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

Antonio Fernández Anta[a,1], Miguel A. Mosteiro[a,b], Christopher Thraves[c]

[a]*LADyR, GSyC, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, Spain*
[b]*Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA*
[c]*ASAP Project team, IRISA/INRIA Rennes, Campus Universitaire de Beaulieu, 35043 Rennes Cedex, France*

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.

[☆]This research was supported in part by Comunidad de Madrid grant S-0505/TIC/0285; MICINN TIN2008-06735-C02-01 and MEC PR2008-0015; NSF grants CCF0621425, CCF 05414009, CCF 0632838; Marie Curie International Reintegration Grant within the 7th European Community Framework Programme, IRG 210021; and French ANR project Shaman.

[☆☆]A preliminary version of this work is included in *Proceedings of the IEEE 15th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2009.

Email addresses: anto@gsyc.es (Antonio Fernández Anta), mosteiro@cs.rutgers.edu,miguel.mosteiro@urjc.es (Miguel A. Mosteiro), christopher.thraves-caro@irisa.fr (Christopher Thraves)

[1]Partially done while on leave at Bell labs.

Key words: sensor networks, aggregate computation, early-stopping algorithm, failure model, average computing

1. Introduction

A *Sensor Network* is a simplified abstraction of a large monitoring infrastructure, formed by *sensor nodes* (or sensors) that create a radio communication network from scratch. Each sensor node is equipped with communication, processing, and sensing capabilities. However, given its small size and low-cost, it is assumed that a sensor node will operate under strict limitations on energy supply and computational resources such as memory size [29]. Thus, due mainly to the energy constraint, individual sensor nodes are unreliable. Additionally, deterministic deployment of sensors is not feasible because Sensor Networks are expected to be used in remote or hostile areas. Random deployment and unreliability, together with the limited range of communication and harsh resource restrictions, make solving even basic problems very challenging. Therefore, classical solutions for basic problems such as establishing the network upon deployment, or achieving reliable communication among nodes, had to be revised [2, 1, 36].

A natural question is which problems that are useful for monitoring purposes can be solved in a Sensor Network. 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 [25, 8, 5]. Popular examples of relevant aggregate functions are the computation of the maximum or the average of some variable (e.g.: temperature) 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 (maybe one) 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 precludes 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 Network setting, 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 [4] 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, there are node-failure schedules that maintain such connectivity but partition the network[2].

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 collisions, 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 [30, 26]. 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,

[2]E.g.: a bipartition in two connected components that are powered off by the adversary in odd and even steps respectively.

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 [25, 8, 5]. The approach of choice is to aggregate the information at *every* node of the network in a *mass-distribution* fashion as in load balancing [35, 17] algorithms. In this manner, all nodes arrive at the final result concurrently. A potential shortcoming of this approach 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. As opposed to their hierarchical counterpart, non-hierarchical approaches usually obtain *some* result even in presence of node failures. Thus, 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 by combining them. The protocol presented interleaves two algorithms, one following a tree-based approach and one following a mass-distribution approach. The tree-based algorithm will provide the correct result with low time and energy complexity if failures are not too frequent. If the frequency of failures prevents the tree-based computation from finishing, the mass-distribution algorithm will compute and disseminate an approximation of the result. The time taken by this algorithm is larger, but it is only incurred in presence of frequent failures, since as soon as the tree-based algorithm finishes, the execution of the mass-distribution algorithm is aborted. Hence, the combined algorithm is *early stopping*. An algorithm is early stopping if it works more efficiently in a failure-free execution than in an execution with failures.

Combining these two algorithms for the harsh SN setting where energy is scarce and failures may be frequent is not a trivial task. 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*. This structure has several advantages. First, collisions are reduced since they can only occur while defining this hierarchy or when the delegate nodes collect the sensed input-values from the slug nodes. After that, delegate nodes are able to communicate in a collision-free fashion. Second, the subnetwork of delegate nodes has constant degree, which allows to easily build a constant-degree spanning tree. Collision-free communication and a constant-degree structure let us obtain a time-optimal hierarchical algorithm. Third, energy is saved because the slug nodes can idle

during the computation. Finally, since the set of delegate nodes is small, there is a smaller probability that the tree-based algorithm will fail (since only failures of delegate nodes impact on it). In presence of failures, the two-level structure may have to be reconstructed; but, this can be done fast and locally.

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 assume that the area covered by a sensor node coincides with the range of transmission. Otherwise, the analysis can be augmented with a sensing radius. 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. Thus, we do not restrict to a node distribution or a particular shape of area deployment as in the popular random geometric graph model. 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 [19] for uniform node-distribution gives a bound of $r \in \Omega(\sqrt{\log n/n})$ to achieve connectivity with high probability (w.h.p.)[3], 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. The following assumptions are standard in the Radio Networks literature. Refer to [9] and the references therein for further details. 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 use of a slotted scenario instead of a more realistic unslotted one was justified in [37], where it was shown that they differ only by a factor of 2 because a packet can interfere in no more than 2 time slots. The length of a (communication) slot is the time to transmit one message. All nodes have the same clock frequency, and are synchro-

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

nized at the slot level, 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. Sensors nodes have *non-simultaneous reception and transmission*.

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[4]. We assume that sensor nodes can adjust their power of transmission (effectively adjusting their communication radius) 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. Hence, a node-failure model is defined. 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 number of failures is bounded by f and the time between any pair of consecutive failures is at least T time steps.

In order to highlight the relevance of this work, we compare our model with respect to previous models of node constraints. Our model is a relaxation of the Weak Sensor Model [12], which includes the same set of restrictions but where failures are arbitrary. As mentioned before, arbitrary failures would yield the problem unsolvable. Bar-Yehuda, Goldreich, and Itai [3] used a formal model of Radio Network, which specifies many of the node restrictions here, including limits on contention resolution, but they make no mention of computational limits such as small memory. Later on, more restrictions have been added to the model in various papers, such as in the unstructured Radio Network model [28]. Notice that the unstructured Radio Network model does not include all the restrictions of our model. For instance, that model does not include limits on the number of levels of transmission power and lack of position information. But, more importantly, the unstructured Radio Network model does not include limits on memory size,

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

a fundamental restriction [29]. As detailed in the related work section, relevant work in aggregate computation cannot be implemented under such restriction, given that information about $\omega(1)$ neighbors has to be gathered.

We further assume that the assignment of input-values is adversarial. In other words, 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 know only the total number of nodes n. However, the deployment of nodes is not an uncontrolled experiment. For instance, a scenario frequently used for motivation is an area where nodes are dropped from a helicopter. Hence, by keeping track of the number of nodes dropped in each region, an asymptotic upper bound on the maximum degree can be obtained. Additionally, provided that enough number of nodes are deployed to guarantee connectivity, the transmission radius and the maximum length of the area of deployment give also an asymptotic upper bound on the network diameter. So, information about the resultant topology can be introduced at a sink node after deployment. In this paper, we assume the presence of one distinguished node called sink, that does not fail and knows tight bounds on the diameter of the network D and the maximum degree Δ. D is the maximum diameter during the whole execution of the algorithm which, in presence of nodes failures, may be bigger than the diameter at the time of deployment. Assuming that the computation time-cost is negligible in comparison with the communication time-cost, time efficiency is studied in terms of slots (or steps).

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 [26], lack of position information [11, 18] and limited range of transmission [23, 25]. Others, use purely experimental evaluation [39, 30, 34, 31, 27, 21, 22, 20].

A hierarchical approach to aggregate information is presented in [18]. The solution proposed defines a tree structure that requires node location information to carry out the aggregation and it contemplates node failures in large networks. The protocol presented computes aggregation functions with $O(n \log^2 n)$ messages in $O(\log^2 n)$ *rounds*. Contention resolution and other communication issues are assumed to be resolved by an underlying protocol.

Generic non-hierarchical gossip-based[5] protocols for average computations in arbitrary networks were studied in [5, 25]. Results in [5] are presented for all gossip-based algorithms by characterizing them 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 ν_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 ν_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 [25] giving similar bounds. In these papers no details about collision resolution are included, and the algorithms presented require $\omega(1)$ memory size.

A closely related research was done by Chen, Pandurangan, and Hu [7]. They present an aggregate computation algorithm, which they call DRR-gossip algorithm, for Sensor Network. Their algorithm, first, builds a forest over the network, where each root collects the information and computes locally the aggregate function of its tree. Then, a gossiping algorithm is used among the roots to compute the global aggregate function. The authors prove that DRR-gossip algorithm requires $O(n)$ messages and $O(\frac{n^{3/2}}{\log^{1/2} n})$ transmissions to compute the function, using the same number of rounds than the uniform gossip algorithm. But, it reduces the energy consumption by at least a factor of $1/\log n$ over the standard uniform gossip algorithm. Nevertheless, the main difference with the work presented here is framework used for the analysis. The DRR-gossip algorithm is analyzed under a quite stable framework, since collision resolution and failures are not taken into account. We believe that two main challenge for Sensor Network analysis are, precisely, collisions in the communication due to the use of a single radio channel, and their natural instability due to the unreliable nature of the sensor nodes.

Another unstructured protocol for aggregate computations was presented in [8]. A mass-distribution algorithm is also used there, 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 Network of n nodes with underlying graph G with algebraic connectivity $a(G)$[6], where each node i holds a value ν_i and

[5] In gossip-based algorithms interactions occur only in pairs.

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

all nodes start synchronously; then, 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 presented in [8] computes a value ν_i' such that $|\nu_i'-\overline{\nu}| \leq \epsilon$ for all 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 $O(\Delta \log(\sum_i(\nu_i-\overline{\nu})^2/\epsilon^2)/a(G)p)$ *rounds*. Finally, the energy metric, used in that paper, is the expected number of transmissions, 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 [6] to networks with a time-varying connection graph.

Results. The main contribution of this paper is the presentation of an early-stopping protocol that computes the average function in Sensor Networks under a harsh model of sensor restrictions. Although the protocol presented builds incrementally over known techniques, the careful combination of them in the restricted Sensor Network setting is not trivial. More precisely, it is shown here that, in presence of f non-frequent failures, w.h.p., this protocol returns a value and terminates in $O(\Delta + D + f \log^2 n)$ steps which, given that D and Δ cannot be both asymptotically smaller than a polynomial, is optimal if $f \in o(n^c)$ for any $c \in O(1)$. The overall number of transmissions is in $O(n((f+1)\log n + \Delta/\log n + \log \Delta))$ in expectation. On the other hand, 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. Therefore, in presence of failures, we aim 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 to converge, where $\varepsilon > 0$ is the maximum relative error (i.e., the maximum relative difference between two results), Φ_{min} is the minimum conductance [24] of the network underlying the Markov chain characterizing the algorithm, and ν_{max} and ν_{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, the efficiency bounds of the tree algorithm are obtained. Regarding other aggregate functions such as the sum, quantiles or count, they can be computed using a protocol for average without extra cost as described in [25, 8].

All in all, we obtain an energy-efficient algorithm that, even in the hardest model of Sensor Network, and including the construction of the two-level struc-

ture, solves the problem fast. In all our analyses communication costs due to contention resolution are included since we do not assume the existence of any medium access control layer. Furthermore, by proving a matching lower bound, this time is optimal in presence of $f \in o(n^c)$ failures. To the best of our knowledge, this is the first optimal early-stopping algorithm for aggregate computations in Sensor Network.

Roadmap. A lower bound on aggregate computations is proved in Section 2. Upper bounds are shown in Section 3. In Section 3.1 the preprocessing algorithms are detailed. The Aggregate Computation Scheme is presented and each of its phases analyzed in Section 3.2.

2. Lower Bound

A lower bound on the time steps needed to compute an aggregate function in a Sensor Network is shown in the theorem below. Let $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}, n \in \mathbb{N}$ 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 Network of 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 Network of n nodes, where D is the diameter of the network and Δ the maximum degree, under the model described in Section 1, 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.*

PROOF. The lower bound is shown exploiting the adversarial assignment of input-values, the adversarial wake-up and the topology. Consider a Sensor Network of n nodes with maximum degree Δ and diameter D, where some node y is located at $\Omega(D)$ hops of any node in a subset $X, |X| \in \Omega(\Delta)$ of nodes that form a clique. Such a set X exists as proved in the following claim.

Claim 1. Given a geometric graph of n nodes and maximum degree Δ, there exists a subset of nodes S that form a clique such that $|S| \in \Omega(\Delta)$.

PROOF. Let x be a node of degree Δ. Then, there are $\Delta + 1$ nodes located in a circle of radius r centered on x, call this circle C. In order to prove the claim,

it is enough to show that there is a circle of radius $r/2$ inside C that contains at least $\Omega(\Delta)$ nodes. For the sake of contradiction, assume there is no such circle. A constant number of circles of radius $r/2$ are enough to cover completely C. By our assumption, each of these circles contains $o(\Delta)$ of the nodes in C. But then, the total number of nodes in C is in $o(\Delta)$ which is a contradiction.

For the sake of contradiction, assume first that there exists an assignment-oblivious protocol $\mathcal{P}$ that computes a one-node-sensitive function $\mathcal{F}$ over any values assigned to nodes in this network in time $o(D)$. Consider an assignment of values $\vec{\nu}_1$ such that a value $\nu_1(x)$ is assigned to some node $x \in X$ and $\mathcal{F}(\vec{\nu}_1)$ is the value returned to y by $\mathcal{P}$ in $o(D)$ steps. Since $\mathcal{F}$ is sensitive, there exists an assignment $\vec{\nu}_2$ such that $\nu_1(x) \neq \nu_2(x)$ that makes $\mathcal{F}(\vec{\nu}_1) \neq \mathcal{F}(\vec{\nu}_2)$. Consider the execution of $\mathcal{P}$ under this new assignment $\vec{\nu}_2$. It is not possible that a value different than $\mathcal{F}(\vec{\nu}_1)$ is returned to y by $\mathcal{P}$ in $o(D)$ steps under this new assignment because x and y are separated by $\Omega(D)$ hops, which is a contradiction.

Similarly, for the sake of contradiction, assume that there exists an assignment-oblivious protocol $\mathcal{P}$ that computes a one-node-sensitive function $\mathcal{F}$ over any values assigned to nodes in this network in time $o(\Delta)$. In order to compute $\mathcal{F}$, all nodes have to transmit at least once[7]. Consider a node $x \in X$ that is scheduled to transmit last in X by $\mathcal{P}$[8]. Consider now an assignment of values $\vec{\nu}_1$ such that a value $\nu_1(x)$ is assigned to x and $\mathcal{F}(\vec{\nu}_1)$ is the value returned to y by $\mathcal{P}$ in $o(\Delta)$ steps. Since $\mathcal{F}$ is sensitive, there exists an assignment $\vec{\nu}_2$ such that $\nu_1(x) \neq \nu_2(x)$ that makes $\mathcal{F}(\vec{\nu}_1) \neq \mathcal{F}(\vec{\nu}_2)$. Consider the execution of $\mathcal{P}$ under this new assignment $\vec{\nu}_2$. It is not possible that a value different than $\mathcal{F}(\vec{\nu}_1)$ is returned to y by $\mathcal{P}$ in $o(\Delta)$ steps under this new assignment because x is the last node to transmit in a clique of $\Omega(\Delta)$ nodes, which is also a contradiction.

3. 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

[7]Eventually the sink node might only receive values. To disregard this node does not change the analysis.

[8]If $\mathcal{P}$ is randomized, take x to be a node that has a positive probability of transmitting the last in X.

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.

3.1. Preprocessing and Maintenance

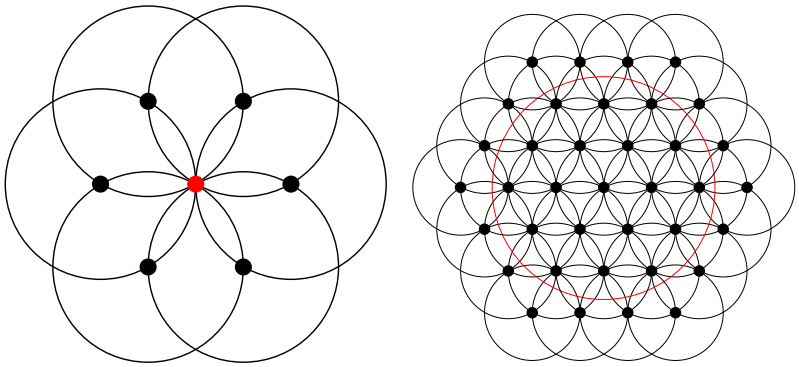
The preprocessing/maintenance algorithm includes two phases. Due to the memory size limitations, in the first phase delegate nodes are defined so that each delegate node is within range of $\Theta(1)$ delegates. Additionally, a second phase establishes schedules of transmissions so that each group delegate-slugs can communicate without colliding with neighboring groups. The first and second preprocessing phases can be implemented as in [14]. We overview here some necessary details.

The first phase of preprocessing is implemented as a distributed maximal independent set (MIS) computation. Members of that set take the role of delegate nodes and the rest become slug nodes. After running this phase using a radius of transmission αr for some $0 < \alpha \leq 1$, no delegate node is within distance αr of another delegate node and every slug node is within distance αr of some delegate node. Furthermore, given these geometric properties and the fact that the hexagonal lattice is the densest of all possible plane packings [15], every slug node is within distance αr of less than 6 delegate nodes (see Figure 1(a)). This phase is triggered when an active node does not receive transmissions from any delegate node. Thus, the preprocessing procedure is by default used to re-build the hierarchy in face of a delegate node failure, at the same cost.

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 [15], that the radius lower bound to achieve connectivity w.h.p. under uniform distribution of nodes is $r \in \Omega(\sqrt{\log n/n})$ [19], the assumption of complete coverage, and the assumption regarding the area of deployment.

Remark 1. Given a Sensor Network of 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.

After becoming a delegate, a node reserves some time slots to be used periodically and exclusively by itself and its slugs, in the second phase of preprocessing. Slug



(a) (b)

Figure 1: Illustration of maximum degree.

nodes can be in range of more than one delegate node. Hence, due to the hidden-terminal problem, there could be collisions of reserved slots at a slug node. In order to avoid that, the value of α in the previous phase is limited to $0 < \alpha \leq 1/4$, whereas the time-slots reservation is performed using the maximum range r. With this modification, when using a bigger range of transmission βr, for $2\alpha \leq \beta \leq 1/2$, any delegate node can send a message to another delegate node in a radius βr without interference from other nodes in a radius r, avoiding the hidden-terminal problem. Given that the hexagonal lattice is the densest of all possible plane packings [15], a geometric calculation gives that there are at most $3\lceil 2\beta/\alpha\sqrt{3}\rceil(\lceil 2\beta/\alpha\sqrt{3}\rceil + 1)$ delegate nodes within distance of βr of any delegate node.

After preprocessing, each delegate node has reserved $b \in \Theta(1)$ time slots with a period of $\gamma \in \Theta(1)$. These slots are reserved to be used by itself and its slugs exclusively within radius r. The actual value of b is chosen accordingly depending on the specific protocol.

Among the reserved b slots (see Figure 2), the first step is used by the delegate to transmit a beacon message. This message allows the slugs to identify which are the incoming – in terms of time – reserved slots. In this way, local synchronism is achieved. Additionally, two steps are reserved for the delegate to communicate with neighboring delegates and one step to broadcast the computation result. Another two slots are reserved for the communication between slugs and the delegate, one slot for slugs transmissions and one slot for delegate acknowledgement. In this manner, collision detection is implemented. The slugs compete for these slots over various rounds of γ slots and the acknowledgement is used to signal

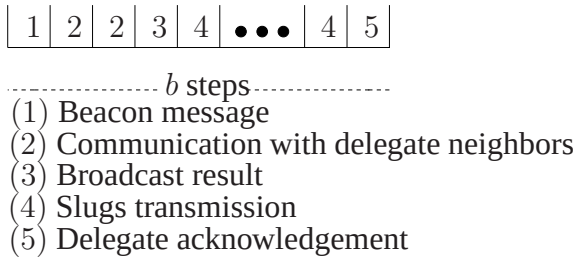


Figure 2: Illustration of time-slots usage.

success.

The following lemma establishes formally the efficiency of these phases. Further details can be found in [14, 13, 12] and the references therein.

Lemma 1. *(a) For any node i running the first phase of preprocessing, for any $0 < \alpha \leq 1$, at least one node within distance αr of i becomes a delegate within $O(\log^2 n)$ time steps and no two delegate nodes are within distance αr 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.*

3.2. The Aggregate Computation Scheme

In the Aggregate Computation Scheme, nodes use only time slots reserved as described in the previous section. Thus, local synchronism, collision detection among slugs and their delegates, and non-colliding transmission schedules among delegates are available. For the sake of clarity, we focus on describing the scheme omitting these details. We also omit the fact that nodes use only the reserved slots for transmissions, since this overhead only introduces a constant factor in the efficiency analysis. Regarding the range of transmission used, throughout the Aggregate Computation Scheme, a slug node uses a radius αr, whereas a delegate node uses a radius βr. For clarity of presentation, the Aggregate Computation Scheme is described assuming that nodes do not fail and later this assumption is removed. Also, in order to obtain worst-case bounds, we assume that all nodes are active.

Before describing the protocol, the following notation is defined. Denote the set of delegate nodes and the set of slug nodes defined in preprocessing as 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 ν_k.

The Aggregate Computation Scheme includes the following four phases. TRIGGER: the sink node broadcasts (τ_1, D, Δ), where τ_1 is the time slot to measure the input-values, D is the diameter of the network and Δ 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. By the definition of a MIS, the sink node is either a delegate node or it is in range of a delegate node. Therefore, the TRIGGER phase can be implemented as follows. If the sink node is not a delegate, using a reserved slot, the sink node transmits the message to one of its delegates. Upon receiving such a message or if the sink node is a delegate node, 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 τ_1. Since only reserved slots are used, the total time taken by this phase is in $O(D)$ and using the Remark 1 the total number of transmissions in this phase is in $O(n/\log n)$. Hence, τ_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 2. *After the sink node starts disseminating the trigger message, all dele-*

gate nodes have received the message within $O(D)$ steps and the overall number of transmissions is $O(n/\log n)$.

Collection Phase. At time τ_1, nodes start running the COLLECTION phase using the input-values at that time step. Before describing the implementation of this phase, we explain the communication primitive used. 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 appropriately chosen appropriately [14]. The intuition of the algorithm is the following. After the window size is in the same order of the number of neighboring nodes, the message of a constant fraction of slugs is received by the delegate in each round w.h.p. Since the reception of those messages is acknowledged by the delegate, these slug nodes do not transmit in future rounds. The final round is included so that transmissions are successful when only $o(\log n)$ messages remain. We refer to this protocol as the *windowed protocol*.

The COLLECTION phase is specified in Algorithm 1. Each slug node $i \in S$ begins this phase choosing one of its delegates to pass its input-value. The reason to do that is 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 ν_i and the ID of the delegate chosen. Given the availability of delegate acknowledgements, a delegate receives exactly one input-value per slug node.

A delegate node running the COLLECTION phase does the following. Each delegate node maintains two magnitudes that we call *sum* and *weight*. For each node $j \in M$, we denote the sum and weight as σ_j and ω_j respectively. Each delegate node j initializes the sum $\sigma_j = \nu_j$ and the weight $\omega_j = 1$. Upon receiving (and acknowledging) the transmission of one of its slug nodes i, the delegate adds the input-value received to the sum and increases the weight. Notice that 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.

Lemma 3. *Let V be the set of n nodes in a Sensor Network, ν_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 τ_2, the following*

Algorithm 1: The COLLECTION phase for the Average Problem.

For slug node $i \in S$:
  choose arbitrarily a delegate node $j \in M(i)$
  using the windowed protocol, transmit message (i, j, ν_i) with radius αr

For delegate node $j \in M$:
  $\sigma_j \leftarrow \nu_j$
  $\omega_j \leftarrow 1$
  for τ_2 *steps* **do**
    if *a message* (i, j, ν_i) *is received* **then**
      $\sigma_j \leftarrow \sigma_j + \nu_i$
      $\omega_j \leftarrow \omega_j + 1$

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 σ_j and ω_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 $O(n(\log n + \log \Delta))$.

PROOF. Using well-known techniques as in [14], it can be proved that, after running the windowed protocol, each delegate node has received the input-value of all its slugs w.h.p., the time taken by the protocol is in $O(\Delta + \log^2 n)$ steps, the number of transmissions of a delegate node during the windowed protocol is in $O(\Delta + \log^2 n)$, and the expected number of transmissions of a slug node is in $O(\log n + \log \Delta)$. Thus, the claim follows from these facts, by definition of the algorithm and using Remark 1.

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 Δ 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 3.2 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 τ_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 τ_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 1). 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 2. *Given a Sensor Network with a set of nodes V running the Aggregate Computation Scheme as described, where Δ 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 τ_1 is the time slot at which the input-values are assigned. Under the model described in Section 1, if the number of node failures after τ_1 is bounded by $f \geq 0$, and T is the minimum time between any pair of consecutive failures. There exist positive constants κ_1, κ_2 such that, if $V' \subseteq V$ is the set of nodes awake in all the interval $[\tau_1 - \kappa_1(D + \log^2 n), \tau_1]$, ν_i is the input-value assigned to node $i \in V'$ at time τ_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 1. 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 1, 2, and 3, given that D and Δ 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 1, 2, 3, Remark 1 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 3. 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* (see Algorithm 2 for details).

Given that the shares are the same for all neighbors and δ and Δ are known[9], 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

[9] δ is a constant that can be stored before deployment. Δ is broadcasted in the TRIGGER phase.

Algorithm 2: Mass Distribution for each delegate node $i \in M$.

$\overline{\nu}_i \leftarrow \sigma_i/\omega_i$
for $k = 1$ *to* τ_3 **do**
  transmit $(i, k, \overline{\nu}_i, \omega_i)$
  $d \leftarrow 0$
  $v \leftarrow 0$
  for γ *time steps* **do**
    if *a message* $(j, k, \overline{\nu}_j, \omega_j)$ *is received and* $j \in N(i)$ **then**
      $d \leftarrow d + \omega_j$
      $v \leftarrow v + \omega_j \overline{\nu}_j$
  $\overline{\nu}_i \leftarrow \overline{\nu}_i + (v - d\overline{\nu}_i)/2\delta\Delta$

reserved slots. Notice that Mass Distribution does 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 3.2, 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 Distribution we assume that delegate-node failures occur between rounds.

We analyze now Mass Distribution[10]. Assume first that nodes do not fail. Given that the fraction shared in Mass Distribution is round independent, the algorithm can be characterized by a matrix of shares as follows. Let $\vec{\overline{\nu}}^{(t)} = (\overline{\nu}_1^{(t)} \ldots \overline{\nu}_n^{(t)}), i \in V$ be the vectors[11] 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 $r \in N(s)$, $p_{ij} = 1 - (\sum_{s \in N(r)} \omega_s)/2\delta\Delta$

[10]We assume familiarity with Markov chains and spectral graph theories. For an introduction refer to [16, 10].

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

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 [33]. We establish formally the correctness of the average computation in the following lemma.

Lemma 4. *(Correctness) Let V be the set of n nodes in a Sensor Network, ν_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 $\varepsilon > 0$.*

PROOF. The Markov chain $\mathbf{X}$ characterizing Mass Distribution is finite and irreducible. Additionally, given that the underlying graph has self-loops, the g.c.d. of all closed walks is 1. Therefore, $\mathbf{X}$ is aperiodic. Then, by the fundamental theorem of Markov chains [33], $\mathbf{X}$ is ergodic and it has a unique stationary distribution. Since $\vec{P}$ is doubly stochastic, the system $\vec{\pi} = \vec{\pi}\vec{P}$ admits the solution $\vec{\pi} = (1/n \ldots 1/n)$ which, by the aforementioned theorem, is unique. Let $\vec{\mu}^{(t)}$ be the distribution at round t. Given that the chain converges to the stationary distribution, we know that, for each $\varepsilon > 0$, there is a $\tau \geq 0$ such that, for all $t \geq \tau$, $|\mu_i^{(t)} - \pi_i|/\pi_i \leq \varepsilon$, for all $i \in V$. Then, for any initial distribution $\vec{\mu}^{(0)}$ and for all $t \geq \tau$ it holds that, $(1-\varepsilon)/n \leq \sum_{i \in V} \mu_i^{(0)} (\vec{P}^t)_{ij} \leq (1+\varepsilon)/n$, for all $j \in V$. But then, it must also hold that $(1-\varepsilon)/n \leq (\vec{P}^t)_{ij} \leq (1+\varepsilon)/n$, for all $i, j \in V$. To see why, assume for the sake of contradiction that there is a pair $i', j' \in V$ such that $(\vec{P}^t)_{i'j'} < (1-\varepsilon)/n$ or $(\vec{P}^t)_{i'j'} > (1+\varepsilon)/n$. Then, it would be enough to set $\mu_{i'}^{(0)} = 1$ and the rest of the components to 0 to make the previous assertion false. For any node $i \in V$, $\overline{\nu}_i^{(t)} = (\vec{\overline{\nu}}^{(0)} \vec{P}^t)_i = \sum_j \overline{\nu}_j^{(0)} (\vec{P}^t)_{ji}$. Then, for all $t \geq \tau$, $(1-\varepsilon) \sum_j \overline{\nu}_j^{(0)}/n \leq \overline{\nu}_i^{(t)} \leq (1+\varepsilon) \sum_j \overline{\nu}_j^{(0)}/n$. Given that $\sum_j \overline{\nu}_j^{(0)} = \sum_j \nu_j$, then

$\sum_j \overline{\nu}_j^{(0)}/n = \overline{\nu}$. Then, we have that for all $t \geq \tau$, $\overline{\nu}(1-\varepsilon) \leq \overline{\nu}_i^{(t)} \leq \overline{\nu}(1+\varepsilon)$. Thus, $|\overline{\nu}_i^{(t)} - \overline{\nu}|/\overline{\nu} \leq \varepsilon$ for all $i \in V$ and the claim follows.

In presence of node failures, the fraction of average shared in Mass Distribution is not round independent. Therefore, instead, we characterize the computation carried out by Mass Distribution under failures as a sequence of matrices $\vec{P}_0, \vec{P}_1, \vec{P}_2, \ldots$ 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 [38]. 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. The details are presented in Lemma 5.

Lemma 5. *Given a Sensor Network with 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 Distribution as 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 $\Phi_{min} = \min_{k \in \{0,1,\ldots,f\}} \Phi_k$ where Φ_k is the conductance of the underlying graph after the kth failure, all nodes have converged to a value with relative error $0 < \varepsilon < 1$.*

PROOF. We show first the convergence time to achieve a multiplicative reduction factor $0 < \rho < 1$ with respect to the initial range of values. Let $\overline{\nu}_{max}^{(t)} = \max_i \overline{\nu}_i^{(t)}$ and $\overline{\nu}_{min}^{(t)} = \min_i \overline{\nu}_i^{(t)}$ for all $t \geq 0$.

Claim 2. After $\tau_4 \in O((f + \log(1/\rho))/\Phi_{min}^2)$ rounds of Mass Distribution, where $0 < \rho < 1$ is a parameter, f is the number of node failures, and $\Phi_{min} = \min_{k \in \{0,1,\ldots,f\}} \Phi_k$, where Φ_k is the conductance of the graph underlying the Markov chain characterizing the algorithm after the kth failure. It holds for all $t \geq \tau_4$ that:

$$\overline{\nu}_{max}^{(t)} - \overline{\nu}_{min}^{(t)} \leq \rho(\overline{\nu}_{max}^{(0)} - \overline{\nu}_{min}^{(0)}).$$

PROOF. Given a Markov chain $\mathbf{X}_k$ with transition matrix $\vec{P}_k$, let $\vec{\pi}_k$ be the unique stationary distribution and $(\lambda_k)_1$ the second largest eigenvalue of $\vec{P}_k$. From Proposition 3.1 in [38], using the fact that the stationary distribution is uniform and that $\forall i \in V_k, (\vec{P}_k)_{ii} \geq 1/2$, we know that $\forall t > 0, \max_{i,j \in V_k}\{|(\vec{P}_k^t)_{ij} - (\vec{\pi}_k)_j|\} \leq (\lambda_k)_1^t$. Let $\Delta\overline{\nu}_k$ be the range of averages held by delegates at the time of failure k, and t_k be the number of rounds between failures k and $k+1$. Then, $\Delta\overline{\nu}_{k+1} \leq \rho_k \Delta\overline{\nu}_k$ where $\rho_k \leq 2(\lambda_k)_1^{t_k}$, since removing one value can not increase the range. Given that we want to obtain a reducing factor of ρ, after f failures we want $\rho_0\rho_1 \ldots \rho_f \leq \rho$. Replacing, we want

$$\begin{aligned} 2(\lambda_0)_1^{t_0} 2(\lambda_1)_1^{t_1} \ldots 2(\lambda_f)_1^{t_f} &\leq \rho \\ 2^f \left((\lambda_{max})_1\right)^t &\leq \rho, \\ \text{where } (\lambda_{max})_1 = \max_{k \in \{0,1,\ldots,f\}} &(\lambda_k)_1. \end{aligned}$$

Using that $1 - x \leq e^{-x}, 0 < x < 1$ [32, §2.68], we obtain the minimum number of rounds to achieve that reducing factor as

$$t \geq \frac{1}{1 - (\lambda_{max})_1} \left(f \ln 2 + \ln \frac{1}{\rho} \right).$$

Bounding the second largest eigenvalue using the conductance as in [38]

$$t \geq \frac{2}{\Phi_{min}^2} \left(f \ln 2 + \ln \frac{1}{\rho} \right).$$

Removing some values from the initial range of values (the values of delegates that fail) can not increase that range. Thus, the claim follows.

The range of averages held by delegate nodes at the beginning of the COMPUTATION phase is $\overline{\nu}_{max}^{(0)} - \overline{\nu}_{min}^{(0)} \leq \nu_{max} - \nu_{min}$. Then, after running the Mass Distribution algorithm for t rounds to achieve a reduction factor ρ, the range of averages held by delegate nodes is $\overline{\nu}_{max}^{(t)} - \overline{\nu}_{min}^{(t)} \leq \rho(\nu_{max} - \nu_{min})$. Since the result to which delegates converge after f failures cannot be smaller than ν_{min}, in order to achieve a relative error ε, we want $\rho(\nu_{max} - \nu_{min})/\nu_{min} \leq \varepsilon$. Replacing in Claim 2, the Lemma is proved.

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

Theorem 3. *Given a Sensor Network with a set of nodes V running the Aggregate Computation Scheme as described, where Δ 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 τ_1 is the time slot at which the input-values are assigned. Under the model described in Section 1, if the number of node failures after τ_1 is bounded by $f \geq 0$, and T is the minimum time between any pair of consecutive failures. There exist positive constants κ_1, κ_2 such that, if $V' \subseteq V$ is the set of nodes awake in all the interval $[\tau_1 - \kappa_1(D + \log^2 n), \tau_1]$, ν_i is the input-value assigned to node $i \in V'$ at time τ_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 Φ_k is the conductance of the underlying graph after the kth 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 1, 2, 3 and, given that each round takes $O(1)$ time steps, Lemma 5, and using that D and Δ can not be in $o(\log^3 n)$ simultaneously. The claimed number of transmissions follows from Lemmas 1, 2, 3, 5, and Remark 1.

4. Conclusions

In this paper, we have presented an early-stopping algorithm that computes the average function in a weak model of Sensor Network. The algorithm behavior was analyzed assuming an unstable framework, in which collisions can occur during communication due to the use of a single radio channel, and in which sensor nodes can fail during the computation. The presented algorithm first structures the Sensor Network into a two-level hierarchy, partitioning the sensors into delegates and slug nodes.

In the computation, each delegate node represents its neighborhood of slug nodes, and can communicate with the neighboring delegates without collisions. Then, once the delegates collect the values of the slugs, they run the main part of the algorithm to compute the average. Finally, they provide their slugs with the computed average. The algorithm run by the delegates combines two types of computation, a tree-based and a mass-distribution computation. The algorithm

uses the speed and simplicity of the tree-based computation to quickly obtain the average value if failures are not frequent. On the other hand, the algorithm uses the robustness of the mass-distribution process to obtain a result in the presence of frequent failures.

The combination of both algorithms yields a protocol that is proven to complete and disseminate the aggregate computation in $O(\Delta + D + f \log^2 n)$ time steps if there are at most $f \geq 0$ node failures distanced by time in $\Omega(\log^2 n)$. Otherwise, after $O(\Delta + D + (f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min}))/\Phi^2_{min})$ time steps, all nodes converge to the same result with relative error $0 < \varepsilon < 1$ w.h.p., where $\Phi_{min} = \min_{k \in \{0,1,\ldots,f\}} \Phi_k$, and Φ_k is the conductance of the underlying graph after the kth failure.

As established in Theorem 1, $\Omega(\Delta + D)$ is a lower bound for this problem. Thus, it is pertinent to observe the impact of the remaining additive terms in the proven upper bounds. Given that D and Δ cannot be both asymptotically smaller than a polynomial on n, the algorithm presented in this paper is optimal if $f \log^2 n \in o(n^c)$, for any $c \in O(1)$, and failures are not frequent as defined in Theorem 2. Furthermore, the presented algorithm is also optimal in general if $(f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min}))/\Phi^2_{min} \in o(n^{c'})$, for any $c' \in O(1)$, although in this case the bound is w.h.p.

Regarding the impact of the conductance in the latter bound, recall that we use the conductance only to bound the second largest eigenvalue of the graph underlying the mass-distribution algorithm as in [38]. As elucidated in [17], the second largest eigenvalue of a 2-dimensional mesh is in $\Theta(1/n)$. This type of network is relevant to gain intuition on how the algorithm might behave on an average Sensor Network, because it resembles a uniform distribution of nodes over an area without bottlenecks[12]. Using the bound on the number of delegate nodes established in Remark 1, the second largest eigenvalue of a 2-dimensional mesh is in $\Omega(\log n/n)$ if only delegate nodes are taken into account. Re-creating the analysis in Lemma 5 using this more specific bound, the additive term for arbitrarily spaced failures becomes $(f + \log(1/\varepsilon) + \log(\nu_{max}/\nu_{min}))/(1 - \log n/n)$. Given that $n/(n - \log n) \in o(\log n)$, it can be seen that if $1/\varepsilon$ and ν_{max}/ν_{min} are at most polynomial on n, the mass-distribution algorithm may be even asymptotically faster than the tree algorithm if the network does not have bottlenecks.

[12] A popular model of sensor nodes deployment is the Random Geometric Graph, where nodes are deployed uniformly at random over a unit circle.

References

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, I, November 2003.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cyirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.

[3] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45:104–126, 1992.

[4] 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.

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

[6] 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.

[7] Jen-Yeu Chen, Gopal Pandurangan, and Jianghai Hu. Brief announcement: locality-based aggregate computation in wireless sensor networks. In *PODC '09: Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 298–299, New York, NY, USA, 2009. ACM.

[8] 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.

[9] B. S. Chlebus. Randomized Communication in Radio Networks. *Handbook on Randomized Computing*, 1:401–456, 2001.

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

[11] A. G. Dimakis, A.D. Sarwate, and M.J. Wainwright. Geographic gossip: Efficient averaging for sensor networks. *IEEE Trans. Signal Processing*, 56(3):1205–1216, 2008.

[12] M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Bootstrapping a hop-optimal network in the weak sensor model. *ACM Transactions on Algorithms*, 5(4), 2009.

[13] 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.

[14] 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.

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

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

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

[18] 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.

[19] 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.

[20] John S. Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *SOSP*, pages 146–159, 2001.

[21] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[22] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John S. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS*, pages 457–458, 2002.

[23] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[24] 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.

[25] 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.

[26] 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.

[27] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *ICDCS Workshops*, pages 575–578. IEEE Computer Society, 2002.

[28] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proc. of the 10th ACM Ann. Intl. Conference on Mobile Computing and Networking*, 2004.

[29] Philip Levis, Eric Brewer, David Culler, David Gay, Samuel Madden, Neil Patel, Joe Polastre, Scott Shenker, Robert Szewczyk, and Alec Woo. The emergence of a networking primitive in wireless sensor networks. *Commun. ACM*, 51(7):99–106, 2008.

[30] 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.

[31] Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 49, 2002.

[32] D. S. Mitrinović. *Elementary Inequalities*. P. Noordhoff Ltd. - Groningen, 1964.

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

[34] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 250–262, 2004.

[35] 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.

[36] P. Rentala, R. Musumuri, U. Saxena, and S. Gandham. Survey on sensor networks. http://citeseer.nj.nec.com/479874.html.

[37] L. G. Roberts. Aloha Packet System with and without Slots and Capture. *Computer Communication Review*, 5(2):28–42, 1975.

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

[39] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proc. of the 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications*, pages 139–148, 2003.