A resilient leader election algorithm using aggregate computing blocks

A Resilient Leader Election Algorithm Using Aggregate Computing Blocks ⋆

Yuanqiu Mo * **Giorgio Audrito** ** **Soura Dasgupta** *** **Jacob Beal** ****

* *Westlake Institute for Advanced Study, Westlake University, Hangzhou 310024, China (email: moyuanqiu@westlake.edu.cn)*
** *Department of Computer Science, University of Turin, Turin, Italy (e-mail: giorgio.audrito@gmail.com)*
*** *University of Iowa, Iowa City, Iowa 52242 USA and Shandong Academy of Science. (e-mail: soura-dasgupta@uiowa.edu).*
**** *Raytheon BBN Technologies Cambridge, MA, USA 02138 (e-mail: jakebeal@ieee.org)*

Abstract: Leader election, a fundamental coordination problem in distributed systems, has been addressed in many different ways. Among these works, resilient leader election algorithms are of particular interest due to the ongoing emergence of open, complex distributed systems such as smart cities and the Internet of Things. However, previous algorithms with $O(diameter)$ stabilization time complexity either assume some prior knowledge of the network or that very large messages can be sent. In this paper, we present a resilient leader election algorithm with $O(diameter)$ stabilization time, small messages, and no prior knowledge of the network. This algorithm is based on aggregate computing, which provides a layered approach to algorithm development based on composition of resilient algorithmic "building blocks." With our algorithm, a key design parameter K defines important performance attributes: a larger K will delay the recovery from loss of current leader, while a small K may lead to multiple leaders, and the algorithm will stabilize with $O(diameter)$ time complexity when $K \geq 2$.

Keywords: Leader election, multiagent system, resilience, aggregate computing.

1. INTRODUCTION

Recent intense activity in the control and stability of multiagent systems has included consensus, (e.g., Olfati-Saber et al. (2007)), distributed agreement, (e.g., Cao et al. (2008)) and formation control, (e.g., Baillieul and Suri (2003), Dasgupta et al. (2011), Summers et al. (2009), Fidan et al. (2013) and Summers et al. (2011)). All are distributed graph algorithms with local control action guided by limited information flow between neighbors. Most, though not all, represent nonlinear systems. In this tradition, this paper presents and analyzes a leader election algorithm that involves the feedback composition of two nonlinear systems, each of which is a distributed graph algorithm.

Leader election is a fundamental problem in distributed systems, where a network elects a single node as a leader in a distributed, and in our case, resilient fashion. Resilience implies the eventual election of a *single* leader and the ability to recover from transient perturbations like the disappearance of leaders, temporary emergence of false leaders and link failures that do not disconnect the graph.

The study of leader election algorithms (e.g., Gallager (1977) and Le Lann (1977)), has considered time, space and message complexity of *deterministic* leader election on general networks with identifiers, that of *probabilistic* leader election on anonymous networks, of leader election on specific networks like ring and complete graphs, and of leader election in asynchronous graphs.

This paper considers election in open, complex distributed systems such as smart cities and the Internet of Things, that have many devices and intermittent perturbation of both network membership and topology. We assume that each node carries a unique identifier of size $O(\log N)$ bits with N the number of nodes and that each node exchanges messages with its neighbors in synchronous rounds (a simplifying assumption for analytical purposes; our results will also hold for fair asynchronous execution). We measure time, space, and message complexities by communication rounds, bits, and multiples of $\log N$ bits respectively.

In general synchronous networks, the global lower bound on the time required for a leader election algorithm is

⋆ 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 32200, 10/31/19). Mo was also partially supported by the Australian Research Council under grant DP190100887 and DP160104500.

$\Omega(\mathcal{D})$ [1] rounds (Kutten et al. (2015)), where $\mathcal{D}$ is the diameter of the network. This is achieved by Peleg (1990), which elects a leader via the construction of a *breadth first search* tree (BFS) rooted at the leader. Its time and message complexities are $O(\mathcal{D})$ and $O(\mathcal{DE})$ respectively, where $\mathcal{E}$ is the number of edges. Closely competitive is Casteigts et al. (2016), which elects a leader in $O(\mathcal{D} + \log N)$ rounds but sends messages of size $O(1)$ instead of $O(\log N)$ bits. Both assume a *perturbation free graph.*

Some time optimal (i.e. with $O(\mathcal{D})$ complexity) solutions consider perturbations but also assume such prior knowledge of the graph as the number of nodes N or the diameter $\mathcal{D}$. In Awerbuch et al. (1993), the authors implement the Bellman-Ford algorithm with IDs requiring that nodes know an upper bound on the diameter of the network. Similarly, Burman and Kutten (2007) assumes that nodes know an upper bound on $\mathcal{D}$. The space complexity of both solutions is $O(\log N \log \mathcal{D})$, as they require $\Theta(\log N \log \mathcal{D})$ bits per node. These assumptions do not hold in open networks where nodes are lost or the network grows.

Algorithms in Aggarwal and Kutten (1993) and Kravchik and Kutten (2013) accommodate perturbations without such knowledge. In the former, an *Extend-ID* mechanism is used to eliminate a corrupted leader, but their messages may increase to unbounded length. The latter implicitly uses the network size by utilizing a synchronizer requiring knowledge of an upper bound on N. Solutions in Datta et al. (2011a,b) and Altisen et al. (2017) assume a distributed unfair daemon and have space complexity of $O(\log N)$ bits per node, but stabilize in a far from optimal $O(N)$ as opposed to the best achievable $O(\mathcal{D})$ rounds.

We take an approach based on *aggregate computing* (Beal et al. (2015)), which offers a layered abstraction approach to simplifying the design, creation, and maintenance of complex open distributed systems, much like the OSI model for communication. Key for this paper is its middle layer, comprising a collection of three general "building-block" operators for resilient coordination applications: (i) G-blocks that spread information through a network of devices, (ii) C-blocks that summarize information about the network to be used by interacting units, and (iii) T-blocks that maintain temporary state. Their compositions realize a broad class of dispersed services, as described in Beal and Viroli (2014) and Viroli and Damiani (2014).

We introduce a globally, uniformly, asymptotically stable (GUAS) leader election algorithm via a feedback interconnection of *aggregate computing* building blocks. Our algorithm is time optimal with a stabilization time of $O(\mathcal{D})$, space optimal at $O(\log N)$ bits per node (given that every node needs to store its own ID using $O(\log N)$ bits), and has a message complexity of $O(\mathcal{DE})$. While running a slightly different Bellman-Ford algorithm than Awerbuch et al. (1993) with IDs, our algorithm also simultaneously estimates the diameter of the network, and *thus does not need any prior knowledge of the network.* Like most feedback systems, it has one free design parameter K that determines important performance attributes. A large value accelerates convergence but impairs resilience by delaying recovery from loss of the current leader. A small value improves resilience but slows convergence or can elect multiple leaders. Trade-off between resilience and convergence speed is common in most algorithms.

Section 2 provides preliminaries. Section 3, gives a formal description of our leader election algorithm. Section 4 demonstrates its resilience by upper bounding the time to recovery from transient perturbations and proves GUAS. Section 5 validates our results through simulations and shows that our algorithm compares favorably to Datta et al. (2011b). In particular, it recovers much faster from perturbations caused by a leader loss or the temporary advent of a fake leader. Section 6 concludes. Proofs are omitted due to space constraints.

2. PRELIMINARIES

We consider an undirected graph $G = (V, E)$, with $V = \{1, 2, ..., N\}$ the set of nodes (devices), and E the set of edges. Without loss of generality, we assume the index of a node represents its ID and also reflects its priority, i.e., node i has a higher priority than node $i+1$. The algorithm must elect the node $i = 1$ (unless it is lost) to be the single leader in the graph. An edge indicates the existence of a communication link between nodes and i is a neighbor of j if there is an edge between i and j, defining $\mathcal{N}(i)$ as the set of all neighbors of node i. All edge lengths are 1, i.e., distances are hop counts.

The shortest distance d_i of i from 1, the desired leader, obeys the following recursion from Mo et al. (2019):

$$d_i = \begin{cases} 0, & i = 1 \\ \min\limits_{k \in \mathcal{N}(i)} \{d_k + 1\}, & i \neq 1 \end{cases} \tag{1}$$

Based on (1), we introduce the following definition.

Definition 1. *A k that minimizes the right hand side of (1) is a* true constraining node *of i. As there may be many neighbors k and l of i such that $d_l = d_k$, a node may have multiple true constraining nodes. The set of true constraining nodes of i is defined as $\mathcal{C}(i)$. Moreover, the true constraining node of 1 is itself.*

Further, we also make the following related definition:

Definition 2. *The* effective diameter $\mathcal{D}(G)$ *of G is:*

$$\mathcal{D}(G) = \max\{d_i \mid i \in V\}.$$

Note that $\mathcal{D}(G)$ may be smaller than the diameter of G.

3. ALGORITHM

The leader election algorithm, in Figure 1, uses a feedback interconnection comprising two aggregate computing building blocks, each of which is a distributed algorithm.

Before providing details of this algorithm, we summarize the basic approach. A node i is *attached* to the leader σ_i estimated as being nearest to it (a leader is attached to itself). The distance estimates $\hat{d}_i$ are obtained using an algorithm similar to the adaptive Bellman-Ford algorithm analyzed in Mo et al. (2019). Each node i carries a diameter estimate D_i of the sub-network of the nodes attached to the same leader as itself, which the leader converts (through a design parameter K) into a *propagation radius* R_i that bounds its influence area.

[1] A quantity X is $\Omega(D)$ if there exist positive constants k_1 and D_0 such that $X \geq k_1 D$ for $D \geq D_0$. It is $\Theta(D)$ if there is an additional constant k_2 such that it is between $k_1 D \leq X \leq k_2 D$ for $D \geq D_0$.

A node accepts the leader of a neighbor as its own only if the resulting distance estimate is smaller than the propagation radius. Among acceptable leaders, the one with higher priority (lower i) is chosen. A leader i relinquishes its status if a neighbor j is attached to a higher priority leader within R_j hops from i. If none exists, the node assumes or retains the role of a leader.

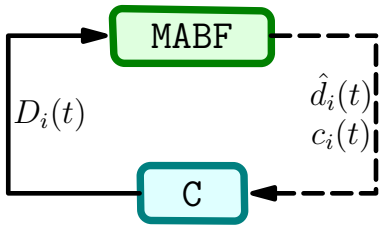

Fig. 1. Block diagram of our leader election algorithm. Block C computes values $D_i(t)$, feeding them directly to block MABF. This block in turn produces values $\hat{d}_i(t)$ and $c_i(t)$, which are fed to C in the following round.

This diameter-based strategy provides resilience. When a leader is lost, all its neighbors are thereafter constrained by others. As shown in Lemma 3, their distance estimates $\hat{d}_i$ keep increasing while the propagation radii R_i remain unchanged. Eventually, as quantified in Lemma 4, they will not find a neighbor j within R_j hops from the leader it is attached to. New leaders will propose themselves, and the highest priority one will become the new leader.

Figure 1 describes the structure of the algorithm. The *modified adaptive Bellman-Ford* (MABF) block is a G-block that *broadcasts* radius bounds R_i to nodes attached to a leader, determines current leaders σ_i, and obtains distance estimates $\hat{d}_i$ of nodes attached to it. The C block maximizes over the $\hat{d}_i$ to obtain diameter estimates D_i, which are then used by the MABF block to produce the propagation bounds R_i. The dotted line from MABF to C indicates the latter works on MABF output with a delay of one cycle. Thus although there is no delay in the C to MABF connection, the closed loop *avoids a delay free loop.*

3.1 The MABF block

As introduced in Section 2, the node ID, represented by node index, is unique and cannot be falsified. Thus, we define $\sigma_i(t)$ as the estimated leader ID of node i at time t, which may be corrupted under perturbations. The purpose of leader election is to elect the highest priority node as the unique leader, i.e., $\sigma_i(t) = 1$ for all $i \in V$ and $t \geq T$ with some finite T. We say that node i is a current leader if $\sigma_i(t) = i$, and that i is *attached* to a leader j if $\sigma_i(t) = j$.

MABF decides whether a node i should be a leader, by selecting leader estimates from neighbors $j \in \mathcal{N}(i)$ that *(i)* carry distance estimate (hop count) $\hat{d}_j$ that is smaller than the propagation radius $\hat{d}_j < R_j$, and *(ii)* their leader σ_j has a higher priority than i. A node becomes a leader whenever no such neighbor exists. More precisely, define $V_i(t+1)$ comprising the neighbors of i whose distance estimates are within their propagation radius:

$$V_i(t+1) = \{j \in \mathcal{N}(i) \mid \hat{d}_j(t) < R_j(t)\} \tag{2}$$

Then define $\bar{V}_i(t+1) \subset V_i(t+1)$ comprising members j of $V_i(t+1)$ that carry the smallest leader estimate $\sigma_j(t) < i$:

$$\bar{V}_i(t+1) = \{j \in V_i(t+1) \mid \sigma_j(t) = k \text{ and } \sigma_j(t) < i\} \tag{3}$$

where $k = \min_{\ell \in V_i(t+1)} \{\sigma_\ell(t)\}$. *If $\bar{V}_i(t+1)$ is empty, then i becomes a leader,* with a distance estimate of 0, and a propagation radius calculated by the C block. Thus,

$$\sigma_i(t+1) = i, \quad \hat{d}_i(t+1) = 0, \quad R_i(t+1) = g(D_i(t+1)) \tag{4}$$

assuming that $g(D_i(t+1)) = KD_i(t+1)+1$ where $K \geq 1$ is a design parameter.

If $\bar{V}_i(t+1)$ is nonempty, i cannot be a leader, and will *attach itself to the leader one of the members of $\bar{V}_i(t+1)$ is attached to.* This neighbor also becomes the *current constraining* node $c_i(t+1)$ of i, used to update the MABF variables $\sigma_i(t), \hat{d}_i(t), R_i(t)$. If i becomes a leader, we say that its current constraining node is itself $c_i(t+1) = i$.

The *current constraining node* is a member of $\bar{V}_i(t+1)$ with *(i)* the smallest distance estimate $\hat{d}_j(t)$; *(ii)* if tied, the largest propagation radius $R_j(t)$; *(iii)* if still tied, the smallest j. *Because of the definition of $\bar{V}_i$, the selected j is the node in V_i that is attached to the highest priority leader.* Thus we define $\hat{V}_i(t+1) \subset \bar{V}_i(t+1)$ as set of neighbors with the smallest distance estimate:

$$\hat{V}_i(t+1) = \{j \in \bar{V}_i(t+1) \mid \hat{d}_j(t) = \min_{\ell \in \bar{V}_i(t+1)} \{\hat{d}_\ell(t)\}\} \tag{5}$$

and $\tilde{V}_i(t+1)$ as the subset of $\hat{V}_i(t+1)$ of neighbors carrying the largest propagation radius:

$$\tilde{V}_i(t+1) = \{j \in \hat{V}_i(t+1) \mid D_j(t) = \max_{\ell \in \hat{V}_i(t+1)} \{D_\ell(t)\}\}. \tag{6}$$

Finally, we set the $c_i(t+1)$ to $\min \tilde{V}_i(t+1)$. Each non-leader node then updates its leader, distance, and propagation radius estimates as below with $j = c_i(t+1)$:

$$\sigma_i(t+1) = \sigma_j(t), \quad \hat{d}_i(t+1) = \hat{d}_j(t)+1, \quad R_i(t+1) = R_j(t). \tag{7}$$

This also explains the strict inequality in (2). It forces the distance estimate of node i to obey $\hat{d}_i = \hat{d}_j + 1 \leq R_j$. It leads to the following lemma that is key to resiliency.

Lemma 1. *Under (2)- (7), there holds:*

$$\hat{d}_i(t) \leq R_i(t), \quad \forall i \in V \text{ and } \forall t \geq 1. \tag{8}$$

The set of *nodes constrained by* i at time t is:

$$\mathcal{C}_i(t) = \{j \in V \setminus \{i\} \mid c_j(t) = i\}, \tag{9}$$

and the set of leaders at time t is defined as:

$$S(t) = \{i \mid \sigma_i(t) = i\}. \tag{10}$$

3.2 The C block

The C block, specialized from Viroli et al. (2018), collects and sends to each leader its current estimated diameter D_i, by maximizing among the estimated distances $\hat{d}_i(t)$ and diameter estimates at nodes that i constrains:

$$D_i(t+1) = \max\{\hat{d}_i(t), \{D_j(t) \mid j \in \mathcal{C}_i(t)\}\} \tag{11}$$

with $\mathcal{C}_i(t)$ defined in (9).

3.3 Desired stationary point

Evidently the desired stationary state has the following components. Every node accepts node 1 as the leader:

$\sigma_i(t) = 1,\ \forall\ i \in V$. Consequently, $\hat{d}_i(t) = d_i$ and $D_i(t) = \mathcal{D}(G)\ \forall\ i \in V$.

4. STABILITY AND RESILIENCE ANALYSIS

We now prove the GUAS of the algorithm, while bounding the time required for its convergence. The following constitutes our standing assumption.

Assumption 1. *The undirected graph $G = (V, E)$ is connected, $t_0 = 0$ is the initial time, d_i, $\mathcal{N}(i)$, $c_i(t)$, $\mathcal{C}_i(t)$, $\sigma_i(t)$, $\hat{d}_i(t)$, $R_i(t)$, $D_i(t)$, $g(D) = KD + 1$ with $K \geq 1$ are as in Section 3, and for all $i \in V$, the quantities $\hat{d}_i(0)$, $R_i(0)$, $D_i(0)$ are non-negative integers.*

4.1 Resilience to fake or lost leaders

We demonstrate the *resilience of the algorithm to the fleeting advent of fake leaders or the loss of a legitimate leader*. We assume that the loss or advent occurs before $t = 0$ and that at $t = 0$, even though no fake nodes exist, extant nodes may be attached to a lost or past fake leader. Thus we allow $\sigma_i(0) \notin V$, i.e., $\sigma_i(0)$ may not be a positive integer. We define the set of *unrooted* nodes and prove that only unrooted nodes can carry fake leader IDs.

Definition 3 (Unrooted Node Set)**.** *Define the set of* unrooted nodes $\mathcal{U}(t)$ *as* $\mathcal{U}(0) = V$, *and*

$$\mathcal{U}(t+1) = \{i \in V \mid i \neq c_i(t+1) \in \mathcal{U}(t)\}.$$

Thus $\mathcal{U}(t+1)$ comprises non-leader nodes constrained *by members of $\mathcal{U}(t)$. Let $\mathcal{L}(t) = \{\sigma_i(t) \mid i \in \mathcal{U}(t)\}$ be the set of unrooted leaders, and define $\mathcal{U}^k(t)$ to be the set of unrooted nodes with leader k:*

$$\mathcal{U}^k(t) = \{i \in \mathcal{U}(t) \mid \sigma_i(t) = k\}.$$

Furthermore, define

$$\hat{d}^k_{\min}(t) = \min\{\hat{d}_i(t) \mid i \in \mathcal{U}^k(t)\},$$
$$R^k_{\max}(t) = \max\{R_i(t) \mid i \in \mathcal{U}^k(t)\}.$$

We first show that the set of unrooted leaders cannot grow and more importantly *only unrooted nodes can carry fake IDs, as IDs carried by rooted nodes are in V.*

Lemma 2 (Unrooted Leader Set Decay)**.** *Consider (2-11) under Assumption 1 and Definition 3. The set of unrooted leaders cannot expand over time, i.e., obeys $\mathcal{L}(t+1) \subseteq \mathcal{L}(t)$. Furthermore, leaders $\sigma_i(t)$ of rooted nodes $i \in V \setminus \mathcal{U}(t)$ are in V.*

As the intuition give in Section 3 states, nodes attached to lost leaders attach to legitimate ones as their distance estimates grow, while their estimated R_i do not. The next lemma shows that this occurs for all unrooted nodes.

Lemma 3. *Consider (2-11) under Assumption 1 and Definition 3. Consider $t \geq 0$ and $k \in \mathcal{L}(t)$. Then, $\hat{d}^k_{\min}(t) \geq \hat{d}^k_{\min}(0) + t$ and $R^k_{\max}(t) \leq R^k_{\max}(0)$.*

Thus eventually unrooted nodes violate (8) and attach to a node in V. The next lemma bounds the time it takes for this to happen. Given (2), this recovery takes longer for larger K, though as later shown smaller K slows subsequent convergence.

Lemma 4 (Fake Leader IDs Disappear)**.** *Consider (2-11) under Assumption 1 and Definition 3. Let:*

$$\hat{T} = 1 + \max\{0, R^k_{\max}(0) - \hat{d}^k_{\min}(0) \mid k \in \mathcal{L}(0) \setminus V\}.$$

Then $\sigma_i(t) \in V$ for every node $i \in V$ and $t \geq \hat{T}$.

Thus fake leader IDS disappear after time $\hat{T}$.

4.2 GLOBAL UNIFORM ASYMPTOTIC STABILITY

We now show that a node attached to the desired leader must eventually have an overestimated distance estimate.

Lemma 5 (Underestimates Decay)**.** *Consider (2-11) under Assumption 1. For every $i \in V$ and $t \geq 0$ such that $\sigma_i(t) = 1$, we have $\hat{d}_i(t) \geq \min(d_i, t)$.*

Hereafter correct estimates flow outwards from the desired leader 1 bounded by the propagation radius computed from leader diameter estimates (Lemma 7), while these diameter estimates flow inwards bounded by the stabilization of distances (Lemma 6).

More precisely, suppose at time T_x the distance estimates and the leader ids of all nodes i within x hops of 1 have converged. Then after time $T_x + x + 1$ the diameter estimate collected at 1 cannot be smaller than x.

Lemma 6 (Diameter Collection)**.** *Consider (2-11) under Assumption 1. Assume that $T_x \geq x$ for $x \leq \mathcal{D}(G)$ is such that every device $i \in V$ with $d_i \leq x$ stabilizes to $\sigma_i(t) = 1$, $\hat{d}_i(t) = d_i$ for $t \geq T_x$. Then $D_1(t) \geq x$ for $t \geq T_x + x + 1$.*

We now have the final lemma described before Lemma 6.

Lemma 7 (Diameter Broadcast)**.** *Consider (2-11) under Assumption 1 and Definition 1. Assume that $T_x \geq \hat{T}$ defined in Lemma 4 is such that $\sigma_1(t) = 1$, $\hat{d}_1(t) = 0$ and $D_1(t) \geq x$ for $t \geq T_x$. Then every $i \in V$ with $d_i \leq g(x)$ stabilizes to $\sigma_i(t) = 1$, $\hat{d}_i(t) = d_i$ and $R_i(t) \geq g(x)$ for $t \geq T_x + d_i$.*

These last two lemmas recursively characterize the convergence time T_x for nodes at distance x (Theorem 1).

Definition 4 (Discrete Inverse)**.** *We define $g^{-1}(D) = \left\lceil \frac{D-1}{K} \right\rceil$, which is the smallest number x such that $g(x) = Kx + 1 \geq D$.*

Theorem 1 (Convergence)**.** *Consider (2-11) under Assumption 1. Let T_x for $x \leq \mathcal{D}(G)$ be recursively defined as $T_0 = \hat{T}$ with $\hat{T}$ defined in Lemma 4,*

$$T_x = T_{g^{-1}(x)} + g^{-1}(x) + x + 1.$$

Then for $t \geq T_x$ and $i \in V$ such that $d_i \leq x$, we have that $\sigma_i(t) = 1$, $\hat{d}_i(t) = d_i$.

Notice that the time to convergence derived above is independent of the initial time $t_0 = 0$, and thus the global convergence is uniform. Furthermore, the bound above is strict e.g., whenever $\sigma_i(0) > 1$ for all $i \in V \setminus \{1\}$, and $r_1(0) = D_1(0) = 0$. Define, L to be the smallest integer for which $g^{-L}(x) = 0$. Then we get the following close form expressions verifiable using induction:

$$T_x = \hat{T} + x + \sum_{k=1}^{L} \left(2 \left\lceil \frac{x - \sum_{i=0}^{i<k} K^i}{K^k} \right\rceil + 1 \right) \tag{12}$$

For $K = 1$ $\left\lceil \frac{x - \sum_{i=0}^{i<k} K^i}{K^k} \right\rceil = x - k$ for $x \geq k$ and thus:

$$T_x = \hat{T} + x + \sum_{k=1}^{x} (2(x-k) + 1) = \hat{T} + x(x+1)$$

which is quadratic in x. However, if $K \geq 2$, we have that $L = \lceil \log_K(x(K-1)+1) \rceil$ and:

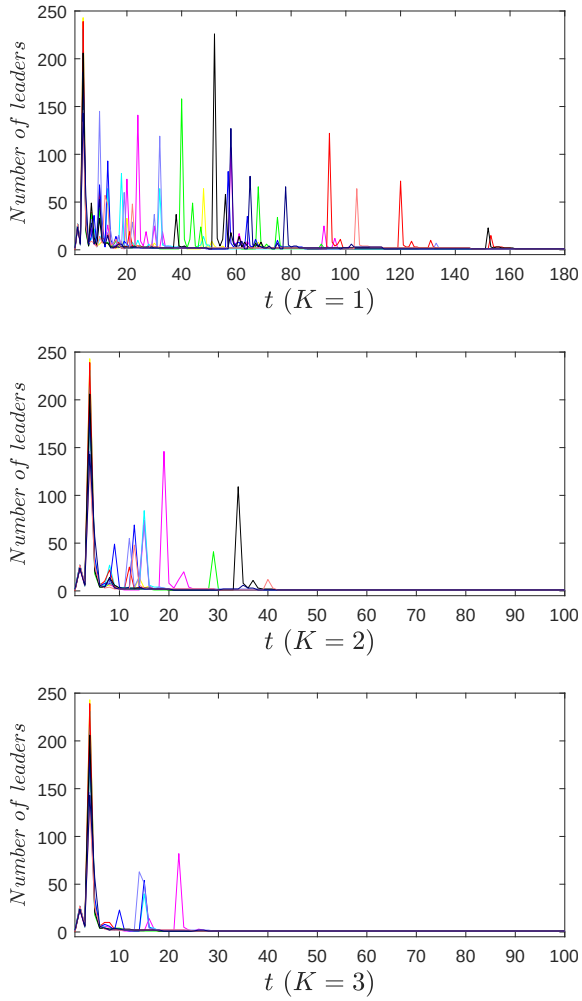

Fig. 2. Leader election with different K in a randomized graph without perturbations.

$$T_x = \hat{T} + \left(1 + \frac{2}{K-1}\right) x + \left(3 - \frac{2}{K-1}\right) L + \frac{2}{(K-1)^2}$$

which is asymptotically linear in x (since L is logarithmic in x): e.g., $T_x \leq \hat{T}+3x+\log_2(x+1)+3$ for $K = 2$. Assuming the initial values $R_i(0)$ to be not greater than the final values $R_i(t) = g(\mathcal{D}(G)) = K\mathcal{D}(G) + 1$ reached after stabilisation (that is, the perturbation did not decrease the graph diameter), we furthermore get that $T_x \leq 5x + \log_2(x+1) + 5$ for $K = 2$, $x = \mathcal{D}(G)$.

5. SIMULATIONS

We first investigate behavior without perturbations comparing performance with Datta et al. (2011b)—the only other small message (i.e., where an ID using $\log_2(N)$-bits is passed) knowledge free leader election algorithm. Five hundred nodes (ids from 1 to 500) are randomly distributed in a 4×1 field, communicating synchronously with a 0.25 unit disc. The initial conditions are: $\forall i \in V$, $\hat{d}_i(0) = R_i(0) = D_i(0) = 0$, and $\sigma_i(0)$ is a random integer between 1 to 500, with 10 runs per initial condition.

Figure 2 depicts results with $K = 1$, $K = 2$, and $K = 3$; $\mathcal{D}(G)$ ranges from 11 to 19 for these 10 trials. With $K = 1$ convergence is much slower (average 115 rounds)

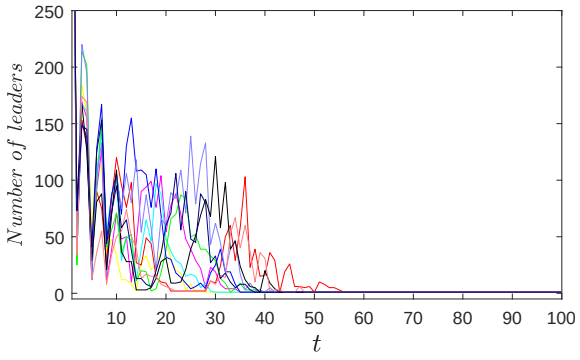

Fig. 3. Leader election by Datta in a randomized graph without perturbations.

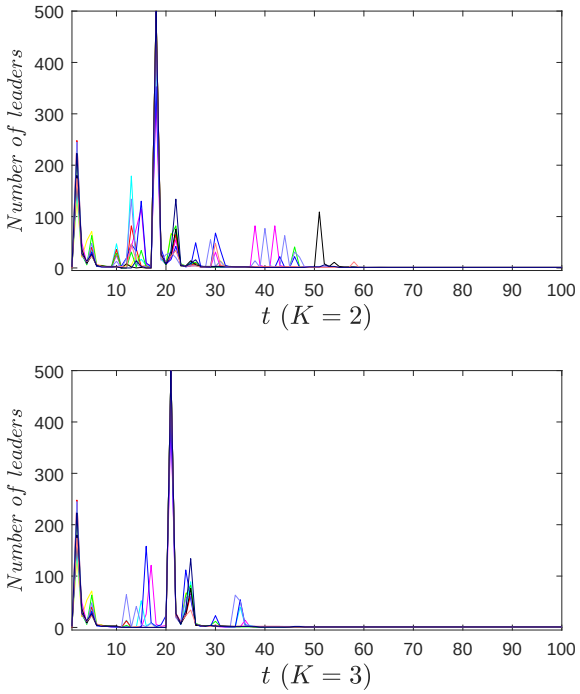

Fig. 4. Leader election using different K in a randomized graph under perturbations.

compared with $K = 2$ and $K = 3$. This is consistent with our result that in this case the time complexity is $O(\mathcal{D}(G)^2)$. With $K = 2$ or $K = 3$, our algorithm has a much better performance, having an average time to convergence of 36 rounds and 28 rounds respectively. Figure 3 depicts the performance of Datta's algorithm. The average time to convergence in this case is 41 rounds, which is worse than our algorithm with $K = 2$ or 3. *Moreover, Datta's algorithm has frequent oscillations with many leaders, whereas in most times ours has very few leaders.*

We next compare resilience to perturbations, otherwise keeping the same setup as above but omitting the $K = 1$ case. For $i = 50$ to 100, $\sigma_i(10) = 0.5$, i.e., 51 nodes are corrupted by a fake leader ID with the highest priority during run time.

As shown in Figure 4 with $K = 2$, the states of nodes will recover and converge in an average of 52 rounds across the 10 trials, and with $K = 3$ an average of 40 rounds. The algorithm of Datta et al. (2011b) in Figure 5, takes

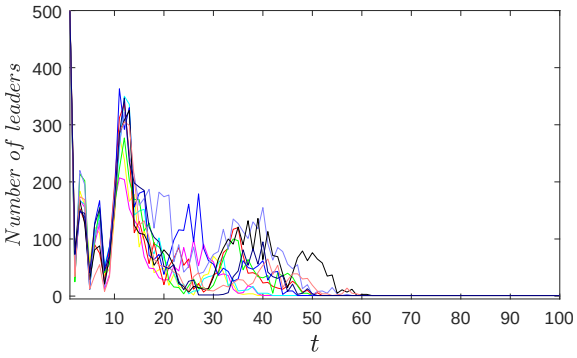

Fig. 5. Leader election by Datta in a randomized graph under perturbations.

an average of 53 rounds: essentially the same as $K = 2$, but slower than $K = 3$, *and again with many more leaders than even our* $K = 2$ *case. If these leaders take decisions in real time then this can cause greater network disruption.*

6. CONCLUSION

In this paper, we have introduced a resilient leader election algorithm implemented using a feedback interconnection of aggregate computing building blocks. This algorithm assumes no prior knowledge about the network, and is not only globally uniformly and asymptotically stable, but also resilient to transient perturbations. The free design parameter K can tune the convergence rate and resilience rate, with the algorithm achieving a time complexity of $O(\mathcal{D})$ when $K \geq 2$. Future work includes analysis of the algorithm under more severe perturbations, optimization of K, application to various distributed systems that make use of leader election, and analysis of its compositional properties within such systems.

REFERENCES

Aggarwal, S. and Kutten, S. (1993). Time optimal self-stabilizing spanning tree algorithms. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, 400–410. Springer.

Altisen, K., Cournier, A., Devismes, S., Durand, A., and Petit, F. (2017). Self-stabilizing leader election in polynomial steps. *Information and Computation*, 254, 330–366.

Awerbuch, B., Kutten, S., Mansour, Y., Patt-Shamir, B., and Varghese, G. (1993). Time optimal self-stabilizing synchronization. In *STOC*, volume 93, 652–661.

Baillieul, J. and Suri, A. (2003). Information patterns and hedging brockett's theorem in controlling vehicle formations. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, 556–563. IEEE.

Beal, J., Pianini, D., and Viroli, M. (2015). Aggregate programming for the internet of things. *Computer*, 48(9), 22–30.

Beal, J. and Viroli, M. (2014). Building blocks for aggregate programming of self-organising applications. In *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, 8–13.

Burman, J. and Kutten, S. (2007). Time optimal asynchronous self-stabilizing spanning tree. In *International Symposium on Distributed Computing*, 92–107.

Cao, M., Morse, A.S., and Anderson, B.D.O. (2008). Agreeing asynchronously. *IEEE Transactions on Automatic Control*, 53(8), 1826–1838.

Casteigts, A., Métivier, Y., Robson, J.M., and Zemmari, A. (2016). Deterministic leader election in $O(D + \log n)$ time with messages of size $O(1)$. In *International Symposium on Distributed Computing*, 16–28.

Dasgupta, S., Anderson, B.D.O., Yu, C., and Summers, T.H. (2011). Controlling rectangular formations. In *2011 Australian Control Conference*, 44–49. IEEE.

Datta, A.K., Larmore, L.L., and Vemula, P. (2011a). An o (n)-time self-stabilizing leader election algorithm. *Journal of Parallel and Distributed Computing*, 71(11), 1532–1544.

Datta, A.K., Larmore, L.L., and Vemula, P. (2011b). Self-stabilizing leader election in optimal space under an arbitrary scheduler. *Theoretical Computer Science*, 412(40), 5541–5561.

Fidan, B., Dasgupta, S., and Anderson, B.D.O. (2013). Adaptive range-measurement-based target pursuit. *International Journal of Adaptive Control and Signal Processing*, 27(1-2), 66–81.

Gallager, R.G. (1977). Finding a leader in network with $O(E) + O(NlogN)$ messages. Unpublished note, MIT.

Kravchik, A. and Kutten, S. (2013). Time optimal synchronous self stabilizing spanning tree. In *International Symposium on Distributed Computing*, 91–105.

Kutten, S., Pandurangan, G., Peleg, D., Robinson, P., and Trehan, A. (2015). On the complexity of universal leader election. *Journal of the ACM (JACM)*, 62(1), 7.

Le Lann, G. (1977). Distributed systems-towards a formal approach. In *IFIP congress*, volume 7, 155–160. Toronto.

Mo, Y., Dasgupta, S., and Beal, J. (2019). Robustness of the adaptive bellman-ford algorithm: Global stability and ultimate bounds. *IEEE Transactions on Automatic Control*, 4121– 4136.

Olfati-Saber, R., Fax, J.A., and Murray, R.M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1), 215–233.

Peleg, D. (1990). Time-optimal leader election in general networks. *Journal of parallel and distributed computing*, 8(1), 96–99.

Summers, T.H., Yu, C., Anderson, B.D.O., and Dasgupta, S. (2009). Formation shape control: Global asymptotic stability of a four-agent formation. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 3002–3007. IEEE.

Summers, T.H., Yu, C., Dasgupta, S., and Anderson, B.D.O. (2011). Control of minimally persistent leader-remote-follower and coleader formations in the plane. *IEEE Transactions on Automatic Control*, 56(12), 2778–2792.

Viroli, M., Audrito, G., Beal, J., Damiani, F., and Pianini, D. (2018). Engineering resilient collective adaptive systems by self-stabilisation. *ACM Transactions on Modelling and Computer Simulation (TOMACS)*, 28(2), 16:1–16:28.

Viroli, M. and Damiani, F. (2014). A calculus of self-stabilising computational fields. In *International Conference on Coordination Languages and Models*, 163–178.