Gathering of Mobile Robots Tolerating Multiple Crash Faults

Zohir Bouzid*, Shantanu Das†, and Sébastien Tixeuil*
*University Pierre et Marie Curie - Paris 6, LIP6-CNRS 7606, France.
†LIF, Aix-Marseille University & CNRS, France.

Abstract—We study distributed coordination among autonomous mobile robots, focusing on the problem of gathering the robots at a single location. The gathering problem has been solved previously using deterministic algorithms even for robots that are anonymous, oblivious, disoriented, and operate in the semi-synchronous ATOM model. However, these solutions require all robots to be fault-free. The recent results of Agmon and Peleg [1] show how to gather all correct robots when one of the robots may crash permanently. We study gathering in $n$-robot systems with $f$ crashes for any $f < n$. In such a scenario, no robot can wait for another robot, i.e., the algorithm must be wait-free. We provide such a wait-free algorithm to gather all correct robots assuming the capabilities of strong multiplicity detection and chirality. Unlike previous solutions, our algorithm does not impose the requirement of initially distinct locations, and works for any arbitrary initial configuration of robots (except the bivalent configuration where deterministic gathering is not possible).

Keywords: Distributed Coordination, Mobile Robots, Gathering, Deterministic, Anonymous, Oblivious, Fault Tolerance.

I. INTRODUCTION

We consider distributed systems consisting of multiple autonomous robots that can move freely on a 2-dimensional plane, observe their surroundings and perform computations. The robots do not communicate explicitly with other robots and there is no central authority that communicates with the robots. Such teams of robots can be deployed in areas inaccessible to humans, to perform collaborative tasks such as search and rescue operations, data collections, environmental monitoring and even extra-terrestrial exploration. From the theoretical point of view, the interest is in determining which tasks can be performed by such robot teams and under what conditions. For example, could the robots arrange themselves in form of a cycle, or a line, or any other given pattern ([21], [22], [14])? One important task useful in multi-robot coordination is gathering the robots at a single location. Since the meeting location is not predetermined, this requires reaching an agreement among the robots to select a unique location for gathering. The gathering problem has been studied in an abstract model where robots are denoted by points on the plane and the objective is to bring them together at any single point of the plane. In this model, robots are assumed to be dimensionless and thus do not block the path or the vision of each other. This is a useful abstraction as it allows the algorithm designer to focus on the problem of symmetry breaking and finding a unique gathering location, without bothering about the mechanical details of actually moving the robots. Several studies have been devoted to the design of algorithms for either convergence (making the robots approach a common location) or gathering (making the robots reach a common location within finite time). The former problem is easier to solve than the latter.

One line of research is to determine the minimum capabilities required by the robots to achieve any given task [23], [14]. A particularly weak model of robots is assumed and additional capabilities are added whenever it is necessary to solve the problem. In our model, the robots are assumed to be anonymous (i.e., indistinguishable from each-other), oblivious (i.e., no persistent memory of the past) and disoriented (i.e., they do not agree on a common coordinate system). The robots operate in Look-Compute-Move (LCM) cycles, where in each cycle a robot Looks at its surroundings and obtains a snapshot containing the locations of all robots as points on the plane with respect to its own location; Based on this visual information, the robot Computes a destination location and then Moves towards the computed location. Since the robots are identical, they all follow the same deterministic algorithm. The algorithm is oblivious if the computed destination in each cycle depends only on the snapshot obtained in the current cycle (and not on the past history of execution). The snapshots obtained by the robots are not consistently oriented in any manner.

The presence of synchronization among the robots significantly affects the solvability of collaborative tasks in this model. Three different levels of synchronization have been considered in previous studies. The strongest model is the fully synchronized (FSYNCH) model where each phase of each cycle is performed simultaneously by all robots. On the other hand, the weakest model, called asynchronous (ASYNCH) allows arbitrary delays between the look, compute and move phases and the movement itself may take an arbitrary amount of time [14], [18], [19]. In this paper, we consider the semi-synchronous (SSYNCH or ATOM) model [1], [3], [13], [22], [23], which lies somewhere between the two extreme models. In the ATOM model, time is discretized into rounds and in each round an arbitrary subset of the robots are active. The robots that are active in a round,
perform exactly one *atomic* Look-Compute-Move cycle in that round. It is assumed that a hypothetical scheduler (an adversary) chooses which robots should be active in any particular round and the only restriction of the scheduler is that it must activate each robot infinitely often in any infinite execution. It was shown [22] that gathering any \( n > 2 \) robots is possible in this model if the robots have *multiplicity detection* capability, i.e., a robot can determine whether any given point in its snapshot is occupied by exactly one robot or more than one robots. A stronger form of multiplicity detection allows a robot to count the exact number of robots located at any given point. If the robots start from mutually distinct locations, a usual algorithm for gathering is to form a single point of multiplicity (e.g. by moving a selected robot to another robot). Using the (weak) multiplicity detection, any robot can uniquely identify the point of multiplicity and all robots can move to this point. However, the algorithm must take care not to form a second point of multiplicity when moving the robots. This can be achieved if the robots move in some specified order depending on their proximity to the destination. For a system of \( n = 2 \) robots, it has been shown [22] that no deterministic algorithm can solve gathering. If the algorithm instructs the robots to move to a common point e.g. the midpoint between the two robots, then the adversary could activate only one of the robots and after the move, the robots would still be in distinct locations. On the other hand, if the algorithm instructs the robot to move to the other robot, then the adversary could activate both robots simultaneously and they would simply swap positions. Based on this logic, it is easy to see why deterministic gathering is not possible, although convergence can be solved. In fact, the convergence problem is easy to solve for any number of robots using the so-called gravitational algorithms [9] that instruct the robots to always move towards the center of gravity. Such algorithms are however not always effective in solving gathering, for the simple reason that the center of gravity is not invariant under movement of a subset of the robots.

**Fault Tolerance:** This paper considers the gathering problem in presence of possible failures. In a large system of robots, some of the robots may invariably fail and we would like to design a fault tolerant algorithm for gathering. Previous work has mainly considered a limited type of faults, namely transient faults [1], [3], [13], [14], [16], [23]. A transient fault implies that a robot may be in an erroneous state (i.e. its memory may be corrupted) but it correctly follows the rules of the algorithm thereafter. Any oblivious algorithm can overcome such failures, since the actions of any robot depends only on the current observations made by the robot. For this reason, oblivious algorithms are often claimed to be self-stabilizing. However, the proposed oblivious algorithms such as the one discussed above, do not work when robots may start from arbitrary locations (e.g. when the initial configuration has multiple points of multiplicity). This observation was made by Dieudonné et al. [13] who presented an algorithm for gathering from any arbitrary configuration, although only for systems consisting of an odd number of robots.

Agmon et al. [1] studied gathering in presence of more permanent faults, such as, when a robot stops moving forever. They considered both crash faults and byzantine faults. A robot is said to have crashed, if it stops participating in the algorithm from some point of time. A robot is said to be byzantine if it behaves in an arbitrary manner and does not follow the rules of the algorithm. (A byzantine robot is supposed to be controlled by some adversary which is trying to make the algorithm fail.) The results of [1] show that a single byzantine fault can prevent the correct robots from gathering even in a system of \( n = 3 \) robots. On positive side, it was shown that in any system of \( n \geq 3 \) robots, gathering can be achieved in the presence of one crash fault. As noticed in that paper, even one single fault can make the known gathering algorithm fail. For example, when the robots are instructed to move in some specific order defined by the algorithm, if one robot crashes all robots that were waiting for this robot would never move, thus creating a deadlock. This problem was avoided in [1] by using an algorithm that always instructs at least two robots to move, thus ensuring progress since there can be at most one crashed robot. This algorithm cannot be trivially generalized to tolerate multiple crash faults. Moreover since the algorithm is once again based on the idea of forming a unique point of multiplicity, it works only if all robots start from initially distinct locations.

We would like to design an algorithm that achieves gathering from arbitrary initial configurations, tolerating up to \( f \) faults in any robot system consisting of \( n > f \) robots. Note that such an algorithm must be wait-free, i.e., no robot can wait for another robot and the algorithm must instruct each robot to move in each step (unless the robot has arrived at the gathering location).

As a consequence of removing the restriction of distinct initial locations, we need to deal with certain initial configurations of robots where gathering is not deterministically solvable. Consider a system of \( n = 2x > 2 \) robots, where the robots are initially located at exactly two distinct points, with \( x = n/2 \) robots at each location. It can be shown (using similar arguments as in the impossibility proof for \( n = 2 \) robots) that deterministic gathering is impossible starting from such an initial configuration, even if all robots are fault-free. We call this particular configuration as the *bivalent* configuration and we show that this is the only initial configuration where gathering is not possible. Starting from any other configuration, we can gather all the correct robots, even when any \( f < n \) robots crash. We achieve these results assuming the additional capabilities of *strong*

\[^{1}\text{This impossibility holds in all models where the robots are identical, irrespective of the capabilities of the robots.}^{\text{\textsuperscript{1}}}\]
multiplicity detection and chirality i.e., a common notion of handedness. The assumption of strong multiplicity detection is a necessary condition for gathering from arbitrary configurations, since otherwise the robots cannot distinguish the bivalent configuration (where gathering is not possible) from any other configuration where robots are unequally distributed among two points (where gathering is possible).

**Our Contribution:** The algorithm presented in this paper combines several techniques, including the concept of Weber points [24]. Given a set of points \( P \), the Weber point \( c \) minimizes \( \sum_{r \in P} \text{distance}(x, r) \) over all points \( x \) in the plane. The Weber point has the key property of remaining unchanged under straight movements of any of the points towards or away from it. If the Weber point can be computed, it is simple to devise a robot protocol that solves gathering: all robots simply move towards the Weber point. Unfortunately, the Weber point cannot be computed by any finite algorithm for an arbitrary set of points. For certain specific configuration of points (e.g. linear [7] or biangular [2] or regular [6] configurations), there exists algorithms for computing the Weber point. In this paper, we present techniques for computing the Weber point of a large class of configurations, which we call quasi-regular – this includes biangular and regular configurations. We also present an algorithm for detecting whether a given configuration is quasi-regular. For configurations that are neither quasi-regular nor linear, we use different techniques. It turns out that such configurations are either completely asymmetric or have only axial symmetry (i.e., mirror symmetry). Using the condition of chirality, we are able to break the symmetry for configurations having axial symmetry and thus, treat them as asymmetric configurations. In any asymmetric configuration, it is possible to elect a robot (or a set of co-located robots) as leader and this location can be used as the gathering point. However since the leader does not remain invariant if the robots move towards it, we need to ensure that gathering is achieved even if the leader may change after each move. We prove that our algorithm achieves gathering of all correct robots, starting from any non-bivalent configuration and tolerating up to \( n - 1 \) crash faults for a system of \( n \) robots.

Note that the chirality condition that we assume in this paper, is a much weaker assumption (and is relatively easier to implement in practice) than the assumption of a common coordinate system. If a common coordinate system is assumed, it is possible to order the robots and overcome the anonymity of the robots, thus making it easy to solve the gathering problem (see e.g. [14]). However in order to have a common coordinate system, the robots need to be equipped with accurate compasses. Achieving robot coordination in spite of inaccuracies in compass readings is already a challenging research topic [16], [20]. The gathering problem has also been studied in models with other limitations, such as robots having limited visibility [3], [15] or robots that are not dimensionless (i.e. they block both the motion and visibility of other robots) [10]. We do not consider such models in this paper.

The rest of the paper explains our algorithm for gathering all correct robots starting from any arbitrary configurations and tolerating \( f < n \) crash faults. The next section describes the model more precisely. In the following section, we define the class of quasi-regular configurations and show how to compute the Weber point for such configurations. Section IV presents a classification of the different types of configurations encountered by the algorithm. Finally in section V we present the complete algorithm and provide a proof of correctness. Due to space limitations, proofs of some of the lemmas have been omitted and they can be found in the full version of the paper [5].

### II. The Model and Notations

**Robot Model:** There are \( n \) robots modelled as points on a geometric plane. A robot can observe its environment and determine the location of other robots in the plane, relative to its own location. All robots are identical (and thus indistinguishable) and they follow the same algorithm. However each robot has its own local coordinates system which may be distinct from that of other robots. The robots only share a common sense of handedness (chirality) i.e., they agree on the clockwise direction. Time is divided to discrete intervals called rounds and in each round a robot can be either active or inactive. In each round, each active robot \( r \) makes exactly one step which consists of LOOK, COMPUTE and MOVE actions. During the LOOK stage, robot \( r \) gets a complete snap-shot containing the locations of every other robot in terms of the local coordinate system and unit distance used by robot \( r \). (Note that multiple robots may occupy the same location in the plane and a robot can determine exactly how many robots are located at the same point.) During the COMPUTE stage, the robot executes its algorithm, using the snapshot as input and determines its next destination point. (Note that robots do not need to remember their previous steps.) During the MOVE stage, the robot moves towards the computed destination\(^2\). A move may end before the robot reaches its destination. However there exists a (small but unknown) constant \( \delta > 0 \) such that if the destination point is closer than \( \delta \), the robot will reach it; otherwise, it will move a distance of at least \( \delta \) towards its destination. A robot that is inactive in a round does not take any actions during that round. We denote the above model of computation as \( \text{ATOM}^M \) model (i.e. the ATOM model enhanced with strong multiplicity detection and chirality).

We consider the crash fault model. A robot is faulty if there is a time at which it stops taking actions (i.e., it crashes). However, a crashed robot remains visible to other robots in the system. A robot that does not crash is correct. Each (correct) robot is active in infinitely many rounds.

\(^2\)If the computed destination is the current location, then the robot does not move.
Notations: \( \mathcal{R} = \{r_1, \ldots, r_n \} \) is the set of robots and \( \mathcal{T} \) is the set of positive natural numbers, denoting time instances. At any time \( \tau \in \mathcal{T} \), the configuration of the set of robots is given by the multiset \( C_\mathcal{R}(\tau) = \{p_1, \ldots, p_n\} \) where each \( p_i \in \mathbb{R}^2 \). We shall drop the subscript \( \mathcal{R} \) when it is obvious from context. Let \( \mathcal{P} \) be the set of all possible configurations of \( n \) robots. Formally, \( \mathcal{P} = \mathbb{R}^{2n} \) where \( \mathbb{R} \) is the set of real numbers. A configuration is linear if all robots lie on the same line. Given any robot \( r \), \( \text{mult}(r) \) denotes the multiplicity of the location occupied by \( r \). Given a multiset of robot positions \( Q \), we denote by \( U(Q) \) the corresponding set of positions in \( Q \) removing multiplicities (i.e. each point in \( U(Q) \) contains at least one robot). Given a configuration \( C \), \( \text{sec}(C) \) denotes the smallest enclosing circle of the point set \( U(C) \). The center of any circle \( G \) is denoted by \( \text{center}(G) \). \( \text{CH}(Q) \) denotes the convex hull of the points in \( Q \). Given two distinct points \( u \) and \( v \) on the plane, let \( \text{line}(u,v) \) denote the straight line passing through these points and \( (u,v) \) denote the open (resp. closed) interval containing all points in this line that lie between \( u \) and \( v \). The half-line starting at point \( u \) (but excluding the point \( u \)) and passing through \( v \) is denoted by \( HF(u,v) \). Formally, \( HF(u,v) = \{ p \in \text{line}(u,v), p \neq v : v \in [u,p) \} \). With respect to some point \( c \in \mathbb{R}^2 \), the angle in the clockwise direction between line segments \( [c,u] \) and \( [c,v] \) is denoted by \( \angle(u,c,v) \). The Euclidean distance between \( u \) and \( v \) is denoted by \( |u,v| \).

III. COMPUTING WEBER POINTS

Definition 1 (Weber Points): The Weber points of a configuration \( C \), denoted \( WP(C) \), are those points that minimize the sum of distances with points of \( C \). Formally, \( WP(C) = \arg \min_{x \in \mathbb{R}^2} \sum_{i=1}^{n} |x,p_i| \).

Non-linear configurations are known to have a unique Weber point while linear configurations may have infinitely many Weber points [7]. The Weber points of a linear configuration \( C \) are points in the interval \([\min(\text{Med}(C)), \max(\text{Med}(C))]\), where \( \text{Med}(C) \) the set of median points. If a linear configuration \( C \) has a single median, then this point is the unique Weber point \( WP(C) \). We now show how to compute the Weber-point of some special non-linear configurations which have some form of symmetry.

A. Symmetries in Configurations

Configurations may exhibit several kinds of symmetry. In this section, we consider a specific form of symmetry called rotational symmetry which we define in a precise sense and show how to quantify it. This is based on the concept of views [22], as described below.

Definition 2 (Views): Let \( C = \{p_1, \ldots, p_n\} \) be a configuration of robots and \( c = \text{center}(\text{sec}(U(C))) \). Given a position \( p \in U(C) \), define the view of \( p \), denoted \( \mathcal{V}(p) \), as the set of points \( \in C \) expressed in the polar coordinate system whose origin \((0,0)\) is \( p \) and whose point \((1,0)\) is defined as follows. If \( c \neq p \), then \((1,0) = c \). Otherwise \((1,0) \) is any point \( x \neq p \in U(C) \) that maximizes \( \mathcal{V}(x) \).

Note that in the definition above, the point \((1,0)\) is not uniquely defined, however the view of any point \( p \in C \) is uniquely defined. Further, there is a total order (e.g. lexicographic) on the set of distinct views. Based on the definition of views, we can define an equivalence relation \( \sim_r \) on the set of robot locations, as follows: \( \forall u, u' \in U(C), (u \sim_r u') \iff (\mathcal{V}(u) = \mathcal{V}(u')) \). The corresponding equivalence class for \( u \) is denoted by \([u]_r \). The following definition formalizes the notion of rotational symmetry.

Definition 3 (Rotational Symmetry): The (rotational) symmetry of a configuration \( C \), denoted \( \text{sym}(C) \), is the cardinality of the biggest equivalence class defined by \( \sim_r \) on \( U(C) \). That is, \( \text{sym}(C) = \max \{|[u]_r| \mid u \in U(C)\} \).

Configuration \( C \) is said to be symmetric if \( \text{sym}(C) = 1 \). Our definition of symmetry differs slightly from the one presented in [11], [22] but only for those configurations that contain either points of multiplicity or a point located in the center of their SEC. The following result follows from the definition of symmetry.

Lemma 3.1: Let \( C \) be a configuration with \( k = \text{sym}(C) > 1 \) and let \( c = \text{center}(\text{sec}(U(C))) \). For every \( u \in U(C) \) with \( (u \neq c) \), it holds that \([u]_r \) is a \( k \)-gon with center \( c \) and whose corners have the same multiplicity.

We now define some weaker forms of symmetry called regularity and quasi-regularity that can be obtained from symmetric configurations when robots move towards the center of symmetry. The following concepts are based on some ideas from [2], [17].

Definition 4: Let \( C = \{p_1, \ldots, p_n\} \) be a configuration and let \( c \in \mathbb{R}^2 \). The clockwise successor of \( p_i \in C \) around \( c \), denoted \( S(p_i, c) \) is the point \( p_j \in C \) defined as follows:

\[
X = \{ p_k \in C \mid (p_k = p_i) \land (k < i) \}
\]

\[
Y = \{ p_k \in C \mid (c, p_k) \}
\]

\[
Z = \{ p_k \in C \mid (\exists p \in C : 0 < \angle(c, p, c) < \angle(p_i, c, p_k)\}
\]

1) If \( X = \emptyset \), then \( p_j = \arg \max_{p_j \in X} |p_j, p_k| \).
2) Otherwise, if \( Y \neq \emptyset \), then \( p_j = \arg \min_{p_j \in Y} |p_i, p_k| \).
3) Otherwise, \( p_j = \arg \max_{p_j \in Z} \{c, p_k\}, \) where \( p_i \in Z \).

Now we can recursively define the \( k \)-th successor of \( p_i \) around \( c \), denoted \( S^k(p_i, c) \) as follows: If \( k > 1 \), \( S^k(p_i, c) = S(S^{k-1}(p_i, c)) \); and \( S^1(p_i, c) = S(p_i, c) \).

The string of angles around \( c \) started in \( p_i \), denoted by \( \text{SA}(p_i, c) \) is the string \( \alpha_1 \ldots \alpha_m \) such that \( m = n - \text{mult}(c) \) and \( \alpha_\ell = \angle(S^{\ell-1}(p_i), c, S^\ell(p_i)) \). The size of \( \text{SA}(p_i, c) \), denoted by \( |\text{SA}(p_i, c)| \) is equal to \( m \). A string \( \text{SA} \) is \( k \)-periodic if it can be written as \( \text{SA} = x^k \) where \( 1 \leq k \leq |\text{SA}| \).
The greatest $k$ for which $SA$ is $k$-periodic is called the periodicity of $SA$ and is denoted by $\text{per}(SA)$.

Definition 5 (Regularity): A configuration $C$ of $n$ points is regular if there exists a point $c \in \mathbb{R}^2$ and $\exists m > 1$ such that $\text{per}(SA(c)) = m > 1$. In this case, the regularity of $C$, denoted $\text{reg}(C)$, is equal to $m$. Otherwise, $\text{reg}(C) = 1$. The point $c$ is called the center of regularity and is denoted by $CR(C)$.

Definition 6 (Quasi Regularity): A configuration $C$ of $n$ points is quasi-regular (or Q-regular) iff there exist (1) a point $c \in \mathbb{R}^2$ and (2) a regular configuration $C'$ with center of regularity $c$ which can be obtained from $C$ by moving only points located at $c$ if any. Formally, $C$ is quasi-regular with center $c \in \mathbb{R}^2$ iff $\exists C' \in \mathcal{P}$ such that $\text{reg}(C') > 1$, $CR(C') = c$ and $\forall p \in C' \setminus C, p = c$. In this case, the quasi-regularity of $C$, denoted $\text{qreg}(C) = \text{reg}(C')$ and the center of quasi-regularity denoted $CQR(C) = c$. If $C$ is not quasi-regular then $\text{qreg}(C) = 1$.

Note that each configuration that is symmetric is also regular. More precisely, $(\text{sym}(C) > 1) \Rightarrow (\text{reg}(C) = \text{sym}(C))$. Each regular configuration is also quasi-regular (with $C' = C$).

B. Weber Points of Quasi-Regular Configurations

The following properties allow us to compute the Weber point of any quasi-regular configuration that is not linear.

Lemma 3.2: If $C$ is a configuration with a unique Weber point $c$ and if $C'$ is a configuration that is obtained from $C$ by moving robots towards $c$, then the Weber point of $C'$ is also unique and is equal to $c$.

Note that every quasi-regular configuration can be obtained from some regular configuration and similarly every regular configuration can be obtained from some symmetric configuration by moving some points towards the center of regularity (resp. center of symmetry). For a symmetric configuration, that is not linear, it is easy to see that the Weber point must be center of the SEC. By the above lemma, we can then show the following.

Lemma 3.3: For every non-linear configuration $C$ that is quasi-regular, $CQR(C) = WP(C)$.

Given a configuration $C$, it is not immediately obvious if $C$ is quasi-regular and we need techniques to detect quasi-regularity of a configuration. To this end we have the following definition.

Definition 7: Let $C$ a configuration with $|U(C)| > 1$ and let $p \in C$. Let $m > 1$. We define $X_m(C, p)$ as the following set of points $\{x \in \text{SEC}(C, p) \mid \exists k \in [1, m] : \exists y \in S(x, p, \frac{2k\pi}{m}) : (p, y) \text{ contains at least one robot}\}$. For each point $x \in X_m(C, p)$, let $\text{LOC}(C, x, p)$ (or $\text{LOC}(C, x)$) denote the number of robots of $C$ that are located in $(p, x)$ and let $\text{OBJ}(C, x)$ denote $\max\{\text{LOC}(C, y)|(y = S(x, p, \frac{2k\pi}{m}) \land (k \in [1, m]))\}$.

Lemma 3.4: Given a configuration $C$ and a point $p \in C$, $C$ is q-regular with center $p$ and $\text{qreg}(C) = m > 1$ iff

$$\text{mult}(p) \geq \sum_{x \in X_m(C, p)} (\text{OBJ}(C, x) - \text{LOC}(C, x)) \quad (1)$$

The above lemma allows the robots to detect quasi-regular configurations and determine the center of quasi-regularity, which, due to Lemma 3.3, is also the Weber point.

Theorem 3.1: Given a non-linear configuration $C$ of $n$ robots, there exists an algorithm that detects if $C$ is quasi-regular and if so, outputs the Weber Point of $C$.

IV. ROBOT CONFIGURATIONS

Recall that the gathering algorithm executed by a robot, takes as input a configuration and the current position of the robot within the configuration, and returns the next destination of the robot. Before presenting the algorithm, we present a classification of the robot configurations which will simplify the algorithm description.

A. Classification

In the following, we formally define five classes of configurations and prove that they constitute a partition of the set $\mathbb{P}$ of all possible configurations of $n$ robots.
• [Bivalent(\mathcal{B})]
  \mathcal{B} = \{C \in \mathbb{P} \mid (\forall u \in U(C) : \text{mult}(u) = n/2)\}. \mathcal{B} is the set of configurations where the robots are equally distributed over two points in the space.

• [Multiple (\mathcal{M})]
  \mathcal{M} = \{C \in \mathbb{P} \mid \exists u \in U(C) : \forall v \neq u \in U(C) : \text{mult}(v) < \text{mult}(u)\}. A configuration \( C \) belongs to \( \mathcal{M} \) if it has a point \( u \) whose multiplicity is greater than that of any other distinct point in \( C \).

• [Collinear(\mathcal{L})]
  \mathcal{L} = \{C \in \mathbb{P} \mid (C \text{ is linear}) \wedge (C \notin \mathcal{B} \cup \mathcal{M})\}. We define the subsets \( \mathcal{L}1W \) and \( \mathcal{L}2W \) of collinear configurations depending on whether their Weber point is unique or not. That is, \( \mathcal{L}1W = \{C \in \mathcal{L} \mid (W(P(C) \text{ is unique})\} \) and \( \mathcal{L}2W = \mathcal{L} \setminus \mathcal{L}1W \).

• [Q^\circ Regular (\mathcal{Q}\mathcal{R})]
  \mathcal{Q}\mathcal{R} = \{C \in \mathbb{P} \mid (\text{argmax}(C) > 1) \wedge (C \notin \mathcal{B} \cup \mathcal{M} \cup \mathcal{L})\}.

• [Asymmetric (\mathcal{A})]
  \mathcal{A} = \{C \in \mathbb{P} \mid (\text{sym}(C) = 1) \wedge (C \notin \mathcal{B} \cup \mathcal{M} \cup \mathcal{L} \cup \mathcal{Q}\mathcal{R})\}.

Let \( X = \{\mathcal{B}, \mathcal{M}, \mathcal{L}, \mathcal{Q}\mathcal{R}, \mathcal{A}\} \). It is easy to see that \( X \) is a partition of \( \mathbb{P} \). By definition, the classes are mutually disjoint. All linear configurations belong to the set \( \mathcal{B} \cup \mathcal{M} \cup \mathcal{L} \). For a non-linear configuration \( C \) either \( \text{sym}(C) > 1 \) which implies \( C \in \mathcal{Q}\mathcal{R} \cup \mathcal{B} \cup \mathcal{M} \), or \( \text{sym}(C) = 1 \) which implies that \( C \in \mathcal{A} \cup \mathcal{B} \cup \mathcal{M} \). Thus \( \bigcup X = \mathbb{P} \).

B. Properties of Configurations

We present the following properties for configurations discussed above. (Proofs can be found in [5])

Lemma 4.1: Let \( C \) be a linear configuration. The following properties hold:
1) \( (|U(C)| = 2) \Rightarrow (C \in B \cup M) \)
2) \( (|U(C)| = 3) \Rightarrow (C \in M \cup L1W) \)
3) \( (C \in L2W) \Rightarrow (|U(C)| \geq 4) \)

We now introduce the concept of safe points within a configuration.

Definition 8 (Safe points): Given a configuration \( C \), a robot position \( p \in C \) is safe iff \( \forall q \in \mathbb{R}^2 \setminus \{p\} : HF(p, q) \) contains at most \( \lceil n/2 \rceil - 1 \) robots of \( C \).

The notion of safe points is important because any safe point can be used as a gathering point without the possibility that the robots form the bivalent \( B \) configuration while moving towards it. We can show the following properties for safe points.

Lemma 4.2: Any non linear configuration contains a safe point.

Lemma 4.3: If \( C \in B \cup L2W \), then \( C \) does not have a safe point.

V. GATHERING ALGORITHM

A. Preliminaries

Given a gathering algorithm \( \mathcal{A} \), for every configuration \( P \) we denote by \( M(P, \mathcal{A}) \) the set of positions of robots that

\[ A \text{ instructs to move when the current configuration is } P. \]

The following lemma is a simple generalization of Lemma 3.1 in [1] that takes into account configurations containing multiplicity points.

Lemma 5.1: A gathering algorithm \( \mathcal{A} \) for \( n \) robots is tolerant against \( f \leq n - 1 \) crashes only if at each configuration \( P, |U(P \setminus M(P, \mathcal{A}))| \leq 1 \).

In other words, there could be at most one location \( c \in U(P) \) such that the robots at \( c \) are allowed to stay in the same position when activated, while all other robots must choose a destination different from the one they are currently occupying. The algorithm must also ensure that robots never reach the configuration \( B \), due to the following impossibility result.

Lemma 5.2: [13] Starting from a configuration of type \( B \), there is no algorithm that achieves gathering even in fault-

\[ \text{free ATOM}[\omega] \] model.

At any time \( \tau \) during the execution of an algorithm, we define \( F(r, \tau) = true \) if robot \( r \) has crashed at time \( t_i \leq \tau \). The set of non-faulty robots at time \( \tau \) is denoted by \( \text{Live}(\mathcal{R}, \tau) = \{r_i \in \mathcal{R} \mid F(r_i, \tau) = false\} \). We say that gathering has been solved by an algorithm \( \mathcal{A} \) if the predicate \( \text{GATHERED}(\mathcal{R}, \tau) \) as defined below, is true.

Definition 9: Given a set of robots \( \mathcal{R} \) that form configuration \( C \) at time \( \tau \), \( \text{GATHERED}(\mathcal{R}, \tau) \) is true iff \( \sum \text{Live}(\mathcal{R}, \tau) = 1 \) and \( (M(C, \mathcal{A}) \cap \text{U}(\text{Live}(\mathcal{R}, \tau)) = \emptyset) \).

The following algorithm WAIT-FREE-GATHER:

Input: \( C \) (The observed configuration during the LOOK phase).
Output: The destination of the robot.

\[
\begin{align*}
(1) & \quad r \leftarrow \text{MY POSITION IN } C \\
(2) & \quad \text{if } C \notin \mathcal{M} \text{ then} \\
(3) & \quad \text{elected } \leftarrow \text{argmax } \text{mult}(p) \\
(4) & \quad \text{if } (r = \text{elected}) \wedge (\exists p \in C : p \in (r, \text{elected})) \text{ then} \\
(5) & \quad \text{return elected} \\
(6) & \quad \text{else} \\
(7) & \quad X \leftarrow \{p \in C : p \notin HF(\text{elected}, r)\} \\
(8) & \quad v \leftarrow \text{argmin} (k : p = S^k(r, \text{elected})) \\
(9) & \quad \text{let } d \in \mathbb{R}^2 \text{ such that} \\
(10) & \quad [d, \text{elected}] = [r, \text{elected}] \text{ and} \\
(11) & \quad \text{let } \text{let } \text{d} \leftarrow [r, \text{elected}, d] = \text{arg}\left(r, \text{elected}, v/3\right) \\
(12) & \quad \text{return } d \\
(13) & \quad \text{if } C \notin \mathcal{Q}\mathcal{R} \cup L1W \text{ then} \\
(14) & \quad \text{return } WP(C) \\
(15) & \quad \text{if } C \notin \mathcal{A} \text{ then} \\
(16) & \quad X \leftarrow \text{the set of safe points in } U(C). \\
(17) & \quad \text{elected } \leftarrow \text{argmax } (\text{mult}(p), \sum_{q \in C, \text{dist}(p, q) = V(p)} \cdot V(p)) \\
(18) & \quad \text{return elected} \\
(19) & \quad \text{if } C \notin L2W \text{ then} \\
(20) & \quad c \leftarrow \text{center}(C) \\
(21) & \quad \text{if } r \notin CH(C) \text{ then} \\
(22) & \quad \text{return } c \\
(23) & \quad \text{else} \\
(24) & \quad \text{let } d \in \mathbb{R}^2 \text{ such that} \\
(25) & \quad [d, c] = [r, c] \text{ and } \text{let } \text{let } \text{d} \leftarrow \text{arg}\left(c, r, d\right) = \Pi/4 \\
(26) & \quad \text{return } d
\end{align*}
\]

Figure 2. Algorithm WAIT-FREE-GATHER:
B. Algorithm WAIT-FREE-GATHER

The algorithm WAIT-FREE-GATHER is presented in Figure 2. The algorithm specifies the set of rules for the robot to compute the next destination when the current configuration is $C$. We have the following cases:

**Configuration $C \in {\mathcal M}$**

Let $c$ be the unique point of maximum multiplicity in $C$. If robot $r$ is located at $c$, it does not move. Otherwise, if there are no robots between $r$ and $c$, robot $r$ moves directly towards $c$ and if not, it does a side-step i.e., it moves to the closest point $d$ on a half-line $HF(c, d)$ such that the angle between half-line $HF(c, d)$ and half-line $HF(c, r)$ is less than or equal to $1/3$ of the angle between half-line $HF(c, p)$ and half-line $HF(c, r)$, for any other robot location $p \in C$. This ensures the robot does not collide with another robot, i.e., it does not create a new point of maximum multiplicity. Note that there may be multiple robots colocated with $r$, these robot may make the same move as $r$. However the value of $\text{mult}(r)$ would never increase unless $r$ reaches the point $c$. Thus, the algorithm ensures that the robots remain in a configuration of type $\mathcal M$ until gathering is achieved.

**Configuration $C \in {\mathcal Q R}$**

By definition, we know that configuration $C$ contains a unique Weber-point $c$ which is also the median and can be computed easily. Each robot $r$ moves directly towards the Weber point $c$ which remains invariant during the movement. Eventually the configuration changes to $\mathcal M$ or a gathered configuration.

**Configuration $C \in {\mathcal A}$**

Since $C$ is not linear, we know that there exists a safe point in $U(C)$. When there are multiple safe points, the algorithm selects a unique point $c$ from among the safe points in $U(C)$. This is always possible since the configuration is asymmetric (i.e., the view of each point is unique). The algorithm chooses the point $c$ based on the multiplicity(c), the sum of distances of all other robots to $c$, and finally the view of $c$ (in this order, and maximizing the first parameter, minimizing the second parameter and maximizing the third parameter). Each robot $r$ moves towards this unique point $c$. We will show that the configuration $C'$ obtained after one step of the algorithm is of type $\mathcal M$, $\mathcal Q R$, $\mathcal L 1 W$, or $\mathcal A$ (but not $B$ or $\mathcal L 2 W$). Further if the next configuration is again of type $\mathcal A$ then either the maximum multiplicity increases or the minimum sum of distance decreases. This ensures that the algorithm converges towards a configuration of type $\mathcal M$ or a gathered configuration.

**Configuration $C \in {\mathcal L 2 W}$**

In this case, there are at least 4 distinct points in the configuration. The algorithm instructs the robots at the two end-points of the line to move away from the line. Any robot that is not located in one of the end-points is instructed to move towards the center of the line. If any of the robots at the end-points move then the next configuration would be non-linear and thus, the algorithm switches to one of the other cases above. Otherwise, if the robots at the end-points never move (i.e., they are crashed) then the configuration remains linear but the sum of distances between correct robots decreases, and the robots would eventually converge to a gathered configuration or a configuration of type $\mathcal M$.

C. **Proof of Correctness**

We now show that starting from any configuration except the bivalent configuration $B$, the algorithm described in Figure 2 eventually forms a gathered configuration. The proof is divided into several parts, each dealing with configurations of a different type.

1) **Configurations of type $\mathcal M$**

**Lemma 5.3:** Let $C(\tau) \in \mathcal M$. There exists a time $\tau' > \tau$ such that $\text{GATHERED}(R, \tau)$=true.

**Proof:** Let $\text{elected}(\tau)$ be the destination chosen by the robots in configuration $C(\tau)$, i.e.,

$$\text{elected}(\tau) = \arg\max_{p \in \mathcal C(\tau)} \text{mult}(p)$$

A robot position $p \in C(\tau)$ is said to be free with respect to $\text{elected}(\tau)$ if no robot is located in the interval $(p, c)$. The lemma follows from the following two claims that we prove below:

**Claim C1:** Let $c = \text{elected}(\tau)$ be the point of maximum multiplicity in the configuration $C(\tau)$. We need to show that $c$ remains the point of maximum multiplicity in $C(\tau + 1)$. In fact we show a stronger result that no two robots that were in distinct locations at $\tau$ can be at the same location at time $\tau + 1$, unless the robots are at $c$. Let us assume the contrary, i.e., let $r_1$ and $r_2$ be robots that occupied distinct locations in $C(\tau)$ but occupy the same location $p \neq c$ in $C(\tau + 1)$. Note that neither of the robots $r_1$ and $r_2$ are located at $c$ at time $\tau$ (since otherwise the algorithm would instruct them to remain at $c$ and they would not be at $p$ at $\tau + 1$).

According to the algorithm any robot $r$ in configuration $C(\tau) \in \mathcal M$ can make two possible moves: (i) either robot $r$ moves directly towards $c$ (Line (5) of algorithm) or, (ii) robot $r$ moves to a point $d$ such that $\text{red}$ is an isosceles triangle with central angle $0 < \theta < \pi/3$ at $c$ (Line (9) of
algorithm). If both the robots \(r_1\) and \(r_2\) both make move of type (i), then they are distinct free points and in this case their paths may not intersect except at \(c\). Otherwise, suppose one of the robots (say \(r_1\)) makes a move of type (ii) directly towards a point \(d\). Consider the triangle \(r_2cd\) and let \(\angle(r_2,c,d) = \theta\). The other robot \(r_2\) is located either on the half-line \(HF(c, r_1)\) or, on a different half-line \(HF(c, r_2)\) which forms an angle greater than \(3\times\theta\) with the half-line \(HF(c, r_1)\) w.r.t. point \(c\). In the second case, the path of robot \(r_2\) will never intersect the line segment between \(c\) and \(d\). In the first case, either robot \(r_2\) is on a free point (and thus, it will move on the line segment \([c, r_2]\) which does not intersect the line segment \([c, d]\) or robot \(r_2\) is not free and thus it makes a move on a line segment parallel to \([c, d]\). In both cases, there is no common point \(p\) in the path of the two robots.

Proof of C2: In this case, if \(c = \text{elected}(\tau)\) is the point of maximum multiplicity in \(C(\tau)\) then \(c\) is the unique point of maximum multiplicity in all subsequent configurations. Whenever a robot on a free point is activated it moves closer to the point \(c\). Whenever a blocked (i.e., not free) robot is activated, at least one robot moves from a blocked position to a free point. Once a robot \(r\) moves to a free point at time \(\tau_i\), it may be blocked in subsequent steps by only robots that moved with robot \(r\) in that same time step (i.e., these robots were live at that time step). Thus an adversary can prevent a non-faulty robot \(r\) from reaching point \(c\) only by changing a live robot to a crashed robot after each step in which robot \(r\) is activated. After a finite time, the adversary will run out of live robots. Thus all live robots will eventually reach \(c\). Once a robot reaches \(c\), the algorithm never instructs the robot to move (since \(c\) is the unique point of maximum multiplicity). Thus, \(\text{GATHERED}(R, \tau_c)\) will be true at that time.

2) Configurations of type \(L1W\):

Lemma 5.4: Let \(C(\tau) \in L1W\). There exists a time \(\tau_c > \tau\) such that either \((C(\tau_c) \in M)\) or, \((\text{GATHERED}(R, \tau_c) = \text{true})\).

Proof: The lemma follows from the following two claims that we prove below:

\[\begin{align*}
\text{C1:} & \quad (C(\tau) \in L1W) \Rightarrow ((C(\tau + 1) \in M \cup L1W) \land (WP(C(\tau + 1)) = WP(C(\tau)))) \\
\text{C2:} & \quad (\forall \tau' \geq \tau : (C(\tau') \in L1W) \land (WP(C(\tau')) = WP(C(\tau))) \Rightarrow (\exists \tau_c \geq \tau : \text{GATHERED}(R, \tau_c) = \text{true}).
\end{align*}\]

Proof of C1: Since \(C(\tau) \in L1W\), it follows that \(WP(C(\tau)) = c\) is unique and \(C(\tau + 1)\) is obtained by moving robots in \(C(\tau)\) towards \(c\) (line 14 of the algorithm). Hence, according to Lemma 3.2, \(WP(C(\tau + 1)) = WP(C(\tau))\) = \(c\). Moreover, \(C(\tau + 1)\) is linear. This, combined with the fact that its Weber-point is unique implies that \(C(\tau + 1)\) cannot be of type \(B\) or \(L2W\). Therefore, \(C(\tau + 1) \in M \cup L1W\).

Proof of C2: Whenever a robot in configuration \(L1W\) is activated it moves towards the Weber-point \(c\) and Weber-point remains invariant due to this movement. Thus, for all configurations \(C(\tau')\) the Weber-point is the same point \(c\). For each non-faulty robot \(r\) the distance between \(r\) and \(c\) decreases every time the robot \(r\) is activated (unless \(r\) is already at \(c\)). Thus all non-faulty robots are at the point \(c\) at some time \(\tau_c\) and \(\text{GATHERED}(R, \tau_c) = \text{true}\).

3) Configurations of type \(QR\):

Lemma 5.5: Let \(C(\tau) \in QR\). There exists a time \(\tau_c > \tau\) such that \((C(\tau_c) \in M \cup L1W) \lor (\text{GATHERED}(R, \tau_c) = \text{true})\).

Proof: Let \(c = WP(C(\tau))\). The lemma follows from the following two claims that we prove below:

\[\begin{align*}
\text{C1:} & \quad (C(\tau) \in QR) \Rightarrow (C(\tau + 1) \in M \cup L1W \cup QR) \land (WP(C(\tau + 1)) = WP(C(\tau))) \\
\text{C2:} & \quad (\forall \tau' \geq \tau : (C(\tau') \in QR) \land (WP(C(\tau')) = WP(C(\tau))) \Rightarrow (\exists \tau_c \geq \tau : \text{GATHERED}(R, \tau_c) = \text{true}).
\end{align*}\]

Proof of C1: Since \(C(\tau) \in QR\), robots that are activated at \(\tau\) move towards \(WP(C(\tau)) = CQR(C(\tau))\) according to line (14) of the code. Hence, since \(C(\tau)\) is Q-regular, the obtained configuration \(C(\tau + 1)\) is Q-regular also with the same center of Q-regularity as \(C(\tau)\) Hence, \(WP(C(\tau + 1)) = CQR(C(\tau + 1)) = CQR(C(\tau)) = WP(C(\tau))\).

As \(C(\tau + 1)\) is Q-regular, it holds according to the definition of configurations \(QR\) that if \(C(\tau + 1) \notin B \cup M \cup L\) then \(C(\tau + 1) \in QR\). Therefore, \(C(\tau + 1) \in B \cup M \cup L\cup QR\). It remains to show that \(C(\tau + 1) \notin B \cup L2W\). For this, it suffices to show that \(C(\tau + 1)\) has a unique Weber point. But this follows Lemma 3.2 and the fact that \(WP(C(\tau))\) is unique.

Proof of C2: Since all the configurations after \(\tau\) are of type \(QR\) and since the Weber point remains invariant after \(\tau\), it follows that all activated robots after \(\tau\) choose the same destination point: \(WP(C(\tau))\). Hence, there is a time \(\tau_c\) at which all live robots have reached this point. That is, \(\text{GATHERED}(R, \tau_c) = \text{true}\).

4) Configurations of type \(A\):

Lemma 5.6: Let \(C(\tau) \in A\). There exists a time \(\tau_c > \tau\) such that \((C(\tau_c) \in M \cup L1W \cup QR \cup A) \lor (\text{GATHERED}(R, \tau_c) = \text{true})\).

Proof: Given a configuration \(C\), let \(\phi(C)\) be the couple of values defined by

\[\phi(C) = \max\{(\text{mult}(p), \frac{1}{\sum_{q \in C} |p, q|}) | p \in C\}\]

The lemma follows from the claims C1 and C3 below. Claim C2 is used to prove C3.

\[\begin{align*}
\text{C1:} & \quad (C(\tau) \in A) \Rightarrow (C(\tau + 1) \in M \cup L1W \cup QR \cup A) \\
\text{C2:} & \quad \text{If } C(\tau) \in A, \text{ then either}
\end{align*}\]
1) \((C(\tau + 1) = C(\tau))\) or,

2) \((\phi(C(\tau + 1)).mult > \phi(C(\tau)).mult)\) or,

3) \((\phi(C(\tau + 1)).mult = \phi(C(\tau)).mult)\) and \\
\((\phi(C(\tau + 1)).sum^{-1} = \phi(C(\tau)).sum^{-1})\)

C3: \((\forall \tau' \geq \tau : C(\tau') \in A) \Rightarrow (\exists c \geq \tau : \text{GATHERED}(R, c) = \text{true}).\)

**Proof of C1:** \(C(\tau) \in A\) means that \(\text{sym}(C(\tau)) = 1\). Hence, each position in \(U(C(\tau))\) has a unique view. This guarantees that the elected position computed in line 17 of the algorithm is unique and every activated robot selects the same point, say \(u\), as the leader. Moreover, \(u\) is safe in \(C(\tau)\). Hence, all activated robots at \(\tau\) move towards the same safe point \(u\) which results in configuration \(C(\tau + 1)\). We observe that \(u\) is also safe in \(C(\tau + 1)\) since for all \(x \in \mathbb{R}^2 \setminus \{u\}\), the number of robots that are located on \(HF(u, x)\) does not increase between \(\tau\) and \(\tau + 1\) (it may even decrease if some of them reach \(u\)). Thus, \(C(\tau + 1)\) contains at least one safe point \(u\). According to Lemma 4.3 this implies that \(C(\tau + 1) \notin B \cup L2W\) which suffices to prove the claim.

**Proof of C2:** Assume that \(C(\tau + 1) = C(\tau)\). Let \(u \in C(\tau)\) be the common elected position chosen by the algorithm (line 17). Hence, by definition, 
\(\sum_{p_i \in C(\tau)} |u, p_i(\tau)| = \phi(C(\tau))\). Since all activated robots at \(\tau\) move to the same destination \(u\), it follows that the resulting configuration \(C(\tau + 1)\) satisfies:

\[
\sum_{p_i \in C(\tau + 1)} |u, p_i(\tau + 1)| \leq \sum_{p_i \in C(\tau)} |u, p_i(\tau)|
\]

Since \((C(\tau + 1) \neq C(\tau))\), there exists at least one robot \(r_i\) whose position at \(\tau + 1\) is distinct from its position at \(\tau\). That is \(p_i(\tau + 1) \neq p_i(\tau)\). Note that since all robots move towards \(u\), it follows that \(p_i(\tau + 1) \in [p_i(\tau), u]\). We distinguish between two cases:

1) \(p_i(\tau + 1) = u\). In this case \(\text{mult}(u)\) is incremented. Note that \(u\) is still safe in \(\tau + 1\) (as shown in the proof of C1). Hence, \(\phi(\tau + 1).mult = \text{mult}(u)^{\tau + 1} \geq \phi(\tau).mult\)

2) \(p_i(\tau + 1) \neq u\), i.e., \(r_i\) is stopped by the adversary before it reaches \(u\). But since the scheduler guarantees to each robot to move by a distance of at least \(\delta\) before it can stop it, it follows that 
\(\phi(\tau + 1).mult = \text{mult}(u)^{\tau + 1} \geq \phi(\tau).mult\) which, combined with the above inequality gives:

\[
\sum_{p_i \in C(\tau + 1)} |u, p_i(\tau + 1)| \leq \phi(C(\tau)).sum^{-1} - \delta
\]

Note that the multiplicity of \(u\) does not decrease even if no robot reaches it, i.e., \(\text{mult}(u)^{\tau + 1} \geq \phi(C(\tau)).mult\).

Note that \(u\) is still safe in \(\tau + 1\) (as shown in the proof of C1). Hence,
\[
\phi(C(\tau + 1)) \geq \text{mult}(u), \frac{1}{\sum_{p_i(\tau + 1) \in C(\tau + 1)} |u, p_i(\tau + 1)|} \geq \phi(C(\tau)).mult.
\]

Therefore, either \((i)\phi(\tau + 1).mult > \phi(\tau)\) or, \((ii)\phi(\tau + 1).mult = \phi(\tau)\) and \(\phi(\tau + 1).sum^{-1} \leq \phi(\tau).sum^{-1} - \delta\).

This proves the claim.

**Proof of C3:** Assume that \((\forall \tau' \geq \tau : C(\tau') \in A)\). We have to prove that:

\[
\exists c \geq \tau : \forall \tau' \geq \tau : C(\tau') = C(\tau)
\]

That is, the configuration does not change after \(\tau_g\) which implies the existence of a time after \(\tau_g\) at which all the live robots lie on the same position. That is \(\exists c \geq \tau_g : \text{GATHERED}(R, c) = \text{true}\).

The claim follows from claim C2 above. There exists a time \(\tau_1 \geq \tau\) after which \(\phi().mult\) cannot increase since the multiplicity of points is upper bounded by \(n\). Moreover, there exists a time \(\tau_2 \geq \tau_1\) after which \(\phi().sum^{-1}\) cannot decrease. Hence, according to claim C2, after time \(\tau_2\), the configuration remains the same and the claim follows by setting \(\tau_c = \tau_2\).

5) Configurations of type \(L2W\):

**Definition 10:** Assume that \(C(\tau)\) is linear. Let \(u(\tau)\) and \(u^+(\tau)\) denote \(\text{min}(U(C(\tau)))\) and \(\text{max}(U(C(\tau)))\) respectively. Denote by \(S^-(\tau)\) and \(S^+(\tau)\) the set of robots located at \(u(\tau)\) and \(u^+(\tau)\) respectively and let \(S^0(\tau) = \mathbb{R} \setminus S^-(\tau) \cup S^+(\tau)\).

If \(C(\tau) \in L2W\), then Lemma 4.1 implies that \(|U(C(\tau))| \geq 4\). Hence, the sets \(S^-(\tau), S^0(\tau)\) and \(S^+(\tau)\) in this case are non empty and pairwise disjoint.

**Lemma 5.7:** If \(C(\tau) \in L2W\), then \(C(\tau + 1) \notin B\).

**Proof:** It suffices to show that \(|U(C(\tau + 1))| \geq 3\). This simply follows from the fact that robots of \(S^-(\tau)\), \(S^0(\tau)\) and \(S^+(\tau)\) occupy distinct positions and these groups remain disjoint when the robots activated at \(\tau\) move towards their destinations (computed in line 20 for \(S^0(\tau)\) and line 24 for \(S^-(\tau)\) and \(S^+(\tau)\)).

**Lemma 5.8:** Assume \(C(\tau) \in L2W\). If at least one robot in \(S^-(\tau) \cup S^+(\tau)\) is activated at \(\tau\), then \(C(\tau + 1) \notin L2W\).

**Proof:** Assume for the sake of contradiction that \(\forall \tau' \geq \tau : (C(\tau') \in L2W \cup B) \land \text{GATHERED}(R, \tau') = \text{true}\).

But since \(C(\tau) \in L2W\) and Lemma 5.7 says that a configuration of type \(B\) cannot come after a configuration of type \(L2W\), it follows that:

345
∀\tau' ≥ \tau : (C(\tau') \in L2W) \land (GATHERED(\mathcal{R}, \tau') = \text{false})

According to Lemma 5.8, this is only possible if the robots located at the endpoints are never activated, which means that they are all faulty. Hence, the center of the configuration \(c = \text{center}(C(\tau))\) remains constant during all the execution and all correct robots eventually reach this point (line 20 of the algorithm). Hence, there is a time \(\tau_c > \tau\) at which all correct robots are located at \(c\). Thus, \(GATHERED(\mathcal{R}, \tau_c) = \text{true}\). A contradiction.

Based on the Lemmas 5.3, 5.5, 5.4, 5.6, and 5.9, we have the following result:

**Theorem 5.1:** The Algorithm \textsc{Wait-Free-Gather} achieves gathering of all correct robots starting from any initial configuration except the Bivalent configuration, even if \(0 ≤ f < n\) robots crash.

**References**


