Manitara, Nicolaos E.; Hadjicostis, Christoforos N.; Charalambous, Themistoklis

Synchronized Stop and Re-Start in Distributed Average Consensus for Control and Coordination Applications

Published in:
2022 58th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2022

DOI:
10.1109/Allerton49937.2022.9929394

Published: 01/01/2022

Document Version
Peer reviewed version

Please cite the original version:
Synchronized Stop and Re-Start in Distributed Average Consensus for Control and Coordination Applications

Nicolaos E. Manitara, Christoforos N. Hadjicostis, and Themistoklis Charalambous

Abstract—In this work we develop and analyze a distributed iterative algorithm that enables the components of a distributed system, each with some initial value, to reach approximate average consensus on their initial values, after executing a finite number of iterations. The proposed algorithm has features that facilitate its use in control/coordination applications where timing constraints might prohibit the implementation of the asymptotic techniques that are used in most distributed average consensus protocols. More specifically, the proposed algorithm allows all the nodes to (i) simultaneously stop the average consensus process once the network has reached consensus, and (ii) restart the process again at any later time, presumably with different initial values. To provide this capability, the proposed algorithm uses a criterion that enables all the nodes to determine, in a distributed manner, when to terminate the operation. Specifically, approximate average consensus has been reached, i.e., all nodes have obtained a value that is within a small distance from the average of their initial values. This criterion enables all the nodes in the system to terminate their operation (together, at the same time-step) after they have reached approximate consensus. Using another distributed protocol, the system can also restart the distributed calculation of a new average value at any later time, as long as any node initiates such a request.

Keywords: Approximate average consensus, distributed stopping, finite-time average consensus, multi-agent systems.

I. INTRODUCTION

The successful operation of distributed systems lies at the core of the latest technological achievements in the so-called era of smart systems. It has long been recognized that distributed (or decentralized) systems have specific characteristics, such as concurrency of components, resource sharing, openness, fault tolerance, transparency and more importantly simplicity of implementation. For the above-mentioned reasons, the study of distributed systems has been on the rise, as also evidenced by the large number of applications, many of which are life and mission critical, ranging from coordinating teams of autonomous vehicles for search and rescue operations, to transmitting patient diagnostic data in hospitals using multi-hop wireless networks. To facilitate the above-mentioned characteristics and applications, distributed algorithms must be designed to be applicable and provide solutions to real life problems, e.g., minimizing the power consumption of the nodes while at the same time increasing the efficiency and reliability of an underlying task.

A distributed system consists of a collection of autonomous components (also referred to as agents and modeled as nodes of a graph), whose actions and reactions are based on the passing of messages that they can receive/transmit at any time instant from/to their immediate neighbors (whose interconnections are modeled as edges of the graph). In distributed systems and networks, it is often necessary for all or some of the nodes to calculate a function of certain parameters that we refer to as initial values. When all nodes calculate the average of these initial values, they are said to reach average consensus. Average consensus (and more generally consensus) has received a lot of attention from the control community due to its usage in various emerging distributed control applications, including wireless smart meters (where all nodes have to determine the average demand or consumption of the network [1]), and multi-agent systems (where agents communicate with neighboring agents in order to coordinate their direction, speed, and position [2]).

Over the last few decades, a variety of algorithms for calculating different functions of these initial values have been proposed by the control [3], [4], communication [5], and computer science [6], [7], communities. One popular approach to average consensus is based on ratio-consensus [1], [8], [9], which simultaneously runs two linear iterations and allows each node to asymptotically obtain the average as the ratio of two state variables it maintains and updates based on the two concurrent iterative computations. Another popular approach is based on a single linear iteration where each node maintains a variable that is initialized at its initial value and updated at each iteration based on a weighted linear combination of its own variable and the variables of its neighbors; it can be shown that, when the weights are chosen to form a primitive doubly stochastic matrix, all node variables converge to the average of the initial values [10].

Most of the above-mentioned popular approaches for average consensus only reach average consensus asymptotically. In this paper, we consider how the nodes can reach approximate average consensus in a finite number of time steps, establishing simultaneous stopping and restarting. Specifically, the nodes need to determine based on locally available information when to terminate their operation and also stop simultaneously (all at the same time
step), and re-start at any later time with new initial values for a new round of average consensus in the distributed system. Simultaneous stopping and re-starting are useful, among others, in applications where nodes are energy constrained, e.g., by enabling them to enter a low-power stand-by mode once they are certain that all nodes have completed their computation or task.

In this paper, we develop an algorithm that allows the nodes to (i) simultaneously stop the average consensus process after the network has reached approximate average consensus, and (ii) re-start the iterative process when a node is awakened for any reason and a new average calculation is required by the network. We establish that all executions of our proposed algorithm enable the nodes to stop simultaneously after a finite number of steps and prove that, when all nodes eventually stop transmitting, the absolute differences of the final values of the nodes from the average of the initial values is smaller than an error bound, whose value depends on a global parameter $\epsilon$ (assumed known to all nodes) and an upper bound $D'$ (assumed known to all nodes) on the diameter $D$ of the graph. One of the key observations in the proposed algorithm is that each node makes a decision as to whether to start the termination process, based on the difference between its calculated value and the values it receives from its neighbors. More specifically, a node obtains the pairwise absolute differences between its own value and the values of each of its neighbors, and decides to start the termination process of the average calculation when all of these differences are smaller than the parameter $\epsilon$.

The remainder of the paper is organized as follows. In Section II, we provide some necessary background on graph theory and we describe existing iterative strategies for reaching asymptotic average consensus in a given distributed system with symmetric and/or asymmetric communication links, also referring to previous works on finite time average consensus. In Section III, we introduce our proposed strategy for distributed stopping and re-starting and discuss the main results. Section IV includes simulation results, and Section V provides conclusions and directions for future work.

II. BACKGROUND

A. Distributed System Model

In distributed systems, we can model the network topology as a generally directed (or asymmetric) graph (digraph) $G=(X, E)$ where $X=\{1,2,\ldots,n\}$ is the set of components in the system, and $E \subseteq X \times X$ is the set of directed edges. In particular, edge $(i, j) \in E$ if node $j$ can send information to node $i$. The nodes that can transmit information to node $i$ are said to be the in-neighbors of node $i$ and are represented by the set $N^+_i=\{j\mid (j,i) \in E\}$; the number of in-neighbors of node $i$ is called the in-degree of node $i$ and is denoted by $D^-_i=|N^+_i|$. Similarly, the set of nodes that receive information from node $i$ are called its out-neighbors and are denoted by $N^-_i=\{l\mid (i,l) \in E\}$; the number of out-neighbors of node $i$ is called the out-degree of node $i$ and is denoted by $D^+_i=|N^-_i|$. By convention, we do not allow self loops, i.e., $(i,i) \notin E$ for all $i \in X$ (though obviously node $i$ can “receive” its own value). The graph is undirected (or symmetric) when $(i,j) \in E$ if and only if $(j,i) \in E$; in an undirected graph, we have $N^+_i=N^-_i$ and $D^+_i=D^-_i$.

We next review some necessary definitions and terminology from graph theory (more details can be found in [11]). A path of length $t$ from node $j$ to node $i$, $j \neq i$, is a sequence of nodes $j=i_0, i_1, \ldots, i_{t-1}, i_t=i$, such that $(i_l,i_{l+1}) \in E$, for all $l=1,2,\ldots,t$.

The minimum distance from node $j$ to node $i$, $j \neq i$, is the length of the shortest path from node $j$ to node $i$; it is denoted by $d_{\text{min}}(i,j)$ and it is taken to be infinite if there is no path from node $j$ to node $i$. By convention, $d_{\text{min}}(i,i)=0$ for all $i \in X$. The graph is strongly connected if there exists a path (of finite length) from each node $j$ to every other node $i$.

Definition 1: (Diameter) The diameter $D$ of graph $G=(X, E)$ is the longest shortest path between any two nodes, i.e.,

$$D = \max_{i,j \in X, i \neq j} d_{\text{min}}(i,j).$$

Our model deals with networks where information is transmitted via a broadcast model, i.e., each node transmits to all of its out-neighbors the same value. We assume that, during the information exchange process, all information is transmitted/received successfully to appropriate recipients in the network. Moreover, the nodes are assumed to have sufficient memory and computational capability in order to store and perform simple mathematical computations (e.g., additions, multiplications, max/min operations, etc.) during each step of the iterative process.

B. Average Consensus via Double Linear Iteration

In average consensus problems the objective is the calculation of the average of the initial values of the nodes in the network. More specifically, each node $i$ has some value $V_i$ and the objective is for the nodes to calculate the average $\frac{1}{n} \sum_{i=1}^{n} V_i$ via a distributed iterative algorithm. Several distributed iterative algorithms for the calculation of the average have been proposed in the literature [12]. One particular approach that we exploit in this paper is the ratio consensus algorithm [1] which allows each node to asymptotically obtain the exact average of the initial values as the ratio of two state variables it maintains and iteratively updates. More specifically, each node $i$ maintains state variables $y_i[k]$ and $z_i[k]$, and updates them, at iteration $k$, as follows:

$$y_i[k+1] = \sum_{j \in N^-_i \cup \{i\}} y_j[k]/(1+D^+_j), \quad k \geq 0, \quad (1)$$

$$z_i[k+1] = \sum_{j \in N^-_i \cup \{i\}} z_j[k]/(1+D^+_j), \quad k \geq 0, \quad (2)$$

where the initial values are set to $y_i[0]=V_i$ and $z_i[0]=1$, for $i \in X$. The protocol is applicable to networks where the exchange of information is described by arbitrary digraphs, assuming that each node $j$ is aware of its out-degree $D^+_j$.

[Note that it makes sense for transmitting node $j$ to simply send the values $y_j[k] := y_j[k]/(1+D^+_j)$, $z_j[k] := z_j[k]/(1+
to all of its out-neighbors, and for receiving node \( i \) to simply add up the weighted values it receives from all of its in-neighbors.}

At each time step \( k \), each node \( i \) calculates the ratio \( r_i[k] \equiv y_i[k]/z_i[k] \); under the assumption that the digraph describing the exchange of information is strongly connected, it can be shown that \( r_i[k] \) asymptotically converges to the average of the initial values. Specifically, with the chosen initial conditions, we have that

\[
\lim_{k \to \infty} r_i[k] = \frac{\sum_j y_i[j]}{\sum_j z_i[j]} = \frac{\sum_j V_{ij}}{n}, \quad \forall i \in X.
\]  

To see that the above holds, one can write the iterations \( y_i[k] \) as the product of the \( k \)-th entry of the left eigenvector \( V \equiv \{v_i \}_{i=1}^n \) and the \( k \)-th entry of the right eigenvector \( z \equiv \{z_i \}_{i=1}^n \); where \( y_i[k] \) and \( z_i[k] \) are an \( n \)-dimensional column vector containing the values \( y_i[k] \) and \( z_i[k] \) for each node, and \( P \) is a primitive \( n \times n \) column stochastic matrix with weights \( P(i, j) = \frac{1}{1 + \delta_{i,j}} \) if \( i \neq j \), \( (\delta_{i,j} \equiv 0 \text{ otherwise}) \). Thus, we have the following eigenvectors of \( P \) corresponding to the eigenvalue 1:

\[
V P = V \lambda_1, \quad z P = z \lambda_1 = \sum_j z_k[j] \lambda_{1,j} = v_1 \lambda_1, \quad \forall i \in X
\]

where \( z \) and \( V \) are, respectively, the right and left eigenvectors of \( P \) that correspond\(^1\) to its eigenvalue at 1. Therefore, the ratio \( (3) \) will satisfy

\[
\lim_{k \to \infty} r_i[k] = \frac{\sum_j y_i[j]}{\sum_j z_i[j]} = \frac{\sum_j V_{ij}}{n}, \quad \forall i \in X,
\]

where \( v_i \) is the \( i \)-th entry of the right eigenvector \( v \). Note that \( v_i \) need not be known to node \( i \).

The above discussion should make it clear that ratio consensus works with any primitive column stochastic matrix \( P \) (whose 0/1 structure—excluding the diagonal elements—reflects the given communication topology). In fact, ratio consensus iterations can also take the time-varying form

\[
y[k + 1] = P[k] y[k], \quad k \geq 0, \tag{4}
\]

\[
z[k + 1] = P[k] z[k], \quad k \geq 0, \tag{5}
\]

where \( P[k] \) are column stochastic \( n \times n \) matrices that vary at each time step. Subject to some joint connectivity conditions on the graphs that correspond to the zero/one structure of the matrices \( P[k] \), the convergence in \( (3) \) still holds though the proof is more involved [8], [9], [13], [14].

### C. Related Work

Finite time average consensus has recently attracted more attention due to its applicability in practical settings. Finite time average consensus is typically handled by pre-determining the number of steps that the nodes in the network need to perform, so that by the end of these iterations the value of each node is guaranteed to be close to the exact average of the system. There have been works where, given a time-invariant graph, one can obtain a finite sequence of time-varying weights whose use in a finite iteration allows the nodes to obtain the exact average (see [15] for the undirected graph case and [16] for the directed graph case). Such approaches, however, require full knowledge of the network topology so that appropriate time-varying weights can be calculated. Below, we describe approaches that do not require such detailed prior knowledge of the network and still allow the nodes to reach either exact or approximate consensus to the average.

For finite time approximate average consensus there have been two main approaches: (i) synchronous, in which nodes assume knowledge of the diameter \( D \) of the network and terminate the process all together at a time step that is multiple of the diameter [17], [18], or (ii) event-triggered, in which nodes terminate the process provided their value is close enough to the desired one [19], [20]. In this work we use a combination of the two above-mentioned approaches, which we now describe in more detail.

#### 1) Utilizing Min- and Max-Consensus:

One of the first influential works on distributed stopping was by the authors of [21] who proposed an algorithm which runs three iterations in parallel in order to identify, in a distributed manner, the time step at which approximate average consensus is reached. The method is applicable to strongly connected digraphs as long as a set of weights \( p_{ij} \) that form a primitive doubly stochastic matrix \( P \) is available. Specifically, each node runs the average consensus algorithm as in

\[
x_i[k + 1] = p_{ii}[k] x_i[k] + \sum_{j \in X} p_{ij}[k] x_j[k], \tag{6}
\]

with fixed weights \( p_{ij}[k] = p_{ij} \), where \( P = [p_{ij}] \) is primitive doubly stochastic. It is well-known that the iteration above converges asymptotically to the average of the initial values \( x_i[0] = V_i \) for all \( i \in X \).

At the same time, each node also runs (in parallel) a max-consensus and a min-consensus iteration:

\[
y_i[k + 1] = \max_{j \in X} \{y_j[k]\} \quad \text{(max-consensus)}, \tag{max-consensus}
\]

\[
z_i[k + 1] = \min_{j \in X} \{z_j[k]\} \quad \text{(min-consensus)}.
\]

The max-consensus and the min-consensus iterations are re-initialized every \( D \) steps, where \( D \) is the diameter of the graph (which is assumed known). When the max-consensus and min-consensus iterations are re-initialized, they use the current values of the iteration in \( (6) \); more specifically, at iteration \( \ell D \) (for some nonnegative integer \( \ell \)) the max-consensus and min-consensus iterations are initialized by each node \( i \in X \) with values \( y_i[\ell D] = z_i[\ell D] = x_i[\ell D] \). At iteration \( (\ell + 1)D \) (before re-initializing the values), each node obtains

\[
y_i[(\ell + 1)D] = \max_i \{y_i[\ell D]\}, \quad \forall i \in X,
\]

\[
z_i[(\ell + 1)D] = \min_i \{z_i[\ell D]\}, \quad \forall i \in X.
\]

The nodes stop iterating when the absolute difference between these two values (at the end of the \( D \) step interval) satisfies the criterion below

\[
|y_i[(\ell + 1)D] - z_i[(\ell + 1)D]| \leq \varepsilon, \tag{7}
\]

which is equivalent to

\[
\max_i |y_i[\ell D]| - \min_i |z_i[\ell D]| \leq \varepsilon, \tag{8}
\]

where parameter \( \varepsilon \) is a given small real number that is
assumed known to all nodes. Note that all nodes will simultaneously stop iterating since they will simultaneously learn that they have reached approximate agreement (which is easily shown in [21] to be close to the average of the initial values).

Effectively, the max- and min-consensus iterations implement an oracle that tells the nodes to stop when their values are within $\epsilon$ of each other (the only difference is that the oracle might be delayed by a few iterations, between $D$ and $2D$ steps). Also note that, in order to implement the oracle, one needs extra transmissions (to run the max- and min-consensus algorithms) which imply extra power and extra information to be shared in the system.

It should also be stated that the method in [21] is applicable to digraphs, as long as a set of weights that forms a primitive doubly stochastic matrix $P$ is available. Obtaining such a set of weights in digraphs is not necessarily an easy task to perform in a distributed manner [22].

2) Event-Triggered Algorithm: In our previous work [20], we proposed an algorithm for reaching approximate average consensus in undirected (symmetric) graphs. The scheme in [20] makes use of a single linear time-varying iteration of the form in (6), with $x_i[0] = V_i$ and $p_{ij}[k]$ forming a set of time-varying weights. At each time step $k$, the nodes consider a subset of “active” edges $E[k] \subseteq E$ (the way this subset is chosen is discussed next) that forms an undirected, but not necessarily connected, graph $G[k] = \{X, E[k]\}$; then, the weights $p_{ij}[k]$ are chosen such that matrix $P[k] = [p_{ij}[k]]$ is doubly stochastic but not necessarily primitive (that will depend on whether $G[k]$ is connected or not). Specifically, assuming each node knows the total number of nodes $n$ or an upper bound $n'$ on the number of nodes ($n' \geq n$), each node $i$ can choose fixed (nonnegative) weights on its links so that

$$
\sum_l p_{li}[k] = \sum_j p_{lj}[k] = 1, \forall i \in X,
$$

by setting

$$
p_{ij}[k] = \begin{cases} \frac{1}{D_i[k]}, & \text{if } (i, j) \in E[k], \\ 0, & \text{if } (i, j) \notin E[k], \end{cases}
$$

$$
p_{il}[k] = 1 - \frac{\mathcal{D}_l[k]}{n},
$$

where $\mathcal{D}_l[k] = \mathcal{D}_l^+ [k] = \mathcal{D}_l^- [k]$ is the degree of node $i$ at iteration $k$ (i.e., the number of edges that node $i$ has active at iteration $k$). It is easy to check that this choice results in a doubly stochastic (and symmetric) weight matrix $P[k]$. As mentioned earlier, when the underlying graph topology is undirected, many possibilities exist for distributively choosing weights that form doubly stochastic matrices [20], though the task becomes more complex when the graph topology is directed.

In order to determine the set of active edges $E[k]$ at each time step $k$, each node $i$ compares its value against the value of each of its neighbors $j \in \mathcal{N}_i$: if the absolute difference $|x_i[k] - x_j[k]| > \epsilon$, then the edge that connects the two nodes is utilized; otherwise, the edge is ignored. In other words, we take

$$
E[k] = \{(i, j) \in E \mid |x_i[k] - x_j[k]| > \epsilon\},
$$

which ensures that $G[k] = \{X, E[k]\}$ is an undirected graph. Matrix $P[k] = [p_{ij}[k]]$ is then adjusted to be doubly stochastic (using the weights in (9) as applied to graph $G[k]$).

Suppose that for node $i$, we have $|x_i[k] - x_j[k]| \leq \epsilon$ for all $j \in \mathcal{N}_i$; then all edges to/from node $i$ become inactive, which implies that the value $x_i[k]$ will not change. Effectively, node $i$ can stop transmitting its value (at least until one of its edges becomes active again). An edge can become active because a neighbor $j \in \mathcal{N}_i$ may change its value in a way that makes $|x_i[k] - x_j[k]| > \epsilon$. Note that node $i$ will be aware of node $j$’s value (because node $i$ can receive the transmissions of node $j$); in fact, node $j$ will also realize that the link $(j, i)$ needs to be re-activated because it can compare its updated value against the value that node $i$ last transmitted. The iterative process described by (6) using the weights in (9) where $G[k] = \{X, E[k]\}$ with $E[k]$ given by (10), terminates with all the links eventually becoming inactive. Moreover, [20] establishes that at the termination of the iterative process, the nodes reach $(D \times \epsilon)$-approximate average consensus on their initial values, where $D$ is the diameter of graph $G$. This means that the absolute difference between the average of the initial values and the value of each node at the termination of the iterative process is at most $D \times \epsilon$.

3) Pre-determining the number of steps that all nodes in the network will perform: The authors of [23] propose a distributed protocol with which nodes agree on terminating their iterations when they have all computed the exact average value of the network. The method in [23] allows the nodes to distributively compute an upper bound on the diameter of the network in a finite number of steps, which can be useful in several algorithms that require time guarantees.

4) Distributed stopping criterion for ratio consensus: The work in [24] presented analytical results for distributed termination using the ratio consensus protocol. The authors of [24] established the monotonicity of the values of the ratios at the various nodes when executing ratio consensus; then, using max- and min-consensus, they proposed a protocol that nodes can use to identify in a distributed manner when they reach consensus. Note that all nodes are able to terminate their operation simultaneously once they reach agreement.

III. Problem Statement and Main Results

In a typical average consensus problem, the nodes, each with some initial value, follow an iterative strategy that enables them to asymptotically reach, in a distributed manner, agreement to the average of their initial values. Here, we are interested in a protocol that allows the nodes to reach approximate agreement to the average, i.e., agreement to the...
average within an error bound. While finite time average consensus has been investigated, only a limited number of works terminate the iterations; thus, in this paper we are interested in a protocol that allows the nodes to identify, in a distributed manner, when to stop transmitting their values (all of them simultaneously) so that, at the termination of the iterative process, all nodes reach approximate agreement on the average of their initial values. Moreover, nodes need to be able to re-start the iterative process at any later time if any node in the network initiates such a request (for restarting the average calculation process, possibly with new initial values).

All of this section assumes the following setting.

**Setting:** Consider a network described by an undirected graph $G = \{X,E\}$, where $X = \{1,2,...,n\}$ and $E \subseteq X \times X - \{(i,i) \mid i \in X\}$. Each node $i$ has some initial value $x_i[0] = V_i$, and updates this value following a distributed iterative algorithm for $f$ steps.

In terms of the definition below, we are interested in reaching $\varepsilon$-approximate average consensus and also in identifying (in a distributed manner) when such approximate average consensus has been reached.

**Definition 2:** ($\varepsilon$-Approximate Average Consensus) At the end of the iterative process in the above setting, the nodes have reached $\varepsilon$-approximate average consensus if the value $x_i[f]$ of each node $i \in X$ satisfies $|x_i[f] - \overline{x}| \leq \varepsilon$, $\forall i \in X$, where $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} V_i$ is the average of the initial values.

**Definition 3:** ($\varepsilon$-Approximate Local Consensus and $\varepsilon$-Approximate Global Consensus) At the end of the iterative process in the above setting, the nodes reach $\varepsilon$-approximate local consensus if for all nodes $i \in X$, we have $|x_i[f] - x_j[f]| \leq \varepsilon$, $\forall j \in N_i$. The nodes reach $\varepsilon$-approximate global consensus if $|x_i[f] - x_j[f]| \leq \varepsilon$, $\forall i, j \in X$.

If at the end of the iterative process in the above setting the nodes have reached $\varepsilon$-approximate local consensus, then the nodes have also reached $(D \times \varepsilon)$-approximate global consensus, where $D$ is the diameter of graph $G$. The proof of the above statement and the following proposition can be found in [20].

**Proposition 1:** Suppose that, at the end of the iterative process in the above setting, the following are true:

1. The nodes reach $\varepsilon$-approximate global consensus;
2. The average $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} V_i$ satisfies $x_{\min}[f] \leq \overline{x} \leq x_{\max}[f]$ where $x_{\min}[f] = \min_i \{x_i[f]\}$ and $x_{\max}[f] = \max_i \{x_i[f]\}$.

Then, the nodes also reach $\varepsilon$-approximate average consensus.

### A. Proposed Strategies and Main Results

1) **Distributed Simultaneous Termination Algorithm:** In what follows we describe the distributed termination algorithm that enables the nodes to decide when to terminate the average calculation process and stop simultaneously (all together). The main idea of the proposed algorithm is that iterations can be used to check when the ratios of the nodes are within $\varepsilon$-approximate local consensus with their neighboring nodes, at which point they can start the simultaneous termination process. More specifically, each node $i$ in the network runs iterations (1) and (2), also checking whether its ratio $r_i$ is within $\varepsilon$-approximate local consensus with the ratios of its neighboring nodes. At the same time, each node maintains one additional state variable ($flag_{stop}[k]$) that indicates the status of node $i$ in terms of $\varepsilon$-approximate local consensus at time step $k$. When $flag_{stop}[k] = 1$ for all $l \in X$, then $\varepsilon$-approximate local consensus (and, thus, $(D \times \varepsilon)$-approximate global consensus) has been reached for the network. If at iteration $k$, at which we have $flag_{stop}[k] = 1$ for all nodes $i \in X$, we start running a min-consensus algorithm on the $MinFlags$ (initializing $MinFlag_i[k] = flag_{stop}[k] = 1$, then, after $D$ steps, all nodes can terminate simultaneously their average calculation operation because the flags $MinFlag_i[k + D] = 1$ for all $i \in X$. A formal description of Algorithm 1 can be found below.

**Algorithm 1** Distributed Simultaneous Termination

**Input:** A strongly connected undirected graph $G = \{X,E\}$, where $X = \{1,2,...,n\}$ and $E \subseteq X \times X - \{(i,i) \mid i \in X\}$.

**Initialization:** Each node $i$ sets $y_i[0] = V_i, z_i[0] = 1$. For $k = 0,1,2,...$, each node $i$ does the following:

- **Broadcast:** $y_i[k] = \frac{1}{|N_i|} \sum_{j \in N_i} y_j[k], z_i[k] = \frac{1}{|N_i|} \sum_{j \in N_i} z_j[k]$
- **Receive:** $y_j[k], z_j[k]$ from all transmitting $j \in N_i \cup \{i\}$

**Local Consensus:** If check $(i,j) = |r_i[k] - r_j[k]| < \varepsilon$ for all $j \in N_i \cup \{i\}$

then

$flag_{stop}[k] = 1$

else

$flag_{stop}[k] = 0$

**Update MinFlag:**

- **Initialize** (only every $D$ steps):
  $MinFlag_i[\ell D] = flag_{stop},[\ell D]$
- **Update:**
  $MinFlag_i[(\ell + 1)D] = 1$

then

- **Terminate average consensus calculation**

**Update Values:**

\[ y_i[k+1] = \sum_{j \in N_i \cup \{i\}} y_j[k](1 + \mathcal{G}_j) \]

\[ z_i[k+1] = \sum_{j \in N_i \cup \{i\}} z_j[k](1 + \mathcal{G}_j) \]

**Theorem 1:** Consider the setting in Section III-A. Each node $i$ has some initial value $V_i$ and runs Algorithm 1. Then, all nodes stop simultaneously at the same time step $f$ and the final values that the nodes obtain satisfy $(D \times \varepsilon)$-approximate average consensus, i.e.,

\[ \left| \frac{y_i[f]}{z_i[f]} - \overline{x} \right| \leq D \times \varepsilon, \forall i \in X, \]

where $D$ is the diameter of the graph that describes the communication topology.

**Proof:** Since the ratios converge to the average of the initial values, there clearly exists some time step $k_0$, such that, for each node $i \in X$, we have the absolute differences $|r_i[k_0] - r_i[k_0]| \leq \varepsilon$ for all $f \in N_i$. We assume without loss of generality that before this point, there is at least one pair of nodes that is not in local agreement ($\varepsilon$-approximate local
consensus is not reached for at least two neighboring nodes for all time steps \( k < k_0 \). This means that the \( \text{MinFlag} \) state variables of all nodes will be evaluating to zero at time steps that are multiple of \( D \) (thus, all nodes will continue participating in the computation).

Since all nodes have \( \varepsilon \)-approximate local consensus at time step \( k_0 \), that means nodes also have \( (D \times \varepsilon) \)-approximate global consensus at \( k_0 \). Note that all flags will be set to 1 at time step \( k_0 \), i.e., \( \text{flag}_{\text{step1}}[k_0] = 1 \) for all \( i \in X \). If we initiate min-consensus on the \( \text{MinFlag} \) state variables at iteration \( k_0 \), after \( D \) time steps, all the nodes will terminate their process simultaneously because \( \text{MinFlag}_i[k_0 + D] = 1 \) for all nodes \( i \in X \). Note that all nodes keep updating their variables during iterations \( k_0, k_0 + 1, \ldots, k_0 + D \); thus, the ratios of the nodes will be changing. However, we can take advantage of certain monotonicity properties provided by [24] on the maximum and minimum ratios of the double linear iteration: if \( m[k_0] = \min_{i \in X} r_i[k] \) and \( M[k_0] = \max_{i \in X} r_i[k] \), then for all \( k' > k_0 \) and all \( i \in X \), we have

\[
m[k_0] \leq r_i[k'] \leq M[k_0].
\]

Since the nodes were at \( (D \times \varepsilon) \)-approximate global (and average) consensus, this means that the ratios \( M[k_0] \) and \( m[k_0] \) were within a distance of at most \( D \times \varepsilon \) of the average of the initial values; moreover, the ratios \( r_i[k'] \) will also be within a distance of at most \( D \times \varepsilon \) of the average of the initial values.

Note that if time step \( k_0 \) above is not a multiple of \( D \), then a min-consensus iteration will not necessarily be initiated at that exact time step. However, it will be initiated at some later time step \( k_0' \) that is a multiple of \( D \) (since the ratios converge to the average, there will certainly be a time step \( k \), after which the ratios of all nodes maintain \( \varepsilon \)-approximate local consensus for all time steps \( k' \geq k \).

2) Distributed Re-start for Average Calculation Algorithm: The distributed Re-start for Average Calculation algorithm is a time-varying version of the ratio consensus algorithm in (1)-(2) for the first \( D \) time steps of the iterative process. When node \( i \) initiates a request to restart the iterative process for average calculation, then each of its edges (i.e., each edge in the set \( \{ (j,i) \in E \mid j \in \mathcal{N}_i \} \)) becomes active in the next time step and signifies to the neighbors of node \( i \) the start of the iterative process. Thus, the set of active edges at iteration \( k \) forms a directed (and not necessarily connected) graph. Moreover, the set of weights form a column stochastic matrix \( P_i[k] \) that is essentially an identity matrix except that its \( i \)th column satisfies

\[
P_i(l, i)[k] = \begin{cases} \frac{1}{1 + \varepsilon}, & \forall l \in \mathcal{N}_i, \\ 0, & \text{otherwise}. \end{cases}
\]

At the next time step, the neighbors of node \( i \) will be activated; thus, the set of weights will form a column stochastic matrix \( P_i[k + 1] \) that is essentially an identity matrix except that its \( j \)th column, for \( j \in \mathcal{N}_i \cup \{ i \} \), satisfy

\[
P_i(l, j)[k + 1] = \begin{cases} \frac{1}{1 + \varepsilon}, & \forall l \in \mathcal{N}_j, \\ 0, & \text{otherwise}. \end{cases}
\]

During the first \( D \) time steps, nodes implement time-varying iterations of the form \( y[k + 1] = P_i[k]y[k] \) and \( z[k + 1] = P_i[k]z[k] \). After \( D \) time steps, all the nodes in the system will be active and the weight matrix \( P_i[k + D] \) will equal to the matrix \( P \) used in iterations (1)-(2). Even when more than one node initiate at the same time step a re-start request, the process described above still works in the same manner (in fact, it will likely end up requiring less time steps to activate all nodes).

IV. SIMULATIONS

We first compare the proposed algorithm (Algorithm 1) against the Yadav and Salapaka algorithm in [21] (Y&S Algorithm) by carrying our simulations on an undirected graph of six nodes with diameter \( D = 5 \), as shown in Fig. 1. We use the following initial conditions \( x[0] = [a, a, a, a, a, b]^T \), where \( a = 0.5 \) and \( b = 1 \). Using \( \varepsilon = 0.0001 \), Fig. 2 and Fig. 3 present the results of Algorithm 1 against the Y&S Algorithm. The figures show, for each algorithm, the evolution of the node values until termination. As expected both algorithms reach \( \varepsilon \)-approximate average consensus in a finite time and all nodes stop simultaneously at the same time step. In terms of comparisons, we can see that Algorithm 1 stops at step 64 (with \( \varepsilon = 0.0001 \)) compared to the Y&S Algorithm that stops at step 177 (using \( \varepsilon' = \varepsilon \times D \)). For a fair comparison, since the Y&S Algorithm runs the single iterative scheme we also run a simulation for the Y&S Algorithm using the double linear iterative scheme. What we have noticed is that we get the exact same results as our proposed algorithm, but in terms of computational complexity and transmitted information, the Y&S Algorithm transmits more information (packets) due to the need to run the max- and min-consensus iterations that are implemented for simultaneous stopping; our proposed Algorithm 1 only requires to transmit the values for the average calculation and a single bit for the min-consensus (used for simultaneous termination).

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this paper, we proposed two distributed event based protocols that enable the nodes to (i) determine when approximate average consensus is reached and terminate their operation simultaneously, and (ii) re-start the iterative process when one or more nodes are awaken for any reason and a new average calculation of the network is required. We demonstrated the performance of our proposed algorithms via simulation and examples.
Fig. 2. Evolution of node values for the undirected graph in Fig. 1 using Algorithm 1 with $\varepsilon = 0.0001$.

Fig. 3. Evolution of node values for the undirected graph in Fig. 1 using the Y&S Algorithm with $\varepsilon' = \varepsilon \times D$.

**B. Future Directions**

In the future we plan to extend the results of this paper to the case where a variation of our proposed protocol can operate over arbitrary directed graphs. Furthermore, achieving distributed simultaneous stopping for finite time average consensus with privacy guarantees is to be investigated in our future work.

**REFERENCES**


