HAL
open science

Optimally Gathering Two Robots

Adam Heriban, Xavier Défago, Sébastien Tixeuil

▶ **To cite this version:**

Adam Heriban, Xavier Défago, Sébastien Tixeuil. Optimally Gathering Two Robots. [Research Report] UPMC Sorbonne Universités. 2017. hal-01575451

HAL Id: hal-01575451

https://hal.sorbonne-universite.fr/hal-01575451

Submitted on 20 Aug 2017

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.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Optimally Gathering Two Robots

Adam Heriban* Xavier Défago† Sébastien Tixeuil*

*UPMC Sorbonne Universités, France
†Tokyo Institute of Technology, Japan

Abstract

We present an algorithm that ensures in finite time the gathering of two robots in the non-rigid ASYNC model. To circumvent established impossibility results, we assume robots are equipped with 2-colors lights and are able to measure distances between one another. Aside from its light, a robot has no memory of its past actions, and its protocol is deterministic. Since, in the same model, gathering is impossible when lights have a single color, our solution is optimal with respect to the number of used colors.

1 Introduction

Networks of mobile robots evolving in a 2-dimensional Euclidean space recently captured the attention of the distributed computing community, as they promise new applications (rescue, exploration, surveillance) in potentially dangerous (and harmful) environments. Since its initial presentation [22], this computing model has grown in popularity and many refinements have been proposed (see [18] for a recent state of the art). From a theoretical point of view, the interest lies in characterizing the exact conditions for solving a particular task.

In the model we consider, robots are anonymous (*i.e.*, indistinguishable from each-other), oblivious (*i.e.*, no persistent memory of the past is available), and disoriented (*i.e.*, they do not agree on a common coordinate system). The robots operate in Look-Compute-Move cycles. In each cycle a robot "Looks" at its surroundings and obtains (in its own coordinate system) a snapshot containing the locations of all robots. Based on this visual information, the robot "Computes" a destination location (still in its own coordinate system) and then "Moves" towards the computed location. Since the robots are identical, they all follow the same deterministic algorithm. The algorithm is oblivious if the computed destination in each cycle depends only on the snapshot obtained in the current cycle (and not on the past history of execution). The snapshots obtained by the robots are not consistently oriented in any manner (that is, the robots' local coordinate systems do not share a common direction nor a common chirality[1]).

The execution model significantly impacts the ability to solve collaborative tasks. Three different levels of synchronization have been considered. The strongest model [22] is the fully synchronized (FSYNC) model where each phase of each cycle is performed simultaneously by all robots. The semi-synchronous (SSYNC) model [22] considers that time is discretized into rounds, and that in each round an arbitrary yet non-empty subset of the robots are active. The robots that are active in a particular round perform exactly one atomic Look-Compute-Move cycle in that round. It is

[1]Chirality denotes the ability to distinguish left from right.

assumed that the scheduler (seen as an adversary) is fair in the sense that in each execution, every robot is activated infinitely often. The weakest model is the asynchronous model [18] (ASYNC), which allows arbitrary delays between the Look, Compute and Move phases, and the movement itself may take an arbitrary amount of time. In this paper, we consider the most general ASYNC model.

Related Work. The gathering problem is one of the benchmarking tasks in mobile robot networks, and has received a considerable amount of attention (see [18] and references herein). The gathering tasks consist in all robots (each considered as a dimensionless point in a 2-dimensional Euclidean space) reaching a single point, not known beforehand, in finite time. A foundational result [22, 9] shows that in the SSYNC model, no deterministic algorithm can solve gathering for two robots without additional assumptions. This impossibility result naturally extends to the ASYNC model.

In hostile environments such as those we envision, robots are likely to fail. So far, three kinds of failures were considered in the context of deterministic gathering [1, 7, 16, 5, 13, 14]:

1. *Transient faults*: robots may experience transient faults that corrupt their current state, leading to an arbitrary initial configuration, from which algorithms may not recover (*e.g.*, if the initial configuration is bivalent, and the execution model is SSYNC). As a result, the only known self-stabilizing solution [16] (that is, able to recover from *any* initial configuration) in the classical fair model considers only odd sets of robots (so, a bivalent configuration cannot exist in this setting). Other self-stabilizing algorithms impose stricter assumptions on the scheduler [13, 14].

2. *Crash faults*: when robots may stop executing their algorithm unexpectedly (and correct robots are not able to distinguish a correct robot from a crashed one at first sight), guaranteeing that correct robots still gather in finite time is a challenge. All proposed solutions so far [1, 7, 5, 8, 13, 14] are restricted to the SSYNC model.

3. *Byzantine faults*: when robots may have completely arbitrary (and possibly malicious) behavior, there exists no deterministic gathering protocol in the SSYNC model even assuming that at most one robot may be Byzantine [1]. This impossibility extends to the ASYNC case.

To circumvent the aforementioned impossibility results, it was proposed to endow each robot with a *light* [12], that is, it is capable of emitting a fixed number of colors visible to all other robots. This additional capacity allows to solve gathering of two robots in the most general ASYNC model, provided that robots lights are capable to emit at least *four* colors. In the more restricted SSYNC model, Viglietta [23] proved that being able to emit two colors is sufficient to solve the gathering problem. In the same paper [23], he also proved that an algorithm that only makes use of observed colors to decide on its next move cannot gather two robots in the ASYNC model using only two colors. Finally, Viglietta proves [23] that three colors and the ability to detect null distances is sufficient in ASYNC. Both solutions in ASYNC [12, 23] and SSYNC [23] output a correct behavior independently of the initial value of the lights' colors. Recently, Okumura *et al.* [20] presented an algorithm with two colors that gathers robots in ASYNC assuming *rigid* moves (that is, the move of every robot is never stopped by the scheduler before completion), or assuming non-rigid moves but robots are aware of δ (the minimum distance before which the scheduler cannot interrupt their

Reference	Synchrony	Rigid	Initial Color	δ known	# Colors
[12]	**ASYNC**	**NO**	**NO**	**NO**	*4*
[23]	*SSYNC*	**NO**	**NO**	**NO**	**2 (opt.)**
[23]	**ASYNC**	**NO**	**NO**	**NO**	*3*
[20]	**ASYNC**	*yes*	*yes*	**NO**	**2 (opt.)**
[20]	**ASYNC**	**NO**	*yes*	*yes*	**2 (opt.)**
This paper	**ASYNC**	**NO**	**NO**	**NO**	**2 (opt.)**

Table 1: Gathering robots with lights

move). Also, the solution of Okumura *et al.* [20] requires lights to have a specific color in the initial configuration.

The remaining open case is the feasibility of gathering with only two colors in the most general ASYNC model, without additional assumptions.

Our Contribution. We present a solution to the gathering of two robots that is optimal with respect to all considered criteria: it uses the minimum number of colors (compared to the 4 colors required by the algorithm of Das *et al.* [12] and the 3 colors of the algorithm of Viglietta [23] that operates in the same system ASYNC setting), it performs in the most general ASYNC model (unlike the SSYNC algorithm of Viglietta [23]), while still not requiring additional assumptions such as rigid moves, initial colors, or knowledge about δ (unlike the algorithms of Okumura *et al.* [20]). Characteristics of our solution and comparison with previous work are summarized in Table 1, where boldface characteristics are optimal.

2 Model

We consider the ASYNC model of robots with lights introduced in [12]. In more details, we assume a system of two robots endowed with colored lights (with two possible colors, Black and White) that are modeled as mobile colored points in the Euclidean plane $\mathbb{R}^2$.

Both robots execute cycles that consist of three phases : LOOK, COMPUTE and MOVE. When a robot is not executing a LOOK-COMPUTE-MOVE (LCM) cycle, it is considered to be in a WAIT phase. Each phase can be described as follows:

- WAIT: The robot is idle and waiting for activation.
- LOOK: The robot is taking a snapshot of the position of the other robot, as well as the colors of both robots. This phase is assumed instantaneous.
- COMPUTE: The robot computes its next destination using the snapshot. A robot is able to change the color of its own light at the end of its COMPUTE phase.
- MOVE: The robot moves towards its destination.

The duration of the COMPUTE and MOVE phases, and the delay between two phases, are chosen by an adversary and can be arbitrary long, but finite. The adversary decides when robots are activated assuming a *fair* scheduling *i.e.*, in any configuration, all robots are activated within finite

time. The adversary also controls the robots movement along their target path and can stop a robot before reaching its destination, but not before traveling at least a distance $\delta > 0$ (δ being unknown to the robots). In other words, if a robot has a target at a distance x, we assume that, at the end of its MOVE phase, the robot has moved a distance in the interval $[min(\delta, x), x]$. The exact position reached is determined by the adversary scheduler.

Robots are anonymous, meaning they are indistinguishable and they execute the exact same algorithm. However, for the sake of practicality, they are called A and B in the sequel. Each robot has a local coordinate system about which we make no assumptions, in particular, A and B may have distinct North, chirality, and unit distance. We also assume that, except from their light color, robots have no mean of explicitly communicating with each other.

The robots are also oblivious, in that they do not remember their past actions. This implies that the COMPUTE phase can have no other input than the snapshot from the last LOOK phase.

3 Our Algorithm

3.1 Previous Results

Viglietta observes [23] that, in order to solve the gathering problem in SSYNC, an algorithm must accomplish two things:

- In case robots are synchronized, they need to move towards the midpoint.
- In case robots are activated alternatively, one needs to move towards the other. In that case, the other robot must not move.

In ASYNC, Viglietta also shows [23] that no algorithm using only two colors can solve gathering if the destination computation solely relies on this form of calculation:

$$me.destination = (1 - \lambda) \cdot me.position + \lambda \cdot other.position$$

With:

$$\lambda = f(me.color, other.color)$$

Where f is a function (that is, it associates to a 2-tuple a single image).

It is similarly assumed that the next color of a robot only depends on the current colors of the two robots and not on the distance between the robots.

Algorithms that follow these rules of computation are called class $\mathcal{L}$ algorithms. Then, the only algorithm of class $\mathcal{L}$ that satisfies these criteria is presented in Figure 1.

Now, there exists an execution of this algorithm that does not solve ASYNC gathering (see Lemma 4.9 in Viglietta's paper [23]) when both robots start in the Black color:

1. Let both robots perform a Look phase, so that both will turn White.
2. Let robot A finish the current cycle and perform a new Look, while the B waits. Hence, A remains White and moves to B's position. Now, we let B finish the current cycle and perform a new Look. So, B turns Black and moves to the midpoint m between A and B.

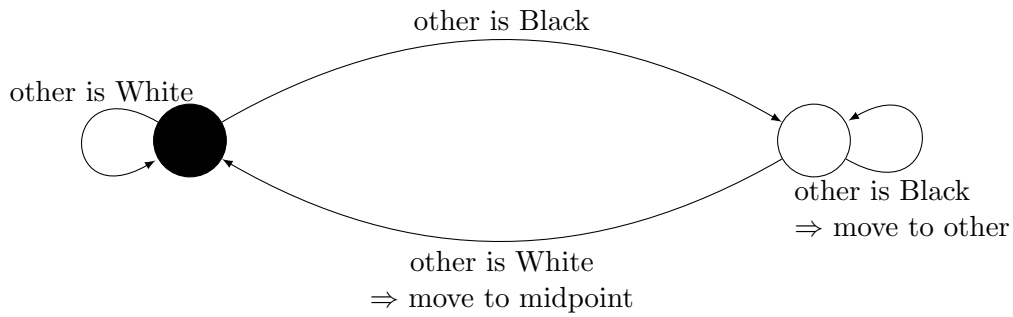

Figure 1: Viglietta's [23] Algorithm

3. Let A finish the current cycle, thus reaching B, and perform a whole new cycle, thus turning Black.

4. Finally, let B finish the current cycle, thus turning Black and moving to m.

As a result, both robots are again set to Black, are in a Wait phase, both have executed at least one cycle, and their distance has halved. Thus, by repeating the same pattern of moves, they approach one another but never gather.

Because of this execution, it is not possible to solve gathering with two colors with an $\mathcal{L}$ class algorithm.

As a result, we do not design our algorithm to be of class $\mathcal{L}$, as our computation of the next color not only depends on the respective colors of the two robots, but also on their respective distance.

3.2 Our Algorithm

We observe that in the problematic aforementioned execution, there is an instant when both robots are actually gathered, but are later separated because of pending moves.

We thus introduce a behavior change in the White state of Viglietta's [23] algorithm to obtain our proposal, presented in Figure 2 and Algorithm 1.

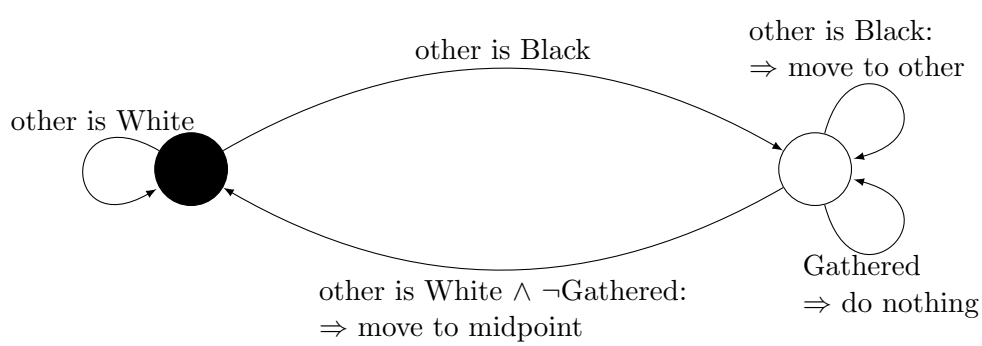

Figure 2: Our Algorithm

Our proposal breaks the infinite loop in the problematic execution, as it prevents robot A from switching to color Black after reaching B and forces it to remain White. This implies that activating

Algorithm 1 ASYNC Robot gathering with two colors

```
if me.color = White then
  if (me.position = other.position) then
    do nothing
  else if other.color = White then
    me.destination ⇐ other.position/2
    me.color ⇐ Black
  else if other.color = Black then
    me.destination ⇐ other.position
  end if
else if me.color = Black and other.color = Black then
  me.color ⇐ White
end if
```

B afterwards actually separates the robots into different colors, and prevents them from going back to both being Black.

Let us observe that our new algorithm no longer belongs to class $\mathcal{L}$, since the same observed 2-tuple of colors may yield different outcomes depending on the distance between A and B. In particular, when both robots are observed White, the next color depends on whether the two robots are gathered. So, the assumption of Viglietta [23] that a new color is solely determined by the current colors no longer holds.[2]

We now need to prove that this new algorithm actually solves the gathering problem in ASYNC in a self-stabilizing manner. Our main result can be stated as follows:

Theorem 1. *Algorithm 1 solves the gathering problem for two robots in a self-stabilizing fashion for the non-rigid ASYNC model.*

4 Correctness

4.1 Describing configurations

We wish to prove the self-stabilizing property of our algorithm in the most general ASYNC model for the gathering specification. As a result, possible configurations must include all possible relative positions of the two robots, but also their current light color, and most importantly their current phase (and phase status) in the look-compute-move cycle. Indeed, the current phase can be seen as a *program pointer* whose value can be initially corrupted in an arbitrary initial setting. A direct consequence of this observation is that all possible combinations of those parameters must be considered as possible initial configurations. Let us first list in Figure 3 the various phases that can be reached by the algorithm, for two robots A and B:

[2] It is worth noting that, while the definition of class $\mathcal{L}$ does not explicitly mention that the new color is also obtained as a function of the two observed colors, the Lemma 4.4 of Viglietta's paper [23] entirely relies on this implicit fact, and so does the 3-color algorithm for the ASYNC model.

	W	C2H	C2O	M2O	W	C2W	C2B	M2H	C2W
W	A1								
C2H	A2	A9							
C2O	A3	D1							
M2O	A4	D2							
W	B1		B3		A5				
C2W	A6								
C2B	B2				A7				
M2H	C1	A11	C2	C3		A8		A10	
C2N	F1	F2	F3	F4	F5	F6	F7	F8	F9

Figure 3: Possible configurations

The possible phases for a robot A are:

- **W**: Wait,
- **C2H**: Compute to Half (at the end of his COMPUTE phase, A switches its color to Black, and moves towards the midpoint),
- **C2O**: Compute to Other (at the end of his COMPUTE phase, A stays White and moves towards the other robot),
- **M2O**: Move to the Other robot's position,
- **C2W**: Compute to White (COMPUTE phase that leads to color change and no motion),
- **C2B**: Compute to Black (COMPUTE phase that leads to no color change and no motion),
- **M2H**: Move to Half point,
- **C2N**: Compute to Nothing (COMPUTE phase that leads to no motion).

Since robots are anonymous, we only need to consider half of the possible configurations (*i.e.*, the combination $(\mathbf{W}, \mathbf{C2H})$ is the same as $(\mathbf{C2H}, \mathbf{W})$). In a given configuration, the scheduler may activate either A, B, or both. In most cases, activating both robots has the same effect as activating A then B, or B then A. However, in a few cases, the outcome of simultaneously activating A and B is not deterministic, and may lead to two different configurations. Those non-deterministic configuration are outlined in yellow color in Figure 3.

To ease the description and reasoning about the various phases of the algorithm, we divide the complete set of configurations into six subsets. Each subset of configurations is drawn as a graph whose vertices represent configurations and (directed) edges represent the possibility to reach another configuration by executing the algorithm of either one or both robots (3 cases). Some vertices appear in several subsets as the subsets we consider do not form a partition of all configurations. To facilitate the reading of each subset, we adopted a color code for vertices described in Figure 4, and the rest of the diagram notations are presented in Figure 5.

Each graph is divided in two parts : the top part, called nominal behavior and a bottom part called terminal behavior. The behavior of the algorithm is called *nominal* when the robots are not gathered. Some nodes, represented in the bottom part, change their behavior if the gathering is achieved during their activation. This is the case *e.g.* when a White robot starts its LOOK phase and perceives the other robot and itself on the same location. Specific gathered behavior is called *terminal*.

Definition 1 (Nominal Configuration)**.** *A configuration c is* nominal *when the two robots are not located in the same position in c.*

Definition 2 (Terminal Configuration)**.** *A configuration c is* terminal *when the two robots are located in the same position in c.*

Definition 3 (Valid Configurations)**.** *A subset $\mathcal{C}_v$ of configurations is* valid *if for any configuration $c \in \mathcal{C}_v$, either* (i) *c is nominal and every execution leads to a terminal configuration in a finite number of steps, or* (ii) *c is terminal and mandates the robots to eventually remain in a gathered configuration forever.*

Definition 4 (Faulty Configurations)**.** *A subset $\mathcal{C}_v$ of configurations is* faulty *if for at least one configuration $c \in \mathcal{C}_v$, A and B are moving towards different targets.*

We now list the subsets we consider in the sequel, assuming A and B denotes the two robots in the system:

(a) **SYM**: configurations where A and B can remain synchronized indefinitely. Those configurations would be considered the regular behavior in a FSYNC model where robots initially share the same color. The **SYM** configurations are presented in Figure 6.

(b) **ASYM**: configurations where A and B are different colors, and no possibility of being synchronized again. The **ASYM** configurations are presented in Figure 7.

(c) **FAULTY 1**: configurations that can be reached after a White to Black de-synchronization which allow different targets for A and B and cannot lead back to **SYM**. The **FAULTY 1** configurations are presented in Figure 8.

Figure 4: Vertex color code

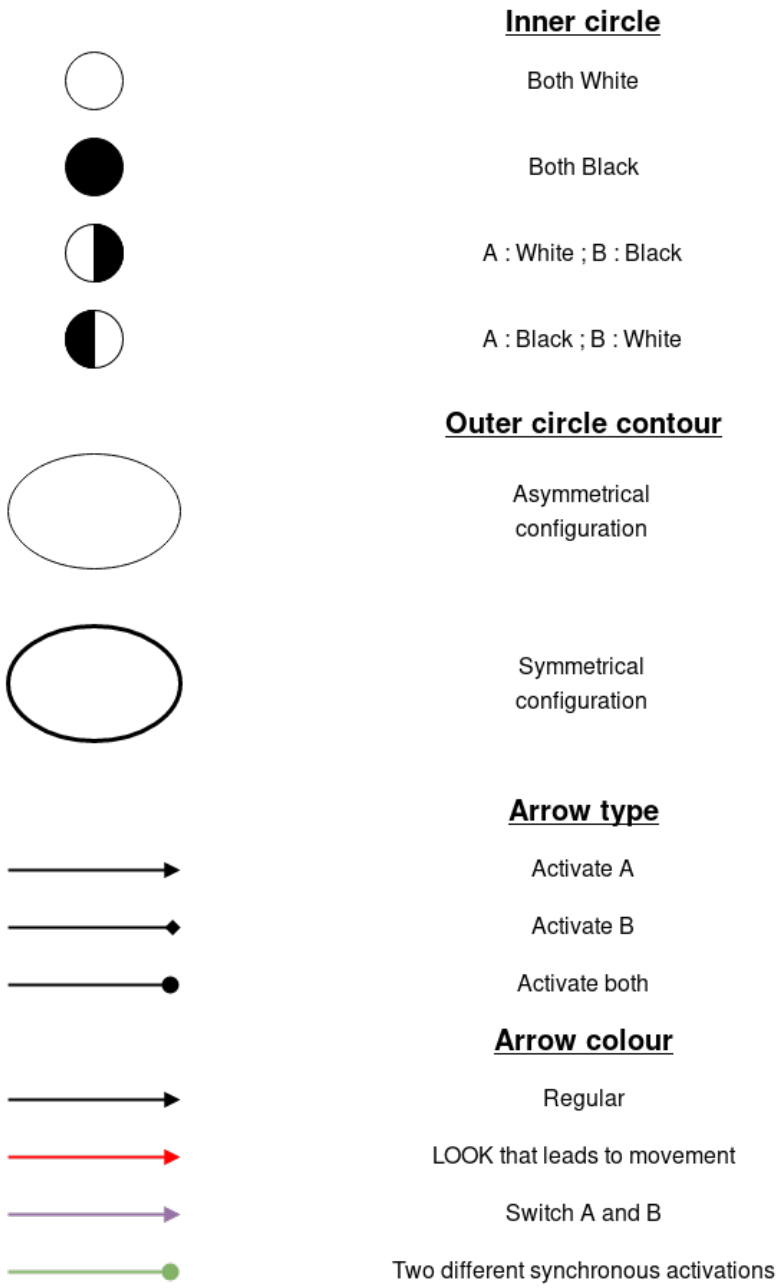

Figure 5: Graph nomenclature

(d) **FAULTY 2**: configurations that can be reached after a Black to White de-synchronization and allow different targets for A and B. The **FAULTY 2** configurations are presented in Figure 9.

(e) **ILLEGAL**: configurations that cannot be reached by the algorithm but need to be taken into account as possible starting configurations to ensure self-stabilization. The **ILLEGAL** configurations are presented in Figure 10.

(f) **GATHERED**: configurations that can only be reached if the gathering is complete. The **GATHERED** configurations are presented in Figure 11.

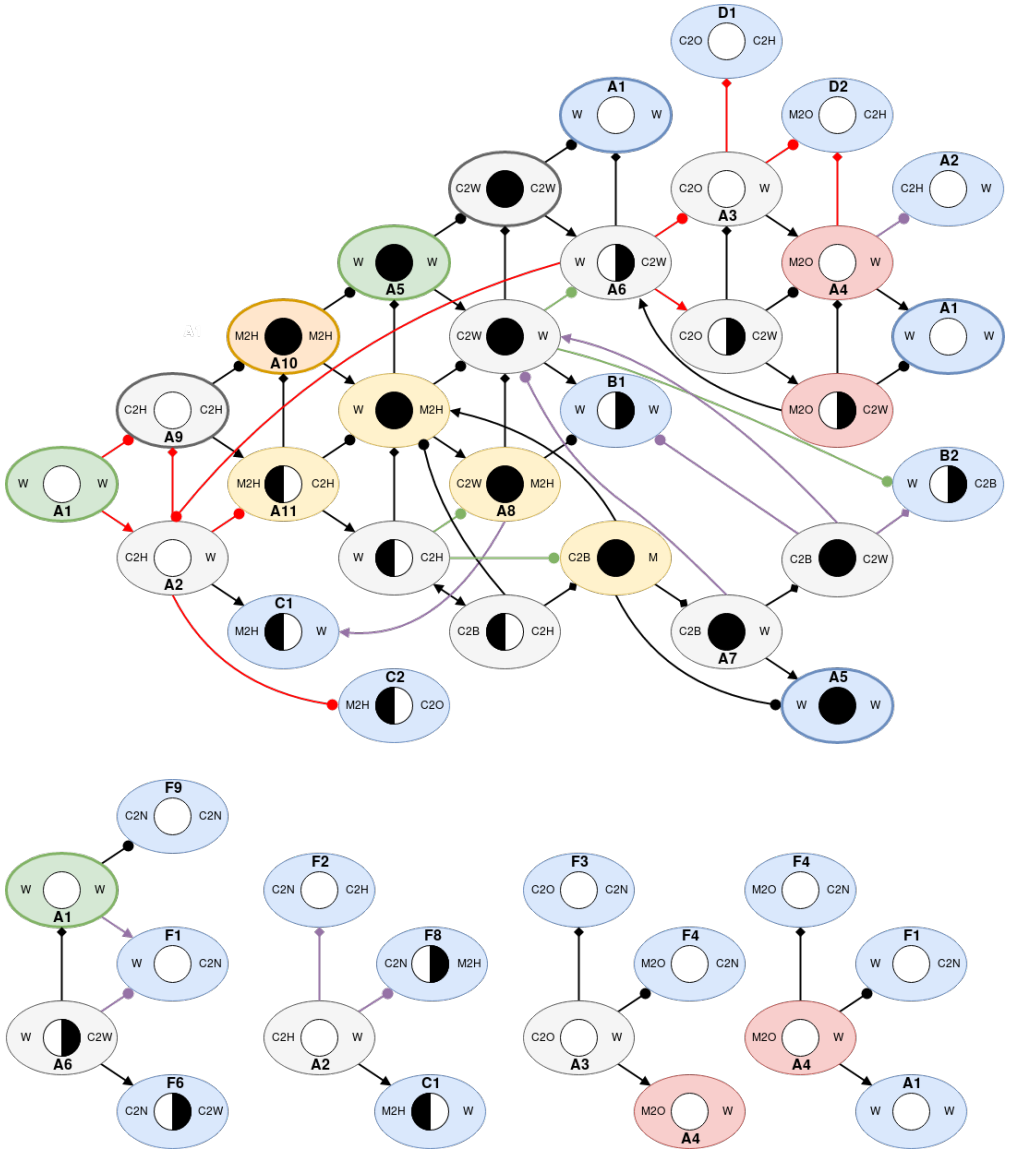

Figure 6: **SYM** configurations

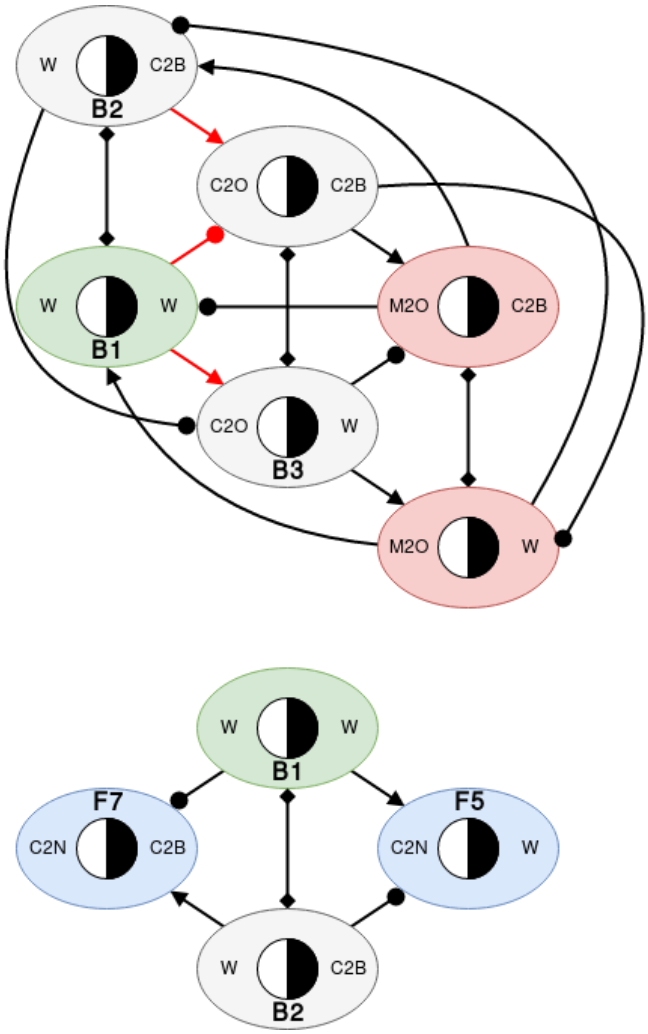

Figure 7: **ASYM** configurations

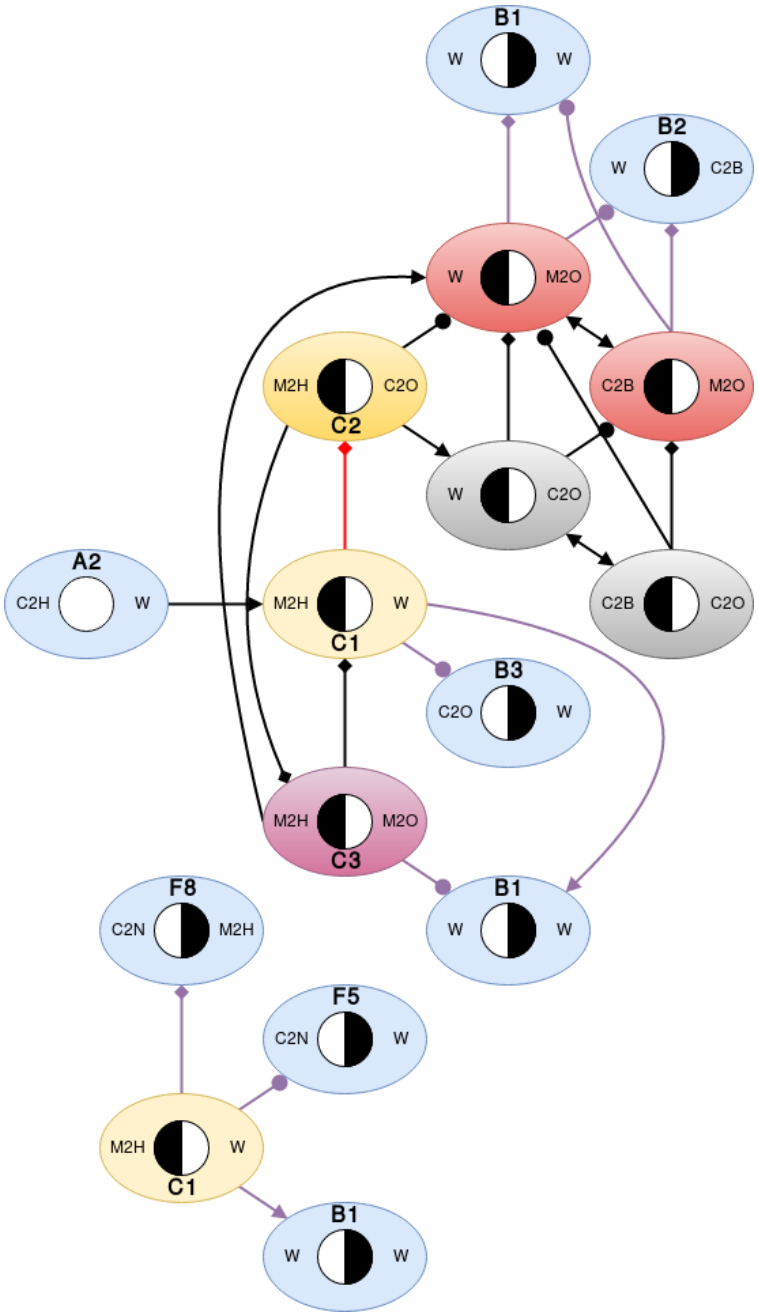

Figure 8: **FAULTY 1** configurations

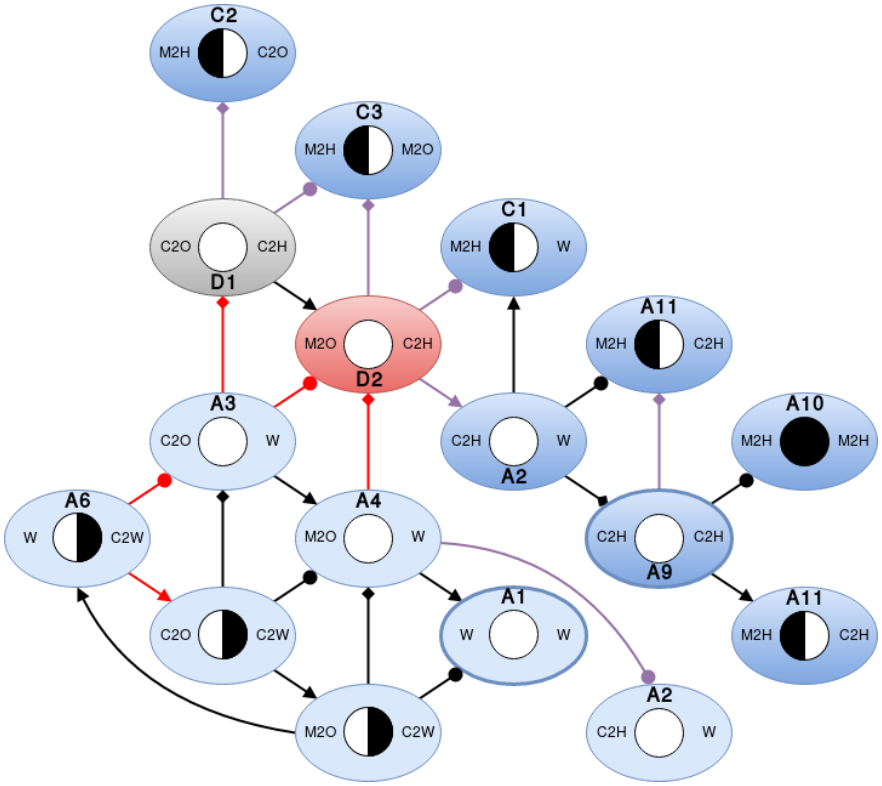

Figure 9: **FAULTY 2** configurations

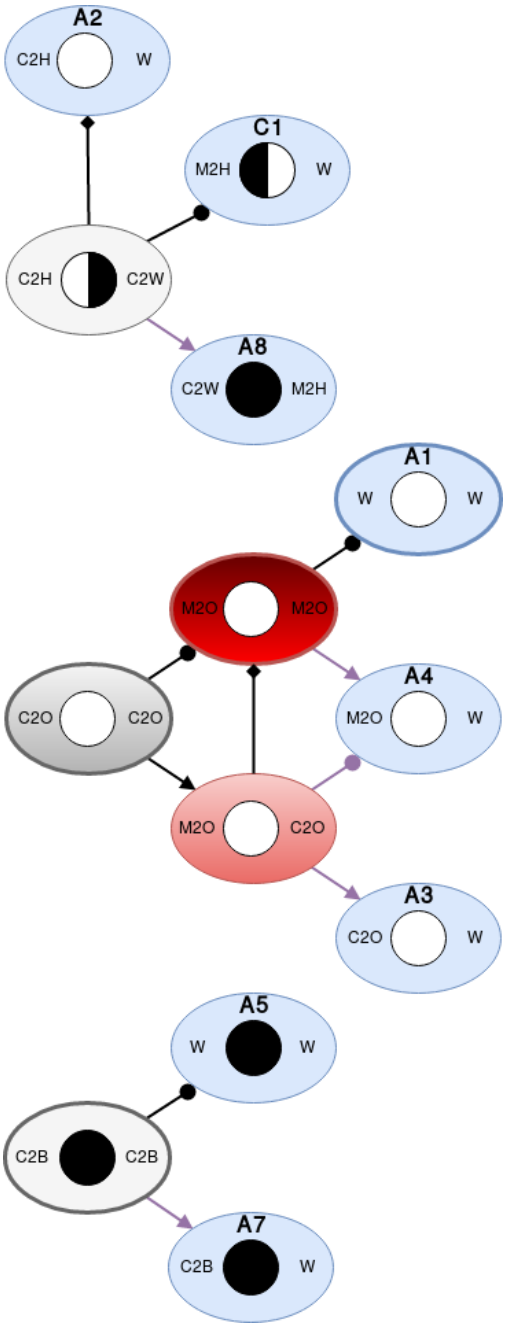

Figure 10: **ILLEGAL** configurations

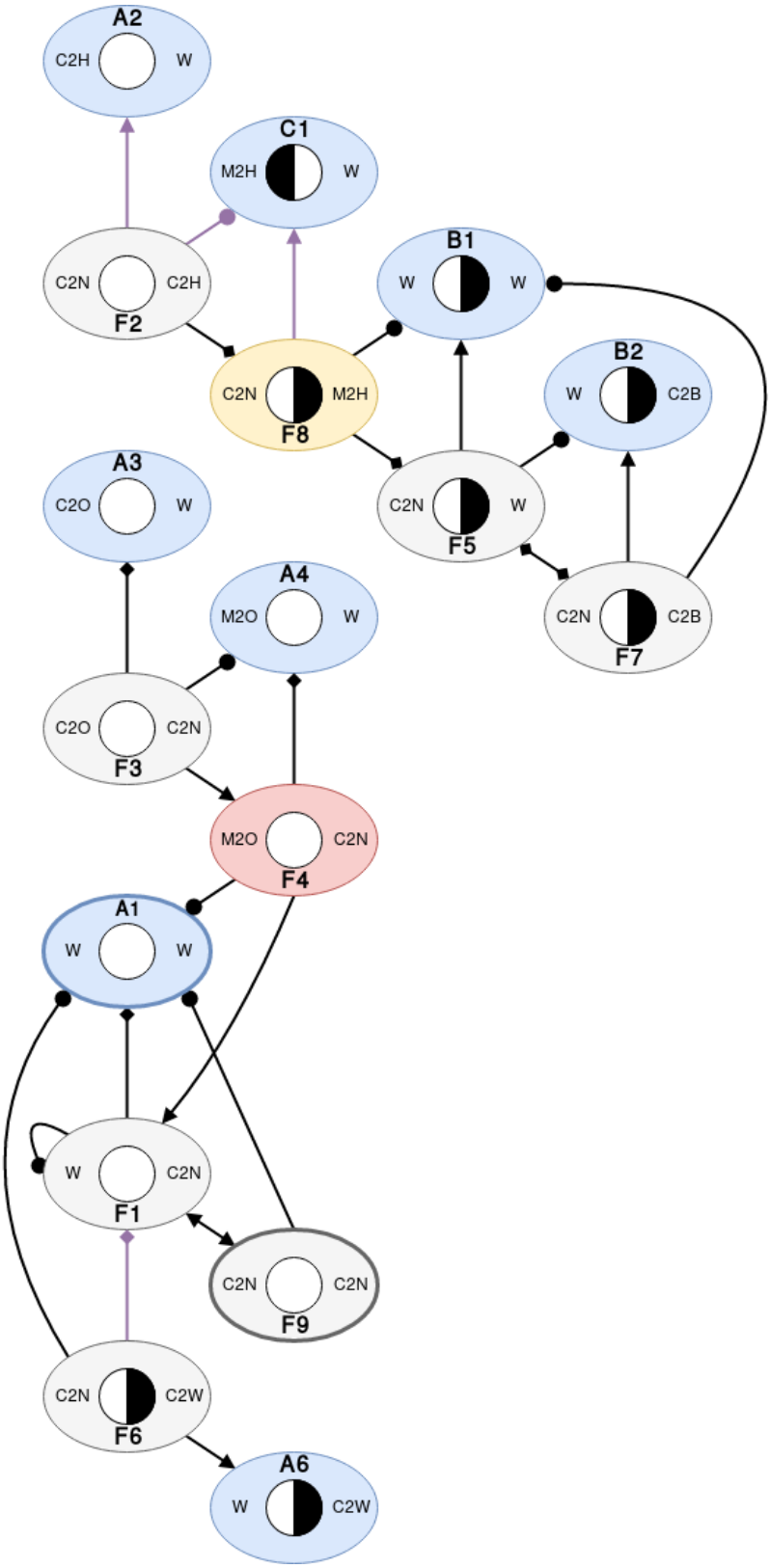

Figure 11: **GATHERED** configurations

4.2 Proof of correctness

Our main result (Theorem 1) is directly obtained by Lemmas 1 and 2, that are presented in the sequel.

Lemma 1. *If every subset of configurations is valid, then the Algorithm 1 solves gathering in a self-stabilizing manner.*

Proof. If all subsets are valid and all configurations are included in the subsets, then it means that all configurations eventually lead to gathering. Thus the algorithm solves the gathering problem in a finite number of steps. □

Lemma 2. *Every subset of configurations of Algorithm 1 is valid.*

Proof. To prove this, we proceed in order: We first prove the validity of **ASYM** (Lemma 3). We then show that **FAULTY 1** can only lead to **ASYM** and is therefore valid (Lemma 4).

As **SYM** and **FAULTY 2** are interdependent, we use a series of Lemmas to obtain validity results for those subsets (see Corollaries 1 and 2 for **FAULTY 2** and **SYM** subsets, respectively).

Finally, we prove **ILLEGAL** (Lemma 10) and **GATHERED** (Lemma 11) to be valid subsets. □

Lemma 3. *The **ASYM** subset is valid.*

Proof. **ASYM** is the subset reached after a successful de-synchronization between A and B, where A and B now have distinct colors with no option to get identical colors again. The **ASYM** subset has no possible exit to another subset, and from a nominal configuration, fair activations of the robots lead to:

- A moves towards B
- B does not move

As A progresses by at least δ or reaches B, nominal behavior leads to both robots sharing the same coordinates in a finite number of steps. Then, the next activation of A least to either configuration $B1$ or $B2$. One more activation of A leads to either $F5$ or $F7$ configuration, which in turn, leads to $B1$ or $B2$. Overall, the system is stuck in a no-movement infinite loop. So, the **ASYM** subset is valid. □

Lemma 4. *The **FAULTY 1** subset is valid.*

Proof. The **FAULTY 1** subset is reached after a White to Black de-synchronization. It is then possible to have robot A move towards the midpoint between A and B, and B moving towards A.

We observe that in the **FAULTY 1** subset, from a nominal configuration, the system can only exit **FAULTY 1** through **ASYM**, which is a valid subset (Lemma 3). It is also worth noticing that, while three cycles exist in **FAULTY 1**, none of them can actually be traversed forever under the fair scheduler assumption, *i.e.* it would require one robot to never be activated. Also, while it is possible to switch from the first cycle to the second, and from the second to the third, it is not possible to move back to a previous cycle, and breakage of the third cycle leads the system to the **ASYM** subset.

Therefore, the nominal behavior of **FAULTY 1** leads to ASYM, which is valid, and is therefore also valid.

Note that the only configuration of **FAULTY 1** whose behavior is modified by reaching gathering is $C1$. $C1$ can then lead to either $B1$ or **ASYM**, or $F5$ and **ASYM**. Finally, if only the White robot is activated, it is possible to cycle between $C1$ and $F8$. However, the fair scheduler assumption requires this cycle to be broken and the system to move to the **ASYM** subset. We can conclude that, despite being a faulty subset, **FAULTY 1** is merely a temporary subset leading to **ASYM**, provided that the scheduler is fair.

Another possible proof of the validity of **FAULTY 1** is found in [20], Lemma 2. It is proven that if robot A performs a LOOK right after B performed a COMPUTE while robots are of different colors, gathering is unavoidable. This happens when reaching C2 from either C1 or A2. Since the only paths in **FAULTY 1** either crosses C2 or immediately exits through **ASYM**, it is easily proven that **FAULTY 1** is valid. □

Lemma 5. *If we exclude the exit to **FAULTY 2**, the **SYM** subset is valid.*

Proof. The **SYM** subset is considered the normal starting starting subset (A and B waiting in White). It includes the FSYNC states ($A1$ to $A1$ through simultaneous activations), and every state that can be reached by $A1$ and lead back to $A1$ (except through **FAULTY 2** by Hypothesis).

In **SYM**, it is possible to exit through **ASYM**, **FAULTY 1**, or **FAULTY 2**. It is also possible to cycle through **SYM**. This can either be done from $A6$ to $A2$, $A7$ to $A5$ or by cycling back to $A1$. In each case, cycling through **SYM** implies both robots target the midpoint while not moving, and then both of them moving towards the midpoint. As the distance between robots decreases by 2δ in each cycle, this behavior is valid. We have already proven **ASYM** and **FAULTY 1** to be valid (Lemmas 3 and 4). Therefore, **SYM** nominal behavior is valid if we do not consider exits to **FAULTY 2**.

Achieving gathering implies terminal behavior in configurations $A1$, $A2$, $A3$, $A4$, and $A6$. Possible terminal behaviors are:

- $A1$ can lead to $F1$, which leads back to $A1$,
- $A1$ can lead to $F9$ which leads back to $A1$,
- $A2$ can lead to **ASYM**,
- $A2$ can lead to $F2$ or $F8$, which leads back to $A2$, **FAULTY 1**, or **ASYM**,
- $A3$ can lead to $F3$ or $F4$, which leads back to $A3$, $A4$ or $A1$,
- $A3$ can lead to $A4$,
- $A4$ can lead to $A1$,
- $A4$ can lead to $F1$, which leads to $A1$,
- $A4$ can lead to $F4$, which leads to $A1$ or back to $A4$,
- $A6$ can lead to $A1$,

- $A6$ can lead to $F1$, which leads to $A1$,
- $A6$ can lead to $F6$, which leads to $A1$ or back to $A6$.

From a terminal configuration, it is thus impossible to reach **FAULTY 2**, as neither $A6$ nor $A2$ can lead to their faulty targets any longer. Furthermore, Reaching $A2$ with an invalid target after achieving gathering now leads to **FAULTY 1** or **ASYM**, which are valid subsets. In turn, looping though $A2$ requires an unfair scheduler. Let us observe that, once gathering is complete, SYM can either loop without motion, or lead to a valid subset. Therefore, **SYM** has a valid terminal behavior.

Overall, **SYM**, excluding exits to **FAULTY 2**, is a valid subset. □

Lemma 6. *If we exclude the exits to **SYM**, the **FAULTY 2** subset is valid.*

Proof. Excluding the path to **SYM** through $A2$ leaves **FAULTY 1** as the only possible exit for **FAULTY 2**. Because we have proven **FAULTY 1** to be valid (Lemma 4), **FAULTY 2** is also valid. □

Lemma 7. *The **FAULTY 2** → **SYM** → **FAULTY 2** cycle reduces the distance x between A and B by at least $min(\frac{x}{2}, 3\delta)$.*

Proof. Looping through the **FAULTY 2** → **SYM** → **FAULTY 2** cycle requires to traverse configuration $A6$, and then configuration $A2$.

Let us assume that in $A6$, we have:

- A is White, in position 0 with no target,
- B is Black, in position x with no target.

Then, by exploring every possible path, when reaching $A2$, the following holds:

- A is in position $[min(\delta, x), x]$ with no target,
- B is in position x with either $\frac{x}{2}$ or $[\frac{x}{2}, x]$ as a target.

Now, to go back to $A6$, the system first needs to exit $A2$, which necessarily makes A target the midpoint between A and B. It then needs to execute both targets. This implies that, when back in $A6$, the distance between A and B is now in the interval $[-\frac{x}{2}, x - 3\delta]$. □

Lemma 8. *After reaching $A6$ with a distance less than δ, reaching $A2$ again implies gathering has been achieved.*

Proof. If the distance x between A and B in configuration $A6$ is less than δ, then at the end of $D2$, gathering is achieved, as A would have moved to B. However, a robot still has an invalid target as it enters $A2$. □

Lemma 9. *Repeating the **FAULTY 2** → **SYM** → **FAULTY 2** cycle leads to a terminal configuration.*

Proof. As we have proven in Lemma 7, looping this cycle reduces the distance by at least $min(\frac{x}{2}, 3\delta)$. Because of this, an scheduler repeating this pattern cannot prevent the robots from getting (strictly) closer than δ from one another in a finite number of steps. Once this has happened, the next use of the cycle necessarily leads to a gathering (by Lemma 8). The nominal behavior of FAULTY 2 is then valid. □

Corollary 1. ***FAULTY 2*** *is valid.*

Proof. By Lemma 9, the nominal behavior of **FAULTY 2** is valid.

Now, **FAULTY 2** does not have a terminal behavior. We can, however, note that in terminal behavior, $A2$ cannot lead to $A6$, as we have proven in Lemma 5. This implies that the cycle leads to gathering, and is then broken when reaching $A2$.

Therefore, **FAULTY 2** is a valid subset. □

Corollary 2. ***SYM*** *is valid.*

Proof. From Corollary 1 and Lemma 5. □

Lemma 10. *The* ***ILLEGAL*** *subset is valid.*

Proof. The **ILLEGAL** subset covers configurations that cannot be reached except through transient errors. Proving its validity is necessary to ensure self-stabilization. We can quickly notice that no cycle exists within the **ILLEGAL** subset, and that this subset necessarily exits to either **SYM** or **FAULTY 1**. As those subsets are valid (Corrolary 2 and Lemma 4), the **ILLEGAL** subset is also valid. □

Lemma 11. *The* ***GATHERED*** *subset is valid.*

Proof. This **GATHERED** subset corresponds to the terminal configuration of all other subsets, which have been proved to be valid. □

5 Conclusions

In this paper we presented a new algorithm to solve gathering of two robots in a self-stabilizing fashion for the non-rigid ASYNC model with an optimal number of lights. Despite being simple to describe, its proof (and the possibility to start from any possible configuration in an asynchronous execution model) mandated significant effort and systematic consideration of all possible cases.

A natural open question is the possibility of self-stabilizing gathering with an arbitrary number of robots (in particular, a solution that starts from a bivalent configuration with two piles of $\frac{n}{2}$ robots would be of interest), and the minimal number of colors that is required in case the problem is solvable. However, we expect the complexity of the proof to go beyond what is tractable by a human, and would like to consider the possibility of using formal methods. Currently, model-checking [15, 4, 17, 21] and program synthesis [6, 19] cannot scale to an arbitrary number of robots, and proof assistant techniques [2, 11, 10, 3] do not yet permit to reason about the ASYNC model. Most likely, solving self-stabilizing gathering with n robots in ASYNC will require significant advances in mobile robot formalization.

References

[1] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. Comput.*, 36(1):56–82, 2006.

[2] Cédric Auger, Zohir Bouzid, Pierre Courtieu, Sébastien Tixeuil, and Xavier Urbain. Certified impossibility results for byzantine-tolerant mobile robots. In Teruo Higashino, Yoshiaki Katayama, Toshimitsu Masuzawa, Maria Potop-Butucaru, and Masafumi Yamashita, editors, *Stabilization, Safety, and Security of Distributed Systems - 15th International Symposium, SSS 2013, Osaka, Japan, November 13-16, 2013. Proceedings*, volume 8255 of *Lecture Notes in Computer Science*, pages 178–190. Springer, 2013.

[3] Thibaut Balabonski, Amélie Delga, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain. Synchronous gathering without multiplicity detection: A certified algorithm. In Borzoo Bonakdarpour and Franck Petit, editors, *Stabilization, Safety, and Security of Distributed Systems - 18th International Symposium, SSS 2016, Lyon, France, November 7-10, 2016, Proceedings*, volume 10083 of *Lecture Notes in Computer Science*, pages 7–19, 2016.

[4] Béatrice Bérard, Pascal Lafourcade, Laure Millet, Maria Potop-Butucaru, Yann Thierry-Mieg, and Sébastien Tixeuil. Formal verification of mobile robot protocols. *Distributed Computing*, 2016.

[5] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. In M. Sohel Rahman and Etsuji Tomita, editors, *WALCOM: Algorithms and Computation - 9th International Workshop, WALCOM 2015, Dhaka, Bangladesh, February 26-28, 2015. Proceedings*, volume 8973 of *Lecture Notes in Computer Science*, pages 149–160. Springer, 2015.

[6] François Bonnet, Xavier Défago, Franck Petit, Maria Potop-Butucaru, and Sébastien Tixeuil. Discovering and assessing fine-grained metrics in robot networks protocols. In *33rd IEEE International Symposium on Reliable Distributed Systems Workshops, SRDS Workshops 2014, Nara, Japan, October 6-9, 2014*, pages 50–59. IEEE, 2014.

[7] Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS 2013)*, pages 337–346, Philadelphia, PA, USA, July 2013. IEEE Press.

[8] Quentin Bramas and Sébastien Tixeuil. Wait-free gathering without chirality. In *Structural Information and Communication Complexity - 22nd Intl. Colloquium (SIROCCO), Post-Proceedings*, volume 9439, pages 313–327, Montserrat, Spain, July 2015.

[9] Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain. Impossibility of gathering, a certification. *Information Processing Letters (IPL)*, 115(3):447–452, January 2015.

[10] Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain. Certified universal gathering in \mathbb R ^2 for oblivious mobile robots. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, volume 9888 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2016.

[11] Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain. Impossibility of Gathering, a Certification. *Information Processing Letters*, 115:447–452, 2015.

[12] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609:171 – 184, 2016.

[13] Xavier Défago, Maria Gradinariu Potop-Butucaru, Julien Clément, Stéphane Messika, and Philippe Raipin-Parvédy. Fault and byzantine tolerant self-stabilizing mobile robots gathering. Research Report IS-RR-2015-003, Japan Adv. Inst. of Science and Tech. (JAIST), Hokuriku, Japan, February 2015.

[14] Xavier Défago, Maria Gradinariu Potop-Butucaru, Stéphane Messika, and Philippe Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering: Feasibility study. In *Proc. 20th Intl. Symp. Distributed Computing (DISC)*, volume LNCS 4167, pages 46–60, September 2006.

[15] Stéphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, and Sébastien Tixeuil. Optimal grid exploration by asynchronous oblivious robots. In Andréa W. Richa and Christian Scheideler, editors, *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium, SSS 2012, Toronto, Canada, October 1-4, 2012. Proceedings*, volume 7596 of *Lecture Notes in Computer Science*, pages 64–76. Springer, 2012.

[16] Yoann Dieudonné and Franck Petit. Self-stabilizing gathering with strong multiplicity detection. *Theor. Comput. Sci.*, 428:47–57, 2012.

[17] Ha Thi Thu Doan, François Bonnet, and Kazuhiro Ogata. Model checking of a mobile robots perpetual exploration algorithm. In Shaoying Liu, Zhenhua Duan, Cong Tian, and Fumiko Nagoya, editors, *Structured Object-Oriented Formal Language and Method - 6th International Workshop, SOFL+MSVL 2016, Tokyo, Japan, November 15, 2016, Revised Selected Papers*, volume 10189 of *Lecture Notes in Computer Science*, pages 201–219, 2016.

[18] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.

[19] Laure Millet, Maria Potop-Butucaru, Nathalie Sznajder, and Sébastien Tixeuil. On the synthesis of mobile robots algorithms: The case of ring gathering. In Pascal Felber and Vijay K. Garg, editors, *Stabilization, Safety, and Security of Distributed Systems - 16th International Symposium, SSS 2014, Paderborn, Germany, September 28 - October 1, 2014. Proceedings*, volume 8756 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2014.

[20] Takashi Okumura, Koichi Wada, and Yoshiaki Katayama. Optimal asynchronous rendezvous for mobile robots with lights. Technical report, 2017.

[21] Arnaud Sangnier, Nathalie Sznajder, Maria Potop-Butucaru, and Sébastien Tixeuil. Parameterized verification of algorithms for oblivious robots on a ring. In *Formal Methods in Computer Aided Design*, Vienna, Austria, 2017.

[22] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[23] Giovanni Viglietta. Rendezvous of two robots with visible bits. In Paola Flocchini, Jie Gao, Evangelos Kranakis, and Friedhelm Meyer auf der Heide, editors, *Algorithms for Sensor Systems: 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2013, Sophia Antipolis, France, September 5-6, 2013, Revised Selected Papers*, pages 291–306, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.