Proof.* First of all, note that the underlying Markov chain graph of such a scheduler is the transition graph without possibly some self-loops. Assume that the statement does not hold. Then the probability that a specific infinite execution $\Xi$ of some protocol $\mathcal{A}$ on some graph $G$ caused by the scheduler is not a computation is non-zero. This means that a $\Xi$ may occur, for which there exists some configuration $i \in \mathcal{F}_\Xi$ and $j \notin \mathcal{F}_\Xi$ such that $i \to j$. Now there are two cases:

1. $i = j$. In this case the contradiction is trivial, because it follows that $i \in \mathcal{F}_\Xi$ while at the same time $i \notin \mathcal{F}_\Xi$.
2. $i \neq j$. However, by assumption $\mathbb{P}_{ij} > 0$, and because $i$ is persistent $j$ must also be with probability 1.

□

### 3.2 Proposed Schedulers

In [2] a probabilistic scheduler that selects the next ordered pair to interact at random, independently and uniformly from all ordered pairs corresponding to arcs of the communication graph (i.e. elements of $E$) was defined. Here we call this scheduler Random Scheduler, and define two new probabilistic schedulers, namely the State Scheduler and the Transition Function Scheduler.

The **Random Scheduler**. To generate $C_{i+1}$ the Random Scheduler selects an ordered pair $(u, v) \in E$ at random, independently and uniformly (each with probability $1/|E|$), and applies the transition function to $(C_i(u), C_i(v))$.

The **State Scheduler**. Consider a population protocol for $k$-mutual exclusion, in which only $k$ agents are in state 1 and the rest of the population is in state 0. When an agent that holds a token interacts with another agent, it passes the token. Now consider an execution where $n \gg k$ and we use the Random Scheduler. In the case in which the communication graph is complete, the probability of selecting a pair with states $(1, 0)$ is much smaller than selecting a pair with states $(0, 0)$, meaning that the scheduler may initiate a large number of interactions that do not help the protocol in making progress. The *State Scheduler* instead of selecting a pair of processes independently and uniformly it selects a pair based on the states of the processes. It first selects a pair of *states* and in the sequel it selects one process from each state. Thus it allows the "meaningful" transitions to be selected more often and may avoid selecting a large number of interactions that delay the protocol's progress.

More formally, an ordered pair of states $(q, q')$ is said to be an *interaction candidate* under configuration $C$ if $\exists (u, v) \in E$ such that $C(u) = q$ and $C(v) = q'$. Then a configuration $C_{i+1}$ is generated from $C_i$ as follows: (i) by drawing an order pair $(q, q')$ of states at random, independently and uniformly from all ordered pairs of states that are interaction candidates under $C_i$, (ii) drawing an ordered pair $(u, v)$ such that $C_i(u) = q$ and $C_i(v) = q'$ from all such pairs at random, independently and uniformly, (iii) applying the transition function $\delta$ to $(C_i(u), C_i(v))$ and updating the states of $u$ and $v$ accordingly to obtain $C_{i+1}$.

The **Transition Function Scheduler**. Continuing the same argument, we define one more scheduler that assumes knowledge of the protocol executed. It examines the transition function $\delta$ and selects pairs of agents based on the defined transitions. In the case in which function $\delta$ defines transitions that do not change the state, neither of the initiator nor of the responder agent (e.g., $(\alpha, \beta) \rightarrow (\alpha, \beta)$), these transitions are ignored by the scheduler. This scheduler guarantees that all interactions will lead to a state change of either the initiator or the responder or both.

More formally, suppose $\rightarrow$ is a binary relation over $Q^2$ which is the relation analogue of the corresponding transition function $\delta$. The reflexive reduction of $\rightarrow$, denoted by $\dot{\rightarrow}$, is simply $\rightarrow$ without members related to themselves by $\rightarrow$. A configuration $C_{i+1}$ is generated from $C_i$ as follows: (i) by drawing a pair $((q_1, q_2), (q_1', q_2'))$ at random, independently and uniformly from all such pairs belonging to $\dot{\rightarrow}$ for which $(q_1, q_2)$ is an interaction candidate under $C_i$, (ii) drawing an ordered pair $(u, v)$ such that $C_i(u) = q_1$ and $C_i(v) = q_2$ from all such pairs

at random, independently and uniformly, (iii) applying the transition function $\delta$ to $(C_i(u), C_i(v))$ and updating the states of $u$ and $v$ accordingly to obtain $C_{i+1}$ (if in step (i) there exists no such interaction candidate, then the Transition Function Scheduler becomes a Random Scheduler, and remains in the same configuration for an infinite number of steps).

Given the above schedulers we can classify any scheduler for population protocols based on whether it assumes any knowledge on the actual protocol executed or not.

**Definition 4.** *We call a scheduler* protocol-oblivious *(or* agnostic*) if it constructs the interaction pattern without any knowledge on the protocol executed and* protocol-aware *if it takes into account information concerning the underlying protocol.*

Based on this classification, the Random Scheduler is a protocol-oblivious scheduler while the State and Transition Function Schedulers are protocol-aware.

**Theorem 4.** *The Random Scheduler, State Scheduler, and Transition Function Scheduler are all fair with probability 1.*

*Proof.* Let $T(\mathcal{A}, G)$ be any transition graph.

- *Random Scheduler.* Let $i$ be any configuration in $V(T)$. Any time $i$ is encountered, any $j$ for which $i \to j$ is selected with probability $\mathbb{P}_{ij} = |K_{ij}|/|E|$, where $K_{ij} = \{e \mid e \in E(G) \text{ and } i \xrightarrow{e} j\}$, which is independent of the number of times $i$ has been encountered. Thus the Random Scheduler is consistent. Moreover, $|K_{ij}| > 0$, because from definition of $i \to j$ we have that $\exists e \in E$ ($E$ is used instead of $E(G)$) such that $i \xrightarrow{e} j$. Thus $\mathbb{P}_{ij} > 0$ and Theorem 3 applies implying that the Random Scheduler is fair with probability 1.
- *State Scheduler.* Let $i, j$ be distinct configurations in $V(T)$ such that $i \to j$. When the State Scheduler has chosen $i$ to select the next configuration of the execution, it performs two experiments. First it selects a pair of states $(q, q')$ from all interaction candidates. Then it selects an arc $e$ from all $(u, v) \in E$ such that $i(u) = q$ and $i(v) = q'$. Let $K_{ij}$ again denote the set of arcs (i.e. interactions) that convert $i$ to $j$. Let also $M_{ij} = \{(q, q') \mid \exists (u, v) \in K_{ij}$ such that $i(u) = q$ and $i(v) = q'\}$ and $IC_i$ denote the set of all interaction candidates under $i$ (note that $M_{ij} \subseteq IC_i$). Now $1/|IC_i|$ is the probability that a specific interaction candidate is selected by the scheduler. Let $K_{ij}^{(q,q')} = \{(u, v) \mid (u, v) \in K_{ij}$ and $i(u) = q, i(v) = q'\}$ (the subset of $K_{ij}$ containing all arcs $(u, v)$ that convert $i$ to $j$ and where the state of $u$ is $q$ and the state of $v$ is $q'$) and $E_i^{(q,q')} = \{(u, v) \mid (u, v) \in E$ and $i(u) = q, i(v) = q'\}$. Now given a chosen interaction candidate $(q, q') \in M_{ij}$ the probability that $j$ is selected is equal to $|K_{ij}^{(q,q')}|/|E_i^{(q,q')}|$. Thus we have

$$\mathbb{P}_{ij} = \sum_{(q,q') \in M_{ij}} \frac{|K_{ij}^{(q,q')}|}{|IC_i||E_i^{(q,q')}|}.$$

$|K_{ij}^{(q,q')}|$, $|IC_i|$ and $|E_i^{(q,q')}|$ for all $(q, q') \in M_{ij}$ only depend on the specific configurations $i$ and $j$ and are always the same w.r.t. different times at which $i$ is encountered by the scheduler. Thus the State Scheduler is consistent. Moreover, since $(i \to j) \Rightarrow \exists e = (u, v) \in E$ such that $i \xrightarrow{e} j$. Let $q$ and $q'$ be the states of $u$ and $v$ under $i$, respectively. It follows that $(q, q') \in M_{ij}$ and that $|M_{ij}| > 0$. Finally, note that $e \in K_{ij}^{(q,q')}$, because $e \in K_{ij}$, $i(u) = q$, and $i(v) = q'$. Thus $\mathbb{P}_{ij} > 0$, Theorem 3 applies and as a consequence the State Scheduler is fair with probability 1.

– *Transition Function Scheduler.* In the case in which $i \neq j$, $\mathbb{P}_{ij}$ is defined as in the State Scheduler, by simply replacing the phrase "interaction candidate" with "interaction candidate that constitutes the lhs of some rule in the reflexive reduction of $\delta$". So also this scheduler is consistent and fair with probability 1. Note that when $i = j$ and $i$ has at least one out-neighbor in $T$ different from $i$, then $\mathbb{P}_{ij} = 0$, since this scheduler does not select transitions that leave the states of the participating agents unaffected. Moreover, if $i$ has a unique out-going arc (in $T$) pointing to itself, then the scheduler selects $i$ for an infinite number of steps with probability 1 (in this case becomes a Random Scheduler). In both cases no problem arises, because for Theorem 3 to apply we only require $\mathbb{P}_{ij} > 0$ for all $i \neq j$ such that $i \to j$.

□

### 3.3 Equivalence Between Schedulers

**Definition 5.** *Two fair probabilistic schedulers $S_1$ and $S_2$ are called* time equivalent *w.r.t. a protocol $\mathcal{A}$ iff all computations of $\mathcal{A}$ under $S_1$ and $S_2$ beginning from the same initial configuration take asymptotically the same expected time (number of steps) to convergence.*

**Definition 6.** *Two fair probabilistic schedulers $S_1$ and $S_2$ are called* computationally equivalent *w.r.t. a protocol $\mathcal{A}$ iff for all computations of $\mathcal{A}$ under $S_1$ and $S_2$ beginning from the same initial configuration, w.h.p., $\mathcal{A}$ stabilizes to the same output assignment (the output assignment of a configuration $C$ is $y_C : V \to Y$ defined as $y_C(u) = O(C(u))$ for all $u \in V$).*

## 4 Not All Fair Probabilistic Schedulers are Time Equivalent

We use a simple protocol, called the *OR Protocol* or the *One-Way Epidemic Protocol*, based on an example of [7], in which each agent with input 0 simply outputs 1 as soon as it interacts with some agent in state 1. We, also, assume that the underlying communication graph is complete. Formally, we have $Q = X = Y = \{0, 1\}$ and the transitions defined by $\delta$ are the following:

$(0,0) \to (0,0)$ $\quad$ $(1,0) \to (1,1)$
$(0,1) \to (1,1)$ $\quad$ $(1,1) \to (1,1)$

Essentially, if all agents have input 0, no agent will ever be in state 1. If some agent has input 1, given a fair scheduler, we expect that the number of agents with state 1 will increase and will eventually reach $n$. In both cases, due to fairness, all agents will eventually stabilize to the correct output value, though an important fundamental questions is "how fast is stability reached?" and "how do different schedulers affect the performance of the protocol?".

In [4], Angluin et al. characterized the behavior of the OR Protocol in complete communication graphs as a *one-way epidemic*. They showed that the number of interactions for the epidemic to finish in the case of the Random Scheduler is $\Theta(n \log n)$ w.h.p., by exploiting the well-known coupon collector problem.

**Theorem 5.** *The State Scheduler and the Transition Function Scheduler are time equivalent w.r.t. the One-Way Epidemic Protocol.*

*Proof.* Both schedulers require only $\mathcal{O}(n)$ interactions. In particular, the Transition Function Scheduler can choose only between transitions $(1,0) \rightarrow (1,1)$ and $(0,1) \rightarrow (1,1)$ that both increase the number of agents in state 1 by one. If initially at least one agent is in state 1, then in each step one agent goes from state 0 to state 1 (no new agents in state 0 emerge) and because the agents are $n$, in at most $n-1$ steps all agents will be in state 1 and stability will have been reached. In the case of the State Scheduler, assume the worst-case scenario in which initially only one agent is in state 1. Because the graph is complete, the interaction candidates are in the first step $(0,0)$, $(0,1)$, and $(1,0)$. So, initially, there is a $2/3$ probability to select a transition that gives birth to a new 1. When this happens, in an expected number of 1.5 steps, all four left-hand sides of the rules of $\delta$ will be interaction candidates (until the step in which only one 0 remains, when again the probability of progress becomes $2/3$). In all other possible configurations the probability to progress is $1/2$, thus progress is always made with at least probability $1/2$, which in turn implies that on average at most $2(n-1)$ (i.e. again $\mathcal{O}(n)$) steps are expected until stability is reached. □

The above discussion indicates that the performance of a population protocol clearly depends on the scheduler's functionality. In fact, it seems here that the additional knowledge, concerning the transition function, allowed to the State Scheduler and the Transition Function Scheduler provides us with interaction patterns that always lead to optimal computations. However, we can show that the same knowledge may also allow the definition of fair schedulers that lead the protocols to worst-case scenarios. To do so we slightly modify the State Scheduler to obtain a new scheduler, called the *Modified Scheduler*. Let us consider the case in which the scheduler is *weakly protocol-aware* in the sense that it can only partition the rules of the transition function to classes (possibly with elements sharing some common property and assign some probability to each class.

**Definition 7.** *The Modified Scheduler selects from the class of the identity rules (rules that leave both the state of the initiator and that of the responder unaffected) with probability* $1-\varepsilon$ *and from all the remaining rules with probability* $\varepsilon$*,*

*where $0 < \varepsilon < 1$. Those probabilities are then evenly divided into the corresponding class members. All other components of the Modified Scheduler's definition remain the same as in the case of the State Scheduler.*

**Theorem 6.** *The Modified Scheduler can lead the One-Way Epidemic Protocol to arbitrarily bad performance.*

*Proof.* First of all, note that the Modified Scheduler is fair with probability 1, because the transition probabilities may have been modified but still remain non-zero for non-loop arcs of $T$ and independent of the number of steps. Consider now the situation in which $n-2$ nodes are initially in state 0 and the remaining 2 are in state 1. Because $n-2$ 0s have to be converted to 1s, it follows that the probability that the computation stabilizes in less than $n-2$ steps is 0. Let the random variable $D$ denote the number of steps until the computation stabilizes (all agents become 1). We have already shown that $\mathbb{P}[D = i] = 0$ for $i < n-2$. Note that $\mathbb{P}[D = i]$ equals $\mathbb{P}$[the last remaining 0 becomes 1 in step $i$]. Let also $N_i$ denote the number of non-identity rules that have appeared in $i$ steps. For the computation to stabilize in $i$ steps, exactly $n-3$ non-identity rules must have been chosen in the first $i-1$ steps ($n-3$ 0s converted to 1s and one 0 remaining) and also a non-identity rule in the last step (the last 0 is converted to a 1). Note that $N_i$ is a binomial random variable having parameters $(i, \varepsilon)$. Then for all $i \geq n-2$

$$\begin{aligned}\mathbb{P}[D = i] &= \mathbb{P}[N_{i-1} = n-3] \cdot \mathbb{P}[\text{non-identity rule appears in step } i] \\ &= \left[\binom{i-1}{n-3}\varepsilon^{n-3}(1-\varepsilon)^{i-1-(n-3)}\right] \cdot \varepsilon \\ &= \binom{i-1}{n-3}\varepsilon^{n-2}(1-\varepsilon)^{i-n+2}\end{aligned}$$

and the expectation of $D$ is

$$\mathbb{E}[D] = \frac{(n-2)}{\varepsilon}.$$

The calculation of the above result can be found in the technical report at http://fronts.cti.gr/aigaion/?TR=93.

Obviously, $\mathbb{E}[D]$ can become arbitrarily large, by decreasing $\varepsilon$ (that is, the probability that a non-identity rule is selected) and the theorem follows. For example, if we set $\varepsilon = (n-2)/2^n$, given that $n > 2$, then the expected number of steps to convergence is exponential in the size of the population. □

Thus, it is evident that the fairness condition, as has been defined by Angluin et al. in [2], is not sufficient to guarantee the construction of protocols that perform well under all kinds of allowed schedulers. It seems that a protocol may perform optimally under some fair scheduler but at the same time reach its worst-case performance under some other, also provably fair, scheduler. Obviously, either some stronger definition of fairness needs to be proposed, that,

for example, would characterize the Modified Scheduler as unfair in the case in which $\varepsilon$ is far away from $1/2$, possibly because it always seems to prefer some class of rules from others, or maybe protocol-aware schedulers and other kinds of yet unknown schedulers that can be adjusted to lead to divergent performance scenarios, should somehow be formally prohibited.

**Theorem 7.** *There exists at least one protocol w.r.t. which some fair probabilistic schedulers are not time equivalent.*

*Proof.* Follows by comparing the expected running time of the One-Way Epidemic Protocol under the State and Transition Function Schedulers to its expected running time under the Random and Modified Schedulers (the latter expected times are from [4] and Theorem 6). □

## 5 Not all Fair Probabilistic Schedulers are Computationally Equivalent

Now we are about to show that, due to the weakness characterizing the selected notion of fairness, not only performance but also protocol correctness depends greatly on the underlying scheduler. Assume that each agent initially votes for one of some election candidates $x$ and $y$ or chooses to vote blank, denoted by $b$. If $x$ is the majority vote, then we want every agent to eventually output $x$, otherwise $y$ (we assume here that the state of an agent is also its output). Now let us consider the following one-way protocol that was proposed in [6].

$$(x,b) \to (x,x) \qquad (x,y) \to (x,b)$$
$$(y,b) \to (y,y) \qquad (y,x) \to (y,b)$$

In words, when an $x$ meets a $b$ it convinces it to vote $x$, when a $y$ meets a $b$ it convinces it to vote $y$, an $x$ switches a $y$ to the blank-undecidable state, and a $y$ does the same to an $x$. Given an initial configuration of $x$s, $y$s and blanks that contains at least one non-blank, the goal is for the agents to reach consensus on one of the values $x$ or $y$. Additionally, the value chosen should be the majority non-blank initial value, provided it exceeds the minority by a sufficient margin. In [6] it was proven that if the above protocol runs under the Random Scheduler on any complete graph with $n$ nodes then with high probability consensus is reached in $\mathcal{O}(n \log n)$ interactions and the value chosen is the majority provided that its initial margin is $\omega(\sqrt{n \log n})$.

It seems that this is not the case when the underlying scheduler is the Transition Function Scheduler. Intuitively, the Transition Function Scheduler does not take into great account the advantage of $x$s. Let $N_x(t)$, $N_y(t)$, and $N_b(t)$ denote the number of $x$s, $y$s, and $b$s before step $t+1$, respectively. Note that when all $x$s, $y$s and $b$s appear in the population then the probability of $N_x(t+1) = N_x(t)+1$ is $1/4$ and the same holds for $N_y(t+1) = N_y(t)+1$. But when the Random Scheduler is assumed, then the greater the number of $x$s, the more the arcs leading from $x$s to $b$s, thus the greater the probability of $N_x(t+1) = N_x(t)+1$.

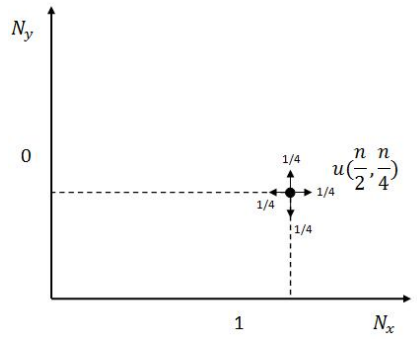


**Fig. 1.** The two-dimensional symmetric random walk. We show that the probability that the particle will reach the $N_y$ axis before reaching the $N_x$ axis is constant.

**Lemma 1.** *The Majority Protocol errs under the Transition Function Scheduler with constant probability, when $x = \Theta(y)$ in the initial configuration ($x$ and $y$ are used instead of $N_x$ and $N_y$, respectively).*

*Proof.* The probability of the minority to win is equal to the probability that the symmetric walk $(N_x, N_y)$ beginning from the initial point $(x_0, y_0)$ will meet the $N_y$ axis before meeting the $N_x$ axis. The particle moves to each of its 4 neighboring points with probability 1/4 (see Figure 1, where 0 and 1 are the probabilities that we assign to the boundaries that constitute the collection of points for which the system stabilizes to a winning vote). The only exception is when the number of $b$s becomes equal to zero. But in this case the $x$s decrease by one with probability 1/2, the same holds for the $y$s and with probability 1 a $b$ appears again and the walk returns to its initial symmetric distribution (to simplify the argument we ignore those states, because they do not strongly affect the probability that we want to compute). To the best of our knowledge, this kind of symmetric random walk in two dimensions has only been studied in [15], a paper cited by Feller [13], and is closely related to the Dirichlet problem. For any interior point $(x, y)$, if $u(x, y)$ denotes the probability that the minority wins (the walk meets the $N_y$ axis before meeting the $N_x$ axis), then

$$u(x,y) = \frac{1}{4}(u(x+1,y) + u(x,y+1) + u(x-1,y) + u(x,y-1)), \qquad (1)$$

and we are interested in the value of $u(x, y)$ when $x = \Theta(y)$, that is the initial number of $x$s and the initial number of $y$s are of the same order (e.g. $x = n/2$ and $y = n/4$). The homogeneous solution of (1) is $u(x, y) = \frac{x+y}{2n}$ and the general (with the boundary conditions into account) is $\frac{x+y}{2n} + f(x, y)$, where $f(x, y)$ is a particular non-homogeneous solution. When $x, y = \Theta(n)$ the $u(x, y)$ behaves as the homogeneous, thus $u(n/2, n/4)$ is equal to 3/8, which is constant. □

**Theorem 8.** *There exists at least one protocol w.r.t. which two fair probabilistic schedulers are not computationally equivalent.*

*Proof.* The Random Scheduler is not computationally equivalent to the Transition Function Scheduler w.r.t. the Majority Protocol, because there exists some initial margin in the case in which the majority is $x$, which is $\omega(\sqrt{n \log n})$ and also the initial number of $x$s and the initial number of $y$s are of the same order. For example, in the case in which $x = 3n/4 - k$ (where $k \ll n$) and $y = n/4$, $x$ and $y$ are of the same order and $x - y \simeq n/2 = \omega(\sqrt{n \log n})$ for sufficiently large $n$. But given an initial configuration satisfying the above dynamics, under the Random Scheduler the protocol w.h.p. stabilizes to a majority winning configuration, while under the Transition Function Scheduler from Lemma 1 there is a constant probability that the protocol will stabilize to a minority winning configuration. Thus, it does not hold that w.h.p. those schedulers make the protocol stabilize to the same output assignment (see again Definition 6). □

## 6 Future Research Directions

In the area initiated by the proposal of the Population Protocol (PP) model [2] many unresolved problems remain. The PP model makes absolutely minimal assumptions about the underlying system. The agents follow a completely unpredictable movement, they cannot store unique identifiers, and even a single Byzantine failure can lead to global failure of the system. How can we readjust (relax) those assumptions to more accurately reflect practical sensor network systems? For example in [14], Guerraoui and Ruppert assumed that the agents are equipped with read-only IDs (from the industry) and that they are also capable of storing a constant number of other agents' IDs. In this manner they obtained a very strong model, which they call the *Community Protocol* model, that can solve any decision problem in $NSPACE(n \log n)$ (and is additionally robust to Byzantine failures of a constant number of agents). In [11] they allowed the communication links to store states from a set of cardinality that is independent of the population size, to obtain the *Mediated Population Protocol* model that is also stronger than the PP model. In the case of wireless communication is there some architecture to reasonably implement the proposed model without using a global storage (for more information about the global storage idea the reader is referred to http://fronts.cti.gr/aigaion/?TR=65, i.e. the corresponding technical report of [11])? In the latter model either an exact characterization of the class of solvable problems has to be found or at least some impossibility results should appear to provide a first insight of what the model is incapable of computing (a first attempt can be found in [9], and in [11] it was proven that all stably computable predicates belong to $NSPACE(m)$, where $m$ denotes the number of edges of the communication graph). Finally, how can someone verify safely and quickly, in a distributed or centralized way, that a specific protocol meets its design objectives? This crucial problem remains open and has to be solved if our protocols are to be run in real critical application scenarios.

**Acknowledgements.** We wish to thank an anonymous reviewer who made very useful comments to a previous version of this work.

## References


1. D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, H. Jiang, and R. Peralta. Stably computable properties of network graphs. In Proc. Distributed Computing in Sensor Systems: 1st IEEE International Conference, pages 63-74, 2005.
2. D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *23rd Annual ACM Sympsium on Principles of Distributed Computing* (*PODC*), pages 290-299, New York, NY, USA, 2004. ACM.
3. D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4): 235-253, 2006.
4. D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3): 183-199, Sept. 2008.
5. D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing*, pages 292-299, 2006.
6. D. Angluin, J. Aspnes, and D. Eisenstat. A simple population protocol for fast robust approximate majority. In *21st International Symposium on Distributed Computing (DISC)*, volume 4731 of *Lecture Notes in Computer Science*, pages 20–32. Springer, 2007.
7. J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98-117, October 2007. Columns: *Distributed Computing*, Editor: M. Mavronicolas.
8. I. Chatzigiannakis and P. G. Spirakis. The dynamics of probabilistic population protocols. In *Distributed Computing*, *22nd International Symposium*, *DISC*, volume 5218 of *Lecture Notes in Computer Science*, pages 498-499, 2008.
9. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Decidable Graph Languages by Mediated Population Protocols. In *23nd International Symposium on Distributed Computing (DISC)*, Elche, Spain, Sept. 2009. (Also FRONTS Technical Report FRONTS-TR-2009-16, http://fronts.cti.gr/aigaion/?TR=80)
10. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Experimental verification and performance study of extremely large sized population protocols. FRONTS Technical Report FRONTS-TR-2009-3, http://fronts.cti.gr/aigaion/?TR=61, Jan. 2009.
11. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Mediated Population Protocols. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 363-374, Rhodes, Greece, 2009.
12. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Recent Advances in Population Protocols. In *34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, August 24-28, 2009, Novy Smokovec, High Tatras, Slovakia.
13. W. Feller. An Introduction to Probability Theory and Its Applications. Vol. 1, Wiley, 3rd Edition, 1968.
14. R. Guerraoui and E. Ruppert. Names Trump Malice: Tiny Mobile Agents Can Tolerate Byzantine Failures. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 484-495, Rhodes, Greece, 2009.
15. W. H. McCrea and F. J. W. Whipple. Random Paths in Two and Three Dimensions. *Proc. Roy. Soc. Edinburgh* 60, 281-298, 1940.
16. J.P. Queille and J. Sifakis. Fairness and Related Properties in Transition Systems - A temporal Logic to Deal with Fairness. Acta Informatica 19, 195-220, 1983.