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

Load balancing over heterogeneous networks with gossip-based algorithms

Mauro Franceschelli, Alessandro Giua and Carla Seatzu

Abstract— **In this paper we consider the problem of load balancing over heterogeneous networks, i.e. networks whose nodes have different speeds. We assume that tasks are indivisible and with different weights. Our goal is that of minimizing the maximum execution time over nodes.**

We provide a gossip-based distributed algorithm whose convergence to a bounded set is guaranteed. We show that the convergence time of the proposed algorithm relies ultimately on the average meeting time between two agents performing a random walk on a graph.

I. Introduction

In this paper we consider the problem of load balancing over heterogeneous networks, i.e. networks whose nodes have different speeds. Our goal is that of determining, using *consensus* algorithms based on *gossip* [4], [6], [16], [18], the solution that minimizes the maximum execution time over nodes. It is based on the recent work by Kashyap *et al.* [18] and on our previous results in [12] where homogeneous networks have been considered.

The study of consensus networks has recently stirred much interest in the control community with a particular focus on the deep connection between consensus and algebraic graph theory [9], [14], [15], [17], [20], [23], [24], [25], [26].

In several applicative domains related to consensus the assumption that the state of each node is a continuous variable is clearly an oversimplified assumption, and it is necessary to explicitly take into account the discrete nature of loads composed by indivisible tasks (*discrete* or *quantized consensus*) [2], [3], [10], [18], [19], [22]. An interesting application in this sense is given by load balancing over networks [1], [5], [7], [11], [13], [21]. This is the reason why our presentation will be carried out within this framework even if the proposed results can also be applied to other application domains.

In particular, in this paper we propose a general framework for quantized consensus assuming the network is heterogenous, i.e., composed by nodes of different speeds. This is an appropriate formalism to describe several real applications, e.g., a network consisting of a low cost cluster made by off-the-shelf, low cost, processing units where the heterogeneity is the result of the low cost requirement (second hand hardware for instance). The consensus problem for this type of nets, as far as we know, has not received much attention in the control literature.

The objective is that of balancing the total load in the net assigning to each node i a fraction x_i of the total load proportional to its speed, so as to minimize the maximum execution time. We develop a gossip algorithm that converges to a predefined configuration under the constraint that at each time the total load of the net remains constant.

We assume that the total load is composed by discrete tasks with weights of arbitrary size. In such a case the optimal solution does not always correspond to a configuration in which all nodes have the same execution time. As discussed in [12] this is not related to our particular approach but is intrinsic in the nature of gossip, that implements at each step a pairwise optimization, and does not always yield an optimal solution. However, we prove that there exists a bounded set that contains the optimal solution that is always reachable and we study the convergence properties and the convergence time to this bounded set.

As mentioned in the literature [12], [18], in the case of discrete consensus to ensure good convergence properties it is necessary to enrich the gossip algorithm with an appropriate *swapping rule*. Whenever a balancing between two nodes is not possible, the swap "shakes" the network configuration to redistribute the load and allows loads composed by discrete tasks to travel in the network, reaching a situation in which a new balancing may occur.

As a final remark we observe that no task status exchange nor task transfer costs have been considered here for fine granularity load balancing. This can be reasonable in the case of load balancing on Massively Parallel Processing (MPP) or heterogeneous nodes with specially dedicated, high–speed communication channels. However, assuming that no cost is associated to transfer is an oversimplified assumption in most of the load balancing applications where nodes should only exchange their loads when strictly necessary. We made such an assumption because it is necessary to introduce swaps that allow the balancing among nodes that are not connected. Note however that, as already discussed above, even if the proposed procedure is presented with reference to the load balancing application, it provides a general result in terms of quantized consensus over heterogeneous networks.

Finally, we assume that no random perturbations occur for the convenience of system analysis, even if we are aware that such perturbations may be not negligible in certain realities (e.g., network traffic, memory utilization, ect.).

The main contribution of this paper is twofold. Firstly, starting from the results in [12], [18] it provides a gossip-based algorithm for heterogeneous networks using the notion of swap domains. Secondly, it provides an analysis of the convergence properties of the proposed algorithm for some classes of network topology.

M. Franceschelli, A. Giua and C. Seatzu are with the Department of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy {mauro.franceschelli,giua,seatzu@diee.unica.it}.

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

II. Problem statement

We consider a heterogeneous network of n nodes whose connections can be described by an undirected connected graph $\mathcal{G} = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. To each node $i \in V$ is allocated a *load* x_i that must be processed. The *speed factor*, denoted γ_i, represents the amount of load that can be processed in a time unit by node i.

We assume that K indivisible tasks should be assigned to the nodes, and an integer cost c_j ($j = 1, \ldots, K$) is associated to each task. We define a cost vector $c \in \mathbb{N}^K$ whose j-th component is equal to c_j, and n binary vectors $\vec{y}_i \in \{0,1\}^K$ such that

$$y_{i,j} = \begin{cases} 1 & \text{if the } j\text{-th task is assigned to node } i \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Note that with the proposed notation the load of each node can be expressed as $x_i = c^T \vec{y}_i$.

In the following we denote $\gamma_{\min}$ the smallest speed in the network (clearly $\gamma_{\min} > 0$), and $c_{\max}$ the maximum cost of tasks in the network.

The load distribution which we are looking for is the one that minimizes the *maximum execution time*, starting from any initial condition. Namely, if we define the load and speed vectors

$$\vec{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^T$$
$$\gamma = \begin{bmatrix} \gamma_1 & \gamma_2 & \ldots & \gamma_n \end{bmatrix}^T$$

and $\Gamma = \mathrm{diag}(\gamma)$, we would like to minimize the following objective function:

$$f(x) = \max_{i=1,\ldots,n} \frac{x_i}{\gamma_i} = \|\Gamma^{-1} x\|_\infty \quad (2)$$

under the assumption that the total load remains constant, namely $\mathbf{1}^T x = \mathbf{1}^T x(0)$, where $x(0)$ represents the initial load configuration.

Using a centralized approach an optimal solution to this problem can be determined solving the following integer programming problem with binary variables:

$$\begin{cases} \min V = \|c^T Y \Gamma^{-1}\|_\infty \\ s.t. \\ Y\mathbf{1} = \mathbf{1} \\ y_{i,j} \in \{0,1\} \quad \forall\, i = 1, \ldots, n; \;\; j = 1, \ldots, K. \end{cases} \quad (3)$$

We denote Y^* (resp., V^*) the optimal solution (resp., the optimal value of the performance index) of Problem (3).

In the following section we provide a dynamic decentralized balancing scheme based on gossip.

III. Gossip algorithm

A. Swap definition

We first define a task exchange process between two adjacent nodes that, while not changing the value of the objective function, modifies the load configuration. The definition we propose here is an extension of the one in [12].

***Definition** 1 (Swap):* Let us consider two nodes i and r incident on the same edge. Let $\mathcal{K}_i(t)$ ($\mathcal{K}_r(t)$) be the set of tasks contained in node i (r) at time t. Let $\mathcal{I}_i \subseteq \mathcal{K}_i(t)$ and $\mathcal{I}_r \subseteq \mathcal{K}_r(t)$ be two subsets of their tasks such that $\mathcal{I}_i \cup \mathcal{I}_r \neq \emptyset$.

Let us call *swap* the operation that moves the tasks in $\mathcal{I}_i$ to r, and the tasks in $\mathcal{I}_r$ to i at time $t+1$, reaching the distribution

$$\mathcal{K}_i(t+1) = \mathcal{I}_r \cup (\mathcal{K}_i(t) \setminus \mathcal{I}_i),$$
$$\mathcal{K}_r(t+1) = \mathcal{I}_i \cup (\mathcal{K}_r(t) \setminus \mathcal{I}_r)$$

provided that the objective function locally defined for the two nodes does not change, i.e.,

$$\max\left\{ \sum_{j \in \mathcal{K}_{i(t+1)}} \left(\frac{c_j}{\gamma_i}\right), \sum_{j \in \mathcal{K}_{r(t+1)}} \left(\frac{c_j}{\gamma_r}\right) \right\} =$$
$$\max\left\{ \sum_{j \in \mathcal{K}_{i(t)}} \left(\frac{c_j}{\gamma_i}\right), \sum_{j \in \mathcal{K}_{r(t)}} \left(\frac{c_j}{\gamma_r}\right) \right\}.$$

In particular, a *total swap* occurs if $\mathcal{I}_i = \mathcal{K}_i(t)$ and $\mathcal{I}_r = \mathcal{K}_r(t)$, while a *partial swap* occurs if either $\mathcal{I}_i \subsetneq \mathcal{K}_i(t)$ or $\mathcal{I}_r \subsetneq \mathcal{K}_r(t)$. ■

We point out that there are many ways to implement a *swap*. We deliberately left it undefined since the problem of finding the optimal way to swap loads to minimize the convergence time of dynamic load balancing is an open problem of research.

B. A distributed algorithm

We now provide a decentralized rule to solve the optimization problem presented in Section II that is based on gossip.

We define two binary vectors $\hat{y}_i$ and $\hat{y}_r$ with the same meaning as $\vec{y}_i$ and $\vec{y}_r$ but with a number of elements equal to the number of tasks locally present in the nodes. We denote $\hat{K}_{ir}(t) = |\mathcal{K}_i(t) \cup \mathcal{K}_r(t)|$ the set of tasks present in nodes i and r at time t. We define $\hat{c} = c \uparrow_{\hat{K}_{ir}(t)}$ the projection of c in $\hat{K}_{ir}(t)$, namely a vector whose elements are the costs of the tasks present in nodes i and r at time t.

***Algorithm** 2 (Gossip Algorithm with discrete tasks):*

1) Let $t = 0$.
2) Select an edge $\{i, r\}$ at random.
3) Solve the integer programming problem (IPP):

$$\begin{cases} k^* = & \min k \\ & s.t. \\ & \dfrac{\hat{c}^T \hat{y}_i}{\gamma_i} \leq k \\ & \dfrac{\hat{c}^T (1 - \hat{y}_i)}{\gamma_r} \leq k \\ & k \in \mathbb{R}_+ \cup \{0\} \\ & \hat{y}_i \in \{0,1\}^{\hat{K}_{ir}(t)} \end{cases} \quad (4)$$

4) If $k^* < \max\left\{ \dfrac{\hat{c}^T \hat{y}_i(t)}{\gamma_i}, \dfrac{\hat{c}^T (1 - \hat{y}_i(t))}{\gamma_r} \right\}$ then let

$$\hat{y}_i(t+1) = \hat{y}_i,$$
$$\hat{y}_r(t+1) = \vec{1} - \hat{y}_i$$

else execute a swap.

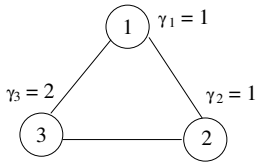

Fig. 1. The network discussed in Example 3.

t	edge	node 1	node 2	node 3	$V(Y)$
0		4, 10		1, 2, 2, 3, 3, 5, 6, 7	14.5
1	$\{1,3\}$	3, 5, 6		1, 2, 2, 3, 4, 7, 10	14.5
2	$\{2,3\}$	3, 5, 6	1, 2, 7	2, 3, 4, 10	14
3	$\{1,3\}$	5, 6	1, 2, 7	2, 3, 3, 4, 10	11

TABLE I

THE RESULTS OF APPLYING ALGORITHM 2 AT THE NET IN EXAMPLE 3.

5) Let $t = t + 1$ and goto step 2.

■

In practice IPP (4) provides the load distribution that minimizes the execution times at the two nodes. If the resulting distribution is better than the previous one, the load is assigned accordingly, otherwise a swap is executed.

Note that Algorithm 2 is based on the solution of NP-Hard problems. Appropriate heuristics, with a polynomial complexity in the number of tasks, can be formulated that still guarantee the convergence to a set $\widetilde{\mathcal{Y}}$ that will be defined in the following. An example of such heuristics is given in [12] in the case of unitary speeds.

The swap allows to overcome several blocking conditions: anytime the network reaches a local minimum of the objective function the swap "shakes" the network ensuring convergence within some precise bounds (see Theorem 7).

***Example** 3:* Let us consider the fully connected[1] net in Fig. 1 composed by 3 nodes with speeds $\gamma_1 = \gamma_2 = 1$ and $\gamma_3 = 2$. Assume that it contains 10 tasks whose weights are equal to $c_1 = 1$, $c_2 = c_3 = 2$, $c_4 = c_5 = 3$, $c_6 = 4$, $c_7 = 5$, $c_8 = 6$, $c_9 = 7$ and $c_{10} = 10$.

Assume that the initial configuration is

$$\mathcal{K}_1(0) = \{3, 10\}, \quad \mathcal{K}_2(0) = \emptyset,$$
$$\mathcal{K}_3(0) = \{1, 2, 4, 5, 6, 7, 8, 9\}.$$

Using Algorithm 2, we obtain the optimal load balancing in three steps, as summarized in Table I. In particular, here we have pointed out the selected edges, the weights of the tasks in each node, and the resulting values of the objective function. ■

C. Convergence properties

We now discuss the convergence properties of Algorithm 2 that are strictly related to the possibility of having swaps.

***Definition** 4 (Swap domain):* We call "swap domain" $G_\gamma \subseteq G$ a connected subgraph induced by nodes with the same speed. ■

[1]A network is *fully connected* if there is an arc from each node to any other one.

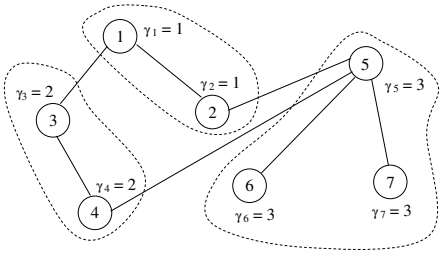

Fig. 2. The network discussed in Example 5.

In practice, given a graph that can be partitioned in a certain number of swap domains, if we perform graph compression and merge all the nodes belonging to the same swap domain under a single aggregate node, and repeat the procedure for all the swap domains, we finally obtain a fully connected (compressed) graph.

***Example** 5:* Let us consider the network in Fig. 2 that has seven nodes with three different speeds. This network can be partitioned in three different subgraphs G_1, G_2 and G_3 induced respectively by nodes $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6, 7\}$. In this case each swap domain is connected to each other. ■

Each swap domain identifies a set of nodes where "total swaps" may happen. On the contrary "total swaps" between adjacent nodes of different domains cannot occur.

It is relevant to note that the definition of "swap domain" is embedded in the graph topology. In particular the nodes don't need to know in which domain they are or even that any domain exists.

***Definition** 6:* We call *final set*

$$\widetilde{\mathcal{Y}} = \{Y = [\vec{y}_1\ \vec{y}_2\ \cdots\ \vec{y}_n] \ \mid \ \left|\frac{c^T\vec{y}_i}{\gamma_i} - \frac{c^T\vec{y}_r}{\gamma_r}\right| \leq \frac{c_{\max}}{\gamma_{\min}}, \quad \forall\ i, r \in \{1, \ldots, n\}\} \tag{5}$$

i.e., the set of configurations such that, for any couple of nodes $i, r \in V$, the difference among their execution times is at most equal to the ratio $c_{\max}/\gamma_{\min}$. ■

***Theorem** 7:* Let $Y(t)$ be the matrix that summarizes the load balancing resulting from Algorithm 2 at the generic time t. If each swap domain is connected to each other, it holds

$$\lim_{t\to\infty} \Pr\left(Y(t) \in \widetilde{\mathcal{Y}}\right) = 1$$

where $\Pr(Y(t) \in \widetilde{\mathcal{Y}})$ denotes the probability that $Y(t) \in \widetilde{\mathcal{Y}}$.

Proof: We define a Lyapunov-like function

$$V(t) = [V_1(t), V_2(t)] \tag{6}$$

consisting of two terms. The first one is equal to the objective function of (3), namely

$$V_1(t) = \|c^T Y(t)\Gamma^{-1}\|_\infty. \tag{7}$$

The second one is a measure of the number of nodes whose execution time is equal to $\|c^T Y(t)\Gamma^{-1}\|_\infty$, i.e.,

$$V_2(t) = \left|\arg \max_{i=1,\ldots,n} \frac{c^T\vec{y}_i(t)}{\gamma_i}\right|. \tag{8}$$

Note that we impose a lexicographic ordering on the performance index, i.e., $V = \bar{V}$ if $V_1 = \bar{V}_1$ and $V_2 = \bar{V}_2$; $V < \bar{V}$ if $V_1 < \bar{V}_1$ or $V_1 = \bar{V}_1$ and $V_2 < \bar{V}_2$.

The proof is based on three arguments.

(1) We first prove that $V(t)$ is a non increasing function of t.

This is trivially true when a swap is executed, since in such a case $V(t+1) = V(t)$.

Consider the case in which the selected nodes i and r balance their load. It holds

$$\max\left\{\frac{c^T\vec{y}_i(t+1)}{\gamma_i}, \frac{c^T\vec{y}_r(t+1)}{\gamma_r}\right\} < \max\left\{\frac{c^T\vec{y}_i(t)}{\gamma_i}, \frac{c^T\vec{y}_r(t)}{\gamma_r}\right\},$$

hence three different cases may happen.

(a) One of the selected nodes is the only node in the network such that its execution time is equal to $\|c^T Y \Gamma^{-1}\|_\infty$. In such a case $V_1(t+1) < V_1(t)$ hence $V(t+1) < V(t)$.

(b) One of selected nodes is such that its execution time is equal to $\|c^T Y(t)\Gamma^{-1}\|_\infty$ but there exists at least one other node in the network with the same execution time. In such a case $V_1(t+1) = V_1(t)$ and $V_2(t+1) = V_2(t) - 1$, hence $V(t+1) < V(t)$.

(c) The execution time of both the selected nodes is smaller than $\|c^T Y(t)\Gamma^{-1}\|_\infty$. In such a case $V(t+1) = V(t)$.

(2) Secondly, we observe that, if the current configuration is outside the final set $\widetilde{\mathcal{Y}}$, then there exists at least one node whose execution time is equal to $\|c^T Y(t)\Gamma^{-1}\|_\infty$ that could balance his load with (at least) one other node if they were incident on the same arc: this would reduce function $V(t)$ (see cases (a) and (b) of the previous item).

To prove this we observe that if the current configuration is outside the final set $\widetilde{\mathcal{Y}}$, then there exists (at least) one couple of nodes i and r such that

$$\frac{c^T\vec{y}_i(t)}{\gamma_i} - \frac{c^T\vec{y}_r(t)}{\gamma_r} > \frac{c_{\max}}{\min\{\gamma_i, \gamma_r\}} \tag{9}$$

where $\frac{c^T\vec{y}_i(t)}{\gamma_i}$ is equal to the maximum execution time. If we move a task $c_j \leq c_{\max}$ from node i to node r we have:

$$c^T\vec{y}_i(t+1) = c^T\vec{y}_i(t) - c_j,$$

$$c^T\vec{y}_r(t+1) = c^T\vec{y}_r(t) + c_j.$$

Now

$$\frac{c^T\vec{y}_i(t+1)}{\gamma_i} = \frac{c^T\vec{y}_i(t) - c_j}{\gamma_i} < \frac{c^T\vec{y}_i(t)}{\gamma_i} \tag{10}$$

and

$$\frac{c^T\vec{y}_r(t)}{\gamma_r} + \frac{c_j}{\gamma_r} \leq \frac{c^T\vec{y}_r(t)}{\gamma_r} + \frac{c_{\max}}{\min\{\gamma_i, \gamma_r\}} < \frac{c^T\vec{y}_i(t)}{\gamma_i}$$

where the second inequality follows from assumption (9); thus

$$\frac{c^T\vec{y}_r(t+1)}{\gamma_r} = \frac{c^T\vec{y}_r(t) + c_j}{\gamma_r} < \frac{c^T\vec{y}_i(t)}{\gamma_i}. \tag{11}$$

By (10) and (11) it follows that

$$\max\left\{\frac{c^T\vec{y}_i(t+1)}{\gamma_i}, \frac{c^T\vec{y}_r(t+1)}{\gamma_r}\right\} < \max\left\{\frac{c^T\vec{y}_i(t)}{\gamma_i}, \frac{c^T\vec{y}_r(t)}{\gamma_r}\right\}.$$

(3) Finally, we observe that being each swap domain connected to each other, there exists a series of swaps that lead to a configuration in which the loads of the two nodes identified in the previous item are adjacent and the arc between them is selected. This happens with probability 1 as t goes to infinity. □

***Remark** 8:* Theorem 7 characterizes the convergence properties of Algorithm 2 in terms of a finite set $\widetilde{\mathcal{Y}}$. This obviously does not imply that an optimal load balancing is achieved.

As shown in [12] this is not a limitation of the particular proposed algorithm. An optimal load balancing with non-unitary tasks cannot always be achieved by greedy gossip algorithms, that balance the load between two nodes at each step, even on a fully connected network. In fact, to reach consensus an optimization involving more than two nodes at the same time may be necessary. ■

We also note that Theorem 7 provides only a sufficient condition for the convergence inside the set $\widetilde{\mathcal{Y}}$. To prove that it is not necessary we may consider an initial load distribution that is already balanced, i.e. $Y(0) \in \widetilde{\mathcal{Y}}$. Furthermore, due to the random nature of the gossip algorithm, it is also easy to formulate other examples that end in $\widetilde{\mathcal{Y}}$ even if the assumptions of Theorem 7 do not hold.

Finally we also observe that no swap is necessary to obtain a solution inside $\widetilde{\mathcal{Y}}$ in the case of a fully connected network, since any node can communicate with any other node.

IV. Convergence time of Algorithm 2

In this section we discuss the expected convergence time of Algorithm 2, and provide an upper bound to it in the case of two different net topologies[2].

In the following we assume that only total swaps are allowed inside each swap domain.

The *convergence time* is a random variable defined for a given initial load configuration $Y(0) = Y$ as:

$$T_{conv}(Y) = \inf\{t \mid \forall\, t' \geq t,\ Y(t') \in \widetilde{\mathcal{Y}}\}.$$

Thus, $T_{conv}(Y)$ represents the number of steps required at a certain execution of Algorithm 2 to reach the convergence set $\widetilde{\mathcal{Y}}$ starting from a given tasks distribution.

Let us firstly introduce the following notation.

- $N_{\max}$ is the maximum number of improvements of $V(t)$ defined as in (6), needed by any realization of Algorithm 2 to reach the set $\widetilde{\mathcal{Y}}$, starting from a given configuration.
- $T_{\max}$ is the maximum average time between two consecutive improvements of $V(t)$ defined as in (6), needed

[2]The approach we use for this evaluation is inspired by the methodology used by Kashyap *et al.* in [18].

by any realization of Algorithm 2, starting from a given configuration.

Using the previous notation, it follows that the *expected convergence time* is

$$\mathcal{E}[T_{conv}(Y)] \leq N_{\max} \cdot T_{\max}. \tag{12}$$

The following proposition provides an upper bound on $N_{\max}$ that is independent from the net topology.

***Proposition** 9:* Let us consider a net with n nodes and let γ be the corresponding speed vector. Let $x(0)$ be the vector representative of the initial amount of load at nodes. It holds:

$$N_{max} \leq (n-1) \cdot \varrho \cdot (M-m) \tag{13}$$

where

$$\begin{aligned} M &= \|\Gamma^{-1}x(0)\|_\infty, \\ m &= \frac{\sum_{i=1}^{n} x_i(0)}{\sum_{i=1}^{n} \gamma_i} = \frac{\mathbf{1}^T x(0)}{\mathbf{1}^T \Gamma \mathbf{1}}, \\ \varrho &= \max_{\{i,r\}\in E} \text{mcm}\{\gamma_i, \gamma_r\}, \end{aligned} \tag{14}$$

and mcm denotes the minimum common multiple.

Proof: By definition the maximum number of improvements of $V_1 = f$ needed by any realization of Algorithm 2 to reach the set $\widetilde{\mathcal{Y}}$ is smaller or equal to the ratio between the global improvement of f needed before reaching the convergence set $\widetilde{\mathcal{Y}}$ starting from $x(0)$, and its minimum admissible improvement.

By Step 5 of Algorithm 2 the load distribution is updated if and only if leads to an improvement of the objective function, otherwise a swap is executed. Thus, the largest value of $f(x)$ occurs at the initial configuration and is equal to $M = f(x(0)) = \|\Gamma^{-1}x(0)\|_\infty$.

The minimum value of $f(x)$ corresponds to the case of perfect load balancing, that in general is not achievable in the discrete case. However, a lower estimate of it is given by its optimal value in the case of infinitely divisible tasks, namely by $f(x^*)$ where $x^* = \alpha\gamma$ and $\alpha = \dfrac{\mathbf{1}x(0)}{\mathbf{1}^T\Gamma\mathbf{1}}$. Thus, if we define $m = f(x^*) = \alpha$, then for any load balancing x it holds $m \leq f(x)$.

We also observe that the minimum load exchange is equal to 1 since all tasks have an integer cost. Now, if we consider the generic edge $\{i, r\}$, we know that the minimum improvement of f that we may obtain when balancing this edge is equal to $1/\text{mcm}\{\gamma_i, \gamma_r\}$. As a consequence the minimum improvement of f at a generic step of Algorithm 2 is equal to $1/\varrho = 1/\max_{\{i,r\}\in E} \text{mcm}\{\gamma_i, \gamma_r\}$, where E is the set of edges.

Thus, we may conclude that the largest number of improvements of f before reaching the convergence set $\widetilde{\mathcal{Y}}$ starting from $x(0)$ is at most equal to $\varrho \cdot (M-m)$.

Finally, in the worst case $n-1$ consecutive balancing may occur before having an improvement of f, namely $n-1$ consecutive reductions of V_2 may occur before having a reduction of $V_1 = f$. In particular, this case may happen if $n-1$ nodes have the same execution time that is equal to the maximum one. In this case, a first balancing may occur between the only "different" node and any of the other ones. Then, a new balancing may occur between any of the remaining $n-2$ nodes with the maximum execution time and one with a smaller execution time, and so on. □

We now focus on $T_{\max}$. Evaluating $T_{\max}$, and hence the average convergence time (12) of Algorithm 2, is in general a difficult issue because it is strictly related to the particular topology of the net.

In the following we consider two cases: *fully connected* networks and *generalized ring topology* nets. In both cases the computation is carried out using Markov chains. It is not easy to generalize such analytical results to arbitrary net topologies. However, similar approaches based on Markov chains can always be used to evaluate numerically an upper bound on $T_{\max}$ for a particular net example.

A. Fully connected networks

***Proposition** 10:* Let us consider a fully connected network, and let n be the number of nodes.

It holds

$$T_{\max} = \frac{n(n-1)}{2}. \tag{15}$$

Proof: The maximum average time between two consecutive balancing occurs when only one balancing is possible. Thus, if N is the number of arcs of the net, then the probability of selecting the only arc whose incident nodes may balance their load is equal to $p = 1/N$, while the average time needed to select it is equal to N. Since the network is fully connected, if n is the number of nodes, the number of arcs is $N = n(n-1)/2$ and so $T_{\max} = n(n-1)/2$. □

***Proposition** 11:* If a net is fully connected, the average convergence time of Algorithm 2 is

$$\mathcal{E}[T_{conv}(Y)] \leq \varrho \cdot (M-m) \cdot \frac{n(n-1)^2}{2} = \mathcal{O}(n^3).$$

Proof: Follows from equation (12) and Propositions 9 and 10. □

B. Generalized ring topology

***Definition** 12 (Generalized ring topology):* A graph $\mathcal{G} = (E, V)$ has a generalized ring topology if it satisfies the following assumptions.

- It is composed by s rings, each one with k nodes. The generic j-th ring R_j is a graph $R_j = (V_j, E_j)$ with $V_j = \{1, \ldots, k\}$ and

  $$E_j = \{\{i, r\} \in E \mid r = i+1,\ \forall i = 1, \ldots, k-1\} \cup \{k, 1\}.$$

- The same speed is associated to all nodes in the same ring, while nodes of different rings have different speeds. Thus each ring defines a different swap domain.
- Let (i, j), with $i = 1, \ldots, k$ and $j = 1, \ldots, s$, be the i-th node of ring R_j. Let $\Sigma_i = \{(i, j) \in V,\ j = 1, \ldots, s\}$ be the set of the nodes of *index* i in all rings. All nodes in Σ_i are fully connected, i.e., for all $i = 1, \ldots, k$, there

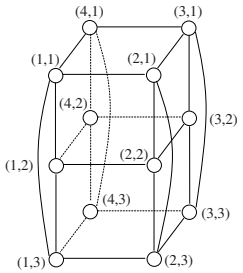

Fig. 3. A net with a generalized ring topology where $s = 3$ and $k = 4$.

exists an edge in E that connects each node in Σ_i with any other node in Σ_i.

■

An example of a net with a generalized ring topology is reported in Fig. 3: here $s = 3$ and $k = 4$.

Note that such a topology well fits with our problem for two main reasons. Firstly, it is scalable both in the number of nodes in the rings and in the number of rings (namely in the number of swap domains). Secondly, the diameter of the net, namely the maximum distance among nodes that may balance, increases with the number of nodes in the ring.

***Proposition** 13:* Let us consider a net with a generalized ring topology. Let s be the number of rings and $n = k \cdot s$ be the total number of nodes in the net.

If only total swaps are executed[3], then it holds

$$T_{\max} \leq \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) = \frac{k^2 s(s+1)}{32} \cdot (k+16)\,. \tag{16}$$

Proof: We preliminary observe that, due to the gossip nature of Algorithm 2 and to the random rule used to select the edges, the problem of evaluating an upper bound on $T_{\max}$ can be formulated as the problem of finding the average meeting time of two agents walking on a graph executing a random walk[4]. In fact, the average meeting time of the two agents may be thought as the average time of selecting an edge whose incident nodes may balance their load. Note that in general more than two edges may balance their load, thus assuming that only two agents are walking on the graph provides us an upper bound on the value of $T_{\max}$. In particular, the worst case in terms of meeting time occurs when the two agents are on different rings.

In the following we compute the average meeting time using discrete Markov chains. For the sake of simplicity, we assume that the number of nodes k in each ring is even[5].

We call distance between two agents in nodes (i, j) and (i', j'), $d_{i,i'} = 1 + \min\{|i - i'|, k - |i - i'|\}$, namely the number of arcs in the shortest path connecting node i with node i'. In simple words the above distance is equal to the distance between the two agents, computed as if they were in the same ring, plus 1. This is consistent with the assumption that, in a generalized ring topology net, any node with a given index in a certain ring is connected to all the other nodes having the same index in different rings. Therefore nodes with a unitary distance are nodes within the same section Σ. Under the assumption that k is even, the maximum distance between the two agents is equal to $D = k/2 + 1$.

The Markov chain relative to a net with an even value of k is shown in Fig. 4, thus it is a particular birth-death process.

Each node (apart from the first one, named A) is characterized by an integer number that denotes the distance between the two nodes. Let us now discuss the weight of the arcs in the Markov chain.

— The weight of the arcs going from nodes i to $i + 1$, and viceversa, for $i = 2, \ldots, D - 1$ is equal to $2/N$ where $N = ks(s + 1)/2$ is the number of arcs[6]. This follows from the fact that if a net has N arcs the probability of selecting a generic edge is equal to $1/N$; moreover, if the distance between the two agents is $i = 1, \ldots, D - 1$, two are the edges whose selection leads to an increasing or decreasing of their distance.

The same reasoning explains the weight of the arc going from $D - 1$ to D and the weight of the arc going from 2 to 1.

— If the distance between the two agents is unitary (the state of the Markov chain is 1) two different cases may occur: either we select an edge that leads to a distance equal to 2, or the edge incident on the nodes containing the agents is selected. The first case occurs with a probability equal to $4/N$; the second case occurs with a probability equal to $1/N$ and leads to the absorbing state A.

— Now, assume that the distance between the agents is equal to D. In such a case the selection of 4 different arcs may lead to a decreasing of their distance. Therefore the arc of the Markov chain going from node D to node $D - 1$ has a weight equal to $4/N$.

— Finally, the weights of all self-loops are due to the fact that the sum of the weights of arcs exiting a node is equal to 1 in a discrete Markov chain.

Given the Markov chain in Fig. 4 it is easy to compute the average hitting time of the absorbing state from any admissible distance. This can be done solving analytically the following linear system of equations:

$$(I - P')\,\tau = \mathbf{1} \tag{17}$$

where I is the D-dimensional identity matrix; P' has been obtained by the probability matrix P of the Markov chain in Fig. 4 removing the row and the column relative to the absorbing state[7]; τ is the D-dimensional vector of unknowns: its i-th component $\tau(i)$ is equal to the hitting time of the absorbing state starting from an initial distance equal to i,

[3]In this paper we will always assume that only total swaps are executed. When partial swaps are also allowed all the bounds on the convergence time change. We do not provide a bound for such a case.

[4]This problem has been extensively studied in different fields [8].

[5]The case of rings with an odd number of nodes k is upper bounded by the case of rings with $k + 1$ nodes.

[6]The number of arcs of a ring topology net is equal to k times the number of arcs of each section Σ, plus k times the number of arcs of each ring. Being each Σ a fully connected graph with s nodes, its number of arcs is equal to $s(s - 1)/2$. Therefore, $N = ks(s + 1)/2 + ks = ks(s + 1)/2$.

[7]It obviously holds that the hitting time of the absorbing state is null from the absorbing state itself.

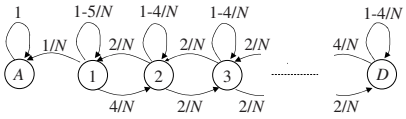

Fig. 4. The Markov chain associated to a generalized ring topology net with an even value of k.

for $i = 1, \ldots, D$; finally, $\mathbf{1}$ is the D-dimensional column vector of ones. We found out that the worst case in terms of hitting time occurs when the two agents are at their maximum distance, i.e., for $i = D$. In particular it is

$$\tau(D) = \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) = \frac{k^2 s(s+1)}{32} \cdot (k+16)$$

where the last equality follows from the fact that $n = ks$. This proves the statement being $T_{\max} \leq \tau(D)$. □

Proposition 14: If a net has a generalized ring topology and only total swaps are executed, then the average convergence time of Algorithm 2 in terms of the number of nodes n is

$$\begin{aligned}&\mathcal{E}[T_{conv}(Y)]\\ &\leq \varrho \cdot (M-m) \cdot \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) \cdot (n-1) = \mathcal{O}(n^4)\end{aligned}$$

or, in terms of the net parameters k and s

$$\begin{aligned}&\mathcal{E}[T_{conv}(Y)]\\ &\leq \varrho \cdot (M-m) \cdot \frac{k^2 s(s+1)}{32} \cdot (k+16) \cdot (k\ s-1)\\ &= \mathcal{O}(k^4 s^3).\end{aligned}$$

Proof: Follows from equation (12) and Propositions 9 and 13. □

V. Conclusions and future work

In this paper we addressed the problem of determining an optimal load balancing over networks with nodes of different speed. The case of finite tasks with different costs has been considered. A solution based on gossip has been proposed and convergence properties have been examined in detail.

Then, we studied the convergence time of the proposed quantized gossip algorithm. In particular we examined two different net topologies, namely fully connected and generalized ring topologies.

In this paper we assumed that swaps are executed randomly. One of our future lines of research in this topic will be that of determining appropriate (deterministic) rules to execute swaps that improve the convergence properties of Algorithm 2 and provide a stop criterion when the optimality set is reached.

References

[1] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. In *Proc. of the 25th ACM Symposium on Theory of Computing*, pages 632–641, San Diego, CA, US, May 1993.

[2] T.C. Aysal, M.J. Coates, and M.G. Rabbat. Distributed average consensus using probabilistic quantization. In *Proc. of the IEEE Statistical Signal Processing Workshop*, Madison, Wisconsins, US, 2007.

[3] T.C. Aysal, M.J. Coates, and M.G. Rabbat. Rates of convergence for distributed average consensus with probabilistic quantization. In *Proc. of the Allerton Conference on Communication, Control, and Computing*, Urbana-Champaign, Illinois, US, 2007.

[4] O. Babaoglu, M. Jelasity, A. Kermarrec, A. Montresor, and M. van Steen. Managing coulds: a case for a fresh look at large unreliable dynamic networks. *ACM SIGOPS Operating Systems Review*, 40 (3):9–13, 2006.

[5] D. Bauso, L. Giarrè, and R. Presenti. Mechanism design for optimal consensus problems. In *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, CA, US, 2006.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: design, analysis and applications. In *Proc. IEEE Infocom 2005*, Miami, USA, March 2005.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Trans. on Information Theory*, 52 (6):2508–2530, 2006.

[8] N.H. Bshouty, L. Higham, and J. Warpechowska-Gruca. Meeting times of random walks on graphs. *Inf. Process. Lett.*, 69(5):259–265, 1999.

[9] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri. Communication constraints in the average consensus problem. *Automatica*, 44 (3):671–684, 2008.

[10] R. Carli and S. Zampieri. Efficient quantization in the average consensus problem. *Advances in Control Theory and Applications*, pages 31–49, 2007.

[11] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. of Parallel and Distributed Computing*, 7:279–301, 1989.

[12] M. Franceschelli, A. Giua, and C. Seatzu. Load balancing on networks with gossip based distributed algorithms. In *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, December 2007.

[13] B. Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. of Computer and Systems Sciences*, 53(3):357–370, 1996.

[14] N. Hayashi, T. Ushio, F. Harada, and A. Ohno. Consensus problem of multi-agent systems with non-linear performance functions. *International Journal of Robust and Nonlinear Control*, E90-A(10):2261–2264, 2007.

[15] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Autom. Control*, 48:988 –1001, 2003.

[16] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. on Computer Systems*, 23 (3):219–252, 2005.

[17] M. Ji, G. Ferrari-Trecate, M. Egerstedt, and A. Buffa. Containment control in mobile networks. *IEEE Trans. on Automatic Control*, 53:65–78, 2008.

[18] A. Kashyap, T. Başar, and R. Srikant. Quantized consensus. *Automatica*, 43,7:1192–1203, 2007.

[19] A. Nedic, A. Olshevsky, A. Ozdaglar, and J.N. Tsitsiklis. On distributed averaging algorithms and quantization effects. In *LIDS Technical Report 2778*, MIT, Lab. for Information and Decision Systems, 2007.

[20] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Automatic Control*, 49:1520–1533, 2004.

[21] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of Markov chains and the analysis of iterative load-balancing schemes. In *Proc. of the 39th Annual Symposium on Foundations of Computer Science*, pages 694–703, Palo Alto, CA, USA, November 1998.

[22] M.G. Rabbat. On spatial gossip algorithms for distributed averaging. In *Proc. of the IEEE Statistical Signal Processing Workshop*, Madison, Wisconsins, US, 2007.

[23] W. Ren and R.W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Trans. on Automatic Control*, 50 (5):655–661, 2005.

[24] W. Ren and R.W. Beard. *Distributed consensus in multi-vehicle cooperative control. Theory and applications*. Springer Verlag, 2008.

[25] G. Xie and L. Wang. Consensus control for networks of dynamic agents via active switching topology. In *Advances in Natual Computation*, Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2005.

[26] G. Xie and L. Wang. Consensus control for a class of networks of dynamic agents. *International Journal of Robust and Nonlinear Control*, 17 (10-11):941–959, 2006.