Safe and Stabilizing Distributed Flocking in Spite of Actuator Faults[*]

Taylor Johnson and Sayan Mitra

Coordinated Science Laboratory

University of Illinois at Urbana-Champaign

Urbana, IL 61801

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 an affected agent to be stuck moving 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 that illustrate the algorithm in operation and the effects of failures on progress towards flocking.

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 [1, 2], for example). 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 [3, 4, 5, 6, 7] 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 [5], 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

[*]University of Illinois at Urbana-Champaign, Center for Reliable and High-Performance Computing, Technical Report No. UILU-ENG-10-2204 (CRHC-10-02)

collision avoidance, as illustrated by an error—a forgotten case for the special dynamics of the rightmost (N^{th}) agent—that we found in the inductive proof of safety in [3]. To fix the error, the algorithm from [3] requires the modification presented later in this paper (Figure 7, Line 31). The model obtained with rectangular dynamics overapproximates any behavior that can be obtained with double integrator dynamics with bounded acceleration. Our algorithm combines the corrected algorithm from [3] with Chandy-Lamport's global snapshot algorithm [8]. The key idea is that each agent periodically computes its target based on messages received from its neighbors, then moves toward this target with some arbitrary but bounded velocity. The targets are computed such that the agents preserve safe separation and 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 [8]. Once this is detected, the detecting agent moves toward the destination, sacrificing the strong flock in favor of making progress toward the goal, but still preserving the weak flock.

Unlike the algorithms in [3, 4, 5, 6] 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.

We allow agents to be affected by *actuator faults*. This physically corresponds to, for example, an agent's motors being stuck at an input voltage or a control surface becoming immobile. 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 not safety, actuator faults can also violate safety. A faulty agent has to be detected (and possibly avoided) by the non-faulty agents to avoid collisions and ensure progress. In this paper, we assume that after an actuator fault, a faulty agent continues to communicate and compute, but its actuators continue to move with the arbitrary but constant failure velocity.

Some attention has been given to failure detection in flocking such as [10], which works with a similar model of actuator faults. While [10] uses the therein developed *motion probes* in failure detection scenarios, no bounds are stated on detection time. Instead, convergence was ensured assuming that failure detection had occurred within some bounded time, while our work states an $O(N)$ detection time bound, where N is the number of agents.

Our flocking algorithm determines *only* the direction in which an agent should move, based on the positions of adjacent agents. The speed with which an agent moves is chosen nondeterministically over a range, making the algorithm implementation independent with respect to the lower-level motion controller. The intuition behind failure detection is based on this nondeterministic choice, and simply is to observe that an agent has moved in the wrong direction, or moved when it should not have. 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, and thus we establish that no such failure detector exists. But, under a restricted class of actuator faults, 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. This requires non-faulty agents to be able to avoid faulty ones. In one dimension (such as on highways), this is possible if there are multiple *lanes*. To avoid collisions and ensure progress, non-faulty agents avoid faulty ones by moving to a lane with no nearby faulty agents.

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

(a) Formal introduction of the notion of actuator faults and stabilization in the face of such faults.

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

Paper organization In Section 2 we introduce preliminary notation and a formal model of the complete system of agents and the properties to be analyzed. Then in Section 3, analysis of the system is presented. First basic system properties are established in Section 3.1. Then in Section 3.2, safety is established in spite of actuator faults, followed by establishing progress of the fail-free executions of the system toward a flock and goal in Section 3.3. Section 3.4 presents the lower-bound on failure detection, and analysis of the failure detection scheme incorporated in our algorithm. Next, Section 3.5 presents simulation studies of the system. Finally in Section 4, we present conclusions from this study and future work.

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. First are some preliminary definitions.

2.1 Preliminaries

Denote the sets of natural, real, and nonnegative real numbers by $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$, respectively. For $K \in \mathbb{N}$, $[K] \triangleq \{1, \ldots, K\}$ and for a set S $S_{\perp} \triangleq S \cup \{\perp\}$. For a variable x, its type is denoted by $type(x)$ and it is the set of values that it can take. 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 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 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 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 [11]. In this paper, we model actuator faults 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 $\mathcal{A}$ along execution fragments without fail-transitions, and (b) from every reachable state of $\mathcal{A}$ (including states reached via fail transitions), every *fail*-free execution fragment eventually reaches G.

2.2 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. Refer to Figures 1 and 2 for clarity, which show the system at a potential state with faulty agents, as well as a potential terminal configuration. Also see Figures 3 and 4, which show the system making progress (reaching the origin as a flock), and note that agents 1 through 5 move to lane 2 around round 375 to avoid the faulty agent 6. We assume synchrony and the communication graph is complete, regardless of lanes. That is, agents have synchronized clocks, message delays are bounded, and computations are instantaneous. At each round, each agent exchanges messages bearing state information with its neighbors, and note that this means agents in different lanes communicate. The neighbors of an agent are the other agents that are sufficiently close to the agent, regardless of lane. 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.

Figure 1: System at state $\mathbf{x}$ for $N = 8$, $\bar{F}(\mathbf{x}) = \{2, 3, 5, 6, 8\}$, $F(\mathbf{x}) = \{1, 4, 7\}$. Faulty actuator velocities are labeled vf_i. Non-faulty agents have avoided faulty agents by changing lanes. Note that $L(\mathbf{x}, 6) = 5$. Also, if $4 \in \mathit{Suspected}_6$, then $L_S(\mathbf{x}, 6) = L(6) = 5$, else $L_S(\mathbf{x}, 6) = 4$. Assuming $S(\mathbf{x}) = F(\mathbf{x})$, $\mathit{Flock}_W(\mathbf{x})$, but $\neg \mathit{Flock}_S(\mathbf{x})$, since $|\mathbf{x}.x_6 - \mathbf{x}.x_5 - r_f| \leq \delta$ (and not $\delta/2$).

Figure 2: System potential $\mathit{Terminal}$ configuration, having achieved $\mathit{Flock}_S(\mathbf{x})$ and reached the goal (the origin). Observe that faulty agents 4 and 7 with nonzero velocity have diverged and that non-faulty agents do not necessarily end on the same lane. Faulty agent 1 with zero velocity remains stationary, but does not prevent formation of the flock due to proper left L_S and right R_S neighbor selection.

Let $\mathit{ID} \triangleq [N]$ be the set of unique agent identifiers and $\mathit{LD} \triangleq [N_L]$ 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_c: communications distance,

(d) r_f: desired maximum inter-agent gap which defines a flock,

(e) δ: flocking tolerance parameter—that is, the maximum deviation from r_f agents may be spaced and constitute a *flock*,

(f) β: quantization parameter, used to prevent agents from moving if the algorithm decides too small a movement so that eventually the algorithm terminates, and

(g) v_{min}, v_{max}: minimum and maximum velocities.

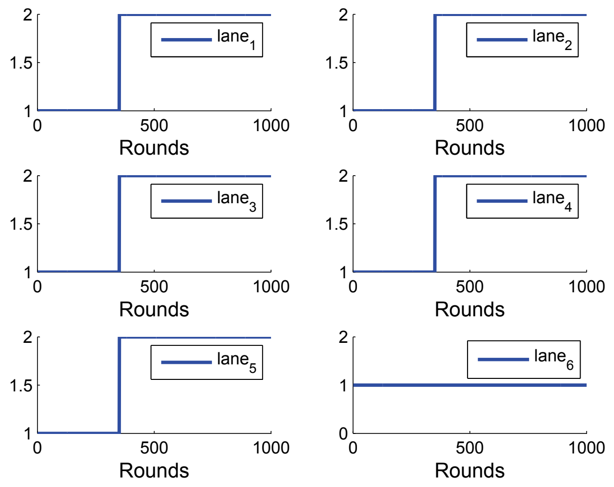

Figure 3: System progressing: eventually the agents have formed a flock and the faulty agent 6 with nonzero velocity has diverged.

Figure 4: System progressing: illustrating that non-faulty agents have avoided faulty agent 6 by changing lanes.

State Variables. The discrete transition system corresponding to Agent$_i$ has the following *private* variables, where initial values of the variables are shown in Figure 5 using the ':=' notation.

```
variables
    $x, xo : \mathbb{R}$
    $u, uo : \mathbb{R} := x$
    $lane : ID_L := 1$
    $sr : \mathbb{B} := false$
    $gsf : \mathbb{B} := false$
    $failed : \mathbb{B} := false$
    $vf : \mathbb{R}_\perp := \perp$
    $Suspected : \texttt{Set}[ID_\perp] := \{\}$
    $Nbrs : \texttt{Set}[ID] := Nbrs(\mathbf{x}, i)$
    $L : Nbrs := L_S(\mathbf{x}, i)$
    $R : Nbrs := R_S(\mathbf{x}, i)$
```

Figure 5: Variables of Agent$_i$.

(a) gsf: indicates whether the stable predicate detected by the global snapshot is satisfied or not,

(b) sr: indicates whether the global snapshot algorithm has been initiated,

(c) $failed$: indicates whether or not agent i has failed,

(d) vf: velocity with which agent i has failed, and

(e) $Nbrs$: set of identifiers of agents that are neighbors of agent i, at the pre-state $\mathbf{x}$ of any transition, so it is $Nbrs(\mathbf{x}, i)$, and

(f) 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 (also see Figure 6):

(a) x and xo: current position and position from the previous round of agent i,

(b) u and uo: target position and target position from the previous round of agent i,

(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 $\mathsf{Agent_i}$ 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 $\mathsf{Agent_i}$ by $\mathbf{x}.x_i$, $\mathbf{x}.u_i$, etc. When necessary to distinguish i's knowledge of neighbor j's state variables for a state $\mathbf{x}_k$, the notation $\mathbf{x}_k.x_{i,j}$ will be used to indicate this is j's position $\mathbf{x}_{k-1}.x_j$ from the perspective of i at round k.

Figure 6: Interaction between a pair of neighboring agents is modeled with shared variables x, xo, u, uo, $lane$, and $Suspected$.

Actuator Faults and Failure Detection. The failure of agent i's actuators 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 fault causes the affected agent to move forever with a constant but arbitrary *failure velocity*. At state $\mathbf{x}$, $F(\mathbf{x})$ and $\bar{F}(\mathbf{x})$ denote the sets of faulty and non-faulty agent identifiers, respectively.

Agents do not have any direct information regarding the failure of other agents' actuators (i.e., agent i cannot read $failed_j$). Agents 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 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. At state $\mathbf{x}$, $\mathsf{Agent_i}$ where $\mathbf{x}.failed_i = true$ is a *faulty* agent, otherwise it is a *non-faulty* agent. $\mathsf{Agent_j}$ is said to be *suspected* if some agent i suspects it, otherwise it is *unsuspected*. At state $\mathbf{x}$, if $\exists i \in ID$ such that $i \in \mathbf{x}.Suspected_j$, then i is called an agent *suspected by* j. Denote the sets of suspected and unsuspected agents by $S(\mathbf{x})$ and $\bar{S}(\mathbf{x})$, 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 3.4, we will discuss specific conditions under which k_d is in fact finite and then give upper and lower bounds for it. The failure detection strategy used by our flocking algorithm 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 7, Lines 6–8, j is added to $Suspected_i$. This precondition checks that either j moved when it should not have (due to being quantized), or if j should have moved, that j moved in the wrong direction away from its computed target u_j. The rationale behind this condition will become clear as we discuss the flocking algorithm.

```
$\mathsf{fail}_i(v)$, $|v| \leq v_{max}$
pre $\neg$ failed
eff failed := $true$; vf := $v$

$\mathsf{suspect}_i(j)$, $j \in Nbrs$
pre $j \notin$ Suspected $\wedge$ (if $|xo_j - uo_j| \geq \beta$
   then $\mathrm{sgn}\,(x_j - xo_j) \neq \mathrm{sgn}\,(uo_j - xo_j)$
   else $|x_j - uo_j| \neq 0$)
eff Suspected := Suspected $\cup \{j\}$

$\mathsf{snapStart}_i$
pre $L = \perp \wedge \neg sr$
eff $sr := true$ // global snapshot invoked

$\mathsf{snapEnd}_i(GS)$, $GS \in \{false, true\}$
eff gsf := GS; // global snapshot returns
   $sr := false$

$\mathsf{update}_i$
eff uo := u; xo := x
   for each $j \in$ Nbrs, Suspected := Suspected $\cup$ $Suspected_j$
$Mitigate$:
   if $\neg$ failed $\wedge$ ($\exists\, s \in$ Suspected : $lane_s = lane$)
      $\wedge$ ($\exists\, \mathcal{L} \in LD : \forall\, j \in Nbrs$, ($lane_j = \mathcal{L} \Rightarrow$
      $x_j \notin [x - r_s - 2v_{max}, x + r_s + 2v_{max}]$))
      then lane := $\mathcal{L}$ fi
$Target$:
   if $L = \perp \wedge gsf$ then $u := x - \min\{x, \delta/2\}$;
      gsf := $false$
   elseif $L = \perp$ then $u := x$
   elseif $R = \perp$ then $u := (x_L + x + r_f)/2$
   else $u := (x_L + x_R)/2$ fi
$Quant$: if $|u - x| < \beta$ then $u := x$ fi
$Move$: if failed then $x := x + vf$
   else $x := x + \mathrm{sgn}\,(x - u)$ choose $[v_{min}, v_{max}]$ fi
```

Figure 7: Agent$_i$'s transitions: failure detection, global snapshots, and target updates.

Neighbors. Agent$_i$ is said to be a *neighbor* of a different Agent$_j$ at state $\mathbf{x}$ if and only if $|\mathbf{x}.x_i - \mathbf{x}.x_j| \leq r_c$. The set of identifiers of all neighbors of Agent$_i$ at state $\mathbf{x}$ is denoted by

$$Nbrs(\mathbf{x}, i) \triangleq \{j \in ID : i \neq j \wedge |\mathbf{x}.x_i - \mathbf{x}.x_j| \leq r_c\}.$$

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 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 Agent$_i$ at state $\mathbf{x}$, or $\perp$ if no such agents exist. $L_S(\mathbf{x}, i)$ and $R_S(\mathbf{x}, i)$ take values from $\{\perp\} \cup Nbrs(\mathbf{x}, i) \setminus \mathbf{x}.Suspected_i$, and thus, $L_S(\mathbf{x}, i)$ (and $R_S(\mathbf{x}, i)$) is the identifier of nearest non-suspected agent positioned to the left (right) of i on the real line. Observe that this is regardless of lane. An unsuspected 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 Agent$_i$ without an unsuspected left neighbor is the *head agent*, and is denoted by the singleton $H(\mathbf{x})$. If Agent$_i$ is unsuspected, is not the head, and does not have an unsuspected right neighbor, it is the *tail agent* and is denoted by the singleton $T(\mathbf{x})$. Let $RMids(\mathbf{x}) \triangleq Mids(\mathbf{x}) \setminus \{R(\mathbf{x}, H(\mathbf{x}))\}$. Observe that the following holds,

$$\begin{aligned} H(\mathbf{x}) &\triangleq \min \bar{S}(\mathbf{x}), \text{and} \\ T(\mathbf{x}) &\triangleq \max \bar{S}(\mathbf{x}). \end{aligned}$$

Flocking Algorithm. The state transitions are fails, $\mathsf{snapStart}$s, $\mathsf{snapEnd}s$, $\mathsf{suspect}$s, and updates. A $\mathsf{fail}_i(v)$ transition where $i \in NF(\mathbf{x})$ for a state $\mathbf{x}$ models the permanent failure of Agent$_i$. As a result of this transition, $failed_i$ is set to $true$ and vf_i is set to v. This causes Agent$_i$ to move forever with velocity v. Assume that $|v| \leq v_{max}$, which is reasonable due to actuation constraints.

The distributed flocking algorithm executed at 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 Agent$_i$.

The $\mathsf{snapStart}$ and $\mathsf{snapEnd}$ transitions model the periodic initialization and termination of a distributed global snapshot protocol—such as Chandy and Lamport's snapshot algorithm [8]—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 $\mathsf{snapStart}_i$ transition is performed periodically by the head agent when the precondition is enabled. If the global predicate holds, then $\mathsf{snapEnd}(true)$ occurs, otherwise $\mathsf{snapEnd}(false)$ occurs. Chandy-Lamport's algorithm can be applied since (a) we are detecting a stable predicate, (b) the communications graph is complete, and (c) the stable predicate being detected is reachable. Thus, we assume that in any infinite execution, a $\mathsf{snapEnd}_i$ transition occurs within $O(N)$ rounds from the occurrence of the corresponding $\mathsf{snapStart}_i$ transition.

The update transition models the evolution of all (faulty and non-faulty) agents over a synchronous round. It is composed of four subroutines: $\mathit{Mitigate}$, Target, Quant, and Move, which are executed in this sequence for updating the state of System. The entire update is instantaneous and atomic; the subroutines are used for clarity of presentation. 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 $\mathit{Mitigate}$, Target, Quant, and Move as $\mathbf{x}_M$, $\mathbf{x}_T$, $\mathbf{x}_Q$, and $\mathbf{x}_V$, respectively. That is, $\mathbf{x}_M \triangleq \mathit{Mitigate}(\mathbf{x})$, $\mathbf{x}_T \triangleq \mathit{Target}(\mathbf{x}_M)$, etc., and note $\mathbf{x}' = \mathbf{x}_V = \mathit{Move}(\mathbf{x}_Q)$.

$\mathit{Mitigate}$ is executed by non-faulty agents and may cause them to change lanes, thus restoring safety and progress properties that may be reduced or violated by failures. Target determines a new target to move toward. 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 7, Line 32). 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 7, Line 31). The head agent periodically invokes a global snapshot and attempts to detect a certain stable global predicate Flocks_S (defined below). If this predicate is detected, then the head agent moves towards the goal (Figure 7, Line 29), otherwise it does not change its target u from its current position x. 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. Note that Move abstractly captures the physical evolution of the system over a round; that is, it is the time-abstract transition corresponding to physical evolution over an interval of time.

2.3 Model as a Discrete-Time Switched Linear System

The following is a view of the system as a discrete-time switched system and displays that failures can be modeled as a combination of an additive affine control and a switch to another system matrix.

Discrete-time switched systems can be described as $x[k+1] = f_p(x[k])$ in general where $x \in \mathbb{R}^N$, $p \in \mathcal{P}$ for some index set $\mathcal{P}$, such as $\mathcal{P} = \{1, 2, \ldots, m\}$, or as $x[k+1] = A_p x[k]$ for linear discrete-time switched systems [12]. For the following, assume that Figure 7, Line 35 is deleted and replaced with $x := u$. This deletion removes the nondeterministic choice of velocity with which to set position x, and instead sets it to be the computed control value u. This nondeterministic choice can be modeled through the use of a time-varying system matrix $A[k]$ as in [3], but we omit it for simplicity of presentation.

The effect of an update transition on the position variables of all agents in System can be represented by the difference equation $x[k+1] = A_p x[k] + b_p$ where for a state $\mathbf{x}_k$ at round k,

$$x[k] = \begin{pmatrix} \mathbf{x}_k.x_{H(\mathbf{x}_k)} \\ \mathbf{x}_k.x_{x.R_{H(\mathbf{x}_k)}} \\ \vdots \\ \mathbf{x}_k.x_{T(\mathbf{x}_k)} \end{pmatrix},$$

$$A_p = \begin{pmatrix} a_{1,1} & 0 & 0 & 0 & 0 & 0 & \dots \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & \dots \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & \dots \\ 0 & 0 & 0 & a_{i,i-1} & a_{i,i} & a_{i,i+1} & \dots \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \dots \\ 0 & 0 & 0 & 0 & 0 & a_{N,N-1} & a_{N,N} \end{pmatrix}, \text{ and}$$

$$b_p = \begin{pmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}.$$

The following are the family of matrices A_p and vectors b_p that are switched among based on the state of System; refer to Figure 7 for the following referenced line numbers. From Line 29, for $H(\mathbf{x}_k)$, if $Flock_S(\mathbf{x}_k)$, then either (a) if $\mathbf{x}_k.x_{H(\mathbf{x}_k)} \geq \delta$, then $a_{1,1} = 1$ and $b_1 = -\frac{\delta}{2}$, otherwise (b) $a_{1,1} = 0$ and $b_1 = 0$. From Line 30, if $\neg Flock_S(\mathbf{x}_k)$, then $a_{1,1} = 1$ and $b_1 = 0$. From Line 32, for $i \in Mids(\mathbf{x}_k)$, $a_{i,i} = 0$, $a_{i,i-1} = \frac{1}{2}$, $a_{i,i+1} = \frac{1}{2}$, and $b_i = 0$. Finally, from Line 31, for $T(\mathbf{x}_k)$, $a_{N,N-1} = \frac{1}{2}$, $a_{N,N} = \frac{1}{2}$, and $b_N = \frac{r_f}{2}$.

Next, all coefficients in the matrix can change due to the quantization law in Line 33. If the conditional on Line 33 is satisfied for agent $i \in Mids(\mathbf{x}_k)$, then $a_{i,i} = 1$, $a_{i,\mathbf{x}_k.L_i} = 0$, $a_{i,\mathbf{x}_k.R_i} = 0$, and $b_i = 0$, for agent $i = H(\mathbf{x}_k)$, then $a_{i,i} = 1$ and $b_i = 0$, and for agent $i = T(\mathbf{x}_k)$, then $a_{i,\mathbf{x}_k.L_i} = 0$, $a_{i,i} = 1$, and $b_i = 0$.

Failures also cause a switch of system matrices. The actuator stuck-at failures being modeled are representative of an additive error term in the b_p vector [13]. From Line 34, for $i \in Mids(\mathbf{x}_k)$, $a_{i,i} = 1$, $a_{i,\mathbf{x}_k.L_i} = 0$, $a_{i,\mathbf{x}_k.R_i} = 0$, and $b_i = \mathbf{x}_k.vf_i$, for $i = H(\mathbf{x}_k)$, $a_{i,i} = 1$ and $b_1 = \mathbf{x}_k.vf_{H(\mathbf{x}_k)}$, and for $i = T(\mathbf{x}_k)$, $a_{N,N-1} = 0$, $a_{N,N} = 1$, and $b_N = \mathbf{x}_k.vf_{T(\mathbf{x}_k)}$.

2.4 Key Predicates

We now define a set of predicates on the state space of System that capture the 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, $Safety(\mathbf{x}) \triangleq \forall i, j \in ID, i \neq j, \mathbf{x}.lane_i = \mathbf{x}.lane_j \implies |\mathbf{x}.x_i - \mathbf{x}.x_j| \geq r_s$. When failures occur, a reduced inter-agent gap of r_r will be guaranteed. We call this weaker property *reduced safety*: $Safety_R(\mathbf{x}) \triangleq \forall i \in \bar{F}(\mathbf{x}), \forall j \in ID, i \neq j, \mathbf{x}.lane_i = \mathbf{x}.lane_j \implies |\mathbf{x}.x_i - \mathbf{x}.x_j| \geq r_r$.

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 the left neighbor. Formally, $Flock(\mathbf{x}, \epsilon) \triangleq \forall\ i \in \bar{S}(\mathbf{x})$, $L_S(\mathbf{x}, i) \neq \perp$, $\left|\mathbf{x}.x_i - \mathbf{x}.x_{L_S(\mathbf{x},i)} - r_f\right| \leq \epsilon$. 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 $Flock_W(\mathbf{x}) \triangleq Flock(\mathbf{x}, \delta)$, and $Flock_S(\mathbf{x}) \triangleq Flock(\mathbf{x}, \frac{\delta}{2})$, 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, because their computed targets are less than β (quantization constant) away from their current positions. $NBM(\mathbf{x}) \triangleq \forall i \in \bar{F}(\mathbf{x}), L_S(\mathbf{x}, i) \neq \perp, |\mathbf{x}_T.u_i - \mathbf{x}.x_i| \leq \beta$, 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), that is, $Goal(\mathbf{x}) \triangleq \mathbf{x}.x_{H(\mathbf{x})} \in [0, \beta]$. Finally, a state satisfies the $Terminal$ predicate if it satisfies both $Goal$ and NBM.

An outline of the properties to be introduced is presented in Figure 8. In this figure, start states Q_0 at least satisfy $Safety$. Fail-free executions are represented by lines with arrows labeled α_{ff}. $Safety$ is invariant along fail-free executions. Eventually NBM—and thus also $Flock_W$ and $Flock_S$—is satisfied along any fail-free execution, upon which after termination of the global snapshot algorithm, the head agent may move towards states satisfying $Goal$ causing $Flock_S$ to no longer be satisfied, while $Flock_W$ remains stable. However, along executions with failures, represented by the red line with an arrow labeled α_f, $Safety$ is not necessarily upheld, but $Safety_R$ is invariant when combined with a failure detector, whose action is represented by the green line with an arrow labeled α_{fd}. Upon this detection, any fail-free execution is then guaranteed to reach states satisfying NBM and also eventually $Goal$.

Figure 8: Set view of desired properties of System.

3 Analysis

The main result of the paper (Theorem 3.1) is that the algorithm in Figure 7 achieves safe flocking in spite of failures provided: (a) there exists a failure detector that detects actuator faults *sufficiently fast*, and (b) there is *enough room* some lane such that each non-faulty agent can 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 3.4, 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 3.1 *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 $\alpha = \mathbf{x}_0, \ldots, \mathbf{x}_p, \mathbf{x}_{p+1}$ 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. Suppose either*

(a) $N_L > f$, or

(b) *$N_L \leq f$ and along α_{ff}, $\forall \mathbf{x} \in \alpha_{ff}$, $\exists \mathcal{L} \in LD$ such that $\forall i \in \bar{F}(\mathbf{x})$, $\forall j \in F(\mathbf{x})$, $\mathbf{x}.lane_j \neq \mathcal{L}$ 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 prove a sequence of lemmas that culminate in Theorem 3.1. Under the assumptions and analysis of this section, the following relationships are satisfied: $\mathit{NBM} \subset \mathit{Flock}_S \subset \mathit{Flock}_W \subset \mathit{Safety} \subset \mathit{Safety}_R$. We begin with some assumptions.

Assumptions. Except where noted in Section 3.4, the remainder of the paper utilizes the assumptions of Theorem 3.1. Additionally, these assumptions are required throughout the paper:

(a) $N_L \geq 2$: there are at least 2 lanes,

(b) $r_r < r_s < r_f < r_c$: 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 r_f, all of which are less than the communication distance r_c,

(c) $\frac{\delta}{2} < r_c$,

(d) $0 < v_{min} \leq v_{max} \leq \beta \leq \delta/(4N)$, and

(e) the communication graph of the non-faulty agents is always connected, so the graph of non-faulty agents cannot partition.

Assumption (a) allows the safety and progress properties to be maintained in spite of failures by allowing agents to move among a set of N_L lanes, preventing collisions of failed and non-failed agents and allowing non-failed agents to pass failed agents which are not moving in the direction of the goal. Assumption (b) states the desired inter-agent spacing r_f is strictly greater than these safety margins and strictly less than the communications radius r_c, as well as that the safety and reduced safety margins are smaller than these. Assumption (c) prevents the agent nearest to the goal from moving beyond the communications radius of any right agent it is adjacent to, that is, it prevents disconnection of the graph of neighbors. Assumption (d) 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 $\mathit{NBM} \subseteq \mathit{Flock}_S$. 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. Assumption (e) is a natural assumption indicating there is a single network of agents. Note that this is weaker than the network being complete. It further states that failures do not cause the graph of non-faulty neighbors to partition.

Finally, assume that in all start states $\mathbf{x} \in Q_0$ of System, $Safety(\mathbf{x}) \wedge \mathbf{x}.x_{H(\mathbf{x})} \geq 0$.

3.1 Basic Analysis

The following lemma ensures that the set of neighbors of an agent is well defined and matches the definition of $Nbrs(\mathbf{x}, i)$ for agent i at the pre-state $\mathbf{x}$ of any transition. It follows by observing that only update transitions modify $Nbrs(\mathbf{x}, i)$.

Lemma 3.2 *For any reachable state $\mathbf{x}$ such that for all $i \in \bar{F}(\mathbf{x})$, $Nbrs(\mathbf{x}, i) = \mathbf{x}.Nbrs_i$. For any agent $i \in ID$, for a state $\mathbf{x}'$ such that $\mathbf{x} \overset{a}{\rightarrow} \mathbf{x}'$ for $a \in A \setminus \{\mathsf{update}\}$, $Nbrs(\mathbf{x}', i) = \mathbf{x}'.Nbrs_i$ and $Nbrs(\mathbf{x}, i) = \mathbf{x}.Nbrs_i$.*

The next lemma states that if neighbors change, then they do so symmetrically. This is used to establish safety upon agents no longer relying on suspected agents for target computation.

Lemma 3.3 *For any reachable state $\mathbf{x}$ such that $\mathbf{x} \overset{a}{\rightarrow} \mathbf{x}'$ for any $a \in A$, $\forall i, j \in ID$, if $\mathbf{x}.L_i \neq j$ and $\mathbf{x}'.L_i = j$, then $\mathbf{x}'.R_j = i$.*

Proof: Fix i and j and observe that only the suspect or update action changes $L_S(\mathbf{x}, i)$ or $R_S(\mathbf{x}, j)$ by changing either the positions of agents x_i or the sets of suspected agents. By Lemma 3.2, we consider L and R. There are two cases when $\mathbf{x}.L_i \neq \mathbf{x}'.L_i = j$. The first is upon agents that were not neighbors at $\mathbf{x}$ becoming neighbors at $\mathbf{x}'$, that is, $j \notin \mathbf{x}.Nbrs_i$ and $j \in \mathbf{x}'.Nbrs_i$. This is only possible due to the update action since no other action changes x_i. By definition of neighbor, also $i \notin \mathbf{x}.Nbrs_j$ and $i \in \mathbf{x}'.Nbrs_j$. By the symmetric definitions of $L_S(\mathbf{x}', i)$ and $R_S(\mathbf{x}', j)$, we have $\mathbf{x}'.R_j = i$.

The second case is when agents i and j were neighbors at $\mathbf{x}$, so $j \in \mathbf{x}.Nbrs_i$ and $i \in \mathbf{x}.Nbrs_j$, but now have at least one suspected agent f where $i > f > j$ between them and $f \in \mathbf{x}.Nbrs_i \cap \mathbf{x}.Nbrs_j$. This is possible due to the suspect or update transitions. Prior to suspecting that f is failed, no change of $L_S(\mathbf{x}, i)$ and $R_S(\mathbf{x}, j)$ occurs by definition, implying that for the hypothesis of the lemma to be satisfied, $\mathbf{x}'$ must be a state where $f \in \mathbf{x}'.Suspected_i \cap \mathbf{x}'.Suspected_j$, since i and j both use the same suspect action at Figure 7, Line 5. In this case, the symmetric switch occurs by definition of $L_S(\mathbf{x}, i)$ and $R_S(\mathbf{x}, j)$, we have $\mathbf{x}'.R_j = i$. Otherwise, $f \notin \mathbf{x}'.Suspected_i \cap \mathbf{x}'.Suspected_j$ and a contradiction that $\mathbf{x}.L_i \neq \mathbf{x}'.L_i$ occurs. ■

Next, assuming there are no failures, Lemma 3.4 shows that Agent_i's knowledge about the identities of its neighbors is true to its actual identities and remains so in all reachable states. This would be an invariant, but can be violated with failures.

Lemma 3.4 *If there are no failures, in any reachable state* $\mathbf{x}$*, for all* $i \in ID$*,*

$$\mathbf{x}.left_i = L_S(\mathbf{x}, i) \quad = \quad \begin{cases} \bot & \text{if } i = 1, \\ i - 1 & i \in \{2, \dots, N-1\}, \text{ and} \\ N - 1 & i = N. \end{cases}$$

$$\mathbf{x}.right_i = R_S(\mathbf{x}, i) \quad = \quad \begin{cases} 2 & \text{if } i = 1, \\ i + 1 & i \in \{2, \dots, N-1\}, \text{ and} \\ \bot & i = N. \end{cases}$$

Safe Failures. Next we note that there exists failures which do not violate safety. In particular, let $\mathbf{x}$ be a state along any execution of System and assume that $F(\mathbf{x}) = \emptyset$. Consider the execution fragment $\alpha = \mathbf{x} \,.\, \text{fail}_1(v) \,.\, \mathbf{x}' \,.\, \text{fail}_2(v) \,.\, \mathbf{x}'' \,\ldots\, \text{fail}_N(v) \,.\, \mathbf{x}_f$. That is, $\forall i \in ID$, let $\text{fail}_i(v)$ occur where v is the same for each of these fail_i transitions. Then, for any round $\mathbf{x}_s$ appearing after $\mathbf{x}_f$ in α, $Safety(\mathbf{x}_s)$.

Sharing Suspected Sets. The motivation for sharing sets of suspected agents among neighbors in Figure 7, Line 21, is illustrated by the following observation. It gives a failure condition under which no moves are possible and hence no progress can be made if sets of suspected agents are not shared. Assume that neighbors do not share sets of suspected agents, so Figure 7, Line 21 is deleted. Let $\mathbf{x}$ be a state along some execution where there are no failures and let agents $i \in ID \setminus H(\mathbf{x})$ be spaced evenly at a distance greater than flocking such that $\neg Flock_S(\mathbf{x})$, and particularly $\mathbf{x}.x_i - \mathbf{x}.x_{L_i} = \mathbf{x}.x_{R_i} - \mathbf{x}.x_i = \ldots = \mathbf{x}.x_{T(\mathbf{x})} - \mathbf{x}.x_{L_{T(\mathbf{x})}} > r_f \pm \frac{\delta}{2}$. Let there be a non-faulty agent p which is located farther than r_c from agent $T(\mathbf{x})$ so that $p \notin \mathbf{x}.Nbrs_{T(\mathbf{x})}$. Consider an execution fragment starting from $\mathbf{x}$ such that for every state $\mathbf{x}'$ in the execution fragment, $F(\mathbf{x}') = ID \setminus \{p\}$ and $\mathbf{x}'.vf_j = 0$ for all $j \in F(\mathbf{x}')$. Then, for states $\mathbf{x}''$ reachable from $\mathbf{x}'$, $\mathbf{x}''.Suspected_p = \emptyset$ and hence no progress is made as p never learns it must change lanes, so $\forall i \in ID$, $\mathbf{x}''.x_i = \mathbf{x}'.x_i$.

3.2 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 Figure 7, Line 35.

Lemma 3.5 *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 following corollary states that any two agents move towards or away from one another by at most $2v_{max}$ from one round to another and follows from Lemma 3.5.

Corollary 3.6 *For any execution* α, *for states* $\mathbf{x}, \mathbf{x}' \in \alpha$ *such that* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for any* $a \in A$, $\forall i, j \in ID$ *such that* $i \neq j$, *then* $|(\mathbf{x}'.x_i - \mathbf{x}.x_i) - (\mathbf{x}'.x_j - \mathbf{x}.x_j)| \leq 2v_{max}$.

Proof: By Lemma 3.5, both of $|\mathbf{x}'.x_i - \mathbf{x}.x_i| \leq v_{max}$ and $|\mathbf{x}'.x_j - \mathbf{x}.x_j| \leq v_{max}$. This gives that $\mathbf{x}'.x_i \in [\mathbf{x}.x_i - v_{max}, \mathbf{x}.x_i + v_{max}]$ and $\mathbf{x}'.x_j \in [\mathbf{x}.x_j - v_{max}, \mathbf{x}.x_j + v_{max}]$, and the result follows. ■

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

Lemma 3.7 *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* $c > 0$.

Proof: Only suspect and update modify $L_S(\mathbf{x}, i)$, $R_S(\mathbf{x}, i)$, or x_i for any i. By Lemma 3.2, we discuss L and R. By Lemma 3.3, which states that neighbor switching occurs symmetrically, if $\mathbf{x}.L_i \neq j$ and $\mathbf{x}'.L_i = j$, then $\mathbf{x}'.R_j = i$. It remains to be established that $\mathbf{x}'.x_{\mathbf{x}'.R_j} - \mathbf{x}'.x_{\mathbf{x}'.L_i} \geq r_s$. For convenient notation, observe that $\mathbf{x}'.x_{\mathbf{x}'.R_j} = \mathbf{x}'.x_i$ and $\mathbf{x}'.x_{\mathbf{x}'.L_i} = \mathbf{x}'.x_j$. Now,

$$\begin{aligned} \mathbf{x}'.x_j &= \frac{\mathbf{x}.x_{\mathbf{x}.L_j} + \mathbf{x}.x_i}{2}, \text{ and} \\ \mathbf{x}'.x_i &= \frac{\mathbf{x}.x_j + \mathbf{x}.x_{\mathbf{x}.R_i}}{2}, \end{aligned}$$

and thus

$$\begin{aligned} \mathbf{x}'.x_i - \mathbf{x}'.x_j &= \frac{\mathbf{x}.x_j + \mathbf{x}.x_{\mathbf{x}.R_i}}{2} - \frac{\mathbf{x}.x_{\mathbf{x}.L_j} + \mathbf{x}.x_i}{2} \\ &= \frac{\mathbf{x}.x_j - \mathbf{x}.x_{\mathbf{x}.L_j} + \mathbf{x}.x_{\mathbf{x}.R_i} - \mathbf{x}.x_i}{2}. \end{aligned}$$

Finally, by the hypothesis,

$$\mathbf{x}'.x_i - \mathbf{x}'.x_j \geq \frac{r_s + r_s}{2} \geq r_s.$$

The cases for $i = N$ and $j = 1$ follow by similar analysis, as does the case when $\mathbf{x}'.x_m$ is quantized so that $\mathbf{x}.x_m = \mathbf{x}'.x_m$ for any $m \in ID$. ■

Invariant 3.8 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 faulty agents—they may collide.

Invariant 3.8 *For any reachable state* $\mathbf{x}$, $Safety_R(\mathbf{x})$.

Proof: 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 $\mathbf{x} \xrightarrow{a} \mathbf{x}' \wedge \mathbf{x} \in \mathit{Safety}_R$, then $\mathbf{x}' \in \mathit{Safety}_R$. 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 $\mathit{Safety}_R(\mathbf{x}')$ For the update transition, the inductive hypothesis gives that Safety 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 3.7. 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 7, Lines 27–32 and the inductive hypothesis.

For the first case showing the spacing between any two non-faulty agents, it is sufficient to show if $\forall i \in \bar{F}(\mathbf{x})$, $\mathbf{x}.u_i - \mathbf{x}.u_l \geq r_r$ 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 33 follow by similar analysis and are omitted for space. Thus, $\mathit{Safety}_R(\mathbf{x})$.

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 3.1. For a failed agent j, $\mathbf{x}'.x_j = \mathbf{x}.x_j + \mathbf{x}.vf_j$ by Line 34. 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 3.5, 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 $\mathit{Safety}_R(\mathbf{x}_m)$ 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 $\mathit{Safety}_R(\mathbf{x}'_d)$ for a state $\mathbf{x}'_d$ reachable from state $\mathbf{x}_d$. By the detection time assumption, any agent i will have $j \in \mathbf{x}_d.\mathit{Suspected}_i$, which changes $L_S(\mathbf{x}_d)$ and $R_S(\mathbf{x}_d)$, but now apply Lemma 3.7, which shows there is at least r_r space between i and j. Finally, by Figure 7, Line 26, $\mathbf{x}'_d.\mathit{lane}_i \neq \mathbf{x}_d.\mathit{lane}_j$, since $N_L \geq 2$, and by Assumption b of Theorem 3.1, $\mathit{Safety}_R(\mathbf{x}'_d)$. ■

3.3 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 [14], 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 $\mathit{Terminal}$ is satisfied. However, note that the first state in such an execution is not entirely arbitrary, as Section 3.2 established that such states satisfy at least Safety_R, and all the following analysis relies on this assumption.

Influence of Failures on Progress. 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 7, Line 34. This observation also highlights why Flock is quantified over agents with identifiers in the set of suspected agents $\bar{S}(\mathbf{x})$ and not the set of failed agents $\bar{F}(\mathbf{x})$ or all agents ID—if it were quantified over ID, at no future point could $\mathit{Flock}(\mathbf{x})$ be attained if a failed agent has diverged. Furthermore, if $\mathit{Flock}(\mathbf{x})$ relied on $\bar{F}(\mathbf{x})$ instead of $\bar{S}(\mathbf{x})$, then potentially the failure detection algorithm could rely upon the head agent's detection of this predicate on the global snapshot for detection of failures. 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 3.1.

Progress along Fail-Free Executions. In the remainder of this section, we show that once new actuator faults cease occurring, System eventually reaches a state satisfying $\mathit{Terminal}$. 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}. Note that this does not mean there are no failures, so for each state $\mathbf{x} \in \alpha_{\mathit{ff}}$, it is not necessary that $F(\mathbf{x}) = \emptyset$, only that new failures do not occur, so $\forall \mathbf{x}, \mathbf{x}' \in \alpha_{\mathit{ff}}, F(\mathbf{x}) = F(\mathbf{x}')$.

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 35 of Figure 7 that if a non-faulty agent i moves in some round in spite of quantization, then it moves by at least a positive amount v_{min}. Observe that an agent may not move in a round if the conditional in Figure 7, Line 33 is satisfied, but this does not imply $v_{min} = 0$. This follows from Figure 7, Line 35.

Lemma 3.9 *For any failure-free execution fragment α and for two adjacent rounds $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ in α, for any $i \in \bar{F}(\mathbf{x}_k) \cap \bar{F}(\mathbf{x}_{k+1})$, if $|\mathbf{x}_{k,T}.u_i - \mathbf{x}_k.x_i| > \beta$, then $|\mathbf{x}_{k+1}.x_i - \mathbf{x}_k.x_i| \geq v_{min} > 0$.*

Next, Lemma 3.10 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 computing x_i in the Move subroutine does not increase any $e(\mathbf{x}, i)$.

Lemma 3.10 *For reachable states $\mathbf{x}, \mathbf{x}'$, if $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ and $\mathbf{x} \notin \mathit{NBM}$, for some $a \in A$, then $\max_{i \in \bar{F}(\mathbf{x})} e(\mathbf{x}', i) \leq \max_{i \in \bar{F}(\mathbf{x})} e(\mathbf{x}, i)$.*

Proof: Target and Quant are the only subroutines of update$_i$ to modify u_i. Now $\max_{i \in \bar{F}(\mathbf{x}_T)} eu(\mathbf{x}_T, i) \leq \max_{i \in \bar{F}(\mathbf{x})} eu(\mathbf{x}, i)$ which follows from $eu(\mathbf{x}_T, i)$ being computed as convex combinations of positions from $\mathbf{x}$, specifically

$$\begin{aligned} i = H(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T, i) = 0 \\ i = \mathbf{x}_T.R_{H(\mathbf{x}_T)} &\Rightarrow eu(\mathbf{x}_T, i) = \frac{eu(\mathbf{x}, \mathbf{x}.R_i)}{2} \\ i \in \mathit{RMids}(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T, i) = \frac{eu(\mathbf{x}, \mathbf{x}.L_i) + eu(\mathbf{x}, \mathbf{x}.R_i)}{2} \\ i = T(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T, i) = \frac{eu(\mathbf{x}, \mathbf{x}.L_i) + eu(\mathbf{x}, i)}{2}. \end{aligned}$$

Finally, Quant sets $\mathbf{x}_Q.u_i = \mathbf{x}_T.u_i$ or $\mathbf{x}_Q.u_i = \mathbf{x}_T.x_i$. In the first case, when $\mathbf{x}_Q.u_i = \mathbf{x}_T.u_i$, the result follows by the above reasoning. In the other case, when $\mathbf{x}_Q.u_i = \mathbf{x}_T.x_i$, if u_i and u_L are each quantized, then e_i does not change for any i and the result follows. If, however, u_i is quantized and u_L is not quantized, then

e_i is computed as

$$\begin{aligned}
i = H(\mathbf{x}_T) &\Rightarrow & eu(\mathbf{x}_T, i) &= 0 \\
i = \mathbf{x}_T.R_{H(\mathbf{x}_T)} &\Rightarrow & eu(\mathbf{x}_T, i) &= eu(\mathbf{x}, i) \\
i \in RMids(\mathbf{x}_T) &\Rightarrow & eu(\mathbf{x}_T, i) &= \frac{eu(\mathbf{x}, \mathbf{x}.R_i) + eu(\mathbf{x}, i)}{2} \\
i = T(\mathbf{x}_T) &\Rightarrow & eu(\mathbf{x}_T, i) &= \frac{eu(\mathbf{x}, i) + eu(\mathbf{x}, \mathbf{x}.L_i)}{2}.
\end{aligned}$$

Likewise, if u_L is quantized and u_i is not quantized, then e_i is computed as

$$\begin{aligned}
i = H(\mathbf{x}_T) &\Rightarrow & eu(\mathbf{x}_T, i) &= 0 \\
i = \mathbf{x}_T.R_{H(\mathbf{x}_T)} &\Rightarrow & eu(\mathbf{x}_T, i) &= \frac{eu(\mathbf{x}, i) + eu(\mathbf{x}, \mathbf{x}.R_i)}{2} \\
i \in RMids(\mathbf{x}_T) &\Rightarrow & eu(\mathbf{x}_T, i) &= \frac{eu(\mathbf{x}, \mathbf{x}.L_i) + eu(\mathbf{x}, i)}{2} \\
i = T(\mathbf{x}_T) &\Rightarrow & eu(\mathbf{x}_T, i) &= \frac{eu(\mathbf{x}, i)}{2}.
\end{aligned}$$

Finally, applying Lemma 3.5 indicates that error between actual positions and not target positions is non-increasing. ■

Next, Lemma 3.11 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 $V(\mathbf{x}) \triangleq \sum_{i \in \bar{F}(\mathbf{x})} e(\mathbf{x}, i)$. Note the similarity of this candidate with the one found in [15]. In particular, it is not quadratic and is the sum of absolute values of the positions of the agents. 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 3.11 *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 $k < c \leq \lceil (V(\mathbf{x}_k) - \gamma)/\psi \rceil$, where $\psi = v_{min}$.*

Proof: By definition of NBM since $\mathbf{x}_k \notin NBM$, $\exists j \in \bar{F}(\mathbf{x}_k)$ such that $|\mathbf{x}_{k,T}.x_j - \mathbf{x}_k.x_j| > \beta$, where $\mathbf{x}_{k,T}$ is the state obtained applying the subroutines of the update transition through *Target*. Let $j = \underset{i \in NF(\mathbf{x}_k)}{\operatorname{argmax}} e(\mathbf{x}_k, i)$. Thus Figure 7, Line 33 is not satisfied for j and $v_{max} \geq |\mathbf{x}_{k+1}.x_j - \mathbf{x}_k.x_j| \geq v_{min}$ by Figure 7, Line 35. Let $\Delta V(\mathbf{x}_k, \mathbf{x}_{k+1}) \triangleq V(\mathbf{x}_{k+1}) - V(\mathbf{x}_k)$, and we show $\Delta V(\mathbf{x}_k, \mathbf{x}_{k+1}) < \psi$ for some $\psi < 0$. Observe that $-v_{min} \geq \Delta V(\mathbf{x}_k, \mathbf{x}_{k+1}) \geq -v_{max}$ and since $v_{max} \geq v_{min} > 0$ let $\psi = v_{min}$. Therefore a transition $\mathbf{x}_k \overset{\text{update}}{\rightarrow} \mathbf{x}_{k+1}$ causes $V(\mathbf{x}_{k+1})$ to decrease by at least a positive constant v_{min}. By repeated application of this reasoning, $\exists c$, $k < c \leq \left\lceil \frac{V(\mathbf{x}_k) - \gamma}{v_{min}} \right\rceil$ such that $V(\mathbf{x}_c) \in NBM$ and $V(\mathbf{x}_c) \leq \gamma$. ■

Lemma 3.11 stated a bound on the time it takes for System to reach the set of states satisfying NBM. However, to satisfy $Flocks_S(\mathbf{x})$, all $\mathbf{x} \in NBM$ must be inside the set of states that satisfy $Flocks_S$, and the following lemma states this. From any state $\mathbf{x}$ that does not satisfy $Flocks_S(\mathbf{x})$, 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 $\beta \leq \delta/(4N)$.

Lemma 3.12 *If $Flocks_S(\mathbf{x})$, then $V(\mathbf{x}) \leq \sum_{i \in \bar{F}(\mathbf{x})} e(\mathbf{x}, i) = (\delta \left|\bar{F}(\mathbf{x})\right|)/4$.*

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 $V(\mathbf{x}') < (\delta \left|\bar{F}(\mathbf{x})\right|)/2$.

Lemma 3.13 *$Flock_W$ is a stable predicate.*

Proof: We show $\forall \mathbf{x}, \mathbf{x}' \in \alpha_{ff}$ such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ for any $a \in A$, if $Flock_W(\mathbf{x})$, then $Flock_W(\mathbf{x}')$. If $Flock_W(\mathbf{x})$, there are two cases to consider. In the first case, System satisfies $Flock_W(\mathbf{x}) \wedge \neg\, Flock_S(\mathbf{x})$, then $Flock_W(\mathbf{x}')$ holds by application of Lemma 3.11 since $\mathbf{x}'.x_{H(\mathbf{x}')} = \mathbf{x}.x_{H(\mathbf{x}')}$ by Figure 7, Line 30. In the second case, System satisfies $Flock_W(\mathbf{x}) \wedge Flock_S(\mathbf{x})$ so upon termination of the global snapshot algorithm, if $\mathbf{x}.x_{H(\mathbf{x})} \neq \mathbf{x}.x_g$, then $H(\mathbf{x})$ computes $\mathbf{x}'.u_{H(\mathbf{x}')} < \mathbf{x}.u_{H(\mathbf{x})}$ and applies this target $\mathbf{x}'.u_{H(\mathbf{x}')} < \mathbf{x}.u_{H(\mathbf{x})}$ by Figure 7, Line 29, and we show $Flock_S(\mathbf{x}) \Rightarrow Flock_W(\mathbf{x}')$. If $\mathbf{x}.x_{H(\mathbf{x})} \in [0, \beta]$ such that the predicate on Line 33 is satisfied, then $\mathbf{x}'.x_{H(\mathbf{x}')} = \mathbf{x}.x_{H(\mathbf{x})}$ and the proof is complete. If not, then by the definition of $\mathbf{x}'.u_{H(\mathbf{x}')}$ in Figure 7, Line 29, $H(\mathbf{x})$ will compute a target no more than $\delta/2$ to the left, so $\left|\mathbf{x}'.u_{H(\mathbf{x}')} - \mathbf{x}.u_{H(\mathbf{x})}\right| \leq \delta/2$. Now, for Agent$_i$ to have moved, the error between the distance of $H(\mathbf{x})$ and $\mathbf{x}.R_{H(\mathbf{x})}$ and the flocking distance must have been at most $\delta/2$ by the definition of $Flock_S$. Agent$_{R_{H(\mathbf{x})}}$ will have moved to the center of $H(\mathbf{x})$ and $R_{R_{H(\mathbf{x})}}$, so $\mathbf{x}'.u_{R_{H(\mathbf{x}')}}$ may be less than, equal to, or greater than its previous position $\mathbf{x}.x_{R_{H(\mathbf{x})}}$, requiring a case analysis of each of these three possibilities. In the first two cases $\mathbf{x}'.u_{R_{H(\mathbf{x}')}} \leq \mathbf{x}.x_{R_{H(\mathbf{x}')}}$ and the proof is complete. The other case follows by applying Lemma 3.5 to $H(\mathbf{x})$ and $\mathbf{x}.R_{H(\mathbf{x})}$ and observing that the most they would ever move apart by is $2\beta \leq \delta/2$ and are now separated by at most δ, hence $Flock_W(\mathbf{x}')$ is satisfied. ■

The following corollary follows from Lemma 3.11, 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 3.14 *For $\mathbf{x} \in \alpha_{ff}$ such that, if $Flock_S(\mathbf{x})$, $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ for any $a \in A$, 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 3.1 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 $j \in \bar{F}(\mathbf{x})$ 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 3.1 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 overall time constant c to ensure safety and progress being linear in the number of agents.

Lemma 3.15 *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 following general sequence convergence lemma is used to show that System eventually satisfies the desired properties and terminates. Essentially, it states that for a lexicographically ordered tuple, at least one component decreases in every step, and the other component does not increase beyond some bound.

Lemma 3.16 *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: First, note that by assumption, a_{j+1} is bounded from above by a_j (that is, by a_1). Now assume for the purpose of contradiction that there exists a pair $\langle a_p, b_p \rangle$ where $a_p > c_5$ and $b_p > c_4$ such that $\forall f \geq p$, $\langle a_f, b_f \rangle = \langle a_p, b_p \rangle$. Then, we show there exists a $q > p$ such that $\langle a_p, b_p \rangle = \langle a_q, b_q \rangle$ where $a_q < a_p$ and $b_q < b_p$.

Without loss of generality, assume that $b_p > c_2$ initially. Now, starting from $\langle a_p, b_p \rangle$, the next step in the sequence is such that $b_{p+1} \leq b_p - c_1$, since it must be the case that $a_p = a_{p+1}$ as we assumed $b_p > c_2$. This process of b_j decreasing continues in the form of $b_n \leq b_p - nc_1$ where n is the step that $b_n \leq c_2$, thus $b_n \leq b_p - nc_1 \leq c_2$ and $n \geq \frac{b_p - c_2}{c_1}$. At the next step from n, that is $n+1$, it must be the case that $a_{n+1} \leq \max\{0, a_n - c_3\}$ since $b_n \leq c_2$ and $a_n = a_p > c_5$. Since $a_{n+1} < a_n$, it is the case that $b_{n+1} \leq c_6 - c_2 = c_6 - nc_1$. Note that it would seem to remain to be established that $b_n > c_4$ so that the decrease of b_{n+1} could occur, but, if it is in fact the case that $b_n \leq c_4$, then $b_p \in B$ as desired. Therefore, $q = n+1 > p$ and since $\langle a_p, b_p \rangle$ becomes smaller at a larger step in the sequence, we reach the contradiction. By repeatedly applying the previous arguments, existence of such a t is established. ■

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 3.17 *Let* $\alpha_{ff} = \mathbf{x}_0, \mathbf{x}_1, \ldots$. *Consider the infinite sequence* $\langle \mathbf{x}_0.x_{H(\mathbf{x}_0)}, V(\mathbf{x}_0) \rangle$, $\langle \mathbf{x}_1.x_{H(\mathbf{x}_1)}, V(\mathbf{x}_1) \rangle, \ldots$, $\langle \mathbf{x}_t.x_{H(\mathbf{x}_t)}, V(\mathbf{x}_t) \rangle, \ldots$. *Then, there exists t at most* $\left\lceil \frac{(V(\mathbf{x}_0) - |\bar{F}(\mathbf{x})| \delta/4)}{v_{min}} \right\rceil + \left\lceil \frac{|\bar{F}(\mathbf{x})| \delta/4}{v_{min}} \right\rceil \max\{1, \frac{\mathbf{x}_0.x_{H(\mathbf{x}_0)}}{v_{min}} O(N)\}$ *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) $V(\mathbf{x}_t) \leq |\bar{F}(\mathbf{x})| \frac{\delta}{4}$,

(e) $Terminal(\mathbf{x}_t)$*, and*

(f) $Flock_S(\mathbf{x}_t)$.

Proof: This follows from Lemma 3.11, the $O(N)$ termination time of the snapshot algorithm, and from Lemma 3.16 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) $c_6 = (N-1)\frac{\delta}{2}$. ■

3.4 Failure Detection

In the earlier analysis we assumed that it is possible to detect all actuator faults within a 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 zero 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.

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 3.11.

Lemma 3.18 *The detection time lower-bound for any detectable actuator fault is $O(N)$.*

Proof: 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 $\mathbf{x} \notin Flock_S$. In both executions, assume that any time Line 35 of Figure 7 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 $Flock_S$. Lemma 3.11 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 $\mathbf{x}' \in Flock_S$. Once $\mathbf{x}' \in Flock_S$, 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 7, Line 29 Thus, α_f and α_n are indistinguishable up to state $\mathbf{x}'$ and by Lemma 3.11, $\mathbf{x}'$ is a constant a number of rounds from $\mathbf{x}$ where a is $O(N)$. ■

Next we show that the the failure detection mechanism incorporated in Figure 7 does not produce any false positives.

Lemma 3.19 *In any reachable state $\mathbf{x}$, $\forall j \in \mathbf{x}.Suspected_i \Rightarrow \mathbf{x}.failed_j$.*

Proof: Suppose $\exists i, j$ such that $j \in \mathbf{x}.Suspected_i$, then the precondition for 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 7, Line 33 for some $j \notin \mathbf{x}_{k_s-1}.Suspected_i$.

The first case occurs if the quantization constraint $|\mathbf{x}_s.x_j - \mathbf{x}_{s,T}.u_j| \leq \beta$ is not satisfied (Figure 7, Line 33), so Agent$_j$ applies a velocity in the direction of $\operatorname{sgn}(u_j - x_j)$. If $\operatorname{sgn}(\mathbf{x}'_s.x_j - \mathbf{x}_s.x_j) \neq \operatorname{sgn}(\mathbf{x}_s.u_j - \mathbf{x}_s.x_j)$, 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 $\operatorname{sgn}(\mathbf{x}'_s.u_j - \mathbf{x}'_s.x_j) \neq \operatorname{sgn}(\mathbf{x}_s.u_j - \mathbf{x}_s.x_j)$, 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 7, Line 33, 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 $\mathbf{x}_s.vf_j \neq 0$, $\mathbf{x}'_s.x_j = \mathbf{x}_s.x_j + \mathbf{x}_s.vf_j$. ■

The next lemma shows a partial *completeness* property [16] of the failure detection mechanism incorporated in Figure 7.

Lemma 3.20 *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 $\operatorname{sgn}(\mathbf{x}.x_j - \mathbf{x}.xo_j) \neq \operatorname{sgn}(\mathbf{x}.uo_j - \mathbf{x}.xo_j)$. Then, $\mathbf{x} \xrightarrow{\text{suspect}_i(j)} \mathbf{x}'$.*

Proof: For a suspect$_i$ transition to be taken, the precondition at Lines 6–8 of Figure 7 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 $\operatorname{sgn}(\mathbf{x}.x_j - \mathbf{x}.xo_j) \neq \operatorname{sgn}(\mathbf{x}.uo_j - \mathbf{x}.xo_j)$. These are the two hypotheses of the lemma and thus the result follows that the suspect$_i$ transition is enabled. ■

Now we show an upper-bound on the number of rounds to detect any failure which may be detected using the failure detection mechanism incorporated in Figure 7 by applying Lemma 3.18 with Lemmata 3.19 and 3.20, and that agents share suspected sets in Figure 7, Line 21. This states an $O(N)$ upper-bound on the detection time of our failure detector and shows that eventually all non-faulty agents know the set of failed agents.

Corollary 3.21 *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 $\forall i \in \bar{F}(\mathbf{x}_s)$, $\mathbf{x}_s.Suspected_i = F(\mathbf{x})$ and $k - s$ is $O(N)$.*

3.5 Simulations

Simulation studies were performed, where flocking convergence time (as by Lemma 3.11), goal convergence time (as by Theorem 3.17), and failure detection time (as by Corollary 3.21) were of interest. Unless otherwise noted, the parameters are chosen as $N = 6$, $N_L = 2$, $r_s = 20$, $r_f = 40$, $\delta = 10$, $\beta = \delta/(4N)$, $v_{min} = \frac{\beta}{2}$, $v_{max} = \beta$, the head agent starts with position at r_f, and the goal is chosen as the origin.

Figure 9 shows System without failures "expanding" [3] to form a strong flock prior to moving towards the goal, and Figure 10 shows the value of the Lyapunov function, V, and maximum agent error from flocking, e_{max}, during this simulation. The initial state is that each agent is spaced by r_s from its left neighbor. Observe that while moving towards the goal, $Flock_S$ is repeatedly satisfied and violated, with invariance of $Flock_W$.

Figure 11 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 3.21. The initial state was for expansion, so each agent was spaced at r_s from its left neighbor.

Figure 9: Agent expansion simulation showing agent positions, x_i. Observe that first the agents form a flock by expanding and then move toward the goal.

Figure 10: Expansion simulation showing max error e_{max}, Lyapunov function value V, with weak and strong flocking constants.

Figure 11: Rounds k (upper bounded by k_u) to reach $Terminal$ versus number of agents N with fixed and varying values of v_{min} (i.e., as a function of N or not).

Figure 12 illustrates the positions of the agents as they move towards the goal, and highlights that $Flock_W$ is a stable predicate. The simulation began with System satisfying $Flock_S$, which was then invalidated as the head agent learned this through the global snapshot protocol and made a move toward the goal. Figure 13 shows the values of the maximum agent error e_{max} along with the Lyapunov function V from the simulation in Figure 12, and displays that $Flock_S$ is not invariant, while $Flock_W$ is a stable predicate, since V is bounded by $\frac{N\delta}{2}$.

Figure 14 shows the detection time with varying N and f from a fixed initial condition of inter-agent spacings at $2r_f$. The $f_{Avg} = i$ lines show the total detection time divided by f. Failures were fixed with $vf_i = 0$, failing each combination of agents, so for $f = 2$ and $N = 3$, each combination of $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$ were failed individually, and the detection time is the average over the number of these combinations for each choice of f and N. The detection time averaged over the number of failure indicates that the detection time to detect any failure in a multiple failure scenario is on the same order as that in the single failure case. However, the detection time not averaged over the number of failures indicates that the detection time to detect all failures increases linearly in f and on the order of N, as predicated by Corollary 3.21.

Figure 15 shows the detection time as a function of which agent fails with what failure velocity from three different types of initial states. In all single-failure simulations, a trend was observed on the detection time. When failing each agent individually, and with all else held constant (initial conditions, round of

Figure 12: Movement of the flock towards the goal with $Flock_W$ invariance.

Figure 13: Invariance of $Flock_W$ and repeated entry to NBM and $Flock_S$.

Figure 14: Simulating f zero velocity failures at round 0 from initial state of $2r_f$ inter-agent spacing.

failure, etc.), only one of the detection times for failure velocities of $-v_{max}$, 0, or v_{max} is ever larger than one round. The frequent occurrence of a single round detection is interesting. 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. However, detecting that the head agent has failed with zero velocity requires convergence of the system to a strong flock prior to detection, as does detecting that the tail agent failed with v_{max}, as this mimics the desired expansive behavior up to the point where the tail moves beyond the flock. 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. Thus the observation is, for a reachable state $\mathbf{x}$, if $|F(\mathbf{x})| = 1$, let the identifier 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 $\mathbf{x}.vf_i \in [-v_{max}, 0)$. 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. This 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 failure velocity conditions violate either progress or safety immediately and lead to an immediate detection.

Figure 15: 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.

4 Conclusion

This report presented an algorithm for the safe flocking problem—where the desired properties are safety invariance and eventual progress, that eventually a strong flock is formed and a destination reached by that flock—in spite of permanent actuator faults. An $O(N)$ lower-bound was presented for the detection time of actuator faults, as well as conditions under which the given failure detector can match this bound, although it was established that this is not always possible. The main result was that the algorithm is self-stabilizing when combined with a failure detector. 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. Simulation results served to reiterate the formal analysis, and demonstrated the influence of certain factors—such as multiple failures and failure velocity—on the failure detection time.

References

[1] E. Shaw, "Fish in schools," *Natural History*, vol. 84, no. 8, pp. 40–45, 1975.

[2] A. Okubo, "Dynamical aspects of animal grouping: Swarms, schools, flocks, and herds," *Adv. Biophys.*, vol. 22, pp. 1–94, 1986.

[3] V. Gazi and K. M. Passino, "Stability of a one-dimensional discrete-time asynchronous swarm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 4, pp. 834–841, Aug. 2005.

[4] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.

[5] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[6] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[7] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[8] K. M. Chandy and L. Lamport, "Distributed snapshots: determining global states of distributed systems," *ACM Trans. Comput. Syst.*, vol. 3, no. 1, pp. 63–75, 1985.

[9] T. Johnson, "Fault-tolerant distributed cyber-physical systems: Two case studies," M.S. thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, May 2010.

[10] M. Franceschelli, M. Egerstedt, and A. Giua, "Motion probes for fault detection and recovery in networked control systems," in *American Control Conference, 2008*, June 2008, pp. 4358–4363.

[11] S. Dolev, *Self-stabilization*. Cambridge, MA: MIT Press, 2000.

[12] D. Liberzon, *Switching in Systems and Control*. Boston, MA, USA: Birkhäuser, 2003.

[13] J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*. New York, NY, USA: Marcel Dekker, 1998.

[14] A. Arora and M. Gouda, "Closure and convergence: A foundation of fault-tolerant computing," *IEEE Trans. Softw. Eng.*, vol. 19, pp. 1015–1027, 1993.

[15] J. Yu, S. LaValle, and D. Liberzon, "Rendezvous without coordinates," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, Dec. 2008, pp. 1803–1808.

[16] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, no. 2, pp. 225–267, 1996.