Networking 2013 1569696585

Reducing Communication Overhead for Average Consensus

Mahmoud El Chamie Giovanni Neglia Konstantin Avrachenkov
INRIA Sophia Antipolis - Méditerranée
2004 Route des Lucioles, B.P. 93
06902 Sophia Antipolis, France
{mahmoud.el_chamie, giovanni.neglia, konstantin.avratchenkov}@inria.fr

***Abstract*— An average consensus protocol is an iterative distributed algorithm to calculate the average of local values stored at the nodes of a network. Each node maintains a local estimate of the average and, at every iteration, it sends its estimate to all its neighbors and then updates the estimate by performing a weighted average of the estimates received. The average consensus protocol is guaranteed to converge only asymptotically and implementing a termination algorithm is challenging when nodes are not aware of some global information (e.g. the diameter of the network or the total number of nodes). In this paper, we are interested in decreasing the rate of the messages sent in the network as nodes estimates become closer to the average. We propose a totally distributed algorithm for average consensus where nodes send more messages when they have large differences in their estimates, and reduce their message sending rate when the consensus is almost reached. The convergence of the system is guaranteed to be within a predefined margin η. Tuning the parameter η provides a trade-off between the precision of consensus and communication overhead of the protocol. The proposed algorithm is robust against nodes changing their initial values and can also be applied in dynamic networks with faulty links.**

I. Introduction

Average consensus protocols are used to calculate in a distributed manner the average of a set of initial values (e.g. sensor measurements) stored at nodes in a network. This problem is gaining interest nowadays due to its wide domain of applications as in cooperative robot control [1], resource allocation [2], and environmental monitoring [3], sometimes in the context of very large networks such as wireless sensor networks for which a centralized approach can be unfeasible, see [4]. Under this decentralized approach, each node maintains a local estimate of the network average and selects weights for the estimates of its neighbors. Then at each iteration of the consensus protocol, each node transmits its current estimate to its neighbors and update its estimate to a weighted average of the estimates in its neighborhood. Under some easy-to-satisfy conditions on the selected weights, the protocol is guaranteed to converge asymptotically to the average consensus. For an extensive literature on average consensus protocol and its applications, check the surveys [5], [6] and the references therein.

The asymptotic convergence rate of consensus protocols depends on the selected weights. Xiao and Boyd in [7] have formulated the problem as a Semi Definite Program that can be efficiently and **globally** solved. However, speeding up the convergence rate does not automatically reduce the number of messages that are sent in the network. The reason is that the convergence is reached only asymptotically, and even if nodes' estimates are very close to the average, nodes keep on performing the averaging and sending messages to their neighbors.

In this paper we propose an algorithm that relies only on limited local information to reduce communication overhead for average consensus. As the nodes' estimates approach the true average, nodes exchange messages with their neighbors less frequently. The algorithm has a nice self-adaptive feature: even if it has already converged to a stable state and the message exchange rate is very small, when an exogenous event leads the value at a node to change significantly, the algorithm detects the change and ramps up its communication rate. The proposed algorithm provides also a trade-off between the precision of the estimated average and the number of messages sent in the network by setting the parameter η. Being totally decentralized, the message reduction algorithm can also be applied in a dynamic network with faulty links.

The paper is organized as follows: Section II presents the notation used and a formulation of the problem. Section III describes the previous work on the termination of the average consensus protocol. Section IV motivates the work by an impossibility result for finite time termination. Section V presents the proposed algorithm, its analysis, and the simulations of the algorithm. Section VI concludes the paper.

II. Problem Formulation

Consider a network of n nodes that can exchange messages among each other through communication links. Every node in this network stores a local value, e.g. a sensor measurement of pressure or temperature, and we need each node to calculate the average of the initial measurements by following a distributed linear iteration approach. The network of nodes can be modeled as a graph $G = (V, E)$ where V is the set of vertices ($V = \{1, \ldots, n\}$) and E is the set of edges such that $\{i, j\} \in E$ if nodes i and j are connected and can communicate (they are neighbors). Let also N_i be the neighborhood set of node i. Let $x_i(0) \in \mathbb{R}$ be the initial

value at node i. We are interested in computing the average $x_{ave} = (1/n)\sum_{i=1}^{n} x_i(0)$, in a decentralized manner with nodes only communicating with their neighbors. We consider that nodes operate in a synchronous way: when the clock ticks, all nodes in the system perform the iteration of the averaging protocol. In particular at iteration $k+1$, node i updates its state value x_i:

$$x_i(k+1) = w_{ii}x_i(k) + \sum_{j \in N_i} w_{ij}x_j(k) \tag{1}$$

where w_{ij} is the weight selected by node i for the value sent by its neighbor j and w_{ii} is the weight selected by node i for it own value. The topology of the network may change dynamically. This can be easily taken into account in (1) by letting the neighborhood and the weights be time-dependent (then we have $N_i(k)$ and $w_{ij}(k)$). For the sake of simplicity, in what follows we omit this explicit dependance. The matrix form equation is:

$$\mathbf{x}(k+1) = W\mathbf{x}(k) \tag{2}$$

where $\mathbf{x}(k)$ is the state vector of the system and W is the $n \times n$ weight matrix such that $(W)_{i,i} = w_{ii}$, $(W)_{i,j} = w_{ij}$ if $j \in N_i$, and $(W)_{i,j} = 0$ if $j \notin N_i \cup \{i\}$.

In this paper, we consider W to be $n \times n$ real doubly stochastic matrix having $\lambda_2(W) < 1$ where λ_2 is the second largest eigenvalue in magnitude[1] of W (these are sufficient conditions for the convergence of the average consensus protocol, see [8]). We also consider that W is constructed locally, some methods for constructing the weight W using only local information can be found in [9]. Let $\mathbf{1}$ be the vector of all 1s, the convergence to the average consensus is in general asymptotic:

$$\lim_{k\to\infty} \mathbf{x}(k) = x_{ave}\mathbf{1}. \tag{3}$$

Since average consensus is usually reached only asymptotically (3), the nodes will always be busy sending messages. Let $N(k)$ be the number of nodes transmitting at iteration k, so without a termination procedure all nodes are transmitting at iteration k, $N(k) = n$ independently from the current estimates. In this paper we present an algorithm that reduces communication overhead and provides a trade-off between precision of the consensus and number of messages sent.

III. Related Work

Some previous works considered protocols for average consensus protocol to terminate (in finite time) to converge to the exact average or to guaranteed error bounds. For example, the approach proposed in [10] is based on the *minimal polynomial* of the matrix W. The authors show that a node, by using coefficients of this polynomial, can calculate the exact average from its own estimate on K consecutive iterations. The drawback is that nodes must have high memory capabilities to store $n \times n$ matrix, and high processing capabilities to calculate the coefficients of the minimal polynomial by solving a set of n linearly independent equations. Another approach for finite time termination is given in [11], where the proposed algorithm does not calculate the exact average, but estimates are guaranteed to be within a predefined distance from the average. This approach runs three consensus protocols at the same time: the average consensus which runs continuously and the maximum and the minimum consensus restarted every U iterations where U is an upper bound on the diameter of the network. The difference between the maximum and the minimum consensus provides a stopping criteria for nodes.

Under the assumption of asynchronous iterations, the authors in [12] proposed an algorithm that leads to the termination of average consensus in finite time with high probability. In their approach, each node has a counter c_i that stores the number of times the difference between the new estimate and the old one was less than a certain threshold τ. When the counter reaches a certain value, say C, the node will stop initiating the algorithm. They proved that by a correct choice of C and τ (depending on some networks' parameters as the maximum degree in the network, the number of nodes, and the number of edges) the protocol terminates with high probability.

A major drawback of these algorithms —beside the memory requirements and the robustness of the system to changes— is the assumption that each node should know some global network parameters. This intrinsically contradicts the spirit of distributed consensus protocols. Designing a decentralized algorithm for average consensus that terminates in finite time without using any global network information (as the diameter of the network or the number of nodes) is still an open problem for which we prove a strong negative result in the next section.

IV. Motivation

We address the problem of termination of average consensus in this paper. We will start by an impossibility result for termination of the average consensus protocol in finite time without using some network information.

Theorem 1. *Given a static network where nodes run the synchronous consensus protocol described by* (1) *and each node only knows its history of estimates, there is no deterministic distributed algorithm that can correctly terminate the consensus with guaranteed error bounds after a finite number of steps for any set of initial values.*

Proof: The proof is conducted by contradiction where we show that there exists a graph with specific initial state values which fails to terminate with guaranteed error bounds. Consider a path graph G of three nodes a, b, and c as in Fig. 1 where the weight matrix is real and doubly stochastic with $0 \leq \lambda_2(W) < 1$ (so we have $w_{aa}, w_{cc} > 0$). Let $x_a(0)$, $x_b(0)$, and $x_c(0)$ be the initial estimates for the nodes and consider $\alpha = \frac{x_a(0)+x_b(0)+x_c(0)}{3}$, so with the average consensus protocol using the synchronous iterations in (1), all nodes' estimates will converge to α asymptotically:

$$\lim_{k\to\infty} x_a(k) = \lim_{k\to\infty} x_b(k) = \lim_{k\to\infty} x_c(k) = \alpha.$$

[1]The second largest largest eigenvalue in magnitude of a symmetric matrix is the second largest singular value of that matrix so $\lambda_2 \geq 0$.

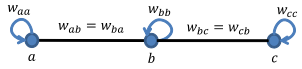

Fig. 1. Path graph G with 3 nodes.

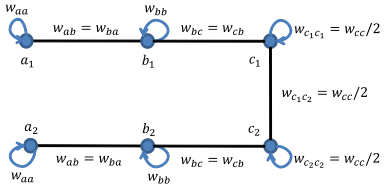

Fig. 2. Extended mirror graph of G with 6 nodes and $F = 2$ fragments.

We will prove the theorem by contradiction. Suppose there exists a termination algorithm for nodes to use only the history of their estimates and terminate the average protocol in finite time within guaranteed error bounds. Then if we run this algorithm on this graph, there exists an iteration $K > 0$ and $\eta > 0$ such that node a (also true for b and c) decides to terminate at iteration K on the basis of the history of its estimate: $x_a(0), x_a(1), x_a(2), ..., x_a(K)$, and it is guaranteed that $|x_a(K) - x_{ave}| < \eta$, where $x_{ave} = \alpha$.

We will define the F extended mirror graph of G to be a path with $n = 3F$ nodes $a_1, a_2, ..., a_F, b_1, b_2, ..., b_F, c_1, c_2, ..., c_F$, formed by $G_1, G_2, ..., G_F$ (F graphs identical to G) connected by additional links to form a path, the added links are $\{c_l, c_{l+1}\}$ if l is odd and $\{a_l, a_{l+1}\}$ if l is even (e.g. the graph for $F = 2$ is shown in Fig. 2). Let us assume first that the initial estimates for nodes in the subgraphs G_1,...,G_F are identical to the estimates of the nodes in graph G (e.g. for node a we have $x_{a_1}(0) = x_{a_2}(0) = ... = x_{a_F}(0) = x_a(0)$), the weight matrix for G_1,...,G_F is also identical to the weight matrix of G except for nodes incident to the added links, if $\{c_l, c_{l+1}\}$ is an added link, then $w_{c_l c_l} = w_{c_{l+1}c_{l+1}} = w_{c_l c_{l+1}} = \frac{w_{cc}}{2}$ and similarly if $\{a_l, a_{l+1}\}$ is an added link, then $w_{a_l a_l} = w_{a_{l+1}a_{l+1}} = w_{a_l a_{l+1}} = \frac{w_{aa}}{2}$. Notice that on the new generated graph we still have $x_{ave} = \alpha$ and also:

$$x_{a_1}(k) = x_{a_2}(k) = ... = x_{a_F}(k) = x_a(k) \ \forall k \leq K,$$

so node a_1 applying the termination algorithm on the new graph will decide to terminate after the same number of iterations K. Consider now a value $F > K$ and that the initial estimate of node c_{K+1} is changed to $x_{c_{K+1}}(0) = x_c(0) + n(2\eta + x_{a_1}(K) - \alpha)$ and the new average is now $x_{ave} = \alpha + \frac{x_{c_{K+1}}(0) - x_c(0)}{n}$. The estimates at node a_1 would not change during the first K steps, then node a_1 would again terminate at step K, but the error bound is no more guaranteed, because $|x_{a_1}(K) - x_{ave}| = |x_{a_1}(K) - \alpha - \frac{x_{c_{K+1}}(0) - x_c(0)}{n}| = 2\eta > \eta$. This contradicts the fact that a_1 terminates with **guaranteed** error bounds. The proof can be extended to include any graph G, not just path graphs, by using the same technique of generating extended mirror graphs of G. ■

Theorem 1 shows that in general, nodes cannot stop executing the algorithm. Motivated by this result, we investigate in what follows algorithms where nodes can refrain from sending messages at every iteration (e.g. when estimates have not changed significantly during the recent iterations). We will then say that an algorithm terminates when the number of messages sent in the network disappears at least asymptotically, even if the nodes are still running the algorithm internally, i.e.:

$$\lim_{t \to \infty} \frac{\sum_{k=1}^{t} N(k)}{t} = 0, \tag{4}$$

where $N(k)$ is the number of nodes transmitting their estimate to their neighbors at iteration k. In other words, the rate of messages in the network should decrease as the estimates converge to the average consensus or to a bounded approximation.

V. Our Approach

Even if the nodes cannot terminate the algorithm in finite time, we are interested in reducing communication overhead by considering asymptotic termination of messages and by decreasing the rate of the messages sent in the network correspondingly to estimates' improvement. For example, if nodes' estimates are widely different, the messages sent at a given iteration can significantly reduce the error by making the estimates approach to the real average. However, when the estimates have "almost converged", the improvement from each message in terms of error reduction can be negligible. Up to our knowledge, this issue was not taken into account in the related work literature. So from an engineering perspective, it is desirable that nodes send more messages when they have large differences in their estimates, and less messages when the estimates have almost converged. In what follows we first present a centralized algorithm to provide the intuition of our approach and then we describe a more practical decentralized solution.

A. A Centralized Algorithm

In this section, we discuss a simple centralized algorithm for termination of average consensus protocols. We call it a centralized protocol because in this protocol there are some global variables known to all the nodes in the network, and each node can send a broadcast signal that triggers an averaging operation (1) at all nodes. Then, if any of the nodes in the network sends this signal, all the nodes will respond by sending the new estimates to their neighbors according to the averaging equation (2): $\mathbf{x}(t+1) = W\mathbf{x}(t)$.

On the contrary, if no signal is sent, the nodes will preserve the same estimate:

$$\mathbf{x}(t+1) = \mathbf{x}(t). \tag{5}$$

If the rate of broadcast signals converges to 0, also the rate of the messages containing the estimates will converge to 0 and asymptotically no node in the network will transmit. As above we consider a time-slotted model where t represents a discrete time iteration.

We now introduce formally the algorithm. Let $e(t)$ and $\eta(t)$ be the values of two global variables known to all the nodes at time t, such that $e(0) = 0$, $\eta(0) = \eta_0$ and $0 \leq e(t) < \eta(t)$. As we are going to see, both the values of the two variables cannot decrease. Let W be the weight matrix of the network satisfying convergence conditions of average consensus and $\mathbf{x}(t)$ be the state vector of the system at iteration t. We let L_t be a Boolean variable (either true or false) defined at every iteration t as:

$$L_t : \ e(t-1) + y(t-1) < \eta(t-1), \tag{6}$$

where $y(t-1) = ||W\mathbf{x}(t-1) - \mathbf{x}(t-1)||_\infty$ and with $L_0 := False$. Then $y(t-1)$ stores the estimates change if the linear iterations (1) would be executed at step t and L_t evaluates if the change is negligible ($L_t = False$) and then no message is transmitted or not ($L_t = True$). Different actions are taken on the basis of the L_t value at timeslot t. We also define the simple point process $\psi = \{t_k : k \geq 1\}$ to be the sequence of strictly increasing points

$$0 < t_1 < t_2 < \dots ,$$

such that $t_k \in \psi$ if and only if $L_{t_k} = False$. Let $K(t)$ denote the number of points of the set ψ that falls in the interval $]0, t]$, i.e. $K(t) = \max\{k \ : \ t_k \leq t\}$, with $K(0) := 0$. If L_t is false, a broadcast signal is sent in the network and all nodes will perform an averaging iteration; while if L_t is true, then there is no signal in the network, and the nodes keep the same estimate as the previous iteration. Network variables of the centralized algorithm are changed at time $t > 0$ according to the equations given in following table:

If L_t is $True$	If L_t is $False$
$K(t) = K(t-1)$	$K(t) = K(t-1) + 1$
$\mathbf{x}(t) = \mathbf{x}(t-1)$	$\mathbf{x}(t) = W\mathbf{x}(t-1)$
$e(t) = e(t-1) + y(t-1)$	$e(t) = e(t-1)$
$\eta(t) = \eta(t-1)$	$\eta(t) = \eta(t-1) + \frac{\eta_0}{K(t)^2}$

When $t \notin \psi$, we call t a silent iteration because the nodes have the same estimate as the previous iteration (i.e. $x_i(t) = x_i(t-1)$) and there is no need to exchange messages of these estimates in the network. On the other hand, when $t \in \psi$, we call t as a busy iteration because nodes will perform an averaging (i.e. $\mathbf{x}(t) = W\mathbf{x}(t-1)$) and the estimates must be exchanged in the network. Let α_k be the number of silent iterations between t_k and t_{k+1}, so we have that $\alpha_k = t_{k+1} - t_k - 1$[2].

After introducing this deterministic procedure, we show by the following lemma that the messages according to this algorithm disappear asymptotically:

Proposition 1. *For any initial condition $\mathbf{x}(0)$, the message rate of the centralized deterministic algorithm described above disappears asymptotically, i.e.:*

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{t} N(k)}{t} = 0,$$

where $N(k)$ is the number of nodes transmitting messages at iteration k.

Proof: The number of nodes transmitting at an iteration t depends on the condition L_t. If $t \in \psi$, then $N(t) = n$ (all nodes are transmitting messages), otherwise $N(t) = 0$ (no nodes transmitting messages). Therefore,

$$\sum_{k=1}^{t} N(k) = \sum_{k=1}^{K(t)} N(t_k) = nK(t),$$

where $K(t)$ as described earlier is the number of busy iterations until time t. We will consider two cases depending on the evolution of $K(t)$ as function of t. The simpler case is when $\lim_{t\to\infty} K(t) \leq K < \infty$ (the number of busy periods is bounded, e.g. nodes reach consensus in a finite number of iterations), then since $K(t)$ is an increasing positive integer sequence, the proposition follows from the following inequality and $t \to \infty$,

$$0 \leq \frac{\sum_{k=1}^{t} N(k)}{t} \leq \frac{nK}{t}.$$

We consider now the other case, i.e. $\lim_{t\to\infty} K(t) = \infty$. Notice that for any time iteration t, we have

$$\sum_{k=1}^{K(t)} (t_k - t_{k-1}) \leq t \leq \sum_{k=1}^{K(t)+1} (t_k - t_{k-1}),$$

or in other words

$$\sum_{k=0}^{K(t)-1} (\alpha_k + 1) \leq t \leq \sum_{k=0}^{K(t)} (\alpha_k + 1).$$

So we have

$$\frac{\sum_{k=1}^{t} N(k)}{t} = \frac{nK(t)}{t} \leq \frac{nK(t)}{\alpha_{K(t)-1} + 1}$$

We will prove now that the right hand side of the inequality goes to 0 as t diverges. Since $\lim_{t\to\infty} K(t) = \infty$, it is sufficient to prove that $\lim_{k\to\infty}(\alpha_k + 1)/k = \infty$. Let $\mathbf{z}(k) = W\mathbf{x}(t_k) - \mathbf{x}(t_k)$, we can see that according to this algorithm,

$$\begin{aligned} \alpha_k &= \lfloor \frac{\eta(t_k) - e(t_k)}{||\mathbf{z}(k)||_\infty} \rfloor \\ &\geq \frac{\eta(t_k - 1) + \eta_0/k^2 - e(t_k - 1)}{||\mathbf{z}(k)||_\infty} - 1 \\ &\geq \frac{\eta_0}{k^2 ||\mathbf{z}(k)||_2} - 1. \end{aligned} \tag{7}$$

The last inequality derives from the fact that for any iteration t we have $\eta(t) > e(t)$, and that for any vector $\mathbf{v}$, the norm inequality $||\mathbf{v}||_2 \geq ||\mathbf{v}||_\infty$ holds. Moreover, $\mathbf{z}(k)$ evolves according to the following equation:

$$\begin{aligned} \mathbf{z}(k) &= (W - J)\mathbf{z}(k-1) \\ &= (W - J)^k \mathbf{z}(0), \end{aligned}$$

where $J = 1/n\mathbf{1}\mathbf{1}^T$, so

$$||\mathbf{z}(k)||_2 \leq C \left(\rho(W - J)\right)^k, \tag{8}$$

[2] $t_k - t_{k-1}$ is sometimes called the k^{th} interarrival time in the context of point processes.

where $C = ||\mathbf{z}(0)||_2$ and $\rho(W - J) = \lambda_2(W) \geq 0$ is the spectral radius of the matrix $W - J$. We know that $0 < \lambda_2 < 1$ ($0 < \lambda_2$ because $\lim_{t\to\infty} K(t) = \infty$ and $\lambda_2 < 1$ because W satisfies the condition of a converging matrix (see [8])). Putting everything together, we get finally that:

$$\alpha_k \geq \frac{\eta_0}{Ck^2\lambda_2^k} - 1, \tag{9}$$

and

$$\frac{\alpha_k + 1}{k} \geq \frac{\eta_0}{Ck^3\lambda_2^k},$$

hence $(\alpha_k + 1)/k \to \infty$ as $k \to \infty$. Consequently, the rate of messages sent in the network vanishes, namely

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{t} N(k)}{t} = 0.$$

■

Three main factors in the above algorithm cause the algorithm to be centralized: the global scalar $e(t)$, the global scalar $\eta(t)$, and the broadcast signal. In the following sections, we will present a decentralized algorithm inspired by the centralized one, but all global scalars are changed to local ones, and the nodes are not able to send a broadcast signal to trigger an iteration.

B. Decentralized Environment

1) Modified Settings: The analysis of the system becomes more complicated when we deal with the decentralized scenario. Each node works independently. We keep the assumption of synchronous operation, but the decision to transmit or not is local, so a node can be silent, while its neighbor is not. In this scenario, even the convergence of the system might not be guaranteed and we see that within an iteration, some nodes will be transmitting and others will be silent. This can cause instability in the network because the average of the estimates at every iteration is now not conserved (this is an important property of the standard consensus protocols that can be easily checked), and the scalars $\eta(k)$ and $e(k)$ defined in the previous subsection are now vectors $\eta(k)$ and $\mathbf{e}(k)$ where $\eta_i(k)$ and $e_i(k)$ are the values corresponding to a node i and are local to every node. To conserve the average in the decentralized setting, $\mathbf{e}(k)$ must take part in the state equation as we will show in what follows.

2) System Equation: In our approach, we consider a more general framework for average consensus where we study the convergence of the following equation:

$$\mathbf{x}(k+1) + \mathbf{e}(k+1) = W\mathbf{x}(k) + \mathbf{e}(k). \tag{10}$$

Some work has studied the following equation as a perturbed average consensus and considered $\mathbf{e}(k)$ to be zero mean noise with vanishing variance (see [13], [14]). However, in our model, we consider $\mathbf{e}$ as a deterministic part of the state of the system and not a random variable. We consider sufficient conditions for the system to converge, and we use these conditions to design an algorithm that can reduce the number of messages sent in the network.

In the standard consensus algorithms, the state of the system is defined by the state vector $\mathbf{x}$, but in the modified system, the state equation is defined by the couple $\{\mathbf{x}, \mathbf{e}\}$. In the following we present a key theorem for the convergence in a decentralized setting.

Theorem 2. *Consider a system governed by the equation* (10)*, let* $F(k) = F(k, \mathbf{x}(k), \mathbf{x}(k-1))$ *be a matrix that depends on the iteration* k *and two history state vectors* $\mathbf{x}(k)$ *and* $\mathbf{x}(k-1)$*. Suppose that* $\mathbf{e}(k+1) = \mathbf{e}(k) - F(k)\mathbf{x}(k)$ *and assume the following conditions on the matrices* $A(k) = W + F(k)$ *and* $F(k)$*:*

(a) $a_{ij}(k) \geq 0$ *for all* i*,* j*, and* k*, and* $\sum_{j=1}^{n} a_{ij}(k) = 1$ *for all* i *and* k*,*
(b) *Lower bound on positive coefficients: there exists some* $\alpha > 0$ *such that if* $a_{ij}(k) > 0$*, then* $a_{ij}(k) \geq \alpha$*, for all* i*,* j*, and* k*,*
(c) *Positive diagonal coefficients:* $a_{ii}(k) \geq \alpha$*, for all* i*,* k*,*
(d) *Cut-balance: for any* i *with* $a_{ij}(k) > 0$*, we have* j *with* $a_{ji}(k) > 0$*,*
(e) $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star \Rightarrow \lim_{k\to\infty} F(k, \mathbf{x}(k), \mathbf{x}(k-1))\mathbf{x}(k) = \mathbf{0}$.

Then $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$ *where* $x'_{ave} \in [\min_j x_j(0), \max_j x_j(0)]$*; if furthermore* $\mathbf{e}(0) = \mathbf{0}$ *and* $e_i(k) < \eta$ *for all* i *and* k*, then* $|x_{ave} - x'_{ave}| < \eta$*.*

Proof: Let us first prove that $\mathbf{x}(k)$ converges. By substituting the equation of $\mathbf{e}(k+1)$ in (10), we obtain:

$$\mathbf{x}(k+1) = A(k)\mathbf{x}(k), \tag{11}$$

where $A(k) = W + F(k)$. From the conditions (a),(b),(c), and (d) on $A(k)$, we have from [15] that $\mathbf{x}$ converges, i.e. $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star$. Since the system is converging, then from equation (10), we can see that:

$$\begin{aligned}\mathbf{x}^\star &= W\mathbf{x}^\star + \lim_{k\to\infty} (\mathbf{e}(k) - \mathbf{e}(k+1)) \\ &= W\mathbf{x}^\star + \lim_{k\to\infty} F(k)\mathbf{x}(k) \\ &= W\mathbf{x}^\star.\end{aligned}$$

Therefore, $\mathbf{x}^\star$ is an eigenvector corresponding to the highest eigenvalue ($\lambda_1 = 1$) of W. So we can conclude that $\mathbf{x}^\star = x'_{ave}\mathbf{1}$ where x'_{ave} is a scalar (Perron-Frobenius theorem).

The condition $\mathbf{1}^T W = \mathbf{1}^T$ on the matrix W in equation (2) leads to the preservation of the average in the network, $\mathbf{1}^T\mathbf{x}(k) = nx_{ave}\ \forall k$. This condition is not necessary satisfied by $A(k)$, so let us prove now that the system preserves the average x_{ave}:

$$\mathbf{1}^\top(\mathbf{x}(k+1) + \mathbf{e}(k+1)) = \mathbf{1}^\top(W\mathbf{x}(k) + \mathbf{e}(k)) \tag{12}$$

$$= \mathbf{1}^\top(\mathbf{x}(k) + \mathbf{e}(k)). \tag{13}$$

The last equality comes from the fact that W is sum preserving since $\mathbf{1}^\top W = \mathbf{1}^\top$.

Finally by a simple recursion we have that $\mathbf{1}^\top(\mathbf{x}(k) + \mathbf{e}(k)) = \mathbf{1}^\top\mathbf{x}(0) = nx_{ave}$, and the average is conserved. Moreover, since $|e_i(k)| \leq \eta$ for all i and k we have:

$$|(1/n)\mathbf{1}^\top\mathbf{x}(k) - x_{ave}| \leq \eta\ \forall k. \tag{14}$$

But we just proved that $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$, so this consensus is within η from the desired x_{ave}:

$$|x'_{ave} - x_{ave}| \leq \eta. \tag{15}$$

This ends the proof. ■

In the decentralized environment, we gave the conditions for the system to converge. In the following section we will design an algorithm that satisfies these conditions and needs only local communications.

C. Message Reducing Algorithm

We try to solve the termination problem through a *fully decentralized* approach. We consider large-scale networks where nodes have limited resources (in terms of power, processing, and memory), do not use any global estimate (e.g. diameter of the network or number of nodes), keep only one iteration history, and can only communicate with their neighbors. Our goal is to reduce the number of messages sent while guaranteeing that the protocol converges within a given margin from the actual average.

The main idea is that a node, say i for example, will compare its new calculated value with the old one. According to the change in the estimate, i will decide either to broadcast its new value or not to do so. We divide an iteration into two parts, in the first part of the iteration, only nodes with significant change in their estimates are allowed to send messages. However, in the second part of the iteration, only nodes polled by their neighbors from phase 1 are allowed to send an update.

Before starting the linear iterative equation, nodes will select weights as in the standard consensus algorithm. The weight matrix considered here must be doubly stochastic with $0 < \alpha < w_{ii} < 1 - \alpha < 1$ for some constant α. Each node i in the network keeps two state values at iteration k:

- $x_i(k)$: the estimate of node i used in the iterative equations by the other nodes.
- $e_i(k)$: a real value that monitors the shift from the average due to the iterations where node i did not send a message to its neighbors. It is initially set to zero, $e_i(0) = 0$.

Each node also keeps its own boundary threshold $\eta_i(k)$ where $\eta_i(1) = \frac{\eta}{2} = constant\ \forall i$. Note that $\eta_i(k)$ is increased after every transmission as in the centralized case, but the difference here is that it is local to every node.

Each iteration is divided into two phases:
In the first phase, a node i can be in one of the two following states:

- *Transmit*: The set of nodes corresponding to this state is T_k, where the subindex k corresponds to the fact that the set can change with every iteration k. The nodes in T_k send their new calculated estimate to their neighbors. They also poll the nodes having maximum and minimum estimates in their neighborhood to transmit in phase 2.
- *Wait*: The set of nodes corresponding to this state is W_k. The node's decision will be taken in the second phase of the iteration based on the action of nodes in the *Transmit* state (depending if they were polled by any of their neighbors).

Algorithm 1 Termination Algorithm -node i- Phase 1

$\{x_i(k), e_i(k)\}$ are the state values of node i at iteration k, $0 < \alpha < w_{ii} < 1-\alpha < 1$, $counter_i = 1$ is the counter for the number of transmissions so far. $\eta_i(1) = \eta/2 \in \mathbb{R}$, T_k is set of $Transmit$ state. W_k set corresponding to $Wait$ state. Initially we have $T_k = W_k = \varnothing$. Every node i follows the following algorithm at iteration k.
$y_i(k+1) \leftarrow w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k)$
$d_i \leftarrow y_i(k+1) - x_i(k) + e_i(k)$
if $|d_i| < \eta_i(counter_i)$ **then**
  i changes to a *Wait* state. $\setminus\setminus\ i \in W_k$
else
  $counter_i = counter_i + 1$
  $\eta_i(counter_i) = \eta_i(counter_i - 1) + \eta_i(1)/counter_i^2$
  $c_i \leftarrow \frac{\alpha}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big)$
  if $|c_i| \leq |e_i(k)|$ **then**
    $x_i(k+1) \leftarrow y_i(k+1) + \text{sign}(c_i.e_i(k))c_i$
    $e_i(k+1) \leftarrow e_i(k) - \text{sign}(c_i.e_i(k))c_i$
  else
    $x_i(k+1) \leftarrow y_i(k+1) + e_i(k)$
    $e_i(k+1) \leftarrow 0$
  end if
  i changes to a *Transmit* state. $\setminus\setminus\ i \in T_k$
  Notify the neighbors having maximum and minimum values.
end if
Go to Phase 2

In the second part of the iteration, nodes that are in W_k will be classified as follows:

- *Silent*: The set of nodes corresponding to this state is S_k. These are the nodes that will remain silent with no message sent from their part in the network. The nodes in S_k have that non of their neighbors sending them any poll message.
- *Cut-Balance*: The set of nodes corresponding to this state is B_k. They are called $Cut{-}Balance$ because they insure the cut-balance condition (d) of Theorem 2. They are the nodes in W_k that have been polled by at least one neighbor in T_k .

The two phases of the termination protocol implemented at each node are described by pseudocode in Algorithm 1 and 2. Nodes in the T_k set (the set of nodes that are in a $Transmit$ state) will broadcast their estimate to their neighbors at the end of the first phase, while nodes in W_k set (or $Wait$ state) will postpone their decision to send or not till the next phase. Nodes that do not receive a message from their neighbors at a certain iteration, use the last seen estimate from the specified neighbors (note: absence of messages from a neighbor during an iteration does *not* mean the failure of link, it means that the neighbor is broadcasting the same old estimate as before, so we may differentiate the link failure by a "keep alive"

Algorithm 2 Termination Algorithm - Phase 2

1: $\{x_i(k), e_i(k)\}$ are the state values of node i at iteration k.
2: **for all** nodes i having *Wait* state **do**
3: $y_i(k+1) \leftarrow w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k)$
4: **if** i received a poll message from any neighbor **then**
5: $z_i(k+1) \leftarrow (w_{ii} + \sum_{j\in N_i\cap W_k} w_{ij})x_i(k) + \sum_{j\in N_i\cap T_k} w_{ij}x_j(k)$
6: $x_i(k+1) \leftarrow z_i(k+1)$
7: $e_i(k+1) \leftarrow y_i(k+1) - z_i(k+1) + e_i(k)$
8: i changes to a $Cut-Balance$ state. $\setminus\setminus\, i \in B_k$
9: **else**
10: $x_i(k+1) \leftarrow x_i(k)$
11: $e_i(k+1) \leftarrow y_i(k+1) - x_i(k) + e_i(k)$
12: i changes to a *Silent* state. $\setminus\setminus\, i \in S_k$
13: **end if**
14: **end for**
15: $k+1 \leftarrow k$

message sent frequently to maintain connectivity and set of neighbors). The **input** for the algorithm are the estimates of the neighbor of i, the weights selected for these neighbors, and the state values $\{x_i(k), e_i(k)\}$. The **output** of the first phase is the new state values $\{x_i(k+1), e_i(k+1)\}$ for nodes in T_k and the output of the second phase is the new state values $\{x_i(k+1), e_i(k+1)\}$ for nodes in W_k. Let us go through the lines of the algorithm. In phase 1, $y_i(k+1)$ of line 2 is the weighted average of the estimates received by node i; without the termination protocol this value would be sent to all its neighbors. The protocol evaluates how much $y_i(k+1)$ differs from the state value $x_i(k)$. This difference accumulates in d_i in line 3. If this shift is less than a given threshold η_i, the node will wait for next phase to take decision. If the condition in line 4 is not satisfied, that means the node will send a new value to its neighbors. Lines $7-8$ concerns the extending of the boundary threshold $\eta_i(k)$ after every transmission. Note that by this extension method, we have $\eta_i(k) < \eta\ \forall i,k$ since

$$\lim_{k\to\infty} \eta_i(k) = \eta_i(1)(\sum_{i=1}^{\infty} 1/k^2) < \eta_i(1) \times 2 = \eta.$$

We introduce in line 9 a new scalar c_i used for deciding which portion of $e_i(k)$ the node will send in the network. In lines $11-12$ and $14-15$, the algorithm satisfies the equation (10). Then the new state value $x_i(k+1)$ is sent to the neighbors and $e_i(k+1)$ is updated accordingly. In Phase 2 of the algorithm (Algorithm 2), nodes initially in the wait state will decide either to send a cut-balance massage or to remain silent, the cut balance messages are sent when a node receives a poll message from any of its neighbors.

D. Convergence study

The convergence of the previous algorithm is mainly due to the fact that the proposed algorithm satisfies the conditions of convergence given in V-B2. In fact, the algorithm is designed to satisfy all these conditions that guarantee convergence. Starting with the state equation, we can notice from the Algorithm 1 given that whatever the condition the nodes face, it is always true that the sum of the new generated state values $\{x_i(k+1), e_i(k+1)\}$ is as follows:

$$x_i(k+1) + e_i(k+1) = y_i(k+1) + e_i(k),$$

where $y_i(k+1) = w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k)$. As a result the system equation is the one studied in section V-B2 (equation (10)). It can also be checked that according to the algorithm given in pseudo code, we have $\mathbf{e}(k+1) = \mathbf{e}(k) - F(k)\mathbf{x}(k)$ for some matrix $F(k)$ such that $F(k)\mathbf{1} = \mathbf{0}$ (see [16] for more details).

Now we can study the conditions mentioned in the Theorem 2 on the matrix $A(k) = W + F(k)$.

Lemma 1. *$A(k)$ is a stochastic matrix that satisfies conditions (a),(b),and (c) of Theorem 2.*

Proof: First, we can see that $A(k)\mathbf{1} = \mathbf{1}$ since $W\mathbf{1} = \mathbf{1}$ and $F(k)\mathbf{1} = \mathbf{0}$. It remains to prove that all entries in the matrix $A(k)$ are non negative, due to space limits the proof is in our technical report [16]. ■

Definition 1. *Two matrices, A and B, are said to be equivalent with respect to a vector $\mathbf{v}$ if and only if $A\mathbf{v} = B\mathbf{v}$.*

Notice that $A(k)$ satisfies conditions (a),(b), and (c) of Theorem 2, but possibly not the cut balance condition (d) because for a node $i \in T_k$ that transmits, $a_{ij}(k) > 0\ \forall\, j \in N_i$, but it can be that $\exists j \in N_i$ such that $a_{ji} = 0$ if j was silent at that iteration ($j \in S_k$). However, the next lemma shows that there is a matrix $B(k)$ equivalent to $A(k)$ with respect to $\mathbf{x}(k)$ that satisfies all the conditions.

Lemma 2. *For all k, there exists a matrix $B(k)$ equivalent to $A(k)$ with respect to $\mathbf{x}(k)$ such that $B(k)$ satisfies the conditions (a),(b),(c), and (d) of Theorem 2.*

Proof: Due to space limits, the proof is presented in the technical report [16]. ■

Lemma 3. *The message reduction algorithm (Phases 1, 2) satisfies condition (e) of Theorem 2.*

Proof: We will prove it by contradiction. Suppose $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star$, but $\lim_{k\to\infty} F(k)\mathbf{x}(k) \neq \mathbf{0}$, then there exists a node i such that $\lim_{k\to\infty} x_i(k) = x_i^\star$ and $\lim_{k\to\infty} y_i(k+1) = w_{ii}x_i^\star + \sum_{j\in N_i} w_{ij}x_j^\star = y_i^\star$, but $y_i^\star - x_i^\star = \delta^\star > 0$. From Algorithm 1, we can see that the node will enter a transmit state infinitely often (because d_i increases linearly with δ^* and it will reach the threshold η_i). Then, the node i will update its estimate according to the equation

$$x_i(k+1) = y_i(k+1) + \frac{\alpha}{1-w_{ii}}(y_i(k+1) - x_i(k)).$$

Letting $k \to \infty$ yields

$$(1 + \frac{\alpha}{1-w_{ii}})\delta^\star = 0.$$

Thus, $\delta^\star = 0$ which is a contradiction, and the algorithm satisfies condition (e) of Theorem 2. ■

The algorithm also provides that $|e_i(k)| \leq \eta_i(k)\ \forall k, i$ and $\eta_i(k) < \eta\ \forall i, k$, as in the first phase this condition is satisfied by construction, and for the second phase of the iteration, nodes from Phase 1 can check for worst case analysis and they only enter into *Wait* state if they are sure that the condition can be satisfied in the next phase iteration.

Now we are ready to state the main Theorem in this section:

Theorem 3. *The nodes applying the message reducing algorithm given in pseudocode by Algorithm 1 and 2, have estimates converging to a consensus within a margin η from x_{ave}, i.e. $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$ and $|x'_{ave} - x_{ave}| \leq \eta$.*

Proof: The theorem is due to the fact that the Lemmas given in this subsection show that the algorithm satisfies all the convergence conditions of Theorem 2. ■

As a result, the convergence of nodes' estimates of the distributed algorithm for message reduction is guaranteed. We study in the next section the performance of this algorithm on random networks, we also address the case of faulty unreliable links, and we show the stability of the algorithm in the presence of nodes changing their estimate possibly due to faulty estimates or due to a changing environment.

E. Simulations

The termination algorithm (the message reduction algorithm) is simulated on two types of random graphs, the Random Geometric Graphs (RGG) and the Erdos Renyi (ER) graphs. To measure the distance from the average, we considered the normalized error metric defined as,

$$normalized\ error(k) = \frac{||\mathbf{x}(k) - \bar{\mathbf{x}}||_2}{||\mathbf{x}(0) - \bar{\mathbf{x}}||_2},$$

where the vector $\bar{\mathbf{x}}$ is $\bar{\mathbf{x}} = x_{ave}\mathbf{1}$. For example when $\log(normalizederror) = -3$, that means the error became 0.1% of the initial one. Initially, each node has a uniformly random value between 0 and 10. On the RGG with 500 nodes and connectivity radius 0.093, Fig. 3 gives a comparison between standard average consensus algorithms and the termination algorithm proposed in this paper. The standard algorithm referred here is the one that uses Eq. (1) without applying any attempts to terminate the protocol. The figure shows the effect of varying the precision η in the termination algorithm on the number of messages (active nodes per iteration). In the study of the convergence of the algorithm, we showed that the algorithm converges to at most η from the true average. As the figure shows, with termination algorithm the error converges to a value x'_{ave} different from the real average x_{ave}, smaller η gives closer estimate to x_{ave} but more messages are sent. In fact, the termination algorithm passes through three phases: the *first* phase is the initial start where nodes usually have large differences in their estimates and they tend to send many messages while decreasing the error (same start as standard algorithm), the *second* phase is the most efficient phase where nodes saves messages while continuing to decrease the error.

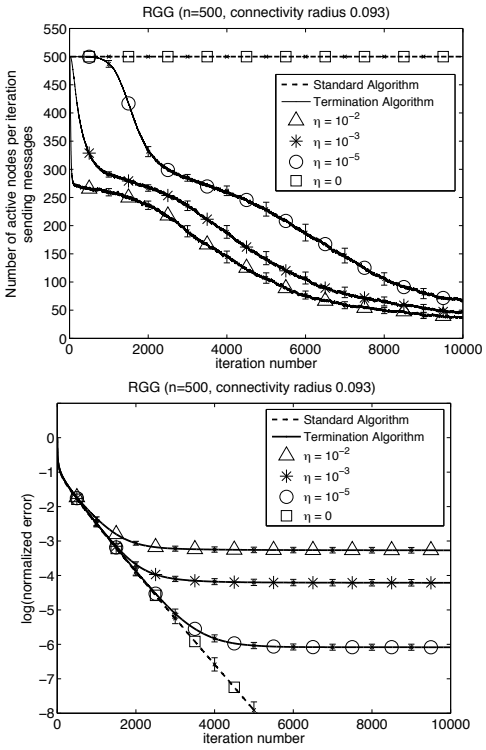

Fig. 3. (a) The number of active nodes at every iteration. (b) The error from the average at every iteration. This shows the effect of η in the message reduction algorithm on RGG networks. In standard algorithms, nodes' estimates converge to the real average, so the error decreases linearly, but nodes are not aware of how close they are to consensus, so they are all always active sending messages. With termination algorithm, nodes converge to a value at most η away from the real average, different values of η give different precision error. The algorithm gives a trade-off between precision and number of messages (The standard algorithm is just a special case of termination algorithm for $\eta = 0$).

The *final* phase is the stabilizing phase where nodes converge to a value close to the true average. The start and duration of each phase depends on the value of η. Similar results were given on ER graphs but are omitted due to lack of space.

Links in networks (specially wireless networks) can be unreliable. The algorithm being totally decentralized, and uses only one history estimate can be applied for the dynamic scenario. The weight matrix is then dynamic and at every iteration k a different $W(k)$ is considered and constructed locally as following. Before starting the algorithm we let $W(0)$ be generated locally satisfying convergence conditions as throughout the paper. At iteration k, a weight on a link can take two values, the original weight ($w_{ij}(k) = w_{ij}(0)$) if link $l \sim \{i, j\}$ is active or $w_{ij}(k) = 0$ if the link failed. When there are failures of links, some weight is added to the self-weight of nodes to preserve the double stochastic property of the matrix $W(k)$. In Fig. 4, we consider the RGG of 100 nodes and connectivity radius 0.19 with unreliable links. $\eta = 0.01$ is fixed for both graphs (the graph with the high link failure probability graph and the low link failure probability one). With high link failure probability, the network sends

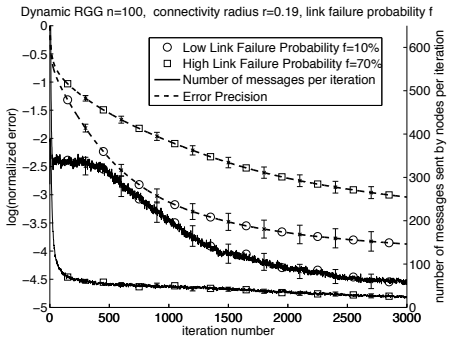

Fig. 4. Termination algorithm on a dynamic RGG with different link failure probabilities. On low link failure probability graphs, the messages are less than that of the high failure probability, but the convergence speed is slower.

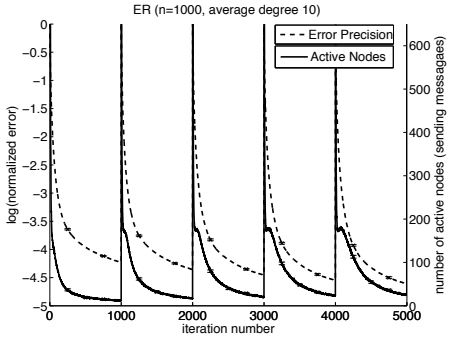

Fig. 5. Normalized error and number of nodes transmitting on the ER graphs with 1000 nodes, where every 1000 iterations random 10% of the nodes change their estimates. The figure shows the self-adaptive feature of the algorithm: when the algorithm has already converged to a stable state and is communicating little, and an exogenous event pushes the value in a node away from the current average, the algorithm detects the change and ramps up its communication intensity.

less messages because there are less links in the network, but the speed to consensus is slower than that of the low failure probability. Note that non of the synchronous termination algorithms given in the related work consider a dynamic topology.

In some scenarios, we are interested in a rapid detection of a sudden change in the true average due to the environment change. In the real life, if sensor nodes are measuring the temperature of a building, and the temperature changed largely (probably due to a fire in a certain room in the building), fast detection of this change can be very useful. In Fig. 5, we assumed that at certain iterations some nodes for a certain reason change their estimates to a completely different one. We considered an Erdos Renyi graph with 1000 nodes and average degree 10, as an initial state, nodes estimate takes a value in the interval $[0, 10]$ uniformly at random. Every 1000 iterations, on average, 10% of the nodes restart there estimate by a new one in the interval $[10, 20]$ chosen uniformly at random. The figure shows the self-adaptive feature of the algorithm: when the algorithm has already converged to a stable state and is communicating little, and an exogenous event pushes the value in a node away from the current average, the algorithm detects the change and ramps up its communication intensity and stabilizes again. So the algorithm is able to cope with the sudden change and the system automatically adapts its behavior. With every change in the estimates, the network give a burst of messages to stabilize the network to the new average.

VI. Conclusion

In this paper, we give an algorithm to reduce the messages sent in average consensus. The algorithm is totally decentralized and does not depend on any global variable, it only uses the weights selected to neighbors and one iteration history of the estimates to decide to send a message or not. We proved that this algorithm is converging to a consensus at most η from the true average. The algorithm can be applied on dynamic graphs and is also robust and adaptive to errors caused by a node suddenly changing its estimate.

References

[1] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[2] N. Hayashi and T. Ushio, "Application of a consensus problem to fair multi-resource allocation in real-time systems." in *CDC*. IEEE, 2008, pp. 2450–2455.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2508–2530, June 2006.

[4] F. Bénézit, A. G. Dimakis, P. Thiran, and M. Vetterli, "Order-optimal consensus through randomized path averaging," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5150 –5167, October 2010.

[5] R. Olfati-saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proc. of the IEEE*, Jan. 2007.

[6] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, June 2005, pp. 1859 – 1864 vol. 3.

[7] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.

[8] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM REVIEW*, vol. 46, pp. 667–689, April 2004.

[9] K. Avrachenkov, M. El Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks," *LOCALGOS international workshop held in conjunction with IEEE DCOSS'11*, June 2011.

[10] S. Sundaram and C. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *American Control Conference (ACC '07)*, July 2007, pp. 711 –716.

[11] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," *Forty-Fifth Annual Allerton Conference Allerton House, UIUC, Illinois, USA*, September 2007.

[12] A. Daher, M. Rabbat, and V. K. Lau, "Local silencing rules for randomized gossip," *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2011.

[13] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, pp. 33–46, January 2007.

[14] Y. Hatano, A. Das, and M. Mesbahi, "Agreement in presence of noise: pseudogradients on random geometric networks," in *in Proc. of the 44th IEEE CDC-ECC*, December 2005.

[15] J. Hendrickx and J. Tsitsiklis, "A new condition for convergence in continuous-time consensus seeking systems," in *IEE 50th CDC-ECC conference*, Dec. 2011, pp. 5070 –5075.

[16] M. El Chamie, G. Neglia, and K. Avrachenkov, "Reducing communication overhead for average consensus," INRIA, Technical Report, Jul. 2012. [Online]. Available: http://hal.inria.fr/hal-00720687