

論文 / 著書情報  
Article / Book Information

題目(和文)	
Title(English)	Resilient Consensus of Multi-agent Systems via Multi-hop Communication
著者(和文)	YuanLiwei
Author(English)	Liwei Yuan
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第12264号, 授与年月日:2022年9月22日, 学位の種別:課程博士, 審査員:石井 秀明,三宅 美博,DEFAGO XAVIER,小野 功,小野 峻佑
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第12264号, Conferred date:2022/9/22, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

# Resilient Consensus of Multi-agent Systems via Multi-hop Communication



Liwei Yuan

Graduate Major in Artificial Intelligence  
Department of Computer Science  
Tokyo Institute of Technology

*Supervisor*

Prof. Hideaki Ishii

In partial fulfillment of the requirements for the degree of  
*Doctor of Philosophy in Computer Science*

July 23, 2022



## **Acknowledgements**

I am grateful to all of those people with whom I have had the pleasure of working during my time at Tokyo Tech. I would like to especially thank my adviser Prof. Hideaki Ishii for his insightful advices and patient guidance with the whole work. I am grateful as well to the members of his research group, and in particular, Yuan Wang, Xin Wang, and Shuai Feng for their sincere guidance. Most importantly, I am indebted to my parents, whose love and support guide me and motivate me to accomplish my PhD program in Japan. Last but not least, I want to mention the support and love from my girlfriend You Wang. In the company of her, I was able to advance in these difficult times during the Pandemic.

## Abstract

In this thesis, we consider a multi-agent resilient consensus problem, where some of the nodes are adversarial and transmit false data to the normal agents. As the applications of multi-agent systems widely used in the areas of wireless sensor networks, Internet of Things and distributed computing, the resiliency of the multi-agent systems needs to be improved. The goal of the resilient consensus problem is that normal agents should reach a common state value regardless the misbehaviors taken by the adversary agents. Generally, there are two main approaches in this area: (i) mean subsequence reduced (MSR) algorithms, and (ii) detection-based algorithms. We develop several algorithms tackling this problem under different threat models. All of our approaches require the nodes to have the abilities to communicate with their multi-hop neighbors. Such multi-hop communication techniques are widely used in the wireless communication field as well as the computer science field. Yet, only a few works studied the application of the multi-hop techniques in the resilient consensus problem. One remarkable advantage of applying such techniques is that the requirement on the network connectivity can be reduced compared to the conventional one-hop algorithms.

In the literature, the adversarial agents are categorized into basically two types: Malicious agents and Byzantine agents. These agents are capable to manipulate their data arbitrarily. Malicious agents are limited as they must broadcast the same messages to all of their neighbors, while Byzantine agents can send individual messages to different neighbors.

First, we develop an MSR-based algorithm with multi-hop communication. We analyze the performance of the proposed algorithm under different threat models (i.e., malicious or Byzantine model). Moreover, we generalize the notion of graph robustness to the multi-hop setting, which is the necessary graph structure for the proposed algorithm to achieve resilient

consensus. Taking account of that the delays are critical in the multi-hop communication environment, we also provide the analysis of the proposed algorithm when the agents update asynchronously and the communication is subject to time delays. This has broadened the scope of our algorithms. So far, we only consider the real-valued consensus where agents take real values. Note that the integer-valued consensus is also widely used in the sensor networks especially when the sensors only have limited registers to store the integer values. Another advantage of taking integer values is that the communication resources can be saved compared to the real-valued case. Such integer-valued consensus is also considered for our algorithms.

Second, we develop an event-triggered algorithm based on the abovementioned algorithm. The event-triggered scheme is well known in the systems control area. Notably, it can reduce the communication loads while guaranteeing the given network control system to be stable. By introducing the event-triggered scheme to our algorithm, the transmissions between agents are heavily reduced compared to the non-event-triggered algorithms. We also highlight that through multi-hop communication, the necessary network connectivity for achieving resilient consensus can be reduced especially in comparison with the conventional one-hop communication case.

Lastly, we develop a detection-based resilient consensus algorithm where the two-hop communication is utilized. It is shown that the detection scheme becomes effective by requiring certain connectivity properties in the network so that the non-adversarial nodes can share enough information about their neighbors. In the second detection scheme, majority voting is used for normal agents to determine the identities of their neighbors (normal or adversarial). The detection function is realized by the verification of two consecutive information sets sent by the same neighbor. The proposed algorithm is more distributed and requires no additional authorized or centralized verification procedures compared to the existing detection method.

At the end of each chapter, numerical examples are provided to demonstrate the efficiency of the proposed methods.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivations . . . . .	1
1.2	Context of the Study . . . . .	4
1.3	Main Contributions . . . . .	6
1.4	Overview of the Thesis . . . . .	9
<b>2</b>	<b>Preliminaries and Related Works</b>	<b>13</b>
2.1	Fundamentals on Graph Theory . . . . .	13
2.1.1	Connectivity . . . . .	14
2.1.2	Algebraic Graph Theory . . . . .	14
2.2	Resilient Asymptotic Consensus . . . . .	16
2.2.1	Graph Robustness . . . . .	17
2.2.2	Resiliency Notions and Threat Models . . . . .	18
2.2.3	Resilient Consensus . . . . .	20
2.2.4	Weighted MSR Algorithm . . . . .	21
2.3	Consensus with Multi-hop Communication . . . . .	22
2.3.1	Multi-hop Communication Model . . . . .	23
2.3.2	Discussion on Manipulation in Message Path Information . . . . .	24
2.4	Quantized Consensus . . . . .	26
2.5	Event-based Communication . . . . .	27
2.6	Fault Detection and Identification . . . . .	28
2.6.1	Distributed Fault Detection . . . . .	29
2.6.2	Voting Algorithms . . . . .	30

<b>3 Resilient Consensus under Malicious Attacks with Multi-hop Communication</b>	<b>31</b>
3.1 Problem Formulation . . . . .	32
3.1.1 Multi-hop Communication for Multi-agent Consensus . . . . .	32
3.1.2 Threat Model . . . . .	33
3.1.3 Resilient Asymptotic Consensus . . . . .	33
3.2 Multi-hop Weighted MSR Algorithm . . . . .	34
3.3 Robustness with Multi-hop Communication . . . . .	37
3.4 Synchronous Network . . . . .	40
3.4.1 Matrix Representation . . . . .	40
3.4.2 Consensus Analysis with Multi-hop Communication . . . . .	41
3.5 Asynchronous Network . . . . .	44
3.6 Discussions on Graph Robustness with Multi-hop Communication . . . . .	50
3.6.1 Properties of Robustness with $l$ Hops . . . . .	50
3.6.2 The Case of Unbounded Path Length . . . . .	54
3.7 Numerical Examples . . . . .	57
3.7.1 Synchronous MW-MSR Algorithm . . . . .	57
3.7.2 Asynchronous MW-MSR Algorithm . . . . .	58
3.7.3 Simulations in Large Wireless Sensor Networks . . . . .	59
3.8 Summary . . . . .	61
<b>4 Asynchronous Approximate Byzantine Consensus via Multi-hop Communication</b>	<b>64</b>
4.1 Problem Formulation . . . . .	65
4.1.1 Update Rule . . . . .	65
4.1.2 Threat Models . . . . .	66
4.1.3 Resilient Asymptotic Consensus and Algorithm . . . . .	66
4.2 Graph Robustness with Multi-hop Communication . . . . .	67
4.2.1 The Notion of $(r, s)$ -Robustness with $l$ Hops . . . . .	67
4.2.2 The Notion of $r$ -Strongly Robust Graphs with $l$ Hops . . . . .	68
4.3 Synchronous Byzantine Consensus . . . . .	69
4.3.1 Consensus Analysis . . . . .	69
4.3.2 Discussions on Different Graph Conditions . . . . .	72
4.4 Asynchronous Byzantine Consensus . . . . .	73



4.4.1	Discussions on Asynchrony . . . . .	73
4.4.2	Consensus Analysis . . . . .	74
4.4.3	Comparison with Conventional Methods . . . . .	80
4.4.3.1	Advantages in Threat Models and Graph Conditions . . . . .	80
4.4.3.2	Advantages in Computational Complexity . . . . .	81
4.5	Numerical Example . . . . .	83
4.6	Summary . . . . .	84
<b>5</b>	<b>Resilient Quantized Consensus with Multi-hop Communication</b>	<b>85</b>
5.1	Problem Formulation . . . . .	86
5.1.1	Quantized Consensus and Update Rule . . . . .	86
5.1.2	Threat Model . . . . .	87
5.1.3	Resilient Quantized Consensus and Algorithm . . . . .	88
5.2	Synchronous Networks . . . . .	88
5.2.1	Matrix Representation . . . . .	89
5.2.2	Consensus Analysis . . . . .	90
5.3	Asynchronous Network . . . . .	93
5.3.1	Randomized Updates versus Deterministic Updates . . . . .	93
5.3.2	Asynchronous Updates with Delays . . . . .	95
5.4	Summary . . . . .	101
<b>6</b>	<b>Event-triggered Approximate Byzantine Consensus with Multi-hop Communication</b>	<b>102</b>
6.1	Problem Formulation . . . . .	103
6.1.1	Update Rule . . . . .	103
6.1.2	Threat Model . . . . .	105
6.1.3	Resilient Asymptotic Consensus . . . . .	105
6.2	Event-triggered Algorithm Design . . . . .	106
6.2.1	Asynchronous Event-triggered MW-MSR algorithm . . . . .	106
6.3	Consensus Analysis . . . . .	106
6.4	An Alternative Event-triggered Scheme . . . . .	114
6.5	Numerical Examples . . . . .	120
6.5.1	Topology Gap between One-hop and Multi-hop Algorithms . . . . .	120
6.5.2	The Amount of Transmissions of Different Algorithms . . . . .	121

---

6.6	Summary . . . . .	122
<b>7</b>	<b>Secure Consensus with Distributed Detection via Two-hop Communication</b>	<b>124</b>
7.1	Problem Formulation . . . . .	125
7.1.1	Update Rule and Threat Model . . . . .	125
7.1.2	Information Set for Two-hop Communication . . . . .	127
7.2	Scheme 1 with Detection Share . . . . .	130
7.2.1	Resilient Consensus Scheme 1 . . . . .	130
7.2.2	Detection Algorithm Design . . . . .	132
7.2.3	Necessary Graph Structure for Scheme 1 . . . . .	133
7.3	Scheme 2 with Fully Distributed Detection . . . . .	135
7.3.1	Resilient Consensus Scheme 2 . . . . .	135
7.3.2	Detection Algorithm Design . . . . .	136
7.3.3	Necessary Graph Structure for Scheme 2 . . . . .	137
7.3.4	Discussion . . . . .	140
7.4	Numerical Examples . . . . .	143
7.4.1	Resilient Consensus under Scheme 1 . . . . .	143
7.4.2	Resilient Consensus under Scheme 2 . . . . .	144
7.4.3	Application to Large Wireless Sensor Networks . . . . .	146
7.5	Summary . . . . .	148
<b>8</b>	<b>Conclusion</b>	<b>150</b>
8.1	Summary of Contributions . . . . .	150
8.2	Future Works . . . . .	151
	<b>References</b>	<b>166</b>

# List of Figures

2.1	The graph is $(2, 2)$ -robust. . . . .	17
3.1	(a) A 5-node graph. (b) Values removed by node 1 in one-hop and two-hop algorithms. . . . .	36
3.2	(a) Node $i$ has two independent paths originating from the outside of $\mathcal{V}_1$ with respect to set $\mathcal{F} = \{j\}$ . (b) Node $i$ only has one independent path sharing the same property. . . . .	38
3.3	(a) The graph is not $(2, 2)$ -robust with one hop, but it is $(2, 2)$ -robust with 2 hops. (b) The graph is $(2, 2)$ -robust with one hop and $(3, 3)$ -robust with 2 hops. . . . .	39
3.4	The graph is not 2-robust with one hop, but it is 3-robust with 2 hops and $(2, 2)$ -robust with 2 hops. . . . .	50
3.5	Illustration for cycle graphs. . . . .	56
3.6	Time responses of the synchronous one-hop W-MSR algorithm. . . . .	57
3.7	Time responses of the synchronous two-hop MW-MSR algorithm. . . . .	57
3.8	Time responses of the synchronous one-hop W-MSR algorithm. . . . .	58
3.9	Time responses of the synchronous two-hop MW-MSR algorithm. . . . .	58
3.10	Time responses of the asynchronous two-hop MW-MSR algorithm. . . . .	60
3.11	The 100-node sensor network. The red nodes are set as malicious one by one as $f$ increases up to 11. . . . .	60
3.12	Success rate of the MW-MSR algorithm. . . . .	62
3.13	Consensus error of the MW-MSR algorithm. . . . .	63
4.1	Both graphs are not 2-strongly robust with one hop but are 2-strongly robust with 2 hops. . . . .	69

**LIST OF FIGURES**

---

4.2	(a) (2,2)-robust. (b) 2-strongly robust. (c) 3-robust. . . . .	71
4.3	Time responses of the synchronous one-hop W-MSR algorithm. . . . .	82
4.4	Time responses of the synchronous two-hop MW-MSR algorithm. . . . .	83
4.5	Time responses of the asynchronous two-hop MW-MSR algorithm. . . . .	84
6.1	(a) The graph is not 2-strongly robust with one hop but is 2-strongly robust with 2 hops. (b) The graph is 2-strongly robust with one hop. . .	120
6.2	Time responses using different event-triggered MSR algorithms. . . . .	121
6.3	Time responses using different event-triggered MSR algorithms. . . . .	123
7.1	9-node graphs: (a) 4-connected and (b) (4,4)-robust. . . . .	129
7.2	Example graphs: Nodes 1 and 2 are malicious. In (a), there is no common normal neighbor, while in (b), node 4 is. . . . .	133
7.3	Example graphs satisfying the criteria for Scheme 2. . . . .	141
7.4	Undirected graph of 5 nodes with nodes 2 and 3 to be malicious. . . . .	141
7.5	9-node graphs under 2-total malicious model. . . . .	144
7.6	Scheme 1: Time responses of the states of all nodes. . . . .	145
7.7	9-node network satisfying the criteria for Scheme 2. . . . .	145
7.8	Scheme 2: Time responses of the states of all nodes. . . . .	146
7.9	Performance of different resilient consensus algorithms under attack sce- nario 1. . . . .	148
7.10	Performance of different resilient consensus algorithms under attack sce- nario 2. . . . .	149

# List of Tables

1.1	Resilient consensus under different adversary models. . . . .	7
5.1	Related works for resilient quantized consensus. . . . .	86
6.1	Average triggering times per normal node . . . . .	122
7.1	Tolerable number of malicious nodes for complete graphs. . . . .	143
7.2	Differences between Schemes 1 and 2. . . . .	143

# Chapter 1

## Introduction

### 1.1 Motivations

Networked systems have undergone significant change from centralized to distributed because of the strong flexibility and good computational performance of the distributed methods [18; 19; 71]. In the traditional centralized network or leader-follower network, not only heavy computational burden is placed on the central server, but also the ability of strong resilience against attacks need to be guaranteed on the central server to keep the entire network functional in the presence of adversaries. The nature of distributed algorithms where each agent can only interact with its neighbors is also the hard part for developing proper distributed algorithms. Consensus is one of the fundamental problems in distributed algorithms, which aims to achieve global agreement for each agent only using local information [11; 56; 62; 98]. This means that each agent can know and utilize only the information from its neighbors and be unaware of any information beyond its neighbors. One advantage of such consensus achieved by distributed computing over the centralized approach lies in the scalability of distributed algorithms. Many results in theories and applications based on consensus have been accomplished over the last decade such as distributed optimization [49; 75; 76; 77; 103; 104], average consensus [13; 135], clock synchronization [47; 50; 80; 118], energy management [29; 132], formation control [28; 81], and so on.

With large scale implementations of consensus based applications, consensus problems in the presence of adversary agents creating failures and attacks have become crucial; see, e.g., [22; 43; 56; 86; 87; 122]. Adversarial attacks can lead the systems

to hazardous operations and might cause physical faults or even accidents. For instance, the cyber attacks on the power grids can cause unexpected connection cuts and widespread power outages [32]. Here, we list several types of attacks studied in systems control area. Attack scenarios in the security studies of multi-agent systems can be categorized into replay, false data injection, denial-of-service (DoS), and so on [107]. Among these attacks, false data injection attacks and DoS attacks have attracted most the attention in the literature [16]. In this thesis, we study the false data injection attacks where misbehaving agents can send malicious data to prevent the normal agents from achieving the global goal. Along this line of works, it is typically assumed that each normal agent knows the maximum number of misbehaving agents in the entire network or at least among its neighbors. Let  $f$  denote this upper bound.

The misbehaving nodes or attacks considered in existing works can be characterized as three threat models according to the scope of threat levels: non-colluding/faulty, malicious, and Byzantine models. Faulty agents are caused by random node failures or errors, and they are unaware of the presence of faults in other agents and hence do not cooperate, which are also called non-colluding [86]. In the malicious or Byzantine case, misbehaving agents are capable to manipulate their own states arbitrarily and may even collude with each other to prevent regular nodes from achieving the global goal. Malicious nodes are limited as they must send the same false messages to all of their neighbors, while Byzantine nodes are capable to send different messages to different neighbors [62].

### Consensus under Byzantine Attacks

In security studies in computer science area [62; 108], the faults can be classified in two dimensions: time and nature. In the dimension of time, faults can be transient, permanent, and intermittent. Each type of faults exist for a different period of time. In the dimension of nature, faults are classified into crashes, omissions, duplications, desequencing, and Byzantine.

On the other hand, there are two kinds of fault-tolerant algorithm categories in the literature: (i) robust algorithms and (ii) self-stabilizing algorithms. In robust algorithms, redundancy on several levels of information, of communications, or of the system's nodes is used to overlap to the extent that the normal nodes can safely ex-

ecute their codes. In self-stabilizing algorithms, starting from an arbitrary state of the distributed system, each non-faulty node eventually reaches a state from where it behaves correctly. For instance, [78] considered a successful self-stabilizing algorithm as long as the only faults that occur henceforth (regardless of their number) are outside of the locality of this process. Note that the  $f$ -local model studied in this thesis can be related to such attacks.

As we mentioned earlier, among different types of attacks, Byzantine attacks simply correspond to an arbitrary type of fault. Resilient consensus problems under the Byzantine model have a rich history in the area of distributed algorithms in computer science [26; 62; 88]. There, such problems are also called *Byzantine agreement* where the faulty agents are capable to send erroneous and inconsistent messages to the normal neighbors. Many applications do not require exact agreement. It is satisfactory as long as the normal agents achieve consensus within a small bounded interval. This kind of convergence problems is called *approximate agreement* [27; 62]. A correct approximate Byzantine agreement algorithm must satisfy the following two conditions:

1. Validity: The values of normal agents must be within the range of initial values of normal agents for  $k > 0$ .
2. Agreement: For a given  $\epsilon > 0$ ,  $|x_i[k] - x_j[k]| < \epsilon$  for any normal agents  $i$  and  $j$ .

It has been shown in [26] that there are necessary and sufficient conditions for achieving Byzantine agreement in synchronous undirected networks expressed in terms of the node connectivity of the communication graph. Furthermore, in [34], it has been reported that under deterministic asynchronous updates, even one misbehaving agent can make it impossible for the system to reach exact consensus. Then, to avoid this constraint of exact consensus, [27] introduced the approximate Byzantine consensus problem in complete networks (i.e., under all-to-all communication), where the non-adversarial, normal nodes are required to achieve approximate agreement by converging to a relatively small interval in finite time. [102; 113] studied the synchronous approximate Byzantine consensus problem in networks with general topologies.

Even in recent years, reaching consensus resiliently in the presence of Byzantine faults has been studied extensively in distributed computing area [93; 113]. In [110], the authors explore the correctness of the Certified Propagation Algorithm (CPA) in solving broadcast with locally bounded Byzantine faults. In their work, there is as-



sumed to be a fault-free source node  $s$  with some value  $x_s$ , and the target of CPA is for every normal node in the network to output the same value  $x_s$  through the propagation algorithm. They provide a tight necessary and sufficient condition on the network topology for the correctness of CPA. In [64], the authors develop a fast protocol that reaches asynchronous Byzantine consensus in two communication steps in the common case. And they prove that the protocol is optimal in terms of both number of communication steps and number of processes for two-step consensus.

For the asynchronous approximate Byzantine consensus problem, [4] proposed a flooding-based algorithm for complete networks only. Recently, [94] studied the same problem for incomplete networks. We will compare our results with those in [94] in Chapter 4.

There are different techniques to mitigate the effects of attacks. Here, we introduce two types of approaches to cope with the problem: (i) the mean subsequence reduced (MSR) algorithms and (ii) detection-based algorithms. In the MSR algorithms [6; 56], normal agents simply neglect the information received from suspicious agents or those with unsafe values (outliers) whether or not they are truly adversarial. Thus, the normal agents do not have detection capabilities and do not have any memory to store the neighbors' past behaviors. Such techniques usually require the graph structure to be complex in order to achieve resilient consensus. On the other hand, in the detection-based algorithms [40; 86; 130], each agent has a bank of observers to identify the misbehaving agents using their past information. Then, normal agents can remove the values from these agents detected as adversarial. However, in the existing works, additional authorized detection resources are needed or each agent needs to detect the adversary agents in the entire network. In the following, we present our advancements in both directions through the introduction of multi-hop communication techniques to existing algorithms.

## 1.2 Context of the Study

In this thesis, we study resilient asymptotic consensus under malicious/Byzantine model from the viewpoint of the so-called mean subsequence reduced (MSR) algorithms, which are known for the simplicity and scalability (e.g., [3; 6; 56; 97; 113; 115; 116]). This line of work has gained much attention in the last decade as the malicious model can be widely assumed for broadcasting networks [56], wireless sensor networks [54], and

so on. A basic assumption in MSR algorithms is the knowledge regarding an upper bound on the maximum number of malicious agents among the neighbors. Such a bound represents the level of caution assumed by the system operator and can be set based on past experience, with possibly some safety margin. Then, at each iteration, each node eliminates extreme values received from neighbors to avoid being influenced by such potentially faulty values. In particular, it removes the  $f$  largest values and the  $f$  smallest values from neighbors. Moreover, the graph property called robustness is shown to be critical for the network structure, guaranteeing the success of resilient consensus algorithms in static networks [21; 56] as well as time-varying networks [95]. Nevertheless, such robustness requires the networks to be relatively dense and complex. Therefore, how to enhance resilience of more sparse networks without changing the original network topologies has become an urgent problem.

To tackle this problem, we develop several MSR-based algorithms with multi-hop communication. Such techniques are commonly used in the areas of wireless communication [37] and computer science [62]. Intuitively, with multi-hop communication, agents can communicate with their multi-hop neighbors through the relaying process [37]. It is clear that with multi-hop communication, each node can have more information for updating compared to the one-hop case. Yet, it also elaborates more on how the attackers may exploit the communication method to attack in different ways. It is not immediately clear if multi-hop can raise the security level. These questions motivate to study this problem in depth. For non-resilient case, multi-hop communication can be treated very easily. Typically, the communication delays is critical for the development of distributed algorithms in the multi-hop environment. It is hence of significant importance to extend our algorithm to the case of asynchronous updates with time delays.

More specifically, we prove necessary and sufficient conditions for the proposed algorithms under  $f$ -total/local malicious/Byzantine model. The graph conditions for related works are summarized in Table 1.1. For resilient consensus under malicious attacks, the works [56] and [21] considered the one-hop case. In Chapter 3, we extend the graph condition for the multi-hop case for the first time. Furthermore, in Chapter 4, we prove a tighter sufficient condition. For resilient consensus under Byzantine attacks, the work [113] considered the synchronous one-hop case and proposed a necessary and sufficient condition. The work [102] extended the graph condition for the synchronous

multi-hop case. Then, the work [94] proposed a flooding-based algorithm solving the Byzantine consensus under asynchronous unbound-hop case, where agents send their values throughout the entire network. Compared to these works, we analyze the  $f$ -local case and provide a unified analysis for synchronous and asynchronous cases. Moreover, our algorithm is of less computational complexity than the one in [94].

In various applications of wireless sensor networks (WSNs), the sensors usually have only limited computational resources. Calculating the real-valued states requires more resources which could be out of the capabilities of some low-cost sensors. In such networks, quantized state values are desirable. Thus, we also provide the analysis of our algorithm with the quantized state values. Furthermore, we introduce an event-triggered update scheme into our algorithms for the reduction of the communication loads between agents.

As an alternative approach to resilient consensus, we are also interested in fault detection and isolation (FDI) methods requiring only local information by agents. We propose a distributed detection method which is capable to detect malicious nodes in the network. In our detection scheme, each node acts as a local detector monitoring the behaviors of its neighbors. By transmitting its own state and relaying its neighbors' states to the neighboring nodes, each node is able to verify if the neighbors make any changes in the information set that they send. One key feature in the malicious model that we exploit is that the adversarial nodes are restricted to send the same information to its neighbors. Thus, as long as there are sufficiently many common neighbors in the network, the nonfaulty nodes are able to detect the malicious nodes, even if they collaborate with each other, and they can further achieve consensus among the nonfaulty ones in a resilient manner. We clarify the requirement on the network structure in terms of the graph connectivity for the proposed detection and consensus algorithms to properly function.

### 1.3 Main Contributions

In Chapter 2, we first provide a detailed review of the related works on cooperative control of networked multi-agent systems, as well as some background and preliminaries of the graph theory and multi-agent systems. Especially, we introduce some fundamental notions for the consensus problems, time synchronization, multi-hop communication and event-triggered techniques, which are directly related to the contributions of this

### 1.3 Main Contributions

Table 1.1: Resilient consensus under different adversary models.

		Synchronous	Asynchronous
Malicious	$f$ -total	$(f + 1, f + 1)$ -robust with $l$ hops ([21]; [56]; This work: Theorem 3.4.1)	$(2f + 1)$ -robust with $l$ hops ([21]; This work: Theorem 3.5.1) A tighter condition in Corollary 4.4.1
	$f$ -local	$(2f + 1)$ -robust with $l$ hops ([21]; [56]) A tighter condition in Corollary 4.3.1	$(2f + 1)$ -robust with $l$ hops ([21]; This work: Theorem 3.5.1) A tighter condition in Corollary 4.4.1
Byzantine	$f$ -total	$(f + 1)$ -strongly robust with $l$ hops ([102]; [113]; This work: Proposition 4.3.1 )	$(f + 1)$ -strongly robust with $l$ hops ([94]; This work: Theorem 4.4.1 )
	$f$ -local	$(f + 1)$ -strongly robust with $l$ hops (This work: Proposition 4.3.1)	$(f + 1)$ -strongly robust with $l$ hops (This work: Theorem 4.4.1)

Note that the notion of robustness (or strong robustness) is different under the  $f$ -total and  $f$ -local models. See Section 4.2 for details.

thesis.

In Chapters 3-7, we present our results on resilient consensus of networked multi-agent systems. Overall, the main contributions can be outlined as follows. First, we prove necessary and sufficient conditions for resilient consensus using the proposed algorithm under malicious/Byzantine attacks. We prove that with more communication (i.e., more hops), the proposed algorithm provides more robustness in sparse networks. We also extend our results to the agent systems with quantized states. Moreover, by introducing event-triggered updates to our algorithm, the transmissions of multi-hop relaying can be greatly reduced.

These chapters study different resilient consensus problems in the presence of compromised nodes (or adversaries). The adversary nodes attempt to mislead the uncompromised nodes by transmitting venomous data to these nodes. The goal of resilient consensus problems is for the uncompromised nodes (or normal nodes) to still achieve the global objective in the presence of the adversary nodes. As we discussed, the adversary nodes considered in existing works can be characterized as three threat models according to the scope of threat levels: non-colluding/faulty, malicious, and Byzantine models. There are also models which give the upper bound on the number of the adversary nodes in the network: (i) the  $f$ -total model, and (ii) the  $f$ -local model. The  $f$ -total model means there are at most  $f$  adversary nodes in the network. The  $f$ -local

model means there are at most  $f$  adversary nodes among the direct neighbors of each normal node [56]. In this thesis, we extend the two models to the multi-hop settings.

Specifically, in Chapter 3, we first define a discrete-time resilient asymptotic consensus problem under the malicious model. We introduce a mean subsequence reduced (MSR) algorithm where nodes can exchange state values with their multi-hop neighbors (Multi-hop Weighted-MSR algorithm). Then, we characterize the minimal requirements for network structures to guarantee the success of the abovementioned resilient consensus. The condition is expressed by the class of graph robustness with multi-hop communication. Our analysis highlights that through multi-hop communication, the network connectivity can be reduced especially in comparison with the common one-hop communication case. Moreover, we analyze the MW-MSR algorithm with delays in communication since the values from different multi-hop neighbors may arrive at the agents at different time steps.

In Chapter 4, utilizing the MW-MSR algorithm proposed in Chapter 3, we study an approximate consensus problem for the class of Byzantine adversaries when agents update asynchronously and there are time delays in the agents' communication. We find a tight graph condition for the proposed MW-MSR algorithm to solve the asynchronous approximate Byzantine consensus problem. The condition is denoted as strong robustness with multi-hop communication, which is a generalized form of the graph robustness with multi-hop communication introduced in Chapter 3. Since the Byzantine model is more adversarial than the malicious model, the network connectivity guaranteeing resilient consensus is more dense compared to the malicious case. We also compare our results with several related works. Overall, our algorithm is more light weighted than the conventional flooding-based algorithm.

In Chapter 5, we study the problem of resilient consensus where agents have integer-valued states. We develop a variation of the MW-MSR algorithm with randomized quantization. Quantized consensus has been motivated by concerns on limited capabilities in communications and computations of the agents. We analyze the performance of the proposed algorithm when agents update asynchronously and there are time delays in the communication between agents. Moreover, we find necessary and sufficient conditions for our algorithm to achieve resilient quantized consensus for synchronous/asynchronous updates under malicious/Byzantine attacks. Compared to existing approaches, our algorithm has a tighter graph condition and we provide a unified

analysis for different network settings.

In Chapter 6, we study the event-triggered resilient consensus problem where agents transmit their states only when local events are triggered. As we can see in the previous chapters, through multi-hop communication, the connectivity requirement becomes less stringent for guaranteeing the same level of resilience as for the one-hop case. This is enabled by increasing the amount of data exchanged among agents through multi-hop message relaying. Hence, the motivation for introducing the event-triggered update scheme to our algorithms is to reduce the transmissions among agents. Then we can achieve resilient consensus within a certain error level while keeping the transmissions among agents minimum. We also highlight that through multi-hop communication, the network connectivity can be reduced especially in comparison with the common one-hop event-triggered algorithms.

In Chapter 7, we switch our perspective to the detection-based resilient consensus algorithms. We develop a detection-based method that can achieve distributed fault detection under the malicious model by two approaches. Both are based on verification of two consecutive information sets of each agent. The difference between two ways is that one has the function that once a normal agent detects some malicious agents it is able to inform all agents in the network through an encrypted channel while the other one does not possess this function. More specifically, both approaches utilize the property of the malicious model where a malicious agent is limited to send the same information to its neighbors. We highlight that for the second approach, each normal agent will detect and isolate the malicious agents in its neighbor set independently through a simple majority voting scheme [62; 84; 101]. In this sense, the detection and isolation of malicious agents is achieved in a purely distributed fashion. It is shown that the proposed method may even handle some cases where the number of adversary nodes exceeds half of the total number of nodes in the network. Such a case cannot be achieved using the MSR-based algorithms.

The final chapter includes a summary of the work contained in this thesis, and provides directions for future research.

## 1.4 Overview of the Thesis

This section outlines the contents of this manuscript, and describes in broad terms the problems addressed within each chapter.

**Chapter 2**

- We provide a detailed review of a subset of the literature on resilient consensus of networked multi-agent systems, as well as some background in graph theory.

**Chapter 3**

- We formulate the resilient consensus problem under the malicious model with multi-hop communication.
- We introduce the multi-hop weighted-MSR algorithm, which utilizes the notion of message covers to filter the possible faulty values.
- We introduce a definition of network robustness that characterizes the network structure for resilient consensus with the multi-hop communication. The novel definition is referred to as  $(r, s)$ -robustness with  $l$  hops.
- We prove several properties of the networks that meet the conditions for  $(r, s)$ -robustness with  $l$  hops.
- We prove a tight necessary and sufficient condition on graph structure for the synchronous MW-MSR algorithm to achieve resilient consensus under the  $f$ -total malicious model.
- We prove a sufficient condition on graph structure for the asynchronous MW-MSR algorithm to achieve resilient consensus under the  $f$ -total malicious model.

**Chapter 4**

- We utilize the MW-MSR algorithm to solve the approximate Byzantine consensus problem.
- We introduce a definition of strong robustness that characterizes the network structure for Byzantine consensus. The novel definition is referred to as  $r$ -strongly robust graphs with  $l$  hops.
- We prove the relations between the notions of  $(r, s)$ -robust graphs with  $l$  hops and  $r$ -strongly robust graphs with  $l$  hops.

- We prove a tight necessary and sufficient condition on graph structure for the asynchronous MW-MSR algorithm to achieve resilient consensus under the  $f$ -local Byzantine model.

### Chapter 5

- We introduce the quantized MW-MSR algorithm with randomized quantization.
- We prove a necessary and sufficient condition on graph structure for the synchronous QMW-MSR algorithm to achieve resilient consensus under the  $f$ -total malicious model.
- We prove a necessary and sufficient condition on graph structure for the asynchronous QMW-MSR algorithm to achieve resilient consensus under the  $f$ -local Byzantine model.

### Chapter 6

- We formulate the event-triggered resilient consensus problem under the Byzantine model and introduce two relay models for transmission of the event-triggered values.
- We introduce the event-triggered MW-MSR algorithm as well as an alternative event-triggered scheme.
- We prove a necessary and sufficient condition on graph structure for the asynchronous EMW-MSR algorithm to achieve resilient consensus with error bound  $c$  under the  $f$ -local Byzantine model.

### Chapter 7

- We develop two distributed fault detection methods based on voting algorithms, where the second approach requires only local information for each agent to detect malicious neighbors.
- We prove necessary and sufficient conditions on graph structure for both protocols to achieve fault detection in time-invariant directed networks under the  $f$ -total/local malicious model.



- We demonstrate the effectiveness of the proposed detection method by numerical examples. We show that the second protocol can guarantee resilient consensus even if half of the total nodes are adversarial.

### Chapter 8

- We summarize the works contained in this thesis and provide possible directions for future research.

## Chapter 2

# Preliminaries and Related Works

This chapter discusses works most closely related to the contributions of this thesis. We first present material on multi-agent networks, and then some background is provided on related topics that are important to our contributions, such as graph theory, with an emphasis on algebraic graph theory, multi-hop communication, graph robustness, and voting algorithms. Then related research works in the literature are presented.

### 2.1 Fundamentals on Graph Theory

It is common to model the networked multi-agent system with a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where the agents have asymmetric sensing or communication capabilities. Here,  $\mathcal{V} = \{1, \dots, n\}$  represents the vertex set or the node<sup>2</sup> set of the  $n$  agents in the network. A directed edge set  $\mathcal{E}$  models the communication between agents. The edge  $(j, i) \in \mathcal{E}$  indicates that node  $i$  can get information from node  $j$ . Agent  $j$  is said to be an in-neighbor of agent  $i$  and agent  $i$  is an out-neighbor of agent  $j$ . The set of in-neighbors of agent  $i$  is defined by  $\mathcal{N}_i^- = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ . The set of out-neighbors of agent  $i$  is defined by  $\mathcal{N}_i^+ = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ . The in-degree of agent  $i$  is denoted by  $d_i^- = |\mathcal{N}_i^-|$ . Here,  $|\mathcal{S}|$  is the cardinality of a set  $\mathcal{S}$ . There are, of course, analogous definitions for out-neighbors, e.g., the out-degree of  $i$  is  $d_i^+ = |\mathcal{N}_i^+|$ . A complete digraph,  $K_n = (\mathcal{V}, \mathcal{E})$ , is defined by  $\mathcal{E} = \{(i, j) \in \mathcal{V} \times \mathcal{V} : i \neq j\}$ , and is also referred to as the complete network (all-to-all communication network). A graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is said to be undirected if all the communications between agents are bidirectional, i.e., if  $(j, i) \in \mathcal{E}$  then  $(i, j) \in \mathcal{E}$ .

---

<sup>2</sup>The terms ‘agent’ and ‘node’ are used synonymously throughout this thesis.

for any  $i, j \in \mathcal{V}$ .

Next, we introduce certain structural properties of digraphs which are related to our work. A path from node  $i_1$  to  $i_m$  is a sequence of distinct nodes  $(i_1, i_2, \dots, i_m)$ , where  $(i_j, i_{j+1}) \in \mathcal{E}$  for  $j = 1, \dots, m - 1$ . Such a path is referred to as an  $(m - 1)$ -hop path (or a path of length  $m - 1$ ) and also as  $(i_1, i_m)$ -path when the number of hops is not relevant but the source and destination nodes are. We also say that node  $i_m$  is reachable from node  $i_1$ . An  $\mathcal{X}u$ -path is a path from a node in set  $\mathcal{X}$  to node  $u \notin \mathcal{X}$ . We also denote the set minus symbol by  $\mathcal{X} \setminus \mathcal{Y}$ .

For node  $i$ , let  $\mathcal{N}_i^{l-}$  be the set of nodes that can reach node  $i$  via at most  $l$ -hop paths, where  $l$  is a positive integer. Also, let  $\mathcal{N}_i^{l+}$  be the set of nodes that are reachable from node  $i$  via at most  $l$ -hop paths. The  $l$ -th power of the graph  $\mathcal{G}$ , denoted by  $\mathcal{G}^l$ , is a multigraph<sup>1</sup> with the same vertices as  $\mathcal{G}$  and a directed edge from node  $j$  to node  $i$  is defined by a path of length at most  $l$  from  $j$  to  $i$  in  $\mathcal{G}$ .

### 2.1.1 Connectivity

A directed graph has a directed rooted spanning tree if there exists a node  $r$ , the root, such that every  $i \in \mathcal{V}$  is reachable from  $r$ . A graph  $\mathcal{G}$  (or digraph  $\mathcal{D}$ ) is said to be (strongly) connected if every node is reachable from every other node. A cut, node cut, or separating set of a connected graph  $\mathcal{G}$  is a set of nodes whose removal renders  $\mathcal{G}$  disconnected. The connectivity or node connectivity  $\kappa(\mathcal{G})$  (where  $\mathcal{G}$  is not a complete graph) is the size of a minimal node cut. A graph is called  $k$ -connected or  $k$ -node-connected if its node connectivity is  $k$  or greater. More precisely, any graph  $\mathcal{G}$  (complete or not) is said to be  $k$ -connected if it contains at least  $k + 1$  nodes, but does not contain a set of  $k - 1$  nodes whose removal disconnects the graph; and  $\kappa(\mathcal{G})$  is defined as the largest  $k$  such that  $\mathcal{G}$  is  $k$ -connected. In particular, a complete graph with  $n$  nodes, denoted  $K_n$ , has no node cuts at all, but  $\kappa(K_n) = n - 1$ .

### 2.1.2 Algebraic Graph Theory

In this part, some of the matrices associated to graphs and digraphs are introduced, such as the adjacency matrix, degree matrix, and Laplacian. The spectra of these matrices are of special interest. Define these objects with respect to a weighted digraph,  $\mathcal{D}_\omega = (\mathcal{V}, \mathcal{E}, \omega)$ , which has, in addition to a vertex set and directed edge set, a weight

---

<sup>1</sup>In a multigraph, two nodes can have multiple edges between them.

function  $\omega : \mathcal{E} \rightarrow \mathbb{R}$ . In this case, a weight  $\omega_{ij} \triangleq \omega(e) \in \mathbb{R}$  is associated to each directed edge  $e = (i, j) \in \mathcal{E}$ . A digraph  $\mathcal{D}$  is the specific weighted digraph in which  $\omega : \mathcal{E} \rightarrow \{1\}$ .

The weighted adjacency matrix,  $A_\omega(\mathcal{D}_\omega) = [a_{ij}]$ , associated with weighted digraph  $\mathcal{D}_\omega$  is the  $|\mathcal{V}| \times |\mathcal{V}|$  matrix defined by

$$a_{ij} = \begin{cases} \omega_{ji}, & (j, i) \in \mathcal{E}, \\ 0, & (j, i) \notin \mathcal{E}. \end{cases} \quad (2.1)$$

For digraphs, we define the adjacency matrix,  $A(\mathcal{D}) = [a_{ij}]$ , or just  $A$ , as in (2.1), but with  $\omega_{ji} = 1$ , for all  $(j, i) \in \mathcal{E}$ . Observe that for undirected graphs,  $A(\mathcal{D}) = A^T(\mathcal{D})$ , that is, the adjacency matrix is a symmetric matrix. Note that the weighted adjacency matrix of an undirected weighted graph is not necessarily symmetric.

The weighted in-degree matrix  $D_\omega^{\text{in}}$  of a weighted digraph  $\mathcal{D}_\omega$  is the diagonal  $|\mathcal{V}| \times |\mathcal{V}|$  matrix with entries along the main diagonal given by

$$[D_\omega^{\text{in}}]_{ii} = \sum_{j \in \mathcal{N}_i^{\text{in}}} \omega_{ji}, \quad i \in \{1, 2, \dots, n\}. \quad (2.2)$$

Similarly, we can also define the weighted out-degree matrix  $D_\omega^{\text{out}}$  as

$$[D_\omega^{\text{out}}]_{ii} = \sum_{j \in \mathcal{N}_i^{\text{out}}} \omega_{ij}, \quad i \in \{1, 2, \dots, n\}. \quad (2.3)$$

For an undirected graph,  $D_\omega^{\text{in}} = D_\omega^{\text{out}} \triangleq D$  is the degree matrix of  $\mathcal{G}$ .

Finally, the (in-degree) weighted Laplacian is the  $|\mathcal{V}| \times |\mathcal{V}|$  matrix defined by

$$L(\mathcal{D}_\omega) = D_\omega^{\text{in}} - A_\omega(\mathcal{D}_\omega). \quad (2.4)$$

Whenever we consider undirected graphs, the graph Laplacian, or just Laplacian,  $L(\mathcal{G})$  is of interest (and is defined as in (2.4) by  $L(\mathcal{G}) = D - A$ ). In this case,  $L(\mathcal{G})$  is a real symmetric matrix, so the eigenvalues are real. In fact,  $L(\mathcal{G})$  is symmetric and positive semidefinite, and thus all eigenvalues are nonnegative. It is advantageous to label the eigenvalues of  $L(\mathcal{G})$  in nondecreasing order by

$$\lambda_1(\mathcal{G}) \leq \lambda_2(\mathcal{G}) \leq \dots \leq \lambda_n(\mathcal{G}). \quad (2.5)$$

It holds that  $\lambda_1(\mathcal{G}) = 0$ . In fact, the multiplicity of the zero eigenvalue is equal to the number of components of  $\mathcal{G}$ . For connected graphs,  $\lambda_2(\mathcal{G}) > 0$ , and is called the Fiedler eigenvalue or the algebraic connectivity of the graph. This is because the magnitude of  $\lambda_2(\mathcal{G})$  is directly related to the sparsity of the graph. That is,  $\lambda_2(\mathcal{G})$  is small for sparse graphs and large for dense graphs [36].

The adjacency matrix  $A = [a_{ij}]$  of  $\mathcal{G}^l$  is given by  $\alpha \leq a_{ij} < 1$  if  $j \in \mathcal{N}_i^{l-}$  and otherwise  $a_{ij} = 0$ , where  $\alpha > 0$  is a fixed lower bound. We assume that  $\sum_{j=1, j \neq i}^n a_{ij} \leq 1$  for all  $i$ . Let  $L = [b_{ij}]$  be the Laplacian matrix of  $\mathcal{G}^l$ , whose entries are defined as  $b_{ii} = \sum_{j=1, j \neq i}^n a_{ij}$  and  $b_{ij} = -a_{ij}, i \neq j$ ; we can see that the sum of the elements of each row of  $L$  is zero.

## 2.2 Resilient Asymptotic Consensus

In the systems control literature, resilient consensus in the case of the so-called malicious model has been widely studied. This model is suitable for multi-agent applications such as wireless sensor networks and autonomous robotic networks, where the information exchange among the nodes is via broadcasting and sensing. Fault tolerant techniques using the so-called mean subsequence reduced (MSR) type algorithms can be found in, e.g., [21; 22; 47; 56]. A basic assumption in MSR algorithms is the knowledge regarding an upper bound on the maximum number of malicious agents among the neighbors; this bound is denoted by  $f$  throughout this thesis. Such a bound represents the level of caution assumed by the system operator and can be set based on past experience, with possibly some safety margin. Then, at each iteration, each node eliminates extreme values received from neighbors to avoid being influenced by such potentially faulty values. In particular, it removes the  $f$  largest values and the  $f$  smallest values from neighbors. Moreover, the graph property called robustness is shown to be critical for the network structure, guaranteeing the success of resilient consensus algorithms in static networks [21; 56] as well as time-varying networks [95]. A recent work [112] attempts to check robustness of given graphs using mixed integer linear programming. Nevertheless, such robustness requires the networks to be relatively dense and complex. Moreover, in general, MSR algorithms do not have the functionality to detect the adversaries.

In [21], the authors extend the MSR-type algorithms to the second order systems with asynchrony scheme and delays. Similar topological conditions in terms of robust graphs have been developed there. In [22], resilient consensus over the network consists

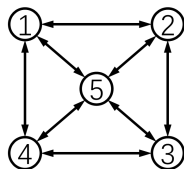


Figure 2.1: The graph is  $(2, 2)$ -robust.

of agents taking integer-valued (i.e., quantized) states under directed communication links is studied. They solve the resilient quantized consensus problems in the presence of totally/locally bounded adversarial agents and provide necessary and sufficient conditions in terms of the connectivity notion of graph robustness which is similar to the conditions in [56]. Furthermore, they show that randomization is essential both in quantization and in the updating times when normal agents interact in an asynchronous manner.

A common approach for increasing network robustness is redundancy: deploying additional nodes and establishing new links between nodes, which could be prohibitively expensive. The authors in [1] address the problem of improving structural robustness of networks without adding extra links. Instead, by introducing the notion of network connectivity with respect to trusted nodes, they prove that existence of such nodes has an effect of having a higher network connectivity or an improved  $r$ -robustness property.

### 2.2.1 Graph Robustness

For the theorem stated in [56; 57], the authors characterize the topology of the network by graph robustness. It measures the connectivity in a graph by showing how well the subgraphs are connected.

**Definition 2.2.1** *A digraph  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$  is said to be  $(r, s)$ -robust if for every pair of nonempty disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ , at least one of the following conditions holds:*

1.  $|\mathcal{X}_{\mathcal{V}_1}^r| = |\mathcal{V}_1|$ ;
2.  $|\mathcal{X}_{\mathcal{V}_2}^r| = |\mathcal{V}_2|$ ;
3.  $|\mathcal{X}_{\mathcal{V}_1}^r| + |\mathcal{X}_{\mathcal{V}_2}^r| \geq s$ ;

where  $\mathcal{X}_{\mathcal{V}_a}^r$  is the set of nodes in  $\mathcal{V}_a$  having at least  $r$  incoming edges from outside  $\mathcal{V}_a$ . As the special case with  $s = 1$ , graphs which are  $(r, 1)$ -robust are called  $r$ -robust.

For example, the graph in Fig. 2.1 is  $(2, 2)$ -robust. Note that the checking of robustness requires the combinations of all agent sets [112].

The following lemma in [56] provides a better understanding of graph robustness. Here, let  $\lceil \cdot \rceil$  denote the ceiling function.

**Lemma 2.2.1** *For an  $(r, s)$ -robust digraph  $\mathcal{D}$ , the following properties hold:*

1.  $\mathcal{D}$  is  $(r', s')$ -robust, where  $0 \leq r' \leq r$  and  $1 \leq s' \leq s$ , and it is  $r$ -robust.
2.  $\mathcal{D}$  has a directed spanning tree. Moreover a graph is 1-robust if and only if it has a directed spanning tree.
3.  $r \leq \lceil n/2 \rceil$ . Further,  $\mathcal{D}$  is a complete graph if  $r = \lceil n/2 \rceil$ .
4. The minimum in-degree of  $\mathcal{D}$ ,  $\delta^{in}(\mathcal{D})$ , is at least

$$\delta^{in}(\mathcal{D}) = \begin{cases} r + s - 1, & s < r; \\ 2r - 2, & s \geq r. \end{cases}$$

Moreover, a graph  $\mathcal{D}$  is  $(r, s)$ -robust if it is  $(r + s - 1)$ -robust.

Apparently, the definition of  $(r, s)$ -robustness is stronger than  $r$ -robustness. Moreover,  $(r, s)$ -robustness plays a key role to obtain a tight necessary and sufficient condition for the MSR-type algorithms to achieve consensus. More specifically, for the normal nodes in the network which has at most  $f$  malicious nodes to achieve resilient consensus, the underlying graph needs to be  $(f + 1, f + 1)$ -robust. In general, to verify the robustness property of a given graph is computationally difficult since it involves combinatorial procedures.

## 2.2.2 Resiliency Notions and Threat Models

Here, we define the attacks and threat models studied in this thesis. The attacks are launched by *adversary agents*, and they might send venomous data to deceive the *normal agents* and prevent them from reaching consensus. Specifically, agents are said to be normal if they follow the pre-specified protocols or update rules. Otherwise, they are said to be *adversarial*. As we mentioned earlier, in the literature [22; 43; 56; 86; 87; 122], adversary agents are characterized as three threat models according to the scope of threat levels: non-colluding/faulty, malicious, and Byzantine models. Faulty agents

simply stops working. In the malicious or Byzantine case, adversary agents are capable to manipulate their own states arbitrarily and may even collude with each other to prevent normal nodes from achieving the global goal.

In the network, the node set  $\mathcal{V}$  is partitioned into the set of normal nodes  $\mathcal{N}$  and the set of adversary nodes  $\mathcal{A}$ . The latter set  $\mathcal{A}$  is unknown to the normal nodes at all times. We generate the conventional adversary models to the multi-hop setting since adversary nodes do not only send their state values but also relay the values from neighbors [102]. Their formal definitions are given as follows.

**Definition 2.2.2** (*Malicious nodes*) *An adversary node  $i \in \mathcal{A}$  is said to be malicious if it can arbitrarily modify its own value and relayed values, but sends the same state and relayed values to its neighbors at each iteration. It can also decide not to send any value.*<sup>2</sup>

**Definition 2.2.3** (*Byzantine nodes*) *An adversary node  $i \in \mathcal{A}$  is said to be Byzantine if it can arbitrarily modify its own value and relayed values and sends different state values and relayed values to its neighbors at each iteration. It can also decide not to send any value.*

Byzantine models are well studied in the area of computer science [26; 62; 113]. Usually, such a model is assumed for point-to-point networks, where agents communicate with their neighbors one by one. Note that the malicious model studied in [56], [21] is a weaker threat model compared to Byzantine model. Moreover, the malicious model is reasonable in applications such as wireless sensor networks, where neighbors' information is obtained by broadcast communication.

Despite anonymity of the adversary agents, at least a scope of them must be known by normal agents. An upper-bound for attackers is usually a minimum knowledge for normal agents in security studies [62]. This upper bound can be defined for the whole network or just for the neighborhood of each normal agent. We provide the multi-hop generalization of the models proposed in [56].

**Definition 2.2.4** (*f-total set*) *The set of adversary nodes  $\mathcal{A}$  is said to be f-total if it contains at most f nodes, i.e.,  $|\mathcal{A}| \leq f$ .*

---

<sup>2</sup>This behavior corresponds to the omissive/crash model [62].



**Definition 2.2.5** (*f*-local set) *The set of adversary nodes  $\mathcal{A}$  is said to be *f*-local (in *l*-hop neighbors) if any normal node  $i \in \mathcal{N}$  has at most *f* adversary nodes as its *l*-hop neighbors, i.e.,  $|\mathcal{N}_i^{l-} \cap \mathcal{A}| \leq f$ .*

As commonly done in the resilient consensus literature [21; 56; 102], we assume that each normal node knows the value of *f* and the topology information of the graph up to *l* hops. Up to Chapter 6, we adopt the *l*-hop version of the threat models.

### 2.2.3 Resilient Consensus

In the well-studied discrete-time multi-agent consensus problem [82] (without any attacks), each node *i* has a state value  $x_i[k]$  at each time *k* and updates it as

$$\begin{aligned} x_i[k+1] &= x_i[k] + u_i[k], \\ u_i[k] &= - \sum_{j \in \mathcal{N}_i^-} a_{ij}[k] (x_i[k] - x_j[k]). \end{aligned} \tag{2.6}$$

where  $a_{ij}[k]$  is the (*j*, *i*)th entry of the adjacency matrix corresponding to the graph  $\mathcal{G}[k]$ .

The objective of the networked agents is consensus in the sense that the agents come to agreement and stop asymptotically:

$$\exists x^* \in \mathbb{R}, \lim_{k \rightarrow \infty} x_i[k] = x^*, \forall i \in \mathcal{V}. \tag{2.7}$$

A well-known condition guaranteeing consensus of the networks without adversary agents is that the graph  $\mathcal{G}$  has a directed rooted spanning tree [66; 83; 90].

In this thesis, we aim to find resilient algorithms and conditions on the topology of the network to ensure consensus among normal agents. The following definition describes mathematically what we seek as resilient consensus in discrete-time networks [21; 56; 102].

**Definition 2.2.6** *If for any possible sets and behaviors of the adversary agents and any state values of the normal agents, the following two conditions are satisfied, then we say that the normal agents reach resilient asymptotic consensus:*

1. *Safety: There exists a bounded safety interval  $\mathcal{S}$  determined by the initial values of the normal agents such that  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .*

2. *Agreement: There exists a state  $x^* \in \mathcal{S}$  such that  $\lim_{k \rightarrow \infty} x_i[k] = x^*, \forall i \in \mathcal{N}$ .*

### 2.2.4 Weighted MSR Algorithm

The weighted mean subsequence reduced (W-MSR) algorithm was proposed in [56; 57] for solving the resilient consensus problem under the one-hop communication environment. It can be outlined as follows.

1. Each normal node  $i$  receives the state values of its neighbors and sorts them in an increasing order.
2. If there are less than  $f$  values strictly larger than  $x_i[k]$ , then node  $i$  removes all the values that are strictly larger than  $x_i[k]$ . Otherwise, it removes the  $f$  largest state values in the sorted list. Similarly, if there are less than  $f$  values strictly smaller than  $x_i[k]$ , then node  $i$  removes all the values that are strictly smaller than  $x_i[k]$ . Otherwise, it removes the  $f$  smallest state values.
3. Node  $i$  takes the average of the remaining values in the list as the next state value.

The following propositions from [56] state the topological conditions required for resilient consensus of such networks using the W-MSR algorithm.

**Proposition 2.2.1** *Under the  $f$ -total malicious model, the multi-agent system using the W-MSR algorithm achieves resilient consensus if and only if the underlying graph is  $(f + 1, f + 1)$ -robust.*

**Proposition 2.2.2** *Under the  $f$ -local malicious model, the multi-agent system using the W-MSR algorithm achieves resilient consensus if the underlying graph is  $(2f + 1)$ -robust. Moreover, resilient consensus is achieved only if the underlying graph is  $(f + 1, f + 1)$ -robust.*

In the rest of this chapter, we introduce the background on different problem settings studied in this thesis.

## 2.3 Consensus with Multi-hop Communication

In this section, we discuss the multi-hop communication, which is the main topic of this thesis in the context of resilient consensus.

Multi-hop communication techniques are commonly used in the areas of wireless communication [37] and computer science [62]. Such techniques are also used for consensus problems. In [48], a multi-hop relay technique is introduced in the consensus problem to increase the speed of consensus forming. In [134], a similar method based on multi-hop relay is developed to solve the global leader-following consensus problem. Moreover, application of multi-hop communication in wireless sensor networks from the viewpoint of control is investigated in [63]. In the systems and control area, there are security-related works which analyze the stability of the networked control systems with control inputs and observer information sent over multi-hop networks ([15; 25]). It is clear that with multi-hop communication, each node can have more information for updating compared to the one-hop case. Thus, the network may have more resilience against adversary nodes.

A multi-hop communication network can be characterized by two parameters:  $k$ -hop topology knowledge and the relay depth  $k'$ . More specifically, for different values of integers  $k, k'$ , we consider that each node knows all its neighbors of at most  $k$ -hop distance ( $k$ -hop topology knowledge), and the relay depth  $k'$  to represent the maximum number of hops any message can be relayed.

For resilient consensus problems, the work [113] for Byzantine adversaries has been extended to multi-hop communication network by introducing multi-hop communication (referred as  $k$ -hop) techniques in [102]. There, by assuming  $k = k'$ , the authors solved the approximate Byzantine consensus problem with a weaker condition on network structures compared to that derived under the one-hop communication model [113]. Moreover, they also proved that when  $k = 1$ , the necessary condition for iterative approximate Byzantine consensus algorithm to exist is the same as the result from [62].

Further, in [93], the authors develop algorithms for the asynchronous crash-tolerant consensus problem under such multi-hop communication networks. And they develop multi-hop fault-tolerant consensus algorithms for  $k \neq k'$  cases. In [94], the authors tackled the same problem under asynchronous updates based on rounds, which is different from the asynchrony setting used in this thesis (see the discussions in Chapter 4). The

work [53] studied Byzantine binary consensus under the local broadcast model (malicious model) using a flooding algorithm, where nodes relay their values over the entire network.

We adopt the same communication model as the one in [102], i.e., we assume  $k = k'$  throughout the thesis.

### 2.3.1 Multi-hop Communication Model

In this part, we introduce the model of multi-hop communication used in this thesis. Our model is similar to that in [102] under the Byzantine attacks. For the communication model under malicious attacks, we use a relay model based on broadcast communication. In the multi-hop communication setting, the agents not only communicate with their direct neighbors as in conventional schemes, but also with their multi-hop neighbors by having their messages relayed. Let  $l$  be the maximum number of hops allowed in the network. Specifically, node  $i_1$  can send messages of its own to an  $l$ -hop neighbor  $i_{l+1}$  via different paths. We represent a message as a tuple  $m = (w, P)$ , where  $w = \text{value}(m) \in \mathbb{R}$  is the message content and  $P = \text{path}(m)$  indicates the path via which message  $m$  is transmitted. Moreover, nodes  $i_1$  and  $i_{l+1}$  are, respectively, the message source and the message destination. When source node  $i_1$  sends out a message,  $P$  is a path vector of length  $l + 1$  with the source node being  $i_1$  and other entries being empty. Then the one-hop neighbor  $i_2$  receives this message from  $i_1$ , and it stores the value of node  $i_1$  for consensus and relays the value of node  $i_1$  to all the one-hop neighbors of  $i_2$  with the second entry of  $P$  being  $i_2$  and other entries being unchanged. This relay procedure will continue until every entry of  $P$  of this message is occupied, i.e., this message reaches node  $i_{l+1}$ . We denote by  $\mathcal{V}(P)$  the set of nodes in  $P$ .

We now outline the message exchanges among the agents. This notion of messages is used until Chapter 7, where information sets are exchanged among agents. At each time  $k$ , normal node  $i$  conducts the following steps:

1. *Transmit step*: Transmit message  $m_{ij}[k] = (x_i[k], P_{ij}[k])$  over each  $l$ -hop path to node  $j \in \mathcal{N}_i^{l+}$ .
2. *Receive step*: Receive messages  $m_{ji}[k] = (x_j[k], P_{ji}[k])$  from  $j \in \mathcal{N}_i^{l-}$ , whose destination is  $i$ . Let  $\mathcal{M}_i[k]$  be the set of messages that node  $i$  received in this step.

3. *Update step*: Update the state  $x_i[k]$  as

$$x_i[k + 1] = g_i(\mathcal{M}_i[k]), \quad (2.8)$$

where  $g_i(\cdot)$  can be a real-valued or quantized function of the states received in this time step, to be defined later in each chapter.

In the Transmit step and Receive step, nodes exchange messages with others that are up to  $l$  hops away. Then in the Update step, node  $i$  updates its state using the received values in  $\mathcal{M}_i[k]$ . Note that the adversary nodes may deviate from this specification.

In the multi-hop setting studied in this thesis (up to Chapter 6), it is important to impose the following assumption.

**Assumption 2.3.1** *Each adversary node  $i$  cannot manipulate the path values in the messages containing its own state  $x_i[k]$  and those that it relays.*

This is introduced for ease of analysis, but is not a strong constraint. In fact, manipulating message paths can be easily detected and hence does not create problems. We show how this can be done in the following section.

### 2.3.2 Discussion on Manipulation in Message Path Information

It is notable that multi-hop communication is vulnerable to false data injection in the information relayed by nodes, which can make the problem of resilient consensus more complicated than the one-hop case. Earlier, Assumption 2.3.1 was introduced stating that the malicious nodes however cannot manipulate the path information in messages that they relay. We briefly explain here how such attacks can be detected, inspired by the discussion in [102].

Such detection requires that each node can identify the neighbor from which it receives each message, which is commonly assumed (e.g., [26; 102]). Moreover, there are many methods to realize this function in real-world applications. For instance, by using the encryption technique of the RSA algorithm [92], each node can send out its value associated with a digital signature using its own private key. Then, using the sender's public key, the receiver can confirm that this message is indeed sent by the particular sender.

## 2.3 Consensus with Multi-hop Communication

---

In each iteration of the synchronous algorithm, there are three potential cases where a message sent to normal node  $i$  is manipulated in its path information: (i) Node  $i$  receives multiple messages along the same path  $P$ ; (ii) it receives messages along an unknown path  $P'$ ; or (iii) it does not receive any message along a known path  $P$ . Note that a normal node receives only one message along each path in each iteration when no adversarial node is present.

For case (i), this faulty behavior is caused either by duplicating messages or by manipulating path information in messages. We show that in both situations, the receiving node  $i$  can find that there is at least one faulty node in path  $P$ . It is obvious for the first situation. For the path manipulating situation, consider the case where a normal node  $h$  receives a message  $m = (w, P)$  directly from node  $j$  but the path  $P$  does not contain node  $j$ . Then node  $h$  knows that node  $j$  is faulty, and will not forward the message. This indicates that in general, if there is a sequence of faulty nodes along a path, then the last one in the sequence must keep its own index within the path information in the messages that it transmits. Moreover, this argument also holds for case (ii), i.e., node  $i$  knows that at least one node in path  $P'$  is faulty. Therefore, in cases (i) and (ii), from the perspective of node  $i$ , manipulating the message path data is equivalent to having a faulty node in  $P$  or  $P'$  sending additional messages with manipulated values, and thus it will remove any values in this path by the MW-MSR algorithm.

For case (iii), either a faulty node does not send/forward the message  $m$ , or it manipulates the message path  $P$ . In the latter case, for node  $i$ , manipulating the message path is equivalent to having a faulty node in  $P$  not sending/forwarding the message.

Actually, this analysis can be extended to the algorithms with asynchronous updates. In each update of such algorithms, consider the following three path manipulating cases for node  $i$ : (i) Node  $i$  receives multiple messages along one path  $P$  at the same time step; (ii) it receives messages along an unknown path  $P'$ ; or (iii) it does not receive any message along path  $P$  in a period of time  $\tau$ , where  $\tau$  is the maximum time delay of normal agents. Note that in case (i) for the asynchronous algorithm, faulty nodes can send multiple messages along  $P$  as long as these faulty messages do not arrive at node  $i$  at the same time step and this behavior will not affect normal agents, since only the most recent values of multi-hop neighbors will be used in the

asynchronous MW-MSR algorithm. The analysis of cases (ii) and (iii) is similar to that of the synchronous algorithm.

The above analysis is based on the assumption that there is no packet loss in the fault-free networks. In real-world applications, packet losses can happen even in fault-free networks. We note that there are methods to deal with this issue. Packet losses in the communication between two normal nodes can also cause the situation of case (iii) mentioned above. Like the one-hop W-MSR algorithm, if a packet loss happens in the communication from neighbor  $j$  to node  $i$ , then node  $i$  may receive only  $|\mathcal{N}_i| - 1$  values at this particular time step and still remove  $f$  largest and  $f$  smallest received values; hence, node  $i$  uses less information from normal nodes to update. This behavior will not violate the safety interval, but it may slow down the speed of consensus. If the packet loss behavior happens frequently in this transmission path, then node  $i$  can consider this path containing faulty nodes.

## 2.4 Quantized Consensus

In this section, we introduce the related works and background that are relevant to the resilient quantized consensus problem studied in Chapter 5.

In many applications of wireless sensor networks, the sensor nodes may have access to only limited memories and transmission bandwidth [37]. In such cases, the agents can only compute the integer-valued states. Quantized consensus has been motivated by such concerns on limited capabilities in communications and computations of the agents. There are various studies that have looked into the case without any adversary agents [5; 12; 14; 30; 38; 55; 58].

There are also several works that have dealt with the resilient quantized consensus problem, where some agents are under attacks [22; 115]. The authors of [22] studied the resilient quantized consensus using randomized quantizer with one-hop communication. There, a necessary and sufficient condition for synchronous resilient consensus under malicious attacks was provided. The graph condition is same as the one in the real-valued case [56].

Exact Byzantine consensus is a popular and historical topic in the area of computer science [62]. The work [26] first proposed two conditions which are necessary and sufficient for exact Byzantine consensus in undirected networks: (i)  $n > 3f$ ; (ii) node-connectivity is no less than  $2f + 1$ . Then, the authors of [109] provided a necessary and

sufficient condition for exact Byzantine consensus (integer-valued) in directed networks. Both works are studied for synchronous updates. However, in a real-world environment, delays are almost natural in the communication among nodes. Thus, it is important to analyze whether the proposed algorithm can successfully achieve resilient quantized consensus in asynchronous updates with delays. Until recently, the work [117] studied binary Byzantine consensus in asynchronous updates with delays and they derived the same necessary and sufficient conditions as the two in [26]. The resilient quantized consensus (including binary consensus) works [26; 52; 109; 117] from computer science commonly assume that each normal node sends its values to the entire network through different paths, which corresponds to one of our cases called unbounded path length case ( $l \geq l^*$ ). Moreover, the authors of [52; 53] provide a necessary and sufficient condition for synchronous binary Byzantine consensus under local broadcast model in undirected/directed networks, respectively. Note that the local broadcast model is equivalent to the malicious model.

## 2.5 Event-based Communication

In this section, we discuss the background related to the event-triggered resilient consensus problem studied in Chapter 6.

Through multi-hop communication, the connectivity requirement may become less stringent for guaranteeing the same level of resilience as for the one-hop case. This is enabled by increasing the amount of data exchanged among agents through multi-hop message relaying as well as the algorithm with more complexity. Hence, the motivation for introducing the event-triggered update scheme to our algorithms is to reduce the transmissions among agents. We aim to reduce the transmissions for the agents using the multi-hop weighted MSR algorithm [125] through event-triggered protocols [45].

In traditional periodic control systems, control signals are sent periodically. This may consume a large amount of energy and communication resources. Event-triggered and self-triggered control systems have become popular in the systems control area because they are energy-efficient. In event-triggered control, a triggering condition based on current measurements is continuously monitored and an event is triggered only when this condition is satisfied. In self-triggered control, the controller will compute the next update time based on predictions using previously received data and the knowledge on the plant dynamics [45].



Event-based protocols have been developed for conventional consensus without adversary agents in, e.g., [23; 50; 68]. Moreover, the resilient consensus work [114] proposed two event-based MSR algorithms using one-hop communication to reduce the transmissions. Among these works, event-triggered schemes have shown their effectiveness in reducing the transmissions for the agents using distributed algorithms even under adversarial environments. Moreover, time delays can be a critical factor affecting the performance of agents in the multi-hop communication. Hence, we introduce event-triggered protocols to the multi-hop weighted MSR algorithm, and we are interested to analyze the performance of the proposed algorithm with delays in the communication between agents. More discussions are given in Chapter 6.

## 2.6 Fault Detection and Identification

In this section, we introduce related works and the majority voting algorithm that are used in the distributed detection approach proposed in Chapter 7.

From the security viewpoints, it is desirable to equip the nodes with distributed algorithms for fault detection and identification (FDI). For consensus-type problems, FDI techniques based on unknown input observers were proposed in, e.g., [86; 106]. These schemes however impose strong assumptions that each agent should have the global knowledge of the entire network and also have sufficient computation resources to run a number of observers.

In [86], by using unknown input observers (UIO), the authors proposed a fault detection method for detecting and isolating the malicious attacks. The distributed FDI scheme has been extended to the second-order systems in [99]. Nevertheless, the FDI schemes proposed in these works often require each node to have the global network knowledge.

Furthermore, in [7], the authors proposed a method for detection and isolation of faulty nodes based on clustering techniques. In [100], the authors proposed a fault detection technique in randomized gossip algorithms which is addressed using Set-valued Observers (SOVs). However, this SOV based method requires information on the global network structure.

### 2.6.1 Distributed Fault Detection

In this thesis, we are interested in FDI methods requiring only local information by agents. In our fault detection framework, every agent, namely agent  $i$ , will also act as a local monitor that watches the behavior of its neighboring agent, namely agent  $j$ , during the iterations of consensus algorithm. It becomes crucial for agent  $i$  to have the information about the control input for its neighbor  $j$ , which consists of the values of agent  $j$ 's neighbors in the consensus algorithm. However, agent  $i$  may not have full access to the control elements of agent  $j$  in the traditional one-hop communication network, i.e., agents in such a network can only communicate with their direct neighbors. It becomes possible for agent  $i$  to get all the neighbors' information of agent  $j$  if we introduce the information set on each agent, which can be seen as the so-called two-hop communication in [62; 102], where the Byzantine model is studied.

Such an approach can be found in [31; 40], where each agent acts as a detector. Specifically, it monitors its neighbors by iteratively exchanging more information than in conventional consensus algorithms. There, the nonfaulty, normal agents not only send their own states but also relay their neighbors' state values. Then, they can verify if the states sent by a particular neighbor are consistent with those of other neighbors.

In [40], the authors consider the adversary nodes to be faulty nodes. In their work, agent  $i$  can get the neighbors' information of agent  $j$  through two means. One is that agent  $j$  will send its neighbors' values to agent  $i$  at each time step. This requires that agent  $j$  to be honest about the sending information, which is true for the faulty agent case while this no longer stands for the malicious and Byzantine agent case. The other is that agent  $i$  gets the neighbors' information of agent  $j$  through on-board sensing based method, and this requires a bank of sensors loaded on each agent and it is difficult to apply this method in large scale network.

Similar ideas are explored in [44; 130]. However, these methods often impose strong assumptions. For example, the adversarial nodes cannot be neighbors so that they cannot share information nor collaborate. Another one is the introduction of mobile agents which randomly visit the agents and communicate with each other; they carry out a certain portion of the workload for detection. Thus a fault detection method which only uses local information and is able to detect general malicious or Byzantine agents is highly desirable.

### 2.6.2 Voting Algorithms

Voting is important for reliable distributed systems that are based on the multi-agent computation paradigm. A voting algorithm specifies how the voting result is obtained from the input data and can be the basis for implementing a hardware voting network or a software voting routine [84]. Voting algorithms have many categories based on the type of voting (exact, inexact, or approval), rule for output selection (plurality or threshold) and properties of the input object space (size and structure).

For our concern, exact voting is the most common voting method and is the easiest to implement. Inexact voting algorithms are more complicated due to intransitivity of approximate equality. As an example, when approximate equality  $a \cong b$  for numerical inputs  $a$  and  $b$  is defined as  $|a - b| \leq \epsilon$ , then  $a \cong b$  and  $b \cong c$  do not imply  $a \cong c$ . In approval voting, each input to the voting process consists of a finite or infinite set of values that have been “approved” by the corresponding computation channel and the value, or the set of values, with the highest approval voting must emerge as output.

In [42], the authors present a voting protocol that reduces the vulnerability of the voting process to both attacks and faults. This algorithm is applicable to exact and inexact voting in networks where atomic broadcast and predetermined message delays are present, such as local area networks.

In wireless sensor networks where accurate location of sensors is vital, one common method for location discovery uses a set of specialty nodes known as beacon nodes (BNs) that assist other sensor nodes to determine their location. In [101], the authors propose a novel reputation based scheme called Distributed Reputation-based Beacon Trust System for excluding malicious BNs that provide false location information.

## Chapter 3

# Resilient Consensus under Malicious Attacks with Multi-hop Communication

In this chapter, we study the resilient consensus problem under malicious attacks. Our approach is based on that of the weighted MSR algorithm with multi-hop communication. The MSR algorithm is a powerful tool for achieving resilient consensus under minimal requirements for network structures, characterized by the class of robust graphs. Our analysis highlights that through multi-hop communication, the network connectivity can be reduced especially in comparison with the common one-hop case.

Specifically, we develop a multi-hop version of MSR algorithms to solve the resilient consensus problem. Unlike in the case with one-hop communication, the MSR algorithm in the multi-hop case may not exclude all the possible effects from malicious nodes if each normal node just eliminates the  $f$  largest and the  $f$  smallest received values. Since a malicious node can manipulate not only its own value but also the values it relays, such nodes can produce more than  $f$  false values even if there are at most  $f$  malicious nodes. To completely exclude the effects from malicious nodes, we propose the multi-hop weighted mean subsequence reduced (MW-MSR) algorithm. Normal nodes using the MW-MSR algorithm will exclude the extreme values which are produced precisely by  $f$  multi-hop neighbors. To realize this trimming capability in the multi-hop setting requires the notion of *message cover*, which represents the set of nodes intersecting with a given set of different message paths.

Then we derive necessary and sufficient graph conditions based on the new notion of robustness with  $l$  hops for the proposed MW-MSR algorithms to achieve resilient consensus under synchronous updates and asynchronous updates with time delays in the communication. Moreover, we present examples to illustrate how multi-hop communication helps to improve graph robustness without changing the network topology. As a side result, we prove that for the case of unbounded path length in message relaying, our graph condition is equivalent to the necessary and sufficient graph condition for binary consensus under malicious attacks studied in [53].

The rest of this chapter is organized as follows. In Section 3.1, we outline the system model. In Sections 3.2 and 3.3, we present the MW-MSR algorithm and define graph robustness with multi-hop communication, respectively. Then in Sections 3.4 and 3.5, we derive tight graph conditions under which the MW-MSR algorithms guarantee resilient asymptotic consensus under synchronous and asynchronous updates, respectively. In Section 3.6, we provide some properties of the new robustness and in Section 3.7, we present examples to demonstrate that multi-hop communication can improve robustness of graphs in general. Lastly, in Section 3.8, we conclude this chapter.

## 3.1 Problem Formulation

In this section, we provide preliminaries on the network models and introduce the basic settings for the resilient consensus problems studied in this chapter. Note that some basic notions of graph theory are introduced in Section 2.1 of Chapter 2.

### 3.1.1 Multi-hop Communication for Multi-agent Consensus

First, we introduce the multi-agent system with multi-hop communication and the update rule used by the agents under no attacks. Consider a time-invariant network modeled by the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . Each node  $i$  has a real-valued state  $x_i[k]$ . The goal of the agents is to arrive at consensus in their state values asymptotically, that is,  $|x_i[k] - x_j[k]| \rightarrow 0$  as  $k \rightarrow \infty$  for all  $i, j \in \mathcal{V}$ . This is to be achieved by updating the states at each time step  $k$  based on the information exchanged among the nodes. Their initial values  $x_i[0]$  are given. Until we reach Section 3.6, we assume that no delay is present in the communication among nodes.

Recall that agents communicate with each other according to the communication

model indicated in Section 2.3.1. In an agent network equipped with multi-hop communication, as the consensus update rule (2.8), we can extend the common one (e.g., [82]). Let  $u_i[k]$  denote the control input for node  $i$  at time  $k$ . Each node updates as

$$\begin{aligned} x_i[k+1] &= x_i[k] + u_i[k], \\ u_i[k] &= - \sum_{j \in \mathcal{N}_i^{l-}} a_{ij}[k] (x_i[k] - x_j[k]). \end{aligned} \tag{3.1}$$

This system can be given in the compact form as

$$\begin{aligned} x[k+1] &= x[k] + u[k], \\ u[k] &= -L[k]x[k], \end{aligned} \tag{3.2}$$

where  $x[k] \in \mathbb{R}^n$  and  $u[k] \in \mathbb{R}^n$  are the state vector and control input vector, respectively, and  $L[k]$  is the Laplacian matrix of the  $l$ -th power of  $\mathcal{G}$  determined by the messages  $m_{ij}[k]$ ,  $i \in \mathcal{V}$  and  $j \in \mathcal{N}_i^{l-}$ . As a generalization of the one-hop result (e.g., [11; 66]), it is obvious that with  $l$ -hop communication, consensus is possible if  $\mathcal{G}^l$  has a rooted spanning tree.

#### 3.1.2 Threat Model

In this chapter, we study resilient consensus under the  $f$ -total malicious attacks. Note that all the results for asynchronous updates in this chapter also hold for the  $f$ -local malicious model. We will review the gaps between different graph conditions in Chapter 4. As commonly done in the literature [56; 102], we assume that each normal node knows the value of  $f$  and the topology information of the graph up to  $l$  hops. Moreover, the malicious model is reasonable in applications such as wireless sensor networks, where neighbors' information is obtained by broadcast communication. We assume that each adversary node  $i$  cannot manipulate the path values in the messages containing its own state  $x_i[k]$  and those that it relays as stated in Assumption 2.3.1.

#### 3.1.3 Resilient Asymptotic Consensus

We now introduce the type of consensus among the normal agents to be sought in this chapter [21; 56; 102].

**Definition 3.1.1** *If for any possible sets and behaviors of the malicious agents and*

any state values of the normal nodes, the following two conditions are satisfied, then we say that the normal agents reach resilient asymptotic consensus:

1. *Safety:* There exists a bounded safety interval  $\mathcal{S}$  determined by the initial values of the normal agents such that  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .
2. *Agreement:* There exists a state  $x^* \in \mathcal{S}$  such that  $\lim_{k \rightarrow \infty} x_i[k] = x^*, \forall i \in \mathcal{N}$ .

The problem studied in this chapter is to develop an MSR-based algorithm for agents that can make  $l$ -hop communication to reach resilient consensus under the  $f$ -total malicious model and to characterize conditions on the network topology for the algorithm to properly perform. Note that in general, for MSR-based algorithms with one-hop communication, resilient consensus can be achieved under the  $f$ -total model with the necessary and sufficient condition expressed in terms of the so-called graph robustness; see, e.g., [22; 56], and the following sections for the definition of robust graphs and related discussions.

## 3.2 Multi-hop Weighted MSR Algorithm

In this section, we introduce the multi-hop weighted MSR (MW-MSR) algorithm, which is designed to solve the resilient consensus problem under the multi-hop setting. We first introduce the notion of message cover which plays a key role in the trimming function of our MSR algorithm. Then we outline the structure of the MW-MSR algorithm and provide examples to illustrate the idea behind the algorithm.

The notion of message cover [102] is crucial in the update rule of our algorithm to be proposed in this section. It evaluates the effects of adversary nodes that can possibly manipulate the updates of normal nodes in a multi-hop communication setting. Its formal definition is given as follows.

**Definition 3.2.1** For a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , let  $\mathcal{M}$  be a set of messages transmitted over  $\mathcal{G}$ , and let  $\mathcal{P}(\mathcal{M})$  be the set of message paths of all the messages in  $\mathcal{M}$ , i.e.,  $\mathcal{P}(\mathcal{M}) = \{\text{path}(m) : m \in \mathcal{M}\}$ . A message cover of  $\mathcal{M}$  is a set of nodes  $\mathcal{T}(\mathcal{M}) \subset \mathcal{V}$  whose removal disconnects all message paths, i.e., for each path  $P \in \mathcal{P}(\mathcal{M})$ , we have  $\mathcal{V}(P) \cap \mathcal{T}(\mathcal{M}) \neq \emptyset$ . In particular, a minimum message cover of  $\mathcal{M}$  is defined by

$$\mathcal{T}^*(\mathcal{M}) \in \arg \min_{\mathcal{T}(\mathcal{M}): \text{Cover of } \mathcal{M}} |\mathcal{T}(\mathcal{M})|.$$

### 3.2 Multi-hop Weighted MSR Algorithm

---



---

**Algorithm 1:** MW-MSR Algorithm
 

---

- 1) At each time  $k$ , normal node  $i$  sends its own message to nodes in  $\mathcal{N}_i^{l+}$ . Then, it obtains messages of the nodes in  $\mathcal{N}_i^{l-}$  and itself, whose set is denoted by  $\mathcal{M}_i[k]$ , and sorts the values in  $\mathcal{M}_i[k]$  in an increasing order.
- 2) (a) Define two subsets of  $\mathcal{M}_i[k]$  based on the message values:

$$\overline{\mathcal{M}}_i[k] = \{m \in \mathcal{M}_i[k] : \text{value}(m) > x_i[k]\},$$

$$\underline{\mathcal{M}}_i[k] = \{m \in \mathcal{M}_i[k] : \text{value}(m) < x_i[k]\}.$$

- (b) Then, let  $\overline{\mathcal{R}}_i[k] = \overline{\mathcal{M}}_i[k]$  if the cardinality of a minimum cover of  $\overline{\mathcal{M}}_i[k]$  is less than  $f$ , i.e.,  $|\mathcal{T}^*(\overline{\mathcal{M}}_i[k])| < f$ . Otherwise, let  $\overline{\mathcal{R}}_i[k]$  be the largest sized subset of  $\overline{\mathcal{M}}_i[k]$  such that (i) for all  $m \in \overline{\mathcal{M}}_i[k] \setminus \overline{\mathcal{R}}_i[k]$  and  $m' \in \overline{\mathcal{R}}_i[k]$  we have  $\text{value}(m) \leq \text{value}(m')$ , and (ii) the cardinality of a minimum cover of  $\overline{\mathcal{R}}_i[k]$  is exactly  $f$ , i.e.,  $|\mathcal{T}^*(\overline{\mathcal{R}}_i[k])| = f$ .
  - (c) Similarly, let  $\underline{\mathcal{R}}_i[k] = \underline{\mathcal{M}}_i[k]$  if the cardinality of a minimum cover of  $\underline{\mathcal{M}}_i[k]$  is less than  $f$ , i.e.,  $|\mathcal{T}^*(\underline{\mathcal{M}}_i[k])| < f$ . Otherwise, let  $\underline{\mathcal{R}}_i[k]$  be the largest sized subset of  $\underline{\mathcal{M}}_i[k]$  such that (i) for all  $m \in \underline{\mathcal{M}}_i[k] \setminus \underline{\mathcal{R}}_i[k]$  and  $m' \in \underline{\mathcal{R}}_i[k]$  we have  $\text{value}(m) \geq \text{value}(m')$ , and (ii) the cardinality of a minimum cover of  $\underline{\mathcal{R}}_i[k]$  is exactly  $f$ , i.e.,  $|\mathcal{T}^*(\underline{\mathcal{R}}_i[k])| = f$ .
  - (d) Finally, let  $\mathcal{R}_i[k] = \overline{\mathcal{R}}_i[k] \cup \underline{\mathcal{R}}_i[k]$ .
- 3) Node  $i$  updates its value as follows:

$$x_i[k+1] = \sum_{m \in \mathcal{M}_i[k] \setminus \mathcal{R}_i[k]} a_i[k] \text{value}(m), \quad (3.3)$$

where  $a_i[k] = 1/(|\mathcal{M}_i[k] \setminus \mathcal{R}_i[k]|)$ .

---

As a simple example, consider the set  $\mathcal{M}$  of paths connecting node  $i$  to node  $j$  which do not overlap. Then, its message cover must contain at least one node per path. Clearly, there may be multiple minimum message covers if the paths are of length greater than three.

Now, we are ready to introduce the structure of the synchronous *MW-MSR algorithm* in Algorithm 1. Note that the one-hop version of the MW-MSR algorithm (i.e., with  $l = 1$ ) is equivalent to the W-MSR algorithm in [56]. However, we can see that the difference between the MW-MSR algorithm and the W-MSR algorithm mainly lies in the trimming function in step 2 when  $l \geq 2$ . For general MSR algorithms of one-



### 3.2 Multi-hop Weighted MSR Algorithm

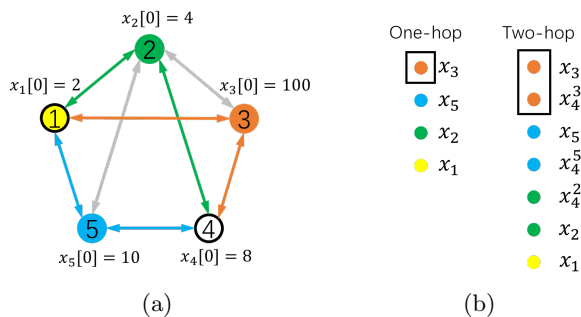


Figure 3.1: (a) A 5-node graph. (b) Values removed by node 1 in one-hop and two-hop algorithms.

hop communication, the essential idea for the normal nodes to avoid being affected by adversary nodes is that each normal node  $i$  will exclude the effects from  $f$  neighbors with extreme values (possibly sent by faulty nodes). This can guarantee that values outside the safety interval will not be used by any normal node at any time step. In the one-hop case, for each normal node  $i$ , the number of values received from such  $f$  neighbors is exactly  $f$ , i.e., node  $i$  will trim away  $f$  largest and  $f$  smallest values at each step. This is because each neighbor sends only one value of its own to node  $i$  at each step under the typical assumptions made in MSR-related works [56; 113].

Under the multi-hop setting, the situation changes significantly even if we assume that each node can only send out one value of its own to its neighbors at each step. Since each node relays the values from different neighbors, normal node  $i$  can receive more than one value from one direct neighbor. Thus, in the MW-MSR algorithm, normal node  $i$  cannot just trim away  $f$  largest and  $f$  smallest values at each step. Instead, it needs to trim away the largest and smallest values from exactly  $f$  nodes within  $l$  hops away, which is the generalization of the essential idea in the one-hop W-MSR algorithm.

To characterize the number of the extreme values from exactly  $f$  nodes for node  $i$ , the notion of minimum message cover (MMC) is designed. Intuitively speaking, for normal node  $i$ ,  $\overline{\mathcal{R}}_i[k]$  and  $\underline{\mathcal{R}}_i[k]$  are the largest sized sets of received messages containing very large and small values that may have been generated or tampered by  $f$  adversary nodes, respectively. Here, we focus on how  $\underline{\mathcal{R}}_i[k]$  is determined, as  $\overline{\mathcal{R}}_i[k]$  can be obtained in a similar way. When the cardinality of the MMC of set  $\underline{\mathcal{M}}_i[k]$  is no more than  $f$ , node  $i$  simply takes  $\underline{\mathcal{R}}_i[k] = \underline{\mathcal{M}}_i[k]$ . Otherwise, node  $i$  will check the first  $f + 1$  values of  $\underline{\mathcal{M}}_i[k]$ , and if the MMC of these values is of cardinality  $f$ , then it will check the first  $f + 2$  values of  $\underline{\mathcal{M}}_i[k]$ . This procedure will continue until for the first  $f + h$  ( $h \geq 1$ )

### 3.3 Robustness with Multi-hop Communication

---

values of  $\underline{\mathcal{M}}_i[k]$ , the MMC of these values is of cardinality  $f + 1$ . Then  $\underline{\mathcal{R}}_i[k]$  is taken as the first  $f + h - 1$  values of  $\underline{\mathcal{M}}_i[k]$ . After sets  $\overline{\mathcal{R}}_i[k]$  and  $\underline{\mathcal{R}}_i[k]$  are determined, in the control input  $u_i(k)$  computed by (3.3) in step 3, values in these sets are excluded. Note that this control is consistent with the one in (3.2) when  $f = 0$ .

We also illustrate the determination of such subsets through a simple example. Consider the network in Fig. 3.1 with initial states  $x[0] = [2 \ 4 \ 100 \ 8 \ 10]^T$ , where node 3 is set to be malicious ( $f = 1$ ) and constantly broadcasts the value 100 as its own value as well as those in the relayed messages. We look at node 1 at time  $k = 0$  and drop the time index  $k$ . In the one-hop version of the MW-MSR algorithm, the input for node 1 is  $\{x_1, x_2, x_5, x_3\}$ , and it chooses  $\overline{\mathcal{R}}_1[0] = \{x_3 = 100\}$  and  $\underline{\mathcal{R}}_1[0] = \emptyset$  in step 2 of the algorithm (since the value  $x_1$  is the smallest in the input).

In the two-hop version of the MW-MSR algorithm, node 1 receives the state values  $x_2, x_3$ , and  $x_5$  directly from nodes 2, 3, and 5, respectively. Moreover, it receives the relayed values of node 4 through nodes 2, 3, and 5, denoted by  $x_4^2, x_4^3$ , and  $x_4^5$ . Then the sorted input for node 1 is  $\{x_1, x_2, x_4^2, x_4^5, x_5, x_4^3, x_3\}$ , and node 1 checks the MMC of the subset of the largest values starting from the  $(f + 1)$ th value (since the values before the  $f$ th one are definitely removed by node 1). First, it evaluates  $\{x_4^3, x_3\}$ , and the MMC of this message set is the node set  $\{3\}$  with cardinality 1. Then, it evaluates  $\{x_5, x_4^3, x_3\}$  and the MMC of this message set can be found to be the node set  $\{3, 5\}$  with cardinality 2, which is bigger than  $f = 1$ . As a result, node 1 confirms that  $\{x_4^3, x_3\}$  is the largest sized set of the large values that may have been generated or tampered by  $f$  adversary nodes. Therefore, node 1 chooses  $\overline{\mathcal{R}}_1[0] = \{x_4^3 = 100, x_3 = 100\}$  and  $\underline{\mathcal{R}}_1[0] = \emptyset$  in step 2 of the algorithm.

In this chapter, the key question to be addressed is, under what conditions on the network can the above algorithm achieve resilient asymptotic consensus? Our approach is to develop a generalization of the results and analysis of the one-hop case. In particular, this necessitates us to extend the notion of graph robustness by taking account of multi-hop communication. This is carried out in the next section.

### 3.3 Robustness with Multi-hop Communication

In this section, we discuss the notion of graph robustness. This notion was first introduced in [56], which corresponds to the one-hop case. We provide its multi-hop generalization, which plays a crucial role in our resilient consensus problem.



### 3.3 Robustness with Multi-hop Communication

---

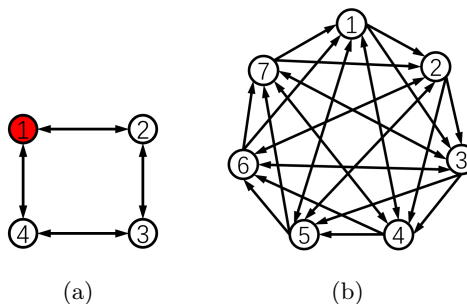


Figure 3.3: (a) The graph is not  $(2, 2)$ -robust with one hop, but it is  $(2, 2)$ -robust with 2 hops. (b) The graph is  $(2, 2)$ -robust with one hop and  $(3, 3)$ -robust with 2 hops.

paths of at most two hops originating from the nodes outside  $\mathcal{V}_1$  with respect to the set  $\mathcal{F}$ . In contrast, in a similar graph shown in Fig. 3.2(b), such a property is lost and node  $i$  has only one path from the outside of  $\mathcal{V}_1$  w.r.t. the set  $\mathcal{F}$ .

Now, we are ready to generalize this notion to the entire graph and define  $r$ -robustness and  $(r, s)$ -robustness with  $l$  hops as follows.

**Definition 3.3.2** *A directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is said to be  $(r, s)$ -robust with  $l$  hops with respect to a given set  $\mathcal{F} \subset \mathcal{V}$ , if for every pair of nonempty disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ , at least one of the following conditions holds:*

1.  $\mathcal{Z}_{\mathcal{V}_1}^r = \mathcal{V}_1$ ,
2.  $\mathcal{Z}_{\mathcal{V}_2}^r = \mathcal{V}_2$ ,
3.  $|\mathcal{Z}_{\mathcal{V}_1}^r| + |\mathcal{Z}_{\mathcal{V}_2}^r| \geq s$ ,

where  $\mathcal{Z}_{\mathcal{V}_a}^r$  is the set of nodes in  $\mathcal{V}_a$  ( $a = 1, 2$ ) that are  $r$ -reachable with  $l$ -hop communication with respect to  $\mathcal{F}$ . Moreover, if the graph  $\mathcal{G}$  satisfies this property with respect to any set  $\mathcal{F}$  following the  $f$ -total model, then we say that  $\mathcal{G}$  is  $(r, s)$ -robust with  $l$  hops under the  $f$ -total model. When it is clear from the context, we will just say  $\mathcal{G}$  is  $(r, s)$ -robust with  $l$  hops. Furthermore, when the graph is  $(r, 1)$ -robust with  $l$  hops, we also say it is  $r$ -robust with  $l$  hops.

Generally, robustness of a graph increases as the relay range  $l$  increases. We will illustrate this point using the graphs in Fig. 3.3. Note that graph robustness with multi-hop communication needs to be checked for every possible set  $\mathcal{F}$  satisfying the

$f$ -total model. In the context of this paper, we are interested to check  $(r, s)$ -robustness under  $(r - 1)$ -total model.

First, the graph in Fig. 3.3(a) is not  $(2, 2)$ -robust with one hop since for the sets  $\{1, 2\}$  and  $\{3, 4\}$ , none of the nodes has two in-neighbors outside the corresponding set. However, under the 1-total model, this graph is  $(2, 2)$ -robust with 2 hops. For instance, we first check the condition for the set  $\mathcal{F} = \{1\}$ . For sets  $\{1, 2\}$  and  $\{3, 4\}$ , all of the nodes 1, 3 and 4 have two independent paths of at most two hops originating from the outside of the set with node 1 not being the internal node. After checking all the possible subsets of  $\mathcal{V}$ , one can confirm that this graph is  $(2, 2)$ -robust with 2 hops with respect to set  $\mathcal{F} = \{1\}$ . Since this graph is actually symmetric for each node, we can conclude that for the set  $\mathcal{F} = \{v\}$  ( $v = 2, 3, 4$ ), this graph is  $(2, 2)$ -robust with 2 hops with respect to this set. Hence, this graph is  $(2, 2)$ -robust with 2 hops.

Next, we look at the graph in Fig. 3.3(b). When  $l = 1$ , this graph is  $(2, 2)$ -robust with 1 hop but not  $(3, 3)$ -robust with 1 hop. When  $l = 2$ , it becomes  $(3, 3)$ -robust with 2 hops. It is further noted that the level of robustness is constrained by the in-degrees of the nodes. In the graph of Fig. 3.3(b), each node has four incoming edges. As a result, for  $r \geq 4$ , this graph cannot be  $(r, s)$ -robust with any number of hops. We elaborate more on this aspect in Section 3.6.

### 3.4 Synchronous Network

In this section, we analyze the MW-MSR algorithm under synchronous updates, i.e., each normal node will update its value using those received from all of its  $l$ -hop neighbors in a synchronous manner with other nodes at each time  $k$ .

#### 3.4.1 Matrix Representation

First, we write the system in a matrix form. For ease of notation in our analysis, reorder the node indices so that the normal nodes take indices  $1, \dots, n_N$  and the malicious nodes are  $n_N + 1, \dots, n$ . Then the state vector and control input vector can be written as

$$x[k] = \begin{bmatrix} x^N[k] \\ x^F[k] \end{bmatrix}, u[k] = \begin{bmatrix} u^N[k] \\ u^F[k] \end{bmatrix}, \quad (3.4)$$

where the superscript  $N$  stands for normal and  $F$  for faulty. Regarding the control inputs  $u^N[k]$  and  $u^F[k]$ , the normal nodes follow (3.3) while the malicious nodes may

not. Hence, they can be expressed as

$$\begin{aligned} u^N[k] &= -L^N[k]x[k], \\ u^F[k] &: \text{arbitrary}, \end{aligned} \quad (3.5)$$

where  $L^N[k] \in \mathbb{R}^{n_N \times n}$  is the matrix formed by the first  $n_N$  rows of  $L[k]$  associated with normal nodes. The row sums of this matrix  $L^N[k]$  are zero as in  $L[k]$ . Thus, we can rewrite the system as

$$x[k+1] = \left( I_n - \begin{bmatrix} L^N[k] \\ 0 \end{bmatrix} \right) x[k] + \begin{bmatrix} 0 \\ I_{n_F} \end{bmatrix} u^F[k]. \quad (3.6)$$

### 3.4.2 Consensus Analysis with Multi-hop Communication

Now we are ready to provide a necessary and sufficient condition for resilient consensus applying the synchronous MW-MSR algorithm. The following theorem is the first main contribution of this chapter.

**Theorem 3.4.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the synchronous MW-MSR algorithm with parameter  $f$ . Under the  $f$ -total malicious model, resilient asymptotic consensus is achieved if and only if the network topology is  $(f+1, f+1)$ -robust with  $l$  hops. Moreover, the safety interval is given by*

$$\mathcal{S} = [\min x^N[0], \max x^N[0]].$$

*Proof: (Necessity)* If  $\mathcal{G}$  is not  $(f+1, f+1)$ -robust with  $l$  hops, then there are nonempty, disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$  such that none of the conditions in Definition 3.3.2 holds. Suppose that the initial value of each node in  $\mathcal{V}_1$  is  $a$  and each node in  $\mathcal{V}_2$  takes  $b$ , with  $a < b$ . Let all other nodes have initial values taken from the interval  $(a, b)$ . Since  $|\mathcal{Z}_{\mathcal{V}_1}^{f+1}| + |\mathcal{Z}_{\mathcal{V}_2}^{f+1}| \leq f$ , suppose that all nodes in  $\mathcal{Z}_{\mathcal{V}_1}^{f+1}$  and  $\mathcal{Z}_{\mathcal{V}_2}^{f+1}$  are malicious and take constant values. Then there is still at least one normal node in both  $\mathcal{V}_1$  and  $\mathcal{V}_2$  since  $|\mathcal{Z}_{\mathcal{V}_1}^{f+1}| < |\mathcal{V}_1|$  and  $|\mathcal{Z}_{\mathcal{V}_2}^{f+1}| < |\mathcal{V}_2|$ , respectively. Then these normal nodes remove all the values of incoming neighbors outside of their respective sets since the message cover of these values has cardinality equal to  $f$  or less. According to the synchronous MW-MSR algorithm, such normal nodes will keep their values and consensus cannot

be achieved.

(*Sufficiency*) First, we show that the safety condition of resilient consensus is satisfied. Let  $\bar{x}^N[k]$  and  $\underline{x}^N[k]$  to be the maximum and minimum values of the normal nodes at time  $k$ , respectively. We can show that  $\bar{x}^N[k]$  is monotonically nonincreasing and  $\underline{x}^N[k]$  is monotonically nondecreasing, and thus each of them has some limit. This can be directly shown from the definitions and the facts that the values used in the MW-MSR update rule always lie within the interval  $[\underline{x}^N[k], \bar{x}^N[k]] \subseteq \mathcal{S}$  for  $k \geq 0$ . Since at each time  $k$ , in step 2 of Algorithm 1, node  $i$  wipes out the possibly manipulated values from at most  $f$  nodes within  $l$  hops. Moreover, the update rule (3.6) uses a convex combination of the values in  $[\underline{x}^N[k], \bar{x}^N[k]]$ . Therefore, the safety condition is satisfied.

Then, we denote the limits of  $\bar{x}^N[k]$  and  $\underline{x}^N[k]$  by  $\bar{\omega}$  and  $\underline{\omega}$ , respectively. We will prove by contradiction to show that  $\bar{\omega} = \underline{\omega}$ , and thus the normal nodes will reach consensus. Suppose that  $\bar{\omega} > \underline{\omega}$ . We can then take  $\epsilon_0 > 0$  such that  $\bar{\omega} - \epsilon_0 > \underline{\omega} + \epsilon_0$ . Fix  $\epsilon < \epsilon_0 \alpha^{n_N} / (1 - \alpha^{n_N})$ , where  $0 < \epsilon < \epsilon_0$  and  $\alpha$  is the minimum of all  $a_i[k]$  in step 3 of the MW-MSR algorithm. For  $1 \leq \gamma \leq n_N$ , define  $\epsilon_\gamma$  recursively as

$$\epsilon_\gamma = \alpha \epsilon_{\gamma-1} - (1 - \alpha)\epsilon.$$

So we have  $0 < \epsilon_\gamma < \epsilon_{\gamma-1} \leq \epsilon_0$  for all  $\gamma$ , since it holds that

$$\begin{aligned} \epsilon_\gamma &= \alpha \epsilon_{\gamma-1} - (1 - \alpha)\epsilon = \alpha^\gamma \epsilon_0 - (1 - \alpha^\gamma)\epsilon \\ &\geq \alpha^{n_N} \epsilon_0 - (1 - \alpha^{n_N})\epsilon > 0. \end{aligned} \tag{3.7}$$

At any time step  $k$  and for any  $\epsilon_t > 0$ , define two sets:

$$\begin{aligned} \mathcal{Z}_1(k, \epsilon_t) &= \{i \in \mathcal{V} : x_i[k] > \bar{\omega} - \epsilon_t\}, \\ \mathcal{Z}_2(k, \epsilon_t) &= \{i \in \mathcal{V} : x_i[k] < \underline{\omega} + \epsilon_t\}. \end{aligned}$$

By the definition of  $\epsilon_0$ ,  $\mathcal{Z}_1(k, \epsilon_0)$  and  $\mathcal{Z}_2(k, \epsilon_0)$  are disjoint.

Let  $k_\epsilon$  be the time such that  $\bar{x}^N[k] < \bar{\omega} + \epsilon$  and  $\underline{x}^N[k] > \underline{\omega} - \epsilon$ ,  $\forall k \geq k_\epsilon$ . Such a  $k_\epsilon$  exists since  $\bar{x}^N[k]$  and  $\underline{x}^N[k]$  converge to  $\bar{\omega}$  and  $\underline{\omega}$ , respectively, in monotonic manners as discussed above. Consider the nonempty and disjoint sets  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$  and  $\mathcal{Z}_2(k_\epsilon, \epsilon_0)$ . Notice that the network is  $(f + 1, f + 1)$ -robust with  $l$  hops w.r.t. any set  $\mathcal{F}$  following the  $f$ -total model and the set of malicious nodes  $\mathcal{A}$  also satisfies the  $f$ -total model.

Hence, the network is  $(f + 1, f + 1)$ -robust with  $l$  hops w.r.t. the set  $\mathcal{A}$  and at least one of the three conditions in Definition 3.3.2 holds. Also notice that the normal node with value  $\bar{x}^N[k_\epsilon]$  will definitely be in set  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$ , and it is similar for the case of  $\mathcal{Z}_2(k_\epsilon, \epsilon_0)$ . Hence, all nodes in either  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$  or  $\mathcal{Z}_2(k_\epsilon, \epsilon_0)$  have the  $(f + 1)$ -reachable property, or the union of the two sets contains at least  $f + 1$  nodes having the  $(f + 1)$ -reachable property. Since there are at most  $f$  malicious nodes, for all cases, there must exist a normal node in the union of  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$  and  $\mathcal{Z}_2(k_\epsilon, \epsilon_0)$  such that it has at least  $f + 1$  independent paths originating from different nodes outside of its set and these paths do not have any internal node in  $\mathcal{A}$ .

Suppose that normal node  $i \in \mathcal{Z}_1(k_\epsilon, \epsilon_0) \cap \mathcal{N}$  has the  $(f + 1)$ -reachable property. Thus, node  $i$  has at least  $f + 1$  neighbors within  $l$  hops outside set  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$ , i.e., the values of these neighbors are smaller than  $x_i[k_\epsilon]$  and are at most equal to  $\bar{\omega} - \epsilon_0$ . Moreover, the original values of these multi-hop neighbors of node  $i$  will definitely reach node  $i$  even if the source nodes are malicious (since the internal nodes of these paths are all normal and they relay the values as received, without making any changes). Hence, node  $i$  will use at least one of these values to update its own. This is because in step 2(c) of Algorithm 1, node  $i$  will remove values lower than its own value of which the cardinality of the minimum message cover is at most  $f$ . As a result, among the neighbors within  $l$  hops outside set  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$ , the values from up to  $f$  of them will be disregarded by node  $i$ .

Now, in Algorithm 1, the update rule (3.3) of step 3 is applied. Here, each coefficient of the neighbors is lower bounded by  $\alpha$ . Since the largest value that node  $i$  will use at time  $k_\epsilon$  is  $\bar{x}^N[k_\epsilon]$ , placing the largest possible weight on  $\bar{x}^N[k_\epsilon]$  produces

$$\begin{aligned} x_i[k_\epsilon + 1] &\leq (1 - \alpha)\bar{x}^N[k_\epsilon] + \alpha(\bar{\omega} - \epsilon_0) \\ &\leq (1 - \alpha)(\bar{\omega} + \epsilon) + \alpha(\bar{\omega} - \epsilon_0) \leq \bar{\omega} - \alpha\epsilon_0 + (1 - \alpha)\epsilon. \end{aligned}$$

Note that this upper bound also applies to the updated value of any normal node not in  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$ , because such a node will use its own value in its update. Similarly, if node  $i \in \mathcal{Z}_2(k_\epsilon, \epsilon_0) \cap \mathcal{N}$  has the  $(f + 1)$ -reachable property, then  $x_i[k_\epsilon + 1] \geq \underline{\omega} + \alpha\epsilon_0 - (1 - \alpha)\epsilon$ . Again, any normal node not in  $\mathcal{Z}_2(k_\epsilon, \epsilon_0)$  will have the same lower bound.

Next, consider the sets  $\mathcal{Z}_1(k_\epsilon + 1, \epsilon_1)$  and  $\mathcal{Z}_2(k_\epsilon + 1, \epsilon_1)$ . By  $\epsilon_1 < \epsilon_0$ , these two sets are still disjoint. Since at least one of the normal nodes in  $\mathcal{Z}_1(k_\epsilon, \epsilon_0)$  decreases at least to  $\bar{\omega} - \epsilon_1$  (or below), or one of the nodes in  $\mathcal{Z}_2(k_\epsilon, \epsilon_0)$  increases at least to  $\underline{\omega} + \epsilon_1$  (or above), it



must hold  $|\mathcal{Z}_1(k_\epsilon + 1, \epsilon_1) \cap \mathcal{N}| < |\mathcal{Z}_1(k_\epsilon, \epsilon_0) \cap \mathcal{N}|$ ,  $|\mathcal{Z}_2(k_\epsilon + 1, \epsilon_1) \cap \mathcal{N}| < |\mathcal{Z}_2(k_\epsilon, \epsilon_0) \cap \mathcal{N}|$ , or both. Recall that  $0 < \epsilon_\gamma < \epsilon_{\gamma-1} \leq \epsilon_0$ . As long as there are still normal nodes in  $\mathcal{Z}_1(k_\epsilon + \gamma, \epsilon_\gamma)$  and/or  $\mathcal{Z}_2(k_\epsilon + \gamma, \epsilon_\gamma)$ , we can repeat the above analysis for time step  $k_\epsilon + \gamma$ , which will result in either  $|\mathcal{Z}_1(k_\epsilon + \gamma, \epsilon_\gamma) \cap \mathcal{N}| < |\mathcal{Z}_1(k_\epsilon + \gamma - 1, \epsilon_{\gamma-1}) \cap \mathcal{N}|$ ,  $|\mathcal{Z}_2(k_\epsilon + \gamma, \epsilon_\gamma) \cap \mathcal{N}| < |\mathcal{Z}_2(k_\epsilon + \gamma - 1, \epsilon_{\gamma-1}) \cap \mathcal{N}|$ , or both.

Since  $|\mathcal{Z}_1(k_\epsilon, \epsilon_0) \cap \mathcal{N}| + |\mathcal{Z}_2(k_\epsilon, \epsilon_0) \cap \mathcal{N}| \leq n_N$ , there must be some time step  $k_\epsilon + T$  (with  $T \leq n_N$ ) such that either  $\mathcal{Z}_1(k_\epsilon + T, \epsilon_T) \cap \mathcal{N}$  or  $\mathcal{Z}_2(k_\epsilon + T, \epsilon_T) \cap \mathcal{N}$  is empty. In the former case, all normal nodes in the network at time step  $k_\epsilon + T$  have values at most  $\bar{\omega} - \epsilon_T$ , while in the latter case all normal nodes at time step  $k_\epsilon + T$  have values no less than  $\underline{\omega} + \epsilon_T$ . By (3.7) and  $T \leq n_N$ , it holds that  $\epsilon_T > 0$ . Hence, we have contradiction to the fact that the largest value monotonically converges to  $\bar{\omega}$  (in the former case) or that the smallest value monotonically converges to  $\underline{\omega}$  (in the latter case). Hence, it must be the case that  $\epsilon_0 = 0$ , proving that  $\bar{\omega} = \underline{\omega}$ .  $\blacksquare$

We emphasize that the graph condition based on the notion of robustness with  $l$  hops is tight for our MW-MSR algorithm. Our notion captures the capability of agents to be influenced by the outside of the set in the multi-hop settings. We note that in [102], an idea similar to robustness is proposed, and based on it, a tight necessary and sufficient condition for Byzantine consensus using an MSR-type algorithm with multi-hop communication is provided. However, the focus there is on the Byzantine model and the condition is expressed in terms of the subgraph consisting of only the normal nodes. Part of the reason to focus on only the subgraph of normal nodes is that Byzantine nodes are more adversarial compared to malicious nodes as they can send different values to different neighbors. Hence, the subgraph of normal nodes has to be sufficiently robust to fight against the possible attacks. The condition there is an extension of the one for the one-hop case shown in [113]. Moreover, to meet the condition there, each node in  $\mathcal{G}$  must have at least  $2f + 1$  incoming edges. This is different from the case for the malicious model studied in this paper, where at least  $2f$  incoming edges are required. Further discussions on the minimum requirement for our algorithm to guarantee resilient consensus are given in Section 3.6.

### 3.5 Asynchronous Network

In this section, we analyze the MW-MSR algorithm under asynchronous updates with time delays in the communication among nodes.

We employ the control input taking account of possible delays in the values from the neighbors as

$$u_i[k] = \sum_{j \in \mathcal{N}_i^{l-}} a_{ij}[k] x_j^P[k - \tau_{ij}^P[k]], \quad (3.8)$$

where  $x_j^P[k]$  denotes the value of node  $j$  at time  $k$  sent along path  $P$  and  $\tau_{ij}^P[k] \in \mathbb{Z}_+$  is the delay in this  $(j, i)$ -path  $P$  at time  $k$ . The delays are time varying and may be different in each path, but we assume the common upper bound  $\tau$  on any *normal* path  $P$ , over which all internal nodes are normal, as

$$0 \leq \tau_{ij}^P[k] \leq \tau, \quad j \in \mathcal{N}_i^{l-}, \quad k \in \mathbb{Z}_+. \quad (3.9)$$

Hence, each normal node  $i$  becomes aware of the value of each of its normal  $l$ -hop neighbor  $j$  on each normal  $(j, i)$ -path  $P$  at least once in  $\tau$  time steps, but possibly at different time instants [21]. Although we impose this bound on the delays for transmission of messages, the normal nodes need neither the value of this bound nor the information that whether a path  $P$  is a normal one or not.

The structure of the asynchronous MW-MSR algorithm can be outlined as follows. At each time  $k$ , each normal node  $i$  chooses to update or not. If it chooses not to update, then it keeps its value as  $x_i[k+1] = x_i[k]$ . Otherwise, it uses the most recently received values of all its  $l$ -hop neighbors on each  $l$ -hop path to update its value using the MW-MSR algorithm in Algorithm 1. Like the one-hop MSR algorithm, if node  $i$  does not receive any value along some path  $P$  originating from its  $l$ -hop neighbor  $j$  (i.e., the crash model), then node  $i$  will take this value on path  $P$  as an empty value and will discard this value when it applies the WM-MSR algorithm. As we discussed earlier in Section II-E, in the asynchronous case also, manipulating message paths is equivalent to manipulating message values only and hence can be disregarded in our analysis.

Let  $D[k]$  be a diagonal matrix whose  $i$ th entry is given by  $d_i[k] = \sum_{j=1}^n a_{ij}[k]$ . Then, let the matrices  $A_\gamma[k] \in \mathbb{R}^{n \times n}$  for  $0 \leq \gamma \leq \tau$  and  $L_\tau[k] \in \mathbb{R}^{n \times (\tau+1)n}$  be given by

$$A_\gamma[k] = \begin{cases} a_{ij}[k] & \text{if } i \neq j \text{ and } \tau_{ij}[k] = \gamma, \\ 0 & \text{otherwise,} \end{cases} \quad (3.10)$$

and  $L_\tau[k] = [D[k] - A_0[k] - A_1[k] \cdots - A_\tau[k]]$ , respectively. Note that the summation of each row of  $L_\tau[k]$  is zero. The delay  $\tau_{ij}[k]$  will be set to be one of the delays  $\tau_{ij}^P[k]$

corresponding to the normal paths as we discuss further later.

Now, the control input can be expressed as

$$\begin{aligned} u^N[k] &= -L_\tau^N[k]z[k], \\ u^F[k] &: \text{arbitrary}, \end{aligned} \quad (3.11)$$

where  $z[k] = [x[k]^T x[k-1]^T \dots x[k-\tau]^T]^T$  is a  $(\tau+1)n$ -dimensional vector for  $k \geq 0$  and  $L_\tau^N[k]$  is a matrix formed by the first  $n_N$  rows of  $L_\tau[k]$ . Here, to simplify the discussion, we assume that  $z[0]$  consists of the given initial values of the agents. Then, the agent dynamics can be written as

$$x[k+1] = \Gamma[k]z[k] + \begin{bmatrix} 0 \\ I_{n_F} \end{bmatrix} u^F[k], \quad (3.12)$$

where  $\Gamma[k]$  is an  $n \times (\tau+1)n$  matrix given by  $\Gamma[k] = [I_n \ 0] - [L_\tau^N[k]^T \ 0]^T$ .

The main result of this section now follows. Here, the safety interval differs from the synchronous case and is given by

$$\mathcal{S}_\tau = \left[ \min z^N[0], \max z^N[0] \right]. \quad (3.13)$$

**Theorem 3.5.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous MW-MSR algorithm with parameter  $f$ . Under the  $f$ -total malicious model, resilient asymptotic consensus is achieved only if the underlying graph is  $(f+1, f+1)$ -robust with  $l$  hops. Moreover, if the underlying graph is  $(2f+1)$ -robust with  $l$  hops, then resilient consensus is attained with the safety interval given by (3.13).*

*Proof:* (Necessity) Since the synchronous algorithm is a special case of the asynchronous algorithm, the necessary condition in Theorem 3.4.1 also holds here.

(Sufficiency) First, we show the safety condition. For  $k=0$ , by the assumption on  $z[0]$ , it holds  $z^N[0] \in \mathcal{S}_\tau$ , and thus  $x_i[0] \in \mathcal{S}_\tau, \forall i \in \mathcal{N}$ . Next, for  $k \geq 0$ , let  $\bar{x}_\tau^N[k]$  and  $\underline{x}_\tau^N[k]$  be the largest value and the smallest value, respectively, of the normal agents from time  $k, k-1, \dots, k-\tau$ . That is,

$$\begin{aligned} \bar{x}_\tau^N[k] &= \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ \underline{x}_\tau^N[k] &= \min(x^N[k], x^N[k-1], \dots, x^N[k-\tau]). \end{aligned} \quad (3.14)$$

Then we prove that  $\bar{x}_\tau^N[k]$  is a nonincreasing function of  $k \geq 0$ . By (3.12), at time  $k \geq 0$ , each normal agent updates its value based on a convex combination of the neighbors' values from  $k$  to  $k - \tau$ . Moreover, the values outside of the interval determined by the normal agents' values  $[\underline{x}_\tau^N[k], \bar{x}_\tau^N[k]]$  will be ignored by step 2 of the MW-MSR algorithm. This is because in this step, node  $i$  will remove the largest sized subsets of large and small values that can be manipulated by at most  $f$  nodes within  $l$  hops. Hence, we obtain  $x_i[k+1] \leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau])$  for any  $i \in \mathcal{N}$ . We also have

$$\begin{aligned} x_i[k] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ x_i[k-1] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ &\vdots \\ x_i[k+1-\tau] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) \end{aligned}$$

for any  $i \in \mathcal{N}$ . Therefore,  $\bar{x}_\tau^N[k]$  is nonincreasing in time as

$$\begin{aligned} \bar{x}_\tau^N[k+1] &= \max(x^N[k+1], x^N[k], \dots, x^N[k+1-\tau]) \\ &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) = \bar{x}_\tau^N[k]. \end{aligned}$$

We can similarly prove that  $\underline{x}_\tau^N[k]$  is nondecreasing in time. This indicates that for  $k \geq 0$ , we have  $x_i[k] \in \mathcal{S}_\tau, \forall i \in \mathcal{N}$ . Thus, we have shown the safety condition.

Next, we show the convergence. As shown above,  $\bar{x}_\tau^N[k]$  and  $\underline{x}_\tau^N[k]$  are monotonically decreasing and increasing, respectively, and moreover bounded. Thus, both of their limits exist and are denoted by  $\bar{\omega}_\tau$  and  $\underline{\omega}_\tau$ , respectively. We claim that the limits satisfy  $\bar{\omega}_\tau = \underline{\omega}_\tau$ , i.e., consensus is achieved. We prove by contradiction and assume that  $\bar{\omega}_\tau > \underline{\omega}_\tau$ .

Recall that  $\alpha$  lower bounds the nonzero entries of  $\Gamma[k]$ . Choose  $\epsilon_0 > 0$  small enough that  $\bar{\omega}_\tau - \epsilon_0 > \underline{\omega}_\tau + \epsilon_0$ . Fix

$$\epsilon < \frac{\epsilon_0 \alpha^{(\tau+1)n_N}}{(1 - \alpha^{(\tau+1)n_N})}, \quad 0 < \epsilon < \epsilon_0. \quad (3.15)$$

Define the sequence  $\{\epsilon_\gamma\}$  by

$$\epsilon_{\gamma+1} = \alpha \epsilon_\gamma - (1 - \alpha)\epsilon, \quad \gamma = 0, 1, \dots, (\tau+1)n_N - 1.$$

So we have  $0 < \epsilon_{\gamma+1} < \epsilon_\gamma$  for all  $\gamma$ . In particular, they are positive because by (3.15), it holds that

$$\begin{aligned}\epsilon_{(\tau+1)n_N} &= \alpha^{(\tau+1)n_N} \epsilon_0 - \sum_{m=0}^{(\tau+1)n_N-1} \alpha^m (1-\alpha) \epsilon \\ &= \alpha^{(\tau+1)n_N} \epsilon_0 - (1 - \alpha^{(\tau+1)n_N}) \epsilon > 0.\end{aligned}$$

Take  $k_\epsilon \in \mathbb{Z}_+$  such that  $\bar{x}_\tau^N[k] < \bar{\omega}_\tau + \epsilon$  and  $\underline{x}_\tau^N[k] > \underline{\omega}_\tau - \epsilon$  for  $k \geq k_\epsilon$ . Such  $k_\epsilon$  exists due to the convergence of  $\bar{x}_\tau^N[k]$  and  $\underline{x}_\tau^N[k]$ . Then we can define the two disjoint sets as

$$\begin{aligned}\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma) &= \{j \in \mathcal{N} : x_j[k_\epsilon + \gamma] > \bar{\omega}_\tau - \epsilon_\gamma\}, \\ \mathcal{Z}_{2\tau}(k_\epsilon + \gamma, \epsilon_\gamma) &= \{j \in \mathcal{N} : x_j[k_\epsilon + \gamma] < \underline{\omega}_\tau + \epsilon_\gamma\}.\end{aligned}$$

Next, we show that one of the two sets becomes empty in a finite number of steps, which contradicts the assumption on  $\bar{\omega}_\tau$  and  $\underline{\omega}_\tau$  being the limits. Consider the set  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$ . Due to the definition of  $\bar{x}_\tau^N[k]$  and its limit  $\bar{\omega}_\tau$ , one or more normal nodes are contained in the union of the sets  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$  for  $0 \leq \gamma \leq \tau + 1$ . We claim that  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  is in fact nonempty. To prove this, it is sufficient to show that if a normal node  $j$  is not in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$ , then it is not in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma + 1, \epsilon_{\gamma+1})$  for  $\gamma = 0, \dots, \tau$ .

Suppose that node  $j$  satisfies  $x_j[k_\epsilon + \gamma] \leq \bar{\omega}_\tau - \epsilon_\gamma$ . Every normal node updates its value to a convex combination of the multi-hop neighbors' values at the current or previous times. Moreover, the values greater than  $\bar{x}_\tau^N[k_\epsilon + \gamma]$  are ignored in step 2 of the MW-MSR algorithm. Hence, the value of node  $j$  at the next time step is upper bounded as

$$\begin{aligned}x_j[k_\epsilon + \gamma + 1] &\leq (1 - \alpha) \bar{x}_\tau^N[k_\epsilon + \gamma] + \alpha(\bar{\omega}_\tau - \epsilon_\gamma) \\ &\leq (1 - \alpha)(\bar{\omega}_\tau + \epsilon) + \alpha(\bar{\omega}_\tau - \epsilon_\gamma) \\ &\leq \bar{\omega}_\tau - \alpha\epsilon_\gamma + (1 - \alpha)\epsilon = \bar{\omega}_\tau - \epsilon_{\gamma+1}.\end{aligned}\tag{3.16}$$

It thus follows that node  $j$  is not in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma + 1, \epsilon_{\gamma+1})$ . This means that the cardinality of the set  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$  is nonincreasing for  $\gamma = 0, \dots, \tau + 1$ . The same holds for  $\mathcal{Z}_{2\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$ , and hence  $\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)$  is nonempty too.

We next show that one of these two sets in fact becomes empty in finite time. Since the graph is  $(2f + 1)$ -robust with  $l$  hops w.r.t. any set  $\mathcal{F}$  satisfying the  $f$ -total model, the graph is also  $(2f + 1)$ -robust with  $l$  hops w.r.t. set  $\mathcal{A}$  (i.e., the set of adversarial nodes). Therefore, between the two nonempty disjoint sets  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  and  $\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)$ , one of them has a normal agent with at least  $2f + 1$  independent paths originating from

the nodes outside and these paths do not have any internal node in the set  $\mathcal{A}$ .

Suppose that normal node  $i \in \mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  has this property. Since there are at most  $f$  malicious nodes and node  $i$  can only remove the values of which the cardinality of the minimum message cover is  $f$ . Moreover, node  $i$  is supposed to update once in at most  $\tau$  time steps. Therefore, when node  $i$  makes an update at time  $k_\epsilon + \tau$ , it will use at least one delayed value from the normal nodes outside the set  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$ , upper bounded by  $\bar{\omega}_\tau - \epsilon_\tau$ . It thus follows that, at time  $k_\epsilon + \tau$ , when node  $i$  makes an update, its value can be bounded as

$$x_i[k_\epsilon + \tau + 1] \leq (1 - \alpha)\bar{x}_\tau^N[k_\epsilon + \tau] + \alpha(\bar{\omega}_\tau - \epsilon_\tau).$$

By (3.16), we have  $x_i[k_\epsilon + \tau + 1] \leq \bar{\omega}_\tau - \epsilon_{\tau+1}$ . We can conclude that if node  $i$  in  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  has  $2f + 1$  independent paths originating from the nodes outside the set, then it goes outside of  $\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  after  $\tau + 1$  steps. Consequently,  $|\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})| < |\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)|$ . Likewise, it follows that if  $\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)$  has a node having at least  $2f + 1$  independent paths originating from the nodes outside, then  $|\mathcal{Z}_{2\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})| < |\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)|$ .

Since there are only  $n_N$  normal nodes, we can repeat the steps above until one of the sets  $\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  and  $\mathcal{Z}_{2\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  becomes empty, and it takes no more than  $(\tau + 1)n_N$  steps. Once the set becomes empty, it remains so indefinitely. This contradicts the assumption that  $\bar{\omega}_\tau$  and  $\underline{\omega}_\tau$  are the limits. Therefore, we obtain  $\bar{\omega}_\tau = \underline{\omega}_\tau$ . ■

**Remark 3.5.1** *In comparison to the synchronous update case studied in the previous section, the graph condition to achieve resilient consensus under the asynchronous updates with delays is more restrictive. This is because when the nodes updates asynchronously, the normal nodes may not receive the same values from the malicious nodes, which creates a more adversarial situation for resilient consensus. Moreover, in the next section, in Lemma 3.6.1, we prove that under the  $f$ -total model, a graph which is  $(2f + 1)$ -robust with  $l$  hops is also  $(f + 1, f + 1)$ -robust with  $l$  hops. For instance, the graph in Fig. 3.4 is 3-robust with 2 hops and hence  $(2, 2)$ -robust with 2 hops. Thus, as expected, the sufficient condition for the asynchronous algorithm guaranteeing resilient consensus is also sufficient for the synchronous algorithm.*

**Remark 3.5.2** *The difference in the network requirements discussed above for the syn-*

### 3.6 Discussions on Graph Robustness with Multi-hop Communication

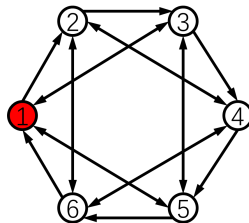


Figure 3.4: The graph is not 2-robust with one hop, but it is 3-robust with 2 hops and (2, 2)-robust with 2 hops.

*chronous and asynchronous algorithms is a consequence partly due to the deterministic nature in transmission times. In [22], it is shown that for an asynchronous algorithm with one-hop communication without delays, we can recover the tight necessary and sufficient network condition of the synchronous case, i.e., the graph to be  $(f + 1, f + 1)$ -robust under the  $f$ -total malicious model. It is interesting that this result requires randomization in agents' communication instants whereas in the deterministic case, the sufficient graph condition remains to be  $(2f + 1)$ -robustness, which coincides with the implication of Theorem 3.5.1. In the multi-hop setting, however, delays are critical and hence we do not pursue such results in this paper.*

## 3.6 Discussions on Graph Robustness with Multi-hop Communication

In this section, we demonstrate some properties of graph robustness with  $l$  hops, which generalize the properties of robustness with one hop in [56] when  $l = 1$  for this new notion. Moreover, we provide the analysis of graph robustness with  $l$  hops for the case where  $l$  is sufficiently large. This corresponds to the case of unbounded path length.

### 3.6.1 Properties of Robustness with 1 Hops

As discussed earlier, our definition of robustness with  $l$  hops is a generalization of the definition of robustness with one-hop communication from [56]. Here, we are interested in investigating how properties of robustness with the one-hop case can be extended to the multi-hop case. In what follows, we present a series of lemmas that analyze the generalized notion of robustness. Recall that robustness with  $l$  hops is defined w.r.t. the given set  $\mathcal{F}$  satisfying the  $f$ -total model, but we omit saying this when it is clear from

### 3.6 Discussions on Graph Robustness with Multi-hop Communication

---

the context in this section. Typically, we are interested in  $f = r - 1$  for  $r$ -robustness.

The first result is simple, stating that  $(r, s)$ -robustness with  $l$  hops of a graph holds with smaller  $r$  and  $s$ .

**Lemma 3.6.1** *If a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is  $(r, s)$ -robust with  $l$  hops, then it is also  $(r', s')$ -robust with  $l$  hops when  $0 \leq r' \leq r$  and  $1 \leq s' \leq s$ .*

In the second result, we show that the level of robustness of a given graph with  $l$  hops does not decrease by adding edges to the graph nor by increasing the relay range  $l$ .

**Lemma 3.6.2** *Suppose that a directed subgraph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  of  $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$  is  $(r, s)$ -robust with  $l$  hops, where  $\mathcal{E} \subseteq \mathcal{E}'$ . Then  $\mathcal{G}'$  is  $(r, s)$ -robust with  $l$  hops. Moreover,  $\mathcal{G}$  is  $(r, s)$ -robust with  $l'$  hops, where  $l' \geq l$ .*

The maximum robustness with  $l$  hops for a graph consisting of  $n$  nodes is the same as that with the one-hop case. The following lemma suggests that the bound  $n \geq 2f + 1$  cannot be breached by introducing multi-hop communication to the MSR algorithms. This bound is dependent on the nature of the MSR algorithms, which cannot tolerate half or more of the agents to be adversarial.

**Lemma 3.6.3** *No directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  on  $n$  nodes is  $(\lceil n/2 \rceil + 1)$ -robust with  $l$  hops. Moreover, the complete graph  $\mathcal{K}_n$  is  $(\lceil n/2 \rceil, s)$ -robust with  $l$  hops for  $1 \leq s \leq n$ .*

*Proof:* We consider the nontrivial case with  $n \geq 3$ . Then pick  $\mathcal{V}_1$  and  $\mathcal{V}_2$  by taking any bipartition of  $\mathcal{V}$  (i.e.,  $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$  and  $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ ) such that  $|\mathcal{V}_1| = \lceil n/2 \rceil$  and  $|\mathcal{V}_2| = \lfloor n/2 \rfloor$ . Neither  $\mathcal{V}_1$  nor  $\mathcal{V}_2$  has  $\lceil n/2 \rceil + 1$  nodes; thus, neither one has a node being  $(\lceil n/2 \rceil + 1)$ -reachable with  $l$ -hop communication. Hence,  $\mathcal{G}$  is not  $(\lceil n/2 \rceil + 1)$ -robust with  $l$  hops.

The proof for the complete graph case follows an analysis similar to the one for the one-hop case [56]. ■

The following lemma exposes that the notion of  $(r, s)$ -robust graphs has a more complicated structure in the relation between the two parameters  $r$  and  $s$ .

**Lemma 3.6.4** *If a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is  $(r, s)$ -robust with  $l$  hops under the  $f$ -total model, then it is also  $(r - 1, s + 1)$ -robust with  $l$  hops under the  $f$ -total model.*



### 3.6 Discussions on Graph Robustness with Multi-hop Communication

---

*Proof:* Consider any nonempty disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ . If  $|\mathcal{Z}_{\mathcal{V}_a}^r| = |\mathcal{V}_a|$  or  $|\mathcal{Z}_{\mathcal{V}_a}^{r-1}| = |\mathcal{V}_a|$  for  $a = 1, 2$ , then this pair of subsets satisfies conditions 1) or 2) of  $(r - 1, s + 1)$ -robustness with  $l$  hops. Thus, assume  $|\mathcal{Z}_{\mathcal{V}_a}^r| < |\mathcal{V}_a|$  and  $|\mathcal{Z}_{\mathcal{V}_a}^{r-1}| < |\mathcal{V}_a|$  for  $a = 1, 2$ . Then condition 3) for  $(r, s)$ -robustness with  $l$  hops must be satisfied, i.e.,  $|\mathcal{Z}_{\mathcal{V}_1}^r| + |\mathcal{Z}_{\mathcal{V}_2}^r| \geq s$ .

Since  $s \geq 1$ , at least one of  $\mathcal{Z}_{\mathcal{V}_1}^r, \mathcal{Z}_{\mathcal{V}_2}^r$  is nonempty. Suppose that  $\mathcal{Z}_{\mathcal{V}_1}^r$  is nonempty. Then, we choose  $i \in \mathcal{Z}_{\mathcal{V}_1}^r$ . Pick the new pair of nonempty disjoint subsets as  $\mathcal{V}'_1 = \mathcal{V}_1 \setminus \{i\}$  and  $\mathcal{V}'_2 = \mathcal{V}_2$ . Observe that if  $j \in \mathcal{Z}_{\mathcal{V}'_1}^r$  then  $j \in \mathcal{Z}_{\mathcal{V}_1}^{r-1}$ . Because node  $i$  is the only difference between  $\mathcal{V}'_1$  and  $\mathcal{V}_1$ , and node  $i$  can only be part of one independent path whose destination is node  $j$ . Consequently, even if node  $i$  is on one independent path whose destination is node  $j$ , node  $j$  still has the  $(r - 1)$ -reachable property, i.e., it has at least  $r - 1$  other independent paths of at most  $l$  hops originating from nodes outside  $\mathcal{V}_1$  and these paths do not have any internal nodes from set  $\mathcal{F}$ . Then, by adding  $i \in \mathcal{Z}_{\mathcal{V}'_1}^r \subseteq \mathcal{Z}_{\mathcal{V}_1}^{r-1}$  back into the set  $\mathcal{V}_1$ , we have

$$|\mathcal{Z}_{\mathcal{V}'_1}^{r-1}| \geq |\mathcal{Z}_{\mathcal{V}'_1}^r| + 1. \quad (3.17)$$

Moreover,  $|\mathcal{Z}_{\mathcal{V}'_1}^r| < |\mathcal{V}'_1|$  and  $|\mathcal{Z}_{\mathcal{V}'_2}^r| < |\mathcal{V}'_2|$  (since  $\mathcal{V}'_2 = \mathcal{V}_2$ ). The first inequality holds because if  $|\mathcal{Z}_{\mathcal{V}'_1}^r| = |\mathcal{V}'_1|$  and from the observation we just proved (if  $j \in \mathcal{Z}_{\mathcal{V}'_1}^r$  then  $j \in \mathcal{Z}_{\mathcal{V}_1}^{r-1}$ ), we have  $|\mathcal{Z}_{\mathcal{V}_1}^{r-1}| = |\mathcal{V}_1|$ , leading us to a contradiction.

Therefore, for nonempty disjoint subsets  $\mathcal{V}'_1, \mathcal{V}'_2$ , it must hold that  $|\mathcal{Z}_{\mathcal{V}'_1}^r| + |\mathcal{Z}_{\mathcal{V}'_2}^r| \geq s$ . Together with (3.17), we have

$$|\mathcal{Z}_{\mathcal{V}_1}^{r-1}| + |\mathcal{Z}_{\mathcal{V}_2}^{r-1}| \geq |\mathcal{Z}_{\mathcal{V}'_1}^r| + |\mathcal{Z}_{\mathcal{V}'_2}^r| + 1 \geq s + 1,$$

suggesting that  $\mathcal{G}$  is  $(r - 1, s + 1)$ -robust with  $l$  hops under the  $f$ -total model. ■

From this result, we can directly derive the following corollary, which indicates that if a graph is  $(2f + 1)$ -robust with  $l$  hops, then it is also  $(f + 1, f + 1)$ -robust with  $l$  hops.

**Corollary 3.6.1** *If a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is  $(r + s - 1)$ -robust with  $l$  hops, where  $1 \leq r + s - 1 \leq \lceil n/2 \rceil$ , then  $\mathcal{G}$  is  $(r, s)$ -robust with  $l$  hops.*

Similar to the one-hop case, robustness with  $l$  hops can guarantee a certain level of graph connectivity and a minimum in-degree of the graph. The proof for the connectivity result is trivial and hence omitted.

### 3.6 Discussions on Graph Robustness with Multi-hop Communication

---

**Lemma 3.6.5** *If a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is  $r$ -robust with  $l$  hops, then  $\mathcal{G}$  is at least  $r$ -connected.*

**Lemma 3.6.6** *If a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is  $(r, s)$ -robust with  $l$  hops under the  $(r - 1)$ -total model, where  $0 \leq r \leq \lceil n/2 \rceil$  and  $1 \leq s \leq n$ , then  $\mathcal{G}$  has the minimum in-degree  $\delta(\mathcal{G})$  as*

$$\delta(\mathcal{G}) \geq \begin{cases} r + s - 1 & \text{if } s < r, \\ 2r - 2 & \text{if } s \geq r. \end{cases}$$

*Proof:* The case for  $n \leq 2$  and  $r \leq 1$  is trivial. Consider the case for  $n \geq 3$  and  $2 \leq r \leq \lceil n/2 \rceil$ . Fix node  $i \in \mathcal{V}$ . Choose  $\mathcal{V}_1 = \{i\}$  and  $\mathcal{V}_2 = \mathcal{V} \setminus \mathcal{V}_1$ . Then,  $|\mathcal{Z}_{\mathcal{V}_2}^r| = 0$  and  $|\mathcal{Z}_{\mathcal{V}_1}^r| = |\mathcal{V}_1|$ . Hence, node  $i$  must have at least  $r$  independent paths from outside and the number of the in-neighbors of node  $i$  should be  $|\mathcal{N}_i^-| \geq r$ . (Note that the malicious nodes can be the source nodes of these independent paths.)

When  $s < r$ , form  $\mathcal{V}_1$  by choosing  $s - 1$  in-neighbors of node  $i$  along with node  $i$  itself. Then, choose  $\mathcal{V}_2 = \mathcal{V} \setminus \mathcal{V}_1$ . Since  $|\mathcal{V}_1| = s < r$ , then,  $|\mathcal{Z}_{\mathcal{V}_2}^r| = 0$  and  $|\mathcal{Z}_{\mathcal{V}_1}^r| = |\mathcal{V}_1|$ . The worst case is that the  $s - 1$  in-neighbors of node  $i$  are all malicious, node  $i$  should have at least  $r$  independent paths from outside, and these paths do not contain any malicious nodes as internal nodes. This implies that node  $i$  has additional  $r$  in-neighbors outside of  $\mathcal{V}_1$ , thereby guaranteeing  $|\mathcal{N}_i^-| \geq r + s - 1$ .

When  $s \geq r$ , form  $\mathcal{V}_1$  by choosing  $r - 2$  in-neighbors of node  $i$  along with node  $i$  itself. Then, choose  $\mathcal{V}_2 = \mathcal{V} \setminus \mathcal{V}_1$ . Since  $|\mathcal{V}_1| < r$  and  $s \geq r$ , we have  $|\mathcal{Z}_{\mathcal{V}_2}^r| = 0$  and  $|\mathcal{Z}_{\mathcal{V}_1}^r| = |\mathcal{V}_1|$ . Consider the worst case that the  $r - 2$  in-neighbors of node  $i$  are all malicious. It must be that node  $i$  has additional  $r$  in-neighbors outside of  $\mathcal{V}_1$ , thereby guaranteeing  $|\mathcal{N}_i^-| \geq 2r - 2$ . Since we choose  $i \in \mathcal{V}$  arbitrarily, we have proved the bound for  $\delta(\mathcal{G})$ . ■

Here, we have extended the proof in [56] to the multi-hop case. From Lemma 3.6.6, we conclude that  $\mathcal{G}$  should have the minimum in-degree no less than  $2f$  to guarantee resilient consensus using the MW-MSR algorithm under the  $f$ -total malicious model. This holds since the underlying graph  $\mathcal{G}$  is at least  $(f + 1, f + 1)$ -robust with  $l$  hops for achieving resilient consensus. For MSR algorithms, nodes with  $2f$  in-neighbors may not use any values from neighbors, which may appear problematic especially if there are multiple such nodes. However, there are examples such as cycle graphs<sup>3</sup> satisfying

---

<sup>3</sup>A cycle graph is an undirected graph consisting of only a single cycle.

### 3.6 Discussions on Graph Robustness with Multi-hop Communication

---

$(f + 1, f + 1)$ -robust with  $l$  hops, while every node in cycle graphs has in-degree as  $2f$ . Detailed analysis is given in the next subsection.

#### 3.6.2 The Case of Unbounded Path Length

In this subsection, we discuss the relation of the graph conditions used in this paper and in the recent work [53]. The authors there studied the Byzantine binary consensus under the local broadcast model, which is essentially equivalent to the  $f$ -total malicious mode studied in the current paper. The proposed algorithm in [53] is based on a non-iterative flooding algorithm, where nodes must relay their values over the entire network along with the path information. This model corresponds to the case of unbounded path length in our work, i.e.  $l \geq l^*$ , where  $l^*$  is the longest cycle-free path length of the network. Moreover, they propose a tight necessary and sufficient graph condition for their algorithm to achieve binary consensus under synchronous updates. Our aim in this part of the paper is to establish that our graph condition is equivalent to theirs for the case of unbounded path length ( $l \geq l^*$ ). Further, we will highlight that to achieve the same tolerance as the algorithm in [53], our algorithm does not in general require  $l^*$ -hop communication necessarily for general graphs.

To show the equivalence between the two graph conditions, we introduce some graph notions from [53]. There, normal nodes update the states based on a modified certified propagation algorithm [110], i.e., when a normal node receives  $f + 1$  same binary values from different paths excluding a suspicious set  $\mathcal{F}$ , it commits its value to this value. Hence, their graph notion is closely related to the partitions of sets  $\mathcal{V}$  and  $\mathcal{F}$ .

**Definition 3.6.1** *For disjoint node sets  $\mathcal{X}, \mathcal{Y}$ , we say  $\mathcal{X} \rightarrow \mathcal{Y}$  if and only if set  $\mathcal{X}$  contains at least  $f + 1$  distinct incoming neighbors of  $\mathcal{Y}$ , i.e.,  $|\{i : (i, j) \in \mathcal{E}, i \in \mathcal{X}, j \in \mathcal{Y}\}| > f$ . Denote  $\mathcal{X} \not\rightarrow \mathcal{Y}$  when  $\mathcal{X} \rightarrow \mathcal{Y}$  is not true.*

**Definition 3.6.2** *For disjoint node sets  $\mathcal{X}, \mathcal{Y}$  and for set  $\mathcal{F}$ , we say  $\mathcal{X} \overset{\mathcal{F}}{\rightsquigarrow} \mathcal{Y}$  if and only if for every node  $u \in \mathcal{Y}$ , there exist at least  $f + 1$  disjoint  $\mathcal{X}u$ -paths that have only  $u$  in common and none of them contains any internal node from the set  $\mathcal{F}$ . Denote  $\mathcal{X} \not\overset{\mathcal{F}}{\rightsquigarrow} \mathcal{Y}$  when  $\mathcal{X} \overset{\mathcal{F}}{\rightsquigarrow} \mathcal{Y}$  is not true.*

Two graph notions are introduced next, called conditions NC and SC. They are known to be equivalent [53].

### 3.6 Discussions on Graph Robustness with Multi-hop Communication

---

**Definition 3.6.3** (Condition NC) Given a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , condition NC is said to hold if for every partition  $\mathcal{L}, \mathcal{C}, \mathcal{R}$  of  $\mathcal{V}$ , and for every set  $\mathcal{F}$  with  $|\mathcal{F}| \leq f$ , where both  $\mathcal{L} \setminus \mathcal{F}$  and  $\mathcal{R} \setminus \mathcal{F}$  are non-empty, we have that either  $\mathcal{R} \cup \mathcal{C} \rightarrow \mathcal{L} \setminus \mathcal{F}$  or  $\mathcal{L} \cup \mathcal{C} \rightarrow \mathcal{R} \setminus \mathcal{F}$ .

(Condition SC) Given a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , condition SC is said to hold if for every partition  $\mathcal{L}, \mathcal{R}$  of  $\mathcal{V}$ , and for every set  $\mathcal{F}$  with  $|\mathcal{F}| \leq f$ , where both  $\mathcal{L} \setminus \mathcal{F}$  and  $\mathcal{R} \setminus \mathcal{F}$  are non-empty, we have that either  $\mathcal{L} \xrightarrow{\mathcal{F}} \mathcal{R} \setminus \mathcal{F}$  or  $\mathcal{R} \xrightarrow{\mathcal{F}} \mathcal{L} \setminus \mathcal{F}$ .

We are now ready to show that condition NC (and hence condition SC) is equivalent to our robust graph notion with multi-hop communication used in Theorem 3.4.1.

**Proposition 3.6.1** Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication where  $l \geq l^*$ . The graph  $\mathcal{G}$  is  $(f + 1, f + 1)$ -robust with  $l$  hops if and only if condition NC holds.

*Proof:* We first show the only if part. Since the graph is  $(f + 1, f + 1)$ -robust with  $l$  hops under the  $f$ -total model, at least one of the three conditions in Definition 3.3.2 holds. For every partition  $\mathcal{L}, \mathcal{R}$  of  $\mathcal{V}$ , and for every set  $|\mathcal{F}| \leq f$ , where both  $\mathcal{L} \setminus \mathcal{F}$  and  $\mathcal{R} \setminus \mathcal{F}$  are non-empty, we conclude that there is at least one normal node  $i$  in the union of  $\mathcal{L}$  and  $\mathcal{R}$  that has  $f + 1$  independent paths from outside and these paths do not contain any internal nodes in  $\mathcal{F}$ . This can be seen in the proof of Theorem 3.4.1. Suppose that node  $i \in \mathcal{L}$  has this property. For each independent path, there exists at least one edge that goes from the outside of  $\mathcal{L}$  to a node in  $\mathcal{L}$ , and thus,  $\mathcal{R} \cup \mathcal{C} \rightarrow \mathcal{L} \setminus \mathcal{F}$ . The case for  $i \in \mathcal{R}$  can be proved similarly.

Next, we show the if part by contradiction. Suppose that  $\mathcal{G}$  is not  $(f + 1, f + 1)$ -robust with  $l$  hops. Then, none of the three conditions in Definition 3.3.2 holds. For every partition  $\mathcal{L}, \mathcal{R}$  of  $\mathcal{V}$ , and for every set  $|\mathcal{F}| \leq f$ , where both  $\mathcal{L} \setminus \mathcal{F}$  and  $\mathcal{R} \setminus \mathcal{F}$  are non-empty, we conclude that all the normal nodes in the union of  $\mathcal{L}$  and  $\mathcal{R}$  have at most  $f$  independent paths from outside where these paths do not contain any internal nodes in  $\mathcal{F}$ . Hence,  $\mathcal{L} \not\xrightarrow{\mathcal{F}} \mathcal{R} \setminus \mathcal{F}$  and  $\mathcal{R} \not\xrightarrow{\mathcal{F}} \mathcal{L} \setminus \mathcal{F}$  (i.e., condition SC does not hold), and thus we have contradiction.

Finally, since condition SC and condition NC are equivalent, we have proved that condition NC implies the  $(f + 1, f + 1)$ -robustness with  $l$  hops. ■

Although our condition coincides with those in [53] when  $l \geq l^*$ , we note that the maximum robustness of a given graph does not require  $l \geq l^*$  necessarily. That is, for a given graph under the  $f$ -total model, our algorithm may not require  $l^*$ -hop

### 3.6 Discussions on Graph Robustness with Multi-hop Communication

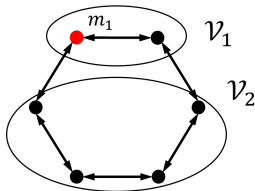


Figure 3.5: Illustration for cycle graphs.

communication to get the same tolerance as the algorithm using  $l^*$ -hop communication in [53]. We illustrate this fact by presenting the examples of cycle graphs.

From the example graph in Fig. 3.3(a), we can see that for any cycle graph, the following lemma holds. This indicates that resilient consensus is guaranteed using the MW-MSR algorithm in such graphs when one node behaves maliciously. In [53], it is reported that a cycle graph can tolerate one malicious node, but their algorithm uses the  $l^*$ -hop communication. In this respect, as the following lemma suggests, our algorithm is efficient by exploiting the ability of the MW-MSR algorithm and the graph condition is tighter than the one in [53]. Moreover, note that a cycle graph is 2-connected, which is the minimum connectivity requirement for any MSR-based algorithm to guarantee resilient consensus for  $f = 1$ .

**Lemma 3.6.7** *The cycle graph  $\mathcal{C}_n$  with  $n > 2$  nodes is  $(2, 2)$ -robust with  $\lceil l^*/2 \rceil$  hops under the 1-total model.*

*Proof:* We need to show that for any node partition  $\mathcal{V}_1, \mathcal{V}_2$  of  $\mathcal{V}$ , at least one of the conditions for  $(2, 2)$ -robustness with  $l$  hops holds. Let  $\mathcal{F}$  be a set of a single node, i.e.,  $\mathcal{F} = \{m_1\}$  (satisfying the 1-total model). We first select a single node as set  $\mathcal{V}_1$ . Then, for any set  $\mathcal{F}$  and for any set  $\mathcal{V}_2$ , condition 1) for  $(2, 2)$ -robustness with  $l$  hops holds. (The case is similar when we select non-neighboring nodes as set  $\mathcal{V}_1$ .) Second, we select two neighboring nodes as set  $\mathcal{V}_1$ . If node  $m_1 \notin \mathcal{V}_1$ , then condition 1) for  $(2, 2)$ -robustness with 2 hops holds. If node  $m_1 \in \mathcal{V}_1$ , then node  $m_1$  has 2 independent 2-hop paths originating from outside. To meet the conditions for  $(2, 2)$ -robustness with  $l$  hops, we need to find another node having this property in  $\mathcal{V}_2$  (see the illustration in Fig. 3.5). The worst case is when all the remaining nodes are in  $\mathcal{V}_2$ . Then the middle node in  $\mathcal{V}_2$  has 2 shortest paths originating from outside, which are of length  $\lceil l^*/2 \rceil$  hops.

We can continue this process and select three neighboring nodes as set  $\mathcal{V}_1$ . We can follow an analysis as above: If node  $m_1 \in \mathcal{V}_1$  and all the remaining nodes are in  $\mathcal{V}_2$ ,

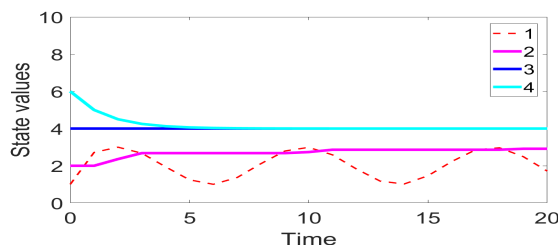


Figure 3.6: Time responses of the synchronous one-hop W-MSR algorithm.

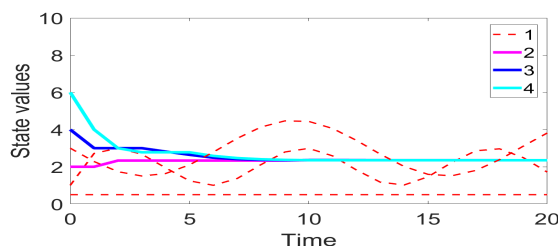


Figure 3.7: Time responses of the synchronous two-hop MW-MSR algorithm.

then the middle node in  $\mathcal{V}_2$  has 2 shortest paths originating from outside, which are shorter than the length  $\lceil l^*/2 \rceil$  hops. This process can be continued until we switch sets  $\mathcal{V}_1$  and  $\mathcal{V}_2$ . Hence, we conclude that the cycle graph  $\mathcal{C}_n$  is  $(2, 2)$ -robust with  $\lceil l^*/2 \rceil$  hops. ■

## 3.7 Numerical Examples

In this section, we conduct numerical simulations over networks using both synchronous and asynchronous versions of the proposed MW-MSR algorithm to verify their effectiveness.

### 3.7.1 Synchronous MW-MSR Algorithm

In this part, we conduct simulations for the synchronous MW-MSR algorithm. Consider the undirected network in Fig. 3.3(a) with  $f = 1$ . Let the initial states be  $x[0] = [1 \ 2 \ 4 \ 6]^T$ . This graph is not  $(2, 2)$ -robust with one hop, and hence, is not robust enough to tolerate  $f = 1$  using the conventional one-hop W-MSR algorithm. Here we set node 1 to be malicious and let the value of node 1 evolve based on the sine function w.r.t. time. Then, normal nodes update their values using the one-hop W-MSR algorithm. The results are given in Fig. 3.6, and resilient consensus among normal nodes is not achieved.

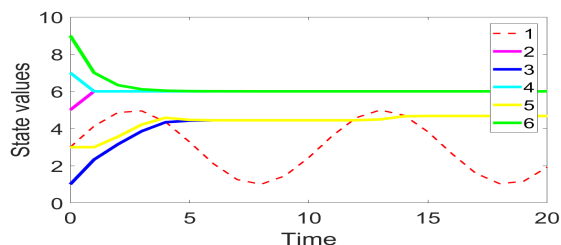
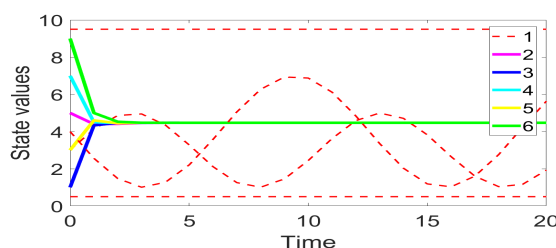
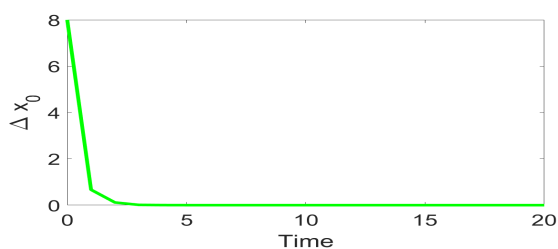


Figure 3.8: Time responses of the synchronous one-hop W-MSR algorithm.



(a) The states of all nodes.



(b) Consensus error.

Figure 3.9: Time responses of the synchronous two-hop MW-MSR algorithm.

Next, consider the two-hop case for this graph. For malicious node 1, we assume that it does not only manipulate its own value as in the one-hop case, but also relays false information. Specifically, when node 1 receives a message from node 4 and relays the value  $x_4[k]$  to node 2, it manipulates this value based on the sine function w.r.t. time. Similarly, when node 1 receives a message from node 2 and relays the value  $x_2[k]$  to node 4, it manipulates this value to a fixed value of 0.5. Then, we observe that resilient consensus is achieved as shown in Fig. 3.7.

### 3.7.2 Asynchronous MW-MSR Algorithm

In this part, we conduct simulations for the asynchronous MW-MSR algorithm. Consider the directed network in Fig. 3.4 with  $f = 1$ . Let the nodes take initial states as  $x[0] = [3 \ 5 \ 1 \ 7 \ 3 \ 9]^T$ . This graph is not 2-robust with one hop (e.g., consider the

sets  $\{1, 3, 5\}$  and  $\{2, 4, 6\}$ ), and hence, is not robust enough to tolerate  $f = 1$  using the one-hop W-MSR algorithm. Here, we set node 1 to be malicious and assume that the value of node 1 evolves based on the sine function w.r.t. time. Then, we apply the one-hop W-MSR algorithm and observe that resilient consensus among normal nodes is not achieved as shown in Fig. 3.8.

Next, consider the two-hop case for this graph. It becomes 3-robust with 2 hops, and hence, it is also (2, 2)-robust with 2 hops as Corollary 3.6.1 indicates. Therefore, in the two-hop case, it can tolerate one malicious node under both synchronous and asynchronous updates. For malicious node 1, we assume that it does not only manipulate its own value as in the one-hop case, but also relays false information. Specifically, when node 1 receives a message from node 3 and relays the value  $x_3[k]$  to its other neighbors, it manipulates this value to a fixed value of 0.5. Similarly, when node 1 receives a message from node 6 and relays the value  $x_6[k]$  to other neighbors, it manipulates this value to a fixed value of 9.5. Additionally, when node 1 receives a message from node 5 and relays the value  $x_5[k]$  to other neighbors, it manipulates this value based on the sine function w.r.t. time. In Fig. 3.9, we plot the consensus error given by  $\Delta x_0[k] = \max x^N[k] - \min x^N[k]$  and observe that resilient consensus is achieved.

Lastly, we examine the two-hop algorithm under asynchronous updates with delays. We consider the same attack, but let each normal node update in a periodic manner. Specifically, nodes 2, 3, 4, 5, and 6 update in every 1, 5, 4, 3, and 2 steps, respectively. The delays for the messages from one-hop neighbors and two-hop neighbors are set as 0 and 1 step, respectively. Fig. 3.10 shows the states as well as the consensus error given by  $\Delta x_\tau[k] = \max z^N[k] - \min z^N[k]$ . It indicates that consensus is attained despite the malicious attacks. Note that in this situation, we can only guarantee the nonincreasing property of  $\Delta x_\tau[k]$  (with  $\tau = 5$ ). Through these simulations, we have verified the effectiveness of the MW-MSR algorithms to achieve resilient consensus in small-scale networks.

### 3.7.3 Simulations in Large Wireless Sensor Networks

In this part of the simulations, we create a WSN composed of 100 nodes located in a grid structure as shown in Fig. 3.11. Let the nodes take indices  $0, 1, \dots, 99$  and the coordinate of node  $i$  is  $(i \bmod 10, \lfloor \frac{i}{10} \rfloor)$ . Each node can communicate only with the nodes located within the communication radius of  $r$ . Once  $r$  is determined, the



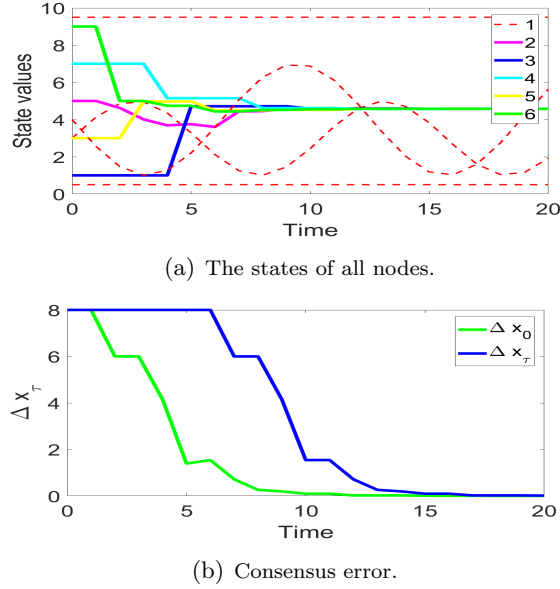


Figure 3.10: Time responses of the asynchronous two-hop MW-MSR algorithm.

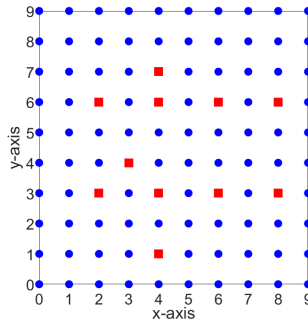


Figure 3.11: The 100-node sensor network. The red nodes are set as malicious one by one as  $f$  increases up to 11.

topology of the network is formed. Then we apply the one-hop, two-hop, and three-hop MW-MSR algorithms to the network. Recall that  $f$  denotes the maximum number of malicious nodes in the network, and we increase  $f$  from 0 to 11 by selecting the malicious nodes with indices in the order of 32, 34, 36, 38, 43, 62, 64, 66, 68, 74, 14.

Here, we examine how the network connectivity affects the performance of the MW-MSR algorithms with different hops. Using different values for the number of malicious agents  $f$  and the communication radius  $r$ , the results of the one-hop algorithm are presented in Fig. 3.12(a). For each  $f$  and each  $r$  (corresponding to one cell in the figure), we compute the success rate of the algorithm to achieve resilient consensus

over 50 Monte Carlo runs with randomly chosen initial values (within  $[0, 100]$ ) of the normal agents for each run. The malicious nodes take values based on sine functions w.r.t. time. Similarly, we also conduct simulations for the two-hop and the three-hop algorithms and the results are given in Figs. 3.12(b) and (c), respectively.

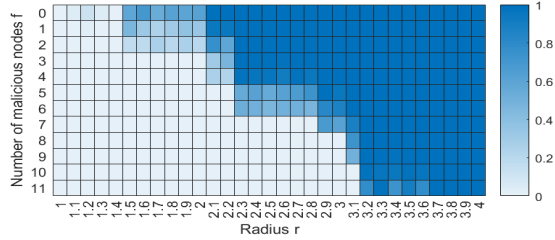
One can see that by increasing the number of hops  $l$ , the success rate of the algorithm to achieve resilient consensus increases almost for every value of  $f$ . Such improvement is especially significant when  $f \leq 6$  and  $r \leq 3$ . This verifies our intuition as well as theoretical findings that graph robustness increases as  $l$  increases. However, the difference between the results for the two-hop and the three-hop algorithms is somewhat minor. One reason is that the maximum robustness under the  $f$ -total model of a given graph is bounded by the minimum in-degree  $2f$ . This may indicate that the two-hop communication for this graph already reaches the number of hops for the maximum graph robustness.

In these simulations, the success rate for reaching consensus is determined by the level of consensus error at time  $k = 70$ . If the consensus error is below the threshold  $c = 1$ , then the run is considered as a success. Hence, if the consensus process is very slow, it may be considered as a failure. For example, observe that for the case of  $f = 0$  (i.e., with no attacks) the success rate increases with larger  $r \leq 1.5$  while the network is connected as long as  $r > 1$ .

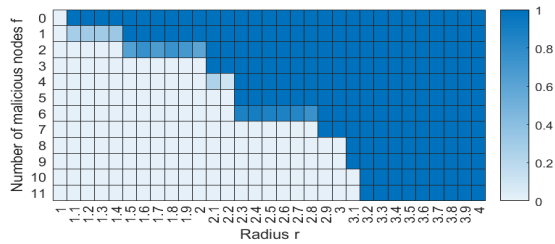
We should remark that the process of consensus forming can be accelerated by increasing the number of hops, as discussed in [48] for the fault-free case. This can be clearly seen in all plots in Fig. 3.13, presenting the time responses of the consensus errors for several cases of  $f$  and  $r$ , where  $\Delta x_1$ ,  $\Delta x_2$  and  $\Delta x_3$  stand for the consensus errors for the one-hop, two-hop, and three-hop algorithms, respectively. Based on these examples, we conclude that by introducing multi-hop communication to the MSR algorithms, it does not only improve the robustness of the network but also accelerates the convergence in consensus forming even in adversarial environments.

### 3.8 Summary

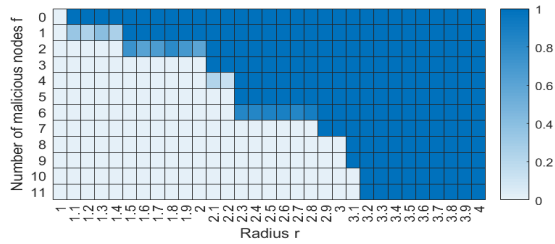
In this chapter, we have investigated the resilient consensus problem when multi-hop communication is available. We have proposed generalized versions of MSR algorithms to correctly use the additional values received from multi-hop neighbors. Moreover, we have fully characterized the network requirement for the algorithms in terms of



(a) One-hop algorithm.



(b) Two-hop algorithm.



(c) Three-hop algorithm.

Figure 3.12: Success rate of the MW-MSR algorithm.

robustness with  $l$  hops. By introducing multi-hop communication, the convergence of the resilient consensus process can be accelerated. Furthermore, it provides an effective way to enhance robustness of networks without increasing physical communication links. In future works, we intend to extend our algorithms to the asynchronous Byzantine consensus problem using multi-hop communication with a fixed number of hops.

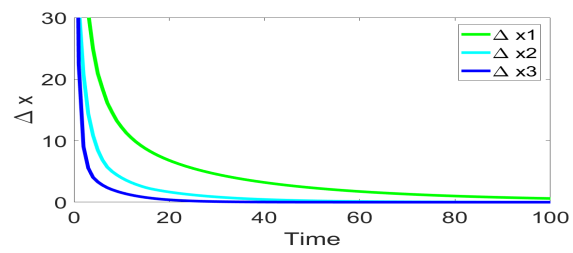
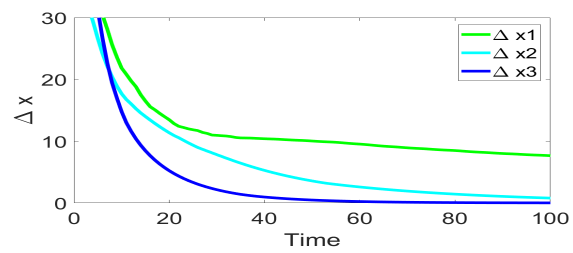
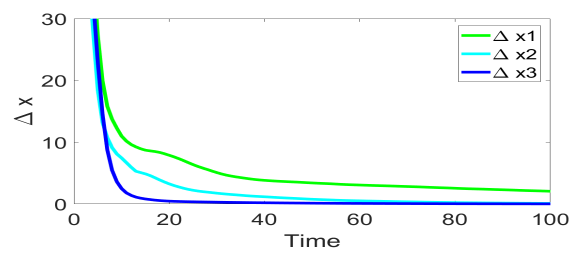
(a) Consensus error for  $f = 0, r = 1.2$ .(b) Consensus error for  $f = 1, r = 1.2$ .(c) Consensus error for  $f = 9, r = 3.1$ .

Figure 3.13: Consensus error of the MW-MSR algorithm.

## Chapter 4

# Asynchronous Approximate Byzantine Consensus via Multi-hop Communication

In this chapter, we study a resilient consensus problem with a distinctive feature in comparison to the one in Chapter 3: The adversaries are of the Byzantine type. We solve the approximate Byzantine consensus problem based on the multi-hop extensions of MSR algorithms from Chapter 3. We find a tight graph condition expressed in terms of strongly robust graphs. We show that while multi-hop communication may increase the attackers options for attacks, it is capable to enhance the resilience of the multi-agent system under Byzantine attacks.

Compared to related works, the contributions of this chapter can be outlined as follows. The asynchronous Byzantine consensus problem was studied in [94], but the algorithm there requires agents to send their values to the entire network. We emphasize that the case studied in [94] can be viewed as a special case. In fact, it corresponds to multi-hop paths with unbounded lengths, and their condition coincides with our graph condition by setting the path length to be the longest cycle-free one in the graph. Since in our model, the number of hops is limited, our approach is more distributed in the sense that we only require each normal node to have the local topology information and neighbors' values up to  $l$  hops away. We also study the  $f$ -local model for our algorithm, which is even more adversarial than the  $f$ -total model studied in [94]. Besides, there are further differences in asynchrony settings between our approach and [94]. More

details are given in Section 4.4.

The rest of this chapter is organized as follows. Section 4.1 outlines the system model. Section 4.2 presents the notion and properties of graph robustness with  $l$  hops. Sections 4.3 and 4.4 derive conditions under which the MW-MSR algorithms guarantee Byzantine consensus under synchronous and asynchronous updates, respectively. Section 4.5 provides examples to demonstrate that multi-hop communication can improve the robustness of general graphs. Lastly, Section 4.6 concludes this chapter.

## 4.1 Problem Formulation

In this section, we first state the system model studied in this chapter. Then, we outline the structure of the resilient consensus algorithm.

### 4.1.1 Update Rule

Consider the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  consisting of the node set  $\mathcal{V} = \{1, \dots, n\}$  and the edge set  $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ . The subgraph of  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  induced by the node set  $\mathcal{H} \subset \mathcal{V}$  is the subgraph  $\mathcal{G}_{\mathcal{H}} = (\mathcal{V}(\mathcal{H}), \mathcal{E}(\mathcal{H}))$ , where  $\mathcal{V}(\mathcal{H}) = \mathcal{H}$ ,  $\mathcal{E}(\mathcal{H}) = \{(i, j) \in \mathcal{E} : i, j \in \mathcal{H}\}$ . We adopt the same relay model for the normal nodes as the one in Chapter 3, which is similar to the ones in the multi-hop works [37; 102]. However, the threat model studied in Chapter 3 is different from that of the current chapter.

Consider a time-invariant network modeled by the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . The node set  $\mathcal{V}$  is partitioned into the set of normal nodes  $\mathcal{N}$  and the set of adversary nodes  $\mathcal{A}$ , where  $n_{\mathcal{N}} = |\mathcal{N}|$  and  $n_{\mathcal{A}} = |\mathcal{A}|$ .

Recall that agents communicate with each other according to the communication model indicated in Section 2.3.1. When there is no attack in the network, we can employ the common consensus update rule (e.g., [82]). It can be given in the compact form as

$$x[k+1] = x[k] + u[k], \quad u[k] = -L[k]x[k], \quad (4.1)$$

where  $x[k] \in \mathbb{R}^n$  and  $u[k] \in \mathbb{R}^n$  are the state vector and the control input vector respectively, and  $L[k]$  is the Laplacian matrix of the  $l$ -th power of  $\mathcal{G}$  determined by the messages  $m_{ij}[k]$  for  $i \in \mathcal{V}$  and  $j \in \mathcal{N}_i^{l-}$ .

Then we introduce asynchrony in our algorithm [21; 120]. At each time  $k$ , normal node  $i$  may or may not update its value. If node  $i$  does not update, then  $x_i[k+1] = x_i[k]$ ,

i.e.,  $u_i[k] = 0$ . Denote by  $\mathcal{U}[k] \subset \mathcal{V}$  the set of agents updating at time  $k$ . The system is said to be synchronous if  $\mathcal{U}[k] = \mathcal{V}$  for all  $k$ , and otherwise it is asynchronous. Discussions about different asynchrony settings are presented in Section 4.4.

### 4.1.2 Threat Models

In this chapter, we study the resilient consensus under  $f$ -total/local Byzantine attacks. The definitions of the adversary model is given in Section 2.2.2. Byzantine models are well studied in the area of computer science [26; 62; 113]. Note that the *malicious* model studied in [21; 56] and Chapter 3 is a weaker threat model compared to Byzantine model as malicious nodes must send the same information to their neighbors, which is suitable for broadcast networks.

As commonly done in the literature, we assume that each normal node knows the value of  $f$  and the topology information of the graph up to  $l$  hops. Moreover, to keep the problem tractable, we assume that each adversary node  $i$  cannot manipulate the path values in the messages containing its own state  $x_i[k]$  and those that it relays as stated in Assumption 2.3.1.

### 4.1.3 Resilient Asymptotic Consensus and Algorithm

We now introduce the type of consensus among the normal agents to be sought in this chapter [21; 56; 102; 125].

**Definition 4.1.1** *If for any possible sets and behaviors of the adversary agents and any state values of the normal nodes, the following two conditions are satisfied, then we say that the normal agents reach resilient asymptotic consensus:*

1. *Safety: There exists a bounded safety interval  $\mathcal{S}$  determined by the initial values of the normal agents such that  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .*
2. *Agreement: There exists a state  $x^* \in \mathcal{S}$  such that  $\lim_{k \rightarrow \infty} x_i[k] = x^*, \forall i \in \mathcal{N}$ .*

In this chapter, the main goal is to characterize the conditions on the network structure that guarantee approximate asynchronous Byzantine consensus using the MW-MSR algorithm (presented in Algorithm 1). Before proceeding to such an analysis, in Section 4.3, we introduce several notions related to robust graphs. In Section 4.4, we first consider the case of synchronous updates. Then, in Section 4.5, we consider the

more realistic situation using multi-hop settings, which is the asynchronous updates with time delays in the communication among agents.

## 4.2 Graph Robustness with Multi-hop Communication

In this section, we provide the definition of robustness with multi-hop communication, which is the key graph condition to guarantee resilient consensus.

### 4.2.1 The Notion of $(r, s)$ -Robustness with $l$ Hops

The notion of graph robustness was first introduced in [56], and it was proved that graph robustness gives a tight graph condition guaranteeing resilient consensus using MSR-based algorithms. In [125], we generalized this notion to the multi-hop case, where nodes can exchange values with their  $l$ -hop neighbors through different paths. Its definition is as follows. See Section 3.3 for more discussions for this notion.

**Definition 4.2.1** *A directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is said to be  $(r, s)$ -robust with  $l$  hops with respect to a given set  $\mathcal{F} \subset \mathcal{V}$ , if for every pair of nonempty disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ , at least one of the following conditions holds:*

$$(1) \mathcal{Z}_{\mathcal{V}_1}^r = \mathcal{V}_1; (2) \mathcal{Z}_{\mathcal{V}_2}^r = \mathcal{V}_2; (3) |\mathcal{Z}_{\mathcal{V}_1}^r| + |\mathcal{Z}_{\mathcal{V}_2}^r| \geq s,$$

where  $\mathcal{Z}_{\mathcal{V}_a}^r$  is the set of nodes in  $\mathcal{V}_a$  ( $a = 1, 2$ ) that have at least  $r$  independent paths of at most  $l$  hops originating from nodes outside  $\mathcal{V}_a$  and all these paths do not have any nodes in set  $\mathcal{F}$  as intermediate nodes (i.e., the nodes in  $\mathcal{F}$  can be source or destination nodes in these paths). Moreover, if the graph  $\mathcal{G}$  satisfies this property with respect to any set  $\mathcal{F}$  satisfying the  $f$ -total model, then we say that  $\mathcal{G}$  is  $(r, s)$ -robust with  $l$  hops under the  $f$ -total model. When it is clear from the context, we just say  $\mathcal{G}$  is  $(r, s)$ -robust with  $l$  hops.

Next, we provide some properties of graph robustness with multi-hop communication shown in Section 3.6. Note that all the properties listed coincide with the ones of one-hop case in [56] when  $l = 1$ . Here,  $\lceil \cdot \rceil$  denotes the ceiling function.

**Lemma 4.2.1** *If a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is  $(r, s)$ -robust with  $l$  hops, then the following hold:*

1.  $\mathcal{G}$  is  $(r', s')$ -robust with  $l$  hops, where  $0 \leq r' \leq r, 1 \leq s' \leq s$ .



## 4.2 Graph Robustness with Multi-hop Communication

---

2.  $\mathcal{G}$  is  $(r - 1, s + 1)$ -robust with  $l$  hops.
3.  $\mathcal{G}$  has a directed spanning tree. Moreover, if  $\mathcal{G}$  is undirected, then it is  $r$ -connected.
4.  $r \leq \lceil n/2 \rceil$ . Moreover,  $\mathcal{G}$  is  $(r, s)$ -robust with  $l$  hops if it is  $(r + s - 1)$ -robust with  $l$  hops.

### 4.2.2 The Notion of $r$ -Strongly Robust Graphs with $l$ Hops

To deal with the Byzantine model, we need to focus on the subgraph consisting of only the normal nodes. Define such a subgraph as the *normal network* as follows.

**Definition 4.2.2** For a network  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , define the normal network of  $\mathcal{G}$ , denoted by  $\mathcal{G}_{\mathcal{N}}$ , as the network induced by the normal nodes, i.e.,  $\mathcal{G}_{\mathcal{N}} = (\mathcal{N}, \mathcal{E}_{\mathcal{N}})$ , where  $\mathcal{E}_{\mathcal{N}}$  is the set of directed edges among the normal nodes.

Note that for the malicious model, we use the notion of robustness with  $l$ -hops to guarantee that any normal node  $i$  can be influenced by the normal nodes outside the set it belongs to. Generally, the Byzantine model is more adversarial in the sense that Byzantine nodes can send different values to different neighbors. That means for an  $l$ -hop ( $l \geq 2$ ) Byzantine neighbor  $j$  of node  $i$ , node  $i$  can receive different values of node  $j$  even from the normal nodes between nodes  $j$  and  $i$ . Therefore, more connections among the normal nodes are required to guarantee Byzantine consensus.

For the one-hop algorithms in [56] and [113], the graph condition that the normal network is  $(f + 1)$ -robust is proved to be necessary and sufficient for achieving resilient consensus under the  $f$ -total Byzantine model. We now extend this notion to the multi-hop setting and define it as  $r$ -strongly robust graphs with  $l$  hops as follows.

**Definition 4.2.3** Let  $\mathcal{F}$  be a subset of nodes in  $\mathcal{G}$  and denote the subgraph of  $\mathcal{G}$  induced by node set  $\mathcal{H} = \mathcal{V} \setminus \mathcal{F}$  as  $\mathcal{G}_{\mathcal{H}}$ . Then graph  $\mathcal{G}$  is said to be  $r$ -strongly robust with  $l$  hops with respect to  $\mathcal{F}$  if the induced subgraph  $\mathcal{G}_{\mathcal{H}}$  is  $r$ -robust with  $l$  hops. If graph  $\mathcal{G}$  satisfies this property with respect to any set  $\mathcal{F}$  satisfying the  $f$ -total/local model, then we say that  $\mathcal{G}$  is  $r$ -strongly robust with  $l$  hops under the  $f$ -total/local model. When it is clear from the context, we just say  $\mathcal{G}$  is  $r$ -strongly robust with  $l$  hops.

We can see that both robustness and strong robustness depend on the choice of set  $\mathcal{F}$ . Moreover, this set depends on the upper bound of the number of the adversary nodes

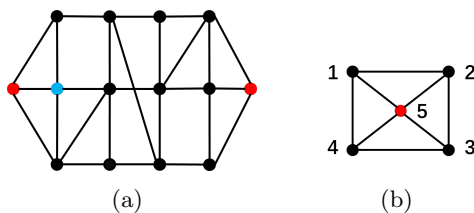


Figure 4.1: Both graphs are not 2-strongly robust with one hop but are 2-strongly robust with 2 hops.

and the adversarial models ( $f$ -total/local model). We finally illustrate how multi-hop communication can enhance resiliency through an example. The two graphs in Fig. 4.1 are not 2-strongly robust with one hop. See, for example, in the graph in Fig. 4.1(a), if we remove the node in blue, the remaining graph is not 2-robust. The graphs are however 2-strongly robust with 2 hops under both 1-total/local models.

### 4.3 Synchronous Byzantine Consensus

In this section, we provide the analysis of the MW-MSR algorithm under synchronous updates.

It is worth noting that the recent paper [102] investigated an MSR-based algorithm with multi-hop communication in the presence of  $f$ -total Byzantine model under synchronous updates. There, they provided a necessary and sufficient condition for their algorithm to achieve Byzantine consensus. While their proof techniques are different, the condition can be interpreted by the notion of robustness with multi-hop communication as well. Here, we extend the proof for the  $f$ -local model, which contains the case of the  $f$ -total model. Besides, based on our proof scheme, we can provide the analysis of our MW-MSR algorithm applied in asynchronous updates with time delays as we will see in Section 4.5; such a case is absent in [102]. Moreover, our condition for the unbounded path length case is equivalent to the conditions in [26] for synchronous undirected networks.

#### 4.3.1 Consensus Analysis

Denote the vectors consisting of the states of the normal nodes and those of the Byzantine nodes by  $x^N[k]$  and  $x^A[k]$ , respectively. Then, for the agents using the synchronous

MW-MSR algorithm, the safety interval is given by

$$\mathcal{S} = [\min x^N[0], \max x^N[0]]. \quad (4.2)$$

We are ready to state the main result for this section.

**Proposition 4.3.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the synchronous MW-MSR algorithm with parameter  $f$ . Under the  $f$ -local Byzantine model, resilient asymptotic consensus is achieved with safety interval (4.2) if and only if  $\mathcal{G}$  is  $(f + 1)$ -strongly robust with  $l$  hops.*

*Proof:* (Necessity) If  $\mathcal{G}$  is not  $(f + 1)$ -strongly robust with  $l$  hops, then there exists a normal network  $\mathcal{G}_N$  which is not  $(f + 1)$ -robust with  $l$  hops. In such a case, there are nonempty, disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{N}$  such that any node in the two sets has at most  $f$  distinct paths (only the node itself is common in these paths) of at most  $l$  hops originating from normal nodes outside of its respective set. Let the nodes in the two sets take the maximum and minimum values in the network, respectively. Suppose that the Byzantine nodes send the maximum and minimum values to the nodes in  $\mathcal{V}_1$  and  $\mathcal{V}_2$ , respectively.

Consider node  $i \in \mathcal{V}_1$ . Since the cardinality of the minimum message cover of the values larger than itself (values from the Byzantine nodes) is at most  $f$ , node  $i$  will discard these values. We claim that the cardinality of the minimum message cover of the values smaller than itself (values from the normal nodes outside of  $\mathcal{V}_1$ ) is also at most  $f$ . This can be proved in three cases: (i) All the incoming neighbors outside of  $\mathcal{V}_1$  are direct neighbors of node  $i$ , (ii) all the incoming neighbors outside of  $\mathcal{V}_1$  are  $l$ -hop ( $l \geq 2$ ) neighbors of node  $i$ , and (iii) situations other than (i) and (ii). For case (i), it is clear that this statement holds. For case (ii), either node  $i$  has at most  $f$  independent paths from the  $l$ -hop ( $l \geq 2$ ) neighbors outside of  $\mathcal{V}_1$ , where the cardinality of the  $l$ -hop neighbors can be bigger than  $f$ ; or node  $i$  has more than  $f$  independent paths from the  $l$ -hop ( $l \geq 2$ ) neighbors outside of  $\mathcal{V}_1$ , where the cardinality of the  $l$ -hop neighbors can be at most  $f$ . In either ways, the cardinality of the minimum message cover of the minimum values is at most  $f$ . For case (iii), note that the direct neighbors will be part of the minimum message cover always. For the remaining  $l$ -hop neighbors outside, following the analysis for case (ii), we can conclude that the cardinality of the minimum

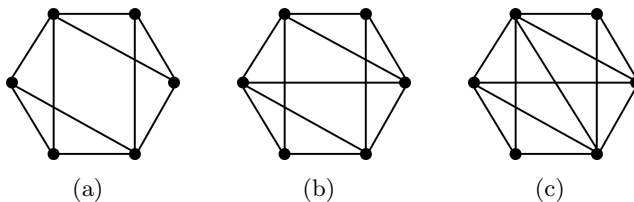


Figure 4.2: (a)  $(2, 2)$ -robust. (b) 2-strongly robust. (c) 3-robust.

message cover of the minimum values is at most  $f$ . Thus, in all cases, node  $i$  discards the values from the outside of  $\mathcal{V}_1$  and keeps its value.

Similar analysis applies when  $i \in \mathcal{V}_2$ . Therefore, nodes in these two sets never use any values from outside their respective sets and consensus cannot be reached.

(Sufficiency) Besides the method used in [102], we can prove the sufficiency part using the analysis as shown in the proof of Theorem 4.4.1, which is for asynchronous updates, since synchronous updates form one special case. ■

We must note that if  $\mathcal{G}$  is  $(f + 1)$ -strongly robust with  $l$  hops, then the normal network  $\mathcal{G}_N$  is guaranteed to be  $(f + 1)$ -robust with  $l$  hops for any possible cases of the adversary set  $\mathcal{A}$  under the  $f$ -local model. The latter condition is tighter than the former one, but it is not checkable in practice since the identities of the adversary nodes in  $\mathcal{A}$  are unknown. Thus, in Proposition 4.3.1, we provide the graph condition on  $\mathcal{G}$  instead of the condition on the normal network  $\mathcal{G}_N$ .

We would like to emphasize that our result is a generalization of those in the literature. As mentioned earlier, the work by [102] is restricted to the  $f$ -total model. On the other hand, [26] has studied the undirected networks case where the multi-hop communication has unbounded path lengths. In fact, our condition is equivalent to the two conditions there: (i)  $n \geq 3f + 1$  and (ii) the graph connectivity is no less than  $2f + 1$ . We can establish the condition (i) by noticing that complete networks have the largest robustness. By Lemma 4.2.1(4), the robustness of such a graph after removing any  $f$  nodes is no greater than  $\lceil \frac{n-f}{2} \rceil$ . Thus, our result implies  $\lceil \frac{n-f}{2} \rceil \geq f + 1$ , which is equivalent to  $n \geq 3f + 1$ . For the connectivity condition (ii), note that according to our graph condition, the graph after removing any  $f$  nodes needs to be  $(f + 1)$ -robust with  $l$  hops. Then, we can conclude that the graph connectivity is no less than  $2f + 1$  for any graph being  $(f + 1)$ -strongly robust with  $l$  hops.

### 4.3.2 Discussions on Different Graph Conditions

In Table 1.1, we have seen a summary of graph conditions for resilient consensus under different threat models and update schemes. Notably, there are three conditions under the  $f$ -total/local model. We will clarify the relations among these conditions in the next proposition. It shows a certain order among the different models as we will discuss further.

**Proposition 4.3.2** *For the following graph conditions on any directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  under the  $f$ -total/local model, where  $l \in \mathbb{Z}_+$ :*

- (A)  $\mathcal{G}$  is  $(2f + 1)$ -robust with  $l$  hops,
- (B)  $\mathcal{G}$  is  $(f + 1)$ -strongly robust with  $l$  hops,
- (C)  $\mathcal{G}$  is  $(f + 1, f + 1)$ -robust with  $l$  hops,

*it holds that  $A \Rightarrow B$  and  $B \Rightarrow C$ . Moreover,  $C \not\Rightarrow B$  and  $B \not\Rightarrow A$ .*

*Proof:* ( $A \Rightarrow B$ ) For a graph satisfying  $A$ , take a set  $\mathcal{F}$  satisfying the  $f$ -total/ $f$ -local model. Select any nonempty disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{H}$ , where  $\mathcal{H} = \mathcal{V} \setminus \mathcal{F}$ . Choose  $i \in \mathcal{Z}_{\mathcal{V}_1}^{2f+1}$ . Then, after removing nodes in the set  $\mathcal{F}$  from  $\mathcal{V}$ , it must hold that  $i \in \mathcal{Z}_{\mathcal{V}_1}^{f+1}$  in  $\mathcal{G}_{\mathcal{H}}$ . Hence,  $\mathcal{G}_{\mathcal{H}}$  is  $(f + 1)$ -robust with  $l$  hops. This is true for any set  $\mathcal{F}$ . Thus,  $B$  holds.

( $B \Rightarrow C$ ) We show that  $\neg C \Rightarrow \neg B$ . In a graph satisfying  $\neg C$ , for some nonempty disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ , at most  $f$  nodes in  $\mathcal{V}_1, \mathcal{V}_2$  have  $f + 1$  independent paths originating from the nodes outside. We choose these  $f$  nodes as the set  $\mathcal{F}$ . As a consequence, none of the remaining nodes in  $\mathcal{V}_1, \mathcal{V}_2$  has  $f + 1$  independent paths originating from the nodes outside. Hence this  $\mathcal{G}_{\mathcal{H}}$  is not  $(f + 1)$ -robust with  $l$  hops.

( $C \not\Rightarrow B, B \not\Rightarrow A$ ) We show these cases through counter examples in Fig. 4.2. Suppose that the set  $\mathcal{F}$  satisfies 1-total model (i.e.,  $|\mathcal{F}| = f = 1$ ). The graph in Fig. 4.2(a) is  $(2, 2)$ -robust (satisfying  $A$ ), but does not satisfy that any  $\mathcal{G}_{\mathcal{H}}$  is 2-robust where  $\mathcal{H} = \mathcal{V} \setminus \mathcal{F}$  (not satisfying  $B$ ). The graph in Fig. 4.2(b) satisfies that any  $\mathcal{G}_{\mathcal{H}}$  is 2-robust (satisfying  $B$ ), but this graph is not 3-robust (not satisfying  $C$ ). Moreover, this graph needs one more edge to be 3-robust as indicated in Fig. 4.2(c). ■

For synchronous update schemes, conditions (A) and (C) are for malicious adversaries under the  $f$ -local and  $f$ -total models, respectively. On the other hand, condition (B) is for Byzantine adversaries for both  $f$ -local/total models as we have seen in this section. It is clear that in comparison to the malicious model, Byzantine adversaries

are more adversarial and must require further connectivity in the network. Condition (A) has been known as a sufficient condition for the  $f$ -local malicious model, but we have now established a tighter result as shown in the following corollary.

**Corollary 4.3.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the synchronous MW-MSR algorithm with parameter  $f$ . Under the  $f$ -local malicious model, resilient asymptotic consensus is achieved with safety interval (4.2) if  $\mathcal{G}$  is  $(f + 1)$ -strongly robust with  $l$  hops and only if  $\mathcal{G}$  is  $(f + 1, f + 1)$ -robust with  $l$  hops.*

## 4.4 Asynchronous Byzantine Consensus

In this section, we provide the analysis of the MW-MSR algorithm under asynchronous updates with delays in communication. Furthermore, we show that our approach has certain advantages over the conventional ones in terms of threat models and computational complexity.

### 4.4.1 Discussions on Asynchrony

In the synchronous updates case studied in Section 4.3 as well as in [102], normal nodes are assumed to exchange values with  $l$ -hop neighbors within the same time step. In practice, however, normal nodes may not be synchronized nor have access to the current values of all  $l$ -hop neighbors simultaneously, especially when  $l$  is large. Thus, for the MSR algorithm with multi-hop communication, it is clearly desirable to carry out the analysis for asynchronous updates with time delays. In [94], the authors studied the Byzantine consensus under asynchronous updates using a flooding-based algorithm, which essentially requires the normal nodes to know the global network topology and send their values to the entire network. From the perspective of our study, this is a specific case where the path lengths are unbounded (i.e.,  $l \geq l^*$ , where  $l^*$  is the length of the longest cycle-free path in the network).

In this section, we provide the analysis for a network using the MW-MSR algorithm in the presence of  $f$ -total/local Byzantine model under asynchronous updates with time delays. The asynchrony setting in this paper follows the approach generally assumed in asynchronous consensus works for fault-free networks [59; 120], and those considering

malicious attacks (e.g., [21]). That is, when a normal node updates, it uses the most recently received values of its  $l$ -hop neighbors.

Here, we would like to briefly highlight how delays in asynchronous algorithms for resilient consensus are handled in the area of computer science especially through the notion of *rounds* commonly used in, e.g., [4; 6; 94]. There, each node labels its updated value with round  $r$ , representing the number of transmissions made so far. Moreover, if a normal node wants to update its next value with round  $r + 1$ , it has to wait until receiving a sufficient number of values labeled with the same round  $r$ . This may cause potentially large delays in making the  $(r + 1)$ th update for some nodes. We note that the use of rounds can create further problems due to following the fixed order in the indices of rounds. That is, node  $i$  may receive the value of round  $r + 1$  before the one of round  $r$  from its neighbor. This may occur even along a non-faulty path. In this case, based on rounds, the old data from round  $r$  will be used even though more recent data of round  $r + 1$  is available at the node. This is because the FIFO (first-in-first-out) message receiving mechanism is applied in [4], [94]. However, in our asynchrony setting, this is not a problem since node  $i$  will use the most recently received value of node  $j$  to update whenever node  $i$  chooses to update.

#### 4.4.2 Consensus Analysis

When communication among nodes is subject to possible time delays, we can write the control input as

$$u_i[k] = \sum_{j \in \mathcal{N}_i^{l-}} a_{ij}[k] x_j^P[k - \tau_{ij}^P[k]], \quad (4.3)$$

where  $\tau_{ij}^P[k] \in \mathbb{Z}_+$  denotes the delay in this  $(j, i)$ -path  $P$  at time  $k$  and  $x_j^P[k]$  denotes the value of node  $j$  at time  $k$  sent along path  $P$ . The delays are time varying and may be different in each path, but we assume the common upper bound  $\tau$  in any normal path  $P$  (i.e., all nodes on path  $P$  are normal) as

$$0 \leq \tau_{ij}^P[k] \leq \tau, \quad j \in \mathcal{N}_i^{l-}, \quad k \in \mathbb{Z}_+. \quad (4.4)$$

Hence, each normal node  $i$  becomes aware of the value of each of its normal  $l$ -hop neighbor  $j$  in each normal  $(j, i)$ -path  $P$  at least once in  $\tau$  time steps, but possibly at

different time instants [21]. This assumption also indicates that for each normal node, the gap between two consecutive updates should be less than  $\tau$ . Although we have this bound on the delays of values of normal nodes, normal nodes need neither the value of this bound nor the information that whether a path  $P$  is a normal path or not.

Compared to the synchronous case, in the asynchronous MW-MSR algorithm, we use the delayed values from neighbors as the input values. The structure of the asynchronous algorithm can be outlined as follows. At each time  $k$ , each normal node  $i$  will choose to update or not. If it chooses not to update, i.e.,  $i \notin \mathcal{U}[k]$ , then it will keep its value as  $x_i[k+1] = x_i[k]$ . Otherwise, it will use the most recently received values of its  $l$ -hop neighbors on each  $l$ -hop path to update its value using the MW-MSR algorithm in Algorithm 1. As in the one-hop MSR algorithm, if node  $i$  does not receive any value along some path  $P$  originating from its  $l$ -hop neighbor  $j$  (which corresponds to the crash model), then node  $i$  will consider this value on path  $P$  as one empty value and will discard this value when it applies the WM-MSR algorithm. As we discussed earlier, in the asynchronous case also, manipulation in message path information can be handled by normal nodes.

To proceed with our analysis, we introduce some notations. Let  $D[k]$  be a diagonal matrix whose  $i$ th entry is given by  $d_i[k] = \sum_{j=1}^n a_{ij}[k]$ . Then, let the matrices  $A_\gamma[k] \in \mathbb{R}^{n \times n}$  for  $0 \leq \gamma \leq \tau$ , and  $L_\tau[k] \in \mathbb{R}^{n \times (\tau+1)n}$  be given by

$$A_\gamma[k] = \begin{cases} a_{ij}[k] & \text{if } i \neq j \text{ and } \tau_{ij}[k] = \gamma, \\ 0 & \text{otherwise,} \end{cases} \quad (4.5)$$

and  $L_\tau[k] = \begin{bmatrix} D[k] - A_0[k] & -A_1[k] & \cdots & -A_\tau[k] \end{bmatrix}$ . Note that the summation of each row of  $L_\tau[k]$  is zero.

Now, the control input can be expressed as

$$\begin{aligned} u^N[k] &= -L_\tau^N[k]z[k], \\ u^A[k] &: \text{arbitrary,} \end{aligned} \quad (4.6)$$

where  $z[k] = [x[k]^T x[k-1]^T \cdots x[k-\tau]^T]^T$  is a  $(\tau+1)n$ -dimensional vector for  $k \geq 0$  and  $L_\tau^N[k]$  is a matrix formed by the first  $n_N$  rows of  $L_\tau[k]$ . Here,  $z[0]$  is the vector of given initial values.



Then, the agent dynamics can be written as

$$x[k+1] = \Gamma[k]z[k] + \begin{bmatrix} 0 \\ I_{n_A} \end{bmatrix} u^A[k], \quad (4.7)$$

where  $\Gamma[k]$  is an  $n \times (\tau + 1)n$  matrix given by  $\Gamma[k] = [I_n \ 0] - [L_\tau^N[k]^T \ 0]^T$ . The safety interval is given by

$$\mathcal{S}_\tau = \left[ \min z^N[0], \max z^N[0] \right]. \quad (4.8)$$

The theorem stated below is the main result of this paper providing a necessary and sufficient graph condition for the MW-MSR algorithm under the asynchronous updates.

**Theorem 4.4.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous MW-MSR algorithm with parameter  $f$ . Under the  $f$ -local Byzantine model for the adversarial nodes, resilient asymptotic consensus is achieved with the safety interval given by (4.8) if and only if  $\mathcal{G}$  is  $(f + 1)$ -strongly robust with  $l$  hops.*

*Proof:* (Necessity) The synchronous network is a special case of the asynchronous ones with  $\tau = 0$ . Thus, the necessary condition in Proposition 4.3.1 is valid here.

(Sufficiency) We first show that the safety holds. For  $k = 0$ , by (4.8), we have  $x_i[0] \in \mathcal{S}_\tau, \forall i \in \mathcal{N}$ . For  $k = 1$ , by (4.7), the values of normal agents can be expressed as

$$x^N[1] = \left( [I_n \ 0] + \begin{bmatrix} L_\tau^N[k] \\ 0 \end{bmatrix} \right) z[0]. \quad (4.9)$$

By step 2 in MW-MSR, for any normal agent, if some  $l$ -hop neighbors are Byzantine and the values passed through these Byzantine neighbors are outside of  $[\min z^N[0], \max z^N[0]]$ , then they will be ignored. Thus, the right-hand side of (4.9) becomes convex combinations of values in the interval  $[\min z^N[0], \max z^N[0]] = \mathcal{S}_\tau$ . Then we have  $x_i[1] \in \mathcal{S}_\tau, \forall i \in \mathcal{N}$ .

Next, for  $k \geq 1$ , define two variables by

$$\begin{aligned} \bar{x}_\tau[k] &= \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ \underline{x}_\tau[k] &= \min(x^N[k], x^N[k-1], \dots, x^N[k-\tau]). \end{aligned} \quad (4.10)$$

We must prove that  $\bar{x}_\tau[k]$  is a nonincreasing function of  $k \geq 1$ . By (4.7), at time  $k \geq 2$ ,

each normal agent updates its value based on a convex combination of the neighbors' values from  $k-1$  to  $k-\tau$ . We know from step 2 of MW-MSR that the values outside of the interval determined by the normal agents' values will be ignored. Hence, we obtain  $x_i[k+1] \leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau])$  for any  $i \in \mathcal{N}$ . It also follows that

$$\begin{aligned} x_i[k] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ x_i[k-1] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ &\vdots \\ x_i[k+1-\tau] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) \end{aligned}$$

for any  $i \in \mathcal{N}$ . Therefore,  $\bar{x}_\tau[k]$  is nonincreasing in time as

$$\begin{aligned} \bar{x}_\tau[k+1] &= \max(x^N[k+1], x^N[k], \dots, x^N[k+1-\tau]) \\ &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) = \bar{x}_\tau[k]. \end{aligned}$$

We can similarly prove that  $\underline{x}_\tau[k]$  is nondecreasing in time. This indicates that for  $k \geq 2$ , we have  $x_i[k] \in \mathcal{S}_\tau$  for  $i \in \mathcal{N}$ . Thus, we have shown the safety condition.

Next, we show the convergence. As shown above,  $\bar{x}_\tau[k]$  and  $\underline{x}_\tau[k]$  are monotone and bounded, and thus both of their limits exist and are denoted by  $\bar{x}_\tau^*$  and  $\underline{x}_\tau^*$ , respectively. We claim that the limits satisfy  $\bar{x}_\tau^* = \underline{x}_\tau^*$ , i.e., consensus is achieved. We prove by contradiction and assume that  $\bar{x}_\tau^* > \underline{x}_\tau^*$ .

Recall that  $\alpha$  lower bounds the nonzero entries of  $\Gamma[k]$ . Choose  $\epsilon_0 > 0$  small enough that  $\bar{x}_\tau^* - \epsilon_0 > \underline{x}_\tau^* + \epsilon_0$ . Fix

$$\epsilon < \frac{\epsilon_0 \alpha^{(\tau+1)n_N}}{(1 - \alpha^{(\tau+1)n_N})}, \quad 0 < \epsilon < \epsilon_0. \quad (4.11)$$

Define the sequence  $\{\epsilon_\gamma\}$  by

$$\epsilon_{\gamma+1} = \alpha \epsilon_\gamma - (1 - \alpha) \epsilon, \quad \gamma = 0, 1, \dots, (\tau+1)n_N - 1.$$

So we have  $0 < \epsilon_{\gamma+1} < \epsilon_\gamma$  for all  $\gamma$ . In particular, they are positive because by (4.11),

it holds that

$$\begin{aligned}\epsilon_{(\tau+1)n_N} &= \alpha^{(\tau+1)n_N} \epsilon_0 - \sum_{m=0}^{(\tau+1)n_N-1} \alpha^m (1-\alpha) \epsilon \\ &= \alpha^{(\tau+1)n_N} \epsilon_0 - (1-\alpha)^{(\tau+1)n_N} \epsilon > 0.\end{aligned}$$

Take  $k_\epsilon \in \mathbb{Z}_+$  such that  $\bar{x}_\tau[k] < \bar{x}_\tau^* + \epsilon$  and  $\underline{x}_\tau[k] > \underline{x}_\tau^* - \epsilon$  for  $k \geq k_\epsilon$ . Such  $k_\epsilon$  exists due to the convergence of  $\bar{x}_\tau[k]$  and  $\underline{x}_\tau[k]$ . Then we can define the two disjoint sets as

$$\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma) = \{j \in \mathcal{N} : x_j[k_\epsilon + \gamma] > \bar{x}_\tau^* - \epsilon_\gamma\},$$

$$\mathcal{Z}_{2\tau}(k_\epsilon + \gamma, \epsilon_\gamma) = \{j \in \mathcal{N} : x_j[k_\epsilon + \gamma] < \underline{x}_\tau^* + \epsilon_\gamma\}.$$

Next, we show that one of the two sets becomes empty in a finite number of steps, which contradicts the assumption on  $\bar{x}_\tau^*$  and  $\underline{x}_\tau^*$  being the limits. Consider the set  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$ . Due to the definition of  $\bar{x}_\tau[k]$  and its limit  $\bar{x}_\tau^*$ , one or more normal nodes are contained in the union of the sets  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$  for  $0 \leq \gamma \leq \tau + 1$ . We claim that  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  is in fact nonempty. To prove this, it is sufficient to show that if a normal node  $j$  is not in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$ , then it is not in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma + 1, \epsilon_{\gamma+1})$  for  $\gamma = 0, \dots, \tau$ .

Suppose that node  $j$  satisfies  $x_j[k_\epsilon + \gamma] \leq \bar{x}_\tau^* - \epsilon_\gamma$ . Every normal node updates its value to a convex combination of the multi-hop neighbors' values at the current or previous times. Though such neighbors may be Byzantine here, the ones with values greater than  $\bar{x}_\tau[k_\epsilon + \gamma]$  are ignored in step 2 of the MW-MSR algorithm. Hence, the value of node  $j$  at the next time step is bounded as

$$\begin{aligned}x_j[k_\epsilon + \gamma + 1] &\leq (1-\alpha)\bar{x}_\tau[k_\epsilon + \gamma] + \alpha(\bar{x}_\tau^* - \epsilon_\gamma) \\ &\leq (1-\alpha)(\bar{x}_\tau^* + \epsilon) + \alpha(\bar{x}_\tau^* - \epsilon_\gamma) \\ &\leq \bar{x}_\tau^* - \alpha\epsilon_\gamma + (1-\alpha)\epsilon = \bar{x}_\tau^* - \epsilon_{\gamma+1}.\end{aligned}\tag{4.12}$$

It thus follows that node  $j$  is not in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma + 1, \epsilon_{\gamma+1})$ . This means that the cardinality of the set  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$  is nonincreasing for  $\gamma = 0, \dots, \tau + 1$ . The same holds for  $\mathcal{Z}_{2\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$ , hence  $\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)$  is nonempty too.

We next show that one of these two sets in fact becomes empty in finite time. Since  $\mathcal{G}$  is  $(f+1)$ -strongly robust with  $l$  hops with respect to any set  $\mathcal{F}$  satisfying the  $f$ -local model, the normal network  $\mathcal{G}_N$  is guaranteed to be  $(f+1)$ -robust with  $l$  hops with respect to the set  $\mathcal{A}$  (i.e., the set of adversarial nodes). Therefore, between the two

nonempty disjoint sets  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  and  $\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)$ , one of them has a normal agent with at least  $f + 1$  independent normal paths originating from normal nodes outside.

Suppose that  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  has this property and let  $i$  be the node in this set which has at least  $f + 1$  independent normal paths originating from the normal nodes outside  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$ . By the argument above, these normal nodes will not be in  $\mathcal{Z}_{1\tau}(k_\epsilon + \gamma, \epsilon_\gamma)$  for  $0 \leq \gamma \leq \tau$ . By step 2 of the MW-MSR algorithm, one of these normal neighbors will be used in the updates of node  $i$  at any time since node  $i$  can only remove the values of which the cardinality of the minimum message cover is  $f$ . In particular, when node  $i$  makes an update at time  $k_\epsilon + \tau$ , a normal node's delayed value is used, upper bounded by  $\bar{x}_\tau^* - \epsilon_\tau$ . It thus follows that, at time  $k_\epsilon + \tau$ , when node  $i$  makes an update, its value can be bounded as

$$x_i[k_\epsilon + \tau + 1] \leq (1 - \alpha)\bar{x}_\tau[k_\epsilon + \tau] + \alpha(\bar{x}_\tau^* - \epsilon_\tau).$$

By (4.12), we have  $x_i[k_\epsilon + \tau + 1] \leq \bar{x}_\tau^* - \epsilon_{\tau+1}$ . We can conclude that if node  $i$  in  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$  has  $f + 1$  independent normal paths originating from the normal nodes outside the set, then it goes outside of  $\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  after  $\tau + 1$  steps. Consequently,  $\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  has the cardinality smaller than that of  $\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)$ , that is,  $|\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})| < |\mathcal{Z}_{1\tau}(k_\epsilon, \epsilon_0)|$ . Likewise, it follows that if  $\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)$  has a node having at least  $f + 1$  independent normal paths originating from the normal nodes outside, then  $|\mathcal{Z}_{2\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})| < |\mathcal{Z}_{2\tau}(k_\epsilon, \epsilon_0)|$ .

Since there are only  $n_N$  normal nodes, we can repeat the steps above until one of the sets  $\mathcal{Z}_{1\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  and  $\mathcal{Z}_{2\tau}(k_\epsilon + \tau + 1, \epsilon_{\tau+1})$  becomes empty, and it takes no more than  $(\tau + 1)n_N$  steps. Once the set becomes empty, it remains so indefinitely. This contradicts the assumption that  $\bar{x}_\tau^*$  and  $\underline{x}_\tau^*$  are the limits. Therefore, we obtain  $\bar{x}_\tau^* = \underline{x}_\tau^*$ . ■

Note that  $f$ -total model can be viewed as a special case of  $f$ -local model, and thus the necessary and sufficient conditions stated in Theorem 4.4.1 is also valid for  $f$ -total Byzantine model. We emphasize that  $f$ -local is more suitable for a large scale network because the model locally focuses on each node with a small  $f$ -total model. If the locations of adversary nodes are spread in a more uniform way over the network, then the total tolerable number of adversary can be very large. However, with the same number of adversary nodes for the  $f$ -total model, the network requires more connections. See the example in Fig. 4.1(a). Asynchronous Byzantine consensus can

be reached using the MW-MSR algorithm in this network for the 1-local model when the normal nodes communicate with their two-hop neighbors. For instance, the two nodes in red can be Byzantine under such a model.

### 4.4.3 Comparison with Conventional Methods

In this part, we outline the advantages of the proposed method compared to the conventional works. We would like to highlight the following four aspects: (i) Our algorithm does not use “rounds” that can cause possibly large delays in forming consensus as we mentioned before; (ii) we consider the  $f$ -local model, which is more adversarial than the  $f$ -total model; (iii) our graph condition is tight and generalizes the ones in the literature for both synchronous and asynchronous cases; (iv) the algorithm is computationally more efficient than existing ones.

#### 4.4.3.1 Advantages in Threat Models and Graph Conditions

In what follows, we discuss further details about these advantages. Specifically, the  $f$ -total model in [94; 102] can be viewed as a special case of the  $f$ -local model, and thus the condition stated in Theorem 4.4.1 is also sufficient for the  $f$ -total Byzantine model. We emphasize that the  $f$ -local model is more suitable for large-scale networks especially if the adversary nodes are well spread within the system. Then even if the total number of such nodes is large, under the  $f$ -local model, we can take a small  $f$ . If we treat such a system based on the  $f$ -total model, the requirement on network connectivity will be much larger.

We remark that the condition of the graph being  $(f + 1)$ -strongly robust makes the system sufficiently resilient to mitigate the influence of asynchrony and delays in communication. Here, it is important to note that this graph condition is the same for the case of synchronous updates using multi-hop communication. The condition first appeared in [102], which studies the synchronous update case of approximate Byzantine consensus with  $l$ -hop communication. It also appeared in [94; 109] for synchronous and asynchronous schemes, respectively, which study the special case of  $l \geq l^*$ .

Similar to the discussion in Section 4.3.2, based on Theorem 4.4.1, we can obtain a tight result for the resilient consensus under the  $f$ -total/local malicious model for the asynchronous update scheme since the malicious model is as a special case of Byzantine model. For this case, it has been known that  $(2f + 1)$ -robustness with  $l$  hops is a

sufficient condition (see Table 1.1) while  $(f + 1, f + 1)$ -robustness with  $l$  hops is a necessary condition; see, e.g., [21; 56] for the one-hop case and [125] for the multi-hop case. The following corollary is immediate in view of Proposition 4.3.2.

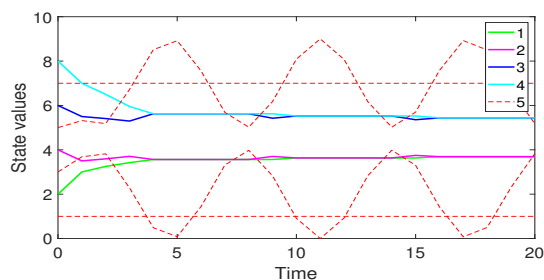
**Corollary 4.4.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous MW-MSR algorithm with parameter  $f$ . Under the  $f$ -local malicious model, resilient asymptotic consensus is achieved with safety interval (4.8) if  $\mathcal{G}$  is  $(f + 1)$ -strongly robust with  $l$  hops and only if  $\mathcal{G}$  is  $(f + 1, f + 1)$ -robust with  $l$  hops.*

#### 4.4.3.2 Advantages in Computational Complexity

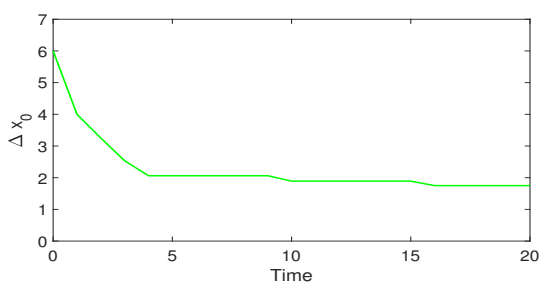
Finally, we highlight that the proposed MW-MSR algorithm is more light weighted and efficient in terms of complexity in comparison with the algorithm in [94]. The asynchrony settings of the two algorithms are similar, but we exploit the power of the MW-MSR algorithm to filter the possible extreme values caused by Byzantine nodes.

To show this, we first outline the structure of the algorithm in [94]. Intuitively, the algorithm there can be divided into two parts: Verification of the received values and the MSR algorithm (called Filter and Average algorithm). More specifically, each node is required to send its value to the entire network at the beginning of each asynchronous round. Then in the verification part, for each possible set of Byzantine nodes  $\mathcal{F}$  (satisfying the  $f$ -total model), each normal node  $i$  receives values from the neighbors and for each received value, it verifies if this value is consistent in the paths excluding the nodes in set  $\mathcal{F}$ . Then node  $i$  waits for enough verified values with round  $r$  as the input for the Filter-and-Average part. There, each normal node updates its value using these inputs. To satisfy the safety condition, the normal nodes will remove the largest and smallest values of which the message cover is  $f$ .

The Filter-and-Average algorithm and our MW-MSR algorithm are similar, but the main difference is that the former algorithm uses verified values with round  $r$  as input and the MW-MSR algorithm uses the most recent values of  $l$ -hop neighbors on each  $l$ -hop path. Hence, all the operations before the Filter-and-Average algorithm in the main algorithm for verification in [94] are additional in terms of computation. Besides, the verification algorithm there should be executed for each possible set  $\mathcal{F}$ , i.e., at least  $\binom{n}{f}$  executions of the main algorithm on each node for each asynchronous round. Although this can be executed in parallel threads (one  $\mathcal{F}$  per thread), it still requires



(a) The states of all nodes.

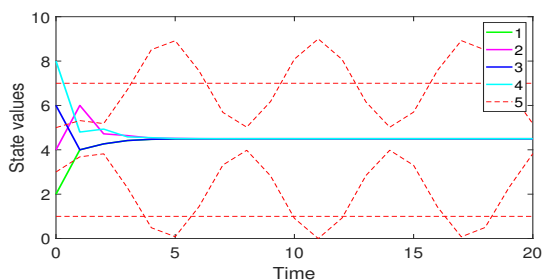


(b) Consensus error.

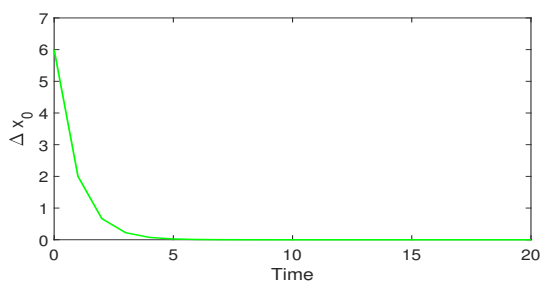
Figure 4.3: Time responses of the synchronous one-hop W-MSR algorithm.

a huge amount of computation resources and memory to verify and store the values from the nodes in the entire network. Even for the case of  $l \geq l^*$ , the computational complexity of the MW-MSR algorithm is less than the algorithm in [94].

Why the verification part is essential for the algorithm in [94] is partially because of their asynchrony setting based on rounds and the verification can prevent the duplication of messages of normal nodes with same round  $r$ . In contrast, in our asynchrony setting, we need not guarantee the correctness of a received value and we use the most recent value for each  $l$ -hop path (hence, no duplication). Thus, we can fully utilize the ability of MW-MSR algorithm to filter the extreme values that could possibly be manipulated by Byzantine nodes. The trade-off is that we can only guarantee  $\Delta x_\tau[k] = \max z^N[k] - \min z^N[k]$  to be nonincreasing, while for the round based asynchrony,  $\Delta x[r] = \max x^N[r] - \min x^N[r]$  is guaranteed to be nonincreasing. Besides, since our algorithm is iterative and only requires values and topology information up to  $l$  hops away, our algorithm is more light-weighted compared to that in [94].



(a) The states of all nodes.



(b) Consensus error.

Figure 4.4: Time responses of the synchronous two-hop MW-MSR algorithm.

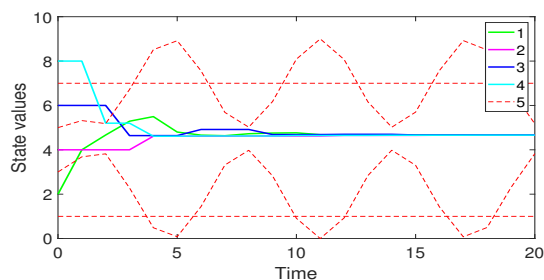
## 4.5 Numerical Example

In this section, we carry out simulations for the asynchronous MW-MSR algorithm with time delays.

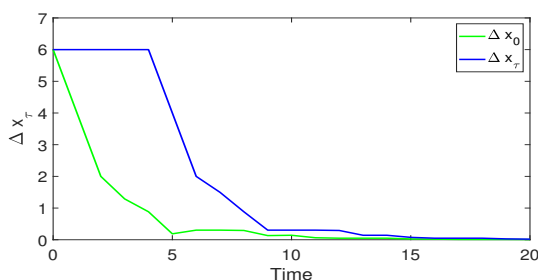
Consider the network in Fig. 4.1(b). This graph is not 2-strongly robust with one hop, but is 2-strongly robust with 2 hops. Suppose that node 5 is Byzantine and is capable to send four different values to its four neighbors (indicated by red dashed lines in Fig. 4.3(a)). Let the initial normal states be  $x^N[0] = [2 \ 4 \ 6 \ 8]^T$ . According to [56], [113], this graph does not meet the condition for 1-total Byzantine model even for synchronous updates. Therefore, consensus among normal nodes cannot be reached in this network with one-hop communication. The results for state values and consensus error ( $\Delta x_0[k] = \max x^N[k] - \min x^N[k]$ ) are given in Fig. 4.3. Then, we examine the synchronous two-hop MW-MSR algorithm under the same attack. The results for state values and consensus error are given in Fig. 4.4. Observe that resilient consensus is achieved.

Lastly, we perform simulations for the asynchronous two-hop MW-MSR algorithm under the same attack. Let the normal nodes update in an asynchronous periodic sense, which means that for nodes 1, 2, 3, and 4, they update in every 1, 4, 3, 2 steps,





(a) The states of all nodes.



(b) Consensus error.

Figure 4.5: Time responses of the asynchronous two-hop MW-MSR algorithm.

respectively (all nodes update once at  $k = 0$ ). The time delay for the values from one-hop neighbors and two-hop neighbors are set as 0 and 1 step, respectively. Thus, in the current setting, we can choose  $\tau = 4$ . The results are presented in Fig. 4.5. Observe that resilient consensus is achieved, delays have some effects and the convergence takes more time than the synchronous algorithm. We can also notice in Fig. 4.5(b) that the safety interval  $\Delta x_\tau[k] = \max z^N[k] - \min z^N[k]$  is nonincreasing.

## 4.6 Summary

We have solved the approximate Byzantine consensus problem under asynchronous updates and time delays in the communication between agents. Our approach is based on the multi-hop weighted MSR algorithm. We have also provided a tight necessary and sufficient graph condition for the network using the MW-MSR algorithm. An important implication of our results is that under the  $f$ -total/local Byzantine model, the graph condition remains the same even if the algorithm becomes asynchronous and the communication is subject to time delays. Moreover, our algorithm is iterative and only require local information and topology for each node, and hence it is more light weighted and distributed compared to the conventional flooding-based algorithms.

## Chapter 5

# Resilient Quantized Consensus with Multi-hop Communication

In this chapter, we study the problem of resilient quantized consensus where agents take integer-valued states. To solve this problem, we develop a quantized version of the MW-MSR algorithm. We provide necessary and sufficient conditions for our algorithm to achieve resilient quantized consensus for synchronous/asynchronous updates under malicious/Byzantine attacks. Compared to existing literature, our algorithm has a tighter graph condition and we provide a unified analysis for different network settings.

Related literature for resilient quantized consensus (and exact consensus) is summarized in Table 5.1 ([\*] represents our work). We extend the results in [22] under the malicious model to a multi-hop setting and provide extra analysis for Byzantine attacks. We also provide a tighter sufficient condition than the condition given in [22] for asynchronous resilient quantized consensus under malicious attacks.

The resilient quantized consensus works using the flooding communication [26; 52; 109; 117] correspond to our unbounded path length case ( $l \geq l^*$ ). On the other hand, we deal with the general  $l$ -hop case. With different network settings, the works in Table 5.1 have different graph conditions. For the synchronous binary consensus under the malicious model, our graph condition is equivalent to [53] in the case of unbounded path length. Details are given in Section 5.2. The recent work [117] studies binary Byzantine consensus in asynchronous updates with delays and derives the same conditions as the two in [26]. We claim that the graph conditions of unbounded path length case for our algorithm coincides with the conditions in [26; 109; 117].

Table 5.1: Related works for resilient quantized consensus.

		Synchronous	Asynchronous
Malicious	Undirected $\mathcal{G}$	[22; 52; 53], [*]	[22], [*]
	Directed $\mathcal{G}$		
Byzantine	Undirected $\mathcal{G}$	[26], [*]	[117], [*]
	Directed $\mathcal{G}$	[109], [*]	[*]

The rest of this chapter is organized as follows. Section 5.1 outlines the system model. Sections 5.2 and 5.3 derive conditions under which the QMW-MSR algorithms guarantee resilient quantized consensus under synchronous and asynchronous updates, respectively. Section 5.4 concludes this chapter.

## 5.1 Problem Formulation

### 5.1.1 Quantized Consensus and Update Rule

Consider a time-invariant network modeled by the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  composed of  $n$  nodes. The node set  $\mathcal{V}$  is partitioned into the set of normal nodes  $\mathcal{N}$  and the set of adversary nodes  $\mathcal{A}$ , where  $|\mathcal{N}| = n_N$  and  $|\mathcal{A}| = n_A$ . The latter set is unknown to the normal nodes at time step  $k = 0$ .

Recall that agents communicate with each other according to the communication model indicated in Section 2.3.1. Note that in quantized consensus, the function  $g_i(x)$  in (2.8) is quantized. When there is no attack in the network, we can employ the common consensus update rule for (2.8) (e.g., [82]), which can be given in the compact form as

$$\begin{aligned} x[k+1] &= x[k] + u[k], \\ u[k] &= -L[k]x[k], \end{aligned} \tag{5.1}$$

where  $x[k] \in \mathbb{R}^n$  and  $u[k] \in \mathbb{R}^n$  are the state vector and control input vector respectively, and  $L[k]$  is the Laplacian matrix of the  $l$ -th power of  $\mathcal{G}$  determined by the messages  $m_{ij}[k], i \in \mathcal{V}$  and  $j \in \mathcal{N}_i^{l-}$ .

In many multi-agent system applications, state values of agents are preferred to be integers due to digitalization or limited memory of the agents. In this chapter, we focus on the quantized consensus using the following quantization function  $Q : \mathbb{R} \rightarrow \mathbb{Z}$  to

transform the real-valued input in (5.1) to integers, which is studied in [5], [22]. Hence, the states and the inputs are constrained as  $x_i[k] \in \mathbb{Z}$  and  $u_i[k] \in \mathbb{Z}$  for  $i \in \mathcal{V}$ .

$$Q(y) = \begin{cases} \lfloor y \rfloor & \text{with probability } p(y), \\ \lceil y \rceil & \text{with probability } 1 - p(y), \end{cases} \quad (5.2)$$

where  $p(y) = \lceil y \rceil - y$ , and  $\lfloor * \rfloor$  denotes the floor function. Then based on (5.1), we can write the quantized control input for normal node  $i$  as

$$u_i[k] = Q\left(\sum_{j \in \mathcal{N}_i^{l-}} a_{ij}[k] x_j[k]\right), \quad (5.3)$$

where  $a_{ij}[k]$  is the  $(i, j)$ th entry of the adjacency matrix  $A[k]$  of graph  $\mathcal{G}^l$  at time  $k$ . Then we denote by  $x^N[k] \in \mathbb{Z}^{n_N}$  and  $x^A[k] \in \mathbb{Z}^{n_A}$  the state vectors of normal nodes and adversary nodes respectively.

Note that the probabilistic quantizer equipped on each agent is independent and each node will choose the floor or ceiling function at each time. Moreover, the probability  $p$  can be different on each node and each time as long as  $0 < p < 1$ . Thus, the control input (5.3) can be implemented in a distributed fashion.

Then we introduce the asynchrony in our algorithm [21], [22]. At each time  $k$ , normal node  $i$  may and may not update its value. If node  $i$  does not update, then  $x_i[k+1] = x_i[k]$ , i.e.,  $u_i[k] = 0$ . Denote by  $\mathcal{U}[k] \subset \mathcal{V}$  the set of agents updating at time  $k$ . The system is said to be synchronous if  $\mathcal{U}[k] = \mathcal{V}$  for all  $k$ , and otherwise it is asynchronous.

### 5.1.2 Threat Model

In this chapter, we consider the same adversary models studied in Chapters 3 and 4, i.e.,  $f$ -total/ $f$ -local malicious/Byzantine models. The malicious model is reasonable in applications such as wireless sensor networks and robotic networks, where neighbors' information is obtained by broadcast communication or vision sensors. Byzantine model is possible in point-to-point networks and is more adversarial than malicious model given that all malicious nodes are Byzantine, but not vice versa [62], [56].

As commonly done in the literature [56], [102], we assume that each normal node knows the value of  $f$  and the topology information of the graph up to  $l$  hops. We assume that each adversary node  $i$  cannot manipulate the path values in the messages

containing its own state  $x_i[k]$  and those that it relays as stated in Assumption 2.3.1.

### 5.1.3 Resilient Quantized Consensus and Algorithm

We now introduce the type of consensus among the normal agents to be sought in this chapter [22].

**Definition 5.1.1** *If for any possible sets and behaviors of the malicious agents and any state values of the normal nodes, the following two conditions are satisfied, then we say that the normal agents reach resilient quantized consensus:*

1. *Safety: There exists a bounded safety interval  $\mathcal{S}$  determined by the initial values of the normal agents such that  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .*
2. *Agreement: There exists a state  $x^* \in \mathcal{S}$  such that  $\lim_{k \rightarrow \infty} x_i[k] = x^*, \forall i \in \mathcal{N}$ . There exists a finite time  $k_a \geq 0$  such that  $\text{Prob}\{x^N[k_a] \in \mathcal{C} \mid x[0]\} = 1$ , where the consensus set  $\mathcal{C}$  is defined as*

$$\mathcal{C} = \{x \in \mathbb{Z}^{n_N} \mid x_1 = \dots = x_{n_N}\}.$$

Next, we introduce our resilient quantized consensus algorithm. It is a quantized version of our MW-MSR algorithm in previous work. We outline its structure in Algorithm 2. Intuitively speaking, for normal node  $i$ , it uses only the quantized values from its multi-hop neighbors and itself as the input for the algorithm. Then, in step 3, it updates using the randomized quantizer introduced before.

In this chapter, we are interested in finding the conditions on the network such that the above algorithm achieve resilient quantized consensus almost surely under different attack models. We consider the case of synchronous updates in Section 5.3. Then, in Section 5.4, we consider the more realistic situation using multi-hop settings, that is the asynchronous updates with time delays in the communication among agents.

## 5.2 Synchronous Networks

In this section, we analyze the performance of the QMW-MSR algorithm under synchronous updates, i.e., all the nodes update values synchronously at each time  $k$  ( $\mathcal{U}[k] = \mathcal{V}$  for all time  $k$ ).

**Algorithm 2:** QMW-MSR Algorithm

- 1) At each time  $k$ , normal node  $i$  sends its own message to nodes in  $\mathcal{N}_i^{l+}$ . Then, it obtains the quantized values of the nodes in  $\mathcal{N}_i^{l-}$  and itself, whose set is denoted by  $\mathcal{M}_i[k]$ , and sorts the values in  $\mathcal{M}_i[k]$  in an increasing order.
- 2) (a) Define two subsets of  $\mathcal{M}_i[k]$  based on the message values:

$$\overline{\mathcal{M}}_i[k] = \{m \in \mathcal{M}_i[k] : \text{value}(m) > x_i[k]\},$$

$$\underline{\mathcal{M}}_i[k] = \{m \in \mathcal{M}_i[k] : \text{value}(m) < x_i[k]\}.$$

- (b) Then, let  $\overline{\mathcal{R}}_i[k] = \overline{\mathcal{M}}_i[k]$  if the cardinality of a minimum cover of  $\overline{\mathcal{M}}_i[k]$  is less than  $f$ , i.e.,  $|\mathcal{T}^*(\overline{\mathcal{M}}_i[k])| < f$ . Otherwise, let  $\overline{\mathcal{R}}_i[k]$  be the largest sized subset of  $\overline{\mathcal{M}}_i[k]$  such that (i) for all  $m \in \overline{\mathcal{M}}_i[k] \setminus \overline{\mathcal{R}}_i[k]$  and  $m' \in \overline{\mathcal{R}}_i[k]$  we have  $\text{value}(m) \leq \text{value}(m')$ , and (ii) the cardinality of a minimum cover of  $\overline{\mathcal{R}}_i[k]$  is exactly  $f$ , i.e.,  $|\mathcal{T}^*(\overline{\mathcal{R}}_i[k])| = f$ .
  - (c) Similarly, let  $\underline{\mathcal{R}}_i[k] = \underline{\mathcal{M}}_i[k]$  if the cardinality of a minimum cover of  $\underline{\mathcal{M}}_i[k]$  is less than  $f$ , i.e.,  $|\mathcal{T}^*(\underline{\mathcal{M}}_i[k])| < f$ . Otherwise, let  $\underline{\mathcal{R}}_i[k]$  be the largest sized subset of  $\underline{\mathcal{M}}_i[k]$  such that (i) for all  $m \in \underline{\mathcal{M}}_i[k] \setminus \underline{\mathcal{R}}_i[k]$  and  $m' \in \underline{\mathcal{R}}_i[k]$  we have  $\text{value}(m) \geq \text{value}(m')$ , and (ii) the cardinality of a minimum cover of  $\underline{\mathcal{R}}_i[k]$  is exactly  $f$ , i.e.,  $|\mathcal{T}^*(\underline{\mathcal{R}}_i[k])| = f$ .
  - (d) Finally, let  $\mathcal{R}_i[k] = \overline{\mathcal{R}}_i[k] \cup \underline{\mathcal{R}}_i[k]$ .
- 3) Node  $i$  updates its value as follows:

$$x_i[k+1] = Q\left(\sum_{m \in \mathcal{M}_i[k] \setminus \mathcal{R}_i[k]} a_i[k] \text{value}(m)\right), \quad (5.4)$$

where  $a_i[k] = 1/(|\mathcal{M}_i[k] \setminus \mathcal{R}_i[k]|)$ .

### 5.2.1 Matrix Representation

For ease of notation in our analysis, reorder the agents so that the normal agents take indices  $1, \dots, n_N$  and the malicious agents take indices  $n_N + 1, \dots, n$ . Then the state vector and control input vector can be written as

$$x[k] = \begin{bmatrix} x^N[k] \\ x^A[k] \end{bmatrix}, \quad u[k] = \begin{bmatrix} u^N[k] \\ u^A[k] \end{bmatrix}. \quad (5.5)$$

Regarding the control inputs  $u^N[k]$  and  $u^A[k]$ , the normal agents follow (5.1) while the adversary agents may not. Hence, they can be expressed as

$$\begin{aligned} u^N[k] &= Q(-L^N[k]x[k]), \\ u^A[k] &: \text{arbitrary}, \end{aligned} \tag{5.6}$$

where  $L^N[k] \in \mathbb{R}^{n_N \times n}$  is the matrix formed by the first  $n_N$  rows of  $L[k]$  associated with normal agents. The row sums of this matrix  $L^N[k]$  are zero as in  $L[k]$ .

Thus, we can rewrite the system as

$$x[k+1] = Q\left(\left(I_n - \begin{bmatrix} L^N[k] \\ 0 \end{bmatrix}\right)x[k]\right) + \begin{bmatrix} 0 \\ I_{n_A} \end{bmatrix}u^A[k]. \tag{5.7}$$

### 5.2.2 Consensus Analysis

In this section, we characterize network conditions to guarantee resilient consensus using the QMW-MSR algorithm.

First, we present the safety condition for resilient quantized consensus. For the agents using the synchronous QMW-MSR algorithm, the safety interval is given by

$$x_i[k] \in \mathcal{S} = [\min x^N[0], \max x^N[0]], \forall i \in \mathcal{N}, k \in \mathbb{Z}_+. \tag{5.8}$$

Now we are ready to provide a necessary and sufficient condition for resilient consensus using the QMW-MSR algorithm. The following theorem is the main contribution of this chapter.

**Theorem 5.2.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the synchronous QMW-MSR algorithm with parameter  $f$ . Under the  $f$ -total malicious model, resilient quantized consensus is achieved almost surely if and only if the network topology is  $(f+1, f+1)$ -robust with  $l$  hops.*

To establish quantized consensus in this probabilistic setting, we need the following lemma, which is proved to be sufficient for guaranteeing resilient quantized consensus almost surely [22].

**Lemma 5.2.1** *Consider the network modeled by graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with the QMW-MSR algorithm. Suppose that the following three conditions are satisfied for the normal nodes:*

- C1) There exists a bounded set  $\mathcal{S}$  determined by the initial states of the normal nodes such that  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .
- C2) For each state  $x[k] = x_0$  at time  $k$ , there exist a finite time  $k_b$  such that  $\text{Prob}\{x^N[k+k_b] \in \mathcal{C} \mid x[k] = x_0\} > 0$ .
- C3) If  $x[k] \in \mathcal{C}$ , then  $x[k'] \in \mathcal{C}, \forall k' > k$ .

Then, the network reaches quantized consensus almost surely.

Intuitively, if the algorithm satisfies these conditions for normal agents, then, the scenarios for reaching consensus occur infinitely often with high probability. This is because the probability for such an event to occur is positive based on the condition (C2). Then, once normal agents reach consensus, consensus is preserved infinitely by (C3).

*Proof of Theorem 5.2.1:* (Necessity) If  $\mathcal{G}$  is not  $(f+1, f+1)$ -robust with  $l$  hops, then there are nonempty, disjoint subsets  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ , such that none of the conditions in Definition 3.3.2 holds. Suppose the initial value of each node in  $\mathcal{V}_1$  is  $a$  and each node in  $\mathcal{V}_2$  is  $b$ , with  $a < b$ . Let all other nodes have initial values as  $\lfloor (a+b)/2 \rfloor$ . Since  $|\mathcal{Z}_{\mathcal{V}_1}^{f+1}| + |\mathcal{Z}_{\mathcal{V}_2}^{f+1}| \leq f$ , suppose all nodes in  $\mathcal{Z}_{\mathcal{V}_1}^{f+1}$  and  $\mathcal{Z}_{\mathcal{V}_2}^{f+1}$  are malicious and take constant values. Then there is still at least one normal node in both  $\mathcal{V}_1$  and  $\mathcal{V}_2$  since  $|\mathcal{Z}_{\mathcal{V}_1}^{f+1}| < |\mathcal{V}_1|$  and  $|\mathcal{Z}_{\mathcal{V}_2}^{f+1}| < |\mathcal{V}_2|$  respectively. Then these normal nodes in  $\mathcal{V}_1$  and  $\mathcal{V}_2$  remove all the values of incoming neighbors outside of their respective sets since the message cover of these values has cardinality equal to  $f$  or less. According to the QMW-MSR algorithm, such normal nodes will keep their values and consensus cannot be achieved.

(Sufficiency) We show that the conditions (C1)-(C3) in Lemma 5.2.1 hold. To prove the safety condition (C1), let  $\bar{x}[k]$  and  $\underline{x}[k]$  to be the maximum and minimum values of the normal nodes at time  $k$ , respectively, i.e.,  $\bar{x}[k] = \max x^N[k]$ ,  $\underline{x}[k] = \min x^N[k]$ . Observe that the values used in the QMW-MSR update rule always lie within the interval  $[\underline{x}[k], \bar{x}[k]]$ . Moreover, the update rule in (5.7) is a quantized convex combination of the values in the interval  $[\underline{x}[k], \bar{x}[k]] \subset \mathcal{S}$ . Hence,  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .

Then we prove condition (C2). We can see from above that both  $\bar{x}[k]$  and  $\underline{x}[k]$  are monotone and bounded sequences, thus there is a finite time  $k_c$  such that they both reach their final values with probability 1. Denote the final values of  $\bar{x}[k]$  and  $\underline{x}[k]$  by



$\bar{x}^*$  and  $\underline{x}^*$ , respectively. We will prove by contradiction to show that  $\bar{x}^* = \underline{x}^*$ , thus the normal nodes will reach consensus. Suppose that  $\bar{x}^* > \underline{x}^*$ . When  $k \geq k_c$ , we can define the following sets

$$\mathcal{Z}_1[k] = \{i \in \mathcal{V} : x_i[k] \geq \bar{x}^*\},$$

$$\mathcal{Z}_2[k] = \{i \in \mathcal{V} : x_i[k] \leq \underline{x}^*\}.$$

We first show that with positive probability, the normal agents in  $\mathcal{Z}_1[k]$  decrease their values, and the normal agents in  $\mathcal{Z}_2[k]$  increase their values at the next time step. Clearly,  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$  are nonempty and disjoint by assumption. Since the network is  $(f+1, f+1)$ -robust with  $l$  hops with respect to any set satisfying the  $f$ -total model and there are at most  $f$  malicious nodes. Thus, there always exists a normal node  $i$  either in  $\mathcal{Z}_1[k]$  or  $\mathcal{Z}_2[k]$  with  $f+1$  independent paths originating from the nodes outside of its respective set and these paths do not contain any intermediate node in the adversary set  $\mathcal{A}$ . Without loss of generality, we suppose that the normal agent  $i$  in  $\mathcal{Z}_1[k]$  has this property. Since at time  $k$ , normal nodes in  $\mathcal{Z}_1[k]$  reach the value  $\bar{x}^*$ , we have  $x_i[k] = \bar{x}^*$ . Then node  $i$  updates its value as

$$x_i[k+1] \leq Q((1-\alpha)\bar{x}^* + \alpha(\bar{x}^* - 1)) = Q(\bar{x}^* - \alpha). \quad (5.9)$$

By (5.2), the quantizer takes the floor function as  $Q(\bar{x}^* - \alpha) = \bar{x}^* - 1$  with probability  $\alpha$ . Thus, with positive probability, we have  $x_i[k+1] \leq \bar{x}^* - 1$ . This indicates that with positive probability, one of the normal agents taking the maximum value  $\bar{x}^*$  decreases its value by at least one. Similarly, if the normal node  $i$  is in  $\mathcal{Z}_2[k]$ , then with positive probability, its quantizer chooses the ceiling function, then its value will increase above  $\underline{x}^*$ .

Next, we show that with positive probability, none of the normal nodes in  $\mathcal{V} \setminus \mathcal{Z}_1[k]$  enters  $\mathcal{Z}_1[k+1]$  at the next time step. If the normal node  $i$  is in  $\mathcal{V} \setminus \mathcal{Z}_1[k]$  at time  $k$ , it is upper bounded by  $\bar{x}^* - 1$ . According to the QMW-MSR algorithm, all values bigger than  $\bar{x}^*$  will be discarded by node  $i$ , and the inequality (5.9) holds for node  $i$ . Therefore, with positive probability  $\alpha$ , node  $i$  will not enter  $\mathcal{Z}_1[k+1]$ . Follow the similar reasoning, we can conclude that with positive probability, none of the normal nodes in  $\mathcal{V} \setminus \mathcal{Z}_2[k]$  enters  $\mathcal{Z}_2[k+1]$  at the next time step.

Combining the two arguments, we conclude that for any  $k \geq k_c + n_N$ , the number of normal agents in one of the sets  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$  is zero with positive probability

because the number of normal nodes is  $n_N$ . This is a contradiction and proves (C2).

Lastly, we show condition (C3) holds. Assume that the normal nodes have reached the common value  $x^*$  at time  $k$ . Since there are at most  $f$  malicious nodes, according to the QMW-MSR algorithm, all malicious nodes  $j$  with values  $x_j[k] \neq x^*$  are ignored by the normal nodes. Thus,  $x_i[k+1] = x^*, \forall i \in \mathcal{N}$ . Complete the proof. ■

Note that this necessary and sufficient condition for resilient quantized consensus,  $(f+1, f+1)$ -robustness with  $l$  hops, is consistent with the graph condition for resilient asymptotic real-valued consensus with  $l$ -hop communication studied in [125]. There, the updating rules there are without any randomization and that work is more delicate to reveal the graph condition for MSR based algorithms to guarantee resilient consensus in a multi-hop setting. While this paper studies agents taking quantized values and the convergence is in finite time in a probabilistic sense.

It is noted that our approach can be applied to the binary valued consensus case [62], [34], [52], [53], [117]. As long as the initial states of all agents are restricted to 0 and 1, the safety interval in (5.8) indicates that the normal agents' values will remain binary and come to agreement eventually. All results presented in this chapter remain true for the binary case.

As mentioned earlier, the authors of [52; 53] study the synchronous binary Byzantine consensus under malicious attacks and they provide a necessary and sufficient graph condition for their algorithm to achieve binary consensus. We note that our graph condition is equivalent to theirs for the case of unbounded hops, while our algorithm also contains the general  $l$ -hop case. See the discussions in Section 3.6.2. There, we have shown more details about the relation between our condition and theirs.

## 5.3 Asynchronous Network

In this section, we analyze the QMW-MSR algorithm under asynchronous updates.

### 5.3.1 Randomized Updates versus Deterministic Updates

In this part, we first analyze the QMW-MSR algorithm under asynchronous randomized updates without delays.

Recall that we denote the set of normal nodes updating at time  $k$  is represented by  $\mathcal{U}[k]$ . For deterministic updates, we assume that each normal node  $i$  makes an update

at least once in  $\bar{k}$  time steps, that is,

$$\bigcup_{m=k}^{k+\bar{k}-1} \mathcal{U}[m] = \mathcal{N} \text{ for } k \in \mathbb{Z}_+, \quad (5.10)$$

while adversary nodes may deviate from this update setting.

For randomized updates, we assume that each normal node  $i$  makes an update at time  $k \geq 0$  with probability  $p_i \in (0, 1]$  in an i.i.d. fashion. That is, for node  $i$ , at each time  $k$

$$\text{Prob}\{i \in \mathcal{U}[k]\} = p_i, \text{ Prob}\{i \notin \mathcal{U}[k]\} = 1 - p_i. \quad (5.11)$$

Note that with randomized updates, the algorithm remains fully distributed since the probabilities  $p_i$  at each node can be different.

An advantage of randomized updates is that the malicious nodes cannot predict the update times of the normal nodes in advance. Moreover, there is always nonzero probability that all normal nodes in the system update their states simultaneously at each time  $k$ . This feature enables us to establish a stronger result than that for the deterministic case.

The following theorem shows that for the randomized updates, the necessary and sufficient condition for resilient quantized consensus is the same as that for the synchronous case in Theorem 5.2.1.

**Theorem 5.3.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the randomized QMW-MSR algorithm with parameter  $f$ . Under the  $f$ -total malicious model, resilient quantized consensus is achieved almost surely if and only if the network topology is  $(f + 1, f + 1)$ -robust with  $l$  hops.*

*Proof:* (Necessity) We omit the proof as the synchronous network is a special case of asynchronous network, thus the necessary condition for synchronous case holds for asynchronous case.

(Sufficiency) We show that the conditions (C1)–(C3) in Lemma 5.2.1 hold. It is easy to see that conditions (C1) and (C3) hold in the randomized updates too. Therefore, we only need to show (C2). We know from (C1) that  $\bar{x}[k]$  and  $\underline{x}[k]$  will reach their final values  $\bar{x}^*$  and  $\underline{x}^*$  at some time  $k_c$ , respectively. Like the proof of Theorem 5.2.1,

we prove (C2) by contradiction. Assume  $\bar{x}^* > \underline{x}^*$ , then the sets  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$  are disjoint and nonempty.

We can first prove that, at each  $k \geq k_c$ , a normal node in at least one of these sets goes out from the corresponding set at the next time step with positive probability. Note that in the asynchronous randomized updates, the probability for any normal node to update at this time  $k$  is positive. Besides, since the graph is  $(f + 1, f + 1)$ -robustness with  $l$  hops, there is a normal node  $i$  in either  $\mathcal{Z}_1[k]$  or  $\mathcal{Z}_2[k]$  which has the  $(f + 1)$ -reachability. Thus, with positive probability, node  $i$  updates at time  $k$  and it will go out of  $\mathcal{Z}_1[k]$  or  $\mathcal{Z}_2[k]$  depending on which set it belongs to.

Then we show that with positive probability, none of the normal nodes outside  $\mathcal{Z}_j[k]$  enters  $\mathcal{Z}_j[k + 1]$  for  $j = 1, 2$ . Consider the normal node  $i$  in  $\mathcal{V} \setminus \mathcal{Z}_1[k]$  at time  $k$ , with positive probability, it updates at this time and will not enter  $\mathcal{Z}_1[k + 1]$  by the same reasoning in the proof of Theorem 5.2.1.

Lastly, we can conclude that for any  $k \geq k_c + n_N$ , one of the two sets,  $\mathcal{Z}_1[k]$  or  $\mathcal{Z}_2[k]$ , contains no normal node with positive probability, which contradicts  $\bar{x}^* > \underline{x}^*$ . ■

### 5.3.2 Asynchronous Updates with Delays

Then we see how the QMW-MSR algorithm will perform under asynchronous updates with delays.

We employ the control input taking account of possible delays in the values from the neighbors as

$$u_i[k] = Q \left( \sum_{j \in \mathcal{N}_i^{l-}} a_{ij}[k] x_j^P[k - \tau_{ij}^P[k]] \right), \quad (5.12)$$

where  $\tau_{ij}^P[k] \in \mathbb{Z}_+$  denotes the delay in this  $(j, i)$ -path  $P$  at time  $k$  and  $x_j^P[k]$  denotes the value of node  $j$  at time  $k$  sent along path  $P$ . The delays are time varying and may be different at each path, but we assume the common upper bound  $\tau$  on any normal path  $P$  (all internal nodes on path  $P$  are normal nodes) as

$$0 \leq \tau_{ij}^P[k] \leq \tau, \quad j \in \mathcal{N}_i^{l-}, \quad k \in \mathbb{Z}_+. \quad (5.13)$$

Hence, each normal node  $i$  becomes aware of the value of each of its normal  $l$ -hop neighbor  $j$  on each normal  $(j, i)$ -path  $P$  at least once in  $\tau$  time steps, but possibly at different time instants [21]. This assumption also indicates that for each normal node,

the gap between two consecutive updates should be less than  $\tau$ , i.e.,  $\bar{k} \leq \tau$ . Although we have this bound on the delay of values of normal nodes, normal nodes need neither the value of this bound nor the information that whether a path  $P$  is a normal path or not.

The structure of asynchronous QMW-MSR algorithm can be outlined as follows. At each time  $k$ , each normal node  $i$  will choose to update or not. If it chooses not to update, i.e.,  $i \notin \mathcal{U}[k]$ , then it will keep its value as  $x_i[k+1] = x_i[k]$ . Otherwise, it will use the most recently received values of all its  $l$ -hop neighbors on each  $l$ -hop path to update its value using the QMW-MSR algorithm. Like the one-hop MSR algorithm, if node  $i$  does not receive any value along some path  $P$  originating from its  $l$ -hop neighbor  $j$  (crash model), then node  $i$  will take this value on path  $P$  as one empty value and will discard this value when it applies QWM-MSR algorithm. As we discussed earlier, in the asynchronous case also, manipulating message paths is equivalent to manipulating message values only.

Let  $D[k]$  be a diagonal matrix whose  $i$ th entry is given by  $d_i[k] = \sum_{j=1}^n a_{ij}[k]$ . Then, let the matrices  $A_\gamma[k] \in \mathbb{R}^{n \times n}$  for  $0 \leq \gamma \leq \tau$ , and  $L_\tau[k] \in \mathbb{R}^{n \times (\tau+1)n}$  be given by

$$A_\gamma[k] = \begin{cases} a_{ij}[k] & \text{if } i \neq j \text{ and } \tau_{ij}[k] = \gamma, \\ 0 & \text{otherwise,} \end{cases} \quad (5.14)$$

and  $L_\tau[k] = [D[k] - A_0[k] - A_1[k] \cdots - A_\tau[k]]$ . Note that the summation of each row of  $L_\tau[k]$  is zero.

Now, the control input can be expressed as

$$\begin{aligned} u^N[k] &= Q(-L_\tau^N[k]z[k]), \\ u^F[k] &: \text{arbitrary,} \end{aligned} \quad (5.15)$$

where  $z[k] = [x[k]^T x[k-1]^T \cdots x[k-\tau]^T]^T$  is a  $(\tau+1)n$ -dimensional vector for  $k \geq 0$  and  $L_\tau^N[k]$  is a matrix formed by the first  $n_N$  rows of  $L_\tau[k]$ . Here,  $z[0]$  is the given initial values of the agents. Then, the agent dynamics can be written as

$$x[k+1] = Q(\Gamma[k]z[k]) + \begin{bmatrix} 0 \\ I_{n_A} \end{bmatrix} u^A[k], \quad (5.16)$$

where  $\Gamma[k]$  is an  $n \times (\tau+1)n$  matrix given by  $\Gamma[k] = [I_n \ 0] - [L_\tau^N[k]^T \ 0]^T$ .

The safety interval differs from the synchronous case and is given by

$$\mathcal{S}_\tau = \left[ \min z^N[0], \max z^N[0] \right]. \quad (5.17)$$

The main result of this section now follows.

**Theorem 5.3.2** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous QMW-MSR algorithm with parameter  $f$  under deterministic updates and time delays in the communication. Under the  $f$ -total/local malicious model, resilient quantized consensus is achieved almost surely only if the underlying graph is  $(f + 1, f + 1)$ -robust with  $l$  hops. Moreover, if the underlying graph is  $(f + 1)$ -strongly robust with  $l$  hops, then resilient quantized consensus is reached almost surely with the safety interval given by (5.17).*

*Proof:* (Necessity) We omit the proof as the synchronous network is a special case of asynchronous network, thus the necessary condition for synchronous case holds for asynchronous case.

(Sufficiency) We show that the conditions (C1)–(C3) in Lemma 5.2.1 hold. To show (C1), define the minimum and maximum values of the normal agents at time  $k$  and the previous  $\tau$  time steps by

$$\begin{aligned} \bar{z}[k] &= \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ \underline{z}[k] &= \min(x^N[k], x^N[k-1], \dots, x^N[k-\tau]). \end{aligned} \quad (5.18)$$

Then we prove that  $\bar{z}[k]$  is a nonincreasing function. By (5.16), at time  $k \geq 1$ , each normal agent updates its value based on a quantized convex combination of the neighbors' values from  $k$  to  $k - \tau$ . We know from step 2 of the QMW-MSR algorithm that the values outside of the interval  $[\underline{z}[k], \bar{z}[k]]$  will be ignored. Hence, we obtain  $x_i[k+1] \leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) = \bar{z}[k]$  for any  $i \in \mathcal{N}$ . It also follows that

$$\begin{aligned} x_i[k] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) \\ x_i[k-1] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) \\ &\vdots \\ x_i[k+1-\tau] &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) \end{aligned}$$

for any  $i \in \mathcal{N}$ . Therefore, we have

$$\begin{aligned}\bar{z}[k+1] &= \max(x^N[k+1], x^N[k], \dots, x^N[k+1-\tau]), \\ &\leq \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]) = \bar{z}[k].\end{aligned}$$

We can similarly prove that  $\underline{z}[k]$  is nondecreasing in time, i.e.,  $\underline{z}[k+1] \geq \underline{z}[k]$ . This indicates that for  $k \geq 1$ , we have  $x_i[k] \in \mathcal{S}_\tau$  for  $i \in \mathcal{N}$ . When  $k = 0$ , the safety condition clearly holds, thus, the safety condition holds for all time  $k$ .

Next, we show (C2). Since  $\bar{z}[k]$  and  $\underline{z}[k]$  are contained in  $\mathcal{S}_\tau$  and are monotone, there is a finite time  $k_c$  such that they both reach their final values with probability 1, denoted by  $\bar{z}^*$  and  $\underline{z}^*$ , respectively. We will prove by contradiction to show that  $\bar{z}^* = \underline{z}^*$ . Assume  $\bar{z}^* > \underline{z}^*$ . When  $k \geq k_c$ , we can define the following sets

$$\begin{aligned}\mathcal{Z}_1[k] &= \{i \in \mathcal{N} : x_i[k] = \bar{z}^*\}, \\ \mathcal{Z}_2[k] &= \{i \in \mathcal{N} : x_i[k] = \underline{z}^*\}.\end{aligned}$$

Due to the convergence of  $\bar{z}[k]$ , at least one normal node is contained in the union of the sets  $\mathcal{Z}_1[k_c + \gamma]$  for  $0 \leq \gamma \leq \tau$ . The same holds for  $\underline{z}[k]$ . We claim that  $\mathcal{Z}_1[k_c]$  is in fact nonempty. To prove this, it is sufficient to show that if  $\mathcal{Z}_1[k_c + \gamma]$  is empty, then the probability of  $\mathcal{Z}_1[k_c + \gamma + 1]$  to be empty is nonzero for  $\gamma = 0, \dots, \tau$ .

First, we show that none of the normal agents in  $\mathcal{V} \setminus \mathcal{Z}_1[k]$  enters  $\mathcal{Z}_1[k+1]$  at the next time step with positive probability. If there is no node updating at time  $k$ , then no node can enter  $\mathcal{Z}_1[k+1]$ . However, we know that each normal node makes an update at least once in  $\tau$  time steps. Assume that the normal node  $i$  makes an update at time  $k_c + \gamma$ . Since  $\mathcal{Z}_1[k_c + \gamma]$  is empty by assumption, node  $i$  is upper bounded by  $\bar{z}^* - 1$ . Then we have

$$x_i[k_c + \gamma + 1] \leq Q((1 - \alpha)\bar{z}^* + \alpha(\bar{z}^* - 1)) = Q(\bar{z}^* - \alpha), \quad (5.19)$$

where the quantizer takes  $Q(\bar{x}^* - \alpha) = \bar{x}^* - 1$  with probability  $\alpha$ . We can also prove that  $\mathcal{Z}_2[k_c]$  is nonempty by the same analysis.

Then we show that with positive probability, the nodes in  $\mathcal{Z}_1[k]$  decrease their values, and the nodes in  $\mathcal{Z}_2[k]$  increase their values at the next time step. Since  $\mathcal{G}$  is  $(f+1)$ -strongly robust with  $l$  hops, for nonempty and disjoint sets  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$ , at least one of these two sets includes a node having at least  $f+1$  independent paths

originating from the normal nodes outside the corresponding set and these paths do not contain any intermediate nodes in set  $\mathcal{A}$ . Without loss of generality, assume node  $i \in \mathcal{Z}_1[k]$ , then  $x_i[k] = \bar{z}^*$ . Due to the  $f$ -total/local model of adversary nodes, node  $i$  will discard all the values larger than  $\bar{z}^*$  by step 2 of the QMW-MSR algorithm, and will use at least one value from the normal nodes outside  $\mathcal{Z}_1[k]$ , which is smaller than  $\bar{z}^*$ . Thus,

$$x_i[k+1] \leq Q((1-\alpha)\bar{z}^* + \alpha(\bar{z}^* - 1)) = Q(\bar{z}^* - \alpha), \quad (5.20)$$

where the quantizer takes  $Q(\bar{x}^* - \alpha) = \bar{x}^* - 1$  with probability  $\alpha$ . Therefore, with positive probability, one of the normal nodes in  $\mathcal{Z}_1[k]$  decreases its values by at least one. Similarly, we can prove that with positive probability, one of the normal nodes in  $\mathcal{Z}_2[k]$  increases its values by at least one.

Then we conclude that at each time step, some nodes in  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$  will go out from the corresponding set. Hence, for any  $k \geq k_c + \bar{k} \cdot n_N$ , the number of normal agents in one of the sets  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$  is zero with positive probability because the number of normal nodes is  $n_N$  and each normal node will update at least once in  $\bar{k}$  steps. This is a contradiction and proves (C2).

Lastly, we show (C3). Once the normal nodes have reached the common value  $x^*$  at time  $k$ . Since there are at most  $f$  adversary nodes in the  $l$ -hop neighbors of any normal node, according to the QMW-MSR algorithm, all malicious nodes  $j$  with values  $x_j[k] \neq x^*$  are ignored by the normal nodes. Thus,  $x_i[k+1] = x^*, \forall i \in \mathcal{N}$ . Complete the proof.  $\blacksquare$

In [22], a sufficient condition for resilient quantized consensus under asynchronous deterministic updates is provided as the graph  $\mathcal{G}$  is  $(2f+1)$ -robust (with one-hop). As mentioned in Proposition 4.3.2, if graph  $\mathcal{G}$  is  $(2f+1)$ -robust, then it is  $(f+1)$ -strongly robust (with one-hop), not vice versa. Hence, our sufficient condition is tighter than that in [22]. However, we note that checking strong robustness of a given graph is more complex compared to checking robustness because we need to check  $\binom{n}{f}$  times for robustness of  $\mathcal{G}_{\mathcal{H}}$ .

Recall that in Section 5.3.1, using randomized updates, we are able to derive a tighter sufficient condition for resilient consensus of asynchronous networks without delay, which is that  $\mathcal{G}$  is  $(f+1, f+1)$ -robust with  $l$  hops. However, when time delays are in presence in the communication among nodes, this condition is not sufficient anymore even if we apply randomized updates. Simply speaking, a reason for this



phenomenon is that if the nonuniform delays are in presence in different paths, then malicious nodes can update frequently, and their delayed values received by the normal neighbors may appear different for different neighbors. In this case, malicious nodes can have attack ability close to Byzantine nodes, and hence a stricter condition is needed for guaranteeing resilient consensus. The proof for the following proposition is similar to that for Theorem 5.3.2.

**Proposition 5.3.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous QMW-MSR algorithm with parameter  $f$  under randomized updates and time delays in the communication. Under the  $f$ -total/local malicious model, resilient asymptotic consensus is achieved almost surely only if the underlying graph is  $(f + 1, f + 1)$ -robust with  $l$  hops. Moreover, if the underlying graph is  $(f + 1)$ -strongly robust with  $l$  hops, then resilient consensus is reached almost surely with the safety interval given by (5.17).*

For Byzantine attacks, the necessary condition needs to be even stricter than for malicious attacks.

**Proposition 5.3.2** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous QMW-MSR algorithm with parameter  $f$  under randomized/deterministic updates and time delays in the communication. Under the  $f$ -total/local Byzantine model, resilient asymptotic consensus is achieved almost surely with the safety interval given by (5.17) if and only if the underlying graph is  $(f + 1)$ -strongly robust with  $l$  hops.*

*Proof:* The proof of the necessity part is similar to the real-valued consensus in our previous work and we omit it here.

For the sufficiency part, we need to show the conditions (C1)–(C3) in Lemma 5.2.1 hold. Note that in the proof of Theorem 5.3.2, conditions (C1) and (C3) still hold for Byzantine attacks. For condition (C2), note that the sets  $\mathcal{Z}_1[k]$  and  $\mathcal{Z}_2[k]$  are defined as the subsets of the set of normal nodes  $\mathcal{N}$ , and node  $i$  in one of these two sets will use at least one value from the normal nodes outside its corresponding set. Both situations are applied for normal nodes, thus, all the analysis in the proof of Theorem 5.3.2 still holds even under Byzantine attacks. ■

## 5.4 Summary

We have studied the problem of resilient quantized consensus with asynchronous updates and time delays in the communication between agents. We have proved necessary and sufficient conditions for our algorithm to guarantee resilient quantized consensus for synchronous/asynchronous updates under malicious/Byzantine attacks. Compared to the existing methods, our algorithm has a tighter graph condition and we provide a unified analysis for different network settings.

## Chapter 6

# Event-triggered Approximate Byzantine Consensus with Multi-hop Communication

In this chapter, we consider a resilient consensus problem where some agents are under Byzantine attacks. In particular, we develop an event-triggered update rule to tackle this problem as well as reduce the communication for each agent.

As we discussed in Chapters 3 and 4, through multi-hop communication, the connectivity requirement for resilient consensus becomes less stringent than the conditions for the one-hop case. This is enabled by increasing the amount of data exchanged among agents through message relaying. In this chapter, we aim to reduce the transmissions for the agents using the MW-MSR algorithm through event-triggered protocols [45]. In the literature [23; 50; 68; 114], event-triggered schemes have shown their effectiveness in reducing the transmissions for the agents using distributed algorithms even under adversarial environments.

Specifically, agents using the event-triggered MW-MSR algorithm will update locally, and they send their own state values along with relayed values only when the difference between the current value and the past communicated value exceeds a given threshold. Through simulations, we can see that the agents' transmissions can be significantly reduced compared to the multi-hop algorithm without the event-triggered protocol in Chapter 4. Furthermore, compared to the one-hop MSR algorithm with or without event-triggered protocols [114], [56], the connectivity requirement for our

algorithm is less stringent. Besides, we analyze the performance of our algorithm with delays in communication, which is a case not studied in [114].

The rest of this chapter is organized as follows. Section 6.1 outlines the system model. Section 6.2 presents the event-triggered MW-MSR algorithm. In Section 6.3, we derive a condition under which the proposed algorithm reaches resilient consensus under asynchronous updates with delays. In Section 6.4, we provide an alternative event-triggered scheme and its analysis. Section 6.5 provides numerical examples to show the effectiveness of the proposed algorithm. Lastly, Section 6.6 concludes this chapter.

## 6.1 Problem Formulation

### 6.1.1 Update Rule

In the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , the node set  $\mathcal{V}$  is partitioned into the set of normal nodes  $\mathcal{N}$  and the set of adversary nodes  $\mathcal{A}$ , where  $|\mathcal{N}| = N$ . The partition is unknown to the normal nodes at all times.

The update rule for normal agent  $i$  is described by

$$x_i[k+1] = x_i[k] + u_i[k], \quad (6.1)$$

where  $x_i[k] \in \mathbb{R}$  is the state and  $u_i[k]$  is the control input given by

$$u_i[k] = \sum_{j \in \mathcal{N}_i^+[k]} a_{ij}[k] (\hat{x}_j[k] - x_i[k]). \quad (6.2)$$

Here,  $\hat{x}_j[k] \in \mathbb{R}$  is an auxiliary state, representing the last communicated state of node  $j$  at time  $k$ . It is defined as

$$\hat{x}_j[k] = x_j[t_h^j], k \in [t_h^j, t_{h+1}^j), \quad (6.3)$$

where  $t_0^j, t_1^j, \dots$  denote the transmission times of node  $j$  determined by the triggering function to be given below. The initial values  $x_i[0], x_j[0]$  are given, and  $a_{ij}[k]$  is the weight for the edge  $(j, i)$ . Note that at initial time,  $\hat{x}_i[0]$  need not be the same as  $x_i[0]$ . Let  $a_{ii}[k] = 1 - \sum_{j \in \mathcal{N}_i^+[k]} a_{ij}[k]$ . Assume that  $\gamma \leq a_{ij}[k] < 1$  if  $a_{ij}[k] \neq 0$  or  $i = j$  for  $i, j \in \mathcal{V}$ , where  $0 < \gamma < 1$ . In the resilient consensus algorithm to be introduced, the

neighbors whose values are used for updates change over time and, hence, the weights  $a_{ij}[k]$  are time varying.

We now introduce the triggering function. Denote the error at time  $k$  between the updated state  $x_i[k+1]$  and the auxiliary state  $\hat{x}_i(k)$  by  $e_i[k] = \hat{x}_i[k] - x_i[k+1]$  for  $k \geq 0$ . Then, let

$$f_i[k] = |e_i[k]| - (c_0 + c_1[k]), \quad (6.4)$$

where  $c_0 \geq 0$  is a constant and  $c_1[k]$  takes nonnegative and decreasing values with  $c_1[k] \rightarrow 0$  in finite time. The roles of  $c_0$  and  $c_1[k]$  are to reduce the triggering frequency, and especially  $c_1[k]$  allows the threshold to be large in the initial phase. Each node  $i$  will check this function and whenever it finds  $f_i[k]$  to be positive, it will transmit its new state  $x_i[k+1]$  to its neighbors.

We employ the control input taking account of possible delays in the transmission. Thus, we extend (6.2) as

$$u_i[k] = \sum_{j \in \mathcal{N}_i^{l^-}} a_{ij}[k] (\hat{x}_j^P[k - \tau_{ij}^P[k]] - x_i[k]), \quad (6.5)$$

where  $\hat{x}_j^P[k]$  denotes the value of node  $j$  at time  $k$  sent along path  $P$  and  $\tau_{ij}^P[k] \in \mathbb{Z}_+$  denotes the delay in this  $(j, i)$ -path  $P$  at time  $k$ . The delays are time varying and may be different in each path. We assume the common upper bound  $\tau$  on any *normal* path  $P$ , over which all internal nodes are normal, as

$$0 \leq \tau_{ij}^P[k] \leq \tau, \quad j \in \mathcal{N}_i^{l^-}, \quad k \in \mathbb{Z}_+. \quad (6.6)$$

In the following part, we also assume that every normal node  $i$  updates its value at least once in every  $\theta \geq 1$  steps. When  $\theta = 1$ , updates are synchronous. Although we impose this bound on the delays for message transmissions, the normal nodes need neither the value of this bound nor the information whether a path  $P$  is a normal one or not. Also, there is no constraint on the size of  $\tau$ .

Under the delay bound  $\tau$  imposed in (6.5), triggered values of each node must reach all the multi-hop neighbors in  $\tau$  steps. We have two possible relay models that can be employed in the proposed multi-hop algorithm:

(i) *Periodic relay model*: Each node relays all the recently received messages to its one-hop neighbors every  $\lambda$  steps. If  $\lambda = 1$ , each node must immediately relay the

received messages. This is referred to as the *immediate relay model*.

(ii) *Package relay model*: Each node relays all the recently received messages along with its own values (e.g., in a message package) to its one-hop neighbors when its own event is triggered.

Among the two modes, clearly, the package relay model requires less frequent message transmissions and may be a more natural model in the event-based algorithm studied here. We note however that with this model, it must be assumed that at time  $k = 0$ , the neighboring agents exchange their state values. This is to cope with the situation where no event is triggered by any of the agents. This can occur since the event triggering function only takes account of the local states. We will illustrate the difference of the effects of the two relay models through simulations later.

### 6.1.2 Threat Model

In this chapter, we study the resilient consensus under  $f$ -total/local Byzantine attacks. The definitions of the adversary model is given in Section 2.2.2. As commonly done in the literature, we assume that each normal node knows the value of  $f$  and the topology information of the graph up to  $l$  hops. We assume that each adversary node  $i$  cannot manipulate the path values in the messages containing its own state  $x_i[k]$  and those that it relays as stated in Assumption 2.3.1.

### 6.1.3 Resilient Asymptotic Consensus

We now introduce the type of consensus among the normal agents to be sought in this chapter [114]. Note that in event-triggered schemes, the consensus condition is different from the one in Chapters 3 and 4.

**Definition 6.1.1** *Given  $c \geq 0$ , if for any possible sets and behaviors of the adversary agents and any state values of the normal nodes, the following two conditions are satisfied, then we say that the normal agents reach resilient consensus at the error level  $c$ :*

1. *Safety: There exists a bounded safety interval  $\mathcal{S}$  determined by the initial values of the normal agents such that  $x_i[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .*
2. *Agreement: For all  $i, j \in \mathcal{N}$ , it holds that  $\limsup_{k \rightarrow \infty} |x_i[k] - x_j[k]| \leq c$ .*

## 6.2 Event-triggered Algorithm Design

In this section, we outline the structure of the event-triggered multi-hop weighted MSR (EMW-MSR) algorithm.

### 6.2.1 Asynchronous Event-triggered MW-MSR algorithm

At each time  $k$ , each normal node  $i$  updates as follows:

1. *Receive step*: Node  $i$  receives neighbors' values through different paths (described in (6.5)) and chooses to update its state or not. If it chooses to update, then it proceeds to step 2. Otherwise, it keeps its value as  $x_i[k+1] = x_i[k]$ .

2. *Update step*: Node  $i$  updates its value  $x_i[k+1]$  according to Algorithm 3 using the values most recently received from neighbors and its own value  $x_i[k]$ .

3. *Transmit step*: Node  $i$  checks the value of  $f_i[k]$  and sets the value of  $\hat{x}_i[k+1]$  as

$$\hat{x}_i[k+1] = \begin{cases} x_i[k+1], & \text{if } f_i[k] > 0, \\ \hat{x}_i[k], & \text{otherwise.} \end{cases} \quad (6.7)$$

Here, the auxiliary variable will be updated only when the current value has varied enough to exceed a threshold, and only at this time the node sends its value and the relayed values over each  $l$ -hop path to node  $j \in \mathcal{N}_i^{l+}$ .

In the Transmit step and Receive step, the nodes exchange messages with others that are up to  $l$  hops away. Then in the Update step, node  $i$  updates its state using Algorithm 3. Note that the adversary nodes may deviate from this specification as we describe in the next subsection.

One important feature here to further reduce the amount of data in each transmission when an event is triggered is to require that the nodes can send only the relayed values that have changed since last event.

## 6.3 Consensus Analysis

In this section, we first prove the convergence of the asynchronous event-triggered MW-MSR algorithm. Then we discuss the effects of different relay models on the performance of the proposed algorithm.

To prove the convergence, we introduce two kinds of minimum and maximum of the

---

**Algorithm 3:** EMW-MSR Algorithm (Update step)
 

---

- 1) At time  $k$ , normal node  $i$  obtains the most recently received event-triggered values of the nodes in  $\mathcal{N}_i^{l-}$  and its state  $x_i[k]$ , whose set is denoted by  $\mathcal{M}_i[k]$ , and sorts the values in  $\mathcal{M}_i[k]$  in an increasing order.
- 2) (a) Define two subsets of  $\mathcal{M}_i[k]$  based on the message values:

$$\overline{\mathcal{M}}_i[k] = \{m \in \mathcal{M}_i[k] : \text{value}(m) > x_i[k]\},$$

$$\underline{\mathcal{M}}_i[k] = \{m \in \mathcal{M}_i[k] : \text{value}(m) < x_i[k]\}.$$

- (b) Then, let  $\overline{\mathcal{R}}_i[k] = \overline{\mathcal{M}}_i[k]$  if the cardinality of a minimum cover of  $\overline{\mathcal{M}}_i[k]$  is less than  $f$ , i.e.,  $|\mathcal{T}^*(\overline{\mathcal{M}}_i[k])| < f$ . Otherwise, let  $\overline{\mathcal{R}}_i[k]$  be the largest sized subset of  $\overline{\mathcal{M}}_i[k]$  such that (i) for all  $m \in \overline{\mathcal{M}}_i[k] \setminus \overline{\mathcal{R}}_i[k]$  and  $m' \in \overline{\mathcal{R}}_i[k]$  we have  $\text{value}(m) \leq \text{value}(m')$ , and (ii) the cardinality of a minimum cover of  $\overline{\mathcal{R}}_i[k]$  is exactly  $f$ , i.e.,  $|\mathcal{T}^*(\overline{\mathcal{R}}_i[k])| = f$ .

- (c) Similarly, we can get  $\underline{\mathcal{R}}_i[k]$  from  $\underline{\mathcal{M}}_i[k]$ , which contains the smallest values.

- (d) Finally, let  $\mathcal{R}_i[k] = \overline{\mathcal{R}}_i[k] \cup \underline{\mathcal{R}}_i[k]$ .

- 3) Node  $i$  updates its value as follows:

$$x_i[k+1] = \sum_{m \in \mathcal{D}_i[k]} a_i[k] \text{value}(m), \quad (6.8)$$

where  $a_i[k] = 1/|\mathcal{D}_i[k]|$  and  $\mathcal{D}_i[k] = \mathcal{M}_i[k] \setminus \mathcal{R}_i[k]$ .

---

states of the normal agents. Denote the state vector and the transmitted state vector of normal agents at time  $k$  by  $x^N[k]$  and  $\hat{x}^N[k]$ , respectively.

First, we denote the minimum and maximum of the states of the normal agents from time  $k - \tau$  to time  $k$  as

$$\begin{aligned} \overline{x}_\tau[k] &= \max(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \\ \underline{x}_\tau[k] &= \min(x^N[k], x^N[k-1], \dots, x^N[k-\tau]), \end{aligned} \quad (6.9)$$

respectively. Next, we denote the joint minimum and maximum of the states and the transmitted states of the normal agents from time  $k - \tau$  to time  $k$ , respectively, as

$$\begin{aligned} \overline{\hat{x}}_\tau[k] &= \max(x^N[k], \dots, x^N[k-\tau], \hat{x}^N[k], \dots, \hat{x}^N[k-\tau]), \\ \underline{\hat{x}}_\tau[k] &= \min(x^N[k], \dots, x^N[k-\tau], \hat{x}^N[k], \dots, \hat{x}^N[k-\tau]). \end{aligned} \quad (6.10)$$

We are ready to state the main theorem of the chapter.



**Theorem 6.3.1** Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous event-triggered MW-MSR algorithm. Under the  $f$ -local Byzantine model, the normal nodes reach resilient consensus at an error level  $c$  if and only if the underlying graph is  $(f+1)$ -strongly robust with  $l$  hops. Moreover, the safety interval is given by  $\mathcal{S} = [\underline{\hat{x}}_\tau[0], \bar{\hat{x}}_\tau[0]]$ , and the consensus error level  $c$  is achieved if the parameter  $c_0$  in the triggering function (??) satisfies

$$c_0 \leq \frac{\gamma^{N\theta}}{4N\theta} c. \quad (6.11)$$

*Proof: (Necessity)* This part follows from our previous work in Chapter 4, which considers the special case without the triggering function, that is,  $c_0 = c_1[k] = 0$ .

*(Sufficiency)* First, we show by induction that the safety condition is satisfied. Note that the update rule (6.8) in Algorithm 1 can be rewritten as

$$x_i[k+1] = a_i[k]x_i[k] + \sum_{j \in \mathcal{D}_i[k]} a_i[k]\hat{x}_j^P[k - \tau_{ij}^P[k]], \quad (6.12)$$

where  $a_i[k] = 1/|\mathcal{D}_i[k]|$ . At time  $k = 0$ , it is clear by definition that  $x_i[0], \hat{x}_i[0] \in \mathcal{S}$ . We first show that  $\bar{\hat{x}}_\tau[k]$  is nonincreasing in time. From (6.12), we have  $x_i[k+1] \leq \bar{\hat{x}}_\tau[k]$  for all  $i \in \mathcal{N}$  since the values larger than  $\bar{\hat{x}}_\tau[k]$  are ignored in step 2 of Algorithm 1. Moreover, by (6.7), it follows that  $\hat{x}_i[k+1] \leq \bar{\hat{x}}_\tau[k]$  for all  $i \in \mathcal{N}$ . Together, we have  $\bar{\hat{x}}_\tau[k+1] \leq \bar{\hat{x}}_\tau[k]$ . We can similarly prove that  $\underline{\hat{x}}_\tau[k]$  is nondecreasing in time.

We next show the consensus part. Note that for time  $k \in (t_h^j, t_{h+1}^j)$  between two triggering instants, we have  $f_i[k] \leq 0$ . Moreover, for the neighbor node  $j \in \mathcal{N}_i^{j-}$ , if  $f_j[k] > 0$ , then we have  $\hat{x}_j[k+1] = x_j[k+1]$ . If  $f_j[k] \leq 0$ , then  $\hat{x}_j[k+1] = \hat{x}_j[k] = x_j[k+1] + e_j[k]$ . As a result, it holds  $\hat{x}_j[k] = x_j[k] + \hat{e}_j[k-1]$  for  $k \geq 1$ , where

$$\hat{e}_j[k] = \begin{cases} e_j[k], & \text{if } f_i[k] \leq 0, \\ 0, & \text{otherwise.} \end{cases} \quad (6.13)$$

Note that

$$|e_j[k]| \leq c_0 + c_1[k], \quad \forall k \geq 0. \quad (6.14)$$

Then, we can write (6.12) as

$$x_i[k+1] = a_i[k]x_i[k] + \sum_{j \in \mathcal{M}_i[k] \setminus \mathcal{R}_i[k]} a_i[k](x_j^P[k - \tau_{ij}^P[k]] + \hat{e}_j[k - \tau_{ij}^P[k] - 1]). \quad (6.15)$$

This can be bounded as

$$\begin{aligned} x_i[k+1] &\leq a_i[k]\bar{x}_\tau[k] + \sum_{j \in \mathcal{M}_i[k] \setminus \mathcal{R}_i[k]} a_i[k](\bar{x}_\tau[k] + \hat{e}_j[k - \tau_{ij}^P[k] - 1]) \\ &\leq \bar{x}_\tau[k] + \max_{j \in \mathcal{M}_i[k] \setminus \mathcal{R}_i[k]} |\hat{e}_j[k - \tau_{ij}^P[k] - 1]|. \end{aligned} \quad (6.16)$$

Thus, by (6.14), letting  $c_1[k] = c_1[0]$  for  $k < 0$ , we have

$$x_i[k+1] \leq \bar{x}_\tau[k] + c_0 + c_1[k - \tau - 1]. \quad (6.17)$$

Similarly, we have

$$x_i[k+1] \geq \underline{x}_\tau[k] - c_0 - c_1[k - \tau - 1]. \quad (6.18)$$

Let  $V[k] = \bar{x}_\tau[k] - \underline{x}_\tau[k]$ . Then, define two sequences by

$$\begin{aligned} \bar{x}_0[k+1] &= \bar{x}_0[k] + c_0 + c_1[k - \tau - 1], \\ \underline{x}_0[k+1] &= \underline{x}_0[k] - c_0 - c_1[k - \tau - 1], \end{aligned} \quad (6.19)$$

where  $\bar{x}_0[0] = \bar{x}_\tau[0] - \sigma_0$ , and  $\underline{x}_0[0] = \underline{x}_\tau[0] + \sigma_0$  with  $\sigma_0 = \sigma V[0]$ . Then the following inequalities hold:

$$\begin{aligned} \bar{x}_\tau[k] &\leq \bar{x}_0[k] + \sigma_0, \\ \underline{x}_\tau[k] &\geq \underline{x}_0[k] - \sigma_0. \end{aligned} \quad (6.20)$$

We show  $\bar{x}_\tau[k] \leq \bar{x}_0[k] + \sigma_0$  by induction, and  $\underline{x}_\tau[k] \geq \underline{x}_0[k] - \sigma_0$  can be proved in a similar way. When  $k = 0$ , we clearly have  $\bar{x}_\tau[0] = \bar{x}_0[0] + \sigma_0$ . Suppose that (6.20) holds. Then, we have at time  $k + 1$

$$\begin{aligned} \bar{x}_\tau[k+1] &= \max(x^N[k+1], x^N[k], \dots, x^N[k+1-\tau]) \\ &\leq \bar{x}_\tau[k] + c_0 + c_1[k - \tau - 1] \\ &\leq (\bar{x}_0[k] + \sigma_0) + c_0 + c_1[k - \tau - 1] \\ &= \bar{x}_0[k+1] + \sigma_0. \end{aligned} \quad (6.21)$$

The first inequality holds because from (6.17), we have  $\max x^N[k+1] \leq \bar{x}_\tau[k] + c_0 + c_1[k - \tau - 1]$ . Moreover, from (6.9), we have  $\max(x^N[k], \dots, x^N[k+1 - \tau]) \leq \bar{x}_\tau[k]$ .

We next introduce another sequence  $\varepsilon_0[k]$  defined by

$$\varepsilon_0[k+1] = \gamma\varepsilon_0[k] - (1 - \gamma)\sigma_0, \quad (6.22)$$

where  $\varepsilon_0[0] = \varepsilon V[0]$ . Take the positive  $\varepsilon$  and  $\sigma$  so that

$$\varepsilon + \sigma = \frac{1}{2}, \quad 0 < \sigma < \frac{\gamma^{N\theta}}{1 - \gamma^{N\theta}}\varepsilon. \quad (6.23)$$

Here, we claim that it holds

$$0 < \varepsilon_0[k+1] < \varepsilon_0[k], \quad k = 0, 1, \dots, N\theta - 1. \quad (6.24)$$

This is proved as follows. Since  $0 < \gamma < 1$ , from (6.22), we can easily have  $\varepsilon_0[k+1] < \varepsilon_0[k]$ . It is thus sufficient to show  $\varepsilon_0[N\theta] > 0$ . From (6.22), we have

$$\begin{aligned} \varepsilon_0[N\theta] &= \gamma^{N\theta}\varepsilon_0[0] - \sum_{j=0}^{N\theta-1} \gamma^j(1 - \gamma)\sigma_0 \\ &= \left(\gamma^{N\theta}\varepsilon - (1 - \gamma^{N\theta})\sigma\right) V[0]. \end{aligned}$$

This is positive because we have chosen  $\varepsilon$  and  $\sigma$  as in (6.23).

For the sequence  $\varepsilon_0[k]$ , define two sets as

$$\begin{aligned} \mathcal{Z}_1(k, \varepsilon_0[k]) &= \{i \in \mathcal{N} : x_i[k] > \bar{x}_0[k] - \varepsilon_0[k]\}, \\ \mathcal{Z}_2(k, \varepsilon_0[k]) &= \{i \in \mathcal{N} : x_i[k] < \underline{x}_0[k] + \varepsilon_0[k]\}. \end{aligned}$$

These sets are both nonempty at time  $k = 0$  and, in particular, each contains at least one normal node; this is because, by definition,  $\bar{x}_\tau[0] > \bar{x}_0[0] - \varepsilon_0[0]$  and  $\bar{x}_\tau[0] < \underline{x}_0[0] + \varepsilon_0[0]$ .

In the following, we show that  $\mathcal{Z}_1(k, \varepsilon_0[k])$  and  $\mathcal{Z}_2(k, \varepsilon_0[k])$  are disjoint sets. To this end, we must show

$$\bar{x}_0[k] - \varepsilon_0[k] \geq \underline{x}_0[0] + \varepsilon_0[0]. \quad (6.25)$$

By (6.19) for  $\bar{x}_0[k]$  and  $\underline{x}_0[k]$ , we have

$$\begin{aligned} & (\bar{x}_0[k] - \varepsilon_0[k]) - (\underline{x}_0[k] + \varepsilon_0[k]) \\ &= \left( \bar{x}_0[0] + c_0 k + \sum_{j=-\tau-1}^{k-\tau-2} c_1[j] \right) - \left( \underline{x}_0[0] - c_0 k - \sum_{j=-\tau-1}^{k-\tau-2} c_1[j] \right) - 2\varepsilon_0[k]. \end{aligned}$$

Since  $\bar{x}_0[0] = \bar{x}_\tau[0] - \sigma_0$  and  $\underline{x}_0[0] = \underline{x}_\tau[0] + \sigma_0$  with  $\sigma_0 = \sigma V[0]$ , we have

$$\begin{aligned} & (\bar{x}_0[k] - \varepsilon_0[k]) - (\underline{x}_0[k] + \varepsilon_0[k]) \\ &= (\bar{x}_\tau[0] - \underline{x}_\tau[0]) - 2\sigma_0 + 2c_0 k + 2 \sum_{j=-\tau-1}^{k-\tau-2} c_1[j] - 2\varepsilon_0[k] \\ &= V[0] - 2\sigma V[0] + 2c_0 k + 2 \sum_{j=-\tau-1}^{k-\tau-2} c_1[j] - 2\varepsilon_0[k] \\ &> (1 - 2\sigma - 2\varepsilon)V[0] + 2c_0 k + 2 \sum_{j=-\tau-1}^{k-\tau-2} c_1[j] \geq 0. \end{aligned}$$

The last inequality holds since  $\varepsilon + \sigma = 1/2$  and  $\varepsilon_0[k] < \varepsilon_0[0] = \varepsilon V[0]$ . Thus, we have proved (6.25).

So far, we have shown that the two sets  $\mathcal{Z}_1(k, \varepsilon_0[k])$  and  $\mathcal{Z}_2(k, \varepsilon_0[k])$  are disjoint. Notice that the network is  $(f+1)$ -strongly robust with  $l$  hops w.r.t. any set  $\mathcal{F}$  following the  $f$ -local model and the set of Byzantine nodes  $\mathcal{A}$  also satisfies the  $f$ -local model. Hence, the network is  $(f+1)$ -strongly robust with  $l$  hops w.r.t. the set  $\mathcal{A}$  and at least one of the conditions in Definition 3.3.2 for robustness holds. Therefore, if the two sets are both nonempty, then for these two nonempty disjoint sets  $\mathcal{Z}_1(k, \varepsilon_0[k])$  and  $\mathcal{Z}_2(k, \varepsilon_0[k])$ , one of them has a normal agent with at least  $f+1$  independent normal paths originating from some normal nodes outside.

Suppose that normal node  $i \in \mathcal{Z}_1(k, \varepsilon_0[k])$  has the above-mentioned property. A similar argument holds when  $i \in \mathcal{Z}_2(k, \varepsilon_0[k])$ . Now, we go back to the update rule (6.15) for node  $i$  and rewrite it by partitioning the neighbor set into two parts: those that belong to  $\mathcal{Z}_1(k, \varepsilon_0[k])$  and those that do not. Node  $i$  has at least  $f+1$  independent normal paths originating from the normal nodes outside. According to Algorithm 3, it will use at least one value originating from the normal nodes outside  $\mathcal{Z}_1(k, \varepsilon_0[k])$ ; thus,

we obtain

$$\begin{aligned}
 x_i[k+1] &= a_i[k]x_i[k] + \sum_{j \in \mathcal{D}_i[k] \cap \mathcal{Z}_1} a_i[k]x_j^P[k - \tau_{ij}^P[k]] \\
 &\quad + \sum_{j \in \mathcal{D}_i[k] \setminus \mathcal{Z}_1} a_i[k]x_j^P[k - \tau_{ij}^P[k]] + \sum_{j \in \mathcal{D}_i[k]} a_i[k]\hat{e}_j[k - \tau_{ij}^P[k] - 1] \\
 &\leq a_i[k]\bar{x}_\tau[k] + \sum_{j \in \mathcal{D}_i[k] \cap \mathcal{Z}_1} a_i[k]\bar{x}_\tau[k] \\
 &\quad + \sum_{j \in \mathcal{D}_i[k] \setminus \mathcal{Z}_1} a_i[k](\bar{x}_0[k] - \varepsilon_0[k]) + \sum_{j \in \mathcal{D}_i[k]} a_i[k]\hat{e}_j[k - \tau_{ij}^P[k] - 1].
 \end{aligned}$$

Combining (6.20) and the fact that  $a_i[k]$  is lower bounded by  $\gamma$ , we have

$$\begin{aligned}
 x_i[k+1] &\leq (1 - \gamma)\bar{x}_\tau[k] + \gamma(\bar{x}_0[k] - \varepsilon_0[k]) + c_0 + c_1[k - \tau - 1] \\
 &\leq (1 - \gamma)(\bar{x}_0[k] + \sigma_0) + \gamma(\bar{x}_0[k] - \varepsilon_0[k]) + c_0 + c_1[k - \tau - 1] \\
 &\leq \bar{x}_0[k] + c_0 + c_1[k - \tau - 1] + (1 - \gamma)\sigma_0 - \gamma\varepsilon_0[k] \\
 &= \bar{x}_0[k+1] - \varepsilon_0[k+1]
 \end{aligned} \tag{6.26}$$

for  $k = 0, 1, \dots, N\theta - 1$ , where the first inequality follows from the assumption that  $\mathcal{Z}_1(k, \varepsilon_0[k])$  is nonempty, and the equality follows from (6.19) and (6.22). The relation in (6.26) shows that once an update happens at node  $i$ , then this node will move out of  $\mathcal{Z}_1(k+1, \varepsilon_0[k+1])$ . It is further noted that inequality (6.26) also holds for the normal nodes that are not in  $\mathcal{Z}_1(k, \varepsilon_0[k])$  at time  $k$ . This indicates that the nodes outside  $\mathcal{Z}_1(k, \varepsilon_0[k])$  will not move in  $\mathcal{Z}_1(k+1, \varepsilon_0[k+1])$ . Similar results hold for the set  $\mathcal{Z}_2(k+1, \varepsilon_0[k+1])$ .

Recall that the normal nodes update at least once for every  $\theta$  steps. As a result, if the two sets  $\mathcal{Z}_1(k, \varepsilon_0[k])$  and  $\mathcal{Z}_2(k, \varepsilon_0[k])$  are both nonempty at time  $k$ , then after  $N\theta$  time steps, all the normal nodes will be out of at least one of them. Suppose that  $\mathcal{Z}_1(k, \varepsilon_0[k])$  is empty. When such an event occurs at  $k = 0$ , it clearly follows that

$\bar{x}_\tau[N\theta] \leq \bar{x}_0[N\theta] - \varepsilon_0[N\theta]$ . From the definition of  $V[k]$ , we have

$$\begin{aligned}
 V[N\theta] &= \bar{x}_\tau[N\theta] - \underline{x}_\tau[N\theta] \\
 &\leq (\bar{x}_0[N\theta] - \varepsilon_0[N\theta]) - (\underline{x}_0[N\theta] - \sigma_0) \\
 &= \bar{x}_0[0] - \underline{x}_0[0] + 2c_0N\theta + 2 \sum_{j=-\tau-1}^{N\theta-\tau-2} c_1[j] - \varepsilon_0[N\theta] + \sigma_0 \\
 &= (\bar{x}_\tau[0] - \sigma_0) - (\underline{x}_\tau[0] + \sigma_0) + 2c_0N\theta + 2 \sum_{j=-\tau-1}^{N\theta-\tau-2} c_1[j] \\
 &\quad - \varepsilon_0[N\theta] + \sigma_0 \\
 &= V[0] - \sigma V[0] + 2c_0N\theta + 2 \sum_{j=-\tau-1}^{N\theta-\tau-2} c_1[j] \\
 &\quad - \left( \gamma^{N\theta} \varepsilon - (1 - \gamma^{N\theta}) \sigma \right) V[0] \\
 &= \left( 1 - \gamma^{N\theta} (\varepsilon + \sigma) \right) V[0] + 2c_0N\theta + 2 \sum_{j=-\tau-1}^{N\theta-\tau-2} c_1[j].
 \end{aligned}$$

By (6.23), we have

$$V[N\theta] \leq \left( 1 - \frac{\gamma^{N\theta}}{2} \right) V[0] + 2c_0N\theta + 2 \sum_{j=-\tau-1}^{N\theta-\tau-2} c_1[j]. \quad (6.27)$$

If there are more updates by node  $i$  after time  $k = N\theta$ , this argument can be extended further as

$$V[hN\theta] \leq \left( 1 - \frac{\gamma^{N\theta}}{2} \right) V[(h-1)N\theta] + 2c_0N\theta + 2 \sum_{j=(h-1)N\theta-\tau-1}^{hN\theta-\tau-2} c_1[j]. \quad (6.28)$$

Hence, we have

$$\begin{aligned}
V[hN\theta] &\leq \left(1 - \frac{\gamma^{N\theta}}{2}\right)^h V[0] + \sum_{t=0}^{h-1} \left(1 - \frac{\gamma^{N\theta}}{2}\right)^{h-1-t} \left(2c_0N\theta + 2 \sum_{j=tN\theta-\tau-1}^{(t+1)N\theta-\tau-2} c_1[j]\right) \\
&\leq \left(1 - \frac{\gamma^{N\theta}}{2}\right)^h V[0] + 2c_0N\theta \frac{1 - \left(1 - \frac{\gamma^{N\theta}}{2}\right)^h}{1 - \left(1 - \frac{\gamma^{N\theta}}{2}\right)} \\
&\quad + \sum_{t=0}^{h-1} \left(1 - \frac{\gamma^{N\theta}}{2}\right)^{h-1-t} \left(2 \sum_{j=tN\theta-\tau-1}^{(t+1)N\theta-\tau-2} c_1[j]\right).
\end{aligned} \tag{6.29}$$

Since  $c_1[k] \rightarrow 0$  in finite time, there exists a finite time  $h_0$  such that  $c_1[k] = 0, k \geq h_0N\theta$ . Then, for  $h \geq h_0$ , we can obtain from (6.29)

$$\limsup_{h \rightarrow \infty} V[hN\theta] \leq \frac{2c_0N\theta}{1 - \left(1 - \frac{\gamma^{N\theta}}{2}\right)} = \frac{4c_0N\theta}{\gamma^{N\theta}} \leq c. \tag{6.30}$$

The analysis is similar for the dynamics of  $V[hN\theta + t], t = 0, 1, \dots, N\theta - 1$ , and we obtain as in (6.29):

$$\limsup_{h \rightarrow \infty} V[hN\theta + t] \leq \frac{4c_0N\theta}{\gamma^{N\theta}} \leq c. \quad \blacksquare$$

As we can see from (6.27), the delays make the consensus error bigger than the one under no delays for every  $N\theta$  steps, i.e., the term containing  $c_1[k]$  is bigger than the one under no delays. However, when the iteration number is large enough as in (6.29), the term containing  $c_1[k]$  converges to 0, which results in the same error bound  $c$  as the one under no delays in the one-hop case [114]. This fact shows that although delays can slow down the consensus process, they do not affect the consensus error bound as also observed in [21], [125].

## 6.4 An Alternative Event-triggered Scheme

In this section, we provide an alternative event-triggered scheme 2, which uses only the event-triggered values to update.

The new update rule is modified from (6.12) (i.e., scheme 1). In (6.12), for node  $i$  to update the new state  $x_i[k+1]$ , its current state  $x_i[k]$  is required. On the one hand, this means that even when the new state is not communicated, it still needs to be stored

---

## 6.4 An Alternative Event-triggered Scheme

at every time step. On the other, since the current state  $x_i[k]$  is newer than  $\hat{x}_i[k]$ , it seems desirable to speed up the convergence. We will, however, show that it may be better to use only  $\hat{x}_i[k]$  for both storage and convergence reasons. Here, scheme 2 is a generalization of the one in [114] which considered the synchronous algorithm with one-hop communication. This type of event-triggered scheme is also motivated by those in [50] and [121].

In the local update, for  $k \in \mathbb{Z}_+$ , each normal node  $i \in \mathcal{N}$  updates its current state by

$$x_i[k+1] = \hat{x}_i[k] + \sum_{j \in \mathcal{D}_i[k]} a_i[k] (\hat{x}_j^P[k - \tau_{ij}^P[k]] - \hat{x}_i[k]), \quad (6.31)$$

where  $a_i[k] = 1/|\mathcal{D}_i[k]|$ .

Note that the new state  $x_i[k+1]$  need not be stored until the next time step, but is merely used for checking the condition of the triggering function  $f_i[k]$  in (6.4). Accordingly, in Algorithm 1, the input should be adjusted such that node  $i$  uses  $\hat{x}_i[k]$  instead of  $x_i[k]$  in determining the set  $\mathcal{D}_i[k]$ .

Next, we denote the state vector containing the event-triggered values of normal nodes from time  $k - \tau$  to time  $k$  as

$$z[k] = [\hat{x}^N[k]^T \ \hat{x}^N[k-1]^T \ \dots \ \hat{x}^N[k-\tau]^T]^T, \quad (6.32)$$

which is a  $N(\tau+1)$ -dimensional vector for  $k \geq 0$ . We also define its maximum value and minimum value as  $\bar{z}[k] = \max z[k]$ ,  $\underline{z}[k] = \min z[k]$ , respectively.

Then, we are ready to present the main result of this section.

**Theorem 6.4.1** *Consider a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $l$ -hop communication, where each normal node updates its value according to the asynchronous event-triggered scheme 2. Under the  $f$ -local Byzantine model, the normal nodes reach resilient consensus at an error level  $c$  if and only if the underlying graph is  $(f+1)$ -strongly robust with  $l$  hops. Moreover, the safety interval is given by  $\mathcal{S} = [\underline{\hat{x}}_\tau[0], \bar{\hat{x}}_\tau[0]]$ , and the consensus error level  $c$  is achieved if the parameter  $c_0$  in the triggering function (6.4) satisfies*

$$c_0 \leq \frac{\gamma^{N(\tau+1)-1}(1-\gamma)}{1-\gamma^{N(\tau+1)-1}}c. \quad (6.33)$$

*Proof: (Necessity)* This part follows from the proof of Theorem 6.3.1.



---

## 6.4 An Alternative Event-triggered Scheme

(*Sufficiency*) First, we show by induction that the safety condition is satisfied. Note that the update rule (6.31) can be rewritten as

$$x_i[k+1] = a_i[k]\hat{x}_i[k] + \sum_{j \in \mathcal{D}_i[k]} a_i[k]\hat{x}_j^P[k - \tau_{ij}^P[k]], \quad (6.34)$$

where  $a_i[k] = 1/|\mathcal{D}_i[k]|$ . At time  $k = 0$ , it is clear by definition that  $x_i[0], \hat{x}_i[0] \in \mathcal{S}$ . We first show that  $\bar{\hat{x}}_\tau[k]$  is nonincreasing in time. From (6.34), we have  $x_i[k+1] \leq \bar{\hat{x}}_\tau[k]$  for all  $i \in \mathcal{N}$  since the values larger than  $\bar{\hat{x}}_\tau[k]$  are ignored in step 2 of Algorithm 1. Moreover, by (6.7), it follows that  $\hat{x}_i[k+1] \leq \bar{\hat{x}}_\tau[k]$  for all  $i \in \mathcal{N}$ . Together, we have  $\bar{\hat{x}}_\tau[k+1] \leq \bar{\hat{x}}_\tau[k]$ . We can similarly prove that  $\underline{\hat{x}}_\tau[k]$  is nondecreasing in time. Besides, using a similar proof, we have  $\bar{z}[k]$  is nonincreasing in time and  $\underline{z}[k]$  is nondecreasing in time. Hence, we have shown the safety condition.

Next, we shown the consensus part. We first sort the normal nodes' values in the vector  $z[k]$  at time  $k$  in increasing order. Denote by  $s_i[k]$  the index of the agent taking the  $i$ th value from the smallest to the largest. Hence, the values are sorted as  $z_{s_1}[k] \leq z_{s_2}[k] \leq \dots \leq z_{s_{N(\tau+1)}}[k]$ .

Introduce two sequences of conditions for the bound of gaps between two values. The condition sequence  $\{A_m\}$  is defined as

- $A_1 : z_{s_2}[k] - z_{s_1}[k] \leq (c_0 + c_1[k])/\gamma$ .
- $A_2 : z_{s_3}[k] - z_{s_2}[k] \leq (c_0 + c_1[k])/\gamma^2$ .
- $\dots$
- $A_{N(\tau+1)-1} : z_{s_{N(\tau+1)}}[k] - z_{s_{N(\tau+1)-1}}[k] \leq (c_0 + c_1[k])/\gamma^{N(\tau+1)-1}$ .

The condition sequence  $\{B_m\}$  is defined as

- $B_{N(\tau+1)} : z_{s_{N(\tau+1)}}[k] - z_{s_{N(\tau+1)-1}}[k] \leq (c_0 + c_1[k])/\gamma$ .
- $B_{N(\tau+1)-1} : z_{s_{N(\tau+1)-1}}[k] - z_{s_{N(\tau+1)-2}}[k] \leq (c_0 + c_1[k])/\gamma^2$ .
- $\dots$
- $B_2 : z_{s_2}[k] - z_{s_1}[k] \leq (c_0 + c_1[k])/\gamma^{N(\tau+1)-1}$ .

Let  $m_A$  be the minimum  $m = 1, \dots, N(\tau+1) - 1$  such that condition  $A_m$  is not satisfied. Also, Let  $m_B$  be the maximum  $m = 2, \dots, N(\tau+1)$  such that condition  $B_m$

---

## 6.4 An Alternative Event-triggered Scheme

is not satisfied. Thus, we have

$$\begin{aligned} z_{s_{m_A+1}}[k] - z_{s_{m_A}}[k] &> \frac{c_0 + c_1[k]}{\gamma^{m_A}}, \\ z_{s_{m_B}}[k] - z_{s_{m_B-1}}[k] &> \frac{c_0 + c_1[k]}{\gamma^{N(\tau+1)-m_B+1}}. \end{aligned} \tag{6.35}$$

Moreover, conditions  $A_1$  to  $A_{m_A-1}$  and  $B_{m_B+1}$  to  $B_{N(\tau+1)}$  are satisfied. Then, for  $0 \leq k \leq k'$ , we introduce two sets

$$\begin{aligned} \mathcal{Z}_1(k, k') &= \{i \in \mathcal{N} : \hat{x}_i[k'] < z_{s_{m_A}}[k] + c_0 + c_1[k]\}, \\ \mathcal{Z}_2(k, k') &= \{i \in \mathcal{N} : \hat{x}_i[k'] > z_{s_{m_B}}[k] - c_0 - c_1[k]\}. \end{aligned}$$

There are several cases concerning the relationship between  $m_A, m_B$  and  $\mathcal{Z}_1(k, k), \mathcal{Z}_2(k, k)$ . We study them separately.

**Case 1.**  $m_A < m_B$ : There are four subcases, denoted by (1-a) to (1-d).

**(1-a)**  $\mathcal{Z}_1(k, k) \neq \emptyset, \mathcal{Z}_2(k, k) \neq \emptyset$ : For a normal node  $j \notin \mathcal{Z}_1(k, k)$ , by definition, the inequality  $\hat{x}_j[k] \geq z_{s_{m_A}}[k] + c_0 + c_1[k]$  holds. Moreover, since the minimum element of  $z[k]$  that exceeds  $z_{s_{m_A}}[k]$  is  $z_{s_{m_A+1}}[k]$ , we also have  $\hat{x}_j[k] \geq z_{s_{m_A+1}}[k]$ . If  $j \in \mathcal{U}[k]$ , then values less than  $z_{s_1}[k]$  will be ignored. Additionally, since the update is based on the convex combination as shown in (6.34), it holds

$$x_j[k+1] \geq (1 - \gamma)z_{s_1}[k] + \gamma z_{s_{m_A+1}}[k]. \tag{6.36}$$

Using conditions  $A_1$  to  $A_{m_A-1}$ , we can bound  $z_{s_1}[k]$  as

$$\begin{aligned} z_{s_1}[k] &\geq z_{s_2}[k] - \frac{c_0 + c_1[k]}{\gamma} \\ &\geq z_{s_3}[k] - \left(\frac{1}{\gamma} + \frac{1}{\gamma^2}\right)(c_0 + c_1[k]) \geq \dots \\ &\geq z_{s_{m_A}}[k] - \left(\frac{1}{\gamma} + \frac{1}{\gamma^2} + \dots + \frac{1}{\gamma^{m_A-1}}\right)(c_0 + c_1[k]). \end{aligned}$$

Substitute this into (6.36) and obtain

$$\begin{aligned}
x_i[k+1] &\geq z_{s_{m_A}}[k] + \gamma \left( z_{s_{m_A+1}}[k] - z_{s_{m_A}}[k] \right) - \frac{1}{\gamma^{m_A-1}} (c_0 + c_1[k]) + c_0 + c_1[k] \\
&> z_{s_{m_A}}[k] + \gamma \frac{c_0 + c_1[k]}{\gamma^{m_A}} - \frac{1}{\gamma^{m_A-1}} (c_0 + c_1[k]) + c_0 + c_1[k] \\
&= z_{s_{m_A}}[k] + c_0 + c_1[k],
\end{aligned} \tag{6.37}$$

where the second inequality is from (6.35).

On the other hand, for an agent  $j \notin \mathcal{U}[k]$ , we have  $x_j[k+1] = x_j[k]$ . By assumption  $j \notin \mathcal{Z}_1(k, k)$ , it holds  $j \notin \mathcal{Z}_1(k, k+1)$ . Either the case is, we conclude that if a normal node is not in  $\mathcal{Z}_1(k, k)$ , then it will not move into  $\mathcal{Z}_1(k, k')$  at time  $k' > k$ . Likewise, for  $\mathcal{Z}_2(k, k)$ , we can show a similar statement.

Furthermore,  $\mathcal{Z}_1(k, k)$  and  $\mathcal{Z}_2(k, k)$  are disjoint. This is because by using the two inequalities in (6.35), from  $1 \leq m_A < m_B \leq N(\tau + 1)$  and  $0 < \gamma \leq 1/2$ , it follows that

$$z_{s_{m_B}}[k] - z_{s_{m_A}}[k] > \max \left\{ \frac{1}{\gamma^{m_A}}, \frac{1}{\gamma^{N(\tau+1)-m_B+1}} \right\} (c_0 + c_1[k]) \geq 2(c_0 + c_1[k]).$$

Notice that the network is  $(f+1)$ -strongly robust with  $l$  hops w.r.t. any set  $\mathcal{F}$  following the  $f$ -local model. Use the discussions in the proof of Theorem 6.3.1, it holds that for the two nonempty disjoint sets  $\mathcal{Z}_1(k, k)$  and  $\mathcal{Z}_2(k, k)$ , one of them (or both) has a normal agent with at least  $f+1$  independent normal paths originating from some normal nodes outside.

Suppose that normal node  $i \in \mathcal{Z}_1(k, k)$  has the abovementioned property. A similar argument holds when  $i \in \mathcal{Z}_2(k, k)$ . If node  $i$  chooses to update at time  $k \leq k^i < k + \theta$  (recall that each normal node should update once in  $\theta$  steps), then  $\hat{x}_i[k'+1] = \hat{x}_i[k']$  and  $x_i[k'+1] = x_i[k']$  for  $k \leq k' < k^i$ . Thus, it holds that  $i \in \mathcal{Z}_1(k, k^i) \subseteq \mathcal{Z}_1(k, k)$ . Hence, node  $i$  still has the abovementioned property. When it updates at time  $k^i$ , it should use at least one value outside the set  $\mathcal{Z}_1(k, k^i)$  (i.e., this value is greater than  $z_{s_{m_A+1}}[k]$ ). Similar to (6.36) and (6.37), we have

$$\begin{aligned}
x_i[k^i+1] &\geq (1-\gamma)z_{s_1}[k^i] + \gamma z_{s_{m_A+1}}[k] \\
&\geq (1-\gamma)z_{s_1}[k] + \gamma z_{s_{m_A+1}}[k] \\
&> z_{s_{m_A}}[k] + c_0 + c_1[k],
\end{aligned}$$

## 6.4 An Alternative Event-triggered Scheme

---

indicating that at time  $k^i + 1$ , node  $i$  moves out of  $\mathcal{Z}_1(k, k)$  (since in this case,  $f_i[k^i] > 0$ ). Thus after  $\theta$  steps, the cardinality of  $\mathcal{Z}_1(k, k + \theta)$  is smaller than that of  $\mathcal{Z}_1(k, k)$ .

Similar results also hold for the case  $i \in \mathcal{Z}_2(k, k)$ . If  $\mathcal{Z}_1(k, k)$  and  $\mathcal{Z}_2(k, k)$  are nonempty, we can repeat the above steps until one of them becomes empty. As a result,  $\bar{z}$  will decrease by  $c_0 + c_1[k]$  (or  $\underline{z}$  will increase by  $c_0 + c_1[k]$ ) after  $N\theta$  steps.

**(1-b)**  $\mathcal{Z}_1(k, k) = \emptyset, \mathcal{Z}_2(k, k) \neq \emptyset$ : We have  $\mathcal{Z}_1(k, k + \theta) = \emptyset$ , i.e.,

$$\underline{z}[k + \theta] \geq z_{s_{m_A+1}}[k] + c_0 + c_1[k] \geq z_{s_1}[k] + c_0 + c_1[k].$$

and  $\underline{z}$  increases by  $c_0 + c_1[k]$  after  $\theta$  steps.

**(1-c)**  $\mathcal{Z}_1(k, k) \neq \emptyset, \mathcal{Z}_2(k, k) = \emptyset$ : Similarly,  $\bar{z}$  decreases by  $c_0 + c_1[k]$  after  $\theta$  steps.

**(1-d)**  $\mathcal{Z}_1(k, k) = \emptyset, \mathcal{Z}_2(k, k) = \emptyset$ : Both the arguments in (1-b) and (1-c) hold.

**Case 2.**  $m_A \geq m_B$ : This case is impossible. We can show this by contradiction. Since  $m_A \geq m_B$ , we know that conditions  $A_{m_B-1}$  and  $B_{m_A+1}$  are both satisfied. Combined with  $A_{m_A}$  and  $B_{m_B}$  not being satisfied, we have

$$\begin{aligned} \frac{c_0 + c_1[k]}{\gamma^{m_A}} &< z_{s_{m_A+1}}[k] - z_{s_{m_A}}[k] \leq \frac{c_0 + c_1[k]}{\gamma^{N(\tau+1)-m_A}}, \\ \frac{c_0 + c_1[k]}{\gamma^{N(\tau+1)-m_B+1}} &< z_{s_{m_B}}[k] - z_{s_{m_B-1}}[k] \leq \frac{c_0 + c_1[k]}{\gamma^{m_B-1}}. \end{aligned} \tag{6.38}$$

The inequalities in the first line in (6.38) indicate that it must hold  $m_A < N(\tau + 1)/2$ . The inequalities in the second line in (6.38) imply that  $m_B > (N(\tau + 1) + 1)/2$ . This results in  $m_A < m_B$ , which contradicts  $m_A \geq m_B$ .

We can now conclude that after a finite number of time steps, all conditions in the sequences  $\{A_m\}$  and  $\{B_m\}$  must be satisfied. Otherwise, the difference between  $\bar{z}$  and  $\underline{z}$  will decrease more than  $c_0$  by an update induced by an event. If such events continuously occur,  $\bar{z} - \underline{z}$  will become smaller and eventually negative, which cannot happen. Therefore, we have  $\limsup_{k \rightarrow \infty} \bar{z}[k] - \underline{z}[k] \leq c$ . ■

**Remark 6.4.1** *In scheme 1, the update frequency  $\theta$  has effects on the consensus error bound. However, in scheme 2, the consensus error bound is affected by the delay bound  $\tau$ . We provide an intuitive explanation for the differences between the two error bounds. In scheme 1, nodes update using the state value  $x_i[k]$  and compare the neighbors' values with  $x_i[k]$ . Hence, if all the nodes update frequently (e.g.,  $\theta = 1$ ), the variance in the values caused by the delays is lower. In other words, the power for an event to be*

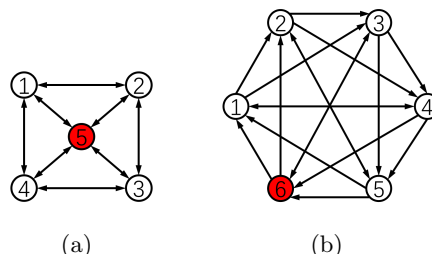


Figure 6.1: (a) The graph is not 2-strongly robust with one hop but is 2-strongly robust with 2 hops. (b) The graph is 2-strongly robust with one hop.

triggered at each node is accumulated at every  $\theta$  steps. In scheme 2, nodes update using the triggered value  $\hat{x}_i[k]$ . Even if all the nodes update frequently, the local update values  $x_i[k+1]$  are not stored. Therefore, the events will not occur until the new event-triggered values from neighbors have arrived.

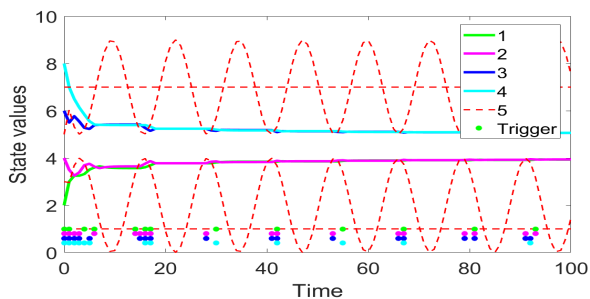
## 6.5 Numerical Examples

In this section, we conduct simulations for networks applying the event-triggered MW-MSR algorithm. For all the simulations, we set the parameters  $c_0$  and  $c_1[k]$  of the triggering function as  $c_0 = 1.215 \times 10^{-2}$  and  $c_1[k] = 0.5 \times e^{-0.06(k+20)}$ , respectively.

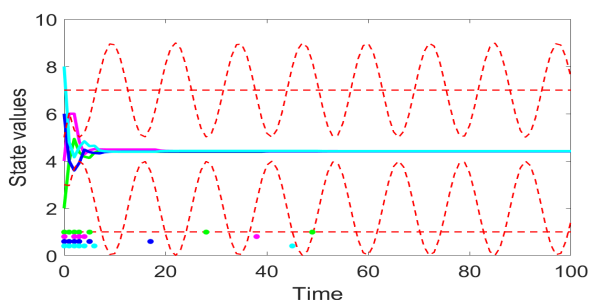
### 6.5.1 Topology Gap between One-hop and Multi-hop Algorithms

In this part, we show that the proposed algorithm can guarantee resilient consensus in a network where the conventional one-hop algorithm cannot. Consider the network in Fig. 6.1(a). This graph is not 2-strongly robust with one hop, but is with 2 hops. Suppose that node 5 is Byzantine and sends four different values to its four neighbors. Let the initial normal states be  $x^N[0] = [2 \ 4 \ 6 \ 8]^T$ . According to [56], [114], this graph does not meet the condition for 1-total Byzantine model even for synchronous updates. Thus, resilient consensus is impossible as shown in Fig. 6.2(a) where the four red dashed lines indicate the adversarial values and the dots represent the time instants when events are triggered by the normal nodes.

Then, we perform simulations for the asynchronous two-hop event-triggered MW-MSR algorithm under the same attacks. Let the normal nodes update synchronously with delays in communication ( $\theta = 1$ ). Moreover, we choose the package relay model, i.e., nodes only relay the messages when events are triggered at the nodes. Observe that



(a) One-hop case without delays.



(b) Two-hop case with delays.

Figure 6.2: Time responses using different event-triggered MSR algorithms.

resilient consensus is achieved as shown in Fig. 6.2(b). This verifies the effectiveness of the proposed algorithm.

### 6.5.2 The Amount of Transmissions of Different Algorithms

In this part, we show that the amount of transmissions of the proposed algorithm can be further reduced compared to the one-hop algorithm. This time, we consider the network in Fig. 6.1(b). This graph is 2-strongly robust with one hop, and hence, with 2 hops. Node 6 is Byzantine and is capable to send two different values to its neighbors (including different relayed values). Let the initial normal states be  $x^N[0] = [2 \ 4 \ 6 \ 8 \ 10]^T$ . By [56] and Theorem 6.3.1, this graph satisfies the condition for 1-total Byzantine model. Thus, resilient consensus can be achieved with both one-hop and two-hop algorithms, and the results are given in Fig. 6.3.

From Fig. 6.3, we can also see that the numbers of events of the two-hop algorithm with immediate relays and package relays are both smaller than that of the one-hop algorithm. This is because by introducing the multi-hop communication, each node can have more information of the network, which may result in faster speed of the consensus

Table 6.1: Average triggering times per normal node

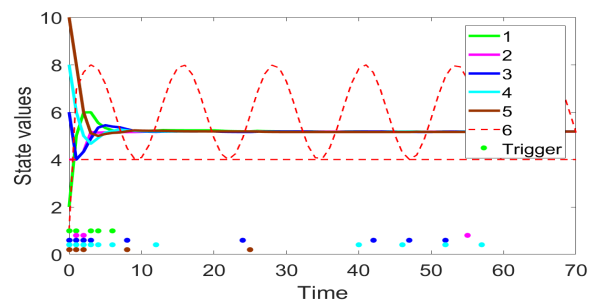
Algorithms	Average events	Average transmissions
One-hop	7.26	7.26
Two-hop with immediate relays	3.05	12.20
Two-hop with package relays	6.99	6.99

process and less events. Moreover, observe that the two-hop algorithm with immediate relays has less events than the algorithm with package relays. Obviously, the immediate relay model is an ideal model and it requires additional communication resources for the relaying process. Note that for this model, each event is accompanied with additional transmissions for relays as each node has three neighbors. In contrast, the package relay model is more realistic and energy-saving since it requires only communication for the events, but reaching consensus takes longer.

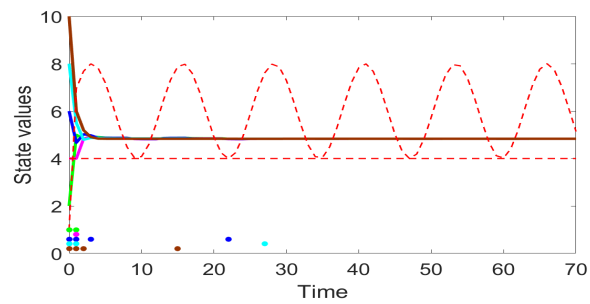
To verify these properties of the algorithms, we further conducted Monte Carlo simulations in the same network for 50 runs by randomly taking initial normal states within  $[0, 10]$ . The Byzantine node 6 misbehaves as in the previous simulation. Table I displays the average times of events and transmissions per normal node of the three algorithms. In all runs, consensus was achieved and the results are consistent with our analysis so far. In particular, the package relay model requires the least number of transmissions overall.

## 6.6 Summary

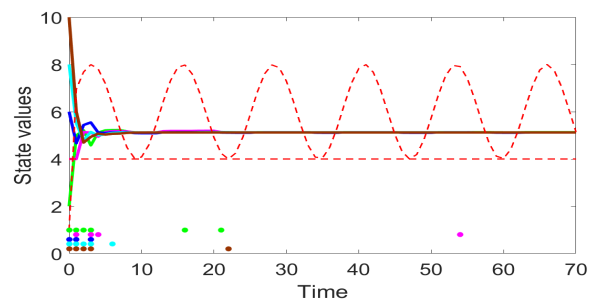
In this chapter, we have investigated the resilient consensus problem using the event-triggered MSR algorithm with multi-hop communication. We have characterized the network requirement for the proposed algorithm to guarantee resilient consensus with a certain error level. We found that the delays in communication may slow down the consensus process, but they do not affect the consensus error. By introducing multi-hop communication, even sparse graphs can meet the condition for robustness. Furthermore, the event-triggered scheme provides an effective way to reduce the number of transmissions for the multi-hop communication.



(a) One-hop case without delays.



(b) Two-hop case with immediate relays.



(c) Two-hop case with package relays.

Figure 6.3: Time responses using different event-triggered MSR algorithms.



## Chapter 7

# Secure Consensus with Distributed Detection via Two-hop Communication

In this chapter, we consider a resilient consensus problem under malicious attacks from the view point of detection algorithms. The approach is to equip all nodes with a scheme to detect neighboring nodes when they behave in an abnormal fashion. To this end, the nodes exchange not only their own states but also information regarding their neighbor nodes which can be considered as a two-hop communication algorithm. It is shown that the detection scheme becomes effective by requiring certain connectivity properties in the network so that the non-malicious nodes can share enough information about their common neighbors.

Specifically, the main objective of this chapter is to address the problem of designing fully distributed FDI methods requiring only local information by the agents. In this setting, each non-faulty, normal agent in the network acts as a detector, monitoring its neighbors by iteratively exchanging more information than in conventional consensus. Specifically, the agents not only send their own states but also relay their neighbors' state values. In this way, they can verify if the states sent by a particular neighbor are consistent with those of others. In the course, we exploit the property in the malicious model that the adversarial nodes are restricted to send the same information to its neighbors.

The key to achieve distributed detection is to impose the network to have sufficient

connectivity. In particular, for the agents to receive trustable information regarding its one/two-hop neighbors, it is critical that there are multiple ways to have access to the neighbors through the presence of common neighbors and multiple paths. By both schemes, we can further achieve consensus among the non-faulty ones in a resilient manner. We clarify tight conditions on the network structures in terms of the graph connectivity for the proposed algorithms. The first scheme has a simple structure as the normal agents can detect malicious neighbors but must rely on secure mobile agents to communicate the detection information to others. On the other hand, the second scheme can perform fully distributed detection. The major difference in their requirements lies in the necessary network structures. The first scheme functions on networks with less connectivity than the second one; this is because in the latter scheme, majority voting [84] is used for agents to decide the true values of the two-hop neighbors. We will see however that the required connectivity level can be less than that in MSR-based resilient consensus algorithms. Moreover, the proposed algorithms can function properly when more than half of the nodes turn malicious under certain topologies, which is a case out of the capabilities of conventional algorithms.

The rest of this chapter is organized as follows. In Section 7.1, the system model is introduced. Section 7.2 is devoted to the basics of the distributed detection framework with detection share. In Section 7.3, we present our main algorithm being capable of fully distributed detection of adversaries. In both cases, we provide necessary and sufficient graph conditions for the detection schemes. In Section 7.4, numerical examples are provided to illustrate the effectiveness of the proposed schemes. We conclude the chapter in Section 7.5.

## 7.1 Problem Formulation

### 7.1.1 Update Rule and Threat Model

Consider a time-invariant network modeled by the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . The node set  $\mathcal{V}$  is partitioned into the set of normal nodes  $\mathcal{N}$  and the set of adversary nodes  $\mathcal{A}$ . The latter set is unknown to the normal nodes at time step  $k = 0$ . The adversary nodes in  $\mathcal{A}$  try to prevent the normal nodes in  $\mathcal{N}$  from reaching consensus. Denote by  $\mathcal{A}_i[k]$  the set of indices of the adversary nodes known to or detected by node  $i$  by time step  $k$ . This set is updated differently in the two proposed schemes and is specified later. The

set of agent  $i$ 's neighbors not behaving adversarially is denoted by  $\mathcal{M}_i[k] = \mathcal{N}_i \setminus \mathcal{A}_i[k]$ . All algorithms in this paper are synchronous. Each normal node  $i$  updates its state value  $x_i[k]$  by

$$x_i[k+1] = \sum_{j \in \mathcal{M}_i^+[k]} \omega_{ij}[k] x_j[k], \quad (7.1)$$

where  $\mathcal{M}_i^+[k] = \{i\} \cup \mathcal{M}_i[k]$  and  $\omega_{ij}[k] = 1/(1 + |\mathcal{M}_i[k]|)$ .

Next, we introduce the threat model studied here. Note that they are the one-hop versions of the threat models shown in Section 2.2.2.

**Definition 7.1.1** (*f-total / f-local set*) *The set of adversary nodes  $\mathcal{A}$  is said to be f-total if it contains at most f nodes, i.e.,  $|\mathcal{A}| \leq f$ . Similarly, it is said to be f-local if for any normal node  $i \in \mathcal{N}$ , it has at most f adversary nodes as its in-neighbors, i.e.,  $|\mathcal{N}_i \cap \mathcal{A}| \leq f, \forall i \in \mathcal{N}$ .*

In this chapter, we focus on the malicious model. This model is reasonable in applications such as wireless sensor networks, where neighbors' information is obtained by broadcast communication. This class of adversaries has not been well studied in computer science where the Byzantine model is traditionally more common [62].

We now introduce the type of consensus among the normal agents to be sought in this chapter. Here, we use the notation  $x_i^{(i)}[k]$  to indicate the state  $x_i[k]$  of each normal agent  $i$  stored by itself. Its formal definition is given in Section 7.1.2.

**Definition 7.1.2** *If for any possible sets and behaviors of the malicious agents and any state values of the normal nodes, the following two conditions are satisfied, then we say that the normal agents reach resilient consensus:*

1. *Safety condition: All normal states remain in the interval  $\mathcal{S} = [\min_{i \in \bar{\mathcal{V}}} x_i^{(i)}[0], \max_{i \in \bar{\mathcal{V}}} x_i^{(i)}[0]]$ , where  $\bar{\mathcal{V}} = \{i \in \mathcal{V} : x_i^{(i)}[0] \in [\bar{x}_{\min}, \bar{x}_{\max}]\}$  determined by the initial states of all agents:  $x_i^{(i)}[k] \in \mathcal{S}, \forall i \in \mathcal{N}, k \in \mathbb{Z}_+$ .*
2. *Consensus condition: There exists a state  $x^* \in \mathcal{S}$  such that  $\lim_{k \rightarrow \infty} x_i^{(i)}[k] = x^*, \forall i \in \mathcal{N}$ .*

Note that for our algorithms it is hard to detect adversary nodes which take extreme initial values but perform the consensus like normal nodes. In fact, it is hard for any algorithm to detect such nodes [31; 40; 44; 130; 131]. Since one can never know that such a node is normal with an extreme initial value or it is simply adversarial. To mitigate

the impact of such adversary nodes, we set the safety interval  $[\bar{x}_{\min}, \bar{x}_{\max}]$  for normal nodes so that neighbors taking values outside this interval will be considered malicious. Here, the safety condition is different from those in MSR-based works [6; 21; 56; 65], where the interval  $\mathcal{S}$  is set only by the initial values of the normal agents.

In this chapter, we develop two distributed schemes for achieving both detection of malicious agents and resilient consensus under the  $f$ -total/local model. A common characteristic of these algorithms is that they both employ two-hop communication, where each agent transmits its own state and the states of its neighbors. We will clarify the necessary network structure for the schemes to accomplish this goal. In particular, our schemes require less connectivity for the networks in comparison with the conventional MSR-based algorithms.

The two proposed schemes are dealt with in Sections 7.2 and 7.3. They differ in two aspects related to communication and network connectivity: The first scheme is introduced more for illustrating the idea behind adversary detection via two-hop communication; it uses secure mobile agents to verify the detection report and send the detection information to all nodes, but the network can be more sparse. The second scheme is the main algorithm of this paper, being capable of fully distributed detection under more dense networks.

Before we proceed, we explain more about the class of MSR-based algorithms. In such algorithms, the normal agents remove the extreme state values at each iteration. Specifically they may remove up to  $f$  largest values and  $f$  smallest values. This indicates that the network must be at least  $(2f + 1)$ -connected. However, it is known that more connectivity is needed. In general, under the  $f$ -total model, resilient consensus can be reached by MSR-based algorithms if and only if the underlying graph is  $(f + 1, f + 1)$ -robust (e.g., [56]).

### 7.1.2 Information Set for Two-hop Communication

In our detection frameworks, each normal node acts as a detector for its neighbors. To this end, the nodes update and exchange their information sets containing their own and neighbors' values, IDs and corresponding identities (normal or malicious). Specifically, at each time step, each node will check the information sets received from its neighbors by comparing them and also with the past information sets. Then, it determines whether the information set is from an adversary node. After confirming the identities

of its neighbors, it will utilize the values of nodes that have behaved normally in the update rule (7.1), and send a new information set containing its updated value and the values of all the neighbors from the previous time step along with corresponding identities.

In the update rule (7.1), each normal agent uses its own state and its neighbors' states. However, to explicitly indicate the difference between the broadcast states and the manipulated states, we introduce two notations. For agent  $i$ , we denote its value by  $x_i^{(i)}[k]$ , to indicate that it is stored in agent  $i$  itself and then broadcasted. Let  $x_i^{(j)}[k]$  be the value stored by its neighbor node  $j$  after receiving the broadcast value  $x_i^{(i)}[k]$ . If node  $j$  is malicious, this information can be modified from  $x_i^{(i)}[k]$  and take a different value when it is stored by agent  $j$ .

Now, for each normal node  $i \in \mathcal{N}$ , we rewrite the update scheme in (??) using these notations as

$$x_i^{(i)}[k+1] = \sum_{j \in \mathcal{M}_i^+[k]} \omega_{ij}[k] x_j^{(i)}[k]. \quad (7.2)$$

For each malicious node  $i \in \mathcal{A}$ , it can update its broadcast state arbitrarily as

$$x_i^{(i)}[k+1] = u_i[k], \quad (7.3)$$

where  $u_i[k]$  may even be a function of states of all nodes in the network by time step  $k$ . (See also Assumption 7.1.2.)

Both normal and malicious nodes transmit information sets to their neighbors. Specifically, the information set  $\Phi_i[k]$  of node  $i$  is sent to its neighbors at time  $k \geq 1$  and contains the value of itself at time  $k$  and past values of itself and its neighbors at time  $k-1$  and corresponding identities of the neighbors, and is set as

$$\Phi_i[k] = \left( (i, x_i^{(i)}[k|k]), \{ (j, x_j^{(i)}[k-1|k]) \}_{j \in \mathcal{N}_i \cup \{i\}}, \mathcal{A}_i[k-1] \right). \quad (7.4)$$

We use the notation  $x_i^{(i)}[k-1|k]$  to indicate that this state is in the set  $\Phi_i[k]$  from time  $k$ . Note that  $\Phi_i[k-1]$  and  $\Phi_i[k]$  contain  $x_i^{(i)}[k-1|k-1]$  and  $x_i^{(i)}[k-1|k]$ , and if node  $i$  is malicious, these values may be different. Moreover, under the malicious model, all neighbors of agent  $i$  receive the same information set from agent  $i$ .

As an example, consider the graph in Fig. 7.1(a). Let the initial state be  $x[0] =$

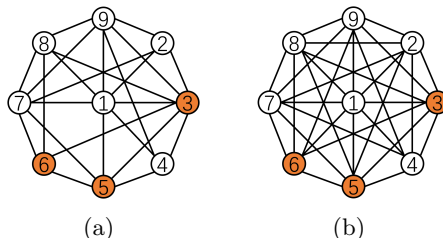


Figure 7.1: 9-node graphs: (a) 4-connected and (b) (4,4)-robust.

$[8 \ 10 \ 4 \ 2 \ 1 \ 5 \ 9 \ 3 \ 6]^T$ . The information set of node 2 at time step 1 is given by

$$\Phi_2[1] = ((2, 7.4), \{(1, 8), (2, 10), (3, 4), (7, 9), (9, 6)\}, \mathcal{A}_2[0]). \quad (7.5)$$

Here, the state of node 2 is updated by (7.2) as the average of the current values with equal weights:  $x_2[1] = (1/5) \sum_{j \in \mathcal{M}_2^+[0]} x_j[0] = 7.4$ . This information set  $\Phi_2[1]$  is sent to node 2's neighbors (i.e., nodes 1, 3, 7, and 9). After confirming that node 2 is normal through the detection algorithm, nodes 1, 3, 7, and 9 will then forward node 2's value to their neighbors at the next time step.

We introduce assumptions on the nodes' knowledge and the attacks that the malicious nodes can generate.

**Assumption 7.1.1** *Each normal node has access to only the information sets received from its neighbors. It also has the topology information of its two-hop neighbors, i.e., the neighbors of its neighbors.*

**Assumption 7.1.2** *Each malicious node has all the information of the network (even if there is no edge from some nodes) and can manipulate its own information set in (7.4) before broadcasting it to the neighbors. It can change the state values and IDs of its own and its neighbors.*

As stated in Assumption 7.1.1, each normal node has only partial knowledge about the network. This is actually a relaxed version of the assumption that each fault-free node knows the topology of the entire network, which is commonly made in the observer based detection works [86; 106], multi-hop communication related works [62; 93], and Byzantine agreement works [109]. In contrast, for most of the MSR-based works [21; 56], each fault-free node is assumed to have access to only the information from its one-hop neighbors. Thus, MSR-based works impose a weaker assumption than

Assumption 7.1.1, however with the tradeoff of not having detection capabilities for malicious agents. In an uncertain environment where neighbors cannot be trusted, it may not be possible to obtain accurate status regarding the two-hop neighbors. To keep the problem tractable, each node is aware of the topology up to its two-hop neighbors in this paper. This setting may be justified and of low cost as in many sensor networks, the nodes are geographically fixed and the network topology will not change.

On the other hand, a malicious node is capable to manipulate any value in its own information set by changing or deleting the value or adding some pairs of values and agent IDs. Since the normal agents have the knowledge of the topology up to their two-hop neighbors, attackers will be known by their direct neighbors when they do not send out their information sets, delete values from neighbors, or add non-existing agents as neighbors. Moreover, in the case that a malicious neighbor of some node adapts the same ID as a normal neighbor, such attacks will be detected too.

At this point, we summarize the common settings for the two schemes mentioned so far: (i) We deal with malicious adversary nodes (including omissive/crash model). (ii) The underlying network graph is time invariant. (iii) The update rules are synchronous. (iv) Both schemes are applied for scalar consensus.

## 7.2 Scheme 1 with Detection Share

In the first scheme for distributed detection and resilient consensus, the normal nodes are capable to detect malicious neighbors by using the two-hop information in undirected networks. It provides the basics for using two-hop communication in an adversarial environment, which is motivated by the works [130; 131].

### 7.2.1 Resilient Consensus Scheme 1

The update rule together with the detection algorithm can be outlined as follows:

**Scheme 1** *Each agent  $i \in \mathcal{V}$  exchanges with its neighbors the information set  $\Phi_i[k]$  in (7.4). Each normal agent first runs the detection algorithm in Algorithm 4. Once it detects any malicious agent in its neighbors, then the detection information is broadcasted to all agents through the secure mobile agents. Finally, it will use the values from its normal neighbors to update its value by the update rule (7.1).*

**Algorithm 4:** Detection Algorithm for Scheme 1**Input:**  $\Phi_j[k], j \in \mathcal{N}_i \cup \{i\}$ **Output:** IDs of the malicious neighbors

Initialization: Take the check set  $\mathcal{C}_i[0]$  and malicious node set  $\mathcal{A}_i[0]$  to be empty. Moreover, node  $i$  receives and stores the initial states of its neighbors as  $x_j^{(i)}[0] = x_j^{(j)}[0]$  for all  $j \in \mathcal{N}_i$ . If  $x_j^{(i)}[0] \notin [\bar{x}_{\min}, \bar{x}_{\max}]$  for any  $j \in \mathcal{N}_i$ , node  $i$  will consider node  $j$  as malicious and put  $j$ 's ID in  $\mathcal{A}_i[0]$ . Then, node  $i$  can update using the values from the nodes in  $\mathcal{M}_i^+[0]$  to get  $x_i^{(i)}[1]$ .

At each time  $k \geq 1$ , node  $i$  executes the following steps:

Let  $\mathcal{A}_i[k] = \mathcal{A}_i[k-1]$ . By Assumption 7.2.1, the malicious nodes detected at time  $k-1$  is shared among all the nodes by time  $k$ .

**for**  $j \in \mathcal{M}_i[k]$  **do**

**if** (Step 1)  $\Phi_j[k]$  contains any different identities of the nodes known by node  $i$  (any node in  $\mathcal{A}_i[k]$  is labeled as malicious, otherwise is labeled as normal.) **then**  
    | it will output node  $j$  as malicious.

**end**

**if** (Step 2)  $\Phi_j[k]$  contains IDs of neighbors of node  $j$  which are different from those known to node  $i$  **then**  
    | it will output node  $j$  as malicious.

**end**

**if** (Step 3) any value of  $x_h^{(j)}[k-1|k], h \in \mathcal{N}_i$ , in  $\Phi_j[k]$  is not equal to the value in the check set  $\mathcal{C}_i[k-1]$  **then**  
    | it will output node  $j$  as malicious.

**end**

**if** (Step 4)  $x_j^{(j)}[k|k]$  in  $\Phi_j[k]$  does not follow the update rule (??) **then**  
    | it will output node  $j$  as malicious.

**end**

**if** (Step 5) node  $j$  has not been detected as malicious by node  $i$  through the 4 steps above **then**  
    | it will output node  $j$  as normal.

**end**

**end**

Return identities (malicious or normal) of neighbors. If node  $i$  detects node  $j$  as malicious or it receives a report on node  $j$  through the detection share, it will put node  $j$ 's ID in  $\mathcal{A}_i[k]$ .

Node  $i$  stores  $x_j^{(j)}[k|k]$  from  $\Phi_j[k], j \in \mathcal{N}_i \cup \{i\}$ , into  $\mathcal{C}_i[k]$ .



### 7.2.2 Detection Algorithm Design

For the first scheme, the detection share function explained below is needed for the communication among the nodes when events of detecting adversaries occur.

**Assumption 7.2.1** *Once a malicious node is detected by any of the normal nodes, its ID will be securely notified to all nodes within the same time step as the malicious node is detected.*

This type of assumptions appears in [130; 131] as well. We however stress that our results have advantages over these works in terms of the network structure requirements. We will make more precise comparisons later. In practice, for this detection share, a certain level of resources is necessary. This can be realized by introducing fault-free mobile nodes which are appropriately distributed throughout the network and are capable to verify if the detection reports from a node is true or false. Here, we suppose that the nodes may turn malicious over time with the upper bound  $f$  on the total number of such nodes. Thus, the verification must take place in real-time. Each time a node claims to have detected a malicious neighbor, the mobile agent nearest to the node visits it and verifies the evidence of the report, i.e., by collecting the information sets of the node and its neighbors of that time step. If it finds the detection report to be valid, then it broadcasts the detection information to all nodes through secure communication [60]. Otherwise, it broadcasts that the node sending the report is malicious. We emphasize that these mobile agents must verify the detection reports only when they receive from agents, and they need not carry out the detection of adversaries themselves, which requires keeping track of the entire network all the time as in [130].

We now present our distributed detection scheme in Algorithm 4. To ensure that all nodes follow the specified update rule, the normal nodes utilize the information set  $\Phi_i[k]$  given in (7.4) and check consistency among the data received from their neighbors. In Algorithm 4, step 1 is to guarantee that each normal node should not use the information from the nodes detected to be malicious by the previous time step. Moreover, it ensures that a node does not falsely claim another node being malicious. Step 2 is to prevent the malicious nodes from faking any neighbors. Step 3 is to enforce the normal nodes not to modify the values received from their neighbors. Finally, step 4 is to guarantee that the normal nodes follow the given update rule.

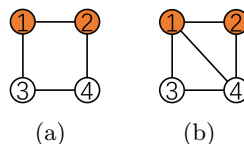


Figure 7.2: Example graphs: Nodes 1 and 2 are malicious. In (a), there is no common normal neighbor, while in (b), node 4 is.

Our approach is distributed as this detection scheme is implemented on each node. By contrast, in [131], a strict assumption on the malicious nodes is imposed so that the cooperation between the malicious nodes is not allowed. In particular, it is assumed that malicious nodes can not be neighbors. It is difficult to guarantee this in practice since clearly the identities of the malicious nodes are unknown prior to operation. Even faults may occur simultaneously in two neighboring nodes. In the later work [130] by the same authors, to relax this assumption, mobile agents are employed. Such agents execute the fault detection and isolation (FDI) function by collecting information as they continuously circulate within the network. However, for Scheme 1, the mobile agents are used only for verification of the detection reports when the events of detection occur. Later, in the next section, we will introduce another detection algorithm with the ability of fully distributed detection of adversaries.

Furthermore, a common assumption made in [40; 130; 131] is that the normal nodes form a connected graph. Although this is a necessary requirement for the normal nodes to achieve consensus, it is impossible to check whether a given graph has this property a priori even if the bound  $f$  on malicious agents is known. In Scheme 1, we address this issue by imposing a connectivity condition to guarantee that the original network has a certain redundant structure.

### 7.2.3 Necessary Graph Structure for Scheme 1

Here, we introduce some conditions on the network structure to fully utilize the detection capability of Algorithm 4. From the definitions of the information sets and the detection algorithm, it is clear that a malicious node can be detected if there is at least one normal node among its neighbors that monitors its behavior. However, such detection may fail if neighboring malicious nodes cooperate with each other. Hence, it is critical that one or more normal nodes are present as their common neighbors. We illustrate this point using the simple four-node network in Fig. 7.2(a). Take nodes 1

and 2 to be malicious. They can cooperate as follows: Node 1 manipulates  $x_2^{(1)}[k-1|k]$  in its information set, and node 2 manipulates  $x_1^{(2)}[k-1|k]$  in its information set. In this network, since there is no normal node having access to the information sets of both nodes 1 and 2, such an attack will not be detected. Now, in the network in Fig. 7.2(b), the normal node 4 is a common neighbor of nodes 1 and 2. As it has access to both  $x_1^{(1)}[k-1|k-1]$  and  $x_2^{(2)}[k-1|k-1]$ , it can detect when node 1 or 2 changes the value of the other.

The following lemma formally states this requirement and its proof can be found in Appendix A of [127].

**Lemma 7.2.1** *Consider the network of nodes modeled by the undirected graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . Algorithm 1 detects every pair of neighboring misbehaving nodes if and only if they have at least one normal node as their common neighbor.*

Since the identities of the malicious nodes are unknown, we must impose a connectivity requirement so that the condition in the lemma holds for any combination of nodes being malicious neighbors in the network. The theorem below provides the main result of this section. Its proof can be found in Appendix B of [127].

**Theorem 7.2.1** *Consider the network modeled by the undirected graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with the adversary set  $\mathcal{A}$  to be an  $f$ -total malicious set. Suppose that Assumptions 7.1.1, 7.1.2 and 7.2.1 hold. Then, under Scheme 1, the following hold.*

(a) *All malicious nodes that behave against the given update rule (7.1) are detected if and only if for every pair of neighboring nodes, they have at least  $f - 1$  two-hop paths connecting them.*

(b) *Under the condition of (a), normal nodes can achieve resilient consensus if  $\mathcal{G}$  is  $(f + 1)$ -connected.*

We note that in the case of undirected graphs, for a pair of neighboring nodes to share common neighbors is equivalent to having two-hop paths connecting them. For directed graphs, we need to be more careful as we will see in Section 7.3. Note that the conditions in Theorem 7.2.1 do not require dense graph structures. For example, we can check by inspection that both graphs in Fig. 7.1 satisfy the conditions for the case with  $f = 3$ .

Our study is motivated by the MSR algorithms studied in, e.g., [22; 56]. There, the notion of graph robustness is shown to be critical to guarantee consensus in the presence

of malicious agents. This can be achieved by the nodes removing extreme values of neighbors while no detection is performed. On the other hand, our approach is to achieve adversary detection for resilient consensus, and this is realized by using extended information sets. As a result, the connectivity requirement becomes less restrictive compared to the MSR algorithms though the necessary resources for communication and computation are higher.

As mentioned before, for the MSR algorithms under the  $f$ -total malicious model, resilient consensus is guaranteed if and only if the graph is  $(f + 1, f + 1)$ -robust. It is known from [56] that such a graph has the property of being  $(f + 1)$ -connected, but the converse does not hold in general. The difference between these classes of graphs can be checked by the two graphs in Fig. 7.1. The graph in (a) is 4-connected and moreover satisfies the two-hop condition in Theorem 7.2.1 for  $f = 3$ . The one in (b) on the other hand is  $(4,4)$ -robust as required by the MSR algorithm with  $f = 3$ . Hence, when the number of malicious nodes in the network is the same, the constraint on the graph structure for Scheme 1 is less stringent than that for MSR algorithms.

## 7.3 Scheme 2 with Fully Distributed Detection

Having the basics established for the detection via two-hop communication, we present our main result on Scheme 2 for resilient consensus in this part. Compared to Scheme 1, it is notable that the new scheme can achieve fully distributed detection for each normal node without the use of detection share. This feature can be realized by introducing majority voting [8; 84] and requiring a more dense graph structure. Then we provide a necessary and sufficient condition on graph structures for the detection part of Scheme 2. Lastly, we show that Scheme 2 can tolerate more malicious nodes in both complete networks and incomplete networks compared to MSR-based algorithms.

### 7.3.1 Resilient Consensus Scheme 2

The update rule together with the detection algorithm can be outlined as follows:

**Scheme 2** *Each agent  $i \in \mathcal{V}$  sends its out-neighbors the information set  $\Phi_i[k]$  in (7.4). After receiving the information sets from its in-neighbors, it first runs the detection algorithm in Algorithm 5. If it detects any malicious neighbors, it reports this detection information to its out-neighbors. Then, it will utilize the values from its normal*

neighbors to update its value by the update rule (7.1).

It is important to note that malicious nodes may send fake detection reports to normal nodes in this scheme, which is different from Scheme 1 where the secure detection share is utilized. Hence, the normal nodes must verify if each received data is authentic.

#### 7.3.2 Detection Algorithm Design

Malicious neighbor sets here enable the normal nodes to keep track of its neighbors identified to be malicious.

**Definition 7.3.1** (*Malicious neighbor set  $\mathcal{A}_i[k] \subset \mathcal{V}$* ) *Once node  $i$  detects any malicious node among its neighbors at time  $k$ , it puts the node's ID in  $\mathcal{A}_i[k]$ . Also, once node  $i$  receives at least  $f + 1$  detection reports on some node, it will put the node's ID in  $\mathcal{A}_i[k]$ . This set is accessible only to node  $i$  itself.*

From Scheme 1, we see that the information set plays a crucial role in our detection framework. For any normal node  $i$  to verify the identity of its neighbors, the information sets of the neighbors need to be investigated in two parts: (i) the current value, i.e., if it is updated according to the given update rule; (ii) the past values, i.e., if they are manipulated and different from the true values of the corresponding nodes.

The two proposed schemes share common features in checking the neighbors' current values. Nevertheless, certain difference lies between the two schemes about checking the past values of the neighbors. With the detection share function in Scheme 1, normal node  $i$  needs to check whether its own past value is manipulated in the information sets of its one-hop neighbors. By contrast, in Scheme 2, node  $i$  should check whether any entries of the past values are manipulated in the information sets of its one-hop neighbors. Thus node  $i$  needs to obtain the true state values of its neighbors' neighbors, i.e., two-hop neighbors. For this purpose, the information sets of its one-hop neighbors must be utilized. Among the multiple information sets containing the state value of its two-hop neighbor  $h$ , there may be some malicious node relaying a wrong value of node  $h$ . Thus node  $i$  needs to carry out a majority voting on the true state value of two-hop neighbor  $h$  through all the one-hop neighbors' information sets containing the value of node  $h$ . Here, majority voting means that if node  $i$  receives  $m$  values of node  $h$ , among

### 7.3 Scheme 2 with Fully Distributed Detection

---

the  $m$  values, if more than  $m/2$  values are the same, then node  $i$  will take this value as the true value of node  $h$ .

To fully utilize the capability of Algorithm 5, we here define the necessary graph condition for Algorithm 5.

**Definition 7.3.2** *A directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is said to satisfy the condition for Algorithm 2 with parameter  $f$  if for any in-neighbor  $j \in \mathcal{N}_i$  of node  $i$ , and any  $h \in \mathcal{N}_j, h \neq i$ , one of the following conditions holds:*

1.  $h \in \mathcal{N}_i$ ;
2.  $h \notin \mathcal{N}_i$ , and there are at least  $2f + 1$  directed two-hop paths from  $h$  to  $i$  (including the one through  $j$ ).

We now present our distributed detection in Algorithm 5. Here, each node  $i$  performs majority voting on two things: the nodes' values and detection information. Since we consider the  $f$ -total/ $f$ -local model in this paper, at most  $f$  values could be false in the neighborhood of node  $i$ . Thus if node  $i$  receives the same information from at least  $f + 1$  distinct neighbors, it considers this information trustable. After obtaining the true values of its one-hop neighbors and two-hop neighbors, it follows the same detection procedures as the ones in Scheme 1.

#### 7.3.3 Necessary Graph Structure for Scheme 2

In directed networks, the necessary condition for node  $i$  to detect its neighbor  $j$  when it misbehaves is the following: node  $i$  has full access to the information that node  $j$  must use to update its value if node  $j$  is normal, that is, the true values  $x_h^{(h)}[k - 1]$ ,  $\forall h \in \mathcal{N}_j$ , used in the control input of node  $j$  and the correct detection information of the two-hop neighbor  $h$ .

Similar to Scheme 1, we must impose a connectivity requirement on every node and its neighbors for Scheme 2, such that the detection is guaranteed for any combination of nodes being malicious in the network. The following theorem is the main result of this section.

Note that this result is stated for the  $f$ -local model. However, it applies to the case of  $f$ -total model. This is because  $f$ -local model is more general and adversarial than the  $f$ -total model, and more than  $f$  malicious agents in total may be in the entire network.

---

**Algorithm 5:** Detection Algorithm for Scheme 2

---

**Input:**  $\Phi_j[k], j \in \mathcal{N}_i \cup \{i\}$

**Output:** IDs of the malicious neighbors

Initialization: Node  $i$  creates a check set  $\mathcal{C}_i[k-1]$  to store the values of one-hop neighbors  $x_j^{(j)}[k-1]$  and also the values of two-hop neighbors  $x_h^{(h)}[k-1]$  at time  $k-1$ . Then, node  $i$  executes the same procedures as those in the initialization of Algorithm 1.

At each time  $k \geq 1$ , node  $i$  executes the following steps:

For each value of two-hop neighbor  $x_h^{(h)}[k-1]$ , it gathers  $x_h^{(j)}[k-1|k]$  from  $\Phi_j[k], j \in \mathcal{N}_i$ , does the majority voting on the value of  $x_h^{(h)}[k-1]$  and stores  $x_h^{(h)}[k-1]$  into  $\mathcal{C}_i[k-1]$ .

Let  $\mathcal{A}_i[k] = \mathcal{A}_i[k-1]$ . Once node  $i$  receives at least  $f+1$  detection reports on some node (contained in  $\mathcal{A}_j[k-1]$  in  $\Phi_j[k], j \in \mathcal{N}_i$ ), it puts the node's ID in  $\mathcal{A}_i[k]$ .

**for**  $j \in \mathcal{M}_i[k]$  **do**  
| **Steps 1-5 in Algorithm 4.**  
**end**

Return identities (malicious or normal) of neighbors. If node  $i$  detects node  $j$  as malicious, it will put node  $j$ 's ID in  $\mathcal{A}_i[k]$ .

Node  $i$  stores  $x_j^{(j)}[k|k]$  from  $\Phi_j[k], j \in \mathcal{N}_i \cup \{i\}$ , into  $\mathcal{C}_i[k]$ .

---

**Theorem 7.3.1** *Consider the network modeled by the directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  where the adversary set  $\mathcal{A}$  follows the  $f$ -local malicious model. Suppose that Assumptions 7.1.1 and 7.1.2 hold. Then, under Scheme 2, the following hold.*

(a) *All malicious nodes that behave against the given update rule (7.1) are detected if and only if  $\mathcal{G}$  satisfies the condition for Algorithm 2 with parameter  $f$  (Definition 7.3.2).*

(b) *Under the condition of (a), normal nodes can achieve resilient consensus if  $\mathcal{G}$  has  $(f+1)$ -connected rooted spanning trees.*

*Proof:* (a) *Necessity:* We prove by contradiction. Suppose that there is a node  $h \in \mathcal{N}_j$  with  $h \notin \mathcal{N}_i$ , and that there are at most  $2f$  two-hop paths from node  $h$  to node  $i$  including the path containing node  $j$ . Take node  $j$  to be malicious. In this case, node  $i$  will get copies of  $x_h^{(h)}[k-1]$  from at most  $2f$  different information sets. Here, note that node  $i$  can also obtain copy of  $x_h^{(h)}[k-1]$  from  $\Phi_j[k]$ , i.e.,  $x_h^{(j)}[k-1]$ . Among the  $2f$  copies of  $x_h^{(h)}[k-1]$ , no majority is guaranteed when we consider the worst case. That is, there may be  $f$  identical values created by malicious nodes and  $f$  identical

### 7.3 Scheme 2 with Fully Distributed Detection

---

values created by normal nodes. Thus node  $i$  cannot determine which one is actually the true value of  $x_h^{(h)}[k-1]$ . Hence, node  $j$  cannot be detected by node  $i$ .

*Sufficiency:* For detection, we prove sufficiency by showing that node  $i$  can confirm the true value of every entry of the information set  $\Phi_j[k]$  of neighbor node  $j$  by obtaining the true value  $x_h^{(h)}[k-1]$  of every neighbor  $h \in \mathcal{N}_j$  of node  $j$ , from the previous time step  $k-1$ . Moreover, node  $i$  can obtain the correct detection information of its two-hop neighbors before the detection loop at time  $k$ . Then we can prove that node  $i$  will detect node  $j$  at time  $k$  if node  $j$  sends out faulty  $\Phi_j[k]$ .

For node  $i$ , consider the following two cases separately: (i) only condition 1 holds; (ii) only condition 2 holds.

(i) In the case where  $h \in \mathcal{N}_i$ , it is clear that node  $i$  can receive the true value of  $x_h^{(h)}[k-1]$  from  $\Phi_h[k-1]$  and have the correct detection information of its one-hop neighbor  $h$  before time  $k$ .

(ii) Suppose that  $h \notin \mathcal{N}_i$ , and there are at least  $2f+1$  directed two-hop paths from node  $h$  to node  $i$ . In this case, there is some normal node  $l \in \mathcal{N}_h^{\text{out}} \cap \mathcal{N}_i$  which carries the true value of  $x_h^{(h)}[k-1]$  in its information set  $\Phi_l[k]$ . If the majority of the  $2f+1$  paths from  $h$  to  $i$  contains nodes as  $l$ , then node  $i$  can get the true value of  $x_h^{(h)}[k-1]$ . Since there are at most  $f$  malicious nodes among the in-neighbors of node  $i$ , and there are at least  $2f+1$  directed two-hop paths from  $h$  to  $i$  including the path containing node  $j$ , we have the needed majority.

We can apply the same analysis on the detection information of node  $i$ 's two-hop neighbors. In the same case, if node  $h$  sends out faulty  $\Phi_h[k-1]$ , then it is detected by its one-hop neighbors at time  $k-1$ . Recall that there are at least  $2f+1$  directed two-hop paths from node  $h$  to node  $i$  and at most  $f$  malicious nodes among the in-neighbors of node  $i$ . Thus, node  $i$  can obtain the correct identities of its two-hop neighbors by majority voting before the detection loop of time  $k$ .

Therefore, node  $i$  knows the true value of  $x_h^{(h)}[k-1]$  and obtains the correct detection information of its two-hop neighbors  $h$  before running the detection loop at time  $k$ . Thus if node  $j \in \mathcal{N}_i$  sends out faulty  $\Phi_j[k]$  by possible manipulation including modifying the entry of  $x_h^{(j)}[k-1]$  in  $\Phi_j[k]$ , by simply breaking the update rule, or by sending false information on the identity of node  $h$ , then node  $i$  will detect.

(b) Malicious nodes will be detected immediately once they misbehave. Thus misbehaviors of malicious nodes cannot affect normal nodes since normal nodes exclude



values from detected malicious nodes. Hence, the safety condition is guaranteed. Moreover, by the graph  $\mathcal{G}$  having  $(f + 1)$ -connected rooted spanning trees, after removing  $f$  malicious nodes, the subgraph of normal nodes contains at least one rooted spanning tree. Therefore, resilient consensus is achieved ([83]). ■

#### 7.3.4 Discussion

The conditions for Scheme 2 guarantee that for any out-neighbor  $i$  of node  $j$ , it has full access to the broadcast values of in-neighbors of node  $j$ . It gains the values either by being a direct neighbor of node  $j$ 's in-neighbors or through majority voting over at least  $2f + 1$  paths. For the latter case, the majority of the voting always exists and is correct since there are at most  $f$  malicious nodes in the neighbors of node  $i$  due to the  $f$ -total/ $f$ -local model. Also, the majority voting enables node  $i$  to verify if a detection report on its two-hop neighbors is valid. In the computer science literature, similar redundancy schemes are often used to provide security and reliability to systems. For example, if a transmission system is designed to tolerate up to  $f$  failures, it must have  $2f + 1$  copies of the transmitted information along with majority voting for verification [8].

Here we provide some example graphs satisfying the conditions in Theorem 7.3.1. The network in Fig. 7.3(a) satisfies the conditions for Scheme 2 under the 1-local model, i.e., there is at most one malicious node in the neighbors of each normal node. Moreover, there is a characteristic three-layer structure. We can extend this idea to the cases with any  $f$ . Each layer should have  $2f + 1$  nodes for  $f$ -total/ $f$ -local model. Each node in one layer should be connected with every node in the neighbor layers and have no connection with the nodes in its own layer. This structure can also have many layers as long as the  $f$ -total or  $f$ -local set is satisfied for each  $i \in \mathcal{N}$ . Furthermore, combining this structure with complete subgraphs (i.e., cliques), we can have graph structures like Fig. 7.3(b), which satisfies the conditions for Scheme 2 as well.

It is observed that each node in a complete graph can detect every malicious node since it has access to the state value of every node in the network. Thus we can enhance the performance of Scheme 2 by introducing nodes having such properties. Node  $i$  is said to be a full access node if it is an out-neighbor of all other nodes in the network, i.e.,  $d_i = n - 1$ .

It is important to note that we do not assume such full access nodes to be normal.

### 7.3 Scheme 2 with Fully Distributed Detection

---

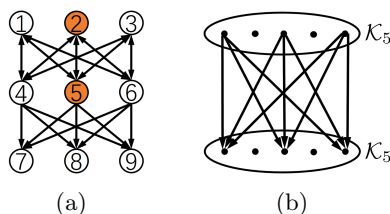


Figure 7.3: Example graphs satisfying the criteria for Scheme 2.

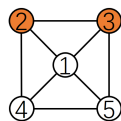


Figure 7.4: Undirected graph of 5 nodes with nodes 2 and 3 to be malicious.

As long as the conditions for Scheme 2 are met, a full access node can also be detected by its normal neighbors when it behaves maliciously. This setting is different from the authorized central nodes and the mobile detectors in [130], which randomly visit each node in the network and are assumed to be fault-free. Nevertheless, a full access node has the following property when it is normal. This result can be easily proved by Theorem 7.3.1 since any node in the network is an in-neighbor of the full access node.

**Corollary 7.3.1** *A normal full access node can detect any node that behaves against the update rule (7.1) in the network under Scheme 2.*

As a result, Scheme 2 can guarantee resilient consensus in incomplete networks when the majority of the nodes are normal, if a full access node is deployed properly in the network. For example, the five-node network in Fig. 7.4 could tolerate two malicious nodes when the conditions for 1-local are met except for the full access node 1. In the same graph, if only node 1 becomes malicious and the conditions for 1-local are also met for other nodes, then resilient consensus is still guaranteed.

The use of full access nodes may be difficult in practice. In the simulation, we will examine another approach to enhance the connectivity of the network by the introduction of relay nodes. Such nodes are limited in number, but forward the received messages with stronger transmission power so that the messages reach more nodes in the system. The use of relay nodes has been well studied for wireless sensor networks (e.g., [129]).

Now, by the role that full access nodes can play, we can derive the maximum

### 7.3 Scheme 2 with Fully Distributed Detection

---

tolerable number of malicious nodes in incomplete graphs for Scheme 2. See Appendix C of [127] for the proof.

**Proposition 7.3.1** *Consider the network of nodes modeled by the incomplete directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $n$  nodes. Assume that it meets the conditions for Scheme 2 given in Theorem 7.3.1 (under the  $f$ -total model). Then Scheme 2 detects all the  $f$  malicious nodes in the network and guarantees resilient consensus only if  $n > 2f$ .*

As we have seen in Proposition 7.3.1, Scheme 2 can tolerate  $n \geq 2f + 1$  in incomplete networks if full access nodes are deployed properly. For Scheme 1, we can similarly obtain a necessary bound on the number of malicious nodes as  $n \geq f + 3$  by following the proof technique of Proposition 7.3.1. These bounds are for incomplete graphs and are conservative if applied to complete graphs. For Scheme 2, the bound for complete graphs is shown to be  $n \geq f + 2$  as stated in the following corollary of Theorem 7.3.1. This bound in fact holds for Scheme 1 too, which can be derived as a corollary of Theorem 7.2.1.

**Corollary 7.3.2** *For a complete graph  $\mathcal{K}_n$ , it can tolerate  $f \leq n - 2$  malicious nodes in the graph for the normal ones to reach resilient consensus by using Scheme 2.*

We summarize the maximum tolerable numbers of malicious nodes in complete graphs for several algorithms in Table 7.1. In computer science [9; 56; 62], a common limitation is that MSR-based algorithms have the maximum tolerable number of malicious nodes  $n \geq 2f + 1$  only for complete graphs. In comparison, it is clear that the proposed Schemes 1 and 2 can tolerate more adversaries.

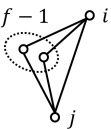
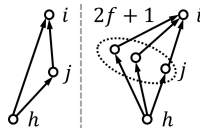
We conduct some comparisons between Schemes 1 and 2. Their differences are shown in Table 7.2. Observe that Scheme 1 requires less connections in graphs compared to Scheme 2. For example, in the 9-node networks in Fig. 7.5, networks (a) and (b) satisfy the conditions for Schemes 1 and 2 under 2-total malicious model, respectively. It is observed that in this case strictly more connections are required for Scheme 2. Thus, in general networks without any full access node, the requirement for Scheme 1 is easier to meet than that for Scheme 2.

We remark that for Scheme 2 under  $f$ -local model the tolerable number of malicious nodes in incomplete graphs could be more than the bound  $n \geq 2f + 1$ . As mentioned in Corollary 7.3.1, full access nodes can detect any malicious node in the network. Thus in

Table 7.1: Tolerable number of malicious nodes for complete graphs.

Scheme 1	Scheme 2	W-MSR	MSR
$n \geq f + 2$	$n \geq f + 2$	$n \geq 2f + 1$	$n \geq 2f + 1$

Table 7.2: Differences between Schemes 1 and 2.

	Scheme 1	Scheme 2
Network	Undirected	Directed
Distributed detection	With detection share	Fully distributed detection
Malicious model	$f$ -total	$f$ -total $f$ -local
Graph condition (detection)	 (a)	 (b)

dense graphs which are close to complete graphs, Scheme 2 can function properly even when more than half of the nodes turn malicious. For example, in the 9-node incomplete network in Fig. 7.7, Scheme 2 performs well even when there are 6 malicious nodes. More details are discussed in the numerical examples.

## 7.4 Numerical Examples

In this section, we demonstrate the performance of the proposed detection schemes through numerical examples. We first use small-scale networks to verify the theoretical results and then conduct extensive simulations based on geometric random graphs of larger scale.

### 7.4.1 Resilient Consensus under Scheme 1

Consider the network shown in Fig. 7.1(a). It is a 4-connected graph with at least two two-hop paths connecting every pair of neighbors. Given these properties, Theorem 7.2.1 indicates that Scheme 1 can detect and remove at most three malicious nodes, i.e.,  $f \leq 3$ , and resilient consensus is guaranteed. Here, we set nodes 3, 5, and 6 to be

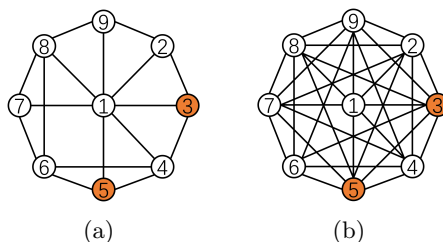


Figure 7.5: 9-node graphs under 2-total malicious model.

malicious as indicated in orange in Fig. 7.1(a).

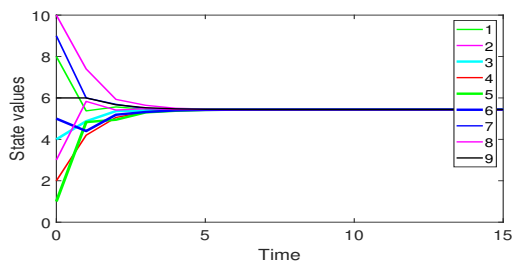
First, we examined the case without attacks. The time responses of the states of all nodes arriving at consensus are shown in Fig. 7.6(a). Next, in attack scenario 1, nodes 3 and 6 try to cooperate to avoid being detected. The simulation result is shown in Fig. 7.6(b). By time  $k = 4$ , consensus is almost achieved among normal nodes, but the malicious nodes start to manipulate their information sets. Specifically, node 3 changes the past value received from node 6 and similarly node 6 changes the past value received from node 3. These attacks are quickly detected. We indicate the events of malicious node detections by dashed lines. Here, for instance, the dashed line at time 5 indicates that nodes 3 and 6 are detected as malicious. In this case, we observe that Scheme 1 performs well. Finally, we note that for the W-MSR algorithm from [56], the 4-connected network in Fig. 7.1(a) is not sufficient to achieve resilient consensus.

#### 7.4.2 Resilient Consensus under Scheme 2

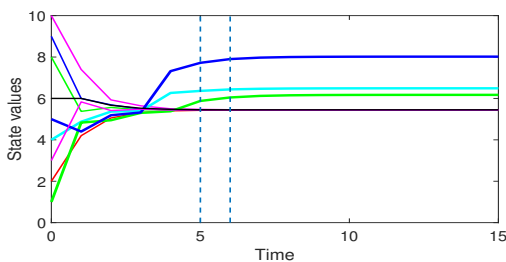
Next, we consider the network shown in Fig. 7.7 with 5 in-coming edges of node 1 removed from the complete graph  $\mathcal{K}_9$ . It satisfies the condition for Scheme 2 under 1-local malicious model in Theorem 7.3.1 for non-full access node 1. Here, we set nodes 2, 3, 4, 5, 6 and 7 to be malicious and the initial state to be  $x[0] = [8 \ 10 \ 4 \ 2 \ 1 \ 5 \ 9 \ 3 \ 6]^T$ .

First, we examined the case without attacks. The time responses of the states of all nodes are shown in Fig. 7.8(a). Next, in attack scenario 1, malicious nodes 2, 3, 5, 6 and 7 manipulate their own values; also, node 4 is malicious and keeps using the values received from them. The simulation result is shown in Fig. 7.8(b). At time  $k = 3$ , these attacks start, but are immediately detected at the next time step, with normal nodes not affected. In both cases, the normal nodes achieve consensus.

Now, we discuss the applicability of the MSR algorithm from [56] under the same network. As discussed earlier, for this algorithm, the connectivity structure of the



(a) No attack.



(b) Attack scenario 1.

Figure 7.6: Scheme 1: Time responses of the states of all nodes.

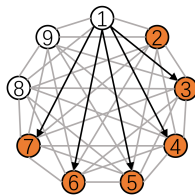
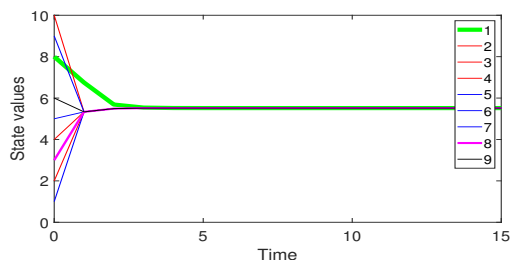
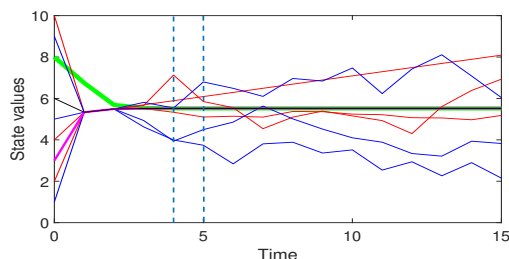


Figure 7.7: 9-node network satisfying the criteria for Scheme 2.

network in Fig. 7.7 is not sufficient for tolerating 6 malicious nodes. In fact, for a network with 9 nodes, even if it is a complete graph, only up to 4 malicious nodes can be tolerated ([56]). In general, it is impossible for MSR algorithms to function properly when more than half of the nodes are malicious. Moreover, we also analyze how the iterative approximate Byzantine consensus (IABC) algorithm from [102] performs under the same network. Consider the case when each node knows the topology of two-hop neighbors and the relay depth is two-hop, as we assume for Scheme 2 here. To meet the condition in [102], node 8 should be connected with at least 9 nodes if we consider the 4-local case for node 8, but this is obviously not true in this network. Thus node 8 cannot make agreement with other normal nodes. This is because in [102], the more adversarial class of Byzantine nodes is considered.



(a) No attack.



(b) Attack scenario 1.

Figure 7.8: Scheme 2: Time responses of the states of all nodes.

### 7.4.3 Application to Large Wireless Sensor Networks

In this simulation, we create a WSN composed of 100 nodes. At first, we place them at random locations in a  $100 \times 100$  planar box. Each node can communicate only with the nodes located within the communication radius of  $r$ . Once  $r$  is determined, a random geometric network is formed. By increasing communication radius  $r$ , the network becomes denser and eventually a complete network when  $r \geq 122$ . After the network is formed,  $f$  nodes are randomly selected to be malicious nodes satisfying our assumptions mentioned before. When we add more malicious nodes in the network, we keep the malicious nodes chosen before and turn normal nodes to new malicious nodes. Then we apply the proposed schemes and the W-MSR algorithm to the network.

Recall that the malicious nodes can manipulate their own information sets by manipulating either the current value or values from the last time step. In particular, we consider the following two attack scenarios: (i) The first scenario is static in the sense that each malicious node takes a fixed value of 120. (ii) The second scenario is more dynamic as each malicious node randomly selects a neighbor and modifies the value from that neighbor arbitrarily and follows the update rule.

Here, we examine how the network connectivity affects the performance of the proposed resilient consensus schemes under the two attack scenarios. Using different values

for the number of malicious agents  $f$  and the communication radius  $r$ , we compare the following four algorithms: (i) The original consensus algorithm without adversaries, (ii) W-MSR algorithm, (iii) Scheme 1, and (iv) Scheme 2. For each  $f$  and  $r$ , we computed the success rate of each algorithm over 20 Monte Carlo runs with randomly chosen initial values of the normal agents in the interval  $[0,100]$ . The results of the consensus algorithm without adversaries provide the baseline, indicating when the network becomes connected.

The results under the first attack scenario are presented in Fig. 7.9. In the plots (a)–(b), we increased the number  $f$  of adversaries. Notice that the two proposed schemes are clearly effective against the malicious nodes, and their success rates remain almost the same as the case without any adversaries, where the success rates become 1 around  $r = 20$ . In contrast, the conventional W-MSR degrades in its performance as the number of malicious nodes increases.

The difference between the two proposed schemes becomes more evident under the second attack scenario. In Fig. 7.10, the results are shown as in Fig. 7.9. It is obvious that Scheme 1 is capable to reach resilient consensus similarly to the previous case under scenario 1. However, under this scenario, Scheme 2 performs even worse than the W-MSR approach. On the other hand, an interesting phenomenon can be observed for the case  $f = 60$  in Fig. 7.10(c), where  $f > n/2$  holds. Both Schemes 1 and 2 can guarantee resilient consensus when the network becomes complete while for the W-MSR algorithm, this is not possible. This verifies our analysis before in Section 7.3. We highlight again that among the two proposed algorithms, Scheme 2 is fully distributed and more scalable. Thus, there is a tradeoff between the requirements on network connectivities and computation resources.

We next check that the performance of Scheme 2 can be significantly improved by increasing the number of edges in the network. Earlier in Section 7.3, through example graphs in Fig. 7.5, we have seen such a property more analytically. Here, to increase edges, we introduce 16 additional relay nodes as discussed in Section 7.3.4. Such a node has strong communication capabilities; it receives data from the nodes within their communication ranges of radius  $r$  and then simply sends out, or relays, the received data to nodes within its own communication radius  $r_{\text{relay}} = r + 27$ . Relay nodes are located at the coordinates  $(20x, 20y)$ ,  $x, y = 1, 2, 3, 4$ . In our setting, such nodes only relay information received from general nodes and not from other relay



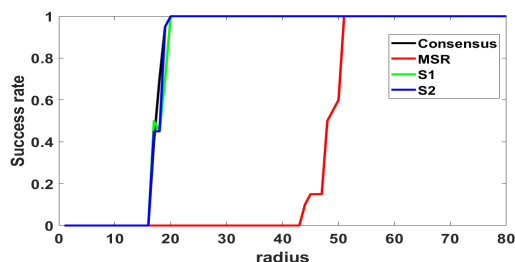
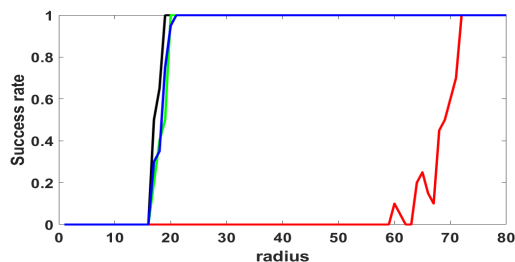
(a)  $f = 15$ .(b)  $f = 30$ .

Figure 7.9: Performance of different resilient consensus algorithms under attack scenario 1.

nodes; also they do not conduct any detection nor consensus algorithm. These nodes bring in the same effects as introducing more directed edges in the network. In Fig. ??, the success rates of Scheme 2 and the W-MSR algorithm are indicated, respectively, by the blue and red dashed lines. One can observe that Scheme 2 performs almost the same as Scheme 1 and much better than the W-MSR algorithm especially when  $f$  grows.

## 7.5 Summary

In this chapter, we have designed two novel distributed detection schemes to solve the resilient consensus problem. The key features of the schemes lie in the assumption on the adversaries based on the malicious agent model and the use of two-hop communication among the agents. We have clarified that the levels of network connectivities for both schemes can be more sparse compared to conventional approaches. In the two schemes, the normal agents perform as detectors by monitoring the behaviors of their neighbors, but they are different in terms of distributed computation capabilities and the required network connectivities. We have presented their properties through extensive numerical examples.

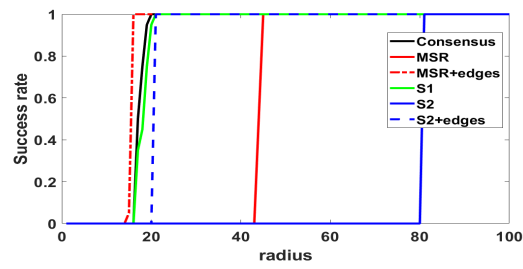
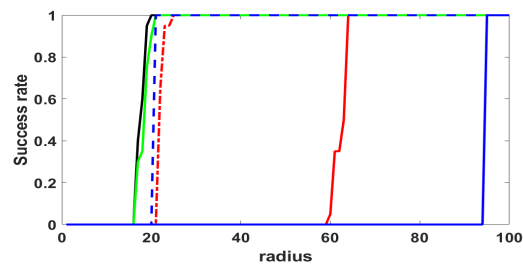
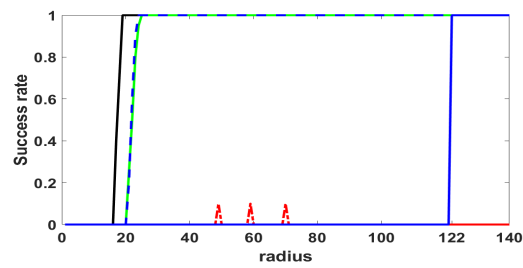
(a)  $f = 15$ .(b)  $f = 30$ .(c)  $f = 60$ .

Figure 7.10: Performance of different resilient consensus algorithms under attack scenario 2.

# Chapter 8

## Conclusion

### 8.1 Summary of Contributions

In the last decade, distributed systems under cyber-attacks have gained much attention. Simultaneously, with the growth of security issues in Cyber-Physical Systems, consensus in the presence of adversary agents or attacks has become a vital issue and needs to be studied urgently. For the conventional one-hop MSR algorithms, the required graph connectivity is stringent, which has limited applications for general networks. On the other hand, for the conventional detection based algorithms, they usually require additional centralized/authorized resources for the detection function. To broaden the applications of resilient consensus algorithms for general networks, we have made advances in both directions as presented in this thesis.

More concretely, we have generalized the MSR algorithms with multi-hop communication in order to make the resilient consensus succeed in sparse networks. Specifically, we have developed the multi-hop weighted MSR algorithm with synchronous and asynchronous updates. We analyzed its performance under different types of attacks (i.e., the  $f$ -total/local malicious/Byzantine models). We found necessary and sufficient conditions guaranteeing resilient consensus under different attacks. These conditions are expressed in terms of graph robustness with  $l$  hops, which is a notion extended from the existing literature. Example graphs and simulations have been provided to verify that the proposed algorithm can achieve resilient consensus in sparse networks.

Besides, we have also developed the quantized version of the MW-MSR algorithms such that our algorithm can also be applied in the multi-agent networks where agents have only limited computational resources for computing integer values. Lastly, to

further reduce the transmissions among agents, we have developed the event-triggered MW-MSR algorithm. Compared to conventional methods, our approach does not only achieve the resilient consensus in sparse networks but also has a smaller number of transmissions per node when reaching consensus.

Recently, the fault detection techniques have been studied extensively in the systems control area, which are practical for solving the cyber security issues of multi-agent systems. We are also interested in the resilient consensus algorithms with detection functions. Based on the detection techniques, we have developed two protocols which aim to solve resilient consensus with an emphasis on distributed detection. We have proved necessary and sufficient conditions on graph structures for the two protocols to achieve fault detection in time invariant networks under the  $f$ -total/local malicious model, respectively. In order to achieve such distributed detection, we have introduced the two-hop communication techniques and voting scheme in the detection protocols, which make our protocols novel algorithms compared to conventional one-hop consensus algorithms. The effectiveness of both protocols has been proved by numerical examples.

## 8.2 Future Works

Due to the growth of the applications of network systems, distributed algorithms and cyber-physical systems, more and more devices and systems are connected by the networking services. Yet, a single cyber-attack lunched by adversaries can cause a huge damage to the critical systems, e.g., power grids. It is of vital importance for the network systems to have resilience against cyber-attacks. There are two main approaches to enhance the resilience of multi-agent systems: fault detection based algorithms and resilient algorithms without detection functions. In the following researches, how to make the detection algorithms more distributed and of less computational complexity is a crucial problem.

On the other hand, when centralized systems are converted to distributed systems, one can easily imagine that not all the local systems will play their role properly. Hence, the resilience against faults should be part of the nature of the well-designed distributed algorithms. The research on the various techniques enhancing the resilience of distributed algorithms is a long-standing research topic in the next decades.

For the future works in short terms, the techniques for resilience studied in this thesis can be applied to the security studies of more complex group objectives such as

WSNs, flocking, formation control and economic dispatch problems in smart grids. In the following, we list several directions for future research.

### **Resilient Multi-dimensional Consensus**

So far, our algorithms are all designed for agents having one dimensional states. However, in robotic networks, agents usually have two dimensional or three dimensional states (e.g., locations of robots on a flat surface or locations of drones in the air). One possible direction is to make use of resilient consensus techniques on each dimension of each agent. This method is simple and effective, although it is slightly conservative in the sense that it only guarantees the safety interval of each dimension.

Another possible direction is to apply the detection method to the multi-dimensional consensus problem. That means agents exchange the information sets containing the multi-dimensional states with neighbors. The detection part then can be designed to check if the neighbors are following the given update rule which involves multi-dimensional consensus.

### **Resilient Average Consensus**

In economic dispatch problem and some applications of WSNs, it is desirable for the agents to reach a common average value of their initial states using distributed algorithms. Such algorithms are called average consensus algorithms [13]. Here, the possible direction is to apply the detection method to the resilient average consensus problem, where normal nodes should reach a common average value of their initial states. The running sum ratio consensus algorithm [41] could be the basis for such a successful resilient average consensus algorithm. Because such an algorithm can remove the effects from the detected adversary neighbors if each node can have the true detection information of their neighbors.

### **Resilient Synchronization of Pulse-coupled Oscillators**

The goal of resilient synchronization problem is that the local clocks of the normal

agents must move on together at the end despite malicious attacks. Pulse-based synchronization has attracted much attention in sensor networks and wireless communications [118] because of its simple and identical messages (so-called pulses), i.e., each agent only sends a pulse signal to its neighbors in a period. In this sense, pulse-based synchronization has much less energy consumption compared with conventional packet-based synchronization approaches. Hence, a new type of resilient synchronization of the pulse-coupled oscillators is also an interesting direction to explore.

### **Resilient Consensus under Mobile Adversaries**

The recent work [116] studied the resilient consensus under the mobile malicious model. In such a model, nodes can be infected by adversarial neighbors and become adversarial agents. Moreover, adversary agents can also recover from attacks and become normal agents. This model was inspired by the infection disease that happened in social environment. A more interesting direction could be the analysis of our algorithms under such mobile adversary models.

# References

- [1] W. Abbas, A. Laszka, and X. Koutsoukos. Improving network connectivity and robustness using trusted nodes with application to resilient consensus. *IEEE Transactions on Control of Network Systems*, 5(4):2036–2048, 2018.
- [2] W. Abbas, A. Laszka, and X. Koutsoukos. Diversity and trust to increase structural robustness in networks. In *Proc. American Control Conference*, pages 4043–4048, 2019.
- [3] W. Abbas, M. Shabbir, J. Li, and X. Koutsoukos. Interplay between resilience and accuracy in resilient vector consensus in multi-agent networks. In *Proc. IEEE Conference on Decision and Control*, pages 3127–3132, 2020.
- [4] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *International Conference On Principles Of Distributed Systems*, pages 229–239, 2004.
- [5] T. C. Aysal, M. J. Coates, and M. G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, 56(10):4905–4918, 2008.
- [6] M. H. Azadmanesh and R. M. Kieckhafer. Asynchronous approximate agreement in partially connected networks. *International Journal of Parallel and Distributed Systems and Networks*, 5(1):26–34, 2002.
- [7] G. Bianchin, A. Cenedese, M. Luvisotto, and G. Michieletto. Distributed fault detection in sensor networks via clustering and consensus. In *Proceedings of 54th IEEE Conference on Decision and Control*, pages 3828–3833, 2015.
- [8] R. E. Blahut. *Theory and Practice of Error Control Codes*, volume 126. Addison-Wesley Reading, 1983.

- 
- [9] S. Bonomi, A. Del Pozzo, M. Potop-Butucaru, and S. Tixeuil. Approximate agreement under mobile byzantine faults. *Theoretical Computer Science*, 758:17–29, 2019.
- [10] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. 4th Inter. Conf. Mobile computing netw.*, pages 85–97, 1998.
- [11] F. Bullo, J. Cortés, and S. Martinez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.
- [12] K. Cai and H. Ishii. Quantized consensus and averaging on gossip digraphs. *IEEE Transactions on Automatic Control*, 56(9):2087–2100, 2011.
- [13] K. Cai and H. Ishii. Average consensus on general strongly connected digraphs. *Automatica*, 48(11):2750–2761, 2012.
- [14] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri. Gossip consensus algorithms via quantized communication. *Automatica*, 46(1):70–80, 2010.
- [15] A. Cetinkaya, H. Ishii, and T. Hayakawa. A probabilistic characterization of random and malicious communication failures in multi-hop networked control. *SIAM Journal on Control and Optimization*, 56(5):3320–3350, 2018.
- [16] A. Cetinkaya, H. Ishii, and T. Hayakawa. An overview on denial-of-service attacks in control systems: Attack models and security analyses. *Entropy*, 21(2):210, 2019.
- [17] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267, 1996.
- [18] M. Chen, D. Gündüz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications*, 39(12):3579–3605, 2021.



- 
- [19] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.
- [20] S. M. Dibaji and H. Ishii. Resilient multi-agent consensus with asynchrony and delayed information. *IFAC-PapersOnLine*, 48(22):28–33, 2015.
- [21] S. M. Dibaji and H. Ishii. Resilient consensus of second-order agent networks: Asynchronous update rules with delays. *Automatica*, 81:123–132, 2017.
- [22] S. M. Dibaji, H. Ishii, and R. Tempo. Resilient randomized quantized consensus. *IEEE Transactions on Automatic Control*, 63(8):2508–2522, 2018.
- [23] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2012.
- [24] D. V. Dimarogonas and K. H. Johansson. Stability analysis for multi-agent systems using the incidence matrix: Quantized communication and formation control. *Automatica*, 46(4):695–700, 2010.
- [25] A. D’Innocenzo, F. Smarra, and M. D. Di Benedetto. Resilient stabilization of multi-hop control networks subject to malicious attacks. *Automatica*, 71:1–9, 2016.
- [26] D. Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.
- [27] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM*, 33(3):499–516, 1986.
- [28] X. Dong, B. Yu, Z. Shi, and Y. Zhong. Time-varying formation control for unmanned aerial vehicles: Theories and applications. *IEEE Transactions on Control Systems Technology*, 23(1):340–348, 2014.

- 
- [29] F. Dorfler and F. Bullo. Synchronization and transient stability in power networks and nonuniform kuramoto oscillators. *SIAM Journal on Control and Optimization*, 50(3):1616–1642, 2012.
- [30] M. El Chamie, J. Liu, and T. Başar. Design and analysis of distributed averaging with quantized communication. *IEEE Transactions on Automatic Control*, 61(12):3870–3884, 2016.
- [31] A. Fagiolini, F. Babboni, and A. Bicchi. Dynamic distributed intrusion detection for secure multi-robot systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2723–2728, 2009.
- [32] J. P. Farwell and R. Rohozinski. Stuxnet and the future of cyber war. *Survival*, 53(1):23–40, 2011.
- [33] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.
- [34] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- [35] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sitchitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, pages 151–159, 2003.
- [36] C. Godsil and G. F. Royle. *Algebraic Graph Theory*. Springer, 2013.
- [37] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.
- [38] E. Gravelle and S. Martínez. Quantized distributed load balancing with capacity constraints. In *Proc. 53rd IEEE Conference on Decision and Control*, pages 3866–3871, 2014.
- [39] L. Guerrero-Bonilla, A. Prorok, and V. Kumar. Formations for resilient robot teams. *IEEE Robotics and Automation Letters*, 2(2):841–848, 2017.

- 
- [40] M. Guo, D. V. Dimarogonas, and K. H. Johansson. Distributed real-time fault detection and isolation for cooperative multi-agent systems. In *Proceedings of American Control Conference*, pages 5270–5275, 2012.
- [41] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García. Robust distributed average consensus via exchange of running sums. *IEEE Transactions on Automatic Control*, 61(6):1492–1507, 2015.
- [42] B. Hardekopf, K. Kwiat, and S. Upadhyaya. Secure and fault-tolerant voting in distributed systems. In *Proceedings of IEEE Aerospace Conference*, volume 3, pages 1117–1126, 2001.
- [43] A. Haseltalab and M. Akar. Approximate Byzantine consensus in faulty asynchronous networks. In *Proc. American Control Conference*, pages 1591–1596, 2015.
- [44] J. He, P. Cheng, L. Shi, and J. Chen. SATS: Secure average-consensus-based time synchronization in wireless sensor networks. *IEEE Transactions on Signal Processing*, 61(24):6387–6400, 2013.
- [45] W. P.M.H. Heemels, K. H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *Proc. 51st IEEE Conference on Decision and Control*, pages 3270–3285, 2012.
- [46] M. R. Henzinger, S. Rao, and H. N. Gabow. Computing vertex connectivity: New bounds from old techniques. *Journal of Algorithms*, 34(2):222–250, 2000.
- [47] Y. Iori and H. Ishii. A resilient synchronization protocol for pulse-coupled oscillators over robust networks. In *Proc. American Control Conference*, pages 713–718, 2020.
- [48] Z. Jin and R. M. Murray. Multi-hop relay protocols for fast consensus seeking. In *Proc. IEEE Conf. Dec. Control*, pages 1001–1006, 2006.
- [49] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson. Subgradient methods and consensus algorithms for solving convex optimization problems. In *Proc. 47th IEEE Conference on Decision and Control*, pages 4185–4190, 2008.

- 
- [50] Y. Kadowaki and H. Ishii. Event-based distributed clock synchronization for wireless sensor networks. *IEEE Transactions on Automatic Control*, 60(8):2266–2271, 2015.
- [51] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.
- [52] M. S. Khan, S. S. Naqvi, and N. H. Vaidya. Exact Byzantine consensus on undirected graphs under local broadcast model. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 327–336, 2019.
- [53] M. S. Khan, L. Tseng, and N. H. Vaidya. Exact Byzantine consensus on arbitrary directed graphs under local broadcast model. In *Proc. International Conference on Principles of Distributed Systems*, pages 30:1–30:16, 2020.
- [54] Y. Kikuya, S. M. Dibaji, and H. Ishii. Fault-tolerant clock synchronization over unreliable channels in wireless sensor networks. *IEEE Transactions on Control of Network Systems*, 5(4):1551–1562, 2018.
- [55] J. Lavaei and R. M. Murray. Quantized consensus by means of gossip algorithm. *IEEE Transactions on Automatic Control*, 57(1):19–32, 2011.
- [56] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications*, 31(4):766–781, 2013.
- [57] H. J. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos. Consensus of multi-agent networks in the presence of adversaries using only local information. In *Proceedings of the 1st International Conference on High Confidence Networked Systems*, pages 1–10, 2012.
- [58] T. Li, M. Fu, L. Xie, and J. Zhang. Distributed consensus with limited communication data rate. *IEEE Transactions on Automatic Control*, 56(2):279–292, 2010.

- 
- [59] P. Lin and Y. Jia. Consensus of second-order discrete-time multi-agent systems with nonuniform time-delays and dynamically changing topologies. *Automatica*, 45(9):2154–2158, 2009.
- [60] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated Byzantine agreement. *Journal of the ACM (JACM)*, 53(6):881–917, 2006.
- [61] R. Lu, X. Lin, H. Zhu, X. Liang, and X. Shen. Becan: A bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(1):32–43, 2012.
- [62] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [63] S. Manfredi. Design of a multi-hop dynamic consensus algorithm over wireless sensor networks. *Control Engineering Practice*, 21(4):381–394, 2013.
- [64] J. P. Martin and L. Alvisi. Fast Byzantine consensus. *IEEE Transactions on Dependable and Secure Computing*, 3(3):202–215, 2006.
- [65] H. Mendes, M. Herlihy, N. H. Vaidya, and V. K. Garg. Multidimensional agreement in Byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.
- [66] M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks*, volume 33. Princeton University Press, 2010.
- [67] N. Meskin and K. Khorasani. Actuator fault detection and isolation for a network of unmanned vehicles. *IEEE Transactions on Automatic Control*, 54(4):835–840, 2009.
- [68] R. K. Mishra and H. Ishii. Event-triggered control for discrete-time multi-agent average consensus. *International Journal of Robust and Nonlinear Control*, 2022, to appear.
- [69] A. Mitra, W. Abbas, and S. Sundaram. On the impact of trusted nodes in resilient distributed state estimation of LTI systems. In *Proc. IEEE Conference on Decision and Control*, pages 4547–4552, 2018.

- 
- [70] A. Mitra, F. Ghawash, S. Sundaram, and W. Abbas. On the impacts of redundancy, diversity, and trust in resilient distributed state estimation. *IEEE Transactions on Control of Network Systems*, 8(2):713–724, 2021.
- [71] A. Mitra, H. Hassani, and G. J. Pappas. Online federated learning. In *Proc. 60th IEEE Conference on Decision and Control*, pages 4083–4090, 2021.
- [72] A. Mitra, J. A. Richards, and S. Sundaram. A new approach to distributed hypothesis testing and non-Bayesian learning: Improved learning rate and Byzantine resilience. *IEEE Transactions on Automatic Control*, 66(9):4084–4100, 2021.
- [73] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli. False data injection attacks against state estimation in wireless sensor networks. In *Proc. 49th IEEE Conference on Decision and Control*, pages 5967–5972, 2010.
- [74] H. Nagamochi and T. Ibaraki. *Algorithmic Aspects of Graph Connectivity*. Cambridge University Press, 2008.
- [75] A. Nedić and A. Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2014.
- [76] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.
- [77] A. Nedic, A. Ozdaglar, and P. A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.
- [78] M. Nesterenko and A. Arora. Tolerance to unbounded Byzantine faults. In *Proc. 21st IEEE Symposium on Reliable Distributed Systems*, pages 22–29, 2002.
- [79] Y. Nugraha, A. Cetinkaya, T. Hayakawa, H. Ishii, and Q. Zhu. Dynamic resilient network games with applications to multiagent consensus. *IEEE Transactions on Control of Network Systems*, 8(1):246–259, 2021.
- [80] F. Núñez, Y. Wang, and F. J. Doyle III. Global synchronization of pulse-coupled oscillators interacting on cycle graphs. *Automatica*, 52:202–209, 2015.
- [81] K. Oh, M. Park, and H. Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

- 
- [82] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proc. IEEE*, 95(1):215–233, 2007.
- [83] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.
- [84] B. Parhami. Voting algorithms. *IEEE Transactions on Reliability*, 43(4):617–629, 1994.
- [85] H. Park and S. A. Hutchinson. Fault-tolerant rendezvous of multirobot systems. *IEEE Transactions on Robotics*, 33(3):565–582, 2017.
- [86] F. Pasqualetti, A. Bicchi, and F. Bullo. Consensus computation in unreliable networks: A system theoretic approach. *IEEE Transactions on Automatic Control*, 57(1):90–104, 2012.
- [87] F. Pasqualetti, F. Dörfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729, 2013.
- [88] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.
- [89] W. Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.
- [90] W. Ren and Y. Cao. *Distributed Coordination of Multi-agent Networks: Emergent Problems, Models, and Issues*. Springer, 2011.
- [91] X. Ren and Y. Mo. Secure detection: Performance metric and sensor deployment strategy. *IEEE Transactions on Signal Processing*, 66(17):4450–4460, 2018.
- [92] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [93] D. Sakavalas, L. Tseng, and N. H. Vaidya. Effects of topology knowledge and relay depth on asynchronous consensus. *arXiv preprint arXiv:1803.04513*, 2018.

- 
- [94] D. Sakavalas, L. Tseng, and N. H. Vaidya. Asynchronous byzantine approximate consensus in directed networks. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 149–158, 2020.
- [95] D. Saldana, A. Prorok, S. Sundaram, M. F.M. Campos, and V. Kumar. Resilient consensus for time-varying networks of dynamic agents. In *Proc. American Control Conference*, pages 252–258, 2017.
- [96] L. Schenato and F. Fiorentin. Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, 47(9):1878–1886, 2011.
- [97] D. M. Senejohnny, S. Sundaram, C. De Persis, and P. Tesi. Resilience against misbehaving nodes in asynchronous networks. *Automatica*, 104:26–33, 2019.
- [98] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson. Event-based broadcasting for multi-agent average consensus. *Automatica*, 49(1):245–252, 2013.
- [99] I. Shames, A. M. H. Teixeira, H. Sandberg, and K. H. Johansson. Distributed fault detection for interconnected second-order systems. *Automatica*, 47(12):2757–2764, 2011.
- [100] D. Silvestre, P. Rosa, J. Hespanha, and C. Silvestre. Stochastic and deterministic fault detection for randomized gossip algorithms. *Automatica*, 78:46–60, 2017.
- [101] A. Srinivasan, J. Teitelbaum, and J. Wu. DRBTS: Distributed reputation-based beacon trust system. In *Proceedings of 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 277–283, 2006.
- [102] L. Su and N. H. Vaidya. Reaching approximate Byzantine consensus with multi-hop communication. *Information and Computation*, 255:352–368, 2017.
- [103] L. Su and N. H. Vaidya. Defending non-Bayesian learning against adversarial attacks. *Distributed Computing*, 32(4):277–289, 2019.
- [104] L. Su and N. H. Vaidya. Byzantine-resilient multiagent optimization. *IEEE Transactions on Automatic Control*, 66(5):2227–2233, 2020.



- 
- [105] S. Sundaram and B. Gharesifard. Distributed optimization under adversarial nodes. *IEEE Transactions on Automatic Control*, 64(3):1063–1076, 2018.
- [106] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Transactions on Automatic Control*, 56(7):1495–1508, 2011.
- [107] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson. Attack models and scenarios for networked control systems. In *Proc. 1st International Conference on High Confidence Networked Systems*, pages 55–64, 2012.
- [108] S. Tixeuil. Self-stabilizing Algorithms. *Algorithms and Theory of Computation Handbook*, pages 26.1–26.45, 2009.
- [109] L. Tseng and N. H. Vaidya. Fault-tolerant consensus in directed graphs. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 451–460, 2015.
- [110] L. Tseng, N. H. Vaidya, and V. Bhandari. Broadcast using certified propagation algorithm in presence of Byzantine faults. *Information Processing Letters*, 115(4):512–514, 2015.
- [111] J. Usevitch and D. Panagou.  $r$ -robustness and  $(r, s)$ -robustness of circulant graphs. In *Proc. IEEE Conf. Dec. Control*, pages 4416–4421, 2017.
- [112] J. Usevitch and D. Panagou. Determining  $r$ - and  $(r, s)$ -robustness of digraphs using mixed integer linear programming. *Automatica*, 111:108586, 2020.
- [113] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate Byzantine consensus in arbitrary directed graphs. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, pages 365–374, 2012.
- [114] Y. Wang and H. Ishii. Resilient consensus through event-based communication. *IEEE Transactions on Control of Network Systems*, 7(1):471–482, 2019.

- 
- [115] Y. Wang and H. Ishii. An event-triggered approach to quantized resilient consensus. *International Journal of Robust and Nonlinear Control*, 30(11):4188–4204, 2020.
- [116] Y. Wang, H. Ishii, F. Bonnet, and X. Défago. Resilient real-valued consensus in spite of mobile malicious agents on directed graphs. *IEEE Transactions on Parallel and Distributed Systems*, 33(3):586–603, 2021.
- [117] Y. Wang and R. Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 178–188, 2020.
- [118] Z. Wang and Y. Wang. Global synchronization of pulse-coupled oscillator networks under Byzantine attacks. *IEEE Transactions on Signal Processing*, 68:3158–3168, 2020.
- [119] D. B. West. *Introduction to Graph Theory*. 2nd ed. Prentice Hall, Upper Saddle River, 2001.
- [120] F. Xiao and L. Wang. State consensus for multi-agent systems with switching topologies and time-varying delays. *International Journal of Control*, 79(10):1277–1284, 2006.
- [121] F. Xiao and L. Wang. Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays. *IEEE Transactions on Automatic Control*, 53(8):1804–1816, 2008.
- [122] J. Yan, X. Li, Y. Mo, and C. Wen. Resilient multi-dimensional consensus in adversarial environment. *arXiv preprint arXiv:2001.00937*, 2020.
- [123] S. Yang, S. Tan, and J. Xu. Consensus based approach for economic dispatch problem in a smart grid. *IEEE Transactions on Power Systems*, 28(4):4416–4426, 2013.
- [124] L. Yuan and H. Ishii. Resilient consensus with distributed fault detection. In *Proceedings of 8th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, pages 285–290, 2019.

- 
- [125] L. Yuan and H. Ishii. Resilient consensus with multi-hop communication. In *Proc. IEEE Conference on Decision and Control*, pages 2696–2701, 2021. Also, arXiv preprint, arXiv:2201.03214, 2022.
- [126] L. Yuan and H. Ishii. Asynchronous approximate Byzantine consensus via multi-hop communication. In *Proc. American Control Conference*, pages 755–760, 2022.
- [127] L. Yuan and H. Ishii. Secure consensus with distributed detection via two-hop communication. *Automatica*, 131, no. 109775, 2021.
- [128] H. Zhang, E. Fata, and S. Sundaram. A notion of robustness in complex networks. *IEEE Transactions on Control of Network Systems*, 2(3):310–320, 2015.
- [129] W. Zhang, G. Xue, and S. Misra. Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms. In *Proc. IEEE 26th IEEE International Conference on Computer Communications*, pages 1649–1657, 2007.
- [130] C. Zhao, J. He, and J. Chen. Resilient consensus with mobile detectors against malicious attacks. *IEEE Transactions on Signal and Information Processing over Networks*, 4(1):60–69, 2018.
- [131] C. Zhao, J. He, P. Cheng, and J. Chen. Secure consensus against message manipulation attacks in synchronous networks. *IFAC Proceedings Volumes*, 47(3):1182–1187, 2014.
- [132] C. Zhao, J. He, P. Cheng, and J. Chen. Consensus-based energy management in smart grid with transmission losses and directed communication. *IEEE Transactions on Smart Grid*, 8(5):2049–2061, 2017.
- [133] C. Zhao, J. He, and Q. Wang. Resilient distributed optimization algorithm against adversarial attacks. *IEEE Transactions on Automatic Control*, 65(10):4308–4315, 2019.
- [134] Z. Zhao and Z. Lin. Global leader-following consensus of a group of general linear systems using bounded controls. *Automatica*, 68:294–304, 2016.
- [135] M. Zhu and S. Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.