Distributed Integer Weight Balancing in the Presence of Time Delays in Directed Graphs

Apostolos I. Rikos, Student Member, IEEE, Christoforos N. Hadjicostis, Senior Member, IEEE

Abstract—A digraph with positive weights on its edges is weight-balanced if, for each node, the sum of the weights of the incoming edges equals the sum of the weights of the outgoing edges. Weight-balanced digraphs play an important role in a variety of cooperative control problems, including formation control, event-triggered coordination, distributed averaging, and optimization. Classical distributed algorithms for asymptotic average consensus typically assume timely and reliable exchange of information between neighboring components of a given multi-component system. These assumptions are not necessarily valid in practice due to varying delays that might affect computations at different nodes and/or transmissions at different links. In this work, we propose a distributed algorithm which solves the integer weight balancing problem in the presence of arbitrary (time-varying and inhomogeneous) time delays that might affect the transmission at a particular link at a particular time. The algorithm converges after a finite number of steps (that we explicitly bound) in the presence of bounded delays and converges in finite time (with probability one) in the case of unbounded delays (packet drops). Furthermore, we show that the resulting weight balanced digraph is unique regardless of how time delays or packet drops manifest themselves. We also analyze the computational and communication complexity of the proposed algorithm, and provide examples to illustrate its operation and performance.

I. INTRODUCTION

A distributed system or network consists of a set of components (nodes) that can share information with neighboring components via connection links (edges), forming a generally directed interconnection topology (digraph). The digraph that describes the underlying communication topology typically proves to be of vital importance for the effectiveness of distributed strategies in performing various tasks [2]–[4]. A weighted digraph has a real or integer value (called the edge weight) associated with each edge, and is said to be weight balanced if, for each node, the sum of the weights of the incoming edges equals the sum of the weights of the outgoing edges.

Weight-balanced digraphs find numerous applications in distributed adaptive control and synchronization in complex networks. Examples of applications where balance plays a key role include modeling of flocking behavior [2], network adaptation strategies based on the use of continuous second order models [5], prediction of distribution matrices for telephone traffic [6], and distributed adaptive strategies to tune the coupling weights of a network based on local information of node dynamics [7]. In particular, integer weight balance also finds applications in problems like swarm guidance [8] and fractional packing [9], [10].

It is worth pointing out that weight balance is closely related to weights that form a doubly stochastic matrix [11], which find applications in multicomponent systems (such as sensor networks) where one is interested in distributively averaging measurements at each component. In particular, the distributed average consensus problem has received significant attention from the computer science community [12] and the control community [13] due to its applicability to diverse areas, including multi-agent systems, distributed estimation and tracking [14] and distributed optimization [15]. Averaging in such settings follows a linear iteration, where each node repeatedly updates its value to be a linear combination of its own value and the values of its neighboring nodes, weighted according to the edge weights. Asymptotic average consensus is guaranteed (i.e., the nodes asymptotically reach consensus to the average of their initial values) if the weights used in the linear iteration form a doubly stochastic matrix [16].

Previous work has introduced various algorithms for balancing a weighted digraph in a distributed fashion, either with real edge-weights (following a geometric rate of convergence, [17]–[20]) or with integer edge-weights (after a finite number of steps, [11], [20]–[22]). It has been shown in [20] that, under a fixed interconnection topology, weight balancing with real weights can be reached by performing a linear iteration in a distributed fashion, i.e., by having each node update the weights of its outgoing links based on the balance it currently has. Furthermore, weight balance is reached asymptotically (with exponential rate of convergence), as long as the interconnection topology is strongly connected (or is a collection of strongly connected digraphs). To our knowledge, all previous works considered the transmission between nodes to happen instantaneously and no work (other than our work in [1], [20]) has been done for the case where time delays might be introduced during communication among different components (nodes) of the system. This assumption is not necessarily valid in practice due to varying delays that might affect computations at different nodes and transmissions at different links.

In this paper, we investigate the problem of integer weight-balancing in a multi-component system under a directed interconnection topology in the presence of bounded delays or unbounded delays (packet drops) in the communication links. We consider a fixed topology (digraph) and we devise a protocol, based on our previous work in [1], where each node
updates its state by combining the available (possibly delayed) weight information received by its in-neighbors. We establish that the proposed balancing algorithm reaches (after a finite number of steps that we explicitly bound) a set of weights that form a weight-balanced digraph despite the presence of arbitrary but bounded delays in the communication links. In the presence of packet drops over the communication links, the algorithm can be modified to converge to a set of weights that form a balanced graph after a finite number of iterations (with probability one). In both cases, the resulting digraph is shown to be unique and independent of how delays or packet drops manifest themselves.

The remainder of this paper is organized as follows. In Section II the notation used throughout the paper is provided, along with background on graph theory (needed for our subsequent development) and the problem formulation. In Section III we present the distributed algorithm which achieves integer weight-balancing in the presence of bounded delays after a finite number of iterations. We also analyze the case of unbounded delays (packet drops) in the communication links and discuss an event-triggered version of the algorithm (that can be used to avoid unnecessary transmissions). In Section IV we present simulation results and comparisons. Finally, we conclude in Section V with a brief summary and remarks about future work.

II. NOTATION AND PROBLEM FORMULATION

A. Notation

We list review some basic notions from [20] that are needed for our development. The sets of real, integer and natural numbers are denoted by $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{N}$, respectively. The symbol $\mathbb{N}_0$ denotes the set of nonnegative integers while the positive part of $\mathbb{Z}$ is denoted by the subscript $+$ (e.g. $\mathbb{Z}_+$). Vectors are denoted by small letters whereas matrices are denoted by capital letters. A matrix with nonnegative elements is called nonnegative matrix and is denoted by $A \geq 0$ while a matrix with positive elements is called positive matrix and is denoted by $A > 0$.

A distributed system whose components can exchange information via (possibly directed) interconnection links, can be captured by a digraph (directed graph). A weighted digraph of order $n$ ($n \geq 2$), is defined as $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of edges, and $\mathcal{W} = \{w_{ji}\} \in \mathbb{R}^{n \times n}$ is a weighted $n \times n$ adjacency matrix where $w_{ji}$ are nonnegative values. A directed edge from node $v_i$ to node $v_j$ is denoted by $m_{ji} \triangleq (v_j, v_i) \in \mathcal{E}$, and captures the fact that node $v_j$ can receive information from node $v_i$. The nonnegative weights satisfy $w_{ji} > 0$ if and only if $(v_j, v_i) \in \mathcal{E}$. The definition of $G$ excludes self-edges (though self-weights need to be added if we want to consider bistochastic digraphs [11]). We do not require bi-directional communication links, but we do assume that the given digraph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is strongly connected. A digraph is called strongly connected if for each pair of nodes $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed path from $v_i$ to $v_j$ i.e., we can find a sequence of nodes $v_i \equiv v_{i_0}$, $v_{i_1}$, \ldots, $v_{i_t} \equiv v_j$ such that $(v_{i_{\tau+1}}, v_{i_{\tau}}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t - 1$.

The subset of nodes that can directly transmit information to node $v_j$ is called the set of in-neighbors of $v_j$ and is represented by $\mathcal{N}_j^{-} = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$. Respectively, the subset of nodes that can receive information from $v_j$ is called the set of out-neighbors of $v_j$ and is represented by $\mathcal{N}_j^{+} = \{v_i \in \mathcal{V} \mid (v_i, v_j) \in \mathcal{E}\}$. The number of in-neighbors of $v_j$ is the in-degree of $v_j$ and is denoted by $\mathcal{D}_j^{-} = |\mathcal{N}_j^{-}|$. The number of out-neighbors of $v_j$ is the out-degree of $v_j$ and is denoted by $\mathcal{D}_j^{+} = |\mathcal{N}_j^{+}|$. We assume that node $v_j$ assigns a unique priority to each outgoing edge in the set $\{1, 2, \ldots, \mathcal{D}_j^{+}\}$. The priority of link $(l,j)$ is denoted by $p_{lj}$ (such that $\{p_{lj} \mid v_l \in \mathcal{N}_j^{+}\} = \{1, 2, \ldots, \mathcal{D}_j^{+}\}$) and will be used later on as a way of allowing node $v_j$ to make changes to its outgoing edge-weight values in a predetermined (cyclic) order.

**Definition 1.** Given a weighted digraph $G(\mathcal{V}, \mathcal{E}, \mathcal{W})$ of order $n$, the total in-weight of node $v_j$ is $S_j^{-} = \sum_{v_i \in \mathcal{N}_j^{-}} w_{ji}$, whereas the total out-weight of node $v_j$ is $S_j^{+} = \sum_{v_i \in \mathcal{N}_j^{+}} w_{ij}$. Also, the weight imbalance of node $v_j$ is $x_j = S_j^{-} - S_j^{+}$ while the total imbalance of digraph $G$ is $\varepsilon = \sum_{j=1}^{n} |x_j|$.

**Definition 2.** A weighted digraph $G(\mathcal{V}, \mathcal{E}, \mathcal{W})$ is called weight-balanced if its total imbalance is 0, i.e., $\varepsilon = \sum_{j=1}^{n} |x_j| = 0$.

B. Modeling Delays

For the development of the majority of the results in this paper, we assume that a transmission on the link from node $v_j$ to node $v_i$ initiated at time step $k$ undergoes an *a priori unknown* delay $\tau_{ij}[k]$, where $\tau_{ij}[k]$ is an integer that satisfies $0 \leq \tau_{ij}[k] \leq \tau_{ij} \leq \infty$ (i.e., delays are bounded\(^1\)). The maximum delay is denoted by $\tau = \max_{(v_i,v_j) \in \mathcal{E}} \tau_{ij}$. In the weight balancing setting we consider, node $v_j$ is in charge of assigning weights $\tilde{w}_{ji}[,k]$ to each link $(v_i,v_j)$ and of sending to each out-neighboring node $v_i$ the value $\tilde{w}_{ij}[k]$. Under the above delay model, the weight $\tilde{w}_{ij}[k]$ becomes available to node $v_i$ at time step $k + \tau_{ij}[k]$. From the perspective of node $v_j$, the delayed weight for link $(v_j,v_i)$, $\forall v_i \in \mathcal{N}_j^{-}$, at time step $k$ is given by

$$\tilde{w}_{ji}[k] = \tilde{w}_{ji}[k_{last}], \text{ where } k_{last} = \max\{s \mid s + \tau_{ji}[s] \leq k\},$$

i.e., $\tilde{w}_{ji}[k]$ is the most recently sent weight $\tilde{w}_{ji}[\cdot]$ seen at node $v_j$ by time step $k$.

**Remark 1.** In our case, the above definition of $\tilde{w}_{ij}[k]$ is equivalent to $\tilde{w}_{ij}[k] = \max\{\tilde{w}_{ij}[s] \mid s + \tau_{ji}[s] \leq k\}$ because, during the execution of the algorithm presented later, the weights $\tilde{w}_{ij}$ on each edge $(v_j,v_i)$ are assigned by node $v_i$ in a non-decreasing manner, i.e., for $v_j \in \mathcal{N}_i^{+}$, $\tilde{w}_{ij}[k] \leq \tilde{w}_{ij}[k+1], \forall k \in \mathbb{N}_0$.

**Proposition 1.** When $\tilde{w}_{ji}[k]$ are non-decreasing and delays are bounded, the above delay model, implies that $\tilde{w}_{ji}[k+\tau] \geq \tilde{w}_{ji}[k]$.\(^1\)We later relax the assumption of bounded delays and consider packet dropping links.
Proof. The proof follows directly from the definition of $\pi_j[k]$. We have that $w_{ji}^{k}[k + \tau] = w_{ji}[k_{last}]$ where $k_{last} = \max\{s \mid s + \tau_{ji}[s] \leq k + \tau\}$. Obviously, $k_{last} \geq k$ because $\tau_{ji}[k] \leq \tau$. Since the weights of each edge are non-decreasing we have that $w_{ji}[k + \tau] = w_{ji}[k_{last}] \geq w_{ji}[k]$.

Definition 3. Given a weighted digraph $G(\mathcal{V}, \mathcal{E}, \mathcal{W})$ of order $n$, the total in-weight seen at time step $k$ by node $v_j$ is

$$S_j^-[k] = \sum_{v_i \in N^-_j} w_{ji}[k].$$

Since, for every node, the weights of its outgoing edges are available without delay, the total out-weight of node $v_j$ at time step $k$ is the same as in Definition 1 denoted by

$$S_j^+[k] = \sum_{v_i \in N^+_j} w_{ji}[k].$$

Furthermore, the delayed weight imbalance of node $v_j$, calculated at time step $k$, is

$$\tau_j[k] = S_j^-[k] - S_j^+[k],$$

while the total delayed imbalance of digraph $G$, at a time step $k$, is

$$\tau[k] = \sum_{j=1}^n |\tau_j[k]|.$$

C. Problem Formulation

Given a strongly connected digraph $G(\mathcal{V}, \mathcal{E})$ and considering (for now) that link transmissions undergo arbitrary but bounded delays, we want to develop a distributed algorithm that allows the nodes to iteratively adjust the weights on their outgoing edges so that they eventually obtain a weight matrix $\mathcal{W} = [w_{ji}]$ that satisfies the following:

1) $w_{ji} > 0$ for each edge $(v_j, v_i) \in \mathcal{E}$;
2) $w_{ji} = 0$ if $(v_j, v_i) \notin \mathcal{E}$;
3) $S_j^+ = S_j^-$ for every $v_j \in \mathcal{V}$.

We introduce and analyze a distributed algorithm that allows each node to assign integer weights to its outgoing links, so that the resulting weight assignment is balanced. The proposed algorithm is able to handle arbitrary but bounded time-delays that may affect the information exchange between agents in the system. We also explicitly bound the number of steps required for convergence. Among other implications, this bound establishes that the proposed algorithm completes its operation in polynomial time, as long as the underlying digraph is strongly connected or is a collection of strongly connected digraphs. More specifically, the proposed algorithm completes in at most $O(n^6\tau)$ steps, where $n$ is the number of nodes of the given digraph and $\tau$ is the maximum delay in the digraph. For packet dropping links, we consider the case when transmissions reside on packers that can be dropped with some nonzero probability and show that the algorithm will converge after a finite number of steps with probability one.

III. DISTRIBUTED INTEGER BALANCING WITH TIME DELAYS

We now introduce a distributed algorithm in which the nodes iteratively adjust the integer weights on their outgoing edges such that the digraph becomes weight-balanced after a finite number of iterations in the presence of arbitrary but bounded delays. As in [21] and [22] we assume that each node observes (but cannot set) the weights of its incoming edges and that each node can separately adjust the weights on its outgoing edges (note that this assumption is not present in [17] where each node sets equal real weights to all of its outgoing edges, but becomes necessary when restricting ourselves to integer weights).

Given a strongly connected digraph $G = (\mathcal{V}, \mathcal{E})$, the algorithm has each node initialize the weights of all of its outgoing edges to unity. We consider for now that in digraph $G$, each link transmission can undergo an arbitrary but bounded delay. In order to handle delays, we employ a strategy where the nodes run a weight balancing protocol and process weight information as soon as it arrives. According to this protocol, each node enters an iterative stage where it performs the following steps:

1) It computes its delayed weight imbalance according to the latest received weight values from its in-neighbors.
2) If it has positive (delayed) imbalance, it increases by 1 the integer weights of its outgoing edges one at a time, following the priority order until it becomes weight-balanced. This means that the outgoing edges are assigned, if possible, equal integer weights; otherwise, if this is not possible, they are assigned integer weights such that the maximum difference among them is equal to one (it should be clear that for a given $S_j$ the priorities among the outgoing links of node $v_j$ make this assignment unique).

We argue that the above distributed algorithm obtains integer weights that balance the digraph after a finite number of iterations (which we bound in terms of the number of nodes/edges of the given digraph). Using a path-based analysis of the algorithm, we prove that the resulting weight-balanced digraph is unique and independent of the link-delays that may occur during the execution of the algorithm. For simplicity, we assume that during the execution of the distributed algorithm, the nodes, update the weights on their outgoing edges in a synchronous manner based on the information available at each node at that particular instant.

A. Description of the Distributed Algorithm

Input: A strongly connected digraph $G = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges.

Initialization: Set $k = 0$; each node $v_j \in \mathcal{V}$ does the following:

1) It assigns a unique priority to its outgoing edges as $P_{ij}$, for
\(v_i \in \mathcal{N}_j^+(\text{such that } \{P_{ij} \mid v_i \in \mathcal{N}_j^+ \} = \{1, 2, \ldots, D_j^+\})\).

2) It sets its outgoing edge weights as
\[
w_{ij}[0] = \begin{cases} 
0, & \text{if } (v_i, v_j) \notin \mathcal{E}, \\
1, & \text{if } (v_i, v_j) \in \mathcal{E}.
\end{cases}
\]

3) It sets its (perceived) incoming weights to be equal to unity, \(w_{ji}[0] = 1, \forall v_i \in \mathcal{N}_j^-\).

4) It transmits the weights \(w_{ij}[1] = w_{ij}[0]\) on each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to \(v_i \in \mathcal{N}_j^+\).

**Iteration:** For \(k = 0, 1, 2, \ldots\), each node \(v_j \in \mathcal{V}\) does the following:

1) It receives the weights on its incoming edges \(w_{ji}[k+1]\).

More specifically, for each node \(v_i \in \mathcal{N}_j^-\) let \(\mathcal{W}_{ji} = \{w_{ji}[s + \tau_{ji}[s]] \mid s \in \mathcal{N}_j^-\}\) be the set of weights of link \((v_j, v_i) \in \mathcal{E}\) that arrive at node \(v_j\) at time step \(k+1\). We have that
\[
w_{ji}[k+1] = \left\{ \frac{\max\{w_{ji}[k], \max_{v_i \in \mathcal{W}_{ji}} w_{ji}\}}{w_{ji}[k]} \right\}_{w_{ji}[k]} \quad \text{if } \mathcal{W}_{ji} = \emptyset, \quad \text{if } \mathcal{W}_{ji} = \emptyset.
\]

2) It computes its (delayed) weight imbalance according to the latest received weight values from its in-neighbors
\[
\pi_j[k] = \sum_{v_i \in \mathcal{N}_j^-} w_{ji}[k] - \sum_{v_i \in \mathcal{N}_j^+} w_{ij}[k].
\]

Recall that \(\pi_{ji}\) was defined in Equation (1) and \(\pi_j\) was defined in Definition 3.

3) If \(\pi_j[k] = br_j^+ > 0\), it sets the values of the weights on its outgoing edges to \(w_{ij}[k+1] = \left[ \frac{\pi_j[k]}{D_j^+} \right]\), \(\forall v_i \in \mathcal{N}_j^+\). Then, it chooses the set of the first (according to the priority order) \(br_j^+ - D_j^+ \left[ \frac{\pi_j[k]}{D_j^+} \right]\) outgoing edges, and increases their weight by 1 so that \(|w_{ij}[k+1] - w_{ij}[k+1]| \leq 1, \forall v_i, v_i \in \mathcal{N}_j^+\) and \(\pi_j[k+1] = \pi_j[k+1]\).

If \(\pi_j[k] < 0\) then the node sets \(w_{ij}[k+1] = w_{ij}[k], \forall v_i \in \mathcal{N}_j^+\).

4) It transmits the new weights \(w_{ij}[k+1]\) on each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to each \(v_i \in \mathcal{N}_j^+\).

5) It repeats (increases \(k\) to \(k+1\) and goes back to Step 1).

The intuition behind the proposed algorithm is that each node \(v_j\) computes the sum of the weights on its outgoing edges \(\pi_j[k]\) against the sum of the weights on its incoming edges \(\pi_{ji}[k]\) received at iteration \(k\) (i.e., it calculates \(\pi_j[k]\)). Then, if \(\pi_j[k] > 0\) it increases the weights of its outgoing edges according to the priority order it choose at initialization, i.e., it sets all \(w_{ij}[k+1] = \left[ \frac{\pi_j[k]}{D_j^+} \right]\), \(\forall v_i \in \mathcal{N}_j^+\) as described in step 2 of the algorithm and then it increases by 1 the values of the edges \(m_{ij}\) with priority \(P_{ij} = \{1, 2, \ldots, br_j^+ - D_j^+ \left[ \frac{\pi_j[k]}{D_j^+} \right]\}\), (obviously \(0 \leq br_j^+ - D_j^+ \left[ \frac{\pi_j[k]}{D_j^+} \right] < D_j^+\)). Finally, it receives the weights on its incoming edges, updates \(\pi_{ji}[k+1]\), and repeats the iteration.

The following lemma is useful in our analysis later on.

**Lemma 1.** Suppose that at iteration \(k\) node \(v_j\) has inweights \(\{\pi_{ji}[k] \mid v_i \in \mathcal{N}_j^-\}\). Given priorities \(P_{ij}\), where \(v_i \in \mathcal{N}_j^+,\) on its outgoing links (such that \(\{P_{ij} \mid v_i \in \mathcal{N}_j^+\} = \{1, 2, \ldots, D_j^+\}\), we have that
\[
w_{ij}[k+1] = \mathcal{F}_{ij} \left( \sum_{v_i \in \mathcal{N}_j^+} \pi_{ji}[k], \ v_i \in \mathcal{N}_j^+ \right).
\]

Moreover, \(\mathcal{F}_{ij}\) is monotonic in its argument, i.e., \(\mathcal{F}_{ij}(x) \geq \mathcal{F}_{ij}(y)\) if \(x \geq y\).

**Proof.** From the algorithm description we have that for integer \(x\), we have that
\[
\mathcal{F}_{ij}(x) = \left[ \frac{x}{D_j^+} \right] + \text{ind}_{ij}(x),
\]
where
\[
\text{ind}_{ij} = \begin{cases} 
1, & \text{if } P_{ij} \leq x - \left[ \frac{x}{D_j^+} \right] D_j^+, \\
0, & \text{otherwise}.
\end{cases}
\]

\(\mathcal{F}_{ij}(x)\) is clearly monotonic in its argument. \(\square\)

We now illustrate the distributed algorithm via an example and then explain why it asymptotically results in a weight-balanced digraph after a finite number of iterations. We also obtain bounds on its execution time.

**Example 1.** Consider the digraph \(\mathcal{G} = (\mathcal{V}, \mathcal{E})\) in Fig. 1, where \(\mathcal{V} = \{v_1, v_2, \ldots, v_6\}, \mathcal{E} = \{m_{31}, m_{32}, m_{13}, \ldots, m_{46}\}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}\). The weight on each edge is initialized to \(w_{ij}[0] = 1\) for \((v_i, v_j) \in \mathcal{E}\) and each node assigns a unique priority to each of its outgoing edges. For the purposes of this example, let us assume that these priorities are as follows:

- \(v_1 : P_{31} = 1,\)
- \(v_2 : P_{32} = 1,\)
- \(v_3 : P_{13} = 1, P_{23} = 2, P_{43} = 3,\)
- \(v_4 : P_{34} = 1, P_{64} = 2,\)
- \(v_5 : P_{15} = 1, P_{35} = 2, P_{45} = 3,\)
- \(v_6 : P_{26} = 1, P_{46} = 2.\)

(For example, node \(v_4\) will first increase \(w_{46}\) and then \(w_{64}\).) As a first step all the nodes receive the new weights of their incoming edges. Then, each node computes its weight imbalance \(\pi_{ji}[0] = \pi_{i}[0] - \pi_{ji}[0], \ \forall v_j \in \mathcal{V}\) (these values are shown in Fig. 1).

Once each node computes its imbalance, the distributed algorithm requires each node with positive imbalance to increase the value of the weights on its outgoing edges by equal integer amounts (or with maximum difference between them equal to one) according to the predetermined priority that each node assigned to its outgoing edges, so that the total increase makes the node balanced. In this case, the nodes
V6
V3
V5
V4
V1
V2
X1 = 0
X2 = 0
X3 = 0
X4 = 0
X5 = 0
X6 = 0
3
4
3
4
3
2
1
3
1
1
1
1

that have positive imbalance are nodes v1, v2 and v4 (equal to \( \tau_1[0] = 1, \tau_2[0] = 1 \) and \( \tau_4[0] = 1 \)) respectively, and they increase their outgoing edges as shown in Fig. 2.

In Fig. 2 we can see that the edges \( m_{31}, m_{32}, m_{54} \) have new values equal to 2. Note here that the nodes v1, v2 and v3 increased the edge-weights \( w_{31}, w_{32} \) and \( v_{54} \) respectively, since the corresponding nodes had the highest priorities (as chosen by the nodes during the initialization step). Nodes v3 and v5 will receive the new weights of their incoming edges after a number of iterations equal to the corresponding time-delay for each edge i.e., \( v_3 \) and \( v_5 \) will receive them after \( \tau_{31}[0], \tau_{32}[0] \) and \( \tau_{54}[0] \), respectively. For example, let us consider that the time delays are equal to \( \tau_{31}[0] = 6, \tau_{32}[0] = 3 \) and \( \tau_{54}[0] = 7 \). This means that node v3 will receive the new weight of \( m_{31} \) at \( k = 6 \) and the new weight of \( m_{32} \) at \( k = 3 \), while \( v_5 \) will receive the new weight of \( m_{54} \) at \( k = 7 \) (at least assuming long enough delays in subsequent transmissions between the nodes). In Fig. 3, we can see the digraph at time step \( k = 5 \). Here node v3 has received the new weight of edge \( m_{32} \) and has increased its outgoing edge \( m_{13} \) by 1 (because it has the highest priority) while it maintains the value of its outgoing edge \( m_{23} \) (which has the second priority) the same (and equal to 1) because the new weight of the edge \( m_{31} \) has not yet arrived.

Note here that all the nodes in the digraph continue to send the same values on their outgoing edges in every iteration until they receive updated weights on their incoming edges. This means that the time delays \( \tau_{31}[0], \tau_{32}[0] \) and \( \tau_{54}[0] \) are not necessarily the time-steps after which the nodes \( v_3 \) and \( v_5 \) will be informed of the new weights on their incoming edges. For example, if \( \tau_{32}[0] = 3 \) then \( v_3 \) will receive the new weight \( w_{32}[1] = 2 \) at iteration \( k = 3 \); however, at iteration \( k = 1 \) node \( v_2 \) re-sends its outgoing weights to its out-neighbors; thus, if \( \tau_{32}[1] = 1 \) then \( v_3 \) will receive the new weight \( w_{32}[1] = 2 \) at iteration \( k = 2 \) (it will also receive it at \( k = 3 \)) and it will act accordingly (it will essentially ignore it).

After the integer weight update on the outgoing edges of each node with positive imbalance at \( k = 0 \), the nodes check for updated incoming edge weights \( w_{v_j}[1], \forall (v_j, v_i) \in E \) (assuming that \( w_{v_j}[1] = w_{v_j}[0] = w_{v_j}[1] \) if no weight is received). Then they recalculate their imbalances \( \tau_{v_j}[1], \forall v_j \in V \), and the process is repeated. After a finite number of iterations, which we explicitly bound in the next sections, we reach the weighted digraph with integer weights shown in Fig. 4. As we will argue later in the paper, this weighted digraph is the same irrespective of how time-delays manifest themselves.

\[ \Box \]

B. Completion Time of Distributed Algorithm

In this section we analyze the functionality of the distributed algorithm and we prove that it solves the weight-balancing problem in the presence of arbitrary (time-varying, inhomogeneous) but bounded time delays that may appear during the information exchange between agents in the system. Specifically, we prove that the proposed distributed algorithm results in a set of weights that form a weight balanced matrix after \( O(n^5 \tau) \) iterations, where \( n \) is the number of nodes of the given digraph and \( \tau \) is the maximum delay in the digraph, while we show that the resulting weight balanced digraph is unique (irrespective of how delays manifest themselves) and identical to the one we obtain when transmissions between nodes happen instantaneously (no delays). We begin the analysis with the following theorem.

Setup: Consider an arbitrary strongly connected digraph \( G = (V, E) \), where \( V = \{v_1, v_2, \ldots, v_n\} \) is the set of nodes, and \( E \subseteq V \times V - \{(v_j, v_j) | v_j \in V \} \) is the set of edges. Consider an execution of the proposed distributed algorithm
where there are no delays ($\tau = 0$) and denote the resulting weights on the edges as
\[ w_{ij}^*[0] = 1, w_{ij}^*[1], \ldots, w_{ij}^*[k], \ldots \forall (v_i, v_j) \in E. \]

Consider another execution of the proposed distributed balancing algorithm where there are arbitrary but bounded delays ($0 < \tau < \infty$) and denote the resulting set of weights as
\[ \text{Transmitted} : w_{ij}[0] = 1, w_{ij}[1], \ldots, w_{ij}[k], \ldots \forall (v_i, v_j) \in E, \]
\[ \text{Received} : w_{ij}[0] = 1, w_{ij}[1], \ldots, w_{ij}[k], \ldots \forall (v_i, v_j) \in E. \]

**Theorem 1.** Under the above setup, we have for all $(v_i, v_j) \in E$ and all $k \geq 0$

1. $w_{ij}^*[k + 1] \geq w_{ij}^*[k]$,
2. $w_{ij}[k + 1] \geq w_{ij}[k]$,
3. $w_{ij}[k + 1] \geq w_{ij}[k]$.

**Proof.** Consider the case when $\tau = 0$. At Step 2 of the proposed algorithm (at an arbitrary iteration $k$), if node $v_i$ has positive imbalance $x_i[k] > 0$ then it increases the weights on its outgoing edges so that it becomes weight balanced (i.e., $w_{ij}^*[k + 1] \geq w_{ij}^*[k], \forall v_i \in N_j^+$. If node $v_i$ has negative (or zero) imbalance $x_i[k] \leq 0$, it leaves the weights of its outgoing edges unchanged (i.e., $w_{ij}^*[k + 1] = w_{ij}^*[k], \forall v_i \in N_j^+$. As a result, we have
\[ w_{ij}^*[k + 1] \geq w_{ij}^*[k], \forall (v_i, v_j) \in E. \]

Consider now the case when arbitrary but bounded time-delays ($\tau > 0$) affect link transmissions. Using a similar argument as above, we easily establish that $w_{ij}[k + 1] \geq w_{ij}[k], \forall (v_i, v_j) \in E$. By the definition of $w_{ij}[k + 1]$, we have that $w_{ij}[k + 1] = w_{ij}[k]_{last}$, where $k_{last} = \max\{s | s + \tau s [s] \leq k + 1\}$. Similarly, $w_{ij}[k] = w_{ij}[k]_{last}$, where $k_{last} = \max\{s | s + \tau s [s] \leq k\}$. Clearly, $k_{last} \leq k_{last}$ and since $w_{ij}[k + 1] \geq w_{ij}[k]$ and $k_{last} \geq k_{last}$, we have that
\[ w_{ij}[k + 1] \geq w_{ij}[k], \forall v_i \in N_j^+, \]
which completes the proof. $\square$

After establishing monotonicity for the weights of the outgoing edges for every node in the digraph, we continue with the following theorem.

**Theorem 2.** Under the above setup, it holds that for every $k,$
\[ w_{ij}[k] \leq w_{ij}[k] \leq w_{ij}[k], \forall (v_i, v_j) \in E. \] (2)

**Proof.** The proof is by induction. For $k = 0$, we have at initialization $w_{ij}[0] = w_{ij}[0] = 1$ and (2) holds. Let us assume that for every $(v_i, v_j) \in E$ we have
\[ w_{ij}[k] \leq w_{ij}[k] \leq w_{ij}[k], \]
by the induction hypothesis. We would like to show that
\[ w_{ij}[k + 1] \leq w_{ij}[k + 1] \leq w_{ij}[k + 1], \forall (v_i, v_j) \in E. \]

The fact that $w_{ij}[k + 1] \leq w_{ij}[k + 1]$ is a consequence of Theorem 1; we have that $w_{ij}[k + 1] = w_{ij}[k]_{last}$, where $k_{last} = \max\{s | s + \tau s [s] \leq k + 1\}$. Clearly, $k_{last} \leq k + 1$ and (from Theorem 1) $w_{ij}[k + 1] \leq w_{ij}[k + 1]$. As a result, we have that
\[ w_{ij}[k + 1] \leq w_{ij}[k + 1] \leq w_{ij}[k + 1], \forall (v_i, v_j) \in E. \]
resulting edge weights of the case where no delays affect link transmissions.

Remark 2. Note that finite time execution is obtained in this paper (as well as in the case of no delays in [22]) through graph path analysis; this should be contrasted to the finite time techniques for average consensus in [23], [24], which (treat a quite distinct problem and) employ linear system observability properties to argue that at most \( n \) values (where \( n \) is the number of nodes) are sufficient to determine the average.

C. Operation in the Presence of Packet Dropping Links

Apart from bounded delays, unreliable communication links in practical settings could also result in possible packet drops (i.e., unbounded delays) in the corresponding communication network. In this section, we analyze the performance of the proposed distributed weight-balancing algorithm in the presence of possible packet drops in the communication links.

To model packet drops, we assume that, at each time step \( k \), a packet that is sent from node \( v_i \) to node \( v_j \) on link \((v_j, v_i) \in E\) is dropped with probability \( q_{ji} \), where we have \( q_{ji} \leq 1\). For simplicity, we assume independence between packet drops at different time steps or different links. We establish that, despite the presence of packet drops, the proposed distributed algorithm converges, with probability one, to a weight balanced digraph after a finite number of iterations. This weight balanced digraph is identical to the one obtained under no packet drops.

Proposition 2. Consider the above setup, where the proposed balancing algorithm, with no packet drops and no delays, converges to a set of weights \( w_{ij}^* \) that form a weight-balanced digraph after a finite number of steps bounded by \( O(n^6) \). In the presence of packet drops occurring with probability \( q_{ij} \), \( q_{ji} \leq 1 \), \( \forall (v_i, v_j) \in E \) (independently between different links and different time steps), the proposed balancing algorithm also converges, with probability one, to a set of weights \( w_{ij} = w_{ij}^* \), \( \forall (v_i, v_j) \in E \), after a finite number of iterations.

Proof. Consider an execution of the proposed distributed balancing algorithm where packets containing information are dropped with probability \( q_{ij} \leq 1 \) for each communication link \((v_i, v_j) \in E\), and assume independence between packet drops at different time steps and different links.

During transmissions on link \((v_i, v_j)\), we have that at each transmission, a packet goes through with probability \( 1 - q_{ij} > 0 \). Thus, if we consider \( k_{ij} \) consecutive uses of link \((v_i, v_j)\), the probability that at least one packet will go through is \( 1 - q_{ij}^{k_{ij}} \), which will be arbitrarily close to 1 for a sufficiently large \( k_{ij} \). Specifically, for any (arbitrarily small) \( \epsilon > 0 \), we can choose

\[
 k_{ij} = \left\lfloor \log \frac{1}{\epsilon} \log q_{ij} \right\rfloor,
\]

to ensure that each transmission goes through by \( k_{ij} \) steps with probability \( 1 - \epsilon \).

Let \( \tau = \max_{(v_i, v_j) \in E} \{k_{ij}\} \); then since the proposed distributed algorithm under no packet drops reaches a set of weights \( w_{ij}^* \), \( \forall (v_i, v_j) \in E \) that forms a weight balanced digraph after \( O(n^6) \) steps, we can conclude that it will complete by \( O(n^6\tau) \) steps with probability \( 1 - (1 - \epsilon)^n |E| \) in the presence of packet drops (note that \( |E| \) is the number of edges in the given digraph). By making \( \epsilon \) arbitrarily small we can make this probability arbitrarily close to 1. Moreover, since this particular execution of the algorithm (that occurs with probability \( 1 - (1 - \epsilon)^n |E| \)) is essentially identical to an execution of the algorithm in Section III-A with delays that are bounded by \( \tau \), the final weights are identical to the weights of that algorithm (i.e., for large enough \( k \) we have \( w_{ij}[k] = w_{ij}^* \) for all \((v_i, v_j) \in E\)).

Remark 3. It is worth pointing out that the proposed distributed algorithm is able to converge (with probability one) to a set of weights that form a balanced graph after a finite number of iterations in the case where there are both possible packet drops and arbitrary but bounded time delays in the communication links, while the resulting weight balanced digraph is again unique and independent of how packet drops and delays manifest themselves in link transmissions.

D. Event Triggered Operation

Motivated by the need to reduce energy consumption, communication bandwidth, network congestion, and/or processor usage, many researchers have considered the use of event-triggered communication and control [25], [26]. In this section, we discuss an event-triggered operation of the proposed distributed algorithm where each agent autonomously decides when communication and control updates should occur so that the resulting network executions still result in a weight balanced digraph after a finite number of steps in the presence of arbitrary (time-varying, inhomogeneous) but bounded time delays that might affect link transmissions. More specifically, following the proposed event-triggered strategy, we can prove that (i) all nodes eventually stop transmitting, and (ii) the proposed distributed algorithm is able to obtain a set of weights that balance the corresponding digraph after a finite number of iterations.

Due to space limitations we describe only the differences of the algorithm’s event-triggered operation compared to the proposed distributed algorithm which was previously introduced.

Input: A strongly connected digraph \( \mathcal{G} = (V, E) \) with \( n = |V| \) nodes, \( m = |E| \) edges.

Initialization: (Steps 1, 2, 3, 4 same as previous Algorithm).

Iteration: For \( k = 0, 1, 2, \ldots \), each node \( v_j \in V \) does the following:

1) Node \( v_j \) receives the weights on each of its incoming edges \( \overline{w}_{ji}[k+1] \). More specifically, for each in-neighboring node \( v_i \in \mathcal{N}_j^- \) let \( W_j = \{w_{ji}[s + \tau_{ji}(s)] | s + \tau_{ji}(s) = k + 1\} \) be the set of weights of link \((v_j, v_i) \in E\) that arrive at node \( v_j \) at time step \( k + 1 \). We have that

\[
\overline{w}_{ji}[k+1] = \begin{cases} \overline{w}_{ji}[k], & \text{if } W_j = \emptyset, \\ \max\{\overline{w}_{ji}[k], \max_{(v_i, v_j) \in E_j} \overline{w}_{ji}\}, & \text{if } W_j \neq \emptyset. \end{cases}
\]

2) Event triggered condition: If \( \overline{w}_{ji}[k+1] = \overline{w}_{ji}[k] \) for each \( v_i \in \mathcal{N}_j^- \) then node \( v_j \) skips Steps 3, 4, and 5 below; Otherwise (event triggered condition), it performs the steps below.
(Same as Step 2 in previous algorithm).
4) (Same as Step 3 in previous algorithm).
5) (Same as Step 4 in previous algorithm).
6) It increases \( k \) to \( k + 1 \) and goes back to Step 1.

**Proposition 3.** Under the above setup, the proposed event-triggered balancing algorithm converges, in the presence of bounded delays (\( \tau > 0 \)), to a set of weights \( w_{ij} = w_{ij}^* \), \( \forall (v_i, v_j) \in \mathcal{E} \), after a finite number of steps bounded by \( O(n^6\tau) \) iterations (where the set of weights \( w_{ij}^* \) is the set of weights obtained by the nominal algorithm that runs with no event-triggering and no delays).

**Proof.** Consider the event-triggered operation of the proposed distributed balancing algorithm in the presence of bounded delays in the communication links. The event-triggered operation is identical to the operation of the proposed distributed algorithm with delays introduced in Section III-A if we assume that in the latter algorithm all transmissions of identical weights (that occur in the original version of the algorithm but not in the event-triggered version) suffer the maximum possible delay. As a result, since the operation of both algorithms is identical\(^5\), we have that the event-triggered operation of the distributed algorithm will converge to a set of weights that form a weight balanced digraph after a finite number of steps bounded by \( O(n^6\tau) \) iterations (where \( O(n^6) \) is the number of steps required for the distributed algorithm to converge to a weight balanced digraph when \( \tau = 0 \) \([22]\)). Also, since \( \exists k_0 \in \mathbb{N}_0 \) for which \( w_{ij}[k + 1] = w_{ij}[k] = w_{ij}^* \), \( \forall k \geq k_0 \), from Step 3 of the algorithm, we can see that all nodes eventually stop transmitting (and the weights are identical to the weights obtained by the algorithm in Section III-A). \( \square \)

**Remark 4.** It is interesting to note here that event-triggering comes at the cost of speed in the sense that retransmissions of identical weights could have potentially allowed the receiving node to learn the weight change earlier (particularly if the delay after a triggering is large, in which case it could be offset by a smaller delay in a subsequent transmission).

IV. SIMULATION RESULTS

In this section, we present simulation results and comparisons for the proposed distributed algorithm. Specifically, we present detailed numerical results for a random graph of size \( n = 20 \) and for the average of 1000 random digraphs of 20 and 50 nodes each. We illustrate the behavior of the proposed distributed algorithm for the following three different scenarios: (i) the scenario where there are no packet drops in the communication links \((v_j, v_i) \in \mathcal{E}\) and each node \(v_j\) transmits the weights \(w_{ij}[k + 1]\) of each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to each \(v_i \in \mathcal{N}_j^+\) at each iteration \(k\), (ii) the scenario where there are packet drops with equal probability \(q\) (where \(0 \leq q < 1\)) for every communication link \((v_j, v_i) \in \mathcal{E}\) and each node \(v_j\) transmits the weights \(w_{ij}[k + 1]\) of each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to each \(v_i \in \mathcal{N}_j^+\) at each iteration \(k\), (iii) the scenario where there are no packet drops at the communication links \((v_j, v_i) \in \mathcal{E}\) and each node \(v_j\) transmits only once the updated weights \(w_{ij}[k + 1]\) of each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to each \(v_i \in \mathcal{N}_j^+\). Each scenario of the proposed distributed algorithm is analyzed in a) the absence of time-delays in the communication links (i.e., \(\tau_j[k] = 0\), \(\forall (v_i, v_j) \in \mathcal{E}\)) and b) the presence of time-delays in the communication links (i.e., \(0 \leq \tau_j[k] \leq \tau_j\), \(\forall (v_i, v_j) \in \mathcal{E}\)).

Note here that in the case where \(\tau_j = 0\), \(\forall (v_i, v_j) \in \mathcal{E}\) we have that \(\overline{w}_{ij} = w_{ij}\) and the proposed distributed algorithm is identical to the algorithm presented in \([22]\), where we consider the transmission between nodes to happen instantaneously.

Figure 5 shows what happens in the case of a randomly created graph of 20 nodes, in which the operation of the proposed distributed algorithm includes no packet drops at the communication links \((v_j, v_i) \in \mathcal{E}\) and each node \(v_j\) transmits the weights \(w_{ij}[k + 1]\) of each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to each \(v_i \in \mathcal{N}_j^+\) at each iteration \(k\). We plot the total delayed imbalance (as defined in Definition 3) as a function of the number of iterations \(k\) for the cases where \(\tau = 0\) (solid line), \(\tau = 10\) (dashed line) and \(\tau_j = \tau = 10\), \(\forall (v_i, v_j) \in \mathcal{E}\) (dashed-dotted line). The plot demonstrates that the proposed distributed algorithm is able to obtain a set of weights that balance the corresponding digraph after a finite number of iterations as argued in the previous section.

Fig. 5. Total delayed imbalance plotted against the number of iterations for a random digraph of 20 nodes in the case where \(\tau = 0\) (solid line), \(0 < \tau < 10\) where \(\tau = 10\) (dashed line) and in the case where \(\tau_j = \tau = 10\) (dashed-dotted line).

Figure 6 shows the same case as Figure 5, with the difference that the operation of the proposed distributed algorithm includes packets drops with equal probability \(q\) (where \(0 < q < 1\)) for every communication link \((v_j, v_i) \in \mathcal{E}\) and each node \(v_j\) transmits the weights \(w_{ij}[k + 1]\) of each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to each \(v_i \in \mathcal{N}_j^+\) at each iteration \(k\). Here, the plot suggest that the proposed distributed algorithm is able to obtain a set of flows that balance the corresponding digraph after a finite number of iterations.

Figure 7 shows the same case as Figures 5 and 6, with the difference being that the operation of the proposed distributed algorithm includes no packet drops at the communication links \((v_j, v_i) \in \mathcal{E}\) but each node \(v_j\) transmits only once the updated weights \(w_{ij}[k + 1]\) of each outgoing edge \((v_i, v_j) \in \mathcal{E}\) to
for the case where there are time-delays in the communication links (i.e., $0 \leq \tau_{ij}[k] \leq \tau_{ij}, \forall (v_i, v_j) \in \mathcal{E}$).

Figure 9 shows the same cases as Figure 8, with the difference being that the network consists of 50 nodes. The performances of the proposed distributed algorithm do not change due to the network size and the conclusions are the same as in Figure 8.

**V. CONCLUSIONS**

We have considered the integer weight balancing problem in a distributed system whose interconnection topology forms a digraph, in the presence of arbitrary (time-varying, inhomogeneous) but bounded time delays that might affect link transmissions. We developed an iterative distributed algorithm and established that it converges to a weight-balanced digraph after a finite number of steps. We have also bounded the execution time of the proposed algorithm as $O(n^6\tau)$, where $n$ is the number of nodes and $\tau$ is the maximum delay in the
digraph, and argued that the resulting weight balanced digraph is unique and independent of how the delays that affect link transmissions materialize. We also added extensions to handle the cases of packet drops over the communication links and event-triggered operation; in both scenarios, we established that the proposed algorithm converges (with probability one) to a set of weights that form a balanced graph after a finite number of iterations, while the resulting weight balanced digraph is unique and independent of how packet drops affect link transmissions.

In the future, we plan to study how these techniques can be adjusted for the case of distributed weight balancing with real weights, in the presence of delays and packet drops, as well as whether all nodes (including nodes with negative imbalance) can participate in the update simultaneously. We will also apply these techniques to quantized average consensus and balancing problems, including possible applications to distributed power control in wireless transmitters with quantized power levels.

REFERENCES


Apostolos I. Rikos received the B.Sc. and M.Sc degrees in Electrical Engineering from the Department of Electrical and Computer Engineering, University of Cyprus in 2010 in 2012, respectively.

Since September 2012, he has been working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Cyprus. His research interests are in the area of distributed coordination and control of multi-agent systems.

Christoforos N. Hadjicostis (M’99, SM’05) received the S.B. degrees in electrical engineering, computer science and engineering, and in mathematics, the M.Eng. degree in electrical engineering and computer science in 1995, and the Ph.D. degree in electrical engineering and computer science in 1999, all from Massachusetts Institute of Technology, Cambridge. In 1999, he joined the Faculty at the University of Illinois at Urbana–Champaign, where he served as Assistant and then Associate Professor with the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Information Trust Institute. Since 2007, he has been with the Department of Electrical and Computer Engineering, University of Cyprus, where he is currently Professor and Dean of Engineering. His research focuses on fault diagnosis and tolerance in distributed dynamic systems, error control coding, monitoring, diagnosis and control of large-scale discrete-event systems, and applications to network security, energy distribution systems, diagnosis, and genetic regulatory models. He currently serves as Associate Editor of IEEE Transactions on Automatic Control, and IEEE Transactions on Automation Science and Engineering; he has also served as Associate Editor of IEEE Transactions on Control Systems Technology, and IEEE Transactions on Circuits and Systems I.