Fault-tolerant Gathering of Semi-synchronous Robots

Subhash Bhagat
Advanced Computing and Microelectronics Unit
Indian Statistical Institute
203 B.T. Road
Kolkata, India
sbhagat_r@isical.ac.in

Krishnendu Mukhopadyaya
Advanced Computing and Microelectronics Unit
Indian Statistical Institute
203 B. T. Road
Kolkata, India
krishnendu@isical.ac.in

ABSTRACT

This paper addresses the *Gathering* problem which asks robots to gather at a single point which is not fixed in advance, for a set of small, autonomous, mobile robots. The problem is studied for a set of semi-synchronous robots under $SSYNC$ model when the robots may become faulty (crash fault). Depending upon the capabilities of the robots, the algorithms are designed to tolerate maximum number of faults. This work assumes weak multiplicity detection capability of the robots. The contribution of this work is in two folds. First, a distributed algorithm is presented which can tolerate at most $(\lfloor \frac{n}{2} \rfloor - 1)$ crash faults for $n \geq 7$ robots with weak multiplicity detection only. For the second algorithm, it is also assumed that robots know the mobility capacity of all the robots. The algorithm presented here can tolerate at most $(n - 6)$ crash faults for $n \geq 7$ robots.

CCS Concepts

•Theory of computation → Design and analysis of algorithms; Distributed algorithms; Self-organization;

Keywords

Swarm robotics, semi-synchronous robots, oblivious, gathering, crash faults

1. INTRODUCTION

The collective and cooperative behaviours of a set of small, autonomous, inexpensive mobile robots have been the main focus of *Swarm robotics*. The robots are autonomous i.e., they work without any centralized control. In general settings, they are oblivious, homogeneous and anonymous. Since they are oblivious, they do not carry forward any information of their previous computational cycles. Homogeneity means that all the robots have same capabilities and they execute same algorithm. The anonymity of the robots makes them indistinguishable by their nature or identity. The robots do not have any explicit communication abilities. However, implicit communications are achieved via sensing the locations of the robots in the system. They lack of any global coordinate system. Each robot has its own local coordinate system which may differ from others in orientation, directions of axes and unit distance.

Each robot follow same *Look-Compute-Move* cycle repeatedly. First, an active robot takes the snapshot of its surrounding to obtain the locations of the robots w.r.t. its local coordinate system (*Look* phase). This information is used to compute a destination point (*Compute* phase). Finally, the robot moves towards this computed destination point (*Move* phase). The activations of the robots, the timings of the operations and the completion times depend on the scheduler. An asynchronous scheduler ($ASYNC$) does not impose any restriction on the activation of the robots [14]. The timing of the operations and their completion times are unpredictable but finite. A semi-synchronous ($SSYNC$) scheduler discretizes the time into several rounds [16]. In each round, it allows a subset of robots to be activated simultaneously and to operate all together instantaneously. The unpredictability lies in the activated subset of robots in each round. A *fully synchronous* scheduler ($FSYNC$) is the strongest of these schedulers which allows all robots to be activated in all rounds. We assume a fair scheduler which activates each robot infinitely often.

The robots may have some additional capabilities. The *weak multiplicity detection* capability enables a robot to identify the multiple occurrences of robots at a single point. Whereas, *strong multiplicity detection* helps them to count the total number of robots at that location. The robots may have common *chirality* (i.e., clockwise direction). They can also have agreements on the directions and orientations of the axes of their local coordinate systems. The mobility of a robot may be *rigid* or *non-rigid*. In *rigid* motion, a robot reaches its destination without halting in between. In *non-rigid* movements, a robot may be stopped by an adversary before reaching its destination point. However, to guarantee finite time reachability for the robots, it is assumed that a robot always moves at least a distance *minimum*$\{\delta, d\}$ towards its destination point where d is the distance of its destination point from its current position, for some constant $\delta > 0$.

Another aspect of the system comes from the fact that the robots may become faulty at any stage of execution. Three basic types of faults are considered. The *transient* fault corrupts the memory of the robots. The obliviousness of the robots, makes them naturally resilient to this kind

ICDCN '17, January 04-07, 2017, Hyderabad, India

DOI: http://dx.doi.org/10.1145/3007748.3007781

of faults. The *crash* fault impairs the robots to perform any kind of action. However, they physically remains in the system. The *byzantine* fault is a malicious type of fault. A byzantine robot behaves arbitrarily. The algorithms which tolerate any of these faults, should successfully terminate for correct robots in finite time. By (n, f), we denote the model which permits at most f faulty robots among total n robots.

This paper considers the *gathering* problem under crash fault model. The problem is defined as follows: a set of robots is deployed in the two dimensional Euclidean plane. The non-faulty robots should coordinate their movements to meet at a single point, not fixed a priori, in finite time.

1.1 Earlier works

The gathering problem has been studied extensively under different models and considering different capabilities of the robots [11]. The primary goal is to identify minimal sets of constraints needed to solve the problem. In FSYNC model, gathering is solvable even with $f < \frac{n}{3}$ byzantine robots [2]. Under SSYNC model, gathering is deterministically unsolvable in the absence of multiplicity detection and any form of coordinate axes agreement [15]. One of the most significant works in the crash fault model for semi-synchronous robots is presented by Bramas and Tixeuil [6]. The work presented a distributed algorithm for the gathering problem under $(n, n-1)$ crash fault model for semi-synchronous robots with strong multiplicity detection capability. Izumi et al. [13] proved that the problem is deterministically unsolvable in the presence of a single byzantine robot even if robots have agreement in both coordinate axes, with unlimited mobility and they are not oblivious. Defago et al. [10] presented a study of probabilistic gathering under crash faults and byzantine faults considering different types of schedulers. In ASYNC model, gathering is deterministically solvable for $n > 2$ robots with weak multiplicity detection [8]. When robots have limited visibility range, it is shown that gathering is possible if robots have agreements in direction and orientation of the both axes [12]. Bhagat et al. [3] proved that gathering is solvable for asynchronous robots in the presence of arbitrary number of crash faults under one axis agreement even if robots are opaque i.e., they obstruct the visibility of the other robots. Gathering problem has also been studied for fat robots (robots are represented as unit discs) [1, 4, 9, 7].

1.2 Our Contribution

We consider the gathering problem with weak multiplicity detection under $SSYNC$ model when robots may develop crash faults. In these settings, Agmon and Peleg [2] first considered the problem and presented a distributed algorithm to solve the problem which can tolerate a single crash fault. This paper proposes algorithms which solve the problem with more number of admissible crash faults. The contribution of this paper is in two folds. First, it proposes a distributed algorithm to solve the gathering problem for a set of $n \geq 7$ semi-synchronous robots in $(n, \lfloor \frac{n}{2} \rfloor - 1)$ crash fault model. Secondly, the problem is solved for $(n, n-6)$ crash fault robots when robots have the knowledge of δ also.

Following is the organization of the paper: Section 2 states the assumptions of the robot model used in this paper and presents the definitions and notations used to describe the algorithms. Sections 3 and 4 explain the two algorithms for the gathering problem and the corresponding proofs of correctness are given. Finally the section 5 presents the conclusion.

2. GENERAL MODEL AND DEFINITIONS

The system consists of n homogeneous, autonomous, oblivious robots. The robots are represented as points in the two-dimensional Euclidean plane where they can move freely. They do not share any global coordinate system. However, each robot has its own local coordinate system centred at its current position. The directions and orientations of the axes and the unit distance may vary from other robots. They do not share any common chirality. They also lack any form of explicit communication capability. We assume that initially all the robots occupy distinct positions. Each robot has unlimited visibility range. We consider the $SSYNC$ model with some additional assumptions. The movements of the robots are *non-rigid* i.e., a robot moves at least a distance δ towards its destination point, if it does not reach its destination. A robot may develop crash fault at any stage of execution i.e., the model is the (n, f) crash fault model. The robots are endowed with weak multiplicity detection capability.

- **Configuration of the robots:** The set of n robots is denoted by $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$. Let $r_i(t)$ be the point occupied by the robot r_i at time t. A robot configuration is denoted by the multi set $\mathcal{R}(t) = \{r_1(t), \ldots, r_n(t)\}$. Let $\widetilde{\mathcal{R}}$ denote the set of all configurations which contain at most one multiplicity point (a point containing more than one robot on it).

- **Measurement of angles:** Since the robots do not have common chirality, the angle between two given line segments is considered as the angle which is less than or equal to π.

- **Smallest Enclosing Circle:** Let us denote the smallest enclosing circle of the points in $\mathcal{R}(t)$ by $SEC(\mathcal{R}(t))$ and its centre by $\mathcal{O}_t$. We define two sets $C_{out}(t)$ and $C_{int}(t)$ as the collections of robot positions on the circumference of $SEC(\mathcal{R}(t))$ and the robot positions lying strictly within $SEC(\mathcal{R}(t))$ respectively. For each robot $r_i \in \mathcal{R}$ such that $r_i(t) \neq \mathcal{O}_t$, let $rad_i(t)$ denote the half line starting from $\mathcal{O}_t$ (but excluding $\mathcal{O}_t$) and passing through $r_i(t)$ (Figure 1(a)). Let $|rad_i(t)|$ denote the number of distinct robot positions on $rad_i(t)$. Note that $1 \leq |rad_i(t)| \leq n-1$. When there is no ambiguity, we use $SEC(t)$ instead of $SEC(\mathcal{R}(t))$.

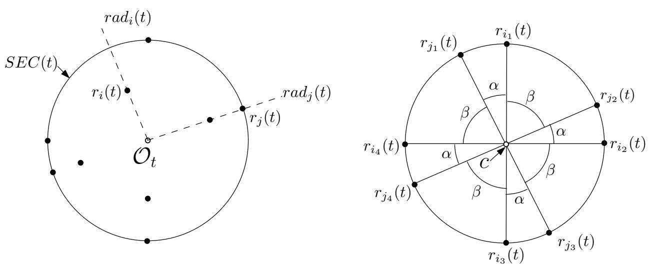
- Let (a, b) and $\overline{ab}$ denote the open and closed line segments joining the points a and b respectively (excluding and including the two end points a and b respectively). Let $|a, b|$ denote the distance between the points a and b. For two sets A and B, by $A \backslash B$, we denote the set difference of A and B.

  We use some concepts defined in [5]. Following is the list of these definitions and notions:

- **View of a robot:** The view $\mathcal{V}(r_i(t))$ of a robot $r_i \in \mathcal{R}$ is defined as the set of polar coordinates of the points in

$\mathcal{R}(t)$, where the polar coordinate system of r_i is defined as follows: (i) the point $r_i(t)$ is the origin of the coordinate system and (ii) the point $(1, 0)$ is $\mathcal{O}_t$ if $r_i(t) \neq \mathcal{O}_t$, otherwise it is any point $r_k(t) \neq r_i(t) \in \mathcal{R}(t)$ that maximizes $\mathcal{V}(r_k(t))$. The orientation of the polar coordinate system should maximize $\mathcal{V}(r_i(t))$. The view of each robot is defined uniquely. The views of two robots are compared in lexicographic order.

- **Rotational symmetry:** Let a relation $\sim$ be defined on $\mathcal{R}(t)$ as follows: $\forall r_i(t), r_j(t) \in \mathcal{R}(t)$, $r_i(t) \sim r_j(t)$ if and only if $\mathcal{V}(r_i(t)) = \mathcal{V}(r_j(t))$ with same orientation. The relation $\sim$ is an equivalence relation and it partitions $\mathcal{R}(t)$ into disjoint equivalence classes. Let $sym(\mathcal{R}(t))$ denote the cardinality of the largest equivalence class defined by $\sim$. The set $\mathcal{R}(t)$ is said to be rotationally symmetric if $sym(\mathcal{R}(t)) > 1$ (Figure 1(b)).

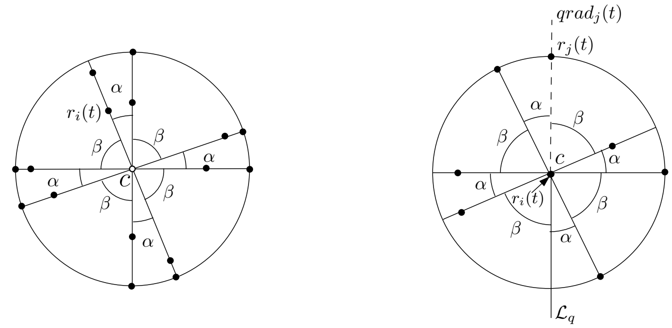


(a) (b)

Figure 1: An example of (a) an illustration of $SEC(t)$, $rad_i(t)$, $\mathcal{O}_t$ and $|rad_j(t)| = 2$ (b) a symmetric configuration with $sym(C) = 4$ where $\{r_{i_1}, r_{i_2}, r_{i_3}, r_{i_4}\}$ and $\{r_{j_1}, r_{j_2}, r_{j_3}, r_{j_4}\}$ are two equivalence classes

- **Successor:** Let us consider a robot configuration $\mathcal{R}(t)$ and a fixed point $c \in \mathbb{R}^2$. Suppose, $S(r_i(t), c)$ denotes the clockwise successor of a point $r_i(t) \in \mathcal{R}(t)$ around c which is defined as follows: if $\mathcal{R}(t) \cap (c, r_i(t)) \neq \phi$, then the point $r_j(t) \in \mathcal{R}(t) \cap (c, r_i(t))$ which minimizes $|r_i(t), r_j(t)|$, is the clockwise successor of r_i. Otherwise, $r_j(t)$ is the point in clockwise direction such that $\angle(r_i(t), c, r_j(t))$ contains no other point of $\mathcal{R}(t)$ and $|c, r_j(t)|$ is maximized. The k^{th} *clockwise successor* of $r_i(t)$ around c, denoted by $S^k(r_i(t), c)$, is defined by the recursive relation: *for* $k > 1$,
$S^k(r_i(t), c) = S(S^{k-1}(r_i(t), c), c)$,
where $S^1(r_i(t), c) = S(r_i(t), c)$ and $S^0(r_i(t), c) = r_i(t)$.
The counter-clockwise successor of r_i can be defined analogously.

- **String of angles:** Let $mult(c)$ denote the number of robots occupying the point c. Let $SA(r_i(t), c)$ denote the string of angles $\alpha_1(t), \alpha_2(t), \ldots, \alpha_m(t)$ where $m = n - mult(c)$ and $\alpha_i(t) = \angle(S^{i-1}(r_i(t)), c, S^i(r_i(t)))$. The length of $SA(r_i(t), c)$ is denoted by $|SA(r_i(t), c)| = m$. The string $SA(r_i(t), c)$ is *k-periodic* if there exists a constant $1 \leq k \leq m$ such that $SA(r_i(t), c) = X^k$, where X is a sub-string of $SA(r_i(t), c)$. The *periodicity* of $SA(r_i(t), c)$, denoted by $per(SA(r_i(t), c))$, is the largest value of k for which $SA(r_i(t), c)$ is *k-periodic* (Figure 2(a)).



(a) (b)

Figure 2: An example of (a) a regular configuration with $reg(C) = 4$ where $SA(r_i(t), c) = (0, \alpha, 0, \beta)^4$ (b) a Q-regular configuration with $qreg(C) = 4$ where the robot at c is r_i and if it is shifted to any point on $\mathcal{L}_q$ the configuration becomes regular

- **Regularity:** A robot configuration $\mathcal{R}(t)$ is said to be *regular* if $\exists$ a point $c \in \mathbb{R}^2$ and an integer m such that $per(SA(r_i(t), c)) = m > 1$ $\forall r_i(t) \in \mathcal{R}(t)$. The *regularity* of $\mathcal{R}(t)$ is denoted by $reg(\mathcal{R}(t)) = m$. The point c is called the centre of regularity (Figure 2(a)).

- **Quasi Regularity:** We consider the quasi regularity only for those configurations in $\widetilde{\mathcal{R}}$ which contains no multiplicity point. A robot configuration $\mathcal{R}(t) \in \widetilde{\mathcal{R}}$ is said to be *quasi regular* or *Q-regular* if and only if $\exists$ a configuration $\mathcal{B}(t)$ and a point $c \in \mathbb{R}^2$ such that $reg(\mathcal{B}(t)) > 1$, c is the centre of regularity of $\mathcal{B}(t)$ and $p \in \mathcal{R}(t) \backslash \mathcal{B}(t)$ implies that $p = c$. In other words, $\mathcal{B}(t)$ can be obtained from $\mathcal{R}(t)$ only by moving the robot position at c, if any, along a particular half line starting at c (including c). Let us denote this half line by $\mathcal{L}_q$. If $\mathcal{R}(t)$ is Q-regular, then the point c is called the centre of Q-regularity. By c_q, we denote the centre of Q-regularity. The quasi-regularity of $\mathcal{R}(t)$ is denoted by $qreg(\mathcal{R}(t)) = reg(\mathcal{B}(t))$. If $\mathcal{R}(t)$ is not quasi-regular, then $qreg(\mathcal{R}(t)) = 1$. For each robot position $r_i(t) \in \mathcal{R}(t)$, let $qrad_i(t)$ denote the half line starting from c (but excluding c) and passing through $r_i(t)$ if $r_i(t) \neq c$. Otherwise, $qrad_i(t)$ is the line segment $\mathcal{L}_q$ (Figure 2(b)).

  If $\mathcal{R}(t)$ is Q-regular, we define an equivalence relation $\sim_q$ on $\mathcal{R}(t)$: $\forall r_i(t), r_j(t) \in \mathcal{R}(t)$, $r_i(t) \sim_q r_j(t)$ iff $qrad_i(t)) = qrad_j(t)$. Let $[qrad_i(t)]$ denote the equivalence class in which $r_i(t)$ belongs to and $|[qrad_i(t)]|$ denote the number of distinct robot positions in this class. Let $e_q(t)$ denote the total number of different equivalence classes defined by $\sim_q$ on $\mathcal{R}(t)$. For a non-linear configuration, the centre of Q-regularity coincides with its unique Weber point and it is computable in finite time [5]. Note that if a configuration has multiple lines of symmetry, then it is Q-regular. However, the converse is not true.

3. ALGORITHM WITH UNKNOWN δ

This section describes the study of the gathering problem under crash fault model when robots do not have the knowledge of δ. To guarantee a finite time gathering in a fault prone system, we have to show that the non-faulty robots in the system find their ways to gather at a point in finite time. While designing the algorithm, the things to be taken into considerations are: (i) since robots have weak multiplicity detection capability, a robot should not collide during the motion to avoid creation of more than one multiplicity point and (ii) there should not be any dead-lock or live-lock in the system. The main idea of the algorithm is to create a unique point of multiplicity where correct robots will gather finally. Different strategies are adopted to create a unique point of multiplicity. Our objective is to design a distributed algorithm which can tolerate at most $\lfloor \frac{n}{2} \rfloor - 1$ crash faults. If in the initial configuration $\mathcal{R}(t_0)$, the set $C_{int}(t_0)$ contains at least two non-faulty robots, the algorithm instructs all of them to move towards $\mathcal{O}_{t_0}$ to create a unique point of multiplicity, avoiding collisions during movements. Otherwise, our strategy is to move robots from the set $C_{out}(t_0)$ within the circle $SEC(t_0)$ to ensure that $C_{int}(t)$ contains at least two correct robots, for some $t > t_0$. Depending upon the initial configuration different scenarios arise.

3.1 Different Configurations

We divide $\widetilde{\mathcal{R}}$ into the following sub-classes:

- **Multiple ($\boldsymbol{\mathcal{M}}$)**: It contains all those configurations which has a unique point of multiplicity.
- **Dense ($\boldsymbol{\mathcal{D}}$)**: All the configurations for which $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$, belongs to this class.
- **Non-Dense ($\boldsymbol{\mathcal{ND}}$)**: It contains all the configurations for which $|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$.

It is easy to see that $\widetilde{\mathcal{R}} = \mathcal{M} \cup \mathcal{D} \cup \mathcal{ND}$.

3.2 Algorithm $GatheringFault()$

We assume that in the initial configuration $\mathcal{R}(t_0)$, all the robots occupy distinct positions. An active robot $r_i \in \mathcal{R}$ first checks in which sub-class $\mathcal{R}(t)$ belongs to and accordingly takes the decision. If r_i finds a unique multiplicity point p_m i.e., $\mathcal{R}(t) \in \mathcal{M}$, then it does one of the followings: (i) if $r_i(t) \neq p_m$, it moves towards p_m using the algorithm $MoveToDestination()$ or (ii) otherwise, it does not move. If there is no such point p_m, the robot r_i executes algorithm $CreateMultiplicity()$ to create a unique multiplicity point. The robots use $MoveToDestination()$ to reach their respective destination points. Algorithm $MoveToDestination()$ takes care of collision-less movements for the robots so that during the whole execution of the gathering algorithm no more than one point of multiplicity can be created.

ALGORITHM 1: GatheringFault()

Input: $r_i \in R$
Output: r_i moves towards its destination.
if $\mathcal{R}(t) \in \mathcal{M}$ **then**
  $r \leftarrow p_m$;
else
  $r \leftarrow CreateMultiplicity(\mathcal{R}(t))$;
end
Move to r using $MoveToDestination(\mathcal{R}(t), r)$;

3.3 Algorithm $MoveToDestination()$

In this section, we describe an algorithm which robots use to reach their respective destination points without colliding with other robots. Since robots have weak multiplicity detection capability, the robots should avoid creation of multiple points of multiplicity. Followings are the different scenarios:

Case-1 Robots have specific destinations: First consider the case when a robot $r_i \in \mathcal{R}$ has a specific destination point. Let w be a destination point of the robot r_i at time t. If current position of r_i coincides with the destination point i.e., $r_i(t) = w$, the robot r_i does not move. Otherwise, the robot r_i uses the following strategies to reach w:

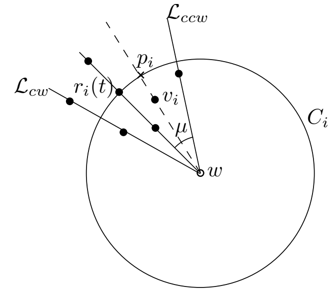


Figure 3: An illustration of scenario (a) in the algorithm $MoveToDestination()$

- If $(r_i(t), w) \cap \mathcal{R}(t) = \phi$ i.e., there is no other robot position in between $r_i(t)$ and w on the line segment $\overline{r_i(t)w}$, then r_i moves straight towards w along $\overline{r_i(t)w}$.
- Otherwise, r_i does the followings:

  (a) Consider all distinct half lines starting from w (excluding w) such that each of them contains at least one robot position from $\mathcal{R}(t)$. Let $\mathcal{L}_i$ be the half line that passes through $r_i(t)$. Let $\mathcal{L}_{cw}$ and $\mathcal{L}_{ccw}$ be the two closest half lines from $\mathcal{L}$ in clockwise and counterclockwise directions (according to the local coordinate system of r_i) respectively. Let μ_{cw} and μ_{ccw} be the two angles made by $\mathcal{L}_{cw}$ and $\mathcal{L}_{ccw}$ with $\mathcal{L}_i$ at the point w. Let $\mu = maximum\{\mu_{cw}, \mu_{ccw}\}$. Let $\mathcal{D}$ be the wedge defined by the angle μ (Figure 3).

  (b) Let v_i be the point in the wedge $\mathcal{D}$ such that $\angle(r_i(t), w, v_i) = \frac{1}{3ls}\mu$, where l is total number of robots on $\overline{r_i(r)w}$ (excluding the robots at w) and s is total number of robots between $r_i(t)$ and w on $\overline{r_i(t)w}$. Consider the circle C_i centred at w and passing through $r_i(t)$. Let C_i intersect $\overline{wv_i}$ at p_i. The robot r_i moves straight towards p_i.

Case-2 Robots want to move inside $SEC(t)$: Consider the case when a robot having position in $C_{out}(t)$, wants move inside $SEC(t)$. Let $r_i \in \mathcal{R}$ be a robot which wants to move inside $SEC(t)$. If $rad_i(t)$ contains only $r_i(t)$, then the destination point of r_i is $\mathcal{O}_t$ and it moves straight towards this point. Otherwise, let $r_j(t)$ be the closest robot position on $rad_i(t)$ from $r_i(t)$. The middle point of $\overline{r_i(t)r_j(t)}$ is the destination point for r_i and it moves towards this point.

3.4 Algorithm $CreateMultiplicity()$

For a robot configuration $\mathcal{R}(t)$, different solution approaches are adopted depending upon the class in which $\mathcal{R}(t)$ belongs.

- **Case–1: $\mathcal{R}(t) \in \mathcal{D}$ ($|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$)**
  Each active robot in $C_{int}(t)$ moves towards $\mathcal{O}_t$, following algorithm $MoveToDestination()$. The robots in $C_{out}(t)$ do not move.

- **Case–2: $\mathcal{R}(t) \in \mathcal{ND}$ ($|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$)**

  - **Case–2.1: $\mathcal{R}(t) \notin \mathcal{QR}$ and $\mathcal{R}(t)$ has no line of symmetry**
    In this case, $|C_{out}(t)| \geq 4$ (since $n \geq 7$). The robot positions in $\mathcal{R}(t)$ are orderable [7]. Let $\mathcal{H}(t)$ be an ordered set of the robot positions in $\mathcal{R}(t)$. Consider the robot $r_i(t) \in C_{out}(t)$ which has highest order in $\mathcal{H}(t)$. Let $\mathcal{L}_i(t)$ be the straight line joining $r_i(t)$ and $\mathcal{O}_t$. Let $v \neq r_i(t)$ be the other point of intersection between $\mathcal{L}_i(t)$ and $SEC(t)$. Let $r_l(t)$ and $r_k(t)$ be the two robot positions in $C_{out}(t)$ such that they lie on two different sides of $\mathcal{L}_i(t)$ and they are closest to v. Let $\mathcal{F}_1(t) = \{r_i(t), r_l(t), r_k(t)\}$. The robots in $\mathcal{F}_1(t)$ do not move. Rest of the robots in $C_{out}(t)$ move inside $SEC(t)$ using strategy stated in case-2 of algorithm $MoveToDestination()$. This will satisfy the condition $|C_{int}(t')| \geq \lfloor \frac{n}{2} \rfloor + 1$, $t' > t$. A robot r_s in $C_{int}(t)$ moves towards $\mathcal{O}_t$.

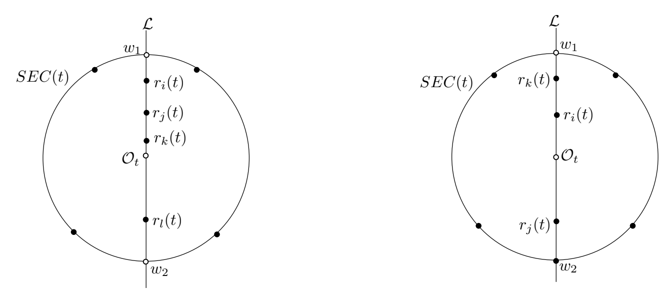
  - **Case–2.2: $\mathcal{R}(t) \notin \mathcal{QR}$ and $\mathcal{R}(t)$ has exactly one line of symmetry**
    Let $\mathcal{L}$ be the line of symmetry. There are two scenarios:

    * **Case–2.2.1: $\mathcal{L}$ passes through at least one point in $C_{out}(t)$**
      First, suppose, $\mathcal{L}$ passes through exactly one point, say $r_i(t)$, in $C_{out}(t)$. Fix $r_i(t), r_l(t)$ and $r_k(t)$ where $r_l(t)$ and $r_k(t)$ are found by same technique as in the asymmetric case above using $\mathcal{L}$ instead of $\mathcal{L}_i(t)$. Let $\mathcal{F}_3(t) = \{r_i(t), r_l(t), r_k(t)\}$. Now, consider the case when $\mathcal{L}$ passes through two robot positions, say $r_c(t)$ and $r_d(t)$ in $C_{out}(t)$. Let $\mathcal{F}_2(t) = \{r_c(t), r_d(t)\}$. The robots having positions in $\mathcal{F}_3(t)$ and $\mathcal{F}_2(t)$ do not move. Rest of the robots in $C_{out}(t)$ move inside $SEC(t)$ and each robot r_i in $C_{int}(t)$ moves towards $\mathcal{O}_t$. They use algorithm $MoveToDestination()$ to reach their destination.

    * **Case–2.2.2: $\mathcal{L}$ does not pass through any point in $C_{out}(t)$**
      There are four points on $C_{out}(t)$ which are closest to $\mathcal{L}$. Let $\mathcal{F}_4(t)$ denote the set of these robot positions. The robots at these four points do not move. Rest of the robots on $C_{out}(t)$ move inside $SEC(t)$. A robot r_i in $C_{int}(t)$ moves towards $\mathcal{O}_t$, except the following two special case:

      (a) For $n = 8$, $|C_{int}(t)| = 4$ and all the robots in $C_{int}(t)$ lie on $\mathcal{L}$. Since there could be 3 faulty robots in $C_{int}(t)$, only a single robot could be able to reach $\mathcal{O}_t$ and the configuration may remain symmetric. This could lead to a dead-lock in the system. To handle this, the robots do the following: Let $\mathcal{L}$ intersects $SEC(t)$ at the point w_1 and w_2 (Figure 4(a)). If the line segments $\overline{\mathcal{O}_t w_1}$ and $\overline{\mathcal{O}_t w_2}$ contain same number of robot positions, the nearest robots from $\mathcal{O}_t$ on these two line segments, move towards $\mathcal{O}_t$ and other two robots move towards w_1 and w_2 respectively. Otherwise, one of the lines, say $\overline{\mathcal{O}_t w_1}$, contains more robot positions than the other one. The robots on these two line segments having free corridors to w_1 and w_2, move towards these points. The other robots in between them move towards the next robot position on $\overline{\mathcal{O}_t w_1}$ and in the direction of w_1. Let d_i be the destination point of a robot $r_i \in C_{int}(t)$. Note that $\mathcal{R}(t)$ remains symmetric during movements of the robots on $\mathcal{L}$.

      (b) For $n = 7$, $|C_{int}(t)| = 3$ and at least one robot position in $C_{int}(t)$ lie on $\mathcal{L}$. In this case, $\mathcal{L}$ contains either exactly one robot position or three robot positions of $C_{int}(t)$. Suppose that exactly one robot lies on $\mathcal{L}$. This robot on $\mathcal{L}$ moves towards nearest among w_1 and w_2 (tie is broken arbitrarily). Other robots in $C_{int}(t)$ move towards $\mathcal{O}_t$. If three robots lie on $\mathcal{L}$, two robots among them have free corridors to w_1 and w_2 and they move towards these points (Figure 4(b)). The third robot, lying on $\mathcal{L}$, moves towards the nearest robot position on $\mathcal{L}$ (tie is broken arbitrarily). Let d_i be the destination point of a robot $r_i \in C_{int}(t)$.



(a) (b)

Figure 4: An illustration of (a) special case of $n = 8$ and (b) special case of $n = 7$ in the algorithm $CreateMultiplicity()$

  - **Case–2.3: $\mathcal{R}(t) \in \mathcal{QR}$**
    Each active robot r_i has c_q as its destination point if $(r_i(t), c_q) \cap \mathcal{R}(t) = \phi$ and it moves towards this point along $qrad_i(t)$. Otherwise, r_i does not move.

3.5 Correctness of $GatheringFault()$

ALGORITHM 2: CreateMultiplicity()

```
Input: R(t)
Output: A destination point.
if |C_int(t)| ≥ ⌊n/2⌋ + 1 then
    if r_i(t) ∈ C_out(t) then
        r ← r_i(t);
    else
        r ← O_t ;
    end
else
    if R(t) ∈ QR then
        if (r_i(t), c_q) ∩ R(t) = ∅ then
            r ← c_q ;
        else
            r ← r_i(t) ;
        end
    else
        if R(t) is asymmetric then
            if r_i(t) ∈ F(t) then
                r ← r_i(t);
            else
                if (r_i(t), w) ∩ R(t) = φ then
                    r ← O_t;
                else
                    r_j(t) ← nearest robot position from
                    r_i(t) on rad_i(t);
                    r ← middle{overline(r_i(t), r_j(t))};
                end
            end
        else
            if r_i(t) ∈ F_2(t) ∨ F_3(t) ∨ F_4(t) then
                r ← r_i(t);
            else
                if (r_i(t), w) ∩ R(t) = φ ∧ n ≠ 7, 8 then
                    r ← O_t;
                else
                    if n ≠ 7, 8 then
                        r_j(t) ← nearest from r_i(t) on
                        rad_i(t);
                        r ← middle{overline(r_i(t), r_j(t))};
                    else
                        r ← d_i
                    end
                end
            end
        end
    end
end
return r;
```

In this section, it is proved that the non-faulty robots shall be able to gather in finite time, by executing algorithm $GatheringFault()$, when there are at most ($\lfloor \frac{n}{2} \rfloor$-1) crashed robots and during the whole execution of the algorithm no collision occurs, where $n \geq 7$.

Lemma 1. *Algorithm $MoveToDestination()$ provides collision free movements for the robots during the whole execution of algorithm $GatheringFault()$.*

Proof. According to algorithm $GatheringFault()$, a moving robot in $\mathcal{R}$ has one of the followings as its destination point: (i) a specific destination point (either $\mathcal{O}_t$ or c_q or a point on the line of symmetry) or (ii) a point inside $SEC()$.

Let $r_i \in \mathcal{R}$ be a robot which moves towards its destination point, say w, according to $MoveToDestination()$. Followings are two cases in which the directions of movements of all the robots, in a particular round, are towards the same point: (i) $(r_i(t), w) \cap \mathcal{R}(t) = \phi$ and (ii) a robot $r_j \in C_{out}(t)$ wants to move inside $SEC(t)$. For the second case, if $(r_j(t), w) \cap \mathcal{R}(t) \neq \phi$, the robot r_j stops far enough from its next robot on the corresponding line segment $rad_j(t)$. Thus the robots, executing these two moves in a particular round, do not collide.

If $\overline{(r_i(t), w)} \cap \mathcal{R}(t) \neq \phi$, the robot r_i shifts from the line segment $\overline{r_i(t)w}$. This shifted destination point of r_i is in the one third sub-sector of the sector defined by μ and closest to $\overline{r_i(t)w}$. Thus, the movement of r_i does not obstruct the free movements of the other robots moving towards w. If $\overline{r_i(t)w}$ contains more than one robot position (excepting at w), algorithm $MoveToDestination()$ divides the closest one third sub-sector of μ among these robots, according to their distances from w. Furthermore, the paths towards the destination points of these robots do not intersect each other (since the lines joining the robots to the destination points lie in different concentric circles centred at w). This implies that, these robots do not collide during their shifting. At the new position $r_i(t')$, the robot r_i has one of the two possibilities (i) r_i has a free corridor to w or (ii) $(r_i(t'), w) \cap \mathcal{R}(t') \neq \phi$. For the second case, r_i has to shift again to find a free corridor towards w. Since there are finite number of robots, after at most finite number of shifts r_i would find a free corridor to w.

Hence, $MoveToDestination()$ guarantees a collision free arrivals to the destination points for the robots. □

Lemma 2. *Algorithm $CreateMultiplicity()$ creates a unique multiplicity point in finite time when the maximum number of admissible crashed robots is $\lfloor \frac{n}{2} \rfloor - 1$ and $n \geq 7$.*

Proof. Let us consider a robot configuration $\mathcal{R}(t) \notin \mathcal{M}$, $t \geq t_0$.

- **Case–1: $\boldsymbol{\mathcal{R}(t) \in \mathcal{D}}$ $\boldsymbol{(|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1)}$**
  Since robots in $C_{out}(t)$ do not move, $SEC(t)$ does not change. There could be at most $\lfloor \frac{n}{2} \rfloor - 1$ faulty robots in $C_{int}(t)$. Since $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$ and $n \geq 7$, there are at least two non-faulty robots in $C_{int}(t)$. Algorithm $MoveToDestination()$ guarantees collision free reachability of the robots to the destination points. According to the algorithm, $\mathcal{O}_t$ is the destination point for each robot in $C_{int}(t)$. Since $C_{int}(t)$ contains at least two non-faulty robots, $\mathcal{O}_t$ will become a multiplicity point in finite time and by lemma 1, it is unique.

- **Case–2: $\mathcal{R}(t) \in \mathcal{ND}$ ($|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$):**

  - **Case–2.1 and Case–2.2: $\mathcal{R}(t) \notin \mathcal{QR}$ and $\mathcal{R}(t)$ has at most one line of symmetry**
    First, consider the scenarios when $n \geq 9$. At most four non-faulty robots retain their positions in $C_{out}(t)$ and rest of the active robots on $C_{out}(t)$ move inside $SEC(t)$ along the corresponding $rad()$. The robots in $C_{int}(t)$ move to $\mathcal{O}_t$ according to $MoveToDestination()$. Due to the movements of the robots, any one of the following scenarios will occur:

    (i) a unique multiplicity point is created at $\mathcal{O}_t$.
    (ii) the condition $|C_{int}(t')| \geq \lfloor \frac{n}{2} \rfloor + 1$ is satisfied.
    We are done in of the above two cases.
    (iii) $\mathcal{R}(t')$ is Q-regular and $|C_{int}(t')| < \lfloor \frac{n}{2} \rfloor + 1$.
    This case is discussed below.
    (iv) $\mathcal{R}(t')$ has at most one line of symmetry with $|C_{int}(t')| < \lfloor \frac{n}{2} \rfloor + 1$.

    In this case, the robots repeat their actions.

    Since, at least two robots (if there are exactly two robots, then they are diametrically opposite) retain their positions on the circumference of $SEC(t)$, until any one of the above scenarios (i), (ii) or (iii) is satisfied, the circle $SEC(t)$ does not change. Since $n \geq 9$ and there could be at most $\lfloor \frac{n}{2} \rfloor - 1$ faulty robots, within finite time, either $C_{int}(t)$ contains at least two non-faulty robots or $\mathcal{O}_t$ becomes the unique multiplicity point.

    Now consider the cases when $7 \leq n \leq 8$. The above arguments are also valid for $7 \leq n \leq 8$ other than the special scenarios as considered in the algorithm section 3.4 (case-2.2.2). For the special scenarios, since $C_{int}(t)$ contains at least one non-faulty robot, after a finite time, either (i) a point of multiplicity is created on $\mathcal{L}$ by the robots on it or (ii) $\mathcal{L}$ would contain at least one robot position in $C_{out}(t')$, $t' > t$. The scenario (ii) is same as the case-2.2.1 in section 3.4. Thus, a unique multiplicity point would be created in finite time. Note that in the special cases, the multiplicity point could be created at the point w_1 or w_2.
  - **Case–2.3: $\mathcal{R}(t) \in \mathcal{QR}$**
    In this case, the centre of Q-regularity c_q is the initial destination point of the moving robot. Since $|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$, the value of $e_q(t)$ satisfies; $\lfloor \frac{n}{2} \rfloor \leq e_q \leq n$. If $e_q = n$, each robot has free corridor towards c_q and it moves straight towards this point. Since, c_q is the Weber point of $\mathcal{R}(t)$ and robots move along straight lines towards it, this point remains invariant under the movements of the robots. Now, suppose that $\lfloor \frac{n}{2} \rfloor \leq e_q(t) < n$. Since the maximum number of faulty robots is $\lfloor \frac{n}{2} \rfloor - 1$, there are three possibilities: (i) $C_{int}(t)$ contains at least two non-faulty robots which has free corridors to c_q or (ii) there is at least one $qrad_i(t)$ containing at least two non-faulty robots such that they are closest to $\mathcal{O}_t$ than the other robot positions on $qrad_i(t)$ or (iii) $C_{out}(t)$ contains at least one non-faulty robot which has a free corridor to c_q (e.g., all the non-faulty robots on the $qrad()$s containing multiple robot positions, are blocked by the faulty robots). In the scenario (iii), the circle $SEC(t)$ does not change due to the movements of the robots in $C_{out}(t)$ (a subset of robots in $C_{out}(t)$ which do not move, are rotationally symmetric and they keep the circle intact). Thus, in finite finite, either c_q will become the unique multiplicity point or the condition $|C_{int}()| \geq \lfloor \frac{n}{2} \rfloor + 1$ will be satisfied. □

Lemma 3. *Algorithm $GatheringFault()$ solves the gathering problem in $(n, \lfloor \frac{n}{2} \rfloor - 1)$ crash fault model for $n \geq 7$ semi-synchronous robots (initially placed at different positions) with weak multiplicity detection in finite time.*

PROOF. Follows from lemma 1 and 2. □

Theorem 1. *The gathering problem is solvable in $(n, \lfloor \frac{n}{2} \rfloor - 2)$ crash fault model for $n \geq 3$ semi-synchronous robots (initially placed at different positions) with weak multiplicity detection in finite time. Further, if $n \geq 7$, gathering is possible in $(n, \lfloor \frac{n}{2} \rfloor - 1)$ crash fault model.*

PROOF. If $n \leq 6$, the maximum number of admissible faults i.e., $(\lfloor \frac{n}{2} \rfloor - 2)$ is 1. The results of [2] imply that the problem is solvable for $n \leq 6$ robots. Again, for $n \geq 7$, the maximum number of admissible faults is more than one and the lemma 3 implies that gathering is possible with at most $\lfloor \frac{n}{2} \rfloor - 1)$ faulty robots. Hence the theorem. □

4. FAULT TOLERANT ALGORITHM WITH KNOWN δ

Different assumptions on the capabilities of the robots help them to solve a variety of problems. Without these assumptions, those problems are either proven impossible or no deterministic solutions are known. Whenever prices are paid in terms of these assumptions, a natural question is asked? What are we gaining in returns? In this section, we consider one of such assumption and see its effect on the solutions of the gathering problem. We assume that each robot has the knowledge of δ. In return, it helps the robots to solve the gathering problem in $(n, n-6)$ crash fault model with multiplicity detection capability and semi-synchronous scheduler ($SSYNC$ model).

4.1 Configurations

We divide $\widetilde{\mathcal{R}}$ into following classes:

- **Multiple ($\mathcal{M}$) :** This class contains all the configurations which have a unique multiplicity point.
- **Q-regular ($\mathcal{QR}$) :** Each configuration in this class is Q-regular.
- **Dense-In ($\mathcal{DI}$) :** All the configurations for which the condition $|C_{out}(t)| \leq 4 \wedge \neg\mathcal{QR}$ is satisfied.
- **Dense-Out ($\mathcal{DO}$) :** All the configurations for which the condition $|C_{out}(t)| > 4 \wedge \neg\mathcal{QR}$ is satisfied.

The configurations in both the classes $\mathcal{DI}$ and $\mathcal{DO}$ have at most one line of symmetry. It is easy to see that $\widetilde{\mathcal{R}} = \mathcal{M} \cup \mathcal{QR} \cup \mathcal{DI} \cup \mathcal{DO}$.

4.2 Algorithm $GatheringFault_\delta()$

We assume that $n \geq 7$ and in the initial configuration $\mathcal{R}(t_0)$, all the robots have distinct positions. If an active robot at time t finds that $\mathcal{R}(t) \in \mathcal{M}$, it sets the multiplicity point p_m as its destination point. Otherwise, robots adopt algorithm $CreateMultiplicity_\delta()$ to create a unique multiplicity point. Until a multiplicity point is created, algorithm $CreateMultiplicity_\delta()$ maintains one of the two invariants (i) if the configuration is Q-regular, it remains Q-regular (ii) if the configuration is not Q-regular or does not become Q-regular due to the movements of the robots, the smallest enclosing circle $SEC(t_0)$ of the initial configuration does not change. The robots move according to algorithm $MoveToDestination_\delta()$. This algorithm provides a collision free movements for the robots.

ALGORITHM 3: $GatheringFault_\delta()$

```
Input: r_i ∈ R
Output: r_i moves towards its destination.
if R(t) ∈ M then
 | r ← p_m;
else
 | r ← CreateMultiplicity_δ(R(t));
end
Move to r using MoveToDestination_δ(R(t), r) ;
```

4.3 Algorithm $MoveToDestination_\delta()$

Algorithm $MoveToDestination_\delta()$ should provide collision free movements for the robots. The algorithm takes the advantage of known δ. Let w' be the destination point for a robot r_i at time t. The robot r_i moves in any one of the following ways:

- If r_i has a free corridor to w' i.e., $(r_i(t), w') \cap \mathcal{R}(t) = \phi$, robot r_i moves towards w' along the line segment $\overline{r_i(t)w}$. If $dist(r_i(t), w') \leq \delta$, robot r_i moves directly to w' even if $(r_i(t), w') \cap \mathcal{R}(t) \neq \phi$.
- Suppose, $(r_i(t), w') \cap \mathcal{R}(t) \neq \phi$. Let $\mathcal{L}$ be the line joining $r_i(t)$ and w'. Let r_j be the nearest robot from r_i on $\mathcal{L}$, lying in between $r_i(t)$ and w'. If $dist(r_i(t), r_j(t)) > \frac{\delta}{2}$, the destination point of r_i is the point $r_i(t') \in (r_i(t), r_j(t))$ where $dist(r_i(t'), r_j(t)) = \frac{\delta}{2}$. If $dist(r_i(t), r_j(t)) \leq \frac{\delta}{2}$, robot r_i does one of the followings:

    1. If $(r_i(t), w') \cap \mathcal{R}(t) = 1$, robot r_i moves a distance $minimum\{\delta, dist(r_i(t), w')\}$ along $\mathcal{L}$ towards w'.

    2. Let $(r_i(t), w') \cap \mathcal{R}(t) > 1$. If $(r_j(t), w') \cap \mathcal{R}(t) > 1$, let $r_k(t)$ be the nearest robot position from $r_j(t)$ on $\mathcal{L}$ lying in between $r_j(t)$ and w'. Otherwise, we consider $r_k(t) = w'$. The destination point of r_i is computed as follows:

        (a) If $dist(r_j(t), r_k(t)) > \delta$, robot r_i moves a δ distance towards w'.

        (b) If $\frac{\delta}{2} < dist(r_j(t), r_k(t)) < \delta$, robot r_i moves a distance $dist(r_i(t), r_j(j)) + \frac{1}{2}dist(r_j(t), b)$, where b is the point in $(r_j(t), r_k(t))$ such that $dist(b, r_k(t)) = \frac{\delta}{2}$.

        (c) If $dist(r_j(t), r_k(t)) \leq \frac{\delta}{2}$, robot r_i moves a distance $dist(r_i(t), r_j(j)) + \frac{1}{2}dist(r_j(t), r_k(t))$.

ALGORITHM 4: $MoveToDestination_\delta()$

```
Input: R(t), w'
Output: Move towards the destination.
if r_i(t) = w' then
 | r ← r_i(t) ;
else
 | if (r_i(t), w') ∩ R(t) = φ ∨ dist(r_i(t), w') ≤ δ then
 |  | move directly to w' along r_i(t)w';
 | else
 |  | r_j(t) ← nearest robot position from r_i(t) on r_i(t)w';
 |  | r_k(t) ← nearest robot position from r_k(t) on r_i(t)w', if any, otherwise w';
 |  | if dist(r_i(t), r_j(t)) > δ/2 then
 |  |  | r ← the point u in (r_i(t), r_j(t)) such that dist(u, r_j(t)) = δ/2;
 |  | else
 |  |  | if (r_i(t), w') ∩ R(t) = 1 then
 |  |  |  | r ← the point u' on r_i(t)w' at a distance minimum{δ, dist(r_i(t), w')} r_i(t));
 |  |  | else
 |  |  |  | if dist(r_j(t), r_k(t)) > δ then
 |  |  |  |  | r ← the point v on r_i(t)w' at a distance δ;
 |  |  |  | else
 |  |  |  |  | if δ/2 < dist(r_j(t), r_k(t)) < δ then
 |  |  |  |  |  | r ← the point u' on r_i(t)w' at a distance dist(r_i(t), r_j(j)) + 1/2 dist(r_j(t), b) from r_i(t), where b is the point in (r_j(t), r_k(t)) such that dist(b, r_k(t)) = δ/2;
 |  |  |  |  | else
 |  |  |  |  |  | r ← the point v' on r_i(t)w' at a distance dist(r_i(t), r_j(j)) + 1/2 dist(r_j(t), r_k(t)) from r_i(t)
 |  |  |  |  | end
 |  |  |  | end
 |  |  | end
 |  | end
 | end
end
move directly to r along r_i(t)r;
```

4.4 Algorithm $CreateMultiplicity_\delta()$

- **Case–1: $\mathcal{R}(t)$ is Q-regular ($\mathcal{QR}$)**
  Each active robot sets c_q as its destination point. The robots which have free corridors to c_q, move towards

this point along corresponding $qrad()$s. Otherwise, the robots follow algorithm $MoveToDestination_\delta()$ to reach c_q.

- **Case–2: $\mathcal{R}(t)$ is not Q-regular:** We have following scenarios:
  - **Case–2.1: $\mathcal{R}(t) \in \mathcal{DO}$ ($|C_{out}| \leq 4$)**
    The Robots on $C_{out}(t)$ do not move. The destination point of the robots in $C_{int}(t)$ is $\mathcal{O}_t$. They follow $MoveToDestination_\delta()$ to reach their destination points.
  - **Case–2.2: $\mathcal{R}(t) \in \mathcal{DI}$ ($|C_{out}| > 4$)**
    Since $\mathcal{R}(t)$ is not Q-regular, $\mathcal{R}(t)$ has at most one line of symmetry. In this case, the robots follow same strategies as described in the algorithm $CreateMultiplicity()$ of section 3.4 for Case–2.1 and Case–2.2 (excluding the two special cases of Case–2.2.2 when $n = 7$ and $n = 8$). However, two differences are there (i) the active robots on $C_{out}(t)$ which want to move inside $SEC(t)$ have $\mathcal{O}_t$ as their destination point and (ii) the robots follow $MoveToDestination_\delta()$ to reach their destination instead of $MoveToDestination()$.

ALGORITHM 5: $CreateMultiplicity_\delta()$

```
Input: R(t)
Output: A destination point.
if R(t) ∈ QR then
    r ← c_q ;
else
    if |C_out(t)| ≤ 4 then
        if r_i(t) ∈ C_out(t) then
            r ← r_i(t);
        else
            r ← O_t ;
        end
    else
        if R(t) has no line of symmetry then
            if r_i(t) ∈ F_1(t) then
                r ← r_i(t);
            else
                r ← O_t;
            end
        else
            if r_i(t) ∈ F_2(t) ∨ F_3(t) ∨ F_4(t) then
                r ← r_i(t);
            else
                r ← O_t;
            end
        end
    end
end
return r;
```

4.5 Correctness of $GatheringFault_\delta()$

In this section, it is proved that $GatheringFault_\delta()$ solves the gathering problem with at most $(n-6)$ faulty robots when $n \geq 7$.

Lemma 4. *Algorithm $MoveToDestination_\delta()$ provides collision free movements during the whole execution of algorithm $GatheringFault_\delta()$.*

Proof. Two robots collide when they meet at an undesirable point. During the execution of $GatheringFault_\delta()$, in a particular round, all the active robots have exactly one desired destination point, either c_q or $\mathcal{O}_t$. Let w' denote the destination point for the active robots in a particular round. The robots move towards w' along lines segments joining their positions to this point. Thus, the different paths of the robots towards w' meet only at this point. We show that two robots on a same path do not land up at a single point other than w' when they try to reach this point. Let r_i and r_j be two robots on a line segment $\mathcal{L}$ joining their positions to w' such that $dist(r_i(t), w') > dist(r_j(t), w')$. Until, r_i crosses r_j on $\mathcal{L}$ their is no possibility of collision. Again, if there is at least one robot position, say $r_s(t)$, in between r_i and r_j on $\mathcal{L}$ or $dist(r_i(t), r_j(t)) > \frac{\delta}{2}$, the destination point of r_i falls into the interval $(r_i(t), r_j(t))$. Thus consider the case when there is no other robot positions in between r_i and r_j on $\mathcal{L}$ and $dist(r_i(t), r_j(t)) \leq \frac{\delta}{2}$.

- If $(r_i(t), w') \cap \mathcal{R}(t) = 1$, robot r_i moves a distance $minimum\{\delta, dist(r_i(t), w')\}$. Also, robot r_j moves a distance $minimum\{\delta, dist(r_j(t), w')\}$. Since r_i and r_j are at different positions on $\mathcal{L}$ and they move along this line segment, the only possible meeting point for them is w'.

- Let $(r_i(t), w') \cap \mathcal{R}(t) > 1$. Consider the point $r_k(t)$, as defined in the algorithm. If $dist(r_j(t), r_k(t)) > \delta$, robot r_i moves exactly a δ distance from $r_i(t)$ whereas r_j moves at least δ distance from $r_j(t)$. Since $r_i(t) \neq r_j(t)$, they do not collide. Let $dist(r_j(t), r_k(t)) < \delta$. If $dist(r_j(t), r_k(t)) \leq \frac{\delta}{2}$, robot r_j crosses $r_k(t)$ whereas destination point of r_i falls into the interval $(r_j(t), r_k(t))$ and the robots do not collide. Otherwise, the destination point of r_i falls into the interval $(r_j(t), b)$ whereas the destination point of r_j is the point $b \in (r_j(t), r_k(t))$ where $dist(b, r_k(t)) = \frac{\delta}{2}$. Thus, r_i and r_j do not collide. □

Lemma 5. *Algorithm $CreateMultiplicity_\delta()$ creates a unique multiplicity point in finite time when the number of faulty robots is at most $(n-6)$ and $n \geq 7$.*

Proof. Let us consider a robot configuration $\mathcal{R}(t) \notin \mathcal{M}$, $t \geq t_0$. Algorithm $CreateMultiplicity_\delta()$ maintains one of the two invariants (i) if the initial configuration is Q-regular, it remains Q-regular (ii) otherwise, if the configuration does not become Q-regular, the smallest enclosing circle $SEC(t_0)$ of the initial configuration remains the same. Thus, in each round, all active robots have exactly one desired destination point either c_q or $\mathcal{O}_t$.

- **Case–1: $\mathcal{R}(t)$ is Q-regular ($\mathcal{QR}$)**
  Each active robot has c_q as its destination point and it moves along corresponding $qrad()$s towards this point. Since c_q is the Weber point of $\mathcal{R}(t)$, it remains invariant under the movements of the robots towards it along straight lines. The robots follow algorithm $MoveToDestination_\delta()$ to reach c_q along corresponding $qrad()$s and it guarantees collision free movements

of the robots. Since there are at least 6 non-faulty robots in the system and they do not wait for each other, the point c_q becomes the unique multiplicity point in finite time.

- **Case–2: $\mathcal{R}(t)$ is not Q-regular:**
In this case, at most four active robots retain their positions on $C_{out}(t)$ (provided $\mathcal{R}(t)$ does not become Q-regular). Thus, $C_{int}(t')$ will contain at least two non-faulty robots, $t' \geq t$. If $\mathcal{R}(t)$ does not become Q-regular due to the movements of the robots, the circle $SEC(t)$ remains the same (since at least two robots in $C_{out}(t)$ which define the $SEC(t)$, do not move) and thus $\mathcal{O}_t$ becomes fixed. Hence, in this case either $\mathcal{R}(t)$ becomes Q-regular or $\mathcal{O}_t$ becomes the multiplicity point. Algorithm $MoveToDestination_\delta()$ guarantees the uniqueness of the multiplicity point. Note that once the configuration becomes Q-regular, it remains the same. Thus no live-lock occurs during the whole execution of $CreateMultiplicity_\delta()$. $\square$

Lemma 6. *Algorithm $GatheringFault_\delta()$ solves the gathering problem in $(n, n-6)$ crash fault model for $n \geq 7$ semi-synchronous robots (initially placed at different positions) in finite time with weak multiplicity detection and the knowledge of δ.*

PROOF. Follows from lemma 4 and 5. $\square$

Theorem 2. *The gathering problem is solvable in $(n, n-6)$ crash fault model for $n \geq 3$ semi-synchronous robots (initially placed at different positions) in finite time with weak multiplicity detection and the knowledge of δ.*

PROOF. If $n \leq 6$, the value of the number of admissible faults is 0 and by [2] the gathering problem is solvable. For $n \geq 7$, the lemma 6 implies that gathering is possible with at most $(n-6)$ faulty robots. Hence the theorem. $\square$

5. CONCLUSION

This paper presents two distributed gathering algorithms for semi-synchronous robots in crash fault model under following assumptions:

- When robots are endowed only with weak multiplicity detection and the system permits at most $\lfloor \frac{n}{2} \rfloor - 1$ crash faults for $n \geq 7$.
- When robots are endowed with weak multiplicity detection, have the knowledge of δ and the system permits at most $(n-6)$ crash faults for $n \geq 7$.

It would be interesting to find out that whether the number of tolerable faults can be increased or the algorithms are optimal.

6. REFERENCES

[1] C. Agathangelou, C. Georgiou, and M. Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 250–259, 2013.

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

[3] S. Bhagat, S. G. Chaudhuri, and K. Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *J. Discrete Algorithms*, 36:pages 50–62, 2016.

[4] K. Bolla, T. Kovacs, and G. Fazekas. Gathering of fat robots with limited visibility and without global navigation. In *Proc. Swarm and Evolutionary Computation*, pages 30–38. 2012.

[5] Z. Bouzid, S. Das, and S. Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *Proc. IEEE 33rd International Conference on Distributed Computing Systems*, pages 337–346, 2013.

[6] Q. Bramas and S. Tixeuil. Wait-free gathering without chirality. In *International Colloquium on Structural Information and Communication Complexity.*, pages 313–327, 2014.

[7] S. G. Chaudhuri and K. Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *J. Discrete Algorithms*, 33:171–192, 2015.

[8] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. volume 41, pages 829–879. 2012.

[9] J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(6):pages 481–499, 2009.

[10] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *Proc. 20th International Symposium on Distributed Computing*, pages 46–60. 2006.

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

[12] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1-3):pages 147–168, 2005.

[13] T. Izumi, Z. Bouzid, S. Tixeuile, and K. Wada. Brief announcement: The bg-simulation for byzantine mobile robots. In *Distributed Computing*, volume 6950 of *Lecture Notes in Computer Science*, pages 330–331. Springer Berlin Heidelberg, 2011.

[14] G. Prencipe. Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. In *Proc. 7th Italian Conference on Theoretical Computer Science*, pages 154–171. 2001.

[15] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2 - 3):pages 222–231, 2007.

[16] I. Suzuki and M. Yamashita. Formation and agreement problems for anonymous mobile robots. In *Proc. 31st Annual Conference on Communication, Control and Computing*, pages 93 – 102, 1993.