Safe Flocking in Spite of Actuator Faults using Directional Failure Detectors

Taylor T. Johnson and Sayan Mitra*†

Abstract. The safe flocking problem requires a collection of 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. In this paper, we study one-dimensional safe flocking in the presence of actuator faults and directional failure detectors (DFDs). Actuator faults cause affected agents to move permanently with arbitrary velocities, and DFDs detect failures only when actuation and required motion are in opposing directions. First, assuming existence of a DFD for actuator faults, we present an algorithm for safe flocking. Next, we show that certain actuator faults cannot be detected with DFDs, while detecting others requires time that grows linearly with the number of participating agents. Finally, we show that our DFD algorithm achieves the latter bound.

Keywords. failure detector, flocking, self-stabilization, switched systems.

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 called a *flock*, (b) reach a specified destination called the *goal*, 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, this problem and variants have been studied in the robotics, control, and multi-agent systems literature (see, for example, [3, 4, 5, 6, 7, 8, 9, 10]). Most works on flocking assume agents communicate synchronously and that there are no failures [11, 4]. In [11, 4], flocking is studied where agents move at constant velocities and update only their orientations, while in [12, 7], agents have double-integrator dynamics. Even in these settings, to the best of our knowledge, practical safe-flocking in general Euclidean spaces is an open problem, as existing algorithms require unbounded accelerations for guaranteeing safety [7].

*Taylor T. Johnson and Sayan Mitra are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801. E-mails: johnso99@illinois.edu, mitras@crhc.uiuc.edu

†Manuscript received February 17, 2011; revised x.

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. At the beginning of each synchronous 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. That is, our flocking algorithm calculates only the direction in which an agent should move, based on the positions of adjacent agents, and then the speed with which an agent moves is chosen nondeterministically over a range, making the algorithm implementation independent with respect to lower-level motion controllers. Furthermore, the model obtained with rectangular dynamics overapproximates any behavior that can be obtained with double-integrator dynamics with bounded acceleration. Even in this setting with simpler dynamics, it is tricky to develop distributed algorithm that provide collision avoidance, as evidenced by an error that we uncovered in the algorithm proposed in [6]; see Section 4 for details of the error.

Unlike the algorithms in [6, 5, 7, 4] that provide convergence to a flock, we require termination, that is, agents should eventually stop moving. To this end, we use a form of quantization [13, 14]: 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 agent's current position x_i. We believe that such quantized control is appropriate for realistic actuators, where power constraints make 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 involved.

Our algorithm combines the corrected algorithm from [6] with Chandy-Lamport's distributed global snapshot algorithm [15]. 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 snapshot algorithm [15], and the detecting agent moves toward the destination. This breaks the strong flock, but preserves the weak flock, and in addition, the detecting agent makes progress toward the goal.

In addition, we allow agents to be affected by *actua-*

tor 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 [16]. Unlike byzantine faults, behaviors resulting from actuator faults are constrained by physical laws. Also, unlike crash failures [17] which typically thwart progress but not safety, actuator faults can also violate safety.

In this paper, 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 cause the agent to continue to move with the arbitrary but bounded failure velocity. For detecting faults, we restrict our attention to the class of failure detectors that rely on incorrect motion, which we call directional failure detectors (DFDs). A fault is detected by a DFD when either the actuation and the desired motion are in opposing directions, or there is actuation in a situation where there should be none. With DFDs, some faults are undetectable, such as an agent failing with zero velocity at the goal. For DFDs, we establish a simple lower-bound for faults which are detectable in finite time, which shows that there are faults that cannot be detected in fewer than $O(N)$ rounds, where N is the total number of agents. Unfortunately, certain faults may lead to a violation of safety in fewer rounds, so DFDs cannot ensure safety in such cases.

With DFDs, under some additional assumptions—such as sufficient initial spacing and detectable faults—our algorithm for safe flocking works in spite of failures. This ensures that non-faulty agents are 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.

Overall, our system model is described by a nondeterministic switched system, where nondeterminism arises from (a) agents choosing velocities in the range $[v_{min}, v_{max}]$, and (b) actuator faults, which cause a nondeterministic switch to a different system matrix upon occurring. We prove safety properties using inductive invariants. For proving progress and stabilization, that is, recovery after faults, we show that the DFD eventually notifies agents that any of their neighbors are faulty so that eventually faults do not violate progress, and then we combine this with a convergence proof which relies on Lyapunov theory to show that eventually flocking is achieved and the goal reached. In summary, the key contributions of this paper are:

(a) Introduction of actuator faults, directional failure detectors, and stabilization in spite of such failures in distributed cyber-physical systems.

(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 the formal model of the system and its properties. In Section 3, we introduce the main result—that the algorithm solves the safe flocking problem in spite of actuator faults. In Section 3.2, we establish safety in spite of actuator faults, then in Section 3.3, we show progress of the system toward a flock and goal, and Section 3.4 presents analysis of DFDs. Section 3.5 extends the main result by removing the assumption used in Sections 3.2, 3.3, and 3.4 that faults cannot cause the communication graph of non-faulty agents to partition. Finally, Section 4 reviews relevant related work and we conclude in Section 5. Appendix A presents a switched systems model of the system, and Appendix B presents simulation studies of the system.

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.

2.1 Preliminaries

For a natural number $K \in \mathbb{N}$, $[K] \triangleq \{1, \ldots, K\}$ and for a set S, we define $S_\perp \triangleq S \cup \{\perp\}$. For a set S, $|S|$ denotes the cardinality of S. For a variable x, its type is denoted by $type(x)$ and is the set of values that it can take. For a set of variables X, a *valuation* is a function that maps each $x \in X$ to a value in $type(x)$. 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*.

A state $\mathbf{x} \in Q$ of $\mathcal{A}$ is a valuation of all the variables in X and a valuation of a variable $x \in X$ is written $\mathbf{x}.x$. An *execution fragment* of A is an (possibly infinite) alternating sequence of states and transition names, $\alpha = \mathbf{x}_0$, a_1, $\mathbf{x}_1$, ..., such that for each index k appearing in α, $(\mathbf{x}_k, a_{k+1}, \mathbf{x}_{k+1}) \in \rightarrow$ and is represented by the notation $\mathbf{x}_k \overset{a_{k+1}}{\rightarrow} \mathbf{x}_{k+1}$. 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 $\mathbf{x}$, and the set of reachable states is denoted $Reach(\mathbf{x})$. 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 stable set [18]. Self-stabilization has been widely employed in developing distributed algorithms which operate in fault-prone and dynamic environments [19, 20]. In this paper, we model actuator faults by transitions with the special label fail, where such a transition will update the corresponding agent's state variables such that it moves with a constant failure velocity.

We define a *fail-free execution fragment* as an execution fragment α_{ff} beginning from some reachable state $\mathbf{x}$ that does not contain any more fail actions. Along such execution fragments α_{ff}, no non-faulty agents fail, nor do any faulty agents recover from being failed. We also define the *fail-free reachable states*, $FFReach(\mathbf{x})$ to be the set of states that is reachable from a reachable state $\mathbf{x}$ with any fail-free execution fragment. If we discuss $FFReach(\mathbf{X})$ where $\mathbf{X}$ is a set of reachable states, this is defined as $\bigcup_{\mathbf{x}\in\mathbf{X}} FFReach(\mathbf{x})$. We define the *fail-free states reachable in t rounds* as $FFReach(\mathbf{x}, t)$ to be the fail-free reachable states reached in t rounds without any new faults—this constrains any corresponding fail-free execution fragment starting from a state in $FFReach(\mathbf{x})$ for a reachable state $\mathbf{x}$ to be a sequence of at most length t. We define $FFReach(\mathbf{X}, t)$ analogously to $FFReach(\mathbf{X})$ for a set of reachable states $\mathbf{X}$.

Given $S \subseteq Q$, $\mathcal{A}$ *self-stabilizes* to S if (a) S is a stable predicate for $\mathcal{A}$ for all transitions except possibly the fail transitions, (b) from every reachable state of $\mathcal{A}$ (including states reached via fail transitions), every *fail*-free execution fragment eventually reaches S.

2.2 Model of Safe Flocking System

One desired property of safe flocking is that the system reaches a goal position, and we assume the goal is the origin without loss of generality (it could be any other real value), and that all agent positions are initially in the nonnegative reals (for a non-origin goal, they just need to be greater than or equal to the goal position). A distributed system consists of a set of N mobile *agents* physically positioned on N_L parallel *lanes*, which extend infinitely over nonnegative reals. The system can be thought of as a collection of cars in the lanes on a highway (see Figures 2 and 3). We assume $N_L \geq 2$, that is, there are at least two lanes, and we will later see that this allows safety and progress properties to be maintained in spite of failures. Having lanes allows non-faulty agents to avoid collisions with faulty agents, and allows them to pass faulty agents which are not moving toward the goal.

We assume synchronous communication between agents: agents have synchronized clocks, message delays are bounded, and computations are instantaneous. The communication neighbors of an agent are the other agents that are sufficiently close to the agent (within r_c, the communication distance defined below), regardless of lane. At each round, each agent exchanges messages bearing state information with its neighbors, and we emphasize that agents in different lanes communicate. 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. Additionally, the following positive real 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 inter-agent gap which defines a flock,

(e) δ: flocking tolerance parameter,

(f) β: quantization parameter, and

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

State Variables. The domains and initial values of the state variables for each agent are shown in Figure 1 using the ':=' notation. This notation defines what valuation the variables initially take. The discrete transition system corresponding to agent i has the following *private* variables:

(a) $failed_i$: indicates whether or not agent i is faulty,

(b) vf_i: if agent i is faulty, then this is the velocity with which it has failed,

(c) $Nbrs_i$: is the set of identifiers of agents that are neighbors of agent i,

(d) L_i and R_i: are respectively the identifiers of the nearest left and right neighbors of agent i that i believes to not be faulty,

(e) sr_i: indicates whether the global snapshot algorithm has been initiated, and

(f) gsf_i: indicates whether a certain stable predicate (strong flocking, defined below) detected by the global snapshot is satisfied or not.

x_i, xo_i :	$\mathbb{R}$		vf_i :	$\mathbb{R}_\perp :=$	$\perp$
u_i, uo_i :	$\mathbb{R} :=$	x_i	L_i :	$ID :=$	$L_S(\mathbf{x}, i)$
$lane_i$:	$LD :=$	1	R_i :	$ID :=$	$R_S(\mathbf{x}, i)$
sr_i :	$\mathbb{B} :=$	*false*	$Nbrs_i$:	$Set[ID] :=$	$Nbrs(\mathbf{x}, i)$
gsf_i :	$\mathbb{B} :=$	*false*	$Susp_i$:	$Set[ID] :=$	$\emptyset$
$failed_i$:	$\mathbb{B} :=$	*false*			

Figure 1: Variables of $\mathsf{Agent_i}$.

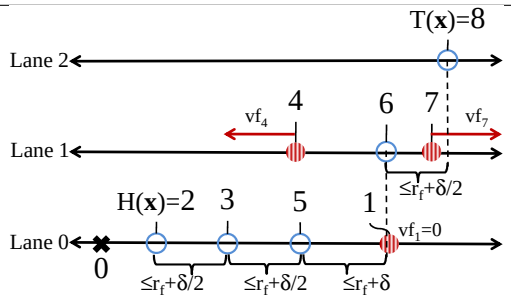

Figure 2: System at state $\mathbf{x}$ for $N = 8$ agents and $N_L = 3$ lanes. Agent positions are indicated by the circles on the lanes. The non-faulty agents identifiers are $\bar{F}(\mathbf{x}) = \{2, 3, 5, 6, 8\}$ and the faulty agent identifiers are $F(\mathbf{x}) = \{1, 4, 7\}$. Failure velocities of faulty agents are labeled vf_i. Non-faulty agents have avoided faulty agents by changing lanes. Note that $L(\mathbf{x}, 6) = 5$. Also, if $4 \in Susp_6$, then $L_S(\mathbf{x}, 6) = L(6) = 5$, else $L_S(\mathbf{x}, 6) = 4$. Assuming $S(\mathbf{x}) = F(\mathbf{x})$, $Flock_W(\mathbf{x})$, but $\neg Flock_S(\mathbf{x})$, since $|\mathbf{x}.x_6 - \mathbf{x}.x_5 - r_f| \leq \delta$, but not $\delta/2$.

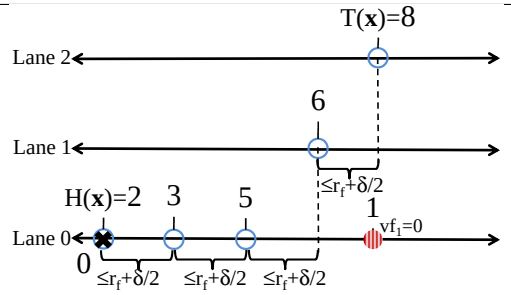

Figure 3: A terminal configuration, where the system has achieved $Flock_S(\mathbf{x})$ and reached the head has reached the goal (the origin). Observe that faulty agents 4 and 7 with nonzero velocity have diverged and that non-faulty agents do not necessarily terminate on the same lane. Faulty agent 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.

For the sake of convenient modeling[1], we assume the following *shared* variables are controlled by agent i, and are instantaneously read by the neighbors of i at the beginning of each round (see Figure 4):

[1]In a real implementation, these variables could be broadcast to the neighbors of i using messages at the beginning of each round.

(a) x_i and xo_i: agent i's current position and position from the end of the previous round,

(b) u_i and uo_i: agent i's target position and target position from the end of the previous round,

(c) $lane_i$: the lane currently occupied by agent i, and

(d) $Susp_i$: set of neighbors that agent i believes to be faulty.

The discrete transition system modeling the complete ensemble of agents is called System. We refer to states—valuations of all variables—of System with bold letters $\mathbf{x}$, $\mathbf{x}'$, etc., and at a given state $\mathbf{x}$, the valuation of $\mathsf{Agent_i}$'s variable x_i is denoted by $\mathbf{x}.x_i$

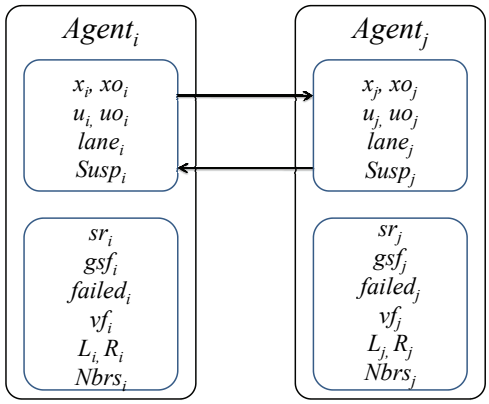

Figure 4: Interaction between a pair of neighboring agents is modeled with shared variables x, xo, u, uo, $lane$, and $Susp$.

Actuator Faults and Failure Detection. The failure of agent i's actuators is modeled by the occurrence of a transition labeled by $\mathsf{fail}_i(v)$, where v is the failure velocity parameter. Assume that $|v| \leq v_{max}$, which is reasonable due to physical actuation constraints. As a result of this transition, $failed_i$ is set to *true* and vf_i is set to v. This transition is always enabled unless i is already faulty. At state $\mathbf{x}$, if $\mathbf{x}.failed_i = true$, then $\mathsf{Agent_i}$ is a *faulty* agent, otherwise it is a *non-faulty* agent. At state $\mathbf{x}$, $F(\mathbf{x})$ and $\bar{F}(\mathbf{x})$ denote the sets of faulty and non-faulty agent identifiers, respectively. An actuator fault causes the affected agent to move forever with a constant but arbitrary failure velocity.

Agents do not have any direct information regarding the failure of other agents' actuators (that is, agent i cannot read $failed_j$ nor vf_j). Instead, they 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 $Susp_i$ variable and a transition labeled by $\mathsf{suspect_i}$. The $\mathsf{suspect_i}(j)$ transition models a failure detection of a faulty 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 $\mathbf{x}$, $\mathbf{x}.Susp_i \subseteq ID$ is the set of agent identifiers that agent i's failure detector suspects as faulty. Agent j is said to be *suspected* if some agent i suspects it, otherwise it is *unsuspected*. At state $\mathbf{x}$, if there exists $i \in ID$ such that $i \in \mathbf{x}.Susp_j$, then i is called an agent *suspected by* j. We denote the sets of suspected and unsuspected agents by $S(\mathbf{x})$ and $\bar{S}(\mathbf{x})$, respectively. Later in Section 3.4 we will introduce a specific kind of failure detector called the *directional failure detector* (DFD), which will explain the precondition of the suspect action.

We define *detection time* as the minimum number of rounds within which every faulty agent is suspected by all of its non-faulty neighbors. For safe flocking, we require all faulty agents to have finite detection time. However, in Section 3.4, we identify a class of faults—called *undetectable faults*—which do not have finite detection time. All other faults are called *detectable faults*—and by definition have finite detection time—and for all of these faults, we present an algorithm which accomplishes safe flocking. In most parts of Section 3 we will restrict our attention to detectable faults, and in Section 3.4, we discuss specific conditions under which the detection time is finite and state upper and lower bounds for it.

```
1  fail_i(v), |v| ≤ v_max
     pre ¬failed
3    eff failed := true; vf := v

5  suspect_i(j), j ∈ Nbrs
     pre j ∉ Susp ∧ (if |xo_j − uo_j| ≥ β
7          then sgn(x_j − xo_j) ≠ sgn(uo_j − xo_j)
           else |x_j − uo_j| ≠ 0)
9    eff Susp := Susp ∪ {j}

11 snapStart_i
     pre L = ⊥ ∧ ¬sr
13   eff sr := true // global snapshot invoked

15 snapEnd_i(GS), GS ∈ {false, true}
     eff gsf := GS; // returns whether strong flocking satisfied
17     sr := false

19 update_i
     eff uo := u; xo := x; Nbrs := Nbrs(x, i)
21     for each j ∈ Nbrs, Susp := Susp ∪ Susp_j
       L := L_S(x, i); R := R_S(x, i)
23     Mitigate:
         if ¬ failed ∧ (∃ s ∈ Susp : lane_s = lane)
25         ∧ (∃ L ∈ LD : ∀ j ∈ Nbrs, (lane_j = L ⇒
             x_j ∉ [x − r_s − 2v_max, x + r_s + 2v_max ]))
27       then lane := L fi
       Target:
29       if L = ⊥ ∧ gsf then u := x − min{x, δ/2}; gsf := false
         elseif L = ⊥ then u := x
31       elseif R = ⊥ then u := (x_L + x + r_f)/2
         else u := (x_L + x_R)/2 fi
33     Quant:
         if |u − x| ≤ β then u := x; fi
35     Move:
         if failed then x := x + vf
37       else x := x + sgn(x − u) choose [v_min, v_max] fi
```

Figure 5: $\mathsf{Agent_i}$'s transitions: failures, global snapshot initiation and termination, failure detection and mitigation, target computations, and movement. Variables without subscripts are those of $\mathsf{Agent_i}$.

Communication Neighbors and Groups. Before presenting the update transition, which models the flocking algorithm, we introduce some additional notation. The set of identifiers of all neighbors of $\mathsf{Agent_i}$ at state $\mathbf{x}$ is defined as

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

For a state $\mathbf{x}$, neighbors induces a communication graph $\Gamma(\mathbf{x}) \triangleq (\nu(\mathbf{x}), E(\mathbf{x}))$, where the set of vertices are agent identifiers—that is, $\nu(\mathbf{x}) \triangleq ID$—and the set of undirected edges are

$$E(\mathbf{x}) \triangleq \{(i, j) \in ID \times ID : j \in Nbrs(\mathbf{x}, i)\}.$$

The non-faulty communication graph $\Gamma_{\bar{F}}(\mathbf{x})$ is the subgraph of $\Gamma(\mathbf{x})$ restricted to non-faulty agent identifiers. We say that agent i is *connected* to agent j at state $\mathbf{x}$ if there exists a sequence of vertices $\rho(\mathbf{x}) \triangleq i, v_1, v_2, \ldots, v_{n-1}, j$ in $\Gamma(\mathbf{x})$, beginning with i and ending with j such that each pair of adjacent vertices $(i, v_1), (v_1, v_2), \ldots, (v_{n-1}, j)$ is an edge in $E(\mathbf{x})$. We analogously define *non-faulty connectedness* between non-faulty agents i and j, except in terms of $\Gamma_{\bar{F}}(\mathbf{x})$ for a state $\mathbf{x}$. We define the *communication group*—or group for short—of an agent i, denoted $G(\mathbf{x}, i)$, to be the set of vertices of the maximal connected subgraph of $\Gamma(\mathbf{x})$ containing i. By this definition, if agents i and j are connected at a state $\mathbf{x}$, then $G(\mathbf{x}, i) = G(\mathbf{x}, j)$.

For a state $\mathbf{x}$, we denote the set of all distinct communication groups as $G(\mathbf{x})$, where the cardinality of $G(\mathbf{x})$ is equal to the number of disjoint communication groups, which we denote by $|G(\mathbf{x})|$. Similarly, we define $G_{\bar{F}}(\mathbf{x})$ to be the set of all the non-faulty communication groups.

For any two states $\mathbf{x}, \mathbf{x}'$ of System, if $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ for some action $a \in A$, we say that two groups have *merged* if $\exists i, j \in ID$ such that i was not connected to j at state $\mathbf{x}$, but i is connected to j at state $\mathbf{x}'$; the two groups that merged are $G(\mathbf{x}, i)$ and $G(\mathbf{x}, j)$ and the resulting group is $G(\mathbf{x}', i) = G(\mathbf{x}', j)$. We will see below where we describe the update action that merges are possible because any agent i's position x_i may change, which will also modify $Nbrs(\mathbf{x}, i)$ and thus Γ and connectivity. Similarly, we will say two groups have *partitioned* if $\exists i, j \in ID$ such that i was connected to j at state $\mathbf{x}$, but i is not connected to j at state $\mathbf{x}'$; the two groups that partitioned are $G(\mathbf{x}, i) = G(\mathbf{x}, j)$ and $G(\mathbf{x}', i) \neq G(\mathbf{x}', j)$. Where we talk about merges and partitions of non-faulty groups, we assume the groups are elements of $G_{\bar{F}}(\mathbf{x})$ and not $G(\mathbf{x})$. Note that it is insufficient to define group merges and partitions in terms of $|G(\mathbf{x})|$ since it is possible for a single group to partition into two groups, where one of these partitions merges with another group.

At state $\mathbf{x}$, let $L(\mathbf{x}, i)$ (and symmetrically $R(\mathbf{x}, i)$) be the nearest non-faulty agent strictly left (resp. right) of $\mathsf{Agent_i}$, with ties broken arbitrarily (by taking the minimum agent identifier). We will see in Section 3 that since agents start from distinct positions, the only situation in

which $\mathbf{x}.x_i = \mathbf{x}.x_j$ is when at least one of them is faulty. If no such neighbor exists, then $L(\mathbf{x}, i)$ and $R(\mathbf{x}, i)$ are defined to be $\perp$. Let $L_S(\mathbf{x}, i)$ (and symmetrically $R_S(\mathbf{x}, i)$) be the nearest unsuspected agent strictly left (resp. right) of $\mathsf{Agent}_\mathsf{i}$ 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 $\{\perp\} \cup \mathit{Nbrs}(\mathbf{x}, i) \setminus \mathbf{x}.\mathit{Susp}_i$. An unsuspected $\mathsf{Agent}_\mathsf{i}$ with both unsuspected left and right neighbors is called a *middle agent*. Let the set of middle agent identifiers be $\mathit{Mids}(\mathbf{x})$ for a state $\mathbf{x}$. An unsuspected $\mathsf{Agent}_\mathsf{i}$ without an unsuspected left neighbor is called a *head agent*. If $\mathsf{Agent}_\mathsf{i}$ is unsuspected, is not a head agent, and does not have an unsuspected right neighbor, then it is called a *tail agent*. For a state $\mathbf{x}$, we define

$$\begin{aligned}
\mathit{Heads}(\mathbf{x}) &\triangleq \{i \in \bar{S}(\mathbf{x}) : L_S(\mathbf{x}, i) = \perp\},\\
\mathit{Tails}(\mathbf{x}) &\triangleq \{i \in \bar{S}(\mathbf{x}) : L_S(\mathbf{x}, i) \neq \perp \wedge R_S(\mathbf{x}, i) = \perp\},\\
\mathit{Mids}(\mathbf{x}) &\triangleq \bar{S}(\mathbf{x}) \setminus (\mathit{Heads}(\mathbf{x}) \cup \mathit{Tails}(\mathbf{x})), \text{ and}\\
\mathit{RMids}(\mathbf{x}) &\triangleq \mathit{Mids}(\mathbf{x}) \setminus \{R(\mathbf{x}, i) : i \in \mathit{Heads}(\mathbf{x})\}.
\end{aligned}$$

Also for a state $\mathbf{x}$ and a group $g \in G(\mathbf{x})$, we define

$$\begin{aligned}
H(\mathbf{x}, g) &\triangleq \mathit{Heads}(\mathbf{x}) \cap g, \text{ and}\\
T(\mathbf{x}, g) &\triangleq \mathit{Tails}(\mathbf{x}) \cap g.
\end{aligned}$$

For a state $\mathbf{x}$ and group $g \in G(\mathbf{x})$, each of $H(\mathbf{x}, g)$ and $T(\mathbf{x}, g)$ are singletons, so by abuse of notation, we will refer to $H(\mathbf{x}, g)$ and $T(\mathbf{x}, g)$ as the element in the singleton. Likewise, if $\mathit{Heads}(\mathbf{x})$ or $\mathit{Tails}(\mathbf{x})$ are singletons, we drop the group in the definitions of $H(\mathbf{x}, g)$ and $T(\mathbf{x}, g)$, and denote the singleton element by $H(\mathbf{x})$ and $T(\mathbf{x})$, respectively. Observe that the following hold for a state $\mathbf{x}$,

$$\begin{aligned}
\mathit{Heads}(\mathbf{x}) &\equiv \min_{g \in G(\mathbf{x}), i \in \bar{S}(\mathbf{x}) \cap g} \mathbf{x}.x_i,\\
\mathit{Tails}(\mathbf{x}) &\equiv \max_{g \in G(\mathbf{x}), i \in \bar{S}(\mathbf{x}) \cap g} \mathbf{x}.x_i, \text{ and}\\
|\mathit{Heads}(\mathbf{x})| &\geq |\mathit{Tails}(\mathbf{x})|.
\end{aligned}$$

The last relation reiterates the observation that agents without any communication neighbors are defined to be heads.

Distributed Detection of a Strong Flock. The distributed flocking algorithm executed at $\mathsf{Agent}_\mathsf{i}$ uses two separate processes (threads): (a) a process for taking distributed global snapshots, and (b) a process for updating the target position for $\mathsf{Agent}_\mathsf{i}$.

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 [15]—by a head agent. This global snapshot is used only to detect a stable global predicate, not the entire state of the system. This stable predicate influences the target computation for a head agent. Although we have not modeled this explicitly, we assume that the $\mathsf{snapStart}_i$ transition is performed periodically by a head agent; a new snapshot starts only after the previous one completes. Note that several instances of the global snapshot algorithm may be running simultaneously, one instance for each element in $\mathit{Heads}(\mathbf{x})$. If the global predicate holds at some state along the execution fragment, then $\mathsf{snapEnd}(\mathit{true})$ occurs, otherwise $\mathsf{snapEnd}(\mathit{false})$ occurs; see [15] for more details. Chandy-Lamport's algorithm can be applied since (a) we are detecting a stable predicate and (b) the stable predicate being detected is reachable. The algorithm also requires connectivity of the communications graph, which we will assume in all sections except Section 3.5. In Section 3.5, we will see that the non-faulty communication groups we are concerned about do not partition, and thus we can still apply the algorithm, assuming messages are restricted to non-faulty groups. 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.

Flocking Algorithm. The update transition models the evolution of all (faulty and non-faulty) agents over a synchronous round. For the purpose of presentation, we have decomposed it into four parts: *Mitigate*, *Target*, *Quant*, and *Move*, which are executed in sequence for updating the state of System. The entire update is instantaneous and atomic; the breakdown into parts are used for clarity of presentation. To be clear, for $\mathbf{x} \overset{\mathsf{update}}{\rightarrow} \mathbf{x}'$, $\mathbf{x}'$ is obtained by applying each of these subroutines. By abuse of notation, we refer to the intermediate states after *Mitigate*, *Target*, *Quant*, and *Move* as $\mathbf{x}_M$, $\mathbf{x}_T$, $\mathbf{x}_Q$, and $\mathbf{x}_V$, respectively. That is, we let $\mathbf{x}_M \triangleq \mathit{Mitigate}(\mathbf{x})$, $\mathbf{x}_T \triangleq \mathit{Target}(\mathbf{x}_M)$, etc., and note $\mathbf{x}' = \mathbf{x}_V = \mathit{Move}(\mathbf{x}_Q)$.

Mitigate is executed by non-faulty agents and may cause them to change lanes, mitigating the effect faults have on the system properties introduced below in Section 2.3. *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 5, Line 32). Since the goal is always left of the tail agents, the tail agents attempt to maintain r_f distance from their nearest unsuspected left neighbor (Figure 5, Line 31). As described above, head agents periodically invoke a global snapshot and attempt to detect a certain stable global predicate Flock_S (defined below), indicating that the agents have formed a tight flock formation and are nearly equidistant. If this predicate is detected, then the detecting head agent moves towards the goal (Figure 5, 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. Quantization is a key requirement for any 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* updates agent positions x_i toward the quantized targets u_i. Note that *Move* abstractly captures the physical evolution of the system over a round, and thus is the time-abstract transition corresponding to physical evolution over an interval of time.

2.3 Key System Properties

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,

$$\begin{aligned} \mathit{Safety}(\mathbf{x}) \triangleq &\forall i, j \in \mathit{ID}, i \neq j, \mathbf{x}.\mathit{lane}_i = \mathbf{x}.\mathit{lane}_j \\ &\implies |\mathbf{x}.x_i - \mathbf{x}.x_j| \geq r_s. \end{aligned}$$

We assume that in all initial states $\mathbf{x} \in Q_0$ of System, $\mathit{Safety}(\mathbf{x})$ and $\mathbf{x}.x_{H(\mathbf{x})} \geq 0$. When failures occur, a reduced inter-agent gap of r_r will be guaranteed. We call this weaker property *reduced safety*:

$$\begin{aligned} \mathit{Safety}_R(\mathbf{x}) \triangleq &\forall i \in \bar{F}(\mathbf{x}), \forall j \in \mathit{ID}, i \neq j, \mathbf{x}.\mathit{lane}_i = \mathbf{x}.\mathit{lane}_j \\ &\implies |\mathbf{x}.x_i - \mathbf{x}.x_j| \geq r_r. \end{aligned}$$

The following predicates are defined in terms of communication groups, so for each of the following fix some state $\mathbf{x}$ and some group $g \in G(\mathbf{x})$. If there is a single non-faulty group, then we omit the group identifier g. An *ϵ-flock* for group g 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,

$$\begin{aligned} \mathit{Flock}(\mathbf{x}, \epsilon, g) \triangleq &\forall i \in g \cap \bar{S}(\mathbf{x}), L_S(\mathbf{x}, i) \neq \perp, \\ &\left|\mathbf{x}.x_i - \mathbf{x}.x_{L_S(\mathbf{x},i)} - r_f\right| \leq \epsilon. \end{aligned}$$

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 instantiated respectively as

$$\begin{aligned} \mathit{Flock}_W(\mathbf{x}, g) &\triangleq \mathit{Flock}(\mathbf{x}, \delta, g), \text{ and} \\ \mathit{Flock}_S(\mathbf{x}, g) &\triangleq \mathit{Flock}(\mathbf{x}, \frac{\delta}{2}, g). \end{aligned}$$

Related to quantization, we have the *no big moves (NBM)* predicate, where none of the non-faulty agents—except possibly a head agent—have any valid moves, because their computed targets are less than β, the quantization constant, away from their current positions.

$$\begin{aligned} \mathit{NBM}(\mathbf{x}, g) \triangleq &\forall i \in g \cap \bar{F}(\mathbf{x}), L_S(\mathbf{x}, i) \neq \perp, \\ &|\mathbf{x}_T.u_i - \mathbf{x}.x_i| \leq \beta, \end{aligned}$$

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 leftmost non-faulty head agent—let $LH(\mathbf{x}) = \min_{i \in \mathit{Heads}(\mathbf{x}) \cap \bar{F}(\mathbf{x})} \mathbf{x}.x_i$ be this agent's identifier—is within β distance of the goal—recall that we assumed it to be the origin without loss of generality—that is,

$$\mathit{Goal}(\mathbf{x}) \triangleq \mathbf{x}.x_{LH(\mathbf{x})} \in [0, \beta]\,.$$

Finally, a state satisfies the *Terminal* predicate if it satisfies both *Goal* and *NBM*.

An outline of these properties is presented in Figure 6. In this figure, start states Q_0 at least satisfy *Safety* as mentioned at the start of this section. Fail-free executions and execution fragments α_{ff} are represented by solid black lines ending with arrows, an execution fragment with failures is indicated by the red dash-dotted line, and an execution where failure detection occurs as shown by the green dashed line. *Safety* is shown to be invariant along any fail-free execution. Figure 6 shows that eventually *NBM*—and thus also Flock_W and Flock_S—is satisfied along any fail-free execution. After Flock_S is satisfied and the global snapshot algorithm terminates, a head agent may move toward states satisfying *Goal* causing both *NBM* and Flock_S to break, while Flock_W remains stable; this all occurs many times as shown. However, along executions with failures (the red dashed-dotted line ending in a diamond), *Safety* is not necessarily upheld. But the reduced safety property Safety_R is invariant when combined with a failure detector, as shown by the execution with failure detection (the green dashed line ending in a circle). Without failure detection, a fail-free execution fragment beginning from a state where there are undetected faults may leave Safety_R as shown by the black line going outside all the predicates, thus, failure detection is necessary to maintain even the reduced safety property. Upon all detectable failures being detected by their neighbors, any fail-free execution is then guaranteed to reach states satisfying *NBM* (and thus also Flock_S) and also to eventually reach *Goal*.

3 Analysis of Safe Flocking

The main result of the paper (Theorem 3.1) is that the algorithm in Figure 5 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 in some lane such that each non-faulty agent can safely avoid faulty agents by changing lanes and eventually make progress. For the first part of our analysis,

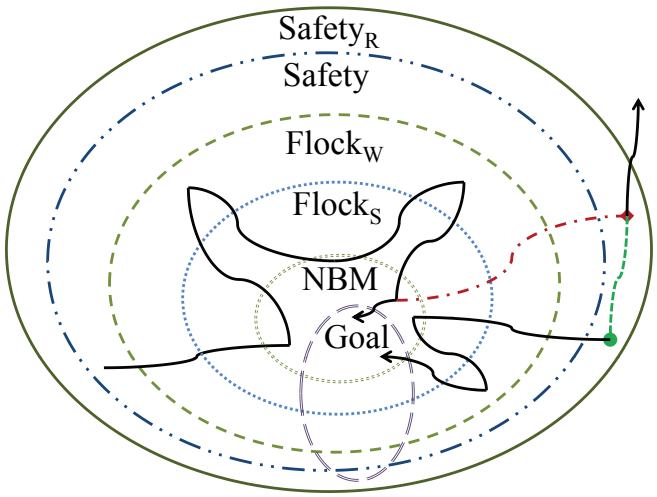

Figure 6: Fail-free executions and executions with faults, and different properties satisfied under these executions.

we will simply assume that any faulty agent is detected within a finite number of rounds. In Section 3.4, we shall examine conditions under which this holds.

We begin with some assumptions.

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

(I) All initial states $\mathbf{x} \in Q_0$ satisfy $Safety(\mathbf{x})$ and $\mathbf{x}.x_{H(\mathbf{x})} \geq 0$,

(II) $r_r < r_s < r_f < r_c$: the reduced safety gap r_r required when faulty agents are present is strictly less than the safety gap r_s in the absence of faulty agents, which in turn is strictly less than the flocking distance r_f, all of which are less than the communication distance r_c,

(III) $\frac{\delta}{2} < r_c$: this prevents head agents from moving beyond the communications radius of their nearest unsuspected right agent, that is, it prevents motion of non-faulty head agents from partitioning the communication graph, and

(IV) $0 < v_{min} \leq v_{max} \leq \beta \leq \frac{\delta}{4N}$: this upper bounds the maximum velocity to be less than or equal to the quantization parameter β. This is necessary to prevent agents from overshooting computed targets u_i, which could violate safety. Finally, β is upper bounded to ensure $NBM \subseteq Flock_S$, where intuitively, the bound on β guarantees that errors from flocking due to quantization do not accumulate along the flock from a head agent to a tail agent. This is used to show that eventually $Flock_S$ is satisfied by showing eventually NBM is reached.

Also, in most of this section, we assume that the graph of non-faulty agents $\Gamma_{\bar{F}}(\mathbf{x})$ is connected for all reachable states $\mathbf{x}$, thus the graph of non-faulty agents cannot partition; this restricts the type of actuator faults that are allowed to occur. This assumption is relaxed to require no connectivity requirement at all in Section 3.5.

Theorem 3.1. *Let us assume that there exists a finite number of rounds k_d within which every faulty agent is suspected, and $v_{max} \leq (r_s - r_r)/(2k_d)$. Let $\mathbf{x}$ be any state reachable after at most f faults.*

(Non-Blocking Faults) Suppose that for every $\mathbf{x}' \in FFReach(\mathbf{x})$, for every non-faulty agent $i \in \bar{F}(\mathbf{x}')$, there is a lane $\mathcal{L} \in LD$ such that there is no faulty agent $j \in F(\mathbf{x}')$ in lane $\mathcal{L}$ with position $\mathbf{x}'.x_j$ in the range $[\mathbf{x}'.x_i - r_s - 2v_{max}k_d, \mathbf{x}'.x_i + r_s + 2v_{max}k_d]$.

Then, (a) $FFReach(\mathbf{x}) \subseteq Safety_R$, that is, all fail-free states reachable from $\mathbf{x}$ satisfy the reduced safety property, and (b) every fail-free execution fragment starting from a state $\mathbf{x}' \in FFReach(\mathbf{x})$ eventually satisfies Terminal and $Flock_S$.

The non-blocking faults assumption of Theorem 3.1 states a sufficient spacing requirement of faulty agents, which ensures safety and progress by guaranteeing that at any reachable state, any non-faulty agent always has a sufficiently large space in some other lane where there are no faulty agents. The non-blocking faults assumption is trivially satisfied if the number of lanes is greater than the total number of faults, $N_L > f$, but it is also satisfied with fewer lanes, provided the faulty agents are sufficiently apart in space as stated. Progress will be guaranteed by observing that eventually any non-faulty agent will use its left and right unsuspected neighbors for target computations, which are non-faulty.

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:

$$NBM \subset Flock_S \subset Flock_W \subset Safety \subset Safety_R,$$

as previously shown in Figure 6.

3.1 Communication Neighbor Properties

The following lemma ensures that the variables $Nbrs_i$, L_i, and R_i of agent i are well defined and match the definitions of $Nbrs(\mathbf{x}, i)$, $L_S(\mathbf{x}, i)$, and $R_S(\mathbf{x}, i)$, respectively, for any reachable state $\mathbf{x}$.

Lemma 3.1. *For any reachable state $\mathbf{x}$, for all $i \in ID$, $\mathbf{x}.Nbrs_i = Nbrs(\mathbf{x}, i)$, $\mathbf{x}.L_i = L_S(\mathbf{x})$, and $\mathbf{x}.R_i = R_S(\mathbf{x})$.*

The next lemma states that if the unsuspected neighbor left of an agent i changes to some agent j, then the unsuspected neighbor right of agent j becomes i; that is,

the neighbors used in target computation switch symmetrically. This is used to establish safety upon non-faulty agents no longer relying on suspected agents for target computation. The proof follows by observing only suspect or update modify L_S, R_S, or $Nbrs$, and then observing that $Nbrs$ is defined symmetrically.

Lemma 3.2. *For any reachable state* $\mathbf{x}$*, if* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for any* $a \in A$*,* $\forall i, j \in ID$*, if the agent left of* i *is not* j *at state* $\mathbf{x}$*, but is* j *at state* $\mathbf{x}'$*, then at state* $\mathbf{x}$*,* i *is the agent right of* j*, that is, if* $\mathbf{x}.L_i \neq j$ *and* $\mathbf{x}'.L_i = j$*, then* $\mathbf{x}'.R_j = i$*.*

Next, we remark that there exist failures which do not violate safety. Assume there are no faulty agents at a state $\mathbf{x}$, and consider a sequence of actuator faults which causes every agent to move with the identical failure velocity. Then, for any state $\mathbf{x}_s$ appearing after $\mathbf{x}$ in an execution, $Safety(\mathbf{x}_s)$ holds, since for all future rounds the agents will have moved at the same velocity and thus the difference of their positions is constant.

Sharing Suspected Sets. The motivation for sharing sets of suspected agents among neighbors in Figure 5, Line 21, is illustrated by the following observation. We give a failure configuration under which no movement is possible and hence no progress can be made if the sets of suspected agents are not shared. Assume that neighbors do not share sets of suspected agents, so Line 21 in Figure 5 is deleted. Let $\mathbf{x}$ be a fail-free reachable state and let agents $i \in ID \setminus Heads(\mathbf{x})$ be spaced evenly at a distance greater than $r_f + \frac{\delta}{2}$ such that $\neg Flock_S(\mathbf{x})$, and particularly

$$\begin{aligned}\mathbf{x}.x_i - \mathbf{x}.x_{\mathbf{x}.L_i} =& \mathbf{x}.x_{\mathbf{x}.R_i} - \mathbf{x}.x_i \\ =& \vdots \\ =& \mathbf{x}.x_{T(\mathbf{x})} - \mathbf{x}.x_{\mathbf{x}.L_{T(\mathbf{x})}} > \quad r_f \pm \frac{\delta}{2}.\end{aligned}$$

Let there be a non-faulty agent p which is located farther than r_c to the left of the last 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 $\mathbf{x}'$, $\mathbf{x}''.Susp_p = \emptyset$, since only the last agent could be suspected and only its neighbors suspect it, so that $\forall j \in Nbrs(\mathbf{x}'', T(\mathbf{x}''))$, $p \in \mathbf{x}''.Susp_j$, but p is not a neighbor. Hence no progress is made as p never learns it must change lanes, so $\forall i \in ID$, $\mathbf{x}''.x_i = \mathbf{x}'.x_i$ and no progress is made toward flocking or the goal, but safety remains invariant.

3.2 Safety

First, we establish that System satisfies the safety part of the safe flocking problem. The following lemma states that any two agents move towards or away from one another by at most $2v_{max}$ from one round to another. The proof follows from the fact that in each round, each agent moves by at most v_{max}, which follows immediately from Figure 5, Line 37.

Lemma 3.3. *For any reachable state* $\mathbf{x}$*, if* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for any* $a \in A$*,* $\forall i, j \in ID$ *such that* $i \neq j$*, then* $|(\mathbf{x}'.x_i - \mathbf{x}.x_i) \pm (\mathbf{x}'.x_j - \mathbf{x}.x_j)| \leq 2v_{max}$*.*

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

Lemma 3.4. *For any reachable state* $\mathbf{x}$*, if* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *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 $i \in ID$. By Lemma 3.1, we discuss L and R. By Lemma 3.2, 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 $\mathbf{x}'.x_{\mathbf{x}'.L_i} = \mathbf{x}'.x_j$, and let $a = \mathbf{x}.L_j$ and $b = \mathbf{x}.R_i$. Now,

$$\mathbf{x}'.x_j = \frac{\mathbf{x}.x_a + \mathbf{x}.x_i}{2}, \text{ and } \mathbf{x}'.x_i = \frac{\mathbf{x}.x_j + \mathbf{x}.x_b}{2},$$

and thus

$$\begin{aligned}\mathbf{x}'.x_i - \mathbf{x}'.x_j &= \frac{\mathbf{x}.x_j + \mathbf{x}.x_b}{2} - \frac{\mathbf{x}.x_a + \mathbf{x}.x_i}{2} \\ &= \frac{\mathbf{x}.x_j - \mathbf{x}.x_a + \mathbf{x}.x_b - \mathbf{x}.x_i}{2}.\end{aligned}$$

Finally, by the hypothesis that $\mathbf{x}.x_j - \mathbf{x}.x_a \geq c$ and $\mathbf{x}.x_b - \mathbf{x}.x_i \geq c$, we have

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

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 3.2 states 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 3.2. *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 $Safety$ and that $Safety \Rightarrow Safety_R$. For the inductive case, for each transition $a \in A$, we show if $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ and $\mathbf{x} \in Safety_R$, then $\mathbf{x}' \in Safety_R$. The $\mathsf{fail}_\mathsf{i}(v)$, $\mathsf{snapStart}_\mathsf{i}$, $\mathsf{snapTerm}_\mathsf{i}$, and $\mathsf{suspect}_\mathsf{i}$ transitions do not modify any x_i or u_i, so $Safety_R$

is preserved. 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 3.4. 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 5, Lines 28–32 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 34 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 this, we must consider further states in the executions, namely up to the time at which detection occurs. For simplicity of presentation, assume we are dealing with any fail-free execution fragment starting with $\mathbf{x}$, under which case, the detection time until all faulty agents 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 faulty agent j, $\mathbf{x}'.x_j = \mathbf{x}.x_j + \mathbf{x}.vf_j$ by Line 36. Given the assumption that $v_{max} \leq \frac{r_s - r_r}{2k_d}$, it is the case that at round k_d for the corresponding state $\mathbf{x}_d$, $\mathbf{x}_d.x_j \leq \mathbf{x}.x_j + k_d v_{max} = \mathbf{x}.x_j + \frac{r_s - r_r}{2}$ where we considered the case for $\mathbf{x}.v_j > 0$ and the negative failure velocity case follows symmetrically. By the assumption that any faulty agent is detected by round k_d and by Lemma 3.3 and Figure 5, Line 37, any faulty agent j and any non-faulty agent i have moved toward one another by at most $2k_d v_{max}$, and thus $|\mathbf{x}_d.x_j - \mathbf{x}_d.x_i| \geq 2k_d v_{max} \geq r_r$.

This implies at least $Safety_R(\mathbf{x}_m)$ for any states $\mathbf{x}_m$ in any state in any 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 $j \in \mathbf{x}_d.Susp_i$, which changes $L_S(\mathbf{x}_d)$ and $R_S(\mathbf{x}_d)$, but now applying Lemma 3.4 shows there is at least r_r distance between i and j. Finally, by Figure 5, Line 27, $\mathbf{x}'_d.lane_i \neq \mathbf{x}_d.lane_j$, since $N_L \geq 2$, and by the non-blocking faults assumption of Theorem 3.1, $Safety_R(\mathbf{x}'_d)$ since the definition can only be violated if agents reside on the same lane. □

3.3 Progress Toward Flock Formation and Goal

In the following progress analysis, we work with fail-free execution fragments, beginning from an arbitrary state that is reached after f faults. We define the set of states which are reachable from initial states $\mathbf{x}_0 \in Q_0$ after f actuator faults as,

$$\mathbf{X}_f \triangleq \{\mathbf{x} \in Reach(\mathbf{x}_0) : \mathbf{x}_0 \in Q_0 \text{ and } |F(\mathbf{x})| = f\}.$$

In the remainder of this section, we fix α_{ff} to be an arbitrary infinite fail-free execution fragment starting from a state $\mathbf{x}_f \in \mathbf{X}_f$. So, for any state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, $F(\mathbf{x}) = F(\mathbf{x}_f)$. Thus, for any state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, we fix respectively the sets of faulty and non-faulty agent identifiers as:

$$F \triangleq F(\mathbf{x}_f) \text{ and } \bar{F} \triangleq \bar{F}(\mathbf{x}_f).$$

Fail-free execution fragments are frequently used to show convergence from an arbitrary state back to a stable set [21], albeit we note that we are dealing with permanent faults instead of transient ones. In our case, the stable set eventually reached are states where *Terminal* is satisfied.

Recall that f is assumed to be the number of actuator faults which occur in Theorem 3.1, so $\mathbf{X}_f$ is the set of reachable states for all possible actuator faults under consideration. Under the assumptions of Theorem 3.1—faults are detected sufficiently fast and are non-blocking—by Invariant 3.2, we know that $\mathbf{X}_f \subseteq Safety_R$, and since this is an invariant, we have that $FFReach(\mathbf{X}_f) \subseteq Safety_R$.

Influence of Faulty Agents on Progress. First observe that, like safety, progress may be violated by faulty agents. Any faulty agent with nonzero failure velocity diverges, and therefore, cannot be a part of a flock that comes to rest at the goal. This observation also highlights why *Flock* is defined over unsuspected agents and not the set of all non-faulty agents. Faulty agents with zero velocity could also cause progress to be violated, where a "wall" of faulty agents may prevent non-faulty agents from making progress, but such situations are excluded by non-blocking faults assumption of Theorem 3.1.

Progress along Fail-Free Execution Fragments. In the remainder of this section, we show that once new actuator faults 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 following descriptions of error dynamics are used

in the analysis:

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

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

Here $e(\mathbf{x},i)$ gives the error with respect to r_f of $\mathsf{Agent_i}$ and its unsuspected 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 37 of Figure 5 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 5, Line 34 is satisfied, but this does not imply $v_{min} = 0$. The lemma follows from Line 37 of Figure 5.

Lemma 3.5. *For two adjacent rounds $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ in $FFReach(\mathbf{X}_f)$, for any non-faulty agent i, if $|\mathbf{x}_{k,T}.u_i - \mathbf{x}_k.x_i| > \beta$, then*

$$|\mathbf{x}_{k+1}.x_i - \mathbf{x}_k.x_i| \geq v_{min} > 0,$$

where we recall that $\mathbf{x}_{k,T}$ is the state obtained applying the subroutines of the update *transition through Target to the state $\mathbf{x}_k$.*

The next lemma states that from any reachable state $\mathbf{x}$ outside *NBM*, the maximum error from flocking 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 3.6. *For any fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ for some $a \in A$, and $\neg NBM(\mathbf{x})$ is satisfied, then*

$$\max_{i \in \bar{F}} e(\mathbf{x}',i) \leq \max_{i \in \bar{F}} e(\mathbf{x},i).$$

Proof. Only an $\mathsf{update_i}$ changes any u_i or x_i, so we only consider it, as in the other cases, equality follows. *Target* and *Quant* are the only subroutines of $\mathsf{update_i}$ to modify u_i. Now, let $\mathbf{x}_T = Target(\mathbf{x})$, and we have

$$\max_{i \in \bar{F}} eu(\mathbf{x}_T,i) \leq \max_{i \in \bar{F}} eu(\mathbf{x},i),$$

which follows from $eu(\mathbf{x}_T,i)$ being computed as convex combinations of positions from $\mathbf{x}$. In the following, we refer to the unsuspected left and right neighbors of an agent i, so let $l = \mathbf{x}_T.L_i$ and $r = \mathbf{x}_T.R_i$. Specifically the convex combinations are,

$$\begin{aligned}
i \in Heads(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T,i) = 0, \\
i = \mathbf{x}_T.R_j \text{ for } j \in Heads(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T,i) = \frac{eu(\mathbf{x},r)}{2}, \\
i \in RMids(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T,i) = \frac{eu(\mathbf{x},l) + eu(\mathbf{x},r)}{2}, \\
i \in Tails(\mathbf{x}_T) &\Rightarrow eu(\mathbf{x}_T,i) = \frac{eu(\mathbf{x},l) + eu(\mathbf{x},i)}{2}.
\end{aligned}$$

Next, *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

$$\begin{aligned}
i \in Heads(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = 0, \\
i = \mathbf{x}_Q.R_j \text{ for } j \in Heads(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = eu(\mathbf{x},i), \\
i \in RMids(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = \frac{eu(\mathbf{x},r) + eu(\mathbf{x},i)}{2}, \\
i \in Tails(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = \frac{eu(\mathbf{x},i) + eu(\mathbf{x},l)}{2}.
\end{aligned}$$

Similarly, if u_l is quantized and u_i is not quantized, then e_i is computed as

$$\begin{aligned}
i \in Heads(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = 0, \\
i = \mathbf{x}_Q.R_j \text{ for } j \in Heads(\mathbf{x}_Q) &\Rightarrow \\
&\qquad eu(\mathbf{x}_Q,i) = \frac{eu(\mathbf{x},i) + eu(\mathbf{x},r)}{2}, \\
i \in RMids(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = \frac{eu(\mathbf{x},l) + eu(\mathbf{x},i)}{2}, \\
i \in Tails(\mathbf{x}_Q) &\Rightarrow eu(\mathbf{x}_Q,i) = \frac{eu(\mathbf{x},i)}{2}.
\end{aligned}$$

Finally, by Figure 5, Line 37, the maximum error between actual positions and not just target positions is non-increasing. □

The next lemma shows that if the head agent does not move, then *NBM* is a stable predicate, a state satisfying *NBM* is reached, and gives an $O(N)$ bound on the number of rounds required to reach such a state. Define a candidate Lyapunov-like function at a state $\mathbf{x}$ as

$$V(\mathbf{x}) \triangleq \sum_{i \in \bar{F}} e(\mathbf{x},i).$$

Lemma 3.7. *For any fail-free reachable state $\mathbf{x}_k \in FFReach(\mathbf{X}_f)$, if the head agent's position is fixed and $\neg NBM(\mathbf{x}_k)$ is satisfied, then the* update *transition decreases $V(\mathbf{x}_k)$ by at least a positive constant ψ. Furthermore, there exists a state reachable from $\mathbf{x}_k$ in a finite c number of rounds, $\mathbf{x}_c \in FFReach(\mathbf{x}_k, c)$, and there exists a constant $\sigma \geq 0$, if $V(\mathbf{x}_c) \leq \sigma$, then $NBM(\mathbf{x}_c)$ and $k < c \leq \lceil V(\mathbf{x}_k - \sigma)/\psi \rceil$, where $\psi = v_{min}$ and c is $O(N)$.*

Proof. First, $V(\mathbf{x}_k)$ is nonnegative since it is a sum of nonnegative terms. Since $NBM(\mathbf{x}_k)$ is not satisfied, by the definition of $NBM(\mathbf{x}_k)$, there are some non-faulty agents with control values and positions which do not satisfy the quantization condition, so let this set of agents be $M \triangleq \{j \in \bar{F}(\mathbf{x}) : |\mathbf{x}_{k,T}.x_j - \mathbf{x}_k.x_j| > \beta\}$, where we recall that $\mathbf{x}_{k,T}$ is the state obtained by applying the subroutines of the update transition to $\mathbf{x}_k$ through *Target*. Thus each $j \in M$ will have a position at the next round $\mathbf{x}_{k+1}$ which satisfies $v_{max} \geq |\mathbf{x}_{k+1}.x_j - \mathbf{x}_k.x_j| \geq v_{min}$ by the definition of position updates and by Lemma 3.5, which says that if an agent moves, it moves by at least a positive constant $v_{min} > 0$. 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 $\psi < 0$. By applying the same reasoning using convex combinations from Lemma 3.6, we have that at the very least, there is one agent $j \in M$, so we have $-v_{min} \geq \Delta V(\mathbf{x}_k, \mathbf{x}_{k+1})$, and thus we fix $\psi = v_{min}$ since $v_{min} > 0$. 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}.

Next, we show for some constant $\sigma > 0$, for a state $\mathbf{x}_c$ reachable from $\mathbf{x}_k$, if $V(\mathbf{x}_c) \leq \sigma$, then $NBM(\mathbf{x}_c)$ is satisfied. To accomplish this, we will bound the quantization constant β used to define NBM using σ. For the following, we will discuss the unsuspected left and right neighbors of some agent i, so let $\mathbf{x}_c.L_i = l$ and $\mathbf{x}_c.R_i = r$ for brevity, and likewise let the set of non-faulty middle agents be $\bar{M} = Mids(\mathbf{x}_c) \cap \bar{F}$. A sufficient condition for $V(\mathbf{x}_c) \leq \sigma$, is $e(\mathbf{x}_c, i) \leq \frac{\sigma}{|\bar{F}|}$, for each non-faulty middle or tail agent i, since each $e(\mathbf{x}_c, i)$ is nonnegative.

We now derive a constraint on β using $\frac{\sigma}{|\bar{F}|}$ which implies $NBM(\mathbf{x}_c)$ is satisfied. For each non-faulty middle agent, we have from the definition of NBM and the *Target* computation, that

$$\bigwedge_{i \in \bar{M}} \left| \frac{\mathbf{x}_c.x_l + \mathbf{x}_c.x_r}{2} - \mathbf{x}_c.x_i \right| \leq \beta.$$

Now, rearranging terms and adding zero (by adding and subtracting r_f), we have

$$\bigwedge_{i \in \bar{M}} \left| \frac{\mathbf{x}_c.x_r - \mathbf{x}_c.x_i - r_f - \mathbf{x}_c.x_i + \mathbf{x}_c.x_l + r_f}{2} \right| \leq \beta,$$

each of which we see is a sum of error terms, so

$$\bigwedge_{i \in \bar{M}} \left| \frac{e(\mathbf{x}_c, r) - e(\mathbf{x}_c, i)}{2} \right| \leq \beta.$$

Since it could be the case that either of these terms are zero, we assume without loss of generality that the first term is zero to maximize the sum, and have

$$\bigwedge_{i \in \bar{M}} \frac{e(\mathbf{x}_c, i)}{2} \leq \beta.$$

Finally, by the condition $e(\mathbf{x}_c, i) \leq \frac{\sigma}{|\bar{F}|}$, we let $\beta \leq \frac{\sigma}{2|\bar{F}|}$ and we have the desideratum

$$\bigwedge_{i \in \bar{M}} e(\mathbf{x}_c, i) \leq 2\beta.$$

The same reasoning we just applied for non-faulty middle agents applies for a non-faulty tail agent $t \in Tails(\mathbf{x}_c)$, since we rewrite the consequent as $\left| \frac{\mathbf{x}_c.x_t + \mathbf{x}_c.x_l + r_f}{2} - \mathbf{x}_c.x_t \right| = e(\mathbf{x}_c, t) \leq \beta$. Thus by the condition that $e(\mathbf{x}_c, t) \leq \frac{\sigma}{|\bar{F}|}$, by letting $\beta \leq \frac{\sigma}{2|\bar{F}|}$, then we have $e(\mathbf{x}_c, t) \leq \beta$ as desired. Since every non-faulty agent $i \in \bar{F}$ satisfies $|\mathbf{x}_{c,T}.u_i - \mathbf{x}_c.x_i| \leq \beta$, we have that $NBM(\mathbf{x}_c)$ is satisfied if $V(\mathbf{x}_k) \leq 2|\bar{F}|\beta \leq \sigma$, where we recall that $\mathbf{x}_{c,T}$ is the state obtained by applying the subroutines of the update transition to $\mathbf{x}_c$ through *Target*.

By repeated application of the reasoning that $V(\mathbf{x}_k)$ decreases by at least v_{min} at round $k+1$, there exists such a c with $k < c \leq \left\lceil \frac{V(\mathbf{x}_k) - \sigma}{v_{min}} \right\rceil$ such that $NBM(\mathbf{x}_c)$ is satisfied since $V(\mathbf{x}_c) \leq \sigma$. To show c is $O(N)$ rounds from $\mathbf{x}_k$, we observe for the state $\mathbf{x}_k$ that

$$V(\mathbf{x}_k) \leq \sum_{i \in \bar{F}} \max_{i \in \bar{F}} e(\mathbf{x}_k, i) = |\bar{F}| \max_{i \in \bar{F}} e(\mathbf{x}_k, i),$$

the value of $e(\mathbf{x}_k, i)$ is not a function of N for any $i \in \bar{F}$ (aside from being a projection), and $\bar{F} \leq N$. □

The previous lemma stated a bound on the time it takes for System to reach the set of states satisfying NBM, which is linear in the number of non-faulty agents. For the previous lemma to show that $Flock_S$ is eventually satisfied, we require that $Flock_S \supseteq NBM$, and the next lemma states a more general result using ϵ-flocking. The next lemma follows from the analysis used to choose σ in the proof of the previous lemma. This yields $\beta \leq \frac{\delta}{4N}$ from the $\delta/2$ constant used in strong flocking, and that in any reachable state, there are at most N non-faulty agents.

Lemma 3.8. *For any fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, if $\beta \leq \frac{\epsilon}{2N}$, then $NBM(\mathbf{x}) \subseteq Flock(\mathbf{x}, \epsilon)$. Furthermore, if $\beta \leq \frac{\delta}{4N}$, then $NBM(\mathbf{x}) \subseteq Flock_S(\mathbf{x}) \subseteq Flock_W(\mathbf{x})$.*

The next lemma states a bound on the Lyapunov-like candidate V when supposing that ϵ-flocking is satisfied, and also states the same result for the instantiated strong and weak ϵ-flocks. Note that the converse of this lemma is in general false.

Lemma 3.9. *For any reachable state $\mathbf{x}$, if $Flock(\mathbf{x}, \epsilon)$, then $V(\mathbf{x}) \leq \epsilon|\bar{F}|$. Furthermore, if $Flock_W(\mathbf{x})$, then $V(\mathbf{x}) \leq \delta|\bar{F}|$, and if $Flock_S(\mathbf{x})$, then $V(\mathbf{x}) \leq \frac{\delta|\bar{F}|}{2}$.*

Now we observe that for the single non-faulty communication group being analyzed, $Flock_W$ is a stable predicate—that is, once the group forms a weak flock, it remains invariant. This result follows from analyzing the

Target subroutine which computes the new targets for the agents in each round. If $Flock_S$ holds, as learned from the global snapshot algorithm, a head agent moves by at most a fixed distance $\frac{\delta}{2}$, which guarantees that $Flock_W$ is maintained even though $Flock_S$ may be violated. When the head agent does move, because of invariance of $Flock_W$, a bound is given on the amount the Lyapunov candidate V can increase. Recall from Section 2 that if the predicates $Flock_W$, $Flock_S$, or NBM appear without a group identifier, we assumed they refer to the single non-faulty communication group being analyzed in this section; in Section 3.5 we will analyze a similar property for multiple non-faulty groups.

Lemma 3.10. *For any fail-free reachable state* $\mathbf{x} \in FFReach(\mathbf{X}_f)$*, for a state* $\mathbf{x}'$ *such that* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for some* $a \in A$*, if* $Flock_W(\mathbf{x})$*, then* $Flock_W(\mathbf{x}')$*. Furthermore, if* $Flock_S(\mathbf{x})$ *and* $\neg Flock_S(\mathbf{x}')$*, then* $V(\mathbf{x}') \geq V(\mathbf{x})$ *but also* $V(\mathbf{x}') \leq \left|\bar{F}\right| \delta$*.*

Proof. We show for any fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, for any state $\mathbf{x}'$ such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ for any $a \in A$, if $Flock_W(\mathbf{x})$, then $Flock_W(\mathbf{x}')$. First, the only action which modifies positions of agents is update, so it is the only one considered. If $Flock_W(\mathbf{x})$, there are two cases to consider based on whether the stronger property $Flock_S(\mathbf{x})$ is satisfied or not.

In the first case, System satisfies $Flock_W(\mathbf{x}) \wedge \neg Flock_S(\mathbf{x})$, then $Flock_W(\mathbf{x}')$ holds since level sets of the Lyapunov candidate $V(\mathbf{x})$ are invariant by Lemma 3.7, and since the head agent only moves if $Flock_S(\mathbf{x})$ is satisfied, so by the target computation of the head agent, we have $\mathbf{x}'.x_{H(\mathbf{x}')} = \mathbf{x}.x_{H(\mathbf{x}')}$.

In the second case, System satisfies $Flock_W(\mathbf{x}) \wedge Flock_S(\mathbf{x})$. Upon termination of the global snapshot algorithm the head agent learns of strong flocking being satisfied, and we assume without loss of generality that the snapshot has terminated at state $\mathbf{x}$, so we show $Flock_W(\mathbf{x}')$; if it had not yet terminated, pick the round at which it did terminate, since for every round before that, weak flocking would be invariant by the next argument because the head would not move.

If the head is not at the goal, $\mathbf{x}.x_{H(\mathbf{x})} \neq 0$, then the head computes $\mathbf{x}'.u_{H(\mathbf{x}')} < \mathbf{x}.u_{H(\mathbf{x})}$ and applies this target, so $\mathbf{x}'.u_{H(\mathbf{x}')} < \mathbf{x}.u_{H(\mathbf{x})}$. Otherwise, if the head is at the goal $\mathbf{x}.x_{H(\mathbf{x})} = 0$ or if $\mathbf{x}.x_{H(\mathbf{x})} \in [0, \beta]$ such that the head agent's target is quantized, then $\mathbf{x}'.x_{H(\mathbf{x}')} = \mathbf{x}.x_{H(\mathbf{x})}$ and the proof is complete since the head agent does not move. If the head's target is not quantized and was not at the goal or not in the range $[0, \beta]$, then the head will compute a target no more than $\delta/2$ to the left of its position, so $\left|\mathbf{x}'.u_{H(\mathbf{x}')} - \mathbf{x}.u_{H(\mathbf{x})}\right| \leq \delta/2$. Because $Flock_S(\mathbf{x})$, we have for the agent right of the head, $r = \mathbf{x}.R_{H(\mathbf{x})}$, that $\left|\mathbf{x}.x_{H(\mathbf{x})} - \mathbf{x}.x_r - r_f\right| \leq \delta/2$, and now since the head moved by at most $\delta/2$, $\left|\mathbf{x}'.x_{H(\mathbf{x}')} - \mathbf{x}'.x_r - r_f\right| \leq \delta$, so $Flock_W(\mathbf{x})$ is satisfied.

The bound $V(\mathbf{x}') \leq \left|\bar{F}\right| \delta$ follows as it did in Lemma 3.9, by summing the errors of all non-faulty agents since $Flock_W(\mathbf{x}')$. □

The following corollary states that if a head agent does not move, then $Flock_S$ remains satisfied and follows by Lemma 3.7.

Corollary 3.1. *For any fail-free reachable state* $\mathbf{x} \in FFReach(\mathbf{X}_f)$*, for any* $\mathbf{x}'$ *reachable from* $\mathbf{x}$*, if* $Flock_S(\mathbf{x})$ *and* $\mathbf{x}.x_{H(\mathbf{x})} = \mathbf{x}'.x_{H(\mathbf{x}')}$*, then* $Flock_S(\mathbf{x}')$*.*

The next lemma states that eventually no non-faulty agent $j \in \bar{F}$ believes any faulty agent $i \in F$ is its left or right neighbor, and thereby any faulty agents diverge without colliding with non-faulty agents along their own lanes if $|\mathbf{x}.v_i| > 0$.

Lemma 3.11. *For any fail-free reachable state* $\mathbf{x} \in FFReach(\mathbf{X}_f)$*, for any faulty agent* $i \in F$*, there exists a state* $\mathbf{x}'$ *reachable in at most* $O(N)$ *rounds from* $\mathbf{x}$*, such that* $\forall j \in \bar{F}$*,* $\mathbf{x}'.L_j \neq i$ *and* $\mathbf{x}'.R_j \neq i$*.*

The previous lemma—with the non-blocking faults assumption of Theorem 3.1 that provides 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 faults. In particular, after some faulty agent j has been suspected by all its non-faulty neighbors, the *Mitigate* subroutine of the update transition changes all the non-faulty neighbors of j to be in a different lane than the faulty agent j.

The following sequence convergence lemma is used to show that System eventually satisfies the desired properties and terminates. The lemma essentially states that for a lexicographically ordered tuple representing different states of System, that if at least one component of the tuple decreases in every step, and the other component does not increase beyond some bound, then eventually the sequence reaches a fixed-point and the components of the tuple are within specified ranges. The argument employed is similar in spirit to the average dwell-time condition used to show stability of switched systems [22].

Lemma 3.12. *Consider any infinite sequence of lexicographically ordered pairs of nonnegative reals* $\langle a_1, b_1\rangle$*, …,* $\langle a_j, b_j\rangle$*, …. Suppose* $\exists c_1 > 0$*,* $c_2 > 0$*,* $c_3 \geq 0$*,* $c_4 \geq 0$*,* $c_5 \geq 0$*. If, for all* $j \geq 1$*,*

$$\begin{aligned} a_{j+1} = a_j &\Rightarrow b_{j+1} \leq b_j - c_1, \\ a_{j+1} < a_j &\Rightarrow b_{j+1} \leq c_4, \\ a_j > c_3 \wedge b_j \leq c_2 &\Rightarrow a_{j+1} \leq \max\{0, a_j - c_1\}, \\ a_{j+1} = a_j > c_3 \wedge b_{j+1} = b_j &\Rightarrow a_{j+c_5} < a_j. \end{aligned}$$

Then, there exists $t \leq \left\lceil \frac{b_1 - c_2}{c_1} \right\rceil + \left\lceil \frac{a_1 c_4 c_5}{c_1^2} \right\rceil$ *such that for all* $t' \geq t$*,* $\langle a_t, b_t\rangle = \langle a_{t'}, b_{t'}\rangle$*,* $a_t \in [0, c_3]$*, and* $b_t \in [0, c_2]$*.*

Proof. First, note that by assumption, for all $j \geq 2$, a_j is bounded from above by a_{j-1} and below by 0. Likewise, for all $j \geq 2$, b_j is bounded from above by the maximum

of b_{j-1} and c_4, and from below by 0. Now assume for the purpose of contradiction that there exists a pair $\langle a_p, b_p \rangle$ where $a_p > c_3$ such that $\forall p' \geq p$, $\langle a_{p'}, b_{p'} \rangle = \langle a_p, b_p \rangle$. Now we show there exists a $q > p$ such that $\langle a_p, b_p \rangle > \langle a_q, b_q \rangle$.

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 element in the sequence such that $b_n \leq c_2$, thus $b_n \leq c_2 \leq b_p - nc_1$ and $n \leq \left\lceil \frac{b_p - c_2}{c_1} \right\rceil$. The observation that $n \leq \frac{b_p - c_2}{c_1}$ establishes the $\left\lceil \frac{b_1 - c_2}{c_1} \right\rceil$ term of the bound on t. Within the next c_5 elements from n in the sequence, that is at element with index $n + c_5$, it must be the case that $a_{n+c_5} \leq \max\{0, a_n - c_1\}$ since $a_n = a_p > c_3$ and $b_n \leq c_2$. If $a_{n+c_5} = 0$, then the result follows by letting $q = n + c_5 > p$ and since $\langle a_p, b_p \rangle$ becomes smaller at a larger step in the sequence, we reach the contradiction. Otherwise, $a_{n+c_5} = a_n - c_1 < a_n$ and $b_{n+c_5} \leq c_4$, but by the lexicographic ordering of the tuples, $\langle a_{n+c_5}, b_{n+c_5} \rangle < \langle a_n, b_n \rangle$. Thus for $q = n + c_5 > p$, since $\langle a_p, b_p \rangle$ becomes smaller at a larger step in the sequence, we reach the contradiction to the assumption that $\langle a_p, b_p \rangle$ does not decrease, and thus the components of the tuple eventually lie in the desired ranges.

By repeatedly applying the previous arguments, we bound such a t. In particular, the other term in the bound on t follows since b_j is bounded from above by c_4 when a_j increases, we have that a_j may decrease at most $\left\lceil \frac{a_1}{c_1} \right\rceil$ times prior to lying in the range $[0, c_3]$, each of which may be delayed by an additional c_5 steps of no decrease. Thus, b_j would need to decrease from c_4 at most $\left\lceil \frac{a_1}{c_1} \right\rceil$ times, each of which may take at most $\left\lceil \frac{c_4 c_5}{c_1} \right\rceil$ steps, and multiplying these yields the bound. Finally, we then take the ceiling function of these since indices in the sequence are natural numbers, while these values are reals. □

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

Theorem 3.3. *For any fail-free reachable state $\mathbf{x}_0 \in \mathit{FFReach}(\mathbf{X}_f)$, let a fail-free execution fragment starting with state $\mathbf{x}_0$ be $\alpha_{ff} \triangleq \mathbf{x}_0, \mathbf{x}_1, \ldots$. Then, there exists*

$$t \leq \left\lceil \frac{V(\mathbf{x}_0) - \left|\bar{F}\right| \frac{\delta}{2}}{v_{min}} \right\rceil + \left\lceil \frac{\delta \left|\bar{F}\right| \mathbf{x}_0.x_{H(\mathbf{x}_0)} O(N)}{v_{min}^2} \right\rceil,$$

such that for all $t' \geq t$:

(a) $\mathbf{x}_t.x_{H(\mathbf{x}_t)} = \mathbf{x}_{t'}.x_{H(\mathbf{x}_{t'})} \in [0, \beta]$,

(b) $V(\mathbf{x}_t) = V(\mathbf{x}_{t'}) \leq \left|\bar{F}\right| \frac{\delta}{2}$,

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

(d) $\mathit{Flock}_S(\mathbf{x}_t)$.

Proof. Corresponding to the fail-free execution fragment α_{ff}, we consider the infinite sequence of tuples $\langle \mathbf{x}_0.x_{H(\mathbf{x}_0)}, V(\mathbf{x}_0) \rangle$, $\langle \mathbf{x}_1.x_{H(\mathbf{x}_1)}, V(\mathbf{x}_1) \rangle$, …, $\langle \mathbf{x}_t.x_{H(\mathbf{x}_t)}, V(\mathbf{x}_t) \rangle$, …. The proof now follows from Lemma 3.12 by instantiating (a) $c_1 = v_{min}$, by Lemma 3.5, that any agent which moves does so by at least $v_{min} > 0$ and that this is the constant the Lyapunov-like function V decreases by in the convergence to *NBM* result, Lemma 3.7, (b) $c_2 = \left|\bar{F}\right| \frac{\delta}{2}$, by applying $V(\mathbf{x})$ to any state $\mathbf{x}$ that satisfies $\mathit{Flock}_S(\mathbf{x})$ and by Corollary 3.1, (c) $c_3 = \beta$, by Figure 5, Line 34, that agents do not move if their targets are not farther than the quantization constant β from their current positions, (d) $c_4 = \left|\bar{F}\right| \delta$ by computing the value of $V(\mathbf{x})$ for any state $\mathbf{x}$ that satisfies $\mathit{Flock}_W(\mathbf{x})$, by Lemma 3.10, that Flock_W is a stable predicate, and (e) $c_5 \leq O(N)$, from the $O(N)$ termination time of the global snapshot algorithm. □

3.4 Directional Failure Detectors

First we give an overview of the directional failure detector (DFD) used by our flocking algorithm, which is encoded as the precondition of the 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 $\mathsf{suspect}_i(j)$ is satisfied, j is added to Susp_i. This precondition checks that either j moved when it should not have due to being quantized, or that j's actual movement was in the opposite direction from its target u_j.

In Sections 3.1, 3.2, and 3.3, we proved safety and progress of the flocking algorithm assuming that all actuator faults are detected within a finite number of rounds. Unfortunately, there exist faults which cannot be detected at all by DFDs. A trivial example 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. Another example is a faulty agent not connected to any non-faulty agent. While such faults are undetectable in any number of rounds, these faults do not violate *Safety* or *Terminal*. It turns out that only faults which cause a violation of safety or progress may be detected by DFDs.

Lower-Bound on Detection Time. We identify the class of actuator faults that can be detected by DFDs and show a lower-bound on detection time. The following lower-bound applies for executions beginning from states that do not *a priori* satisfy *Terminal*. It intuitively says that a faulty 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 3.7. We show this by establishing that there is an execution with faults which is indistinguishable from a fail-free execution until at least $O(N)$ rounds have occurred.

Lemma 3.13. *There are actuator faults which cannot be detected in fewer than $O(N)$ rounds by a DFD.*

Proof. Consider a fail-free execution fragment α_f which begins with a state where there is a single faulty agent i, and a fail-free execution fragment α_n which begins with a state without any faulty agents. Let the initial state $\mathbf{x}$ of both these executions be the same (except that the head agent is faulty with zero failure velocity in α_f) and satisfy $\neg(\mathit{Terminal}(\mathbf{x}) \wedge \mathit{Flock}_S(\mathbf{x}))$. In both executions, assume that any time the movement velocity of any agent is nondeterministically chosen, it is chosen to be 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. From the convergence rate linear in N provided by Lemma 3.7, there exists a state $\mathbf{x}'$ such that $\mathit{NBM}(\mathbf{x}')$ is satisfied in each of the fail-free execution fragments α_f and α_n that is at most c rounds away from $\mathbf{x}$ in each of α_f and α_n where c is $O(N)$. As was established by Lemma 3.8, only once $\mathit{NBM}(\mathbf{x}')$ is satisfied can it be guaranteed that $\mathit{Flock}_S(\mathbf{x}')$ is satisfied. Once $\mathit{Flock}_S(\mathbf{x}')$ is satisfied, a state $\mathbf{x}''$ is reached which is some d rounds from $\mathbf{x}'$ in each of α_f and α_n, will the head agent compute a target such that $u_{H(\mathbf{x}'')} \neq 0$, and d is $O(N)$ by the $O(N)$ termination of the snapshot algorithm Figure 5, Line 29. Thus, α_f and α_n are indistinguishable up to state $\mathbf{x}''$, which is $c+d$ rounds from $\mathbf{x}$, each of which are $O(N)$, so their sum is $O(N)$. □

Next we show that the the failure detection mechanism incorporated in Figure 5 does not produce any false positives.

Lemma 3.14. *In any reachable state* $\mathbf{x}$, $\forall j \in \mathbf{x}.\mathit{Susp}_i \Rightarrow \mathbf{x}.\mathit{failed}_j$.

Proof. Suppose $\exists i, j \in \mathit{ID}$ such that $j \in \mathbf{x}.\mathit{Susp}_i$, then the precondition for $\mathsf{suspect}_\mathsf{i}(j)$ must have been satisfied at some round k_s in the past when j was added to Susp_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 on whether the target u_j is quantized or not for some $j \notin \mathbf{x}_{k_s-1}.\mathit{Susp}_i$ (Figure 5, Line 33).

We recall that the notation $\mathbf{x}_{s,T}$ represents the state $\mathbf{x}_s$ after the *Target* subroutine has executed. The first case occurs if the quantization constraint $|\mathbf{x}_{s,T}.u_j - \mathbf{x}_s.x_j| \leq \beta$ is not satisfied (Figure 5, 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 target $\mathbf{x}_s.u_j$ but in actuality applied a velocity that caused it to move away from $\mathbf{x}_s.u_j$ instead of toward 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 $\mathbf{x}_s.\mathit{failed}_j = \mathit{true}$.

The second case is when the quantization constraint $|\mathbf{x}_{s,T}.u_j - \mathbf{x}_s.x_j| \leq \beta$ was satisfied (Figure 5, Line 33), so $|\mathbf{x}_s.u_j - \mathbf{x}_s.x_j| = 0$ 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 $\mathbf{x}_s.\mathit{failed}_j = \mathit{true}$ since the only way $|\mathbf{x}'_s.x_j - \mathbf{x}_s.x_j| \neq 0$ is if $\mathbf{x}_s.vf_j \neq 0$ so that $\mathbf{x}'_s.x_j = \mathbf{x}_s.x_j + \mathbf{x}_s.vf_j$. □

The next lemma shows a partial *completeness* property [23] of DFDs. It upper-bounds the number of rounds to detect any detectable fault—which recall is any actuator fault which can be detected in a finite number of rounds using a DFD. This shows also that eventually all agents connected to some faulty agents know at least the set of faulty agents with which they are connected. The agents may know more faulty agents than those they are connected to, as some faulty agents might have left the communication group due to diverging.

Lemma 3.15. *For any fail-free reachable state* $\mathbf{x} \in \mathit{FFReach}(\mathbf{X}_f)$ *such that* $\neg\mathit{Terminal}(\mathbf{x})$ *is satisfied, there exists a state* $\mathbf{x}'$ *reachable from* $\mathbf{x}$ *such that, for any agent* i, $\mathbf{x}'.\mathit{Susp}_i \supseteq (F \cap G(\mathbf{x}', i))$, *and the number of rounds between* $\mathbf{x}$ *and* $\mathbf{x}'$ *is* $O(N)$.

Proof. Since $\neg\mathit{Terminal}(\mathbf{x})$, either $\mathit{Goal}(\mathbf{x})$ or $\mathit{NBM}(\mathbf{x})$ are not satisfied. If $\neg\mathit{NBM}(\mathbf{x})$, then by Lemmata 3.7 and 3.8, we know that there exists a state $\mathbf{x}'$ at most $O(N)$ rounds past $\mathbf{x}$ such that $\mathit{NBM}(\mathbf{x}')$ and $\mathit{Flock}_S(\mathbf{x}')$. Similarly, if $\neg\mathit{Goal}(\mathbf{x})$ and $\mathit{NBM}(\mathbf{x})$ are satisfied, then by Lemma 3.8, we know that $\mathit{Flock}_S(\mathbf{x})$, and due to the $O(N)$ termination of the distributed global snapshot algorithm, there exists a state $\mathbf{x}'$ which is $O(N)$ rounds from $\mathbf{x}$ at which a head agent could make a movement toward a state satisfying *Goal*. Since each of these states $\mathbf{x}'$ are $O(N)$ rounds from $\mathbf{x}$, we consider them simultaneously.

For both of these cases, we show that either at, before, or a constant number of rounds after such a state $\mathbf{x}'$, the precondition of a $\mathsf{suspect}_\mathsf{i}(j)$ action (Lines 6–8 of Figure 5) is satisfied for any faulty agent j and all its neighbors $i \in \mathit{Nbrs}(\mathbf{x}', j)$. By the above, at state $\mathbf{x}'$, we have that $\mathit{NBM}(\mathbf{x}')$ is satisfied. So, at or prior to round $\mathbf{x}'$, for any faulty agent j and any agent $i \in \mathit{Nbrs}(\mathbf{x}', j)$, we have that the precondition of the $\mathsf{suspect}_\mathsf{i}(j)$ action is satisfied, since for any of the faulty agents, either (a) a faulty middle or tail j moves and violates *NBM* at the round following $\mathbf{x}'$, (b) a faulty middle or tail j did not move between states $\mathbf{x}$ and $\mathbf{x}'$ and prevented *NBM* from being satisfied at $\mathbf{x}'$, (c) a faulty head j did not move at the round after state $\mathbf{x}'$ upon learning $\mathit{Flock}_S(\mathbf{x}')$ held, or (d) a faulty head j moved prior to Flock_S being satisfied at state $\mathbf{x}'$. Thus, any agent $i \in \mathit{Nbrs}(\mathbf{x}', j)$ has added by round $\mathbf{x}'$ (or the round following $\mathbf{x}'$) any faulty agent j to the set $\mathbf{x}'.\mathit{Susp}_i$.

By the assumption that the communication group of non-faulty agents cannot partition, we have that any non-faulty agents i and j have shared their suspected sets

within $O(N)$ (Figure 5, Line 21), since in the worst case, any two agents in the communication graph of agents are separated by at most $N-2$ agents, requiring $O(N)$ communication hops. Summing these $O(N)$ bounds yields the $O(N)$ upper bound on the detection time of DFDs. □

3.5 Generalization to Disconnected Sets of Agents

In this section, we show that the proposed flocking algorithm works (that is, Theorem 3.1 holds) even if faults cause the communication graph of agents to become partitioned, or if the communication graph is partitioned at an initial state. In Sections 3.2, 3.3, and 3.4, we proved safety and progress with the assumption that the graph of non-faulty agents *always* remains connected, which is quite restrictive. With no connectivity assumption, we will now establish Theorem 3.1. The theorem states that eventually $Flock_S$ and *Terminal* are established for a single communication group, and we will show that eventually there is a single group of agents satisfying these properties. Outside of states satisfying *Terminal*, it will be the case that several groups each satisfy $Flock_S$ and *NBM*, as defined on their disjoint groups, but eventually a single group emerges satisfying $Flock_S$, *NBM*, and *Goal*. This formulation eliminates redefining Theorem 3.1 to state that eventually every non-faulty communication group satisfies *Terminal* and $Flock_S$, as we do not need to state this if there is eventually a single non-faulty communication group satisfying these properties.

Invariance of safety without faults and reduced safety in the presence of faults are each straightforward due the upper bound v_{max} on velocity. The next invariant states that reduced safety is invariant for any reachable state and extends Invariant 3.2 since we now have no connectivity assumption. Intuitively, we just need to show that agents communicate before they collide, which is already ensured by the assumptions of Theorem 3.1, since by combining the existing Assumptions III and IV, we have that the communication radius $r_c > 2v_{max}$.

Invariant 3.4. *For any reachable state* $\mathbf{x}$, $Safety_R(\mathbf{x})$.

Proof. We sketch the proof, as the proof of this invariant is the same as the inductive invariance proof of Invariant 3.2, except we need an additional case for when faults have caused groups to partition and subsequently merge. A case for when groups partition is unnecessary, as the safe separation obviously follows since $r_r < r_c$—if agents cannot communicate because they are farther than r_c apart, then clearly they are at least r_r apart.

If any group of non-faulty agents merges with any other group (composed of faulty and/or non-faulty agents), we need to establish that the positions of all non-faulty agents with all other agents are at least separated by r_r. We analyze the cases for when a non-faulty group and a non-faulty group merge, and subsequently for when a non-faulty group and an arbitrary group merge. If two non-faulty groups $g_1, g_2 \in G_{\bar{F}}(\mathbf{x})$ merge, then since $Safety_R(\mathbf{x})$ is satisfied, either (a) a head of one group and tail of another are neighbors after the merge, or (b) the heads of both groups are neighbors after the merge, which can occur if each group has a single member. In both of these cases, the separation between any such non-faulty agents follows by first applying Lemma 3.3, which states that any two agents move toward one another with at most speed $2v_{max}$. Now from existing assumption that $r_r > r_s > r_c > 2v_{max}$, we have that any two non-faulty agents communicate prior to violating r_r since $r_c > r_r$, and from the *Target* computation, they remain spaced by at least r_r based on the same case reasoning used for each agent type (head, tail, middle) in the proof of Invariant 3.2.

If a non-faulty group merges with an arbitrary group, either the head and tail agents are non-faulty and the result follows by the same argument above, or one of the head or tail of the arbitrary group is faulty. In this case, by the detection time assumption of Theorem 3.1 that any faulty agent is suspected within k_d rounds by any of its neighbors—which is regardless of whether they began in a group or merged into a group at a later state—and $v_{max} \leq (r_s - r_r)/(2k_d)$ from the hypothesis of Theorem 3.1, we have that any agent in a non-faulty group with nearby faulty agents has detected them, and subsequently switches lanes prior to violating the r_r spacing, which can be done due to the non-blocking faults assumption of Theorem 3.1. □

The following lemma states that communication groups composed of non-faulty agents cannot partition (they may however merge).

Lemma 3.16. *For any two fail-free reachable states* $\mathbf{x}, \mathbf{x}' \in FFReach(\mathbf{X}_f)$ *such that* $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ *for some* $a \in A$, *no non-faulty communication group partitions between* $\mathbf{x}$ *and* $\mathbf{x}'$.

Proof. The only action $a \in A$ to be analyzed is update, as no other transition modifies the variable $Nbrs_i$ for any agent i. There are two cases based on whether the update transition moves agent positions so that any groups merge. If no non-faulty communication group merges, then the result follows by application of the following extension to Lemma 3.6 for each non-faulty communication group. For any communication group $g \in G_{\bar{F}}(\mathbf{x})$, if $\neg NBM(\mathbf{x}, g)$ is satisfied, then

$$\max_{i \in g} e(\mathbf{x}', i) \leq \max_{i \in g} e(\mathbf{x}, i).$$

The previous claim follows by the same convex combination arguments used in Lemma 3.6. The claim of the lemma now follows from the definition of connected.

If it is the case that $NBM(\mathbf{x}, g)$ is satisfied, then the claim of the lemma follows since no non-faulty agent $i \in g$ makes a move, however head agents may move. In the case that some head moves, one technical point must be

noted due to the distributed global snapshot algorithm used to detect $Flocks_S$. This algorithm can detect stable predicates, but upon groups merging or splitting, for some group $g \in G_{\bar{F}}(\mathbf{x})$, $Flocks_S(\mathbf{x}, g)$ may not be stable like $Flocks_S(\mathbf{x}, g)$ previously was when head agents are not moving and there is a single group (Corollary 3.1). Thus, a head agent may move even if $\neg Flocks_S(\mathbf{x}, g)$, however, it would only move at most one time, since after moving $\mathbf{x}.gsf_H$ for $H \in Heads(\mathbf{x})$ is set to false after a move, and thus this will not cause a partitioning of $G_{\bar{F}}(\mathbf{x})$.

In the other case that group merges do occur, we need to show that simultaneously no non-faulty groups partitioned. So, for any communication groups $g_1, g_2 \in G_{\bar{F}}(\mathbf{x})$ such that g_1 and g_2 have merged into a group g at state $\mathbf{x}'$, if $\neg NBM(\mathbf{x}, g_1)$ and $\neg NBM(\mathbf{x}, g_1)$ are satisfied, then

$$\max_{i \in g_1} e(\mathbf{x}', i) \leq \max_{i \in g_1} e(\mathbf{x}, i) \text{ and } \max_{j \in g_2} e(\mathbf{x}', j) \leq \max_{j \in g_2} e(\mathbf{x}, j).$$

The previous claim again follows by the same convex combination arguments used in Lemma 3.6, and by observing that we have restricted our consideration to the agents in the individual groups g_1 and g_2 both before and after the merge. The claim of the lemma now follows from the definition of connected.

If it is the case that either $NBM(\mathbf{x}, g_1)$ or $NBM(\mathbf{x}, g_2)$ are satisfied, then the claim of the lemma follows again since no non-faulty agents i, except possibly a head agent, in whichever of g_1 or g_2 satisfied NBM will make a move. If a head agent is involved, let us without loss of generality say this is $h_1 \in g_1$, then we know from the reasoning used in the proof of reduced safety for groups Invariant 3.4 that either a head h_2 or a tail t_2 from g_2 was involved in the merge. Now either h_1 is a neighbor of h_2 or h_1 is a neighbor of t_2, and the result follows by the definition of connected. □

Progress is more difficult to establish than safety. We begin by establishing properties of DFDs, including accuracy, partial completeness, and finite-detection time of detectable faults for groups as we did for the single communication group in Section 3.4. Upon establishing these properties, after failure detection we may restrict our focus to communication groups of non-faulty agents. The accuracy property, Lemma 3.14, holds without modification. The detection time lower-bound from Lemma 3.13 is extended to say that for a state $\mathbf{x}$ and an individual group $g \in G(\mathbf{x})$, there exist actuator faults which cannot be detected in fewer than $O(|g|)$ rounds. Failure detection for groups culminates in an extension to Lemma 3.15, that every detectable faulty agent which remains in a group with other agents is eventually suspected by those other agents. This allows us to discuss only non-faulty communication groups, as the next lemma establishes that within $O(N)$ time, all faulty agents in any communication group with other agents are suspected. The next lemma follows by the same arguments used in the proof of Lemma 3.15, completeness for a single group, except where the reasoning was used that the single non-faulty communication group does not partition, we instead reason by Lemma 3.16 that no non-faulty communication group partitions.

Lemma 3.17. *For any fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, if $\neg Terminal(\mathbf{x})$, then within $O(N)$ rounds a state $\mathbf{x}'$ is reached such that for all agents i connected to any faulty agents, $\mathbf{x}'.Susp_i \supseteq (F \cap G(\mathbf{x}', i))$.*

Now, rather than defining a Lyapunov function for a single communication group, a set of functions are defined, one for each non-faulty communication group in $G_{\bar{F}}(\mathbf{x})$, with a "composite" Lyapunov candidate defined as a set containing these candidates. Then, establishing that along any fail-free execution fragment α_{ff} either (a) one of the Lyapunov candidates decreases by a constant amount, or (b) the number of groups decreases because groups merge, guarantees progress. For a fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, for each non-faulty communication group $g \in G_{\bar{F}}(\mathbf{x})$, define a Lyapunov candidate as

$$V_g(\mathbf{x}) \triangleq \sum_{i \in g} e(\mathbf{x}, i).$$

We define the set of Lyapunov functions at a state $\mathbf{x}$ as

$$\Lambda(\mathbf{x}) = \{V_g(\mathbf{x}) : g \in G_{\bar{F}}(\mathbf{x})\}.$$

From Lemma 3.16, we do not have to consider partitions of non-faulty groups, and hence $|\Lambda(\mathbf{x})|$ is bounded from above by the number of non-faulty groups at the fail-free reachable state $\mathbf{x}$. If two groups $g_1, g_2 \in G_{\bar{F}}(\mathbf{x})$ merge to form a single group g at $\mathbf{x}'$, the value of V_g defined at state $\mathbf{x}'$ potentially increases such that $V_g > V_{g_1} + V_{g_2}$, but it is bounded from above by $\delta\bar{F}$ from Lemma 3.9—where at its greatest this would correspond to the case that there are no faulty agents and a single group g containing all N agents has emerged at state $\mathbf{x}'$.

The following theorem generalizes Theorem 3.3 to establish that eventually each non-faulty group $g \in G_{\bar{F}}(\mathbf{x})$ either makes progress to states satisfying *Terminal*, or merges with another non-faulty group, which decreases the cardinality of Λ, and thus eventually there exists a single group. The proof of the following also combines the proof of eventually reaching NBM for each group, that is, it also generalizes Lemma 3.7 to groups. We reiterate that if group merges occur, the new Lyapunov candidate is still bounded from above by $\delta\bar{F}$.

Theorem 3.5. *For any fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, there exists a state $\mathbf{x}'$ reachable from $\mathbf{x}$, such that there is a single element in the set of communication groups $G_{\bar{F}}(\mathbf{x}')$, that is, there is a single group of non-faulty agents.*

Proof. For any fail-free reachable state $\mathbf{x} \in FFReach(\mathbf{X}_f)$, let

$$G_{nbm}(\mathbf{x}) \triangleq \{g \in G_{\bar{F}}(\mathbf{x}) : \neg NBM(\mathbf{x}, g)\}$$

be the groups of non-faulty agents $g \in G_{\bar{F}}(\mathbf{x})$ that do not satisfy $NBM(\mathbf{x}, g)$. We now break the proof into cases based on whether the heads of each $g \in G_{nbm}$ are fixed and whether groups merge.

For the groups $g \in G_{nbm}(\mathbf{x})$ with fixed head agent positions, and which do not merge, the update transition decreases each of these $V_g(\mathbf{x})$ by at least a positive constant ψ. By the same reasoning used in the proof that states satisfying NBM are eventually reached for a single-group in Lemma 3.7, we let $\psi = v_{min}$, since none of the groups with agents which move interact with other groups. Also, there exists $\sigma_g \geq 0$ for each group, such that if $V_g(\mathbf{x}') \leq \sigma_g$, then $NBM(\mathbf{x}', g)$, which is achieved as in Lemma 3.7 by choosing $\beta \leq \frac{\sigma_g}{2N}$.

Next, for the groups $g \in G_{nbm}(\mathbf{x})$ where head agent positions may change, but which do not cause a merge, we instantiate Lemma 3.12 as we did in the earlier proof of termination (Theorem 3.1) with (a) $c_1 = v_{min}$, (b) $c_2 = |g| \frac{\delta}{2}$, (c) $c_3 = \beta$, (d) $c_4 = \left|\bar{F}\right| \delta$, and (e) $c_5 \leq O(N)$. and we have that any such groups make progress towards states satisfying $Goal$ and NBM, and since $NBM \subseteq Flock_S$, also toward flocking. However, the bound on the number of rounds from Lemma 3.12 does not necessarily apply, since group merges will occur in the intermediary states, but we have not yet considered merges; we do this now.

Suppose some non-faulty groups merge. Such merges may occur either due to (a) head agents moving and coming to be connected to some other non-faulty agents, or (b) non-faulty groups move toward states satisfying NBM, and we consider both of these cases together. Define a lexicographically ordered tuple

$$\mathcal{T}(\mathbf{x}) \triangleq \langle |\Lambda(\mathbf{x})|, V_{g_1}(\mathbf{x}), \ldots, V_{g_n}(\mathbf{x}) \rangle,$$

where $g_1, \ldots, g_n$ correspond to n non-faulty groups, the order of which is arbitrary, since we know that while the corresponding $V_{g_1}, \ldots, V_{g_n}$ may increase, when they do, their values are bounded. Since $|\Lambda(\mathbf{x})|$ is bounded from above and non-faulty communication groups cannot partition by Lemma 3.16, if a group merge occurs at the next state $\mathbf{x}'$, we have that $|\Lambda(\mathbf{x}')| < |\Lambda(\mathbf{x})|$, and thus $\mathcal{T}(\mathbf{x}') < \mathcal{T}(\mathbf{x})$, because the first component of $\mathcal{T}(\mathbf{x}')$ decreased and it is lexicographically ordered. After a merge has occurred, reapply the above arguments for the cases when merges do not occur until the next time a merge occurs, then reapply this argument showing that the cardinality of Λ decreases. This process continues until there is a single communication group of non-faulty agents, and at such a state $\mathbf{x}''$, $\mathcal{T}(\mathbf{x}'') = \langle 1, V(\mathbf{x}'') \rangle$, where $V(\mathbf{x}'')$ is the Lyapunov-like function summing errors across all non-faulty agents, since there is a single group.

□

Upon there being a single communication group of non-faulty agents, all the progress results of Section 3.3 can then be applied, thus eventually $Terminal$ is established. This yields Theorem 3.1 since we established $Safety_R$ is invariant and eventually $Terminal$ and $Flock_S$ are satisfied, even when actuator faults have caused the communication group to partition, or if the communication graph begins partitioned.

4 Related Work

Several works have studied distributed function computation, of which flocking is a special case [24, 3, 4, 5, 6, 7, 25] (see also the recent books [8, 9, 10] for a comprehensive treatment). An early investigation of flocking in computer science appeared in [24], where the agents were positioned on a circle and became equally spaced by moving to the midpoints of their nearest neighbors. Another early work on flocking in engineering was [12], where the concern was to generate group behavior for computer animations of movies, and this work uses double-integrator dynamics combined and three control rules to achieve flocking. Another model for flocking, but with single-integrator dynamics of point particles, is the Vicsek model [11], where agents update orientations but otherwise move with constant velocity magnitude. These and the majority of the prior works on flocking are still limited in that they have not considered either quantization or faults, and to the best of our knowledge, no works have considered both, particularly while ensuring agents do not collide. A few works have considered quantization [13, 14], where in particular [14] uses similar Lyapunov analysis as we do to bound convergence time. There are several research articles that study the general problem of coordinating mobile agents with different kinds of failures. For example, [26, 20] present a replicated state machine-based approach for handling failures in distributed coordination.

However, very few papers have attempted to prove avoidance of collisions in flocking. One such paper is [6], upon which we based our model and algorithm. We found an error in the flocking algorithm which allowed agents to collide. The error was in the special dynamics of the rightmost agent (a tail agent in our terminology), which could cause it to collide with other agents. To fix the error, we modified the algorithm as presented earlier in this paper (Figure 5, Line 31), which had previously been $x_N[k+1] = x_{N-1} + r_f$.

Very recently some attention has been given to robustness to agent faults in distributed averaging and flocking [27, 28, 29, 30, 31, 32, 33, 34]. There is an extensive body of work on crash faults. The earliest work on faults in flocking we know of is [27], where agents may fail by stopping—which in our framework would correspond to a fault with zero failure velocity—but the method is implemented on physical hardware.

One fault model considered in [28], "failure mode 3", considers byzantine-like behavior where a single faulty agent sets its position $x_i[k+1]$ to an arbitrary point on the real line—which clearly generalizes the actuator faults

considered here, where a faulty agent updates its position as $x_i[k+1] = x_i[k] + vf_i$ for some constant failure velocity vf_i. However, this work does not consider any notion of safety and defines robustness to be whether and how well the non-faulty agents can compute some function.

A similar fault model to ours is considered in [29], where motion probes are used in failure detection scenarios, but 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 a detection time bound linear in the number of agents in the system. This work also does not consider safe separation between agents in spite of faults, and the method is illustrated for a single fault.

A series of works has explored flocking first where agents permanently crash [31], and then where agents may fail due to memory corruption or permanently crashing [32]. The most recent work in this series [34] is very similar to our own work, which also establishes absence of collisions in spite of faults, but the faults considered here are crashes, which in our framework would correspond to zero velocity faults, although unlike crashes, in our failure model, agents continue to communicate after failing. A fault model is considered in [33] where faulty agents are malicious and may alter messages, but collision is not a concern, only function computation.

An earlier version of some results in this paper appeared as a master's thesis [16], which was subsequently presented at a conference [35]. Both [16, 35] assumed that faults could not partition the underlying communication graph of non-faulty agents, but in practice this could occur, so these works have been extended by relaxing this assumption as presented in Section 3.5. In particular in this paper, rather than assuming that faults cannot partition the communication graph of non-faulty agents, we show that even if the communication graph of agents partitions, then agents never collide and eventually there is a single communication graph of non-faulty agents which satisfies the desired flocking properties. This paper also improves upon the presentation of the results of [16, 35]. Previously, switching topologies for communication graphs in flocking were studied in [36, 37, 38], of which the partitioning of the communications graph allowed in this paper is a case. Also in the context of communication faults, it is shown in [39, 40] how convergence of a general class of distributed algorithms over asynchronous networks can be derived from convergence of their synchronous counterparts.

Finally, we note that the Lyapunov function used in Section 3.3 is similar to the one used in [41], as it is not quadratic and is the sum of absolute values of the errors of agent positions from flocking. It is also similar to the one used in [39], which is the maximum of the sum of absolute values of the errors.

5 Conclusion

This paper presented an algorithm for the safe flocking problem—where the desired properties are safety invariance, as well as eventual strong flock formation and destination reaching—in spite of permanent actuator faults. A lower-bound linear in the number of agents in the system was presented for the detection time of actuator faults for the class of directional failure detectors (DFDs). Without a means to detect faults in some way, such as through a DFD, the system would not be able to maintain safety as agents could collide, nor make progress to states satisfying flocking or the destination, since faulty agents may diverge, causing their neighbors to follow and diverge as well.

With the one-dimensional framework in this paper, it would be interesting to study the problem in more complicated lane scenarios, such as in finite-length lanes or the lanes of a traffic roundabout. In the future, we would like to establish collision avoidance in spite of actuator and other fault classes for flocking models with double-integrator dynamics in two and three-dimensions. Another interesting direction is whether it is possible to automatically establish safety properties of such systems, while in general, progress seems difficult due to reliance on existence of a Lyapunov-like function.

Acknowledgements

The authors wish to thank anonymous reviewers for careful reading and useful feedback on earlier revisions of this paper.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] W. Ren and R. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*, 1st ed., ser.

Communications and Control Engineering. London: Springer-Verlag, 2008.

[9] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, to appear. Electronically available at http://coordinationbook.info.

[10] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton, NJ: Princeton University Press, 2010. [Online]. Available: http://press.princeton.edu/titles/9230.html

[11] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, no. 6, pp. 1226–1229, Aug. 1995.

[12] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1987, pp. 25–34.

[13] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, May 2007.

[14] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2506–2517, Nov. 2009.

[15] 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.

[16] T. 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.

[17] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.

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

[19] J. McLurkin, "Analysis and implementation of distributed algorithms for Multi-Robot systems," Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[20] S. Gilbert, N. Lynch, S. Mitra, and T. Nolte, "Self-stabilizing robot formations over unreliable networks," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, pp. 1–17, July 2009.

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

[22] J. Hespanha and A. Morse, "Stability of switched systems with average dwell-time," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, Dec. 1999, pp. 2655–2660.

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

[24] E. Dijkstra, "Ewd386 the solution to a cyclic relaxation problem," in *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, 1982, pp. 34–35, eWD386's original date is 1973.

[25] O. Ben-Shahar, S. Dolev, A. Dolgin, and M. Segal, "Direction election in flocking swarms," in *Proceedings of the 6th International Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '10. New York, NY, USA: ACM, 2010, pp. 73–80.

[26] N. Lynch, S. Mitra, and T. Nolte, "Motion coordination using virtual nodes," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Dec. 2005, pp. 2823–2828.

[27] A. Hayes and P. Dormiani-Tabatabaei, "Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 4, May 2002, pp. 3900–3905.

[28] V. Gupta, C. Langbort, and R. Murray, "On the robustness of distributed algorithms," in *Decision and Control. 45th IEEE Conference on*, Dec. 2006, pp. 3473–3478.

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

[30] W. Ren and N. Sorensen, "Distributed coordination architecture for multi-robot formation control," *Robotics and Autonomous Systems*, vol. 56, no. 4, pp. 324–333, 2008.

[31] Y. Yang, N. Xiong, N. Y. Chong, and X. Defago, "A decentralized and adaptive flocking algorithm for autonomous mobile robots," in *Grid and Pervasive Computing Workshops, 2008. GPC Workshops '08. The 3rd International Conference on*, May 2008, pp. 262–268.

[32] Y. Yang, X. Defago, and M. Takizawa, "Self-stabilized flocking of a group of mobile robots under memory corruption," in *Proceedings of the 2009 International Conference on Network-Based Information Systems*, ser. NBIS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 532–538.

[33] S. Sundaram and C. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *Automatic Control, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2010.

[34] Y. Yang, S. Souissi, X. Défago, and M. Takizawa, "Fault-tolerant flocking for a group of autonomous mobile robots," *Journal of Systems and Software*, vol. 84, no. 1, pp. 29–36, 2011, information Networking and Software Services.

[35] T. T. Johnson and S. Mitra, "Safe flocking in spite of actuator faults," in *12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2010)*, ser. Lecture Notes in Computer Science, S. Dolev, J. Cobb, M. Fischer, and M. Yung, Eds. Springer Berlin / Heidelberg, Sep. 2010, vol. 6366, pp. 588–602.

[36] R. Olfati-Saber and R. Murray, "Flocking with obstacle avoidance: cooperation with limited communication in mobile networks," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 2, Dec. 2003, pp. 2022–2028.

[37] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[38] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 863–868, May 2007.

[39] K. Chandy, S. Mitra, and C. Pilotto, "Convergence verification: From shared memory to partially synchronous systems," in *Formal Modeling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science, F. Cassez and C. Jard, Eds. Springer Berlin / Heidelberg, 2008, vol. 5215, pp. 218–232.

[40] K. Chandy, B. Go, S. Mitra, C. Pilotto, and J. White, "Verification of distributed systems with localglobal predicates," *Formal Aspects of Computing*, pp. 1–31, 2010.

[41] 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.

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

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

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 [42]. For the following, assume that Figure 5, Line 37 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 [6], 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,

$$x[k] = \begin{pmatrix} \mathbf{x}_k.x_{H(\mathbf{x}_k)} \\ \mathbf{x}_k.x_{x.R_{H(\mathbf{x}_k)}} \\ \vdots \\ \mathbf{x}_k.x_{T(\mathbf{x}_k)} \end{pmatrix}, b_p = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix}, \text{ and}$$

$$A_p = \begin{pmatrix} a_{1,1} & 0 & 0 & 0 & 0 & \ldots \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & \ldots \\ 0 & \ddots & \ddots & \ddots & 0 & \ldots \\ 0 & \ldots & a_{i,i-1} & a_{i,i} & a_{i,i+1} & \ldots \\ 0 & 0 & 0 & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & a_{N,N-1} & a_{N,N} \end{pmatrix}.$$

The following are the family of matrices A_p and vectors b_p that are switched among based on the state of System, and we describe them assuming there is a single communication group; refer to Figure 5 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, and cause us to update which neighbors agent i uses to compute its control, so a new family of matrices A_q needs to be defined, where all coefficients not specified are 0. The actuator stuck-at failures being modeled are representative of an additive error term in the b_p vector [43]. From Line 36, 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 $b_N = \mathbf{x}_k.vf_{T(\mathbf{x}_k)}$. Finally, we have omitted that non-faulty agents would also change which neighbors they would eventually use for target computation.

B Simulations

Simulation studies were performed, where flocking convergence time (as by Lemma 3.7), goal convergence time (as by Theorem 3.3), and failure detection time (as by Lemma 3.15) 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.

We first mention Figures 7 and 8, which show an execution of the system where the non-faulty agents are progressing (reaching the origin as a flock) while detecting and avoiding a faulty agent by switching lanes. After suspecting that agent 6 is faulty, agents 1 through 5 move to lane 2 at approximately round 375 to avoid collision with the faulty agent.

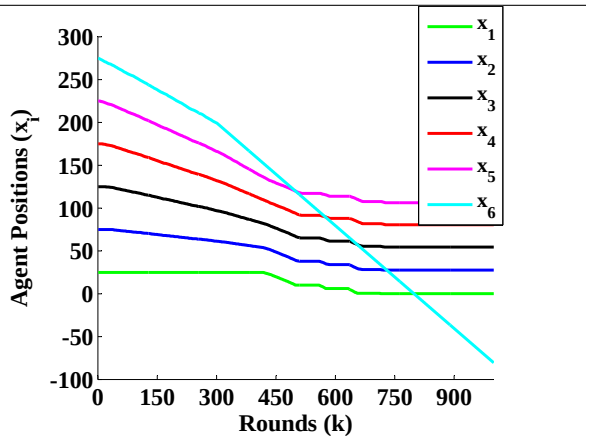

Figure 7: System progressing: eventually the agents have formed a flock and the faulty agent 6 with nonzero velocity has diverged.

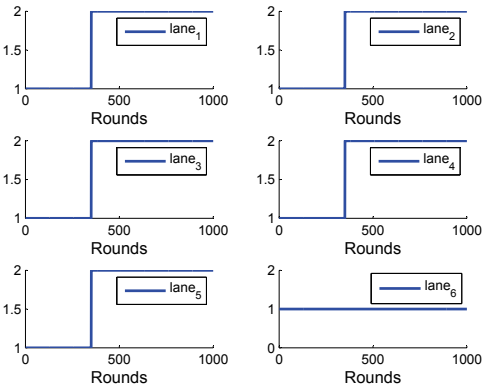

Figure 8: System progressing: illustrating that non-faulty agents have avoided faulty agent 6 by changing lanes.

Figure 9 shows System without failures "expanding" [6] to form a strong flock prior to moving towards the goal, and Figure 10 shows the value of the Lyapunov-like function, V, and maximum agent error from flocking, $e_{max} = \max_{i \in F} e(\mathbf{x}, i)$, 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 Lemma 3.15. The

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

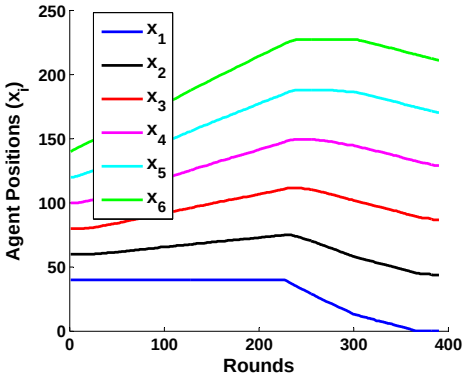

Figure 9: Agent expansion simulation showing agent positions, x_i. Observe that first the agents form a flock by expanding and then move toward the goal.

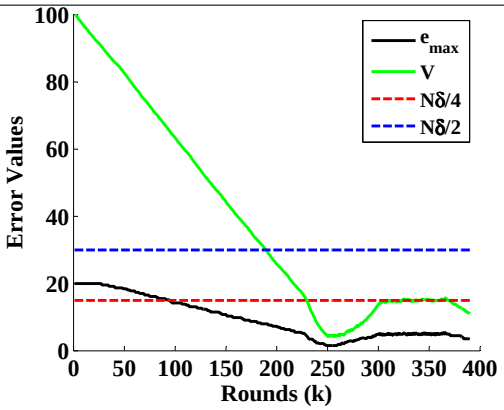

Figure 10: Expansion simulation showing max error e_{max}, Lyapunov function value V, with weak and strong flocking constants.

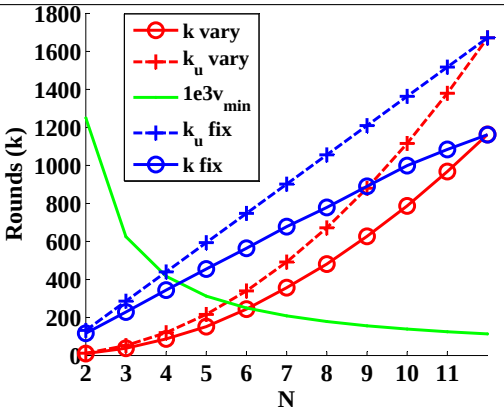

Figure 11: Rounds k (upper bounded by k_u) to reach *Terminal* versus number of agents N with fixed and varying values of v_{min} (i.e., as a function of N or not).

Figure 12 shows the values of the maximum agent error e_{max} along with the Lyapunov function V, and displays that $Flock_S$ is not a stable predicate, while $Flock_W$ is a stable predicate, since V is bounded by $\frac{N\delta}{2}$. 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.

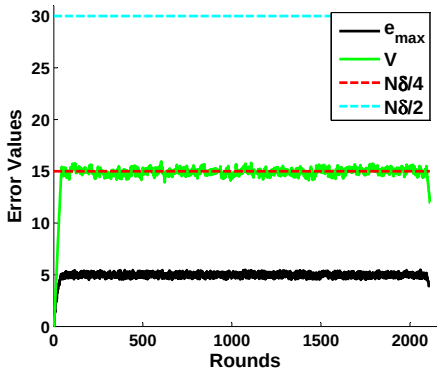

Figure 12: Invariance of $Flock_W$ and repeated entry to *NBM* and $Flock_S$.

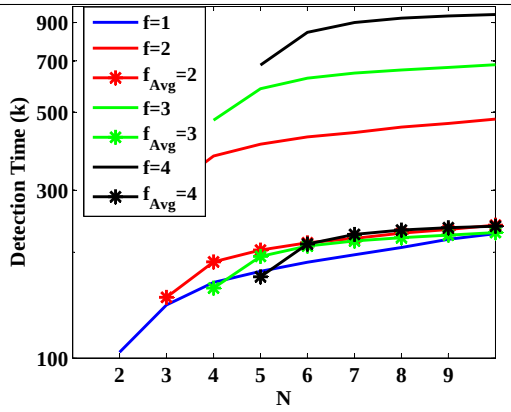

Figure 13: Simulating f zero velocity failures at round 0 from initial state of $2r_f$ inter-agent spacing.

Figure 13 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 Lemma 3.15.

Figure 14 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 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.

	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

Figure 14: 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.