2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

# Consensus-based Estimation Protocol for Decentralized Dynamic Load Balancing over Partially Connected Networks

Zhuoyao Wang*, Majeed M. Hayat*‡, Mahshid Rahnamay-Naeini*, Yasamin Mostofi*, and Jorge E. Pezoa*†
*Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA
E-Mail: zywang@unm.edu, {hayat,mrahnama,ymostofi,jpezoa}@ece.unm.edu
‡M. M. Hayat is also with Center for High Technology Materials, University of New Mexico, Albuquerque, NM, USA
†J. E. Pezoa is also with Electrical Engineering Department, Universidad de Concepción, Concepción, Chile

***Abstract*—A novel consensus-based protocol is developed for estimating the load information at nodes in a distributed computing system operating over a partially connected communication network. The challenge in such estimation process arises from the existence of tangible delays in the exchange of information, and the fact that loads are dynamic since nodes continue to execute their loads while the estimation process is ongoing. The protocol utilizes the concept of trust weight that each node has about any other node, based on the number of hops in between, to periodically form an updated estimate of the loads of the other nodes. The probability of consensus at any given time as well as the probability density function of the time to the first consensus are analytically characterized and used to determine the best instant for executing a dynamic load balancing (DLB) action. Detailed Monte-Carlo simulations of the average completion time of a workload by a distributed system under different system configurations are presented and discussed. The results suggest a range of time for executing the DLB over which satisfactory average completion time is achieved. The results also provide insight on the effects of network connectivity and the frequency of communication on the DLB performance.**

***Index Terms*—consensus-based estimation, distributed computing, partially connected network, load balancing, trust-weight**

## I. Introduction

Distributed computing systems (DCSs) offer an efficient and inexpensive way to process workloads composed of a large number of independent tasks in a cooperative manner. The completion time required to execute a workload on a DCS is mainly dependent on the distribution of tasks on the computing nodes as well as their individual processing power. As computing nodes in the system process tasks, some nodes may be overloaded while other nodes may run out of tasks, thereby becoming idle. To avoid such scenarios that defeat the purpose of cooperative computing, load balancing (LB) techniques are used [1] so that the system resources are utilized optimally, and consequently the workload completion time is minimized. Specifically, the goal of LB is to give every computational node its fair share of the workload so that *ideally* no node becomes idle prematurely before the *entire* workload is completed. The LB problem belongs to a more general class of problems in resource allocation. These problems appear not only in DCSs but also in routing in wireless networks, telecommunications, and other problems in computer science and operational research [2]–[4].

From the perspective of the type of system information (online vs. off-line) used by the DCS to perform task redistribution, LB can be categorized as either dynamic or static [5]. In dynamic load balancing (DLB), the current state of the DCS is utilized to redistribute tasks across nodes. From the perspective of the structure of the decision-making process, on the other hand, there are two methods to pursue DLB in a DCS: centralized and decentralized. In the centralized DLB [6], a central coordinating node fuses the global load information of the DCS and conducts appropriate LB decisions. For the decentralized DLB [7], [8], no such central coordination exits; nodes execute DLB autonomously based on their own local estimates of the loads at other nodes. The estimation is facilitated by a process of information exchange among the nodes that accompanies the DLB process. Decentralized DLB is a more useful approach for many distributed systems, such as wireless sensor networks and cloud computing, because it alleviates the dependence on a centralized control node that can make the system vulnerable to its failure.

The performance of decentralized DLB also relies vitally on the accurate estimation of the load at each node. As nodes exchange information, they form their own individual estimates of the loads across the DCS based upon the received information, which can be local or global. When a DCS operates over a partially connected network, where nodes communicate over multiple hops, the information exchange is not as effective as that in the scenario for which a fully connected network is utilized. This is because of the presence of tangible delays in communication, especially between distant nodes, that deem the information dated. Furthermore, the frequency of information exchange is equally important in forming accurate estimates of the loads at the nodes. Due to bandwidth or energy constraints in certain practical distributed systems such as wireless sensor networks [9], the frequency of information exchange can be limited. We emphasize that both communication delays and infrequent load-information exchange affect the quality of estimates of the load at each node. This is because nodes continue to process tasks during

978-1-61284-799-3/11/$26.00 ©2011 IEEE

the time when the information is exchanged; as such, the load of the system is dynamic. Thus, lack of accurate global load information in real time at every node cannot be ignored in DCSs operating over a partially connected network.

While DLB can be executed repeatedly over the span of time for which the workload is being executed, limiting the number of DLB executions and optimizing the choice of the DLB instant have been shown to be a viable and more-efficient alternative approach [10], [11]. This motivated us to look into the so-called one-shot DLB policy [11]: once a workload arrives a DLB is to be executed at an optimized "best" instant. Clearly, there is a trade-off between delaying the DLB instant, in order to acquire more accurate estimates by receiving more information, and wasting computing resources as nodes may be kept idle due to large delays before a DLB execution.

In as much as it is clear that large estimation errors of the loads (across the system) degrade the performance of DLB, it is less obvious that any disagreement among nodes in their estimates of the loads also degrades the performance. Our experience suggests that in most cases of interest, the inconsistency along the nodes in the network about their estimates of the loads can worsen the performance of the DLB compared to cases of similar average estimation error but with agreed-upon estimates. Disagreement in the estimated loads results in conflict in LB decisions, which causes unnecessary task transfers among nodes. In a simple scenario of two disagreeing nodes, for example, one node may send tasks to the other node while the other node may also send tasks to the first node concurrently [7]. Therefore, estimation mechanisms leading to consensus cases can be beneficial to DLB [12].

Gonzalez-Ruiz and Mostofi [12] studied the problem of DLB over partially connected networks and examined the advantage of consensus amongst nodes in estimating the average overall load in the DCS. This advantage was viewed in the context of reducing the time required to reach the so-called "balanced state" of the DCS. According to [12], an informed load-balancing (I-LB) policy is defined as that for which nodes exchange information about their locally estimated global load and reach an agreement over the "balanced state" of the system before executing DLB repeatedly. In [12] DLB is performed locally, namely, tasks are transferred only among neighboring nodes. They showed that in the case of the I-LB approach, the number of LB actions required so that all nodes attain their fair share of the load is less compared to the scenario where DLB is executed repeatedly *without* prior consensus about the "balanced state."

While the work in [12] points to the benefit of consensus-based estimation of loads it assumes that nodes do not process tasks during the time when consensus is being reached. Therefore, the load at each node prior to DLB actions is assumed to be deterministic and constant; hence, the value of a "balanced state" to be agreed upon is fixed. However, the load of the nodes is not only dynamic but also stochastic as the processing powers of nodes change stochastically in a shared-computing environment due to other processes in the system, randomness in length of tasks and varying memory latencies in the computing nodes [13]. We further point out that in the problem stated in this paper, the consensus equilibrium to be estimated by nodes, namely the unknown vector representing the global load information of the DCS, is time varying and stochastic. Meanwhile, an effective DLB requires the estimation error to be small. Therefore, the problem at hand is different from the canonical consensus problem where the dynamics of the states are modeled by deterministic differential equations [14]–[16]. As such, linear-consensus protocols and related analysis proposed in the literature are not suitable for our problem.

In this paper we propose a novel consensus-based protocol for estimating the load information at nodes in a DCS setting operating over a partially connected communication network. Our formulation assumes that the load at each computing node is both dynamic and stochastic. The estimation protocol utilizes the concept of trust weight, that each node has of any other node based on the number of hops in between, to periodically form an updated estimate of the loads of the other nodes by fusing the information it receives from neighboring nodes. The consensus-based protocol is of practical interest for DLB because it provides a simple and efficient way to disseminate the information among the network. We analytically characterize the probability of consensus at any given time as well as the probability density function (PDF) of the time to the first consensus; these quantities facilitate our understanding of the best instant for executing a DLB after a workload is assigned to the DCS. Because the load is dynamic and stochastic, there is no guarantee that the consensus of the true global load information will be achieved asymptotically. Detailed Monte-Carlo (MC) simulations on the average completion time of a workload under different system configurations are presented and discussed. The results provide insight on the effects of network connectivity and frequency of communication on the DLB performance.

This paper is organized as follows. In Section II we propose a novel consensus-based protocol for updating the global load information. In Section III, we conduct a probabilistic analysis of the time to consensus using the proposed protocol. We present the simulation results on the average completion time of a workload in Section IV. Our conclusions and suggestions for future work are presented in Section V.

## II. Problem statement

Consider a DCS consisting of $n$ nodes operating over a partially connected network with an arbitrary topology denoted by the undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{V} = \{1,\ldots,n\}$, and $\mathcal{E}$ is the set of bidirectional communication links between nodes. At an initial time $t_0$, a workload consisting of total $M$ individual tasks is allocated onto the computing nodes of the DCS. We assume that tasks are the smallest unit of the work to be processed and they have random length since they consist of various number of instructions. Let $Q_i(t_k)$ represent the random number of the tasks in the queue of node $i$ at time $t_k$. At time $t_0$ we have $M = \sum_1^n Q_i(t_0)$.

Nodes process tasks and estimate the dynamic global load information vectors as follows. We assume that the execution time of tasks queued at the $i$th node are independently and identically distributed (i.i.d.) following an exponential distribution with a rate $\lambda_i$, i.e., the average execution time per task is $1/\lambda_i$. This assumption is commonly adopted in computing literature [17], [18]. Between every two consecutive discrete time instants, $(t_{k-1}, t_k)$ and $k = 1, 2, \ldots$, each node, the $i$th node say, estimates the dynamic global load information vector, $\hat{\mathbf{Q}}^i(t_k) = [\hat{Q}_1^i(t_k), \ldots, \hat{Q}_j^i(t_k), \ldots, \hat{Q}_n^i(t_k)]$, where $\hat{Q}_j^i(t_k)$ denotes the load of node $j$ at time $t_k$ estimated by node $i$. The estimation protocol is discussed in Section II-A.

At every $t_k$, single-hop communications are allowed in the network. Here, we assume a fixed time interval between successive communications, $\Delta t = t_k - t_{k-1}$ for all $k$. At $t_k$, each node $i$ shares its estimate, $\hat{\mathbf{Q}}^i(t_{k-1})$, with its neighbors, $\mathcal{N}(i)$, defined to be the set of nodes that are one hop from $i$. Besides the dynamic global load information vector, at time $t_0$ each node $i$ broadcasts its processing rate, $\lambda_i$, to all the nodes in the network. However, the broadcast packets are also assumed to travel one communication hop during every $\Delta t$, and nodes forward the broadcast packets they received at time $t_{k-1}$ to their neighbors at time $t_k$. By convention, if node $i$ has not received the processing rate information about another node $j$ at any $t_k$, then $\hat{Q}_j^i(t_k)$ is assumed to be zero. Therefore, at time $t_0$, $\hat{\mathbf{Q}}^i(t_0)$ has only one nonzero element since $\hat{Q}_i^i(t_0) = Q_i(t_0)$ and $\hat{Q}_j^i(t_0) = 0$ for all $j \in \mathcal{V}\backslash i$, where $\mathcal{V}\backslash i$ represents the set of nodes in $\mathcal{V}$ except node $i$.

In this paper, we assume that the processing rates, $\lambda_i$, for all nodes are known and fixed, and the delays in every one-hop communications are negligible compared to $\Delta t$. These assumptions are realistic, for example, in a dedicated computing cluster environment where the type and speed of processors are known *a priori*, processors are geographically close to each other, and the communication channel is a dedicated, high-speed local-area network. We also assume that there are no external incoming workloads after time $t_0$ until the present workload is executed.

### A. Trust-weight-based load-estimation protocol for DLB

In this subsection, we propose a consensus-based estimation protocol called trust-weight protocol. We first explain the concept of trust weight. In a partially connected network, the distance, $d_j(i)$, from node $j$ to node $i$ is defined as the number of hops between the two nodes when choosing the shortest path connecting them. Considering two nodes $i$ and $j$ that are not directly connected, node $i$ estimates the dynamic load of node $j$ relying on its neighbors' estimates of node $j$. As information flows over the network, it becomes outdated and the estimates become less accurate. Therefore, it is intuitive that node $i$ possesses a more reliable estimate of node $j$'s dynamic load if $d_j(i)$ is small. We introduce the *trust weight* of node $i$ with respect to node $j$, $r_j(i)$, as a positive integer providing an indication on the reliability of the dynamic load information that node $i$ has about the node $j$. For each node $j$, there is an associated positive integer

$$R_j \triangleq \max_i \{d_j(i)\}, \tag{1}$$

according to its location in the network. (Note that the diameter of the network, $D$, is given by $\max_j R_j$.) We next define the trust-weight value, $r_j(i)$, as

$$r_j(i) = R_j - d_j(i). \tag{2}$$

If $r_j(i) > r_j(\ell)$, then we say node $i$ has more reliable dynamic load information about node $j$ compared with node $\ell$. The possible values for $r_j(i)$ are $0, 1, \ldots, R_j$, and only node $j$ has the largest trust-weight value $R_j$ since $d_j(j) = 0$. Nodes can be classified into different equivalence classes, $\mathcal{C}_j(r)$, based on their trust-weight values with respect to a node $j$, where $\mathcal{C}_j(r)$ is the set of nodes that have the same trust-weight value $r$, i.e., $\mathcal{C}_j(r) \triangleq \{i : \ r_j(i) = r\}$ for $r = 0, 1, \ldots, R_j$.

In order to lead nodes to reach consensus about the current global load, our idea is that node $i$ estimates the load of node $j$ using information exclusively from nodes with high trust-weight values. The trust-weight protocol forming $\hat{\mathbf{Q}}^i(t_k)$ is described below.

*Trust-weight protocol*: In this estimation, we use the fact that node $i$ always has the exact information about itself at all $k$, that is,

$$\hat{Q}_i^i(t_k) = Q_i(t_k). \tag{3}$$

For other nodes $j \in \mathcal{V}\backslash i$, we form the estimate for node $j$ at $t_k$, $k = 1, 2, \ldots$, as a weighted-average of the estimated load of node $j$ by the neighbors of node $i$ with larger trust-weight values than node $i$'s trust-weight value, then minus the average number of executed tasks by node $j$ during $\Delta t$, i.e.,

$$\hat{Q}_j^i(t_k) = \left\lfloor \frac{\sum_{\ell \in \mathcal{N}_j(i)} r_j(\ell) \hat{Q}_j^\ell(t_{k-1})}{\sum_{\ell \in \mathcal{N}_j(i)} r_j(\ell)} - \lfloor \lambda_j \Delta t \rfloor \right\rfloor_+, \tag{4}$$

where $\mathcal{N}_j(i)$ defines a set of nodes $\{\ell \in \mathcal{N}(i) : r_j(\ell) > r_j(i)\}$ and $\lfloor x \rfloor$ and $\lfloor x \rfloor_+$ are the greatest integer and greatest non-negative integer smaller than or equal to $x$, respectively.

### B. Important properties of trust-weight protocol

Here, we state three key properties of the trust-weight values and estimates of the dynamic load of a node $j$. In the analysis that follows in this subsection and Section III, we assume that $Q_j(t_k) > 0$ for all $k$. This is a reasonable assumption in DCSs processing large workloads.

*Property A*: *Neighbor trust-weight values.* For $i \in \mathcal{V}\backslash j$, $\mathcal{N}_j(i) \neq \emptyset$ and $|r_j(i) - r_j(\ell)| \leq 1$ for all $\ell \in \mathcal{N}(i)$.

What the above property says is that to estimate the dynamic load of a node $j$, every distinct node $i$ has at least one neighbor with higher trust-weight value than itself. Moreover, the difference of the trust-weight values between node $i$ and its neighbors is less than or equal to 1.

*Proof* Consider a representative (specific) shortest path between nodes $i$ and $j$. Assume that node $\ell$ is on the specific shortest path and directly connected to node $i$, i.e., $\ell \in \mathcal{N}(i)$. Note that the route between node $\ell$ and $j$ is also a shortest path. Therefore, $r_j(\ell) = r_j(i) + 1 > r_j(i)$ and $\mathcal{N}_j(i) \neq \emptyset$.

Without loss of generality, assume that $d_j(i) < d_j(\ell)$ and $r_j(\ell) = r_j(i) - h$, where $h$ is an integer larger than 1. Since $\ell \in \mathcal{N}(i)$, according to the definition of the trust-weight value we have $r_j(\ell) = r_j(i) - 1$ . Hence, $r_j(\ell) < r_j(i) - h = r_j(\ell)$, a contradiction, which proves the latter part of Property A. □

*Property B*: *Same-class consensus*. For any value, $r$, of trust weight and for any $k$, $\hat{Q}_j^i(t_k) = \hat{Q}_j^\ell(t_k)$ if $i, \ell \in \mathcal{C}_j(n)$.

This property says that nodes in the same class always stay in agreement with the estimated load of a node $j$.

*Proof* From Property A, $\mathcal{N}_j(i)$ can be expressed as $\mathcal{N}_j(i) = \{\ell \in \mathcal{N}(i) : r_j(\ell) = r_j(i) + 1\}$. Now consider a node $i \in \mathcal{C}_j(R_j - 1)$, $\hat{Q}_j^i(t_k) = \frac{R_j \hat{Q}_j^j(t_{k-1})}{R_j} - \lfloor \lambda_j \Delta t \rfloor = Q_j(t_{k-1}) - \lfloor \lambda_j \Delta t \rfloor$. It is clear that $\hat{Q}_j^i(t_k)$ has the same value for all nodes $i \in \mathcal{C}_j(R_j - 1)$ and all $k$. Let $\hat{Q}_j^{\mathcal{C}_j(R_j-1)}(t_k)$ denote $\hat{Q}_j^i(t_k)$ if $i \in \mathcal{C}_j(R_j - 1)$.

Next, consider a node $i \in \mathcal{C}_j(R_j - 2)$ and note that $\hat{Q}_j^i(t_k) = \frac{\sum_{\ell \in \mathcal{N}_j(i)} r_j(\ell) \hat{Q}_j^\ell(t_{k-1})}{\sum_{\ell \in \mathcal{N}_j(i)} r_j(\ell)} - \lfloor \lambda_j \Delta t \rfloor = \hat{Q}_j^{\mathcal{C}j(R_j-1)}(t_{k-1}) - \lfloor \lambda_j \Delta t \rfloor$. Therefore, the estimates by all nodes $i \in \mathcal{C}_j(R_j - 2)$ also have the same value for all $k$, which could be denoted as $\hat{Q}_j^{\mathcal{C}_j(R_j-2)}(t_k)$. Similarly, we can derive $\hat{Q}_j^i(t_k)$ for nodes $i \in \mathcal{C}_j(r)$, where $r = R_j - 3, ..., 0$. □

*Property C*: *Boundedness of the estimator's bias*. The bias of the estimate $\hat{Q}_j^i(t_k)$ is bounded by $d_j(i)$ for all $k > d_j(i)$, namely, $\left|\mathsf{E}[\hat{Q}_j^i(t_{k+d_j(i)})] - \mathsf{E}[Q_j(t_{k+d_j(i)})]\right| \leq d_j(i)$.

*Proof* According to (4) and the two properties (A and B) stated above, if we condition on $Q_j(t_k) = m$ then $\hat{Q}_j^i(t_{k+d_j(i)})$ is deterministic and it equals to $m - d_j(i)\lfloor \lambda_j \Delta t \rfloor$. Since the number of tasks executed by node $j$ during one $\Delta t$ follows a Poisson distribution with rate $\lambda_j \Delta t$, $Q_j(t_{k+d_j(i)}) = m - \text{Pois}\big(d_j(i)\lambda_j \Delta t\big)$, where $\text{Pois}(x)$ is a Poisson random variable with mean $x$. Therefore,

$$\begin{aligned}
&\mathsf{E}[\hat{Q}_j^i(t_{k+d_j(i)})|Q_j(t_k)] \\
&\quad = \quad Q_j(t_k) - d_j(i)\lfloor \lambda_j \Delta t \rfloor \\
&\quad = \quad \mathsf{E}[Q_j(t_{k+d_j(i)})|Q_j(t_k)] + d_j(i)\big(\lambda_j \Delta t - \lfloor \lambda_j \Delta t \rfloor\big) \\
&\quad \leq \quad \mathsf{E}[Q_j(t_{k+d_j(i)})|Q_j(t_k)] + d_j(i).
\end{aligned}$$

By taking expectations of both sides we obtain

$$\left|\mathsf{E}[\hat{Q}_j^i(t_{k+d_j(i)})] - \mathsf{E}[Q_j(t_{k+d_j(i)})]\right| \leq d_j(i). \ \square$$

### C. Estimation error

Due to the uncertainty in the estimation caused by the randomness in the processing time of tasks at each node, there would be a concern that, in general, the mean of estimation error (over all nodes) may diverge in time. We will show that the mean and the variance of the total estimation error using the trust-weight protocol are both bounded. It is shown in the proof of Property C that for any $k \geq R_j$, the error in the estimate of node $i$ of the load of a node $j$ at time $t_k$ is $e_j^i(t_k) \triangleq \left|\hat{Q}_j^i(t_k) - Q_j(t_k)\right| = d_j(i)\left|\text{Pois}\big(\lambda_j \Delta t\big) - \lambda_j \Delta t\right|$. The mean of the above error is upper bounded by $d_j(i)$ and its variance is $d_j(i)\lambda_j \Delta t$, both of which are independent of $t_k$. Thus, the mean of the total estimation error of the trust-weight protocol is bounded by $n(n-1)D$ and its variance is bounded by $n(n-1)D\Delta t \max_{j \in \mathcal{V}} \lambda_j$.

To demonstrate the above features of the estimation error's mean, consider the following partially connected network with diameter $D = 4$ and whose topology is represented in Fig. 1. The average task execution times of the nodes are assumed to be $\lambda_1^{-1} = 2$ s, $\lambda_2^{-1} = 2.5$ s, $\lambda_3^{-1} = 1.5$ s, $\lambda_4^{-1} = 1$ s, $\lambda_5^{-1} = 1$ s, $\lambda_6^{-1} = 3.5$ s, $\lambda_7^{-1} = 3$ s, and $\lambda_8^{-1} = 2.5$ s.

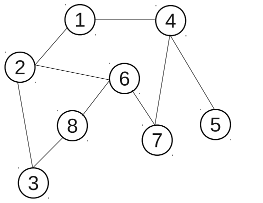


Fig. 1: Topology of the partially connected network for the examples considered.

Figure 2(a) represents the MC estimate of the mean of the total estimation error as a function of time for five different values of $\Delta t$, 2 s, 4 s, 8 s, 16 s and 32 s. In this simulation we have assumed that each node is initially assigned 100 tasks. Results for each $\Delta t$ value are averaged over 1000 independent trials. Note that for all the cases, the average total estimation error sharply drops at the beginning to a local minimum at the discrete time instant corresponding to one diameter of the network, i.e., at $k = D$, which is marked by a larger diamond. The reason is that $t_D$ is the minimum time for all nodes to spread their load information in a partially connected network. When $k \geq D$, the average total estimation error saturates at around 100 seconds, which is the average time for the fastest node to run out of its initial tasks. In addition, as shown in Fig. 2(a), the average total estimation error is decreasing to zero when nodes become idle.

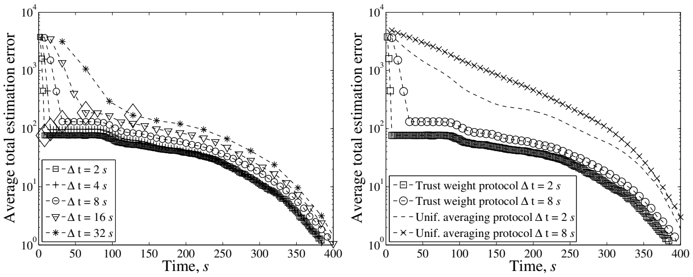
To appreciate the benefit of the trust-weight protocol over a uniformly averaging protocol, we compare the average total estimation errors under the two protocols using MC simulation, as shown in Fig. 2(b). The uniformly averaging protocol takes the average of the estimates at each node from its neighbors without assigning trust-weight values. The results show that the average total estimation error is always smaller when using the trust-weight protocol. Also, the local minimum value of the mean total error in the trust-weight protocol case is much smaller than that for the uniformly averaging protocol. Additionally, it is clear from Fig. 2(b) that the effect of $\Delta t$ on the average total estimation error is much smaller when using the trust-weight protocol compared with the uniformly averaging protocol. This result shows the efficiency of the trust-weight protocol in the amount of communication-exchanges required to reach a satisfactory error performance.

## III. Probabilistic analysis of the time to consensus

Developed an efficient estimation protocol, the next challenge is to determine the best time to execute a DLB action.

### A. Sufficient and necessary conditions for achieving consensus estimation for an arbitrary target node

We first study the probability of reaching consensus at time $k$, denoted by $P_C(k)$. The consensus instant is the time when nodes agree on their estimated global load information vector. If there is a particular time $k_0$ in which $P_C(k_0)$ is maximized,



(a) (b)

Fig. 2: (a) Average total estimation error using the trust-weight protocol for five different values of $\Delta t$; (b) Comparison of the average total estimation error under the trust-weight protocol and the uniformly averaging protocol.

then $t_{k_0}$ would be the best instant to execute a DLB since it will be optimal in terms of computing resource utilization.

To characterize the $P_C(k)$, we first state and prove sufficient and necessary conditions in order to have a consensus estimation by all the nodes on the load of any target node $j$. We emphasize again that the load of the node $j$ is always assumed to be positive at all time for analysis purpose.

**Lemma 1**. All nodes have consensus on their estimates of the load of a target node $j$ at time $k$ if and only if node $j$ executes $\lfloor \lambda_j \Delta t \rfloor$ tasks during every time interval $\Delta t$ for $R_j$ consecutive times from time $k - R_j$ to $k$, where $k \geq R_j$.

*Proof Sufficiency* — After exchange of information at time $k - R_j$, the estimates at the next time $k - R_j + 1$ are, $\hat{Q}_j^{\mathcal{C}_j(R_j)}(k-R_j+1) = Q_j(k-R_j+1) = Q_j(k-R_j) - \lfloor \lambda_j \Delta t \rfloor$, and $\hat{Q}_j^{\mathcal{C}_j(R_j-1)}(k - R_j + 1) = \hat{Q}_j^{\mathcal{C}_j(R_j)}(k - R_j) - \lfloor \lambda_j \Delta t \rfloor = Q_j(k-R_j) - \lfloor \lambda_j \Delta t \rfloor$. Similarly after exchange of information at time $k - R_j + 1$, the estimates at the next time are $\hat{Q}_j^{\mathcal{C}_j(R_j)}(k - R_j + 2) = Q_j(k - R_j + 2) = Q_j(k - R_j + 1) - \lfloor \lambda_j \Delta t \rfloor = Q_j(k - R_j) - 2\lfloor \lambda_j \Delta t \rfloor$, $\hat{Q}_j^{\mathcal{C}_j(R_j-1)}(k - R_j + 2) = \hat{Q}_j^{j(R_j)}(k - R_j + 1) - \lfloor \lambda_j \Delta t \rfloor = Q_j(k - R_j) - 2\lfloor \lambda_j \Delta t \rfloor$, and $\hat{Q}_j^{\mathcal{C}_j(R_j-2)}(k - R_j + 2) = \hat{Q}_j^{\mathcal{C}_j(R_j-1)}(k - R_j + 1) - \lfloor \lambda_j \Delta t \rfloor = Q_j(k - R_j) - 2\lfloor \lambda_j \Delta t \rfloor$. Continuing a similar analysis, after the exchange of information at time $k - 1$, the estimates are $\hat{Q}_j^{\mathcal{C}_j(R_j)}(k) = \hat{Q}_j^{\mathcal{C}_j(R_j-1)}(k) = \cdots = \hat{Q}_j^{\mathcal{C}_j(0)}(k) = Q_j(k - R_j) - R_j\lfloor \lambda_j \Delta t \rfloor$. This means that all nodes agree on their estimates of node $j$'s load.

*Necessity* — Assume that $\hat{Q}_j^{\mathcal{C}_j(0)}(k) = \hat{Q}_j^{\mathcal{C}_j(1)}(k) = \cdots = \hat{Q}_j^{\mathcal{C}_j(R_j)}(k) = Q_j(k)$. Then the estimates for all the classes of trust-weight values, $i = 1, \ldots, R_j$, at time $k-1$, can be written as $\hat{Q}_j^{\mathcal{C}_j(i)}(k-1) = \hat{Q}_j^{\mathcal{C}_j(i-1)}(k) + \lfloor \lambda_j \Delta t \rfloor = Q_j(k) + \lfloor \lambda_j \Delta t \rfloor$. Therefore, $Q_j(k-1) = \hat{Q}_j^{\mathcal{C}_j(0)}(k-1) = \hat{Q}_j^{\mathcal{C}_j(1)}(k-1) = \cdots = \hat{Q}_j^{\mathcal{C}_j(R_j)}(k-1) = Q_j(k) + \lfloor \lambda_j \Delta t \rfloor$. Following the same analysis and after the exchange of information at $k-2, k-3, \ldots, k - R_j$, it is trivially to see that $Q_j(k-n) = Q_j(k) + n\lfloor \lambda_j \Delta t \rfloor$, for $n = 1, 2, \ldots, R_j$. □

### B. Probability of reaching consensus at any given instant

In order to characterize $P_C(k)$, we define the event $A^j$ as the event that node $j$ executes $\lfloor \lambda_j \Delta t \rfloor$ tasks during $\Delta t$. Since the execution times of tasks queued at node $j$ are i.i.d. and exponentially distributed, $\mathsf{P}(A^j)$, denoted as $p^{(j)}$, follows a Poisson law for all discrete time $k$,

$$p^{(j)} = \frac{e^{-\lambda_j \Delta t}(\lambda_j \Delta t)^{\lfloor \lambda_j \Delta t \rfloor}}{(\lfloor \lambda_j \Delta t \rfloor)!}. \tag{5}$$

Based on the trust-weight protocol, at least $R_j$ hops are necessary for perceiving the information on node $j$ by every node in the network. Therefore, the probability of consensus on the load of node $j$, $P_C^{(j)}(t_k)$, when $k < R_j$, is zero. On the other hand, when $k \geq R_j$, in order for nodes to reach a consensual estimate on node $j$, event $A$ should emerge $R_j$ consecutive times according to Lemma 1. Since occurrence of event $A$ is independent in each time interval, we can write

$$P_C^{(j)}(k) = \left(p^{(j)}\right)^{R_j}. \tag{6}$$

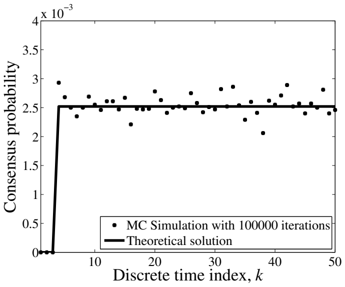


Fig. 3: Probability of consensus at different discrete time instants.

Figure 3 shows the MC simulation results which confirms the previous statement. We choose node 3 as the target node in the partially connected network shown in Fig. 1, where $R_3 = 4$ and $\Delta t$ is 2 s. The MC simulation results are averaged by 100000 independent iterations for each $k$.

Note that the procedure for estimating the load of a target node $j$ by all nodes is independent for different target nodes. Hence, the consensus probability at time $k$ for the global load is

$$P_C(k) = \prod_{j=1}^{n} P_C^{(j)}(k). \tag{7}$$

It is clear that $P_C^{(j)}(k)$ is zero when $k < R_j$ and a constant $\left(p^{(j)}\right)^{\sum_{j=1}^{n} R_j}$ when $k \geq R_j$. The above result indicates that DLB performance could not be improved, on average, for time $k \geq R_j$ as long as none of the nodes is idle.

### C. PDF of the time to first consensus

The best instant for DLB cannot be clearly identified from the consensus probability characterized in the previous subsection. In turn, here, we derive and examine the PDF of the time to *first consensus*, denoted as $\tilde{P}_C(k)$. First-time consensus is when all the nodes agree upon their estimates of the dynamic global load for the first time. Choosing such instant as the time to execute a DLB action presumably benefits the efficiency of the DLB as it allows the use of consensual estimates of the loads in the LB action, while keeping the required communications to a minimum.

We firstly characterize $\tilde{P}_C^{(j)}(k)$, which is the PDF of the time when all the nodes agree upon their estimates of a target node $j$ for the first time. We define a discrete random variable $T_n^{(j)}$ indicating the first time that node $j$ executes $\lfloor \lambda_j \Delta t \rfloor$ tasks during $\Delta t$ for $n$ consecutive steps. By recalling the sufficient and necessary conditions of achieving a consensual estimate of node $j$, we can cast $\tilde{P}_C^{(j)}(k)$ as $\mathsf{P}\{T_{R_j}^{(j)} = k\}$.

**Theorem 1**. The probability $\tilde{P}_C^{(j)}(k)$ in a partially connected network can be analytically characterized by the following recursive equations and initial conditions. When $1 \leq k < R_j$,

$$\tilde{P}_C^{(j)}(k) = 0, \tag{8}$$

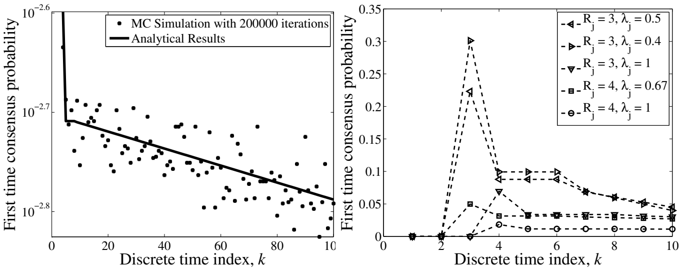
and for $k > R_j$,

$$\tilde{P}_C^{(j)}(k) = p^{(j)}\mathsf{P}\{T_{R_j-1}^{(j)} = k-1\} + \big(1 - p^{(j)}\big) \times \sum_{k'=1}^{k-2} \mathsf{P}\{T_{R_j-1}^{(j)} = k'\}\mathsf{P}\{T_{R_j}^{(j)} = k-(k'+1)\}. \tag{9}$$

*Proof* When $1 \leq k < R_j$, there exist a node, say $i$, so far from node $j$ that it has not yet received the processing rate information of node $j$; this implies that $\hat{Q}_j^i(t_k) = 0$. However, it is assumed that $Q_j(t_k) \neq 0$ for all $k$; therefore, $\tilde{P}_C^{(j)}(k) = 0$.

Next, when $k > R_j$, there are two possibilities for the event $\{T_{R_j}^{(j)} = k\}$ to occur. The first is that the event $\{T_{R_j-1}^{(j)} = k-1\}$ occurs and then the event $A^j$ occurs at time $k$. The second possibility is that the event $\{T_{R_j}^{(j)} \neq k'+1\}$ occurs following the occurrence of the event $\{T_{R_j-1}^{(j)} = k'\}$, where $k' = 1, 2, \ldots, k-2$, and then in the remaining time, i.e., from $k'+1$ to $k$, the event $A^j$ occurs for $R_j$ consecutive times firstly at time $k$ implies the event $\{T_{R_j}^{(j)} = k\}$. By conditioning on the event $\{T_{R_j-1}^{(j)} = k'\}$, we can derive

$$\begin{aligned}
\tilde{P}_C^{(j)}(k) = \;& \mathsf{P}\{T_{R_j}^{(j)} = k\} = \mathsf{P}\{T_{R_j-1}^{(j)} = k-1, T_{R_j}^{(j)} = k\} \\
& + \textstyle\sum_{k'=1}^{k-2} \mathsf{P}\{T_{R_j-1}^{(j)} = k', T_{R_j}^{(j)} \neq k'+1\} \\
& \times \mathsf{P}\{T_{R_j}^{(j)} = k-(k'+1)\} \\
= \;& \mathsf{P}\{T_{R_j}^{(j)} = k | T_{R_j-1}^{(j)} = k-1\}\mathsf{P}\{T_{R_j-1}^{(j)} = k-1\} \\
& + \textstyle\sum_{k'=1}^{k-2} \mathsf{P}\{T_{R_j}^{(j)} \neq k'+1 | T_{R_j-1}^{(j)} = k'\} \\
& \times \mathsf{P}\{T_{R_j-1}^{(j)} = k'\}\mathsf{P}\{T_{R_j}^{(j)} = k-(k'+1)\} \\
= \;& p^{(j)}\mathsf{P}\{T_{R_j-1}^{(j)} = k-1\} + \big(1-p^{(j)}\big)\textstyle\sum_{k'=1}^{k-2} \\
& \quad \mathsf{P}\{T_{R_j-1}^{(j)} = k'\}\mathsf{P}\{T_{R_j}^{(j)} = k-(k'+1)\}. \;\square
\end{aligned}$$



(a) (b)

Fig. 4: (a) PDF of first-time consensus for node 3; and (b) PDF of first-time consensus instant for different nodes.

In Fig. 4(a), we represent the PDF for the first-time consensus by solving the recursion in Theorem 1 numerically; MC simulations are also shown. It is interesting to note that the PDF for first-time consensus of node $j$ has a peak at $k = R_j$, after which it decays exponentially and asymptotically. The three parameters, $R_j$, $\lambda_j$ and $\Delta t$, determine the exponential rate of the first-time consensus PDF. Table I and II list the exponential rates by curve fitting for a node $j$ with different parameters. (An analytical proof of the exponential asymptotic behavior can also be established in a straight forward fashion but it is not included here due to space limitation.) Based on Table I and II, we see that for a node $j$ with fixed processing rate $\lambda_j$, a larger $R_j$ or $\Delta t$ leads to a larger decay rate; under a fixed communication interval $\Delta t$, a node $j$ with larger $R_j$ or $\lambda_j$ leads to a larger decay rate. Figure 4(b) compares the first-time consensus probability for nodes, 1, 2, 3, 4 and 5, under the same $\Delta t = 2$ s; however, each node has its own $R_j$ and $\lambda_j$. Note that nodes with larger $R_j$ and larger $\lambda_j$ lead to smaller first-time consensus probabilities than other nodes. Therefore, we conclude that one node dominates the first-time consensus probability for the whole system. As such, the probability $\tilde{P}_C(k)$ can be approximated as $\tilde{P}_C^{(J)}(k)$, where node $J$ is the node with $R_J = D$ and largest processing rate. According to the above analysis on the first-time consensus PDF, it is reasonable to choose the DLB instant at one diameter $D$ time instant, $t_b = t_D$.

TABLE I: Exponential decay rate of PDF for the first-time consensus under various values of $R_j$ and $\Delta t$, but same $\lambda_j$.

| | Exponential decay rate | | | | |
|---|---|---|---|---|---|
| $\Delta t$ | $R_j = 2$ | $R_j = 3$ | $R_j = 4$ | $R_j = 5$ | $R_j = 6$ |
| 0.5 | 0.7234 | 0.8926 | 0.9100 | 0.9395 | 0.9452 |
| 1.0 | 0.8638 | 0.9650 | 0.9729 | 0.9849 | 0.9869 |
| 1.5 | 0.9283 | 0.9878 | 0.9914 | 0.9960 | 0.9967 |
| 2.0 | 0.9605 | 0.9956 | 0.9972 | 0.9989 | 0.9992 |
| 2.5 | 0.9776 | 0.9984 | 0.9991 | 0.9997 | 0.9998 |

TABLE II: Exponential decay rate of PDF for the first-time consensus under various values of $R_j$ and $\lambda_j$, but same $\Delta t$.

| | Exponential decay rate | | | | |
|---|---|---|---|---|---|
| $\lambda_j$ | $R_j = 2$ | $R_j = 3$ | $R_j = 4$ | $R_j = 5$ | $R_j = 6$ |
| 0.5 | 0.8285 | 0.9311 | 0.9704 | 0.9867 | 0.9939 |
| 1.0 | 0.9100 | 0.9729 | 0.9914 | 0.9972 | 0.9991 |
| 1.5 | 0.9411 | 0.9855 | 0.9962 | 0.9990 | 0.9997 |
| 2.0 | 0.9575 | 0.9910 | 0.9980 | 0.9996 | 0.9999 |
| 2.5 | 0.9635 | 0.9928 | 0.9985 | 0.9999 | 0.9999 |

## IV. Dynamic load balancing with consensus-based load estimation

In this section, we present MC simulation results on the average completion time, a common metric for evaluating the performance of DLB, of a workload by applying the proposed protocol for different system configurations. We also compare the results with that by applying the "local multi-shot DLB policy" employed in [12]. In simulating our protocol, we adopt the DLB policy presented in [7], which is a one-shot DLB policy for decentralized DLB in a fully connected network, and extend it to the case over a partially connected network by assuming that tasks could be relayed through median node(s) between the two indirectly connected nodes without processing. The one-shot DLB policy means that the task redistribution by all nodes are executed only one time. The one-shot DLB policy is an efficient approach in DLB performance as well as bandwidth savings [10], [11]. We also adopt the method presented in [7] in order to calculate the task reallocation partitions, which specifies the redistribution decisions of the nodes on how many tasks to transfer to other nodes. We also assume that if a node $i$ has not received the processing rate information, $\lambda_j$, at the DLB instant, $t_b$, it would assume that node $j$ does not exist in the system and therefore it does not send any tasks to node $j$, even though $\hat{Q}^i_j(t_b) = 0$. We call the one-shot DLB policy used in this paper for the partially connected network along with the added assumptions, the "extended one-shot DLB" policy. We emphasize that all the MC simulation results in this section correspond to centers of 95% confidence intervals.

Next, we present the average completion time of a workload as a function of DLB instant $t_b$. In order to investigate the effect of information exchange frequency, five $\Delta t$ values, 2 s, 4 s, 8 s, 16 s and 32 s (same as in Section II-C), are considered. In our simulations, the total number of tasks in the workload is fixed to be 800 tasks. The task execution rates of nodes are also same as in Section II-C. In the results shown in Fig. 5(a), the workload is initially allocated evenly at the nodes of the topology presented in Fig. 1, i.e., 100 tasks at each node. As it can be seen in Fig. 5(a), the average completion time of the workload using the trust-weight protocol along with the extended one-shot DLB policy indicates a similar pattern for all the five $\Delta t$ values. Note that the first point of each curve (for $t_b = 0$) corresponds to the case for which nodes execute their own initial tasks without executing DLB. The larger diamond markers represent the results for which the DLB has been executed at the time $t_D$ (namely, after a "diameter time" of information exchange). The sharp decay in the beginning of the curves in Fig. 5(a) are due to the fact that more accurate dynamic load information become available to every node in the network as information is exchanged. By performing DLB after exchanging information by an amount equal to the diameter time, the average completion time becomes stable due to saturation of the estimation error shown in Section II-C. However, if the DLB is executed with a large delay, some of the nodes may become idle, which causes a waste in computing resources of the system. Therefore, the average completion time starts to increase around 100 s, which is the average time for the fastest node to become idle.

We now investigate the effect of the connectivity and the diameter of the network on the completion time. Here we modify the topology of the network presented in Fig. 1 by removing the links between node 2 and 6 as well as node 4 and 7 in order to have an extreme-case topology (all nodes in a line). Note that in the modified topology the fastest two nodes (4 and 5) are at one end of the line and the slowest two nodes (6 and 7) are at the other end of the line. We keep other settings the same as in Section II-C and present the simulation results in Fig. 5(b). Note that the results presented in Fig. 5(b) represent a similar behavior as the results shown in Fig. 5(a).

It is interesting to note that when $D\Delta t$ is less than the average time for the fastest node to execute its initial tasks, i.e., when $\Delta t = 2$ s, 4 s, 8 s, and 16 s, the minimum average completion time of the workload is approximately 200 s under both network connections. This indicates adaptability of the trust-weight protocol to various topology and communication constraints. Now, we propose what we term a *safe zone*, $[t_{min}, t_{max}]$ for $t_b$, where $t_{min} = t_D$, and $t_{max}$ is the average time for the fastest node in the network to become idle. Based on the results for these two topologies satisfactory average completion time can be achieved when $t_b$ is chosen to be in the safe zone. Note that by executing DLB with more exchanges of information after time $t_D$, the average completion time of a workload is not reduced evidently. However, if $t_D$ is greater than the average time for the fastest node in the network to become idle, as in the case $\Delta t = 32$ s, then there is no safe zone for choosing $t_b$. The results in Fig. 5(b) also show that the best DLB execution instant, emphasized by the diamond marker, is postponed due to increased diameter of the modified network, $D'$, from 4 to 7.

Next, to explore the effect of initial task distribution over the nodes, we consider the case for which all the tasks of the workload are initially allocated to the fastest node in the DCS operating over the original topology (Fig. 1). The simulation results are shown in Fig. 5(c). In this scenario, the best choice for $t_b$ is definitely after $D$ exchanges of information has taken place for all the $\Delta t$ values because it is the first time that the computing resources of all the nodes in the system can be utilized after a DLB action is executed.

Finally, we will compare the efficiency, in terms of the average completion time, of the trust-weight protocol in conjunction with the extended one-shot DLB policy with the local multi-shot DLB policy employed in [12]. In the local multi-shot policy, each node exchanges information with its neighboring nodes and then executes a DLB action only within its neighborhood after each information exchange. Figure 5(d) represents the results of the average completion time using the local multi-shot DLB policy for the same settings as in Fig. 5(a). In the local multi-shot policy, it is obvious that as the number of DLB executions increases, the average completion time gradually reduces and converges to the best achievable completion time. Based on these results, our approach outperforms the local multi-shot policy, in terms of the minimum achievable average completion time of a workload except when $\Delta t = 2$ s. This is expected as the latter does not take into account the task executions as the load estimation is ongoing. The minimum achievable average completion time of a workload when $\Delta t = 2$ s is around 188 s in Fig. 5(d), which is nearly 7% lower than the minimum average completion time for same $\Delta t$ in Fig. 5(a). However, the local multi-shot policy achieves the same performance using larger number of DLB actions than our approach (in this example, at least 50 vs. 4). Note also that transferring tasks in the network is much more bandwidth consuming than exchanging information, and therefore, our approach reduces the communication costs.

## V. Conclusions

In this paper we presented a novel consensus-based protocol for estimating the load information at nodes in a DCS setting

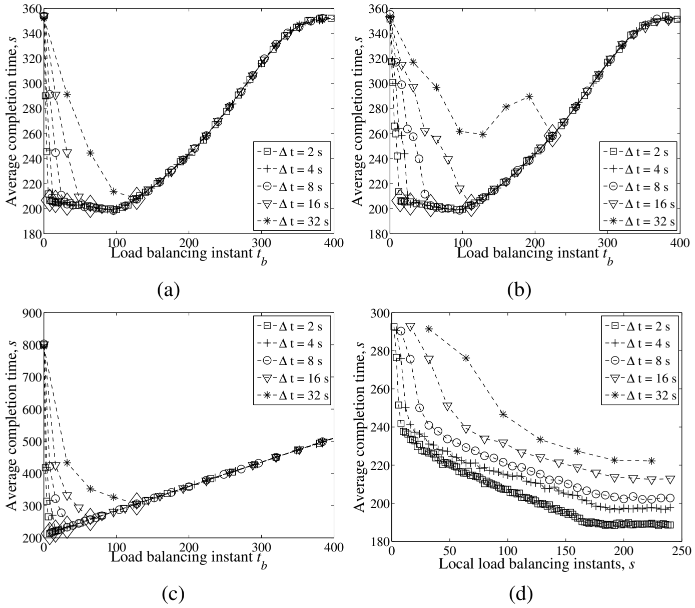


Fig. 5: Average completion time of a workload for; (a) evenly distributed tasks over the nodes of the original topology, (b) evenly distributed tasks over the nodes of the modified topology, (c) unevenly distributed tasks over the fastest node of the original topology, by applying extended one-shot DLB policy, and (d) evenly distributed tasks over the nodes of the original topology, by applying the local multi-shot DLB policy [12].

operating over a partially connected communication network where loads are both dynamic and stochastic. The necessity of forcing all the estimates to be consensual (in order to have higher effectiveness of DLB) motivated the development of the protocol. Our results indicate that the proposed protocol is of great advantage in terms of the estimation error as well as the efficiency in communication cost compared with the uniformly averaging protocol that does not utilize the trust weight. To determine the time to execute DLB, we have analytically characterized the probability of consensus at any given time as well as the PDF of the time to the first consensus; the latter suggests that the best DLB instant is at a diameter number of exchanges of information. We presented MC simulation results of the average completion time of a workload using the proposed protocol along with the extended one-shot DLB policy. The results show that our method works better than the local multi-shot DLB method when the information exchange period is long relative to the average task-processing time. In addition, our method offers an advantage in the bandwidth usage compared with the existing method. We also defined a safe zone for executing DLB over which the performance is stable if DLB instant is chosen in this zone.

Future work will consider more general distributed-computing scenarios for which nodes deal with different types of tasks at the same time. In such scenarios, the processing rates become varying, which makes the consensus-estimation problem far more challenging. In addition, in such scenarios there is the possibility that consensus may not even happen before the workload is accomplished. One possible approach for such extension would be to consider an adjustable estimation protocol while performing multiple DLBs periodically with a period governed by the diameter of the network.

## Acknowledgment

This work was supported by the Defense Threat Reduction Agency (Combating WMD Basic Research Program).

## References

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

[2] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," in *Proc. IEEE INFOCOM*, San Francisco, CA, 2003.

[3] G. Koole, P. Sparaggis, and D. Towsley, "Minimizing response times and queue lengths in systems of parallel queues," *J. Applied Probability*, vol. 36, pp. 1185–1193, 1999.

[4] A. Brandt and M. Brandt, "On a two-queue priority system with impatience and its application to a call center," *Methodology and Computing in Applied Probability*, vol. 1, pp. 191–210, 1999.

[5] C. K. H. Kameda, J. Li and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*. Springer, 1997.

[6] G. K. M. Dobber and R. van der Mei, "Dynamic load balancing experiments in a grid," *IEEE International Symposium on Cluster Computing*, vol. 2, pp. 1063–1010, 2005.

[7] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. Yang, and D. A. Bader, "Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach," *IEEE Trans. Parallel and Dist. Systems*, vol. 18, pp. 485–497, 2007.

[8] J. E. Pezoa, S. Dhakal, and M. M. Hayat, "Decentralized load balancing for improving reliability in heterogeneous distributed systems," in *Proc. ICPP 09*, Vienna, Austria, 2009.

[9] Y. S. I. F. Akyildiz, W. Su and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[10] M. M. Hayat, S. Dhakal, C. T. Abdallah, J. D. Birdwell, and J. Chiasson, *Dynamic time delay models for load balancing. Part II: Stochastic analysis of the effect of delay uncertainty*, ser. LNCS v. 38. Springer-Verlag, 2004, ch. Advances in Time Delay Systems, pp. 355–368.

[11] S. Dhakal, B. Paskaleva, M. M. Hayat, E. Schamiloglu, and C. T. Abdallah, "Dynamical discrete-time load balancing in distributed systems in the presence of time delays," in *Proc. IEEE CDC 03*, Maui, HI, 2003.

[12] A. Gonzlez-Ruiz and Y. Mostofi, "Distributed load balancing over directed network topologies," in *Proc. ACC 09*, St. Louis, Missouri, USA, 2009.

[13] T. Hageru, "Allocating independent tasks to parallel processors: an experimental study," *J. Parallel and Distributed Computing*, vol. 47, no. 2, pp. 185–197, 1997.

[14] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 401–420, 2004.

[15] W. Ren and R. W. Beard, "Stability of continuous-time distributed consensus algorithms," in *Proc. ACC 05*, Portland, OR, MA, 2005, pp. 1859–1864.

[16] F. Xiao and L. Wang, "Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays," *IEEE Trans. Automat. Contr.*, vol. 53, no. 8, pp. 1804–1816, 2008.

[17] Y.-S. Dai and G. Levitin, "Optimal resource allocation for maximizing performance and reliability in tree-structured grid services," *IEEE Trans. Reliability*, vol. 56, pp. 444–453, 2007.

[18] J. E. Pezoa, S. Dhakal, and M. M. Hayat, "Maximizing service reliability in distributed computing systems with random failures: Theory and implementation," *IEEE Trans. Parallel and Dist. Systems*, vol. 21, no. 10, pp. 1531–1544, 2010.