# Analysis of Distributed Random Grouping for Aggregate Computation in Wireless Sensor Networks with Randomly Changing Graphs

Jen-Yeu Chen, *Member, IEEE,* and Jianghai Hu, *Member, IEEE*

***Abstract*— Dynamical connection graph changes are inherent in networks such as peer-to-peer networks, wireless ad hoc networks, and wireless sensor networks. Considering the influence of the frequent graph changes is thus essential for precisely assessing the performance of applications and algorithms on such networks. In this paper, using Stochastic Hybrid Systems (SHSs), we model the dynamics and analyze the performance of an epidemic-like algorithm, Distributed Random Grouping (DRG), for average aggregate computation on a wireless sensor network with dynamical graph changes. Particularly, we derive the convergence criteria and the upper bounds on the running time of the DRG algorithm for a set of graphs that are individually disconnected but jointly connected in time. An effective technique for the computation of a key parameter in the derived bounds is also developed. Numerical results and an application extended from our analytical results to control the graph sequences are presented to exemplify our analysis.**

***Index Terms*— Performance Analysis, Aggregate Computation, Sensor Networks, Randomized Algorithms, Distributed Algorithms, Stochastic Hybrid Systems, Graph Theory.**

## I. Introduction

Dynamical graph changes are inherent in networks such as peer-to-peer networks, wireless ad hoc networks, and wireless sensor networks. Take the example of the wireless sensor networks, which have attracted tremendous research interests in the recent years. In a practical or even hostile environment, the connection graph of a sensor network may vary frequently in time due to various reasons. For instance, the communication links (edges of the graph) may fail for being interfered, jammed or obstructed; sensor nodes may be disabled or relocate in the field; to save energy, some sensor nodes may sleep or adjust their transmission ranges, thus altering the connection graph. Algorithms and protocols developed on these networks need to take this nature into consideration. In this setting, distributed and localized[1] algorithms requiring no global data structure, such as routing table or tree hierarchy, are preferable for their scalability and robustness to the frequent graph changes [1], [2], [3], [4], [5].

Although various algorithms have been proposed to deal with networks with dynamical graphs, their performances are usually analyzed under the assumption of a fixed connection graph [2], [3], [4]. In this paper, using the notion of stochastic hybrid systems (SHSs), we present an analytical framework to model the dynamics of a class of distributed and localized algorithms on a network with a time-varying connection graph. As a particular example, we analyze the performance of a distributed randomized algorithm, namely, the DRG (Distributed Random Grouping) algorithm proposed in [2], for average aggregate computation on sensor networks with randomly changing graphs.

Distributed average consensus has been an important problem with many applications in distributed and parallel computing [6]. Recently it also finds applications in the coordination of distributed dynamic systems and multi-agent systems [7], [8], [9], [10], as well as in distributed data fusion in sensor networks [2], [3], [4], [5]. In analyzing the performance of the proposed distributed schemes for average consensus, authors of [2], [3], [4] bound the running time of their schemes on fixed connected graphs. In [7] Olfati-Saber and Murray characterize the convergence speed of their scheme – over a set of possible directed graphs assumed to be balanced and strongly connected – by the minimal algebraic connectivity of the mirror (undirected) graph of each possible network graph. Authors of [5], [8], [9], [10] provide criteria for their schemes to converge on a dynamical changing graph which is possibly disconnected during some time period but do not characterize the convergence speed. Different from these works, the goal of this paper is to not only determine the convergence criteria but also bound the running time of the DRG algorithm on a randomly changing graph which especially could be *disconnected* all the time. It turns out that stochastic hybrid systems provide the appropriate framework for modeling and analyzing such a system of two-fold randomness: one from the randomly changing environment (the connection graph), the other from the execution of the *randomized* algorithm, DRG.

Proposed to model dynamical systems with both continuous and discrete dynamics, a hybrid system has a state that consists of a continuous part and a discrete part (mode). In particular, stochastic hybrid systems are hybrid systems with stochastic continuous dynamics and random discrete mode transitions, and have found applications in a diverse range of scientific and engineering problems such as air traffic management systems, multi-vehicle coordination control, computer networks [11], embedded systems, and biological systems. The average computation on a sensor network with a randomly changing graph can be naturally modeled as a stochastic hybrid system: its discrete mode is the network connection graph which varies with a finite discrete value stochastic process, and its continuous state is the data value stored at sensor nodes, which will be updated in each iteration of the DRG algorithm.

This paper has the following contributions. (a) We explicitly model the dynamics of a distributed randomized algorithm, namely the DRG algorithm, running on a randomly changing

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907. E-mail:jenyeu@ieee.org. and jianghai@purdue.edu.

[1]A localized algorithm in this paper refers to an algorithm only requires local communications, i.e., a node only communicates with its direct one-hop neighbors.

graph by a stochastic hybrid system. This modeling framework may be extended to model other similar algorithms on randomly changing graphs. (b) For the DRG algorithm, we provide the criteria of its convergence (correctness) on randomly changing graphs. (c) By characterizing the graph changing patterns as specific stochastic processes (sequences), we extend our previous analytical results of the DRG algorithm on a fixed connected graph and obtain the upper bound on the running time (the convergence speed) of the DRG algorithm on wireless sensor networks with randomly changing graphs. In particular, the cases considered in this paper is a network graph that randomly switches amid a set of individually disconnected but jointly connected graphs, making this paper the first contribution to deal with this case.

## II. RELATED WORKS

In a wireless sensor network, it is often important to compute statistics such as the average, the maximum/minimum, and the count of data stored in the nodes of the network [12], [13]. In these cases, the information of interest is not the data stored at an individual sensor node, but the aggregate statistics (*aggregates*) amid a group of sensor nodes. Possible applications of aggregates include the average temperature in a area, the minimum remaining battery life of all the sensor nodes, the count of some endangered animal in a natural area, and the maximal noise level in a group of acoustic sensors, to name a few. The operations for computing basic aggregates such as average, max/min, sum, and count can be further adapted to more sophisticated data query or information processing operations.

Many tree-based or multi-path-routing approaches, e.g. the algorithms in [14], [15], [16], [17], have been successfully developed to compute aggregates in a sensor network. Requiring global data structures such as routing tables or the aggregation tree hierarchy, these approaches suffer from high overheads on reconstructing the global data structure when the network graph change frequently. On the other hand, epidemic-like distributed localized algorithms [2], [3], [4], [5] compute aggregates with only local one-hop communications. Without the need to maintain a global data structure, they can be robust and scalable in a large scale and versatile sensor network. Even in the presence of dynamical graph changes, the aggregate computation by these algorithms can continue without interruption; and the error of the computation results will converge to zero under some assumptions on the changing graphs [5], [7]. For more discussions on the advantages of distributed localized algorithms, the readers can refer to [2], [3], [4]. In analyzing the performances of these algorithms, most existing works assume a fixed network graph during the whole computation process, and derive asymptotic bounds on the running time in terms of the *fixed graph's* eigen-structure [2], [3]. These bounds may be inadequate in characterizing the performances of the algorithms on networks with a time-varying graph. For example, bounds obtained by assuming the worst-case network graph are often too conservative. This motivates us to develop analytical tools and frameworks for performance analysis on a time-varying graph.

This paper is organized as follows. In Section III we review some backgrounds and elaborate on the DRG algorithm and its performance on a fixed graph. Then in Section IV, we formally model the dynamics of the DRG algorithm on a randomly changing graph by SHS. From Section V to Section VI, we analyze the performance of the DRG algorithm on a network graph randomly switching among a set of individually disconnected but jointly connected graphs and develop an effective technique to compute a key parameter in the derived performance bounds. Numerical results on four typical sets of such graphs are provided in Section VII. An application based on our analytical results to optimize and control the sensors' sleep/awake schedule is presented in section VIII. Finally, we conclude our work in Section IX.

## III. BACKGROUND AND ASSUMPTIONS

### A. Network graph

The communication graph, or network graph, of a wireless sensor network can be modeled as a geometric graph $G(\mathcal{V}, \mathcal{E})$ where each vertex $v \in \mathcal{V}$ represents a sensor node and each edge, $e \in \mathcal{E}$, is a valid communication link, i.e., two end nodes of an edge can successfully receive the wireless messages from each other in a satisfactory quality such as BER. Since the wireless transmission will degrade by distance due to path loss and fading, according to the transmitting power and the appropriate radio propagation model of the environment, for each sensor node, a radio radius $r$ is decided as the threshold distance within which other senor nodes can receive its wireless transmissions in a satisfactory quality. In this paper we assume that the network graph is time-varying and randomly switches among the elements of a finite set of geometric graphs induced on the same set of nodes, $\mathcal{V}$. In other words, the network graph is a stochastic sequence with the set of all possible graphs as its state space. Note that the geometric graph is not necessarily a *random* geometric graph.

It is important to have a sufficient large radio radius or a sufficient large node density so that the network graph keeps connected [18]. Although the network graph shall be pre-engineered to be connected in the deployment stage, it may become disconnected because some communication links may fail later due to environmental changes or adversary attacks. Also, for saving energy, sensor nodes may periodically sleep or reduce the transmission power, making the network graph disconnected temporarily. In this paper we will prove that the DRG algorithm can successfully compute the average aggregate even though the time-varying network graph is disconnected all the time given the condition that the union of all graphs visited infinitely often is connected.

### B. Distributed Random Grouping – DRG

In our previous work [2], we present a distributed, localized, and randomized algorithm called the *Distributed Random Grouping (DRG)* algorithm to compute aggregate statistics in a wireless sensor network. The DRG algorithm is similar to the Gossip algorithm [3], [4] but with a better performance. It requires only local (one-hop) communications among nodes to save the overhead on constructing and maintaining global data structures such as routing tables or aggregation tree hierarchies. In [2], we show that the performance of the DRG algorithm is related to the eigen-structure of the network graph, which is assumed to be fixed throughout the aggregate computation. Specifically, we use the algebraic connectivity [19], i.e., the second smallest eigenvalue of the Laplacian matrix, of the fixed network graph to bound the running time and the total number of transmissions. The

results show that the DRG algorithm is more efficient than other representative distributed algorithms such as the Flooding [3] algorithm and the Gossip algorithm as it can take advantage of the broadcasting nature of wireless transmissions. In the following, we will briefly describe the DRG algorithm, which will be the focus of this paper in a generalized setting of randomly changing network graphs.

Each sensor node $i$ is associated with an initial observation or measurement value denoted by $v_i(0) \in \mathbb{R}$. The values over all nodes form a initial value vector $\mathbf{v}(0)$. The goal is to compute (aggregate) functions such as the average, sum, max, min, etc. of the entries of $\mathbf{v}(0)$. Throughout this paper we use $v_i(k)$ to denote the value of node $i$ and $\mathbf{v}(k) = [v_1(k), v_2(k), \ldots]^T$ the *value vector* after running the DRG algorithm for $k$ rounds. Note that the $\mathbf{v}(k)$ is not the set of measurements that sensor nodes take from the environment but the estimations of the global aggregate altered from the initial value vector $\mathbf{v}(0)$ after $k$ rounds of the DRG algorithm. The sensor network may take several independent measurements at different times from the environment to have several initial value vectors, e.g. $\mathbf{v}^t(0)$ at time $t$. On these initial value vectors, distinguished by the time stamp $t$, separate processes of the DRG algorithm can be executed to compute their respective aggregates individually.

The main idea of the DRG algorithm is as follows. In each round of the iteration, each node independently becomes a group leader with a probability $p_g$ and then invites its one-hop neighbors to join its group by wireless broadcasting an invitation message[2]. A neighbor who successfully[3] receives the invitation message then join its group. Note that unlike the concept of a *cluster* in the sensor network literature, a group contains only the group leader and its *one-hop* neighbors. Several disjoint groups are thus formed over the network. Next, in each group, all members other than the group leader then send the leader their values so that the leader can compute the *local aggregate* and broadcast it back to the members to update their values. Since in each round, groups are formed at different places of the network, through this randomized process, the values of all nodes will diffuse and mix over the network and converge to the correct aggregate value asymptotically almost surely, provided that the fixed network graph is *connected* [2]. The DRG iterations stop when certain aggregate accuracy criteria are satisfied.

A high-level description of a round (iteration) of the DRG algorithm to compute the average aggregate, is shown in Fig. 1. Aggregates other than the average can be obtained by an easy modification of this algorithm [2]. For simplicity, in this paper we will focus on the average aggregate only.

**Alg: DRG: Distributed Random Grouping for Average**

**1.1** Each node in the ***idle mode*** independently originates to form a group and becomes the group leader with a probability $p_g$.

**1.2** A node $i$ that decides to become a group leader enters the ***leader mode*** and broadcasts a group call message, $GCM \equiv (group_{id} = i)$, to all its neighbors and waits for $JACK$ message from its neighbors.

**2.1** A neighboring node $j$, in the ***idle mode*** and successfully receiving a $GCM$, responds to the group leader by a joining acknowledgment, $JACK \equiv (group_{id} = i, v_j, join(j) = 1)$, with its value $v_j$ included. It then enters the ***member mode*** and waits for the group assignment message $GAM$ from its leader.

**3.1** The group leader, node $i$, gathers the received $JACKs$ from its neighbors; count the total number of group members, $J = \sum_{j \in g_i} join(j) + 1$; and compute the average value of the group, $Ave(i) = (\sum_{k \in g_i} v_k)/J$.

**3.2** The group leader, node $i$, broadcasts the group assignment message $GAM \equiv (group_{id} = i, Ave(i))$ to its group members and then returns to the ***idle mode***.

**3.3** A neighboring node $j$, in the ***member mode*** and upon receiving receiving $GAM$ from its leader node $i$, updates its value $v_j = Ave(i)$ and then returns to the ***idle mode***.

Fig. 1. A round of DRG algorithm to compute average aggregate

## C. Performance of the DRG algorithm on a fixed network graph

In [2], a Lyapunov function called the *potential* (function) is defined to assess the convergence of the DRG algorithm.

***Definition** 1:* Consider an undirected connected graph $G(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$ nodes. Given a value distribution $\mathbf{v}(k) = [v_1(k), ..., v_n(k)]^T$ where $v_i(k)$ is the value of node $i$ after $k$ rounds of the DRG algorithm, the potential $\phi_k$ of round $k$ is defined as

$$\phi_k = ||\mathbf{v}(k) - \overline{v}\mathbf{1}||_2^2 = \mathbf{x}^T(k)\mathbf{x}(k),$$

where the constant $\overline{v} = \frac{1}{n}\sum_{i \in \mathcal{V}} v_i(k)$ is the global average value over the network; the vector $\mathbf{1}$ is the vector with all entries one and $\mathbf{x}(k) = [v_1(k) - \overline{v}, ..., v_n(k) - \overline{v}]^T$ is the error vector.

Running the DRG algorithm on a fixed connected graph, it is easy to show (see [2]) that the potential $\phi_k$ will monotonically decrease to zero from its initial value $\phi_0$, i.e., the values of all nodes will converge to the global average $\overline{v}$ asymptotically almost surely. Given an accuracy requirement $\varepsilon$, we showed in [2] that the upper bound of the running time of the DRG algorithm in a fixed graph $G$ is as follows.

***Theorem** 2:* Given a connected undirected graph $G(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$, and an arbitrary initial value distribution $\mathbf{v}(0)$ with the initial potential $\phi_0$, with a high probability (at least $1 - (\frac{\varepsilon^2}{\phi_0})^{\sigma-1}$ for some $\sigma > 2$), the average aggregate on $G(\mathcal{V}, \mathcal{E})$ can be solved by the DRG algorithm within an $\varepsilon > 0$ accuracy, i.e., $|v_i - \overline{v}| \leq \varepsilon$ for all $i$, in

$$O\left(\frac{\sigma}{\gamma}\log(\frac{\phi_0}{\varepsilon^2})\right) \text{ rounds},$$

where the convergence rate $\gamma := \inf_{\mathbf{v} \neq \overline{v}\mathbf{1}} \left\{E\left[\frac{\delta\phi_k}{\phi_k}\right]\right\} = (1 + \alpha)a(G)\frac{p_g p_s}{d}$ ; $a(G)$ is the algebraic connectivity of the graph $G$ [19]; $\alpha > 1$ is a parameter dependent only on the topology of $G$; $d = \max(d_i) + 1 \approx \max(d_i)$ is the maximum degree of nodes in $G$; $p_g$ is the grouping probability; and $p_s$ is the probability of no collision occurring to a group leaders' group call message, GCM.

When the fixed graph $G$ is a connected graph, the convergence rate is strictly greater than zero, $\gamma > 0$. Otherwise, if the graph $G$ is disconnected, $\gamma = 0$, implying that the DRG algorithm may never converge on a fixed *disconnected* graph.

[2]A wireless broadcast transmission by the group leader can be received by all its one-hop neighbors.

[3]Collisions amid multiple invitation messages from different group leaders may occur at some nodes. Also, a group leader will ignore invitations from its neighbors.

### D. Stochastic hybrid systems

In this section, we will introduce the framework of stochastic hybrid systems that can be used to model the execution of the DRG algorithm on a network with randomly evolving graph.

A hybrid system is a dynamical system whose state $(q, \mathbf{x})$ consists of two parts: (1) a discrete state (mode), $q$, taking values in a discrete set $Q = \{q_1, q_2, \ldots\}$; (2) a continuous state, $\mathbf{x}$, taking values in a continuous space $X = \mathbb{R}^d$. As shown in Fig. 2, the state space of the hybrid system is $Q \times X$, which consists of $|Q|$ copies of $X$. For each mode $q \in Q$, the actual feasible values of $\mathbf{x} \in X$ may be a subset of $X$, called $Dom(q)$, domain of mode $q$, that varies with mode $q$.

To model the dynamics of the DRG algorithm, we need the concept of stochastic hybrid systems [11], [20], [21], [22] which are hybrid systems with stochastic continuous dynamics and random mode transitions. The evolution of a stochastic hybrid system is described by (1) *Continuous dynamics:* the continuous state $\mathbf{x}$ evolves according to stochastic differential equations (SDE) (stochastic difference equation for discrete-time systems) with mode-dependent coefficients; (2) *Discrete dynamics:* mode transitions follow a stochastic process (sequence) defined on $Q$ or occur with a probability when certain conditions, called the *guards* on $\mathbf{x}$, such as the continuous state $\mathbf{x}$ reaches the boundary of the feasible set $Dom(q)$, are satisfies; (3) *Reset conditions:* when a discrete mode transition occurs, the continuous state $\mathbf{x}$ is restarted in the new domain according to some specified rules.

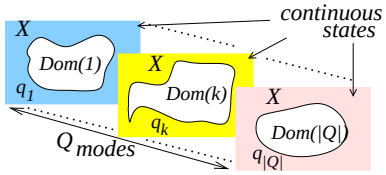



Fig. 2. A possible state space of a hybrid automaton.

We next give the formal definition of hybrid systems.

***Definition** 3:* A hybrid system is a collection $H = (Q, X, Dom, f, \Psi, \mathcal{G}, R)$ where

- $q$ is a discrete variable (mode) taking values in $Q$;
- $\mathbf{x}$ is a continuous variable taking values in $X = \mathbb{R}^d$;
- $Dom : Q \to 2^X$ assigns to each $q \in Q$ a domain $Dom(q)$ of $X$;
- $f : Q \times X \to TX$ are vector fields on $X$ that define the evolution of $\mathbf{x}$ in mode $q$: $\dot{\mathbf{x}} = f(q, \mathbf{x})$ or $\mathbf{x}(k+1) = f(q(k), \mathbf{x}(k))$;
- $\Psi \subset Q \times Q$, where each $(q, q') \in \Psi$ specifies a valid transition from mode $q$ to mode $q'$;
- $\mathcal{G} : \Psi \to 2^X$ assigns to each transition $(q, q') \in \Psi$ a set (called *guard*) $\mathcal{G}(q, q') \subset X$ such that a transition from $q$ to $q'$ occurs whenever $\mathbf{x}$ reaches $\mathcal{G}(q, q')$;
- $R : \Psi \times X \to 2^X$ assigns to each transition $\psi = (q, q') \in \Psi$ the set of values $R(\psi, \mathbf{x}) \subset Dom(q')$ that $\mathbf{x}$ can be reset to after transition from mode $q$ to mode $q'$.

## IV. Modeling the dynamics of DRG on a time-varying network graph by SHS

In our modeling, since the number of possible graphs is finite and discrete, each possible graph can be represented as a discrete mode of the SHS. Therefore, the time-varying network graph is modeled as a stochastic sequence $\{g_k\}_{k=0}^{\infty}$ with that the graph of round $k$ is $g_k$ and the state space is the set of all possible graphs. The network graph only changes at the beginning of each round the DRG algorithm (we will validate this assumption in next sub-section). The values on the sensor nodes are continuous variables; hence they can be chosen as the continuous state of the SHS which evolves by the execution of the DRG algorithm. In summary, the formal definition of the stochastic hybrid system $H_{DRG}$ for modeling the DRG algorithm is given below.

***Definition** 4:* The stochastic hybrid system $H_{DRG} = (Q, X, Dom, f, \mathcal{G}, \Psi)$ is given by

- $Q = \{q\}$ is the set of all possible graphs and the graph of round $k$ is $g_k \in Q$;
- $X$ : $\mathbf{x} = \mathbf{v} - \bar{v}\mathbf{1} \in \mathbb{R}^n$ is the offset value distribution;
- $Dom$: $Dom(q) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \perp \mathbf{1}\}$ consists of all $\mathbf{x}$ whose entries add up to zero;
- $f$: $\mathbf{x}(k+1) = W(k)\mathbf{x}(k)$ for some random matrix $W(k)$ dependent to graph $g_k$ and the grouping rules of the DRG algorithm.
- $\Psi$: the switching rules of the graph sequence $\{g_k\}_{k=0}^{\infty}$;
- $\mathcal{G} = X$; $R$ is the trivial reset: $R((q, q'), \mathbf{x}) = \mathbf{x}$.

### A. Validation of the graph changing model

In our model, we assume that the graph only changes at the beginning of each round of the DRG algorithm. Within a round of the DRG algorithm, the graph is fixed. This assumption is valid due to two facts: (1) the DRG algorithm uses only local one-hop communications and (2) the graph changes occurring in the middle of a round have the same effect as if they occur at the beginning of a round due to the implementation of messages passing mechanism in the DRG algorithm. For the first fact, as all communications are local, the time period of a round of the DRG algorithm is very short so that the probability that the network graph keeps the same in a round is large. In this case, the possible complicated analysis of a multi-hop routing on a time-varying graph is also avoided. Further, for the second fact, which is the main reason, if a communication link fails (hence the graph changes) in the middle of the transmission of a GCM message or a JACK message, the corresponding neighbor will not join the group, which is of the same effect as a link fails at the beginning of a round. A link failure happening in the middle of a GAM message can be easily detected by the group leader through the timeout of the Acknowledgment of the GAM message. A leader who does not receive the Acknowledgment from one of its group members needs to roll back and excludes the corresponding group member from the computation of the local aggregate. This exclusion occurring at the stage of GAM message has the same effect as the exclusion due to a link failure occurring at the beginning of a round.

## V. Performance of DRG on time-varying network graphs

It is nontrivial to extend our results of the DRG algorithm on a fixed connected graph in [2] to the general case of time-varying graphs. For a fixed graph, we have shown in [2] that all the node values will eventually reach consensus by converging to the global average, starting from an arbitrary initial value if and only if the graph is connected. However, in the case when the graph is time-varying, even if the graph is disconnected in some time periods,

it is still possible that consensus can be reached, provided that the union of the graphs appearing infinitely often is connected. (This convergence criterion has been shown in [5], [8] for their schemes. Later, we will briefly prove that it is also valid for the DRG algorithm.)

Moreover, characterizing the convergence rate of the DRG algorithm in this case is a challenging task, as it depends on the possible graphs of the network, as well as the rules for the (random) evolution of the network graph in time. If some (at least one) of the possible graphs are connected and visited infinitely often, then our results in [2] can be directly extended. We can conservatively lower bound the convergence rate for the whole execution by the minimum among those non-zero convergence rates on connected and infinitely-often-visited graphs. If these graphs are visited according to some particular frequency, e.g., the stationary distribution of an ergodic stochastic process, then we can obtain a normalized convergence rate

$$\gamma* = 1 - \prod (1 - \gamma_G)^{p_G}, \tag{1}$$

where $\gamma_G$ is the convergence rate on a candidate graph $G$ and $p_G$ is the frequency that the graph $G$ is visited. We refer readers to [23] for the performance analysis of the DRG algorithm on two useful stochastic models for the time-varying network graph, namely, independently and identical distributed process and pure jump process. In these two stochastic models, at least one candidate graph visited infinitely often is connected so that the aforementioned normalized convergence rate can be derived.

If all graphs are disconnected, the results in [2] can not be directly applied to find the normalized convergence rate since the convergence rate on each graph is zero ($\gamma = 0$) for all graphs. However, if the union of all graphs visited infinitely often is connected, the DRG algorithm can still converge. In the following subsections we will first prove this convergence criterion and then show the convergence rate of the DRG algorithm running on a network graph randomly switching among *a set of individually disconnected but jointly connected graphs*.

**Remark:** A graph $G_u(\mathcal{V}, \mathcal{E}_u)$ is the union of two graphs, $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$, induced on the same set of vertices $\mathcal{V}$ if and only if $\mathcal{E}_u = \mathcal{E}_1 \cup \mathcal{E}_2$, i.e., $G_u(\mathcal{V}, \mathcal{E}_u) = G_1(\mathcal{V}, \mathcal{E}_1) \cup G_2(\mathcal{V}, \mathcal{E}_2) \Longleftrightarrow \mathcal{E}_u = \mathcal{E}_1 \cup \mathcal{E}_2$. If the union of a set of graphs is connected, for simplicity, we say that this set of graphs are jointly connected.

### A. Convergence criterion for DRG on time-vary network graphs

We can show the convergence criterion, i.e., the union of all graphs visited infinitely often is connected, by extending the proof of Theorem 1 in [5]. Note that this convergence criterion is general enough for any set of possible graphs, but not just for a set of individually disconnected but jointly connected graphs. Suppose there are $|Q| < \infty$ possible graphs, where $Q = \{G_i\}$, each of which is visited infinitely often. On each graph $G_i$ there are a set of possible group assignments $\{\mathcal{D}_{G_i}\}$ resulting from the randomized grouping rule of the DRG algorithm. Each $\mathcal{D}_{G_i}$ is associated with a double stochastic, symmetric and paracontracting [4] matrix $W^{\mathcal{D}_{G_i}}$ (w.r.t. Euclidean norm) so that the value vector is updated by $\mathbf{v}(k+1) = W^{\mathcal{D}_{G_i}}\mathbf{v}(k)$ when $\mathcal{D}_{G_i}$ occurs at round $k$. (The value updating matrix $W_i$ of [5] depends only on the chosen network graph $G_i$; but our $W^{\mathcal{D}_{G_i}}$ is determined by the group assignment $\mathcal{D}_{G_i}$ which in turn depends on the graph $G_i$ and the *randomized* grouping strategy of the DRG algorithm.) Each $G_i$ is visited infinitely often, so is $\mathcal{D}_{G_i}$. Hence, there exists at least a set of updating matrices $\Gamma = \{W^{\mathcal{D}_{G_i}}\}$ for $i = 1 \cdots |Q|$ such that $\mathcal{M} = \frac{1}{|Q|}\sum_{i=1}^{|Q|} W^{\mathcal{D}_{G_i}}$ is stochastic, symmetric and irreducible if the union of possible graphs is connected. This implies that $\mathcal{M}$'s fix-point subspace, i.e., the eigenspace associated with the eigenvalue 1, $\mathcal{H}(\mathcal{M}) = \mathbf{span}(\mathbf{1})$. Therefore, by [5], $\bigcap_\Gamma \mathcal{H}(W^{\mathcal{D}_{G_i}}) = \mathbf{span}(\mathbf{1})$, which by [25] leads to the conclusion that the DRG algorithm will asymptotically converge to the *unique fixed point* $\left(\frac{1}{n}\mathbf{1}^T\mathbf{v}\right)\mathbf{1}$, i.e., the status of the average consensus.

### B. Performance of DRG on individually disconnected but jointly connected graphs

We consider a model where all the possible graphs are individually disconnected but jointly connected. We assume that each graph occurs with some positive probability in a round of the DRG algorithm and is visited infinitely often in the whole stochastic graph sequence. In such a model, the expected potential decrement in a single round in the worst case is uniformly zero, i.e., $\gamma_q = 0$, $\forall q$. We can not directly extend the results of [2] to find a positive normalized convergence rate by (1). However, even though all possible graphs are disconnected, since their union graph is connected, the DRG algorithm can still converge to global average. In the following we derive the convergence rate and the upper bound of running time for the DRG algorithm running on such a set of graphs.

For illustration purpose, we analyze the simplest case where the network graph switches randomly infinitely often between two graphs, namely $G_1$ and $G_2$, which are individually disconnected but jointly connected. Our analysis can be easily extended to the general case of switching amid more than two graphs. Fig. 11(a) is one of the simplest examples of such a set of graphs. Note that $G_1$ and $G_2$ can be of any graph topology and size as long as they are individually disconnected but jointly connected.

In each round, the network graph can be either $G_1$ or $G_2$. Hence in the stochastic hybrid system model, the space of discrete modes is $Q = \{G_1, G_2\}$. The mode transition pattern can be characterized by a two-state Markov chain shown in Fig. 12(a). Since $G_1$ and $G_2$ are each disconnected, the lower bound on the convergence rate for each of them in a single round is zero. However, in two rounds, the network may switch between these two jointly connected graphs with positive probability, resulting in a positive expected potential decrement. Thus to lower bound the expected potential decrement rate, we need to consider two rounds of DRG iterations. Since this is a worst-case analysis, we assume the worst scenario: only one group is formed in each round. The DRG algorithm can have more groups in a round and hence converge faster than the upper bound derived here. Also, we assume every node has equal probability to become a leader. Without loss of generality, define $\mathbf{x}(k) = \mathbf{v}(k) - \overline{v}\mathbf{1}$, which is orthogonal to the vector $\mathbf{1}$, i.e., $\mathbf{x}(k) \perp \mathbf{1}$. Then each round of DRG iteration can be expressed as $\mathbf{x}(k+1) = W(k)\mathbf{x}(k)$ for some random matrix $W(k)$ depending on the choice of group leader. For example, if in round $k$, the network graph is $g_k \in \{G_1, G_2\}$ and node $i$ becomes the group leader, then $W(k) = W^{g_k,i} = [w_{\eta\varsigma}^{g_k,i}]$

[4] A matrix $W$ is paracontracting w.r.t. a vector norm $\|\cdot\|$ if $Wx \neq x \Leftrightarrow \|Wx\| < \|x\|$. [24]

where

$$w_{\eta\varsigma}^{g_k,i} = \begin{cases} \frac{1}{d_i+1}, & \text{if } \eta,\varsigma \in \{N_{g_k}(i) \cup i\}; \\ 1, & \text{if } \eta,\varsigma \notin \{N_{g_k}(i) \cup i\} \text{ and } \eta = \varsigma; \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Here $N_{g_k}(i)$ is the set of neighbors of node $i$ in graph $g_k$.

In summary, the formal definition of the stochastic hybrid system $H_{DRG,\,j.c.}$ for modeling the DRG algorithm on this randomly changing graph model is given below.

***Definition** 5:* The stochastic hybrid system $H_{DRG,\,j.c.} = (Q, X, Dom, f, \mathcal{G}, \Psi)$ is given by

- $Q = \{G_1, G_2\}$ and the graph of round $k$ is $g_k \in Q$;
- $X:\ \mathbf{x} = \mathbf{v} - \bar{v}\mathbf{1} \in \mathbb{R}^n$ is the offset value distribution;
- $Dom$: $Dom(q) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \perp \mathbf{1}\}$ consists of all $\mathbf{x}$ whose entries add up to zero;
- $f$: $\mathbf{x}(k+1) = W(k)\mathbf{x}(k)$ for some random matrix $W(k)$ defined by (2);
- $\Psi:\ P(g_{k+1} = G_2 | g_k = G_1) > 0$; $P(g_{k+1} = G_1 | g_k = G_2) > 0$;
- $\mathcal{G} = X$; $R$ is the trivial reset: $R((q, q'), \mathbf{x}) = \mathbf{x}$.

From the continuous dynamics, in two rounds, we have $\mathbf{x}(k+2) = W(k+1)W(k)\mathbf{x}(k) = \widetilde{W}\mathbf{x}(k)$. The ratio of potential decrement after two rounds is

$$\frac{\phi_k - \phi_{k+2}}{\phi_k} = \frac{\|\mathbf{x}(k)\|^2 - \|\mathbf{x}(k+2)\|^2}{\|\mathbf{x}(k)\|^2} = 1 - \frac{\mathbf{x}(k)^T \widetilde{W}^T \widetilde{W} \mathbf{x}(k)}{\mathbf{x}(k)^T \mathbf{x}(k)}. \quad (3)$$

Define $\gamma_2$ as the lower bound on the expected convergence rate after two consecutive rounds:

$$\gamma_2 \equiv \inf_{\substack{\mathbf{x}(k) \perp \mathbf{1}; \\ \mathbf{x}(k) \neq \mathbf{0}}} \left\{ E\left[ \frac{\phi_k - \phi_{k+2}}{\phi_k} \right] \right\}.$$

From (3), we have

$$E\left[\frac{\phi_k - \phi_{k+2}}{\phi_k}\right] = 1 - \frac{\mathbf{x}(k)^T E\left[\widetilde{W}^T \widetilde{W}\right] \mathbf{x}(k)}{\mathbf{x}(k)^T \mathbf{x}(k)}$$
$$= 1 - \frac{\mathbf{x}(k)^T \mathbf{K}\, \mathbf{x}(k)}{\mathbf{x}(k)^T \mathbf{x}(k)} \geq 1 - \lambda_2(\mathbf{K}) > 0, \quad (4)$$

where $\lambda_2(\mathbf{K})$ is the *second largest* eigenvalue of the matrix $\mathbf{K}$ defined by

$$\mathbf{K} = E\left[\widetilde{W}^T \widetilde{W}\right] = \sum_{(g_k, g_{k+1})} \sum_{i,j} \frac{P_{g_k, g_{k+1}}}{n^2} (W^{g_{k+1},j} W^{g_k,i})^T (W^{g_{k+1},j} W^{g_k,i}).$$

In the above, $P_{g_k,g_{k+1}} = P(g_k)P(g_{k+1}|g_k)$ is the probability that the graph of round $k$ is $g_k$ and the graph of round $k+1$ is $g_{k+1}$. So, the lower bound on the expected convergence rate after two consecutive rounds is

$$\gamma_2 = 1 - \lambda_2(\mathbf{K}) > 0.$$

Note that the largest eigenvalue of $\mathbf{K}$ is always one so the second largest eigenvalue $\lambda_2(\mathbf{K}) < 1$ since the two possible graphs are jointly connected.

***Theorem** 6:* For the SHS $H_{DRG,\,j.c.}$ with an arbitrary initial value distribution $\mathbf{v}(0)$ and the initial potential $\phi_0$, with high probability (at least $1 - (\frac{\varepsilon^2}{\phi_0})^{\sigma-1}$ ; $\sigma > 2$), the average consensus problem can be solved by the DRG algorithm within an $\varepsilon > 0$ accuracy, i.e., $|v_i - \overline{v}| \leq \varepsilon$ for all $i$, in

$$O\left(\sigma \log_{\lambda_2(\mathbf{K})}(\frac{\varepsilon^2}{\phi_0})\right) \text{ rounds.}$$

*Proof:* To meet the accuracy criterion after $2\tau$ rounds of the DRG algorithm, by (4), we need $E[\phi_{2\tau}] \leq (1-\gamma_2)^\tau \phi_0 = (\lambda_2(\mathbf{K}))^\tau \phi_0 \leq \varepsilon^2$, from which we get $\tau \geq \log_{\lambda_2(\mathbf{K})}(\frac{\varepsilon^2}{\phi_0})$. Choose $\tau = \sigma \log_{\lambda_2(\mathbf{K})}(\frac{\varepsilon^2}{\phi_0})$ where the $\sigma \geq 2$. Then because $(\frac{\varepsilon^2}{\phi_0}) \ll 1$ and $(\sigma - 1) \geq 1$, the Markov inequality

$$P(\phi_{2\tau} > \varepsilon^2) < \lambda_2(\mathbf{K})^{\sigma \log_{\lambda_2(\mathbf{K})}(\frac{\varepsilon^2}{\phi_0})} (\frac{\phi_0}{\varepsilon^2}) = (\frac{\varepsilon^2}{\phi_0})^{\sigma-1}. \quad (5)$$

Thus, $P(\phi_{2\tau} \leq \varepsilon^2) \geq 1 - (\frac{\varepsilon^2}{\phi_0})^{(\sigma-1)}$ is arbitrarily close to 1 by choosing large $\sigma$. (Since typically $\phi_0 \gg \varepsilon^2$, taking $\sigma = 2$ is sufficient to have high probability at least $1 - O(\frac{1}{n})$; in case $\phi_0 > \varepsilon^2$, then a larger $\sigma$ is needed to have a high probability). Hence running the DRG algorithm for $2\tau = O\left(\sigma \log_{\lambda_2(\mathbf{K})}(\frac{\varepsilon^2}{\phi_0})\right)$ rounds, , with high probability $1 - (\frac{\varepsilon^2}{\phi_0})^{(\sigma-1)}$, the DRG algorithm converges to an $\varepsilon$ accuracy. ■

Similar procedures can be carried out to obtain the convergence rate for network graphs randomly switching among a set of individually disconnected but jointly connected graphs consisting of more than two graphs. In the following section, we provide an effective way to compute the compound matrix, $\mathbf{K}$, for two useful families of individually disconnected but jointly connected graphs on an arbitrarily large set of nodes.

## VI. COMPUTATION OF THE COMPOUND MATRIX $\mathbf{K}$

From the previous section, the compound matrix $\mathbf{K}$ is a key element in the upper bound of the running time of the DRG algorithm. Here we introduce an effective way to compute the compound matrix $\mathbf{K}$ for two representative families of graphs whose $n$ and $|Q|$ can be arbitrarily large.

The computation of matrix $K$ can be very time consuming if the possible graphs are complicated. However, due to the fact that the sparser the graph, the longer the time the DRG algorithm will take to converge, the sparest graph, namely, linear array, with only one edge connecting two nodes in each possible graph, is considered in this section as the *worst case benchmark* for the performance of the DRG algorithm running on a set of individually disconnected but jointly connected graphs. Another sparse graph, a star graph with only one edge connecting two nodes in a time, is also presented for comparison.

### A. The compound matrix $\mathbf{K}$ for linear arrays

The possible graphs are from a set of $h = n - 1$ graphs each with $n$ nodes positioned as a linear array and with only one edge connecting two consecutive nodes. Shown in Fig. 3(a) is a possible graph with an edge between node $m$ and node $m+1$, and in Fig. 11(c) is an example set of all possible graphs with $n = 4$. Each graph itself is disconnected but the union of these graphs together is connected. Thus, the DRG algorithm will converge to the global average. Because of the extreme sparsity of each graph, a time-varying network graph randomly switching among these graphs is among the worst cases for the DRG algorithm to converge. In each round, we conservatively assume that only one leader is chosen among the $n$ nodes with equal probability. As a result, unless one of the two end nodes of the only edge becomes

a leader, the transition matrix $W(k)$ is the identity matrix $I_n$ of order $n$. For example, suppose that the network graph is $G_m$ in Fig. 3(a) at round $k$. When node $m$ or node $m+1$ becomes the group leader,

$$W(k) = W^{G_m,m} = W^{G_m,m+1} = \mathbf{diag}\left(I_{m-1}, \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}, I_{n-m-1}\right); \quad (6)$$

when the other nodes become the leader, $W(k)$ is the identity matrix of order $n$, i.e., $W(k) = W^{G_m, i \notin \{m,m+1\}} = I_n$.

For convenience, we define two independent sequences: the graph sequence, $\Lambda = \{g_k\}_{k=1}^h$, namely the realization of the randomly switching network graph over rounds $k = 1, \dots h$; and the leader sequence, $l = \{l_k\}_{k=1}^h$, where $l_k$ is the leader node for the randomly switching network graph $g_k$ at round $k$. The leader sequence $l = \{l_k\}_{k=1}^h$ is an i.i.d. sequence where each $l_k$ is of the uniform distribution over all $n$ nodes. We re-write

$$\mathbf{K} = E\left[\widetilde{W}^T\widetilde{W}\right] = \sum_{\Lambda} P(\Lambda) E\left[\widetilde{W}^T\widetilde{W}\middle|\Lambda\right] = \sum_{\Lambda} P(\Lambda)\mathbf{K}_\Lambda.$$

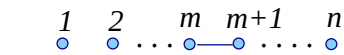
The graph sequence $\Lambda$ and $P(\Lambda)$ depend on the graph changing pattern; and the matrix $\mathbf{K}_\Lambda \equiv E\left[\widetilde{W}^T\widetilde{W}\middle|\Lambda\right]$ is the compound matrix for a given graph sequence $\Lambda$. Since the computation of $\mathbf{K}_\Lambda$ is the key to computing $\mathbf{K}$, in the following, we illustrate the computation of $\mathbf{K}_\Lambda$ through an example.

Because $W(k)^T = W(k)$ and the group leaders in different rounds are chosen independently, we have

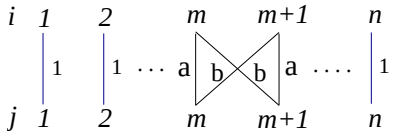
$$\begin{aligned}\mathbf{K}_\Lambda = E\left[\widetilde{W}^T\widetilde{W}\middle|\Lambda\right] &= E\left[\prod_{k=1}^h W(k)\prod_{k=1}^h W(h-k+1)\middle|\Lambda\right] \\ &= \sum_l P(l = \{l_k\}_{k=1}^h|\Lambda)\cdot \mathbf{W}_{l,\Lambda} \\ &= \sum_l P(l_1|\Lambda)\cdots P(l_h|\Lambda)\cdot \mathbf{W}_{l,\Lambda} \\ &= \sum_l P(l_1|g_1)\cdots P(l_h|g_h)\cdot \mathbf{W}_{l,\Lambda},\end{aligned}$$

where $\mathbf{W}_{l,\Lambda} = W^{g_1,l_1}\cdots W^{g_h,l_h}\cdot W^{g_h,l_h}\cdots W^{g_1,l_1}$ and each $W^{g_k,l_k}$ is the $W(k)$ decided by $g_k$ and $l_k$ at round k. Note that $W^{g_k,l_k}$ appears twice on the right hand side. Since we assume that there is only one leader in each round, $P(l_k|g_k) = 1/n$, $1 \le k \le h$.

Each $W(k) = W^{g_k,l_k}$ is a basic computation block for the computation of $\mathbf{K}_\Lambda$ and can be represented as a bipartite graph. An example of this basic computation block for $W(k) = W^{g_k,l_k} = [w_{ij}^{g_k,l_k}]$ on the graph $g_k = G_m$ of Fig. 3(a) is shown in Fig. 3(b). Specifically, corresponding to each entry $w_{ij}^{g_k,l_k}$ of $W(k)$, there is a link with weight $w_{ij}^{g_k,l_k}$ connecting the upper node (entry node) $i$ and the lower node (exit node) $j$ of the bipartite graph. Those links with zero weight ($w_{ij}^{g_k,l_k} = 0$) are omitted since they will not contribute to the computation of $\mathbf{K}_\Lambda$. For the example of Fig. 3(b), with probability $\frac{2}{n}$, when either of node $m$ or $m+1$ becomes the leader, $a = b = \frac{1}{2}$ , i.e., $W(k)$ is given by (6); with probability $\frac{n-2}{n}$, $a = 1$, $b = 0$, i.e., $W(k)$ will be the identity matrix of order $n$. To compute $\mathbf{K}_\Lambda$ these computation blocks are cascaded as in Fig. 4. The computation

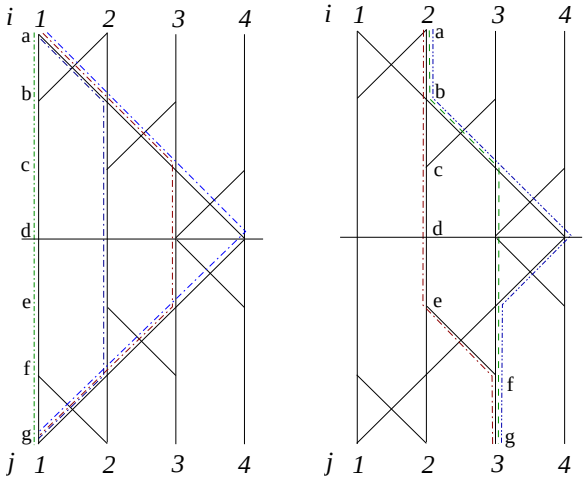



(a) An example graph $G_m$ of linear array with only nodes $m$ and $m+1$ connected.




(b) The computation block for $W(k)$ on the graph $G_m$, where with probability $\frac{2}{n}$, $a = b = \frac{1}{2}$ and with probability $\frac{n-2}{n}$, $a = 1$, $b = 0$.

Fig. 3. An example graph of linear array and its corresponding computation block for computing $\mathbf{K}_\Lambda$.




(a) paths for $\mathbf{K}_\Lambda(1,1)$ (b) paths for $\mathbf{K}_\Lambda(2,3)$

Fig. 4. The un-solid lines are the possible paths (multiplication combinations) for an entry of the matrix $\mathbf{K}_\Lambda$.

blocks are mirror symmetric across the horizontal line in Fig. 4: the first and the last computation blocks are the same, etc.

As an example, we show how to compute $K_\Lambda$ for a graph sequence $\Lambda = \{g_k\}_{k=1}^h$ in which $g_k$ has only one edge connecting nodes $k$ and $k+1$ (i.e., $m = k$ in Fig. 3(a)). Corresponding to the graph sequence $\Lambda$, the computation blocks are cascaded in a way as in Fig. 4 for the case $n = 4$, i.e., $h = 3$. We define a path $\psi := (s_1, \dots, s_f, \dots, s_{2h+1})$ to be a possible node sequence starting from the entry node $s_1$ of the top (first) computation block to the exit node $s_{2h+1}$ of the bottom (last) computation block. Each intermediate node $s_f$ is the entry node of the $f$-th computation block and the exit node of the $(f-1)$-th computation block. Fig. 4 illustrates the possible paths for computing $\mathbf{K}_\Lambda(1,1)$ and $\mathbf{K}_\Lambda(2,3)$, where $\mathbf{K}_\Lambda(1,1)$ has four different paths and $\mathbf{K}_\Lambda(2,3)$ has three different paths. For generalization, denote a path with the first node $s_1 = i$ and the last node $s_{2h+1} = j$ by $\psi(i,j)$, and the set of all possible $\psi(i,j)$ by $\Psi(i,j) = \{\psi := (s_1, \dots, s_{2h+1}) | s_1 = i, s_{2h+1} = j\}$. In the example of Fig. 4(a), the number of possible paths for $K_\Lambda(1,1)$ is $|\Psi(1,1)| = 4$. In general, the number of possible paths for computing $\mathbf{K}_\Lambda(i,j)$ is

$$|\Psi(i,j)| = n - \max(0, \max(i,j) - 2).$$

Furthermore, define the *average weight* of a path $\psi$ as

$$\bar{w}(\psi) = \sum_{l} \prod_{k=1}^{h} P(l_k|g_k) \cdot w_{s_k s_{k+1}}^{g_k,l_k} \cdot w_{s_{2h-k+1} s_{2h-k+2}}^{g_k,l_k}$$
$$= \prod_{k=1}^{h} \sum_{l} P(l_k|g_k) \cdot w_{s_k s_{k+1}}^{g_k,l_k} \cdot w_{s_{2h-k+1} s_{2h-k+2}}^{g_k,l_k}, \quad (7)$$

where $w_{s_k s_{k+1}}^{g_k,l_k}$ is the $(s_k s_{k+1})$ entry of the matrix $W^{g_k,l_k} = W(k)$ of round $k$. Also, the last equality in the above equation follows from the fact that $l = \{l_k\}_{k=1}^{h}$ is an i.i.d. sequence. Each entry of $\mathbf{K}_\Lambda$, i.e., $\mathbf{K}_\Lambda(i,j)$, therefore can be computed by summing the average weights of all possible paths connecting the entry node $s_1 = i$ on the top to the exit node $s_{2h+1} = j$ at the bottom:

$$\mathbf{K}_\Lambda(i,j) = \sum_{\psi \in \Psi(i,j)} \bar{w}(\psi).$$

Take the possible path $\psi_1 := (a = 1, b, c, d, e, f, g = 1)$ of Fig. 4(a) for example. With probability $2/n$, the weights of link $(a,b)$ and $(f,g)$ are both $1/2$; with probability $(n-2)/n$, they are both 1. All the other links on this path are always of weight 1 (with probability 1). Hence by (7) the average weight of path $\psi_1$ is

$$\bar{w}(\psi_1) = \frac{2}{n} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{n-2}{n} \cdot 1 \cdot 1 = \frac{2n-3}{2n},$$

where $n = 4$ in this example. As another example for the possible path $\psi_2 :=(a = 2, b, c, d, e, f, g = 3)$ in Fig. 4(b), with probability $2/n$, the weight of $(a,b)$ is $1/2$ and the weight of $(f,g)$ is 1, while with probability $(n-2)/n$, both the weights of $(a,b)$ and $(f,g)$ are 1. (Hence, the weight of $(f,g)$ is always 1 in this case.) In order for the link $(e,f)$ to have nonzero weight, node 2 or node 3 must be the leader of round 2, which occurs with probability $2/n$. In this case, the weights of $(b,c)$ and $(e,f)$ are both $1/2$. The average weight of this path is therefore

$$\bar{w}(\psi_2) = \left(\frac{2}{n} \cdot \frac{1}{2} \cdot 1 + \frac{n-2}{n} \cdot 1 \cdot 1\right) \left(\frac{2}{n} \cdot \frac{1}{2} \cdot \frac{1}{2}\right) = \frac{n-1}{n} \frac{1}{2n}.$$

In this equation, we only need to consider two rounds, since at round 3, the links $(c,d)$ and $(d,e)$ are of weight 1 with probability one.

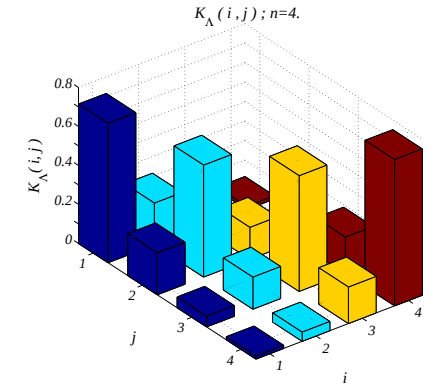
From these cascading computation blocks, we obtain the analytical expression of $K_\Lambda$ as follows. Let $r = \frac{1}{2n}$, $\zeta = (1-3r)$ and $\Im = (2r)^{x-1}(r + r\zeta\frac{1-r^{n-x-i}}{1-r} + r^{n+1-x-i})$; $\mathbf{K}_\Lambda(i,j) =$

$$\begin{cases} \zeta\frac{1-r^h}{1-r} + r^h, & i = j = 1; \\ r + \zeta^2\frac{1-r^{n-i}}{1-r} + \zeta r^{n-i}, & 1 < i = j \le n; \\ \Im, & i = 1,\ j = 1+x, 1 \le x \le n-1; \\ (1-2r)\Im, & 1 < i \le n,\ j = i+x, 1 \le x \le n-i; \\ \mathbf{K}_\Lambda(j,i), & i > j. \end{cases}$$

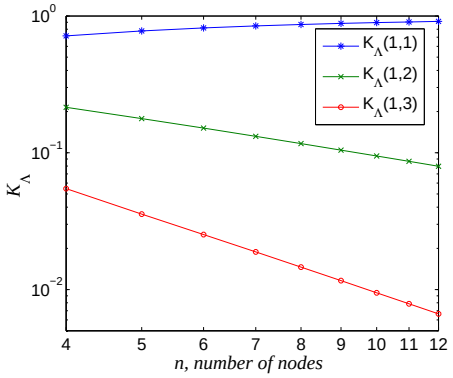
Note that $\mathbf{K}_\Lambda$ is a double stochastic matrix which can be easily verified from the above expression.

The computed $K_\Lambda$ is shown in Fig. 5(a) for $n = 4$ and in Fig. 5(b) for $n = 4, \ldots, 12$. It can be seen from Fig. 5(b) that the matrix $\mathbf{K}_\Lambda$ will approach the identity matrix of order $n$ when $n$ is large, i.e., if $n \to \infty$, then $r \to 0$, $K_\Lambda(i,i) \to 1$ and $K_\Lambda(i, i+x) \sim \frac{1}{n^x}$, indicating that the second largest eigenvalue of the matrix $K_\Lambda$ will approach 1, i.e., $\lambda_2(K_\Lambda) \to 1$, as $n \to \infty$.

If the graph sequence $\Lambda$ occurs with probability 1, i.e., if the graph changing pattern is deterministic (a special case of the general random switching setting), then $\mathbf{K} = \mathbf{K}_\Lambda$ can be directly




(a) $K_\Lambda$, $n = 4$.




(b) The trend of $K_\Lambda$ in logarithm axes, when $n$ is increasing from 4 to 10. Three typical entries of $K_\Lambda$ are shown.

Fig. 5. Properties of $K_\Lambda$.

obtained from the above computation of $\mathbf{K}_\Lambda$. Taking this case as an example to discuss the convergence trend of the DRG algorithm on such a set of graphs, we show the second largest eigenvalue $\lambda_2(\mathbf{K}) = \lambda_2(\mathbf{K}_\Lambda)$ and the convergence rate of the DRG algorithm $\gamma_2$ in Fig. 6 for $n = 4, \ldots, 12$. It can be seen that the larger the number $n$, the smaller the convergence rate $\gamma$, implying a slower convergence of the DRG algorithm while computing the aggregates on such kind of graphs. The reason is straightforward. Only when an end node of the only edge of each graph becomes a group leader will the DRG averaging process really take effect to reduce the value variations on nodes. When the number of node $n$ becomes large, the chance of the two end nodes of the only edge independently becoming a leader dwindles, slowing down the convergence process.

Running the DRG algorithm on such a set of $h = n-1$ possible graphs, the convergence rate $\gamma$ is the minimal ratio of the expected potential decrement $E[\delta\phi]$ after $h$ rounds of the DRG algorithm to the (known) original potential, i.e., $\gamma \le E\left[\frac{\phi_k - \phi_{k+h}}{\phi_k}\right]$. For fair comparison, we also show in Fig. 6 the normalized parameter $\lambda_2^* = \sqrt[h]{\lambda_2}$ and normalized convergence rate $\gamma^* = 1 - \lambda_2^*$ which indicates the minimal ratio of the expected potential decrement $E[\delta\phi]$ after a round of the DRG algorithm to the potential at the beginning of that round of the DRG algorithm, following from the relationship $E[\phi_{k+h}|\phi_k] \le \lambda_2\phi_k = (\lambda_2^*)^h\phi_k$.

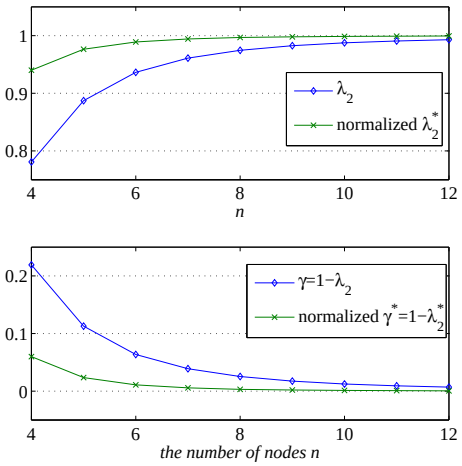



Fig. 6. The second largest eigenvalue of compound matrix $\mathbf{K}$, $\lambda_2(\mathbf{K})$, and convergence rate, $\gamma = 1 - \lambda_2(\mathbf{K})$ vs the number of nodes, $n$, in the jointly connected linear graphs, under a deterministic graph changing pattern $\Lambda$.

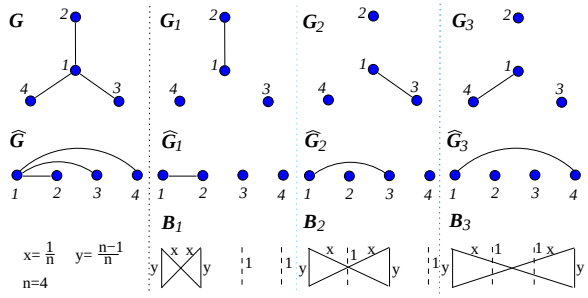
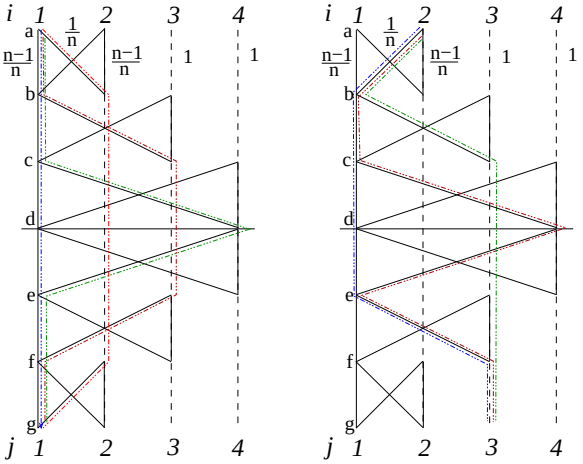



Fig. 7. The star topology, disconnected star graphs and their corresponding computation blocks.

### B. The compound matrix $\mathbf{K}$ for star topology

Another useful graph topology is the star topology, which is also a common topology for networks. We illustrate a simple example for $n = 4$ nodes in Fig. 7. The graph $G$ at the top left of Fig. 7 is the jointly connected star graph with four nodes. For convenience we can rearrange graph $G$ into a linear array topology as graph $\hat{G}$ shown below $G$. We consider the graph sequence $\Lambda = \{G_1, G_2, G_3\}$, which is of the equivalent representation $\Lambda = \{\hat{G}_1, \hat{G}_2, \hat{G}_3\}$. From the linear arrangement of nodes, it becomes clear that the principle of cascading computation blocks in the previous example of linear array can also be applied again here. We show the corresponding computation blocks $B_1$, $B_2$, $B_3$ at the bottom of graphs $\hat{G}_1$, $\hat{G}_2$, $\hat{G}_3$. Similar to the previous example of linear array, to compute $\mathbf{K}_\Lambda$, we cascade the computation blocks in a way in Fig. 8 where Fig. 8(a) illustrates the possible paths for $\mathbf{K}_\Lambda(1, 1)$ and Fig. 8(b) shows those for $\mathbf{K}_\Lambda(2, 3)$.

To generalize this star topology, we number the center node as node '1', which sequentially connects to only one other node in each round, i.e. there is only one edge in each round. We number the node connected in round $k$ as node $k+1$. There will be a total of $h = n - 1$ rounds. Let $r = (\frac{n-1}{n})^2$ and $\zeta(j) = \frac{1-r^{(n-j)}}{1-r}$. By Fig. 8, we obtain, for star topology with $n$ nodes, the compound




(a) paths for $\mathbf{K}_\Lambda(1, 1)$ (b) paths for $\mathbf{K}_\Lambda(2, 3)$

Fig. 8. The possible paths (multiplication combinations) for an entry of the matrix $\mathbf{K}_\Lambda$ in star topology.

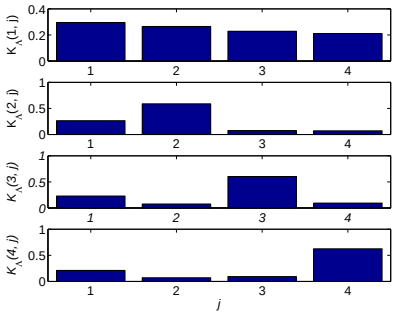
matrix

$$\mathbf{K}_\Lambda(i,j) = \begin{cases} r^h + \frac{1}{n^2}(\frac{1-r^h}{1-r}), & i = j = 1; \\ \frac{1}{n} r^{\frac{1}{2}(2n-j-1)} + r^{\frac{1}{2}(j-1)}(\frac{1}{n} + \frac{\zeta(j)}{n^3}), & i = 1,\ 1 < j \le n; \\ r + \frac{r^{(n-i)}}{n^2} + \frac{\zeta(j)}{n^4}, & 1 < i = j \le n; \\ \frac{r^{\frac{1}{2}(j-i)}}{n^2}\left[1 + r^{(n-j)} + \frac{\zeta(j)}{n^2}\right], & 1 < i < j \le n; \\ \mathbf{K}_\Lambda(j,i), & i > j. \end{cases}$$

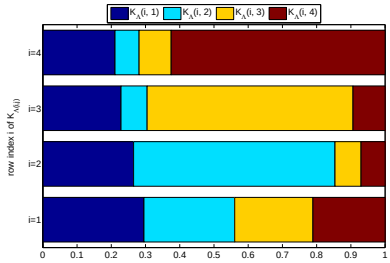
Fig. 9(a) shows the matrix $K_\Lambda$ for a star topology with four nodes, i.e., $n = 4$. Each sub-figure presents a row of the matrix $K_\Lambda$. It is also easy to verify that the $K_\Lambda$ is a double stochastic matrix by Fig. 9(b), Every column of $K_\Lambda$ is depicted by a distinct color whereas the values on each row are summed up together to be 1. Note that $K_\Lambda$ is a symmetric matrix.

We also compare the normalized convergence rate $\gamma^* = 1 - \lambda_2^*$ of the DRG algorithm on linear topology with that on star topology. It is shown in Fig. 10 that with larger normalized convergence rate the DRG algorithm will converge faster in star topology than in the linear topology of the same size. This is because, in the star topology, the center node is always connected to be a bridge for data exchanges, providing a better connection. Meanwhile, the diameter of the star topology is only two whereas the diameter of the linear topology will go up to $h = n - 1$. Any two nodes in the star topology need at most two edges to exchange data but in the linear topology they may require as many as $h$ edges.

We also see from Fig. 10 that both the normalized convergence rates of two topologies decrease at least exponentially fast with the number of nodes $n$. (Note that the y-axis of bottom sub-figure is in logarithmic scale.) The normalized convergence rate of the linear topology drops faster than that of the star topology. We can use the slope of each line in the bottom sub-figure of Fig. 10 as the *scalability indicator* of a set of switching graphs of the same size to run the DRG algorithm—-for the extreme example, the slope close to zero indicates a constant normalized convergence rate of the DRG algorithm regardless of the size of the graph. So given a threshold $n_s$, the *scalability* $\Gamma_{n_s}$ of the DRG algorithm




(a) The values of the matrix $K_\Lambda$ on a star topology.




(b) The row stack chart for $K_\Lambda$ on a star topology.

Fig. 9. The $K_\Lambda$ on a star topology with the number of nodes $n = 4$.

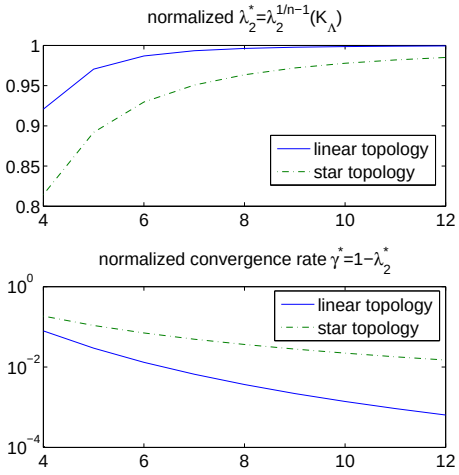



Fig. 10. The normalized $\lambda_2^* = \sqrt[h]{\lambda_2}$ and the normalized convergence rate $\gamma^* = 1 - \lambda_2^*$ for the linear and star topologies in various graph sizes from $n = 4$ to $n = 12$.

on a set of switching graphs $\mathbb{G}$ is

$$\Gamma_{n_s}(\mathbb{G}) = \max_{n>n_s} \left( \frac{-n}{\ln \gamma^*(n)} \right).$$

## VII. Numerical Results

We now present some numerical results on the convergence rate of the DRG algorithm on the randomly graph-changing models studied in Section V-B. Recall that, in this case, there are a total of $h$ possible graphs for the network, each of which is disconnected. As a family, however, the $h$ graphs are jointly connected. A lower bound $\gamma_h$ on the $h$-step convergence rate is given by $\gamma_h = 1 -$

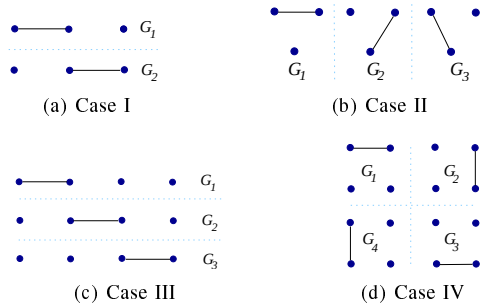



Fig. 11. Example cases for numerical results.

$\lambda_2(\mathbf{K})$, where $\mathbf{K} = E\left[\widetilde{W}^T\widetilde{W}\right]$ and $\widetilde{W} = W(k+h-1)\cdots W(k+1)W(k)$.

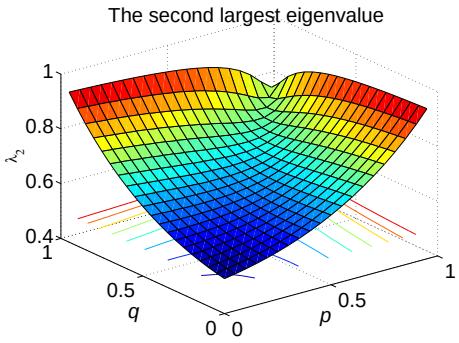
In Fig. 11, four cases under study are plotted. In case I and case III, the union of possible graphs forms a linear array with three and four nodes, respectively. In case II and case IV, the union of possible graphs forms a ring with three and four nodes, respectively. The Markov chains (the randomly graph-changing model) describing the transitions among possible graphs are shown in Fig. 12: Fig. 12(a) for case I; Fig. 12(b) for case II and case III; and Fig. 12(c) for case IV. We compute $\gamma_2$ for case I, $\gamma_3$ for case II and III, and $\gamma_4$ for case IV, under different transition probabilities $p$ and $q$.

Fig. 13(a) plots the computed $\lambda_2(\mathbf{K})$ of case I as a function of the transition probabilities $p$ and $q$. It can be seen that, as $p$ and $q$ both approach 0, $\lambda_2(\mathbf{K})$ achieves its minimum; hence $\gamma_2 = 1 - \lambda_2(\mathbf{K})$ achieves its maximum, implying the fastest convergence rate of the DRG algorithm. This is understandable as, in this case, the transitions between the two possible graphs are the most frequent and occur in each round, remedying the slow convergence caused by the individual disconnected graph. On the other hand, by requiring that $p + q = 1$, $\lambda_2(\mathbf{K})$ becomes a function of $p$ only, and is plotted in Fig. 13(b). Note that the plot in Fig. 13(b) is a slice of the plot in Fig. 13(a) along the line $p + q = 1$. As can be seen from the plot, the minimum $\lambda_2(\mathbf{K})$, hence the maximal convergence rate $\gamma_2$, occurs at $p = q = 0.5$ when the two graphs have identical stationary probability 0.5. For all other choices of $p$ and $q$ satisfying $p + q = 1$, the transitions have a tendency of staying in one graph longer, which slows down the convergence of the DRG algorithm.

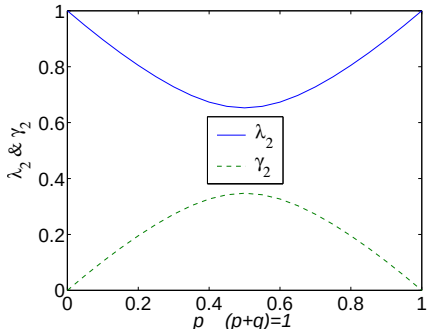
Fig. 13(c) compares the convergence rates of the DRG algorithm for these four cases as well as an additional case of the ring topology that is of five disconnected graphs, $h = 5$. The computed convergence rate $\gamma_h$ for these five cases are plotted in Fig. 13(c) as functions of the transition probability $p$ (in case I, we set $q = p$). We observe that, the larger the number of nodes, the slower the convergence rate. In addition, with the same number of nodes, the case whose union graph is a linear array has the slower convergence rate than the corresponding case whose union graph is a ring. This is because on a linear array each of the two end nodes has only one direction to spread out its value whereas all nodes in a ring have two.

## VIII. An Application: Sleep/Awake Scheduling

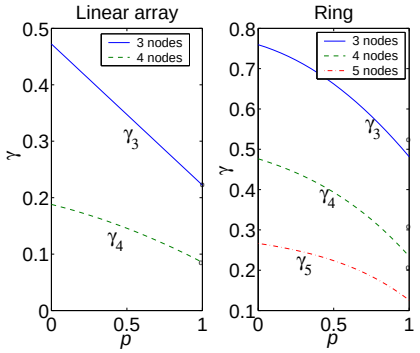
One of the efforts to save energy consumption in sensor network is to let some sensor nodes sleep (in power saving mode) from time to time without affecting the correctness of



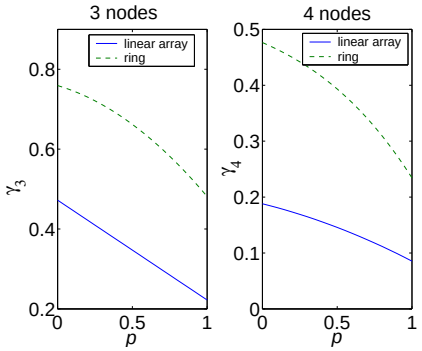

(a) $\lambda_2(\mathbf{K})$ for the two-mode model.




(b) $\lambda_2(\mathbf{K})$ and $\gamma_2$ for the two-mode model when $p+q=1$.

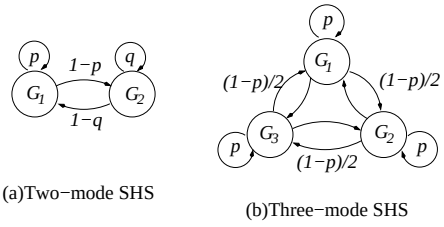



(c) Lower bound $\gamma$ for different types of graphs.




(d) $\gamma$ for various graphs with the same number of nodes.

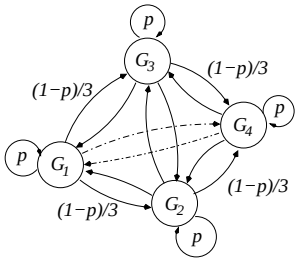
Fig. 13. Numerical results




(a)Two−mode SHS




(b)Three−mode SHS




(c)Four−mode SHS

Fig. 12. The Markov chain for jointly connected switching graphs each of which is disconnected.

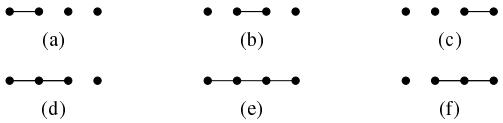



Fig. 14. Connection graphs for sleep/awake scheduling

the execution of the algorithm but possibly with some acceptable degradation on the performance of the algorithm. In our example, the network graph may become disconnected when some nodes sleep. This is especially true for sparse network graphs. However, from the results of previous sections, we know that the DRG algorithm still converges as long as the time-varying network graph is jointly connected. In this section we discuss the DRG's performance on several sleep/awake scheduling sequences and try to find the best controlled graph sequence in terms of both energy saving and convergence time.

We consider a sparse graph: a linear array with 4 sensor nodes in a row. The network graph "e" of Fig. 14(e) is the connected graph while all four nodes are awake. Other graphs in Fig. 14 are disconnected because some sensor nodes are in sleep mode, e.g., in graph "a" (Fig. 14(a)) node 3 and node 4 are in sleep mode. When a node sleeps, its CPU is at power saving mode and its radio components are deactivated. We compare nine different periodic graph sequences (i.e., different sleeping schedules for sensor nodes): $\Lambda_1 = \{e\}$, $\Lambda_2 = \{abc\}$, $\Lambda_3 = \{abcb\}$, $\Lambda_4 = \{abce\}$, $\Lambda_5 = \{dc\}$, $\Lambda_6 = \{df\}$, $\Lambda_7 = \{adefc\}$, $\Lambda_8 = \{edef\}$, $\Lambda_9 = \{eaebec\}$, where $\{abc\}$ means that the network graph repeats in "abc" pattern periodically, i.e., in first round the network graph is Fig. 14(a), the second round Fig. 14(b), the third round Fig. 14(c) and the fourth round Fig. 14(a) again ... etc. Since the graphs in each sequence are jointly connected, the DRG algorithm will converge for these graph sequences.

Running the DRG algorithm on an $n$-nodes linear array with $m_k$ awake nodes, we model the expected energy consumption in

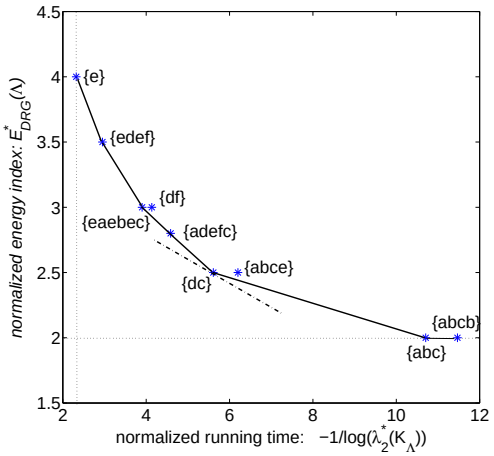



Fig. 15. The normalized energy per round and normalized convergence time for different graph sequences.

the round $k$ of the DRG algorithm as follows.

$$\begin{aligned} E_{DRG} &= \frac{2}{n}(3E_{tx} + 2(E_{r/w} + E_{CPU-active})) \\ &+ \frac{m_k - 2}{n}(4E_{tx} + 3(E_{r/w} + E_{CPU-active})) \\ &+ \frac{m_k}{n}(E_{CPU-awake} + E_{rx}) \\ &= \frac{m_k}{n}(3E_0 + E_{tx} + E_1) - \frac{2}{n}E_0, \end{aligned}$$

where $E_0 = E_{tx} + E_{r/w} + E_{CPU-active}$ and $E_1 = E_{CPU-awake} + E_{rx}$; $E_{tx}$ is the energy for transmitting a message and $E_{rx}$ is for nodes to listen to MAC channels and receive messages. Every awake node consumes the same $E_{rx}$ even though it does not receive any messages in a round. In a round of the DRG algorithm, the CPU of an awake node consumes $E_{CPU-awake}$ and an extra $E_{CPU-active}$ when it becomes busy, i.e. the awake but idle nodes consume only $E_{CPU-awake}$. The $E_{r/w}$ is the total energy required for an awake node to read/ write information from/to its EEPROM in a round of the DRG algorithm. Except the number of awake nodes $m_k$, all the other parameters in the above equation are constants in rounds of the DRG algorithm. (For detail energy quantities consumed by a sensor node, we refer to [26].) The total energy consumption $E_{DRG}$ in a round of the DRG algorithm is a linear function of the number of awake nodes $m_k$ which may vary by rounds. To compare the average energy consumed in a round for different graph sequences, we take the average number of awake nodes

$$E^*_{DRG}(\Lambda) = \sum_{k\in\Lambda} m_k/|\Lambda|$$

, where $|\Lambda|$ is the number of graphs of a sequence pattern, as the normalized energy index for the sequence $\Lambda$, e.g., for $\{dc\}$ the normalized energy index is $E^*_{DRG}(\{dc\}) = (3+2)/2 = 2.5$.

From Theorem 2, given $\phi_0$ and $\epsilon$, the running time is proportional to $-|\Lambda|\log^{-1}(\lambda_2(\mathbf{K}_\Lambda))$. Thus, we define the normalized index for running time of the DRG algorithm

$$Time^*(\Lambda) = -\log^{-1}(\lambda^*_2(\mathbf{K}_\Lambda)) = -|\Lambda|\log^{-1}(\lambda_2(\mathbf{K}_\Lambda)).$$

Fig. 15 shows our simulation results for the nine graph sequences mentioned previously. The x-axis is the normalized index for running time; the y-axis is the normalized energy index $E^*_{DRG}(\Lambda)$. The sequence $\Lambda_1 = \{e\}$ where nodes never sleep consumes the most energy but converges fastest. In contrast, the sequence $\Lambda_3 = \{abcb\}$ where two nodes sleep in turn round by round consumes least energy but converges the slowest. There will be always a tradeoff between these two performance indices. To incorporate both energy consumption and convergence time into the performance assessment, we can minimize the combined indices:

$$\mathbf{min}(\alpha \cdot E^*_{DRG}(\Lambda) + (1-\alpha)\cdot Time^*(\Lambda)).$$

Two extreme cases are $\alpha = 1$ and $\alpha = 0$, representing the consideration of only energy consumption and only convergence time correspondingly. By linear programming on the convex hull of Fig 15, we can find the proper graph sequence for a desired $\alpha$. For example, the sequence $\Lambda_5 = \{dc\}$ should be used when we set $0.0982 < \frac{1-\alpha}{\alpha} < 0.2926$.

## IX. Conclusion and Future Works

To guarantee the correctness and precisely bound the running time of an algorithm developed on a sensor network, we need to consider one of the senor network's salient nature: frequently changing graphs. In this paper, we model the execution of the Distributed Random Grouping (DRG) algorithm for computing the average aggregate on a sensor network with randomly changing graphs by stochastic hybrid systems (SHSs). Criteria are given for the convergence of the DRG algorithm on a randomly changing graph. Particularly for two families of graphs which are individually disconnected but jointly connected, bounds on the convergence rate and the running time are presented. Numerical results are provided to illustrate our analytical results. An application built on our analytical results to optimize and control the sleep/awake schedule for sensor nodes is also introduced.

This work can be extended in several ways. For example, we can use SHS to model the dynamics of the sensed values and develop a new version of the DRG algorithm that exploits the auto-correlation of the sensed data to expedite the convergence of aggregation; the SHSs framework can also be applied to model and analyze other similar distributed localized algorithms such as gossip and flooding for aggregate computation on a sensor network with a time-varying network graph.

## Acknowledgments

This work was partially supported by the National Science Foundation under Grant CNS-0643805, and by the Purdue Research Foundation.

## References

[1] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Mobicom*, 1999, pp. 263–270.

[2] J.-Y. Chen, G. Pandurangan, and D. Xu, "Robust aggregate computation in wireless sensor network: distributed randomized algorithms and analysis," in *IEEE Trans. on parallel and distributed system,*, Sep. 2006, pp. 987–1000.

[3] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. FOCS*, 2003.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *INFOCOM'05*.

[5] L. Xiao, S. Boyd., and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. IPSN*, 2005.

[6] N. Lynch, *Distributed Algorithms*. Morgan KauRman Publishers, 1997.

[7] R. Olfati-Saber and R. M. Murray, "Consensus problem in networks of agents with switching topology and time delays," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, sep 2004.

[8] L. Fang and P. Antsaklis, "Information consensus of asynchronous discrete-time multi-agent systems," in *Proc. ACC'05*.

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

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

[11] J. Hespanha, "Polynomial stochastic hybrid systems," in *Hybrid Systems: Computation and Control, 8th Int. Workshop (HSCC 2005)*. Springer Verlag, 2005, pp. 322–338.

[12] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 137–142, 2003.

[13] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor netowrks," in *Proc. SPNA*, 2003.

[14] B. Krishnamachari, D. Estrin, and S. Wicker, "Impact of data aggregation in wireless sensor networks," in *Proc. DEBS*, 2002.

[15] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Proc. OSDI*, 2002.

[16] M. Enachescu, A. Goel, R. Govindan, and R. Motwani, "Scale-free aggregation in sensor networks," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 15–29, 2005.

[17] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proc. SenSys*, 2004.

[18] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity in wireless networks," in *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, W.M. McEneaney, G. Yin, and Q. Zhang (Eds.), Birkhauser, Boston*, 1998.

[19] M. Fiedler, "Algebraic connectivity of graphs," in *Czechoslovak Math. J., vol. 23*, 1973, pp. 298–305.

[20] M. Bujorianu and J. Lygeros, "General stochastic hybrid systems: modeling and optimal control," in *43rd IEEE CDC*, 2004, pp. 1872–1877.

[21] G. Pola, M. Bujorianu, J. Lygeros, and M. D. Benedetto, "Stochastic hybrid models: an overview," in *ADHS*, 2003.

[22] J. Hu, J. Lygeros, and S. Sastry, "Towars a theory of stochastic hybrid systems." in *In Proceeding of HSCC*, 2000, pp. 160–173.

[23] J.-Y. Chen and J. Hu, "Analysis of a class of distributed randomized algorithms on randomly changing network graphs," in *Technical Report, TR ECE 07-22, Purdue Univ.*, 2007, submitted for publication.

[24] D. J. Hartfiel, *Nonhomogeneous Matrix Products*. World Scientific, 2002.

[25] L. Elsner, I. Koltracht, and M. Neumann, "On the convergence of asynchronous paracontractions with applications to tomographic reconstruction from incomplete data," *Linear Algebra Appl.*, vol. 130, pp. 65–82, 1990.

[26] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. SenSys*, 2004.


**Jen-Yeu Chen** obtained his BS and MS degrees from National Cheng Kung University, Taiwan, R.O.C., and his Ph.D degree from Purdue University, USA (2007), all in Electrical Engineering. Prior to studying at Purdue University, he was with Chunghwa Telecom Laboratories Co., LTD, Taiwan, R.O.C., as an associate researcher. His current research interests are the design and probabilistic analysis of algorithms in the areas of communications, networking and control with focuses on wireless networks, distributed systems, and multi-agent systems. He is a member of IEEE.

**Jianghai Hu** (S'99-M'04) received the B.E. degree in automatic control from Xi'an Jiaotong University, P.R. China, in 1994, and the M.A. degree in Mathematics and the Ph.D. degree in Electrical Engineering from the University of California, Berkeley, in 2002 and 2003, respectively. He is currently an assistant professor at the School of Electrical and Computer Engineering, Purdue University. His research interests include hybrid systems, multi-agent coordinated control, control of systems with uncertainty, and applied mathematics.