Fast Discrete Consensus Based on Gossip for Makespan Minimization in Networked Systems

Mauro Franceschelli [a], Alessandro Giua [b,a], Carla Seatzu [a].

[a] *Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy.*

[b] *Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296,13397, Marseille, France.*

Abstract

In this paper we propose a novel algorithm to solve the discrete consensus problem, i.e., the problem of distributing evenly a set of tokens of arbitrary weight among the nodes of a networked system. Tokens are tasks to be executed by the nodes and the proposed distributed algorithm minimizes monotonically the makespan of the assigned tasks. The algorithm is based on *gossip*-like asynchronous local interactions between the nodes. The convergence time of the proposed algorithm is superior with respect to the state of the art of discrete and quantized consensus by at least a factor $\mathcal{O}(n)$ in both theoretical and empirical comparisons.

Key words: Discrete consensus, quantized consensus, gossip algorithms, distributed task assignment, multi-agent systems, networked systems, distributed optimization.

Published as:

Mauro Franceschelli, Alessandro Giua, Carla Seatzu, Fast discrete consensus based on gossip for makespan minimization in networked systems, Automatica, Volume 56, June 2015, Pages 60-69, ISSN 0005-1098.

⋆ Mauro Franceschelli is corresponding author. This research has been partially funded by the Artemis JU (GA number 333020) as part of the ACCUS project and by Regione Sardegna, LR 7/2007 (call 2010) under project SIAR (CRP-24709).

Email addresses: `mauro.franceschelli@diee.unica.it` (Mauro Franceschelli), `alessandro.giua@lsis.org,giua@diee.unica.it` (Alessandro Giua), `seatzu@diee.unica.it` (Carla Seatzu).

1 Introduction

The problem of *quantized consensus* consists in the design of a decentralized algorithm to steer a set of quantized state variables toward a common value. One of the early formulations of the quantized consensus problem was in [1] where the issue of quantization to implement consensus algorithms was brought to attention and a solution inspired by distributed load balancing of quantized indivisible tasks was proposed [2–5]. Since then, several approaches were developed to study the issues of quantization for the consensus problem [6–11]. All these approaches consider undirected connected graphs with the exception of [7] where the quantized consensus on directed graphs is investigated.

In [10] we proposed a generalization of the quantized consensus problem called *discrete consensus*, i.e., the problem of distributing evenly a set of indivisible tokens of different weight. Thus, quantized consensus is a special case of discrete consensus where all tokens are of equal weight.

The first approaches to the study of the convergence time of quantized consensus problems [1,8,9] are based on the computation of the average meeting time between two random walks in a graph. This derives from the necessity of swapping loads – that cannot be balanced due to their discrete nature – between nodes. In [11] we proposed an efficient algorithm for discrete consensus that executes swaps along a preassigned direction thus significantly reducing the expected convergence time. This approach required the existence in the graph of a known Hamiltonian cycle and only adjacent nodes along this cycle may swap their loads. However, in many applications the requirement that a Hamiltonian cycle exists is not satisfied and furthermore, even when it exists, the computation of such a cycle may not be feasible due to the network size or the absence of global information.

In this paper, we propose a novel decentralized algorithm based on gossip to solve the *discrete consensus* problem. The proposed algorithm improves the convergence time of previous approaches, including [11], and in particular it converges in linear time with respect to the number of nodes for graphs of given diameter. Furthermore, it does not make any assumption on the structure of the network topology which is considered unknown. The proposed approach can be used in all applications discussed in the quantized consensus literature, such as the distributed averaging in sensor networks. Furthermore, it can also be applied in a wider setting that includes problems such as distributed load balancing or task assignment in networks of processors or mobile robots, where a set of indivisible tasks of different size is to be assigned so that the maximum execution time is minimized, exploiting only local state information exchange between the processing units.

Summarizing, the contributions of this paper are the following.

- A novel algorithm based on gossip that solves the discrete consensus problem is proposed.
- We compute an upper bound for the expected convergence time, showing that it is $\mathbb{E}\left[T_{conv}\right] \approx \mathcal{O}\left(n\right) d(\mathcal{G})$ in normalized time units, which improves the state of the art with respect to the mentioned literature.
- With respect to [11], we remove all assumptions on the structure of the network topology which is only required to be an undirected connected graph.
- The proposed algorithm has an embedded stopping criterion so that token exchanges stop within a finite time interval after the *discrete consensus* state has been achieved.
- Simulations are provided to compare the theoretical upper bound to the convergence time with the simulated convergence time for random graphs and line graphs with increasing number of nodes. An empirical comparison of convergence time between Fast Discrete Consensus and Discrete Consensus algorithms is provided to highlight the increased performance with respect to the state of the art.

Related works

In [6] the quantized consensus problem is formulated with nodes with continuous states but capable of transmitting only a finite number of symbols. The authors study algorithms in the framework of randomized gossip algorithms [12] by applying deterministic and probabilistic uniform quantizers with a combination of three local state update rules defined as partially quantized, totally quantized and compensating. They proved that, with a totally quantized state update rule, finite time consensus is achieved almost surely but the initial states' average is not preserved. On the other hand, a compensating update rule that preserves the initial state average at each iteration does not guarantee to reach consensus. In [6] and [12] the state variables in the networks are quantized but they are not considered as indivisible tokens or tasks as in this paper.

All the above mentioned works investigate the quantized consensus problem in undirected graphs. Notably, in [7] a quantized consensus algorithm for networks described by directed graphs is proposed. The authors design a local state update strategy that applied randomly by the nodes achieves quantized consensus exploiting only mono-directional quantized communications while preserving the average of the initial states during the updates.

The issue of good characterization of the convergence time of quantized consensus algorithms is investigated in [9] and [8]. In [8] the quantized consensus problem is investigated by considering nodes with arbitrarily quantized states. For the special case of uniform quantization, bounds on the convergence time related to the spectral properties of the principal sub-matrices of the Laplacian matrix of the graph describing the network are given. In [9] the quantized consensus problem for nodes with uniformly quantized states is investigated and a general formulation for the convergence time of a class of distributed quantized consensus algorithms is presented. The derivation is based on the computation of meeting times of random walks in a graph that can be of fixed or time-varying topology. The proposed bounds on convergence times are all polynomial in the number of nodes.

In [10] the discrete consensus problem is introduced, it differs from the other approaches in that it considers the state of each node as composed by a set of indivisible tasks with different weight, size or cost, to be exchanged by the nodes with different execution speed. In [11] an upper bound on the convergence time of the discrete consensus problem presented in [10] is improved by a factor $\mathcal{O}(n)$, where n is the number of nodes. In [13] the problem of distributed task assignment is formalized as a distributed consensus algorithm. The authors consider tasks of different cost and type which can be executed only by subsets of nodes in the network. Furthermore, the authors consider a constraint on the maximum number of tasks executable by each single node and prove that almost surely the task assignment converges to a feasible and time-invariant configuration. With respect to this paper, in [13] task execution constraints and tasks with node-dependent weight are considered. On the other hand, their characterization of the convergence time is preliminary and requires several working assumptions on the network topology which do not allow to guarantee the convergence properties in general undirected graphs. In [14] the distributed task assignment in a network of heterogeneous mobile robots with heterogeneous tasks is investigated. The authors exploit gossip based local optimizations to both assign tasks located in a plane and compute optimized routes for the robots.

Finally, we mention that in [15] a variant to the load balancing problem on multi-processors systems called *token distribution* problem is investigated. The token distribution problem can be addressed as a discrete consensus problem. In [15] the token distribution problem is addressed by assuming that all the nodes know the average load of the network and compute in a decentralized way the total number of nodes. Furthermore the network topology is assumed to be a known connected tree. In our work we do not assume that nodes have access to any kind of global information about the network state.

The paper is structured as follows. In Section 2 the problem of discrete consensus is formalized. In Section 3 the proposed algorithm is presented. In Section 4 the convergence properties are characterized. In Section 5 the expected convergence time is characterized. In Section 6 simulations are provided to corroborate the theoretical results. In Section 7 final remarks are discussed.

2 Problem Statement

Consider a network of n nodes whose connections can be described by an undirected connected graph $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of nodes and $E \subseteq \{\mathcal{V} \times \mathcal{V}\}$ is the set of edges that represent the existence of a communication link. We consider K indivisible tokens to be assigned to the nodes, and a weight $c_j \in \mathbb{N}^+$, $j = 1, \ldots, K$, is associated with each token. We define a weight vector $c \in \mathbb{N}^K$ whose j-th component is equal to c_j and n binary vectors $y_i \in \{0,1\}^K$ such that $y_{i,j} = 1$ if the j-th token is assigned to node i, $y_{i,j} = 0$ otherwise. Finally, $c_{\max} = \max_{j=1,\ldots,K} c_j$. To each node $i \in \mathcal{V}$ is allocated a *load* $x_i = c^T y_i$ consisting of the sum of the weight of tokens assigned to node i.

Denoting $Y(t) = [y_1(t)\ y_2(t)\ \ldots\ y_n(t)]$ the state of the network at time t, we wish to minimize the maximum load in the network, which may equivalently correspond to a maximum execution time or makespan. An optimal solution to this problem requires complete knowledge of the network state and the solution of an integer programming problem with binary variables

$$\begin{cases} \min & F(Y) = \max_{i \in \mathcal{V}} c^T y_i \\ s.t. & Y\mathbf{1} = \mathbf{1} \\ & y_{i,j} \in \{0,1\} \quad \forall i \in \mathcal{V}, \;\; j = 1, \ldots, K. \end{cases} \tag{1}$$

Since in our framework we consider networks with large number of nodes and tokens, the computational complexity of the integer programming problem (1) can be extremely high. In particular, Problem 1 is a formulation of the makespan minimization problem on identical parallel machines. The complexity of finding an exact solution to this problem is known to be NP-hard (see [16] for reference). Furthermore, it requires a centralized coordinator with full knowledge of the network state and ability to communicate with all the nodes. To address these disadvantages we look for a distributed solution based only on randomized local state updates with limited available information. Exploitation of randomized and asynchronous local state updates simplifies greatly the network design since it avoids the need for synchronization in a large network and does not require knowledge of the network topology. The assumption that the network topology is unknown is needed in large scale systems and applications which involve networks of mobile robots, whose pattern of interaction may be time-varying. The necessary tradeoff to address this problem in a distributed and randomized fashion consists in not being able to guarantee an optimal solution. On the other hand, in this paper we are able to characterize an *absolute performance guarantee* which does not depend on the network size. This can be achieved by aiming at finding an assignment which belongs to the set

$$\begin{aligned} \mathcal{Y} = \{Y = [y_1 \; \ldots \; y_n] \;\; | \;\; \left|c^T y_i - c^T y_j\right| \leq c_{\max}, \\ \forall i,j \in \mathcal{V}\}\,. \end{aligned} \tag{2}$$

We say that *discrete consensus* is achieved when the state of all nodes in the network satisfies condition (2). There are several reasons for defining the discrete consensus set as in eq. (2). Consider the following cases:

- If all tokens have equal weight, it is easy to show that when the number of tokens is not a multiple of the number of agents then, due to the discrete nature of the tokens, the optimal assignment is one in which the maximum difference between the agents' loads is exactly $c_{min} = c_{max}$.
- If set $\mathcal{Y}$ is defined via a constant smaller than the maximum token weight, then the considered problem may not be feasible and a network state inside $\mathcal{Y}$ may not be reached even by solving a centralized optimization problem with full knowledge of the network state and structure. To prove this, consider the case in which the number of tokens is less than the number of nodes. In this case, the maximum load difference between nodes in the network cannot be smaller than c_{max} since there exist empty nodes.
- An optimal solution to problem (1) may not be found if we are constrained to exploit only gossip based pairwise optimizations, as shown in [17]. On the contrary, set $\mathcal{Y}$ in eq. (2) can be reached through pairwise optimizations as proven in this paper.

By denoting Y^* the optimal solution of problem (1), $F(Y^*)$ the value of the objective function for an optimal solution of (1) and $F(Y)$ the value of the objective function for an assignment $Y \in \mathcal{Y}$ within the set of discrete consensus, it can be shown (see [18]) that

$$\frac{\sum_{j=1}^{K} c_j}{n} \leq F(Y^*) \leq F(Y) \leq \frac{\sum_{j=1}^{K} c_j}{n} + c_{max}, \quad Y \in \mathcal{Y}.$$

Therefore, a solution $Y \in \mathcal{Y}$ represents an absolute performance guarantee with bounded error which does not depend on the size of the network since

$$F(Y) \leq F(Y^*) + c_{max}, \qquad Y \in \mathcal{Y}.$$

It is easy to show (see [1]) that if all tokens have the same weight and $Y \in \mathcal{Y}$ either $F(Y^*) = F(Y) = \lfloor \frac{\sum_{j=1}^{K} c_j}{n} \rfloor + c_{max}$ or $F(Y^*) = F(Y) = \lfloor \frac{\sum_{j=1}^{K} c_j}{n} \rfloor$, therefore in this case a solution $Y \in \mathcal{Y}$ corresponds to an optimal solution. If all tokens have the same weight, our problem is formally equivalent to that of *quantized consensus* [1].

In the following we denote an arbitrary node as the *sink* node and without loss of generality we assume that it is the node with id $i = 1$. The sink node is chosen prior to the execution of our algorithm. There are several ways and methods to choose a node arbitrarily in a network in a distributed way. One is for instance to let every node extract a number at random and then execute a consensus on the maximum value algorithm which converges in a finite number of steps less then the diameter of the network. Since the selection of such node is not related to our major contribution, we consider the existence of a sink node as a working assumption, similar to the existence of a leader in a leader-follower network.

Let d_i be the length of the shortest path from node i to the sink node, i.e., node 1. Note that $d_1 = 0$ and $d_i > 0$ for $i \neq 1$.

Definition 2.1 *Given a graph $\mathcal{G} = (\mathcal{V}, E)$ we call* depth of the graph with respect to the sink node*, or shortly* depth*, the length of the greatest* distance d_i *from any node i and the sink node, i.e., $d(\mathcal{G}) = \max_{i \in \mathcal{V}} d_i$.*

Obviously, for any choice of the sink node the depth is always smaller than or equal to the diameter of $\mathcal{G}$.

3 Proposed Algorithm

In this section we present the main result of the paper, i.e., the *Fast Discrete Consensus* (FDC) algorithm. We first present a heuristic able to partition a set of tokens into two balanced sets with linear complexity, then we describe a local state update rule which exploits such heuristic and specifies how two given nodes should cooperate to exchange tokens. Finally, we present the FDC algorithm, based on the proposed local state update rule, together with an intuitive explanation of how it works and an example of execution.

To each node $i \in \mathcal{V}$ we associate the three state variables $x_i(t), z_i(t), h_i(t)$ defined in Table 1. Let $\mathcal{K}_i(t)$ be the set

$x_i(t)$	Load associated with the i-th node at time t (sum of token weights assigned to node i).
$z_i(t)$	Local estimation of the sink node load (node 1) at time t of node i.
$h_i(t)$	Local estimation of the distance d_i between node i and the sink node (node 1).

Table 1
Notation table

of indices of tokens assigned to node i at time t. In Algorithm 1 it is presented a simple balancing rule to average the load of two nodes incident on the selected edge. Variations of this simple and widely known heuristic have been investigated in the context of load distribution between two parallel machines and as polynomial time approximation of the 2-partitioning problem [19]. This rule computes two partitions of tokens $\mathcal{K}_a$, $\mathcal{K}_b$ given two sets of tokens $\mathcal{K}_i$ and $\mathcal{K}_j$.

Remark 3.1 *The balancing rule in Algorithm 1 is completed in $|\mathcal{K}|$ steps, thus with linear complexity with respect to the number of tokens contained in nodes i and j. This heuristic is a slight modification of the very well known Johnson's algorithm for the 2-machine N job problem [20]. This heuristic has the advantage of being extremely simple and easy to implement while guaranteeing that the maximum difference in cost between the two sets of tokens given as output is at most equal to c_{max}, which is sufficient for our purposes.*

Let us now introduce a local state update rule which exploits Algorithm 1 to locally balance the tokens between pairs of nodes, such a rule is presented in Algorithm 2. In the following we denote $x_i(t_k) = c^T y_i(t_k)$ the sum of the costs of tasks assigned to node i at time t_k.

In simple words, in Step 2 of Algorithm 2 if the load difference between two nodes is bigger than c_{max} a new partition of tokens between the two nodes is computed using Algorithm 1. In Step 3 the node j estimated to be furthest from the sink node updates its estimate of the sink node load by adopting the estimate of node i, which is assumed to be closer to the sink. Node i keeps its estimate of the sink load unaltered. If the sink node itself is involved, then both nodes update their estimation with the actual value of the load of the sink node. Furthermore, in Step 3 the shortest distance between nodes i and j from the sink node is estimated. In particular, during each update each node verifies to be either the closest one to the sink node, and then leaves its distance estimation unchanged, or the furthest, and

Algorithm 1. Balancing rule

```
Input   : Sets $\mathcal{K}_i$ and $\mathcal{K}_j$.
Output: Sets $\mathcal{K}_a$ and $\mathcal{K}_b$
1 Initialize: Let $\mathcal{K} = \mathcal{K}_i \cup \mathcal{K}_j$, let $\mathcal{K}_a := \emptyset$ and $\mathcal{K}_b := \emptyset$.
2 while $\mathcal{K} \neq \emptyset$ do
    let $\delta := \text{argmax}_{j \in \mathcal{K}} c_j$;
    if $\sum_{r \in \mathcal{K}_a} c_r \leq \sum_{r \in \mathcal{K}_b} c_r$ then
        let $\mathcal{K}_a := \mathcal{K}_a \cup \{\delta\}$,    $\mathcal{K}_b := \mathcal{K}_b$;
        (if the load of $\mathcal{K}_b$ is greater than that of $\mathcal{K}_a$, then assign the token to $\mathcal{K}_a$)
    else
        let $\mathcal{K}_a := \mathcal{K}_a$,    $\mathcal{K}_b := \mathcal{K}_b \cup \{\delta\}$.
    $\mathcal{K} := \mathcal{K} \setminus \{\delta\}$.
3 if $\left| \sum_{j \in \mathcal{K}_a} c_j - \sum_{j \in \mathcal{K}_b} c_j \right| > \left| \sum_{r \in \mathcal{K}_i} c_r - \sum_{r \in \mathcal{K}_j} c_r \right|$,
(A partition less balanced than the one given as input is found) then
    let $\mathcal{K}_a := \mathcal{K}_i$,    $\mathcal{K}_b := \mathcal{K}_j$.
```

Algorithm 2. Local State Update Rule

```
Input   : An edge $(i, j) \in E$, variables $z_i, z_j$ and sets $\mathcal{K}_i$ and $\mathcal{K}_j$. Without loss of generality, assume in the following
          that estimated distance $h_i$ between node $i$ and the sink node 1 is smaller than or equal to that of node $j$.
Output: $z_i, z_j$, $h_i, h_j$, sets $\mathcal{K}_i$ and $\mathcal{K}_j$ which correspond to the new assignment of tokens and variables to node $i$ and
          node $j$.
1 Initialize: Let $x_a, x_b$ be respectively the load of partitions $\mathcal{K}_a, \mathcal{K}_b$.
2 if $|x_i(t_k) - x_j(t_k)| > c_{max}$ then
    compute sets $\mathcal{K}_a$ and $\mathcal{K}_b$ with the Balancing Rule in Algorithm 1
else
    let $\mathcal{K}_a = \mathcal{K}_i$ and $\mathcal{K}_b = \mathcal{K}_j$.
3 if $i = 1$ (sink node) then
    $z_j := x_i$.
else
    $z_j := z_i$, $h_j := min(h_j, h_i + 1)$.
4 if $z_i - x_a > c_{max}$ or $z_i - x_b > c_{max}$ then
    assign the smallest load between $\mathcal{K}_a$ and $\mathcal{K}_b$ to node $i$ and assign the largest load to node $j$.
else
    assign the largest load between $\mathcal{K}_a$ and $\mathcal{K}_b$ to node $i$ and assign the smallest load to node $j$.
```

then updates its current estimate with that of the neighbor plus one. Finally, in Step 4, node loads are ordered in decreasing order with respect to their presumed distance from the sink node unless the load difference between the estimated load of the sink node in variable z_i and the load in the computed partitions $\mathcal{K}_a$ and $\mathcal{K}_b$ is greater than c_{max}. In this case, the partition with the smallest load is assigned to the node closer to the sink. By iterating this behavior it is possible to ensure that if the network is globally unbalanced, in finite time a balancing involving the sink node which improves the considered global objective function will occur.

The main contribution of this paper is presented in Algorithm 3.

The stopping criterion of Algorithm 3 is discussed in Proposition 5.5, where it is shown that after *discrete consensus* has been achieved there exists a maximum interval of time after which no more load exchanges between nodes may happen. Algorithm 3 iterates on the number of selected edges. This implies that several edges may perform the state update rule simultaneously. If a state update fails because the selected edge is incident on a node which is busy performing a state update with another node, then the edge selection can simply be discarded and the algorithm execution proceeds with further edge selections until a feasible one is found. The proposed algorithm is robust against communication failures because it is asynchronous and is not based on the order in which state updates are performed. Moreover, Algorithm 3 exploits only locally available information at each step. Next we show an example

```
Algorithm 3. Fast Discrete Consensus (FDC)
Input  : Sets $\mathcal{K}_i$ for $i = 1, \dots, n$ ( Tokens are assigned arbitrarily to nodes).
Output: Updated sets $\mathcal{K}_i$ for $i = 1, \dots, n$.
1 Initialize: Let $k = 0$, $t_k = 0$, $h_1(t_k) = 0$, $z_1(t_k) = x_1(t_k)$, $h_i(t_k) = \infty$, $z_i(t_k) = 0$ for $i = 2, \dots, n$. Without loss of generality, let node 1 be the sink node.
2 while TRUE do
    3 Let $t = t_k$.
    4 A random node selects an edge $(i, j)$ according to a given stochastic selection process.
    5 Update the state of nodes $i$ and $j$ according to the Local State Update Rule in Algorithm 2.
    6 Let $k = k + 1$.
```

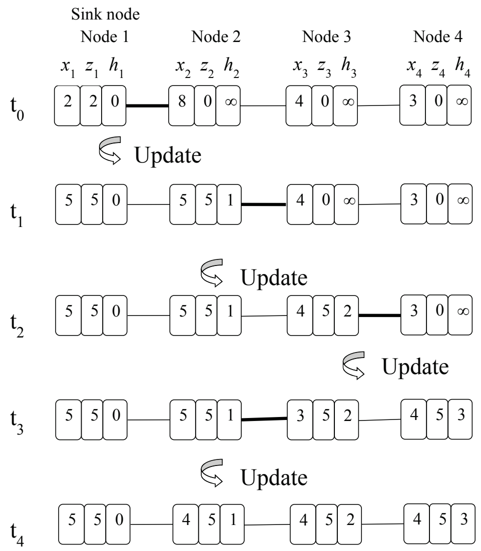

Fig. 1. Example of execution of Algorithm 3 described in Example 3.2.

of execution of Algorithm 3 to further clarify its functioning.

Example 3.2 *Consider the network of* 4 *nodes depicted in Fig. 1. Nodes are labeled from* 1 *to* 4*. The sink node is node* 1*. Thus, node* 2 *has distance* 1 *with respect to the sink node, node* 3 *has distance* 2 *and node* 4 *has distance* 3*. In this example for sake of clarity we consider tokens of unit weight $c_j = 1$ for $j = 1, \dots, K$, thus representing a standard quantized consensus problem. The initial state of the network is at $t = t_0$: $x_1(t_0) = 2$, $x_2(t_0) = 8$, $x_3(t_0) = 4$, $x_4(t_0) = 3$. All variables containing the estimation of the sink load are initialized to zero except for the sink node, thus $z_1(t_0) = 2$, $z_2(t_0) = z_3(t_0) = z_4(t_0) = 0$. The estimated distance from the sink node is set to $h_1 = 0$ by the sink node and $h_i = \infty$ by all other nodes. In this example we consider an arbitrary edge selection sequence for sake of clarity.*

(1) At t_0 edge $(1, 2)$ is selected. Since $|x_1 - x_2| > c_{max}$, the nodes balance their load leading to $x_1 = x_2 = 5$. Then, they update their estimations of the sink load that are set to $z_1 = z_2 = 5$ and the estimated distance of node 2 *is set to $h_2 = 1$.*
(2) At t_1 edge $(2, 3)$ is selected. Since $|x_2 - x_3| \le c_{max}$ and the largest load is already the closest to the sink therefore the nodes update only the estimates to $z_2 = 5$ and $z_3 = 5$ and the estimated distance of node 3 *is set to $h_3 = 2$.*
(3) At t_2 edge $(3, 4)$ is selected. Since $|x_3 - x_4| \le c_{max}$ no balancing occurs but estimations are updated as $z_3 = 5$ and $z_4 = 5$. Furthermore, since $z_4 - x_4 > c_{max}$ the smallest load is put closer to the sink. The estimated distance of node 4 *is set to $h_4 = 3$.*
(4) At t_3 edge $(2, 3)$ is selected. Since $|x_2 - x_3| > c_{max}$ the nodes balance their load.
(5) At t_4 the discrete consensus condition (2) *is reached.*

4 Convergence Properties

In this section we study the convergence properties of Algorithm 3. In particular, we now formally define a *stochastically persistent* edge selection process. In the following, without loss of generality, let node 1 be the sink node.

Definition 4.1 (Edge Selection Process) *An edge selection process $\mathfrak{e} : \mathbb{R}^+ \times E \to \{0, 1\}$ maps each time instant $t \in \mathbb{R}^+$ and each edge $(i, j) \in E$ to a binary value: if $\mathfrak{e}(t, (i, j)) = 1$ then edge (i, j) is active at time t, not active otherwise.*

We denote as $\hat{E}(t, t+T) = \{(i, j) \in E : \mathfrak{e}(\tau, (i, j)) = 1 \text{ for some } \tau \in [t, t+T]\}$, the set of edges selected at least once during the time interval $[t, t+T]$. We now recall a definition that has been used in [14] to prove almost sure finite time stability in gossip based vehicle routing problems.

Definition 4.2 (Stochastic persistence) *An edge selection process $\mathfrak{e}$ is said to be* stochastically persistent *if $\forall t \geq 0$ there exist a finite $T > 0$ and a probability $p \in (0, 1)$ such that*

$$Pr(\hat{E}(t, t+T) = E) \geq p \tag{3}$$

where $Pr(\cdot)$ denotes a probability.

Stochastic persistence implies that, if we consider a finite but sufficiently large time interval, then each edge has a probability greater than or equal to a finite value p of being selected during such an interval. In particular, the case in which $p = 1$ is called *deterministic persistence*. Deterministic persistence implies that, if we consider a finite but sufficiently large time interval, then for sure all edges are selected at least once during such interval. Deterministically persistent edge selection processes can be easily implemented in a networked system if algorithms for distributed time synchronization are implemented as well. We refer the reader to the works by J. He *et al.* [21,22] for a representative example of distributed time synchronization approaches based on consensus on the maximum value algorithms.

Definition 4.3 *Given a stochastically persistent edge selection process we define a continuous random variable τ that represents the smallest interval of time in which $\hat{E}(t, t+\tau) \equiv E$.*

An upper bound to the expected value $\mathbb{E}[\tau]$ of the random variable τ can be computed directly from Definition 4.2 of stochastically persistent edge selection process as $\mathbb{E}[\tau] \leq \sum_{i=1}^{\infty} iTp(1-p)^{i-1}$. The particular case in which the edge selection process is *deterministically persistent*, clearly implies that $\tau \leq T$ with probability $p = 1$ and thus $\mathbb{E}[\tau] \leq T$. Next, we state a definition of convergence time.

Definition 4.4 *We define convergence time the quantity*

$$T_{conv} = inf\left\{T \in \mathbb{R}^+ : Y(t) \in \mathcal{Y}, \quad \forall t \geq T\right\}.$$

Thus, according to Definition 4.4, the convergence time T_{conv} is the minimum time it takes for the network to reach the discrete consensus set $\mathcal{Y}$.

In the following, let τ_i, $i = 1, \ldots, k$, represent k realizations of the continuous random variable τ defined as in Definition 4.3. Now, let us present two propositions needed to prove our main result. We first show that, in finite time, each node estimates correctly its distance with respect to the unknown sink node.

Proposition 4.5 *Consider a stochastically persistent edge selection process $\mathfrak{e}$. Let $T_{d(\mathcal{G})} = \sum_{i=1}^{d(\mathcal{G})} \tau_i$. Let d_i be the minimum distance of node i from the unknown sink node and let $h_i(t)$ be its estimated distance at time t. Let t_0 be the initial instant of time. It holds*

$$h_i(t) = d_i(t) \qquad \forall t \geq T_{d(\mathcal{G})} + t_0, \quad \forall i \in \mathcal{V}.$$

Proof: See Appendix A. □

We now consider the case in which the load of the sink node does not change for some time. Proposition 4.6 characterizes in what interval of time the generic node estimates correctly the load of the sink node under such assumption.

Proposition 4.6 *Consider a stochastically persistent edge selection process $\mathfrak{e}$. Let $T_{d(\mathcal{G})} = \sum_{i=1}^{d(\mathcal{G})} \tau_i$. If the sink node's load $x_1(t) = x_1$ for $t \in [t_1, t_2]$ with $t_2 - t_1 \geq T_{d(\mathcal{G})}$, then*

$$z_i(t) = x_1 \qquad \forall t \in \left[t_1 + T_{d(\mathcal{G})}, t_2\right], \quad \forall i \in \mathcal{V}.$$

Proof: See Appendix B. □

The following proposition guarantees that if the number of nodes whose load is maximum in the network remains constant for a sufficiently long time, then the sink node within a given time interval is guaranteed to hold the maximum load in the network. In the next result, $2d(\mathcal{G})$ occurrences of the continuous random variable τ are necessary as opposed to $d(\mathcal{G})$ as in Proposition 4.6 because two processes involving both the dynamics of variables z_i and the dynamics of the nodes which hold the maximum value in the network need to be considered.

Proposition 4.7 *Consider a stochastically persistent edge selection process $\mathfrak{e}$. Let $T_{2d(\mathcal{G})} = \sum_{i=1}^{2d(\mathcal{G})} \tau_i$. If in a time interval $[t_1, t_2]$ with $t_2 - t_1 \geq T_{2d(\mathcal{G})}$, both the maximum load and the number of nodes with the maximum load remain constant, then*

$$x_1(t) = \max_{i \in \mathcal{V}} x_i(t) \qquad \forall t \in \left[t_1 + T_{2d(\mathcal{G})}, t_2\right].$$

Proof: See Appendix C. □

We now prove that, in the case of a stochastically persistent edge selection process, Algorithm 3 converges almost surely in finite time to discrete consensus. The proof of Theorem 4.8 is based on the results introduced in Propositions 4.5, 4.6 and 4.7. These results are exploited to state an upper bound to the interval of time between two consecutive decrements of the maximum load in the network until the discrete consensus condition is achieved. The basic idea is to consider the scenario in which by chance no decrement of the maximum load occurs and thus show that after a sufficiently long but finite interval of time, almost surely i) every node is aware of its distance with respect to the unknown sink node (Proposition 4.5), ii) the sink node holds the maximum load in the network (Proposition 4.7), iii) every node correctly estimates the actual load of the sink node (Proposition 4.6). As soon as the sink node holds the maximum load, the smallest load in the network is ensured to filter through the network up to the sink node in finite time to allow a decrement of the maximum load. The occurrence of these events ensures in Theorem 4.8 that if a decrement of the maximum load is possible then it will occur in the characterized interval of time until the discrete consensus condition is achieved.

Theorem 4.8 *Consider a network $\mathcal{G}$ of n nodes that executes Algorithm 3. If the edge selection process $\mathfrak{e}$ is stochastically persistent, then*

$$Pr\left(\exists T_{conv} \in \mathbb{R}^+ \ : \ \forall t \geq T_{conv}, \quad Y(t) \in \mathcal{Y}\right) = 1$$

where $\mathcal{Y}$ is defined as (2) and $Pr\left(\cdot\right)$ denotes a probability.

Proof: See Appendix D. □

5 Expected convergence time

In this section we determine the expected convergence time of the Algorithm 3 when edges are selected according to a stochastically persistent process.

Proposition 5.1 *Let τ be the continuous random variable introduced in Definition 4.3. If the edge selection process $\mathfrak{e}$ is stochastically persistent, then*

$$\mathbb{E}\left[T_{conv}\right] \leq (4(M-m)(n-1)+1)d(\mathcal{G})\mathbb{E}\left[\tau\right], \tag{4}$$

where $M = \|x(0)\|_\infty$, $m = \frac{\mathbf{1}^T x(0)}{n}$ and $\mathbb{E}\left[\cdot\right]$ denotes the expected value.

Proof: The proof is carried out in three steps.

(a) Following the arguments of the proof of Theorem 4.8, due to Proposition 4.5 after $T_{d(\mathcal{G})}$ units of time every node estimates correctly its distance from the unknown sink node, therefore the corresponding expectation is $\mathbb{E}\left[T_{d(\mathcal{G})}\right] \leq d(\mathcal{G})\mathbb{E}\left[\tau\right]$. From this instant of time, the maximum number of improvements (decrements) of $V(t) = \|c^T Y(t)\|_\infty = \|x(t)\|_\infty$ occurs in an interval of time at most equal to $T_{conv} \leq \sum_{i=1}^{4N(n-1)d(\mathcal{G})} \tau$, where N is the maximum number of improvements of function $V(t)$, needed by any realization of Algorithm 3 to reach the set $\mathcal{Y}$, starting from any initial token assignment.

(b) We now prove that $N \leq M - m$. By definition $M - m$ is the difference between the maximum load and the average load at the initial configuration. Therefore, since the smallest decrement of $V(t)$ is equal to one, the maximum number of decrements of $V(t)$ is less or equal to $M - m$.

(c) Finally, if edges are selected by independent and identically distributed (i.i.d.) stochastic processes, the expected convergence time depends on the expected value of T_{conv} which can be bounded as

$$\begin{aligned}\mathbb{E}\left[T_{conv}\right] &\leq (M-m)\mathbb{E}\left[T_{4(n-1)d(\mathcal{G})}\right] + \mathbb{E}\left[T_{d(\mathcal{G})}\right] \\ &\leq (4(M-m)(n-1)+1)d(\mathcal{G})\mathbb{E}\left[\tau\right].\end{aligned}$$

□

The next corollary shows that if the initial load is distributed in the network in such a way that the number of tokens is distributed evenly, for instance the case in which in the starting configuration each node contains at most $\lceil \frac{k}{n} \rceil$ tokens, then it is possible to state an upper bound to the expected convergence time which is better with respect to the general case by a factor $\mathcal{O}(n)$.

Corollary 5.2 *Let k be the total number of tokens. Assume that tokens are initially distributed at random in the network so that the node with the greatest number of tokens has at most $\lceil \frac{k}{n} \rceil$ of them. If the edge selection processes $\mathfrak{e}$ are i.i.d. and stochastically persistent, then*

$$T_{conv} \leq \left(4\left(\left(c_{max} - c_{min}\right)k + c_{max}\right) + 1\right)d(\mathcal{G})\mathbb{E}\left[\tau\right] \tag{5}$$

Proof: The proof follows from Proposition 5.1. If the maximum initial load consists in $\lceil \frac{k}{n} \rceil$ tokens of arbitrary size, then parameters M and m are respectively bounded by $M \leq c_{max}\frac{k+1}{n}$ and $m \geq c_{min}\frac{k}{n}$. Therefore, since $\frac{n-1}{n} < 1$, we can manipulate the bound in eq. (6) into eq.(5). □

Since Algorithm 3 consists only of local and asynchronous state updates between nodes, we now consider the case where there exists an independent stochastic process for each edge in the network that in parallel governs the activation of each edge.

Corollary 5.3 *Let τ_e be a random variable that represents the time interval between two consecutive selections of a given edge by its own stochastic process. If edges are selected according to i.i.d.* stochastically persistent *processes, then*

$$\mathbb{E}\left[T_{conv}\right] \leq (4(M-m)(n-1)+1)d(\mathcal{G})\mathbb{E}\left[\tau_e\right], \tag{6}$$

where $M = \|x(0)\|_\infty$, $m = \frac{\mathbf{1}^T x(0)}{n}$, and $\mathbb{E}\left[\cdot\right]$ denotes the expected value.

Proof: The proof follows from Proposition 5.1 and the fact that we are considering i.i.d. *stochastically persistent* processes to activate each edge. This implies that the average time for the selection of all edges is equal to the average interval of time for activation of any given edge, i.e., the expected value of random variable τ_e corresponds to the expected value of random variable τ in Definition 4.3. □

We now consider as an example the case where edges are chosen according to Poisson stochastic processes with parameter λ_e, which is a stochastically persistent process. The probability that a given edge $e \in E$ is selected k times in the unit of time is $P\left[k\right] = e^{-\lambda_e}\frac{\lambda_e^k}{k!}$, $k = 0, 1, \ldots$. As a consequence the continuous random variable τ_e defining the waiting time among two consecutive selections of the same edge follows an exponential distribution and its expected value is $\mathbb{E}\left[\tau_e\right] = \frac{1}{\lambda_e}$. Summarizing, the expected convergence time of Algorithm 3 if edges are selected according to a Poisson process with parameter λ_e is upper bounded by

$$\mathbb{E}\left[T_{conv}\right] \leq 4(M-m)(n-1)\frac{d(\mathcal{G})}{\lambda_e} \approx \mathcal{O}\left(n\right)d(\mathcal{G}). \tag{7}$$

Let us now define the instant of time T_{stop} in which no further token exchanges between nodes may occur.

Definition 5.4 *We define* stopping time *the quantity $T_{stop} \in \mathbb{R}^+$ such that it holds*

$$T_{stop} = inf \left\{ T \in \mathbb{R}^+ : Y(t) = Y(T), \quad \forall t \geq T \right\}.$$

The next proposition characterizes the existence of a *stopping time* as in Definition 5.4.

Proposition 5.5 *If the edge selection process $\mathfrak{e}$ is stochastically persistent, then*

$$P\left(\exists T_{stop} : \forall t \geq T_{stop}, \quad Y(t) = Y(T_{stop})\right) = 1.$$

Proof: By Theorem 4.8 after at most T_{conv} units of time the network reaches discrete consensus. Using the same arguments as in the proof of Theorem 4.8, by Propositions 4.6, and 4.7 at most at time $T_{conv} + T_{4d(\mathcal{G})}$ the sink node holds the maximum load of the network and each node has a correct estimation of the sink load. At this point, no more balancing may occur since no pair of nodes has load difference less than c_{max} and therefore the local balancing rule is never triggered. Thus, according to the local state update rule, the loads are ordered in decreasing order from largest (in the sink node) to smallest (in nodes at maximum distance). It is easy to show that after at most $T_{d(\mathcal{G})^2}$ units of time, corresponding to the time it takes to order a set of $d(\mathcal{G})$ loads with random swaps of a pair of loads, all nodes reach a local equilibrium in which no condition of the local state update rule that motivates a load exchange is triggered. Therefore, $T_{stop} \leq T_{conv} + T_{d(\mathcal{G})^2}$. Since T_{conv} and $T_{d(\mathcal{G})}$ exist almost surely with unitary probability the statement is proved. $\square$

6 Numerical simulations

A series of numerical simulations has been carried out to compare the convergence time of Algorithm 3 resulting from numerical simulations with the discrete consensus algorithm [10] which generalizes quantized consensus [1]. Simulations have been performed by considering tokens with random size between 1 and 10, thus $c_{max} = 10$. Each token is initially assigned to a random node with uniform probability. Two graphs topologies have been considered to highlight the scalability of the algorithm with respect to the number of nodes: random graphs and line graphs. For random graphs, the probability of edge existence between any pair of nodes has been chosen to be $p = 1.1\frac{ln(n)}{n}$, where n is the number of nodes and $ln(n)$ is the natural logarithm. This choice was made to ensure a high chance of graph connectivity with high probability of a diameter equal to 6. Simulations have been repeated 100 times for graphs size from 10 to 200 nodes. For each simulation a total of $K = 4n$ tokens have been considered so that the average load is kept constant for different network sizes. Figure 2 shows the average convergence time, in number of local updates, of FDC (continuous line) and Discrete Consensus (dashed line). In the simulation edges are chosen sequentially at random with equal probability to allow a fair comparison with previous results that characterize the convergence time of quantized consensus and discrete consensus in terms of number of local state updates. Simulations show that the FDC algorithm has greatly improved convergence time. The difference between convergence times in random graphs for the two algorithms shown in Figure 2 scales as $\mathcal{O}(n)$. In Figure 3 the worst case convergence time over the same simulations is shown. It can be seen that also the worst case convergence time of FDC scales similarly as the average convergence time for random graphs with roughly the same performance gap with respect to discrete consensus.

In Figure 4 and Figure 5 simulations for a line graph with a number of nodes varying from 10 to 200 is shown. Each algorithm execution has been repeated 100 times with random initial conditions. Figure 4 shows a comparison of average convergence time, in number of local updates with semi-logarithmic scale, of FDC (continuous line) and Discrete Consensus (dashed line). For line graphs, where the diameter of the network is equal to $n - 1$, the worst case expected convergence time of Discrete Consensus grows as $O(n^4)$ while FDC grows as $O(n^2)$. Figure 5 shows a comparison of the maximum convergence time between the two algorithms.

Finally, Figure 6 shows a very significant feature of the proposed algorithm. Simulations are performed on random graphs with depth equal to 6. In this case we express the convergence time in normalized units of time as function of the random variable τ_e, which represents the interval of time between two consecutive selections of the same edge. By choosing $\mathbb{E}\left[\tau_e\right] = 1$, in Figure 6 is shown a comparison between the predicted upper bound on the convergence

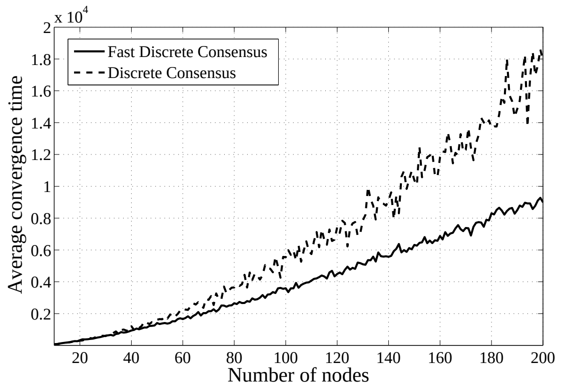

Fig. 2. Comparison of average convergence time between FDC and Discrete Consensus on random graphs (average over 100 executions.)

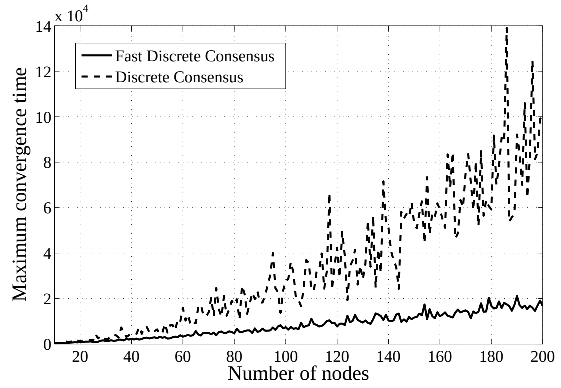

Fig. 3. Comparison of maximum convergence time between FDC and Discrete Consensus over random graphs (maximum over 100 executions).

time, which varies depending on the initial condition according to parameters M and m as in Proposition 5.1, and the simulated convergence time for the FDC algorithm. In this simulation it appears that convergence time does not depend on the number of nodes for random graphs. There are multiple reasons for this behavior. First, the chosen network topologies are random graphs which exhibit constant diameter as n grows. Second, the simulation takes into account parallelism of execution of the local state updates, i.e., to an increased number of edges it corresponds an increased number of local state updates in the unit of time. In particular, if each stochastic process that governs activation of each single edge is identical and independent, then as discussed in Section 4 the expected value of the random variable τ in Definition 4.3 does not increase with the number of nodes. Third, the upper bound on the convergence time discussed in Proposition 5.1 is conservative because it is computed for a worst case scenario and appears to be higher than the actual expected convergence time by a factor $\mathcal{O}(n)$ as observed in simulations.

Thus, simulations show that the discrete consensus problem can be solved in an interval of time independent from the number of nodes when considering random graphs of bounded diameter and exploiting the inherent parallelism of local state updates.

7 Conclusions

In this paper we have presented a novel decentralized algorithm, the FDC algorithm, that solves the discrete consensus problem, a generalization of the quantized consensus problem. It has been shown that the proposed algorithm has improved convergence time with respect to other discrete and quantized consensus algorithms in the literature. Furthermore, the proposed algorithm can be implemented in any undirected connected network topology as opposed to other approaches which require Hamiltonian graphs [11]. A stochastic characterization of the algorithm in terms of different stochastic edge/node selection processes has been presented taking into account the inherent parallelism of local state updates in large networks. Simulations have been provided to compare the theoretical upper bound

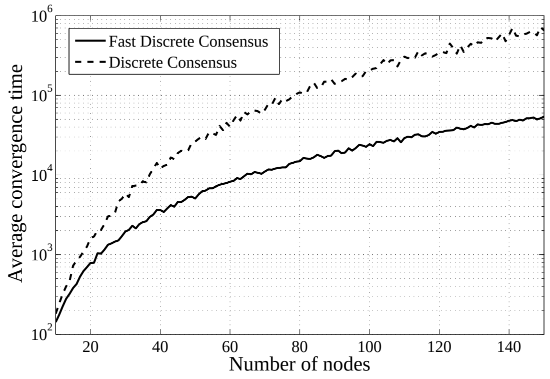

Fig. 4. Comparison of average convergence time between FDC and Discrete Consensus over a line graph (average over 100 executions), with semi-logarithmic scale.

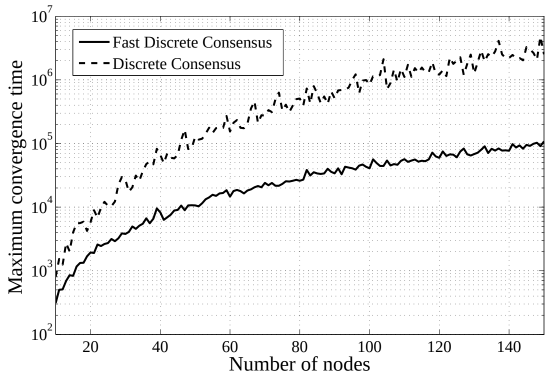

Fig. 5. Comparison of maximum convergence time between FDC and Discrete Consensus over a line graph (maximum over 100 executions), with semi-logarithmic scale.

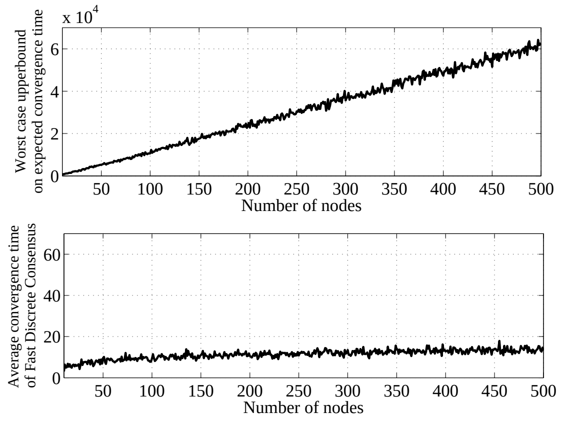

Fig. 6. Comparison between the upper bound on the average convergence time of Proposition 5.1 and simulated convergence time of FDC for random graphs expressed in normalized time with expected interval time between selection of the same edge equal to $\mathbb{E}[\tau_e] = 1$, the y-axes differ by 10^3 in scale.

to the convergence time with the simulated convergence time for random graphs and line graphs with increasing number of nodes. An empirical comparison of convergence time between FDC and Discrete Consensus has been provided to highlight the increased performance with respect to the state of the art.

Future work will consist in addressing time-varying network topologies.

References

[1] A. Kashyap, T. Başar, and R. Srikant, “Quantized consensus,” *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.

[2] G. Cybenko, “Dynamic load balancing for distributed memory multiprocessors,” *J. of Parallel and Distributed Computing*, vol. 7, pp. 279–301, 1989.

[3] B. Ghosh and S. Muthukrishnan, “Dynamic load balancing by random matchings,” *J. of Computer and Systems Sciences*, vol. 53, no. 3, pp. 357–370, 1996.

[4] M. Herlihy and S. Tirthapura, “Self-stabilizing smoothing and balancing networks,” *Distributed Computing*, 2005.

[5] M. Houle, E. Tempero, and G. Turner, “Optimal dimension exchange token distribution on complete binary trees,” *Theoretical Computer Science*, vol. 220, pp. 363–377, 1999.

[6] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri, “Gossip consensus algorithms via quantized communication,” *Automatica*, vol. 46, no. 1, pp. 70–80, 2010.

[7] K. Cai and H. Ishii, “Quantized consensus and averaging on gossip digraphs,” *IEEE Transactions on Automatic Control,*, vol. 56, no. 9, pp. 2087 –2100, 2011.

[8] R. J. Lavaei and Murray, “Quantized consensus by means of gossip algorithm,” *IEEE Transactions on Automatic Control,*, vol. 57, no. 1, pp. 19 –32, 2012.

[9] M. Zhu and S. Martínez, “On the convergence time of asynchronous distributed quantized averaging algorithms,” *IEEE Transactions on Automatic Control*, vol. 56, pp. 386–390, 2011.

[10] M. Franceschelli, A. Giua, and C. Seatzu, “A gossip-based algorithm for discrete consensus over heterogeneous networks,” *IEEE Transactions on Automatic Control,*, vol. 55, no. 5, pp. 1244 –1249, 2010.

[11] ——, “Quantized consensus in Hamiltonian graphs,” *Automatica*, vol. 47, no. 11, pp. 2495 – 2503, 2011.

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

[13] M. Fanti, A. Mangini, and W. Ukovich, “A quantized consensus algorithm for distributed task assignment,” *Proceedings of the IEEE Conference on Decision and Control*, pp. 2040–2045, Dec 2012.

[14] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, “Gossip algorithms for heterogeneous multi-vehicle routing problems,” *Nonlinear Analysis: Hybrid Systems*, vol. 10, no. 1, pp. 156–174, 2013.

[15] L. Margara, A. Pistocchi, and M. Vassura, “Perfect token distribution on trees,” *Structural Information and Communication Complexity*, pp. 221–232, 2004.

[16] M. R. Garey and D. S. Johnson, “Computers and intractability: a guide to the theory of np-completeness,” *WH Freeman & Co., San Francisco*, 1979.

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

[18] M. Fanti, M. Franceschelli, A. Mangini, G. Pedroncelli, and W. Ukovich, “Discrete consensus in networks with constrained capacity,” in *Proceedings of the IEEE Conference on Decision and Control*, Dec 2013, pp. 2012–2017.

[19] L. Babel, H. Kellerer, and V. Kotov, “The k-partitioning problem,” *Mathematical Methods of Operations Research*, vol. 47, no. 1, pp. 59–82, 1998.

[20] S. M. Johnson, “Optimal two- and three-stage production schedules with setup times included,” *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[21] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, “Time synchronization in wsns: A maximum-value-based consensus approach,” *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 660–675, 2014.

[22] J. He, P. Cheng, L. Shi, and J. Chen, “Sats: Secure average-consensus-based time synchronization in wireless sensor networks,” *IEEE Transactions on Signal Processing*, vol. 61, no. 24, pp. 6387–6400, 2013.

Appendix

A Proof of Proposition 4.5

The random variable τ in Definition 4.3 represents the length of the interval of time in which all edges have been selected at least once, including those edges incident on the sink node. The sink node has $h_1 = d_1 = 0$ and never changes this value. As a result of the local state update rule all nodes at unit distance from the sink node estimate correctly their distance from it as $h_i = 1$ after a time interval equal to τ. Iteratively, whenever Step 3 of the local state update rule in Algorithm 2 is executed, the nodes compare their estimated distance with that of their neighbor and update it accordingly. After q realizations of the random variable τ, corresponding to $T_q = \sum_{i=1}^{q} \tau_i$ units of time, all nodes that are at a distance less or equal to q with respect to the sink node have a correct estimation of their distance, i.e., for such nodes it is $h_i(t) = d_i$ for all $t \geq t_0 + T_q$. Since the distance between any node and the sink node is at most equal to the depth $d(\mathcal{G})$ of graph $\mathcal{G}$, after $T_{d(\mathcal{G})} = \sum_{i=1}^{d(\mathcal{G})} \tau_i$ units of time, every node in the network knows its correct distance from the unknown sink node.

B Proof of Proposition 4.6

The random variable τ in Definition 4.3 represents the length of the interval of time in which all edges have been selected at least once, including those edges incident on the sink node. As a result of the local state update rule and the assumption that the load of the sink node is not changing in the considered time interval, all nodes at unit distance from the sink node have an exact knowledge of the sink load after a time interval equal to τ_1. For the same reason, after q realizations of the random variable τ, corresponding to $T_q = \sum_{i=1}^{q} \tau_i$ units of time, all nodes that are at a distance less or equal to q with respect to the sink node have a correct estimation of the sink load, i.e., for such nodes it is $z_i(t) = x_1$ for all $t \in [t_1 + T_q, t_2]$. Therefore, since the distance between any node and the sink node is at most equal to the depth $d(\mathcal{G})$ of graph $\mathcal{G}$, after $T_{d(\mathcal{G})} = \sum_{i=1}^{d(\mathcal{G})} \tau_i$ units of time, it is $z_i(t) = x_1 \ \forall i \in \mathcal{V}$.

C Proof of Proposition 4.7

The sink load is by definition at most equal to the maximum load $x_1 \leq \max_{i\in\mathcal{V}} x_i(t)$. On the other hand, during the algorithm execution, the sink load is time-varying, thus other nodes' estimations of the sink load may be incorrect after each change. If the maximum load of the network is constant in the considered time interval, then in at most $T_{d(\mathcal{G})}$ units of time it holds $z_i(t) \geq x_1$ for all $i \in \mathcal{V}$ and $z_i \leq \max_{i\in\mathcal{V}} x_i(t)$, i.e., every node knows an upper bound to the actual value of the sink load. Consider now the selection of edge (i, j), in which node j holds the maximum value in the network. Two cases may occur.

(1) $x_i < x_j - c_{max}$: The nodes balance their load and the number of nodes which hold the maximum value is reduced by one, contradicting the assumption that the number of nodes with maximum value is constant in the considered interval of time.
(2) $x_i \geq x_j - c_{max}$: This implies that $z_i - x_i < c_{max}$, thus the largest load in the two nodes is moved closer to the sink node according to the Local State Update Rule in Algorithm 2.

Now we consider an interval of time in which the number of nodes which hold the maximum load is constant. After $T_{d(\mathcal{G})}$ units of time an estimation by every node of the sink load which cannot be greater than the maximum load in the network is ensured. If the edge selection process is stochastically persistent every τ units of time with τ as in Definition 4.3, then the distance between the sink node and a node with load equal to the maximum is reduced by at least one due to Step 4 of the local state update rule in Algorithm 2. It follows that in at most after a further $T_{d(\mathcal{G})}$ units of time the statement of the proposition holds, i.e., $x_1(t) = \max_{i\in\mathcal{V}} x_i(t) \ \forall t \in \left[t_1 + T_{2d(\mathcal{G})}, t_2\right]$, where $i = 1$ is the index of the sink node.

D Proof of Theorem 4.8

Consider function

$$V(t) = \|c^T Y(t)\|_\infty = \|x(t)\|_\infty. \tag{D.1}$$

The proof is based on four main arguments.

(1) Due to Proposition 4.5 after $T_{d(\mathcal{G})}$ units of time, every node correctly estimates its distance from the unknown sink node. In the following we consider the case in which $t \geq T_{d(\mathcal{G})}$ and the nodes have already estimated their distance.

(2) We now prove that $\forall t_k \geq 0$ it holds $V(t_{k+1}) \leq V(t_k)$.

If there is only one node with the maximum load and such a node is one of the two selected nodes at Step 5 of Algorithm 3, when the local state update rule in Algorithm 2 is executed, it holds $V(t_{k+1}) \leq V(t_k)$ for all $t_k \geq 0$ due to Step 3 of the Balancing rule which prevents any increase of the maximum value of load between any pair of nodes involved in a state update. If none of the selected nodes has the maximum load in the network, it holds $V(t_{k+1}) = V(t_k)$. The same occurs when one of the selected nodes has the maximum load but there also exist other nodes with the maximum load.

(3) Whenever the maximum load is reduced it holds $V(t_{k+1}) < V(t_k)$, the local maximum must decrease of at least 1 being $c_j \in \mathbb{N}$ for all $j = 1, \ldots, K$, thus $V(t_{k+1}) \leq V(t_k) - 1$.

(4) We now prove that if at a given time instant t_k it is

$$\max_{i,j\in\mathcal{V}} |x_i(t_k) - x_j(t_k)| > c_{max} \tag{D.2}$$

then Algorithm 3 after a finite interval of time at most equal to $\Delta = (n-1)T_{4d(\mathcal{G})}$ it holds $V(t_k+\Delta) < V(t_k)$. Let τ_i, $i = 1, \dots, d(\mathcal{G})$, be $d(\mathcal{G})$ realizations of the continuous random variable τ defined as in Definition 4.3. Let $T_s = \sum_{i=1}^{s} \tau_i$.

By Proposition 4.7 if the maximum load and the number of nodes with the maximum load remain constant, then if the edge selection process is stochastically persistent there exists $T_{2d(\mathcal{G})} = \sum_{i=1}^{2d(\mathcal{G})} \tau_i$ such that after $T_{2d(\mathcal{G})}$ units of time the sink node holds the maximum load. By Proposition 4.6 after at most a time equal to $T_{d(\mathcal{G})}$ all nodes have an exact estimation of the sink load. If inequality (D.2) holds, then there exists at least one node whose load is smaller than the maximum load of a quantity larger than $c_{\max}$. After further $T_{d(\mathcal{G})}$ units of time such load is brought closer to the sink node by at least one node due to Step 4 of the local state update rule in Algorithm 2. The maximum distance that such load needs to travel to get close to the sink node is at most equal to the graph depth $d(\mathcal{G})$. Thus, after at most $T_{d(\mathcal{G})}$ units of time a local balancing involving the sink node occurs, thus reducing the number of nodes verifying inequality (D.2). The same reasoning can be iterated until there exists some node verifying inequality (D.2). The number of nodes whose load corresponds to the maximum while the network is not in the discrete consensus state may be at most equal to $n-1$. It follows that, after at most $\Delta = (n-1)T_{4d(\mathcal{G})}$ units of time, $V(t_k+\Delta) \leq V(t_k) - 1$. According to Definition 4.2, $\forall t \geq 0$ there exist a finite $T > 0$ and a probability $p \in (0,1)$ such that $Pr(\hat{E}(t, t+T) \equiv E) \geq p$. Therefore, the probability that the event $\hat{E}(t, t+T) \equiv E$ occurs at least $(n-1)4d(\mathcal{G})$ times in a finite interval of time is at least

$$\begin{aligned} &Pr\Big(\hat{E}(t_i, t_i+T) \equiv E, i = 1, \dots (n-1)4d(\mathcal{G}), \\ &\text{with } t_{(n-1)4d(\mathcal{G})} \leq (n-1)T_{4d(\mathcal{G})}\Big) \geq p^{(n-1)4d(\mathcal{G})}. \end{aligned}$$

Since function $V(t_k)$ has integer values, it may decrease only a finite number of times before the state of discrete consensus is achieved, the condition $|x_i(t) - x_j(t)| \leq c_{max}$ $\forall i, j \in \mathcal{V}$ is achieved with a finite number of occurrences of the event $\hat{E}(t_i, t_i+T) \equiv E$. It follows that with probability one (almost surely), $\exists T_{conv} \in \mathbb{R}^+$ such that $\forall t \geq T_{conv}$, and $\forall i, j \in \mathcal{V}$, it is $|x_i(t) - x_j(t)| \leq c_{max}$, thus proving the statement. □