LINEAR ITERATIVE STRATEGIES FOR INFORMATION DISSEMINATION AND PROCESSING IN DISTRIBUTED SYSTEMS

BY

SHREYAS SUNDARAM

BASc., University of Waterloo, 2003
M.S., University of Illinois at Urbana-Champaign, 2005

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Doctoral Committee:

Adjunct Associate Professor Christoforos N. Hadjicostis, Chair
Professor M. Tamer Başar
Professor P. R. Kumar
Professor Muriel Médard, MIT
Professor Nitin H. Vaidya

ABSTRACT

Given an arbitrary network of interconnected nodes, each with some initial value, we develop and analyze distributed strategies that enable a subset of the nodes to calculate any function of the initial values of the nodes. Specifically, we focus on linear iterative strategies where each node updates its value to be a weighted average of its previous value and those of its neighbors over the course of several time-steps. We show that this approach can be viewed as a linear system with dynamics that are given by the weight matrix of the linear iteration, and with outputs for each node that are captured by the set of values that are available to that node at each time-step. Based on this insight, we develop a framework to analyze linear iterative strategies from the perspective of control and linear system theory, drawing upon the notions of observability theory, structured system theory, dynamic system inversion, invariant zeros of linear systems, and coding theory.

We start by showing that in networks with time-invariant topologies, choosing the weights for the linear iteration randomly from a field of sufficiently large size will (with high probability) allow every node to calculate any arbitrary function of the initial node values after running the linear iteration for a finite number of time-steps. We show that the number of time-steps required can be determined from the structure of the network topology, and in fact, may be minimal over all possible strategies for information dissemination. We demonstrate that the nodes can implement the linear iterative strategy in a purely decentralized manner, without requiring a centralized coordinator to design and assign the weights in the network.

We then apply our results to calculate linear functions in networks with real-valued transmissions and updates, where communications between nodes are corrupted by additive noise. We show that by using a linear iterative strategy with almost any set of real-valued weights for a finite number of time-steps, any node in the network will be able to calculate an unbiased estimate of any linear function by taking a linear combination of the values that it sees over the course of the linear iteration. Furthermore, for a given set of weights, we describe how to choose this linear combination to minimize the variance of the unbiased estimate calculated by each node.

Next, we consider the problem of distributed function calculation in the presence of faulty or malicious agents, and we study the susceptibility of linear iterative strategies to

misbehavior by some nodes in the network; specifically, we consider a node to be malicious if it updates its value arbitrarily at each time-step, instead of following the predefined linear iterative strategy. If a certain node x_i has $2f$ or fewer node-disjoint paths from another node x_j, we show that it is possible for a set of f malicious nodes to conspire in a way that makes it impossible for x_i to correctly calculate any function of x_j's value. Conversely, when every node in the network has at least $(2f+1)$ node-disjoint paths to x_i, we show that x_i can calculate any arbitrary function of all initial node values after running the linear iteration for a finite number of time-steps (upper bounded by the number of nodes in the network) using almost any set of real-valued weights (i.e., for all weights except for a set of measure zero), despite the (possibly coordinated) actions of up to f malicious nodes. Our analysis is constructive, in that it provides specific attacking and decoding strategies which the malicious and non-malicious nodes can use to achieve their objectives.

We then utilize our framework to study the topic of network coding in the presence of malicious nodes. Specifically, for any fixed network with a given set $\mathcal{S}$ of source nodes, a given set $\mathcal{T}$ of sink nodes, and an unknown set $\mathcal{F}$ of malicious nodes, we examine the problem of transmitting a stream of information from every source node to all of the sink nodes (possibly after some delay). We consider the wireless broadcast communication model, where each transmission by a node is received by all of its neighbors in the network. We use linear network coding to disseminate information, whereby at each time-step, each node transmits a value that is a linear combination of the most recent transmissions of its neighbors. We allow for the possibility that the malicious nodes conspiratorially transmit arbitrary values in an effort to disrupt the network, and we show that linear network codes in the presence of such attackers can be conveniently modeled as a linear dynamical system with unknown inputs. This allows us to use techniques from control theory pertaining to dynamic system inversion to show that if there are $|\mathcal{S}|+2f$ node-disjoint paths from $\mathcal{S}\cup\mathcal{J}$ (where $\mathcal{J}$ is any set of at most $2f$ nodes) to a given sink node, then that sink node can uniquely recover the stream of data from every source node and also locate and isolate f (possibly coordinated) malicious nodes. In particular, we show that linear network codes inherently make use of the redundancy in the network topology to achieve this robustness, and do not require any additional redundancy to be added to the source data. Our approach can be applied over arbitrary fields (of sufficiently large size), and immediately provides a systematic decoding procedure that can be used by the sink nodes, along with an upper bound on the latency required to recover the source streams.

Finally, we generalize existing theory on structured linear systems to encompass finite fields, and apply these results at various points in the thesis. Specifically, we show that certain structural properties of linear systems remain valid with high probability if the parameters are chosen from a finite field of sufficiently large size. In the process, we obtain new insights into structural properties of linear systems (such as an improved upper bound on the observability index in terms of the topology of the graph associated with the system).

ACKNOWLEDGMENTS

I am indebted to my adviser, Christoforos N. Hadjicostis, for offering me the perfect mix of guidance and independence during my time as a graduate student. This thesis would not have been possible without his insight and expertise, not to mention his careful proofreading of countless drafts of my work. I could not have asked for a better adviser. I would also like to thank the other members of my thesis committee (Professors Tamer Başar, P. R. Kumar, Muriel Médard and Nitin Vaidya) for their helpful comments and suggestions, and for their patience during the scheduling process.

The strength of the Decision and Control group stems not only from its unparalleled technical expertise, but also from the genuine atmosphere of congeniality that infuses it. I have enjoyed my time with the group tremendously, and I would like to thank Professors Andrew Alleyne, Tamer Başar, Carolyn Beck, Geir Dullerud, P. R. Kumar, Cedric Langbort, Daniel Liberzon, Sean Meyn and Dusan Stipanovic for the pleasant interactions that we have had over the years (whether they were in the hallways of the Coordinated Science Laboratory, on the beaches of Cancun, in between the stalls of the farmers' market, or high atop Times Square). We are also lucky to have some of the greatest support staff that one can ask for, and I would like to thank Francie Bridges, Jana Lenz, Becky Lonberger and Ronda Rigdon for always cheerfully going above and beyond the call of duty to keep the group running.

My attachment to Illinois is inextricably linked to the many good friends that I have made during my time here. I would like to thank Karan Bhatia, Cathy Chu, Sriram Narayanan, Pavan Sannuti, and especially Neera Jain for all of the meals, laughter, debates, and workouts that we have shared. I would also like to thank Elina Athanasopoulou, Kira Barton, Doug Bristow, Andrew Cannon, Vikas Chandan, Debasish Chatterjee, Nikhil Chopra, Tim Deppen, Louis and Linnea DiBerardino, Nikos Freris, Bobby Gregg, Ramakrishna Gummadi, Brian Helfrich, Brandon Hencey, David Hoelzle, Peter Hokayem, Hemant Kowshik, Tung Le, Bin Li, Lingxi Li, Deepti Narla, Scott Manwaring, Silvia Mastellone, Yu Ru, Anoosh Saboori, Yoav Sharon, Alex Shorter, Shyama Sridharan, Kunal Srivastava, George Takos, Rouzbeh Touri, Serena Tyson, Vladimeros Vladimerou, Linh Vu, Rosanna Yeh and Serdar Yuksel for their invaluable friendship. I have no doubt that all of them will go on to achieve tremendous things in the future.

Finally, I would like to thank my family for their constant love and support. This thesis is for them.

The material in this thesis is based upon work supported in part by the National Science Foundation under NSF Career Award 0092696, NSF ITR Award 0218939, NSF ITR Award 0426831, and NSF CNS Award 0834409. The research leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. I would also like to thank Boeing for partial support of this research. Any opinions, findings, and conclusions or recommendations expressed in this document are my own, and do not necessarily reflect the views of NSF, the European Community, or Boeing.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND BACKGROUND

1.1 Introduction

It has long been recognized that distributed and decentralized systems have certain desirable characteristics such as modularity, fault-tolerance, and simplicity of implementation. For these reasons, the distributed system paradigm is finding a foothold in an extraordinarily large number of applications (many of which are life- and mission-critical), ranging from transmitting patient diagnostic data in hospitals via multi-hop wireless networks [1], to coordinating teams of autonomous aircraft for search and rescue operations [2]. A key requirement in such systems and networks is for some or all of the nodes to calculate some function of certain parameters, or for some of the nodes to recover sequences of values sent by other nodes. For example, sink nodes in sensor networks may be tasked with calculating the average value of all available sensor measurements [3, 4, 5]. Another example is the case of multi-agent systems, where agents communicate with each other to coordinate their speed and direction [6, 7]. Yet another example is the case of transmitting streams of values in communication networks from a set of source nodes to a set of sink nodes [8, 9]. The problems of information dissemination and function calculation in networks have been studied by the computer science, communication, and control communities over the past few decades, leading to the development of various protocols [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. A special case of the distributed function calculation problem is the *distributed consensus* problem, where all nodes in the network calculate the same function [11], and this topic has experienced a resurgence in the control community due to its applicability to tasks such as coordination of multi-agent systems, and its ability to model physical and biological behavior such as flocking (e.g., see [6, 7], and the references therein). In these cases, the approach to consensus is to use a linear iteration where each node repeatedly updates its value as a weighted linear combination of its own value and those of its neighbors [20, 21, 22, 23, 24, 25, 26]. These works have revealed that if the network topology satisfies certain conditions, the weights for the linear iteration can be chosen so that all of the nodes asymptotically converge to the same value (even when the topology is time-varying). One of the benefits of using linear iteration-based consensus schemes is that, at each time-step, each node only has to transmit a single value to each of its neighbors, and perform simple

linear operations (consisting of additions and multiplications by scalars). However, almost all of the linear iterative schemes currently present in the literature only produce asymptotic convergence (i.e., exact consensus is not reached in a finite number of steps).

In this thesis, we study the general problem of disseminating information and distributively calculating functions in networks, and we treat the topic of distributed consensus as a special case of this problem. Specifically, we analyze linear iterative strategies for information dissemination and use a control theoretic framework to show that such strategies are substantially more powerful than previously recognized: they can actually allow all nodes to calculate *arbitrary* and different functions of the initial values in a *finite* number of time-steps (possibly minimal over all possible information dissemination strategies), they are resilient to noise in the system, they can tolerate and overcome nodes in the network that behave in malicious and unpredictable ways, and they can be used to transmit streams of information through networks (rather than just initial values). In order to describe our contributions in more detail, we will first need to introduce some terminology, discuss the distributed system model, and provide a mathematical description of the linear iterative strategy.

1.2 Notation and Graph-Theoretic Terminology

In our development, we use $\mathbf{e}_{i,l}$ 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 $\mathbf{e}_i$, $\mathbf{1}$ or $\mathbf{I}$. We will say that an eigenvalue of a matrix $\mathbf{A}$ is *simple* to indicate that it is *algebraically simple* (i.e., it appears only once in the spectrum of $\mathbf{A}$) [27]. The notation $\mathbf{A}'$ indicates the transpose of matrix $\mathbf{A}$. We will denote the rank of matrix $\mathbf{A}$ by $\text{rank}(\mathbf{A})$, and we will denote the range (or column space) of matrix $\mathbf{A}$ by $\text{range}(\mathbf{A})$. The notation $\text{diag}\,(\cdot)$ indicates a square matrix with the quantities inside the brackets on the diagonal, and zeros elsewhere. The expected value of a random parameter A is denoted by $E[A]$, and the probability of an event A is denoted by $\Pr[A]$. The cardinality of a set $\mathcal{S}$ will be denoted by $|\mathcal{S}|$. For two sets $\mathcal{S}_1$ and $\mathcal{S}_2$, the notation $\mathcal{S}_1 \setminus \mathcal{S}_2$ denotes all elements in $\mathcal{S}_1$ that are not in the set $\mathcal{S}_2$. We will denote an arbitrary field by $\mathbb{F}$, and use the symbol $\mathbb{F}_q$ to denote the field of size q.

1.2.1 Graph-Theoretic Terminology

We will require the following terminology in order to facilitate our discussion. Further details can be found in standard texts on graph theory, such as [28].

A graph is an ordered pair $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is a set of vertices (or nodes), and $\mathcal{E}$ is a set of ordered pairs of vertices, called directed edges. If, $\forall x_i, x_j \in \mathcal{X}$,

$(x_i, x_j) \in \mathcal{E} \Leftrightarrow (x_j, x_i) \in \mathcal{E}$, the graph is said to be undirected. The nodes in the set $\mathcal{N}_i = \{x_j | (x_j, x_i) \in \mathcal{E}, x_j \neq x_i\}$ are said to be neighbors of node x_i, and the in-degree of node x_i is denoted by $\deg_i = |\mathcal{N}_i|$. Similarly, the number of nodes that have node x_i as a neighbor is called the out-degree of node x_i, and is denoted by out-$\deg_i$. A *subgraph* of $\mathcal{G}$ is a graph $\mathcal{H} = \{\bar{\mathcal{X}}, \bar{\mathcal{E}}\}$, with $\bar{\mathcal{X}} \subseteq \mathcal{X}$ and $\bar{\mathcal{E}} \subseteq \mathcal{E}$ (where all edges in $\bar{\mathcal{E}}$ are between vertices in $\bar{\mathcal{X}}$). The subgraph $\mathcal{H}$ is said to be *induced* if, whenever $x_i, x_j \in \bar{\mathcal{X}}$, $(x_i, x_j) \in \bar{\mathcal{E}} \Leftrightarrow (x_i, x_j) \in \mathcal{E}$. The subgraph $\mathcal{H}$ is called *spanning* if it contains all vertices of $\mathcal{G}$ (i.e., $\bar{\mathcal{X}} = \mathcal{X}$).

A *path* P from vertex x_{i_0} to vertex x_{i_t} is a sequence of vertices $x_{i_0} x_{i_1} \cdots x_{i_t}$ such that $(x_{i_j}, x_{i_{j+1}}) \in \mathcal{E}$ for $0 \leq j \leq t-1$. The nonnegative integer t is called the length of the path. A path is called a *cycle* if its start vertex and end vertex are the same, and no other vertex appears more than once in the path. A graph is called *acyclic* if it contains no cycles. A graph $\mathcal{G}$ is a *spanning tree rooted at* x_i if it is an acyclic graph where every node in the graph has a path to x_i, and every node except x_i has out-degree exactly equal to 1. Similarly, a graph is a *spanning forest rooted at* $\mathcal{R} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$ if it is a disjoint union of a set of trees, each of which is rooted at one of the vertices in $\mathcal{R}$. An example of a spanning tree rooted at x_1 is shown in Figure 1.1.a, and an example of a spanning forest rooted at $\mathcal{R} = \{x_1, x_2, x_3\}$ is shown in Figure 1.1.b. For any given set of vertices $\mathcal{S} \subseteq \mathcal{X}$ and a subset $\mathcal{R} \subseteq \mathcal{S}$, we will use the terminology $\mathcal{S}$*-spanning forest rooted at* $\mathcal{R}$ to indicate a forest that includes all of the nodes in the set $\mathcal{S}$.

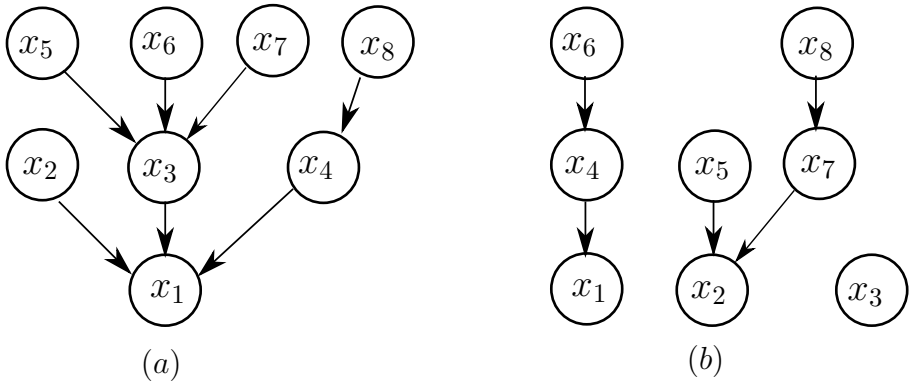

Figure 1.1: (a) Spanning tree rooted at x_1. (b) Spanning forest rooted at $\{x_1, x_2, x_3\}$.

Paths P_1 and P_2 are *vertex disjoint* if they have no vertices in common. Similarly, paths P_1 and P_2 are *internally vertex disjoint* if they have no vertices in common, with the possible exception of the end points. A set of paths $P_1, P_2, \ldots, P_r$ are (internally) vertex disjoint if the paths are pairwise (internally) vertex disjoint. Given two subsets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$, a set of r vertex disjoint paths, each with start vertex in $\mathcal{X}_1$ and end vertex in $\mathcal{X}_2$, is called an r*-linking* from $\mathcal{X}_1$ to $\mathcal{X}_2$.[1] Note that if $\mathcal{X}_1$ and $\mathcal{X}_2$ are not disjoint, we will take each of their common vertices to be a vertex disjoint path between $\mathcal{X}_1$ and $\mathcal{X}_2$ of length zero.

[1]There are various algorithms to find linkings, such as the Ford-Fulkerson algorithm, which has run-time polynomial in the number of vertices [28, 29].

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

The following classical result will play an important role in our derivations (e.g., see [28]).

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

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

1.3 Distributed System Model and Linear Iterative Strategies

The distributed systems and networks that we will be studying in this thesis will be modeled as a directed graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is the set of nodes in the system and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of directed edges (i.e., directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_i can receive information from node x_j). Note that undirected graphs can be readily handled by treating each undirected edge as two directed edges.

We will deal with networks where information is disseminated via the *wireless broadcast* model, whereby each node sends the same information to all of its neighbors. This model, while obviously applicable to wireless networks, also holds when information is obtained by *direct sensing* (i.e., where each node measures or senses the values of its neighbor, as

opposed to receiving that value through a transmission). We assume that every node in the network has an identifier (so that nodes can associate each piece of information that they sense or receive with the corresponding neighbor). Every node is also assumed to have some memory (so that they can store the information that they sense or receive from their neighbors), and sufficient computational capability to perform mathematical operations on this stored information (such as calculating the rank of a matrix, multiplying matrices, etc.). We will generally assume that the network topology is fixed and known *a priori* to certain nodes in the network, but we will discuss relaxations of this assumption at various points in the thesis.[2] We will also generally assume that all communications between nodes are reliable (i.e., any transmission will be eventually received). However we do not assume any fixed time-limit on message delivery;[3] in other words, we assume that nodes in the network wait until they have received transmissions from all of their neighbors, and then execute their transmission or update strategies before waiting for the next transmissions from their neighbors. We will capture this behavior by referring to the behavior of a node at *time-step* k, by which we mean the k–th transmission or update step executed by that node. We assume that all messages are either delivered in the order they were transmitted, or have an associated time-stamp or sequence number to indicate the order of transmission.

1.3.1 Linear Iterative Strategy

Suppose that each node x_i in the distributed system has some initial value, given by $x_i[0] \in \mathbb{F}$ (for some field $\mathbb{F}$). Our goal is for the nodes to calculate some function of $x_1[0], x_2[0], \ldots, x_N[0]$ by updating and/or exchanging their values at each time-step k, based on some distributed strategy that adheres to the constraints imposed by the network topology (which is assumed to be time-invariant). The scheme that we study in this thesis makes use of linear iterations; specifically, each node in the network repeatedly updates its own value to be a weighted linear combination of its previous value and those of its neighbors, which it then transmits. Mathematically, at each time-step, each node updates and transmits its value as

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

[2]One option to handle limited or periodic changes in the network could be for nodes that become aware of changes in graph topology to inform the rest of the network, similar to what is proposed in [9]. Another option would be to treat dropped or added links as faults in the network, and utilize the techniques for fault-tolerant function calculation that we propose later in the thesis.

[3]However, when we discuss networks with potentially faulty or malicious nodes, we will either assume that nodes always transmit a value (even if they are faulty), or we will assume that messages are delivered in a fixed amount of time. We will have to strengthen our assumptions in this way due to the fact it is impossible to perform fault-diagnosis without bounds on the time required to deliver messages – the receiving node will never be able to determine whether an expected message from another node is simply delayed, or if the transmitting node has failed [11].

where the w_{ij}'s are a set of weights from $\mathbb{F}$.[4] Once again, note that the terminology *time-step* k represents the k–th update and transmission by the node; however, we do not assume that all nodes perform their updates or transmissions at the same instant in time (i.e., the k–th time-step for a given node can occur at a different instant in time than the k–th time-step for another node). For ease of analysis, the values of all nodes at time-step k can be aggregated into the value vector $\mathbf{x}[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_N[k] \end{bmatrix}'$, and the update strategy for the entire system can be represented as[5]

$$\mathbf{x}[k+1] = \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{\mathbf{W}} \mathbf{x}[k] \tag{1.1}$$

for $k = 0, 1, \ldots$, where $w_{ij} = 0$ if $j \notin \mathcal{N}_i$ (i.e., if $(x_j, x_i) \notin \mathcal{E}$).

Definition 1.1 (Calculable Function) *Let $g : \mathbb{F}^N \mapsto \mathbb{F}^r$ be a function of the initial values of the nodes (note that $g(\cdot)$ will be a vector-valued function if $r \geq 2$). We say $g(x_1[0], x_2[0], \ldots, x_N[0])$ is calculable by node x_i if it can be calculated by node x_i from some or all of the information that it receives from the linear iteration* (1.1) *over a sufficiently large number of time-steps. We call $g(\cdot)$ a* linear *function if $g(x_1[0], x_2[0], \ldots, x_N[0]) = \mathbf{Q}\mathbf{x}[0]$ for some $r \times N$ matrix $\mathbf{Q}$.*

Definition 1.2 (Distributed Consensus) *The system is said to achieve distributed consensus if all nodes in the system calculate the value $g(x_1[0], \ldots, x_N[0])$ after running the linear iteration for a sufficiently large number of time-steps. When the field under consideration is $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, and*

$$g(x_1[0], x_2[0], \ldots, x_N[0]) = \frac{1}{N}\mathbf{1}'\mathbf{x}[0] \ ,$$

the system is said to perform distributed averaging (i.e., the consensus value is the average of all initial node values).

Definition 1.3 (Asymptotic Consensus) *The system is said to reach asymptotic consensus if*

$$\lim_{k\to\infty} x_i[k] = g(x_1[0], x_2[0], \ldots, x_N[0])$$

for every $i \in \{1, 2, \ldots, N\}$, where $g(x_1[0], x_2[0], \ldots, x_N[0]) \in \mathbb{F}$.

[4] The methodology for choosing the weights in order to achieve certain objectives will be discussed in the subsequent chapters. Much of our discussion will encompass weights from general fields, but we will restrict our attention to real-valued weights (over the field of complex numbers) for certain parts of the thesis.

[5] We will generalize this model in subsequent chapters to include noise, malicious updates, or streams of data from a set of sources.

When $\mathbb{F} = \mathbb{C}$ and $g(x_1[0], x_2[0], \ldots, x_N[0]) = \mathbf{c}'\mathbf{x}[0]$ for some vector $\mathbf{c}'$, the following result by Xiao and Boyd from [22] characterizes the conditions under which the iteration (1.1) achieves asymptotic consensus.

Theorem 1.1 ([22]) *The iteration given by* (1.1) *reaches asymptotic consensus on the linear function* $\mathbf{c}'\mathbf{x}[0]$ *(under the technical condition that* $\mathbf{c}$ *is normalized so that* $\mathbf{c}'\mathbf{1} = 1$*) if and only if the weight matrix* $\mathbf{W}$ *satisfies the following conditions:*

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

Furthermore, Xiao and Boyd showed that the rate of convergence is determined by the eigenvalue with second-largest magnitude. Consequently, to maximize the rate of convergence, they formulated a convex optimization problem to obtain the weight matrix satisfying the above conditions while minimizing the magnitude of the second largest eigenvalue. There is a large amount of literature dealing with choosing the weights in (1.1) so that the system reaches asymptotic consensus on a linear function (including cases where the topology of the network is time-varying). Much of the analysis in these works focuses on modeling the linear iterative strategy as a Markov chain (i.e., by choosing the weight matrix $\mathbf{W}$ in (1.1) to be a stochastic matrix) [6, 24, 30]; the stationary distribution of the chain then determines the consensus function $\mathbf{c}'\mathbf{x}[0]$. As a consequence of this type of analysis, almost all of the existing work on the topic of linear iterative strategies only focuses on reaching asymptotic consensus on a linear function, and only over the field of complex numbers (with real-valued weights and initial values).

1.4 Contributions of Thesis

The main contribution of this thesis is the development of a control– and linear system–theoretic framework to design and analyze linear iterative strategies for information dissemination and distributed function calculation in networks. More precisely, we show how to model the linear iterative strategy as a linear dynamical system and, based on this insight, we leverage tools from control theory and linear system theory (such as observability theory, structured system theory and dynamic system inversion) to characterize the capabilities of such strategies. Our analysis produces the following key results.

Information Dissemination and Function Calculation in a Finite Number of Time-Steps

We consider the linear iteration given by (1.1), and ask the question: What is the total amount of information that any node in the network can infer about the initial values of

other nodes after running the linear iteration for a certain number of time-steps? To answer this question, we use observability theory to show that, after running the linear iteration in connected time-invariant networks for a finite number of time-steps with a random choice of weights from a field of sufficiently large size (including the field of real numbers), each node in the network can actually obtain all of the initial values of the other nodes, and can therefore calculate arbitrary (and possibly different) functions of these values. This result is in contrast to the vast majority of the existing studies of linear iterative strategies, which only focus on the network reaching *asymptotic* consensus on a *linear* function of the initial values over the field of real numbers. Furthermore, our approach allows us to show that the time required in order for any node to accumulate all of the initial values via a linear iterative strategy is upper-bounded by the size of the largest tree in a certain subgraph of the network; we conjecture that this quantity is actually the minimum amount of time required by any protocol to disseminate information, in which case linear iterative strategies will be time-optimal for any given network.

Function Calculation in the Presence of Noise

Having shown that linear iterative strategies of the form (1.1) allow every node to calculate arbitrary functions of the initial values in finite time, we then consider what happens when communications between nodes are corrupted by additive noise. To tackle this problem, we focus on the case of linear iterative strategies with real-valued transmissions and updates, and show that in connected time-invariant networks, every node in the network can obtain an unbiased estimate of any linear function of the initial values after running the linear iteration with almost any real-valued choice of weights for a finite number of time-steps. Furthermore, we show how each node can use the values that it receives during the course of the linear iteration in order to minimize its variance of the estimate. This result is in stark contrast to the majority of existing work on linear iterative strategies which focus on obtaining an unbiased estimate asymptotically (i.e., in an infinite number of time-steps).

Information Dissemination and Function Calculation in the Presence of Malicious Agents

After establishing that function calculation can be achieved in networks when all nodes apply the linear iterative strategy, we examine the case where some nodes in the network *do not* follow the strategy. Specifically, we analyze the susceptibility of such strategies to faulty or (possibly coordinated) malicious behavior by a subset of the nodes in the network. Using the notion of invariant zeros of linear systems, we show that the network connectivity is a determining factor in the capability of the linear iterative strategy to tolerate malicious behavior. Specifically, we show that if there is some node x_j that has $2f$ or fewer node-disjoint paths to some other node x_i, then a set of f malicious nodes can conspire to behave

in such a way that x_i cannot calculate any function of x_j's value. However, for the case of real-valued transmissions and updates, we also show that if all nodes in the network have at least $2f + 1$ node-disjoint paths to x_i, the linear iterative strategy makes it impossible for f or fewer nodes to prevent x_i from calculating any arbitrary function of all initial values. Furthermore, node x_i can uniquely identify all nodes that were malicious during the linear iteration. This topological condition is no stricter than that required by any other information dissemination strategy in the presence of malicious agents; thus, we establish linear iterative strategies as viable and effective methods for disseminating information in networks with malicious agents, thereby narrowing the gap between such strategies and other existing schemes [11, 31, 32, 33]. Once again, we demonstrate that linear iterative strategies exhibit this resilience to malicious behavior for almost any choice of real-valued weights, and only require a finite number of time-steps to disseminate information to all nodes that satisfy the necessary connectivity requirements.

Transmitting Streams of Values

We extend our techniques to treat the problem of linear network coding in the presence of malicious nodes, where a set of sources in the network has to transmit *streams of data* (as opposed to only initial values) to a set of sinks, despite the actions of a set of nodes that transmit arbitrary values at each time-step. In particular, we use dynamic system inversion and structured system theory to show that if the weights for the linear network code are chosen randomly (independently and uniformly) from a field of sufficiently large size, and if the network topology satisfies certain conditions, then with high probability, every sink node can decode each of the source streams and identify all malicious nodes. We show that linear network codes are inherently robust to malicious nodes by virtue of the network topology, and do not require additional redundancy to be added at the source node (unlike all of the existing work that considers the problem of network coding with malicious nodes [34, 35, 36, 37, 38, 39, 40]). Our approach also readily handles networks with cycles (in which case we obtain *convolutional network codes*), and unlike existing solutions for the design of decoders, it does not require manipulation of matrices of rational functions; instead our design procedure focuses entirely on numerical matrices, thereby potentially simplifying the analysis and design of linear network codes.

1.4.1 Contributions to Linear System Theory

Our analysis in this thesis will make extensive use of concepts from linear system theory, and in particular, an area known as *structured system theory* (we will provide further background on these topics in the next section). However, the current work on structured system theory only deals with linear systems with real-valued parameters over the field of complex numbers. While these existing results will be useful when we design linear iterative

strategies with real-valued transmissions and updates, we will not be able to apply them to linear iterative strategies over arbitrary (e.g., finite) fields. In order to deal with this shortcoming, we develop in this thesis a theory of linear structured systems over finite fields. Specifically, we show that certain structural properties of linear systems remain valid with high probability even if the parameters are chosen from a finite field of sufficiently large size. In the process, we obtain new insights into structural properties of linear systems (such as an improved upper bound on the observability index in terms of the graph topology of the system). We use these new insights to derive our results on linear iterative strategies for function calculation and transmitting streams of values through networks.

1.5 Background on Linear System Theory

The control theoretic perspective that we adopt in this thesis will allow us to leverage certain fundamental properties of linear systems in order to characterize the capabilities of linear iterative strategies. In this section, we will review some of the important concepts that we will be using during our derivations.

Consider a linear system of the form

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k] \ , \end{aligned} \tag{1.2}$$

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

$$\underbrace{\begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \\ \vdots \\ \mathbf{y}[L] \end{bmatrix}}_{\mathbf{y}[0:L]} = \underbrace{\begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^L \end{bmatrix}}_{\mathcal{O}_L} \mathbf{x}[0] + \underbrace{\begin{bmatrix} \mathbf{D} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{CB} & \mathbf{D} & \cdots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{L-1}\mathbf{B} & \mathbf{CA}^{L-2}\mathbf{B} & \cdots & \mathbf{D} \end{bmatrix}}_{\mathcal{M}_L} \underbrace{\begin{bmatrix} \mathbf{u}[0] \\ \mathbf{u}[1] \\ \mathbf{u}[2] \\ \vdots \\ \mathbf{u}[L] \end{bmatrix}}_{\mathbf{u}[0:L]} . \tag{1.3}$$

When $L = N - 1$, the matrix $\mathcal{O}_L$ in the above expression is termed the *observability matrix* for the pair $(\mathbf{A}, \mathbf{C})$, and the matrix $\mathcal{M}_L$ is termed the *invertibility matrix* for the set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$. Based on Equation (1.3), one can ask certain natural questions about the system. For example, 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 the outputs? Conversely, if the inputs to the system are not known, but the initial state is known, what

can one infer about the inputs from the outputs? Now if both the initial state and the inputs are not known, what can one infer about these quantities based on Equation (1.3)? The answers to these questions form the basis for much of classical control theory, and are explored in further detail below. More details on linear system theory can be found in standard textbooks (such as [41]); such books typically deal with linear systems over the field of real numbers, but much of the basic theory also applies to arbitrary fields.

Observability of Linear Systems

Definition 1.4 (Observability) *A linear system of the form* (1.2) *is said to be observable if* $\mathbf{y}[k] = 0$ *for all* k *implies that* $\mathbf{x}[k] = 0$ *when* $\mathbf{u}[k] = 0$ *for all* k*. By the linearity of the system, this is equivalent to saying that two different initial states with the same input sequence cannot produce the same output sequence, and thus the initial state* $\mathbf{x}[0]$ *can be uniquely determined from the outputs of the system* $\mathbf{y}[0], \mathbf{y}[1], \ldots, \mathbf{y}[L]$ *(for some* L*) when* $\mathbf{u}[k] = 0$ *for all* k*.*

Since $\mathbf{y}[0 : L] = \mathcal{O}_L \mathbf{x}[0]$ when the inputs are zero (from Equation (1.3)), the system is observable if and only if the observability matrix $\mathcal{O}_L$ has rank N for some nonnegative integer L. Note that the rank of $\mathcal{O}_L$ is a nondecreasing function of L, and bounded above by N. Suppose ν is the first integer for which $\text{rank}(\mathcal{O}_\nu) = \text{rank}(\mathcal{O}_{\nu-1})$. This implies that there exists a matrix $\mathbf{K}$ such that $\mathbf{C}\mathbf{A}^\nu = \mathbf{K}\mathcal{O}_{\nu-1}$. In turn, this implies that

$$\mathbf{C}\mathbf{A}^{\nu+1} = \mathbf{C}\mathbf{A}^\nu \mathbf{A} = \mathbf{K}\mathcal{O}_{\nu-1}\mathbf{A} = \mathbf{K} \begin{bmatrix} \mathbf{C}\mathbf{A} \\ \mathbf{C}\mathbf{A}^2 \\ \vdots \\ \mathbf{C}\mathbf{A}^\nu \end{bmatrix} ,$$

and so the matrix $\mathbf{C}\mathbf{A}^{\nu+1}$ can be written as a linear combination of the rows in $\mathcal{O}_\nu$. Continuing in this way, we see that

$$\text{rank}(\mathcal{O}_0) < \text{rank}(\mathcal{O}_1) < \cdots < \text{rank}(\mathcal{O}_{\nu-1}) = \text{rank}(\mathcal{O}_\nu) = \text{rank}(\mathcal{O}_{\nu+1}) = \cdots ,$$

i.e., the rank of $\mathcal{O}_L$ monotonically increases with L until $L = \nu - 1$, at which point it stops increasing. Since the matrix $\mathbf{C}$ contributes $\text{rank}(\mathbf{C})$ linearly independent rows to the observability matrix, the rank of the observability matrix can increase for at most $N - \text{rank}(\mathbf{C})$ time-steps before it reaches its maximum value, and thus the integer ν is upper bounded as $\nu \leq N - \text{rank}(\mathbf{C}) + 1$. In particular, this means that the system is observable if and only if the rank of $\mathcal{O}_{N-\text{rank}(\mathbf{C})}$ is N. In the linear systems literature, the integer ν is called the *observability index* of the pair $(\mathbf{A}, \mathbf{C})$. In the appendix, we derive a stronger characterization of the observability index for a class of *linear structured systems* (which we describe later in this chapter).

Invertibility of Linear Systems

When the values of the inputs at each time-step are completely unknown and arbitrary in the linear system (1.2), the system is called a *linear system with unknown inputs* [42, 43]. For such systems, it is often of interest to "invert" the system in order to reconstruct some or all of the unknown inputs (assuming that the initial state is known), and this problem has been studied under the moniker of *dynamic system inversion*. This concept will be useful when we design linear iterative strategies to transmit streams of information through networks, and so we will review some of the pertinent concepts related to system inversion here.

We start by noting that if we take the z-transform of the system (1.2), we obtain (assuming that $\mathbf{x}[0] = \mathbf{0}$)

$$\mathbf{Y}(z) = \underbrace{\left(\mathbf{C}\left(z\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} + \mathbf{D}\right)}_{\mathbf{T}(z)}\mathbf{U}(z) \ .$$

Definition 1.5 (Transfer Function Matrix) *The transfer function matrix $\mathbf{T}(z)$ is a $p \times m$ matrix of rational functions of z (with coefficients in the field $\mathbb{F}$), with the (i, j) entry of the matrix capturing how the j–th component of the input sequence $\mathbf{u}$ affects the i–th component of the output sequence $\mathbf{y}$.*

Definition 1.6 (Invertibility) *The system (1.2) is said to have an L-delay inverse if there exists a system with transfer function $\widehat{\mathbf{T}}(z)$ such that*

$$\widehat{\mathbf{T}}(z)\mathbf{T}(z) = \frac{1}{z^L}\mathbf{I}_m \ ;$$

note that $\widehat{\mathbf{T}}(z)$ is an $m \times p$ matrix. The system is invertible if it has an L-delay inverse for some finite L. The least integer L for which an L-delay inverse exists is called the inherent delay of the system.

In order for the system to be invertible, its transfer function must clearly have rank m over the field of rational functions in z (with coefficients in the field $\mathbb{F}$). The following result from [43] and [44] provides a test for invertibility directly in terms of the system matrices $\mathbf{A}, \mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ (in the form of the invertibility matrix).

Theorem 1.2 ([43, 44]) *For any nonnegative integer L,*

$$\mathit{rank}(\mathcal{M}_L) \leq m + \mathit{rank}(\mathcal{M}_{L-1}) \tag{1.4}$$

with equality if and only if the system has an L-delay inverse (note that $\mathit{rank}(M_{-1})$ is defined to be zero). If the system is invertible, its inherent delay will not exceed $L = N - \mathit{nullity}(\mathbf{D}) + 1$, where $\mathit{nullity}(\mathbf{D})$ denotes the dimension of the nullspace of $\mathbf{D}$.

Based on the form of the invertibility matrix in Equation (1.3), note that condition (1.4) is equivalent to saying that the first m columns of $\mathcal{M}_L$ must be linearly independent of each other, and of all other columns in $\mathcal{M}_L$.

Strong Observability and Invariant Zeros of Linear Systems

While the notions of observability and invertibility deal with the separate relationships between the initial states and the output, and between the input and the output, respectively, they do not consider the relationship between the states and input (taken together) and the output. To deal with this, the following notion of *strong observability* has been established in the literature (e.g., see [42, 45, 46, 47]).

Definition 1.7 (Strong Observability) *A linear system of the form* (1.2) *is said to be strongly observable if* $\mathbf{y}[k] = 0$ *for all* k *implies* $\mathbf{x}[0] = 0$ *(regardless of the values of the unknown inputs* $\mathbf{u}[k]$*). This is equivalent to saying that knowledge of the output over some length of time is sufficient to uniquely determine the initial state, regardless of the inputs.*

Recall that observability and invertibility of the system could be determined by examining the observability and invertibility matrices of the system; in order to characterize strong observability, we must examine the relationship between the observability and invertibility matrices. Specifically, it is easy to argue that the system is strongly observable if and only if

$$\operatorname{rank}\left(\begin{bmatrix}\mathcal{O}_L & \mathcal{M}_L\end{bmatrix}\right) = N + \operatorname{rank}\left(\mathcal{M}_L\right)$$

for some nonnegative integer L; in other words, all columns of the observability matrix must be linearly independent of all columns of the invertibility matrix. For real-valued systems, it was shown in [48] that an upper-bound for L is $N-1$ (i.e., if the above condition is not satisfied for $L = N-1$, it will never be satisfied); however the proof in [48] extends to systems over arbitrary fields as well.

In our development, it will be convenient to use an alternative characterization of strong observability. Before stating the result formally, the following definitions will be required.

Definition 1.8 (Matrix Pencil) *For the linear system* (1.2)*, the matrix*

$$\mathbf{P}(z) = \begin{bmatrix}\mathbf{A} - z\mathbf{I}_N & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}$$

is called the matrix pencil of the set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$*.*

Definition 1.9 (Normal-Rank) *The normal-rank of the matrix pencil* $\mathbf{P}(z)$ *is defined as* $rank_n(\mathbf{P}(z)) \equiv \max_{z_0 \in \mathbb{F}} rank(\mathbf{P}(z_0))$*.*

Definition 1.10 (Invariant Zero) *The complex number $z_0 \in \mathbb{F}$ is called an invariant zero of the system* (1.2) *if rank* $(\mathbf{P}(z_0)) < rank_n(\mathbf{P}(z))$.

When the field under consideration is $\mathbb{F} = \mathbb{C}$ (the field of complex numbers), the following theorem provides a characterization of the strong observability of a system in terms of the matrix pencil.

Theorem 1.3 ([42, 45, 46, 48]) *The set* $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ *has no invariant zeros if and only if the matrices* $\mathcal{O}_{N-1}$ *and* $\mathcal{M}_{N-1}$ *satisfy*

$$rank\left(\begin{bmatrix}\mathcal{O}_{N-1} & \mathcal{M}_{N-1}\end{bmatrix}\right) = N + rank\left(\mathcal{M}_{N-1}\right) \ ,$$

(i.e., if and only if the system (1.2) *is strongly observable).*

It is important to note that the above theorem *does not hold* for arbitrary fields. For example, consider the system given by

$$\mathbf{x}[k+1] = \underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 0\end{bmatrix}}_{\mathbf{A}} \mathbf{x}[k] + \underbrace{\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}}_{\mathbf{B}} \mathbf{u}[k]$$
$$\mathbf{y}[k] = \underbrace{\begin{bmatrix}1 & 0 & 0\end{bmatrix}}_{\mathbf{C}} \mathbf{x}[k]$$

over the binary field $\mathbb{F}_2 = \{0, 1\}$. The matrix pencil for this system is

$$\mathbf{P}(z) = \begin{bmatrix}\mathbf{A} - z\mathbf{I}_3 & \mathbf{B}\\ \mathbf{C} & 0\end{bmatrix} = \begin{bmatrix}1-z & 0 & 0 & 1\\ 0 & 1-z & 1 & 0\\ 0 & 1 & -z & 0\\ 1 & 0 & 0 & 0\end{bmatrix} \ .$$

One can verify that the above matrix has full rank for $z \in \{0, 1\}$, and thus this system has no invariant zeros over this field. However, the observability matrix and invertibility matrix are given by

$$\mathcal{O}_2 = \begin{bmatrix}1 & 0 & 0\\ 1 & 0 & 0\\ 1 & 0 & 0\end{bmatrix}, \ \mathcal{M}_2 = \begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 0\\ 1 & 1 & 0\end{bmatrix} \ ,$$

and $\text{rank}\left(\begin{bmatrix}\mathcal{O}_2 & \mathcal{M}_2\end{bmatrix}\right) \neq 3 + \text{rank}\left(\mathcal{M}_2\right)$ (which means that the system is not strongly observable). The reason for this discrepancy is that the field $\mathbb{F}_2$ is not *algebraically closed*, which means that not all polynomials with coefficients in this field have a root in the field [49]. In the above example, the submatrix $\left[\begin{smallmatrix}1-z & 1\\ 1 & -z\end{smallmatrix}\right]$ has determinant z^2+z+1 over $\mathbb{F}_2$, which

does not have any roots. Thus, the middle two columns of the matrix pencil are always linearly independent, which causes the system not to have any invariant zeros over this field. However, in an algebraically closed field (such as the field of complex numbers), there will be some values of z for which the middle two columns will have rank less than 2, and thus the system will have invariant zeros. One can show that no finite field is algebraically closed, and thus the above theorem will not hold for any such field. Nevertheless, the above theorem is important because it will provide us with a convenient method to devise robust linear iterative strategies with real-valued transmissions and operations.

1.5.1 Background on Structured System Theory

While the above properties of linear systems were developed for systems with given (numerically specified) $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ matrices, there is frequently a need to analyze systems whose parameters are not exactly known, or where numerical computation of properties like observability are not feasible (e.g., due to the need to manipulate large numerical matrices). In response to this, control theorists have developed a theory of system properties based on the *structure* of the system. Specifically, a linear system of the form (1.2) is said to be *structured* if every entry in the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ is either zero or an independent free parameter [50, 51, 52, 53]. A property is said to hold structurally for the system if that property holds for at least one choice of free parameters. In fact, if the parameters are taken to be real-valued parameters (with the underlying field of operation taken as the field of complex numbers), structural properties will hold generically (i.e., the set of parameters for which the property does not hold has Lebesgue measure zero); this is the situation that is commonly considered in the literature on structured systems [52, 53]. In the appendix of this thesis, we will develop a theory of structured systems over finite fields, but for now, we will review some of the salient results on structured systems with real-valued parameters. These results will be crucial to the development in this chapter.

To study properties of structured systems, one first associates a graph $\mathcal{H}$ with the set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ as follows. The vertex set of $\mathcal{H}$ is given by $\mathcal{X} \cup \mathcal{U} \cup \mathcal{Y}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is the set of state vertices, $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ is the set of input vertices, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_p\}$ is the set of output vertices. The edge set of $\mathcal{H}$ is given by $\mathcal{E}_{xx} \cup \mathcal{E}_{ux} \cup \mathcal{E}_{xy} \cup \mathcal{E}_{uy}$, where

- $\mathcal{E}_{xx} = \{(x_j, x_i) \mid \mathbf{A}_{ij} \neq 0\}$ is the set of edges corresponding to interconnections between the state vertices,
- $\mathcal{E}_{ux} = \{(u_j, x_i) \mid \mathbf{B}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the state vertices,
- $\mathcal{E}_{xy} = \{(x_j, y_i) \mid \mathbf{C}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the state vertices and the output vertices, and

- $\mathcal{E}_{uy} = \{(u_j, y_i) \mid \mathbf{D}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the output vertices.

In our development, we will frequently work with systems where the parameters in the $\mathbf{B}$ and $\mathbf{C}$ matrices are already specified to be either ones or zeros, but the nonzero entries in the $\mathbf{A}$ matrix remain free parameters. In such cases, we will sometimes focus only on the portion of the graph corresponding to the interconnections between the state vertices (given by the edge set $\mathcal{E}_{xx}$). We discuss this issue in greater detail in the Appendix. Interestingly, the topology of the graph associated with the system (in either case) is sufficient to determine whether a system with a given zero/nonzero structure will possess certain properties (such as observability). We will describe some of these graph-based tests in further detail below. It is important to note that these tests only apply to structured systems where the parameters are allowed to take *real* values (over the field of complex numbers). We will later extend some of these tests to handle structured systems over finite fields.

Structural Observability

Definition 1.11 (Structural Observability) *A structured pair $(\mathbf{A}, \mathbf{C})$ is said to be structurally observable if the corresponding observability matrix has full column rank for at least one choice of free parameters.*

The following theorem characterizes the conditions on $\mathcal{H}$ for the system to be structurally observable. The terminology *$\mathcal{Y}$-topped path* is used to denote a path with end vertex in $\mathcal{Y}$.

Theorem 1.4 ([53]) *The structured pair $(\mathbf{A}, \mathbf{C})$ is structurally observable if and only if the graph $\mathcal{H}$ satisfies both of the following properties:*

1. *Every state vertex $x_i \in \mathcal{X}$ is the start vertex of some $\mathcal{Y}$-topped path in $\mathcal{H}$.*
2. *$\mathcal{H}$ contains a subgraph that covers all state vertices, and that is a disjoint union of cycles and $\mathcal{Y}$-topped paths.*

Structural Invertibility

Definition 1.12 (Structural Invertibility) *A structured set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ is said to be structurally invertible if the corresponding transfer function matrix has full column rank for at least one choice of free parameters.*

For structured systems, we will be interested in the *generic normal rank* of the transfer function matrix, which is the maximum rank (over the field of rational functions in z) of the transfer function matrix over all possible choices of free parameters. The following theorem characterizes the generic normal rank of the transfer function of a structured linear system $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ in terms of the associated graph $\mathcal{H}$.

Theorem 1.5 ([53, 54]) *Let the graph of a structured linear system be given by $\mathcal{H}$. Then the generic normal rank of the transfer function of the system is equal to the maximal size of a linking in $\mathcal{H}$ from $\mathcal{U}$ to $\mathcal{Y}$.*

The above result says that if the graph of the structured system has m vertex disjoint paths from the inputs to the outputs, then for almost any (real-valued) choice of free parameters in $\mathbf{A}$, $\mathbf{B}, \mathbf{C}$ and $\mathbf{D}$, the transfer function matrix $\mathbf{C}(z\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}+\mathbf{D}$ will have rank m. Based on Theorem 1.2, this means that the first m columns of the matrix $\mathcal{M}_{N-\text{nullity}(\mathbf{D})+1}$ will be linearly independent of all other columns in $\mathcal{M}_{N-\text{nullity}(\mathbf{D})+1}$.

Structural Invariant Zeros

It turns out that the number of invariant zeros of a linear system is also a structured property (i.e., a linear system with a given zero/nonzero structure will have the same number of invariant zeros for almost any choice of the free parameters) [47]. The same holds true for the normal rank of the matrix pencil (which is the maximum rank that the matrix pencil takes over all choices of free parameters). The following theorems from [47] and [53] characterize the generic number of invariant zeros of a structured system and the normal-rank of a structured matrix pencil in terms of the associated graph $\mathcal{H}$. Once again, the terminology *$\mathcal{Y}$-topped path* is used to denote a path with end vertex in $\mathcal{Y}$.

Theorem 1.6 ([47]) *Let $\mathbf{P}(z)$ be the matrix pencil of the structured set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$. Then the generic normal-rank of $\mathbf{P}(z)$ is equal to N plus the maximum size of a linking from $\mathcal{U}$ to $\mathcal{Y}$.*

Theorem 1.7 ([47, 53]) *Let $\mathbf{P}(z)$ be the matrix pencil of the structured set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, and let the generic normal-rank of $\mathbf{P}(z)$ be $N+m$ even after the deletion of an arbitrary row from $\mathbf{P}(z)$. Then the generic number of invariant zeros of system (1.2) is equal to N minus the maximal number of vertices in $\mathcal{X}$ contained in the disjoint union of a size m linking from $\mathcal{U}$ to $\mathcal{Y}$, a set of cycles in $\mathcal{X}$, and a set of $\mathcal{Y}$-topped paths.*

From Theorem 1.3, one can characterize strong observability of a system (over the field of complex numbers) by examining the number of invariant zeros of that system. Theorem 1.7 thus provides us with a method to analyze the strong observability of a given (real-valued) system over the field of complex numbers.

As mentioned at the start of this section, all of the above results were derived under the assumption that the parameters for the system matrices can take on any real values, and all of the derivations were performed by working in the field of complex numbers [52]. We will be using these existing results to derive linear iterative strategies with real-valued transmissions and operations in this thesis, but we will also be interested in extending our results to transmissions and operations in finite fields. To do this, we will be deriving

(in the Appendix) theorems on structural observability and invertibility over finite fields. Unfortunately, obtaining a characterization of strong observability for structured systems over finite fields appears to be rather difficult, and thus we leave this topic for future work (we will restrict ourselves to linear iterative strategies with real-valued parameters when we make use of the results on strong observability of systems).

1.6 Thesis Outline

We start our analysis in Chapter 2 by showing how to model the linear iterative strategy as a linear dynamical system, and then use observability theory to show how each node in the network can calculate any arbitrary function of the initial values after running the iteration for a finite number of time-steps (upper bounded by the number of nodes in the network). In Chapter 3, we demonstrate that our framework readily handles networks where communications between nodes are corrupted by noise, and show how to distributively calculate unbiased minimum-variance estimates of arbitrary linear functions of the initial values via a linear iterative strategy. In Chapter 4, we show how the linear iterative strategy can be used to effectively deal with faulty or malicious behavior by nodes in the network. In Chapter 5, we extend our framework to treat the case where a set of nodes in the network must transmit a stream of data (as opposed to a single initial value) to a set of sink nodes despite the presence of malicious nodes. We examine linear network codes that have been developed to enable such transmissions and show how one can use techniques from the theory of dynamic system inversion to design decoders for each sink node. We conclude with some directions for future research in Chapter 6. At various points in the thesis, we will be calling on results pertaining to linear structured systems over finite fields, which we derive in the Appendix of the thesis.

CHAPTER 2

OBSERVABILITY THEORY FRAMEWORK FOR DISTRIBUTED FUNCTION CALCULATION

2.1 Introduction and Main Result

We begin our analysis of linear iterative strategies (of the form described in Chapter 1.3.1) by showing how to model them as linear dynamical systems, which will allow us to use techniques from observability theory to analyze the capabilities of these schemes. Specifically, we utilize results on structured systems to show that if the weights for the linear iteration are chosen randomly (independently and uniformly) from a field of sufficiently large size, then with high probability, each node can calculate any desired function of the initial values after observing the evolution of its own value and those of its neighbors over a *finite* number of time-steps.

Before going into the details of our main result, it will be helpful to discuss a motivating example. Consider the network $\mathcal{G}$ shown in Figure 2.1.a; each node in this network possesses a single value from a field $\mathbb{F}$, and node x_1 needs to obtain all of these values. Each node in the network is allowed to transmit a single value from the field $\mathbb{F}$ at each time-step. Under these conditions, how many time-steps will it take for node x_1 to obtain all of the values? To answer this question, note that node x_1 can obtain at most one new value at each time-step from each of its neighbors. Since node x_2 has one child in this network, it will take exactly two time-steps for x_1 to receive the values of x_2 and x_3. In parallel, x_1 is also receiving values from x_4 and x_7; it will take exactly three time-steps for x_1 to receive the values of x_4, x_5, x_6, and exactly four time-steps to receive the values of x_7, x_8, x_9 and x_{10}. Thus, x_1 can receive all values in the network after exactly four time-steps. This example demonstrates that one bottleneck in the network is related to the number of values that must pass through any given neighbor of x_1. To state this in a form that will be easier for us to analyze, we decompose the network into a spanning forest rooted at $\{x_1\} \cup \mathcal{N}_1$ (shown in Figure 2.1.b). The largest tree in this forest has four nodes, which is equal to the number of time-steps required for x_1 to obtain all of the values.

Now consider the network $\mathcal{G}$ shown in Figure 2.2.a, which is no longer a simple spanning tree rooted at x_1. How many time-steps will it take for node x_1 to receive the values of all nodes in this network? One can answer this question by noting that the forest shown in Figure 2.1.b is a subgraph of $\mathcal{G}$, and thus it is definitely possible for x_1 to obtain the

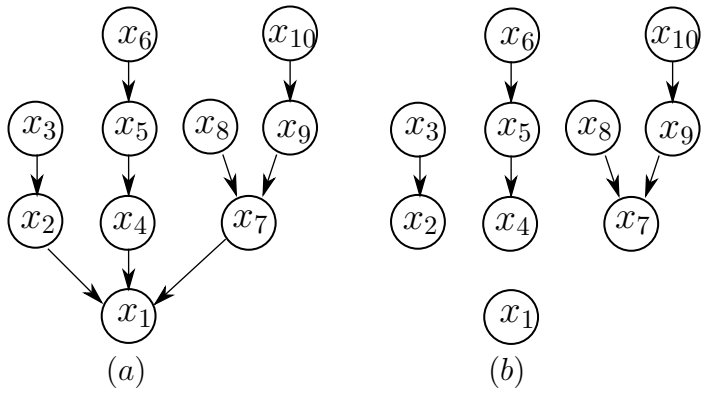

Figure 2.1: (a) Node x_1 needs to receive the values of all of the nodes in this network. (b) A subgraph of the original network that is a spanning forest rooted at x_1, x_2, x_4 and x_7.

values of all nodes in four time-steps. However, one can actually do better by noting that $\mathcal{G}$ also contains the spanning forest rooted at $\{x_1\} \cup \mathcal{N}_1$ shown in Figure 2.2.b, which only has three nodes in any tree. Thus, x_1 can actually receive the values of all nodes in three time-steps in this example.

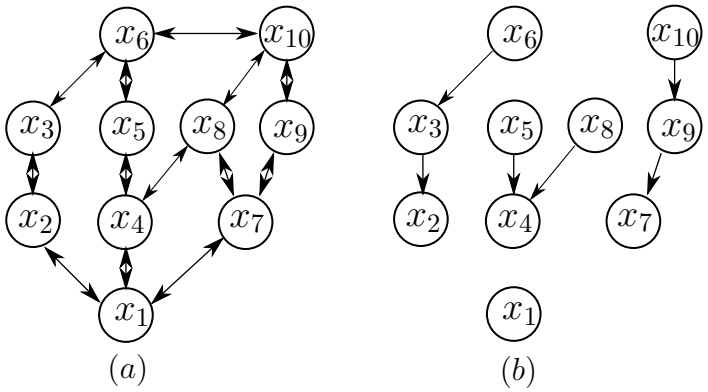

Figure 2.2: (a) Node x_1 needs to receive the values of all of the nodes in this network. (b) A subgraph of the original network that is a spanning forest rooted at x_1, x_2, x_4 and x_7, with only three nodes in the largest tree.

The above examples show that the amount of time required by any node to receive the values of all other nodes is related to the size of the largest tree in a subgraph of the network that is a spanning forest rooted at that node and its neighbors. However, given an arbitrary network (with potentially numerous and complicated interconnections), finding a spanning forest with the smallest possible number of nodes in its largest tree is a daunting task. Furthermore, even if such a tree could be found, it does not address problems where

multiple nodes in the network have to receive all of the values (since by removing edges to create an optimal forest for a certain node, we could be degrading the optimal forest for some other node).

In this chapter, we will demonstrate that linear iterative strategies provide a novel, simple and effective solution to this problem. Specifically, consider the linear iterative strategy given by Equation (1.1), where each node simply updates its value at each time-step to be a weighted linear combination of its previous value and those of its neighbors. For this strategy, we will demonstrate the following key result in this chapter.

Theorem 2.1 *Let $\mathcal{G}$ denote the (fixed) graph of the network, and*

$$\mathcal{S}_i = \{x_j |\ \textit{There exists a path from } x_j \textit{ to } x_i \textit{ in } \mathcal{G}\} \cup \{x_i\}\ .$$

Consider a subgraph $\mathcal{H}$ of $\mathcal{G}$ that is a $\mathcal{S}_i$-spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$, with the property that the size of the largest tree in $\mathcal{H}$ is minimal over all possible $\mathcal{S}_i$-spanning forests rooted at $\{x_i\} \cup \mathcal{N}_i$. Let D_i denote the size of the largest tree of $\mathcal{H}$. Suppose that the weights for the linear iteration are chosen randomly (independently and uniformly) from a field $\mathbb{F}_q$ of size $q \geq (D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$. Then, with probability at least $1 - \frac{1}{q}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$, node x_i can calculate the value $x_j[0]$, $x_j \in \mathcal{S}_i$, after running the linear iteration (1.1) for at most D_i time-steps, and can therefore calculate any arbitrary function of the values $\{x_j[0]\ |\ x_j \in \mathcal{S}_i\}$.

As an example of the quantities specified in the above theorem, consider node x_1 in the network shown in Figure 2.3. The set of neighbors of x_1 is given by $\mathcal{N}_1 = \{x_2, x_4\}$. Since every node in the network has a path to x_1, the set $\mathcal{S}_1$ contains all of the nodes in the network. Next, consider the set of all $\mathcal{S}_1$-spanning forests rooted at $\{x_1, x_2, x_4\}$; all such forests are shown in Figure 2.4. The first forest has 3 nodes in its largest tree, the second forest has 3 nodes in its largest tree, the third forest has 2 nodes in its largest tree, and the fourth forest has 3 nodes in its largest tree. Thus, the third spanning forest satisfies the property that the size of the largest tree is minimal over all spanning forests rooted at $\{x_1, x_2, x_4\}$, and thus $D_1 = 2$. Since $\deg_1 = 2$, we have $(D_1 - 1)(|\mathcal{S}_1| - \deg_1 - \frac{D_1}{2}) = 2$, and Theorem 2.1 implies that x_1 can obtain the values of all nodes after $D_1 = 2$ time-steps in this network with probability at least $1 - \frac{2}{q}$ (if the weights for the linear iteration are chosen randomly from the field $\mathbb{F}_q$ of size $q \geq 2$).

Remark 2.1 *Note that one does not necessarily need to know the quantity D_i in order to make use of the above theorem. Specifically, one can obtain more convenient (but looser) expressions in the above theorem by noting that $1 \leq D_i \leq |\mathcal{S}_i| - \deg_i$ (the upper bound will hold with equality if $\deg_i - 1$ neighbors of x_i are connected only to x_i, and all other nodes in the set $\mathcal{S}_i$ connect to the last neighbor of x_i). With these bounds, one can verify that the*

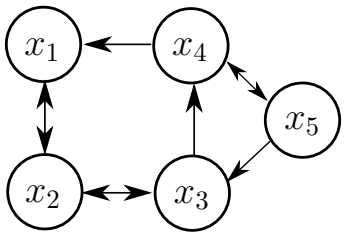

Figure 2.3: Network with $\mathcal{S}_1 = \{x_1, x_2, x_3, x_4, x_5\}$ and $\mathcal{N}_1 = \{x_2, x_4\}$.

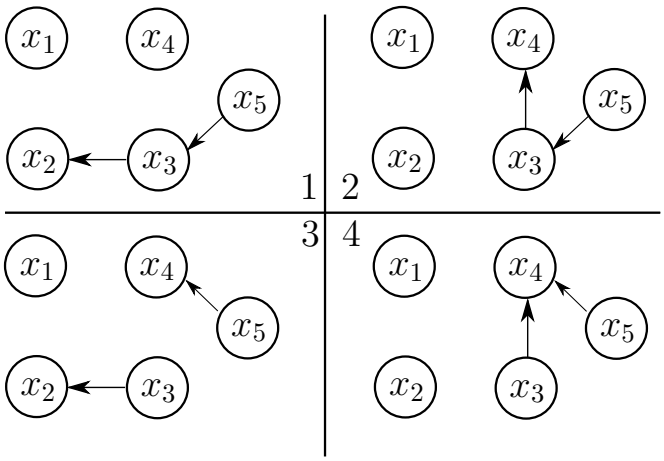

Figure 2.4: All $\mathcal{S}_1$-spanning forests rooted at $\{x_1, x_2, x_4\}$ for the network shown in Figure 2.3.

quantity $(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$ *achieves a maximum value of* $\frac{(|\mathcal{S}| - \deg_i - 1)(|\mathcal{S}_i| - \deg_i)}{2}$ *(and this occurs at* $D_i = |\mathcal{S}_i| - \deg_i$*). Furthermore, since* $N > |\mathcal{S}_i| - \deg_i$*, we can write*

$$\frac{N(N-1)}{2} > \frac{(|\mathcal{S}| - \deg_i - 1)(|\mathcal{S}_i| - \deg_i)}{2} \geq (D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$$

and so Theorem 2.1 implies that if the weights are chosen randomly (independently and uniformly) from a field $\mathbb{F}_q$ *of size* $q \geq \frac{N(N-1)}{2}$*, then with probability at least* $1 - \frac{N(N-1)}{2q}$*,* x_i *can obtain the value* $x_j[0]$*,* $x_j \in \mathcal{S}_i$ *after running the linear iteration for at most* $D_i (\leq |\mathcal{S}_i| - \deg_i)$ *time-steps. For example, in the network shown in Figure 2.3 (with* $N = 5$*), these looser bounds say that the probability that node* x_1 *can obtain the initial values of all nodes is at least* $1 - \frac{10}{q}$ *(for* $q \geq 10$*). If one chooses* $q = 11$*, this bound evaluates to* 0.091*, whereas the tighter bound from Theorem 2.1 evaluates to* $1 - \frac{2}{q} = 0.818$*. Nevertheless, the looser bound is easier to evaluate for a given network (since one does not have to explicitly calculate the quantity* D_i*), and thus it will be useful when we discuss a decentralized implementation of our scheme. Note that even though we may not know the exact value of* D_i*, if we choose the weights from a field of size* $q \geq \frac{N(N-1)}{2}$*,* x_1 *will still be able to obtain the initial values of the nodes in* $\mathcal{S}_i$ *after at most* D_i *time-steps with probability at least* $1 - \frac{1}{q}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$ *(since the condition in Theorem 2.1 will be satisfied with this larger choice of field). In*

particular, if the nodes are allowed to transmit and operate on values from a field of infinite size (such as the field of real numbers), the above properties will hold with probability 1.

Theorem 2.1 reveals that linear iterative strategies essentially bypass the problem of finding an optimal spanning forest in graphs – one can simply choose weights at random from a field of sufficiently large size, and the linear iterative strategy will allow node x_i to receive all of the values in at most D_i time-steps (where D_i is the size of the largest tree in the optimal spanning forest). The precise procedure that node x_i can use to accomplish this will be described later in the chapter. Furthermore, linear iterative strategies also solve the subsequent problem where the initial values are supposed to be disseminated to some or *all* of the nodes in the network. Specifically, once we prove Theorem 2.2, we will also be able to prove the following theorem.

Theorem 2.2 *Let $\mathcal{G}$ denote the (fixed) graph of the network, and*

$$\mathcal{S}_i = \{x_j |\ \textit{There exists a path from } x_j \textit{ to } x_i \textit{ in } \mathcal{G}\} \cup \{x_i\}\ .$$

For each $x_i \in \mathcal{X}$, consider a subgraph $\mathcal{H}_i$ of $\mathcal{G}$ that is a $\mathcal{S}_i$-spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$, with the property that the size of the largest tree in $\mathcal{H}_i$ is minimal over all possible $\mathcal{S}_i$-spanning forests rooted at $\{x_i\} \cup \mathcal{N}_i$. Let D_i denote the size of the largest tree of $\mathcal{H}_i$. Suppose that the weights for the linear iteration are chosen randomly (independently and uniformly) from a field $\mathbb{F}_q$ of size $q \geq \sum_{i=1}^{N}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$. Then, with probability at least $1 - \frac{1}{q}\sum_{i=1}^{N}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$, every node x_i can obtain the value $x_j[0]$, $x_j \in \mathcal{S}_i$, after running the linear iteration (1.1) *for at most D_i time-steps, and can therefore calculate any arbitrary function of the values $\{x_j[0] \mid x_j \in \mathcal{S}_i\}$.*

By following the same reasoning as in Remark 2.1, we can restate the above theorem in terms of looser (but more convenient) bounds as follows.

Corollary 2.1 *Let $\mathcal{G}$ denote the (fixed) graph of the network, and*

$$\mathcal{S}_i = \{x_j |\ \textit{There exists a path from } x_j \textit{ to } x_i \textit{ in } \mathcal{G}\} \cup \{x_i\}\ .$$

For each $x_i \in \mathcal{X}$, consider a subgraph $\mathcal{H}_i$ of $\mathcal{G}$ that is a $\mathcal{S}_i$-spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$, with the property that the size of the largest tree in $\mathcal{H}_i$ is minimal over all possible $\mathcal{S}_i$-spanning forests rooted at $\{x_i\} \cup \mathcal{N}_i$. Let D_i denote the size of the largest tree of $\mathcal{H}_i$. Suppose that the weights for the linear iteration are chosen randomly (independently and uniformly) from a field $\mathbb{F}_q$ of size $q \geq \frac{N^2(N-1)}{2}$. Then, with probability at least $1 - \frac{N^2(N-1)}{2q}$, every node x_i can obtain the value $x_j[0]$, $x_j \in \mathcal{S}_i$, after running the linear iteration (1.1) *for at most D_i time-steps, and can therefore calculate any arbitrary function of the values $\{x_j[0] \mid x_j \in \mathcal{S}_i\}$.*

The above results reveal that linear iterative strategies are simple and powerful methods for disseminating information rapidly in networks; we will develop the proofs of these results over the next few sections. In fact, it seems reasonable to expect that *any* information dissemination strategy will take at least D_i time-steps in order to accumulate all of the node values at node x_i (based on the reasoning that the values of all nodes have to pass through the neighbors of node x_i); however, this result does not appear to have been established anywhere in the literature, and the proof is rather elusive. Based on our intuition, we can make the following conjecture.

Conjecture 1 *Let $\mathcal{G}$ denote the (fixed) graph of the network, and*

$$\mathcal{S}_i = \{x_j | \text{ There exists a path from } x_j \text{ to } x_i \text{ in } \mathcal{G}\} \cup \{x_i\} \ .$$

Consider a subgraph $\mathcal{H}$ of $\mathcal{G}$ that is a $\mathcal{S}_i$-spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$, with the property that the size of the largest tree in $\mathcal{H}$ is minimal over all possible $\mathcal{S}_i$-spanning forests rooted at $\{x_i\} \cup \mathcal{N}_i$. Let D_i denote the size of the largest tree of $\mathcal{H}$. Then it will take at least D_i time-steps for node x_i to receive the values of all nodes regardless of the protocol, and thus linear iterative strategies are time-optimal means of disseminating information in any arbitrary network.

The rest of the chapter is organized as follows. In Section 2.2, we cast function calculation via linear iterative strategies as an observability problem, and then apply concepts from observability theory to analyze properties of these strategies. We use these concepts (along with results from the Appendix on structured observability over finite fields) in Section 2.3 to design weight matrices that allow the desired functions to be calculated at certain nodes. In Section 2.4, we discuss how to implement our scheme in a decentralized manner, and in Section 2.5, we compare our scheme to previous work on linear iterative strategies and distributed consensus. We present an example in Section 2.6.

2.2 Distributed Computation of Linear Functions

We start by asking the question: after running the linear iteration (1.1) with a given weight matrix[1] $\mathbf{W}$ (with entries in field $\mathbb{F}$) for $L_i + 1$ time-steps (for some L_i), what is the set of all linear functions that are calculable by node x_i? The following lemma provides an answer to this question.

Lemma 2.1 *Let $\mathbf{C}_i$ be the $(\deg_i + 1) \times N$ matrix where each row has a single nonzero entry equal to "1" denoting the positions of the vector $\mathbf{x}[k]$ that are available to node x_i (i.e., these positions correspond to nodes that are neighbors of node x_i, along with node x_i itself).*

[1] We will discuss ways to choose appropriate weight matrices in the next section.

Then, node x_i can calculate the linear function $\mathbf{Q}\mathbf{x}[0]$ after running the linear iteration (1.1) for $L_i + 1$ time-steps if and only if the row space of $\mathbf{Q}$ is contained in the row space of the matrix

$$\mathcal{O}_{i,L_i} \equiv \begin{bmatrix} \mathbf{C}_i \\ \mathbf{C}_i\mathbf{W} \\ \mathbf{C}_i\mathbf{W}^2 \\ \vdots \\ \mathbf{C}_i\mathbf{W}^{L_i} \end{bmatrix} \quad . \tag{2.1}$$

Proof: Consider the linear iteration given by (1.1). At each time-step, node x_i has access to its own value and those of its neighbors. From the perspective of node x_i, the linear iteration can then be viewed as the dynamical system

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] \\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}[k] \quad , \end{aligned} \tag{2.2}$$

where $\mathbf{y}_i[k]$ denotes the outputs (node values) that are seen by node x_i during the k–th time-step. Since $\mathbf{x}[k] = \mathbf{W}^k\mathbf{x}[0]$, we have $\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{W}^k\mathbf{x}[0]$, and the set of all values seen by node x_i over $L_i + 1$ time-steps is given by

$$\begin{bmatrix} \mathbf{y}_i[0] \\ \mathbf{y}_i[1] \\ \mathbf{y}_i[2] \\ \vdots \\ \mathbf{y}_i[L_i] \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_i \\ \mathbf{C}_i\mathbf{W} \\ \mathbf{C}_i\mathbf{W}^2 \\ \vdots \\ \mathbf{C}_i\mathbf{W}^{L_i} \end{bmatrix}}_{\mathcal{O}_{i,L_i}} \mathbf{x}[0] \quad . \tag{2.3}$$

The row space of $\mathcal{O}_{i,L_i}$ characterizes the set of all calculable linear functions for node x_i up to time-step L_i. Suppose that the row space of matrix $\mathbf{Q}$ is not contained in the row space of $\mathcal{O}_{i,L_i}$. This means that

$$\text{rank}\left(\begin{bmatrix} \mathcal{O}_{i,L_i} \\ \mathbf{Q} \end{bmatrix}\right) > \text{rank}\left(\mathcal{O}_{i,L_i}\right) \quad ,$$

and so there exists a nonzero vector $\mathbf{v}$ such that $\mathcal{O}_{i,L_i}\mathbf{v} = 0$, but $\mathbf{Q}\mathbf{v} \neq 0$. If $\mathbf{x}[0] = \mathbf{v}$, the values seen by node x_i during the first $L_i + 1$ time-steps of the linear iteration are all zeros, and so node x_i cannot calculate $\mathbf{Q}\mathbf{v}$ from the outputs of the system (i.e., it cannot distinguish the initial value vector $\mathbf{x}[0] = \mathbf{v}$ from the case where all initial values are zero).

On the other hand, if the row space of $\mathbf{Q}$ is contained in the row space of $\mathcal{O}_{i,L_i}$, one can find a matrix Γ_i such that

$$\Gamma_i \mathcal{O}_{i,L_i} = \mathbf{Q} \ . \tag{2.4}$$

Thus, after running the linear iteration (1.1) for $L_i + 1$ time-steps, node x_i can immediately calculate $\mathbf{Qx}[0]$ as a linear combination of the outputs of the system over those time steps, i.e.,

$$\Gamma_i \begin{bmatrix} \mathbf{y}_i[0] \\ \mathbf{y}_i[1] \\ \vdots \\ \mathbf{y}_i[L_i] \end{bmatrix} = \Gamma_i \mathcal{O}_{i,L_i} \mathbf{x}[0] = \mathbf{Qx}[0] \ . \tag{2.5}$$

This concludes the proof of the lemma. ■

Remark 2.2 *The above lemma shows (via Equation* (2.5)*) how node x_i can obtain the function $\mathbf{Qx}[0]$ after running the linear iteration for $L_i + 1$ time-steps (for some L_i, and assuming that $\mathbf{Q}$ is in the row space of $\mathcal{O}_{i,L_i}$). Note that node x_i does not necessarily have to store the entire set of values $\mathbf{y}_i[0], \mathbf{y}_i[1], \ldots, \mathbf{y}_i[L_i]$ in order to calculate the quantity $\mathbf{Qx}[0]$ via* (2.5)*. Instead, if we partition Γ_i as $\Gamma_i = \begin{bmatrix} \Gamma_{i,0} & \Gamma_{i,1} & \cdots & \Gamma_{i,L_i} \end{bmatrix}$ and allow node x_i to have r extra registers (where r is the number of rows in $\mathbf{Q}$), we can utilize the following recursive strategy: we initialize the values of the r registers with $\bar{x}_i[0] = \Gamma_{i,0}\mathbf{y}_i[0]$ and update them at each time-step k as $\bar{x}_i[k] = \bar{x}_i[k-1] + \Gamma_{i,k}\mathbf{y}_i[k]$. After L_i iterations, $\bar{x}_i[L_i]$ will hold the value of $\mathbf{Qx}[0]$.*

Note that the matrix $\mathcal{O}_{i,L_i}$ in (2.1) resembles the *observability matrix* for the pair $(\mathbf{W}, \mathbf{C}_i)$ (it is exactly the observability matrix when $L_i = N - 1$). As described in Section 1.5, the rank of $\mathcal{O}_{i,L_i}$ is a nondecreasing function of L_i, and bounded above by N. Specifically the rank of $\mathcal{O}_{i,L_i}$ monotonically increases with L_i until $L_i = \nu_i - 1$ (where ν_i is the observability index of the system), at which point it stops increasing. This means that the outputs of the system $\mathbf{y}_i[0], \mathbf{y}_i[1], \ldots, \mathbf{y}_i[\nu_i - 1]$ contain the maximum amount of information that node x_i can possibly obtain about the initial values, and future outputs of the system do not provide any extra information. Recall from Section 1.5 that ν_i can be upper bounded as $\nu_i \leq N - \text{rank}(\mathbf{C}_i) + 1$. Since $\mathbf{C}_i$ has rank $\deg_i + 1$ in the linear iteration model (2.2), the above bound becomes $\nu_i \leq N - \deg_i$. This immediately produces the following fact: if it is possible for node x_i to obtain the necessary information to calculate the linear function $\mathbf{Qx}[0]$ via the linear iteration (1.1), it will require at most $N - \deg_i$ time-steps.

Remark 2.3 *Note that the above discussion only provides a trivial upper bound that is obtained purely from observability theory. We will later tighten this bound to be equal to the size of the largest tree in any spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$ (as described in Section 2.1); we will do this by using results on structured systems (that we derive in the Appendix). On*

the other hand, a lower bound on the number of time-steps required by node x_i can be obtained as follows. Let $\mathcal{F}_i$ denote the set of all nodes whose values are required by node x_i in order to calculate its function (if node x_i's function depends on all initial values, the set $\mathcal{F}_i$ will contain all nodes in the graph). Pick a node $x_j \in \mathcal{F}_i$ that is farthest away from node x_i in the network, and let the distance[2] between node x_j and node x_i be denoted by d_i. Since it takes one time-step for a value to propagate along an edge, it will take at least d_i time-steps before node x_i is able to obtain node x_j's value, and thus a lower bound on the number of time-steps required for function calculation by node x_i is d_i. Note that if $\mathcal{F}_i$ contains all nodes in the network, d_i is simply the eccentricity of node x_i [28]. In particular, if $d_i = N - \deg_i$, then node x_i will take exactly $N - \deg_i$ time-steps to calculate its desired function (since the lower bound and upper bound coincide in this case). Also note that the quantity D_i defined in Theorem 2.1 satisfies $D_i \geq d_i$, since if there is a node x_j at distance d_i from node x_i, then the tree containing x_j in any spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$ must contain at least d_i nodes (i.e., the $d_i - 1$ nodes between x_j and x_i, along with x_j). Thus, the fact that linear iterative strategies require at most D_i time-steps to disseminate information to node x_i does not violate the lower bound d_i. Based on Conjecture 1, it may be the case that D_i is actually a tighter lower bound on the number of time-steps required for node x_i to obtain all initial values, but it is helpful to keep in mind the various constraints on time-complexity that are introduced by the network topology.

If the objective in the system is for some subset of the nodes to calculate the linear function $\mathbf{Q}\mathbf{x}[0]$, one has to choose the weight matrix $\mathbf{W}$ so that $\mathbf{Q}$ is in the row space of the observability matrices for all those nodes. More generally, suppose we require node x_i to calculate the function $g(x_{t_1}[0], x_{t_2}[0], \ldots, x_{t_S}[0])$, where $\{x_{t_1}, x_{t_2}, \ldots x_{t_S}\}$ is some subset of the nodes in the system. If $\mathbf{W}$ can be designed so that the matrix $\mathbf{Q} = \begin{bmatrix} \mathbf{e}_{t_1,N} & \mathbf{e}_{t_2,N} & \cdots & \mathbf{e}_{t_S,N} \end{bmatrix}'$ is in the row space of $\mathcal{O}_{i,\nu_i-1}$, node x_i can recover the initial values $x_{t_1}[0], x_{t_2}[0], \ldots, x_{t_S}[0]$ from $\mathbf{y}_i[0], \mathbf{y}_i[1], \ldots, \mathbf{y}_i[\nu_i - 1]$, and then calculate the desired function of those values. If the pair $(\mathbf{W}, \mathbf{C}_i)$ is observable (i.e., $\text{rank}(\mathcal{O}_{i,\nu_i-1}) = N$), node x_i can uniquely determine the entire initial value vector $\mathbf{x}[0]$ from the outputs of the system and calculate any function of those values. Based on this discussion, we see that we can recast the function calculation problem as a weight matrix design problem, where the objective is to make the row space of the observability matrix for each node contain some appropriate matrix $\mathbf{Q}$. More generally, our goal will be to choose the weight matrix $\mathbf{W}$ to maximize the set of functions that can be calculated by every node, and we describe how to do this in the next section.

[2]Recall that the distance between node x_j and node x_i in a graph is the length of the shortest path between node x_j and node x_i in the graph [28].

2.3 Designing the Weight Matrix

We will begin our discussion of designing the weight matrix for the linear iterative strategy by considering the case of undirected ring networks. We will then generalize our discussion to arbitrary networks, and obtain the proof of Theorem 2.1 (provided in Section 2.1).

2.3.1 Weight Matrix Design for Ring Networks

Consider an undirected ring network with N nodes, where node x_i is connected to nodes $x_{(i \mod N)+1}$ and $x_{((i-2) \mod N)+1}$. In other words, when the nodes are arranged in a circle, each node connects to the nodes immediately to its left and to its right, as shown in Figure 2.5.

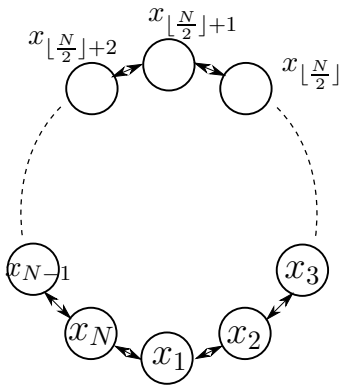

Figure 2.5: Ring Network.

For such networks, we will demonstrate the following result.

Theorem 2.3 *Let $\mathcal{G}$ denote an undirected ring network with N nodes. Suppose that all initial values in the network are elements of a field $\mathbb{F}$, and all weights for the linear iterative strategy are chosen to be nonzero elements of the same field. Then, every node in the network can obtain the initial values of all other nodes after running the linear iterative strategy for exactly $\lfloor \frac{N}{2} \rfloor$ time-steps (performing all operations in the field $\mathbb{F}$).*

Example 1 *Before providing a general proof of the above theorem, it will be helpful to consider a small ring network of $N = 7$ nodes. The weight matrix for such a network is given by*

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & 0 & 0 & 0 & 0 & w_{17} \\ w_{21} & w_{22} & w_{23} & 0 & 0 & 0 & 0 \\ 0 & w_{32} & w_{33} & w_{34} & 0 & 0 & 0 \\ 0 & 0 & w_{43} & w_{44} & w_{45} & 0 & 0 \\ 0 & 0 & 0 & w_{54} & w_{55} & w_{56} & 0 \\ 0 & 0 & 0 & 0 & w_{65} & w_{66} & w_{67} \\ w_{71} & 0 & 0 & 0 & 0 & w_{76} & w_{77} \end{bmatrix}.$$

Consider node x_2 in this network; the values that this node receives at each time-step are given by $\mathbf{y}_2[k] = \mathbf{C}_2\mathbf{x}[k]$, where $\mathbf{C}_2 = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}\end{bmatrix}$. The values seen by node x_2 over $\lfloor \frac{N}{2} \rfloor = 3$ time-steps are given by $\mathbf{y}_2[0:2] = \mathcal{O}_{2,2}\mathbf{x}[0]$, where

$$\mathcal{O}_{2,2} = \begin{bmatrix}\mathbf{C}_2 \\ \mathbf{C}_2\mathbf{W} \\ \mathbf{C}_2\mathbf{W}^2\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & 0 & 0 & 0 & w_{17} \\ w_{21} & w_{22} & w_{23} & 0 & 0 & 0 & 0 \\ 0 & w_{32} & w_{33} & w_{34} & 0 & 0 & 0 \\ * & * & * & 0 & 0 & w_{17}w_{76} & * \\ * & * & * & * & 0 & 0 & * \\ * & * & * & * & w_{34}w_{45} & 0 & 0 \end{bmatrix};$$

in the above matrix, $*$ represents certain (unimportant) polynomials in the weights w_{ij}. Now suppose that all weights are chosen to be nonzero elements of some field $\mathbb{F}$. Then it is easy to see that the above matrix will be guaranteed to be of full column rank over that field. Specifically, the first three columns each contain a "1" in some position, whereas all other columns contain a "0" in that position, and thus the first three columns are guaranteed to be independent of all other columns. Similarly, columns four through seven are guaranteed to be linearly independent of all other columns due to the nonzero entries $w_{34}, w_{34}w_{45}, w_{17}w_{76}$ and w_{17}, respectively. Thus, node x_2 can obtain the initial values of all nodes from the values that it receives over the first 3 time-steps of the linear iteration, regardless of the field over which the iteration is operated.

Based on the intuition provided by the above example, we can now present a proof of Theorem 2.3 for ring networks with an arbitrary number of nodes.

Proof: [Theorem 2.3] The weight matrix for a ring network with N nodes has the form

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & 0 & \cdots & 0 & w_{1N} \\ w_{21} & w_{22} & w_{23} & \cdots & 0 & 0 \\ 0 & w_{32} & w_{33} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & w_{(N-1)(N-1)} & w_{(N-1)N} \\ w_{N1} & 0 & 0 & \cdots & w_{N(N-1)} & w_{NN} \end{bmatrix}$$

where each of the weights is chosen to be a nonzero element of the field $\mathbb{F}$. Since the graph is node-transitive (i.e., the graph "looks the same" from every node), we can focus our analysis on any single node. For convenience, consider node x_2 in the network; the values that this node receives at each time-step are given by $\mathbf{y}_2[k] = \mathbf{C}_2\mathbf{x}[k] = \mathbf{C}_2\mathbf{W}^k\mathbf{x}[0]$, where

$\mathbf{C}_2 = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} \end{bmatrix}$. The observability matrix for node x_2 is given by the matrix $\mathcal{O}_{i,L}$ in (2.3) (with $i = 2$).

To see that the matrix $\mathcal{O}_{2,\lfloor \frac{N}{2} \rfloor - 1}$ will have rank N over the field $\mathbb{F}$, first note that the first three rows of the observability matrix are given by matrix $\mathbf{C}_2$. From the structure of $\mathbf{C}_2$ above, this implies that the first three columns of the observability matrix are guaranteed to be linearly independent of all of the other columns (because each of the first three columns contains a nonzero element in some position, and all other columns have a zero in that position). Now consider the first six rows of the observability matrix, given by

$$\begin{bmatrix} \mathbf{C}_2 \\ \mathbf{C}_2\mathbf{W} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ w_{11} & w_{12} & 0 & 0 & 0 & \cdots & 0 & w_{1N} \\ w_{21} & w_{22} & w_{23} & 0 & 0 & \cdots & 0 & 0 \\ 0 & w_{32} & w_{33} & w_{34} & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

This set of rows introduces a nonzero element in the $(4, N)$ and $(6, 4)$ entries of the observability matrix (given by the nonzero weights w_{1N} and w_{34}, respectively). These elements will cause the fourth and N-th columns of the observability matrix to be linearly independent of all other columns, again because they each contain a nonzero element in some position that is zero in all other columns (recall that we do not have to consider the first three columns anymore, since we have already shown them to be linearly independent of all other columns).

We continue in this way by noting that each set of subsequent rows in the observability matrix is of the form $\mathbf{C}_2\mathbf{W}^k$ for $k = 2, 3, \ldots, \lfloor \frac{N}{2} \rfloor - 1$. The matrix $\mathbf{C}_2$ simply selects the first three rows of $\mathbf{W}^k$, and each entry in $\mathbf{W}^k$ is simply a polynomial in the weights $w_{11}, w_{12}, \ldots, w_{NN}$. Specifically, the (i, j)–th entry of $\mathbf{W}^k$ is a sum of products of the weights on paths of length k from node x_j to node x_i in the graph [28].

For example, in the ring network, there is only one path of length two from x_{N-1} to x_1, and the weights on that path are $w_{N(N-1)}$ and w_{1N}. Thus, the $(1, N-1)$ entry in matrix $\mathbf{W}^2$ will be $w_{N(N-1)}w_{1N}$. The only other nodes that have a path of length two to node x_1 are nodes x_N, x_1, x_2 and x_3 (self-loops are included in the length of the path for the latter three nodes), and thus entries $(7, 4), (7, 5), \cdots, (7, N-2)$ in the observability matrix will be identically zero, and entry $(7, N-1)$ will be $w_{N(N-1)}w_{1N}$. This means that column $N-1$ of the observability matrix will be linearly independent of columns 4 through $N-2$, and since we already know that columns 1 through 4 and column N are linearly independent of all other columns, we see that column $N-1$ is also linearly independent of all other columns.

Similarly, there is only one path of length two from x_5 to x_3, and thus the $(3, 5)$ entry

in matrix $\mathbf{W}^2$ will be $w_{45}w_{34}$. The other nodes that have a path of length two to node x_3 are nodes x_1, x_2, x_3 and x_4, and those are the only entries that will be nonzero in the third row of matrix $\mathbf{W}^2$. Thus, entries $(9,6),(9,7),\cdots,(9,N)$ in the observability matrix will be identically zero, and entry $(9,5)$ will be $w_{45}w_{34}$. This means that the fifth column of the observability matrix will be linearly independent of columns 6 through N, and since we already know that columns 1 through 4 and column N of the observability matrix are linearly independent of all other columns, we see that column 5 is also linearly independent of all other columns.

At this point we have shown that columns $1, 2, 3, 4, 5, N-1$ and N are all guaranteed to be linearly independent of each other and all other columns in the observability matrix. We can continue this reasoning to show that each additional set of rows of the form $\mathbf{C}_2\mathbf{W}^k$ causes two new columns of the observability matrix to be linearly independent of all other columns. At $k = \lfloor \frac{N}{2} \rfloor - 1$, the rows $\mathbf{C}_2\mathbf{W}^{\lfloor \frac{N}{2} \rfloor - 1}$ will cause the final two columns (one column if N is even) to become linearly independent of all other columns, thereby causing the matrix $\mathcal{O}_{2,\lfloor \frac{N}{2} \rfloor - 1}$ to have rank N over $\mathbb{F}$. ■

Remark 2.4 *Note that the diameter of the ring network is $\lfloor \frac{N}{2} \rfloor$, and so this is the minimum possible time required for any protocol to disseminate the initial values to all the nodes. Theorem 2.3 indicates that linear iterative strategies achieve this minimum bound in ring networks even when the nodes operate on values from fields of arbitrary size (including the binary field $\mathbb{F}_2 = \{0,1\}$). It is also of interest to note that one can form a spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$ where the size of the largest tree is equal to $D_i = \lfloor \frac{N}{2} \rfloor$, and this is the minimal size over all possible spanning forests. Thus, the number of time-steps required to disseminate information in ring networks provided by the above theorem agrees with the result in Theorem 2.2 (which was provided in Section 2.1 and which we will prove in the next section).*

Example 2 *Consider a ring network with $N = 5$ nodes, operating over the binary field $\mathbb{F}_2 = \{0,1\}$. Choosing all weights for the linear iteration to be the nonzero element "1", we obtain the weight matrix*

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Consider node x_1 in the network; the values that this node receives at each time-step are

given by

$$\mathbf{y}_i[k] = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}[k] \ .$$

The observability matrix for this node is given by

$$\mathbf{O}_{1,\lfloor \frac{N}{2} \rfloor - 1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix},$$

which has rank 5 *over the field* $\mathbb{F}_2$*. Thus, we can find a matrix* Γ_1 *satisfying* $\Gamma_1 \mathcal{O}_{1,2} = \mathbf{I}_5$ *as*

$$\Gamma_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

and provide this matrix to node x_1*. The procedure can be repeated for all other nodes in the network.*

Now suppose that the initial values of the nodes are $\mathbf{x}[0] = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \end{bmatrix}'$*. After one iteration, the values of the nodes are* $\mathbf{x}[1] = \mathbf{W}\mathbf{x}[0] = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix}'$ *(note that all operations are being performed in the binary field). The values seen by node* x_1 *over these two time-steps are* $\mathbf{y}_1[0] = \mathbf{C}_1\mathbf{x}[0] = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}'$ *and* $\mathbf{y}_1[1] = \mathbf{C}_1\mathbf{x}[1] = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}'$*. Node* x_1 *can now obtain the entire set of initial values as* $\mathbf{x}[0] = \Gamma_1 \begin{bmatrix} \mathbf{y}_1'[0] & \mathbf{y}_1'[1] \end{bmatrix}'$*, and can therefore calculate any desired function of those values. The same holds true for any other node in the network, which also implies that the nodes can reach consensus (if desired) on any function of the initial values after two time-steps.*

While Theorem 2.3 shows that fields of arbitrary size can be used to disseminate information in ring networks, the same conclusion cannot be drawn for more general networks. For example, consider the star network shown in Figure 2.6. For this network, one can verify that for *any* choice of weights from the binary field $\mathbb{F}_2$, the observability matrix for node x_1 will *not* be of full column rank, and thus node x_1 *cannot* obtain the initial values of all nodes via a linear iterative strategy; one requires a field of size at least 3 in order to enable x_1 to obtain the values. While it is difficult to explicitly characterize the minimum field

size required for any arbitrary network, we will now derive a lower bound on the probability that weights chosen at random from the field $\mathbb{F}_q$ (of sufficiently large size q) will maximize the rank of the observability matrix for every node in the network.

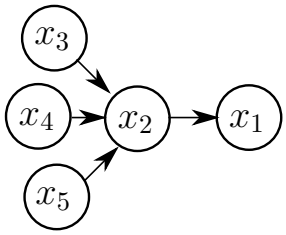

Figure 2.6: Star Network.

2.3.2 Weight Matrix Design for Arbitrary Networks

Recall from Section 2.2 that choosing $\mathbf{W}$ to maximize the set of calculable functions for any node x_i in an arbitrary network essentially amounts to maximizing the rank of the observability matrix for x_i. To design a weight matrix that achieves this, recall the set

$$\mathcal{S}_i = \{x_j | \text{ There exists a path from } x_j \text{ to } x_i \text{ in } \mathcal{G}\} \cup \{x_i\}$$

(originally defined in Theorem 2.1), and let $\bar{\mathcal{S}}_i$ denote its complement ($\mathcal{G}$ is the graph of the network). Let $\mathbf{x}_{\mathcal{S}_i}[k]$ denote the vector that contains the values of the nodes in $\mathcal{S}_i$, and let $\mathbf{x}_{\bar{\mathcal{S}}_i}[k]$ denote the vector of values for the remaining nodes. Without loss of generality, assume that the vector $\mathbf{x}[k]$ in (2.2) has the form $\mathbf{x}[k] = \begin{bmatrix}\mathbf{x}'_{\mathcal{S}_i}[k] & \mathbf{x}'_{\bar{\mathcal{S}}_i}[k]\end{bmatrix}'$ (the vector can always be arranged in this form by an appropriate permutation of the node indices). Since there is no path from any node in $\bar{\mathcal{S}}_i$ to node x_i (and hence no path from any node in $\bar{\mathcal{S}}_i$ to any node in $\mathcal{S}_i$), Equation (2.2) takes the form

$$\begin{aligned}\begin{bmatrix}\mathbf{x}_{\mathcal{S}_i}[k+1]\\ \mathbf{x}_{\bar{\mathcal{S}}_i}[k+1]\end{bmatrix} &= \begin{bmatrix}\mathbf{W}_{i,\mathcal{S}_i} & \mathbf{0}\\ \mathbf{W}_{i,\mathcal{S}_i\bar{\mathcal{S}}_i} & \mathbf{W}_{i,\bar{\mathcal{S}}_i}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathcal{S}_i}[k]\\ \mathbf{x}_{\bar{\mathcal{S}}_i}[k]\end{bmatrix}\\ \mathbf{y}_i[k] &= \begin{bmatrix}\mathbf{C}_{i,\mathcal{S}_i} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathcal{S}_i}[k]\\ \mathbf{x}_{\bar{\mathcal{S}}_i}[k]\end{bmatrix} \quad . \end{aligned} \tag{2.6}$$

The outputs seen by node x_i over the first L_i+1 time-steps of the linear iteration are given by

$$\begin{bmatrix}\mathbf{y}_i[0]\\ \mathbf{y}_i[1]\\ \vdots\\ \mathbf{y}_i[L_i]\end{bmatrix} = \begin{bmatrix}\mathbf{C}_{i,\mathcal{S}_i} & \mathbf{0}\\ \mathbf{C}_{i,\mathcal{S}_i}\mathbf{W}_{i,\mathcal{S}_i} & \mathbf{0}\\ \vdots & \vdots\\ \mathbf{C}_{i,\mathcal{S}_i}\mathbf{W}_{i,\mathcal{S}_i}^{L_i} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{S_i}[0]\\ \mathbf{x}_{\bar{\mathcal{S}}_i}[0]\end{bmatrix} = \begin{bmatrix}\mathbf{C}_{i,\mathcal{S}_i}\\ \mathbf{C}_{i,\mathcal{S}_i}\mathbf{W}_{i,\mathcal{S}_i}\\ \vdots\\ \mathbf{C}_{i,\mathcal{S}_i}\mathbf{W}_{i,\mathcal{S}_i}^{L_i}\end{bmatrix}\mathbf{x}_{S_i}[0] \quad . \tag{2.7}$$

This shows that node x_i receives no information about any node in the set $\bar{\mathcal{S}}_i$, regardless of the choice of weight matrix, and therefore cannot calculate any function of the node values from that set.[3] Furthermore, the matrix multiplying $\mathbf{x}_{\mathcal{S}_i}[0]$ is the observability matrix for the pair $(\mathbf{W}_{i,\mathcal{S}_i}, \mathbf{C}_{i,\mathcal{S}_i})$. Thus, maximizing the rank of the observability matrix for node x_i is equivalent to maximizing the rank of the observability matrix for the pair $(\mathbf{W}_{i,\mathcal{S}_i}, \mathbf{C}_{i,\mathcal{S}_i})$.

To choose a set of weights that accomplishes this, we first note that matrix $\mathbf{W}_{i,\mathcal{S}_i}$ is a structured matrix (since each entry is either identically zero, or an independent free parameter from the field $\mathbb{F}$; see the Appendix and Section 1.5.1 for a discussion of structured systems). We can now use the result on structured system observability over finite fields (Theorem A.3 in Section A.2) to prove the key result in Theorem 2.1 (introduced at the end of Section 2.1).

Proof: [Theorem 2.1] From (2.7), we see that the output available to node x_i after $L_i + 1$ time-steps is simply the observability matrix for the pair $(\mathbf{W}_{i,\mathcal{S}_i}, \mathbf{C}_{i,\mathcal{S}_i})$ multiplied by the vector $\mathbf{x}_{\mathcal{S}_i}[0]$. To prove the theorem, we have to show that the pair $(\mathbf{W}_{i,\mathcal{S}_i}, \mathbf{C}_{i,\mathcal{S}_i})$ will be observable with a certain probability if the weights for the linear iteration are chosen independently and uniformly from the field $\mathbb{F}_q$. To accomplish this, we examine the graph $\mathcal{H}$ associated with the matrix $\mathbf{W}_{i,\mathcal{S}_i}$. In this case, $\mathcal{H}$ is obtained as follows:

1. Take the subgraph of $\mathcal{G}$ induced by the nodes in $\mathcal{S}_i$.

2. In this subgraph, add a self-loop to every node to correspond to the free parameters w_{jj} on the diagonal of the matrix $\mathbf{W}_{i,\mathcal{S}_i}$.

Every node in $\mathcal{H}$ has a path to node x_i (by definition of the set $\mathcal{S}_i$), and furthermore, every node has a self-loop, which satisfies the conditions in Theorem A.3. This implies that, with probability at least $1 - \frac{1}{q}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$, the observability matrix multiplying $\mathbf{x}_{\mathcal{S}_i}[0]$ in (2.7) will have full column rank and the observability index will be at most D_i. Node x_i can therefore recover the vector $\mathbf{x}_{\mathcal{S}_i}[0]$ from the outputs of the system over at most D_i time-steps. ■

Remark 2.5 *Note that the probability bound obtained in the above theorem is potentially quite loose because of several conservative assumptions in the proof of Theorem A.3. For example, that proof uses the Schwartz-Zippel lemma to bound the probability of a given polynomial being zero after a random choice of parameters; while this lemma is quite convenient to state and use, the probability bound can be tightened by using more precise measures of the probability of a given polynomial being nonzero with random choices of parameters (such as the bound derived in [55]). The actual size of the field required might be much smaller than that specified by Theorem A.3 (as we saw for undirected ring networks earlier in this section). The main contribution of the above result is to show that there always exists a*

[3] Note that this is not surprising, since there is no path in the network from any node in the set $\bar{\mathcal{S}}_i$ to node x_i.

finite field of sufficiently large size over which linear iterative strategies can be applied, and furthermore, to show that (with high probability) node x_i can recover all initial values after at most D_i time-steps.

Theorem 2.1 indicates that node x_i can calculate any function of the initial values of nodes that have a path to node x_i, since it can reconstruct those initial values after running the linear iteration for a finite number of time-steps. Note that this is the most general result that can be obtained for a given (time-invariant) network, since if a node x_j does not have a path to node x_i in the network, it will be impossible for node x_i to calculate any function of node x_j's value (regardless of the protocol). Since the above theorem holds for any node x_i, we can now prove Theorem 2.2 (also provided in Section 2.1).

Proof: [Theorem 2.2] Let $\mathcal{O}_i$ denote the observability matrix for node x_i after choosing the weight matrix as specified in the statement of Theorem 2.2. Note from the proof of Theorem 2.1 that the probability that the observability matrix for any node x_i is not full column rank is upper bounded by $\frac{1}{q}(D_i-1)(|\mathcal{S}_i|-\deg_i-\frac{D_i}{2})$. Thus the probability that at least one node x_i cannot obtain the initial value of some node that has a path to it in the network is upper bounded by

$$\Pr\left[\bigcup_{i=1}^{N}\{\text{rank}(\mathcal{O}_i)<|\mathcal{S}_i|\}\right]\leq\sum_{i=1}^{N}\Pr[\text{rank}(\mathcal{O}_i)<|\mathcal{S}_i|]\leq\sum_{i=1}^{N}\frac{1}{q}(D_i-1)(|\mathcal{S}_i|-\deg_i-\frac{D_i}{2}) \ ,$$

where we have used the union bound to obtain the first inequality. The probability that every node $x_i\in\mathcal{X}$ can obtain the initial values of all nodes in $\mathcal{S}_i$ is thus given by

$$\begin{aligned}\Pr\left[\bigcap_{i=1}^{N}\{\text{rank}(\mathcal{O}_i)=|\mathcal{S}_i|\}\right]&=1-\Pr\left[\bigcup_{i=1}^{N}\{\text{rank}(\mathcal{O}_i)<|\mathcal{S}_i|\}\right]\\&\geq 1-\sum_{1=1}^{N}\frac{1}{q}(D_i-1)(|\mathcal{S}_i|-\deg_i-\frac{D_i}{2}) \ .\end{aligned}$$

Finally, note that if the observability matrix is of rank $|\mathcal{S}_i|$ for some pair $(\mathbf{W},\mathbf{C}_i)$, then it will achieve this rank after at most D_i time-steps with probability $1-\frac{1}{q}(D_i-1)(|\mathcal{S}_i|-\deg_i-\frac{D_i}{2})$ (as discussed in Theorem 2.1). One can easily show (by following the same reasoning as above) that the observability index for every node x_i will be at most D_i with probability at least $1-\sum_{1=1}^{N}\frac{1}{q}(D_i-1)(|\mathcal{S}_i|-\deg_i-\frac{D_i}{2})$. This concludes the proof of the theorem. ■

For the special case when all nodes are required to reach consensus (i.e., they are all required to calculate the same function of the initial values), the above results produce the following corollary.

Corollary 2.2 *Define the set $\mathcal{S}=\cap_{i=1}^{N}\mathcal{S}_i$, where*

$$\mathcal{S}_i=\{x_j|\ \textit{There exists a path from } x_j \textit{ to } x_i \textit{ in } \mathcal{G}\}\cup\{x_i\} \ .$$

In other words, $\mathcal{S}$ is the set of nodes that have a path to all nodes in the system. For each $x_i \in \mathcal{X}$, consider a subgraph $\mathcal{H}_i$ of $\mathcal{G}$ that is a $\mathcal{S}_i$-spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$, with the property that the size of the largest tree of the forest is minimal over all $\mathcal{S}_i$-spanning forests rooted at $\{x_i\} \cup \mathcal{N}_i$. Let D_i denote the size of the largest tree of $\mathcal{H}_i$, and let $D = \max_i D_i$. Suppose that the weights for the linear iteration are chosen randomly (independently and uniformly) from a field $\mathbb{F}_q$ of size $q \geq \sum_{i=1}^{N}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$. Then, with probability at least $1 - \frac{1}{q}\sum_{i=1}^{N}(D_i - 1)(|\mathcal{S}_i| - \deg_i - \frac{D_i}{2})$, all nodes can reach consensus on any function of the initial values of nodes in $\mathcal{S}$ after running the linear iteration for at most D time-steps.

Corollary 2.2 indicates that the nodes can reach consensus after a finite number of iterations as long as the network has at least one node that has a path to every other node. If the network is strongly connected (i.e., there is a path from every node to every other node), the nodes can reach consensus on any arbitrary function of the initial values. This result is more general than those currently existing in the literature (for time-invariant networks), which typically focus on the nodes reaching *asymptotic* consensus on a real-valued *linear function* of the initial values (e.g., see [6, 7, 22], and the discussion in Chapter 1).

2.4 Decentralized Calculation of Observability Matrices

In the previous sections, we saw that if the weight matrix is chosen appropriately, the observability matrix for each node will contain enough information for that node to calculate any desired function of the initial values. Specifically, there will exist a matrix Γ_i satisfying (2.4) for each node x_i, and node x_i can use this matrix to calculate $\mathbf{Q}\mathbf{x}[0]$ from (2.5). If required, it can then calculate more general (nonlinear) functions of $\mathbf{Q}\mathbf{x}[0]$. However, finding Γ_i requires knowledge of the observability matrix $\mathcal{O}_{i,\nu_i-1}$. If the global topology of the graph and all of the weights are known *a priori*, then Γ_i can be calculated from the matrix $\mathcal{O}_{i,\nu_i-1}$ and conveyed to node x_i. Note that in graphs with time-invariant topologies, Γ_i only has to be computed once for each node x_i. However, in practice, it may be the case that there is no opportunity to calculate the Γ_i's *a priori*, and therefore it will be necessary for the nodes to calculate these matrices using only local information. In this section, we show how each node x_i can calculate Γ_i in a decentralized manner. To accomplish this, we will assume that the nodes know N (or an upper bound for N). We will also assume that each node has a unique identifier, and that the nodes know their position in an appropriate ordering of the identifiers (e.g., the node with the j–th smallest identifier takes its position to be j). We assume without loss of generality that the vector $\mathbf{x}[k]$ is consistent with this ordering (i.e., $x_i[k]$ is the value of the node whose position is i–th in the ordering).

As noted in the previous section, if the nodes choose their own weights independently and uniformly from a field of appropriately large size, the observability matrix for each node

will be of maximum possible rank with high probability. Specifically, from Corollary 2.1, if the nodes choose their own weights from a field of size $q \geq \frac{N^2(N-1)}{2}$, then the observability matrix for each node will be of maximum rank with probability at least $1 - \frac{N^2(N-1)}{2q}$. If the nodes are told beforehand to choose q of a certain form (say $q = 2^m$, for some m, and with a certain primitive polynomial for each m) to ensure a certain probability of success, the nodes will be able to choose the field $\mathbb{F}_q$ based on N. Once the weights are chosen (randomly) from this field, suppose the nodes perform N runs of the linear iteration, each for $N-1$ time-steps. For the j–th run, node x_j sets its initial condition to be "1", and all other nodes set their initial conditions to be zero. In other words, if $\mathbf{x}_{*,j}[k]$ denotes the vector of node values during the k–th time-step of the j–th run, the nodes calculate $\mathbf{x}_{*,j}[k+1] = \mathbf{W}\mathbf{x}_{*,j}[k]$, $0 \leq k \leq N-2$, $1 \leq j \leq N$, where $\mathbf{x}_{*,j}[0] = \mathbf{e}_{j,N}$. Suppose that for each of the N runs, node x_i stores the values it sees over the first $N - \deg_i$ time-steps (since each node knows that $D_i \leq N - \deg_i$). Each node x_i then has access to the matrix

$$\Psi_{i,L} = \begin{bmatrix} \mathbf{y}_{i,1}[0] & \mathbf{y}_{i,2}[0] & \cdots & \mathbf{y}_{i,N}[0] \\ \mathbf{y}_{i,1}[1] & \mathbf{y}_{i,2}[1] & \cdots & \mathbf{y}_{i,N}[1] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{i,1}[L] & \mathbf{y}_{i,2}[L] & \cdots & \mathbf{y}_{i,N}[L] \end{bmatrix} , \tag{2.8}$$

for any $0 \leq L \leq N - \deg_i - 1$, where $\mathbf{y}_{i,j}[k] = \mathbf{C}_i \mathbf{x}_{*,j}[k]$. Using (2.3), the above matrix can be written as

$$\Psi_{i,L} = \mathcal{O}_{i,L} \begin{bmatrix} \mathbf{x}_{*,1}[0] & \mathbf{x}_{*,2}[0] & \cdots & \mathbf{x}_{*,N}[0] \end{bmatrix} ,$$

and since $\mathbf{x}_{*,j}[0] = \mathbf{e}_{j,N}$, we see that $\Psi_{i,L} = \mathcal{O}_{i,L}$. Node x_i now has access to its observability matrix, and can find the matrix Γ_i and the smallest integer L_i such that $\Gamma_i \mathcal{O}_{i,L_i} = \mathbf{Q}$ (for the desired matrix $\mathbf{Q}$).[4] For future initial conditions $\mathbf{x}[0]$, node x_i can perform function calculation in the network by running the linear iteration (1.1) for $L_i + 1$ time-steps to obtain the values $\mathbf{y}_i[0], \mathbf{y}_i[1], \ldots, \mathbf{y}_i[L_i]$. It can then immediately obtain $\mathbf{Q}\mathbf{x}[0]$ from (2.5), and use this to calculate any function $g(\mathbf{Q}\mathbf{x}[0])$ (note that g can be a completely arbitrary function of all initial values if $\mathbf{Q} = \mathbf{I}_N$).

It is important to note that the above discovery algorithm only needs to be run once (in time-invariant networks) in order for the nodes to obtain the Γ_i matrices. After they have obtained these matrices, they can use them to perform function calculation for arbitrary sets of initial conditions. In other words, the cost of discovering the Γ_i matrices will be amortized over the number of times they are used by the nodes to perform function calculation. A full description of the algorithm (including the initialization phase) can be found in Figure 2.7 and Figure 2.8.

[4]Note that this integer L_i will be no greater than the quantity $D_i - 1$ defined in Corollary 2.1 with probability at least $1 - \frac{N^2(N-1)}{2q}$.

Initialization Phase: Calculation of Network Parameters

INPUT: Network $\mathcal{G}$, with N nodes $x_1, x_2, \ldots, x_N$, each with an ID and an associated function $g_i : \mathbb{F}^N \to \mathbb{F}^{r_i}$.

1: Each node determines the smallest field of the specified type (e.g., $\mathbb{F}_{2^m}$ for some m and an associated primitive polynomial) such that the observability matrix for each node is guaranteed to have maximum rank with probability at least equal to some pre-specified amount (say, 95%). Each node x_i independently chooses a set of weights w_{ij} for the nodes in its neighborhood, and a weight w_{ii} for itself, from a uniform distribution on this field.
2: **for** $j = 1$ **to** N
3: $x_j[0] = 1$, $x_i[0] = 0$ for all $i \neq j$.
4: **for** $k = 0$ **to** N-2
5: Each node x_i updates its value as

$$x_i[k+1] = w_{ii} x_i[k] + \sum_{l \in \mathcal{N}_i} w_{il} x_l[k].$$

6: Each node x_i stores $\mathbf{y}_{i,j}[k] = \mathbf{C}_i \mathbf{x}[k]$.
7: **end for**
8: **end for**
9: Each node x_i forms the matrix Ψ_{i,L_i} in (2.8) and finds a value L_i and a matrix Γ_i satisfying $\Gamma_i \Psi_{i,L_i} = \mathbf{Q}_i$, for an appropriate matrix $\mathbf{Q}_i$ which would allow node x_i to calculate $g_i(x_1[0], x_2[0], \ldots, x_N[0])$.
10: All nodes use a simple distributed protocol to determine $\max_{1 \leq i \leq N} L_i$.

OUTPUT: Γ_i, L_i and weights w_{ij} for each node x_i, and $\max_{1 \leq i \leq N} L_i$.

Figure 2.7: The initialization phase of the protocol. This phase is used by the nodes to distributively determine the necessary information about the network in order to later perform distributed function calculation.

Remark 2.6 *If the nodes only know an upper bound for N, the observability matrix obtained by each node will have one or more columns that are entirely zero. This is because there will be some runs of the linear iteration where all nodes set their initial values to be zero, under the assumption that some (nonexistent) node will be setting its value to* 1. *In such cases, the nodes can simply drop these zero columns from their observability matrices, and continue as normal.*

Remark 2.7 *Note that the protocol described above for finding the observability matrix requires each node x_i to store $(\deg_i + 1) \times N \times (N - \deg_i)$ values, because each output vector $\mathbf{y}_{i,j}[k]$ has $\deg_i + 1$ components, there are $N - \deg_i$ outputs stored per run, and there are N runs in total. However, the observability index ν_i for node x_i will most likely be upper bounded by D_i (which may be much smaller than $N - \deg_i$), and so node x_i may be storing more values than required (since after time-step ν_i, the rows of the observability matrix are linear combinations of the previous rows). This problem can be easily circumvented by having the nodes run the N linear iterations in parallel, as opposed to serially. In this*

Calculating Functions of Arbitrary Initial Conditions

INPUT: Network $\mathcal{G}$, with N nodes $x_1, x_2, \ldots, x_N$, each with an (arbitrary) initial value $x_i[0]$ and an associated function $g_i : \mathbb{F}^N \to \mathbb{F}^{r_i}$. Each node x_i knows Γ_i, L_i, weights w_{ij} and $\max_{1 \leq j \leq N} L_j$.

1: **for** $k = 0$ **to** $\max_{1 \leq j \leq N} L_j$

2: Each node x_i updates its value as

$$x_i[k+1] = w_{ii} x_i[k] + \sum_{l \in \mathcal{N}_i} w_{il} x_l[k].$$

3: Each node x_i stores $\mathbf{y}_i[k] = \mathbf{C}_i \mathbf{x}[k]$.

4: **if** $k == L_i$ for some i **then**

5: node x_i calculates

$$\Gamma_i \begin{bmatrix} \mathbf{y}_i[0] \\ \mathbf{y}_i[1] \\ \vdots \\ \mathbf{y}_i[L_i] \end{bmatrix} = \mathbf{Q}_i \mathbf{x}[0] \ ,$$

which it uses to calculate $g_i(x_1[0], x_2[0], \ldots, x_N[0])$.

6: **end if**

7: **end for**

OUTPUT: $g_i(x_1[0], x_2[0], \ldots, x_N[0])$ for each node x_i.

Figure 2.8: The protocol to perform distributed function calculation via linear iterations. The inputs to this protocol can be obtained by running the initialization phase (given in Figure 2.7), or can be calculated by a centralized entity and provided to each node x_i.

way, the nodes construct the (observability) matrix $\Psi_{i,L}$ in (2.8) row-by-row, as opposed to column-by-column. If node x_i finds that the new rows of the observability matrix do not increase its rank, or if the existing rows of the observability matrix already contain the desired matrix $\mathbf{Q}$, node x_i can stop storing values. Note, however, that all nodes still have to complete $N-1$ time-steps of each run in order to ensure that other nodes have an opportunity to find their observability matrices. This slight modification of the protocol would require each node to store at most $(\deg_i + 1) \times N \times \nu_i$ values.

The communication cost incurred by the above protocol for discovering the observability matrix can be characterized as follows. Since each node transmits a single value on each outgoing edge at each time-step, and since there are $N-1$ time-steps per run, with N runs in total, each node x_i will have to transmit $N(N-1)$ out-$\deg_i$ messages in order to run this protocol. Summing over all nodes in the network, there will be $\sum_{i=1}^{N} N(N-1)$ out-$\deg_i = N(N-1)|\mathcal{E}|$ messages transmitted in total (since $\sum_{i=1}^{N}$ out-$\deg_i$ is equal to the number of edges in the graph [28]). Note that in wireless networks, a single transmission by node x_i will be equivalent to communicating a value along each outgoing edge of node x_i (since all neighbors of node x_i will receive the message after just one transmission), and thus each

node would only have to transmit $N(N-1)$ messages in order to run the protocol, for a total of $N^2(N-1)$ messages in the network.

Remark 2.8 *It may be the case that the number of time-steps required by each node to calculate its desired function will vary from node to node (i.e., the value of L_i might be different for different i's). There are several different ways to handle this. One option is to have the nodes run the linear iteration for a full $N-1$ time-steps, even though they can calculate their desired function after L_i+1 time-steps (similar to what was suggested in Remark 2.7). However, this could cause the linear iteration to run for more time-steps than necessary (i.e., for $N-1$ time-steps, instead of $\max_i L_i+1$ time-steps). To get around this, one can have the nodes calculate the maximum of all the L_i's after they have calculated their Γ_i matrices. This can be done via a simple protocol where each node starts by broadcasting its own value of L_i, and then subsequently maintains and broadcasts only the largest value it receives from a neighbor. After at most N time-steps, all nodes will have the maximum value of L_i [11], and then the nodes can run subsequent linear iterations for $\max_i L_i+1$ time-steps.*

2.5 Comparison to Previous Results on Linear Iterative Strategies

As discussed in Chapter 1, the vast majority of the existing literature on linear iterative strategies has focused on their ability to produce asymptotic consensus on a linear function (typically the average) of the initial values. The topic of finite-time consensus via linear iterations has received only limited attention in the literature. Specifically, it was briefly discussed by Kingston and Beard in [56], but the method described in that paper requires the network to be fully connected (i.e., every node needs to be directly connected to every other node) for at least one time-step, which is a very strict condition. Finite-time consensus was also studied for continuous-time systems in [57]. The approach in that paper was to have the nodes evolve according to the gradient of a suitably defined function of their values. The resulting protocol, which is nonlinear, does not directly translate to discrete-time systems, and the author of [57] only considered a restricted class of possible consensus values. In contrast to the above works, our method can be used in networks with arbitrary topologies, achieves finite-time consensus in discrete-time systems by running a simple linear iteration, and allows the consensus value to be any function of the node values. In fact, our method allows the nodes to calculate arbitrary (and different) functions of the initial values, and thus is much more general than allowing the nodes to simply reach consensus.

Another benefit of the method developed in this chapter is that it allows linear iterative strategies to be performed over finite fields. This is an important feature for many practical applications, such as in networks that contain bandwidth restrictions in the transmission

channels between nodes, or in networks with nodes that are limited in the precision of the computations that they perform. The topic of *quantized consensus* (where the values in the network are quantized to lie in some discrete set) has recently started to receive attention in the control literature [26, 58, 59, 60, 61]. For example, the authors of [26] propose a *gossip*-based protocol to reach quantized consensus on the average of the initial values (which are assumed to be integers). In their protocol, each node periodically chooses one of its neighbors at random; these two nodes then either bring their values as close together as possible (while keeping the sum of the two values constant), or swap their values if the difference of their values is equal to 1. The analysis in [26] revealed that this strategy will eventually cause the values of all nodes in the network to be within 1 of the average of the initial values. The expected convergence time of this (nonlinear) scheme is finite, but difficult to characterize explicitly; for fully connected networks, the authors of [26] show that the expected convergence time is lower bounded by $\frac{N(N-1)}{2}$ (where N is the number of nodes in the network) and upper bounded by $O(N^3)$. Other works have also studied ways to obtain consensus by incorporating quantization into variants of a linear iterative strategy, where nodes update their values as a linear combination of quantized versions of their neighbors' values [59, 60, 61].

The topic of information dissemination (as opposed to consensus) via a gossip algorithm with *random network coding* has also been investigated in [19, 62]. In these schemes, every node in the network periodically sends a random linear combination of the messages that it has previously received to a randomly chosen neighbor. The analysis of such a protocol (for complete graphs in [19] and for more general graphs in [62]) is complex due to the probabilistic nature of the algorithm, but the authors of these works provide bounds on the expected value of the number of gossip rounds required in order for all of the nodes to obtain all of the information.

In comparison to the above works that focus on consensus and gossip-based information dissemination, our approach has several benefits. First, our work provides an algorithm for disseminating information where multiple nodes in the network are allowed to simultaneously exchange information (rather than a gossip-based algorithm where only two random nodes exchange information at each round). Thus, our analysis provides a guaranteed upper bound (no greater than the number of nodes in the network, and most likely equal to the size of the largest tree in a spanning forest of the graph) on the number of time-steps required for every node to obtain the initial values of the other nodes. In ring networks, our results show that no other protocol can disseminate information faster than the linear iterative strategy, and that this can be accomplished by using as few as two quantization levels (i.e., by performing operations over the binary field $\mathbb{F}_2$). This is in contrast to the work on quantized consensus and gossip-based network coding, where the expected convergence time can be much larger than the number of nodes in the network (as described above and in [26, 60, 19, 62]). Second, our approach allows the nodes to obtain *all* of the initial

values in a strongly connected network, and thus allows them to calculate *any* function of the initial values; in contrast, the other works on quantized consensus only focus on having all nodes calculate the same function (typically an approximation to the average of the initial values) [26, 59, 60, 61]. Third, our protocol only requires the use of linear operations at every step, unlike the above quantized consensus protocols that incorporate nonlinear quantization operations.

2.6 Example

Consider the network introduced at the beginning of the chapter in Figure 2.3. The objective in this network is for every node to obtain the initial value of every other node. The nodes do not know the entire topology, and there is no centralized decision maker to calculate the Γ_i matrices for each node. However, suppose that all nodes know that there are $N = 5$ nodes in the system.

The first step is for the nodes to choose their weights. Since the graph is strongly connected, Corollary 2.1 indicates that if the nodes independently pick weights from a uniform distribution on a field $\mathbb{F}_q$ of size $q \geq \frac{N^2(N-1)}{2}$, then the observability matrix for every node will have rank N with probability at least $1 - \frac{N^2(N-1)}{2q}$. Suppose that the nodes agree beforehand that the field size should be $q = 2^m$ for the smallest m that ensures at least a 95% chance of success (i.e., $1 - \frac{N^2(N-1)}{2(2^m)} \geq 0.95$). In this case, the nodes use $m = 10$ with primitive polynomial[5] $\alpha^{10} + \alpha^3 + 1$, which produces a probability of success at least 95.12%. Each node then chooses its weights independently and uniformly from the field $\mathbb{F}_{2^{10}}$, which produces the weight matrix

$$\mathbf{W} = \begin{bmatrix} 282 & 509 & 0 & 981 & 0 \\ 695 & 981 & 260 & 0 & 0 \\ 0 & 348 & 517 & 0 & 833 \\ 0 & 0 & 715 & 152 & 249 \\ 0 & 0 & 0 & 263 & 950 \end{bmatrix}.$$

Note that each entry of the above weight matrix is the decimal representation of the appropriate field element. For example, the quantity "282" is used to represent the field element $\alpha^8 + \alpha^4 + \alpha^3 + \alpha$, where α is a root of the primitive polynomial $\alpha^{10} + \alpha^3 + 1$.

Next, the nodes use the protocol described in Section 2.4 to discover their observability matrices. Specifically, they run $N = 5$ different linear iterations, with initial condition $\mathbf{x}_{*,j}[0] = \mathbf{e}_{j,5}$ for the j–th linear iteration. As discussed in Remark 2.7, we will have the nodes run the N different linear iterations in parallel, so that they compute the observability

[5]This is needed to ensure that all nodes have a consistent representation of the field elements; to perform the calculations for this example, we use the **gf** command in MATLAB, which produces a Galois field with the specified primitive polynomial.

matrix row-by-row. For instance, consider node x_4 in Figure 2.3. The matrix $\mathbf{C}_4$ in (2.2) is given by

$$\mathbf{C}_4 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} .$$

After running the $N = 5$ different linear iterations for 3 time-steps each, node x_4 has access to the matrix

$$\begin{bmatrix} \mathbf{y}_{4,1}[0] & \mathbf{y}_{4,2}[0] & \mathbf{y}_{4,3}[0] & \mathbf{y}_{4,4}[0] & \mathbf{y}_{4,5}[0] \\ \mathbf{y}_{4,1}[1] & \mathbf{y}_{4,2}[1] & \mathbf{y}_{4,3}[1] & \mathbf{y}_{4,4}[1] & \mathbf{y}_{4,5}[1] \\ \mathbf{y}_{4,1}[2] & \mathbf{y}_{4,2}[2] & \mathbf{y}_{4,3}[2] & \mathbf{y}_{4,4}[2] & \mathbf{y}_{4,5}[2] \end{bmatrix} = \left[\begin{array}{rrrrr} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 348 & 517 & 0 & 833 \\ 0 & 0 & 715 & 152 & 249 \\ 0 & 0 & 0 & 263 & 950 \\ \hline 526 & 972 & 661 & 908 & 420 \\ 0 & 15 & 45 & 1009 & 169 \\ 0 & 0 & 577 & 322 & 254 \end{array}\right] ,$$

where $\mathbf{y}_{4,j}[k]$ is the output seen by node x_4 at time-step k of the j–th run. At this point, node x_4 finds that this matrix is of full column rank, and so it does not need to store any further values. However, it continues to perform the linear iterations for 1 more time-step (for a total of $N - 1 = 4$ time-steps), so that the observability matrix for every node in the system will be of full column rank with at least a probability of 95%.

After all N runs are complete, each node x_i has access to its observability matrix, and calculates the matrix Γ_i satisfying $\Gamma_i \mathcal{O}_{i,L_i} = \mathbf{I}_5$. For example, node x_4 obtains

$$\Gamma_4 = \begin{bmatrix} 630 & 735 & 444 & 480 & 570 & 301 & 484 & 286 & 777 \\ 191 & 858 & 314 & 817 & 850 & 694 & 0 & 826 & 898 \\ 811 & 664 & 73 & 314 & 96 & 534 & 0 & 384 & 88 \\ 664 & 517 & 290 & 974 & 815 & 860 & 0 & 489 & 736 \\ 73 & 290 & 752 & 675 & 872 & 377 & 0 & 543 & 244 \end{bmatrix} .$$

We omit the values of the Γ_i matrices for the other nodes in the interest of space. In this example, it is found that $L_1 = 1, L_2 = 1, L_3 = 1, L_4 = 2, L_5 = 3$; one can readily verify that $L_i = D_i - 1$ (where D_i is the size of the largest tree in the optimal spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$). At this point, the nodes find $\max_i L_i = 3$ via the protocol discussed in Remark 2.8, and so they all agree to stop running subsequent linear iterations after $\max_i L_i + 1 = 4$ time-steps.

Now that each node x_i has access to Γ_i, it can use this matrix to reconstruct any arbitrary set of initial values. Suppose that on the next run, the initial values on the

nodes are given by $\mathbf{x}[0] = \begin{bmatrix} 426 & 50 & 923 & 966 & 502 \end{bmatrix}'$. In order to disseminate these values, the nodes run the linear iteration for four time-steps. The values of the nodes over those time-steps are given by

$$\begin{bmatrix} \mathbf{x}[0] & \mathbf{x}[1] & \mathbf{x}[2] & \mathbf{x}[3] \end{bmatrix} = \begin{bmatrix} 426 & 471 & 498 & 575 \\ 50 & 871 & 945 & 971 \\ 923 & 755 & 876 & 714 \\ 966 & 539 & 310 & 770 \\ 502 & 916 & 618 & 10 \end{bmatrix} .$$

Using the outputs of the system $\mathbf{y}_4[0] = \mathbf{C}_4\mathbf{x}[0], \mathbf{y}_4[1] = \mathbf{C}_4\mathbf{x}[1], \mathbf{y}_4[2] = \mathbf{C}_4\mathbf{x}[2]$ and $\mathbf{y}_4[3] = \mathbf{C}_4\mathbf{x}[3]$, node x_4 can now reconstruct the initial values as

$$\Gamma_4 \begin{bmatrix} \mathbf{y}_4[0] \\ \mathbf{y}_4[1] \\ \mathbf{y}_4[2] \end{bmatrix} = \begin{bmatrix} 426 & 50 & 923 & 966 & 502 \end{bmatrix}' .$$

It can then use these values to calculate any desired function of the initial values. All other nodes obtain the initial values in the same manner, and the linear iterative strategy stops after 4 time-steps.

2.7 Summary

We showed that in any given time-invariant connected network, if the weights for the linear iteration are chosen randomly (independently and uniformly) from a field of sufficiently large size, then with some nonzero probability (that increases with the size of the field), every node can obtain the initial values of any other node in the network. Furthermore, with the same probability, the number of time-steps required by any node to recover these values is upper bounded by the size of the largest tree in any spanning forest rooted at that node and its neighbors. Finally, we showed that it is not necessary for the entire network topology to be known *a priori* in order to use our scheme, but that it is possible for the nodes to obtain the necessary information about the network by running the linear iteration with several different initial conditions.

In the next chapters, we will further extend these results to networks where communications between nodes are corrupted by noise (Chapter 3), where some nodes act in a malicious or faulty manner (Chapter 4), or where a set of nodes have to transmit streams of data through the network (Chapter 5).

CHAPTER 3

DISTRIBUTED CALCULATION OF LINEAR FUNCTIONS IN NOISY NETWORKS

3.1 Introduction

In the previous chapter, we showed that linear iterative strategies allow any node in connected time-invariant networks to calculate any arbitrary function of the initial node values in a finite number of time-steps (upper bounded by the size of the network). In this chapter, we extend these results on finite-time function calculation to the case where each node only obtains a noisy (or uncertain) measurement of its neighbors' values. While noisy transmissions between nodes in networks can often be handled by utilizing source or channel coding, the model that we consider here applies to situations where coding is not available, or where nodes directly sense the values of their neighbors, and their sensing or measurement capabilities are subject to noise [63]. Due to the noise model and techniques that we consider in this chapter, we will assume throughout that the field under consideration is the field of real numbers. Using only the first order statistics of the noise, we show that each node can obtain an unbiased estimate of any desired linear function of the initial values as a linear combination of the noisy values it receives from its neighbors during the linear iteration, along with its own values. Furthermore, this can be achieved after running the linear iteration for a finite number of steps with almost any choice of real-valued weight matrix. If the second order statistics of the noise are also known (perhaps only after running the linear iteration), we show how each node can refine its estimate of the linear function by choosing this linear combination of the noisy values in a way that minimizes the variance of the estimation error.

The special case of asymptotic consensus on a linear combination of the initial node values via noisy linear iterations has only recently started to receive attention (e.g., see [63, 58, 64, 65]). For example, it was shown in [64] that if the linear iteration is affected by additive noise at each time-step, the reliance on asymptotic convergence can cause the nodes to be driven arbitrarily far away from the desired consensus value. Other works have focused on addressing this issue by using predictive coding techniques [58], by using time-varying weight matrices [63], or by considering second-order recursions (where the value of each node at the next time-step depends not only on the values of the nodes at the current time-step, but on the values during the previous time-step as well) [65]; however, unlike the

method that we develop in this chapter, all of these previous works only allow each node to calculate an unbiased[1] estimate of the consensus value asymptotically (i.e., they do not obtain an unbiased estimate in a finite number of time-steps).

The rest of the chapter is organized as follows. In Section 3.2, we introduce the noise model and discuss why asymptotic consensus schemes perform poorly in noise. In Section 3.3, we show how linear iterative schemes can be used to allow each node to obtain a minimum-variance unbiased estimate of the function in a finite number of time-steps. We provide an example in Section 3.4, and we finish with our conclusions in Section 3.5.

3.2 The Noise Model

In the rest of the chapter, we will extend the results from Chapter 2 to the case where the network is operating in the presence of noise. We will first introduce the noise model, and then show that each node in the system can use the above techniques to obtain an unbiased estimate of any desired linear function[2] of the initial values in a finite number of time-steps.

Consider the noise-free linear iteration considered in Equation (2.2) of Chapter 2:

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] \\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}[k], \;\; 1 \leq i \leq N \;\; , \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{W}$ is an $N \times N$ weight matrix conforming to the network topology (i.e., w_{ij} is zero if $(x_j, x_i) \notin \mathcal{E}$). To simplify the development, we will assume (without loss of generality) in this chapter that the rows of $\mathbf{C}_i$ are ordered such that the first row of each $\mathbf{C}_i$ corresponds to node x_i's own value in the state vector $\mathbf{x}[k]$ (i.e., the i–th element of the first row of $\mathbf{C}_i$ is 1, and all other entries in that row are zero). Suppose that the values that each node x_i receives (or senses) from its neighbors are corrupted by noise (i.e., there is a noise component associated with each exchange of values between two neighboring nodes). Let $\mathbf{n}_i[k]$ denote the $\deg_i \times 1$ vector containing the noise that affects the values received by node x_i at time-step k. For each $1 \leq i \leq N$, let $\widehat{\mathbf{D}}_i$ denote the $(\deg_i + 1) \times (\deg_i)$ matrix given by $\widehat{\mathbf{D}}_i = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{\deg_i} \end{bmatrix}$; i.e., the first row of $\widehat{\mathbf{D}}_i$ has all entries equal to zero, and the remaining rows form the $\deg_i \times \deg_i$ identity matrix. The noisy values that node x_i receives at time-step k are then given by $\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k] + \widehat{\mathbf{D}}_i\mathbf{n}_i[k]$. Note that the reason for setting the top row of

[1] An estimate $\widehat{\Theta}$ of a parameter Θ is said to be *unbiased* if the expected value of $\widehat{\Theta}$ is equal to Θ [66].

[2] We will focus on linear functions in the rest of the chapter because, as we will see, unbiased estimates of such functions can be obtained as a linear combination of the values seen by each node over the linear iteration, thereby maintaining the implementation benefits of the linear iterative scheme. A special case of this result is that each node can also obtain an unbiased estimate of each initial node value via the linear iteration; if desired, it can potentially use these estimates to construct unbiased estimates of more complicated nonlinear functions as well. However, the exact form of such estimates will depend on the function to be calculated (and might require additional knowledge of the noise properties beyond simply the first order statistics) [66], whereas unbiased estimates of linear functions can be obtained simply by taking linear combinations of the unbiased estimates of each initial value.

$\widehat{\mathbf{D}}_i$ equal to zero is to model the fact that node x_i has noise-free access to its own value. If we define

$$\mathbf{n}[k] \equiv \begin{bmatrix} \mathbf{n}_1'[k] & \mathbf{n}_2'[k] & \cdots & \mathbf{n}_N'[k] \end{bmatrix}' \ ,$$
$$\mathbf{D}_i \equiv \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \widehat{\mathbf{D}}_i & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \ ,$$

where each $\mathbf{D}_i$ matrix has $\sum_{j=1}^{i-1} \deg_j$ columns of zeros, followed by the matrix $\widehat{\mathbf{D}}_i$, followed by $\sum_{j=i+1}^{N} \deg_j$ columns of zeros, the output seen by node x_i then becomes $\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k] + \mathbf{D}_i\mathbf{n}[k]$.

Now consider the update equation. Recall that each node uses the values that it receives from its neighbors to update its own value. In particular, node x_i multiplies the value that it receives from node x_j by the weight w_{ij}. Let $\bar{\mathbf{w}}_i$ denote the $1 \times \deg_i$ vector that contains the weights corresponding to the neighbors of node x_i. The update for node x_i is then given by

$$\begin{aligned} x_i[k+1] &= \begin{bmatrix} w_{ii} & \bar{\mathbf{w}}_i \end{bmatrix} \mathbf{y}_i[k] \\ &= \begin{bmatrix} w_{ii} & \bar{\mathbf{w}}_i \end{bmatrix} (\mathbf{C}_i\mathbf{x}[k] + \mathbf{D}_i\mathbf{n}[k]) \\ &= w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] + \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \bar{\mathbf{w}}_i & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \mathbf{n}[k] \ . \end{aligned}$$

Defining the matrix

$$\mathbf{B} \equiv \begin{bmatrix} \bar{\mathbf{w}}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{w}}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \bar{\mathbf{w}}_N \end{bmatrix} \ , \tag{3.1}$$

one obtains the noisy linear iteration model

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] + \mathbf{B}\mathbf{n}[k] \\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}[k] + \mathbf{D}_i\mathbf{n}[k], \ \ 1 \leq i \leq N \ . \end{aligned} \tag{3.2}$$

Note that the noise vector $\mathbf{n}[k]$ in this case has dimension $\left(\sum_{j=1}^{N} \deg_j\right) \times 1$ (since $\sum_{j=1}^{N} \deg_j$ is equal to the number of edges in the graph [28], and since each edge corresponds to a noisy transmission, one requires a noise term at each time-step for every edge in the graph).

Remark 3.1 *Note that the noise model in* (3.2) *can also handle the case where noise only affects the update equation for each node, but does not affect the exchange of values between nodes, simply by setting* $\mathbf{B}$ *to be the* $N \times N$ *identity matrix, and choosing each* $\mathbf{D}_i$ *to be the zero matrix. Note that in this case, the noise vector* $\mathbf{n}[k]$ *will only have* N *components.*

In [64], the authors considered the problem of asymptotic consensus in the presence of

update noise via a linear iteration of the form $\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] + \mathbf{n}[k]$, where $\mathbf{n}[k]$ is zero mean white noise with covariance matrix $E[\mathbf{n}[k]\mathbf{n}'[k]] = \mathbf{I}_N$, and $\mathbf{W}$ is a symmetric matrix providing asymptotic consensus to $\frac{1}{N}\mathbf{1}'\mathbf{x}[0]$. They showed that as $k \to \infty$, $E[x_i[k]] \to \frac{1}{N}\mathbf{1}'\mathbf{x}[0]$ for $1 \leq i \leq N$, but the variance of the node values $x_i[k]$ from the value $\frac{1}{N}\mathbf{1}'\mathbf{x}[0]$ increases without bound. This phenomenon is essentially due to the fact that any weight matrix that provides asymptotic consensus must necessarily have a (marginally stable) eigenvalue at 1 (from Theorem 1.1), and thus the components of the noise that excite this mode of the system will accumulate and cause the values of the nodes to evolve according to a random walk. However, the authors of [64] also examined the error between the node values and the average of the node values at *each* time-step. In other words, the authors examined the quantity $\mathbf{x}[k] - \frac{1}{N}\mathbf{1}\mathbf{1}'\mathbf{x}[k]$, and showed that the variance of this quantity remains bounded and reaches a steady state value as $k \to \infty$. Furthermore, they showed that the weight matrix that minimizes this variance can be obtained by solving a convex optimization problem. However, even though the variance of the distance between the node values remains bounded, the variance of each node value from the intended value $\frac{1}{N}\mathbf{1}'\mathbf{x}[0]$ can be arbitrarily large. As mentioned in Section 3.1, other works have focused on addressing this issue in various ways, but with the common element of obtaining convergence in an asymptotic number of time-steps.

In the next section, we provide a solution to the problem of finite-time unbiased function calculation by using the results on finite-time function calculation described in the previous chapter. In particular, we show that in strongly connected networks and with almost any choice of real-valued weight matrix $\mathbf{W}$ (subject to the constraint that $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i$), each node can obtain an unbiased estimate of any linear function of $\mathbf{x}[0]$ after running the linear iteration for a finite number of time-steps. Furthermore, if the second order statistics of the noise are known, we show how each node can minimize the variance of its estimate of the function (for a given choice of weight matrix $\mathbf{W}$).

Remark 3.2 *In this chapter, we will assume that the network topology is fixed, and that the corresponding weight matrix* $\mathbf{W}$ *is also fixed and chosen a priori. As we will see in the next section, given an appropriate weight matrix* $\mathbf{W}$*, each node can obtain an unbiased estimate of its desired linear function simply by knowing the first order statistics of the noise (and not necessarily the second order statistics). If the second order statistics are also known (perhaps only after running the linear iteration), we will show that each node can refine its estimate of its linear function by taking an appropriate linear combination of the values it sees over the course of the linear iteration. Note that this assumption of a fixed and known topology is also made in much of the existing literature on distributed consensus in the presence of noise (e.g., see [58, 64]). As described in Chapter 2.4, it is possible for the nodes to calculate their observability matrices (and the gains* Γ_i*) in a distributed manner when the network is noise-free, but the extension of such techniques to noisy networks is an*

open question and an avenue for future research. Along the same lines, one can investigate the extension of our techniques to handle noisy function calculation in time-varying graphs. As discussed in Chapter 1.3, one method to do this could be for nodes that become aware of changes in graph topology to inform the rest of the network, similar to what is proposed in [9]. Another option would be to treat dropped or added links as faults in the network, and utilize the techniques for fault-tolerant function calculation proposed in the next chapter.

3.3 Unbiased Minimum-Variance Estimation

We start by showing that each node can obtain an unbiased estimate of its desired linear function in a finite number of time-steps, and then describe how to minimize the variance of these estimates.

3.3.1 Unbiased Estimation

Consider the noisy system model given by (3.2). The output seen by node x_i over $L_i + 1$ time-steps is given by

$$\underbrace{\begin{bmatrix}\mathbf{y}_i[0]\\ \mathbf{y}_i[1]\\ \mathbf{y}_i[2]\\ \vdots\\ \mathbf{y}_i[L_i]\end{bmatrix}}_{\mathbf{y}_i[0:L_i]} = \underbrace{\begin{bmatrix}\mathbf{C}_i\\ \mathbf{C}_i\mathbf{W}\\ \mathbf{C}_i\mathbf{W}^2\\ \vdots\\ \mathbf{C}_i\mathbf{W}^{L_i}\end{bmatrix}}_{\mathcal{O}_{i,L_i}} \mathbf{x}[0] + \underbrace{\begin{bmatrix}\mathbf{D}_i & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{C}_i\mathbf{B} & \mathbf{D}_i & \cdots & \mathbf{0}\\ \mathbf{C}_i\mathbf{W}\mathbf{B} & \mathbf{C}_i\mathbf{B} & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{C}_i\mathbf{W}^{L_i-1}\mathbf{B} & \mathbf{C}_i\mathbf{W}^{L_i-2}\mathbf{B} & \cdots & \mathbf{D}_i\end{bmatrix}}_{\mathcal{M}_{i,L_i}} \underbrace{\begin{bmatrix}\mathbf{n}[0]\\ \mathbf{n}[1]\\ \vdots\\ \mathbf{n}[L_i]\end{bmatrix}}_{\mathbf{n}[0:L_i]}. \tag{3.3}$$

We will assume here that the noise is zero mean (i.e., $E[\mathbf{n}[k]] = 0$ for all k); this assumption can be easily relaxed, but we adopt it for simplicity. Suppose each node x_i wants to calculate an unbiased estimate of the function $\mathbf{c}_i'\mathbf{x}[0]$, for some vector $\mathbf{c}_i'$. We will find a gain Γ_i and the smallest integer L_i for each node x_i so that the quantity $\Gamma_i\mathbf{y}_i[0:L_i]$ is an unbiased estimate of $\mathbf{c}_i'\mathbf{x}[0]$. To this end, we use (3.3) to examine the estimation error

$$\begin{aligned}\epsilon_i &\equiv \Gamma_i\mathbf{y}_i[0:L_i] - \mathbf{c}_i'\mathbf{x}[0]\\ &= \left(\Gamma_i\mathcal{O}_{i,L_i} - \mathbf{c}_i'\right)\mathbf{x}[0] + \Gamma_i\mathcal{M}_{i,L_i}\mathbf{n}[0:L_i]\ .\end{aligned} \tag{3.4}$$

The estimate $\Gamma_i\mathbf{y}_i[0:L_i]$ will be unbiased (i.e., $E[\epsilon_i] = 0$) for any $\mathbf{x}[0]$ if and only if matrix Γ_i satisfies

$$\Gamma_i\mathcal{O}_{i,L_i} = \mathbf{c}_i'\ . \tag{3.5}$$

In other words, the vector $\mathbf{c}_i'$ must be in the row-space of the matrix $\mathcal{O}_{i,L_i}$. Following the notation in Theorem 2.1, let $\mathcal{S}_i$ denote the set of all nodes that have a path to node x_i in the network. As long as all nonzero entries of $\mathbf{c}_i'$ are in columns corresponding to nodes

in $\mathcal{S}_i$, Theorem 2.1 indicates that for almost any[3] real-valued choice of weight matrix $\mathbf{W}$, $\mathbf{c}_i'$ will be in the row space of $\mathcal{O}_{i,L_i}$, for some $0 \leq L_i < D_i \leq |\mathcal{S}_i| - \deg_i$ (where D_i is the size of the largest tree in an optimal spanning forest of the network). Assuming that $\mathbf{W}$ is chosen appropriately, one can find the smallest L_i for which the vector $\mathbf{c}_i'$ is in the row-space of the matrix $\mathcal{O}_{i,L_i}$, and this will also be the smallest number of time-steps required for unbiased estimation by node x_i (for that choice of $\mathbf{W}$). The above discussion, along with Theorem 2.1, immediately leads to the following theorem.

Theorem 3.1 *Let $\mathcal{G}$ denote the graph of the network. Define the set*

$$\mathcal{S}_i = \{x_j | \text{ There exists a path from } x_j \text{ to } x_i \text{ in } \mathcal{G}\} \cup \{x_i\} \ .$$

Consider a subgraph $\mathcal{H}$ of $\mathcal{G}$ that is a $\mathcal{S}_i$-spanning forest rooted at $\{x_i\} \cup \mathcal{N}_i$, with the property that the size of the largest tree in $\mathcal{H}$ is minimal over all possible $\mathcal{S}_i$-spanning forests rooted at $\{x_i\} \cup \mathcal{N}_i$. Let D_i denote the size of the largest tree of $\mathcal{H}$. Then, for almost any real-valued choice of weight matrix $\mathbf{W}$, node x_i can obtain an unbiased estimate of any linear function of the values $\{x_j[0] \mid x_j \in \mathcal{S}_i\}$ after running the linear iteration (3.2) for $L_i + 1$ time-steps, for some $0 \leq L_i < D_i$.

Remark 3.3 *Note that since we are only interested in calculating a linear function of the initial values and not necessarily the initial values themselves, it may be the case that the observability matrix for node x_i contains the vector $\mathbf{c}'$ before it becomes full column rank, in which case node x_i would be able to calculate $\mathbf{c}'\mathbf{x}[0]$ in less than D_i time-steps.*

Note that for a given $\mathbf{W}$, there may be multiple choices of Γ_i satisfying (3.5). This leads us to ask the question: If the second order statistics of the noise are also known (perhaps *a posteriori*), can one obtain a better estimate of the linear function by choosing the gain Γ_i appropriately? We will address this question in the following section.

3.3.2 Minimizing the Variance of the Estimate

In order to minimize the mean square error of each node's estimate of its linear function, suppose that the covariance of the noise is known (or obtained over the course of the linear iteration), and given by $E[\mathbf{n}[k]\mathbf{n}'[j]] = \mathbf{Q}_{kj}$, for some positive semi-definite matrix $\mathbf{Q}_{kj}$. Note that we are not assuming any constraints on the second order statistics (e.g., the noise does not have to be stationary, and can be colored). Examining the estimation error given by (3.4), we note that after satisfying the unbiased condition (3.5), the expression for the

[3] In this context, the phrase "almost any" indicates that the set of weights for which the property does not hold has Lebesgue measure zero. In other words, if the weights are chosen independently from some uniform distribution on the field of real numbers, the property will hold with probability 1.

variance of the error is given by

$$\begin{aligned}\sigma_i &\equiv E\left[\epsilon_i \epsilon_i'\right] \\ &= \Gamma_i \mathcal{M}_{i,L_i} E\left[\mathbf{n}[0:L_i]\mathbf{n}[0:L_i]'\right] \mathcal{M}_{i,L_i}' \Gamma_i' \\ &= \Gamma_i \mathcal{M}_{i,L_i} \underbrace{\begin{bmatrix} \mathbf{Q}_{00} & \mathbf{Q}_{01} & \cdots & \mathbf{Q}_{0L_i} \\ \mathbf{Q}_{10} & \mathbf{Q}_{11} & \cdots & \mathbf{Q}_{1L_i} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{L_i 0} & \mathbf{Q}_{L_i 1} & \cdots & \mathbf{Q}_{L_i L_i} \end{bmatrix}}_{\Pi_{L_i}} \mathcal{M}_{i,L_i}' \Gamma_i' \ . \end{aligned} \tag{3.6}$$

Our objective in this section will be to find the matrix Γ_i that minimizes the error variance in (3.6) (for a given weight matrix $\mathbf{W}$ and delay L_i), while maintaining unbiased estimation. To achieve this, we will parameterize the gain Γ_i, and use the remaining freedom after satisfying (3.5) to minimize the expression in (3.6).

Suppose L_i is chosen as the smallest integer for which (3.5) has a solution (this will be the smallest delay required for unbiased estimation by node x_i with the given weight matrix $\mathbf{W}$). Let the singular value decomposition of the matrix $\mathcal{O}_{i,L_i}$ be given by $\mathcal{O}_{i,L_i} = \mathbf{U}_i \left[\begin{smallmatrix} \Lambda_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] \mathbf{V}_i'$, where $\mathbf{U}_i$ and $\mathbf{V}_i$ are unitary matrices, and Λ_i is a diagonal matrix with positive entries. Furthermore, $\text{rank}(\Lambda_i) = \text{rank}(\mathcal{O}_{i,L_i})$ [27]. Substituting this into (3.5), we get

$$\Gamma_i \mathbf{U}_i \begin{bmatrix} \Lambda_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{c}_i' \mathbf{V}_i \ . \tag{3.7}$$

Clearly, since $\mathbf{c}_i'$ is in the row-space of $\mathcal{O}_{i,L_i}$, it must be the case that

$$\mathbf{c}_i' \mathbf{V}_i = \begin{bmatrix} \mathbf{a}_i' & \mathbf{0} \end{bmatrix} \tag{3.8}$$

for some vector $\mathbf{a}_i'$ with $\text{rank}(\mathcal{O}_{i,L_i})$ entries. Define the matrix

$$\widehat{\Gamma}_i \equiv \Gamma_i \mathbf{U}_i \ , \tag{3.9}$$

and partition it as $\widehat{\Gamma}_i = \begin{bmatrix} \widehat{\Gamma}_{i1} & \widehat{\Gamma}_{i2} \end{bmatrix}$, where $\widehat{\Gamma}_{i1}$ has $\text{rank}(\mathcal{O}_{i,L_i})$ columns. Equation (3.7) then becomes $\begin{bmatrix} \widehat{\Gamma}_{i1} & \widehat{\Gamma}_{i2} \end{bmatrix} \left[\begin{smallmatrix} \Lambda_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] = \begin{bmatrix} \mathbf{a}_i' & \mathbf{0} \end{bmatrix}$. From this equation, it is apparent that

$$\widehat{\Gamma}_{i1} = \mathbf{a}_i' \Lambda_i^{-1} \ , \tag{3.10}$$

and $\widehat{\Gamma}_{i2}$ is completely unconstrained. In other words, $\widehat{\Gamma}_{i2}$ represents the freedom in the gain Γ_i after satisfying the unbiased constraint given by (3.5).

To minimize the variance of the estimation error, we substitute the above parameteri-

zation of the gain Γ_i into (3.6) to obtain

$$\sigma_i = \begin{bmatrix} \widehat{\Gamma}_{i1} & \widehat{\Gamma}_{i2} \end{bmatrix} \mathbf{U}_i' \mathcal{M}_{i,L_i} \Pi_{L_i} \mathcal{M}_{i,L_i}' \mathbf{U}_i \begin{bmatrix} \widehat{\Gamma}_{i1} & \widehat{\Gamma}_{i2} \end{bmatrix}' \ .$$

Define

$$\begin{bmatrix} \mathbf{\Phi}_i \\ \mathbf{\Psi}_i \end{bmatrix} \equiv \mathbf{U}_i' \mathcal{M}_{i,L_i} \ , \tag{3.11}$$

where $\mathbf{\Phi}_i$ has $\text{rank}(\mathcal{O}_{i,L_i})$ rows. Using (3.10), the variance of the error becomes

$$\begin{aligned} \sigma_i &= \left(\mathbf{a}_i' \Lambda_i^{-1} \mathbf{\Phi}_i + \widehat{\Gamma}_{i2} \mathbf{\Psi}_i \right) \Pi_{L_i} \left(\mathbf{a}_i' \Lambda_i^{-1} \mathbf{\Phi}_i + \widehat{\Gamma}_{i2} \mathbf{\Psi}_i \right)' \\ &= \mathbf{a}_i' \Lambda_i^{-1} \mathbf{\Phi}_i \Pi_{L_i} \mathbf{\Phi}_i' \Lambda_i^{-1} \mathbf{a}_i + \mathbf{a}_i' \Lambda_i^{-1} \mathbf{\Phi}_i \Pi_{L_i} \mathbf{\Psi}_i' \widehat{\Gamma}_{i2}' \\ &\quad + \widehat{\Gamma}_{i2} \mathbf{\Psi}_i \Pi_{L_i} \mathbf{\Phi}_i' \Lambda_i^{-1} \mathbf{a}_i + \widehat{\Gamma}_{i2} \mathbf{\Psi}_i \Pi_{L_i} \mathbf{\Psi}_i' \widehat{\Gamma}_{i2}' \ . \end{aligned}$$

To minimize the above expression, we take the gradient with respect to $\widehat{\Gamma}_{i2}$ and set it equal to zero, which produces

$$\widehat{\Gamma}_{i2} = -\mathbf{a}_i' \Lambda_i^{-1} \mathbf{\Phi}_i \Pi_{L_i} \mathbf{\Psi}_i' \left(\mathbf{\Psi}_i \Pi_{L_i} \mathbf{\Psi}_i' \right)^\dagger \ ,$$

where the notation $(\cdot)^\dagger$ indicates the pseudo-inverse of a matrix [67]. From (3.9) and (3.10), we now obtain the optimal gain for node x_i as

$$\Gamma_i = \mathbf{a}_i' \Lambda_i^{-1} \begin{bmatrix} \mathbf{I}_{\text{rank}(\mathcal{O}_{i,L_i})} & -\mathbf{\Phi}_i \Pi_{L_i} \mathbf{\Psi}_i' \left(\mathbf{\Psi}_i \Pi_{L_i} \mathbf{\Psi}_i' \right)^\dagger \end{bmatrix} \mathbf{U}_i' \ . \tag{3.12}$$

The variance of the optimal estimate can now be obtained from (3.6).

Remark 3.4 *In the above derivation, we took L_i to be the smallest delay for which unbiased estimation is possible by node x_i. If one increases L_i past this minimum value, node x_i can potentially reduce the variance of its estimate (since it is obtaining more information about the initial state). The variance of the estimate will be a nonincreasing function of the delay L_i, and thus the tradeoff between delay and variance can be taken as a design parameter for a given graph. The gain Γ_i and the variance σ_i for any value of L_i (above the minimum required for unbiased estimation) can be obtained by following the above procedure. A quantitative characterization of the relationship between delay and variance will be the subject of future research.*

Remark 3.5 *It may be the case that some nodes can obtain an unbiased estimate of their desired functions faster than others (i.e., there may exist nodes x_i and x_j such that $L_i < L_j$). In such cases, one can have all nodes run the linear iteration for $\max_{1 \leq j \leq N} L_j + 1$ time-steps, so that every node receives enough information to obtain an unbiased estimate. Each node x_i can either calculate the function $\mathbf{c}_i' \mathbf{x}[0]$ after the first $L_i + 1$ time-steps of the linear*

iteration (i.e., with minimum delay), or it can use the outputs over all $\max_{1\leq j\leq N} L_j + 1$ *time-steps (which could reduce the variance of its estimate).*

3.4 Example

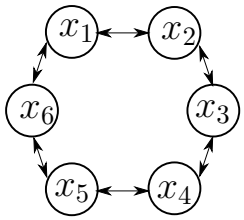

Figure 3.1: Ring with 6 nodes. Each transmission is corrupted by zero-mean white noise with unit variance.

Consider the ring network with $N = 6$ nodes in Figure 3.1. The transmissions between nodes are assumed to be corrupted by zero-mean white noise with unit variance. The objective in this system is for each node to calculate an unbiased estimate of the average of the initial values (i.e., each node must calculate an unbiased estimate of $\frac{1}{6}\mathbf{1}'\mathbf{x}[0]$). To accomplish this, recall from Section 2.3.1 that the observability matrix for every node will have rank N after $\lfloor \frac{N}{2} \rfloor = 3$ time-steps as long as all weights are chosen to be nonzero entries of any field. In this case, we operate with the field of real numbers and choose all of the weights to be "1", which produces the weight matrix

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{3.13}$$

One can verify that the above weight matrix allows every node to calculate the average. For example, node x_1 in Figure 3.1 receives values from nodes x_2 and x_6, and has access to its own value, which means that

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The observability matrix for node x_1 is given by

$$\mathcal{O}_{1,2} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_1\mathbf{W} \\ \mathbf{C}_1\mathbf{W}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 3 & 2 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 2 & 3 \end{bmatrix},$$

which has rank 6. Since the row-space of this matrix contains the vector $\mathbf{c}' = \frac{1}{6}\mathbf{1}'$, node x_1 can calculate an unbiased estimate of the average after three time-steps (i.e., after it sees the outputs $\mathbf{y}_1[0]$, $\mathbf{y}_1[1]$ and $\mathbf{y}_1[2]$). The same analysis holds for all nodes in the network.

Our task is to choose the gain Γ_i for each node x_i satisfying the unbiased condition (3.5), while minimizing the variance of the estimation error. Recall that the transmissions between nodes are assumed to be corrupted by zero-mean white noise with unit variance. To determine the optimal gain matrix Γ_i for each node x_i, we have to first construct the noisy system model in (3.2). For example, consider node x_1 in Figure 3.1. Since the values received by node x_1 from its neighbors are corrupted by noise, the output seen by node x_1 at time-step k is given by

$$\mathbf{y}_1[k] = \mathbf{C}_1\mathbf{x}[k] + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\widehat{\mathbf{D}}_1} \mathbf{n}_1[k] \ ,$$

where $\mathbf{n}_1[k]$ contains the additive noise on the links from node x_2 and node x_6 to node x_1 at time-step k. The outputs seen by all the other nodes can be obtained in a similar manner. We can group the noise vectors seen by each node into the single noise vector

$$\mathbf{n}[k] = \begin{bmatrix} \mathbf{n}_1'[k] & \mathbf{n}_2'[k] & \mathbf{n}_3'[k] & \mathbf{n}_4'[k] & \mathbf{n}_5'[k] & \mathbf{n}_6'[k] \end{bmatrix}' \ ,$$

which has 12 entries, since there are six bidirectional links in the graph. Note that $\mathbf{n}[k]$ is white noise with $E[\mathbf{n}[k]] = 0$ and $E[\mathbf{n}[k]\mathbf{n}'[k]] = \mathbf{I}_{12}$ (by assumption in this example). With this notation, the output for node x_1 is given by

$$\mathbf{y}_1[k] = \mathbf{C}_1\mathbf{x}[k] + \underbrace{\begin{bmatrix} \widehat{\mathbf{D}}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{D}_1} \mathbf{n}[k] \ .$$

Since each node multiplies the values that it receives from its neighbors by "1" when it updates its value, the state update equation is given by the first equation in (3.2), where the matrix $\mathbf{B}$ is obtained from (3.1) as

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

We now have to find the gain Γ_1 for node x_1 satisfying (3.5), while minimizing the error variance in (3.6). To do this, we first find the singular value decomposition of $\mathcal{O}_{1,2}$ as $\mathcal{O}_{1,2} = \mathbf{U}_1 \begin{bmatrix} \Lambda_1 \\ \mathbf{0} \end{bmatrix} \mathbf{V}_1'$, where $\Lambda_1 = \text{diag}(7.403, 3.244, 1.414, 0.945, 0.689, 0.554)$. We omit the values of $\mathbf{U}_1$ and $\mathbf{V}_1$ in the interest of space. From the above decomposition, we obtain the vector $\mathbf{a}_1'$ in (3.8) as

$$\mathbf{a}_1' = \frac{1}{6}\mathbf{1}'\mathbf{V}_1 = \begin{bmatrix} -0.371 & 0 & -0.087 & -0.146 & 0 & 0.0123 \end{bmatrix}.$$

We also obtain the matrices $\mathbf{\Phi}_i$ and $\mathbf{\Psi}_i$ from (3.11) (again, these values are omitted in the interest of space). Substituting these values into the expression for the optimal gain in (3.12), with $\Pi_2 \equiv E\left[\mathbf{n}[0:2]\mathbf{n}'[0:2]\right] = \mathbf{I}_{36}$ (since the noise is white with unit variance), we obtain $\Gamma_1 = \frac{1}{60}\begin{bmatrix} -2 & -2 & -2 & -6 & 2 & 2 & -2 & 5 & 5 \end{bmatrix}$. The mean-square error for the estimate of the average obtained by node x_1 is calculated by substituting the above gain into (3.6), and is found to be $\sigma_1 = 0.0972$. The above procedure can be repeated to obtain the optimal gains for all nodes, and in this example the minimum mean-square error for all nodes (with the weight matrix (3.13)) is the same (i.e., $\sigma_i = 0.0972$ for all i).

Once the optimal gains are calculated and provided to each node, suppose that the initial values of the nodes are given by

$$\mathbf{x}[0] = \begin{bmatrix} 0.8372 & -5.3279 & 3.6267 & 4.4384 & -2.7324 & 8.9546 \end{bmatrix}',$$

which has a mean of 1.6328. The nodes run the linear iteration given by (3.2) for three time-steps with $\mathbf{x}[0]$ as given above, and driven by zero-mean white noise with unit variance. An example of the outputs seen by node x_1 during the three time-steps with a particular

sampling of noise vectors $\mathbf{n}[0], \mathbf{n}[1]$ and $\mathbf{n}[2]$ is given by

$$\begin{aligned}
\mathbf{y}_1[0] &= \begin{bmatrix} 0.8372 & -5.7605 & 7.2890 \end{bmatrix}', \\
\mathbf{y}_1[1] &= \begin{bmatrix} 2.3658 & -1.0393 & 9.7817 \end{bmatrix}', \\
\mathbf{y}_1[2] &= \begin{bmatrix} 11.1081 & 3.9822 & 24.3226 \end{bmatrix}' .
\end{aligned}$$

Node x_1 then obtains a minimum-variance unbiased estimate of the average as

$$\Gamma_1 \begin{bmatrix} \mathbf{y}_1[0]' & \mathbf{y}_1[1]' & \mathbf{y}_1[2]' \end{bmatrix}' = 1.9644.$$

The other nodes follow the same procedure, and the values calculated by nodes x_2, x_3, x_4, x_5 and x_6 (in this sample run) are 1.8009, 1.4784, 1.78888, 1.8473 and 1.8395, respectively.

To verify empirically that the variance of the estimation error at each node x_i is indeed $\sigma_i = 0.0972$, we perform 1000 runs of the linear iteration; for each run, the initial value of each node is chosen as an independent random variable uniformly distributed in the interval $[-5, 5]$, and the linear iteration is performed for 3 time-steps while being driven by white noise with unit variance. Let $\mathbf{x}^j[0]$ denote the vector of initial values during the j–th run, and let $\mathbf{y}_i^j[k]$ denote the values seen by node x_i during the k–th time-step of the j–th run. At the end of the j–th run, each node x_i calculates an estimate of the average of the initial conditions as $\Gamma_i \begin{bmatrix} \mathbf{y}_i^j[0]' & \mathbf{y}_i^j[1]' & \mathbf{y}_i^j[2]' \end{bmatrix}'$, where Γ_i is the optimal gain matrix calculated above for each node x_i. The estimation error during the j–th run for the i–th node is given by $\epsilon_i^j \equiv \Gamma_i \begin{bmatrix} \mathbf{y}_i^j[0]' & \mathbf{y}_i^j[1]' & \mathbf{y}_i^j[2]' \end{bmatrix}' - \frac{1}{6}\mathbf{1}'\mathbf{x}^j[0]$. After all 1000 runs are completed, we calculate an empirical mean square error for each node x_i as $\widehat{\sigma}_i = \frac{1}{1000}\sum_{j=1}^{1000}\left(\epsilon_i^j\right)^2$, and these mean square errors are found to be

$$\begin{bmatrix} \widehat{\sigma}_1 & \widehat{\sigma}_2 & \widehat{\sigma}_3 & \widehat{\sigma}_4 & \widehat{\sigma}_5 & \widehat{\sigma}_6 \end{bmatrix} = \begin{bmatrix} 0.0965 & 0.0907 & 0.0998 & 0.0901 & 0.1020 & 0.0986 \end{bmatrix}.$$

As expected, the empirical mean square errors are found to be close to the theoretical value of 0.0972, which was calculated above via equation (3.6).

Remark 3.6 *It is worth considering what happens to the variance of the estimation error if the nodes perform the linear iteration for more time-steps than necessary. As noted in Remark 3.4, the variance will be a nonincreasing function of the delay L_i. In the above example, we noted that with the weight matrix $\mathbf{W}$ given in* (3.13)*, the nodes had to run the linear iteration for at least $L_i + 1 = 3$ time-steps in order to obtain unbiased estimation, and the corresponding minimum mean square error was $\sigma_i = 0.0972$ for every i. Now suppose that we allow the nodes to run the linear iteration for* 4 *time-steps instead of* 3 *(i.e., we take $L_i = 3$ for each node x_i). Repeating the procedure in Section 3.3.2, we find that the minimum mean square error drops to $\sigma_i = 0.0478$ for each node x_i. In other words,*

allowing the nodes to run the linear iteration for one more time-step provides them with enough information to reduce the variance of their estimation error from 0.0972 *to* 0.0478*. By repeating this analysis for values of* L_i *between* 2 *and* 10*, we obtain the graph shown in Figure 3.2. This graph shows that, for the weight matrix* $\mathbf{W}$ *in* (3.13)*, the minimum mean square error asymptotically approaches a value of* 0.0415 *as the delay* L_i *increases (and in fact, after* $L_i = 5$*, there is very little reduction in the variance of the error). A quantitative relationship between the delay* L_i *and the variance* σ_i *will be left to future research.*

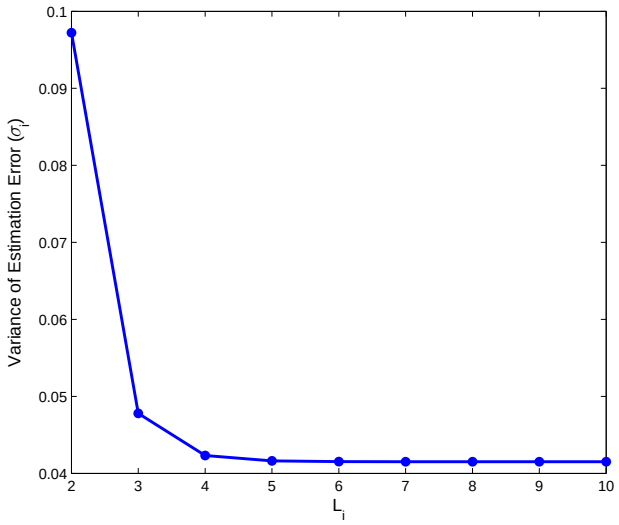

Figure 3.2: Variance of estimation error (σ_i) versus the delay L_i. Note that the delay L_i is taken to be the same for every node x_i, and this causes the variance of the estimation error for each node x_i to also be the same (for this example).

3.5 Summary

We studied the problem of distributively calculating linear functions in networks operating in the presence of noise. In particular, we analyzed a linear iterative strategy where each node updates its value as a linear combination of its own value and the noisy transmissions of its neighbors. In the study of traditional linear iterative schemes that rely on asymptotic convergence, it has been shown that the presence of noise can drive the node values arbitrarily far away from the desired function. To solve this problem, we utilized our framework on finite-time function calculation via linear iterations. Specifically, for a given set of update weights, we showed that it is possible to calculate a set of gains for each node that allows it to obtain a minimum-variance unbiased estimate of the function after running the linear iteration for a finite number of time-steps with almost any choice of real-valued weight matrix.

CHAPTER 4

DISTRIBUTED FUNCTION CALCULATION IN THE PRESENCE OF MALICIOUS AGENTS

4.1 Introduction

In Chapter 2, we showed that in connected networks with time-invariant topologies, the nodes can calculate any desired function of the initial values with high probability after running a linear iteration with random weights from a field of sufficiently large size. Furthermore, we showed that only a finite number of time-steps (upper bounded by the size of the network) are required in order to achieve this. In this chapter, we extend those results to address the problem of function calculation in the presence of faulty or malicious nodes. Specifically, we allow for the possibility that some nodes in the network update their values at each time-step in an arbitrary manner, instead of following the predefined strategy of using a weighted linear combination of their neighbors' (and own) values. Such arbitrary updates can occur, for example, if some nodes in the network are compromised by a malicious attacker whose objective is to disrupt the operation of the network [11], or they might be the result of hardware malfunctions which cause the nodes to incorrectly calculate their update value [68] (note that the former case is more challenging to handle than the latter case). Due to the increasingly prevalent use of sensor networks and multi-agent systems in life- and mission-critical applications [1, 69, 70], it is imperative to analyze any proposed network protocols to determine their limitations with regard to nodes that behave in such erroneous or unexpected ways. The robustness of various existing information dissemination schemes to this type of faulty behavior has been investigated in the literature (e.g., see [10]); however, a similar analysis of the susceptibility of *linear iterative* strategies to malicious or incorrect behavior by a subset of nodes has received scant attention, and this is the contribution of this chapter. Specifically, we ask (and answer) the following questions. With the only constraint being that malicious nodes cannot send different values to different neighbors (such as in a wireless network), how many nodes have to be malicious in order to disrupt the linear iterative strategy? How should these malicious nodes be chosen in the network, and how should they update their values at each time-step in order to prevent other nodes from calculating certain functions? Is it possible to defend linear iterative strategies against such malicious behavior? If so, what is an appropriate decoding strategy for the nodes to follow?

We answer the above questions by showing that the graph connectivity is the determining factor for the ability of linear iterative strategies to tolerate malicious (or faulty) agents. Specifically, if a given node x_i has $2f$ or fewer vertex-disjoint paths from some other node x_j in the network, then we show that it is possible for only f nodes to maliciously update their values in a way such that x_i cannot obtain sufficient information to calculate any function that involves node x_j's value, regardless of the number of time-steps for which the linear iteration is run. Our analysis is constructive, in that it provides a specific choice of malicious nodes, along with a strategy for the malicious nodes to follow; as we will show, the linear iterative protocol will require the malicious nodes to update their values in certain specific and clever ways in order to disrupt the network. This is in contrast to other information distribution strategies (such as flooding) where the malicious nodes have a clear-cut and simple method to prevent certain nodes from calculating functions or from obtaining certain information about the values of other nodes [10, 11, 71].

Conversely, we also show that if a given node has at least $2f+1$ node-disjoint paths from any other node in the network, it can work around up to f misbehaving nodes to correctly calculate any arbitrary function of the initial node values. Furthermore, we show that this can be achieved after running the linear iteration with almost any set of real-valued weights for a finite number of time-steps (upper bounded by the number of nodes in the network). To derive our results, we build upon the function calculation protocol from Chapter 2, and exploit concepts from classical linear system theory (such as strong observability and invariant zeros of linear systems) to overcome malicious or faulty nodes. Once again, our analysis provides a specific decoding strategy for the nodes to follow in order to overcome malicious behavior. Together, these results serve to narrow the gap between linear iterative schemes and existing fault-tolerant consensus protocols in the literature (such as those described in [11]), and show (perhaps surprisingly) that simple linear-iterative schemes are as powerful as any other protocol in terms of the number of malicious nodes they can handle under the wireless broadcast model (i.e., they impose the same constraints on the underlying network topology).

Remark 4.1 *The concept of strong observability of linear systems (see Section 1.5) will play a key role in our derivations. As mentioned in Section 1.5, Theorem 1.3 provides a convenient test for strong observability in terms of the invariant zeros of linear systems over the field of complex numbers, but this test does not transfer to finite fields (as shown by the counterexample in that chapter). While we were able to derive direct proofs of the properties of structural observability and structural invertibility over finite fields in the Appendix, we do not currently have a similar proof for structural strong observability. For this reason, our analysis in this chapter will focus on linear iterative strategies with real-valued transmissions and updates (over the field of complex numbers), and this will allow us to leverage Theorem 1.3 to derive strategies that are robust to malicious nodes. An extension*

of these results (with an accompanying proof of structural strong observability over finite fields) remains an avenue for future research.

The rest of the chapter is organized as follows. In Section 4.2, we briefly describe existing work that deals with the robustness of information dissemination strategies (including linear iterative strategies) to faulty behavior by certain nodes. In Section 4.3, we formulate the fault model and introduce the main results of the chapter. In Section 4.4, we show how to choose a set of malicious nodes and provide a strategy for them to follow in order to prevent other nodes in the network from calculating their desired functions. In Section 4.5, we show how to defend the linear iterative strategy against a fixed number of malicious nodes (provided that the connectivity of the network satisfies certain conditions). We conclude the chapter in Section 4.6.

4.2 Previous Results on Fault-Tolerant Function Calculation

The problem of transmitting information over networks (and specifically, reaching consensus) in the presence of faulty or malicious nodes has been studied thoroughly over the past several decades (e.g., see [10, 11] and the references therein). It is known in the literature that if there are only $2f$ independent paths between two nodes in a network, then f malicious nodes (with the capability to act in an arbitrary and coordinated manner) can prevent at least one of the nodes from receiving any information about the other (*regardless* of the protocol). The intuition behind this requirement is simple: If there are only $2f$ independent paths between the two nodes, then a set of malicious nodes on f of the paths could collude to pretend that the transmitted value was something other than the true value, and the receiving node would not know whether to believe the f malicious nodes, or the f nodes transmitting the true value on the other f independent paths [10, 11, 71].

The computer science community in particular has extensively investigated scenarios where malicious nodes are allowed to behave in a completely arbitrary manner, and have the ability to send different values to different neighbors; such behavior is referred to as *Byzantine*. In such cases, there are various algorithms to reach consensus in the presence of f Byzantine nodes provided that (i) the number of nodes in the network is at least $3f + 1$ and (2) the connectivity of the network is at least $2f + 1$ [11, 72, 73]. The requirement that fewer than one-third of the nodes be Byzantine is due to the fact that Byzantine nodes are allowed to send different (and conflicting) messages to their neighbors: thus, in order for the non-Byzantine nodes to have a consistent representation of the values transmitted in the network, each of them needs to be reassured by a set of at least $f + 1$ other nodes about the values that were transmitted by the Byzantine nodes. The requirement that the connectivity of the network be at least $2f + 1$ is in order to allow nodes to securely

communicate their values to each other. Specifically, if node x_i wants to send a message to node x_j, it can simply send this message along $2f+1$ different paths, and x_j would take the value that appears on at least $f+1$ of the paths to be the true value. Note that this condition is not sufficient to reach consensus on its own when Byzantine nodes are allowed to send differing values to different nodes, since a non-Byzantine node still would not be able to tell what information was communicated to the other nodes in the network. Under both of these conditions, one method to solve the problem is to use a *flooding* protocol, where every node in the network simply forwards any value that it receives from a neighbor to all of its other neighbors, annotated with the path that the message followed through the network. With such a protocol, all of the non-Byzantine nodes in the network can parse the messages that they receive in order to come to an agreement about the values that were transmitted [11]. Along the same lines, researchers have also developed algorithms to reliably broadcast values in radio networks (where each transmission by a node is heard by all of its neighbors) that contain malicious nodes. For example, the authors of [31, 32] characterized bounds on the maximum number of nodes that can be malicious in the neighborhood of any given node (in grid networks) before reliable broadcast becomes impossible, and provided protocols to achieve these bounds.

However, as mentioned in the previous section, the vulnerability of linear iterative function calculation schemes to malicious behavior has not received much attention in the literature. For example, what happens if some nodes update their values to be something other than the predefined weighted combination of their neighbors' and own values? In [24], it was shown that if a node (called a *leader*) in the network does not update its value at each time-step (i.e., it maintains a constant value), then a particular class of linear iterative strategies (where each node updates its value to be a convex combination of its neighborhood values) will cause all other nodes to asymptotically converge to the value of the leader (i.e., they behave as *followers*). While this may be acceptable behavior when the network has a legitimate leader, it also seems to indicate that a simple asymptotic consensus scheme can be easily disrupted by just a single malicious node. A similar analysis was done in [74], where it was argued that since the asymptotic consensus scheme can be disrupted by a single node that maintains a constant value, it can also be disrupted by a single node that updates its values arbitrarily (since maintaining a constant value is a special case of arbitrary updates). Both of these works only considered a straightforward application of the linear iteration for asymptotic consensus (whereby each node simply updates its value as a weighted linear combination of its own current value and those of its neighbors, *ad infinitum*), without making more explicit use of the information available to the nodes via the iteration.

The problem was revisited by Pasqualetti et al. [75] for the specific case where all nodes have to reach agreement (not necessarily on any specific function of the initial values), and where the malicious nodes do not behave in a completely arbitrary manner (specifically,

the additive errors introduced by the malicious nodes in that setting are not allowed to asymptotically decay faster than a certain rate). Under these special conditions, Pasqualetti et al. showed that if every node in the system uses the information it receives from the linear iteration to try and estimate the malicious updates injected into the network (via an *unknown input observer*), then a graph connectivity of $2f$ is sufficient to detect and isolate up to $2f-1$ malicious nodes, and have the other nodes reach agreement. In other words, Pasqualetti et al. showed that at least $2f$ nodes would have to be malicious (if they acted in a constrained manner) in a graph with connectivity $2f$ in order to prevent the other nodes from reaching agreement. This result shows that linear iterative strategies actually have some degree of robustness against malicious nodes if one uses the information available to each node appropriately. It is also worth noting that although this result seems to contradict the intuition that only f malicious nodes should be required to disrupt a network with connectivity $2f$, this discrepancy arises due to the fact that the malicious nodes are assumed to act in a *restricted* manner in this analysis.

Based on the above discussion, the fundamental question that we examine in this chapter is the following: How should f malicious nodes behave in a network of $2f$ connectivity in order to disrupt the linear iterative strategy? Note that this question is easy to answer for certain other protocols. For example, suppose that the f malicious nodes are chosen to be one-half of a vertex cut of size $2f$ between nodes x_j and x_i. Then in the flooding protocol described above, the malicious nodes can prevent node x_i from discovering node x_j's true value by simply changing x_j's value in any message that they pass through to node x_i (this is easy to achieve, since the flooding protocol simply requires each node to forward any message that it receives). Node x_i would then receive f messages indicating that node x_j has a certain value, and f messages indicating that node x_j has a different value, and would therefore not know which set to believe. However, this strategy will not work when the nodes in the network follow a linear iterative protocol because in this case, the messages that the malicious nodes receive and transmit are not simply the explicit value of node x_j, but are instead messages containing linear combinations of node x_j's value and the values of many other nodes in the network. Furthermore, the value of node x_j will pass through the malicious nodes multiple times in many different linear combinations due to cycles in the network and the memory inherent in the linear iterative strategy. It will become apparent from our development that the mechanics of the linear iterative strategy will force the malicious nodes to update their values in a clever (and nontrivial) manner in order to avoid detection and prevent function calculation in the network. Furthermore, our result applies regardless of how each node analyzes the information that it receives from the linear iteration, of the choice of weights for the linear iteration, and of the number of time-steps for which the iteration is run. In other words, our results apply to *any* linear iterative strategy with constant weights.[1] As we will show, our strategy for the malicious

[1] A similar analysis can be performed for linear iterative strategies with time-varying weights, but we will

nodes does not require them to send different information to different neighbors in order to disrupt the network, and can therefore be applied in networks with the wireless broadcast model (where transmissions by each node are heard by all of its neighbors).

Despite the above (negative) result, we also show that the linear iterative strategy is actually *robust* to arbitrary malicious behavior on the part of a fixed number of nodes (under the wireless broadcast model) if the connectivity of the network is sufficiently high. Specifically, we show that if the network contains at least $2f+1$ paths from any node to a given node x_i, then f malicious nodes *cannot* prevent x_i from calculating *any* function of the initial values when the linear iterative strategy is used, even if they update their values in a completely arbitrary and coordinated manner. Since *any* distributed function calculation strategy requires a connectivity of at least $2f+1$ in order to tolerate up to f malicious nodes (as described above and in [71]), our results show that linear iterative strategies are as powerful as any other strategy for overcoming malicious behavior, under the condition that the malicious nodes cannot send conflicting values to different neighbors.[2]

4.3 Modeling Malicious Behavior and Main Results

Suppose the objective in the system is for node x_i to calculate $g_i(x_1[0], x_2[0], \ldots, x_N[0])$, for some function $g_i : \mathbb{R}^N \rightarrow \mathbb{R}^{r_i}$ (note that different nodes can calculate different functions). When there are no malicious nodes in the network, we saw in Chapter 2 that this can be accomplished by having the nodes run the linear iteration $\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k]$ with almost any real-valued weight matrix $\mathbf{W}$ (conforming to the network topology) for a finite number of time-steps. Suppose, however, that instead of simply updating their values as a predetermined linear combination of the values in their neighborhood, some nodes update their values incorrectly; for example, node x_l updates its value at each time-step as

$$x_l[k+1] = w_{ll}x_l[k] + \sum_{x_j \in \mathcal{N}_l} w_{lj}x_j[k] + u_l[k] \ , \tag{4.1}$$

where $u_l[k]$ is the additive error at time-step k.

Definition 4.1 *Suppose all nodes run the linear iteration for T time-steps in order to perform function calculation. Node x_l is said to be malicious (or faulty) if $u_l[k]$ is nonzero for at least one time-step k, $0 \leq k \leq T-1$.*

Remark 4.2 *Note that the fault model considered here is extremely general, and allows node x_l to update its value in a completely arbitrary manner (via appropriate choices of the error $u_l[k]$ at each time-step). As such, this fault model is capable of encapsulating a*

not treat this topic here in the interest of clarity.

[2]Note that dealing with f Byzantine nodes that can send different values to different neighbors would require the network to have at least $3f+1$ nodes, which is not a constraint that we require in our setting.

wide variety of faults, with the only restriction being that each malicious node sends the same value to all of its neighbors. For example, suppose a node x_l in the network exhibits a stopping failure, whereby it stops transmitting information (so that the neighbors of node x_l simply receive the value zero from node x_l at each time-step). This stopping failure can be captured by our fault model by selecting the error $u_l[k]$ at each time-step to set node x_l's state $x_l[k+1]$ to zero. Another example of a fault is when the link from node x_j to node x_l drops out at time-step k. One can capture these errors in our fault model by simply selecting $u_l[k]$ to cancel out the term $w_{lj}x_j[k]$ in the update equation for node x_l. Note that we will have to assume an upper bound on the amount of time required to transmit messages in order to deal with these cases, since otherwise, the receiving node would not know whether the sending node/link has failed, or if the message is simply delayed [11]. Also note that if these errors are not accidental, but intentional, the corresponding node will be called malicious. More generally, one could have multiple faulty and/or malicious nodes (the latter could be coordinating their actions to disrupt the network). In the rest of the chapter, we will be using the terms faulty *and* malicious *interchangeably. We will also assume that malicious nodes are omniscient, and know the network topology along with the weights that are being used by the nodes in the network. They are also allowed to know all of the initial values in the network (they will not require these initial values in order to disrupt the network, however, as we will show in the next section).*

Let $\mathcal{F} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_f}\}$ denote the set of nodes that are malicious during a run of the linear iteration. Combining (4.1) with (1.1), the linear iteration can then be modeled as

$$\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] + \underbrace{\begin{bmatrix}\mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_f,N}\end{bmatrix}}_{\mathbf{B}_{\mathcal{F}}} \underbrace{\begin{bmatrix} u_{i_1}[k] \\ u_{i_2}[k] \\ \vdots \\ u_{i_f}[k] \end{bmatrix}}_{\mathbf{u}_{\mathcal{F}}[k]}$$

$$\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k], \;\; 1 \leq i \leq N \;\; . \tag{4.2}$$

Recall that $\mathbf{e}_{l,N}$ denotes a vector of length N with a single nonzero entry with value 1 in its l–th position, and that the vector $\mathbf{y}_i[k]$ denotes the set of outputs (or node values) seen by node x_i during time-step k of the linear iteration (the matrix $\mathbf{C}_i$ is a $(\deg_i + 1) \times N$ matrix with a single 1 in each row capturing the positions of the state-vector $\mathbf{x}[k]$ that are available to node x_i). The set of all values seen by node x_i during the first $L+1$ time-steps of the

linear iteration (for any nonnegative integer L) is given by

$$\underbrace{\begin{bmatrix}\mathbf{y}_i[0]\\ \mathbf{y}_i[1]\\ \mathbf{y}_i[2]\\ \vdots\\ \mathbf{y}_i[L]\end{bmatrix}}_{\mathbf{y}_i[0:L]} = \underbrace{\begin{bmatrix}\mathbf{C}_i\\ \mathbf{C}_i\mathbf{W}\\ \mathbf{C}_i\mathbf{W}^2\\ \vdots\\ \mathbf{C}_i\mathbf{W}^L\end{bmatrix}}_{\mathcal{O}_{i,L}} \mathbf{x}[0] + \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{C}_i\mathbf{B}_{\mathcal{F}} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\mathcal{F}} & \mathbf{C}_i\mathbf{B}_{\mathcal{F}} & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\mathcal{F}} & \mathbf{C}_i\mathbf{W}^{L-2}\mathbf{B}_{\mathcal{F}} & \cdots & \mathbf{C}_i\mathbf{B}_{\mathcal{F}}\end{bmatrix}}_{\mathcal{M}_{i,L}^{\mathcal{F}}} \underbrace{\begin{bmatrix}\mathbf{u}_{\mathcal{F}}[0]\\ \mathbf{u}_{\mathcal{F}}[1]\\ \mathbf{u}_{\mathcal{F}}[2]\\ \vdots\\ \mathbf{u}_{\mathcal{F}}[L-1]\end{bmatrix}}_{\mathbf{u}_{\mathcal{F}}[0:L-1]}. \tag{4.3}$$

The matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{F}}$ will characterize the ability of node x_i to calculate the required function of the initial values; we will call $\mathcal{M}_{i,L}^{\mathcal{F}}$ the *invertibility matrix*[3] for the triplet $(\mathbf{W}, \mathbf{B}_{\mathcal{F}}, \mathbf{C}_i)$; note that a matrix of this form also appeared in Equation (3.3) in Chapter 3 due to noise in the network. While the characteristics of this matrix were not important in that context, it will turn out to be crucial when analyzing the susceptibility of linear iterative strategies to malicious behavior (as we will see in this chapter and the next). In our development, we will use the fact that matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{F}}$ can be expressed recursively as

$$\mathcal{O}_{i,L} = \begin{bmatrix}\mathbf{C}_i\\ \mathcal{O}_{i,L-1}\mathbf{W}\end{bmatrix}, \quad \mathcal{M}_{i,L}^{\mathcal{F}} = \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathcal{O}_{i,L-1}\mathbf{B}_{\mathcal{F}} & \mathcal{M}_{i,L-1}^{\mathcal{F}}\end{bmatrix}, \tag{4.4}$$

where $\mathcal{O}_{i,0} = \mathbf{C}_i$ and $\mathcal{M}_{i,0}^{\mathcal{F}}$ is the empty matrix (with zero columns). We will also call upon the following simple lemma.

Lemma 4.1 *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ denote two subsets of $\mathcal{X}$, and define $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$. Then, the column space of $\mathcal{M}_{i,L}^{\mathcal{F}}$ is the same as the column space of $\begin{bmatrix}\mathcal{M}_{i,L}^{\mathcal{F}_1} & \mathcal{M}_{i,L}^{\mathcal{F}_2}\end{bmatrix}$ for any nonnegative integer L.*

Proof: Let

$$\mathcal{F}_1 = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{|\mathcal{F}_1|}}\}, \quad \mathcal{F}_2 = \{x_{j_1}, x_{j_2}, \ldots, x_{j_{|\mathcal{F}_2|}}\}, \quad \mathcal{F} = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|\mathcal{F}|}}\},$$

so that

$$\mathbf{B}_{\mathcal{F}_1} = \begin{bmatrix}\mathbf{e}_{i_1,N} & \cdots & \mathbf{e}_{i_{|\mathcal{F}_1|},N}\end{bmatrix}, \quad \mathbf{B}_{\mathcal{F}_2} = \begin{bmatrix}\mathbf{e}_{j_1,N} & \cdots & \mathbf{e}_{j_{|\mathcal{F}_2|},N}\end{bmatrix}, \mathbf{B}_{\mathcal{F}} = \begin{bmatrix}\mathbf{e}_{l_1,N} & \cdots & \mathbf{e}_{l_{|\mathcal{F}|},N}\end{bmatrix}.$$

[3]This terminology is due to the fact that matrices of the form $\mathcal{M}_{i,L}^{\mathcal{F}}$ arise in the study of *dynamic system inversion*, where the objective is to recover a set of unknown inputs from the output of a linear system [43]; see the discussion in Section 1.5. We will discuss the topic of dynamic system inversion in more detail in Chapter 5.

Since $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$, we have $\text{range}\left(\mathcal{O}_{i,L}\mathbf{B}_{\mathcal{F}}\right) = \text{range}\left(\mathcal{O}_{i,L}\begin{bmatrix}\mathbf{B}_{\mathcal{F}_1} & \mathbf{B}_{\mathcal{F}_2}\end{bmatrix}\right)$ for any $L \geq 0$ (this is easily seen from the structure of $\mathbf{B}_{\mathcal{F}_1}, \mathbf{B}_{\mathcal{F}_2}$ and $\mathbf{B}_{\mathcal{F}}$), and from this, we obtain

$$\begin{aligned}
\text{range}\left(\mathcal{M}_{i,L}^{\mathcal{F}}\right) &= \text{range}\left(\begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{B}_{\mathcal{F}} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\mathcal{F}} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\mathcal{F}} & \cdots & \mathbf{C}_i\mathbf{B}_{\mathcal{F}} \end{bmatrix}\right) \\
&= \text{range}\left(\begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\begin{bmatrix}\mathbf{B}_{\mathcal{F}_1} & \mathbf{B}_{\mathcal{F}_2}\end{bmatrix} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{W}\begin{bmatrix}\mathbf{B}_{\mathcal{F}_1} & \mathbf{B}_{\mathcal{F}_2}\end{bmatrix} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_i\mathbf{W}^{L-1}\begin{bmatrix}\mathbf{B}_{\mathcal{F}_1} & \mathbf{B}_{\mathcal{F}_2}\end{bmatrix} & \cdots & \mathbf{C}_i\begin{bmatrix}\mathbf{B}_{\mathcal{F}_1} & \mathbf{B}_{\mathcal{F}_2}\end{bmatrix} \end{bmatrix}\right) \\
&= \text{range}\left(\begin{bmatrix}\mathcal{M}_{i,L}^{\mathcal{F}_1} & \mathcal{M}_{i,L}^{\mathcal{F}_2}\end{bmatrix}\right) \ .
\end{aligned}$$

■

Our goal in this chapter will be to study the robustness of linear-iterative strategies to malicious or faulty behavior by a subset of nodes in the network. Specifically, over the remainder of the chapter, we demonstrate the following key results.

Theorem 4.1 *Given a fixed network described by a graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, let κ_{ij} denote the size of the smallest ij-cut between vertices x_i and x_j. If $\kappa_{ij} \leq 2f$ for some positive integer f, then there exists a set $\mathcal{F} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_f}\}$ of f malicious nodes along with a sequence of malicious updates $u_{i_1}[k], u_{i_2}[k], \ldots, u_{i_f}[k]$, $k = 0, 1, \ldots$, for each malicious node such that the values seen by node x_i during the linear iteration do not provide it with any information about the initial value of node x_j.*

Theorem 4.2 *Given a fixed network described by a graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, let f denote the maximum number of malicious or faulty nodes that are to be tolerated in the network, and let κ_{ij} denote the size of the smallest ij-cut between any two vertices x_i and x_j. Define*

$$\mathcal{T} = \{x_i \mid \kappa_{ij} \geq 2f + 1 \text{ for all } x_j \in \mathcal{X}\} \ .$$

Then, with almost any choice of real-valued weight matrix (with $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i$), every node in $\mathcal{T}$ can uniquely determine all of the initial values in the network after running the linear iteration for at most N time-steps, regardless of the updates applied by up to f malicious or faulty nodes.

Together, these theorems can be encapsulated into the following result.

Theorem 4.3 *Given a fixed network described by a graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, let κ_{ij} denote the size of the smallest ij-cut between any two vertices x_i and x_j. Then, node x_i can uniquely determine all initial values in the network via a linear iterative strategy, regardless of the update strategies used by up to f malicious nodes, if and only if x_i has $2f + 1$ vertex-disjoint paths from every other node in the network. If this condition is satisfied, node x_i can accomplish this by running the linear iteration with almost any choice of real-valued weight matrix (with $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i$) for at most N time-steps.*

Note that when we talk about tolerating a set $\mathcal{F}$ of faulty or malicious nodes, our focus is on eliminating the effect of the term $\mathbf{u}_{\mathcal{F}}[k]$ in Equation (4.2). However, a set of malicious nodes can obviously try to influence the result of a computation by changing their own *initial* values. Of course, if all initial values are equally valid, then it will be impossible for *any* protocol to detect this type of malicious behavior, since any initial value that a malicious node chooses for itself is a legitimate value that could also have been chosen by a node that is functioning correctly. On the other hand, if there is a statistical distribution on the initial values, techniques that exploit these relationships among the initial values can potentially be used to identify outliers due to malicious nodes and eliminate their influence on the system [76, 77, 78]. Similarly, if the nodes in the network are assumed to be rational and are trying to maximize some objective function that depends on the initial values in the network, one can potentially design a *mechanism* for the network that incentivizes all nodes to report their values truthfully [79]. We will not discuss the issue of verifying initial values further in our work because of its philosophically different nature, and because of the fact that our problem formulation remains valid in cases where malicious nodes do not contribute initial values (i.e., they function as routers – the functions calculated in the network do not depend on the initial values of these nodes). Instead, our goal is to avoid the more pernicious case where malicious nodes spread confusion about the initial values of *non-malicious* nodes; the following scenarios illustrate cases where this issue is of primary importance.

Scenario 1: Distributed Voting. Consider a distributed system consisting of agents that are trying to vote on a course of action. In our framework, the vote of each agent can be modeled by setting that agent's initial value to be 1 for 'yes' and 0 for 'no'. The agents wish to reach consensus on the majority vote in a distributed manner (i.e., by exchanging their values via other agents in the network). In this scenario, both 0 and 1 are equally valid votes for every agent, and so it is not important whether malicious agents change their own initial value (i.e., their vote). What *is* important, however, is to ensure that votes are not changed as they are passed through other agents in the network; in other words, the malicious nodes should not be able to affect the result of the majority vote other than by choosing their own vote for the course of action.

Scenario 2: Distributed Auction. Consider a distributed system consisting of agents that are participating in an auction. In our framework, the initial value of each agent represents its bid for the object that is being auctioned. The agents wish to pass their bids through the network (via the other agents) to the auctioneer. There is a large amount of work in the economics literature dealing with formulating incentives for the individual agents that will cause them to truthfully report their bid (i.e., their initial value). For example, one prominent mechanism is the so-called *Vickrey second-price auction*, whereby the object goes to the agent who makes the highest bid, but that agent only has to pay the value of the second-highest bid. One can show that, under this mechanism, no agent can gain by misreporting their bid [80], and thus there is no need to consider the problem of malicious nodes changing their own initial value (or bid). However, in a distributed setting, there is still the problem of malicious agents colluding and changing the values of bids that they are passing to the auctioneer. In such cases, one requires a method to disseminate the bids correctly through the network in order to facilitate the Vickrey auction.

The model for malicious behavior that we have introduced in this section can readily capture the malevolent behavior represented in the above scenarios. Theorems 4.1 and 4.2 thus demonstrate that linear iterative strategies can be used to disseminate information in these (and other) scenarios, provided that the topology of the distributed system contains sufficiently many node-disjoint paths between nodes in the system.

4.4 Attacking Linear Iterative Strategies with Malicious Nodes

In this section, we derive a lower bound on the connectivity of any node x_i in order for it to calculate an arbitrary function of the node values in the presence of faulty or malicious nodes via a linear iterative strategy (by the end of the section, we will have obtained the proof of Theorem 4.1). We start by establishing a relationship between the column space of the invertibility matrices and the column space of the observability matrix for certain nodes in the network. We will use the following terminology in our discussion.

Definition 4.2 *Let $\mathcal{R} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{|\mathcal{R}|}}\}$ denote a set of nodes in the network, and let x_i be any node in the network. The columns of the observability matrix $\mathcal{O}_{i,L}$ corresponding to nodes in $\mathcal{R}$ are given by $\mathcal{O}_{i,L}\begin{bmatrix}\mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_{|\mathcal{R}|},N}\end{bmatrix}$.*

Consider node x_i in the network $\mathcal{G}$, and let node x_j be any other node in the network that is not a neighbor of node x_i. Let $\mathcal{F}_1 = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|\mathcal{F}_1|}}\}$ and $\mathcal{F}_2 = \{x_{h_1}, x_{h_2}, \ldots, x_{h_{|\mathcal{F}_2|}}\}$ denote disjoint sets of vertices such that $\mathcal{F}_1 \cup \mathcal{F}_2$ forms an ij-cut of $\mathcal{G}$. Let $\mathcal{H}$ denote the set of all nodes that have a path to node x_i in the graph induced by $\mathcal{X} \setminus (\mathcal{F}_1 \cup \mathcal{F}_2)$ (including node x_i), and let $\bar{\mathcal{H}} = \mathcal{X} \setminus (\mathcal{H} \cup \mathcal{F}_1 \cup \mathcal{F}_2)$.

Theorem 4.4 *For any nonnegative integer L, the columns of the observability matrix $\mathcal{O}_{i,L}$ corresponding to the nodes in set $\bar{\mathcal{H}}$ can be written as a linear combination of the columns in the matrices $\mathcal{M}_{i,L}^{\mathcal{F}_1}$ and $\mathcal{M}_{i,L}^{\mathcal{F}_2}$.*

Proof: Let $\mathbf{x}_{\mathcal{H}}[k]$ denote the vector of values of nodes in set $\mathcal{H}$, $\mathbf{x}_{\mathcal{F}_1}[k]$ denote the vector of values of nodes in set $\mathcal{F}_1$, $\mathbf{x}_{\mathcal{F}_2}[k]$ denote the vector of values of nodes in set $\mathcal{F}_2$, and $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ denote the vector of values of nodes in set $\bar{\mathcal{H}}$. Note that $x_i[k]$ is contained in $\mathbf{x}_{\mathcal{H}}[k]$, $x_j[k]$ is contained in $\mathbf{x}_{\bar{\mathcal{H}}}[k]$, and that the sets $\mathcal{H}, \mathcal{F}_1, \mathcal{F}_2$, and $\bar{\mathcal{H}}$ are disjoint (they actually form a partition of the set of nodes $\mathcal{X}$). Assume without loss of generality that the vector $\mathbf{x}[k]$ in (4.2) is of the form

$$\mathbf{x}[k] = \begin{bmatrix} \mathbf{x}'_{\mathcal{H}}[k] & \mathbf{x}'_{\mathcal{F}_1}[k] & \mathbf{x}'_{\mathcal{F}_2}[k] & \mathbf{x}'_{\bar{\mathcal{H}}}[k] \end{bmatrix}'$$

(it can always be put into this form via an appropriate permutation of the node indices). Then, since no node in set $\mathcal{H}$ has an incoming edge from any node in set $\bar{\mathcal{H}}$ (otherwise, there would be a path from a node in $\bar{\mathcal{H}}$ to node x_i), the weight matrix for the linear iteration must necessarily have the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} & \mathbf{0} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} & \mathbf{W}_{24} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} & \mathbf{W}_{34} \\ \mathbf{W}_{41} & \mathbf{W}_{42} & \mathbf{W}_{43} & \mathbf{W}_{44} \end{bmatrix} . \tag{4.5}$$

The $\mathbf{C}_i$ matrix in (4.2) for node x_i must be of the form $\mathbf{C}_i = \begin{bmatrix} \mathbf{C}_{i,1} & \mathbf{C}_{i,2} & \mathbf{C}_{i,3} & \mathbf{0} \end{bmatrix}$, again because node x_i has no neighbors in set $\bar{\mathcal{H}}$. Furthermore, from the definition of the matrix $\mathbf{B}_{\mathcal{F}}$ in (4.2), note that this ordering of nodes implies that we can write $\mathbf{B}_{\mathcal{F}_1} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{|\mathcal{F}_1|} & \mathbf{0} & \mathbf{0} \end{bmatrix}'$, $\mathbf{B}_{\mathcal{F}_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I}_{|\mathcal{F}_2|} & \mathbf{0} \end{bmatrix}'$.

Let n denote the number of nodes in set $\bar{\mathcal{H}}$ (i.e., $\mathbf{x}_{\bar{\mathcal{H}}}[k] \in \mathbb{R}^n$). For any nonnegative integer L, the set of columns of the observability matrix $\mathcal{O}_{i,L}$ corresponding to $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ is given by $\mathcal{O}_{i,L} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix}$. Using the recursive definition of $\mathcal{O}_{i,L}$ in (4.4), and the fact that $\mathbf{C}_i \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \mathbf{0}$, we obtain

$$\mathcal{O}_{i,L} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \mathbf{C}_i \\ \mathcal{O}_{i,L-1}\mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{W} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_{24} \\ \mathbf{W}_{34} \\ \mathbf{W}_{44} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\mathcal{F}_1} \mathbf{W}_{24} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\mathcal{F}_2} \mathbf{W}_{34} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} \mathbf{W}_{44} \ .$$

Applying the above procedure recursively for the matrix $\begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix}$, we obtain

$$\mathcal{O}_{i,L} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\mathcal{F}_1} \mathbf{W}_{24} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\mathcal{F}_2} \mathbf{W}_{34} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-2} \end{bmatrix} \mathbf{B}_{\mathcal{F}_1} \mathbf{W}_{24} \mathbf{W}_{44}$$
$$+ \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-2} \end{bmatrix} \mathbf{B}_{\mathcal{F}_2} \mathbf{W}_{34} \mathbf{W}_{44} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-2} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} \mathbf{W}_{44}^2.$$

Continuing the above procedure recursively for matrices of the form $\begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-\alpha} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix}$, $2 \leq \alpha \leq L$, we obtain (after some algebra)

$$\begin{aligned}
\mathcal{O}_{i,L} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} &= \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1}\mathbf{B}_{\mathcal{F}_1} \end{bmatrix} \mathbf{W}_{24} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-2}\mathbf{B}_{\mathcal{F}_1} \end{bmatrix} \mathbf{W}_{24}\mathbf{W}_{44} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-3}\mathbf{B}_{\mathcal{F}_1} \end{bmatrix} \mathbf{W}_{24}\mathbf{W}_{44}^2 + \cdots \\
&\quad + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,1}\mathbf{B}_{\mathcal{F}_1} \end{bmatrix} \mathbf{W}_{24}\mathbf{W}_{44}^{L-2} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,0}\mathbf{B}_{\mathcal{F}_1} \end{bmatrix} \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \\
&\quad + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1}\mathbf{B}_{\mathcal{F}_2} \end{bmatrix} \mathbf{W}_{34} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-2}\mathbf{B}_{\mathcal{F}_2} \end{bmatrix} \mathbf{W}_{34}\mathbf{W}_{44} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-3}\mathbf{B}_{\mathcal{F}_2} \end{bmatrix} \mathbf{W}_{34}\mathbf{W}_{44}^2 + \cdots \\
&\quad + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,1}\mathbf{B}_{\mathcal{F}_2} \end{bmatrix} \mathbf{W}_{34}\mathbf{W}_{44}^{L-2} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,0}\mathbf{B}_{\mathcal{F}_2} \end{bmatrix} \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \\
&= \mathcal{M}_{i,L}^{\mathcal{F}_1} \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_2} \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} \ . \qquad (4.6)
\end{aligned}$$

In the last step above, we have used the recursive definition of the matrices $\mathcal{M}_{i,L}^{\mathcal{F}_1}$ and $\mathcal{M}_{i,L}^{\mathcal{F}_2}$ from Equation (4.4). This concludes the proof of the theorem. ■

Note that the above theorem applies to *any* decomposition of a vertex-cut into sets $\mathcal{F}_1$ and $\mathcal{F}_2$ (including the case where $\mathcal{F}_1$ is the entire vertex cut and $\mathcal{F}_2 = \emptyset$, if desired). By framing the theorem in this general manner, we will later have freedom to choose the sets $\mathcal{F}_1$ and $\mathcal{F}_2$ as part of an appropriate vertex-cut in order to demonstrate how a set of malicious nodes can disrupt the linear iterative strategy. Specifically, expression (4.6) will allow us to show how a certain set of nodes $\mathcal{F}_1$ (or $\mathcal{F}_2$) can maliciously update their values so that some node x_i cannot obtain any information about the initial values of some other nodes in the network. This is precisely the subject of the following lemma.

Lemma 4.2 *If nodes in set $\mathcal{F}_1$ are malicious, it is possible for them to update their values in such a way that the values seen by node x_i (over any number of time-steps of the linear iteration) are indistinguishable from the values seen by node x_i when nodes in set $\mathcal{F}_2$ are malicious. Furthermore, these indistinguishable faults make it impossible for node x_i to determine the initial values of nodes in the set $\bar{\mathcal{H}}$.*

Proof: As in the proof of Theorem 4.4, let $\mathbf{x}_{\mathcal{H}}[k], \mathbf{x}_{\mathcal{F}_1}[k]$, $\mathbf{x}_{\mathcal{F}_2}[k]$, and $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ denote the vector of values of nodes in sets $\mathcal{H}$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\bar{\mathcal{H}}$, respectively, and assume (without loss of generality) that the vector $\mathbf{x}[k]$ in (4.2) is of the form

$$\mathbf{x}[k] = \begin{bmatrix} \mathbf{x}'_{\mathcal{H}}[k] & \mathbf{x}'_{\mathcal{F}_1}[k] & \mathbf{x}'_{\mathcal{F}_2}[k] & \mathbf{x}'_{\bar{\mathcal{H}}}[k] \end{bmatrix}'.$$

Let n be the number of nodes in set $\bar{\mathcal{H}}$ and let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ be arbitrary vectors. We will now show that the values seen by node x_i when nodes in $\mathcal{F}_1$ are malicious and $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$ will be indistinguishable from the values seen by node x_i when nodes in $\mathcal{F}_2$ are malicious and $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$. This will imply that node x_i cannot determine whether the initial values of nodes in $\bar{\mathcal{H}}$ are given by vector $\mathbf{a}$ or vector $\mathbf{b}$.

To this end, suppose the nodes in set $\mathcal{F}_1$ are malicious. From (4.3), the values seen by node x_i over $L+1$ time-steps are given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{F}_1}\mathbf{u}_{\mathcal{F}_1}[0:L-1].$$

From Theorem 4.4 (specifically, Equation (4.6)), this expression can be written as

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\begin{bmatrix} \mathbf{x}_{\mathcal{H}}[0] \\ \mathbf{x}_{\mathcal{F}_1}[0] \\ \mathbf{x}_{\mathcal{F}_2}[0] \\ \mathbf{0} \end{bmatrix} + \left(\mathcal{M}_{i,L}^{\mathcal{F}_1}\begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_2}\begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} \right) \mathbf{x}_{\bar{\mathcal{H}}}[0] + \mathcal{M}_{i,L}^{\mathcal{F}_1}\mathbf{u}_{\mathcal{F}_1}[0:L-1]. \quad (4.7)$$

Suppose $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$, and that nodes in $\mathcal{F}_1$ update their values at each time-step k with the error values $\mathbf{u}_{\mathcal{F}_1}[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k(\mathbf{b} - \mathbf{a})$, producing the error vector

$$\mathbf{u}_{\mathcal{F}_1}[0:L-1] = \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} (\mathbf{b} - \mathbf{a}) \ . \tag{4.8}$$

Substituting this into the expression for $\mathbf{y}_i[0:L]$ (with $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$), the values seen by node x_i under this fault scenario are given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L} \begin{bmatrix} \mathbf{x}_{\mathcal{H}}[0] \\ \mathbf{x}_{\mathcal{F}_1}[0] \\ \mathbf{x}_{\mathcal{F}_2}[0] \\ \mathbf{0} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_1} \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} \mathbf{b} + \mathcal{M}_{i,L}^{\mathcal{F}_2} \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} \mathbf{a} \ . \tag{4.9}$$

Now suppose that nodes in $\mathcal{F}_2$ are malicious (instead of nodes in $\mathcal{F}_1$). Again, from (4.3) and Theorem 4.4, the values seen by node x_i over $L+1$ time-steps will be given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L} \begin{bmatrix} \mathbf{x}_{\mathcal{H}}[0] \\ \mathbf{x}_{\mathcal{F}_1}[0] \\ \mathbf{x}_{\mathcal{F}_2}[0] \\ \mathbf{0} \end{bmatrix} + \left(\mathcal{M}_{i,L}^{\mathcal{F}_1} \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_2} \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} \right) \mathbf{x}_{\bar{\mathcal{H}}}[0] + \mathcal{M}_{i,L}^{\mathcal{F}_2} \mathbf{u}_{\mathcal{F}_2}[0:L-1]. \tag{4.10}$$

If $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$, and nodes in $\mathcal{F}_2$ update their values at each time-step k with the error values $\mathbf{u}_{\mathcal{F}_2}[k] = \mathbf{W}_{34}\mathbf{W}_{44}^k(\mathbf{a} - \mathbf{b})$, the error vector $\mathbf{u}_{\mathcal{F}_2}[0:L-1]$ will have the form

$$\mathbf{u}_{\mathcal{F}_2}[0:L-1] = \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} (\mathbf{a} - \mathbf{b}) \ .$$

Substituting this into Equation (4.10) with $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$, one can verify that the set of values seen by node x_i under this fault scenario will be identical to the expression in (4.9), and thus the values received by node x_i when $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$ and the nodes in $\mathcal{F}_1$ are malicious will be indistinguishable from the values seen by node x_i when $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$ and the nodes in $\mathcal{F}_2$ are malicious. Since this holds for all nonnegative integers L, this fault scenario makes it impossible for node x_i (and in fact, any node in set $\mathcal{H}$) to obtain the initial values of node x_j (or any other node in set $\bar{\mathcal{H}}$). ■

Remark 4.3 *Equation* (4.8) *provides the additive errors for nodes in set* $\mathcal{F}_1$ *to use in updating their values in order to disrupt the linear iterative strategy; note that these nodes do not actually need to know the initial values of the nodes in set* $\bar{\mathcal{H}}$ *(taken to be* $\mathbf{a}$ *in the above exposition) in order to apply this strategy. If they simply choose any random vector* $\bar{\mathbf{a}}$*, and update their values at each time-step as* $\mathbf{u}_{\mathcal{F}_1}[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k\bar{\mathbf{a}}$*, then they are effectively applying the update* (4.8) *with* $\mathbf{b} = \mathbf{a} + \bar{\mathbf{a}}$*, without knowing the value of* $\mathbf{a}$*. The same reasoning holds if nodes in set* $\mathcal{F}_2$ *are malicious.*

Remark 4.4 *Note also that the above strategies for the malicious nodes apply regardless of the choice of weight matrix. This means that the observability properties of the system are irrelevant when considering the ability of malicious nodes to disrupt the network. Indeed, if the system is unobservable from node* x_i*, the malicious nodes will have to do less work in order to disrupt the network since node* x_i *will already be unable to determine the initial values of certain other nodes in the network (even when there are no faults).*

We are now ready to prove Theorem 4.1 (provided at the end of Section 4.3).

Proof: In Lemma 4.2, we saw that if the union of the disjoint sets of vertices

$$\mathcal{F}_1 = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|S_1|}}\},\ \ \mathcal{F}_2 = \{x_{h_1}, x_{h_2}, \ldots, x_{h_{|S_2|}}\}$$

forms an ij-cut, then node x_i cannot distinguish a particular set of errors by nodes in $\mathcal{F}_1$ from another set of errors by nodes in $\mathcal{F}_2$. Furthermore, these errors make it impossible for node x_i to obtain any information about the initial value of node x_j. Choose $\mathcal{F}_1$ and $\mathcal{F}_2$ such that $|\mathcal{F}_1| = \lfloor \frac{\kappa_{ij}}{2} \rfloor$ and $|\mathcal{F}_2| = \lceil \frac{\kappa_{ij}}{2} \rceil$. Since $\kappa_{ij} \leq 2f$, we have $\lfloor \frac{\kappa_{ij}}{2} \rfloor \leq f$ and $\lceil \frac{\kappa_{ij}}{2} \rceil \leq f$, and so $\mathcal{F}_1$ and $\mathcal{F}_2$ are both legitimate sets of malicious nodes (if one is interested in tolerating a maximum of f malicious nodes in the system). Thus, if $\kappa_{ij} \leq 2f$, it is possible for a particular set of f malicious nodes to update their values in such a way that node x_i cannot obtain any information about node x_j's value. ■

Remark 4.5 *Clearly, the above theorem can easily be modified to show that errors by a set* $\mathcal{F}_1$ *of* $f + t$ *malicious nodes would be indistinguishable from errors by a set* $\mathcal{F}_2$ *of* $f - t$ *malicious nodes, for any* $0 \leq t \leq f$*. The reason for focusing on the case* $t = 0$ *is that we are interested in dealing with a maximum number of malicious nodes in the network. In other words, if we know that a malicious attacker can only target at most* f *nodes (e.g., due to the costs incurred by attacking a node), then we can disregard all cases where more than* f *nodes are potentially malicious. Thus, even though we can show that a set* $\mathcal{F}_1$ *of* $f - t$ *malicious nodes can masquerade as a set* $\mathcal{F}_2$ *of* $f + t$ *malicious nodes, the latter case can be discounted because of the fact that* $f + t$ *nodes are not likely to be malicious, and so we can potentially identify the set* $\mathcal{F}_1$*. On the other hand, when* $t = 0$*,* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *are equally valid (and likely) sets of* f *malicious nodes, and thus we cannot discount one over the other.*

4.4.1 Example

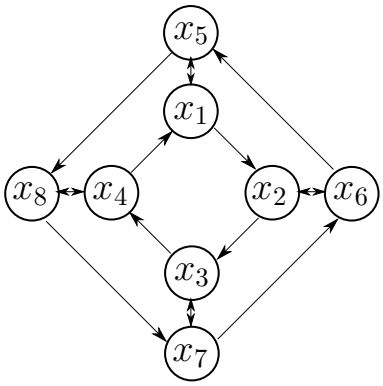

Figure 4.1: Network considered in [75]. Node x_6 is malicious, and will try to prevent node x_5 from calculating the average of the initial values.

Consider the network shown in Figure 4.1; this is the same network considered in [75]. The objective in this network is for all nodes to calculate the average of the initial values via a linear iterative strategy.

Absence of Malicious Nodes

The weights for the linear iteration are taken to be $w_{ij} = \frac{1}{3}$ if $x_j \in \mathcal{N}_i \cup \{x_i\}$, and 0 otherwise; these weights are the so-called *Metropolis-Hastings weights* for this network, and as shown in [22, 64, 75], they allow the nodes to reach asymptotic consensus on the average when there are no malicious nodes. We will now demonstrate an instance of finite-time consensus using these weights, and then show how a single malicious node can disrupt the linear iteration.

For brevity, consider node x_5 in the network. At each time-step, node x_5 has access to its own value and the values of nodes x_1 and x_6; the matrix $\mathbf{C}_5$ (which encapsulates the values that node x_5 receives at each time-step) is thus given by

$$\mathbf{C}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The first step is to find the smallest number of time-steps required by node x_5 to calculate the average. This is equivalent to finding the smallest nonnegative integer L_5 such that the row-space of the observability matrix $\mathcal{O}_{5,L_5}$ contains the vector $\frac{1}{8}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$. One can verify that this condition is not satisfied for the matrices $\mathcal{O}_{5,0} = \mathbf{C}_5$ and $\mathcal{O}_{5,1} = \begin{bmatrix} \mathbf{C}_5 \\ \mathbf{C}_5\mathbf{W} \end{bmatrix}$.

However, with $L_5 = 2$, the observability matrix for node x_5 is given by

$$\mathcal{O}_{5,2} = \begin{bmatrix} \mathbf{C}_5 \\ \mathbf{C}_5\mathbf{W} \\ \mathbf{C}_5\mathbf{W}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{2}{9} & 0 & \frac{1}{9} & \frac{2}{9} & \frac{2}{9} & \frac{1}{9} & 0 & \frac{1}{9} \\ \frac{2}{9} & \frac{1}{9} & 0 & \frac{1}{9} & \frac{2}{9} & \frac{2}{9} & \frac{1}{9} & 0 \\ \frac{1}{9} & \frac{2}{9} & \frac{1}{9} & 0 & 0 & \frac{2}{9} & \frac{2}{9} & \frac{1}{9} \end{bmatrix},$$

and one can verify that the row-space of this matrix contains the vector $\frac{1}{8}\mathbf{1}'$. In particular, we can find a matrix Γ_5 solving $\Gamma_5\mathcal{O}_{5,2} = \frac{1}{8}\mathbf{1}'$ as

$$\Gamma_5 = \frac{1}{8}\begin{bmatrix} -1 & 0 & 0 & 3 & 0 & -3 & 0 & 0 & 9 \end{bmatrix}.$$

This means that node x_5 can calculate the average of all initial values after $L_5 + 1 = 3$ iterations. A similar analysis is performed for each node in the network, and in this example, it is found that every node in the network can calculate the average after three iterations.

Now suppose that each node x_i is provided with the matrix Γ_i, and that the initial values of the nodes are

$$\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 & -3 & 0 \end{bmatrix}'.$$

The nodes run the linear iteration with the Metropolis-Hastings weights specified above, and at each time-step k, node x_5 receives the values $\mathbf{y}_5[k] = \mathbf{C}_5\mathbf{x}[k]$. For this instance of initial values, the values received by node x_5 over the first three iterations are $\mathbf{y}_5[0] = \begin{bmatrix} 3 & 7 & 11 \end{bmatrix}'$, $\mathbf{y}_5[1] = \begin{bmatrix} 2 & 7 & 2.33 \end{bmatrix}'$ and $\mathbf{y}_5[2] = \begin{bmatrix} 3 & 3.78 & 2.33 \end{bmatrix}'$. Node x_5 can now calculate the average as

$$\Gamma_5 \begin{bmatrix} \mathbf{y}_5[0] \\ \mathbf{y}_5[1] \\ \mathbf{y}_5[2] \end{bmatrix} = 2.125.$$

All of the other nodes calculate their averages in the same way, and thus the system reaches consensus on the average after three iterations. One can also verify that after an infinite number of iterations, the values of all the nodes converge to 2.125 (this is consistent with the fact that the choice of weights for the linear iteration allows the nodes to reach asymptotic consensus on the average).

Disrupting Function Calculation with One Malicious Node

We will now investigate what happens when some node in the network does not follow the linear iterative protocol. Consider node x_5 in the network, and note that $\min_j \kappa_{5j} = 2$; in other words, there are some nodes that can be disconnected from node x_5 by the removal of only two nodes (e.g., by removing nodes x_1 and x_6, node x_5 receives no information from any of the other nodes). Suppose that node x_6 is malicious in this network, and updates its values at each time-step as

$$x_6[k+1] = \frac{1}{3}x_2[k] + \frac{1}{3}x_6[k] + \frac{1}{3}x_7[k] + u_6[k] \ , \tag{4.11}$$

where $u_6[k]$ is the additive error at time-step k. It was shown in [75] that if $u_6[k]$ is constant for all k, then node x_5 can detect the fact that node x_6 is faulty and remove it from the network. However, according to our results in this chapter, if node x_6 chooses its error terms cleverly at each time-step, then it will forever evade identification by node x_5 and prevent it from calculating the average of all the values. We now provide the exact malicious update $u_6[k]$ at each time-step k for node x_6 to apply in order to disrupt the network in the above manner.

Define the sets $\mathcal{H} = \{x_5\}$, $\mathcal{F}_1 = \{x_6\}$, $\mathcal{F}_2 = \{x_1\}$ and $\bar{\mathcal{H}} = \{x_2, x_3, x_4, x_7, x_8\}$ (recall that the set $\mathcal{F}_1 \cup \mathcal{F}_2$ acts as a vertex cut, disconnecting the node in $\mathcal{H}$ from the nodes in $\bar{\mathcal{H}}$). Next, as specified in Theorem 4.4, reorder the nodes so that the nodes in $\mathcal{H}$ come first in the ordering, followed by $\mathcal{F}_1$, $\mathcal{F}_2$ and $\bar{\mathcal{H}}$; in accordance with Equation (4.5), the weight matrix $\mathbf{W}$ for this ordering has the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} & \mathbf{0} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} & \mathbf{W}_{24} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} & \mathbf{W}_{34} \\ \mathbf{W}_{41} & \mathbf{W}_{42} & \mathbf{W}_{43} & \mathbf{W}_{44} \end{bmatrix} = \frac{1}{3} \left[\begin{array}{c|c|c|cccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right] .$$

Now, according to Lemma 4.2, if node x_6 updates its values at each time-step with the additive error $u_6[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k(\mathbf{b} - \mathbf{x}_{\bar{\mathcal{H}}}[0])$ (where $\mathbf{x}_{\bar{\mathcal{H}}}[0]$ is the vector of initial values of nodes in $\bar{\mathcal{H}}$ and $\mathbf{b}$ is any arbitrary vector), then node x_5 will never be able to determine whether the initial values of the nodes in $\bar{\mathcal{H}}$ are in fact $\mathbf{x}_{\bar{\mathcal{H}}}[0]$ or $\mathbf{b}$, and whether node x_6 is malicious or node x_1 is malicious.

For example, suppose again that the initial values of the nodes are

$$\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 & -3 & 0 \end{bmatrix}' .$$

Since $\bar{\mathcal{H}} = \{x_2, x_3, x_4, x_7, x_8\}$, we have $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \begin{bmatrix} -1 & 4 & -4 & -3 & 0 \end{bmatrix}'$. Node x_6 would like to pretend that the initial values of the nodes in $\bar{\mathcal{H}}$ are actually given by the vector $\mathbf{b} = \begin{bmatrix} 5 & -3 & -2 & -1 & 3 \end{bmatrix}'$. Thus, at each time-step, node x_6 updates its value as

$$\begin{aligned} x_6[k+1] &= \frac{1}{3}x_2[k] + \frac{1}{3}x_6[k] + \frac{1}{3}x_7[k] + \mathbf{W}_{24}\mathbf{W}_{44}^k(\mathbf{b} - \mathbf{x}_{\bar{\mathcal{H}}}[0]) \\ &= \frac{1}{3}x_2[k] + \frac{1}{3}x_6[k] + \frac{1}{3}x_7[k] + \frac{1}{3^{k+1}} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}^k \begin{bmatrix} 6 \\ -7 \\ 2 \\ 2 \\ 3 \end{bmatrix} . \end{aligned}$$

Since node x_5 receives the values $\mathbf{y}_5[k] = \begin{bmatrix} x_1[k] & x_5[k] & x_6[k] \end{bmatrix}'$ at each time-step k, the explicit values seen by node x_5 over the linear iteration are given by

$$\mathbf{y}_5[0] = \begin{bmatrix} 3 \\ 7 \\ 11 \end{bmatrix}, \ \mathbf{y}_5[1] = \begin{bmatrix} 2 \\ 7 \\ 5 \end{bmatrix}, \ \mathbf{y}_5[2] = \begin{bmatrix} 3 \\ 4.67 \\ 3.67 \end{bmatrix}, \ \mathbf{y}_5[3] = \begin{bmatrix} 2.67 \\ 3.78 \\ 3 \end{bmatrix}, \ \cdots .$$

One can verify that these are exactly the same values seen by node x_5 when the initial values of nodes in $\bar{\mathcal{H}}$ are given by the vector $\mathbf{b}$ and node x_1 is malicious, with additive error $u_1[k] = \mathbf{W}_{34}\mathbf{W}_{44}^k(\mathbf{x}_{\bar{\mathcal{H}}}[0] - \mathbf{b})$. In other words node x_5 cannot distinguish the case when node x_6 is faulty from the case where node x_1 is faulty (with different initial values in the network). As discussed in Theorem 4.1, node x_6 can continue to update its values erroneously so that node x_5 can never resolve this ambiguity, regardless of the number of time-steps for which the linear iteration is run, and thus x_5 cannot correctly calculate the average of all the values in the network. It is of interest to note that this malicious behavior by node x_6 *cannot* be detected by the scheme in [75], where the asymptotic behavior of the malicious updates is used to detect and isolate malicious nodes. In this example, the additive error term $u_6[k]$ asymptotically decays to zero (and thus its asymptotic behavior cannot be used to deduce that node x_6 was malicious). Also note that despite the fact that the malicious behavior by node x_6 disappears asymptotically, its effect on the network remains; in this example, if the nodes continue to run the linear iteration, their values will all converge to the value 2.6875, which is different from the true average (2.1250) of the initial values.

4.5 Defending Linear Iterative Strategies Against Malicious Behavior

In the last section, we showed that when $\kappa_{ij} \leq 2f$ for some i and j, it is possible to find two sets of vertices $\mathcal{F}_1$ and $\mathcal{F}_2$ (each with cardinality f or less) such that node x_i cannot distinguish the case when nodes in $\mathcal{F}_1$ are malicious from the case when nodes in $\mathcal{F}_2$ are malicious. As a consequence, node x_i could not determine the initial value of node x_j, and therefore could not calculate the desired function of the initial values. In this section, we will show that if $\kappa_{ij} \geq 2f+1$ for all j, it will be impossible for any set of f or fewer malicious nodes to prevent node x_i from calculating any function of the initial values in the system in this way. We start our development with the following theorem, which provides a procedure for node x_i to calculate functions in the presence of malicious or faulty nodes. After the proof of the theorem, we will state the assumptions that we are making about the information that is available *a priori* to each node.

Theorem 4.5 *Suppose that there exists an integer L and a weight matrix $\mathbf{W}$ such that, for all possible sets $\mathcal{J}$ of $2f$ nodes, the matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{J}}$ for node x_i satisfy*

$$rank\left(\begin{bmatrix}\mathcal{O}_{i,L} & \mathcal{M}_{i,L}^{\mathcal{J}}\end{bmatrix}\right) = N + rank\left(\mathcal{M}_{i,L}^{\mathcal{J}}\right) \ . \tag{4.12}$$

Then, if the nodes run the linear iteration for $L+1$ time-steps with the weight matrix $\mathbf{W}$, node x_i can calculate any arbitrary function of the initial values $x_1[0], x_2[0], \ldots, x_N[0]$, even when up to f nodes are malicious.

Proof: Let $\mathbf{W}$ be a weight matrix that satisfies the conditions in the above theorem, and let the nodes run the linear iteration for $L+1$ time-steps. Suppose that the malicious nodes during the linear iteration are a subset of the set $\mathcal{F} = \{x_{j_1}, x_{j_2}, \ldots, x_{j_f}\}$. From (4.3), the values seen by node x_i over $L+1$ time-steps are given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[0:L-1] \ . \tag{4.13}$$

Let $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{\binom{N}{f}}$ denote all possible sets of f nodes, and let $\mathcal{M}_{i,L}^{\mathcal{F}_1}, \mathcal{M}_{i,L}^{\mathcal{F}_2}, \ldots, \mathcal{M}_{i,L}^{\mathcal{F}_{\binom{N}{f}}}$ denote the corresponding invertibility matrices. With these matrices in hand,[4] suppose node x_i finds the first $j \in \{1, 2, \ldots, \binom{N}{f}\}$ such that the vector $\mathbf{y}_i[0:L]$ is in the column space of the matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{F}_j}$. This means that node x_i can find vectors $\bar{\mathbf{x}}$ and $\mathbf{u}_{\mathcal{F}_j}[0:L-1]$ such that $\mathcal{O}_{i,L}\bar{\mathbf{x}} + \mathcal{M}_{i,L}^{\mathcal{F}_j}\mathbf{u}_{\mathcal{F}_j}[0:L-1] = \mathbf{y}_i[0:L]$. Equating this to (4.13) and

[4]Note from Equation (4.3) and Equation (4.4) that the columns of the invertibility matrix $\mathcal{M}_{i,L}^{\mathcal{F}_j}$ are simply a subset of the columns of $\mathcal{O}_{i,0}, \mathcal{O}_{i,1}, \ldots, \mathcal{O}_{i,L-1}$. In other words, node x_i can obtain the invertibility matrices simply from the knowledge of the matrix $\mathcal{O}_{i,L}$, and therefore does not necessarily need to store the invertibility matrices for every possible set of f nodes.

rearranging, we have

$$\mathcal{O}_{i,L}(\mathbf{x}[0] - \bar{\mathbf{x}}) + \mathcal{M}_{i,L}^{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[0:L-1] - \mathcal{M}_{i,L}^{\mathcal{F}_j}\mathbf{u}_{\mathcal{F}_j}[0:L-1] = \mathbf{0} \ .$$

Letting $\mathcal{J} = \mathcal{F} \cup \mathcal{F}_j$, we note from Lemma 4.1 that the above expression can be written as

$$\mathcal{O}_{i,L}(\mathbf{x}[0] - \bar{\mathbf{x}}) + \mathcal{M}_{i,L}^{\mathcal{J}}\mathbf{u}_{\mathcal{J}}[0:L-1] = \mathbf{0} \ ,$$

for some appropriately defined vector $\mathbf{u}_{\mathcal{J}}[0:L-1]$. From Equation (4.12) in the statement of the theorem, the observability matrix is assumed to be of full column rank, and all its columns are linearly independent of the columns of the invertibility matrix $\mathcal{M}_{i,L}^{\mathcal{J}}$ (since the set $\mathcal{J}$ has $2f$ or fewer nodes). This means that $\bar{\mathbf{x}} = \mathbf{x}[0]$ in the above expression, and thus node x_i has recovered the entire initial value vector $\mathbf{x}[0]$, despite the efforts of the nodes in $\mathcal{F}$. This concludes the proof of the theorem. ■

Remark 4.6 *Note that in order to apply the procedure described in the above theorem, node x_i needs to know its observability matrix $\mathcal{O}_{i,L}$ for some sufficiently large L (upper bounded by the size of the network, as we will show later in this section). It is sufficient (but not necessary) for node x_i to know the weight matrix $\mathbf{W}$ in order for it to obtain the observability matrix. This assumption of some (or full) knowledge of the network topology via the observability matrix (or the weight matrix) is similar to the assumptions made by much of the literature on fault-tolerant distributed systems [10, 11, 72, 73], and we will adopt it here to demonstrate the resilience of linear iterative strategies. As described in Chapter 2, it is possible for the nodes in the network to distributively determine their observability matrices when there are no faulty or malicious nodes; the extension of this result to faulty/malicious networks is an area for future research. Note also that node x_i does not necessarily need to know precisely how many nodes were malicious during the linear iteration; it only needs to know that the number of malicious nodes is upper bounded by f. If node x_i does not know the value of f a priori, but does know the network topology (e.g., via the weight matrix $\mathbf{W}$), it can determine the maximum number of malicious nodes that it can tolerate by calculating the size of the minimum vertex-cut between itself and any other node x_j, and then make the assumption that the actual number of malicious nodes does not exceed this maximum value.*

Remark 4.7 *It is worth noting that checking up to $\binom{N}{f}$ possibilities when trying to determine the possible sets of malicious nodes is equivalent to the brute force method of determining up to f errors in an N-dimensional real number codeword with distance $2f+1$. In the coding theory literature, there exist efficient ways of performing this check for both structured and random real number codes (e.g., see [81] and the references therein), and one can potentially exploit those results to streamline the procedure in the proof of Theorem 4.5 (this is left as an avenue for future research).*

Remark 4.8 *Note that if transmissions between nodes are corrupted by noise (as in Chapter 3), then the vector* $\mathbf{y}_i[0:L]$ *might not fall strictly within the column space of the matrices* $\mathcal{O}_{i,L}$ *and* $\mathcal{M}_{i,L}^{\mathcal{F}_j}$ *for any* j*. One potential method to handle this situation (if the magnitude of the noise is sufficiently "small") is to find the set* $\mathcal{F}_j$ *to minimize the smallest singular value of the matrix* $\left[\mathbf{y}_i[0:L] \quad \mathcal{O}_{i,L} \quad \mathcal{M}_{i,L}^{\mathcal{F}_j}\right]$*. Clearly the line between noise and malicious errors becomes blurred as the magnitude of the noise increases, which makes it harder to detect and isolate malicious behavior. This issue has been investigated in the context of real number error correcting codes in [81], and we will leave connections to such work for future research.*

In the sequel, we will show that when $\kappa_{ij} \geq 2f+1$ for all j, one can find a weight matrix $\mathbf{W}$ and an integer L so that all columns of the observability matrix $\mathcal{O}_{i,L}$ will be linearly independent of each other, and of the columns in $\mathcal{M}_{i,L}^{\mathcal{J}}$, where $\mathcal{J}$ is any set of up to $2f$ nodes (i.e., Equation (4.12) will be satisfied). From Theorem 4.5, node x_i will therefore be able to obtain all initial values from the outputs that it sees during the course of the linear iteration, and can calculate any arbitrary function of those values. To find such a weight matrix, recall from Section 1.5 that condition (4.12) is equivalent to *strong observability* of the linear system $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ (strong observability means that knowledge of the outputs of the system over a certain length of time suffices to uniquely determine the initial state, regardless of the inputs to the system). Furthermore, Theorem 1.3 indicated that strong observability of a linear system over the field of complex numbers is equivalent to the system having no invariant zeros. This indicates that if we can choose the weight matrix $\mathbf{W}$ so that the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ has no invariant zeros for all possible sets $\mathcal{J}$ of $2f$ nodes, then the rank condition in equation (4.12) of Theorem 4.5 will be satisfied with $L = N-1$; therefore, node x_i will be able to calculate any desired function of the initial values, even in the presence of up to f malicious or faulty nodes. In fact, our development will reveal that one can choose $\mathbf{W}$ so that multiple nodes in the system can simultaneously calculate any desired functions of the initial values (e.g., they can reach consensus, or calculate different functions of the initial values, all with the same weight matrix $\mathbf{W}$).

4.5.1 Invariant Zeros

Let x_i be any given node in the network, let $\mathcal{J} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{2f}}\}$ denote any set of $2f$ nodes (possibly containing x_i), and let $\bar{\mathcal{J}} = \mathcal{X} \setminus \mathcal{J}$. Further partition the sets $\mathcal{J}$ and $\bar{\mathcal{J}}$ as follows (an illustration of these sets is shown in Figure 4.2):

- $\mathcal{J}_{\mathcal{N}} = \mathcal{J} \cap (\mathcal{N}_i \cup \{x_i\})$. This set contains all nodes in set $\mathcal{J}$ that are neighbors of node x_i (or node x_i itself).
- $\mathcal{J}_{\bar{\mathcal{N}}} = \mathcal{J} \setminus \mathcal{J}_{\mathcal{N}}$. This set contains all nodes in set $\mathcal{J}$ that are not neighbors of node x_i (nor node x_i itself).

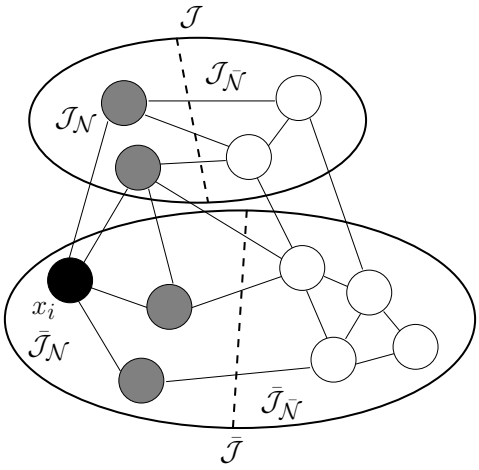

Figure 4.2: A sample partition of the vertex set. The black node is x_i, the gray nodes are $\mathcal{N}_i$ (the neighbors of node x_i), and the white nodes are all the other nodes.

- $\bar{\mathcal{J}}_{\mathcal{N}} = \bar{\mathcal{J}} \cap (\mathcal{N}_i \cup \{x_i\})$. This set contains all nodes in set $\bar{\mathcal{J}}$ that are neighbors of node x_i (or node x_i itself).
- $\bar{\mathcal{J}}_{\bar{\mathcal{N}}} = \bar{\mathcal{J}} \setminus \bar{\mathcal{J}}_{\mathcal{N}}$. This set contains all nodes in set $\bar{\mathcal{J}}$ that are not neighbors of node x_i (nor node x_i itself).

Note that these sets partition the entire set of nodes. For any choice of weights for the linear iteration, and for any subsets $\mathcal{A} \subseteq \mathcal{X}$ and $\mathcal{B} \subseteq \mathcal{X}$, let $\mathbf{W}_{\mathcal{A}}$ denote the square weight matrix corresponding to interconnections within the set $\mathcal{A}$, and let $\mathbf{W}_{\mathcal{A},\mathcal{B}}$ denote the weight matrix corresponding to connections from nodes in set $\mathcal{A}$ to nodes in set $\mathcal{B}$.

Lemma 4.3 *For any set $\mathcal{J}$ of $2f$ nodes, the invariant zeros of the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ are exactly the invariant zeros of the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$.*

Proof: The matrix pencil for the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ is given by $\mathbf{P}(z) = \begin{bmatrix} \mathbf{W} - z\mathbf{I}_N & \mathbf{B}_{\mathcal{J}} \\ \mathbf{C}_i & \mathbf{0} \end{bmatrix}$. Without loss of generality, we can assume that $\mathbf{W}$ is of the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\bar{\mathcal{J}}_{\mathcal{N}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\mathcal{J}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}}} \end{bmatrix},$$

and that $\mathbf{B}_{\mathcal{J}}$ and $\mathbf{C}_i$ have the form

$$\mathbf{B}_{\mathcal{J}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{|\mathcal{J}_{\bar{\mathcal{N}}}|} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} \end{bmatrix}, \quad \mathbf{C}_i = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{|\bar{\mathcal{J}}_{\mathcal{N}}|} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} \end{bmatrix},$$

since $\mathbf{B}_{\mathcal{J}}$ contains a single 1 in each row corresponding to a node in $\mathcal{J}$, and $\mathbf{C}_i$ measures nodes that are neighbors of node x_i, or node x_i itself; the above forms are obtained simply by an appropriate renumbering of the nodes.[5] The pencil $\mathbf{P}(z)$ can be written as

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} - z\mathbf{I}_{|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}}} - z\mathbf{I}_{|\bar{\mathcal{J}}_{\mathcal{N}}|} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}}} - z\mathbf{I}_{|\mathcal{J}_{\bar{\mathcal{N}}}|} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{I}_{|\mathcal{J}_{\bar{\mathcal{N}}}|} & \mathbf{0} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}}} - z\mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} & \mathbf{0} & \mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} \\ \mathbf{0} & \mathbf{I}_{|\bar{\mathcal{J}}_{\mathcal{N}}|} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

From this expression, we see that

$$\operatorname{rank}(\mathbf{P}(z)) = |\mathcal{J}_{\bar{\mathcal{N}}}| + |\mathcal{J}_{\mathcal{N}}| + |\bar{\mathcal{J}}_{\mathcal{N}}| + |\mathcal{J}_{\mathcal{N}}| + \operatorname{rank}\left(\begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} - z\mathbf{I}_{|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} \end{bmatrix}\right),$$

and so the invariant zeros of the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ are exactly the invariant zeros of the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$. ■

This lemma, along with Theorem 1.3, reveals that in order to ensure that the rank condition (4.12) in Theorem 4.5 is satisfied for node x_i, we can focus on the problem of choosing the weights so that the set

$$(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$$

will have no invariant zeros for any set $\mathcal{J}$ of $2f$ nodes. To choose a set of weights that accomplishes this, we will once again use techniques from control theory pertaining to *linear structured systems*. Recall from Section 1.5.1 that a linear system with a given zero/nonzero structure will have the same number of invariant zeros for almost any real-valued choice of free parameters. The exact number of invariant zeros can be obtained by examining the graph of the system, as specified in Theorems 1.6 and 1.7. To apply these results to the problem of determining the number of invariant zeros of the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$, we note that all matrices in this set are essentially structured matrices (since all of the weights in the above matrices can be chosen arbitrarily and independently). In particular, the nodes in set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$ act as the state vertices for the structured system, the nodes in the set $\mathcal{J}_{\bar{\mathcal{N}}}$ act as the input vertices, and the nodes in the set $\bar{\mathcal{J}}_{\mathcal{N}}$ act as the output vertices. Recall that the above results on structured systems assumed that the number of outputs is larger than (or equal to) the number of inputs. The

[5] Note that this does not affect the invariant zeros of the system, since this renumbering of the nodes simply corresponds to pre– and post–multiplying the original matrix pencil by appropriate permutation matrices. Since permutation matrices are nonsingular, the rank of the new pencil matrix will be the same as the rank of the original matrix pencil for all values of z.

following lemma shows that this property does in fact hold for the sets $\bar{\mathcal{J}}_{\mathcal{N}}$ and $\mathcal{J}_{\bar{\mathcal{N}}}$ if the in-degree of node x_i is sufficiently high.

Lemma 4.4 *If* $\deg_i \geq 2f+1$*, then for any set* $\mathcal{J}$ *of* $2f$ *nodes,* $|\bar{\mathcal{J}}_{\mathcal{N}}| > |\mathcal{J}_{\bar{\mathcal{N}}}|$.

Proof: First, note from the definition of $\bar{\mathcal{J}}_{\mathcal{N}}$ and $\mathcal{J}_{\mathcal{N}}$ that $|\bar{\mathcal{J}}_{\mathcal{N}}| + |\mathcal{J}_{\mathcal{N}}| = \deg_i + 1$ (since the sets $\mathcal{J}$ and $\bar{\mathcal{J}}$ partition the vertex set of the graph, and therefore contain all neighbors of node x_i, along with node x_i itself). Furthermore $|\mathcal{J}_{\mathcal{N}}| + |\mathcal{J}_{\bar{\mathcal{N}}}| = 2f$ (since those two sets partition the set $\mathcal{J}$). Thus, $|\bar{\mathcal{J}}_{\mathcal{N}}| = \deg_i + 1 - |\mathcal{J}_{\mathcal{N}}| = \deg_i + 1 - 2f + |\mathcal{J}_{\bar{\mathcal{N}}}|$. Since $\deg_i \geq 2f+1$, this expression becomes $|\bar{\mathcal{J}}_{\mathcal{N}}| \geq |\mathcal{J}_{\bar{\mathcal{N}}}| + 2 > |\mathcal{J}_{\bar{\mathcal{N}}}|$, thereby proving the lemma. ■

Note that if $\kappa_{ij} \geq 2f+1$ for every j, and there is at least one node that is not a neighbor of node x_i, then $\deg_i$ must necessarily be no smaller than $2f+1$ (since otherwise, there would not be $2f+1$ vertex disjoint paths from any node x_j to node x_i). We can now use the above results on structured systems to prove the following lemma for the linear iteration in (4.2); the lemma will be used in conjunction with Lemma 4.3 and Theorem 1.3 to show that node x_i can uniquely determine the initial values $\mathbf{x}[0]$ after running the linear iteration for at most N time-steps, even with up to f malicious nodes.

Lemma 4.5 *Consider node* x_i *in the network* $\mathcal{G}$*, and suppose that* $\kappa_{ij} \geq 2f+1$ *for all* j*. Then for almost any real-valued choice of weights (with* $w_{ij} = 0$ *if* $x_j \notin \mathcal{N}_i$*), the set* $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$ *will have no invariant zeros for any set* $\mathcal{J}$ *of* $2f$ *nodes.*

Proof: The matrix pencil for the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$ is

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} - z\mathbf{I}_{|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} \end{bmatrix}.$$

We will show that this pencil has full normal-rank even after the deletion of an arbitrary row, and then use Theorem 1.7 to prove the lemma.

We start by constructing the graph $\mathcal{H}$ associated with the set

$$(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}).$$

For this set of matrices, note that the state vertices in the graph $\mathcal{H}$ are given by the set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$, the input vertices are given by the set $\mathcal{J}_{\bar{\mathcal{N}}}$, and the output vertices are given by the set $\bar{\mathcal{J}}_{\mathcal{N}}$. In particular, $\mathcal{H}$ can be obtained by first taking the graph of the network $\mathcal{G}$ and removing all incoming edges to nodes in $\mathcal{J}_{\bar{\mathcal{N}}}$ (since the nodes in $\mathcal{J}_{\bar{\mathcal{N}}}$ are treated as inputs), and removing all outgoing edges from nodes in $\bar{\mathcal{J}}_{\mathcal{N}}$ (since those nodes are treated as outputs). Furthermore, remove all nodes that are in the set $\mathcal{J}_{\mathcal{N}}$ and their incident edges (since these

nodes do not appear anywhere in the matrix pencil), and add a set of self loops to every state vertex to correspond to the nonzero entries on the diagonal of the weight matrix $\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}$.

We will now examine what happens when we remove a single row from $\mathbf{P}(z)$. Suppose we remove one of the rows of $\mathbf{P}(z)$ corresponding to a vertex $v \in \bar{\mathcal{J}}_{\bar{\mathcal{N}}}$ (i.e., one of the top $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|$ rows of $\mathbf{P}(z)$), and denote the resulting matrix by $\bar{\mathbf{P}}(z)$. The generic rank of $\bar{\mathbf{P}}(z)$ can be found by examining the associated graph, which we will denote by $\bar{\mathcal{H}}$. Note that $\bar{\mathcal{H}}$ is obtained from $\mathcal{H}$ simply by removing all incoming edges to vertex v in $\mathcal{H}$, since we removed the row corresponding to v from $\mathbf{P}(z)$; however, all outgoing edges from v are still left in the graph (since the column corresponding to vertex v is left in matrix $\bar{\mathbf{P}}(z)$). Thus, we see that vertex v can be treated as an input vertex in $\bar{\mathcal{H}}$, leaving $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| - 1$ state vertices (corresponding to the set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}} \setminus \{v\}$). Now, note that the set $\mathcal{J} \cup \{v\}$ has cardinality $2f+1$, and thus Lemma 1.1 indicates that there is a linking of size $2f+1$ from $\mathcal{J} \cup \{v\}$ to $\{x_i\} \cup \mathcal{N}_i$ in $\mathcal{G}$. In this linking, consider the paths starting at vertices in the set $\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}$; none of these paths passes through the vertices in $\mathcal{J}_{\mathcal{N}}$ (since these vertices are the start vertices of other paths in the linking). Graph $\mathcal{G}$ thus contains a linking from $\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}$ to $\bar{\mathcal{J}}_{\mathcal{N}}$, where each path in the linking only passes through vertices in the set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$, and therefore this linking also exists in the graph $\bar{\mathcal{H}}$. According to Theorem 1.6, $\bar{\mathbf{P}}(z)$ has generic normal-rank equal to the number of state vertices in $\bar{\mathcal{H}}$ (equal to $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| - 1$) plus the maximal size of a linking from the inputs to the outputs (equal to $|\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}|$), for a total of $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| - 1 + |\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}| = |\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| + |\mathcal{J}_{\bar{\mathcal{N}}}|$. The pencil $\bar{\mathbf{P}}(z)$ thus has generically full normal-rank.

Now, suppose that we remove a row of $\mathbf{P}(z)$ corresponding to a vertex $v \in \bar{\mathcal{J}}_{\mathcal{N}}$ (i.e., one of the bottom $|\bar{\mathcal{J}}_{\mathcal{N}}|$ rows of $\mathbf{P}(z)$), and again denote the resulting matrix by $\bar{\mathbf{P}}(z)$. The associated graph $\bar{\mathcal{H}}$ is obtained by simply removing vertex v from $\mathcal{H}$. Since $\kappa_{ij} \geq 2f+1$ for every j in graph $\mathcal{G}$, Lemma 1.1 indicates that there will be a linking of size $2f+1$ from the set $\mathcal{J} \cup \{v\}$ to the set $\{x_i\} \cup \mathcal{N}_i$ in $\mathcal{G}$. In particular, there will be a linking of size $|\mathcal{J}_{\bar{\mathcal{N}}}|$ from the set $\mathcal{J}_{\bar{\mathcal{N}}}$ to the set $\bar{\mathcal{J}}_{\mathcal{N}} \setminus \{v\}$, and none of the paths in this linking would go through any of the vertices in the set $\mathcal{J}_{\mathcal{N}} \cup \{v\}$ (since these vertices are the start vertices of other paths in the linking). The linking from the set $\mathcal{J}_{\bar{\mathcal{N}}}$ to the set $\bar{\mathcal{J}}_{\mathcal{N}} \setminus \{v\}$ will therefore also exist in graph $\bar{\mathcal{H}}$, and once again, Theorem 1.6 indicates that the matrix pencil will have generically full normal-rank equal to $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| + |\mathcal{J}_{\bar{\mathcal{N}}}|$.

We have thus shown that $\mathbf{P}(z)$ will have generically full normal-rank even after the deletion of an arbitrary row. From Theorem 1.7, the number of invariant zeros of $\mathbf{P}(z)$ is equal to $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|$ minus the maximal number of vertices in $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$ contained in the disjoint union of a size $|\mathcal{J}_{\bar{\mathcal{N}}}|$ linking from $\mathcal{J}_{\bar{\mathcal{N}}}$ to $\bar{\mathcal{J}}_{\mathcal{N}}$, a cycle family in $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$, and a $\bar{\mathcal{J}}_{\mathcal{N}}$-topped path family. If we simply take all the self-loops in $\mathcal{H}$ (corresponding to the nonzero weights on the diagonal of the weight matrix $\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}$), we will have a set of disjoint cycles that covers all $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|$ vertices in $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$. Thus, the matrix pencil $\mathbf{P}(z)$ will generically have no invariant zeros, thereby proving the lemma. ■

With the above lemma in hand, it is a simple matter to prove Theorem 4.2 (provided at the end of Section 4.3).

Proof: [Theorem 4.2] If node x_i is in set $\mathcal{T}$, then from Lemmas 4.3 and 4.5, we see that for almost any choice of weight matrix $\mathbf{W}$, and for any set $\mathcal{J}$ of cardinality $2f$, the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ will have no invariant zeros. Furthermore, since the set of weights for which this property does not hold has measure zero, it will hold generically for all nodes in set $\mathcal{T}$. From Theorem 1.3, we see that $\text{rank}\left(\begin{bmatrix}\mathcal{O}_{i,N-1} & \mathcal{M}^{\mathcal{J}}_{i,N-1}\end{bmatrix}\right) = N + \text{rank}\left(\mathcal{M}^{\mathcal{J}}_{i,N-1}\right)$, for all $x_i \in \mathcal{T}$, and all sets $\mathcal{J}$ of $2f$ nodes. Thus, Theorem 4.5 indicates that every node in $\mathcal{T}$ can calculate any arbitrary function of the initial values after running the linear iteration for at most N time-steps. ■

4.5.2 Identifying the Malicious Nodes

Note that although the procedure described in the proof of Theorem 4.5 allows a given node x_i to determine the entire set of initial values, it does not necessarily require node x_i to determine the *actual* set of malicious nodes. Instead, node x_i is only required to find a *candidate* set $\mathcal{F}_j$ of malicious nodes, and then write the values that it has received over the past $L+1$ time-steps as a linear combination of the columns in the invertibility matrix for the candidate set and the observability matrix. In many cases, this candidate set will be the actual set of malicious nodes, but there can also be cases where the two sets are different. As a trivial example, consider the case of a network where two nodes x_i and x_j are separated by distance D, and suppose that node x_i requires $L+1$ time-steps in order to calculate the initial values of all nodes via the linear iteration. Now suppose that node x_j is malicious, but only commits its first additive error during one of the last $D-1$ time-steps. This additive error will not propagate to node x_i before time-step L, and thus node x_i will not be aware that node x_j has acted maliciously. In other words, as far as node x_i is concerned, the case where node x_j is malicious during the last $D-1$ time-steps of the linear iteration is indistinguishable from the case where node x_j behaves completely correctly for all time-steps. Note that this fact does not hamper node x_i from correctly obtaining the initial values of all the nodes. However, there may be applications when it is desirable for nodes to distributively identify the exact set of malicious nodes (e.g., in order to remove them from the network). The following theorem indicates that if node x_i examines the values that it receives from the linear iteration over a larger number of time-steps than that required purely for function calculation, it can determine the exact set of nodes that were malicious during the initial stages of the iteration.

Theorem 4.6 *Suppose node x_i can calculate the entire set of initial values after running the linear iteration for $L_i + 1$ time-steps, despite the actions of up to f malicious nodes (i.e., there is a weight matrix $\mathbf{W}$ that satisfies Equation (4.12) with $L = L_i$). Let $\mathcal{F}$ be the set of nodes that are malicious during the course of the linear iteration (with cardinality f*

or lower). Then, for any nonnegative integer $\bar{L}$, after $L_i + 1 + \bar{L}$ time-steps of the linear iteration, node x_i can uniquely identify the nodes in set $\mathcal{F}$ that were malicious during the first $\bar{L}$ time-steps of the iteration.

Proof: Note that the theorem holds trivially for $\bar{L} = 0$, and so we focus on the case of $\bar{L} \geq 1$ in the rest of the proof. The values seen by node x_i over $L_i + 1 + \bar{L}$ time-steps of the linear iteration are

$$\mathbf{y}_i[0 : L_i + \bar{L}] = \mathcal{O}_{i,L_i+\bar{L}}\mathbf{x}[0] + \mathcal{M}^{\mathcal{F}}_{i,L_i+\bar{L}}\mathbf{u}_{\mathcal{F}}[0 : L_i + \bar{L} - 1] \ .$$

As in the proof of Theorem 4.5, node x_i finds the first set $\mathcal{F}_j$ of f nodes such that

$$\mathbf{y}_i[0 : L_i + \bar{L}] = \mathcal{O}_{i,L_i+\bar{L}}\bar{\mathbf{x}} + \mathcal{M}^{\mathcal{F}_j}_{i,L_i+\bar{L}}\mathbf{u}_{\mathcal{F}_j}[0 : L_i + \bar{L} - 1] \ , \tag{4.14}$$

for some vectors $\bar{\mathbf{x}}$ and $\mathbf{u}_{\mathcal{F}_j}[0 : L_i + \bar{L} - 1]$. Equating the two expressions, we obtain

$$\mathcal{O}_{i,L_i+\bar{L}}(\bar{\mathbf{x}} - \mathbf{x}[0]) + \mathcal{M}^{\mathcal{F}_j}_{i,L_i+\bar{L}}\mathbf{u}_{\mathcal{F}_j}[0 : L_i + \bar{L} - 1] - \mathcal{M}^{\mathcal{F}}_{i,L_i+\bar{L}}\mathbf{u}_{\mathcal{F}}[0 : L_i + \bar{L} - 1] = \mathbf{0} \ .$$

Using the definition of $\mathcal{M}^{\mathcal{F}_j}_{i,L_i+\bar{L}}$ and $\mathcal{M}^{\mathcal{F}}_{i,L_i+\bar{L}}$ in Equation (4.3), the above expression can be written as

$$\begin{aligned}
\mathbf{0} = \mathcal{O}_{i,L_i+\bar{L}}(\bar{\mathbf{x}} - \mathbf{x}[0]) + &\sum_{k=0}^{L_i+\bar{L}-1} \begin{bmatrix} \mathbf{0}_{(k+1)(\deg_i+1)\times N} \\ \mathcal{O}_{i,L_i+\bar{L}-1-k} \end{bmatrix} \mathbf{B}_{\mathcal{F}_j}\mathbf{u}_{\mathcal{F}_j}[k] \\
&- \sum_{k=0}^{L_i+\bar{L}-1} \begin{bmatrix} \mathbf{0}_{(k+1)(\deg_i+1)\times N} \\ \mathcal{O}_{i,L_i+\bar{L}-1-k} \end{bmatrix} \mathbf{B}_{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[k] \\
= \mathcal{O}_{i,L_i+\bar{L}}(\bar{\mathbf{x}} - \mathbf{x}[0]) + &\sum_{k=0}^{L_i+\bar{L}-1} \begin{bmatrix} \mathbf{0}_{(k+1)(\deg_i+1)\times N} \\ \mathcal{O}_{i,L_i+\bar{L}-1-k} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{\mathcal{F}_j} & \mathbf{B}_{\mathcal{F}} \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\mathcal{F}_j}[k] \\ -\mathbf{u}_{\mathcal{F}}[k] \end{bmatrix} \ ,
\end{aligned}$$

where the subscripts on the zero matrices indicate their sizes. Let $\mathcal{J}$ be the set obtained by concatenating the sets $\mathcal{F}_j$ and $\mathcal{F}$ (note that $\mathcal{J}$ has at most $2f$ elements, possibly with duplications). Defining $\mathbf{B}_{\mathcal{J}} \equiv \begin{bmatrix} \mathbf{B}_{\mathcal{F}_j} & \mathbf{B}_{\mathcal{F}} \end{bmatrix}$ and $\mathbf{u}_{\mathcal{J}}[k] \equiv \begin{bmatrix} \mathbf{u}'_{\mathcal{F}_j}[k] & -\mathbf{u}'_{\mathcal{F}}[k] \end{bmatrix}'$, we obtain[6]

$$\mathcal{O}_{i,L_i+\bar{L}}(\bar{\mathbf{x}} - \mathbf{x}[0]) + \mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}}\mathbf{u}_{\mathcal{J}}[0 : L_i + \bar{L} - 1] = \mathbf{0} \ .$$

Now, note that since Equation (4.12) is satisfied for $L = L_i$, it will also be satisfied for any $L \geq L_i$. In particular, this means that $\bar{\mathbf{x}} = \mathbf{x}[0]$ in the above equation, and using the

[6]Note that the invertibility matrix $\mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}}$ has the same form as in Equation (4.3), even though the set $\mathcal{J}$ here has (possibly) duplicated entries. In particular, if Equation (4.12) is satisfied for $L = L_i$ when $\mathcal{J}$ has no duplicated entries, it will also be satisfied for $L = L_i$ when $\mathcal{J}$ has duplicated entries (since duplicated entries in $\mathcal{J}$ simply mean that certain columns of the matrix $\mathcal{M}^{\mathcal{J}}_{i,L_i}$ are duplicated).

recursive definition of the invertibility matrix in Equation (4.4), we obtain

$$\mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}}\mathbf{u}_{\mathcal{J}}[0:L_i+\bar{L}-1] = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathcal{O}_{i,L_i+\bar{L}-1}\mathbf{B}_{\mathcal{J}} & \mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}-1} \end{bmatrix} \mathbf{u}_{\mathcal{J}}[0:L_i+\bar{L}-1] = \mathbf{0} \ . \quad (4.15)$$

The matrix $\mathbf{B}_{\mathcal{J}}$ simply selects the columns of the matrix $\mathcal{O}_{i,L_i+\bar{L}-1}$ corresponding to the nodes in $\mathcal{J}$. Since Equation (4.12) is satisfied for any $L \geq L_i$, the first $|\mathcal{J}|$ columns of $\mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}}$ will be linearly independent of all the other columns of $\mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}}$. This means that

$$\begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L_i+\bar{L}-1} \end{bmatrix} \mathbf{B}_{\mathcal{J}}\mathbf{u}_{\mathcal{J}}[0] = \mathbf{0},$$

and since the matrix $\mathcal{O}_{i,L_i+\bar{L}-1}$ is full column rank for any $\bar{L} \geq 1$, we have $\mathbf{B}_{\mathcal{J}}\mathbf{u}_{\mathcal{J}}[0] = \mathbf{0}$. Repeating this reasoning recursively for the matrices $\mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}-1}, \mathcal{M}^{\mathcal{J}}_{i,L_i+\bar{L}-2}, \ldots, \mathcal{M}^{\mathcal{J}}_{i,L_i+1}$ (in Equation (4.15)), we obtain $\mathbf{B}_{\mathcal{J}}\mathbf{u}_{\mathcal{J}}[k] = \mathbf{0}$ for $k = 0, 1, \ldots, \bar{L}-1$. Now, using the definitions of $\mathbf{B}_{\mathcal{J}}$ and $\mathbf{u}_{\mathcal{J}}[k]$ from earlier in the proof, we can write

$$\mathbf{B}_{\mathcal{J}}\mathbf{u}_{\mathcal{J}}[k] \equiv \begin{bmatrix} \mathbf{B}_{\mathcal{F}_j} & \mathbf{B}_{\mathcal{F}} \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\mathcal{F}_j}[k] \\ -\mathbf{u}_{\mathcal{F}}[k] \end{bmatrix} = \mathbf{0}, \ \ k = 0, 1, \ldots, \bar{L}-1 \ .$$

Thus, if some entry of the vector $\mathbf{u}_{\mathcal{F}}[k]$ has some nonzero value a (corresponding to a malicious update by a node in $\mathcal{F}$ during time-step k), then the only way to eliminate the column of $\mathbf{B}_{\mathcal{F}}$ selected by that entry will be for that same column to appear in the matrix $\mathbf{B}_{\mathcal{F}_j}$ (which requires the malicious node to be in the set $\mathcal{F}_j$), and the corresponding entry of the vector $\mathbf{u}_{\mathcal{F}_j}[k]$ to also have value a. This means that in order for the above equation to be satisfied, all nodes in $\mathcal{F}_F$ that are malicious during the first $\bar{L}$ time-steps must be contained in $\mathcal{F}_j$, and the corresponding entries in $\mathbf{u}_{\mathcal{F}_j}[0:\bar{L}-1]$ must be equal to the errors caused by those nodes. Furthermore, all entries in $\mathbf{u}_{\mathcal{F}_j}[0:\bar{L}-1]$ corresponding to nodes in $\mathcal{F}_j \setminus \mathcal{F}$ must be zero, since otherwise, the corresponding columns of the matrix $\mathbf{B}_{\mathcal{F}_j}$ could not be canceled out. Thus, the candidate set $\mathcal{F}_j$ that node x_i finds in order to satisfy Equation (4.14) contains all nodes that are malicious during the first $\bar{L}$ time-steps, and the corresponding entries in $\mathbf{u}_{\mathcal{F}_j}[0:\bar{L}-1]$ contain the errors caused by those nodes. ■

4.5.3 Summary of the Protocol

We now summarize the steps needed to apply our fault-tolerant function calculation protocol to a given network $\mathcal{G}$. We split the protocol into a network design phase (a procedure used once during the initialization of the network) and the distributed function calculation phase (used repeatedly during the operation of the network).

Network Design

1. Let $\mathcal{T}$ represent the set of nodes in the network that are required to perform function calculation. Let f denote the maximum number of malicious or faulty nodes in the network, and suppose that the nodes in set $\mathcal{T}$ would like to identify all nodes that are malicious during the first $\bar{L}$ time-steps of the linear iteration, for some nonnegative integer $\bar{L}$.

2. If $\kappa_{ij} \geq 2f+1$ for all $j \in \{1, 2, \ldots, N\}$, node x_i is guaranteed to be able to calculate any arbitrary function, even when there are up to f malicious/faulty nodes (as shown in Theorem 4.2). Furthermore, in this latter case, node x_i can uniquely identify the nodes that were malicious during the first $\bar{L}$ time-steps of the protocol (assuming that the iteration will execute for a sufficiently large number of time-steps, as shown in Theorem 4.6). In the remainder of the procedure, we will assume that $\kappa_{ij} \geq 2f+1$ holds for each $x_i \in \mathcal{T}$ and for all $j \in \{1, 2, \ldots, N\}$.

3. Choose a set of independent random weights for each edge in the network, and also as self-weights for each node. For example, these weights could be chosen as independent and identically distributed (i.i.d.) Gaussian random variables, or i.i.d. random variables drawn from a uniform distribution on some interval. This produces a weight matrix $\mathbf{W}$.

4. For each $x_i \in \mathcal{T}$, verify that the following condition (given as Equation (4.12) in our development) is satisfied for some integer L_i and for all possible sets $\mathcal{J}$ of $2f$ nodes:

$$\text{rank}\left(\begin{bmatrix}\mathcal{O}_{i,L_i} & \mathcal{M}_{i,L_i}^{\mathcal{J}}\end{bmatrix}\right) = N + \text{rank}\left(\mathcal{M}_{i,L_i}^{\mathcal{J}}\right) \quad .$$

   Note that this will be satisfied with probability 1 for some $L_i \leq N-1$. Let $L_{\max} = \max_i L_i$. Then each node $x_i \in \mathcal{T}$ is guaranteed to be able to calculate its function after at most L_i+1 time-steps, and all nodes in $\mathcal{T}$ are guaranteed to be able to calculate their desired functions after at most $L_{\max}+1$ time-steps (note that $L_i \leq L_{\max} \leq N-1$). Furthermore, each node $x_i \in \mathcal{T}$ is guaranteed to be able to uniquely identify the nodes that were malicious during the first $\bar{L}$ time-steps of the linear iteration after at most $L_i + \bar{L} + 1$ time-steps. Note that if we require node x_i to identify the nodes that were malicious during any of the first L_i+1 time-steps (i.e., those nodes that were potentially trying to influence the result of node x_i's calculation), then node x_i should choose $\bar{L} = L_i + 1$. More generally, if all nodes run the linear iteration for $2(L_{\max}+1)$ time-steps, every node in $\mathcal{T}$ can calculate its function *and* identify any nodes that were trying to affect its calculation (since this corresponds to running the linear iteration for $L_{\max}+1+\bar{L}$ time-steps, with $\bar{L} = L_{\max}+1$).

5. Provide each node $x_i \in \mathcal{T}$ with the integers L_i and $L_{\max}$, the local weights w_{ij},

$x_j \in (\{x_i\} \cup \mathcal{N}_i)$, along with matrix[7] $\mathcal{O}_{i,L_i+\bar{L}}$, which node x_i can use to generate the matrices $\mathcal{M}_{i,L_i+\bar{L}}^{\mathcal{F}_j}$ for all possible sets $\mathcal{F}_j$ of f nodes.

Performing Function Calculation

1. Let $\mathbf{x}[0]$ denote the vector of initial values in the network.

2. At each time-step k, each node x_i in the network updates its value as

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] + u_i[k] \ ,$$

   where $u_i[k]$ is nonzero if node x_i is faulty/malicious and commits an error at time-step k. The nodes perform these updates for $L_{\max} + 1 + \bar{L}$ time-steps, and we suppose that there are f or fewer malicious or faulty nodes during these time-steps.

3. After $L_i + 1 + \bar{L}$ time-steps, node x_i has access to the values

$$\mathbf{y}_i[0 : L_i + \bar{L}] = \begin{bmatrix} \mathbf{y}_i'[0] & \cdots & \mathbf{y}_i'[L_i + \bar{L}] \end{bmatrix}' \ ,$$

   where $\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k]$ (the matrix $\mathbf{C}_i$ selects the portions of $\mathbf{x}[k]$ that correspond to neighbors of node x_i, along with node x_i itself).

4. Each node $x_i \in \mathcal{T}$ finds the first set $\mathcal{F}_j$ of f nodes such that $\mathbf{y}_i[0 : L_i + \bar{L}]$ is in the column space of $\mathcal{O}_{i,L_i+\bar{L}}$ and $\mathcal{M}_{i,L_i+\bar{L}}^{\mathcal{F}_j}$.

5. Node x_i concludes that $\mathcal{F}_j$ is a candidate set of malicious/faulty nodes. It then obtains all initial values in the network by finding a pair of vectors $\bar{\mathbf{x}}$ and $\mathbf{u}_{\mathcal{F}_j}[0 : L_i + \bar{L} - 1]$ such that $\mathbf{y}_i[0 : L_i + \bar{L}] = \mathcal{O}_{i,L_i+\bar{L}}\bar{\mathbf{x}} + \mathcal{M}_{i,L_i+\bar{L}}^{\mathcal{F}_j}\mathbf{u}_{\mathcal{F}_j}[0 : L_i + \bar{L} - 1]$; from Theorem 4.5, it will be the case that $\bar{\mathbf{x}} = \mathbf{x}[0]$. Node x_i can then calculate any function of the initial values. Furthermore, from Theorem 4.6, the set $\mathcal{F}_j$ will contain (as a subset) all nodes that were malicious during the first $\bar{L}$ time-steps of the linear iteration, and the entries of the vector $\mathbf{u}_{\mathcal{F}_j}[0 : \bar{L}-1]$ will contain the exact errors committed by those malicious nodes (all the entries of $\mathbf{u}_{\mathcal{F}_j}[0 : \bar{L} - 1]$ corresponding to correctly behaving nodes in $\mathcal{F}_j$ will be equal to zero).

6. After $L_{\max} + 1 + \bar{L}$ time-steps, all nodes in $\mathcal{T}$ will have calculated their desired functions and identified the malicious nodes in the above manner, and the protocol ends.

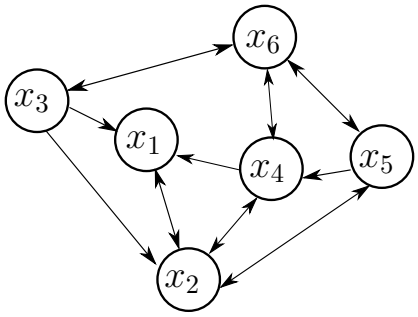

Figure 4.3: Node x_1 needs to calculate the sum of squares of all initial values in the network, even if one node in the network is malicious.

4.5.4 Example

Consider the network shown in Figure 4.3. The objective in this network is for node x_1 to calculate the function $g(x_1[0], x_2[0], x_3[0], x_4[0], x_5[0], x_6[0]) = \sum_{j=1}^{6} x_j^2[0]$, even if there is up to $f = 1$ malicious node in the network.

Network Design

Examining the network, we see that there are three internally vertex disjoint paths from node x_5 to node x_1, and also from node x_6 to node x_1. Since all other nodes are neighbors of node x_1, we have that $\kappa_{1j} \geq 3$ for all j, and so Theorem 4.2 indicates that node x_1 can calculate the desired function after running the linear iteration (with almost any choice of weights) for at most $N = 6$ time-steps, despite the presence of up to 1 malicious node. For this example, we will take each of the edge and self weights to be i.i.d. random variables chosen from the set[8] $\{-5, -4, -3, -2, -1, 1, 2, 3, 4, 5\}$ with equal probabilities. These weights produce the weight matrix

$$\mathbf{W} = \begin{bmatrix} -3 & 1 & -1 & 3 & 0 & 0 \\ 1 & 2 & -1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 0 & 4 & 0 & 2 & 2 & 5 \\ 0 & 2 & 0 & 0 & -1 & 5 \\ 0 & 0 & 1 & 3 & -4 & -4 \end{bmatrix}.$$

[7] Alternatively, each node could be provided with the weight matrix $\mathbf{W}$, which is sufficient information to obtain the desired matrices.

[8] In general, the result in Theorem 4.2 will hold with high probability if one chooses the weights for the linear iteration from a continuous distribution over the real numbers (such as a Gaussian distribution). For this pedagogical example, however, it will suffice to consider a distribution on a small set of integers; this makes the presentation of the numerical values more concise.

Since node x_1 has access to its own value, as well as those of its neighbors (nodes x_2, x_3 and x_4) at each time-step, the matrix $\mathbf{C}_1$ is given by $\mathbf{C}_1 = \left[\begin{smallmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{smallmatrix}\right]$. With the above weight matrix, one can verify that Equation (4.12) is satisfied with $i = 1$ and $L = L_1 = 2$ for all sets $\mathcal{J}$ of $2f = 2$ nodes, where $\mathcal{O}_{i,L_i}$ and $\mathcal{M}_{i,L_i}^{\mathcal{J}}$ are defined in Equation (4.3). In particular, the matrix $\mathcal{O}_{1,2}$ is given by $\mathcal{O}_{1,2} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_1\mathbf{W} \\ \mathbf{C}_1\mathbf{W}^2 \end{bmatrix}$ (the exact numerical values are omitted in the interest of space), and the invertibility matrices for each candidate set $\mathcal{F}_j = \{x_j\}$ (for $1 \leq j \leq 6$) of a single faulty node are summarized by

$$\begin{bmatrix} \mathcal{M}_{1,2}^{\mathcal{F}_1} & \mathcal{M}_{1,2}^{\mathcal{F}_2} & \mathcal{M}_{1,2}^{\mathcal{F}_3} & \mathcal{M}_{1,2}^{\mathcal{F}_4} & \mathcal{M}_{1,2}^{\mathcal{F}_5} & \mathcal{M}_{1,2}^{\mathcal{F}_6} \end{bmatrix} =$$

$$\left[\begin{array}{cc|cc|cc|cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -3 & 1 & 1 & 0 & -1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 2 & 1 & 2 & 0 & 5 & 0 \end{array}\right]. \quad (4.16)$$

We make the above matrices available to node x_1, and at this point, node x_1 has all the information it needs to be able to calculate the function $\sum_{j=1}^{6} x_j^2[0]$ (or indeed, any arbitrary function of the initial values) after $L_1 + 1 = 3$ time-steps of the linear iteration, even in the presence of one malicious node.

Performing Function Calculation

Suppose that the initial values of the nodes are $\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 \end{bmatrix}'$. In order for node x_1 to calculate the function $\sum_{j=1}^{6} x_j^2[0]$, the nodes run the linear iteration with the weight matrix provided above. Suppose that node x_4 is malicious, and at time-steps 1 and 2, it updates its value as $x_4[1] = 4x_2[0] + 2x_4[0] + 2x_5[0] + 5x_6[0] - 8$ and $x_4[2] = 4x_2[0] + 2x_4[0] + 2x_5[0] + 5x_6[0] - 12$, i.e., during the first time-step, it commits an additive error of $u_4[0] = -8$, and during the second time-step, it commits an additive error of $u_4[1] = -12$. All other nodes follow the predefined (correct) strategy of updating their values according to the weighted average specified by the weight matrix $\mathbf{W}$. The values of all nodes over the first three time-steps of the linear iteration are given by $\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 \end{bmatrix}'$,

$\mathbf{x}[1] = \begin{bmatrix}-26 & 0 & 26 & 49 & 46 & -80\end{bmatrix}'$, and $\mathbf{x}[2] = \begin{bmatrix}199 & 43 & -134 & -222 & -446 & 309\end{bmatrix}'$. Since the values seen by node x_1 at time-step k are given by $\mathbf{y}_1[k] = \mathbf{C}_1\mathbf{x}[k]$, the set of all values seen by node x_1 over the three time-steps of the linear iteration are

$$\mathbf{y}_1[0:2] = \begin{bmatrix}\mathbf{C}_1\mathbf{x}[0]\\ \mathbf{C}_1\mathbf{x}[1]\\ \mathbf{C}_1\mathbf{x}[2]\end{bmatrix} = \begin{bmatrix} 3 & -1 & 4 & -4 & -26 & 0 & 26 & 49 & 199 & 43 & -134 & -222 \end{bmatrix}'.$$

Node x_1 can now use these values to calculate the vector of initial values, despite the efforts of the malicious node. As discussed in Theorem 4.5, node x_1 finds the first set $\mathcal{F}_j$ for which $\mathbf{y}_1[0:2]$ is in the column space of $\mathcal{O}_{1,2}$ and $\mathcal{M}_{1,2}^{\mathcal{F}_j}$; using the matrices in (4.16), node x_1 finds that this is the case for $j = 4$. It now finds vectors $\bar{\mathbf{x}}$ and $\mathbf{u}_{\mathcal{F}_4}[0:1]$ such that $\mathbf{y}_i[0:2] = \mathcal{O}_{1,2}\bar{\mathbf{x}} + \mathcal{M}_{1,2}^{\mathcal{F}_4}\mathbf{u}_{\mathcal{F}_4}[0:1]$ as $\bar{\mathbf{x}} = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 \end{bmatrix}'$ and $\mathbf{u}_{\mathcal{F}_4}[0:1] = \begin{bmatrix} -8 & -12 \end{bmatrix}'$. Node x_1 now has access to $\mathbf{x}[0] = \bar{\mathbf{x}}$, and can calculate the function $\sum_{j=1}^{6} x_j^2[0]$ to obtain the value 212.

Note that in this example, the candidate set $\mathcal{F}_j$ found by node x_1 does, in fact, contain the actual malicious node x_4. This will not be true in general; if we want node x_1 to be guaranteed to locate any node that is malicious during the first $\bar{L}$ time-steps of the iteration, we should have node x_1 repeat the above procedure after $L_i + \bar{L} + 1 = 3 + \bar{L}$ time-steps of the linear iteration, as described in Theorem 4.6. The numerical details are very similar to those in the above example, and are omitted here.

4.6 Summary

In this chapter, we considered the problem of using linear iterative strategies for distributed function calculation with a set of malicious agents that have the potential to update their values in an arbitrary and coordinated manner (with the only restriction being that they cannot send conflicting values to different neighbors). We showed that the ability of linear iterative strategies to tolerate such malicious behavior is completely determined by the topology of the network. Specifically, if there exists a pair of nodes x_i and x_j such that $\kappa_{ij} \leq 2f$, then we showed that it is possible for a particular set of f malicious nodes to coordinate to update their values in such a way that node x_i cannot correctly determine the value of node x_j, and thus cannot calculate any function that involves that value, regardless of the number of time-steps for which the linear iteration is run. Conversely, we showed that it is possible for a given node x_i to calculate any arbitrary function in the presence of up to f malfunctioning or malicious nodes as long as the size of the smallest ij-cut with any other node x_j is at least $2f + 1$. For all nodes that satisfy the connectivity requirements, we showed that they can calculate their desired functions after running the linear iteration with almost any real-valued choice of weights for at most N time-steps. Furthermore, under

these connectivity requirements, any node that is malicious or faulty during the first part of the linear iteration can be uniquely identified by the other nodes in the network if they run the linear iteration for more time-steps than that required to simply perform function calculation.

CHAPTER 5

A STATE-SPACE FRAMEWORK FOR LINEAR NETWORK CODING

5.1 Introduction

In the previous chapters, we considered scenarios where each node has a single initial value, and developed a linear iterative strategy to disseminate these values (or perhaps functions of these values) throughout the network. In this chapter, we consider a slightly different (but still common) scenario in distributed systems and networks, where a subset of nodes in the network receive a stream of information (i.e., a new value at each time-step), and the requirement is to transmit these streams to a set of sink nodes. The traditional approach to this type of information dissemination in networks has been to treat data packets as distinct entities, and to route them through the network without performing intermediate operations on the packets (e.g., see the discussion in [82]). In recent years, however, the concept of *network coding* has generated a great deal of interest as it discards the notion of treating packets as individual elements, and instead allows intermediate nodes in the network to mathematically combine several packets into a single packet for transmission [9, 82, 83, 84, 85, 86]. It has been shown that, for certain networks, network coding can be used to achieve the transmission capacity of the network even when traditional store-and-forward approaches fail [83]. Of particular interest are *linear network codes*, where the packet transmitted by each node is a simple linear combination of the packets that it receives; such codes have been shown to be sufficient for achieving the transmission capacity for the multisource multicast problem (where every sink has to receive all of the data from all of the source nodes) [9].

Recently, researchers have started to examine the susceptibility of linear network codes to malicious or faulty behavior by a subset of the nodes [34, 35, 36, 37, 38, 39, 40]. Due to the fact that packets are combined at each node, a maliciously injected packet has the potential to corrupt *all* of the information that is received by the sink nodes, thereby preventing them from correctly recovering any of the source data. Various methods have been developed in an attempt to counteract such behavior, including the use of hashes or secret keys (under the assumption of computationally bounded attackers) [34, 36], or the introduction of redundancy to the source data in the form of an error-correcting code whose distance properties ensure robustness against a fixed number of attackers [35, 37,

38, 39, 40]. In particular, under the wired communication model (where each node can send a different value to each of its neighbors), it was shown in [37, 87, 88] that if the minimum size of an edge-cut between any source and any sink has capacity κ, and there are f (computationally unbounded) malicious attackers in the network, then the maximum rate at which information can be transmitted from the source to the sink is $\kappa - 2f$.

In this chapter, we extend the linear iterative strategies from the previous chapters to develop a control- and linear system-theoretic framework for designing linear network codes that are tolerant to faulty/malicious behavior under the wireless broadcast model (where each node transmits the same information to all of its neighbors [89]). Unlike some of the existing work on network coding with Byzantine adversaries, we assume that the network has a fixed structure, and that the sink nodes know the weights for the linear network code. The latter assumption will play a key role when malicious nodes are present and enables us to obtain a recursive algorithm which can recover the transmitted streams of information with a known bound on the delay. Specifically, using results from dynamic system inversion, structured system theory, and fault diagnosis schemes, we show that linear network codes (in fixed, known networks) can make use of redundancy in the network topology in order to safely transmit streams of information despite the presence of malicious agents without the need to pre-process the source data with error-correcting codes, hashes or cryptographic schemes (as done in existing schemes). We also demonstrate that linear network codes also allow the sink nodes to *identify and locate* all malicious nodes in a distributed manner as long as the connectivity of the network is large enough. This is in contrast to previous work on locating malicious nodes [39], which requires all nodes to send their received packets to a central supervisor to isolate faulty behavior. Finally, the state-space model representation of linear network codes that we develop in this chapter allows our techniques to be applied to arbitrary networks (directed/undirected, cyclic/acyclic), under the wireless broadcast communication model. Unlike existing work on network codes for cyclic networks, our approach does not require manipulation of matrices of rational functions, and instead focuses entirely on numerical matrices, thereby potentially simplifying the analysis and design of linear network codes (including convolutional codes). Our approach immediately allows us to obtain an upper bound on the latency required by any sink node to decode the input streams, and provides a systematic decoding procedure that can be applied at each sink node. It is worth noting that the work in [85] also proposes a state-space model for convolutional network codes that is similar to the model in this chapter; however, that work does not fully make use of the capabilities of this model in order to design decoders for the sink nodes, and furthermore, it does not consider the problem of malicious nodes in the network. We will discuss the differences between our work and existing work in the network coding literature in more detail later in the chapter, once we have had an opportunity to describe our approach and results in more detail.

5.2 Problem Formulation

Consider a network modeled by the directed graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is the set of nodes in the system and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ represents the communication constraints in the network (i.e., directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_i can receive information directly from node x_j). Let $\mathcal{S} \subseteq \mathcal{X}$ be a set of *source* nodes, and let $\mathcal{T} \subseteq \mathcal{X}$ be a set of *sink* nodes (note that $\mathcal{S}$ and $\mathcal{T}$ are not necessarily disjoint). Also, let $\mathcal{F} \subset \mathcal{X}$ be a set of nodes that act maliciously and transmit arbitrary values to the rest of the network (possibly by conspiring with each other) instead of following the strategy that we specify. We will assume that the sets $\mathcal{S}$ and $\mathcal{F}$ are disjoint (since otherwise, maliciously injected values would be indistinguishable from the values that the source node receives at each time-step). Note that the set $\mathcal{F}$ is unknown *a priori* to the nodes in the network.

At each time-step k, each source node $x_i \in \mathcal{S}$ receives new information from an entity external to the network; for example, if the network represents a set of sensors, this new information could represent measurements of the environment made by sensor x_i. We model this information as a stream of elements from a field $\mathbb{F}$, and the goal is for every source node to transmit its stream to each sink node via the network; this is the so-called multisource multicast problem (e.g., see [84]). In particular, at each time-step k, all nodes in the network can transmit a value based on some strategy that adheres to the constraints imposed by the network topology; let $x_i[k]$ denote the value transmitted by the i-th node at time-step k. The scheme that we study in this chapter makes use of linear network coding;[1] specifically, the value transmitted by each node x_i at time-step $k+1$ is given by

$$x_i[k+1] = \begin{cases} w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] + s_i[k] & \text{if } x_i \in \mathcal{S}, \\ w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] + f_i[k] & \text{if } x_i \in \mathcal{F}, \\ w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] & \text{if } x_i \notin \mathcal{S} \cup \mathcal{F}, \end{cases} \tag{5.1}$$

where the w_{ij}'s are a set of weights from the field $\mathbb{F}$,[2] $s_i[k]$ is the new value (information) obtained by x_i at time-step k (if it is a source node), and $f_i[k]$ is an additive error[3] committed by node x_i at time-step k (if it is a malicious node).

Definition 5.1 *Node x_i is said to be malicious (or faulty) during an interval $k, k+1, k+$*

[1] Note that linear network codes are essentially linear iterative strategies as defined in the earlier parts of the thesis, with the exception being that we are interested in recovering a *stream* of data from a set of designated sources, as opposed to a single initial value from every other node. In order to keep this distinction clear, and to maintain consistency with the literature on network coding, we will refer to linear iterative strategies as linear network codes in this chapter.

[2] The methodology for choosing the weights appropriately and the implications of this choice are discussed later in the chapter.

[3] Note that this model allows a malicious node to update and transmit an arbitrary value by choosing the error term $f_i[k]$ appropriately. It also allows multiple malicious nodes to update their values in a coordinated and conspiratorial manner. Note that if a node $x_i \in \mathcal{F}$ behaves normally at time-step k, we can simply set $f_i[k] = 0$.

$2, \ldots, k+L$ *if* $f_i[l]$ *is nonzero for some* $l \in \{k, k+1, k+2, \ldots, k+L\}$.

If we let $\mathcal{S} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{|\mathcal{S}|}}\}$ denote the set of source nodes, $\mathcal{F} = \{x_{j_1}, x_{j_2}, \ldots, x_{j_{|\mathcal{F}|}}\}$ denote the set of malicious nodes, and aggregate the values transmitted by all nodes at time-step k into the value vector $\mathbf{x}[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_N[k] \end{bmatrix}'$, the transmission strategy for the entire system can be represented as

$$\begin{aligned}
\mathbf{x}[k+1] = & \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{\mathbf{W}} \mathbf{x}[k] + \underbrace{\begin{bmatrix} \mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_{|\mathcal{S}|},N} \end{bmatrix}}_{\mathbf{B}_{\mathcal{S}}} \underbrace{\begin{bmatrix} s_{i_1}[k] \\ s_{i_2}[k] \\ \vdots \\ s_{i_{|\mathcal{S}|}}[k] \end{bmatrix}}_{\mathbf{s}[k]} \\
& + \underbrace{\begin{bmatrix} \mathbf{e}_{j_1,N} & \mathbf{e}_{j_2,N} & \cdots & \mathbf{e}_{j_{|\mathcal{F}|},N} \end{bmatrix}}_{\mathbf{B}_{\mathcal{F}}} \underbrace{\begin{bmatrix} f_{j_1}[k] \\ f_{j_2}[k] \\ \vdots \\ f_{j_{|\mathcal{F}|}}[k] \end{bmatrix}}_{\mathbf{f}[k]} \\
= & \mathbf{W}\mathbf{x}[k] + \underbrace{\begin{bmatrix} \mathbf{B}_{\mathcal{S}} & \mathbf{B}_{\mathcal{F}} \end{bmatrix}}_{\mathbf{B}_{\mathcal{S}\cup\mathcal{F}}} \underbrace{\begin{bmatrix} \mathbf{s}[k] \\ \mathbf{f}[k] \end{bmatrix}}_{\mathbf{u}[k]}, \qquad (5.2)
\end{aligned}$$

for all nonnegative integers k. In the above equation, the weight $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i$ (i.e., if $(x_j, x_i) \notin \mathcal{E}$), and we take $\mathbf{x}[0]$ to be the vector of all zeros.[4] Recall that the symbol $\mathbf{e}_{i,N}$ denotes a vector of length N with a single "1" in the i–th position and zeros elsewhere. The weight matrix $\mathbf{W}$ uniquely specifies the linear network coding strategy that is used.

At each time-step k, sink node $x_i \in \mathcal{T}$ has access to its own transmitted value as well as the values transmitted by its neighbors. Letting $\mathbf{y}_i[k]$ denote the transmitted values seen by node x_i during time-step k, we obtain the equation

$$\mathbf{y}_i[k] = \mathbf{C}_i \mathbf{x}[k], \qquad (5.3)$$

where $\mathbf{C}_i$ is a $(\deg_i + 1) \times N$ matrix with a single "1" in each row capturing the positions of the vector $\mathbf{x}[k]$ that are available to node x_i (i.e., these positions correspond to node x_i and its neighbors). The values $\mathbf{y}_i[k]$, $k = 0, 1, \ldots$, will characterize the ability of node x_i to reconstruct the source streams despite the actions of the malicious nodes in $\mathcal{F}$. In the next section, we will discuss conditions on the network topology and ways to choose the weights for each node so that each sink node in $\mathcal{T}$ can recover each of the values $s_i[k]$, $k = 0, 1, \ldots$,

[4]Note that the initial conditions are assumed to be *known* in this setting, as opposed to the previous chapters where we were trying to recover the initial values.

for all $x_i \in \mathcal{S}$ (possibly with some delay), and can also identify and locate all malicious nodes. Specifically, we will demonstrate the following key result in the sequel.

Theorem 5.1 *Let $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$ denote the graph of a fixed and known network with N nodes, and let $\mathcal{S}$ denote the set of source nodes and $\mathcal{T}$ denote the set of sink nodes. Suppose that for any set $\mathcal{J}$ of $2f$ nodes (disjoint from $\mathcal{S}$), the network contains an $(|\mathcal{S}|+2f)$–linking from $\mathcal{S} \cup \mathcal{J}$ to $\{x_i\} \cup \mathcal{N}_i$, for every $x_i \in \mathcal{T}$. Then, if the weights for the linear network code are chosen independently and uniformly from the field $\mathbb{F}_q$ (of size $q \geq |\mathcal{T}|\binom{N-|\mathcal{S}|}{2f}(N-|\mathcal{S}|-2f)$), then with probability at least $1 - |\mathcal{T}|\binom{N-|\mathcal{S}|}{2f}\frac{N-|\mathcal{S}|-2f}{q}$, there exists a nonnegative integer $L \leq N - |\mathcal{S}| - 2f + 1$ such that each sink node can uniquely determine the source streams transmitted by every source node with a delay of at most $L+1$ time-steps, as long as there are no more than f malicious nodes (anywhere in the network) during any time interval of length L. Furthermore, every sink node can uniquely determine the identities of the malicious nodes in a purely distributed manner.*

Remark 5.1 *Note that if the connectivity of the network is at least $|\mathcal{S}| + 2f$, then the linking condition in the above theorem is guaranteed to be satisfied due to the Fan Lemma (Lemma 1.1). Also, if the field under consideration has infinite size (such as the field of real numbers), the above theorem indicates that almost any choice of weights will produce a network code that allows the sink nodes to decode the source streams and identify the malicious nodes (i.e., the set of weights for which the property does not hold has Lebesgue measure zero). Finally, note that the set of f nodes that are malicious during some time interval of length L need not be the same as the set of nodes that are malicious during another time interval of length L. The only requirement is that no more than f nodes commit additive errors during any L contiguous time-steps.*

5.3 Decoding by Sink Nodes

To develop a robust decoding strategy for linear network codes, we start by making the observation that the code representation given by Equations (5.2) and (5.3) together form a linear system in state-space form [41]. This representation will provide us with a convenient and compact method of analyzing the values seen by the sink nodes during the operation of the network. Specifically, for any set $\mathcal{Q} = \{x_{q_1}, x_{q_2}, \ldots, x_{q_{|\mathcal{Q}|}}\}$ of nodes, let $\mathbf{B}_{\mathcal{Q}} = \begin{bmatrix}\mathbf{e}_{q_1,N} & \mathbf{e}_{q_2,N} & \cdots & \mathbf{e}_{q_{|\mathcal{Q}|},N}\end{bmatrix}$. Then the output of the system

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] + \mathbf{B}_{\mathcal{Q}}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}_i\mathbf{x}[k]\end{aligned}$$

over $L+1$ time-steps is given by

$$\underbrace{\begin{bmatrix} \mathbf{y}_i[k] \\ \mathbf{y}_i[k+1] \\ \mathbf{y}_i[k+2] \\ \vdots \\ \mathbf{y}_i[k+L] \end{bmatrix}}_{\mathbf{y}_i[k:k+L]} = \underbrace{\begin{bmatrix} \mathbf{C}_i \\ \mathbf{C}_i\mathbf{W} \\ \mathbf{C}_i\mathbf{W}^2 \\ \vdots \\ \mathbf{C}_i\mathbf{W}^L \end{bmatrix}}_{\mathcal{O}_{i,L}} \mathbf{x}[k] + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{B}_{\mathcal{Q}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\mathcal{Q}} & \mathbf{C}_i\mathbf{B}_{\mathcal{Q}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\mathcal{Q}} & \mathbf{C}_i\mathbf{W}^{L-2}\mathbf{B}_{\mathcal{Q}} & \cdots & \mathbf{C}_i\mathbf{B}_{\mathcal{Q}} \end{bmatrix}}_{\mathcal{M}_{i,L}^{\mathcal{Q}}} \underbrace{\begin{bmatrix} \mathbf{u}[k] \\ \mathbf{u}[k+1] \\ \mathbf{u}[k+2] \\ \vdots \\ \mathbf{u}[k+L-1] \end{bmatrix}}_{\mathbf{u}[k:k+L-1]} . \tag{5.4}$$

Recall from the previous chapters that when $L = N-1$, the matrix $\mathcal{O}_{i,L}$ is called the *observability matrix*, and the matrix $\mathcal{M}_{i,L}^{\mathcal{Q}}$ is called the *invertibility matrix corresponding to the set* $\mathcal{Q}$.

The following theorem provides a decoding procedure for the sink nodes to recover the source streams and identify the malicious nodes for the linear network code defined by equations (5.2) and (5.3), provided that a certain algebraic condition holds. We will later relate this algebraic condition to conditions on the network topology and choices of weight matrix $\mathbf{W}$.

Theorem 5.2 *Let $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$ denote the graph of a fixed and known network with N nodes, and let $\mathcal{S}$ denote the set of source nodes and $\mathcal{T}$ denote the set of sink nodes. Suppose that there exists an integer L and a weight matrix $\mathbf{W}$ (with $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i$) such that, for all possible sets $\mathcal{J}$ of up to $2f$ nodes (disjoint from $\mathcal{S}$), the matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{S}\cup\mathcal{J}}$ (defined in Equation (5.4)) for sink node x_i satisfy*

$$rank\left(\mathcal{M}_{i,L}^{\mathcal{S}\cup\mathcal{J}}\right) = |\mathcal{S}| + |\mathcal{J}| + rank\left(\mathcal{M}_{i,L-1}^{\mathcal{S}\cup\mathcal{J}}\right) \quad . \tag{5.5}$$

Then, if the nodes perform linear network coding with the weight matrix $\mathbf{W}$, node x_i can uniquely decode all of the source streams $s_j[k]$, $\forall x_j \in \mathcal{S}$, as long as no more than f nodes in the network are malicious during any time interval of length L. Furthermore, all of these malicious nodes can be uniquely identified by sink node x_i.

Before proceeding with the proof of the above theorem, we provide a more detailed explanation of condition (5.5). Specifically, note from (5.4) that the last $(L-1)|\mathcal{Q}|$ columns of $\mathcal{M}_{i,L}^{\mathcal{Q}}$ have the form $\begin{bmatrix} \mathbf{0} \\ \mathcal{M}_{i,L-1}^{\mathcal{Q}} \end{bmatrix}$, and thus have rank equal to the rank of $\mathcal{M}_{i,L-1}^{\mathcal{Q}}$. Thus, condition (5.5) means that the first $|\mathcal{S}| + |\mathcal{J}|$ columns of $\mathcal{M}_{i,L}^{\mathcal{S}\cup\mathcal{J}}$ are linearly independent of each other, and of all other columns in $\mathcal{M}_{i,L}^{\mathcal{S}\cup\mathcal{J}}$. With this interpretation in hand, we are now ready to continue with the proof of Theorem 5.2.

Proof: [Theorem 5.2] Let $\mathbf{W}$ be a weight matrix that satisfies the conditions in the above theorem, and let the nodes run the linear iteration for $L+1$ time-steps. Suppose that the malicious nodes during the first L time-steps of the linear iteration are a subset of

the set $\widehat{\mathcal{F}} = \{x_{l_1}, x_{l_2}, \ldots, x_{l_f}\} \subseteq \mathcal{F}$. From (5.2), (5.3) and (5.4), the values seen by node x_i over the first $L+1$ time-steps are given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}}\mathbf{u}[0:L-1] \ , \tag{5.6}$$

where $\mathbf{u}[k] = \begin{bmatrix}\mathbf{s}'[k] & \mathbf{f}'[k]\end{bmatrix}'$ ($\mathbf{s}[k]$ is the vector of values injected by the source nodes, and $\mathbf{f}[k]$ is the vector of additive errors injected by the malicious nodes at time-step k). Note that node x_i knows the quantities $\mathbf{y}_i[0:L]$, $\mathcal{O}_{i,L}\mathbf{x}[0]$, and the set $\mathcal{S}$, but it does not know the set $\widehat{\mathcal{F}}$ or the values $\mathbf{u}[0:L-1]$. Node x_i will try to find the nodes in the set $\widehat{\mathcal{F}}$ (to identify the malicious nodes) and the values $\mathbf{s}[k]$ based on the known quantities.

Let $\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \ldots, \widehat{\mathcal{F}}_{\binom{N-|\mathcal{S}|}{f}}$ denote all possible sets of f nodes (disjoint from $\mathcal{S}$), and let $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}_1}, \mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}_2}, \ldots, \mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}_{\binom{N-|\mathcal{S}|}{f}}}$ denote the invertibility matrices corresponding to these possible sets of malicious nodes together with the source nodes. With these matrices in hand, suppose node x_i finds the first $j \in \{1, 2, \ldots, \binom{N-|\mathcal{S}|}{f}\}$ such that the vector

$$\mathbf{y}_i[0:L] - \mathcal{O}_{i,L}\mathbf{x}[0]$$

is in the column space of the matrix $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}$. This means that node x_i can find a vector $\bar{\mathbf{u}}[0:L-1]$ such that

$$\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}\bar{\mathbf{u}}[0:L-1] = \mathbf{y}_i[0:L] - \mathcal{O}_{i,L}\mathbf{x}[0].$$

The vector $\bar{\mathbf{u}}[0:L-1]$ is node x_i's *estimate* of the value of $\mathbf{u}[0:L-1]$ (note that $\bar{\mathbf{u}}[k] = \begin{bmatrix}\bar{\mathbf{s}}'[k] & \bar{\mathbf{f}}'[k]\end{bmatrix}'$ contains estimates of the source input and the fault input at time-step k). Substituting (5.6) into the above expression and rearranging, we have

$$\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}}\mathbf{u}[0:L-1] - \mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}\bar{\mathbf{u}}[0:L-1] = \mathbf{0} \ .$$

Based on the form of the invertibility matrix in (5.4), the above equation can be written as

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} & \mathbf{C}_i\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} & \mathbf{C}_i\mathbf{W}^{L-2}\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} & \cdots & \mathbf{C}_i\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} \end{bmatrix} \begin{bmatrix} \mathbf{u}[0] \\ \mathbf{u}[1] \\ \mathbf{u}[2] \\ \vdots \\ \mathbf{u}[L-1] \end{bmatrix} -$$

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} & \mathbf{C}_i\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} & \mathbf{C}_i\mathbf{W}^{L-2}\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} & \cdots & \mathbf{C}_i\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{u}}[0] \\ \bar{\mathbf{u}}[1] \\ \bar{\mathbf{u}}[2] \\ \vdots \\ \bar{\mathbf{u}}[L-1] \end{bmatrix} = \mathbf{0},$$

where $\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}} = \begin{bmatrix} \mathbf{B}_{\mathcal{S}} & \mathbf{B}_{\widehat{\mathcal{F}}} \end{bmatrix}$ and $\mathbf{B}_{\mathcal{S}\cup\widehat{\mathcal{F}}_j} = \begin{bmatrix} \mathbf{B}_{\mathcal{S}} & \mathbf{B}_{\widehat{\mathcal{F}}_j} \end{bmatrix}$. Let $\{\widehat{\mathcal{F}}, \widehat{\mathcal{F}}_j\}$ denote the set that is obtained by concatenating sets $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{F}}_j$ (i.e., it is the union of the two sets, with duplications allowed). Then, the above expression can be written[5] as

$$\underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} & \mathbf{C}_i\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} & \mathbf{C}_i\mathbf{W}^{L-2}\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} & \cdots & \mathbf{C}_i\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} \end{bmatrix}}_{\mathcal{M}_{i,L}^{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}}} \begin{bmatrix} \tilde{\mathbf{u}}[0] \\ \tilde{\mathbf{u}}[1] \\ \tilde{\mathbf{u}}[2] \\ \vdots \\ \tilde{\mathbf{u}}[L-1] \end{bmatrix} = \mathbf{0}, \quad (5.7)$$

where $\mathbf{B}_{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}} = \begin{bmatrix} \mathbf{B}_{\mathcal{S}} & \mathbf{B}_{\widehat{\mathcal{F}}} & \mathbf{B}_{\widehat{\mathcal{F}}_j} \end{bmatrix}$ and

$$\tilde{\mathbf{u}}[k] = \begin{bmatrix} \mathbf{s}[k] - \bar{\mathbf{s}}[k] \\ \mathbf{f}[k] \\ -\bar{\mathbf{f}}[k] \end{bmatrix}.$$

Now consider the matrix $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}\cup\widehat{\mathcal{F}}_j}$. Since $\widehat{\mathcal{F}} \cup \widehat{\mathcal{F}}_j$ has at most $2f$ nodes, Equation (5.5) in the statement of the theorem indicates that the first $|\mathcal{S}| + |\widehat{\mathcal{F}} \cup \widehat{\mathcal{F}}_j|$ columns of the matrix $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}\cup\widehat{\mathcal{F}}_j}$ are linearly independent of each other, and of all other columns of the matrix. Now, note that the matrix $\mathcal{M}_{i,L}^{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}}$ is obtained from matrix $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}\cup\widehat{\mathcal{F}}_j}$ simply by duplicating certain columns (namely, the columns corresponding to nodes that appear in both $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{F}}_j$). Consider a node $x_l \in \widehat{\mathcal{F}}$. If $x_l \notin \widehat{\mathcal{F}}_j$, then the column corresponding to x_l within the first $|\mathcal{S}| + 2f$ columns of $\mathcal{M}_{i,L}^{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}}$ will be linearly independent of all other columns in $\mathcal{M}_{i,L}^{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}}$ (since this column will also appear in the first $|\mathcal{S}| + |\widehat{\mathcal{F}} \cup \widehat{\mathcal{F}}_j|$ columns of $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}\cup\widehat{\mathcal{F}}_j}$). This means that Equation (5.7) will be satisfied only if $f_l[0] = 0$. On the other hand, if $f_l[0] \neq 0$, the only way for Equation (5.7) to be satisfied is if $x_l \in \widehat{\mathcal{F}}_j$ and $\bar{f}_l[0] = f_l[0]$. In other words, if Equation (5.5) is satisfied, any malicious node that commits an error during the first time-step will appear in set $\widehat{\mathcal{F}}_j$, and its additive error can be found

[5]Note that $\mathcal{S} \cup \widehat{\mathcal{F}} = \{\mathcal{S}, \widehat{\mathcal{F}}\}$ since $\widehat{\mathcal{F}}$ and $\mathcal{S}$ are disjoint (the same is true for $\mathcal{S}$ and $\widehat{\mathcal{F}}_j$). However, we allow for the possibility that the sets $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{F}}_j$ are not disjoint.

by sink node x_i.

Next, note that the columns corresponding to nodes in $\mathcal{S}$ in $\mathcal{M}_{i,L}^{\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}}$ will be linearly independent of each other and of all other columns in that matrix (since these columns appear in $\mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}\cup\widehat{\mathcal{F}}_j}$, and are not duplicated in $\{\mathcal{S},\widehat{\mathcal{F}},\widehat{\mathcal{F}}_j\}$). This means that the only way for Equation (5.7) to be satisfied is if $\bar{\mathbf{s}}[0] = \mathbf{s}[0]$. Thus, sink node x_i has also recovered the source inputs injected into the network at time-step $k = 0$.

At this point, sink node x_i knows $\mathbf{s}[0]$ and the identities of those nodes in $\widehat{\mathcal{F}}$ that committed errors during time-step 0, along with the exact values of their additive errors. Node x_i can then obtain the value of the transmitted values at time-step $k = 1$ as

$$\mathbf{x}[1] = \mathbf{W}\mathbf{x}[0] + \mathbf{B}_{\mathcal{S}}\mathbf{s}[0] + \mathbf{B}_{\widehat{\mathcal{F}}_j}\bar{\mathbf{f}}[0] \ .$$

Now, if we again let $\widehat{\mathcal{F}}$ denote the set of f (or fewer) nodes that were malicious during the interval $k = 1, 2 \ldots, L$, then using the identity

$$\mathbf{y}_i[1 : L+1] = \mathcal{O}_{i,L}\mathbf{x}[1] + \mathcal{M}_{i,L}^{\mathcal{S}\cup\widehat{\mathcal{F}}}\mathbf{u}[1 : L] \ ,$$

node x_i can repeat the above process to find the values of $\mathbf{s}[1]$ along with the identities of the nodes that are malicious during time-step $k = 1$. Note that the set $\widehat{\mathcal{F}}$ during this time interval need not be the same as the set $\widehat{\mathcal{F}}$ during the first L time-steps (but the nodes that are malicious between time-steps $k = 1$ and $k = L - 1$ will be common to both sets). By repeating the above procedure for all positive values of k, node x_i can obtain the source streams $\mathbf{s}[k]$ for all k, along with the identities of all malicious nodes and the errors that they commit. ■

Remark 5.2 *Note that node x_i does not actually have to store all of the matrices $\mathcal{M}_{i,L}^{\mathcal{S}\cup\mathcal{F}_j}$ for every possible value of j. Specifically, note from the form of matrix $\mathbf{B}_{\mathcal{Q}}$ and the invertibility matrix $\mathcal{M}_{i,L}^{\mathcal{Q}}$ in Equation* (5.4) *that the columns of this matrix can be obtained by simply selecting certain columns of the observability matrix $\mathcal{O}_{i,l}$, for $0 \leq l \leq L-1$. Thus, node x_i only needs to store the observability matrix $\mathcal{O}_{i,L}$ in order to generate all of the required invertibility matrices.*

Remark 5.3 *Note that the decoding procedure in the above proof hinges on the fact that the observability matrix (or more generally, the weight matrix) for the linear iteration is known to the sink nodes. We will comment on this in further detail later in the chapter.*

The above theorem provides a recursive decoding procedure for sink node x_i provided that condition (5.5) is true. We will now discuss ways to choose the weight matrix in order to satisfy this condition; in the process, we will obtain the proof of the main theorem (Theorem 5.1) of the chapter.

5.3.1 Network Topology Conditions for Robust Linear Network Codes

To see how to choose a weight matrix $\mathbf{W}$ to satisfy condition (5.5) in Theorem 5.2, recall from Section 1.5 that this condition simply means that the linear system $(\mathbf{W}, \mathbf{B}_{\mathcal{S}\cup\mathcal{J}}, \mathbf{C}_i)$ must be *invertible* for all possible sets $\mathcal{J}$ of $2f$ nodes (disjoint from $\mathcal{S}$). To achieve this, we will be using the results on structured invertibility over finite fields discussed in Section A.3. Specifically, note that for any given set $\mathcal{J}$, the tuple $(\mathbf{W}, \mathbf{B}_{\mathcal{S}\cup\mathcal{J}}, \mathbf{C}_i)$ essentially defines a structured linear system (since the entries in the weight matrix $\mathbf{W}$ are either zeros or independent free parameters from the field $\mathbb{F}$). With this observation, we are now in place to prove Theorem 5.1 (given at the end of Section 5.2).

Proof: [Theorem 5.1] Consider any sink node x_i and any set $\mathcal{J}$ of $2f$ nodes (disjoint from $\mathcal{S}$), and denote the graph of the system $(\mathbf{W}, \mathbf{B}_{\mathcal{S}\cup\mathcal{J}}, \mathbf{C}_i)$ by $\mathcal{H}_{i,\mathcal{J}}$. Specifically, $\mathcal{H}_{i,\mathcal{J}}$ is obtained by taking the graph of the network $\mathcal{G}$ and adding $|\mathcal{S}| + 2f$ input vertices (each with a single outgoing edge to a vertex in the set $\mathcal{S} \cup \mathcal{J}$), and $\deg_i + 1$ output vertices (each with a single incoming edge from $\{x_i\} \cup \mathcal{N}_i$). Furthermore, add a self-loop to every state vertex in $\mathcal{H}_{i,\mathcal{J}}$ to correspond to the nonzero entries on the diagonal of the weight matrix $\mathbf{W}$.

From the statement of the theorem, note that graph $\mathcal{G}$ contains a linking of size $|\mathcal{S}| + 2f$ from $\mathcal{S} \cup \mathcal{J}$ to $\{x_i\} \cup \mathcal{N}_i$, for any set $\mathcal{J}$ of $2f$ nodes (disjoint from $\mathcal{S}$) and every $x_i \in \mathcal{T}$. This linking also exists in the graph $\mathcal{H}_{i,\mathcal{J}}$, since $\mathcal{G}$ is a subgraph of $\mathcal{H}_{i,\mathcal{J}}$. Since every input vertex in $\mathcal{H}_{i,\mathcal{J}}$ connects to exactly one vertex in $\mathcal{S} \cup \mathcal{J}$, and since every output vertex in $\mathcal{H}_{i,\mathcal{J}}$ connects to exactly one vertex in $\{x_i\} \cup \mathcal{N}_i$, the linking of size $|\mathcal{S}| + 2f$ from $\mathcal{S} \cup \mathcal{J}$ to $\{x_i\} \cup \mathcal{N}_i$ can be extended to a linking of size $|\mathcal{S}| + 2f$ from the input vertices to the set of output vertices.

Suppose we choose all of the free parameters in $\mathbf{W}$ (i.e., the weights) independently and uniformly from the field $\mathbb{F}_q$, and let $\phi_{i,\mathcal{J}}$ denote the indicator function for the invertibility of the system $(\mathbf{W}, \mathbf{B}_{\mathcal{S}\cup\mathcal{J}}, \mathbf{C}_i)$ (i.e., $\phi_{i,\mathcal{J}} = 1$ if the system is invertible, and zero otherwise). The graph $\mathcal{H}_{i,\mathcal{J}}$ satisfies the conditions in Theorem A.5 (in Section A.3), and thus that theorem implies that $\Pr[\phi_{i,\mathcal{J}} = 0] \leq \frac{N-|\mathcal{S}|-2f}{q}$. The probability that there is at least one sink node or one set $\mathcal{J}$ for which the system $(\mathbf{W}, \mathbf{B}_{\mathcal{S}\cup\mathcal{J}}, \mathbf{C}_i)$ is not invertible is given by

$$\begin{aligned}\Pr\left[\bigcup_{\substack{x_i\in\mathcal{T},\\ \{\mathcal{J}|\mathcal{J}\cap\mathcal{S}=\emptyset,|\mathcal{J}|=2f\}}} (\phi_{i,\mathcal{J}} = 0)\right] &\leq \sum_{\substack{x_i\in\mathcal{T},\\ \{\mathcal{J}|\mathcal{J}\cap\mathcal{S}=\emptyset,|\mathcal{J}|=2f\}}} \Pr\left[\phi_{i,\mathcal{J}} = 0\right] \\ &\leq |\mathcal{T}|\binom{N-|\mathcal{S}|}{2f}\frac{N-|\mathcal{S}|-2f}{q} \ .\end{aligned}$$

Thus, the probability that all such systems are invertible is given by

$$\Pr\left[\bigcap_{\substack{x_i\in\mathcal{T},\\ \{\mathcal{J}|\mathcal{J}\cap\mathcal{S}=\emptyset,|\mathcal{J}|=2f\}}} (\phi_{i,\mathcal{J}}=1)\right] \geq 1-|\mathcal{T}|\binom{N-|\mathcal{S}|}{2f}\frac{N-|\mathcal{S}|-2f}{q}.$$

Now, recall from the discussion on invertibility of linear systems (Theorem 1.2 in Section 1.5) that if a linear system with matrices $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$ is invertible, then the number of time-steps required to do so will be less than $N-\text{nullity}(\mathbf{D})+1$. In our case, the $\mathbf{D}$ matrix is simply zero, and thus it has a nullspace of dimension equal to the number of inputs (which is $|\mathcal{S}|+2f$). Thus, with probability at least $1-|\mathcal{T}|\binom{N-|\mathcal{S}|}{2f}\frac{N-|\mathcal{S}|-2f}{q}$, the first $|\mathcal{S}|+2f$ columns of the matrix $\mathcal{M}^{\mathcal{S}\cup\mathcal{J}}_{i,N-|\mathcal{S}|-2f+1}$ will be linearly independent of each other and of all other columns in $\mathcal{M}^{\mathcal{S}\cup\mathcal{J}}_{i,N-|\mathcal{S}|-2f+1}$ for all $x_i\in\mathcal{T}$ and all sets $\mathcal{J}$ of $2f$ nodes (disjoint from $\mathcal{S}$). Thus, condition (5.5) in Theorem 5.2 is satisfied for every sink node $x_i\in\mathcal{T}$, and they can uniquely determine the exact values of the source streams, along with the identities of the malicious nodes. ■

Remark 5.4 *Note from Theorem 5.1 that the ability to decode the source streams depends solely on the existence of sufficiently large linkings in the network. This implies that the self-weights on the diagonal of the weight matrix are not necessary in order to form an invertible system (since they do not appear in the linking anyway). For this reason, the self-weights can be set to zero, if desired.*

5.4 Example

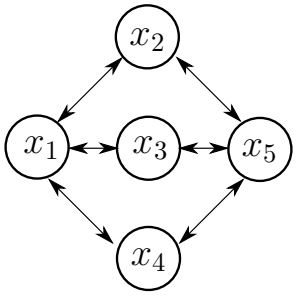

Figure 5.1: Network for multisource multicast. Nodes x_1 and x_5 wish to exchange their streams of information with each other. One of the intermediate nodes x_2, x_3 or x_4 might be malicious, but the identity of the malicious node is not known *a priori*.

Consider the undirected network in Figure 5.1. The set of source nodes in the network is given by $\mathcal{S}=\{x_1,x_5\}$, and those are also the sink nodes (i.e., $\mathcal{T}=\mathcal{S}$). In other words, the objective in the network is for nodes x_1 and x_5 to exchange streams of information with

each other via the intermediate nodes x_2, x_3 and x_4. One of the intermediate nodes might be malicious, but its identity is not known to the sink nodes *a priori*. For simplicity, we will assume that the same node is malicious throughout the linear iteration (this simply means that the set $\widehat{\mathcal{F}}$ of malicious nodes during any time interval is equal to the set $\mathcal{F}$ of malicious nodes over all time intervals).

In order for the nodes to be able to exchange streams of information and identify up to $f = 1$ malicious node, Theorem 5.1 indicates that there must exist a linking of size $|\mathcal{S}| + 2f = 4$ from the set $\mathcal{S} \cup \mathcal{J}$ to node x_1 and its neighbors, and also from $\mathcal{S} \cup \mathcal{J}$ to node x_5 and its neighbors. Here, $\mathcal{J}$ is any set of $2f = 2$ nodes (different from x_1 and x_5). One can verify that both of these conditions are satisfied in this network. For example, if $\mathcal{J} = \{x_2, x_3\}$, we obtain a linking of size 4 from $\mathcal{S} \cup \mathcal{J} = \{x_1, x_5, x_2, x_3\}$ to $\{x_1\} \cup \mathcal{N}_1 = \{x_1, x_2, x_3, x_4\}$ via the following set of paths: $\{x_1, x_2, x_3, x_5 x_4\}$; note that some of these paths have length zero. Thus, if the streams and weights take on values from the field $\mathbb{F}_q$ (of sufficiently large size), nodes x_1 and x_5 will be able to recover each other's streams (and clearly their own streams). In particular, we have $|\mathcal{T}|\binom{N-|\mathcal{S}|}{2f}(N-|\mathcal{S}|-2f) = 6$, and so if we choose the weights randomly (independently and uniformly) from a field of size $q \geq 6$, the probability that the network code will allow x_1 and x_5 to exchange streams of information and identify any malicious node in the network will be at least $1 - \frac{6}{q}$. Suppose we choose $q = 128$, which would produce a probability of success equal to 0.953; using MATLAB, we generate a random set of weights from $\mathbb{F}_{2^7}$ (with primitive polynomial $\alpha^7 + \alpha^3 + 1$), which produces the weight matrix

$$\mathbf{W} = \begin{bmatrix} 0 & 12 & 20 & 18 & 0 \\ 115 & 0 & 0 & 0 & 4 \\ 16 & 0 & 0 & 0 & 107 \\ 115 & 0 & 0 & 0 & 118 \\ 0 & 122 & 101 & 121 & 0 \end{bmatrix} ;$$

note that the self-weights (on the diagonal of the weight matrix) can be set to zero without loss of generality, as discussed in Remark 5.4. Each of the numbers in the above matrix is the decimal representation of the appropriate field element (e.g., the entry "12" represents the element $\alpha^3 + \alpha^2$, where α satisfies $\alpha^7 + \alpha^3 + 1 = 0$). To complete the state-space model in (5.2), we note that $\mathbf{B}_{\mathcal{S}} = \left[\begin{smallmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{smallmatrix}\right]'$ and $\mathbf{B}_{\widehat{\mathcal{F}}}$ is a vector with a single "1" in the j–th position (if node x_j is malicious) and zeros elsewhere. Since sink node x_1 has access to its own transmission and the transmissions of nodes x_2, x_3 and x_4 at each time-step, the matrix $\mathbf{C}_1$ is given by

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} .$$

The matrix $\mathbf{C}_5$ for sink node x_5 can be obtained in a similar way.

The next step is to determine the smallest delay L for which the system given by the tuple $(\mathbf{W}, \mathbf{B}_{\mathcal{S}\cup\mathcal{J}}, \mathbf{C}_1)$ is invertible (for any set $\mathcal{J}$ of 2 nodes, disjoint from $\mathcal{S}$). Following Theorem 1.2, we find that $L = 2$ is the smallest integer for which rank $\left(\mathcal{M}_{1,L}^{\mathcal{S}\cup\mathcal{J}}\right)$ − rank $\left(\mathcal{M}_{1,L-1}^{\mathcal{S}\cup\mathcal{J}}\right) = |\mathcal{S}| + 2f = 4$. For example, if $\mathcal{J} = \{x_2, x_3\}$, we have

$$\mathcal{M}_{i,2}^{\mathcal{S}\cup\mathcal{J}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{C}_1\mathbf{B}_{\mathcal{S}\cup\mathcal{J}} & \mathbf{0} \\ \mathbf{C}_1\mathbf{W}\mathbf{B}_{\mathcal{S}\cup\mathcal{J}} & \mathbf{C}_1\mathbf{B}_{\mathcal{S}\cup\mathcal{J}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 12 & 20 & 1 & 0 & 0 & 0 \\ 115 & 4 & 0 & 0 & 0 & 0 & 1 & 0 \\ 16 & 107 & 0 & 0 & 0 & 0 & 0 & 1 \\ 115 & 118 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and it is easy to verify that the first $|\mathcal{S}| + 2f = 4$ columns of this matrix are linearly independent of all other columns. We omit the exact values of the matrices for other sets $\mathcal{J}$ in the interest of space. The above analysis indicates that node x_1 can recover node x_5's stream (and its own stream) despite the presence of $f = 1$ malicious node after $L + 1 = 3$ time-steps. The observability matrix $\mathcal{O}_{1,2}$ in Equation (5.4) is given by

$$\mathcal{O}_{1,2} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_1\mathbf{W} \\ \mathbf{C}_1\mathbf{W}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 12 & 20 & 18 & 0 \\ 115 & 0 & 0 & 0 & 4 \\ 16 & 0 & 0 & 0 & 107 \\ 115 & 0 & 0 & 0 & 118 \\ 3 & 0 & 0 & 0 & 41 \\ 0 & 3 & 22 & 94 & 0 \\ 0 & 72 & 107 & 7 & 0 \\ 0 & 64 & 104 & 2 & 0 \end{bmatrix}.$$

Node x_1 is provided with this matrix, and it can use this to generate the matrices $\mathcal{M}_{1,2}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}$

for any set $\widehat{\mathcal{F}}_j = \{x_j\}$ of one malicious node from the set $\{x_2, x_3, x_4\}$. Similarly, node x_5 is provided with the observability matrix $\mathcal{O}_{5,2}$, which it can use to generate the matrices $\mathcal{M}_{5,2}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}$. With this information in hand, the nodes are ready to exchange streams of information through the network.

Suppose that node x_1 wishes to transmit the stream of numbers

$$s_1[k] = \{104, 88, 40, 120, 4, 55, 48, \ldots\}$$

to node x_5, and node x_5 wants to transmit the stream of numbers

$$s_5[k] = \{63, 121, 43, 74, 28, 95, 32, \ldots\}$$

to node x_1. To accomplish this, all nodes in the network use the linear network code modeled by Equation (5.2) with the weight matrix $\mathbf{W}$ specified above. For example, node x_2 transmits the value $x_2[k+1] = 115x_1[k] + 4x_5[k]$ at time-step $k+1$. Suppose that node x_4 is malicious, and decides to modify its transmitted value in arbitrary ways. For example, at time-step $k = 1$, node x_4 decides to transmit the value $x_4[1] = 115x_1[0] + 118x_5[0] + 0$ (i.e., it commits an additive error of 0, and thus behaves normally). However, at time-step $k = 2$, it transmits the value $x_4[2] = 115x_1[1] + 118x_5[1] + 14$ (i.e., it commits an additive error of 14). The set of all additive errors committed by node x_4 is given by $f_4[k] = \{0, 14, 26, 39, 111, 104, \ldots\}$. All other nodes in the network behave normally at each time-step. The values transmitted by each node at each time-step with these source values and additive errors are given by

$$\begin{bmatrix}\mathbf{x}[0] & \mathbf{x}[1] & \mathbf{x}[2] & \mathbf{x}[3] & \mathbf{x}[4] & \mathbf{x}[5] & \cdots\end{bmatrix} = \begin{bmatrix}0 & 104 & 88 & 39 & 95 & 104 & \cdots \\ 0 & 0 & 109 & 60 & 61 & 26 & \cdots \\ 0 & 0 & 126 & 102 & 98 & 101 & \cdots \\ 0 & 0 & 107 & 122 & 10 & 31 & \cdots \\ 0 & 63 & 121 & 126 & 64 & 68 & \cdots\end{bmatrix}.$$

At each time-step k, node x_1 has access to the transmitted values $\mathbf{y}_1[k] = \mathbf{C}_1\mathbf{x}[k]$. Starting at time-step 0, the values seen by node x_1 up to time-step $L = 2$ are given by

$$\begin{aligned}\mathbf{y}_1[0] &= \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}' \\ \mathbf{y}_1[1] &= \begin{bmatrix}104 & 0 & 0 & 0\end{bmatrix}' \\ \mathbf{y}_1[2] &= \begin{bmatrix}88 & 109 & 126 & 107\end{bmatrix}'.\end{aligned}$$

Based on this information, node x_1 can use Theorem 5.2 to determine the first value transmitted by node x_5, along with the identity of any node that was malicious in the first

time-step. Specifically, from Equation (5.6), node x_1 knows that the values $\mathbf{y}_1[0:2]$ can be written as

$$\mathbf{y}_1[0:2] = \mathcal{O}_{1,2}\mathbf{x}[0] + \mathcal{M}_{1,2}^{\mathcal{S}\cup\widehat{\mathcal{F}}}\mathbf{u}[0:1] \ ,$$

where $\mathbf{u}[k] = \begin{bmatrix} s_1[k] & s_5[k] & f_4[k] \end{bmatrix}'$. Node x_1 does not know the set $\widehat{\mathcal{F}}$, so it tries to find a set $\widehat{\mathcal{F}}_j = \{x_j\}$, $j \in \{2,3,4\}$, of one node such that the vector $\mathbf{y}_1[0:2] - \mathcal{O}_{1,2}\mathbf{x}[0]$ is in the column space of the matrix $\mathcal{M}_{1,2}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}$. Performing this test, it finds that the above condition is satisfied only for $j = 4$ and finds a vector $\bar{\mathbf{u}}[0:1]$ such that

$$\mathbf{y}_1[0:2] - \mathcal{O}_{1,2}\mathbf{x}[0] = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 18 & 1 & 0 & 0 \\ 115 & 4 & 0 & 0 & 0 & 0 \\ 16 & 107 & 0 & 0 & 0 & 0 \\ 115 & 118 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathcal{M}_{1,2}^{\mathcal{S}\cup\{x_4\}}} \underbrace{\begin{bmatrix} 104 \\ 63 \\ 0 \\ 88 \\ 0 \\ 14 \end{bmatrix}}_{\bar{\mathbf{u}}[0:1]} .$$

Based on this, node x_1 determines that node x_5 transmitted the value $s_5[0] = 63$ at the first time-step, and that node x_4 committed an "additive error" of 0. Node x_1 therefore correctly concludes that there were no malicious nodes during the first time-step. It can now calculate the values transmitted by all nodes at time-step $k = 1$ as

$$\mathbf{x}[1] = \mathbf{W}\mathbf{x}[0] + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{\mathbf{B}_{\mathcal{S}\cup\{x_4\}}} \begin{bmatrix} 104 \\ 63 \\ 0 \end{bmatrix} = \begin{bmatrix} 104 \\ 0 \\ 0 \\ 0 \\ 63 \end{bmatrix} .$$

At time-step $k = 3$, node x_1 receives the values $\mathbf{y}_1[3] = \begin{bmatrix} 39 & 60 & 102 & 122 \end{bmatrix}'$. Once again, it tries to find a candidate set $\widehat{\mathcal{F}}_j = \{x_j\}$ of one malicious node so that the quantity $\mathbf{y}_1[1:3] - \mathcal{O}_{1,2}\mathbf{x}[1]$ is in the column space of the matrix $\mathcal{M}_{1,2}^{\mathcal{S}\cup\widehat{\mathcal{F}}_j}$. With the given values of $\mathbf{x}[1]$ and $\mathbf{y}_1[1:3]$, node x_1 tests $\widehat{\mathcal{F}}_4 = \{x_4\}$ (since that was the suspected malicious node

during the previous time-step) and finds that the required condition is satisfied. Node x_1 then finds a vector $\bar{\mathbf{u}}[1:2]$ such that

$$\mathbf{y}_1[1:3] - \mathcal{O}_{1,2}\mathbf{x}[1] = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 18 & 1 & 0 & 0 \\ 115 & 4 & 0 & 0 & 0 & 0 \\ 16 & 107 & 0 & 0 & 0 & 0 \\ 115 & 118 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathcal{M}_{1,2}^{\mathcal{S}\cup\{x_4\}}} \underbrace{\begin{bmatrix} 88 \\ 121 \\ 14 \\ 40 \\ 0 \\ 26 \end{bmatrix}}_{\bar{\mathbf{u}}[1:2]}.$$

Node x_1 thus finds that node x_5 transmitted the value $s_5[1] = 121$ at time-step 1, and that node x_4 was in fact malicious/faulty and committed an additive error of $f_5[1] = 14$. It can now update its estimate of the transmitted values as

$$\mathbf{x}[2] = \mathbf{W}\mathbf{x}[1] + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{\mathbf{B}_{\mathcal{S}\cup\{x_4\}}} \begin{bmatrix} 88 \\ 121 \\ 14 \end{bmatrix} = \begin{bmatrix} 88 \\ 109 \\ 126 \\ 107 \\ 121 \end{bmatrix}.$$

Node x_1 continues in this way and thereby recovers all of the values transmitted by source node x_5, as well as all of the additive errors caused by node x_4. Note that node x_1 does not need to find the candidate set $\widehat{\mathcal{F}}_j$ for future iterations of the decoding procedure, since it already knows that node x_4 is the malicious node. However, if we drop our assumption that the same node is malicious throughout the linear iteration, but instead assume that at most $f = 1$ node can be malicious during any time interval of length $L = 2$, the above decoding procedure will still allow node x_1 to recover the sources streams and identify the malicious nodes; in this case, node x_1 would have to test all possible sets of candidate malicious nodes at every time-step.

The decoding procedure for node x_5 is completely symmetric to the above procedure, and we omit the details here.

5.5 Comparisons to Existing Work

In order to more conveniently compare the results in this chapter to the existing literature on network coding (both with and without Byzantine adversaries), we first briefly discuss the extension of our results to networks where nodes are allowed to send *different* values on each of their outgoing edges (e.g., under the wired communications model). This is the standard scenario considered in much of the existing literature on network coding (although those results translate quite readily to the wireless model as well). We will now discuss this translation of a wired communication network to the "wireless broadcast" model that we consider in this thesis (similar translations can be found, for example, in [9, 90]), and then delve into a comparison of the proposed schemes.

5.5.1 Application to Wired Networks

Consider a network $\bar{\mathcal{G}} = \{\bar{\mathcal{X}}, \bar{\mathcal{E}}\}$, where $\bar{\mathcal{X}} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{\bar{N}}\}$ is the set of nodes in the network, and $\bar{\mathcal{E}} = \{(\bar{x}_i, \bar{x}_j)\}$ is the set of edges between nodes. The *head* of an edge $e = (\bar{x}_i, \bar{x}_j)$ is given by $\text{head}(e) = \bar{x}_j$ and the tail of the edge is given by $\text{tail}(e) = \bar{x}_i$. In this section, we allow for multiple edges between nodes and thus $\bar{\mathcal{G}}$ is a *multigraph* [28]. For convenience, suppose that there is a single source node $\bar{x}_1$ and a set $\mathcal{T}$ of sink nodes in the network.[6] The source node receives $|\mathcal{S}|$ input streams, which it must transmit through the network to the sink nodes. To model these streams, we will add a "dummy" source node $\bar{x}_S$ to the network, with $|\mathcal{S}|$ edges of the form $(\bar{x}_S, \bar{x}_1)$. At each time-step k, each node (other than $\bar{x}_S$) transmits on each of its outgoing edges a linear combination of the packets that it receives on its incoming edges (the linear combination used for each outgoing edge can be different). We also assume that up to f edges in the network can be under the control of a malicious attacker; specifically, if edge $(\bar{x}_i, \bar{x}_j)$ is controlled by the attacker, the value received by node $\bar{x}_j$ is the value that was transmitted by $\bar{x}_i$ on that edge along with an additive error. We assume that the edges $(\bar{x}_S, \bar{x}_1)$ are fault-free (since otherwise, additive errors will be indistinguishable from the source streams).

Note that in the above scenario, each *edge* in the network has some linear combination of the source values (along with a potential additive error) associated with it. However, this model can be easily translated to the model that we consider in this thesis (where each *node* has an associated linear combination of the source streams, along with a potential additive error) in the following manner. Define the *line graph* of the network as $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\} = \bar{\mathcal{E}} \cup \mathcal{T}$; in other words, every edge in the network $\bar{\mathcal{G}}$ (including every edge between the dummy source $\bar{x}_{\mathcal{S}}$ and $\bar{x}_1$) becomes a node in the network $\mathcal{G}$, and the sink nodes from $\bar{\mathcal{G}}$ are duplicated in $\mathcal{G}$, which means that $N = |\bar{\mathcal{E}}| + |\mathcal{T}|$.[7] The edge set $\mathcal{E}$

[6] This can be extended to the case of multiple source nodes, but we will consider this simple scenario here for pedagogical reasons and to enable comparisons to the existing work in the literature.

[7] It is actually not necessary to include the source edges and the sink nodes in the graph $\mathcal{G}$ in order to analyze the system, but we do it here in order to obtain a model that is equivalent to the one considered

is defined as follows. Consider two nodes $x_i, x_j \in \mathcal{X}$, and suppose that both of these two nodes correspond to edges in $\bar{\mathcal{G}}$. Then, $(x_i, x_j) \in \mathcal{E}$ if $\text{head}(x_i) = \text{tail}(x_j)$ in $\bar{\mathcal{G}}$ (i.e., there is an edge from x_i to x_j in $\mathcal{G}$ if, in the graph $\bar{\mathcal{G}}$, the edge corresponding to x_i feeds into the edge corresponding to x_j). Furthermore, consider an edge that feeds into a sink node in $\bar{\mathcal{G}}$, and let x_i and x_j denote the corresponding edge node and sink node in $\mathcal{G}$, respectively; then, $(x_i, x_j) \in \mathcal{E}$. An example of this transformation is shown in Figure 5.2.

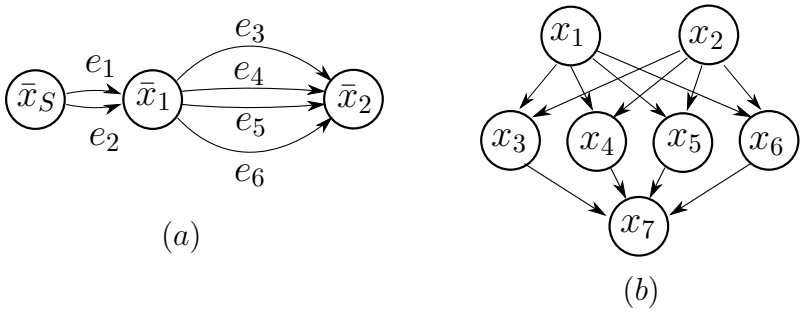

Figure 5.2: (a) The network $\bar{\mathcal{G}}$. The source node $\bar{x}_1$ has four edge-disjoint paths to the sink node $\bar{x}_2$. The dummy source node $\bar{x}_S$ injects two source streams into the source node $\bar{x}_1$ for transmission to $\bar{x}_2$. (b) The corresponding network $\mathcal{G}$. The mapping from $\bar{\mathcal{G}}$ to $\mathcal{G}$ is as follows: $e_1 \to x_1$, $e_2 \to x_2$, $e_3 \to x_3$, $e_4 \to x_4$, $e_5 \to x_5$, $e_6 \to x_6$, $\bar{x}_2 \to x_7$. The sink node in $\mathcal{G}$ is given by $\mathcal{T} = \{x_7\}$ and the source nodes are given by $\mathcal{S} = \{x_1, x_2\}$.

The graph $\mathcal{G}$ is now essentially in the form that was considered earlier in this chapter. Specifically, $\mathcal{G}$ contains a set of source nodes $\mathcal{S}$ (each corresponding to one of the edges of the form $(\bar{x}_S, \bar{x}_1)$ in $\bar{\mathcal{G}}$), and a set $\mathcal{T}$ of source nodes. The f (or fewer) malicious edges in $\bar{\mathcal{G}}$ are now given by a set $\mathcal{F}$ of f (or fewer) malicious nodes in $\mathcal{G}$. The sink node $x_i \in \mathcal{T}$ in $\mathcal{G}$ has access to the values transmitted by its neighbors $\mathcal{N}_i$ at each time-step (note that these neighbors represent the incoming edges to the sink node in the graph $\bar{\mathcal{G}}$); these values can be represented as $\mathbf{y}_i[k] = \mathbf{C}_i \mathbf{x}[k]$, where $\mathbf{C}_i$ is a $(\deg_i + 1) \times N$ matrix with a single "1" in each row corresponding to a node in $\mathcal{N}_i \cup \{x_i\}$. Denoting the value transmitted by node x_j (or equivalently, the value transmitted on the corresponding edge in $\bar{\mathcal{G}}$) at time-step k by $x_j[k]$, we obtain the linear system model given by Equations (5.2) and (5.3).

Before proceeding, it will be useful for us to understand how topological conditions in the graph $\bar{\mathcal{G}}$ are reflected in the graph $\mathcal{G}$. Consider two nodes $\bar{x}_i$ and $\bar{x}_j$ in the graph $\bar{\mathcal{G}}$. Let $x_{i,1}, x_{i,2}, \ldots, x_{i,a}$ denote the vertices in $\mathcal{G}$ corresponding to the outgoing edges from $\bar{x}_i$ in $\bar{\mathcal{G}}$, and let $x_{j,1}, x_{j,2}, \ldots, x_{j,b}$ denote the vertices in $\mathcal{G}$ corresponding to the incoming edges to $\bar{x}_j$ in $\bar{\mathcal{G}}$. It is then easy to argue that if there are r edge-disjoint paths from $\bar{x}_i$ to $\bar{x}_j$ in $\bar{\mathcal{G}}$, then there will be an r-linking from the set $\{x_{i,1}, x_{i,2}, \cdots, x_{i,a}\}$ to the set $\{x_{j,1}, x_{j,2}, \ldots, x_{j,b}\}$ in $\mathcal{G}$ [28]. Note that if $\bar{x}_j$ is a sink node in $\bar{\mathcal{G}}$, and x_j is the corresponding sink node in $\mathcal{G}$, the above condition is equivalent to saying that there are r internally node-disjoint paths from $\{x_{i,1}, x_{i,2}, \ldots, x_{i,a}\}$ to x_j in $\mathcal{G}$ (since x_j explicitly appears in $\mathcal{G}$ and has incoming edges

earlier in this chapter.

from each of the nodes in $\{x_{j,1}, x_{j,2}, \ldots, x_{j,b}\}$ by construction). For example, the graph $\bar{\mathcal{G}}$ shown in Figure 5.2.a has two edge-disjoint paths from $\bar{x}_S$ to $\bar{x}_2$, and the associated graph $\mathcal{G}$ has two internally node-disjoint paths from the nodes $\{x_1, x_2\}$ to x_7. This fact will allow us to translate existing topological conditions that are described in the network coding literature to the wireless model, and then compare them to the condition that is presented in Theorem 5.1.

5.5.2 Comparison with Existing Work on Network Coding

Comparison of Network Topology Conditions for Tolerating Byzantine Adversaries

In [37], it was shown that if there are $|\mathcal{S}| + 2f$ edge-disjoint paths from the source node $\bar{x}_1$ to every sink node in $\bar{\mathcal{G}}$, then every sink node can recover the $|\mathcal{S}|$ source streams despite the presence of up to f malicious edges in the network. In the graph $\mathcal{G}$, let $\{x_{i,1}, x_{i,2}, \ldots, x_{i,a}\}$ denote the nodes that correspond to the outgoing edges from the source node in $\bar{\mathcal{G}}$. The condition that there are $|\mathcal{S}| + 2f$ edge-disjoint paths from the source node to every sink node in $\bar{\mathcal{G}}$ then translates to the following condition.

Condition 1: To transmit $|\mathcal{S}|$ streams of information despite f malicious edges in $\bar{\mathcal{G}}$, there must exist $|\mathcal{S}| + 2f$ internally node-disjoint paths from $\{x_{i,1}, x_{i,2}, \ldots, x_{i,a}\}$ to every sink node in $\mathcal{G}$.

For example, consider once again the network in Figure 5.2.a. The source node $\bar{x}_1$ has 4 edge-disjoint paths to the sink node $\bar{x}_2$, and if there is up to $f = 1$ malicious edge, the results in [37] indicate that one can transmit $|\mathcal{S}| = 4 - 2f = 2$ streams of information through the network. Equivalently, we can examine the graph $\mathcal{G}$ in Figure 5.2.b and note that there exist $|\mathcal{S}| + 2f = 4$ internally node-disjoint paths from the set $\{x_3, x_4, x_5, x_6\}$ to the sink node x_7.

The analysis that we performed earlier in this chapter differs from the above work (and other related works such as [40]) in that we consider the problem of both recovering the source stream *and* identifying the set of malicious nodes (or edges in the wired model); as might be expected, the conditions required on the network topology to achieve this objective are stronger than the conditions required for only recovering the source streams. Specifically, from Theorem 5.1, our procedure requires the following condition.

Condition 2: To transmit $|\mathcal{S}|$ streams of information and identify f malicious edges in $\bar{\mathcal{G}}$, there must exist $|\mathcal{S}| + 2f$ node-disjoint paths from the set $\mathcal{S} \cup \mathcal{J}$ (where $\mathcal{J}$ is *any* set of $2f$ nodes disjoint from $\mathcal{S}$) to every sink node in $\mathcal{G}$.

Note that if Condition 2 is satisfied, then Condition 1 will also hold. To see this, let $\mathcal{J}$ be a set of $2f$ nodes from the set $\{x_{i,1}, x_{i,2}, \ldots, x_{i,a}\}$ (these nodes correspond to the outgoing edges from the source node in $\bar{\mathcal{G}}$). If $\mathcal{G}$ contains $|\mathcal{S}| + 2f$ internally node-disjoint paths from $\mathcal{S} \cup \mathcal{J}$ to every sink node, then it must also contain $|\mathcal{S}| + 2f$ internally node-disjoint paths

from $\{x_{i,1}, x_{i,2}, \ldots, x_{i,a}\}$ to every sink node (since every path originating from a source node in $\mathcal{S}$ must pass through one of the nodes in $\{x_{i,1}, x_{i,2}, \ldots, x_{i,a}\}$), and this is precisely Condition 1. However, in general, there will be graphs where Condition 1 is satisfied but Condition 2 is not. For example, the graph shown in Figure 5.2 satisfies both Condition 1 and Condition 2, whereas the graph shown in Figure 5.3 satisfies Condition 1 (there are $|\mathcal{S}| + 2f = 3$ internally node-disjoint paths from $\{x_2, x_3, x_4\}$ to x_8) but not Condition 2. For instance, if we consider $\mathcal{J} = \{x_2, x_5\}$, one can immediately see that there are not $|\mathcal{S}| + 2f = 3$ internally node-disjoint paths from $\mathcal{S} \cup \mathcal{J}$ to x_8. Thus, one cannot identify all malicious edges in this network (even though one can recover all the source streams using the techniques in [37]). The intuition behind this is clear from the graph $\bar{\mathcal{G}}$: the sink node would never be able to distinguish malicious behavior on edge e_5 from malicious behavior on edge e_2.[8]

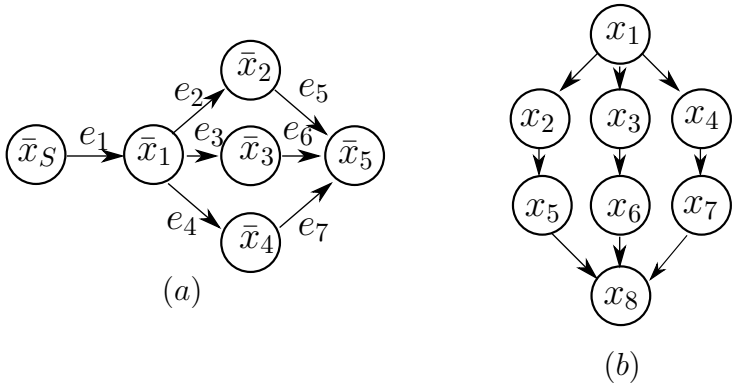

Figure 5.3: (a) The network $\bar{\mathcal{G}}$. The source node $\bar{x}_1$ has three edge-disjoint paths to the sink node $\bar{x}_5$. The dummy source node $\bar{x}_S$ injects one source stream into the source node $\bar{x}_1$ for transmission to $\bar{x}_5$. (b) The corresponding network $\mathcal{G}$. The sink node in $\mathcal{G}$ is given by $\mathcal{T} = \{x_8\}$ and the source node is given by $\mathcal{S} = \{x_1\}$.

Comparison of Decoding Schemes: Convolutional Network Codes

One of the benefits of our framework is that it provides a systematic method to design recursive decoders for linear network codes for arbitrary networks (with or without Byzantine attackers). Specifically, when the network topology contains cycles, the input packets to some nodes in the network could be functions of previous output packets at those nodes. To deal with this phenomenon, the network coding literature introduces a delay parameter z into the network (e.g., by modeling either the links or the nodes as having delays), and views the packets generated by each node as a power series in z [9, 84, 82], thereby producing a *convolutional network code*. Decoders for such codes are traditionally designed by forming

[8] This phenomenon of indistinguishable faults on links has been previously noted in [91, 39]; the difference here is that we provide conditions on the network topology that allow all malicious edges to be located, along with a decoding procedure to do so.

a transfer function matrix (with elements that are rational functions in z) from the source stream to the sink nodes, and then using this matrix to extract the source streams [9, 82]. For example, an algorithmic procedure for choosing the weights for convolutional network codes was provided in [86, 90], and a decoding procedure was proposed that essentially requires the sink nodes to solve a system of equations in the delay parameter z. However, a drawback of this approach is the need to construct and manipulate (potentially large) matrices of rational functions in the parameter z.

In contrast, our decoding procedure is based on the linear state-space model given by Equations (5.2) and (5.3); inspired by results pertaining to dynamic system inversion, we show that every sink node only has to examine the values that it receives over a certain length of time in order to decode the source streams. Furthermore, the operations performed by the sink node involve purely numerical matrices, and do not require manipulation of transfer function matrices in the delay variable z. Indeed, this was the main motivation that drove the development of dynamic inversion schemes in the control literature, and there are various results in this area that can be leveraged to design *system inverters* (or decoders, in our context) purely in the state-space domain (e.g., see [43, 92]). As mentioned in the introduction, a state-space model for network codes (without malicious attackers) was also formulated in [85], but this model was not fully utilized in order to design a decoder for the code.

Comparison of Decoding Schemes: Byzantine Adversaries

The details of the decoding scheme described in [37] to overcome Byzantine attacks on network codes are as follows. If we let U denote the source packets, and Z denote the malicious packets that are injected on f edges, then the packets that are received by the sink node are given by

$$Y = TU + T'Z, \tag{5.8}$$

where T is the transfer function from the source node to the sink node, and T' is the transfer function from the malicious edges to the sink. The authors of [37] assume that the graph topology is unknown (or even changing), and thus use the standard technique of sending the coefficients along with the data packets to the sink nodes. In the presence of malicious nodes, this technique is complicated by the fact that T and T' in (5.8) are partially unknown. Under this condition, the authors show that the input stream can only be determined to lie within a certain subspace, but cannot be uniquely decoded otherwise. However, the authors also show that this problem can be circumvented by adding redundancy to the input stream in a certain manner, which then allows each sink node to exactly recover the input stream despite the maliciously injected packets (with high probability).

The key difference between our work and [37, 40] is that the network topology and weights are assumed to be *known* in our analysis. This essentially amounts to knowing the

transfer functions T and T' (for any set of f malicious edges); we show that this information (together with the topological condition discussed earlier) is enough to allow the sink nodes to recover the source streams and identify the malicious nodes (or edges), without requiring preprocessing or added redundancy at the source. A benefit of our decoding procedure is that it will work even with time-varying sets of malicious nodes (as long as there are no more than f malicious nodes within any time interval of a certain length). It is not immediately clear whether the approach in [37, 40] can handle such cases.

Summary of Differences

In summary, the key differences between the work in this chapter and previous work on network coding are as follows.

- We consider the problem of decoding the source streams *and* identifying the malicious nodes, which imposes stronger conditions on the network topology than that required purely for recovering the source streams.
- The work in [37, 40] has the benefit of not assuming any knowledge about the network topology (aside from the existence of sufficiently many edge-disjoint paths from the source to the sink). In contrast, we assume in our work that the network topology and weights are known to the sink nodes in the network, but we show that this knowledge allows us to design a decoding procedure that does not require preprocessing or the addition of redundancy at the source.
- In contrast to existing work on convolutional network codes, we provide a systematic decoding procedure that involves purely numerical matrices, and does not require manipulation of transfer function matrices in the delay variable z.

5.6 Extensions

There are a variety of interesting extensions that can be pursued within the framework that we have developed in this chapter. For example, the decoding procedure specified by the proof of Theorem 5.2 essentially requires each sink node to reconstruct the values transmitted by all nodes in the network at each time-step (i.e., the decoder for each sink node has dimension equal to the number of nodes in the system). However, based on the connections that we have made in this chapter between linear network coding and the theory of dynamic system inversion, it may be possible to design more efficient decoders by using results from the topic of *minimal dynamic inversion*, which consider the problem of designing system inverses of smallest possible dimension [93, 94].

As another extension, one can investigate conditions on the network topology that are needed for each sink node to reconstruct certain *functions* of the source streams and fault

streams (reconstructing all of the source and fault streams is a special case of this problem). Depending on the function to be reconstructed, the condition that there exist $|\mathcal{S}|+2f$ node-disjoint paths from $\mathcal{S}$ and any other set of $2f$ nodes to each sink node may no longer be necessary; however, with this relaxation, each sink node will no longer be able to determine the values transmitted by all nodes in the system (e.g., it may not know what values were transmitted by the malicious nodes), and thus designing a decoder for the sink nodes will become more complicated. One possible way to handle this could be to make use of results on *partial system inversion*, which deal with the topic of constructing a system inverse that recovers only some of the unknown inputs to a linear system [95, 96, 97]. Extending such results to finite fields (with appropriate graph-based characterizations), and leveraging them to design decoders for the sink nodes would be an interesting avenue for future research.

5.7 Summary

In this chapter, we considered the problem of using linear network codes to perform multisource multicast in networks with malicious nodes. We showed that, under the wireless broadcast communication model, the network code can be compactly represented as a linear system with unknown inputs. We then used concepts pertaining to system inversion and structured system theory to show that the multisource multicast problem with up to f malicious nodes is solvable if there exist vertex disjoint paths from the set of source nodes and any set of $2f$ other nodes to each sink node and its neighbors. Furthermore, we showed that if this topological condition is satisfied, choosing the weights for the network code independently and uniformly from the field $\mathbb{F}_q$ of sufficiently large size will allow all sink nodes to recover the source streams and identify malicious nodes with high probability. We showed that the maximum latency required to decode the source streams is given by $N-|\mathcal{S}|-2f+1$, and provided a systematic procedure for each sink node to follow in order to decode the source streams and identify all malicious nodes in a distributed manner.

CHAPTER 6

SUMMARY AND FUTURE WORK

6.1 Summary

In this thesis, we have studied the use of linear iterative strategies for information dissemination and distributed computation in networks and networked systems. In such strategies, each node in the network repeatedly updates its value to be a weighted linear combination of its previous value, and those of its neighbors. We showed how networks running these strategies can be modeled as linear dynamical systems, and developed a control theoretic framework to analyze the features and capabilities of the strategies. Specifically, we demonstrated the following key results.

1. Using observability theory, we showed that if the weights for the linear iteration are chosen randomly from a field of sufficiently large size, then with high probability, the linear iterative strategy allows any node to obtain the values of any other node (in strongly connected networks) after a finite number of iterations. In fact, our analysis revealed that the number of time-steps required for any node to accumulate all of the initial values via a linear iterative strategy is no greater than the size of the largest tree in a certain subgraph of the network; we conjectured that this quantity is actually the minimum number of time-steps required for *any* protocol to disseminate information in networks, in which case linear iterative strategies are time-optimal for any given network.

2. When transmissions and information exchanges in the network are affected by additive noise, we showed that linear iterative strategies allow nodes in the network to obtain unbiased estimates of any linear function of the initial values after a finite number of time-steps (once again, no greater than the size of the largest tree in a certain subgraph of the network). Furthermore, we showed that if the second-order statistics of the noise are also known, each node can minimize the mean square error associated with its estimate (for a given weight matrix) by taking an appropriate linear combination of the noisy values that it receives over the course of the linear iterative strategy.

3. We demonstrated that linear iterative strategies are inherently robust to malicious nodes by virtue of the network topology. Specifically, we showed that if there exists

a pair of nodes x_i and x_j that can be disconnected from each other by removing $2f$ or fewer nodes, then f malicious nodes can conspire to update their values in such a way that x_i cannot calculate any function of x_j's initial value. We provided a specific strategy for the malicious nodes to follow in order to disrupt the linear iterative strategy in this fashion. However, we also showed that if every node in the network has at least $2f + 1$ node-disjoint paths to x_i, then f malicious nodes *cannot* prevent x_i from correctly calculating any function of the initial values (and identifying the malicious nodes) when the linear iterative strategy is used. In particular, we showed that this can be achieved with almost any choice of real-valued weights for the linear iterative strategy, and will require at most N time-steps (where N is the number of nodes in the network).

4. We treated the problem of transmitting a stream of data from a set of source nodes in the network (as opposed to initial values from every node in the network) to a set of sink nodes, even when some nodes in the network potentially transmit arbitrary values at each time-step. We considered linear network codes that have been developed to solve this problem, and showed that such codes can be conveniently modeled as linear dynamical systems with unknown inputs. We showed that if the network topology contains node-disjoint paths from the set of source nodes and any other set of $2f$ nodes to each sink node and its neighbors, then a random choice of weights for the linear network code from a field of sufficiently large size will allow the sink nodes to recover the source streams and identify the malicious nodes with high probability.

5. We developed a theory of structured linear systems over finite fields; specifically, we showed that if the graph associated with a given structured linear system satisfies certain topological properties, then the system will be observable and invertible with high probability after a random choice of free parameters from a field of sufficiently large size. In the process of deriving our results, we obtain an improved bound on the observability index of linear systems in terms of the topology of the associated graph.

6.2 Future Work

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

1. We showed that linear iterative strategies inherently allow every node to obtain all of the initial values after at most D_i time-steps (where D_i is the size of the largest tree in an optimal spanning forest of the network). We conjectured that no protocol can disseminate information in a fewer number of time-steps – proving this conjecture

would establish linear iterative strategies as time-optimal methods of disseminating information in networks.

2. As described in Section 2.4, it is possible for the nodes to calculate their observability matrices (and the gains Γ_i) in a distributed manner when the network is noise-free and all nodes follow the specified protocol. However, using such techniques in networks with noisy transmissions or with malicious nodes is complicated by the fact that each node can only obtain uncertain information about the network from its neighbors – developing methods to minimize or eliminate this uncertainty will be necessary in order to obtain fully decentralized versions of the linear iterative strategies.

3. In our study of linear iterative strategies in noisy networks, we had each node run the linear iteration for the smallest number of time-steps required for unbiased estimation of the desired linear function of the initial values. If we allow each node to run the linear iteration for more time-steps than this minimum value, each node can potentially reduce the variance of its estimate (since it is obtaining more information about the initial state). The variance of the estimate will be a nonincreasing function of the number of time-steps, and thus the tradeoff between delay and variance can be taken as a design parameter for a given graph. A quantitative characterization of the relationship between delay and variance will be the subject of future research.

4. While we were able to obtain characterizations of structural observability and invertibility over finite fields, we do not currently have a similar result pertaining to strong observability. Obtaining such a characterization would allow us to immediately show that linear iterative strategies are resilient to malicious nodes even when all operations and transmissions are performed over finite fields.

5. We provided a checking strategy for each node in the network to detect and overcome malicious behavior; this strategy requires the node to test up to $\binom{N}{f}$ possibilities when trying to determine the possible sets of malicious nodes. It is worth noting that this is equivalent to the brute force method of determining up to f errors in an N-dimensional real number codeword with distance $2f+1$. In the coding theory literature, there exist efficient ways of performing this check for both structured and random real number codes (e.g., see [81] and the references therein), and one can potentially exploit those results to streamline the checking procedure that we have proposed. Similarly, if the network involves both malicious nodes *and* noisy transmissions between nodes, then more complex decoding procedures will have to be developed to identify malicious behavior. Clearly the line between noise and malicious errors becomes blurred as the magnitude of the noise increases, which makes it harder to handle such behavior. This issue have been investigated in the context of real-number error correcting codes in [81], and we will leave connections to such work for future research.

6. As discussed in Section 5.6, there are a variety of interesting extensions that can be pursued within our framework for linear network coding. For example, one can pursue some of the results from the literature on *minimal dynamic inversion* (which consider the problem of designing system inverses of smallest possible dimension [93, 94]) in order to design decoders for each sink node that are of smallest possible dimension. Similarly, one can also potentially use results from the literature on *partial system inversion* in order to derive conditions on the network topology and to design decoders that allow sink nodes to recover the source streams without necessarily identifying and locating the malicious nodes. These extensions (if successful) would reduce the computational effort needed by the sink nodes in order to successfully recover the streams of values transmitted by the source nodes.

7. In our current work, we have designed the linear iterative scheme for fixed (time-invariant) network topologies. Small or intermittent changes in network topology can be handled within our framework by treating them as faults and using our results from Chapter 4; however, it will be of interest to extend our results to cases when the network topology is inherently time varying, with potentially drastic changes in structure. In the consensus literature, such cases are handled by only requiring the system to reach consensus asymptotically (as discussed in Chapter 1), and not in finite time. In the network coding literature, robustness to changes in network topology is handled by having each node transmit the weights of the linear combination along with its updated value; after a node receives a sufficient number of values and coefficients, it can reconstruct the desired information by inverting the matrix of coefficients. It is an open question as to whether the framework that we have developed in this work can be extended or modified to handle general time-varying networks.

APPENDIX

PROPERTIES OF STRUCTURED LINEAR SYSTEMS OVER FINITE FIELDS

A.1 Introduction

Recall from Section 1.5.1 that a structured system is one where each entry in the system matrices is either identically zero, or an independent free parameter from a field $\mathbb{F}$. Certain properties of such systems can be inferred by examining the graph $\mathcal{H}$ associated with the system. This graph-based analysis will turn out to be quite useful to our discussion on linear iterative strategies in distributed systems, and so structured system theory will play a key role in this thesis. However, as mentioned in Chapter 1, almost all of the existing work on structured systems only deals with the case where the free parameters are chosen from the field of real numbers (with the analysis being performed over the field of complex numbers). With this assumption, these previous works rely on tests (such as the Hautus-Rosenbrock condition for observability) to determine properties of real-valued matrix sets [41, 52]; however, these proof techniques do not extend to the case where the parameters in the matrices are chosen from finite fields due to the fact that such fields are not algebraically closed (as discussed in Section 1.5.1). While we will be able to use these existing results to derive linear iterative strategies with real-valued transmissions and operations, we will also be interested in designing linear iterative strategies where nodes can only perform operations in a finite field. This situation arises in many practical situations, such as in networks with bandwidth restrictions in the transmission channels between nodes, or networks with nodes that are limited in the precision of the computations that they perform. In order to design such strategies, we will first need to develop a theory of structured linear systems over finite fields, and that is topic of this appendix.

We will start by considering the topic of structural system observability over finite fields, and then we will investigate the problem of structural system invertibility over finite fields. Specifically, we will show that if the graph associated with the system satisfies certain conditions, and if the free parameters for the system are chosen randomly (independently and uniformly) from the finite field $\mathbb{F}_q$ (of size q), then the system will possess certain properties (such as observability or invertibility) with some probability that increases with q. For fields of infinite size, our results corroborate the notion of genericness of structural properties that has been established in the literature, since the system properties will hold

with probability 1. Our development will also allow us to obtain a characterization of the *generic observability index* of a given structured system, which we will define to be the observability index that is attained with high probability for a random choice of free parameters; such a characterization does not currently exist in the literature.

In our development, we will make use of the following lemma pertaining to the roots of a given multivariate polynomial over finite fields (e.g., see [55, 98]). In this lemma, the *total degree* of a multivariate polynomial $p(\xi_1, \xi_2, \ldots, \xi_n)$ is defined as the maximum sum of the degrees of the variables $\xi_1, \xi_2, \ldots, \xi_n$ in any term of the polynomial.

Lemma A.1 (Schwartz-Zippel) *Let $p(\xi_1, \xi_2, \ldots, \xi_n)$ be a nonzero polynomial of total degree d with coefficients in the finite field $\mathbb{F}_q$ (with $q \geq d$). If p is evaluated on an element $(s_1, s_2, \ldots, s_n)$ chosen uniformly at random from $\mathbb{F}_q^n$, then*

$$\Pr[p(s_1, s_2, \ldots, s_n) = 0] \leq \frac{d}{q} \ .$$

It is worth noting that tighter bounds can be obtained on the probability of a given polynomial being zero after a random choice of parameters (e.g., see [55]), and these can also be applied in the proofs of our results. We will work with the bound provided by the above lemma for convenience.

A.2 Structural Observability over Finite Fields

We will start by investigating the observability of matrix pairs of the form $(\mathbf{A}, \mathbf{e}'_{1,N})$, where $\mathbf{A}$ is an $N \times N$ matrix, and $\mathbf{e}'_{1,N}$ is a row-vector of length N with a 1 in its first position and zeros elsewhere. Matrix $\mathbf{A}$ may be *structured* (i.e., every entry of $\mathbf{A}$ is either zero, or an independent free parameter to be chosen from a field $\mathbb{F}$), or it may be numerically specified. Our analysis will be based on a graph representation of matrix $\mathbf{A}$, denoted by $\mathcal{H}$, which we obtain as follows. The vertex set of $\mathcal{H}$ is $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, and the edge set is given by $\mathcal{E} = \{(x_j, x_i) \mid \mathbf{A}_{ij} \neq 0\}$. The weight on edge (x_j, x_i) is set to the value of $\mathbf{A}_{ij}$ (this can be a free parameter if $\mathbf{A}$ a structured matrix). Note that if $\mathbf{A}$ is the weight matrix for a linear iteration, $\mathcal{H}$ is simply the graph of the network $\mathcal{G}$ augmented with a self-loop on node x_i if $\mathbf{A}_{ii} \neq 0$.

A.2.1 Observability of a Spanning Tree

We will start by considering a linear system whose graph is a spanning tree. For such systems, we will show the following result.

Theorem A.1 *Consider the matrix pair $(\mathbf{A}, \mathbf{e}'_{1,N})$, where $\mathbf{A}$ is an $N \times N$ matrix with elements from a field $\mathbb{F}$ of size at least N. Suppose that the following two conditions hold:*

- *The graph $\mathcal{H}$ associated with $\mathbf{A}$ is a spanning tree rooted at x_1 with self-loops on every node.*

- *The weights on the self-loops are different elements of $\mathbb{F}$ for every node, and the weights on the edges between different nodes are equal to* 1.

Then the pair $(\mathbf{A}, \mathbf{e}'_{1,N})$ is observable over the field $\mathbb{F}$.

In the proof of the theorem, we will use the well-known fact that the rank of the observability matrix for the pair $(\mathbf{A}, \mathbf{e}'_{i,N})$ is equal to the rank of the observability matrix for the pair $(\mathbf{TAT}^{-1}, \mathbf{e}'_{1,N}\mathbf{T}^{-1})$, for any invertible matrix $\mathbf{T}$ [41].

Proof: [Theorem A.1] Since the graph associated with $\mathbf{A}$ is a spanning tree rooted at x_1 (and, in particular, it is acyclic and each node has out-degree equal to 1, except for node x_1), there exists a numbering[1] of the nodes such that the $\mathbf{A}$ matrix is upper-triangular, with the self-loop weights on the diagonal [28, 52]. Denote the self-loop weight on node x_i by λ_i. Since all of the self-loop weights are different, this matrix will have N distinct eigenvalues (given by $\lambda_1, \lambda_2, \ldots, \lambda_N$), with N corresponding linearly independent eigenvectors. We will now show that the first element in each of these eigenvectors will be "1", and then we will use this fact to prove observability.

Consider the eigenvalue λ_i. Let x_l be any node in the graph with in-degree 0 such that the path from x_l to x_1 passes through x_i (if the in-degree of x_i is zero, we can take $x_l = x_i$). Let r_i denote the number of nodes in this path, and reorder the nodes (leaving x_1 unchanged) so that all nodes on the path from x_l to x_1 come first in the ordering, and all other nodes come next. Let $\mathbf{P}_i$ denote the permutation matrix that corresponds to this reordering, and note that the matrix $\mathbf{P}_i\mathbf{A}\mathbf{P}_i^{-1}$ has the form

$$\mathbf{P}_i\mathbf{A}\mathbf{P}_i^{-1} = \begin{bmatrix} \mathbf{J}_i & \bar{\mathbf{A}}_1 \\ \mathbf{0} & \bar{\mathbf{A}}_2 \end{bmatrix} , \tag{A.1}$$

for some matrices $\bar{\mathbf{A}}_1$ and $\bar{\mathbf{A}}_2$. The matrix $\mathbf{J}_i$ has the form

$$\mathbf{J}_i = \begin{bmatrix} \lambda_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \lambda_3 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \lambda_{r_i-1} & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \lambda_{r_i} \end{bmatrix} ,$$

[1] This renumbering simply corresponds to performing a similarity transformation on $\mathbf{A}$ with a permutation matrix, and thus does not change the eigenvalues of the matrix.

where $\lambda_1, \lambda_2, \ldots, \lambda_{r_i}$ are different elements of $\mathbb{F}$. This matrix has r_i distinct eigenvalues (given by the λ_t's) in the field $\mathbb{F}$, and thus the matrix has r_i eigenvectors over $\mathbb{F}$. Note that there exists some $t \in \{1, 2, \ldots, r_i\}$ such that $\lambda_t = \lambda_i$ (where λ_i is the eigenvalue that we are considering in matrix $\mathbf{A}$). It is easy to verify that the eigenvector $\mathbf{v}_t$ of $\mathbf{J}_i$ associated with the eigenvalue λ_t is given by

$$\mathbf{v}_t = \begin{bmatrix} 1 & (\lambda_t - \lambda_1) & (\lambda_t - \lambda_1)(\lambda_t - \lambda_2) & \cdots & \prod_{s=1}^{t-1}(\lambda_t - \lambda_s) & 0 & \cdots & 0 \end{bmatrix}' ,$$

and thus the eigenvector corresponding to eigenvalue λ_t for the matrix $\mathbf{P}_i \mathbf{A} \mathbf{P}_i^{-1}$ in Equation (A.1) is given by

$$\mathbf{w}_t = \begin{bmatrix} \mathbf{v}_t \\ \mathbf{0} \end{bmatrix} .$$

Next, note that the eigenvector corresponding to eigenvalue λ_t (or equivalently, λ_i) for matrix $\mathbf{A}$ will be given by $\mathbf{P}_i \mathbf{w}_t$. Since $\mathbf{P}_i$ is a permutation matrix, and node x_1 was left unchanged during the permutation, the first row of $\mathbf{P}_i$ is given by the vector $\mathbf{e}'_{1,N}$. This means that the first element of the eigenvector $\mathbf{P}_i \mathbf{w}_t$ will be "1" (based on the matrices $\mathbf{w}_t$ and $\mathbf{v}_t$ shown above). Since the above analysis holds for any eigenvalue λ_i, we can conclude that all eigenvectors for the matrix $\mathbf{A}$ will have a "1" as their first element. Let $\mathbf{V}$ be the matrix whose columns are these eigenvectors (so that each entry in the first row of $\mathbf{V}$ is "1"); since the eigenvectors are linearly independent, this matrix will be invertible over the field $\mathbb{F}$. We thus have $\mathbf{V}^{-1}\mathbf{A}\mathbf{V} = \Lambda$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$, and furthermore, $\mathbf{e}'_{1,N}\mathbf{V} = \mathbf{1}'_N$.

Now, consider the observability matrix for the pair $(\Lambda, \mathbf{1}'_N)$, given by

$$\begin{bmatrix} \mathbf{1}'_N \\ \mathbf{1}'_N\Lambda \\ \mathbf{1}'_N\Lambda^2 \\ \vdots \\ \mathbf{1}'_N\Lambda^{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \cdots & \lambda_N \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \cdots & \lambda_N^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_1^{N-1} & \lambda_2^{N-1} & \lambda_3^{N-1} & \cdots & \lambda_N^{N-1} \end{bmatrix} ;$$

this is a *Vandermonde matrix* in the parameters $\lambda_1, \lambda_2, \ldots, \lambda_N$ [99]. It is well-known that such matrices are invertible if and only if all of the parameters are distinct, and thus the above observability matrix has rank N over $\mathbb{F}$. This means that the pair $(\mathbf{A}, \mathbf{e}'_{1,N})$ will also be observable. ■

A.2.2 Observability of a Spanning Forest

We will now generalize the discussion in the previous section to the case where the graph of the system is a spanning forest (i.e., it consists of disjoint trees rooted at certain nodes). The following result will later provide us with a way to obtain a tight bound on the observability

index of a given structured matrix pair.

Theorem A.2 *Consider the matrix pair* $(\mathbf{A}, \mathbf{C})$*, where* $\mathbf{A}$ *is an* $N \times N$ *matrix with elements from a field* $\mathbb{F}$*, and* $\mathbf{C}$ *is a* $p \times N$ *matrix of the form* $\mathbf{C} = \begin{bmatrix} \mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_p,N} \end{bmatrix}'$*. Suppose the graph* $\mathcal{H}$ *associated with the matrix* $\mathbf{A}$ *satisfies the following two conditions:*

- *The graph* $\mathcal{H}$ *is a spanning forest rooted at* $\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$*, with self-loops on every node.*
- *No two nodes in the same tree have the same weight on their self-loops, and the weights on the edges between different nodes are equal to* 1*.*

Let D *denote the maximum number of nodes in any tree in* $\mathcal{H}$*. Then, the pair* $(\mathbf{A}, \mathbf{C})$ *is observable with observability index equal to* D*.*

Proof: Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_p$ denote the trees in $\mathcal{H}$, and let r_i denote the number of nodes in tree $\mathcal{T}_i$ (so that $N = r_1 + r_2 + \cdots + r_p$). Since the graph associated with $\mathbf{A}$ is a spanning forest rooted at $\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$, there exists a numbering of the nodes such that the pair $(\mathbf{A}, \mathbf{C})$ has the form

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_3 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_p \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{e}'_{1,r_1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}'_{1,r_2} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{e}'_{1,r_3} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}'_{1,r_p} \end{bmatrix},$$

where the $r_i \times r_i$ matrix $\mathbf{A}_i$ corresponds to the tree $\mathcal{T}_i$. The observability matrix $\mathcal{O}_{D-1}$ for this pair is given by

$$\mathcal{O}_{D-1} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{D-1} \end{bmatrix} = \left[\begin{array}{ccccc} \mathbf{e}'_{1,r_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}'_{1,r_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}'_{1,r_p} \\ \hline \mathbf{e}'_{1,r_1}\mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}'_{1,r_2}\mathbf{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}'_{1,r_p}\mathbf{A}_p \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{e}'_{1,r_1}\mathbf{A}_1^{D-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}'_{1,r_2}\mathbf{A}_2^{D-1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}'_{1,r_p}\mathbf{A}_p^{D-1} \end{array}\right],$$

and if we denote the observability matrix for the pair $(\mathbf{A}_i, \mathbf{e}'_{1,r_i})$ as $\mathcal{O}_{i,D-1}$, it is easy to see that

$$\text{rank}(\mathcal{O}_{D-1}) = \sum_{i=1}^{p} \text{rank}(\mathcal{O}_{i,D-1}) \ .$$

Since each matrix $\mathbf{A}_i$ is a spanning tree rooted at the first node in $\mathbf{A}_i$, and this matrix satisfies the conditions in Theorem A.1, we see that the pair $(\mathbf{A}_i, \mathbf{e}'_{1,r_i})$ will be observable; specifically, the matrix $\mathcal{O}_{i,r_i-1}$ will have rank equal to r_i. Since D is the maximum value of all the r_i's, the above expression for the rank of the observability matrix becomes

$$\text{rank}(\mathcal{O}_{D-1}) = \sum_{i=1}^{p} r_i = N \ ,$$

which concludes the proof of the theorem. ■

A.2.3 Observability of Arbitrary Graphs

So far, we have shown that if the graph associated with $\mathbf{A}$ is a spanning forest rooted at certain nodes, with self-weights that are different elements of a field $\mathbb{F}$ and other edge weights equal to 1, then the pair $(\mathbf{A}, \mathbf{C})$ will be observable. We will now consider matrices $\mathbf{A}$ with arbitrary graphs (not necessarily spanning forests). The following corollary is immediate from the previous section.

Corollary A.1 *Consider the matrix pair $(\mathbf{A}, \mathbf{e}'_{1,N})$, where $\mathbf{A}$ is an $N \times N$ structured matrix (i.e., every entry of $\mathbf{A}$ is either a fixed zero or an independent free parameter from a field $\mathbb{F}$), and $\mathbf{C}$ is a $p \times N$ matrix of the form $\mathbf{C} = \begin{bmatrix} \mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_p,N} \end{bmatrix}'$. Suppose the graph $\mathcal{H}$ associated with the matrix $\mathbf{A}$ contains a path from every node to at least one node in the set $\{x_{i_1}, x_{i_2}, \ldots, x_{i_N}\}$, and furthermore, every node has a self-loop (i.e., the diagonal elements of $\mathbf{A}$ are free parameters). Let $\bar{\mathcal{H}}$ be a subgraph of $\mathcal{H}$ that is a spanning forest rooted at $\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$, with the property that the size of the largest tree is minimal over all possible subgraphs that are spanning forests rooted at those nodes. Let D denote the size of the largest tree in $\bar{\mathcal{H}}$. Then if $\mathbb{F}$ has size at least D, there exists a choice of parameters from $\mathbb{F}$ such that the observability matrix corresponding to the pair $(\mathbf{A}, \mathbf{C})$ has rank N over that field, with observability index equal to D.*

Proof: Consider the spanning forest $\bar{\mathcal{H}}$, and set the values of all parameters corresponding to edges that are not in $\bar{\mathcal{H}}$ to zero, and set the values of all parameters corresponding to edges between different nodes in $\bar{\mathcal{H}}$ to "1". Finally, select the values of the parameters corresponding to self-loops to be such that no two nodes in the same tree of $\bar{\mathcal{H}}$ have the same value (this is possible since the size of the field is at least D). This produces a matrix $\mathbf{A}$ satisfying the conditions in Theorem A.2, and thus the pair $(\mathbf{A}, \mathbf{C})$ is observable with this choice of parameters (with observability index equal to D). ■

The above corollary shows that one can explicitly choose parameters from a field of size D or greater in order to make the pair $(\mathbf{A}, \mathbf{C})$ observable. However, we will also be interested in the case where each nonzero parameter in $\mathbf{A}$ is chosen *randomly* (i.e., independently and uniformly) from field $\mathbb{F}_q$ of size q. In this case, we would like to characterize the *probability* that the pair $(\mathbf{A}, \mathbf{C})$ will be observable, and this is the subject of the following theorem.

Theorem A.3 *Consider the matrix pair $(\mathbf{A}, \mathbf{C})$, where $\mathbf{A}$ is an $N \times N$ structured matrix (i.e., every entry of $\mathbf{A}$ is either a fixed zero or an independent free parameter from the field $\mathbb{F}_q$), and $\mathbf{C}$ is a $p \times N$ matrix of the form $\mathbf{C} = \begin{bmatrix} \mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_p,N} \end{bmatrix}'$. Suppose the graph $\mathcal{H}$ associated with the matrix $\mathbf{A}$ contains a path from every node to at least one node in the set $\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$, and furthermore, every node has a self-loop (i.e., the diagonal elements of $\mathbf{A}$ are free parameters). Let $\bar{\mathcal{H}}$ be a subgraph of $\mathcal{H}$ that is a spanning forest rooted at $\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$, with the property that the size of the largest tree is minimal over all other subgraphs that are spanning forests rooted at those nodes. Let D denote the size of the largest tree in $\bar{\mathcal{H}}$. If each free parameter in $\mathbf{A}$ is chosen randomly (independently and uniformly) from the field $\mathbb{F}_q$ (of size $q \geq (D-1)(N-p+1-\frac{D}{2})$), then with probability at least $1-\frac{1}{q}(D-1)(N-p+1-\frac{D}{2})$, the following two properties will hold: (i) the observability matrix for the pair $(\mathbf{A}, \mathbf{C})$ will have rank N, and (ii) the observability index will be upper bounded by D.*

Proof: Let the free parameters of matrix $\mathbf{A}$ be given by $\lambda_1, \lambda_2, \ldots, \lambda_l \in \mathbb{F}_q$ (note that these λ_i's are no longer used to simply refer to the diagonal entries, as in the proof of Theorem A.1, but to all nonzero entries in the matrix). When convenient, we will aggregate these parameters into a vector $\lambda \in \mathbb{F}_q^l$. With this notation, the matrix $\mathbf{A}$ can also be denoted as $\mathbf{A}(\lambda)$ to explicitly show its dependence on the free parameters. Any particular choice of the free parameters will be denoted by $\lambda^* = \begin{bmatrix} \lambda_1^* & \lambda_2^* & \cdots & \lambda_l^* \end{bmatrix}$, with corresponding numerical matrix $\mathbf{A}(\lambda^*)$.

If the graph of matrix $\mathbf{A}$ satisfies the conditions in the theorem, then we know from Corollary A.1 that there exists a choice of parameters $\lambda^* \in \mathbb{F}_q^l$ (where $q \geq N$) such that the observability matrix

$$\mathcal{O}(\lambda^*)_{D-1} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A}(\lambda^*) \\ \mathbf{C}\mathbf{A}^2(\lambda^*) \\ \vdots \\ \mathbf{C}\mathbf{A}^{D-1}(\lambda^*) \end{bmatrix}$$

has rank N over the field $\mathbb{F}_q$. This means that the observability matrix $\mathcal{O}(\lambda^*)_{D-1}$ contains an $N \times N$ submatrix (denoted by $\mathbf{Z}(\lambda^*)$) whose determinant will be nonzero. Suppose (without loss of generality) that $\mathbf{Z}(\lambda^*)$ is constructed by taking the first N rows of $\mathcal{O}(\lambda^*)_{D-1}$ that form a linearly independent set. We will now derive an expression for the determinant of $\mathbf{Z}(\lambda)$ in terms of the symbolic parameters λ.

First, note that the first p rows of $\mathbf{Z}(\lambda^*)$ are given by the matrix $\mathbf{C}$. Next, recall from Section 1.5 that every set of rows of the form $\mathbf{CA}^k(\lambda^*)$ must increase the rank of the observability matrix by at least one. In the worst case, each such set of rows increases the rank by exactly one, and the last set of rows $\mathbf{CA}^{D-1}(\lambda^*)$ contributes the rest of the rank. Thus, in the worst case, matrix $\mathbf{Z}(\lambda^*)$ has p rows from $\mathbf{C}$, one row from each of $\mathbf{CA}(\lambda^*), \mathbf{CA}^2(\lambda^*), \ldots, \mathbf{CA}^{D-2}(\lambda^*)$, and $N - p - (D-2)$ rows from $\mathbf{CA}^{D-1}(\lambda^*)$. Next, consider the matrix $\mathbf{Z}(\lambda)$ (which is obtained by reverting the special choice of parameters λ^* back to the original symbolic parameters). The determinant of $\mathbf{Z}(\lambda)$ will therefore be a nonzero polynomial in these parameters over the field $\mathbb{F}_q$ (since this polynomial is nonzero after a specific choice of parameters from that field). Specifically, the determinant of an $N \times N$ matrix is a sum of products, where no two entries in any product come from the same row or column of that matrix. Thus, each product in $\det \mathbf{Z}(\lambda)$ will consist of an entry from each row of $\mathbf{Z}(\lambda)$. Now, note that the matrix $\mathbf{CA}^k(\lambda)$ is simply a set of rows from $\mathbf{A}^k(\lambda)$ (from the form of $\mathbf{C}$), and recall that entry (i,j) in $\mathbf{A}^k(\lambda)$ is a polynomial in λ where every term corresponds to the product of weights on a path of length k from node x_j to x_i (i.e., every term is a product of k parameters from λ). Since $\mathbf{Z}(\lambda)$ consists of at least one row from matrices of the form $\mathbf{CA}(\lambda), \mathbf{CA}^2(\lambda), \ldots, \mathbf{CA}^{D-2}(\lambda)$, and at most $N-p-D+2$ rows from $\mathbf{CA}^{D-1}(\lambda)$, we see that each term in $\det \mathbf{Z}(\lambda)$ will be a product of at most

$$1 + 2 + \cdots + D - 2 + (N - p - D + 2)(D-1) = (D-1)\left(N - p - \frac{D}{2} + 1\right)$$

free parameters. Thus, $\det \mathbf{Z}(\lambda)$ is a polynomial of total degree no greater than $(D-1)\left(N - p - \frac{D}{2} + 1\right)$ and as noted earlier, it is not identically zero over the field $\mathbb{F}_q$. If we now choose a set of parameters $(\lambda_1^*, \lambda_2^*, \ldots, \lambda_l^*)$ independently and uniformly from $\mathbb{F}_q^l$, we can apply the Schwartz-Zippel lemma (Lemma A.1) to obtain

$$\Pr[\det \mathcal{O}(\lambda^*) = 0] \leq \frac{(D-1)\left(N - p - \frac{D}{2} + 1\right)}{q},$$

or equivalently,

$$\Pr[\det \mathcal{O}(\lambda^*) \neq 0] \geq 1 - \frac{(D-1)\left(N - p - \frac{D}{2} + 1\right)}{q}.$$

Since this particular choice of observability matrix has full rank after D time-steps, the observability index for this choice of parameters is upper bounded by D, which concludes the proof of the theorem. ■

Remark A.1 *Note that in the worst case, $D = N - (p-1)$ (i.e., the spanning forest consists of $p-1$ isolated vertices, along with a single tree rooted at some vertex). In this case, the observability index meets the upper bound $N - p + 1$ stated in Section 1.5 (where $p = rank(\mathbf{C})$). In general, however, the characterization of the observability matrix provided by the above theorem will be much smaller than this trivial upper bound.*

Remark A.2 *While the above theorem only states that the observability matrix is upper bounded by D, we can also conjecture that the observability index will be exactly equal to D; proving this conjecture appears to be rather difficult, however.*

Now that we have a characterization of the observability of structured systems over finite fields, we turn our attention to the problem of structured invertibility.

A.3 Structural Invertibility over Finite Fields

Consider the linear system

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] \ , \end{aligned} \tag{A.2}$$

where $\mathbf{x} \in \mathbb{F}^N, \mathbf{u} \in \mathbb{F}^m, \mathbf{y} \in \mathbb{F}^p$ for some field $\mathbb{F}$ and $p \geq m$. Furthermore, assume that the matrices $\mathbf{B}$ and $\mathbf{C}$ are of the form

$$\mathbf{B} = \begin{bmatrix} \mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_m,N} \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} \mathbf{e}'_{j_1,N} \\ \mathbf{e}'_{j_2,N} \\ \vdots \\ \mathbf{e}'_{j_p,N} \end{bmatrix}$$

for some sets of indices $\{i_1, i_2, \ldots, i_m\}$ and $\{j_1, j_2, \ldots, j_p\}$; in other words, we assume that $\mathbf{B}$ has a single "1" in each column (and at most one "1" in each row), and $\mathbf{C}$ has a single "1" in each row (and at most one "1" in each column).[2] We assume that $\mathbf{A}$ is an $N \times N$ matrix with entries from the field $\mathbb{F}$. As described in Section A.2, $\mathbf{A}$ is said to be *structured* if every entry of $\mathbf{A}$ is either zero or an independent free parameter (to be chosen from a field $\mathbb{F}$).

The transfer function matrix of system (A.2) is given by $\mathbf{T}(z) = \mathbf{C}(z\mathbf{I}_N - \mathbf{A})^{-1}\mathbf{B}$. When the matrix $\mathbf{A}$ is numerically specified, the system is said to be invertible if $\mathbf{T}(z)$ has rank m over the field of rational functions in z (with coefficients from $\mathbb{F}$). Recall that the matrix pencil of the set $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is given by

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{A} - z\mathbf{I}_N & \mathbf{B} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \ .$$

The rank of the transfer function matrix can be related to the rank of the matrix pencil of the system by using the following lemma from [55]; this lemma was also derived independently

[2] This form corresponds to having each sensor and actuator in the system measuring or affecting only a single state variable. Linear systems of this form commonly occur in practice (e.g., see [41]), and in particular, are applicable to the linear iterative strategies studied in this thesis.

in the context of linear structured systems (with real-valued parameters) in [54].

Lemma A.2 ([54, 55]) *Consider the linear system given by Equation* (A.2)*. For this system,*

$$rank\left(\mathbf{C}(z\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\right) = rank\begin{bmatrix}\mathbf{A}-z\mathbf{I} & \mathbf{B}\\ \mathbf{C} & \mathbf{0}\end{bmatrix} - N \ .$$

Note that the matrix pencil $\mathbf{P}(z)$ has $N+m$ columns, and thus if the transfer function matrix $\mathbf{T}(z)$ is to have rank m over the field of rational functions in z (i.e., if it is of full column rank), then the matrix pencil $\mathbf{P}(z)$ must be of rank $N+m$ over that field (i.e., it must also be of full column rank). Note that this lemma applies regardless of the field $\mathbb{F}$.

Our analysis of structured system invertibility over finite fields will be based on a graph representation of the matrix set $(\mathbf{A}, \mathbf{B}, \mathbf{C})$, denoted by $\mathcal{H}$, which we obtain as follows. The vertex set of $\mathcal{H}$ is $\mathcal{X} \cup \mathcal{U} \cup \mathcal{Y}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is a set of state vertices, $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ is a set of input vertices, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ is a set of output vertices. The edge set of $\mathcal{H}$ is given by $\mathcal{E} = \mathcal{E}_{xx} \cup \mathcal{E}_{ux} \cup \mathcal{E}_{xy}$, where

- $\mathcal{E}_{xx} = \{(x_j, x_i) | \mathbf{A}_{ij} \neq 0\}$ is the set of edges corresponding to interconnections between the state vertices,
- $\mathcal{E}_{ux} = \{(u_j, x_i) | \mathbf{B}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the state vertices, and
- $\mathcal{E}_{xy} = \{(x_j, y_i) | \mathbf{C}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the state vertices and the output vertices.

The weight on edge (x_j, x_i) is set to the value of $\mathbf{A}_{ij}$ (this can be a free parameter if $\mathbf{A}$ is a structured matrix). Note that if $\mathbf{A}$ is the weight matrix for a linear iteration, $\mathcal{H}$ is simply the graph of the network $\mathcal{G}$, augmented with self-loops on every node and with a set of input and output vertices (each with a single outgoing or incoming edge to a state vertex). We will now provide a graph-based analysis of invertibility over finite fields; our development will be similar to the derivation in [55], which considered the probability of a linear network code being feasible when the coefficients are chosen at random from a finite field. However, the bounds that we obtain in our derivation will be slightly different than the bounds in [55], due to differences in the forms of the systems that are considered.

A.3.1 Invertibility of Systems Whose Graphs Are Linkings

We start by considering a system $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ whose graph $\mathcal{H}$ is an m-linking from $\mathcal{U}$ to $\mathcal{Y}$ (through some state vertices) along with a set of isolated state vertices. We will show that such systems are invertible regardless of the choice of field.

Theorem A.4 *Consider the matrix set* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$*, where the entries in the* $N \times N$ *matrix* $\mathbf{A}$ *are elements of a field* $\mathbb{F}$*, the* $N \times m$ *matrix* $\mathbf{B}$ *has a single "*1*" in each column (and at most one "*1*" in each row), and the* $p \times N$ *matrix* $\mathbf{C}$ *has a single "*1*" in each row (and at most one "*1*" in each column). Suppose that the following two conditions hold:*

- *The graph* $\mathcal{H}$ *associated with* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *is a disjoint union of an* m*-linking from the inputs to the outputs and a set of isolated state vertices (with the exception of possible outgoing edges to output vertices).*
- *The weights on the edges involved in the linking are all equal to "*1*", and the weights on the self-loops are all equal to zero.*

Then the system given by $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *is invertible over the field* $\mathbb{F}$*.*

Proof: Consider the m disjoint paths that constitute the linking from the inputs to the outputs, and let r_i denote the number of state vertices that are involved in the path from input vertex u_i to output vertex y_i (for $i = 1, 2, \ldots, m$). Note that the inputs and outputs can be renumbered accordingly for this to be possible. We will let $r = r_1 + r_2 + \ldots + r_m$. Without loss of generality, we can assume that the matrix pencil for the set $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ has the form

$$\mathbf{P}(z) = \left[\begin{array}{c|c} \mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{0} \end{array}\right]$$

$$= \left[\begin{array}{ccccc|cccc} -z\mathbf{I}_{N-r} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_1(z) & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{e}_{r_1,r_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{J}_2(z) & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{e}_{r_2,r_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{J}_m(z) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}_{r_m,r_m} \\ \hline \bar{\mathbf{C}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}'_{1,r_1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{e}'_{1,r_2} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}'_{1,r_m} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{array}\right],$$

where the $r_i \times r_i$ matrix $\mathbf{J}_i(z)$ corresponds to the state vertices involved in the i–th path in the linking and has the form

$$\mathbf{J}_i(z) = \begin{bmatrix} -z & 1 & 0 & 0 & \cdots & 0 \\ 0 & -z & 1 & 0 & \cdots & 0 \\ 0 & 0 & -z & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & \cdots & -z \end{bmatrix} .$$

The matrix $-z\mathbf{I}_{N-r}$ corresponds to the isolated vertices in $\mathcal{H}$, and the matrix $\bar{\mathbf{C}}$ represents the portion of $\mathbf{C}$ associated with those vertices. The above form for the matrix pencil can always be obtained by a simple permutation of the state vertices, which does not change the rank of the pencil (or the transfer function matrix). The rows and columns of this matrix pencil can further be rearranged (without affecting the rank) to produce a matrix of the form

$$\begin{bmatrix} -z\mathbf{I}_{N-r} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_1(z) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_2(z) & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_m(z) \\ \bar{\mathbf{C}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} , \tag{A.3}$$

where

$$\mathbf{P}_i(z) = \begin{bmatrix} \mathbf{J}_i(z) & \mathbf{e}_{r_i,r_i} \\ \mathbf{e}'_{1,r_i} & 0 \end{bmatrix} .$$

Based on the form of matrix $\mathbf{J}_i(z)$ given above, one can readily verify that $\mathbf{P}_i(z)$ is non-singular over the field of rational functions in z (with coefficients from $\mathbb{F}$); in fact, the determinant of $\mathbf{P}_i(z)$ is equal to 1. Since each $\mathbf{P}_i(z)$ is an $(r_i + 1) \times (r_i + 1)$ matrix, we see that the last $r + m$ columns of the matrix (A.3) will be linearly independent. Finally, note that the first $N - r$ columns of this matrix are also guaranteed to be linearly independent over the field of rational functions in z due to the matrix $-z\mathbf{I}_{N-r}$. Thus, matrix (A.3) has rank $N + m$ over the field of rational functions in z, which means that the matrix pencil will also have rank $N + m$. From Lemma A.2, we see that the system is invertible. ■

A.3.2 Invertibility of Arbitrary Graphs

So far, we have shown that if the graph of the system $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ consists of an m-linking from the inputs to the outputs and a set of isolated vertices, then the system will be invertible (regardless of the choice of field $\mathbb{F}$). In this section, we extend this result to systems that

have more general graphs. We start with the following corollary, which is immediate from the previous discussion.

Corollary A.2 *Consider the matrix set* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$*, where* $\mathbf{A}$ *is a structured matrix (i.e., every entry of* $\mathbf{A}$ *is either a fixed zero or an independent free parameter to be chosen from an arbitrary field* $\mathbb{F}$*), the* $N \times m$ *matrix* $\mathbf{B}$ *has a single "*1*" in each column (and at most one "*1*" in each row), and the* $p \times N$ *matrix* $\mathbf{C}$ *has a single "*1*" in each row (and at most one "*1*" in each column). Suppose that graph* $\mathcal{H}$ *associated with* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *contains an* m*-linking from the inputs to the outputs. Then there exists a choice of free parameters from* $\mathbb{F}$ *so that the system is invertible.*

Proof: Since there is an m-linking from the input vertices to the output vertices in $\mathcal{H}$, it is possible to find a subgraph of $\mathcal{H}$ that contains only those vertices and edges that are involved in the linking. Now, set the values of all parameters corresponding to edges that are not in this subgraph to zero, and set the values of all parameters corresponding to the edges in the linking to 1. Finally, set all of the parameters corresponding to the self-loops to be zero. This produces a matrix $\mathbf{A}$ satisfying the conditions in Theorem A.4, and thus the system $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is invertible with this choice of parameters. ■

The above corollary shows that one can explicitly choose parameters from a field of arbitrary size in order to make the system $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ invertible. However, as in Section A.2, we will also be interested in the case where each nonzero parameter in $\mathbf{A}$ is chosen *randomly* (i.e., independently and uniformly) from the field $\mathbb{F}_q$ of size q. In this case, we would like to characterize the *probability* that the system $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ will be observable, and this is the subject of the following theorem.

Theorem A.5 *Consider the matrix set* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$*, where* $\mathbf{A}$ *is a structured matrix (i.e., every entry of* $\mathbf{A}$ *is either a fixed zero or an independent free parameter to be chosen from the field* $\mathbb{F}_q$ *of size* $q \geq N - m$*), the* $N \times m$ *matrix* $\mathbf{B}$ *has a single "*1*" in each column (and at most one "*1*" in each row), and the* $p \times N$ *matrix* $\mathbf{C}$ *has a single "*1*" in each row (and at most one "*1*" in each column). Suppose that the graph* $\mathcal{H}$ *associated with* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *contains an* m*-linking from the inputs to the outputs. Then, if each free parameter in* $\mathbf{A}$ *is chosen randomly (independently and uniformly) from the field* $\mathbb{F}_q$*, the probability that the system* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *will be invertible is at least* $1 - \frac{N-m}{q}$*.*

Proof: Let the free parameters of matrix $\mathbf{A}$ be given by $\lambda_1, \lambda_2, \ldots, \lambda_l \in \mathbb{F}_q$. When convenient, we will aggregate these parameters into a vector $\lambda \in \mathbb{F}_q^l$. With this notation, the matrix $\mathbf{A}$ can also be denoted as $\mathbf{A}(\lambda)$ to explicitly show its dependence on the free parameters. Any particular choice of the free parameters will be denoted by $\lambda^* = \begin{bmatrix} \lambda_1^* & \lambda_2^* & \cdots & \lambda_l^* \end{bmatrix}$, with corresponding numerical matrix $\mathbf{A}(\lambda^*)$. The matrix pencil corresponding to these parameters will be denoted by $\mathbf{P}(\lambda, z)$.

Next, note from Corollary A.2 that if the graph $\mathcal{H}$ contains an m-linking from the inputs to the outputs, then there exists a choice of free parameters $\lambda^* \in \mathbb{F}_q^l$ so that the matrix pencil $\mathbf{P}(\lambda^*, z)$ will have rank $N+m$. This means that there exists an $(N+m) \times (N+m)$ submatrix of $\mathbf{P}(\lambda^*, z)$ that is invertible; denote this submatrix by $\mathbf{Z}(\lambda^*, z)$, and note that $\det \mathbf{Z}(\lambda, z)$ will be a polynomial in λ and z that is not identically zero. To obtain a more careful characterization of this polynomial, note from the structure of $\mathbf{P}(\lambda, z)$ that the last m columns of $\mathbf{Z}(\lambda, z)$ must each have a single "1" in some entry, and zeros elsewhere. Since the determinant of a matrix is a sum of products of the entries in the matrix, where no two entries in any product are from the same row or column, we see that the rows corresponding to those 1's do not contribute any parameters to $\det \mathbf{Z}(\lambda, z)$. Thus, there are at most $N-m$ rows of $\mathbf{Z}(\lambda, z)$ that can contribute free parameters to the determinant. We can thus write

$$\det \mathbf{Z}(\lambda, z) = \sum_{\tau_{i_1}+\tau_{i_2}+\ldots+\tau_{i_l}+\tau_{i_z} \leq N-m} c_{i_1,i_2,\ldots,i_l,i_z} \lambda_1^{\tau_{i_1}} \lambda_2^{\tau_{i_2}} \ldots \lambda_l^{\tau_{i_l}} z^{\tau_{i_z}} \quad ,$$

where the exponents on the free parameters are either 0 or 1, the exponent on z is nonnegative, and $c_{i_1,\ldots,i_l,i_z} \in \{-1, 0, 1\}$ for each valid choice of exponents. We can equivalently write this as a polynomial in z, with coefficients that are polynomials in the free parameters λ, and since the determinant is not identically zero, at least one of these coefficient polynomials is not identically zero. Denote any one of these nonzero coefficient polynomials by $c(\lambda)$, and note that the total degree of this polynomial will be at most $N-m$ (since it will contain at most $N-m$ free parameters). If $c(\lambda^*)$ is nonzero for some choice of free parameters $\lambda^* \in \mathcal{F}_q^l$, then the determinant of $\mathbf{Z}(\lambda^*, z)$ will also be nonzero, and thus the matrix pencil $\mathbf{P}(\lambda^*, z)$ will have rank $N+m$. To calculate the probability of $c(\lambda^*)$ being nonzero for a random choice of parameters, we use the Schwartz-Zippel lemma (Lemma A.1) to obtain

$$\Pr[c(\lambda^*) = 0] \leq \frac{N-m}{q};$$

since

$$\Pr[\text{rank}(\mathbf{P}(\lambda^*, z)) < N+m] \leq \Pr[\det \mathbf{Z}(\lambda^*, z) = 0] \leq \Pr[c(\lambda^*) = 0],$$

we obtain

$$\Pr[\text{rank}(\mathbf{P}(\lambda^*, z)) = N+m] \geq 1 - \frac{N-m}{q} \quad .$$

■

REFERENCES

[1] S. D. Baker and D. H. Hoglund, "Medical-grade, mission-critical wireless networks [Designing an enterprise mobility solution in the healthcare environment]," *IEEE Engineering in Medicine and Biology Magazine*, vol. 27, no. 2, pp. 86–95, Mar. 2008.

[2] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick, "An overview of emerging results in cooperative UAV control," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004, pp. 602–607.

[3] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[4] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 20–27.

[5] C. G. Cassandras and W. Li, "Sensor networks and cooperative control," *European Journal of Control*, vol. 11, no. 4–5, pp. 436–463, 2005.

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

[7] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the American Control Conference*, 2005, pp. 1859–1864.

[8] M. Effros, R. Koetter, and M. Médard, "Breaking network logjams," *Scientific American*, pp. 78–85, June 2007.

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

[10] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks*. Berlin, Germany: Springer-Verlag, 2005.

[11] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1996.

[12] D. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.

[13] E. Kushilevitz and N. Nisan, *Communication Complexity*. Cambridge, UK: Cambridge University Press, 2006.

[14] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, Apr. 2006.

[15] P. Barooah and J. Hespanha, "Estimation on graphs from relative measurements," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 57–74, Aug. 2007.

[16] M. Dietzfelbinger, "Gossiping and broadcasting versus computing functions in networks," *Discrete Applied Mathematics*, vol. 137, no. 2, pp. 127–153, Mar. 2004.

[17] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2006, pp. 113–122.

[18] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient aggregation for sensor networks," in *Proceedings of the 5th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2006, pp. 69–76.

[19] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2486–2507, June 2006.

[20] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[21] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, 2005, pp. 2996–3000.

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

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

[24] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[25] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, Feb. 2005.

[26] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, July 2007.

[27] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.

[28] D. B. West, *Introduction to Graph Theory.* Upper Saddle River, NJ: Prentice-Hall Inc., 2001.

[29] A. Gibbons, *Algorithmic Graph Theory.* Cambridge, UK: Cambridge University Press, 1985.

[30] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, Mar. 1974.

[31] V. Bhandari and N. H. Vaidya, "Reliable broadcast in wireless networks with probabilistic failures," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM)*, 2007, pp. 715–723.

[32] V. Bhandari and N. H. Vaidya, "On reliable broadcast in a radio network," in *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2005, pp. 138–147.

[33] A. Pelc and D. Peleg, "Broadcasting with locally bounded Byzantine faults," *Information Processing Letters*, vol. 93, no. 3, pp. 109–115, Feb. 2005.

[34] S. Jaggi, M. Langberg, and M. Effros, "Correction of adverserial errors in networks," in *Proceedings of the International Symposium on Information Theory (ISIT)*, 2005, pp. 1455–1459.

[35] D. Silva, F. R. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," in *Proceedings of the 2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks*, 2007, pp. 1–5.

[36] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Karger, "Byzantine modification detection in multicast networks using random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2798–2803, 2008.

[37] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros, "Resilient network coding in the presence of Byzantine adverseries," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2596–2603, June 2008.

[38] L. Nutman and M. Langberg, "Adverserial models and resilient schemes for network coding," in *Proceedings of the IEEE International Symposium on Information Theory*, 2008, pp. 171–175.

[39] M. J. Siavoshani, C. Fragouli, and S. Diggavi, "On locating Byzantine attackers," in *Proceedings of the Fourth Workshop on Network Coding, Theory and Applications (NetCod)*, 2008, pp. 1–6.

[40] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3579–3591, Aug. 2008.

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

[42] M. L. J. Hautus, "Strong detectability and observers," *Linear Algebra and its Applications*, vol. 50, pp. 353–368, Apr. 1983.

[43] M. K. Sain and J. L. Massey, "Invertibility of linear time-invariant dynamical systems," *IEEE Transactions on Automatic Control*, vol. AC-14, no. 2, pp. 141–149, Apr. 1969.

[44] A. S. Willsky, "On the invertibility of linear systems," *IEEE Transactions on Automatic Control*, vol. 19, no. 2, pp. 272–274, June 1974.

[45] D. Rappaport and L. M. Silverman, "Structure and stability of discrete-time optimal systems," *IEEE Transactions on Automatic Control*, vol. 16, no. 3, pp. 227–233, June 1971.

[46] W. Kratz, "Characterization of strong observability and construction of an observer," *Linear Algebra and its Applications*, vol. 221, pp. 31–40, May 1995.

[47] J. van der Woude, "The generic number of invariant zeros of a structured linear system," *SIAM Journal on Control and Optimization*, vol. 38, no. 1, pp. 1–21, Nov. 1999.

[48] L. M. Silverman, "Discrete Riccati equations: Alternative algorithms, asymptotic properties and system theory interpretations," in *Control and Dynamic Systems*, C. T. Leondes, Ed. New York, NY: Academic Press, 1976, vol. 12, pp. 313–386.

[49] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2003.

[50] C.-T. Lin, "Structural controllability," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 201–208, June 1974.

[51] S. Hosoe, "Determination of generic dimensions of controllable subspaces and its application," *IEEE Transactions on Automatic Control*, vol. 25, no. 6, pp. 1192–1196, Dec. 1980.

[52] K. J. Reinschke, *Multivariable Control A Graph-Theoretic Approach*. Berlin, Germany: Springer-Verlag, 1988.

[53] J.-M. Dion, C. Commault, and J. van der Woude, "Generic properties and control of linear structured systems: a survey," *Automatica*, vol. 39, no. 7, pp. 1125–1144, July 2003.

[54] J. W. van der Woude, "A graph-theoretic characterization for the rank of the transfer matrix of a structured system," *Mathematics of Control, Signals and Systems*, vol. 4, no. 1, pp. 33–40, Mar. 1991.

[55] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of the 41st Allerton Conference on Communications, Control and Computing*, 2003.

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

[57] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, Nov. 2006.

[58] M. E. Yildiz and A. Scaglione, "Differential nested lattice encoding for consensus problems," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 89–98.

[59] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, Oct. 2008.

[60] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," in *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008, pp. 4825–4830.

[61] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri, "Average consensus by gossip algorithms with quantized communication," in *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008, pp. 4837–4842.

[62] D. Mosk-Aoyama and D. Shah, "Information dissemination via network coding," in *Proceedings of the 2006 IEEE International Symposium on Information Theory*, 2006, pp. 1748–1752.

[63] M. Huang and J. H. Manton, "Stochastic double array analysis and convergence of consensus algorithms with noisy measurements," in *Proceedings of the American Control Conference*, 2007, pp. 705–710.

[64] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, Jan. 2007.

[65] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links – Part I: distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[66] H. V. Poor, *An Introduction to Signal Detection and Estimation.* Berlin, Germany: Springer-Verlag, 1994.

[67] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Englewood Cliffs, NJ: Prentice-Hall Inc., 1979.

[68] C. N. Hadjicostis, *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems.* Boston, MA: Kluwer Academic Publishers, 2002.

[69] S. E. Haupt, G. S. Young, K. J. Long, and A. Beyer, "Data requirements from evolvable sensor networks for homeland security problems," in *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems*, 2007, pp. 58–66.

[70] K. Sampigethaya, M. Li, R. Poovendran, R. Robinson, L. Bushnell, and S. Lintelman, "Secure wireless collection and distribution of commercial airplane health data," in *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference*, 2007, pp. 4.E.6–1–4.E.6.8.

[71] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *Journal of the Association for Computing Machinery*, vol. 40, no. 1, pp. 17–47, Jan. 1993.

[72] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, July 1982.

[73] D. Dolev, "The Byzantine Generals strike again," *Journal of Algorithms*, vol. 3, no. 1, pp. 14–30, Mar. 1982.

[74] V. Gupta, C. Langbort, and R. M. Murray, "On the robustness of distributed algorithms," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 3473–3478.

[75] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007, pp. 5594–5599.

[76] V. Barnett and T. Lewis, *Outliers in Statistical Data*. West Sussex, England: John Wiley and Sons Ltd., 1996.

[77] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets," *Data Mining and Knowledge Discovery*, vol. 12, no. 2-3, pp. 203–228, 2006.

[78] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, 2006, pp. 51–58.

[79] J. Feigenbaum and S. Shenker, "Distributed algorithmic mechanism design: Recent results and future directions," in *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2002, pp. 1–13.

[80] D. Lucking-Reiley, "Vickrey auctions in practice: From nineteenth-century philately to twenty-first-century e-commerce," *Journal of Economic Perspectives*, vol. 14, no. 3, pp. 183–192, 2000.

[81] G. Takos and C. N. Hadjicostis, "Error correction in DFT codes subject to low-level quantization noise," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1043–1054, Mar. 2008.

[82] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, "Network coding theory," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 4/5, pp. 241–381, 2005.

[83] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[84] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[85] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," in *Proceedings of the 2004 IEEE International Conference on Communications*, 2004, pp. 661–666.

[86] E. Erez and M. Feder, "Convolutional network codes," in *Proceedings of the 2004 IEEE International Symposium on Information Theory*, 2004, p. 146.

[87] R. W. Yeung and N. Cai, "Network error correction, part I: Basic concepts and upper bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 19–36, 2006.

[88] N. Cai and R. W. Yeung, "Network error correction, part II: Lower bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 37–54, 2006.

[89] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *Proceedings of the 2005 Conference on Information Sciences and Systems*, 2005, pp. 113–122.

[90] E. Erez and M. Feder, "Efficient network codes for cyclic networks," in *Proceedings of the 2005 International Symposium on Information Theory*, 2005, pp. 1982–1986.

[91] T. Ho, B. Leong, Y.-H. Chang, Y. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," in *Proceedings of the 2005 International Symposium on Information Theory*, 2005, pp. 1977–1981.

[92] L. M. Silverman, "Inversion of multivariable linear systems," *IEEE Transactions on Automatic Control*, vol. AC-14, no. 3, pp. 270–276, June 1969.

[93] F.-M. Yuan, "Minimal dimension inverses of linear sequential circuits," *IEEE Transactions on Automatic Control*, vol. AC-20, no. 1, pp. 42–52, Feb. 1975.

[94] E. Emre and L. Silverman, "Minimal dynamic inverses for linear systems with arbitrary initial states," *IEEE Transactions on Automatic Control*, vol. AC-21, no. 5, pp. 766–769, Oct. 1976.

[95] T. Yoshikawa and S. P. Bhattacharyya, "Partial uniqueness: Observability and input identifiability," *IEEE Transactions on Automatic Control*, vol. 20, no. 5, pp. 713–714, October 1975.

[96] T. Yoshikawa and T. Sugie, "Inverse systems for reproducing linear functions of inputs," *Automatica*, vol. 17, no. 5, pp. 763–769, 1981.

[97] S. Sundaram and C. N. Hadjicostis, "Partial state observers for linear systems with unknown inputs," *Automatica*, vol. 44, no. 12, pp. 3126–3132, Dec. 2008.

[98] D. C. Kozen, *Theory of Computation*. London, UK: Springer-Verlag London Ltd., 2006.

[99] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge, UK: Cambridge University Press, 2003.

AUTHOR'S BIOGRAPHY

Shreyas Sundaram is a candidate for the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign. He received the BASc. degree in computer engineering from the University of Waterloo in 2003, and the M.S. degree in electrical engineering from the University of Illinois at Urbana-Champaign in 2005. His research interests lie in the areas of secure and fault-tolerant control of large-scale distributed systems and networks, linear system and estimation theory, and the application of algebraic graph theory to system analysis. He received the M. E. Van Valkenburg Graduate Research Award and the Robert T. Chien Memorial Award from the ECE department at the University of Illinois at Urbana-Champaign for excellence in research. He also received the E. A. Reid Fellowship Award from the ECE department and the Mavis Memorial Fund Scholarship from the College of Engineering, both in recognition of commitment to engineering education. He was a finalist for the Best Student Paper Award at the 2007 and 2008 American Control Conferences.