Not All Fair Probabilistic Schedulers are Equivalent

Othon Michail, Ioannis Chatzigiannakis,
Shlomi Dolev, Sándor Fekete, and Paul Spirakis

Research Academic Computer Technology Institute (RACTI)
Department of Computer Science, Ben-Gurion University of the Negev
Department of Computer Science, Braunschweig University of Technology

OPODIS '09
December 2009

Outline I

1. Schedulers
   - Intro - Population Protocols
   - Fair Probabilistic Schedulers
   - Proposed Schedulers

2. Equivalence
   - Time Equivalence
   - Computational Equivalence

Population Protocol Model

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

- A **communication graph** $G = (V, E)$, where V is a **population** of n **agents** (finite-state machines with sensing capabilities) and E represents the permissible interactions.
- Agents interact in pairs under the commands of some **adversary scheduler**.
  - The scheduler is only required to be **fair**.
- During interaction the states of the agents are updated according to a **global transition function** δ.
- A **configuration** $C : V \to Q$ is simply a snapshot of the population states.

The Transition Graph

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, Dist. Comp. '06]

$T(\mathcal{A}, G)$, $\mathcal{A}$ is a **population protocol** and G is a **communication graph**.

- **node set $V(T)$:** set of all possible configurations $\mathcal{C} = Q^V$.
- **edge set $E(T)$:** $(C, C') \in E(T)$ iff $C \rightarrow C'$.
- directed graph, possibly containing self-loops.

Probabilistic Schedulers

Definition (Generic Definition of Probabilistic Schedulers)

A **probabilistic scheduler** defines for each $C \in V(T)$ an infinite sequence of probability distributions $(d_1^C, d_2^C, \ldots)$ over the set $\Gamma^+(C) = \{C' \mid C \to C'\}$.

Thus, for all $t \in \mathbb{Z}^+$ and $C \in V(T)$ it holds that

- $d_t^C : \Gamma^+(C) \to [0,1]$, and
- $\sum_{C' \in \Gamma^+(C)} d_t^C(C') = 1$.

$d_l^C(C')$: denotes the probability that C goes to C' when C is encountered for the lth time during the execution.

Consistent Schedulers

Definition (Consistency)

We call a probabilistic scheduler **consistent**, w.r.t. a transition graph $T(\mathcal{A}, G)$, if for all $C \in V(T)$ and all $t, t' \in \mathbb{Z}^+$ it holds that

$$d_t^C = d_{t'}^C = d^C.$$

- That is, 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).

Consistent Schedulers

- A consistent scheduler is simply a **labeling** $P : E(T) \to [0,1]$ on the arcs of T,
  - such that $\sum_{j \in \Gamma^+(i)} P(i,j) = 1$ for any $i \in V(T)$.
- Let C be the current configuration.
- Here the probability distribution according to which the next neighbor-configuration is selected, is **invariant of the number of times C has already occured in the execution**.
- It is defined by the labels of the arcs leaving C.

Remark

By removing all $e \in E(T)$ for which $P(e) = 0$ we obtain the **underlying graph of a finite Markov chain** where the state space is $\mathcal{C}$ and for all $i, j \in \mathcal{C}$, if $i \to j$ then $\mathbb{P}_{ij} = P(i,j)$, otherwise $\mathbb{P}_{ij} = 0$.

Fair Consistent Schedulers

Fair Execution (Computation): If $C \in \mathcal{C}$ appears infinitely often in the execution and $C \rightarrow C'$ then C' also appears infinitely often in the execution.

Theorem

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* $i \rightarrow j$ *and* $i \neq j$, *is* **fair with probability 1**.

Proof.

If i is persistent then all its out-neighbors are persistent with probability 1 because they occur with nonzero probability. □

Remark: This holds also if we require $\mathbb{P}_{ij} > 0$ only for persistent configurations i.

Random Scheduler

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Let E be the set of edges of the communication graph G.

Definition (Random Scheduler)

Under C_i picks an ordered pair of agents $(u, \upsilon) \in E$ at random, independently and uniformly. Then the agents generate C_{i+1} by applying the transition function to $(C_i(u), C_i(\upsilon))$.

- It is a **protocol-oblivious** (or **agnostic**) scheduler.
  - **Constructs the interaction pattern without any knowledge on the protocol executed.**

The Random Scheduler is Fair

Theorem

The Random Scheduler is fair with probability 1.

Proof.

1. **It is consistent:** If i is current configuration and $i \rightarrow j$, j is selected with probability $\mathbb{P}_{ij} = |K_{ij}|/|E|$, $K_{ij} = \{e \mid e \in E \text{ and } i \xrightarrow{e} j\}$.
2. For all $i, j \in \mathcal{C}$ s.t. $i \rightarrow j$, from definition of '$\rightarrow$' $\Rightarrow \exists e \in E$ s.t. $i \xrightarrow{e} j$, thus $|K_{ij}| > 0$ and $\mathbb{P}_{ij} > 0$.

$\square$

State Scheduler

$(q, q') \in Q^2$ **interaction candidate under** C: $\exists (u, \upsilon) \in E$ s.t. $C(u) = q$ and $C(\upsilon) = q'$.

Definition (State Scheduler)

Under C_i draws an interaction candidate (q, q') uniformly at random and then a $(u, \upsilon) \in E$ s.t. $C_i(u) = q$ and $C_i(\upsilon) = q'$ uniformly at random. Then the agents generate C_{i+1} by applying the transition function to $(C_i(u), C_i(\upsilon))$.

- It is a **protocol-aware** scheduler.
  - **Takes into account information concerning the underlying protocol.**
  - It **inspects the protocol's set of states** Q.

Transition Function Scheduler

$\dot{\delta}$: the **reflexive reduction** of the binary transition relation δ over Q^2; identity rules are excluded.

Definition (Transition Function Scheduler)

Under C_i draws a $((q_1, q_2), (q_1', q_2')) \in \dot{\delta}$, s.t. (q_1, q_2) is an interaction candidate, uniformly at random (if no such exists, becomes Random Scheduler and remains in C_i forever) and then a $(u, \upsilon) \in E$ s.t. $C_i(u) = q_1$ and $C_i(\upsilon) = q_2$ uniformly at random. Then the agents generate C_{i+1} by applying the transition relation to $(C_i(u), C_i(\upsilon))$.

- It is a **protocol-aware** scheduler.
  - It **inspects the protocol's transition relation δ and set of states Q**.

State Scheduler and Transition Function Scheduler are Fair

Theorem

The State Scheduler and the Transition Function Scheduler are both fair with probability 1.

Proof.

Simply show that both are consistent and assign nonzero probabilities to all edges of the transition graph that are not self-loops. □

Time Equivalent Schedulers

Definition (Time Equivalent Schedulers)

Two fair probabilistic schedulers S_1 and S_2 are called **time equivalent** w.r.t. a protocol $\mathcal{A}$ iff, for any initial configuration C_0, the expected time (number of steps) to convergence of $\mathcal{A}$ under S_1 when the initial configuration is C_0 is asymptotically the same as the expected time to convergence under S_2 when the initial configuration is again C_0.

Time Equivalence

Theorem

Not all fair probabilistic schedulers are time equivalent.

Proof.

Consider the **OR protocol**:

$$
\begin{array}{ll}
(x,x) \to (x,x) & (y,x) \to (y,y) \\
(x,y) \to (y,y) & (y,y) \to (y,y)
\end{array}
$$

- N_y: number of ys in the initial configuration.
- If $N_y = 0$ or $N_y = n$, then the system is already stable (0 steps).

Time Equivalence

- If $0 < N_y < n$, then
  1. **Transition Function Scheduler:** can only choose from the rules $(x, y) \rightarrow (y, y)$ and $(y, x) \rightarrow (y, y)$, thus in each step increases the number of ys by one, and takes $n - N_y$ steps to convergence.
  2. **State Scheduler:** progress is always made with probability $1/2$, because the lhs (x, y) and (y, x) are always interaction candidates, thus $2(n - N_y)$ steps to convergence.
- **Corollary:** The State Scheduler and the Transition Function Scheduler are time equivalent.
- In fact, both have **optimal behavior** w.r.t. to the OR protocol by exploiting the fact that they are protocol-aware.

Time Equivalence

Now, consider the **Modified Scheduler**:

- The same as the State Scheduler except for the fact that it selects from the class of identity interaction candidates with probability $1 - \varepsilon$ and from all the remaining rules with probability ε, where $0 < \varepsilon < 1$.
- It is also fair with probability 1.
- It turns out that when $N_y = 2$, the Modified Scheduler needs an expected number of $\frac{n-2}{\varepsilon}$ steps to convergence.
  - But $\frac{n-2}{\varepsilon}$ can be made **arbitrarily large**, because ε can be arbitrarily close to 0.
- Implies that **the Modified Scheduler is not time equivalent to the others** (being aware of the protocol is not always an advantage).

Computationally Equivalent Schedulers

Definition (Computationally Equivalent Schedulers)

Two fair probabilistic schedulers S_1 and S_2 are called **computationally equivalent** w.r.t. a protocol $\mathcal{A}$ iff, for any initial configuration C_0, $\mathcal{A}$ under S_1, when the initial configuration is C_0, stabilizes w.h.p. to an output assignment $y \in Y^V$ iff $\mathcal{A}$ under S_2, when the initial configuration is again C_0, stabilizes w.h.p. to the same output assignment.

Computational Equivalence

Theorem

Not all fair probabilistic schedulers are computationally equivalent.

Proof.

Consider the **MAJORITY protocol**:

$$\begin{array}{ll} (x,b) \rightarrow (x,x) & (x,y) \rightarrow (x,b) \\ (y,b) \rightarrow (y,y) & (y,x) \rightarrow (y,b) \end{array}$$

- Under the Random Scheduler w.h.p. the majority wins provided that its initial margin is $\omega(\sqrt{n \log n})$ [Angluin, Aspnes, and Eisenstat, DISC '07].
- Under the Transition Function Scheduler, for the same margin, the majority may lose with constant probability. □

Conclusions

- For most natural systems of this kind we will know some **probabilistic mobility patterns** (or possibly some good approximations for them).
- The first crucial question will always be: **“Are those patterns fair?”**
- We provided a fairly easy to use, **general theoretical framework for proving probabilistic fairness**.
- We defined the **protocol-aware** and **protocol-oblivious** **scheduler classes**, and proposed three protocol-aware schedulers that are **fair**.

Conclusions

- We defined equivalence between schedulers w.r.t. to performance and computability, and proved that **not all fair probabilistic schedulers are equivalent**.
- This implies something fundamental: **Even minor modifications of the mobility pattern may render our protocols useless.**
- **Is there some other stronger notion of fairness that prohibits such modifications?**
- In fact, **most natural systems do not follow a unique mobility pattern**.
    - Agents may **skip interactions** due to **battery or signal degradation**.
    - **Agent's carriers may change their mobility habits** due to various reasons, e.g. **environment modification**.
- **How can we make our protocols adapt to such changes in order to continue being fast and/or correct?**

FRONTS

- This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).
- FRONTS is a joint effort of eleven academic and research institutes in foundational algorithmic research in Europe.
- The effort is towards establishing the **foundations of adaptive networked societies of tiny artefacts**.

SEVENTH FRAMEWORK PROGRAMME

Thank You!