ANALYSIS OF AN AVERAGING BASED SYNCHRONIZATION ALGORITHM IN CONTINUOUS-TIME

by

Didem Ersöz

B.Sc., Electrical and Electronics Engineering, Boğaziçi University, 2004

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 Department of Electrical and Electronics Engineering

Boğaziçi University

2010

ACKNOWLEDGEMENTS

This thesis would not have been possible if my supervisor Dr. Mehmet Akar had not supported me. I would like to express my profound gratitude to him for his trust, support, supervision and leadership.

I owe my deepest gratitude to Prof. Dr. Emin Anarım and I would like to thank him for his valuable guidance.

I also wish to thank Onur Cihan for his support.

Last but never the least, I wish to express my sincere gratitude to my family who made their support available at all times during my study.

ABSTRACT

ANALYSIS OF AN AVERAGING BASED SYNCHRONIZATION ALGORITHM IN CONTINUOUS-TIME

Synchronization in distributed networks plays a significant role in a broad range of applications that consist of mobile and/or wireless nodes which need to agree on a common notion of time to successfully fulfill the tasks that they are designed for. In this thesis, we introduce an averaging based distributed synchronization algorithm in continuous time and investigate its convergence properties for networks with time varying topology and time delay. The proposed distributed synchronization scheme is advantageous in the sense that it can be applied to large-scale networks due to its energy efficiency and its robustness to node failures. In the study of convergence properties, we utilize related results from graph theory, tools of Lyapunov functions, and Laplace transform. For fixed and time varying topology networks, it is shown that average consensus is achieved in the presence of free running frequencies, if the network is strongly connected and balanced. The result is also extended to networks with equal time delay, if the network topology is fixed, undirected, and connected. Extensive simulation results are carried out in Matlab to examine the related convergence scenarios.

ÖZET

ORTALAMA ALMA TEMELLİ BİR EŞ ZAMANLAMA ALGORİTMASININ SÜREKLİ ZAMANDA İNCELENMESİ

Bir çok ağ uygulamasında, ağı oluşturan düğümlerin ortak bir saat değerine sahip olması önemli bir rol oynar. Bu sayede, ağı oluşturan düğümler dizayn edildikleri işleri gerçekleştirebilirler. Bu tez çalışmasında, ortalama alma temelli bir dağıtık eş zamanlama algoritması tanıtılarak, yakınsama koşulları ilingesi sabit ya da değişen ve gecikmeli ağlar için incelenmiştir. Dağıtık eş zamanlama algoritmasının avantajları geniş ölçekli ağlara enerji verimliliği sağlayarak uygulanabilmesi ve zaman içerisinde iletişim kanallarında oluşan bozulmalara karşı dayanıklı olmasıdır.

Bu tezde tanıttığımız dağıtık eş zamanlama algoritmasının yakınsama koşullarını incelerken çizgi kuramından ilgili tanımlar, Lyapunov fonksiyonu, ve Laplas dönüşümü kullanılmıstır. İlingesi sabit ve değişen ağlarda, düğümlerin iç frekansları da dahil edildiğinde, eğer ağ yapısı güçlü bağlaşık ve dengeli ise, eş zamanlama algoritması uygulandığında, düğümlerin ortalama değere yakınsadığı gösterilmiştir. Gecikmeli ağlarda ise, tüm iletişim kanallarının iletilen mesajlara aynı miktarda gecikme uyguladığı kabulu ile, eğer ağ yapısı sabit, yönsüz ve bağlaşık ise, düğümlerin ortalama değere yakınsadığı gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The technological advances in low-cost wireless devices in the recent past have enabled the design of large scale wireless networks, which in turn led to vast and diverse application areas. One key challenge that have appeared with the increasing usage of wireless networks is the time synchronization. In this thesis, we introduce an averaging based synchronization algorithm in continuous time, and discuss its stability and average consensus in networks. Examples of wireless network application areas where synchronization is significant is as below [1]:

(i) Military applications: Monitoring friendly forces, equipment and ammunition; battlefield surveillance; reconnaissance of opposing forces and terrain; targeting; battle damage assessment; and nuclear, biological and chemical attack detection and reconnaissance.

(ii) Environmental applications: Forest fire monitoring, flood detection, and earthquake detection; monitoring ecological and biological habitats; monitoring environmental conditions that affect crops and livestock; macro-instruments for large scale earth monitoring and planetary exploration.

(iii) Commercial applications: Building virtual keyboards; managing inventory control; monitoring product quality; constructing smart office spaces; and environmental control in office buildings.

1.1. Synchronization of Wireless Networks. Why is it essential?

Synchronization can be defined as achieving and maintaining coordination among independent local agents via exchanging local time information to keep a system in unison [2]. In other words it is the task of achieving a common notion of time for the nodes that constitute a wireless network.

It is not technically possible to manufacture two network nodes that are identical to each other. This necessitates the use of synchronization algorithms which is achieved

by the exchange of local time information within wireless networks. In addition to the inherent manufacturing differences, clocks may deviate from each other due to several external reasons. They may drift due to environment changes, such as temperature, pressure, battery voltage, etc. Initial synchronization of the network may not be properly conducted. In addition, the outcomes of events on specific nodes may negatively affect the clock value. For example, a node that is busy while transmitting a message may miss the clock interrupts [3].

Applications that involve the coordination of the communication, computation, sensing, and actuation of distributed nodes, an accurate and consistent sense of time is vital for wireless networks. Examples for wireless network tasks where synchronization is significant is as below [4]:

(i) Time-stamping measurements: In data collection applications of wireless networks, readings from different nodes may only be meaningful with supporting time stamps in addition to the location information.

(ii) In-network signal processing: Time stamps are needed to determine which information from different sources can be fused or aggregated within the network. For example, signal processing algorithms for tracking unknown phenomena or targets require consistent and accurate time information from the nodes of the network.

(iii) Localization: Similar to target tracking, time synchronization is required for node localization, such as time-of-flight and time-difference-of-arrival based techniques.

(iv) Cooperative Communication: Cooperative communications requires multiple transmitters to transmit in-phase signals to a given receiver. This approach that enables significant energy savings is only possible with tight time synchronization.

(v) Medium access: Time-division-multiple-access based medium access schemes require the nodes to be synchronized in time so that they can be assigned communication slots without collision.

(vi) Sleep scheduling: Sleep scheduling is a significant approach for energy saving. Synchronization is a key enabler to coordinate the sleep schedules of neighboring nodes, so that they can communicate with each other efficiently.

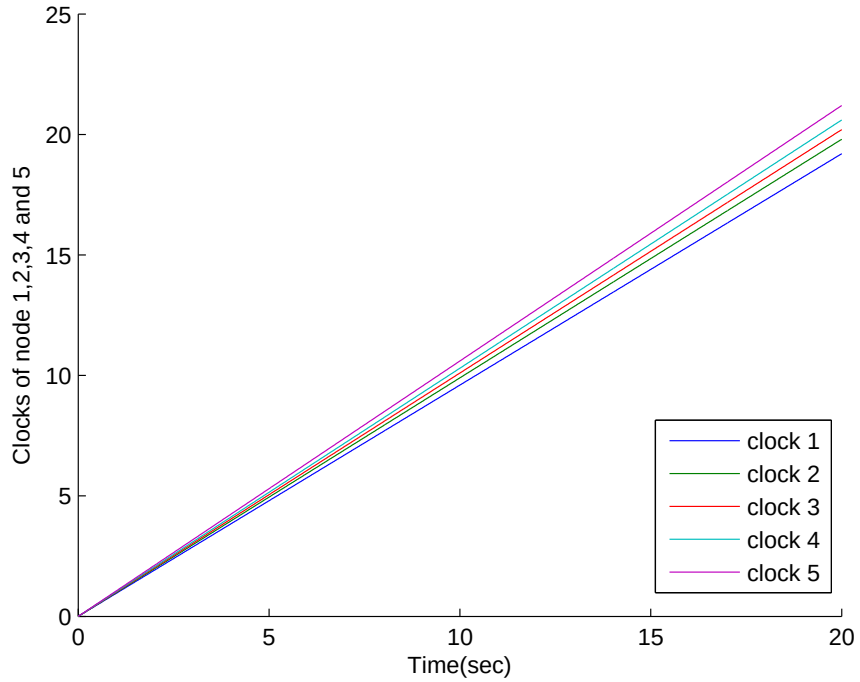

Figure 1.1. Clock values of nodes under the lack of synchronization.

(vii) Coordinated Actuation: Time synchronization is vital for networks which contains actuators in addition to sensors, in order to coordinate the actuators via distributed control algorithms.

In the lack of synchronization, local clocks of the nodes in a network may drift from each other seconds per day and in the long run add up to huge differences. Fig 1.1 shows an example for the clock values of nodes under the lack of synchronization. In this example, all clock values are reset at $t = 0$. However, different clock rates, that is, different free running frequencies, cause divergence of the clock values.

As a result, synchronization is essential in distributed networks. The errors that occur under the lack of synchronization, sooner or later, may cause serious failure of a network task.

1.2. Synchronization Algorithms

There are various approaches for time synchronization in distributed networks including external synchronization, internal synchronization and hybrid synchronization [5].

1.2.1. External Synchronization

In external synchronization approach, there is a global clock that sets the target clock value of the network. Every local clock synchronizes its value according to this master clock. An example of a global clock is the Universal Coordinated Time. This approach is also known as centralized synchronization. This approach has the highest complexity. Cristian's algorithm [6] and the Berkeley Algorithm [7] are examples of centralized synchronization approaches.

In Cristian's Algorithm [6], there is a global time server in the network. A node N in the network first requests the time from the time server. Receiving this request, the server prepares a response time T and sends it to the requesting node. The node N sets its time to $T + RTT/2$, where RTT is the round trip time.

Cristian's Algorithm can be applied to distributed computer applications, however, it only achieves synchronization if the RTT is small compared to the desired accuracy. Another shortcoming of this algorithm is the use of single server for distributed applications where redundancy is significant.

In Berkeley's Algorithm [7], a master node in the network is chosen based on an election process, e.g., Chang and Roberts's algorithm [8]. All other nodes in the network are assumed to be slaves. The master node collects the time stamps of the slaves. The master determines the round-trip time of the messages, estimates the time of each slave and its own and calculates the average time. While the average time is calculated, any node that has a clock value far from the average will be neglected. Based on the average calculation, the master determines the amount (positive or negative)

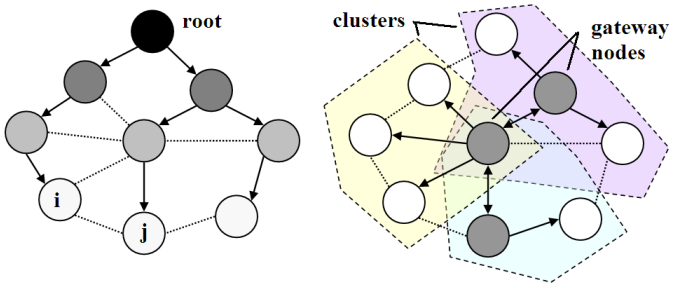

Figure 1.2. Distributed algorithm examples, TPSN and FTSP on the left and RBS on the right [9].

of adjustment that each clock is required to apply to correct its own clock. The master sends out these amount to each slave in the network. This approach avoids the uncertainty for the slave that is caused by the round-trip time.

1.2.2. Internal Synchronization

Contrary to external synchronization, in internal synchronization there is no global clock. Each local node gets clock information of neighboring nodes, applies a synchronization algorithm that uses these values, and updates its own value based on this algorithm. This approach can be used to actively maintain a common time in a distributed network if only internal correctness within a network is required. This approach is also known as distributed synchronization [10].

Examples of this approach are Time-synchronization protocol for Sensor Networks (TPSN) [11], Flooding Time Synchronization Protocol (FTSP) [12], Reference Broadcast Synchronization (RBS) [13], and Reachback Firefly Algorithm (RFA) [14].

In the TPSN approach, a wireless network is organized as a rooted tree. One node is initially elected to be the reference clock of the network, in other words, the root of the tree. After the root node, branches of the tree are built covering all the nodes in the network. By a two-way message exchange, the offset between two nodes (in adjacent levels of the tree) is calculated. This approach bounds the transmit and propagation delay. The offset of any node with respect to the root node is calculated by adding the offset of the edges in the unique path from each node to the root as shown in

Figure 1.2. This approach has two disadvantages. Firstly, if the root node dies, another selection is required to determine the new reference node of the network. This increases the complexity of the code and requires additional period of time for the network to be synchronized again after a new root node is elected. Secondly, in this approach, geographically close nodes, such as node i and node j in the Figure 1.2 might be far in terms of the tree distance. This is a disadvantage for applications where clock errors between one node and the other should decrease sufficiently smooth with distance [11]. Examples of such applications are object tracking and TDMA scheduling.

FTSP approach is topologically similar to TPSN approach, however, FTSP uses broadcast communication and MAC layer time-stamping and hence achieves better precision. In addition, FTSP considers not only clock value synchronization, but also skew compensation, dynamic topology adaptation and root failure recovery [12].

In RBS approach, a wireless network is divided into interconnected single-hop clusters. Within every single-hop cluster a reference node is selected to synchronize all the other nodes in that cluster. The reference nodes of different clusters are synchronized with respect to each other. After they are synchronized, the local nodes in each cluster are synchronized with respect to the reference node of that cluster. Similar to the TPSN approach, the disadvantages of the RBS are the complexity to divide the network into clusters and to elect the reference nodes. It is also fragile to node failures [13].

The final example of internal synchronization, RSA approach differentiates itself from the examples explained above. The RSA approach is inspired by firefly synchronization mechanism. Every node in the network periodically broadcasts a synchronization message to the other nodes in the network. When nodes hear a message, they increase the phase of their internal clock, which schedules message broadcasting, by a small quantity. All nodes eventually increase their phase until they are all synchronized, i.e., they broadcast a message at the same time. This approach does not consider clock skew, thus, the firing period needs to be rather small [14].

1.2.3. Hybrid Synchronization

Internal and external synchronization can be used in a combined, and hybrid way. For example, the gateway nodes in RBS approach are synchronized periodically with respect to an external source of time, and the rest of the nodes in each cluster of the network are synchronized with respect to their gateway node. This approach maintains logical correctness over longer periods of time [5].

1.3. Organization of the Thesis

In this thesis, we introduce an averaging based synchronization algorithm [15], apply it as a continuous time synchronization algorithm for distributed systems and investigate its convergence for fixed and varying topologies, and time delay. So far, we have presented a general literature review. Previous studies for time synchronization in wireless networks either assume a rooted tree structure or a union of interconnected single-hop clusters or distributed structure [11], [12], [13]. The former two structures necessitates the selection of reference nodes or gateways and they are not robust to node failures. In distributed topology approach there are no such special nodes, hence it is robust to node failure or new node appearance on this sense.

In Chapter 2, we introduce the relevant definitions and theorems from graph theory that are necessary for the analysis of the proposed distributed continuous-time synchronization algorithm. In Chapter 3, we discuss the stability and average consensus of the proposed algorithm in the presence of free running frequencies for fixed and time varying networks. In Chapter 4, we investigate the synchronization of the networks with time delay. In Chapter 5, we investigate the proposed algorithm for a network with faulty nodes. Finally, some concluding remarks are given in Chapter 6.

2. MATHEMATICAL SYSTEM DESCRIPTION AND PRELIMINARIES

In this chapter, we will discuss the mathematical background related to the synchronization algorithm that is the focus of this thesis. As explained in the previous chapter, local clock values may drift away from each other due to both internal (inherent frequency of the clock) and external (temperature, battery, etc. of the clock) reasons. If the clocks are allowed to run freely without applying control, the resultant clock values would be as in Figure 1.1. In order for the network to successfully fulfill its tasks that require time information, synchronization is necessary. In this thesis, we will carry out a continuous-time analysis of the averaging based synchronization algorithm in which the nodes in a network use both their own values and values of the neighboring nodes to synchronize their clocks.

2.1. Averaging Based Synchronization Algorithm In Continuous-Time

The mathematical model of the synchronization algorithm can be stated as below [15]:

$$\dot{p}_i(t) = F_i(t) + \gamma_i \sum_{j \in N_i} w_{ij}(t)[p_j(t) - p_i(t)] \tag{2.1}$$

This equation indicates that the concerned network consists of n nodes numbered as $i = 1, ..., n$ and $N_i \subseteq \{1, ..., n\}$ denotes the set of neighbors of node i. Each node calculates the difference between its clock value and neighboring clock values, and takes a weighted average of these differences to update its value. The description of the variables in (2.1) is given in Table 2.1.

If no synchronization were applied, then each clock would run at its own free running frequency. When synchronization is applied, each clock takes a proportion of

Table 2.1. Variables of the averaging based synchronization algorithm.

n	The number of nodes in the network, $i = 1, ..., n$.
$\dot{p}_i(t)$	Instantaneous frequency of the i-th node at time t.
$p_i(t)$	Clock value of that node at time t
$F_i(t)$	Free running frequency of each node
γ_i	Proportional control parameter
$w_{ij}(t)$	Weighing coefficient of the clock value sent from node j to node i

the weighted average of differences between its value and neighboring clock values. If the weighing coefficients, w_{ij}'s, of a network are constant over time, then it is a *fixed topology* network. If they are changing over time, then it is called a *switching network*, or a network with *time varying topology*. Throughout the thesis, we will assume that the weighing coefficients satisfy the following assumptions.

Assumption 2.1.1.

$$\begin{aligned} &(i) \quad w_{ij}(t) \geq 0, \ for\ all\ i,\ j,\ t. \\ &(ii) \quad w_{ij}(t) = 0, \ for\ all\ i,\ j\ if\ there\ is\ no\ connection\ in\ between. \\ &(ii) \quad \sum_{j \in N_i} w_{ij}(t) = 1, \ for\ all\ i. \end{aligned} \tag{2.2}$$

Assumption 2.1.1(i) is related to the network connectivity and ensures that the data are received from neighboring nodes with positive weights. Assumption 2.1.1(ii) indicates that if there is no connection between any two nodes then the related weighing coefficient is zero. Assumption 2.1.1(iii) requires that the weighing coefficients sum up to one for each node i in the network.

The averaging based synchronization algorithm is distributed in the sense that there is no central or external source of time that acts as a master node. Each node in the network uses the clock values of its neighbor nodes to update its own value. Let us illustrate this with a simple example.

Example 2.1.1. Assume a three node network where node two receives clock values from node one, and node three receives from node two. In this case, $N_1 = \{1\}$, $N_2 = \{1, 2\}$, $N_3 = \{2, 3\}$. Let the weighing coefficients be: $w_{11} = 1$, $w_{21} = \frac{1}{2}$, $w_{22} = \frac{1}{2}$, $w_{32} = \frac{1}{2}$, $w_{33} = \frac{1}{2}$. Due to the given connectivity of the network, we have $w_{12} = w_{13} = w_{23} = w_{31} = 0$. The synchronization equations for each node is as below. Note that each node uses information only from neighboring nodes.

$$\begin{aligned} \dot{p}_1(t) &= F_1(t) \\ \dot{p}_2(t) &= F_2(t) + \frac{1}{2}(p_1(t) - p_2(t)) \\ \dot{p}_3(t) &= F_3(t) + \frac{1}{2}(p_2(t) - p_3(t)) \end{aligned} \tag{2.3}$$

The synchronization algorithm in (2.1) can be reformulated in matrix form as:

$$\dot{p}(t) = F(t) - \Gamma(I - W(t))p(t) \tag{2.4}$$

where $F(t) = [F_1(t), F_2(t), ..., F_n(t)]^T$, and $p(t) = [p_1(t), p_2(t), ..., p_n(t)]^T$; I is an $n \times n$ identity matrix; $\Gamma = diag\{\gamma_1, \gamma_2, ..., \gamma_n\}$ is a diagonal matrix that holds the proportional control parameters in its diagonals. $W(t) = [w_{ij}(t)]_{n \times n}$ in (2.4) is called the *averaging matrix*, where each entry corresponds to one of the weighing coefficients of the synchronization algorithm (2.1).

2.2. Graph Representation

When a network is under investigation, graphical representations are widely used. In this thesis, we will also utilize graphical representations to represent the relations between the nodes and data flow in the network.

Graphs are vastly used to model a binary relationship between the objects from a certain collection. Hence they will be used to represent the nodes or agents of a network and model the communication between them. For this reason, it is essential

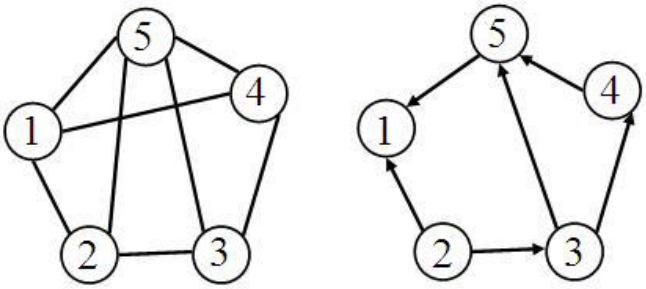

Figure 2.1. Example of an undirected (on the left) and a directed (on the right) graph.

to state the basics of the Algebraic Graph Theory that is used in this thesis [16].

A graph G is a collection of *vertices* and *edges*, where an edge connects pairs of vertices. The set of vertices corresponds to the set of nodes from 1 to n in a network and is denoted by $V = \{v_1, ..., v_n\}$. The set of edges corresponds to the communication between vertices, and is a two element subset of V, that is, E $\subseteq$ V $\times$ V. If an edge connects v_1 and v_2, then it means v_1 and v_2 are adjacent or v_2 is a neighbor of v_1.

A graph can be *undirected* or *directed* as shown in Figure 2.1 . Undirected graphs imply symmetrical communication between its nodes while in directed graphs, the communication can be asymmetric and the direction of communication is shown by arrows. In a directed graph, an edge is represented as $e_{ij} = (v_i,v_j)$ which implies data flow from node j to node i. The neighbors of a node v_i is represented as N_i which contains all $v_j \in V$ that communicate with node v_i.

Definition 2.2.1. A directed graph is called *strongly connected* if there exists a path that connects any two distinct nodes following the direction of the edges.

There are types of matrices that are naturally associated with graphs such as *adjacency matrix* and the *Laplacian*. Adjacency matrix is a way to represent which vertices of a graph are adjacent to, or neighbor to which vertices, and is defined as below.

Definition 2.2.2. *An adjacency matrix* A of the graph $G = (V, E)$ is an $n \times n$ matrix $A = [a_{ij}]$. In a graph with $0 - 1$ adjacency, $a_{ij} = 1$ if there is communication between

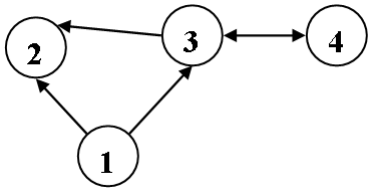

Figure 2.2. The graph representation of a four node network

node j to node i, and $a_{ij} = 0$ if there is no communication.

Definition 2.2.3. The *in-degree* and *out-degree* of node v_i in a directed graph is defined as:

$$deg_{in}(v_i) = \sum_{j=1}^{n} a_{ij} \qquad deg_{out}(v_i) = \sum_{j=1}^{n} a_{ji} \tag{2.5}$$

Definition 2.2.4. *Degree matrix* of a directed graph G is an $n \times n$ diagonal matrix $\Delta = [\Delta_{ij}]$, where $\Delta_{ij} = 0$ for all $i \neq j$ and $\Delta_{ij} = deg_{in}(v_i)$ for all $i = j$.

Definition 2.2.5. For a directed graph with $0-1$ adjacency elements, the graph *Laplacian* is defined as:

$$L(G) = \Delta - A \tag{2.6}$$

It can also be shown as:

$$l_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} a_{ik} & \text{if } j = i \\ -a_{ij} & \text{if } j \neq i \end{cases} \tag{2.7}$$

Example 2.2.1. Let the topology of a network be as given in Figure 2.2. For this network, the adjacency matrix, diagonal matrix, and Laplacian are as shown below:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \ \Delta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \ L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{2.8}$$

As the definitions imply, the sum of the values of each row of the Laplacian matrix equals to zero. Therefore, the Laplacian matrix always has an eigenvalue that equals to zero. This corresponds to a right eigenvector $w_r = [1, 1, ..., 1]^T$ with identical nonzero elements. This means that the rank of the Laplacian is, $rank(L) \leq n - 1$. In the case of strongly connected graphs, we have the following result.

Theorem 2.2.1. *[17] If a directed graph G of n nodes with a weighted adjacency matrix A is strongly connected, then the rank of its corresponding Laplacian matrix is n - 1.*

However, it must be noted that the converse of the statement in Theorem 2.2.1 does not hold. In other words, a Laplacian with $rank(L) = n - 1$ does not necessarily imply the strong connectivity of the corresponding graph.

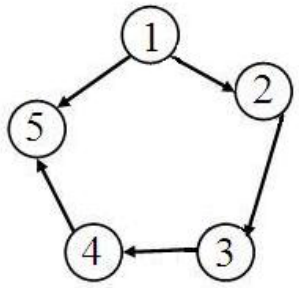

Figure 2.3. The graph representation of a five node network.

Example 2.2.2. The Laplacian that corresponds to the network in Figure 2.3 is

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}. \tag{2.9}$$

The rank of the Laplacian is, $rank(L) = 4$, where $n = 5$. However, as it can be seen in Figure 2.3, the network is not strongly connected.

Theorem 2.2.2. *(Gershgorin circle theorem [17]) Every eigenvalue of an n×n matrix*

A, lies within at least one of the Gershgorin circles defined by:

$$D(a_{ii}, R_i) = \left\{ z \in C : |z - a_{ii}| \leq \sum_{j=1, j \neq i}^{n} |a_{ij}| \right\} \tag{2.10}$$

where a_{ii} *is the* $i-th$ *diagonal element of* A.

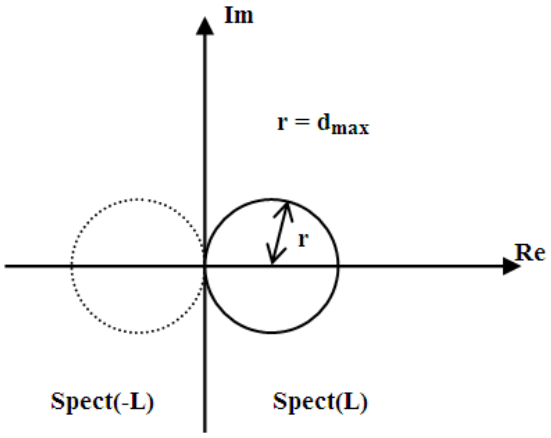

Figure 2.4. Gershgorin Theorem applied to Graph Laplacian

Applying Gershgorin circle theorem to the Laplacian L of the directed graph G, all the eigenvalues of L are located in the union of the following n disks:

$$D_i = \left\{ z \in C : |z - l_{ii}| \leq \sum_{j=1, j \neq i}^{n} |l_{ij}| \right\}. \tag{2.11}$$

It must be noted that for the directed graph G, $l_{ii} = \Delta_{ii}$, and $\sum_{j=1, j \neq i}^{n} |l_{ij}| = deg_{in}(v_i) = \Delta_{ii}$. It follows from Theorem 2.2.2 and the definitions above that

$$D_i = \{ z \in C : |z - \Delta_{ii}| \leq \Delta_{ij}| \}. \tag{2.12}$$

All these n disks are contained in the largest disk $D(G)$ with radius $d_{max}(G)$.

Up until now, we have investigated the L matrix. As shown in Figure 2.4, the

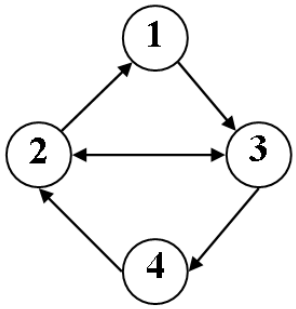

Figure 2.5. A strongly connected and balanced network with four nodes.

eigenvalues of the matrix $-L$ are located in the disk

$$D_G^{'} = \{z \in C : |z + d_{max}(G)| \leq d_{max}(G)|\}\,. \tag{2.13}$$

Definition 2.2.6. A node v_i of a directed graph G is *balanced* if and only if its in-degree and out-degree are equal, that is, $deg_{out}(v_i) = deg_{in}(v_i)$. A graph is called a balanced graph if and only if all of its nodes are balanced. It can also be stated as, for all i,

$$\sum_j a_{ij} = \sum_j a_{ji}. \tag{2.14}$$

Example 2.2.3. The adjacency and Laplacian matrices corresponding to the graph in Figure 2.5 are as below:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},\ L = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{2.15}$$

Theorem 2.2.3. *[17] Given a network G with fixed topology and an adjacency matrix A. The network is balanced if and only if $w_l = \boldsymbol{e}$, that is the unit vector $e = [1, 1, ..., 1]^T$ is the left eigenvector of the Laplacian L of G, that is, $\boldsymbol{e}^T \boldsymbol{L} = \boldsymbol{0}$.*

The proof of the above result proceeds as follows [17]. We know that $\Delta_{ii} = deg_{in}(v_i)$

and $deg_{out}(v_i) = \sum_{j,j\neq i} a_{ji}$, thus the $i-th$ column sum of L is equal to zero, or $\sum_j l_{ji} = \sum_{i,j\neq i} l_{ji} + l_{ii} = -deg_{out}(v_i) + deg_{in}(v_i) = 0$, if and only if node v_i of G is balanced. Noting that the $i-th$ column sum of L equals to the $i-th$ element of the row vector $\mathbf{e}^T\mathbf{L}$, it is concluded that $\mathbf{e}^T\mathbf{L} = \mathbf{0}$ if and only if all the nodes of G are balanced, that is G is balanced.

2.3. Stability Analysis of $\dot{p}(t) = -Lp(t)$

In the study of the distributed synchronization algorithm, it is important to understand how the time-varying system $\dot{p}(t) = -Lp(t)$ behaves over time, where $L = \Delta - A$ and L is the Laplacian of the network. If the network is strongly connected, then based on Theorems 2.2.1 and 2.2.2, the linear system representing the network that applies the the averaging based synchronization algorithm is stable and the algorithm solves the consensus problem.

The solution of the system in (2.1) with a fixed topology can be stated as

$$p(t) = e^{-Lt}p(0). \tag{2.16}$$

A limit theorem for exponential matrices is needed to address the convergence value of the system.

Theorem 2.3.1. *[17] Assume G is a strongly connected directed graph with the Laplacian L. w_r and w_l are the right and the left eigenvectors of the Laplacian L, respectively, associated with the eigenvalue $\lambda_1 = 0$. They satisfy $Lw_r = 0$; $w_l^T L = 0$. Then*

$$R = \lim_{t\to+\infty} e^{-Lt} = w_r w_l^T \in M_n \tag{2.17}$$

where M_n is the set of square $n \times n$ real matrices.

The proof of the above theorem as done in [17] is as follows. We assume $-L = A$ and take the Jordan Canonical Form of A such that $A = SJS^{-1}$. If we take the

exponential of A, then we have $e^{At} = Se^{Jt}S^{-1}$. As time goes to infinity, e^{Jt} converges to a matrix $Q = [q_{ij}]$ that has only one nonzero element $q_{11} = 1$ due to the isolated eigenvalue at zero. The other elements in the diagonal of $e^{At} = e^{Jt}$ converge to zero, because the real parts of the corresponding eigenvalues of A are negative. Thus, $R = SQS^{-1}$. Furthermore, the Jordan form is formed as $AS = SJ$ and $Aw_r = \lambda w_r$. This implies that the columns of S are right eigenvectors, w_r. Similarly $S^{-1}A = JS^{-1}$ and $w_l^T A = \lambda w_l^T$. This implies that the rows of S^{-1} are left eigenvectors, w_l^T. Hence, R is calculated as $R = w_r w_l^T$.

2.4. Analysis of Average Consensus

Until now, we have covered the convergence analysis of the algorithm and showed the value that the clocks will converge to. In this section, we focus on achieving average consensus. Average consensus is the case where the nodes in the network converges to a value which equals to the average of the initial values of the nodes. Achieving an average consensus is significant in distributed network problems such as coordination of mobile autonomous agents, and data fusion in sensor networks as well as clock synchronization [18].

Theorem 2.4.1. *[17] Given a strongly connected network G with fixed topology and an adjacency matrix A. The network reaches an average consensus if and only if $\boldsymbol{e}^T \boldsymbol{L} = \boldsymbol{0}$.*

From Theorem 2.3.1, we know that the system will converge to $CV = w_r w_l^T x(0)$. It follows from Theorem 2.2.3 and Theorem 2.4.1 that if a network is strongly connected and balanced, then it achieves average consensus, that is, the clock values of the nodes will converge to the average of the initial clock values.

Example 2.4.1. Consider the strongly connected and balanced graph given in Example 2.2.3. The convergence of the system based on the algorithm given by (2.1) that omits the free-running frequencies, with initial condition $p(0) = [10,\ 5,\ 0,\ 3]^T$ is shown in Figures 2.6 and 2.7, where the consensus value of the system is 4.5 as expected.

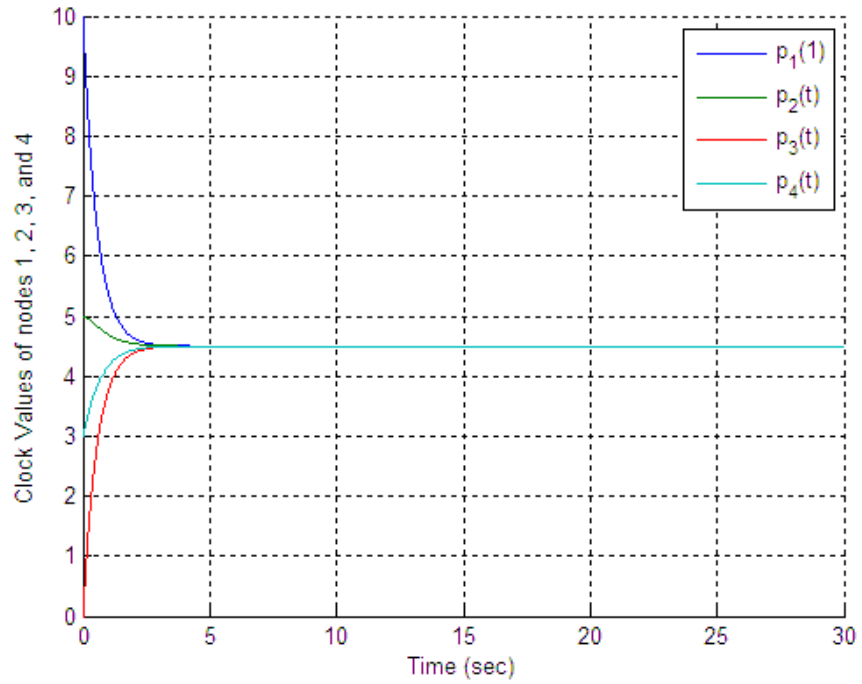

Figure 2.6. The clock values of the network when synchronization algorithm is applied omitting the free running frequencies.

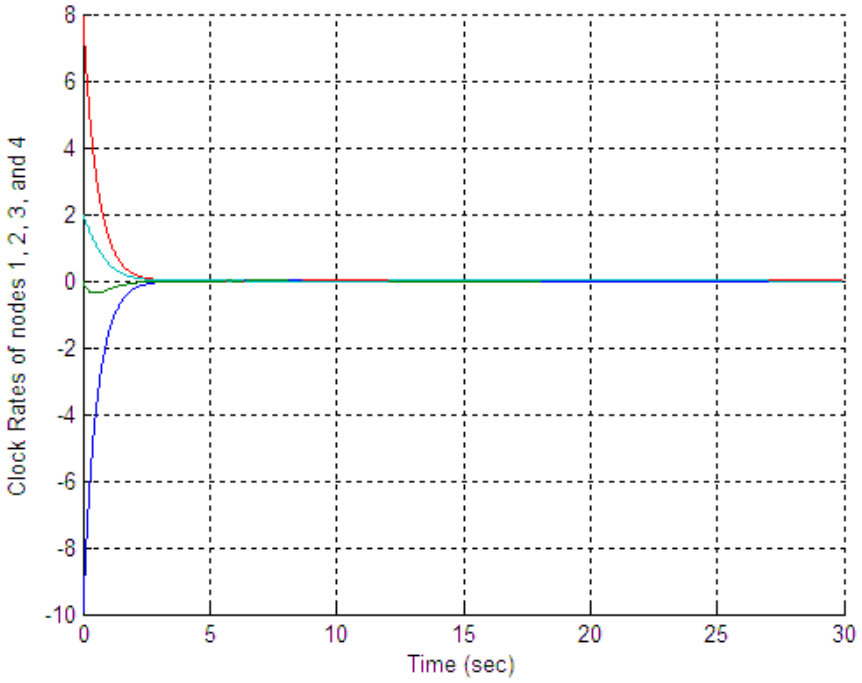

Figure 2.7. The clock rates of the network when synchronization algorithm is applied omitting the free running frequencies.

2.5. Stability Analysis of $\dot{p}(t) = -L(t)p(t)$ and Average Consensus

In the previous section, we assumed a fixed network topology corresponding to constant weighing coefficients. However, in real life applications, the communication links between nodes may not stay fixed at all times. The change in the network topology may be caused by the node itself or the environment. If a node is a mobile agent in the network, then it may go out of the communication range and cause a communication link to fail, or conversely come close enough so that a new link is created. In addition, since the wireless network is not protected against external interference, this may lead to high level of message loss, hence link failures [19]. In order to account for these kind of topology changes, it is important to understand how the time-varying system $\dot{p}(t) = -L(t)p(t)$ behaves over time, where $L(t)$ reflects the time varying property of a switching network.

The communication links between nodes will change by time, and with every change a new network topology will be valid for the system. The behavior of the system can be described as switching from one topology to the other with every change in the network. To show the convergence in case of a switching topology, we will assume the system will switch among a set of network topologies represented by $\mathcal{G} = \{G_1, G_2, ..., G_k, ...\}$ where each topology, G_k, is strongly connected and balanced. Under this assumption, our synchronization algorithm becomes:

$$\dot{p}(t) = -\text{Ł}(G_k)p(t) \tag{2.18}$$

where k represents the topology that the system has switched to at time t.

In order to show the stability of the network, the authors of [17] stated the disagreement dynamics of the system and defined the Lyapunov function, that is smooth, positive-definite, and proper. The synchronization of the network implies that the network will reach a consensus, hence the clocks will reach a common value. This indicates

that the clock values, $p(t)$, can be decomposed as below

$$p(t) = \rho\mathbf{e} + \delta(t) \tag{2.19}$$

where ρ represents the consensus value for the clocks of the network, $\mathbf{e}= [1, 1, ..., 1]^T$ is the unit vector, and $\delta(t)$ is the disagreement vector of the network. If we take the derivative of (2.19), we obtain

$$\dot{p} = \dot{\delta}. \tag{2.20}$$

Inserting (2.18) and (2.19) in (2.20)

$$\dot{\delta} = -L(G_k)(\rho\mathbf{e} + \delta) = -\rho L(G_k)\mathbf{e} - L(G_k)\delta. \tag{2.21}$$

Noting that $L(G_k)\mathbf{e} = 0$, the disagreement dynamics of the network can be represented as

$$\dot{\delta} = -L(G_k)\delta. \tag{2.22}$$

We define the smooth, positive-definite, and proper Lyapunov function as a function of δ as below

$$V(\delta) = \frac{1}{2}\|\delta\|^2. \tag{2.23}$$

If we take the derivative of V with respect to time, then we obtain

$$\begin{aligned} \dot{V} &= -\delta^T L(G_k)\delta = -\delta^T L(\hat{G}_k)\delta \\ &\leq -M^*(L(\hat{G}_k))\|\delta\|^2 = -2M^*(L(\hat{G}_k))V < 0 \end{aligned} \tag{2.24}$$

where $\hat{G}_k$ represents the mirror graph of G_k, and $M^* = \min_{\mathcal{G}} \lambda_2(L(\hat{G}_k))$ represents the minimum λ_2, the second largest eigenvalue, of the Laplacian $L_s(\hat{G}_k)$ among G_k's from

the set of strongly connected and balanced graphs.

This shows that $V(\delta)$ is a valid Lyapunov function and consensus is achieved. Analogous to the previous section, since all of the topologies that the network switches in between are strongly connected and balanced, each of them will force the network to converge to the average of the initial values when the proposed algorithm is applied. Hence, networks with time varying topologies achieves an average consensus as long as the topologies that the network is switching between are strongly connected and balanced [17].

2.6. Summary Of The Chapter and Concluding Remarks

In this chapter, we first introduced the averaging based synchronization algorithm in continuous-time and defined its parameters. The clock values of any two nodes in a network may differ from each other in time. Averaging based synchronization algorithm provides distributed synchronization where each node uses the clock values of neighbor nodes to update its running frequency. In order to analyze the networks, graph theory is widely used. In the second part of this chapter, we introduced the graphical representation of networks, definitions and theorems that proved useful in the stability analysis such as the Laplacian matrix of a network. In the third part, we discussed the stability of the algorithm for networks with fixed topologies when free running frequencies are omitted using the results of [17]. The network is required to be strongly connected to achieve consensus. In the fourth part, we discussed the average consensus for the fixed topology networks, that is, the consensus value for the network is exactly the average of the initial values of the nodes. The network is required to be balanced if the consensus value is desired to be the average of the initial values [17]. In the fifth part, we discussed the stability and average consensus for networks with time-varying topology. Analogous to the networks with fixed topology, the network can achieve average consensus if and only if it switches between topologies that are all strongly connected and balanced. In the next chapter, we will analyze the stability of the proposed averaging based synchronization algorithm in the presence of free running frequencies for both fixed and time varying topologies.

3. ANALYSIS OF THE AVERAGING BASED SYNCHRONIZATION ALGORITHM

In this chapter we will analyze the convergence of the averaging based synchronization algorithm in the presence of free running frequencies. In the previous section, we have only focused on the dynamics of the averaging based synchronization algorithm and omitted the F_i's, the inherent free running frequencies of the nodes in the network. Free running frequency is the internal frequency at which a local clock would operate in the absence of a synchronization algorithm. The algorithm provides an input to each clock to deviate their free running frequencies to consensus, a common frequency value.

The averaging based synchronization algorithm in the presence of free running frequency F in (2.4), can be reformulated with the Laplacian as below, where we assume the proportional control input $\Gamma = I$

$$\dot{p}(t) = -L(t)p(t) + F(t). \tag{3.1}$$

In order to investigate the stability of the averaging based synchronization algorithm including the free-running frequencies, we will benefit from Lyapunov stability.

3.1. Stability Analysis of The Algorithm for Fixed Topology Networks

For the stability analysis of the averaging based synchronization algorithm (3.1), we first focus on the case where all the neighbor relations between nodes are preserved over time throughout the network. This indicates that the Laplacian of the network is time invariant, i.e., $L(t) = L$ is fixed over time.

As the synchronization of the network implies that the network will reach a consensus, the deviation between clock values will decrease by time and clocks will run

with a common clock rate. This indicates that, similar to the previous chapter, the clock values, $p(t)$, can be decomposed as below

$$p(t) = \alpha t\mathbf{e} + \delta(t) \tag{3.2}$$

where α represents the consensus value for the clock rates of the network. If we take the derivative of (3.2), we obtain

$$\dot{p} = \alpha\mathbf{e} + \dot{\delta} \Rightarrow \dot{\delta} = \dot{p} - \alpha\mathbf{e}. \tag{3.3}$$

Inserting (3.1) and (3.2) in (3.3)

$$\dot{\delta} = -L(\alpha t\mathbf{e} + \delta) + F - \alpha\mathbf{e} = -\alpha tL\mathbf{e} - L\delta + F - \alpha\mathbf{e}. \tag{3.4}$$

By letting $F' = F - \alpha$ in (3.4), and noting that $L\mathbf{e} = 0$, the disagreement dynamics of the network can be represented as

$$\dot{\delta} = -L\delta + F'. \tag{3.5}$$

In order to investigate the stability of the network, we need to define a Lyapunov function. Below is a valid, smooth, and positive-definite Lyapunov function.

$$V(\delta) = \frac{1}{2}\|\delta\|^2 \tag{3.6}$$

If we take the derivative of $V(\delta)$ with respect to time, we obtain

$$\begin{aligned} \dot{V}(\delta) &= \frac{1}{2}\dot{\delta}^T\delta + \frac{1}{2}\delta^T\dot{\delta} = \frac{1}{2}(-L\delta + F')^T\delta + \frac{1}{2}\delta^T(-L\delta + F') \\ &= \frac{1}{2}(-\delta^T L^T\delta + F'^T\delta - \delta^T L\delta + \delta^T F') = -\frac{1}{2}\delta^T(L^T + L)\delta + F'^T\delta \end{aligned} \tag{3.7}$$

At this point, if the network is balanced, we benefit from the following equality:

$$L_s = \frac{1}{2}(L + L^T) \tag{3.8}$$

where L_s is the symmetric part of the Laplacian L, representing the mirror graph of L, that is, it has the reverse edges in addition to the existing nodes and edges of the graph that L represents. Inserting (3.8) in (3.7)

$$\begin{aligned} \dot{V}(\delta) &= -\delta^T L_s \delta + F'^T \delta = -\delta^T \hat{L} \delta + F'^T \delta \\ &\leq -\lambda_2(\hat{G})\|\delta\|^2 + F'_{max}\|\delta\| \end{aligned} \tag{3.9}$$

where F'_{max} is the maximum of the differences between the free running frequencies and consensus frequency of clocks, and λ_2 is the second largest eigenvalue of L_s. In order to have $\dot{V}(\delta) < 0$:

$$\|\delta\| > \frac{F'_{max}}{\lambda_2(\hat{G})} \tag{3.10}$$

So long as the above inequality is satisfied, the system responds to a bounded input with a bounded output.

3.2. Analysis of Average Consensus in The Presence of the Free Running Frequencies In Fixed Topologies

In the previous section, we have showed that as long as the free running frequencies have bounded values, the system responds with bounded output. In this section we focus on the average consensus of the clock rates in the presence of the free running frequencies.

If we take the Laplace Transform of (3.1)

$$\begin{aligned} sP(s) - p(0) &= -LP(s) + F(s) \\ (sI + L)P(s) &= p(0) + F(s) \\ P(s) &= (sI + L)^{-1}p(0) + (sI + L)^{-1}F(s) \end{aligned} \tag{3.11}$$

If we assume that the free running frequencies are constant over time, then the Laplace Transform of F(t) is $F(s) = \frac{F}{s}$, and inserting it in (3.11)

$$P(s) = (sI + L)^{-1}p(0) + (sI + L)^{-1}\frac{F}{s} \tag{3.12}$$

If we take the inverse Laplace transform of (3.14), we obtain

$$\begin{aligned} p(t) &= e^{-Lt}p(0) + L^{-1}(\mathbf{e} - e^{-Lt})F \\ p(t) &= e^{-Lt}p(0) + L^{-1}F - L^{-1}e^{-Lt}F \end{aligned} \tag{3.13}$$

If we take the derivative of (3.13), we obtain $\dot{p}(t)$ as

$$\dot{p}(t) = -Le^{-Lt}p(0) + e^{-Lt}F \tag{3.14}$$

In order to find the consensus value of the clock rates, we need to take the limit of $\dot{p}(t)$ as time goes to infinity.

$$CR = \lim_{t\to\infty} \dot{p}(t) = \lim_{t\to\infty} \{-Le^{-Lt}p(0) + e^{-Lt}F\} \tag{3.15}$$

From Theorem 2.3.1, we know that $\lim_{t\to\infty} e^{-Lt} = w_r w_l^T$. Inserting this limit equality in (3.15), we obtain

$$CR = -Lw_r w_l^T p(0) + w_r w_l^T F \tag{3.16}$$

Since $Lw_r = 0$, the first term in (3.16) vanishes.

From Theorem 2.4.1, we know that the network, while free running frequencies are omitted, achieves average consensus if the network is strongly connected and balanced, and the solution of the system, $CV = w_r w_l^T x(0)$, gives the average of the initial clock values. Similarly, in the presence of the free running frequencies, the second term in (3.16), $w_r w_l^T F$, equals to the average of the initial clock rates which are the free running frequencies, if the network is strongly connected and balanced. Hence the alpha term in (3.2) becomes $\alpha \mathbf{e} = w_r w_l^T F$. As a result, the network achieves average consensus for its clock rates as long as it is strongly connected and balanced.

Example 3.2.1. Consider a network of five nodes represented by the strongly connected directed graph G shown in Figure 3.1 with the following adjacency matrix A.

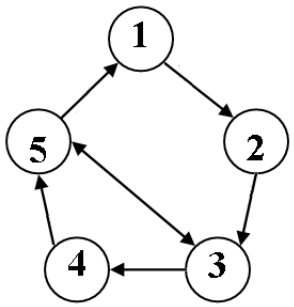

Figure 3.1. A strongly connected, and balanced network of five nodes.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \; L = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}. \tag{3.17}$$

The free running frequencies of the nodes are taken as $F = [0.98, 0.99, 1, 1.1, 1.2]^T$ and the initial condition is $p(0) = [0, 0, 0, 0, 0]^T$.

Figure 3.2 shows that the clock rates of the nodes in the network reach a consensus value, 1.054, which is the average of the initial free running frequencies. Figure 3.3

shows that the maximum difference between the clock rates of the nodes reaches zero over time.

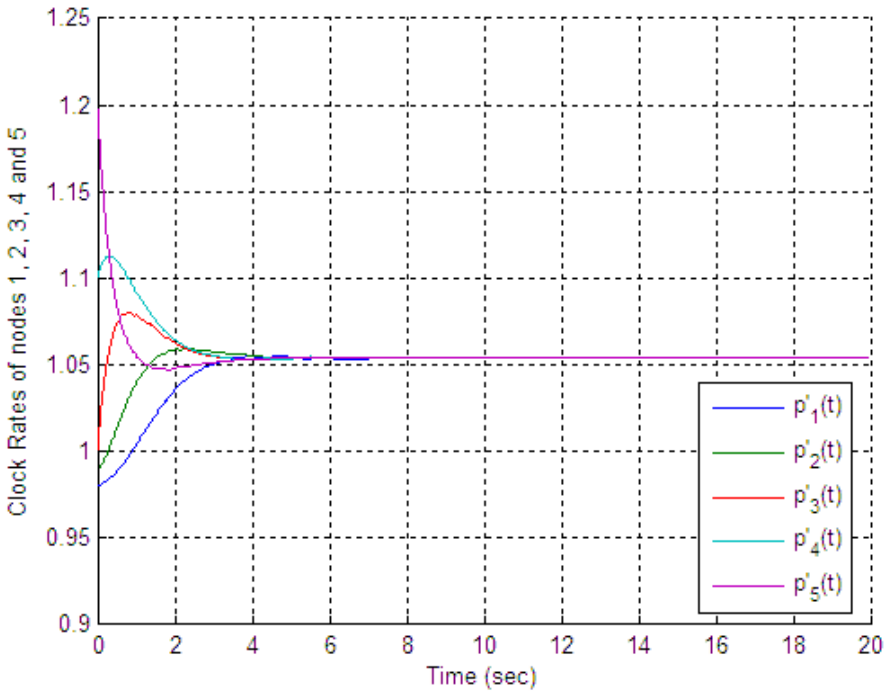

Figure 3.2. Synchronization of the clock rates of the nodes in a fixed topology network, in the presence of free running frequencies.

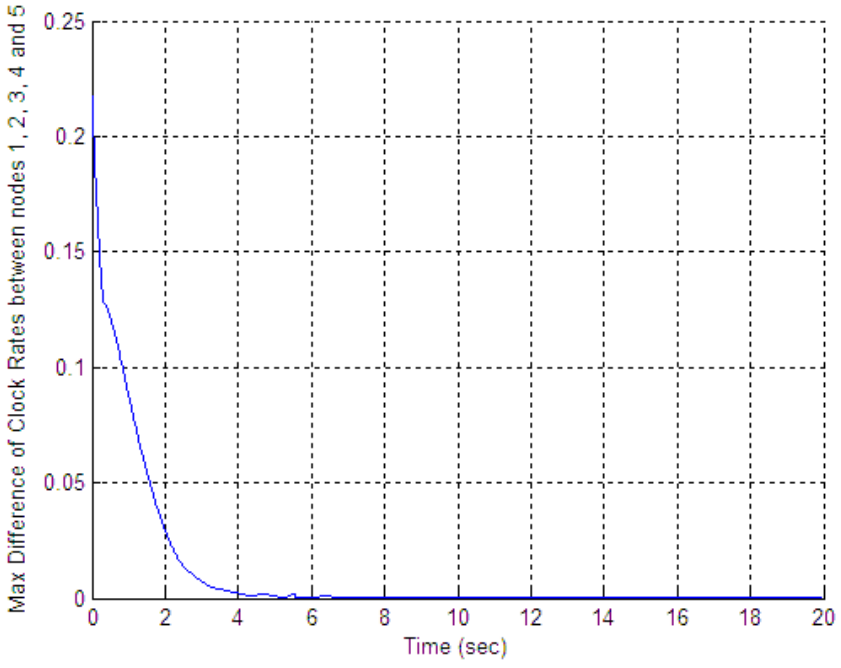

Figure 3.3. Maximum difference between the clock rates of the nodes in a fixed topology network.

Figures 3.4 and 3.5 show that as the network reaches a consensus, the maximum difference between the clock values of the nodes remain bounded over time.

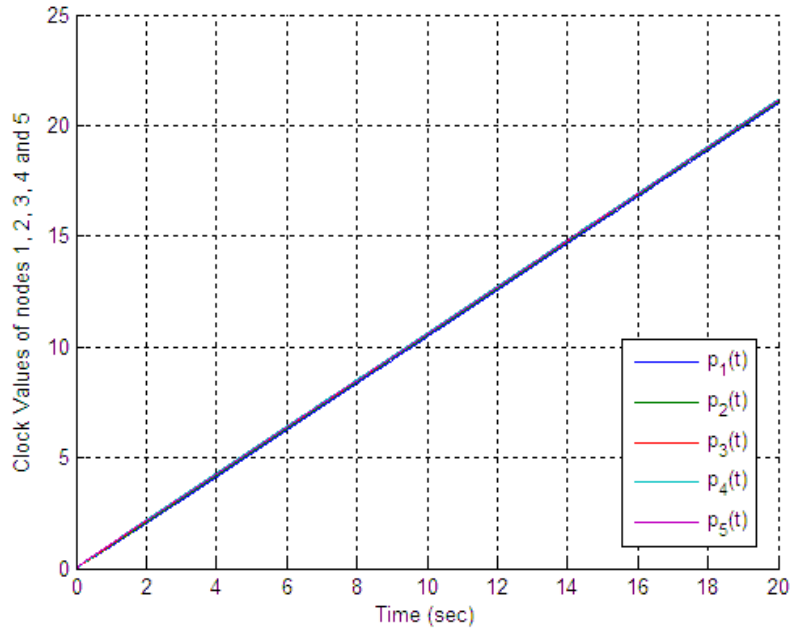

Figure 3.4. Synchronization of the clock values of the nodes in the fixed topology network.

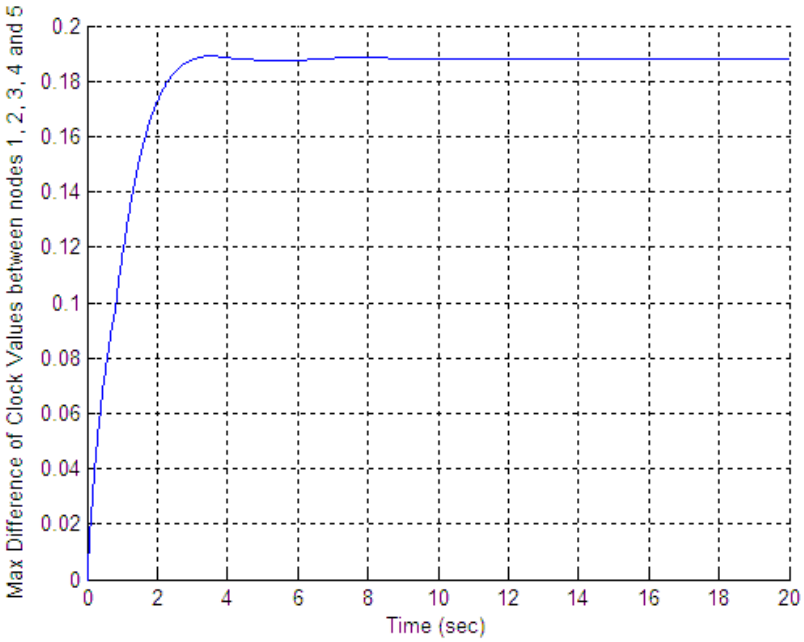

Figure 3.5. Maximum difference between the clock values of the nodes in the fixed topology network.

3.3. Stability Analysis of The Algorithm for Time Varying Topology Networks

In the previous section, we discussed the stability of the averaging based synchronization algorithm for fixed network topologies corresponding to a constant Laplacian. However, as mentioned in the previous chapter, in real life applications, the communication links between nodes may not stay fixed at all times and the Laplacian of the network changes over time according to the changes in the network.

We can redefine the averaging based synchronization algorithm to represent the network switching from one topology to another over time in the presence of free running frequencies as:

$$\dot{p}(t) = L(G_k)p(t) + F(t) \tag{3.18}$$

where G_k is from a set of graphs that are strongly connected and balanced and the network switches between these topologies.

Similar to the previous section, in order to analyze the stability of the algorithm, we will first derive the disagreement dynamics of the network and then utilize a Lyapunov function. Analogous to the fixed topology case, the disagreement dynamics can be represented as $\dot{\delta} = \dot{p} - \alpha\mathbf{e}$, where α represents the consensus value for the clock rates of the network, $\mathbf{e}$ is the unit vector, and δ is the disagreement vector of the network. If we insert (3.18) and (3.2) in (3.1), we obtain

$$\dot{\delta} = -L(G_k)(\alpha t\mathbf{e} + \delta) + F - \alpha\mathbf{e} = -\alpha tL(G_k)\mathbf{e} - L(G_k)\delta + F - \alpha\mathbf{e} \tag{3.19}$$

Substituting $F' = F - \alpha$ in (3.4), the disagreement dynamics of the network can be represented as

$$\dot{\delta} = -L(G_k)\delta + F' \tag{3.20}$$

We utilize the same Lyapunov function in (3.6). If we take the derivative of $V(\delta)$ with respect to time, we obtain

$$\begin{aligned}\dot{V}(\delta) &= \frac{1}{2}\dot{\delta}^T\delta + \frac{1}{2}\delta^T\dot{\delta} = \frac{1}{2}(-L(G_k)\delta + F')^T\delta + \frac{1}{2}\delta^T(-L(G_k)\delta + F') \\ &= \frac{1}{2}(-\delta^T L(G_k)^T\delta + F'^T\delta - \delta^T L(G_k)\delta + \delta^T F') \\ &= -\frac{1}{2}\delta^T(L(G_k)^T + L(G_k))\delta + F'^T\delta \end{aligned} \tag{3.21}$$

At this point, we again assume a balanced network and utilize the equality in (3.8) in order to obtain

$$\begin{aligned}\dot{V}(\delta) &= -\delta^T L_s(G_k)\delta + F'^T\delta = -\delta^T L_s(\hat{G}_k)\delta + F'^T\delta \\ &\leq -M^*\|\delta\|^2 + F'_{max}\|\delta\| \end{aligned} \tag{3.22}$$

where M^* is the minimum λ_2 of the Laplacian $L_s(G_k)$, among G_k's from a set of strongly connected and balanced graphs.

So long as

$$\|\delta\| > \frac{F'_{max}}{M^*} \tag{3.23}$$

we have $\dot{V}(\delta) < 0$, which implies that the system responds to a bounded input with a bounded output.

3.4. Analysis of Average Consensus in The Presence of Free Running Frequencies In Time Varying Topology Networks

In the previous section, we showed that as long as the input, that is, the free running frequencies of the network, is bounded, then the output remains bounded as well for networks that switch among a set of strongly connected and balanced topologies.

We can investigate the consensus value for the clock rates of the nodes separately

for each topology as if only that topology determines the connections between nodes over time. How the clock rates behave over time for each topology, g_k, can be shown as

$$\dot{p}(t) = -L(G_k)e^{-L(G_k)t}p(0) + e^{-Lt}F \tag{3.24}$$

Analogous to the fixed network case, if we take the limit of (3.24) as time goes to infinity, the clock rates of the nodes can be shown as

$$CR_{G_k} = w_{r_{G_k}} w_{l_{G_k}}^T F \tag{3.25}$$

Equation (3.25) shows that the clock rates over time for each topology, are independent of the network topology G_k, but dependent on the left and right eigenvectors of the Laplacian of the network. Since each topology is strongly connected and balanced, their Laplacian allows that the consensus value for the clock rates, CR_{G_k}, is the average of the free running frequencies. When the network switches among the set of G_k's, each topology will drive the clock rates to the average of the free running frequencies, hence, the network achieves average consensus.

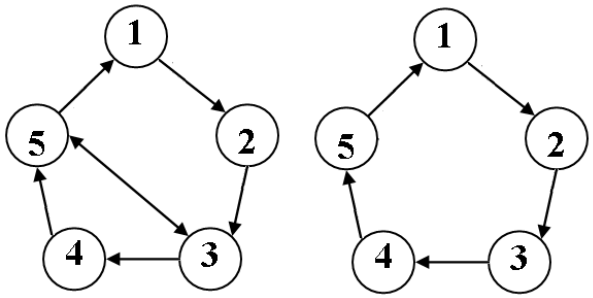

Figure 3.6. Switching between two strongly connected and balanced topologies.

Example 3.4.1. A network with five nodes is simulated between two strongly connected and balanced topologies as in Figure 3.6. The free running frequencies of the nodes are, $F = [0.98, 0.99, 1, 1.1, 1.2]^T$ and the initial clock values are, $p\,(0) = [0, 0, 0, 0, 0]^T$. As it can be seen from the results graph, the clock rates of the nodes in the network reaches a consensus value, 1.054, which is the average of the initial free running frequencies.

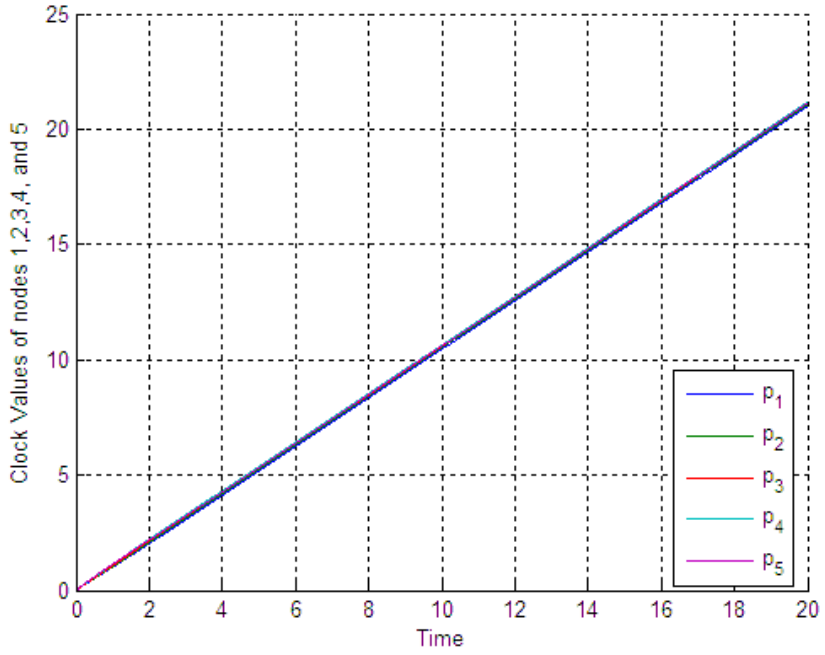

Figure 3.7. Synchronization of the clock values in a network switching between two strongly connected and balanced topologies.

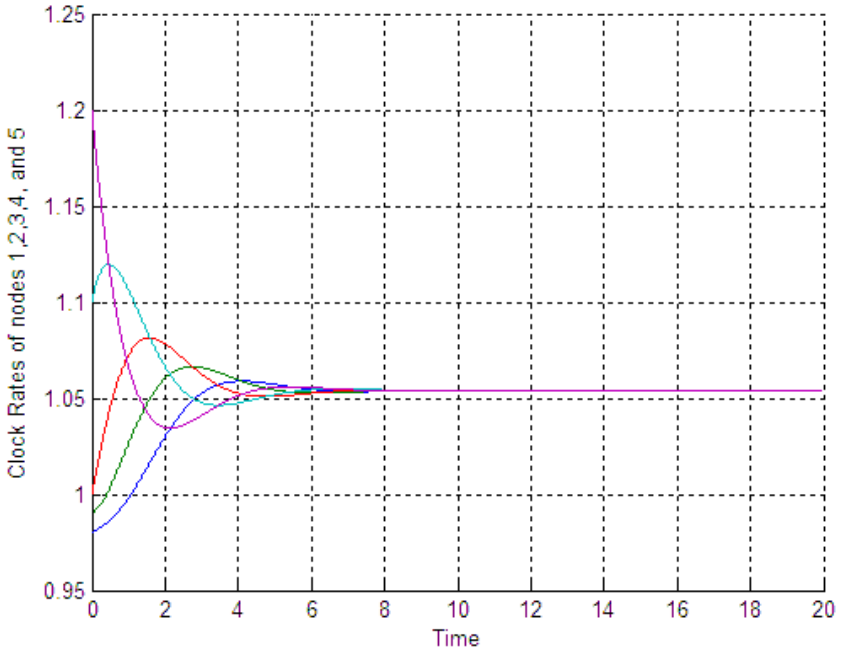

Figure 3.8. Synchronization of the clock rates in a network switching between two strongly connected and balanced topologies.

3.5. Summary Of The Chapter And Concluding Remarks

In this chapter we focused on the stability analysis of the averaging based synchronization algorithm in the presence of the free running frequencies. We first have discussed the stability for fixed topology networks, that is, the connections hence the Laplacian L of the network stays constant over time. For the stability analysis, we first have derived the disagreement dynamics of the network. Then, we introduced a Lyapunov function that is a function of the disagreement vector of the network. By enforcing that the network is strongly connected and balanced, we have been able to show that the differences between the clock values remain bounded.

In the second part of the chapter, we have discussed the average consensus of the clock rates in the presence of the free running frequencies. By applying Laplace transform, we obtained the solution of the system and utilizing the outcomes of the average consensus analysis in the previous chapter, we showed that average consensus of the clock rates is achieved if the network is strongly connected and balanced.

In the third part of the chapter, we have discussed the stability for the time varying topology cases where the Laplacian of the network, $L(t)$, is a function of time. Analogous to the fixed topology case, we have rederived the disagreement dynamics of the network and then have utilized a Lyapunov function to show that the system remains bounded if the network is switching between topologies that are strongly connected and balanced.

In the fourth part, we have discussed the average consensus for switching networks in the presence of the free running frequencies. We showed that the consensus value is independent of the network topology but dependent on the Laplacian of the network. Each topology, as long as strongly connected and balanced, forces the clock rates of the network towards the average of the free running frequencies. Thus, as long as the network switches between strongly connected and balanced topologies, average consensus can be achieved.

4. ANALYSIS OF THE AVERAGING BASED SYNCHRONIZATION ALGORITHM IN NETWORKS WITH TIME DELAY

Message delivery from one node to another in a network may take a random amount of time, which constitutes a major problem for the synchronization of distributed networks. In the previous chapter, we have assumed perfect timing and no delays in the network. However in real life applications, message travel time, communication media and network congestions cause delays in the delivery of the clock values between nodes.

4.1. Mathematical Model of The Algorithm In Networks With Time Delay

When time delays in a network are considered, the averaging based synchronization algorithm takes the form

$$\dot{p}_i(t) = F_i(t) + \gamma_i \sum_{j=1}^{n} a_{ij}[p_j(t-\tau_{ij}) - p_i(t-\tau_{ij})] \tag{4.1}$$

where τ_{ij} corresponds to the time delay of the clock value from node j to node i. Equation (4.1) indicates that node i updates its clock rate utilizing the clock value of node j at time $t-\tau_{ij}$. This can also be shown in matrix form as

$$\dot{p}(t) = F(t) - Lp(t-\tau) \tag{4.2}$$

In [17], the average value consensus of networks with time delay is proved to be achievable with some restrictions on the network topology and assumptions on the time delay in the network. All nodes are assumed to use time delayed clock values when updating their clocks, even if it is their own value.

4.2. Stability Analysis In The Presence of Free Running Frequencies

The analysis in [17] corresponds to the analysis of (4.2) when $F(t) = 0$. In this section, we investigate the performance of (4.1) in detail. The time delay of the communication link is represented by the transfer function

$$h_{ij}(s) = e^{\tau_{ij}s} \tag{4.3}$$

in Laplace domain. If we take the Laplace transform of $\dot{p}(t) = -Lp(t-\tau)$, we obtain

$$P(s) = (sI + L(s))^{-1}p(0) \tag{4.4}$$

where the Laplace transform of the Laplacian matrix is $L(s) = Le^{\tau s}$. The convergence of the synchronization algorithm is equivalent to the stability analysis of the transfer function of the system:

$$G(s) = (sI + L(s))^{-1} \tag{4.5}$$

Theorem 4.2.1. *[17] Consider a network where all the nodes have communication time delays which are equal. The network topology is restricted to be fixed, undirected, and connected. The synchronization algorithm provides average consensus in the network if and only if one of the following equivalent conditions is satisfied.*

(i) $\tau \in (0, \tau_*)$ *where* $\tau_* = \frac{\pi}{2\lambda_n}$, $\lambda_n = \lambda_{max}(L)$
(ii) The Nyquist plot of $\Gamma(s) = \frac{e^{\tau s}}{s}$ *has a zero encirclement around* $-1/\lambda_k$, $\forall k > 1$.

For the stability analysis of the averaging based synchronization algorithm, we again assume equal time delays at all communication links. If we take the Laplace transform of (4.2), we obtain

$$P(s) = (sI + L(s))^{-1}p(0) + (sI + L(s))^{-1}F(s) \tag{4.6}$$

The transfer function due to both terms in (4.6) is identical to the transfer function in (4.5). This indicates that as long as the conditions of Theorem 4.2.1 are satisfied, the network achieves average consensus in the presence of free running frequencies.

4.3. Numerical Analysis of Time Delay Effect

In this section we focus on the numerical analysis of the time delay effect on network synchronization. We simulate different scenarios for the time delay. We have selected a network topology that complies with the topology requirements of Theorem 4.2.1, i.e., the network is fixed, undirected, and connected. The selected network topology is shown in Figure 4.1. The Laplacian L of the selected topology is as below

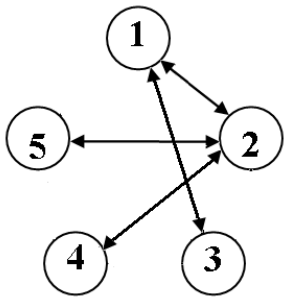

Figure 4.1. A fixed, undirected, and connected network of five nodes.

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}. \tag{4.7}$$

The largest eigenvalue of the Laplacian L is, $\lambda_{max}(L) = 4.17$. According to the first part of Theorem 4.2.1, the network can tolerate time delay τ, as long as $\tau \in (0, 0.37)$. This network topology is used for the numerical analysis of time delay effect in different scenarios.

4.3.1. Equal Time Delay At All Communication Links

First, we investigate the network behavior when equal time delay is applied to all communication links, including when a node uses its own value.

Example 4.3.1. An equal time delay of $\tau = 0.01s$ is assumed for all communication links for the selected network. It is also assumed that each node uses its own value with the same time delay. Since the applied time delay, $\tau = 0.03s$ is below the maximum time delay that the selected network can tolerate, $\tau_{max} = 0.37$, we expect the network achieves average consensus.

We have first applied the synchronization algorithm omitting the free running frequencies where the initial condition for the clock values is taken as $p(0) = [4, 3, 5, 4.2, 3.8]^T$. Figures 4.2 and 4.3 show that the system achieves average consensus, where the clock values converge to the average of the initial values, which is 4, while the clock rates converge to zero.

Next, we have included the free-running frequencies as $F = [0.98, 0.99, 1, 1.1, 1.2]^T$. The simulation results show that average consensus is achieved and the clock rates converge to the average of the free running frequencies, that is, 1.054 as seen in Figures 4.4 and 4.5.

4.3.2. Equal Time Delays At All Communication Links While Each Node Uses Its Own Value Without Time Delay

In the previous section, we have assumed that each node uses its own value with the same time delay that is applied to all communication links in the network. However, a node can access its value without a time delay. In this section, we will investigate the network behavior where each node uses the clock values of the neighboring nodes with equal time delay, while using its own value without a time delay.

Example 4.3.2. The selected network is simulated for three different time delays, $\tau_1 = t_d = 0.03s$, $\tau_2 = 3 \times t_d = 0.09s$, and $\tau_3 = 7 \times t_d = 0.21s$, where each node

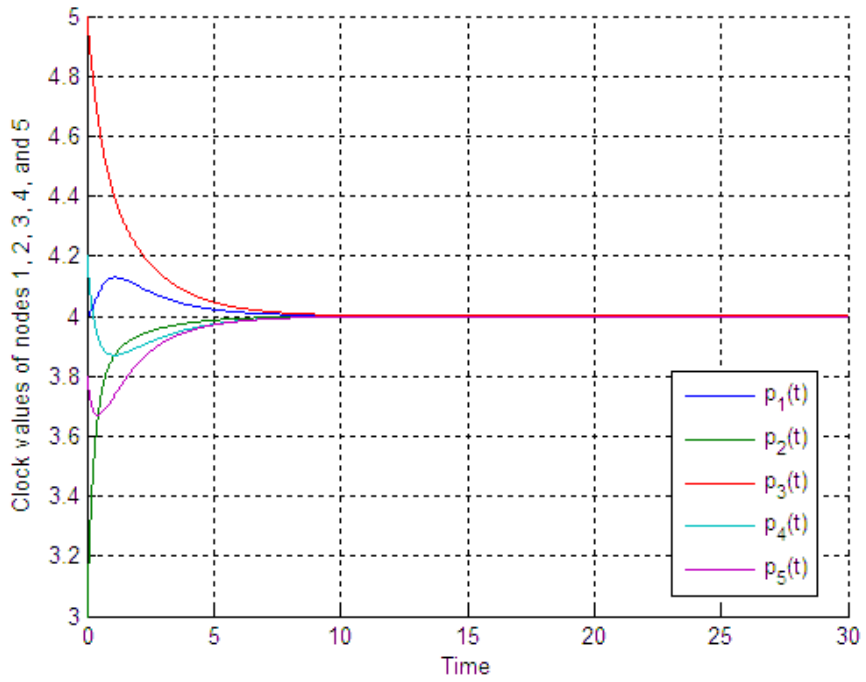

Figure 4.2. Synchronization of clock values in a fixed, undirected, and connected network with equal time delay at all communication links.

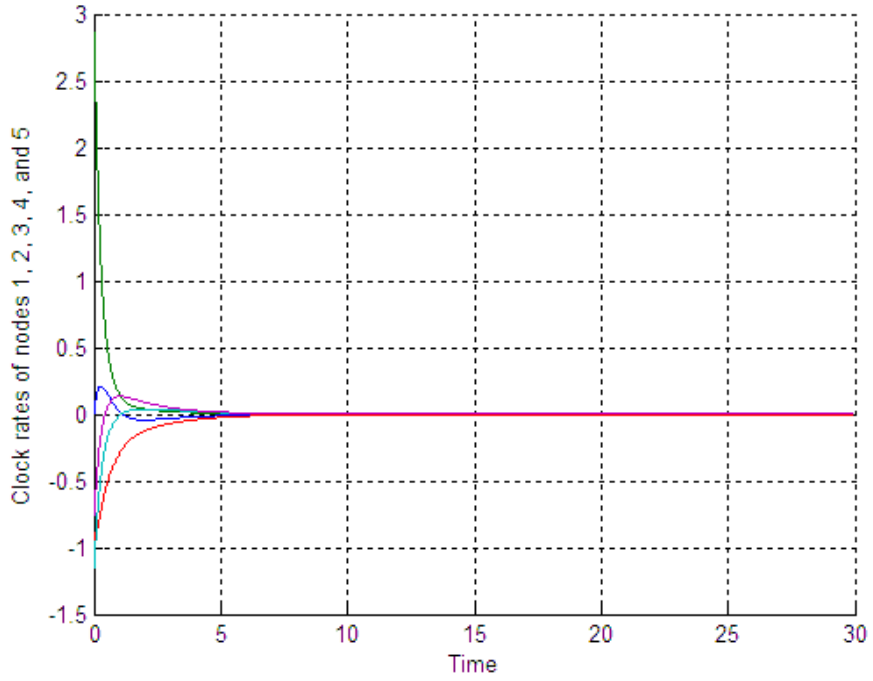

Figure 4.3. Synchronization of clock rates in a fixed, undirected, and connected network with equal time delay at all communication links.

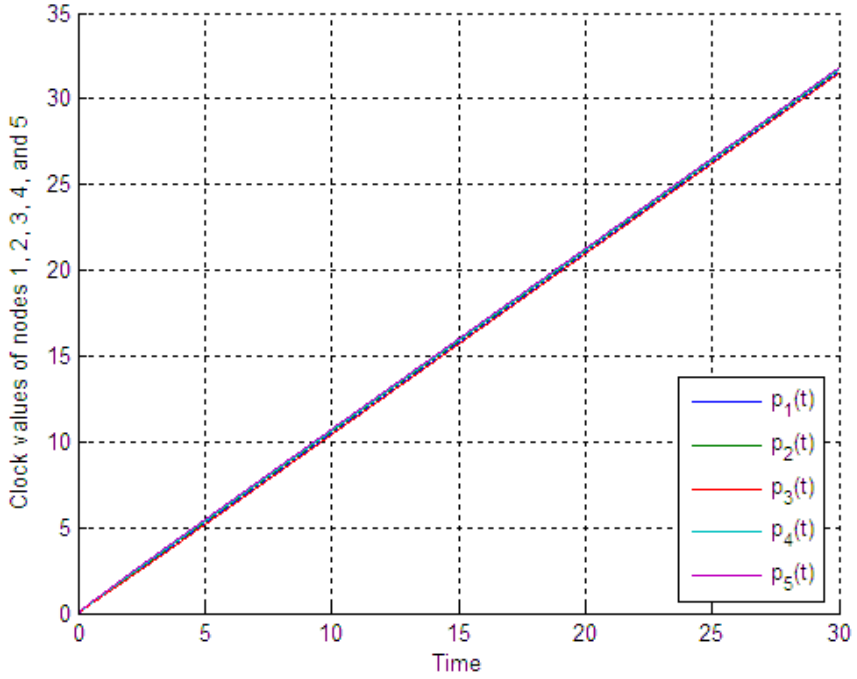

Figure 4.4. Synchronization of clock values with equal time delays at all communication links and free running frequencies.

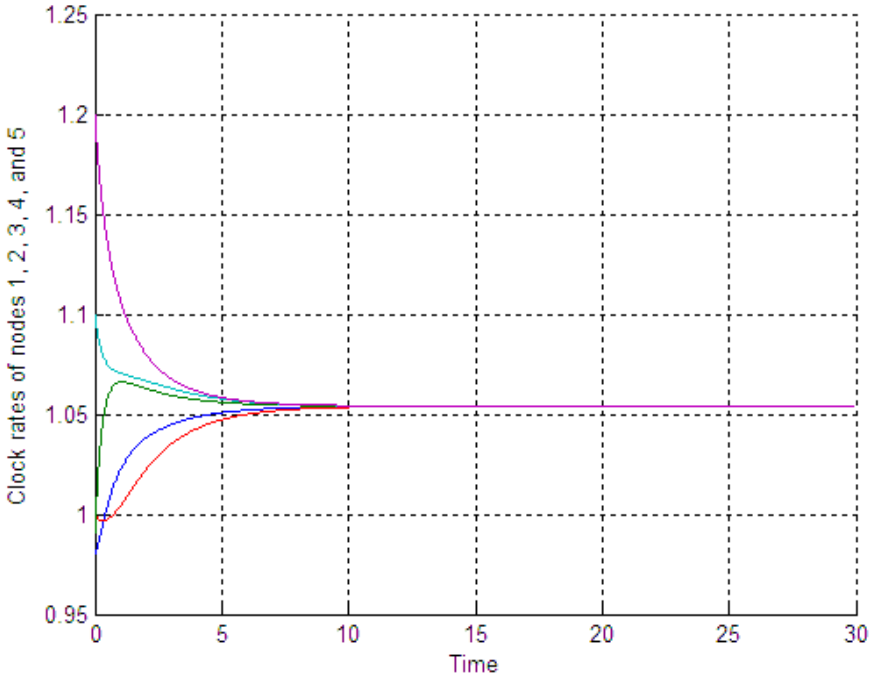

Figure 4.5. Synchronization of clock rates with equal time delays at all communication links and free running frequencies.

uses its current value without a delay, and all other values from neighboring nodes with equal time delay. Free running frequencies are taken as in the previous example, $F = [0.98,\ 0.99,\ 1,\ 1.1,\ 1.2]^T$. The simulation results are shown in Figure 4.6. In this figure, *Simulation* 1 represents the results of the previous example where the nodes uses their own values with the same time delay applied to the network. This is included in the figure for comparison purposes. *Simulation* 2, *Simulation* 3, and *Simulation* 4 represent the results of the network synchronization, where the time delays, $\tau_1 = t_d = 0.03s$, $\tau_2 = 3 \times t_d = 0.09s$, and $\tau_3 = 7 \times t_d = 0.21s$, are applied respectively to all communication links in the network. Time delay is not applied only when a node uses its own value to update its frequency.

The results show that the network achieves consensus for all of the scenarios. However, the network can not achieve average consensus unless each node uses its current value with the same time delay that is applied to the network. The results also indicate that the consensus value decreases as the time delay, τ, increases.

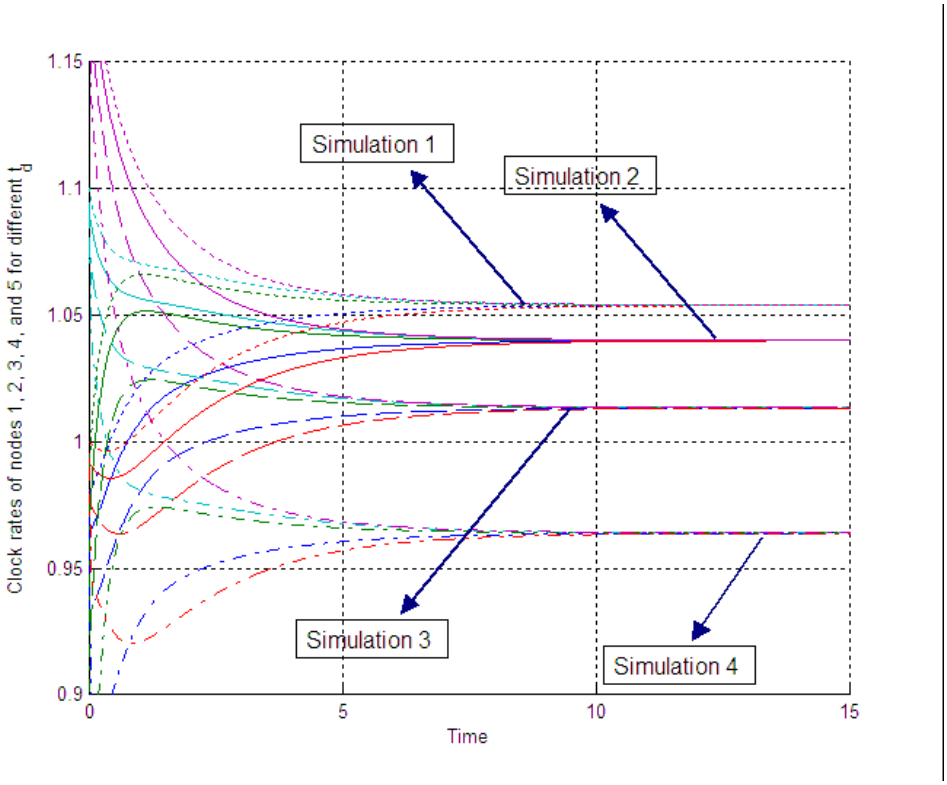

Figure 4.6. Synchronization of clock rates with equal time delay at all communication links and no delay when a node uses its own value.

4.3.3. Time Invariant Unique Time Delay At Each Communication Link

In the previous sections, we have assumed equal time delay across the network. In this section we investigate the network synchronization where each communication link has its unique time delay that stays constant over time.

Example 4.3.3. The selected network is simulated with unique time delay at each communication link, while nodes use their own values without a time delay. Time delay for each link is assumed to vary between $0.03s$ and $0.21s$ and the exact values that are used in the network are shown in matrix format below, where τ_{ij} represents the time delay value that is unique to the communication link from node j to node i. The free-running frequencies as $F = [0.98,\ 0.99,\ 1,\ 1.1,\ 1.2]^T$.

$$\tau = \begin{bmatrix} 0 & 0.03 & 0.015 & 0 & 0 \\ 0.06 & 0 & 0 & 0.18 & 0.12 \\ 0.09 & 0 & 0 & 0 & 0 \\ 0 & 0.15 & 0 & 0 & 0 \\ 0 & 0.21 & 0 & 0 & 0 \end{bmatrix}. \tag{4.8}$$

The results of the simulation in Figure 4.7 show that the network achieves consensus, however not average consensus.

4.3.4. Time Varying Random Time Delay At Each Communication Link

Due to internal or external reasons, such as battery life, temperature, and network congestions, time delay of communication links may vary in time. In this section, we will assume time varying random time delays for each communication link. Similar to the previous two sections, each node is assumed to use its own value without a time delay.

Example 4.3.4. The selected network is simulated with random time delays that vary over time. The free-running frequencies are taken as $F = [0.98,\ 0.99,\ 1,\ 1.1,\ 1.2]^T$.

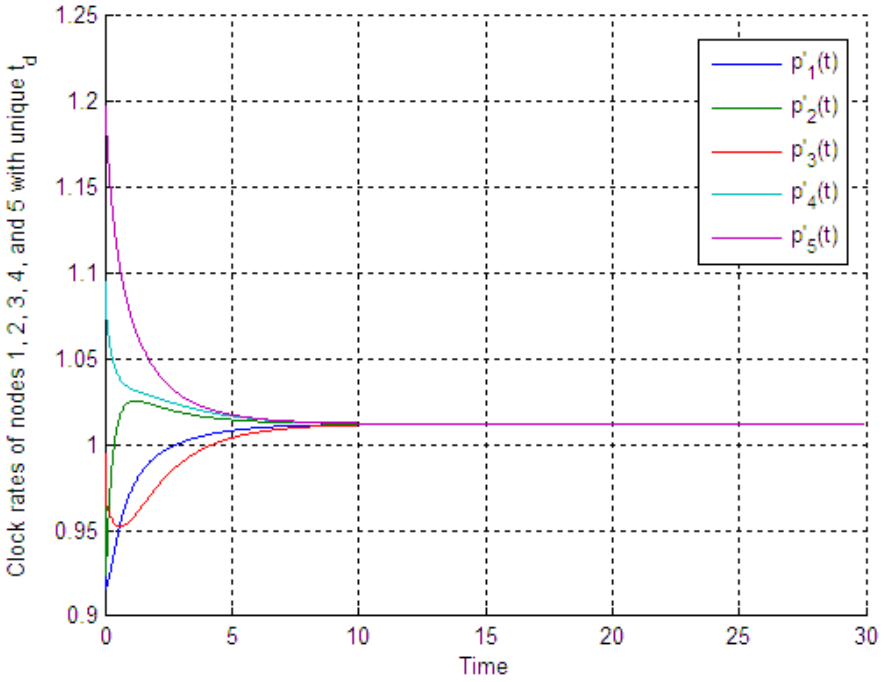

Figure 4.7. Synchronization of clock rates with unique time delay at each communication link.

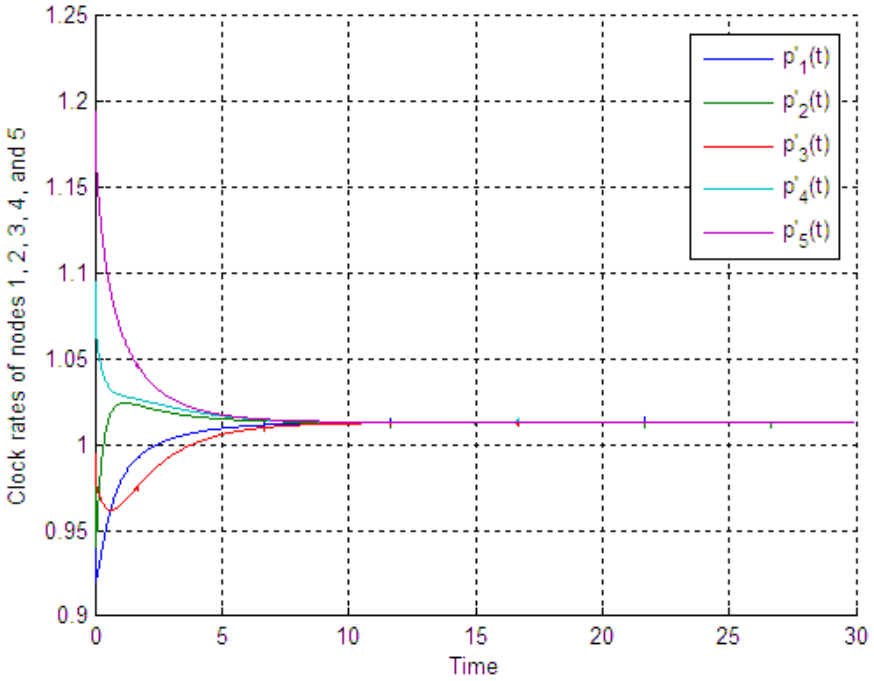

Figure 4.8. Synchronization of clock rates with time varying random time delay at each communication link, updated every 10s.

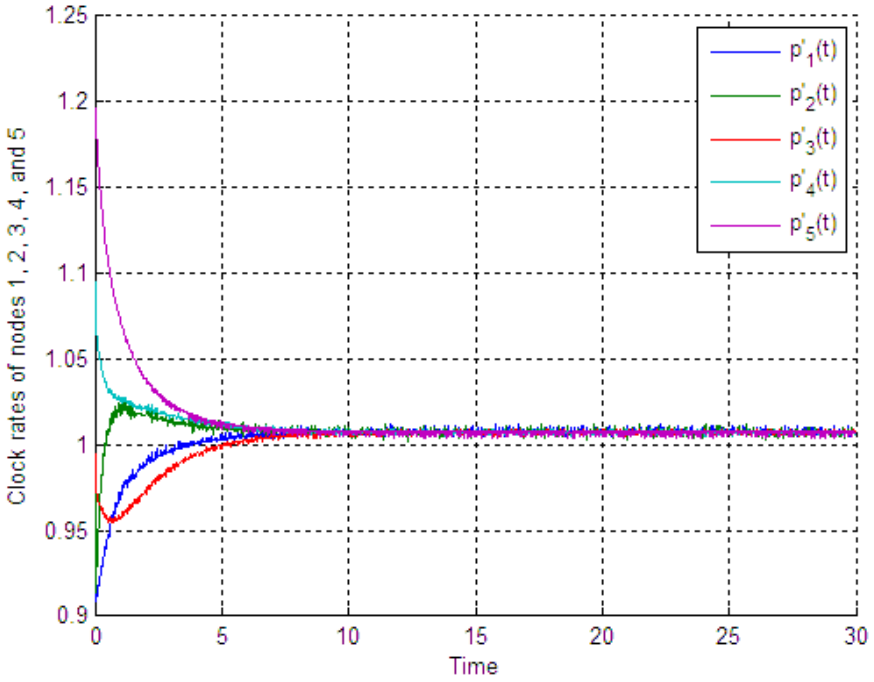

Figure 4.9. Synchronization of clock rates with time varying random time delay at each communication link, updated every 0.1s.

The random time delays are assumed to vary between $0.001s$ and $0.21s$. We simulate the synchronization of the network for two scenarios. In the first scenario, time delay at each communication link is randomly updated at every $10s$; in the second scenario, time delay at each communication link is updated every $0.1s$. The results of the simulations in Figures 4.8 and 4.9 show that the network achieve consensus, however not average consensus.

4.4. Trade off Between Performance and Robustness

Robustness of a network is a measure of how much time delay can the network tolerate. Performance of a network is a measure of how fast the network converges to the consensus value.

The first part of Theorem 4.2.1 sets the upper bound for the time delay τ that a network can tolerate as $\frac{\pi}{2\lambda_n}$, where λ_n is the largest eigenvalue of the Laplacian of the network. The authors of [17] utilized Theorem 2.2.2 (Gershgorin Circle Theorem)

to relate the upper limit for the time delay to the maximum in-degree, d_{max}, of the network. Theorem 2.2.2 states that the all the eigenvalues of the Laplacian L should be smaller than or equal to the maximum in-degree of the network. The maximum in-degree sets the upper limit for the eigenvalues of the Laplacian, where the second largest eigenvalue determines the speed of convergence. Hence the maximum in-degree is proportional to the convergence speed, in other words, performance of the network. The relation between the time delay and maximum in-degree is given as below [17]:

$$\tau \leq \frac{\pi}{4d_{max}(G)} \tag{4.9}$$

Equation (4.9) shows that the maximum time delay that a network can tolerate is inversely proportional to the maximum in-degree of that network. As a result, there is a trade off between robustness and performance of a network. If the maximum in-degree is increased, then the system reaches consensus faster; however the maximum time delay that the network can tolerate decreases.

Example 4.4.1. The network topology in Figure 4.1 of Example 4.3.1 is simulated with equal time delay, $\tau = 0.03$, for three different maximum in-degree values, where $deg_{max_1} = 3$, $deg_{max_2} = 15$, and $deg_{max_3} = 150$.

The convergence of the clock rates of all three networks are shown in Figure 4.10. In this figure, part a represents the clock rates of the network with $deg_{max_1} = 3$, part b represents the clock rates of the network with $deg_{max_2} = 15$, and part c represents the clock rates of the network with $deg_{max_3} = 150$. The figure clearly shows that as the maximum in-degree increases, the system converges faster to the consensus. The network with $deg_{max_2} = 15$ shown in part b, converges faster than the network with $deg_{max_1} = 3$, shown in part a. However, as the maximum in-degree increases, the maximum time delay that the network can tolerate decreases, that is why the network with $deg_{max_3} = 150$, shown in part c, can not tolerate the equal amount of time delay with others, and the clock rates diverge.

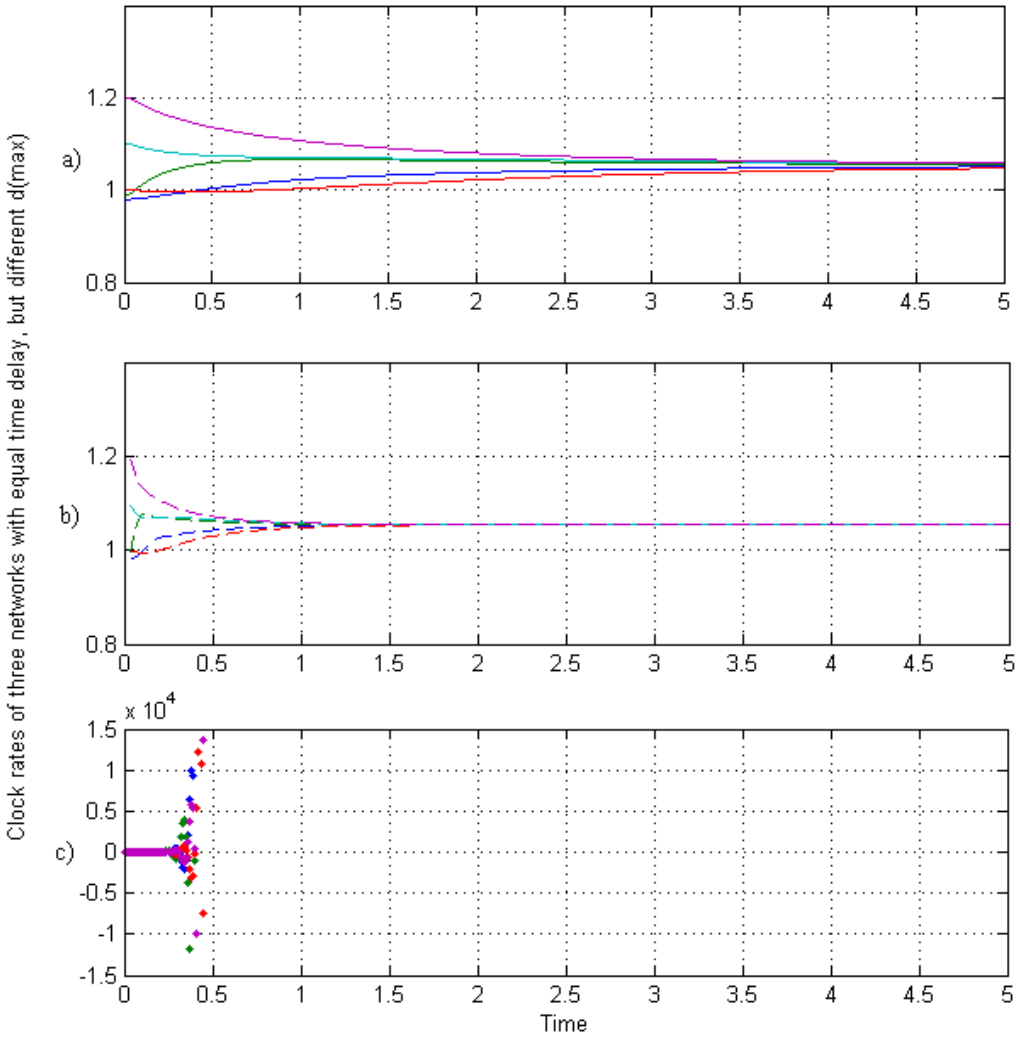

Figure 4.10. Simulation of the trade off between robustness and performance in networks with time delay.

4.5. Summary of The Chapter and Concluding Remarks

In this chapter, we have introduced time delays in message communications of a network. We have utilized the findings of [17] to show the average consensus of the network when free running frequencies are omitted. Then, we have showed that the transfer function of the system in the presence of the free running frequencies is identical to the transfer function of the system when the free running frequencies are omitted. Hence we have showed that if the average consensus conditions given in [17] are satisfied, then the network will achieve average consensus in the presence of free running frequencies as well. In the next section, we have made numerical analysis of

the time delay effect in network synchronization. We have selected one topology which is fixed, undirected and connected, and simulated various scenarios with respect to the time delay.

First, we have investigated the synchronization of the network when equal time delay is applied to all communication links, and also each node uses its own value with the same time delay. In this scenario, the network achieves average consensus. Second, we have applied equal time delay to all communication links, however, each node uses its own value without a time delay. In this scenario, we have observed that the network reaches consensus, however, it does not achieve average consensus. We have further extended this scenario, by applying different time delays. We have observed that as the time delay increases, the consensus value decreases. Then, we have investigated the effect of unique time delay at each communication link. We have simulated for both constant and random time delay values over time. The results are similar to the previous scenario, that is, the network reaches a consensus, however, does not achieve average consensus.

In the last section we have discussed the robustness versus performance in networks with time delay. We have utilized the findings of [17], which shows that the maximum in-degree of the network is proportional to the convergence speed, and inversely proportional to the maximum time delay that the network can tolerate. This accounts for the trade off between robustness and performance.

5. AVERAGING BASED SYNCHRONIZATION ALGORITHM IN THE PRESENCE OF FAULTY NODES

In previous chapters, we have investigated the averaging based synchronization algorithm under the assumption that the nodes in the network are reliable. However, some nodes may as well be faulty. In this chapter, we focus on the synchronization of the network when the proposed algorithm is applied in the presence of faulty nodes.

5.1. Fault Tolerant Synchronization

Faulty nodes may occur due to internal malfunctioning of the node, such as a hardware failure, or due to external reasons that cause a distorted message. These nodes are referred to as Byzantine nodes [20]. A faulty node may cause false data as well as inconsistent data during its communication with other nodes. In the presence of this kind of faulty nodes, a synchronization algorithm may deviate from delivering what it is designed for. There exists fault tolerant algorithms which compensate for faulty nodes in the network. It is important to apply a fault tolerant algorithm to ensure that the clocks of a network are synchronized despite the presence of faulty nodes [21]. In the following example, the behavior of a network without a fault tolerant algorithm is explored when there is a Byzantine node.

Example 5.1.1. In this example, the network of the Example 3.2.1 is taken as base, where the network is as in Figure 5.1. The fifth node of the network is turned into a Byzantine node. The Byzantine node does not update its clock value based on the

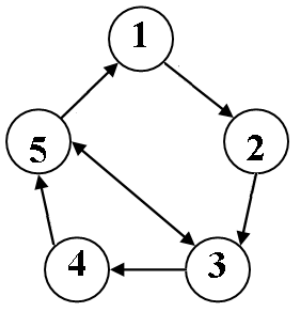

Figure 5.1. A strongly connected, balanced network of five nodes.

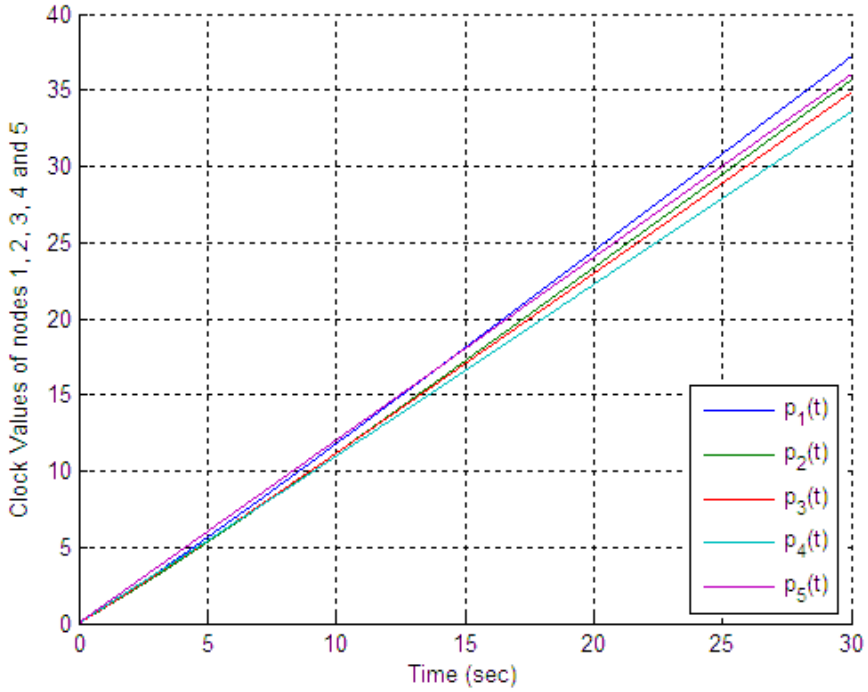

Figure 5.2. Clock values of a five node network, where node five is a Byzantine node.

other nodes in the network. In addition, the nodes one, three, and four receive incorrect and inconsistent clock values from node five. The values that node one, three and four receive are $k_1 p_5$, $k_3 p_5$, and $k_4 p_5$ where k_i is the Byzantine constant and different than one. Assume $k_1 = 1.1$, $k_3 = 1.05$, and $k_4 = 0.9$. The free running frequencies of the nodes are, $F = [0.98, 0.99, 1.0, 1.1, 1.2]^T$.

As Figures 5.2 and 5.3 show, if the synchronization algorithm is not fault tolerant, the presence of Byzantine nodes that generate inconsistent data to other nodes cause instability in the system, i.e., clock rates do not converge to a common rate, and clock values drift away from each other.

5.2. Single Faced Byzantine Nodes

Byzantine nodes send wrong clock information to non-faulty nodes in the network. If a Byzantine node sends the same incorrect value to all non-faulty nodes, then it is called a single faced Byzantine. This implies that all Byzantine constants, $k_i s$, are equal to each other in a single faced Byzantine network.

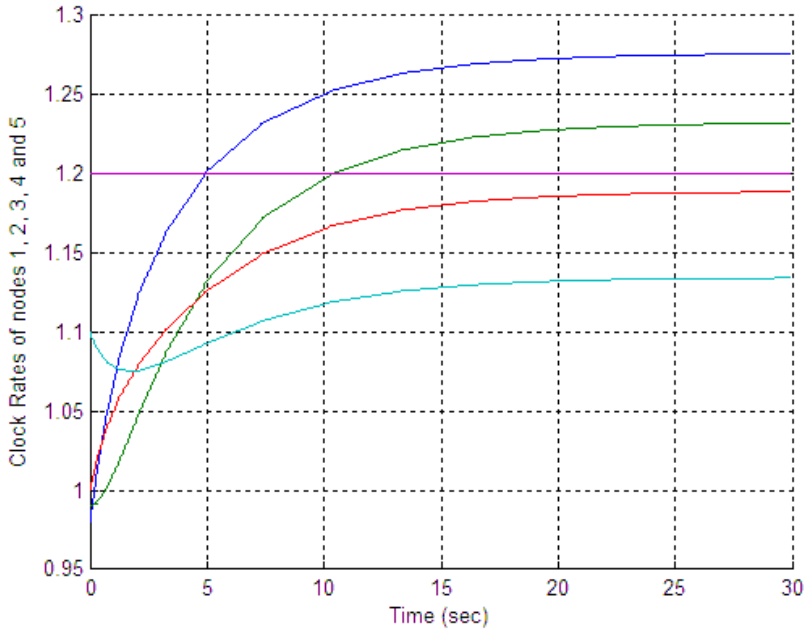

Figure 5.3. Clock rates of a five node network, where node five is a Byzantine node.

If all single faced Byzantine nodes send their clock information to all non-faulty nodes, and non-faulty nodes apply averaging based synchronization with the received values, then non-faulty nodes converge to the average of Byzantine clocks even if the non-Byzantine system is not convergent.

Example 5.2.1. Consider the five node network shown in Figure 5.4. Node three and node five are the two single faced Byzantine clocks in the network. Let clock three send 70 per cent of its original clock value, while node five send 90 per cent of its original clock value. Single faced Byzantine clocks three and five send their clock values to

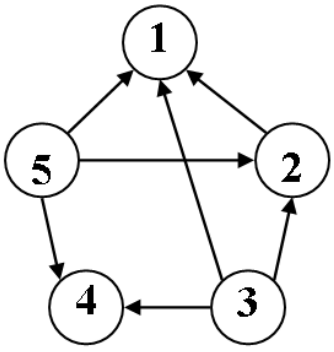

Figure 5.4. A five node network, where node three and node five are the two single faced Byzantine nodes.

all other non–faulty clocks in the network. Free running frequency of the nodes are, $F = [0.98, 0.99, 1.0, 1.1, 1.2]^T$.

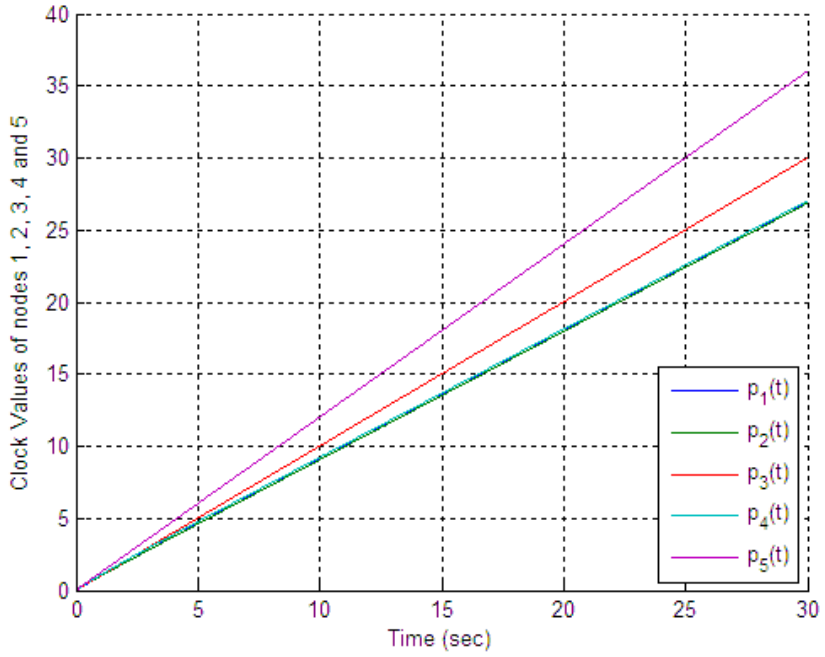

Figure 5.5. Clock values of the network with two single faced Byzantine nodes.

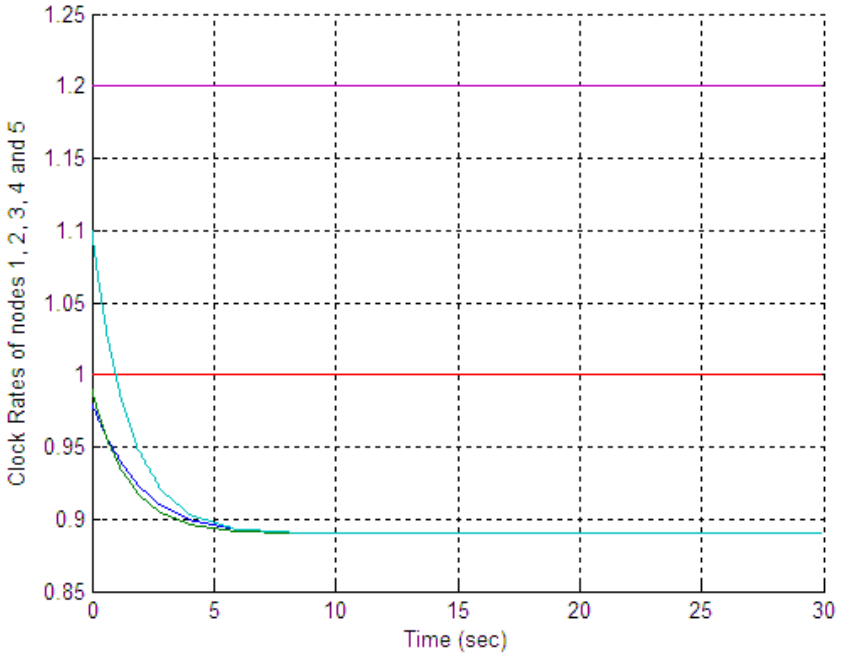

Figure 5.6. Clock rates of the network, with two single faced Byzantine nodes.

As it can be seen from the simulation results in Figures 5.5 and 5.6, the rates of the non-faulty clocks converge to a common value which equals to the mean of the received clock information from single faced Byzantine nodes.

$$p_\infty = \frac{p_3 + p_5}{2} = \frac{1\frac{70}{100} + 1.2\frac{90}{100}}{2} = 0.89 \tag{5.1}$$

5.3. Summary Of The Chapter And Concluding Remarks

In this chapter, we have introduced the faulty nodes in a network that are also known as Byzantine nodes. We have showed the lack of synchronization in the presence of faulty nodes, even if a synchronization algorithm is applied which is not fault tolerant.

We then introduced single faced Byzantine nodes. In the presence of single faced Byzantine nodes, we have showed by a numerical example that, if the Byzantine nodes send their clock values to all other non faulty nodes in the network, then the network converges to the average of the Byzantine nodes. The theoretical proof of this convergence can be carried out as a future study.

6. CONCLUSION

In this thesis, we introduced an averaging based synchronization algorithm in continuous time. The clocks in a network may differ from each other over time. This algorithm provides distributed synchronization where each node uses the clock values of neighbor nodes to update its running frequency. The general literature review we presented in the first chapter shows the evolution in synchronization algorithms with the developments in wireless networks and the advantages of distributed synchronization algorithms.

In the second chapter, we set the basics of graphical representations, definitions and theorems which are significant tools in the analysis of synchronization algorithms in networks. After building the basics for the analysis, we discussed the stability of the network when the proposed algorithm is applied. We utilized the findings of [17] which show the stability and average consensus for network synchronization which share the same system dynamics with our proposed algorithm in the case when free running frequencies are omitted. These findings show that, for fixed and time varying topology networks, the network is required to be strongly connected to achieve consensus. Furthermore, if average consensus is desired then the network should also be balanced.

In the third chapter, we extended the stability and average convergence analysis to the case where free running frequencies are included for both fixed and time varying topology networks. For the stability analysis of both fixed and time varying topology networks, we derived the disagreement dynamics of the network and defined a Lyapunov function that is a function of the disagreement vector. Enforcing the condition for the network to be strongly connected and balanced, we showed that the difference between the clock rates of the system remains bounded for bounded input. For average consensus, we utilized Laplace transform to obtain the solution of the system and showed that the average consensus is achieved if the network topology is strongly connected and balanced for fixed topology networks. To show the average consensus in time varying topology networks, we first note that the consensus value is independent

of the network topology but dependent of the Laplacian of the network. Laplacian of each strongly connected and balanced topology forces the clock rates towards average consensus. Thus, as long as the network switches between strongly connected and balanced topologies, average consensus can be achieved.

In the fourth chapter, we focused on time delays in message communications. We investigated the stability and average consensus of the networks both excluding and including the free running frequencies. Utilizing the findings of [17] and noting that in either case, the network has identical transfer functions, we show that the system achieves average consensus when it is fixed, undirected, and connected. In the second part, we focused on numerical analysis of the time delay effect. We simulated a selected network which is fixed, undirected, and connected, for different time delay scenarios. We showed that if equal time delay is applied across the network, including when a node uses its own value, average consensus is achieved. However, if equal time delay is applied across the network, but a node uses its own value without time delay, consensus is achieved, but not average consensus. In this scenario, when time delay applied to the network increases, the consensus value decreases. Next, unique time delay is applied to each communication link, both constant over time and random. In both cases, the network achieved consensus but not average consensus.

In the last chapter, we introduced faulty nodes in a network that share incorrect and inconsistent data with the network. We show that unless the synchronization algorithm is fault tolerant, consensus can not be achieved.

REFERENCES

1. Jeremy, E. and D. Estrin, "Sensor Networks: A Bridge to The Physical World", in C. S. Raghavendra, K. M. Sivalingam, and T. Znati (eds), *Wireless Sensor Networks*, pp. 3-20, Springer, New York, 2004.

2. Simeone, O., U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, "Distributed Synchronization in Wireless Sensor networks", *IEEE Signal Processing Magazine*, pp. 81-97, September 2008.

3. Qun, L. and D. Rus, "Global Clock Synchronization in Sensor Networks", *IEEE Transactions on Computers*, Vol. 55, No. 2, February 2006.

4. Krishnamavhari, B., *Networking Wireless Sensors*, pp. 57-69, Cambridge University Press, Cambridge, 2005.

5. Dressler, F., *Self-Organization in Sensor and Actor Networks*, pp. 220-231, John Wiley & Sons Ltd, Wiltshire, 2007.

6. Cristian, F., "Probabilistic Clock Synchronization", *Distributed Computing*, Vol. 3, No. 3, pp. 146-158, July 1986.

7. Gusella, R. and S. Zatti, "The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD", *IEEE Transactions on Software Engineering*, Vol. 15, No. 7, pp. 847-853, July 1989.

8. Chang, E. and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processors", *Communications of the ACM*, Vol. 22, No. 5, pp. 281-283, 1979.

9. Schenato, L. and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network", *Proceedings of the 46th IEEE Conference on*

Decision and Control, pp. 2289-2294, December 2007.

10. Ranganathan, P. and K. Nygard, "Time Synchronization in Wireless Sensor Networks: A Survey", *International Journal of UbiComp*, Vol.1, No.2, 2010.

11. Ganeriwal, S., R. Kumar, and M. Srivastaya, "Timing-sync protocol for sensor networks", *Proceedings of the first ACM conference on Embedded networked sensor systems*, pp. 133–149, November 2003.

12. Maroti, M., B. Kusy, G. Simon, and A. Ldeczi, "The flooding time synchronization protocol", *Proceedings of the 2nd international conference on Embedded networked sensor systems SenSys04*, pp. 39-40, 2004.

13. Elson, J., L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts", *Proceedings of the 5th symposium on Operating systems design and implementation (OSDI02)*, pp. 147-163, 2002.

14. Werner-Allen, G., A. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects", *In ACM Conference on Embedded Networked Sensor Systems (SenSys05)*, 2005.

15. Brilliant, M. B., E. Chang, and R. Roberts, "The determination of frequency in systems of mutually synchronized oscillators", *The Bell System Tecnhical Journal*, Vol. XLV, pp. 1737-1748, 1966.

16. Godsil, C. and G. Royle, *Algebraic Graph Theory, Graduate Texts in Mathematics*, Springer-Verlag, New York, 2000.

17. 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, September 2004.

18. Tuncer, A. C., M. Coates, and M. Rabbat, "Distributed Average Consensus Using Probabilistic Quantization", *Proceedings of the 2007 IEEE/SP 14th Workshop on*

Statistical Signal Processing, pp. 640-644, 2007.

19. Sundararaman, B., U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey", *Ad Hoc Networks*, Vol. 3, No. 3, pp. 281–323, 2005.

20. Lamport, L. and P. M. Melliar-Smith, "Byzantine Clock Synchronization", *Proceedings of The Third Annual ACM Symposium on Principles of Distributed Computing*, pp. 68-74, August, 1984.

21. Lamport, L. and P. M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults", *Journal of the Association for Computing Machinery*, Vol. 32, No. 1, pp. 52-78, January, 1985.