Fault Detection and Identification by Autonomous Mobile Robots

Stefano Clemente
Department of Computer Science, Università degli Studi di Milano, Italy

Caterina Feletti
Department of Computer Science, Università degli Studi di Milano, Italy

Abstract

The *Look-Compute-Move* model (LCM) is adopted to study swarms of mobile robots that have to solve a given problem. Robots are generally assumed to be autonomous, indistinguishable, anonymous, homogeneous and to move on the Euclidean plane. Different LCM sub-models have been theorized to study different settings and their computational power. Notably, the literature has focused on four base models (i.e., $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, $\mathcal{LUMI}$) that differ in memory and communication capabilities, and in different synchronization modes (e.g., *fully synchronous* `FSYNCH`, *semi-synchronous* `SSYNCH`).

In this paper, we consider *fault-prone* models where robots can suffer from *crash faults*: each robot may irremediably stop working after an unpredictable time. We study the general Fault Detection (FD) problem which is solved by a swarm if it correctly detects whether a faulty robot exists in the swarm. The Fault Identification (FI) problem additionally requires identifying which robots are faulty. We consider 12 LCM sub-models ($\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, $\mathcal{LUMI}$, combined with `FSYNCH`, `SSYNCH`, and the *round-robin* `RROBIN`) and we study the (im)possibility of designing reliable procedures to solve FD or FI. In particular, we propose three distributed algorithms so that a swarm can collectively solve FD under the models $\mathcal{LUMI}^{\texttt{FSYNCH}}$, $\mathcal{FCOM}^{\texttt{FSYNCH}}$, and $\mathcal{LUMI}^{\texttt{RROBIN}}$.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Autonomous mobile robots, Faulty robots, Look-Compute-Move, Fault detection, Fault identification, Round-robin

Digital Object Identifier 10.4230/LIPIcs.SAND.2025.10

1 Introduction

Imagine having a swarm of firefighter robots still in an area, aimed at surveilling it from fires and extinguishing them. Such robots perform an infinite sequence of computational cycles: at each clock, some of them are activated, they look at the environment, they process the snapshot just taken and, if a fire has been sensed, they properly move to fight it. They are provided with limited, energy-saving message protocols for intra-swarm communication; external communication is allowed only in case of emergency, due to its consumption.

From a theoretical point of view, systems and scenarios like this have been deeply studied over the last two decades. Specifically, the *Look-Compute-Move* (LCM) model [19] has been adopted to study dynamic distributed systems and formalize swarm of elementary and mobile robots with computational capabilities (network of sensors, drones, mobile software agents...). In this model, an activated robot performs an LCM cycle, so it looks at its surroundings, computes its next position by executing a distributed algorithm, and then reaches it. Much literature has been interested in designing and analyzing distributed algorithms the swarm can use to solve a collaborative task, called *problem*. Typically, robots are required to arrange according to some conditions (`Pattern Formation` [11, 17, 27, 29], `Gathering` [10, 16, 18, 21], or `Scattering` [20, 23]), or to travel in the environment (`Flocking` [7], `Exploration` [3]). Generally, robots are assumed to be autonomous, homogeneous, indistinguishable, anonymous, and punctiform entities in the Euclidean plane. According to storage and communication capabilities, four settings have been proposed to study different computational powers: $\mathcal{OBLOT}$ (oblivious and silent robots), $\mathcal{FSTA}$ (silent but with constant-size storage capabilities), $\mathcal{FCOM}$ (oblivious but with constant-size communication capabilities) and $\mathcal{LUMI}$ (with constant-size communication and storage capabilities). Transversally, different modes have been proposed in terms of robot activation and synchronization. In the *semi-synchronous* mode (SSYNCH), time is split into atomic *rounds* and at each round, an arbitrary non-empty subset of the swarm is activated and it executes an LCM cycle in perfect synchrony. The *fully synchronous* mode (FSYNCH), where all the robots are activated at each round, is the most studied SSYNCH sub-mode; instead, little attention was given to the *round-robin* mode (RROBIN), where all robots are activated one by one before being reactivated with the same sequential order [15].

Most of the literature has considered *fault-free* (aka *correct*) robots: robots have infinite precision in perception and computation, they correctly compute the target destination, they always reach the computed destination, and they never crash. However, such strong assumptions are unrealistic: therefore, literature has started considering *fault-prone* systems. Various types of faults have been considered in the LCM model [15]: robots may suffer from imprecise perceptions [9], inaccurate or non-rigid movements [2, 24], byzantine behaviors [1, 13, 14, 22], or crash faults after which they stop working forever [6, 13, 26]. For this purpose, *fault-tolerant* algorithms have been designed to solve problems under fault-prone systems.

In this work, we focus on systems prone to crash faults, i.e., where robots may stop working irremediably and thus do not execute their LCM cycle when activated. Let us go back to the introduction example: in the case of a firefighter swarm, a lower number of functioning robots than the scheduled number may be insufficient to fight a possible fire. Due to the criticism of the scenario, all the robots should be ready to intervene, thus it is important to design trusty systems where the not-crashed (aka, correct) robots can promptly and autonomously detect whether there exists a crashed robot and, in this case, send an external message to request human intervention and maintenance. This illustrative scenario exemplifies the importance of a distributed system to be able to autonomously detect and/or identify the presence of faulty entities within the system. Different approaches can be theorized to be applied when a fault is correctly detected:

- robots stop working to avoid compromising the integrity of the system (*swarm freezing*);
- robots try to solve the original task (*fault tolerance*) or a weaker version of it (*best effort*);
- robots solve a different problem, e.g., try to fix faulty robots (*fault recovery*).

Note that freezing the swarm may be a prior step to fault recovery. Preliminary to these approaches, robots may have to solve the Fault Detection (FD) and Fault Identification (FI) problems. A swarm solves Fault Detection if the correct robots detect whether there exists at least a faulty robot in the swarm; Fault Identification requires the correct robots to identify which robots are correct and which are faulty. Here, we propose an initial study for the FD and FI problems under the LCM model, paving the way for further work that may study the consequent reactions to recover the systems from faults.

Related work and our contribution. The research in faulty-prone robot swarms has focused on providing fault-tolerant algorithms for the classic problems [4, 5, 6, 12, 13, 15, 25, 26, 28, 30]. Yet, a fault-tolerant algorithm does not necessarily contain a fault detection module, hence it does not assume that the correct robots are aware of the presence of the faulty ones.

The concept of *failure detector* was first introduced by Chandra and Toueg in [8] for solving `Consensus` in asynchronous systems prone to crash faults. Although the authors considered a different distributed model than the LCM one (i.e., static system of n processes, each one equipped with a specific algorithm, and completely connected through trusty message channels), their work defines some properties (e.g., strong/weak completeness and accuracy) that a failure detector should reasonably enjoy, independently of the distributed system to which is applied. For the LCM model, the idea of failure detector has been considered in [28, 30], where the authors provide some crash-fault detector modules as part of solutions for the `Flocking` problem[1]. Thus, their failure detectors are strictly based on the algorithms provided for `Flocking`. Their idea is to make correct robots compare the positions of the others, and thus understand when a robot has crashed since it has not changed position for too many activations. For this comparison, robots are allowed to store a variable $S_{PosPrevObser}$ which contains the set of positions of robots in the system during the last previous observation, thus granting each robot an infinite persistent memory[2].

In this paper, we embark on a general study of how a swarm of robots can detect whether there are faulty robots in the swarm, and, possibly, identify them, independently of the primary problem the swarm is solving. Note that we aim to detect/identify faults even when the swarm has reached a static configuration since it has terminated its task or because the swarm must stay still in a particular configuration (e.g., a regular polygon and a point). One can criticize the sense of detecting faults in a swarm where apparently robots have to do nothing. The motivation lies in the fact that robots may be required to stay in a particular configuration ready to be activated by an event scheduler simulating an external event occurring in the environment (e.g., a fire for the firefighter swarm). Always being correct is a key requirement for trustworthy and fault-aware systems that can aptly detect and react to a failure.

Preparatory to our investigation, the first part of this work formalizes the occurrence of crash faults and defines the Fault Detection (FD) and Fault Identification (FI) problems. As in [8], we define two properties (i.e., reliability and punctuality) that a FD or FI algorithm should enjoy; yet, differently from [8], we work only on algorithms that never provide false positives, and we analyze the delays within which a fault is detected/identified. We take into account 12 robot models ($\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, $\mathcal{LUMI}$, combined with the modes `FSYNCH`, `RROBIN`, `SSYNCH`), and we study generic solutions for FD or FI under such models. Note that, differently from [8, 28, 30], we consider models where robots can store and communicate at most a constant-size amount of data, thus they can neither use their persistent memories to track the past snapshots of the swarm [28, 30], nor send the list of their suspects to the other robots [8]. We prove a reliable FI procedure cannot exist under $\mathcal{LUMI}^{\texttt{FSYNCH}}$. Instead, we provide three FD procedures for three models: $\mathcal{LUMI}^{\texttt{FSYNCH}}$ (punctual), $\mathcal{FCOM}^{\texttt{FSYNCH}}$ (punctual), and $\mathcal{LUMI}^{\texttt{RROBIN}}$ (reliable). We prove the impossibility of designing a reliable or punctual FD procedure under the other 9 models. Due to a lack of space, the proofs of the majority of our results have been moved to Section A. To the best of our knowledge, this is the first work addressing fault detection for the LCM model using a general approach.

2 Preliminaries

2.1 Models

This work investigates how FD and FI can be solved under different robot models. All the models we consider present some *core features*, while they differ in a set of *variable features*.

[1] The `Flocking` problem requires robots to move together on the plane while maintaining a pattern.
[2] If the swarm is composed of n robots, $S_{PosPrevObser} \in (\mathbb{R}^2)^n$.

Core features. A swarm of robots is modeled as a set of autonomous mobile robots $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$ acting in the Euclidean plane $\mathbb{R}^2$. Robots are *indistinguishable* (they cannot be distinguished by external appearance), *anonymous* (they are not provided with any id), *homogeneous* (they execute the same algorithm), and *punctiform* entities. Each robot has a local coordinate system and a sensor through which it perceives all the robots in the plane. Each robot r has a light $\texttt{lig}_r$ whose color is chosen from a constant-size palette $\mathfrak{L}$. We assume that $\mathfrak{L}$ contains at least the *blank color* $\not{b}$, the default one. All the robots in the swarm are provided with the same deterministic algorithm, which is executed every time a robot is activated. The algorithm also defines the colors in $\mathfrak{L}$ and a total order of $\mathfrak{L}$; such colors are used to update the light of each robot. By default, a robot is *idle*. When activated by the *activation scheduler*, a robot r executes a *Look-Compute-Move* cycle: it takes the *snapshot* of the whole swarm (*Look*), it executes the algorithm using the sole snapshot as input (*Compute*), and it can update the color of $\texttt{lig}_r$ and subsequently it travels along a straight trajectory towards the computed destination (*Move*). After that, it returns idle, so that it can be activated by the scheduler again. The snapshot of r contains the positions (according to the local coordinate system of r) and colors of all the robots of the swarm: no other information about the swarm can be perceived (e.g., whether robots are idle/active, still/moving). We assume *rigid* movements, i.e., each robot reaches its computed destination during its *Move* step. The local coordinate systems of the robots in the swarm may not be the same (*disoriented* robots); moreover, the local coordinate system of any robot may not persist from one activation to another (*variable disorientation*). Robots are allowed to form *multiplicities*, i.e., to occupy the same point of the plane at the same time. However, we assume *strong multiplicity detection*, so that each robot can see all the robots (and their colors) which form a multiplicity.

Variable features. We now present the features under study: *memory*, *communication*, and *synchronization*. Regarding robots' *memory and communication* capabilities, we consider the four models mainly proposed in the literature. Formally, such differences are implemented in the visibility of the robot lights which can be used as a persistent memory and/or a communication means. In the $\mathcal{OBLOT}$ model (w/o memory, w/o communication), the robots' lights are visible to no robot in the swarm (equivalently, $\mathfrak{L} = \{\not{b}\}$ for any algorithm). In the $\mathcal{FSTA}$ model (with memory, w/o communication), $\texttt{lig}_r$ is visible only to r, for each $r \in \mathcal{R}$. In the $\mathcal{FCOM}$ model (w/o memory, with communication), $\texttt{lig}_r$ is visible to all robots but r. In the $\mathcal{LUMI}$ model (with memory, with communication), each $\texttt{lig}_r$ is visible to each robot in $\mathcal{R}$. Regarding the *synchronization* of robots, we consider only *synchronous* modes, so that time is divided into atomic rounds; a subset of robots is activated at each round, and they perform their LCM cycle in perfect synchrony. In particular, in the *semi-synchronous* mode (SSYNCH) an arbitrary non-empty subset of robots is activated at each round. We always assume the *fairness condition*: for each time t and for each robot r, there exists a finite time $t' > t$ when r is activated. The fairness condition allows us to measure time as a sequence of consecutive and finite *epochs*: an epoch is the minimal time frame within which all robots are activated at least once. We consider two sub-modes of SSYNCH. In the *fully synchronous* mode (FSYNCH), the whole swarm is activated at each round. In the *round-robin* mode (RROBIN), all robots are activated in sequence, one robot in each round, before being reactivated in the same order. An epoch always lasts one round for FSYNCH and n rounds for RROBIN. The choice of the subset of robots activated at every time is made by an *activation scheduler*.

▶ **Definition 1** (Activation Scheduler). *Given a swarm of robots $\mathcal{R}$, an* activation scheduler *is a function $\mathfrak{A} : \mathbb{N} \to 2^{\mathcal{R}}$ defining the subset of the swarm $\mathfrak{A}(t) \subseteq \mathcal{R}$ activated at time*[3] *$t \geq 0$.*

Notation. We use the notation X^Y to indicate a model for robots that possess all the above core features and that has $X \in \{\mathcal{OBLOT}, \mathcal{FSTA}, \mathcal{FCOM}, \mathcal{LUMI}\}$ as communication-memory setting and $Y \in \{\mathtt{S}, \mathtt{F}, \mathtt{RR}\}$ as synchronization mode (i.e., SSYNCH, FSYNCH, RROBIN, resp.). We indicate with $\mathcal{M}$ the set of the resulting 12 robot models. For a model $X^Y \in \mathcal{M}$, we will consider both the *fault-free* version (i.e., where all robots are assumed to be always correct) and the *fault-prone* version (i.e., where robots may suffer from crash faults).

Configurations and snapshots. Given a global coordinate system $\hat{\Xi}$ on $\mathbb{R}^2$ (unknown by the swarm), we define the *global state* of a robot r_i at time $t \in \mathbb{N}$ as the pair $\hat{\zeta}_i = (\hat{\underline{x}}_i, \hat{c}_i)$ where $\hat{\underline{x}}_i \in \mathbb{R}^2$ is the position of r_i according to $\hat{\Xi}$, and $\hat{c}_i \in \mathfrak{L}$ is the light color of r_i, at time t. If r_i performs an LCM cycle at time t and r_i changes its global state in $(\hat{\underline{x}}'_i, \hat{c}'_i)$ in the related Move step, we assume that such change becomes effective at time $t+1$. If a group of $\# > 0$ robots have the same state $\hat{\zeta}$ at time t, we say they form a (global) *color-multiplicity*, denoted as $\{\hat{\zeta}\}_{\#}$. A *configuration* of the swarm at time t is the set $C = \{\{\hat{\zeta}_0\}_{\#_0}, \dots, \{\hat{\zeta}_k\}_{\#_k}\}$ defining all the distinct color-multiplicities that exist at time t (thus, $\sum_{j=0}^{k} \#_j = n$).

From a robot's point of view, the configuration of the swarm can be perceived differently: as a matter of fact, each robot r has its own local (and not persistent) coordinate system, and the color of the robots' lights may be not visible to r. By convention, we always assume that when a light is not visible to a robot r, then r sees such light as $\not{b}$-colored[4]. Let Ξ be the local coordinate system of r at time t. A local color-multiplicity $\{\zeta\}_{\#}$ is a set of $\# > 0$ robots with the same *local state* $\zeta = (\underline{x}, c)$, where $\underline{x}$ is their position according to Ξ, while c is their light color *seen by* r. A *snapshot* can be described as a "local configuration" from the point of view of r. Thus, the snapshot of r at time t is the tuple $\langle \{\zeta_0\}_{\#_0}, \dots, \{\zeta_h\}_{\#_h} \rangle$ containing all the distinct local color-multiplicities seen by r. We always assume that the first color-multiplicity $\{\zeta_0\}_{\#_0}$ of a snapshot represents the local color-multiplicity containing the robot r itself. Note that a snapshot is represented and managed as an ordered data structure: such an order allows an activated robot to distinguish its state ζ_0 from the others $\zeta_{i>0}$ (which are ordered according to the local coordinate system of r and the total order of the colors in $\mathfrak{L}$).

2.2 Crash fault schedulers

A crash fault on a robot r at time t implies that r will no longer execute an LCM cycle from time t onward. In other words, r remains idle for any time $t' \geq t$. Until t, r is said to be *correct*. According to this, we formalize the occurrence of crash faults for a swarm of robots through a scheduler of crashes.

▶ **Definition 2** (Crash Scheduler). *A* crash scheduler *for a swarm $\mathcal{R}$ is a function $\mathfrak{c} : \mathcal{R} \to \mathbb{N} \cup \{\infty\}$ defining for each robot r the time where r crashes (i.e., stops working). If $\mathfrak{c}(r) = \infty$, r never crashes.*

We say that a crash scheduler $\mathfrak{c}$ is *k-survival* if $|\{r \in \mathcal{R} \mid \mathfrak{c}(r) = \infty\}| \geq k$, i.e., it keeps at least k robots always correct. We always assume that a crash scheduler is at least 1-survival.

[3] The time domain we consider is $\mathbb{N}$, including 0.
[4] Note that r may not see even its own light.

▶ **Definition 3** (Suppression Scheduler). *Given a crash scheduler $\mathfrak{c}$ defined for a swarm $\mathcal{R}$, we define the related* suppression scheduler *as the function $\mathfrak{C}_{\mathfrak{c}} : \mathbb{N} \to 2^{\mathcal{R}}$ so that*

$$\mathfrak{C}_{\mathfrak{c}}(t) := \{r \in \mathcal{R} \mid \mathfrak{c}(r) \leq t\}.$$

For a given time $t \in \mathbb{N}$, $\mathfrak{C}_{\mathfrak{c}}(t)$ is the set of robots crashed at a time $t' \leq t$ according to $\mathfrak{c}$. Obviously, it holds that $\mathfrak{C}_{\mathfrak{c}}(t') \subseteq \mathfrak{C}_{\mathfrak{c}}(t)$. When no ambiguity arises, we will use $\mathfrak{C}$ instead of $\mathfrak{C}_{\mathfrak{c}}$ by omitting to indicate the related crash scheduler.

A swarm is governed by both an activation scheduler and a suppression scheduler, which operate simultaneously, each independently of the other. Hence, it is convenient to define the behavior of the *combined scheduler* which determines, for each time $t \in \mathbb{N}$, the subset of robots that will *actually* become active and start performing their LCM cycle at time t.

▶ **Definition 4** (Combined Scheduler). *Given $\mathfrak{A}, \mathfrak{C}$ respectively an activation and a suppression scheduler defined on the same swarm $\mathcal{R}$, the* combined scheduler *is the function $\mathfrak{A} \setminus \mathfrak{C} : \mathbb{N} \to 2^{\mathcal{R}}$ defined as*

$$\mathfrak{A} \setminus \mathfrak{C}(t) := \mathfrak{A}(t) \setminus \mathfrak{C}(t).$$

Note that, despite $\mathfrak{A}$ is always assumed fair, a combined scheduler $\mathfrak{A} \setminus \mathfrak{C}$ guarantees fairness iff $\mathfrak{C}(t) = \varnothing$ for each time $t \in \mathbb{N}$.

3 Fault Detection and Fault Identification

3.1 Problems and procedures

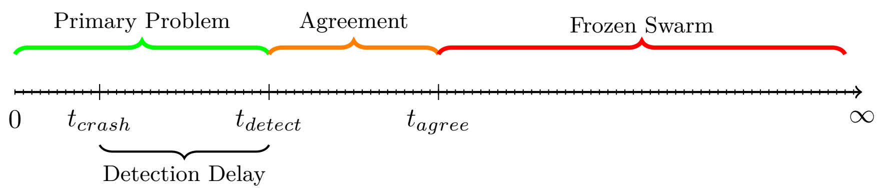
Let $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$ be a swarm of robots executing a given distributed algorithm $\mathbb{A}$ for solving a problem P under a fault-free model M. If we consider the same swarm and the same problem under the fault-prone version of M, we are interested in making robots concurrently cope with two problems: the *primary problem* P, and the Fault Detection problem. The Fault Detection problem (FD) is an agreement-related problem asking for a swarm of robots to reach awareness about the presence of a faulty robot if it exists. Basically, the swarm is required to behave in this way:

- if the first crash fault has occurred at time t_{crash}, there must exist a finite time t_{detect} when at least a robot detects there exists a fault and stays still. Afterward, all the other (correct) robots must reach the awareness of the presence of a fault and stay still;
- until t_{detect}, the robots have to continue solving the primary problem by executing $\mathbb{A}$.

As for the other agreement-related problems typical of distributed systems (e.g., `Consensus`, `Broadcast`, `Leader Election`), FD aims to make all robots in the swarm agree on the same information (i.e., the absence or the presence of a faulty robot) and the same reaction (i.e., solve the primary problem or freeze the swarm, resp.). The *awareness* of a crash can be formally implemented in each robot by setting a local boolean variable z whose value is updated at each LCM cycle by running a Fault Detection procedure during the Compute step.

Fault Detection procedure
Input: A snapshot σ of the swarm.
Output: $z \leftarrow$ FAULT if a faulty robot is detected in σ; $z \leftarrow$ CORRECT otherwise.

Formally, if a crash fault has occurred for the first time at t_{crash}, then t_{detect} is the first time when at least one robot detects the fault by executing a FD procedure, sets $z \leftarrow$ FAULT, and stays still. Accordingly, a finite time $t_{agree} \geq t_{detect}$ exists where all the correct robots of the swarm agree on the presence of the fault by setting their variable z to FAULT and thus stay still. As we will prove in this paper, there could be an intrinsic delay between t_{crash} and t_{detect} when the first robot detects that fault and/or t_{agree} when the swarm reaches the agreement on that fault (thus solving FD).



The Fault Identification problem (FI) is more sophisticated than FD since it requires robots to distinguish the faulty robots from the correct ones in the swarm. Formally, if a robot $r_i \in \mathcal{R}$ has crashed at time t_{crash_i}, then FI requires that there exists a finite time t_{detect_i} when at least a robot identifies r_i as faulty. Afterward, as for FD, all the correct robots are required to converge on the information "r_i is faulty" by a finite time $t_{agree_i} \geq t_{detect_i}$.

To solve FI, each robot queries a Fault Identification procedure, which returns a vector $\underline{z}$ so that $z_j = \mathsf{FAULT}$ ($z_j = \mathsf{CORRECT}$, resp.) if the corresponding j-th robot in the snapshot σ has been identified as faulty (correct, resp.). Indeed, a FI procedure simulates a FD one. For the sake of clarity, we will use the function-like notation $\underline{z}(r_i)$ to indicate the value in $\underline{z}$ which refers to the robot r_i: this allows us to ignore the local index of r_i according to the snapshot σ.

Fault Identification procedure
Input: A snapshot σ of the swarm $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$.
Output: $\underline{z} \in \{\mathsf{FAULT}, \mathsf{CORRECT}\}^n$, where $\underline{z}(r_i) = \mathsf{FAULT}$ iff r_i is identified as faulty in σ.

3.2 Reliability and puntuality

We here define and analyze some properties about FD/FI procedures. Let $\mathcal{F}$ be a FD procedure for $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$. We define $\Delta = t_{crash} - t_{detect}$ as the *detection delay*, where t_{crash} is the time of the first crash in $\mathcal{R}$, and t_{detect} is the first time when $\mathcal{F}$ returns FAULT. If $\mathcal{F}$ is a FI procedure, we define $\Delta_i = \mathfrak{c}(r_i) - t_i$ as the *identification delay* for a robot r_i, where $\mathfrak{c}(r_i) \neq \infty$ is the crash time of r_i and t_i is the first time $\mathcal{F}$ returns $\underline{z}(r_i) = \mathsf{FAULT}$. We say that Δ and Δ_i are undefined if the related faults have never occurred.

We firstly state that, if $\mathcal{F}$ is a FD (FI, resp.) procedure that never outputs a false positive, then a fault in the swarm (of r_i, resp.) cannot be detected (identified, resp.) in the same round of its occurrence. This means that $\Delta > 0$ ($\Delta_i > 0$, resp.). We prove the statement for FD; the proof for FI is similar.

▶ **Lemma 5** (No Instantaneous Detection). *Let $\mathcal{F}$ be a FD procedure which never outputs false positives. Then, if t_{crash} is the time of the first crash, then $\mathcal{F}$ returns* $\mathsf{CORRECT}$ *at time t_{crash}.*

We say that a FD/FI procedure guarantees *reliability* if:
- (i) (*no false positive*) it never detects/identifies a fault when it does not exist in the swarm (for FD) or in a particular robot (for FI);
- (ii) (*finite delay*) it detects/identifies a fault with a finite delay after its occurrence;
- (iii) (*coherence*) once detected/identified, it continues to detect/identify the fault in the following rounds.

Formally, let $\mathcal{F}$ be a FD procedure for a swarm $\mathcal{R} = \{r_0, \dots, r_{n-1}\}$. We say that $\mathcal{F}$ is *reliable* when, given any crash scheduler $\mathfrak{c}$ for $\mathcal{R}$, the following conditions hold:

- (i) if $\mathcal{F}$ returns FAULT at time t then $\mathfrak{C}_\mathfrak{c}(t-1) \neq \varnothing$;
- (ii,iii) if $t_{crash} = \min\{\mathfrak{c}(r_i) \mid r_i \in \mathcal{R}\} \neq \infty$ then $\exists\, t_{detect} > t_{crash} \mid t_{detect} \in \mathbb{N}$ and $\mathcal{F}$ returns FAULT for any time $t' \geq t_{detect}$.

At the same time, if $\mathcal{F}$ is a FI procedure, we say that it is *reliable* if we have:

- (i) if $\mathcal{F}$ returns $\underline{z}(r_i) =$ FAULT with $i \in \{0, \dots, n-1\}$ at time $t \in \mathbb{N}$ then $\mathfrak{c}(r_i) < t$;
- (ii,iii) if $\mathfrak{c}(r_i) = t_{crash} \neq \infty$ then $\exists\, t_i > t_{crash} \mid t_i \in \mathbb{N}$ so that $\mathcal{F}$ returns $\underline{z}(r_i) =$ FAULT at any time $t' \geq t_i$.

We say that a FD/FI procedure $\mathcal{F}$ is *punctual* if it is reliable and it holds the stronger condition that:

- if $\mathfrak{c}(r_i) = t_{crash} \neq \infty$ then $\mathcal{F}$ returns FAULT (for FD) or $\underline{z}(r_i) =$ FAULT (for FI) for any time $t' \geq t_{crash} + 1$.

In other words, $\mathcal{F}$ is punctual when it detects/identifies the fault starting from the subsequent round after a crash occurrence; thus, it never outputs false negatives (starting from $t_{crash}+1$).

3.3 Time conditions

In the previous section, we proved that a fault cannot be detected/identified at the same round when it occurs. Here, we provide further time conditions on when a fault can or cannot be detected/identified.

▶ **Definition 6** (Missing Activation). *Let $\mathfrak{A}, \mathfrak{C}$ be an activation and a suppression scheduler for a swarm $\mathcal{R}$, respectively. We say that there is a* missing activation *of $r \in \mathcal{R}$ at time $t \in \mathbb{N}$ if $r \in \mathfrak{A}(t) \cap \mathfrak{C}(t)$. We denote with $\mathfrak{fma}(r)$ the time of the* first missing activation *of r.*

▶ **Definition 7** (Fault Ignorance Time, FIT). *Let $\mathfrak{A}, \mathfrak{C}_\mathfrak{c}$ be an activation and a suppression scheduler for a swarm $\mathcal{R}$, respectively. Let $r \in \mathcal{R}$ be a robot whose crash time is $\mathfrak{c}(r) \neq \infty$ and $\mathfrak{fma}(r)$ is its first missing activation time, with $\mathfrak{fma}(r) \geq \mathfrak{c}(r)$. We define the* Fault Ignorance Time *(FIT) of r as the interval $\mathcal{I}(r) = [\mathfrak{c}(r), \mathfrak{fma}(r)] \cap \mathbb{N}$.*

The following lemma holds since, during $\mathcal{I}(r)$, r is faulty but has not yet missed an activation according to $\mathfrak{A}$, thus its behavior would have been the same as if it had been correct. The related corollary imposes a lower bound on the time for r to be identified as faulty, without guaranteeing the actual possibility of identifying such a fault.

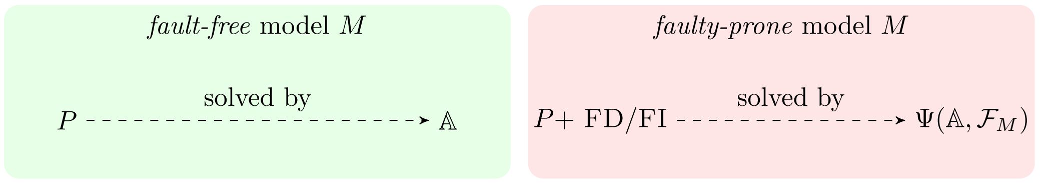
▶ **Lemma 8** (Missing Activation Rule). *Let $r \in \mathcal{R}$ be a crashed robot. Then, the crash of r cannot be identified during $\mathcal{I}(r)$ by a reliable FI procedure.*

▶ **Corollary 9.** *The crash of a robot r cannot reliably be identified at time $t \leq \mathfrak{fma}(r)$.*

The Missing Activation Rule can be generalized for an FD procedure.

▶ **Lemma 10.** *Let t_{detect} be the first time when a reliable FD procedure detects a fault in a swarm $\mathcal{R}$. Then, $t_{detect} > \min_{r \in \mathcal{R}}\{\mathfrak{fma}(r)\}$.*

3.4 Combined Algorithm


fault-free model M
$P \xrightarrow{\text{solved by}} \mathbb{A}$
faulty-prone model M
$P +$ FD/FI $\xrightarrow{\text{solved by}} \Psi(\mathbb{A}, \mathcal{F}_M)$

Besides the reliability feature, we aim to design FD or FI procedures that are the most general as possible, i.e., which can detect/identify faults independently of the primary problem that a swarm must solve. Let M be a faulty-prone model: we say that $\mathcal{F}_M$ is a *model-dependent* FD (FI, resp.) procedure if it reliably solves FD (FI, resp.) under M for *any* swarm and *any* primary problem the swarm can solve. Thus, let $\mathbb{A}$ be a solving algorithm for a (primary) problem P under the fault-free version of M. We want to combine $\mathbb{A}$ with $\mathcal{F}_M$ so that the *combined algorithm* $\Psi(\mathbb{A}, \mathcal{F}_M)$ becomes the new algorithm executed by each robot in its Compute step. The combined algorithm $\Psi(\mathbb{A}, \mathcal{F}_M)$ must behave in this way:

- *Correctness*: $\Psi(\mathbb{A}, \mathcal{F}_M)$ must continue solving P until no crash is detected/identified. If no crash occurs, $\Psi(\mathbb{A}, \mathcal{F}_M)$ eventually solves P.
- *Soundness*: $\Psi(\mathbb{A}, \mathcal{F}_M)$ must *freeze* the swarm (i.e., no robot will move anymore) when a crash is detected/identified. If a crash occurs, $\Psi(\mathbb{A}, \mathcal{F}_M)$ detects/identifies it after a finite delay after its occurrence, and will continue to detect/identify it in the next rounds.

Augmented snapshots. To solve FD/FI, $\mathcal{F}_M$ may adopt an *ad hoc* palette of colors $\mathfrak{L}_{\mathcal{F}}$ during the evolution of $\Psi(\mathbb{A}, \mathcal{F}_M)$. For the sake of clarity and w.l.o.g., we can assume that, under the fault-prone model M, each robot r is embedded with two lights, the *primary light* $\texttt{lig}_r$ which assumes the colors in $\mathfrak{L}$ provided by the primary algorithm $\mathbb{A}$, and the *service light* $\texttt{ser}_r$ which assumes the colors in $\mathfrak{L}_{\mathcal{F}}$ provided by $\mathcal{F}_M$. So, the global state of a robot r is the triple $\hat{\zeta} = (\hat{\underline{x}}, \hat{c}, \hat{f})$ where $\hat{\underline{x}}$ is its position, $\hat{c} \in \mathfrak{L}$ is the color of $\texttt{lig}_r$ and $\hat{f} \in \mathfrak{L}_{\mathcal{F}}$ is the color of $\texttt{ser}_r$. For the sake of uniformity, we assume $\mathfrak{L}_{\mathcal{F}} = \{\not{b}\}$ under $\mathcal{OBLOT}$; moreover, when the light of r is not visible to a robot, then this robot will see $\texttt{ser}_r$ (as well as $\texttt{lig}_r$) as $\not{b}$-colored. Thus, a combined algorithm $\Psi(\mathbb{A}, \mathcal{F}_M)$ takes in input an augmented snapshot $\sigma = \langle \{\zeta_0\}_{\#_0}, \ldots, \{\zeta_h\}_{\#_h} \rangle$ with $\zeta_i = (\underline{x}_i, c_i, f_i)$, and returns a triple $(\underline{x}', c', f')$ where $\underline{x}'$ is the position to be reached, and c' (f', resp.) is the possible color of the primary (service, resp.) light to be set during the related Move step[5]. Remind that a robot may not update the color of its lights; in that case, the value c' (f', resp.) returned by the algorithm will be denoted with the symbol $-$.

Algorithm 1 Combined algorithm $\Psi(\mathbb{A}, \mathcal{F}_M)$ where $\mathcal{F}_M$ is a model-dependent FI procedure for a model M.

```
Input: $\sigma = \langle \{\zeta_0\}_{\#_0}, \ldots, \{\zeta_h\}_{\#_h} \rangle$ snapshot taken by $r$; $\zeta_0 = (\underline{x}_0, c_0, f_0)$ is the state of $r$;
Output: $(\underline{x}', c', f')$ new state of $r$;

  procedure $\Psi(\mathbb{A}, \mathcal{F}_M)(\sigma)$
      $(\underline{z}, f') \leftarrow \mathcal{F}_M(\sigma)$;
      if $\forall i \in \{1, \ldots, n-1\}$ $z_i$ = CORRECT then
          $\sigma_{\mathbb{A}} \leftarrow$ restricted version of $\sigma$ by removing all $f_i$ components;
          $(\underline{x}', c') \leftarrow \mathbb{A}(\sigma_{\mathbb{A}})$;            ▷ Continue solving the primary problem
      else
          $(\underline{x}', c') \leftarrow (\underline{x}_0, -)$;            ▷ Awareness about the faulty robots + Freezing
      end if
      return $(\underline{x}', c', f')$;
  end procedure
```

[5] Remind that hatted and unhatted notations are used to distinguish the global states of robots from the local views of them.

Let us present our combined algorithm Ψ, which uses $\mathbb{A}$ and $\mathcal{F}_M$ as black-box procedures for its implementation. The pseudo-code of Ψ is listed in Algorithm 1. We here assume that $\mathcal{F}_M$ is a FI procedure (if it were a FD procedure, the implementation would be similar). Let r be an activated robot in a swarm $\mathcal{R}$ of n robots that executes the algorithm Ψ using the just taken snapshot σ as input. At first, Ψ runs the FI procedure $\mathcal{F}_M(\sigma)$, which outputs the vector $\underline{z} \in \{\mathsf{CORRECT}, \mathsf{FAULT}\}^n$, and the new color $f' \in \mathfrak{L}_{\mathcal{F}}$ to be set for the service light of r. Then Ψ implements two different actions according to the value of $\underline{z}$:

- *(correctess)*: If the vector $\underline{z}$ contains only $\mathsf{CORRECT}$ values, Ψ computes the restricted snapshot $\sigma_{\mathbb{A}}$ that is obtained by σ by removing the service color f_i from any state ζ_i. Then, Ψ runs the primary algorithm $\mathbb{A}(\sigma_{\mathbb{A}})$ and obtains the pair $(\underline{x}', c')$, i.e., the possibly new position and primary color for r: this result perfectly simulates the behavior of $\mathbb{A}$ in the solution of the primary problem under the fault-free version of M.
- *(soundness)*: Otherwise, r is aware that some robots are faulty in the swarm, and it can identify them. In this case, Ψ returns the triple $(\underline{x}_0, -, f')$ where $\underline{x}_0$ is the current position r (i.e., r freezes itself).

This algorithm generalizes the case when $\mathcal{F}_M$ is a FD procedure: in that case, $\underline{z}$ consists of only one element.

The next theorem allows the use of $\mathcal{F}_M$ or $\Psi(\mathbb{A}, \mathcal{F}_M)$ interchangeably in the following sections.

▶ **Theorem 11.** *$\mathcal{F}_M$ is a reliable model-dependent FD or FI procedure for a model $M \in \mathcal{M}$ if and only if $\Psi(\mathbb{A}, \mathcal{F}_M)$ guarantees correctness and soundness for any primary algorithm $\mathbb{A}$ under M.*

3.5 Fault Identification unreliability

Here, we present a witness problem through which we prove the impossibility of designing a reliable model-dependent FI procedure for $\mathcal{LUMI}^{\mathsf{F}}$.

▶ **Problem 1** (`SuperPoint`)**.** Consider n robots lying on the same point p of the Euclidean plane. The problem asks them never to change position[6].

The `SuperPoint` problem can be trivially solved under any fault-free model M thanks to a NOP algorithm $\mathbb{A}_{NOP}$ (basically, robots do nothing)[7]. Let $\hat{\Xi}$ be a coordinate system whose origin lies in p. Let $\mathcal{F}$ be a hypothetical reliable $\mathcal{LUMI}^{\mathsf{F}}$-dependent FI procedure. In the following, we state some properties about $\Psi(\mathbb{A}_{NOP}, \mathcal{F})$ while solving `SuperPoint`, assuming the local coordinate system of all the robots is always $\hat{\Xi}$.

▶ **Lemma 12.** *Let $\mathcal{R}$ be a swarm executing $\Psi(\mathbb{A}_{NOP}, \mathcal{F})$ to solve `SuperPoint` as primary problem, where $\mathcal{F}$ is a reliable model-dependent FI procedure for $\mathcal{LUMI}^{\mathsf{F}}$. Let us assume all the robots always use the coordinate system $\hat{\Xi}$, and let $\mathfrak{C}$ be a suppression scheduler for $\mathcal{R}$. Then at each round t, all the robots in $\mathcal{R} \setminus \mathfrak{C}(t-1)$ start their round with the same color.*

Proof. Let $\mathfrak{L}$ be the palette used by $\mathbb{A}_{NOP}$ to solve `SuperPoint` under the fault-free $\mathcal{LUMI}^{\mathsf{F}}$ model[8]. Suppose $\mathfrak{L}_{\mathcal{F}}$ is the palette used by $\mathcal{F}$. By hypothesis, the snapshot of a robot r has this form: $\sigma = \langle \{(\mathbf{O}, b_0)\}_{\#_0}, \ldots, \{(\mathbf{O}, b_h)\}_{\#_h} \rangle$ where $\mathbf{O} = (0,0)$ is the position of any

[6] The motivation of this kind of problem can be found in the Introduction.
[7] Reference to the assembly instruction `NOP`, i.e., No Operation.
[8] Although the most efficient NOP algorithm can solve the problem with $\mathfrak{L} = \{\not{b}\}$, we here assume nothing else about $\mathbb{A}_{NOP}$ other than that it effectively solves `SuperPoint`.

robot according to $\hat{\Xi}$, and where $b_i = (c_i, f_i) \in \mathfrak{L} \times \mathfrak{L}_{\mathcal{F}}$ represents the pair of colors for the primary light and the service light of the robots. Note that b_0 represents the colors of the robot r itself. The lemma can be easily proved by induction: in fact, at time $t = 0$, all the robots have the same color $b_i = (\not{b}, \not{b})$ (by convention, we set that $\mathfrak{C}(-1) = \varnothing$). Let us now assume the fact true for a time t, i.e., all the robots in $\mathcal{R} \setminus \mathfrak{C}(t-1)$ have the same color, say $\mathbf{b}$, during the t-th round. Since $\mathcal{R} \setminus \mathfrak{C}(t) \subseteq \mathcal{R} \setminus \mathfrak{C}(t-1)$, all the active robots in the t-th Look step take the same snapshot $\sigma = \langle \{(\mathbf{O}, \mathbf{b})\}_{\#_0}, \ldots, \{(\mathbf{O}, b_h)\}_{\#_h} \rangle$, and accordingly they compute the same next color $\mathbf{b}'$ by executing $\Psi(\mathbb{A}_{NOP}, \mathcal{F})(\sigma)$. Thus, at the beginning of the $(t+1)$-th round, all the robots in $\mathcal{R} \setminus \mathfrak{C}(t)$ have set the same color $\mathbf{b}'$. ◀

By the contrapositive of Lemma 12, we have:

▶ **Corollary 13.** *Let $\mathcal{F}$ be a reliable FI model-dependent procedure for $\mathcal{LUMI}^{\mathrm{F}}$. Let us consider the problem* `SuperPoint` *for a swarm of robots executing $\mathcal{F}$ and let us assume all the robots always use the coordinate system $\hat{\Xi}$. If at the t-th round two robots have two different colors, then at least one belongs to $\mathfrak{C}(t-1)$.*

▶ **Theorem 14.** *There can be no reliable model-dependent FI procedure for $\mathcal{LUMI}^{\mathrm{F}}$.*

Proof. By contradiction, let us assume that there exists a reliable FI procedure $\mathcal{F}$ for $\mathcal{LUMI}^{\mathrm{F}}$ which uses a k-size palette $\mathfrak{L}_{\mathcal{F}}$. Let us consider the `SuperPoint` problem under $\mathcal{LUMI}^{\mathrm{F}}$, with $n \geq k+2$ robots, and let $\mathbb{A}_{NOP}$ be the optimal algorithm solving `SuperPoint` with $\mathfrak{L} = \{\not{b}\}$. By definition, the problem starts from a configuration where all the robots have the same color. Let us assume all the robots always use the coordinate system $\hat{\Xi}$. Thus, for the sake of conciseness, we can omit to state the position $(0,0)$ and the primary light color $\not{b}$ in the snapshots of the robots; instead, we only specify the colors of the service lights. As proved in Lemma 12, at each round t, all the active and correct robots take the same snapshot $\sigma = \langle \{f_0\}_{\#_0}, \ldots, \{f_h\}_{\#_h} \rangle$, execute $\mathcal{F}(\sigma) = (\underline{z}, f')$ and update their service color from f_0 to f'. Note in fact that f_0 corresponds to the current service color of all the robots in $\mathcal{R} \setminus \mathfrak{C}(t-1)$ (and thus in $\mathcal{R} \setminus \mathfrak{C}(t)$).

Let $\mathfrak{C}$ be a suppression scheduler so that it crashes a number of robots $\in [k, n-2]$ in k different rounds within a finite time t_k. Formally, there exists a series of k finite crash times $t_1 < \cdots < t_k$ such that $\mathfrak{C}(t_i) \subset \mathfrak{C}(t_{i+1})$, and such that $|\mathfrak{C}(t_k)| \in [k, n-2]$. Let $\hat{t}$ be the first time when the robots in $\mathcal{R} \setminus \mathfrak{C}(\hat{t})$ execute $\mathcal{F}(\sigma) = (\underline{z}, \mathbf{f})$ and $\mathbf{f}$ is a color already present in the current snapshot σ. In other words, there is at least a faulty $\mathbf{f}$-colored robot in the swarm at time $\hat{t}$. Note that this time exists and it holds $\hat{t} \leq t_k$. In fact, the worst case (i.e., $\hat{t} = t_k$) is reached when, at each crash time t_i, $\mathcal{F}$ outputs a new service color not yet present in the swarm: after the k-th crash time t_k, the swarm contains all the k colors of $\mathfrak{L}_{\mathcal{F}}$. So, at time $\hat{t}$, the correct robots set their color as $\mathbf{f}$.

Let r be an active robot at time $\hat{t}+1$ and let $\langle \{\mathbf{f}\}_{\#_0}, \ldots, \{f_h\}_{\#_h} \rangle$ be the snapshot r takes. Note that, since the correct robots at time $\hat{t}$ are at least 2 by hypothesis, then $\#_0 > 2$. By applying $\mathcal{F}$, r cannot distinguish which among the $\#_0$ $\mathbf{f}$-colored robots are faulty or correct. This contradicts the hypothesis that $\mathcal{F}$ is reliable. ◀

4 Model-dependent Fault Detection

In this section, we present three model-dependent FD procedures respectively for $\mathcal{LUMI}^{\mathrm{F}}$, $\mathcal{FCOM}^{\mathrm{F}}$ and $\mathcal{LUMI}^{\mathrm{RR}}$. In all of them, we assume that the related $\mathfrak{L}_{\mathcal{F}}$ always contains a special color `ALARM` which will be used as soon as a robot detects the presence of a faulty robot during its Compute step. Accordingly, in the related Move step, the robot will set its service light as `ALARM` to notify itself and/or the other robots that a faulty robot has been detected. When a robot r sees an `ALARM` in its snapshot, it automatically is aware that a fault has been detected in the previous rounds. Thus, r sets its $\texttt{ser}_r$ to `ALARM` too, whereas it updates neither its position nor its $\texttt{lig}_r$: this causes the alarm to be broadcast throughout the correct robots of the swarm, and the swarm to freeze at the same time.

Eventually in Section 4.4, we prove that a reliable FD procedure cannot exist under the other robot models. Table 1 summaries the properties of our proposed model-dependent FD procedures.

Table 1 Proposed model-dependent FD procedures. $X \in \{\mathcal{OBLOT}, \mathcal{FSTA}, \mathcal{FCOM}, \mathcal{LUMI}\}$, $Y \in \{\texttt{F}, \texttt{RR}, \texttt{S}\}$.

Model	Reliable	Min Survivals	Detection Delay	$\lvert\mathfrak{L}_{\mathcal{F}}\rvert$
$\mathcal{LUMI}^{\texttt{F}}$	Yes	1	$\Delta = 1$ (Punctual)	3
$\mathcal{FCOM}^{\texttt{F}}$	Yes	2	$\Delta = 1$ (Punctual)	3
$\mathcal{LUMI}^{\texttt{RR}}$	Yes	1	$1 \leq \Delta \leq 2n-1$	4
$\mathcal{FCOM}^{\texttt{F}}$	Impossible	1		
$\mathcal{FCOM}^{\texttt{RR}}$	Impossible	2		
$X^{\texttt{S}}$	Impossible			
$\{\mathcal{FSTA}, \mathcal{OBLOT}\}^{Y}$	Impossible			

4.1 Fault detection in $\mathcal{LUMI}^{\texttt{F}}$

Let us present a simple reliable FD procedure $\mathcal{F}_{\mathcal{LUMI}^{\texttt{F}}}$ which can be executed by any swarm $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$ under $\mathcal{LUMI}^{\texttt{F}}$. This procedure tolerates up to $n-1$ crash faults and uses $\mathfrak{L}_{\mathcal{F}} = \mathbb{Z}_2 \cup \{\texttt{ALARM}\}$ as the service palette, 0 being the initial value. Working under `FSYNCH`, all the correct robots become active simultaneously, and the $\mathcal{F}_{\mathcal{LUMI}^{\texttt{F}}}$ procedure makes them update their service light cycling over the values of $\mathbb{Z}_2 = \{0, 1\}$. Let $t_{crash} = \min_i\{\mathfrak{c}(r_i)\}$ be the first time when a robot crashes according to a crash time scheduler $\mathfrak{c}$ (which is 1-survival). Note that t_{crash} can be ∞. Then, all the robots synchronously alternate their service colors during the rounds before t_{crash}. If one robot crashes at time t_{crash} and so it does not update its light, its failure will be detected in the successive round, as it will display a different light than any other activated robot. Let r be an activated robot at time $t_{crash}+1$. So r detects from its snapshot that all the robots with a service light different from itself have crashed at the previous round, and thus sets $\texttt{ser}_r$ as `ALARM`. Because of this immediate detection, the proposed procedure is punctual. Note that, although r can identify the crashed robots at time $t_{crash}+1$, this procedure is not a model-dependent FI procedure since it cannot guarantee reliability in identifying the faulty robots in the following rounds (see Theorem 14 for the impossibility general result). Algorithm 2 and Figure 1a show the proposed FD procedure and the color update diagram of $\texttt{ser}_r$, respectively. Note that in Figure 1a (as well as for the next diagrams) the `ALARM` node represents a trap state, so that no outgoing transition is permitted: this property ensures the proposed procedure is coherent.

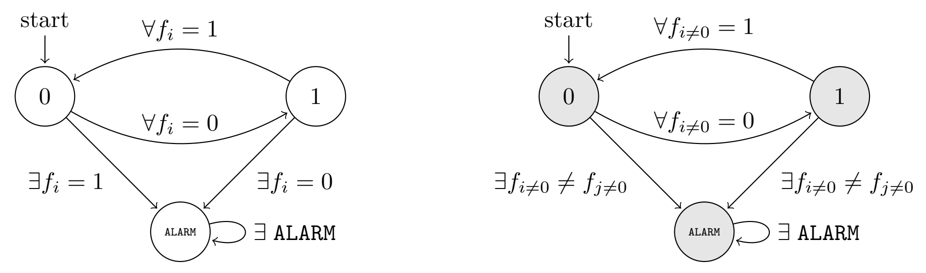
4.2 Fault detection in $\mathcal{FCOM}^{\texttt{F}}$

The fact that a robot under $\mathcal{FCOM}$ cannot see its own lights (`lig` and `ser`) imposes the first main difference in the number of tolerated crashes with respect to the previous FD procedure.

Algorithm 2 A *punctual* FD procedure for $\mathcal{LUMI}^{\mathsf{F}}$.

Input: $\sigma = \langle \{\zeta_0\}_{\#_0}, \dots, \{\zeta_h\}_{\#_h} \rangle$ snapshot taken by a robot r, with $\zeta_i = (\underline{x}_i, c_i, f_i)$.
Output: (z, y) where $z \in \{\mathsf{FAULT}, \mathsf{CORRECT}\}$ and $y \in \mathbb{Z}_2 \cup \{\texttt{ALARM}\}$.

```
procedure $\mathcal{F}_{\mathcal{LUMI}^{\mathsf{F}}}(\sigma)$
    if $\exists i \in \{0, \dots, h\} \mid f_i = \texttt{ALARM}$ then
        return $(\mathsf{FAULT}, \texttt{ALARM})$;                    ▷ Alarm broadcasting
    end if
    if $\exists i \in \{1, \dots, h\} \mid f_i \neq f_0$ then
        return $(\mathsf{FAULT}, \texttt{ALARM})$;                    ▷ First detection of a fault
    end if
    return $(\mathsf{CORRECT}, (f_0 + 1) \bmod 2)$;                   ▷ Service color alternation in $\mathbb{Z}_2$
end procedure
```


start
$\forall f_i = 1$
0
1
$\forall f_i = 0$
$\exists f_i = 1$
$\exists f_i = 0$
ALARM
$\exists$ ALARM
start
$\forall f_{i\neq 0} = 1$
0
1
$\forall f_{i\neq 0} = 0$
$\exists f_{i\neq 0} \neq f_{j\neq 0}$
$\exists f_{i\neq 0} \neq f_{j\neq 0}$
ALARM
$\exists$ ALARM

(a) Algorithm 2 for $\mathcal{LUMI}^{\mathsf{F}}$. **(b)** Algorithm 3 for $\mathcal{FCOM}^{\mathsf{F}}$.

Figure 1 Diagrams showing the color update of $\texttt{ser}_r$ according to the proposed FD procedures. The gray nodes represent the impossibility for r to see the color of its own $\texttt{ser}_r$ (i.e., f_0).

▶ **Theorem 15.** *There can be no reliable FD procedure for $\mathcal{FCOM}^{\mathsf{F}}$ assuming 1-survival crash schedulers.*

Assuming that no more than $n - 2$ robots can crash, we can aptly apply some slight but necessary changes to Algorithm 2 and provide a *punctual* FD procedure for $\mathcal{FCOM}^{\mathsf{F}}$ (see Algorithm 3 and the corresponding color diagram in Figure 1b). Notably, robots alternate their service colors in $\mathbb{Z}_2$: a robot can promptly detect a fault and set its service light to ALARM as soon as it sees two robots (different from itself) with two different service colors.

Algorithm 3 A *punctual* FD procedure for $\mathcal{FCOM}^{\mathsf{F}}$.

Input: $\sigma = \langle \{\zeta_0\}_{\#_0}, \dots, \{\zeta_h\}_{\#_h} \rangle$ snapshot taken by a robot r, with $\zeta_i = (\underline{x}_i, c_i, f_i)$.
Output: (z, y) where $z \in \{\mathsf{FAULT}, \mathsf{CORRECT}\}$ and $y \in \mathbb{Z}_2 \cup \{\texttt{ALARM}\}$.

```
procedure $\mathcal{F}_{\mathcal{FCOM}^{\mathsf{F}}}(\sigma)$
    if $\exists i \in \{1, \dots, h\} \mid f_i = \texttt{ALARM}$ then
        return $(\mathsf{FAULT}, \texttt{ALARM})$;                    ▷ Alarm broadcasting
    end if
    if $\exists i, j \in \{1, \dots, h\} \mid f_i \neq f_j$ then
        return $(\mathsf{FAULT}, \texttt{ALARM})$;                    ▷ First detection of a fault
    end if
    return $(\mathsf{CORRECT}, (f_1 + 1) \bmod 2)$;                   ▷ Service color alternation in $\mathbb{Z}_2$
end procedure
```

4.3 Fault detection in $\mathcal{LUMI}^{\text{RR}}$

Under $\mathcal{LUMI}^{\text{RR}}$, we can design a reliable FD procedure even considering 1-survival crash time schedulers. However, we prove that designing a punctual FD procedure for this model is impossible.

▶ **Theorem 16.** *There can be no punctual FD procedure for $\mathcal{LUMI}^{\text{RR}}$, even considering $(n-1)$-survival crash time schedulers.*

Algorithm 4 A *reliable* FD procedure under $\mathcal{LUMI}^{\text{RR}}$.

Input: $\sigma = \langle\{\zeta_0\}_{\#_0}, \dots, \{\zeta_h\}_{\#_h}\rangle$ snapshot taken by a robot r, with $\zeta_i = (\underline{x}_i, c_i, f_i)$.
Output: (z, y) where $z \in \{\mathsf{FAULT}, \mathsf{CORRECT}\}$ and $y \in \mathbb{Z}_3 \cup \{\mathtt{ALARM}\}$.

```
procedure $\mathcal{F}_{\mathcal{LUMI}^{\text{RR}}}(\sigma)$
    if $\exists i \in \{0, \dots, h\} \mid f_i = \mathtt{ALARM}$ then
        return $(\mathsf{FAULT}, \mathtt{ALARM})$;                ▷ Alarm broadcasting
    end if
    if $\exists i \in \{1, \dots, h\} \mid f_i \equiv (f_0 - 1) \bmod 3$ then
        return $(\mathsf{FAULT}, \mathtt{ALARM})$;                ▷ First Fault detection
    end if
    return $(\mathsf{CORRECT}, (f_0 + 1) \bmod 3)$;        ▷ Service color alternation in $\mathbb{Z}_3$
end procedure
```

We now present our reliable FD procedure $\mathcal{F}_{\mathcal{LUMI}^{\text{RR}}}$ (see Algorithm 4 and Figure 2). In this case, we need one more value in the palette $\mathfrak{L}_{\mathcal{F}}$: in particular, we make our robots cyclically take any value in $\{0, 1, 2\}$ (i.e., $\mathbb{Z}_3$), 0 being the default value. Note that it is possible to define a cyclic order $[0, 1, 2]$ on $\mathbb{Z}_3$, which would be impossible in $\mathbb{Z}_2$; this order defines our color transition rule $i \mapsto (i+1) \bmod 3$. Thus, during the i-th epoch (with $i \geq 0$), our strategy is to make each activated robot set its $\mathtt{ser}_r$ from $i \bmod 3$ to $(i+1) \bmod 3$.

Suppose a robot r crashes at its activation during the i-th epoch. So, its $\mathtt{ser}_r$ remains outdated to $i \bmod 3$. During the remainder of the i-th epoch, all the activated robots see robots whose service lights are colored as either $i \bmod 3$ or $(i+1) \bmod 3$, thus they may have no means to understand if r has crashed or has not been activated yet. So they update their service light to $(i+1) \bmod 3$. Let r' be the first robot activated during the $(i+1)$-th epoch. Now, r' can detect the fault since it sees a robot whose service light has the preceding value (according to the cyclic order $[0, 1, 2]$) than its own. So r' sets its service light to $\mathtt{ALARM}$, thus starting the agreement process among the other correct robots; in fact, in the following rounds, all the correct robots will set the $\mathtt{ALARM}$ light too, freezing the swarm.

To compute the maximum detection delay of $\mathcal{F}_{\mathcal{LUMI}^{\text{RR}}}$, consider the worst case. If $(r_0, \dots, r_{n-1})$ is the round-robin sequence by which robots are activated, suppose that all the robots except for r_{n-1} crash during the first epoch, during their activation. So, the first crash (i.e., of r_0) will be detected by r_{n-1} in the second epoch, thus $2n-1$ rounds later. Instead, the minimum delay occurs when r_{n-1} crashes (at the end of an epoch) and r_0 detects its fault in the following round (at the beginning of the subsequent epoch).

4.4 Impossibility results

Using a similar argument as in Theorem 15, we prove that a reliable FD procedure for $\mathcal{FCOM}^{\text{RR}}$ cannot exist assuming 1-survival crash schedulers. However, unlike $\mathcal{FCOM}^{\text{F}}$, such impossibility holds even with 2-survival crash schedulers.

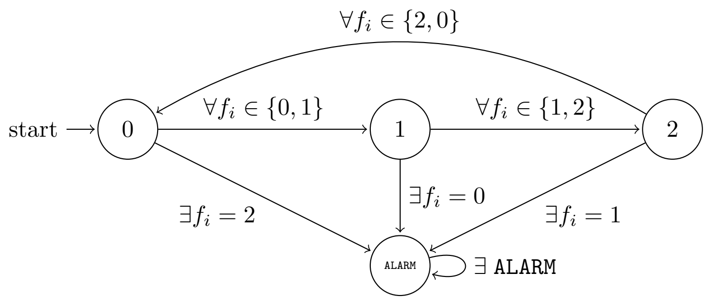


Figure 2 Color update of $\texttt{ser}_r$ in Algorithm 4.

▶ **Theorem 17.** *There can be no reliable FD procedure for $\mathcal{FCOM}^{\text{RR}}$ even assuming* 2*-survival crash schedulers.*

The following theorem states the impossibility of distinguishing a not-yet-activated robot from a failed one in the SSYNCH mode.

▶ **Theorem 18.** *There can be no reliable FD procedure for a model $M \in \mathcal{M}$ under SSYNCH.*

▶ **Problem 2** (Flip-Flop)**.** Consider two robots p, q lying on a point O and a third robot r lying on a distinct point x. Robot r must perform these two movements perpetually: reach the symmetric point x' w.r.t. O, and then come back to x.

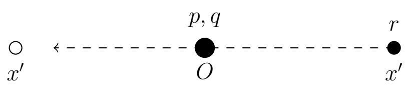


Figure 3 Flip-Flop problem.

▶ **Theorem 19.** *There can be no reliable FD procedure for a model $M \in \mathcal{M}$ under $\mathcal{FSTA}$.*

Proof. By contradiction, let $\mathcal{F}$ be a reliable FD procedure using a palette $\mathfrak{L}_{\mathcal{F}} = \{f_0, \ldots, f_{k-1}\}$ of $k \geq 1$ colors for an $\mathcal{FSTA}$ model. We show that there exists a problem for which $\mathcal{F}$ is not reliable. Consider the problem Flip-Flop that can be trivially solved under any fault-free $\mathcal{FSTA}$ model without the use of colors: let $\mathbb{A}$ be a solving algorithm with $\mathfrak{L} = \{\not{b}\}$. It is self-evident that the crash of a robot which is required to be still by the primary problem cannot be detected without assuming communication. In fact, r can never detect whether a robot in O has crashed. However, this impossibility is not limited to the static robots: we prove that p (the same holds for q) cannot reliably detect the fault of r.

Let us denote the following coordinates: $\mathbf{O} = (0,0)$, $\mathbf{1} = (1,0)$, and $-\mathbf{1} = (-1,0)$. Let Ξ^+ be a coordinate system which makes a robot perceives O in position $\mathbf{O}$ and x in position $\mathbf{1}$ (and thus x' in position $-\mathbf{1}$), and let Ξ^- be a coordinate system under which O is in position $\mathbf{O}$ and x in $-\mathbf{1}$ (and thus x' in $\mathbf{1}$). Let $\mathfrak{A}$ be an activation scheduler (any under SSYNCH, FSYNCH, RROBIN) for p, q, r. Consider the swarm under the combined scheduler $\mathfrak{A} \setminus \mathfrak{C}$ where $\mathfrak{C}$ is a suppression scheduler that crashes no robot. Suppose p is activated at times $t_0, t_1, t_2, \ldots$, and suppose that its coordinate system is Ξ^+ when r lies on x and Ξ^- when r lies on x'. In other words, p always perceives r in position $\mathbf{1}$. Since $\mathcal{F}$ is reliable and no robot has crashed, p will always obtain CORRECT by executing $\mathcal{F}$ at times $t_0, t_1, \ldots$. In particular, let $\sigma_0, \sigma_1, \ldots, \sigma_h$ be the set of the snapshots taken with some periodicity by p along its activations $t_0, t_1, \ldots$, where $0 \leq h < k$ and where σ_i has this form[9]: $\langle(\mathbf{O}, \not{b}, f_i), (\mathbf{O}, \not{b}, \not{b}), (\mathbf{1}, \not{b}, \not{b})\rangle$.

Now, consider the same swarm under the combined scheduler $\mathfrak{A} \setminus \mathfrak{C}'$ where $\mathfrak{C}'$ only crashes r at time t_0. W.l.o.g. suppose that r lies on x at t_0, and suppose that from t_0 onward p will always have Ξ^+ as coordinate system, thus perceiving r in position $\mathbf{1}$. Thus p will be activated at times $t_0, t_1, t_2, \ldots$ and it will perform $\mathcal{F}(\sigma_0), \mathcal{F}(\sigma_1), \ldots \mathcal{F}(\sigma_h)$ at the same times and periodicity as under $\mathfrak{A} \setminus \mathfrak{C}$, thus never detecting the fault of r. Contradiction achieved. ◀

▶ **Corollary 20.** *There can be no reliable FD procedure for a model $M \in \mathcal{M}$ under $\mathcal{OBLOT}$.*

5 Conclusions

In this paper, we have investigated the ability of correct robots to detect or identify the presence of crashed robots in fault-prone swarms. We have defined the problems associated with this purpose, namely FAULT DETECTION (FD) and FAULT IDENTIFICATION (FI), and we have studied the (im)possibility of solving FD or FI under 12 robot models ($\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, $\mathcal{LUMI}$, combined with `FSYNCH`, `RROBIN`, `SSYNCH`). We have formally defined two properties (reliability and punctuality) that a FD or FI procedure should satisfy, and we have soon proved that a reliable FI procedure cannot exist under the strong model $\mathcal{LUMI}^{\mathtt{F}}$. So, we have provided three reliable (or even punctual) FD procedures for the three models $\mathcal{LUMI}^{\mathtt{F}}$, $\mathcal{FCOM}^{\mathtt{F}}$ and $\mathcal{LUMI}^{\mathtt{RR}}$; for the others 9 models, we have proved that it is impossible to design a reliable FD procedure.

In this vein, future works should investigate the (im)possibility of developing reliable FD or FI procedures for models not considered in our work (e.g., operating under *k-bounded* activation schedulers), or finding weaker properties (w.r.t. reliability and punctuality) under which it is possible to solve FD or FI. Furthermore, this preliminary study about FD and FI paves the way for subsequent work studying *fault recovery* distributed algorithms, i.e., where correct robots must fix the faulty ones.

References

1 Yotam Ashkenazi, Shlomi Dolev, Sayaka Kamei, Fukuhito Ooshita, and Koichi Wada. Forgive & forget: Self-stabilizing swarms in spite of byzantine robots. In *Proc. of 7th International Symposium on Computing and Networking, CANDAR*, pages 188–194. IEEE, 2019. `doi:10.1109/CANDARW.2019.00041`.

2 Kaustav Bose, Archak Das, and Buddhadeb Sau. Pattern formation by robots with inaccurate movements. In *Proc. of 25th International Conference on Principles of Distributed Systems, OPODIS*, volume 217 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.OPODIS.2021.10`.

3 Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. Infinite grid exploration by disoriented robots. In *Proc. of 8th International Conference on Networked Systems, NETYS*, volume 12129 of *LNCS*, pages 129–145. Springer, 2020. `doi:10.1007/978-3-030-67087-0_9`.

4 Quentin Bramas, Sayaka Kamei, Anissa Lamani, and Sébastien Tixeuil. Stand-up indulgent gathering on lines. *Theoretical Computer Science*, 1016:114796, 2024. `doi:10.1016/J.TCS.2024.114796`.

[9] For the sake of conciseness, we here omit the notation for multiplicities.

5 Quentin Bramas, Anissa Lamani, and Sébastien Tixeuil. Stand up indulgent gathering. *Theoretical Computer Science*, 939:63–77, 2023. doi:10.1016/j.tcs.2022.10.015.

6 Quentin Bramas and Sébastien Tixeuil. Wait-free gathering without chirality. In Christian Scheideler, editor, *Proc. of 22nd International Colloquium On Structural Information and Communication Complexity, SIROCCO*, volume 9439 of *LNCS*, pages 313–327. Springer, 2015. doi:10.1007/978-3-319-25258-2_22.

7 Davide Canepa and Maria Gradinariu Potop-Butucaru. Stabilizing flocking via leader election in robot networks. In *Proc. of 9th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS*, volume 4838 of *LNCS*, pages 52–66. Springer, 2007. doi:10.1007/978-3-540-76627-8_7.

8 Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996. doi:10.1145/226643.226647.

9 Reuven Cohen and David Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. *SIAM J. Comput.*, 38(1):276–302, 2008. doi:10.1137/060665257.

10 Gianlorenzo D'Angelo, Gabriele Di Stefano, Ralf Klasing, and Alfredo Navarra. Gathering of robots on anonymous grids and trees without multiplicity detection. *Theoretical Computer Science*, 610:158–168, 2016. doi:10.1016/J.TCS.2014.06.045.

11 Shantanu Das, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Forming sequences of patterns with luminous robots. *IEEE Access*, 8:90577–90597, 2020. doi:10.1109/ACCESS.2020.2994052.

12 Shantanu Das, Riccardo Focardi, Flaminia L. Luccio, Euripides Markou, and Marco Squarcina. Gathering of robots in a ring with mobile faults. *Theoretical Computer Science*, 764:42–60, 2019. doi:10.1016/J.TCS.2018.05.002.

13 Xavier Défago, Maria Gradinariu, Stéphane Messika, and Philippe Raipin Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *Proc. of 20th International Symposium on Distributed Computing, DISC*, volume 4167 of *LNCS*, pages 46–60. Springer, 2006. doi:10.1007/11864219_4.

14 Xavier Défago, Maria Potop-Butucaru, and Philippe Raipin Parvédy. Self-stabilizing gathering of mobile robots under crash or byzantine faults. *Distributed Computing*, 33(5):393–421, 2020. doi:10.1007/S00446-019-00359-X.

15 Xavier Défago, Maria Potop-Butucaru, and Sébastien Tixeuil. Fault-tolerant mobile robots. In *Chapter 10 of [19]*, pages 234–251. Springer, 2019. doi:10.1007/978-3-030-11072-7_10.

16 Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *Proc. of 7th International Conference on Principles of Distributed Systems, OPODIS*, volume 3144 of *LNCS*, pages 34–46. Springer, 2003. doi:10.1007/978-3-540-27860-3_6.

17 Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. $O(\log n)$-time uniform circle formation for asynchronous opaque luminous robots. In *Proc. of 27th International Conference on Principles of Distributed Systems, OPODIS*, volume 286 of *LIPIcs*, pages 5:1–5:21, 2023. doi:10.4230/LIPICS.OPODIS.2023.5.

18 Paola Flocchini. Gathering. In *Chapter 4 of [19]*, pages 63–82. Springer, 2019. doi:10.1007/978-3-030-11072-7_4.

19 Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*. Springer, 2019. doi:10.1007/978-3-030-11072-7.

20 Taisuke Izumi, Daichi Kaino, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. On time complexity for connectivity-preserving scattering of mobile robots. *Theoretical Computer Science*, 738:42–52, 2018. doi:10.1016/J.TCS.2018.04.047.

21 Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, Sébastien Tixeuil, and Koichi Wada. Gathering on rings for myopic asynchronous robots with lights. In *Proc. of 23rd International Conference on Principles of Distributed Systems, OPODIS*, volume 153 of *LIPIcs*, pages 27:1–27:17, 2019. doi:10.4230/LIPICS.OPODIS.2019.27.

22 Anisur Rahaman Molla, Kaushik Mondal, and William K. Moses Jr. Efficient dispersion on an anonymous ring in the presence of weak byzantine robots. In *Algorithms for Sensor Systems, ALGOSENSORS*, volume 12503 of *LNCS*, pages 154–169. Springer, 2020. doi:10.1007/978-3-030-62401-9_11.

23 Moumita Mondal and Sruti Gan Chaudhuri. Uniform scattering of robots on alternate nodes of a grid. In *Proc. of 23rd International Conference of Distributed Computing and Networking, ICDCN*, pages 254–259. ACM, 2022. doi:10.1145/3491003.3493231.

24 Takashi Okumura, Koichi Wada, and Xavier Défago. Optimal l-algorithms for rendezvous of asynchronous mobile robots with external-lights. *Theoretical Computer Science*, 979:114198, 2023. doi:10.1016/J.TCS.2023.114198.

25 Debasish Pattanayak, Klaus-Tycho Foerster, Partha Sarathi Mandal, and Stefan Schmid. Conic formation in presence of faulty robots. In *Algorithms for Sensor Systems, ALGOSENSORS*, volume 12503 of *LNCS*, pages 170–185. Springer, 2020. doi:10.1007/978-3-030-62401-9_12.

26 Pavan Poudel, Aisha Aljohani, and Gokarna Sharma. Fault-tolerant complete visibility for asynchronous robots with lights under one-axis agreement. *Theoretical Computer Science*, 850:116–134, 2021. doi:10.1016/J.TCS.2020.10.033.

27 Giuseppe Prencipe. Pattern formation. In *Chapter 3 of [19]*, pages 37–62. Springer, 2019. doi:10.1007/978-3-030-11072-7_3.

28 Samia Souissi, Yan Yang, and Xavier Défago. Fault-tolerant flocking in a k-bounded asynchronous system. In *Proc. of 12th International Conference on Principles of Distributed Systems, OPODIS*, volume 5401 of *LNCS*, pages 145–163. Springer, 2008. doi:10.1007/978-3-540-92221-6_11.

29 Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999. doi:10.1137/S009753979628292X.

30 Yan Yang, Samia Souissi, Xavier Défago, and Makoto Takizawa. Fault-tolerant flocking for a group of autonomous mobile robots. *Journal of Systems and Software*, 84(1):29–36, 2011. doi:10.1016/J.JSS.2010.08.026.

A Proofs of theorems

▶ **Lemma 5** (No Instantaneous Detection). *Let $\mathcal{F}$ be a FD procedure which never outputs false positives. Then, if t_{crash} is the time of the first crash, then $\mathcal{F}$ returns* CORRECT *at time t_{crash}.*

Proof. By contradiction, suppose that $\mathcal{F}$ detects the fault at time t_{crash}. Let σ be the snapshot through which $\mathcal{F}(\sigma)$ returns FAULT. However, σ would be the same if no robot has crashed at time t_{crash}. This contradicts the hypothesis that $\mathcal{F}$ never outputs false positives. ◀

▶ **Lemma 8** (Missing Activation Rule). *Let $r \in \mathcal{R}$ be a crashed robot. Then, the crash of r cannot be identified during $\mathcal{I}(r)$ by a reliable FI procedure.*

Proof. Assume all the robots in $\mathcal{R}$ have the same coordinate system at any activation. Consider an activation scheduler $\mathfrak{A}$ and a suppression scheduler $\mathfrak{C}_\mathfrak{c}$ such that $\mathcal{I}(r) = [\mathfrak{c}(r), \mathfrak{fma}(r)]$ is the FIT of a robot r, with $\mathfrak{c}(r) \leq \mathfrak{fma}(r)$. By contradiction, let $\mathcal{F}$ be a FI reliable procedure and let $t' \in [\mathfrak{c}(r), \mathfrak{fma}(r)]$ be the time when another robot r' identifies r as crashed by executing $\mathcal{F}(\sigma)$ where σ is the snapshot of r' taken at time t'. Now, let us consider another suppression scheduler $\mathfrak{C}'_{\mathfrak{c}'}$ which is identical to $\mathfrak{C}_\mathfrak{c}$ except that the crash time of r is $\mathfrak{c}'(r) > t'$. According to the combined scheduler $\mathfrak{A} \setminus \mathfrak{C}'_{\mathfrak{c}'}$, all the robots perform the same actions as under $\mathfrak{A} \setminus \mathfrak{C}'_{\mathfrak{c}'}$ until $t' - 1$. So, robot r' at time t' takes the same snapshot as under $\mathfrak{A} \setminus \mathfrak{C}_\mathfrak{c}$ and thus identifies r as faulty by executing $\mathcal{F}(\sigma)$. However, under $\mathfrak{C}'_{\mathfrak{c}'}$, robot r will be crashed at $\mathfrak{c}'(r) > t'$. This contradicts the hypothesis that $\mathcal{F}$ is reliable. ◀

▶ **Lemma 10.** *Let t_{detect} be the first time when a reliable FD procedure detects a fault in a swarm $\mathcal{R}$. Then, $t_{detect} > \min_{r\in\mathcal{R}}\{\mathfrak{fma}(r)\}$.*

Proof. The proof is similar as in Lemma 8. Assume all the robots have the same coordinate system at any activation. Consider an activation scheduler $\mathfrak{A}$ and a suppression scheduler $\mathfrak{C}_{\mathfrak{c}}$ for $\mathcal{R}$, so that $t_{\mathfrak{fma}} = \min_{r\in\mathcal{R}}\{\mathfrak{fma}(r)\}$ and $\mathcal{R}_{\mathfrak{fma}} = \arg\min_{r\in\mathcal{R}} \mathfrak{fma}(r)$ is the subset of robots which first miss an activation (at time $t_{\mathfrak{fma}}$). By contradiction, let $t_{detect} \leq t_{\mathfrak{fma}}$ be the first time when a correct robot r' detects a fault exists in the swarm by executing a reliable FD procedure $\mathcal{F}$. Now, let us consider another suppression scheduler $\mathfrak{C}'_{\mathfrak{c}'}$ which is identical to $\mathfrak{C}_{\mathfrak{c}}$ except that the crash time of any robot $r \in \mathcal{R}$ is $\mathfrak{c}'(r) > t_{detect}$. According to the combined scheduler $\mathfrak{A} \setminus \mathfrak{C}'_{\mathfrak{c}'}$, all the robots perform the same actions until $t_{detect} - 1$. So, robot r' at time t_{detect} takes the same snapshot as under $\mathfrak{A} \setminus \mathfrak{C}_{\mathfrak{c}}$ and thus detects a fault by executing $\mathcal{F}$. However, under $\mathfrak{C}'_{\mathfrak{c}'}$, no robot has crashed before t_{detect}. This contradicts the hypothesis that $\mathcal{F}$ is reliable. ◀

▶ **Theorem 11.** *$\mathcal{F}_M$ is a reliable model-dependent FD or FI procedure for a model $M \in \mathcal{M}$ if and only if $\Psi(\mathbb{A}, \mathcal{F}_M)$ guarantees correctness and soundness for any primary algorithm $\mathbb{A}$ under M.*

Proof. Below, we use $\mathcal{F}$ instead of $\mathcal{F}_M$ to lighten the notation. We prove both the directions:

(⇒) By hypothesis, $\mathcal{F}$ detects/identifies a fault after a finite delay (so $\mathcal{F}(\sigma)$ returns a FAULT value after a finite delay from the fault occurrence). Moreover, after such a delay, $\mathcal{F}$ will always detect/identify the fault in the subsequent rounds. If $\mathcal{F}$ returns at least a FAULT value, then $\Psi(\mathbb{A}, \mathcal{F})$ freezes the robot. Since this happens for any correct robot activated in the swarm, the whole swarm will freeze (thus $\Psi(\mathbb{A}, \mathcal{F})$ is *sound*).
By hypothesis, $\mathcal{F}$ will always return a CORRECT value if no fault occurs: this makes $\Psi(\mathbb{A}, \mathcal{F})$ behave exactly as $\mathbb{A}$ (thus $\Psi(\mathbb{A}, \mathcal{F})$ is *correct*).

(⇐) By hypothesis, if a crash occurs, then $\Psi(\mathbb{A}, \mathcal{F})$ makes all correct robots detect/identify it in finite delay, and freeze the swarm always maintaining coherence about the detected/identified faults in the following rounds. Let us assume that $\mathbb{A}$ is an algorithm that makes any activated robot change its position at any activation. Thus, according to Algorithm 1, $\Psi(\mathbb{A}, \mathcal{F})$ must stop entering into the "if" body and instead it must continue executing the "else" statement from a certain time onward. Thus, $\mathcal{F}$ must guarantee (ii) to detect/identify the crash after a *finite delay* and (iii) to continue detecting/identifying in the next rounds (*coherence*).
By hypothesis, if no crash occurs, then $\Psi(\mathbb{A}, \mathcal{F})$ continues solving the primary problem. So, it must always execute the "if" body. Thus, $\mathcal{F}$ must always return CORRECT, guaranteeing to provide *no false positives* (i). ◀

▶ **Theorem 15.** *A reliable FD procedure for $\mathcal{FCOM}^{\mathsf{F}}$ cannot exist assuming 1-survival crash schedulers.*

Proof. By contradiction let $\mathcal{F}$ be a reliable model-dependent procedure for $\mathcal{FCOM}^{\mathsf{F}}$ which tolerates up to $n-1$ crashes. Let us consider a swarm $\mathcal{R}$ executing $\Psi(\mathbb{A}, \mathcal{F})$ for a given primary algorithm $\mathbb{A}$. Let $\mathfrak{c}_1$ be a 1-survival crash scheduler such that $\mathfrak{c}_1(r) = \infty$, while $\mathfrak{c}_1(r') = t \neq \infty$ for any $r' \in \mathcal{R} \setminus \{r\}$. Let us assume r has a fixed local coordinate system, and let us assume that $\mathbb{A}$ makes r always stay still in its original position. By hypothesis, r is activated at time t, it executes $\mathcal{F}(\sigma)$ and detects no fault in $\mathcal{R}$ exists (by Lemma 10). From time t onward, r will always be the only robot to be activated. Since r cannot see its lights' color and since neither its coordinate system nor position changes round by round, the taken snapshot will always be σ. Thus, r will never detect the fault. Contradiction achieved. ◀

▶ **Theorem 16.** *A punctual FD procedure for $\mathcal{LUMI}^{\text{RR}}$ cannot exist, even considering $(n-1)$-survival crash time schedulers.*

Proof. By contradiction, let $\mathcal{F}$ be a punctual FD procedure for $\mathcal{LUMI}^{\text{RR}}$. Let $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$ be a swarm of robots under $\mathcal{LUMI}^{\text{RR}}$ executing $\Psi(A, \mathcal{F})$ for a given primary algorithm $\mathbb{A}$. Consider the two combined schedulers $\mathfrak{A}_0 \setminus \mathfrak{C}_0$ and $\mathfrak{A}_1 \setminus \mathfrak{C}_1$ defined as

$$\mathfrak{A}_0(t) = \{r_{0+t}\}, \quad \mathfrak{C}_0(t) = \{r_0\} \text{ (i.e., } r_0 \text{ crashes at time 0)}$$
$$\mathfrak{A}_1(t) = \{r_{1+t}\}, \quad \mathfrak{C}_1(t) = \varnothing \text{ (i.e., no robot crashes)}$$

for any $t \in \mathbb{N}$ (indices are considered in modulo n). So, by hypothesis, r_1 is activated at time 1 under $\mathfrak{A}_0 \setminus \mathfrak{C}_0$, and it correctly detects that r_0 has crashed at time 0 by executing $\mathcal{F}$. Formally, r_1 executes $\mathcal{F}(\sigma)$ where σ is the snapshot taken by r_1 at time 1 using Ξ as coordinate system, and returns a FAULT.

Now, let us consider the same swarm under $\mathfrak{A}_1 \setminus \mathfrak{C}_1$ and suppose that r_1 has Ξ as its coordinate system. So, r_1 is activated at time 0, takes the same snapshot σ and executes $\mathcal{F}(\sigma)$ which still returns FAULT. However, no crash has occurred. Contradiction achieved. ◀

▶ **Theorem 17.** *A reliable FD procedure for $\mathcal{FCOM}^{\text{RR}}$ cannot exist even assuming 2-survival crash schedulers.*

Proof. By contradiction, let $\mathcal{F}$ be a reliable FD procedure for $\mathcal{FCOM}^{\text{RR}}$. W.l.o.g. assume that $\mathfrak{L}_{\mathcal{F}} = \{f_0, \ldots, f_{k-1}\}$ for a constant $k > 0$, with f_0 being the default color. Let us consider the `SuperPoint` problem where $n > k$ robots lie on the same point and have to stay still, and let us consider the algorithm $\mathbb{A}_{NOP}$ which trivially solves the problem making robots maintain their initial position and primary color $\not{b}$. Assume all the robots always have the same egocentric coordinate system considering the super-point as the origin $\mathbf{O} = (0, 0)$. Assume that $(r_0, \ldots, r_{n-1})$ is the round-robin activation sequence of the robots, and let $0 \leq t \leq k$ be the first time where a robot sets its `ser` light to a color $\bar{f} \in \mathfrak{L}_{\mathcal{F}}$ which is already set in at least another robot, say r_h. Note that this time always exists since $|\mathfrak{L}_{\mathcal{F}}| = k < n$. Formally, r_t is activated at time t and computes $\Psi(\mathbb{A}_{NOP}, \mathcal{F})(\sigma) = (\mathbf{O}, \not{b}, \bar{f})$, where σ contains the multiplicity $\{(\mathbf{O}, \not{b}, \bar{f})\}_{\geq 1}$. Moreover, assume that at time t, all the robots except for r_h and r_t crash, while the two remaining robots remain alive forever. So, from any $t' > t$, the two survival robots will be activated in alternate rounds, will take the same snapshot σ, and will always execute $\Psi(\mathbb{A}_{NOP}, \mathcal{F})(\sigma) = (\mathbf{O}, \not{b}, \bar{f})$, thus never detecting the fault. Contradiction achieved. ◀

▶ **Theorem 18.** *A reliable FD procedure for a model $M \in \mathcal{M}$ under `SSYNCH` cannot exist.*

Proof. By contradiction, let $\mathcal{F}$ be a reliable FD procedure under $\mathcal{LUMI}^{\text{S}}$ (i.e., the strongest `SSYNCH` model). Let $\mathcal{R} = \{r_0, \ldots, r_{n-1}\}$ be a swarm under $\mathcal{LUMI}^{\text{S}}$. Suppose all the robots use the same coordinate system Ξ. Suppose the swarm works under a `SSYNCH` activation scheduler $\mathfrak{A}$ and a crash activation scheduler $\mathfrak{c}$ so that r_0 is the only robot to crash, and it crashes at time t_1. Formally, $\mathfrak{c}(r_0) = t_1$ and $\mathfrak{c}(r_{i \neq 0}) = \infty$. We assume w.l.o.g. that r_1 is the first robot to correctly detect the fault at time $t_2 > t_1$ by running $\mathcal{F}(\sigma)$ where σ is the snapshot of r_1 at time t_2. Moreover, let $r_0 \in \mathfrak{A}(t_1)$ and $r_0 \notin \mathfrak{A}(t')$ for any $t_1 < t' \leq t_2$. Now consider the schedulers $\mathfrak{A}'$ and $\mathfrak{c}'$ so that

$$\mathfrak{A}' = \mathfrak{A} \text{ except for: } \mathfrak{A}'(t_1) = \mathfrak{A}(t_1) \setminus \{r_0\}$$
$$\mathfrak{c}'(r_i) = \infty \ \forall i$$

Under $\mathfrak{A}' \setminus \mathfrak{c}'$, the robots that execute their LCM are the same as under $\mathfrak{A} \setminus \mathfrak{c}$ until t_2, so the evolution of the swarm is the same. Specifically, r_1 is activated at time t_2, takes the same snapshot σ, and executes $\mathcal{F}(\sigma)$ which still detects the fault. However, no crash has occurred. Contradiction achieved. ◀