

An Aggregate Computing Approach to Self-Stabilizing Leader Election

Yuanqiu Mo
University of Iowa
Iowa City, Iowa 52242
Email: yuanqiu-mo@uiowa.edu

Jacob Beal
Raytheon BBN Technologies
Cambridge, MA, USA 02138
Email: jakebeal@ieee.org

Soura Dasgupta
University of Iowa*
Iowa City, Iowa 52242
Email: soura-dasgupta@uiowa.edu

Abstract—Leader election is one of the core coordination problems of distributed systems, and has been addressed in many different ways suitable for different classes of systems. It is unclear, however, whether existing methods will be effective for resilient device coordination in open, complex, networked distributed systems like smart cities, tactical networks, personal networks and the Internet of Things (IoT). Aggregate computing provides a layered approach to developing such systems, in which resilience is provided by a layer comprising a set of adaptive algorithms whose compositions have been shown to cover a large class of coordination activities. In this paper, we show how a feedback interconnection of these basis set algorithms can perform distributed leader election resilient to device topology and position changes. We also characterize a key design parameter that defines some important performance attributes: Too large a value impairs resilience to loss of existing leaders, while too small a value leads to multiple leaders. We characterize the smallest value of this parameter for which the only stationary points have single leaders, and demonstrate resilience of this algorithm through simulations.

I. INTRODUCTION

The last few decades have witnessed a proliferation of systems like smart-cities, tactical networks, personal networks and the Internet of Things (IoT), that are not only complex, networked, and distributed, but also open in the sense that they must support an unbounded and rapidly evolving collection of distributed services. In all of these, realizing the full potential of these systems requires devices to interact *safely and seamlessly* with others in their vicinity through low latency peer to peer communication, and to share tasks. These open systems cannot effectively meet user needs unless they support frequent and non-centralized changes in the applications and services being hosted. By contrast, current modes of device interactions are typically either highly constrained and inflexible (e.g., single-use devices) or else rely on remote infrastructure like cloud services. The former impairs reusability and prevents devices from contributing to multiple overlapping applications. The latter is centralized with high latency and lacks the agility to exploit local communication, services, and devices.

Aggregate computing offers a potential approach to this challenge, based on viewing the basic computing unit as a physical region with a collection of interacting computing devices, rather than an individual physical device [1]. It involves

* Prof. Dasgupta is also a Visiting Professor at Shandong Computer Science Center, Shandong Provincial Key Laboratory of Computer Networks, China.

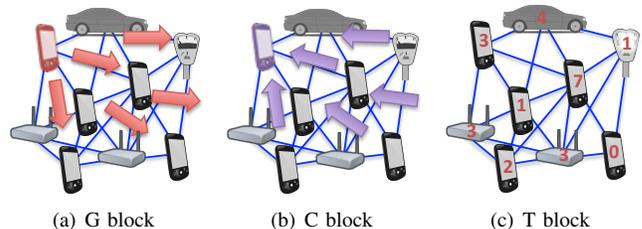


Fig. 1. Illustration of three basis block operators: (a) information-spreading (G), (b) information aggregation (C), and (c) temporary state (T)

a separation of concerns into multiple abstraction layers, much like the OSI model for communication [2], factoring the overall task of distributed system design into sub-tasks of device-level communication and discovery, coherence between collective and local operations, resilience, programmability, and applications programming.

The lowest layers comprise fundamental device interactions and a small universal calculus of aggregate level field calculus constructs, implementing services such as neighborhood discovery and distributed scoping of shared information, agnostic to the protocols invoking them. The next layer facilitates resilient device interactions and comprises classes of basis set modules that are themselves distributed algorithms. Currently, there are three such classes: (i) *G*-blocks that spread information through a network of devices, (ii) *C*-blocks that summarize salient information about the network to be used by interacting units, and (iii) *T*-blocks that maintain temporary state. Introduced in [3] and [4], and depicted in Figure 1, one can show that a broad class of dispersed services can be described by various compositions of these three blocks. The blocks interact seamlessly through distributed interactions, blind to each other’s internal computations, and agnostic to the applications and protocols that invoke them, the latter comprising the highest layer.

A frequent task in applications like IoT is for a network of devices to elect a leader that can serve as a coordinator, e.g., for resource and task allocation. Leader election (and related symmetry breaking problems) are of course, a long standing and well studied problem in the distributed algorithms community [5]. More recently, the notion of resilience to perturbations has been formalized in that community in the form of self-

stabilization [6], [7], and a number of self-stabilizing leader election algorithms have been introduced (e.g., [8]–[11]). These algorithms, however, guarantee only that a safe state will be reached eventually, but provide no guarantees about the behavior of the system during recovery from perturbations. In a complex environment with many devices, as in many IoT applications and other large-scale distributed systems, small perturbations due to device mobility occur frequently and it is difficult to determine if such algorithms will retain stability under typical conditions of persistent perturbations.

In this paper we present a different approach, exploiting the dynamical systems analysis performed on aggregate computing blocks, to formulate a leader election algorithm amenable to analysis not just of its converged state but also of its dynamical behavior. We use a feedback combination of G and C blocks to perform leader election in a resilient, distributed manner (Section II). We observe that we have previously analyzed distinguished G and C blocks for their behavior under persistent perturbations, [12], [13] and [14]. The algorithm has one free design parameter that defines certain important performance attributes. Too small a value of this parameter will lead to multiple leaders. Too large a value will impair resilience by delaying recovery from loss of current leaders. A key analytical result, presented in Section III, is thus to characterize the smallest value of this parameter, so that the only possible convergence is to a single leader. Section IV confirms resilience via simulations, and Section V concludes.

II. THE LEADER ELECTION ALGORITHM

This section describes our leader election algorithm. This algorithm must elect a single leader from the nodes $\{1, 2, \dots, N\}$ of an undirected graph \mathcal{G} in a *distributed manner that is resilient to the loss of nodes, including leaders, and edges, regardless of initial state*. Each node has a priority, and contentions between potential leaders is resolved in favor of the higher priority nodes. *Without loss of generality, we will assume that a node i has a higher priority than node $i + 1$.* Two nodes i, j are neighbors if they share an edge. This edge length will be denoted $e_{ij} > 0$. A node only communicates with those in its set of neighbors, $\mathcal{N}(i)$. A subset $S(t)$ of the nodes in the graph will form a leader or source set at time t . The goal is to ensure that

$$\lim_{t \rightarrow \infty} |S(t)| = 1, \quad (1)$$

i.e., $S(t)$ eventually comprises only one element.

A definition of *pseudodiameter* is crucial for our algorithm:

Definition 1. *A node is attached to a source if that source is the nearest source to it (breaking ties by priority). Pseudodiameter at a source is the largest distance from the source in the union of the set of nodes attached to it and their neighbors.*

Thus in Figure 2(a), nodes 3, 4, and 5 are attached to source 1, while 6 and 7 are attached to source 2. The pseudodiameter at 1 is thus 4, as the union of the set of nodes attached to 1 and its neighbors is $\{1, 3, 4, 5, 6\}$, and 6, the node furthest from 1 in this set is at a distance 4 from it.

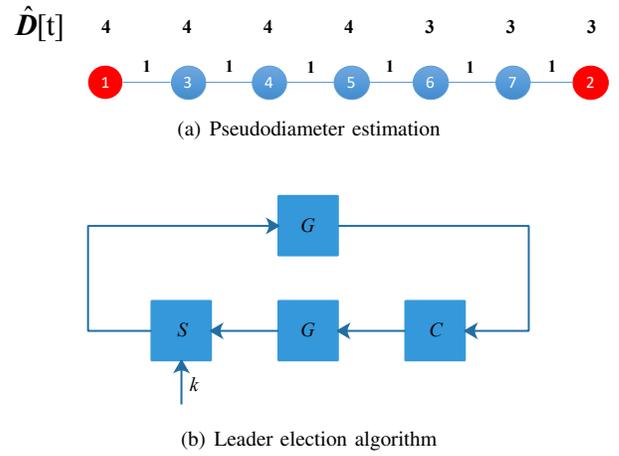


Fig. 2. (a) Illustration of pseudodiameter estimation for two sources, 1 and 2, on a line graph where all edges are length 1. The top numbers are the pseudodiameter estimates for each node. (b) Block diagram of leader election. The top G block is a distance estimate algorithm; C block provides estimated pseudodiameters to the leaders; the lower G block broadcasts each leader's pseudodiameter estimate to nodes attached to it. The block labeled S , itself a G -block, changes leaders.

Our algorithm is a closed loop of four aggregate computing blocks following the diagram in Figure 2(b):

- The top G block estimates the shortest distance $\hat{d}_i(t)$ from node i to the nearest element in $S(t)$.
- The C block collects and sends to each source i its current pseudodiameter estimate $D_i(t)$.
- The lower G block *broadcasts* $D_i(t)$ to each node attached to source i . Thus at the conclusion of the broadcast operation a node j attached to node i carries the pseudodiameter estimate $D_i(t)$. For example, in Figure 2(a) nodes 3 and 4, attached to source 1, each carry the pseudodiameter estimate 4.
- The block labeled S , driven by a design parameter k , either suppresses or creates sources. It is itself a G block that spreads to each node j within $kD_j(t)$ from a source its distance from the source and the latter's priority. Should j itself be a source and find that there is a higher priority source within $kD_j(t)$ from it, then j ceases to be a source. If on the other hand a non-source node j cannot find a higher priority source within $kD_j(t)$ from it, then it becomes a source. Thus in Figure 2(a) if $k = 1$ then neither source is suppressed as neither 1 nor 2 are respectively within $kD_1(t) = 4$ and $kD_2(t) = 3$ within each other. On the other hand with $k = 2$, the lower priority source 2 is suppressed as it is less than or equal to $kD_2(t) = 6$ away from source 1.

All these blocks operate *iteratively and in tandem* with only *nearest neighbor information exchange*. Thus, the C block does not wait for the distance estimation in the top G block to conclude before it commences its operations. Further its pseudodiameter estimation also occurs recursively. On the face of it, choosing a very large k should allow (1) to hold. *However, as argued in Section III, a larger k prolongs the*

time to recovery from a lost source. Indeed the key analytical contribution of this paper is to characterize the smallest k that ensures (1). In the remainder of this section we present the state equations that define the closed loop dynamics.

A. The G block for distance estimation

The goal of the top G block in Figure 2(b) is to find the shortest distance from each non-leader node to the nearest leader. This block implements the Adaptive Bellman-Ford (ABF) algorithm analyzed in [14], [15], in which it has been proved to be globally uniformly asymptotically stable and robust to perturbations caused by node mobility. Unlike the classical Bellman-Ford algorithm, [16], [17], it permits the initial distance estimates to be underestimates. As classical Bellman-Ford mandates that all initial estimates be overestimates it cannot withstand source perturbations induced at the input of this block by the S block.

Defining the estimated distance for node i from the source set $S(t)$ in the t -th iteration as $\hat{d}_i(t)$, the algorithm follows

$$\hat{d}_i(t+1) = \begin{cases} \min_{j \in \mathcal{N}(i)} \{\hat{d}_j(t) + e_{ij}\} & i \notin S(t) \\ 0 & i \in S(t) \end{cases} \quad (2)$$

The set $S(t)$, serving as input, is from the S block, and $\hat{d}_i(0)$ may be initialized arbitrarily, though per [14] it is expected to converge most quickly if all values are initially overestimates (e.g., infinity); $D_i(0)$ below may likewise be initialized arbitrarily.

B. The C block

This block collects and conveys to each source its estimated pseudodiameter as defined in Definition 1. Its inputs are estimated distance $\hat{d}_i(t)$ from the source set computed by the prior G block, and certain distinguished neighbors computed by G and defined below.

Definition 2. A minimizing j in the first case of (2) is a current constraining node $c_i(t)$ of i at time t . A source is its own current constraining node. In other words

$$c_i(t) = \begin{cases} \arg \min_{j \in \{1, \dots, N\}} \{\hat{d}_j(t) + e_{ij}\} & i \notin S(t) \\ i & i \in S(t) \end{cases} \quad (3)$$

Note $c_i(t)$ need not be unique. We define $\mathcal{C}_i(t)$ as the set of nodes for which i is a current constraining node at time t excepting i itself.

With $\hat{d}_i(t)$ and $\mathcal{C}_i(t)$ as inputs, the C block provides each leader a current estimated pseudodiameter, as defined in Definition 1.

Define $r_i(t)$ as a temporary variable that at steady state has the pseudodiameter value at the sources. Its values at non-source nodes are not the pseudodiameter defined in Definition 1. Specifically for node i at time t , there holds:

$$r_i(t+1) = \max\{\hat{d}_i(t+1) + e_{i\max}, \{r_j(t) | j \in \mathcal{C}_i(t)\}\} \quad (4)$$

where $e_{i\max} = \max_{j \in \mathcal{N}(i)} \{e_{ij}\}$. When the C block stabilizes, each leader will get its estimated pseudodiameter.

C. Broadcast

The lower G broadcasts the pseudodiameter estimate at a source to the nodes attached to that source. It receives as input the $r_i(t)$ from the C block, and although not shown in Figure 2(b) the current constraining nodes from the top G block. Each node updates its pseudodiameter estimate to the value held at its current constraining node:

$$D_i(t) = \begin{cases} r_i(t) & c_i(t) = i \\ D_j(t-1) & c_i(t) = j \end{cases} \quad (5)$$

D. The S block

The S block suppresses a source i if it is within $kD_i(t)$ of a higher priority source, and chooses i as a leader if there is no higher priority source within $kD_i(t)$ of i . The state equations of this block are ABF with a twist. The distance estimate $\bar{d}_i(t)$ at i is its estimated distance from the highest priority source that is within $kD_i(t)$ of i . Thus i also receives $\sigma_i(t)$, the index of this highest priority source. Recall, i itself reflects the index, in that i has higher priority than $i+1$. Thus by definition

$$S(t) = \{i \in \{1, \dots, N\} \mid \sigma_i(t) = i\} \quad (6)$$

Like ABF $\bar{d}_i(t)$ is updated with respect to one of its neighbors. This neighbor must have a $\bar{d}_i(t)$ no more than $kD_i(t)$. Thus we define a valid set of neighbors of i as,

$$V_i(t) = \{j \in \mathcal{N}(i) \mid \bar{d}_j(t) + e_{ij} \leq kD_i(t)\} \quad (7)$$

To make $\bar{d}_i(t)$ the distance from the highest priority node we further exclude neighbors by choosing one which carries the distance from a source that has the highest priority. The neighbor picked is in the set

$$M_i(t) = \{j \in V_i(t) \mid \sigma_j(t) = \min_{l \in V_i(t)} \sigma_l(t)\}, \quad (8)$$

i.e., those valid neighbors with the highest priority source for their distance estimates. Tie breaks can be arbitrary, and all $j \in M_i(t)$ carry identical values of $\sigma_j(t)$.

Node i updates $\sigma_i(t)$, the index of the source from which it must estimate its distance, using (9) with any $j \in M_i(t)$:

$$\sigma_i(t+1) = \begin{cases} \sigma_j(t) & V_i(t) \neq \emptyset \text{ and } \sigma_j(t) < i \\ i & V_i(t) = \emptyset \text{ or } \sigma_j(t) \geq i \end{cases} \quad (9)$$

Thus, if $V_i(t)$, i 's set of valid neighbors, is empty then i has no sources within $kD_i(t)$, and must designate itself a source and will measure its distance from itself, i.e., $\sigma_i(t+1) = i$, and in (10) below its new distance estimate will be zero. Likewise $\sigma_{M_i(t)}(t) \geq i$ indicates that though i is within $kD_i(t)$ of a source, that source has a lower priority than i and i must become a source. On the other hand if $\sigma_{M_i(t)}(t) < i$ then the neighbor $M_i(t)$ has a distance estimate from a higher priority source within $kD_i(t)$ of i , and i must now find its distance from this source as in (10), i.e., over the set of $j \in M_i(t)$:

$$\bar{d}_i(t+1) = \begin{cases} \min_{l \in M_i(t)} \{\bar{d}_l(t) + e_{il}\} & V_i(t) \neq \emptyset \text{ and } \sigma_j(t) < i \\ 0 & V_i(t) = \emptyset \text{ or } \sigma_j(t) \geq i \end{cases} \quad (10)$$

III. DESIGNING k

The only design parameter in the Leader Election algorithm given in Section II is k . Taking a stationary point to be a topology and state such that if topology does not change then neither does state, to ensure (1) *it is necessary that stationary points cannot have two leaders*. Thus k must be large enough so that at a stationary point at least two among multiple sources are within k times the pseudodiameter estimates they hold. This holds the prospect of the lower priority source among them being suppressed. Too small a k would of course mandate multiple sources, as for example with $k = 1$ in Figure 2(a).

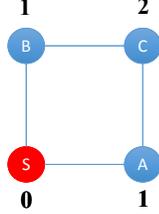


Fig. 3. Illustration regarding the impact of gain k on resilience.

The larger that k is, however, the longer the delay in recovery from the loss of a source. Consider the graph in Figure 3 with a single leader S . Suppose this leader is lost. All its neighbors know is that they have lost a neighbor, but do not know if it is still in the graph, connected by another route. The ABF in the top G -block continues apace. Thus at successive rounds the distance estimates $[\hat{d}_A, \hat{d}_B, \hat{d}_C]$ are $[3, 3, 2]$, $[3, 3, 4]$, etc. This increase in distance estimates continues until at least one node exceeds k times its pseudodiameter estimate and becomes a source, and the time this takes is proportional to k .

A. Characterizing k

We now show that for $k \geq 2$, a stationary point cannot have multiple sources in a *fixed graph*. On the other hand, by choosing edge e_{27} arbitrarily small in the graph of Figure 2(a), it is readily verified that for all $k < 2$, one can have a stationary point in which both 1 and 2 are sources. Thus the smallest value of k that permits convergence to a single leader is $k = 2$.

We will denote the states of the stationary point by dropping the argument t . For example $\hat{d}_i(t) = \hat{d}_i$. By definition of a stationary point, all successive values of a variable are the same, e.g., in (2) for a non-source node:

$$\hat{d}_i = \min_{j \in \mathcal{N}(i)} \{\hat{d}_j + e_{ij}\} \quad (11)$$

As there may be multiple nodes that minimize (11), we shall say that any such is a valid c_i , a current constraining node of i . Further a node will be said to be attached to any source it is the nearest to, i.e., it can be attached to multiple sources.

Lemma 1. Suppose \mathcal{G} is connected, the source set at a stationary point obeys $|S| > 1$ and without loss of generality the highest priority node in S is 1. Then there is without loss of generality $2 \in S$, and a set of nodes $\{i_1, \dots, i_l, \dots, i_L\}$,

$i_1 = 1$, $i_L = 2$, and $1 \leq p \leq L$ such that the following hold. (i) For all l , such that $2 < l \leq p$, i_{l-1} , is a valid c_{i_l} . (ii) For all l , such that $p < l < L$, i_{l+1} is a valid c_{i_l} . (iii) For all l , such that $1 \leq l \leq q$, $\sigma_{i_l} = 1$ and for all $q + 1 \leq l \leq L$, $\sigma_{i_l} = L$. (iv) The distance estimates obey,

$$\hat{d}_{i_m} = \begin{cases} \sum_{j=1}^{m-1} e_{l_j, l_{j+1}} & \forall 2 \leq m \leq p \\ \sum_{j=m+1}^L e_{l_j, l_{j-1}} & \forall p+1 \leq m \leq L-1 \end{cases} \quad (12)$$

(v) The distance estimates in the S -block obey

$$\bar{d}_{i_m} = \begin{cases} \sum_{j=1}^{m-1} e_{l_j, l_{j+1}} & \forall 2 \leq m \leq q \\ \sum_{j=m+1}^L e_{l_j, l_{j-1}} & \forall q+1 \leq m \leq L-1 \end{cases} \quad (13)$$

(vi) The pseudodiameter estimates obey:

$$D_2 \geq \hat{d}_{i_p} + e_{i_p, i_{p+1}} \quad (14)$$

Proof. At a stationary point the ABF is also at a stationary point. As shown in [14], ABF's stationary point is unique and \hat{d}_i is the true distance of i from S . As S has at least two sources apart from 1, suppose 2 is the source nearest to 1. Now define the nodes on this shortest path from 1 to 2 to be i_l defined in the theorem statement. Suppose i_p is the last node attached to 1. First suppose i_{p+1} is attached to 2. Then for all $l \geq p$, i_l is attached to 2, as otherwise as each node is attached to some source, this source must be closer to 1 than is 2. If on the other hand if i_{p+1} is not attached to 2. then by the same argument 2 cannot be the closest source to 1. Then the recursion in (2) and the definition of valid c_i , proves, (i), (ii) and (12). Further (iii) follows as 2 is a valid source. Then the recursion in (10) proves (13), and that in (4) proves (14). ■

Lemma 1 leads to the following theorem:

Theorem 1. Suppose the graph \mathcal{G} is connected and $k \geq 2$. Then at a stationary point $|S| = 1$.

Proof. To establish a contradiction suppose $|S| > 1$. Then Lemma 1 holds. Consider the various quantities defined in Lemma 1. In particular (i), (ii) and (12) imply that nodes $\{i_1, \dots, i_p\}$ and $\{i_{p+1}, \dots, i_L\}$ are attached to 1 and 2, respectively. The fact that i_p is attached to 1 and i_{p+1} to 2 means that

$$\hat{d}_{i_{p+1}} + e_{i_p, i_{p+1}} \geq \hat{d}_{i_p}. \quad (15)$$

Also observe from (13) and (12) that

$$\bar{d}_{i_q} + e_{i_q, i_{q+1}} \leq \hat{d}_{i_p} + \hat{d}_{i_{p+1}} + e_{i_p, i_{p+1}}. \quad (16)$$

Further as 1 has higher priority than 2, $k \geq 2$, $\sigma_{i_q} = 1$ and $\sigma_{i_{q+1}} = 2$,

$$\bar{d}_{i_q} + e_{i_q, i_{q+1}} > 2D_1.$$

Then using (14), (15) and (16) we get a contradiction from:

$$\begin{aligned} 2 \left(\hat{d}_{i_{p+1}} + e_{i_p, i_{p+1}} \right) &\leq 2D_1 \\ &< \bar{d}_{i_q} + e_{i_q, i_{q+1}} \\ &\leq \hat{d}_{i_{p+1}} + \hat{d}_{i_p} + e_{i_p, i_{p+1}} \\ &\leq 2\hat{d}_{i_{p+1}} + 2e_{i_p, i_{p+1}}. \end{aligned}$$

■

B. The need for hysteresis

We conclude with yet another subtlety regarding the selection of k . As presented, the algorithm is fragile to mobility and loss of nodes. In particular suppose a node oscillates to move in and out of the network. Then the source set perpetually oscillates. Thus we need two parameters $k_L < k_A$, such that a leader is created if the node does not find a higher priority leader within k_A times its pseudodiameter estimate, and is suppressed if it finds a higher priority leader within k_L times its pseudodiameter estimate. This also has stability implications beyond resilience to mobility. As confirmed by simulations in Section IV, the choice of $k_L = k_A$ decreases convergence speed as leaders that are suppressed can quickly return while the pseudodiameter estimates change. Thus hysteresis is needed for stability for fast convergence even without mobility or other structural perturbations in the network. This calls for carrying two instead one set of valid neighbors like $V_i(t)$ and $M_i(t)$, one for source suppression the other for creation. Thus (7) should be replaced by the two sets

$$V_{iL}(t) = \{j \in \mathcal{N}(i) \mid \bar{d}_j(t) + e_{ij} \leq k_L D_i(t)\} \quad (17)$$

$$V_{iA}(t) = \{j \in \mathcal{N}(i) \mid \bar{d}_j(t) + e_{ij} \leq k_A D_i(t)\} \quad (18)$$

Likewise (8) must be replaced by

$$M_{iL}(t) = \{j \in V_{iL}(t) \mid \sigma_j(t) = \min_{l \in V_{iL}(t)} \sigma_l(t)\}, \quad (19)$$

$$M_{iA}(t) = \{j \in V_{iA}(t) \mid \sigma_j(t) = \min_{l \in V_{iA}(t)} \sigma_l(t)\}, \quad (20)$$

The equations (9) and (10) respectively become, for any $j \in M_{iL}(t)$ and $m \in M_{iA}(t)$, (21) and (22).

$$\sigma_i(t+1) = \begin{cases} \sigma_j(t) & V_{iL}(t) \neq \emptyset \text{ and } \sigma_j(t) < i \\ i & V_{iA}(t) = \emptyset \text{ or } \sigma_m(t) \geq i \\ \sigma_m(t) & \text{otherwise} \end{cases} \quad (21)$$

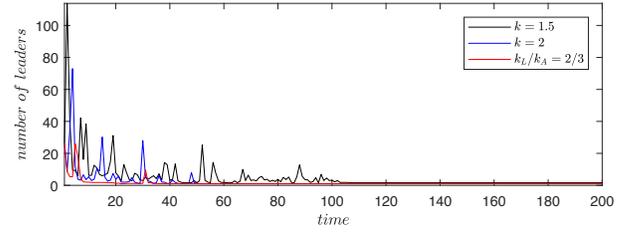
$$\bar{d}_i(t+1) = \begin{cases} \min_{l \in M_{iL}(t)} \{\bar{d}_l(t) + e_{il}\} & V_{iL}(t) \neq \emptyset \text{ and } \sigma_j(t) < i \\ 0 & V_{iA}(t) = \emptyset \text{ or } \sigma_m(t) \geq i \\ \min_{l \in M_{iA}(t)} \{\bar{d}_l(t) + e_{il}\} & \text{otherwise} \end{cases} \quad (22)$$

The analysis in Section III-A will carry through mutatis mutandis with k replaced by k_L . Our experiments indicate that $k_L = 2$ and $k_A = 3$ are reasonable values to trade off stability and speed.

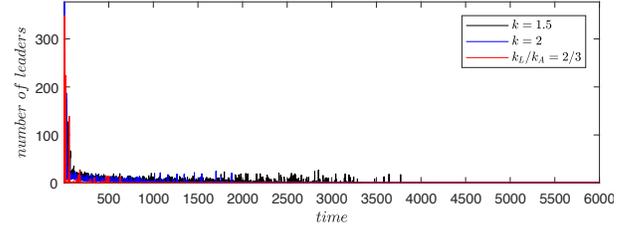
IV. SIMULATION

In this section, we empirically confirm the results presented in the prior sections through simulations for two graph types. We also include hysteresis, as described in Section III.

The first graph has 500 nodes randomly distributed in a 4×1 km² area, each communicating over a 0.25 km radius i.e., each has roughly 20 neighbors. The second, which we conjecture represents a worst case scenario, is a line graph with 500 nodes, priority randomly assigned, and an edge



(a) Average number of leaders vs time (randomized graph)



(b) Average number of leaders vs time (line graph)

Fig. 4. Comparison between (a) the average number of leaders using different k in the randomized graph, and (b) the average number of leaders using different k in the line graph.

length of 1 km. Initial conditions for each state are set as: $\hat{d}_i(0) = \bar{d}_i(0) = 0, r_i(0) = D_i(0) = 1, c_i(0) = \sigma_i(0) = i, \mathcal{C}_i(0) = \emptyset$ and $S(0) = \{1, 2, \dots, N\}$. All simulations are run 10 times with the same initial states.

We first run the algorithm with $k_A = k_L = k = 1.5$ and $k_A = k_L = k = 2$, and $k_A = 3$ and $k_L = 2$. In Figure 4(a), both $k = 2$ and the hysteresis method elect a single leader in the randomized graph, with hysteresis speeding convergence. With $k = 1.5$, the average number of leaders is 1.6. This confirms our observation in Section III that beyond instilling robustness to mobility, hysteresis increases the speed of convergence.

Compared to the randomized graph, the line graph performs much worse in leader suppression as shown in Figure 4(b). In this case, we run the algorithm for 6000 simulated seconds, finding that although convergence time is long, $k = 1.5$, $k = 2$, and the hysteresis method do all eventually converge to a single leader.

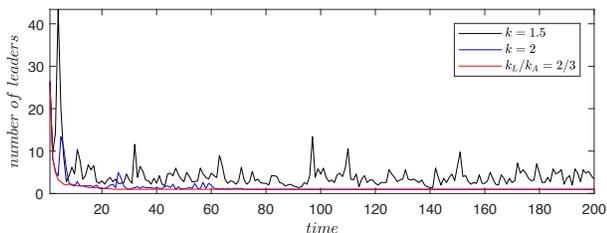
Our second simulation adds measurement errors: Edge lengths change from their nominal values as

$$\bar{e}_{ij}(t) = 1 + \epsilon_{ij}(t), \quad |\epsilon_{ij}(t)| < 1. \quad (23)$$

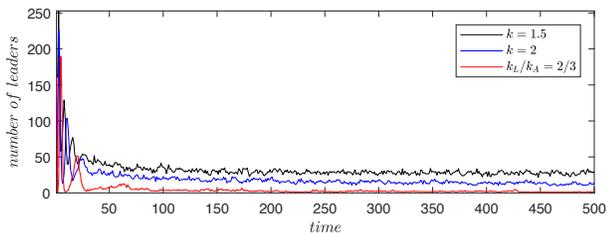
The bound ensures that no edge length is negative. Moreover, the noise is asymmetric, i.e., the following is allowed:

$$\bar{e}_{ij}(t) \neq \bar{e}_{ji}(t) \quad (24)$$

Figure 5 depicts the setting where measurement errors are sampled from a uniform distribution of $\mathcal{U}(-0.5, 0.5)$ in each round. In Figure 5(a), for randomized graph, the hysteresis method and $k = 2$ still successfully elect a single leader, while the average number of leaders under $k = 1.5$ keeps oscillating. As before, the line graph behaves the worst in leader suppression as shown in Figure 5(b).

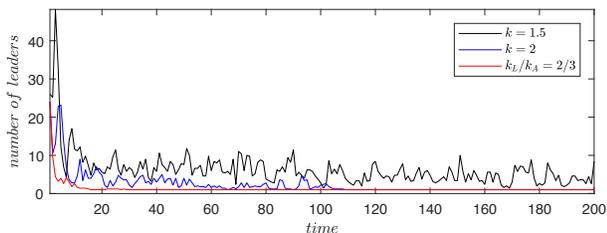


(a) Average number of leaders vs time(randomized graph)

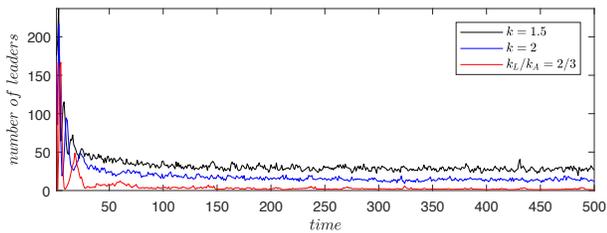


(b) Average number of leaders vs time(line graph)

Fig. 5. Comparison between (a) the average number of leaders using different k in the randomized graph under measurement error, and (b) the average number of leaders using different k in the line graph under measurement error.



(a) Average number of leaders vs time(randomized graph)



(b) Average number of leaders vs time(line graph)

Fig. 6. Comparison between (a) the average number of leaders using different k in the randomized graph under node mobility, and (b) the average number of leaders using different k in the line graph under node mobility.

Finally, Figure 6 depicts the effect of node mobility: at each t , a node is perturbed from its nominal location by $[r \cos \theta, r \sin \theta]^T$ with $r \sim \mathcal{U}(0, 0.5)$ and $\theta \sim \mathcal{U}(0, 2\pi)$ (though always remaining connected). The behavior is similar to the case of measurement error.

V. CONCLUSION

In this paper, we have presented a feedback combination of aggregate programming blocks to elect a leader among a network of devices in a distributed resilient manner. We have considered design issues associated with the key parameter k , showing that too large a value leads to delayed recovery,

while too small a value leads to multiple leaders. We have shown that as long as $k \geq 2$, the only stationary point of the algorithm comprises a solitary leader. Simulations confirm our theoretical predictions. Next steps for this work include an analytical stability analysis as done for G blocks in [12] and [14] and that of an open loop $G - C$ combination as done in [13], along with application to large-scale open systems.

ACKNOWLEDGMENT

This work has been supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0049. The views, opinions, and/or findings expressed 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. This document does not contain technology or technical data controlled under either U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations. Approved for public release, distribution unlimited (DARPA DISTAR case 29610, 6/13/2018).

REFERENCES

- [1] J. Beal, D. Pianini, and M. Viroli, "Aggregate programming for the Internet of Things," *IEEE Computer*, vol. 48, no. 9, pp. 22–30, 2015.
- [2] H. Zimmermann, "Osi reference model—the iso model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.
- [3] J. Beal and M. Viroli, "Building blocks for aggregate programming of self-organising applications," in 8th Int'l. Conf. on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2014, pp. 8–13.
- [4] M. Viroli and F. Damiani, "A calculus of self-stabilising computational fields," in 16th Int'l. Conf. on Coordination Models and Languages (COORDINATION), vol. 8459 of LNCS, pp. 163–178. Springer, 2014.
- [5] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann, San Francisco, USA, 1996.
- [6] S. Dolev, *Self-Stabilization*, MIT Press, 2000.
- [7] M. Schneider, "Self-stabilization," *ACM Computing Surveys*, vol. 25, pp. 45–67, 1993.
- [8] S. Dolev, A. Israeli, and S. Moran, "Uniform dynamic self-stabilizing leader election," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 4, pp. 424–440, 1997.
- [9] A. Derhab and N. Badache, "A self-stabilizing leader election algorithm in highly dynamic ad hoc mobile networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 926–939, 2008.
- [10] A. K. Datta, L. L. Larmore, and P. Vemula, "Self-stabilizing leader election in optimal space under an arbitrary scheduler," *Theoretical Computer Science*, vol. 412, no. 40, pp. 5541–5561, 2011.
- [11] K. Altisen, A. Cournier, S. Devismes, A. Durand, and F. Petit, "Self-stabilizing leader election in polynomial steps," *Information and Computation*, vol. 254, pp. 330–366, 2017.
- [12] S. Dasgupta and J. Beal, "A lyapunov analysis for the robust stability of an adaptive bellman-ford algorithm," in *Decision and Control (CDC)*, 2016 IEEE 55th Conference on. IEEE, 2016, pp. 7282–7287.
- [13] Y. Mo, J. Beal, and S. Dasgupta, "Error in self-stabilizing spanning-tree estimation of collective state," in *Int'l Workshops on Foundations and Applications of Self* Systems (FAS*W)*. IEEE, 2017, pp. 1–6.
- [14] Y. Mo, S. Dasgupta, and J. Beal, "Robustness of the adaptive bellman-ford algorithm: Global stability and ultimate bound," submitted to *IEEE Transactions on Automatic Control*, 2018.
- [15] A. Kumar, J. Beal, S. Dasgupta, and R. Mudumbai, "Toward predicting distributed systems dynamics," in *Int'l Conf. on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*. IEEE, 2015, pp. 68–73.
- [16] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.
- [17] L. R. Ford Jr, "Network flow theory," *Tech. Rep. Paper P-923*, RAND Corporation, 1956.