Efficient In-Network Processing Through Local Ad-Hoc Information Coalescence*

Onur Savas, Murat Alanyali, and Venkatesh Saligrama

Department of Electrical and Computer Engineering,
Boston University, Boston, MA, 02215
{savas, alanyali, srv}@bu.edu

Abstract. We consider in-network processing via local message passing. The considered setting involves a set of sensors each of which can communicate with a subset of other sensors. There is no designated fusion center; instead sensors exchange messages on the associated communication graph to obtain a global estimate. We propose an asynchronous distributed algorithm based on local fusion between neighboring sensors. The algorithm differs from other related schemes such as gossip algorithms in that after each local fusion one of the associated sensors ceases its activity until it is re-activated by reception of messages from a neighboring sensor. This leads to substantial gains in energy expenditure over existing local ad-hoc messaging algorithms such as gossip and belief propagation algorithms. Our results are general and we focus on some explicit graphs, namely geometric random graphs, which have been successfully used to model wireless networks, and d-dimensional lattice torus with n nodes, which behave exactly like mesh networks as n gets large. We quantify the time, message and energy scaling of the algorithm, where the analysis is built upon the coalescing random walks. In particular, for the planar torus the completion time of the algorithm is $\Theta(n \log n)$ and energy requirement per sensor node is $O((\log n)^2)$ and for 3-d torus these quantities are $\Theta(n)$ and $O(\log n)$ respectively. The energy requirement of the algorithm is thus scalable, and interestingly there appears little practical incentive to consider higher dimensions. Furthermore, for the planar torus the algorithm exhibits a very favorable tradeoff relative to gossip algorithms whose time and energy requirements are shown here to be $\Omega(n)$. Also, the proposed algorithm can be generalized to robustify against packet losses and permanent node failures without entailing significant energy overhead. The paper concludes with numerical results.

1 Introduction

Wireless sensors bear a vast potential as they can be networked to form amorphous systems that are far more capable than their parts. This potential is

* This research was supported by Presidential Early Career Award N00014-02-100362, NSF CAREER Programs ANI-0238397, ECS-0449194 and NSF programs CCF-0430983, and CNS-0435353.

P. Gibbons et al. (Eds.): DCOSS 2006, LNCS 4026, pp. 252–265, 2006.

accompanied by substantial technical challenges, namely such systems call for distributed models of operation that comply with requirements that arise due to energy limitations of sensors, packet losses along wireless links, sensor failures, and possibly uncontrollable network topologies.

In this paper we study ad-hoc distributed computation of a wide class of functions of network-wide measurements. The topic of distributed computation has received significant interest recently in the context of sensor networks (see [17, 8, 5 ,13, 14, 15, 11, 9, 12, 2, 3, 4, 1] and included references). Distributed computation and optimization arises in a number of different applications ranging from signal processing such as distributed localization/detection/estimation/tracking to load balancing and self-organization in communication networks [8, 7]. Previously proposed techniques can be broadly categorized into two groups: The fusion-centric approach (see [17, 11, 9] and included references) assumes that each sensor has a communication link to a data fusion center and each sensor node computes a local decision, which is then communicated to the fusion center. The fusion center then forms a global estimate of the desired function based on local decisions. The ad-hoc approach, on the other hand, involves no designated fusion center but focuses on establishing consensus within the network via local message exchanges. This approach is appears to be more suitable to address energy issues in large-scale networks and also appears to have robustness advantages. This is because: (a) it requires much less energy to communicate to neighboring sensors; (b) in-network processing through locally fusing information results in compression; (c) no fusion center implies no single point of failure; (d) they are well suited for asynchronous operation and hence robust to packet losses and node failures. Specifically, consider the so called gossip algorithm [5] for computing the average of all the sensor observations. Gossip algorithms accomplish this task by randomly choosing two neighboring sensor nodes at each time and replacing their current values by their average. It turns out that this process over time converges to the average of all sensor values at all the sensors, i.e., all the sensors achieve a consensus. Such consensus algorithms have been recently explored in other contexts such as detection [2, 3] and control [13].

Nevertheless, these algorithms have fundamental disadvantages from an energy efficiency perspective. In particular, if n nodes are uniformly distributed in a given area and E_b is the average communication energy-per-message required to communicate information to a neighboring node, then the energy-per-node required to achieve consensus scales as $\Omega(nE_b)$, which can be significant for a large sensor network. The fundamental reason is that energy efficiency resulting from in-network processing is offset by ad-hoc message passing that results in redundant computations, i.e., the same set (or largely similar set) of nodes repeatedly fuse their information at different points in time. In a related problem involving distributed detection, the significant energy scaling can be attributed to the loopy nature of the network where messages sent from one node repeatedly arrive at the node at different points in time. In order to ensure that no information from any node is forgotten, each node must re-inject their messages in to the network to reinforce their information [3]. At a fundamental level the significant

scaling of energy arises due to the slow mixing rate of large networks, which can be attributed to rather large second eigenvalues of certain connectivity matrices associated with the underlying communication graph. An immediate solution to reduce such redundant computations is to design a communication tree so that message from the leaf nodes arrive at accumulation points (or clusterheads), which fuse the information and forward this information to a parent node. However, such a construction requires centralized planning and is inconsistent with the requirements of ad-hoc sensor network operation, where packet losses and node failures are common.

To address these issues we present a novel distributed computing approach in this paper, where not only is the ad-hoc network operation preserved but where the energy-related disadvantages of the existing local message passing resulting from repeated redundant computations is also minimized. The emphasis of the paper is on two important but conflicting figures of merit for wireless sensor networks, namely the time and the energy required to complete the computation. We introduce an asynchronous distributed algorithm that is based on autonomous pairwise communications between pairs of neighboring sensors in the underlying communication topology. The algorithm forces one of the communicating nodes to cease transmitting new messages until it is re-activated due to reception of a message from a neighboring node. This results in exponential energy gains compared other similar schemes. We adopt messaging complexity as a proxy for energy requirement (as described earlier) and quantify the trade-off between this quantity and the completion time. For the d-dimensional lattice torus with n sensors, we show that the completion time is $\Theta(n(\log n)^{\alpha})$ and energy requirement per sensor node is $O((\log n)^{\alpha+1})$ where $\alpha = 1$ for $d = 2$ and $\alpha = 0$ for $d \geq 3$. The algorithm thus has scalable energy requirement, furthermore its performance is almost insensitive to changes in the network connectivity represented by different values of $d \geq 2$, hinting at the possibility of predictable performance over mesh topologies. Section 5 focuses on averaging of sensor measurements as a case study for comparison with gossip algorithms. We show that both the time and the per-node energy requirement of the gossip algorithm of [5] are $\Omega(n)$ on the 2-dimensional torus with n nodes, irrespective of the choice of the parameters of this algorithm. The algorithm introduced in the present paper thus exhibits a very favorable trade-off between the two performance measures for $d = 2$, but the gains in energy requirements come at the expense of more substantial compromise in the completion time for larger d.

With regard to other important practical considerations, the algorithm is robust against packet losses provided that reliable link-layer protocols are employed in wireless exchanges. The algorithm is also resilient against permanent node failures, furthermore the tolerable number of node failures can be provisioned and the energy requirement of the algorithm increases linearly with this number.

The rest of the paper is organized as follows. The distributed computation problem is formulated in Section 2 and the proposed algorithm is specified in Section 3. Section 4 is devoted to the analysis of this algorithm on d-dimensional tori for $d \geq 1$. A comparative study with gossip algorithms, specifically on geo-

metric random graphs, is given in Section 5, and the paper concludes with final remarks in Section 6.

2 Problem Definition

Let Λ be a set that is closed under operation f. We shall assume that f is commutative and associative, and that Λ has an identity element with respect to f, that is there exists $e \in \Lambda$ such that $f(\lambda, e) = \lambda$ for $\lambda \in \Lambda$. Let $F_2 = f$. Given $n \geq 3$ and a permutation π of $\{1, 2, \cdots, n\}$ define the mapping $F_n : \Lambda^n \mapsto \Lambda$ recursively by

$$F_n(\lambda_1, \lambda_2, \cdots, \lambda_n) = f(\lambda_{\pi(1)}, F_{n-1}(\lambda_{\pi(2)}, \lambda_{\pi(3)}, \cdots, \lambda_{\pi(n)})), \tag{1}$$

for $\lambda_1, \lambda_2, \cdots, \lambda_n \in \Lambda$. The notational dependence on the permutation π is dropped here because the mapping F_n does not depend on π due to the commutativity and associativity of f.

We will be concerned with distributed computation of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ in cases when each λ_i is known to a distinct agent. In typical applications that motivate this work such an agent represents one of n sensors involved in a statistical inferencing procedure, λ_i represents a measurement taken by sensor i or a function thereof that reflects a local estimate, and $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ represents a global estimate or a sufficient statistic of the measurements. For example, in the simplistic case of computing the maximum value of the sensor measurements one may take $\Lambda = \mathbb{R} \cup \{-\infty\}$, $e = -\infty$, and $f(\lambda_1, \lambda_2) = \max(\lambda_1, \lambda_2)$. Representation of weighted vector averages, which are of interest in finding least-squares estimates, is illustrated in the following example:

Example 1. Let $k \geq 1$ be an integer and let $M^{k\times k}$ be the set of k-dimensional positive definite matrices. For $i = 1, 2, \cdots, n$ let $x_i \in \mathbb{R}^k$ and let $w_i \in M^{k\times k}$. The weighted average $(\sum_{i=1}^n w_i)^{-1} \sum_{i=1}^n w_i x_i$ can be expressed in the form (1) by choosing $\Lambda = \{\mathbb{R}^k \times M^{k\times k}\} \cup \{e\}$ with $\lambda_i = (x_i, w_i)$, and by setting

$$f(\lambda, \lambda') = ((w + w')^{-1}(wx + w'x')\ ,\ w + w'),$$

for $\lambda = (x, w), \lambda' = (x', w') \in \Lambda - \{e\}$.

We consider computation of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ under communication constraints that are specified by an undirected graph $G = (V, E)$ where each node in V denotes a sensor (hence $|V| = n$) and an edge $(i, j) \in E$ indicates a bidirectional communication link between sensors i and j. To avoid trivialities we shall assume that G is connected. Note that since F_n admits flexible decomposition in terms of the atomic operation f, there exist a sequence $(e_1, e_2, \cdots, e_{n-1})$ of distinct edges in G such that $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ can be computed by sequentially executing f on the values at the two ends of each link in the given order. Furthermore the edge set $\{e_1, e_2, \cdots, e_{n-1}\}$ forms a spanning tree for G, and some of the aforementioned operations can be executed in parallel so that the overall computation can be completed in time proportional to the diameter of G. Rather than such centralized algorithms, our focus here is on decentralized algorithms that require neither global information about G nor network-wide synchronization.

3 Information Coalescence

Consider the following distributed, asynchronous algorithm which requires each sensor to be aware of only the sensors that it can communicate directly:

Algorithm COALESCENT(f, λ, G)**:** Each sensor maintains a variable *value* which is an element of the set Λ, and a variable *status* which is either *'carrier'* or *'idle'*. We denote the value of sensor $i \in V$ at time $t \geq 0$ by $v_i(t)$, and indicate the status of the sensor via $\xi_i(t)$ defined by

$$\xi_i(t) = \begin{cases} 1 & \text{if status of sensor } i \text{ is 'carrier' at time } t, \\ 0 & \text{else.} \end{cases}$$

Initially $(v_i(0), \xi_i(0)) = (\lambda_i, 1)$ and these variables evolve as follows: Each sensor has an independent Poisson clock that ticks at unit rate. When the local clock of sensor i ticks, say at time t_o, the sensor does not take any action if its current status is 'idle' (i.e. $\xi_i(t_o^-) = 0$). Otherwise, if $\xi_i(t_o^-) = 1$, the sensor chooses a neighbor at random, sends its current value $v_i(t_o^-)$ to that neighbor, and sets $(v_i(t_o), \xi_i(t_o)) = (e, 0)$ (in particular sets is status to 'idle'). The selected neighbor, say sensor j, sets $v_j(t_o) = f(v_j(t_o^-), v_i(t_o^-))$ and $\xi_j(t_o) = 1$. A pseudo-code for the algorithm is given in Figure 1.

```
Procedure Initialize()
   {v_i ← λ_i;
    status←'carrier';}

Procedure Send()
   if(status=='carrier'){
     choose neighbor;
     send(v_i);
     v_i ← e;
     status←'idle';}

Procedure Receive(message)
   {v_i ← f(v_i, message);
    status←'carrier';}
```

Fig. 1. Three subroutines that specify Algorithm COALESCENT(f, λ, G) at node i. Send() is activated by the local Poisson clock, and Receive() is activated by message reception from another sensor.

Algorithm COALESCENT(f, λ, G) is based on sequential execution of f on edges of G, but these edges are selected in a distributed and randomized manner, without particular regard to any optimality notion. Note that when an idle node receives a message the value at the originating node simply passes onto the idle node, whereas if the receiving node is also a carrier then its value becomes the image of the two values under f, thereby executing a step towards computation of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$. The following proposition points out a sample-path property that is useful in proving correctness of the algorithm.

Proposition 2. *Under Algorithm* COALESCENT(f, λ, G)*,*

$$F_n(v_1(t), v_2(t), \cdots, v_n(t)) = F_n(\lambda_1, \lambda_2, \cdots, \lambda_n),$$

for all $t > 0$*.*

We say that a *coalescence* occurs whenever a message is transmitted from a carrier sensor to another carrier sensor. Note that the number of carrier sensors $|\xi(t)| \triangleq \sum_{i=1}^{n} \xi_i(t)$ is non-increasing in time t and it decreases by 1 at the times of coalescence in the network. For each integer $k \leq n$ define the random time σ_k as

$$\sigma_k = \inf\{t \geq 0 : |\xi(t)| = k\}.$$

That is, σ_k is the first time that k carrier nodes remain in the network. The random variable σ_1 is of particular interest since $v_i(t) = e$ for each sensor i such that $\xi_i(t) = 0$, and thus by Proposition 2 for $t \geq \sigma_1$ there exists a unique sensor $i(t)$ such that $v_{i(t)}(t) = F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$. We therefore regard σ_1 as the stopping time of Algorithm `COALESCENT`(f, λ, G).

Issues in implementation and robustness. The basic form of the algorithm can be modified to recognize the termination time σ_1 by maintaining local counters that keep track of how many coalescence operations were involved in obtaining the present value of each sensor. Note that this can be implemented in a distributed manner by including the local counter as part of each transmitted message. The algorithm is robust against packet losses provided that reliable protocols such as those based on handshaking are employed at the link layer. A more serious mode of failure is permanent failure of sensor nodes, since if a sensor dies when its status is 'carrier', then in addition to its initial value λ_i a set of other such sensor values are also lost. The impact of such failures is more pronounced in later stages of the algorithm when each carrier node is the unique bearer of typically many sensor values. This issue can be mitigated by running multiple independent instances of the algorithm simultaneously in the network. The resulting cost in the energy expenditure is a constant factor, which is the number of such instances.

For more insight on σ_1 define $\xi(t) = (\xi_i(t) : i \in V)$ and note that $(\xi(t) : t \geq 0)$ is a time-homogeneous Markov process with state space $\{0, 1\}^V$. It can be verified that $(\xi(t) : t \geq 0)$ can be constructed as follows: At time 0 simultaneously start n simple symmetric random walks, one at each node of the graph G. Namely each random walk jumps at the ticks of an independent Poisson clock of unit rate, to a neighboring node chosen at random. Let distinct random walks evolve independently until two of them occupy the same node, and coalesce these two random walks into one (that is, bind these two walks together so that they make the same moves) from that time on. Finally, set $\xi_i(t) = 1$ if there is a random walk occupying node i at time t, and set $\xi_i(t) = 0$ otherwise. The process $(\xi(t) : t \geq 0)$ is known as the coalescing random walk, and has been extensively studied as dual process for voter models of interacting particle systems.

The observation of the previous paragraph will be useful in the analysis of the algorithm in the following section. Here we note that the algorithm terminates almost surely on any finite graph:

Lemma 3. $P(\sigma_1 < \infty) = 1$.

Despite its close relationship to random walks, drawing more detailed conclusions about the complexity of Algorithm `COALESCENT`(f, λ, G) on arbitrary graphs ap-

pears difficult. In the next section we pursue this goal for the special case of d-dimensional torus for which substantial understanding of the coalescing random walk has been developed in the applied probability literature.

4 Time and Energy Requirements on the d-Dimensional Torus

Given integers $d, N \geq 1$ let $\mathbb{T}_N^d$ denote the d-dimensional lattice torus with N^d nodes. The graph $\mathbb{T}_N^d$ can be formally defined by identifying its nodes by members of $\{1, 2, \cdots, N\}^d$ and its edges by pairs in $\{1, 2, \cdots, N\}^d$ that are at Hamming distance 1 under modulo arithmetic with respect to N. In this section we analyze the complexity of Algorithm $\texttt{COALESCENT}(f, \lambda, \mathbb{T}_N^d)$ by examining the coalescing random walk process $(\xi(t) : t \geq 0)$.

We start with considering the termination time σ_1. It appears reasonable to expect σ_1 to be stochastically increasing in the network size; in fact for fixed d the following result of Cox [10] provides the precise growth rate of each σ_k with N. Let

$$s_N = \begin{cases} N^2 & \text{if } d = 1 \\ N^2 \log N & \text{if } d = 2 \\ N^d & \text{if } d \geq 3, \end{cases} \quad \text{and} \quad Q = \begin{cases} 1/6 & \text{if } d = 1 \\ 2/\pi & \text{if } d = 2 \\ \gamma(d) & \text{if } d \geq 3, \end{cases}$$

where each $\gamma(d)$ is a finite and strictly positive constant as identified in [10, Equation (1.2)].

Theorem 4. *([10, Theorem 6]) Let $G = \mathbb{T}_N^d$. For each integer $k \geq 1$ there exists a random variable σ_k^* such that σ_k / s_N converges in distribution to σ_k^*. Furthermore $\lim_{N\to\infty} E[\sigma_1/s_N] = E[\sigma_1^*] = Q$.*

Distributions of the limiting random variables σ_k^* are also obtained in [10]. The characterization of σ_k^* therein reveals an interesting and somewhat surprising relationship between $(|\xi(t)| : t \geq 0)$ and a far simpler random process that provides substantial insight for the present analysis. Namely, let $(D_t : t \geq 0)$ be the Markovian pure death process where for each integer state $m \geq 1$ a transition from m to $m-1$ occurs at rate $\binom{m}{2}$. For $t \geq 0$ and integers n, k let

$$q_{n,k}(t) = P(D_t = k | D_0 = n).$$

The exact form of $q_{n,k}(t)$ is not immediately relevant to the present discussion, however we note that $\lim_{n\to\infty} q_{n,k}(t)$ exists for each $k \geq 1$ and $t \geq 0$, and denote the limit value by $q_{\infty,k}(t)$. Explicit expressions for $q_{n,k}(t)$ and $q_{\infty,k}(t)$ can be found in [10, 16].

Theorem 5. *([10, Theorem 6]) For each integer $k \geq 1$ and $t \geq 0$, $P(\sigma_k^* \leq t) = \sum_{l=1}^{k} q_{\infty,l}(2t/Q)$.*

In informal terms, Theorems 4 and 5 suggest that for large values of N the time scaled process $(|\xi(ts_N)| : t \geq 0)$ behaves almost like the death process $(D_{tQ/2} : t \geq 0)$. It can be verified via straightforward comparison of generators that $(D_t : t \geq 0)$ represents the number of distinct random walks in a coalescing random walk process when G is completely connected. Hence in the limit of large N the spatial dependence of the process $(|\xi(ts_N)| : t \geq 0)$ completely washes out; in the new time-scale distinct walks coalesce with randomly chosen counterparts, at rates that do not depend on locations. This intuition, however, should be treated with caution, since like Theorems 4–5 it is valid only after the number of remaining distinct walks reduce to bounded values (for example, Theorem 5 does not provide any insight on $\sigma_{N/3}$, i.e. the time required to have $N/3$ carriers left).

Remark 6. In search of some insight about the conclusions drawn in previous paragraphs, suppose for the moment that at each time t the $|\xi(t)|$ carrier nodes are uniformly distributed over $\mathbb{T}_N^d$, independently of the history prior to t. Note that there is no particular reason for this to hold for $t > 0$. The instantaneous rate of decrease of $|\xi(t)|$ is proportional to the expected number of edges that connect two carrier nodes; thus if the above assumption were correct, then this rate would be equal to

$$\begin{aligned} \sum_{\text{edges } l \in E} & P(l \text{ connects carrier nodes}) \\ &= \binom{dN^d}{2} \frac{|\xi(t)|(|\xi(t)|-1)}{N^d(N^d-1)} \\ &\approx \binom{|\xi(t)|}{2} \frac{d}{N^d}; \end{aligned}$$

and in turn one would expect $(|\xi(tN^d)| : t \geq 0)$ to behave like the death process $(D_{td} : t \geq 0)$. Theorems 4–5 indicate that this conclusion is not too far off for dimensions $d \geq 3$.

We next turn to the energy requirement of the algorithm. Let

$$\eta(t) = \text{Total number messages sent in the network by time } t.$$

Note that each carrier node transmits messages at rate 1 and idle nodes do not engage in message transmission; hence $|\xi(t)|$ is the rate of message generation in the network at time t. The mean number of messages transmitted in the network before the termination of the algorithm is thus given by

$$E[\eta(\sigma_1)] = \sum_{k=2}^{N^d} kE\left[\sigma_{k-1} - \sigma_k\right] = E\left[\int_0^{\sigma_1} |\xi(t)|dt\right]. \tag{2}$$

It is appealing to apply Theorem 5 and compute $E[\eta(\sigma_1)]$ for large values of N by computing the limiting expectation of each σ_k, however only tail of the

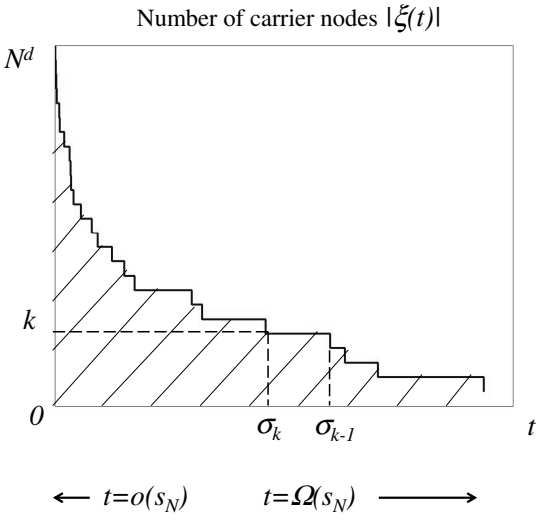

Fig. 2. The figure illustrates a sample path of the number of carrier nodes in the network. This trajectory has qualitatively different statistics for small and large values of t/s_N. The mean of the shaded area is the mean aggregate number of transmitted messages in the network.

integral in equality (2) can be computed in this manner. More explicitly, the algorithm has two qualitatively different phases as illustrated by Figure 2. For $t = o(s_N)$ there are many carrier nodes in the network, but their number diminish very quickly due to their high density. For $t = \Omega(s_N)$ only a bounded number of carrier nodes exist and σ_k can be approximated by Theorem 5 in this time interval. The former phase is short but it involves a high rate of message transmissions, whereas the latter phase lasts long but fewer messages are transmitted per unit time.

An estimate of the message complexity over the interval $t = \Omega(s_N)$ may be obtained via the following heuristic argument. Since in this interval $(|\xi(ts_N)| : t \geq 0)$ is informally approximated by a death process that dies at a rate that is roughly proportional to the square of its current value, use the solution of

$$\dot{y}_t = -y_t^2, \qquad y_0 = N^d,$$

as a proxy to $(|\xi(ts_N)| : t \geq 0)$. In particular $y_t = (t + N^{-d})^{-1}$. One may then expect

$$\begin{aligned}\int_0^{\sigma_1} |\xi(t)|dt &= s_N \int_0^{\sigma_1/s_N} |\xi(ts_N)|dt \\ &\approx s_N \int_0^{\sigma_1/s_N} y_t dt \\ &= s_N \ln(t + N^{-d})|_0^{\sigma_1/s_N}.\end{aligned}$$

Since $\sigma_1/s_N = O(1)$ by Theorem 4 this argument suggests that $O(s_N \log N)$ messages are transmitted in the considered interval. This intuition turns out to be correct and in fact the bound applies to both intervals. A formal statement is provided by the next theorem:

Let

$$m_N = \begin{cases} N^2 & \text{if } d = 1 \\ N^2(\log N)^2 & \text{if } d = 2 \\ N^d \log N & \text{if } d \geq 3. \end{cases}$$

Theorem 7. *Under Algorithm* `COALESCENT`$(f, \lambda, \mathbb{T}_N^d)$

$$\begin{aligned} &\limsup_{N\to\infty} E[\eta(\sigma_1)/m_N] < \infty, && \textit{for } d \geq 1, \\ &\liminf_{N\to\infty} E[\eta(\sigma_1)/m_N] > 0, && \textit{for } d = 1, \\ &\liminf_{N\to\infty} E[\eta(\sigma_1)/s_N] > 0, && \textit{for } d \geq 2. \end{aligned}$$

5 A Comparative Case Study

This section compares the time and energy requirements of the proposed algorithm and a gossip algorithm that has been previously studied for distributed computation of averages. Namely, the task here is to obtain the algebraic average of N^d numbers $x_1, x_2, \cdots, x_{N^d}$ each of which is known to a distinct node on the torus $\mathbb{T}_N^d$. As noted by Example 1 this task can be accomplished by Algorithm `COALESCENT`$(f, \lambda, \mathbb{T}_N^d)$ by proper choice of Λ, f and λ_i (namely by choosing $\lambda_i = (x_i, 1)$ for node i).

In broad terms, gossip algorithms refer to distributed randomized algorithms that are based on pairwise relaxations between randomly chosen node pairs. In the context of the present section a pairwise relaxation refers to averaging the two values available at the associated nodes. For completeness we next give a full description of this algorithm as studied in [5]. The algorithm is specified by a stochastic matrix $P = [P_{ij}]_{N^d \times N^d}$ such that $P_{ij} > 0$ only if nodes i and j are neighbors in $\mathbb{T}_N^d$:

Algorithm `GOSSIP-AVE`(P)**:** Each sensor i maintains a real valued variable with initial value $z_i(0) = x_i$. Each sensor has a local Poisson clock that tick independently of other such clocks. At the tick of this clock, say at time t_o, sensor i chooses a neighbor j with respect to the distribution $(P_{ij} : j = 1, 2, \cdots, N^d)$ and both nodes update their internal variables as $z_i(t_o) = z_j(t_o) = (z_i(t_o^-) + z_j(t_o^-))/2$.

Let the vector $z(t) = (z_1(t), z_2(t), \cdots, z_{N^d}(t))$ denote the sensor values at time t, $\bar{x}$ denote the average of $x_1, x_2, \cdots, x_{N^d}$, and $\mathbf{1}$ denote the vector of all 1s. Define τ_k as the kth time instant such that some local clock ticks and thereby triggers messaging in the network. For $\varepsilon > 0$ let the deterministic quantity $K(\varepsilon, P)$ be defined by

$$K(\varepsilon, P) = \sup_{z(0)} \inf \left\{ k \ : \ Pr\left(\frac{\|z(\tau_k) - \bar{x}\mathbf{1}\|_2}{\|z(0)\|_2} \geqslant \varepsilon \right) \leqslant \varepsilon \right\}.$$

Table 1. Comparison of the two algorithms on the d-dimensional torus with N^d nodes, $\mathbb{T}_N^d$

	Energy requirement per node		Termination time	
	COALESCENT$(f, \lambda, \mathbb{T}_N^d)$	GOSSIP-AVE(P)	COALESCENT$(f, \lambda, \mathbb{T}_N^d)$	GOSSIP-AVE(P)
$d = 1$	$\Theta(N)$	$\Omega(N^2)$	$\Theta(N^2)$	$\Omega(N^2)$
$d = 2$	$O((\log N)^2)$	$\Omega(N^2)$	$\Theta(N^2 \log N)$	$\Omega(N^2)$
$d \geq 3$	$O(\log N)$	$\Omega(N^2)$	$\Theta(N^d)$	$\Omega(N^2)$

In [5] $K(\varepsilon, P)$ is considered as a termination time for Algorithm GOSSIP-AVE(P) and minimization of $K(\varepsilon, P)$ is sought by proper choice of P. While the choice of $K(\varepsilon, P)$ as a stopping criterion might be questioned ($\|z(\tau_{K(\varepsilon,P)}) - \bar{x}\mathbf{1})\|_\infty / |\bar{x}|$ may be much larger than ε), here we adopt the same interpretation for this quantity for comparison purposes. The following theorem determines the order of $K(\varepsilon, P)$ uniformly for all P on the torus $\mathbb{T}_N^d$:

Theorem 8. *For fixed $\varepsilon > 0$, $K(\varepsilon, P) = \Omega(N^{d+2})$ uniformly for P such that $P_{ij} > 0$ only if (i, j) is an edge in $\mathbb{T}_N^d$.*

Note that $K(\varepsilon, P)$ is a termination criterion in terms of the transmitted messages in the network. Specifically, $K(\varepsilon, P)$ represents half of the aggregate number of messages transmitted in the network before Algorithm GOSSIP-AVE(P) terminates (to be precise, every time a clock ticks two messages are transmitted in opposite directions on some edge). This observation translates to a termination-time estimate since the point process $(\tau_1, \tau_2, \tau_3, \cdots)$ is Poisson with rate N^d and it takes roughly $K(\varepsilon, P)/N^d$ time units to produce a total of $2K(\varepsilon, P)$ messages. Theorem 8 thus has the following corollary:

Corollary 9. *Algorithm* GOSSIP-AVE(P) *terminates within $\Omega(N^2)$ time on $\mathbb{T}_N^d$.*

The obtained complexity results are summarized in Table 1.

Interestingly, there appears little practical incentive to consider higher dimensions for COALESCENT$(f, \lambda, \cdot)$. Namely Table 1 indicates that in arranging n nodes on $\mathbb{T}^d_{\sqrt[d]{n}}$, the marginal gain in either performance measure by going from d to $d + 1$ is a factor of $O(\log n)$ for $d = 2$, and a factor of $O(1)$ for larger values of d. This observation hints at favorable properties of the 3-dimensional torus and suggests that it is unlikely to experience substantial performance losses due to edge failures. A formal statement of this intuition does not seem straightforward. However the suggested insensitivity to nodal degrees is likely to have important practical implications with regard to robustness, especially in situations where the network topology cannot be planned or controlled.

We next provide numerical results on geometric random graphs, which have received substantial interest as suitable models for wireless ad-hoc networks. See for example [18] and [5]. A geometric random graph $\Gamma = (V_\Gamma, E_\Gamma)$ is obtained by uniformly distributing a set V_Γ of nodes, hence $|V_\Gamma| = n$, on the unit square and drawing an edge between any pair of nodes that fall within distance $r(n) = \sqrt{2 \log n / n}$ of each other. We have first created geometric random graphs and

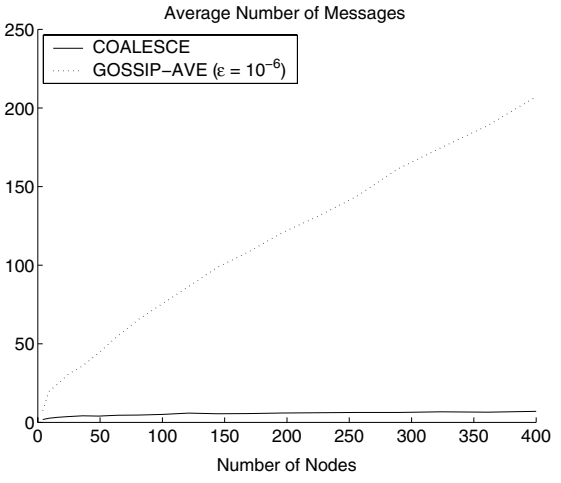

Fig. 3. The average number of transmitted messages per-node on geometric random graphs with n nodes. $\epsilon = 10^{-6}$ for `GOSSIP-AVE`.

discarded the disconnected ones in order to avoid trivialities. Then we have run both `COALESCE` and `GOSSIP-AVE` on the created geometric random graphs 50 times in order to quantify the number of transmitted messages. We have taken $\epsilon = 10^{-6}$ for `GOSSIP-AVE`. The results are plotted in Figure 3. It is observed from Figure 3 that `COALESCE` favors significantly against `GOSSIP-AVE` in terms of number of messages transmitted. The asymptotic properties of `COALESCE` on geometric random graphs as the number of nodes gets large is still ongoing research.

Remark 10. Time and message complexities of Algorithm `GOSSIP-AVE`(P) are inherently related to mixing times of random walks on the torus, and thereby to the spectral gap of the stochastic matrix P. The conclusions of this section should be expected to hold for other distributed algorithms, such as those in [13, 3, 15], that are based on powers of the incidence matrix of an underlying connectivity graph and have similar stopping criteria.

6 Conclusions

This paper concerns distributed algorithms for in-network computation of decomposable functions in sensor networks. Such algorithms typically need to comply with two conflicting requirements: On the one hand expedited convergence of the algorithm is desirable from an application viewpoint. On the other hand limitations due to energy-limited sensors impose frugal usage of energy as an indispensable feature. The emphasis of this paper is the trade-off between time and energy requirements.

We introduced and analyzed a distributed algorithm that is based on coalescing random walk on the communication graph of a sensor network. The

algorithm is asynchronous, requires local information for each sensor node, and it is resilient against packet losses and node failures. In informal terms, the main theme of this algorithm is parsimonious message transmissions. Namely, upon transmitting a message a node enters a quiescent state which lasts until the node receives a message from a neighbor. This operational mode leads to substantial energy savings as the long-term rate of message transmissions at any node decreases in time and tends to the reciprocal of the total number sensors in the network. We show that the algorithm terminates with the exact value of interest, regardless of the network topology. We pursue more detailed analysis on the d-dimensional torus with n nodes and show that the completion time of the algorithm is $\Theta(n(\log n)^{\alpha})$ and energy requirement per sensor node is $O((\log n)^{\alpha+1})$ where $\alpha = 1$ for $d = 2$ and $\alpha = 0$ for $d \geq 3$. The algorithm thus has a scalable energy requirement, furthermore its performance fairly insensitive to the dimension d of the torus so long as $d \geq 2$. This latter observation may prove useful in estimating the performance of the algorithm on less regular network topologies.

We also studied the relative time and energy requirements of the algorithm with respect to other in-network processing algorithms based on powers of the network connectivity graph. In particular we focused on a gossip algorithm for computing averages, and showed that both time and per-node energy requirements scale as $\Omega(n)$ on the 2-dimensional torus irrespectively of the choice of distribution for neighbor selection for pairwise relaxation. Hence the proposed algorithm achieves a factor of $\Omega(n/(\log n)^2)$ gain in energy at the expense of a factor of $O(\log n)$ loss in completion time.

References

1. O. Savas, M. Alanyali and V. Saligrama, Randomized Sequential Algorithms for Data Aggregation in Sensor Networks, CISS 2006, Princeton, NJ.
2. M. Alanyali, V. Saligrama, O. Savas, and S. Aeron. Distributed bayesian hypothesis testing in sensor networks. In *American Control Conference*, Boston, MA, July 2004.
3. V. Saligrama, M. Alanyali and O. Savas. Distributed detection in sensor networks with packet losses and finite capacity links. *IEEE Transactions on Signal Processing*, to appear, 2005.
4. M. Alanyali and V. Saligrama, "Distributed target tracking on multi-hop networks," *IEEE Statistical Signal Processing Workshop*, Bordeaux, France, July 2005.
5. S. Boyd, A. Ghosh, B. Prabhakar and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings of IEEE INFOCOM 2005*, 2005.
6. Fan R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.
7. D. Bertsekas, R. Gallager *Data Networks.* Pearson Education, 1991.
8. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989, republished by Athena Scientific, 1997.
9. J. F. Chamberland and V. V. Veeravalli. Decentralized detection in sensor networks. *IEEE Transactions on Signal Processing*, 2003.
10. J. T. Cox. Coalescing random walks and voter model consensus times on the torus in $\mathbb{Z}^d$. *The Annals of Probability,* 17(4):1333-1366, 1989.

11. A. Giridhar and P. R. Kumar. Computing and communicating functions over sensor networks. *IEEE JSAC Special Issue on Self-Organizing Distributed Collaborative Sensor Networks*, 23, 2005.
12. A. O. Hero and D. Blatt. Sensor network source localization via projection onto convex sets(pocs). In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, March 2005.
13. R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions On Automatic Control,* vol. 49, pp. 1520-1533, Sep., 2004.
14. M. G. Rabbat and R. D. Nowak. Distributed optimization in sensor networks. In *Third International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, April 2004.
15. D.S. Scherber and H.C. Papadopoulos. Distributed computation of averages over ad hoc networks. *IEEE Journal on Selected Areas in Communications,* vol. 23(4), 2005.
16. S. Tavaré. Line-of-descent and genealogical processes, and their applications in population genetics models. *Theoret. Population Biol.,* vol. 26, pp. 119-164, 1984.
17. J. N. Tsitsiklis. Decentralized detection. *in Advances in Statistical Signal Processing, H. V. Poor and J. B. Thomas Eds*, 2, 1993.
18. P. Gupta and P. Kumar, The capacity of wireless networks, IEEE Trans. on Information Theory, 46(2):388-404, March 2000.