Efficiency of the Gossip Algorithm for Wireless Sensor Networks

Elma Zanaj, Marco Baldi, and Franco Chiaraluce

Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni
Università Politecnica delle Marche
Ancona, Italy
E-mail: {e.zanaj, m.baldi, f.chiaraluce}@univpm.it

Abstract: **Gossip is a well-known technique for distributed computing in an arbitrarily connected network of nodes. The gossip algorithm, which is very simple to implement, takes into account strong limitations in computational, communication and energy resources that usually characterize nodes in sensor networks. In gossip, the computational burden is distributed among the nodes, and computation and communication are managed in a very quick and efficient way: in essence, each node acquires its own measure and, starting from it, exchanges information with its neighbors, according with a connection probability distribution. In this paper, we study the performance of gossip algorithms, aiming to test the agreement between analytical results on the averaging time and the actual convergence time for specific scenarios, estimated through numerical simulations. Furthermore, we apply an optimization technique, recently appeared in the literature, that theoretically allow to accelerate convergence. Numerical simulations permit to assess the effects of optimization in real cases, thus evaluating the actual improvement in performance it achieves.**

1. Introduction

Wireless sensors are small devices, characterized by limited communications capabilities due to energy and bandwidth constraints. However, they can merge their scarce resources in networks that aim at supporting very important decisions or actions. They are individually cheap, unintelligent, imprecise and unreliable but, being part of a large network, they together may produce reliable and robust information, as the result of their physical measurements and of a collaborative process. Sensors are seen as nodes of a network that can work with the data they obtain. There are two ways of processing sensors data:

1) elaborating the whole amount of data in a sink node;
2) elaborating data in each node, exchanging information between them; so, at the end, each node possesses the whole information.

We adopt the second model, where nodes are supposed to implement some elementary operations like averages, sums and products.

It is interesting to investigate how such nodes average their values, trying to achieve a general consensus in the shortest possible time. This way, they use a limited amount of local information to allow distributed knowledge of global network properties.

As mentioned above, nodes can elaborate together; so the result is a collective property of the network, and, this way, the area will eventually have uniform information. We mean the nodes talk together, or, in more explicit, though pictorial terms, they "gossip". This kind of gossip is similar to human gossip. For example, when, in a crowd, somebody has a piece of news, he shares it with his neighbors. All the peoples "connected" with him learn the news item and repeat it to their neighbors, so that the new information is spread like an epidemic illness.

A similar process exists in wireless sensor networks, that are peer-to-peer networks formed by many small and simple devices, able to measure some quantities and to transmit their measured values to neighboring nodes. When the whole network (or a subset of it) is expected to produce a unique value for a measured quantity, a group of nodes must communicate in order to merge their single contributions into a common result; in other terms, the network self-stabilizes the measured value [1].

Noting by x_i and x_j the local measures of the i-th and j-th nodes, an interaction among them produces as output $(x_i + x_j)/2$, that is acquired by both nodes and used for the subsequent interaction. Through information propagation, the object is to find, in the shortest possible time, an estimate of the average $\hat{x} = \sum_{i=1}^{N} x_i \Big/ N$, where N is the total number of nodes in the network. As known, the average provides the minimum mean squared error (MMSE) estimate of the sensed quantity. Provided that effective communication among nodes is achieved through suitable protocols, the fundamental aspect of gossip algorithms to be investigated concerns convergence to the average. This topic has been discussed extensively in previous literature [2]-[5]. Most of these papers, however, are focused on the theoretical issues and system modeling. In [2], for example, an analytical framework for the design and study of a randomized distributed averaging problem was presented, together with specific tools for network optimization. In this kind of problems, optimization basically consists in minimizing the time taken for the value at each node to become sufficiently close to the average value, independently of the initial condition. This time is called *averaging time* and noted by

T_{ave} in the following. In [2] it was found that T_{ave} depends on the second largest eigenvalue, λ_2, of a stochastic matrix characterizing the averaging algorithm (this matrix will be defined afterward): the smaller this eigenvalue, the faster the averaging algorithm. So, the fastest averaging algorithm is obtained by minimizing the eigenvalue over the set of allowed gossip algorithms. An efficient procedure was proposed to solve the problem. Moreover, a lower and an upper bound for the averaging time, in terms of λ_2, were derived. Similar, though less explicit, bounds can be found in other papers (see [1] and [6] for example).

In spite of these in-depth analytical studies, however, the performance of the gossip algorithms has been rarely evaluated in simulation experiments. To support the analytical study through simulations is an important issue, as it permits to validate the analytical model and to test the distance between the theoretical expectations and the true results. Moreover, it is also important to quantify the advantage of the analytical optimization procedure, in terms of reduced averaging time, with respect to the non optimized situation.

The aim of this paper is to present some examples of such simulations, and discuss the corresponding results. Our simulations evidence that the effect of optimization is questionable in some practical cases. In Section 2 we introduce the notation and provide a short description of the analytical treatment. In Section 3 we describe the simulator, focusing on the relevant parameters it is based on. Finally, in Section 4 we present some numerical results, and in Section 5 we draw some conclusions.

2. Notation and previous analytical results

A network of N nodes can be described by a connected graph $G(V,E)$, where V is the vertex set containing the nodes and E is the edge set. The class of gossip algorithms we consider is characterized by an $N\times N$ matrix $\mathbf{P}=[P_{ij}]$ of non-negative entries with the condition that $P_{ij}\neq 0$ only if $(i,j)\in E$ and $i\neq j$. We consider an asynchronous time model, in which each node has a clock which ticks at the times of a rate 1 Poisson process. Therefore, the inter-tick times at each node are rate 1 exponentials, independent across nodes and over time. Equivalently, this corresponds to a single clock ticking according to a rate n Poisson process at times Z_k, $k\geq 1$. Time is discretized according to clock ticks, since these are the only times at which the measured values change. Therefore, the interval $[Z_k;Z_{k+1})$ denotes the k-th time-slot and, on average, there are n clock ticks per unit of absolute time. In [2] it is shown the way to pass from quantities measured in terms of the number of clock ticks to quantities expressed in absolute time.

We assume that each node i, when its clock ticks, contacts one of its neighbors j by choosing it according with the connection probability P_{ij}. At each clock tick, node i necessarily communicates; therefore $\mathbf{P}$ is a stochastic matrix (*i.e.* its rows elements sum to 1). Consequently, its largest eigenvalue is equal to 1, while all remaining $N-1$ eigenvalues have magnitude strictly smaller than 1 [7].

Let us denote by $\mathbf{x}(0)=[x_1(0),x_2(0),...,x_N(0)]^T$ a vector collecting the initial (sensed) values at the nodes. So, $x_{ave}=\sum_{i=1}^{N}x_i(0)\Big/N$ is the average of the entries of $\mathbf{x}(0)$; the goal of a gossip algorithm is to compute x_{ave} in a distributed manner. If the clock in the i-th node ticks at the k-th time instant, it contacts one of its neighbors, say node j, with probability P_{ij}. Only one node is contacted at each time, and simultaneous transmissions are avoided. After k ticks, the updated values are collected in vector $\mathbf{x}(k)=[x_1(k),x_2(k),...,x_N(k)]^T$; on the other hand, it is easy to verify that the following recursive relationship holds:

$$\mathbf{x}(k)=\mathbf{W}(k)\cdot\mathbf{x}(k-1)$$

where, with probability P_{ij}/N, the random matrix $\mathbf{W}(k)$ is:

$$\mathbf{W}(k)=\mathbf{W_{ij}}=\mathbf{I}-\frac{(\mathbf{e_i}-\mathbf{e_j})(\mathbf{e_i}-\mathbf{e_j})^T}{2}$$

with $\mathbf{I}$ identity matrix of size $N\times N$, and $\mathbf{e_i}=[0,0,...,0,1,0,...,0,0]^T$ the $N\times 1$ unit vector with the i-th component equal to 1. Now, for a given matrix $\mathbf{P}$, let us define the ε-averaging time of a gossip algorithm as follows [2]:

$$T_{ave}(\varepsilon,\mathbf{P})=\sup_{\mathbf{x}(0)}T_{ave}^{\mathbf{x}(0)}(\varepsilon,\mathbf{P}) \qquad (1)$$

with $T_{ave}^{\mathbf{x}(0)}(\varepsilon,\mathbf{P})=\inf\left\{k:\Pr\left(\frac{\|\mathbf{x}(k)-x_{\text{ave}}\mathbf{1}\|}{\|\mathbf{x}(0)\|}\geq\varepsilon\right)\leq\varepsilon\right\}$, where $\Pr(A)$ stays for the probability of A, $\|\mathbf{v}\|$ denotes the l^2-norm of vector $\mathbf{v}$, and $\mathbf{1}$ is an $N\times 1$ vector with all components equal to 1. In practice, $T_{ave}(\varepsilon,\mathbf{P})$ is the smallest time, expressed in number of clock ticks, it takes for $\mathbf{x}(k)$ to get within ε of $x_{\text{ave}}\mathbf{1}$ with probability $1-\varepsilon$, regardless of the initial value $\mathbf{x}(0)$. If ε is small, then this probability is high.

As mentioned, in [2] it was proved that the ε-averaging time, for the asynchronous time model, can be bounded as follows:

$$\frac{0.5\log e^{-1}}{\log[\lambda_2(\mathbf{C})]^{-1}} \le T_{ave}(\varepsilon,\mathbf{P}) \le \frac{3\log e^{-1}}{\log[\lambda_2(\mathbf{C})]^{-1}}$$

where $\lambda_2(\mathbf{C})$ is the second largest eigenvalue of matrix

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij}\mathbf{W_{ij}}$$

It is evident, from their expressions, that the ratio between the upper bound and the lower bound is always equal to 6, and that their absolute values decrease for decreasing λ_2. This observation introduces the optimization problem, as finding the fastest averaging algorithm corresponds to finding $\mathbf{P}$ such that $\lambda_2(\mathbf{C})$ is the smallest one, while satisfying constraints on $\mathbf{P}$. Formally, the optimization problem is as follows:

- minimize $\lambda_2(\mathbf{C})$,
- subject to $P_{ij} \ge 0$; $P_{ij} \ne 0$ if $(i,j) \in E$; $\sum_j P_{ij} = 1$ for any i.

In [2] a distributed subgradient method was proposed, able to solve the optimization problem over the network. In essence, the solving method can be summarized as follows.

First of all, for the sake of convenience, the non-zero entries of matrix $\mathbf{P}$ are collected in a vector $\mathbf{p}$. Noting by m the number of edges in the graph G, a number l is assigned to each edge (i,j), with $i<j$; this way, the l-th entry of $\mathbf{p}$, with $l = 1,2,...,m$ is $p_l = P_{ij}$. On the other hand, as $\mathbf{P}$ is stochastic (but not, necessarily, double stochastic), there is no symmetry requirement on $\mathbf{P}$, and the entries of $\mathbf{p}$ corresponding to P_{ji} must be separately specified. This is done by setting, $p_{-l} = P_{ji}$. Globally, vector $\mathbf{p}$ has therefore $2m$ entries, that corresponds to replace each edge in the undirected graph G with two directed edges, one in each direction. Then, a classic subgradient method is applied to vector $\mathbf{p}$, which consists in updating this vector, through an iterative procedure, as follows:

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - v_k\mathbf{g}^{(k)}$$

where $\mathbf{g}^{(k)}$ is a subgradient for vector $\mathbf{p}$ at the k-th clock tick ($\mathbf{p}^{(k)}$) and v_k is a step size (to be properly chosen). The l-th component of $\mathbf{g}$ can be obtained as

$$g_l = -\frac{1}{2N}(u_i - u_j)^2$$

where u_i is the i-th component of the unit eigenvector associated with $\lambda_2(\mathbf{C})$, *i.e.* a solution of the equation

$$\lambda_2(\mathbf{C}) = \mathbf{u}^T\mathbf{C}\mathbf{u}$$

Obviously, matrix $\mathbf{C}$ is updated in turn, according with its definition, because of updating of the P_{ij}'s, when the subgradient method proceeds.

On the other hand, it is not sure that the updated values of $\mathbf{p}$ are feasible; for this reason, the subgradient step is followed by a projection onto feasible set step. In practice, noting by $\tilde{p}_{ij}$ the non-zero entries in the i-th row of $\mathbf{P}$, updated according with the subgradient $\mathbf{g}^{(k)}$, it is checked if $\sum_j \tilde{p}_{ij} \le 1$ for $\forall i$; if not, a real δ_i is found such that $\sum_j (\tilde{p}_{ij} - \delta_i) = 1$ and the $\tilde{p}_{ij}$'s changed in $\tilde{p}_{ij} - \delta_i$ [2].

3. Simulation parameters

We have developed numerical programs in Matlab and C++ language that permit:

- to simulate the performance of the gossip algorithm for a given matrix $\mathbf{P}$;
- to optimize matrix $\mathbf{P}$, in such a way as to find a good approximation of the fastest averaging algorithm;
- to compute lower and upper bounds for both cases.

Simulation aims at exploring where the simulated convergence time is located against the lower and upper bounds and, more important, to determine any real (*i.e.* not just theoretically foreseen) improvement achievable through optimization, in terms of reduced convergence time.

The starting values P_{ij} are generated according to an assigned distribution, satisfying the property to have a stochastic matrix. For the gossip algorithm, simulation is also necessary to produce samples of the sensed quantities. For this purpose, we have used different probability distributions. As a first example, we have adopted a gaussian generator, and acted on its parameters to simulate different operational scenarios. This choice is suited to model practical cases, like the measure of the temperature, where the distributed computation strategy can be efficiently applied. As a second example, we have adopted a uniform generator to model more critical situations, with larger dispersion of the measured values, with respect to the previous case.

Once obtained matrix $\mathbf{P}$, simulation proceeds as follows: a uniform random generator selects one node at a time (each node has a probability $1/N$ to be selected). Noting by i the selected node, the node j it contacts is chosen, once again, at random, according with the P_{ij}'s distribution. Changing the seed of the uniform random generator, the

communication sequence is changed as well, this way obtaining different realizations of the considered random experiment.

The random variable $\|\mathbf{x}(k) - x_{ave}\mathbf{1}\| / \|\mathbf{x}(0)\| = e(k)$ is estimated in R experiments, thus obtaining a set of R curves $\varepsilon_{sim}^{r}(k)$, $r = 1...R$, expressed as functions of the number of clock ticks k. This permits to draw $T_{sim}(\varepsilon, \mathbf{P})$ curves that can be compared with the bounds on $T_{ave}(\varepsilon, \mathbf{P})$, as provided by the analytical model. The simulated curves are also averaged among the R realizations, *i.e.*

$$\langle \varepsilon_{sim}(k) \rangle = \frac{1}{R} \sum_{r=1}^{R} \varepsilon_{sim}^{r}(k), \qquad \forall k$$

thus obtaining an estimated average curve $\langle T_{sim}(\varepsilon, \mathbf{P}) \rangle$, intended as a set of (ε, k) couples and referred to the specific matrix $\mathbf{P}$ and the specific (through arbitrary) initial condition $\mathbf{x}(0)$. According to the probability theory, it must be:

$$\lim_{R \to \infty} \langle \varepsilon_{sim}(k) \rangle = \widehat{e(k)}, \qquad \forall k$$

where $\widehat{e(k)}$ represents the average of $e(k)$. In order to prove that the variance of the input vector p.d.f. has no impact on $e(k)$, let us consider a simple network with $N = 3$ nodes, and let us suppose that, because of the assumptions on the p.d.f., the input vector is $\mathbf{x}^a(0) = [x_1(0), x_2(0), x_3(0)]^T$. If, at the first clock tick, node #1 talks with node #2, the vector at $k = 1$ results in:

$$\mathbf{x}^a(1) = \left[x_1(1) = \frac{x_1(0) + x_2(0)}{2}, x_2(1) = x_1(1), x_3(1) = x_3(0) \right]^T$$

Correspondingly, we have:

$$e^a(1) = \frac{\sqrt{\left(x_1(1) - x_{ave}^a\right)^2 + \left(x_2(1) - x_{ave}^a\right)^2 + \left(x_3(1) - x_{ave}^a\right)^2}}{\sqrt{x_1^2(0) + x_2^2(0) + x_3^2(0)}}$$

If the variance of the p.d.f. is scaled by a factor α^2, the random extraction of the input vector would provide $\mathbf{x}^b(0) = \alpha \mathbf{x}^a(0)$. The same scale factor acts on x_{ave}^b, since $x_{ave}^b = \alpha \sum_{i=1}^{N} x_i(0) \Big/ N = \alpha x_{ave}^a$. Under the same assumption that node #1 and node #2 talk between them first, we have $\mathbf{x}^b(1) = \alpha \mathbf{x}^a(1)$; anyway, the value of $e^b(1)$ is unchanged:

$$e^b(1) = \sqrt{\alpha^2 \sum_{i=1}^{3} \left(x_i(1) - x_{ave}^a \right)^2} \Bigg/ \sqrt{\alpha^2 \sum_{i=1}^{3} x_i^2(0)} = e^a(1)$$

From now on, the evolution of the two networks with different input variance proceeds exactly in the same way.

To give an idea of the dispersion of the simulated curves around the mean, the standard deviation is also computed as

$$\sigma(k) = \sqrt{\frac{1}{R-1} \sum_{r=1}^{R} \left(\varepsilon_{sim}^{r}(k) - \langle \varepsilon_{sim}(k) \rangle \right)^2}$$

As known, division by $R-1$, instead of R, makes impartial the estimator. Another parameter we have simulated is $T_{ave}^{\mathbf{x}(0),R}(\varepsilon, \mathbf{P})$, that represents an estimate of $T_{ave}^{\mathbf{x}(0)}(\varepsilon, \mathbf{P})$, for the specific vector of initial values $\mathbf{x}(0)$ and for a finite number (R) of realizations. In fact, though important from a theoretic viewpoint, $T_{ave}(\varepsilon, \mathbf{P})$ is impossible to evaluate in practical cases. Also $T_{ave}^{\mathbf{x}(0)}(\varepsilon, \mathbf{P})$, that is referred to a fixed initial vector $\mathbf{x}(0)$, should be evaluated considering, in principle, an infinite number of realizations.

In order to calculate $T_{ave}^{\mathbf{x}(0),R}(\varepsilon, \mathbf{P})$, the results of gossip simulations are stored in a matrix **S**. Each row of **S** contains the value of $\varepsilon_{sim}^{r}(k)$ for the *r*-th realization; therefore **S** has *R* rows and *K* columns, where *K* is the maximum number of simulated clock ticks (assumed to be the same for all realizations), *i.e.* $1 \le k \le K$.

The estimate of $T_{ave}^{\mathbf{x}(0),R}(\varepsilon, \mathbf{P})$ is the result of a classic quantile evaluation. The value of ε in Eq. (1) is varied within an interval of interest $[\varepsilon_{\min}, \varepsilon_{\max}]$, according with a given step size. The *r*-th row of **S** is scanned, searching for the minimum *k* that gives $\varepsilon_{sim}^{r}(k) < \varepsilon$. The values of $k(r)$ so found are stored in a vector **t** whose elements are sorted in ascending order. The element in position $l = R \cdot (1 - \varepsilon) + 1$ is then identified and used as $T_{ave}^{\mathbf{x}(0),R}(\varepsilon, \mathbf{P})$.

For $R \to \infty$, the curves of $\langle T_{sim}(\varepsilon, \mathbf{P}) \rangle$ and $T_{ave}^{\mathbf{x}(0),R}(\varepsilon, \mathbf{P})$ intersect at the point corresponding to $\varepsilon = 1/2$ [8].

4. Numerical examples

Fig. 1 reports simulation results for a fully mesh network with $N = 50$ nodes. Explicitly, this means that $P_{ij} \neq 0$ for any (*i*, *j*), with $i \neq j$. The values of P_{ij} are obtained through a uniform random generator, under the constraint $\sum P_{ij} = 1$; suitable normalization is applied to satisfy the constraint. The assumption of a fully meshed network is unrealistic in most operational environments, but represents a useful benchmark.

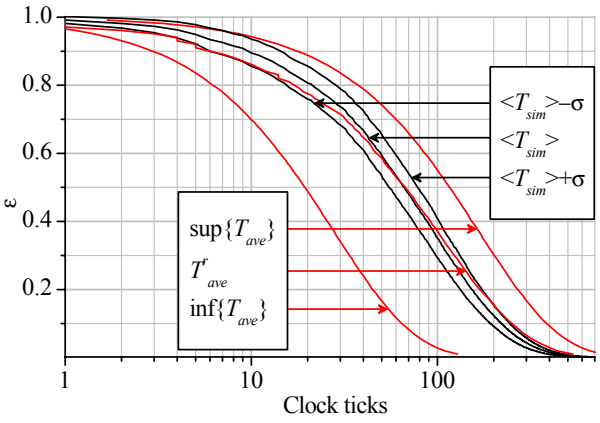

Figure 1. Bounds and simulated results for a fully mesh network with $N = 50$ nodes.

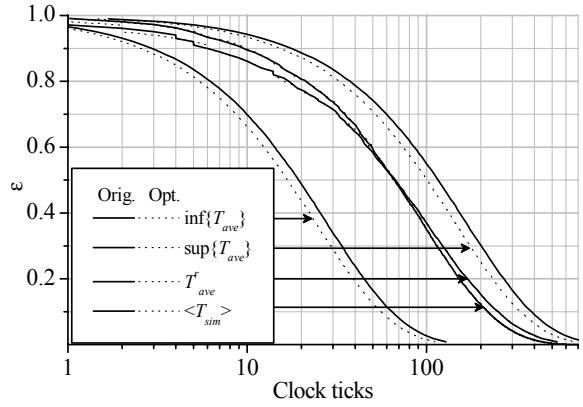

Figure 2. Original (solid lines) and optimized (dotted lines) case for a fully mesh network with $N = 50$ nodes.

As expected, the simulated curves place themselves within the lower and upper bounds, determined through the calculation of $\lambda_2(\mathbf{C})$. In the figure these bounds have been denoted by $inf\{T_{ave}\}$ and $sup\{T_{ave}\}$ respectively. We observe that these bounds are rather loose (because of factor 6 in their ratio). We also note that the curves of $\langle T_{sim}(\varepsilon,\mathbf{P})\rangle$ and $T_{ave}^{\mathbf{x}(0),R}(\varepsilon,\mathbf{P})$ intersect approximately at the point corresponding to $\varepsilon = 0.5$, as theoretically foreseen [8]. The curves $\langle T_{sim}(\varepsilon,\mathbf{P})\rangle \pm \sigma$ have been also plotted in the figure, for the sake of clarity.

Matrix **P** can be optimized according with the subgradient method described in Section 2. This corresponds to minimize the eigenvalue $\lambda_2(\mathbf{C})$, this way reducing both the upper and lower bounds to the averaging time. In Fig. 2 we compare the results already shown in Fig. 1 with those of numerical simulations for the same example, but considering the optimized version of matrix **P**.

From the figure we observe that the optimization process increases, as expected, the slope of the bound curves, thus accelerating, in principle, the averaging process. Nevertheless, the simulated realizations of the gossip algorithm seem scarcely affected by optimization: the curves of $\langle T_{sim}(\varepsilon,\mathbf{P})\rangle$ and $T_{ave}^{\mathbf{x}(0),R}(\varepsilon,\mathbf{P})$, for the two cases, are in fact practically superposed.

5. Conclusions

Based on our numerical evaluations, we can say that the lower and upper bounds to the averaging time are sensitive to the optimization process, as theoretically expected, and can be significantly improved through it. Nevertheless, our simulations demonstrate that there are cases of practical interest where to minimize the eigenvalue, certainly effective as regards the bounds reduction, is not equally effective as regards lowering of the actual averaging time. Obviously, as based on a limited number of experiments, these conclusions cannot have a general meaning, but remain valid in many real cases, since referred to actual scenarios. Future research could adopt more complete network simulator tools in order to evaluate more complex network topologies and conditions and to consider networking issues in a thorough manner.

REFERENCES

[1] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information", in *Proc. IEEE Conference on Foundation of Computer Science*, Cambridge, MA, Oct. 2003, pp. 482–491.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms", *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Analysis and optimization of randomized gossip algorithms", in *Proc. IEEE Conf. Decision and Control*, Nassau, Bahamas, Dec. 2004, pp. 5310–5315.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis, and applications", in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, vol. 3, pp. 1653–1664.

[5] D. Kempe, J. Kleinberg, and A. Demers, "Spatial gossip and resource location protocols", in *Proc. 33rd ACM Symp. Theory of Computing, 2001*, Crete, Greece, July 2001, pp. 163–172.

[6] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading", in *Proc. IEEE Symposium on Foundation of Computer Science*, Redondo Beach, CA, Nov. 2000, pp. 564–574.

[7] M. Hazewinkel, "Encyclopaedia of Mathematics", vol. 9, Springer, 7993.

[8] M. Baldi, F. Chiaraluce, and E. Zanaj, "Analysis and Optimization of the Gossip Algorithm for Wireless Sensor Networks", *in preparation.*