Distributed Average Consensus in Open Multi-Agent Systems

Christoforos N. Hadjicostis and Alejandro D. Domínguez-García

Abstract— **In this paper, we consider the problem of distributed average consensus in multi-agent systems, where each agent can come in or move out of the system, possibly multiple times. In the literature, such systems are referred to as open multi-agent systems. A typical goal in such settings is to use an iterative distributed algorithm to calculate the average of some quantities of interest each agent possesses, which can be crucial in many estimation, control, or optimization applications. We consider an open multi-agent setting and propose a distributed algorithm that allows the participating agents to track their average. More specifically, if the set of agents remaining in the computation eventually settles to a certain subset of agents, then the proposed algorithm allows them (under some mild connectivity conditions) to asymptotically reach consensus to the average of the quantities of interest these remaining agents hold. Analysis and numerical examples to illustrate the operation of the proposed algorithm are also provided.**
Keywords: Distributed averaging, open multi-agent systems, dynamic average consensus.

I. Introduction and Motivation

Multi-agent systems (MAS) consist of a set of agents that can share information with neighboring agents via connection links that form a generally directed interconnection topology [1], [2]. In an open multi-agent system (OMAS), agents are allowed to come in or move out of the system at different points in time [3]. The time-varying nature of the set of agents poses additional challenges in the management of OMAS and has prompted significant research in recent years (see, e.g., [4]–[6]).

In this paper, we consider an average consensus problem in an OMAS where each agent (allowed to come in or move out of the distributed computation, perhaps multiple times) holds a value for some quantity of interest (e.g., a sensor measurement, a vote, an opinion, etc.). In particular, we consider two different modes in which de-activated agents can re-activate themselves: (i) they simply re-initiate their state variables (as if they start from scratch); (ii) they retain their state variables at the values they had at the last point they were active. We propose a distributed algorithm that can handle either type of re-activation, and which allows the set of agents that remain in the computation to asymptotically compute the average of the values they hold. In particular, if the set of agents that remain in the computation stabilizes to a fixed set (at least for a sufficiently large number of iterations), these remaining agents are able, under some mild connectivity conditions, to reach consensus to the average value of the quantities of interest they possess. This should be contrasted to more traditional settings, in which the set of agents remains fixed but the interconnection topology might be time-varying (see, for example, approaches to reach average consensus under a switching topology in [2]).

The proposed scheme is a modified version of the *running-sum ratio consensus* algorithm [7], which normally operates over a static network (fixed set of agents and fixed set of connections) and is used mainly to overcome network-induced errors, such as delays and packet drops on the communication links [8], [9]. The main approach also shares some of the features of the distributed algorithm for trustworthy average consensus in [10], [11], which considers a MAS (under a fixed set of agents and a time-invariant interconnection topology), and allows the agents to reach asymptotic average consensus in the presence of malicious agents.

The proposed algorithm is also related to techniques for adding and removing agents from a distributed average consensus computation using the push-sum algorithm (e.g., [9], [12]). Unlike the work in this paper, however, the aforementioned works focus on agents that willingly remove themselves (e.g., by collaborating with their neighboring agents). Instead, in this paper, we use techniques from [10], [11] to handle the addition/removal of (re-)activated/deactived agents. The mere (re-)appearance/absence of an agent triggers an adjustment procedure performed in a distributed manner by its neighbors, which is crucial for ensuring convergence to the *exact* average of the values of the remaining nodes. This should be contrasted with the approach in [4], which guarantees that, if the set of remaining nodes stabilizes, the nodes will track the average with bounded variation (but not necessarily converge to the exact average).

Apart from differences in the underlying setting and motivation discussed above, the main theoretical contribution of this paper is that it extends the invariant properties of the running-sum ratio consensus algorithm in [10], [11] to a setting where agents can come in and move out at different points in time. Specifically, agents that have been active for two consecutive steps are shown to obey a key invariant property.

II. Graph-Theoretic OMAS Model

We can conveniently capture all agents and links that are active at some point during the operation of a given OMAS

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering at the University of Cyprus, Nicosia, Cyprus, and also with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: hadjicostis.christoforos@ucy.ac.cy.

Alejandro D. Domínguez-García is with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: aledan@illinois.edu.

by a nominal digraph of order M ($M \geq 2$), defined as $\mathcal{G}_N = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$ is the universe (set) of all agents (nodes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the (universe) set of all communication links (edges). A directed edge from node v_i to node v_j is denoted by $(v_j, v_i) \in \mathcal{E}$, and indicates that, when the link is active (e.g., when both nodes v_j and v_i are active), node v_i can send information to node v_j.

The nominal digraph $\mathcal{G}_N = (\mathcal{V}, \mathcal{E})$ is assumed to be *strongly connected*, i.e., 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. All nodes that can send information to node v_j directly are said to be the in-neighbors of node v_j and belong to the set $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^-$ is called the *in-degree* of v_j and is denoted by $D_j^- = |\mathcal{N}_j^-|$. The nodes that can receive information from node v_j comprise its out-neighbors and are denoted by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$ is called the *out-degree* of v_j and is denoted by $D_j^+ = |\mathcal{N}_j^+|$.

We consider a discrete-time setup, where at each time step k, $k = 0, 1, 2, ...$, we can capture the active nodes and links of a given OMAS by a digraph $\mathcal{G}[k] = (\mathcal{V}[k], \mathcal{E}[k])$, where $\mathcal{V}[k] \subseteq \mathcal{V}$ and $\mathcal{E}[k] = (\mathcal{V}[k] \times \mathcal{V}[k]) \cap \mathcal{E}$. We refer to $\mathcal{V}[k]$ as the subset of *active nodes* at iteration k and to $\mathcal{E}[k]$ as the subset of *active edges* at iteration k (note that non-active nodes cannot have any edges associated with them). Effectively, $\mathcal{G}[k]$ is the digraph induced from $\mathcal{G}_N$ by restricting attention to the set of active nodes $\mathcal{V}[k]$.

All nodes that, at iteration k, can send information to active node v_j directly are said to be in-neighbors of node v_j at iteration k, and belong to the set $\mathcal{N}_j^-[k] = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}[k]\}$ (which satisfies $\mathcal{N}_j^-[k] = \mathcal{N}_j^- \cap \mathcal{V}[k]$). Similarly, all nodes that, at iteration k, can receive information from node v_j directly are said to be out-neighbors of node v_j at iteration k, and belong to the set $\mathcal{N}_j^+[k] = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}[k]\}$ (which satisfies $\mathcal{N}_j^+[k] = \mathcal{N}_j^+ \cap \mathcal{V}[k]$). We also define the in- and out-degree of each node v_j at iteration k as $D_j^-[k] = |\mathcal{N}_j^-[k]|$ and $D_j^+[k] = |\mathcal{N}_j^+[k]|$.

We assume a simple wireless broadcast communication model. Specifically, when (active) node v_j broadcasts information at iteration k, its transmissions are received at all of its (active) out-neighbors in the set $\mathcal{N}_j^+[k]$; similarly, (active) node v_j receives all transmissions made by each of its (active) in-neighbors in the set $\mathcal{N}_j^-[k]$.

III. A Primer on Running-Sum Ratio Consensus

Consider a MAS captured by a *fixed* (time-invariant) digraph $\mathcal{G}_N = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ possesses some value x_j. Running-sum ratio consensus is a distributed algorithm that allows the nodes to asymptotically calculate

$$\overline{X} = \frac{1}{M} \sum_{\ell=1}^{M} x_\ell \; . \tag{1}$$

To execute the running-sum ratio consensus algorithm, each node v_j maintains two state variables $y_j[k]$ and $z_j[k]$ (initialized to $y_j[0] = x_j$ and $z_j[0] = 1$), and updates them, at iteration k ($k \geq 0$), as follows

$$y_j[k+1] \;\; = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k]) \; , \tag{2}$$

$$z_j[k+1] \;\; = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\nu_{ji}[k+1] - \nu_{ji}[k]) \; , \tag{3}$$

where ρ_{ji} and ν_{ji} (initially, $\rho_{ji}[0] = \nu_{ji}[0] = 0$ for $v_i \in \mathcal{N}_j^- \cup \{v_j\}$) satisfy $\rho_{ji}[k] = \sigma_j[k]$ and $\nu_{ji}[k] = \eta_j[k]$ with

$$\sigma_j[k+1] \;\; := \;\; \sum_{t=0}^{k} y_j[t]/(1 + D_j^+) \; ,$$

$$\eta_j[k+1] \;\; := \;\; \sum_{t=0}^{k} z_j[t]/(1 + D_j^+) \; .$$

At each $k \geq 0$, each node v_j can calculate the ratio $r_j[k] := y_j[k]/z_j[k]$. Then, as $\mathcal{G}_N$ is strongly connected, it can be shown (see, e.g., [7]) that the $r_j[k]$'s asymptotically converge to the average $\overline{X}$ in (1); specifically,

$$\lim_{k \to \infty} r_j[k] = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \overline{X} \; , \quad \forall v_j \in \mathcal{V} \; . \tag{4}$$

Note that, in order to implement running-sum ratio consensus, each node v_j needs to (i) track its own (*outgoing*) running-sums, i.e., $\sigma_j[k+1]$ and $\eta_j[k+1]$ that it sends to its out-neighbors, and (ii) maintain the values of the (*incoming*) running-sums for each of its in-neighbors, i.e., $\rho_{ji}[k]$ and $\nu_{ji}[k]$ for each in-neighbor $v_i \in \mathcal{N}_j^-$. Notice that the updates for y and z effectively reduce to

$$y_j[k+1] \;\; = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} y_i[k]/(1 + D_i^+), \tag{5}$$

$$z_j[k+1] \;\; = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} z_i[k]/(1 + D_i^+), \tag{6}$$

where $y_j[0] = x_j$ and $z_j[0] = 1$, for $v_j \in \mathcal{V}$. The above is the standard ratio consensus algorithm [7].

Notice that both running-sum ratio consensus and ratio consensus require each node v_j to be aware of its out-degree D_j^+. Indirectly, running-sum ratio consensus also assumes that each node is able to associate an incoming message with the node that sends it (e.g., via a unique identifier). The additional bookkeeping introduced by the running-sum ratio consensus has advantages in cases where transmissions may be delayed or dropped [8], [9].

IV. Distributed Averaging in OMAS

Setting S1. Consider an OMAS with a nominal digraph $\mathcal{G}_N = (\mathcal{V}, \mathcal{E})$. Consider a sequence of digraphs $\mathcal{G}[k] = (\mathcal{V}[k], \mathcal{E}[k])$, where $\mathcal{V}[k] \subseteq \mathcal{V}$ and $\mathcal{E}[k] = (\mathcal{V}[k] \times \mathcal{V}[k]) \cap \mathcal{E}$, for $k = 0, 1, 2, ...$

Each node $v_j \in \mathcal{V}$ possesses a value x_j and the nodes execute a distributed iterative algorithm that obeys the constraints imposed by the time-varying communication topology (active nodes can only transmit information to their active out-neighbors at iteration k and can only receive information from their active in-neighbors at iteration k).

The objective is to design such a distributed algorithm so as to enable the nodes that remain in the computation to obtain the average of their x_j's. For this reason, we make the following assumption.

Assumption A1. There exists a finite time k_0 such that

$$\mathcal{V}[k] = \mathcal{V}_R \ , \forall k \geq k_0,$$

where $\mathcal{V}_R$ is the set of remaining nodes.

OMAS Average Consensus Problem: Following a distributed iterative algorithm that respects the communication constraints at each iteration imposed in Setting S1, we want the remaining nodes $\mathcal{V}_R$ to reach consensus to the average

$$\overline{X}_R = \frac{\sum_{v_l \in \mathcal{V}_R} x_l}{|\mathcal{V}_R|} \ , \tag{7}$$

where $|\mathcal{V}_R|$ is the cardinality of the set $\mathcal{V}_R$.

Since the set of active nodes stabilizes after iteration k_0 in Assumption A1, the set of active edges also stabilizes after iteration k_0. In other words, $\mathcal{E}[k] = \mathcal{E}_R = (\mathcal{V}_R \times \mathcal{V}_R) \cap \mathcal{E}$ for $k \geq k_0$, so that the distributed computation is performed over a communication topology captured by the digraph $\mathcal{G}_R = (\mathcal{V}_R, \mathcal{E}_R)$. The assumption below is rather standard for reaching average consensus in any fixed topology.

Assumption A2. Digraph $\mathcal{G}_R = (\mathcal{V}_R, \mathcal{E}_R)$ where $\mathcal{E}_R = \{(v_j, v_i) \in \mathcal{E} \mid v_j, v_i \in \mathcal{V}_R\}$ is strongly connected.

Remark 1: Note that k_0 is *not* known to the nodes. Therefore, the challenge for solving the above problem is to be able to add/remove nodes systematically so that, once the network settles to $\mathcal{G}_R$, the remaining nodes are in position to reach average consensus. In practice, nodes only have to settle for an interval of time that is sufficiently long in order for them to reach consensus to $\overline{X}_R$ (see the numerical example at the end of the paper).

A. Running-Sum Based Averaging in OMAS

Assumption A3. At each iteration, each active node $v_j \in \mathcal{V}[k]$ is aware of its out-degree $D_j^+[k]$. Furthermore, each transmission is associated with a unique ID that allows the receiving node to identify the sender of the message.

Remark 2: The requirement about knowledge of the out-degree is rather mild when connections between pairs of nodes are bidirectional. In the case of directional connections, this requirement has frequently been used (see, e.g., [13]) but can also be relaxed in various ways. We adopt it at this point to keep the description of the distributed algorithm for calculating $\overline{X}_R$ simple.

The distributed calculation of the average $\overline{X}_R$ in (7) is based on a variation of the running-sum ratio consensus algorithm in (2)–(3), where each (active) node v_j uses the number of its out-neighbors at iteration k (given by $D_j^+[k] = |\mathcal{N}_j^+[k]|$) to update its y- and z-running sums as follows:

$$\sigma_j[k+1] \ := \ \sum_{t=0}^{k} y_j[t]/(1 + D_j^+[t]) \ , \tag{8}$$

$$\eta_j[k+1] \ := \ \sum_{t=0}^{k} z_j[t]/(1 + D_j^+[t]) \ . \tag{9}$$

Note that these sums can be easily updated iteratively since $\sigma_j[k+1] = \sigma_j[k] + y_j[k]/(1 + D_j^+[k])$ and $\eta_j[k+1] = \eta_j[k] + z_j[k]/(1 + D_j^+[k])$.

As in the case of the standard running-sum ratio consensus, this variation requires each node v_j to (i) track the running-sum values $\sigma_j[k]$ and $\eta_j[k]$ it broadcasts (updated as described in (8)–(9)), and (ii) maintain, for each in-neighbor $v_i \in \mathcal{N}_j^-$, two incoming running sums, namely $\rho_{ji}[k]$, which keeps track of the y-running sum broadcast by node v_i, with $\rho_{ji}[0] = 0$, and $\nu_{ji}[k]$, which keeps track of the z-running sum broadcast by node v_i, with $\nu_{ji}[0] = 0$. Of course, this assumes that each node is able to associate an incoming message with the node that sends it (e.g., via a unique ID as stated in Assumption A3).

With these modifications, node v_j updates its y and z values using only the running sums it receives from its in-neighbors at iteration k, i.e.,

$$y_j[k+1] \ = \ \sum_{v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k]), \tag{10}$$

$$z_j[k+1] \ = \ \sum_{v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}} (\nu_{ji}[k+1] - \nu_{ji}[k]), \tag{11}$$

where (apart from some exceptional conditions discussed in the next section) $\rho_{ji}[k] = \sigma_i[k]$ and $\nu_{ji}[k] = \eta_i[k]$.

At this point, it should be clear that, in the long run, the nodes that are not in $\mathcal{V}_R$ are excluded from the distributed computation (since they become inactive after iteration k_0 in Assumption A1). In fact, Assumption A2 ensures that, after iteration k_0, the distributed computation is performed over the strongly connected digraph $\mathcal{G}_R$. Thus, if $k_0 = 0$, then it follows from the discussion in Section III that the algorithm will converge to the average $\overline{X}_R$. Therefore, the main challenge is that nodes can become active or inactive multiple times during the first k_0 iterations and one needs to ensure that, at the point active nodes eventually stabilize to $\mathcal{V}_R$ (iteration k_0), the values maintained at the remaining nodes guarantee convergence to $\overline{X}_R$.

Note that it is not really relevant for the proposed algorithm how inactive nodes behave because, in any case, they are excluded from the computation. However, if an inactive node becomes active again, we need to specify how re-activation takes place.

Re-Activation Mode RM. When an inactive node v_j becomes re-activated, it either (i) re-initializes all of its state variables as if it is starting from scratch (RM1), or (ii) retains its state variables at the values they had when node v_j was last active (RM2).

B. Adding and Removing Active/Inactive Nodes

In order to perform the calculation of the average, $\overline{X}_R$, nodes that are active at a given iteration need a way to add or remove the effects of inactive nodes. The main way this is achieved is by adjusting their out-degree and ignoring in-neighbors that are inactive, as described in (10)–(11). However, this is not enough because active nodes need to take corrective actions when a previously active

neighboring node becomes inactive or vice-versa; moreover, this corrective action depends on whether the neighboring node that changed status is an in-neighbor or an out-neighbor.

We describe what active nodes do in four different cases from the point of view of node v_j, assumed active at the current iteration (k) as well as the previous iteration ($k-1$).

Case 1. *Previously active out-neighbor v_l becomes inactive:* In this case, node v_j needs to compensate for all transmissions that it sent to node v_l while it was active. It does so by adding to its y and z values its own y-running sum and z-running sum:

$$\begin{aligned} y_j[k] &:= y_j[k] + \sigma_j[k] \ , \\ z_j[k] &:= z_j[k] + \eta_j[k] \ . \end{aligned}$$

The reason is that $\sigma_j[k]$ ($\eta_j[k]$) represents the cumulative y (z) values that were sent to the now inactive out-neighbor v_l (this value is the same for all out-neighbors at any given time). This adjustment essentially "diverts" all messages sent to node v_l up to time instant $k-1$, back to node v_j.

Note that if multiple out-neighbors become inactive at iteration k, this adjustment has to be performed for each such out-neighbor. Thus, letting $\Delta I_j^+[k] = |\mathcal{N}_j^+[k-1] \setminus \mathcal{N}_j^+[k]|$ be the number of previously active out-neighbors of node v_j that become inactive at iteration k, we have that

$$\begin{aligned} y_j[k] &:= y_j[k] + \Delta I_j^+[k] \sigma_j[k] \ , \\ z_j[k] &:= z_j[k] + \Delta I_j^+[k] \eta_j[k] \ . \end{aligned}$$

Case 2. *Previously inactive out-neighbor v_l becomes active:* This is effectively the reverse of the previous case: node v_j needs to account for all the cumulative values that node v_l is now receiving from v_j. Node v_j can achieve this adjustment by subtracting from its own y and z values, its corresponding y-running sum and z-running sum:

$$\begin{aligned} y_j[k] &:= y_j[k] - \sigma_j[k] \ , \\ z_j[k] &:= z_j[k] - \eta_j[k] \ . \end{aligned}$$

Again, as in Case 1, the above adjustment has to be performed for each out-neighbor that becomes active, i.e., if we let $\Delta A_j^+[k] = |\mathcal{N}_j^+[k] \setminus \mathcal{N}_j^+[k-1]|$ be the number of previously inactive out-neighbors of node v_j that become active at iteration k, then

$$\begin{aligned} y_j[k] &:= y_j[k] - \Delta A_j^+[k] \sigma_j[k] \ , \\ z_j[k] &:= z_j[k] - \Delta A_j^+[k] \eta_j[k] \ . \end{aligned}$$

Overall, Case 1 and Case 2 can be combined by letting $\Delta D_j^+[k] := -(\Delta I_j^+[k] - \Delta A_j^+[k])$, and having node v_j set

$$\begin{aligned} \tilde{y}_j[k] &:= y_j[k] - \Delta D_j^+[k] \sigma_j[k] \ , \\ \tilde{z}_j[k] &:= z_j[k] - \Delta D_j^+[k] \eta_j[k] \ , \end{aligned}$$

and then update its $\sigma_j[k+1]$ and $\eta_j[k+1]$ as

$$\begin{aligned} \sigma_j[k+1] &:= \sigma_j[k] + \tilde{y}_j[k]/(1+D_j^+[k]) \ , \\ \eta_j[k+1] &:= \eta_j[k] + \tilde{z}_j[k]/(1+D_j^+[k]) \ . \end{aligned}$$

Case 3. *Previously active in-neighbor v_i becomes inactive:* In this case, apart from ignoring node v_i's transmissions, node v_j has to also remove the effect of all previous transmissions that it received from node v_i. This can be done by subtracting from its y and z values the corresponding y-running sum and z-running sum. In addition, node v_j has to set ρ_{ji} and ν_{ji} to zero in order to ensure that, if node v_i becomes active again in the future, node v_i's effect on the computation will be properly accounted for. In summary, for every node in the set $\Delta \mathcal{I}_j^-[k] = \mathcal{N}_j^-[k-1] \setminus \mathcal{N}_j^-[k]$, the following steps are taken by node v_j:

$$\begin{aligned} y_j[k] &:= y_j[k] - \rho_{ji}[k] \ , \ \rho_{ji}[k] := 0 \ , \\ z_j[k] &:= z_j[k] - \nu_{ji}[k] \ , \ \nu_{ji}[k] := 0 \ . \end{aligned}$$

In other words, node v_j sets

$$\begin{aligned} y_j[k] &:= y_j[k] - \sum_{v_i \in \Delta \mathcal{I}_j^-[k]} \rho_{ji}[k] \ , \\ z_j[k] &:= z_j[k] - \sum_{v_i \in \Delta \mathcal{I}_j^-[k]} \nu_{ji}[k] \ . \end{aligned}$$

and then sets $\rho_{ji}[k] = \nu_{ji}[k] = 0$ for all $v_i \in \Delta \mathcal{I}_j^-[k]$.

Case 4. *Previously inactive in-neighbor v_i becomes active:* In this case, node v_j needs to simply incorporate the effect of node v_i's transmissions: since $\rho_{ji}[k]$ and $\nu_{ji}[k]$ have been set to zero in previous iterations, all node v_j needs to do is to simply incorporate the values of the y-running sum and the z-running sum it receives from node v_i into its update in (10) and (11) with

$$\begin{aligned} \rho_{ji}[k+1] &:= \sigma_i[k+1] \ , \\ \nu_{ji}[k+1] &:= \eta_i[k+1] \ . \end{aligned}$$

Note that Cases 3 and 4 can be easily handled by having node v_j simply set $\rho_{ji}[k+1] = 0$ and $\nu_{ji}[k+1] = 0$ for all in-neighbors v_i that are inactive at iteration k, and then update $y_j[k+1]$ and $z_j[k+1]$ over all in-neighbors (active and inactive). In other words, node v_j first sets

$$\rho_{ji}[k+1] = \begin{cases} \sigma_i[k+1] \ , & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\} \ , \\ 0 \ , & \text{otherwise,} \end{cases}$$

$$\nu_{ji}[k+1] = \begin{cases} \eta_i[k+1] \ , & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\} \ , \\ 0 \ , & \text{otherwise,} \end{cases}$$

and then updates $y_j[k+1]$ and $z_j[k+1]$ as

$$\begin{aligned} y_j[k+1] &= \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k]) \ , \\ z_j[k+1] &= \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\nu_{ji}[k+1] - \nu_{ji}[k]) \ . \end{aligned}$$

V. Proof of Correctness

The pseudocode for the algorithm described in the previous section is presented as Algorithm 1 from the perspective of node v_j. Note that all nodes are assumed to be active initially but this is not critical. The status of each node v_j is captured by the binary variable $a_j[k]$, which equals "1" if node v_j is active at iteration k ("0" otherwise), and it

is initialized to $a_j[-1] = 1$. Similarly, the assumption that each node v_j knows the nominal set of its in-neighbors $\mathcal{N}_j^-$ is done for ease of the algorithmic description. In reality, at each time step k when node v_j receives a message from a new neighbor (i.e., in-neighbor v_i that becomes active for the first time at iteration k), node v_j can simply add v_i to its list of (known) neighbors (this is easy to do under Assumption A3). It is also worth pointing out that, following Assumption A3, each active node v_j ($v_j \in \mathcal{V}[k]$) is aware, at iteration k, of its out-degree $D_j^+[k]$ and receives messages from its in-neighbors (so it can identify the set $\mathcal{N}_j^-[k]$). Note that either Lines 6–7 or Lines 8-9 are executed by node v_j when it gets re-activated at iteration k.

Theorem 1: Consider Setting S1. Let x_j be the value of the quantity possessed by node v_j and assume that each node v_j executes Algorithm 1. Then, at the beginning of each iteration $k = 0, 1, ...$, it holds that

$$y_j[k] + D_j^+[k-1]\sigma_j[k] - \sum_{v_i \in \mathcal{N}_j^-[k-1]} \sigma_i[k] = x_j, \quad (12)$$

$$z_j[k] + D_j^+[k-1]\eta_j[k] - \sum_{v_i \in \mathcal{N}_j^-[k-1]} \eta_i[k] = 1, \quad (13)$$

for each node v_j that has been active for at least two consecutive time steps (i.e., $v_j \in \mathcal{V}[k-1] \cap \mathcal{V}[k-2]$ with $\mathcal{V}[-1] = \mathcal{V}[-2] = \mathcal{V}$).

The proof of the theorem uses techniques from [14] and is omitted due to space considerations.

Theorem 2: Consider Setting S1. Let x_j be the value of the quantity held at node v_j and assume that each node v_j executes Algorithm 1 under Re-activation Mode RM1 or RM2. Let x_j be the value of the quantity held at node v_j and assume that each node v_j executes Algorithm 1. Under Assumptions A1–A3 and Re-Activation Modes RM1 or RM2, the remaining nodes in $\mathcal{V}_R$ converge to $\overline{X}_R$.

Proof: Let k_0 be the constant in Assumption A1. For $k \geq k_0$, the remaining nodes execute the ratio consensus algorithm over digraph $\mathcal{G}_R$. Since $\mathcal{G}_R$ is strongly connected under Assumption A2, the ratios $r_j[k] = y_j[k]/z_j[k]$ of all remaining nodes will converge to

$$\lim_{k\to\infty} r_j[k] = \frac{\sum_{v_l \in \mathcal{V}_R} y_l[k_0'+1]}{\sum_{v_l \in \mathcal{V}_R} z_l[k_0'+1]}, \quad \forall v_j \in \mathcal{V}_R, \quad (14)$$

where $k_0' \geq k_0$. We next argue that $\sum_{v_l \in \mathcal{V}_R} y_l[k_0'+1] = \sum_{v_l \in \mathcal{V}_R} x_l$ and $\sum_{v_l \in \mathcal{V}_R} z_l[k_0'+1] = |\mathcal{V}_R|$.

Using the invariant in Theorem 1 at time $k_0' = k_0 + 1$ (so that all nodes in $\mathcal{V}_R$ have been active for at least two time steps), we have that, for each $v_j \in \mathcal{V}_R$, it holds that

$$y_j[k_0'+1] + D_j^+[k_0']\sigma_j[k_0'+1] - \sum_{v_i \in \mathcal{N}_j^-[k_0']} \sigma_i[k_0'+1] = x_j,$$

where $D_j^+[k_0']$ ($\mathcal{N}_j^-[k_0']$) is the out-degree (set of in-neighbors) of node v_j in the digraph $\mathcal{G}_R$. If we sum all such equations over all v_j in $\mathcal{V}_R$, on the right, we have the sum $\sum_{v_j \in \mathcal{V}_R} x_j$, whereas on the left we have

$$\sum_{v_j \in \mathcal{V}_R} y_j[k_0'+1] + \sum_{v_j \in \mathcal{V}_R} (D_j^+[k_0']\sigma_j[k_0'+1]) - \\ - \sum_{v_j \in \mathcal{V}_R} \left(\sum_{v_i \in \mathcal{N}_j^-[k_0']} \sigma_i[k_0'+1] \right) = \\ = \sum_{v_j \in \mathcal{V}_R} y_j[k_0'+1],$$

where the simplification occurs because, for each link (v_j, v_i) of $\mathcal{G}_R$, we have σ_j appearing twice, once with a positive sign and once with a negative sign. This establishes that $\sum_{v_l \in \mathcal{V}_R} y_l[k_0'+1] = \sum_{v_l \in \mathcal{V}_R} x_l$. One can similarly establish that $\sum_{v_l \in \mathcal{V}_R} z_l[k_0'+1] = |\mathcal{V}_R|$. ■

Algorithm 1: OMAS Distributed Averaging

1 **Input:** Node v_j knows x_j, $\mathcal{N}_j^+$, $\mathcal{N}_j^-$
Initialization:
2 Set $a_j[-1] = 1$, $D_j^+[-1] = D_j^+$, $y_j[0] = x_j$, $\sigma_j[0] = 0$, $z_j[0] = 1$, $\eta_j[0] = 0$
3 Set, $\forall v_i \in \mathcal{N}_j^-$, $\rho_{ji}[0] = \nu_{ji}[0] = 0$

for $k \geq 0$:

4 Receive $a_j[k]$ and set $\mathcal{N}_j^+[k]$ and $\mathcal{N}_j^-[k]$
Under RM1: If ($a_j[k] = 1$ **and** $a_j[k-1] = 0$)
5 Re-initialize $y_j[k+1] = x_j$, $z_j[k+1] = 1$, $\sigma_j[k+1] = 0$, $\eta_j[k+1] = 0$
Reset, for $\forall v_i \in \mathcal{N}_j^-$, $\rho_{ji}[k+1] = \nu_{ji}[k+1] = 0$
Under RM2: If ($a_j[k] = 0$ **or** ($a_j[k] = 1$ **and** $a_j[k-1] = 0$))
Maintain $D_j^+[k] = D_j^+[k-1]$, $y_j[k+1] = y_j[k]$, $z_j[k+1] = z_j[k]$, $\sigma_j[k+1] = \sigma_j[k]$, $\eta_j[k+1] = \eta_j[k]$,
Maintain, for $\forall v_i \in \mathcal{N}_j^-$, $\rho_{ji}[k+1] = \rho_{ji}[k]$, $\nu_{ji}[k+1] = \nu_{ji}[k]$
If ($a_j[k] = 1$ **and** $a_j[k-1] = 1$)
Calculate Out-Degree Change:
6 Set $D_j^+[k] = |\mathcal{N}_j^+[k]|$
7 Set $\Delta D_j^+[k] = D_j^+[k] - D_j^+[k-1]$
8 Cases 1 and 2:
9 $\tilde{y}_j[k] := y_j[k] - \Delta D_j^+[k]\sigma_j[k]$
10 $\tilde{z}_j[k] := z_j[k] - \Delta D_j^+[k]\eta_j[k]$
Compute:
11 $\sigma_j[k+1] = \sigma_j[k] + \tilde{y}_j[k]/(1 + D_j^+[k])$
12 $\eta_j[k+1] = \eta_j[k] + \tilde{z}_j[k]/(1 + D_j^+[k])$
13 **Broadcast:** $\sigma_j[k+1]$, $\eta_j[k+1]$ to all $v_l \in \mathcal{N}_j^+[k]$
14 **Receive:** $\sigma_i[k+1]$, $\eta_i[k+1]$ from $v_i \in \mathcal{N}_j^-[k]$
Update Running Sums of Active In-Neighbors:
15 Cases 3 and 4:
16 For each $v_i \in \mathcal{N}_j^-$ set
$\rho_{ji}[k+1] = \begin{cases} \sigma_i[k+1], & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\} \\ 0, & \text{otherwise} \end{cases}$
17 $\nu_{ji}[k+1] = \begin{cases} \eta_i[k+1], & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\} \\ 0, & \text{otherwise} \end{cases}$
Compute:
18 $y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k])$
19 $z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\nu_{ji}[k+1] - \nu_{ji}[k])$
20 **Output:** $r_j[k+1] = y_j[k+1]/z_j[k+1]$

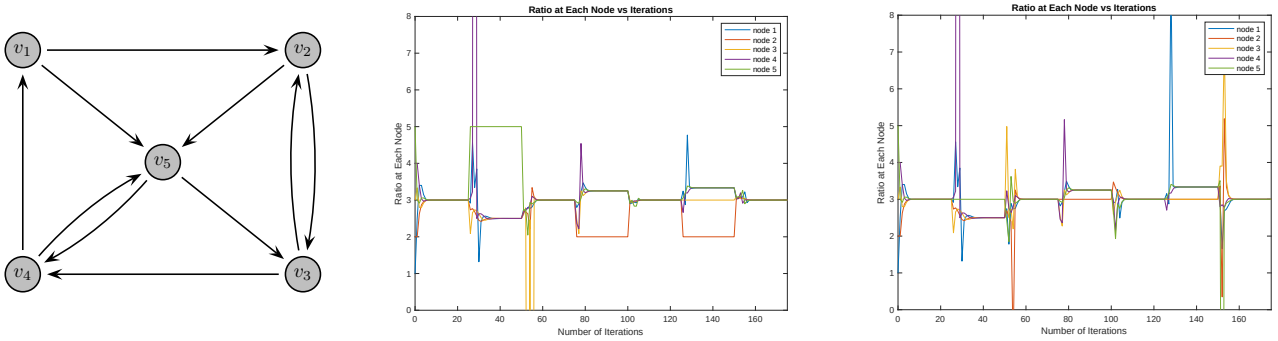

Fig. 1. Digraph considered in the example (left) and ratios of nodes plotted against the number of iterations k under Re-Activation Mode RM1 (middle) and Re-Activation Mode RM2 (right).

VI. Numerical Experiments

Consider the OMAS nominal digraph $\mathcal{G}_N = (\mathcal{V}, \mathcal{E})$ on the left of Fig. 1, with node initial values given by $x_i = i$ for $i = 1, 2, ..., 5$. We consider the following scenario to illustrate the operation of the proposed OMAS distributed averaging algorithm.

Period 1, $k = 0$ to $k = 25$: all nodes are active;
Period 2, $k = 26$ to $k = 50$: node v_5 becomes inactive;
Period 3, $k = 51$ to $k = 75$: all nodes are active;
Period 4, $k = 76$ to $k = 100$: node v_2 becomes inactive;
Period 5, $k = 101$ to $k = 125$: all nodes are active;
Period 6, $k = 126$ to $k = 150$, nodes v_2, v_3 become inactive.
Period 7, $k = 150$ onwards, all nodes are active.

In the middle of Fig. 1 we plot the ratios obtained at each node against the number of iteration k assuming that the nodes run Algorithm 1 with Re-Activation Mode RM1. We see that by the end of Period 1, the ratios have converged to $\overline{X} = 3$ (since all nodes are active). During Period 2, node v_5 is inactive and its ratio remains at 5, which is its initial ratio; however, the remaining nodes converge to 2.5 (which is the average of their initial values $\{1, 2, 3, 4\}$). By the end of Period 3, when all nodes are again active, the ratios are back to $\overline{X} = 3$. During Period 4, node v_2 is inactive and its ratio stays at its initial ratio of 2, whereas the remaining four nodes converge to 3.25 (which is the average of their initial values $\{1, 3, 4, 5\}$). At the end of Period 5, all nodes become active and the nodes converge back to $\overline{X} = 3$. During Period 6, nodes v_2 and v_3 are inactive and their ratios remain at their initial values (2 and 3, respectively), whereas the other three nodes converge to 3.3333 (i.e., the average of their initial values $\{1, 4, 5\}$). Finally, at the end of Period 7, the nodes are back to $\overline{X} = 3$. It worth pointing out that, in each period, the digraphs $\mathcal{G}_R$ which we get (thinking of the period as extending indefinitely) are strongly connected.

On the right of Fig. 1 we plot the ratios obtained at each node against the number of iteration k assuming that the nodes run Algorithm 1 with Re-Activation Mode RM2. The main difference is that nodes that become inactive remain at the ratios they had at the point they became inactive.

VII. Future Work

In the future, we plan to develop a version of the proposed algorithm that can operate over time-varying interconnections and is resilient to packet drops and transmission delays. We also intend to consider variations that would allow the nodes to smooth out the effect of changes in the status of nodes. Other interesting future extensions include the development of algorithms that compute the average of all values (including nodes that entered the computation at some point in time but eventually left), and incorporating trust and reputation mechanisms in the OMAS computation.

References

[1] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[2] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[3] C. M. de Galland and J. M. Hendrickx, "Fundamental performance limitations for average consensus in open multi-agent systems," *IEEE Trans. on Automatic Control*, vol. 68, no. 2, pp. 646–659, 2023.

[4] M. Franceschelli and P. Frasca, "Proportional dynamic consensus in open multi-agent systems," in *Proc. of 57th IEEE Conference on Decision and Control (CDC)*, 2018, pp. 900–905.

[5] V. S. Varma, I.-C. Morărescu, and D. Nešić, "Open multi-agent systems with discrete states and stochastic interactions," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 375–380, 2018.

[6] D. Deplano, M. Franceschelli, and A. Giua, "Dynamic min and max consensus and size estimation of anonymous multiagent networks," *IEEE Trans. on Automatic Control*, vol. 68, no. 1, pp. 202–213, 2021.

[7] C. N. Hadjicostis, A. D. Domínguez-García, and T. Charalambous, "Distributed averaging and balancing in network systems, with applications to coordination and control," *Foundations and Trends® in Systems and Control*, vol. 5, no. 3–4, 2018.

[8] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Trans. on Automatic Control*, vol. 59, no. 3, pp. 763–768, 2014.

[9] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Trans. on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2016.

[10] C. N. Hadjicostis and A. D. Domínguez-García, "Trustworthy distributed average consensus," in *Proc. of 61st IEEE Conference on Decision and Control (CDC)*, 2022, pp. 7403–7408.

[11] ——, "Trustworthy distributed average consensus based on locally assessed trust evaluations," *IEEE Trans. on Automatic Control*, 2024.

[12] I. Eyal, I. Keidar, and R. Rom, "LiMoSense: live monitoring in dynamic sensor networks," *Distributed Computing*, vol. 27, no. 5, pp. 313–328, 2014.

[13] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014.

[14] C. N. Hadjicostis and A. D. Domínguez-García, "Invariant properties of linear-iterative distributed averaging algorithms and application to error detection," *European Journal of Control*, p. 101035, 2024.