# An Early-stopping Protocol for Computing Aggregate Functions in Sensor Networks

Antonio Fernández Anta, Miguel Mosteiro, Christopher Thraves-Caro

# An Early-stopping Protocol for Computing Aggregate Functions in Sensor Networks

Antonio Fernández Anta
*LADyR, GSyC*
*Universidad Rey Juan Carlos*
*28933 Móstoles, Madrid, Spain*
*anto@gsyc.es*

Miguel A. Mosteiro
*Computer Science Dept., Rutgers University*
*Piscataway, NJ 08854, USA*
*mosteiro@cs.rutgers.edu*
*LADyR, GSyC, Univ. Rey Juan Carlos*
*28933 Móstoles, Madrid, Spain*
*miguel.mosteiro@urjc.es*

Christopher Thraves
*ASAP Project team*
*IRISA/INRIA Rennes*
*Campus Univ. de Beaulieu*
*35043 Rennes Cedex, France*
*christopher.thraves-caro@irisa.fr*

***Abstract*—In this paper, we study algebraic aggregate computations in Sensor Networks. The main contribution is the presentation of an early-stopping protocol that computes the average function under a harsh model of the conditions under which sensor nodes operate. This protocol is shown to be time-optimal in presence of unfrequent failures. The approach followed saves time and energy by relying the computation on a small network of *delegate* nodes that can be rebuilt fast in case of node failures and communicate using a collision-free schedule. Delegate nodes run simultaneously two protocols, namely, a collection/dissemination tree-based algorithm, which is shown to be optimal, and a mass-distribution algorithm. Both algorithms are analyzed under a model where the frequency of failures is a parameter. Other aggregate computation algorithms can be easily derived from this protocol. To the best of our knowledge, this is the first optimal early-stopping algorithm for aggregate computations in Sensor Networks.**

***Keywords*-Sensor networks, Aggregate computation, Early-stopping algorithm, Failure model, Average computing.**

## I. Introduction

A *Sensor Network* is a simplified abstraction of a large monitoring infrastructure, formed of *sensor nodes* (or sensors) that create a radio communication network from scratch. Each sensor node is equipped with communication, processing, and sensing capabilities. Nodes can collaborate to process the sensed data but, due to unreliability, a monitoring strategy can not rely on individual sensors data. Instead, the network should use aggregated information from groups of sensor nodes [2], [4], [16]. Popular examples of relevant aggregate functions are the computation of the maximum or the average of some variable sensed by the nodes in some area. Nevertheless, any algebraic aggregate function of the sensed input-values is also of interest.

Typically, in Sensor Networks, the aggregated information is collected by a small number of distinguished nodes called *sinks*. Given that the information has to be collected to be of any use, a sink node is generally assumed to be failure-free, and to have access to more resources than a regular sensor node. For some applications, it might be useful to compute aggregations restricted to specific areas of the network, and to route the result of those computations to the sink nodes. However, lack of position information and limitations on storage space prevents area delimitation and routing. Hence, for the most restrictive and general scenario, only aggregation among *all* nodes is feasible. Additionally, the result must be propagated to all nodes in the network to guarantee that sink nodes receive it.

Algebraic aggregate functions are well defined. However, the implementation of such computations in practice, and specially in the harsh Sensor Networksetting, has to deal with various issues that make even the definition of the problem difficult. First, the input-values at each node might change over time. Therefore, it is necessary to fix to which time step those input-values correspond. This fact implies that any protocol has to achieve some form of global synchronization. Second, the multi-hop nature of Sensor Networks makes impossible to completely aggregate these values in one single time step. Hence, arbitrary node failures make the design of protocols challenging. Furthermore, it has been shown [1] that the problem of computing an aggregate function among all nodes in a network where some nodes join and leave the network arbitrarily in time is intractable. The only limit on adversarial failures that is customarily used in the Sensor Networks literature is a guarantee on connectivity among *active* nodes in each time step. An active node at time $t$ is a node that is up and running at time $t$. However, for any Sensor Network, it is easy to give a node-failure schedule that maintains such connectivity but partitions the network.

The topic of this paper is the efficient computation of aggregate functions on a Sensor Network. The efficiency is measured here in two dimensions: time and energy. The energy efficiency is evaluated in terms of number of transmissions, as customary in the Sensor Networks literature. These efficiency metrics are strongly influenced by colli-

This research was supported in part by Comunidad de Madrid grant S-0505/TIC/0285; MICINN TIN2008-06735-C02-01 and MEC PR2008-0015; EU Marie Curie European Reintegration Grant IRG 210021; NSF grants CCF0621425, CCF 05414009, CCF 0632838; and French ANR Masse de Donnèes project ALPAGE.

The first author worked partially while on leave at Bell labs.

sions, especially because no collision detection mechanisms are available in this setting. The response of the algorithm to sensor failures is also an important characteristic of any protocol. Some algorithms have to restart in presence of failures, while others simply compute an aggregated value that may be only an approximation.

Hierarchical aggregate computations where the few compute for the many have been studied. The most frequent hierarchical approach is to construct a tree that spans all nodes in the network [17], [18]. The spanning tree is used to collect and gradually aggregate the input-values at each level of the tree, relying the partial results to the root. Then, the root computes the overall aggregate result and distributes it down the tree. Due to memory size limitations, it might not be possible to implement these techniques unless the degree of each node in the tree is bounded. Another drawback of this approach comes from its rigid structure. If an internal node of the tree fails during the computation, the tree is partitioned, and the result, if computed, may not consider the input-values of an unbounded number of nodes. Furthermore, these nodes may never obtain the result.

Non-hierarchical computations have also been studied [2], [4], [16]. The approach of choice is to aggregate the information at *every* node of the network in a *mass-distribution* fashion as in load balancing algorithms [12], [20]. In this manner, all nodes arrive at the final result concurrently. A potential shortcoming is the energy consumption overhead of having all nodes transmitting and computing. Furthermore, the fact that all nodes communicate with other nodes during all the algorithm greatly increase collisions with the consequent time and energy cost. On the other hand, non-hierarchical approaches are more resilient to failures.

These arguments indicate that both pure approaches, hierarchical and non-hierarchical, may have advantages and shortcomings. The algorithm presented in this paper benefits from the good properties of both approaches interleaving them. If failures are not too frequent, the tree-based algorithm provides a result with low time and energy complexity. If the frequency of failures prevents the tree-based computation from finishing, the mass-distribution algorithm will compute an approximation of the result. Hence, the combined algorithm is *early stopping*.

In order to reduce collisions and energy consumption, a two-level hierarchy of nodes is used. The actual computation is done by a small set of nodes, called *delegate nodes,* that collect the sensed input-values from the non-computing nodes, called *slug nodes.*

**Model.** Sensor nodes are expected to be deployed at random in large quantities over an area of interest. Hence, we model the connectivity of nodes with the *Geometric Graph Model*, noted as $\mathcal{G}_{n,r}$, where $n$ nodes are deployed at random in $\mathbb{R}^2$ in a unit area, and an edge between two nodes exists if and only if they are located at an Euclidean distance of at most a parameter $r$. We further characterize the area of deployment assuming that, if expanded in all directions by a distance of $r$, the new area would not be asymptotically bigger. As customary in the Sensor Networks literature, we assume that nodes are deployed densely enough to ensure network connectivity and sensing coverage even under failures. Nevertheless, we do not restrict ourselves to an specific spatial distribution. Given that we will use a radius of transmission reduced by a constant factor in some algorithms, we further assume that such density is adjusted accordingly by a constant factor to still accomplish connectivity and coverage using the reduced radius. This assumption does not change the asymptotic cost. A straightforward application of the bound in [14] for uniform node-distribution gives a bound of $r \in \Omega(\sqrt{\log n/n})$ to achieve connectivity with high probability (w.h.p.)[1], even with non-uniform node-distribution, since the radius cannot be smaller if some areas have smaller density of nodes.

Regarding models of sensor node constraints, we use the following model, a relaxation (regarding failures) of the Weak Sensor Model [6]. The communication among neighboring nodes is through broadcast on a shared channel. Time is assumed to be slotted, and each transmission occurs in a given slot (or step). The length of a slot is the time to transmit one message. All nodes have the same clock frequency, but no global synchronizing mechanism is assumed. A node receives a message in a slot if and only if exactly one of its neighbors transmits in the slot. There is no collision detection mechanism available and the channel is assumed to have only two states: single transmission and silence/collision. A sensor node can not receive and transmit in the same slot. Nodes are woken up by an adversary, perhaps at different times. Sensor nodes may store only a constant number of $O(\log n)$ bit words[2]. We assume that sensor nodes can adjust their power of transmission to only a *constant* number of levels. Nodes are assumed to have limited life cycle. Other restrictions include: short transmission range ($r << 1$), only one shared channel of communication, and lack of position information.

As pointed out before, the problem can not be solved under arbitrary adversarial failures. We consider a scenario where, upon starting the algorithm, some nodes fail due to lack of power supply or any other event such that, as a consequence, the node stops participating in the algorithm and all the information stored in its memory is lost. A node may recover from a failure later (for instance, after replenishing its battery) but no information was kept in its memory. Thus, we assume that a node that recovers after a failure has to start the protocol from scratch. The rate or time at which failures occur is modeled as follows. Given two parameters $f \geq 0$ and $T > 0$, it is assumed that the

[1]A parameterized event $E_p$ occurs w.h.p. if, for any constant $\gamma > 0$, there exists a valid value $p$ such that $Pr\{E_p\} \geq 1 - n^{-\gamma}$.

[2]Throughout this paper, log means $\log_2$ unless otherwise stated.

number of failures is bounded by $f$ and the time between any pair of consecutive failures is at least $T$ time steps.

Regarding the assignment of input-values, it is assumed to be adversarial, i.e., we do not assume any specific distribution of input-values among nodes but just a worst-case scenario. Without loss of generality, we assume those values to be positive. We also assume that nodes are assigned a unique ID of $O(\log n)$ bits also adversarially and they start "knowing" only the total number of nodes $n$. However, the deployment of nodes is not an uncontrolled experiment. So, information about the resultant topology can be introduced at a sink node after deployment. The presence of one distinguished node called *sink* is assumed. The sink does not fail and "knows" tight bounds on the maximum degree $\Delta$ and on the maximum diameter during the whole execution of the algorithm $D$.

**Related Work.** There is a large body of literature on aggregate computations in sensor networks that includes both, theoretical and experimental work. Many of these results are obtained under models that do not include important restrictions such as, limited memory size [17], lack of position information [13] and limited range of transmission [16].

A hierarchical approach to aggregate information is presented in [13]. The solution proposed defines a tree structure that requires node location information to carry out the aggregation at a cost of $O(n \log^2 n)$ messages in $O(\log^2 n)$ *rounds*. Contention resolution and other communication issues are assumed to be resolved by an underlying protocol.

Non-hierarchical gossip-based protocols for average computations in arbitrary networks were studied [2], [16]. All gossip-based algorithms are characterized in [2] with a matrix that models how the algorithm evolves sharing values in pairs iteratively. It is shown there that, given a value $\epsilon > 0$, and an arbitrary network of $n$ nodes, where each node $i$ holds a value $\nu_i$ and all nodes start synchronously; then, with probability at least $1 - \epsilon$, in $O(\log n + \log(n/\epsilon)/(1 - \lambda_{max}((\vec{I} + \vec{P})1/2)))$ *rounds*, each node $i$ running a gossip-based algorithm characterized by the matrix $\vec{P}$, computes a value $\nu_i'$ such that $\sum_i (\nu_i' - \overline{\nu})^2 / \sum_i \nu_i^2 \leq \epsilon^2$, where $\overline{\nu}$ is the average $\sum_i \nu_i / n$ and $\lambda_{max}(\cdot)$ is the second largest eigenvalue. Additionally, an algorithm that takes advantage of the broadcast nature of radio networks is included in [16] giving similar bounds. In these papers no details about collision resolution are included, and the algorithms presented require $\omega(1)$ memory size.

Another mass-distribution algorithm was presented in [4], although relying on a different randomly chosen local leader in each round to perform such distribution. It is shown that, given a value $\epsilon > 0$, and a Sensor Networkof $n$ nodes with underlying graph $G$ with algebraic connectivity $a(G)$[3], where each node $i$ holds a value $\nu_i$ and all nodes start synchronously, with probability at least $1 - \epsilon^2 / \sum_i (\nu_i - \overline{\nu})^2$, in $O(\Delta^3 \log(\sum_i (\nu_i - \overline{\nu})^2 / \epsilon^2) / a(G))$ *rounds*, each node $i$ running the algorithm computes a value $\nu_i'$ such that $|\nu_i' - \overline{\nu}| \leq \epsilon \ \ \forall i$, where $\overline{\nu}$ is the average $\sum_i \nu_i / n$. If the deployment topology is known in advance, a parameter probability $p$ can be tuned to improve that bound to $O(\Delta \log(\sum_i (\nu_i - \overline{\nu})^2 / \epsilon^2) / pa(G))$ *rounds*. The expected number of transmissions is bounded by $O(n\Delta^2 \log((\sum_i (\nu_i - \overline{\nu})^2) / \epsilon^2) / a(G))$, again, aside from communication and synchronization overhead. This result was extended recently [3] to networks with a time-varying connection graph.

**Results.** An early-stopping protocol that computes the average function in Sensor Networks under a harsh model of sensor restrictions is presented in this paper. Although this protocol builds incrementally over known techniques, the careful combination of them in the restricted Sensor Networksetting is not trivial. It is shown here that, in presence of $f$ non-frequent failures, w.h.p., the protocol returns a value and terminates in $O(\Delta + D + f \log^2 n)$ steps which, given that $D$ and $\Delta$ cannot be both asymptotically smaller than a polynomial, is optimal if $f \in o(n^c)$ for any $c \in O(1)$; and that the overall number of transmissions is in $O(n((f+1)\log n + \Delta / \log n + \log \Delta))$ in expectation. In presence of frequent failures, nodes running the protocol still converge to some result, whose accuracy with respect to the average depends on the failure model and the distribution of input-values. The aim in this case is to obtain a result that does not diverge significantly from one node to another. More precisely, it is shown that, w.h.p., the protocol takes $O(\Delta + D + (f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min}))/\Phi_{min}^2)$ time steps and $O(n(\log n + \Delta / \log n + \log \Delta + (f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min})))/\Phi_{min}^2 \log n)$ expected transmissions, where $\varepsilon > 0$ is the maximum relative error (i.e., the maximum relative difference between two results), $\Phi_{min}$ is the minimum conductance [15] of the network underlying the Markov chain characterizing the algorithm, and $\nu_{max}$ and $\nu_{min}$ are the maximum and minimum input-values respectively.

A time-optimal protocol to compute the maximum function can be easily derived from the average protocol by flooding the delegates network with the maximum input-value seen so far. Other aggregate functions such as the sum, quantiles, or count, can be computed using a protocol for average without extra cost as described in [4], [16].

To the best of our knowledge, this is the first optimal early-stopping algorithm for aggregate computations in Sensor Network.

## II. Lower Bound

A lower bound on the time steps needed to compute an aggregate function in a Sensor Networkis established in the theorem below. For the sake of brevity, the details of the proof that uses the adversarial assignment of input-values

[3] A characterization of the deployment topology by the second smallest eigenvalue of the Laplacian matrix of $G$.

and the topology, is left to the full version of this paper [11]. Let $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}, n \in \mathbb{N}$ be an algebraic aggregate function over $n$ real numbers. We say that $\mathcal{F}$ is **one-node sensitive** if, for any choice of values $\vec{\nu}_1 \in \mathbb{R}^n$, there exists another choice of values $\vec{\nu}_2 \in \mathbb{R}^n$ such that $\vec{\nu}_1$ and $\vec{\nu}_2$ differ only in one value, and $\mathcal{F}(\vec{\nu}_1) \neq \mathcal{F}(\vec{\nu}_2)$. Given a Sensor Networkof $n$ nodes, where each node is assigned an input-value, we say that a protocol to compute an aggregate function over these values is **assignment oblivious** if it is independent of the specific assignment of input-values.

**Theorem 1.** *Given a Sensor Networkof $n$ nodes, where $D$ is the diameter of the network and $\Delta$ the maximum degree, under the model described in Section I, and independently of randomization and failures, $\Omega(D + \Delta)$ time steps are needed in order to compute a one-node-sensitive algebraic aggregate function using an assignment-oblivious protocol.*

## III. Upper Bounds

The computation of aggregate functions is carried out by a protocol following a template called *Aggregate Computation Scheme*. A key factor of our approach is the inclusion of a preprocessing phase that defines a delegate-slug hierarchy and a schedule of transmissions to avoid collisions. Such preprocessing is asynchronous and uses time slots that are not used in the Aggregate Computation Scheme. Hence, it is also used as a maintenance algorithm in case of node failures because nodes running it do not collide with nodes running the main part. For the sake of clarity, both parts, preprocessing and the main procedure, are described separately omitting these details.

### A. Preprocessing and Maintenance

The preprocessing/maintenance algorithm includes two phases. Due to the memory size limitations, in the first phase delegate nodes are defined so that each delegate node is within range of $\Theta(1)$ delegates. Additionally, a second phase establishes schedules of transmissions so that each group delegate-slugs can communicate without colliding with neighboring groups. The first and second preprocessing phases can be implemented as in [8]. Further details can be found in the full version of this paper [11]. The following upper bound on the number of delegate nodes can be proved using that the hexagonal lattice is the densest of all possible plane packings [9], that the radius lower bound to achieve connectivity w.h.p. under uniform distribution of nodes is $r \in \Omega(\sqrt{\log n/n})$ [14], the assumption of complete coverage, and the assumption regarding the area of deployment.

**Remark 2.** *Given a Sensor Networkof $n$ nodes deployed at random to ensure connectivity and coverage over a unit area such that, if expanded in all directions by $r$, the expanded area is still in $O(1)$, after running the first phase of preprocessing as described, there are $O(n/\log n)$ delegate nodes.*

The following lemma establishes formally the efficiency of these phases. Further details can be found in [6]–[8] and the references therein.

**Lemma 3.** *(a) For any node $i$ running the first phase of preprocessing, for any $0 < \alpha \leq 1$, at least one node within distance $\alpha r$ of $i$ becomes a delegate within $O(\log^2 n)$ time steps and no two delegate nodes are within distance $\alpha r$ of each other w.h.p. The expected number of transmissions of $i$ during this phase is in $O(\log n)$ w.h.p. (b) For any node $i$ running the second phase of preprocessing, if $i$ is a delegate node, after $O(\log n)$ time steps $i$ reserves a block of $b \in O(1)$ steps every $\gamma \in O(1)$ steps for local use, i.e., this block does not overlap with the block of any other delegate node separated by a distance at most $r$, w.h.p. During this phase, if $i$ is a delegate node the expected number of transmissions of $i$ is in $O(\log n)$ w.h.p., and if $i$ is a slug node it does not transmit.*

### B. The Aggregate Computation Scheme

After preprocessing, local synchronism, collision detection among slugs and their delegates, and non-colliding transmission schedules among delegates are available, because nodes use only reserved time slots. We omit in the analysis this constant factor overhead for clarity. In the Aggregate Computation Scheme, a slug node uses a radius of transmission $\alpha r$, whereas a delegate node uses $\beta r$. Also for clarity, the scheme is described assuming that nodes do not fail and later this assumption is removed. In order to obtain worst-case bounds, we assume that all nodes are active.

Let the set of delegate nodes and the set of slug nodes defined in preprocessing be $M$ and $S$ respectively. For each slug node $i$, denote the set of its delegates as $M(i)$. For each delegate node $j$, denote the subset of delegate nodes located at one-hop of $j$ as $N(j)$. Each node $j \in M$ keeps track of its delegate-neighborhood $N(j)$. Furthermore, node $j$ updates $N(j)$ online by keeping track of the beacon messages of its delegate-neighbors. This bookkeeping can be done by storing the IDs of the neighboring delegates, because $|N(j)| \in \Theta(1)$. For each node $k$ in the network, denote the input-value as $\nu_k$.

The Aggregate Computation Scheme includes the following four phases. TRIGGER: the sink node broadcasts $(\tau_1, D, \Delta)$, where $\tau_1$ is the time slot to measure the input-values, $D$ is the diameter of the network and $\Delta$ the maximum degree, thus, synchronizing the computation; COLLECTION: delegate nodes aggregate slugs input; COMPUTATION: delegate nodes compute the aggregate function; and DISSEMINATION: delegate nodes distribute the result. The details of the implementation of each of these phases follow.

**Trigger Phase.** The TRIGGER phase can be implemented as follows. Upon receiving the message, delegates flood the

network of delegates with the message using only reserved slots. Each delegate node forwards the message broadcasted, including the ID of the node from which it has received the message first. In this manner, a BFS spanning tree among the delegate nodes is obtained at the same time that the trigger signal is disseminated in preparation for our tree-based algorithm. Due to the broadcast nature of a Sensor Network, while passing the message among delegates, slug nodes receive also $\tau_1$. Since only reserved slots are used, the total time taken by this phase is in $O(D)$ and using the Remark 2 the total number of transmissions in this phase is in $O(n/\log n)$. Hence, $\tau_1$ is tuned to ensure that active nodes receive this message on time to start the COLLECTION phase. For nodes becoming active late, upon becoming active, nodes run the preprocessing phase, which includes an initial waiting period. Nodes in this period that hear that the computation has already started do not join the computation, although they do complete the preprocessing phase in preparation for future queries. The following lemma establishes formally the bounds of this phase.

**Lemma 4.** *After the sink node starts disseminating the trigger message, all delegate nodes have received the message within $O(D)$ steps and the overall number of transmissions is $O(n/\log n)$.*

**Collection Phase.** At time $\tau_1$, nodes start running the COLLECTION phase using the input-values at that time step. Slug nodes communicate in this phase using the following procedure. In each round, slug nodes choose uniformly at random a slot within a window of slots to transmit their messages. Starting with a window of size $c_1\Delta$, the window size is repeatedly halved in each round down to $c_2 \log n$, where $c_1 > 0$ and $c_2 > 0$ are constants chosen appropriately. After that, a final round of $c_3 \log^2 n$ steps where nodes repeatedly transmit with probability $c_4/\log n$ is included. Again, $c_3 > 0$ and $c_4 > 0$ are constants chosen appropriately [8]. We refer to this protocol as the *windowed protocol*. Further details can be found in the full version of this paper [11].

Each slug node $i \in S$ begins the COLLECTION phase choosing one of its delegates to pass its input-value, to ensure that each input-value is used exactly once in the computation. Using the windowed protocol, each slug node transmits a message to the delegate chosen. The message transmitted contains $\nu_i$ and the ID of the delegate chosen. Given the availability of delegate acknowledgements, a delegate receives exactly one input-value per slug node.

Each delegate node $j \in M$ running the COLLECTION phase maintains two magnitudes that we call *sum* $\sigma_j$ and *weight* $\omega_j$. Each delegate node $j$ initializes the sum $\sigma_j = \nu_j$ and the weight $\omega_j = 1$. Upon receptions, delegates update these values appropriately. Sum and weight values are polynomially upper bounded so memory restrictions are not violated. The following lemma establishes formally the correctness and efficiency of the COLLECTION phase. The proof uses well-known techniques and the details are left to the full version of this paper [11] for brevity.

**Lemma 5.** *Let $V$ be the set of $n$ nodes in a Sensor Network, $\nu_i$ be the input-value of node $i \in V$, and let $M$ be the set of delegate nodes. There exists a $\tau_2 \in O(\Delta + \log^2 n)$ such that, after running the* COLLECTION *phase with that $\tau_2$, the following holds. (i) $V$ has been partitioned in $|M|$ disjoint subsets $\{V_1, V_2, \ldots, V_{|M|}\}$ and each node $j \in M$ holds two values $\sigma_j$ and $\omega_j$ such that, $\forall k \in \{1, \ldots, |M|\}; \forall j \in M : j \in V_k \Rightarrow (\sigma_j = \sum_{i \in V_k} \nu_i \wedge \omega_j = |V_k|)$, w.h.p. (ii) The time taken by the algorithm is in $O(\Delta + \log^2 n)$. (iii) The number of transmissions of delegate nodes during this phase is in $O(n(\Delta/\log n + \log n))$, and the expected number of transmissions of slug nodes during this phase is in $O(n(\log n + \log \Delta))$.*

**Computation and Dissemination Phases.** Upon completion of the COLLECTION phase, slug nodes standby waiting for the delegates to compute in the COMPUTATION phase and send back to them the result in the DISSEMINATION phase. In the following sections, the two approaches used – tree-based and mass-distribution – are described separately for clarity, although they are run simultaneously in two different slots reserved to communicate among delegates. If the result of the tree-based computation is obtained, the mass-distribution-based computation is just stopped. Otherwise, the mass-distribution algorithm continues until some result is returned.

Before moving to the details of the analysis of both algorithms, recall that thanks to the delegate/slug hierarchy, a failure of a slug node after passing its input-value does not impact the protocol neither in time nor in correctness. On the other hand, if a unique copy of an input-value is lost before being passed to other nodes that value is inevitably lost. Furthermore, given the shared nature of the channel, no algorithm can guarantee that all input-values are passed to some other node in less than $\Delta$ time steps under adversarial failures. Hence, we consider from now on slug failures only during the COLLECTION phase.

If a delegate node fails early enough before the time slot in which input-values are obtained, the lack of its beacon triggers the execution of the preprocessing phase by its slug nodes and this failure does not impact the protocol. On the other hand, if the failure occurs at a time slot such that its slug nodes do not have enough time to elect new delegates and receive the synchronization message of the TRIGGER phase, its slug nodes do nothing. Given that most of the slug nodes have more than one delegate, this failure does not impact the protocol except in some marginal cases (boundary nodes or multiple neighboring delegates failure). Given that a delegate failure during the COLLECTION phase has the same impact as a failure of a delegate during the first round of the COMPUTATION phase. Thus, we consider from now

on only delegate failures during the COMPUTATION phase.

**Tree-based Algorithm.** For the sake of clarity, we describe first the algorithm assuming that nodes are activated early enough to receive the trigger, stay active long enough to receive the result of the computation, and do not fail. The slug and delegate failures described in Section III-B are considered afterwards. The tree-based algorithm is well known and simple to describe. Once a rooted tree is built, it includes three steps: the root broadcasts a query to all nodes in the tree, then nodes convergecast the aggregated input-values to the root and finally the root computes the function and broadcasts back the result to all nodes in the tree. The details follow.

While broadcasting the time slot $\tau_1$ of the input-values that have to be used in the computation, in the TRIGGER phase, a BFS rooted tree of constant degree is built among delegate nodes by making each delegate node keep track of its tree neighbors. The root of such a tree is either the sink node (if delegate) or a delegate node at one hop of the sink node (if slug). Without loss of generality we assume it is the sink node. At $\tau_1$, all nodes run the COLLECTION phase using the windowed protocol as described. Then, at time $\tau_1 + \tau_2$, the COMPUTATION phase starts. In this phase, each delegate node $i$ aggregates the input-values by passing to its parent in the tree the average and weight of the subtree rooted at $i$. Thus, the root of the tree receives the average and weight of the subtrees rooted at its children and computes the total average. Finally, in the DISSEMINATION phase, the root node floods the network of delegates with the result which in turn is disseminated to the slug nodes by each delegate node upon receiving it.

In order to handle failures, the tree-based algorithm is enhanced as follows. Upon defining the tree, each delegate node broadcasts to its slugs its view of the tree topology, i.e., its parents and children, and the slugs store that information. Since the tree has constant degree, such a bookkeeping is feasible. Slug nodes detect the failure of their chosen delegate due to the lack of beacon. In presence of such a failure, slug nodes compete to replace the missing delegate running the preprocessing phase at a $O(\log^2 n)$ cost (Lemma 3). Given the assumption of coverage even under failures, there must exist enough slug nodes to replace the failed delegate. Due to the geometry, more than one of them may become delegate but only a constant number. Upon becoming delegates and using the view of the tree broadcasted by the failed delegate, the new delegates repair the structure locally at a $O(1)$ cost and continue with the computation appropriately. The details are omitted for brevity. Then, if the time between failures is large enough, this procedure repairs the structure successfully. The following theorem shows the overall efficiency of the Aggregate Computation Scheme when the tree-based algorithm is used.

**Theorem 6.** *Given a Sensor Networkwith a set of nodes $V$ running the Aggregate Computation Scheme as described, where $\Delta$ is a tight upper bound on the maximum number of neighbors of any node, $D$ is a tight upper bound of the diameter of the network during all the execution of the algorithm, and $\tau_1$ is the time slot at which the input-values are assigned. Under the model described in Section I, if the number of node failures after $\tau_1$ is bounded by $f \geq 0$, and $T$ is the minimum time between any pair of consecutive failures. There exist positive constants $\kappa_1, \kappa_2$ such that, if $V' \subseteq V$ is the set of nodes awake in all the interval $[\tau_1 - \kappa_1(D + \log^2 n), \tau_1]$, $\nu_i$ is the input-value assigned to node $i \in V'$ at time $\tau_1$, $\nu_{max} = \max_{i \in V'} \nu_i$, $\nu_{min} = \min_{i \in V'} \nu_i$, and $\overline{\nu} = \sum_{i \in V'} \nu_i / |V'|$, the following holds. If $T \geq \kappa_2 \log^2 n$, w.h.p., within $O(\Delta + D + f \log^2 n)$ time steps after $\tau_1 - \kappa_1(D + \log^2 n)$, all nodes running the algorithm receive (or hold) the same value, that value is in the range $[(\overline{\nu}|V'| - f\nu_{min})/(|V'| - f), (\overline{\nu}|V'| - f\nu_{max})/(|V'| - f)]$ and the expected number of transmissions is in $O(n((f + 1)\log n + \Delta/\log n + \log \Delta))$ w.h.p.*

*Proof:* As explained before, the failure of a slug node $i$ only impacts the computation if $i$ is running the COLLECTION phase and did not pass its input-value to its delegate yet. Due to the assumption of coverage, even under failures, delegate nodes that fail during the COLLECTION and COMPUTATION phases are replaced in the tree-based algorithm by one or more of its slugs, introducing a $O(\log^2 n)$ overhead for each failure as shown in Lemma 3. Thus, the claimed range of the result follows. Regarding the running time, given that broadcast and convergecast is run in reserved slots the time taken by the last two phases is $O(D + f \log^2 n)$, using Lemmas 3, 4, and 5, given that $D$ and $\Delta$ can not be in $o(\log^3 n)$ simultaneously, and given that the number of failures is at most $f$ the claim follows. The claimed number of transmissions is a direct consequence of Lemmas 3, 4, 5, Remark 2 and including the worst-case overhead of replacing the failed delegates. ■

**Mass-distribution Algorithm.** For clarity, let us assume first that nodes do not fail. In the mass-distribution protocol used in this paper, after aggregating input-values in the COLLECTION phase, delegate nodes share a fraction with each delegate neighbor. More precisely, recall that $\max_{i \in M}\{|N(i)|\} \leq 3\lceil 2\beta/\alpha\sqrt{3}\rceil(\lceil 2\beta/\alpha\sqrt{3}\rceil + 1)$ as shown in Section III. Then, fix $\delta = 1 + 3\lceil 2\beta/\alpha\sqrt{3}\rceil(\lceil 2\beta/\alpha\sqrt{3}\rceil + 1)$. Upon termination of the COLLECTION phase, each delegate node $i \in M$ computes a local average $\overline{\nu}_i = \sigma_i/\omega_i$. From there on, the computation progresses as if all nodes in the network (slugs and delegates) were participating in it using this initial local-average value, although delegate nodes take on the task for the slug nodes. More precisely, in each round, each delegate node passes its weight and a fraction $1/2\delta\Delta$ of its weighted average to each neighboring delegate node, keeping the rest of the weighted average for itself. Then, delegate nodes update their average values appropriately

using the shares and weights received, and repeat. After sufficient number of iterations, all average values converge to the average sought. We call this protocol *Mass Distribution.*

Given that the shares are the same for all neighbors and $\delta$ and $\Delta$ are known[4], delegate nodes do not need to specify the destination and simply transmit the average and weight. After enough number of rounds of Mass Distribution, each delegate node $i$ obtains the average with the accuracy desired. Furthermore, the DISSEMINATION phase is integrated in the COMPUTATION phase by default given that, although averages and weights are transmitted to neighboring delegate nodes, all neighboring nodes receive those transmissions because they are produced in reserved slots. Notice that Mass Distributiondoes not violate the memory restrictions since only a constant number of values are received in each round and the average and weight values are still polynomially upper bounded. Of course, precision limitations due to real number computations are still in order.

In presence of the slug and delegate failures described in Section III-B, slug nodes do nothing and delegate nodes adjust their delegate neighborhood appropriately. If a delegate node fails before broadcasting its values, the failure has the same impact on the computation as if it fails at the beginning of the round. If, on the other hand, the node fails after broadcasting its values, the failure has the same impact in the computation as if it fails at the end of the round. Therefore, without loss of generality, to analyze the convergence of Mass Distributionwe assume that delegate-node failures occur between rounds.

We analyze now Mass Distribution[5]. Assume first that nodes do not fail. Given that the fraction shared in Mass Distributionis round independent, the algorithm can be characterized by a matrix of shares as follows. Let $\vec{\overline{\nu}}^{(t)} = (\overline{\nu}_1^{(t)} \dots \overline{\nu}_n^{(t)}), i \in V$ be the vectors[6] of averages held by nodes after round $t$. (Let the average held by a slug node be the average held by its chosen delegate.) Let $\vec{P} = (p_{ij})$ be a matrix in $\mathbb{R}^{n \times n}$ such that $p_{ij} = 1/2\delta\Delta$ if $i$ and $j$ have chosen delegates $r$ and $s$ respectively such that $r \in N(s)$, $p_{ij} = 1 - (\sum_{s \in N(r)} \omega_s)/2\delta\Delta$ if $j = i$ and $i$ has chosen delegate $r$, and $p_{ij} = 0$ otherwise. Then, $\vec{\overline{\nu}}^{(t)} = \vec{\overline{\nu}}^{(0)} \vec{P}^t$ is the vector of averages in round $t$. Given that $\vec{P}$ is stochastic, this characterization can be also seen as a Markov chain $\mathbf{X} = \{\mathbf{X}_t\}$ where the state space is $V$ and the transition matrix is $\vec{P}$.

Mass-distribution algorithms only converge to the result. Hence, a metric of such an approximation has to be defined. In this paper, we use the *relative point-wise distance*, which is defined as $\max_i |\nu_i - \overline{\nu}|/\overline{\nu}$. The correctness of the average computation implemented with mass distribution algorithms is a well-known fact that can be proved using the fundamental theorem of Markov chains [19]. We establish formally the correctness of the average computation in the following lemma. For the sake of brevity, the proof is left to the full version of this paper [11].

**Lemma 7.** *(Correctness) Let $V$ be the set of $n$ nodes in a Sensor Network, $\nu_i$ be the input-value of node $i \in V$, and $\overline{\nu} = \sum_{i \in V} \nu_i/n$ their average. Let $\overline{\nu}_i^{(t)}$ be the average held by the delegate node chosen by node $i \in V$ obtained $t$ rounds after the* COLLECTION *phase of the Aggregate Computation Scheme. Then, if delegate nodes do not fail, implementing the* COMPUTATION *phase using Mass Distribution, there exists a $\tau_3 \geq 0$ such that, for all $t \geq \tau_3$, $|\overline{\nu} - \overline{\nu}_i^{(t)}|/\overline{\nu} \leq \varepsilon$, for all $i \in V$ and for a given parameter $\varepsilon > 0$.*

In presence of node failures, the fraction of average shared in Mass Distributionis not round independent. Therefore, instead, we characterize the computation carried out by Mass Distributionunder failures as a sequence of matrices $\vec{P}_0, \vec{P}_1, \vec{P}_2, \dots$ such that $\vec{P}_k, k \geq 0$, is the matrix of shares characterizing the algorithm between failures $k$ and $k+1$. Given that these matrices are stochastic, each of these characterizations can be also seen as a Markov chain $\mathbf{X}_k$ where the state space is $V_k$, the set of nodes whose delegate is active between failures $k$ and $k+1$, and the transition matrix is $\vec{P}_k$.

To analyze the efficiency of Mass Distribution, we leverage the vast body of research work on bounding the mixing time of Markov chains. We bound the mixing time using the *conductance* as in [21]. The conductance is a natural notion for Markov chains with underlying graphs with geometric properties. To show the overall efficiency of Mass Distribution, we bound the convergence time of each of the Markov chains and combine the effect of all of them in Lemma 8. For the sake of brevity, the details of the proof are left to the full version of this paper [11].

**Lemma 8.** *Given a Sensor Networkwith a set of nodes $V$, where $\nu_{max} = \max_{i \in V} \nu_i$ and $\nu_{min} = \min_{i \in V} \nu_i$ are the maximum and minimum input-values assigned to nodes respectively, nodes are running the Aggregate Computation Scheme as described, and the* COMPUTATION *phase is implemented using Mass Distributionas described, after $O((f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min}))/\Phi_{min}^2)$ rounds, where $f$ is the number of node failures and $\Phi_{min} = \min_{k \in \{0,1,\dots,f\}} \Phi_k$ where $\Phi_k$ is the conductance of the underlying graph after the $k$th failure, all nodes have converged to a value with relative error $0 < \varepsilon < 1$.*

The following theorem shows the overall efficiency of the Aggregate Computation Scheme when the mass-distribution algorithm is used.

**Theorem 9.** *Given a Sensor Networkwith a set of nodes $V$*

[4] $\delta$ is a constant that can be stored before deployment. $\Delta$ is broadcasted in the TRIGGER phase.

[5] We assume familiarity with Markov chains and spectral graph theories. For an introduction refer to [5], [10].

[6] Throughout the paper, we use row vectors for clarity.

*running the Aggregate Computation Scheme as described, where $\Delta$ is a tight upper bound on the maximum number of neighbors of any node, $D$ is a tight upper bound of the diameter of the network during all the execution of the algorithm, and $\tau_1$ is the time slot at which the input-values are assigned. Under the model described in Section I, if the number of node failures after $\tau_1$ is bounded by $f \geq 0$, and $T$ is the minimum time between any pair of consecutive failures. There exist positive constants $\kappa_1, \kappa_2$ such that, if $V' \subseteq V$ is the set of nodes awake in all the interval $[\tau_1 - \kappa_1(D + \log^2 n), \tau_1]$, $\nu_i$ is the input-value assigned to node $i \in V'$ at time $\tau_1$, $\nu_{max} = \max_{i \in V'} \nu_i$, $\nu_{min} = \min_{i \in V'} \nu_i$, and $\overline{\nu} = \sum_{i \in V'} \nu_i / |V'|$, the following holds. If $T < \kappa_2 \log^2 n$, within $O(\Delta + D + (f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min}))/\Phi^2_{min})$ time steps of $\tau_1 - \kappa_1(D + \log^2 n)$, where $\Phi_{min} = \min_{k \in \{0,1,\ldots,f\}} \Phi_k$ where $\Phi_k$ is the conductance of the underlying graph after the $k$th failure, all nodes running the algorithm have converged to the same result in the range $[\nu_{max}, \nu_{min}]$, with relative error $0 < \varepsilon < 1$ w.h.p., and the expected number of transmissions is in $O(n(\log n + \Delta/\log n + \log \Delta + (f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min})))/\Phi^2_{min} \log n)$ w.h.p.*

*Proof:* The running time is a direct consequence of Lemmas 3, 4, 5 and, given that each round takes $O(1)$ time steps, Lemma 8, and using that $D$ and $\Delta$ can not be in $o(\log^3 n)$ simultaneously. The claimed number of transmissions follows from Lemmas 3, 4, 5, 8, and Remark 2. ■

## REFERENCES


[1] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Stanford University, Database group, 2003.

[2] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530, 2006.

[3] Jen-Yeu Chen and Jianghai Hu. Analysis of distributed random grouping for aggregate computation on wireless sensor networks with randomly changing graphs. *IEEE Trans. Parallel Distr. Syst.*, 19(8):1136–1149, 2008.

[4] Jen-Yeu Chen, Gopal Pandurangan, and Dongyan Xu. Robust computation of aggregates in wireless sensor networks: distributed randomized algorithms and analysis. *IEEE Trans. Parallel Distr. Syst.*, 17(9):987–1000, 2006.

[5] Fan Chung. Spectral graph theory. http://www.math.ucsd.edu/ fan/research/revised.html, 2006.

[6] M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Bootstrapping a hop-optimal network in the weak sensor model. *ACM Transactions on Algorithms*, 2008. In press.

[7] M. Farach-Colton and M. A. Mosteiro. Initializing sensor networks of non-uniform density in the weak sensor model. In *Proc. of 10th Intl. Workshop on Algorithms and Data Structures*, volume 4619 of *Lecture Notes in Computer Science*, pages 565–576. Springer-Verlag, Berlin, 2007.

[8] M. Farach-Colton and M. A. Mosteiro. Sensor network gossiping or how to break the broadcast lower bound. In *Proc. of the 18th Intl. Symp. on Algorithms and Computation*, volume 4835 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, Berlin, 2007.

[9] L. Fejes-Tóth. über einen geometrischen satz. *Mathematische Zeitschrift*, 46(1):83–85, 1940.

[10] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley & Sons, Inc., New York, NY, USA, 3rd edition, 1968.

[11] Antonio Fernández Anta, Miguel A. Mosteiro, and Christopher Thraves. An early-stopping protocol for computing aggregate functions in sensor networks. Technical Report RoSaC-2008-3, GSyC, LADyR, Universidad Rey Juan Carlos, 2008.

[12] B. Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *Journal of Computer and System Sciences*, 53(3):357–370, 1996.

[13] Indranil Gupta, Robbert van Renesse, and Kenneth P. Birman. Scalable fault-tolerant aggregation in large process groups. In *DSN*, pages 433–442. IEEE Computer Society, 2001.

[14] P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. In *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming.*, pages 547–566. Birkhauser, Boston, 1998.

[15] Mark Jerrum and Alistair Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of permanent resolved. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing*, pages 235–244, 1988.

[16] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th IEEE Ann. Symp. on Foundations of Computer Science*, pages 482–491, 2003.

[17] G. Kollios, J. W. Byers, J. Considine, M. Hadjieleftheriou, and F. Li. Robust aggregation in sensor networks. *IEEE Data Engineering Bulletin*, 28(1):26–32, 2005.

[18] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proc. of the 5th Symp. on Operating Systems Design and Implementation*, pages 131–146, 2002.

[19] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[20] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load-balancing schemes. In *Proc. of the 39th IEEE Ann. Symp. on Foundations of Computer Science*, pages 694–703, 1998.

[21] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.