2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Controlled Hopwise Averaging: Bandwidth/Energy-Efficient Asynchronous Distributed Averaging for Wireless Networks

Choon Yik Tang and Jie Lu

Abstract— **This paper addresses the problem of averaging numbers across a wireless network from an important, but largely neglected, viewpoint: *bandwidth/energy efficiency*. We show that existing distributed averaging schemes are inefficient, producing networked dynamical systems that evolve with wasteful communications. To improve efficiency, we develop *Controlled Hopwise Averaging* (CHA), a distributed asynchronous algorithm that attempts to "make the most" out of each transmission. Unlike the existing schemes, CHA fully exploits the broadcast nature of wireless medium and enables greedy, decentralized, feedback control of when to initiate an iteration. We show that CHA admits a common quadratic Lyapunov function for analysis and control, establish its exponential convergence, and characterize its worst-case convergence rate. Finally, through extensive simulation on random geometric graphs, we show that CHA is substantially more efficient than several existing schemes, requiring far fewer transmissions to complete an averaging task.**

I. Introduction

Averaging numbers across a network is a need that arises in many applications of mobile ad hoc networks [1] and wireless sensor networks [2]. In order to collaboratively accomplish a task, nodes often have to compute the network-wide average of their individual observations. For examples, by averaging their individual throughputs, an ad hoc network of computers can assess how well the network, as a whole, is performing; by averaging their noisy observations, a team of unmanned aerial vehicles tracking a target can jointly determine the minimum mean-square-error estimate of the target position; and by averaging their humidity measurements, a wireless network of sensing agents can cooperatively detect the occurrence of local, deviation-from-average anomalies. Therefore, methods that enable such computation are of notable interest. Moreover, since such nodes often have to operate autonomously in dynamic and infrastructureless environments, communicate in a multi-hop fashion over unreliable wireless channels, and cope with severe bandwidth and energy constraints, it is highly desirable that the methods developed be robust, scalable, and efficient.

In principle, computation of the network-wide average may be accomplished via *flooding*, whereby every node floods the network with its observation, as well as *centralized computation*, whereby a central node uses an overlay tree to collect all the node observations, calculate their average, and send it back to every node. These two methods, unfortunately, have serious limitations: flooding is extremely bandwidth and energy inefficient because it propagates redundant information across the network, ignoring the fact that the ultimate goal is to simply determine the average. Centralized computation, on the other hand, is vulnerable to node mobility, node membership changes, and single-point failures, making it necessary to frequently maintain the overlay tree and occasionally start over with a new central node, both of which are rather costly to implement.

The authors are with the School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK 73019, USA (e-mail: cytang@ou.edu; jie.lu-1@ou.edu).

The limitations of flooding and centralized computation have motivated the search for *distributed averaging* algorithms of iterative nature, that require neither flooding of node observations, nor construction of overlay trees and routing tables, to execute. To date, numerous such algorithms have been developed in continuous-time [3]–[5] as well as in discrete-time for both synchronous [3], [5]–[12] and asynchronous [13]–[20] models, including *Pairwise Averaging* [13], *Anti-Entropy Aggregation* [14], [15], *Randomized Gossip Algorithm* [16], *Accelerated Gossip Algorithm* [17], *Distributed Random Grouping* [18], *Consensus Propagation* [19], and *Algorithms A1* and *A2* of [20] for the asynchronous case. The closely related topic of *distributed consensus*, where nodes seek to achieve an arbitrary network-wide consensus on their individual opinions, has also been extensively studied; see [21], [22] for early treatments, [3], [23]–[28] for more recent work, and [29] for a survey.

Although the current literature offers a rich collection of distributed averaging schemes along with in-depth analysis of their behaviors, their efficacy from a *bandwidth/energy efficiency* standpoint has not been examined. This paper is devoted to addressing the distributed averaging problem from this standpoint. The contributions of the paper are as follows:

1) We provide, in Section III, a detailed explanation of why the existing schemes, regardless of whether they are developed in continuous- or discrete-time, for synchronous or asynchronous models, are inefficient, producing networked dynamical systems that evolve with wasteful communications. We also point out a number of other drawbacks of these schemes.

2) To improve efficiency and overcome such drawbacks, we propose, in Sections IV and V, respectively, *Random* and *Controlled Hopwise Averaging* (RHA and CHA), two asynchronous distributed averaging algorithms with convergence guarantees: RHA is almost surely asymptotically convergent, while CHA is exponentially convergent with the worst-case convergence rate governed by the solution of a convex maximization problem.

3) To the best of our knowledge, CHA is the first asynchronous distributed averaging algorithm that fully exploits the broadcast nature of wireless medium, so that no potentially useful overheard information is discarded. It is also the first distributed averaging algorithm that enables *feedback iteration control*, leading to a networked dynamical system with state-dependent switching.

4) We show that both RHA and CHA admit a common quadratic Lyapunov function for analysis. This result casts a positive light on a nonexistence result for a class of consensus algorithms, numerically verified in [23] and analytically proven in [30]. In addition, we show that the same Lyapunov function may be used by CHA to perform

978-1-4244-4524-0/09/$25.00 ©2009 AACC

greedy, decentralized, feedback iteration control. Although Lyapunov functions have been used to analyze consensus algorithms (e.g., in the form of a disagreement function [3] or a set-valued convex hull [25]), their use for control purposes has not been reported.

5) We demonstrate, via extensive simulation on random geometric graphs in Section VI, that CHA is substantially more efficient than Pairwise Averaging [13], Anti-Entropy Aggregation [14], [15], Consensus Propagation [19], Algorithm A2 of [20], and Distributed Random Grouping [18], requiring far fewer transmissions to complete an averaging task. In particular, CHA is twice more efficient than the most efficient existing scheme—Distributed Random Grouping [18]—when the network is sparsely connected.

The outline of this paper is as follows: Section II formulates the distributed averaging problem. Section III identifies and explains the deficiencies of the existing schemes. Sections IV and V develop RHA and CHA and characterize their convergence properties. In Section VI, their comparison with several existing schemes is carried out. Finally, Section VII concludes the paper. Due to space limitations, all proofs are omitted and can be found in [31].

II. Problem Formulation

Consider a multi-hop wireless network consisting of $N \geq 2$ nodes, connected by L bidirectional links in a fixed topology. The network is modeled as a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of N nodes (vertices) and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ represents the set of L links (edges). Any two nodes $i, j \in \mathcal{V}$ are one-hop neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$, and the set of one-hop neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$. Each node $i \in \mathcal{V}$ observes a scalar $y_i \in \mathbb{R}$, and all the N nodes wish to determine the network-wide average $x \in \mathbb{R}$ of their individual observations, given by

$$x = \frac{1}{N} \sum_{i \in \mathcal{V}} y_i. \tag{1}$$

Given the above model, the problem addressed in this paper is how to construct a distributed averaging algorithm—continuous- or discrete-time, synchronous or otherwise—with which each node $i \in \mathcal{V}$ repeatedly communicates with its one-hop neighbors, iteratively updates its estimate $\hat{x}_i \in \mathbb{R}$ of the unknown average x in (1), and asymptotically drives $\hat{x}_i$ to x—all while consuming bandwidth and energy efficiently.

The bandwidth/energy efficiency of an algorithm is measured by *the number of real-number transmissions it needs to drive all the $\hat{x}_i$'s to a sufficiently small neighborhood of x*, essentially completing the averaging task. This quantity is a natural measure of efficiency because the smaller it is, the lesser bandwidth is occupied, the lesser energy is expended for communications, and the faster an averaging task may be completed. These, in turn, imply more bandwidth and time for other tasks, smaller probability of collision, longer lifetime for battery-powered nodes, and possible earlier return to sleep mode, all of which are desirable. The quantity also allows algorithms with different numbers of real-number transmissions per iteration to be fairly compared. Although, in networking, every message inevitably contains overhead (e.g., transmitter/receiver IDs and message type), we exclude such overhead when measuring efficiency since it is not inherent to an algorithm, may be reduced by piggybacking messages, and becomes negligible when averaging long vectors.

Similar to most existing work on distributed averaging [3]–[20] and distributed consensus [3], [21]–[29], we assume ideal internode communications, so that every message from each node $i \in \mathcal{V}$ is not subject to quantization, takes negligible time to transmit and propagate, and is received with negligible error. Moreover, since the nodes are wirelessly connected, the same message is received by every one-hop neighbor $j \in \mathcal{N}_i$, regardless of the intended recipient(s).

III. Deficiencies of Existing Schemes

As was pointed out in Section I, the current literature offers a variety of distributed averaging schemes for solving the problem formulated in Section II. Unfortunately, as is explained below, they suffer from a number of deficiencies, especially a lack of bandwidth/energy efficiency, by producing networked dynamical systems that evolve with wasteful real-number transmissions.

The continuous-time algorithms in [3]–[5] have the following deficiency:

D1) *Costly discretization*: As immensely inefficient as flooding is, the continuous-time algorithms in [3]–[5] may be more so: flooding only requires N^2 real-number transmissions for all the N nodes to exactly determine the average x (since it takes N transmissions for each node $i \in \mathcal{V}$ to flood the network with its y_i), whereas these algorithms may need far more than that to essentially complete an averaging task. For instance, the algorithm in [3] updates the estimates $\hat{x}_i$'s of x according to the differential equation

$$\frac{d\hat{x}_i(t)}{dt} = \sum_{j \in \mathcal{N}_i} (\hat{x}_j(t) - \hat{x}_i(t)), \quad \forall i \in \mathcal{V}. \tag{2}$$

To realize (2), each node $i \in \mathcal{V}$ has to continuously monitor the $\hat{x}_j(t)$ of every one-hop neighbor $j \in \mathcal{N}_i$. If this can be done without wireless communications (e.g., by direct sensing), then the bandwidth/energy efficiency issue is moot. If wireless communications must be employed, then (2) has to be discretized, either exactly via a zero-order hold, i.e.,

$$\hat{x}_i((k+1)T) = \sum_{j \in \mathcal{V}} h_{ij} \hat{x}_j(kT), \quad \forall i \in \mathcal{V}, \tag{3}$$

or approximately via numerical techniques such as the Euler forward difference method, i.e.,

$$\frac{\hat{x}_i((k+1)T) - \hat{x}_i(kT)}{T} = \sum_{j \in \mathcal{N}_i} (\hat{x}_j(kT) - \hat{x}_i(kT)), \ \forall i \in \mathcal{V}, \tag{4}$$

where each $h_{ij} \in \mathbb{R}$ is the ij-entry of $e^{-\mathcal{L}T}$, $\mathcal{L} \in \mathbb{R}^{N \times N}$ is the Laplacian matrix of the graph $\mathcal{G}$ that governs the dynamics (2), and $T > 0$ is the sampling period. Regardless of (3) or (4), they may be far more costly to realize than flooding: with (3), N^2 real-number transmissions are already needed per iteration (since, in general, $h_{ij} \neq 0$, so that each node $i \in \mathcal{V}$ has to flood the network with its $\hat{x}_i(kT)$, for every k). In contrast, with (4), only N real-number transmissions are needed per iteration (since each node $i \in \mathcal{V}$ only has to wirelessly transmit its $\hat{x}_i(kT)$ once, to every one-hop neighbor $j \in \mathcal{N}_i$, for every k). However, the number of iterations, needed for all the $\hat{x}_i(kT)$'s to converge to an acceptable neighborhood of x, may be very large, since the sampling period T must be sufficiently small for (4) to be

stable. If the number of iterations needed exceeds N—which is entirely possible and likely so with a conservatively small T—then (4) would be more inefficient than flooding.

The discrete-time synchronous algorithms in [3], [5]–[12] have the following deficiencies:

D2) *Clock synchronization*: The discrete-time synchronous algorithms in [3], [5]–[12] require all the N nodes to always have the same clock to operate. Although techniques for reducing clock synchronization errors are available, it is still desirable that this requirement can be completely eliminated.

D3) *Forced transmissions*: The discrete-time synchronous algorithms in [3], [5], [7]–[11] update the estimates $\hat{x}_i$'s of x according to the difference equation

$$\hat{x}_i(k+1) = w_{ii}\hat{x}_i(k) + \sum_{j\in\mathcal{N}_i} w_{ij}\hat{x}_j(k), \quad \forall i \in \mathcal{V}, \qquad (5)$$

where each $w_{ij} \in \mathbb{R}$ is a weighting factor. The w_{ij}'s may be specified in a number of ways [3], [5], [7]–[11], including choosing them to optimize the convergence rate [7] or minimize the mean-square deviation [11]. However, no matter how the w_{ij}'s are chosen, these algorithms are bandwidth/energy inefficient because the underlying update rule (5) simply *forces* every node $i \in \mathcal{V}$ at each iteration k to transmit its $\hat{x}_i(k)$ to its one-hop neighbors, irrespective of whether the transmission is worthy. It is possible, for example, that the $\hat{x}_i(k)$'s of a cluster of nearby nodes are almost equal, so that the $\hat{x}_i(k+1)$'s, being convex combinations of the $\hat{x}_i(k)$'s, are also almost equal, causing their transmissions to be unworthy. The fact that N real-number transmissions are needed per iteration also implies that, in order to be more efficient than flooding, the algorithms must drive all the $\hat{x}_i(k)$'s to an acceptable neighborhood of x within at most N iterations.

D4) *Computing intermediate quantities*: The scheme in [10] uses two parallel runs of a consensus algorithm to obtain two consensus values and defines each $\hat{x}_i(k)$ as the ratio of these two values. While possible, this scheme is likely inefficient because it attempts to compute two *intermediate* quantities, as opposed to computing x directly.

The discrete-time asynchronous algorithms in [13]–[20] have the following deficiencies:

D5) *Wasted receptions*: The algorithms in [13]–[20] do not exploit the broadcast nature of wireless medium. With Pairwise Averaging [13], Anti-Entropy Aggregation [14], [15], Randomized Gossip Algorithm [16], and Accelerated Gossip Algorithm [17], each iteration involves a pair of nodes exchanging their state variables. Unintended nearby nodes that overhear a transmission would simply discard the message, resulting in wasted receptions. The same can be said about Consensus Propagation [19] and Algorithm A2 of [20], although they do not assume pairwise exchanges. A partial exception is Distributed Random Grouping [18], which only slightly exploits such broadcast nature (i.e., the leader of a group does but the members, which contribute to the majority of the transmissions, do not). It is of interest to investigate how such "free" information may be exploited and to what extent may such exploitation speed up convergence.

D6) *Overlapping iterations*: Pairwise Averaging [13], Anti-Entropy Aggregation [14], [15], Randomized Gossip Algorithm [16], Accelerated Gossip Algorithm [17], and Distributed Random Grouping [18] require sequential transmissions from more than one nodes to execute an iteration. Thus, before an iteration completes, nodes participating in that iteration may be asked by other nodes unaware of the ongoing iteration to begin other iterations, thereby creating potentially complex situations of overlapping iterations. It is noted that this practical issue is naturally avoided in Consensus Propagation [19] and explicitly handled in Algorithms A1 and A2 of [20].

D7) *Uncontrolled iterations*: The algorithms in [13]–[20] only specify how nodes would update their state variables *whenever* they communicate during an iteration. They do not specify *when* and/or *how often* they should communicate, i.e., the iterations are not controlled. Two exceptions are Randomized Gossip Algorithm [16] and Distributed Random Grouping [18], which specify the probabilities with which nodes communicate with their neighbors or become leaders. Although these probabilities may be optimized [16], [18], the optimization is carried out *a priori*, independent of the nodes' state variables during run-time. As a result, it is possible that the state variables of nodes involving in an iteration take similar values before and after the iteration, thereby wasting that particular iteration.

D8) *Steady-state errors*: Consensus Propagation [19] ensures that all the $\hat{x}_i(k)$'s asymptotically converge to the same value. However, this value is, in general, not equal to x (see Figure 2), although theoretically the difference can be made arbitrarily small, at the expense of slower convergence.

D9) *Lack of convergence guarantee*: Accelerated Gossip Algorithm [17], developed based on the power method in numerical analysis, is shown by simulation to have the potential of speeding up the convergence of Randomized Gossip Algorithm [16] by a factor of 10. However, it has yet been proven that the algorithm always converges.

IV. Random Hopwise Averaging

The deficiencies D1 of continuous-time, D2–D4 of discrete-time synchronous, and D5–D9 of discrete-time asynchronous distributed averaging algorithms raise the question of whether it is possible to develop an algorithm that does not suffer at all from these deficiencies. In this section, we present a step-by-step development of an algorithm that avoids all but deficiency D7. In the next section, we modify it to eliminate D7.

To circumvent D1, the costly discretization issue with continuous-time algorithms, as well as D2 and D3, the clock synchronization and forced transmissions issues with discrete-time synchronous algorithms, the algorithm we develop must be asynchronous. To avoid D6, the issue with overlapping iterations, each iteration of the algorithm must involve only a single node, say, node i, transmitting exactly once to its one-hop neighbors $j \in \mathcal{N}_i$, i.e., no replies from the neighbors are allowed. To address D5, the issue with wasted receptions, all the neighbors $j \in \mathcal{N}_i$, upon receiving the message from node i, have to "meaningfully" incorporate the message into updating their estimates, rather than discarding it. To circumvent D8 and D9, the issues with steady-state errors and convergence guarantee, the algorithm must be provably asymptotically convergent to the correct average. Finally, to eliminate D4, it must avoid computing intermediate quantities.

To develop an algorithm that has the aforementioned properties, consider a networked dynamical system, where the state variables are associated with the set $\mathcal{E}$ of L links of the graph $\mathcal{G}$, rather than with the set $\mathcal{V}$ of N nodes,

as is typically assumed, for example, in [13]–[18]. Let $x_{\{i,j\}} \in \mathbb{R}$ represent the state variable associated with each link $\{i,j\} \in \mathcal{E}$, $x_{\{i,j\}}(0)$ be its initial value, and $x_{\{i,j\}}(k)$ be its value upon completion of each iteration $k \in \mathbb{P}$, where $\mathbb{P}$ denotes the set of positive integers. Let $c_{\{i,j\}} > 0$ represent a to-be-determined constant associated with each link $\{i,j\} \in \mathcal{E}$. Suppose, upon completing each iteration $k \in \mathbb{P}$, the state variables $x_{\{i,j\}}(k)$'s are such that the expression $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(k)$ is conserved, i.e.,

$$\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(k) = \sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(k-1),\ \forall k \in \mathbb{P}. \quad (6)$$

Also suppose the differences among the $x_{\{i,j\}}(k)$'s decrease to zero as $k \to \infty$, so that they all converge to the same steady-state value $\tilde{x} \in \mathbb{R}$, i.e.,

$$\lim_{k\to\infty} x_{\{i,j\}}(k) = \tilde{x}, \quad \forall \{i,j\} \in \mathcal{E}. \quad (7)$$

Then, due to (6) and (7), $\tilde{x}$ is given by

$$\tilde{x} = \frac{\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(0)}{\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}}}. \quad (8)$$

To solve the averaging problem, the steady-state value $\tilde{x}$ in (8) needs to be equal to the average x in (1), i.e.,

$$\frac{\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(0)}{\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}}} = \frac{\sum_{i\in\mathcal{V}} y_i}{N}. \quad (9)$$

Clearly, (9) may be satisfied by letting the constants $c_{\{i,j\}}$'s and initial values $x_{\{i,j\}}(0)$'s be

$$c_{\{i,j\}} = \frac{1}{|\mathcal{N}_i|} + \frac{1}{|\mathcal{N}_j|}, \quad \forall \{i,j\} \in \mathcal{E}, \quad (10)$$

$$x_{\{i,j\}}(0) = \frac{\frac{y_i}{|\mathcal{N}_i|} + \frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall \{i,j\} \in \mathcal{E}, \quad (11)$$

where $|\cdot|$ denotes the cardinality of a set. Although this choice is not unique, defining the $c_{\{i,j\}}$'s and $x_{\{i,j\}}(0)$'s as in (10) and (11) offers the advantage that each node $i \in \mathcal{V}$ can compute $c_{\{i,j\}}$ and $x_{\{i,j\}}(0)$ $\forall j \in \mathcal{N}_i$ as long as, during algorithm initialization, it transmits $|\mathcal{N}_i|$ and y_i once, to every node $j \in \mathcal{N}_i$, resulting in an initialization overhead of $2N$ real-number transmissions.

The above paragraph shows that the averaging problem can be solved by letting the $c_{\{i,j\}}$'s and $x_{\{i,j\}}(0)$'s be as in (10) and (11) and ensuring that (6) and (7) are satisfied. To satisfy (6) and (7) while circumventing the deficiencies, for each link $\{i,j\} \in \mathcal{E}$, let nodes i and j each maintain a local copy of the state variable $x_{\{i,j\}}(k)$, denoted as $x_{ij}(k)$ and $x_{ji}(k)$, respectively, where they are meant to be always equal, so that the ordering of the subscripts is only used to distinguish between the local copies. Since each node $i \in \mathcal{V}$ has $|\mathcal{N}_i|$ one-hop neighbors, it maintains $|\mathcal{N}_i|$ local copies of the state variables $x_{\{i,j\}}(k)$'s $\forall j \in \mathcal{N}_i$. In addition to these local copies, let each node $i \in \mathcal{V}$ also maintain an estimate $\hat{x}_i(k) \in \mathbb{R}$ of the unknown average x, defined as

$$\hat{x}_i(k) = \frac{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}} x_{\{i,j\}}(k)}{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}}, \quad \forall k \geq 0, \forall i \in \mathcal{V}, \quad (12)$$

which is just a convex combination of the $x_{\{i,j\}}(k)$'s $\forall j \in \mathcal{N}_i$. To circumvent D1–D3, consider an asynchronous iteration, say, iteration $k \in \mathbb{P}$, initiated by a randomly and equiprobably selected node, say, node i, and involving all its one-hop neighbors $j \in \mathcal{N}_i$ in the following manner. To avoid D6, node i sets all its local copies $x_{ij}(k)$'s $\forall j \in \mathcal{N}_i$ to $\hat{x}_i(k-1)$, sets its estimate $\hat{x}_i(k)$ also to $\hat{x}_i(k-1)$ (or, equivalently, calculates $\hat{x}_i(k)$ from (12)), and transmits the value of $\hat{x}_i(k)$ once, to every one-hop neighbor $j \in \mathcal{N}_i$, but does not expect any replies from its neighbors. To address D5 and ensure that the local copies are always equal, each neighbor $j \in \mathcal{N}_i$, upon receiving the value of $\hat{x}_i(k)$, sets its local copy $x_{ji}(k)$ to $\hat{x}_i(k)$ and calculates its estimate $\hat{x}_j(k)$ from (12). The above process defines an iteration k.

Since each iteration is initiated by a *randomly* selected node, and since all the node does is updating the state variables within a *hop* as a convex combination of their previous values, we refer to the resulting algorithm as *Random Hopwise Averaging* (RHA). RHA yields an L-dimensional networked dynamical system with random switching, given by

$$x_{\{i,j\}}(k) = \begin{cases} \dfrac{\sum_{\ell\in\mathcal{N}_{u(k)}} c_{\{u(k),\ell\}} x_{\{u(k),\ell\}}(k-1)}{\sum_{\ell\in\mathcal{N}_{u(k)}} c_{\{u(k),\ell\}}}, & \text{if } u(k)\in\{i,j\}, \\ x_{\{i,j\}}(k-1), & \text{otherwise,} \end{cases}$$
$$\forall k \in \mathbb{P},\ \forall \{i,j\} \in \mathcal{E}, \quad (13)$$

where $u(k) \in \mathcal{V}$ represents the node that initiates iteration $k \in \mathbb{P}$, i.e., causes a switch in the dynamics. The system (13) with initial condition (11) and output equation (12) can be formalized in algorithm form as follows:

Algorithm 1 (Random Hopwise Averaging)**.**
Initialization:
1) Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i to every node $j \in \mathcal{N}_i$.
2) Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}$ $\forall j \in \mathcal{N}_i$ and $\hat{x}_i \in \mathbb{R}$ and initializes them sequentially:
$x_{ij} \leftarrow \frac{\frac{y_i}{|\mathcal{N}_i|} + \frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall j \in \mathcal{N}_i, \quad \hat{x}_i \leftarrow \frac{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}} x_{ij}}{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}}$.

Operation: At each iteration:
3) A node, say, node i, is selected randomly and equiprobably out of the set $\mathcal{V}$ of N nodes.
4) Node i updates x_{ij} $\forall j \in \mathcal{N}_i$: $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i$.
5) Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6) Each node $j \in \mathcal{N}_i$ updates x_{ji} and $\hat{x}_j$ sequentially:
$x_{ji} \leftarrow \hat{x}_i, \quad \hat{x}_j \leftarrow \frac{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}}{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}}$. ■

Note that (13) ensures (6). To show that it ensures (7), let $\mathbf{x}(k) \in \mathbb{R}^L$ represent the vector obtained by stacking the state variables $x_{\{i,j\}}(k)$'s and consider the following quadratic Lyapunov function candidate $V : \mathbb{R}^L \to \mathbb{R}$:

$$V(\mathbf{x}(k)) = \sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} (x_{\{i,j\}}(k) - x)^2. \quad (14)$$

The following lemma shows that $V(\mathbf{x}(k))$ is non-increasing.

Lemma 1. *Consider the wireless network modeled in Section II and the use of RHA described in Algorithm 1. Then, for any $y_i \in \mathbb{R}$ $\forall i \in \mathcal{V}$, and any sequence $\{u(k)\}_{k\in\mathbb{P}}$, $V(\mathbf{x}(k))$ is non-increasing $\forall k \in \mathbb{P}$ and satisfies*

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\sum_{j\in\mathcal{N}_{u(k)}} c_{\{u(k),j\}} (x_{\{u(k),j\}}(k-1) - \hat{x}_{u(k)}(k-1))^2. \quad (15)$$

The following theorem establishes the almost sure asymptotic convergence of RHA, so that D8 and D9 are avoided:

Theorem 1. *Consider the wireless network modeled in Section II and the use of RHA described in Algorithm 1. Then, for any $y_i \in \mathbb{R}$ $\forall i \in \mathcal{V}$, with probability* 1*,*

$$\lim_{k\to\infty} V(\mathbf{x}(k)) = 0, \tag{16}$$

$$\lim_{k\to\infty} x_{\{i,j\}}(k) = x, \quad \forall \{i,j\} \in \mathcal{E}, \tag{17}$$

$$\lim_{k\to\infty} \hat{x}_i(k) = x, \quad \forall i \in \mathcal{V}. \tag{18}$$

V. Controlled Hopwise Averaging

In Section IV, we develop RHA and show that it successfully overcomes all the deficiencies of the existing schemes except D7. RHA fails to account for D7 because it randomly selects a node to initiate an iteration, i.e., the iterations are uncontrolled. In this section, we show that the iterations can be feedback controlled with a suitable modification of RHA, beginning with the assumption that a "genie" exists for control purposes, followed by a relaxation of this assumption.

Observe from Lemma 1 that if node $u(k)$ initiates iteration k, the value of the Lyapunov function V would drop by an amount equal to the right-hand side of (15). Also observe that the state variables appearing on the right-hand side of (15), i.e., characterizing the amount of drop, are locally maintained by node $u(k)$. Therefore, node $u(k)$ knows that if it spontaneously decides to initiate iteration k, the value of V, *whatever it may be*, would drop by an amount which it knows. This implies that every node $i \in \mathcal{V}$ at any given time knows by how much the value of V would drop if it elects to become the node that initiates "the next" iteration. Hence, if there is a genie in the network that knows all the potential drops, the genie may choose to behave greedily and always let the node that causes the largest drop in the value of V initiate the next iteration.

To formalize this idea, consider the function $\Delta V_i : \mathbb{R}^L \to \mathbb{R}$, $i \in \mathcal{V}$, defined as

$$\Delta V_i(\mathbf{x}(k)) = \sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{\{i,j\}}(k) - \hat{x}_i(k))^2, \tag{19}$$

where $\hat{x}_i(k)$ is given in (12). Notice that each node $i \in \mathcal{V}$ has sufficient information to determine $\Delta V_i(\mathbf{x}(k))$ itself. With ΔV_i defined as in (19), equation (15) can be rewritten as

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = \begin{cases} -\Delta V_1(\mathbf{x}(k-1)), & \text{if } u(k) = 1, \\ -\Delta V_2(\mathbf{x}(k-1)), & \text{if } u(k) = 2, \\ \quad\vdots & \\ -\Delta V_N(\mathbf{x}(k-1)), & \text{if } u(k) = N. \end{cases} \tag{20}$$

Thus, the genie, equipped with knowledge of all the $\Delta V_i(\mathbf{x}(k-1))$'s, may select $u(k)$, the node that initiates iteration k, according to

$$u(k) = \arg\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k-1)), \quad k \in \mathbb{P}, \tag{21}$$

so that the value of V always exhibits maximum decrease upon completing each iteration.

Due to the idealized assumption of having a genie to carry out feedback control, we refer to the resulting distributed averaging algorithm as *Ideal Controlled Hopwise Averaging* (ICHA). Note that while RHA leads to a random switched system (13), ICHA yields a state-dependent switched system governed by (13) and (21). This can be formalized in algorithm form as follows:

Algorithm 2 (Ideal Controlled Hopwise Averaging)**.**
Initialization:

1) Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i to every node $j \in \mathcal{N}_i$.
2) Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}$ $\forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathbb{R}$, and $\Delta V_i \in [0,\infty)$ and initializes them sequentially:
$x_{ij} \leftarrow \frac{\frac{y_i}{|\mathcal{N}_i|} + \frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}$, $\forall j \in \mathcal{N}_i$, $\hat{x}_i \leftarrow \frac{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}} x_{ij}}{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}}$,
$\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{ij} - \hat{x}_i)^2$.

Operation: At each iteration:

3) Let $i \in \arg\max_{j\in\mathcal{V}} \Delta V_j$.
4) Node i updates x_{ij} $\forall j \in \mathcal{N}_i$ and ΔV_i sequentially:
$x_{ij} \leftarrow \hat{x}_i$, $\forall j \in \mathcal{N}_i$, $\Delta V_i \leftarrow 0$.
5) Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6) Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, and ΔV_j sequentially:
$x_{ji} \leftarrow \hat{x}_i$, $\hat{x}_j \leftarrow \frac{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}}{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}}$,
$\Delta V_j \leftarrow \sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}(x_{j\ell} - \hat{x}_j)^2$. ■

The following theorem asserts the asymptotic convergence of ICHA:

Theorem 2. *Consider the wireless network modeled in Section II and the use of ICHA described in Algorithm 2. Then, for any $y_i \in \mathbb{R}$ $\forall i \in \mathcal{V}$,* (16)–(18) *hold.*

The following theorem asserts a stronger result—ICHA is exponentially convergent with the worst-case convergence rate governed by the solution of a convex maximization (i.e., non-convex optimization) problem:

Theorem 3. *Consider the wireless network modeled in Section II and the use of ICHA described in Algorithm 2. Then,*

$$V(\mathbf{x}(k)) \le \rho V(\mathbf{x}(k-1)), \quad \forall k \in \mathbb{P}, \tag{22}$$

and $\exists \mathbf{x}(k-1) \in \mathbb{R}^L$ such that

$$V(\mathbf{x}(k)) = \rho V(\mathbf{x}(k-1)),$$

where $\rho \in [0,1)$ is such that $\frac{1}{1-\rho}$ is the optimal value of the following convex maximization problem:

$$\begin{array}{ll} \text{maximize}_{\mathbf{z}\in\mathbb{R}^L} & V(\mathbf{z}) \\ \text{subject to} & \Delta V_i(\mathbf{z}) \le 1, \quad \forall i \in \mathcal{V} \\ & \sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} z_{\{i,j\}} = 0. \end{array} \tag{23}$$

Obviously, ICHA is not implementable since it assumes the presence of a genie. Fortunately, it is possible to closely mimic the greedy behavior of ICHA with a practical, decentralized controller. To describe this controller, the notions of time $t \ge 0$ and a discrete-event system are needed. Suppose each node $i \in \mathcal{V}$ maintains a time-to-initiate variable $\tau_i > 0$, so that when time $t = \tau_i$, node i initiates the next iteration. Suppose also that τ_i is a function of ΔV_i:

$$\tau_i(k-1) = \Phi(\Delta V_i(\mathbf{x}(k-1))), \tag{24}$$

where $\Phi : [0,\infty) \to (0,\infty]$ is a continuous, strictly decreasing function satisfying $\lim_{v\to 0} \Phi(v) = \infty$ and $\Phi(0) = \infty$.

Since Φ is strictly decreasing, $\Delta V_i(\mathbf{x}(k-1))$ is inversely proportional to $\tau_i(k-1)$. Hence, with (24), the node with the largest $\Delta V_i(\mathbf{x}(k-1))$'s would also have the smallest τ_i's and, thus, would become the node that initiates iteration k. Although (24) attempts to foster a greedy behavior, it has two issues. First, it is possible that τ_i may become smaller than t upon completion of an iteration. This is undesirable because every node's time to initiate the next iteration should be in the future, not the past. Second, although unlikely it is theoretically possible that $\tau_i = \tau_j$ for some $i, j \in \mathcal{V}$ and, hence, if nodes i and j are one-hop neighbors, a collision of wireless transmissions would occur. To alleviate these two issues, we slightly modify (24), resulting in

$$\tau_i(k-1) = \max\{\Phi(\Delta V_i(\mathbf{x}(k-1))), t\} + \varepsilon(\Delta V_i(\mathbf{x}(k-1))) \cdot \mathrm{rand}(), \qquad (25)$$

where $\varepsilon : [0,\infty) \to (0,\infty)$ is a continuous function meant to take on a small positive value and $\mathrm{rand}()$ returns a uniformly distributed random variable on the unit interval. Note from (25) that introducing the $\max\{\cdot,\cdot\}$ function ensures that τ_i is never less than t. In addition, inserting a little randomness into (25) reduces the probability of collision.

With each node $i \in \mathcal{V}$ utilizing a decentralized, time-to-initiate-the-next-iteration controller (25), the resulting system becomes a simple discrete-event system, in which there are always N events scheduled, one from each node. The node $u(k)$ that initiates iteration k can therefore be determined from

$$u(k) = \arg\min_{i \in \mathcal{V}} \tau_i(k-1), \quad k \in \mathbb{P}. \qquad (26)$$

Expression (26), along with (25), (13), and (19), defines the practical *Controlled Hopwise Averaging* (CHA), which can be formalized in algorithm form as follows:

Algorithm 3 (Controlled Hopwise Averaging)**.**
Initialization:

1) Let time $t = 0$.
2) Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i to every node $j \in \mathcal{N}_i$.
3) Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}\ \forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathbb{R}$, $\Delta V_i \in [0,\infty)$, and $\tau_i \in (0,\infty]$ and initializes them sequentially:
$x_{ij} \leftarrow \frac{\frac{y_i}{|\mathcal{N}_i|} + \frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall j \in \mathcal{N}_i, \quad \hat{x}_i \leftarrow \frac{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}} x_{ij}}{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}}$,
$\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{ij} - \hat{x}_i)^2$,
$\tau_i \leftarrow \max\{\Phi(\Delta V_i), t\} + \varepsilon(\Delta V_i) \cdot \mathrm{rand}()$.

Operation: At each iteration:

4) Let $i \in \arg\min_{j\in\mathcal{V}} \tau_j$ and $t = \tau_i$.
5) Node i updates $x_{ij}\ \forall j \in \mathcal{N}_i$, ΔV_i, and τ_i sequentially:
$x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i, \quad \Delta V_i \leftarrow 0, \quad \tau_i \leftarrow \infty$.
6) Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
7) Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, ΔV_j, and τ_j sequentially:
$x_{ji} \leftarrow \hat{x}_i, \quad \hat{x}_j \leftarrow \frac{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}}{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}}$,
$\Delta V_j \leftarrow \sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}(x_{j\ell} - \hat{x}_j)^2$,
$\tau_j \leftarrow \max\{\Phi(\Delta V_j), t\} + \varepsilon(\Delta V_j) \cdot \mathrm{rand}()$. ■

VI. Performance Comparison

In this section, we compare the performances of RHA and CHA with those of *Pairwise Averaging* (PA) [13] (to which *Anti-Entropy Aggregation* [14], [15] is identical), *Consensus Propagation* (CP) [19], *Algorithm A2* (A2) of [20], *Distributed Random Grouping* (DRG) [18], and *flooding* via simulation. The algorithm parameters are selected as follows: for CP, $\beta = 10^6$; for A2, $\gamma_i = \frac{0.3}{|\mathcal{N}_i|+1}\ \forall i \in \mathcal{V}$ and $\phi_{ij} = \phi_{ji} = 0.49\ \forall \{i,j\} \in \mathcal{E}$; and for CHA, $\Phi(v) = 1/v$ and $\varepsilon = 0.001$. For PA and DRG, we assume that overlapping iterations cannot occur.

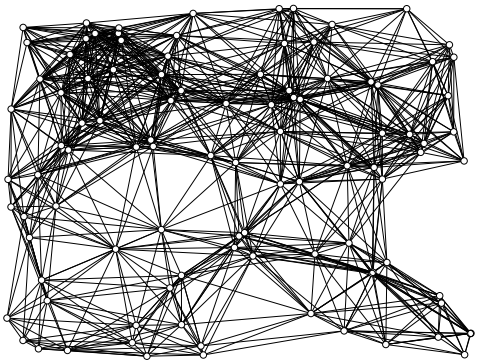

Fig. 1. A 100-node, 20-neighbor multi-hop wireless network.

Two sets of simulation results are generated. The first set corresponds to a single scenario of a multi-hop wireless network with $N = 100$ nodes, where each node, on the average, has $\frac{2L}{N} = 20$ neighbors, as shown in Figure 1. The second set corresponds to multi-hop wireless networks modeled by random geometric graphs, with number of nodes varying from $N = 100$ to $N = 500$, and average number of neighbors varying from $\frac{2L}{N} = 10$ to $\frac{2L}{N} = 60$. For each N and each $\frac{2L}{N}$, a total of 50 randomly generated network topologies are considered, so that every resulting data point is actually an average over these 50 topologies.

Results from the first set of simulation are shown in Figure 2. Observe from the figure that PA and A2 have roughly similar performance, requiring approximately $7,000$ real-number transmissions to drive all the node estimates $\hat{x}_i$'s to ± 0.005 of the average x. In contrast, CP fails to converge after $10,000$ transmissions, despite experimenting with several choices of its parameter β [19]. On the other hand, DRG is found to be quite efficient, needing only approximately $2,100$ transmissions to complete the averaging task. The proposed RHA outperforms PA, CP, and A2, but not DRG, while CHA is the most efficient among all the schemes, requiring only roughly $1,300$ transmissions to converge.

Results from the second set of simulation are shown in Figure 3. Analyzing the figure, we see that regardless of the number of nodes N and the average number of neighbors $\frac{2L}{N}$, CP has the worst efficiency, followed by PA and A2. DRG, RHA, and CHA are all fairly efficient, with CHA again having the best efficiency. In particular, CHA is at least 20% more efficient than DRG, and around 50% more so when the network is sparsely connected at $\frac{2L}{N} = 10$. The significant difference in performance between RHA and CHA represents the benefit of greedy, decentralized, feedback iteration control.

VII. Conclusion

In this paper, we have studied the distributed averaging problem from the bandwidth/energy efficiency standpoint

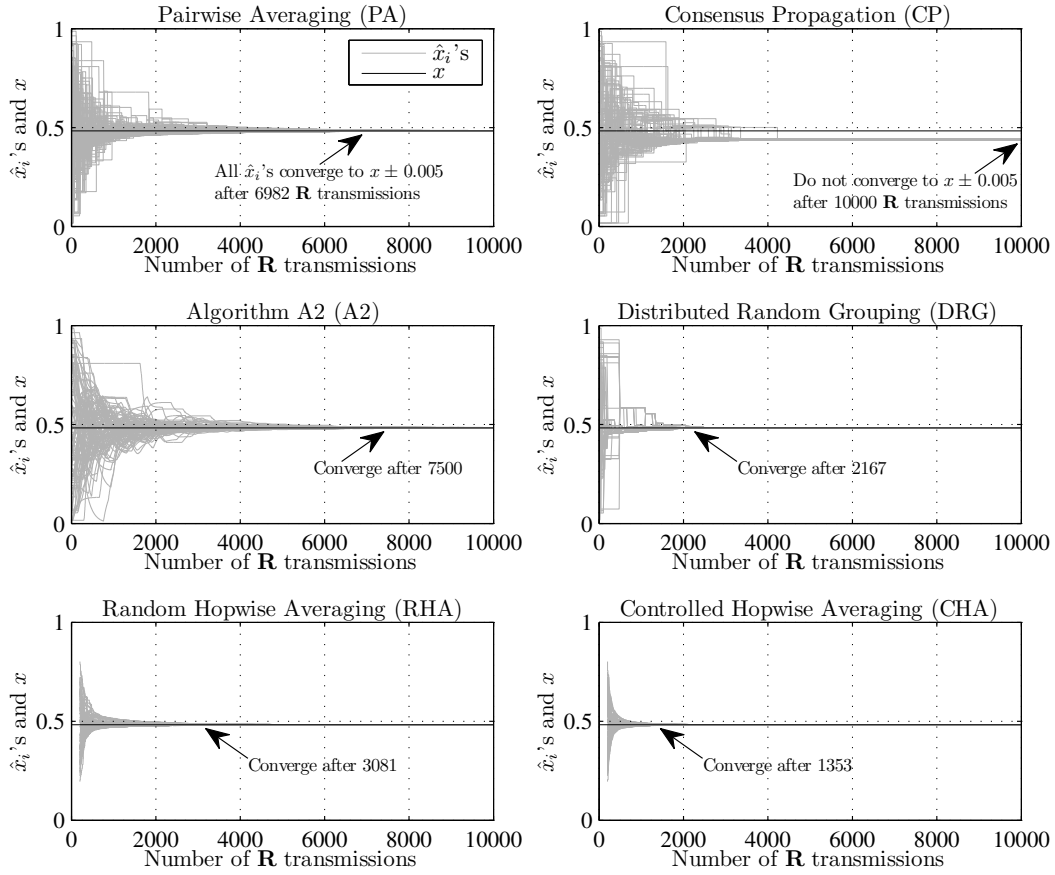

Fig. 2. Performance comparison on the wireless network shown in Figure 1.

and explained the deficiencies of the existing solutions, especially their lack of bandwidth/energy efficiency. To alleviate these deficiencies, we have developed Random and Controlled Hopwise Averaging (RHA and CHA), two distributed asynchronous algorithms capable of fully exploiting the broadcast nature of wireless medium, and the latter capable of enabling greedy, decentralized, feedback iteration control. We have shown that CHA admits a common quadratic Lyapunov function for analysis and control, established its exponential convergence, characterized its worst-case convergence rate, and shown through extensive simulation that it is substantially more efficient than several existing schemes.

REFERENCES

[1] B. Tavli and W. Heinzelman, *Mobile Ad Hoc Networks: Energy-Efficient Real-Time Data Communications*. Berlin, Germany: Springer-Verlag, 2006.

[2] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[3] 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, no. 9, pp. 1520–1533, 2004.

[4] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, 2006.

[5] A. Tahbaz-Salehi and A. Jadbabaie, "Small world phenomenon, rapidly mixing Markov chains, and average consensus algorithms," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 276–281.

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

[7] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[8] D. S. Scherber and H. C. Papadopoulos, "Distributed computation of averages over ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 776–787, 2005.

[9] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proc. American Control Conference*, Minneapolis, MN, 2006, pp. 3551–3556.

[10] A. Olshevsky and J. N. Tsitsiklis, "Convergence rates in distributed consensus and averaging," in *Proc. IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 3387–3392.

[11] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[12] M. Zhu and S. Martínez, "Dynamic average consensus on synchronous communication networks," in *Proc. American Control Conference*, Seattle, WA, 2008, pp. 4382–4387.

[13] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[14] M. Jelasity and A. Montresor, "Epidemic-style proactive aggregation in large overlay networks," in *Proc. IEEE International Conference on Distributed Computing Systems*, Tokyo, Japan, 2004, pp. 102–109.

[15] A. Montresor, M. Jelasity, and O. Babaoglu, "Robust aggregation protocols for large-scale overlay networks," in *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks*, Florence, Italy, 2004, pp. 19–28.

[16] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[17] M. Cao, D. A. Spielman, and E. M. Yeh, "Accelerated gossip algorithms for distributed computation," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2006, pp. 952–959.

[18] J.-Y. Chen, G. Pandurangan, and D. Xu, "Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 987–1000, 2006.

[19] C. C. Moallemi and B. Van Roy, "Consensus propagation," *IEEE*

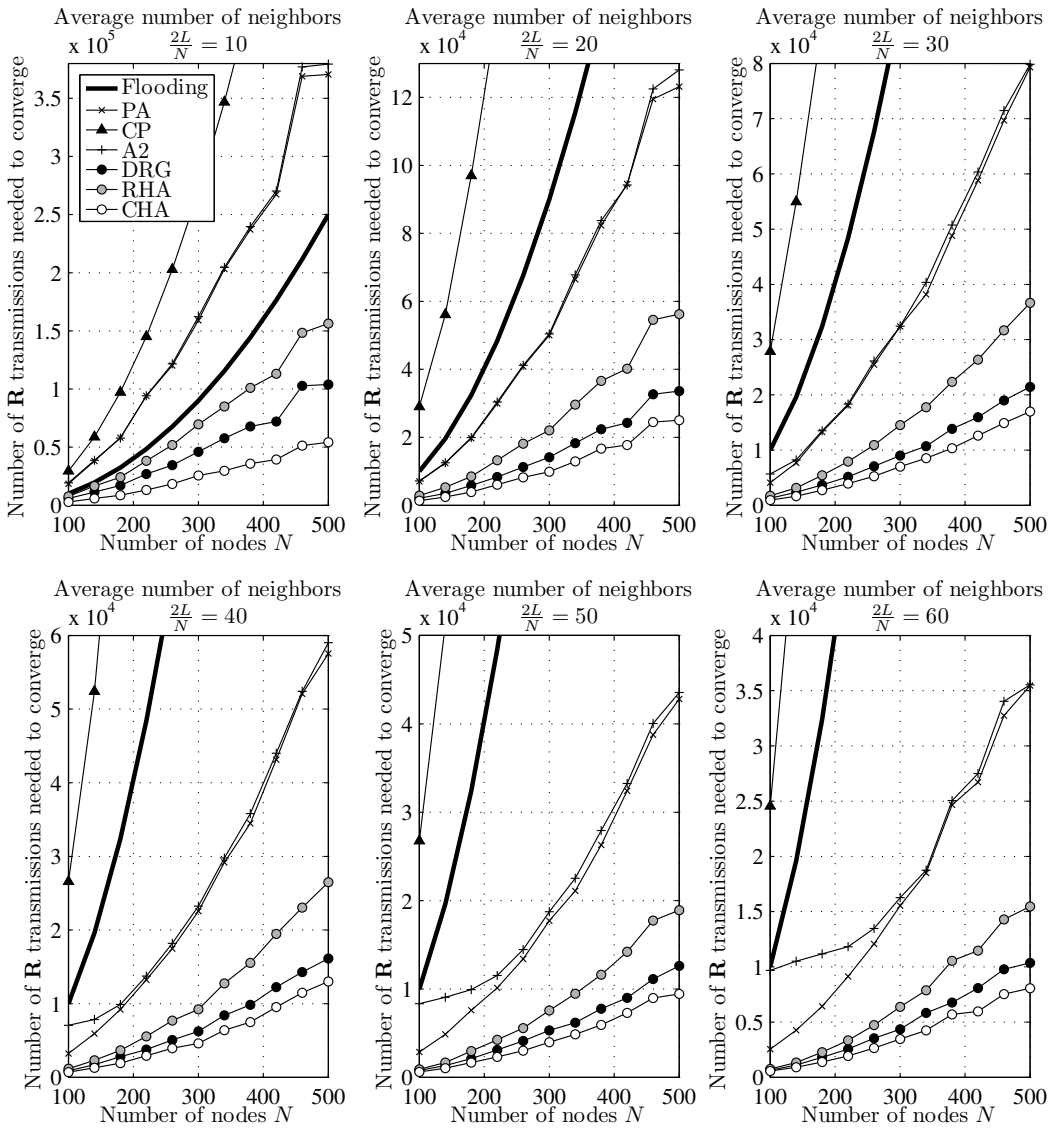

Fig. 3. Performance comparison on randomly generated multi-hop wireless networks with varying number of nodes, varying average number of neighbors, and varying network topologies.

Transactions on Information Theory, vol. 52, no. 11, pp. 4753–4766, 2006.

[20] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Asynchronous distributed averaging on communication networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 512–520, 2007.

[21] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[22] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA: Morgan Kaufmann Publishers, 1996.

[23] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[24] Y. Hatano and M. Mesbahi, "Agreement over random networks," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1867–1872, 2005.

[25] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[26] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[27] L. Fang and P. J. Antsaklis, "On communication requirements for multi-agent consensus seeking," in *Networked Embedded Sensing and Control*, ser. Lecture Notes in Control and Information Sciences, P. J. Antsaklis and P. Tabuada, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 331, pp. 53–67.

[28] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proc. American Control Conference*, New York, NY, 2007, pp. 711–716.

[29] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[30] A. Olshevsky and J. N. Tsitsiklis, "On the nonexistence of quadratic Lyapunov functions for consensus algorithms," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2642–2645, 2008.

[31] C. Y. Tang and J. Lu, "Controlled hopwise averaging: Bandwidth/energy-efficient asynchronous distributed averaging for wireless networks," submitted to *IEEE/ACM Transactions on Networking*, 2009.