Fault and Byzantine Tolerant Self-stabilizing Mobile Robots Gathering

Julien Clement, Xavier Défago, Maria Gradinariu Potop-Butucaru, Stephane Messika, Philippe Raipin-Parvedy

Distributed Computing manuscript No.
(will be inserted by the editor)

Fault and Byzantine Tolerant Self-stabilizing Mobile Robots Gathering

— Feasibility Study —

Julien Clement · Xavier Défago · Maria Gradinariu Potop-Butucaru · Stéphane Messika · Philippe Raipin-Parvédy

the date of receipt and acceptance should be inserted later

Abstract Gathering is a fundamental coordination problem in cooperative mobile robotics. In short, given a set of robots with *arbitrary* initial locations and no initial agreement on a global coordinate system, gathering requires that all robots, following their algorithm, reach the exact same but not predetermined location. Gathering is particularly challenging in networks where robots are oblivious (i.e., stateless) and direct communication is replaced by observations on their respective locations. Interestingly any algorithm that solves gathering with oblivious robots is inherently self-stabilizing if no assumption is made on the initial distribution of the robots.

In this paper, we significantly extend the studies of deterministic gathering feasibility under different assumptions related to synchrony and faults (crash and Byzantine). Unlike prior work, we consider a larger set of scheduling strategies, such as bounded schedulers, and derive interesting lower bounds on these schedulers. In addition, we extend our study to the feasibility of probabilistic self-stabilizing gathering in both fault-free and fault-prone environments.

J. Clement
LIAFA/ Université Paris Diderot, Paris 7, France
E-mail: jclement@liafa.jussieu.fr

X. Défago
School of Information Science, JAIST, Ishikawa, Japan
E-mail: defago@jaist.ac.jp

M. Gradinariu Potop-Butucaru
IRISA/Université Pierre et Marie Curie, Paris 6, France
E-mail: maria.gradinariu@lip6.fr

S. Messika
LRI/Université Paris Sud, France
E-mail: messika@lri.fr

P. Raipin-Parvédy
France Telecom R&D, France
E-mail: philippe.raipin@orange-ft.com

1 Introduction

Many applications of mobile robotics envision groups of mobile robots self-organizing and cooperating toward the resolution of common objectives. In many cases, the group of robots is aimed at being deployed in adverse environments, such as space, deep sea, or after some natural (or unnatural) disaster. It results that the group must self-organize in the absence of any prior infrastructure (e.g., no global positioning), and ensure coordination in spite of the presence of faulty robots and unanticipated changes in the environment.

The *gathering problem*, also known as the *Rendez-Vous* problem, is a fundamental coordination problem in cooperative mobile robotics. In short, given a set of robots with arbitrary initial location and no initial agreement on a global coordinate system, gathering requires that all robots, following their algorithm, reach the exact same location—one not agreed upon initially—within a *finite* number of steps, and remain there.

Similar to the Consensus problem in conventional distributed systems, gathering has a simple definition but the existence of a solution greatly depends on the synchrony of the systems as well as the nature of the faults that may possibly occur. In this paper, we investigate some of the fundamental limits of deterministic and probabilistic gathering in the face of various synchrony and fault assumptions.

To study the gathering problem, we consider a system model first defined by Suzuki and Yamashita (1999), and some variants with various degrees of synchrony. The model represents robots as points that evolve on a plane. At any given time, a robot can be either idle or active. In the latter case, the robot observes the locations of the other robots, computes a target position, and moves toward it. The time when a robot becomes active is governed by an activation daemon (scheduler). In the original definition of Suzuki and

Yamashita, called the SYm model, activations (i.e., look–compute–move) are atomic, and the scheduler is assumed to be fair and distributed, meaning that each robot is activated infinitely often and that any subset of the robots can be active simultaneously. In the CORDA model of Prencipe (2001), activations are completely asynchronous, for instance allowing robots to be seen while moving.

Suzuki and Yamashita (1999) proposed a gathering algorithm for non-oblivious robots in the SYm model. They also proved that gathering can be solved in systems with three or more oblivious robots, but not in systems with only two.[1] Prencipe (2005) studied the problem of gathering in both SYm and CORDA models. He showed that the problem is impossible without additional assumptions such as being able to detect the multiplicity of a location (i.e., knowing the number of robots that may simultaneously occupy that location). Flocchini et al (2005) proposed a solution to gathering, for oblivious robots with limited visibility in the CORDA model, where robots share the knowledge of a common direction (e.g., as given by a compass). Based on that work, Souissi et al (2009) consider a system in which compasses are not necessarily consistent initially. Ando et al (1999) propose a gathering algorithm for SYm model with limited visibility. Cohen and Peleg (2006) study the problem when robots' observations and movements are subject to errors.

None of the studies mentioned above address the feasibility of gathering in fault-prone environments. One of the first steps in this direction was done by Agmon and Peleg (2006). They prove that gathering of correct robots (referred in this paper *weak gathering*) can be achieved in the SYm model even in the face of the crash of a single robot. Furthermore, they prove that no deterministic gathering algorithm exists in SYm model that can tolerate a Byzantine[2] robot. Finally, they consider a stronger model, called fully synchronous, in which all robots are always activated simultaneously, and show that weak gathering can be solved in that model provided that less than one third of the robots are Byzantine.

*Contribution.*In this paper, we study further the feasibility of gathering in the SYm model in both fault-free and fault-prone (crash and Byzantine) environments. In particular, we consider centralized schedulers[3] (i.e., activations occurring in mutual exclusion) and bounded schedulers (i.e., between any two consecutive activations of a robot, no other robot is activated more than a bounded number of times).

More specifically, we obtain the following important results. Firstly, we strengthen the impossibility results of Agmon and Peleg (2006) since we show that their impossibility results also hold in strictly stronger models. Secondly, we outline the limits under which Byzantine and crash-tolerant gathering become possible. In particular, we propose interesting lower bounds on the value k that a k-bounded scheduler must take for the problem to become solvable. Thirdly, we show in what situations randomized algorithms can help solve the problem, and when they cannot. To the best of our knowledge our work was the first to study the feasibility of probabilistic gathering in both fault-free and fault-prone systems.[4]

*Structure of the paper.*The rest of the paper is structured as follows. Section 2 describes the system model and basic terminology. Section 3 formally defines the gathering problem and recalls important lemmas found in the literature. Section 4 proposes possibility and impossibility results for deterministic and probabilistic gathering in fault-free environments. Section 5 and 6 extend the study to crash and Byzantine prone environments. Section 7 summarizes the results, and Section 8 concludes the paper.

2 Model

We define the system model used in the paper, as well as define important terminology. Most definitions are due to Suzuki and Yamashita (1999); Prencipe (2001); Agmon and Peleg (2006).

2.1 Robot networks

We consider a robot network consisting of a finite set of robots arbitrarily deployed in a geographical area. A robot is a device with the ability to sense, compute and move. It can observe (sense) the positions of the other robots in the plane, it can perform local computations based on these observations, and it can move toward a target location it has computed.

When robots are able to sense the entire set of robots (regardless of the distance) it is said that they have *unlimited visibility*; otherwise robots are said to have limited vis-

[1] With two robots, all configurations are symmetrical and may lead to robots endlessly swapping their positions. In contrast, with three or more robots, an algorithm can be made such that, at each step, either the robots remain symmetrical and they eventually reach the same location, or symmetry is broken and this is used to move one robot at a time into the same location.

[2] A Byzantine robot is a faulty robot that behaves arbitrarily, possibly in a way to deliberately prevent the other robots from gathering in a stable way.

[3] The rationale for considering a centralized scheduler is that, with communication facilities, the robots can synchronize by running a mutual exclusion algorithm, such as token passing.

[4] An extended abstract of this work was published as [Défago et al (2006)].

ibility. In this paper, we consider that robots have unlimited visibility.

The robots are said to have *multiplicity knowledge* when they are able to distinguish, in their observations, those locations that are shared by several robots.

2.2 System model

A network of robots that exhibit a discrete behavior can be modeled as an I/O automaton (Lynch (1996)). A network of robots exhibiting a continuous behavior can be modeled as a hybrid I/O automaton (Lynch et al (2003)). This framework allows for a modeling of systems that exhibit both a discrete and continuous behavior and in particular the modeling of robots networks.

The automaton modeling a robot can perform the following actions:

- *Observation (input type action).*
  An observation returns a snapshot of the positions of all the robots within the visibility range. In our case, this observation returns a snapshot of the positions of all robots;
- *Local computation (internal action).*
  A local computation returns a target destination (point);
- *Motion (output type action).*
  A motion directs the actual motion of the robot towards a target destination. The robot may or may not reach the destination, but a progress assumption ensures that it travels a non-infinitesimal distance.

The local state of a robot at time t is the state of its input/output variables and the state of its local variables and registers. A network of robots is modeled by the parallel composition of the individual automata modeling each of the robots. A configuration of the system at time t is the union of the local states of the robots in the system at time t. An execution $e = (c_0, \ldots, c_t, \ldots)$ of the system is an infinite sequence of configurations, where c_0 is the initial configuration[5] of the system, and every transition $c_i \rightarrow c_{i+1}$ is associated with the execution of a subset of the previously defined actions.

Schedulers. A scheduler decides, for every configuration, which subset of the robots is activated (i.e., allowed to perform their actions). A scheduler is (weakly)[6] fair if, in an infinite execution, a robot is activated infinitely often. In this paper we consider the fair version of the following schedulers:

- *arbitrary (unfair)*: At each activation, any non-empty subset of the robots is activated, without further limitations. Among other things, this means that the scheduler may decide to never activate some subset of the robots. The only guarantee is that, in an infinite execution, one of the correct robots must be activated infinitely often.
- *unfair centralized*: The scheduler is unfair (as described above) with the additional restriction that at most one (i.e., exactly one) robot is activated at each activation.
- *fair*: At each activation, any non-empty subset of the robots is activated, with the guarantee that every robot becomes active infinitely often in an infinite execution.
- *fair centralized*: The scheduler is fair (see above) with the additional guarantee that no more than one (i.e., exactly one) robot is activated at each activation.
- *fair bounded*: The scheduler is fair with the additional guarantee that there exists some bound k (unknown to the robots) such that between any two consecutive activations of some robot, no other robot is activated more than k times.
- *fair k-bounded*: The scheduler is fair bounded, but the bound k is fixed and known to the robots.
- *fair bounded regular*: The scheduler is fair k-bounded with $k = 1$.
- *fair centralized bounded regular*: The scheduler is fair bounded regular with the additional guarantee that exactly one robot is active at each activation.
- *fully synchronized*: Every robot is active at every activation.

We now summarize the relationships between the schedulers presented above. Given two schedulers A and B, $A \supset B$ means that the set of all possible executions allowed by scheduler A strictly contains the set of all executions allowed by scheduler B. As a result, any algorithm that is correct under scheduler A is also correct under scheduler B. Likewise, any impossibility proven under scheduler B also holds under scheduler A.

- unfair $\supset$ fair
- unfair $\supset$ unfair centralized
- unfair $\supset$ fair centralized
- fair $\supset$ fair centralized
- fair $\supset$ fair bounded
- fair bounded $\supset$ fair k-bounded
- fair k-bounded $\supset$ fair bounded regular
- fair centralized $\supset$ fair centralized bounded regular
- fair bounded regular $\supset$ fair centralized bounded regular
- fair bounded regular $\supset$ fully synchronized

The schedulers presented so far are deterministic in the sense that a scheduler can make any choice according to an algorithm that meets the constraints set by its type. In contrast, we consider the following *probabilistic* scheduler:

[5] Unless stated otherwise, this paper makes no specific assumption regarding the respective positions of robots in initial configurations.

[6] Weak fairness requires that every robot that is continuously ready to take an action is eventually activated. But, since a robot is always ready to be activated in our model, a distinction between weak and strong fairness is not relevant.

– *probabilistic*: At each activation the robots that become active are selected randomly.

There are many variants of probabilistic schedulers (e.g., fair, centralized, ...), similar to the deterministic ones. In the paper, we only consider the *fair probabilistic* scheduler, which ensures that, in an infinite execution, a robot is activated infinitely often with probability 1.

Faults. In this paper, we address the following failures:

– *crash failures*: In this class, we further distinguish two subclasses: (1) robots physically disappear from the network, and (2) robots stop all their activities, but remain physically present in the network;
– *Byzantine failures*: In this case, robots may have an arbitrary behavior.

2.3 Computational Models

The literature proposes mainly two computational models, namely, SYm and CORDA. The SYm model was introduced by Suzuki and Yamashita (1999). In this model each robot performs, once activated by the scheduler, a *computation cycle* composed of the following three actions: observation, computation and motion. The atomic action performed by a robot in this model is a computation cycle. The execution of the system can be modeled as an infinite sequence of rounds. In a round one or more robots are activated and perform a computation cycle. The SYm model was refined by Agmon and Peleg (2006). The authors distinguish the case of hyperactive systems where all robots are activated simultaneously and non-hyperactive systems where a strict subset of robots are simultaneously activated.

The CORDA model was introduced by Prencipe (2001). This model refines the atomicity of the actions performed by each robot. Hence, robots may perform in a decoupled fashion, the atomic actions of a computation cycle. They may be interrupted by the scheduler in the middle of a computation cycle. Moreover, while a robot performs an action A, where A can be one of the following atomic actions: observation, local computation or motion, another robot may perform a totally different action B.

In this paper, we consider the SYm model,[7] refined with the scheduling strategies presented above. We consider that robots are oblivious (i.e., stateless) in that they do not conserve any information between two computational cycles. Additionally, we make no assumption on the initial positions of the robots.[8] As stated earlier, we assume that robots have unlimited visibility.

[7] Note that all impossibility results proven in the SYm model also hold in the CORDA model

[8] One of the major motivations for considering oblivious robots is that, as observed by Suzuki and Yamashita (1999), algorithms designed for that model are inherently self-stabilizing.

3 The Gathering Problem

A network of robots is in a *terminal (legitimate) configuration* with respect to the gathering requirement if all the robots share the same position in the plane. Let denote by $\mathscr{P}_{Gathering}$ this predicate.

An algorithm solves the gathering problem in an oblivious system if the following two properties are verified:

– **Convergence** Any execution of the system starting in an arbitrary configuration reaches in a finite number of steps a configuration that satisfies $\mathscr{P}_{Gathering}$.
– **Termination** Any execution starting in a terminal configuration with respect to the $\mathscr{P}_{Gathering}$ predicate contains only legitimate configurations.

Gathering is difficult to achieve in most of the environments. Therefore, weaker forms of gathering were studied so far. An interesting version of this problem requires robots to *converge* toward a single location rather than reach that location in a finite time. The convergence is however considerably easier to deal with. For instance, with unlimited visibility, convergence can be achieved trivially by having robots moving toward the barycenter of the network [Suzuki and Yamashita (1999)].

Note that an algorithm that solves the gathering problem with oblivious or stateless robots is self-stabilizing [Dolev (2000)].

3.1 Existing Results

We now present a few lemmas proved previously by others, that are related to our study. When appropriate, the lemmas have been rephrased in order to keep the terminology consistent. First, the following two lemmas have been proved by Suzuki and Yamashita (1999) and refer to oblivious robots under a fair scheduler.

Lemma 1 (Suzuki and Yamashita (1999); Th. 3.1) *There is no deterministic algorithm that solves gathering in the SYm model for $n = 2$ robots.*

Notice that, although the above lemma is expressed according to a fair scheduler (SYm model), the execution used in the proof to show the impossibility is compatible with a fair bounded regular scheduler. It follows that the result also applies to a system based on that scheduler.

Lemma 2 (Suzuki and Yamashita (1999); Th. 3.4) *Gathering of $n \geq 3$ robots can be solved deterministically in the SYm model with multiplicity detection.*

The following lemma, proved by Prencipe (2005), also refers to oblivious robots under a fair scheduler.

Lemma 3 (Prencipe (2005); Th. 2) *In the SYm model (with a fair scheduler), there is no deterministic algorithm that solves gathering for $n \geq 2$ oblivious robots without additional assumptions (e.g., multiplicity detection).*

Finally, the following two lemmas, proved by Agmon and Peleg (2006), refer to models with the presence of faulty robots. These lemmas propose results for the correct robots gathering.

Lemma 4 (Agmon and Peleg (2006); Th. 3.5) *Gathering can be solved deterministically in a $(3,1)$-Crash model in the SYm model (under a fair scheduler with multiplicity detection).*

Note 1 Agmon and Peleg (2006) also show (Th. 3.8) that gathering can be solved deterministically in a $(n,1)$-Crash model for any $n \geq 3$, but under the restriction that the system is never in a configuration with more than one point of multiplicity.

Lemma 5 (Agmon and Peleg (2006); Th. 4.1) *In the SYm model, any deterministic non-hyperactive algorithm fails to solve gathering in a $(3,1)$-Byzantine model.*

Lemma 6 (Agmon and Peleg (2006); Th. 4.4) *In the SYm model (with a fair scheduler), there is no deterministic algorithm that solves gathering in a $(3,1)$-Byzantine model.*

Obviously, the above impossibility also holds in a (n,f)-Byzantine model for any number of robots $n \geq 3$ and in the presence of any number of faulty robots $1 \leq f \leq n$.

Lemma 7 (Agmon and Peleg (2006); Th. 5.3) *Gathering can be solved in an $(3,1)$-Byzantine system in the fully synchronized model.*

Lemma 8 (Agmon and Peleg (2006); Th. 5.10) *Gathering can be solved in an (n,f)-Byzantine system in the fully synchronized model for any $n \geq 3f+1$.*

4 Gathering in Fault-Free Environments

In this section, we refine results showing the impossibility of gathering [Prencipe (2005); Agmon and Peleg (2006)] by proving first that these results hold even under more restrictive schedulers than the ones considered by Prencipe (2005) and Agmon and Peleg (2006). Interestingly, we also prove that some of these impossibility results hold even in probabilistic settings. Additionally, to circumvent these impossibility results, we propose a probabilistic algorithm that solves the fault-free gathering, under a special class of schedulers, known as bounded schedulers.

The following lemma shows that the impossibility result of Prencipe (2005) (Lemma 3) holds even under a weaker scheduler—the centralized fair bounded regular scheduler. Intuitively, a schedule of this particular scheduler is characterized by two properties: each robot is activated infinitely often and between two executions of a robot every robot in the network executes its actions exactly once. Moreover, in each configuration a single robot is allowed to execute its actions.

Lemma 9 *There is no deterministic algorithm that solves gathering for $n \geq 3$ under a fair centralized bounded regular scheduler, without additional assumptions (e.g., multiplicity knowledge).*

Proof With no loss of generality, consider a system with three robots: r_1, r_2 and r_3 arranged as shown on Figure 1. In the following, we construct an infinite execution that never converges. Consider a schedule *Sch* that invariably applies the following scenario: r_3 is chosen first, then r_1, then r_2. The schedule verifies the fair centralized bounded regular scheduler.

Consider now an initial configuration of the system such that robots r_2 and r_3 have the same initial location. Since r_3 does not have the ability to detect multiplicity, it observes the existence of two groups of robots but does not know how many robots are in each group. Since r_3 is oblivious, it cannot deterministically localize r_2 in the groups. Then, according to *Sch*, we can always find a scenario in which the system will cycle infinitely. For instance, *Sch* can generate the following change in robots topology: $(r_1, r_2r_3) \rightarrow (r_1r_3, r_2) \rightarrow (r_3, r_1r_2) \rightarrow (r_2r_3, r_1)$. It follows that the system never converges.

Algorithm 4.1 Probabilistic gathering for robot p.

Functions:
observe_neighbors :: returns the set of robots within visibility range of robot p (the set of p's neighbors). Note that, in a system with unlimited visibility, *observe_neighbors* returns all the robots in the network.

Actions:
$\mathscr{A}_1$ :: *true* $\longrightarrow$
$\mathscr{N}_p = observe\_neighbors()$;
with probability $\alpha = \frac{1}{|\mathscr{N}_p \bigcup \{p\}|}$ do
select a robot $q \in \mathscr{N}_p \bigcup \{p\}$;
move towards q;
Remark: with probability $1-\alpha$*, the position remains unchanged*;

Note that Suzuki and Yamashita (1999) have proved that the deterministic gathering of two oblivious robots is impossible. The scenario is the following: the two robots are always activated simultaneously. Consequently, they continuously swap positions, and the system never converges. In the following, we prove that, for the case of two robots, there exists a probabilistic solution for gathering in the SYm

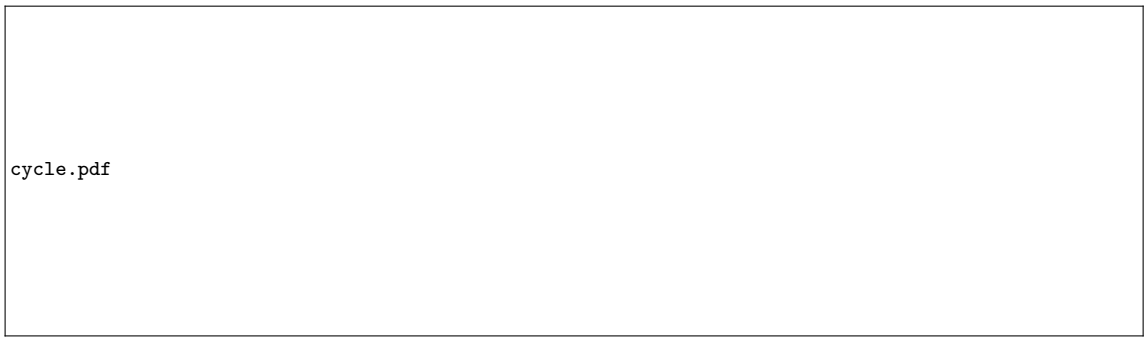

Fig. 1 Lemma 9: constructing an infinite execution.

model, under any type of scheduler. Algorithm 4.1 describes the probabilistic strategy of a robot. When chosen by the scheduler, a robot decides, with probability α, whether it will actually compute a location and move whereas, with probability $1-\alpha$, the robot will remain stationary. The following lemma shows that Algorithm 4.1 reaches a terminal configuration with probability 1.

Lemma 10 *Algorithm 4.1 solves the 2-gathering problem probabilistically under an arbitrary scheduler. It converges in 2 steps in expectation.*

Proof Consider two robots r_1 and r_2, and an arbitrary initial configuration c.

If r_1 and r_2 are gathered at the same location, they are already in a terminal configuration and thus neither will move, regardless of activations and the probability t. This leaves the case when r_1 and r_2 are not gathered initially.

Consider some step i when the two robots have distinct locations. If only one of the robots is activated by the scheduler, then there is a probability α that the robot moves, and thus the robots end up gathered in the next configuration (terminal). If both robots are activated at step i, then, they end up in a terminal configuration if either one of them changes its position. This occurs with probability $2\alpha(1-\alpha)$.

Consequently, the probability of reaching a terminal configuration during step i is at least $q = \min(\alpha, 2\alpha(\alpha-1)) > 0$, regardless of the choice of the scheduler. It follows that the probability of reaching a terminal configuration at step $s > i$ is at least $p = 1-(1-q)^s$. Thus, the probability for the system to reach a terminal configuration is $\lim_{s\to\infty}(1-(1-q)^s) = 1$.

In the following we evaluate the convergence time of the algorithm. Let X_t be the probabilistic variable that models the configuration of the system at time t. Let $\mathscr{L}$ be the set of terminal configurations (robots are gathered at the same location). Let $T_{\mathscr{L}}$ the time to reach a configuration in $\mathscr{L}$, $T_{\mathscr{L}} = min\{t, X_t \in \mathscr{L}\}$. The convergence time of the algorithme (also known as the hitting time) is $E[T_{\mathscr{L}}] = \sum_{i=1}^{\infty} i(\frac{1}{2})^i = 2$.

We complement Lemma 10 with two very simple lemmas.

Lemma 11 *The 2-gathering problem can be solved deterministically under a centralized scheduler (fair or unfair).*

Proof Let r_1 and r_2 be the two robots. Consider the simple algorithm which consists for one robot to move to the location of the next robot. Given that the scheduler is centralized, at most one robot must become active, thus the situation where the two robots exchange their position does not occur. Besides, since there are only two robots, at least one of them must repeatedly become active even if the scheduler is unfair. Thus, gathering is eventually reached.

The next lemma extends the impossibility result proved in Lemma 9 to probabilistic algorithms under fair schedulers.

Lemma 12 *There is no probabilistic algorithm that solves the n-gathering problem, for $n \geq 3$, under a fair centralized scheduler without additional assumptions (e.g., multiplicity knowledge).*

Proof Consider an initial configuration in which nodes form two groups, G_1 and G_2. Consider a schedule such that robots move from G_1 to G_2 and vice-versa, without ever reaching a terminal configuration. Consider now the following simple schedule: select one robot from the group with maximal multiplicity.[9] With no loss of generality, let the selected robot be from G_1. Recall that the algorithm executed by each robot is probabilistic. Therefore, it is possible that the selected robot needs to be activated several times until its coin allows it to actually move. Once the selected robot arrives in the opposite group, G_2, select one of the robots in G_2 and activate it until it is ready to move.

Even if the scheduler is fair the above schedule generates an infinite execution that does not lead to a terminal configuration.

The key issue leading to the above impossibility is the freedom that the scheduler has in selecting a robot r until its probabilistic local computation allows r to actually move.

[9] If the groups have the same multiplicity then select one of the two groups arbitrarily.

The scenario can however no longer hold with systems in which the scheduler is k-bounded. That is, in systems where a robot cannot be activated more than k times before the activation of another robot. In this type of game robots win against the scheduler and the system converges to a terminal configuration.

Lemma 13 *Algorithm 4.1 probabilistically solves the n-gathering problem, $n \geq 3$, under a fair bounded scheduler and without multiplicity knowledge.*

Proof We show that with high probability a group of at least two robots is formed and then, with high probability this group will attract the other robots (even if these robots are not aware of the system multiplicity). Assume an arbitrary initial configuration - no two robots share the same position. Let c_0 be this configuration. Initially, the scheduler chooses s robots ($s \geq 1$). With probability α^s all the s robots choose as meeting point the same robot and move towards this robot. Let's call this robot the core of G. Let G be the group formed by these $s+1$ robots. Let G' be the robots which do not share the same position as the robots in G. Let c_1 be the configuration obtained after the first choice of the scheduler. Starting with c_1 consider the following scenario. Each time a robot in G is chosen by the scheduler it selects with probability α a node in G (itself for example or the core of G). Each time a robot in G' is chosen by the scheduler it selects with probability α a robot in G (for example the core of G). In the worst case, the robots in G' have to wait to enter in G until each robot, already in this set, is chosen at most k times where k is the scheduler bound. Note that each time a robot in G' is chosen, the size of G' probabilistically decreases and the size of G probabilistically increases. Following the above scenario, the probability to obtain a configuration c_2 such that the size of G increases by one and the size of G' decreases by one is superior to $\alpha^{k(s+1)}\alpha$. The size of G' is bounded. Therefore, by applying the previous scenario a terminal configuration is reached in a finite number of steps with positive probability, superior to $\alpha^s\alpha^{k\sum_{i=1,(n-s-1)}(s+i)}\alpha^{n-s-1} \geq \alpha^{k\frac{n(n-1)}{2}+(n-1)} = \varepsilon$, where k is the scheduler bound. The probability that the system converges to a terminal configuration is $\lim_{m\to\infty}\varepsilon(1+\sum_{i=1,m}(1-\varepsilon)^i) = 1$.

5 Fault-tolerant and Self-stabilizing Gathering

In the following we extend the study of the gathering feasibility to fault-prone environments. In this section, (n,f) denotes a system with n robots and up to $f < n$ are faulty (i.e., they can crash). As mentioned in the model, Section 2, in an (n,f) crash-prone system there are two types of crashes: (1) the crashed robots completely disappear from the system, and (2) the crashed robots are still physically present in the system, however they stop the execution of any action. In the first case the problem reduces to the situation where the number of robots in the systems becomes $n-f$ and for this point onward all the results proposed in the fault-free case apply. Therefore, in the sequel we analyze only the case where the faulty robots are still present in the system.

Lemma 14 *In a crash-prone system, $(3,1)$-strong gathering is deterministically impossible under a fair centralized bounded regular scheduler even with multiplicity knowledge.*

Proof Assume a $(3,1)$ system where the faulty robot is still present in the system. Let us denote the three robots r_1, r_2 and r_3. Assume there exists an algorithm, $\mathscr{A}$, that solves the strong gathering problem under a fair centralized bounded regular scheduler. We will analyze in the following two infinite suffixes of an execution of $\mathscr{A}$, named e and e', starting in a configuration c where robots r_1 and r_2 share the same location while r_3 is on a different position. Let e be an execution in which no robot crashes, and let e' be an execution where r_3 crashes. Consider the execution e. Since r_1 and r_2 share the same position in configuration c, the robots r_1 and r_2 will be instructed to not move, and thus continue to share the same location. Otherwise the system may never converge following Lemma 9. Consider now the execution e' starting in the same configuration c. Since r_1 and r_2 have the same view as in the execution e then they will be instructed by Algorithm $\mathscr{A}$ to not move. Assume robot r_3 crashes in c, just before being scheduled. Therefore, even if r_3 is scheduled in c it cannot move and the strong gathering cannot be achieved. Overall, there is no algorithm that solves $(3,1)$-strong gathering even under fair centralized bounded regular scheduler.

The following two corollaries extend the result to systems with more than three robots. The corollaries derive trivially from Lemma 9 and Lemma 12 respectively.

Corollary 1 (of Lemma 9) *In a crash-prone system, there is no deterministic algorithm that solves the $(n,1)$-gathering problem, $n \geq 4$, under a fair centralized bounded regular scheduler without additional assumptions (e.g, multiplicity knowledge).*

Corollary 2 (of Lemma 12) *In a crash-prone system, there is no probabilistic algorithm that solves the $(n,1)$-gathering problem, $n \geq 3$, under a fair centralized scheduler without additional assumptions (e.g., multiplicity knowledge).*

Lemma 15 *In a crash-prone system, $(3,1)$-strong gathering is probabilistically impossible under a fair centralized scheduler even with multiplicity knowledge.*

Proof Assume a $(3,1)$ system where the faulty robot is still present in the system. Let us denote the three robots r_1, r_2 and r_3. Assume there exists a probabilistic algorithm, $\mathscr{A}$,

that solves the strong gathering problem under a fair centralized scheduler. We will analyze in the following two infinite suffixes of an execution of $\mathscr{A}$, named e and e', starting in a configuration c where robots r_1 and r_2 share the same location while r_3 is on a different position. Let e be an execution in which no robot crashes, and let e' be an execution where r_3 crashes. Consider the execution e. Since r_1 and r_2 share the same position in configuration c, the robots r_1 and r_2 will be instructed to not move, and thus continue to share the same location. Consider now the execution e' starting in the same configuration c. Since r_1 and r_2 have the same view as in the execution e then they will be instructed by Algorithm $\mathscr{A}$ to not move. Assume robot r_3 crashes in c, just before being scheduled. Therefore, even if r_3 is scheduled in c it cannot move and the strong gathering cannot be achieved.

Assume in c, r_1 is instructed to move and r_3 is crashed. Since the scheduler is centralized fair the next configuration is symmetrical with c and assume the scheduler chooses r_3, then r_2 till it moves then r_1 till it moves, then again r_3 and r_1 till it moves back. The new configuration is identical to the configuration c and it verifies the fair scheduler predicate.

Overall, there is no algorithm that solves $(3,1)$-strong gathering even with multiplicity knowledge under fair centralized scheduler.

The following lemma proves that $(n,1)$-gathering is probabilistically possible under a bounded scheduler and without additional assumptions.

Lemma 16 *In a crash-prone system, Algorithm 4.1 is a probabilistic solution for the gathering problem in systems with n correct robots but one and under a bounded scheduler.*

Proof If the faulty robot disappears from the system then the result directly follows from Lemma 13 applied to a system with $(n-1)$ robots.

Assume the faulty robot is still physically present in the system. We have to prove that with high probability all correct robots are attracted by the faulty robot. From this point further the proof is identical with the proof of Lemma 13.

In the following, we extend our study to systems with more than one faulty robot. Hereafter, (n,f)-gathering refers to the gathering problem in a system with n correct robots but $f < n$. If the faulty robots disappear from the system, then the problem trivially reduces to the study of a fault-free gathering with $n-f$ correct robots. In contrast, in systems where faulty robots remain physically present in the network after crashing, the problem is far from being trivial. Obviously, gathering all the robots including the faulty ones, is impossible since faulty robots may possibly have crashed at different locations.

From this point on, we study the feasibility of a weaker version of gathering, referred to as *weak gathering*. The (n,f)-*weak gathering* problem requires that, in a terminal configuration, only the *correct* robots must share the same position. The following lemma proves the impossibility of deterministic and probabilistic weak gathering under centralized, bounded, and fair schedulers, without additional assumptions.

Lemma 17 *In a crash-prone system, there is neither a probabilistic nor a deterministic algorithm that solves the (n,f)-weak gathering problem, $n \geq 3$ and $f \geq 2$, under a fair centralized regular scheduler without additional assumptions.*

Proof Consider without restraining the generality $f = 2$. We will prove that each of the faulty robots becomes an attraction point and the correct robots will infinitely migrate between the faulty nodes. Therefore the system never converges. Assume an initial configuration such that the faulty nodes do not share the same position and all the correct robots are quasi-equitable divided in two groups - each group including a faulty node. Let color white and black the two groups respectively. The impossibility result for the deterministic case follows using the same construction as in the proof of Lemma 1. In the probabilistic case, with high probability a correct robot migrates between the two groups.

Algorithm 5.1 Deterministic fault-tolerant weak gathering for robot p

Functions:
observe_neighbors :: returns the set of robots within the vision range of robot p (the set of p's neighbors);
maximal_multiplicity :: returns a robot in the group with the maximal multiplicity; or, if several such groups exists, makes an arbitrary choice among them (different from the curent location)

Actions:
$\mathscr{A}_1$ :: $true \longrightarrow$
$\mathscr{N}_p = observe\_neighbors()$;
$q = maximal\_multiplicity(\mathscr{N}_p)$;
move towards q;

An immediate consequence of the previous lemma is the necessity of an additional assumption (e.g., multiplicity knowledge), even for probabilistic solutions under bounded schedulers.

In the sequel, we identify the conditions under which the weak gathering accepts deterministic and probabilistic solutions. Algorithm 5.1 proposes a deterministic solution for the weak gathering that works under both centralized and bounded schedulers. The idea of the algorithm is as follows: a robot, once chosen by the scheduler, moves to the group with the maximal multiplicity – "attraction action". In case that all groups have the same multiplicity, the chosen robot

will go to the location of another robot – "unbalanced action". The attraction action helps the convergence while the unbalanced action breaks the symmetry.

Lemma 18 *In a crash-prone system, Algorithm 5.1 deterministically solves the (n,f)-weak gathering problem, with $n > f$ under a centralized scheduler (fair or unfair) if robots are aware of the system multiplicity.*

Proof The idea of the proof is the following. We show that eventually a group of maximal multiplicity is created and this group further attracts all correct robots.

Assume an initial configuration such that all nodes have the same multiplicity. More precisely, nodes are evenly distributed in the plane. After the scheduler choice, the chosen robot will create with its target a group of maximal multiplicity. Let denote this group attractor. Each subsequent choice of the scheduler will increase the size of the attractor. Since the number of the correct robots is finite, eventually the system converges to a terminal configuration.

Assume that in the initial configuration all robots are organized in two or more groups with equal multiplicity. After one choice of the scheduler one of these groups increases its multiplicity, hence the above described scenario applies and the system converges to a terminal configuration.

Note that in the previous lemma we considered unfair scheduling strategies with respect to the correct robots. Otherwise, the scheduler can choose infinitly often the crashed robot.

Algorithm 5.2 Probabilistic fault-tolerant gathering for robot p with multiplicity knowledge

Functions:
observe_neighbors :: returns the set of robots within the vision range of robot p (the set of p's neighbors);
maximal_multiplicity :: returns the set of robots with the maximal multiplicity;

Actions:

$\mathscr{A}_1$ :: *true* $\longrightarrow$
  $\mathscr{N}_p = observe\_neighbors()$;
  if $p \in maximal\_multiplicity(\mathscr{N}_p) \wedge |maximal\_multiplicity(\mathscr{N}_p)| > 1$ then
    with probability $\frac{1}{|maximal\_multiplicity(\mathscr{N}_p)|}$ do
      select a robot $q \in maximal\_multiplicity(\mathscr{N}_p)$;
      move towards q;
  else
    select a robot $q \in maximal\_multiplicity(\mathscr{N}_p)$;
    move towards q;

In the following we show that (n,f)-weak gathering can be solved under arbitrary schedulers using a probabilistic algorithm, Algorithm 5.2, and multiplicity knowledge. Algorithm 5.2 works as follows. When a robot is chosen by the scheduler it moves to the group with maximal multiplicity. When all groups have the same size, then the robot tosses a coin to decide if it moves or holds the current position.

Lemma 19 *In a crash-prone system, Algorithm 5.2 probabilistically solves the (n,f)-weak gathering problem, $n > f$, under an unfair scheduler if robots are aware of the system multiplicity.*

Proof If the system starts in a configuration with an unique group of maximal multiplicity then this group will be the attraction point for the other robots in the network. The proof is similar with the proof of Lemma 18. Each robot, once chosen by the scheduler, will deterministically choose as destination the group with maximal multiplicity. Since the number of robots is finite, then the system converges to a terminal configuration in a finite number of steps.

In the case when the system starts in a configuration with equal sized (multiplicity) groups, we will show that with high probability an unbalanced group is formed. Then all the other nodes will be attracted by this group. Assume without restraining the generality, the scheduler chooses a robot in each group with maximal multiplicity. Let $\alpha = \frac{1}{|maximal\_multiplicity(\mathscr{N}_p)|}$ be the probability that a robot chooses a particular target. In the following we evaluate the probability that the initial configuration does not change. With probability $s!\alpha^s$ the system reaches a new configuration which is a permutation of the old configuration (either the s robots didn't change their positions or, once all robots have reached their targets, all groups are again equilibrated). The probability that the new obtained configuration is symmetrical to the old configuration is $s!\alpha^s$. However, the probability that the previous scenario repeats as the number of scheduler choices goes to infinite is $\lim_{m\to\infty}(s!\alpha^s)^m = 0$. Eventually, with high probability the system reaches a configuration where an unique group has the maximal multiplicity. Starting from this point further the proof is similar to the previous case since nodes execute only deterministic actions.

6 Byzantine Tolerant and Self-stabilizing Gathering

In the following we study the gathering feasibility in systems prone to Byzantine failures. In the sequel (n,f) denotes a system with n correct robots but f. Agmon and Peleg (2006) proved that gathering in Byzantine environments is impossible in SYm and CORDA models for the case $(3,1)$. The impossibility proof is given for the case of the SYm model and algorithms that are not hyperactive. The following lemma proves the $(3,1)$-gathering impossibility under the weakest scheduler, in particular the centralized, fair and regular.

Lemma 20 *In a Byzantine-prone system, there is no deterministic algorithm that solves $(3,1)$-weak gathering under*

a fair centralized bounded regular scheduler without additional assumptions.

Proof Consider the following scheduling strategy. Every time a byzantine robot is chosen it chooses to move to an arbitrary location. In particular, it can chose to move to the previous location of the correct robot that moved in the previous round. In the following we show that the system may cycle between two non-terminal configurations. Assume 3 robots, r_1,r_2, r_3 and assume r_3 byzantine. Assume that robots initially does not share the same position. The system topology changes from the topology (r_1, r_2, r_3) to $(-, r_2, r_1r_3) \rightarrow (r_3, r_2, r_1) \rightarrow (r_3r_2, -, r_1) \rightarrow (r_1r_2r_3, -, -)$. Let's denote c the configuration corresponding to the last topology. Since r_3 is byzantine it can move and the system may reach the following topology $(r_1r_2, r_3, -)$ (robots r_1 and r_2 are gathered). If robots do not have the multiplicity knowledge they can be attracted by the byzantine node, hence the system evolves to the following topology $(r_1, r_2r_3, -)$ then $(-, r_1r_2r_3, -)$. Note that the system reached again the configuration c. Overall, the system never reaches a terminal configuration.

Note 2 Note that Algorithm 5.1 solves the Byzantine $(3,1)$-weak gathering under a fair centralized bounded regular scheduler and multiplicity knowledge. The cycle created in the impossibility proof is broken because the Byzantine robot cannot play the attractor role.

The following lemma shows that if the scheduler is relaxed, the $(3,1)$-weak gathering becomes impossible even if robots are aware of the system multiplicity.

Lemma 21 *In a Byzantine-prone system, there is no deterministic algorithm that solves the $(3,1)$-weak gathering, even when robots are aware of the system multiplicity, under a fair centralized k-bounded scheduler with $k \geq 2$.*

Proof The general proof idea is the following : the byzantine node plays the attractor role, hence the system never reaches a terminal configuration. Consider a schedule *Sch* such that after each execution of a correct robot the scheduler gives the permission to the byzantine robot to move. This schedule verifies the specification of the 2-bounded scheduler. Assume that each time a correct node chooses to move, it chooses as target the location of the Byzantine node. Then, following the scheduler *Sch* the Byzantine node will replace the location of the node that just joined its location. Therefore, the system never converges.

Note 3 Byzantine $(n,1)$-weak gathering for any odd $n > 4$ is possible under any fair centralized scheduler and multiplicity knowledge. The algorithm is trivial: a robot moves to the group with maximal multiplicity.

The following lemma establishes a lower bound for the fair centralized bounded scheduler that prevents the deterministic gathering.

Lemma 22 *In a Byzantine-prone system, there is no deterministic algorithm that solves $(n,1)$-weak gathering, with $n \geq 2$ even, under a fair centralized k-bounded scheduler for $k \geq (n-1)$. This result holds even when robots are aware of the system multiplicity.*

Proof Assume by contradiction that there is an algorithm, $\mathscr{A}$, that solves the (n-1)-gathering under a fair k-bounded scheduler for $k \geq (n-1)$. Assume the following initial configuration: robots are organized in two groups of equal size. One of the groups contains the byzantine robot. Let refer the group containing the byzantine robot as the black group and the other group - the white group. Assume the scheduler applies iteratively the following two choices: (C1) choose a robot in the white group; (C2) choose the byzantine robot. Since $\mathscr{A}$ is a deterministic algorithm that solves gathering whenever a correct robot is chosen it moves towards another robot, otherwise the termination property is not verified. Consider in the following that the byzantine robot always moves towards the group with the weaker weight. After the choices C1 and C2 the system reaches a configuration symmetrical to the initial configuration: two groups of equal size one black and one white. In order to win the game the byzantine robot should do a move for each move of a correct robot. Therefore, for any fair centralized k-bounded scheduler with $k \geq (n-1)$ the byzantine robot wins and the gathering becomes impossible.

Corollary 3 *Byzantine $(n,1)$-weak gathering is possible under a centralized scheduler in systems where $n \geq 4$ is odd, robots have multiplicity knowledge and the scheduler is fair.*

Corollary 4 *Byzantine $(n,1)$-weak gathering is possible under a fair centralized scheduler in systems where $n \geq 2$ is even, robots have multiplicity knowledge and the scheduler is k-bounded with $k \leq (n-2)$.*

The following lemma states the lower bound for a bounded scheduler that prevents deterministic gathering.

Lemma 23 *In Byzantine-prone systems, there is no deterministic algorithm that solves (n,f)-weak gathering, $f \geq 2$, under a fair centralized k-bounded scheduler with $k \geq \left\lceil \frac{n-f}{f} \right\rceil$ when n is even, and with $k \geq \left\lceil \frac{n-f}{f-1} \right\rceil$ when n is odd, even when the robots can detect multiplicity.*

Proof **– Even case.** Similar to the $(n,1)$ case above, assume that the system starts in an initial configuration in which all robots are arranged in two groups. Assume the same scheduler as in the $(n,1)$ case: for each move of a correct robot the scheduler chooses a Byzantine robot. The Byzantine robot will try to balance the system equilibrium hence it will move towards the old location of the correct robot. In order to win the game the Byzantine robots need to move each time a correct robot moves.

Since there are $n-f$ correct robots in the system, the scheduler has to be bounded by no less than $\left\lceil \frac{n-f}{f} \right\rceil$ for the Byzantine team to win.

– **Odd case.** For the odd case assume an initial configuration where robots but one (a Byzantine one) are arranged in two groups. When chosen by the scheduler the Byzantine robot not member of a group moves such that the equilibrium between the two groups does not change. Let denote G_1 and G_2 the two groups. Consider the following schedule. Every time a correct robot, member of G_i, moves, a Byzantine robot moves as well in the opposite direction. Hence the system equilibrium does not change. The game is similar to the even case. The only difference is that the number of Byzantine robots that influence the faith of the game is $f-1$. Therefore, in order to win the game, the Byzantine team needs a fair k-bounded scheduler bounded by $k \geq \left\lceil \frac{n-f}{f-1} \right\rceil$.

□

The next lemma is a possibility result when randomization is used. Note that this possibility results needs probabilistic behavior from both algorithm and scheduler.

Lemma 24 *In systems with Byzantine faults, Algorithm 5.2 probabilistically solves the (n,f)-weak gathering, $n \geq 3$, problem under a probabilistic scheduler and multiplicity detection.*

Proof The proof is based on the fact that as soon as there exists a group of $\frac{N}{2}+1$ correct robots gathered, it just needs a"few" more rounds to achieve convergence.

This idea is that every time a robot is selected by the scheduler, it joins this group. Therefore, beyond this point, the convergence time only depends on the scheduler.

Let's study the probability for such a group to be created. We define $\mathscr{L}$: There exists a group of $\frac{N}{2}+1$ correct robots gathered.

$$\mathbb{P}\left[\text{reach } \mathscr{L} \text{ in } \frac{N}{2}+1 \text{ steps}\right] > \left(\frac{1}{N}\right)^{(\frac{N}{2}+1)} = \varepsilon$$

So

$$\mathbb{P}\left[\neg(\text{reach } \mathscr{L} \text{ in } (\frac{N}{2}+1) \text{ steps})\right] \leq (1-\varepsilon)$$

Therefore :

$$\forall k\ \mathbb{P}\left[\neg(\text{reach } \mathscr{L} \text{ in } k(\frac{N}{2}+1) \text{ steps})\right] \leq (1-\varepsilon)^k$$

$$\lim_{k\to\infty} \mathbb{P}\left[\neg(\text{reach } \mathscr{L} \text{ in } k(\frac{N}{2}+1) \text{ steps})\right] = 0$$

Note that in our scenario the convergence time is exponential. In order to simplify the calculations we look at : $\left(\frac{1}{N}\right)^N$ instead of $\left(\frac{1}{N}\right)^{(\frac{N}{2}+1)}$.

So, $\left[1-\left(\frac{1}{N}\right)^N\right]^t \leq \alpha$. This leads to: $tln\left(1-\left(\frac{1}{N}\right)^N\right) \leq ln\alpha$. Overall, t verifies: $t \geq \frac{ln\alpha}{ln\left(1-\left(\frac{1}{N}\right)^N\right)} \sim \ln\left(\frac{1}{\alpha}\right) N^N$

7 Summary

8 Conclusion

The results presented here extend the prior work on the possibility and impossibility of gathering in fault-free and both crash-prone and Byzantine-prone systems. For instance, we strengthen several prior impossibility results by showing that they still hold against weaker schedulers, and under various failure models. We also mark out more accurately the limit between possibility and impossibility by deriving appropriate upper and lower bounds.

To the best of our knowledge, this is actually the first extended study that considers probabilistic solutions to solve the gathering problem in both fault-free and fault-prone models. Here, we identify conditions under which a probabilistic solution exists, as well as conditions for which not even a probabilistic solution exists. The main results of the paper are summed up in Table 1 for fault-free systems; in Table 2 and Table 3 for strong respectivelly weak gathering in crash-prone systems; and in Table 4 for the weak gathering problem in Byzantine-prone systems. Several research directions are opended by our study:

1. Our work is only a first step in investigating the Byzantine tolerant gathering. It should be noted that the only probabilistic possibility result is derived only under a probabilistic scheduler. This open a vaste field of investigation related to the possibility of probabilistic gathering under different schedulers. Also the investigation of the deterministic case is not completly closed.
2. Some of the proofs proposed in the curent document only consider the use of randomization for determining whether a robot takes actions or not when it is activated. One can argue that using randomization in a different way may possibly change some of the lower bounds presented here.
3. The impossibility results and the scheduler lower bounds are computed only for the memoriless robots. An interesting exercise would be to revisit these lower bounds and impossibility results when robots are enhanced with memory and communication skils.
4. In order to close the problem a similar study can be conducted for the CORDA model. Note that the impossibility results we proposed for the SYm model also hold for the CORDA model. As far as the possibility results are concerned we conjecture that our probabilistic algorithms verify the gathering specification in the CORDA model under bounded schedulers.

References

Agmon N, Peleg D (2006) Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM Journal of Computing 36(1):56–

Table 1 Summary: strong gathering; fault-free model.

multiplicity				**Scheduler**	without multiplicity			
deterministic		probabilistic			deterministic		probabilistic	
$n=2$	$n\geq 3$	$n=2$	$n\geq 3$		$n=2$	$n\geq 3$	$n=2$	$n\geq 3$
NO(*L.1*)	**OK(L.2)**	**OK(L.10)**	**OK(L.2)**	unfair	NO(*L.3*)	NO(*L.3*)	**OK(L.10)**	NO(*L.12*)
OK(L.11)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	unfair centr.	**OK(L.11)**	NO(*L.9*)	OK(*L.10*)	NO(*L.12*)
NO(L.1)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	fair	**NO(L.3)**	**NO(L.3)**	OK(*L.10*)	NO(*L.12*)
OK(*L.11*)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	fair centr.	OK(*L.11*)	NO(*L.9*)	OK(*L.10*)	**NO(L.12)**
NO(*L.1*)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	fair bounded	NO(*L.1*)	NO(*L.9*)	OK(*L.10*)	**OK(L.13)**
NO(*L.1*)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	fair k-bounded	NO(*L.1*)	NO(*L.9*)	OK(*L.10*)	OK(*L.13*)
NO(L.1)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	fair bounded reg.	**NO(L.1)**	NO(*L.9*)	OK(*L.10*)	OK(*L.13*)
OK(*L.11*)	OK(*L.2*)	OK(*L.10*)	OK(*L.2*)	fair centr. bounded reg.	OK(*L.11*)	**NO(L.9)**	OK(*L.10*)	OK(*L.13*)

Table 2 Summary: strong gathering; crash model.

multiplicity				**Scheduler**	without multiplicity ($f=1$)			
deterministic		probabilistic			deterministic		probabilistic	
$n=3$	$n\geq 4$	$n=3$	$n\geq 4$		$n=3$	$n\geq 4$	$n=3$	$n\geq 4$
NO(*L.14*)	NO(*L.14*)	NO(*L.15*)	NO(*L.15*)	unfair	NO(*L.3*)	NO(*L.3*)	NO(*L.12*)	NO(*L.12*)
NO(*L.14*)	NO(*L.14*)	NO(*L.15*)	NO(*L.15*)	unfair centr.	NO(*L.14*)	NO(*L.14*)	NO(*L.12*)	NO(*L.12*)
NO(*L.14*)	NO(*L.14*)	NO(*L.15*)	NO(*L.15*)	fair	NO(*L.3*)	NO(*L.3*)	NO(*L.12*)	NO(*L.12*)
NO(*L.14*)	NO(*L.14*)	NO(*L.15*)	NO(*L.15*)	fair centr.	NO(*L.14*)	NO(*L.14*)	NO(*L.12*)	NO(*L.12*)
NO(*L.14*)	NO(*L.14*)	OK(*L.16*)	OK(*L.16*)	fair bounded	NO(*L.14*)	NO(*L.14*)	**OK(L.16)**	**OK(L.16)**
NO(*L.14*)	NO(*L.14*)	OK(*L.16*)	OK(*L.16*)	fair k-bounded	NO(*L.14*)	NO(*L.14*)	OK(*L.16*)	OK(*L.16*)
NO(*L.14*)	NO(*L.14*)	OK(*L.16*)	OK(*L.16*)	fair bounded reg.	NO(*L.14*)	NO(*L.14*)	OK(*L.16*)	OK(*L.16*)
NO(L.14)	NO(*L.14*)	OK(*L.16*)	OK(*L.16*)	fair centr. bounded reg.	**NO(L.14)**	**NO(L.14)**	OK(*L.16*)	OK(*L.16*)

Table 3 Summary: weak gathering; crash model.

$f=1$				**Scheduler**	$f\geq 2$			
multiplicity		without multiplicity			multiplicity		without multiplicity	
determ.	proba.	determ.	proba.		determ.	proba.	determ.	proba.
OK(L.4)	**OK(L.19)**	NO(*L.3*)	NO(*L.12*)	unfair[a]	**NO(N. 1)**	**OK(L.19)**	NO(*L.17*)	NO(*L.17*)
OK(L.18)	OK(*L.19*)	NO(*L.9*)	NO(*L.12*)	unfair centr.[a]	**OK(L.18)**	OK(*L.19*)	NO(*L.17*)	NO(*L.17*)
OK(*L.4*)[b]	OK(*L.4*)	NO(*L.3*)	NO(*L.12*)	fair	**NO(N. 1)**	OK(*L.19*)	NO(*L.17*)	NO(*L.17*)
OK(*L.4*)	OK(*L.4*)	NO(*L.9*)	NO(*L.12*)	fair centr.	**OK(L.18)**	OK(*L.18*)	NO(*L.17*)	NO(*L.17*)
OK(*L.4*)	OK(*L.4*)	NO(*L.9*)	OK(*L.16*)	fair bounded	**NO(N. 1)**	OK(*L.19*)	NO(*L.17*)	NO(*L.17*)
OK(*L.4*)	OK(*L.4*)	NO(*L.9*)	OK(*L.16*)	fair k-bounded	**NO(N. 1)**	OK(*L.19*)	NO(*L.17*)	NO(*L.17*)
OK(*L.4*)	OK(*L.4*)	NO(*L.9*)	OK(*L.16*)	fair bounded reg.	**NO(N. 1)**	OK(*L.19*)	NO(*L.17*)	NO(*L.17*)
OK(*L.4*)	OK(*L.4*)	NO(*L.9*)	OK(*L.16*)	fair centr. bounded reg.	OK(*L.18*)	OK(*L.18*)	**NO(L.17)**	**NO(L.17)**

[a] The unfair behavior of the scheduler is restricted only to correct processes, otherwise the impossibility result follows trivially (the scheduler chooses only the crashed process).

[b] Note that the results derived from Lemma 4 hold for the case (3,1). According to Note 1 in the case (n,1) weak gathering is possible only if during the execution each configuration has at most one multiplicity point. Therefore, the self-stabilizing (n,1) weak-gathering is impossible since the initial configuration can contain more than one multiplicity point.

82, DOI http://dx.doi.org/10.1137/050645221

Ando H, Oasa Y, Suzuki I, Yamashita M (1999) Distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Trans on Robotics and Automation 15(5):818–828

Cohen R, Peleg D (2006) Convergence of autonomous mobile robots with inaccurate sensors and movements. In: Durand B, Thomas W (eds) 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS'06), Springer, Marseille, France, LNCS, vol 3884, pp 549–560

Défago X, Gradinariu M, Messika S, Raipin-Parvédy P (2006) Fault-tolerant and self-stabilizing mobile robots gathering: Feasibility study. DISC'06 pp 46–60

Dolev S (2000) Self-Stabilization. MIT Press

Flocchini P, Prencipe G, Santoro N, Widmayer P (2005) Gathering of asynchronous mobile robots with limited visibility. Theor Comput Sci 337:147–168

Lynch NA (1996) Distributed Algorithms. Morgan Kaufmann, San Francisco, CA, USA

Lynch NA, Segala R, Vaandrager FW (2003) Hybrid I/O automata. Information and Computation 185(1):105–157

Prencipe G (2001) CORDA: Distributed coordination of a set of autonomous mobile robots. In: Proc. 4th European Research Seminar on Advances in Distributed Systems (ERSADS'01), Bertinoro, Italy, pp 185–190

Prencipe G (2005) On the feasibility of gathering by autonomous mobile robots. In: Pelc A, Raynal M (eds) Proc. Structural Information and Communication Complexity, 12th Intl Coll., SIROCCO 2005, Springer, Mont Saint-Michel, France, LNCS, vol 3499, pp 246–261

Souissi S, Défago X, Yamashita M (2009) Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. ACM Trans Autonomous and Adaptive Systems 4(1):9:1–27

Table 4 Summary: weak gathering; Byzantine model.

	multiplicity; deterministic				
	$f=1$			$f\geq 2$	
Scheduler	$n=3$	$n\geq 4$ (even)	$n\geq 4$ (odd)	$n\geq 4$ (even)	$n\geq 4$ (odd)
unfair	NO(*L.6*)+NO(*L.5*)	NO(*L.6*)	NO(*L.6*)	NO(*L.6*)	NO(*L.6*)
unfair centr.	NO(*L.6*)+NO(*L.5*)	NO(*L.22*)	**OK(N.3)**	NO(*L.22*)	NO(*L.23*)
fair	NO(*L.6*) + NO(*L.5*)	NO(*L.6*)	NO(*L.6*)	NO(*L.6*)	NO(*L.6*)
fair centr.	NO(*L.6*)+NO(*L.5*)	NO(*L.22*)	OK(*N.3*)	NO(*L.22*)	NO(*L.23*)
fair bounded	NO(*L.6*)+NO(*L.5*)	NO(*L.22*)		NO(*L.22*)	NO(*L.23*)
$(k\geq n-1)$-bounded	NO(*L.6*)+NO(*L.5*)	NO(*L.22*)		NO(*L.22*)	NO(*L.23*)
$(\Gamma(n,f)\leq k\leq n-2)$-bounded	NO(*L.6*)+NO(*L.5*)	**OK(C.4)**		NO(*L.22*)	NO(*L.23*)
$(2\leq k<\Gamma(n,f))$-bounded	NO(*L.6*)+NO(*L.5*)	**OK(C.4)**			
fair bounded reg.	NO(*L.6*)+NO(*L.5*)	OK(*C.4*)			
centr. $(k\geq n-1)$-bound.	NO(*L.6*)+NO(*L.5*)	**NO(L.22)**	OK(*N.3*)	NO(*L.22*)	NO(*L.23*)
centr. $(\Gamma(n,f)\leq k\leq n-2)$-bound.	NO(*L.6*)+NO(*L.5*)	OK(*C.4*)	OK(*N.3*)	**NO(L.22)**	**NO(L.23)**
centr. $(2\leq k<\Gamma(n,f))$-bound.	NO(*L.6*)+NO(*L.5*)	OK(*C.4*)	OK(*N.3*)		
fair centr. bounded reg.	NO(*L.6*)+NO(*L.5*)	OK(*C.4*)	OK(*N.3*)		
fully synchronized	**OK(L.7)**	**OK(L.8)**	**OK(L.8)**	**OK(L.8)** if $n\geq 3f+1$	

$\Gamma(n,f)=\left\lceil\frac{n-f}{f}\right\rceil$ if n even ; $\Gamma(n,f)=\left\lceil\frac{n-f}{f-1}\right\rceil$ if n odd.

The unfair behavior of the scheduler is limited to correct processes only, or else the impossibility would be trivial.

Suzuki I, Yamashita M (1999) Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal of Computing 28(4):1347–1363