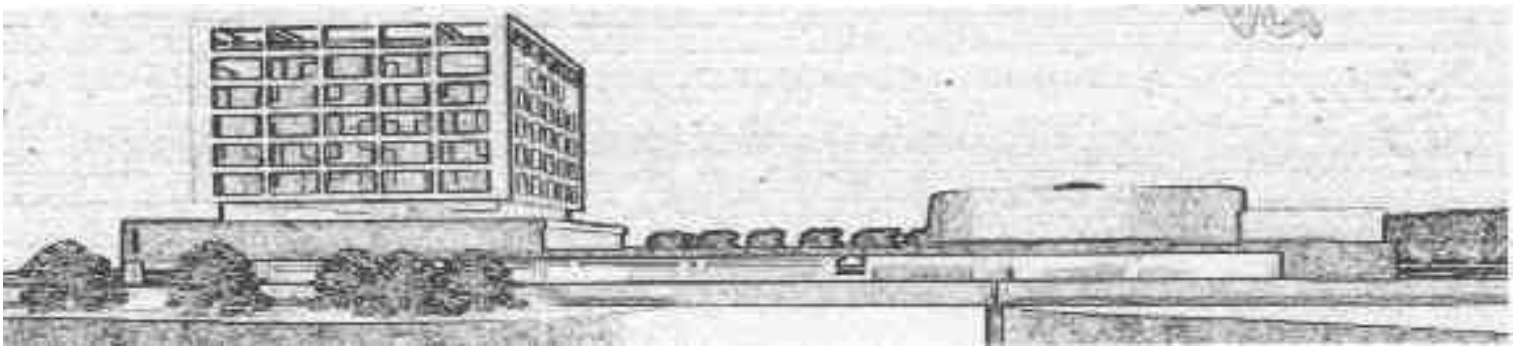
VOLUME VIII, NUMBER 3



REPORTS ON SYSTEMS AND COMMUNICATIONS

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

Antonio Fernández Anta, Miguel A. Mosteiro and Christopher Thraves

Móstoles (Madrid), May 2008
Depósito Legal: M-50653-2004
ISSN: 1698-7489

Table of Contents

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

Antonio Fernández Anta[1], Miguel A. Mosteiro[1,2], and Christopher Thraves[3]

LADyR, GSyC, Univ. Rey Juan Carlos, 28933 Móstoles, Madrid, Spain
anto@gsyc.es,miguel.mosteiro@urjc.es
Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA
mosteiro@cs.rutgers.edu
Universite Bordeaux I, LaBRI, domaine Universitaire, 351 cours de la Libération, 33405 Talence, France
cbthraves@gmail.com

Abstract. In this paper, we study algebraic aggregate computations in Sensor Networks. The main contribution is the presentation of a time-optimal early-stopping protocol that computes the average function under the harsh Weak Sensor Model. The approach followed saves time and energy by relaying the computation on a small network of *delegate* nodes that can be rebuilt fast in face of failures. It is shown that, in a failure-free setting, w.h.p. [1], this protocol returns the exact value and terminates in $O(D+\Delta)$ steps, which is also shown to be optimal, and the overall number of transmissions is in $O(n(\log n + \Delta/\log n + \log \Delta))$ [2] in expectation. On the other hand, in presence of failures, the protocol computes the average of the input-values of a subset of nodes that depends on the failure model. More precisely, it is shown that, after the last node fails and w.h.p., the protocol takes an extra additive factor of $O(\log(n/\varepsilon)/\Phi^2)$ in time and an extra additive factor of $O(n\log(1/\varepsilon)/\Phi^2)$ in the expected number of transmissions, where $\varepsilon > 0$ is the maximum relative error, and Φ is the conductance of the network of delegates. Other aggregate computation algorithms can be easily derived from this protocol.

1 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 equiped 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. Thus, due mainly to the energy constraint, individual sensor nodes are unreliable. Additionally, deterministic deployment of sensors is not feasible because Sensor Networks

[1] We say that a parameterized event E_p occurs *with high probability*, or *w.h.p.* for short, if for any constant $\gamma > 0$ there exists a valid choice of parameter p such that $Pr\{E_p\} \geq 1 - n^{-\gamma}$.

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

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 [1, 2, 31].

A natural question is which problems that are useful for monitoring purposes can be solved in a Sensor Network. Sensors 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 [4, 5, 21]. Popular examples of a 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.

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 to the desired value.

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 more 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 correspond those input-values. This fact, implies that any protocol has to achieve some form of global synchronization. Similarly, 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 [3] 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*[3] nodes

[3] An *active* node at time t is a node that it is up and running at time t.

in each time step. However, for any Sensor Network, it is easy to give a node-failure schedule that maintains such connectivity but partitions the network[4].

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 [22, 24]. 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 [4, 5, 21]. The approach of choice is to aggregate the information at *every* node of the network in a *mass-distribution* fashion as in load balancing [13, 30] 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. However, it is known[5] that mass-distribution algorithms yield the wrong result if those failures are arbitrary.

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 in a failure-free setting. If the presence 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 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*[6].

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 the delegate nodes collect the sensed input-values from the slug nodes.

[4] E.g.: consider the set of nodes partitioned in two connected components that are powered off by the adversary in odd and even steps respectively.

[5] *"If nodes crash during the computation, then our results do not carry over." [21]*

[6] An algorithm is early stopping if it works more efficiently in a failure-free execution than in an execution with failures.

After that, delegate nodes are able to communicate in a collision-free fashion. Second, energy is saved because the slug nodes can idle during the computation. Third, the subnetwork of delegate nodes has constant degree, which allows to easily build a constant-degree spanning tree. 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). Notice that, in presence of failures, the two-level structure may have to be reconstructed; fortunately, 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 reachability of nodes with the *Geometric Graph Model*[7], 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[8]. As customary in the Sensor Networks literature, we assume that nodes are deployed densely enough to ensure network connectivity and sensing coverage. Thus, a straightforward application of [15] gives a bound of $r \in \Omega(\sqrt{\log n/n})$ to achieve connectivity w.h.p. [9]

We assume that the area covered by a sensor node coincides with the range of transmission. Otherwise, the analysis can be easily augmented with a sensing radius. 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.

Regarding models of sensor node constraints, we use the Weak Sensor Model [8], a harsh and comprehensive model that summarizes the literature on sensor node restrictions taking the most restrictive choices when possible. In this model, the communication among neighboring nodes is through broadcast on a *shared channel*. A node receives a message if and only if exactly one of its neighbors transmits. 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*. Time is assumed to be slotted and all nodes have the same clock frequency, but no global synchronizing mechanism is assumed. Nodes are woken up by an adversary, perhaps at different times. Sensor nodes may store only a constant number of $O(\log n)$ [10] bit words. 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, i.e., nodes can fail by stopping due to lack of power supply. However they can restart once their batteries have been reloaded. Hence the failure model is crash-recovery. Other restrictions include: short transmission range ($r << 1$), only one shared channel of communication and lack of position information.

We further assume that the assignment of input-values is adversarial. Without loss of generality, we assume those values to be positive. We also assume that nodes are

[7] A.k.a. *Unit Disk Graph* when the radius is normalized instead of the area.

[8] Notice that, in contrast with the popular random geometric graph model, we do not restrict ourselves to a uniform distribution of nodes or a square area of deployment.

[9] We say that a parameterized event E_p occurs *with high probability*, or *w.h.p.* for short, if for any constant $\gamma > 0$ there exists a valid choice of parameter p such that $Pr\{E_p\} \geq 1 - n^{-\gamma}$.

[10] Througout this paper, $\log$ means $\log_2$ unless otherwise stated.

assigned a unique ID of $O(\log n)$ bits and they know 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. 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.

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 [22], lack of position information [7, 14] and limited range of transmission [19, 21]. Others, are purely experimental [16–18, 23–25, 29, 33].

A hierarchical approach to aggregate information is presented in [14]. 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 $O(n \log^2 n)$ and $O(\log^2 n)$ message and *rounds* complexity respectively. Contention resolution and other communication issues are assumed to be resolved by other underlying protocols.

Generic non-hierarchical gossip-based[11] protocols for average computations in arbitrary networks were studied in [4,21]. Results in [4] 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 $O(\log n+\log(n/\epsilon)/(1-\lambda_{max}((\boldsymbol{I}+\boldsymbol{P})1/2)))$ *rounds*, each node i running a gossip-based algorithm characterized by the matrix $\boldsymbol{P}$, 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 [21] giving similar bounds. In both papers, no details about collision resolution or routing are included, and both require a $\omega(1)$ memory size.

Another unstructured protocol for aggregate computations was presented in [5]. 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 $a(G)$[12], where each node i holds a value ν_i and 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 [5] computes a value ν_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_g can be tuned to improve that bound to $O(\Delta \log(\sum_i(\nu_i-\overline{\nu})^2/\epsilon^2)/p_g p_s a(G))$ *rounds*, where p_s is a probability that depends also on the deployment topology. 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. **Results.** The main contribution of this paper is the presentation

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

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

of a time-optimal early-stopping protocol that computes the average function in Sensor Networks under the harsh Weak Sensor Model. More precisely, it is shown here that, in a failure-free setting, w.h.p., this protocol returns the exact value and terminates in $O(D + \Delta)$ steps, which is also shown to be optimal, and the overall number of transmissions is in $O(n(\log n + \Delta/\log n + \log \Delta))$ in expectation. On the other hand, in presence of failures, the protocol computes the average of the input-values of a subset of nodes that depends on the failure model. More precisely, it is shown that, after the last node fails and w.h.p., the protocol takes an extra additive factor of $O(\log(n/\varepsilon)/\Phi^2)$ in time and an extra additive factor of $O(n \log(1/\varepsilon)/\Phi^2)$ in the expected number of transmissions, where $\varepsilon > 0$ is the maximum relative error, and Φ is the conductance [20] of the network of delegates. (E.g., for the unit square $\Phi = \Omega(\log n/rn)$.)

Most of the previous works for the aggregation problem in Sensor Networks do not take into account communication issues, make strong assumptions regarding node resources, or do not give time or energy analyses. Among those papers that do analyze efficiency, the bounds presented in [5] include among other parameters the initial distribution of input-values, which introduces an unbounded factor in the worst case (recall that our analysis is a worst case scenario regarding such distribution). On the other hand, the protocol in [14] requires position estimation hardware. The protocols in [4, 21] give the same asymptotic bounds in the number of rounds than ours in presence of failures, but both are not early-stopping, both require $\omega(1)$ memory size, and [21] requires a clique topology. The analysis of all these protocols do not take collisions into account.

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 [5, 21]. Therefore, in this paper, the attention is focused in computing the average of the input-values.

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 structure, solves the problem fast[13]. Furthermore, by proving a matching lower bound, this time is optimal in absence of failures. To the best of our knowledge, this is the fist 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. Finally, the overall efficiency of the protocol is shown in Section 3.3. The details of some proofs as well as the figures are left to the appendix for brevity.

2 Lower Bound

In this section we present a lower bound on the time steps needed to compute an aggregate function in a Sensor Network. The following definitions will be useful for this

[13] 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.

purpose. For the sake of brevity, the details of the proof that uses the adversarial assignment of input-values and the topology, is left to the appendix.

Definition 1. *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 $\boldsymbol{\nu}_1 \in \mathbb{R}^n$, there exists another choice of values $\boldsymbol{\nu}_2 \in \mathbb{R}^n$ such that $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ differ only in one value, and $\mathcal{F}(\boldsymbol{\nu}_1) \neq \mathcal{F}(\boldsymbol{\nu}_2)$.*

Definition 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 restrictions of the Weak Sensor Model, 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.*

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 preprocesing 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 face 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 [27] and [10] respectively. We overview here some necessary details.

The first phase of preprocessing is implemented as a distributed maximal independent set[14] (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

[14] An *independent set* is a set of vertices in a graph such that for any pair of vertices, there is no edge between them. An independent set is *maximal* if no more vertices can be added to the set and still be independent.

these geometric properties and the fact that the hexagonal lattice is the densest of all possible plane packings [11], every slug node is within distance αr of less than 6 delegate nodes (see Figure 2(a) in the appendix). 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 [11], the radius lower bound to achieve connectivity w.h.p. under uniform distribution of nodes is $r \in \Omega(\sqrt{\log n/n})$ [15], and the assumption of complete coverage.

Remark 1. Given a Sensor Network of n nodes deployed as a geometric graph, 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. Slave nodes can be in range of more than one delegate node. Hence, due to the hidden-terminal problem[15], there could be collision 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 $2\alpha \leq \beta \leq 1/2$, delegate nodes are in range of other delegate nodes. Nonetheless, given that the hexagonal lattice is the densest of all possible plane packings [11], 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 3 in the appendix), the first step is used by the delegate to transmit a beacon message. This message allows the slugs to identify which are the incoming 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 success.

The following lemma establishes formally the efficiency of these phases. Further details can be found in [8–10] 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*

[15] The *hidden-terminal problem*, a well-known problem in wireless networks, occurs when node x is in range of nodes y and z, but y and z are not in range of each other.

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, we describe the Aggregate Computation Scheme assuming that nodes do not fail. In Section 3.3, we remove this assumption. 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 varios phases of the Aggregate Computation Scheme can be broadly described as in Algorithm 1 in the appendix. In the following sections, we detail the implementation of each of these phases.

TRIGGER *Phase.* By 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 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 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. Nodes communicate in this phase using the following well-known procedure. In each round, nodes choose uniformly at random a slot within a window of slots to transmit its message. 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 according with the specific application. 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 2 in the appendix. 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. The proof uses well-known techniques and the details are left to the appendix for brevity.

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 holds. 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. The time taken by the algorithm is in $O(\Delta + \log^2 n)$.*

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. We describe the two approaches used in the following phases separately for clarity. Although, as explained before, they are run simultaneously. Two different time slots to communicate among delegates were reserved for this purpose. 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. Also for the sake of clarity, we describe both approaches assuming that nodes are activated early enough to receive the trigger and stay active long enough to receive the result of the computation. In Section 3.3 we specify the overall efficiency including the case where this is not true.
Tree-based Algorithm. 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 or a delegate node at one hop of the sink node. 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 aggregates the input-values by passing to its parent in the tree its sum and weight aggregated with the sum and weight of its children. Thus, the root of the tree receives the total sum and weight of the whole network and computes the 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. Given that broadcast and convergecast is run in reserved slots the time taken by the last two phases is just $O(D)$ and by Remark 1 the total number of transmissions is in $O(n/\log n)$. The following lemma formalizes these bounds.

Lemma 4. *Let* M *be the set of delegate nodes and* σ_i *and* ω_i *be the sum and weight of node* $i \in M$ *respectively obtained after running the* COLLECTION *phase as described. After running the* COMPUTATION *and* DISSEMINATION *phases implemented with a tree-based algorithm as described, within* $O(D)$ *steps all nodes have received the value* $\sum_{i\in M} \sigma_i / \sum_{i\in M} \omega_i$ *and the overall number of transmissions in both phases is in* $O(n/\log n)$.

Mass-distribution Algorithm. 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, $\max_{i\in M}\{|N(i)|\} \leq 3\lceil 2\beta/\alpha\sqrt{3}\rceil(\lceil 2\beta/\alpha\sqrt{3}\rceil + 1)$ as shown in Section 3. So, fix $\delta = 1 +$

$3\lceil 2\beta/\alpha\sqrt{3}\rceil(\lceil 2\beta/\alpha\sqrt{3}\rceil+1)$. In each round, each delegate node passes a fraction $1/2\delta$ of its sum and weight to each neighboring delegate node and keeps the rest for itself. The delegate nodes update their sum and weight values with the shares received and repeat. After enough number of iterations, all sum and weight values converge to a pair of values whose ratio is the average sought. We call this protocol *Stingy Share* (see Algorithm 3 in the appendix for details).

Given that the shares are the same for all neighbors and δ is known, delegate nodes do not need to specify the destination and simply transmit the sum and weight. After enough number of rounds, each delegate node i can compute the average with the accuracy desired as the ratio σ_i/ω_i. Furthermore, the DISSEMINATION phase is integrated in this phase by default given that, although sums and weights are transmitted to neighboring delegate nodes, all neighboring nodes receive those transmissions because they are produced in reserved slots. Notice that Stingy Share does not violate the memory restrictions since only a constant number of values are received in each round and the sum and weight values are still polynomially upper bounded. Of course, precision limitations due to real number computations are still in order[16].

We analyze now Stingy Share and prove its correctness[17]. Given that the fraction of σ and ω shared in Stingy Share is round independent, the algorithm can be characterized by a matrix of shares as follows. For the sake of clarity, without loss of generality assume that delegate nodes IDs are in $\{1,\ldots,|M|\}$. Let $\boldsymbol{\sigma}^{(t)} = (\sigma_1^{(t)} \ldots \sigma_{|M|}^{(t)})$ and $\boldsymbol{\omega}^{(t)} = (\omega_1^{(t)} \ldots \omega_{|M|}^{(t)})$ be the vectors[18] of sums and weights of all delegate nodes after round t. Let $\boldsymbol{P} = (p_{ij})$ be a matrix in $\mathbb{R}^{|M|\times|M|}$ such that $p_{ij} = 1/2\delta$ if $j \in N(i)$, $p_{ij} = 1 - |N(i)|/2\delta$ if $j = i$, and $p_{ij} = 0$ otherwise. Then, $\boldsymbol{\sigma}^{(t)} = \boldsymbol{\sigma}^{(0)}\boldsymbol{P}^t$ and $\boldsymbol{\omega}^{(t)} = \boldsymbol{\omega}^{(0)}\boldsymbol{P}^t$ are the vectors of sums and weights in round t respectively. Given that $\boldsymbol{P}$ is stochastic, this characterization can be also seen as a Markov chain $\mathbf{X} = \{\mathbf{X}_t\}$ where the state space is M and the transition matrix is $\boldsymbol{P}$.

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 pointwise distance*, which is defined as $\max_i |\nu_i - \overline{\nu}|/\overline{\nu}$. The following lemma, showing the correctness of Stingy Share, can be proved using the fundamental theorem of Markov chains [28]. The details are left to the appendix for brevity.

Lemma 5. *(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 M be the set of delegate nodes defined in the preprocessing of the Aggregate Computation Scheme. Let $\sigma_i^{(t)}$ and $\omega_i^{(t)}$ be the sum and weight of delegate node $i \in M$ obtained t rounds after the* COLLECTION *phase of the Aggregate Computation Scheme. Then, implementing the* COMPUTATION *phase using Stingy Share, there exists a $\tau_3 \geq 0$ such that, for all $t \geq \tau_3$, $|\overline{\nu} - \sigma_i^{(t)}/\omega_i^{(t)}|/\overline{\nu} \leq \varepsilon$, for all $i \in M$ and for a given parameter $\varepsilon > 0$.*

[16] Were precision requirements more relevant than memory restrictions, nodes could simply avoid dividing by 2δ in each round, compute the result as a rational number, and the division could then be performed by the powerful sink nodes only.

[17] We assume familiarity with elementary Markov chains and spectral graph theories. For an introduction refer to [6, 12].

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

To analyze the efficiency of Stingy Share, we leverage the vast body of research work on bounding the rate of convergence of Markov chains to a stationary distribution, i.e., the *mixing time*. For Markov chains with underlying graphs with geometric properties, there is a natural notion useful to bound λ_1, called *conductance* [20]. Consider the geometric graph $G_M = \{M, E_M\}$, underlying the Markov chain $\mathbf{X}$, where $\{i, j\} \in E_M$ if and only if i and j are located within an Euclidean distance of at most r. Let the *weight* of an edge $\{i, j\}$ in the underlying graph G_M be $w_{ij} = p_{ij}\pi_i = p_{ji}\pi_j$. Then, the conductance of $\mathbf{X}$ with underlying graph G_M is $\Phi(G_M) \triangleq \min\{\sum_{i\in C, j\notin C} w_{ij} / \sum_{i\in C} \pi_i\}$, where the minimization is over all subsets $C \subset M$ such that $0 < \sum_{i\in C} \pi_i \leq 1/2$. Then, the conductance can be seen as the minimum, taken over all state subspaces, of the conditional probability that the chain in stationarity moves out of a state subspace given that it is there. We bound the mixing time using the conductance as in [32]. (For the sake of brevity, we leave the details to the appendix.) The following lemma, which can be derived from Lemma 7 and Remark 1, establishes formally the efficiency of the last two phases implemented as a mass-distribution algorithm.

Lemma 6. *Let M be the set of delegate nodes and σ_i and ω_i be the sum and weight of node $i \in M$ respectively obtained after running the* COLLECTION *phase as described. After running the* COMPUTATION *and* DISSEMINATION *phases implemented with a mass-distribution algorithm as described, within $O(\log(n/\varepsilon)/\Phi^2)$ steps all nodes have received the value $\sum_{i\in M} \sigma_i / \sum_{i\in M} \omega_i$ with maximum relative error ε and the overall number of transmissions in both phases is in $O(n \log(1/\varepsilon)/\Phi^2)$.*

3.3 Overall Efficiency of the Average Protocol

Theorem 2. *Given a Sensor Network of n nodes, where Δ is a tight upper bound on the maximum number of neighbors of any node and D the diameter of the network. Let τ_1 be the time step at which the average of the input-values of sensor nodes has to be computed. Under the restrictions of the Weak Sensor Model, the following holds. (a) There exist constants $\kappa_1 > 0$ and $\kappa_2 > 0$ such that, if the set of active nodes does not change in the interval $[\tau_1 - \kappa_1(D + \log^2 n), \tau_1 + \kappa_2(D + \Delta)]$ then, the Aggregate Computation Scheme solves the Average Problem within $O(\Delta + D)$ time steps w.h.p. and the expected number of transmissions is in $O(n(\log n + \Delta/\log n + \log \Delta))$ w.h.p. (b) Otherwise, the Aggregate Computation Scheme solves the Average Problem over a subset of nodes, that depends on the failure model, with maximum relative error ε over the input-values of the nodes in that subset, within $O(\Delta + D + \log(n/\varepsilon)/\Phi^2)$ time steps, after the last failure, w.h.p., and the expected number of transmissions is in $O(n(\log n + \Delta/\log n + \log \Delta + \log(1/\varepsilon)/\Phi^2))$, after the last failure, w.h.p.*

Proof. (a) By Lemmas 1 and 2, the total time of preprocessing and the TRIGGER phase is in $O(D + \log^2 n)$ w.h.p. Therefore, for large enough κ_1, nodes active in $[\tau_1 - \kappa_1(D + \log^2 n), \tau_1 + \kappa_2(D + \Delta)]$ receive the trigger message on time to participate in the protocol w.h.p. By Lemmas 3, 4, and given that D and Δ can not be in $o(\log^3 n)$ simultaneously, these nodes complete the COLLECTION, COMPUTATION and DISSEMINATION phases in $O(D + \Delta)$ steps. Thus, the claimed time follows. On the other hand, the claimed number of transmissions is a direct consequence of Lemmas 1,

2, 3, 4, and Remark 1. *(b)* The running time is a direct consequence of Lemmas 1, 2, 3 and 6. The claimed number of transmissions follows from Lemmas 1, 2, 3, 6, and Remark 1.

It is important to notice that both algorithms, tree-based and mass-distribution based, compute the correct result even if *all* slug nodes crash, as long as this event happens after the COLLECTION phase. Of course, non-active slug nodes will not receive the result, but the sink node is either a delegate node or a neighbor of a delegate node. Therefore, the sink node still receives the result. The resilience to this event is not a minor advantage, given that, by Remark 1, there are at least $\Omega(n(1 - 1/\log n))$ slug nodes in the network. Additionally, the mass-distribution algorithm continue working even in presence of delegate failures, returning some result whose validity is bounded as long as the number of nodes failing is bounded and the distribution of input-values is known. Furthermore, this algorithm can be easily improved to still return the correct result in presence of a constant number of delegate node failures in each one-hop neighborhood without extra cost. To see how, consider the following modification roughly described. In each round of the COMPUTATION phase, each delegate node transmits together with its sum and weight, the number of its delegate neighbors. Then, each neighboring delegate node can include the effect of the mass lost by the delegate crashed in its computation proportionally.

In order to gain intuition on the significance of the bounds obtained for the mass-distribution algorithm, it is useful to bound the conductance using the geometry of the deployment area. For example, It can be proved that for any unit area of rectangular shape, with smaller side g, the conductance is bounded by $\Omega(g \log n/rn)$. For the sake of brevity, we leave the details to the appendix.

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. 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.
4. S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530, 2006.
5. J.-Y. Chen, G. Pandurangan, and D. Xu. Robust computation of aggregates in wireless sensor networks: distributed randomized algorithms and analysis. In *Proc. of the 4th Intl. Symp. on Information Processing in Sensor Networks*, page 46, 2005.
6. F. Chung. Spectral graph theory. http://www.math.ucsd.edu/ fan/research/revised.html, 2006.
7. A. G. Dimakis, A. Sarwate, and M. Wainwright. Geographic gossip : Efficient aggregation for sensor networks. In *Proc. of the 5th Intl. Symp. on Information Processing in Sensor Networks*, 2006.
8. 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.
9. 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.
10. 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.
11. L. Fejes-Tóth. über einen geometrischen satz. *Mathematische Zeitschrift*, 46(1):83–85, 1940.
12. W. Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley & Sons, Inc., New York, NY, USA, 3rd edition, 1968.
13. B. Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *Journal of Computer and System Sciences*, 53(3):357–370, 1996.
14. I. Gupta, R. van Renesse, and K. P. Birman. Scalable fault-tolerant aggregation in large process groups. In *DSN*, pages 433–442. IEEE Computer Society, 2001.
15. 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.
16. J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *SOSP*, pages 146–159, 2001.
17. C. Intanagonwiwat, D. Estrin, R. Govindan, and J. S. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS*, pages 457–458, 2002.
18. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.
19. M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.
20. M. Jerrum and A. 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.

21. 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.
22. 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.
23. B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *ICDCS Workshops*, pages 575–578. IEEE Computer Society, 2002.
24. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. 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.
25. S. Madden, R. Szewczyk, M. J. Franklin, and D. 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.
26. D. S. Mitrinović. *Elementary Inequalities*. P. Noordhoff Ltd. - Groningen, 1964.
27. T. Moscibroda and R. Wattenhofer. Maximal independent sets in radio networks. In *Proc. 24th Ann. ACM Symp. on Principles of Distributed Computing*, pages 148–157, 2005.
28. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
29. S. Nath, P. B. Gibbons, S. Seshan, and Z. 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.
30. 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.
31. P. Rentala, R. Musumuri, U. Saxena, and S. Gandham. Survey on sensor networks. http://citeseer.nj.nec.com/479874.html.
32. A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.
33. J. Zhao, R. Govindan, and D. 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.

Appendix

A Proof of Theorem 1

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. 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 $\boldsymbol{\nu}_1$ such that a value $\nu_1(x)$ is assigned to some node $x \in X$ and $\mathcal{F}(\boldsymbol{\nu}_1)$ is the value returned to y by $\mathcal{P}$ in $o(D)$ steps. Since $\mathcal{F}$ is sensitive, there exists an assignment $\boldsymbol{\nu}_2$ such that $\nu_1(x) \neq \nu_2(x)$ that makes $\mathcal{F}(\boldsymbol{\nu}_1) \neq \mathcal{F}(\boldsymbol{\nu}_2)$. Consider the execution of $\mathcal{P}$ under this new assignment $\boldsymbol{\nu}_2$. It is not possible that a value different than $\mathcal{F}(\boldsymbol{\nu}_1)$ 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 now 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[19]. Consider a node $x \in X$ that is scheduled to transmit last in X by $\mathcal{P}$[20]. Consider now an assignment of values $\boldsymbol{\nu}_1$ such that a value $\nu_1(x)$ is assigned to x and $\mathcal{F}(\boldsymbol{\nu}_1)$ is the value returned to y by $\mathcal{P}$ in $o(\Delta)$ steps. Since $\mathcal{F}$ is sensitive, there exists an assignment $\boldsymbol{\nu}_2$ such that $\nu_1(x) \neq \nu_2(x)$ that makes $\mathcal{F}(\boldsymbol{\nu}_1) \neq \mathcal{F}(\boldsymbol{\nu}_2)$. Consider the execution of $\mathcal{P}$ under this new assignment $\boldsymbol{\nu}_2$. It is not possible that a value different than $\mathcal{F}(\boldsymbol{\nu}_1)$ 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.

B Proof of Lemma 3

Proof. Using well-known techniques as in [10], 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.

C Mixing-time of Stingy Share

Given that $\mathbf{X}$ is ergodic, the eigenvalues of $\boldsymbol{P}$ are $1 = \lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{|M|-1} > -1$. It is known that the mixing time is related to the *spectral gap* of the transition matrix $1 - \lambda_{max}$, where $\lambda_{max} = \max\{\lambda_1, |\lambda_{|M|-1}|\}$. We precise that relation as in [32]. First, in order to measure the deviation from stationarity, define the *relative pointwise distance* after t rounds over a non-empty subset $U \subseteq M$ as $\Delta_U(t) = \max_{i,j \in U}\{|p_{ij}^{(t)} - \pi_j|/\pi_j\}$.

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

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

Then, $\Delta_U(t)$ is the largest relative difference between $\boldsymbol{\mu}^{(t)}$ and $\boldsymbol{\pi}$ at any state $j \in U$, maximized over all possible initial states $i \in U$. Given that for all $i, j \in M$, $p_{ij}\pi_i = p_{ji}\pi_j$, the Markov chain $\mathbf{X}$ is *time-reversible*. The following proposition establishes the relation between the spectral gap and mixing time.

Proposition 1. *([32, Prop. 3.1]) Let $\boldsymbol{P}$ be the transition matrix of an ergodic time-reversible Markov chain, $\boldsymbol{\pi}$ its stationary distribution and $\{\lambda_i : 0 \leq i \leq |M| - 1\}$ its eigenvalues, with $\lambda_0 = 1$. Then, for any non-empty subset $U \subseteq M$ and all $t > 0$, the relative pointwise distance $\Delta_U(t)$ satisfies $\Delta_U(t) \leq \lambda_{max}^t / \min_{j \in U} \pi_j$, where $\lambda_{max} = \max\{|\lambda_i| : 1 \leq i \leq |M| - 1\}$.*

The eigenvalue $\lambda_{|M|-1}$ is relevant only if it is negative. However, given that $\forall i \in M, p_{ii} \geq 1/2$ by definition of Stingy Share, it holds that $\lambda_{|M|-1} \geq 0$ [32]. Therefore, $\lambda_{max} = \lambda_1$. Therefore, in order to obtain an asymptotic bound on the mixing time, it is enough to bound λ_1. The following result [32], bounds λ_1 using the conductance as defined in Section 3.2.

$$\lambda_1 \leq 1 - \frac{\Phi^2}{2}. \tag{1}$$

Lemma 7. *(Mixing time) 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. Consider the Aggregate Computation Scheme to solve the Average Problem as described. Let M be the set of delegate nodes defined in the preprocessing phase. Let $\sigma_i^{(0)}$ and $\omega_i^{(0)}$ be the sum and weight of delegate node $i \in M$ obtained after the* COLLECTION *phase. Then, implementing the* COMPUTATION *phase using Stingy Share as described, after $\tau_3 \in O((\ln(1+2/\varepsilon)+\ln|M|)/\Phi^2)$ rounds the following condition is satisfied. For all $t \geq \tau_3$, $|\sigma_i^{(t)}/\omega_i^{(t)} - \overline{\nu}|/\overline{\nu} \leq \varepsilon$, for all $i \in M$ and for a given parameter $\varepsilon > 0$.*

Proof. As shown in Lemma 5, in order to prove this claim, given an $\varepsilon > 0$, it is enough to find a time $\tau_3 \geq 0$ such that, for all $t \geq \tau_3$, $\max_{i \in M}\{|\mu_i^{(t)} - \pi_i|/\pi_i\} \leq \varepsilon'$, where $\varepsilon' = \varepsilon/(2+\varepsilon)$. From Proposition 1 and using the fact that the stationary distribution is uniform, we have $\max_{i \in M}\{|\mu_i^{(t)} - \pi_i|/\pi_i\} \leq \lambda_1^t |M|$. Hence, we look for the minimum $t \geq 0$ such that $\lambda_1^t |M| \leq \varepsilon/(2+\varepsilon)$. Using that $1 - x \leq e^{-x}, 0 < x < 1$ [26, §2.68], we want

$$\begin{aligned} \frac{|M|}{e^{t(1-\lambda_1)}} &\leq \frac{\varepsilon}{2+\varepsilon} \\ t &\geq \frac{1}{1-\lambda_1}\left(\ln\left(1+\frac{2}{\varepsilon}\right) + \ln|M|\right) \end{aligned} \tag{2}$$

Replacing Equation 1, the claim follows.

D Proof of Lemma 5

Proof. The Markov chain $\mathbf{X}$ characterizing Stingy Share 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 [28], $\mathbf{X}$ is ergodic and it has a unique stationary distribution. Since $\boldsymbol{P}$ is doubly stochastic, the system $\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}$ admits the solution $\boldsymbol{\pi} = (1/|M| \dots 1/|M|)$ which, by the aforementioned theorem, is unique. Let $\boldsymbol{\mu}^{(t)}$ 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 $i \in M$. Then, for any initial distribution $\boldsymbol{\mu}^{(0)}$ and for all $t \geq \tau$ it holds that, $(1-\varepsilon')/|M| \leq \sum_{i \in M} \mu_i^{(0)} (\boldsymbol{P}^t)_{ij} \leq (1+\varepsilon')/|M|$, for all $j \in M$. But then, it must also hold that $(1-\varepsilon')/|M| \leq (\boldsymbol{P}^t)_{ij} \leq (1+\varepsilon')/|M|$, for all $i, j \in M$. To see why, assume for the sake of contradiction that there is a pair $i', j' \in M$ such that $(\boldsymbol{P}^t)_{i'j'} < (1-\varepsilon')/|M|$ or $(\boldsymbol{P}^t)_{i'j'} > (1+\varepsilon')/|M|$. 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 delegate node $i \in M$, $\sigma_i^{(t)}/\omega_i^{(t)} = (\boldsymbol{\sigma}^{(0)}\boldsymbol{P}^t)_i / (\boldsymbol{\omega}^{(0)}\boldsymbol{P}^t)_i = \sum_j \sigma_j^{(0)} (\boldsymbol{P}^t)_{ji} / \sum_j \omega_j^{(0)} (\boldsymbol{P}^t)_{ji}$. Then, for all $t \geq \tau$, $(1-\varepsilon') \sum_j \sigma_j^{(0)} / (1+\varepsilon') \sum_j \omega_j^{(0)} \leq \sigma_i^{(t)}/\omega_i^{(t)} \leq (1+\varepsilon') \sum_j \sigma_j^{(0)} / (1-\varepsilon') \sum_j \omega_j^{(0)}$. Given that $\sum_j \sigma_j^{(0)} / \sum_j \omega_j^{(0)} = \overline{\nu}$, we have that $t \geq \tau$, $\overline{\nu}(1 - 2\varepsilon'/(1+\varepsilon')) \leq \sigma_i^t/\omega_i^t \leq \overline{\nu}(1 + 2\varepsilon'/(1-\varepsilon'))$. Thus, $|\sigma_i^{(t)}/\omega_i^{(t)} - \overline{\nu}|/\overline{\nu} \leq 2\varepsilon'/(1-\varepsilon')$ and the claim follows making $\varepsilon' = \varepsilon/(2+\varepsilon)$.

E Bounding the Conductance

Consider the geometric graph $G_M = \{M, E_M\}$, underlying the Markov chain $\mathbf{X}$, where $\{i, j\} \in E_M$ if and only if i and j are located within an Euclidean distance of at most r. In order to further bound the mixing time, it is useful to bound the conductance using the geometric properties of G_M. Although the distribution of delegate nodes in the plane is close to uniform, G_M is not a regular graph and bounding the conductance precisely is difficult. Given that we are interested in asymptotic bounds on the time taken by the COMPUTATION phase, it is enough to show an asymptotic bound on the conductance. The magnitude of the conductance depends on the bottlenecks present in the network. Given that the distribution of delegate nodes is roughly uniform, the shape of the area where nodes are deployed governs the presence of bottlenecks.

In order to gain intuition about the rate of convergence, we analyze the conductance of the network when nodes are deployed in a rectangular area of smaller side g.

Lemma 8. *Given a unit rectangular area of deployment with minimum side g, the conductance of $G_M = \{M, E_M\}$ is in $\Omega(g \log n / rn)$.*

Proof. Recall that the conductance of a Markov chain $\mathbf{X}$ with underlying graph G_M is $\Phi(G_M) \triangleq \min\{\sum_{i \in C, j \notin C} w_{ij} / \sum_{i \in C} \pi_i\}$. Where $w_{ij} = p_{ij}\pi_i = p_{ji}\pi_j$ is the *weight* of an edge $\{i, j\}$ in G_M and the minimization is over all subsets $C \subset M$ such that $0 < \sum_{i \in C} \pi_i \leq 1/2$. Given that the stationary distribution is uniform, $\sum_{i \in C} \pi_i = |C|/|M|$. Hence, $\sum_{i \in C} \pi_i$ is maximized when $|C|$ is maximized up to $|M|/2$. However, $\sum_{i \in C, j \notin C} w_{ij}$ depends on how C is chosen so, a more careful analysis is needed.

Consider a complete cover of the area with circles of radius αr as in Figure 1. Let ℓ be the number of circles and let C_i be the set of delegate nodes covered by circle i.

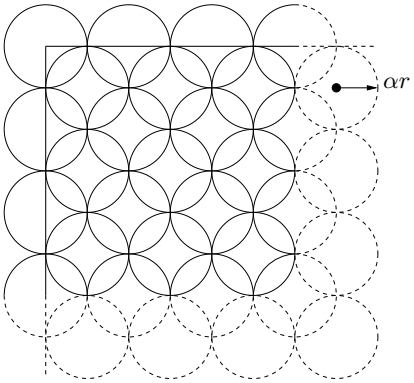


Fig. 1. Illustration of Lemma 8.

Under complete coverage assumptions, by definition of the maximal independent set, we know that for every circle i it must hold that $C_i \neq \emptyset$. Furthermore, given a pair of overlapping circles i, j, it must hold that $C_i \setminus C_j \neq \emptyset$ and $C_j \setminus C_i \neq \emptyset$. Then, it is possible to define sets of delegate nodes $C'_1 \subseteq C_1, \ldots, C'_\ell \subseteq C_\ell$ so that $\bigcup_i C'_i = M$ and $\forall i, j : C'_i \cap C'_j = \emptyset$.

Given the graph $G_M = \{M, E_M\}$, consider a subgraph $G' = \{V', E'\}$ where $V' = M$ and $E' = \{\{i, j\} | \{i, j\} \in E_M \wedge i \in C_t \wedge j \in C_s \wedge (t = s \vee \forall k \in C_s : (k \neq j \Rightarrow \{i, k\} \notin E'))\}$. In words, all edges connecting delegate nodes within the same subset plus one edge between each pair of sets of overlapping circles. Using the same algorithm on this subgraph, the Markov chain characterizing it still is finite, irreducible and aperiodic so, it is ergodic, time-reversible and it has uniform stationary distribution $\boldsymbol{\pi} = (1/|M| \ldots 1/|M|)$. Therefore, $\sum_{i \in C} \pi_i = |C|/|M|$.

define properly

Consider now the inductive process of adding delegate nodes to C, one subset at a time. Each new subset contributes to increase $|C|$ by at least one. On the other hand, there is always a way of adding an overlapping circle so that the number of edges of G' crossing the boundary of C does not increase by more than one. Therefore, the minimum conductance for G' is achieved when $\sum_{i \in C} \pi_i$ is big.

We consider then only subsets where the boundary is a single curve. Among all subsets of size close to $|M|/2$, we want to find the one with the minimum number of edges crossing the boundary, i.e., the boundary of minimum length. Finding the precise boundary among circles that makes $|C| = |M|/2$ may be impossible because circles have different number of nodes. However, we look for an asymptotic bound, and the number of nodes in any circle is also bounded from above by a constant. Therefore, an approximate boundary is enough. Such approximation is a line, parallel to the side of length g, that halves the area. Under complete coverage assumptions, there are at least $g/\alpha r$ circles touching that line, partially or completely in one side of it. Each of these circles has at least 1 edge crossing the boundary. Thus, the total number of edges crossing such a boundary is at least $g/\alpha r$. Therefore, $\sum_{i \in C, j \notin C} w_{ij} = \sum_{i \in C, j \notin C} p_{ij} \pi_i \in \Omega(g/r|M|)$. On the other hand, $\sum_{i \in C} \pi_i \in \Theta(1)$ thus, the conductance of G' is also

$\Omega(g/r|M|)$. Adding edges to G' can not decrease the conductance so this is also a lower bound for G_M. As observed before, $|M| \in O(n/\log n)$, thus the claim follows.

F Figures

Algorithm 1: The Aggregate Computation Scheme. τ_1 is the time slot to measure the input-values, D is the diameter of the network and Δ the maximum degree.

TRIGGER: The sink node broadcasts (τ_1, D, Δ).
COLLECTION: Delegate nodes aggregate slugs input.
COMPUTATION: Delegate nodes compute the aggregate function.
DISSEMINATION: Delegate nodes distribute the result.

Algorithm 2: 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 $j \in M$:
  Set $\sigma_j \leftarrow \nu_j$.
  Set $\omega_j \leftarrow 1$.
  for τ_2 *steps* **do**
    if *a message* (i, j, ν_i) *is received* **then**
      Set $\sigma_j \leftarrow \sigma_j + \nu_i$.
      Set $\omega_j \leftarrow \omega_j + 1$.

Algorithm 3: The COMPUTATION phase for the Average Problem: Stingy Share.

For delegate node $i \in M$:
  for *k=1 to* τ_3 **do**
    Transmit $(i, k, \sigma_i, \omega_i)$.
    Receive $(j, k, \sigma_j, \omega_j)$ for all $j \in N(i)$.
    Set $\sigma_i \leftarrow (1 - |N(i)|/2\delta)\sigma_i + \sum_{j \in N(i)} \sigma_j/2\delta$.
    Set $\omega_i \leftarrow (1 - |N(i)|/2\delta)\omega_i + \sum_{j \in N(i)} \omega_j/2\delta$.

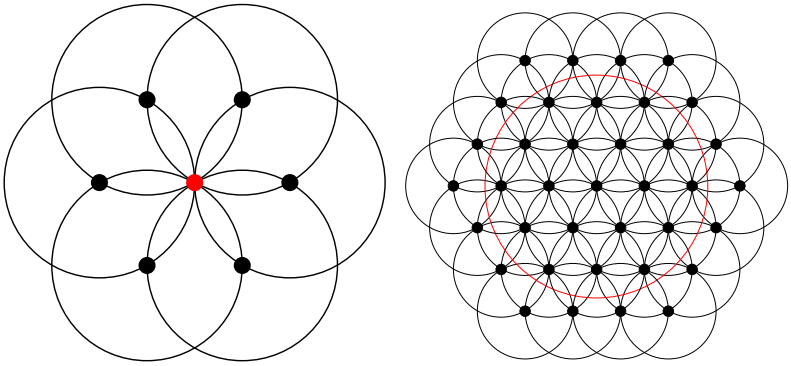


Fig. 2. Illustration of maximum degree.

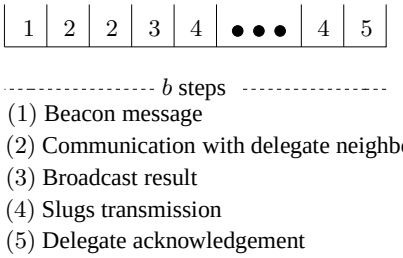


Fig. 3. Illustration of time-slots usage.