SIAM J. COMPUT.
Vol. 36, No. 1, pp. 56–82

FAULT-TOLERANT GATHERING ALGORITHMS FOR AUTONOMOUS MOBILE ROBOTS*

NOA AGMON† AND DAVID PELEG‡

Abstract. This paper studies fault-tolerant algorithms for the problem of gathering N autonomous mobile robots. A gathering algorithm, executed independently by each robot, must ensure that all robots are gathered at one point within finite time. In a failure-prone system, a gathering algorithm is required to successfully gather the nonfaulty robots, independently of the behavior of the faulty ones. Both crash and Byzantine faults are considered. It is first observed that most existing algorithms fail to operate correctly in a setting allowing crash failures. Subsequently, an algorithm tolerant against one crash-faulty robot in a system of three or more robots is presented.

It is then observed that all known algorithms fail to operate correctly in a system prone to Byzantine faults, even in the presence of a single fault. Moreover, it is shown that in an asynchronous environment it is impossible to perform a successful gathering in a 3-robot system, even if at most one of them might fail in a Byzantine manner. Thus, the problem is studied in a fully synchronous system. An algorithm is provided in this model for gathering $N \geq 3$ robots with at most a single faulty robot, and a more general gathering algorithm is given in an N-robot system with up to f faults, where $N \geq 3f + 1$.

Key words. robot swarms, autonomous mobile robots, convergence

AMS subject classifications. 68Q22, 70B15

DOI. 10.1137/050645221

1. Introduction.

Background. Systems of multiple autonomous mobile robots engaged in cooperative activities have been extensively studied throughout the past decade [10, 5, 16, 17, 7, 11, 3, 27]. This subject is of interest for a number of reasons. For one, it may be possible to use a multiple robot system in order to accomplish tasks that no single spatially limited robot can achieve. Another advantage of multiple robot systems has to do with the decreased cost due to the use of simpler and cheaper individual robots. Also, these systems have immediate applicability in a wide variety of tasks, such as military operations and space missions. Subsequently, studies of autonomous mobile robot systems can be found in different disciplines, from engineering to artificial intelligence (e.g., [18, 4, 15, 19]).

Our interest is in problems related to the *distributed control* of systems of autonomous mobile robots. Most studies on robot control problems resulted in the design of algorithms based on heuristics, with little emphasis on formal analysis of the correctness, termination, or complexity properties of the algorithms. During the last few years, various aspects of this problem have been studied from the point of view of distributed computing (cf. [5, 20, 25, 26, 23, 2]), where the focus is on trying to model an environment consisting of mobile robots, and studying the capabilities the robots must have in order to achieve their common goal. A number of computational

*Received by the editors August 4, 2005; accepted for publication (in revised form) December 1, 2005; published electronically May 3, 2006. An extended abstract of this paper appeared in *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, 2004.
http://www.siam.org/journals/sicomp/36-1/64522.html

†Department of Computer Science, Bar-Ilan University, Ramat-Gan, 52900 Israel (segaln@cs.biu.ac.il).

‡Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel (david.peleg@weizmann.ac.il). The work of this author was supported in part by a grant from the Israel Science Foundation.

models were proposed in the literature, and some studies attempted to characterize the influence of the models on the ability of a group of robots to perform certain basic tasks under different constraints.

The primary motivation of the studies presented in [23, 26, 20, 21, 25] is to identify the minimal capabilities a collection of distributed robots must have in order to accomplish certain basic tasks and produce interesting interaction. Consequently, the models adopted in these studies assume the robots to be relatively weak and simple. In particular, these robots are generally assumed to be dimensionless (namely, treated as points that do not obstruct each other's visibility or movement), oblivious (or memoryless; namely, they do not remember their previous actions or the previous positions of the other robots), lacking a common coordinate system, orientation, or scale, using no explicit communication, and anonymous (some of these assumptions are modified in order to achieve goals that are otherwise unfeasible). They operate in simple "look-compute-move" cycles. Thus the robots base their movement decisions on viewing their surroundings and analyzing the configuration of robot locations. A robot is capable of locating all robots within its visibility range (which can be either limited or unlimited) and laying them in its private coordinate system, thereby calculating their position (distance and angles) with respect to one another and with respect to itself. Hence, from the "distributed computing" angle, such problems are particularly interesting since they give rise to a different type of communication model, based solely on "positional" or "geometric" information exchange.

A basic task that has received considerable attention is the *gathering* problem, defined as follows. Given an initial configuration of N autonomous mobile robots, all N robots should occupy a single point within a finite number of steps. The closely related *convergence* problem is defined similarly, except that the robots are required only to converge to a single point, rather than reach it. Namely, instead of demanding that the robots gather to one point within finite time, the convergence requirement is that for every $\epsilon > 0$, there is a time t_ϵ by which all robots are within distance of at most ϵ of each other.

Fault tolerance. As the common models of multiple robot systems assume cheap, simple, and relatively weak robots, the issue of resilience to failure becomes prominent, since in such systems one cannot possibly rely on assuming fail-proof hardware or software, especially when such robot systems are expected to operate in hazardous or harsh environments. At the same time, one of the main attractive features of multiple-robot systems is their potential for enhanced fault tolerance. It seems plausible that the inherent redundancy of such systems may be exploited in order to enable them to perform their tasks even in the presence of faults.

Following the common "f of N" classification often used in the area, a fault-tolerant algorithm for a given task is required to ensure that in a system consisting of N robots where it is assumed that at most f robots might fail in any execution, the task is achieved by all nonfaulty robots, regardless of the actions taken by the faulty ones. In the gathering task, for example, when faults are introduced into the system, the requirement applies only for the nonfaulty robots; i.e., if f' robots fail, then all the remaining $N - f'$ nonfaulty robots are required to occupy a single point within a finite time.

Perhaps surprisingly, however, this aspect of multiple robot systems has been explored to very little extent so far. In fact, almost all results we are aware of in the literature rely on the assumption that all robots function properly and follow their protocol without any deviation. One exception concerns *transient* failures. As

observed in [26, 23, 13], any algorithm that works correctly on oblivious robots is necessarily *self-stabilizing*; i.e., it guarantees that after any transient failure the system will return to a correct state and the goal will be achieved. Yet another line of study concerns a fault model where it is assumed that restricted sensor and control failures might occur, but if faults do occur in the system, then the identity of the faulty robots becomes known to all robots [23]. This may be an unrealistic assumption in many typical settings, and it clearly provides an easy means of overcoming the faults: Each nonfaulty robot may simply ignore the failed ones, effectively removing them from the group of robots, so the algorithm continues to function properly. However, in case unidentified faults occur in the system, it is no longer guaranteed that the algorithms of [23, 26] remain correct; i.e., the goal might not be achieved. The only concrete attempt we are aware of for dealing with crash faults is described in [29], where an algorithm is given for the *active robot selection problem* (ARSP) in the presence of initial crash faults. The ARSP creates a subgroup of nonfaulty robots from a group that also includes initially crashed robots and makes the robots in that subgroup recognize one another. This allows the nonfaulty robots in the subgroup to overcome the existence of faults in the system, and they can further execute any algorithm within the group.

Hence the design of fault-tolerant distributed control algorithms for multiple robot systems is still a largely unexplored direction, which the current paper investigates.

Related work. A number of basic mobile robot coordination problems were considered in the literature. One class of problems involves the formation of geometric patterns. The robots are required to arrange themselves in a given geometric form, such as a circle, a simple polygon or a line, within finite time (see, e.g., [23, 9, 12]). The task of *flocking*, requiring the robots to follow the movement of a predefined leader, was studied in [22]. The even distribution problem, requiring the robots to spread out uniformly over a specified region of a simple geometric shape, and the related task of partitioning the robots into groups were studied in [23].

The problem of gathering autonomous mobile robots, dealt with in this paper, requires the robots to gather to the same point within finite time (see, e.g., [24, 25, 14, 7, 6, 8]). This problem was studied extensively in two computational models. The first is the model of [23, 26], hereafter referred to as the *semisynchronous* ($\mathcal{SSYNC}$) model. The second is the closely related CORDA model [20, 21, 25], hereafter referred to as the *asynchronous* ($\mathcal{ASYNC}$) model.

The gathering problem was first discussed in [25, 26] in the $\mathcal{SSYNC}$ model. It was proved there that it is impossible to achieve gathering of *two* oblivious autonomous mobile robots that have no common sense of orientation under the $\mathcal{SSYNC}$ model. The algorithms presented therein for $N \geq 3$ robots rely on the assumption that a robot can identify a point p^* occupied by two or more robots (also known as *multiplicity point*). This assumption was later proved to be essential for achieving gathering in all asynchronous and semisynchronous models [22]. Another necessary requirement for solvability in the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ models is that the input configuration does not include more than one multiplicity point of nonfaulty robots (it is easy to show that if two multiplicity points of nonfaulty robots are allowed, the situation is equivalent to the 2-robot system, and thus gathering is impossible). In fact, all known gathering algorithms for $N \geq 3$ rely on a strategy by which a single multiplicity point p^* is formed during the execution of the algorithm, and once this happens, all robots move to the point p^*. Under these assumptions, an algorithm is developed in [26] for gathering $N \geq 3$ robots in the $\mathcal{SSYNC}$ model. In the $\mathcal{ASYNC}$ model, an algorithm

for gathering $N = 3, 4$ robots is brought in [22, 7], and an algorithm for gathering $N \geq 5$ robots is described in [6].

The gathering problem was also studied (in both the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ models) in a system where the robots have limited visibility. The visibility conditions are modelled by means of a *visibility graph*, representing the (symmetric) visibility relation of the robots with respect to one another; i.e., an edge exists between R_i and R_j if and only if R_i and R_j are visible to each other. It was shown that the problem is unsolvable in the case that the visibility graph is not connected [14]. In [1] a convergence algorithm was provided for any N in limited visibility systems. An algorithm that achieves gathering in the $\mathcal{ASYNC}$ model is described in [14], under the assumption that all robots share a compass (i.e., agree on a direction in the plane).

Our results. This paper presents a systematic study of failure-prone robot systems through examining the gathering problem under the crash and Byzantine fault models. An (N, f)-*fault system* is a system consisting of N robots, of which at most f might fail at any execution. An (N, f)-*crash system* (resp., (N, f)-*Byzantine system*) is an (N, f)-fault system where the faults considered are according to the crash or Byzantine model. A fault-tolerant algorithm for a given task in an (N, f)-fault system is required to ensure that so long as at most f robots have failed, the task is achieved by all nonfaulty robots, regardless of the actions taken by the faulty ones.

Under the crash fault model, we show that the gathering problem is solvable in current computational models such as the $\mathcal{SSYNC}$ model, though most existing algorithms fail to deal correctly with such faults and, in particular, there is currently no algorithm for $N \geq 4$ that solves the gathering problem in the presence of one faulty robot under the crash fault model. We propose an algorithm that solves the gathering problem in an $(N, 1)$-crash system, for any $N \geq 3$, under the $\mathcal{SSYNC}$ model.

We then consider (N, f)-Byzantine systems for $N \geq 3$. We first observe that all existing algorithms fail to deal correctly with this situation. Moreover, we show that it is impossible to perform a successful gathering in $(3, 1)$-Byzantine systems under the $\mathcal{SSYNC}$ model. We then introduce the *fully synchronous* ($\mathcal{FSYNC}$) model, which is similar to the synchronous model mentioned in [26], and present an algorithm solving the gathering problem under this model in (N, f)-Byzantine systems for every[1] $N \geq 3f + 1$.

2. The model. We follow the common computational model of distributed robot systems. In particular, we make the following assumptions: The visibility range of the robots is assumed to be unlimited. The robots are treated as points (dimensionless objects) which do not obstruct each other's visibility or movement. The robots are anonymous and cannot communicate with each other. For the sake of analysis, denote the robots in the system by $R_1, \ldots, R_N$. Each robot R_i has its private coordinate system, consisting of the position of the origin, direction of the positive x-axis, and the size of one unit distance. It is assumed that the direction of the positive y-axis is 90° counterclockwise of the direction of the positive x-axis. The coordinate systems of the various robots might all be different and not share the same direction or scale.

Following most previous papers on the gathering problem in the literature [24, 25, 14, 7, 22], the model adopted throughout this paper is the *oblivious* model, where it is assumed that the robots cannot remember their previous states, and thus the

[1] A peculiarity of our algorithms is that $N = 3$ robots can tolerate $f = 1$ failures, but $N > 3$ robot systems require $N \geq 3f + 1$ rather than $N \geq 3f$.

decisions they make in each step are based only on the current configuration. The main motivation for developing algorithms for the oblivious model is twofold. First, solutions developed on the basis of assuming nonobliviousness do not necessarily work in a dynamic environment where the robots are activated in different cycles or might be added/removed from the system dynamically. Second, as mentioned earlier, any algorithm that works correctly for oblivious robots is inherently self-stabilizing, i.e., it withstands transient errors. More generally, it is advantageous to develop algorithms for the weakest robot types possible, as an algorithm that works correctly for weak robots will clearly work correctly in a system of stronger robot types. In contrast, our lower bounds serve mainly to draw the borderlines where the various models become too weak to allow solutions.

Robot operation cycle. Each robot R_i in the system is assumed to operate individually in simple cycles. Every cycle consists of three steps, Look, Compute, and Move. In the $\mathcal{FSYNC}$ and $\mathcal{SSYNC}$ models the length of this cycle is uniform for all robots.

- *Look*: Identify the locations of all robots in R_i's private coordinate system; the result of this step is a multiset of points $P = \{p_1, \ldots, p_N\}$ defining the current *configuration*. As the robots are indistinguishable, each robot R_i knows its own location p_i but does not know the identity of the robots at each of the other points.
- *Compute*: Execute the given algorithm, resulting in a goal point p_G.
- *Move*: Move towards the point p_G. The robot might stop before reaching its goal point p_G but is promised to traverse a distance of at least S (unless it has reached the goal).

Note that the Look and Move steps are carried out identically in every cycle, independently of the algorithm used. The differences between different algorithms occur in the Compute step. Moreover, the procedure carried out in the Compute step is identical for all robots. If the robots are oblivious, then the algorithm cannot rely on information from previous cycles; thus the procedure can be fully specified by describing a single Compute step, and its only input is the current configuration $P = \{p_1, \ldots, p_N\}$, giving the robot locations. Throughout, we may denote the location of R_i in the configuration P by $p(R_i)$. Also, whenever no confusion may arise, we identify $p(R_i)$ as the point p_i.

Three synchronization models. As mentioned earlier, our computational model for studying and analyzing problems of coordinating and controlling a set of autonomous mobile robots follows two well-studied models: the $\mathcal{SSYNC}$ model and the $\mathcal{ASYNC}$ model. The semisynchronous ($\mathcal{SSYNC}$) model is partially synchronous, in the sense that all robots operate according to the same clock cycles, but not all robots are necessarily active in all cycles. The activation of the different robots can be thought of as managed by a hypothetical "scheduler," whose only "fairness" obligation is that each robot must be activated and given a chance to operate infinitely often in any infinite execution. The fully asynchronous ($\mathcal{ASYNC}$) model differs from the $\mathcal{SSYNC}$ model in that each robot acts independently in a cycle composed of *four* steps: Wait, Look, Compute, Move. The length of this cycle is finite, but not bounded. Consequently, there is no bound on the length of the walk in a single cycle, and different cycles of the same robot may vary in length. In contrast, in the $\mathcal{SSYNC}$ model a bound exists on the cycle length due to the common clock, and as a result the robot will not necessarily reach the target point p in the current cycle but stop somewhere on its trajectory to p.

In this paper we also consider the extreme *fully synchronous* ($\mathcal{FSYNC}$) model.

This model is similar to the $\mathcal{SSYNC}$ model, where the robots operate according to the same clock cycles, except that here all robots are active on all cycles. In this model we assume discrete time $0, 1, \ldots$, and let $p_i(t)$ denote the position of R_i at time t, where $p_i(0)$ is the initial position of R_i. In each cycle t, the set of positions is a multiset, as two robots are not prohibited from occupying the same position simultaneously. In each cycle t, each robot R_i is capable of moving over distance at least $S > 0$ in one step (S is unknown to the robots). Therefore it is guaranteed that if $\text{dist}(p_G^i, p_i(t)) \leq S$, where p_G^i is R_i's goal point in the current cycle, then R_i will reach its goal in the current cycle. Otherwise, it will traverse a distance of at least S towards p_G^i.

Failure models. The fault models discussed throughout the paper are the *crash* fault model and the *Byzantine* fault model. In the *Byzantine* fault model, it is assumed that a faulty robot might behave in arbitrary and unforeseeable ways. For the sake of analysis, it is convenient to model the behavior of the system by means of an *adversary* which has the ability to control the behavior of the faulty robots, as well as the "undetermined" features in the behavior of the nonfaulty processors (e.g., the distance to which they move). Specifically, in each cycle the adversary has the following roles. For each faulty robot, it determines its course of action in that cycle, which can be arbitrary. For each nonfaulty robot, it determines the distance to which the robot will move in this cycle (for a robot R_i located at p_i and headed for the goal point p_G, if $\text{dist}(p_i, p_G) \leq S$, then the robot must be allowed to reach p_G; else, the adversary may stop R_i at any point on the line segment $\overline{p_i p_G}$ that is at least distance S away from p_i).

In the *crash* fault model, the behavior of the system is similar to the one described in the Byzantine fault model, except that for each faulty robot the adversary is only allowed to stop its movement. This may be done at any point in time during the cycle, i.e., either during the movement toward the goal point or before it has started. Once the adversary has crashed the faulty robot, that robot will remain stationary indefinitely.

3. Gathering under the crash fault model.

3.1. Inadequacy of known algorithms. Most gathering algorithms proposed in the literature fail to withstand even a single crash failure because they depend, in certain configurations, on the movement of a single robot. More formally, let $\mathcal{A}$ be a gathering algorithm for an N-robot system. In every configuration C, the algorithm instructs some robots to move and some to remain stationary. Denote the number of robots $\mathcal{A}$ instructs to move in configuration C by $M(C, \mathcal{A})$, and let

$$\check{M}(N, \mathcal{A}) \;=\; \min\{M(C, \mathcal{A}) \mid C \text{ is a configuration in an } N\text{-robot system}\}.$$

LEMMA 3.1. *In an (N, f)-crash system, an algorithm $\mathcal{A}$ with $\check{M}(N, \mathcal{A}) \leq f$ will fail in achieving gathering or convergence.*

Proof. Consider an (N, f)-crash system and a gathering algorithm $\mathcal{A}$ with $\check{M}(N, \mathcal{A}) \leq f$, and let C' be the configuration of the system realizing $\check{M}$, i.e., such that $M(C', \mathcal{A}) \leq f$. Starting in that configuration, the adversary can fail the (f or fewer) robots instructed by the algorithm to move. This will cause the next configuration to be identical to C' again, and the $N - f$ nonfaulty robots will remain in the same configuration C' indefinitely. □

In fact, every gathering algorithm $\mathcal{A}$ we are aware of in the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ models [24, 25, 26, 7] has $\check{M}(N, \mathcal{A}) = 1$ for $N \geq 4$, and, consequently, by Lemma 3.1,

these algorithms fail to achieve gathering even in the presence of one crash faulty robot. The algorithm described in [6] for gathering $N \geq 5$ robots in the $\mathcal{ASYNC}$ model can also fail in the presence of one crash faulty robot if that robot lies between some other robot and its goal point. On the positive side, it turns out that the gathering algorithm given in [7] for $N = 3$ under the $\mathcal{ASYNC}$ model can be shown to operate correctly also in the presence of one crashed robot (we give a slightly simpler algorithm for this case in the $\mathcal{SSYNC}$ model below), and the algorithm given therein for $N = 4$ can be transformed into an algorithm for $(4, 1)$-crash systems with some minor changes.

An additional difficulty in handling a robot system with crash faults is caused by the assumption, made by all current algorithms, that only a single multiplicity point is created throughout the execution of the algorithm. In the presence of faults, the fact that the adversary has the ability to stop the nonfaulty robots after traversing a minimal distance S and the ability to crash the faulty robots at any step during the execution makes it easy for the adversary to create a second multiplicity point once the first is created, whenever the trajectories of two or more robots moving towards their goals intersect. In particular, it is easy to see that in a collinear configuration with $N > 3$ robots and given an algorithm that instructs all robots to move towards a point on the line, the adversary can create two multiplicity points on the line, simply by crashing some robot R at a point p between the multiplicity point and another robot R', thus forcing R' to pass through p, and stopping it there.

3.2. An algorithm for a $(3, 1)$-crash system. Consider the following Procedure $\mathsf{3\text{-}Gather}_{\mathrm{crash}}$ for gathering in a $(3, 1)$-crash system in the $\mathcal{SSYNC}$ model. As discussed earlier, we need only present the procedure used for the Compute step. The input to this procedure is the configuration $P = \{p_1, p_2, p_3\}$. The procedure classifies the configuration according to its state, and acts in each case as follows.

Procedure $\mathsf{3\text{-}Gather}_{\mathrm{crash}}(P)$

1. **State [MULT]:** P contains a multiplicity point p^*:
   Set $p_G \leftarrow p^*$.
2. **State [Collinear]:** p_1, p_2, p_3 are collinear (say, with p_2 in the middle):
   Set $p_G \leftarrow p_2$.
3. **State [Obtuse]:** $\exists i \in \{1, 2, 3\}$ such that $\angle p_j p_i p_k \geq \pi/2$:
   Set $p_G \leftarrow p_i$.
4. **State [Acute]:** p_1, p_2, p_3 form an acute triangle:
   Set p_G to be the intersection point of the three angle bisectors.

Note that state [Collinear] is redundant, since it is covered by state [Obtuse]. It is included merely for convenience of presentation.

Analysis. In analyzing our algorithms, we use the following notation regarding points and lines in the Euclidean plane. Denote the Euclidean distance between two points p and q by $\mathrm{dist}(p, q)$. Also, denote the Euclidean distance between two current locations p_i and p_j of the two robots R_i and R_j, respectively, by $\mathrm{dist}(R_i, R_j)$. Denote the line segment between the points p and q by $\overline{pq}$. We use the following well-known fact.

LEMMA 3.2. *In a triangle $\triangle p_1 p_2 p_3$, the intersection point p_M of the three angle bisectors satisfies $\angle p_i p_M p_j \geq \pi/2$ for every $1 \leq i < j \leq 3$.*

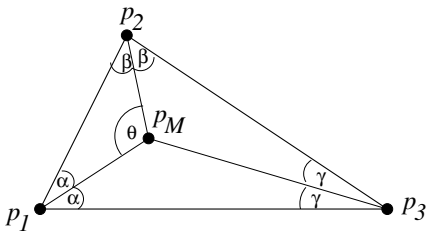


FIG. 1. *Proof of Lemma* 3.2.

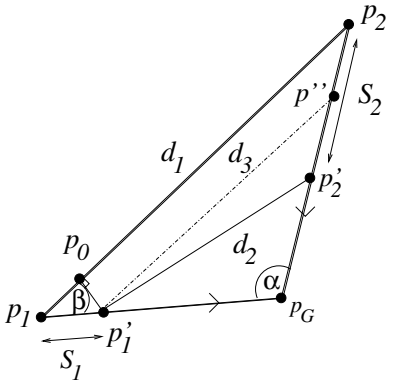


FIG. 2. *Proof of Lemma* 3.3.

Proof. Let $\alpha = \angle p_M p_1 p_2$, $\beta = \angle p_M p_2 p_3$, $\gamma = \angle p_M p_3 p_1$, and $\theta = \angle p_1 p_M p_2$ (see Figure 1). It follows that $2\alpha + 2\beta + 2\gamma = \pi$; hence $\alpha + \beta = \pi/2 - \gamma < \pi/2$, and since in the triangle $\triangle p_M p_1 p_2$, $\theta + \alpha + \beta = \pi$, it follows that $\theta > \pi/2$. A similar argument applies in $\triangle p_M p_2 p_3$ and $\triangle p_M p_1 p_3$. ∎

LEMMA 3.3. *If two robots R_1 and R_2, initially located at the points p_1 and p_2, respectively, move towards a common meeting point p_G and $\alpha = \angle p_1 p_G p_2 \geq \pi/2$, then the distance between them decreases by at least* $0.7S$.

Proof. For $i = 1, 2$, let S_i denote the distance traversed by the robot R_i, and let p_i' denote the new location of R_i. Consider the triangle $\triangle p_1 p_2 p_G$, and let $\beta = \angle p_2 p_1 p_G$ (see Figure 2). Denote the distance between the robot locations before and after the movement by $d_1 = \text{dist}(p_1, p_2)$ and $d_2 = \text{dist}(p_1', p_2')$, respectively. Without loss of generality, suppose that p_1' is closer than p_2' to the line $\overline{p_1 p_2}$. Draw a line parallel to d_1 through p_1', denote its intersection with $\overline{p_2 p_G}$ by p'', and let $d_3 = \text{dist}(p_1', p'')$.

We need to prove that $d_1 - d_2 > 0.7S$. Since it is clear that $d_2 \leq d_3$, it suffices to show that $\Delta = d_1 - d_3 > 0.7S$. Drop a perpendicular line from p_1' to $\overline{p_1 p_2}$, and let p_0 be its intersection point with the line $\overline{p_1 p_2}$. Let $\Delta' = \text{dist}(p_0, p_1)$. It is also clear that $\Delta' \leq \Delta$; hence it remains to prove that $\Delta' \geq 0.7S$. Since $\alpha \geq \pi/2$, the remaining two angles in $\triangle p_1 p_2 p_G$ sum to at most $\pi/2$. Without loss of generality, let $\beta \leq \pi/4$. Also, according to our model assumption, each robot moves a distance of at least S in each cycle, i.e., $S_i \geq S$ for $i = 1, 2$. Therefore, by the sine theorem on the triangle $\triangle p_1 p_1' p_0$ (see Figure 3), $S \leq S_1 = \frac{S_1}{\sin(\pi/2)} = \frac{\Delta'}{\sin(\pi/2 - \beta)}$, and hence $\Delta' \geq S \cdot \cos(\beta) \geq S \cdot \cos(\pi/4) \geq 0.7S$, completing the proof. ∎

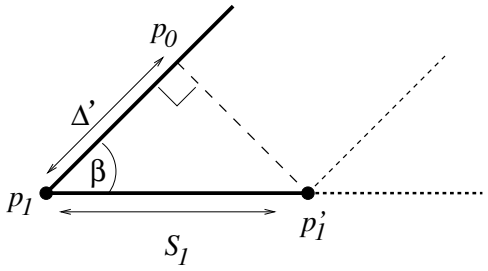


FIG. 3. *Triangle enlargement from proof of Lemma* 3.3.

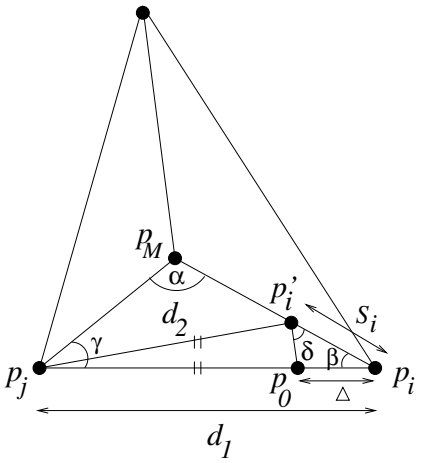


FIG. 4. *Proof of Lemma* 3.4.

LEMMA 3.4. *There exists a constant $c > 0$ such that, given three robots R_1, R_2, and R_3 located at points p_1, p_2, and p_3, respectively, where $\triangle p_1p_2p_3$ is an acute triangle, if one or more of the robots traverses a distance of at least S towards the intersection point p_M of the three angle bisectors, then the circumference of the triangle decreases by at least cS.*

Proof. By Lemma 3.2, $p_ip_Mp_j > \pi/2$ for every $1 \leq i, j \leq 3$. Thus, by Lemma 3.3, if two of the three robots move towards p_M, then the distance between them decreases by at least $0.7S$, and as the other distances do not increase, the circumference of the triangle decreases by at least $0.7S$. It remains to show that even if only one robot moves towards p_M it decreases the distance between itself and its neighbors by at least $c'S$ for some constant c'; thus altogether the circumference of the triangle decreases by at least $2c'S = cS$.

Consider the triangle $\triangle p_ip_Mp_j$, $1 \leq i, j \leq 3$. Let $\alpha = \angle p_ip_Mp_j$, $\beta = \angle p_Mp_ip_j$, and $\gamma = \angle p_Mp_jp_i$ (see Figure 4). Since $\triangle p_1p_2p_3$ is acute, it follows that $2\beta < \pi/2$; thus $\beta < \pi/4$ and, similarly, $\gamma < \pi/4$. Assume, without loss of generality, that robot R_i traversed a distance $S_i \geq S$ towards p_M and lies on a point p'_i. Let $d_1 = \text{dist}(p_i, p_j)$, $d_2 = \text{dist}(p'_i, p_j)$, and $\Delta = d_1 - d_2$. By the sine theorem on the triangles $\triangle p_ip_jp_M$ and $\triangle p'_ip_jp_M$ it follows that $\frac{d_1}{d_2} = \frac{\sin(\angle p_jp'_ip_M)}{\sin\beta}$, and since $\angle p_jp'_ip_M > \beta$, it follows that $d_1 > d_2$. Let p_0 be the point on the segment $\overline{p_ip_j}$ that creates an isosceles triangle

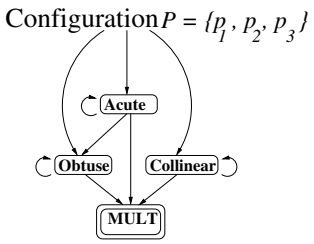


FIG. 5. *Statechart for Procedure* 3-Gather$_{\text{crash}}$.

$\triangle p_0 p_j p_i'$; i.e., $\text{dist}(p_j, p_0) = d_2$. Note that $\text{dist}(p_0, p_i) = \Delta$. Let $\delta = \angle p_i p_i' p_0$. Since

$$3\pi/8 < \frac{\pi - \gamma}{2} < \frac{\pi - \angle p_i p_j p_i'}{2} = \angle p_j p_0 p_i' = \angle p_j p_i' p_0 < \pi/2,$$

it follows that $3\pi/8 < \beta + \delta < \pi/2$, and as $\beta < \pi/4$, we have $\pi/8 < \delta < \pi/2$. By the sine theorem on the triangle $\triangle p_i p_i' p_0$ it follows that

$$\Delta = S_i \frac{\sin\delta}{\sin(\delta+\beta)} \geq S \frac{\sin\delta}{\sin(\delta+\beta)} \geq S \frac{\sin(\pi/8)}{\sin(\pi/2)} > 0.3S.$$

Therefore by choosing $c' = 0.3$, the lemma holds for $c = 0.6$. ∎

THEOREM 3.5. *Algorithm* 3-Gather$_{\text{crash}}$ *solves the gathering problem in a* $(3,1)$*-crash system under the* $\mathcal{SSYNC}$ *model.*

Proof. Consider an initial configuration $P = \{p_1, p_2, p_3\}$. It suffices to show that the algorithm causes the system to reach state [MULT]; i.e., either it gathers all robots together in one point or it causes the creation of one multiplicity point, since if the remaining robot is nonfaulty, then it will join the multiplicity point in finite time by step 1 of the algorithm, and if it is faulty, then gathering has been achieved. Consider the flow of states the system could be in. It suffices to show that the states used for classifying the configurations in Procedure 3-Gather$_{\text{crash}}$ form a finite connected directed acyclic graph (DAG) (possibly with self-loops), where all paths lead to a final state [MULT] in which a multiplicity point exists (see Figure 5), such that starting with a configuration in any of the states, we reach the final state within a finite number of cycles.

If in the initial configuration p_1, p_2, p_3 are collinear, then the configuration will remain collinear, and within finite time, either both extreme robots will arrive at the location of the middle robot (if both are nonfaulty) or only one of them will arrive (if one of the extreme robots is faulty); thus in any case a multiplicity point will be created in the location of the middle robot, leading to state [MULT].

Next, suppose that the initial configuration is not collinear but obtuse; i.e., there exists a point p_i such that $\angle p_j p_i p_k \geq \pi/2$. Then the configuration remains obtuse until one robot reaches R_i. Since at least one of the robots instructed to move towards R_i is nonfaulty, it will reach its goal point within finite time, thus reaching state [MULT].

Finally, if p_1, p_2, p_3 create an acute triangle in cycle t, then one of the following two cases holds. In the first case, at least one robot was active in the current cycle and traversed a distance of at least S towards p_M. There are two subcases to be examined. If the system remains in state [Acute], then by Lemma 3.4 the circumference of $\triangle p_1 p_2 p_3$ decreases by at least $0.6S$. Therefore, if the robots constantly remain in state [Acute], then at least two robots will eventually meet in p_M, leading to state [MULT]. The other subcase is that this movement causes $\triangle p_1 p_2 p_3$ to become

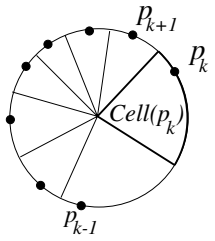


FIG. 6. *Example of a circle division according to the Voronoi cells.*

obtuse. In this case, the system changes to state [Obtuse]. In the second case, all active robots in this cycle traversed a distance smaller than S towards p_M; thus they are now located at p_M. Again there are two subcases. If two or more robots were active, then they meet at p_M, leading to state [MULT]. Otherwise, only one robot, say R_i, is located in p_M. Then by Lemma 3.2, $\angle p_j p_i p_k \geq \pi/2$; thus $\triangle p_1 p_2 p_3$ becomes obtuse, and the system changes to state [Obtuse], leading to gathering as discussed above. □

3.3. An algorithm for an $(N, 1)$-crash system, $N \geq 3$. Let us start with some terminology. A *legal* configuration in the $(N, 1)$-crash system is a set P of robot locations that has at most one multiplicity point. Denote the *smallest enclosing circle* of the set P by $\mathcal{SEC}(P)$ and the points on its circumference by $\mathcal{C}_{cir}(P)$ and let $\mathcal{C}_{\rm int}(P) = P \setminus \mathcal{C}_{\rm cir}(P)$. For a circle C and the set of points $P = \{p_1, \ldots, p_l\}$ on its circumference, denote the partition of the circle C into Voronoi cells according to the points in P by $\mathrm{Vor}(C, P)$, and denote by $\mathrm{Cell}(p_k)$ the cell defined by the point $p_k \in P$ (see Figure 6). Two points q and q' in C are said to share the cell $\mathrm{Cell}(p_k)$ if they both lie inside the cell or on its boundary.

Consider the following Algorithm $\mathsf{Gather}_{\rm crash}$ for gathering all nonfaulty robots in an $(N, 1)$-crash system under the $\mathcal{SSYNC}$ model. The input to this algorithm is a legal configuration $P = \{p_1, \ldots, p_N\}$. The algorithm classifies the configuration according to its state and acts in each case as follows. If there are no multiplicity points in the configuration, then each robot performs Procedure $\mathsf{Create\_Mult}$ in order to reach a configuration with a multiplicity point. Figure 7 illustrates the three possible cases of substate [IN2] in this procedure. Once a multiplicity point p^* is detected, each robot performs Procedure $\mathsf{GoTo\_Mult}$ in order to achieve gathering of all nonfaulty robots in p^*, while avoiding creation of additional multiplicity points.

Algorithm $\mathsf{Gather}_{\rm crash}(P)$

1. **State [Singletons]:** The configuration P does not contain a multiplicity point: Invoke Procedure $\mathsf{Create\_Mult}(P)$.
2. **State [MULT]:** The configuration P contains a single multiplicity point p^*: Invoke Procedure $\mathsf{GoTo\_Mult}(P)$.

The input to Procedure $\mathsf{GoTo\_Mult}$ is the configuration $P = \{p_1, \ldots, p_N\}$. We say that robot R_i has a *"free corridor"* to the point p if no other robot is currently located on the straight line segment $\overline{p_i p}$. Note that as robots are viewed as dimensionless objects, the availability of a free corridor is not necessarily a prerequisite for allowing a robot to get home free. However, allowing a robot to follow a trajectory through the location of another robot makes the algorithm prone to the creation of more

Procedure Create_Mult(P)
State [N3]: $N = 3$:
Invoke Procedure 3-Gather$_{\text{crash}}$ on p_1, p_2, p_3.
State [N4+]: $N \geq 4$:

1. **Substate [IN0]:** $|\mathcal{C}_{\text{int}}(P)| = 0$:
   Set p_G to be the center of $\mathcal{SEC}(P)$.
2. **Substate [IN1]:** $|\mathcal{C}_{\text{int}}(P)| = 1$ with p_j as the single point in $\mathcal{C}_{\text{int}}(P)$:
   Set $p_G \leftarrow p_j$.
3. **Substate [IN2]:** $|\mathcal{C}_{\text{int}}(P)| = 2$ with p_i and p_j as the two points in $\mathcal{C}_{\text{int}}(P)$:
   Each robot R_k in $\mathcal{C}_{\text{cir}}(P)$ sets $p_G(R_k) \leftarrow p(R_k)$.
   The two robots R_i and R_j in $\mathcal{C}_{\text{int}}(P)$ do:
   (a) Compute the Voronoi partition $\text{Vor}(\mathcal{SEC}(P), \mathcal{C}_{cir}(P))$.
   (b) **Substate [IN2(a)]:** p_i and p_j do not share cells:
   R_i and R_j move towards the center of $\mathcal{SEC}(P)$.
   (c) **Substate [IN2(b)]:** p_i and p_j share a single cell, Cell(R_k):
   R_i and R_j move towards R_k.
   (d) **Substate [IN2(c)]:** p_i and p_j share two cells; i.e., both robots lie on the radius forming the boundary between two adjacent cells Cell(R_k) and Cell(R_{k+1}):
   The robot closer to the circle, say R_i, chooses the first of R_k, R_{k+1} in its clockwise direction, say R_k, and sets $p_G(R_i) \leftarrow p(R_k)$.
   The other robot, R_j, sets $p_G(R_j) \leftarrow p(R_i)$.
4. **Substate [IN3]:** $|\mathcal{C}_{\text{int}}(P)| \geq 3$:
   Each robot R_k in $\mathcal{C}_{\text{cir}}(P)$ sets $p_G(R_k) \leftarrow p(R_k)$.
   Each robot R_k in $\mathcal{C}_{\text{int}}(P)$ recursively invokes Procedure Create_Mult($\mathcal{C}_{\text{int}}(P)$).

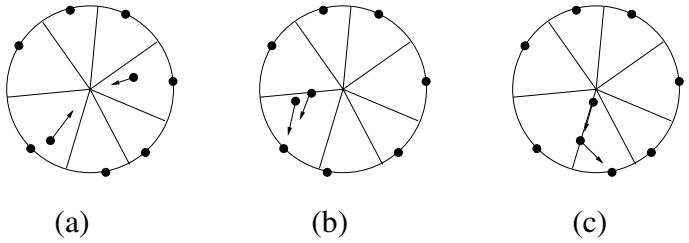


FIG. 7. *Illustration of the three substates of substate* [IN2] *in Procedure* Create_Mult.

than one multiplicity point. Therefore Procedure GoTo_Mult attempts to avoid such trajectories.

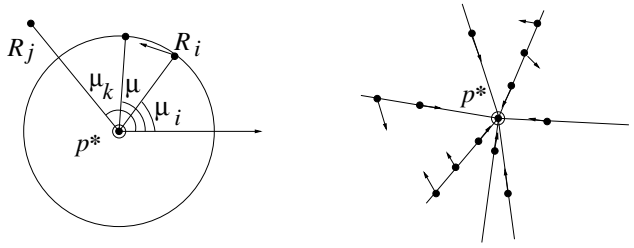
Analysis.

LEMMA 3.6. *If the initial configuration is in state* [Singletons]*, i.e., it contains no multiplicity points, then Procedure* Create_Mult *leads, within finite time, to a configuration in state* [MULT]*, i.e., including a single multiplicity point.*

Proof. We prove the lemma by looking at the flow of states the system could be in. It suffices to show that the states used for classifying the configurations in Procedure Create_Mult form a finite connected DAG (possibly with self-loops), where all paths lead to a final state [MULT] in which a multiplicity point exists (see Figure 9), such that starting with a configuration in any of the states, we reach the final state within a finite number of cycles.

Procedure GoTo_Mult(P) (for robot R_i)
/* The configuration contains a multiplicity point p^* */

1. **State [Free]:** R_i has a free corridor to p^*:
   Set $p_G \leftarrow p^*$.
2. **State [Blocked]:** There exist one or more robots on R_i's trajectory towards p^*:
   a. Translate your coordinate system to be centered at p^*.
   b. Compute for each robot R_j the angle μ_j of $\overrightarrow{p^*p_j}$ counterclockwise from the x-axis.
   c. Find the robot R_k with smallest angle $\mu_k > \mu_i$.
   Let $\mu = (\mu_k + 2\mu_i)/3$, and $d = \text{dist}(R_i, R_k)$ (see Figure 8(a)).
   d. Let p'_i be the point at distance d and angle μ from p^*.
   e. Set $p_G \leftarrow p'_i$.



(a) (b)

FIG. 8. *Illustration of state* [Blocked] *in Procedure* GoTo_Mult.

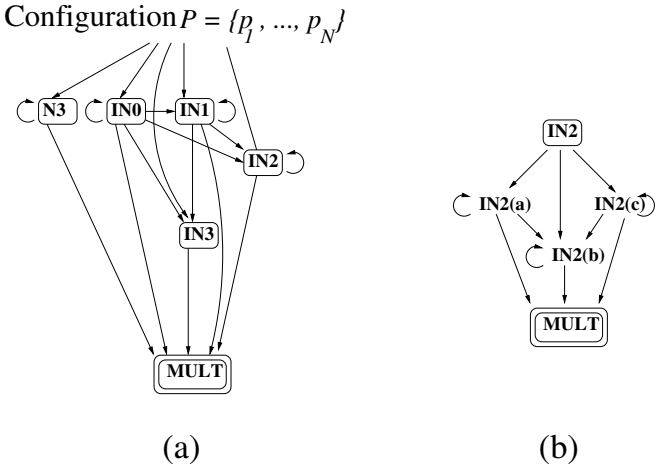


FIG. 9. *Statechart for Procedure* Create_Mult. (a) *The general statechart.* (b) *The substates of state* [IN2].

State [N3]: If $N = 3$, then by Theorem 3.5, Procedure 3-Gather$_{\text{crash}}$ achieves gathering, and in particular one multiplicity point is created.

State [IN0]: If $N \geq 4$ and $|\mathcal{C}_{\text{int}}(P)| = 0$, then if not all robots move together

to the same distance, the system changes either to state [MULT] or to a state where $|\mathcal{C}_{\text{int}}(P)| \geq 1$, i.e., [IN1], [IN2], or [IN3]. If robots on the circumference move to a new configuration P' in which they are all again on $\mathcal{SEC}(P')$, then the system remains in the same state, [IN0], but the radius of $\mathcal{SEC}(P)$ is reduced by at least S, since each of the robots moved by at least S towards the center of $\mathcal{SEC}(P)$. Therefore the self-loop at state [IN0] can be repeated only finitely many times, ending in a configuration where either $|\mathcal{C}_{\text{int}}(P)| \geq 1$, two robots meet and create a multiplicity point, or all robots meet. Note also that if a multiplicity point is created after this step, then it is necessarily unique, as it could be created only by two or more robots from $\mathcal{C}_{\text{cir}}(P)$ meeting at the center point of $\mathcal{SEC}(P)$, which is the only possible intersection point for the trajectories of the robots.

State [IN1]: If $N \geq 4$ and $|\mathcal{C}_{\text{int}}(P)| = 1$, then a similar argument holds; thus also here the self-loop can be taken only finitely many times or the system's state changes to a state where $|\mathcal{C}_{\text{int}}(P)| \geq 2$, namely, [IN2] or [IN3], or a multiplicity point is created, leading to state [MULT]. If a multiplicity point is created as a result of this step, then it is unique, as it could be created only by one or more robots from $\mathcal{C}_{\text{cir}}(P)$ and the inner robot R_i, since the trajectories of any two robots moving towards R_i intersect only at the location of R_i.

State [IN2]: If $N \geq 4$ and $|\mathcal{C}_{\text{int}}(P)| = 2$, then the only outcome of this state could be a single multiplicity point, as can be verified by inspecting the possible substates. In this state, only two robots are active and move towards one goal point; thus the multiplicity point is unique.

State [IN2(a)]: If p_i and p_j do not share a cell, then both robots are instructed to go to the center of $\mathcal{SEC}(P)$. Eventually, either both robots will meet there, leading to state [MULT], or only one will arrive at the center; thus the two robots now share a cell, leading to state [IN2(b)].

State [IN2(b)]: If p_i and p_j share a single cell, $\text{Cell}(R_k)$, then either they meet on their way to R_k or one or both meet R_k at location p_k, thus leading to state [MULT].

State [IN2(c)]: If p_i and p_j share two cells, then the following possibilities may occur. Either the robot closer to $\mathcal{C}_{\text{cir}}(P)$, say R_i, meets with its target, say R_k, or R_j will meet R_i on its way, thus creating a multiplicity point, leading to state [MULT], or R_j enters the interior of the sector $\text{Cell}(R_k)$, thus leading to state [IN2(b)].

State [IN3]: Finally, if $N \geq 4$ and $|\mathcal{C}_{\text{int}}(P)| \geq 3$, then the procedure is applied recursively on the inner robots, while the outer robots remain stationary. Thus, as seen above, a single multiplicity point is created on the lowest level of the recursion. □

LEMMA 3.7. *In an $(N, 1)$-crash system, if the initial configuration is in state* [MULT]*, i.e., it contains a single multiplicity point p^*, then Procedure* GoTo_Mult *guarantees that within finite time all nonfaulty robots gather at p^* while avoiding the creation of additional multiplicity points.*

Proof. Every robot with a free corridor towards p^* is instructed to go towards p^*; thus all nonfaulty robots will arrive at p^* within a finite time. If a robot R_i detects another robot on its trajectory towards p^*, it looks for a free corridor by moving orthogonally to the multiplicity point, while making sure that it does not obstruct the free corridor of any other robot. This is ensured by moving only so as to change its angle with respect to the x-axis and p^* by a third of the angle to the closest-angle neighboring robot R_j (see Figure 8(a)). Note that it is possible that R_j will also enter the same sector, due to the lack of consistent coordinate system (and in particular,

the absence of common orientation, which may cause the R_jth clockwise sector to be the same as the R_ith clockwise sector). However, even if R_j enters that clear sector, it will be in the "far" third of the sector.

It is also possible for $k \geq 3$ robots $R_{i_1}, \ldots, R_{i_k}$ to share a common corridor to p^* (see Figure 8(b)). In this case the one closest to p^*, say R_{i_1}, will move towards p^*, and the others might take the same new trajectory to p^*. However, on this new trajectory, only $k-1$ robots collide, so the closest to p^* has a free corridor, and only $k-2$ robots must shift orthogonally again. Hence if a robot has more than one robot on its trajectory towards p^*, then it will remain in state [Blocked] during finitely many cycles until it has a free corridor towards p^*; thus it will eventually switch to state [Free] and arrive at p^*. □

THEOREM 3.8. *Algorithm* $\mathsf{Gather}_{\mathrm{crash}}$ *solves the gathering problem in an* $(N, 1)$*-crash system under the* $\mathcal{SSYNC}$ *model for any* $N \geq 3$.

Proof. Since the initial configuration is legal, by Lemma 3.6 it is guaranteed that Procedure Create_Mult will lead to a single multiplicity point. By Lemma 3.7, applying Procedure GoTo_Mult on a system with one multiplicity point leads to the gathering of all nonfaulty robots at that point. □

4. Impossibility of gathering under Byzantine faults.

4.1. Impossibility results in the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ models. In [21] it is shown that the class of problems solvable in $\mathcal{ASYNC}$ is contained in the class of problems solvable in the $\mathcal{SSYNC}$ model. It follows that proving impossibility of gathering in an $(N, 1)$-Byzantine system in $\mathcal{SSYNC}$ also proves impossibility in $\mathcal{ASYNC}$. We next prove that in the $\mathcal{SSYNC}$ model it is impossible for any algorithm to achieve either gathering or convergence of three robots in the Byzantine fault model, even in the presence of at most one faulty robot.

Definition. A gathering algorithm $\mathcal{A}$ is called *hyperactive* if it instructs every robot to make a move in every cycle until the task is achieved; i.e., $\check{M}(N, \mathcal{A}) = N$.

THEOREM 4.1. *In a* $(3, 1)$*-Byzantine system under the* $\mathcal{SSYNC}$ *model, any non-hyperactive gathering algorithm will fail in achieving gathering or convergence.*

Proof. Suppose the system consists of three robots R_1, R_2, R_3, and there exists a scenario σ in which at some configuration C_1, R_1 is active, but the algorithm instructs it to stay in place. In this system, the adversary can do the following. It designates R_3 as faulty and executes the scenario σ with R_3 acting correctly up to a configuration C_1. At this cycle, it makes R_1 active and R_2 passive. As a result, neither R_1 nor R_2 moves in this cycle. In addition, the adversary moves R_3 to create a configuration C_2 that from R_2's point of view is equivalent to what R_1 has seen in C_1 (see Figure 10). The adversary now makes R_1 passive and R_2 active. Since R_2's state is equivalent to R_1's state in the previous configuration, the algorithm will now instruct R_2 to stay in place. The adversary can now switch from configuration C_1 to C_2 and back, forcing R_1 and R_2 to stay in place indefinitely. Therefore the algorithm fails to achieve gathering or convergence of the nonfaulty robots. □

Definition. A distributed robot algorithm is N*-diverging* if there exists an (N, f)-Byzantine system and a configuration in which the instructions of the algorithm combined with the actions of the adversary can cause two nonfaulty robots to increase the distance between them. An example of divergence caused by the instructions of the algorithm is illustrated in Figure 11(a). An example of divergence caused by the intervention of the adversary is illustrated in Figure 11(b), where robot R_1 is stopped short of reaching its goal point.

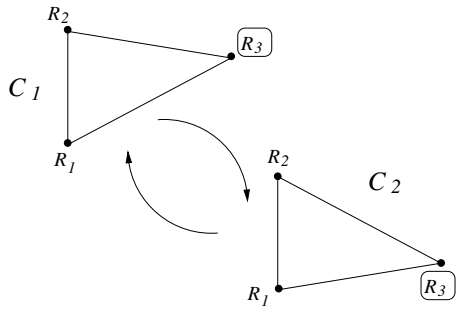


FIG. 10. *Theorem* 4.1.

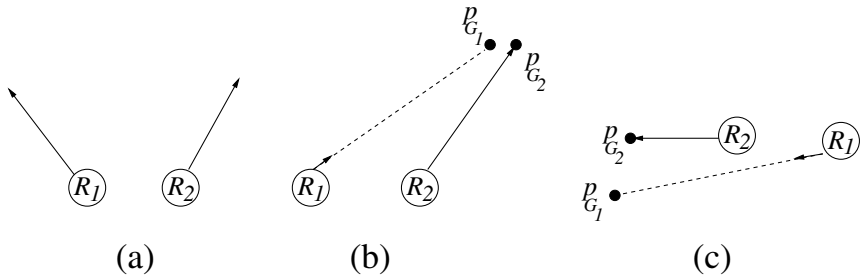


FIG. 11. *Divergence of robots.*

The premature-stopping technique. Our impossibility proofs make extensive use of the following technique. In order to cause two robots to diverge in some given configuration C of a given system $\mathcal{T}$, the adversary can stop a nonfaulty robot R_i after traversing a relatively small distance S_i. Note that S_i might be smaller than S in the current system, in which case the adversary is not permitted to stop R_i prematurely. However, as the algorithm is required to be valid in any system, it is intuitively clear that we may always consider a different system $\mathcal{T}'$ with S small enough to allow a movement of distance S_i. Moreover, since the algorithm is unaware of the value of S, it cannot distinguish between identical configurations in the two systems $\mathcal{T}$ and $\mathcal{T}'$ and will issue the same instructions to each robot in configuration C in $\mathcal{T}$ and $\mathcal{T}'$. Therefore the premature-stopping technique can be applied with any movement length greater than zero. We make this argument more formal in the proofs that follow.

As for the applicability of the premature-stopping technique, the adversary can apply it to cause the robots to diverge in any case where two robots move towards their respective goals on nonintersecting trajectories (see Figure 11(c)). In addition, even if the trajectories do intersect, the adversary can still apply the technique in some cases and again cause divergence, as seen in Figure 11(b). (One example for a case in which the premature stopping technique cannot help the adversary to force divergence is when the trajectories of the two robots R_1 and R_2 intersect and the angle between $p(R_1)$, $p(R_1)$ and the intersection point is at least $\pi/2$; see Lemma 3.3 and Figure 2.)

LEMMA 4.2. *In the $\mathcal{SSYNC}$ (or even the $\mathcal{FSYNC}$) model, a 3-diverging algorithm will fail to achieve gathering or convergence.*

Proof. Suppose, towards contradiction, that there exists a 3-diverging algorithm $\mathcal{A}$ that solves the gathering problem. Consider a $(3, 1)$-Byzantine system $\mathcal{T}$ with robots

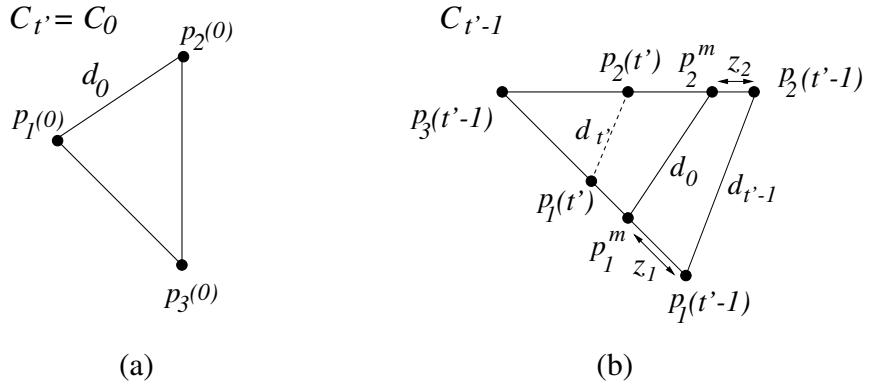


FIG. 12. *Lemma* 4.2.

R_1, R_2, and R_3, and a configuration C_0 on which $\mathcal{A}$'s instructions and the adversary's actions cause R_1 and R_2 to increase their distance. For $t \geq 0$ and $i = 1, 2$, denote by $p_i(t)$ the location of robot R_i in configuration C_t, and let $d_t = \text{dist}(p_1(t), p_2(t))$. Let $\sigma = \{C_0, C_1, \ldots, C_k\}$ be the sequence of configurations in an execution of the algorithm in which the adversary intervenes on the transition from C_0 to C_1 so as to increase $\text{dist}(R_1, R_2)$ but does not intervene thereafter, and in C_k all robots are gathered in one point. Note that $d_1 > d_0$. Let $t' = \min\{t \mid d_t \leq d_0, 2 \leq t \leq k\}$. By continuity considerations, as $d_{t'-1} > d_0$ and $d_{t'} \leq d_0$, there must be a time during the transition from $C_{t'-1}$ to $C_{t'}$ in which the robots R_1 and R_2 were located in middle points p_1^m and p_2^m, respectively, at distance exactly $\text{dist}(p_1^m, p_2^m) = d_0$. For $i = 1, 2$, let $z_i = \text{dist}(p_i(t'-1), p_i^m)$ (see Figure 12) and denote the minimum distance any robot traversed at any cycle $0 \leq t \leq t'-1$ by z_3.

Now replace the $(3,1)$-Byzantine system $\mathcal{T}$ by another system $\mathcal{T}'$ where $S = \min\{z_1, z_2, z_3\}$ and consider the following scenario. The adversary designates R_3 as faulty and executes the scenario σ up to $C_{t'-1}$. At this cycle, it stops R_1 and R_2 at points p_1^m and p_2^m, respectively, and moves the faulty robot R_3 to the exact same position it occupied in C_0. Thus, the new configuration $\tilde{C}_t$ is identical to C_0; hence the robots R_1 and R_2 will diverge again, and the system can be made to cycle through the configuration sequence $\{C_0, \ldots, \tilde{C}_t\}$ indefinitely, and thus R_1 and R_2 will never meet, contradicting the assumption. □

OBSERVATION 4.3. *Let $\mathcal{A}$ be an algorithm operating in a $(3,1)$-Byzantine system. Let $\vec{L}_{i,j}$ be the straight half-line starting at p_i and going through p_j. Suppose that in some configuration C, $\mathcal{A}$ instructs R_i and R_j to move on vectors $\vec{\mathrm{v}}_i$ and $\vec{\mathrm{v}}_j$ towards destination points g_i and g_j, respectively. Denote the angle between $\vec{L}_{i,j}$ and $\vec{\mathrm{v}}_i$ (measured from $\vec{L}_{i,j}$ in the counterclockwise direction) by μ_i, and the angle between $\vec{L}_{i,j}$ and $\vec{\mathrm{v}}_j$ by μ_j (see Figure 13). Then each of the following is a sufficient condition for $\mathcal{A}$ to be 3-diverging:*

(C1) $0 \leq \mu_j \leq \mu_i \leq \pi$.
(C2) $\pi \leq \mu_i \leq \mu_j \leq 2\pi$.
(C3) $0 \leq \mu_i \leq \pi \leq \mu_j \leq 2\pi$ *or* $0 \leq \mu_j \leq \pi \leq \mu_i \leq 2\pi$.
(C4) $0 \leq \mu_i < \mu_j \leq \pi$ *and either* $\mu_i \geq \pi/2$ *or* $\mu_j \leq \pi/2$.
(C5) $\pi \leq \mu_j < \mu_i \leq 2\pi$ *and either* $\mu_i \leq 3\pi/2$ *or* $\mu_j \geq 3\pi/2$.

Proof. To show that $\mathcal{A}$ is 3-diverging in each of these cases, we have to show a scenario in which the instructions of $\mathcal{A}$ combined with the actions of the adversary will

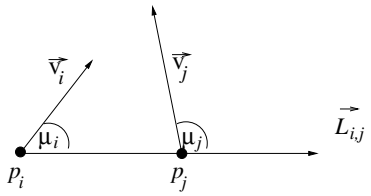


FIG. 13. *Observation* 4.3.

cause R_i and R_j to increase the distance between them. Denote the current distance between R_i and R_j by d_1, the location of R_i after traversing a distance S_i by p'_i, and the location of R_j after traversing a distance S_j by p'_j. Let $d_2 = \text{dist}(p'_i, p'_j)$.

Case (C1). It is easy to see that if $\mu_i = \mu_j = \pi/2$ and the robots move in different distances towards their goals, then $d_2 > d_1$ (as the hypotenuse in a right triangle is the longest side in the triangle). Therefore, in a system where $S = \min\{S_i, S_j\}, S_i \neq S_j$, it is enough that the adversary applies the premature-stopping technique, stops R_i after exactly a distance S_i, and stops R_j after traversing S_j. If $\mu_i = \mu_j < \pi/2$, then, applying again the premature-stopping technique, the adversary stops R_j after exactly a distance S and lets R_i continue traversing any distance greater than S. Similarly, if $\mu_i = \mu_j > \pi/2$, then the adversary stops R_i after traversing exactly a distance S and lets R_j traverse any distance greater than S. Finally, if $0 < \mu_j < \mu_i < \pi$, then the adversary can apply the premature-stopping technique and stop R_j after traversing a small distance and let R_j continue its movement as planned. In all cases, $d_1 > d_2$.

Case (C2). This case is simply a reflection of Case (C1).

Case (C3). It is easy to see that the trajectories of R_i and R_j diverge and never intersect. Traversing on those trajectories might not always cause divergence (for example, if the trajectory of R_j runs close to the current location of R_i as in Figure 11(c)), but by applying the premature-stopping technique as explained earlier, the adversary can cause divergence.

Case (C4). If $0 \leq \mu_i < \mu_j \leq \pi$ and either $\mu_i \geq \pi/2$ or $\mu_j \leq \pi/2$, then $\vec{v}_i$ and $\vec{v}_j$ intersect at some point, p^I. Without loss of generality let $\mu_i \geq \pi/2$. Drop a perpendicular line from p_j to the line going through p^I and p_i and let q_0 be the intersection point (see Figure 14). Let $d_3 = \text{dist}(p_j, p'_i)$. Consider the triangle $\triangle p'_i p_j q_0$. Clearly $d_3 > d_1$; therefore as $S_j \to 0$, $d_2 \to d_3$, and hence $d_2 > d_1$. Now apply the premature-stopping technique by stopping R_j after traversing a distance of exactly S, where S is very small (tending to 0), and allow R_i to traverse a distance S_i, thus causing $d_2 > d_1$.

Case (C5). This case is a reflection of Case (C4). □

THEOREM 4.4. *In a* $(3,1)$*-Byzantine system under the* $\mathcal{SSYNC}$ *model it is impossible to perform successful gathering or convergence.*

Proof. Consider a gathering algorithm $\mathcal{A}$ and an initial setting in which the three robots R_1, R_2, and R_3 are collinear, with R_2 in the middle. If the algorithm instructs R_2 to remain stationary, then it is nonhyperactive and by Theorem 4.1 will not achieve gathering. From Observation 4.3 it follows that if $0 \leq \mu_1, \mu_2, \mu_3 \leq \pi$, then in order to avoid being 3-diverging, necessarily $\mu_3 > \mu_2 > \mu_1$ (see Figure 15). But under this assumption, if $\mu_2 \geq \pi/2$, then applying Case (C4) of Observation 4.3 with respect to p_2 and p_3 yields that $\mathcal{A}$ is 3-diverging. If, on the other hand, $\mu_2 \leq \pi/2$, then applying Case (C4) of Observation 4.3 with respect to p_1 and p_2 yields the same conclusion.

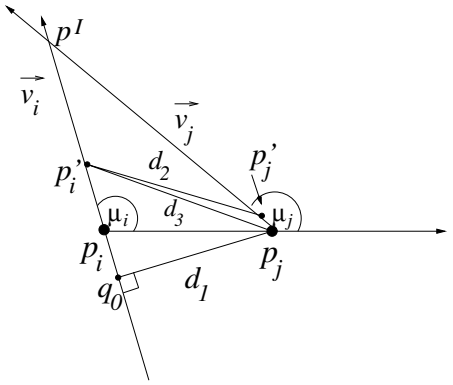


FIG. 14. *Observation* 4.3*, Case* (C4).

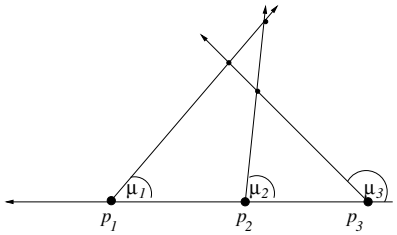


FIG. 15. *Illustration of proof of Theorem* 4.4.

A similar argument applies in case $\pi \leq \mu_1, \mu_2, \mu_3 \leq 2\pi$. Finally, if $\mu_1 > \pi$ and $\mu_2, \mu_3 < \pi$ or $\mu_1, \mu_2 > \pi$ and $\mu_3 < \pi$, then algorithm $\mathcal{A}$ is 3-diverging by Case (C3) of Observation 4.3. Thus, by Lemma 4.2, algorithm $\mathcal{A}$ fails to achieve gathering or convergence. □

We remark that in the $\mathcal{FSYNC}$ model, an N-diverging algorithm for $N > 3$ will not necessarily fail. In particular, the algorithms suggested in subsection 5.3 for the $\mathcal{FSYNC}$ model might be diverging yet still achieve gathering. Also, in the $\mathcal{FSYNC}$ model, a nonhyperactive algorithm will not necessarily fail. In particular, the gathering algorithm for $N = 3$ suggested in section 5 for the $\mathcal{FSYNC}$ model is not always hyperactive (for example, when the three robots are collinear, the robot lying in the middle is instructed to remain still). In fact, the converse may hold; namely, in the $\mathcal{FSYNC}$ model, a *hyperactive* algorithm might be problematic. For example, it is shown in the following lemma that in a one-dimensional setting, the adversary can cause failure of every hyperactive algorithm.

LEMMA 4.5. *In the $\mathcal{FSYNC}$ model, a hyperactive algorithm for a one-dimensional $(3,1)$-Byzantine system will fail to achieve gathering or convergence.*

Proof. Consider a $(3,1)$-Byzantine system on the line. Consider an arbitrary configuration C in which all robots are instructed by the algorithm to move. Without loss of generality suppose that at least two of the robots, say R_1 and R_2, are instructed to move to the right. Let ϵ_i denote the distance traversed by the robot R_i in the current round, and without loss of generality suppose $\epsilon_1 \leq \epsilon_2$. Then the same behavior will occur in a robot system in which $S \leq \epsilon_1$. In such a system, the adversary can stop R_2 after traversing a distance of only ϵ_1. The adversary can also fail the third robot R_3,

making it move to the right to distance ϵ_1. The resulting configuration is identical to the original C, implying that the adversary can keep the system at this configuration indefinitely. □

4.2. Intuition: Problems with previous approaches. The difficulty of handling a system of autonomous mobile robots with Byzantine faults is due to, among other reasons, the conventions regarding multiplicities on which most existing algorithms rely. In particular, these algorithms are based on enforcing the following conventions: (a) No more than one multiplicity point is created throughout the execution of the algorithm until successful gathering is achieved. (b) All robots lying in a multiplicity point remain stationary. (c) Robots lying in a multiplicity point are never separated again. These conventions are used for both gathering algorithms and other pattern formation algorithms; see [9].

All of the above assumptions no longer hold in a system where Byzantine faults might occur. First, the adversary could create a second multiplicity point as soon as it detects one such point, by "failing" a robot that does not lie in the multiplicity point and sending it to the location of yet another currently single robot. As a result, in the gathering problem assumption (b) cannot be relied on. Assumption (c) is violated even if the algorithm instructs all robots lying on the same point to move towards the *same* destination point, as the adversary could stop their movement in different locations.

Since all known algorithms rely on conventions (a)–(c) listed above, which can be violated in a system consisting of N robots with even one Byzantine faulty robot, it is clear that those algorithms fail to achieve gathering.

To get a feel for the possible complications that may occur in this model, let us consider some simple solutions one might propose for the problem. One natural general approach for attacking the problem is to try to gradually reduce the number of distinct points where the robots reside, by gathering partial subsets of robots at different points. A possible algorithm attempting to achieve that is one that requires each robot, in each cycle, to move towards its *closest* neighbor. This may lead to deadlocks once the robots pair up, since each one's closest neighbor already resides at the same location. Therefore the algorithm should instruct each robot to move towards the closest robot among those currently residing at locations *other* than its own. One problem that arises is that sets of robots that have already met might break up again; hence "progress" is hard to measure. Another obvious problem is that of symmetry breaking. Even ignoring this problem, this approach can still lead to nonconverging scenarios. For instance, suppose that the N robots are located on a straight line, with R_i at location $x_i = i(i-1)/2$. Then the algorithm requires R_1 to move towards R_2 and R_i to move towards R_{i-1} for every $2 \le i \le N$. However, if R_1 is faulty and chooses to move *away* from R_2, and all robots traverse exactly a distance S, then the configuration is translated by S in the $-x$ direction and is otherwise unchanged. (See Figure 16.)

Another natural algorithm is based on computing the center of gravity p_G of the configuration and going to p_G. This algorithm can be failed by the adversary in

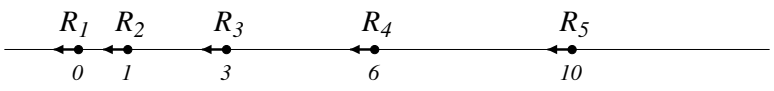


FIG. 16. *Bad scenario for nearest-neighbor algorithm.*

similar manner, by failing a robot located far from the rest and taking it to a walk towards infinity, forcing the entire pack of nonfaulty robots to be dragged along.

5. Fault-tolerant gathering in the $\mathcal{FSYNC}$ model.

5.1. Preliminaries. We now discuss the problem of gathering N autonomous mobile robots in an (N, f)-Byzantine system under the fully synchronous model. We use the following notation. Denote the *geometric span* (or *diameter*) of the set of points P by

$$\mathrm{Span}(P) \;=\; \max\{\mathrm{dist}(p,q) \mid p,q \in P\}.$$

Denote the convex hull of a multiset of points P by $H(P)$, and the set of vertices of $H(P)$ by $V_{\mathrm{H}}(P)$. Denote the set of (N to $N-f$) nonfaulty robots by $\mathcal{R}_{NF}$. Denote the *center of gravity* (or barycenter) of a multiset P of $n \geq 3$ points $p_i = (x_i, y_i), i = 1, \ldots, N$ by

$$C_{\mathrm{grav}}(P) = \left(\frac{\sum_{i=i}^{N} x_i}{N} \;,\; \frac{\sum_{i=i}^{N} y_i}{N} \right).$$

Define the sum of distances between all pairs of nonfaulty robots as

$$D_{\mathrm{tot}}(P) = \sum_{R_i, R_j \in \mathcal{R}_{NF}} \mathrm{dist}(R_i, R_j).$$

Note that for any set of points P, while the center of gravity $C_{\mathrm{grav}}(P)$ is defined in terms of the point coordinates in some specific coordinate system, the resulting point is independent of the particular coordinate system in use. Hence for a set of robots in some arbitrary configuration C in the plane, whenever each of the robots computes $C_{\mathrm{grav}}(P)$, the resulting point computed by the different robots is the same, even if each robot has its own coordinate system.

Definition. A robot algorithm is *concentrating* if it satisfies the following properties:

1. It is nondiverging; i.e., no two nonfaulty robots will increase the distance between them in any round.
2. There exists a constant $c > 0$ such that at each step, at least one pair of nonfaulty robots that are at different locations either meets or decreases the distance between them by at least c.

LEMMA 5.1. *Let $\mathcal{A}$ be a concentrating algorithm. Then in a $(3,1)$-Byzantine system under the $\mathcal{FSYNC}$ model, $\mathcal{A}$ achieves gathering.*

Proof. If in each cycle D_{tot} decreases by a constant amount c, then within a finite number of cycles $\mathcal{A}$ achieves gathering of all nonfaulty robots (since D_{tot} must reach 0). If there is indeed one faulty robot, then there may be only one pair of nonfaulty robots. The algorithm $\mathcal{A}$ ensures that the distance between the two nonfaulty robots decreases by at least a constant c in each cycle; hence $D_{\mathrm{tot}}^{new} \leq D_{\mathrm{tot}}^{old} - c$ and therefore these two robots will eventually meet. If all three robots are nonfaulty, then $\mathcal{A}$ ensures that the distance between at least one pair, say R_1 and R_2, decreases by at least c while $\mathrm{dist}(R_1, R_3)$ and $\mathrm{dist}(R_2, R_3)$ do not increase (since $\mathcal{A}$ is nondiverging); hence $D_{\mathrm{tot}}^{new} \leq D_{\mathrm{tot}}^{old} - c$ and $\mathcal{A}$ achieves gathering. ∎

Definition. A distributed robot algorithm $\mathcal{A}$ is said to *dictate* 2*-pair convergence* in a given cycle if in that cycle it instructs two distinct pairs of robots to decrease the distance between them by a constant amount. A distributed robot algorithm $\mathcal{A}$

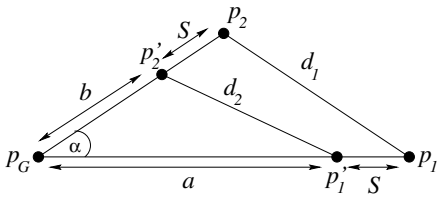


FIG. 17. *Proof of Lemma* 5.2.

is said to *dictate triple convergence* in a given cycle if in that cycle it instructs three robots to decrease by a constant amount the distance between every pair of them.

Note that for $N = 3$, triple convergence implies also nondivergence and hence also concentration. Note also that these conditions do not require that the robots involved be nonfaulty; in particular, one of them may fail and disobey the algorithm's instructions, in which case its distances will not decrease as needed. Nevertheless, these conditions turn out to be sufficient for gathering in certain settings.

LEMMA 5.2. *Consider two robots R_1 and R_2, initially located at the points p_1 and p_2, which traverse the same distance S' towards a common meeting point p_G, and let $\alpha = \angle p_1 p_G p_2$. If $\alpha \le \pi/2$, then the distance between them decreases by at least $S'(1 - \cos\alpha)$.*

Proof. Let p_1' and p_2' denote the new location of R_1 and R_2 after moving a distance S' towards p_G, and let $d_1 = \mathrm{dist}(p_1, p_2)$, $d_2 = \mathrm{dist}(p_1', p_2')$, $a = \mathrm{dist}(p_1', p_G)$, and $b = \mathrm{dist}(p_1', p_G)$ (see Figure 17).

By the cosine theorem on the triangles $\triangle p_1 p_G p_2$ and $\triangle p_1' p_G p_2'$, it follows that

$$d_1^2 = (a + S')^2 + (b + S')^2 - 2(a + S')(b + S')\cos\alpha,$$
$$d_2^2 = a^2 + b^2 - 2ab\cos\alpha.$$

Therefore

$$d_1 - d_2 = \frac{2a + 2b + S'}{d_1 + d_2} \cdot S'(1 - \cos\alpha).$$

By the triangle inequality on triangles $\triangle p_1 p_G p_2$ and $\triangle p_1' p_G p_2'$ it follows that $a + b + 2S' > d_1$ and $a + b > d_2$. Therefore $\frac{2a+2b+S'}{d_1+d_2} > 1$, so $d_1 - d_2 > S'(1 - \cos\alpha)$. In the range $(0, \pi/2]$ this value is always greater than 0 and is equal to 0 if and only if $\alpha = 0$. □

5.2. Gathering on a (3, 1)-Byzantine system in the $\mathcal{FSYNC}$ model. Let us next describe a gathering algorithm for three robots in an $\mathcal{FSYNC}$ model with at most one Byzantine fault. The input to this procedure is a configuration $P = \{p_1, p_2, p_3\}$.

Procedure 3-Gather$_{\mathrm{Byz}}(P)$

1. **State [Collinear]:** p_1, p_2, p_3 are collinear with p_2 in the middle:
   Set $p_G \leftarrow p_2$.
2. **State [Triangle]:** The three points form a triangle:
   Set p_G to be the intersection point of the three angle bisectors of the triangle $\triangle p_1 p_2 p_3$.

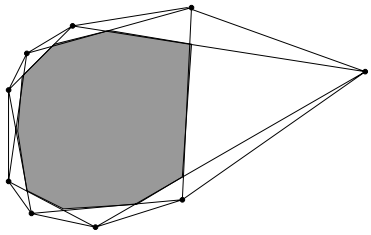


FIG. 18. *Illustration of* H^1_{int}.

Observe that the case of two robots residing at the same point, say $p_1 = p_2$, is handled by step 1 of the procedure. In this case, $p_G = p_1$; thus R_1 and R_2 stay in place and R_3 is required to move towards them.

Analysis.

THEOREM 5.3. *Algorithm* 3-Gather$_{\text{Byz}}$ *solves the gathering problem in a* $(3,1)$*-Byzantine system under the* $\mathcal{FSYNC}$ *model.*

Proof. Let us first consider the case when R_1, R_2, and R_3 are collinear, say, with R_2 in the middle. Since both extreme robots are instructed to move towards R_2, and R_2 is instructed to stay in place, it is clear that the instructions of the algorithm ensure that $\text{dist}(R_1, R_3)$ decreases in each cycle by at least S (or they meet), and also that $\text{dist}(R_1, R_2)$ and $\text{dist}(R_2, R_3)$ decrease by at least S (or they meet). Hence the algorithm dictates triple convergence in each cycle.

Next, suppose that R_1, R_2, and R_3 are not collinear. By Lemma 3.2, the angle between every two robots and p_G is greater than $\pi/2$. Therefore, by Lemma 3.3, we again have triple convergence.

Therefore in each cycle, whether the robots are collinear or not, triple convergence is ensured. Since for $N = 3$ triple convergence implies nondivergence as well, the algorithm achieves gathering by Lemma 5.1. ∎

5.3. Gathering for $f \geq 1$ and $N \geq 3f + 1$ in the $\mathcal{FSYNC}$ model. In this section we propose an algorithm for solving the gathering problem in an (N, f)-Byzantine system, where $N \geq 3f + 1$ in the $\mathcal{FSYNC}$ model. The main idea of the algorithm is to ensure that the goal point selected in each cycle falls in the convex hull of the *nonfaulty* robot locations. As shown later, this ensures that the geometric span of the set of locations of the nonfaulty robots decreases by at least $0.25S$; thus the robots will meet within a finite number of cycles. Due to its high complexity, this algorithm is only of theoretical merit, except for small values of f.

Definition. The *hull intersection* $H^k_{\text{int}}(P)$ is the convex set created as the intersection of all $\binom{N}{k}$ sets $H(P \setminus \{p_{i_1}, \ldots, p_{i_k}\})$ for $1 \leq k \leq N$, $p_{i_j} \in P$. (See Figure 18 for $k = 1$.)

The algorithm. Consider the following Procedure Gather$_{\text{Byz}}$ for determining the goal point p_G in each cycle. The input to this procedure is a configuration $P = \{p_1, \ldots, p_N\}$, and f is the maximum number of faulty robots.

Procedure Gather$_{\text{Byz}}(P)$

1. Compute $Q \leftarrow V_{\text{H}}(H^f_{\text{int}}(P))$.
2. Set $p_G \leftarrow C_{\text{grav}}(Q)$.

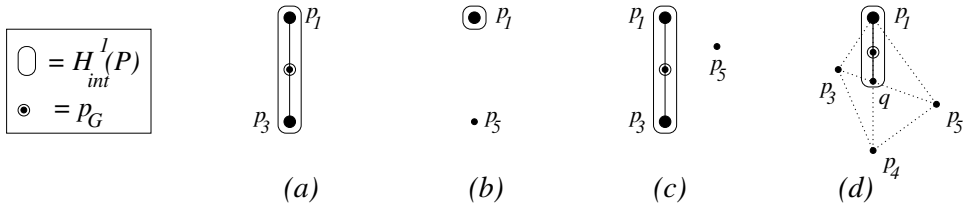


FIG. 19. *Illustration of Procedure* $\mathsf{Gather}_{\mathrm{Byz}}$ *in a* $(5,1)$*-Byzantine system.*

To illustrate the algorithm, let us consider a number of possible configurations of a $(5,1)$-Byzantine system (see Figure 19).

(a) $p_1 = p_2$, and $p_3 = p_4 = p_5$. Then $H^1_{\mathrm{int}}(P)$ is the segment $\overline{p_1p_3}$, and p_G is its midpoint.
(b) $p_1 = p_2 = p_3 = p_4 \neq p_5$. Then $H^1_{\mathrm{int}}(P) = \{p_1\}$, and $p_G = p_1$.
(c) $p_1 = p_2$, $p_3 = p_4$, and p_5 is distinct. Then $H^1_{\mathrm{int}}(P)$ is the segment $\overline{p_1p_3}$, and p_G is its midpoint.
(d) $p_1 = p_2$, and p_3, p_4, p_5 are distinct. Then $H^1_{\mathrm{int}}(P)$ is some segment $\overline{qp_1}$, and p_G is its midpoint.

Analysis. Let us first prove that the algorithm is well defined. For this we have to show that the set Q is nonempty.

Helly's theorem for $d = 2$ (cf. [28, Theorem 4.1.1]). Let $\mathcal{S}$ be a finite family of at least three convex sets in $\mathbb{R}^2$. If every three members of $\mathcal{S}$ have a point in common, then there is a point common to all members of $\mathcal{S}$.

LEMMA 5.4. *For a multiset* $P = \{p_1, \ldots, p_N\}, N \geq 3k+1$, $H^k_{\mathrm{int}}(P)$ *is convex and nonempty.*

Proof. $H^k_{\mathrm{int}}(P)$ is convex as it is the intersection of $\binom{N}{k}$ convex sets. We prove that it is nonempty by Helly's theorem. Consider three arbitrary sets $P^l = \{p^l_1, \ldots, p^l_k\} \subseteq P, 1 \leq l \leq 3$, and let $Q^l = H(P \setminus P^l)$, $1 \leq l \leq 3$. Then $Q^1 \cap Q^2 \cap Q^3$ contains at least $P' = P \setminus (P^1 \cup P^2 \cup P^3)$. As $|P| \geq 3k+1$, $|P'| \geq 1$. It follows that the intersection of every three such sets is nonempty, and by Helly's theorem $V_{\mathrm{H}}(H^k_{\mathrm{int}}(P))$ is nonempty as well. ∎

The analysis of Procedure $\mathsf{Gather}_{\mathrm{Byz}}$ is based on showing that if a set of K robots $R_1, \ldots, R_K$ initially located at the points $P = \{p_1, \ldots, p_K\}$ move towards a point p_G in their convex hull $H(P)$ and their new positions are at the points $P' = \{p'_1, \ldots, p'_K\}$, then their geometric span decreases by at least cS for some constant $c \geq 1/4$; i.e., $\mathrm{Span}(P') \leq \mathrm{Span}(P) - cS$. Consequently, the robots will meet within a finite number of cycles.

LEMMA 5.5. *Let* $P = \{p_1, \ldots, p_k\}$ *be a set of points in the plane.*

1. $\mathrm{Span}(P) \geq \mathrm{dist}(p, p')$ *for every two points* p, p' *in the convex hull* $H(P)$.
2. *The geometric span is attained by two points* $p_a, p_b \in P$ *that occur as vertices in* $H(P)$.
3. *Moreover, for every point* p_G *in* $H(P)$, $\angle p_a p_G p_b \geq \pi/4$.

Proof. Consider two arbitrary points p, p' inside $H(P)$. By the definition of the convex hull, it is clear that the segment $\overline{pp'}$ falls entirely within the convex hull of P. Therefore this segment can be extended in both directions towards the circumference of $H(P)$, hitting it at the points q, q'. Hence $\mathrm{dist}(p, p') \leq \mathrm{dist}(q, q')$. If the points q and q' are vertices of the convex hull, then $q, q' \in P$, and we are done. So now suppose this is not the case. If q is not a vertex of $H(P)$, then it occurs on an edge $\overline{p_i p_{i+1}}$. In this case, at least one of the two adjacent vertices of the convex hull,

without loss of generality p_i, satisfies that $\mathrm{dist}(q', p_i) > \mathrm{dist}(q', q)$. Similarly, if q' is not a vertex, then it occurs on an edge $\overline{p_j p_{j+1}}$, and again without loss of generality $\mathrm{dist}(p_j, p_i) > \mathrm{dist}(q', p_i)$. Hence combined, $\mathrm{Span}(P) \geq \mathrm{dist}(p, p')$. Therefore for each such segment $\overline{pp'}$ there exists a segment $\overline{p_i p_j}$, $p_i, p_j \in P$, whose length is greater than or equal to $\mathrm{dist}(q, q')$.

The proof used to establish the first claim of the lemma also yields the second claim, as it shows that for any two points q, q' that are not both vertices of $H(P)$, there exist two vertices $p_i, p_j \in P$ of $H(P)$ satisfying $\mathrm{dist}(p_i, p_j) > \mathrm{dist}(q, q')$; hence $\mathrm{Span}(P)$ cannot be attained by those points q, q'.

The third claim of the lemma is proved as follows. Let p_a, p_b be the two vertices of $H(P)$ attaining $\mathrm{Span}(P)$, and suppose, towards contradiction, that there exists a point p_G in $H(P)$ such that $\alpha = \angle p_a p_G p_b < \pi/4$. Consider the triangle $\triangle p_a p_G p_b$. Let $\beta = \angle p_a p_b p_G$ and $\gamma = \angle p_b p_a p_G$. Without loss of generality assume that $\beta \geq \gamma$. Then

$$\alpha < \pi/4 < 3\pi/8 < (\pi - \alpha)/2 = (\beta + \gamma)/2 < \beta < \beta + \gamma = \pi - \alpha.$$

Hence $\sin\beta > \sin\alpha$. Also, by the sine theorem on the triangle $\triangle p_a p_G p_b$,

$$\frac{\mathrm{dist}(p_a, p_b)}{\mathrm{dist}(p_a, p_G)} = \frac{\sin\alpha}{\sin\beta}.$$

It follows that $\mathrm{dist}(p_a, p_G) > \mathrm{dist}(p_a, p_b)$. By part 1 of the lemma, $\mathrm{Span}(P) \geq \mathrm{dist}(p_a, p_G) > \mathrm{dist}(p_a, p_b)$, contradicting the assumption. □

LEMMA 5.6. *For every two sets of points P and Q, if $H(P) \subseteq H(Q)$, then* $\mathrm{Span}(P) \leq \mathrm{Span}(Q)$.

Proof. Let $p_a, p_b \in P$ be the vertices attaining the geometric span of P. As $P \subseteq H(P) \subseteq H(Q)$, also $p_a, p_b \in H(Q)$. Thus by Lemma 5.5, $\mathrm{Span}(P) = \mathrm{dist}(p_a, p_b) \leq \mathrm{Span}(Q)$. □

LEMMA 5.7. *If a set of K robots $R_1, \ldots, R_K$ initially located at the points $P = \{p_1, \ldots, p_K\}$ traverse the same distance S towards a point p_G in the convex hull $H(P)$ and their new positions are at the points $P' = \{p'_1, \ldots, p'_K\}$, then their geometric span decreases by at least cS for some constant $c \geq 1/4$; i.e.,* $\mathrm{Span}(P') \leq \mathrm{Span}(P) - cS$.

Proof. Let p_a, p_b be the two vertices of $H(P)$ attaining $\mathrm{Span}(P)$, and let p'_i, p'_j be the two vertices of $H(P')$ attaining $\mathrm{Span}(P')$. Note that p_G is internal also to $H(P')$. Hence, by part 3 of Lemma 5.5, it follows that $\alpha = \angle p_a p_G p_b \geq \pi/4$. If $\pi/4 \leq \alpha < \pi/2$, then according to Lemma 5.2, $\mathrm{dist}(p'_a, p'_b) \leq \mathrm{dist}(p_a, p_b) - (1 - \cos\alpha) \leq \mathrm{dist}(p_a, p_b) - 0.25S$. Also, if $\alpha \geq \pi/2$, then by Lemma 3.3, $\mathrm{dist}(p'_a, p'_b) \leq \mathrm{dist}(p_a, p_b) - 0.7S$. Therefore, in any case $\mathrm{dist}(p'_a, p'_b) \leq \mathrm{dist}(p_a, p_b) - 0.25S$. Also, since p_a, p_b attains $\mathrm{Span}(P)$, it follows that $\mathrm{dist}(p_a, p_b) \geq \mathrm{dist}(p_i, p_j)$; therefore

$$\begin{aligned}\mathrm{Span}(P') &= \mathrm{dist}(p'_i, p'_j) \leq \mathrm{dist}(p_i, p_j) - S/4 \leq \mathrm{dist}(p_a, p_b) - S/4 \\ &= \mathrm{Span}(P) - S/4. \qquad \square\end{aligned}$$

COROLLARY 5.8. *If a set of K robots $R_1, \ldots, R_K$ initially located at the points $P = \{p_1, \ldots, p_K\}$ move towards a point p_G in the convex hull $H(P)$ and their new positions are at the points $P' = \{p'_1, \ldots, p'_K\}$, then their geometric span decreases by at least cS for some constant $c \geq 1/4$; i.e.,* $\mathrm{Span}(P') \leq \mathrm{Span}(P) - cS$.

Proof. By the model assumption, each robot traverses a distance of at least S towards p_G. Let p''_i denote the point at a distance *exactly* S from p_i in the direction of p_G, and let $P'' = \{p''_1, \ldots, p''_K\}$. Clearly $H(P') \subseteq H(P'')$. By Lemma 5.6, $\mathrm{Span}(P') \leq \mathrm{Span}(P'')$. By Lemma 5.7, $\mathrm{Span}(P'') \leq \mathrm{Span}(P) - cS$. The claim follows. □

LEMMA 5.9. *If a set of K robots $R_1, \ldots, R_K$ move in every cycle t towards a point p_G in their convex hull, then the robots will meet within a finite number of cycles.*

Proof. For $t \geq 1$, denote by H_t the convex hull of the robot configuration at the beginning of cycle t. In each cycle, the robots move a distance of at least S towards a point p_G in the convex hull. Thus, by Corollary 5.8, $\mathrm{Span}(H_{t+1}) \leq \mathrm{Span}(H_t) - 0.25S$ for every t. Therefore within at most $4 \cdot \mathrm{Span}(H_1)/S$ cycles, the geometric span of the robot configuration will be 0; thus all robots meet. ∎

THEOREM 5.10. *Algorithm* $\mathsf{Gather}_{\mathrm{Byz}}$ *solves the gathering problem in an (N, f)-Byzantine system under the $\mathcal{FSYNC}$ model for any $N \geq 3f + 1$.*

Proof. By Lemma 5.9 it is sufficient to show that the goal point p_G selected in the cycle falls in $H(\mathcal{R}_{NF})$, the convex hull of the nonfaulty robots. To prove this, we check the goal point p_G determined in each cycle.

By definition, the set $H^f_{\mathrm{int}}(P)$ is contained in its entirety in $H(P)$ as well as in the convex hull of every $N - f$ points of P; thus, in particular, it falls in $H(\mathcal{R}_{NF})$. Since the center of gravity of a set of points is inside its convex hull, it follows that p_G is in $H(\mathcal{R}_{NF})$. By Lemma 5.4, it follows that $H^f_{\mathrm{int}}(P)$ is nonempty; thus the center of gravity of the set $V_{\mathrm{H}}(H^f_{\mathrm{int}}(P))$ is well defined. ∎

6. Open problems. The design of fault-tolerant distributed control algorithms for multiple robot systems is still far from being fully explored. Directions for future research include the following. To begin with, it may be useful to study other kinds of fault models in addition to the crash and Byzantine models, such as a model in which the robots might lose some of their movement control (for instance, lose control of their movement length), or a model in which robots might diverge from their original movement direction up to a certain percentage of error. It is also necessary to develop fault-tolerant algorithms for tasks other than gathering (e.g., formation of geometric patterns). The maximum number of faults under which a solution is still feasible, for gathering and other tasks, has yet to be established. Finally, it would be interesting to examine the effect of changes in some initial assumptions on the model's fault tolerance properties. Examples for possible model changes include partial nonobliviousness of the robots (e.g., robots equipped with a small amount of memory, say, allowing them to remember the subsequence of X most recent cycles), robots capable of partial agreement on their orientation, or robots capable of explicit communication (perhaps under certain limitations, e.g., only with nearby robots).

REFERENCES

[1] H. ANDO, Y. OASA, I. SUZUKI, AND M. YAMASHITA, *A distributed memoryless point convergence algorithm for mobile robots with limited visibility*, IEEE Trans. Robotics and Automation, 15 (1999), pp. 818–828.

[2] H. ANDO, I. SUZUKI, AND M. YAMASHITA, *Formation and agreement problems for synchronous mobile robots with limited visibility*, in Proceedings of the IEEE International Symposium on Intelligent Control, 1995, pp. 453–460.

[3] E. M. ARKIN, M. A. BENDER, S. P. FEKETE, J. S. B. MITCHELL, AND M. SKUTELLA, *The freeze-tag problem: How to wake up a swarm of robots*, in Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 2002, pp. 568–577.

[4] T. BALCH AND R. ARKIN, *Behavior-based formation control for multirobot teams*, IEEE Trans. Robotics and Automation, 14 (1998), pp. 926–939.

[5] Y. U. CAO, A. S. FUKUNAGA, AND A. B. KAHNG, *Cooperative mobile robotics: Antecedents and directions*, Autonomous Robots, 4 (1997), pp. 7–23.

[6] M. CIELIEBAK, P. FLOCCHINI, G. PRENCIPE, AND N. SANTORO, *Solving the robots gathering problem*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 2719, Springer-Verlag, Berlin, 2003, pp. 1181–1196.

[7] M. Cieliebak and G. Prencipe, *Gathering autonomous mobile robots*, in Proceedings of the 9th International Colloquium on Structural Information and Communication Complexity, 2002, pp. 57–72.

[8] R. Cohen and D. Peleg, *Convergence properties of the gravitational algorithm in asynchronous robot systems*, SIAM J. Comput., 34 (2005), pp. 1516–1528.

[9] X. Defago and A. Konagaya, *Circle formation for oblivious anonymous mobile robots with no common sense of orientation*, in Proceedings of the 2nd ACM Workshop on Principles of Mobile Computing, ACM Press, New York, 2002, pp. 97–104.

[10] M. Erdmann and T. Lozano-Perez, *On multiple moving objects*, Algorithmica, 2 (1987), pp. 477–521.

[11] M. Erdmann and T. Lozano-Perez, *On multiple moving objects*, in Proceedings of the IEEE International Conference on Robotics and Automation, 1986, pp. 1419–1424.

[12] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, *Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots*, in Algorithms and Computation, Lecture Notes in Comput. Sci. 1741, Springer-Verlag, Berlin, 1999, pp. 93–102.

[13] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, *Distributed coordination of a set of autonomous mobile robots*, in Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2000), 2000, pp. 480–485.

[14] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, *Gathering of autonomous mobile robots with limited visibility*, in Proceedings of the 18th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Comput. Sci. 2010, Springer-Verlag, Berlin, 2001, pp. 247–258.

[15] D. Jung, G. Cheng, and A. Zelinsky, *Experiments in realising cooperation between autonomous mobile robots*, in Proceedings of the International Symposium on Experimental Robotics, 1997, pp. 609–620.

[16] Y. Kuniyoshi, S. Rougeaux, M. Ishii, N. Kita, S. Sakane, and M. Kakikura, *Cooperation by observation: The framework and basic task patterns*, in Proceedings of the IEEE International Conference on Robotics and Automation, 1994, pp. 767–774.

[17] L. E. Parker, *Designing control laws for cooperative agent teams*, in Proceedings of the IEEE International Conference on Robotics and Automation, 1993, pp. 582–587.

[18] L. E. Parker and C. Touzet, *Multi-robot learning in a cooperative observation task*, in Distributed Autonomous Robotic Systems 4, Springer-Verlag, New York, 2000, pp. 391–401.

[19] L. E. Parker, C. Touzet, and F. Fernandez, *Techniques for learning in multirobot teams*, in Robot Teams: From Diversity to Polymorphism, T. Balch and L. E. Parker, eds., A K Peters, Natick, MA, 2002, pp. 191–236.

[20] G. Prencipe, *CORDA: Distributed coordination of a set of atonomous mobile robots*, in Proceedings of the 4th European Research Seminar on Advances in Distributed Systems, 2001, pp. 185–190.

[21] G. Prencipe, *Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots*, in Theoretical Computer Science, Lecture Notes in Comput. Sci. 2202, Springer-Verlag, Berlin, 2001, pp. 185–190.

[22] G. Prencipe, *Distributed Coordination of a Set of Atonomous Mobile Robots*, Ph.D. thesis, Universita Degli Studi Di Pisa, Pisa, Italy, 2002.

[23] K. Sugihara and I. Suzuki, *Distributed algorithms for formation of geometric patterns with many mobile robots*, J. Robotic Systems, 13 (1996), pp. 127–139.

[24] I. Suzuki and M. Yamashita, *Agreement on a common x-y coordinate system by a group of mobile robots* in Proceedings of the Dagstuhl Seminar on Modeling and Planning for Sensor-Based Intelligent Robots, 1996, pp. 305–321.

[25] I. Suzuki and M. Yamashita, *Distributed anonymous mobile robots: Formation and agreement problems*, in Proceedings of the 3rd Colloquium on Structural Information and Communication Complexity, 1996, pp. 313–330.

[26] I. Suzuki and M. Yamashita, *Distributed anonymous mobile robots: Formation of geometric patterns*, SIAM J. Comput., 28 (1999), pp. 1347–1363.

[27] M. Sztainberg, E. Arkin, M. Bender, and J. Mitchell, *Analysis of heuristics for the freeze-tag problem*, in Algorithm Theory—SWAT 2002, Lecture Notes in Comput. Sci. 2368, Springer-Verlag, Berlin, 2002, pp. 270–279.

[28] R. Wenger, *Helly-type theorems and geometric transversals*, in Handbook of Discrete and Computational Geometry, J. E. Goodman and J. O'Rourke, eds., CRC, Boca Raton, FL, 1997, pp. 63–82.

[29] D. Yoshida, T. Masuzawa, and H. Fujiwara, *Fault-tolerant distributed algorithms for autonomous mobile robots with crash faults*, Systems and Computers in Japan, 28 (1997), pp. 33–43.