Stably Computing Order Statistics with Arithmetic Population Protocols*

George B. Mertzios[1], Sotiris E. Nikoletseas[2], Christoforos L. Raptopoulos[3], and Paul G. Spirakis[4]

1 **School of Engineering and Computing Sciences, Durham University, UK**
george.mertzios@durham.ac.uk

2 **Computer Engineering & Informatics Department, University of Patras, Patras, Greece; and**
Computer Technology Institute & Press "Diophantus", Patras, Greece
nikole@cti.gr

3 **Computer Engineering & Informatics Department, University of Patras, Patras, Greece; and**
Computer Technology Institute & Press "Diophantus", Patras, Greece
raptopox@ceid.upatras.gr

4 **Computer Engineering & Informatics Department, University of Patras, Patras, Greece; and**
Computer Technology Institute & Press "Diophantus", Patras, Greece; and
Department of Computer Science, University of Liverpool, Liverpool, UK
p.spirakis@liverpool.ac.uk

Abstract

In this paper we initiate the study of populations of agents with very limited capabilities that are globally able to compute order statistics of their arithmetic input values via pair-wise meetings. To this extent, we introduce the *Arithmetic Population Protocol (APP)* model, embarking from the well known Population Protocol (PP) model and inspired by two recent papers [1, 11] in which states are treated as integer numbers. In the APP model, every agent has a state from a set $\mathcal{Q}$ of states, as well as a fixed number of registers (independent of the size of the population), each of which can store an element from a *totally ordered* set $\mathcal{S}$ of samples. Whenever two agents interact with each other, they update their states and the values stored in their registers according to a *joint transition function*. This transition function is also restricted; it only allows (a) comparisons and (b) copy / paste operations for the sample values that are stored in the registers of the two interacting agents. Agents can only meet in pairs via a fair scheduler and are required to eventually converge to the same output value of the function that the protocol globally and stably computes. We present two different APPs for stably computing the median of the input values, initially stored on the agents of the population. Our first APP, in which every agent has 3 registers and no states, stably computes (with probability 1) the median under *any* fair scheduler in *any* strongly connected directed (or connected undirected) interaction graph. Under the probabilistic scheduler, we show that our protocol stably computes the median in $O(n^6)$ number of interactions in a connected undirected interaction graph of n agents. Our second APP, in which every agent has 2 registers and $O(n^2 \log n)$ states, computes to the correct median of the input with high probability in $O(n^3 \log n)$ interactions, assuming the probabilistic scheduler and the complete interaction graph. Finally we present a third APP which, for any k, stably computes the kth smallest element of the input of the population under *any* fair scheduler and in *any* strongly connected directed (or connected undirected) interaction graph. In this APP

* Sotiris E. Nikoletseas and Christoforos L. Raptopoulos were partially supported by the MULTIPLEX project – 317532. Paul G. Spirakis was partially supported by the MULTIPLEX project – 317532 and the EEE/CS School initiative NeST of the University of Liverpool.

41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016).
Editors: Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier; Article No. 68; pp. 68:1–68:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

every agent has 2 registers and n states. Upon convergence every agent has a different state; all these states provide a *total ordering* of the agents with respect to their input values.

1998 ACM Subject Classification I.2.11 Distributed Artificial Intelligence - Multiagent systems

Keywords and phrases arithmetic population protocols, order statistics, median, k-minimum element

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.68

1 Introduction

The *population protocol (PP)* model [2, 5] was originally defined to represent sensor networks consisting of very limited mobile agents with no control over their own movement. It has been defined by analogy to population processes [8] in probability theory and has already been used in various fields, such as in statistical physics, genetics, epidemiology, chemistry and biology [6]. An exciting feature of such systems of simple agents is that, by exchanging information through local pairwise interactions (rendezvous), the entities can collectively perform significant global computational tasks. In the basic model, in each pairwise interaction both participating agents update their state according to a (pre-specified) joint transition function that only depends on the states of the two agents. Furthermore the interactions between agents happen under some kind of a *fairness* condition. Assuming the complete interaction graph, it is known that population protocols compute exactly the class of semilinear predicates [3], i.e., the predicates definable in first-order logic Presburger arithmetic. In addition, it is well known that population protocols do not compose, except for certain cases; for a survey we refer to [5, 10].

One limitation of the PP model is that, as every agent is characterized at every time point by its state, the predicates that can be stably computed are defined over variables that count the number of nodes having initially some specific state from the set of allowable states (see for example [4, 9]). However, in several circumstances, it is natural to assume that nodes may also store some *arithmetic* values, for instance a temperature or some other local measurement. Imagine for instance a huge network of elementary sensors that measure local temperature and want to compute some statistical function of the temperatures of the whole network. The computation of elementary functions (e.g. the median) of such arithmetic values by a population of autonomous agents requires an extension of the standard PP model. On the other hand, it is worth noting that computations of statistics in classical distributed network models have been already considered, e.g. [7].

In order to overcome such limitations, we introduce in this paper the *Arithmetic Population Protocols (APP)* model, which is inspired by the PP model and by two recent papers [1, 11]. Similarly to the spirit of population protocols, we still assume in arithmetic population protocols that agents are weak computational devices with very small local memory, thus still staying on the pragmatic side. In [11], the authors consider the problem of determining the exact difference between the majority and the minority type in a two-type population. The basic idea of their protocol generalizes an idea of [1], where the set of states available to each agent was a set of integers, thus implying a total ordering of the states. In particular, an agent at state i was considered to be more firm in supporting her type than another agent at state i', with $|i| > |i'|$.

In our APP model, every agent has a state from a set $\mathcal{Q}$ of states, as well as $r \geq 1$ registers, each of which can store an element from a *totally ordered* set $\mathcal{S}$ of samples (e.g. $\mathcal{S} \subseteq \mathbb{N}$

or $\mathcal{S} \subseteq \mathbb{R}$). We assume that r is a fixed constant, i.e., independent of the population size. Furthermore, agents have limited knowledge of the set $\mathcal{S}$ and limited computational power over $\mathcal{S}$. In particular, no agent is aware of the whole set $\mathcal{S}$. Furthermore, whenever two agents interact with each other, they update their states and the values stored in their registers according to a *joint transition function*. This transition function only allows (a) comparisons and (b) copy / paste operations for values in $\mathcal{S}$ that are stored in the registers of the two interacting agents. Initially, each agent $v \in \mathcal{V}$ is given an element $x_v \in \mathcal{S}$ as input; her initial state and the initial values of her registers are then determined according to x_v. The goal is that eventually, after a sequence of local pairwise interactions (which are planned by a *scheduler* that satisfies a general "fairness" condition), every agent computes the same function on the input values in one of her registers. We assume that agents are oblivious of their own identity and of identities of other agents they interact with. Therefore, at any time, each agent is completely characterized by the values stored in its registers, together with its state.

1.1 Our model

More formally, let $\mathcal{S}$ denote a totally ordered set of *samples* and let $\mathcal{Q}$ be a totally ordered[1] set of states. We assume that there is a population $\mathcal{V}$ of computationally weak agents with limited memory. In particular, each agent can be in one of the states in $\mathcal{Q}$ and has r registers at its disposal, each one of which can store an element of $\mathcal{S}$.

For any $t \geq 0$ let $\mathbf{R}(t)$ be a $|\mathcal{V}| \times r$ matrix, such that $\mathbf{R}_{v,j}(t)$ is the value of the j-th register of agent $v \in \mathcal{V}$ at time t. Furthermore, for every $t \geq 0$, let $\mathbf{q}(t)$ be a $|\mathcal{V}|$-dimensional vector such that $\mathbf{q}_v(t)$ is the state of agent $v \in \mathcal{V}$ at time t. We refer to $\mathbf{C}(t) \overset{def}{=} (\mathbf{R}(t), \mathbf{q}(t))$ as the *configuration* at time t. We will assume that the number of registers r of each agent is a fixed constant independent of the population size, i.e., every agent has a limited number of registers available. For every agent $v \in \mathcal{V}$, we will refer to the first register of v as the *input register*; we also refer to $\mathbf{R}_{v,1}(t)$ as the *value of the input register* of v. Furthermore, we will say that $\mathbf{R}_{:,1}(0)$ (i.e., the first column of $\mathbf{R}(0)$) is the *population input* at time 0.

An *Arithmetic Population Protocol (APP)* is defined on a population $\mathcal{V}$ of agents and consists of an *input initialization function* $\iota : \mathcal{S} \to \mathcal{S}^r \times \mathcal{Q}$, an *output function* $\gamma : \mathcal{S}^r \times \mathcal{Q} \to \mathcal{D}$ ($\mathcal{D}$ is the set of output values; usually it is the same as the sample set $\mathcal{S}$), and a *joint transition function* $f : (\mathcal{S}^r \times \mathcal{Q}) \times (\mathcal{S}^r \times \mathcal{Q}) \to (\mathcal{S}^r \times \mathcal{Q}) \times (\mathcal{S}^r \times \mathcal{Q})$. In particular, if v interacts with u at time $t+1$, then the new values of their registers and states become $(\mathbf{R}_{v,:}(t+1), \mathbf{q}_v(t+1))$ and $(\mathbf{R}_{u,:}(t+1), \mathbf{q}_u(t+1))$ respectively, where $(\mathbf{R}_{v,:}(t+1), \mathbf{q}_v(t+1), \mathbf{R}_{u,:}(t+1), \mathbf{q}_u(t+1)) = f(\mathbf{R}_{v,:}(t), \mathbf{q}_v(t), \mathbf{R}_{u,:}(t), \mathbf{q}_u(t))$. As also mentioned earlier, the transition function f is *not arbitrary*: it only allows (a) comparisons and (b) copy / paste operations for values in $\mathcal{S}$ that are stored in the registers of the two interacting agents. Therefore, if agents v, u interact at time $t+1$, then for any register $j \in [r]$, we have $\mathbf{R}_{v,j}(t+1), \mathbf{R}_{u,j}(t+1) \in \{\mathbf{R}_{x,j'}(t) : x \in \{v,u\}, j' \in [r]\}$.

Initially, the values of the registers of each agent are determined by the input initialization function ι, i.e., for each agent $v \in \mathcal{V}$ that has input $x_v \in \mathcal{S}$, we initially set $(\mathbf{R}_{v,:}(0), \mathbf{q}_v(0)) = \iota(x_v)$, where $\mathbf{R}_{v,:}(0)$ denotes the row of $\mathbf{R}(0)$ corresponding to agent v. Subsequently, in every time step $t+1 \geq 1$, a pair of agents interacts and updates the values of their registers according to the transition function f.

[1] This assumption on $\mathcal{Q}$ is not necessary for our results, but is given for the sake of presentation.

Agent pairwise interactions are planned by a *scheduler* under a general "fairness" condition; the actual mechanism for choosing which agents interact each time is abstracted away. The fairness condition states that *the scheduler cannot postpone a possible finite sequence of agent interactions indefinitely.* The *direction of interaction* may or may not be relevant (see also the discussion below on the probabilistic scheduler); if it is not relevant, we say that the APP is *symmetric.*

Due to lack of coordination and storage restrictions, the agents in a population executing an APP cannot determine when the computation has finished. Instead, the values of the output registers of every agent is required to converge to a common (correct) value. More formally, we have the following:

▶ **Definition 1** (Stable computation). Let $\mathcal{M}$ be the (infinite) set of multi-sets of a (finite or infinite) totally ordered set $\mathcal{S}$ and let $\mathcal{F} : \mathcal{M} \to \mathcal{D}$ be a function. We say that an APP *stably computes* the function $\mathcal{F}$ if and only if under *any fair scheduler*, for *any multi-set* $M \in \mathcal{M}$, and starting from *any configuration* where the elements in M are assigned bijectively to the input registers of the agents in a population $\mathcal{V}$ (where $|\mathcal{V}| = |M|$) we have that, after a finite time τ, $\gamma(\mathbf{R}_{v,:}(t), \mathbf{q}_v(t)) = \mathcal{F}(M)$, for all agents $v \in \mathcal{V}$ and for all $t \geq \tau$.

To allow for the comparison of our APPs in terms of the number of pairwise interactions needed to compute some function, we will consider in this paper a special case of a fair scheduler, namely the *probabilistic scheduler*, which is defined on directed interaction graphs as follows. In each time step a directed edge (v, u) of the interaction graph is chosen uniformly at random, where v (i.e., the tail of (v, u)) is called the *initiator* and u (i.e., the head of (v, u)) is called the *responder* of the interaction. Then, agents v and u update the values of their registers jointly according to the transition function f. The direction of interaction plays an important role in the general case, as the initiator will update its register values according to the first part of the outcome of f (consisting of r numeric values), while the responder will use the second part of the outcome (which also consists of r numeric values). The values of the registers of all other agents remain unchanged. The probabilistic scheduler is defined on undirected graphs similarly, by replacing every undirected edge $\{v, u\}$ by the two directed edges (v, u) and (u, v).

1.2 Our contribution

In this paper, we initiate the study of populations of agents with very limited capabilities that are globally able to compute order statistics of their input via pair-wise meetings. We initially focus on the fundamental problem of computing the *median* of the input values in a population of n agents, which is defined as the $\lceil n/2 \rceil$-th minimum element of the input values. We provide two different APPs for stably computing the median.

In our first APP (see Section 2) every agent has 3 registers and no states. This APP stably computes (with probability 1) the median under *any* fair scheduler in *any* strongly connected directed (or connected undirected) interaction graph. We also show that, under the probabilistic scheduler, our protocol stably computes the median in $O(n^6)$ number of interactions in a connected undirected interaction graph of n agents.

Our second APP for stably computing the median (see Section 3) is considerably faster than the first one, however it works with high probability (rather than with probability 1) and it requires additional assumptions on the scheduler and the structure of the underlying interaction graph. In particular, in our second APP for the median, every agent has 2 registers and $O(n^2 \log n)$ states. Assuming the probabilistic scheduler and the complete interaction graph, this APP converges to the correct median of the input of the population with high probability in $O(n^3 \log n)$ interactions. Additionally, agents are required to know the size of the population. The latter assumption can be dropped by assuming additional states and computational power for the agents.

As our final contribution, we present in Section 4 an APP which, for any k, stably computes the kth smallest element of the input of the population under *any* fair scheduler and in *any* strongly connected directed (or connected undirected) interaction graph. In this APP every agent has 2 registers and n states. Upon convergence every agent has a different state; all these states provide a *total ordering* of the agents with respect to their input values.

2 A 3-register APP for median

In this section we describe a APP using 3 registers and no states that stably computes the median of the input of a population of agents $\mathcal{V}$, i.e., the median of the (multi-) set of elements $\{x_v : v \in \mathcal{V}\} \in \mathcal{S}^n$. We will assume without loss of generality that the set of measurements $\mathcal{S}$ is the set of real numbers, i.e., $\mathcal{S} = \mathbb{R}$. Our protocol is not symmetric, hence in every interaction we distinguish between initiator and responder. For the sake of clarity of presentation, we will assume that the size $n = |\mathcal{V}|$ of the population is odd, so that the median is well (and uniquely) defined. In fact, without further modifications, our protocol may not converge if there is an even number of nodes (because in this case there are 2 candidates for the median).[2] Additionally, we will initially assume that the interaction graph is complete and we later show how this assumption can be dropped.

For every agent $v \in \mathcal{V}$, the first register will contain the (input) number x_v initially assigned to node v (i.e., $\mathbf{R}_{v,1}(0) = x_v$); the value of $\mathbf{R}_{v,1}(t)$ will remain unchanged throughout the computation. The third register of each agent v, i.e., $\mathbf{R}_{v,3}$, will eventually converge to median of the input of the population $\{x_v : v \in \mathcal{V}\}$. The first two registers (i.e., $\mathbf{R}_{v,1}, \mathbf{R}_{v,2}$) are used to make virtual connections between agents; upon convergence, the agent with the smallest number will be virtually connected to the agent with the largest number, the agent with the second smallest number will be virtually connected to the agent with the second largest number and so on. Furthermore, for any agent $v \in \mathcal{V}$, we will denote by $I_v(t) \overset{def}{=} \{s \in \mathbb{R} : \min(\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t)) \leq s \leq \max(\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t))\}$. For simplicity, we will write $I_v(t) = [\min(\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t)), \max(\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t))]$ and we will refer to $I_v(t)$ as the *closed interval* of v at time t (we use this term loosely, since in general the set of measurements may not be compact). Finally, we denote $\mathbf{C}_v(t) \overset{def}{=} (\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t), \mathbf{R}_{v,3}(t))$.

The initialization function of our protocol is given by $\iota_{\text{MDN}}(x_v) = \mathbf{C}_v(0) = [x_v, x_v, x_v]$, for any $x_v \in \mathcal{S}$ and $v \in \mathcal{V}$, and the output function is given by $\gamma_{\text{MDN}}(\mathbf{C}_v(t)) = \mathbf{R}_{v,3}(t)$, for any $v \in \mathcal{V}$. Consequently, for any $v \in \mathcal{V}$, the closed interval $I_v(0)$ contains just one point. The joint transition function is defined as follows: if agent v (the initiator) interacts with u (the responder) at time $t + 1 = 1, 2, \ldots$, then $(\mathbf{C}_v(t+1), \mathbf{C}_u(t+1)) = f_{\text{MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t))$, where f_{MDN} is given below.

[2] Nevertheless, it is not difficult to extend our APP so that it converges to one of the 2 candidate values (say the smallest) for the median in the case where n is even.

```
Transition function $f_{\mathrm{MDN}}$
Input: $\mathbf{C}_v(t), \mathbf{C}_u(t)$
```

Case I: $\mathbf{R}_{v,1}(t) = \mathbf{R}_{v,2}(t)$ AND $I_v(t) \subseteq I_u(t)$.

$$f_{\mathrm{MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = ([\mathbf{C}_v(t)], [x_u, \mathbf{R}_{u,2}(t), x_v])$$

Case II: $I_v(t) \nsubseteq I_u(t)$ AND $I_u(t) \nsubseteq I_v(t)$ AND $x_v \in \{r_2, r_3\}$ AND $x_u \in \{r_1, r_4\}$, where r_1, r_2, r_3 and r_4 are the first, second, third and fourth smallest value in the set $\{\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t), \mathbf{R}_{u,1}(t), \mathbf{R}_{u,2}(t)\}$ respectively.

$$f_{\mathrm{MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = ([x_v, \{r_2, r_3\} \backslash x_v, \mathbf{R}_{v,3}(t)], [x_u, \{r_1, r_4\} \backslash x_u, \mathbf{R}_{u,3}(t)])$$

Case III: Every other case.

$$f_{\mathrm{MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = (\mathbf{C}_v(t), \mathbf{C}_u(t))$$

We can prove the following:

▶ **Theorem 2** (Correctness). *The APP with initialization function ι_{MDN}, output function γ_{MDN} and transition function f_{MDN} stably computes the median function when the underlying interaction graph is the complete graph.*

Proof. Let $n = |\mathcal{V}|$ and define the potential function $\phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as follows:

$$\phi(\mathbf{R}_{:,1}(t), \mathbf{R}_{:,2}(t)) = \sum_{v \in \mathcal{V}} (\mathbf{R}_{v,2}(t) - \mathbf{R}_{v,1}(t))^2.$$

Clearly, ϕ remains unchanged if two agents interact according to one of the cases I, or III, since the values of the first two registers of the agents remain unchanged. On the other hand, ϕ strictly increases in case II. Indeed, suppose that agent v (the initiator) interacts with agent u (the responder) at time $t+1$ according to case II, and let r_1, r_2, r_3 and r_4 be the first, second, third and fourth smallest value in the set $\{\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t), \mathbf{R}_{u,1}(t), \mathbf{R}_{u,2}(t)\}$. Then

$$\begin{aligned}
&\phi(\mathbf{R}_{:,1}(t+1), \mathbf{R}_{:,2}(t+1)) - \phi(\mathbf{R}_{:,1}(t), \mathbf{R}_{:,2}(t)) \\
&\quad = \left[(r_3 - r_2)^2 + (r_4 - r_1)^2\right] - \left[(\mathbf{R}_{v,2}(t) - \mathbf{R}_{v,1}(t))^2 + (\mathbf{R}_{u,2}(t) - \mathbf{R}_{u,1}(t))^2\right] \\
&\quad = 2\left(\mathbf{R}_{v,1}(t)\mathbf{R}_{v,2}(t) + \mathbf{R}_{u,1}(t)\mathbf{R}_{u,2}(t) - r_2 r_3 - r_1 r_4\right) > 0. \qquad (1)
\end{aligned}$$

Notice also that, by the definition of ι_{MDN} and f_{MDN}, our protocol maintains the following invariant:

(INV) For any $v \in \mathcal{V}$ and any time $t \geq 0$, there are exactly 2 copies of x_v, one in $\mathbf{R}_{:,1}(t)$ and one in $\mathbf{R}_{:,2}(t)$ (obviously, if two or more nodes have the same input, there will be two copies for each).

Let $0 \leq i \leq \lceil \frac{n}{2} \rceil$. We say that a configuration $\mathbf{C}(t)$ is *non-crossing up to i* if, for every $1 \leq k \leq i$, the agent with the kth smallest input value is matched in an interval with the kth largest input value. That is, a configuration $\mathbf{C}(t)$ is non-crossing up to i if there are sets of distinct agents $L^{(i)}$ and $U^{(i)}$ for which the following conditions hold:

(C1) $L^{(i)} = \{z_1, \ldots, z_i\} \subseteq \mathcal{V}$, such that $x_{z_1} \leq \cdots \leq x_{z_i}$ and, for each $z \in L^{(i)}$ and $z' \in \mathcal{V} \backslash L^{(i)}$, we have $x_z \leq x_{z'}$.

(C2) $U^{(i)} = \{u_1, \ldots, u_i\} \subseteq \mathcal{V}$, such that $x_{u_1} \geq \cdots \geq x_{u_i}$ and, for each $u \in U^{(i)}$ and $u' \in \mathcal{V} \backslash U^{(i)}$, we have $x_u \geq x_{u'}$.

(C3) $I_{z_j}(t) = I_{u_j}(t) = [z_j, u_j]$, for all $j \in [i]$.

The sets $L^{(i)}, U^{(i)}$ will be called *witnesses* for the fact that $\mathbf{C}(t)$ is non-crossing up to i. We can prove the following:

▶ **Claim 3.** *Let $i < \lceil \frac{n}{2} \rceil$. If $\mathbf{C}(t)$ is non-crossing up to i but not up to $i+1$, then there is a pair of agents that can interact according to Case II. Equivalently, there is an interaction of agents that can strictly increase the value of ϕ.*

Proof. Let i be the maximum index such that $\mathbf{C}(t)$ is non-crossing up to i and let $L^{(i)}, U^{(i)}$ be witnesses of this fact. Let $L^+ \overset{def}{=} \{z : z \notin L^{(i)}, x_z \leq x_{z'}$, for every $z' \in \mathcal{V} \backslash L^{(i)}\}$ and $U^- \overset{def}{=} \{u : u \notin U^{(i)}, x_u \geq x_{u'}$, for every $u' \in \mathcal{V} \backslash U^{(i)}\}$. Notice that, by definition, x_z (respectively x_u) is the same for all $z \in L^+$ (respectively $u \in U^-$).

The fact that $\mathbf{C}(t)$ is non-crossing up to i (and not up to $i+1$) implies that at least one of the following holds:

(i) For all agents $z \in L^+$, there is no agent $u \in U^-$, such that $I_z(t) = [x_z, x_u]$.
(ii) For all agents $u \in U^-$, there is no agent $z \in L^+$, such that $I_u(t) = [x_z, x_u]$.

Assume without loss of generality that (i) holds; the case where (ii) holds is similar, by symmetry. Let $z_{\max} \in L^+$ be such that the length of the interval $I_{z_{\max}}(t)$ is maximal. In particular, this means that there is some agent w such that $I_{z_{\max}}(t) = [x_{z_{\max}}, x_w]$, with $x_{z_{\max}} \leq x_w < x_u$, for each $u \in U^-$. Indeed, by assumption we have that $x_w \neq x_u$, for each $u \in U^-$. But also, we cannot have $x_w > x_u$ or $x_w < x_{z_{\max}}$, by definition of the sets L^+, U^-.

Notice now that, by invariant (INV) there also exists a pair of agents $z', w' \notin L^{(i)} \cup U^{(i)}$, such that $I_{z'}(t) = [x_{z'}, x_{w'}]$, with $x_{w'} > x_w$ (and $x_{z'} \leq x_{w'}$). Indeed, let $I_w(t)$ be the interval of node w at time t. If $\max(\mathbf{R}_{w,1}(t), \mathbf{R}_{w,2}(t)) > x_w$, then we have $z' = w$. Otherwise, let x_y be the other endpoint of $I_w(t)$ (in fact $x_y = \min(\mathbf{R}_{w,1}(t), \mathbf{R}_{w,2}(t))$. Similarly, if $\max(\mathbf{R}_{y,1}(t), \mathbf{R}_{y,2}(t)) > x_w$, then we have $z' = y$. Proceeding inductively, we will eventually find an agent z' with $\max(\mathbf{R}_{z',1}(t), \mathbf{R}_{z',2}(t)) > x_w$. Notice also that z' does not belong to U^-, by construction, but also neither to L^+, by maximality of $I_{z_{max}}(t)$.

We can then see that the pair of agents $(z', z_{\max})$ is suitable for interaction according to Case II. Indeed, $\mathbf{R}_{z',1}(t) = x_{z'} > x_{z_{\max}} = \mathbf{R}_{z_{\max},1}(t)$, so $I_{z_{\max}}(t) \nsubseteq I_{z'}(t)$, $\mathbf{R}_{z',2}(t) = x_{w'} > x_w = \mathbf{R}_{z_{\max},2}(t)$, so $I_{z'}(t) \nsubseteq I_{z_{\max}}(t)$ and finally $x_{z'} \in \{r_2, r_3\} = \{x_{z'}, x_w\}$ and $x_{z_{max}} \in \{r_1, r_4\} = \{x_{z_{max}}, x_{w'}\}$. This completes the proof of the claim, since by equation (1), any interaction according to Case II strictly increases the value of ϕ. ◀

By the above claim, we can now show that the potential function ϕ is maximized whenever the system reaches a configuration $\mathbf{C}(t)$ that is non-crossing up to $\lceil \frac{n}{2} \rceil$. Indeed, by inequality (1), any interaction according to case II strictly increases the value of the potential ϕ, and since this increment is independent of t and the maximum value of ϕ is finite (e.g. it is at most $n(\max\{x_v\} - \min\{x_v\})^2$), we conclude that, in finite time (because of the fairness assumption of the scheduler) we will have reached the desired configuration. Finally, notice that in such a configuration, the node that corresponds to the median will have the same value stored in both its first two registers (by the assumption that n is odd). In particular, whenever it interacts as an initiator with another agent, it will do so according to case I. In fact, no agent with assigned value different than the median will be able to interact with other nodes according to this case after convergence of the protocol. This implies that, eventually, all agents will have the value of the correct median stored in their third register. This completes the proof. ◀

Notice that our protocol does not make any assumption on the uniqueness of assigned values to agents. For example, if there are several agents that have $\mathbf{R}_{v,1}(t) = \mathbf{R}_{v,2}(t)$ and ϕ is maximum, then this means that there are at least 3 nodes that are equal to the median; our protocol will still stably compute the correct median value. It is only required that nodes have registers that can store any element in the multi-set $\{x_v : x \in \mathcal{V}\}$.

Before moving on to the running time analysis of our protocol under the probabilistic scheduler, it is worth noting that we can slightly modify our transition function f_{MDN} so that the protocol works on arbitrary strongly connected directed interaction graphs. Indeed, we just need to guarantee that every two agents will eventually be able to compare their input. This can be achieved if agents that interact also exchange the values of their registers. Therefore, we have the following more general result:

▶ **Theorem 4.** *Let $f'_{MDN} : \mathcal{S}^3 \times \mathcal{S}^3 \to \mathcal{S}^3 \times \mathcal{S}^3$ be a transition function defined as follows: for any $\vec{x}, \vec{y}, \vec{x}', \vec{y}' \in \mathcal{S}^3$, $f'_{MDN}(\vec{x}, \vec{y}) = (\vec{y}', \vec{x}')$ if and only if $f_{MDN}(\vec{x}, \vec{y}) = (\vec{x}', \vec{y}')$. Then the APP with input initialization function ι_{MDN}, output function γ_{MDN} and transition function f'_{MDN} stably computes the median function under any strongly connected directed interaction graph.*

We note that the potential function ϕ defined for the proof of correctness of our protocol, together with inequality (1) can be used to give upper bounds on the expected time needed for the protocol to stably compute the median under the probabilistic scheduler. However, this bound will depend on the input of the population. Nevertheless, by using Claim 3 and the definition of non-crossing configurations from the proof of Theorem 2, we can also provide an upper bound that only depends on the population size.

▶ **Theorem 5.** *Assuming the probabilistic scheduler under any connected undirected interaction graph, the expected time needed for the APP with input initialization function ι_{MDN}, output function γ_{MDN} and transition function f'_{MDN} to stably compute the median of the input of a population $\mathcal{V}$ of n agents is $O(n^6)$.*

Proof. Let T_1 be the time until the population reaches a configuration that is non-crossing up to $\lceil \frac{n}{2} \rceil$ (see the definition in the proof of Theorem 2) and let T_2 be the additional time needed for the value of the median to be propagated to all agents in the population. Clearly, the expected time needed for the protocol to stably compute the correct median value is $\mathbb{E}[T_1 + T_2]$.

Notice that, by definition of f'_{MDN}, whenever two agents interact, they also exchange the values of their registers. Therefore, it is as if each agent is performing a random walk on the interaction graph $G = (V, E)$. Furthermore, these random walks performed by two agents are independent until those 2 agents interact with each other.

Remember that, (arguing as in the proof of Theorem 2) if at some time t the configuration reached is non-crossing up to i, but not up to $i + 1$, we have that at least one of the following holds:

(i) For all agents $z \in L^+$, there is no agent $u \in U^-$, such that $I_z(t) = [x_z, x_u]$.
(ii) For all agents $u \in U^-$, there is no agent $z \in L^+$, such that $I_u(t) = [x_z, x_u]$.

In particular, by definition of f_{MDN}, agents $z \in L^+ \cup U^-$ can never decrease the length of the interval $I_z(t')$, for any $t' \geq t$. But then, the proof of Claim 3 implies that there is a pair of agents z, z', with $z \in L^+ \cup U^-$, who can interact according to Case II, which will strictly increase the length of $I_z(t')$. By Corollary 1 of [12], the expectation of the maximum meeting time is $O(n^3)$. If we also take into account the interactions which do not involve agents z, z', we then have that the expected number of steps before some agent $z \in L^+ \cup U^-$ increases the length of its interval is at most $O(n^4)$, where we also used the fact that the expected number of steps between two interactions that involve z, z' is $O(n)$. Since the length of the interval of any $z \in L^+ \cup U^-$ can increase at most n times and there are at most n agents, we have that $\mathbb{E}[T_1] = O(n^6)$.

To bound $\mathbb{E}[T_2]$, we can use Theorem 3 of [12], which states that the expected number of steps needed for a single random walk to cover all vertices of a graph is $O(n^4)$. If we also take into account the interactions which do not involve the median (there are $O(n)$ such steps in expectation between any interaction involving the median), we have that $\mathbb{E}[T_2] = O(n^5)$, which completes the proof. ◀

3 A faster protocol for median using random walks

In this section we describe another APP for stably computing the median of the input of a population of agents $\mathcal{V}$. Our protocol is not symmetric (hence in every interaction we distinguish between initiator and responder) and it converges with high probability (rather than with probability 1) to the correct median faster than the one in Section 2, at the expense of additional assumptions on the scheduler and the structure of the underlying interaction graph; in particular, we assume the probabilistic scheduler, under the complete interaction graph. For the sake of clarity of presentation, we will assume that the size $n = |\mathcal{V}|$ of the population is known to all agents. However, it is worth noting that this assumption can be dropped if one combines the APP presented here with the APP described in the Remark at the end of Section 4 for stably computing the population size, provided that agents are able to locally compute the value of a certain function $p(n)$ (for any n) that will be defined later.

Our APP uses 2 registers and $3p(n) + 2$ states per agent; the value of $p(n)$ will be determined in Theorem 6 below. For every agent $v \in \mathcal{V}$, the value of the input register $\mathbf{R}_{v,1}(t)$ will be initialized to x_v and will remain unchanged throughout the computation. The second register $\mathbf{R}_{v,2}(t)$ will eventually contain the median, i.e., with high probability, $\mathbf{R}_{v,2}(t)$ will eventually converge to the median of the set $\{x_v : v \in \mathcal{V}\}$. We view the state space $\mathcal{Q}$ for each agent $v \in \mathcal{V}$ as two counters $\mathbf{q}_{v,1}(t)$ and $\mathbf{q}_{v,2}(t)$; the first one can store an integer between $-p(n)$ and $p(n)$, and the second an integer between 0 and $p(n)$. These counters are used as follows: whenever agent v (the initiator) interacts with another agent u (the responder) at time $t + 1$, it will increase (respectively decrease) the value of $\mathbf{q}_{v,1}(t)$ by 1 if $x_v \geq x_u$ (respectively if $x_v < x_u$). Therefore, for each agent v, $\mathbf{q}_{v,1}(t)$ describes a (possibly) biased random walk, the least biased (or ideally unbiased) of which will correspond to the median. This means that, with high probability, if we stop those random walks after a sufficient number of steps, the one closest to 0 will correspond to the median of the population. The total number of steps that the random walk for agent v takes are counted in $\mathbf{q}_{v,2}(t)$.

More precisely, let $p(n)$ be any large enough integer function of the size of the population, which will stand for the maximum number of steps that we allow each random walk to take. We denote $\mathbf{C}_v(t) \overset{def}{=} \big(\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t), \mathbf{q}_{v,1}(t), \mathbf{q}_{v,2}(t)\big)$. Also, for any agent v, we denote by $\mathbf{1}_v$ the indicator variable that is equal to 1 if $\mathbf{q}_{v,2}(t) < p(n)$ and 0 otherwise. The initialization function of our protocol is given by $\iota_{\text{RW-MDN}}(x_v) = \mathbf{C}_v(0) = [x_v, x_v, 0, 0]$, for any $x_v \in \mathcal{S}$ and $v \in \mathcal{V}$, and the output function is given by $\gamma_{\text{RW-MDN}}(\mathbf{C}_v(t)) = \mathbf{R}_{v,2}(t)$, for any $v \in \mathcal{V}$. The transition function is defined as follows: if agent v (the initiator) interacts with agent u (the responder) at time $t + 1 = 1, 2, \ldots$, then $(\mathbf{C}_v(t+1), \mathbf{C}_u(t+1)) = f_{\text{RW-MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t))$, where $f_{\text{RW-MDN}}$ is given below.

```
Transition function $f_{\text{RW-MDN}}$
Input: $\mathbf{C}_v(t), \mathbf{C}_u(t)$
```

Case I: $x_v \geq x_u$ AND $\mathbf{q}_{v,2}(t) < p(n)$.

$$f_{\text{RW-MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) =$$
$$([x_v, \mathbf{R}_{v,2}(t), \mathbf{q}_{v,1}(t) + \mathbf{1}_v, \mathbf{q}_{v,2}(t) + \mathbf{1}_v], [x_u, \mathbf{R}_{u,2}(t), \mathbf{q}_{u,1}(t) - \mathbf{1}_u, \mathbf{q}_{u,2}(t) + \mathbf{1}_u])$$

Case II: $x_v < x_u$ AND $\mathbf{q}_{v,2}(t) < p(n)$.

$$f_{\text{RW-MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) =$$
$$([x_v, \mathbf{R}_{v,2}(t), \mathbf{q}_{v,1}(t) - \mathbf{1}_v, \mathbf{q}_{v,2}(t) + \mathbf{1}_v], [x_u, \mathbf{R}_{u,2}(t), \mathbf{q}_{u,1}(t) + \mathbf{1}_u, \mathbf{q}_{u,2}(t) + \mathbf{1}_u])$$

Case III: $\mathbf{q}_{v,2}(t) = \mathbf{q}_{u,2}(t) = p(n)$ AND $|\mathbf{q}_{v,1}(t)| \leq |\mathbf{q}_{u,1}(t)|$.

$$f_{\text{RW-MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = (\mathbf{C}_v(t), [x_u, \mathbf{R}_{v,2}(t), \mathbf{q}_{u,1}(t), \mathbf{q}_{u,2}(t)])$$

Case IV: Every other case.

$$f_{\text{RW-MDN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = (\mathbf{C}_v(t), \mathbf{C}_u(t))$$

We can prove the following:

▶ **Theorem 6.** *If $p(n) \geq 16n^2 \ln n$, then the APP with input initialization function $\iota_{\text{RW-MDN}}$, output function $\gamma_{\text{RW-MDN}}(v)$ and transition function $f_{\text{RW-MDN}}$ stably computes the median of the input of a population of n agents with high probability (i.e., with probability tending to 0 as n goes to ∞), under the complete interaction graph, assuming the probabilistic scheduler.*

Proof. Consider an ordering $v_1, \ldots, v_n$ of the agents of the population $\mathcal{V}$ in non-decreasing order of their input, i.e., $x_{v_1} \leq x_{v_2} \leq \cdots \leq x_{v_n}$. For any agent v, we will denote by m_v the number of agents that have the same input as v (including v herself), i.e., $m_v \overset{def}{=} |\{u \in \mathcal{V} : x_u = x_v\}|$.

For any agent v, and $\ell = 1, 2, \ldots, p(n)$, define the random variable $X_\ell^{(v)}$ as follows: $X_\ell^{(v)} = 1$ if at the ℓ-th interaction of agent v (either as an initiator or responder) the value of $\mathbf{q}_{v,1}(t)$ increases and $X_\ell^{(v)} = -1$ otherwise. In particular, for any $\ell = 1, 2, \ldots, p(n)$, at the ℓ-th interaction of agent v, the following disjoint events may happen: (a) either v will interact with an agent u that has $x_v > x_u$, in which case $X_\ell^{(v)} = 1$, or (b) v will interact with an agent u that has $x_v < x_u$, in which case $X_\ell^{(v)} = -1$, or (c) v will interact with another agent u that has $x_v = x_u$, in which case if v is the initiator (which happens with probability $\frac{1}{2}$, by definition of the probabilistic scheduler) then $X_\ell^{(v)} = 1$, otherwise (i.e., if v is the responder of the interaction) $X_\ell^{(v)} = -1$.

Therefore, for any fixed agent $v \in \mathcal{V}$, the $p(n)$ random variables $X_\ell^{(v)}$, $1 \leq \ell \leq p(n)$ are independent and also

$$\Pr\left(X_\ell^{(v)} = 1\right) = 1 - \Pr\left(X_\ell^{(v)} = -1\right) = \frac{|\{u : x_v > x_u\}|}{n-1} + \frac{1}{2}\frac{(m_v - 1)}{n-1}.$$

For each agent $v_i, i \in [n]$, define now the discrete time stochastic process $\{Y_\ell^{(v_i)}\}_{\ell \geq 0}$ as follows:

(i) $Y_0^{(v_i)} = 0$

(ii) $Y_\ell^{(v_i)} = Y_{\ell-1}^{(v_i)} + X_\ell^{(v_i)} - \mathbb{E}[X_\ell^{(v_i)}] = \sum_{i=1}^{\ell} X_\ell^{(v_i)} - \ell\frac{n-2|\{u:x_{v_i}>x_u\}|-m_{v_i}}{n-1}$, for any $1 \leq \ell \leq p(n)$

(iii) $Y_\ell^{(v_i)} = Y_{\ell-1}^{(v_i)}$, for any $\ell > p(n)$

Since $X_\ell^{(v_i)}$ are independent, for every $1 \leq \ell \leq p(n)$, it is easy to prove that $\{Y_\ell^{(v_i)}\}_{\ell \geq 0}$ is a Martingale, that also satisfies $|Y_\ell^{(v_i)} - Y_{\ell-1}^{(v_i)}| \leq 2$, for all $\ell \geq 1$. Therefore, by Azuma's inequality, for any $x \geq 0$,

$$\Pr\left(\left|Y^{(v_i)}_{p(n)} - Y^{(v_i)}_0\right| \geq x\right) = \Pr\left(\left|Y^{(v_i)}_{p(n)}\right| \geq x\right) \leq 2e^{\frac{-x^2}{8p(n)}}.$$

Let now $t^*_{v_i}$ the time just after the $p(n)$-th interaction of v_i, i.e., the values of the counters $\mathbf{q}_{v_i,1}, \mathbf{q}_{v_i,2}$ remain unchanged after $t^*_{v_i}$. By the definition of $X^{(v_i)}_\ell, 1 \leq \ell \leq p(n)$, we then have that $\mathbf{q}_{v_i,1}(t^*_{v_i}) = \sum_{\ell=1}^{p(n)} X^{(v_i)}_\ell$. Furthermore, by the previous inequality and by definition of $\{Y^{(v_i)}_\ell\}_{\ell \geq 0}$, we have that

$$\Pr\left(\left|\mathbf{q}_{v_i,1}(t^*_{v_i}) - p(n)\frac{n - 2|\{u : x_{v_i} > x_u\}| - m_{v_i}}{n-1}\right| \geq x\right) = \Pr\left(\left|Y^{(v_i)}_{p(n)}\right| \geq x\right) \leq 2e^{\frac{-x^2}{8p(n)}}. \quad (2)$$

For any agent $v \in \mathcal{V}$, let $A_v = p(n)\frac{n-2|\{u:x_v>x_u\}|-m_v}{n-1}$. Notice now that, for any two agents v, v', such that $x_v \neq x'_v$, we have that $|(|\{u : x_v > x_u\}| - |\{u : x_{v'} > x_u\}|)| \geq 1$, since for one of v, v', the number of agents with strictly greater input must be larger. Similarly, $|(|\{u : x_v > x_u\}| + m_v) - (|\{u : x_{v'} > x_u\}| + m_{v'})| = |(|\{u : x_v \geq x_u\}| - |\{u : x_{v'} \geq x_u\}|)| \geq 1$. Therefore, we have proved the following separation inequality:

$$|A_v - A_{v'}| \geq 2\frac{p(n)}{n-1}, \quad \text{for every } v, v' \text{ with } x_v \neq x'_v.$$

Now set $x = \frac{p(n)}{n}$ in (2). Since $p(n) \geq 16n^2 \ln n$, we have that, for any $v_i, i \in [n]$,

$$\Pr\left(\left|\mathbf{q}_{v_i,1}(t^*_{v_i}) - A_{v_i}\right| \geq \frac{p(n)}{n}\right) \leq \frac{2}{n^2}$$

By the union bound, we conclude that, with probability at least $1 - \frac{2}{n}$, not only are the values $\mathbf{q}_{v_i,1}(t^*_{v_i})$, for all nodes v_i, concentrated around their mean values A_{v_i}, but they are also well separated, i.e., just by looking at $\mathbf{q}_{v_i,1}(t^*_{v_i})$, we can uniquely determine the value of A_{v_i}. Furthermore, node v_i will never change the value of its first register after time $t^*_{v_i}$.

Finally note that the smallest value of $|A_{v_i}|$ (hence also the smallest value of $|\mathbf{q}_{v_i,1}(t^*_{v_i})|$ whp, because of well separation) will correspond to agent $v_{\lceil n/2 \rceil}$, i.e., the agent with the median of $\{x_v, v \in \mathcal{V}\}$ as input. Indeed, it is not hard to see that, by definition, $|A_{v_i}|$ is a function of i, for which $|A_{v_i}| \leq |A_{v_{i-1}}|$ for every $i \leq \lceil \frac{n}{2} \rceil$, and $|A_{v_{i+1}}| \geq |A_{v_i}|$ for every $i \geq \lceil \frac{n}{2} \rceil$. Therefore, with high probability, the correct median will be propagated because of Case III of the protocol. This completes the proof. ◀

4 k-th minimum element

In this section, we present an APP with n states and 2 registers per agent (the first of which serves as the input register and does not change throughout the computation) that stably computes the k-th minimum element of the input values $x_v, v \in \mathcal{V}$. Our protocol is not symmetric, hence in every interaction we distinguish between initiator and responder. For simplicity, we first present a protocol that stably computes the k-minimum function assuming the complete interaction graph and then we generalize it for arbitrary interaction graphs.

The set of states of our protocol is $\mathcal{Q} = \{0, 1, \ldots, n-1\}$; without loss of generality, we treat the state of each agent v at time t as a counter $\mathbf{q}_v(t)$ which is initialized to the value 0 and can count up to $n - 1$. For simplicity, we will denote $\mathbf{C}_v(t) \overset{def}{=} (\mathbf{R}_{v,1}(t), \mathbf{R}_{v,2}(t), \mathbf{q}_v(t))$.

The input initialization function of our protocol is given by $\iota_{k\text{-MIN}}(x_v) = \mathbf{C}_v(0) = (x_v, x_v, 0)$, for any $x_v \in \mathcal{S}$ and $v \in \mathcal{V}$, and the output function is given by $\gamma_{k\text{-MIN}}(\mathbf{C}_v(t)) =$

$\mathbf{R}_{v,2}(t)$, for any $v \in \mathcal{V}$. The transition function is defined as follows: if agent v (the initiator) interacts with agent u (the responder) at time $t+1 = 1, 2, \ldots$, then $(\mathbf{C}_v(t+1), \mathbf{C}_u(t+1)) = f_{k\text{-MIN}}(\mathbf{C}_v(t), \mathbf{C}_u(t))$, where $f_{k\text{-MIN}}$ is given below.

```
Transition function $f_{k\text{-MIN}}$
Input: $\mathbf{C}_v(t), \mathbf{C}_u(t)$
Case I: $\mathbf{q}_v(t) = \mathbf{q}_u(t)$ AND $\mathbf{R}_{v,1}(t) \geq \mathbf{R}_{u,1}(t)$.
    $f_{k\text{-MIN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = ([x_v, \mathbf{R}_{v,2}(t), \mathbf{q}_v(t)+1], [\mathbf{C}_u(t)])$
Case II: $\mathbf{q}_v(t) = \mathbf{q}_u(t)$ AND $\mathbf{R}_{v,1}(t) < \mathbf{R}_{u,1}(t)$.
    $f_{k\text{-MIN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = ([\mathbf{C}_v(t)], [x_u, \mathbf{R}_{u,2}(t), \mathbf{q}_u(t)+1])$
Case III: $\mathbf{q}_v(t) \neq \mathbf{q}_u(t)$ AND $\mathbf{q}_v(t) = k-1$.
    $f_{k\text{-MIN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = ([x_v, x_v, \mathbf{q}_v(t)], [x_u, x_v, \mathbf{q}_u(t)])$
Case IV: Every other case.
    $f_{k\text{-MIN}}(\mathbf{C}_v(t), \mathbf{C}_u(t)) = (\mathbf{C}_v(t), \mathbf{C}_u(t))$
```

▶ **Theorem 7.** *The APP with initialization function $\iota_{k\text{-}MIN}$, output function $\gamma_{k\text{-}MIN}$ and transition function $f_{k\text{-}MIN}$ stably computes the k-minimum of the set $\{x_v : v \in \mathcal{V}\}$.*

Proof. We first prove that, eventually, there will be exactly one agent in state i, for each $0 \leq i \leq n-1$. To this end, define t_i to be the earliest time when exactly one agent has state j, for each $j \leq i$. Note that, after time t_i, every agent v_j, $1 \leq j \leq i+1$, which has $\mathbf{q}_{v_j}(t_i) = j-1$, will never change her state. Indeed, by definition of $f_{k\text{-MIN}}$, for every $v \in \mathcal{V}$, $\mathbf{q}_v(t)$ never decreases; it can only increase if there is another agent $u \neq v$ with $\mathbf{q}_u(t) = \mathbf{q}_v(t)$. Thus we only need to prove that t_{n-1} is finite; we do this by induction on i.

For the base case of our inductive argument we need to prove that t_0 is finite. This is true because, if at some time t there are at least 2 agents at state 0, then, since the scheduler is fair, after finite time, two of these agents will eventually interact, which, by Cases I and II of $f_{k\text{-MIN}}$, will result in one of the agents increasing her state by 1. This will continue until there is only one agent in state 0.

For the inductive step, suppose that t_i is finite, for some i. By definition, at time t_i we have exactly one agent in state j, for each $j = 0, \ldots, i$. However, by definition of $f_{k\text{-MIN}}$, we also have that at time t_i some agent v set $\mathbf{q}_v(t_i) = i+1$ (i.e., either the one with the largest value stored in its first register, or the initiator in the interaction if both agents had the same value stored in their first registers). If this agent v was the only agent in state $i+1$ after t_i, then we also have that $t_i = t_{i+1}$. If not, then, similarly to the base case, after finite time the number of agents in state $i+1$ will decrease by 1, and this will continue until (within finite time) only one agent remains in state $i+1$. This completes the induction step, implying that t_i is finite, for every $i = 0, 1, \ldots, n-1$. In particular, in finite time t_{n-1}, there will be exactly one agent in state i, for each $i = 0, \ldots, n-1$ and these values will never change thereafter. Moreover, after time t_{n-1}, the agent in state $k-1$ will have the k-minimum of the set $\{x_v : v \in \mathcal{V}\}$ stored in its first register and this item will eventually appear in the second register of each agent in the population (because of Case III of $f_{k\text{-MIN}}$). ◀

We now slightly modify our transition function $f_{k\text{-MIN}}$ so that the protocol works on arbitrary strongly connected directed interaction graphs. Notice that we just need to guarantee that every two agents will eventually be able to exchange their local information.

This can be achieved if agents that interact also swap their register values and states. Therefore, we have the following more general result:

▶ **Theorem 8.** *Let $f'_{k\text{-}MIN} : (\mathcal{S}^2 \times \mathcal{Q}) \times (\mathcal{S}^2 \times \mathcal{Q}) \to (\mathcal{S}^2 \times \mathcal{Q}) \times (\mathcal{S}^2 \times \mathcal{Q})$ be a transition function defined as follows: for any $\vec{x}, \vec{y}, \vec{x}', \vec{y}' \in \mathcal{S}^2 \times \mathcal{Q}$, $f'_{k\text{-}MIN}(\vec{x}, \vec{y}) = (\vec{y}', \vec{x}')$ if and only if $f_{k\text{-}MIN}(\vec{x}, \vec{y}) = (\vec{x}', \vec{y}')$. Then the APP with initialization function $\iota_{k\text{-}MIN}$, output function $\gamma_{k\text{-}MIN}$ and transition function $f'_{k\text{-}MIN}$ stably computes the k-minimum of the set $\{x_v : v \in \mathcal{V}\}$ under any strongly connected interaction graph.*

Note that, upon convergence in this APP, each of the n agents has a different state which is an integer between 0 and $n - 1$. Therefore all these states provide a *total ordering* of the agents with respect to their input values.

▶ Remark. This APP can be slightly modified to stably compute the population size $n = |\mathcal{V}|$, under any strongly connected interaction graph. Indeed, we only need that the second register of each agent $v \in \mathcal{V}$ is replaced by another counter $\mathbf{q}'_v(t)$, which stores the value n (instead of the k-minimum element). This can be achieved trivially by initializing $\mathbf{q}'_v(0)$ to 1, for each agent v and modifying $f'_{k\text{-MIN}}$ as follows: whenever agents v, u interact at time $t+1$, we set $\mathbf{q}'_v(t+1) = \mathbf{q}'_u(t+1) = \max\{\mathbf{q}_v(t)+1, \mathbf{q}_u(t)+1, \mathbf{q}'_v(t), \mathbf{q}'_u(t)\}$; the rest of $f'_{k\text{-MIN}}$, i.e., the part that affects the first registers and the states, remains unchanged.

References

1 Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 47–56, 2015.

2 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18:235–253, 2006.

3 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 292–299, 2006.

4 Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

5 James Aspnes and Eric Ruppert. An introduction to population protocols. In Benoît Garbinato, Hugo Miranda, and Luís Rodrigues, editors, *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer-Verlag, 2009.

6 Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. Programmability of chemical reaction networks. In Anne Condon, David Harel, Joost N. Kok, Arto Salomaa, and Erik Winfree, editors, *Algorithmic Bioprocesses*, Natural Computing Series, pages 543–584. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-540-88869-7_27.

7 Fabian Kuhn, Thomas Locher, and Stefan Schmid. Distributed computation of the mode. In *Proceedings of the 27th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 15–24, 2008.

8 Thomas G. Kurtz. *Approximation of Population Processes.* Society for Industrial and Applied Mathematics, 1987.

9 George B. Mertzios, Sotiris E. Nikoletseas, Christoforos L. Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 1, pages 871–882, 2014.

10 Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. *New Models for Population Protocols*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2011.

11 Yves Mocquard, Emmanuelle Anceaume, James Aspnes, Yann Busnel, and Bruno Sericola. Counting with population protocols. In *Proceedings of the 14th International Symposium on Network Computing and Applications (NCA)*, pages 35–42, 2015.

12 Prasad Tetali and Peter Winkler. On a random walk problem arising in self-stabilizing token management. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 273–280, 1991.