Safe and Stabilizing Distributed Cellular Flows

Taylor Johnson, Sayan Mitra, and Karthik Manamcheri
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Email: {johnso99, mitras, manamch1}@illinois.edu

***Abstract*—Advances in wireless vehicular networks present us with opportunities for developing new distributed traffic control algorithms that avoid phenomena such as abrupt phase-transitions. Towards this end, we study the problem of distributed traffic control in a partitioned plane where the movement of all entities (vehicles) within each partition or *cell* is tightly coupled. We present a distributed traffic control protocol that guarantees minimum separation between vehicles at all times, even when some cells' control software may fail. Once failures cease, the protocol is guaranteed to stabilize and the vehicles with feasible paths to a target cell make progress towards it. The algorithm relies on two general principles: temporary blocking for maintenance of safety and local geographical routing for guaranteeing progress. Our proofs use mostly assertional reasoning and may serve as a template for analyzing other safe and stabilizing distributed traffic control protocols. We also present simulation results which provide estimates of throughput as a function of vehicle velocity, safety separation, path complexity, and failure and recovery rates.**

I. Introduction

Highway and air traffic flows are nonlinear dynamical systems that give rise to complex phenomena such as abrupt phase-transitions from fast to sluggish flow [1], [2], [3]. Our ability to monitor, predict, and avoid such phenomena can have significant impact on the reliability and the capacity of traffic networks. Traditional traffic protocols, such as those implemented for air-traffic control are *centralized* [4]—a *coordinator* periodically collects information from the vehicles, decides, and disseminates the waypoints, and subsequently the vehicles try to blindly follow a path to the waypoint. The advent of wireless vehicular networks [5] presents a new opportunity for *distributed* traffic monitoring [6] and control. Such protocols should scale and be less vulnerable to failures, compared to their centralized counterparts. In this paper, we propose such a distributed traffic control protocol and analyze its behavior.

A *traffic control protocol* is a set of rules that determines the routing and movement of certain physical *entities*, such as cars and packages, over an underlying *graph*, such as a road network, air-traffic network, or a warehouse conveyor system. Any traffic control protocol should guarantee: (a) (*safety*) that the entities maintain some minimum physical separation, and (b) (*progress*) that the entities arrive at a given a destination (or target) vertex. In a distributed traffic control protocol each entity determines its own next-waypoint, or each vertex in the underlying graph determines the next-waypoints for the entities in an appropriately defined neighborhood. The idea of distributed traffic control has been around for some time but most of the work has focused on human-factors issues [7], [8], collision avoidance [9], [10], [11], [12], [13], and platooning [14], [15], [16]. A notable exception is the paper [17] which presents a distributed algorithm (executed by entities, vehicles in this case) for controlling a highway intersection without any stop signs.

In this paper, we study the problem of distributed traffic control in a partitioned plane where the motions of entities within a partition are *coupled*. The problem can be described as follows (refer to Figure 1). The geographical space of interest is partitioned into regions or *cells*. There is a designated target cell which consumes entities and some source cells that produce new entities. The entities within a cell are coupled, in the sense that they all either move identically or they remain static (we discuss the motivation for this below). If a cell moves such that some entities within it touch the boundary of a neighboring cell, those get transferred to the neighboring cell. Thus, the role of the distributed traffic control protocol is to control the motion of the cells so that the entities (a) always have the required separation, and (b) they reach the target, if feasible.

The above mentioned coupling which requires entities within a cell to move identically may appear surprising at first sight. After all, under low traffic conditions individual, drivers control the movement of their cars within a particular region of the highway, somewhat independently of the other drivers in that region. However, in highway traffic under high-traffic, high-velocity conditions, it is known that coupling may emerge spontaneously, whereby the vehicles form a fixed lattice structure and move with zero relative speed [2], [18]. In other scenarios coupling arises because passive entities are moved around by active cells, for example, packages being routed on a grid of multi-

directional conveyors [19], and molecules moving on a medium according to some controlled chemical gradient. Finally, even where the entities are active and cells are not, the entities can cooperate to emulate a virtual active cell expressly for the purposes of distributed coordination. This idea has been explored for mobile robot coordination in [20] using a cooperation strategy called virtual stationary automata [21], [22].

In this paper we present such a distributed traffic control protocol that guarantees safety at all times, even when some cells fail permanently by crashing. The protocol also guarantees *progress* of entities towards the target, provided a path through non-faulty cells exists to the target. Specifically, the protocol being *self-stabilizing* [23], in that, after failures stop occurring, the composed system automatically returns to a state from which progress can be made. The algorithm relies on two mechanisms: (a) a rule to maintain local routing tables at each non-faulty cell, and (b) a (more interesting) rule for signaling amongst neighbors which guarantees safety while preventing deadlocks. Roughly speaking, the signaling mechanism at some cell fairly chooses amongst its neighboring cells which contain entities, indicating if it is safe for one of these cells to apply a movement in the direction of the signaling cell. This permission-to-move policy turns out to be necessary, because movement of neighboring cells may otherwise result in a violation of safety in the signaling cell.

We establish these safety and progress properties through systematic assertional reasoning. We believe that these proofs may serve as a template for the analysis of other distributed traffic control protocols and also can be mechanized using automated theorem proving tools, for example [24].

The throughput analysis of this algorithm, and in fact any distributed traffic control algorithm, remains a challenge. We present simulation results that illustrate the influence (or the lack thereof) of several factors on throughput: (a) path length, (b) path complexity measured in number of turns along a path, (c) required safety separation and cell velocity, and (d) failure and recovery rates, under a model where crash failures are not permanent and cells may recover from crashing.

The rest of the paper is organized as follows. First, Section II introduces the system model and protocol. Next in Section III, we define the safety and progress properties to be analyzed and analyze them. Subsection III-A then shows that the safety property is invariant and is satisfied in any reachable state. Then progress is established by showing two properties. First in Subsection III-B, it is shown that the routing protocol to find the target from any cell with a path to the target is self-stabilizing. Secondly in Subsection III-C, it is shown that entities on any cell with a feasible physical path to the target eventually reach the target. Simulation results and interpretation are presented in Section IV, followed by a brief discussion and conclusion in Section V.

II. System Model

For a set K, let $K_\perp \triangleq K \cup \{\perp\}$ and $K_\infty \triangleq K \cup \{\infty\}$. For $N \in \mathbb{N}$, let $[N] \triangleq \{0, \ldots, N\}$. For a variable x, its type is denoted by $type(x)$ and it is the set of values that it can take. A *valuation* for a set of variables X, denoted by $\mathbf{x}$, is a function that maps each $x \in X$ to a point in $type(x)$. Given a valuation $\mathbf{x}$ for X, the valuation for a variable $v \in X$, denoted by $\mathbf{x}.v$, is the restriction of $\mathbf{x}$ to $\{v\}$. The set of all possible valuations of X is denoted by $val(X)$.

A *discrete transition system* $\mathcal{A}$ is a tuple $\langle X, Q_0, A, \rightarrow \rangle$, where (i) X is a set of variables, $val(X)$ is called the set of *states*, (ii) $Q_0 \subseteq val(X)$ is the set of *start states*, (iii) A is a set of transition *names*, and (iv) $\rightarrow \subseteq val(X) \times A \times val(X)$ is a set of *discrete transitions*. An *execution fragment* of A is a (possibly infinite) sequence of states $\alpha = \mathbf{x}_0, \mathbf{x}_1, \ldots$, such that for each index appearing in α, $(\mathbf{x}_k, a, \mathbf{x}_{k+1}) \in \rightarrow$ for some $a \in A$. An *execution* is an execution fragment with $\mathbf{x}_0 \in Q_0$. A state $\mathbf{x}$ is said to be *reachable* if there exists a finite execution that ends in $\mathbf{x}$. $\mathcal{A}$ is said to be safe with respect to a set $S \subseteq val(X)$ if all reachable states are contained in S. A set S is said to be *stable* if for each $(\mathbf{x}, a, \mathbf{x}') \in \rightarrow$, $\mathbf{x} \in S$ implies that $\mathbf{x}' \in S$. $\mathcal{A}$ is said to *stabilize* to S if S is stable and every execution fragment reaches S.

A. Overview of distributed cellular traffic control

The system consists of N^2 cells arranged in an $N \times N$ grid. Each cell physically occupies a unit square region in the plane and may contain a number of entities, each of which occupies a smaller square region. All the entities on a given cell move identically: either they remain static, or they move with some constant velocity either horizontally or vertically. This movement is determined by the software controlling each cell. The software relies on communication amongst adjacent cells. When a moving entity touches an edge of a cell, it is instantaneously transferred to the next neighboring cell.

The software of a cell implements the distributed traffic control protocol. In this paper, we consider synchronous protocols which operate in rounds. At each round, every cell exchanges messages bearing state information with their neighbors. Based on this, the cells update their software state and decide their (possibly zero) velocities. Until the beginning of the next round, the cells continue to operate according to this velocity—this may lead to entity transfers.

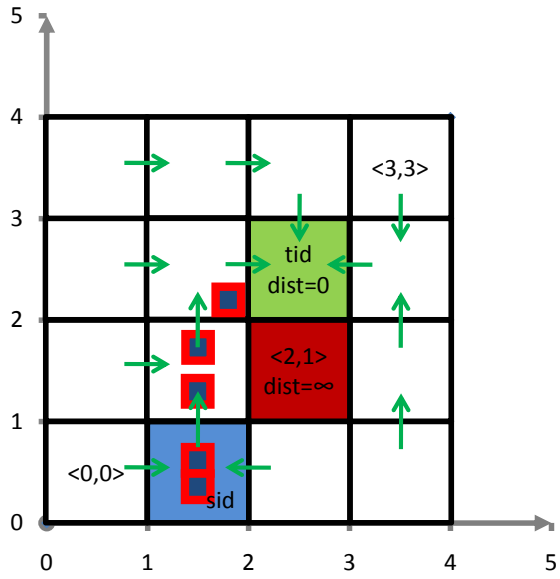

Figure 1. System with 4×4 cells where $tid = \langle 2,2 \rangle$ (in green), $SID = \{\langle 1,0 \rangle\}$ (in light blue), $failed_{2,1} = true$ (in red), the green arrows represent $next$, and the smaller red and blue squares are entities with safety region in red and length region in blue.

Assume that messages are delivered within bounded time, computations are instantaneous, all the entities have the same size, and if moving, any cell does so the same constant velocity. The latter two assumptions are for simplicity of presentation only. Under these assumptions, the system can be modeled as a collection of discrete transition systems that interact through shared variables. In what follows, we present the discrete transition system that is obtained by composing the models for the individual cells.

B. Formal system model

Let $ID \triangleq [N-1] \times [N-1]$ be the set of identifiers for all cells in the system. Each cell has a unique identifier $\langle i,j \rangle \in ID$. Cell $\langle i,j \rangle$ occupies a unit square whose bottom-left corner is the point (i,j) in the Euclidean plane. Cell $\langle m,n \rangle$ is said to be *neighbor* of cell $\langle i,j \rangle$ if $|i-m| + |j-n| = 1$. The set of identifiers of all neighbors of $\langle i,j \rangle$ is denoted by $Nbrs_{i,j}$. For this paper we will consider a system with a unique *target cell* with identifier tid and a set of *source cells*, with identifiers $SID \subset ID$. All other cells are *ordinary cells*. Every entity that may ever be in the system has a unique identifier drawn from a set P. For any entity $p \in P$ that is actually present in the system, we denote the coordinates of its center by $(p_x, p_y) \in \mathbb{R}^2$. Entity p occupies an $l \times l$ square area, with its center at p_x, p_y.

The specification of the system uses the following three parameters: (i) l: length of an entity, (ii) r_s: minimum required inter-entity gap along each axis, and (iii) v: cell velocity, or distance by which an entity may move over one round. It is required that $v < l < 1$ and $r_s + l < 1$. The former is required to ensure cells do not violate the gap requirement from one round to the next when new entities enter a cell. The latter is required so that entities cover at most the same area of the Euclidean plane as the cell in which they are contained, since cells are squares of unit length. Define the center spacing requirement $d \triangleq r_s + l$.

Next, we describe the discrete transition system (state machine) $\mathsf{Cell}_{i,j}$ that specifies the behavior of an individual cell. We specify the variables associated with each $\mathsf{Cell}_{i,j}$; initial values of the variables are shown in Figure 3 using the ':=' notation:

(i) $Members_{i,j}$: set of entities located in cell $\langle i,j \rangle$,
(ii) $next_{i,j}$: neighbor towards which $\langle i,j \rangle$ attempts to move,
(iii) $NEPrev_{i,j}$: nonempty neighbors for which $\langle i,j \rangle$ is the $next$,
(iv) $dist_{i,j}$: estimated Manhattan distance to tid,
(v) $token_{i,j}$: a token used for mutual exclusion to indicate which neighbor may move,
(vi) $signal_{i,j}$: indicates whether a physical region in $\mathsf{Cell}_{i,j}$ is empty, and
(vii) $failed_{i,j}$: indicates whether or not $\langle i,j \rangle$ has failed.

When clear from context, the subscripts in the names of the variables are dropped. A state of $\mathsf{Cell}_{i,j}$ refers to a valuation of all these variables, i.e., a function that maps each variable to a value of the corresponding type. The complete system is an automaton, called System, consisting of the ensemble of all the cells. A state of System is a valuation of all the variables for all the cells. We refer to states of System with bold letters $\mathbf{x}$, $\mathbf{x}'$, etc.

As we shall see shortly, variables $token_{i,j}$, $failed_{i,j}$, and $NEPrev_{i,j}$ are private to $\mathsf{Cell}_{i,j}$, while $Members_{i,j}, dist_{i,j}, next_{i,j}$, and $signal_{i,j}$ can be read by neighboring cells of $\mathsf{Cell}_{i,j}$. What this means for an actual message-passing implementation is the following. At the beginning of each round, $\mathsf{Cell}_{i,j}$ broadcasts messages containing the values of these variables and receives similar values from its neighbors. Then the computation of this round updates the local variables for each cell based on the values collected from its neighbors. Variable $Members_{i,j}$ is a special variable, in that it can also be written to by the neighbors of $\mathsf{Cell}_{i,j}$. This is how we capture transferal of entities amongst cells.

System has two types of state transitions: fails and updates. A fail($\langle i,j \rangle$) transition models the crash failure of the $(i,j)^{th}$ cell and it sets $failed_{i,j}$ to $true$, $dist_{i,j}$ is set to ∞, and $next_{i,j}$ is set to $\perp$. A cell $\langle i,j \rangle$ is called

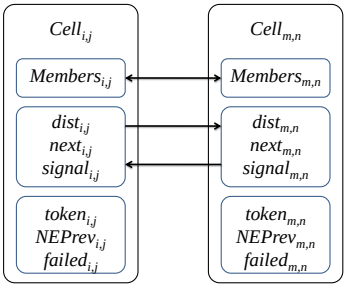

Figure 2. Interaction between a pair of neighboring cells is modeled with shared variables *Members*, *dist*, *next*, and *signal*.

failed if $failed_{i,j}$ is $true$, otherwise it is called *non-faulty*. The set of identifiers of all failed and non-faulty cells at a state $\mathbf{x}$ is denoted by $F(\mathbf{x})$ and $NF(\mathbf{x})$, respectively. A failed cell does nothing; it never moves and it never communicates[1].

An update transition models the evolution of all non-faulty cells over one synchronous round. For readability, we describe the state-change owing to an update transition as a sequence of three functions (subroutines), which for each non-faulty $\langle i,j\rangle$, (i) *Route* computes the variables $dist_{i,j}$ and $next_{i,j}$, (ii) *Signal* computes (primarily) the variable $signal_{i,j}$, and (iii) *Move* computes the new positions of entities within $Members_{i,j}$. Note that the entire update transition is atomic, so there is no possibility to interleave fail transitions between the subroutines of update. To reiterate, in this discrete automaton model, all the changes in the state of System are captured by a single atomic transition brought about by update. Thus, the state of System at (the beginning of) round $k+1$ is obtained by the applying these three functions to the state at round k. Now we proceed to describe the distributed traffic control algorithm which is implemented through these functions.

The *Route* function (Figure 4) is responsible for constructing stable routes in the face of failures. Specifically, it constructs a distance-based routing table for each cell that relies only on neighbors' estimates of distance to the target. Recall that failed cells have *dist* set to infinity. From a state $\mathbf{x}$, for each $\langle i,j\rangle \in NF(\mathbf{x})$, the variable $dist_{i,j}$ is updated as 1 plus the minimum value of *dist* amongst the neighbors of $\langle i,j\rangle$. If this results in $dist_{i,j}$ being infinity, then $next_{i,j}$ is set to $\perp$, otherwise it is set to be the identifier with the minimum *dist* with ties broken with neighbor identifiers.

The *Signal* function (Figure 5) executes after *Route* and is the key part of the protocol for both maintaining safe entity separations and ensuring progress of entities to the target. Roughly this is implemented by each cell by following two policies: (a) accept new entities from a neighbor only when this is safe, and (b) provide opportunities infinitely often for each nonempty neighbor to make progress. First $\langle i,j\rangle$ sets $NEPrev_{i,j}$ to be the subset of $Nbrs_{i,j}$ for which *next* has been set to $\langle i,j\rangle$ *and* $Members$ is nonempty. If $token_{i,j}$ is $\perp$ then it is set to some arbitrary value in $NEPrev_{i,j}$; it continues to be $\perp$ if $NEPrev_{i,j}$ is empty. Otherwise, $token_{i,j} = \langle m,n\rangle$, which is a neighbor of $\langle i,j\rangle$ with nonempty $Members$. It is checked if there is a gap of length d on $\mathsf{Cell}_{i,j}$ in the direction of $\langle m,n\rangle$. This is accomplished through the conditional in Lines 4—7 as a step in guarantying fairness. If there is not enough gap, then $signal_{i,j}$ is set to $\perp$, which blocks $\langle m,n\rangle$ from moving its entities in the direction of $\langle i,j\rangle$, thus preventing entity transfers. On the other hand, if there is sufficient gap, then $signal_{i,j}$ is set to $token_{i,j}$ which enables $\langle m,n\rangle$ to move its entities towards $\langle i,j\rangle$. Finally, $token_{i,j}$ is updated to a value in $NEPrev_{i,j}$ that is different from its previous value, if that is possible according to the rules just described (Lines 10-12).

Finally, the *Move* function (Figure 6) models the physical movement of entities over a given round. For cell $\langle i,j\rangle$, let $\langle m,n\rangle$ be $next_{i,j}$. The entities in $Members_{i,j}$ move in the direction of $\langle m,n\rangle$ if and only if $signal_{m,n}$ is set to $\langle i,j\rangle$. In that case, all the entities in $Members_{i,j}$ are shifted in the direction of cell $\langle m,n\rangle$. This may lead to some entities crossing the boundary of $\mathsf{Cell}_{i,j}$ into $\mathsf{Cell}_{m,n}$, in which case, such entities are removed from $Members_{i,j}$. If $\langle m,n\rangle$ is not the target, then the removed entity is added to $Members_{m,n}$. In this case (Lines 13-20), the transferred is placed at the edge of $\mathsf{Cell}_{m,n}$. However, if $\langle m,n\rangle$ is the target, then

[1] $dist_{i,j} = \infty$ can be interpreted as its neighbors not receiving a timely response from $\langle i,j\rangle$.

variables 1
  $Members_{i,j}$: $\mathtt{Set}[P] := \{\}$
  $NEPrev_{i,j}$: $\mathtt{Set}[ID_\perp] := \{\}$ 3
  $next_{i,j}$, $signal_{i,j}$, $token_{i,j}$: $ID_\perp := \perp$
  $dist_{i,j} : \mathbb{N}_\infty := \infty$ 5
  $failed_{i,j} : \mathbb{B} := false$
transitions 7
  fail($\langle i,j\rangle$)
    eff $failed_{i,j} := true$; $dist_{i,j} := \infty$; $next_{i,j} := \perp$ 9

  update 11
    eff *Route*; *Signal*; *Move*

Figure 3. Specification of $\mathsf{Cell}_{i,j}$.

if $\neg failed_{i,j} \wedge \langle i,j\rangle \neq tid$ **then**
  $dist_{i,j} := \left(\min_{\langle m,n\rangle \in Nbrs_{i,j}} dist_{m,n}\right) + 1$ 2
  if $dist_{i,j} = \infty$ **then** $next_{i,j} := \perp$
  else $next_{i,j} := \operatorname{argmin}_{\langle m,n\rangle \in Nbrs_{i,j}} (dist_{m,n}, \langle m,n\rangle)$ 4

Figure 4. *Route* function.

```
if ¬failed_{i,j} then
  NEPrev_{i,j} := {⟨m,n⟩ ∈ Nbrs_{i,j} : next_{m,n} = ⟨i,j⟩ ∧ Members_{m,n} ≠ ∅}
  if token_{i,j} = ⊥ then token_{i,j} := choose from NEPrev_{i,j}
  if ((token_{i,j} = i + 1 ∧ ∀ p ∈ Members_{i,j} : p_x + l/2 ≤ i + 1 - d)
    ∨ (token_{i,j} = i - 1 ∧ ∀ p ∈ Members_{i,j} : p_x - l/2 ≥ i + d)
    ∨ (token_{i,j} = j + 1 ∧ ∀ p ∈ Members_{i,j} : p_y + l/2 ≤ j + 1 - d)
    ∨ (token_{i,j} = i - 1 ∧ ∀ p ∈ Members_{i,j} : p_y - l/2 ≥ j + d))
  then
    signal_{i,j} := token_{i,j}
    if |NEPrev_{i,j}| > 1 then
      token_{i,j} := choose from NEPrev_{i,j} \ {token_{i,j}}
    elseif |NEPrev_{i,j}| = 1 then token_{i,j} ∈ NEPrev_{i,j}
    else token_{i,j} := ⊥
  else signal_{i,j} := ⊥; token_{i,j} := token_{i,j}
```

Figure 5. *Signal* function.

the removed entity is not added to any cell and thus no longer exists in System.

The source cells $\langle i,j\rangle \in SID$, in addition to the above movements, add at most one entity in each round to $Members_{i,j}$ such that the addition of such an entity does not violate the minimum gap requirement between entities at $\mathsf{Cell}_{i,j}$.

```
if ¬failed_{i,j} ∧ signal_{next_{i,j}} = ⟨i,j⟩ then
  let ⟨m,n⟩ = next_{i,j}
  for each p ∈ Members_{i,j}
    p_x := p_x + v(m - i)
    p_y := p_y + v(n - j)

    if (m = i + 1 ∧ p_x + l/2 > i + 1) ∨ (m = i -1 ∧ p_x - l/2 < i) ∨
      (n = j + 1 ∧ p_y + l/2 > j + 1) ∨ (n = j -1 ∧ p_y - l/2 < j)
    then
      Members_{i,j} := Members_{i,j} \ {p}
      if ⟨m,n⟩ ≠ tid
      then Members_{m,n} := Members_{m,n} ∪ {p}
        if m = i + 1 ∧ p_x + l/2 > i + 1
        then p_x := m + l/2
        elseif m = i -1 ∧ p_x - l/2 < i
        then p_x := m - l/2
        elseif n = j + 1 ∧ p_y + l/2 > j + 1
        then p_y := n + l/2
        elseif n = j -1 ∧ p_y - l/2 < j
        then p_y := n - l/2
```

Figure 6. *Move* function.

III. Analysis

In this section we present an analysis of System with regards to safety and progress properties. Roughly, the safety property requires that there is a minimum gap between entities and the progress property requires that all entities which reside on cells with a feasible path to the target, eventually reach the target.

A. Safety analysis

For any state $\mathbf{x}$ of System, define:

$$\begin{aligned} Safe_{i,j}(\mathbf{x}) &\triangleq \forall p,q \in \mathbf{x}.Members_{i,j}, p \neq q \\ &\quad (|\mathbf{x}.p_x - \mathbf{x}.q_x| \geq d) \vee (|\mathbf{x}.p_y - \mathbf{x}.q_y| \geq d). \\ Safe(\mathbf{x}) &\triangleq \forall \langle i,j\rangle \in ID, Safe_{i,j}(\mathbf{x}). \end{aligned}$$

A state is safe if for every cell, the distance between the centers of any two entities along either coordinate is at least d. Thus, in a safe state, the edges of all entities in a cell are separated by a distance of r_s. However, the entities in two adjacent cells may have edges spaced apart by less, although their centers will be spaced by at least l.

We proceed by proving some preliminary properties of System which will be used for establishing the desired safety property. The following invariant asserts that no entities exist between the boundaries of cells. This is a consequence of transferring entities upon an entity touching the edge of a cell, and then resetting the entity's position across to the new cell.

Invariant 1. In any reachable state $\mathbf{x}$, $\forall \langle i,j\rangle \in ID$, $\forall p \in \mathbf{x}.Members_{i,j}$

$$\begin{aligned} i + \frac{l}{2} &\leq p_x \leq i + 1 - \frac{l}{2}, \text{ and} \\ j + \frac{l}{2} &\leq p_y \leq j + 1 - \frac{l}{2}. \end{aligned}$$

The next invariant states that cells' Memberss are disjoint. This is immediate from the *Move* function as, entities are only added to one cell's Members upon being removed from a different cell's Members.

Invariant 2. In any reachable state $\mathbf{x}$, for any two distinct $\langle i,j\rangle, \langle m,n\rangle \in ID$, $\mathbf{x}.Members_{i,j} \cap \mathbf{x}.Members_{m,n} = \emptyset$.

Next, we define a predicate which states that if $signal_{i,j}$ is set to some $\langle m,n\rangle \in Nbrs_{i,j}$, then there is a large enough gap from the common edge where no entities exist in $\mathsf{Cell}_{i,j}$. $H(\mathbf{x}) \triangleq \forall \langle i,j\rangle \in ID, \langle m,n\rangle \in Nbrs_{i,j}$, if $\mathbf{x}.signal_{i,j} = \langle m,n\rangle$ then exactly one of the following hold:

$$\begin{aligned} m = i+1 \quad &\wedge \quad \forall p \in \mathbf{x}.Members_{i,j}, \mathbf{x}.p_x + \frac{l}{2} \leq i + 1 - d \\ m = i-1 \quad &\wedge \quad \forall p \in \mathbf{x}.Members_{i,j}, \mathbf{x}.p_x - \frac{l}{2} \geq i + d \\ n = j+1 \quad &\wedge \quad \forall p \in \mathbf{x}.Members_{i,j}, \mathbf{x}.p_y + \frac{l}{2} \leq j + 1 - d \\ n = i-1 \quad &\wedge \quad \forall p \in \mathbf{x}.Members_{i,j}, \mathbf{x}.p_y - \frac{l}{2} \geq j + d. \end{aligned}$$

$H(\mathbf{x})$ is not an invariant property because once entities move the property may be violated. However, for proving safety all that need be established is that at the *point of computation of the signal variable* this property holds. The next key lemma states this.

Lemma 3. *For all reachable states* $\mathbf{x}$, $H(\mathbf{x}) \Rightarrow H(\mathbf{x}_s)$ *where* $\mathbf{x}_S$ *is the state obtained by applying the Route and the Signal function to* $\mathbf{x}$.

Proof: Let us fix a reachable state $\mathbf{x}$, $\langle i,j\rangle \in ID$, and $\langle m,n\rangle \in Nbrs_{i,j}$ such that $\mathbf{x}.signal_{i,j} = \langle m,n\rangle$. Let $\mathbf{x}_R$

be the state obtained by applying the $Route$ function of Figure 4 to $\mathbf{x}$ and $\mathbf{x}_S$ be the state obtained by applying the $Signal$ function of Figure 5 to $\mathbf{x}_R$.

Without loss of generality, assume $\langle m,n\rangle = \langle i-1,j\rangle$, so if $\mathbf{x}.signal_{i,j} = \langle i-1,j\rangle$, then $\forall p \in \mathbf{x}.Members_{i,j}$, $\mathbf{x}.p_x - \frac{l}{2} \geq i + d$. First, we observe that $H(\mathbf{x}_R)$. This is because the $Route$ function does not change any of the variables involved in the definition of $H(.)$. Next, we show that $H(\mathbf{x}_R)$ implies $H(\mathbf{x}_S)$. There are two possible cases. First, if $\mathbf{x}_S.signal_{i,j} \neq \langle m,n\rangle$ then the statement holds vacuously. Second, when $\mathbf{x}_S.signal_{i,j} = \langle i-1,j\rangle$, the second condition in $H(\mathbf{x}_R)$ implies $H(\mathbf{x}_S)$. The cases where $\langle m,n\rangle$ takes the other values in $Nbrs_{i,j}$ follow by symmetry. ■

The following lemma asserts that if there is a cycle of length two formed by the $signal$ variables, then entity transfers cannot occur between the involved cells in that round.

Lemma 4. *Let* $\mathbf{x}$ *be any reachable state and* $\mathbf{x}'$ *be a state that is reached from* $\mathbf{x}$ *after a single* update *transition (round). If* $\mathbf{x}.signal_{i,j} = \langle m,n\rangle$ *and* $\mathbf{x}.signal_{m,n} = \langle i,j\rangle$, *then* $\mathbf{x}.Members_{i,j} = \mathbf{x}'.Members_{i,j}$ *and* $\mathbf{x}.Members_{m,n} = \mathbf{x}'.Members_{m,n}$.

Proof: No entities enter either $\mathbf{x}'.Members_{i,j}$ or $\mathbf{x}'.Members_{m,n}$ from any other $\langle a,b\rangle \in Nbrs_{i,j}$ or $\langle c,d\rangle \in Nbrs_{m,n}$ since $\mathbf{x}.signal_{i,j} = \langle m,n\rangle$ and $\mathbf{x}.signal_{m,n} = \langle i,j\rangle$. Assume without loss of generality that $m,n = \langle i-1,j\rangle$. It remains to be established that $\nexists p \in \mathbf{x}.Members_{i-1,j}$ such that $p \in \mathbf{x}'.Members_{i,j}$ or vice-versa. For this to occur, $\mathbf{x}.p_x$ is such that $\mathbf{x}'.p_x = \mathbf{x}.p_x + \frac{l}{2} + v > i$ by Figure 6, Line 5. But since $v < l$, it must have been the case that $\mathbf{x}.p_x - \frac{l}{2} < i + l + r_s$ by Figure 6, Line 5, a contradiction that $\mathbf{x}.signal_{i,j} = \langle i-1,j\rangle$. ■

Now we state and prove the safety property of System.

Theorem 5. *For any reachable state* $\mathbf{x}$, $Safe(\mathbf{x})$.

Proof: The proof is by standard induction over the length of any execution of System. The base case is satisfied by the initialization assumption. For the inductive step, consider reachable states $\mathbf{x}$, $\mathbf{x}'$ and an action a such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$. Fix $\langle i,j\rangle \in ID$ and assuming $Safe_{i,j}(\mathbf{x})$ show that $Safe_{i,j}(\mathbf{x}')$.

If $a = \mathsf{fail}_{i,j}$, then clearly $Safe(\mathbf{x}')$ as none of the entities move.

For $a =$ update, there are two cases to consider. First, $\mathbf{x}'.Members_{i,j} \subseteq \mathbf{x}.Members_{i,j}$. There are two sub-cases: if $\mathbf{x}'.Members_{i,j} = \mathbf{x}.Members_{i,j}$, then all entities in $\mathbf{x}.Members$ move identically and the spacing between two different entities $p \in Members_{i,j}$ and $q \in Members_{i,j}$ is unchanged. That is, $\forall p,q \in \mathbf{x}'.Members_{i,j}$ where $p.id \neq q.id$, $|\mathbf{x}'.p_x - \mathbf{x}'.q_x| = |\mathbf{x}.p_x + vc - \mathbf{x}.q_x - vc|$, where c is a constant. It follows that $|\mathbf{x}'.p_x - \mathbf{x}'.q_x| \geq d$. By similar reasoning it follows that $|\mathbf{x}'.p_y - \mathbf{x}'.q_y|$ is also at least d. The second sub-case arises if $\mathbf{x}'.Members_{i,j} \subsetneq \mathbf{x}.Members_{i,j}$, then $Safe_{i,j}(\mathbf{x}')$ is either vacuously satisfied or it is satisfied by the same argument as above.

The second case is when $\mathbf{x}'.Members_{i,j} \nsubseteq \mathbf{x}.Members_{i,j}$, that is, there exists an entity $p \in \mathbf{x}'.Members_{i,j}$ that was not in $\mathbf{x}.Members_{i,j}$. Let $p \in \mathbf{x}.Members_{i',j'}$ for some $\langle i',j'\rangle \in Nbrs_{i,j}$. If $\langle i,j\rangle \in SID$, then the specification of the source states that the entity p was added to $Members_{i,j}$ without violating $Safe_{i,j}(\mathbf{x}')$, and the proof is complete. Otherwise, $\langle i,j\rangle \notin SID$ and thus $p \in \mathbf{x}.Members_{i',j'}$. Without loss of generality, assume that $i' = i-1$ and $j' = j$. That is, p was transferred to $\mathsf{Cell}_{i,j}$ from its left neighbor. From Line 14 of Figure 6 it follows that $\mathbf{x}'.p_x = i + \frac{l}{2}$. The fact that p transferred from $\mathsf{Cell}_{i',j'}$ in $\mathbf{x}$ to $\mathsf{Cell}_{i,j}$ in $\mathbf{x}'$ implies that $\mathbf{x}.next_{i',j'} = \langle i,j\rangle$ and $\mathbf{x}.signal_{i,j} = \langle i',j'\rangle$—these are necessary conditions for the transfer. Thus, applying at state $\mathbf{x}$ the second inequality from $H(\mathbf{x})$, it follows that for every $q \in \mathbf{x}.Members_{i,j}$,

$$\mathbf{x}.q_x \geq i + d + \frac{l}{2}.$$

It must be established that if p is transfered to $\mathbf{x}'.Members_{i,j}$ it is the case that for every $q \neq p \in \mathbf{x}'.Members_{i,j}$,

$$\mathbf{x}'.q_x \geq i + d + \frac{l}{2},$$

which states that q does not move towards p. This follows by application of Lemma 4, which states that if entities on adjacent cells move towards one another simultaneously, a transfer of entities cannot occur. This implies that all entities q in $\mathbf{x}'.Members_{i,j}$ have edges greater than r_s of the edges of any such entity p, implying $Safe_{i,j}(\mathbf{x}')$, since $\mathbf{x}'.p_x = i + \frac{l}{2}$ and $\mathbf{x}'.q_x \geq i + d + \frac{l}{2}$, so $\mathbf{x}'.q_x - \mathbf{x}'.p_x \geq d$. Finally, since $\langle i,j\rangle$ was chosen arbitrarily, $Safe(\mathbf{x}')$. ■

Theorem 5 establishes that System is safe in spite of failures.

B. Stabilization of Routing and Progress

We show that under mild assumptions, once new failures cease to occurr, System recovers to a state where each entity on a non-faulty cell with a feasible path to the target makes progress towards the target.

We inductively define the *path distance* ρ of any cell $\langle i,j\rangle \in ID$ as the distance to the target through non-

faulty nodes. Let $\rho(\mathbf{x}, \langle i,j \rangle) =$

$$\begin{cases} \infty & \text{if } \mathit{failed}_{i,j}, \\ 0 & \text{if } \langle i,j \rangle = \mathit{tid}, \\ 1 + \min\limits_{\langle m,n \rangle \in \mathit{Nbrs}_{i,j} \cap NF(\mathbf{x})} \rho(\mathbf{x}, \langle m,n \rangle) & \text{otherwise.} \end{cases}$$

A cell is said to be *target connected* if its path distance is finite. We define

$$TC(\mathbf{x}) \triangleq \{\langle i,j \rangle \mid \rho(\mathbf{x}, \langle i,j \rangle) < \infty\}$$

as the set of cell identifiers that are connected to the target through non-faulty cells.

The analysis relies on the following assumptions on the environment of System which controls the occurrence of fail transitions and the insertion of entities by the source. (a) The target cell does not fail. (b) Source cells $\langle s,t \rangle \in SID$ place entities in $\mathit{Members}_{s,t}$ without blocking any of its nonempty non-faulty neighbors perpetually. That is, for any execution α of System, if there exists an $\langle i,j \rangle \in \mathit{Nbrs}_{s,t}$, such that for every state $\mathbf{x}$ in α after a certain round, $\langle i,j \rangle \in \mathbf{x}.\mathit{NEPrev}_{s,t}$, then eventually $\mathit{signal}_{s,t}$ becomes equal to $\langle i,j \rangle$ in some round of α. Let a fault-free execution fragment α be a sequence of states starting from $\mathbf{x}$ along which there are no fail$(\langle i,j \rangle)$ transitions for any $\langle i,j \rangle \in NF(\mathbf{x})$. Intuitively, a fault-free execution fragment is an execution fragment with no new failures, although for the first state $\mathbf{x}$ of α, $F(\mathbf{x})$ need not be empty.

Lemma 6. *Consider any reachable state $\mathbf{x}$ of* System *and any $\langle i,j \rangle \in TC(\mathbf{x}) \setminus \{\mathit{tid}\}$. Let $h = \rho(\mathbf{x}, \langle i,j \rangle)$. Any fault-free execution fragment α starting from $\mathbf{x}$, stabilizes in h rounds to a set of states S with all elements satisfying:*

$$\begin{aligned} \mathit{dist}_{i,j} &= h, \text{ and} \\ \mathit{next}_{i,j} &= \langle i_n, j_n \rangle, \text{where } \rho(\mathbf{x}, \langle i_n, j_n \rangle) = h - 1. \end{aligned}$$

Proof: First fix an arbitrary state $\mathbf{x}$, a fault-free execution fragment α starting from $\mathbf{x}$, and $\langle i,j \rangle \in TC(\mathbf{x}) \setminus \{\mathit{tid}\}$. We have to show that the set of states S defined by the above equations is closed under update transitions and that after h rounds, the execution fragment α enters S.

First, by induction on h we show that S is stable. Consider any state $\mathbf{y} \in S$ and a state $\mathbf{y}'$ that is obtained by applying an update transition to $\mathbf{y}$. We have to show that $\mathbf{y}' \in S$. For the base case, $h = 1$, so $\mathbf{y}.\mathit{dist}_{i,j} = 1$ and $\mathbf{y}.\mathit{next}_{i,j} = \mathit{tid}$. From Lines 2 and 4 of the *Route* function in Figure 4, and that there is a unique tid, it follows that $\mathbf{y}'.\mathit{dist}_{i,j}$ remains 1 and $\mathbf{y}'.\mathit{next}_{i,j}$ remains tid. For the inductive step, the inductive hypothesis is for any given h: if for any $\langle i',j' \rangle \in NF(\mathbf{x})$, $\mathbf{y}.\mathit{dist}_{i',j'} = h$ and $\mathbf{y}.\mathit{next}_{i',j'} = \langle m,n \rangle$, for some $\langle m,n \rangle \in \mathit{ID}$ with $\rho(\mathbf{x}, \langle m,n \rangle) = h - 1$, then

$$\mathbf{y}'.\mathit{dist}_{i',j'} = h \text{ and } \mathbf{y}'.\mathit{next}_{i',j'} = \langle m,n \rangle .$$

Now consider $\langle i,j \rangle$ such that $\rho(\mathbf{y}, \langle i,j \rangle) = \rho(\mathbf{y}', \langle i,j \rangle) = h + 1$. In order to show that S is closed, we have to assume that $\mathbf{y}.\mathit{dist}_{i,j} = h + 1$ and $\mathbf{y}.\mathit{next}_{i,j} = \langle m,n \rangle$, and show that the same holds for $\mathbf{y}'$. Since $\rho(\mathbf{y}', \langle i,j \rangle) = h + 1$, $\langle i,j \rangle$ does not have a neighbor with path distance smaller than h. The required result follows from applying the inductive hypothesis to $\langle m,n \rangle$, and from Lines 2 and 4 of Figure 4.

Next, we have to show that starting from $\mathbf{x}$, α enters S within h rounds. Once again, this is established by inducting on h, which is $\rho(\mathbf{x}, \langle i,j \rangle)$. The base case only includes the paths satisfying $h = \rho(\mathbf{x}, \langle i,j \rangle) = 1$ and follows by instantiating $\langle i_n, j_n \rangle = \mathit{tid}$. For the inductive case, assume that at round h, $\mathit{dist}_{i',j'} = h$ and $\mathit{next}_{i',j'} = \langle i_n, j_n \rangle$ such that $\rho(\mathbf{x}, \langle i_n, j_n \rangle = h - 1$ and $\langle i_n, j_n \rangle$ is the minimum identifier amongst all such cells. Observe that one such $\langle i',j' \rangle \in \mathit{Nbrs}(i,j)$ by the definition of TC. Then at round $h+1$, by Lines 2 and 4 of Figure 4, $\mathit{dist}_{i,j} = \mathit{dist}_{i',j'} + 1 = h + 1$. ∎

The following Corollary of Lemma 6 states that after new failures cease occurring, all target connected cells get their next variables set correctly within at most $O(N^2)$ rounds.

Corollary 7. *Consider any execution of* System *with arbitrary but finite sequence of* fail *transitions. Within $O(N^2)$ rounds of the last* fail*, every target connected cell $\langle i,j \rangle$ in* System *has $\mathit{next}_{i,j}$ fixed permanently to the identifier of the next cell along such a path.*

C. Progress of entities towards the target

Using the results from the previous sections, we show that once new failures cease occurring, every entity on a target-connected cell eventually gets to the target. The result (Theorem 10) uses two lemmas which establish that, along every infinite execution with a finite number of failures, every nonempty target-connected cell gets permission to move infinitely often (Lemma 9), and a permission to move allows the entities on a cell to make progress towards the target (Lemma 8). We start with the latter.

For the remainder of this section, we fix an arbitrary infinite execution α of System with a finite number of failures. Let $\mathbf{x}_f$ be the state of System at the round after the last failure, and α' be the infinite suffix $\mathbf{x}_f$, $\mathbf{x}_{f+1}, \ldots$ of α starting from $\mathbf{x}_f$. Observe that $TC(\mathbf{x}_f) = TC(\mathbf{x}_{f+1}) = TC(\ldots)$, so define TC to be $TC(\mathbf{x}_f)$.

Lemma 8. *For any $\langle i,j \rangle \in TC$, $k > f$, if $\mathbf{x}_k.\mathit{signal}_{m,n} = \langle i,j \rangle$ and $\mathbf{x}_k.\mathit{next}_{i,j} = \langle m,n \rangle$, then $\forall p \in$*

$(\mathbf{x}_k.Members_{i,j} \cap \mathbf{x}_{k+1}.Members_{i,j}) \cup (\mathbf{x}_k.Members_{i,j} \cap \mathbf{x}_{k+1}.Members_{m,n})$ *either*

$$\begin{aligned} |\mathbf{x}_{k+1}.p_x - m| &< |\mathbf{x}_k.p_x - m|, \text{ or} \\ |\mathbf{x}_{k+1}.p_y - n| &< |\mathbf{x}_k.p_y - n|. \end{aligned}$$

Proof: There are two cases. The first is when no entity transfers from $\langle i,j \rangle$ to $\langle m,n \rangle$ in the $k{+}1^{th}$ round. In this case, the result follows since velocity is applied towards $\langle m,n \rangle$ by *Move* in Figure 6, Lines 4 to 5. The second case is when some entity p transfers from $\langle i,j \rangle$ to $\langle m,n \rangle$, in which case $\mathbf{x}_{k+1}.p_x \in [m, m+1]$ and $\mathbf{x}_{k+1}.p_y \in [n, n+1]$ and the result follows. ■

Lemma 9. *Consider any $\langle i,j \rangle \in TC \setminus \{tid\}$, such that for all $k > f$, ($if \mathbf{x}_k.Members_{i,j} \neq \emptyset$, then $\exists k' > k$ such that $\mathbf{x}_{k'}.signal_{next_{i,j}} = \langle i,j \rangle$).*

Proof: Since $\langle i,j \rangle \in TC$, there exists $h < \infty$ such that for all $k > f$, $\rho(\mathbf{x}_k) = h$. We prove the lemma by inducting on h. The base case is $h = 1$. Fix $\langle i,j \rangle$ and instantiate $k' = f + 4$. By Lemma 6, for all non-faulty $\langle i,j \rangle \in Nbrs_{tid}$, $\mathbf{x}_f.next_{i,j} = tid$ since $k > f$. For all $k > f$, if $\mathbf{x}_k.Members_{i,j} \neq \emptyset$, and $signal_{tid}$ changes to a different neighbor with entities every round, within 4 rounds $signal_{tid} = \langle i,j \rangle$.

For the inductive case, let $k_s = k + h$ be the step in α after which all non-faulty $\langle a,b \rangle \in Nbrs_{i,j}$ have $\mathbf{x}_{k_s}.next_{a,b} = \langle i,j \rangle$ by Lemma 6. Also by Lemma 6, $\exists \langle m,n \rangle \in Nbrs_{i,j}$ such that $\mathbf{x}_{k_s}.dist_{m,n} < \mathbf{x}_{k_s}.dist_{i,j}$, implying that after k_s, $|\mathbf{x}_{k_s}.NEPrev_{i,j}| \leq 3$ since $\mathbf{x}_{k_s}.next_{i,j} = \langle m,n \rangle$ and $\mathbf{x}_{k_s}.next_{m,n} \neq \langle i,j \rangle$. By the inductive hypothesis, $\mathbf{x}_{k_s}.signal_{next_{i,j}} = \langle i,j \rangle$ infinitely often. If $\langle i,j \rangle \in SID$, then entity initialization does not prevent $\mathbf{x}_k.signal_{i,j} = \langle a,b \rangle$ from being satisfied infinitely often by the second assumption introduced in Subsection III-B. It remains to be established that $signal_{i,j} = \langle a,b \rangle$ infinitely often. Let $\langle a,b \rangle \in \mathbf{x}_{k_s}.NEPrev_{i,j}$ where $\rho(\mathbf{x}_{k_s}, \langle a,b \rangle) = h + 1$.

If $|\mathbf{x}_{k_s}.NEPrev_{i,j}| = 1$, then since the inductive hypothesis satisfies $signal_{next_{i,j}} = \langle i,j \rangle$ infinitely often, then Lemma 8 applies infinitely often, and thus $Members_{i,j} = \emptyset$ infinitely often, finally implying that $signal_{i,j} = \langle a,b \rangle$ infinitely often.

If $|\mathbf{x}_{k_s}.NEPrev_{i,j}| > 1$, there are two sub-cases. The first sub-case is when no entity enters $\langle i,j \rangle$ from some $\langle c,d \rangle \neq \langle a,b \rangle \in \mathbf{x}_{k_s}.NEPrev$, which follows by the same reasoning used in the $|\mathbf{x}_{k_s}.NEPrev| = 1$ case. The second sub-case is when a entity enters $\langle i,j \rangle$ from $\langle c,d \rangle$, in which case it must be established that $signal_{i,j} = \langle a,b \rangle$ infinitely often. This follows since if $\mathbf{x}_{k'}.token_{i,j} = \langle a,b \rangle$ where $k' > k_t > k_s$ and k_t is the round at which an entity entered $\langle i,j \rangle$ from $\langle c,d \rangle$, and the appropriate case of Lemma 3 is not satisfied, then $\mathbf{x}_{k'+1}.signal_{i,j} = \perp$ and $\mathbf{x}_{k'+1}.token_{i,j} = \langle a,b \rangle$ by Figure 5, Line 14. This implies that no more entities enter $\langle i,j \rangle$ from either cell $\langle c,d \rangle$ satisfying $\langle c,d \rangle \neq \langle a,b \rangle$. Thus $token_{i,j} = \langle a,b \rangle$ infinitely often follows by the same reasoning in the $|NEPrev| = 1$. ■

The final theorem establishes that entities that appear on any cell in TC eventually reach the target in α'.

Theorem 10. *Consider any $\langle i,j \rangle \in TC$, $\forall k > f$, $\forall p \in \mathbf{x}_k.Members_{i,j}$, $\exists k' > k$ such that $p \in \mathbf{x}_{k'}.Members_{next_{i,j}}$.*

Proof: Fix $\langle i,j \rangle \in TC$, a round $k > f$ and $p \in \mathbf{x}_k.Members_{i,j}$. Let $h = \max_{\langle i,j \rangle \in TC} \rho(\mathbf{x}_f, \langle i,j \rangle)$. By Lemma 6, at every round after round $k_s = k + h$ for any $\langle i,j \rangle \in TC$, the sequence of identifiers $\beta = \langle i,j \rangle, \mathbf{x}_{k_s}.next_{i,j}, \mathbf{x}_{k_s}.next_{\mathbf{x}_{k_s}.next_{i,j}}, \ldots$ forms a fixed path to tid. Applying Lemma 9 to $\langle i,j \rangle \in TC$ shows that there exists $k_m \geq k_s$ such that $\mathbf{x}_{k_m}.signal_{next_{i,j}} = \langle i,j \rangle$. Now applying Lemma 8 to $\mathbf{x}_{k_m}$ establishes movement of p towards $\mathbf{x}_{k_s}.next_{i,j}$, which is also $\mathbf{x}_{k_m}.next_{i,j}$. Lemma 9 further establishes that this occurs infinitely often, thus there is a round $k' > k_m$ such that p gets transferred to $\mathbf{x}_{k_m}.Members_{next_{i,j}}$. By a simple induction of the sequence of identifiers in the path β, it follows that p eventually gets consumed by the target. ■

IV. Simulation

We have performed several simulation studies of the algorithm for evaluating its throughput performance. In this section, we discuss the main findings with illustrative examples taken from the simulation results. Let the *K-round throughput* of System be the total number of entities arriving at the target over K rounds, divided by K. We define the *average throughput* (henceforth throughput) as the limit of K-round throughput for large K. All simulations start at a state where all cells are empty and subsequently entities are added to the source cell.

Throughput without failures as a function of r_s, l, v: Rough calculations show that throughput should be proportional to the cell velocity v, and inversely proportional to the safety distance r_s and the entity size l. Figure 7 shows throughput versus r_s for several choices of v for an 8×8 instance of System. The parameters are set to $l = 0.25$, $SID = \{\langle 1,0 \rangle\}$, $tid = \langle 1,7 \rangle$, and $K = 2500$. The entities move along the path $\beta \triangleq \langle 1,0 \rangle, \langle 1,1 \rangle, \langle 1,2 \rangle, \langle 1,3 \rangle, \langle 1,4 \rangle, \langle 1,5 \rangle, \langle 1,6 \rangle, \langle 1,7 \rangle$ with length 8. For the most part, the inverse relationship with v holds as expected: all other factors remaining the same, a lower velocity makes each entity take longer to move away from the boundary, which causes the predecessor cell to be blocked more frequently,

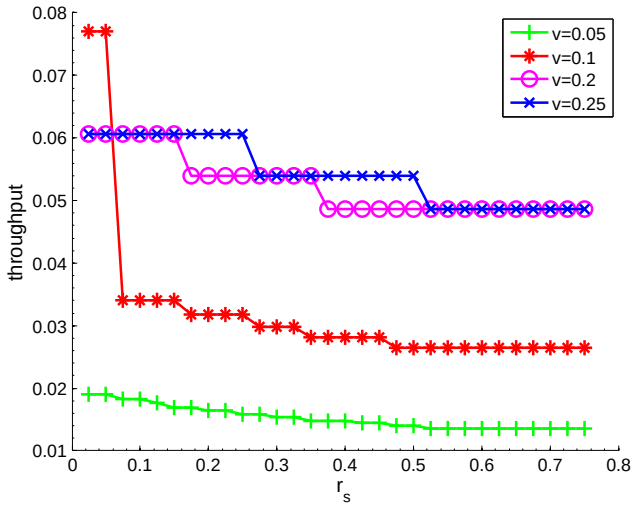

Figure 7. Throughput versus safety spacing r_s for several values of v, for $K = 2500$, $l = 0.25$ for 8×8.

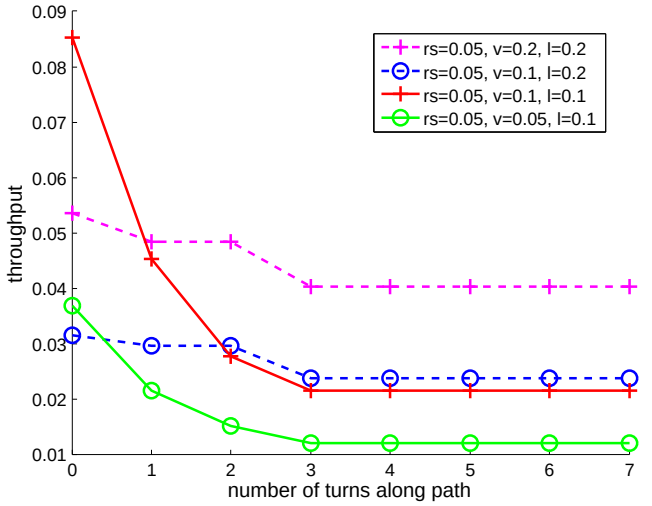

Figure 8. Throughput versus number of turns along a path, for a path of length 8, where $K = 2500$, $r_s = 0.05$, and each of l and v are varied for 8×8.

and thus fewer entities reach tid from any element of SID in the same number of rounds. In cases with low velocity (for example $v = 0.1$) and for very small r_s, however, the throughput can actually be greater than that at a slightly higher velocity. We conjecture that this somewhat surprising effect appears because at very small safety spacing, the potential for safety violation is higher with faster speeds, and therefore there are many more blocked cells per round. We also observe that the throughput saturates a certain value of r_s (≈ 0.55). This situation arises when there is roughly only one entity in each cell.

Throughput without failures as a function of the path: For a sufficiently large K, throughput is independent of the length of the path. This of course varies based on the particular path and system considered, but all other variables fixed, this relationship is observed. More interesting however, is the relationship between throughput and path complexity, measured in the number of turns along a path: Figure 8 shows throughput versus the number of turns along paths of length 8. This illustrates that throughput decreases as the number of turns increase, up to a point at which the decrease in throughput saturates. This saturation is due to signaling and indicates there exists only one entity per cell.

Throughput under failure and recovery of cells: Finally, we considered a random failure and recovery model in which at each round each cell fails with some probability p_f and each faulty cell recovers with some probability p_r [25]. A recovery simply sets $failed_{i,j} = false$ and in the case of tid also resets $dist_{tid} = 0$, so that eventually $Route$ will correct $next_{m,n}$ and

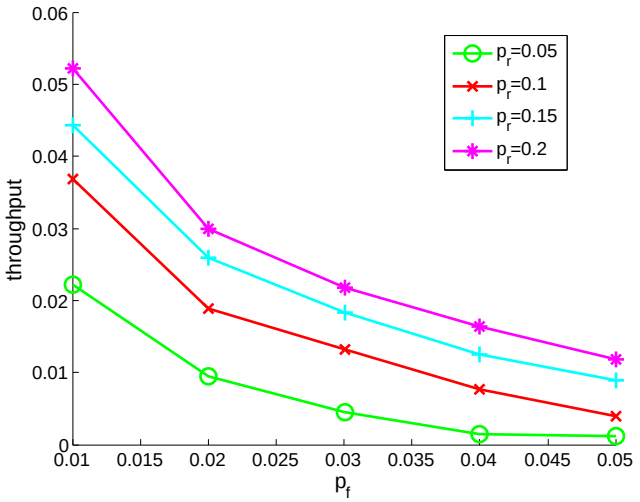

Figure 9. Throughput versus failure rate p_f for several recovery rates p_r with an initial path of length 8, where $K = 20000$, $r_s = 0.05$, $l = 0.2$ and $v = 0.2$ for 8×8.

$dist_{m,n}$ for any $\langle m, n \rangle \in TC$. Intuitively, we expect that throughput will decrease as p_f increases and increase as p_r increases. Figure 9 demonstrates this result for $0.01 \leq p_f \leq 0.05$ and $0.05 \leq p_r \leq 0.2$. Interestingly, there is roughly a marginal return on increasing p_r for a fixed p_f, in that for a fixed p_f increasing p_r results in smaller throughput gains.

V. Conclusion

We presented a self-stabilizing distributed traffic control protocol for the partitioned plane where each partition controls the motion of all entities within

that partition. The algorithm guarantees separation between entities in the face of crash failures of the software controlling a partition. Once new failures cease to occur, it guarantees progress of all entities that are not isolated by failed partitions to the target. Through simulations, we presented estimates of throughput as a function of velocity, minimum separation, path complexity, and failure-recovery rates.

Our algorithm is presented for a two dimensional square-grid partition, however, an extension to three dimensional rectangular partitions follows in an obvious way. The case for arbitrary tessalations of the plane seems interesting as well as challenging, particularly if the algorithms are to have asymptotically optimal throughput. A further generalization would be to develop algorithms for flow control of multiple types of entities with arbitrary flow patterns (not necessarily source-destination flows) specified for each type. Finally, for practical applications, we need algorithms that tolerate a relaxed coupling between entities and allow them some degree of independent movement while preserving safety and progress.

REFERENCES

[1] C. Daganzo, M. Cassidy, and R. Bertini, "Possible explanations of phase transitions in highway traffic," *Transportation Research Part A*, 1999.

[2] D. Helbing and M. Treiber, "Jams, waves, and clusters," *Science*, 1998.

[3] B. S. Kerner, "Experimental features of self-organization in traffic flow," *Phys. Rev. Lett.*, vol. 81, no. 17, pp. 3797–3800, 1998.

[4] M. Nolan, *Fundamentals of air traffic control*. Wadsworth Publishing Company, 1994.

[5] F. Borgonovo, L. Campelli, M. Cesana, and L. Coletti, "MAC for Ad hoc inter-vehicle network: services and performance," in *IEEE Vehicular Technology Conference*, vol. 5. Citeseer, 2003, pp. 2789–2793.

[6] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2008, pp. 15–28.

[7] T. Prevot, "Exploring the many perspectives of distributed air traffic management: The Multi Aircraft Control System MACS," in *Proceedings of the HCI-Aero*, 2002, pp. 149–154.

[8] N. Leveson, M. de Villepin, J. Srinivasan, M. Daouk, N. Neogi, E. Bachelder, J. Bellingham, N. Pilon, and G. Flynn, "A safety and human-centered approach to developing new air traffic management tools," in *Proceedings Fourth USA/Europe Air Traffic Management R&D Seminar*, 2001.

[9] C. Livadas, J. Lygeros, and N. A. Lynch, "High-level modeling and analysis of TCAS," in *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99),Phoenix, Arizona*, Dec 1999, pp. 115–125.

[10] J. A. Misener, R. Sengupta, and H. Krishnan, "Cooperative collision warning: Enabling crash avoidance with wireless technology," in *In 12th World Congress on Intelligent Transportation Systems*.

[11] A. Girard, J. de Sousa, J. Misener, and J. Hedrick, "A control architecture for integrated cooperative cruise control and collision warning systems," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 2, 2001, pp. 1491–1496 vol.2.

[12] C. Tomlin, G. Pappas, , and S. Sastry, "Conflict resolution of air traffic management: A study in multi-agent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, p. 509.

[13] C. Muñoz, V. Carreño, and G. Dowek, "Formal analysis of the operational concept for the Small Aircraft Transportation System," in *Rigorous Engineering of Fault-Tolerant Systems*, ser. LNCS, vol. 4157, 2006, pp. 306–325.

[14] D. Swaroop and J. K. Hedrick, "Constant spacing strategies for platooning in automated highway systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 121, p. 462470, 1999.

[15] E. Dolginova and N. Lynch, "Safety verification for automated platoon maneuvers: A case study," in *HART'97 (International Workshop on Hybrid and Real-Time Systems)*, ser. LNCS, vol. 1201. Springer Verlag, March 1997.

[16] P. Varaiya, "Smart cars on smart roads: Problems of control," *Automatic Control, IEEE Transactions on*, vol. 38, p. 195.

[17] H. Kowshik, D. Caveney, and P. R. Kumar, "Safety and liveness in intelligent intersections," in *HSCC*, ser. LNCS, vol. 4981, 2008, pp. 301–315.

[18] P. Weiss, "Stop-and-go science," vol. 156, no. 1, pp. 8–10, July 1999.

[19] Http://www.omniwheel.com/. [Online]. Available: http://www.omniwheel.com/

[20] S. Gilbert, N. Lynch, S. Mitra, and T. Nolte, "Self-stabilizing robot formations over unreliable networks," *In ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, July 2009.

[21] S. Dolev, L. Lahiani, S. Gilbert, N. Lynch, and T. Nolte, "Virtual stationary automata for mobile networks," in *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2005, pp. 323–323.

[22] T. Nolte and N. A. Lynch, "A virtual node-based tracking algorithm for mobile networks," in *ICDCS*, 2007.

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

[24] S. Owre, S. Rajan, J. Rushby, N. Shankar, and M. Srivas, "PVS: Combining specification, proof checking, and model checking," in *Computer-Aided Verification, CAV '96*, ser. LNCS, R. Alur and T. A. Henzinger, Eds., no. 1102. New Brunswick, NJ: Springer-Verlag, July/August 1996, pp. 411–414. [Online]. Available: http://www.csl.sri.com/papers/pvs-cav96/

[25] R. E. L. DeVille and S. Mitra, "Stability of distributed algorithms in the face of incessant faults," in *Proceedings of 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2009)*, Nov. 2009.