Safe Flocking in Spite of Actuator Faults

Taylor Johnson and Sayan Mitra

University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

Abstract. The safe flocking problem requires a collection of N mobile agents to (a) converge to and maintain an equi-spaced lattice formation, (b) arrive at a destination, and (c) always maintain a minimum safe separation. Safe flocking in Euclidean spaces is a well-studied and difficult coordination problem. Motivated by real-world deployment of multi-agent systems, this paper studies one-dimensional safe flocking, where agents are afflicted by *actuator faults*. An actuator fault is a new type of failure that causes the affected agent to be stuck with an arbitrary velocity. In this setting, first, a self-stabilizing solution for the problem is presented. This relies on a failure detector for actuator faults. Next, it is shown that certain actuator faults cannot be detected, while others may require $O(N)$ time for detection. Finally, a simple failure detector that achieves the latter bound is presented. Several simulation results are presented for illustrating the effects of failures on the progress towards flocking.

Keywords: failure detector, flocking, safety, stabilization, swarming

1 Introduction

Safe flocking is a distributed coordination problem that requires a collection of mobile agents situated in a Euclidean space to satisfy three properties, namely to: (a) form and maintain an equi-spaced lattice structure or a *flock*, (b) reach a specified destination or *goal* position, and (c) always maintain a minimum *safe* separation. The origins of this problem can be traced to biological studies aimed at understanding the rules that govern flocking in nature (see for example [13,11]). More recently, recognizing that such understanding could aid the design of autonomous robotic platoons or swarms, the problem as stated above and its variants have been studied in the robotics, control, and multi-agent systems literature (see [7,5,12,8,14] and references therein). Typically, the problem is studied for agents with synchronous communication, without failures, and with double-integrator dynamics—that is, the distributed algorithm sets the acceleration for each agent. To the best of our knowledge, even in this setting, safe-flocking is an open problem, as existing algorithms require unbounded accelerations for guaranteeing safety [12], which cannot be achieved in practice.

In this paper, we study one-dimensional safe-flocking within the realm of synchronous communication, but with a different set of dynamics and failure assumptions. First, we assume rectangular single-integrator dynamics. That is, at the beginning of each round, the algorithm decides a target point u_i for agent i based on messages received from i's neighbors, and agent i moves with

bounded speed $\dot{x_i} \in [v_{min}, v_{max}]$ in the direction of u_i for the duration of that round. This simplifies the dynamics and achieving safety becomes relatively easy. Even in this setting however, it is nontrivial to develop and prove that an algorithm provides collision avoidance, as illustrated by an error that we found in the inductive proof of safety in [7]. To fix the error, the algorithm from that paper requires the modification presented later in this paper. Nevertheless, the model obtained with this rectangular dynamics overapproximates any behavior that can be obtained with double integrator dynamics with bounded acceleration. Our algorithm combines the corrected version of the algorithm from [7] with Chandy-Lamport's global snapshot algorithm [3]. The key idea is that each agent periodically computes its target based on messages received from its neighbors and moves toward this target with some arbitrary but bounded velocity. The targets are computed such that the agents preserve safe separation and they eventually form a *weak flock*, which remains invariant, and progress is ensured to a tighter *strong flock*. Once a strong flock is attained, this property can be detected through the use of a distributed snapshot algorithm [3]. Once this is detected, the detecting agent makes a move towards the destination, sacrificing the strong flock in favor of making progress towards the goal, but still preserving the weak flock.

Unlike the algorithms in [7,5,8,12] that provide convergence to a flock, we require the stronger *termination*. Our algorithm achieves termination through *quantization*: we assume that there exists a constant $\beta > 0$ such that an agent i moves in a particular round if and only if the computed target u_i is more than β away from the current position x_i. We believe that such quantized control is appropriate for realistic actuators, and useful for most power-constrained settings where it is undesirable for the agents to move forever in order to achieve convergence. Quantization affects the type of flock formation that we can achieve and also makes the proof of termination more interesting.

Finally, we allow agents to be affected by *actuator failures*. This physically corresponds to, for example, an agents's motors being stuck at an input voltage. Actuator faults are permanent and cause the afflicted agents to move forever with a bounded and constant velocity. Actuator faults are a new class of failures that we believe are going to be important in designing and analyzing a wide range of distributed cyber-physical systems [9]. Unlike byzantine faults, behaviors resulting from actuator faults are constrained by physical laws. Also, unlike crash failures which typically thwart progress, but do not violate safety, actuator failures can also violate safety. A faulty agent has to be detected (and possibly avoided) by the non-faulty agents. In this paper, we assume that after an actuator failure, an agent continues to communicate and compute, but its actuators continue to move with the arbitrary but constant velocity.

Our flocking algorithm determines *only* the direction in which an agent should move, based on neighbor information. The speed with which it moves is chosen nondeterministically over a range (this makes the algorithm implementation independent with respect to the lower-level motion controller). Thus, the only way of detecting failures is to observe that an agent has moved in

the wrong direction. Under some assumptions about the system parameters, a simple lower-bound is established, indicating that no detection algorithm can detect failures in less than $O(N)$ rounds. A failure detector is presented that utilizes this idea in detecting certain classes of failures in $O(N)$ rounds. Unfortunately, certain failures lead to a violation of safety in fewer rounds, so a failure detector which detects failures faster than $O(N)$ rounds is necessary to ensure safety. However, some failures are undetectable, such as an agent failing with zero velocity at the goal, thus we establish that no such failure detector exists. But, it is shown that the failure detector with $O(N)$ detection time can be combined with the flocking algorithm to guarantee the required safety and progress properties in the face of a restricted class of actuator failures. Lastly, non-faulty agents need a way to avoid faulty ones. In one dimension (such as on highways), this is possible if there are multiple *lanes*.

In summary, the key contributions of the paper are the following:

(a) A solution to the one-dimensional safe flocking problem in the face of actuator faults, quantization, and with bounded control. The solution brings distributed computing ideas (self-stabilization and failure detection) to a distributed control problem.
(b) Formal introduction of the notion of actuator faults and stabilization in the face of such faults.

2 System Model

This section presents a formal model of the distributed flocking algorithm modeled as a discrete transition system, as well as formal specifications of the system properties to be analyzed. For $K \in \mathbb{N}$, $[K] \triangleq \{1, \ldots, K\}$ and for a set S $S_\perp \triangleq S \cup \{\perp\}$. A *discrete transition system* $\mathcal{A}$ is a tuple $\langle X, Q, Q_0, A, \rightarrow \rangle$, where (i) X is a set of variables with associated types, (ii) Q is the the set of *states* which is the set of all possible valuations of the variables in X, (iii) $Q_0 \subseteq Q$ is the set of *start states*, (iv) A is a set of transition *labels*, and (v) $\rightarrow \subseteq Q \times A \times Q$ is a set of *discrete transitions*. An *execution fragment* of A is an (possibly infinite) alternating sequence of states and transition names, $\alpha = \mathbf{x}_0, a_1, \mathbf{x}_1, \ldots$, such that for each index k appearing in α, $(\mathbf{x}_k, a_{k+1}, \mathbf{x}_{k+1}) \in \rightarrow$.An *execution* is an execution fragment with $\mathbf{x}_0 \in Q_0$.

A state $\mathbf{x}$ is *reachable* if there exists a finite execution that ends in $\mathbf{x}$. A *stable* predicate $S \subseteq Q$ is a set of states that is closed under $\rightarrow$. If a stable predicate S contains Q_0, then it is called an *invariant* predicate and the reachable states of $\mathcal{A}$ are then contained in S. A *safety* property specified by a predicate $S \subseteq Q$ is satisfied by $\mathcal{A}$ if all of its reachable states are contained in S. Self-stabilization is a property of non-masking fault tolerance which guarantees that once new failures cease to occur, the system eventually returns to a legal state [4]. In this paper, we model actuator failures by transitions with the special label fail. Given $G \subseteq Q$, $\mathcal{A}$ *self-stabilizes* to G if (a) G is a stable predicate for $\mathcal{A}$ without the fail-transitions, and (b) from every reachable state of $\mathcal{A}$ (including fail transitions), every *fail*-free execution fragment eventually reaches G.

2.1 Model of Safe Flocking System

The distributed system consists of a set of N mobile *agents* physically positioned on N_L infinite, parallel *lanes*. The system can be thought of as a collection of cars in the lanes on a highway. We assume that the system is synchronous and the communication graph is complete[1], that is, agents have synchronized clocks, message delays are bounded, and computations are instantaneous. We will be concerned with synchronous algorithms that operate in rounds. At each round, each agent exchanges messages bearing state information with its neighbors. Agents then update their software state and (nondeterministically) choose their velocities, which they operate with until the beginning of the next round. Under these assumptions, it is convenient to model the system as a collection of discrete transition systems that interact through shared variables.

Let $ID \triangleq [N]$ be the set of unique agent identifiers and $LD \triangleq [N_L]$ be the set of lane identifiers. The following positive constants are used throughout the paper: (a) r_s: minimum required inter-agent gap or *safety distance* in the absence of failures, (b) r_r: reduced safety distance in the presence of failures, (c) r_f: desired maximum inter-agent gap which defines a flock, (d) δ: flocking tolerance parameter, (e) β: quantization parameter, and (f) v_{min}, v_{max}: minimum and maximum velocities.

State Variables. The discrete transition system corresponding to Agent_i has the following *private* variables with initial values in parentheses: (a) $failed(false)$: indicates whether or not agent i has failed, (b) $vf(\perp)$: velocity with which agent i has failed, and (c) L and R: identifiers of the nearest left and right neighbors of agent i. The following *shared* variables are controlled by agent i, but can also be read by i's neighbors: (a) x and xo: current position and position from the previous round of agent i on the real line, (b) u and uo: target position and target from the previous round of agent i (all positions are on the real line), (c) $lane$: the lane currently occupied by agent i, (d) $Suspected$: set of neighbors that agent i believes to have failed. These variables are *shared* in the following sense: At the beginning of each round k, their values are broadcast by Agent_i and are used by the neighbors of agent i to update their states in that round. The discrete transition system modeling the complete ensemble of agents is called System. We refer to states of System with bold letters $\mathbf{x}$, $\mathbf{x}'$, etc., and individual state components of Agent_i by $\mathbf{x}.x_i$, $\mathbf{x}.u_i$, etc.

Actuator Failures and Failure Detection. The actuator failure of agent i is modeled by the occurrence of a transition labeled by fail_i. This transition is always enabled unless i has already failed, and as a result of its occurrence, the variable $failed_i$ is set to $true$. An actuator failure causes the affected agent to move with a constant but arbitrary *failure velocity* forever[2]. At state $\mathbf{x}$, $F(\mathbf{x})$ and $NF(\mathbf{x})$ denote the sets of faulty and non-faulty agent identifiers, respectively.

[1] This communication assumption is relaxed in [9].

[2] It is worth noting that some attention has been given to failure detection in flocking, and most closely related is [6], which works with a similar model of actuator failures.

Agents do not have any direct information regarding the actuator failures of other agents (i.e., agent i cannot read $failed_j$). They have to rely on timely failure detection to avoid violating safety or drifting away from the goal by following a faulty agent. Failure detection at agent i is abstractly captured by the $Suspected_i$ variable and a transition labeled by $\mathsf{suspect}_i$. The $\mathsf{suspect}_i(j)$ transition models a detection of failure of some agent j by $\mathsf{Agent_i}$. Failures are irreversible in our model, and thus so are failure detector suspicions. For agent i, at any given state $Suspected_i \subseteq ID$ is the set of agent identifiers that agent i's failure detector suspects as faulty. $\mathsf{Agent_j}$ is said to be *suspected* if some agent i suspects it, otherwise it is *unsuspected*. Which particular agent suspects a faulty agent j is somewhat irrelevant. We assume the failure detectors of all the agents share information through some background gossip, and when one agent suspects agent i, all the other agents also suspect i in the same round. Denote the sets of suspected and unsuspected agents by $S(\mathbf{x})$ and $NS(\mathbf{x})$, respectively.

The *detection time* is the minimum number of rounds within which every failure is always suspected. In most parts of Section 3 we will assume that there exists a finite detection time k_d for any failure. In Section 3.3, we will discuss specific conditions under which k_d is in fact finite and then give upper and lower bounds for it. We complete the present discussion by describing the failure detection strategy used by our flocking algorithm, which is encoded as the precondition of the $\mathsf{suspect}$ transition. Note that the precondition assumes that i has access to some of j's shared variables, namely x_j, xo_j, u_j and uo_j. When the precondition of $\mathsf{suspect}(j)$ is satisfied at Figure 1, j is added to the $Suspected_i$. This precondition checks that either j moved when it should not have, or that j moved in the wrong direction, away from its computed target. The rationale behind this condition will become clear as we discuss the flocking algorithm.

Neighbors. At state $\mathbf{x}$, let $L(\mathbf{x}, i)$ (and symmetrically $R(\mathbf{x}, i)$) be the nearest non-failed agent left (resp. right) of $\mathsf{Agent_i}$, with ties broken arbitrarily. If no such neighbor exists, then $L(\mathbf{x}, i)$ and $R(\mathbf{x}, i)$ are defined as $\perp$. Let $L_S(\mathbf{x}, i)$ (and symmetrically $R_S(\mathbf{x}, i)$) be the nearest unsuspected agent left (resp. right) of $\mathsf{Agent_i}$ at state $\mathbf{x}$, or $\perp$ if no such agents exist. An unsuspected $\mathsf{Agent_i}$ with both unsuspected left and right neighbors is a *middle agent*. Let the set of middle agent identifiers be $Mids(\mathbf{x})$ for state $\mathbf{x}$. An unsuspected $\mathsf{Agent_i}$ without an unsuspected left neighbor is said to be a *head agent*, and is denoted by $H(\mathbf{x})$. If $\mathsf{Agent_i}$ is unsuspected, is not the head, and does not have an unsuspected right neighbor, it is a *tail agent* and is denoted by $T(\mathbf{x})$.

Flocking Algorithm. The distributed flocking algorithm executed at $\mathsf{Agent_i}$ uses two separate processes (threads): (a) a process for taking distributed global snapshots, and (b) a process for updating the target position for $\mathsf{Agent_i}$.

However, this work uses the developed *motion probes* in failure detection scenarios, but has no stated bounds on detection time as more effort was spent ensuring convergence, assuming that failure detection has occurred within some time, while our work states a detection time bound.

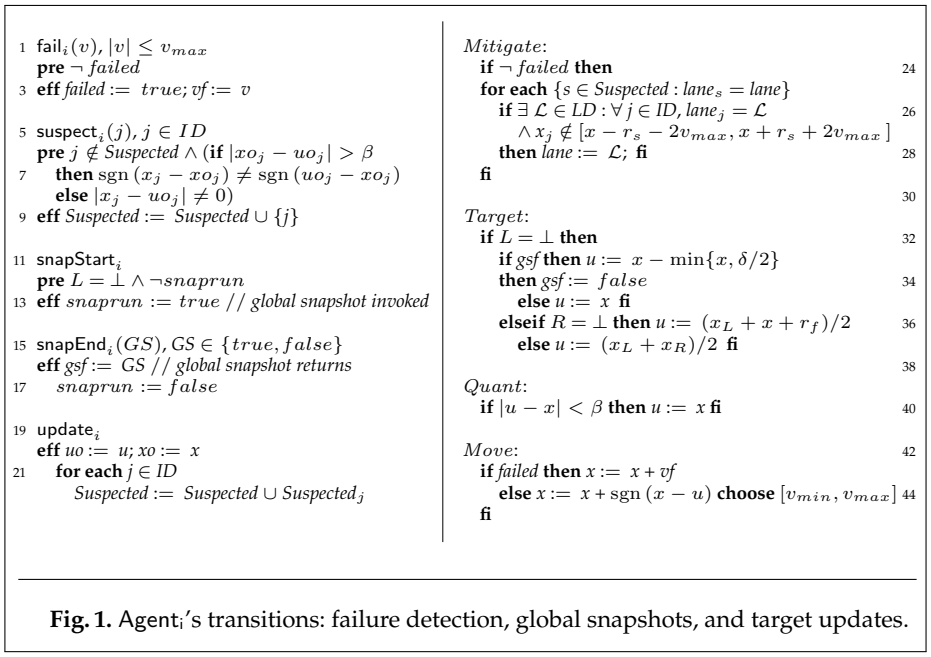

Fig. 1. Agent$_i$'s transitions: failure detection, global snapshots, and target updates.

The snapStart and snapEnd transitions model the periodic initialization and termination of a distributed global snapshot protocol—such as Chandy and Lamport's snapshot algorithm [3]—by the head agent. This global snapshot is used for detecting a stable global predicate, which in turn influences the target computation for the head agent. Although we have not modeled this explicitly, we assume that the snapStart$_i$ transition is performed periodically by the head agent when the precondition is enabled. If the global predicate holds, then snapEnd$(true)$ occurs, otherwise snapEnd$(false)$ occurs. It is straightforward to check that the assumptions necessary for applying Chandy-Lamport's algorithm are satisfied here since (a) we are detecting a stable predicate, (b) the communications graph is always fully connected, and (c) the stable predicates are reachable. Thus, we assume that in any infinite execution, a snapEnd$_i$ transition occurs within $O(N)$ rounds from the occurrence of the corresponding snapStart$_i$ transition.

The update transition models the evolution of all (faulty and non-faulty) agents over a synchronous round. It is composed of four subroutines: $Mitigate$, $Target$, $Quant$, and $Move$, which are executed in this sequence for updating the state of System. The whole update action is instantaneous and atomic; the subroutines are used for clarity of presentation[3]. To be clear, for $\mathbf{x} \overset{\text{update}}{\rightarrow} \mathbf{x}'$, $\mathbf{x}'$ is obtained by applying each of these subroutines. We refer to the intermediate states after $Mitigate$, $Target$, $Quant$, and $Move$ as $\mathbf{x}_M$, $\mathbf{x}_T$, $\mathbf{x}_Q$, and $\mathbf{x}_V$, re-

[3] $Move$ abstractly captures the physical evolution of the system over a round. It is the time-abstract transition corresponding to physical evolution over an interval of time.

spectively. That is, $\mathbf{x}_M \triangleq \mathit{Mitigate}(\mathbf{x})$, $\mathbf{x}_T \triangleq \mathit{Target}(\mathbf{x}_M)$, etc., and observe that $\mathbf{x}' = \mathbf{x}_V = \mathit{Move}(\mathbf{x}_Q)$.

$\mathit{Mitigate}$ is executed by non-faulty agents and attempts to restore safety and progress properties that may be reduced or violated by failures. Target determines a new target to move towards, which is roughly the average of the positions of the closest left and right unsuspected neighbors of any agent. As mentioned before, targets are still computed for faulty agents, but their actuators ignore these new values. Quant is the quantization step which prevents targets u_i computed in the Target subroutine from being applied to real positions x_i, if the difference between the two is smaller than the *quantization parameter* β. It is worth emphasizing that quantization is a key requirement for any realistic algorithm that actuates the agents to move with bounded velocities. Without quantization, if the computed target is very close to the current position of the agent, then the agent may have to move with arbitrarily small velocity over that round. Finally, Move moves agent positions x_i toward the quantized targets.

There are three different rules for target computations based on an agent's belief of whether it is a head, middle, or tail agent. For a state $\mathbf{x}$, each middle agent i attempts to maintain the average of the positions of its nearest unsuspected left and right neighbors (Figure 1, Line 37). Assuming that the goal is to the left of the tail agent, the tail agent attempts to maintain r_f distance from its nearest unsuspected left neighbor (Figure 1, Line 36). The head agent periodically invokes a global snapshot and attempts to detect a certain stable global predicate Flock_S (defined below). If this predicate is detected, then the head agent moves towards the goal (Figure 1, Line 34), otherwise it does not change its target u from its current position x.

2.2 Key Predicates

We now define a set of predicates on the state space of System that capture key properties of safe flocking. These will be used for proving that the algorithm described above solves safe flocking in the presence of actuator faults. We start with safety. A state $\mathbf{x}$ of System satisfies Safety if the distance between every pair of agents on the same lane is at least the safety distance r_s. Formally, $\mathit{Safety}(\mathbf{x}) \triangleq \forall i, j \in ID, i \neq j, \mathbf{x}.lane_i = \mathbf{x}.lane_j \implies |\mathbf{x}.x_i - \mathbf{x}.x_j| \geq r_s$. When failures occur, a reduced inter-agent gap of r_r will be guaranteed. We call this weaker property *reduced safety*: $\mathit{Safety}_R(\mathbf{x}) \triangleq \forall i \in NF(\mathbf{x}), \forall j \in ID, i \neq j, \mathbf{x}.lane_i = \mathbf{x}.lane_j \implies |\mathbf{x}.x_i - \mathbf{x}.x_j| \geq r_r$.

An ϵ-flock is where each non-faulty agent with an unsuspected left neighbor (not necessarily in the same lane) is within $r_f \pm \epsilon$ from that neighbor. Formally, $\mathit{Flock}(\mathbf{x}, \epsilon) \triangleq \forall i \in NF(\mathbf{x}), L_S(\mathbf{x}, i) \neq \perp, \left|\mathbf{x}.x_i - \mathbf{x}.x_{L_S(\mathbf{x},i)} - r_f\right| \leq \epsilon$. In this paper, we will use the Flock predicate with two specific values of ϵ, namely δ (the flocking tolerance parameter) and $\frac{\delta}{2}$. The *weak flock* and the *strong flock* predicates are defined as $\mathit{Flock}_W(\mathbf{x}) \triangleq \mathit{Flock}(\mathbf{x}, \delta)$, and $\mathit{Flock}_S(\mathbf{x}) \triangleq \mathit{Flock}(\mathbf{x}, \frac{\delta}{2})$, respectively.

Related to quantization, we have the *no big moves (NBM)* predicate, where none of the agents (except possibly the head agent) have any valid moves, be-

cause their computed targets are less than β (quantization constant) away from their current positions. $NBM(\mathbf{x}) \stackrel{\Delta}{=} \forall i \in NF(\mathbf{x}), L_S(\mathbf{x}, i) \neq \perp, |\mathbf{x}_T.u_i - \mathbf{x}.x_i| \leq \beta$, where $\mathbf{x}_T$ is the state following the application of *Target* subroutine to $\mathbf{x}$. The *Goal* predicate is satisfied at states where the head agent is within β distance of the goal (assumed to be the origin without loss of generality). Finally, a state satisfies the *Terminal* predicate if it satisfies both *Goal* and NBM.

3 Analysis

The main result of the paper (Theorem 1) is that the algorithm in Figure 1 achieves safe flocking in spite of failures provided: (a) there exists a failure detector that detects actuator failures *sufficiently fast*, and (b) each non-faulty agent has *enough room* to jump to some lane to safely avoid faulty agents and eventually make progress. For the first part of our analysis, we will simply assume that any failure is detected within k_d rounds. In Section 3.3, we shall examine conditions under which k_d is finite and state its lower and upper bounds. Assumption (b) is trivially satisfied if the number of lanes is greater than the total number of failures; but it is also satisfied with fewer lanes, provided the failures are sufficiently apart in space. There are two space requirements for Assumption (b): the first ensures safety and the second ensure progress by preventing "walls" of faulty agents from existing forever and ensuring that infinitely often all non-faulty agents may make progress.

Theorem 1. *Suppose there exists a failure detector which suspects any actuator fault within k_d rounds. Suppose further that $v_{max} \leq (r_s - r_r)/(2k_d)$. Let $\alpha = \mathbf{x}_0, \ldots, \mathbf{x}_p, \mathbf{x}_{p+1}, \ldots$ be an execution where x_p is the state after the last* fail *transition. Let $\alpha_{ff} = \mathbf{x}_{p+1}, \ldots$, be the fail-free suffix of α. Let f be the number of actuator faults.*

(a) If $N_L > f$, or
(b) If $N_L \leq f$ and along α_{ff}, $\forall \mathbf{x} \in \alpha_{ff}$, $\exists \mathcal{L} \in LD$ such that $\forall i \in NF(\mathbf{x}), \forall j \in F(\mathbf{x})$, $\mathbf{x}.lane_j \neq \mathcal{L}$ and $|\mathbf{x}.x_i - \mathbf{x}.x_j| > r_s + 2v_{max}k_d$, and also that infinitely often, $\forall m, n \in F(\mathbf{x})$, $m \neq n$, $|\mathbf{x}.x_m - \mathbf{x}.x_n| > r_s + 2v_{max}$.

Then, (a) Every state in α satisfies the reduced safety property, $Safety_R$, and (b) Eventually $Terminal$ and $Flock_S$ are satisfied.

In what follows, we state and informally discuss a sequence of lemmas that culminate in Theorem 1. Under the assumptions and analysis of this section, the following relationships are satisfied: $NBM \subset Flock_S \subset Flock_W \subset Safety \subset Safety_R$. Detailed proofs of the lemmas appear in the technical report [10]. We begin with some assumptions.

Assumptions. Except where noted in Section 3.3, the remainder of the paper utilizes the assumptions of Theorem 1. Additionally, these assumptions are required throughout the paper: (a) $N_L \geq 2$: there are at least 2 lanes, (b) $r_r < r_s < r_f$: the reduced safety gap r_r required under failures is strictly less than the safety gap r_s in the absence of failures, which in turn is strictly less than the flocking distance, (c) $v_{min} \leq v_{max} \leq \beta \leq \frac{\delta}{4N}$, and (d) the communication graph

of the non-faulty agents is always fully connected, so the graph of non-faulty agents cannot partition. Assumption (c) bounds the minimum and maximum velocities, although they may be equal. It then upper bounds the maximum velocity to be less than or equal to the quantization parameter β. This is necessary to prevent a violation of safety due to overshooting computed targets. Finally, β is upper bounded such that $NBM \subseteq Flock_S$. Intuitively, the bound on β is to ensure that errors from flocking due to quantization do not accumulate along the flock from the head to the tail. This is used to show that eventually $Flock_S$ is satisfied by showing eventually NBM is reached.

3.1 Safety

First, we establish that System satisfies the safety part of the safe flocking problem. The following lemma states that in each round, each agent moves by at most v_{max}, and follows immediately from the specification of System.

Lemma 1. *For any two states* $\mathbf{x}, \mathbf{x}'$ *of* System, *if* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for some transition* a, *then for each agent* $i \in ID$, $|\mathbf{x}'.x_i - \mathbf{x}.x_i| \leq v_{max}$.

The next lemma establishes that, upon changes in which other agents an agent i uses to compute its target position, safety is not violated.

Lemma 2. *For any execution* α, *for states* $\mathbf{x}, \mathbf{x}' \in \alpha$ *such that* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for any* $a \in A$, $\forall i, j \in ID$, *if* $L_S(\mathbf{x}, i) \neq j$ *and* $R_S(\mathbf{x}, j) \neq i$ *and* $L_S(\mathbf{x}', i) = j$ *and* $R_S(\mathbf{x}', j) = i$ *and* $\mathbf{x}.x_{R_S(\mathbf{x},j)} - \mathbf{x}.x_{L_S(\mathbf{x},i)} \geq c$, *then* $\mathbf{x}'.x_{R_S(\mathbf{x}',j)} - \mathbf{x}'.x_{L_S(\mathbf{x}',i)} \geq c$, *for any* $c > 0$.

Invariant 1 shows the spacing between any two non-faulty agents in any lane is always at least r_r, and the spacing between any non-faulty agent and any other agent in the same lane is at least r_r. There is no result on the spacing between any two faulty agents—they may collide. The proof, which is by induction, is given in Appendix A.

Invariant 1. *For any reachable state* $\mathbf{x}$, $Safety_R(\mathbf{x})$.

3.2 Progress

The progress analysis works with fail-free executions, that is, there are no further fail_i transitions. Note that this does not mean $F(\mathbf{x}) = \emptyset$, only that along such executions $|F(\mathbf{x})|$ does not change. This is a standard assumption used to show convergence from an arbitrary state back to a stable set [1], albeit we note that we are dealing with permanent faults instead of transient ones. In this case, the stable set eventually reached are states where $Terminal$ is satisfied. However, note that the first state in such an execution is not entirely arbitrary, as Section 3.1 established that such states satisfy at least $Safety_R$, and all the following analysis relies on this assumption.

First observe that, like safety, progress may be violated by failures. Any failed agent with nonzero velocity diverges by the definition of velocities in Figure 1, Line 43. This observation also highlights why $Flock$ is quantified over agents with identifiers in the set of suspected agents $NS(\mathbf{x})$ and not the set

of failed agents $NF(\mathbf{x})$ or all agents ID—if it were quantified over ID, at no future point could $Flock(\mathbf{x})$ be attained if a failed agent has diverged. Zero velocity failures may also cause progress to be violated, where a "wall" of non-moving failed agents may be created, but such situations are excluded by the second part of Assumption (b) in Theorem 1.

Progress along Fail-Free Executions. In the remainder of this section, we show that once new actuator failures cease occurring, System eventually reaches a state satisfying $Terminal$. This is a convergence proof and we will use a Lyapunov-like function to prove this property. The remainder of this section applies to any infinite fail-free execution fragment, so fix such a fragment α_{ff}.

These descriptions of error dynamics are used in the analysis:

$$e(\mathbf{x}, i) \triangleq \begin{cases} |\mathbf{x}.x_i - \mathbf{x}.x_{\mathbf{x}.L_i} - r_f| & \text{if } i \text{ is a middle or a tail agent,} \\ 0 & \text{otherwise,} \end{cases}$$

$$eu(\mathbf{x}, i) \triangleq \begin{cases} |\mathbf{x}.u_i - \mathbf{x}.u_{\mathbf{x}.L_i} - r_f| & \text{if } i \text{ is a middle or a tail agent,} \\ 0 & \text{otherwise.} \end{cases}$$

Here $e(\mathbf{x}, i)$ gives the error with respect to r_f of Agent_i and its non-suspected left neighbor and $eu(\mathbf{x}, i)$, with respect to target positions $\mathbf{x}.u_i$ rather than physical positions $\mathbf{x}.x_i$.

Now, we make the simple observation from Line 44 of Figure 1 that if a non-faulty agent i moves in some round, then it moves by at least a positive amount v_{min}. Then, Lemma 3 states that from any reachable state $\mathbf{x}$ which does not satisfy NBM, the maximum error over all non-faulty agents in non-increasing. This is shown by first noting that only the update transition can cause any change of $e(\mathbf{x}, i)$ or $eu(\mathbf{x}, i)$, and then analyzing the change in value of $eu(\mathbf{x}, i)$ for each of the computations of u_i in the $Target$ subroutine of the update transition. Then it is shown that applying the $Quant$ subroutine cannot cause any $eu(\mathbf{x}, i)$ to increase, and finally the computation of x_i in the $Move$ subroutine does not cause any $e(\mathbf{x}, i)$ to increase.

Lemma 3. *For reachable states $\mathbf{x}, \mathbf{x}'$, if $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ and $\mathbf{x} \notin NBM$, for some $a \in A$, then* $\max_{i \in NF(\mathbf{x})} e(\mathbf{x}', i) \leq \max_{i \in NF(\mathbf{x})} e(\mathbf{x}, i)$.

Next, Lemma 4 shows sets of states satisfying NBM are invariant, a state satisfying NBM is reached, and gives a bound on the number of rounds required to reach such a state. Define the candidate Lyapunov function as $V(\mathbf{x}) \triangleq \sum_{i \in NF(\mathbf{x})} e(\mathbf{x}, i)$. Define the maximum value the candidate Lyapunov function obtained over any state $\mathbf{x} \in \alpha_{ff}$ satisfying NBM as $\gamma \triangleq \sup_{\mathbf{x} \in NBM} V(\mathbf{x})$.

Lemma 4. *Let $\mathbf{x}_k$ be the first state of α_{ff}, and let the head agent's position be fixed. If $V(\mathbf{x}_k) > \gamma$, then the* update *transition decreases $V(\mathbf{x}_k)$ by at least a positive constant ψ. Furthermore, there exists a finite round c such that $V(\mathbf{x}_c) \leq \gamma$, where $\mathbf{x}_c \in NBM(\mathbf{x})$ and $k < c \leq \left\lceil \frac{V(\mathbf{x}_k) - \gamma}{\psi} \right\rceil$, where $\psi = v_{min}$.*

Lemma 4 stated a bound on the time it takes for System to reach the set of states satisfying NBM. However, to satisfy $Flock_S(\mathbf{x})$, all $\mathbf{x} \in NBM$ must be inside the set of states that satisfy $Flock_S$, and the following lemma states this. From any state $\mathbf{x}$ that does not satisfy $Flock_S(\mathbf{x})$, there exists an agent that computes a control that will satisfy the quantization constraint and hence make a move towards NBM. This follows from the assumption that $\beta \leq \frac{\delta}{4N}$.

Lemma 5. *If $Flock_S(\mathbf{x})$, then $V(\mathbf{x}) \leq \sum_{i \in NF(\mathbf{x})} e(\mathbf{x}, i) = \frac{\delta |NF(\mathbf{x})|}{4}$.*

Now we observe that $Flock_W$ is a stable predicate, that is, that once a weak flock is formed, it remains invariant. This result follows from analyzing the $Target$ subroutine which computes the new targets for the agents in each round. Note that the head agent moves by a fixed distance $\frac{\delta}{2}$, only when $Flock_S$ holds, which guarantees that $Flock_W$ is maintained even though $Flock_S$ may be violated. This establishes that for any reachable state $\mathbf{x}'$, if $V(\mathbf{x}') > V(\mathbf{x})$, then $V(\mathbf{x}') < \frac{\delta |NF(\mathbf{x})|}{2}$.

Lemma 6. *$Flock_W$ is a stable predicate.*

The following corollary follows from Lemma 4, as $Flock_S(\mathbf{x})$ is violated after becoming satisfied only if the head agent moves, in which case $\mathbf{x}'.x_{H(\mathbf{x}')} < \mathbf{x}.x_{H(\mathbf{x})}$, which causes $V(\mathbf{x}') \geq V(\mathbf{x})$.

Corollary 1. *For $\mathbf{x} \in \alpha_{ff}$ such that, if $Flock_S(\mathbf{x})$, $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ $\forall a \in A$, and $\mathbf{x}.x_{H(\mathbf{x})} = \mathbf{x}'.x_{H(\mathbf{x}')}$, then $Flock_S(\mathbf{x}')$.*

The following lemma—with Assumption (b) of Theorem 1 that gives eventually a state is reached such that non-faulty agents may pass faulty agents—is sufficient to prove that $Terminal$ is eventually satisfied in spite of failures. After this number of rounds, no agent $j \in NF(\mathbf{x})$ believes any $i \in F(\mathbf{x})$ is its left or right neighbor, and thereby any failed agents diverge safely along their individual lanes if $|\mathbf{x}.v_i| > 0$ by the observation that failed agents with nonzero velocity diverge. Particularly, after some agent j has been suspected by all non-faulty agents, the $Mitigate$ subroutine of the update transition shows that the non-faulty agents will move to a different lane at the next round. This shows that mitigation takes at most one additional round after detection, since we have assumed in Theorem 1 that there is always free space on some lane. This implies that so long as a failed agent is detected prior to safety being violated, only one additional round is required to mitigate, so the time of mitigation is a constant factor added to the time to suspect, resulting in the constant c being linear in the number of agents.

Lemma 7. *For any fail-free execution fragment α_{ff}, if $\mathbf{x}.failed_i$ at some state $\mathbf{x} \in \alpha_{ff}$, then for a state $\mathbf{x}' \in \alpha_{ff}$ at least c rounds from $\mathbf{x}$, $\forall j \in ID.\mathbf{x}'.L_j \neq i \wedge \mathbf{x}'.R_j \neq i$.*

The next theorem shows that System eventually reaches the goal as a strong flock, that is, there is a finite round t such that $Terminal(\mathbf{x}_t)$ and $Flock_S(\mathbf{x}_t)$ and shows that System is self-stabilizing when combined with a failure detector.

Theorem 2. *Let α_{ff} be written $\mathbf{x}_0$, $\mathbf{x}_1$, ... Consider the infinite sequence of pairs $\langle \mathbf{x}_0.x_{H(\mathbf{x}_0)}, V(\mathbf{x}_0)\rangle$, $\langle \mathbf{x}_1.x_{H(\mathbf{x}_1)}, V(\mathbf{x}_1)\rangle$, ..., $\langle \mathbf{x}_t.x_{H(\mathbf{x}_t)}, V(\mathbf{x}_t)\rangle$, ... Then, there exists t at most $\left\lceil \frac{(V(\mathbf{x}_0)-|NF(\mathbf{x})|\delta/4)}{v_{min}} \right\rceil + \left\lceil \frac{|NF(\mathbf{x})|\delta/4}{v_{min}} \right\rceil \max\{1, \frac{\mathbf{x}_0.x_{H(\mathbf{x}_0)}}{v_{min}} O(N)\}$ rounds from $\mathbf{x}_0$ in α_{ff}, such that: (a) $\mathbf{x}_t.x_{H(\mathbf{x}_t)} = \mathbf{x}_{t+1}.x_{H(\mathbf{x}_{t+1})}$, (b) $V(\mathbf{x}_t) = V(\mathbf{x}_{t+1})$, (c) $\mathbf{x}_t.x_{H(\mathbf{x}_t)} \in [0, \beta]$, (d) $V(\mathbf{x}_t) \leq |NF(\mathbf{x})| \frac{\delta}{4}$, (e) $Terminal(\mathbf{x}_t)$, and (f) $Flocks_S(\mathbf{x}_t)$.*

3.3 Failure Detection

In the earlier analysis we assumed that it is possible to detect all actuator faults within finite number of rounds k_d. Unfortunately this is not true, as there exist failures which cannot be detected at all. A trivial example of such an undetectable failures is the failure of a node with 0 velocity at a terminal state, that is, a state at which all the agents are at the goal in a flock and therefore are static. While such failures were undetectable in any number of rounds, these failures do not violate *Safety* or *Terminal*. It turns out that only failures which cause a violation of safety or progress may be detected. All the proofs for this section are given in Appendix A.

Lower-Bound on Detection Time. While the occurrence of $\mathsf{fail}_i(v)$ may never be detected in some cases as just illustrated, we show a lower-bound on the detection time for all $\mathsf{fail}_i(v)$ transitions that can be detected. The following lower-bound applies for executions beginning from states that do not *a priori* satisfy $Terminal$. It says that a failed agent mimicked the actions of its correct non-faulty behavior in such a way that despite the failure, System still progressed to NBM as was intended. From an arbitrary state, it takes $O(N)$ rounds to converge to a state satisfying NBM by Lemma 4.

Lemma 8. *The lower-bound on detection time of actuator failures which may be detected is $O(N)$.*

Next we show that the the failure detection mechanism incorporated in Figure 1 does not produce any false positives.

Lemma 9. *In any reachable state $\mathbf{x}$, $\forall j \in \mathbf{x}.Suspected_i \Rightarrow \mathbf{x}.failed_j$.*

The next lemma shows a partial *completeness* property [2] and gives an upper bound on the detection time for any detectable failure.

Lemma 10. *Suppose that $\mathbf{x}$ is a state in the fail-free execution fragment α_{ff} such that $\exists$ $j \in F(\mathbf{x})$, $\exists\, i \in ID$, and j is not suspected by i. Suppose that either (a) $|\mathbf{x}.xo_j - \mathbf{x}.uo_j|$ $\leq \beta$ and $|\mathbf{x}.x_j - \mathbf{x}.uo_j| \neq 0$, or (b) $|\mathbf{x}.xo_j - \mathbf{x}.uo_j| > \beta$ and $\mathrm{sgn}\,(\mathbf{x}.x_j - \mathbf{x}.xo_j) \neq$ $\mathrm{sgn}\,(\mathbf{x}.uo_j - \mathbf{x}.xo_j)$. Then, $\mathbf{x} \overset{\mathsf{suspect}_i(j)}{\rightarrow} \mathbf{x}'$.*

Now we show a bound on the number of rounds to detect any failure which may be detected using the failure detection mechanism incorporated in Figure 1 by applying Lemma 8 with Lemmata 9 and 10, and that agents share suspected sets in Figure 1, Line 22. This states that the detection time is $O(N)$ and that eventually all non-faulty agents know the set of failed agents.

Corollary 2. *For any state $\mathbf{x}_k \in \alpha_{ff}$ such that $\mathbf{x}_k \notin Terminal$, there exists a round $\mathbf{x}_s$ in α_{ff} such that $\forall i \in NF(\mathbf{x}_s)$, $\mathbf{x}_s.Suspected_i = F(\mathbf{x})$ and $k - s$ is $O(N)$.*

3.4 Simulations

Simulation studies were performed, where flocking convergence time (as by Lemma 4), goal convergence time (as by Theorem 2), and failure detection time (as by Corollary 2) were of interest. Unless otherwise noted, the parameters are chosen as $N = 6$, $N_l = 2$, $r_s = 20$, $r_f = 40$, $r_c = 250$, $\delta = 10$, $\beta = \frac{\delta}{4(N)}$, $v_{min} = \frac{\beta}{2}$, $v_{max} = \beta$, the head agent starts with position at r_f, and the goal is chosen as the origin. Figure 2 shows the value of the Lyapunov function V and maximum

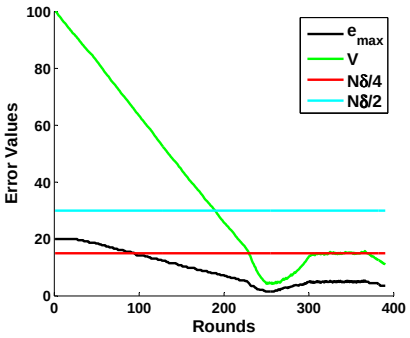

Fig. 2. Expansion showing max error e_{max}, Lyapunov function value V, with weak and strong flocking constants.

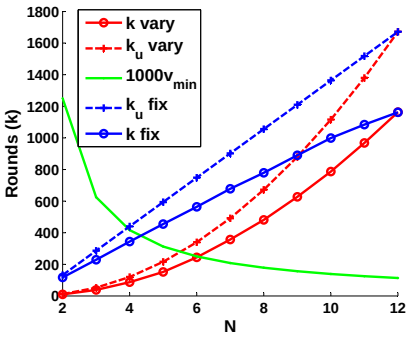

Fig. 3. Rounds k (upper bounded by k_u) to reach $Terminal$ versus number of agents N with fixed and varying v_{min}.

agent error from flocking, e_{max}. The initial state is that each agent is spaced by r_s from its left neighbor, so the flock "expands" [7] to form a strong flock, prior to the head agent moving towards the goal. Observe that while moving towards the goal, $Flock_S$ is repeatedly satisfied and violated, with invariance of $Flock_W$.

Figure 3 shows that for a fixed value of v_{min}, the time to convergence to NBM is linear in the number of agents. This choice of fixed v_{min} must be for the largest number of agents, 12 in this case, as v_{min} is upper bounded by $\beta = \frac{\delta}{4N}$ which is a function of N. As v_{min} is varied the inverse relationship with N is observed, resulting in a roughly quadratic growth of convergence time to NBM. This illustrates linear convergence time as well as linear detection time, as this is bounded by the convergence time from Corollary 2. The initial state was for expansion, so each agent was spaced at r_s from its left neighbor.

Figure 4 shows the detection time as a function of which agent fails with what failure velocity from three different types of initial states. Expansion is where agents start spaced at r_s, contraction is where agents start spaced at $2r_f$, and mixed has expansion and contraction, particularly where agents with even ids are spaced $2r_f$ from their left neighbor, and odd ids are spaced by r_s. Frequently there is one round to detection. For instance, in the expansion case, each failed agent i except the tail are detected in one round when $vf_i \neq 0$ since a violation of safety occurs, whereas detecting that the head agent has failed with zero velocity requires convergence of the system to a strong flock. Detecting the

	Expansion			Contraction			Mixed		
Id	$-v_{max}$	0	v_{max}	$-v_{max}$	0	v_{max}	$-v_{max}$	0	v_{max}
1	1	228	1	1	487	1	1	64	1
2	1	26	1	1	28	1	1	1	49
3	1	18	1	1	19	1	34	1	1
4	1	9	1	1	9	1	1	1	34
5	1	4	1	1	4	1	49	1	1
6	1	1	138	308	1	1	1	1	22

Fig. 4. Detection time when a single agent i fails at round 0 with velocity $-v_{max}$, 0, or v_{max} from an expansion, contraction, and mixed initial state.

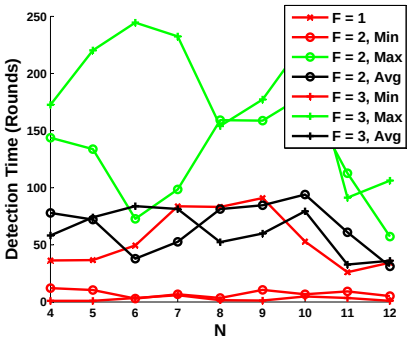

Fig. 5. Detection time versus number of agents N with varying number of failures.

tail agent has failed with v_{max} requires many rounds to detect as well, as this mimics the expansive behavior. In the contraction case, each failed agent i except the tail is detected in one round when $vf_i \neq 0$, since they are at the center of their neighbors positions, while the tail agent failing with $-v_{max}$ takes many rounds to detect, since it should be moving towards its left neighbor to cause the contraction. In the mixed case, failed agents with positive or negative failure velocity cannot be detected until they pass the center point of their left and right neighbors. This leads to the following detection time observation, which illustrates there is only one potentially "bad" mimicking action which allows maintenance of both safety and progress and takes more than one round to detect. The other two failures violate either progress or safety immediately and lead to an immediate detection.

The observation is, for a reachable state $\mathbf{x}$, if $|F(\mathbf{x})| = 1$, let the id of the failed agent be i, and consider the three possibilities of $\mathbf{x}.vf_i = 0$, $\mathbf{x}.vf_i \in (0, v_{max}]$, and $\mathbf{x}.vf_i \in [-v_{max}, 0)$ corresponding to the range of $\text{sgn}(\mathbf{x}.vf_i)$. Then along a fail-free execution fragment starting from $\mathbf{x}$, for one of these choices of vf_i, the detection time is greater than 1, and for the other two, the detection time is 1.

Finally, Figure 5 shows the influence of multiple failures on detection time. For each choice of F and N, 50 simulations were run with initial states satisfying the spacing between each agent being chosen uniformly from the range $[r_s, r_s + 2r_f]$, and f random agents were failed in the initial round with one of $vf_i \in \{-v_{max}, 0, +v_{max}\}$. In this experiment, it is difficult to observe the relationship between average detection time and the number of agents, N, but this is due to the choice of random initial condition. The maximum time to detection increases as a function of f which supports the above detection time observation in the multiple failure case. In particular, the average (over N) maximum detection times for each of $f = 1$, $f = 2$, and $f = 3$ are 62, 139, and 203 rounds, respectively, and the conjecture predicts a linear increase in detection time as a function of f. However, we observe slightly more than this, so it is the case that the maximum detection time may be dependent upon the other failures.

4 Conclusion

This paper presents an algorithm for the safe flocking problem in spite of failures. It does so through self-stabilization when combined with a failure detector. Particularly, it establishes safety invariance and that eventually a strong flock is formed and a destination reached. Without the failure detector, the system would not be able to maintain safety as agents could collide, nor make progress to states satisfying flocking or the destination, since failed agents may diverge, causing their neighbors to follow and diverge as well.

References

1. Arora, A., Gouda, M.: Closure and convergence: A foundation of fault-tolerant computing. IEEE Trans. Softw. Eng. 19, 1015–1027 (1993)
2. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. J. ACM 43(2), 225–267 (1996)
3. Chandy, K.M., Lamport, L.: Distributed snapshots: determining global states of distributed systems. ACM Trans. Comput. Syst. 3(1), 63–75 (1985)
4. Dolev, S.: Self-stabilization. MIT Press, Cambridge, MA (2000)
5. Fax, J., Murray, R.: Information flow and cooperative control of vehicle formations. IEEE Trans. Autom. Control 49(9), 1465–1476 (Sep 2004)
6. Franceschelli, M., Egerstedt, M., Giua, A.: Motion probes for fault detection and recovery in networked control systems. In: American Control Conference, 2008. pp. 4358–4363 (Jun 2008)
7. Gazi, V., Passino, K.M.: Stability of a one-dimensional discrete-time asynchronous swarm. IEEE Trans. Syst., Man, Cybern. B 35(4), 834–841 (Aug 2005)
8. Jadbabaie, A., Lin, J., Morse, A.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Autom. Control 48(6), 988–1001 (Jun 2003)
9. Johnson, T.: Fault-Tolerant Distributed Cyber-Physical Systems: Two Case Studies. Master's thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (May 2010)
10. Johnson, T., Mitra, S.: Safe and stabilizing distributed flocking in spite of actuator faults. Tech. Rep. UILU-ENG-10-2204 (CRHC-10-02), University of Illinois at Urbana-Champaign, Urbana, IL (May 2010)
11. Okubo, A.: Dynamical aspects of animal grouping: Swarms, schools, flocks, and herds. Adv. Biophys. 22, 1–94 (1986)
12. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Trans. Autom. Control 51(3), 401–420 (Mar 2006)
13. Shaw, E.: Fish in schools. Natural History 84(8), 40–45 (1975)
14. Tsitsiklis, J., Bertsekas, D., Athans, M.: Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Trans. Autom. Control 31(9), 803–812 (Sep 1986)

A General Sequence Convergence Lemma and Proofs

Lemma 11. *Consider any infinite sequence of lexicographically ordered pairs* $\langle a_1, b_1\rangle$, $\ldots$, $\langle a_j, b_j\rangle$, $\ldots$ *where* a_j, $b_j \in \mathbb{R}_{\geq 0}$. *Suppose* $\exists c_1$, c_2, c_3, c_4, c_5, c_6 *such that* $c_1 > 0$, $c_2 > 0$, $c_3 > 0$, $c_4 \geq 0$, $c_5 \geq 0$, *and* $c_6 \geq 0$. *If* $\forall j$, *(i)* $a_{j+1} \leq a_j$ *(ii)* $a_{j+1} = a_j \wedge b_j > c_4$ *then* $b_{j+1} \leq b_j - c_1$ *(iii)* $a_{j+1} < a_j$ *then* $b_{j+1} \leq c_6$ *(iv)* $b_j \leq c_2 \wedge a_j > c_5$ *then* $a_{j+1} \leq \max\{0, a_j - c_3\}$ *Then,* $\exists t$ *such that* $\langle a_1, b_1\rangle$, $\ldots$, $\langle a_t, b_t\rangle$, $\langle a_{t+1}, b_{t+1}\rangle$, $\ldots$ *and* $\langle a_t, b_t\rangle = \langle a_{t+1}, b_{t+1}\rangle$, *where* $a_t \in A = [0, c_5]$ *and* $b_t \in B = [0, c_4]$.

Proof of Invariant 1 : The proof is by induction over the length of any execution of System. The base case follows by the assumption that the initial state satisfies $Safety$. For the inductive case, for each transition $a \in A$, we show if $\mathbf{x} \xrightarrow{a} \mathbf{x}' \wedge \mathbf{x} \in Safety_R$, then $\mathbf{x}' \in Safety_R$. The $\mathsf{fail}_i(v)$, $\mathsf{snapStart}_i$, $\mathsf{snapTerm}_i$, and $\mathsf{suspect}_i$ transitions do not modify any x_i or u_i, so $Safety_R(\mathbf{x}')$ For the update transition, the inductive hypothesis gives that $Safety_R$ is satisfied for the pre-state $\mathbf{x}$. For the remainder of the proof, let $l = \mathbf{x}.L_i$ (the unsuspected agent left of i), $r = \mathbf{x}.R_i$ (the unsuspected agent right of i), and $ll = \mathbf{x}.L_{\mathbf{x}.L_i}$ (the unsuspected agent left of l). If these variables change between $\mathbf{x}$ and $\mathbf{x}'$, the result follows by Lemma 2. The remainder of the proof is divided into two cases: the first case analyzes the spacing between two non-faulty agents, and the second case analyzes the spacing between any faulty agent and non-faulty agent, which reside in the same lane. All the following comes from Figure 1, Lines 32–37 and the inductive hypothesis.

For the first case showing the spacing between any two non-faulty agents, it is sufficient to show if $\forall i \in NF(\mathbf{x})$, $\mathbf{x}.u_i - \mathbf{x}.u_l \geq r_r$ and $\mathbf{x}.x_i - x_l \geq r_r$, then $\mathbf{x}'.u_i - \mathbf{x}'.u_l \geq r_r$. If i is any non-faulty middle agent, $\mathbf{x}'.u_i - \mathbf{x}'.u_l = \frac{\mathbf{x}.x_l - \mathbf{x}.x_{ll} + \mathbf{x}.x_r - \mathbf{x}.x_i}{2} \geq r_r$. If i is the non-faulty tail, $\mathbf{x}'.u_i - \mathbf{x}'.u_l = \frac{r_f + \mathbf{x}.x_l - \mathbf{x}.x_{ll}}{2} \geq r_r$. Since $0 \leq x_{H(\mathbf{x})}$, then by the inductive hypothesis, $\mathbf{x}'.u_{H(\mathbf{x}')} \leq \mathbf{x}.u_{H(\mathbf{x})}$. Cases when quantization changes any $\mathbf{x}'.u_i$ in Line 40 follow by similar analysis and are omitted for space. Thus, $Safety_R(\mathbf{x})$.

Next is the proof of the second case, that the spacing between any non-faulty and any faulty agent which reside in the same lane is at least r_r. For simplicity of presentation, assume we are dealing with a fail-free execution, under which case, the detection time until all failures are detected is k_d. If we were not dealing with a fail-free execution, just choose the maximum such k_d and pick v_{max} as in Theorem 1. For a failed agent j, $\mathbf{x}'.x_j = \mathbf{x}.x_j + \mathbf{x}.vf_j$ by Line 43. Given the assumption that $v_{max} \leq \frac{r_s - r_r}{2k_d}$, it is the case that at round k_d, $\mathbf{x}_d.x_j \leq \mathbf{x}.x_j + k_d v_{max} = \mathbf{x}.x_j + \frac{r_s - r_r}{2}$ where we considered the case for $\mathbf{x}.v_j > 0$ and the negative failure velocity case follows symmetrically. By assumption that any failure is detected by round k_d and by Lemma 1, any failed agent j and any non-failed agent i have moved towards one another by at most $2k_d v_{max}$, and thus $\mathbf{x}_d.x_j - \mathbf{x}_d.x_i \leq 2k_d v_{max} = r_s - r_r$.

This implies at least $Safety_R(\mathbf{x}_m)$ for any states $\mathbf{x}_m$ in any state in the execution between $\mathbf{x}$ and $\mathbf{x}_d$. It remains to be established that $Safety_R(\mathbf{x}'_d)$ for a state $\mathbf{x}'_d$ reachable from state $\mathbf{x}_d$. By the detection time assumption, any agent i will have $j \in \mathbf{x}_d.Suspected_i$, which changes $L_S(\mathbf{x}_d)$ and $R_S(\mathbf{x}_d)$, but now apply

Lemma 2, which shows there is at least r_r space between i and j. Finally, by Figure 1, Line 28, $\mathbf{x}'_d.lane_i \neq \mathbf{x}_d.lane_j$, since $N_L \geq 2$, and by Assumption (b) of Theorem 1, $Safety_R(\mathbf{x}'_d)$. □

Proof of Theorem 2 : This follows from Lemma 4, the $O(N)$ termination time of the snapshot algorithm, and from Lemma 11 by instantiating (a) $c_1 = v_{min}$, (b) $c_2 = (N-1)\frac{\delta}{4}$, (c) $c_3 = \frac{\delta}{2}$, (d) $c_4 = \gamma$, (e) $c_5 = \beta$, and (f) $c_6 = (N-1)\frac{\delta}{2}$. □

Proof of Lemma 8 : Consider a fail-free execution α_f which begins with a state with a single failure, and a fail-free execution α_n which begins with a state without any failures. Let the initial state $\mathbf{x}$ of both these executions be the same (except let the head agent be failed with zero failure velocity for α_f) and satisfy $\mathbf{x} \notin Terminal$ and $\mathbf{x} \notin Flocks_S$. In both executions, assume that any time Line 44 of Figure 1 is executed, the nondeterministic choice results in v_{min}. We know that the computed target for the head node is non-zero only if the state of the whole system satisfies $Flocks_S$. Lemma 4 implies that $\mathbf{x}'$ is a constant c number of rounds away from $\mathbf{x}$ in each of α_f and α_n where c is $O(N)$, and only once $\mathbf{x}' \in NBM$ can it be guaranteed that $\mathbf{x}' \in Flocks_S$. Once $\mathbf{x}' \in Flocks_S$, at some state $\mathbf{x}''$, which is a constant d number of rounds from $\mathbf{x}'$ in each of α_f and α_n, will $u_{H(\mathbf{x}'')} \neq 0$, where d is $O(N)$ by the $O(N)$ termination of the snapshot algorithm Figure 1, Line 34 Thus, α_f and α_n are indistinguishable up to state $\mathbf{x}'$ and by Lemma 4, $\mathbf{x}'$ is a constant a number of rounds from $\mathbf{x}$ where a is $O(N)$. □

Proof of Lemma 9 : Suppose $\exists i, j$ such that $j \in \mathbf{x}.Suspected_i$, then the precondition for $\mathsf{suspect}_i$ must have been satisfied at some round k_s in the past when j was added to $Suspected_i$. Let $\mathbf{x}_s$ correspond to the state at round k_s and $\mathbf{x}'_s$ be the subsequent state in the execution. At the round prior to k_s, there are two cases based the computation of u_j in Figure 1, Line 39 for some $j \notin \mathbf{x}_{k_s-1}.Suspected_i$.

The first case is when the quantization constraint $|\mathbf{x}_s.x_j - \mathbf{x}_{s,T}.u_j| \leq \beta$ was not satisfied in Figure 1, Line 39, so Agent_j applies a velocity in the direction of $\mathrm{sgn}(u_j - x_j)$. If $\mathrm{sgn}(\mathbf{x}'_s.x_j - \mathbf{x}_s.x_j) \neq \mathrm{sgn}(\mathbf{x}_s.u_j - \mathbf{x}_s.x_j)$, then Agent_j moved in the wrong direction, since it computed a move $\mathbf{x}_s.u_j$ but in actuality applied a velocity that caused it to move away from $\mathbf{x}_s.u_j$ instead of towards it. This is possible only if $\mathrm{sgn}(\mathbf{x}'_s.u_j - \mathbf{x}'_s.x_j) \neq \mathrm{sgn}(\mathbf{x}_s.u_j - \mathbf{x}_s.x_j)$, implying that $\mathbf{x}_s.vf_j \neq 0$, and thus $\mathbf{x}_s.failed_j = true$.

The second case is when the quantization constraint $|\mathbf{x}_s.x_j - \mathbf{x}_{s,T}.u_j| \leq \beta$ was satisfied in Figure 1, Line 39, so $|\mathbf{x}_s.x_j - \mathbf{x}_s.u_j| = 0$ should have been observed, but instead it was observed that Agent_j performed a move, such that $|\mathbf{x}'_s.x_j - \mathbf{x}_s.x_j| \neq 0$. This implies that $\mathbf{x}_s.failed_j = true$ since the only way $|\mathbf{x}'_s.x_j - \mathbf{x}_s.x_j| \neq 0$ is if for $\mathbf{x}_s.vf_j \neq 0$, $\mathbf{x}'_s.x_j = \mathbf{x}_s.x_j + \mathbf{x}_s.vf_j$. □

Proof of Lemma 10 : For a $\mathsf{suspect}_i$ transition to be taken, the precondition at Line 8 of Figure 1 must satisfy that $j \notin \mathbf{x}.Suspected_i$, and that either (a) $|\mathbf{x}.xo_j - \mathbf{x}.uo_j| \leq \beta$ and $|\mathbf{x}.x_j - \mathbf{x}.uo_j| \neq 0$, or (b) $|\mathbf{x}.xo_j - \mathbf{x}.uo_j| > \beta$ and $\mathrm{sgn}(\mathbf{x}.x_j - \mathbf{x}.xo_j) \neq \mathrm{sgn}(\mathbf{x}.uo_j - \mathbf{x}.xo_j)$. These are the two hypotheses of the lemma and thus the result follows that the $\mathsf{suspect}_i$ transition is enabled. □