NOVEL FAULT-TOLERANT DISTRIBUTED ALGORITHMS FOR APPROXIMATE BYZANTINE CONSENSUS

by

Ali Haseltalab

B.S., Electrical-Control Engineering, University of Tabriz, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2014

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help of so many people in so many ways. I would like to express my deepest appreciation to my supervisor Assoc. Prof. Mehmet Akar who continually and convincingly conveyed the spirit of adventure in regard to science to me. His patience, support and advice throughout my study for Master of Science degree were outstanding and memorable.

I would like to thank Prof. Yağmur Denizhan for her invaluable guidance throughout my study and serving as a member of my thesis jury. Furthermore, I thankfully appreciate Prof. Ufuk Çağlayan for doing me a favor by taking part in my thesis jury.

I am grateful to my dear colleagues in the Networked and Embedded Control Systems Laboratory, Özlem Feyza Erkan, Onur Cihan, Huzeyfe Esen and Kamil Şenel, for their ultimate support and cooperation throughout the stages of my study. Moreover, I would like to thank my other colleagues and friends especially the Iranian students community at Boğaziçi University for helping me survive all the stress during this study and not letting me give up.

There is no word to describe the priceless and endless support of my family, not only in the last two years, but throughout every single stage of life. I would like to dedicate my sincerest gratitude to my parents Hassan Haseltalab and Faride Ghozat and my brother Omid Haseltalab for pushing me forward in my studies.

Last but not least, as an international student, I am indebted to people of Turkey as a solid and strong nation with a notable background. Their hospitality as well as kindness and support conducted the warm sense of home to me during this era.

This thesis is partially supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant 110E196.

ABSTRACT

NOVEL FAULT-TOLERANT DISTRIBUTED ALGORITHMS FOR APPROXIMATE BYZANTINE CONSENSUS

Reaching consensus in a network which contains faulty nodes is a critical problem in the field of distributed and multi-agent systems. A distributed system which intends to do a certain task needs to display robustness against adverse behavior of some of its faulty nodes, known also as Byzantine nodes. In this thesis, a novel Mean-Select-Reduced (MSR) fault tolerant algorithm is proposed for achieving Approximate Byzantine Consensus. It is shown that the topological condition required for the success of the algorithm is more relaxed compared to the previous results. In contrary to results that appeared in the literature, it is proved that synchronicity of networks and presence of delay on communication paths do not change this condition. Subsequently, the convergence rate and time analysis for the proposed fault-tolerant algorithm is carried out and the results are extended to time-varying networks. In most of the fault-tolerant algorithms that have been introduced for Byzantine networks, it is assumed that each node has knowledge of the maximum number of faulty nodes, f_t, in the network. In this thesis, we also propose a new family of algorithms which do not require this a priori information and evaluate their performance facing Byzantine failures.

ÖZET

YAKLAŞIK BİZANS ONAYLAŞIMI İÇİN HATA TOLERANSLI DAĞITIK ÖZGÜN ALGORİTMALAR

Dağıtık ve çok etmenli sistemler alanında, hatalı düğümler içeren bir ağın onaylaşımı sağlaması önemli bir problemdir. Belirli bir görevi yerine getirmesi istenen dağıtık bir sistem, bazı hatalı düğümlerinin olumsuz davranışlarına karşı gürbüzlük göstermelidir. Bu hatalı düğümler aynı zamanda Bizans düğümleri olarak da bilinmektedir. Bu tezde, Yaklaşık Bizans onaylaşımını sağlamak için Ortalama-Seçimli-İndirgenmiş (OSİ) hata toleranslı özgün bir algoritma önerilmiştir. Algoritmanın başarımı için gerekli olan ilinge koşulunun, önceki sonuçlarla kıyaslandığında, gevşetildiği gösterilmiştir. Literatürde yer alan sonuçların aksine, ağların senkronizasyonunun ve iletişim kanallarındaki gecikmenin varlığının bu koşulu değiştirmediği ispatlanmıştır. Daha sonra, önerilen hata toleranslı algoritma için yakınsama hızı ve zamanı analizi gerçekleştirilmiş ve sonuçlar zamanla değişen ağlara genişletilmiştir. Bizans ağları için sunulan hata toleranslı algoritmaların birçoğunda, ağdaki her bir düğümün, hatalı düğümlerin maksimum sayı, f_t, bilgisine sahip olduğu kabulü yapılmaktadır. Ayrıca bu tezde, bu ön bilgiyi gerektirmeyen yeni bir algoritma ailesi önerilmiş ve Bizans hatalarının varlığında performansları değerlendirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	The adjacency matrix of graph G
a_{ij}	Elements of the adjacency matrix of a graph
C	The semi-transition matrix of a network
c_{ij}	Elements of matrix C
E	The set of edges of graph G
D_i	The set of eliminated values by node i in the MSR algorithm
e	The maximum agreement error
F	The set of faulty nodes in a network
f	The maximum number of faulty nodes in a network
f_t	The number of faulty nodes in a network
f_a	The elimination parameter of the MSR algorithm
G	The underlying directed graph of a network
H_{χ_i}	The higher bound for convex hull of node i
i_{re}	The indice of reachability
i_{ro}	The indice of robustness
L_{χ_i}	The lower bound for convex hull of node i
M	The maximum state value of the set of non-faulty nodes
m	The minimum state value of the set of non-faulty nodes
N_i	The set of neighbors of node i
N_i'	The set of acceptable state values received by i
N_{χ_i}	The set of nodes with convex hulls acceptable for node i
n	Total number of nodes in a graph
$\mathbb{R}$	The set of real values
r	The number of update round
t	The index of time
t_r^i	The update time of node i at round r
t_{M_r}	The maximum sequence of time in a network
t_{m_r}	The minimum sequence of time in a network

V	The set of vertices of graph G
W	The transition matrix of a network
x	The vector of state values
x_i	The state value of node i
x_{ij}	The state value of node j received by node i
α	The minimum of ω_{ij} weights
α_t	The maximum update time interval of non-faulty nodes in a network
α_τ	The maximum communication delay between two non-faulty nodes in a network
β_t	The minimum update time interval of non-faulty nodes in a network
$\gamma(W)$	The ergodicity value of transition matrix W
$\gamma'(W)$	The ergodicity value of the network of non-faulty nodes of transition matrix W
Δt_r^i	The time interval between two successive update in state of node i
Δt_M	The sum of maximum update time interval and maximum communication delay in a network
ϵ	A chosen consensus accuracy level
κ_M	The subset of non-faulty nodes in a network with state value M
κ_m	The subset of non-faulty nodes in a network with state value m
ξ	The set of semi transition matrices of a time-varying faulty network
τ_{ij}	The communication delay between two nodes j and i
τ_e	The maximum time for decrease of maximum agreement error
χ_i	The convex hull of node i
ω_{ij}	The averaging coefficients

LIST OF ACRONYMS/ABBREVIATIONS

LCA	Linear Consensus Algorithm
MSR	Mean-Select-Reduced
SCH	Shrinking Convex Hulls
SCH-CBN	Shrinking Convex Hulls for Complete Byzantine Networks
SCH-CMN	Shrinking Convex Hulls for Complete Malicious Networks
W-MSR	Weighted Mean-Select-Reduced

1. INTRODUCTION

In a distributed system in which a common task is intended to be accomplished, the individual nodes of the system are required to reach *Consensus* on one or more common values. The problem of reaching consensus in a network has received considerable attention recently. It is employed in many fields such as collective behavior of flocks and swarms [1, 2], formation control [3–5] and control of distributed systems [6–8], etc.

Consider a set of autonomous robots where each has an initial position and they want to get a formation to monitor an area. Let the desired formation be a square and the number of robots be four. These robots also need to maintain certain distances from each other while moving in different directions. As result, by exchanging information, they need to reach consensus on the size of square. Moreover, since they want to move, they need to achieve consensus on the speed and the direction of the movement so that the formation is always held [9].

One of the most popular distributed algorithms for achieving consensus is the linear consensus algorithm [10,12]. In this algorithm, each node receives the state value of its neighbor nodes in each iteration. Then, it takes a weighted-mean to update its state using its own current state value and the received state values. It is proved that the state values of all nodes converge to a common value if and only if the underlying graph of the network contains a rooted out branching [10–12]. Moreover, this problem has been well considered in networks with time-varying underlying graph [13, 14].

In real-life applications, there might exist some nodes that represent adverse behaviors due to some damages that they encountered. Moreover, a network may face an intruder node which tries to prevent the other nodes reaching consensus. Existence of these faulty nodes necessitates the adoption of fault-tolerant algorithms which allow non-faulty nodes to achieve consensus on a common value.

Reaching consensus in a network which contains faulty nodes is a critical problem

in the field of distributed and multi-agent systems. A distributed system which intends to do a certain task needs to display robustness against adverse behavior of some of its faulty nodes, known also as Byzantine nodes. Byzantine failure is intended to model any arbitrary malfunction of network components [15]. This problem was first introduced in [16] in terms of Byzantine generals who need to reach agreement on a common action while some of them are traitors and attempt to prevent the agreement. In [16, 17], it is shown that using the suggested algorithm and decision making rules, reaching consensus is guaranteed if and only if, $n > 3f_t$ where n is the total number of generals and f_t is the number of traitor generals. Subsequently, in [15, 17], it is proved that there exists no algorithm that can tolerate f_t number of faulty nodes if $n \leq 3f_t$. In [18], the authors presented a fault-tolerant algorithm for asynchronous networks and proved its correctness facing f_t number of Byzantine nodes where $f_t \leq \frac{n-1}{3}$. It might worth mentioning that more restrictive types of fault are considered in the literature such as stopping [15, 19], non-colluding [20], Malicious [20, 21].

Dolev *et al.* [22] presented the notion of Approximate Byzantine Consensus for averaging-based consensus algorithms. The nature of such algorithms [10, 12] imposes such an approximation, since it is not guaranteed that state values of nodes become exactly equal after a certain time. In [22], a family of algorithms called Mean-Subsequence (Select)-Reduced (MSR) is proposed for fault-tolerance. Later these algorithms were applied to partially connected networks in [23–25]. MSR algorithms are still considered as the main solution for approximate Byzantine consensus in faulty networks. Recently, Vaidya *et al.* [26] after stating the necessary condition which needs to be satisfied by the underlying graph so that an iterative algorithm can achieve approximate consensus, they proved the correctness of an MSR algorithm in such a graph. Furthermore, an alternate proof for the correctness of their MSR algorithm using matrix representation was provided in [27]. In [28, 29], the authors considered a more restrictive type of fault and proved the necessary and sufficient conditions for achieving consensus. Also, in [29] the authors proposed a variation of the MSR algorithm for Byzantine fault and proved its success using the notion of robustness in graphs which we adopt in this thesis.

The problem of approximate Byzantine consensus is well considered in both syn-

chronous and asynchronous networks. Consensus in asynchronous networks is a more realistic case facing practical problems, since agents (nodes) do not update their state at the same time in real environments. Moreover, delay over communication paths (graph edges) is inherited in many systems. The problem of approximate Byzantine consensus in asynchronous networks is considered in [22, 25, 26]. Dolev *et al.* [22], proposed an MSR algorithm for asynchronous networks with complete underlying graph topology. The problem in partially connected networks is assessed in [25] where topological necessary and sufficient conditions for approximate Byzantine Consensus are presented. These conditions are presented in [26] as well. In all of the mentioned works, the restriction on the topology of underlying graph of networks is the same. Comparing to synchronous networks, it is shown that the number of non-faulty nodes and communication paths between them should be more in asynchronous networks so that approximate Byzantine consensus can be achieved.

Most of the mentioned algorithms assume that each node has an initial (and true) knowledge on the maximum number of faulty nodes (f_t) in the network. This is a huge initial undistributed assumption and if f_t is not chosen precisely, probability of failure is extremely high. For example, in an MSR algorithm as much as f_t is considered greater probability of consensus in an arbitrary graph is less. On the other hand, if we consider a small f_t then it is more probable that number of Byzantine nodes pass from f_t. Since elimination of up to $2f_t$ of received values to each node is key to the performance of these algorithms, it is very common that nodes of a network can not reach consensus in absence of any faulty node because a node may disregard all of the received state values. For different examples, please refer to [29, 30].

In this thesis, we first clarify the notion of Byzantine fault. A more restrictive but common type of fault known as Malicious fault is also introduced. Furthermore, we explain the notion of approximate Byzantine consensus and describe why its presence is necessary. The concept of robustness in graphs is discussed and utilized in developing our contributions. Then, we introduce a novel MSR algorithm and we show many advantages of this algorithm compared to other MSR algorithms that appeared in the literature. Convergence analysis of the proposed MSR algorithm is carried out to

justify its success in synchronous and asynchronous networks. Moreover, the case of networks with delay on communication paths (underlying graph edges) is considered and the success of the proposed algorithm is proved. Convergence rate and time analysis of the MSR algorithm is done and a higher bound for convergence time is derived. Finally, a new family of algorithms is introduced and their success in networks with complete underlying graph topology is proved. These algorithms do not employ a priori knowledge on maximum number of faulty nodes. In each chapter, numerical analysis is included, not only for clarifying the problem that we are dealing with, but also, for evaluating our findings.

1.1. Contributions of the Thesis

The contributions of this thesis are as follows:

- A novel MSR algorithm that requires more relaxed topological condition for its success compared to other algorithms in the literature is introduced. It is shown that synchronicity of networks and presence of delay on communication paths do not affect this condition which is in contrary to the results that appeared previously. Moreover, it is justified that the algorithm succeeds in time-varying networks.
- Convergence rate and time analysis of the proposed MSR algorithm is carried out. Although convergence rate and time analysis is studied extensively for consensus algorithms, the problem is not considered in detail for fault-tolerant algorithms. We determine upper bounds for the convergence rate of the algorithm. Then, we show that some properties of linear consensus algorithm in non-faulty networks are valid in a faulty network that adopts our proposed MSR algorithm. Using these properties and matrix representation, we derive more optimized bounds for the convergence rate and time.
- The above results are also extended to the case of time-varying networks. Furthermore, the existence of a Lyapunov function for time-varying networks is shown.
- A new family of algorithms which do not use a priori knowledge on maximum number of faulty nodes is proposed. Two algorithms are introduced for Malicious

and Byzantine failures in complete networks and their proof of success along with their numerical analysis are presented.

1.2. Organization of the Thesis

The thesis is organized as follows. In Chapter 2, the notions of Byzantine fault, approximate Byzantine consensus and robustness are defined. Moreover, well-known averaging-based consensus algorithms are presented. In Chapter 3, our proposed MSR algorithm is presented and its convergence analysis is carried out. In Chapter 4, convergence rate and time of the proposed MSR algorithm is derived and the existence of a Lyapunov function is assessed. In Chapter 5, a new family of algorithms called *Shrinking Convex Hull* algorithms is proposed. Finally, some concluding remarks are provided in Chapter 6.

2. APPROXIMATE BYZANTINE CONSENSUS: DEFINITIONS AND MATHEMATICAL PRELIMINARIES

In this chapter, the notions of Byzantine node, approximate Byzantine consensus and robustness are explained. Furthermore, the Linear consensus protocol is introduced and it is shown that it can not resist against adversarial behavior of faulty nodes. Two fault-tolerant algorithms from literature are presented and the necessary and sufficient topological conditions required for their success is discussed and evaluated.

2.1. Linear Consensus Algorithm

Consider a network of nodes which can be represented by a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Suppose each node i communicates with its neighbors by sending its state value $x_i(t)$ and receiving the state value of its neighbor node j where $x_{ij}(t)$ is the value that node i receives from node j. The objective of exact consensus is that all nodes reach agreement on a common value, i.e., $x_i(t) = x_j(t)$, for all $t \geq t_0$ and some $t_0 \geq 0$, for all i, $j \in V$ and any initial condition $x_i(0)$, $i \in V$. In many applications, exact consensus is impossible and as result, approximate consensus rather than exact consensus is the objective. Therefore, the goal is developing distributed algorithms so that as $t \to \infty$, state values of nodes converge asymptotically to a common value, i.e., $|x_i(t) - x_j(t)| \leq \epsilon$ for all i,$j \in V$ and any initial condition $x_i(0)$, $i \in V$ where $\epsilon > 0$ is a pre-assigned value. In the following, we describe a well-known algorithm for reaching approximate consensus.

Among diverse approaches to achieve approximate consensus in a network, an averaging-based algorithm which is known as Linear Consensus algorithm has attracted considerable interest due to its applicability in a variety of contexts. In this algorithm, each node receives information from its neighbors at time t and changes its value

according to

$$x_i(t+1) = \sum_{j \in N_i(t)} \omega_{ij}(t) x_{ij}(t) \tag{2.1}$$

where $\omega_{ij}(t)$ is the weight assigned to node j's value by node i at time-step t. For the algorithm, the following conditions on $\omega_{ij}(t)$ are assumed to hold.

- $\omega_{ij}(t) = 0$ whenever $j \notin N_i(t)$.
- $0 < \alpha \leq \omega_{ij}(t) \leq 1$, $\forall j \in N_i(t)$.
- $\sum_{j=1}^{n} \omega_{ij}(t) = 1$, $\forall i \in V$.

where $N_i(t)$ is the set of node i's neighbors at time t. We assume that $N_i(t)$ contains its own index i.

Theorem 2.1. *[5] In a time-invariant network in which the nodes update their state values using the linear consensus algorithm, approximate consensus is guaranteed if and only if the underlying graph of the network has a rooted out-branching.*

A rooted out-branching is a directed path that connects all the nodes. The above theorem presents the necessary and sufficient condition required for the success of the linear consensus algorithm in time-invariant networks. The sufficient condition for success of the algorithm in time-varying networks is explained in the following theorem.

Theorem 2.2. *[13] In a time-varying network, in which the nodes update their state values using the linear consensus algorithm, approximate consensus is guaranteed if there exist infinite bounded sequences that the union of graphs in each sequence has a rooted out branching. Furthermore, if the union of the graphs after some finite time does not have a rooted out branching, then consensus cannot be achieved asymptotically.*

2.2. Byzantine Fault

A Byzantine fault models any arbitrary malfunction of faulty nodes such as, stopping, crashing, sending different values to different neighbor nodes or taking any

arbitrary function instead of averaging [15]. In the literature, various types of faults are introduced that are more restrictive than the Byzantine fault. However, the definition of Byzantine node encompasses all of these types. In this thesis, we consider a Byzantine node as a faulty node that can not follow the exact steps of the pre-assigned algorithm that all non-faulty nodes of a network are using.

Definition 2.1 (Byzantine node)**.** *[29] A node $i \in V$ is called Byzantine if it does not send the same value through all its outgoing edges at the same time, i.e., $x_{ji}(t) \neq x_{ki}(t)$, $i \in N_j(t) \cap N_k(t)$ or performs any arbitrary function instead of the averaging function in (2.1).*

We believe that the above definition embraces all adverse behaviors that a faulty node may represent. Therefore, a node is faulty if and only if it is Byzantine. Throughout this thesis, the set of Byzantine nodes is shown with F.

Definition 2.2 (Malicious node)**.** *[29] A node $i \in V$ is said to be Malicious (one-sided Byzantine) if it sends the same value to all of its neighbors at a time step, i.e., $x_{ji}(t) = x_{ki}(t)$, $i \in N_j(t) \cap N_k(t)$ but it applies another arbitrary function $f_i'(t)$ to update its state instead of (2.1), i.e., $x_i(t+1) = f_i'(t)$.*

This type of fault is prevalent in distributed systems where each agent acquires the state of its neighbors by observation. In other words, agents do not send their state values to each other. In this type of networks, Byzantine nodes can not send different values to different agents at the same time and as result, Byzantine fault is reduced to Malicious fault. For instance, consider a flock of birds. Each agent (bird) obtains information about the position of its neighbors by observation. As result, there exists no two non-faulty agents which obtain different information about an agent at the same time.

Malicious failure is a more restrictive type of fault compared to Byzantine failure. As result, a malicious node is Byzantine but vice versa is not valid [29].

2.3. Exact and Approximate Byzantine Consensus

In this section, we define the Exact and Approximate Byzantine Consensus concepts.

Definition 2.3 (Exact Byzantine Consensus)**.** *[15]: A network is said to reach Exact Byzantine Consensus if each one of its nodes starts from an arbitrary value and all of its non-faulty nodes reach consensus with the following conditions:*

- ***Agreement:*** *No two non-faulty nodes decide on different values, i.e., there exists a finite t_0 such that we have $x_i(t) = x_j(t)$ for all i, $j \in V \setminus F$, and for all $t \geq t_0$ and $x_i(0)$.*
- ***Validity:*** *If all non-faulty nodes start with the same initial value x_0, then x_0 is the only possible decision value for a non-faulty node.*

Distributed consensus using averaging protocols has received considerable attention recently. For a review on related literature please refer to [12]. The averaging nature of these algorithms necessitates presence of approximate Byzantine consensus in which approximate rather than exact agreement is the desired goal for non-faulty nodes. In networks whose nodes use averaging protocols to reach consensus, it is not guaranteed that all nodes reach consensus on *exactly* the same value after a certain time.

Definition 2.4 (Approximate Byzantine Consensus)**.** *[22]: A network is said to reach Approximate Byzantine Consensus if its nodes start from an arbitrary value $x_i(0)$ and all of its non-faulty nodes reach consensus with the following two conditions:*

- ***Agreement:*** *For any pre-assigned $\epsilon > 0$, all non-faulty nodes eventually decide on output values that are within ϵ of each other, i.e., there exists a finite t_0 such that we have $|x_i(t) - x_j(t)| \leq \epsilon$ for all i, $j \in V \setminus F$, and for all $t \geq t_0$ and $x_i(0)$.*
- ***Validity:*** *The output value of each non-faulty node must be in the range of initial values of the non-faulty nodes.*

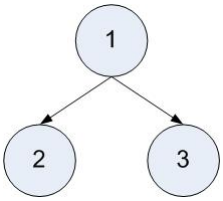


Figure 2.1. A network with three nodes where state values of Nodes 2 and 3 converge asymptotically to state value of Node 1.

Next, an example is presented to show that exact consensus is not guaranteed in networks whose nodes adopt averaging-based algorithms to achieve consensus.

Example 2.1. Consider the network depicted in Figure 2.1. At each iteration, Node 1 sends its state value to Nodes 2 and 3 which update their state values by taking the average of the received values and their own state values. Assume that the initial state value of Node 1 is 2 and the initial state values of Nodes 2 and 3 are 1. It can be inferred that there exists no time, after which state values of all nodes become exactly equal. However, the state values of Nodes 2 and 3 converge asymptotically to the state value of Node 1.

The next example, shows the effect of a Byzantine node in a network where nodes use the linear consensus algorithm to update their states.

Example 2.2. Consider a network with 4 nodes. Let the underlying graph of the network be complete (i.e., all nodes are connected to each other). First, consider the case when none of the nodes is Byzantine. As depicted in Figure 2.2, it is seen that all nodes achieve consensus. Now assume that Node 2 is Byzantine and at each iteration sends arbitrary state values to its neighbor nodes. Figure 2.3 illustrates the simulation results of this network. The state values of Nodes 1, 3 and 4 are depicted with respect to time and it can be inferred that these values are not converging to a common value.

As Example 2.2 suggests, the linear consensus algorithm is not capable of resisting against adversarial behavior of Byzantine nodes. Therefore, a Byzantine node can easily prevent the network from reaching agreement. This major problem necessitates

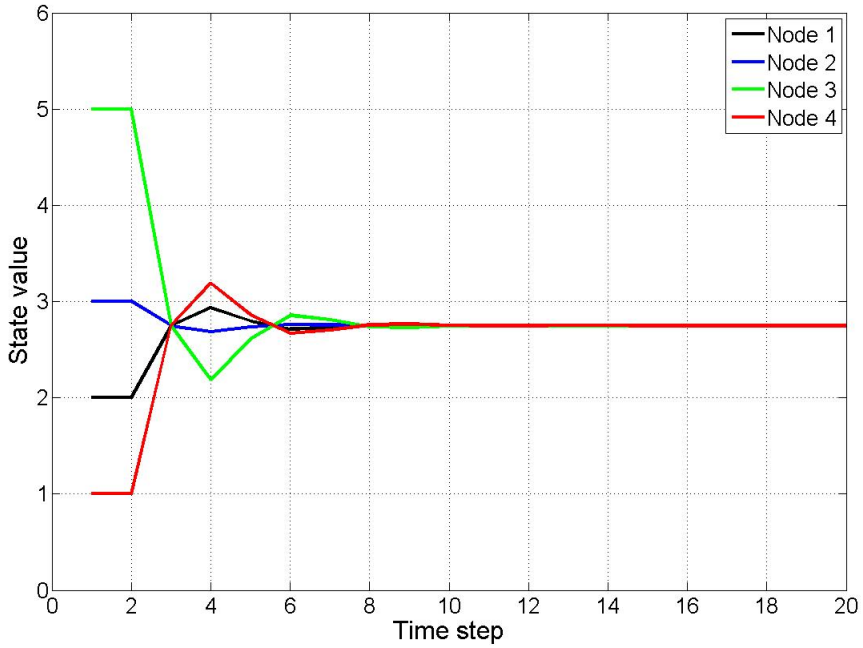


Figure 2.2. Simulation result of a network which contains no Byzantine node and uses the linear consensus algorithm.

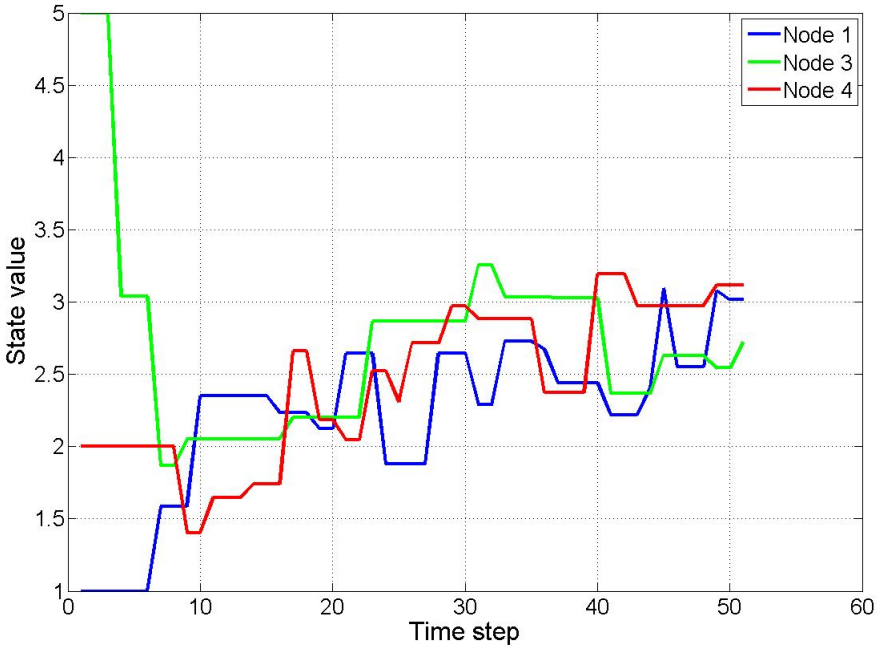


Figure 2.3. Simulation result of a network which contains a Byzantine node and uses the linear consensus algorithm.

adoption of averaging-based fault-tolerant algorithms. Hence, the objective is to design a distributed algorithm so the network achieves approximate Byzantine consensus in presence of Byzantine nodes.

2.4. Robustness of Graphs

In this part, the robustness of the underlying graph of consensus networks is discussed. Let the underlying directed graph of the network be denoted by $G = (V, E)$ where V is the set of vertices with $|V| = n$ ($|V|$ denotes the cardinality of V) and E is the set of edges. Furthermore, the set of neighbors of node i is denoted by N_i.

Definition 2.5 (Reachability)**.** *[29] Given a digraph $G = (V, E)$ and a nonempty subset S of nodes ($S \subset V$) of G, we say that S is an r-reachable subgraph if there exists an $i \in S$ such that $|N_i \setminus S| \geq r$, where $r \in \mathbb{N}$.*

Let A be the adjacency matrix of the digraph G and A_S be a matrix with the same dimensions as A which contains the adjacency matrix of S with the condition that if i or $j \notin S$ then $a_{S_{ij}} = 0$, where $a_{S_{ij}}$ is an element of A_S at row i and column j. We refer to this matrix as adjacency matrix of S in G environment. With this definition, the reachability of the subgraph S can be determined as:

$$i_{re}(G, S) = \max_{i \in S} \left(\sum_{j=1}^{n} (a_{ij} - a_{S_{ij}}) \right) \tag{2.2}$$

Definition 2.6 (Robustness)**.** *[29] A digraph $G = (V, E)$ ($n \geq 2$) is said to be r-robust with $r \in \mathbb{N}$, if for every pair of nonempty and disjoint subsets of V, at least one of the subsets is r-reachable.*

Similar to the previous approach, if S and U are two subgraphs of G and A_S and A_U are their adjacency matrices in G environment respectively, then the robustness of

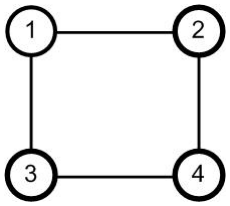


Figure 2.4. A graph with four nodes.

graph G can be calculated as below:

$$i_{ro}(G) = \min_{S,U \subset V} \left(\max_{i \in S \cup U} \left(\sum_{j=1}^{n} (a_{ij} - a_{S_{ij}} - a_{U_{ij}}) \right) \right) \tag{2.3}$$

The notion of robustness is useful in understanding how the total information generated by nodes is accessible for an individual node. The more a node is connected with subgraphs of the underlying graph of its network, the more reliable it can make decisions and as result, the total network behavior is more robust facing failures.

In the following, we present an example regarding the notion of robustness.

Example 2.3. Consider the graph depicted in Figure 2.4. The adjacency matrix of the graph A is as below:

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

Consider a subgraph of the depicted graph which consists of Nodes 1 and 2. Since each node in this subgraph has at most one neighbor out of the subgraph, the reachability of the subgraph is 1. Therefore, we say the subgraph is 1-reachable. The fact can be verified using (2.2). Moreover, from (2.3), the depicted graph is 1-robust because for every two disjoint subgraphs at least one of them is 1-reachable.

2.5. Fault-Tolerant Distributed Consensus Algorithms

In this section, we present two fault-tolerant algorithms from the literature [22,26, 29]. These algorithms are designed to resist against adversarial behavior of Byzantine nodes. We also enumerate their shortcomings.

2.5.1. Weighted Mean-Select-Reduced Algorithm

This fault-tolerant algorithm has been proposed recently by Leblanc *et al.* [29] for synchronous networks without delay on communication paths. At each time-step t of the W-MSR algorithm, each normal node i obtains the values of its neighbor nodes. If the set of faulty nodes in the network is represented by F with $|F| = f_t$, at most f_t neighbor nodes of node i may be Byzantine; however, node i is unsure of which neighbors may be faulty. To ensure that node i updates its value in a safe manner, each node removes the extreme values with respect to its own value. More specifically, the steps of the W-MSR algorithm is as below.

W-MSR algorithm [29]:

(S1) At each time-step t, each non-faulty node i receives the values of its neighbors, and forms a sorted list.

(S2) If there are less than f_t values strictly larger than its own value, $x_i(t)$, then normal node i removes all values that are strictly larger than its own. Otherwise, it removes precisely the largest f_t values in the sorted list (breaking ties arbitrarily). Likewise, if there are less than f_t values strictly smaller than its own value, then node i removes all values that are strictly smaller than its own. Otherwise, it removes precisely the smallest f_t values.

(S3) Let $D_i(t)$ denote the set of nodes whose values are deleted by normal node i in step 2 at time-step t. Each normal node i applies the update

$$x_i(t+1) = \sum_{j \in N_i(t) \setminus D_i(t)} \omega_{ij}(t) x_{ij}(t) \tag{2.4}$$

where the weights $\omega_{ij}(t)$ satisfy the conditions in (2.1) with $N_i(t) \setminus D_i(t)$.

In the following theorem, we discuss the necessary and sufficient topological condition required for the success of the algorithm in time-invariant networks.

Theorem 2.3. *[29] In a time-invariant synchronous network with maximum f_t number of Byzantine nodes where each non-faulty node updates its state value according to the W-MSR algorithm with parameter f_t, approximate Byzantine consensus is guaranteed if and only if the topology of the normal network is $(f_t + 1)$-robust.*

The next theorem presents the sufficient condition that the algorithm succeeds in time-varying networks.

Theorem 2.4. *[29] In a synchronous time-varying network with maximum f_t number of Byzantine nodes, if $\{t_l\}$ with $l = 1, 2, ...$ is the set of time instances that the topology of the network of non-faulty nodes is at least $(f_t + 1)$-robust, approximate Byzantine consensus is guaranteed using W-MSR algorithm with parameter f_t if $|\{t_l\}| \to \infty$ as $t \to \infty$, for all $i \in V \setminus F$.*

The results in Theorems 2.3 and 2.4 for the W-MSR algorithm are valid for synchronous networks not for asynchronous ones. Furthermore, in networks with delay on communication paths achievement of approximate Byzantine consensus using W-MSR algorithm has not been discussed in [29].

2.5.2. Asynchronous MSR Algorithm

This fault-tolerant distributed consensus algorithm is designed for asynchronous time-invariant networks. In this algorithm, the sent messages are tagged by update round r and each non-faulty node waits to receive a certain number of values with a specific round tagged to it.

Asynchronous MSR Algorithm [22, 26]:

Steps to be performed by each non-faulty node $i \in V \setminus F$ in its r^{th} round are as follow:

(S1) Transmit current state $x_i(t)$ on all outgoing edges. The messages are tagged by round number r.

(S2) Wait until $|N_i| - f_t$ number of state values with round number r are received where f_t is the maximum number of Byzantine nodes.

(S3) Sort the received values in an increasing order and remove f_t smallest and f_t greatest values. If the set of deleted values are represented by $D_i(t)$, node i updates its state according to the below equation.

$$x_i(t+1) = \frac{1}{|N_i| - 3f_t + 1} \sum_{j \in N_i(t) \setminus D_i(t)} x_{ij}(t) \tag{2.5}$$

In the following theorem, we discuss the necessary and sufficient topological condition for the success of the above algorithm.

Theorem 2.5. *[26] In an asynchronous time-invariant network with delay on communication paths and f_t number of Byzantine nodes, where non-faulty nodes update their state values based on the asynchronous MSR algorithm with parameter f_t, approximate Byzantine consensus is guaranteed if and only if the network of non-faulty nodes is $(2f_t + 1)$-robust.*

In this algorithm, the messages are tagged by time and each non-faulty node should wait to receive certain number of messages. Moreover, its necessary and sufficient condition for success is more restrictive compared to W-MSR algorithm in synchronous networks.

2.6. Summary of the Chapter

In this chapter, we have explained notions of exact and approximate consensus and presented a well-known approximate consensus algorithm and discussed its behavior facing a Byzantine node after defining Byzantine fault. Then, approximate Byzantine consensus has been defined and it has been shown that use of approximate agreement is essential since exact agreement is not guaranteed in networks which adopt averaging-based algorithms.

The notion of robustness is presented and formulated. Furthermore, two well-known fault-tolerant algorithms have been introduced and their shortcomings have been discussed.

In the next chapter, we propose a novel fault-tolerant algorithm and prove its success in asynchronous and synchronous networks with delay on communication path. We show that our novel algorithm compensates the shortcomings of the presented algorithms in this chapter.

3. THE PROPOSED MSR ALGORITHM

In this chapter, we present our MSR algorithm and derive the necessary and sufficient conditions for its success using the notion of robustness. Some numerical examples are included to evaluate the performance of the algorithm.

3.1. The MSR Algorithm

In this part, a novel fault-tolerant MSR algorithm is proposed. This algorithm injects the capability of resisting against adversarial behavior of Byzantine nodes to non-faulty nodes in asynchronous and synchronous networks with or without time delay on communication paths.

Let the iteration instants of node i be represented by the set $\{t_0^i, t_1^i, t_2^i, ..., t_r^i\}$ where $i \in V$ and r, a non-negative integer, is the number of the iteration. $x_i(t)$ is the state value of node i at time t and $x_{ij}(t)$ is the state value of node j received by node i. F is the set of faulty nodes and $|F| = f_t$. Hence, $V \setminus F$ comprises the set of non-faulty nodes.

MSR Fault-Tolerant Algorithm:

(S1) Each non-faulty node i sends its state value $x_i(t_r^i)$ at t_r^i.

(S2) Each non-faulty node i saves the values that it has received in time interval Δt_r^i. If it has received more than one state value from a node, it considers the most recent value that it has received. If it has not received any value from its neighbor node in its update time interval Δt_r^i, it uses the old received values. Then, it sorts the received values in the increasing order.

(S3) Node i eliminates f_a number of maximum values in the sorted list that are greater than its current state value. If the number of greater values is less than f_a, it eliminates all of them. Similarly, it deletes the f_a number of minimum values in the list that are less than its state value. Let $D_i(t)$ be the set of nodes with

deleted state values

(S4) Node i updates its state value as follows.

$$x_i(t_{r+1}^i) = \sum_{j \in N_i(t_r^i) \setminus D_i(t_r^i)} \omega_{ij}(t_r^i) x_{ij}(t_r^i) \tag{3.1}$$

Each node uses its state value in the update, i.e., $x_i(t_r^i) \in N_i(t_r^i)$.

The averaging coefficients $\omega_{ij}(t_r^i)$ are supposed to satisfy the following conditions:

- $\omega_{ij}(t_r^i) = 0$ whenever $j \notin N_i(t_r^i) \setminus D_i(t_r^i)$.
- There exists a pre-assigned positive constant α such that $\alpha \leq \omega_{ij}(t_r^i) \leq 1$, for all $j \in N_i(t_r^i) \setminus D_i(t_r^i)$.
- $\sum_{j=1}^n \omega_{ij}(t_r^i) = 1$, $\forall i \in V$.

Note that, the above algorithm with $f_a = 0$ reduces to the Linear Consensus algorithm which is presented in Chapter 2 and widely discussed in the literature (see e.g. [10], [12], [11] and the references there).

The proposed algorithm is an MSR algorithm due to the following reasons:

(i) It eliminates at most $2f_a$ number of received state values at (S3). Therefore, it *reduces* the cardinality of $N_i(t)$.

(ii) It *selects* the state values that are in $N_i(t) \setminus D_i(t)$ at (S4).

(iii) It takes the weighted *mean* of the selected values at (S4).

Our proposed MSR algorithm has several differences with the other fault-tolerant algorithms in the literature. The above algorithm does not necessarily delete $2f_a$ number of received state values at each iteration. Each non-faulty node does not wait to receive a certain number of state values to update its state. Messages are not tagged by time. Furthermore, taking weighted rather than just arithmetic mean is possible for non-faulty nodes and they can change the weights in each iteration respect to (3.1).

3.2. Proof of Success

Considering the definition of Byzantine fault that has appeared in Chapter 2 and the MSR algorithm, the overall network dynamics can be represented as a non-autonomous nonlinear system because (i) control over the behavior of a Byzantine node is not possible (it may behave non-linearly), (ii) it is not guaranteed that in any iteration a non-faulty node eliminates the received state values of its Byzantine neighbor(s) using MSR algorithm and (iii) the nature of the MSR algorithm and asynchronous behavior of the system, makes it a time dependent system.

Let us define the network of non-faulty nodes as a subgraph which contains all of the non-faulty nodes. Let $M(t)$ be the maximum state value of the set of non-faulty nodes and $m(t)$ the minimum of them at time t. The maximum agreement error in the network is defined as:

$$e(t) = M(t) - m(t) \tag{3.2}$$

Note that, $e(t) \geq 0$ for all $t \geq 0$. Furthermore, $e(t) = 0$ if and only if $M(t) = m(t)$.

Hence, a network is said to reach approximate Byzantine consensus if the following conditions are satisfied.

- **Agreement:** There exists a $t_0 \geq 0$ such that for all $t \geq t_0$, $e(t) \leq \epsilon$
- **Validity:** For all $t \geq 0$, $m(0) \leq m(t) \leq M(t) \leq M(0)$.

In an asynchronous Byzantine network, each non-faulty node might update its state at arbitrary times. However, the interval between the update instants is bounded. Let

$$\Delta t_r^i = t_{r+1}^i - t_r^i \tag{3.3}$$

denote the time interval between two successive updates in the state of node i at round

r. Since this time interval is bounded for all of the non-faulty nodes, we can claim there exists an $\alpha_t \in \mathbb{R}$ such that $\Delta t_r^i \leq \alpha_t$, for all $i \in V \setminus F$ and non-negative integers r. Moreover, because of inherent discrete dynamics of consensus networks, there exists a $\beta_t \in \mathbb{R}$ such that $\Delta t_r^i \geq \beta_t$. Communication delay between non-faulty nodes is assumed to be bounded. As result, there exists an $\alpha_\tau \in \mathbb{R}$ such that $0 \leq \tau_{ij} \leq \alpha_\tau$, for all i, $j \in V \setminus F$. In this way, we define $\Delta t_m = \beta_t$ which is the minimum update time interval in the network. In addition, $\Delta t_M = \alpha_t + \alpha_\tau$ is defined as the sum of maximum update time interval and maximum communication delay between non-faulty nodes. $\{0, t_{M_1}, t_{M_2}, ...\}$ is defined as the set of time sequences with $t_{M_{r+1}} = t_{M_r} + \Delta t_M$. Consequently, it can be inferred that a non-faulty node in Δt_M interval is guaranteed to receive at least one state value from each of its non-faulty neighbors. Similar to t_{M_r} we define, $t_{m_{r+1}} = t_{m_r} + \Delta t_m$.

In [22], [25], [26] it is assumed that non-faulty nodes can not consider a bound in which they are guaranteed to receive all values of their non-faulty neighbors. However, as discussed above, it is reasonable to assume that there exist bounded sequences of time that each node is guaranteed to receive at least one value from each of its non-faulty neighbors. But they have no knowledge about this bound (Δt_M). We use this bound and its related sequences to analyze the convergence of the algorithm.

Lemma 3.1. *In an asynchronous network with bounded delay on communication links and f_t number of Byzantine nodes, in which, nodes update their state values using the MSR algorithm with parameter f_a, it is guaranteed that $M(t_{m_{r+1}}) \leq M(t_{m_r})$ and $m(t_{m_{r+1}}) \geq m(t_{m_r})$, for all non-negative integer r, if and only if $f_a \geq f_t$.*

Proof. (Sufficiency) Suppose the MSR algorithm is applied to the system with $f_a \geq f_t$. Note that, at any time t_{m_r}, there is no non-faulty node i that accepts any value greater than $M(t_{m_r})$ or less than $m(t_{m_r})$ because $f_a \geq f_t$. By utilizing (3.1) with its conditions on the weights, we have $m(t_{m_r}) \leq x_i(t_{m_{r+1}}) \leq M(t_{m_r})$ for each non-faulty node i. As result, $M(t_{m_{r+1}}) \leq M(t_{m_r})$ and $m(t_{m_{r+1}}) \geq m(t_{m_r})$.

(Necessity) Suppose $f_a < f_t$. Consider a network with f_t number of Byzantine

nodes that send values strictly greater than $M(t_{m_r})$ to non-faulty node $i^\star$ at time t_{m_r}. Node $i^\star$ has state value $x_{i^\star}(t_{m_r}) = M(t_{m_r})$ and no non-faulty node as neighbor. Also, suppose that node $i^\star$ updates its state value at time $t_{m_{r+1}}$ using the MSR algorithm with parameter f_a. From (3.1), we have:

$$\begin{aligned} x_{i^\star}(t_{m_{r+1}}) &= \sum_{j \in N_i(t_{m_{r+1}}) \setminus D_i(t_{m_{r+1}})} \omega_{ij}(t_{m_{r+1}}) x_{ij}(t_{m_{r+1}}) \\ &= \omega_{ii}(t_{m_{r+1}}) x_i(t_{m_{r+1}}) \\ &+ \sum_{j \in N_i(t_{m_{r+1}}) \setminus D_i(t_{m_{r+1}}), j \neq i} \omega_{ij}(t_{m_{r+1}}) x_{ij}(t_{m_{r+1}}) > M(t_{m_r}) \end{aligned}$$

Therefore, $M(t_{m_{r+1}}) > M(t_{m_r})$ which leads to a contradiction. A similar approach can be adopted for $m(t_{m_r})$. □

Corollary 3.1. *In an asynchronous network with bounded delay on communication links and f_t number of Byzantine nodes in which nodes use the MSR algorithm to update their state values with parameter f_a ($f_a \geq f_t$), we have $m(0) \leq x_i(t) \leq M(0)$, for all $t \geq 0$ and for all $i \in V \setminus F$.*

Lemma 3.1 implies that using the MSR algorithm with parameter f_a, the maximum agreement error $e(t)$ stays bounded in any graph topology for all $t \geq 0$ if and only if the number of Byzantine nodes (f_t) is less than or equal to f_a. Corollary 3.1 indicates that state values of non-faulty nodes will remain within bounds $m(0)$ and $M(0)$ if the stated condition in Lemma 3.1 holds.

Theorem 3.1. *In an asynchronous network with bounded delay on communication links and maximum f_t number of Byzantine nodes, approximate Byzantine consensus is guaranteed using the MSR algorithm with parameter f_a ($f_a \geq f_t$), if and only if the network of non-faulty nodes is at least $(f_a + 1)$-robust.*

Proof. (Sufficiency) From Corollary 3.1, it is known that any state value of non-faulty nodes is bounded by $M(0)$ and $m(0)$. Furthermore, from Lemma 3.1, $e(t)$ is a non-

increasing function, upper bounded by $e(0) = M(0) - m(0)$ and lower bounded by 0. Hence, $lim_{t\to\infty} e(t) = \epsilon$, $\epsilon \in \mathbb{R}$ [31]. Now, we show by contradiction that if network of non-faulty nodes topology is at least $(f_a + 1)$-robust then $\epsilon = 0$ and asymptotic Byzantine consensus is guaranteed.

Assume that $\epsilon > 0$. In this way, there exists a non-negative integer $r^{'}$ that for all $r \geq r^{'}$, $M(t_{M_r}) \geq L_M$ and $m(t_{M_r}) \leq L_m$, where $L_M, L_m \in \mathbb{R}$, $L_M > L_m$ and $(L_M - L_m) \geq \epsilon$. Suppose the networks of non-faulty nodes with values greater than or equal to L_M and less or equal to L_m are denoted by κ_M and κ_m, respectively. From (3.1), the required conditions for nodes of κ_M to be able to keep their values greater or equal to L_M for all time sequences t_{M_r}, $r \in \{r^{'}, r^{'} + 1, r^{'} + 2, ...\}$, are that κ_M should be connected and there exists no $i \in \kappa_M$ that $|N_i(t_{M_r}) \setminus \kappa_M| \leq f_a$. So, by definition of MSR algorithm with value f_a, none of the nodes in κ_M can accept values from the nodes which are not member of κ_M. Similarly, this condition should be held for all nodes in κ_m so that $L_M > L_m$ for all t_{M_r}. It emphasizes that at most κ_M and κ_m can be f_a-reachable. On the other hand, since the overall structure of the network of non-faulty nodes is (f_a+1)-robust, due to the definition of robustness, f_a-reachability of κ_M and κ_m at the same time is not possible. So, the condition on the bounds $(L_M > L_m)$ is not valid and $L_M = L_m$.

(Necessity) Suppose the structure of the network of non-faulty nodes is not at least (f_a+1)-robust. From the definition of robustness, there exist two disjoint subsets S_1 and S_2 with reachability less than or equal to f_a. Now, suppose that nodes of S_1 and S_2 have initial state values a and b respectively. Also, assume the Byzantine nodes do not send any value to their neighbors. Due to the definition of the MSR algorithm with parameter f_a, under these assumptions the asymptotic Byzantine consensus is impossible since nodes of S_1 and S_2 eliminate all the received values. □

Theorem 3.1 states that the proposed MSR algorithm achieves approximate Byzantine consensus in asynchronous networks with delay on communication paths if and only if the network of non-faulty nodes is at least $(f_a + 1)$-robust with $f_a \geq f_t$.

This condition is the same condition that is presented in [25], [29], [22], [26] for synchronous systems without time delay. Indeed, the main difference between the previous fault-tolerant algorithms and the proposed algorithm in this thesis is that *the synchronicity of the network and presence of delay on communication paths do not affect the topological condition required for the success of the algorithm.*

In the following, we show that the algorithm can also be applied to synchronous networks while the topological required condition for the success of the algorithm remains unchanged.

Theorem 3.2. *In a synchronous network with bounded delay on communication links and f_t number of Byzantine nodes, approximate Byzantine consensus is guaranteed using the MSR algorithm with parameter f_a ($f_a \geq f_t$) if and only if the network of non-faulty nodes is at least $(f_a + 1)$-robust.*

Proof. The proof is similar to the proof of Theorem 3.1 with $t_r^i = t_r^j$, for all $i, j \in V \setminus F$ and $r \in \mathbb{N}$. □

It is worthy to mention that the required topological condition for success of the algorithm is the same in the absence of delay in networks. In [26], [29], the authors obtained the same result for case of synchronous networks without delay on communication paths. This fact can be inferred from Theorem 3.1 and 3.2 by taking $\alpha_\tau = 0$.

3.3. Extension to Time-Varying Networks

The results can be extended to the case where the underlying graph of the network is time-varying. It is well known that the problem of consensus for networks with switching topology is not as straightforward as fixed topology. This problem is examined in [10, 11, 13]. The sufficient condition for reaching consensus in this case is that there exist infinite bounded sequences that the union of graphs in each sequence has a rooted out branching.

By adopting this fact, we present necessary and sufficient conditions for reaching approximate Byzantine consensus in time varying faulty networks.

Theorem 3.3. *In an asynchronous network with bounded delay on communication links and f_t number of Byzantine nodes, in which, the underlying graph topology is time-varying, if $\{t_l\}$ with $l = 1, 2, ...$ is the set of time instances that the topology of the network of non-faulty nodes is at least (f_a+1)-robust with bounded Δt_l and $\{t_r^i\}$ is the set of update time steps of node i, approximate Byzantine consensus is guaranteed using the proposed MSR algorithm with parameter f_a ($f_a \geq f_t$) if $|\{t_l\} \cap \{t_r^i\}| \to \infty$ as $t \to \infty$, for all $i \in V \setminus F$.*

Proof. Assume each sequence of the set $\big\{\{t_l\} \cap \{t_r^i\}\big\}$ is bounded by α_l. We take $\Delta t_M = \alpha_t + \alpha_\tau + \alpha_l$. The remainder of the proof is the same as the sufficiency part for the proof of Theorem 3.1. □

3.4. Numerical Analysis

In this section, we examine some examples to evaluate the performance of the proposed algorithm.

Example 3.1. Consider a complete graph with 4 nodes in which Node 4 is Byzantine. From the definition of robustness, the non-faulty network is 2-robust. Therefore, using asynchronous MSR algorithm that has been presented in Chapter 2, approximate Byzantine consensus is not guaranteed. Suppose that Nodes 1, 2 and 3 update their states every 5, 2 and 3 seconds, respectively. Additionally, we assume that the Byzantine node sends its state value every 4 seconds. Hence, the network is asynchronous and W-MSR algorithm can not guarantee approximate Byzantine consensus. The maximum communication delay between any node i and j is 5 seconds and the initial values of non-faulty nodes are $x_N(0) = [1, 3, 5]^T$. The maximum update sequence (t_{M_r}) is $\{0, 10, 20, ...\}$. Table 3.1 shows the progress of the algorithm.

In Table 3.1, the values sent by the Byzantine node are represented with a star index. At each time instant the stored state value of a non-faulty node by different

Table 3.1. Progress of the MSR algorithm in a complete network with four nodes.

t_{M_r}	Received values by Node 1	Received values by Node 2	Received values by Node 3
0	-	-	-
10	$\{3, 3.3, 5^\star\}$	$\{1, 5, 5^\star\}$	$\{1, 2.5, 5^\star\}$
20	$\{3.43, 2.63, 2^\star\}$	$\{2.76, 2.83, 4^\star\}$	$\{2.76, 2.7, 2^\star\}$
30	$\{2.73, 2.73, 2^\star\}$	$\{2.71, 2.72, 1^\star\}$	$\{2.71, 2.73, 1^\star\}$
40	$\{2.72, 2.72, 4^\star\}$	$\{2.72, 2.72, 2^\star\}$	$\{2.72, 2.72, 3^\star\}$

nodes might not be equal. This is due to the asynchronous behavior of the network and the different amount of delay in the communication paths. The simulation result of this network is depicted in Figure 3.1 which illustrates the convergence of state values of the non-faulty nodes to a common value.

Now we apply the asynchronous MSR algorithm to the network discussed in Example 3.1. The simulation results are illustrated in Figure 3.2 which shows that the consensus is not achieved since each non-faulty node deletes the values that it has received in each iteration.

Example 3.2. Consider the partially connected network that is illustrated in Figure 3.4 (which is also considered in [29]). The network of non-faulty nodes structure is 2-robust. The delay between the communication links are generated by `randi(5,n,n)` (we use Matlab notations) where n is the number of the total nodes. The values that are sent to different nodes by the Byzantine node (which is Node 2 in this example) are obtained by `5*rand`. Moreover, the interval between two successive update in each node is assumed to be irregular and in each iteration is updated as `randi(10,1,n)`. The result of simulation for this example is illustrated in Figure 3.4 which shows that the state values of the non-faulty nodes converges to a common value.

Now we apply the W-MSR algorithm to the network discussed in Example 3.2. The simulation results show that W-MSR algorithm can not succeed due to the asynchronous behavior of the network (Figure 3.5). Furthermore, asynchronous MSR al-

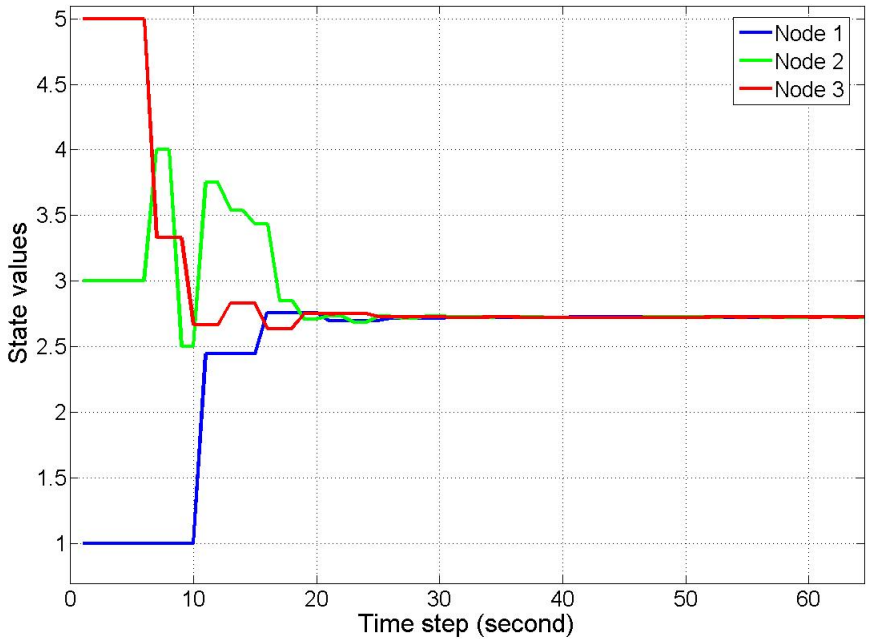


Figure 3.1. The simulation results of a complete network with 3 non-faulty nodes and one Byzantine node using the proposed MSR algorithm.

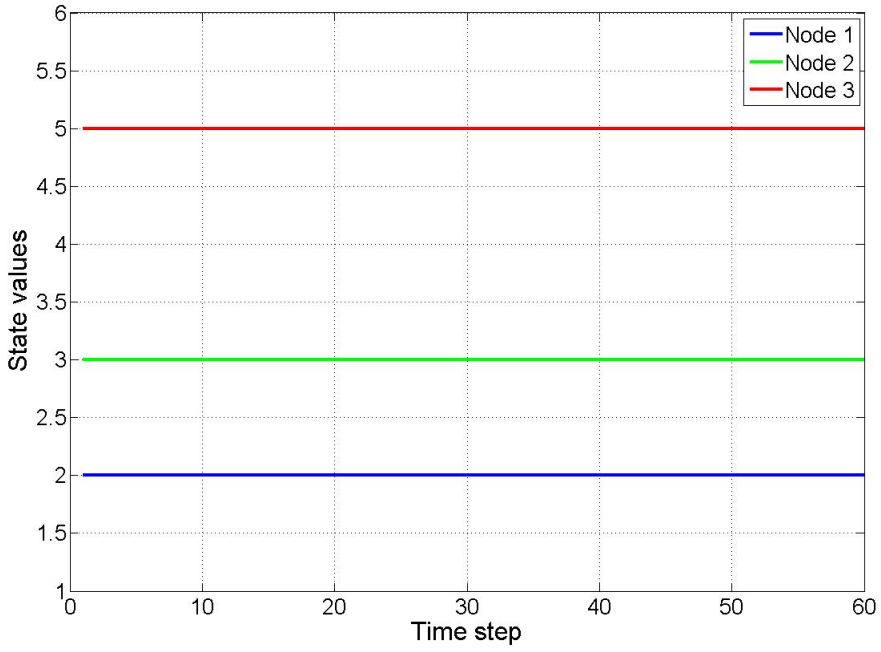


Figure 3.2. The simulation result of the network discussed in Example 3.1 using the asynchronous MSR algorithm.

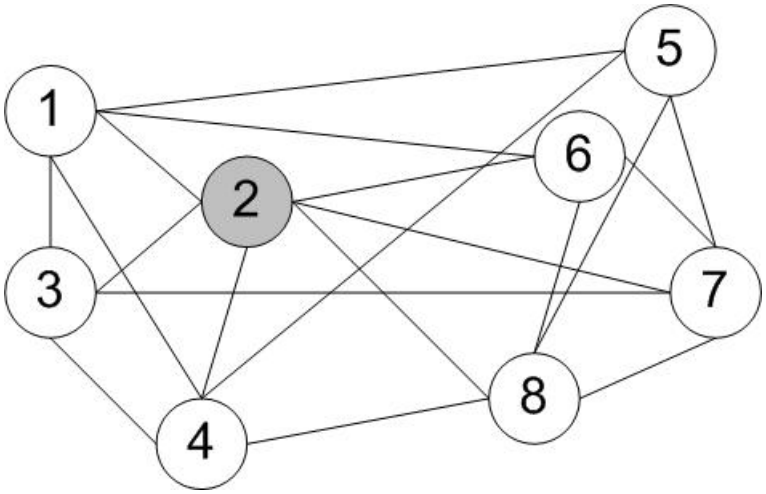


Figure 3.3. Network with seven normal nodes and one Byzantine node.

gorithm is not capable of guaranteeing approximate Byzantine consensus since the structure of the network of non-faulty nodes is 2-robust.

3.5. Implementation on a Multi-Robot System

In this section, we evaluate the performance of the proposed MSR algorithm by implementing it on multi-robot platform which is developed based on the small size soccer league of Robocup [32].

Consider four robots, each of which is placed initially at a corner of a square. The objective is that all of them gather at one point within the square while one of them is crashed and can not move flawlessly. In other words, the non-faulty robots should reach consensus on a point within the square and try to gather at that point. In the following, we describe the overall system.

The overall platform consists of different parts and components such as network components, high level controller, vision system and robots. Cameras shoot movement of robots frame by frame and send the coordination data to high level controller of the robots. Period of sampling in the system equals to one frame which is about 16.7 ms. In this experiment, it is assumed that each non-faulty robot receives the coordinates of

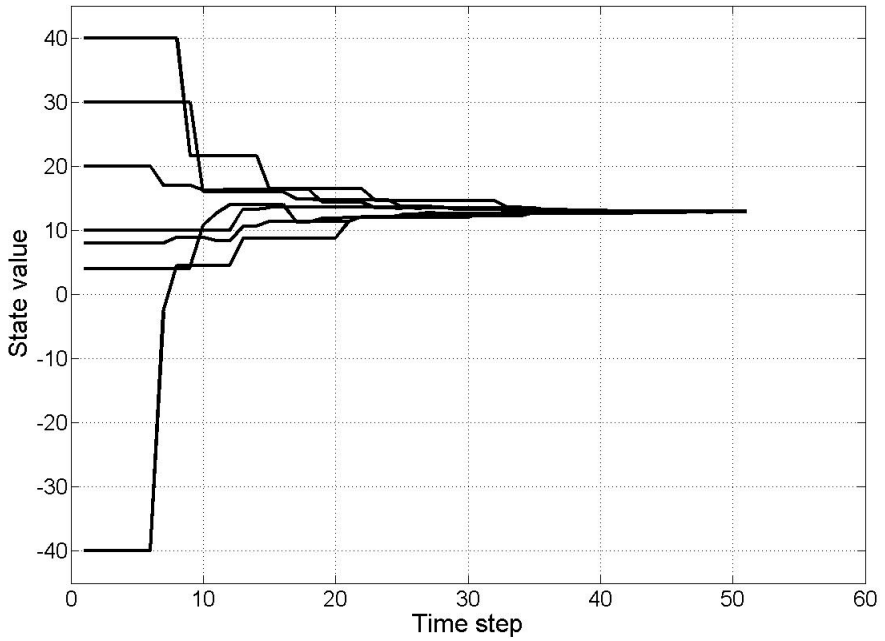


Figure 3.4. Simulation result of the partially connected network of Figure 3.3 which shows the convergence of non-faulty nodes to a common value.

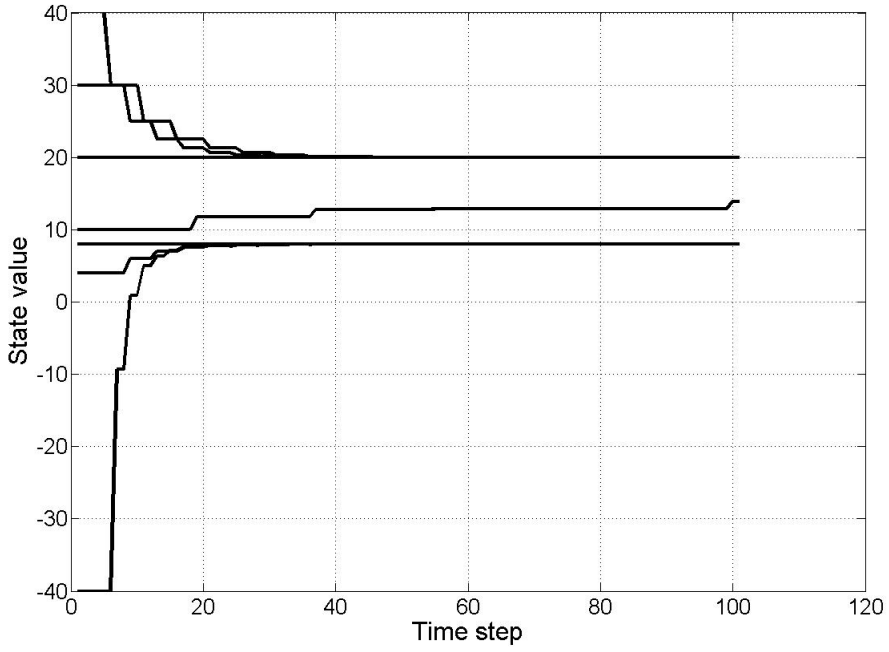


Figure 3.5. Simulation result of the partially connected network of Figure 3.3 using W-MSR algorithm.

other robots and update its reference position at arbitrary sequences that are bounded by 10 frames. Moreover, data processing and transmission between the components causes delay in the system and it is bounded by 8 frames. Note that, the overall network of robots can be modeled by a complete graph with four nodes.

The coordinates of each robot i can be summarized as:

$$p_i(t) = [p_{ix}(t),\ p_{iy}(t),\ p_{i\theta}(t)]^T$$

where p_{ix} and p_{iy} represent position of the robot and $p_{i\theta}$ shows heading of the robot. Moreover, t is the number of frame. The future coordinates of a robot in horizon of one frame can be approximated as

$$p_i(t+1) = p_i(t) + Tu_i(t)$$

where $T = 16.7$ ms and $u_i(t)$ is control signal which equals to:

$$u_i(t) = [v_{ix}(t),\ v_{iy}(t),\ \omega_i(t)]^T.$$

v_{ix} and v_{iy} are speeds of a robot in direction of x and y respectively and ω_i is the rotational speed. Let the reference position for each robot be

$$s_{ir}(t) = [p_{ix_r}(t),\ p_{iy_r}(t),\ p_{i\theta_r}(t)]^T.$$

Assume each robot updates its reference coordinate using the proposed MSR algorithm based on its current coordinate and other robots coordinates. Each robot applies the MSR algorithm to $p_{ix}(t)$ and $p_{iy}(t)$ variables to calculate new $p_{ix_r}(t)$ and $p_{iy_r}(t)$, at each update instance, respectively. The simulation results show that all non-faulty robots reach consensus and gather on a common reference position within the square that they are placed initially at its corners while the faulty robot can not prevent the consensus. Figure 3.6 illustrates the trajectory that each non-faulty robot has traveled

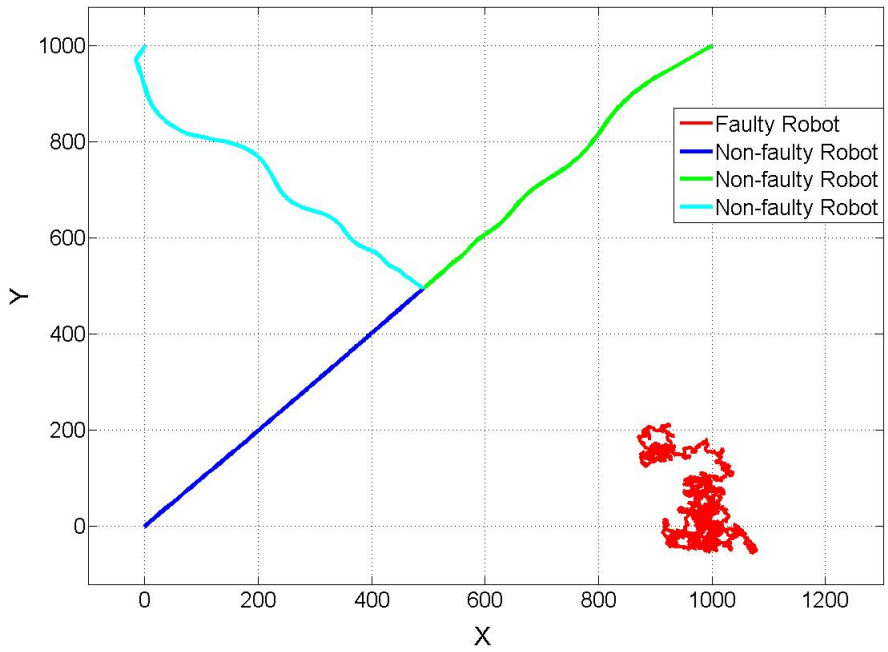


Figure 3.6. Trajectories that each robot has traveled.

to reach the common position.

3.6. Summary of the Chapter and Concluding Remarks

In this chapter, we have proposed a new MSR algorithm which has many advantages compared to similar algorithms that appeared in the literature. We have proved that this algorithm succeeds if and only if the underlying graph topology of the network is at least $(f_a + 1)$-robust.

We have also shown that the same topological condition required for the convergence of the algorithm is also valid for synchronous networks. Furthermore, the same fact has been concluded for networks with delay on communication paths. These results are in contrary to the previous results in the literature. We have extended our results to networks with time-varying underlying graph topology and extracted the required condition for success of the proposed algorithm. Some numerical examples have been presented to evaluate the performance of the algorithm. Finally, we applied the algorithm to multi-robot platform to assess its performance in real-life problems.

In the next chapter, we analyze the convergence rate of the proposed algorithm in time-invariant and varying networks. We also derive an upper bound for the convergence time of the algorithm.

4. CONVERGENCE ANALYSIS OF THE PROPOSED MSR ALGORITHM

In this chapter, convergence analysis of our proposed fault-tolerant algorithm is carried out. We derive two sets of higher bounds for the convergence rate and time of the proposed MSR algorithm. The bounds for the convergence rate and time of the algorithm are extracted by taking into account that the overall system is non-linear. Then, it is shown that some properties of non-faulty networks which use the linear consensus algorithm (or the proposed MSR algorithm with parameter $f_a = 0$) are valid in faulty networks which adopt the proposed MSR algorithm. Using these properties and matrix representation, we calculate more optimized bounds for the convergence rate and time. Later, the results are extended to time-varying networks and the existence of a Lyapunov function is shown. Throughout the chapter, it is shown that the extracted bounds and other results are applicable to both asynchronous and synchronous networks with or without time delay on communication paths. Furthermore, it is proved that the bounds can be applied to non-faulty networks which adopt the linear consensus algorithm.

4.1. Convergence Rate and Time Analysis

Due to the non-linear behavior of networks that contain Byzantine node(s) (explained in Section 3.2), we can not apply linear properties of row stochastic matrices. So, the methods that were used in [33–35] can not be adopted here. Instead, in this section, we derive bounds for convergence rate and time taking into account that the overall system dynamics are non-linear.

Theorem 4.1. *In an asynchronous network with bounded delay on communication links and maximum f_t number of Byzantine nodes, in which, each non-faulty node updates its state value using the proposed MSR algorithm with parameter f_a ($f_a \geq f_t$)*

we have

$$e\Big(t + (n - f_a - 1)\Delta t_M\Big) < e(t) \tag{4.1}$$

if and only if the structure of the network of non-faulty nodes is $(f_a + 1)$*-robust.*

Proof. (Sufficiency) Suppose the structure of the network of non-faulty nodes is (f_a+1)-robust. Let $S_M(t)$ and $S_m(t)$ indicate the set of non-faulty nodes with values $M(t)$ and $m(t)$, respectively. Due to the definition of robustness, either $S_m(t)$ or $S_M(t)$ is at least (f_a+1)-reachable. Without loss of generality, suppose $S_m(t)$ is (f_a+1)-reachable. This means that the complement of $S_m(t)$ has at least $f_a + 1$ nodes which implies that

$$|S_m(t)| \leq n - (f_a + 1).$$

From (3.1) after Δt_M, at least one node in $S_m(t)$ will increase its value. As result, the cardinality of S_m will decrease, i.e,

$$|S_m(t + \Delta t_M)| \leq |S_m(t)| - 1.$$

Suppose the network is iterated $(n - f_a - 2)\Delta t_M$. Hence,

$$|S_m\big(t + (n - f_a - 2)\Delta t_M\big)| \leq 1$$

The above equations imply that after $(n - f_a - 1)\Delta t_M$, $m(t)$ increases so $e(t)$ decreases which results in (4.1).

(Necessity) Suppose the structure of the network of non-faulty nodes is not at least $(f_a + 1)$-robust. Regarding the definition of robustness, there exist two disjoint subsets S_1 and S_2 with reachability less than or equal to f_a. Now, assume S_1 is S_m and S_2 is S_M with initial state values m and M respectively. Also, assume the Byzantine nodes do not send any value to their neighbors. Due to the MSR algorithm

with parameter f_a, under these assumptions the approximate Byzantine consensus is impossible because nodes of S_m and S_M eliminate all of the received values that come from other sets. Consequently, occurrence of (4.1) is not guaranteed. □

Theorem 4.1 is valid for asynchronous networks without delay on communication links as well as synchronous networks with and without delay. In networks which delay on communication links does not exist, one can take $\alpha_\tau = 0$ in the proof. Furthermore, in synchronous networks without delay on communication links, $\Delta t_r^i = \Delta t_M$ for all $i \in V \setminus F$. Note that, Theorem 4.1 suggests a bound which is $\tau_e = (n - f_a - 1)\Delta t_M$ on decrease of the maximum agreement error.

In Theorem 4.1, we proved that after τ_e, the maximum agreement error decreases. In the next theorem, we find a maximum bound on the amount of this decrease.

Theorem 4.2. *In a network with maximum f_t number of faulty nodes and $(f_a + 1)$-robust structure ($f_a \geq f_t$), in which, each node updates its state value using the MSR algorithm,*

$$e(k) \leq \big(1 - \frac{\alpha\epsilon}{e(0)}\big)^k e(0) \tag{4.2}$$

where α is a pre-assigned positive constant value such that $\alpha \leq \omega_{ij}(t_r^i)$, $i \in V \setminus F$ and $j \in N_i(t_r^i) \setminus D_i(t_r^i)$ for all $t_r^i \geq 0$ explained in (3.1), ϵ is the consensus accuracy pre-assigned value and $k = \lfloor \frac{t}{\tau_e} \rfloor$ where $y = f(x) = \lfloor x \rfloor$ is the greatest integer function.

Proof. From Theorem 4.1, it is guaranteed that after τ_e amount of time, $M(t)$ decreases or $m(t)$ increases or both. Let us assume just one of them changes its value and it changes just once, e.g, $M(k)$ decreases. Therefore we have

$$\frac{e(k+1)}{e(k)} \leq \frac{\Big[M(k) - \sum\limits_{j \in N_i(k) \setminus D_i(k), j \neq i} \omega_{ij}(t_r^i)\big(M(k) - x_{ij}(t_r^i)\big)\Big] - m(k)}{M(k) - m(k)} \tag{4.3}$$

where $k \leq t_r^i \leq k+1$ and i is the node with maximum value.

$$\frac{\Big[M(k) - \sum_{j \in N_i(k) \setminus D_i(k), j \neq i} \omega_{ij}(t_r^i)\big(M(k) - x_{ij}(t_r^i)\big)\Big] - m(k)}{M(k) - m(k)} \leq \big(1 - \frac{\alpha\epsilon}{e(0)}\big) \tag{4.4}$$

ω_{ij} is replaced by α, ϵ is used instead of $M(k) - x_{ij}$ and $e(0)$ is written instead of $e(k)$. □

Theorem 4.1 and 4.2 suggest a bound on the convergence rate of the MSR algorithm. For a pre-assigned accuracy level ϵ, from Theorem 4.2 we can guarantee that $e(k) \leq \epsilon$ for $k \geq k_{max}$ where k_{max} is given as

$$k_{max} = \log_{\left(1 - \frac{\alpha\epsilon}{e(0)}\right)}^{\frac{\epsilon}{e(0)}}. \tag{4.5}$$

Using the proposed MSR algorithm with parameter f_a, if the network of non-faulty nodes is (f_a+1)-robust, $\alpha \leq \min_{i \in V \setminus F, j \in N_i(t_r^i) \setminus D_i(t_r^i)} \omega_{ij}(t_r^i) < 1$ for all $t_r^i \geq 0$ since, (i) all of the non-faulty nodes can not eliminate all of the received values (which is discussed in the proof of Theorem 3.1) and (ii) each non-faulty node i considers a pre-assigned minimum bound $\alpha > 0$ for assigning weights to the received values that it has not deleted $(N_i(t_r^i) \setminus D_i(t_r^i))$ which comes from the definition of the proposed algorithm and discussed in (3.1) for the conditions on the $\omega_{ij}(t_r^i)$.

Example 4.1. Consider a network with f_t number of Byzantine nodes and (f_a+1)-robust structure for the network of non-faulty nodes in which each non-faulty node uses the proposed MSR algorithm with parameter f_a $(f_a \geq f_t)$ to update its state. Let each non-faulty node i assign equal weights to the state values of its neighbor nodes that it takes into account to update its own state value, i.e., $\omega_{ij}(t_r^i) = \frac{1}{|N_i(t_r^i) \setminus D_i(t_r^i)|}$ where $j \in N_i(t_r^i) \setminus D_i(t_r^i)$. In this example, $\alpha = \frac{1}{n}$ where n is the total number of nodes in the networks. Note that, from equation (4.5), $\alpha = \frac{1}{n}$ is the best choice which minimizes k_{max}.

4.2. Convergence Rate and Time Analysis Using Matrix Representation

In this section, we extract better bounds for the convergence rate and time using matrix representation.

Definition 4.1 (Ergodicity)**.** *A stochastic matrix W is called ergodic if $\lim_{t\to\infty} W^t = \mathbf{1}d^T$. Where $\mathbf{1}$ is a vector with all elements equal to 1 and d is a vector with elements $0 \leq d_i \leq 1$ and $\mathbf{1}^T d = 1$.*

Lemma 4.1. *[36] Let y be a non-negative vector and W a stochastic matrix. If $z = Wy$ then*

$$\max_i z - \min_i z \leq \gamma(W)\big(\max_i y - \min_i y\big) \tag{4.6}$$

where

$$\gamma(W) = \frac{1}{2}\max_{i,j}\sum_k |\omega_{ik} - \omega_{jk}| = 1 - \min_{i,j}\sum_k \min\{\omega_{ik}, \omega_{jk}\} \tag{4.7}$$

$\gamma(W)$ is called the coefficient of ergodicity of W. Furthermore, if $\gamma(W) < 1$, W is scrambling.

Let $W(t)$ be defined as the transition matrix of network, constructed using the coefficients that are presented in (3.1). $W(t)$ is a non-negative row-stochastic matrix for all $t \geq 0$. In the transition matrix of a network one can assign $\omega_{ii} = 1$ if $i \in F$.

Considering Definition 4.1 and Lemma 4.1, it is not guaranteed that matrix $W(t)$ is ergodic since there might exist more than one faulty nodes and the matrix $W(t)$ would have more than one zero eigenvalue.

As mentioned before, the proposed MSR algorithm with $f_a = 0$ resembles Linear Consensus algorithm. From Theorem 4.1 and Lemma 4.1, the following lemma can be concluded.

Lemma 4.2. *In an asynchronous network with bounded delay on communication links, no Byzantine node ($f_t = 0$) and $W(t)$ as the transition matrix at time t, in which, nodes update their state value using the proposed MSR algorithm with parameter $f_a = 0$ (or using Linear Consensus algorithm), if*

$$C = W(t)W(t + \Delta t_M)...W\big(t + (n - r)\Delta t_M\big)$$

then C is scrambling if and only if the structure of the network is r-robust.

Lemma 4.2 implies that using Linear Consensus algorithm, the property presented in Theorem 4.1 for faulty networks, is valid in non-faulty networks as well. In the following theorem, using the facts that are presented in this chapter, we show that the property of ergodicity value is valid for the network of non-faulty nodes in a faulty network.

Theorem 4.3. *In an asynchronous network with bounded delay on communication links, maximum f_t number of Byzantine nodes, in which, nodes update their state values using the MSR algorithm with parameter f_a ($f_a \geq f_t$),*

$$e(t + \tau_e) \leq (1 - \alpha)e(t) \tag{4.8}$$

if and only if the network of non-faulty nodes is (f_a+1)-robust where α is a pre-assigned positive constant value such that $\alpha \leq \omega_{ij}(t_r^i)$, $i \in V \setminus F$ and $j \in N_i(t_r^i) \setminus D_i(t_r^i)$ for all $t_r^i \geq 0$ explained in (3.1) and $\tau_e = (n - f_a - 1)\Delta t_M$.

Proof. (Sufficiency) We present the sufficiency proof in two parts for clarity. First, we assume that non-faulty nodes eliminate all the received values by Byzantine nodes or Byzantine nodes do not send different values at the same time to non-faulty nodes. Let

$$C(t) = W(t)W(t + \Delta t_M)...W\big(t + (n - f_a - 1)\Delta t_M\big)$$

and

$$\gamma'(C) = \frac{1}{2} \max_{i,j \in V \setminus F} \sum_k |c_{ik} - c_{jk}|$$

$$= 1 - \min_{i,j \in V \setminus F} \sum_k \min\{c_{ik}, c_{jk}\} \tag{4.9}$$

where

$$0 \leq \gamma'(C) \leq 1.$$

If $x_l(t) = M(t)$ and $x_{l'}(t) = m(t)$, from Lemma 3.1 and Theorem 4.1, we have:

$$x_l(t + \tau_e) - x_{l'}(t + \tau_e) < x_l(t) - x_{l'}(t)$$

Since $f_a \geq f_t$ and the structure of the network of non-faulty nodes is $(f_a + 1)$-robust, we have:

$$x_l(t + \tau_e) - x_{l'}(t + \tau_e) \leq \sum_j u_j x_j(t)$$

where $u_j = c_{lj} - c_{l'j}$ and $x_j(t) = x_{lj}(t) = x_{lj'}(t)$. Note that $\sum_j u_j = 0$. Let j' represent the indices j that $u_j > 0$ and j" the indices for which $u_j < 0$. Hence,

$$\theta = \sum_{j'} u_{j'} = -\sum_{j"} u_{j"} = \frac{1}{2} \sum_j |u_j| = \frac{1}{2} \sum_j |c_{lj} - c_{l'j}|.$$

Then,

$$x_l(t + \tau_e) - x_{l'}(t + \tau_e) \leq \theta \left(\frac{\sum\limits_{j'} |u_{j'}| x_{j'}(t)}{\sum\limits_{j'} |u_{j'}|} - \frac{\sum\limits_{j"} |u_{j"}| x_{j"}(t)}{\sum\limits_{j"} |u_j"|} \right)$$

$$\leq \theta\big(\max_{i\in V\setminus F} x_i(t) - \min_{i\in V\setminus F} x_i(t)\big) \leq \gamma'(C)e(t).$$

Now, suppose Byzantine nodes send different values to different nodes at the same time and non-faulty nodes might accept these values. As result,

$$x_l(t+\tau_e) - x_{l'}(t+\tau_e) \leq \sum_j c_{lj}x_{lj}(t) - \sum_j c_{lj'}x_{lj'}(t).$$

Moreover,

$$\min_{i,j\in V\setminus F} \sum_{k\in V\setminus F} \min\{c_{ik}, c_{jk}\} \geq \alpha.$$

Theorem 4.1 states that after τ_e, $e(t)$ decreases. Due to above equation, the amount of decrease has direct relation with c_{ij} weights. Therefore by applying (4.9),

$$e(t+\tau_e) \leq (1-\alpha)e(t)$$

which completes the sufficiency part of the proof.

(Necessity) Suppose the structure of the network of non-faulty nodes is not at least (f_a+1)-robust. From the definition of robustness, there exist two disjoint subsets S_1 and S_2 with reachability less than or equal to f_a. Now, suppose that nodes of S_1 and S_2 have initial state values a and b respectively. Also, assume the Byzantine nodes do not send any value to their neighbors. Due to the definition of the MSR algorithm with parameter f_a, nodes of S_1 and S_2 eliminate all the received values. Therefore,

$$\max_{i\in V\setminus F} x_i(t+\tau_e) - \min_{i\in V\setminus F} x_i(t+\tau_e) = \max_{i\in V\setminus F} x_i(t) - \min_{i\in V\setminus F} x_i(t)$$

and as result,

$$\gamma'(C) = 1$$

which completes the necessity part of the proof. □

For a faulty network, we refer to C as *semi transition matrix* over τ_e period. We use the word *semi* since, in faulty networks that nodes adopt the MSR algorithm to update their states, the following equation does not necessarily hold because of the behavior of Byzantine nodes.

$$x(t+\tau_e) = C(t)x(t)$$

On the other hand, C can represent many features of the network.

Using the above properties found in Theorems 4.1 and 4.3, we have

$$e(k) \leq \big(1-\alpha\big)^k e(0) \tag{4.10}$$

if and only if the network of non-faulty nodes is (f_a+1)-robust where α is a pre-assigned positive constant value such that $\alpha \leq \omega_{ij}(t_r^i)$, $i \in V \setminus F$ and $j \in N_i(t_r^i) \setminus D_i(t_r^i)$ for all $t_r^i \geq 0$ explained in (3.1) and $k = \lfloor \frac{t}{\tau_e} \rfloor$. Recall that, $y = f(x) = \lfloor x \rfloor$ is the greatest integer function.

(4.10) suggests a better bound for the convergence rate of the MSR algorithm compared to (4.4). In the following, we show that this bound is applicable to non-faulty networks as well.

Theorem 4.4. *In a network with no faulty node and connected underlying graph, in which, each node updates its state value using linear consensus algorithm proposed in Chapter 2,*

$$e(k) \leq \big(1-\alpha\big)^k e(0). \tag{4.11}$$

where α is a pre-assigned positive constant value such that $\alpha \leq \omega_{ij}(t)$, $i \in V$ and $j \in N_i(t)$ for all $t \geq 0$ explained in (2.1) and $k = \lfloor \frac{t}{\tau_e} \rfloor$ where $y = f(x) = \lfloor x \rfloor$ is the

greatest integer function.

Proof. Due to the definition of robustness, a network with connected underlying graph is at least 1-robust. Therefore, $\tau_e = (n-1)\Delta t_M$. The remainder of the proof is the same as the method that we used to derive (4.10). □

For a pre-assigned accuracy level ϵ, from (4.10) we can guarantee that $e(k) \leq \epsilon$ for $t \geq t_{max}$ where t_{max} is given as

$$t_{max} = \tau_e \big(\log^{\frac{\epsilon}{e(0)}}_{(1-\alpha)} + 1 \big). \tag{4.12}$$

The presented bounds for the convergence time and rate are applicable to both synchronous and asynchronous networks with or without time delay on communication paths.

4.3. Convergence Analysis of Time-Varying Networks

In this section, we extend our previous results to the case of networks with time-varying underlying graph topology.

Theorem 4.5. *In an asynchronous network with time-varying underlying graph topology, bounded delay on communication links and maximum f_t number of Byzantine nodes, in which, each non-faulty node updates its state value using the proposed MSR algorithm with parameter f_a ($f_a \geq f_t$),*

$$e\left(t + (n - f_a - 1)\Delta t_M \right) < e(t) \tag{4.13}$$

if $|\{t_l\} \cap \{t_r^i\}| \to \infty$ as $t \to \infty$, for all $i \in V \setminus F$, where $\{t_l\}$ with $l = 1, 2, ...$ represents the set of time instances that the topology of the network of non-faulty nodes is at least $(f_a + 1)$-robust with bounded Δt_l and $\{t_r^i\}$ is the set of update time steps of node i.

Proof. Assume each sequence of the set $\big\{\{t_l\} \cap \{t_r^i\}\big\}$ is bounded by α_l. Then,

$$\Delta t_M = \alpha_t + \alpha_\tau + \alpha_l. \tag{4.14}$$

The remainder of the proof is the same as the sufficiency part for the proof of Theorem 4.1. □

The above theorem is the extension of Theorem 4.1 for networks with time-varying underlying graph topology. The result of the theorem is applicable to synchronous systems as well. This fact can be concluded by taking $\{t_r^i\} = \{t_r^j\}$ for all $i, j \in V \setminus F$. Moreover, it is valid for networks without delay on communication paths where $\alpha_\tau = 0$.

Remark 4.1. *$v\big(x(k)\big) = e(k)$ is a Lyapunov function since $e(k) \geq 0$ and $e(t + \tau_e) - e(t) \leq 0$.*

Considering (4.14), Theorem 4.3 is applicable to the case of time-varying networks.

Theorem 4.6. *In an asynchronous network with time-varying underlying graph topology, bounded delay on communication links and maximum f_t number of Byzantine nodes, in which, nodes update their state values using the MSR algorithm with parameter f_a ($f_a \geq f_t$),*

$$e(t + \tau_e) \leq (1 - \alpha)e(t) \tag{4.15}$$

if $|\{t_l\} \cap \{t_r^i\}| \to \infty$ as $t \to \infty$, for all $i \in V \setminus F$, where $\{t_l\}$ with $l = 1, 2, ...$ represents the set of time instances that the topology of the network of non-faulty nodes is at least $(f_a + 1)$-robust with bounded Δt_l, $\{t_r^i\}$ is the set of update time steps of node i, α is a pre-assigned positive constant value such that $\alpha \leq \omega_{ij}(t_r^i)$, $i \in V \setminus F$ and $j \in N_i(t_r^i) \setminus D_i(t_r^i)$ for all $t_r^i \geq 0$ explained in (3.1) and $\tau_e = (n - f_a - 1)\Delta t_M$.

Proof. The proof is the same as the sufficiency part of the proof of Theorem 4.3 using

(4.14). □

Using Theorem 4.7, (4.14) and by applying the same method that has been adopted for deriving a bound for the convergence rate of time-invariant networks, the maximum bound for the convergence rate for time-varying networks would be:

$$e(k) \leq \big(1-\alpha\big)^k e(0) \tag{4.16}$$

where $k = \lfloor \frac{t}{\tau_e} \rfloor$ in which $y = f(x) = \lfloor x \rfloor$ is the greatest integer function. Therefore, for a pre-assigned accuracy level ϵ, from (4.16) we can guarantee that $e(k) \leq \epsilon$ for $t \geq t_{max}$ where t_{max} is given as

$$t_{max} = \tau_e \big(\log_{(1-\alpha)}^{\frac{\epsilon}{e(0)}} + 1 \big) \tag{4.17}$$

if $|\{t_l\} \cap \{t_r^i\}| \to \infty$ as $t \to \infty$, for all $i \in V \setminus F$.

The results are applicable to synchronous networks as well where $\{t_r^i\} = \{t_r^j\}$ for all $i, j \in V \setminus F$.

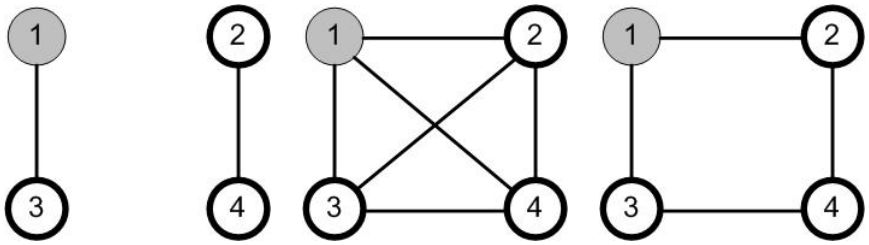
4.4. Numerical Analysis

In this section, we present some examples to evaluate our results in this chapter.

Example 4.2. Consider Example 3.1 where a complete graph with 4 nodes is the underlying graph of the network. Node 4 is Byzantine and the network of non-faulty nodes is 2-robust. $\Delta t_M = 10$ and as result, $\tau_e = 20$. It is assumed that each non-faulty node i assigns equal weights to state values of its neighbors and then applies (3.1). Hence, $\alpha = \frac{1}{4}$. $e(0) = 4$ and ϵ is supposed to be 0.1. Using (4.2), we have:

$$e(k) \leq (0.99375)^k e(0)$$

where $k = \lfloor \frac{t}{20} \rfloor$. By applying the second set of higher bounds for the convergence rate



(a) (b) (c)

Figure 4.1. (a) is a disconnected graph. (b) is a complete graph in which the network of non-faulty nodes is 2-robust and (c) is an incomplete graph with ring topology.

and time that are proposed in Section 4.3, we have:

$$e(k) \leq (0.75)^k e(0)$$

which is compromised using (4.11). Moreover, convergence time can be calculated using (4.12) which is $t_{max} \approx 275$.

Example 4.3. Consider the set of graphs depicted in Figure 4.1. Assume a time varying-network $G(t)$ which uses these graphs for its underlying graph topology. The network changes its topology every second. It starts from topology (a) in Figure 4.1. In the next second, it changes to topology (b). Afterwards, to (c). Then, to (a) and continues this loop. It is assumed that Node 1 is Byzantine which sends different values to its neighbor nodes. The values of the Byzantine node are generated by 5*rand (we use Matlab notations). The initial state values of non-faulty Nodes 2, 3 and 4 are $x_N(0) = [3, 5, 1]^T$. These nodes update their state value every 2, 4 and 5 seconds. Furthermore, the maximum communication delay between every two neighbor nodes is 5 seconds. As result, $\Delta t_M = 14$ which implies $\tau_e = 28$. If the nodes adopt equal weighting in (3.1) to update their states, $\alpha = \frac{1}{4}$. Let $\epsilon = 0.1$. Therefore, using (4.17) the maximum convergence time would be around 380 seconds. However, Figure 4.2 shows that non-faulty nodes converge to a common value and approximate Byzantine consensus is achieved much faster.

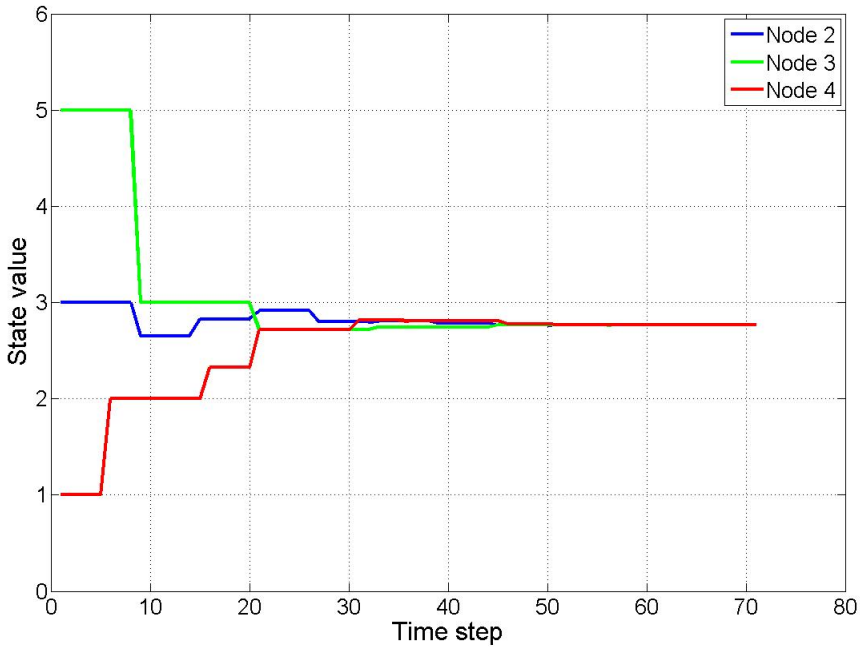


Figure 4.2. Simulation result of the network explained in Example 2.

4.5. Summary of the Chapter and Concluding Remarks

In this chapter, we have analyzed convergence properties of the proposed MSR algorithm. We started by introducing maximum bounds for the convergence rate and time of the algorithm, considering the network as a non-linear non-autonomous system. Then, using matrix representation, we have derived better bounds for the convergence rate and time. In Section 4.3, we extended our results to time-varying networks and bounds for the convergence rate and time of such networks are introduced. Moreover, the existence of a Lyapunov function for time-varying networks is shown. Finally, numerical analysis is included to evaluate our results.

5. USE OF CONVEX HULLS IN DESIGNING FAULT-TOLERANT DISTRIBUTED ALGORITHMS

In most of the fault-tolerant algorithms that have been introduced for Byzantine networks, it is assumed that each node has knowledge of the maximum number of faulty nodes, f_t, in the network. In this chapter, we propose a new family of algorithms which do not require this a priori information and evaluate its performance in complete networks facing Byzantine failures. We also propose a fault-tolerant algorithm for the case of Malicious failure. In the construction of our algorithms which we refer to as Shrinking Convex Hulls (SCH), we employ an interesting feature of non-faulty nodes in Byzantine networks in order to maintain each node's state value in a convex hull. We analytically examine the convergence properties of the proposed algorithm and compare its performance in simulations with respect to the existing literature. Note that, throughout this chapter, we refer to a network which contains at least one Byzantine node as *Byzantine network* and a network which contains Malicious nodes, *Malicious network*. Also, throughout the chapter, we refer to a synchronous network without time delay on communication paths, as simply, *network*. Therefore, in this chapter, t is a positive integer.

5.1. The Shrinking Convex Hull Algorithm for Complete Malicious Networks

In this section, we first investigate one of the significant features in consensus networks that can be adopted to develop a new family of fault-tolerant algorithms which do not require any initial knowledge on the maximum number of faulty nodes in the network. By use of the proposed algorithm, nodes do not throw away useful state values of their neighbors in each iteration. In other words, they can determine their suspected neighbors with more accuracy than the other algorithms. An additional advantage of this algorithm is that information loss is less due to the above mentioned reasons and as result, the final consensus value is nearer to mean of the initial values

of non-faulty nodes in complete networks.

In the following, we first introduce our algorithm which is designed to resist against Malicious failures in Byzantine complete networks and then prove its success.

Suppose that, each node i maintains a convex hull χ_i at time t that is defined as

$$\chi_i(t) = [\min_{j \in N_i'(t-1)} x_j(t-1), \max_{j \in N_i'(t-1)} x_j(t-1)] \tag{5.1}$$

where $N_i'(t)$ is a set that consists of state values of node i's neighbors which are in $\chi_i(t)$ (including node i's value too). Also, $\chi_i(0) = [-\infty, \infty]$.

For each time step the algorithm repeats the following.

SCH-CMN algorithm:

(S1) Each node i receives its neighbor values at time t.

(S2) If $x_j(t) \notin \chi_i(t)$ then node i throws it away.

(S3) Each normal node i applies the update rule

$$x_i(t+1) = \sum_{j \in N_i'(t)} \omega_{ij}(t) x_j(t) \tag{5.2}$$

where the weights $\omega_{ij}(t)$ satisfy the conditions discussed in (3.1).

(S4) If $|N_i'(t)| = 1$ (which is i's value) at time t, then node i picks the latest previous convex hull that $|N_i'| > 1$. Else, it updates the convex hull using the below protocol.

(I) After the transmitting sequence, each node i chooses $\min_{j \in N_i'(t)} x_j(t)$ and $\max_{j \in N_i'(t)} x_j(t)$.

(II) If $\min_{j \in N_i'(t)} x_j(t) > \min_{j \in N_i'(t-1)} x_j(t-1)$, it chooses the $\min_{j \in N_i'(t)} x_j(t)$ as the lower bound of $\chi_i(t+1)$, else, it chooses the second minimum value that fulfills the stated condition.

(III) If $\max_{j\in N_i'(t)} x_j(t) < \max_{j\in N_i'(t-1)} x_j(t-1)$, it chooses the $\max_{j\in N_i'(t)} x_j(t)$ as the higher bound of $\chi_i(t+1)$, else, it chooses the second maximum value that fulfills the stated condition.

The above algorithm is designed to overcome Malicious failures in a Complete Malicious Network (CMN); therefore we call it the SCH-CMN algorithm. Recall that, designing fault-tolerant algorithms for complete Byzantine networks is a challenging field which has been studied extensively in the literature. SCH-CMN algorithm is light-weight and guarantees Approximate Byzantine Consensus regardless of the number of faulty nodes in the network. As result, in the worst situation, two non-faulty nodes can resist against adverse behavior of as many as possible Malicious nodes in any complete network.

5.1.1. Proof of Success

In this part, we prove the success of the algorithm in the presence of Malicious failures in complete networks. First, we present two lemmata, which are utilized subsequently in proving the success of the SCH-CMN algorithm.

Lemma 5.1. *In a complete network in which nodes update their state value using the SCH-CMN algorithm, we have* $\chi_i(t) = \chi_j(t) \subset [m(0), M(0)]$*, for all* $i, j \in V \setminus F$ *and* $t > 0$*, if for all* $k \in F, m(0) \leq x_k(0) \leq M(0)$*.*

Proof. In order to prove this lemma, we proceed by induction. Note that we have $\chi_i(0) = \chi_j(0) = [-\infty, \infty]$ and $\chi_i(1) = \chi_j(1) = [m(0), M(0)]$, for all $i, j \in V \setminus F$. Suppose $\chi_i(t) = \chi_j(t)$ holds for all $i, j \in V \setminus F$. Since all non-faulty nodes receive the same information at time t, we have $N_i'(t) = N_j'(t)$ which results in $\min_{k\in N_i'(t)} x_k(t) = \min_{k\in N_j'(t)} x_k(t)$ and $\max_{k\in N_i'(t)} x_k(t) = \max_{k\in N_j'(t)} x_k(t)$. Utilize (5.1) to obtain $\chi_i(t+1) = \chi_j(t+1)$, for all $i, j \in V \setminus F$. □

Lemma 5.2. *In a complete network, in which nodes update their state value using the SCH-CMN algorithm, we have* $|N_i'(t)| \geq |V \setminus F|$*, for all* $i \in V \setminus F$*, if for all* $k \in F, m(0) \leq x_k(0) \leq M(0)$*.*

Proof. In order to prove this lemma, we proceed by contradiction. Suppose that we have $|N_i'(t)| < |V \setminus F|$, there exist a $t_1 \geq 0$ and an $i \in V \setminus F$. Consequently, there exists a $j \in V \setminus F$ that $x_j(t_1) \notin N_i'(t_1)$. Hence, we obtain $\chi_i(t_1) \neq \chi_j(t_1)$ which contradicts with Lemma 5.1. □

Theorem 5.1. *In a complete network in which nodes update their state value using the SCH-CMN algorithm, if for all* $k \in F, m(0) \leq x_k(0) \leq M(0)$*, approximate Byzantine consensus is guaranteed, i.e.,* $\forall i \in V \setminus F$, $lim_{t\to\infty} x_i(t) = L$, $m(0) \leq L \leq M(0)$.

Proof. From Lemmata 5.1 and 5.2, we have for all $t \geq 0$, for all $i, j \in V \setminus F$, $\chi_i(t) = \chi_j(t)$ and $|N_i'(t)| \geq |V \setminus F| \geq 2$. Also, from the algorithm $\chi_i(t+1) \subset \chi_i(t) \subset [m(0), M(0)]$. As result, $\min_{j\in N_i'(t+1)} x_j(t+1) > \min_{j\in N_i'(t)} x_j(t)$ and $\max_{j\in N_i'(t+1)} x_j(t+1) < \max_{j\in N_i'(t)} x_j(t)$. Subsequently, we conclude for all $t \geq 0$, there exists an $\epsilon \in \geq 0$ that $|\max_{j\in N_i'(t)} x_j(t) - \min_{j\in N_i'(t)} x_j(t)| \leq \epsilon$. It means $lim_{t\to\infty} \max_{j\in N_i'(t)} x_j(t) = lim_{t\to\infty} \min_{j\in N_i'(t)} x_j(t) = L$ and since, node i's convex hull always maintains its own value $lim_{t\to\infty} x_i(t) = L$. Thus, approximate Byzantine consensus is guaranteed. □

The proof verifies that in any complete network approximate Byzantine consensus is guaranteed, in the presence of any number and type of Malicious nodes. In other words, each non-faulty node can recognize its faulty neighbors precisely. This is an advantage comparing to other types of fault-tolerant algorithms.

5.1.2. Numerical Analysis

In this part, we will analyze performance of SCH-CMN algorithm in the presence of different Malicious behaviors of faulty nodes. The results confirm superior performance of the algorithm in various Malicious complete networks.

Example 5.1. Figure 5.1 shows the simulation results of a complete network with 5 nodes where 3 of them are Malicious. Two of the faulty nodes behave unstably and increase their values arbitrarily over time and the other one uses a White Gaussian Noise sequence to update its value. As depicted in Figures 5.1, the linear consensus

algorithm and the proposed MSR algorithm are unable to achieve approximate Byzantine consensus to the network while approximate Byzantine consensus is guaranteed by adopting SCH-CMN.

Example 5.2. Another case is analyzed in Figure 5.2 which illustrates the simulation results for a complete network with 8 nodes. In this example, 4 nodes are representing Malicious behavior such that 3 of them are unable to update their state values and just send constant values while another updates its value arbitrarily. The convergence of convex hulls of non-faulty nodes in the same network is demonstrated in Figure 5.3.

5.2. The Shrinking Convex Hull Algorithm for Complete Byzantine Networks

In this section, we introduce a new algorithm which can tolerate against Byzantine failures in complete networks. We use shrinking convex hull feature to facilitate our algorithm. It is shown that the algorithm has feasible performance in determining adverse behavior of nodes and it detects suspicious nodes accurately. In other words, there exists no t such that a non-faulty node disregards state value of another non-faulty node due to bad recognition.

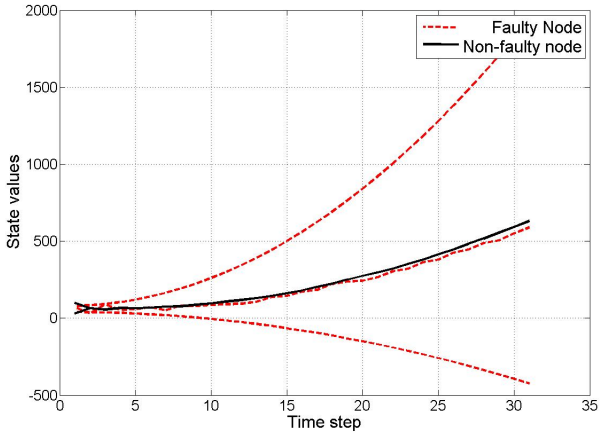
In the following, we first introduce our algorithm which is designed to resist against Byzantine failures in Byzantine complete networks and then prove its success.

Suppose that, each node i maintains a convex hull χ_i at time t that is defined as

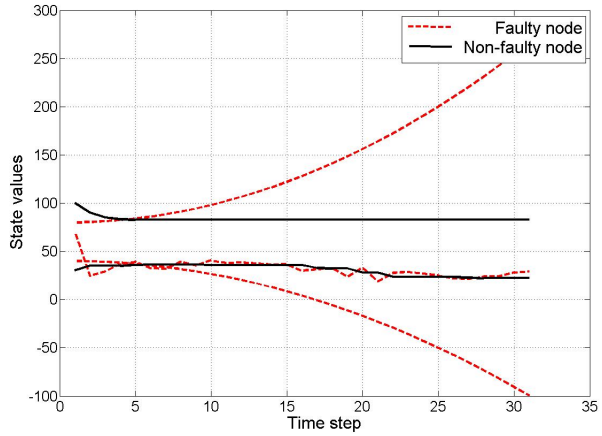
$$\chi_i(t) = [L\chi_i(t), H\chi_i(t)] \tag{5.3}$$

where $L\chi_i(t)$ and $H\chi_i(t)$ are lower and higher bound of node i's convex hull respectively. These values can be calculated using below equations.

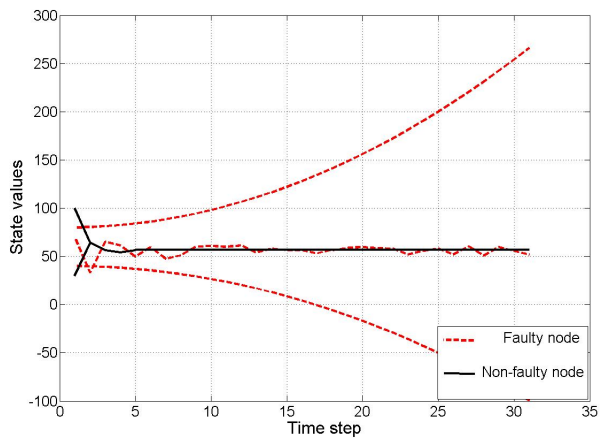
$$L\chi_i(t) = \frac{1}{n}\Big(x_i(t-1) + (n-1)[\min_{k \in N_{\chi_i}(t)} L\chi_k(t-1)]\Big) \tag{5.4}$$



(a) Linear consensus algorithm

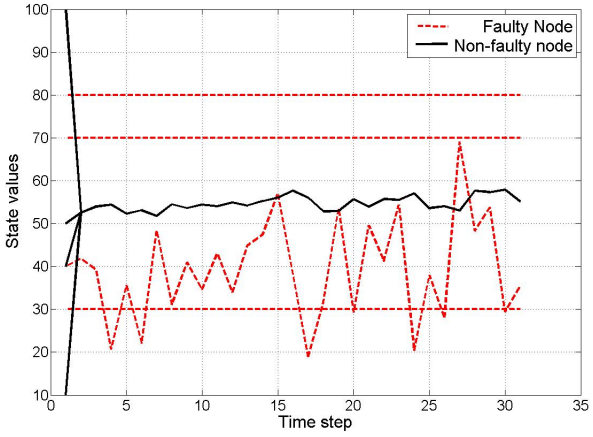


(b) MSR

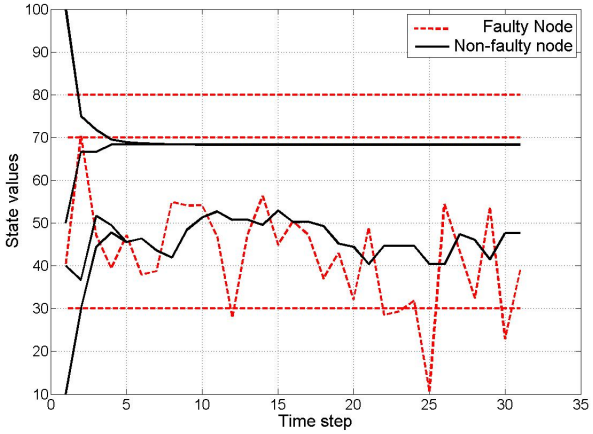


(c) SCH-CMN

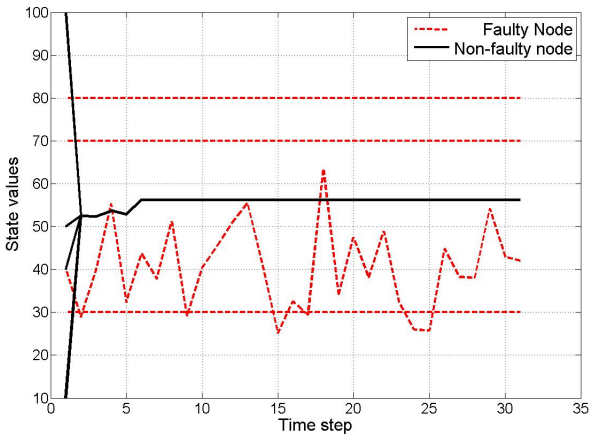
Figure 5.1. (a) and (b) show failure of linear consensus algorithm and the MSR respectively in the presence of 3 Malicious nodes while SCH-CMN succeeds in the same situation.



(a) Linear consensus algorithm



(b) MSR



(c) SCH-CMN

Figure 5.2. (a) and (b) show failure of linear consensus algorithm and the MSR respectively in the presence of 4 Malicious nodes, where 3 of them are just sending constant values, (c) shows SCH-CMN succeeds in the same situation.

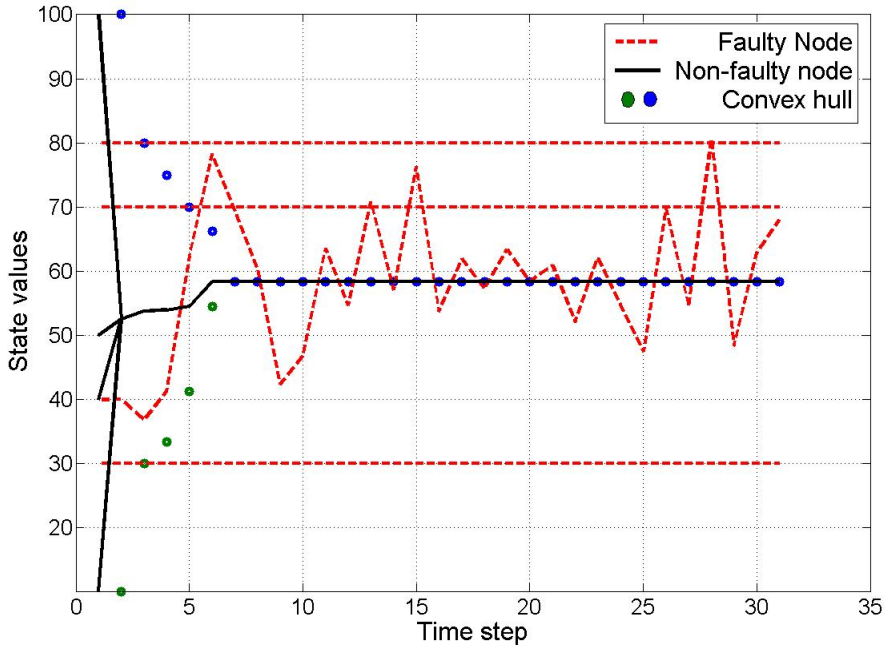


Figure 5.3. Simulation results for a complete network. The figure shows that convex hulls of non-faulty nodes converge as their states converge.

$$H\chi_i(t) = \frac{1}{n}\Big(x_i(t-1) + (n-1)[\max_{k \in N_{\chi_i}(t)} H\chi_k(t-1)]\Big) \tag{5.5}$$

$N_{\chi_i}(t)$ is a set that consists of acceptable convex hulls which are belong to node i's neighbors (including node i too) and $N_{\chi_i}(t) \subset N_i'(t)$ where $N_i'(t)$ is the set of state values that i uses them to update its own state. Also, note that $\chi_i(0) = [-\infty, \infty]$ and $m(0)$ and $M(0)$ are the minimum and maximum initial values of the non-faulty nodes.

For each time step the algorithm repeats the following.

SCH-CBN algorithm:

(S1) Each node i receives its neighbor values at time t.

(S2) If $x_j(t) \notin \chi_i(t)$ then node i throws it away and will not accept state value of node j for all future iterations.

(S3) Each normal node i applies the update rule

$$x_i(t+1) = \sum_{j \in N_i'(t)} \omega_{ij}(t) x_(t) \tag{5.6}$$

where $\omega_{ij}(t) = \frac{1}{|N_i'(t)|}$.

(S4) If $t = 1$, non-faulty node i chooses the minimum and maximum received state value as lower and higher bound of its convex hull respectively. Else, it updates its convex hull using the below protocol.

(I) All nodes send their convex hulls to their neighbor nodes.

(II) For $j \in N_i'(t)$, if $\chi_j(t) \subset \chi_j(t-1)$ (but $\chi_j(t) \neq \chi_j(t-1)$) and $x_i(t) \in \chi_i(t)$ then node i keeps the $\chi_j(t)$, else, it throws it away and will not accept any value from node j for all future iterations. Using the accepted convex hulls node i calculates $L\chi_i(t+1)$ and $H\chi_i(t+1)$ by adopting (5.4) and (5.5).

(III) If $L\chi_i(t+1) < L\chi_i(t)$ then node i does not update $L\chi_i$ which means $L\chi_i(t+1) = L\chi_i(t)$. Similarly, if $H\chi_i(t+1) > H\chi_i(t)$ then node i does not update $L\chi_i$ which means $H\chi_i(t+1) = H\chi_i(t)$.

(IV) Node i computes $\chi_i(t+1)$ by employing (5.3).

The above algorithm is designed to overcome Byzantine failures in a Complete Byzantine Networks (CBN); therefore we call it the SCH-CBN algorithm. Note that, designing fault-tolerant algorithms for complete Byzantine networks is a challenging task which was examined in many other articles. SCH-CBN algorithm guarantees approximate Byzantine consensus regardless of the number of faulty nodes in a network. As result, in the worst situation, two non-faulty nodes can resist against adverse behavior of as many as possible Byzantine nodes in any complete network. Compared to the previous algorithm (SCH-CMN), in this algorithm, each node uses decision rule of its trusted neighbors. It means, nodes decide not only on basis of their opinions but also, they inquire opinion of their trusted neighbor nodes to judge about states of their neighbors. Hence, decision making is distributed and decision rules are made not solely but locally. Indeed, this feature emphasizes on the notion of distributed decision making in determining decision rules.

5.2.1. Proof of Success

In this part, we prove the success of the algorithm with any number of Byzantine nodes. In this way, assume F is the set of faulty nodes and $x_{ij}(t)$ is the state value which is sent by node j to node i.

Lemma 5.3. *In a complete Byzantine network where nodes update their state values using SCH-CBN algorithm, $x_i(t) \in \chi_i(t)$, for all $i \in V \setminus F$ and for all $t \geq 0$, if $m(0) \leq x_{kj}(0) \leq M(0)$, for all $k \in F$ and $j \in V$.*

Proof. In order to prove this lemma we proceed by induction. We know $x_i(0) \in \chi_i(0) = [-\infty, \infty]$ and $x_i(1) \in \chi_i(1) = [m(0), M(0)]$, for all $i \in V \setminus F$. Suppose $x_i(t) \in \chi_i(t)$ holds for $t \in \mathbb{Z}_{>1}$. A non-faulty node always accepts convex hull of its non-faulty neighbors because convex hull range of a non-faulty node is decreasing as time passes. Therefore,

$$x_i(t+1) > L\chi_i(t) \geq \min_{j \in N_i'(t)} L\chi_j(t)$$

and

$$x_i(t+1) < H\chi_i(t) \leq \max_{j \in N_i'(t)} \chi_j(t).$$

Hence, by adopting (5.3), we conclude $x_i(t+1) \in \chi_i(t+1)$. □

Lemma 5.4. *In a complete Byzantine network where nodes update their state values using SCH-CBN algorithm, $x_i(t) \in \chi_j(t)$, for all $i, j \in V \setminus F$ and for all $t \geq 0$, if $m(0) \leq x_{kl}(0) \leq M(0)$, for all $k \in F$ and $l \in V$.*

Proof. In order to prove this lemma we proceed by induction. From the algorithm, $x_i(0) \in \chi_j(0) = [-\infty, \infty]$ and $x_i(1) \in \chi_j(1) = [m(0), M(0)]$, for all $i, j \in V \setminus F$. Suppose $x_i(t) \in \chi_j(t)$ holds for all $t > 1$. As we mentioned in the proof of Lemma 5.4, all non-faulty nodes accept convex hull of their non-faulty neighbors, therefore

$i \in N_{\chi_j}(t+1)$, for all $i, j \in V \setminus F$. Moreover, $V \setminus F \in N'_i(t)$, for all $i \in V \setminus F$ and as result,

$$x_i(t+1) = \frac{x_j(t) + \sum_{k \in N'_i(t), k \neq j} x_k[t]}{|N'_i(t)|}$$

From the algorithm, we have:

$$\max_{k \in N'_j(t)} H\chi_k(t) \geq H\chi_i(t) > \frac{\sum_{k \in N'_i(t), k \neq j} x_k[t]}{|N'_i(t)| - 1} > L\chi_i(t) \geq \min_{k \in N'_j(t)} L\chi_k(t).$$

Hence,

$$\frac{x_j(t) + (n-1)\max_{k \in N'_j(t)} H\chi_k(t)}{n} > x_i(t+1) > \frac{x_j(t) + (n-1)\min_{k \in N'_j(t)} L\chi_k(t)}{n}$$

and by adopting (5.3), we conclude $x_i(t+1) \in \chi_j(t+1)$. □

Theorem 5.2. *In a complete Byzantine network where nodes update their state values using SCH-CBN algorithm, approximate Byzantine consensus is guaranteed if* $m(0) \leq x_{kl}(0) \leq M(0)$, *for all* $k \in F$ *and* $l \in V$.

Proof. Assume $\mathcal{R}_i(t) = |H\chi_i(t) - L\chi_i(t)|$, for all $i \in V \setminus F$ is the range of node i's convex hull. Due to the algorithm, $\mathcal{R}_i(t)$ is a decreasing function. Therefore,

$$\lim_{t \to \infty} H\chi_i(t) = \lim_{t \to \infty} L\chi_i(t) = \Lambda$$

and $\Lambda \in [m(0), M(0)] \in \mathbb{R}$. As result, using Lemmata 5.3 and 5.4 we conclude $\lim_{t \to \infty} x_j(t) = \Lambda$, for all $j \in V \setminus F$. □

Theorem 5.2 demonstrates that SCH-CBN algorithm can guarantee approximate Byzantine consensus in a Byzantine complete network regardless of the number of faulty nodes.

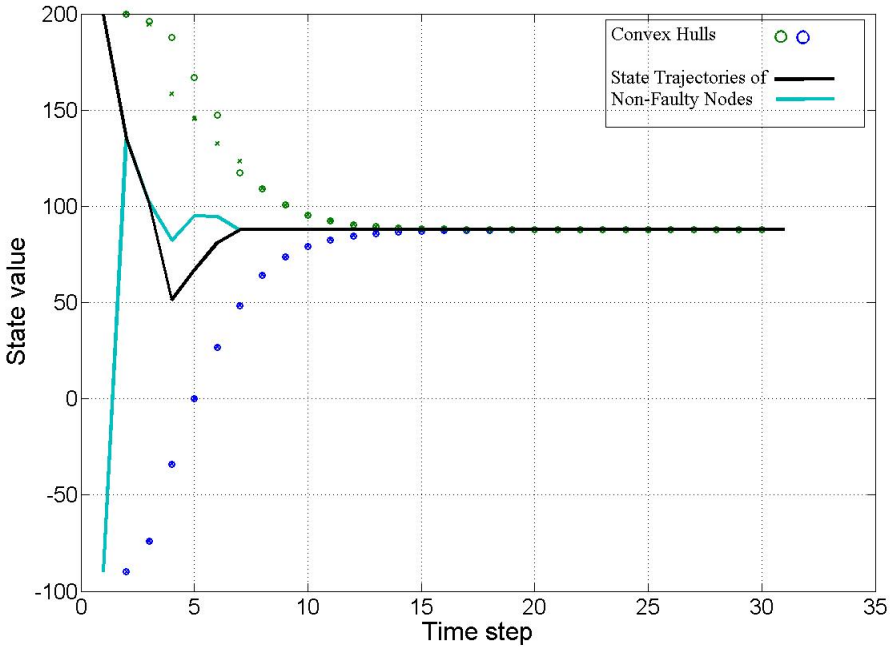


Figure 5.4. Simulation results for a complete network with 5 nodes where 3 of them are Byzantine.

5.2.2. Numerical Analysis

In this section, we analyze the performance of the algorithm by assessing an example.

Example 5.3. Consider a network of 5 nodes with complete underlying graph. Suppose 3 nodes are Byzantine and Nodes 4 and 5 are non-faulty. Nodes 1, 2 and 3 send different values to different nodes at the same time. These values are produced using a white Gaussian noise function of Matlab. Figure 5.4 shows the result of the simulation. It is shown that the non-faulty nodes converge to a common value. Moreover, convergence of the convex hulls are illustrated.

5.3. Comparison with the MSR Algorithm

In this section, we compare Shrinking Convex Hulls algorithms with MSR algorithms. We specifically compare SCH-CMN and SCH-CBN with the MSR algorithm

that is proposed in Chapter 3.

First, we derive the condition that the proposed MSR algorithm succeeds in complete networks.

Theorem 5.3. *In a complete Byzantine network where nodes update their state values using the MSR algorithm proposed in Chapter 3, approximate Byzantine consensus is guaranteed if and only if* $f_t \leq \frac{n}{2} - 1$*, where* f_t *and* n *are maximum number of faulty nodes and total number of nodes, respectively.*

Based on Theorem 5.3, below we enumerate the advantages and disadvantages of SCH algorithms compared to the proposed MSR algorithm.

- SCH algorithms can guarantee approximate Byzantine consensus in presence of $n - 2$ faulty nodes (which is the worst probable case) while the MSR algorithm can tolerate against $\frac{n}{2} - 1$ number of faulty nodes in a complete network. Note that, the term faulty here deputes both Malicious and Byzantine faults.
- SCH algorithms do not require any a priori knowledge on maximum number of faulty nodes while by adopting the MSR algorithm, each individual node needs this knowledge. Note that, this value should be chosen accurately due to different possible failure scenarios.
- By using SCH algorithms, information loss never happens which indicates there exist no iteration that state value of a non-faulty node gets rejected by another non-faulty node. On the other hand, this may happen several times by adopting the MSR algorithm.
- Using SCH algorithms, $m(0) \leq x_{kl}(0) \leq M(0)$, for all $k \in F$ and for all $l \in V$ while this assumption is not necessary if the proposed MSR algorithm is applied.
- The MSR algorithm is more light-weight than the SCH algorithms.
- The MSR algorithm can be applied to partially connected and time-varying networks.
- The MSR algorithm can also be applied to asynchronous networks.

5.4. Summary of the Chapter and Concluding Remarks

In this chapter, we have introduced a new family of algorithms, called Shrinking Convex Hulls based on the interesting property of non-faulty nodes. We have proposed two algorithms SCH-CMN and SCH-CBN for Malicious and Byzantine complete networks, respectively. We have derived the required condition for success of the algorithms. The numerical examples have been proposed to evaluate the performance of the algorithm as well.

While several other numerical examples can be examined, we believe that the results presented above verify the feasibility and superior performance of the SCH algorithms in complete networks in the presence of any number of Malicious and Byzantine nodes. Hence, the proposed algorithms are deemed to be robust enough to be applied in critical fields where the probability of nodes failure is high or security is a concern.

6. CONCLUSIONS

In this thesis, the problem of achieving approximate Byzantine consensus in faulty synchronous and asynchronous networks is examined. We have proposed a novel MSR algorithm and shown that the topological condition required for the success of the algorithm is more relaxed in asynchronous networks with delay on communication paths compared to the previous results. Furthermore, it is proved that the synchronicity of the network and presence of delay on communication paths do not affect this condition. This result is in contrary to the results appeared in the literature. We have also extended our results to the case of time-varying synchronous and asynchronous networks and proved the success of the algorithm there.

In Chapter 4, by using matrix representation, we have determined the convergence rate and time of the proposed MSR algorithm and extended the results to time-varying networks. Finally, a new family of algorithms, referred to as Shrinking Convex Hulls algorithms for Byzantine failure and a more restrictive but commonplace type of Byzantine failure known as Malicious failure has been presented in Chapter 5.

Our results not only extend the previous results in the literature but some of them can be considered as novel solutions to the problem of achieving approximate Byzantine consensus in faulty networks.

Future potential research problems related to this thesis can be itemized as follows:

- Designing fast MSR algorithms by selecting the averaging weights appropriately.
- Designing SCH algorithms that can be applied to partially connected networks.
- Analysis of SCH algorithms in asynchronous networks and the presence of delay on communication paths.
- Examining the behavior of Byzantine nodes and their impact on the overall dynamics of networks, so that designing new fault-tolerant algorithms without a

priori knowledge on the maximum number of faulty nodes reduces to an easier problem.

- Applying current results to real life applications such as multi-agent coordination, flocking and distributed estimation.
- In all of the relevant results that appeared in the literature, node dynamics are considered to be first order. The analysis of the proposed algorithms in networks where each agent has non-linear uncertain dynamics in the presence of disturbances and communication delay can be considered as an another interesting potential future work.

REFERENCES

1. Olfati-Saber, R., "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory", *IEEE Transactions on Automatic Control*, Vol. 51, No. 3, pp. 401-420, 2006.

2. Savkin, A. V., "Coordinated Collective Motion of Groups of Autonomous Mobile Robots: Analysis of Vicsek's Model", *IEEE Transactions on Automatic Control*, Vol. 49,No. 6, pp. 981-982, 2004.

3. Lin, Z., B. Francis and M. Maggiore, "Necessary and Sufficient Graphical Conditions for Formation Control of Unicycles", *IEEE Transactions on Automatic Control*, Vol. 50, No. 1, pp. 121-127, 2005.

4. Xi, X. and E. H. Abed, "Formation Control with Virtual Leaders and Reduced Communications", *Joint 44th Decision and Control and European Control Conference*, pp. 1854-1860, 2005.

5. Ren, W., R. W. Beard and E. M. Atkins, "Information Consensus in Multivehicle Cooperative Control", *IEEE Control Systems*, Vol. 27, No.2, pp. 71-82, 2007.

6. Das, A. and F. L. Lewis, "Distributed Adaptive Control for Synchronization of Unknown Nonlinear Networked Systems", *Automatica*, Vol. 46, No. 12, pp. 2014-2021, 2010.

7. Qu, Z., *Cooperative Control of Dynamical Systems: Applications to Autonomous Vehicles*, Springer, New York, NY, USA, 2009.

8. Hou, Z. G., L. Cheng and M. Tan, "Decentralized Robust Adaptive Control for the Multiagent System Consensus Problem Using Neural Networks". *"IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics",*, Vol. 39, No. 3, pp. 636-647, 2009.

9. Cheng, T. D. and A. V. Savkin, "Decentralized Control of Multi-Agent Systems for Swarming With a Given Geometric Pattern", *Computers and Mathematics with Applications*, Vol. 61, No. 4, pp. 731-744, 2011.

10. Jadbabaie, A., J. Lin and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules", *IEEE Transactions on Automatic Control*, Vol. 48, No. 6, pp. 988-1001, 2003.

11. Moreau, L., "Stability of Multiagent Systems with Time-Dependent Communication Links", *IEEE Transactions on Automatic Control*, Vol. 50, No. 2, pp. 169-182, 2005.

12. Olfati-Saber, R., J. A. Fax and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems", *Proceedings of the IEEE*, Vol. 95, No.1, pp. 215-233, 2007.

13. Ren, W. and R. W. Beard, "Consensus Seeking in Multiagent Systems Under Dynamically Changing Interaction Topologies", *IEEE Transactions on Automatic Control*, Vol. 50, No. 5, pp. 655-661, 2005.

14. Olfati-Saber, R. and R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays", *IEEE Transactions on Automatic Control*, Vol. 49, No. 9, pp. 1520-1533, 2004.

15. Lynch, N. A., *Distributed Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

16. Lamport, L., R. Shostak and M. Pease, "The Byzantine Generals Problem", *ACM Transactions on Programming Language Systems*, Vol. 4, No. 3, pp. 382-401, 1982.

17. Pease, M., R. Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults", *Journal of ACM*, No. 27, Vol. 2, pp. 228-234, 1980.

18. Castro, M. and B. Liskov, "Practical Byzantine Fault-tolerance and Proactive Recovery", *ACM Transactions on Computer Systems*, Vol. 20, No. 4, pp. 398-461, 2002.

19. Agmon, N. and D. Peleg, "Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots", *15th annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1070-1078, Philadelphia, PA, USA.

20. Pasqualetti, F., A. Bicchi and F. Bullo, "Consensus Computation in Unreliable Networks: A System Theoretic Approach", *IEEE Transactions on Automatic Control*, Vol. 57, No. 1, pp. 90-104, 2012.

21. Sundaram, S. and C. N. Hadjicostis, "Distributed Function Calculation via Linear Iterative Strategies in the Presence of Malicious Agents", *IEEE Transactions on Automatic Control*, Vol. 56, No. 7, pp. 1495-1508, 2011.

22. Dolev, D., N. A. Lynch, S. S. Pinter, E. W. Stark and W.E. Weihl, "Reaching Approximate Agreement in the Presence of Faults", *Journal of ACM*, Vol. 33, No. 3, pp. 499-516, 1986.

23. Kieckhafer, R. M. and M. H. Azadmanesh, "Reaching Approximate Agreement with Mixed-Mode Faults", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, pp. 53-63, 1994.

24. Azadmanesh, M. H. and R. M. Kieckhafer, "Low Cost Approximate Agreement in Partially Connected Networks", *Journal of Computing and Information*, No. 3, pp. 53-85, 1993.

25. Azadmanesh, M. H. and R. M. Kieckhafer, "Asynchronous Approximate Agreement in Partially Connected Networks", *International Journal of Parallel and Distributed Systems and Networks*, Vol. 5, pp. 26-34, 2002.

26. Vaidya, N. H., L. Tseng and G. Liang, "Iterative Approximate Byzantine Consen-

sus in Arbitrary Directed Graphs", *ACM symposium on Principles of Distributed Computing*, pp. 365-374, New York, NY, USA, 2012.

27. Vaidya, N., "Matrix Representation of Iterative Approximate Byzantine Consensus in Directed Graphs". *ArXiv e-prints*, March 2012.

28. LeBlanc, H. J. and X. D. Koutsoukos, "Consensus in Networked Multi-Agent Systems with Adversaries", *14th International Conference on Hybrid Systems: Computation and Control*, pp. 281-290, New York, NY, USA, 2011.

29. LeBlanc, H. J., H. Zhang, X. D. Koutsoukos and S. Sundaram, "Resilient Asymptotic Consensus in Robust Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 31, No. 4, pp. 766-781, 2013.

30. LeBlanc, H. J. and X. D. Koutsoukos, "Low Complexity Resilient Consensus in Networked Multi-Agent Systems with Adversaries", *15th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 5-14, New York, NY, USA, 2012.

31. Ponnusamy, S., *Foundations of Mathematical Analysis*. Springer, Boston, MA, USA, 2012.

32. Varol, O. F., O. Cihan, H. Esen, A. Haseltalab and M. Akar, "Team Description Paper", *Robocup*, 2013.

33. Olshevsky, A. and J. N. Tsitsiklis, "Convergence Speed in Distributed Consensus and Averaging", *SIAM Journal on Control and Optimization*, Vol. 48, No. 1, pp. 33-55, 2009.

34. Landau, H. J. and A. M. Odlyzko, "Bounds for Eigenvalues of Certain Stochastic Matrices", *Linear Algebra and Its Applications*, Vol. 38, pp. 5-15, 1981.

35. Blondel, V. D., J. M. Hendrickx, A. Olshevsky and J. N. Tsitsiklis, "Convergence

in Multiagent Coordination, Consensus, and Flocking", *Joint 44th Decision and Control and European Control Conference*, pp. 2996-3000, 2005.

36. Hartfield, D. J., *Markov Set-Chains*. Springer, New York, NY, USA, 1998.