

Toward Predicting Distributed Systems Dynamics

Amy Kumar

University of Iowa

Iowa City, Iowa 52242

Email: amy-kumar@uiowa.edu

Jacob Beal

Raytheon BBN Technologies

Cambridge, MA, USA 02138

Email: jakebeal@bbn.com

Soura Dasgupta, Raghu Mudumbai

University of Iowa

Iowa City, Iowa 52242

Email: {raghuraman-mudumbai, soura-dasgupta}@uiowa.edu

Abstract—Systems of “building block” algorithms can guarantee that self-organizing systems eventually converge to a predictable state [1], [2], but what of their dynamical behavior in environments with ongoing changes? To begin to address this challenge, we analyze a commonly used distributed distance estimation algorithm from a stability theory perspective, identifying key properties of monotonicity and dynamical behavior envelope. This allows standard stability theory analysis to be applied to predict the behavior of the algorithm in response to persistent perturbation, both in isolation and as part of a composite system, as demonstrated both analytically and in simulation.

I. INTRODUCTION

One of the key ongoing challenges in the engineering of complex distributed systems is ensuring predictable behavior that is resilient to the broad range of configurations and challenges under which the system may need to operate. One promising recent approach has been to identify collections of “building block” algorithms with resilience properties that are preserved through composition [1], [2]. Previously, however, these properties have only been analytically established for systems’ steady-state behavior, with little understanding of their transient behavior. For large and complex distributed systems, however, such periods of stability occur rarely (if ever). It may also be problematic or even dangerous for a system to experience ill-constrained dynamical behavior.

Within the nonlinear systems community, however, there are a wide range of well-established methods for prediction and management of dynamical behavior [3]. If these methods could be brought to bear within a composable “building block” framework such as those developed in [1] and [2], it might provide a good path towards making complex systems dynamics more predictable and manageable. Here we present an initial investigation through a case study of a single widely-used “building block” algorithm: distributed distance estimation. We first review a common distance estimation algorithm and re-formulate it for nonlinear systems analysis (Section II), then establish key properties of monotonic convergence and use the transient behavior to predict the algorithm’s response to ongoing perturbations (Section III). Finally, we use these results to predict the stability of feedback systems comprising multiple instances of distance estimation (Section IV).

A. Notation

This paper discusses algorithms using both field calculus [4] and matrix theory. Field calculus is used to describe distributed algorithms since it is succinct and universal [5], provides

safely scoped composition of distributed algorithms [6] and guaranteed mapping between aggregate behavior and local interactions producing that behavior. Field calculus has also been used to produce “building block algebras,” of distributed algorithms with scalability and static behavior guarantees [1], [7], to which we aim to add dynamical guarantees. Field calculus programs are purely functional programs expressed using parentheses and prefix notation, e.g., $(+ 2 3)$ means “add two and three.” In this manuscript, we annotate additional information with color: special field calculus constructs are red, user-defined functions are blue, and standard built-in functions (e.g., math functions) are green.

Field calculus is not, however, well-suited for applying the mathematics of stability analysis. For that, we instead use matrix algebra with the following notation: individual values are plain type, vectors are lower case bold, matrices are upper case bold. Indices indicate subscripts (i.e., x_i is the i th entry in vector \mathbf{x} , $x_{i,j}$ is the i th row, j th column of matrix \mathbf{X}). Transposition is indicated with superscript T .

II. CASE STUDY: DISTRIBUTED DISTANCE ESTIMATE

For this first exploration of predicting composition of distributed systems via stability theory, we choose to focus on one commonly used algorithm: distributed distance estimation. We have selected self-stabilizing distributed distance estimation algorithms as these are fairly well-studied building blocks used in many different applications—a few of the many examples include distance-vector routing (e.g., [8]), ad hoc networking (e.g., [9]), data collection in sensor networks (e.g., [10]), biologically-inspired shape formation (e.g., [11], [12]), and chemical models of computation (e.g. [13]). More recently, it has been included in several efforts toward systematization of the design of resilient systems [1], [14], [15]. Surprisingly, however, even though there are known issues with convergence speed and instability that affect such algorithms, they do not appear to have previously been examined through the lens of stability theory. In this section, we thus present a simple implementation of self-stabilizing distances estimates and re-formalize it in terms of matrix algebra to facilitate the application of stability theory.

A. Simple Distance Estimation Algorithm

Here, we will consider one of the simplest distributed distance estimate algorithms. This algorithm is based on the Bellman-Ford algorithm [16], [17] in which distance is

Variable	Definition
$\mathbf{s}[t]$	Vector indicating that i is a source at time t by a 1 at s_i ; all other entries are 0.
$\mathbf{d}[t]$	True distance of devices to nearest source at time t
$\hat{\mathbf{d}}[t]$	Vector of distance estimates at time t
$\mathbf{N}[t]$	Distances to neighbors at time t : ∞ for self, non-neighbors
$\mathbf{C}[t]$	Matrix of triangle inequality constraints at time t
$\mathbf{x}[t]$	Vector of new constrained values at time t
$\Delta[t]$	Distance estimate error: $\hat{\mathbf{d}}[t] - \mathbf{d}[t]$
$\Delta^+[t]$	Greatest overestimate: $\max(0, \max_i \Delta_i[t])$
$\Delta^-[t]$	Least underestimate: $-\min(0, \min_i \Delta_i[t])$

Fig. 1. Key mathematical variables used in this manuscript.

estimated by the relaxation of the triangle inequality. The particular distributed algorithm variant that we consider may be specified succinctly in field calculus [4] as:

```
(def simple-distance-to (source)
  (rep d-hat
    infinity
    (mux source 0
      (min-hood (+ (nbr d-hat) (nbr-range))))))
```

In this simple algorithm, every device maintains a periodically updated variable \hat{d} (`d-hat`)¹ containing its current estimate of the distance to the nearest `source` device (designated by a Boolean indicator function²), initialized at infinity. The estimate is updated in one of two ways, depending on whether the device is currently indicated as a `source` (via the “multiplexing” branch `mux`, which shares information between devices that choose different branch options): sources directly set their estimate to zero, while all other devices apply the triangle inequality, setting their value to the minimum estimated distance through any neighboring device.³

This algorithm (and many variants) are known to be self-stabilizing [18], [19], meaning distance estimates are guaranteed to converge to correct values in the absence of ongoing perturbations to the source set, though the time required may be quite long if devices are very close together [20]. Its dynamical behavior, however, has not been formally analyzed.

B. Vector Mathematics Formalization

We now translate the field calculus algorithm above to an equivalent vector mathematics formalization suitable for stability analysis. With a network of n devices, the set of distance estimates at time t may be viewed as an $n \times 1$ vector:

$$\hat{\mathbf{d}}[t] = [\hat{d}_1[t], \hat{d}_2[t], \dots, \hat{d}_n[t]]^T \quad (1)$$

Denoting the true distance between neighboring devices i and j at time t as $d_{i,j}[t]$ (with $d_{i,j}[t] = \infty$ if i and j are equal or not neighbors), the matrix $\mathbf{N}[t]$ of all neighbor distances is:

$$\mathbf{N}[t] = \begin{pmatrix} \infty & d_{1,2} & \dots & d_{1,n} \\ d_{2,1} & \infty & \dots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \dots & \infty \end{pmatrix} \quad (2)$$

¹The `rep` construct creates a variable named by its first argument, initialized by its second argument, and periodically updated by its third argument

²The set of sources may change over time; the experiments in this paper change source position instead, which is equivalent for the systems considered.

³`nbr` returns a map from each neighbor to its argument’s most recent value, and `nbr-range` a map of distances to neighbors; `+` adds maps point-wise, and `min-hood` returns the lowest value in the range of the resulting map.

The triangle inequality constraint is then formed by adding this distance matrix with neighbors’ distance estimates. Those estimates are not communicated instantaneously, but after some time lag. For simplicity, we consider only the synchronous case, in which the time lag is always precisely one round.⁴ The triangle inequality constraint \mathbf{C} is thus:

$$\mathbf{C}[t] = \mathbf{u} \cdot \hat{\mathbf{d}}^T[t-1] + \mathbf{N}[t] \quad (3)$$

where \mathbf{u} is a length n unit vector. Note that this uses distance estimates $\hat{\mathbf{d}}$ from the prior round, not the current (which it is computing). The vector of new values $\mathbf{x}[t]$ asserted by this constraint are the minimums in each row:

$$x_i[t] = \min_j c_{i,j}[t] \quad (4)$$

Notice that if i has no neighbors, then x_i will be infinity, since $n_{i,j}$ is infinity for both self and all non-neighbor devices.

Finally, let $\mathbf{s}[t]$ be a vector denoting sources, whose i th value is 1 if the i th node is a source at time t , and 0 for all other elements. The update equation for $\hat{\mathbf{d}}$ is then set to zero for sources and by the triangle inequality otherwise:

$$\hat{\mathbf{d}}_i[t] = \begin{cases} x_i[t] & , s_i[t] = 0 \\ 0 & , s_i[t] = 1 \end{cases} \quad (5)$$

where \mathbf{I} is the identity matrix. In the default initial condition for the algorithm, $\hat{\mathbf{d}}[0]$ is infinity everywhere; in our analysis, however, we will consider any vectors of non-negative values as a possible initial state into which the system might be driven by some arbitrary perturbation.

III. MONOTONIC CONVERGENCE

Having expressed the `simple-distance-to` algorithm in system theoretic terms, we now analyze this system to show that it has the property of monotonic convergence. This property is important because it allows us to bound the impact that a particular perturbation can have on the values returned, thereby enabling analysis of system stability in terms of response to perturbation. This is particularly important because the time required for convergence may be extremely long, as [20] has established that the worst time for convergence of such simple distance estimates is inversely proportional to the shortest distance in the network.

The distance estimates in $\hat{\mathbf{d}}[t]$, however, are not monotonic, and neither are their errors $\Delta[t] = \hat{\mathbf{d}}[t] - \mathbf{d}[t]$ (where the true distance value is $\mathbf{d}[t]$). Consider, for example, the simple example illustrated in Figure 2, of a line network that initially contains both underestimates and overestimates. Over the course of just a few updates, the network reaches the correct values, but before it terminates, some devices that had initially held correct values go up and down in value multiple times.

We may note, however, that although the pattern of “ups” and “downs” cannot be predicted solely from local information, it proceeds across the network in an orderly fashion,

⁴These results can readily be generalized to a more general partially-synchronous time model, though the notation and analysis becomes significantly more intricate.

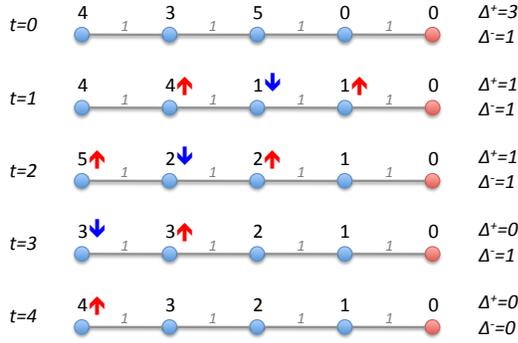


Fig. 2. Individual distance estimates may go up and down, but the greatest overestimate (Δ^+) and least underestimate (Δ^-) are monotonic. This example shows a line network of five devices (circles, source red, others blue) with unit edges (grey links); distance estimates evolve from initial $t = 0$ to converge to their correct values at $t = 4$.

as both bad and good information is propagated from lower-valued devices to higher-valued devices. Moreover, the incorrectness of the estimates is not increased by this propagation.

These observations suggest a different metric that does, in fact, prove to be monotonic: the worst error in the network. In particular, let us track the greatest overestimate Δ^+ and the least underestimate Δ^- , which are defined:

$$\Delta^+ = \max(0, \max_i \Delta_i[t]) \quad (6)$$

$$\Delta^- = -\min(0, \min_i \Delta_i[t]) \quad (7)$$

Both of these are non-negative monotonically decreasing functions. Moreover, if either is non-zero, it must undergo a strict decrease starting no more than *diameter* rounds after the initial time (i.e., long enough for an initial error near the source to propagate out to the devices farthest from the source).

Beyond that initial time, the two metrics behave quite differently, due to the asymmetry of the triangle inequality constraint. In particular, the greatest overestimate Δ^+ is guaranteed to be zero after at most *diameter* rounds: the triangle inequality constraint can bring the value of $\hat{d}_i[t]$ down arbitrarily quickly once information has propagated. The least underestimate Δ^- , however, is subject to the *rising value problem* identified in [20], in which transmission lag creates loops of mutual constraint that limit the rate at which device estimates can rise to as small as $\frac{1}{2} \min_{i,j} d_{i,j}$ per round.

Concerns of speed aside, however, we have at least the important property that convergence is monotonic, and thus the degree of disruption injected by a perturbation cannot grow over time, but can only shrink. This, in turn, will enable us to treat uses of this algorithm as simplified modules in the analysis of systems that use them.

A. Empirical Confirmation of Monotonicity

We confirm these properties empirically in simulation. Figure 3 shows results for simulation of 200 devices distributed randomly in a 100x100 meter environment except for a single source device placed at the center, communicating via a unit disc model with radius 15 meters and executing partially

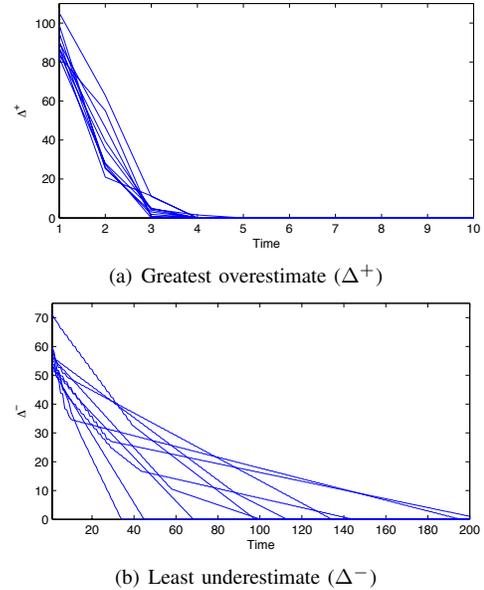


Fig. 3. Trace of greatest overestimate Δ^+ (a) and least underestimate Δ^- (b) for 10 runs of 200 devices randomly distributed in a 100x100 meter environment, communicating within 15 meters.

synchronously (same frequency, random phase), and with initial values uniformly randomly distributed in the range of $[0, 100]$. Figure 3(a) shows Δ^+ traces and Figure 3(b) Δ^- traces for ten simulations executed in Proto [21], each run asynchronously for 1000 simulated seconds with 1 second rounds.

Note that as predicted, both Δ^+ and Δ^- reduce monotonically to zero,⁵ with Δ^+ decreasing much more rapidly and predictably than Δ^- , due to the rising value problem (again, see [20]). The rising value problem is also reflected in the two-round “stair-step” pattern and inflections visible in many of the Δ^- traces: the “stair-steps” are caused by the passing back and forth of constraint between close pairs of devices, while inflections occur when there are two pairs of non-neighboring devices: one with initially worse estimates (thus initially dominating Δ^-) but the other significantly closer together (and thus improving their estimates more slowly).

B. Response to Persistent Perturbation

From the monotonicity property and our envelope analysis, we may then predict the response to a persistent perturbation. The value of Δ^+ drops arbitrarily fast, once information has had time to propagate, so in the limit for a large network we would expect Δ^+ to be bounded by the time for information to propagate times the amount of Δ^+ perturbation that can be injected each round (i.e., how incorrect a far-away device can become before information reaches it):

$$\Delta^+[t] \rightarrow \frac{d\Delta^+[t]}{dt} \cdot \text{diameter} \quad (8)$$

For example, if the perturbation is injected by a source moving with velocity v (while other devices remain stationary),

⁵Actually, very slightly above zero due to numerical imprecision.

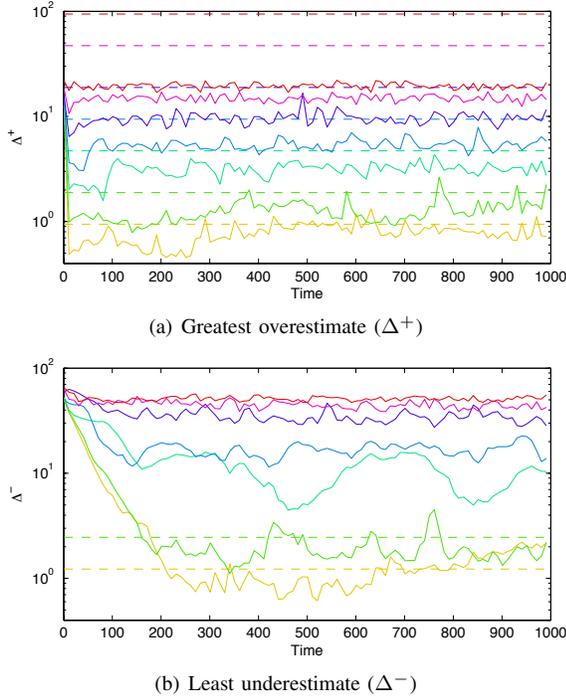


Fig. 4. With persistent perturbation by a randomly moving source, Δ^+ (a) scales linearly with diameter and velocity until error begins to be limited by network scale. Δ^- (b) behaves similarly as long as velocity is less than rise rate (which is constrained by the smallest link in the network); above that level, Δ^- may grow unbounded until it hits the limits of network scale. Data shown is model (dashed) and observed mean (solid) for 10-second sequences for 10 runs of 200 devices randomly distributed in a 100x100 meter environment, communicating within 15 meters, with a single source moving at velocity (indicated by hue) of 0.1 (orange) to 10 (red) meters per second.

then Δ^+ may be expected to converge toward a bound of $v \cdot \text{diameter}$. This relationship breaks down when v is very low or very high: at very low velocity, local topological changes introduce a network-dependent “floor” of possible effect sizes, since Δ^+ increases instantaneously when a connection breaks, based on the size of the new shortest path. On the high end, as long as the network remains connected, Δ^+ cannot be driven higher than the maximum distance between devices, so it will saturate as v rises.

We confirm these predictions empirically in simulations with the same conditions (200 random devices in a 100x100 meter environment, 15 meter communication, random values in $[0, 100]$), except the source moves to randomly chosen locations at velocity v , choosing a new random location each time it reaches its current target. Figure 4(a) shows mean Δ^+ across 10 runs and 10-second intervals, for a source moving at seven logarithmically distributed velocities ($v = 0.1, 0.2, 0.5, 1.0, 2.0, 5.0,$ and 10.0 meters/second). At low velocities, the model well predicts the approximate upper bound; at the highest velocities Δ^+ saturates as the moving source does not go many rounds between direction changes.

The value of Δ^- , on the other hand, does not drop so quickly, but is instead limited by the rising value problem and the shortest edge in the network. If new perturbations are

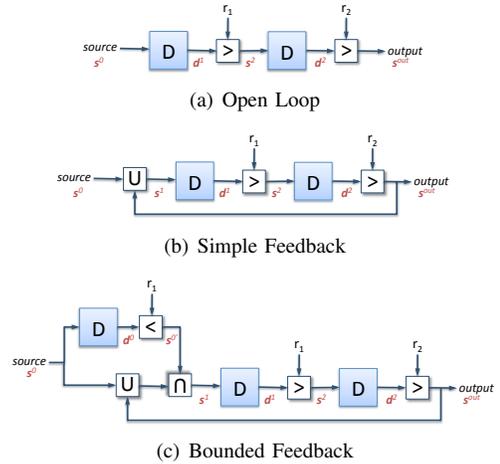


Fig. 5. Three distance cascade architectures used to test analytical results: open loop (a), simple feedback (b), and bounded feedback (c).

on average injected faster than Δ^- can decrease, then error may grow without bound, saturating at a value bounded by the greatest distance in the network (though behavior may be better in practice, depending on the particulars of geometry and perturbation). If Δ^- decreases faster on average than perturbations are added, however, then the situation is the same as for Δ^+ , with a perturbation lasting only as long as it takes for corrective information to arrive—in other words, exactly the same as Equation 8, only substituting Δ^- for Δ^+ .

Figure 4(b) compares this model with the empirical values of Δ^+ recorded in the same experiment reported in Figure 4(a). For these experiments, only the lowest two velocities ($v = 0.1$ and 0.2 meters/second) are predicted to be safe, and for these the model well predicts the observed error after the initial transient. For all higher velocities, error is predicted to grow without bound until it saturates, and indeed for $v \geq 2.0$ meters/second mean error saturates at a level consistent with estimates being essentially unable to correct. In between, at $v = 0.5$ and 1.0 meters/second, error has grown at much more than the linear rate of Equation 8, but the worst case limit is not yet being achieved.

As can be seen, identification of a monotonic property in the convergence of a distributed distance estimate algorithm allows its dynamical response to perturbation to be more easily abstracted. This abstraction then allows prediction of the algorithm’s dynamics in more complex circumstances, such as a persistent perturbation due to a moving source. In the next section, we will see how this abstraction also allows prediction of the properties of a more complex composite system that uses distributed distance estimation as a component.

IV. PREDICTING COMPOSITE SYSTEMS

In theory, once an algorithm has been mapped to an appropriate abstraction of its dynamical behavior (as we have done in the previous section), a wide range of mathematical tools should be applicable in order to predict its behavior under composition with other algorithms into more complex

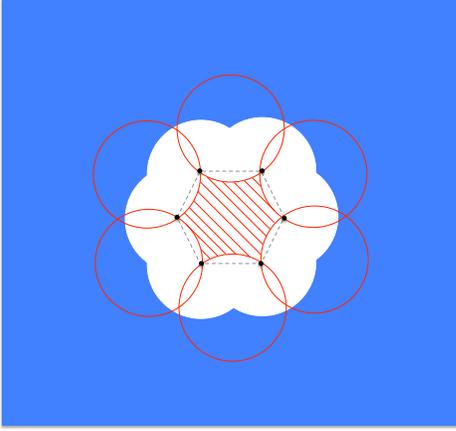


Fig. 6. Distance cascade example: thresholding distance r_1 from a hexagonal set of sources (black dots) selects the second source region (blue). Thresholding distance r_2 from this region selects an area in the interior of the hexagon (red cross-hatching) with boundaries determined by distance from critical points on the second source (red circles).

systems. To illustrate and evaluate this, let us consider a simple system of two distance estimates, arranged to compute a geometric relationship similar to the interior of a point-set. The basic computation is arranged following the system diagram in Figure 5(a), and an example of its application is shown in Figure 6. The computation begins with a set s^0 of sources—Figure 6 gives an example of six sources arranged in a hexagon. Distance is estimated to the source set, then compared with a threshold r_1 to select the set of devices farther than r_1 from any source. This is then used as the source set for a second distance estimate, which is compared against a second threshold r_2 to compute an output set of devices. The result is a region mutually enclosed by the original sources, somewhat similar to a convex hull.

In addition to this basic open-loop system, we also consider two versions that incorporate feedback. The simple feedback system in Figure 5(b) simply adds the output set back into the first distance computation by means of a union operation, while the second “bounded feedback” system in Figure 5(c) bounds feedback growth by also incorporating intersection with the set of devices within r_1 of the original sources. Notably, all three systems should maintain the exact same correct behavior if a static network is initialized with the solution and $r_1 \leq r_2$. Their dynamical behavior, however, is expected to be quite different, and may lead to instability and incorrect behavior.

A. Small Gain Prediction

Having previously established in Section III that each distance estimation block is individually stable, we can predict the stability of a composition of such blocks based on the small gain theorem [3], [22] or sometimes the passivity theorem [23]. The small-gain theorem says, in essence, that a feedback composition of stable systems is stable if the gain around the loop is less than one. The passivity theorem

likewise guarantees stability if there is no net generation of energy from one point to the other in the closed loop.

For the open loop system, the small-gain theorem holds trivially for all r_1 and r_2 , since there is no feedback. As long as perturbation rate is small enough for all Δ^- to remain stable, the system should maintain small errors per the self-composition of Equation 8. The simple feedback system, on the other hand, is only guaranteed to be stable for $r_1 < r_2$. Under this condition, devices selected in the output cannot maintain themselves through feedback, with the selected region shrinking by $r_2 - r_1$ in each cycle. Thus the system should ultimately converge to a pattern dictated by the original sources. With $r_1 = r_2$, however, a device perturbation may persist arbitrarily, and with $r_1 > r_2$ the selected region is expected to expand by $r_1 - r_2$ each cycle. Thus, this system is predicted to be stable only for $r_1 < r_2$. The bounded feedback system is expected to behave similarly, but is limited by the intersection operation to converge toward an expanded region no more than $r_1 + (r_1 - r_2)$ from the original set. These are conservative estimates, and particular topologies may induce better behavior. For example, the output will stop expanding in an unstable regime if it encounters a gap larger than $r_1 - r_2$.

B. Evaluation in Simulation

To evaluate our analytical predictions, we run a sequence of experiments based around perturbations of a set of sources, applying the analysis presented in the previous section to predict the behavior of the system in each case.

For all experiments, we consider a network of 300 devices, 8 meter unit disc communication and partially synchronous execution, all distributed randomly in a 100x100 meter environment except for six source devices that are arranged in a hexagon circumscribed by a circle of radius 10 meters and placed at the center of the environment. Three threshold conditions were tested: balanced ($r_1 = r_2 = 10$), expanding ($r_1 = 12, r_2 = 10$), and contracting ($r_1 = 10, r_2 = 12$). All distance estimates are initialized to infinity. Perturbations were injected by having all non-source devices moving to successive random points in space at $v = 0.1$ meters/second (sources remain stationary). Simulations were implemented and executed using Proto [21], running 10 trials for each condition for 1000 simulated seconds per trial with one second rounds, recording the output every 10 simulated seconds.

In theory, such a set of sources should select an inward-curved star-like region like that illustrated in Figure 6; in practice, even ideal execution will produce at least some error due to discretization (in this case, only a small amount, per [24]). Performance was then evaluated by computing the fraction of devices for which the output disagreed with the theoretically correct output given the geometry of the sources (i.e., true when it should be false or vice versa).

Figure 7 summarizes the results of these simulations, which conform with our predictions in all cases: the open loop system has a very low level of error for all thresholds, simple feedback rapidly expands to many false positives for $r_1 > r_2$ and more slowly for $r_1 = r_2$, but performs as well as open loop for

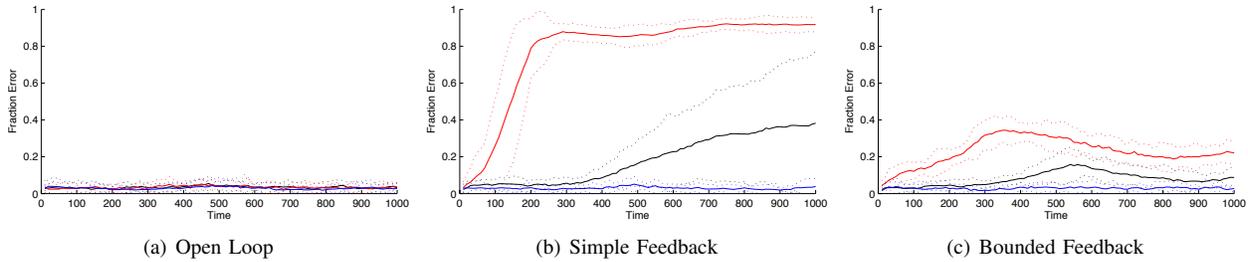


Fig. 7. Comparison of stability across system architecture and relative thresholds. Line color indicates thresholds: balanced $r_1 = 10$, $r_2 = 10$ (black), expanding $r_1 = 12$, $r_2 = 10$ (red), or contracting $r_1 = 10$, $r_2 = 12$ (blue). Dotted lines show ± 2 standard deviations. As predicted, in this simple system, open loop always performs well, while feedback only consistently performs well for the bounded architecture with contracting thresholds.

$r_1 < r_2$. Bounded is also unstable for $r_1 \geq r_2$, but more limited in the impact from this instability, and is stable and performs as well as open loop for $r_1 < r_2$.

V. CONTRIBUTIONS

We have seen that the dynamical behavior of a common distributed distance estimation algorithm can be analyzed and quantitatively predicted using a nonlinear systems approach, and further that this analysis can be combined using well-established mathematical tools in order to predict the behavior of feedback systems in which such algorithms are composed. These results are promising and encourage further development of this approach of bringing together nonlinear systems techniques and the “building block” approach to the engineering of complex distributed systems. Next steps for this research include more thorough analysis and evaluation of the systems discussed herein, with the goal of expanding the range of properties that can be predicted and the generality of the results, as well as expansion of these methods to broadly applicable collections of building block algorithms such as those of the set proposed in [1].

ACKNOWLEDGMENT

This work has been supported by the United States Air Force and the Defense Advanced Research Projects Agency under Contract No. FA8750-10-C-0242. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views, opinions, and/or findings contained in this article are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for public release; distribution is unlimited.

REFERENCES

- [1] Jacob Beal and Mirko Viroli, “Building blocks for aggregate programming of self-organising applications,” in *Workshop on Foundations of Complex Adaptive Systems (FOCAS)*, 2014.
- [2] Mirko Viroli and Ferruccio Damiani, “A calculus of self-stabilising computational fields,” in *Coordination 2014*, 2014, pp. 163–178.
- [3] Hassan K. Khalil, *Nonlinear Systems*, Prentice Hall, 2002.
- [4] Mirko Viroli, Ferruccio Damiani, and Jacob Beal, “A calculus of computational fields,” in *Advances in Service-Oriented and Cloud Computing*, Carlos Canal and Massimo Villari, Eds., vol. 393 of *Communications in Computer and Information Sci.*, pp. 114–128. Springer Berlin Heidelberg, 2013.
- [5] Jacob Beal, Mirko Viroli, and Ferruccio Damiani, “Towards a unified model of spatial computing,” in *7th Spatial Computing Workshop (SCW 2014)*, AAMAS 2014, Paris, France, May 2014.
- [6] Ferruccio Damiani, Mirko Viroli, Danilo Pianini, and Jacob Beal, “Code mobility meets self-organisation: a higher-order calculus of computational fields,” in *35th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems*, 2015.
- [7] Jacob Beal and Mirko Viroli, “Space-time programming,” *Philosophical Trans. of the Royal Society A*, vol. 373, no. 2046, pp. 20140220, 2015.
- [8] Charles L Hedrick, “Routing information protocol,” Tech. Rep. RFC 1058, IETF, 1988.
- [9] Charles Perkins, Elizabeth Belding-Royer, and Samir Das, “Ad hoc on-demand distance vector (AODV) routing,” Tech. Rep. RFC 3561, IETF, 2003.
- [10] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin, “Directed diffusion: a scalable and robust communication paradigm for sensor networks,” in *6th International Conference on Mobile Computing and Networking*. ACM, 2000, pp. 56–67.
- [11] Radhika Nagpal, *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*, Ph.D. thesis, MIT, Cambridge, MA, USA, 2001.
- [12] Daniel Coore, *Botanical Computing: A Developmental Approach to Generating Inter connect Topologies on an Amorphous Computer*, Ph.D. thesis, MIT, Cambridge, MA, USA, 1999.
- [13] Mirko Viroli, Matteo Casadei, Sara Montagna, and Franco Zambonelli, “Spatial coordination of pervasive services through chemical-inspired tuple spaces,” *ACM Transactions on Autonomous and Adaptive Systems*, vol. 6, no. 2, pp. 14:1 – 14:24, June 2011.
- [14] Radhika Nagpal, “A catalog of biologically-inspired primitives for engineering self-organization,” in *Engineering Self-Organising Systems*, pp. 53–62. Springer, 2004.
- [15] JoseLuis Fernandez-Marquez, Giovanna Marzo Serugendo, Sara Montagna, Mirko Viroli, and JosepLluís Arcos, “Description and composition of bio-inspired design patterns: a complete overview,” *Natural Computing*, vol. 12, no. 1, pp. 43–67, 2013.
- [16] RE Bellman, “On a routing problem,” *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.
- [17] Lester R. Ford Jr., “Network flow theory,” Tech. Rep. Paper P-923, RAND Corporation, 1956.
- [18] Shlomi Dolev, *Self-Stabilization*, MIT Press, 2000.
- [19] M Schneider, “Self-stabilization,” *ACM Computing Surveys*, vol. 25, pp. 45–67, 1993.
- [20] Jacob Beal, Jonathan Bachrach, Dan Vickery, and Mark Tobenkin, “Fast self-healing gradients,” in *ACM Symp. on Applied Computing*, 2008.
- [21] Jacob Beal and Jonathan Bachrach, “Infrastructure for engineered emergence in sensor/actuator networks,” *IEEE Intelligent Systems*, vol. 21, pp. 10–19, March/April 2006.
- [22] Zhong-Ping Jiang, Iven M.Y. Mareels, and Yuan Wang, “A lyapunov formulation of the nonlinear small-gain theorem for interconnected ISS systems,” *Automatica*, vol. 32, pp. 1211 – 1215, 1996.
- [23] Minyue Fu and Soura Dasgupta, “Parametric lyapunov functions for uncertain systems: The multiplier approach,” *Advances in linear matrix inequality methods in control*, pp. 95–108, 2000.
- [24] Jonathan Bachrach, Jacob Beal, Joshua Horowitz, and Dany Qumsiyeh, “Empirical characterization of discretization error in gradient-based algorithms,” in *IEEE SASO*, 2008, pp. 203–212.