Shadow Banning in Directed Graphs for Fault-Tolerant Averaging

Christoforos N. Hadjicostis and Alejandro D. Domínguez-García

***Abstract*— We consider the problem of average consensus in a distributed system wherein a subset of the nodes comprising the system becomes faulty. To address this problem, we rely on a class of averaging algorithms whereby each node maintains a collection of variables, which it iteratively updates as a linear combination of other variables received by the node from its in-neighbors, i.e., nodes from which it receives information directly. Algorithms within this class exhibit, throughout their execution, certain invariance properties that are local to each node and reflect the conservation of certain quantities capturing an aggregate of all the values received by a node from its in-neighbors and all the values sent by said node to its out-neighbors, i.e., nodes that receive its information directly. In this paper, we exploit one such invariance property to enable each node to check whether or not any of its in-neighbors becomes faulty during the execution of the algorithm. In addition, we provide a *shadow-banning* mechanism to remove faulty nodes from the execution, and nullify the information they have broadcast up to the time when they were declared faulty.**

I. Introduction

We consider distributed systems that consist of a set of nodes that can exchange information among themselves. In such systems, it is often necessary for the nodes to calculate a function of certain parameters, with each of these possessed by an individual node [1], [2]. For example, when all nodes calculate the average of these parameters, they are said to reach average consensus. An important class of distributed averaging algorithms rely on each node maintaining a state that is updated iteratively as a linear combination of the states maintained by in-neighbors, i.e., nodes from which a node receives information directly; see, e.g., [1]–[3].

Due to, e.g., hardware failures, nodes may introduce errors in the computations performed during the execution of these algorithms, which in turn may lead to convergence to the incorrect average value, or even result in the algorithms failing to converge [4]. The objective of this paper is to address this issue and design algorithms for distributed averaging that are robust against faults. Specifically, we are interested in detecting the presence of faulty nodes and having them removed from the averaging calculations performed by fault-free nodes. Our starting point to attack the problem of interest is an algorithm within the aforementioned class of linear-iterative algorithms, referred to as the running-sum ratio consensus algorithm [5], [6], which is closely related to other algorithms within said class such as the push-algorithm algorithm and its variants [7], [8].

C. N. Hadjicostis is with the ECE Department at the University of Cyprus, and also with the ECE Department at the University of Illinois. E-mail: hadjicostis.christoforos@ucy.ac.cy.

Alejandro D. Domínguez-García is with the ECE Department at the University of Illinois. E-mail: aledan@illinois.edu.

We first demonstrate that, throughout its execution, the running-sum ratio consensus algorithm exhibits an invariance property that is local to each node in the system and reflects the conservation of certain quantities capturing an aggregate of all the values received by a node from its in-neighbors (i.e., nodes from which it receives information), and all the values that said node broadcasts to its out-neighbors (i.e., nodes that receive its information). This invariance property is related to one we introduced in [9] for a slightly different algorithm. However, unlike the earlier one, the newly-introduced invariance property is amenable for implementing *distributed monitoring* schemes that enable any particular node to check whether or not any of its in-neighbors has become faulty.

In the specific monitoring mechanism we propose, a node that is checking an in-neighbor needs to receive information from the in-neighbors of the node being checked; fortunately, such information only needs to be received infrequently. To implement this, we introduce a *probabilistic transmission scheme* whereby each node randomly decides whether to transmit at high or normal power. In the former case, its transmission is received by all of its out-neighbors as well as its two-hop out-neighbors (i.e., all out-neighbors of its out-neighbors). In the latter case, only its (one-hop) out-neighbors receive the transmission. The use of two-hop information is also leveraged in [10] in the context of detection of faulty/malicious nodes in a distributed system. However, unlike our proposed monitoring scheme, this other scheme requires two-hop information at each iteration.

Finally, we show how the running-sum ratio consensus can be modified so as to introduce a fully distributed scheme, which allows the out-neighbors of a faulty node to detect when it becomes faulty, so that they can isolate it and nullify the errors it has introduced, despite the fact that the in-neighbors of the faulty node may be completely oblivious to their out-neighbor becoming faulty. Also the scheme is such that the faulty node might be oblivious to having been removed from the computation; this is similar to the notion of *shadow-banning* used in online communities and social networks to block a user or its content without the ban being readily apparent to the blocked user (see, e.g., [11]). The proposed shadow-banning scheme does not require any additional transmissions or coordination, as it utilizes infrequent two-hop transmissions that each out-neighbor of the faulty node receives from the in-neighbors of this faulty node (to perform the checking described above). The end result is that the average calculation only includes the parameters (to be averaged) that are held by fault-free nodes.

II. PRELIMINARIES

Consider a distributed system comprising N nodes, denoted by $v_i,\ i=1,2,\ldots,N$, that can potentially exchange information among themselves. This exchange of information can be captured by a strongly connected directed graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$,[1] with $\mathcal{V}=\{v_1,v_2,\ldots,v_N\}$ and $\mathcal{E}\subseteq\mathcal{V}\times\mathcal{V}-\{(v_j,v_j)\mid v_j\in\mathcal{V}\}$ such that $(v_j,v_i)\in\mathcal{E}$ if node v_j may receive information from node v_i. Note that the exchange of information between pairs of nodes may be *asymmetric*, i.e., v_j may receive information from node v_i but not vice-versa; in such case, we have that $(v_j,v_i)\in\mathcal{E}$ but $(v_i,v_j)\notin\mathcal{E}$.

Nodes that can potentially send information to node v_j directly are said to be its in-neighbors and belong to the set $\mathcal{N}_j^-:=\{v_i\in\mathcal{V}\mid (v_j,v_i)\in\mathcal{E}\}$, which is referred to as the in-neighborhood of node v_j. Similarly, nodes that can receive information from node v_j directly are said to be its out-neighbors and belong to $\mathcal{N}_j^+:=\{v_l\in\mathcal{V}\mid (v_l,v_j)\in\mathcal{E}\}$, which is referred to as the out-neighborhood of node v_j. The cardinality of $\mathcal{N}_j^-$, which we denote by D_j^-, is referred to as the *in-degree* of v_j, whereas the cardinality of $\mathcal{N}_j^+$, which we denote by D_j^+, is referred to as the *out-degree* of v_j.

We define the two-hop in-neighborhood of node v_j as $\mathcal{M}_j^-=\bigcup_{v_i\in\mathcal{N}_j^-}\mathcal{N}_i^-$, i.e., the elements of this set, which we refer to as the two-hop in-neighbors of node v_j, are the in-neighbors of the in-neighbors of node v_j. Similarly, we define the two-hop out-neighborhood of node v_j as $\mathcal{M}_j^+=\bigcup_{v_l\in\mathcal{N}_j^+}\mathcal{N}_l^+$, i.e., the elements of this set, which we refer to as the two-hop out-neighbors of node v_j, are the out-neighbors of the out-neighbors of node v_j.

A. Distributed Averaging via Running-Sum Ratio Consensus

Consider an N-node distributed system whose communication topology is described by a strongly connected directed graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$. Assume that each node v_j possesses a value V_j and the objective for all the nodes is to compute

$$\overline{V}:=\frac{\sum_{l=1}^{N}V_l}{N}\ . \tag{1}$$

Next, we describe an distributed iterative algorithm, referred to as the running-sum ratio consensus algorithm [2], that allows each node $v_j\in\mathcal{V}$ to compute $\overline{V}$. In the remainder, we assume the following holds:

Assumption 1: Each node v_j knows how many out-neighbors it has, i.e., it knows its out-degree D_j^+.

At discrete time instants indexed by $k=0,1,2,\ldots$, each node v_j broadcasts some information intended to its out-neighbors. However, not all the out-neighbors of node v_j might receive this information at each time k (e.g., due to packet drops in the communication links). This in turn implies that at each time instant k, node v_j might only receive information from a subset of its in-neighbors; we denote such set by $\mathcal{N}_j^-[k]\subseteq\mathcal{N}_j^-$. Each node v_j maintains four state variables, $y_j[k]$, $z_j[k]$, $\sigma_j[k]$, and $\eta_j[k]$, and two additional variables $\sigma_{ji}[k]$ and $\eta_{ji}[k]$ for each $v_i\in\mathcal{N}_j^-$, which it uses to obtain $\overline{V}$ as follows. Let $x_j[k]=\big[y_j[k],z_j[k]\big]^\top$, $\phi_j[k]=\big[\sigma_j[k],\eta_j[k]\big]^\top$, and $\psi_{ji}[k]=\big[\sigma_{ji}[k],\eta_{ji}[k]\big]^\top$; then, each node v_j performs the following operations:

R1. Initially, it sets $x_j[0]=[V_j,1]^\top$, $\phi_j[0]=[0,0]^\top$, and $\psi_{ji}[0]=[0,0]^\top,\ \forall v_i\in\mathcal{N}_j^-$.

R2. At each $k=0,1,2,\ldots$, it recursively updates the value of $\phi_j[k]$ as follows

$$\phi_j[k+1]=\phi_j[k]+\frac{1}{1+D_j^+}x_j[k]. \tag{2}$$

R3. At each $k=0,1,2,\ldots$, it broadcasts $\phi_j[k+1]$ to all $v_l\in\mathcal{N}_j^+$.

R4. At each $k=0,1,2,\ldots$, it receives the value $\phi_i[k+1]$ from each $v_i\in\mathcal{N}_j^-[k]$.

R5. It sets $\psi_{ji}[k+1]=\phi_i[k+1]$ if $v_i\in\mathcal{N}_j^-[k]$, and it sets $\psi_{ji}[k+1]=\psi_{ji}[k]$ if $v_i\notin\mathcal{N}_j^-[k]$.

R6. It updates $x_j[k]$ at each $k=0,1,2,\ldots$, as follows:

$$x_j[k+1]=\frac{1}{1+D_j^+}x_j[k]+\sum_{v_i\in\mathcal{N}_j^-}\big(\psi_{ji}[k+1]-\psi_{ji}[k]\big). \tag{3}$$

R7. At each $k=0,1,2,\ldots$, it also calculates

$$r_j[k]:=\frac{y_j[k]}{z_j[k]}. \tag{4}$$

Then, as $\mathcal{G}$ is strongly connected, it can be shown that under some mild connectivity assumptions, e.g., that each node receives values from each of its in-neighbors infinitely often, the $r_j[k]$'s asymptotically converge to the average $\overline{V}$ in (1); see, e.g., [2]. More specifically,

$$\lim_{k\to\infty}r_j[k]=\frac{\sum_l y_l[0]}{\sum_l z_l[0]}=\overline{V}\ ,\quad \forall v_j\in\mathcal{V}\ . \tag{5}$$

Pseudocode for the scheme above is provided in Algorithm 1.

B. Problem Formulation

During the execution of Algorithm 1, or some variation thereof, it is possible that, due to hardware faults or other causes, one or more nodes misbehave by introducing errors in their calculations. Specifically, at one or more time instants, node v_j might alter (2) by adding some error $n_j[k]$, i.e.,

$$\phi_j[k+1]=\phi_j[k]+\frac{1}{1+D_j^+}x_j[k]+n_j[k], \tag{6}$$

and/or it might alter (3) by adding some error $w_j[k]$, i.e.,

$$x_j[k+1]=\frac{1}{1+D_j^+}x_j[k]+\sum_{v_i\in\mathcal{N}_j^-}\big(\psi_{ji}[k+1]-\psi_{ji}[k]\big)+w_j[k]. \tag{7}$$

[1] We say that a directed graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ is 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 $v_i=:v_{l_0},v_{l_1},\ldots,v_{l_t}:=v_j$ such that $(v_{l_{\tau+1}},v_{l_\tau})\in\mathcal{E}$ for $\tau=0,1,\ldots,t-1$. Strong connectedness of the communication topology graph $\mathcal{G}$ is a necessary and sufficient condition for the convergence of the class of algorithms we consider.

Algorithm 1: Running-Sum Ratio Consensus

Input: $x_j[0] = [V_j, 1]^\top$, $\phi_j[0] = [0, 0]^\top$,
$\psi_{ji}[0] = [0, 0]^\top$, $\forall v_i \in \mathcal{N}_j^-$

for $k \geq 0$:

- **Compute:**
  - $\phi_j[k+1] = \phi_j[k] + 1/(1+D_j^+)x_j[k]$
- **Broadcast:** $\phi_j[k+1]$ to all $v_l \in \mathcal{N}_j^+$
- **Receive:** $\phi_i[k+1]$ from each $v_i \in \mathcal{N}_j^-[k]$
- **Set:**
  - $\psi_{ji}[k+1] = \begin{cases} \phi_i[k+1], & \text{if } v_i \in \mathcal{N}_j^-[k] \\ \psi_{ji}[k], & \text{if } v_i \notin \mathcal{N}_j^-[k] \end{cases}$
- **Compute:**
  - $x_j[k+1] =$
    $1/(1+D_j^+)x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \big(\psi_{ji}[k+1] - \psi_{ji}[k]\big)$
- **Output:** $r_j[k+1] = y_j[k+1]/z_j[k+1]$

Note that the above additive error model is quite general as it can capture any arbitrary change of $\phi_j[k]$ and/or $x_j[k]$. Based on (6) and (7) we can now introduce the notions of fault-free and faulty nodes as follows:

Definition 1: We say a node v_j is fault-free up to iteration k if $w_j[t] = n_j[t] = 0$ for all $t = 0, 1, \ldots, k-1$; otherwise we say it is faulty.

Let $\mathcal{F}[k]$ denote the set of faulty nodes at iteration k; it then follows from Definition 1 that $\mathcal{F}[k] \subseteq \mathcal{F}[k+1]$ for all $k \geq 0$. Assume that there are at most $F < N$ nodes that eventually become faulty with the rest of the $N - F$ nodes remaining fault-free; thus, there exists a finite K such that $\mathcal{F}[k] = \mathcal{F} \subset \mathcal{V}$ for all $k \geq K$, with $|\mathcal{F}| = F$. The goal is then for the fault-free nodes to compute the average of their V_j's, i.e., to compute

$$\overline{V}' := \frac{\sum_{v_l \in \mathcal{V} \setminus \mathcal{F}} V_l}{N - F} . \tag{8}$$

III. Averaging in the Presence of Faulty Nodes

Consider an N-node distributed system whose communication topology is described by a strongly connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We assume each node v_j has a unique identifier that is attached to every message it broadcasts. In addition, each node v_j knows its out-degree D_j^+ (i.e., Assumption 1 holds), and has some knowledge of the graph local topology; specifically:

Assumption 2: Each node $v_j \in \mathcal{V}$ knows the unique identifiers of the nodes in its in-neighborhood, $\mathcal{N}_j^-$, and the unique identifiers of the nodes in the in-neighborhood of each of its in-neighbors, i.e., its knows $\mathcal{N}_i^-$ for each $v_i \in \mathcal{N}_j^-$, which means that it knows the unique identifiers of the nodes in its two-hop in-neighborhood, $\mathcal{M}_j^- = \bigcup_{v_i \in \mathcal{N}_j^-} \mathcal{N}_i^-$. Also, node v_j knows $D_i^+ = |\mathcal{N}_i^+|$ for each $v_i \in \mathcal{N}_j^-$.

For each node $v_j \in \mathcal{V}$, we can define a collection of indicator variables, $t_{ji}[k]$, describing nodes in the system that node v_j believes are faulty at instant k (and therefore it cannot trust the information that receives from them). Specifically, let $t_{ji}[k] = 1$ if node v_j believes that node $v_i \in \mathcal{N}_j^-$ is fault-free at instant k, and therefore it trusts the information it receives from it, and $t_{ji}[k] = 0$ otherwise. We also define the set of nodes that node v_j believes are fault-free among its in-neighbors at instant k, which is denoted by $\mathcal{T}_j[k]$, and defined as follows:

$$\mathcal{T}_j[k] = \big\{ v_i \in \mathcal{N}_j^- \mid t_{ji}[k] = 1 \big\}. \tag{9}$$

Initially we assume that all nodes are fault-free, therefore $t_{ji}[0] = 1$ for all $v_i \in \mathcal{N}_j^-$ and $\mathcal{T}_j[0] = \mathcal{N}_j^-$.

In order to solve the problem of interest, we consider a variation of Algorithm 1, where each node v_j receives information from all of its in-neighbors at every iteration k (i.e., $\mathcal{N}_j^-[k] = \mathcal{N}_j^-$, $\forall k$), and it occasionally receives information from its two-hop in-neighbors. Each node v_j then uses this infrequent two-hop information in the following two ways:

I1. It uses it to check whether an in-neighbor $v_i \in \mathcal{N}_j^-$ whose status was previously believed to be fault-free remains so, or whether it has become faulty.

I2. If an in-neighbor, $v_{i'}$, is declared to be faulty at some time instant, then node v_j will nullify the information received from $v_{i'}$ up to then, and will effectively remove $v_{i'}$ from the algorithm execution by (i) ignoring any future information received from this node, and (ii) incorporating in its updates the two-hop information occasionally received from the in-neighbors of $v_{i'}$.

In the next two subsections we will discuss in detail how the tasks outlined above are implemented and integrated into the execution of Algorithm 1. Note, however, that in order for these tasks to be executed correctly, we need to impose some restrictions on the set of faulty nodes, $\mathcal{F}$. In particular, we assume the following (sufficient) condition holds:

Assumption 3: $\mathcal{N}_j^- \cap \mathcal{F} = \emptyset$ and $\mathcal{N}_j^+ \cap \mathcal{F} = \emptyset$ for any $v_j \in \mathcal{F}$, which is to say that no faulty node can have another faulty node in its in- or out-neighborhod.

This assumption is reasonable if the number of nodes in the distributed system is relatively large with respect to the number of possible faulty nodes, i.e., $|\mathcal{V}| >> |\mathcal{F}|$, and the graph associated with the system communication topology is not very dense, i.e., there exists some $D << |\mathcal{V}|$ such that $D_j^+ < D$ and $D_j^- < D$, for all $v_j \in \mathcal{V}$.

A. Checking Node Status

Our mechanism for checking node status during execution of Algorithm 1 is based on the following invariance result.

Lemma 1: Consider the linear iteration in (3). Then, for each $v_j \in \mathcal{V}$, it holds that

$$\mu_j[k] = x_j[0], \quad k \geq 0, \tag{10}$$

where

$$\mu_j[k] = x_j[k] + D_j^+ \phi_j[k] - \sum_{v_i \in \mathcal{N}_j^-} \psi_{ji}[k]. \tag{11}$$

Proof: The proof is by induction, following similar developments as those in [9]; the details are omitted due to space limitations. ∎

The result in Lemma 1 enables each node v_j to check whether or not any of its in-neighbors has become faulty. In particular, if $v_i \in \mathcal{N}_j^-$ is fault-free at instant $k \geq 0$, it follows from (10) and (11) that

$$0 = x_i[k] + D_i^+ \phi_i[k] - \sum_{v_\ell \in \mathcal{N}_i^-} \psi_{i\ell}[k] - x_i[0]. \quad (12)$$

Thus, in order for node v_j to check whether or not a particular node $v_i \in \mathcal{N}_j^-$ is fault-free at each instant k, it needs to know all the quantities on the right-hand side of (12). A sufficient condition would be that, at every instant k, each node v_j is receiving information from all the nodes in its in-neighborhood, $\mathcal{N}_j^-$, and all the nodes in its two-hop in-neighborhood, $\mathcal{M}_\ell^-$. Instead, we only allow two-hop transmissions to take place infrequently as follows (we describe things in terms of a generic node v_ℓ). At each $k \geq 0$:

H1. with probability $1 - p_\ell$, independently between time steps and from other nodes, node v_ℓ broadcasts the value $\phi_\ell[k+1]$ to all the nodes in its out-neighborhood, $\mathcal{N}_\ell^+$ (as required by Algorithm 1); and

H2. with probability p_ℓ, independently between time steps and from other nodes, node v_ℓ broadcasts the value $\phi_\ell[k+1]$ to all the nodes in both its out-neighborhood and its two-hop out-neighborhood, i.e., $\mathcal{N}_\ell^+ \cup \mathcal{M}_\ell^+$.

In practice, each node v_ℓ decides probabilistically whether to broadcast at low or high power depending on the self-chosen probability p_ℓ. Also, node v_j receives $\phi_i[k+1]$ at each k directly from node v_i. Furthermore, given that node v_j also knows $\phi_i[k]$, it can recover the value of $x_i[k]$ as follows

$$x_i[k] = (1 + D_i^+)\big(\phi_i[k+1] - \phi_i[k]\big), \quad k \geq 0, \quad (13)$$

where $\phi_i[0] = [0, 0]^\top, v_i \in \mathcal{N}_j^-$ (see line 1 of Algorithm 1). Then, (12) can be rewritten as follows:

$$\begin{aligned} 0 = (1 + D_i^+)\big(\phi_i[k+1] - \phi_i[1]\big) - \phi_i[k] \\ - \sum_{v_\ell \in \mathcal{N}_i^-} \mu_{i\ell}^{(j)}[k], \quad k \geq 0, \end{aligned} \quad (14)$$

where $\mu_{il}^{(j)}[k] = \psi_{il}[k]$ for $v_i \in \mathcal{N}_j^-$ and $v_\ell \in \mathcal{N}_i^-$. Note that at $k = 0$, node v_j receives the value of $\phi_i[1]$; however, it is assumed that node v_j knows that $\phi_i[0] = 0$. Therefore, with probability $\Pi_{v_\ell \in \mathcal{N}_j^-} p_\ell > 0$, node v_j can check whether or not (12) holds, i.e., if node v_i has become faulty.

B. Removal of Faulty Nodes

Removing faulty nodes involves three actions: nullify, neglect, and notify, the details of which are as follows:

Nullify. Assume that node v_j declares in-neighbor v_i as faulty at instant $k_f - 1$, thus, $t_{ji}[k_f - 1] = 1$ and $t_{ji}[t] = 0$ for all $t \geq k_f$. At this point, node v_j needs to nullify all the information it has received from the faulty node. More generally, this needs to be done for all in-neighbors that are declared as faulty by node v_j at iteration k_f, i.e., for all the nodes in the set $\Delta\mathcal{T}_j[k_f] := \mathcal{T}_j[k_f - 1] \backslash \mathcal{T}_j[k_f]$. To accomplish this, node v_j resets the value of $x_j[k_f]$ as follows:

$$x_j[k_f] := x_j[k_f] - \sum_{v_i \in \Delta\mathcal{T}_j[k_f]} \psi_{ji}[k_f]. \quad (15)$$

Neglect. In addition, node v_j will ignore all future information that it receives from $v_i \in \Delta\mathcal{T}_j[k]$; this can be accomplished by setting $\psi_{ji}[k]$, $v_i \in \Delta\mathcal{T}_j[k]$, to zero for all $k > k_f$. Equivalently, node v_j sets

$$\psi_{ji}[k+1] = 0, \quad \forall v_i \notin \mathcal{T}_j[k]. \quad (16)$$

Notify. Finally, node v_j notifies all the nodes that are checking its status, i.e., all the nodes in its out-neighborhood, $\mathcal{N}_j^+$, about the fact that it has declared faulty certain nodes in its in-neighborhood, $\mathcal{N}_j^-$. Upon receiving this information, each node $v_m \in \mathcal{N}_j^+$ stops checking the status of node v_j. Note that this will not cause any problem further down the line because, under the condition in Assumption 3, which we adopt for simplicity, node v_j is necessarily fault free.

C. Fault-Tolerant Running-Sum Ratio Consensus Algorithm

In order to enable fault-free nodes to compute the average of their V_j's, we will modify Algorithm 1 using the strategies discussed above for detecting the presence of faulty nodes, removing their effect on the computation, and compensating for their absence. To this end, in addition to maintaining $x_j[k] = \big[y_j[k], z_j[k]\big]^\top$, $\phi_j[k] = \big[\sigma_j[k], \eta_j[k]\big]^\top$, and $\psi_{ji}[k] = \big[\sigma_{ji}[k], \eta_{ji}[k]\big]^\top$, $\forall v_i \in \mathcal{N}_j^-$, each node v_j maintains a two-dimensional vector $\mu_{i\ell}^{(j)}[k]$ for each $v_i \in \mathcal{N}_j^-$ and each $v_\ell \in \mathcal{N}_i^-$. More specifically, node v_j uses $\mu_{i\ell}^{(j)}[k]$ to track the value of $\psi_{i\ell}[k]$ used by its in-neighbor v_i in its updates; this is important when node v_j checks the correct operation of node v_i. All of the above variables are updated through the following steps:

T1. Initially, node v_j sets

$$\begin{aligned} x_j[0] = [V_j, 1]^\top, \phi_j[0] = [0, 0]^\top, \\ \psi_{ji}[0] = [0, 0]^\top, \ \forall v_i \in \mathcal{N}_j^-, \\ \mu_{i\ell}^{(j)}[0] = [0, 0]^\top, \ \forall v_i \in \mathcal{N}_j^-, \ \forall v_\ell \in \mathcal{N}_i^-, \\ \mathcal{T}_j[0] = \mathcal{N}_j^-, \quad \Delta\mathcal{T}_j[0] = \emptyset, \\ \mathcal{T}_i^{(j)}[k] = \mathcal{N}_i^-, \ \forall v_i \in \mathcal{N}_j^-. \end{aligned} \quad (17)$$

T2. At each $k = 0, 1, 2, \ldots$, node v_j recursively updates the value of $\phi_j[k]$ as follows

$$\phi_j[k+1] = \phi_j[k] + \frac{1}{1 + D_j^+} x_j[k]. \quad (18)$$

T3. At each $k = 0, 1, 2, \ldots$, it broadcasts $\phi_j[k+1]$ and $\Delta\mathcal{T}_j[k]$ to all $v_m \in \mathcal{N}_j^+ \cup \mathcal{M}_j^+$ with probability p_j, or to all $v_l \in \mathcal{N}_j^+$ with probability $1 - p_j$ (independently between time steps and from other nodes).

T4. At each $k = 0, 1, 2, \ldots$, it receives $\phi_i[k+1]$ and $\Delta\mathcal{T}_i[k]$ from each $v_i \in \mathcal{N}_j^-$, and, with probability p_ℓ, it also receives $\phi_\ell[k+1]$ and $\Delta\mathcal{T}_\ell[k]$ via a broadcast message from node $v_\ell \in \mathcal{M}_\ell^- = \cup_{v_i \in \mathcal{N}_j^-} \mathcal{N}_i^-$.

T5. Node v_j updates local copies of $\mathcal{T}_i[k],\ v_i \in \mathcal{N}_j^-$, that it maintains as follows

$$\mathcal{T}_i^{(j)}[k+1] = \mathcal{T}_i^{(j)}[k] \setminus \Delta\mathcal{T}_i[k], \quad v_i \in \mathcal{N}_j^-. \tag{19}$$

T6. Node v_j sets

$$\psi_{ji}[k+1] = \begin{cases} \phi_i[k+1]\ , & \text{if } v_i \in \mathcal{T}_j[k], \\ 0\ , & \text{if } v_i \notin \mathcal{T}_j[k]. \end{cases} \tag{20}$$

In addition, node v_j sets $\xi_{j\ell}[k] = 1$ if it receives information from $v_\ell \in \mathcal{M}_j^-$ at instant k, and $\xi_{j\ell}[k] = 0$, otherwise, and updates

$$\mu_{i\ell}^{(j)}[k+1] = \begin{cases} \phi_\ell[k+1]\ , & \text{if } \xi_{j\ell}[k] = 1 \\ & \text{and } v_\ell \in \mathcal{T}_i^{(j)}[k], \\ \mu_{i\ell}^{(j)}[k]\ , & \text{if } \xi_{j\ell}[k] = 0 \\ & \text{and } v_\ell \in \mathcal{T}_i^{(j)}[k], \\ 0\ , & \text{otherwise.} \end{cases} \tag{21}$$

T7. For each $v_i \in \mathcal{T}_j[k]$, if $\xi_{j\ell}[k-1] = 1$ for *all* $v_\ell \in \mathcal{N}_i^-$ and $\mathcal{T}_i^{(j)}[k] = \mathcal{N}_i^-$, it computes

$$\gamma_i^{(j)}[k] = (1 + D_i^+)\big(\phi_i[k+1] - \phi_i[1]\big) - \phi_i[k] - \sum_{v_\ell \in \mathcal{N}_i^-} \mu_{i\ell}^{(j)}[k], \quad k \geq 1, \tag{22}$$

where $\phi_i[0] = [0, 0]^\top$, sets

$$t_{ji}[k+1] = \begin{cases} 1\ , & \text{if } \gamma_i^{(j)}[k] = 0, \\ 0\ , & \text{if } \gamma_i^{(j)}[k] \neq 0, \end{cases} \tag{23}$$

and updates

$$\begin{aligned} \mathcal{T}_j[k+1] &= \big\{v_i \in \mathcal{T}_j[k] \mid t_{ji}[k+1] = 1\big\} \\ \Delta\mathcal{T}_j[k+1] &= \mathcal{T}_j[k] \backslash \mathcal{T}_j[k+1], \end{aligned} \tag{24}$$

accordingly. Note that the condition $\mathcal{T}_i^{(j)}[k] = \mathcal{N}_j^-$ above means that node v_j stops checking node v_i if this node declares any of its in-neighbors faulty; we elaborate why this is necessary after we complete the description of the algorithm.

T8. It updates $x_j[k]$ at each $k = 0, 1, 2, \ldots$, as follows:

$$\begin{aligned} x_j[k+1] = {} & \frac{1}{1 + D_j^+} x_j[k] \\ & + \sum_{v_i \in \mathcal{N}_j^-} \big(\psi_{ji}[k+1] - \psi_{ji}[k]\big) \\ & + \sum_{v_i \in \mathcal{N}_j^- \backslash \mathcal{T}_j[k+1]} \sum_{v_\ell \in \mathcal{N}_i^-} \frac{\mu_{i\ell}^{(j)}[k+1] - \mu_{i\ell}^{(j)}[k]}{D_i^+} \\ & + \sum_{v_i \in \mathcal{N}_j^- \bigcap \Delta\mathcal{T}_j[k+1]} \sum_{v_\ell \in \mathcal{N}_i^-} \frac{\mu_{i\ell}^{(j)}[k]}{D_i^+}\ . \end{aligned} \tag{25}$$

T9. At each $k = 0, 1, 2, \ldots$, it also calculates

$$r_j[k] := \frac{y_j[k]}{z_j[k]}. \tag{26}$$

The peudocode for the iterative procedure in T1 – T9 is provided in Algorithm 2; next, we elucidate some key details in steps T1 – T9.

Algorithm 2: Running-Sum Ratio Consensus with Fault Checking and Shadowing

Input: $x_j[0] = [V_j, 1]^\top$, $\phi_j[0] = [0, 0]^\top$,
$\psi_{ji}[0] = [0, 0]^\top,\ \forall v_i \in \mathcal{N}_j^-$,
$\mu_{i\ell}^{(j)}[0] = [0, 0]^\top,\ \forall v_i \in \mathcal{N}_j^-,\ \forall v_\ell \in \mathcal{N}_i^-$
$\mathcal{T}_j[0] = \mathcal{N}_j^-$, $\Delta\mathcal{T}_j[0] = \emptyset$,
$\mathcal{T}_i^{(j)}[k] = \mathcal{N}_i^-,\ \forall v_i \in \mathcal{N}_j^-$

for $k \geq 0$:

- **Compute:**
  - $\phi_j[k+1] = \phi_j[k] + 1/(1 + D_j^+) x_j[k]$
- **Broadcast:** $\phi_j[k+1]$ and $\Delta\mathcal{T}_j[k]$ to all $v_m \in \mathcal{N}_j^+ \cup \mathcal{M}_j^+$ with probability $p_j < 1$, or to all $v_l \in \mathcal{N}_j^+$ with probability $1 - p_j$
- **Receive:** $\phi_i[k+1]$ and $\Delta\mathcal{T}_i[k]$ from each $v_i \in \mathcal{N}_j^-$ with probability one, and $\phi_\ell[k+1]$ and $\Delta\mathcal{T}_\ell[k]$ from each $v_\ell \in \mathcal{M}_j^-$ with probability $p_\ell < 1$
- **Set:**
  - $\xi_{j\ell}[k] = \begin{cases} 1\ , & v_j \text{ received } \phi_\ell[k+1], \Delta\mathcal{T}_\ell[k] \\ & \text{from } v_\ell \in \mathcal{M}_j^- \text{ at instant } k, \\ 0\ , & \text{otherwise.} \end{cases}$
  - $\mathcal{T}_i^{(j)}[k+1] = \mathcal{T}_i^{(j)}[k] \setminus \Delta\mathcal{T}_i[k]$ for each $v_i \in \mathcal{N}_j^-$
  - $\psi_{ji}[k+1] = \begin{cases} \phi_i[k+1]\ , & \text{if } v_i \in \mathcal{T}_j[k], \\ 0\ , & \text{if } v_i \notin \mathcal{T}_j[k]. \end{cases}$
  - $\mu_{i\ell}^{(j)}[k+1] = \begin{cases} \phi_\ell[k+1]\ , & \text{if } \xi_{j\ell}[k] = 1 \\ & \text{and } v_\ell \in \mathcal{T}_i^{(j)}[k], \\ \mu_{i\ell}^{(j)}[k]\ , & \text{if } \xi_{j\ell}[k] = 0 \\ & \text{and } v_\ell \in \mathcal{T}_i^{(j)}[k], \\ 0\ , & \text{otherwise.} \end{cases}$
- **Compute (only if** $k \geq 1$**):**
  - $\gamma_i^{(j)}[k] = (1 + D_i^+)\big(\phi_i[k+1] - \phi_i[1]\big) - \phi_i[k] - \sum_{v_\ell \in \mathcal{N}_i^-} \mu_{i\ell}^{(j)}[k]$
    for each $v_i \in \mathcal{T}_j[k]$, if $\xi_{j\ell}[k-1] = 1, \forall v_\ell \in \mathcal{N}_i^-$, and $\mathcal{T}_i^{(j)}[k] = \mathcal{N}_i^-$
- **Set:**
  - $t_{ji}[k+1] = \begin{cases} 1\ , & \text{if } \gamma_i^{(j)}[k] = 0, \\ 0\ , & \text{if } \gamma_i^{(j)}[k] \neq 0. \end{cases}$
  - $\mathcal{T}_j[k+1] = \big\{v_i \in \mathcal{T}_j[k] \mid t_{ji}[k+1] = 1\big\}$
  - $\Delta\mathcal{T}_j[k+1] = \mathcal{T}_j[k] \setminus \mathcal{T}_j[k+1]$
  - $\mu_{il}^{(j)}[k] = 0$ for all $v_i \in \Delta\mathcal{T}_j[k+1]$ and all $v_\ell \in \mathcal{N}_i^-$
- **Compute:**
  - $x_j[k+1] = \frac{1}{1 + D_j^+} x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \big(\psi_{ji}[k+1] - \psi_{ji}[k]\big) + \sum_{v_i \in \mathcal{N}_j^- \backslash \mathcal{T}_j[k+1]} \sum_{v_\ell \in \mathcal{N}_i^-} \frac{\mu_{i\ell}^{(j)}[k+1] - \mu_{i\ell}^{(j)}[k]}{D_i^+}$
- **Output:** $r_j[k+1] = y_j[k+1]/z_j[k+1]$

Consider the update performed by node v_j described in Step T8. According to (20) if node v_j declares an in-neighbor v_i as faulty at some instant k_i, it will set $\psi_{ji}[k+1] = 0$ for all $k \geq k_i$, thus, by inspection of (25), we see that at $k = k_i$, node v_j subtracts the value $\psi_{ji}[k_i]$ when updating $x_j[k_i]$ to $x_j[k_i + 1]$; this is how the reseting in (15) (that node v_j needs to perform after declaring v_i faulty) is implemented in practice. Also since $\psi_{ji}[k+1] = \psi_{ji}[k] = 0$ for all $k > k_i$, inspection of (25) reveals that node v_j does not incorporate any information it receives from node v_i after it is declared faulty. However, the in-neighbors of node v_i are in general not aware that this node has been declared faulty and effectively removed from the computation by its out-neighbors; thus, they continue broadcasting as before to all their out-neighbors (including v_i), and without further action this would result in the algorithm failing to converge, or converge to an incorrect value. This issue is resolved by the third term in the right-hand side of (25). Specifically, once node v_i is declared faulty, node v_j starts incorporating information received infrequently from each in-neighbor v_ℓ of node v_i (as if it was their out-neighbor); thus, it adds

$$\sum_{v_\ell \in \mathcal{N}_i^-} \frac{\mu_{i\ell}^{(j)}[k+1] - \mu_{i\ell}^{(j)}[k]}{D_i^+}, \tag{27}$$

which in effect is splitting the values of $\phi_\ell[k],\ v_\ell \in \mathcal{N}_i^-$, that were broadcast to node v_i by its in-neighbor v_ℓ equally among all of its out-neighbors (one of which is node v_j), hence, the factor $1/D_i^+$ in (27). This essentially amounts to bypassing node v_i. Note that throughout this process, unless node v_i is also an out-neighbor of a node that is checking it, node v_i is oblivious to the fact that it has been removed from the computation; as already mentioned, this is similar to the notion of shadow banning used in online communities to block a user without the ban being apparent to said user. Finally, it is important to note that the summation in the last line of (25) is only active at the time instant at which node v_i is declared faulty by node v_j. At that time intant, everything that the in-neighbors of node v_i have sent to that node thus far needs to be redirected to the out-neighbors of node v_i (which includes node v_j). In practice, said summation can be omitted if one sets $\mu_{il}^{(j)}[k] = 0$ for all $v_i \in \Delta\mathcal{T}_j[k+1]$ and all $v_\ell \in \mathcal{N}_i^-$; this is how we implement this in Algorithm 2.

As mentioned in Step T7, node v_j will stop checking any node $v_i \in \mathcal{N}_j^-$ if this node declares any of its own in-neighbors as faulty; the reason for this is as follows. If node v_i declares one of its in-neighbors as faulty, say $v_{\ell'}$, at time $k_{\ell'}$, it will set $\psi_{i\ell'}[k+1] = 0$ for all $k \geq k_{\ell'}$; thus, as argued above, node v_i nullifies any information received from node $v_{\ell'}$ up to instant $k_{\ell'}$ and stops incorporating any future information received from said node. This in turn means that the local invariant in Lemma 1 associated to node v_i must exclude node $v_{\ell'}$ for all $k > k_{\ell'}$, but in light of (20), it would need to include the infrequent two-hop information that node v_i receives from the in-neighbors of node $v_{\ell'}$. However, this information is not available to any of the out-neighbors of node v_i; thus, no node $v_j \in \mathcal{N}_i^+$ can correctly calculate $\gamma_i^{(j)}[k],\ k > k_{\ell'}$, which in turn means it can no longer assess its status. However, under Assumption 3 this is not an issue as node v_i cannot be faulty because one of its in-neighbors, namely node $v_{\ell'}$ has become faulty. In light of Step T5, this information is readily available to node v_j because the set $\mathcal{T}_i^{(j)}[k]$, which is a local copy (maintained by node v_j) of the set of nodes that node v_i considers fault-free, would exclude $v_{\ell'}$ for all $k > k_{\ell'}$.

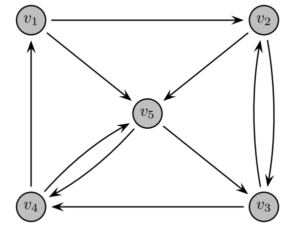

Fig. 1. Directed graph considered in numerical example (from [9]).

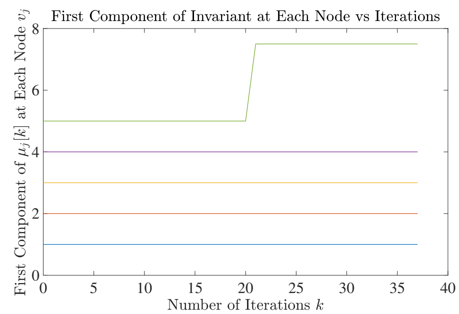

Fig. 2. First component of invariant $\mu_j[k]$ plotted against the number of iterations k for each node v_j for the digraph in Fig. 1.

We can now state the main result regarding convergence of Algorithm 2. The result can be established using (10) and the analysis techniques in our earlier work in [5], [9], [12], however, we omit the details due to lack of space.

Theorem 1: Consider a distributed system with N nodes whose communication topology is described by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whereby each node $v_j \in \mathcal{V}$ possesses some value V_j and executes Algorithm 2. Assume that during the execution of the algorithm, there are at most $F < N$ nodes that eventually become faulty with the rest of the $N - F$ nodes remaining fault-free. Then, under Assumptions 1 – 3,

$$\lim_{k \to \infty} r_j[k] = \frac{\sum_{v_l \in \mathcal{V} \setminus \mathcal{F}} V_l}{N - F}, \quad \forall v_j \in \mathcal{V} \setminus \mathcal{F}, \tag{28}$$

with probability one.

IV. Numerical Example

Consider the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Fig. 1 (from [9]), which represents the communication topology of a distributed system. We assume that the initial values of the nodes are $V_i = i$ for $i = 1, 2, ..., 5$ so that $\overline{V} = 3$.

The nodes run the running-sum ratio consensus algorithm, enhanced to handle faulty nodes, as described in Algorithm 2. Each node is assumed to send a two-hop transmission with probability $p_i = 1/3,\ i = 1, 2, 3, 4, 5$. All nodes behave correctly, except node v_5 which, at iteration $k = 20$, changes

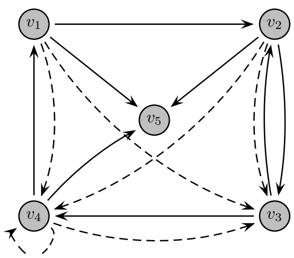

Fig. 3. Effective directed graph after node v_5 is declared faulty: outgoing links from node v_5 become inactive, dashed lines represent links that become active when transmitting node performs a two-hop transmission.

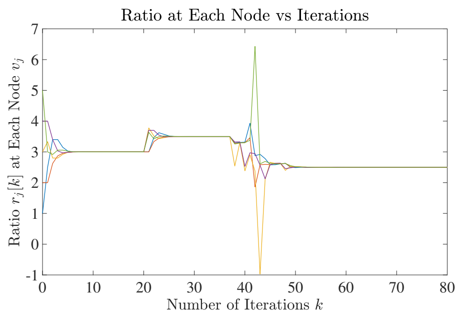

Fig. 4. Ratios $r_j[k]$ plotted against the number of iterations k for each node v_j for the digraph in Fig. 1.

its y-value by adding 2.5 to it (i.e., $y_5[20]$ increases by 2.5). After this fault, the in-neighbors of node v_5 (namely, nodes v_1, v_2 and v_4) happen to simultaneously perform two-hop transmissions at iteration $k = 37$, at which point all out-neighbors of node v_5 (namely, nodes v_4 and v_3) are able to identify node v_5 as faulty. In Fig. 2, we plot the first component of $\mu_j[k]$ in (11) in Lemma 1 against the iteration step k for all nodes v_1, v_2, v_3, v_4, v_5. One can see in Fig. 2 that, after the fault occurs at iteration $k = 20$, the first component of $\mu_5[20]$, $\mu_5[21]$, ..., $\mu_5[37]$ becomes equal to 7.5 (instead of 5) because of the addition of 2.5 at iteration 20. Note that, even though the error only occurred at $k = 20$, the invariant for node v_5 shows a violation until $k = 37$ when the fault is discovered by the out-neighbors of node v_5.

Once node v_5 is identified as faulty at iteration $k = 37$, the shadowing scheme effectively changes the directed graph as in Fig. 3. The links from node v_5 to its out-neighbors (nodes v_3 and v_4) are not present in this graph because these nodes identify node v_5 as faulty and ignore its transmissions. In fact, at iteration $k = 37$, these nodes also subtract the runnings sum they received from node v_5, effectively removing its effects from their calculations. On the other hand, the in-neighbors of node v_5 (nodes v_1, v_2 and v_4) are unaware of the fault and keep sending to node v_5; however, the out-neighbors of node v_5 account for this by using the dashed two-hop edges that only become available when two-hop transmissions take place. Note that it is not necessary for all dashed links to be simultaneously active for this adjustment to take place; for example, when node v_2 happens to send its running sums via a two-hop transmission, both nodes v_3 and v_4 receive these running sums and use a weight of $1/2 (= 1/D_5^+)$ to add them to their value.

In Fig. 4 we see the ratio at each node. Note that in the beginning, node v_5 is not faulty, so all ratios converge to $\overline{V} = 3$ which is the average of $\{1, 2, 3, 4, 5\}$. At $k = 20$ the fault takes place and soon after the nodes converge to 3.5 which is the erroneous average of $\{1, 2, 3, 4, 7.5\}$. At $k = 37$, node v_5 is declared faulty by its out-neighbors; this causes an adjustment because the out-neighbors of node v_5 abruptly remove it from the computation. At subsequent steps, the out-neighbors of node v_5 use the two-hop transmissions whenever they become available (roughly once every three time steps); this allows all nodes to converge to $\overline{V}' = 2.5$ which is the average of $\{1, 2, 3, 4\}$ (with node v_5 excluded).

V. Concluding Remarks

In this paper, we have first established a local invariance property exhibited during its execution by the running-sum ratio consensus algorithm, which is an existing algorithm for averaging in a distributed system whose communication topology is described by a directed graph. We have then shown how such property can be utilized in designing distributed schemes for checking whether or not particular nodes in the system become faulty and start introducing errors in the computations they perform during the execution of said algorithm. Finally, we have introduced a shadow-banning scheme that ensures that nodes that have been declared faulty are isolated and completely removed from the averaging computation in a distributed manner.

References

[1] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1984.
[2] 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.
[3] 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, September 2004.
[4] C. N. Hadjicostis and A. D. Domínguez-García, "Identification of malicious activity in distributed average consensus via non-concurrent checking," *IEEE Control Systems Letters*, vol. 7, pp. 1927–1932, 2023.
[5] A. D. Domínguez-García, C. N. Hadjicostis, and N. H. Vaidya, "Resilient networked control of distributed energy resources," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 6, pp. 1137–1148, July 2012.
[6] 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, June 2016.
[7] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. of Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 482–491.
[8] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proc. of IEEE International Symposium on Information Theory*, 2010, pp. 1753–1757.
[9] 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.
[10] L. Yuan and H. Ishii, "Secure consensus with distributed detection via two-hop communication," *Automatica*, vol. 131, p. 109775, 2021.
[11] Y.-S. Chen and T. Zaman, "Shaping opinions in social networks with shadow banning," *PLOS ONE*, vol. 19, no. 3, p. e0299977, 2024.
[12] C. N. Hadjicostis and A. D. Domínguez-García, "Trustworthy distributed average consensus based on locally assessed trust evaluations," *IEEE Trans. on Automatic Control*, vol. 70, no. 1, p. 371 – 386, 2025.