Fault and Byzantine Tolerant Self-stabilizing Mobile Robots Gathering

Xavier Défago — *School of Information Science, JAIST*

Maria Gradinariu Potop-Butucaru — *LIP6, UPMC, Paris 6, France*

Julien Clément, Stéphane Messika — *LRI/Université Paris Sud, France*

Philippe Raipin-Parvédy — *France Telecom R&D, France*

February 20, 2015

IS-RR-2015-003

Japan Advanced Institute of Science and Technology (JAIST)

School of Information Science
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

http://www.jaist.ac.jp/

ISSN 0918-7553

Fault and Byzantine Tolerant Self-stabilizing Mobile Robots Gathering

A Feasibility Study

Xavier Défago
JAIST, Japan

Maria Gradinariu Potop-Butucaru
LIP6, UPMC, Paris 6, France

Julien Clément, Stéphane Messika
LRI/Université Paris Sud, France

Philippe Raipin-Parvédy
France Telecom R&D, France

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 specific 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. In addition, we extend our study to the feasibility of probabilistic self-stabilizing gathering in both fault-free and fault-prone environments.

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 [19], 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 [16], activations are completely asynchronous, for instance allowing robots to be seen while moving. Flocchini *et al.* [10] provide an excellent overview on the subject.

Suzuki and Yamashita [19] 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 [17] 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.* [11] 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.* [18] considered a system in which compasses are not necessarily consistent initially. Ando *et al.* [2] proposed a gathering algorithm for the SYm model with limited visibility. Cohen and Peleg [6] studied 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 [1]. They

This manuscript considerably extends preliminary results presented as an extended abstract at the DISC conference [7].

[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.

proved that gathering of correct robots (called *weak gathering* in this paper) can be achieved in the SYm model even in the face of the crash of a single robot. Furthermore, they proved that no deterministic gathering algorithm exists in the SYm model that can tolerate a Byzantine[2] robot. Finally, they considered a stronger model, called *fully synchronous*, in which all robots are always activated simultaneously, and showed 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, to some extent, Byzantine) environments. In particular, we consider centralized schedulers[3] (i.e., activations occur in mutual exclusion) and bounded schedulers (i.e., between any two consecutive activations of a robot, no other robot is activated more than k-times for some finite k).

More specifically, we obtain the following important results.

Firstly, we strengthen an important impossibility result of Prencipe [17] by showing that it also holds in strictly stronger models. In particular, in oblivious fault-free environments without multiplicity, Prencipe [17] proved the impossibility of *distinct*[4] gathering *under a fair scheduler*. We considerably strengthen this result by proving that the same problem remains impossible under more restrictive schedulers, even under *a 2-bounded centralized scheduler*. We further prove that the problem of *self-stabilizing* gathering is impossible even under a *round-robin scheduler*, and this is also conjectured for distinct gathering.

Secondly, still without multiplicity, we prove that self-stabilizing gathering can be solved probabilistically under a fair bounded scheduler (with arbitrary by finite bound) when $n \geq 3$ and under an unfair scheduler when $n = 2$, by exhibiting a simple algorithm that solves the problem.

Thirdly, given multiplicity, we prove that gathering can be solved *deterministically* under a *fair centralized scheduler* even if up to $n-1$ robots can crash. We then extend the algorithm to prove that gathering can also be solved *probabilistically* even if the scheduler is not centralized.

Fourthly, we study the case of Byzantine-tolerance by extending the range of impossibility results. Most notably, Agmon and Peleg [1] proved that $(3,1)$-Byzantine gathering is impossible *deterministically* under a *fair scheduler*. We extend the result by showing that even *probabilistic* gathering is impossible under a *round-robin scheduler*. We also prove other impossibility results.

More generally, we show in what situations randomized algorithms can help solve the problem, and when they cannot. To the best of our knowledge our work[5] was the first to investigate the feasibility of probabilistic gathering in both fault-free and fault-prone systems.

[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] *Distinct gathering* is a tighter definition of the gathering problem, in which robots are required to have distinct positions initially. In contrast, *self-stabilizing gathering* puts no such requirements on initial configurations.

[5] An extended abstract of this work was presented at DISC [7] in 2006, although it has been considerably extended since. Meanwhile, some authors have published very insightful results on the problem [12].

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. The model we consider is based on the SYm model [19], and most definitions are due to various authors [19, 16, 1].

2.1 Robot network

A robot network consists of a finite set $\mathbf{R} = \{r_1, \cdots, r_n\}$ of n dimensionless robots evolving in a boundless 2D Euclidean space, devoid of any landmarks or obstacles.

Robots cannot communicate with each other and do not share any notion of a global coordinate system. In particular, they have no agreement on a common origin, unit distance, or directions and orientations of the axis.

2.2 Robot

A robot is modeled as an I/O automaton[6] ([13]).

Robots are *oblivious* which means that they do not retain any information on past actions and observations. The state of a robot consists only of its current position in the environment, which is neither directly readable[7] nor directly writable[8] by the robot's algorithm.

Robots are *anonymous* in that they are not aware of any distinctive identity and all of them execute the same algorithm consisting of cycles of the operations: Observe, Compute, Move. In the SYm model, the three operations are executed atomically. Thus, for simplicity, an algorithm is expressed as one or more *Observe* input actions with effects Compute and Move, and guarded by a possible precondition.

$$\underbrace{\mathit{Observe}(\Pi)}_{\text{input}} :: \langle \text{precondition} \rangle \longrightarrow \underbrace{\langle \text{compute} \rangle ; \langle \text{move} \rangle}_{\text{effect}}$$

In this paper, actions being always enabled, the precondition is always set to true.

- *Observe (input action).*
  The parameter to the action is a set $\mathbf{P}$ or multiset Π of points representing the positions occupied by all robots, as expressed in the private coordinate system of the robot making the observation. The origin of the private coordinate system corresponds to the current position of the robot with arbitrary unit distance and orientation.

[6] In the CORDA model [16], a robot exhibits a continuous behavior that can be modeled by an hybrid I/O automaton ([14]).

[7] The current position is exclusively available in local coordinates.

[8] A robot can change its position only through move operations.

When the system is said to be *with multiplicity*,[9] The observation is a multiset Π of points and the multiplicity of an element in Π corresponds to the number of robots sharing that location. Conversely, when the system is said to be *without multiplicity*, then the observation is a set $\mathbf{P}$.

In this paper, robots are assumed to have *unlimited visibility*, in that all robots are part of each other's observation regardless of their respective distance.

- *Compute*.
  A stateless computation returning a target destination in the private coordinate system.

  If the algorithm is deterministic, the computation is deterministic and depends only on the observation $\mathbf{P}$ (or Π). In contrast, if the algorithm is probabilistic, the output may additionally depend on random choices.

- *Move (effect)*.
  Directs the actual motion of the robot toward a designated target destination.

  The robot may or may not reach this destination. For every robot r, there exists a *reachable distance* $\delta_r > 0$ unknown to r, such that, any target destination computed within a distance δ_r from the current position is reached in that step. Conversely, if the target is not reachable, then r travels at least a distance δ_r. This condition is necessary to ensure progress.

  We denote by $\delta = \min_r \delta_r$ the minimal reachable distance. We often use δ in place of each individual δ_r for simplicity, but only as a worst case choice.

  When not explicitly specified, the trajectory of the robot is assumed to be a straight line to the destination.

2.3 Activations and schedulers

A scheduler decides, for every configuration, which subset of the robots is active (i.e., allowed to perform their actions). In this paper we consider the following schedulers:

- *unfair arbitrary*: At each activation, a non-empty subset of robots is activated. A non-triviality condition ensures that, infinitely often, a non-faulty robot becomes active.
- *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 arbitrary*: 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 k-bounded*: The scheduler is fair with the additional guarantee that there exists some bound k such that between any two consecutive activations of some robot, no other robot is activated more than k times. The bound may be known or unknown to the robots. In the sequel we assume that robots do not know the scheduler bound.
- *round-robin*: The scheduler is fair 1-bounded and centralized. This implies that the robots are activated always in the same sequence.
- *fully synchronized*: Every robot is active at every activation.

[9] Our definition of multiplicity is sometimes called "strong multiplicity", in contrast to a weaker definition where robots are only able to distinguish whether a given location is occupied by one or by several robots [12].

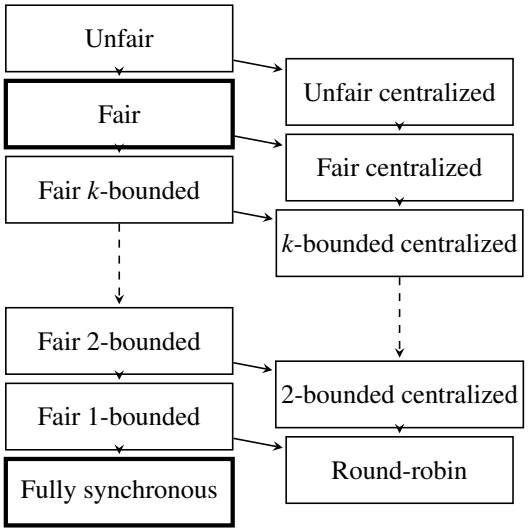


Figure 1: Relationships between scheduler classes. Conventional models are highlighted: SYm [19] and CORDA [16] are fair, and the fully synchronous model [1] is its namesake.

Figure 1 summarizes 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.

2.4 Executions and configurations

A *configuration* is the union of the local states of the robots in the system at some discrete time t. An *execution* $e = (\gamma_0, \ldots, \gamma_t, \ldots)$ of the system is a sequence (finite or infinite) of configurations, where γ_0 is an initial configuration of the system, and every transition $\gamma_t \to \gamma_{t+1}$ corresponds to the activation of a subset of the robots, according to the scheduler. An *execution fragment* is any non-empty subsequence of an execution.

The *valence* of a configuration γ denotes the number of distinct locations occupied by some robot in γ. Thus, a *q-valent* configuration has q distinct locations (where $1 \leq q \leq n$ is the valence and n the number of robots in the system).

A *univalent* configuration is a configuration in which all robots share the same location (valence 1). A univalent configuration γ is said to be *centered at p* if p is the location occupied by the robots in γ.

A *multivalent* configuration is a configuration that is not univalent ($q > 1$).

A *bivalent* configuration is a multivalent configuration with valence 2.

A *1-bivalent* configuration is a bivalent configuration in which one of the two locations is occupied by a single robot.

A *distinct* configuration is a configuration in which all robots have distinct positions (valence n).

2.5 Fault models

The behavior of a *correct* robot never deviates from its specification. In contrast, a robot is considered *faulty* if its behavior deviates from its specification in some executions. In this paper, we consider two classes of faults: crash and Byzantine.

(n, f)-crash model: The system consists of n robots, among which up to f faulty robots may fail by crashing. To rule out the trivial case, $f < n$, so there is at least one correct robot.

A crash may occur at any time. A robot that crashes permanently stops performing any action. In particular, it no longer moves from the position it crashed. A crash cannot be detected by other robots.

(n, f)-Byzantine model: The system consists of n robots, among which up to $f < n$ faulty robots may exhibit an arbitrary behavior.

Byzantine robots are controlled by an adversary. The activations of Byzantine robots are subject to the restrictions imposed by the scheduler. The behavior of the Byzantine robots can however be based on a global awareness of the environment, including all past actions and the current state of all robots.

Since a Byzantine robot may elect to stop performing actions, the Byzantine model is a strict generalization of the crash model.

2.6 Computational Models

The literature proposes mainly two computational models, namely, SYm and CORDA. The SYm model was introduced by Suzuki and Yamashita [19]. In this model each robot performs, once activated by the scheduler, a *computation cycle* consisting 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 CORDA model, introduced by Prencipe [16], refines the atomicity of actions, by decoupling observe and move actions, as well as separate the beginning and the end of a move as distinct events. Robots may be interrupted by the scheduler halfway through a computation cycle. Moreover, while a robot performs an observation, another robot may be partway through a movement.

As stated before, in this paper we consider the SYm model,[10] refined with the above scheduling strategies. We focus our study on the case of oblivious robots, i.e., robots do not conserve any information between two computational cycles. A major motivation for considering oblivious robots is that, as observed by Suzuki and Yamashita [19], algorithms designed for that model are inherently self-stabilizing [9].

[10]Note that all impossibility results proven in the SYm model necessarily hold in the CORDA model.

2.7 Notation

Let γ be a configuration, then $val(\gamma)$ denotes the valence of configuration γ.

Let Π be a multiset of points representing the locations of robots in configuration γ, and let p be a location in Π. Then, $mul(p)$ is the multiplicity of point p and corresponds to the number of robots located at p in configuration γ.

The maximal multiplicity $\mu(\Pi)$ (resp. $\mu(\gamma)$) of a multiset (resp. configuration) is $\mu(\Pi) = \max_{p \in \Pi}(mul(p))$.

We now define the set of points with maximal multiplicity as $MaxMult(\Pi) = \{p \in \Pi \mid mul(p) = \mu(\Pi)\}$. A point in $MaxMult(\gamma)$ is called a point of maximal multiplicity.

For convenience, we introduce the following additional terminology. A *tower* is a location occupied by at least two robots. A *castle* is a tower with maximal multiplicity.

2.8 Geometry Definitions

Given a set of points $\mathbf{P}$, we have the following definitions.

Convex Hull: The *convex hull*, denoted $Conv(\mathbf{P})$, is defined as the smallest convex set that contains $\mathbf{P}$. The convex hull is unique. A point p in $\mathbf{P}$ is a vertex of the convex hull if and only if p is outside of $Conv(\mathbf{P} \setminus \{p\})$.

Smallest Enclosing Circle: The *smallest enclosing circle*, denoted $SEC(\mathbf{P})$, is defined as the smallest circle that contains all point in $\mathbf{P}$. It is unique and can be computed in linear time [15]. It is defined either by two points which form a diameter, or by three or more points located on its circumference and forming no angle greater than π. Any point in $\mathbf{P}$ on the circumference of $SEC(\mathbf{P})$ is also a vertex of the convex hull. The diameter of $SEC(\mathbf{P})$ provides an upper bound on the distance between any pair of points in $\mathbf{P}$.

Voronoi Diagram: The *Voronoi diagram* $Voronoi(\mathbf{P})$ is a division of the space into cells, one for each point in $\mathbf{P}$, such that the Voronoi cell $Vcell(p)$ of point p contains all points whose distance to p is smaller or equal to its distance to any other points in $\mathbf{P}$. The Voronoi diagram is unique and maps the entire space. All Voronoi cells are convex polygons. Given a point p in $\mathbf{P}$, $Vcell(p)$ has vertex at infinity if and only if p is a vertex of the convex hull $Conv(\mathbf{P})$.

3 The Self-Stabilizing Gathering Problem

In the gathering problem, robots are required to eventually reach a configuration in which they all share the same location. There are several variants to the problem.

3.1 Strong gathering

We define the *self-stabilizing strong gathering* problem as follows.

Convergence: Any execution starting in an arbitrary configuration reaches a univalent configuration after a finite number of steps.

Closure: Any execution suffix that starts in a univalent configuration contains only univalent configurations.

The problem is called *point formation* with an equivalent definition by Suzuki and Yamashita [19].

Note 1 *Other authors, such as Prencipe [17], define gathering as the problem of reaching a univalent configuration when starting from any* distinct *configuration rather than arbitrary ones. Let us call that definition "distinct gathering." Distinct gathering is however not self-stabilizing because, solving the problem with oblivious robots does not readily make the algorithm self-stabilizing.*

Distinct gathering is covered by self-stabilizing gathering. In other words, an algorithm that solves self-stabilizing gathering also solves distinct gathering. Conversely, if distinct gathering is impossible in a given system, then self-stabilizing is also impossible in that system.

In the paper, we consider the self-stabilizing definition, except in Section 4 when we extend impossibility results that were originally proved for distinct gathering.

3.2 Weak gathering

The definition of strong gathering and univalent does not distinguish between correct robots and faulty ones. In fault-tolerant contexts, a weaker definition of the problem is often desirable.

Let us define a *gathered configuration* as a configuration in which all *correct* robots are located at a unique point of maximal multiplicity.

Convergence: Any execution starting in an arbitrary configuration reaches a *gathered* configuration after a finite number of steps.

Closure: Any execution suffix that starts in a *gathered* configuration contains only *gathered* configurations.

In a fault-free system, univalent and gathered configurations are identical. Consequently, the distinction between strong and weak gathering is irrelevant in that context.

3.3 Convergence

Gathering is difficult to achieve in most environments. And thus, 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. Convergence is however considerably easier to deal with. For instance, with unlimited visibility, it can be achieved trivially by having robots moving toward the barycenter of the network [19].

3.4 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 [19] and refer to oblivious robots under a fair scheduler.

Theorem 1 ([19]; Th. 3.1) *There is no deterministic algorithm that solves gathering for $n = 2$ robots under a fair scheduler.*

Notice that, although the above theorem 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 scheduler with the bound $k = 1$. It follows that the result also applies to a system based on that scheduler.

Theorem 2 ([19]; Th. 3.4) *Gathering of $n \geq 3$ robots can be solved deterministically under a fair scheduler with multiplicity detection.*

The next theorem, proved by Prencipe [17], considers distinct gathering (i.e., gathering starting from any *distinct* configuration) and also applies to oblivious robots under a fair scheduler.

Theorem 3 ([17]; Th. 2) *Under a fair scheduler, There is no deterministic algorithm that solves distinct gathering for $n \geq 2$ robots without additional assumptions (e.g., multiplicity detection).*

Finally, the following two theorems, proved by Agmon and Peleg [1], refer to models with the presence of faulty robots. These theorems state positive results.

Theorem 4 ([1]; Th. 3.5) *Weak gathering can be solved deterministically in a $(3,1)$-crash model under a fair scheduler with multiplicity detection.*

Note 2 *Agmon and Peleg [1] also show (Th. 3.8) that weak gathering can be solved by a deterministic algorithm in an $(n,1)$-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.*

When $n = 3$, there cannot be more than one point of multiplicity, so this is not an issue. But, for $n > 3$, although their algorithm does solve the distinct *gathering problem, it fails to solve self-stabilizing gathering. The definition of the latter problem indeed requires that any configuration leads to gathering, including any one with several points of multiplicity.*

They also present also two highly relevant results relating to Byzantine models.

Theorem 5 ([1]; Th. 4.4) *There is no deterministic algorithm that solves weak gathering in a $(3,1)$-Byzantine model under a fair scheduler.*

In contrast, they state a positive result in the fully-synchronous model—a model in which all robots are activated at every step.

Theorem 6 ([1]; Th. 5.3) *Weak gathering can be solved deterministically in a $(3,1)$-Byzantine system in the fully-synchronous model.*

Theorem 7 ([1]; Th. 5.10) *Weak gathering can be solved deterministically in an (n,f)-Byzantine system in the fully-synchronous model for any $n \geq 3f+1$.*

Theorem 8 ([8]; Th. 1) *With strong multiplicity detection, there exists a deterministic algorithm solving self-stabilizing gathering in the semi-synchronous model for n robots if, and only if, n is odd.*

The following theorem synthetizes the recent results related to the probabilistic gathering under various multiplicity conditions. In particular, [12], introduces the notions of local-weak and local-strong multiplicity. Local multiplicity means that a robot is able to detect the multiplicity only for its current position. Local-weak multiplicity means that a robot can detect if at its local position there are one or more than one robots. Local-strong multiplicity means that a robot can detect the exact number of robots at its location.

Theorem 9 ([12]) *Probabilistic self-stabilizing gathering is possible in constant expected time with local-strong multiplicity and exponential expected time with local-weak multiplicity. Probabilistic distinct gathering is possible in constant expected time with local-weak multiplicity.*

The next result states the possibility of wait-free[11] distinct gathering (i.e., the initial configuration must exclude balanced bivalent configurations) in the semi-synchronous model, when robots have strong multiplicity detection and chirality knowledge.

Theorem 10 ([3]) *In the semi-synchronous model, wait-free gathering is possible with fair scheduler, under the following assumptions: chirality knowledge and strong multiplicity detection.*

The following results refer to the possibility and impossibility of convergence and, by consequence, of gathering, when some robots in the system have Byzantine behavior.

Theorem 11 ([5]) *Byzantine-resilient convergence in one-dimensional robot networks is impossible under a fully-synchronous scheduler when $n \leq 2f$.*

Theorem 12 ([5]) *Byzantine-resilient convergence In one-dimentional robot networks is impossible under a fair k-bounded scheduler ($k > 1$) when $n \leq 3f$.*

Theorem 13 ([4]) *Starting from a trivalent configuration, no cautious algorithm is able to achieve byzantine-resilient convergence in uni-dimensional networks under an asynchronous scheduler when $3f < n \leq 5f$.*

4 Gathering in Fault-Free Environments

In this section, we refine results showing the impossibility of gathering [17, 1] by proving first that these results hold even under more restrictive schedulers. 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 bounded scheduler.

First, we introduce two support lemmas that apply to any gathering algorithm (deterministic or probabilistic) under any form of centralized scheduler.

[11] An algorithm is said to be wait-free if it tolerates the crash of up to $n-1$ robots.

Lemma 1 *Under a centralized scheduler and in any execution, the valence of two consecutive configurations differs by at most one.*

PROOF: The scheduler being centralized, at most one robot is active at each step. Regardless of the algorithm, the movement of the active robot falls into one of three categories, depending on the respective multiplicities of the departure and destination locations of the movement:

Move 1: distinct $\rightarrow$ multiple.
The valence decreases by one.

Move 2: multiple $\rightarrow$ multiple or distinct $\rightarrow$ distinct.
The valence is unchanged.

Move 3: multiple $\rightarrow$ distinct.
The valence increases by one.

Therefore, when the scheduler is centralized, the valence between any two consecutive configurations differs by at most one. □

Lemma 2 *Under a centralized scheduler, every execution fragment that starts in a multivalent configuration and ends in a univalent configuration contains a 1-bivalent configuration.*

PROOF: By Lemma 1 and the centralized scheduler, we know that the valence between any two consecutive configurations differs by at most one. Since the execution fragment ends in a univalent configuration, the last multivalent configuration in the fragment must be bivalent. This configuration necessarily exists since the fragment starts in a multivalent configuration.

Furthermore, since only one robot moves between any two configurations (centralized scheduler), the last bivalent configuration is 1-bivalent with the distinct robot doing the last move. □

4.1 Deterministic Gathering

We begin by proving a theorem that strengthen the impossibility result of Prencipe [17] (Lemma 3), as applied to the problem of self-stabilizing gathering. The theorem proves that the impossibility not only holds under a fair scheduler, but also under a round-robin scheduler.

Theorem 14 *Under a round-robin scheduler, there is no deterministic algorithm that solves self-stabilizing gathering for $n \geq 3$, without additional assumptions (e.g., multiplicity knowledge).*

PROOF: Assume, by contradiction, that such an algorithm exists. Let $\mathscr{A}$ be a deterministic algorithm that solves (distinct) gathering under a round-robin scheduler.

Without loss of generality, let the reachable distance of the robots be so large that the robots can reach each others' location in a single step.

Consider the initial configuration Γ_1 as described below (see Fig 2). Γ_1 is a 1-bivalent configuration such that all robots are at one location, except one robot, r_1, which is at a distinct location.

Consider an execution e under algorithm $\mathscr{A}$ that starts in Γ_1 and follows the round-robin activation schedule given by the

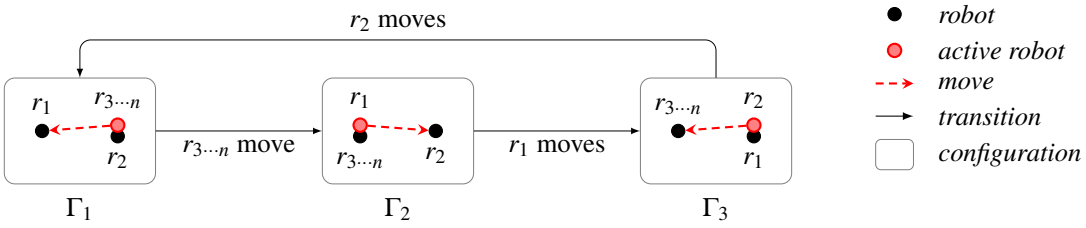


Figure 2: Proof of Theorem 14: From a 1-bivalent configuration Γ_1 with robot r_1 in a distinct location, an activation schedule $r_{3\cdots n}, r_1, r_2, r_{3\cdots n}, \ldots$ generates a cycle of equivalent configurations. $r_{3\cdots n}$ represent the remaining robots r_3 to r_n.

sequence $\sigma = r_3, \cdots, r_n, r_1, r_2$. The application of σ to configuration Γ_1 leads to a cycle of bivalent configurations, as illustrated in Figure 2.

Therefore, a univalent configuration is never reached in execution e, which contradicts the fact that every execution under algorithm $\mathscr{A}$ satisfies the Convergence property.

Thus, algorithm $\mathscr{A}$ does not exist. □

The impossibility of Lemma 3 is for the distinct gathering problem, namely when initial configurations are restricted to distinct ones.

In order to extend that result, we must prove that the impossibility also holds under the same conditions, namely, starting from distinct configurations. As stated earlier, an impossibility for distinct gathering implies an impossibility for self-stabilizing gathering.

We introduce an additional theorem below, which stricto senso extends the impossibility of Lemma 3. We show that, under a $k \geq 2$-bounded scheduler, *every* multivalent configuration (this includes every distinct configuration) can lead to a non-terminating execution. In other words, this proves the impossibility of distinct gathering.

Theorem 15 *Under a centralized $k \geq 2$-bounded scheduler, there is no deterministic algorithm that solves distinct gathering for $n \geq 3$, without additional assumptions (e.g., multiplicity knowledge).*

PROOF: With no loss of generality, assume that the scheduler is 2-bounded since this is the most restrictive case for the adversary given the hypotheses of the lemma.

Let the adversary select an arbitrary sequence σ of the robots and activate them according to a round-robin policy over σ. The scheduler is centralized, so Lemma 2 holds and thus, from any initial multivalent configuration, in particular any distinct configuration, if an algorithm exists, it must necessarily lead the system to a 1-bivalent configuration. Let γ_x be this 1-bivalent configuration and let r denote the distinct robot.

If robot r is not the next robot in σ, then continue the activations in a round-robin fashion until another 1-bivalent configuration is reached, and repeat the argument.

If robot r is the next robot in σ, then apply the following permutation. Let r' be the robot in the one-before-last position in σ and r'' the robot at the last position. The adversary updates σ by letting r swap positions with r'. This leads to the configuration Γ_1 depicted in Fig. 2, where $r_1 = r$, $r_2 = r''$, and $r_3 = r'$, and the cycle follows.

The swap is valid under the 2-bounded scheduler because no further swap occurs, and no robot is activated more than twice between the last activation of r before the swap and the first one after. □

We strongly believe that the impossibility under a round-robin scheduler applies not only to self-stabilizing gathering, but also to distinct gathering. We state the following conjecture under which the impossibility holds.

Conjecture 1 *Given a system with no additional assumptions (e.g, multiplicity knowledge). For every gathering algorithm under a round robin scheduler, there exists an execution starting in a distinct configuration such that a gathered configuration is reached by activating exactly once, every robot except one.*

To substantiate why the claim might be true, let us consider some examples.

First, say that the criteria applied by the algorithm is to select the location of the nearest robot. Then, one distinct configuration that meets the requirement of the lemma is to place a first robot r_1, and then all other robots such that their distance to r_1 follows a geometric progression. Activating each robots except r_1 in the order they were placed let them gather at r_1.

Second, if the criteria is to select the farthest robot, then gathering is obtained from placing the robots along a line and activating them from one extremity to the next.

Third, if the criteria is to select a robot near a centroid, then by placing robots along the circumference of a circle centered at r_1 and an interleaved activation of the robots, r_1 can still remain near the centroid until the system reaches a bivalent configuration.

This is not exhaustive, and more complex criteria can be made to change depending on the valence of the observation.

Theorem 16 *Under the hypothesis that Conjecture 1 holds, There is no deterministic algorithm that solves distinct gathering for $n \geq 3$ under a round-robin scheduler, without additional assumptions (e.g., multiplicity knowledge).*

PROOF: Assume, by contradiction, that such an algorithm exists. Let $\mathscr{A}$ be a deterministic algorithm that solves distinct gathering under a round-robin scheduler.

Let the reachable distance of the robots be so large that the robots can reach each others' location in a single step.

By assumption, Conjecture 1 holds and there exists a configuration Γ_0 such that, by activating every robot at most once, an execution e starting in Γ_0 reaches a univalent configuration. Let

us name the robots such that the successful activation sequence is $r_{3\cdots n}, r_2, r_1$.

Since a robot moves only once, all robots must select the location of r_1 as their target, and the configuration after activating robots $r_{3\cdots n}$ is 1-bivalent with r_2 at the distinct location.

Consider execution e' with the same prefix, but where r_1 is activated before r_2. Without multiplicity, what r_1 observes in e' is the same as what r_2 observes in e. Therefore, r_1 moves to the location of r_2, leading to configuration Γ_3 of Figure 2. And the rest follows. □

Consider now the case when the system consists of two robots. Suzuki and Yamashita [19] have proved that the deterministic gathering of two oblivious robots is impossible under a fair scheduler (Lemma 1). The simple lemma below shows that 2-gathering is however possible when the scheduler is centralized.

Lemma 3 *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 other robot. Given that the scheduler is centralized, at each step only one of the two robots, say r_1, is active.

If r_2 is reachable from r_1, then gathering is achieved in that step. If r_2 is not reachable from r_1, then the distance between both robots decreases by δ_{r_1}.

Thus, by repeating the argument, we see that the distance between the robots decreases monotonically, until they become reachable and then gathering is achieved in the next activation. □

Note that, in the above proof, it does not matter which robot is activated in each round. In particular, even if the scheduler is unfair, it must activate either one of the two robots.

4.2 Probabilistic Gathering

We now look at the case of probabilistic algorithms in a fault-free environment. In the following, we prove that, for the case of two robots, there exists a probabilistic solution for gathering in the SYm model, under any type of scheduler.

Algorithm 4.1 Probabilistic gathering for robot p.

Actions:

Observe(P) :: *true* $\longrightarrow$
  with probability $\alpha = \frac{1}{|P|}$ **do**
    select location $q \in P$ uniformly;
    move towards q;
  else
    stay;

Algorithm 4.1 describes the probabilistic strategy of a robot. When a robot becomes active, it 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 univalent configuration in constant expected steps.

Lemma 4 *Algorithm 4.1 probabilistically solves gathering for $n = 2$ under an unfair scheduler.*

PROOF: Consider two robots r_1 and r_2, and an arbitrary initial configuration γ_0. If r_1 and r_2 are already gathered, the configuration is univalent and neither will move, regardless of activations and the probability α.

Since there are two robots, every non-gathered configuration is bivalent, and thus $\alpha = \frac{1}{2}$.

Assume that both robots have the same reachable distance δ (or, if they are different, define δ conservatively as their minimum).

Let us show how r_1 and r_2 reach a configuration in which they are mutually reachable, from one in which they are not. Let $D_0 > \delta$ be the initial distance between r_1 and r_2. At each successful move of either one of the robots, the distance between them is decreased by at least δ (by 2δ if both move). Thus, it takes at most $x = \lceil \frac{D_0}{\delta} \rceil - 1$ successful moves of either one robot, for them to be within reachable distance. Since the scheduler must activate at least one of the robots, the probability of a successful move at each trial is at least $\frac{1}{2}$. The number of failures until the x^{th} success of a Bernoulli trial with success probability α is known to be a random variate that follows a negative binomial distribution $NB(x,\alpha)$. It follows that the expected number of steps until both robots are within reachable distance δ is at most

$$\begin{aligned}\mathbb{E}[\text{steps to reachable}] &\leq \lceil \frac{D_0}{\delta} \rceil - 1 + \mathbb{E}\left[NB\left(\lceil \frac{D_0}{\delta} \rceil - 1, \frac{1}{2}\right)\right] \\ &\leq 2\left(\left\lceil \frac{D_0}{\delta} \right\rceil - 1\right)\end{aligned}$$

Let us now consider gathering from a configuration in which the two robots are reachable from each other.

Consider some discrete time t when the two robots have distinct locations. If only one of the robots, say r_1, is activated by the scheduler, then there is a probability α that r_1 moves, and thus both robots end up gathered in the next configuration (terminal). If both robots are activated at time t, then they end up in a univalent configuration only if exactly one of them changes its position. This occurs with probability $2\alpha(1-\alpha)$.

Consequently, the probability to reach gathering during at time $t+1$ is at least $q = \min(\alpha, 2\alpha(1-\alpha)) = \frac{1}{2} > 0$, regardless of the choice of the scheduler. The number of failures before first success is a random variate that follows a geometric distribution $G(q)$. This yields the expected number of steps until gathering as

$$\mathbb{E}[\text{steps to gathering}] = 1 + \mathbb{E}[G(q)] = 1 + \frac{(1-q)}{q} = \frac{1}{q} = 2$$

Thus, gathering is achieved in at most $2\lceil \frac{D_0}{\delta} \rceil$ steps in expectation. □

The next lemma extends the impossibility result proved in Theorem 14 to probabilistic algorithms under a fair centralized scheduler.

Lemma 5 *There is no probabilistic algorithm that solves gathering for $n \geq 3$, under a fair centralized scheduler without additional assumptions (e.g., multiplicity knowledge).*

PROOF: A randomized algorithm can use randomization in two different ways. It can select random locations (case A), or it can toss a coin before doing a move (case B).

With respect to the first case, the proofs of Theorem 14 and 15 still stand when destinations are based on random choices, except when the random choice of a robot is to select its current location. This is however equivalent to tossing a coin and stay still with some probability, which is in turn equivalent to the second case.

Hence, we focus on the second case (case B) and represent a randomized algorithm as one in which an active robot tosses a coin and, with some positive probability α, executes an action (and stays still otherwise). Note that, if the probability depends on the robot, α can be defined as the minimum. It must be positive because, since Theorem 15 shows that no algorithm exists based on deterministic choices, a robot cannot set the probability to zero based only on its observations.

Consider an adversary that selects a robot r and activates r until the coin toss is successful, and r actually executes its action. Since α is positive, the activation is fair (albeit unbounded). By doing so, the adversary can actually "*derandomize*" the algorithm with the remainder of the proof being the same as for Theorem 15. □

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. 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 gathered configuration.

Theorem 17 *Algorithm 4.1 probabilistically solves gathering for $n \geq 3$, under a fair bounded scheduler and without multiplicity knowledge.*

PROOF: Let k denote the bound of the scheduler. The scheduler being fair, there are at most $k(n-1)$ steps between any two consecutive activations of any robot. Let δ be the reachable distance of the robots (or their minimum if they are different).

The probability α depends on the valence of the current configuration. However, in multivalent configurations, it is bounded as follows: $\frac{1}{n} = \alpha_{\min} \leq \alpha \leq \alpha_{\max} = \frac{1}{2}$.

For clarity, the proof has two parts. First, we show that, from an arbitrary configuration, the system reaches a configuration in which all robots are within reachable distance from each other. Second, we show that, with high probability, in a configuration where robots are reachable from each other, the valence of successive configurations decreases until gathering is reached.

Theorem 17; Part 1: from arbitrary to reachable. Consider the smallest enclosing circle SEC_t defined by the robots' locations in a configuration γ_t. By definition of the smallest enclosing circle, and because a circle is convex, all locations and all segments between them are either inside the circle or on its circumference. By Algorithm 4.1, a robot r selects a target r' among the robots' locations, r can move only to r' or to some point in the segment between them. Thus, r in γ_{t+1} is necessary enclosed by SEC_t. Thus, $SEC_{t+1} \subseteq SEC_t$. In other words, the smallest enclosing circle is non-increasing.

To show the convergence, we now show that, with some positive probability p, the diameter of the smallest enclosing circle decreases by at least δ, which is a constant positive value.

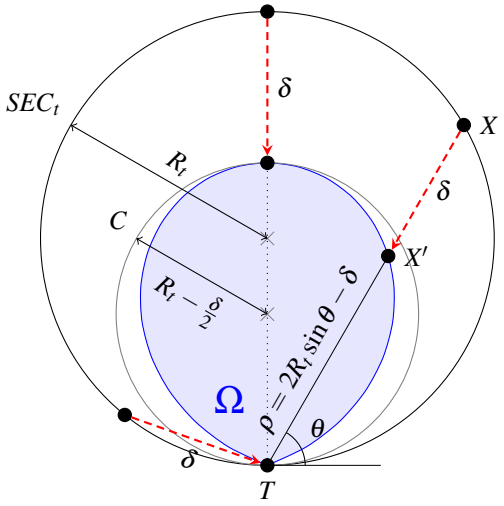


Figure 3: Proof of Theorem 17; Part 1: Strict decrease of the smallest enclosing circle SEC_t (of radius R_t) occurs with positive probability. After all robots select T as a target and move once (dashed lines), they are all contained inside area Ω, which is itself contained within a circle C of radius $R_t - \frac{\delta}{2}$. Positions are expressed in polar coordinates centered at T.

Let γ_t be a configuration and SEC_t the smallest enclosing circle in γ_t. Let T be the position of a robot located on the boundary of SEC_t.

Consider the situation in which all other robots take T as their target for a successful move and let γ_x be the resulting configuration (Fig. 3). We show that the smallest enclosing circle SEC_x in configuration γ_x is smaller than SEC_t in diameter by at least δ.

To characterize the movement of the robots, we consider polar coordinates (θ, ρ) centered at T. The smallest enclosing circle SEC_t is given by

$$\theta \in [0;\pi]\, \rho = 2R_t \sin\theta$$

Let Ω be the area that the robots will reach after moving toward T of a distance at least δ (Fig. 3). This area can be characterized as follows[12]

$$\theta \in \left[\arcsin\frac{\delta}{2}; \pi - \arcsin\frac{\delta}{2}\right] \rho = 2R_t \sin\theta - \delta$$

Let C be a circle with diameter $2R_t - \delta$ and anchored at T.

$$\theta \in [0;\pi]\, \rho = (2R_t - \delta)\sin\theta$$

Let us show that Ω is contained within C. When

$$\theta \in \left[\arcsin\frac{\delta}{2}; \pi - \arcsin\frac{\delta}{2}\right]$$

this holds if the following inequality always holds

$$\begin{aligned}(2R_t - \delta)\sin\theta &\geq 2R_t \sin\theta - \delta \\ -\delta\sin\theta &\geq -\delta \\ \sin\theta &\leq 1\end{aligned}$$

[12] Note that Ω is not a circle; it is best described as the inner loop of a limaon of Pascal.

which is always true. Since all robots are contained within C, it follows that SEC_x is contained within C, and thus its diameter is at most $2R_t - \delta$.

Notice that selecting T on the boundary is a worst case for convergence; the best case occurs when T is located near the center of SEC_t and the decrease in diameter then becomes 2δ.

Let us now show that, with positive probability p, *SEC* decreases by at least δ in diameter in a constant number of activation steps.

Note that we do not need to calculate the probability p accurately. It is sufficient to show that p has a positive lower bound. For this reason, we do not need to consider all cases.

Consider an execution fragment $e[t{:}t+K]$ of $K = nk$ successive configurations starting in γ_t. The scheduler is fair k-bounded so, regardless of its choices, every robot is activated at least once and at most nk times in fragment $e[t{:}t+K]$.

We need to calculate the probability that (1) every robot except one (at T) makes exactly one successful move toward T at first trial and takes no further move in the $K-1$ remaining steps, and (2) one robot (at T) makes no successful move in K steps.

For one of the robots (except one at T), the probability that it makes a successful move toward T at first trial is

$$\mathbb{P}[\text{1 robot moves to } T \text{ at first trial}] \geq \alpha_{\min}\frac{1}{n} = \frac{1}{n^2}$$

Assuming a worst case situation when the scheduler activates the robot every time after the move, the probability for that robot to take no successful move in $K-1$ steps is

$$\mathbb{P}[\text{1 robot stays for } K-1 \text{ steps}] \geq (1-\alpha_{\max})^{K-1} = \frac{1}{2^{K-1}}$$

For the $n-1$ robots, we combine and obtain

$$\mathbb{P}[n{-}1 \text{ robots move once then stay}] \geq \frac{1}{(n^2 2^{K-1})^{n-1}}$$

Assuming again a worst case scheduling decision, the probability that the robot located at T takes no move in all K steps is

$$\mathbb{P}[T \text{ stays for } K \text{ steps}] \geq (1-\alpha_{\max})^{K} = \frac{1}{2^K}$$

and we combine all this to obtain the probability that one robot (at T) takes no move and that all other robots move exactly once with T as their target

$$\mathbb{P}[\cdots] \geq \frac{1}{2^{nK-n+1}n^{2(n-1)}}$$

That probability $\mathbb{P}[\cdots]$ is a strictly positive constant because K and n are both positive constants. To sum up, the probability that all robots are contained in a circle with diameter decreased by constant $\delta > 0$ in constant $K = nk$ steps is *at least* $\mathbb{P}[\cdots]$.

An upper bound on the expected number of steps need for all robots to be reachable is now easily obtained from a negative binomial distribution, following the same method as in Lemma 4. With D_0 being the diameter of *SEC* in the initial configuration, the number of successful progress necessary is $x = \lceil \frac{D_0}{\delta} \rceil - 1$, and we obtain, after much simplification,

$$\mathbb{E}[\text{steps to reachable}] \leq K\frac{x}{\mathbb{P}[\cdots]} = k\frac{\lceil \frac{D_0}{\delta} \rceil - 1}{2^{kn^2-n+1}n^{2n-3}}$$

which is constant since D_0, k, and n are all constant. This completes the proof of the first part.

Theorem 17; Part 2: From reachable to gathering. Starting from a configuration in which all robots are mutually reachable, we show that we reach gathering with high probability.

First, since the smallest enclosing circle is non increasing, once a reachable configuration has been achieved, all subsequent configurations are reachable.

Taking an execution fragment of nk steps starting in a reachable configuration, there is a positive probability that (1) every robot except one (say T') makes exactly one successful move toward T' at first trial and takes no further move in the $nk-1$ remaining steps, and (2) one robot (at T') makes no successful move in nk steps.

This is the same probability as one fragment of nk steps considered in the first part, so one additional successful fragment will reach gathering. This yields,

$$\mathbb{E}[\text{steps from arbitrary to gathering}] \leq k\frac{\lceil \frac{D_0}{\delta} \rceil}{2^{kn^2-n+1}n^{2n-3}}$$

Thus, gathering is achieved in constant expected steps. □

5 Crash-Tolerant Self-Stabilizing Gathering

We now extend the study on the feasibility of gathering to fault-prone environments. In this section, we consider the family of (n,f)-crash models, where n is the total number of robots and up to $f < n$ of them are faulty and may possibly crash.

Recall the two definitions for self-stabilizing gathering, namely, strong and weak. Let us first state a simple impossibility about strong gathering in systems with multiple faults.

Lemma 6 *No algorithm can possibly solve strong gathering in a crash-prone system with $f > 1$.*

PROOF: Consider any multivalent initial configuration ($f > 1$ thus $n > 1$). Take any two robots with distinct locations and let them crash before they move. They cannot move, so they will never share the same location, and hence strong gathering is never achieved. □ Notice that the above lemma holds regardless of the scheduler or additional assumptions of any kind, and obviously applies to both deterministic and probabilistic algorithms.

This leaves for study the case of strong gathering in the face of a single faulty robot (in Sect. 5.1), and the case of weak gathering with multiple faulty robots (in Sect. 5.2).

5.1 Single Crash ($f = 1$); Strong Gathering

We investigate the feasibility of strong gathering in the presence of a single faulty robot. We express the impossibility lemmas to cover one or several robots and simply refer to Lemma 6 for multiple robots so that the proof can focus on the case of a single crash.

Lemma 7 *In an (n,f)-crash system with $n \geq 3$ and $f \geq 1$, strong gathering is deterministically impossible under a round-robin scheduler, even with multiplicity knowledge.*

PROOF: The case when $f > 1$ is covered by Lemma 6, which leaves the case when $f = 1$.

By contradiction, assume that an algorithm $\mathscr{A}$ solves gathering deterministically in an $(n,1)$-crash system under a round-robin scheduler.

From Theorem 14, $\mathscr{A}$ must rely on multiplicity knowledge. The impossibility of Theorem 14 indeed applies here because a crash-free execution is valid in an $(n,1)$-crash system. Assuming that $\mathscr{A}$ could solve gathering without multiplicity knowledge would imply that it can solve it in a fault-free execution; a contradiction.

A round-robin scheduler is centralized by definition, so Lemma 2 applies and the system must necessarily reach a 1-bivalent configuration before it achieves gathering. Let γ_r be such a configuration and let r be the distinct robot. Consider also an arbitrary robot in the other location and call it r'.

It is easy to see that r' cannot chose to move to r in a 1-bivalent configuration, or else, an adversary can lead a fault-free execution to the same cyclic execution described in the proof of Theorem 14 (Fig. 2).

Now, consider the case when r crashes in γ_r. The algorithm is deterministic and robots are oblivious, so r' cannot distinguish that configuration from the fault-free one. The same decision must hence be applied and thus r' and r can never share the same location; a contradiction. □

We now prove a similar impossibility for probabilistic algorithms, but this time, under a fair centralized scheduler.

Lemma 8 *In an (n,f)-crash system with $n \geq 3$ and $f \geq 1$, there is no probabilistic algorithm that solves strong gathering under a fair centralized scheduler, even with multiplicity knowledge.*

PROOF: The case when $f > 1$ is covered by Lemma 6, which leaves the case when $f = 1$.

By contradiction, assume that an algorithm $\mathscr{A}$ solves gathering probabilistically in an $(n,1)$-crash system under a fair centralized scheduler.

Lemma 2 applies since the scheduler is centralized, and any gathering execution must reach a 1-bivalent configuration γ_r. Let r be the distinct robot in γ_r and G the location of the other robots.

We can now construct an adversary that prevents gathering. First, let r be the faulty robot and let it crash in γ_r before it moves. Second, let the adversary activate the correct robots in turn. Each time a robot r' moves to r, it is activated repeatedly until it moves back to G. The move to G must be possible or else G could not form in the first place (recall that robots are oblivious, anonymous, and disoriented). This activation is compatible with the fair centralized scheduler because every correct robot is activated infinitely often (fair) and in mutual exclusion (centralized). This leads to an infinite execution that holds no univalent configuration. Thus, $\mathscr{A}$ violates the Convergence property of gathering; a contradiction. □

We present now a simple lemma for a probabilistic algorithm in a system with two robots.

Lemma 9 *In a $(2,1)$-crash system, Algorithm 4.1 solves the strong gathering problem probabilistically under an unfair scheduler.*

PROOF: In a fault-free execution, the proof of Lemma 3 applies as it stands. In an execution with one crash, gathering is achieved through repeated activations of the correct robot. By the non-triviality condition of the unfair scheduler, it must activate the correct robot infinitely often. □

Based on the same proof, we obtain a lemma for the deterministic gathering of two robots under a centralized scheduler.

Lemma 10 *In a $(2,1)$-crash system, Algorithm 4.1 solves strong gathering probabilistically under an unfair centralized scheduler.*

In a system with more than two robots, strong gathering can still be achieved with a probabilistic algorithm, but requires the scheduler to be fair bounded. The algorithm does not need to rely on multiplicity knowledge.

Lemma 11 *In an $(n,1)$-crash system with $n \geq 3$ and under a fair bounded scheduler, Algorithm 4.1 solves strong gathering probabilistically.*

PROOF: The proof is identical to that of Theorem 17, where the target location T of the first part is chosen as the location of the faulty robot (crashed or not), and the second part requires no adaptation. □

5.2 Multiple Crashes ($f \geq 2$); Weak Gathering

We now extend the study to the case of weak gathering in the presence of multiple faulty robots.

We begin by proving the impossibility of deterministic and probabilistic weak gathering under a round-robin scheduler without additional assumptions.

Lemma 12 *In a (n,f)-crash system with $n \geq 3$ and $f \geq 2$, there is neither a probabilistic nor a deterministic algorithm that solves weak gathering under a round-robin scheduler, without additional assumptions.*

PROOF: By contradiction, assume that such an algorithm exists and call it $\mathscr{A}$.

Consider a fault-free execution e. The scheduler being centralized (implied by round-robin) Lemma 2 holds and every execution under algorithm $\mathscr{A}$ reaches a 1-bivalent configuration γ_b. Let r_1 be the distinct robot, and $\{r_2, \cdots, r_n\}$ the robots in the other location.

Consider an execution e' which differs from e in that r_1 and r_2 both crash in configuration γ_b, leading to the 1-bivalent configuration γ'_b. In the absence of multiplicity, bivalent configurations are undistinguishable for the robots.

So, some robot r_x in $\{r_3, \cdots, r_n\}$ is correct and has a positive probability of moving to the other location. To see this, consider that, otherwise, an adversary can generate an execution that is unable to transit from a 1-bivalent configuration to a univalent configuration in the fault-free case; a contradiction.

Take the more general case and assume that $\mathscr{A}$ is probabilistic. Assuming γ'_b does not change, the scheduler ensures that r_x is activated infinitely often. It follows that, with high probability, r_x moves, violating the Closure property; a contradiction. □

An immediate consequence of the previous lemma is the necessity of an additional assumption, such as multiplicity knowledge, even for probabilistic solutions and even under round-robin or bounded schedulers. Accordingly, we now consider systems in which robots are aware of multiplicity.

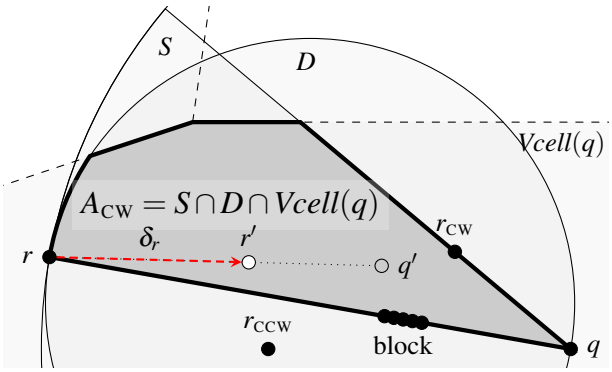


Figure 4: *Side Move:* Robot r selects maximal multiplicity point q as its target. Some robots block the way between r and q. Robot r_{CW} is the next robot clockwise (arbitrary choice) with respect to q. Robot r selects a point inside area A_{CW}, called q', and moves toward it.

5.2.1 Deterministic weak gathering with multiple crashes

Algorithm 5.1 is a deterministic algorithm that relies on multiplicity detection. Roughly, when a robot r becomes active, it considers the castles in the current configuration. If there are castles to which r does not belong, then it moves to the nearest one, say q, with ties broken arbitrarily.

Algorithm 5.1 Deterministic fault-tolerant weak gathering for robot r at location p

Functions:
$\mu(\Pi)$ :: the maximal multiplicity.
MaxMult(Π) :: the set of elements with multiplicity $\mu(\Pi)$.
#*onSegment*(p,q) :: the number of robots on segment $\overline{pq}$.

Actions:
Observe(Π) :: *true* $\longrightarrow$
if $\exists q \neq p : q \in MaxMult(\Pi)$ **then**
select nearest $q \in MaxMult(\Pi) \setminus \{p\}$;
if #*onSegment*$(p,q) < 2\mu(\Pi)$ **then**
// *Straight Move*
move toward q;
else
// *Side Move (see Fig. 4)*
let C := circle with center q and radius pq;
let S := largest sector in C from p with no robot;
let D := disc with diameter pq;
select some q' inside $A = S \cap D \cap Vcell(q)$;
move toward q';
else
stay;

When several robots occupy the space between r and q, moving over these robots brings the risk of a cyclic behavior that can never converge (see details in the appendix). To prevent this, r is required to perform a *side move* (see Fig. 4), in which r selects a destination within a zone A_{CW} (resp. A_{CCW}) constructed as the intersection of three areas (boundaries excluded).

- *Vcell*(q): cell of castle q in the Voronoi diagram built from the set of castles.
- D: the disc with segment $\overline{rq}$ as diameter.
- S: the largest sector clockwise (resp. counter-clockwise) centered at q and starting from r that contains no robot.

By moving within the zone, this ensures that (1) the distance between r and q decreases (r moves within D), (2) q remains the nearest castle from r (r moves within *Vcell*(q)), and (3) there are no robots between r and q (r moves within S).

Before proving the correctness of Algorithm 5.1, we establish two of its important properties.

The first property is an observation that the maximal multiplicity of configurations throughout executions of Algorithm 5.1 is non-decreasing. It holds for any centralized scheduler.

Proposition 1 *The maximal multiplicity of configurations is non-decreasing over any execution of Algorithm 5.1, under an unfair centralized scheduler.*

PROOF: In configurations with a single point of maximal multiplicity, the condition of the test in Algorithm 5.1 evaluates to false for any robot r that is on the point of maximal multiplicity, and thus, r does not move and the multiplicity does not change.

When there are several points of maximal multiplicity, they can be destroyed (one of its robot leaving the location, its multiplicity decreases) only one at a time because the scheduler is centralized. □

Another important property, which holds for any fair scheduler (i.e., not necessarily a centralized one), states that, in distinct configurations, the minimum distance between two robots is non-increasing.

Proposition 2 *Consider an execution of Algorithm 5.1 under a fair scheduler. Let $D(\gamma)$ be a function defined as the shortest distance between a robot and its nearest neighbor in configuration γ. Then, $D(\gamma)$ is non-increasing.*

PROOF: Assume by contradiction that there is a configuration γ_t such that $D(\gamma_t) < D(\gamma_{t+1})$. Let r and r' be two robots with distance $D(\gamma_t)$ from each other in γ_t. If neither r nor r' move at time t, then $D(\gamma_t) = D(\gamma_{t+1})$. So assume that at least one of them moves at time t. In γ_{t+1}, the distance from r to r' must have increased to be at least $D(\gamma_{t+1})$ so, one of the robots must have moved away from the other, say r' moved away from r. This means that r' had a neighbor $r'' \neq r$ such that the distance from r' to r'' was: (1) at most $D(\gamma_t)$, or else r (not r'') would be the nearest neighbor of r' and r' must have moved toward r, and (2) at least $D(\gamma_{t+1}) + \delta$ since, after moving, the nearest distance from a correct robot to its nearest neighbor is at least $D(\gamma_{t+1})$. It follows that $D(\gamma_t) \geq D(\gamma_{t+1}) + \delta$. A contradiction with $D(\gamma_t) < D(\gamma_{t+1})$.

Hence, $D(\gamma)$ is non-increasing. □

We can now show that a distinct configuration eventually leads to a configuration that contains a castle.

Proposition 3 *In an (n,f)-crash system, where $n > f$, a fair centralized scheduler and multiplicity detection, let e be an execution (or execution suffix) when robots move to the nearest robot. Starting from any distinct configuration, then e contains a configuration with maximal multiplicity larger than one.*

PROOF: We show that, starting from any distinct configuration, a location with multiplicity 2 is eventually formed.

Consider again the function $D(\gamma)$ defined as the shortest distance from a robot to its nearest neighbor. We know already from Proposition 2 that $D(\gamma)$ is non-increasing.

We now show that there is a configuration such that $D(\gamma)$ decreases strictly. Consider some distinct configuration γ_t, and let r be a correct robot with distance $D = D(\gamma_t)$ to its nearest neighbor r' in γ_t. Then, there must be a configuration $\gamma_{t'}$ ($t' > t$) during which one of the following situation occurs:

1. r' moves away from r. This means that there is a robot r'', originally at distance D or less from r', toward which r' moves. The distance from r' to r'' is at most $D(\gamma_t) - \delta$, so $D(\gamma_{t'}) \leq D - \delta$.

2. r moves and r' is still its nearest neighbor. r moves toward r' by distance δ, so $D(\gamma_{t'}) \leq D - \delta$.

3. r has a robot r'' as nearest neighbor. There are three cases:

   (a) either r or r' have moved, then we have already encountered one of the two previous cases, or

   (b) r'' has moved near r, then $\text{dist}(r, r'') \leq D$, or

   (c) the criteria used by r to break up ties among several of its nearest neighbors makes it select r'' instead of r', then $\text{dist}(r, r'') = D$.

   So, $D(\gamma_{t'}) \leq D$ and we can rename r' to r'' when iterating over the argument.

Since the scheduler is fair, there is a time after which r moves and $D(\gamma)$ decreases by at least δ. The rest follows. □

Theorem 18 *In an (n, f)-crash system, where $n > f$, Algorithm 5.1 deterministically solves weak gathering under a fair centralized scheduler if robots are aware of multiplicity.*

PROOF: Let us first prove that the Algorithm 5.1 satisfies the closure property of weak gathering, i.e., to a gathered configuration follow only gathered configurations. Let g_i be a gathered configuration. By definition of a gathered configuration, there is a unique point of maximal multiplicity that all correct robots occupy. Since, by construction of Algorithm 5.1, the correct robots do not move, any subsequent configuration is gathered. This proves closure.

By Proposition 1, the maximal multiplicity is nondecreasing. Let us now prove convergence by induction on the maximal multiplicity of configurations.

Proposition 3 forms the basis of the induction by showing that a distinct configuration leads to a configuration with maximal multiplicity larger than one.

Theorem 18; Induction step. We show that, starting from a non-gathered configuration γ_x with maximal multiplicity $M = \mu(\gamma_x) > 1$, a configuration γ_y with maximal multiplicity $M+1$ is eventually reached.

We say that a robot p_b is *blocked* if there are at least $\mu(c)-1$ robots on the segment between p_b and q_b, where q_b is the castle[13] that p_b selects if it is active in configuration γ. Let $\#castle(\gamma)$, $\#blocked(\gamma)$, and $\Sigma(\gamma)$ respectively denote the number of castles, the number of blocked robots, and the sum of distances from each robot to its nearest castle in configuration γ.

We can characterize a configuration γ by the quantities $\mu(\gamma)$, $\#castle(\gamma)$, $\#blocked(\gamma)$, and $\Sigma(\gamma)$. Consider a configuration γ_t with $x \leq t < y$, characterized by $\mu(\gamma_t) = M$, $\#castle(\gamma_t)$, $\#blocked(\gamma_t)$, and $\Sigma(\gamma_t)$, and consider all possible transitions from γ_t to the next configuration γ_{t+1}. Since the scheduler is centralized, one correct robot, say r, is active in configuration γ_t. We can summarize the transitions as follows:

1. r is in a castle.

   (a) $\#castle(\gamma_t) = 1$.
   r does not move. No change.

   (b) $\#castle(\gamma_t) > 1$.
   r aims for castle q.

      i. r is blocked. r takes a side move.
      $\#castle(\gamma_{t+1}) = \#castle(\gamma_t) - 1$.

      ii. r does not reach q.
      $\#castle(\gamma_{t+1}) = \#castle(\gamma_t) - 1$.

      iii. r reaches q.
      GOAL: $\mu(\gamma_{t+1}) = M+1$ and $\#castle(\gamma_{t+1}) = 1$.

2. r is not in a castle (aims for castle q).

   (a) r is blocked. r takes a side move.
   $\#blocked(\gamma_{t+1}) = \#blocked(\gamma_t) - 1$
   and $\Sigma(\gamma_{t+1}) < \Sigma(\gamma_t)$.

   (b) r does not reach q.
   $\Sigma(\gamma_{t+1}) = \Sigma(\gamma_t) - \delta$.

   (c) r reaches q.
   GOAL: $\mu(\gamma_{t+1}) = M+1$ and $\#castle(\gamma_{t+1}) = 1$.

A first observation is that no new castle is created, in other words, $\#castle(\gamma)$ never increases.

As long as there is a castle with at least one correct robot, that robot is eventually active since the scheduler is fair. Provided that there are at least two castles, the number of castles decreases. This happens until the system reaches a configuration $\gamma_{t'}$ ($x \leq t' < y$) in which any one of the following two conditions hold:

- There is one single castle.
- There are several castles, all of which consist only of crashed robots.

In either case, no robot already located in a castle moves.

If the configuration is already gathered, convergence is proved, so assume it is not. Then, there must be some correct robot located outside of a castle, and that robot must eventually become active as the scheduler is fair.

Let r' be a correct robot located outside of a castle. Each time r' is active, it selects one of the castles q' as its destination. In case another robot reaches a castle, the induction step is proved. So again assume that this is not the case. It follow that the number of castles and their locations do not change, thus q' is the same castle across activations of r'.

Let Δ' be the distance between r' in configuration $\gamma_{t'}$. If r' is initially blocked, then it performs a side move and r' is no

[13] C.f., definition of *castle* and *tower* in Section 2.7.

longer blocked (#*blocked*(γ) decreases). Recall that, by construction, performing a side move does not increase the distance between the robot and its destination.

After $\left\lceil \frac{\Delta'}{\delta} \right\rceil$ activations of r', it reaches castle q' thus increasing its multiplicity and proving the step.

The induction ensures that, as long as a gathered configuration is not reached, the maximal multiplicity increases. The multiplicity cannot possibly be larger than the number of robots, so it follows that a gathered configuration is eventually attained.

This proves convergence and, since we have proved closure before, the fact that Algorithm 5.1 solves weak gathering. □

5.2.2 Probabilistic weak gathering with multiple crashes

In the remainder of this section, we show that weak gathering can be solved probabilistically in an (n, f)-crash system (with $f < n$) under a fair scheduler.

Algorithm 5.2 is a probabilistic algorithm constructed on the deterministic Algorithm 5.1. While the latter is for a centralized scheduler, the former is for a fair scheduler, which allows robots to be active simultaneously.

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

Functions:
$\mu(\Pi)$ :: the maximal multiplicity in Π.
MaxMult(Π) :: the set of points with multiplicity $\mu(\Pi)$.

Actions:
Observe(Π) :: *true* $\longrightarrow$
if $p \notin$ *MaxMult*(Π) **then** /* *p not in a castle* */
execute extended Algorithm 5.1;
else if $|$*MaxMult*(Π)$| = 1$ **then** /* *unique castle* */
stay;
else /* *several castles* */
with probability $\alpha = \min\left(\frac{1}{\mu(\Pi)}, \frac{1}{2}\right)$ **do**
execute extended Algorithm 5.1;
otherwise
stay;

The idea of the algorithm is that, in some situations (several castles or distinct configurations), the simultaneous activation of several robots could lead to endless oscillations of the system. For instance, given two robots which are reachable and nearest from each other, activating them together would lead to them swapping their positions. To prevent this situation from occurring endlessly, the robots are required to first toss a coin and actually move only upon success.

In addition, the side move performed in Algorithm 5.1 defines a region from which a target point is selected arbitrarily. Due to concurrent moves under Algorithm 5.2, an arbitrary choice is no longer adequate. Therefore, it becomes necessary to extend Algorithm 5.1 such that the side move prevents two simultaneously moving robots from reaching the same location. The choice of an appropriate target is guided by the following requirements:

- Let two robots r and r', initially collinear with castle q, select target points T and T'. Then, segments $\overline{rT}$ and $\overline{r'T'}$ intersect if and only if r and r' are collocated.

A construction that satisfies this requirement is presented in the appendix (Sect. A.2). The probabilistic algorithm relies on Algorithm 5.1 extended with a side move meeting those requirements.

We first show that the convex hull of positions is non-increasing. This simple result is important as one factor to ensure that the system does not oscillate.

Proposition 4 *In an (n, f)-crash system, where $n > f$, with any scheduler and multiplicity detection, let e be an execution under Algorithm 5.2.*

Let γ_t and $\gamma_{t'}$ be two configurations of e, and Conv(γ_t) (respectively Conv($\gamma_{t'}$)) the convex hull of robot locations in γ_t (resp. $\gamma_{t'}$). Then, $t' > t \implies$ Conv($\gamma_{t'}$) $\subseteq$ Conv(γ_t).

PROOF: Let r be an arbitrary robot that moves through Algorithm 5.2: it can stay, move toward another robot, or perform a side move.

In all three cases, the entire segment between r's location and its target destination must be contained within the convex hull. When r stays, this holds trivially. When r moves toward another robot, this holds because of the convexity of the convex hull. When r performs a side move, this holds from the definition of the side move.

As a result, no move can possibly bring a robot outside of the convex hull, which is thus non-increasing. □ Notice however that the convex hull is not necessarily decreasing since the robots located at the vertices of the convex hull could be crashed robots.

We continue by proving important properties of executions under Algorithm 5.2. The first proposition shows that, if the number of castles can increase from one configuration to the next, then the maximal multiplicity must necessarily have decreased.

Proposition 5 *In an (n, f)-crash system, where $n > f$, a fair scheduler and multiplicity detection, let e be an execution under Algorithm 5.2. Let γ_t and γ_{t+1} be any two consecutive configurations in e. The number of castles increases in γ_{t+1} only if the maximal multiplicity decreases in γ_{t+1}.*

PROOF: Let K denote the number of castles in γ_t, and M the maximal multiplicity in γ_t. Suppose that there are $K+1$ castles in γ_{t+1}. Now, assume by contradiction that the maximal multiplicity in γ_{t+1} is M or more. Since there are $K+1$ castles in γ_{t+1}, at least one castle of multiplicity M or more must have been created from M independent robots (i.e., robots that did not belong to a castle in γ_t).

Consider one of the robots, call it r, independent in γ_t and forming the new castle in γ_{t+1}. There are three possible cases.

- If r did not move, then there must be another robot r' that has moved, or else r would not have been independent in γ_t. Then, consider the case of r' instead.

- r performed a straight move. By construction of the algorithm, there are less than M independent robots on the segment between r and its nearest castle, r included. But, by construction, no robot performing a side move can reach a ray containing robots performing straight moves.

- r performed a side move. The area targeted by the side move is convex, does not contain any robot, and does not contain any point reachable with a straight move.

  Hence, all robots collocated with r in γ_{t+1} must have performed a side move.

  With the extended construction of the side move, every robot r' collocated with r in γ_{t+1} must have been collocated with r in γ_t. Thus, r was forming a castle in γ_t. A contradiction.

The maximal multiplicity in γ_{t+1} is not M or more. The number of castles increases in γ_{t+1} only if the maximal multiplicity decreases in γ_{t+1}. □

Proposition 6 *In an (n,f)-crash system, where $n > f$, a fair scheduler and multiplicity detection, let e be an execution under Algorithm 5.2. If a configuration has a unique castle, then all configurations after that have only one castle and the maximal multiplicity is nondecreasing.*

PROOF: Let γ_t be a configuration in e with a unique castle.

By construction of Algorithm 5.2, no robots in the castle move when activated. Thus, the maximal multiplicity does not decrease in configuration γ_{t+1}. From Proposition 5, the number of castles increases in γ_{t+1} only if the maximal multiplicity decreases in γ_{t+1}. Therefore, the number of castles does not increase in γ_{t+1}.

The rest follows by induction on configurations. □

An important consequence of these two propositions is that, when a configuration with a unique castle is reached, then only configurations with a unique castle can follow. In other words, distinct configurations or configurations with several castles can no longer occur. We now additionally show that the system progresses deterministically to a gathered configuration. As a result, we can later consider the formation of a unique castle to be final, as it deterministically leads to gathering in finite steps.

Proposition 7 *In an (n,f)-crash system, where $n > f$, a fair scheduler and multiplicity detection, any execution e (or execution suffix) under Algorithm 5.2 that starts in a configuration with a unique castle leads to a gathered configuration in finite steps.*

PROOF: Let γ_t be a configuration with a single castle Q_t and maximal multiplicity $M = \mu(\gamma_t) \geq 2$. We prove that, either γ_t is gathered or there exists a configuration $\gamma_{t'}$ in e with $t < t'$ such that $\mu(\gamma_{t'}) > M$.

Since γ_t has a single castle, a robot located in the castle does not move when activated. Thus, only independent robots can move when active. In Algorithm 5.2, independent robots execute the first clause of the test, so the execution is deterministic and depends only on the activations of the scheduler.

If there are no independent correct robots, then the configuration is already gathered. Let us now consider the case when some correct robot is independent. Let r be one such robot and let D_t be the distance from r to Q_t in configuration γ_t.

The scheduler being fair, it must activate r eventually. We consider two cases, depending whether r is blocked or not in configuration γ_t.

- If r is not blocked, then it takes $\lceil D_t/\delta_r \rceil$ activations of r to reach Q_t, thus increasing the multiplicity of Q_t, and hence maximal multiplicity, by at least one.
- If r is blocked in γ_t, then it performs a side move when it is activated. It is possible that all other robots blocked on the same ray as r are activated at the same time, performing a side move in the same direction. Let B_t be the number of robots blocked on the same ray as r. B_t is at most $n-2M$ because M robots form castle Q_t and M robots block the others on the ray (those may have crashed, so they will not necessarily move).

  Let all blocked robots move together with r, with r being the farthest robot on the ray. After completing a side move, the number of blocked robots decreases by M. So, after at most $\frac{n}{M} - 2$ side moves, r is no longer blocked, and the rest follows from the first case.

This proves the claim and the remainder of the proof follows by induction on the maximal multiplicity. □

We now show that the shortest distance between a robot and its nearest neighbor is also non-increasing for Algorithm 5.2.

Proposition 8 *Consider an execution of Algorithm 5.2 under a fair scheduler. Let $D(\gamma)$ be a function defined as the shortest distance between a robot and its nearest neighbors in configuration γ. Then, $D(\gamma)$ is non-increasing.*

PROOF: In distinct configurations, all robots execute only the third clause of the test of Algorithm 5.2. So, a robot either (1) executes Algorithm 5.1 and $D(\gamma)$ is non-increasing by Proposition 2, or (2) stays and $D(\gamma)$ is non-increasing trivially. □

Proposition 9 *In an (n,f)-crash system, where $n > f$, with a fair scheduler and multiplicity detection, let e be an execution under Algorithm 5.2, and let $e[t:]$ be any execution suffix of e starting in a distinct configuration γ_t.*

Then, with high probability, $e[t:]$ contains a configuration with maximal multiplicity larger than one.

PROOF: Given a distinct configuration γ, let us first define its attractor graph $AG(\gamma)$ to be a weighted directed graph in which each robot is a vertex, and such that, there is an arc from robot r to robot r' if and only if r is not crashed in γ and, upon activation in γ, r will select r' as its target destination according to Algorithm 5.2 (and by extension Algorithm 5.1). The weight of an arc is given by the distance separating the two robots. We say that r' is the attractor of r in configuration γ.

Each path in $AG(\gamma)$ has non-increasing weights and ends either in a cycle of equal weights or with a crashed robot. Since $n > f$, there is at least one robot that never crashes, and hence at least one path exists in every configuration of e.

Consider the execution fragment $e[t:]$ starting in distinct configuration γ_t, and take the extremity of one path in $AG(\gamma_t)$ such that the weight of the last arc(s) is minimal. Let us denote this weight by $\Delta(\gamma_t)$, and consider independently the two possible situations regarding the extremity of the path.

1. The path ends with a crashed robot.
   Let r' be the crashed robot and r the last correct robot on the path, with r' as attractor. Then, $\Delta(\gamma_t)$ is the distance separating r and r' in γ_t.

   Over successive activations in fragment $e[t:]$, three situations may occur. When we say that a robot gets close to r, we mean that there is a third robot r'' such that $\text{dist}(r,r'') \leq \text{dist}(r,r')$ and hence r changes its attractor from r' to r''.

(a) Robot r does not crash and no other robot gets close to r.
After $\left\lceil \frac{\Delta(\gamma_t)}{\delta_r} \right\rceil$ successful moves of r, r reaches the location of r', resulting in a configuration with maximal multiplicity larger than one.
The number of activations follows a binomial distribution, and hence this occurs after constant expected number of activations of r. The scheduler being fair, $e[t:]$ contains a configuration with maximal multiplicity larger than one with high probability.

(b) Robot r crashes in configuration $\gamma_{t'}$ with $t' > t$.
We apply the same argument starting with configuration $\gamma_{t'}$ and other robots. This happens at most $f-1$ times.

(c) Robot r changes its attractor to another robot r'' in a configuration $\gamma_{t'}$.
In $\gamma_{t'}$, the distance $\text{dist}(r,r'') \leq \text{dist}(r,r')$. Take the new path in which r is now involved and continue applying the argument over its extremity with $\Delta(\gamma_{t'})$ such that:

$$\Delta(\gamma_{t'}) \leq \text{dist}\left(r,r''\right) \leq \text{dist}\left(r,r'\right) \leq \Delta(\gamma_t)$$

2. The path ends in a cycle.
The cycle involves q non-crashed robots ($2 \leq q \leq n$), all at distance $\Delta(\gamma_t)$ to their attractor.

Over successive activations in fragment $e[t:]$, there are several situations that may occur.

(a) No robots crash and no external robot gets close.
Each time some of the robots involved in the cycle are activated, the following situations may occur.

i. Some other robot in the cycle is not activated.
Let r be an activated robot with attractor r', such that r' is not activated.
With probability at least $\frac{1}{n}$ robot r moves (while r' does not), and the cycle is broken. We apply the argument again starting with the new configuration $\gamma_{t'}$, with $\Delta(\gamma_{t'})$ such that:

$$\Delta(\gamma_{t'}) \leq \text{dist}\left(r,r'\right) - \delta_r \leq \Delta(\gamma_t) - \delta_r$$

ii. All robots in the cycle are activated.
There are three sub-cases:
- With probability $\left(1-\frac{1}{n}\right)^q$ no robots move. The situation does not change.
- With probability $\left(\frac{1}{n}\right)^q$ all robots move. The situation remains if and only if (1) all robots r_i involved in the cycle have the same reachable distance δ_{r_i}, and (2) $\Delta[\gamma_t] = \delta_{r_i}$.
In all other cases, $\Delta[\gamma_{t+1}] < \Delta[\gamma_t]$.
- With remaining probability, a strict subset of the robots move and the other don't. This is identical to the previous case, when some robot is not activated. This results in the cycle being broken, and we apply the argument again starting with the new configuration $\gamma_{t'}$, with $\Delta(\gamma_{t'})$ such that:

$$\Delta(\gamma_{t'}) \leq \text{dist}\left(r,r'\right) - \delta_r \leq \Delta(\gamma_t) - \delta_r$$

(b) Some robot r in the cycle crashes in configuration $\gamma_{t'}$ with $t' > t$.
The path no longer ends in a cycle, and we apply the argument starting in configuration $\gamma_{t'}$ and with $\Delta(\gamma_{t'})$ such that $\Delta(\gamma_{t'}) \leq \Delta(\gamma_t)$.

(c) Some robot r changes its attractor to another robot r'' in a configuration $\gamma_{t'}$.
This breaks the cycle and defines a new path involving r. We apply the argument over the extremity of this path, with $\Delta(\gamma_{t'})$ such that

$$\Delta(\gamma_{t'}) \leq \text{dist}\left(r,r''\right) \leq \Delta(\gamma_t)$$

Regardless of scheduler choices, the minimal distance from a non-crashed robot to its attractor eventually decreases and, with high probability, the system reaches a configuration with multiplicity larger than one. □

Lemma 13 *In an (n,f)-crash system, where $n > f$, with a fair scheduler and multiplicity detection, let e be an execution under Algorithm 5.2. Let e' be any execution suffix starting in a configuration with multiple castles. Then, with high probability, e' contains a configuration with a single castle.*

PROOF: Let γ be a configuration with $K > 1$ castles of multiplicity M in e', and let us calculate the probability to reach a configuration γ' with K' castles of multiplicity M' after the next activation.

Let $\mathbf{K}$ denote the set of castles in γ. For each castle $k \in \mathbf{K}$, let i_k denote the number of (incoming) robots that can enter castle k upon activation. To be counted, a robot must be correct, located outside castle k, activated by the scheduler in configuration γ, have castle k as its destination, and be able to reach k in one step. Similarly, let o_k denote the number of (outgoing) robots that can leave castle k upon activation. To be counted, a robot must be correct, located inside castle k, and activated by the scheduler. Note that, when castles are near, a single robot may be counted simultaneously as an outgoing robot of some castle and an incoming robot of another castle.

We now define a function $\text{BALANCE}(i,o)$ to calculate the probability that the movement of i incoming robots and o outgoing robots exactly compensate each other. This is given by the probability that the same number of incoming and outgoing robots move, so that every departure of an outgoing robot is compensated by the arrival of an incoming one.

$$\begin{aligned}
\text{BALANCE}(i,o) &= \mathbb{P}[\text{none move}] + \sum_{m=1}^{\min(i,o)} \mathbb{P}[m \text{ arrive/depart}] \\
&= (1-\frac{1}{M})^{i+o} + \sum_{m=1}^{\min(i,o)} \left[\begin{array}{l} \binom{i}{m}(\frac{1}{M})^m(1-\frac{1}{M})^{i-m} \\ \times \binom{o}{m}(\frac{1}{M})^m(1-\frac{1}{M})^{o-m} \end{array} \right] \\
&= (1-\frac{1}{M})^{i+o} + \sum_{m=1}^{\min(i,o)} \left[\binom{i}{m}\binom{o}{m}(\frac{1}{M})^{2m}(1-\frac{1}{M})^{(i+o-2m)} \right] \\
&= (1-\frac{1}{M})^{i+o} \left(1 + \sum_{m=1}^{\min(i,o)} \binom{i}{m}\binom{o}{m}(\frac{1}{M-1})^{2m} \right)
\end{aligned}$$

We define the function $\text{INCREASE}(i,o,x)$ to return the probability that the multiplicity of a castle increases by x in the

presence of i incoming robots and o outgoing robots. This is the probability that x incoming robots move with the remaining incoming and outgoing robots compensating each other's movements.

$$\textsc{Increase}(i,o,x) = \binom{i}{x}(\frac{1}{M})^x \cdot \textsc{Balance}(i-x,o)$$

Let $\mathbb{P}_{inc}(K',x)$ return the probability that configuration γ' has exactly K' castles of multiplicity $M' = M+x$. That probability can be expressed as the probability that any subset $\mathbf{K}'$ of K' castles increase their multiplicity by x and all remaining castles do not increase multiplicity to any value x' larger or equal to x. Let $\mathbf{P}_{=K'}(\mathbf{K})$ denote the set of subsets of $\mathbf{K}$ of cardinality K', and we can express $\mathbb{P}_{inc}(K',x)$ as follows.

$$\mathbb{P}_{inc}(K',x) = \left[\begin{array}{l} \displaystyle\prod_{\mathbf{K}' \in \mathbf{P}_{=K'}(\mathbf{K})} \prod_{k' \in \mathbf{K}'} \textsc{Increase}(i_{k'}, o_{k'}, x) \\ \times \displaystyle\prod_{k'' \in \mathbf{K} \setminus \mathbf{K}'} \prod_{x'=x}^{i_{k''}} \left(1 - \textsc{Increase}(i_{k''}, o_{k''}, x')\right) \end{array} \right]$$

The probability of having a single castle in the next configuration is obviously at least as high as having a single castle by increasing the multiplicity of one of them by one. So, we can state the following inequality

$$\mathbb{P}[\gamma' \text{ has one single castle}] \geq \mathbb{P}_{inc}(K'=1, x=1)$$

The exact probability must consider increases of the multiplicity by more than one, and the change in number of castles due to a decrease of the multiplicity. However, this is sufficient for the proof since, as we are not concerned here with measuring an actual convergence rate, the mere existence of a transition with positive probability is sufficient.

Let us consider the configurations for which $\mathbb{P}_{inc}(K'=1, x=1)$ is zero. From the formula obtained for $\mathbb{P}_{inc}$, we see that it is zero when, for all castle k, i_k is zero. This can occur in several situations.

- All robots have crashed.
  This contradicts the assumption that $f < n$ which implies that there is at least one correct robot (i.e., a robots that never fails).
- The "near" robots are never activated.
  This contradicts the assumption that the scheduler is fair. If some "near" and correct robots exist, they must be activated eventually.
- There are no "near" robots.
  When a "far" robot r is activated, its distance to the nearest castle decreases by δ_r (to simplify the discussion we omit the case of the side move). Thus, either the configurations of castle change or r becomes a "near" robot.

So, with high probability, e' contains a configuration with a single castle. □

Theorem 19 *In an (n,f)-crash system, where $n > f$, Algorithm 5.2 probabilistically solves weak gathering under a fair scheduler if robots are aware of multiplicity.*

PROOF: Closure is satisfied by Algorithm 5.2 because, in a gathered configuration, all correct robots are by definition located on a unique castle, and hence do not move when activated. Thus, a gathered configuration always follows after a gathered configuration and closure is satisfied.

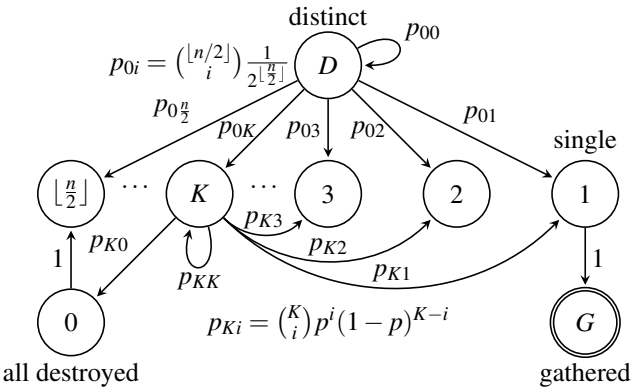


Figure 5: Markov chain representing the transitions of changes in the number of castles. A number represents the number of castles in the configurations. For every $K \in \{2, \ldots, \lfloor \frac{n}{2} \rfloor\}$, outgoing transitions follow a binomial distribution (only K depicted). Transitions from K to distinct are ignored because transitioning to state 0 instead favors an adversary. State 1 (single castle) leads to a gathered configuration, which is absorbing. When all castles are destroyed, a worst-case choice leads to $\lfloor \frac{n}{2} \rfloor$ castles (e.g., with lower multiplicity) in the next configuration.

To show convergence, let us consider an adversary *Adv*, as defined by (1) an initial configuration, (2) an activation strategy, and (3) control of robot crashes. However, *Adv* has no control on random choices made by robots, and no prior knowledge of their outcomes. The goal of *Adv* is then to construct an infinite execution $\bar{\varepsilon}$ that contains no gathered configurations, and such that $\bar{\varepsilon}$ occurs with non-zero probability.

From Proposition 7, the formation of a single castle leads to a gathered configuration. So, *Adv* must prevent the formation of a single castle. Let us now focus on the number of castles in each configuration and look at the transitions when this changes. Figure 5 depicts a Markov chain that represents the changes in the number of castles. The chain provides a conservative estimation by integrating simplifications that systematically favor *Adv*. Since we are not concerned here with measuring the actual convergence rate, the mere existence of transitions with a positive probability is sufficient. We now describe its construction.

Assume first that the system is in a distinct configuration. From Proposition 9, *Adv* cannot prevent the formation of castles. It can however control activations so that several castles are formed simultaneously. To maximize the chance of creating multiple castles, *Adv* can postpone the activations of every robot that can reach its nearest neighbor, until all robots form pairs[14] of mutually nearest neighbors. Then, all robots are activated and move with probability $\frac{1}{2}$, resulting in a number of castles that follows a binomial distribution $B(\lfloor \frac{n}{2} \rfloor, \frac{1}{2})$.

$$\mathbb{P}[x \text{ castles created}] = p_{0x} = \binom{\lfloor \frac{n}{2} \rfloor}{x} \frac{1}{2^{\lfloor \frac{n}{2} \rfloor}}$$

[14] Situations in which robots form a chain or a cycle result in the creation of fewer castles, which is less favorable to the adversary *Adv*.

When no castles are created, the resulting configuration is distinct and the process repeats itself.

Assume now that the system is in a configuration with $K > 1$ castles. The number of castles can change in two possible ways: (1) independent robots moving inside a castle, or (2) robots leaving a castle thus destroying it.

When independent robots move inside a castle, no additional castle can be created in the next configuration (from Proposition 5). Looking at the best case (for *Adv*) when one independent robot is ready to move inside every castle, we obtain that the probability of castle creation follows a binomial distribution $B(K, \frac{1}{M})$.

$$\mathbb{P}[x \text{ castles created}] = p_{Kx} = \binom{K}{x} (\frac{1}{M})^x (1 - \frac{1}{M})^{K-x}$$

When no castles are created, the resulting configuration is identical and the situation is repeated.

When robots leaving castles result in their destruction, there can be three possible outcomes in the configuration that follows:

- Several castles remain. There can be no more than K castles.
- A single castle remain. This is the situation that *Adv* must avoid.
- All castle destroyed. The next configuration has a lower maximal multiplicity, and can result in a larger number of castles of multiplicity lower than M.

The probability that a given castle is not destroyed by some robot moving outside has the following probability

$$\mathbb{P}[\text{castle not destroyed}] = \alpha^M = (\frac{1}{M})^M = p(M)$$

For a configuration to have multiple castles, M must necessarily be between 2 and $\lfloor \frac{n-1}{2} \rfloor$. For any of these values, both $p(M)$ and $1 - p(M)$ are strictly positive. To simplify the model (Fig. 5), we assume that it is a positive constant p with $0 < p < 1$, chosen to be the value that favors the adversary most, and that does not depend on multiplicity. Again, its exact value is secondary, as long as it is strictly positive for any value of K, M, and n finite.

The number of castles in the following configuration follows a binomial distribution $B(K, \alpha^M)$

$$\mathbb{P}[x \text{ castles size } M \text{ remain}] = p_{Kx} = \binom{K}{x} p^x (1-p)^{K-x}$$

When no castles remain, the maximal multiplicity decreases and a larger number of castles of lower multiplicity may be created. Unless the next configuration is distinct, there can be no more than $\lfloor \frac{n}{2} \rfloor$ castles in the resulting configuration. We observe that, from the viewpoint of the adversary *Adv*, the best case is when the maximal number of castles are formed. So, we assume that the destruction of all castles in a configuration always leads to a configuration with the maximal number of castles.

Putting this together gives us the Markov chain depicted in Fig. 5. For configurations with multiple castles, the figure shows only the transitions from state K. According to Proposition 7, configurations with a single castle lead to a gathered configuration with probability 1.

The resulting Markov chain contains a single absorbing state G. It is a well-known result that, in an absorbing Markov chain, the process will be absorbed with probability 1. Since the only absorbing state is G (gathered), convergence is satisfied with probability 1. □

6 Byzantine Tolerant and Self-stabilizing Gathering

We study now the feasibility of gathering in systems prone to Byzantine failures.

Agmon and Peleg [1] proved the impossibility of weak gathering in a $(3,1)$-Byzantine system under a fair scheduler (Theorem 5). The result applies to both SYm and CORDA models. The following lemma proves that the impossibility still holds under a round-robin scheduler, and even if the algorithm is probabilistic.

Lemma 14 *In a Byzantine-prone system, there is no deterministic or probabilistic algorithm that solves (n, f)-weak gathering, $f \geq 1$ and $n > f + 1$, under a round-robin scheduler without additional assumptions.*

PROOF: By contradiction, let $\mathscr{A}$ be an algorithm that solves gathering. Assume that a single robot r_B is Byzantine (or let the other Byzantine robots behave like correct ones). Let $\mathscr{A}$ execute normally until all robots share the same location P. When activated, let r_B move to a second location P' selected as follows:

- if $\mathscr{A}$ is *deterministic*, chose P' such that, applying the criteria used in $\mathscr{A}$ when selecting a target location, some correct robot r will move to P'.
- if $\mathscr{A}$ is *probabilistic*, chose any $P' \neq P$.

In either case, a correct robot r must move because, robots being oblivious, they have no way to know that gathering was already achieved. Furthermore, in the absence of multiplicity detection, there is no way to distinguish P and P' by their multiplicity. Since there are at least two correct robots ($n > f + 1$) and the scheduler is centralized, the move of r toward P' results in a non-gathered configuration.

The situation can be repeated each time the system is in a gathered configuration. This clearly violates the closure property of weak gathering, since closure requires that any execution suffix starting in a gathered configuration contains only gathered configurations. Thus, $\mathscr{A}$ does not solve weak gathering. □

6.1 Deterministic Byzantine Gathering

The following lemma shows that if the power of the scheduler is increased, weak gathering is impossible in a $(3,1)$-Byzantine system, even if robots are aware of the system multiplicity.

Lemma 15 *In a $(3,1)$-Byzantine system, there is no deterministic algorithm that solves weak gathering under a fair centralized k-bounded scheduler with $k \geq 2$, even if robots are aware of multiplicity.*

PROOF: Assume an arbitrary initial configuration, a configuration where robots occupy distinct positions. 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.

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

Lemma 16 *In an (n,f)-Byzantine system with n even and $f \geq 1$, there is no deterministic algorithm that solves weak gathering under a fair centralized k-bounded scheduler with $k \geq \left\lceil \frac{n-f}{f} \right\rceil$, even if robots are aware of multiplicity.*

PROOF: Consider an initial configuration such that the configuration is bivalent with two locations having equal multiplicity (n is even), and robots are reachable from each other. Assume that the Byzantine robots are spread evenly between the two locations. Let g_1 and g_2 be the two groups, such that, if f is odd, g_1 has one more Byzantine robot than g_2. Consider the following activation schedule.

- Activate a correct robot in g_2: it must necessarily move to g_1 or else no execution could possibly reach gathering.
- Activate a Byzantine robot in g_1, and let it move to g_2. The resulting configuration is symmetrical to the original one.

By repeating the same sequence, a Byzantine robot counterbalances every move of a correct robot, and the system is always in a bivalent configuration.

Since there is a total of $n-f$ correct robots and f Byzantine robots, the adversary can distribute the moves between the Byzantine robots. Thus, between each consecutive activation of a correct robot, the adversary must activate a Byzantine robot only $\left\lceil \frac{n-f}{f} \right\rceil$ times. □

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

Lemma 17 *In an (n,f)-Byzantine system with n odd and $f \geq 2$, there is no deterministic algorithm that solves weak gathering under a fair centralized k-bounded scheduler with $k \geq \left\lceil \frac{n-f}{f-1} \right\rceil$, even if robots are aware of multiplicity.*

PROOF: Let the initial configuration be a bivalent configuration such that robots are reachable from each other and the multiplicity of the two locations differ by one.

Let g_a be the small group and g_b the big one. Let all f Byzantine robots be in g_b. If there are more than half Byzantine robots, then simply let all robots in g_b be Byzantine ones.

Now, call one of the Byzantine robots in g_b the switch r_{sw}, and consider the following schedule:

1. Activate a correct robot in g_a, say r. It must be instructed to move to the other point of multiplicity, or else gathering would not possibly be achieved in a fault-free case.
2. Each time a correct robot moves to g_b, activate a Byzantine robot in g_b (except the switch r_{sw}), and let it move to g_a.
3. Repeat the procedure until one of the following condition holds: (1) all correct robots originally in g_a have moved, or (2) all Byzantine robots originally in g_b have moved, except r_{sw}.
4. Move r_{sw} to g_a, which becomes now the larger group.
5. Repeat the procedure with correct robots in g_b so that they move to g_a.

At each iteration of the procedure, $f-1$ correct robots move from one group to the other, while $f-1$ Byzantine robots negate their move.

Thus, a Byzantine robot needs to be activated at most $\left\lceil \frac{n-f}{f-1} \right\rceil$ times between two consecutive activations of a correct robot, . □

7 Summary

We have summarized most of the theorems, their relationships, and their scope into tables (Table 1 and 2). Results are grouped according to the problem (strong or weak gathering) and the fault models: strong gathering in fault-free (Table 1a) and single crash (Table 1b) environments; as well as weak gathering in multiple crashes (Table 2a) and single Byzantine environments (Table 2b).

All tables are designed to be read as follows: Each row represents a different scheduler, while columns distinguish other assumptions, such as multiplicity, conditions on the number of robots n, conditions on the maximum number of faulty robots f, or whether deterministic or probabilistic solutions are admissible.

Each cell answers whether the problem admits a solution under the corresponding set of assumptions. A positive result appears as "OK" followed by the number of the corresponding lemma or theorem in brackets. Conversely, a negative result (impossibility) is denoted by "NO" and a greyed background.

An "**OK**" or "**NO**" in bold means that the cell corresponds to the assumptions stated explicitly in the relevant theorem. When the text appears in normal face, the result comes instead as a consequence of the theorem and the relationship between assumptions. For instance, a positive result expressed and proved with an unfair centralized scheduler (e.g., Table 1a; L.3) necessarily applies to the more restrictive schedulers, such as the fair centralized or round-robin schedulers, even though this is implicit.

7.1 Strong self-stabilizing gathering

The results pertaining to the strong gathering in a fault-free model are summarized in Table 1a, while those related to the crash model with a single faulty robot are in Table 1b. The tables are divided vertically according to the availability of multiplicity detection, then whether gathering is deterministic or probabilistic, and finally to the number of robots n (i.e., $n = 2$ or $n > 2$). Note that, when $n = 2$, the detection of multiplicity is irrelevant, and thus the results are identical in both columns.

Table 1: Strong gathering problem

(a) 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(*Th.1*)		**OK(L.4)**		unfair	NO(*Th.1*)	NO(*Th.3*)	**OK(L.4)**	NO(*L.5*)
OK(L.3)		OK(*L.4*)		unfair centr.	**OK(L.3)**	NO(*Th.15*)	OK(*L.4*)	NO(*L.5*)
NO(Th.1)	**OK(Th.2)**	OK(*L.4*)	OK(*Th.2*)	fair	**NO(Th.1)**	**NO(Th.3)**	OK(*L.4*)	NO(*L.5*)
OK(*L.3*)	OK(*Th.2*)	OK(*L.4*)	OK(*Th.2*)	fair centr.	OK(*L.3*)	NO(*Th.15*)	OK(*L.4*)	**NO(L.5)**
NO(*Th.1*)	OK(*Th.2*)	OK(*L.4*)	OK(*Th.2*)	fair k-bounded	NO(*Th.1*)	NO(*Th.15*)	OK(*L.4*)	**OK(Th.17)**
OK(*L.3*)	OK(*Th.2*)	OK(*L.4*)	OK(*Th.2*)	fair 2-bounded centr.	OK(*L.3*)	**NO(Th.15)**	OK(*L.4*)	OK(*Th.17*)
NO(Th.1)	OK(*Th.2*)	OK(*L.4*)	OK(*Th.2*)	fair 1-bounded	**NO(Th.1)**	NO/?(*Th.14/16*)[a]	OK(*L.4*)	OK(*Th.17*)
OK(*L.3*)	OK(*Th.2*)	OK(*L.4*)	OK(*Th.2*)	round-robin	OK(*L.3*)	**NO/?(Th.14/16)**[a]	OK(*L.4*)	OK(*Th.17*)

[a] Special: Th. 14 proves the impossibility of self-stabilizing gathering. Th. 16 proves it for gathering *provided Conjecture 1 holds*.

(b) Crash model; $f=1$

multiplicity				**Scheduler**	without multiplicity ($f=1$)			
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(*Th.1*)	NO(*L.7*)	**OK(L.9)**	NO(*L.8*)	unfair	NO(*Th.1*)	NO(*Th.3*)	**OK(L.9)**	NO(*L.5*)
OK(L.10)	NO(*L.7*)	OK(*L.9*)	NO(*L.8*)	unfair centr.	**OK(L.10)**	NO(*L.7*)	OK(*L.9*)	NO(*L.5*)
NO(*Th.1*)	NO(*L.7*)	OK(*L.9*)	NO(*L.8*)	fair	NO(*Th.1*)	NO(*Th.3*)	OK(*L.9*)	NO(*L.5*)
OK(*L.10*)	NO(*L.7*)	OK(*L.9*)	**NO(L.8)**	fair centr.	OK(*L.10*)	NO(*L.7*)	OK(*L.9*)	NO(*L.5*)
NO(*Th.1*)	NO(*L.7*)	OK(*L.9*)	OK(*L.11*)	fair k-bounded	NO(*Th.1*)	NO(*L.7*)	OK(*L.9*)	**OK(L.11)**
NO(Th.1)	NO(*L.7*)	OK(*L.9*)	OK(*L.11*)	fair 1-bounded	**NO(Th.1)**	NO(*L.7*)	OK(*L.9*)	OK(*L.11*)
OK(*L.10*)	**NO(L.7)**	OK(*L.9*)	OK(*L.11*)	round-robin	OK(*L.10*)	**NO(L.7)**	OK(*L.9*)	OK(*L.11*)

7.1.1 Fault-free model

As shown on Table 1a, in the absence of multiplicity detection, a bounded scheduler is both necessary and sufficient for solving probabilistic gathering of more than two robots. There is however no deterministic solution, regardless of the scheduler (i.e., even if the scheduler is round-robin).

In the presence of multiplicity detection, gathering is known to be possible with a fair scheduler, as proved by Suzuki and Yamashita [19]. The question remains open in the case of unfair schedulers.

When there are only two robots, gathering is known to be more difficult than with three or more robots, since all configurations are symmetrical. Suzuki and Yamashita [19] have proved the impossibility under a fair scheduler, and their proof actually applies to more restrictive schedulers, such as the fair 1-bounded scheduler. Interestingly, the problem becomes solvable under all classes of *centralized* schedulers, even the unfair ones.

7.1.2 Crash model

Table 1b summarizes the results obtained for the strong gathering problem with at most one robot crash.

Interestingly, without multiplicity detection, the results obtained for the fault-free and the crash models are identical, although they are covered by different theorems. Unlike in the fault-free model, multiplicity detection does not seem to help solve gathering. Indeed, in the crash model, results are identical whether or not robots are able to detect multiplicity, whereas they differed widely in the fault-free case.

In other words, while the introduction of multiplicity detection is indeed determinant in the fault-free case, it has no effect on solvability when faced with a single crashed robot.

7.2 Weak self-stabilizing gathering

Table 2 summarizes the results for weak gathering. Let us first remind that, in the fault-free model, there is actually no difference between strong and weak gathering (since the only difference in definitions is about the requirements put on the faulty robots), and thus the results of Table 1a, although not repeated, are of course also relevant here.

Table 2a summarizes the results for weak gathering (i.e., only the correct robots are required to gather at the same location) and distinguishes between the case of a single crash and multiple crashes. One interesting observation is that, in the case of a single crash (left part of Table 2a), results of weak with respect to strong gathering differ only if robots are able to detect multiplicity. In particular, Theorems 18 and 19 show that weak gathering is possible with schedulers for which strong gathering is not. This is because a system may reach a stable configuration in which all robots except the faulty one share the same location. In such a configuration, weak gathering is achieved but strong gathering is not.

In the case of multiple crashes and without multiplicity detection, even probabilistic gathering is impossible under any of the schedulers considered. With multiplicity detection and fair schedulers, however, probabilistic gathering is possible under any fair scheduler while deterministic gathering is possible if and only if the scheduler is also centralized. The question re-

Table 2: Weak gathering problem

(a) Crash model

$f=1$				**Scheduler**	$2 \leq f < n$			
multiplicity		without multiplicity			multiplicity		without multiplicity	
determ.	proba.	determ.	proba.		determ.	proba.	determ.	proba.
		NO(*Th.3*)	NO(*L.5*)	unfair			NO(*L.12*)	NO(*L.12*)
		NO(*Th.15*)	NO(*L.5*)	unfair centr.			NO(*L.12*)	NO(*L.12*)
OK(Th.4)[b]	**OK(Th.19)**	NO(*Th.3*)	NO(*L.5*)	fair	**NO(N. 2)**	**OK(Th.19)**	NO(*L.12*)	NO(*L.12*)
OK(Th.18)	OK(*Th.18*)	NO(*Th.15*)	NO(*L.5*)	fair centr.	**OK(Th.18)**	OK(*Th.18*)	NO(*L.12*)	NO(*L.12*)
OK(*Th.4*)	OK(*Th.19*)	NO(*Th.15*)	OK(*L.11*)	fair k-bounded	**NO(N. 2)**	OK(*Th.19*)	NO(*L.12*)	NO(*L.12*)
OK(*Th.4*)	OK(*Th.19*)	NO/?(*Th.14/16*)[a]	OK(*L.11*)	fair 1-bounded	**NO(N. 2)**	OK(*Th.19*)	NO(*L.12*)	NO(*L.12*)
OK(*Th.18*)	OK(*Th.18*)	NO/?(*Th.14/16*)[a]	OK(*L.11*)	round-robin	OK(*Th.18*)	OK(*Th.18*)	**NO(L.12)**	**NO(L.12)**

[a] Special: Th. 14 proves the impossibility of self-stabilizing gathering. Th. 16 proves it for gathering *provided Conjecture 1 holds*.

[b] Note that the results derived from Theorem 4 hold for the case (3,1). According to Note 2, in the case of $(n,1)$-crash, 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.

(b) Byzantine model

	multiplicity; deterministic				
	$f=1$			$2 \leq f < n/2$	
Scheduler	$n=3$	$n \geq 4$ (even)	$n \geq 4$ (odd)	$n \geq 4$ (even)	$n \geq 4$ (odd)
unfair	NO(*Th.5*)	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
unfair centr.	NO(*L.15*)	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
fair	**NO(Th.5)**	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
fair centr.	NO(*L.15*)	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
fair k-bounded	NO(*L.15*)	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
$(k \geq n-1)$-bounded	NO(*L.15*)	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
$(\Gamma(n,f) \leq k \leq n-2)$-bounded	NO(*L.15*)	NO(*L.16*)		NO(*L.16*)	NO(*L.17*)
$(2 \leq k < \Gamma(n,f))$-bounded	NO(*L.15*)				
fair 1-bounded					
centr. $(k \geq n-1)$-bound.	NO(*L.15*)	**NO(L.16)**		**NO(L.16)**	**NO(L.17)**
centr. $(\Gamma(n,f) \leq k \leq n-2)$-bound.	NO(*L.15*)	**NO(L.16)**		**NO(L.16)**	**NO(L.17)**
centr. $(2 \leq k < \Gamma(n,f))$-bound.	**NO(L.15)**				
round-robin					
fully synchronized	**OK(Th.6)**	**OK(Th.7)**	**OK(Th.7)**	**OK(Th.7)** if $n \geq 3f+1$	

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

mains open for unfair schedulers, but we believe that the answer depends greatly on minute details in the definition of the unfair scheduler.

7.3 Byzantine model

While Byzantine gathering is possible in fully synchronous environments, other positive results remain quite elusive. We have been able to extend impossibility results, but unable to find additional solutions for other models.

Under very specific assumptions, Algorithm 5.1 is likely to solve Byzantine gathering for some values of f, n, and k. However, this requires very specific assumptions, among which the requirement that *Byzantine robots have a mobility range no larger than the correct ones*. We have found a counter-example where the algorithm fails without this assumption, and thus omitted entirely from the study, thus leaving the question open.

8 Conclusion

The results presented in this paper extend prior work on the self-stabilizing gathering problem in fault-free and fault-prone environments, by shading light on the subtil line between possibility and impossibility. Most notably, we identify the role that additional synchrony, embodied by schedulers, can play toward making the problem possible. So far, our work is the most extensive study on the combined roles that randomization, multiplicity, and schedulers (centralized and bounded) can play in allowing a solution to fault-free and fault-tolerant gathering.

In particular, we have strengthened several key impossibility results on gathering, including Prencipe's [17] impossibility of fault-free gathering in the absence of multiplicity strengthened to cover up to the round-robin or 2-bounded centralized schedulers (depending on the definition of the problem), and Agmon and Peleg's [1] impossibility of Byzantine gathering under a fair scheduler extended to cover bounded centralized schedulers.

The main results of the paper are summarized in Table 1a for fault-free systems; in Table 1b and Table 2a for strong, resp. weak, gathering in crash-prone systems; and in Table 2b for weak gathering problem in Byzantine-prone systems.

The main results of the paper are summed up in Table 1a for fault-free systems; in Table 1b and Table 2a for strong respectivelly weak gathering in crash-prone systems; and in Table 2b for the weak gathering problem in Byzantine-prone systems.

Acknowledgments

We are grateful to François Bonnet, the editor, and the reviewers for their insightful and valuable comments.

Research partly supported by JSPS KAKENHI Grants No. 23500060 and No. 26330020.

References

[1] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal of Computing*, 36(1):56–82, 2006.

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

[3] Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA*, pages 337–346, 2013.

[4] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Byzantine convergence in robot networks: The price of asynchrony. In *Principles of Distributed Systems, 13th International Conference, OPODIS 2009, Nîmes, France, December 15-18, 2009. Proceedings*, pages 54–70, 2009.

[5] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theor. Comput. Sci.*, 411(34-36):3154–3168, 2010.

[6] R. Cohen and D. Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. In B. Durand and W. Thomas, editors, *23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS'06)*, volume 3884 of *LNCS*, pages 549–560, Marseille, France, February 2006. Springer.

[7] X. Défago, M. Gradinariu Potop-Butucaru, S. Messika, and P. Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering: Feasibility study. *DISC'06*, pages 46–60, 2006.

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

[9] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[10] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.

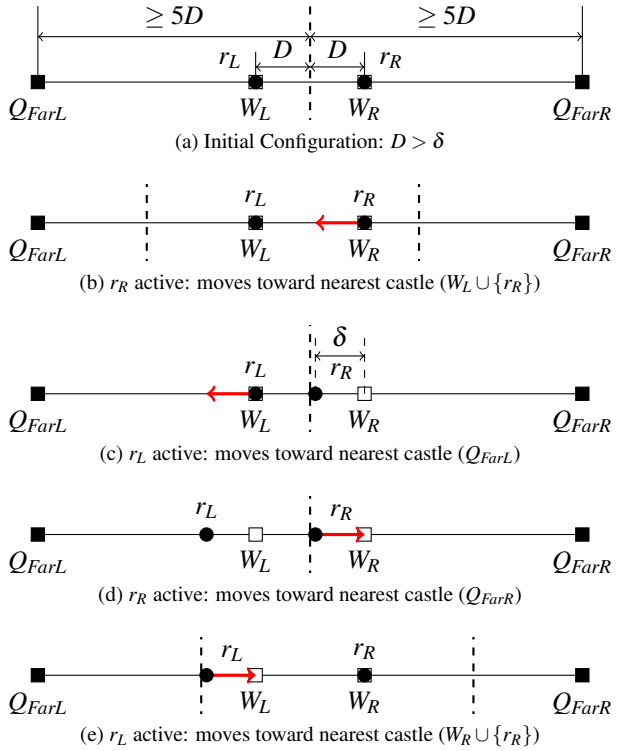
[11] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous mobile robots with limited visibility. *Theoretical Computer Science*, 337:147–168, 2005.

[12] Ta. Izumi, To. Izumi, S. Kamei, and F. Ooshita. Feasibility of polynomial-time randomized gathering for oblivious mobile robots. *IEEE Trans. Parallel Distrib. Syst.*, 24(4):716–723, 2013.

[13] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CA, USA, 1996.

[14] N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, August 2003.

[15] N. Megido. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal of Computing*, 12(4):759–776, 1983.



(a) Initial Configuration: $D > \delta$

(b) r_R active: moves toward nearest castle ($W_L \cup \{r_R\}$)

(c) r_L active: moves toward nearest castle (Q_{FarL})

(d) r_R active: moves toward nearest castle (Q_{FarR})

(e) r_L active: moves toward nearest castle ($W_R \cup \{r_R\}$)

Figure 6: Illustration of the necessity of introducing the side move. The naive algorithm (without side move) can result in and endless cycle.

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

[17] G. Prencipe. On the feasibility of gathering by autonomous mobile robots. In A. Pelc and M. Raynal, editors, *Proc. Structural Information and Communication Complexity, 12th Intl Coll., SIROCCO 2005*, volume 3499 of *LNCS*, pages 246–261, Mont Saint-Michel, France, May 2005. Springer.

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

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

A Appendix

A.1 Necessity of the Side Move for Algorithm 5.1

We must now show the necessity of introducing a side move in Algorithm 5.1. Assuming that robots execute the naive algorithm (Algorithm 5.1 without the clause executing the side move), we exhibit a situation in which the robots are unable to gather (depicted in Fig. 6):

Consider the initial configuration depicted in Figure 6a. Assume that the reachable distance of all robots is the same and call it δ. Let D be some arbitrary distance strictly larger than δ. The robots (or a subset thereof) are initially located such that they form four castles on a segment. Let Q_{FarL}, Q_{FarR} be the two castles at both ends of the segment and assume that they consist only of crashed robots. Let W_L, W_R be two towers such that they become a castle by adding one robot. For simplicity, assume again that they also consist only of crashed robots. Let r_L, r_R be two correct robots initially with W_L, W_R respectively. The location of the four castles is symmetric such that the midpoint between Q_{FarL} and Q_{FarR} is also the midpoint between W_L and W_R. The distance between W_L and W_R is $2D$ and the distance between Q_{FarL} and Q_{FarR} is at least $10D$.

Consider the scheduler as an adversary following a round-robin policy. First, r_R is active (Fig. 6b). According to the naive algorithm, r_R must move toward the nearest castle, which is the castle formed by W_L and r_L. The dashed lines on the figure represent the boundaries of the Voronoi cells of each of the three castles: $\{Q_{FarL}, Q_{FarR}, W_L \cup \{r_L\}\}$. Since r_R is located inside the Voronoi cell of castle $W_L \cup \{r_L\}$, it moves toward it.

Second, r_L is active (Fig. 6c). Since r_R has moved in the previous step, W_R is no longer the location of a castle. Now, r_L is located inside the Voronoi cell of Q_{FarL} and moves toward it.

Third, r_R is active again (Fig. 6d). There are only two castles left on the configuration, namely Q_{FarL} and Q_{FarR}. Since $D > \delta$, r_R is located to the right of the midpoint between W_L and W_R, which is also the midpoint between Q_{FarL} and Q_{FarR}. This means that r_R is in the Voronoi cell of Q_{FarR} and hence moves toward it. But, because r_R is at distance δ to W_R, it ends its movement exactly at W_R, thus forming a castle again.

Fourth, r_L is active and there are three castles (Fig. 6e). By construction, W_R is located at a distance at least $6D$ from Q_{FarL}, and hence r_L remains inside the Voronoi cell of the castle formed by W_R and r_R. r_L is also at a distance δ from W_L, and hence ends its movement exactly at W_L, forming a castle. This leads back to the initial configuration (Fig. 6a), and thus the cycle continues forever.

A.2 Disambiguation of Side Move for Algorithm 5.2

Algorithm A.1 Disambiguation of side move (robot at p).

Procedure:

SIDEMOVE$(p, q, \Pi) \longrightarrow$
- set origin at q
- let $\Pi_q \subset \Pi$ be all robots in $Vcell(q)$ or on its boundary.
- let r_{CW} be the first robot clockwise in Π_q starting from p.
- let θ_{CW} be angle $\angle pqr_{\text{CW}}$ or π, whichever is smaller.
- let θ^+ be one third of θ_{CW}.
- let v_p be the intersection of $\overline{qp}$ and the boundary of $Vcell(q)$.
- let *ray* be the ray from q with clockwise angle θ^+ from $\overline{qp}$.
- let V_a be the intersection of *ray* with $Vcell(q)$.
- let V_b be the intersection of *ray* with the circumference of D.
- let V' be V_a or V_b, whichever is nearest q.
- let vector $\vec{target} = \frac{\text{dist}(q,p)}{\text{dist}(q,v_p)} \vec{qv'}$.
- move toward point $\vec{target}$.

Algorithm A.1 describes one method to disambiguate the side move, and is illustrated in Figure 7. The lengths a and b depend on the position of robot P on the ray from castle Q relative to the boundary of the Voronoi cell of Q.

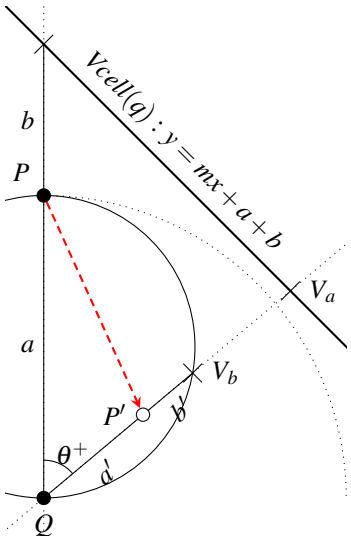


Figure 7: Disambiguated side move. $\frac{a}{a+b} = \frac{a'}{a'+b'}$.

The construction uses a trisection of the sector S calculated in the original side move. This ensures that a robot moving from a different ray does not end up at the same location.

In addition, taking the minimum between points V_a and V_b ensures that the segment $\overline{QV'}$ lies entirely within the zone desired for a side move. Since the zone is convex (intersection of three convex areas), segment PV' lies entirely inside the zone.

$$\alpha = \frac{a}{a+b} \in (0;1] \quad (1)$$

$$a'(\alpha) = \min\left(\alpha^2 \cos\theta^+, \frac{\alpha}{\cos\theta^+ - \frac{1}{m}\sin\theta^+}\right) \quad (2)$$

Since $a'(\alpha)$ is taken as the minimum of two functions that are both monotonic increasing in α over the range considered, $a'(\alpha)$ is itself monotonic increasing. It follows that, for two values α_1 and α_2 with $\alpha_1 \neq \alpha_2$, the segments from $P(\alpha)$ to $P'(\alpha)$ do not cross.