



HAL
open science

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-01790517v3

HAL Id: hal-01790517

<https://inria.hal.science/hal-01790517v3>

Submitted on 4 Aug 2019

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 Burman¹

LRI, Université Paris Sud, CNRS, Université Paris-Saclay, France

janna.burman@lri.fr

Joffroy Beauquier

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@uvsq.fr

Abstract

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

The considered distributed communication 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 of the necessary and sufficient state space conditions for naming. The problem 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 Theory of computation → Distributed algorithms

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

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

The population protocol model was introduced as a minimalist model for mobile sensor networks where the computational devices, called agents, communicate by pair-wise interactions without (almost) any control on their communication schedule [3].² The model originally assumed that the memory of agents is constant, i.e., independent of the population size n . Due to this, agents are prohibited from storing unique identifiers.

Many limitations have been discovered due to this restriction, and a lot of work has been devoted to study the power added by relaxing it. For example, it was obtained in [15] that a non-semi-linear predicate can be computed starting from $\Theta(\log \log n)$ bits of agent's memory (allowing $\Theta(\log^{O(1)} n)$ identifiers), while the original population protocols compute only exactly the semi-linear predicates [4]. Furthermore, it was shown that using $\Theta(\log n)$ bits per agent, allowing to assign unique identifiers (names) to agents, already permits

¹ corresponding author

² Basically, at each step a pair of agents is scheduled to interact (subject to a fairness condition) and each agent observes the other's state updating its own according to the transition function.



42 to compute exactly all the symmetric predicates in the class $NSPACE(n \log n)$ (what is
 43 equivalent to the power of $O(n \log n)$ space Turing machine) [28, 15]. Following these results,
 44 a comprehensive study in [12] provided a complete hierarchy establishing complexity classes
 45 for the cases going from non-identified agents (the original population protocol model) to
 46 uniquely identified ones, passing by the case of homonyms.

47 Another line of works where the possibility of having names plays an important role
 48 concerns fault-tolerant population protocols. As a motivating scenario one can think of
 49 reliability critical mobile sensor networks (not necessary of a very large scale), which may be
 50 hardly accessible and have to recover automatically to the correct behavior after faults. In
 51 the framework of *self-stabilization* [20, 5], i.e. when the reliability concerns transient faults
 52 (e.g., memory or communication errors), it was shown that a linear (on n) state space is
 53 necessary for the realization of many tasks. Interestingly, in these cases, many proposed
 54 solutions perform a sort of naming mechanism. This concerns for example the self-stabilizing
 55 tasks of leader election [13], counting [7, 29, 6] and deterministic oscillation [17]. In [19], for
 56 computing semi-linear predicates while tolerating a known constant number of transient and
 57 crash faults, the algorithm approximately divides the population into a predefined number of
 58 groups, actually creating named homonyms. Finally, some other fault-tolerant population
 59 protocols assume that names are given a priori, e.g., [28, 37].

60 These observations suggest that the task of naming in population protocols should receive
 61 a particular attention. This work presents a comprehensive study of this problem. It
 62 focuses on the necessary and sufficient state space conditions for naming under all possible
 63 combinations of a set of classical model assumptions, like existence of a leader, *weak* or *global*
 64 fairness, uniform or arbitrary initialization and symmetry of transition rules. Note that a
 65 choice of a combination affects the type and the level of difficulty of breaking symmetry or of
 66 achieving fault-tolerance. These parameters, their interest and effect are all discussed below.

67 The first parameter is the nature of the assumed fairness, weak or global. The formal
 68 definitions and an example illustrating the difference between the two appear in the model
 69 section. Intuitively, while global fairness ensures that an infinitely often reachable configura-
 70 tion is unavoidable, weak fairness only ensures that every pair of agents interacts infinitely
 71 often. Global fairness can be viewed as a way for modeling randomized systems (without
 72 introducing randomization explicitly in the model). This explains why it is generally easier
 73 to get solutions under this fairness. However, such randomization cannot be always assumed
 74 to be available, in particular in reliability critical systems.

75 A second parameter is the symmetry nature of the (transition) rules of a protocol. With
 76 *symmetric rules*, two interacting agents in the same state stay in identical states. With
 77 *asymmetric rules*, they can take different states. This latter assumption was the original one
 78 proposed for population protocols and motivated by asymmetric wireless communications.
 79 The symmetric rules assumption is more general (weaker), and many motivating scenarios
 80 can be found for it in nature inspired population protocols, social networks (when equity is
 81 an issue) or in networks with symmetric wireless communication.³

82 A third parameter is the presence or absence of a unique leader (a distinguishable agent),
 83 which is obviously also useful for the sake of breaking symmetry. In the context of sensor
 84 networks, this agent may represent a (possibly mobile) base station, having augmented

³ Note that an asymmetric population protocol can be transformed into a symmetric one using the transformer of [11]. 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 inadequate under weak fairness.

85 resources comparing to the tiny mobile agents. From this perspective and similarly to
 86 previous studies (e.g., [40, 6, 29]), we are not concerned with the space complexity of this
 87 agent.

88 A fourth parameter is related to the initialization of system agents, i.e., of the leader (if
 89 present) and of the other agents (called here *mobile*). In case of mobile agents, if initialization
 90 is assumed, it is always *uniform*, i.e., to the same value for every mobile agent. In case of
 91 arbitrary initialization, the given solution stabilizes starting from an *arbitrary* configuration
 92 (i.e., resulting from any number of transient faults). In this case it is called self-stabilizing [20].
 93 The weaker initialization assumption is (e.g., only the leader is initialized, or nobody), the
 94 stronger the system is against transient faults and the more adapted to repetitive execution
 95 of a task (requiring less or no re-initialization).

96 Finally, the focus of this work is on deterministic protocol design, useful whenever random
 97 behavior is inappropriate or deterministic guarantees are required. Moreover, as many
 98 previous works [40, 17, 6, 29, 7], we perform an *exact* state space analysis. On the negative
 99 side, this requires especially careful analysis and design, in contrast to, e.g., the asymptotic
 100 analysis case. On the positive side, the exact analysis is extremely relevant in the cases
 101 of particularly memory-limited devices, small sized networks and self-stabilizing protocols.
 102 The less volatile memory is used by a self-stabilizing protocol, the less (probabilistically less
 103 frequently) it is vulnerable to corruptions.

104 Contribution

105 Thus, we investigate the naming problem under all the possible combinations of the parameters
 106 described above. All the negative results (impossibilities and lower bounds) are presented
 107 in Section 3, while all the positive ones (state-optimal solutions) are given in Section 4.
 108 Table 1 gives a synthetic view of these results. For each case, it indicates first the statement
 109 establishing the feasibility, either by proving impossibility or by construction of a space-
 110 optimal protocol. In the latter case, the table also indicates the optimal (used) number of
 111 states and the relevant statement of the lower-bound.

112 The results are expressed in terms of P - a known upper bound on the number of
 113 mobile agents n (i.e., the maximum number of agents destined to be named). This technical
 114 assumption is done for dealing with bounded protocols (similarly to the related studies,
 115 e.g., [29, 7]). P can be seen as a function of the manufactured memory size in each
 116 (undistinguishable) mobile agent.⁴ Nevertheless, the results (and the lower bounds in
 117 particular) can be equivalently expressed in terms of n , when considering a particular
 118 population size.

119 Notice that, first of all, the table concerns the case of *arbitrarily initialized mobile agents*.
 120 Together with that, it is also relevant to the case of *uniform initialization of mobile agents*. It
 121 is obvious for the positive results (the protocols stay correct). However, somewhat surprisingly,
 122 the impossibilities and lower bounds also hold under this stronger assumption, but with only
 123 one exception. The exception concerns symmetric rules under weak fairness and with an
 124 initialized leader, i.e., when a leader is present and can be initialized together with the other
 125 agents (refer to the table). Then, there is a simple naming protocol with only P states per
 126 mobile agent (Proposition 14), instead of the necessary $P + 1$ states with arbitrary initialized
 127 mobile agents. In contrast with this simple protocol for completely initialized case, the

⁴ The parameter P is used in the protocols, but this explicit usage can be frequently replaced by an alert mechanism announcing that the bound will be shortly reached.

XX:4 Space-Optimal Naming in Population Protocols

128 analysis of tight negative and positive results assuming no initialization of mobile agents is
 129 much more complex (see Section 3.1 and Propositions 16 and 17).

130 Additional remarks can be made about the table. First, under weak fairness and without
 131 leader, no symmetric deterministic protocol is capable of breaking symmetry, and thus naming
 132 is impossible. Second, if asymmetric rules are allowed, in all cases, there is a space-optimal
 133 solution with P states. On the contrary, with symmetric rules, the norm is $P + 1$ states,
 134 excluding two cases: when there is an initialized leader (in both cases) and (1) either global
 135 fairness or (2) uniform initialization of mobile agents is assumed. In both cases, the problem
 136 can be solved with P states per agent.

	symmetric rules		asymmetric rules
	weak fairness	global fairness	weak/global fairness
no leader	impossible (Prop. 1)	Prop. 13, with $P + 1$ states (Prop. 3)	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	<i>non-initialized agents:</i> Prop. 16, with $P + 1$ states (Theorem 11); <i>initialized agents:</i> Prop. 14, with P states	Prop. 17, with P states	Prop. 12, with P states

■ **Table 1** Synthesis of the relevant statements 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 and uniformly initialized mobile agents.

137 **Note:** Several minor proofs and the additional related work discussion have been moved to
 138 the appendix. Though, the most relevant work is mentioned above.

139 2 Model and Notations

140 We adopt the model of population protocols [3] as a basic model. A system consists of a
 141 collection \mathcal{A} of pairwise interacting agents, also called *population*. Each agent can represent
 142 a sensing and communicating mobile device. Among the agents, there can be a unique
 143 distinguishable agent called the *leader* which can be as powerful as needed, in contrast
 144 with the resource-limited non-leader agents. The non-leader agents are also called *mobile*,
 145 interchangeably. The size of the population n is the number of the mobile agents. It is
 146 unknown (a priori) to the agents, contrary to the upper bound parameter $P \geq n$. Each agent
 147 has a state taken from a set of states Q (depending on P), the same for all mobile agents,
 148 but generally different for the leader.

149 A (*population*) *protocol* can be modeled as a finite transition system defined by transitions
 150 between *configurations*, where a configuration is a vector of states of all the agents. The
 151 transitions between configurations are defined by pairwise interactions between agents. If, in
 152 a configuration C , two agents x and y interact (meet), s.t. x is in state p and y in state q ,
 153 they execute a *transition rule* $(p, q) \rightarrow (p', q')$. As a result, x changes its state from p to p'
 154 and y from q to q' . The new configuration C' , resulting from this state changes, is said to be
 155 reachable from C (in one step), denoted by $C \rightarrow C'$. If $p = p'$ and $q = q'$, the corresponding

156 transition is said *null* (such transition rules are specified by default), and non-null otherwise.⁵
 157 If there is a sequence of configurations $C = C_0, C_1, \dots, C_k = C'$, such that $C_i \rightarrow C_{i+1}$ for all
 158 $i, 0 \leq i < k$, we say that C' is *reachable* from C , denoted $C \xrightarrow{*} C'$.

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

165 Let $(p_1, q_1) \rightarrow (p_2, q_2), (p_2, q_2) \rightarrow (p_3, q_3), \dots, (p_{k-2}, q_{k-2}) \rightarrow (p_{k-1}, q_{k-1}), (p_{k-1}, q_{k-1}) \rightarrow$
 166 (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
 167 denote the successive application of the rules of the sequence, to the same pair of agents,
 168 initially in states p_1 and q_1 , leading them to p_k and q_k , respectively. We sometimes call an
 169 agent in state p a p -state agent, or just a p -agent.

170 An *execution* of a protocol is an infinite sequence of configurations and transitions
 171 $C_0, t_1, C_1, t_2, C_2, t_3, \dots$ such that C_0 is the starting configuration and for each $i \geq 0$,
 172 $C_i \rightarrow C_{i+1}$, t_i being the transition between two particular agents used to reach C_{i+1} from
 173 C_i . When the actual transitions are irrelevant, we denote the execution by C_0, C_1, C_2, \dots
 174 A *segment* or a *sub-execution* is a sub-sequence of an execution. The *trace of transitions*
 175 of a sub-execution $C_0, t_1, C_1, t_2, C_2, \dots, t_k, C_k$ is a sequence $t_1, t_2, t_3, \dots, t_k$ of transitions, and
 176 the corresponding *trace of transition rules* is the sequence $r_1, r_2, r_3, \dots, r_k$ such that r_i is the
 177 transition rule applied in t_i . In a real distributed execution, interactions of distinct agents
 178 are independent and could take place simultaneously (in parallel), but when writing down
 179 an execution we order those simultaneous interactions arbitrarily.

180 An execution is said *weakly fair*, if every pair of agents in \mathcal{A} interacts infinitely often. An
 181 execution is said *globally fair*, if for every two configurations C and C' such that $C \rightarrow C'$, if
 182 C occurs infinitely often in the execution, then C' also occurs infinitely often in the execution.
 183 This also implies that, if in an execution there is an infinitely often reachable configuration,
 184 then it is infinitely often reached [4]. Note that an execution where pairs of agents interact
 185 according to some probabilistic distribution is globally fair with probability 1 [30].

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

196 A *(static) problem* is defined by a predicate \mathcal{D} on configurations. A population protocol
 197 \mathcal{P} is said to *solve a problem* \mathcal{D} , if and only if every execution of \mathcal{P} reaches a configuration
 198 satisfying the conditions defining \mathcal{D} and stays in such configurations forever after. When this
 199 happens, we say that the protocol has *stabilized* (or *terminated*), and a *terminal configuration*
 200 has been reached. A *self-stabilizing protocol* is a protocol that stabilizes from an arbitrary

⁵ 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.

201 configuration (i.e., from a configuration where *any* agent, *including the leader*, can be in any
202 possible state).

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

206 We consider *uniform* or *semi-uniform* protocols (cf. [21, 38]) in the sense that all agents,
207 except the leader (whence semi-), are a priori indistinguishable and interact according to
208 the same transition rules. Moreover, given an upper bound P on n , the protocol functions
209 similarly for any n . Thus, given the bound P , by the definition of naming, an obvious lower
210 bound on the state space of a mobile agent (for solving naming) is P .

211 3 Negative Results

212 In this section, we clarify some boundaries on the possibility to obtain symmetric naming.
213 They are summarized in Table 1 and useful for establishing the space-optimality (tightness)
214 of the solutions presented in the next section. We proceed from simple to more intricate
215 results, gradually adding assumptions helping in breaking symmetry (making harder to prove
216 impossibilities). We conclude this section by Theorem 11 - a subtle result stating impossibility
217 to get a P state symmetric protocol even with an initialized leader, but non-initialized mobile
218 agents and under weak fairness.

219 The first result is obtained by observing a completely symmetric weakly fair execution
220 where at each step the population transits from one uniform configuration (all agents are in
221 the same state) to the other, by applying each time a symmetric rule between two homonyms.

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

224 **Proof.** By contradiction, assume that such a symmetric protocol \mathcal{PP} exists. With or without
225 an initialization, consider a possible starting configuration where each agent is in state s_1
226 and the population size is even (this also corresponds to a uniform initialization). We build
227 a weakly fair infinite execution of \mathcal{PP} during which no configuration with agents in distinct
228 states is reached. This execution can be described by phases. In the first phase, the agents
229 are matched in pairs and interact accordingly. As the protocol is symmetric, after this first
230 phase, each agent is in some state s_2 , the same for all agents. In the next phase, the agents
231 are matched again in pairs, but differently from the previous phase, and interact accordingly.
232 After this phase, each agent is in some state s_3 , the same for all agents. The execution
233 continues in such phases such that eventually every agent has interacted with every other.
234 From now on, such interaction sequence is repeated infinitely often, satisfying weak fairness.
235 However, this execution never assigns distinct names to agents. ◀

236 The following lemma states properties of the transition rules of any P state symmetric
237 naming protocol. Its short proof is given below. The lemma is used in the proofs of the
238 next two propositions: the first one assumes no existing leader, while the second one proves
239 impossibility of a self-stabilizing symmetric naming even with a leader.

240 ► **Lemma 2.** *In any symmetric naming protocol using only P states per agent, the only
241 possible non-null transition rules between two non-leader agents are between two homonyms.*

242 **Proof.** Otherwise, in a population of P agents, after stabilization (every state in $\{1, \dots, P\}$
243 is assigned to some agent), mobile agents would be able to change their states and hence
244 their names. This is a contradiction to the assumed stabilization. ◀

245 Under the conditions of the next proposition and by the lemma above, the only non-null
 246 transitions are between two homonyms (resulting in another two homonyms). Such transitions
 247 are useless for eliminating repeated names, implying the result.

248 ► **Proposition 3.** *Even with a uniform initialization of agents, but without a leader, and*
 249 *using only P states per agent, it is impossible to obtain symmetric naming in the population*
 250 *protocol model, under weak or global fairness.*

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

253 **Proof.** Assume by contradiction that such a protocol exists. By Proposition 3, a leader is
 254 necessary. Consider a population of P mobile agents and a starting configuration C_0 (with
 255 possibly uniformly initialized mobile agents) that has homonyms (with states in $\{1, \dots, P\}$).
 256 By Lemma 2, only transitions with a leader can eliminate homonyms (the only possible non-
 257 null symmetric transition rules between homonyms create necessarily two other homonyms).
 258 Thus, there is a finite execution sequence e , starting from C_0 , during which the leader renames
 259 interacting mobile agents, and finally stabilizes to a correct naming (where every name from
 260 $\{1, \dots, P\}$ is assigned to some mobile agent). Denote by C_e and by s_e the configuration and
 261 the state of the leader, respectively, at the end of e .

262 Then, assume a starting configuration C'_0 (with possibly uniformly initialized mobile
 263 agents, as before) containing only homonyms in state s (in $\{1, \dots, P\}$) and with the leader
 264 in state s_e . There must be a sequence of interactions between the leader and mobile agents
 265 starting from C'_0 which ends up with a transition during which an interacting agent changes
 266 its name. Such a sequence of interactions exists also starting from C_e (with either weak or
 267 global fairness), because in C_e any possible state exists in the population. This contradicts
 268 the assumption that the protocol has stabilized starting from C_e in e . Hence, the proposition
 269 follows. ◀

270 3.1 Impossibility of P state symmetric naming of arbitrarily initialized 271 mobile agents, under weak fairness, even with an initialized leader

272 For proving this stronger impossibility result, let us assume, for the sake of contradiction,
 273 that such a solution exists. Thus, denote by $Name$ any symmetric protocol solving the
 274 naming problem under weak fairness (for any $n \leq P$), with arbitrarily initialized P -state
 275 mobile agents. By Proposition 3, under such conditions, a leader is necessary. Moreover, by
 276 Proposition 4, such an agent has to be initialized. So, in this sub-section, we assume such
 277 initialized unique leader. In the following, some additional necessary properties of protocol
 278 $Name$ are proven and finally imply the impossibility of its existence (see Theorem 11).

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

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

287 **Proof.** Let us assume, by contradiction, that there are infinitely many configurations in e
 288 with a mobile agent in state m . Since there is a finite number of agents, there is a particular

289 mobile agent x in \mathcal{A} which is in state m in infinitely many configurations. Let $C_{j_1}, C_{j_2}, C_{j_3}, \dots$
 290 be these configurations such that $e = e_1, C_{j_1}, e_2, C_{j_2}, e_3, C_{j_3}, \dots$. W.l.o.g., we choose these
 291 configurations such that, in every execution segment e_i , every agent in \mathcal{A} interacts with every
 292 other (this is possible with weak fairness).

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

297 We construct e' based on e . First, we assume that in e' , x' is in state m in the
 298 starting configuration, and $e' = e'_1, C'_{j_1}, e^m, e'_2, C'_{j_2}, e^m, e'_3, C'_{j_3}, e^m, \dots$. Every segment e'_i
 299 follows exactly the same transition sequence as in e_i . In every segment $e'_{2r+1}, C'_{j_{2r+1}}$ (for
 300 $r \geq 0$) the interactions are exactly the same as in $e_{2r+1}, C_{j_{2r+1}}$, and x' does not interact.
 301 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
 302 this case, x is replaced by x' in the appropriate state, and x does not interact. Finally, e^m is
 303 an execution segment where only x and x' interact. They both start in state m , performing
 304 the sequence $(m, m) \xrightarrow{*} (m, m)$. The configurations at the beginning and at the end of e^m
 305 are identical. The construction of e' ensures that in every C'_{j_i} , both x and x' are in the state
 306 m .

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

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

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

316 **Proof.** As *Name* is symmetric, two interacting agents, both in some state $s \in Q$, execute
 317 some symmetric transition of the form $(s, s) \rightarrow (s_1, s_1)$. If they meet several times successively,
 318 there is a possible sequence of transitions $(s, s) \rightarrow (s_1, s_1) \rightarrow (s_2, s_2) \rightarrow (s_3, s_3) \dots$. As mobile
 319 agents are finite state, for some $j > i \geq 1$, $s_i = s_j$, i.e. $(s_i, s_i) \xrightarrow{*} (s_i, s_i)$. By Lem. 5, $s_i = m$
 320 s.t. m does not appear infinitely often in executions with $n < P$. As there are at least $P - 1$
 321 states appearing infinitely often in an execution with $n = P - 1$, there is at most one such
 322 possible state m in a P state protocol. This implies correctness of conditions (2) and (3) of
 323 the sink definition.

324 Finally, by contradiction, if $(m, m) \rightarrow (s, s)$ s.t. $s \neq m$, then the previous part of the proof
 325 implies $(s, s) \xrightarrow{*} (s, s)$. As m is proved (above) to be unique, this is a contradiction, implying
 326 the correctness of condition (1) of the sink definition. ◀

327 From now on, the proof assumes the sink state denoted by m , and shows the impossibility
 328 for a particular kind of executions, called here *reduced*. In any segment of a reduced
 329 execution, each time a pair of $s \neq m$ homonyms appears, it is immediately *reduced* to m ,
 330 i.e., by applying the sequence of transition rules $(s, s) \xrightarrow{*} (m, m)$, whose transitions and by
 331 extension the sequence itself are called (*homonym*) *reducing*. Thus, other transitions can
 332 take place only when there are no more homonyms. Naturally, any configuration without
 333 any homonyms except those in state m is called *reduced*. Notice that, in a reduced execution,
 334 there are non-reduced configurations, but only during the reducing transition sequences. For
 335 example, consider the following reduced sub-execution (where l_i represents a leader state):

336 $[1, 2, 3, 4, m, l_1], (l_1, 1) \rightarrow (l_2, 2), [2, 2, 3, 4, m, l_2], (2, 2) \rightarrow (m, m), [m, m, 3, 4, m, l_2], (l_2, m) \rightarrow$
 337 $(l_3, 3), [m, 3, 3, 4, m, l_3], (3, 3) \rightarrow (m, m), [m, m, m, 4, m, l_3]$. In this example, the reduced
 338 configurations are $[1, 2, 3, 4, m, l_1]$,
 339 $[m, m, 3, 4, m, l_2]$ and $[m, m, m, 4, m, l_3]$.

340 Forcing a reducing sequence of transitions whenever possible, as in a reduced (sub-)
 341 execution, does not prevent an execution from being weakly fair. Thus, we can prove the
 342 following corollary.

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

345 **Proof.** First, recall that, by Prop. 6, and Lemma 2, the only non-null-transitions of $Name$
 346 are the homonym reducing transitions between non- m -mobile agents and the transitions with
 347 the leader. Given a reduced segment, we prolong the execution by forcing mobile agents
 348 to interact with every other agent (including the leader) in a “round-robin fashion” (not
 349 necessarily in consecutive interactions). Whenever homonyms are created, this interaction
 350 pattern is interrupted by the homonym reducing sequence of transitions, and then resumed
 351 after the reduction. The execution constructed that way is weakly fair and uses only
 352 transitions of $Name$, hence it stabilizes towards a naming. ◀

353 The following lemma states a basic property of $Name$, in a population of P agents. It uses
 354 the notion of equivalent configurations. Two configurations are *equivalent* if both correspond
 355 to the same multi-set of states⁶ (e.g., $C_1 = [2, 3, 2, m, l]$ is equivalent to $C_2 = [2, 2, 3, m, l]$,
 356 where l stands for the leader state). This notion is naturally extended to equivalent executions.
 357 The lemma shows that, in a population of P agents, if there is a reduced sub-execution in
 358 which some particular state $s \neq m$ never appears (in its reduced configurations), then there
 359 is also an *equivalent* sub-execution where a particular m -agent never interacts.

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

366 **Proof.** In a population of P agents, in any reduced configuration (of a reduced execution of
 367 $Name$), there is at least one m -state agent. Thus in any reduced configuration of e there are
 368 at least two m -state agents. Consider C and call one of these two agents agent x .

369 Let us construct now e' , starting in C , with exactly the same trace of transition rules of e .
 370 By Prop. 6, and Lemma 2, the only non-null-transitions are homonym reducing transitions
 371 between non- m -mobile agents and the transitions with the leader. By a simple induction
 372 below, one can see that every transition rule in a trace of e , step by step, can be executed
 373 without participation of agent x , and thus added to e' . Since no transition of e creates an
 374 s -agent, no such added transition can create an s -agent.

375 Thus, starting in C , the first transition of e can be executed with any agent, except
 376 x , and thus can be added to e (the base of induction). The resulting configuration C'_1 is
 377 equivalent to C_1 . Then, by induction, assume that after k transitions added to e' , the reached

⁶ 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)

XX:10 Space-Optimal Naming in Population Protocols

378 configuration is equivalent to the one reached after transition k in e (without any s -state
 379 agent in a reduced configuration). For $k + 1$, if a homonym reducing transition takes place in
 380 e , x does not participate (it is already reduced) and thus the same transition can be added to
 381 e' , and an equivalent configuration is reached. Otherwise, if this is a reduced configuration,
 382 there is an additional m -state agent, different from x , so any transition with the leader can
 383 be executed excluding x , and an equivalent configuration is reached. By such construction of
 384 e' , every configuration C'_i is equivalent to C_i in e . ◀

385 Now, we want to prove that *Name*, in a population of size P , can reach a terminal
 386 configuration C (with uniquely named agents and in particular with an agent x in some state
 387 $s \neq m$), but such that the leader may be unaware of that. Then, the leader should manage
 388 to create the possibly missing s -agent for stabilizing towards a configuration where naming
 389 is realized. But, once created, this s -agent can disappear by a reduction with the “hidden”
 390 s -agent x . This contradicts the fact that the reached configuration C is terminal (see the
 391 proof of Theorem 11). This proof uses the following two technical lemmas.

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

398 A configuration C_1 is said to be *far away* from C_2 by one state $s \neq m$ (in agent x),
 399 if there is an agent x such that $C_1[x] = m$, $C_2[x] = s \neq m$ and $\forall y \in \mathcal{A} \setminus \{x\}, C_1[y] =$
 400 $C_2[y] \neq s$. Then, C_1 is denoted by C_2^{-s} and C_2 by C_1^{+s} . Agent x is called the *pivot*. For
 401 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
 402 $C_1 = C_2^{-2}, C_2 = C_1^{+2}$.

403 ► **Lemma 9.** *Consider a population of size P and two reduced configurations C_1 and C_1^{-s} ,
 404 far away by state s , in agent x . Consider a given reduced sub-execution $C_1^{-s}e_1^{-s}C_2$ of *Name*
 405 where: (i) there is no s -agent in every reduced configuration in the segment $C_1^{-s}e_1^{-s}$, (ii)
 406 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,
 407 there exists a reduced sub-execution $C_1e_1C_2^{-s}$ of *Name* such that exactly one agent in state s
 408 exists in every reduced configuration in C_1e_1 , and agent x does not interact in $C_1e_1C_2^{-s}$,
 409 except in the very last (s -homonym) reducing sequence.*

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

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

424 **Proof.** In a given execution $C_1e_1C_2^{-s}$ agent x does not interact, except in the very last
 425 (s -homonym) reducing sequence. Starting from C_1^{-s} we construct an execution $C_1^{-s}e_1^{-s}C_2$,
 426 by using first exactly the same prefix of the trace of transition rules of $C_1e_1C_2^{-s}$, until and
 427 excluding the very last (s -homonym) reducing sequence. In the given configuration, before
 428 this very last reducing sequence, two s -agents necessarily exist, one of them being till now the
 429 non-interacting agent x . However, in the sub-execution constructed at this point, x does not
 430 interact either, but is in state m , so exactly one s -agent exists at the end of the constructed
 431 sub-execution. Hence, C_2 is reached and the required $C_1^{-s}e_1^{-s}C_2$ is obtained. ◀

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

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

440 Assume that the actual obtained configuration is C_1^{-s} . By the correctness of *Name*,
 441 starting from C_1^{-s} , any execution e stabilizes to a configuration C_* where all agents are in
 442 different states and no state changes thereafter. Moreover, by Corollary 7, there is such an
 443 execution, which is reduced. Furthermore, any such e can be decomposed s.t.

444 $e = C_1^{-s}e_1^{-s}C_2 e_2 C_3^{-s}e_3^{-s}C_4 e_4 C_5^{-s} \dots C_k^{-s}e_k^{-s}C_*C_* \dots$. For every odd i , no s -agent exists
 445 in any reduced configuration of $C_i^{-s}e_i^{-s}$, and C_i^{-s} is reduced. For every even i , exactly one
 446 s -state agent exists in every reduced configuration in $C_i e_i$, and C_i is reduced. By Lemma 8,
 447 for every odd i , one can choose a segment $C_i^{-s}e_i^{-s}$ such that a particular m -agent x_i never
 448 interacts in this segment.

449 Furthermore, e is chosen such that, for every segment $C_i e_i C_{i+1}^{-s}$ with an even i , a particular
 450 s -state agent $y_i \neq x_i$ does not interact, except in the very last (s -homonym) reducing sequence.
 451 Let us show (by induction) that such e exists. First, since in $C_i^{-s}e_i^{-s}$, for odd i , there is
 452 an m -agent $y \neq x_i$ in every reduced configuration. Thus, in the following C_{i+1} (even $i + 1$)
 453 configuration, any such agent or other non- m -agent $y_i \neq x_i$ can become an s -agent. This
 454 implies that every agent in e has the opportunity to interact repeatedly. Second, by Lemma 9,
 455 $C_1e_1C_2^{-s}$ exists (such that there is exactly one agent in state s in every reduced configuration
 456 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
 457 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
 458 Lemma 8, such that, in $C_2^{-s}e_2^{-s}$, a particular m -state agent does not interact, e.g., the pivot
 459 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
 460 s -state agent y_2 does not interact, except in the very last (s -homonym) reducing sequence
 461 (this proves the base of induction).

462 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
 463 odd i , in e , and show (by induction) that the chosen e exists. Recall that we assumed that
 464 *Name* has stabilized in C_* and then, the leader has to stop renaming the agents. In addition,
 465 C_* is reduced (all agents are distinctly named). Hence, from this point, only null-transitions
 466 are possible.

467 Notice that the conditions of Lemma 9 are satisfied for the segments $C_i^{-s}e_i^{-s}C_{i+1}$ with
 468 an odd i , and the conditions of Lemma 10 are satisfied for the segments $C_i e_i C_{i+1}^{-s}e_i^{-s}$ with
 469 an even i . Hence, by applying these lemmas repeatedly to the segments of e , one inductively
 470 builds the execution segment $e' = C_1e_1C_2^{-s}e_2^{-s}C_3e_3 \dots C_k e_k C_*^{-s}$. However, C_*^{-s} is reduced,

XX:12 Space-Optimal Naming in Population Protocols

471 and far away by only one state from C_* . No agent except the pivot, can distinguish
472 C_*^{-s} from C_* , since each one is in the same state in both configurations. In particular,
473 $C_*^{-s}[\text{leader}] = C_*[\text{leader}]$. Thus, and by Lemma 2 and Prop. 6, the only possible transitions
474 with the leader are null transitions, as well as the transitions involving mobile agents (there
475 are no non- m -homonyms). Obviously, in C_*^{-s} , the protocol has not stabilized yet. But, no
476 transition can change the configuration C_*^{-s} . This contradicts the assumption that $Name$ is
477 correct. ◀

478 **4 Positive Results**

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

483 The proof is by construction of a protocol with a single asymmetric type of rule,
484 $(s, s) \rightarrow (s, (s + 1) \bmod P)$. This idea is known in the literature, e.g., [13, 7]. It aimed at
485 solving other (than naming) problems, but provided naming as a by-product. [13] considers
486 self-stabilizing leader election, assuming that the exact size of the population n is known
487 (an assumption proven to be necessary). Under this assumption, the presented protocol also
488 solves naming. In [7], a similar idea is used to count the arbitrarily initialized mobile agents,
489 assuming an initialized leader, and realizes also naming.

490 The asymmetric space-optimal naming protocol presented below is proven under more
491 general assumptions (with upper bound P , instead of exact knowledge of n , without a leader,
492 and under both fairness assumptions). Its proof uses the novel technique of *hole* and *hole*
493 *distance* in a configuration.

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

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

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

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

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

513 Now we consider one of the most difficult cases for symmetric protocols - assuming no
514 leader and no initialization - impossible under weak fairness. Recall that global fairness
515 mimics, in some sense, the behavior of randomized environments. The following proposition

516 shows that this pseudo randomization is sufficient for breaking symmetry, and that a
 517 distinguishable agent is not needed for that. Thus, we propose below the first *symmetric*
 518 *space-optimal self-stabilizing* naming obtained without a leader (the complete proof is in the
 519 appendix). Notice, that by Proposition 3, at least $P + 1$ states per agent have to be used in
 520 this case.

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

524 **Proof Sketch.** Consider the following symmetric protocol with state space $Q = \{0, 1, \dots, P\}$
 525 and defined by three types of transition rules:

526 **1.** *if $s \neq P : (s, P) \rightarrow (s, (s+1) \bmod P)$; **2.** *if $s \neq P : (s, s) \rightarrow (P, P)$; **3.** $(P, P) \rightarrow (1, 1)$.**

527 From a configuration with homonyms, rule 2 can be applied repeatedly to obtain a
 528 configuration C' where there are only P -state homonyms, and possibly some other uniquely
 529 named agents. If, in C' , no uniquely named agents exist, let us force transitions using rules
 530 3 and then 1, to create at least one uniquely named agent. Then, rule 2 is applied again, to
 531 reach a configuration C with only P -state homonyms and with at least one uniquely named
 532 agent. Then, whenever there are still some P -state homonyms in C , pick an agent with a
 533 unique name s such that no agent with a unique name $(s + 1) \bmod P$ exists. Make the
 534 s -agent interact with some P -agent, applying rule 1. The obtained configuration contains
 535 one more unique name than in C . If naming is not yet reached, this scenario is repeated until
 536 it is reached. Such an execution segment is possible from any configuration with homonyms.
 537 Hence, naming is reached in any globally fair execution. ◀

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

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

548 The next two results (Prop. 16 and 17) exploit the existing space-optimal *counting*
 549 protocol from [6], which uses P states per mobile agent, and (deterministically and exactly)
 550 counts such non-initialized agents under weak fairness, assuming an initialized leader. Let
 551 us denote it by $Count_P$. It was not originally intended to be a naming protocol, neither a
 552 self-stabilizing one. However, it can be observed that, for the case of $n < P$, it performs (a
 553 non self-stabilizing) naming. This property is reflected in Theorem 15 below.

554 ► **Theorem 15** ([6]). *Protocol $Count_P$ in [6] solves the counting problem, under weak fairness,*
 555 *for up to P mobile agents, each with P states. Moreover, for any $n < P$, the protocol names*
 556 *(up to $P - 1$) mobile agents with distinct names in $\{1, \dots, n\}$.*

557 By augmenting the mobile agents' state space to $P + 1$ and adapting $Count_P$ accordingly,
 558 one obtains a naming protocol (also correct in the case where $n = P$), though using a
 559 non-optimal $P + 1$ number of states. Let us denote the resulting protocol by $Count_{P+1}$.
 560 This protocol is adapted here further for solving the naming problem in a self-stabilizing way
 561 - Prop. 16, while the protocol of Prop. 17 is based on the original $Count_P$.

XX:14 Space-Optimal Naming in Population Protocols

562 For presenting these protocols, some more details of *Count* have to be given. First,
563 note that, in the new protocols here, an appropriate *Count* protocol is a priori executed
564 independently, by every agent. The leader (also assumed in *Count*) manages an estimate
565 for the population size. Let us denote it here by $Count.N$ (or just N when the particular
566 version of *Count* is clear from the context). In the original version of *Count*, N is initialized
567 to 0 and incremented until reaching the actual size n (N is non-decreasing). Each mobile
568 agent x in *Count* has an arbitrary initialized variable $name_x$, which eventually contains a
569 unique name (in $\{1, \dots, n\}$), with $Count_{P+1}$ (for any $n \leq P$), or with $Count_P$ for any $n < P$.
570 Only the leader can assign a new (non 0) name to an interacting agent in state 0. This
571 state plays the role of the sink state (see definitions in Sect. 3.1). The only action of mobile
572 agents is to reduce homonym states to the sink. Because of that, 0-agents appear along an
573 execution until naming with names in $\{1, \dots, n\}$ is reached. Notice that, even though naming
574 may not be reached in $Count_P$ (0-agents may persist forever), it terminates, i.e., agents will
575 eventually execute only null-transitions. This happens whenever a reduced configuration
576 satisfying $Count_P.N = n$ is reached (this is used in Protocol 1, Prop. 17).

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

580 **Proof.** By Theorem 15, for any $n \leq P$, $Count_{P+1}$ assigns unique names in $\{1, \dots, n\}$ to
581 mobile agents, if the leader is well initialized. However, to get a self-stabilizing version, one
582 have to abandon this latter assumption (the leader cannot be initialized). Then, it may
583 happen that $Count_{P+1}.N$ starts in a non 0 value and reaches $P + 1$, before the naming (and
584 the correct count) is realized. Notice that in this case, mobile agents in state 0 exist (since a
585 naming is not yet reached).

586 To overcome this case, we incorporate a reset technique. When a mobile agent x in state
587 0 ($name_x = 0$) interacts with the leader and the estimate $Count_{P+1}.N$ is bigger than P ,
588 the leader resets its internal variables (together with N) to the initialization values of the
589 original $Count_{P+1}$. It is clear that, whenever such a reset is executed, the required naming
590 is eventually and correctly obtained, by the correctness of $Count_{P+1}$. This solves the only
591 problematic case described above. Hence, the proposition follows. ◀

592 Now, for proving Prop. 17, a protocol using only P states per mobile (non-initialized)
593 agent is constructed. It is done by modifying protocol $Count_P$ and by exploiting the global
594 fairness assumption. The resulting protocol is not self-stabilizing, as the leader is initialized
595 (and otherwise impossible by Prop. 4 - with only P states). Notice that this case is similar
596 to the conditions of Theorem 11 (stating impossibility of naming) except for the fairness
597 condition. In fact, it is not easy to see why a similar reasoning used to prove Th. 11 (roughly
598 the fact that the leader can never detect whether the naming is achieved or not) does not
599 also hold in the current case (with global fairness). Intuitively, this is because the power of
600 global fairness allows to eventually reach a favorable (for stabilization) sub-execution even
601 using a deterministic protocol (though a particular one), while with weak fairness unfavorable
602 sub-executions may persist forever.

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

605 **Proof.** The proposed protocol is a modification of $Count_P$ in the code of the leader. The
606 mobile agents are reducing homonyms to the sink state as in the original $Count_P$. The
607 modified code is given below - Protocol 1. For every $n < P$, the new protocol works in the

608 same way as $Count_P$. Hence, by Theorem 15, it eventually stabilizes to naming for every
 609 $n < P$. The case of $n = P$ is treated separately, in lines 3 - 8. For this case, a variable
 610 $name_ptr$ with possible values in $\{0, \dots, P\}$ is used for indicating to the leader the name
 611 to assign to a mobile agent x interacting with it (using variable $name_x$, from the original
 612 protocol). The variable $name_ptr$ is initialized to 0. Below we consider only the case of
 613 $n = P$.

614 Whenever $n = P$, the leader increments $name_ptr$ each time it meets a mobile agent
 615 whose name is the current value of $name_ptr$. Otherwise, the agent is named by the value
 616 of $name_ptr$, and $name_ptr$ is reset.

617 Let us consider only reduced (to 0) executions (see the definition in Sec. 3.1). From any
 618 *non-terminal* configuration, there is the following possible sequence of interactions during
 619 which the leader first resets $name_ptr$ (if not 0 due to initialization), and then meets the
 620 existing $j < P$ uniquely named agents in the increasing order of their names $0, 1, 2, 3, \dots, j - 1$.
 621 Variable $name_ptr$ is then increased to j (≥ 1). After, the leader meets an agent in a state
 622 different from j . It names the agent by the current value of $name_ptr$ ($= j$), and resets
 623 $name_ptr$ again. Then, the scenario repeats, until $name_ptr$ (and j) reaches P . No value
 624 can be changed thereafter, and all agents are named by names in $\{0, \dots, P - 1\}$. By global
 625 fairness, this terminal naming configuration is eventually reached. ◀

Protocol 1 Space-Optimal Naming under Global Fairness (P states per mobile agent)

Variables at the leader:

$name_ptr$: $[0, \dots, P]$, initialized to 0

Variable at a mobile agent x :

$name_x$: non-negative integer in $[0, \dots, P - 1]$, initialized *arbitrarily*

```

1: when a mobile agent  $x$  interacts with the leader do
2:   execute  $Count_P$ 
3:   if  $Count_P.N = P \wedge name\_ptr < P$  then
4:     if  $name_x = name\_ptr$  then
5:        $name\_ptr \leftarrow name\_ptr + 1$ 
6:     else
7:        $name_x \leftarrow name\_ptr$ 
8:        $name\_ptr \leftarrow 0$ 
9: when two mobile agents  $x$  and  $y$  interact do
10:  execute  $Count_P$ 

```

626 Conclusion and Perspectives

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

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

636 — References

- 637 1 D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R. L. Rivest. Time-space trade-offs in
638 population protocols. In *SODA*, pages 2560–2579, 2017. URL: [https://doi.org/10.1137/1.](https://doi.org/10.1137/1.9781611974782.169)
639 [9781611974782.169](https://doi.org/10.1137/1.9781611974782.169), doi:10.1137/1.9781611974782.169.
- 640 2 D. Alistarh, J. Aspnes, and R. Gelashvili. Space-optimal majority in population protocols.
641 In *SODA*, pages 2221–2239, 2018. URL: <https://doi.org/10.1137/1.9781611975031.144>,
642 doi:10.1137/1.9781611975031.144.
- 643 3 D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks
644 of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.
- 645 4 D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population
646 protocols. *Distributed Computing*, 20(4):279–304, 2007.
- 647 5 D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols.
648 *ACM Trans. Auton. Adapt. Syst.*, 3(4), 2008.
- 649 6 J. Beauquier, J. Burman, S. Clavière, and D. Sohier. Space-optimal counting in population pro-
650 tocols. In *DISC*, pages 631–646, 2015. URL: [https://doi.org/10.1007/978-3-662-48653-5_](https://doi.org/10.1007/978-3-662-48653-5_42)
651 [42](https://doi.org/10.1007/978-3-662-48653-5_42), doi:10.1007/978-3-662-48653-5_42.
- 652 7 J. Beauquier, J. Clement, S. Messika, L. Rosaz, and B. Rozoy. Self-stabilizing counting in
653 mobile sensor networks with a base station. In *DISC*, pages 63–76, 2007.
- 654 8 L. Becchetti, A. E. F. Clementi, E. Natale, F. Pasquale, P. Raghavendra, and L. Trevisan.
655 Friend or foe? population protocols can perform community detection. *CoRR*, abs/1703.05045,
656 2017. URL: <http://arxiv.org/abs/1703.05045>, arXiv:1703.05045.
- 657 9 A. Belleville, D. Doty, and D. Soloveichik. Hardness of computing and approximating
658 predicates and functions with leaderless population protocols. In *ICALP*, pages 141:1–
659 141:14, 2017. URL: <https://doi.org/10.4230/LIPIcs.ICALP.2017.141>, doi:10.4230/
660 LIPIcs.ICALP.2017.141.
- 661 10 P. Berenbrink, R. Elsässer, T. Friedetzky, D. Kaaser, P. Kling, and T. Radzik. A population
662 protocol for exact majority with $o(\log^5/3 n)$ stabilization time and $\theta(\log n)$ states. In
663 *DISC*, pages 10:1–10:18, 2018. URL: <https://doi.org/10.4230/LIPIcs.DISC.2018.10>, doi:
664 [10.4230/LIPIcs.DISC.2018.10](https://doi.org/10.4230/LIPIcs.DISC.2018.10).
- 665 11 O. Bournez, J. Chalopin, J. Cohen, X. Koegler, and M. Rabie. Population pro-
666 tocols that correspond to symmetric games. *IJUC*, 9(1-2):5–36, 2013. URL:
667 [http://www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/
668 ijuc-volume-9-number-1-2-2013/ijuc-9-1-2-p-5-36/](http://www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/ijuc-volume-9-number-1-2-2013/ijuc-9-1-2-p-5-36/).
- 669 12 O. Bournez, J. Cohen, and M. Rabie. Homonym population protocols. *Theory Comput.*
670 *Syst.*, 62(5):1318–1346, 2018. URL: <https://doi.org/10.1007/s00224-017-9833-2>, doi:
671 [10.1007/s00224-017-9833-2](https://doi.org/10.1007/s00224-017-9833-2).
- 672 13 S. Cai, T. Izumi, and K. Wada. How to prove impossibility under global fairness: On space
673 complexity of self-stabilizing leader election on a population protocol model. *Theory Comput.*
674 *Syst.*, 50(3):433–445, 2012.
- 675 14 J. Chalopin and D. Paulusma. Graph labelings derived from models in distributed computing:
676 A complete complexity classification. *Networks*, 58(3):207–231, 2011. URL: [https://doi.org/
677 10.1002/net.20432](https://doi.org/10.1002/net.20432), doi:10.1002/net.20432.
- 678 15 I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively
679 mobile communicating machines that use restricted space. *Theor. Comput. Sci.*, 412(46):6469–
680 6483, 2011. URL: <https://doi.org/10.1016/j.tcs.2011.07.001>, doi:10.1016/j.tcs.2011.
681 07.001.
- 682 16 R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Labeling schemes for
683 tree representation. In *Distributed Computing - IWDC 2005, 7th International Workshop,*
684 *Kharagpur, India, December 27-30, 2005, Proceedings*, pages 13–24, 2005. URL: [http://dx.
685 doi.org/10.1007/11603771_2](http://dx.doi.org/10.1007/11603771_2), doi:10.1007/11603771_2.

- 686 17 C. Cooper, A. Lamani, G.i Viglietta, M. Yamashita, and Y. Yamauchi. Constructing self-
687 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.
- 688 18 G. Cordasco and L. Gargano. Space-optimal proportion consensus with population protocols.
689 In *SSS*, pages 384–398, 2017. URL: https://doi.org/10.1007/978-3-319-69084-1_28, doi:
690 10.1007/978-3-319-69084-1_28.
- 691 19 C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making
692 population protocols fault-tolerant. In *DCOSS*, pages 51–66, 2006.
- 693 20 E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. of the ACM*,
694 17(11):643–644, Nov. 1974.
- 695 21 S. Dolev, A. Israeli, and S. Moran. Self-stabilization of dynamic systems assuming only
696 read/write atomicity. *DC*, 7(1):3–16, 1993.
- 697 22 D. Doty, M. Eftekhari, O. Michail, P. G. Spirakis, and M. Theofilatos. Brief announcement:
698 Exact size counting in uniform population protocols in nearly logarithmic time.
- 699 23 D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear
700 time. *Distributed Computing*, 31(4):257–271, 2018. URL: <https://doi.org/10.1007/s00446-016-0281-z>, doi:10.1007/s00446-016-0281-z.
- 701 24 Ö. Egecioglu and A. K. Singh. Naming symmetric processes using shared variables. *Distributed*
702 *Computing*, 8(1):19–38, 1994. URL: <http://dx.doi.org/10.1007/BF02283568>, doi:10.1007/
703 BF02283568.
- 704 25 P. Fraigniaud, A. Pelc, D. Peleg, and S. Perennes. Assigning labels in an unknown anonymous
705 network with a leader. *Distributed Computing*, 14(3):163–183, 2001. URL: <http://dx.doi.org/10.1007/PL00008935>, doi:10.1007/PL00008935.
- 706 26 L. Gasieniec, D. D. Hamilton, R. Martin, P. G. Spirakis, and G. Stachowiak. De-
707 terministic population protocols for exact majority and plurality. In *OPODIS*, pages
708 14:1–14:14, 2016. URL: <https://doi.org/10.4230/LIPIcs.OPODIS.2016.14>, doi:10.4230/
709 LIPIcs.OPODIS.2016.14.
- 710 27 L. Gasieniec, G. Stachowiak, and P. Uznanski. Almost logarithmic-time space optimal leader
711 election in population protocols. In *SPAA*, 2019.
- 712 28 R. Guerraoui and E. Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine
713 failures. In *ICALP (2)*, pages 484–495, 2009.
- 714 29 T. Izumi, K. Kinpara, T. Izumi, and K. Wada. Space-efficient self-stabilizing counting
715 population protocols on mobile sensor networks. *Theor. Comput. Sci.*, 552:99–108, 2014. URL:
716 <http://dx.doi.org/10.1016/j.tcs.2014.07.028>, doi:10.1016/j.tcs.2014.07.028.
- 717 30 H. Jiang. *Distributed Systems of Simple Interacting Agents*. PhD thesis, Yale University, 2007.
- 718 31 A. Kosowski and P. Uznanski. Brief announcement: Population protocols are fast. In *PODC*,
719 pages 475–477, 2018. URL: <https://dl.acm.org/citation.cfm?id=3212788>.
- 720 32 S. Kutten, R. Ostrovsky, and B. Patt-Shamir. The las-vegas processor identity problem (how
721 and when to be unique). *J. Algorithms*, 37(2):468–494, 2000. URL: [http://dx.doi.org/10.](http://dx.doi.org/10.1006/jagm.2000.1110)
722 1006/jagm.2000.1110, doi:10.1006/jagm.2000.1110.
- 723 33 R. J. Lipton and A. Park. The processor identity problem. *Inf. Process. Lett.*, 36(2):91–94, 1990.
724 URL: [http://dx.doi.org/10.1016/0020-0190\(90\)](http://dx.doi.org/10.1016/0020-0190(90)90103-5)
725 90103-5, doi:10.1016/0020-0190(90)
726 90103-5.
- 727 34 O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Naming and counting in anonymous
728 unknown dynamic networks. In *SSS*, pages 281–295, 2013. URL: [http://dx.doi.org/10.](http://dx.doi.org/10.1007/978-3-319-03089-0_20)
729 1007/978-3-319-03089-0_20, doi:10.1007/978-3-319-03089-0_20.
- 730 35 Y. Mocquard, E. Anceaume, and B. Sericola. Optimal proportion computation with population
731 protocols. In *NCA*, pages 216–223, 2016. URL: <https://doi.org/10.1109/NCA.2016.7778621>,
732 doi:10.1109/NCA.2016.7778621.
- 733 36 E. Ruppert. The anonymous consensus hierarchy and naming problems. In *OPODIS*, pages
734 386–400, 2007. URL: http://dx.doi.org/10.1007/978-3-540-77096-1_28, doi:10.1007/
735 978-3-540-77096-1_28.
- 736
- 737

- 738 **37** Y. Sudo, T. Masuzawa, A. K. Datta, and L. L. Larmore. The same speed timer in population
739 protocols. In *ICDCS*, pages 252–261, 2016. URL: <https://doi.org/10.1109/ICDCS.2016.82>,
740 doi:10.1109/ICDCS.2016.82.
- 741 **38** G. Tel. *Introduction to Distributed Algorithms (2nd ed.)*. Cambridge University Press, 2000.
- 742 **39** S.-H. Teng. Space efficient processor identity protocol. *Inf. Process. Lett.*, 34(3):147–154, 1990.
743 URL: [http://dx.doi.org/10.1016/0020-0190\(90\)90094-E](http://dx.doi.org/10.1016/0020-0190(90)90094-E), doi:10.1016/0020-0190(90)
744 90094-E.
- 745 **40** H. Yasumi, F. Ooshita, K. Yamaguchi, and M. Inoue. Constant-space population protocols for
746 uniform bipartition. In *OPODIS*, pages 19:1–19:17, 2017. URL: [https://doi.org/10.4230/](https://doi.org/10.4230/LIPIcs.OPODIS.2017.19)
747 [LIPIcs.OPODIS.2017.19](https://doi.org/10.4230/LIPIcs.OPODIS.2017.19), doi:10.4230/LIPIcs.OPODIS.2017.19.

748 Appendix

749 Additional Related Work

750 The problem of distributing distinct identifiers to undistinguishable processors has received a
751 lot of attention in the model of shared memory. The problem has been formally introduced
752 by Lipton and Clark in [33], under the name of Processor Identity Problem (PIP), where
753 a solution, under the form of a probabilistic $O(n^2)$ protocol, was given. This solution was
754 improved in [39], which presented an $O(n \log^2 n)$ solution. In [24], a randomized protocol
755 assigns distinct identifiers to the processes within an expected polynomial number of rounds
756 using a polynomial number of boolean atomic variables. The authors of [32] propose a Las
757 Vegas randomized algorithm which terminates in $O(\log n)$ expected rounds and uses $O(n)$
758 shared memory space.

759 In [36], it is investigated whether the assumption of unique identifiers is essential for wait-
760 free distributed computing using shared objects of various types. Results on the solvability
761 of two key problems, consensus and naming, are given. In the *synchronous* model of dynamic
762 graphs, [34] studies the naming problem, assuming that the nodes are (uniformly) initialized.
763 In [25], the task of assigning distinct labels to nodes of an unknown anonymous network in a
764 distributed manner is considered. The model is synchronous and there is a distinguishable
765 node. The goal is to assign short labels, as fast as possible. [16] solves the naming problem
766 probabilistically, similarly assuming that no knowledge on the communication graph is given.
767 The size of the identifiers is proven to be $O(\log n)$ with high probability and their expected
768 size is $O(\log n)$ too. Finally, notice that [14] considers eleven different models of distributed
769 systems and studies the computational complexity conditions for the naming problem to be
770 solvable on a given network.

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

774 Concerning the same version of population protocols as considered in the current study,
775 a sort of naming appear (as a by-product) in some protocols for other related problems,
776 like counting [7, 29, 6], leader election [13] and deterministic oscillation [17]. In [15], an
777 initialized naming protocol is given and the obtained unique names are used for simulating a
778 nondeterministic Turing machine and characterizing the computational power of population
779 protocols having $O(\log n)$ bits of agent’s memory. Concerning naming, the current work
780 improves on all these results in terms of either space complexity and/or model assumptions.

781 A series of recent works (e.g., [23, 1, 2, 27]), considering random population protocols
782 (where each interaction is chosen uniformly at random), jointly contribute to the study of the
783 trade-offs between time and space for the tasks of majority and leader election. The most

784 efficient recent results solve these problems in closely logarithmic ($o(\log^2 n)$) expected parallel
 785 time using optimally $O(\log \log n)$ states in case of leader election [27], and using $O(\log n)$
 786 states in case of majority [10]. Other recent works study different important problems, in
 787 the same model and from a similar general perspective of space and/or time optimality. The
 788 results are still in terms of asymptotic complexity and under randomness conditions (may be
 789 available both in the model and in the protocols). Some examples of considered problems are
 790 proportion computation [35], proportion consensus [18], plurality consensus [26], function
 791 computation [9, 31], community detection [8] and counting [22].

792 On the contrary, [17] introduces the problem of (self-stabilizing) oscillator construction in
 793 population protocols, and studies the *exact* state-space complexity of the possible solutions,
 794 in the model version considered here. Finally, another relevant recent work [40] devoted to
 795 uniform bipartition in population protocols presents a similar type of comprehensive analysis,
 796 considering various combinations of model parameters. Differently from the analysis here,
 797 only the feasibility and optimality of *constant*-space complexity solutions are considered.
 798 Cases where the necessary state complexity depends on n are not treated by the exact
 799 state-space analysis.

800 Negative Results - Section 3 - Proofs

801 **Proposition 3.** *Even with a uniform initialization of agents, but without a leader, and*
 802 *using only P states per agent, it is impossible to obtain symmetric naming in the population*
 803 *protocol model, under weak or global fairness.*

804 **Proof.** Assume by sake of contradiction that such symmetric protocol \mathcal{PP} exists. Consider
 805 a starting configuration C_0 with at least two homonyms. By Lemma 2, the only possible
 806 non-null transition rules are between two homonyms. Such transitions cannot eliminate
 807 homonyms. Thus \mathcal{PP} cannot stabilize from C_0 to a terminal configuration with unique
 808 names - a contradiction. ◀

809 Positive Results - Section 4 - Proofs

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

813 **Proof.** Consider the following symmetric protocol with state space $Q = \{0, 1, \dots, P\}$ and
 814 defined by three types of transition rules:

815 **1.** *if $s \neq P : (s, P) \rightarrow (s, s + 1 \bmod P)$; 2.* *if $s \neq P : (s, s) \rightarrow (P, P)$; 3.* $(P, P) \rightarrow (1, 1)$.

816 To prove correctness, let us show that from any configuration C there exists a reachable
 817 configuration C^* such that in C^* all agents have distinct names in $\{0, \dots, P - 1\}$. Thus,
 818 consider a configuration C and make interact homonyms having a state different from P ,
 819 iteratively and as many times as possible. These agents have to execute transition 2 and
 820 change their states to P . When transition 2 cannot be executed anymore, and if, in the
 821 resulting configuration, there are some non-homonyms in states different from P , we denote
 822 this configuration by C' . Otherwise, if all the agents are in state P , let us force interactions
 823 for reaching this kind of a configuration C' . First, make two agents x and y interact, applying
 824 the transition rule 3, changing their states to 1. Then, make y interact with an agent
 825 $z \notin \{x, y\}$ in state P , applying rule 1 (z exists as long as $n > 2$). After that, agents x, y, z
 826 are in states 1, 1, 2 respectively. Make x interact with y to change their states to P . Now,

827 every agent, excluding z , is in state P (z is in state 2). This configuration is of the same
 828 type as C' , with some non-homonyms apart the agents in state P . Next, we show that a
 829 configuration of type C^* is reachable from any such C' .

830 For that, we associate the following sequence of states to C' . We order all the states
 831 different from P and present in C' , to obtain an increasing sequence s_1, s_2, \dots, s_k (recall that
 832 the states are numbers). In addition, let us add s_1 at the end of this sequence, resulting in
 833 $s_1, s_2, \dots, s_k, s_1$. Two consecutive states s_i, s_j in such a sequence are called *distant* if, either
 834 $[j = i + 1 \text{ and } s_j - s_i > 1]$, or $[i = k, j = 1 \text{ and } s_k - s_1 < P - 1]$. Notice that, for any distant
 835 s_i, s_j , $s_i + 1 \pmod P$ is not present in C' , so the transition rule 1 applied with the s_i -agent
 836 in C' cannot create homonyms.

837 If in C' there is an agent x in state P , choose two consecutive distant states s_i, s_j and
 838 make an agent in state s_i interact with x (applying rule 1 of the protocol). The obtained
 839 configuration C'' is of the same type as C' - the only possible homonyms are in state P
 840 and there are agents in states different from P . Associate to C'' an increasing sequence of
 841 ordered states as before. If there is still an agent y in state P and two neighboring distant
 842 states s'_i, s'_j , make y interact with an agent in s'_i , applying rule 1. Continue similarly from
 843 the resulting configuration, until no agents in state P exist. This finally occurs, because each
 844 newly obtained configuration has no homonyms, except possibly the agents in state P , the
 845 number of which strictly decreases in each next configuration. When no agents in state P
 846 exist anymore, configuration C^* is obtained. As C^* is infinitely often reachable (from any
 847 configuration C), by global fairness, such configuration is eventually reached, providing a
 848 naming. In C^* , no transition rule is applicable. ◀

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

851 **Proof.** Consider, for example, the following protocol. The mobile agents are initially in
 852 the state P , and the leader has a counter (of size P) initialized to 1. Whenever the leader
 853 interacts with a P -state agent, and the counter is less than P : the state of the agent is set
 854 to the counter value and the counter is incremented, if less than P . By a simple induction,
 855 one can easily prove that, after the k th interaction of the leader with a P -agent, for $k < P$,
 856 the agent is named k and never changes its name after. If the leader's counter reaches P (for
 857 $n = P$), only null transitions are possible, all the agents are named from 1 to P , and the
 858 protocol terminates. Hence, the protocol stabilizes to a configuration where all $n \leq P$ agents
 859 have distinct names, using only P states per agent. ◀