UNIVERSITY OF CYPRUS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Average Consensus Enhancements for Distributed Stopping and Privacy Enforcement

NIKOLAS E. MANITARA

A Dissertation Submitted to the University of Cyprus in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

May 2019

VALIDATION PAGE

Doctoral Candidate: Nikolas E. Manitara

Doctoral Thesis Title: Average Consensus Enhancements for Distributed Stopping and Privacy Enforcement

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy at the **Department of Electrical and Computer Engineering** and was approved on the 18^{th} of April, 2019 by the members of the **Examination Committee**.

Examination Committee:

Research Supervisor: _____

(Dr. Christoforos N. Hadjicostis, Professor)

Committee Member: _____

(Dr. Charalambos D. Charalambous, Professor)

Committee Member: _____

(Dr. Ioannis Krikidis, Associate Professor)

Committee Member: _____

(Dr. Themistoklis Charalambous, Assistant Professor)

Committee Member: _____

(Dr. Savvas G. Loizou, Assistant Professor)

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

.....[Full Name of Doctoral Candidate]

.....[Signature]

List of Publications

Refereed Journals

J1 Manitara, N. E., and Hadjicostis, C. N. (2016). Distributed stopping for average consensus in undirected graphs via event-triggered strategies. Automatica, 70, 121-127.

J2 Manitara, N. E., and Hadjicostis, C. N. (2018). Distributed stopping for average consensus in digraphs. IEEE Transactions on Control of Network Systems, 5, 957-967.

Refereed Conferences

C1 Manitara, N. E., and Hadjicostis, C. N. (2013, July). Privacy-preserving asymptotic average consensus. In European Control Conference (ECC 2013), pp. 760-765.

C2 Manitara, N. E., and Hadjicostis, C. N. (2013, October). Distributed stopping in average consensus via event-triggered strategies. In 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton 2013), pp. 1336-1343.

C3 Manitara, N. E., and Hadjicostis, C. N. (2014, May). Distributed stopping for average consensus in directed graphs via a randomized event-triggered strategy. In 6th International Symposium on Communications, Control and Signal Processing (ISCCSP 2014), pp. 483-486.

C4 Manitara, N. E., and Hadjicostis, C. N. (2014, July). Distributed stopping for average consensus in directed graphs via randomized event-triggered strategies. In Proceedings of 7th Cyprus Workshop on Signal Processing and Informatics (abstract only).

C5 Manitara, N. E., and Hadjicostis, C. N. (2014, September). Distributed stopping for average consensus using double linear iterative strategies. In 52nd Annual

Allerton Conference on Communication, Control, and Computing (Allerton 2014), pp. 739-746.

ABSTRACT

In this thesis, we study important extensions to the problem of distributed average consensus in multi-agent systems. Distributed average consensus is a problem where each node (agent) has some initial value and the nodes need to compute, in a distributed manner and subject to communication restrictions among the nodes, the average of their values.

The thesis develops and analyzes distributed algorithms that enable the nodes (while calculating the average of these initial values) to also determine when to stop (because approximate average consensus has been reached) and ensure that their privacy is preserved (in the sense that the initial value of a node is not fully exposed to other nodes). We focus on providing these enhancements to linear iterative strategies, in which each node updates its value as a weighted linear combination of its own previous value and the values of its neighbors.

We first develop and analyze a distributed privacy-preserving average consensus algorithm that enables all of the components of a distributed system, each with some initial value, to asymptotically reach average consensus on their initial values, without having to reveal the specific value they contribute to the average calculation. Specifically, we consider a set of components (nodes) that interact via directional communication links (edges) that form a generally directed communication topology (digraph). The proposed protocol can be followed by each node that does not want to reveal its initial value and, under certain conditions on the communication topology, all nodes can calculate the average of their initial values while maintaining privacy (i.e., without exposing the initial values they contribute to the average). We assume that malicious-curious nodes try to identify the initial values of other nodes but do not interfere in the computation in any other way. Accepting a worst case scenario that malicious-curious nodes know the predefined linear strategy and topology of the network (but not the actual values used by the nodes that want to preserve their privacy), we analyze their ability to infer the initial values of other nodes (which may or may not follow the privacy-preserving protocol). Apart from obtaining topological conditions that guarantee that the initial values of certain nodes are not exposed, we also study the ability of the malicious-curious

nodes to estimate the initial values of other nodes and examine conditions that affect privacy preservation.

We also consider how iterative strategies for asymptotic average consensus in undirected and directed graphs (digraphs) can be adapted so that the nodes can determine, in a distributed fashion, a stopping criterion that allows them to terminate the execution of the iteration when approximate average consensus has been reached. The nodes are said to have reached approximate average consensus when each of them has a value that is close (in a way that we precisely define) to the desirable average. The absence of bidirectional communication links makes this task particularly challenging in a digraph (for a pair of nodes, only one of them may be aware of a discrepancy and may have no direct way of informing the other). The proposed algorithms can be used to cap the number of transmissions that are required in order to reach (approximate) average consensus.

Acknowledgements

Completion of this doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude to all of them. First of all, I am indebted, to my advisor Prof. Christoforos N. Hadjicostis for his patience, motivation, immense knowledge, and the consistent encouragement I received throughout the research work. This feat was possible only because of the unconditional support provided by my advisor. Prof. Christoforos N. Hadjicostis, a person with an amicable and positive disposition, always made himself available to clarify my doubts despite his busy schedule and I consider it, a great opportunity to do my doctoral programme under his guidance and to learn from his research expertise. Thank you Sir, for all your help and support.

Besides my advisor, I would like to thank my thesis committee: Prof. Charalambos D. Charalambous, Dr. Ioannis Krikidis, Dr. Themistoklis Charalambous and Dr. Savvas G. Loizou, for their insightful comments and encouragement.

I am also grateful to the administrative staff at the department of Electrical and Computer Engineering, Skevi Ioannou and Vasiliki Mousikou-Dimitriou, for their precious help and encouragement.

My colleagues, Dr. Apostolos Rikos, Dr. Christoforos Keroglou, Dr. Kourtellaris Christos, Dr. Tziortzis Ioannis, have all extended their support in a very special way, and I have gained a lot from them, through our personal and scholarly interactions, and their suggestions at various points of my research programme. I also acknowledge my old pals back at the University of Manchester for their well wishes.

Last but not least, I would like to thank my beloved family. Words cannot express how grateful I am for all the sacrifices you have made on my behalf, and for supporting me spiritually throughout my Ph.D. studies and, in general, in my life.

This thesis

is dedicated to the memory of my beloved mother

Kyriakis Elia (1955 - 2010)

for her advice, her patience
and her faith

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The successful operation of distributed systems lies at the core of the latest technological achievements in the era of the so called *smart and cyberphysical* systems. It has long been recognized that distributed (or decentralized) systems have specific characteristics that are extremely desirable, such as concurrency, resource sharing, openness, fault tolerance, transparency, and more importantly simplicity of implementation. For the above mentioned reasons, distributed systems are increasingly utilized in many applications. High traffic services, large networks, and the Internet itself are examples of distributed systems in which concurrent tasks are broken down and solved, by a multitude of components, using simple distributed algorithms.

A distributed system consists of a collection of autonomous components (in our case nodes), whose actions and reactions involve the passing of messages that they can receive/transmit at any time instant. The nodes in the network interact with their neighboring nodes in order to coordinate their activities and to share the resources of the network. For a network to be functionally efficient and reliable for the purpose that is designed, specific protocols and algorithms must be followed by all the nodes in the network. Probably the simplest and most well known example of a distributed system is the collection of Web servers or more precisely, servers implementing the HTTP protocol that jointly provide the distributed database of hypertext and multimedia documents that we know as the World-Wide Web. Other examples include the computers of a local network that provide a uniform view of a distributed file system and the collection of computers on the Internet that implement the Domain Name Service (DNS). The importance of a distributed system lies in the fact that a large number of applications are employed 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 are designed to run on hardware consisting of many interconnected processors. Pieces of

a distributed algorithm run concurrently and independently, each with limited amount of information, jointly leading to the successful operation of the overall application.
In this thesis we study the following challenges pertaining to the problem of computing the average value of the network in a distributed manner (i.e., assuming each node has some initial value and the objective is to calculate the average of these values).

- The first challenge has to do with privacy enforcement, which has been widely studied and applied in many fields. Data protection still posses serious challenges in distributed systems due to the way information is transmitted and shared in modern networks. The need to release aggregated data (so that the nodes can compute the average of their values) without leaking individual's information about each participant's value has led to the study of privacy preservation techniques, which has been gaining attention in recent years. The major challenge is to create an algorithm that will enable the nodes to preserve the privacy of the initial value they contribute in the consensus calculation of the network. Beyond that, we also describe and analyze the ability of the malicious-curious nodes to estimate the initial values of other nodes.

- The second challenge studied in this thesis is related to the need to reduce the transmissions hence power consumption, and at the same time reduce the complexity needed for reaching agreement on the average value of the network. In particular, when nodes possess certain computational/memory resources, several alternative distributed strategies have been proposed for (exact) average consensus in finite time (see [3–5] and references therein). In this thesis we consider how the nodes can reach *approximate* average consensus in *finite* time while employing appropriately modified versions of simple linear iterative strategies (thus, maintaining advantages in terms of simplicity and reliance to minimal local information). The topic investigates the subject of distributed stopping, i.e., how the nodes can determine when to terminate the execution of the algorithm based on locally available information. This question has received limited attention thus far in the literature, with the notable exception of the works in [1, 6], which we discuss in detail later in Chapter 3 and Chapter 4.

In order to present the results of our work, we will first need to introduce some terminology and background theory, discuss the distributed system model, and provide a mathematical description of the linear iterative scheme that will be used in subsequent chapters.

1.1 Notation

In our work, we use $e_{i,l}$ to denote the column vector of length l with a "1" in its $i-$th position and zeros elsewhere. The symbol $\mathbf{1}_l$ represents the column vector of length l that has all entries equal to "1", and the symbol $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. When the size of the vector or matrix is apparent, we will sometimes drop the corresponding subscript and denote it simply as $e_i, \mathbf{1}$ or $\mathbf{I}$. We will say that an eigenvalue of a matrix $\mathbf{P}$ is simple to indicate that it is algebraically simple. The notation $\mathbf{P}^T$ indicates that transpose of matrix $\mathbf{P}$. We will denote the rank of matrix $\mathbf{P}$ by rank $(\mathbf{P})$, and we will denote the range of matrix $\mathbf{P}$ by range $(\mathbf{P})$. The notation diag $(\cdot)$ indicates a square matrix with quantities inside the brackets on the diagonal, and zeros elsewhere. The expected value of a random parameter P is denoted as $E[P]$, and the probability of an event A is denoted by $\Pr[A]$. The cardinality of a set S will be denoted by $|S|$. For two sets $S1$ and $S2$, the notation $S1 \setminus S2$ denotes all elements in $S1$ that are not in the set $S2$. We also denote an arbitrary field by $\mathbb{F}$, and use the symbol $\mathbb{F}_q$ to denote the finite field of order q.

1.2 Graph-Theoretic Terminology

A graph is an ordered pair G=$\{X,\ E\}$ where X=$\{1, 2, ..., n\}$ (or X=$\{x_1, x_2, ..., x_n\}$) is the set of vertices or components, and $E \subseteq X \times X$ is the set of directed edges (an example of a digraph can be seen in Figure 1.1). 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 $\mathcal{N}_i^- = \{j \mid (i, j) \in E\}$; the number of in-neighbors of node i is called the in-degree of node i and is denoted by $\mathcal{D}_i^-$=$|\mathcal{N}_i^-|$. Similarly, the nodes that receive information from node i are called its out-neighbors and are captured by the set $\mathcal{N}_i^+ = \{l \mid (l, i) \in E\}$; the number of out-neighbors of node i is called the out-degree of node i and is denoted by $\mathcal{D}_i^+$=$|\mathcal{N}_i^+|$. 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). A graph G=$\{X,E\}$ is said to be undirected if $\{(i, j) \in E \Leftrightarrow (j, i) \in E\}$. In such case, $\mathcal{N}_i^- = \mathcal{N}_i^+$ and $\mathcal{D}_i^- = \mathcal{D}_i^+$, and we refer to the neighbors $\mathcal{N}_i = \mathcal{N}_i^- = \mathcal{N}_i^+$ and degree $\mathcal{D}_i = \mathcal{D}_i^- = \mathcal{D}_i^+$ of node i.

A path of length t from node j to node i, $j \neq i$, is a sequence of nodes $j = i_0$, i_1, ..., i_{t-1}, $i_t = i$, such that $(i_l, i_{l-1}) \in E$ for all $l = 1, 2, ..., t$. 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 $d_{\min}(i, j)$ and it is taken to be infinite if there is no path from node j to node i. The graph is strongly connected if there exists a path (of finite length) from each node j to each other node i. By convention, $d_{\min}(i, i) = 0$ for all $i \in X$. The diameter D of graph G=$\{X,\ E\}$ is the longest shortest path between any two nodes, i.e.,

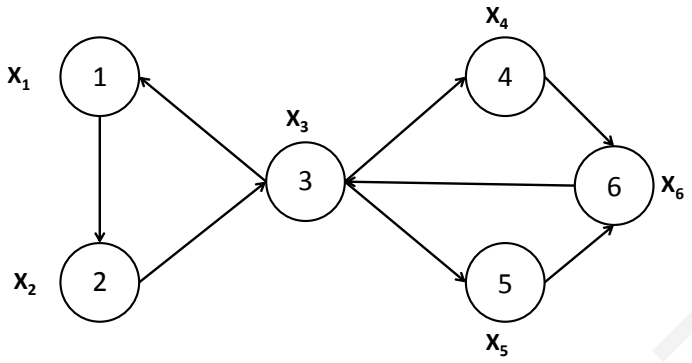

FIGURE 1.1: Example of a digraph.

$D = \max_{i,j \in X, i \neq j} d_{\min}(i,j)$. A graph is said to be strongly connected if there is a path from vertex i to j for every $i, j \in X$. We will call a graph disconnected if there exists at least one pair of vertices $i, j \in X$ such that there is no path from i to j.

A vertex cut in a graph is a subset $S \subset X$ such that removing the vertices in S (and the associated edges) from the graph causes the graph to be disconnected. More specifically, an ij-cut in a graph is a subset $S_{ij} \subset X$ such that the removal of the vertices in S_{ij} (and the associated edges) from the graph causes that graph to have no paths from vertex j to vertex i. We will denote the smallest size of an ij-cut by k_{ij}. If $(j,i) \in E$ (i.e., node j is a neighbor of node i), we will take k_{ij} to be infinite (since removing other vertices will not remove the direct path between j and i). Note that if $\min_j k_{ij}$ is finite, then the in-degree of node i must be at least $\min_j k_{ij}$ (since otherwise, removing all neighbors of node i would disconnect the graph, thereby producing an ij-cut of size less than $\min_j k_{ij}$). The connectivity of the graph is defined as $\min_{i,j} k_{ij}$. A graph is said to be $k-$connected if every vertex cut has cardinality at least k.

Lemma 1.1 *(Fan Lemma)* Let i be a vertex in a graph G, and let c be a nonnegative integer such that $k_{ij} \geq c$ for all $j \in X$. Let $R \subset X$ be any subset of the vertices with $|R| = c$. Then there exists a set of c internally vertex disjoint paths from R to i, where the only common vertex of each of these paths is i.

Since all internally vertex disjoint paths have to pass through different neighbors of i, the Fan Lemma implies that there will be a $c-$linking from R to $\mathcal{N}_i \cup \{i\}$. Note that some of the paths in this linking might have zero length (i.e., if i or some of its neighbors are in R).

1.3 Distributed System Model

In distributed systems, we can model the network topology as a directed graph (digraph) $G=\{X, E\}$ where $X=\{1, 2, ..., n\}$ is the set of components in the system, and $E \subseteq X \times X$ is the set of directed edges (an example of a digraph can be seen in Figure 1.1).
We focus on components that interact via directional links that form a directed (or asymmetric) graph (digraph) that is strongly connected. Our model deals with networks where information is transmitted via a broadcast model, i.e., each node sends to *all* of its out-neighbors the same value, as would be the case in wireless networks. (Note that each node receives generally different values from its in-neighbors.) We assume that, communication links are perfectly reliable, i.e., message exchanges between nodes do not exhibit delays or packet drops, and are received uncorrupted. We also assume that messages are long enough to be able to represent the real value(s) that are being transmitted/received with sufficient accuracy (we ignore quantization effects that arise due to the fact that messages are digitized). 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, comparisons of real numbers, etc.) during the iteration. We also assume that the nodes are synchronized at the granularity of an iteration. An additional assumption that we made for some of the material we develop for distributed averaging in directed graphs is that each node i is aware of its out-degree $\mathcal{D}_i^+$.

1.4 Average Consensus and Linear Iterative Strategies

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 (more generally consensus) has received a lot of attention from the control community due to its usage in various emerging applications, including multi-agent systems [7], and wireless smart meters [8]. Over the last few decades, a variety of algorithms for calculating different functions of these initial values have been proposed by the control [9–11], communication [12], and computer science [13, 14] communities.
One popular approach to consensus (to some value, not necessarily the average) is based on a linear iterative strategy, where each node in the network repeatedly updates its value to be a weighted sum of its own previous value and the previous values of its neighbors. The weights of the linear iteration can be chosen so that all the nodes in the network asymptotically reach agreement to the same value. In particular, earlier work has shown that, if the network topology satisfies certain conditions, the weights for the

linear iteration can be chosen such that all nodes in the network converge asymptotically to the same value (which, under some additional requirements on the weights, can be the average of their initial values [15, 16]). Another popular approach to the calculation of the *average* value of the network is based on ratio-consensus [8, 17, 18] (see also [19] for an enhanced version of this approach that is robust to packet drops and unknown network parameters). Ratio consensus (or push sum) simultaneously runs two linear iterations (of the form described above), and allows each node to asymptotically obtain the average as the ratio of the two state variables it maintains (corresponding to the two concurrent iterations).

1.4.1 Linear Iterative Strategy for Average Consensus

In average consensus problems the objective is the calculation of the average of the initial values of the nodes in the network. Assume that each node i in the network has some initial value $x_i[0] = V_i$ and, at each time-step k, each node updates its value as a weighted sum of its own value and the values of its in-neighbors (e.g., following the method in [16]). Specifically, at each time-step k, each node updates its value as

$$x_i[k+1] = p_{ii}x_i[k] + \sum_{j \in \mathcal{N}_i^-} p_{ij}x_j[k] \,, \tag{1.1}$$

where p_{ij} form a set of (fixed) weights. The values for all the nodes at time-step k can be aggregated into the value vector $x[k] = [x_1[k], x_2[k], ..., x_n[k]]^T$ (where ^{T} denotes matrix/vector transposition) and the update strategy for the entire network can be written compactly as

$$x[k+1] = \underbrace{\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}}_{P} x[k], \tag{1.2}$$

for $k \in \mathbb{N}$, where $p_{ij} = 0$ if $j \notin \mathcal{N}_i^- \cup \{i\}$.

Definition 1.1. *(Asymptotic Consensus)* The system is said to reach asymptotic consensus if $\lim_{k\to\infty} x_i[k] = f(x_1[0], x_2[0], ..., x_n[0])$ for each i, where $f : R^n \to R$.

When $f(x_1[0], x_2[0], ..., x_n[0]) = c^T x[0]$ for some column vector c, the following result from [15] characterizes the conditions under which iteration (1.2) achieves asymptotic consensus.

Theorem 1.2. *[15] Iteration (1.2) reaches asymptotic consensus on the linear functional $c^T x[0]$ (under the technical condition that c is normalized so that $c^T \mathbf{1} = 1$ where $\mathbf{1} = [1, 1, ..., 1]^T$ is the all ones column vector) if and only if the weight matrix P satisfies the conditions below:*

1. *P has a simple eigenvalue at 1, with left eigenvector c^T and right eigenvector $\mathbf{1}$;*
2. *All other eigenvalues of P have magnitude strictly less than 1.*

In particular, if $c = \frac{1}{n}\mathbf{1}$, then average consensus is reached. Also note that if the p_{ij} are restricted to be nonnegative, then the above conditions for asymptotic average consensus are equivalent to P being a primitive doubly stochastic matrix. In an undirected graph, the nodes can easily obtain, in a distributed manner, weights p_{ij} so that matrix $P = [p_{ij}]$ is primitive doubly stochastic, as long as the given graph is connected.
For example, assuming the nodes know the total number of nodes n or an upper bound $n' \geq n$, each node i can choose fixed (nonnegative) weights on its links so that $\sum_l p_{li} = \sum_j p_{ij} = 1, \forall i$, by setting

$$p_{ij} = \begin{cases} \frac{1}{n'}\,, & \text{if } (i,j) \in E\,, \\ 0\,, & \text{if } (i,j) \notin E\,, \qquad j \neq i, \\ 1 - \frac{\mathcal{D}_i}{n'}\,, & \text{if } j = i, \end{cases} \tag{1.3}$$

where $\mathcal{D}_i = \mathcal{D}_i^+ = \mathcal{D}_i^-$. It is easy to check that, as long as the undirected graph is connected, this choice results in a primitive doubly stochastic (and symmetric) weight matrix P.
Another simple choice that results in a primitive doubly stochastic (and symmetric) weight matrix P in connected undirected graphs are the *Metropolis* weights (see, for example, [20]) where

$$p_{ij} = \begin{cases} \frac{1}{1+\max(\mathcal{D}_i, \mathcal{D}_j)}\,, & \text{if } (i,j) \in E\,, \\ 0\,, & \text{if } (i,j) \notin E\,, \qquad i \neq j, \\ 1 - \sum_{j, j \neq i} p_{ij}\,, & \text{if } j = i\,. \end{cases}$$

In both of the above cases, the resulting matrix P is doubly stochastic, primitive and symmetric (as long as the underlying graph G is connected). The rate of convergence to the average consensus is related to $\lambda_2(P)$, the eigenvalue of the doubly stochastic matrix P that has the second largest magnitude. Larger diagonal entries typically lead to an increase in the magnitude of λ_2, resulting to a slower convergence to the average value of the network. For example, a larger value of n' when choosing equal weights leads to a slower convergence rate, this is the main reason the Metropolis weights generally lead to faster convergence compared to the choice of equal weights.

Time varying versions of (1.2) of the form $x[k+1] = P[k]x[k]$ where $P[k] = [p_{ij}[k]]$ for a set of doubly stochastic matrices (induced by time-varying weights $p_{ij}[k]$) have also been studied in [7] and have been shown to lead, under certain conditions[1], to average consensus.

Algorithm 1 Distributed Average Consensus (Single Linear Iteration)

Input: A strongly connected graph $G = \{X, E\}$ with $n = |X|$ nodes
Output: $\bar{x} = \frac{\sum_{x_l \in X} V_l}{n}$
Initialization: Each node i sets its initial value as $x_i[0] = V_i$
Iteration: **For** $k = 0, 1, 2, ...$, each node $x_i \in X$ does the following:
Broadcast: $x_i[k]$
Receive: $x_j[k]$ from all $j \in \mathcal{N}_i$
Update: $x_i[k+1] = p_{ii}[k]x_i[k] + \sum_{j \in \mathcal{N}_i} p_{ij}x_j[k]$
End

Both of the above choices work in undirected graphs, but fail in the case of digraphs because a given node i may not necessarily have $\mathcal{D}_i^+ = \mathcal{D}_i^-$, and it is not as straightforward for nodes to determine appropriate weights so that $\sum_l p_{li} = \sum_j p_{ij} = 1, \forall i$. An iterative algorithm that asymptotically obtains a doubly stochastic matrix in directed graphs can be found in [21].

1.4.2 Average Consensus via Ratio Consensus

Another approach for calculating the average of the initial values of the nodes in the network is the ratio consensus algorithm [8] 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 two state variables $y_i[k]$ and $z_i[k]$, and updates them, at iteration k, as follows:

$$y_i[k+1] = \sum_{j \in \mathcal{N}_i^- \cup \{i\}} \frac{y_j[k]}{(1 + \mathcal{D}_j^+)}, \quad k \geq 0, \tag{1.4}$$

$$z_i[k+1] = \sum_{j \in \mathcal{N}_i^- \cup \{i\}} \frac{z_j[k]}{(1 + \mathcal{D}_j^+)}, \quad k \geq 0, \tag{1.5}$$

where $y_i[0] = V_i$, and $z_i[0] = 1$, for $i \in X$. The protocol assumes that each node j is aware of its out-degree $\mathcal{D}_j^+$. [Note that it makes sense for transmitting node j to simply send the values $\overline{y}_j[k] := y_j[k]/(1+\mathcal{D}_j^+)$, $\overline{z}_j[k] := z_j[k]/(1+\mathcal{D}_j^+)$ 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,

[1] For example, one such condition is the existence of a finite window K such that the products $P[mK + K - 1] \ldots P[mK + 1]\ P[mK]$, for all m, form a primitive doubly stochastic matrix.

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

$$\lim_{k\to\infty} r_i[k] = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \frac{\sum_l V_l}{n}\ , \quad \forall i \in X\ . \tag{1.6}$$

To see that the above holds, one can write the iterations in (1.4)–(1.5) more compactly in the form $y[k+1] = Py[k]$ and $z[k+1] = Pz[k]$, where $y[k]$ ($z[k]$) is an n-dimensional column vector containing the $y_i[k]$ ($z_i[k]$) values for each node, and P is a primitive $n \times n$ column stochastic matrix with weights $P(i,j) = \frac{1}{1+\mathcal{D}_j^+}$ if $i \in \mathcal{N}_j^+ \cup \{j\}$ (zero otherwise). Thus, we have $\lim_{k\to\infty} P^k = \mathbf{v}\mathbf{1}^T$ where $\mathbf{v}$ and $\mathbf{1}^T$ are, respectively, the right and left[2] eigenvectors of P that correspond to its unique eigenvalue at 1. Thus, the ratio in (1.6) will satisfy

$$\lim_{k\to\infty} r_i[k] = \frac{v_i(\mathbf{1}^T y[0])}{v_i(\mathbf{1}^T z[0])} = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \frac{\sum_l V_l}{n}\ , \quad \forall i \in X\ ,$$

where v_i is the ith entry of the right eigenvector $\mathbf{v}$ (notice that $\mathbf{v}$ is strictly positive). 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

$$y[k+1] = P[k]y[k],\ k \geq 0, \tag{1.7}$$
$$z[k+1] = P[k]z[k],\ k \geq 0, \tag{1.8}$$

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 $P[k]$, convergence in (1.6) still holds though the proof is significantly more complex [17–19, 22, 23].

The ratio consensus approach is applicable to asymmetric topologies, at least as long as each node is aware of its out-degree. Most other linear iterative approaches require bidirectional communication links with symmetric weights or unidirectional links with weights that form a doubly stochastic matrix; however, obtaining such a doubly stochastic matrix with distributed algorithms in a digraph setting is not straightforward [21]. Notice also that if the weight matrix P in ratio consensus is doubly stochastic, the second iteration becomes unnecessary, whereas the first iteration reduces to the standard single linear iterative scheme with weights that form a doubly stochastic matrix.

Consider the case when $P[k]$ is a sequence of matrices that satisfies the following conditions:

[2] If the graph is strongly connected, matrix P will be a primitive column stochastic matrix, and will have a single eigenvalue at 1 (with all other eigenvalues having magnitude smaller than one) and a strictly positive right eigenvector $\mathbf{v}$ and left eigenvector $\mathbf{1}^T$ corresponding to this eigenvalue at 1.

Algorithm 2 Distributed Average via Average Ratio Consensus

Input: A strongly connected graph $G = \{X, E\}$ with $n = |X|$ nodes and $m = |E|$ edges (no self-edges). Each node $x_i \in X$ has an initial value V_i and knows its out-degree $\mathcal{D}_j^+$.

Output: $\bar{x} = \frac{\sum_{x_l \in X} V_l}{n}$

Initialization: Each node $x_i \in X$ sets its initial value as $y_i[0] = V_i$ and $z_i[0] = 1$

Iteration: **For** $k = 0, 1, 2, ...$, each node $x_i \in X$ does the following:

Broadcast: $\frac{y_i[k]}{1+\mathcal{D}_i^+}$ and $\frac{z_i[k]}{1+\mathcal{D}_i^+}$ to its out-neighbor $x_l \in \mathcal{N}_i^+$

Receive: $\frac{y_j[k]}{1+\mathcal{D}_j^+}$ and $\frac{z_j[k]}{1+\mathcal{D}_j^+}$ from each in-neighbor $x_j \in \mathcal{N}_i^-$

Update: $y_i[k+1] = \frac{1}{1+\mathcal{D}_i^+} y_i[k] + \sum_{x_j \in \mathcal{N}_i^-} \frac{y_j[k]}{(1+\mathcal{D}_j^+)}$,

$z_i[k+1] = \frac{1}{1+\mathcal{D}_i^+} z_i[k] + \sum_{x_j \in \mathcal{N}_i^-} \frac{z_j[k]}{(1+\mathcal{D}_j^+)}$

End

- At iteration k, the weight matrix $P[k]$ is a row stochastic matrix whose nonzero entries are bounded away from zero (i.e., they are lower bounded by some positive constant).
- There exists a finite window K such that the matrix products $P[mK + K - 1] \dots P[mK + 1]P[mK], m = 0, 1, 2, \dots$, form primitive stochastic matrices.

Note here that the product of row stochastic matrices is necessarily a row stochastic matrix, thus the requirement is that the matrix product is a primitive matrix. The following theorem by Wolfowitz was originally stated for row stochastic matrices in [22], but it is adapted here for collum stochastic matrices. Let us first establish some notation.

Definition 1.1: A column stochastic matrix P_c is called SIA (stochastic indecomposable and aperiodic) if $Q = \lim_{k\to\infty} P_c^k$ exists and has identical columns.

Definition 1.2: Given a column stochastic matrix P_c let,

$$\delta(P_c) = \max_j \max_{i_1, i_2} |P_c(j, i_1) - P_c(j, i_2)|. \tag{1.9}$$

Thus, $\delta(P_c)$ measures how different the columns of P_c are; in particular, $\delta(P_c) = 0$ if and only if P_c has identical columns.

Theorem 1.3. *(Wolfowitz)Let $P_1, P_2, \dots, P_m$ be $n \times n$ column stochastic matrices such that any finite product of them is SIA. For any $\epsilon > 0$, there exists an integer $v(\epsilon)$ such that for $t \geq v(\epsilon)$ any product $B_t = P_{i_t} P_{i_{t-1}} \dots P_{i_1} P_{i_0}$, where $i_k \in \{1, 2, \dots, m\}$ for $k = 0, 1, 2, \dots, t$, satisfies*

$$\delta(B_t) < \epsilon. \tag{1.10}$$

In other words for sufficiently large t, any product of t matrices from the collection $\{P_1, P_2 \dots, P_m\}$ (repetitions allowed) will result in a product B_t with columns that

approximately the same. Note that the above theorem does not imply that B_t necessarily converges.

According to the above mentioned conditions (more details can be found in [24]), it follows from Wolfowitz's Theorem 1.3, that for large enough windows of t, $B_t = P[t]P[t-1]\dots P[0]$ tends to a rank one matrix of the form $w_t \mathbf{1}_n^T$, where w_t is a nonnegative column vector, normalized so that its entries sum to unity.

1.5 Min/Max Consensus

Assume that each node i in the network has some initial value $x_i[0] = V_i$. The distributed Min/Max algorithm allows the nodes to calculate the minimum (or maximum) of these initial values in at most $n-1$ steps. In the Min/Max consensus algorithm each node, updates its value at iteration k, $k = 0, 1, 2, \dots, n-2$, as

$$x_i[k+1] = (min/max)_{x_j \in \mathcal{N}_i^- \cup \{i\}} \{x_j[k]\}. \tag{1.11}$$

From the above execution it can be shown that if the conditions in Section 1.3 are satisfied then each node to i obtains a value $x_i[n-1]$ that satisfies the following

$$x_i[n-1] = (min/max)_l \{x_l[0]\} = (min/max)_l \{V_l\}, \forall x_i \in \mathcal{X}. \tag{1.12}$$

Min/Max consensus is a prototypical example of a finite time algorithm since it completes in a finite number of steps. The pseudocode for the for Min/Max consensus algorithm is shown in Algorithm 3.

Algorithm 3 Min/Max Consensus

Input: A strongly connected graph $G = \{X, E\}$ with $n = |X|$ nodes and $m = |E|$ edges (no self-edges). Each node $x_i \in X$ has an initial value V_i.
Output:$(min/max)_{x_j \in X} V_j$
Initialization: Each node $x_i \in X$ sets its initial value as $x_i[0] = V_i$
Iteration: **For** $k = 0, 1, 2, ..., (n-1)$ each node $x_i \in X$ does the following:
Step 1: It transmits $x_i[k]$ to each out-neighbor $x_l \in \mathcal{N}_i^+$
Step 2: It receives $x_j[k]$ from each in-neighbor $x_j \in \mathcal{N}_i^-$
Step 3: It updates its own variable as
$x_i[k+1] = (min/max)_{x_j \in \mathcal{N}_i^- \cup \{i\}} \{x_j[k]\}$
End

1.6 Background on Linear System Theory and Observability Analysis

The control theoretic perspective that we adopt in this thesis will allow us to introduce and use certain fundamental properties of linear systems in order to present the observability properties, from the perspective of a certain node or nodes, of the linear iterative schemes. In this section, we will review some of the important background material that we will be using in the following chapters. Consider a linear system of the following form

$$x[k+1] = Ax[k] + Bu[k], \tag{1.13}$$

$$y[k] = Cx[k] + Du[k], \tag{1.14}$$

with state vector $x \in \mathbb{F}^N$, input $u \in \mathbb{F}^m$ and output $y \in \mathbb{F}^p$, with $p \geq m$ (for some field $\mathbb{F}$). The matrices A, B, C and D are matrices (of appropriate size) with entries from the filed $\mathbb{F}$. The output of the system over $L+1$ time-steps (for some nonnegative integer L) is given by

$$\underbrace{\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ \vdots \\ y[L-1] \\ y[L] \end{bmatrix}}_{y[0:L]} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{L-1} \\ CA^L \end{bmatrix}}_{\mathcal{O}_L} x[0] + \underbrace{\begin{bmatrix} D & 0 & \dots & 0 & 0 \\ CB & D & \dots & 0 & 0 \\ CAB & CB & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ CA^{L-1}B & CA^{L-2}B & \dots & D & 0 \\ CA^LB & CA^{L-1}B & \dots & CB & D \end{bmatrix}}_{\mathcal{M}^L} \underbrace{\begin{bmatrix} u[0] \\ u[1] \\ u[2] \\ \vdots \\ u[L-1] \\ u[L] \end{bmatrix}}_{u[0:L]} \tag{1.15}$$

When $L = N-1$, the $\mathcal{O}_L$ matrix in the above expression is termed *observability* matrix for the pair (A, C), and the matrix $\mathcal{M}_L$ is termed *invertibility* matrix for the pair (A, B, C, D). Based on (1.15), we can ask certain type of questions such as:

- If the initial state of the system is not known, but the inputs are completely known, what can one infer about the initial state by examining only the outputs in (1.14).
- When the initial state and the inputs are not known, what can one infer about these quantities based on the above equation.

Actually, in the next chapter we focus on these questions (from the perspective of malicious-curious nodes) and give answers on how good their estimates of the initial state can be and how the quality of estimation gets affected by the connectivity of

the network and other protocol parameters. More details about observability of linear systems can be found in [25].

1.7 Contributions of Thesis

The main contribution of this thesis is the development of distributed algorithms and protocols that enable the nodes to (i) preserve their privacy while reaching agreement on the average value of the network and (ii) determine in a distributed manner, when to stop (after a finite number of steps of an asymptotic consensus computation). Our analysis produces the following key results.

- We propose a novel algorithm which enables all of the components of a distributed system, each with some initial value, to asymptotically reach average consensus on their initial values, without having to reveal the specific value they contribute to the average calculation. The proposed algorithm can be followed by each node that does not want to reveal its initial value and, under certain conditions on the information exchange that we characterize precisely, all nodes can calculate the average of their initial values while maintaining privacy (i.e., the initial values contributed to the average by the nodes that follow the protocol are not exposed to malicious nodes). We assume that malicious nodes try to identify the initial values of other nodes but do not interfere in the computation in any other way; malicious nodes are assumed to know the predefined strategy and topology of the network (but not the actual values used by the nodes that want to preserve their privacy).

- Within the above context, we also investigate the ability of the malicious-curious nodes to estimate the initial values of the nodes that follow the proposed privacy preserving protocol, and we provide computational studies to analyze the quality of the estimate obtained by the malicious-curious nodes.

- We consider how iterative strategies for asymptotic average consensus in undirected and directed graphs (digraphs) can be adapted so that the nodes can determine, in a distributed fashion, a stopping criterion that allows them to terminate the execution of the iteration when *approximate* average consensus has been reached. The nodes are said to have reached approximate average consensus when each of them has a value that is close (in a way that we precisely define) to the desirable average.

- We separately investigate iterative strategies for asymptotic average consensus in a digraph, since the absence of bidirectional communication links makes this task particularly challenging, due to the presence of asymmetric information (for a pair

of nodes, only one of them may be aware of a discrepancy and may have no direct way of informing the other).

- The proposed algorithms can be used to cap the number of transmissions that are required in order to reach (approximate) average consensus, and we provide simulation studies that analyze performance with respect to this metric, and provide comparisons against existing work.

1.8 Thesis Organization

This thesis is organized as follows. Chapter 1 has provided some important background on graph theory and has described linear iterative methods for asymptotically reaching agreement on the average value of the network. We next introduce in Chapter 2 our proposed privacy preserving average consensus strategy and analysis of the capability of the malicious-curious nodes to estimate the initial values of the nodes that follow the proposed privacy preserving protocol. In Chapter 3 we present distributed algorithms that accomplish the calculation of approximate average consensus in finite time in undirected graphs. In Chapter 4 we present two algorithms, one randomized and one deterministic, that enable the nodes in a directed graph to identify (only by passing messages) when approximate average consensus has been reached. Finally, in Chapter 5 we conclude this thesis with a brief summary and remarks about future work and extensions.

Chapter 2

PRIVACY PRESERVING ASYMPTOTIC AVERAGE CONSENSUS

With increasing dependence on technology, it has become vital to secure every aspect of data exchanged/stored in various kinds of networks. As the number of smart systems and networks grows, data privacy has emerged as one of the most challenging research topics under investigation. Data privacy relates to how information or data should be handled based on its relative importance. Taking into account that data breaches occur when an outside entity can penetrate databases or channels and access sensitive information, data privacy is not only important but necessary.

This chapter addresses the topic of privacy-preserving asymptotic average consensus in a distributed network. We develop and analyze a distributed algorithm that enables all of the nodes to calculate the *exact* average of their initial values, without loss of privacy and despite the presence of possibly multiple malicious nodes. Malicious nodes are assumed to have full knowledge of the protocol and are allowed to collaborate arbitrarily among themselves, but do *not* interfere in the computation of the average value of the network in any other way (this is why we also refer to them as "malicious-curious" nodes).

Our approach does not depend on any cryptographic algorithm, but operates by allowing the nodes to introduce pseudo-random offsets (unknown to the malicious nodes). Specifically, the proposed protocol is a variation of the standard protocol [16] that is used in the absence of privacy requirements and that allows the nodes to asymptotically obtain the average of their initial values by following a linear iteration with weights that form a doubly stochastic matrix. The main change is that, at each time-step, each node following the protocol adds an arbitrary offset value to the result of its iteration, in an effort to avoid revealing its own initial value as well as the initial values of other nodes. What is important is for each node to ensure that the total (accumulated sum of) offsets

that it adds cancel themselves out in the end.

We establish that, under certain conditions on the communication topology, this protocol allows the nodes to calculate the (exact) average of their initial values in a privacy-preserving manner, despite the presence of malicious agents. For example, we establish that even when a node following the protocol is directly connected to the malicious nodes, but has at least one neighbor that is not directly connected to malicious nodes and whose path(s) to the malicious nodes is (are) through at least one node following the protocol, then privacy is ensured for both nodes (the one following the protocol and this neighbor of it), in the sense that their individual initial values are not revealed to the malicious nodes. Note that, it might still be possible for the malicious nodes to determine the sum of the initial values of nodes that follow the privacy-preserving protocol (but not their individual values).

We also show that, even if there is (are) path(s) to malicious nodes that are not through nodes following the protocol, the nodes following the protocol will still remain protected (i.e., their initial values will not be exposed to the malicious nodes) at least under certain conditions on the communication topology. Apart from obtaining topological conditions that guarantee that the initial values of certain nodes are not exposed, we also study the ability of the malicious-curious nodes to estimate the initial values of other nodes and examine conditions that affect privacy preservation. We include a numerical example and simulations that illustrate the features of the proposed algorithm, and also the ability of the malicious-curious nodes to estimate the initial values of the nodes following the proposed protocol.

This chapter is organized as follows. In Section 2.1 we present previous work on privacy-preserving average consensus schemes. In Section 2.2 we introduce the problem statement of the chapter. In Section 2.3 we introduce our proposed privacy-preserving average consensus strategy, and the main results of the chapter. In Section 2.4 we provide examples and in Section 2.5 we conclude with results and observations from a computational study.

2.1 Previous Work on Privacy-Preserving Average Consensus

Privacy-preserving average consensus in the presence of malicious agents in the network has received extensive attention thus far. In this section we briefly describe some of this earlier work that is related to the developments in this chapter.

The authors of [26] proposed a transformation method using random offset values in a cooperative wireless network. Specifically, each node i that wishes to protect its privacy adds a random offset value u_i to its initial value V_i. This ensures that its initial value

will not be revealed to malicious (curious) nodes that might be observing the exchange of values in the network. The idea is based upon the observation that, when an infinite number of nodes employ the protocol, their offsets will have a zero net effect on the average, allowing the nodes to converge to the true average value of the network, at least with high probability. Specifically, each node i sets $x_i[0]{=}x_i'{=}\ V_i{+}u_i$ where u_i, $i = 1, 2, ..., N$, are i.i.d. random variables with zero mean and finite variance $var(U_i)$. Then, following the protocol for asymptotic average consensus, the nodes converge to

$$\frac{1}{N}\sum_{i=1}^{N} x_i' = \underbrace{\frac{1}{N}\sum_{i=1}^{N} V_i}_{\overline{x}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} u_i}_{U}, \tag{2.1}$$

where $\overline{x}$ is the desirable average of the original initial values and U is a random variable that captures the net effect of the offsets. Since, the u_i are i.i.d. with mean $E[U_i] = 0$ and variance $var(U_i)$, we have that U is zero mean and has variance $var(U) = \frac{1}{N} var(U_i)$. Specifically, as $N \to \infty$ we have $var(U) \to 0$. This means that

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} x_i' = \lim_{N\to\infty} \left[\frac{1}{N}\sum_{i=1}^{N} V_i + \frac{1}{N}\sum_{i=1}^{N} u_i\right] = \overline{x} + 0 = \overline{x}, \tag{2.2}$$

where the equality is in the mean square sense.

For a large number of nodes ($N \to \infty$), this method can give results very close to the true average of the network; however, as the number of nodes decreases, the accuracy of this method (in terms of convergence to the exact average value of the network) also decreases, due to the fact that the offset U from the true average is a random variable (with zero mean but) with some finite variance.

Another popular approach for privacy preservation can be found in [27], which proposed a strategy in which the nodes asymptotically subtract their initial offset values added to the network, and characterizes the mean square convergence rate and the covariance matrix of the maximum likelihood estimate on the initial state. More recent approaches that try to preserve privacy via homomorphic encryption have also been proposed in [28, 29]; a key limitation of these latter methods is the reliance on the existence of a node that is universally trusted.

2.2 Problem Statement

Consider a set of components (nodes) that interact via directional links (edges) in a way that forms a directed communication topology (digraph). All nodes follow the single linear iterative strategy as in (1.1), with a doubly stochastic weight matrix W that allows them to reach agreement to the average of their initial values (note that the

weight matrix was denoted by P in (1.1)). Some nodes are malicious-curious and try to identify the initial values of all or some of the nodes in the network. There exists a set of nodes that would like to preserve their privacy by not revealing to other nodes their initial values. We allow some nodes not to follow the privacy preserving protocol in order to investigate the worst-case scenario that this protocol can handle. We also assume that the malicious-curious nodes have full knowledge of the proposed protocol and are allowed to collaborate arbitrarily among themselves (exchanging information as necessary), but do not interfere in the computation of the average value in any other way. Malicious-curious nodes also know

i) The topology of the network (which is assumed to be time-invariant) and nodes that are trying to preserve their privacy.

ii) The observability matrix $\mathcal{O}_{i,L+1}$ (defined later) for any L and any node i that is malicious-curious (this would be the case, for example, if the weight matrix W is known to the malicious nodes).

We prove that under the above assumptions, malicious-curious nodes are not able to estimate the exact initial value of the nodes that follow the proposed protocol, at least under certain conditions on the network topology and the sets of malicious-curious nodes and nodes that follow the privacy preserving protocol. Even when malicious-curious nodes cannot determine the initial values of other nodes exactly, they can always try to form educated estimates of these initial values. For this reason, we also analyze how well malicious-curious nodes can estimate the initial values of the nodes following the proposed protocol based on the information they observe, using minimum mean square linear estimation.

2.3 Proposed Strategy and Main Results

The objective of the proposed strategy is to calculate the average of the initial values of the nodes in the network, while at the same time preserving the privacy of the nodes that opt to utilize the protocol. The scheme that we study in this work makes use of linear iterations as in (1.1) where the weights w_{ij} form a doubly stochastic matrix $W = [w_{ij}]$ (thus, the nodes asymptotically reach consensus to the average of their initial values). The main difference is that node i following the protocol sets its initial value to $x'_i[0] = x_i[0] + u_i$ (where $x_i[0] = V_i$ and u_i is some random offset), and subsequently updates its value as

$$x'_i[k+1] = w_{ii}x'_i[k] + \sum_{j\in\mathcal{N}_i^-} w_{ij}x'_j[k] + u_i[k],\ k = 0, 1, ..., \tag{2.3}$$

where $u_i[k]$ is a pseudo-random value chosen by node i at time-step k. The constraint is that $u_i[k] = 0$ for $k > L_i$ (for some L_i known only to node i) and

$$u_i[L_i] = -\sum_{k=0}^{L_i-1} u_i[k] - u_i. \tag{2.4}$$

At time-step L_i, node x_i effectively cancels the pseudo-random values it has added during the information exchange in the network up to that point.

Protocol Description: Nodes following the protocol run the linear iteration in (2.3) in order to reach asymptotic average consensus. Specifically, node i follows (2.3) with $x_i'[0] = V_i + u_i$ and

i) Chooses a pseudo random offset $u_i[k],\ k = 0, 1, ..., L_i - 1$ for some randomly chosen integer L_i.

ii) Sets

$$u_i[L_i] = -\sum_{k=0}^{L_i-1} u_i[k] - u_i. \tag{2.5}$$

iii) Sets $u_i[k] = 0$ for $k \geq L_i + 1$.

Note that L_i is a random integer number of steps known only to node i. The remaining nodes follow the iteration in (2.3) with zero offsets. Specifically, node i not following the protocol sets $u_i = 0$ and $u_i[k] = 0$ for $k = 1, 2, ...$, which is the standard protocol for reaching average consensus. The pseudo code for the proposed protocol is provided below.

Note that the weight matrix W is assumed primitive doubly stochastic. There are many ways to choose such weights, even in a distributed manner [30] and [31]. For symmetric communication topologies, this choice of weights can be relatively simple (a couple of different ways to assign weights were described in Chapter 1).

Lemma 2.1*:* Following the iteration in (2.3) and in combination with the constraint in (2.4), the network will reach asymptotic average consensus, as long as the weight matrix W is primitive doubly stochastic.

Proof. It is not hard to see that, if we let $L_{max} = \max_i\{L_i\}$, then

$$\sum_{i=1}^{N} x_i'[L_{max} + 1] = \sum_{i=1}^{N} x_i[0];$$

Algorithm 1: Privacy-Preserving Protocol

Input:A strongly connected graph $G = \{X, E\}$ with $n = |X|$ nodes, with node i having initial value V_i. A set of weights $W = [w_{ij}]$ that forms a doubly stochastic matrix is given.

Output:$\bar{x} = \frac{\sum_{x_l \in \mathcal{X}} V_l}{n}$

Initialization

i) Each node i sets $x_i[0] = V_i$ and sets its initial value $x'_i[0] = x_i[0] + u_i$ (where u_i is some random offset).

ii) Each node i that follows the protocol chooses an integer $L_i > 0$ (otherwise $L_i = -1$).

iii) Sets $u_{i,total} = u_i$.

iv) Transmits $x'_i[0]$ to its out-neighbors.

For $k = 0, 1, 2, ...,$ each node $i \in X$ receives $x'_j[k]$ from its in-neighbors and does the following:

- **if** $k < L_i$ **then:**
  $x'_i[k+1] = w_{ii}[k]x'_i[k] + \sum_{j \in \mathcal{N}_i^-} w_{ij}x'_j[k] + u_i[k]$
  $u_{i,total} = u_{i,total} + u_i[k]$
- **if** $k = L_i$ **then:**
  $x'_i[k+1] = w_{ii}[k]x'_i[k] + \sum_{j \in \mathcal{N}_i^-} w_{ij}x'_j[k] - u_{i,total}$
- **if** $k > L_i$ **then:**
  $x'_i[k+1] = w_{ii}[k]x'_i[k] + \sum_{j \in \mathcal{N}_i^-} w_{ij}x'_j[k]$

End

then, using

$$x'_i[k+1] = w_{ii}x'_i[k] + \sum_{j \in \mathcal{N}_i^-} w_{ij}x'_j[k], \quad k = L_{max} + 1, L_{max} + 2, ...,$$

we obtain the average value of the network:

$$\lim_{k \to \infty} x'_i[k] = \frac{1}{N}\sum_{i=1}^{N} x'_i[L_{max} + 1] = \frac{1}{N}\sum_{i=1}^{N} x_i[0] = \overline{x}\ .$$

This concludes the proof of the lemma. □

2.3.1 Analysis of Inference Capability of Malicious-Curious Nodes

Let $P = \{i_1, i_2, ..., i_p\}$ denote the set of nodes following the protocol during a run of the linear iteration. The linear iteration in (2.3) can be expressed as

$$x'[k+1] = Wx'[k] + \underbrace{[e_{i_{1,N}}\ e_{i_{2,N}}\ ...\ e_{i_{p,N}}]}_{B_p} \underbrace{\begin{bmatrix} u_{i_1}[k] \\ u_{i_2}[k] \\ \vdots \\ u_{i_p}[k] \end{bmatrix}}_{u_p[k]},$$

where $e_{i,N} = [0\ 0\ ...\ 1\ ...\ 0]^T$ is an N-dimensional column vector with a single nonzero entry of value 1 at location i. In this iteration, nodes that follow the protocol set $x'_j[0] = x_j[0] + u_j = V_j + u_j$, $j \in P$, whereas nodes that do not follow the protocol set $x'_j[0] = x_j[0] = V_j$, $j \notin P$.

From the perspective of node i, the values seen (by node i) at each time step of the linear iteration can be expressed as

$$y_i[k] = C_i x'[k], \tag{2.6}$$

where C_i is an $(\mathcal{D}_i^- + 1) \times N$ matrix with a single 1 in each row denoting the positions of the state vector $x'[k]$ that are available to node i (these positions correspond to the nodes that are in-neighbors of node i as well as node i itself). The vector $y_i[k]$ denotes the set of values seen by node i during time-step k of the linear iteration [32], [33], [34]. Without loss of generality, we will assume that there is a single malicious node since we can always choose C_i so as to include all the values seen by malicious nodes (which would essentially allow malicious nodes to collaborate arbitrarily among themselves).We also make the worst-case assumption that malicious nodes know the topology and the predefined strategy of the network, hence the set P and matrix B (as well as matrix W).

The set of values seen by node i during the first L time-steps of the linear iteration (assuming that $L < L_{\min} := \min_{j \in P} L_j$) is given by [33]

$$\begin{aligned} y_i[0:L-1] &= \mathcal{O}_{i,L-1} x'[0] + M^p_{i,L-1} u_p[0:(L-1)], & (2.7) \\ &= \mathcal{O}_{i,L-1} x[0] + \mathcal{O}_{i,L-1} B_p u + M^p_{i,L-1} u_p[0:(L-1)], & (2.8) \end{aligned}$$

where $y_i[0:L-1] = [y_i^T[0]\ y_i^T[1]\ ...\ y_i^T[L-1]]^T$, $u = [u_{i_1}\ u_{i_2}\ ...\ u_{i_p}]^T$, and $u_p[0:(L-1)] = [u_p^T[0]\ u_p^T[1]\ ...\ u_p^T[L-1]]^T$. The matrices $\mathcal{O}_{i,L-1}$ and $M^p_{i,L-1}$ can be expressed recursively as

$$\mathcal{O}_{i,L-1} = \begin{bmatrix} C_i \\ \mathcal{O}_{i,L-2} W \end{bmatrix},\ M^p_{i,L-1} = \begin{bmatrix} 0 & 0 \\ \mathcal{O}_{i,L-2} B_p & M^p_{i,L-2} \end{bmatrix},$$

where $\mathcal{O}_{i,0} = C_i$ and $M^p_{i,0}$ is the empty matrix [33].

For $L \geq L_{\min}$, the above equations need to be modified a bit (see also the discussions in Chapter 1) because some nodes introduce offsets that are linear combinations of the

random noise they already injected in the iteration (specifically, node $j \in P$ introduces an offset of $-u_j - \sum_{t=0}^{L_{j-1}} u_j[t]$ at iteration L_j) or simply stop introducing offsets (specifically, node $j \in P$ introduces an offset of zero for iteration $L_j + 1$, $L_j + 2$, and so forth). Nevertheless, the relationship between the values seen by the malicious-curious nodes and the values that are unknown to it (namely, $x[0]$, u, and $u_p[0:L]$) remains similar to the above. In particular, for any iteration L, $L > L_{\max} := \max_{j \in P} L_j$, we have

$$y_i[0:L-1] \;=\; \tilde{\mathcal{O}}_{i,L-1} x[0] + \tilde{M}^p_{i,L-1} \underbrace{\begin{bmatrix} u \\ u_p[0:(L-1)], \end{bmatrix}}_{U_p}, \tag{2.9}$$

where some of the entries in vector U_p are identically zero and $\tilde{\mathcal{O}}_{i,L-1}$ and $\tilde{M}_{i,L-1}$ are appropriately defined matrices.

The above matrices describe the ability of the malicious node i to identify the initial values $x[0]$ of the nodes (as well as the inputs U_p injected by the nodes following the protocol, if necessary). Note that we make the worst-case assumption that the malicious node i knows exactly which nodes follow the protocol and the weights used in iteration (2.3).

2.3.2 Topological Condition for Privacy Preservation

We first examine the question of whether the malicious-curious nodes can determine exactly the initial values of other nodes. More specifically, we establish topological conditions that ensure privacy for the nodes following the proposed protocol despite the presence of malicious agents in the network.

Theorem 2.1. *Consider a fixed network with N nodes described by a digraph $G = \{X, E\}$. Consider the iteration in (2.3) with weights that form a primitive doubly stochastic weight matrix W. Assume that a set of nodes P follow the predefined privacy-preserving strategy in (2.3) with random offsets chosen as in (2.4). Malicious node i will not be able to identify the initial value of $x_j[0] \in P$, as long as j has at least one other node l connected to it for which all paths from l to the malicious node i are through a node j' following the protocol (i.e., $j' \in P$).*

Specifically, if the condition in Theorem (2.3.2) is satisfied, the network will reach average consensus (this follows from Lemma 2.1) and the privacy of the initial values of the nodes following the protocol will be preserved during the linear iteration process.

Proof. Let $X_1[k]$, $X_2[k]$, $X_3[k]$, $X_4[k]$ denote the vectors of values of nodes in sets V_1 (Malicious), V_2 (Protocol), V_3 and V_4 (following the predefined linear strategy for reaching average consensus). Note that sets V_1, V_2, V_3, V_4 are mutually exclusive and their union comprises X, i.e.,$V_i \cap V_j = \emptyset$ for $i, j \in \{1, 2, 3, 4\}$, and $V_1 \cup V_2 \cup V_3 \cup V_4 = X$.

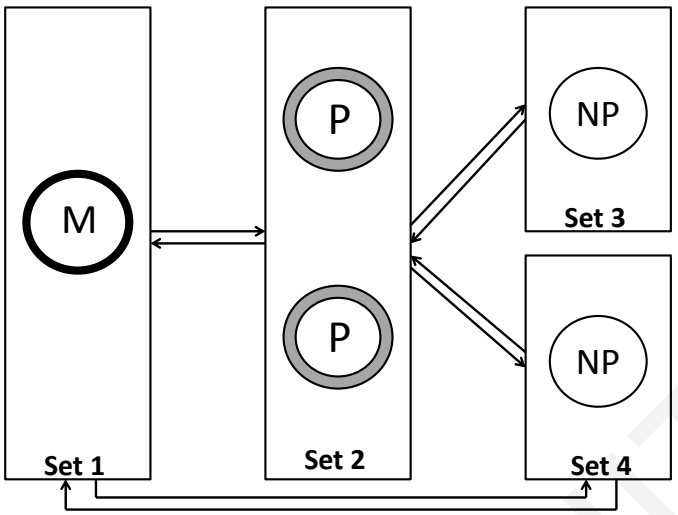

FIGURE 2.1: Example of the key connectivity that guarantees privacy preservation for average consensus: the black node is the malicious set of nodes V_1, the grey nodes are the nodes following the protocol V_2, and the white nodes are the nodes following the predefined strategy for reaching average consensus (V_3 and V_4).

Using the simple network in Figure 2.1, we show that set V_1 (malicious node) is unable to identify the initial values of sets V_2 and V_3 in the network when nodes in set V_2 follow the proposed protocol. To see this, we write the weight matrix as

$$W = \begin{bmatrix} W_{11} & W_{12} & 0 & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ 0 & W_{32} & W_{33} & 0 \\ W_{41} & W_{42} & 0 & W_{44} \end{bmatrix},$$

where $W_{ij},\ i,j \in \{1,2,3,4\}$ are block matrices of appropriate sizes (note that according to the conditions in Theorem 1.2, we have $W_{13} = 0$ and $W_{43} = 0$, and the matrix W is doubly stochastic). The matrix C_1 in (2.7) is given as

$$C_1 = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \equiv \begin{bmatrix} C_{1,1} \\ C_{1,2} \\ C_{1,4} \end{bmatrix},$$

where $C_{1,1}$=$[I\ 0\ 0\ 0]$, $C_{1,2} = [0\ I\ 0\ 0]$ and $C_{1,4} = [0\ 0\ 0\ I]$ (and I are identity matrices of appropriate dimensions). Note that the above C_1 makes a worst case assumption in terms of the malicious nodes being able to directly observe all nodes in set V_2 and all nodes in set V_4. From the definition of matrix $B_p = [e_{i_{1,N}}\ e_{i_{2,N}}\ ...\ e_{i_{p,N}}]$, we can write $B_{p2} = [0\ I\ 0\ 0]^T$ where matrix I is of dimension $|P| \times |P|$.
Using the recursive definition of $\mathcal{O}_{i,L+1}$, and the fact that

$$C_1 \begin{bmatrix} 0 \\ 0 \\ I \\ 0 \end{bmatrix} = 0,$$

we obtain,

$$\mathcal{O}_{1,L+1} \begin{bmatrix} 0 \\ 0 \\ I \\ 0 \end{bmatrix} = \begin{bmatrix} C_1 \\ \mathcal{O}_{1,L}W \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ I \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathcal{O}_{1,L} \end{bmatrix} W \begin{bmatrix} 0 \\ 0 \\ I \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ \mathcal{O}_{1,L} \end{bmatrix} (B_{p2}W_{23} + B_{p3}W_{33}),$$

where $B_{p3} = [0\ 0\ I\ 0]^T$.

Using the above observation, the values seen by the malicious set of nodes V_1 over $L+2$ time steps, given by $y_1[0:L+1] = \mathcal{O}_{1,L+1}x'[0] + M^p_{1,L+1}u_p[0:L]$, can then be written as (see Lemma 2 of [35])

$$y_1[0:L+1] = \mathcal{O}_{1,L+1} \begin{bmatrix} X_1[0] \\ X_2'[0] \\ 0 \\ X_4[0] \end{bmatrix} +$$

$$M^2_{1,L+1} \underbrace{\left(\begin{bmatrix} W_{23} \\ W_{23}W_{33} \\ \vdots \\ W_{23}W^L_{33} \end{bmatrix} X_3[0] + \underbrace{\begin{bmatrix} u_2[0] \\ u_2[1] \\ \vdots \\ u_2[L] \end{bmatrix}}_{e_2[0:L]} \right)}_{\alpha_1}.$$

If we let

$$\alpha = y_1[0:L+1] - \mathcal{O}_{1,L+1} \begin{bmatrix} X_1[0] \\ X_2'[0] \\ 0 \\ X_4[0] \end{bmatrix} \tag{2.10}$$

(note that α is known to the set of V_1 malicious-curious nodes), then we can write

$$\alpha = M^2_{1,L+1}\alpha_1. \tag{2.11}$$

From the above, it can be seen that even if the malicious nodes in set V_1 know (or can determine) $X_4[0]$, which is the assumption we make when we assume that α is known to the malicious nodes, they will not be able to identify the initial values of sets V_2 and V_3 due to the unknowns $X_3[0]$ and $e_2[0:L]$. The reason is that multiple pairs $e_2[0:L]$ and $X_3[0]$ result in the same α_1. From (2.10), it can be seen that the pseudo-random values (protocol) of set V_2 successfully protect set V_3 from revealing its true initial values to malicious nodes, as long as the key connectivity of Theorem 2.3.2 is satisfied. At the same time, this protects nodes in V_2 since multiple $e_2[0:L]$ are possible, even though the malicious nodes know $X_2'[0]$, they cannot determine $X_2[0] = X_2'[0] - \sum_{k=0}^{L} e_2[k]$. □

2.3.3 Analysis of Ability of Malicious-Curious Node to Estimate Initial Values

We now analyze the ability of the malicious-curious node(s) to estimate the values of the nodes that follow the proposed privacy-preserving algorithm, even when they cannot determine the values exactly. For the initial value estimation analysis from the perspective of the malicious-curious nodes we use linear estimation theory, because it is streamlined and straightforward to assess. This analysis is useful when the conditions of the previous section are satisfied meaning that the malicious-curious nodes cannot determine exactly the initial value of other nodes following the protocol, in which case it is useful to determine how well they can estimate them. In order for the malicious-curious nodes to obtain the best estimate for the initial value of the other nodes, they must be able to identify three important time steps in the process of the average calculation. We make the worst case assumption that the malicious-curious nodes have access to the following information.

- The average $\overline{x}$ of the initial values (this will be known eventually to all nodes, including the malicious-curious nodes).
- The time steps where the nodes that follow the protocol cancel out the value of the added noise in the network (denoted earlier by L_j for node $j \in P$) .
- The distribution of the initial values and the distribution of the noise added by the nodes that follow the proposed protocol (however, the malicious-curious node does not know the exact values used by the various nodes).

We consider a fixed network with N nodes described by a digraph $G = \{X, E\}$. The nodes execute iteration (2.3) with weights that form a primitive doubly stochastic weight matrix W. Assume again that the set of nodes P follow the predefined privacy-preserving strategy in (2.3) with random offsets chosen as in (2.4). Malicious node i is interested in collecting data in order to perform initial state estimation for the nodes that follow the proposed protocol.

One important piece of information for the malicious-curious nodes is the average value of the network that becomes known eventually, i.e.,

$$\overline{x} = \frac{1}{n}[1\ 1\ 1\ \ldots\ 1]x[0]. \tag{2.12}$$

Malicious-curious nodes also know the network connectivity and the nodes that follow the proposed protocol, so they can construct the following piece of information.
As mentioned earlier, if we let $P = \{i_1, i_2, i_3, \ldots, i_p\}$ denote the set of nodes that are following the proposed privacy preserving protocol during a run of the linear iteration, the linear iteration can be modeled as

$$x'[k+1] = Wx'[k] + \underbrace{[e_{i_{1,N}}\ e_{i_{2,N}}\ \ldots\ e_{i_{p,N}}]}_{B_p} \underbrace{\begin{bmatrix} u_{i_1}[k] \\ u_{i_2}[k] \\ \vdots \\ u_{i_p}[k] \end{bmatrix}}_{u_p[k]},$$

where $e_{i,N} = [0\ 0\ \ldots\ 1\ \ldots\ 0]^T$ is an N-dimensional column vector with a single nonzero entry of value 1 at location i. What is observed at node i is

$$y_i[k] = \mathcal{C}_i x[k], \tag{2.13}$$

where C_i is $(\mathcal{D}_i^- + 1) \times N$ matrix with a single 1 in each row denoting the positions of the state vector $x'[k]$ that are available to node i (these positions correspond to the nodes that are in-neighbors of node i as well as node i itself). Without loss of generality we will assume that there is a single malicious node since we can always choose C_i so as to include all the values seen by malicious nodes (which would essentially allow malicious nodes to collaborate arbitrarily among themselves).
For simplicity of presentation, assume that for all $j \in P$, we have $L_j = L_{\max}$. The set of all values seen by malicious-curious node x_i during the first L time-steps of the linear iteration (for any nonnegative integer L, $L > L_{\max}$) can be expressed as (see earlier discussion)

$$y_i[0 : L-1] \quad = \quad \tilde{\mathcal{O}}_{i,L-1} x[0] + \tilde{M}^p_{i,L-1} \underbrace{\begin{bmatrix} u \\ u_p[0 : (L_{\max} - 1)] \end{bmatrix}}_{U}, \tag{2.14}$$

where $\tilde{\mathcal{O}}_{i,L-1}$ and $\tilde{M}^p_{i,L-1}$ are appropriately defined matrices, $x[0]$ is the random vector of initial values, and U is an $(1 + L_{\max})|P|$ random vector (containing the variables u_j, $u_j[0], ..., u_j[L_{max} - 1]$, for each $j \in P$).

The matrices $\tilde{\mathcal{O}}_{i,L-1}$ and $\tilde{\mathcal{M}}^p_{i,L-1}$ characterize the ability of the malicious-curious node to perform initial value estimation. Recall that if we have a vector of measurements y that is related to some vector quantity x via an equation of the form

$$y = Ax + Bu \ ,$$

where u is a noise vector, then linear estimation theory says that the linear minimum mean square error estimate for x based on y is given by

$$\hat{x}_{LMMSE}(y) = E[X] + cov(X,Y)\ cov^{-1}(Y,Y)\ (y - E[Y]) \ .$$

Assuming the statistics of x and u are known, then all of the above quantities can be calculated and $\hat{x}_{LMMSE}(y)$ can be obtained easily. Note that the corresponding mean square error is given by

$$MSE = cov(X) - cov(X,Y)\ cov^{-1}(Y,Y)\ cov(Y,X) \ .$$

Clearly, assuming the malicious-curious nodes knows the distributions of $x[0]$, u, and $u_p[0:(L_{\max}-1)]$, and that it can obtain the matrices $\tilde{\mathcal{O}}_{i,L-1}$ it can $\tilde{\mathcal{M}}^p_{i,L-1}$ in (2.14), it can use the above formulas to obtain estimates for initial values of other nodes. This is the approach we follow in the next section to run our simulations.
We will see in the computational study section that different connectivity choices, different numbers of time steps (for canceling the added noise in the network), and other parameters have implications in terms of the quality of the state estimate from the perspective of the malicious-curious node; however, one of the most important factors in ensuring the privacy of the nodes is the topology of the network.

2.4 Example

The network in Fig. 2.2 shows a communication topology that violates the condition of Theorem 2 for both nodes 2 and 5, which are assumed to follow the protocol. We argue that malicious node 1 is unable to identify the initial values of the other nodes in the network of Fig. 2.2 when node 2 and node 5 are following the privacy-preserving protocol. Note that, even when the condition of (2.3.2) is not satisfied it might be possible for the nodes following the protocol to remain protected (in the sense that their initial values will not be revealed to the malicious nodes).
The weight matrix can be written as

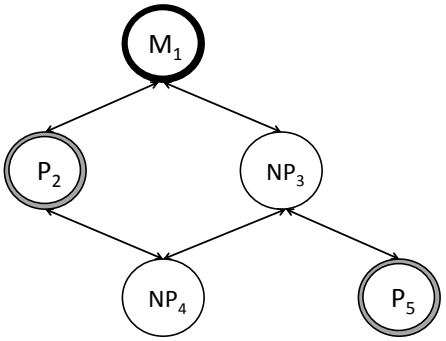

FIGURE 2.2: The black node is the malicious node, the grey nodes are the nodes following the proposed protocol, and the white nodes are the nodes that do not follow the protocol.

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & 0 & 0 \\ w_{21} & w_{22} & 0 & w_{24} & 0 \\ w_{31} & 0 & w_{33} & w_{34} & w_{35} \\ 0 & w_{42} & w_{43} & w_{44} & 0 \\ 0 & 0 & w_{53} & 0 & w_{55} \end{bmatrix}$$

for some nonnegative weights that form a primitive doubly stochastic matrix. The matrix C_1 in (2.7) is given by

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} C_{1,1} \\ C_{1,2} \\ C_{1,3} \end{bmatrix},$$

where $C_{1,1} = [1\ 0\ 0\ 0\ 0]$, $C_{1,2} = [0\ 1\ 0\ 0\ 0]$, and $C_{1,3} = [0\ 0\ 1\ 0\ 0]$.
Using the recursive definition of $\mathcal{O}_{i,L+1}$ we obtain

$$\mathcal{O}_{1,L+1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} C_1 \\ \mathcal{O}_{1,L} W \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathcal{O}_{1,L} \end{bmatrix} W \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

The values seen by the malicious node 1 over $L+2$ time steps are given by

$$y_1[0:L+1] = \mathcal{O}_{1,L+1}\begin{bmatrix} x_1[0] \\ x_2'[0] \\ x_3[0] \\ 0 \\ 0 \end{bmatrix} + \underbrace{M^3_{1,L+1}\begin{bmatrix} w_{34} \\ w_{34}w_{44} \\ \vdots \\ w_{34}w^L_{44} \end{bmatrix} x_4[0]}_{\alpha_1}$$

$$+ \underbrace{M^2_{1,L+1}\left(\begin{bmatrix} w_{24} \\ w_{24}w_{44} \\ \vdots \\ w_{24}w^L_{44} \end{bmatrix} x_4[0] + \begin{bmatrix} u_2[0] \\ u_2[1] \\ \vdots \\ u_2[L] \end{bmatrix}\right)}_{\alpha_2}$$

$$+ \underbrace{M^3_{1,L+1}\left(\begin{bmatrix} w_{35} \\ w_{35}w_{55} \\ \vdots \\ w_{35}w^L_{55} \end{bmatrix} x_5'[0] + \begin{bmatrix} u_5[0] \\ u_5[1] \\ \vdots \\ u_5[L] \end{bmatrix}\right)}_{\alpha_3}$$

If we let

$$\alpha = y_1[0:L+1] - \mathcal{O}_{1,L+1}\begin{bmatrix} x_1[0] \\ x_2'[0] \\ x_3[0] \\ 0 \\ 0 \end{bmatrix} \tag{2.15}$$

(note that α is known to the malicious node), then we have

$$\alpha = \alpha_1 + \alpha_2 + \alpha_3. \tag{2.16}$$

Consider two different scenarios:

i) $x_4[0] = \mu_2$, $x_5'[0] = \mu_4$ and $u_2 = 0$, $u_5 = 0$.

ii) $x_4[0] = \mu_1$, $x_5'[0] = \mu_3$ and $u_2[k] = [w_{24}w^k_{44}](\mu_2 - \mu_1)$, $u_5[k] = [w_{35}w^k_{55}](\mu_4 - \mu_3) + [w_{34}w^k_{44}](\mu_2 - \mu_1)$.

In particular, in the second scenario, nodes x_2 and x_5 are following the protocol and apply the error sequence $u_2[k] = [w_{24}w^k_{44}](\mu_2 - \mu_1)$ and $u_5[k] = [w_{35}w^k_{55}](\mu_4 - \mu_3) + [w_{34}w^k_{44}](\mu_2 - \mu_1)$, $k \in \mathbb{N}$. It is not hard to verify that, the values $y_1[k]$, $k \in \mathbb{N}$, seen by the malicious node x_1, are exactly the same for each of the two above scenarios. This

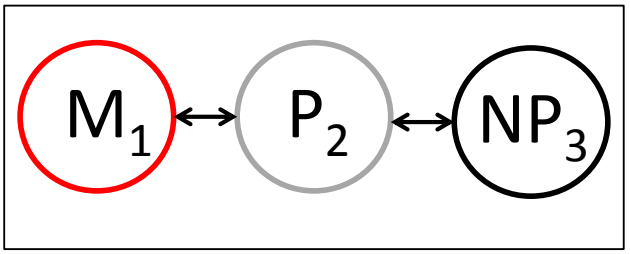

FIGURE 2.3: The simplest form of the key connectivity requirement for privacy preserving average consensus: the red node is the malicious node, the grey node is the node following the protocol, and the black node is the node following the predefined strategy for reaching average consensus.

makes it impossible for node x_1 to obtain the initial values of the nodes following the protocol, hence, the nodes preserve their privacy.

Remark 2: Consider the set-up in Fig. 2.3 which satisfies Theorem 2.3.2. Using the results of Lemma 2.1, we know that the error added by node 2 after $L_{\max}+1$ time steps will be zero and as $k \to \infty$ the network will reach average consensus. This means that the malicious node eventually knows

$$\lim_{k\to\infty} x_1[k] = \frac{1}{3}\sum_{i=1}^{3}\left[x_1[L_{max}+1] + x_2'[L_{\max}+1] + x_3[L_{\max}+1]\right]$$

$$= \frac{1}{3}\sum_{i=1}^{3}\left[x_1[0] + x_2[0] + x_3[0]\right].$$

Since malicious node x_1 knows its own initial value $x_1[0]$ and the average value of the network, we see that the malicious node will eventually know the sum $x_2[0] + x_3[0]$. More generally, the malicious node might be able to eventually determine the sum of the initial values of a subset of the nodes (but not their individual initial values).

2.5 Computational Studies

2.5.1 Computational Study A

Computational studies were carried out in order to compare and calculate the accuracy of the estimates of the malicious-curious nodes when using linear estimation theory. In the previous section we established conditions that ensure that malicious-curious nodes are unable to identify the exact initial values of the nodes that follow the proposed protocol. However, they can attempt to estimate the initial values based on the values they observe and the statistics of the initial values and the noise. For simplicity, we consider the network studied in Section 2.4 (shown in Fig. 2.2), where nodes have initial values that are *i.i.d.* random variables chosen to be uniform in the interval $[-0.5, +0.5]$, and randomly chosen added noise at each time step chosen (from the nodes that follows the proposed protocol) to be *i.i.d* random variables with uniform distribution in $[-0.5, +0.5]$, independent between different nodes and time steps, and independent from the initial values. The malicious-curious node 1 records the transmitted (x'_1) values, and the received values (x'_2 and x'_3) at each time step, and after the network reaches average consensus, malicious-curious node 1 performs state estimation analysis with all stored data and known statistics.

In this part we consider three different scenarios. For each scenario, we assume different input data (information) available to the to the malicious-curious node.

Scenario 1

Let us first assume that the malicious-curious node i knows only the distribution of the initial values. More specially, the initial values are *i.i.d.* random variables, uniform in the interval $[-0.5, +0.5]$. In the absence of any measured data (e.g., the average value and the received values from neighboring nodes) the malicious-curious node will implement a *blind* estimation of the values of other nodes. In other words, the estimate of x_j will be the mean of the value (as determined by the prior distribution of the initial values for each node). In such a case, the estimate will be

$$\hat{x}_{j,LMMSE} = 0$$

for all other nodes j, $j \neq i$ (i.e., the mean of the prior distribution) and the mean square error will be the variance of the distribution of x_j

$$MSE_1 = \int_{-0.5}^{+0.5} x^2 dx = \frac{1}{3}x^3\Big|_{-0.5}^{+0.5} \tag{2.17}$$

$$= \frac{1}{3}2(\frac{1}{8}) = \frac{1}{12}\,. \tag{2.18}$$

Scenario 2

In this scenario, we assume that the malicious node also takes into account the average $\overline{x}$ (which it will discover in the end). In other words, the malicious-curious node i knows the distribution of the initial values, as well as the average value of the network ($\overline{x} = \frac{1}{n}\sum_{l=1}^{N} x_l[0]$). It can now obtain a better estimate since more information is available. If we let $y = \overline{x} - \frac{1}{N}x_i$ (y is known to node i since it knows its own initial value x_i), then one can easily calculate

$$\hat{x}_{j,LMMSE} = y$$

for all other nodes j, $j \neq i$, the j mean square error in this case is

$$MSE_2 = \sigma^2_{x_j} - cov(x_j y)\ cov(yy)^{-1}\ cov(yx), \tag{2.19}$$

where $cov(x_j y) = \left(\frac{1}{N-1}\right)\frac{1}{12}$ and $cov(yy) = \left(\frac{1}{N-1}\right)\frac{1}{12}$, so what we get

$$MSE_2 = \frac{1}{12} - \left(\frac{1}{N-1}\right)\frac{1}{12} \tag{2.20}$$

$$= \left(\frac{N-2}{N-1}\right)\frac{1}{12}\,. \tag{2.21}$$

For example, when $N = 6$, we get $MSE_2 = 0.0667$ whereas for $N = 5$ (as in the case of the network in Fig. 2.2) we get $MSE_2 = 0.0625$. From what we can see from the above analysis, the MSE is smaller from the MSE in Scenario 1, and this is due to the fact that the average value is now known to the malicious-curious node, and this provides additional information. This reduction, however, becomes smaller as N increases.

Scenario 3

In this scenario, the malicious-curious node (node 1 in Fig. 2.2) has all the available information that it can have, including all the values that become available to it during the execution of the iteration (see the y-values in (2.14)). We perform a computational study to examine and compare the quality of linear estimation that the malicious curious node can obtain regarding the initial values of other nodes, including nodes that follow the proposed protocol and nodes that do not.

More specifically, the malicious-curious node uses its knowledge about (i) the distribution of the initial values and the noise that is added by nodes following the protocol, (ii) knowledge of the topology, the weight matrix, and $L_{\max}$ (which is assumed to be the same for all nodes following the protocol), (iii) the average $\overline{x}$ of the initial values, and (iv) the y-values it observes. Its objective is to estimate the values of nodes 2, 3, 4, and

5, i.e., estimate

$$\hat{x}_{LMMSE}(y) = \begin{bmatrix} \hat{x}_{2,LMMSE}(y) \\ \hat{x}_{3,LMMSE}(y) \\ \hat{x}_{4,LMMSE}(y) \\ \hat{x}_{5,LMMSE}(y) \end{bmatrix}.$$

The mean square error in this case is given by MSE_3 and can be calculated using the formulas given earlier. The diagonal entries of the covariance matrix capture the variance associated with the estimates of the initial values of nodes 2, 3, 4, and 5. We consider various scenarios, for different number of iterations following $L_{\max}$. In Fig. 2.4 we plot the calculated covariances from time-step $L_{\max}+1$ up to time-step $L_{\max}+14$ when the nodes reach consensus to the average of the network. (For the computations, we used the regularization technique in order to assure that our matrices will be invertible so we can easily perform all the calculations of the estimated initial values using the standard formulas and without any pre-processing to estimate linear dependencies of the various matrices).

$$Cov = \begin{bmatrix} +0.0075 & +0.0000 & -0.0147 & +0.0074 \\ +0.0000 & +0.0000 & -0.0000 & -0.0000 \\ -0.0147 & +0.0000 & +0.0301 & -0.0148 \\ +0.0074 & -0.0000 & -0.0148 & +0.0076 \end{bmatrix}$$

Now let us investigate the scenario where node 4 also adds some noise and follows the proposed protocol. What can be seen when the matrices bellow are compared with previous results is that the nodes are now more *safe* since there is a small increase in the covariance value of each node, and also node 4 is now more protected and this is due to the fact that more noise is added to the system in the beginning (which increases the uncertainty at the malicious-curious node). In both cases, however, the value of node 3 can be calculated exactly by the malicious-curious node, which was expected.

$$Cov = \begin{bmatrix} 0.0108 & -0.0000 & -0.0215 & +0.0107 \\ +0.0000 & +0.0000 & +0.0000 & -0.0000 \\ -0.0215 & -0.0000 & +0.0430 & -0.0215 \\ +0.0107 & -0.0000 & -0.0215 & +0.0108 \end{bmatrix}$$

We assume that node 2, node 5 and node 3 are now following the proposed protocol. It can be seen from the following result that node 3 is now protected and also there is a small increase in the covariance value of all the nodes resulting in increasing (in a more uniform way) the uncertainty at the malicious-curious node.

$$Cov = \begin{bmatrix} +0.0089 & +0.0036 & -0.0177 & +0.0053 \\ +0.0036 & +0.0284 & -0.0072 & -0.0248 \\ -0.0177 & -0.0072 & +0.0355 & -0.0106 \\ +0.0053 & -0.0248 & -0.0106 & +0.0301 \end{bmatrix}$$

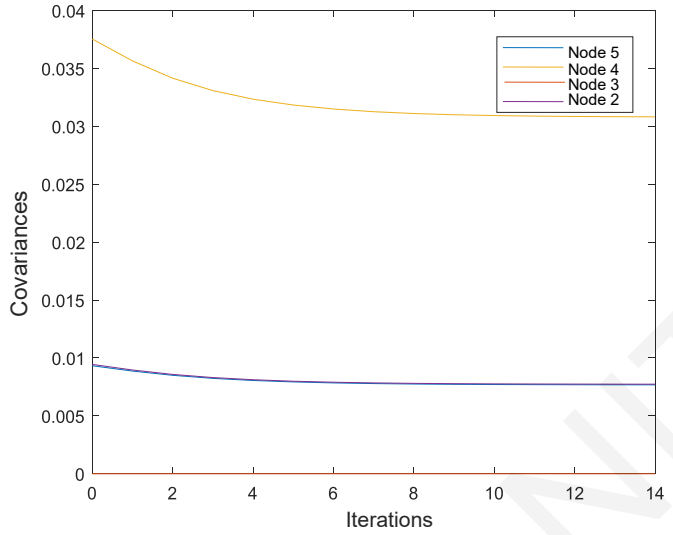

FIGURE 2.4: Values of the Covariance values for each node for the network in Figure 2.2 from time-step $L_{\max}+1$ up to time-step $L_{\max}+14$.

Let us assume that we have zero nodes in the network following the proposed protocol for preserving their privacy. In such case, the malicious-curious nodes will be able to identify the exact initial values of *all* the nodes in the network, and this can be easily seen from the resulting covariance matrix below since we have zero covariance for all the nodes in the network (which that means that the malicious-curious node will perfectly estimate the initial value that each node contributed in the average calculation of the network).

$$Cov = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

.

2.5.2 Computational Study B

In this section, we present computational study results for the network in Fig. 2.5 . In this study we make the assumption that zero nodes in the network follow the proposed protocol, we will show throughout the results that, although the nodes are not following the proposed protocol, some nodes will remain protected due to the connectivity of the network. The resulting covariance matrix indicates that two nodes of the network,

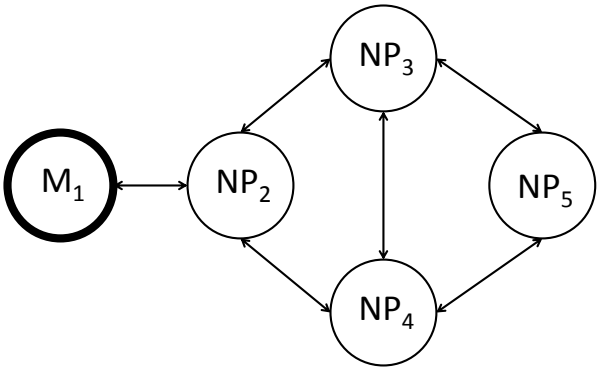

FIGURE 2.5: The black node is the malicious-curious node and the remaining nodes in the network are just following the linear iteration for reaching average consensus.

node 3 and node 4 are protected. Actually malicious-curious node 1 is able to know (identify) the sum of the values of these two nodes (node 3 and node 4) but not the initial value that each node contributed to the average calculation of the network. The covariance matrix is given bellow

$$Cov = \begin{bmatrix} +0.0000 & -0.0000 & -0.0000 & +0.0000 \\ -0.0000 & +0.0411 & -0.0411 & -0.0000 \\ -0.000 & -0.0411 & -0.0411 & -0.0000 \\ +0.0000 & -0.0000 & -0.0000 & +0.0000 \end{bmatrix}.$$

In this part of the computational study we assume that node 3 and node 4 are following the proposed protocol in the network; from what we can see below, there is no significant change in the protection of the nodes. Again, we can see that the covariances of nodes 3 and 4 are approximately equal to the $\frac{1}{2}\sigma_x^2 = \frac{1}{2}0.083 = 0.416$ which is actually what we were expecting, since the malicious-curious node is able to know the average value of the network and any other available information. The main reason that the malicious-curious node calculates this result is the following. In the first iteration ($k = 1$), the malicious-curious node receives the value of node 2, at the next iteration ($k = 2$), the malicious-curious node receives the first accumulated sum of values of node 3 and node 4, and at time-step $k = 3$, it receives the value of node 5 since in the previous two iterations it received the values of all the other nodes (this process of data reception is followed until average consensus is reached). So due to the connectivity and the way the malicious-curious node receives the values, it can identify some node values, but when two or more nodes share a single path at the same time-step, towards the malicious-curious node, the nodes that share the same path at the same time-step (k) will remain

protected even when the proposed protocol for privacy preservation is not followed. The covariance matrix is given below

$$Cov = \begin{bmatrix} +0.0000 & -0.0000 & -0.0000 & +0.0000 \\ -0.0000 & +0.0421 & -0.0421 & -0.0000 \\ -0.000 & -0.0421 & -0.0421 & -0.0000 \\ +0.0000 & -0.0000 & -0.0000 & +0.0000 \end{bmatrix}.$$

We assume now that node 3, node 4 and node 5 follow the proposed protocol in the network. It can be seen from the calculated results that the covariance matrix below is as expected, since now node 5 is also protected and the covariances of the other two nodes are increased compared with their previous values. The main reason that this happens is that, by following the proposed protocol, node 5 adds more noise in the network and this results in the increase of the uncertainty in the calculation of the initial values of node 3 and node 4, and at the same time protects the initial value of node 5.

$$Cov = \begin{bmatrix} +0.0000 & -0.0000 & -0.0000 & +0.0000 \\ -0.0000 & +0.518 & -0.0518 & -0.0202 \\ -0.000 & -0.0316 & -0.0518 & -0.0202 \\ +0.0000 & -0.0202 & -0.0202 & +0.0404 \end{bmatrix}$$

In all the above calculations, we make use of a fixed number of $L_{\max}$ time steps where nodes that followed the proposed protocol were adding noise in the system. Now we will use different windows of $L_{\max}$ time-steps to compare the difference in the covariance matrices that we can have. Consider the network given in Fig. 2.2 where node 2 and node 5 are following the proposed protocol for privacy preserving average consensus. Below, we present three calculated covariance matrices using different values of time-steps such as $L_{\max} = 10$, $L_{\max} = 30$ and $L_{\max} = 100$.

$$Cov[L_{\max} = 10] = \begin{bmatrix} +0.0105 & -0.0000 & -0.0210 & +0.0105 \\ -0.0000 & +0.0000 & -0.0000 & -0.0000 \\ -0.00210 & -0.0000 & +0.0420 & -0.0210 \\ +0.0105 & -0.0000 & -0.0210 & +0.0105 \end{bmatrix}$$

$$Cov[L_{\max} = 30] = \begin{bmatrix} +0.0105 & -0.0000 & -0.0210 & +0.0105 \\ -0.0000 & +0.0000 & -0.0000 & -0.0000 \\ -0.00210 & -0.0000 & +0.0422 & -0.0211 \\ +0.0105 & -0.0000 & -0.0211 & +0.0106 \end{bmatrix}$$

$$Cov[L_{\max} = 100] = \begin{bmatrix} +0.0105 & -0.0000 & -0.0212 & +0.0105 \\ -0.0000 & +0.0000 & -0.0000 & -0.0000 \\ -0.00210 & -0.0000 & +0.0424 & -0.0212 \\ +0.0106 & -0.0000 & -0.0212 & +0.0106 \end{bmatrix}$$

It can be seen from the results that by varying $L_{\max}$ the covariance matrix values remain stable without any significant change in the values, hence we can say that by adding

noise for $L_{\max} > 100$ nothing changes in terms of providing additional protection to the nodes that follow the proposed protocol.

Chapter 3

DISTRIBUTED STOPPING FOR AVERAGE CONSENSUS IN UNDIRECTED GRAPHS

One of the main advantages of the popular approaches for consensus and average consensus discussed in Chapter 1 is the fact that using simple local rules they are able to calculate important quantities like the average. The main problem in the applicability of these techniques, however, is the fact that convergence to the average is asymptotic. Typically, this is handled by a priori determining a number of finite steps that allows the nodes to have values sufficiently close to the average, but to do that they require some knowledge of the network and the convergence rate of the iteration. The approach discussed in this chapter is an alternative that does not require such knowledge and still allows the nodes to reach approximate convergence to the average. In this chapter we consider how the nodes can reach *approximate* average consensus in *finite* time. Perhaps more importantly, this chapter investigates the topic of distributed stopping, i.e., how the nodes can determine when to terminate their transmissions based on locally available information. This question has received limited attention thus far in the control literature, with the notable exception of [1, 6], which we discuss in detail later in the chapter.

In this chapter, we consider distributed systems with communication topologies that are described by undirected graphs, and develop two algorithms, referred to as Algorithm 1 and Algorithm 2, that are event-triggered variations of the two basic linear iterative strategies described in Chapter 1. We establish that all executions of Algorithm 1 and Algorithm 2 stop in finite time, and we prove that, when all nodes eventually stop transmitting, the absolute differences between the final values of the nodes and the exact average of the initial values is smaller than an error bound, whose value depends on a parameter ε and the diameter D of the underlying undirected graph.

The key observation in the proposed algorithms is that each node makes a decision whether to update and/or transmit its value, 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 takes different actions (such as stop communicating with a particular neighbor or stop communicating altogether) depending on whether these differences are smaller than the parameter ε. Note that, during the execution of either algorithm, a link or a node that becomes inactive at a particular time step may be triggered to become active at a later time step if the node value and/or the value of at least one of its neighbors change in a way that makes their absolute difference larger than ε. In both algorithms, the iterative process ends when all nodes cease transmitting their values, in which case they can be shown to have reached approximate average consensus.
This chapter is organized as follows. In Section 3.1 we present previous work on distributed stopping for average consensus in undirected graphs. In Section 3.2 we introduce the problem statement and related concepts of the chapter. In Section 3.3 we introduce our proposed strategy and main results of the chapter. In the last Section 3.4 we provide examples and simulation studies.

3.1 Previous Work on Distributed Stopping Average Consensus in Undirected Graphs

Distributed stopping algorithms for average consensus problems have received limited attention. The authors of [1] proposed a method (referred to here as the Y&S Algorithm) which runs three iterations in parallel in order to identify, in a distributed manner, the time step at which approximate average consensus has been reached. Specifically, each node runs the average consensus algorithm as in (1.2) (with a doubly stochastic matrix P), while also running (in parallel) a max-consensus and a min-consensus iteration:

$$\begin{aligned} M_i[k+1] &= \max_{j\in\mathcal{N}_i^-\cup\{i\}}\{M_j[k]\} \quad \text{(max-consensus)} \\ m_i[k+1] &= \min_{j\in\mathcal{N}_i^-\cup\{i\}}\{m_j[k]\} \quad \text{(min-consensus)} \end{aligned}$$

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 (1.2) more specifically, at iteration lD (for some nonnegative integer l) the max-consensus and min-consensus algorithms are initialized with values $M_i[lD] = m_i[lD] = x_i[lD]$. At iteration $(l+1)D$ each node obtains $M_i[(l+1)D] = \max_j\{x_j[lD]\}$ and $m_i[(l+1)D] = \min_j\{x_j[lD]\}$. When the difference between these two values satisfies the

criterion below, then node i stops iterating. When the criterion is not satisfied, node i continues iterating until

$$|M_i[(l+1)D] - m_i[(l+1)D]| \leq \varepsilon, \tag{3.1}$$

where parameter ε is a given small real number. Note that all nodes will simultaneously stop iterating, at which point $|\max_j\{x_j[lD]\} - \min_j\{x_j[lD]\}| \leq \varepsilon$, and also $|x_i[lD] - \bar{x}| \leq \varepsilon$ for all $i \in X$ [1].

The main difference from what we propose in this work is that the strategy in [1] has the nodes execute (1.2) with fixed weights and imposes an additional mechanism to determine when all nodes have values that are sufficiently close (and also sufficiently close to the average). Instead, the algorithms we propose consider time-varying versions of the iteration in (1.2) or the iterations in (1.7)–(1.8), eliminate the need for max/min consensus iterations, and allow different nodes to stop at different time steps (in fact, a node that has stopped transmitting may later resume transmitting). What is perhaps more important is that the proposed approach avoids the overhead associated with the max- and min-consensus iterations. It is also worth pointing out that, though the approach in [1] also works for directed graphs, the availability of a set of weights that form a doubly stochastic matrix is not as immediate in that case (see, for example, the discussions in [21]).

In the event-based control literature, researchers have also considered distributed algorithms for average consensus. In particular, when we translate to discrete-time the approach in [6] (it was developed for continuous-time systems), we obtain an event-triggering scheme in which (unlike our approach) each node uses a *fixed* set of weights in its updates, but makes sure that it uses its last transmitted value (as opposed to its actual — internal — value, which may be different). Such a scheme indeed leads to node values that are within ε of the average in finite time, but the nodes never stop computing (in other words, internal node values keep changing indefinitely).

3.2 Problem Statement and Related Concepts

All of this section assumes the following setting.

Setting: Consider a network described by a digraph $G = \{X, E\}$, 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 its value following a distributed iterative algorithm for f steps.

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

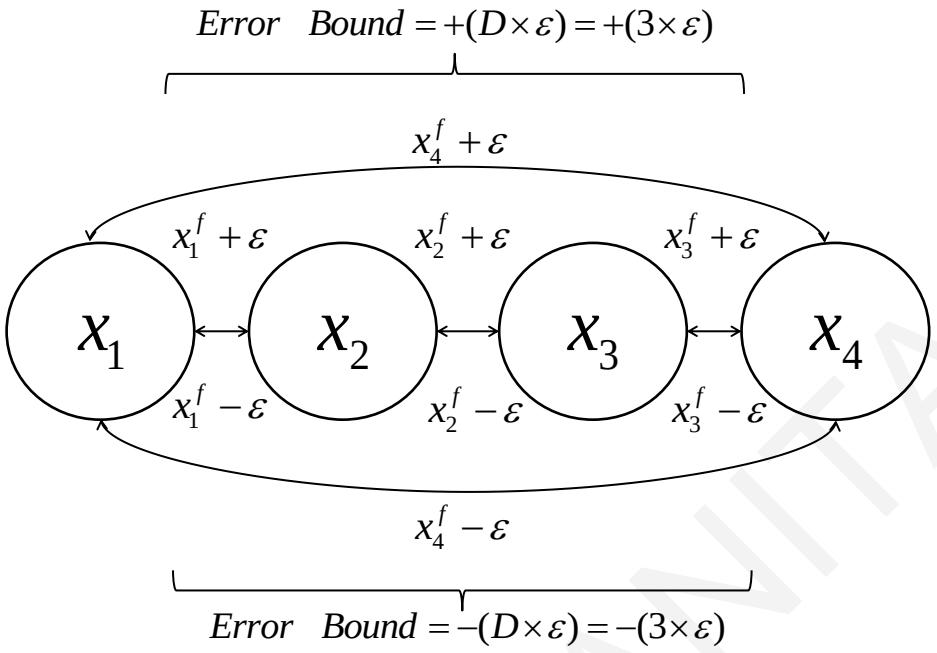

FIGURE 3.1: Digraph with initial error bound for each node and total network error bound.

Definition 3.1. *(ε-Approximate Average Consensus)* At the end of the iterative process in the above setting, the nodes have reached ε-approximate average consensus if the value $x_i[f]$ of each node $i \in X$ satisfies $|x_i[f] - \bar{x}| \leq \varepsilon$, $\forall i \in X$, where $\bar{x} = \frac{1}{n}\sum_{l=1}^{n} V_l$ is the average of the initial values.

Definition 3.2. *(ε-Approximate Local Consensus and ε-Approximate Global Consensus)* At the end of the iterative process in the above setting, the nodes reach ε-approximate *local* consensus if for all nodes $i \in X$, we have $|x_i[f] - x_j[f]| \leq \varepsilon, \forall j \in \mathcal{N}_i^-$ .The nodes reach ε-approximate *global* consensus if $|x_i[f] - x_j[f]| \leq \varepsilon, \forall i, j \in X$.

Proposition 3.3. *If at the end of the iterative process in the above setting the nodes have reached ε-approximate local consensus, then the nodes have also reached (Dε)-approximate global consensus, where D is the diameter of graph G.*

Proof. Let $|x_i[f] - x_j[f]| \leq \varepsilon$ for all i, and all $j \in \mathcal{N}_i^-$ (local consensus). For any $i, j \in X$, $i \neq j$, consider the *minimum* length path (of length t) that connects them (i.e., $(i_1, i_0), (i_2, i_1), ..., (i_t, i_{t-1}) \in E$ and no such path involving $t-1$ or less edges exists). It follows from the definition of the graph diameter that $t \leq D$. The difference $x_j[f] - x_i[f]$ can be written as

$$\begin{aligned} x_j[f] - x_i[f] &= (x_{i_t}[f] - x_{i_{t-1}}[f]) + (x_{i_{t-1}}[f] - x_{i_{t-2}}[f]) \\ &\quad + \ldots + (x_{i_1}[f] - x_{i_0}[f]) \end{aligned}$$

and, using the triangle inequality, we have

$$|x_i[f] - x_j[f]| \leq \sum_{l=1}^{t} |x_{i_l}[f] - x_{i_{l-1}}[f]| \leq t \times \varepsilon \leq D\varepsilon \ .$$

This completes the proof of the proposition. □

Proposition 3.4. *Suppose that, at the end of the iterative process in the above setting, the following are true:*
(1) The nodes reach ε-approximate global consensus;
(2) The average $\bar{x} = \frac{1}{n}\sum_{l=1}^{n} V_l$ satisfies $x_{\min}[f] \leq \bar{x} \leq x_{\max}[f]$ where $x_{\min}[f] \equiv \min_j\{x_j[f]\}$ and $x_{\max}[f] \equiv \max_j\{x_j[f]\}$.
Then, the nodes also reach ε-approximate average consensus.

Proof. Consider two nodes $i, j \in X$, $j \neq i$, and assume without loss of generality that $x_i[f] \geq x_j[f]$. Then we have $|x_i[f] - x_j[f]| = x_i[f] - x_j[f] \leq x_{\max}[f] - x_{\min}[f] \leq \varepsilon$. Since $x_{\min} \leq \overline{x} \leq x_{\max}$ we have that $|x_i[f] - \overline{x}| \leq \varepsilon$.
This completes the proof of the proposition. □

3.3 Proposed Strategies and Main Results

Overview of Algorithm 1: In order to enable the nodes to identify the correct time step to stop transmitting their values, the weights p_{ij} are allowed to be *time-varying* and the iteration in (1.2) becomes

$$x_i[k+1] = p_{ii}[k]x_i[k] + \sum_{j \in \mathcal{N}_i} p_{ij}[k]x_j[k] \ , \tag{3.2}$$

where $p_{ij}[k]$ form a set of time-varying weights. If we aggregate the values for all the nodes at time-step k into the value vector $x[k] = [x_1[k], x_2[k], ..., x_n[k]]^T$, the update strategy for the entire network can be written compactly as

$$x[k+1] = P[k]x[k] \ , \tag{3.3}$$

where the matrix $P[k]$ will be chosen to be doubly stochastic (and symmetric) and the weights will satisfy $p_{ij}[k] = 0$ if $(i, j) \notin E[k]$, with $E[k] \subseteq E$ being the subset of *active* edges. In order to determine the set of active edges $E[k]$ at each time step, each node i compares its value against the value of each of its neighbors $j \in \mathcal{N}_i$: if the absolute difference $|x_i[k] - x_j[k]| > \varepsilon$, then the edge that connects the two nodes is utilized; otherwise, the edge is ignored. In other words, we take

$$E[k] = \{(i, j) \in E \mid |x_i[k] - x_j[k]| > \varepsilon\} \ . \tag{3.4}$$

Given the above choice of $E[k]$, we have a sequence of undirected graphs $G[k] = \{X, E[k]\}$ (since (3.4) is symmetric).

Given an undirected graph $G[k] = \{X, E[k]\}$, we consider three different ways of having the nodes distributively assign the time-varying weights $p_{ij}[k]$ such that the resulting matrix $P[k] = [p_{ij}[k]]$ is doubly stochastic and symmetric (but not necessarily primitive).

(1) *Equal weights:* Assuming the nodes know the total number of nodes n or an upper bound $n' \geq n$, each node i chooses (nonnegative) weights on active links as

$$p_{ij}^{(eq)}[k] = \begin{cases} \frac{1}{n'}\,, & \text{if } (i,j) \in E[k]\,, \\ 0\,, & \text{if } (i,j) \notin E[k], \quad i \neq j, \\ 1 - \frac{\mathcal{D}_i[k]}{n'}\,, & \text{if } j = i, \end{cases} \tag{3.5}$$

where $\mathcal{D}_i[k] = \mathcal{D}_i^+[k] = \mathcal{D}_i^-[k]$ is the degree of node i in graph $G[k]$. We let $P_{eq}[k] = [p_{ij}^{(eq)}[k]]$.

(2) *Metropolis weights:* Another simple choice are the *Metropolis* weights (see, for example, [20]) where

$$p_{ij}^{(M)}[k] = \begin{cases} \frac{1}{1+\max(\mathcal{D}_i, \mathcal{D}_j)}\,, & \text{if } (i,j) \in E[k], \\ 0\,, & \text{if } (i,j) \notin E[k], \quad i \neq j, \\ 1 - \sum_{j, j \neq i} p_{ij}^{(M)}[k]\,, & \text{if } j = i\,, \end{cases} \tag{3.6}$$

where $\mathcal{D}_i$ is the degree of node i in graph G. We let $P_M[k] = [p_{ij}^{(M)}[k]]$.

(3) *Time-Varying Metropolis Weights with Low Self-Weight:* A variation of Metropolis weights that results in a doubly stochastic (and symmetric but not necessarily primitive) weight matrix is provided by

$$p_{ij}^{(tM)}[k] = \begin{cases} \frac{1}{C+\max(\mathcal{D}_i[k], \mathcal{D}_j[k])}\,, & \text{if } (i,j) \in E[k], \\ 0\,, & \text{if } (i,j) \notin E[k], \quad i \neq j, \\ 1 - \sum_{j, j \neq i} p_{ij}^{(tM)}[k]\,, & \text{if } j = i\,, \end{cases} \tag{3.7}$$

where $C > 0$ is a small nonzero constant and $\mathcal{D}_i[k]$ is the degree of node i in graph $G[k]$. We let $P_{tM}[k] = [p_{ij}^{(tM)}[k]]$.

Algorithm 1 essentially involves variations of the execution of (3.3) with weight matrices $P[k]$ given by one of the three choices above ($P_{eq}[kl]$, $P_M[k]$, or $P_{tM}[k]$), where edges $E[k]$ are determined by (3.4). A formal description of Algorithm 1 (for the case when $P[k[= P_{eq}[k])$ is provided below. We will see in the simulations section that different weight choices have implications in terms of the steps and transmissions needed until the nodes stop transmitting and reach approximate average consensus.

Remark 3.5. Suppose that for node i, we have $|x_i[k] - x_j[k]| \leq \varepsilon$ 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.

Algorithm 1 Distributed Stopping for Average Consensus (Single Iteration)

Input: Each node i sets $x_i[0] = V_i$, $p_{ii}[-1] = 0$ and does the following:
For $k \geq 0$, for each node i,
Broadcast: $x_i[k]$ (unless $p_{ii}[k-1] = 1$)
Receive: $x_j[k]\ \forall\ j \in \mathcal{N}_i$
(if no value is received, set $x_j[k] = x_j[k-1]$)
Compute:
For $j \in \mathcal{N}_i$
$check(i,j) = |x_i[k] - x_j[k]|$
if $check(i,j) > \varepsilon$ then
$p_{ij}[k] = \frac{1}{n}$ (active link)
else $p_{ij}[k] = 0$ (inactive link)
End
Set: $p_{ii}[k] = 1 - \sum_{j\in\mathcal{N}_i} p_{ij}[k]$
Update: $x_i[k+1] = p_{ii}[k]x_i[k] + \sum_{j\in\mathcal{N}_i} p_{ij}[k]x_j[k]$
End

Effectively, node i can stop transmitting its value (at least until one of its edges becomes active again). In the simulations in Section 3.4 we compare performance among different stopping algorithms by counting the total number of broadcasts (until the distributed stopping scheme reaches approximate average consensus).

Theorem 3.6. *Consider a network described by a connected undirected graph $G = \{X, E\}$, and the time-varying iteration in (3.3) with weight matrices $P[k]$ obtained according to one of the constructions in (3.5)–(3.7) applied to the graph $G[k] = \{X, E[k]\}$ with edges $E[k]$ given by (3.4). Following Algorithm 1, the nodes reach $(D\varepsilon)$-approximate average consensus after a finite number of iterations.*

Proof. We first establish that the nodes will stop after a finite number of iterations f. The proof is by contradiction: suppose that the iteration runs forever; this means that at each iteration k at least two neighboring nodes are active. Let $A = \{(i_1, j_1), (i_2, j_2), \ldots, (i_\kappa, j_\kappa)\} \subseteq E$ be the *non-empty* set of edges that are active infinitely often. Also, let ℓ be the latest iteration at which an edge in the set $E - A$ is active (note that ℓ is finite). We have that $E[k] \subseteq A$ for $k > \ell$ (i.e., after time step ℓ the active edges belong in the set A).

Since each $E[k]$ results in an undirected graph $G[k] = \{X, E[k]\}$, the graph $\{X, A\}$ is also an undirected graph and can be partitioned into connected components [36]. Suppose that there are q such components, namely $G_1 = \{X_1, A_1\}$, $G_2 = \{X_2, A_2\}$, ..., $G_q = \{X_q, A_q\}$, where $X_1, X_2, ..., X_q$ form a partition of X, and where $A_1, A_2, ..., A_q$ form a partition of A. Moreover, for $k > \ell$, each matrix $P[k]$ is a doubly stochastic matrix (not necessarily primitive because the graph $G[k]$ may be disconnected). Under proper permutation of the nodes, $P[k]$ can be put in block-diagonal form (each block corresponding to one of the connected components in graph $\{X, A\}$). In other words,

$P[k] = \text{diag}(P_1[k], ..., P_q[k])$ where the q blocks may differ in size, but their dimension remains identical at each time step k, $k > \ell$.

Without loss of generality, let us focus at the first block. The node values at iteration $k' > \ell$ in this first connected component of graph G, captured by vector $x^{(1)}[k']$, are given by $x^{(1)}[k'] = P_1[k'-1]...P_1[\ell+2]P_1[\ell+1]x^{(1)}[\ell]$,where $x^{(1)}[\ell]$ are the values of the nodes in this component at iteration ℓ. Since edges in this connected component are activated infinitely often, we can choose k' large enough so that for any finite integer M, we can find M consecutive finite windows of length at most K (for some large enough integer K) such that the products $P_1[mK+\ell+K]\dots P_1[mK+\ell+2]P_1[mK+\ell+1]$, $m = 0, 1, 2, ..., M$, form primitive doubly stochastic matrices. One way to realize this is to choose K large enough so that the union graphs $G_m = \{X_1, \cup_{k=mK+\ell+1}^{mK+\ell+K} E_1[k]\}$ (where $E_1[k] \subseteq A_1$ is the subset of edges in $E[k]$ that belong in A_1) for $m = 0, 1, 2, ..., M$ satisfy $\cup_{k=mK+\ell+1}^{mK+\ell+K} E_1[k] = A_1$ (since edges in A_1 are active infinitely often, we can always choose K large enough). This would imply that the corresponding union graphs G_m are (strongly) connected. Under such conditions, for large enough k', we have $P_1[k'-1]...P_1[\ell+2]P_1[\ell+1] \to \frac{1}{|X_1|}\mathbf{1}\mathbf{1}^T$,i.e., the product of matrices converges (as k' goes to infinity) to the rank one matrix with constant entries equal to $1/|X_1|$ (e.g., see [37] for doubly stochastic matrices or [22] for the more general case of column stochastic matrices). It follows that the values of the nodes in X_1 asymptotically reach the same value (which is actually the average of their values at iteration ℓ since $x^{(1)}[k'] \to (\mathbf{1}^T x^{(1)}[\ell]/|X_1|)\mathbf{1}$). This implies that the edges in the connected component would cease to be active after a finite number of iterations, which is a contradiction.

Clearly, when the nodes stop at some iteration f, we have ε-approximate local consensus (otherwise, at least one edge will be active). By Proposition 3.3, we also have $(D\varepsilon)$-approximate global consensus. Moreover, we clearly have

$$x_{\min}[f] \leq \frac{1}{n}\sum_{l=1}^{n} x_l[f] \leq x_{\max}[f]$$

and, since each matrix $P[k] = [p_{ij}[k]]$ is column stochastic (in fact, the matrices $P[k]$ are all doubly stochastic), we also have $\sum_{l=1}^n x_l[k+1] = \sum_{l=1}^n x_l[k]$, and thus $\frac{1}{n}\sum_{l=1}^n x_l[k+1] = \frac{1}{n}\sum_{l=1}^n x_l[k] = \frac{1}{n}\sum_{l=1}^n V_l = \bar{x}$ for all k. Thus, we establish that $x_{\min}[f] \leq \bar{x} \leq x_{\max}[f]$ and, applying Proposition 3.4, we conclude that the nodes have reached $(D\varepsilon)$-approximate average consensus. □

Overview of Algorithm 2: Algorithm 2 is a time-varying version of the ratio consensus algorithm in Eqs. (1.7)–(1.8). As in the case of Algorithm 1, each edge $(i, j) \in E$ becomes inactive when the two nodes associated with it are in approximate agreement in terms of their ratios (i.e., $|r_i[k] - r_j[k]| \leq \varepsilon$ where $r_i[k] = y_i[k]/z_i[k]$). Thus, the set of active edges at iteration k forms an undirected (but not necessarily connected) graph and weights are

chosen to form a column stochastic matrix $P_c[k]$ (the construction of $P_c[k]$ is described in detail next). Algorithm 2 essentially implements time-varying iterations of the form $y[k+1] = P_c[k]y[k]$ and $z[k+1] = P_c[k]z[k]$.

The set of active edges at time step k resembles the set of active edges in Algorithm 1: $E[k] = \{(i,j) \in E \mid |r_i[k] - r_j[k]| > \varepsilon\}$ and is, of course, symmetric at any given time step. Given $E[k]$, the weight matrix $P_c[k]$ is a column stochastic matrix $P_c[k] = [p_{ij}^{(c)}[k]]$ (not necessarily symmetric and not necessarily primitive) where

$$p_{ij}^{(c)}[k] = \begin{cases} \frac{1}{1+\mathcal{D}_i[k]}\,, & \text{if } (i,j) \in E[k], \text{ or } i = j, \\ 0\,, & \text{if } (i,j) \notin E[k]\,, \end{cases} \tag{3.8}$$

where $\mathcal{D}_i[k]$ is the degree of node i in graph $G[k] = \{X, E[k]\}$. Each diagonal entry $p_{ii}[k]$ is chosen so that the resulting matrix $P[k]$ is column stochastic.

Algorithm 2 is described in detail below. Since nodes may seize transmission, determining whether an edge should be active or not needs to be made against the ratio of the last transmitted values of each node. In Algorithm 2, $lastseenratio_i(j)$ captures the ratio $r_j[k] = y_j[k]/z_j[k]$ of the values of node j, last seen at node i; moreover, node i also updates its own $lastseenratio_i(i)$ each time it transmits its values so that $lastseenratio_j(i) = lastseenratio_i(i)$ for all $j \in \mathcal{N}_i$.

Algorithm 2 Distributed Stopping for Average Consensus (Double Iteration)

Input: Each node i sets $y_i[0] = V_i$, $z_i[0] = 1$, $lastseenratio_i(j) = \infty$ for $j \in \mathcal{N}_i$, $\mathcal{N}_i[0] = \mathcal{N}_i$, and $p_{li}[0] = 1/(1 + |\mathcal{N}_i[0]|)$ $\forall l \in \mathcal{N}_i[0] \cup \{i\}$
For $k \geq 0$, for each node i,
Broadcast: $\bar{y}_i[k] = p_{li}[k]y_i[k]$, $\bar{z}_i[k] = p_{li}[k]z_i[k]$
to all $l \in \mathcal{N}_i$, and update
$lastseenratio_i(i) = \frac{\bar{y}_i[k]}{\bar{z}_i[k]}$
(do not broadcast or update
$lastseenratio_i(i)$ if $p_{li}[k] = 0$, $\forall l \in \mathcal{N}_i$)
Receive: $\bar{y}_j[k]$ and $\bar{z}_j[k]$ from all $j \in \mathcal{N}_i[k]$
and update $lastseenratio_i(j) = r_j[k] = \frac{\bar{y}_j[k]}{\bar{z}_j[k]}$
(if no value is received from $j \in \mathcal{N}_i$,
$lastseenratio_i(j)$ remains unchanged)
Compute:
$y_i[k+1] = p_{ii}[k]y_i[k] + \sum_{j \in \mathcal{N}_i[k]} \bar{y}_j[k]$
$z_i[k+1] = p_{ii}[k]z_i[k] + \sum_{j \in \mathcal{N}_i[k]} \bar{z}_j[k]$
Determine Neighbors:
Set $\mathcal{N}_i[k+1] = \{l \in \mathcal{N}_i \mid |lastseenratio_i(l) - lastseenratio_i(i)| > \varepsilon\}$
Set weights $p_{li}[k+1] = 1/(1 + |\mathcal{N}_i[k+1]|)$, $\forall l \in \mathcal{N}_i[k+1] \cup \{i\}$
End

Theorem 3.7. *Consider a fixed network described by a connected undirected graph $G = \{X, E\}$. Nodes following Algorithm 2 reach $(D\varepsilon)$-approximate average consensus (where D is the diameter of the graph) after a finite time of steps f. The final values that the*

nodes obtain satisfy

$$\left|\frac{y_i[f]]}{z_i[f]} - \bar{x}\right| \leq D\varepsilon \ . \tag{3.9}$$

Proof: We first establish, by contradiction, that the nodes will stop after a finite number of iterations f. If the iteration runs forever, we can use an argument as in the proof of Theorem 3.6 to establish that there is a set of edges $A \subseteq E$ that are active infinitely often such that the graph $\{X, A\}$ is undirected (but not necessarily connected). Letting ℓ be the latest iteration at which an edge in the set $E - A$ is active and re-ordering the nodes, we know that for $k > \ell$ we can write $P_c[k]$ as a block diagonal matrix $P_c[k] = \text{diag}(P_1[k], ..., P_q[k])$ where the q blocks may differ in size, but their dimension remains identical at each time step k. Moreover, each block $P_i[k]$, $i = 1, 2, ..., q$, is a column stochastic matrix (unlike Theorem 3.6, matrices $P_i[k]$ in this case are not necessarily doubly stochastic).

Focusing without loss on generality on the first block, the node values at iteration $k' > \ell$ in this first connected component of graph $\{X, A\}$, captured by vectors $y_1[k']$ and $z_1[k']$, are given by

$$\begin{aligned} y^{(1)}[k'] &= P_1[k'-1]...P_1[\ell+2]P_1[\ell+1]y^{(1)}[\ell] \ , \\ z^{(1)}[k'] &= P_1[k'-1]...P_1[\ell+2]P_1[\ell+1]z^{(1)}[\ell] \ , \end{aligned}$$

where $y^{(1)}[\ell]$ and $z^{(1)}[\ell]$ are the values of the nodes in this component at iteration ℓ. Since edges in this connected component are activated infinitely often, we can choose k' large enough so that, for any desirable integer $M > 0$, we can find consecutive finite windows of length at most K (for some large enough integer K) such that the products $P_1[mK + \ell + K] \dots P_1[mK + \ell + 2]P_1[mK + \ell + 1]$, $m = 0, 1, 2, ..., M$, form primitive column stochastic matrices (see also the proof of Theorem 3.6). Under such conditions, for large enough k', we have [22] $P_1[k'-1]...P_1[\ell+2]P_1[\ell+1] \rightarrow \frac{1}{|X_1|}\mathbf{c}_{k'}\mathbf{1}^T$,where $\mathbf{c}_{k'}$ is a strictly positive column vector (normalized so that its entries sum to $|X_1|$). Notice that in this case there is no convergence of the matrix product (since the column vector $\mathbf{c}_{k'}$ changes with k'); however, as M grows, the product of matrices (increasingly) takes the form of a strictly positive rank one matrix. It follows that the ratio at each node in the connected component asymptotically becomes equal to $(\sum_{i\in X_1} y_i[\ell])/(\sum_{i\in X_1} z_i[\ell])$, i.e., the edges in the connected component would cease to be active after a finite number of iterations, which is a contradiction.

Clearly, when the nodes stop at iteration f, we have ε-approximate local consensus, and (due to Proposition 2) $(D\varepsilon)$-approximate global consensus. Moreover, we have

$$r_{\min}[f] \leq \frac{\sum_{l=1}^{n} y_l[f]}{\sum_{l=1}^{n} z_l[f]} \leq r_{\max}[f] \tag{3.10}$$

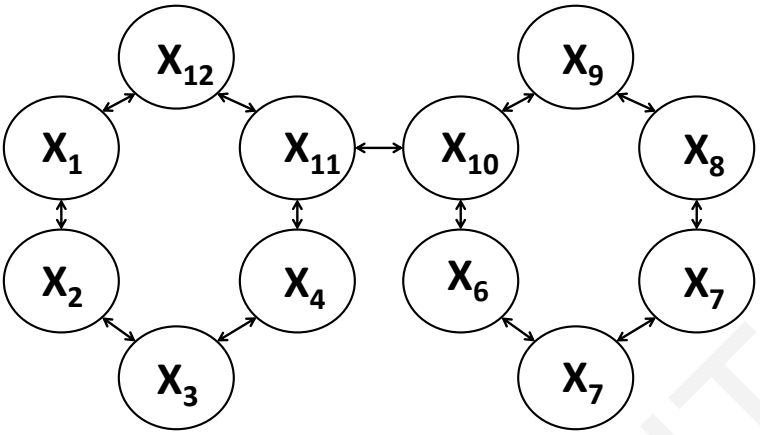

FIGURE 3.2: Example of an undirected graph.

(where $r_{\min}[f] = \min_i \left\{ \frac{y_i[f]}{z_i[f]} \right\}$ and $r_{\max}[f] = \max_i \left\{ \frac{y_i[f]}{z_i[f]} \right\}$). To see this, note that $r_i[f]z_i[f] = y_i[f]$, i.e., $\sum_{l=1}^n r_l[f]z_l[f] = \sum_{l=1}^n y_l[f]$. Since $z_i[f] \geq 0$ for all i, we have

$$r_{\min}[f] \sum_{l=1}^n z_l[f] \leq \sum_{l=1}^n y_l[f] \ , \quad r_{\max}[f] \sum_{l=1}^n z_l[f] \geq \sum_{l=1}^n y_l[f]$$

and, dividing by $\sum_{l=1}^n z_l[f] > 0$, we obtain (3.10). Since the weights at each iteration k form column stochastic matrices, we have $\sum_{l=1}^n y_l[f] = \sum_{l=1}^n y_l[0]$ and $\sum_{l=1}^n z_l[f] = \sum_{l=1}^n z_l[0]$. We conclude that $r_{\min}[f] \leq \bar{x} = \frac{1}{n}\sum_{l=1}^n V_l \leq r_{\max}[f]$, i.e., the nodes have reached $(D\varepsilon)$-approximate average consensus. $\surd$

3.4 Examples and Simulation Studies

3.4.1 Small Graph

We first compare the proposed algorithm (Algorithm 1) against the Y&S algorithm in [1], by carrying out simulations on an undirected graph of twelve nodes with diameter $D = 7$, as shown in Fig. 3.2. We use the following initial values $x[0] = [a, a, a, a, b, b, b, b, b, b, a, a]^T$, where $a = 0.001$ and $b = 100$. Using $\varepsilon = 0.0001$, Fig. 3.5 shows, for each algorithm, the evolution of the node values until termination, and Table 3.1 shows the required number of steps and number of transmissions.

As expected, both algorithms reach approximate average consensus in a finite number of iterations. The two algorithms require about the same number of steps to terminate, but Algorithm 1 requires significantly less transmissions. Note that the way we calculate the number of transmissions is according to the number of values that are required for each

Simulation Results for Graph in Fig. 3.2		
Algorithm	Last Time-Step	No. of Transmissions
Y & S	1059	38376
Algorithm 1	948	11096

TABLE 3.1: Required number of time steps and transmissions for Algorithm 1 and the Y&S Algorithm in [1] for the undirected graph in Fig. 3.2, with $\varepsilon = 0.0001$.

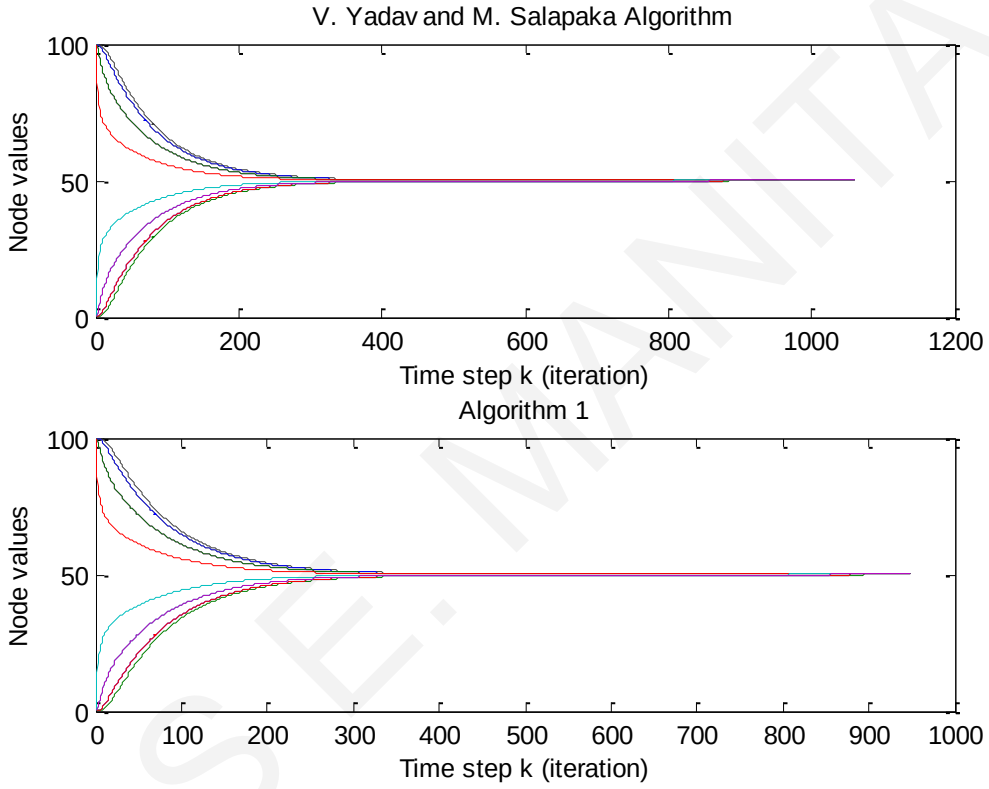

FIGURE 3.3: Evolution of node values for the undirected graph in Fig. 3.2, with $\varepsilon = 0.0001$.

algorithm: for Algorithm 1 a single transmission is required by each *active*[1] node at each time-step (in order to send its value), whereas for the Y&S algorithm proposed in [1], three transmissions are needed from each node (the iteration value $x_i[k]$, the maximum $y_i[k]$, and the minimum $z_i[k]$).

For a fairer comparison of the two algorithms, simulations were carried with different values of ε among the two algorithms, since following the Y&S algorithm in [1] nodes reach ε-approximate average consensus whereas following Algorithm 1 nodes reach $(D \times \varepsilon)$-approximate average consensus. The evolutions of the node values for each algorithm are shown in Fig. 3.4 and the required number of steps and transmissions in Table 3.2.

[1]Recall that a node becomes inactive if *all* links to its neighbors are inactive.

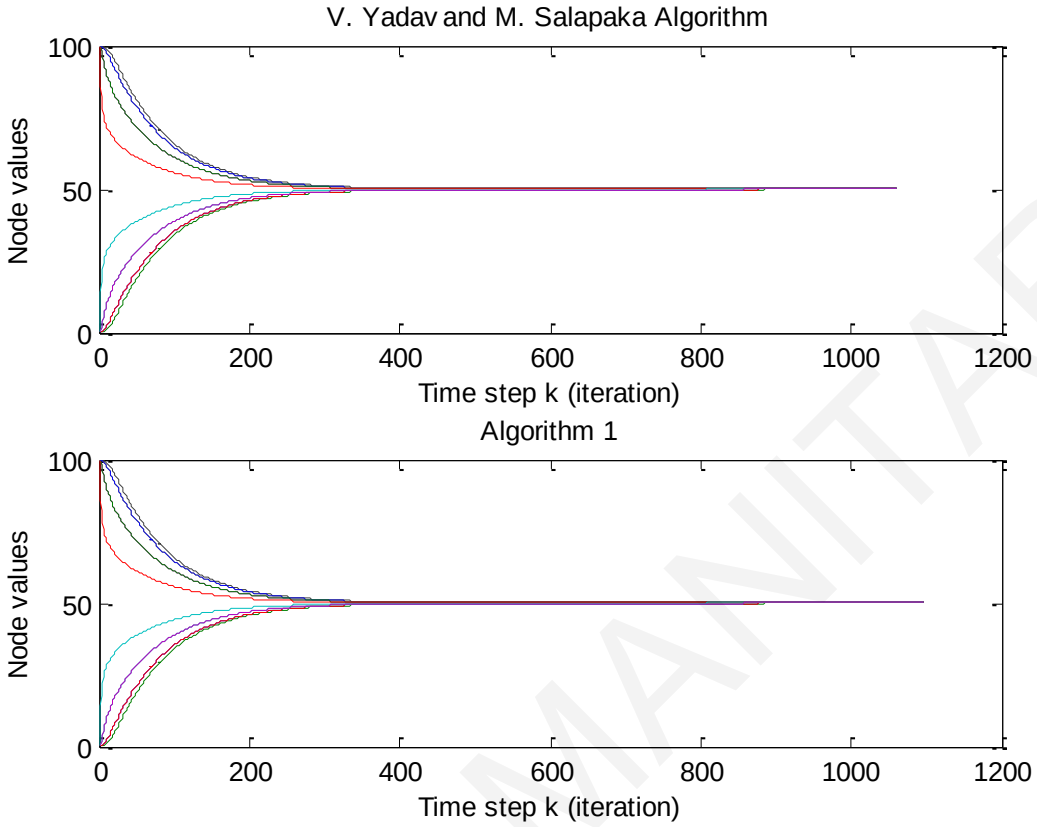

FIGURE 3.4: Evolution of node values for the undirected graph in Fig. 3.2, with parameter $\varepsilon = 0.0001$ for the Y&S Algorithm in [1] and parameter $\varepsilon = 0.0001/D$ for Algorithm 1.

Simulation Results for Graph in Fig. 3.2		
Algorithm	Last Time-Step	No. of Transmissions
Y & S	1059	38376
Algorithm 1	1095	12860

TABLE 3.2: Required number of time steps and transmissions for the undirected graph in Fig. 3.2, with parameter $\varepsilon = 0.0001$ for the Y&S Algorithm in [1] and parameter $\varepsilon = 0.0001/D$ for Algorithm 1.

Comparing the results in Table 3.1 and in Table 3.2, we see that in both cases the number of transmissions that are required for reaching approximate average consensus is significantly smaller with the proposed Algorithm 1 than with the Y&S algorithm in [1].

Finally we illustrate the proposed Algorithms 1 and 2 against the Y&S Algorithm in [1], by carrying out simulations on the undirected graph of twelve nodes and diameter $D = 7$ shown in Fig. 3.2. We take $x[0] = [a, a, a, a, b, b, b, b, b, b, a, a]^T$ with $a = 0$ and $b = 0.1$. For fair comparison of the three algorithms, simulations were carried with different values of ε among the three algorithms. Specifically, we use $\varepsilon = 0.0001$ for the Y&S Algorithm

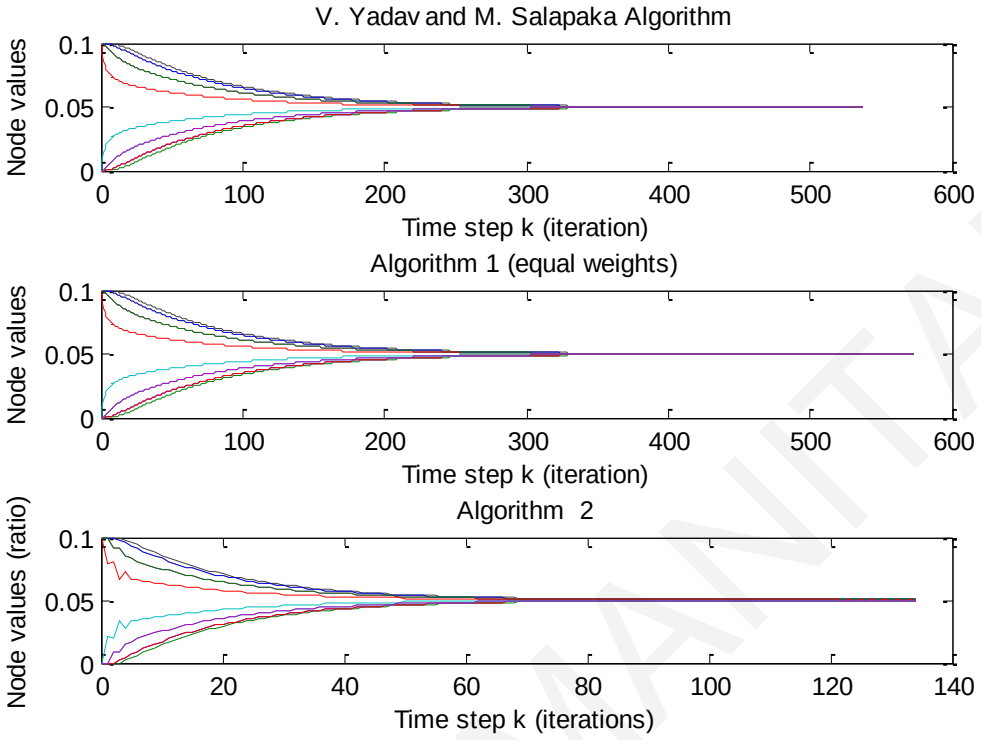

FIGURE 3.5: Evolution of node values for the undirected graph in Fig. 3.2 using equal weights for Algorithm 1.

in [1] and $\varepsilon' = \varepsilon/D$ for the proposed algorithms (so that approximate convergence to the average is of the same order for all algorithms). Fig. 3.5 shows the evolution of the nodes values using the three algorithms. Comparing the results in Table 3.3, we see that in both cases the number of transmitted values that are required for reaching approximate average consensus is significantly smaller (in terms of the last time step and the total transmitted values) for the proposed algorithms compared to the Y&S algorithm in [1]. In particular, time-varying Metropolis weights with low self-weight ($C = 0.02$), result in significant reduction in the number of time steps and transmitted values that are required to reach approximate average consensus.

3.4.2 Random Graphs

In this part we present results based on randomly generated undirected graphs with 100 nodes and probability of connection equal to 0.3 (i.e., for each pair of nodes $i, j \in X$, $i \neq j$, both edges $(i, j), (j, i) \in E$ with probability equal to 0.3). Nodes have initial values randomly chosen to be uniform in the interval $[0, 1]$, independently between each other. We consider 100 random graphs (with different instances of random initial conditions for

Simulation Results for Graph in Fig. 3.2		
Algorithm	Last Time	Transmissions
Y&S Algorithm	534	19476
Algorithm 1(Equal)	573	6596
Algorithm 1(Metropolis)	299	3357
Algorithm 1(Low-self)	145	1620
Algorithm 2	134	3084

TABLE 3.3: Simulation results for the undirected graph in Fig. 3.2, with parameter $\varepsilon = 0.0001$ for Y&S Algorithm and $\varepsilon' = 0.0001/D$ for Algorithm 1 and Algorithm 2.

Simulation Results			
Algorithm	Min. Trans	Max. Trans	Aver. Trans
Y & S	14040	18000	11700
Algorithm 1	3616	4717	4108
Algorithm	Min. Steps	Max. Steps	Aver. Steps
Y & S	37	58	45
Algorithm 1	49	81	60

TABLE 3.4: Minimum, maximum, and average numbers of required time steps and transmissions in simulations for 100 random undirected graphs (with 100 nodes), with parameter $\varepsilon = 0.0001$.

each graph) and run both algorithms; we record the maximum, minimum, and average number of required steps and transmissions for reaching approximate average consensus in Table 3.4. Similarly, in Table 3.5 we present simulation results of 100 random undirected graphs (with 100 nodes), using parameter $\varepsilon = 0.0001$ for the Y&S algorithm in [1] and parameter $\varepsilon = 0.0001/D$ (where D is the diameter of the graph under consideration) for Algorithm 1. As observed previously, Algorithm 1 requires significantly less transmissions for reaching ε-approximate average consensus than the Y&S algorithm in [1] (the number of steps to termination is larger for Algorithm 1, the advantage in the number of transmissions is created due to the need to run the max- and min-consensus iterations in [1]).

Moreover, simulations were carried out for comparing the number of transmissions against the value of parameter ε. It can be seen from Fig. 3.6 that the number of steps to terminate for both algorithms has a linear dependency on the logarithm of parameter ε. In particular, as the value of ε becomes smaller (in the simulations ε varies from 10^{-1} to 10^{-8}), the number of transmissions required for both algorithms to terminate increases monotonically.

From the simulation results, we see that the proposed algorithm (Algorithm 1) requires significantly less transmissions than the Y&S Algorithm, mainly because it is able to converge around the same number of steps but does not require to continuously run the max- and min-consensus iterations.

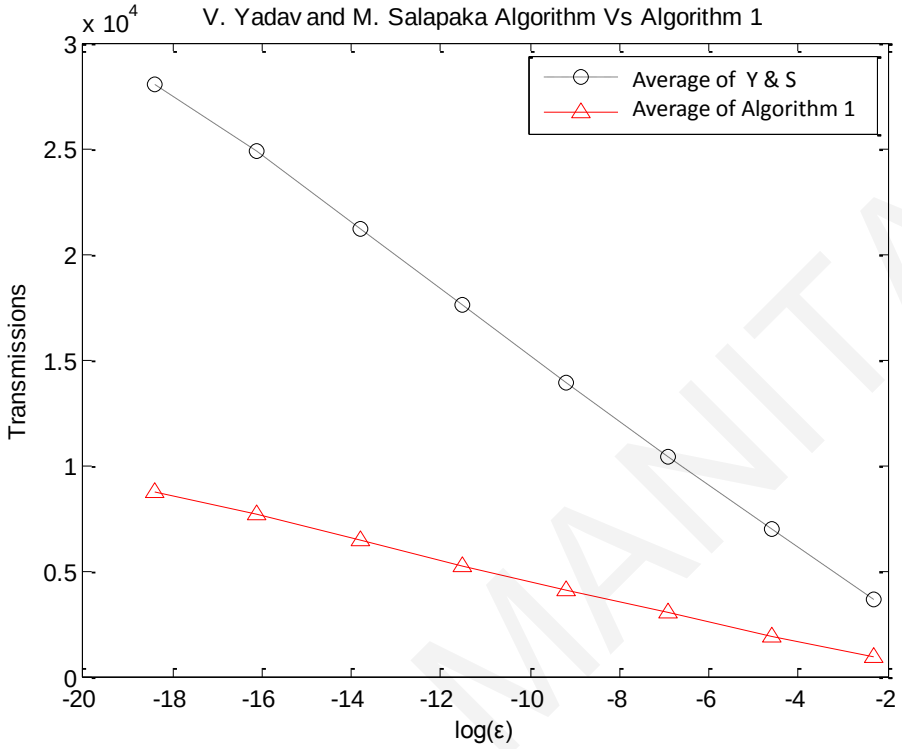

FIGURE 3.6: Average number of transmissions for Algorithm 1 and the Y&S Algorithm, for 100 random graphs with different instances of initial conditions, for each different $\log(\varepsilon)$ point.

Simulation Results			
Algorithm	Min. Trans	Max. Trans	Aver. Trans
Y & S	11700	18000	13833
Algorithm 1	3889	5706	4474
Algorithm	Min. Steps	Max. Steps	Aver. Steps
Y & S	38	57	44
Algorithm 1	53	88	62

TABLE 3.5: Minimum, maximum, and average numbers of required time steps and transmissions in simulations for 100 random undirected graphs (with 100 nodes), with parameter $\varepsilon = 0.0001$ for the Y&S Algorithm and parameter $\varepsilon = 0.0001/D$ for Algorithm 1.

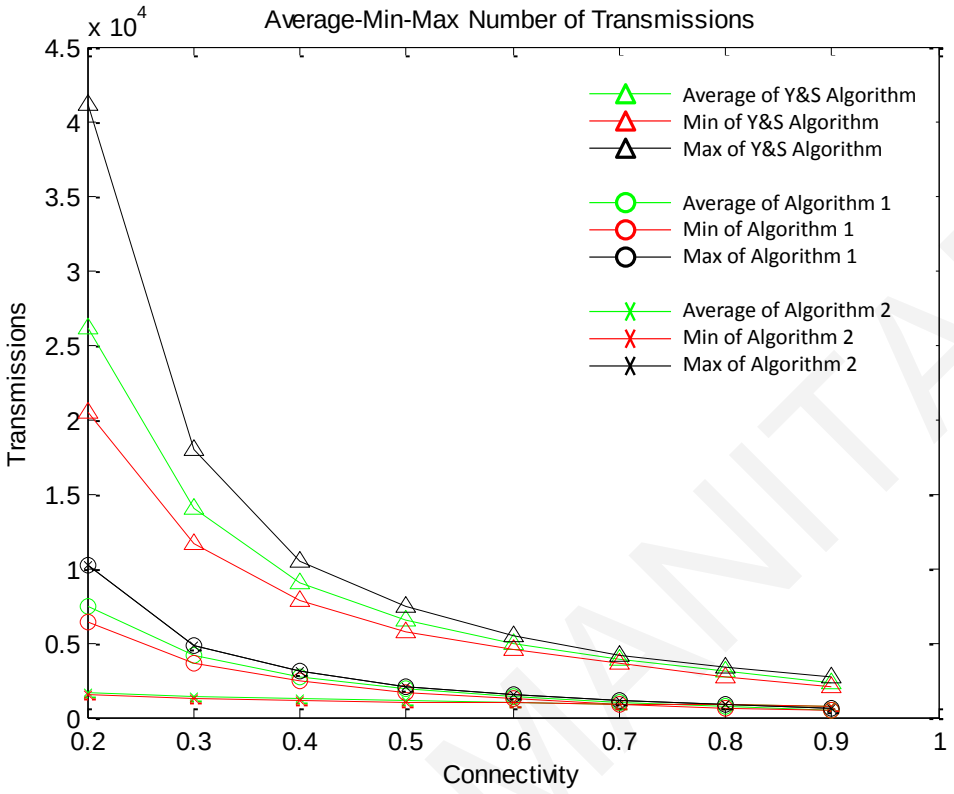

FIGURE 3.7: Average, maximum and minimum number of transmissions for Algorithm 1, Algorithm 2 and $Y\&S$ Algorithm.

3.4.3 Graph Connectivity

Finally, we present results based on the connectivity of the network, based on randomly generated undirected graphs with 100 nodes and probability of connection from 0.2 to 0.9 (i.e., for each pair of nodes $i, j \in X$, $i \neq j$, both edges $(i, j), (j, i) \in E$ with probability equal to $0.2 < pr < 0.9$). Nodes have initial values randomly chosen to be uniform in the interval $[0, 1]$, independently between each other. We consider 100 random graphs (with different instances of random initial conditions for each graph) and run both algorithms; we record the maximum, minimum, and average number of required steps and transmissions for reaching approximate average consensus in as shown in 3.7.

What it can be seen easily from Fig. 3.7 is that the less connected a network is, the more transmissions it requires for the nodes to reach agreement on the average value of the network. It is also observable from the plot that the Y&S Algorithm it requires more transmissions to converge to the mean value of the network, and mainly this is due to the fact that for every time step it requires three different values to be transmitted, instead in our case we only need 1 transmission for the single iterative scheme and 2 transmission for the double iterative scheme (ratio consensus). Although Algorithm 2

requires 2 transmissions at each time step we can see that convergence is faster than Algorithm 1, and this is due to the convergence rate that ratio consensus can give against single iterative scheme.

Chapter 4

DISTRIBUTED STOPPING FOR AVERAGE CONSENSUS IN DIGRAPHS

In this chapter we consider how the nodes can reach *approximate* average consensus in *finite* time while employing appropriately modified versions of the ratio consensus algorithm (thus, maintaining its advantages in terms of simplicity and reliance to minimal local information). Perhaps more importantly, the chapter investigates the topic of distributed stopping, i.e., how the nodes can determine when to terminate the execution of the algorithm based on locally available information.

Chapter 3 has studied this problem in a setting where the communication topology is described by an undirected graph; as we will argue in this chapter, the problem becomes significantly more challenging in the case of communication topologies that are described by directed graphs (digraphs), which is the focus of this chapter. We propose two event-triggered iterative strategies, one randomized and one deterministic, both of which depend on a parameter ε (small real number) that bounds, at the termination of the iterative process, the closeness of the final value of each node to the true average of their initial values. For the proposed event-triggered strategies, we prove that,

- All nodes eventually stop transmitting.
- The absolute difference of the final value of each node from the true average of the initial values is smaller than an error bound whose value depends on the parameter ε and the diameter D of the graph (in the case of the proposed randomized strategy, the above occurs with probability one).

The end results are randomized/deterministic protocols that not only avoid running the consensus algorithm indefinitely, but also allow each node to cease transmitting when its value is within a small distance from the values of its in-neighbors (e.g., in order to save

energy). In this way, the protocol allows each node to reduce the total number of values transmitted in the network before the nodes reach approximate agreement to the average of their initial values; specifically, nodes that are in agreement with their in-neighbors may choose not to update and/or transmit their value to their out-neighbors

The key observation in the proposed algorithm is that each node makes a decision regarding whether to transmit its value or not, primarily based on the difference between its calculated value and the values it receives from its in-neighbors. More specifically, each node calculates the pairwise absolute differences between its own value and the values of each of its in-neighbors, and takes action (such as stop communicating) depending on whether these differences are smaller than the parameter ε. In the probabilistic protocol, for example, a node that has stopped transmitting its value, may be triggered to start transmitting again with a given probability p if it receives a value from one or more of its neighbors. A node also resumes transmission if there are significant changes in the values received from its in-neighbors or in its own value since its last transmission. The iterative process ends when all nodes cease to transmit their values, in which case they can be shown to have reached *approximate* average consensus. It is worth pointing out that there are two proposed protocols, one randomized and one deterministic.

This chapter is organized as follows. In Section 4.1 we present previous work on distributed stopping for average consensus in directed graphs. In Section 4.2 we introduce the problem statement and related concepts of the chapter. In Section 4.3 we introduce our proposed strategy and main results of the chapter. In Section 4.4 we provide examples and simulation analysis. Finally in Section 4.5 we provide some discussion about our proposed algorithms against other algorithms and we compare their capabilities.

4.1 Previous Work on Distributed Stopping for Average Consensus in Digraphs

The method in [1] was described in Chapter 3 and can be applied to a given digraph, if the nodes have set of weights that forms a primitive doubly stochastic matrix P. The availability of such weights, however, is not immediate in directed graphs. The authors of [38] proposed a method which runs three double linear iterations in parallel (ratio consensus) 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* the set of weights p_{ij} forms a primitive column stochastic matrix P_c . The main idea of the work in [38], is that iterations can be used to check when the ratios of the nodes are sufficiently close using the min- and max-consensus iterations to allow nodes to determine the timestep when their ratios are within ϵ of each other. More specifically, each node i runs the ratio consensus iteration as in (1.6), in order

to allow the nodes to determine when the ratios r_i for all nodes are close to average value of the network. Each node maintains two additional state variables, $m_i[k]$ and $M_i[k]$, which are updated using min- and max-consensus respectively. These two state variables are compared every D iterations to check whether $|M_i[k] - m_i[k]| < \epsilon$; if this condition holds node i stops iterating, otherwise values $m_i[k]$ and $M_i[k]$ are initialized to $r_i[k] = y_i[k]/z_i[k]$. Note that all the nodes will simultaneously stop iterating since they will simultaneously learn that they have reached *approximate* agreement(which is easily shown in [38] to be close to the average value 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 ε of each other (the only difference is that the oracle might be delayed by at most $2D$ steps). Also note that, in order to implement the oracle, one needs extra values to be transmitted (to run the max/min-consensus algorithms). The pseudocode of this algorithm can be found below named as Algorithm 3.

Algorithm 3 Distributed Stopping for Average Consensus (Ratio)

Input: Each node $i \in E$ has initial value V_i and knows its out-degree $\mathcal{D}_i^+$. Initial values for the two iterations are set to $y_i[0] = V_i$ and $z_i[0] = 1$.
Set: $M_i[0] = +\infty$, $m_i[0] = -\infty$, $u_i[0] = 0$, $r_i = \frac{y_i[0]}{z_i[0]}$
Set: $p_{li} = \frac{1}{1+D_i^+}$, $\forall v_l \in \mathcal{N}_i^+ \cup \{v_i\}$ (zero otherwise)
For $k \geq 0$, for each node i, **do**
  while $u_i[k] = 0$ **do**
    if $k \mod D = 0$ and $k \neq 0$ **then**
      if $|M_i - m_i| < \epsilon$ **then**
        set $u_i[k] = 1$
      end if
      set $M_i[k] = m_i[k] = r_i[k] = \frac{y_i[k]}{z_i[k]}$
    end if
    Broadcast to all: $v_l \in \mathcal{N}_i^+$, $p_{li}[k]y_i[k]$, $p_{li}[k]z_i[k]$, $M_i[k]$, $m_i[k]$
    Receive from all: $v_j \in \mathcal{N}_i^-$ $p_{ij}[k]y_j[k]$, $p_{ij}[k]z_j[k]$, $M_j[k]$, $m_j[k]$
    Compute:
    $y_i[k] \leftarrow \sum_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} p_{ij}[k]y_j[k]$
    $z_i[k] \leftarrow \sum_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} p_{ij}[k]z_j[k]$
    $M_i[k] \leftarrow \max_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} M_j[k]$
    $m_i[k] \leftarrow \max_{v_j \in \mathcal{N}_i^- \cup \{v_i\}} m_j[k]$
  end while
End

4.2 Problem Statement and Related Concepts

We are interested in protocols that allow the nodes to reach *approximate* agreement to the average value of the network (i.e., agreement to the average within an error bound). At the same time, the nodes need to identify, *in a distributed manner* (only by passing

messages), when to stop transmitting their values (perhaps not all of them simultaneously) so that, at the termination of the iterative process, nodes reach approximate agreement on the average of their initial values. The absence of bidirectional communication links in a digraph makes this task particularly challenging since the decision must be taken according to the received values from the in-neighbors of each node, whereas new values are transmitted to the out-neighbors of each node. The specific setting we consider is described below.

All of this section assumes the following setting.

Setting: Consider a network described by a strongly connected digraph $G = \{X, E\}$, where $X = \{1, 2, ..., n\}$ is the set of components in the system and $E \subseteq X \times X - \{(i, i) \mid i \in X\}$ is the set of directed edges. Each node i has some initial value V_i and follows a distributed iterative algorithm for f steps during which it maintains and updates, at each iteration step k, a variable $r_i[k]$ (and perhaps other variables). With respect to the Definition 3.1 (see Chapter 3), we are interested in devising a distributed algorithm that allows the nodes to reach ε-approximate average consensus and also to identify (in a distributed manner) when such approximate average consensus has been reached.

Definition 4.1. *(ε-Approximate Local/Global Consensus and ε-Approximate Average Consensus [39])* Under the setting described above, we say that at the end of the iterative process

1. The nodes have reached ε-approximate *local* consensus if the final value of $r_i[f]$ of each node $i \in X$ satisfies $|r_i[f] - r_j[f]| \leq \varepsilon$, $\forall j \in \mathcal{N}_i^-$.
2. The nodes have reached ε-approximate *global* consensus if the final values of the nodes satisfy $|r_i[f] - r_j[f]| \leq \varepsilon$, $\forall i, j \in X$.
3. The nodes have reached ε-approximate *average* consensus if the final values of the nodes satisfy $|r_i[f] - \bar{v}| \leq \varepsilon$, $\forall i \in X$, where $\bar{v} = \frac{1}{n}\sum_{l=1}^{n} V_l$ is the average of the initial values. □

It can be shown, using similar techniques as the ones used in Chapter 3 that if, at the end of the iterative process, the nodes have reached ε-approximate *local* consensus, then the nodes have also reached (εD)-approximate global consensus, where D is the diameter of graph G. Note that if each node $i \in X$ uses a different ε_i, then one can show, using steps similar to the ones in the proof of the above proposition, that the nodes reach $(D \times \max_{i \in X}\{\varepsilon_i\})$-approximate average consensus.

Proof. Let $|r_i[f] - r_j[f]| \leq \varepsilon$ for all i, and all $j \in \mathcal{N}_i^-$ (local consensus). For any $i, j \in X$, $i \neq j$, let $i = i_0 \mapsto i_1 \mapsto \ldots \mapsto i_t = j$ be the *minimum* length path (of length t) that connects them (i.e., $(i_1, i_0), (i_2, i_1), ..., (i_t, i_{t-1}) \in E$ and no such path involving $t - 1$ or less edges exists). It follows from the definition of the graph diameter that $t \leq D$.

The difference $r_j[f] - r_i[f]$ can be written as the telescopic sum

$$\begin{aligned} r_j[f] - r_i[f] &= r_{i_t}[f] - r_{i_{t-1}}[f] + r_{i_{t-1}}[f] - r_{i_{t-2}}[f] + ... \\ &\quad ... + r_{i_1}[f] - r_{i_0}[f] \, . \end{aligned}$$

Using the triangle inequality, we have

$$|r_i[f] - r_j[f]| \leq \sum_{l=1}^{t} |r_{i_l}[f] - r_{i_{l-1}}[f]| \leq t \times \varepsilon \leq D \times \varepsilon \, .$$

This completes the proof of the proposition. □

4.3 Proposed Strategy and Main Results

Consider the setting in Section 4.2. We propose two schemes, one randomized and one deterministic, which make use of a time-varying version of the double linear iterative scheme in (1.4)–(1.5). In both proposed schemes (referred to as Algorithm 4 and Algorithm 5), in order to enable the nodes to identify the correct time step to stop transmitting their values, the weights p_{ij} are allowed to be time-varying. At each time step k, the nodes determine a subset of "active" edges $E[k] \subseteq E$ that forms a directed (but not necessarily strongly connected) graph $G[k] = \{X, E[k]\}$; then, the weights are chosen in order to ensure that weight matrix $P[k] = [p_{ij}[k]]$ that describes the set of weights at iteration k is column stochastic (but not necessarily primitive, that will depend on whether the graph $G[k]$ is strongly connected or not).

Using the ideas from Chapter 3, one is tempted to run an event-triggered version of the ratio consensus algorithm described in Chapter 1. The key constraint, however, is that in a digraph setting nodes do not have bidirectional capability; thus, the decision of node i to transmit has to rely solely on the values of its in-neighbors $\mathcal{N}_i^-$, whereas the values that are transmitted will be received by its out-neighbors $\mathcal{N}_i^+$. In fact, node i will have to transmit to *all* of its out-neighbors in $\mathcal{N}_i^+$ because its out-neighboring nodes cannot easily determine (via value comparisons or other means) whether certain links are inactive or not, i.e., whether a value is intended for them or not. Note that, just like regular ratio consensus in (1.4)–(1.5), each node i needs to be aware of its out-degree $\mathcal{D}_i^+$.

Following the above line of thought, one possible strategy would be the following: in order to determine the active edges at each time step k, each node i compares its ratio $r_i[k]$ against the ratio $r_j[k]$ of each *in-neighbor* $j \in \mathcal{N}_i^-$; if the absolute difference $|r_i[k] - r_j[k]| > \varepsilon$ for at *least one* $j \in \mathcal{N}_i^-$, node i transmits its values to *all of its out-neighbors*; if the absolute differences $|r_i[k] - r_j[k]| \leq \varepsilon$, $\forall j \in \mathcal{N}_i^-$, node i does not transmit its values. Since some of the in-neighbors of node i might seize transmission

at some time step, the above rule needs to be implemented by comparing the last seen ratios of each neighbor (i.e., the ratios they had the last time they transmitted values). Note that the last seen value of node j will be the same from the perspective of all of its out-neighbors; thus, we use $lastseen_j[k]$ to denote the last transmitted ratio of node j by time step k (in reality, $lastseen_j$ is not a global variable but rather a local variable that is identical to all out-neighbors of node j). Using this notation, Event-Triggered Rule 1 becomes as described below. [Note that node i also checks the value $|r_i[k] - lastseen_i[k]|$ because its own ratio might change while it is not transmitting (due to the fact that it might be receiving values).]

Event-Triggered Rule 1: In order to determine whether to transmit or not at each time step k, each node i compares its ratio $r_i[k]$ against $lastseen_j[k]$ for each *in-neighbor* (and itself) $j \in \mathcal{N}_i^- \cup \{i\}$; if the absolute difference $|r_i[k] - lastseen_j[k]| > \varepsilon$ for at *least one* $j \in \mathcal{N}_i^- \cup \{i\}$, node i transmits its values ($y_i[k]$ and $z_i[k]$) to *all of its out-neighbors*; otherwise, node i does not transmit its values. □

The above described rule naturally leads to the (time-varying) event-driven version of the iterations in (1.4)–(1.5) described in Algorithm 4 (ignore the text between the dotted horizontal lines for now). The algorithm is written from the perspective of node i. Note that $flag_i$ is a binary variable used by node i to determine if it should be transmitting or non-transmitting at the next iteration. Also $\mathcal{N}_i^-[k]$ is the set of in-neighbors of node i that are transmitting at time step k, i.e., $\mathcal{N}_i^-[k] \subseteq \mathcal{N}_i^-$ and is known to node i based on the transmissions that it receives at time step k.

The following example illustrates that Event-Triggered Rule 1 on its own is insufficient, at least for certain types of digraphs and certain sets of initial values.

Example 4.1. *Consider a ring digraph with four nodes and diameter $D = 3$ as shown in Fig. 4.1 (where $X = \{1, 2, 3, 4\}$ and $E = \{(2, 1), (3, 2), (4, 3), (1, 4)\}$). Suppose that we execute an event-driven version of the ratio consensus algorithm where Event-Triggered Rule 1 is used as described in Algorithm 4 (excluding the code between the two dotted horizontal lines). Assume that a transmitting (active) node uses equal weights on its outgoing edges (including its self-weight). Depending on which of the four nodes are transmitting (due to the fact that their ratios disagree with the ratio of at least one of their in-neighbors), we have 2^4 different possible weight matrices. For example, if nodes 2 and 4 are transmitting at iteration k, the matrix $P[k]$ takes the form*

$$P[k] = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 1 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}. \tag{4.1}$$

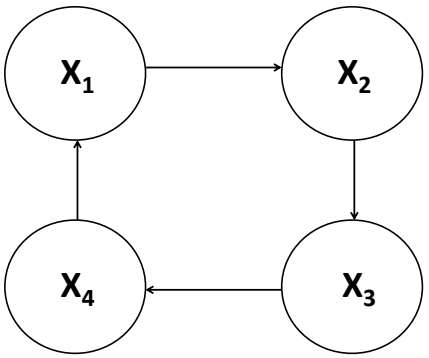

FIGURE 4.1: Digraph with four nodes and diameter $D = 3$.

Assume initial conditions $y[0] = [0.1,\ 0.3,\ 0.3,\ 0.1]^T$ *and* $z[0] = [1,\ 1,\ 1,\ 1]^T$, $\varepsilon = 0.0001$, *and consider the use of Event-Triggered Rule 1. It can easily be seen that the absolute ratio of differences* $|r_2[0] - r_1[0]| = 0.2$ *and* $|r_4[0] - r_3[0]| = 0.2$, *which implies that nodes* 2 *and* 4 *become transmitting at the first iteration as shown in Fig. 4.2; however, nodes* 1 *and* 3 *will be non-transmitting.*
If we follow through the execution of the protocol at subsequent iterations, we see that nodes 2 *and* 4 *will be always transmitting, and nodes* 1 *and* 3 *will be always non-transmitting (inactive). As a result, the state variables of nodes* 2 *and* 4 *eventually satisfy* $\lim_{k\to\infty} y_l[k] = 0$ *and* $\lim_{k\to\infty} z_l[k] = 0$ *for* $l = 2, 4$. *This can also be seen in the plots of Fig. 4.3 that simulate the execution of Algorithm 2 (using only Event-Triggered Rule 1). The key problem is that nodes* 2 *and* 4 *are aware of the discrepancies, but nodes* 1 *and* 3 *are not; thus, the former nodes keep sending values whereas the latter nodes keep receiving.*

To overcome the problem exhibited in the above example, we propose below two alternatives for an additional Event-Triggered Rule 2 that will be used to complement Event-Triggered Rule 1. The first alternative is a randomized strategy (called Event-Triggered Rule 2a) and the second strategy is a deterministic one (called Event-Triggered Rule 2b). We will establish that when either of these rules is used in combination with Event-Triggered Rule 1, the nodes (in any given strongly connected digraph, under any initial conditions) will be able to reach average consensus and identify (in a distributed manner) when to terminate their operation (as shown in Fig. 4.4), once they have reached ε-approximate average consensus. In the case of the randomized strategy, the above occurs with probability one.
A full description of Event-Triggered Rule 2a (randomized) and Event-Triggered Rule 2b (deterministic) can be found below.

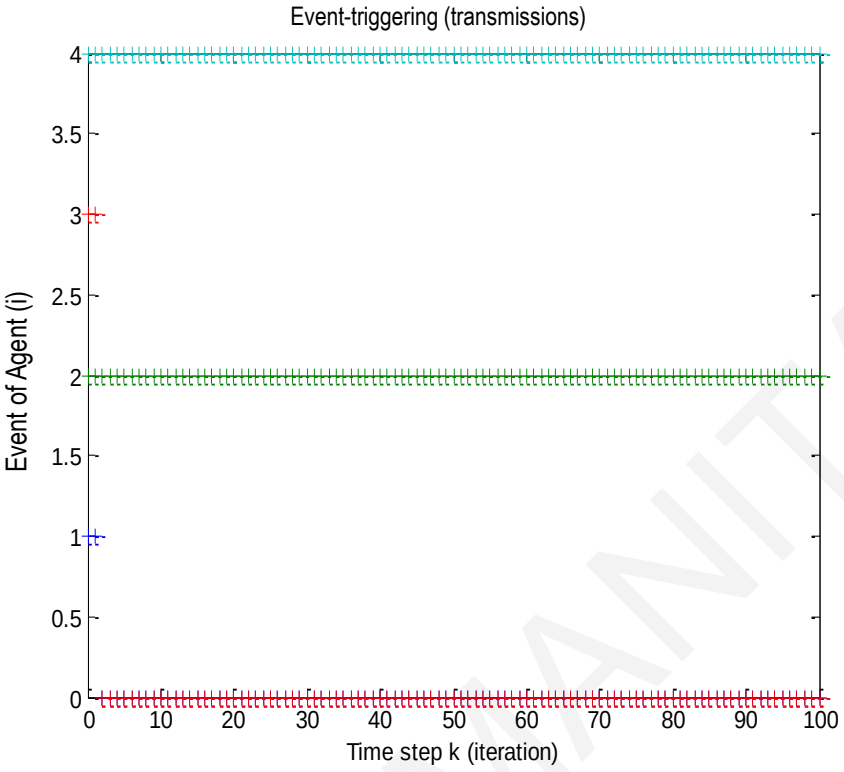

FIGURE 4.2: Event-triggerings (transmissions) of node 2 and node 4 when using Algorithm 4, utilizing only Event-Triggered Rule 1.

4.3.1 Randomized Event-Triggered Strategy

Event-Triggered Rule 2a: At each time step k, even if all values seen at node i satisfy $|r_i[k] - lastseen_j[k]| \leq \varepsilon$ for all $j \in \mathcal{N}_i^- \cup \{i\}$, if node i receives a value from at least one in-neighbor, then node i becomes transmitting with some probability p_i (where $0 < p_i < 1$), independently from triggerings from other nodes (including previous own triggerings). □

A formal description of Algorithm 4 (which incorporates both Event-Triggered Rules 1 and 2a) can be found below.

Theorem 4.2. *Consider the problem setting in Section 4.2. Each node i has some initial value V_i and runs Algorithm 4. Then, with probability one, all nodes become non-transmitting after a finite number of steps f, at which point the ratios satisfy*

$$|r_i[f] - \bar{v}| \leq 2\varepsilon D, \ \forall i \in X \ ,$$

where $\bar{v} = \frac{1}{n}\sum_{l=1}^{n} V_l$ and D is the diameter of graph G.

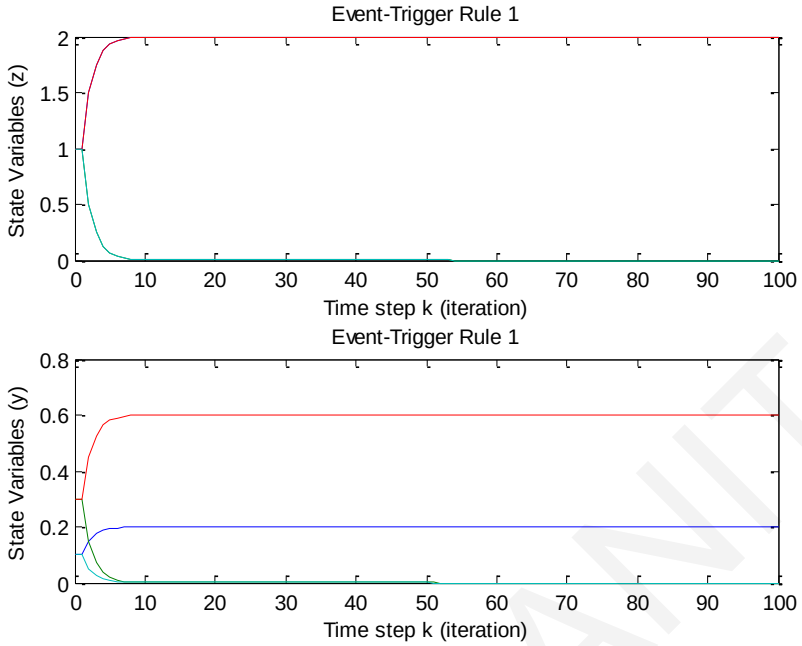

FIGURE 4.3: Values of state variables $y_i[k]$ and $z_i[k]$ $(i = 1(sky\ blue), 2(red), 3(green), 4(blue))$ when using Algorithm 4, utilizing only Event-Triggered Rule 1.

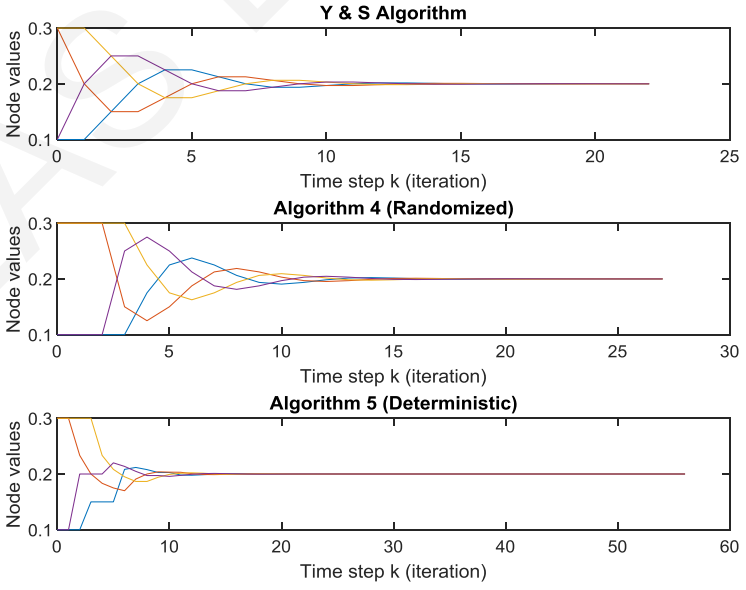

FIGURE 4.4: Evolution of node values for Y & S Algorithm, Algorithm 4 and Algorithm 5, with $\varepsilon = 0.0001$.

Algorithm 4: Distributed Stopping for Average Consensus in Digraphs Using Randomized Event-Triggered Strategy

Input: Each node i initializes $y_i[0] = V_i$ and $z_i[0] = 1$, sets $p_{li}[0] = 1/(1+\mathcal{D}_i^+)$, $\forall l \in \mathcal{N}_i^+ \cup \{i\}$ and $lastseen_j[-1] = \infty$, $\forall j \in \mathcal{N}_i^- \cup \{i\}$, and does the following:

For $k \geq 0$, for each node i,

Set: $lastseen_j[k] = lastseen_j[k-1]$, $\forall j \in \mathcal{N}_i^- \cup \{i\}$

Broadcast:

If $P_{ii}[k] \neq 1$

Send $\bar{y}_i[k] = p_{li}[k]y_i[k]$ and $\bar{z}_i[k] = p_{li}[k]z_i[k]$ to all $l \in \mathcal{N}_i^+$

$lastseen_i[k] = \frac{\bar{y}_i[k]}{\bar{z}_i[k]} = \frac{y_i[k]}{z_i[k]}$

End

Receive: $\bar{y}_j[k]$ and $\bar{z}_j[k]$ from all $j \in \mathcal{N}_i^-[k]$

($\mathcal{N}_i^-[k]$ is the set of transmitting (active) in-neighbors of node i at time step k)

Update Received Values $\forall j \in \mathcal{N}_i^-[k]$, $r_j[k] = \bar{y}_j[k]/\bar{z}_j[k]$, $lastseen_j[k] = r_j[k]$

Compute:

$y_i[k+1] = p_{ii}[k]y_i[k] + \sum_{j\in\mathcal{N}_i^-[k]} \bar{y}_j[k]$

$z_i[k+1] = p_{ii}[k]z_i[k] + \sum_{j\in\mathcal{N}_i^-[k]} \bar{z}_j[k]$

$r_i[k+1] = y_i[k+1]/z_i[k+1]$

$flag_i = 0$

For $j \in \mathcal{N}_i^- \cup \{i\}$:

$check(i,j) = |r_i[k] - lastseen_j[k]|$

If $check(i,j) \geq \varepsilon$ **then**

$flag_i = 1$

End

End

If $\mathcal{N}_i^-[k] \neq \emptyset$ **then**

If $rand() < p_i$ **then**

$flag_i = 1$

End

End

If $flag_i = 0$ then

$P_{ii}[k] = 1 (non-transmittingnode)$

$P_{li}[k] = 0, \quad \forall l \in \mathcal{N}_i^+$

else

$P_{ii}[k] = 1/(1+\mathcal{D}_i^+)$ (transmitting node)

$P_{li}[k] = 1/(1+\mathcal{D}_i^+), \quad \forall l \in \mathcal{N}_i^+$

End

End

Remark 4.3. Theorem 1 states that nodes following Algorithm 4 reach $(2D\times\varepsilon)-$approximate average consensus in finite time and the final weight matrix is an identity matrix (all nodes in the network are absorbing nodes).

Proof. We first establish, by contradiction, that the nodes will reach (2ε)-approximate local consensus after a finite number of iterations f. Suppose that the iteration runs forever; this means that at each iteration k at least one node is transmitting. This implies that at least one node (say node i) is transmitting infinitely often. This means that all of node i's neighbors are also transmitting infinitely often (because each $j \in \mathcal{N}_i^+$ receives packages from node i infinitely often, and $p_j > 0$). Therefore, since the graph is strongly connected, we conclude that all nodes are transmitting infinitely often. Thus, we are running ratio consensus with time-varying column stochastic matrices $P[k]$ as in (1.7)–(1.8), where $y_i[0] = V_i$, $z_i[0] = 1$, for all $i \in X$, and at each iteration the column stochastic matrix $P[k]$ is determined by the nodes that are transmitting at time step k. One sufficient condition for convergence of the ratio $r_i[k] = y_i[k]/z_i[k]$ to the same value is the existence of a finite window K such that the matrix products

$$P_m \equiv P[mK + K - 1] \dots P[mK + 1]P[mK], \quad m = 0, 1, 2, ...,$$

form primitive column stochastic matrices [22]. In turn, a sufficient condition for that would be for the *union graphs* $\{X, \cup_{k=mK}^{mK+K-1} E[k]\}$, $m = 0, 1, 2, ...$, to be strongly connected (where $\{X, E[k]\}$ is the graph that corresponds to the zero/one structure of matrix $P[k]$). Clearly, since the nodes are transmitting infinitely often, we can find a finite window of length l, such that the products of l matrices satisfy

$$P[l]P[l-1] \dots P[1]P[0] \approx C_1 \mathbf{1}^T ,$$

where C_l is a strictly positive column vector (that may be changing with l), 1^T is the all ones row vector, and the approximation can be made arbitrarily tight by increasing l. In other words, $P[l]P[l-1]...P[0]$ asymptotically becomes a matrix with columns that are approximately the same. It follows that the nodes asymptotically reach the same ratio, because $y[l] \approx C_l(1^T y[0])$ and $z[l] \approx C_l(1^T z[0])$, so that

$$r_i[l] = \frac{y_i[l]}{z_i[l]} \to \frac{\sum y_i[0]}{\sum z_i[0]} = \bar{x}, \quad \forall i \in X .$$

Clearly, we can choose a large enough l, so that, for each node i, we have $|r_i[l] - \bar{x}| < \frac{\varepsilon}{2}$ and (since each node transmits infinitely often) $|lastseen_i[l] - \bar{v}| < \frac{\varepsilon}{2}$. The above two inequalities would imply that, for each node i, we have $|r_i[l] - lastseen_j[l]| < \varepsilon$ for all nodes i and j (in particular, for all $j \in \mathcal{N}_i^-$).

Effectively, the above discussion establishes that the nodes will eventually reach ε-approximate local consensus on their ratios.

When the above holds (i.e., when we have $|r_i[l] - lastseen_j[l]| < \varepsilon$ for all nodes i and all $j \in \mathcal{N}_i^-$), we see that in Algorithm 4, each node i can only become transmitting with probability p_i $(0 < p_i < 1)$, *as long as it receives at least one packet*. However, since p_i is strictly less than unity, we know that with probability 1 the nodes will eventually stop transmitting[1]. Since nodes will stop transmitting with probability one, we have reached a contradiction and we know that there exists a finite time f when all nodes stop transmitting.

When all nodes stop transmitting, the following hold true for any node i and node $j \in \mathcal{N}_i^-$: $|r_i[f] - lastseen_j[f]| \leq \varepsilon$ and $|r_j[f] - lastseen_j[f]| \leq \varepsilon$, where the first inequality is guaranteed by the checks performed by node i and the second inequality is guaranteed by the checks performed by node j. Combining the above, we get $|r_i[f] - r_j[f]| \leq 2\varepsilon$, and using the fact that ϵ-approximate local consensus implies $(D\epsilon)$-global consensus [39], we obtain that, at time step f, the absolute value of the difference between the ratios of any two nodes (separated by a distance of at most D) is bounded by $2\varepsilon D$.

Next we establish that, at time step f, the nodes have also reached $(2\varepsilon D)$-approximate *average* consensus. Let $r_{\min}[f] = \min_{i \in X} r_i[f]$ and $r_{\max}[f] = \max_{i \in X} r_i[f]$; we have $r_{\max}[f] - r_{\min}[f] \leq 2\varepsilon D$. Since $r_i[f] = y_i[f]/z_i[f]$, we have $r_{\min}[f] \leq r_i[f] \leq r_{\max}[f]$ or $r_{\min}[f] z_i[f] \leq y_i[f] \leq r_{\max}[f] z_i[f]$ for all $i \in X$. Summing all n of the latter inequalities, we obtain

$$r_{\min}[f] \sum_{i=1}^{n} z_i[f] \leq \sum y_i[f] \leq r_{\max}[f] \sum_{i=1}^{n} z_i[f],$$

which implies (since $\sum_{i=1}^{n} z_i[f] = n$) that $n r_{\min}[f] \leq \sum y_i[f] \leq n r_{\max}[f]$, or

$$r_{\min}[f] \leq \bar{x} \leq r_{\max}[f] \,. \tag{4.2}$$

It is easy to see that, at time step f (when the nodes stop transmitting), the nodes have reached $(2\varepsilon D)$-approximate average consensus. □

Remark 4.4. The conclusion that $|r_i[f] - \bar{x}| \leq 2\varepsilon D$, $\forall i \in X$ in Theorem 4.2 follows from the fact that at time step f the following will necessarily hold: (i) the nodes are guaranteed to have reached (2ϵ)-approximate local consensus and (ii) the diameter of the graph is D (thus, (2ϵ)-approximate local consensus implies $(2D\epsilon)$-approximate global consensus). It is possible that both of the above could be improved, e.g., by studying the worst case ordering with which nodes might stop transmitting or by looking at the

[1] The simplest way to see this intuitively is to realize that there is a nonzero probability of at least $(1 - p_1)(1 - p_2)...(1 - p_n)$ with which all nodes decide not to transmit, at which point they will all stop transmitting. In fact, the number of time steps needed for all nodes to stop transmitting can be described via an absorbing Markov chain (an example is discussed later).

longest simple cycle in the given digraph (for example, in a directed ring of n nodes, the longest simple cycle has length $n-1$ and leads to a bound of $2\varepsilon\lceil\frac{n-1}{2}\rceil$, which is approximately εD instead of $2\varepsilon D$).

Remark 4.5. Note that during the execution of Algorithm 4, once the nodes reach approximate local consensus, they do not stop immediately. To obtain an understanding of how quickly nodes stop transmitting altogether, we can analyze the rate at which an appropriately constructed Markov chain reaches its only absorbing state. This rate depends on the probabilities $\{p_i \mid i \in V\}$ but also on the digraph structure. As an example, consider the digraph shown in Fig. 4.5, and assume that all nodes are using Event-Triggered Rule 2a, and get activated with the same probability $p_i = p$ when they receive at least one value from a neighboring node. The finite state Markov chain in Fig. 4.6 describes the behavior of the randomized protocol once the nodes have reached ϵ-approximate local consensus on their ratios, *assumming the last node that transmitted a value is node 4*. Each state has as a label a subset of {1, 2, 3, 4}, capturing which nodes in the graph are transmitting. For example, if node 4 is transmitting, then nodes 1 and 2 receive this value and may decide to also transmit (independently with probability p). Thus, from the state labeled $\{4\}$ we reach state $\{1\}$ with probability $p(1-p)$, state $\{2\}$ with probability $p(1-p)$, state $\{1,2\}$ with probability p^2, and state $\oslash$ with probability $(1-p)^2$. Notice that the Markov chain in Fig. 4.6 has a single absorbing state $\oslash$ which is reached with probability one. This will be the case in general, since from any other state there will always be a nonzero probability of reaching state $\oslash$. In fact, the Markov chain can also be used to determine the rate with which the absorbing state is reached; this rate is directly related to the expected time it will take for the nodes to stop transmitting once they reach approximate average consensus.

4.3.2 Deterministic Event-Triggered Strategy

The idea in Algorithm 5, is for a node that receives b_i transmissions (from at least one in-neighbor) to eventually react (by transmitting for the next c_i consecutive time steps). Here b_i and c_i are positive integers that are used to ensure that situations like the ones appearing in Example 4.1 are avoided. Node i maintains two counters, a counter-to-start $CStart_i$ that counts up to value b_i, and a counter-to-stop $CStop_i$ that counts down to zero starting from c_i. Event-Triggered Rule 2b takes the form below.

At each time step k, node i compares its value against all the values of its in-neighbors $j \in \mathcal{N}_i^-$. When all received values satisfy

$$|r_i[k] - lastseen_j[k]| \leq \varepsilon, \qquad \forall j \in \mathcal{N}_i^- \cup \{i\},$$

counter $CStart_i$ increases by one each time node i receives values from at least one of its in-neighbors (in case the above condition *is not* satisfied, then $CStart_i$ gets reset

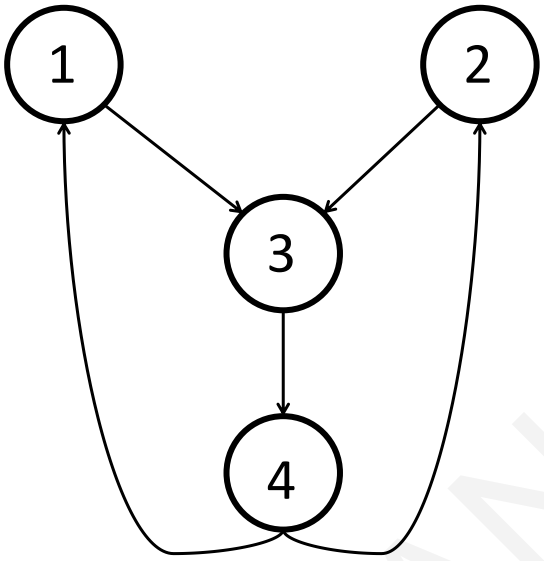

FIGURE 4.5: Digraph discussed in Remark 2.

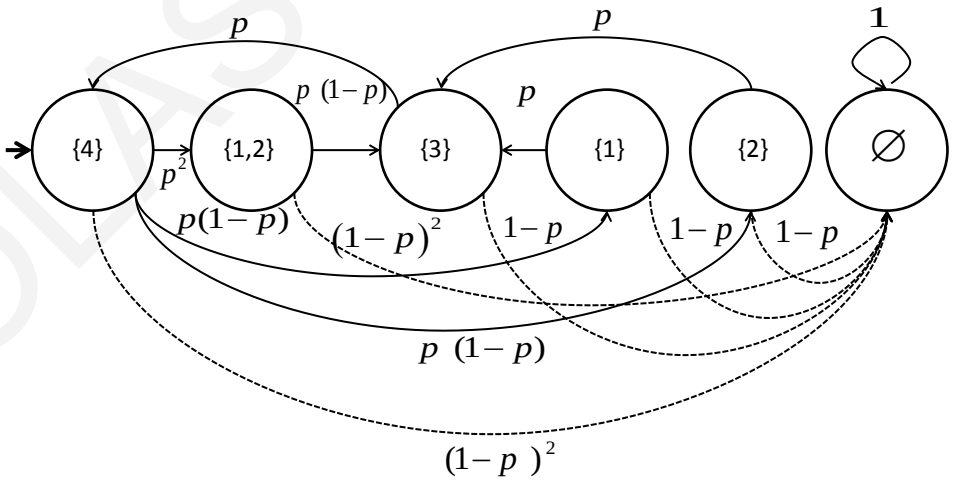

FIGURE 4.6: Markov chain describing the probabilities with which different subsets of transmitting nodes may cause subsequent transmissions (dotted lines indicate transmissions to the absorbing state ⊘).

to zero again and node i transmits). When counter $CStart_i$ becomes equal to a fixed positive number b_i, then node i becomes transmitting for the next c_i time steps (where c_i is a positive constant). In summary we have the following event-triggered rule.

Event-Triggered Rule 2b: At each time step k, even if all values seen at node i satisfy $|r_i[k]-lastseen_j[k]| \leq \varepsilon$ for all $j \in \mathcal{N}_i^- \cup \{i\}$, if node i receives a value from at least one in-neighbor, then node i increases its counter $CStart_i$ by one (in case $|r_i[k]-lastseen_j[k]| > \varepsilon$ *is not* satisfied for at least one $j \in \mathcal{N}_i^- \cup \{i\}$, then $CStart_i$ gets reset to zero). When counter $CStart_i$ becomes equal to a fixed positive number b_i, then node i becomes transmitting for the next c_i time steps.

Theorem 4.6. *Consider the problem statement at the end of Section 4.2. Each node i has some initial value V_i and runs Algorithm 5. If the following (sufficient) condition holds*

$$\mathcal{D}_i^+ c_i < b_i, \forall i \in X \ , \tag{4.3}$$

all nodes will stop transmitting in finite time. Furthermore, when the nodes stop transmitting, they can be shown to have reached $(2\varepsilon D)$-approximate average consensus, i.e., their ratios satisfy $|r_i[f] - \bar{v}| \leq 2\varepsilon D$, for all $i \in X$, where $\bar{v} = \frac{1}{n}\sum_{l=1}^{n} V_l$.

Proof. The first part of the proof is similar to the one in the proof of Theorem 4.2 for Algorithm 5. In other words, we can use a contradiction argument to establish that the nodes will reach (2ε)-approximate local consensus (and thus $(2\varepsilon D)$-approximate global consensus and $(2\varepsilon$D)-approximate average consensus) after a finite number of iterations, say k_0. The key observation is that if a node is transmitting infinitely often then all nodes will also be transmitting infinitely often, because of the use of counters and the fact that the digraph is strongly connected.

Suppose that the nodes do not stop transmitting after they reach (2ε)-approximate local consensus on their ratios. As argued above, this means that all nodes are transmitting infinitely often. Let time step k_i^1 denote the first time (after k_0 steps) when the counter of node i takes the value b_i, causing node i to start transmitting for the next c_i iterations (note that these times are not the same for all nodes). Similarly, let k_i^2 be the next time step at which the counter of node i reaches the value b_i, and so forth. In Fig. 4.7 we use $K_1 = [k_1^1\ k_2^1\ ...\ k_n^1]^T$, $K_2 = [k_1^2\ k_2^2\ ...\ k_n^2]^T$, and so forth, to denote the time indices of firings at different nodes.

The total number of packets that are sent on outgoing links of the nodes that have triggered between K_1 and K_2 is bounded from above by $\sum_i \mathcal{D}_i^+ c_i$ (since node i transmits to all of its out-neighbors for c_i consecutive time steps). Moreover, in order to have the next triggerings at the time instances captured by vector K_2 we need at least $\sum_i b_i$ triggerings (each node i needs to receive at least b_i packets on its incoming links). However, the condition in (4.3) implies that $\sum_i \mathcal{D}_i^+ c_i < \sum_i b_i$. Thus, we have reached

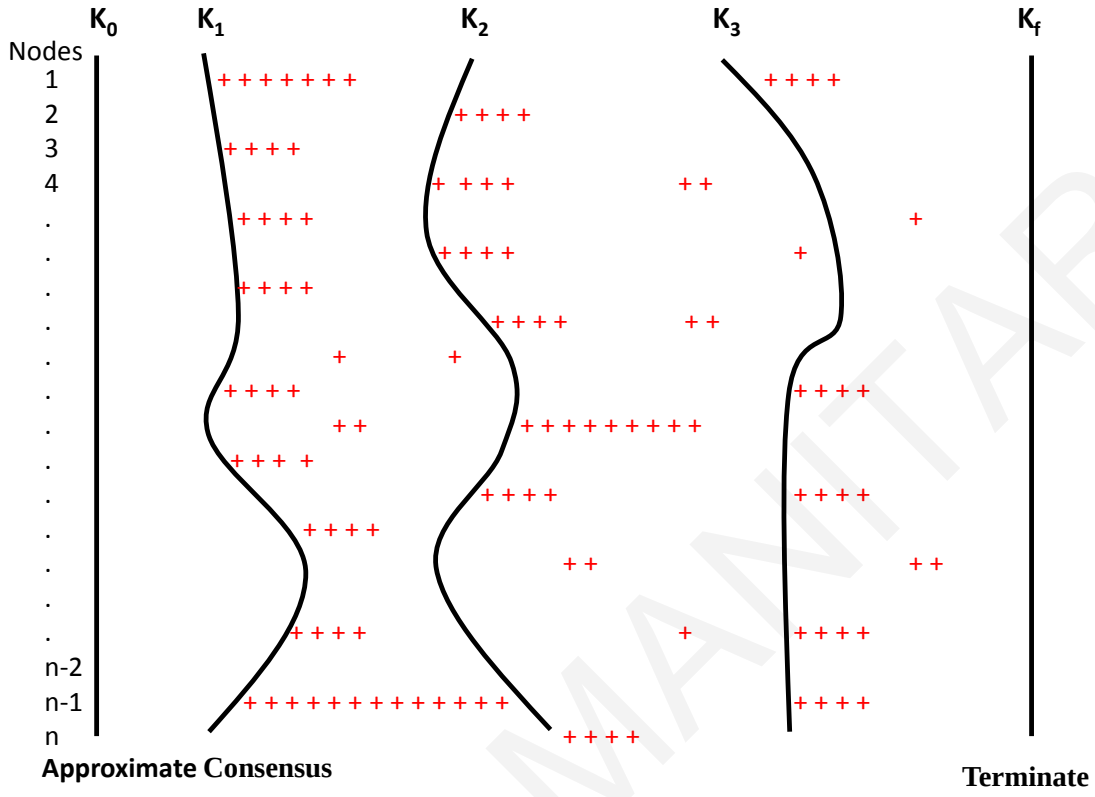

FIGURE 4.7: Time steps showing the triggering of nodes due to Event-Triggered Rule 2b.

a contradiction, and we conclude that there will be a time step f, after which all nodes seize transmitting.

In conclusion, if the condition in (4.3) is satisfied, then we are guaranteed that the nodes will stop transmitting after a finite number of iterations f; furthermore, following identical steps as in the proof of Theorem 4.2, we can show that at this point the nodes have reached (2ε)-approximate local consensus, $(2\varepsilon D)$-approximate global consensus, and $(2\varepsilon D)$-approximate average consensus. □

Remark 4.7. Since we have a time-varying linear iteration in both Algorithms 4 and 5, the question of bounding the number of steps the algorithms require to complete is not an easy one. In ratio consensus, rates of convergence can be obtained by finding finite windows over which the switching matrices are such that the underlying union graph is strongly connected, and then studying the rates of contraction of appropriate coefficients of ergodicity [19, 22]. However, in our case, we have neither convergence nor switching strategies that guarantee the existence of the above types of finite windows. In fact, the switching strategy in the case of Algorithm 5 is entirely deterministic given the initial conditions and it is possible that it would never lead to windows with such

properties. Nevertheless, as mentioned earlier for the case of Algorithm 4, one can focus on characterizing the number of iterations that are needed *after* the nodes have reached approximate average consensus. When nodes execute Algorithm 5 and reach approximate average consensus, they will require at most $n \times c_{\max} \times b_{\max}$ additional iterations (where $c_{max} = \max\{c_i\}$ and $b_{max} = \max\{b_i\}$) before they stop transmitting. To see this, refer to Fig. 4.7 and realize that at the time approximate average consensus is reached, the total number of subsequent transmissions in the next $c_{\max}$ iterations (by all nodes) is bounded by $n \times c_{\max}$ (i.e., when each node i is in a situation where it transmits c_i consecutive times); then, after at most $b_{\max}$ iterations we are guaranteed that the number of transmissions (in the next $c_{\max}$ iterations) goes down by at least one due to the fact that the condition in (4.3) holds.

Remark 4.8. As mentioned earlier, during the execution of Algorithm 4 and Algorithm 5, it is possible for some nodes in the network to stop transmitting and later restart transmitting. This process (of ceasing to transmit and then restarting transmissions) may be repeated until all nodes in the network cease to transmit, in which case the weight matrix $P[f]$ becomes the identity matrix. When this happens, it can be shown that the network has reached approximate agreement on the average of the initial values of the nodes.

4.4 Examples and Simulation Studies

Small Graph

We compare the proposed probabilistic and deterministic algorithms (Algorithm 4 and Algorithm 5) against the Y&S algorithm in [1], by carrying out simulations on a ring digraph as in Fig. 4.8 with eight nodes and diameter $D = 7$, using initial values $x[0] = [0.3, 0.3, 0.3, 1, 1, 1, 1, 0.3]^T$. Fig. 4.9 shows, for each algorithm, the evolution of the node values until termination. [We use different ε values for the algorithms to ensure that the resulting approximation of the average in all three cases is of the same quality.] As expected, both Algorithms 4 and 5 reach approximate average consensus in a finite number of iterations. As pointed out earlier, the approach in [1] needs to have access to a set of weights that forms a primitive doubly stochastic matrix.

Among the three algorithms, the Y&S algorithm in [1] requires the least number of iterations to complete, whereas Algorithms 4 and 5 require more number of iterations; however, Algorithms 4 and 5 require significantly less transmissions (see Table 4.1). Note that the way we calculate the number of transmissions is according to the number of values that are required for each algorithm: for Algorithm 4 and Algorithm 5 two transmissions are required by each *transmitting* node at each time-step (in order to send the iteration values $y_i[k]$ and $z_i[k]$), whereas for the Y&S algorithm, three transmissions are

Algorithm 5: Distributed Stopping for Average Consensus in Digraphs Using the Deterministic Event-Triggered Strategy (Rule 2b)

Input: Each node i initializes $y_i[0] = V_i$ and $z_i[0] = 1$, and sets $p_{li}[0] = 1/(1 + \mathcal{D}_i^+)$, $\forall l \in \mathcal{N}_i^+ \cup \{i\}$, and $lastseen_j[-1] = \infty$, $\forall j \in \mathcal{N}_i^- \cup \{i\}$. Each node selects b_i and c_i such that $b_i > \mathcal{D}_i^+ c_i$, sets $CStop_i = 0$, $CStart_i = 0$, and does the following:

For $k \geq 0$, for each node i,

Set: $lastseen_j[k] = lastseen_j[k-1]$, $\forall j \in \mathcal{N}_i^- \cup \{i\}$

Broadcast:

If $P_{ii}[k] \neq 1$

Send $\bar{y}_i[k] = p_{li}[k]y_i[k]$ and $\bar{z}_i[k] = p_{li}[k]z_i[k]$ to all $l \in \mathcal{N}_i^+$

$lastseen_i[k] = \frac{\bar{y}_i[k]}{\bar{z}_i[k]} = \frac{y_i[k]}{z_i[k]}$

End

Receive: $\bar{y}_j[k]$ and $\bar{z}_j[k]$ from all $j \in \mathcal{N}_i^-[k]$

($\mathcal{N}_i^-[k]$ is the set of transmitting (active) in-neighbors of node i at time step k)

Update Received Ratios: $\forall j \in \mathcal{N}_i^-[k]$, $r_j[k] = \bar{y}_j[k]/\bar{z}_j[k]$, $lastseen_j[k] = r_j[k]$

Compute:

$y_i[k+1] = p_{ii}[k]y_i[k] + \sum_{j \in \mathcal{N}_i^-[k]} \bar{y}_j[k]$

$z_i[k+1] = p_{ii}[k]z_i[k] + \sum_{j \in \mathcal{N}_i^-[k]} \bar{z}_j[k]$

$r_i[k+1] = y_i[k+1]/z_i[k+1]$

$flag_i = 0$

For $j \in \mathcal{N}_i^- \cup \{i\}$:

$check(i,j) = |r_i[k] - lastseen_j[k]|$

If $check(i,j) \geq \varepsilon$ **then**

$flag_i = 1$, $CStart_i = 0$, $CStop_i = 0$

End

End

If $\mathcal{N}_i^-[k] \neq \emptyset$ and $flag_i = 0$ **then**

$CStart_i = CStart_i + 1$

End

If $CStart > b_i$ **then**

$flag_i = 1,\ CStart_i = 0,\ CStop_i = c_i$

End

If $0 < CStop_i < c_i$ **then**

$flag_i = 1,\ CStart_i = 0,\ CStop_i = CStop_i - 1$

End

If $flag_i = 0$ then

$P_{ii}[k] = 1$ (non-transmitting node)

$P_{li}[k] = 0, \quad \forall l \in \mathcal{N}_i^+$

else

$P_{ii}[k] = 1/(1 + \mathcal{D}_i^+)$ (transmitting node)

$P_{li}[k] = 1/(1 + \mathcal{D}_i^+), \quad \forall l \in \mathcal{N}_i^+$

End

End

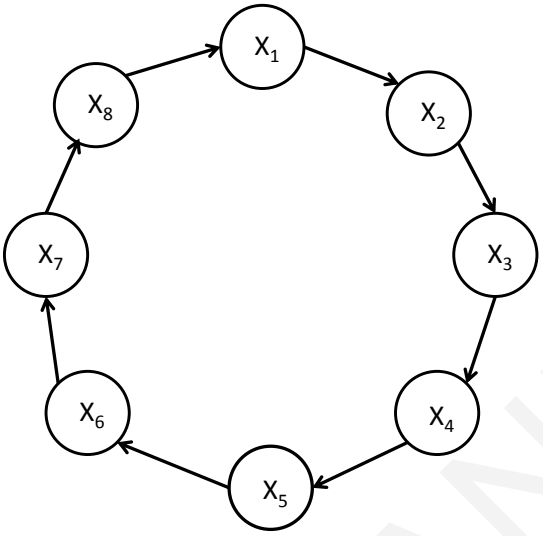

FIGURE 4.8: Ring digraph with eight nodes and diameter $D = 7$, using initial values $x[0] = [0.3, 0.3, 0.3, 1, 1, 1, 1, 0.3]^T$.

Simulation results for ring digraph of eight nodes		
Algorithm	Last Time-Step	No. of Transmissions
Y&S Algorithm	118	3000
Algorithm 4	123	1830
Algorithm 5	146	2201

TABLE 4.1: Required number of time steps and transmissions for the ring digraph of eight nodes in Fig. 4.8, with $\varepsilon = 0.0001$ for the Y&S Algorithm in [1], and $\varepsilon = 0.0001/D$ for Algorithms 4 and 5.

needed from each node (the iteration value $x_i[k]$, the maximum $y_i[k]$, and the minimum $z_i[k]$).

Role of Probability p in Algorithm 4

We consider a ring digraph consisting of fourteen nodes as in Fig 4.10, with diameter $D = 13$, and use $x[0] = [a,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ a,\ a,\ a]^T$ with $a = 0.0001$ and $b = 0.2$ as initial values. Table 4.2 shows the results of running Algorithm 1 with identical values of p_i for all nodes (captured by parameter p).

The main message taken from the above simulations is that it appears that the bigger the probability p in Event-Triggered Rule 2a is, the higher the number of time steps

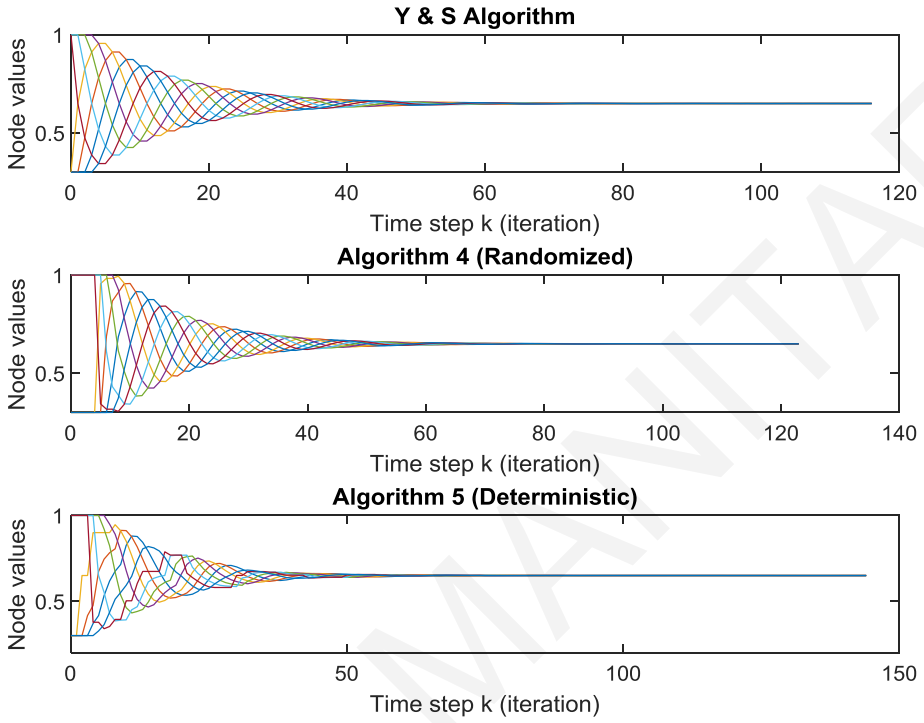

FIGURE 4.9: Evolution of node values for the ring digraph of Fig. 4.8, with $\varepsilon = 0.0001$ for the Y&S Algorithm, and $\varepsilon = 0.0001/D$ for Algorithms 4 and Algorithm 5.

Simulation results on the role of probability p		
Probability	Last Time-Step	No. of Transmissions
p=0.95	*4675*	*31186*
p=0.90	*1347*	*17574*
p=0.85	*1097*	*15570*
p=0.80	*1062*	*15096*
p=0.75	*719*	*12872*
p=0.70	*533*	*11624*
p=0.65	*485*	*10836*
p=0.60	*449*	*10806*
p=0.55	*436*	*10466*
p=0.50	*422*	*10106*

TABLE 4.2: Required number of time steps and transmitted values for Algorithm 4, for different values of probability p.

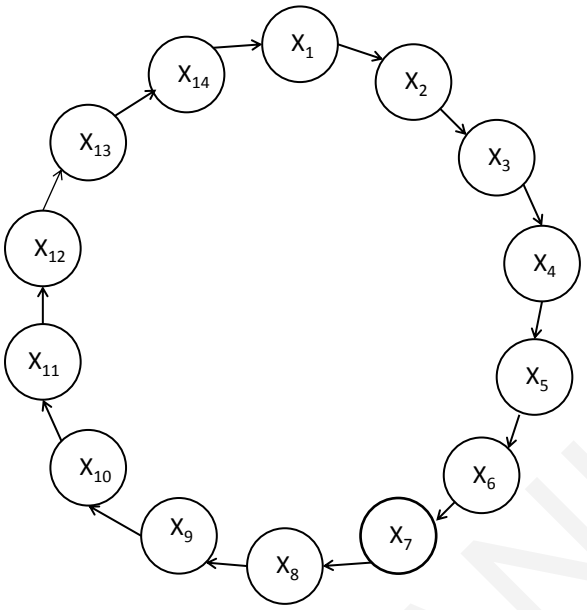

FIGURE 4.10: Ring digraph with fourteen nodes and diameter $D = 13$, using initial values $x[0] = [0.0001,0.3,0.3,0.3,0.3,0.3,0.3,0.3,0.0001,0.0001,0.0001,0.0001, 0.0001, 0.0001]^T$.

that are required to terminate (and the higher the number of transmissions needed in order to reach approximate average consensus). The way p affects the number of time steps *after* the nodes reach approximate average consensus is straightforward: the larger p is, the larger the probability that a node transmits following a transmission by an in-neighbor and the longer it will take for nodes to stop after they reach approximate average consensus (also refer to Remark 2). However, p might also affect the speed with which nodes reach approximate average consensus (for instance, in the ring network of Example 1, a larger p will cause the nodes that are not aware of the discrepancy to transmit more frequently, thus, aiding convergence). We have experimentally observed that, in order to achieve the best performance in terms of the number of time-steps and the number of transmitted values, the in-degree $\mathcal{D}_i^-$ for each node i should be taken into account in order to choose the appropriate value of probability p_i (a bigger in-degree $\mathcal{D}_i^-$ for node i suggests that a smaller probability p_i is desirable), but this also depends on the initial values of the nodes and also on the way that the network is connected.

Role of c_i and b_i in Algorithm 5

We present simulation results for different combinations of c_i and b_i choices, in order to better understand their role on the execution time and the number of transmitted values in Algorithm 5. In the simulations, we make use of the ring digraph introduced

		b				
		1	**2**	**3**	**4**	**5**
c	**1**	NS	7742	14363	66403	26240
	2	*NS*	NS	7674	8658	16178
	3	*NS*	*NS*	NS	6980	6404
	4	*NS*	*NS*	*NS*	1623	9875
	5	*NS*	*NS*	*NS*	*NS*	NS

TABLE 4.3: Required number of time steps for Algorithm 5, for different combinations of values for c and b.

in the previous section, consisting of fourteen nodes ($D = 13$), using the same initial values as before ($x[0] = [a,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ b,\ a,\ a,\ a]^T$ with $a = 0.0001$ and $b = 0.2$) and $\varepsilon = 0.0001$. For simplicity, we assume that c_i and b_i are the same for all fourteen nodes, and denote them by c and b respectively. This means that the sufficient condition in (4.3) becomes $c < b$ (because $\mathcal{D}_i^+ = 1$ for each node i).

Table 4.3 shows, for different combinations of c and b, the number of steps needed for all nodes to stop ("NS" means that the algorithm did not stop within the maximal number of iterations we run it for). As expected, when the condition in (4.3) is satisfied (i.e., $c < b$) the nodes stop in finite time. It is also clear from the table that even when the condition in (4.3) is not satisfied, the nodes may still stop in finite time (for example, when $c = b = 4$, the nodes stop at time step 1623).

What is perhaps more interesting in the simulation results, is that when the values of c and b are close, the nodes are able to start the iterative process earlier but more time steps are required at the end of the process to terminate. Moreover, one can also see that when the values of c and b are not close enough, more time steps are required; this may be due to the fact that at the beginning more transmissions are required for the nodes to start the process of activating the neighboring nodes although their values are within the value of ε.

Random Graph

In this subsection we present results based on a randomly generated digraph with 100 nodes with edge density equal to 0.2, i.e., for each pair of nodes $i, j \in X$, $i \neq j$, edge $(i, j) \in E$ (i.e., the edge is incorporated in the graph) with probability equal to 0.2, independently between different edges. Nodes have initial values randomly chosen to be uniform in the interval $[0, 1]$, independently between each other. Both algorithms require the same number of time-steps and transmissions (11 and 2020 respectively) for reaching approximate average consensus in finite time. Due to the fact that the digraph is generated randomly, obtaining a doubly stochastic matrix is not straightforward, so simulations were only carried out with the two proposed protocols that use a column stochastic matrix.

Simulation Results for a Randomly generated Graph of 100 nodes		
Algorithm	Last Time-Step	No. of Transmissions
Algorithm 4	10	2020
Algorithm 5	10	2020

TABLE 4.4: Required number of time steps and transmissions for Algorithms 4 and Algorithm 5, with $\varepsilon = 0.0001/D$.

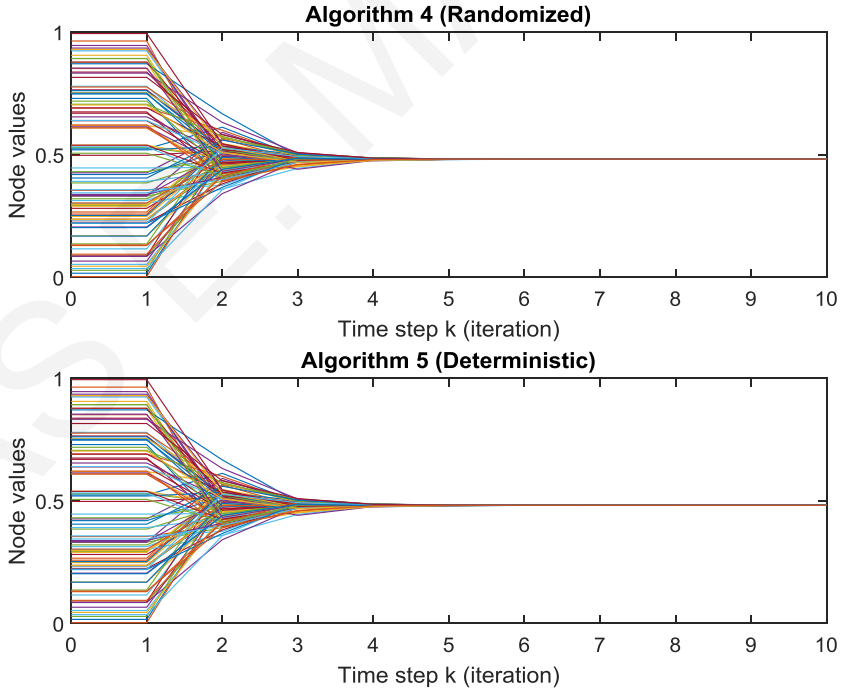

FIGURE 4.11: Evolution of node values for Algorithm 4 and Algorithm 5, with $\varepsilon = 0.0001$.

Comparison of three different approaches on finite-time consensus			
Average consensus Policy	Exact finite-time (1)	Approximate finite-time (2)	Approximate event-triggered (3)
Magnitude of error	0	ϵ	$2\varepsilon D$
global parameters	None	Graph Diameter	None
Handles state uncertainties	No	Yes	Yes
Handle delays	If upper bound is known	If upper bound is known	-

TABLE 4.5: Comparison of the three main finite-time average consensus algorithms: (1) refers to [2], (2) refers to Algorithm 3, and (3) refers to Algorithm 4 and Algorithm 5.

Before closing, we point out that on a random digraph with random initial conditions, both algorithms typically require exactly the same number of time steps to terminate (and the same number of transmitted values). The main reason is that, in both proposed algorithms, the second rule (2a or 2b) is not activated. However, one should keep in mind that there are also special cases where Event-Triggered Rule 2a or 2b become necessary.

4.5 Discussion

In this part of the chapter we present the two mentioned approaches proposed in the literature review of this topic against our own work presented in this chapter. In 4.5 we can clearly see that the proposed method by [2] converges to the exact average value of the network, it does not need to know any global parameters, but it cannot handle the case for which the state has uncertainties (either due to communication with limited capacity or noisy measurements). The method proposed in [38], can handle easily delays since if an upper bound is known (in the simplest case, a node can update its value with a delay equal to the maximum and the approach follows nominal one), but what it requires is knowledge on the graph Diameter (or an upper bound). Our own proposed schemes, the event-triggered mechanisms has an error that depends on the network diameter and it is unknown whether it works under delays.

Chapter 5

SUMMARY AND FUTURE DIRECTIONS

5.1 Summary

In this thesis, we have examined the use of linear iterative strategies for distributed average consensus. In such strategies, each node updates its value based on its own previous value and the values of its neighbors. We presented how networks employing these strategies can be modeled as linear dynamical systems, and developed a control theoretic framework to analyze the capabilities and the features of these strategies. More specifically, we presented the following key results of this thesis.

- We considered the problem of privacy preserving asymptotic average consensus in the presence of malicious agents. We proposed a protocol that can be followed by nodes that desire to maintain their privacy (i.e., avoid exposing their own initial value) and we showed that privacy can be guaranteed when the proposed protocol is used in networks whose topology satisfies certain conditions. In particular, we showed that the malicious nodes are not able to identify the initial values of the nodes following the protocol, as long as the nodes following the protocol have a neighbor that is not directly connected to any of the malicious nodes and all independent paths (if any) to a malicious node are through a node following the protocol. Specifically, if this condition is satisfied, the network will reach average consensus and the initial values of the nodes following the protocol will not be revealed. This condition on the communication topology of the system is sufficient but not necessary.

- We also investigate the ability of the malicious-curious nodes to estimate the initial values of the nodes following the proposed protocol and we showed simulation results that malicious-curious nodes are not able to estimate the exact initial values of the nodes that follow the proposed privacy preserving strategy.

- We considered the problem of distributed stopping for obtaining ε-approximate average consensus in undirected graphs. We show that $(D\varepsilon)$-approximate average consensus is guaranteed when the proposed event-triggered algorithms are followed (where D is the diameter of a given undirected graph). In particular, we showed that by exchanging only local information, the nodes can individually identify when to stop iterating and (if necessary) when to start transmitting again. The results in the chapter are supported through simulation examples.

- We considered the problem of distributed stopping for obtaining ε-approximate average consensus in digraphs. The main challenge in digraphs is the fact that a node that sees a discrepancy with the value of one of its in-neighbors cannot directly inform that in-neighbor, but has to do it indirectly via its out-neighbors. We have shown that $(2\varepsilon D)$-approximate average consensus is guaranteed when one of the proposed event-triggered algorithms is followed (where D is the diameter of the given digraph). In particular, we showed that by exchanging only local information, the nodes can individually identify when to stop iterating and (if necessary) when to start transmitting again. The results in the chapter are supported through simulation examples.

5.2 Future Directions

It is likely that our control-theoretic framework for linear iterative strategies can be extended and generalized to yield greater insights into the capabilities of such strategies. Some topics that deserve further investigation are described below.

- We assumed in our setup in Chapter 2 that malicious-curious nodes have full knowledge of the proposed protocol and are allowed to collaborate arbitrarily among themselves (exchanging information as necessary), but do not interfere in the computation of the average value in any other way. However, malicious nodes may not simply be curious but may also attempt to interfere with the computation of the average value of the network. This is an interesting challenge to consider in the future.

- In our study for privacy preserving average consensus we had each node run the linear iteration algorithm as in (1.1). We can propose a protocol that will be a variation of the standard gossip based protocol and can be used in the absence of privacy requirements. This protocol will allow the nodes to asymptotically obtain the average of their initial values by following a gossip-based linear iteration. Obtaining such a variation of a gossip-based algorithm can create a time-varying condition (on the time that nodes following the proposed protocol cancel out the random offset values that it has added to the network) that will ensure privacy

for the nodes following the protocol despite the presence of malicious agents in the network.

- In the area of finite time approximate average consensus, we will focus on bounding the number of iterations needed to reach approximate average consensus in terms of parameters of interest (e.g., by using the fact that nodes that interact have ratios that differ in absolute difference by at least ε) and on tightening some of the bounds obtained in this thesis. Moreover we will investigate how the proposed protocols can be modified in such a way in order to be able to reach to the exact average value of the network in a finite time.

- Another interesting topic for future investigation will be the modification of the proposed algorithms in Chapters 2-4, in order to enable the nodes to preserve the privacy of their initial value, and at the same time reach agreement on the exact average value of the network in a finite time.

Bibliography

[1] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," in *Proceedings of* 45^{th} *Annual Allerton Conference on Communication, Control, and Computing*, 2007, pp. 715–720.

[2] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed finite-time average consensus in digraphs in the presence of time delays," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 4, pp. 370–381, 2015.

[3] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proceedings of the American Control Conference (ACC)*, July 2007, pp. 711–716.

[4] Y. Yuan, G.-B. Stan, M. Barahona, L. Shi, and J. Gonçalves, "Decentralised minimum-time consensus," *Automatica*, vol. 49, no. 5, pp. 1227–1235, May 2013.

[5] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed finite-time average consensus in digraphs in the presence of time delays," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 4, pp. 370–381, 2015.

[6] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.

[7] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multiagent coordination," in *Proceedings of American Control Conference (ACC)*, 2005, pp. 1859–1864.

[8] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proceedings of First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 537–542.

[9] 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, January 2007.

[10] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, pp. 726–737, March 2008.

[11] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." DTIC Document, Tech. Rep., 1984.

[12] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[13] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[14] J. Hromkovič, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance*. Springer Science & Business Media, 2005.

[15] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[16] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation and consensus using linear iterative strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 650–660, 2008.

[17] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the* 44^{th} *Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 482–491.

[18] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2010, pp. 1753–1757.

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

[20] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, 2006, pp. 168–176.

[21] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed matrix scaling and application to average consensus in directed graphs," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 667–681, 2013.

[22] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proceedings of the American Mathematical Society*, vol. 14, no. 5, pp. 733–737, 1963.

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

[24] C. N. Hadjicostis, A. D. Domínguez-García, T. Charalambous *et al.*, "Distributed averaging and balancing in network systems: with applications to coordination and control," *Foundations and Trends® in Systems and Control*, vol. 5, no. 2-3, pp. 99–292, 2018.

[25] C.-T. Chen, *Linear System Theorey and Design.* New York, NY: Holt, Rinehart and Winston, 1984.

[26] M. Kefayati, M. S. Talebi, B. H. Khalaj, and H. R. Rabiee, "Secure consensus averaging in sensor networks using random offsets," in *IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*, 2007, pp. 556–560.

[27] Y. Mo and R. M. Murray, "Privacy preserving average consensus," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 753–765, 2017.

[28] N. M. Freris and P. Patrinos, "Distributed computing over encrypted data," in *Proceedings 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2016.* IEEE, 2016, pp. 1116–1122.

[29] C. N. Hadjicostis, "Privary preserving distributed average consensus via homomorphic encryption," in *Proceedings 2018 IEEE Conference on Decision and Control (CDC).* IEEE, 2018, pp. 1258–1263.

[30] B. Gharesifard and J. Cortes, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs,," in *European Journal of Control, 2012.* IEEE, 2012, pp. 539–537.

[31] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed computing over encrypted data," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 667–681, 2013.

[32] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterations in the presence of malicious agents-part i: Attacking the network," in *Proc. American Control Conf., 2008.* IEEE, 2008, pp. 1350–1355.

[33] ——, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1495–1508, 2011.

[34] ——, "Distributed function calculation via linear iterations in the presence of malicious agents-part ii: Overcoming malicious behavior," in *Proc. American Control Conf., 2008.* IEEE, 2008, pp. 1356–1361.

[35] L. Xiao and S. Boyd, "Distributed average consensus with time-varying metropolis weights," in *Proc. of the International Conference on Information Processing in Sensor Networks.* IEEE, 2005, pp. 63–70.

[36] D. B. West, *Introduction to Graph Theory.* Pearson, 2001.

[37] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proceedings of American Control Conference (ACC)*, 2006, pp. 3551–3556.

[38] S. T. Cady, A. D. Domínguez-García, and C. N. Hadjicostis, "Finite-time approximate consensus and its application to distributed frequency regulation in islanded ac microgrids," in *2015 48th Hawaii International Conference on System Sciences.* IEEE, 2015, pp. 2664–2670.

[39] N. E. Manitara and C. N. Hadjicostis, "Distributed stopping for average consensus in undirected graphs via event-triggered strategies," *Automatica*, vol. 70, pp. 121–127, August 2016.