Space-Optimal Naming in Population Protocols

Janna Burman, Joffroy Beauquier, Devan Sohier

▶ **To cite this version:**

Janna Burman, Joffroy Beauquier, Devan Sohier. Space-Optimal Naming in Population Protocols. [Research Report] LRI, Université Paris Sud, CNRS, Université Paris-Saclay, France. 2018. hal-01790517v2

HAL Id: hal-01790517

https://inria.hal.science/hal-01790517v2

Submitted on 19 May 2018 (v2), last revised 4 Aug 2019 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Space-Optimal Naming in Population Protocols

Janna Burman[1]
LRI, Université Paris Sud, CNRS, Université Paris-Saclay, France
janna.burman@lri.fr

Joffroy Beauquier[2]
LRI, Université Paris Sud, CNRS, Université Paris-Saclay, France
joffroy.beauquier@lri.fr

Devan Sohier
LI-PaRAD, Université de Versailles, Université Paris-Saclay, France
devan.sohier@prism.uvsq.fr

Abstract

The distributed *naming problem*, assigning different names to the nodes in a distributed system, is a fundamental task. This problem is non trivial, specially when the amount of memory available for the task is low, and when requirements for fault-tolerance are added.

The considered distributed computation model is *population protocols*. In this model, a priori anonymous and indistinguishable mobile nodes (called agents), communicate in pairs and in an asynchronous manner (according to a fairness condition). Fault-tolerance is addressed through *self-stabilization*, in terms of arbitrary initialization of agents.

This work comprises a comprehensive study on the necessary and sufficient state space conditions for naming. It is studied under various combinations of model assumptions: *weak or global fairness*, arbitrary or uniform initialization of agents, existence or absence of a distinguishable agent (arbitrarily initialized or not), possibility of breaking symmetry in pair-wise interactions (*symmetric or asymmetric transitions*). For each possible combination of these assumptions, either an impossibility is proven or the necessary *exact* number of states (per mobile agent) is determined and an appropriate space-optimal naming protocol is presented.

2012 ACM Subject Classification C.2.4 Distributed Systems, I.1.2 Algorithms

Keywords and phrases networks of passively mobile agents, population protocols, naming, self-stabilization, exact space complexity, tight lower bounds, global and weak fairness

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

1.1 Historical Background and Motivation

In the *naming problem*, some distributed entities are initially anonymous and have to be assigned unique names (identities). This problem has received a lot of attention in distributed computing, since it is particularly relevant in this domain. Most of the distributed algorithms assume unique identifiers, and some tasks are impossible without this assumption (e.g., certain types of leader election and fault-tolerance problems [43, 53]). Thus, providing the entities of a system with distinct names is a basic and important task. In the context of

[1] [The work of this author was partially supported by the Israeli-French Maimonide and the INS2I PEPS JCJC research projects.]
[2] [The work of this author was partially supported by the Israeli-French Maimonide research project.]


© Janna Burman, Joffroy Beauquier and Devan Sohier;
licensed under Creative Commons License CC-BY


the population protocol model naming received much less attention. Below, we discuss the reasons for that and motivate our study.

The population protocol model was introduced by Angluin et al. [3] as a minimalist model of finite-state agents interacting in pairs in an asynchronous way under the *global fairness* condition (which ensures that an infinitely often reachable configuration is unavoidable). The memory of agents is supposed to be constant, i.e., independent of the population size N. Due to this, agents are prohibited of storing unique identifiers. This moves aside the interest in the naming problem. This model was mainly motivated by mobile sensor networks of tiny cheap sensing and computing devices, moving unpredictably and uncontrollably. However, it has been proved useful in modeling population dynamics from other various areas like trust and rumors propagation in social networks [26], chemical reactions [35] and even game theory dynamics [16][3].

The interest in the model goes beyond its memory restriction, not only because of the related computational limitations, but also because many practical motivating scenarios do not necessary impose such restriction. For example, many communicating devices are manufactured with (almost) unique identifiers. Together with that, there is a particular interest in a priori anonymous entities in studies of sociodynamic and economic phenomena (using social network models [13] or game theory [16]), and in which constant memory restriction is not obviously a necessity.

It was proven that the original population protocols compute exactly the semi-linear predicates [5], which excludes for example multiplications of agents' inputs. Additional negative results due to the severe memory restriction followed. Consider for example a widely studied model version where each interacting pair is selected uniformly at random [8]. This ensures global fairness with probability 1 [39]. For this case, it was shown that a fundamental task of distributed computing - leader election - cannot be solved in sub-linear (parallel) time when agents are equipped only with a fixed (constant) number of states [29]. Due to this result, a lot of work has been devoted to the time-space trade-offs of leader election (and the related problem of majority consensus), all obligated to assume a non-constant state space contrary to the original population protocols (for overcoming the linear time bound). The most efficient recent results solve this problems in $O(\log^2 N)$ parallel time using optimally $O(\log \log N)$ states in case of leader election [34], and using $O(\log N)$ states in case of majority [2]. In the appendix, we mention additional related trade-off studies.

It is worth noting that the mentioned protocols are not *self-stabilizing* [27], i.e., they do not function starting from an arbitrary configuration, and thus cannot tolerate transient memory faults. Whereas, in the context of limited-resources agents in the population protocols (especially those inspired by biology, or by cheap mass-produced sensor devices), one should expect that agents may naturally be non-initialized and sometimes faulty. The first study of self-stabilization in population protocols appeared in [6]. Since then, many negative results have been established concerning constant space self-stabilizing population protocols. For example, a pioneering work of [19] proves that at least N states, and the knowledge of N, are required for realizing self-stabilizing leader election (under global fairness and in complete graphs). To overcome this impossibility a relaxed version of self-stabilization (concerning the closure requirement) was proposed [51]. It allowed to solve the problem using the knowledge of an upper bound on N (and sometimes with additional assumptions), but

[3] The dynamics of a population of agents playing a game repeatedly against each-other and adapting their strategy according to the PAVLOV behavior can be translated in terms of pairwise interacting agents in population protocols.

still by exploiting a non-constant space (e.g., [50, 52, 37]). In a later work [23] an equivalence between self-stabilizing leader election and self-stabilizing deterministic oscillatory behavior was shown (i.e., given a leader, it is possible to implement an oscillator and vice versa). This implied the use of the necessary N states to implement the oscillator (due to the result of [19]). A recent work [55] shows that self-stabilizing *bipartition* is impossible under *weak fairness* using a constant number of states. Weak fairness (also considered in this work) is a natural, and classical in distributed computing, fairness that requires every pair of agents to interact infinitely often (refer to the model section for formal definitions of weak and global fairness, together with an example illustrating their difference). Additional earlier works show, for a variety of problems, like deterministic logical clock synchronization [9], mutual exclusion [10] and counting [12], that under weak fairness they are impossible when all agents are constant-state and for counting, even if there is a memory powerful leader. All these examples demonstrate an important limitation, related to the constant memory constraint, to the design of self-stabilizing population protocols.

Justified by above-mentioned and additional existing negative results, and by a growing interest in the model going beyond the scenarios of memory limited agents, many extensions were proposed to overcome the impossibilities, augment its computational power (e.g., [21, 18]), improve time complexity (e.g., [48, 2]) or tolerate faults (e.g., [25, 36]). We will not cite all of them here, but we will give examples of such studies relaxing state space requirements. For example, the authors of [36] made a natural assumption of identified agents which are also capable of storing a constant number of other identifiers. In this model, called *community protocols*, Byzantine faults can be tolerated, while this is impossible without having identifiers. Moreover, the model provides the power of a Turing machine with an $O(N \log N)$ space. In a later study [18], authors provide a hierarchy establishing complexity classes for the cases going from non-identified agents (population protocol model) to uniquely identified ones (community protocol model), passing by the case of homonyms. In [21], the authors study the complexity classes for a general case where agents, instead of being automata, are Turing Machines, using $O(f(N))$ memory each. Finally, a wide study have been conducted on *mediated population protocols* in which agents capable of storing additional states per each neighbor they can interact with [44, 45].

To the best of our knowledge, there are no previous studies dedicated to the task of naming in population protocols. This may seam natural considering the original constant-state population protocols. However, the current study is inspired by motivating practical scenarios and above-mentioned studies relaxing this restriction. An additional more specific motivation to this work is that of self-stabilization (which we believe, as explained above, to be important in the context of population protocols). In this context, one may make several observations. First, many self-stabilizing tasks have to use (at least) a linear (on N) space to be realized (see above). Second, in some cases including self-stabilization, the task of naming is frequently performed as a by-product or as an important design module. For example, this can be observed in designs of self-stabilizing counting [12, 38, 11], leader election [19] and deterministic oscillators [23]. In [21], an initialized naming protocol is given and used to characterize the exact complexity classes for the case where each agent has $O(\log N)$ bits of memory ($O(N)$ states). Apart of that, some fault-tolerant and self-stabilizing population protocols assume that names are given a priori, e.g., [36, 50].

These observations suggest that the task of naming in population protocols deserves to be studied thoroughly. This paper gives a synthesized presentation of a deep comprehensive study on this problem. It focuses on the necessary and sufficient state space conditions for naming under all possible permutations of the considered and classical model assumptions.

These assumptions (like existence of a leader, weak or global fairness, initialization of agents or not, symmetry of transition rules) and their interest are all discussed below. Tight positive and negative results (tight lower bounds) on space per agent required for naming for each assumption permutation are proved.

We perform an *exact* state space analysis, in contrast to, e.g., the asymptotic one. Both are certainly interesting from a theoretical point of view, and the widely used asymptotic analysis is known to be practically useful. At the same time, the exact state space analysis also received an important attention in the algorithmic domain in general and in population protocols in particular, e.g., [55, 23, 11, 38, 12]. On the practical side, the exact analysis is motivated, e.g., by extremely cheap resource-limited devices that should already spend memory for storing the names and the controlling firmware and may have severe restrictions on the additionally available memory. Another practical motivation is related to self-stabilization. The less volatile memory is used by a protocol, the less it is vulnerable to corruptions (dealt by self-stabilization). An important contribution of this work is that it shows, by performing a careful and subtle exact analysis and protocol design, that even in cases where an optimal-space naming (using only N states) is impossible, it is realizable using *only one* additional state (less than one bit), and thus very improbable to be corrupted. We believe that such results are interesting both from theoretical and practical point of views.

An additional particularity of this work is the focus on deterministic protocol design. The motivation is that randomization is not always possible, since random generators are not always available and a physical phenomenon does not always guarantee randomness. Indeed, in case of cheap mass produced devices it is unlikely to assume that they are given an embedded random generator. Even though we do not use randomization in protocols, in some cases, we assume global fairness which mimics a randomized environment. Together with that, we also study weak fairness which simulates a completely non-random adversary. See additional discussion below.

For defining the data structures used by the solutions, most of the related studies (e.g., [12, 38, 11]) assume the existence of a known upper bound P on the number of agents N. This assumption is natural, since the agents have eventually to store their name in a variable with a fixed number of bits, from which P can be deduced. When considering a size bound, the space complexity of a solution to the naming problem is expressed in function of the necessary number of states per agent with respect to the parameter P. Obviously, P is a lower bound for naming a population of at most P agents.

We study naming under various combinations of model parameters. The considered parameters affect the type and the level of difficulty of breaking symmetry or of achieving fault-tolerance. A first parameter of the model is the nature of the assumed fairness, weak or global. These notions are already discussed above and we refer the reader to the model section for the definitions and an illustrating example. Note that both types of fairness are classical and natural in population protocols. Weak fairness is widely used in the literature of distributed computing, while it is less studied with population protocols due to the difficulty it imposes in the design and due to certain practical justifications (see discussions in [39] and [7]).[4]

A second parameter is the symmetry nature of the (transition) rules of a protocols. With *symmetric rules*, two interacting agents in the same state, stay in identical states. With

[4] Global fairness can be viewed as a way for simulating randomized systems without introducing randomization explicitly. This explains why it is easier to get solutions under global fairness. The nature of weak fairness is closer to (but still different from) the one so called *local fairness*, proposed in [31].

asymmetric rules, they can take different states. Notice that the symmetric rules assumption is more general, since any asymmetric protocol is symmetric, but not vice versa. This and some previous studies demonstrate that symmetric rules add a (symmetry breaking) difficulty in the design of *exact* space efficient population protocols, certainly in conjunction with weak fairness [12, 38, 11], but also with global fairness [55].[5]

The motivation for considering symmetric transitions is not only theoretical. First of all, one may imagine social networks deployed over mobile sensor networks. There, the possibly known asymmetry in an interaction (expressed by the *initiator* of a communication and its *responder*) may be undesirable in the "application level" (where protocols are executed) to guarantee some sort of equality. Second, as mentioned before, population protocols can be used to analyze repetitive Pavlovian games and many of them are by essence symmetrical relatively to players, so are the transitions of the population protocols modeling them. Note however that symmetric Pavlovian population protocols (assuming symmetric rules) are less powerfull than the original ones [40, 17]. This supports the computational difficulty added by symmetric rules.

A third parameter is the presence or absence of a unique leader (a distinguishable agent), which is obviously also useful for the sake of breaking symmetry. In the context of sensor networks, this agent may represent a (possibly mobile) base station, having augmented resources comparing to the tiny mobile agents. From this perspective and similarly to previous studies (e.g., [55, 11, 38]), we are not concerned with the space complexity of this agent.

A fourth parameter is related to the initialization of system agents, i.e., of the leader (if present) and of the other agents (called here *mobile*). In case of mobile agents, if initialization is assumed, it is always *uniform*, i.e., to the same value for every mobile agent. Notice that, with weaker initialization assumption (only the leader is initialized, or nobody), the system is stronger against state corrupting transient faults, and is more adapted to repetitive execution of a task (requiring less or no re-initialization). In case of arbitrary initialization, the given solution is called self-stabilizing [27]. To emphasize again, as the considered cheap or simply unreliable agents may be prone to failures, it is expected that memory or communication errors happen.

1.2 Contribution

We thoroughly investigate the naming problem, under all the possible combinations of the parameters described above. All the negative results (impossibilities and lower bounds) are presented in Section 3, while all the positive (state-optimal solutions) are given in Section 4.

Table 1 gives a synthetic view of these results. For each case, it indicates first the proposition establishing the feasibility, either by proving impossibility or by construction of a space-optimal protocol. In the latter case, the table also indicates the optimal (used) number of states and the relevant statement of the lower-bound.

Notice that, first of all, the table concerns the case of *arbitrarily initialized mobile agents*. Together with that, it is also relevant to the case of *uniform initialization of mobile agents*. It is obvious for the positive results (the protocols stay correct). However, somewhat surprisingly, the impossibilities and lower bounds also hold under this stronger assumption, but with only

[5] Note that an asymmetric population protocol can be transformed into a symmetric one using the transformer of [17]. However, this transformer requires global fairness and doubles the number of states per agent. This makes it frequently inadequate for obtaining a space efficient symmetric solution from an asymmetric one (in terms of exact space complexity), and certainly under weak fairness.

one exception. The exception concerns symmetric rules under weak fairness and with an initialized leader, i.e., when a leader is present and can be initialized together with the other agents (refer to the table). Then, there is a simple naming protocol with only P states per mobile agent, instead of the necessary $P+1$ states in case of an arbitrary initialization of mobile agents (but with an initialized leader). Refer to Proposition 14. On the contrary, the analysis of tight negative and positive results for the same case, but without the initialization of mobile agents, is much more complicated (see Section 3.1 and Prop. 16).

Additional remarks can be made about the table. First, under weak fairness and without leader, no symmetric deterministic protocol is capable of breaking system symmetry, and so naming is impossible. Second, if asymmetric rules are allowed, in all cases, there is an optimal-space solution with P states. On the contrary, with symmetric rules, the norm is $P+1$ states, excluding two cases: when there is an initialized leader and either global fairness or uniform initialization of mobile agents is assumed. In both cases, the problem can be solved with P states per agent.

To sum up, the general contribution of this thorough work is in that, together with establishing precise space limits of naming in population protocols under different relevant model variations (some of them being particularly non trivial, like the one of Theorem 11), it also provides a complete set of space-optimal naming protocols under all the variations. Some of them are especially subtle with an intricate analysis, in particular, in cases of self-stabilization and non-initialized mobile agents (Prop. 13, Prop. 16 and Prop. 17). The fact that the demonstrated protocols use only one additional state, in addition to the states necessary to store names, is particularly promising in practice when transient memory corruption faults are of concern (see a discussion in the introduction).

	symmetric rules		**asymmetric rules**
	weak fairness	**global fairness**	**weak/global fairness**
no leader	impossible (Prop. 1)	Prop. 13, with $P+1$ states (Prop. 2)	Prop. 12, with P states
non-initialized leader	Prop. 16, with $P+1$ states (Prop. 4)	Prop. 13, with $P+1$ states (Prop. 4)	Prop. 12, with P states
initialized leader	*non-initialized agents:* Prop. 16, with $P+1$ states (Theorem 11); *initialized agents:* Prop. 14, with P states	Prop. 17, with P states	Prop. 12, with P states

Table 1 Synthesis of the relevant propositions and theorems establishing the feasibility of naming and the necessary (optimal) state space, under different model parameters. If not stated otherwise, the indicated results hold for both arbitrarily an uniformly initialized mobile agents.

Note: Some proofs and the additional related work discussion appear in the appendix. Though, the most relevant work on naming is mentioned above.

2 Model and Notations

We adopt the model of population protocols [7] as a basic model. A system consists of a collection $\mathcal{A}$ of pairwise interacting agents, also called *population*. Each agent can represent a sensing and communicating mobile device. Among the agents, there can be a unique

distinguishable agent called the *leader* (possibly representing a base station) which can be as powerful as needed, in contrast with the resource-limited non-leader agents. The non-leader agents are also called *mobile*, interchangeably. The size of the population is the number of mobile agents, denoted by N, and is unknown (a priori) to the agents. Each agent has a state taken from a finite set of states, the same for all mobile agents (denoted Q and depending of the bounded parameter P), but generally different for the leader.

A *(population) protocol* can be modeled as a finite transition system defined by transitions between *configurations*, where a configuration is a vector of states of all the agents. The transitions between configurations are defined by pairwise interactions between agents. When two agents x, in state p, and y, in state q, in a configuration C, interact (meet), they execute a *transition rule* $(p, q) \to (p', q')$. As a result, x changes its state from p to p' and y from q to q'. The new configuration C', resulting from this state changes, is said to be reachable from C (in one step), denoted by $C \to C'$. If $p = p'$ and $q = q'$, the corresponding transition is called *null* (such transitions are specified by default), and non-null otherwise.[6]
If there is a sequence of configurations $C = C_0, C_1, \ldots, C_k = C'$, such that $C_i \to C_{i+1}$ for all $i, 0 \leq i < k$, we say that C' *is reachable* from C, denoted $C \xrightarrow{*} C'$.

The transition rules are *deterministic*, if for every pair of states (p, q), there is exactly one (p', q') such that $(p, q) \to (p', q')$. We consider only deterministic transitions and thus, only *deterministic protocols*. Transitions and protocols can be *symmetric* or *asymmetric*. Symmetric means that, if $(p, q) \to (p', q')$ is a transition rule, then $(q, p) \to (q', p')$ is also a transition rule. In particular, if $(p, p) \to (p', q')$ is symmetric, $p' = q'$. A protocol is called symmetric, if all its transition rules are symmetric, and asymmetric otherwise.

Let $(p_1, q_1) \to (p_2, q_2)$, $(p_2, q_2) \to (p_3, q_3)$, ..., $(p_{k-2}, q_{k-2}) \to (p_{k-1}, q_{k-1})$, $(p_{k-1}, q_{k-1}) \to (p_k, q_k)$ be a sequence of rules of a protocol. Then, we shortly write $(p_1, q_1) \xrightarrow{*} (p_k, q_k)$ to denote the successive application of the rules of the sequence, to the same pair of agents, initially in states p_1 and q_1, leading them to p_k and q_k, respectively. We sometimes call an agent in state p a p-state agent, or just a p-agent.

An *execution* of a protocol is an infinite sequence of configurations and transitions $C_0, t_1, C_1, t_2, C_2, t_3, \ldots$ such that C_0 is the starting configuration and, for each $i \geq 0$, $C_i \to C_{i+1}$ such that t_i is the transition between two particular agents, used to reach C_i from C_{i+1}. When the actual transitions are irrelevant, one can omit them from the illustrated execution sequence $(C_0, C_1, C_2, \ldots)$. A *segment* or a *sub-execution* is a sub-sequence of an execution. The *trace of transitions* of a sub-execution $C_0, t_1, C_1, t_2, C_2, \ldots t_k, C_k$ is a sequence $t_1, t_2, t_3, \ldots t_k$ of transitions, and the corresponding *trace of transition rules* is the sequence $r_1, r_2, r_3, \ldots r_k$ such that r_i is the transition rule applied in t_i. In a real distributed execution, interactions of distinct agents are independent and could take place simultaneously (in parallel), but when writing down an execution we order those simultaneous interactions arbitrarily.

An execution is said *weakly fair*, if every pair of agents in $\mathcal{A}$ interacts infinitely often. An execution is said *globally fair*, if for every two configurations C and C' such that $C \to C'$, if C occurs infinitely often in the execution, then C' also occurs infinitely often in the execution. This also implies that, if in an execution there is an infinitely often reachable configuration, then it is infinitely often reached [5]. Global fairness can be viewed as simulating randomized systems without introducing randomization explicitly (a system where pairs of agents interact in a random fashion is globally fair with probability 1 [39]).

[6] For simplicity, in some cases, we do not present protocols under the form of transition rules, but under the equivalent form of a pseudo-code.

A simple example allows to better understand the difference between weak and global (or probabilistic) fairness. Consider a population of 3 agents. Each agent can be white or black, and initially one agent is black and the two others are white. Consider also the protocol in which, when two white agents interact, they both become black and when two agents of different colors interact, they exchange their colors. It is easy to see that there is an infinite weakly fair execution in which there is always one black and two white agents (the black color "jump" indefinitely from agent to agent). At the contrary, every globally fair execution terminates in a configuration in which the 3 agents are black, because otherwise there would be infinitely many configurations during an execution from which the "all black" configuration could be reached, without ever being reached (contradicting global fairness).

A *(static) problem* is defined by a predicate $\mathcal{D}$ on configurations. A population protocol $\mathcal{P}$ is said to *solve a problem* $\mathcal{D}$, if and only if every execution of $\mathcal{P}$ reaches a configuration satisfying the conditions defining $\mathcal{D}$ and stays in such configurations forever after. When this happens, we say that the protocol has *converged* (or *terminated*), and a *terminal configuration* has been reached.[7] A *self-stabilizing protocol* is a protocol that converges from an arbitrary configuration (i.e., from a configuration where *any* agent, *including the leader*, can be in any possible state).

In the *naming* problem, each mobile agent x has a variable, also called a *name*, that eventually does not change and such that no two agents have the same name. Mobile agents having the same state (thus the same name) are called *homonyms*.

We consider *uniform* or *semi-uniform* protocols (cf. [28, 53]) in the sense that all agents, except the leader (whence semi-), are a priori indistinguishable and interact according to the same transition rules. Moreover, given an upper bound P on N, the protocol functions similarly for any N. Thus, given the bound P, by the definition of naming, an obvious lower bound on the state space of a mobile agent (for solving naming) is P.

3 Negative Results

In this section, we clarify some boundaries on the possibility to obtain symmetric naming. They are summarized in Table 1 and useful for establishing the space-optimality (tightness) of the solutions presented in the next section. The proofs in this section are concise, but not so simple, and some like of Prop. 4 are more intricate. First of all, we show that under weak fairness and without leader, symmetric naming is impossible.

▶ **Proposition 1.** *Under weak fairness and without leader (even with a uniform initialization of mobile agents), symmetric naming is impossible in the population protocol model.*

Proof. By contradiction, assume that such a symmetric protocol $\mathcal{PP}$ exists. With or without an initialization, consider a possible starting configuration where each agent is in state s_1 and the population size is even (also corresponds to a uniform initialization). We build a weakly fair infinite execution of $\mathcal{PP}$ during which no configuration with agents in distinct states is reached. This execution can be described by phases. In the first phase, the agents are matched in pairs and interact accordingly. As the protocol is symmetric, after this first phase, each agent is in some state s_2, the same for all agents. In the next phase, the

[7] In some recent literature on randomized population protocols, e.g., [29], the terms convergence and stabilization are distinguished (in contrast with the more original literature [4, 5]) and used to denote different from here notions. In this paper, to avoid ambiguity, we restrain the use of the term stabilization exclusively to non-initialized protocols (self-stabilizing ones), while convergence is used for both initialized an non-initialized ones.

agents are matched again in pairs, but differently from the previous phase, and interact accordingly. After the phase, each agent is in some state s_3, the same for all agents. The execution continues in such phases such that eventually every agent has interacted with every other. From now on, such an interaction sequence is repeated infinitely often, satisfying weak fairness. However, this execution never assigns distinct names to agents. ◀

The next proposition states the impossibility of a P-state symmetric naming with no leader.

▶ **Proposition 2.** *Even with a uniform initialization of agents, without a leader, and using only P states per agent, it is impossible to obtain symmetric naming in the population protocol model, under weak or global fairness.*

Proof. By contradiction assume that such a symmetric protocol $\mathcal{PP}$ exists. Consider a starting configuration with at least two homonyms. As these two homonyms have to receive distinct names and because the protocol is symmetric, a transition rule of the type $(s_1, s_2 \neq s_1) \rightarrow (s'_1, s'_2)$ *s.t.* $\{s_1, s_2\} \neq \{s'_1, s'_2\}$ has to exist in $\mathcal{PP}$. Notice that, if the size of the population is exactly P, the states s_1, s_2 have to be present in every configuration C after convergence. In this case, even after convergence, such a transition rule will be eventually executed (by either global or weak fairness), creating homonyms. This is a contradiction. ◀

The following lemma states properties of the transition rules of any P state symmetric naming protocol. It is used for proving the next proposition about the impossibility of a P state self-stabilizing symmetric naming.

▶ **Lemma 3.** *In any symmetric naming protocol using only P states per agent, the only possible non-null transition rules between two non-leader agents are between two homonyms (any two agents in the same state).*

Proof. Otherwise, in a population of P agents, after convergence (every state in $\{1, \ldots, P\}$ is assigned to some agent), mobile agents would be able to change their states and hence their names. This is a contradiction to the assumed convergence, according to the definition of the naming problem. ◀

▶ **Proposition 4.** *There is no symmetric naming protocol with P states per agent with an arbitrarily initialized leader (or without it), under weak or global fairness.*

Proof. Assume by contradiction that such a protocol exists. By Proposition 2, a leader is necessary. Consider a population of P mobile agents and a starting configuration C_0 (with possibly uniformly initialized mobile agents) that has homonyms (with states in $\{1, \ldots, P\}$). By Lemma 3, only transitions with a leader can eliminate homonyms (the only possible non-null symmetric transition rules between homonyms create necessarily two other homonyms). Thus, there is a finite execution sequence e, starting from C_0, during which the leader renames interacting mobile agents, and finally converges to a correct naming (where every name from $\{1, \ldots, P\}$ is assigned to some mobile agent). Denote by C_e and by s_e the configuration and the state of the leader, respectively, at the end of e.

Then, assume a starting configuration C'_0 (with possibly uniformly initialized mobile agents, as before) containing only homonyms in state s (in $\{1, \ldots, P\}$) and with the leader in state s_e. There must be a sequence of interactions between the leader and mobile agents starting from C'_0 which ends up with a transition during which an interacting agent changes its name. Such a sequence of interactions exists also starting from C_e (with either weak or global fairness), because in C_e any possible state exists in the population. This contradicts

the assumption that the protocol has converged starting from C_e in e. Hence, the proposition follows. ◀

3.1 Impossibility of P state symmetric naming of arbitrarily initialized mobile agents, under weak fairness, even with an initialized leader

For proving this stronger intricate impossibility result, let us assume, for the sake of contradiction, that such a solution exists. Thus, denote by $Name$ any symmetric protocol solving the naming problem under weak fairness (for any $N \leq P$), with arbitrarily initialized P-state mobile agents. By Proposition 2, under such conditions, a leader is necessary. Moreover, by Proposition 4, such an agent has to be initialized. So, in this sub-section, we assume such initialized unique leader. In the following, some additional necessary properties of protocol $Name$ are proven and finally imply the impossibility of its existence (see Theorem 11).

Using the lemma below, the next proposition establishes an important property of any protocol $Name$ - the existence of a unique state m, called *sink*. This particular state satisfies the following three conditions: (1) $(m,m) \rightarrow (m,m)$; (2) for every state $s \in Q$, there is a transition rule sequence $(s,s) \stackrel{*}{\rightarrow} (m,m)$; (3) for any $N < P$, no mobile agent is assigned a name m at convergence (i.e., m does not appear infinitely often in executions with $N < P$).

▶ **Lemma 5.** *Consider any weakly fair execution $e = C_1, C_2, C_3, \ldots, C_j, \ldots$ of $Name$ on a population $\mathcal{A}$ of size $N < P$. There is an integer k such that, for any $j \geq k$, no mobile agent is in a state $m \in Q$ such that there is a sequence of transitions of $Name$ $(m,m) \stackrel{*}{\rightarrow} (m,m)$.*

▶ **Proposition 6.** *In any protocol $Name$, there is a sink state.*

Proof. As $Name$ is symmetric, two interacting agents, both in some state $s \in Q$, execute some symmetric transition of the form $(s,s) \rightarrow (s_1,s_1)$. If they meet several times successively, there is a possible sequence of transitions $(s,s) \rightarrow (s_1,s_1) \rightarrow (s_2,s_2) \rightarrow (s_3,s_3) \ldots$ As mobile agents are finite state, for some $j > i \geq 1$, $s_i = s_j$, i.e. $(s_i,s_i) \stackrel{*}{\rightarrow} (s_i,s_i)$. By Lem. 5, $s_i = m$ s.t. m does not appear infinitely often in executions with $N < P$. As there are at least $P-1$ states appearing infinitely often in an execution with $N = P-1$, there is at most one such possible state m in a P state protocol. Thus, the first part of the lemma holds.

Finally, by contradiction, if $(m,m) \rightarrow (s,s)$ s.t. $s \neq m$, then the previous part of the proof implies $(s,s) \stackrel{*}{\rightarrow} (s,s)$. As m is proved (above) to be unique, this is a contradiction. ◀

From now on, the proof assumes the sink state denoted by m, and shows the impossibility for a particular kind of executions, called here *reduced*. In any segment of a reduced execution, each time a pair of $s \neq m$ homonyms appears, it is immediately *reduced* to m, i.e., by applying the sequence of transition rules $(s,s) \stackrel{*}{\rightarrow} (m,m)$, whose transitions and the sequence itself are called *(homonym) reducing*. Thus, other transitions can take place only when there are no more homonyms. Naturally, the obtained configuration without any homonyms, except those in state m (non-m ones), is called a *reduced configuration*. Notice that, in a reduced execution, there are non-reduced configurations, but only during the reducing sequences. For example, consider the following reduced sub-execution (where l_i represents a leader state): $[1,2,3,4,m,l_1], (l_1,1) \rightarrow (l_2,2), [2,2,3,4,m,l_2], (2,2) \rightarrow (m,m), [m,m,3,4,m,l_2], (l_2,m) \rightarrow (l_3,3), [m,3,3,4,m,l_3], (3,3) \rightarrow (m,m), [m,m,m,4,m,l_3]$. In this example, the reduced configurations are $[1,2,3,4,m,l_1], [m,m,3,4,m,l_2]$ and $[m,m,m,4,m,l_3]$.

Corollary 7 follows, because forcing a reducing sequence of transitions whenever possible, as in a reduced (sub-) execution, does not prevent an execution from being weakly fair (the proof is in the appendix).

▶ **Corollary 7.** *Given protocol Name, any reduced sub-execution of Name can be prolonged to a reduced weakly fair execution of Name, i.e., in which Name converges.*

The following lemma states a basic property of $Name$, in a population of P agents. It uses the notion of equivalent configurations. Two configurations are *equivalent* if both correspond to the same multi-set of states[8] (e.g., $C_1 = [2, 3, 2, m, l]$ is equivalent to $C_2 = [2, 2, 3, m, l]$, where l stands for the leader state). This notion is naturally extended to equivalent executions. The lemma shows that, in a population of P agents, if there is a reduced sub-execution in which some particular state $s \neq m$ never appears (in its reduced configurations), than there is also an *equivalent* sub-execution where a particular m-agent never interacts.

▶ **Lemma 8.** *In a population of P agents, consider a reduced sub-execution $e = CC_1C_2 \dots C_k$ of Name, starting from a reduced configuration C and such that no agent in state $s \neq m$ (for some s) exists in any reduced configuration of e. Then, there exists a reduced sub-execution $e' = CC'_1C'_2 \dots C'_k$ of Name in which a particular m-agent x never interacts, and, as in e, no agent in state $s \neq m$ exists in any reduced configuration of e'. Moreover, every configuration C'_i of e' is equivalent to the configuration C_i in e.*

Now, we want to prove that $Name$, in a population of size P, can reach a terminal configuration C with an agent x in some state $s \neq m$, in such manner that the leader may be unaware of that. Then, the leader should manage to create the possibly missing s-agent for converging towards a configuration where the naming is realized. But, once created, this s-agent can disappear by a reduction with the "hidden" s-agent x. This contradicts the fact that the reached configuration C is terminal (see the proof of Theorem 11). This proof uses the following two technical lemmas.

The first lemma (Lem. 9) states that, in some conditions, from a reduced configuration with an s-agent, a reduced configuration without such an agent is reachable. The second symmetric lemma (Lem. 10) states that, if there is some constrained sub-execution to a latter configuration without an s-agent, then there also exists a particular sub-execution from the configuration without an s-agent to the one with it. We use the following definitions to denote these aspects formally.

A configuration C_1 is said to be *far away* from C_2 *by one state* $s \neq m$ (in agent x), if there is an agent x such that $C_1[x] = m$, $C_2[x] = s \neq m$ and $\forall y \in \mathcal{A} \setminus \{x\}, C_1[y] = C_2[y] \neq s$. Then, C_1 is denoted by C_2^{-s} and C_2 by C_1^{+s}. Agent x is called the *pivot*. For example, $C_1 = [1, m, 3, 4, m, l_1]$ is far away by one state 2 from $C_2 = [1, 2, 3, 4, m, l_1]$, and $C_1 = C_2^{-2}, C_2 = C_1^{+2}$.

▶ **Lemma 9.** *Consider a population of size P and two reduced configurations C_1 and C_1^{-s}, far away by state s, in agent x. Consider a given reduced sub-execution $C_1^{-s}e_1^{-s}C_2$ of Name where: (i) there is no s-agent in every reduced configuration in the segment $C_1^{-s}e_1^{-s}$, (ii) agent x does not interact in $C_1^{-s}e_1^{-s}C_2$, (iii) C_2 is reduced and has exactly one s-agent. Then, there exists a reduced sub-execution $C_1e_1C_2^{-s}$ of Name such that exactly one agent in state s exists in every reduced configuration in C_1e_1, and agent x does not interact in $C_1\ e_1\ C_2^{-s}$, except in the very last (s-homonym) reducing sequence.*

▶ **Lemma 10.** *Consider a population of size P and two reduced configurations C_1 and C_1^{-s}, far away by state s, in agent x. Consider a given reduced sub-execution $C_1e_1C_2^{-s}$ of Name*

[8] or if one is a permutation of the vector components of the other (recall that a configuration is a vector of states of the agents)

where exactly one agent in state s exists in every reduced configuration in the segment C_1e_1, and agent x does not interact in $C_1\ e_1\ C_2^{-s}$, except in the very last (s-homonym) reducing sequence. Then, there exists a reduced execution $C_1^{-s}e_1^{-s}C_2$ of $Name$ such that there is no s-agent in any reduced configuration in the segment $C_1^{-s}e_1^{-s}$, and C_2 is reduced and has exactly one s-agent.

▶ **Theorem 11.** *Under weak fairness, without the initialization of mobile agents, there is no symmetric naming protocol with P states per agent.*

Proof. By contradiction, assume that such a protocol $Name$ exists. Consider a population of P agents. Assume an agent x that does not communicate (for long enough), while the protocol is converging with only $P-1$ mobile agents. By Proposition 6, when this happens, no agent, except possibly x, is in state m. Thus, consider two possible configurations. In one, C_1, the state of x is m, and in another, x is in state $s \neq m$. In the latter case, reduce the s-state homonyms (to m) (possible by Prop. 6). The reached configuration is C_1^{-s}.

Assume that the actual obtained configuration is C_1^{-s}. By the correctness of $Name$, starting from C_1^{-s}, any execution e converges to a configuration C_* where all agents are in different states and no state changes thereafter. Moreover, by Corollary 7, there is such an execution, which is reduced. Furthermore, any such e can be decomposed s.t. $e = C_1^{-s}e_1^{-s}C_2\ e_2\ C_3^{-s}e_3^{-s}C_4\ e_4\ C_5^{-s}\dots C_k^{-s}e_k^{-s}C_*C_*\dots$. For every odd i, no s-agent exists in any reduced configuration of $C_i^{-s}e_i^{-s}$, and C_i^{-s} is reduced. For every even i, exactly one s-state agent exists in every reduced configuration in C_ie_i, and C_i is reduced. By Lemma 8, for every odd i, one can choose a segment $C_i^{-s}e_i^{-s}$ such that a particular m-agent x_i never interacts in this segment.

Furthermore, e is chosen such that, for every segment $C_ie_iC_{i+1}^{-s}$ with an even i, a particular s-state agent $y_i \neq x_i$ does not interact, except in the very last (s-homonym) reducing sequence. Let us show (by induction) that such e exists. First, since in $C_i^{-s}e_i^{-s}$, for odd i, there is an m-agent $y \neq x_i$ in every reduced configuration. Thus, in the following C_{i+1} (even $i+1$) configuration, any such agent or other non-m-agent $y_i \neq x_i$ can become an s-agent. This implies that every agent in e has the opportunity to interact repeatedly. Second, by Lemma 9, $C_1e_1C_2^{-s}$ exists (such that there is exactly one agent in state s in every reduced configuration of C_1e_1).Thus, starting from C_1, at the end of $C_1e_1C_2^{-s}$, no s-agent would exist. Then, by correctness of $Name$, there exists $C_2^{-s}e_2^{-s}C_3$ (where no s-agent exists in $C_2^{-s}e_2^{-s}$), and by Lemma 8, such that, in $C_2^{-s}e_2^{-s}$, a particular m-state agent does not interact, e.g., the pivot agent of C_2^{-s} and C_2. Thus, by Lemma 9, there exists e, such that, in $C_2e_2C_3^{-s}$, a particular s-state agent y_2 does not interact, except in the very last (s-homonym) reducing sequence (this proves the base of induction).

Now, one can repeat the same arguments, for any segment $C_i^{-s}e_i^{-s}C_{i+1}e_{i+1}C_{i+2}^{-s}$, for an odd i, in e, and show (by induction) that the chosen e exists. Recall that we assumed that $Name$ has converged in C^* and then, the leader has to stop renaming the agents. In addition, C_* is reduced (all agents are distinctly named). Hence, from this point, only null-transitions are possible.

Notice that the conditions of Lemma 9 are satisfied for the segments $C_i^{-s}e_i^{-s}C_{i+1}$ with an odd i, and the conditions of Lemma 10 are satisfied for the segments $C_ie_iC_{i+1}^{-s}e_i^{-s}$ with an even i. Hence, by applying these lemmas repeatedly to the segments of e, one inductively builds the execution segment $e' = C_1e_1C_2^{-s}e_2^{-s}C_3e_3\dots C_ke_kC_*^{-s}$. However, C_*^{-s} is reduced, and far away by only one state from C_*. No agent except the pivot, can distinguish C_*^{-s} from C_*, since each one is in the same state in both configurations. In particular, $C_*^{-s}[leader] = C_*[leader]$. Thus, and by Lemma 3 and Prop. 6, the only possible transitions with the leader are null transitions, as well as the transitions involving mobile agents (there

are no non-m-homonyms). Obviously, in C_*^{-s}, the protocol has not converged yet. But, no transition can change the configuration C_*^{-s}. This contradicts the assumption that $Name$ is correct. ◀

4 Positive Results

We start by a proposition that illustrates the power of asymmetric transition rules, compared to symmetric ones. Basically, with asymmetric rules, a leader is not necessary for breaking symmetry, even under weak fairness. Moreover, P states are sufficient and no initialization is necessary, i.e., self-stabilizing space-optimal naming is possible.

The proof is by construction of a protocol with a single asymmetric type of rule, $(s, s) \to (s, s + 1 \mod P)$. This idea is known in the literature, e.g., [19, 12]. It aimed at solving other (than naming) problems, but provided naming as a by-product. [19] considers self-stabilizing leader election, assuming that the exact size of the population N is known (an assumption proven to be necessary). Under this assumption, the presented protocol also solves naming. In [12], a similar idea is used to count the arbitrarily initialized mobile agents, assuming an initialized leader, and realizes also naming. The asymmetric space-optimal naming protocol presented below is proven under more general assumptions (with upper bound P, instead of exact knowledge of N, without a leader, and under both fairness assumptions). Its proof uses the novel technique of *hole* and *hole distance* in a configuration.

▶ **Proposition 12.** *Even if agents cannot be initialized, asymmetric naming (under global or weak fairness) is possible using an optimal number of states (P) per agent and without leader.*

Proof. Consider the following asymmetric protocol with P-state agents and only one type of transition rules: $(s, s) \to (s, s + 1 \mod P)$.

To prove its correctness let us use the following definitions. A *hole in a configuration C* is an integer i such that no agent is in state i in C. The *hole distance of an agent, in state i, in a configuration C*, is the minimum integer j such that $i + j \mod P$ is a hole, if such an integer j exists, and 0 otherwise. The *hole distance of a configuration C* is the sum of the hole distances of the agents in C. Let f be the function mapping each configuration C to a pair of integers (number of holes in C, hole distance of C).

Let C and C' be two different configurations such that $C \to C'$. Let us show that, for the lexicographical order, $f(C) > f(C')$. First, remark that C' cannot have more holes than C. If C' has one hole less than C, we are done. If not (C' has the same holes as C), there is an agent in state i in C that has changed its state to $i + 1 \mod P$ in C'. Thus, the hole distance of C' is the hole distance of C minus 1. We are done also in this case.

Since f is upper bounded (e.g., by $(P, P(P - 1))$), there is a sequence of transitions that reaches a configuration from which only null transitions are possible, making no effect on agents' states, that are thus necessarily distinct. Then, naming is achieved. ◀

Now we consider one of the most difficult cases for symmetric protocols, without leader and initialization, impossible under weak fairness. Recall that global fairness mimics, in some sense, the behavior of randomized environments. The following proposition shows that this pseudo randomization is sufficient for breaking symmetry, and that a distinguishable agent is not needed for that. Thus, we propose below the first *symmetric space-optimal self-stabilizing* naming obtained without a leader. Notice, that by Proposition 2, at least $P + 1$ states per agent have to be used in this case. The complete proofs of the following sketches appear in the appendix.

▶ **Proposition 13.** *Even if agents cannot be initialized and without a leader, symmetric (self-stabilizing) naming under global fairness, for $N > 2$, is possible using $P + 1$ states per agent.*

Proof Sketch. Consider the following symmetric protocol defined by three types of transition rules:
1. $if\ s \neq P : (s, P) \to (s, s+1 \mod P)$; **2.** $if\ s \neq P : (s, s) \to (P, P)$; **3.** $(P, P) \to (1, 1)$.

From a configuration with homonyms, rule 2 can be applied repeatedly to obtain a configuration C' where there are only P-state homonyms, and possibly some other uniquely named agents. If, in C', no uniquely named agents exist, let us force transitions using rules 3 and 1, to create uniquely named agents. Then, rule 3 is applied again, to reach a configuration C with only P-state homonyms and with at least one uniquely named agent. If there are still some P-state homonyms in C, pick an agent with a unique name s such that no agent with a unique name $s + 1$ exists. Make the s-agent interact with some P-agent, applying rule 1. The obtained configuration contains one more unique name than in C. If naming is not yet reached, this scenario is repeated until it is reached. Such an execution segment is possible from any configuration with homonyms. Hence, naming is reached in any globally fair execution. ◀

Up to this point, we have covered the possible positive cases assuming that no leader is present. Now, we show that the impossibility results of Section 3 can be circumvented by the assumption of a distinguishable agent. This allows to obtain three space-optimal symmetric protocols: 1) a simple P state protocol with all agents being initialized, including the leader (Prop. 14); and two more intricate solutions: 2) a self-stabilizing one (with $P + 1$ states) under weak fairness (Prop. 16); and 3) a protocol using only P states under global fairness (Prop. 17).

▶ **Proposition 14.** *Given a unique initialized leader, and uniform initialization of mobile agents, naming is possible using only P states per agent, under weak or global fairness.*

The next two results exploit the existing space-optimal *counting* protocol from [11], which uses P states per mobile agent, and counts non-initialized mobile agents under weak fairness, assuming an initialized leader . It is given (Protocol 1) together with its description, in the appendix. It was not originally intended to be a naming protocol, neither a self-stabilizing one. However, it can be observed that, for the case of $N < P$, it performs (non self-stabilizing) naming. This property is reflected in Theorem 15 below. By augmenting the mobile agents' state space to $P + 1$ and adapting the protocol accordingly, it can be transformed to naming (also for the case of $N = P$), though using a non-optimal $P + 1$ number of states. This protocol is adapted further again for also solving the problem in a self-stabilizing way (Prop. 16). It is described and proven below (its pseudo-code is given by Protocol 2 in the appendix). In what follows, we adopt the terminology of [11], and call the leader *base station*, shortly BST.

▶ **Theorem 15** ([11]). *Protocol 1 solves the counting problem, under weak fairness, for up to P mobile agents, each with P states. Moreover, the protocol names up to $P - 1$ mobile agents with distinct names (for any $N < P$, the names are in $\{1, \dots N\}$).*

▶ **Proposition 16.** *Self-stabilizing (every agent state is initialized arbitrary) symmetric naming under weak fairness, is possible using $P + 1$ states per mobile agent, and given a unique (non-initialized) leader.*

Proof Sketch. The proof is by construction. We transform the counting Protocol 1 from [11] into a self-stabilizing naming, Protocol 2 (both appear in the appendix). First, as the

number of states now per mobile agent is $P+1$, we adapt the variables' declarations to accommodate the supplementary value P. By Theorem 15, for any $N \leq P$, the modified protocol assigns the mobile agents names in $\{1,\ldots,N\}$, if BST (the leader) is well initialized, i.e., if its only two variables n and k are initialized to 0.

To get a self-stabilizing version, we incorporate a reset technique, by adding lines 11 - 12. In line 12, the variables n and k are reset to 0. It is clear that, if this line is executed once, then the required naming is eventually and correctly obtained.

If naming is not yet reached, homonyms in the sink state $m=0$ exist (or continue to reappear). They will continue to interact with BST, increasing the values of k and n, till $n > P$ (and the condition in line 11) is satisfied. Then, n and k reset to 0 in the next line 12. ◀

Now, a protocol using only P states per mobile (non-initialized) agent is constructed. It is done by modifying Protocol 1 again, but only for the case of $n=P$, and by exploiting the global fairness assumption. The resulting protocol is not self-stabilizing, as the leader is initialized. However, the adaption (though done only for the case of $n=P$) together with the protocol analysis is far from being trivial.

▶ **Proposition 17.** *With an initialized leader (without initialization of mobile agents), symmetric naming under global fairness is possible using only P states per mobile agent.*

Proof Sketch. The proposed protocol is a modification of Protocol 1. It appears in the appendix - Protocol 3. For every $N<P$, it works in the same way as Protocol 1. Hence, by Theorem 15, it eventually converges to naming, for every $N<P$. The case of $N=P$ is treated separately, in lines 11 - 16. For this case, a variable *name_ptr* with possible values in $\{0,\ldots,P\}$ is used for indicating to BST (the leader) the name to assign to a mobile agent x interacting with it (using variable $name_x$, from the original protocol). The variable *name_ptr* is initialized to 0. Below we consider only the case of $N=P$.

Whenever, $n=P$, BST increments *name_ptr* each time it meets a mobile agent whose name is the current value of *name_ptr*. Otherwise, the agent is named by the value of *name_ptr*, and *name_ptr* is reset. Let us consider only reduced (to 0) executions (see the definition in Sec. 3.1). From any non-terminal configuration, there is a sequence of interactions during which, BST, first, resets *name_ptr*, and then meets the existed k uniquely named agents in the order of their names $0,1,2,3,\ldots j$. Variable *name_ptr* increases to j. Then, BST meets a 0-agent, which is named by $j+1$, if $j<P$, and reset *name_ptr* again. Then, the scenario repeats, until j is increased to P. No value can be changed thereafter, and all agents are named by names in $\{0,\ldots,P-1\}$. By global fairness, the terminal naming configuration is eventually reached. ◀

References

1. D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R. L. Rivest. Time-space trade-offs in population protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2560–2579, 2017. URL: https://doi.org/10.1137/1.9781611974782.169, doi:10.1137/1.9781611974782.169.
2. D. Alistarh, J. Aspnes, and R. Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2221–2239, 2018. URL: https://doi.org/10.1137/1.9781611975031.144, doi:10.1137/1.9781611975031.144.
3. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC*, pages 290–299, 2004.
4. 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.
5. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
6. D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. In *OPODIS*, pages 103–117, 2005.
7. D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4), 2008.
8. Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 292–299, New York, NY, USA, 2006. ACM Press. doi:http://doi.acm.org/10.1145/1146381.1146425.
9. J. Beauquier and J. Burman. Self-stabilizing synchronization in mobile sensor networks with covering. In *DCOSS*, volume 6131 of *Lecture Notes in Computer Science*, pages 362–378. Springer, 2010.
10. J. Beauquier and J. Burman. Self-stabilizing mutual exclusion and group mutual exclusion for population protocols with covering. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, volume 7109 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2011.
11. J. Beauquier, J. Burman, S. Clavière, and D. Sohier. Space-optimal counting in population protocols. In *DISC*, pages 631–646, 2015. URL: https://doi.org/10.1007/978-3-662-48653-5_42, doi:10.1007/978-3-662-48653-5_42.
12. J. Beauquier, J. Clement, S. Messika, L. Rosaz, and B. Rozoy. Self-stabilizing counting in mobile sensor networks with a base station. In *DISC*, pages 63–76, 2007.
13. L. Becchetti, L. Bergamini, F. Ficarola, and A. Vitaletti. Population protocols on real social networks. In *Proceedings of the 9th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, PE-WASUN 2012, Paphos, Cyprus, October 21-25, 2012*, pages 17–24, 2012. URL: http://doi.acm.org/10.1145/2387027.2387031, doi:10.1145/2387027.2387031.
14. L. Becchetti, A. E. F. Clementi, E. Natale, F. Pasquale, P. Raghavendra, and L. Trevisan. Friend or foe? population protocols can perform community detection. *CoRR*, abs/1703.05045, 2017. URL: http://arxiv.org/abs/1703.05045, arXiv:1703.05045.
15. A. Belleville, D. Doty, and D. Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 141:1–141:14, 2017. URL: https://doi.org/10.4230/LIPIcs.ICALP.2017.141, doi:10.4230/LIPIcs.ICALP.2017.141.

16 O. Bournez, J. Chalopin, J. Cohen, and X. Koegler. Playing with population protocols. In *Proceedings International Workshop on The Complexity of Simple Programs, CSP 2008, Cork, Ireland, 6-7th December 2008.*, pages 3–15, 2008. URL: http://arxiv.org/abs/0906.3256.

17 O. Bournez, J. Chalopin, J. Cohen, X. Koegler, and M. Rabie. Population protocols that correspond to symmetric games. *IJUC*, 9(1-2):5–36, 2013. URL: http://www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/ijuc-volume-9-number-1-2-2013/ijuc-9-1-2-p-5-36/.

18 O. Bournez, J. Cohen, and M. Rabie. Homonym population protocols. *Theory Comput. Syst.*, 62(5):1318–1346, 2018. URL: https://doi.org/10.1007/s00224-017-9833-2, doi:10.1007/s00224-017-9833-2.

19 S. Cai, T. Izumi, and K. Wada. How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model. *Theory Comput. Syst.*, 50(3):433–445, 2012.

20 J. Chalopin and D. Paulusma. Graph labelings derived from models in distributed computing: A complete complexity classification. *Networks*, 58(3):207–231, 2011. URL: https://doi.org/10.1002/net.20432, doi:10.1002/net.20432.

21 I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively mobile communicating machines that use restricted space. *Theor. Comput. Sci.*, 412(46):6469–6483, 2011. URL: https://doi.org/10.1016/j.tcs.2011.07.001, doi:10.1016/j.tcs.2011.07.001.

22 R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Labeling schemes for tree representation. In *Distributed Computing - IWDC 2005, 7th International Workshop, Kharagpur, India, December 27-30, 2005, Proceedings*, pages 13–24, 2005. URL: http://dx.doi.org/10.1007/11603771_2, doi:10.1007/11603771_2.

23 C. Cooper, A. Lamani, G.i Viglietta, M. Yamashita, and Y. Yamauchi. Constructing self-stabilizing oscillators in population protocols. *Inf. Comput.*, 255:336–351, 2017. URL: https://doi.org/10.1016/j.ic.2016.12.002, doi:10.1016/j.ic.2016.12.002.

24 G. Cordasco and L. Gargano. Space-optimal proportion consensus with population protocols. In *Stabilization, Safety, and Security of Distributed Systems - 19th International Symposium, SSS 2017, Boston, MA, USA, November 5-8, 2017, Proceedings*, pages 384–398, 2017. URL: https://doi.org/10.1007/978-3-319-69084-1_28, doi:10.1007/978-3-319-69084-1_28.

25 C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *DCOSS*, pages 51–66, 2006.

26 Z. Diamadi and M. J. Fischer. A simple game for the study of trust in distributed systems. *Wuhan University Journal of Natural Sciences*, 6(1):72–82, Mar 2001. URL: https://doi.org/10.1007/BF03160228, doi:10.1007/BF03160228.

27 E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. of the ACM*, 17(11):643–644, Nov. 1974.

28 S. Dolev, A. Israeli, and S. Moran. Self-stabilization of dynamic systems assuming only read/write atomicity. *DC*, 7(1):3–16, 1993.

29 D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. In *DISC*, pages 602–616, 2015. URL: http://dx.doi.org/10.1007/978-3-662-48653-5_40, doi:10.1007/978-3-662-48653-5_40.

30 Ö. Egecioglu and A. K. Singh. Naming symmetric processes using shared variables. *Distributed Computing*, 8(1):19–38, 1994. URL: http://dx.doi.org/10.1007/BF02283568, doi:10.1007/BF02283568.

31 M. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *OPODIS*, pages 395–409, 2006.

32 P. Fraigniaud, A. Pelc, D. Peleg, and S. Perennes. Assigning labels in an unknown anonymous network with a leader. *Distributed Computing*, 14(3):163–183, 2001. URL: http://dx.doi.org/10.1007/PL00008935, doi:10.1007/PL00008935.

33 L. Gasieniec, D. D. Hamilton, R. Martin, P. G. Spirakis, and G. Stachowiak. Deterministic population protocols for exact majority and plurality. In *20th International Conference on Principles of Distributed Systems, OPODIS 2016, December 13-16, 2016, Madrid, Spain*, pages 14:1–14:14, 2016. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2016.14, doi:10.4230/LIPIcs.OPODIS.2016.14.

34 L. Gasieniec and G. Stachowiak. Fast space optimal leader election in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2653–2667, 2018. URL: https://doi.org/10.1137/1.9781611975031.169, doi:10.1137/1.9781611975031.169.

35 D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81 (25):2340 – 2361, 1977.

36 R. Guerraoui and E. Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *ICALP (2)*, pages 484–495, 2009.

37 T. Izumi. On space and time complexity of loosely-stabilizing leader election. In *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, pages 299–312, 2015. URL: https://doi.org/10.1007/978-3-319-25258-2_21, doi:10.1007/978-3-319-25258-2_21.

38 T. Izumi, K. Kinpara, T. Izumi, and K. Wada. Space-efficient self-stabilizing counting population protocols on mobile sensor networks. *Theor. Comput. Sci.*, 552:99–108, 2014. URL: http://dx.doi.org/10.1016/j.tcs.2014.07.028, doi:10.1016/j.tcs.2014.07.028.

39 H. Jiang. *Distributed Systems of Simple Interacting Agents*. PhD thesis, Yale University, 2007.

40 X. Koegler. *Population protocols, games, and large populations*. Theses, Paris Diderot Unviersity, September 2012. URL: https://hal.inria.fr/tel-01274140.

41 S. Kutten, R. Ostrovsky, and B. Patt-Shamir. The las-vegas processor identity problem (how and when to be unique). *J. Algorithms*, 37(2):468–494, 2000. URL: http://dx.doi.org/10.1006/jagm.2000.1110, doi:10.1006/jagm.2000.1110.

42 R. J. Lipton and A. Park. The processor identity problem. *Inf. Process. Lett.*, 36(2):91–94, 1990. URL: http://dx.doi.org/10.1016/0020-0190(90)90103-5, doi:10.1016/0020-0190(90)90103-5.

43 N.A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1997.

44 O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Mediated population protocols. *Theor. Comput. Sci.*, 412(22):2434–2450, 2011.

45 O. Michail, I. Chatzigiannakis, and P. G. Spirakis. *New Models for Population Protocols*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2011. URL: https://doi.org/10.2200/S00328ED1V01Y201101DCT006, doi:10.2200/S00328ED1V01Y201101DCT006.

46 O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Naming and counting in anonymous unknown dynamic networks. In *Stabilization, Safety, and Security of Distributed Systems - 15th International Symposium, SSS 2013, Osaka, Japan, November 13-16, 2013. Proceedings*, pages 281–295, 2013. URL: http://dx.doi.org/10.1007/978-3-319-03089-0_20, doi:10.1007/978-3-319-03089-0_20.

47 Y. Mocquard, E. Anceaume, and B. Sericola. Optimal proportion computation with population protocols. In *15th IEEE International Symposium on Network Computing*

and Applications, NCA 2016, Cambridge, Boston, MA, USA, October 31 - November 2, 2016, pages 216–223, 2016. URL: https://doi.org/10.1109/NCA.2016.7778621, doi:10.1109/NCA.2016.7778621.

48 M. Rabie. Global versus local computations: Fast computing with identifiers. In *Structural Information and Communication Complexity - 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017, Revised Selected Papers*, pages 90–105, 2017. URL: https://doi.org/10.1007/978-3-319-72050-0_6, doi:10.1007/978-3-319-72050-0_6.

49 E. Ruppert. The anonymous consensus hierarchy and naming problems. In *Principles of Distributed Systems, 11th International Conference, OPODIS 2007, Guadeloupe, French West Indies, December 17-20, 2007. Proceedings*, pages 386–400, 2007. URL: http://dx.doi.org/10.1007/978-3-540-77096-1_28, doi:10.1007/978-3-540-77096-1_28.

50 Y. Sudo, T. Masuzawa, A. K. Datta, and L. L. Larmore. The same speed timer in population protocols. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 252–261, 2016. URL: https://doi.org/10.1109/ICDCS.2016.82, doi:10.1109/ICDCS.2016.82.

51 Y. Sudo, J. Nakamura, Y. Yamauchi, F. Ooshita, H. Kakugawa, and T. Masuzawa. Loosely-stabilizing leader election in a population protocol model. *Theor. Comput. Sci.*, 444:100–112, 2012. URL: https://doi.org/10.1016/j.tcs.2012.01.007, doi:10.1016/j.tcs.2012.01.007.

52 Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa. Loosely-stabilizing leader election on arbitrary graphs in population protocols without identifiers nor random numbers. In *19th International Conference on Principles of Distributed Systems, OPODIS 2015, December 14-17, 2015, Rennes, France*, pages 14:1–14:16, 2015. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2015.14, doi:10.4230/LIPIcs.OPODIS.2015.14.

53 G. Tel. *Introduction to Distributed Algorithms (2nd ed.)*. Cambridge University Press, 2000.

54 S.-H. Teng. Space efficient processor identity protocol. *Inf. Process. Lett.*, 34(3):147–154, 1990. URL: http://dx.doi.org/10.1016/0020-0190(90)90094-E, doi:10.1016/0020-0190(90)90094-E.

55 H. Yasumi, F. Ooshita, K. Yamaguchi, and M. Inoue. Constant-space population protocols for uniform bipartition. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, pages 19:1–19:17, 2017. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2017.19, doi:10.4230/LIPIcs.OPODIS.2017.19.

Appendix

State of the art.

The problem of distributing distinct identifiers to undistinguishable processors has received a lot of attention in the model of shared memory. The problem has been formally introduced by Lipton and Clark in [42], under the name of Processor Identity Problem (PIP), where a solution, under the form of a probabilistic $O(N^2)$ protocol, was given. This solution was improved in [54], which presented an $O(N\log^2 N)$ solution. In [30], a randomized protocol assigns distinct identifiers to the processes within an expected polynomial number of rounds using a polynomial number of boolean atomic variables. The authors of [41] propose a Las Vegas randomized algorithm which terminates in $O(\log N)$ expected rounds and uses $O(N)$ shared memory space.

In [49], it is investigated whether the assumption of unique identifiers is essential for wait-free distributed computing using shared objects of various types. Results on the solvability of two key problems, consensus and naming, are given. In the *synchronous* model of dynamic graphs, [46] studies the naming problem, assuming that the nodes are (uniformly) initialized. In [32], the task of assigning distinct labels to nodes of an unknown anonymous network in a distributed manner is considered. The model is synchronous and there is a distinguishable node. The goal is to assign short labels, as fast as possible. [22] solves the naming problem probabilistically, similarly assuming that no knowledge on the communication graph is given. The size of the identifiers is proven to be $O(\log N)$ with high probability and their expected size is $O(\log N)$ too. Finally, notice that [20] considers eleven different models of distributed systems and studies the computational complexity conditions for the naming problem to be solvable on a given network.

At the difference of the current work, all the previous references provide either probabilistic solutions or perform an asymptotic complexity analysis. Moreover, the computation models considered there have important differences with population protocols.

Concerning the same version of population protocols as considered in the current study, the naming problem has been solved as a by-product in studies of other related problems, like counting [12, 38, 11] and leader election [19]. The current work improves on all these results in terms of either space complexity or/and model assumptions.

A series of recent works (e.g., [1, 2, 34, 29]), considering random population protocols (where each interaction is chosen uniformly at random) jointly contribute to the study of the trade-offs between time and space, for fundamental tasks of majority and leader election. They lead to a complete suit of asymptotically space optimal randomized solutions for these two tasks. Other recent works study different important problems, in the same model and from a similar general perspective of space and/or time-optimality. The results are still in terms of asymptotic complexity and under randomness conditions (available both in the model and in the protocols). Examples of considered problems are proportion computation [47], proportion consensus [24], plurality consensus [33], function computation [15] and community detection [14].

In contrast, [23] introduces the problem of (self-stabilizing) oscillator construction in population protocols, and studies the *exact* state-space complexity of the possible solutions, in the model version considered here. Finally, another relevant recent work [**?**] devoted to uniform bipartition in population protocols, presents a similar type of comprehensive analysis, considering various combinations of model parameters. Differently from the analysis here, however, only the feasibility and optimality of *constant*-space complexity solutions are considered. Cases where the necessary state complexity depends on N are not treated by the exact state-space analysis.

Conclusion and Perspectives

This paper studies a strong form of symmetry breaking, giving distinct names to indistinguishable agents in population protocols. It provides a comprehensive overview of the results in terms of possibility, impossibility and space optimality, and some insights on the trade-offs between criteria (global vs. weak, symmetric vs. asymmetric, need of a leader, initialization).

A continuation of this work could be the study of the time complexity aspects of naming and, overall, of the trade-offs between time and space complexities. Another perspective would be to consider other forms of symmetry breaking (compact naming, leader election, coloring, two-hop coloring, majority, etc.), under constraints of optimal memory space and

requirements of fault-tolerance.

Negative Results (Section 3) Proofs

Lemma 5. *Consider any weakly fair execution $e = C_1, C_2, C_3, \ldots, C_j, \ldots$ of $Name$ on a population $\mathcal{A}$ of size $N < P$. There is an integer k such that, for any $j \geq k$, no mobile agent is in a state $m \in Q$ such that there is a sequence of transitions of $Name$ $(m, m) \xrightarrow{*} (m, m)$.*

Proof. Let us assume, by contradiction, that there are infinitely many configurations in e with a mobile agent in state m. Since there is a finite number of agents, there is a particular mobile agent x in $\mathcal{A}$ which is in state m in infinitely many configurations. Let $C_{j_1}, C_{j_2}, C_{j_3}, \ldots$ be these configurations such that $e = e_1, C_{j_1}, e_2, C_{j_2}, e_3, C_{j_3}, \ldots$ W.l.o.g., we choose these configurations such that, in every execution segment e_i, every agent in $\mathcal{A}$ interacts with every other (this is possible with weak fairness).

Now consider a population $\mathcal{A}' = \mathcal{A} \cup \{x'\}$ of size $n + 1$. To prove the lemma, we will construct a weakly fair execution e' of $Name$ in population $\mathcal{A}'$ where no agent can distinguish e' from e, and where consequently $Name$ wrongly names the agents. Precisely, in e', x and x' will be simultaneously in state m in infinitely many configurations.

We construct e' based on e. First, we assume that in e', x' is in state m in the starting configuration, and $e' = e'_1, C'_{j_1}, e^m, e'_2, C'_{j_2}, e^m, e'_3, C'_{j_3}, e^m, \ldots$ Every segment e'_i follows exactly the same transition sequence as in e_i. In every segment $e'_{2r+1}, C'_{j_{2r+1}}$ (for $r \geq 0$) the interactions are exactly the same as in $e_{2r+1}, C_{j_{2r+1}}$, and x' does not interact. However, in $e'_{2r}, C'_{j_{2r}}$, all the interactions are as in $e_{2r+2}, C_{j_{2r}}$, but the interactions with x. In this case, x is replaced by x' in the appropriate state, and x does not interact. Finally, e^m is an execution segment where only x and x' interact. They both start in state m, performing the sequence $(m, m) \xrightarrow{*} (m, m)$. The configurations at the beginning and at the end of e^m are identical. The construction of e' ensures that in every C'_{j_i}, both x and x' are in the state m.

It is easy to verify that e' is possible. In particular, this is because, at the end of every segment e'_i, C'_{j_i}, e^m, both x and x' are in the state m, so they can be exchanged in the following transitions of e'_{i+1}. Moreover, e' is weakly fair, because x' interacts with x in every e^m; in every e'_{2r+1} and e'_{2r+2}, x and x', respectively, interact with every other agent (by the assumption on e_i); and all the other agents interact with all the others infinitely often, by the later arguments and by weak fairness of e.

Finally, in e', $Name$ does not name x and x' differently. This is a contradiction to the assumption that $Name$ is a correct naming protocol. ◀

Corollary 7. *Given protocol $Name$, any reduced sub-execution of $Name$ can be prolonged to a reduced weakly fair execution of $Name$, i.e., in which $Name$ converges.*

Proof. First, recall that, by Prop. 6, and Lemma 3, the only non-null-transitions of $Name$ are the homonym reducing transitions between non-m-mobile agents and the transitions with the leader. Given a reduced segment, we prolong the execution by forcing mobile agents to interact with every other agent (including the leader) in a "round-robin fashion" (not necessarily in consecutive interactions). Whenever homonyms are created, this interaction pattern is interrupted by the homonym reducing sequence of transitions, and then resumed after the reduction. The execution constructed that way is weakly fair and uses only transitions of $Name$, hence it converges towards a naming. ◀

Lemma 8. *In a population of P agents, consider a reduced sub-execution $e = CC_1C_2 \dots C_k$ of $Name$, starting from a reduced configuration C and such that no agent in state $s \neq m$ (for some s) exists in any reduced configuration of e. Then, there exists a reduced sub-execution $e' = CC'_1C'_2 \dots C'_k$ of $Name$ in which a particular m-agent x never interacts, and, as in e, no agent in state $s \neq m$ exists in any reduced configuration of e'. Moreover, every configuration C'_i of e' is equivalent to the configuration C_i in e.*

Proof. In a population of P agents, in any reduced configuration (of a reduced execution of $Name$), there is at least one m-state agent. Thus in any reduced configuration of e there are at least two m-state agents. Consider C and call one of these two agents agent x.

Let us construct now e', starting in C, with exactly the same trace of transition rules of e. By Prop. 6, and Lemma 3, the only non-null-transitions are homonym reducing transitions between non-m-mobile agents and the transitions with the leader. By a simple induction below, one can see that every transition rule in a trace of e, step by step, can be executed without participation of agent x, and thus added to e'. Since no transition of e creates an s-agent, no such added transition can create an s-agent.

Thus, starting in C, the first transition of e can be executed with any agent, except x, and thus can be added to e (the base of induction). The resulting configuration C'_1 is equivalent to C_1. Then, by induction, assume that after k transitions added to e', the reached configuration is equivalent to the one reached after transition k in e (without any s-state agent in a reduced configuration). For $k+1$, if a homonym reducing transition takes place in e, x does not participate (it is already reduced) and thus the same transition can be added to e', and an equivalent configuration is reached. Otherwise, if this is a reduced configuration, there is an additional m-state agent, different from x, so any transition with the leader can be executed excluding x, and an equivalent configuration is reached. By such construction of e', every configuration C'_i is equivalent to C_i in e. ◀

Lemma 9. *Consider a population of size P and two reduced configurations C_1 and C_1^{-s}, far away by state s, in agent x. Consider a given reduced sub-execution $C_1^{-s}e_1^{-s}C_2$ of $Name$ where: (i) there is no s-agent in every reduced configuration in the segment $C_1^{-s}e_1^{-s}$, (ii) agent x does not interact in $C_1^{-s}e_1^{-s}C_2$, (iii) C_2 is reduced and has exactly one s-agent. Then, there exists a reduced sub-execution $C_1e_1C_2^{-s}$ of $Name$ such that exactly one agent in state s exists in every reduced configuration in C_1e_1, and agent x does not interact in $C_1\ e_1\ C_2^{-s}$, except in the very last (s-homonym) reducing sequence.*

Proof. Given a sub-execution $C_1^{-s}e_1^{-s}C_2$, the sub-execution C_1e_1 is constructed, starting from a configuration C_1, by applying exactly the trace of transition rules of $C_1^{-s}e_1^{-s}C_2$, on the population excluding the s-state agent x (recall that agent x does not interact in $C_1^{-s}e_1^{-s}C_2$). At the end of the execution constructed till now, there are exactly two s-state homonyms (x and another s-agent). Now, the transitions reducing these homonyms to m are added to reach the desired configuration C_2^{-s}. In this way, the sub-execution $C_1e_1C_2^{-s}$ is obtained. Notice that agent x interacts only in the very last (s-homonym) reducing sequence. ◀

Lemma 10. *Consider a population of size P and two reduced configurations C_1 and C_1^{-s}, far away by state s, in agent x. Consider a given reduced sub-execution $C_1e_1C_2^{-s}$ of $Name$ where exactly one agent in state s exists in every reduced configuration in the segment C_1e_1, and agent x does not interact in $C_1\ e_1\ C_2^{-s}$, except in the very last (s-homonym) reducing sequence. Then, there exists a reduced execution $C_1^{-s}e_1^{-s}C_2$ of $Name$ such that there is no s-agent in any reduced configuration in the segment $C_1^{-s}e_1^{-s}$, and C_2 is reduced and has exactly one s-agent.*

Proof. In a given execution $C_1 e_1 C_2^{-s}$ agent x does not interact, except in the very last (s-homonym) reducing sequence. Starting from C_1^{-s} we construct an execution $C_1^{-s} e_1^{-s} C_2$, by using first exactly the same prefix of the trace of transition rules of $C_1 e_1 C_2^{-s}$, until and excluding the very last (s-homonym) reducing sequence. In the given configuration, before this very last reducing sequence, two s-agents necessarily exist, one of them being till now the non-interacting agent x. However, in the sub-execution constructed at this point, x does not interact either, but is in state m, so exactly one s-agent exists at the end of the constructed sub-execution. Hence, C_2 is reached and the required $C_1^{-s} e_1^{-s} C_2$ is obtained. ◀

Positive Results (Section 4) Proofs

Proposition 13. *Even if agents cannot be initialized and without leader, symmetric (self-stabilizing) naming under global fairness, for $N > 2$, is possible using $P + 1$ states per agent.*

Proof. Consider the following symmetric protocol defined by three types of transition rules: **1.** $s \neq P : (s, P) \to (s, s + 1 \mod P)$; **2.** $s \neq P : (s, s) \to (P, P)$; **3.** $(P, P) \to (1, 1)$.

Notice that the states of the agents are in $\{0, \ldots, P\}$. To prove correctness, let us show that from any configuration C there exists a reachable configuration C^* such that in C^* all agents have distinct names in $\{0, \ldots, P-1\}$. Thus, consider a configuration C and make interact homonyms having a state different from P, iteratively and as many times as possible. These agents have to execute transition 2 and change their states to P. When transition 2 cannot be executed anymore, and if, in the resulting configuration, there are some non-homonyms in states different from P, we denote this configuration by C'. Otherwise, if all the agents are in state P, let us force interactions for reaching this kind of a configuration C'. First, make two agents x and y interact, applying the transition rule 3, changing their states to 1. Then, make y interact with an agent $z \notin \{x, y\}$ in state P, applying rule 1 (z exists as long as $N > 2$). After that, agents x, y, z are in states $1, 1, 2$ respectively. Make x interact with y to change their states to P. Now, every agent, excluding z, is in state P (z is in state 2). This configuration is of the same type as C', with some non-homonyms apart the agents in state P. Next, we show that a configuration of type C^* is reachable from any such C'.

For that, we associate the following sequence of states to C'. We order all the states different from P and present in C', to obtain an increasing sequence $s_1, s_2, \ldots s_k$ (recall that the states are numbers). In addition, let us add s_1 at the end of this sequence, resulting in $s_1, s_2, \ldots s_k, s_1$. Two consecutive states s_i, s_j in such a sequence are called *distant* if, either $[j = i + 1$ and $s_j - s_i > 1]$, or $[i = k, j = 1$ and $s_k - s_1 < P - 1]$. Notice that, for any distant s_i, s_j, $s_i + 1 \mod P$ is not present in C', so the transition rule 1 applied with the s_i-agent in C' cannot create homonyms.

If in C' there is an agent x in state P, choose two consecutive distant states s_i, s_j and make an agent in state s_i interact with x (applying rule 1 of the protocol). The obtained configuration C'' is of the same type as C' - the only possible homonyms are in state P and there are agents in states different from P. Associate to C'' an increasing sequence of ordered states as before. If there is still an agent y in state P and two neighboring distant states s'_i, s'_j, make y interact with an agent in s'_i, applying rule 1. Continue similarly from the resulting configuration, until no agents in state P exist. This finally occurs, because each newly obtained configuration has no homonyms, except possibly the agents in state P, the number of which strictly decreases in each next configuration. When no agents in state P exist anymore, configuration C^* is obtained. As C^* is infinitely often reachable (from any

configuration C), by global fairness, such configuration is eventually reached, providing a naming. In C^*, no transition rule is applicable. ◀

Proposition 14. *Given a unique initialized leader, and uniform initialization of mobile agents, naming is possible using only P states per agent, under weak or global fairness.*

Proof. Consider, for example, the following protocol. The mobile agents are initially in the state P, and the leader has a counter (of size P) initialized to 1. Whenever the leader interacts with a P-state agent, and the counter is less than P: the state of the agent is set to the counter value and the counter is incremented, if less than P. By a simple induction, one can easily prove that, after the kth interaction of the leader with a P-agent, for $k < P$, the agent is named k and never changes its name after. If the leader's counter reaches P (for $N = P$), only null transitions are possible, all the agents are named from 1 to P, and the protocol terminates. Hence, the protocol converges to a configuration where all $N \leq P$ agents have distinct names, using only P states per agent. ◀

Proposition 16. *Self-stabilizing (every agent state is initialized arbitrary) symmetric naming under weak fairness, is possible using $P + 1$ states per mobile agent, and given a unique (non-initialized) leader.*

Proof. The proof is by construction. We transform the counting Protocol 1 from [11] into a self-stabilizing naming, Protocol 2 (both appear below). First, as the number of states now per mobile agent is $P + 1$, we adapt the variables' declarations to accommodate the supplementary value P. By Theorem 15, for any $N \leq P$, the modified protocol assigns the mobile agents names in $\{1, \ldots, N\}$, if BST (the leader) is well initialized, i.e., if its only two variables n and k are initialized to 0.

To get a self-stabilizing version of the obtained till now naming protocol, we incorporate a reset technique, by adding lines 11 - 12. In line 12, the variables n and k are reset to 0. It is clear that, if this line is executed once, then the required naming is eventually and correctly obtained. Thus, let us show that, either a naming is obtained without executing the line 12, or it is executed once and the naming is obtained eventually after (without any additional execution of line 12). In other words, let us show that the condition in line 11 will be eventually satisfied, if and only if the mobile agents are not already distinctly named. Whenever there are homonyms in the population, they are either in state 0, or enter this state eventually, by lines 15 - 16. An agent in state 0 eventually interacts with BST and, if the condition of line 11 ($n > P \wedge (name_x = 0)$) is not satisfied yet, the one in line 2 ($n \leq P \wedge (name_x = 0 \vee name_x > n)$) is satisfied. These are the only two possibilities, for the interaction with a 0-agent. Let us consider the lines 3 - 9, conditioned by line 2. So, if $n \leq P$, and no naming is yet reached, these lines continue to be executed. In this case, after the starting configuration (with some value in k and n), whenever line 6 is executed, $k > l_n$ in line 7 is satisfied, and n is incremented in the next line. Otherwise, line 4 is executed and k is incremented at each interaction of a 0-agent with BST. Eventually, if homonyms do not disappear (naming is not reached) beforehand, k becomes large enough, so that n is incremented in line 8. Thus, eventually n becomes large enough and the condition in line 11 ($n > P \wedge (name_x = 0)$) becomes true, if naming is not obtained yet. Then, variables n and k are reset to 0 in the next line, and the required naming is eventually obtained, by the correctness of the original (non-self-stabilizing) Protocol 1. ◀

Proposition 17. *With an initialized leader (without initialization of mobile agents), symmetric naming under global fairness is possible using only P states per mobile agent.*

Proof. The proposed protocol is a modification of Protocol 1. It appears below - Protocol 3. For every $N < P$, it works in the same way as Protocol 1. Hence, by Theorem 15, it eventually converges to naming, for every $N < P$. The case of $N = P$ is treated separately, in lines 11 - 16. For this case, a variable *name_ptr* with possible values in $\{0, \ldots, P\}$ is used for indicating to BST (the leader) the name to assign to a mobile agent x interacting with it (using variable $name_x$, from the original protocol). The variable *name_ptr* is initialized to 0. Below we consider only the case of $N = P$.

Whenever, $n = P$, i.e., the protocol has detected that the population size is P, initialize to 0 *name_ptr* is either incremented or reset to 0 again, at every interaction involving BST. Whenever it becomes equal to P, only null-transitions with BST are possible. Let us show that the protocol has then converged to naming. For that, we give an execution segment e that ends up with a configuration where every mobile agent is distinctly named, so after that, only null-transitions are possible and the protocol has converged. Moreover, the configurations of the constructed e, are all reachable from any configuration in any execution of the proposed protocol, i.e., infinitely often (in case of $N = P$). This implies that the protocol converges to naming in any globally fair execution.

First of all, observe that, from any configuration, we can reach a reduced (to 0) configuration (see the definition in Sec. 3.1). Let us consider a reduced execution segment e. From any configuration in any execution, e starts by a reducing sequence reaching a reduced configuration. Now, if *name_ptr* $\neq 0$, let a 0-state agent (existing in any reduced configuration) interact with BST, and reset *name_ptr* to 0. In the obtained configuration C, mobile agent states have not been changed and *name_ptr* $= 0$. In C, consider a strictly increasing, each time exactly by 1, sequence of present names, starting from the name 0, i.e., $0, 1, 2, 3, \ldots j$. Consider the corresponding sequence of mobile agents with these names, $x_0, x_1, x_2, \ldots, x_j$. Make these agents interact with BST, one by one, in this order. If no additional 0-agent exists (apart of x_0), at the end of this interaction sequence of e, *name_ptr* $= P$ and naming is reached. Otherwise, if there is an additional 0-agent, make it interact with BST. In the reached configuration C', *name_ptr* $= 0$ and the previous sequence of agents (with strictly increasing, each time by 1, names) is now longer by at least one mobile agent ($x_0, x_1, x_2, \ldots, x_{j+i}$, for $i \geq 1$). Repeat a similar sequence of interactions, for the new strictly increasing sequence of existing names. If then naming is not yet obtained, repeat again until it is obtained. The overall constructed sequence of interactions is the required segment e. ◀

Space-Optimal Counting from [11]

Protocol 1 below is a symmetric space-optimal protocol from [11] solving the counting problem under weak fairness (see Theorem 15). It uses at most P states per mobile agent and is correct starting from arbitrary states in these agents, but not in BST (the leader).

In Protocol 1, BST eventually counts the mobile agents and stores the value in variable n. To realize this, BST successively attempts to guess the number of mobile agents in the population, starting from 1 and ending with N (this guess is stored in n). For each guess $n < P$, BST tries to name (differently) mobile agents in state 0 (zero-state) interacting with BST (lines 3 and 9). That is, BST tries to assign to these agents distinct states from $\{1, \ldots, n\}$ (also called here names). State 0 has a special technical role. Whenever two agents with identical names (*homonyms*) interact, they change their state to 0 (line 20). Thus, 0-state agents indicate to BST that homonyms are still present in the population. The names are given to 0-state agents one by one following some finite sequence U^* of names (line 9), depending on P. Sequence U^* guarantees that, if there are $N < P$ agents, whatever

their starting states are, the naming succeeds. If no naming succeeds, BST concludes that there are more than $P-1$ agents, that is $N=P$. Thus, the protocol actually realizes a naming for any $N<P$ in order to realize finally a counting for any $N \leq P$.

Sequence U^* used in the protocol is obtained recursively for $P-1$, i.e., $U^* = U_{P-1} = U_{P-2}, P-1, U_{P-2}$, where $U_1 = 1$. However, if the protocol uses $U^* = U_P = U_{P-1}, P, U_{P-1}$, and the mobile agents (which are named by BST following U^*) can accommodate one more state, then this is a naming protocol for any $N \leq P$. This is also reflected by Theorem 15.

This important observation is used here to obtain a naming protocol under weak fairness with $P+1$ states per mobile agent. This is in combination with an additional adaption, for ensuring also self-stabilization. See Protocol 2 below.

Protocol 1 [11] Space-Optimal Counting under Weak Fairness (P states per agent)

Variables at BST (the leader):
n: non-negative integer initialized to 0 // guess of N
k: non-negative integer initialized to 0 // pointer to the k^{th} element of U^*
Shortcuts at BST (the leader):
U^*: constant sequence of elements in $[1,\ldots,P-1]$ computed in advance by the recursion $U_1 \equiv 1, U^* \equiv U_{P-1} \equiv U_{P-2}, P-1, U_{P-2}$
$U^*(k)$: returns the k^{th} element of U^*
$l_n = 2^n - 1$ ($\equiv |U_n|$)

Variable at a mobile agent x:
$name_x$: non-negative integer in $[0,\ldots,P-1]$, initialized *arbitrarily*

1: **when** a mobile agent x interacts with BST (the leader) **do**
2: **if** $n < P \wedge (name_x = 0 \vee name_x > n)$ **then**
3: **if** $name_x = 0$ **then**
4: $k \leftarrow k+1$ // advance k to point to the next element of U^*
5: **else if** $name_x > n$ **then**
6: $k \leftarrow l_n + 1$// because agent x with a name $> n$ could not be seen before by BST, the population must be larger than n, and k is updated accordingly
7: **if** $k > l_n$ **then**
8: $n \leftarrow n+1$ // pointer k indicates that the population is larger
9: $name_x \leftarrow U^*(k)$ // set the name of x to the the k^{th} element of U^*
10: **when** two mobile agents x and y interact **do**
11: **if** $name_x = name_y$ **then**
12: $name_x \leftarrow name_y \leftarrow 0$ // set homonym states to 0

Space-Optimal Naming Protocols Based on Protocol 1

Protocol 2 Space-Optimal Self-stabilizing Naming under Weak Fairness ($P+1$ states per agent)

Variables at BST:
n: $[0,\ldots,P+1]$, initialized *arbitrarily* // guess of N
k: $[0,\ldots,2^P]$, initialized *arbitrarily* // pointer to the k^{th} element of U^*
Shortcuts at BST:
U^*: constant sequence of elements in $[1,\ldots,P]$ computed in advance by the recursion $U_1 \equiv 1, U^* \equiv U_P \equiv U_{P-1}, P, U_{P-1}$
$U^*(k)$: returns the k^{th} element of U^*
$l_n = 2^n - 1$ ($\equiv |U_n|$)

Variable at a mobile agent x:
$name_x$: non-negative integer in $[0,\ldots,P]$, initialized *arbitrarily*

1: **when** a mobile agent x interacts with BST **do**
2: **if** $n \leq P \wedge (name_x = 0 \vee name_x > n)$ **then**
3: **if** $name_x = 0$ **then**
4: $k \leftarrow k+1$ // advance k to point to the next element of U^*
5: **else if** $name_x > n$ **then**
6: $k \leftarrow l_n + 1$ // because agent x with a name $> n$ could not be seen before by BST, with "correct initialization", the population must be larger than n, and k is updated accordingly
7: **if** $k > l_n$ **then**
8: $n \leftarrow n+1$ // pointer k indicates that the population is larger
9: $name_x \leftarrow U^*(k)$ // set the name of x to the the k^{th} element of U^*
10:
11: **else if** $n > P \wedge (name_x = 0)$ **then**
12: $n \leftarrow k \leftarrow 0$ // naming is failed; restart it
13:
14: **when** two mobile agents x and y interact **do**
15: **if** $name_x = name_y$ **then**
16: $name_x \leftarrow name_y \leftarrow 0$ // set homonym states to 0

Protocol 3 [11] Space-Optimal Naming under Global Fairness (P states per agent)

Variables at BST:
n: non-negative integer initialized to 0 // guess of N
k: non-negative integer initialized to 0 // pointer to the k^{th} element of U^*
$name\_ptr$: $[0, \ldots, P]$, initialized to 0
Shortcuts at BST:
U^*: constant sequence of elements in $[1, \ldots, P-1]$ computed in advance by the recursion $U_1 \equiv 1, U^* \equiv U_{P-1} \equiv U_{P-2}, P-1, U_{P-2}$
$U^*(k)$: returns the k^{th} element of U^*
$l_n = 2^n - 1$ ($\equiv |U_n|$)

Variable at a mobile agent x:
$name_x$: non-negative integer in $[0, \ldots, P-1]$, initialized *arbitrarily*

1: **when** a mobile agent x interacts with BST **do**
2: **if** $n < P \wedge (name_x = 0 \vee name_x > n)$ **then**
3: **if** $name_x = 0$ **then**
4: $k \leftarrow k + 1$ // advance k to point to the next element of U^*
5: **else if** $name_x > n$ **then**
6: $k \leftarrow l_n + 1$// because agent x with a name $> n$ could not be seen before by BST, the population must be larger than n, and k is updated accordingly
7: **if** $k > l_n$ **then**
8: $n \leftarrow n + 1$ // pointer k indicates that the population is larger
9: $name_x \leftarrow U^*(k)$ // set the name of x to the the k^{th} element of U^*
10:
11: **if** $n = P \wedge name\_ptr < P$ **then**
12: **if** $name_x = name\_ptr$ **then**
13: $name\_ptr \leftarrow name\_ptr + 1$
14: **else**
15: $name_x \leftarrow name\_ptr$
16: $name\_ptr \leftarrow 0$

17: **when** two mobile agents x and y interact **do**
19: **if** $name_x = name_y$ **then**
20: $name_x \leftarrow name_y \leftarrow 0$ // set homonym states to 0