Distributed Load Balancing over Directed Network Topologies

Alejandro Gonzalez-Ruiz and Yasamin Mostofi

Cooperative Network Lab
Department of Electrical and Computer Engineering
University of New Mexico, Albuquerque, New Mexico 87131, USA
Email: {agon,ymostofi}@ece.unm.edu

***Abstract*— In this paper we consider the problem of distributed load balancing over a directed graph that is not fully connected. We study the impact of network topology on the stability and balance of distributed computing. We furthermore propose Informed Load Balancing (I-LB), an approach in which the nodes first reach an agreement over the balanced state, through using a consensus-seeking protocol, before proceeding to redistribute their tasks. We compare the performance of I-LB with that of the Original Load Balancing (O-LB) approach in terms of speed of convergence and bandwidth usage. We prove that the O-LB approach can guarantee convergence to a balanced state as long as the underlying graph is strongly connected while I-LB may not converge. However, I-LB can increase the speed of convergence and/or reduce the bandwidth usage especially for low-connectivity graphs.**

I. INTRODUCTION

Everyday there is an increasing demand for high performance computing. High speed networks have become more common, allowing for interconnecting various geographically distributed Computational Elements (CEs). This has enabled cooperative operation among the nodes, which can result in obtaining an overall better performance than the one achieved by a single CE. Grid-computing is an example of a system that can benefit from cooperative computing [1].

Distributing the total computational load across available processors is referred to as load balancing in the literature. The goal of a load balancing policy is to ensure an optimal use of the available resources so that each CE ends up with a "fair" share of the overall job, thus allowing the overall load to be completed as fast as possible. Load balancing can be implemented in a centralized [2] or a distributed manner [1]. In this paper, we are interested in distributed load balancing, where each node polls other processors in its neighborhood and uses this information to decide on a load transfer.

There has been extensive research in the development of effective load balancing policies. In [3]–[6] various distributed load balancing schemes that use concepts such as queuing- and regeneration-theory have been proposed and analyzed. A common factor in these approaches is that the amount of load to be exchanged is based on the weighted averages of the loads of the neighboring nodes. In [7]–[9] the proposed policies use concepts such as graph coloring and gossip algorithms. These algorithms do pairwise balancing of loads, i.e. two adjacent nodes are randomly chosen to exchange their loads at a given time step.

In the load balancing literature, a common assumption is to take the topology of the underlying graph to be undirected. However, most realistic wireless networks will have asymmetric uplink and downlink, resulting in directed graphs. This is due to the fact that wireless transmissions in uplink and downlink typically occur in two different and uncorrelated pieces of bandwidth [10], resulting in different and uncorrelated link qualities. Furthermore, different nodes may have different transmission power resulting in different reception qualities (even if the links were the same) and as a result directed graphs. Another scenario where the network topology is directed occurs when there are firewalls between certain nodes that allow incoming connections but block outgoing ones. Therefore, in this paper we mainly focus on directed graphs. We study the impact of network topology on the stability and balance of distributed computing. For the case of a one-time arrival of the loads, we show that load balancing over a strongly connected graph reaches a balanced state independent of the initial load distribution while having a spanning tree is not a sufficient condition for reaching a balanced state. We furthermore propose Informed Load Balancing (I-LB), an approach in which the nodes first reach an agreement over the global balanced state before starting the actual load balancing process. We show that while I-LB lacks asymptotic performance guarantees, it has the potential of increasing the speed of convergence and/or reduce the bandwidth usage especially for low-connectivity graphs.

The paper is organized as follows. Section II presents our system model. Section III explores the impact of graph topology on distributed computing. Section IV shows the asymptotic convergence of the distributed load balancing algorithm to the balanced state. Section V introduces I-LB, an alternative load balancing approach, and explores the underling tradeoffs between I-LB and the original load balancing approach. We conclude in Section VI.

II. PROBLEM FORMULATION

Consider a distributed computing system of n nodes connected over a wireless network with a directed topology that can be described by a graph $G = (N, E)$, where N is the set of nodes $N = \{1, ..., n\}$ and E is the set of edges

This work was supported in part by the Defense Threat Reduction Agency through grant HDTRA1-07-1-0036.

connecting the processors. Let $x_i(k)$ represent the load on processor i at time $k \geq 0$. The goal is to spread the subtasks among all n processors as evenly as possible such that no node is overburdened while other nodes are idle. At every time step the nodes assess how overburdened they are and exchange loads in order to increase the overall computational efficiency.

For the purpose of mathematical analysis, we take the load of each processor to be infinitely divisible such that $x_i(k)$ can be described by a continuous nonnegative real number. We furthermore assume that: (i) tasks are independent so that they can be executed by any processor; (ii) processors are homogeneous, i.e. they have the same processing speed and (iii) the link-level delay in the exchange of loads between nodes is negligible. This last assumption is justified when the product of the queue length and the service time (execution time per task) is considerably greater than the time it takes to transfer a group of tasks from one node to the other. We also assume that processors are synchronized, i.e. all the processors perform load balancing at the same time. As indicated in [11], assuming synchrony yields similar results, in terms of the final load distribution, to the ones obtained by working with asynchronous algorithms.

A. Link Symmetry

Consider the wireless link from node j to node i and that from node i to node j. In wireless communication, these two transmissions typically occur at two different and uncorrelated pieces of bandwidth [10]. As a result, the link quality in the transmission from node i to node j can be considerably different from that of the transmission from node j to i, resulting in an asymmetry. Therefore, in this paper we take G to be a directed graph. Then a neighborhood set $\mathcal{N}_i$ of node i is given by all the nodes $j \in N$ such that there is a directed link from i to j. In other words, $\mathcal{N}_i = \{j \in N | (i,j) \in E\}$ with $|\mathcal{N}_i|$ representing its cardinality.

The nodes furthermore exchange their queue lengths with their neighbors continuously in order to assess how overloaded their local neighborhood is. Since this information has a considerably lower volume than the actual loads, it can be transmitted over a separate lower bandwidth channel or with higher transmission power. As a result, it can experience better reception quality and higher probability of symmetry. Therefore, in this paper we assume that the queue length information is transmitted over an undirected version of graph G. We are currently working on relaxing this assumption.

B. Description of the Distributed Load Balancing Algorithm

Let $s_{ij}(k)$ denote the load that node i sends to node j at time k. Let $J_i(k)$ and $C_i(k)$ represent the number of external tasks arriving to node i and the number of tasks serviced by node i respectively at time k. We will have the following for the dynamics of the queue length of node i:

$$x_i(k+1) = x_i(k) - \sum_{j \in \mathcal{N}_i} s_{ij}(k) + \sum_{j \in \{l | i \in \mathcal{N}_l\}} s_{ji}(k) + J_i(k) - C_i(k). \quad (1)$$

Using an approach similar to the one presented in [1] and [3], the amount of load to be transferred from node i to node j can be calculated based on the excess load of node i. Define $\text{ave}_i(k)$ as the local average of node i, i.e. the average load of node i and its neighbors:

$$\text{ave}_i(k) = \frac{1}{|\mathcal{N}_i| + 1} \left(x_i(k) + \sum_{j \in \mathcal{N}_i} x_j(k) \right). \quad (2)$$

Note that although the graph for the exchange of loads is directed, we assumed an undirected graph for the exchange of queue length information. Therefore the ith node has access to the value of $x_j(k)$ for all $j \in \mathcal{N}_i$ and can therefore calculate its local average.

The excess load of node i at time k ($L_i^{ex}(k)$) is then given by the difference between its load and local average:

$$\begin{aligned} L_i^{ex}(k) &= x_i(k) - \text{ave}_i(k) \\ &= x_i(k) - \frac{x_i(k) + \sum_{j \in \mathcal{N}_i} x_j(k)}{|\mathcal{N}_i| + 1}. \end{aligned} \quad (3)$$

This quantity represents how overloaded node i is with respect to its own neighborhood. Next, node i evaluates how overloaded its neighbors are with respect to its local average. Let $L_{j(i)}^{ex}(k)$ be the excess load of node j with respect to the neighborhood average of node i, namely:

$$L_{j(i)}^{ex}(k) = x_j(k) - \text{ave}_i(k) \quad \text{for} \quad j \in \mathcal{N}_i. \quad (4)$$

$L_{j(i)}^{ex}(k)$ represents how overloaded the jth node "appears" to the ith one, based on the partial information available at the ith node. It can be easily verified that $\sum_{j \in \{\mathcal{N}_i \cup\ i\}} L_{j(i)}^{ex}(k) = 0$, with the convention that $L_{i(i)}^{ex}(k) = L_i^{ex}(k)$. Define $p_{ij}(k)$ as the fraction of $L_i^{ex}(k)$ to be sent from node i to node j. Then we will have the following for $s_{ij}(k)$, the amount of load that the ith node sends to the jth one at the kth time step [3]:

$$s_{ij}(k) = p_{ij}(k) \left(L_i^{ex}(k) \right)^+, \quad (5)$$

where $(x)^+ = \max(x, 0)$ and:

$$p_{ij}(k) = \begin{cases} \frac{L_{j(i)}^{ex}(k)}{\sum_{l \in \mathcal{M}_i(k)} L_{l(i)}^{ex}(k)} & j \in \mathcal{M}_i(k) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

with

$$\mathcal{M}_i(k) = \{j | j \in \mathcal{N}_i,\ \text{ave}_i(k) > x_j(k)\}. \quad (7)$$

In other words, node i will send part of its excess load to node j at time k only if $x_j(k)$ is below $\text{ave}_i(k)$.

III. IMPACT OF NETWORK TOPOLOGY ON DISTRIBUTED LOAD BALANCING

In order to motivate the mathematical derivations of the next section, we first consider the impact of different network topologies on distributed load balancing through simulations. Two concepts that are frequently used throughout the paper are that of spanning trees and strongly connected graphs. A directed graph has a spanning tree if there exists a node that has a directed path to all the other nodes [12]. Furthermore, a strongly connected graph is a directed graph that has a directed path from every node to every other one.

Consider the directed graph of Fig. 1 (left), which has no underlying spanning tree. The processing speed of each node is 20 tasks per time step (i.e. $C_i(k) = 20,\ \forall i$). Let $J(k) = [J_i(k)] = [5\ 60\ 8\ 2]$, where the ith element represents the incoming rate of loads at the ith node. Note that the overall incoming load rate of the whole system is less than the overall system's processing rate ($\sum_i C_i(k) > \sum_i J_i(k)$), a necessary condition for successful load balancing.

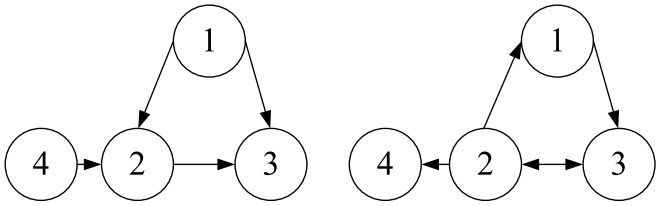


Fig. 1. An example of a directed network topology that contains (left) no spanning tree and (right) a spanning tree.

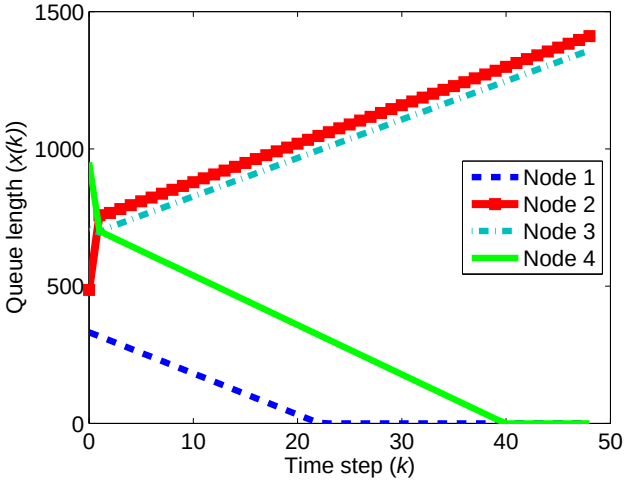


Fig. 2. Queue dynamics for load balancing over the network of Fig. 1.

Figure 2 shows the dynamics of the queues in the distributed system. As expected from the topology, the system becomes unstable, i.e. as $k \to \infty$ nodes 1 and 4 become idle with queue lengths of 0, while $x_3(k) \to \infty$. This example highlights the importance of understanding and characterizing the impact of graph topology on distributed load balancing, one of the goals of this paper.

Since a mathematical analysis of the impact of graph topology on load balancing is considerably challenging, we make a number of simplifying assumptions for the sake of mathematical derivations. We assume a one-time arrival of loads, i.e. $J_i(k) = 0$ for $k \neq 0$ and $\forall i$. We also assume that no tasks will leave the system, i.e. $C_i(k) = 0,\ \forall i$. These assumptions allow us to solely analyze the impact of the underlying graph topology on distributed load balancing. In other words, given the initial loads of the nodes, we seek to understand the conditions under which all the nodes will have a "fair" share of the overall load after a number of load balancing instants. Let L represent the total load of the network, which is given by: $L = \sum_{i=1}^{n} x_i(0)$. Since all the nodes are assumed to have the same processing speed, the optimum load balancing should result in the load of each node approaching the average of the system ϕ given by:

$$\phi = \frac{1}{n}\sum_{i=1}^{n} x_i(0) = \frac{L}{n}. \tag{8}$$

We say that the load balancing algorithm converges if $\lim_{k\to\infty} x_i(k) = d_i$ for all i, where d_i is a nonnegative constant. Similarly, we define *balanced state* as the state in which all the queue lengths converge to the same value ϕ.

Consider the graph topology that is formed by only considering the solid arrows of Fig. 3. This corresponds to a directed network of ten nodes with an underlying spanning tree. The distributed load balancing algorithm is executed according to (1)-(7) with $J_i(k) = 0$ for $k \neq 0$ and $C_i(k) = 0$. Initially, there is a total of 5000 tasks, all located at node 1. Fig. 4 shows the dynamics of the queues. It can be seen that a balanced state is not reached since the leaf nodes have no way of distributing their excess load.

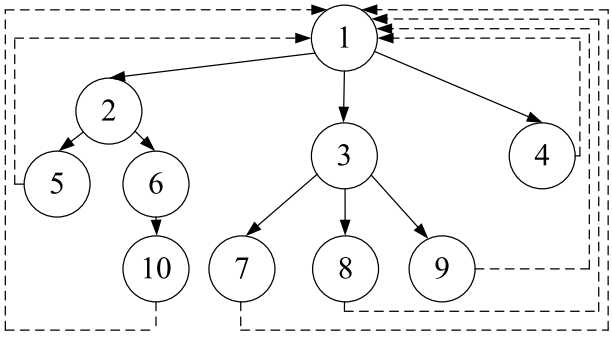


Fig. 3. An example of a network topology with an underlying spanning tree (solid arrows only) and a network topology with an underlying strongly-connected graph (dashed and solid arrows).

By adding the additional links, represented by the dashed lines in Fig. 3, a strongly connected graph is formed. Figure 5 shows the dynamics of the queues for load balancing over this graph. It can be seen that a balanced state of $\phi = 500$ is reached. In the next section, we show that having an underlying strongly connected graph is a sufficient condition for reaching a balanced state asymptotically.

The initial distribution of the loads also plays a key role in reaching the balanced state. Consider the topology shown in Fig. 1 (right), which has a spanning tree. It can be confirmed that for the initial distribution $x(0) = [332\ 486\ 707\ 951]^T$,

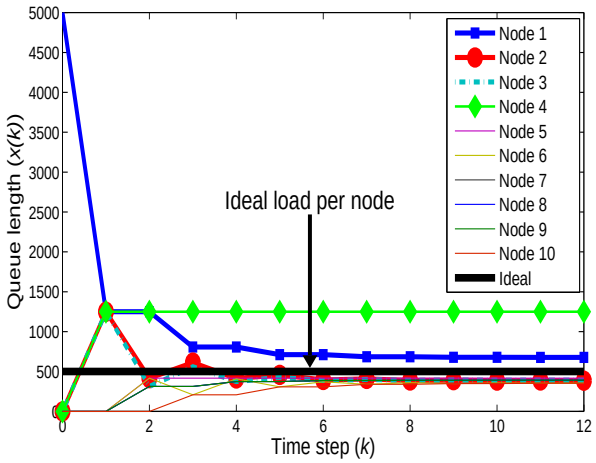


Fig. 4. Queue dynamics for distributed load balancing over the graph topology of Fig. 3 (considering only the solid arrows). The queues cannot reach a balanced state even though there is an underlying spanning tree.

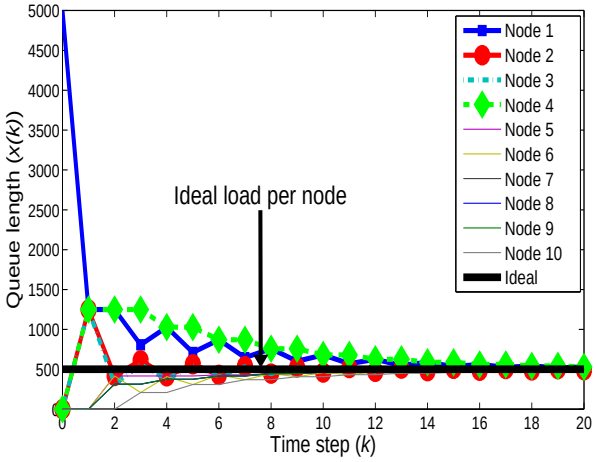


Fig. 5. Queue dynamics for distributed load balancing over the strongly connected graph of Fig. 3 (considering both solid and dashed arrows). The queues reach a balanced state.

there will be no convergence to a balanced state as the final values are $[332\ 596.5\ 596.5\ 951]^T$. Consider the same graph but with the following initial distribution $x(0) = [951\ 707\ 486\ 332]^T$. As seen from Fig. 6, the system reaches the balanced state. This example shows the impact of the initial load distribution on the convergence to the balanced state.

IV. CONVERGENCE TO A BALANCED STATE

In [13] it was shown that a distributed load balancing algorithm converges to a balanced state if it complies with a number of conditions. While those conditions were described with an undirected graph in mind, they can be easily extended to distributed load balancing over directed graphs. In this part we show, following a similar approach to [13], that distributed load balancing according to (1)-(7) converges to the balanced state. First, note that our load balancing policy satisfies the following properties:

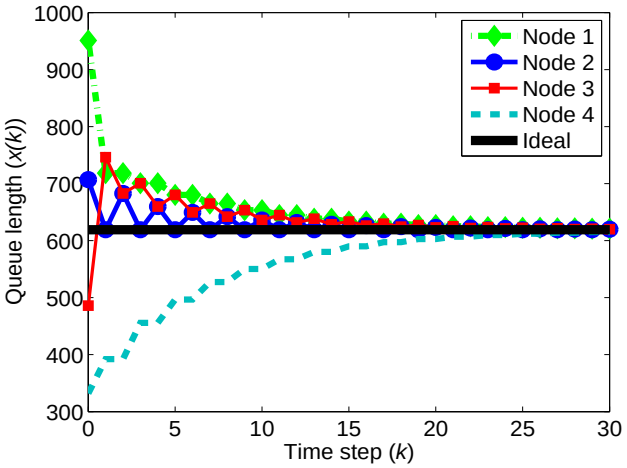


Fig. 6. Queue dynamics for distributed load balancing over the network of Fig. 1 (right), which is not strongly connected but has a spanning tree. The distributed system reaches the balanced state for the initial distribution of $x(0) = [951\ 707\ 486\ 332]^T$.

Property 1: If $x_i(k) > \text{ave}_i(k)$, there exists some $j^* \in \mathcal{N}_i$ such that $\text{ave}_i(k) > x_{j^*}(k)$ and $s_{ij^*}(k) > 0$.

Property 2: For any $j \in \mathcal{N}_i$ and any i such that $x_i(k) > x_j(k)$ and $\text{ave}_i(k) > x_j(k)$, the following can be easily confirmed:

$$x_i(k) - \sum_{l \in \mathcal{N}_i} s_{il}(k) \geq x_j(k) + s_{ij}(k). \tag{9}$$

If $x_i(k) \leq \text{ave}_i(k)$, Eq. (9) is obvious. If $x_i(k) > \text{ave}_i(k)$, the validity of this property is based on the fact that: $L_i^{ex}(k) / \sum_{l \in \mathcal{M}_i(k)} L_{l(i)}^{ex}(k) \in [-1, 0]$. Therefore,

$$\begin{aligned} & x_i(k) - L_i^{ex}(k) - x_j(k) - s_{ij}(k) \\ = \ & -L_{j(i)}^{ex}(k) - \frac{L_{j(i)}^{ex}(k)}{\sum_{l \in \mathcal{M}_i(k)} L_{l(i)}^{ex}(k)} L_i^{ex}(k) \geq 0. \end{aligned}$$

Let $m(k)$ be defined as: $m(k) \triangleq \min_i x_i(k)$.

Lemma 1: There exists some $\beta \in (0, 1)$ such that

$$x_i(k+1) \geq m(k) + \beta(x_i(k) - m(k)), \quad \forall i \in N. \tag{10}$$

Proof: We will follow an approach similar to the one in [13]. Without loss of generality, fix a processor i and a time step k. Also consider the set $\mathcal{W}_i(k) = \{j | j \in \mathcal{N}_i, x_i(k) > x_j(k)\}$ and denote its cardinality as: $|\mathcal{W}_i(k)|$. From (1) and (9), we have:

$$x_i(k+1) \geq x_j(k) + s_{ij}(k)\ \forall j \in \mathcal{W}_i(k).$$

Adding over all $j \in \mathcal{W}_i(k)$:

$$|\mathcal{W}_i(k)|\ x_i(k+1) \geq \sum_{j \in \mathcal{W}_i(k)} x_j(k) + \sum_{j \in \mathcal{W}_i(k)} s_{ij}(k). \tag{11}$$

By noting that $s_{ij}(k) = 0$ if $j \notin \mathcal{W}_i(k)$, we will have:

$$\sum_{j \in \mathcal{W}_i(k)} s_{ij}(k) = \sum_{j \in N_i} s_{ij}(k) \geq x_i(k) - x_i(k+1). \tag{12}$$

Combining (11) and (12) will then result in:

$$|\mathcal{W}_i(k)|\ x_i(k+1) \geq \sum_{j \in \mathcal{W}_i(k)} x_j(k) + x_i(k) - x_i(k+1),$$

which is equivalent to:

$$\begin{aligned} x_i(k+1) &\geq \frac{|\mathcal{W}_i(k)|}{|\mathcal{W}_i(k)|+1} m(k) + \frac{1}{|\mathcal{W}_i(k)|+1} x_i(k) \\ &= m(k) + \frac{1}{|\mathcal{W}_i(k)|+1}(x_i(k) - m(k)) \\ &\geq m(k) + \frac{1}{n}(x_i(k) - m(k)), \end{aligned}$$

which proves the inequality with $\beta = \frac{1}{n}$. ■

Consequently, we will have the following lemma:

Lemma 2: The sequence $m(k)$ is upper bounded by L and is nondecreasing. Therefore it converges.

Proof: From (10) we can easily see that $x_i(k+1) \geq m(k)$, resulting in $m(k+1) \geq m(k)$. Since $J_i(k) = 0$ for $k \neq 0$, $m(k) \leq L$. Therefore it converges. ■

In [13] the following lemma is proved which establishes a lower bound on the queue length of any node j that can be reached from node i by traversing l edges. Since our load balancing algorithm satisfies properties 1 and 2, we will have the following lemma:

Lemma 3: Consider node i. For any node j that can be reached from i by traversing l edges, and for any $k \geq k_0 + 3ln$, we have

$$x_j(k) \geq m(k_0) + (\eta \beta^{k-k_0})^l (x_i(k_0) - m(k_0)), \qquad (13)$$

where η is a nonnegative real number and β is as defined in Lemma 1.

Proof: See [13]. ■

Using the previous lemmas, we can extend the convergence proof of [13] to the following:

Theorem 1: (Convergence of the load balancing policy to the balanced state) Consider the algorithm described in Section II. If the graph G is strongly connected, then

$$\lim_{k \to \infty} x_i(k) = L/n. \qquad \forall i \in N \qquad (14)$$

Proof: Consider a strongly connected graph. Then for a given node i, every other node is at a distance of at most $(n-1)$ from i. We can apply (13) and follow the proof in [13] to conclude that the difference between the highest load and the minimum load of the system ($\max_i x_i(k) - m(k)$) has to converge to 0. From Lemma 2 we have that $m(k)$ converges to a constant c. Therefore, $\lim_{k\to\infty} x_j(k) = c$ for all j. Since $\sum_{i=1}^{n} x_i(k) = L$, we have $c = L/n$. ■

V. DISTRIBUTED LOAD BALANCING WITH A PRIORI KNOWLEDGE OF THE BALANCED STATE (I-LB)

In the distributed load balancing algorithm of the previous sections, the nodes have to constantly distribute their "perceived" extra loads in order to reach a balanced state. Since the loads can have very large sizes, this can result in a considerable use of the available bandwidth. If the nodes could first reach an agreement over the global balanced state, it can potentially reduce the overall time to reach the balanced state and the overall bandwidth usage. In this section we propose Informed Load Balancing (I-LB), a modification to the Original LB algorithm (O-LB) of the previous sections. The main idea of I-LB is to let the nodes exchange information about their queue lengths and reach an agreement over the global average before starting the redistribution of actual tasks. After reaching consensus over the global average, the nodes start exchanging tasks by comparing their own load with the global average (ϕ) instead of the local average ($\text{ave}_i(k)$). I-LB has the potential of reducing unnecessary transmissions and as a result the overall bandwidth usage. In this part, we compare the performance of I-LB with the O-LB approach and explore the underlying tradeoffs.

A. Discrete-Time Consensus Problem

In consensus problems, a group of nodes try to reach an agreement over a certain value (average of the initial queue lengths in our case). They exchange information with their neighboring agents and update their values according to a given update protocol. Define $y_i(k)$ as the status of node i at the kth instant of the consensus process. We have $y_i(0) = x_i(0),\ \forall i \in N$. The network is said to be in consensus if $y_i = y_j$ for all i, j. When each $y_i = \frac{1}{n}\Sigma_j y_j(0)$, the team has reached average consensus [14].

B. Consensus over the Global Average and Informed Load Balancing

Let $A = [a_{ij}]$ represent the adjacency matrix of the underlying graph with $a_{ii} = 0$ and $a_{ij} > 0$ if there exists a directed edge from node j to node i. The Laplacian of the graph is then defined as $L = [l_{ij}]$ with:

$$l_{ij} = \begin{cases} \sum_{k=1, k\neq i}^{n} a_{ik}, & , \quad j = i \\ -a_{ij}, & , \quad j \neq i \end{cases}$$

Let $y(k)$ be the information vector $y(k) = [y_1(k)...y_n(k)]^T$. The discrete-time consensus protocol will then be [14]:

$$y(k+1) = (I - \epsilon L) y(k), \qquad (15)$$

where $\epsilon \in (0, 1/\max_i l_{ii})$ and I represents the identity matrix.

In our case, it is desirable that the network reaches average consensus which corresponds to the system average (ϕ). In [14], it was proved that (15) can achieve average-consensus if the graph is balanced and strongly connected. Larger values of ϵ can furthermore increase the convergence rate. Since we assumed that queue lengths were exchanged over undirected graphs in the previous section, we assume the same here for fair comparison. Once the nodes reach consensus and switch to redistributing the tasks, we take the graph over which they exchange the tasks to be directed.

Once consensus is reached, the redistribution of tasks starts. We have the following modifications to the original algorithm: $L_i^{ex}(k) = x_i(k) - \phi$, $L_{j(i)}^{ex}(k) = x_j(k) - \phi$ and, $\mathcal{M}_i(k) = \{j | j \in \mathcal{N}_i,\ \phi > x_j(k)\}$. Eqs. (5) and (6) remain unchanged. In practice, each node has to switch to exchanging loads after it senses that consensus is reached. In order to do so, it could monitor its value ($y_i(k)$) and those of

its neighbors and declare that consensus is reached if those values do not change over a given time.

C. Underlying Tradeoffs Between O-LB and I-LB

In this section we explore the underlying tradeoffs between O-LB and I-LB in terms of speed of convergence, bandwidth usage and performance guarantees.

For I-LB, we take $\epsilon = 1/(1.1 \max_i l_{ii})$, which will guarantee convergence. We also assume that the nodes can detect accurately when the consensus state is reached in order to proceed to redistributing loads.

Consider the undirected path network of Fig. 7, with an initial load distribution of $x(0) = [707\ 486\ 332\ 951]^T$. If no information exchange is done, 49 load balancing steps are required to reach the balanced state whereas for I-LB, 23 time steps are first required to reach consensus followed by 2 load balancing steps.

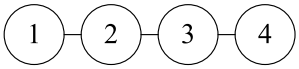
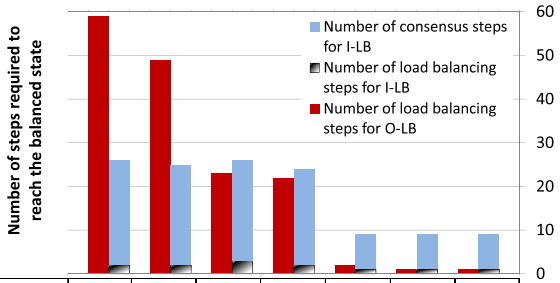


Fig. 7. Network topology of a path network. I-LB can reach a balanced state considerably faster for an initial load distribution $x(0) = [707\ 486\ 332\ 951]^T$.

I-LB, however, does not always take fewer iterations than O-LB. As an example, consider the graph of Fig. 1 (left) with the directed links replaced by undirected ones. Let the initial load distribution be $x(0) = [707\ 486\ 332\ 951]^T$. If O-LB is used, the system reaches the balanced state at time $k = 15$. For I-LB however, it takes 21 time steps to reach average consensus and 2 load balancing steps afterwards to reach the balanced state.

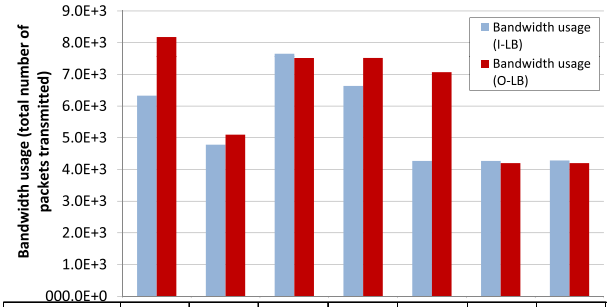
Figures 8 and 9 show a comparison of O-LB and I-LB for various configurations of an undirected network with 4 nodes and initial load $x(0) = [707\ 486\ 332\ 951]^T$. The two approaches are compared in terms of speed of convergence (number of time steps required to reach the balanced state) as well as bandwidth usage. The figures also indicate the second smallest eigenvalue of the underlying graph Laplacian as an indicator of the connectivity of each graph. The larger the second smallest eigenvalue is, the higher the graph connectivity will be. In Fig. 8, the bars representing I-LB include the time steps required to reach average consensus plus the actual load balancing steps required to get to the balanced state. By comparing the last two rows, it can be seen that for graphs with lower connectivity, I-LB can perform considerably better than O-LB in terms of the total time steps required to reach the balanced state. However, as the graph connectivity increases, O-LB can outperform I-LB.

Next we compare the bandwidth usage of O-LB and I-LB. By bandwidth usage we mean the total number of packets that are transmitted in the network. We assume that each unit of task requires 10 times the number of packets required for transmitting the queue length information of a node. Figure 9 shows the corresponding bandwidth usage of I-LB and O-LB for the initial distribution $x(0) = [707\ 486\ 332\ 951]^T$ and various configurations of an undirected network with



Number of links	3	3	3	4	5	5	6
Second smallest eigenvalue of L	0.59	0.59	1.00	1.00	2.00	2.00	4.00
Load balancing steps for I-LB	2	2	3	2	1	1	1
Consensus steps for I-LB	24	23	23	22	8	8	8
Total steps (Consensus + load balancing) for I-LB	26	25	26	24	9	9	9
Load balancing steps for O-LB	59	49	23	22	2	1	1

Fig. 8. Comparison of the time steps required to reach the balanced state for various network topologies using O-LB and I-LB with 4 nodes.



Number of links	3	3	3	4	5	5	6
Second smallest eigenvalue of L	0.59	0.59	1.00	1.00	2.00	2.00	4.00
Bandwidth usage (I-LB)	6.3E+3	4.8E+3	7.7E+3	6.6E+3	4.3E+3	4.3E+3	4.3E+3
Bandwidth usage (O-LB)	8.2E+3	5.1E+3	7.5E+3	7.5E+3	7.1E+3	4.2E+3	4.2E+3

Fig. 9. Comparison of the bandwidth usage (total number of transmitted packets) required to reach the balanced state for various network topologies with 4 nodes.

4 nodes. In most cases I-LB results in a better bandwidth usage. The difference will be more notable as the ratio of the size of task packets to information packets becomes larger. It should be noted that the second column of Fig. 8 and 9 corresponds to the graph of Fig. 7 while the first column is for the same graph but with a different ordering of the nodes. As a result, the nodes experience different loads in their neighborhood. It can be seen that this results in different performances, which highlights the impact of the initial distribution of the loads over the network on the overall behavior.

D. Lack of Asymptotic Performance Guarantee for I-LB

While I-LB has the potential of reducing the time and/or bandwidth required for reaching the balanced state, it lacks asymptotic performance guarantee. In section IV, we showed that load balancing according to (1)-(7) and over a strongly connected graph can guarantee convergence to the balanced state. For I-LB, however, this is not the case. We show this

with a counterexample. Consider the path network of Fig. 7, but with a different initial distribution $x(0) = [20\ 19\ 20\ 17]^T$. If no information exchange is done, i.e. O-LB is performed, the system reaches the balanced state, as proved by Theorem 1 (see Fig. 10). However, for I-LB, although the system reaches consensus, it is not able to reach the balanced state as seen in Fig. 11. To understand this, consider node 1 and 2. At $k = 0$, node 2 is already balanced but node 1 is still overloaded and can only balance itself with node 2. Since node 2 already reached the global average ϕ, it is not a candidate to receive any load, which results in the system not reaching the balanced state. Intuitively, we can see that as long as an overloaded node has a neighbor whose queue is below the global average, it can still reach the balanced state. The initial distribution of the loads, therefore, plays a key role in the asymptotic behavior of I-LB.

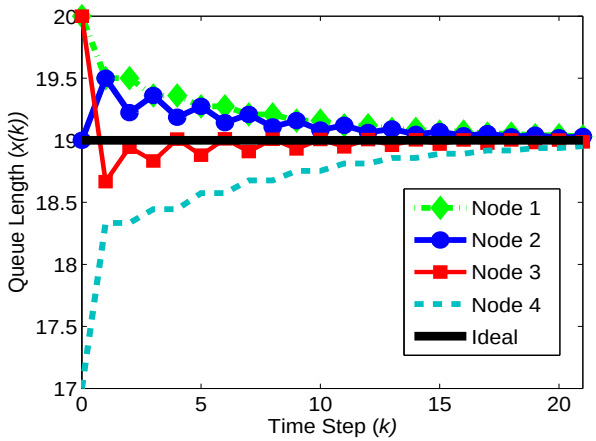


Fig. 10. Queue dynamics for load balancing over the topology of Fig. 7 with no information exchange. The queues reach a balanced state.

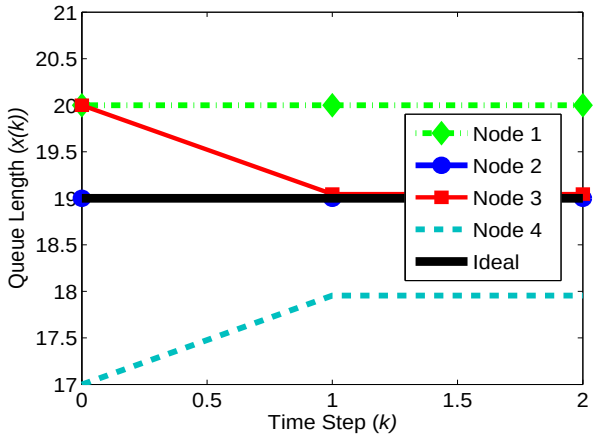


Fig. 11. Queue dynamics for I-LB over the graph of Fig. 7 and with $x(0) = [20\ 19\ 20\ 17]^T$. The queues cannot reach the balanced state.

VI. CONCLUSION AND FUTURE WORK

In this paper we considered the problem of distributed load balancing over a directed graph that is not fully connected. We studied the impact of network topology on the stability and balance of distributed computing. For the case of a one-time arrival of the loads, we showed that load balancing over a strongly connected graph reaches a balanced state, independent of the initial load distribution, while having a spanning tree is not a sufficient condition for reaching a balanced state. We furthermore proposed Informed Load Balancing (I-LB), an approach in which the nodes first reach an agreement over the global balanced state before proceeding to redistribute their tasks. We explored the underlying tradeoffs between I-LB and the Original Load Balancing (O-LB) approach. While I-LB lacks asymptotic performance guarantees of O-LB, it can increase the speed of convergence and/or reduce the bandwidth usage especially for low-connectivity graphs. We are currently working on extending our results to the cases with continuous incoming and outgoing loads. A further extension could include considering communication delays as well as heterogeneous processing speeds.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Majeed Hayat and Jorge Pezoa for useful discussions.

REFERENCES

[1] S. Dhakal, "Load balancing in communication constrained distributed systems," Ph.D. dissertation, University of New Mexico, 2006.

[2] M. Dobber, G. Koole, and R. van der Mei, "Dynamic load balancing experiments in a grid," in *IEEE International Symposium on Cluster Computing*, vol. 2, May 2005, pp. 1063–1070.

[3] S. Dhakal, M. Hayat, J. Pezoa, C. Yang, and D. Bader, "Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 485–497, April 2007.

[4] S. Dhakal, B. Paskaleva, M. Hayat, E. Schamiloglu, and C. Abdallah, "Dynamical discrete-time load balancing in distributed systems in the presence of time delays," in *Proceedings of 42nd IEEE Conference on Decision and Control*, vol. 5, Dec. 2003, pp. 5128–5134.

[5] J. Chiasson, Z. Tang, J. Ghanem, C. Abdallah, J. Birdwell, M. Hayat, and H. Jerez, "The effect of time delays on the stability of load balancing algorithms for parallel computations," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 932–942, Nov. 2005.

[6] S. Dhakal, "Load balancing in delay-limited distributed systems," Master's thesis, University of New Mexico, 2003.

[7] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing on networks with gossip-based distributed algorithms," in *Proceedings of the 46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 500–505.

[8] A. Kashyap, T. Basar, and R. Srikant, "Consensus with quantized information updates," in *45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 2728–2733.

[9] B. Joshi, S. Hosseini, and K. Vairavan, "Stability analysis of a load balancing algorithm," in *Proceedings of the Twenty-Eighth Southeastern Symposium on System Theory*, March-April 1996, pp. 412–415.

[10] W. C. Jakes, *Microwave Mobile Communications*. New York: Wiley-IEEE Press, 1974.

[11] A. Cortes, A. Ripoll, M. Senar, and E. Luque, "Performance comparison of dynamic load-balancing strategies for distributed computing," in *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, 1999.

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

[13] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, New Jersey: Prentice-Hall, 1989, ch. Partially Asynchronous Iterative Methods.

[14] D. Kingston and R. Beard, "Discrete-time average-consensus under switching network topologies," in *Proceedings of the 2006 American Control Conference*, June 2006, pp. 3551–3556.