

**A DISTRIBUTED TIME AVERAGE  
SYNCHRONIZATION PROTOCOL FOR WIRELESS  
SENSOR NETWORKS**

BY  
**IBRAHIM AHMED ABDALLAH NEMER**

A Thesis Presented to the  
DEANSHIP OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**  
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**MASTER OF SCIENCE**

In  
**TELECOMMUNICATION ENGINEERING**

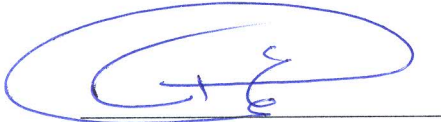
**MAY 2015**

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

**DEANSHIP OF GRADUATE STUDIES**

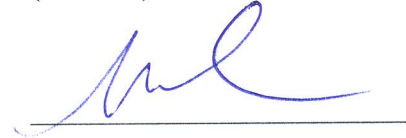
This thesis, written by **Ibrahim Nemer** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN TELECOMMUNICATION ENGINEERING.**



Dr. Ali A. Al-Shaikhi  
Department Chairman



Dr. Ali A. Al-Shaikhi  
(Advisor)



Dr. Ahmed A. Masoud  
(Member)



Dr. Salam A. Zummo  
Dean of Graduate Studies



Dr. Samir Al-Ghadban  
(Member)

11/6/15  
Date

© Ibrahim Ahmed Abdallah Nemer

2015

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ  
{قُلْ إِنَّ صَلَاتِي وَنُسُكِي وَمَحْيَايَ وَمَمَاتِي لِلَّهِ رَبِّ الْعَالَمِينَ}

صدق الله العظيم

*This Thesis is dedicated to*

*My Mother (Fatima)*

*Dear Father (Ahmed)*

*My Wife (Fida')*

*My Sisters*

*My Brothers*

*My Friends*

*My Beloved People*

*My Holy Homeland Palestine*

## ACKNOWLEDGMENTS

In the name of Allah, the most gracious, the most merciful all praise is to almighty Allah for having guided me all over my life. Acknowledgement is due to King Fahd University of Petroleum and Minerals for the great support to this work. My deep appreciation is reserved for thesis advisor **Dr. Ali Al-Shaikhi** for his guidance, valuable time and attention he devoted throughout the course of this work. My numerous intrusions into his office were always met with a considerable response and care. Thanks are also due to committee members **Dr. Ahmad A. Masoud** and **Dr. Samir Al-Ghadhban** for their interest, attention and suggestion. I wish also to thank all parties who have supported me in this work. My great appreciations are also due to all members of my family and to friends who give me the self-confidence to face the challenge.

# TABLE OF CONTENTS

ACKNOWLEDGMENTS .....	V
TABLE OF CONTENTS .....	VI
LIST OF TABLES .....	X
LIST OF FIGURES .....	XII
LIST OF ABBREVIATIONS .....	XV
ABSTRACT.....	XVII
ملخص الرسالة .....	XIX
CHAPTER ONE INTRODUCTION TO WIRELESS SENSOR NETWORKS .....	1
1.1 Background .....	1
1.2 WSNs.....	3
1.2.1 Architecture of the Node .....	3
1.2.2 Protocols in WSNs.....	6
1.2.3 Topologies in WSNs .....	7
1.3 Importance of Using WSNs and Some Applications.....	7
1.4 Types of Sensor Nodes .....	9
1.4.1 TinyOS Overview .....	10
1.4.2 Micaz Mote Platform .....	11
1.4.3 MIB520 USB Interface Board.....	12
1.5 WSNs in Harsh Environments.....	13
1.6 Summary .....	15
CHAPTER TWO TIME SYNCHRONIZATION IN WSNs.....	16

2.1	Clock Model for the Sensor Node.....	16
2.2	Importance of Synchronization in WSNs .....	18
2.3	Time Synchronization Protocols.....	20
2.3.1	Tree Structure Protocols .....	21
2.3.2	Cluster Structure Protocols.....	25
2.3.3	Distributed Protocols .....	28
2.4	Comparison between Time Synchronization Protocols.....	35
2.5	Stopping Criterion .....	37
2.5.1	Literature on Different Stopping Criteria .....	38
2.5.2	Stopping Threshold.....	41
2.6	Thesis Contributions.....	42
<b>CHAPTER THREE AVERAGING PROTOCOL .....</b>		<b>45</b>
3.1	Averaging Protocol Concept .....	45
3.1.1	Description of Asynchronous Average Consensus Algorithm.....	47
3.1.2	Description of Synchronous Average Consensus Algorithm.....	48
3.2	Examples on this Protocol.....	50
3.2.1	Examples on Small Networks.....	50
3.2.2	Examples on Large Networks .....	54
3.3	Simulation Results .....	60
3.3.1	Simulation Results for Small Networks .....	62
3.3.2	Simulation Results for Large Networks.....	68
3.4	Practical Results.....	77
3.4.1	Practical Results for Small Networks.....	77
3.4.2	Practical Results for Large Networks .....	80
3.5	Summary .....	84

<b>CHAPTER FOUR STOPPING CRITERION .....</b>	<b>87</b>
4.1 Stopping Criterion .....	87
4.2 Steady State Stopping Criterion .....	89
4.3 Proposed Stopping Criterion for the Dip Region .....	91
4.4 Simulation Results .....	101
4.4.1 Simulation Results for the Small Networks .....	101
4.4.2 Simulation Results for the Large Networks.....	106
4.5 Practical Results.....	116
4.5.1 Practical Results for the Small Networks.....	116
4.5.2 Practical Results for the Large Networks .....	119
4.6 Hardware Platform and Implementation Details for Real-world Experiments.....	121
4.6.1 4-Nodes Grid Topology.....	123
4.6.2 9-Nodes Grid Topology.....	125
4.6.3 16-Nodes Grid Topology.....	128
4.7 Comparisons and Tests of the protocol Under Various Scenarios .....	132
4.7.1 Summarized Simulation Results for Different Sizes (4, 9, and 16) .....	132
4.7.2 Summarized Practical Results for Different Sizes (4, 9, and 16) .....	135
4.7.3 Simulation vs. Practical Results for Different Sizes (4, 9, and 16) .....	136
4.7.4 Practical Results for Different Synchronization Protocols.....	139
4.7.5 Simulation Results for Changing the Initial Values/Different Runs .....	143
4.8 Summary .....	145
<b>CHAPTER FIVE CONCLUSION: SUMMARY AND FUTURE WORK .....</b>	<b>147</b>
5.1 Summary .....	147
5.2 Future Work.....	150
<b>REFERENCES.....</b>	<b>152</b>



<b>APPENDIX.....</b>	<b>156</b>
<b>VITAE.....</b>	<b>158</b>

## LIST OF TABLES

Table 1-1	Technical specifications of Micaz mote .....	12
Table 1-2	Technical specifications of MIB520CB .....	13
Table 2-1	Capabilities for different protocols.....	33
Table 2-2	Average absolute errors in millisecond .....	41
Table 2-3	Convergence and accuracy results (1 tick equal to $=1/32K=30.5\mu s$ ).....	42
Table 3-1	Specifications of the network .....	61
Table 3-2	Summarized data for 4-Nodes Grid Topology .....	64
Table 3-3	Summarized data for 4-Nodes Hexa Topology .....	66
Table 3-4	Summarized data for 4-Nodes Random Topology .....	68
Table 3-5	Summarized data for 9-Nodes Grid Topology .....	70
Table 3-6	Summarized data for 9-Nodes Hexa Topology .....	72
Table 3-7	Summarized data for 9-Nodes Random Topology .....	74
Table 3-8	Summarized data for 16-Nodes Grid Topology .....	75
Table 3-9	Summarized data for 16-Nodes Hexa Topology .....	76
Table 3-10	Summarized data for 16-Nodes Random Topology .....	77
Table 3-11	Summarized practical data for 4-Nodes Grid Topology .....	79
Table 3-12	Summarized practical data for 9-Nodes Grid Topology .....	81
Table 3-13	Summarized practical data for 16-Nodes Grid Topology .....	84
Table 4-1	Summarized Data of different filter sizes ( $W$ ) for Grid network .....	95
Table 4-2	Optimized $C$ values for different topologies and sizes.....	95
Table 4-3	Simulation outputs for 4-nodes Grid Topology.....	101
Table 4-4	Simulation outputs for 4-nodes Hexa Topology .....	104
Table 4-5	Simulation outputs for 4-nodes Random Topology .....	105
Table 4-6	Simulation outputs for 9-nodes Grid Topology.....	106
Table 4-7	Simulation outputs for 9-nodes Hexa Topology .....	109
Table 4-8	Simulation outputs for 9-nodes Random Topology .....	110
Table 4-9	Simulation outputs for 16-nodes Grid Topology.....	111
Table 4-10	Simulation outputs for 16-nodes Hexa Topology .....	114
Table 4-11	Simulation outputs for 16-nodes Random Topology .....	115
Table 4-12	Practical outputs for 4-nodes Grid Topology .....	117
Table 4-13	Practical outputs for 9-nodes Grid Topology .....	120
Table 4-14	Practical outputs for 16-nodes Grid Topology .....	121
Table 4-15	Specification of the implementation part .....	123
Table 4-16	Practical Iteration and Error Values for 4-nodes .....	125
Table 4-17	Iteration and Error Values for 9-nodes .....	128
Table 4-18	Iteration and Error Values for 16-nodes .....	131
Table 4-19	Specifications of three protocols .....	140
Table 4-20	Number of iterations for different periods of 9-nodes.....	144

Table 4-21 Error value for different periods of 9-nodes..... 145

## LIST OF FIGURES

Figure 1-1 Architecture of Wireless Sensor Node.....	4
Figure 1-2 Structure of the Microcontroller.....	4
Figure 1-3 Different type of sensors .....	5
Figure 1-4 Distribution of the sensor nodes.....	6
Figure 1-5 ZigBee and IEEE 802.15.4 protocols.....	7
Figure 1-6 Different topologies .....	7
Figure 1-7 List of WSN applications .....	9
Figure 1-8 Different types of sensor nodes.....	10
Figure 1-9 Micaz mote and the block diagram .....	11
Figure 1-10 Top view of MIB520CB .....	13
Figure 2-1 Clock time for the sensor node.....	17
Figure 2-2 Time Synchronization Protocols .....	21
Figure 2-3 WSN graph with the links (left). TPSN, FTSP (center). RBS: (right).....	23
Figure 3-1 Continuous re-setting of node clock by consensus protocol.....	46
Figure 3-2 Flow chart of the Averaging Protocol.....	47
Figure 3-3 Network with N sensor nodes and L links .....	48
Figure 3-4 4-nodes with Grid Topology .....	50
Figure 3-5 4-nodes with Hexa Topology.....	52
Figure 3-6 4-nodes with Random Topology.....	53
Figure 3-7 9-nodes with Grid Topology .....	55
Figure 3-8 9-nodes with Hexa Topology.....	56
Figure 3-9 9-nodes with Random Topology.....	57
Figure 3-10 16-nodes with Grid Topology .....	58
Figure 3-11 16-nodes with Hexa Topology.....	59
Figure 3-12 16-nodes with Random Topology.....	60
Figure 3-13 Pseudo code of the Averaging Protocol.....	61
Figure 3-14 Time values for each node in 4-Nodes Grid Topology.....	63
Figure 3-15 Error values for each node in 4-Nodes Grid Topology.....	63
Figure 3-16 Time values for each node in 4-Nodes Hexa Topology.....	65
Figure 3-17 Error values for each node in 4-Nodes Hexa Topology.....	65
Figure 3-18 Time values for each node in 4-Nodes Random Topology.....	67
Figure 3-19 Error values for each node in 4-Nodes Random Topology.....	67
Figure 3-20 Time values for each node in 9-Nodes Grid Topology.....	69
Figure 3-21 Error values for each node in 9-Nodes Grid Topology.....	69
Figure 3-22 Time values for each node in 9-Nodes Hexa Topology.....	71
Figure 3-23 Error values for each node in 9-Nodes Hexa Topology.....	71
Figure 3-24 Time values for each node in 9-Nodes Random Topology.....	73
Figure 3-25 Error values for each node in 9-Nodes Random Topology.....	73

Figure 3-26	Time values for each node for a 4-Nodes Grid Topology .....	78
Figure 3-27	Error values for each node for a 4-Nodes Grid Topology .....	79
Figure 3-28	Time values for each node for a 9-Nodes Grid Topology .....	80
Figure 3-29	Error values for each node for a 9-Nodes Grid Topology .....	81
Figure 3-30	Time values for each node in 16-Grid Topology .....	82
Figure 3-31	Error values for each node in 16-Grid Topology .....	83
Figure 4-1	SC Concept.....	88
Figure 4-2	Error Curve.....	89
Figure 4-3	Flow chart of the modified SS-SC .....	90
Figure 4-4	Mathematical representation of the modified absolute SC .....	91
Figure 4-5	Flow Chart of the propose SC .....	93
Figure 4-6	Cost function for 9-nodes with Grid Distribution .....	96
Figure 4-7	Filter in time domain with $C = 1$ .....	97
Figure 4-8	Filter in frequency domain .....	97
Figure 4-9	Response of the rectangular pulse.....	98
Figure 4-10	Block Diagram of the Dip SC .....	99
Figure 4-11	Maximum and Average detected iterations by the 2-Filters .....	100
Figure 4-12	Maximum and Average error values for the 2-Filters.....	100
Figure 4-13	Simulation Error and the stopping locations for node 1 in 4-Grid.....	102
Figure 4-14	Simulation Error and the stopping locations for node 2 in 4-Grid.....	103
Figure 4-15	Simulation Error and the stopping locations for node 3 in 4-Grid.....	103
Figure 4-16	Simulation Error and the stopping locations for node 1 in 9-Grid.....	107
Figure 4-17	Simulation Error and the stopping locations for node 2 in 9-Grid.....	108
Figure 4-18	Simulation Error and the stopping locations for node 3 in 9-Grid.....	108
Figure 4-19	Simulation Error and the stopping locations for node 1 in 16-Grid.....	112
Figure 4-20	Simulation Error and the stopping locations for node 2 in 16-Grid.....	113
Figure 4-21	Simulation Error and the stopping locations for node 3 in 16-Grid.....	113
Figure 4-22	Practical error curve and the stopping locations for node 1 in 4-Grid .....	118
Figure 4-23	Practical error curve and the stopping locations for node 2 in 4-Grid .....	118
Figure 4-24	Practical error curve and the stopping locations for node 3 in 4-Grid .....	119
Figure 4-25	Real Time implementation for 4-nodes.....	123
Figure 4-26	Practical Time Values for each node in 4-Grid.....	124
Figure 4-27	Practical Error Values for each node in 4-Grid.....	124
Figure 4-28	Real Time implementation of 9-nodes.....	126
Figure 4-29	Practical Time Values for each node in 9-Grid.....	126
Figure 4-30	Practical Error Values for each node in 9-Grid.....	127
Figure 4-31	Real Time implementation of 16-nodes.....	129
Figure 4-32	Practical Time Values for each node in 16-Grid.....	129
Figure 4-33	Practical Error Values for each node in 16-Grid.....	130
Figure 4-34	Deviation error for Grid Topology.....	133

Figure 4-35 Deviation error for Hexa Topology.....	134
Figure 4-36 Deviation error for Random Topology .....	135
Figure 4-37 Deviation error for the practical results .....	136
Figure 4-38 Simulation and practical deviation error for 4-nodes Grid .....	137
Figure 4-39 Simulation and practical deviation error for 9-nodes Grid .....	138
Figure 4-40 Simulation and practical deviation error for 16-nodes Grid .....	139
Figure 4-41 Average error curve of different protocols with 9-Grid nodes .....	141
Figure 4-42 Maximum error curve of different protocols with 9-Grid nodes .....	141
Figure A-1 Averaging Protocol with Steady State Stopping Criterion.....	156
Figure A-2 Averaging Protocol with Dip Stopping Criterion .....	157

## **LIST OF ABBREVIATIONS**

- WSN:** Wireless Sensor Network.
- AP:** Average Consensus Time Synchronization Protocol.
- DSN:** Distributed Sensor Networks.
- DARPA:** Defense Advanced Research Projects Agency.
- ARPANET:** Advanced Research Projects Agency Network.
- FPGA:** Field Programmable Gate Arrays.
- ADC:** Analog to Digital Converter.
- RAM:** Random Access Memory.
- I/O:** Input/Output.
- SPI:** Serial Peripheral Interface.
- DC:** Direct Current.
- IEEE:** Institute of Electrical and Electronics Engineers.
- MAC:** Medium Access Control.
- NesC:** Network embedded systems C.
- TinyOS:** Tiny Operating System.
- USB:** Universal Serial Bus.
- EM:** Electro Magnetic.
- API:** Application Programming Interface.
- TDMA:** Time Division Multiple Access.
- TPSN:** Time-synchronization Protocol for Sensor.
- FTSP:** Flooding Time Synchronization Protocol.
- LTS:** Lightweight Tree-based Synchronization.
- DMTS:** Delay Measurement Time Synchronization.
- FBS:** Feedback-Based Synchronization.

**NTP:** Network Time Protocol.

**CPU:** Central Processing Unit.

**PI:** Proportional Integral.

**PBS:** Pairwise Broadcast Synchronization.

**RBS:** Reference Broadcast Synchronization.

**HRTS:** Hierarchy Referencing Time Synchronization.

**PCS:** Probabilistic Clock Synchronization.

**ITR:** Individual Time Request.

**CCS:** Consensus Clock Synchronization.

**TDP:** Time Diffusion Protocol.

**RFA:** Reachback Firefly Algorithm.

**GTSP:** Gradient Time Synchronization Protocol.

**EGSyn:** External Gradient Time Synchronization Protocol.

**ATS:** Average Time-Sync Protocol.

**MTS:** Maximum Time Synchronization Protocol.

**WMTS:** Weighted Maximum Time Synchronization Protocol.

**TSMA:** Time synchronization Protocol using the maximum and average values.

**EGTSP:** Energy-Efficient Gradient Time Synchronization Protocol.

**DTSC:** Distributed Time Synchronization Protocol.

**PLLs:** Phase Locked Loop.

**SC:** Stopping Criterion.

**2LTSP:** Long Term and Large Scale Time Synchronization Protocol.

**DSSS:** Direct Sequence Spread Spectrum.

**O-QPSK:** Orthogonal Quadrature Phase Shift Keying.

**UTC:** Coordinated Universal Time.



## ABSTRACT

**Full Name** : Ibrahim Ahmed Abdallah Nemer  
**Thesis Title** : A Distributed Time Average Synchronization Protocol for Wireless Sensor Networks  
**Major Field** : Telecommunication  
**Date of Degree** : May 2015

The exact and effective operation for most WSN applications need synchronized notion of time. In this thesis, we introduce a new control distributed time synchronization protocol, called Average Consensus Time Synchronization Protocol (AP), in respect of synchronizing the sensor nodes in Wireless Sensor Networks (WSNs). AP protocol is based on simple operations: sum and division to evaluate the average time of the neighbours for each node in every communication cycle; this update for each cycle is represented by an iterative process. Our aim is to stop this process at the minimum error with less number of communication cycles; since the error curve of each node for this protocol has two regions: Dip and Steady State regions, where the minimum errors; locate in the dip region. So, we propose a stopping creation that consists from filtration stage that tracks the local time values and detects the minimal value, and then stops at this value for each node. We present an evaluation of this strategy on a testbed setup including 9 and 16 Micaz sensor nodes to highlight the benefits of this approach in terms of improved dip error and scalability as compared to existing synchronization protocols. We show through real-world experiments and MATLAB simulations that AP protocol has multiple advantages over the previous protocols that mentioned in the literature, as using only local information, simple with little communication overhead, the code size of

this protocol is independent on the network size and topology, scalable, power efficient and less error value with less communication cycle.

|

## ملخص الرسالة

الاسم الكامل: إبراهيم أحمد عبد الله نمر

عنوان الرسالة: بروتوكول فعال للتوافق الزمني عن طريق حساب متوسط الزمن لشبكات الاستشعار اللاسلكية.

التخصص: هندسة كهربائية – اتصالات.

تاريخ الدرجة العلمية: أيار 2015

العمليات الدقيقة و الفعالة لمعظم التطبيقات في شبكات الاستشعار اللاسلكية يحتاج أن يكون هناك تزامن لعامل الوقت في عمل جميع عقد الاستشعار. في هذه الرسالة نقدم بروتوكول جديد للتحكم بتوافق الزمن ويطلق على هذا البروتوكول بروتوكول متوسط اجماع التوافق الزمني، فيما يتعلق بمزامنة عقد الاستشعار في شبكات الاستشعار اللاسلكية. هذا البروتوكول يعتمد على عمليات بسيطة: كالجمع و القسمة لحساب متوسط الزمن لجميع العقد المجاورة لكل عقدة على حدة في كل دورة اتصال؛ هذا التحديث في قيمة الزمن لكل دورة يتمثل بعملية تكرارية. هدفنا هو كيفية التوقف عند أقل قيمة خطأ و بأقل عدد من الدورات؛ و تمثيل الخطأ في الزمن لكل عقدة يحتوي على منطقتين: منطقة الانحراف و منطقة الثبات، و أقل نسبة خطأ تقع دائما في منطقة الانحراف. لذلك، نقدم طريقة للتوقف عند هذه قيم الأقل خطأ، و هذه الطريقة تتكون من مرحلة الترشيح التي تعمل على فحص قيم الزمن في كل دورة حتى تجد القيم الأقل خطأ و تتوقف عندها عملية التحديث. و نعرض في هذه الرسالة تقييما تجريبيا لهذا البروتوكول باستخدام 9 و 16 عقدة استشعار لنرى أهمية هذا البروتوكول اعتمادا على قيمة الخطأ و امكانية زيادة حجم الشبكة مع فعالية هذا البروتوكول. و أيضا نعرض بالإضافة للتجارب محاكاة باستخدام برنامج الماتلاب لدراسة البروتوكول بشكل معمق و مقارنته بما هو موجود بالسابق، اعتمادا على الزمن المحلي لكل عقدة، بسيط مع قليل من الضغوط على العقد وقت العمل، حجم البرنامج لا يعتمد على حجم الشبكة و توزيعها، قابل لزيادة حجم الشبكة، و أقل استهلاك للطاقة بأقل قيمة خطأ و أقل عدد من الدورات.

# CHAPTER ONE

## **Introduction to Wireless Sensor Networks**

This chapter is an introduction to wireless sensor networks (WSNs). First, it shows the flow of the communication systems; starting with the wired networks and then wireless networks. Additionally, it compares the two networks and shows why the wireless network is more suitable and preferable in the communication system over the wired networks. After that, it introduces the sensor network in wireless communication. Also it shows the construction of sensor network (SN) and the main advantages and disadvantages of using WSNs. Next, it shows some of the network applications, topologies, types, and investigates the harsh environments in WSNs.

### **1.1 Background**

Each day comes with a tremendous improvement and development in the wireless communication technology and thus the wireless market is growing rapidly. As the demand grows, the customers' expectations raise the leading to an equivalent increase in the future challenges. The wireless technology is expanded beyond voice applications to include many other types of applications such as mobile e-commerce, real-time internet, audio and as well as those can respond to the changing demands and also make the human life safer, more accurate and easier.

Due to the fact that this technology has passed many magnificent steps and proved its efficiency, researchers and engineers decided that sensor nodes need to be used in this technology. This conclusion came up because sensor nodes have the ability to sense different physical environments and convert the information into processable data that can be used in various systems to provide information and/or even act depending on the information that the sensor nodes have provided to the overall network. The required information received from various remote places at which physical transmission lines are not reliable or sometimes almost impossible such as a volcano, underwater and unpopulated areas. This called to the need for an efficient wireless communication system that provides accessibility to harsh environments. The solution is based on wireless networks that can transmit information efficiently and effectively. This is achieved by employing these types of networks that can provide information about physical or environmental conditions using various WSNs.

The first generation of WSNs returns to the Distributed Sensor Networks (DSN) project of the Defense Advanced Research Projects Agency (DARPA) in 1980. After that, ARPANET (Advanced Research Projects Agency Network) constructed a group from 200 hosts through universities and research centers around the world. DSN project were searched to achieve a network with multiple sensor nodes that communicate with each other with less cost. WSNs faced some challenges in the 21<sup>th</sup> century because of the need of reliable power supplies like that incorporated in the traditional wireless systems. This means that it is inefficient to use the predefined wireless protocols and algorithms because these networks require a reliable power supply which is absent in WSNs. Due to

this fact, researchers were motivated to introduce different protocols that consider energy, time and security as important parameters in WSNs.

WSNs are used in the numerous sensitive applications such as health, military, home, environments, and industrial. However, it is the task of today's researchers to increase the WSNs' ability to support high data rate, low power consumption, security, and reliability. These goals can be discussed from different points of view since WSNs depend on variable parameters and conditions that are involved in routing, synchronization and data transmission protocols.

## **1.2 WSNs**

### **1.2.1 Architecture of the Node**

WSNs consist of multiple devices called sensor nodes that spread over the required area. The distribution of these nodes depends on the application and the required coverage area. Usually, the network may contain small number of sensor nodes or large number up to hundreds of sensor nodes. Each of these sensor nodes consists of transducer or sensor, radio transceiver with wireless capabilities, low complexity processing units, and power supply supported with recharging capabilities or contains harvesting device. Every node consists of sensing, processing, communication, and power subsystems as shown in Figure 1-1.

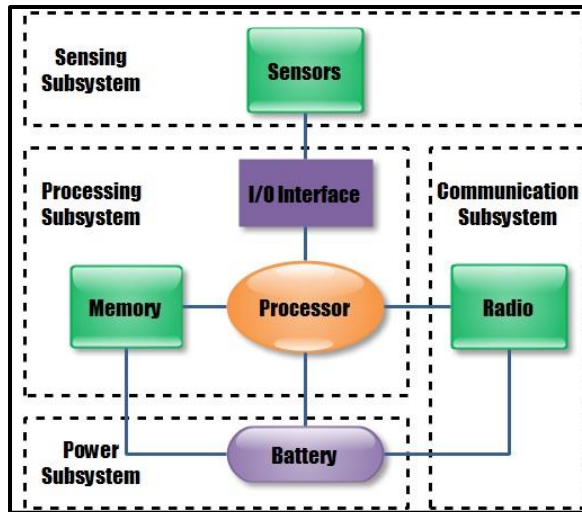


Figure 1-1 Architecture of Wireless Sensor Node

The processor subsystem is the central element in WSN and the choice of a processor specifies the tradeoff between flexibility and efficiency which is related to energy and performance. The processors have many components which include: microcontrollers, digital signal processors, application-specific integrated circuits, and field programmable gate arrays (FPGA) as shown in Figure 1-2.

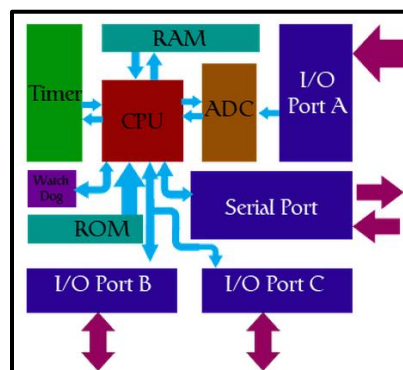


Figure 1-2 Structure of the Microcontroller

The sensing subsystem consists of more than one analog sensor as shown in Figure 1-3. Sensors are equipped with an analog or digital output for reading the sensor values. Some of these sensors have their own built-in analog-to-digital convertor (ADC) which can be directly connected with the processor through a standard chip-to-chip protocol.

Most of microcontrollers have one or more internal ADCs to interface the analog devices. Modern microcontrollers integrate flash storage, RAM, ADC, and digital I/O onto a single integrated circuit. When selecting a microcontroller family, many factors should be considered such as energy consumption, support for peripherals, voltage requirements, cost, and number of external components required.

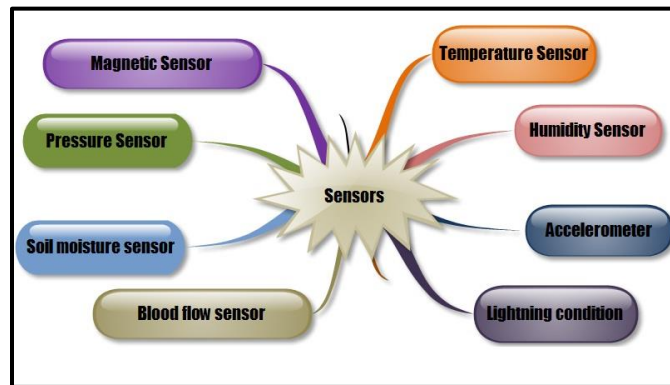


Figure 1-3 Different type of sensors

The communication subsystem connected to the processor subsystem by using the serial port interface (SPI) bus. The communication subsystem is the most energy intensive subsystem and the power consumption should be managed. Most of the commercially available transceivers provide a controlling functionality to switch the transceiver between various operation levels such as active, idle and sleep state.

The power subsystem provides the direct current (DC) power to all subsystems and their components. This subsystem comprises the energy storage, voltage regulation, and optionally energy scavenging unit. The energy is usually stored inside a primary battery. Additionally, some equipment could help in providing energy to the sensor nodes to increase the life time of the network such as those equipment that are exposed to the sun in order to provide power supply to the system.



## 1.2.2 Protocols in WSNs

Generally, sensor nodes collect and process the data that are sensed from the surrounding area. This data can be transmitted to a base station or sink node in a centralized network, or can be processed rather than sending it to the base station as in distributed network as shown in Figure 1-4. Different kinds of communication channels such as microwave, radio links and satellite links can be used to transmit and extract the acquired data from the WSNs [1].



Figure 1-4 Distribution of the sensor nodes

There are different standards that use in WSNs like: IEEE 802.15.4 and ZigBee as described in Figure 1-5. IEEE 802.15.4 standard used in low data rate networks that cover small area. It is a power and efficient standard. ZigBee operates at low data rate and low power consumption. For upper layers (application and network), ZigBee is considered as the main protocol, while for the lower layers (MAC and physical) IEEE 802.15.4 is considered as the main protocol.

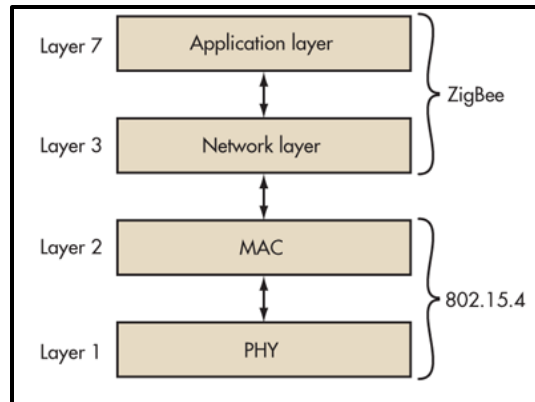


Figure 1-5 ZigBee and IEEE 802.15.4 protocols

### 1.2.3 Topologies in WSNs

WSNs can be categorized in two types. The first one is centralized networks, while the second one is distributed networks. Similarly, the sensor nodes in WSNs can be deployed in different topologies depending on the used application. For example, linear, random, grid, and ring topologies as indicated in Figure 1-6. Each topology designs to serve a specific purpose in WSNs.

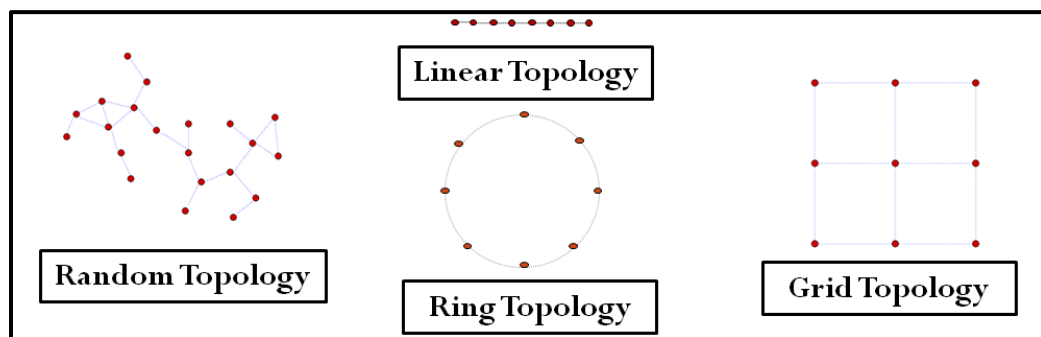


Figure 1-6 Different topologies

## 1.3 Importance of Using WSNs and Some Applications

There are different factors that force designers to use WSNs frequently in the communication/networking part such as:

- Network with less energy consumption.
- Monitoring area with no infrastructure.
- Reducing the cabling costs.
- Flexibility, deployment and scalability.

WSNs are usually designed with the main purpose of measuring different physical variables or tracking events in different fields such as military, automation and civil applications (monitoring and tracking animals and humans), battlefield surveillance, monitoring the forests against fire outbreaks. They are also applied for different alarming systems in monitoring the oil and gas lines as well as detecting lines leakages. In addition to home automation and health care applications [1-3] as in Figure 1-7, there are other applications that are using WSNs such as:

- Metrological monitoring that studies and supervises storms, flooding, volcanoes, and weather forecast.
- Geological monitoring that studies several geological phenomena that have a future look about the disasters that may happen such as landslide, earthquake, and volcanoes.
- Pollution monitoring that gives speed, accuracy, and can specify the exact place of the pollutions including water pollution, noise pollution, and radioactive.
- Energy monitoring that deals with reducing the wasted energy.

- Health care monitoring that uses different sensors such as: blood pressure sensors, skin temperature sensors, and blood oxygen level sensors.

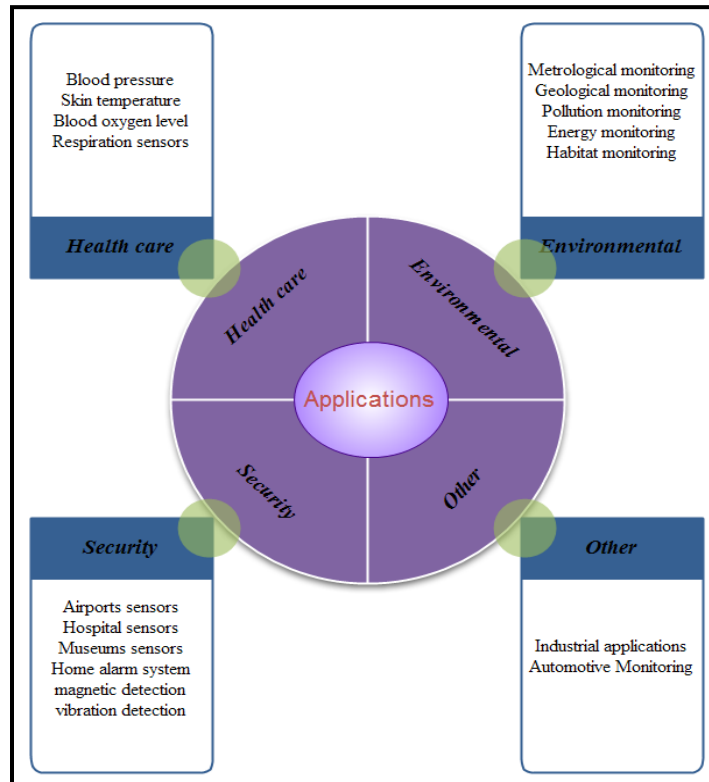


Figure 1-7 List of WSN applications

In some of these applications such as data fusion, human and animal tracking, speed estimation, the network needs to know the time of all nodes in order to determine the time occurrence of the events. Exact values of time can help in saving the energy by reducing the guard times that are attached to the transmitted packets among nodes. This is mainly true for the networks that use duty-cycling techniques and switch off the radio to reduce the energy consumption.

## 1.4 Types of Sensor Nodes

There are different types of sensor nodes that had been invented during the previous 20 years up to now. Each type has different properties that differ from one to other type such

as range, frequency, data rate, cost... etc. Figure 1-8 shows some of these sensor nodes and their specifications.

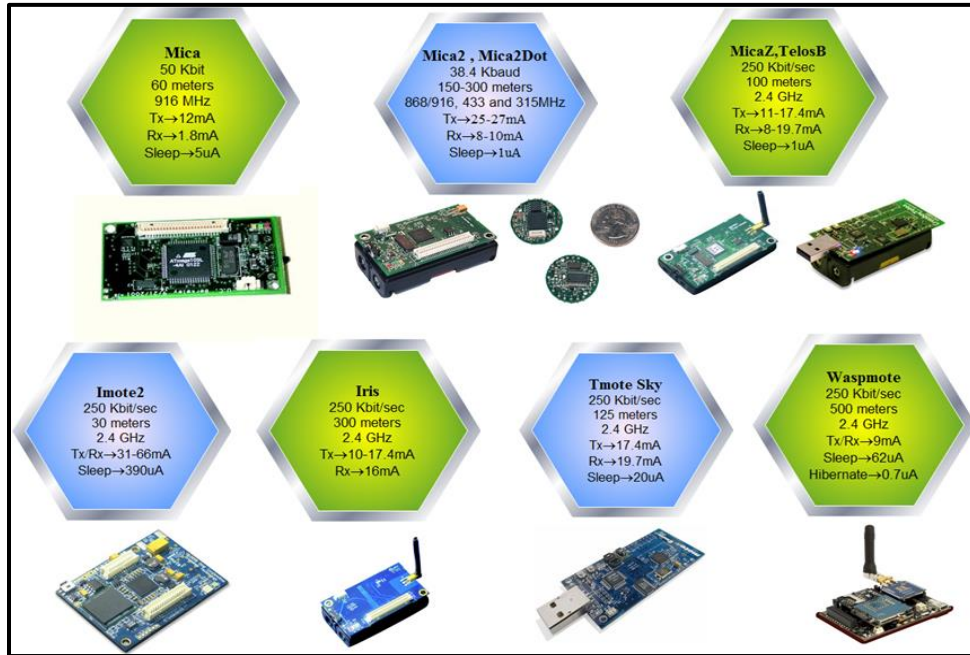


Figure 1-8 Different types of sensor nodes

There are different sensor nodes that are used in the research fields, such as Micaz, TelosB and IRIS. These nodes share the same operating system (TinyOS) and they use nesC language for implementing WSN application. Mica node consists from three main components they are; MPR2400 (Micaz mote), MIB520 gateway, and sensing boards. The following sections describe the hardware and software parts of the Micaz node.

### 1.4.1 TinyOS Overview

TinyOS is an event driven operating system designed for low-power wireless devices, specifically for sensor networks. TinyOS is written in nesC language like C programming language that is designed for structured component based applications. Applications that are written in nesC language are built using interfaces and components that encourage the

hardware abstraction and reuse. Components are wired together using the configuration modules that link specific implementation to dependencies. Using this approach reduces the application program size and overall memory, which is usually a priority for embedded devices.

The TinyOS operating system is an open source and it is developed and supported by different companies and universities. TinyOS supports different platforms including the Micaz mote. Each new release of TinyOS adds new support directory for wireless sensor based platforms.

### 1.4.2 Micaz Mote Platform

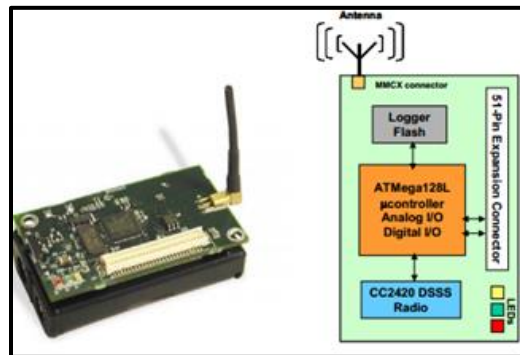


Figure 1-9 Micaz mote and the block diagram

Micaz is the latest generation of motes from Memsic. Figure 1-9 shows Micaz mote which is composed of different hardware components such as processor, radio transceiver, and external flash (logger). The MPR2400 (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant and ZigBee ready radio frequency transceiver integrated with an Atmega128L micro-controller that described in Table 1-1. It has 51 pin I/O connectors, and serial flash memory is used (all MICA application, software and sensor boards are compatible with the MPR2400).

Table 1-1 Technical specifications of Micaz mote

<b>Micaz</b>		
	<b>MPR2400CA</b>	<b>Description</b>
<b>Size (mm)</b>	58 X 32 X 7	The weight and size does not include batteries.
<b>Weight (grams)</b>	18	
<b>Connector</b>	51-pin	The Micaz mote is connected to the sensor board via this connector.
<b>Power</b>	2 X AA batteries	The batteries can be rechargeable.
<b>User Interface</b>	3 LEDs (red, green, yellow)	These lights indicate when data is received, sent, synchronized.
<b>RF Transceiver</b>		
<b>Frequency band</b>	2400 MHz – 2483.5 MHz	Data is transmitted using this frequency band.
<b>TX data rate</b>	250 kbps	Maximum data rate allowed.
<b>Indoor Range</b>	20m -30m	The distance will suffice this project.

### 1.4.3 MIB520 USB Interface Board

MIB520 provides USB connectivity to the Micaz motes for communication and in-system programming. It supplies power to the devices through USB bus. MIB520CB has a male connector as shown in Figure 1-10 and its specifications are described in Table 1-2. Usually, this board connects to Micaz mote to construct the base station node that is connected to the PC for recording the received data.

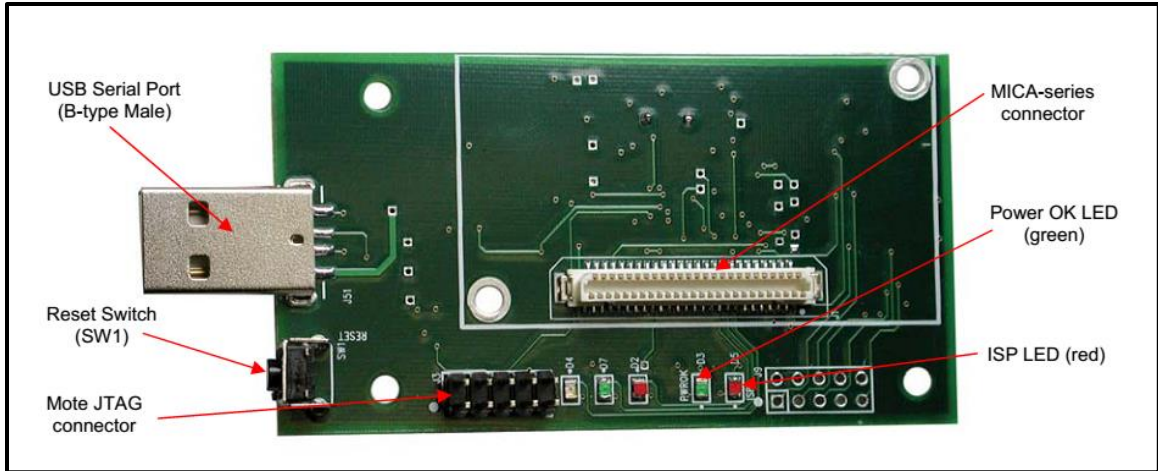


Figure 1-10 Top view of MIB520CB

Table 1-2 Technical specifications of MIB520CB

<b>Programming Board</b>		
	MIB520	Description
<b>USB Interface</b>		
<b>Baud Rate</b>	57.6K	57.6K is a typical rate for regular USB interface.
<b>Connection</b>	Male to Female	This programming board is connected to the computer via this cable.
<b>Cable</b>	USB	
<b>Mote Interface</b>		
<b>Connector</b>	51-pin	This programming board is connected to the mote via this connector.

## 1.5 WSNs in Harsh Environments

WSNs are usually deployed in harsh environments and unstable conditions at which regular communication may not be practical. This is challenging for the applications that consider time as an important factor in their operation. The harsh environment is an unpredictable and uncontrolled environment where environmental factors such as vast fluctuation in temperatures, rain, vibration, humidity, chemicals, electrical shock, pressure, physical damage, etc. may affect the normal operation of the nodes [4]. WSNs



along with these obstacles coupled with the goal to achieve other primary requirements including being computationally light, scalable, and robust to node and link failure (sometimes do not require a master or controlling node). In this case, using synchronization protocols may help to increase the packet delivery with minimum errors [5].

In most cases, sensor network architectures have a task to maximize the performance of the sensor network by increasing the reliability of the network, decreasing latency, increasing power efficiency, and increasing lifetime. Moreover, designers are supposed to take care from the variations in the network topology either permanently such as shutting down the node, or temporarily like changing the status of the mode.

Designing the network with specific components requires some knowledge about the performance indicators under specified energy constraints and environmental conditions[6]. Additionally, the performance of the WSNs in electromagnetically and physically harsh environments such as in industrial floors will be affected by these conditions. Wireless communication systems are constructed in industrial environments to transmit important parameters for monitoring purposes. This kind of transmission is preferred over the wired one because it has less cost in installing, maintaining, easier troubleshooting, and fast speed [7]. Performance evaluation in WSNs uses different measures; (i) network lifetime, (ii) energy costs, (iii) survival rate of sensor nodes, (iv) data received, and (v) accurate received time.

## 1.6 Summary

WSNs are considered as one of the technologies that are used to sense different parameters such as light, pressure, temperature...etc., then process the collected data and send it to the base station node to take the suitable action accordingly. There are different types of the sensor nodes as discussed in this chapter. Micaz node is one of the most important types that mostly used in research fields. In this work, we deployed multiple of Micaz nodes to serve a certain task.

As we have seen in the WSNs introduction, it has been observed that some applications are sensitive to the transmission time between different nodes. Several researchers investigated the issue of minimizing power consumption and increasing the efficiency of the WSNs depending on the time parameter. This was done by developing different time synchronization protocols that achieve less value of errors with less time using simple operations. Chapter 2 introduces the time synchronization concept for this kind of networks. ]

## CHAPTER TWO

### Time Synchronization in WSNs

This chapter discusses the clock model of the sensor node and the time synchronization concept in WSNs. First, it shows the representation of the clock model for any sensor node and the most important parameters that describes the clock model. Then, it shows the importance of the synchronization process in WSNs and what are the reasons that cause clocks to lose synchronization with each other. After that, it mentions different time synchronization protocols and divides them into different groups depending on the architectures of the WSNs. Next, it shows the previous researches in the literature that already have been done regarding time synchronization protocols. Finally, the contributions of this work are presented.

#### 2.1 Clock Model for the Sensor Node

WSNs consist of multiple sensor nodes that communicate with each other to serve a certain purpose. Each sensor node  $i$  is equipped with a clock that depends on both hardware and software parts. A clock consists of an oscillator and counter that is decremented by every oscillation of the quartz crystal oscillator. When the counter back to 0, it is reset to the original value and an interrupt is generated. Each interrupt called (clock tick) increments software clock (another counter); software clock can be read using application programming interface (API). Software clock provides the local time with  $\tau_i(t)$  being the clock reading at real time  $t$  is given by:

$$\tau_i(t) = a_i t + b_i \quad 2.1$$

where  $a_i$  represents the hardware skew/drift that shows the clock speed,  $b_i$  is the clock offset and  $t$  represents the real time for all nodes  $i = 1, 2, \dots, N$ .

The clock offset is defined as the difference between the local times of two nodes. Additionally, the clock drift (skew) is defined as the difference in frequencies of two clocks as shown in Figure 2-1.

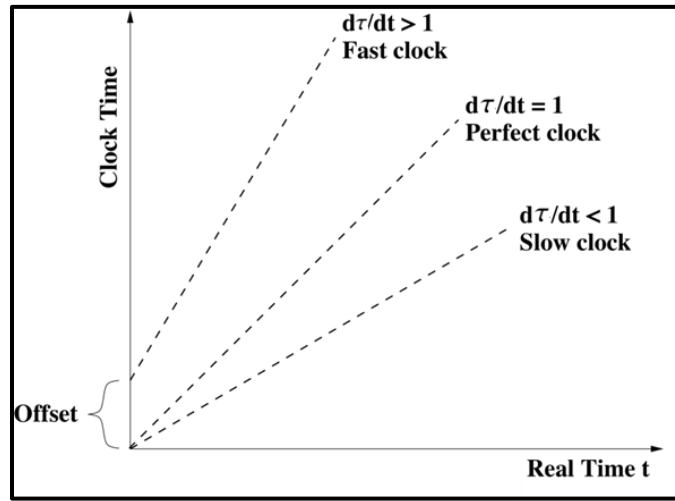


Figure 2-1 Clock time for the sensor node

The relative skew between node  $i$  and node  $j$  is defined by the ratio between the skew of node  $i$  and the skew of node  $j$  and can be evaluated using this equation:

$$\tau_i(t) = \frac{a_i}{a_j} \tau_j(t) + \left( b_i - \frac{a_i}{a_j} b_j \right) = a_{ji} \tau_j(t) + b_{ji} \quad 2.2$$

Usually, time synchronization depends on the method that is used to synchronize all nodes with the master clock:

$$\tau_c(t) = a_c t + b_c \quad 2.3$$

From this, all nodes will converge to the same clock and the value of this clock depends on the master node. Usually, when ( $\text{offset} \neq 0$  or  $\text{drift} \neq 1$ ) at this point (the nodes are not synchronized to each other). Time Synchronization can be defined as a problem that results from the time differences in the internal clock of several sensor nodes in same network. This difference may be caused from the drift or the offset value which has a unique effect for each node.

## **2.2 Importance of Synchronization in WSNs**

Synchronization is an important factor for different applications that require an accurate mapping of the gathered data between sensor nodes with the timestamps as in tracking and surveillance. However, usually some nodes suffer from missing the synchronization and hence, this will cause some drift/skew on the clock values of these sensor nodes. Drift values should be minimized to a reasonable level or completely eliminated if possible in such applications that consider time as important factor that affect the operation performance [3]. Clocks of the nodes may be incompatible and have different values due to several reasons they are:

- Clocks may drift due to several harsh environment changes, such as temperature, pressure, battery voltage ... etc.
- The construction of the networks change from time to time.
- Clocks of the sensor nodes start with different speeds due to the use of non-ideal oscillators. Additionally, each clock has its own starting time. This will create differences in their local times. Usually, this changes the initial values of these clocks

and influence on the transmission and reception packets with different timestamps in the real time applications.

Nowadays, simple synchronization algorithms are not applicable to deploy or work for most of the applications especially in the dense networks that need an accurate time values to do their operations. This is due to several reasons as follows:

- Synchronization in sensor network depends only on some nodes such as reference node (this will increase the failure of the WSN).
- Achieving high precision in the synchronization process needs to use an expensive clock or complex algorithms.
- Centralized algorithms make the network not scalable; errors will be cumulative when the number of clocks increases.
- Usually node should be operated with self-managing, low-cost construction, lightweight, and self-stabilizing.
- Increase the lifetime can be achieved by using some power saving techniques such as:
  - I. Sleep scheduling: It is one of the most important factors that decrease the consumption energy by switching off the radios of the nodes when they are not in the active mode. Then, when these nodes return to active mode, they should agree on the transmission times to keep this network synchronized, work correctly, and efficiently.

- II. Medium-access: TDMA medium-access protocols need those sensor nodes to be synchronized with each other. To do that, it is necessary to assign time slots to minimize the collisions within the network.
- III. Coordinated signal processing: Time stamps are required to specify which data from different nodes can be aggregated in the network.

There are many applications that are sensitive to the time factor such as tracking objects, home monitoring [8], scheduling, time division multiple access (TDMA) [9], and leakage control in power lines. These applications need to know times of each node to measure the elapsed time, schedule wakeups, and compare time coordinates of sensor readings with different nodes. Therefore, the time synchronization design should depend on the clock readings of the whole system [10],[11].

### **2.3 Time Synchronization Protocols**

Time Synchronization protocols can be divided into two groups. The first one is synchronous protocols at which all nodes update their values at the same time [1]. The second one is asynchronous protocols at which nodes update their values at different time [2]. Large WSNs requires complex time synchronization algorithms especially, if there is a dynamic change in the network from time to time where the communication in WSNs is unreliable and suffering from packet losses. Accordingly, there is a time delay between any two clocks and this value of delay results from the accessing time (that requires for reading the clock value), propagation delays, and received delays. Different time synchronization protocols have been proposed to solve the synchronization problem in WSNs and these protocols can be divided into three groups based on the architectures of

WSNs (tree structure [12, 13], cluster structure [14, 15] and fully distributed protocols [16-18]) as in Figure 2-2. All these protocols will be discussed in section 2.3.

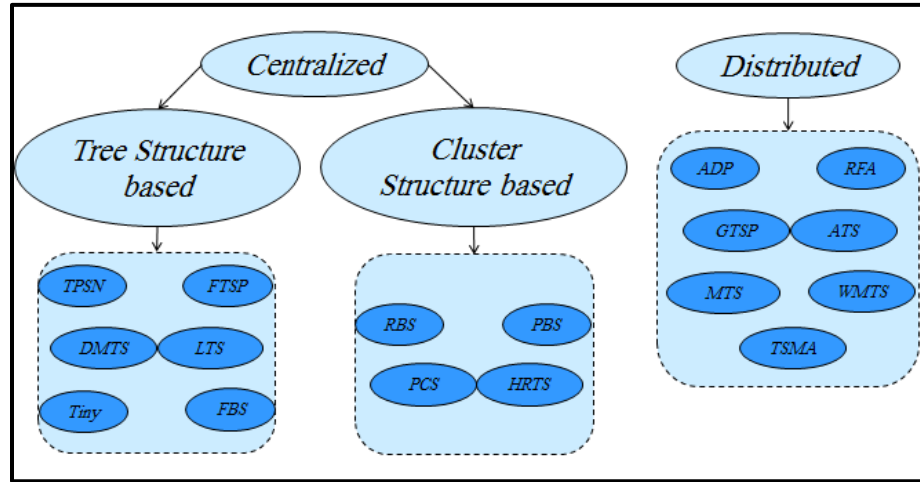


Figure 2-2 Time Synchronization Protocols

### 2.3.1 Tree Structure Protocols

The first group is based on the hierarchical structure that is used to build the network. Usually, one node within the network is chosen as a reference node proceeded by a spanning tree which is created with respect to this reference node. Subsequently, each node synchronizes with this parent node by compensating the skew and offset value of each node depending on the clock of the parent node. There are different time synchronization protocols represent this group such as time-synchronization protocol for sensor network (TPSN)[12], flooding time synchronization protocol (FTSP)[13], lightweight tree-based synchronization (LTS)[19], delay measurement time synchronization (DMTS)[17], feedback-based synchronization (FBS)[20], and tiny-sync[21]. All these protocols are described as follow:



TPSN [12] represents sender-receiver based time synchronization protocol in WSNs and it is a centralized synchronization protocol. This scheme consists of two steps they are discovery step and synchronization step. In the discovery step, TPSN builds the network with spanning tree shape where each node knows its level and its parent. Level 0 is returned to one node and named as a root node and the responsibility of this node is to build the tree by triggering level discovery step. (i) Root node sends a level discovery packet with its level 0 to all neighbors, then the nodes that receive this packet within one hop, set their level to 1, parent to 0 and send another level discovery packet with its level 1. (ii) They wait for a random time between two sending steps to avoid the collisions and errors and then the process continues for other nodes. On the other hand, for the synchronization step; (i) Node  $i$  builds a synchronization message and sends this message to the operating system and the network stack for transmission. (ii) Before starting the transmission, the message is labeled with time  $T1$  and transmitted over the medium. (iii) Message will be forwarded to node  $j$  with label  $T2$  after taking care of the propagation delay and packet transmission time to prevent the errors from occurring again. (iv) Node  $j$  builds synchronization acknowledgment message and sends it to the operating system and network stack. (v) Message is labeled with  $T3$  and then delivered. (vi) Node  $i$  receives this message with label  $T4$ . (vii) Node  $i$  estimates the clock offset and fixes its clock using a specific relation between these parameters. The advantages of using TPSN are that scalable and synchronization precision is suitable with the variation in the network and TPSN has less complexity compared to other protocols such as NTP [12, 22]. On the other hand, TPSN has the following drawbacks [12, 22]; it suffers from the

link/node failures, not energy conserver, not suitable for the networks that have movable nodes since it depends on tree-based structure, and not suitable for multi-hop networks.

FTSP is an ad-hoc, multi-hop time synchronization algorithm for WSNs. In FTSP [13], the node with a low value of ID is selected as a root node to act as a reference time for other nodes. This root node periodically floods a synchronization packet with its local time to the network. Other nodes will receive this packet and save the incoming timestamp and the arrival time of this packet and then broadcast this packet to all the neighboring nodes with the updated values. These timestamps are normalized by subtracting the latency value from the receiver side and then a linear regression operation will be used to estimate the clock drift. This algorithm achieves higher accuracy by using timestamp of the messages at low layers of the network stack and removing the access time. Figure 2-3 shows the differences between this and the previous protocols.

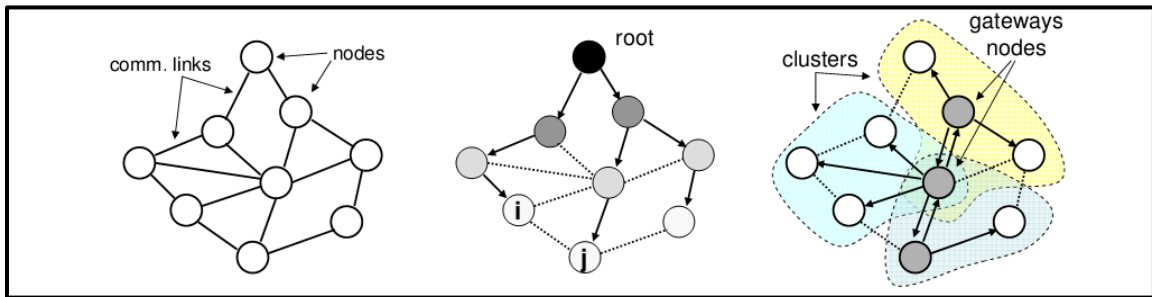


Figure 2-3 WSN graph with the links (left). TPSN, FTSP (center). RBS: (right).

Lightweight tree-based synchronization (LTS) protocol discussed in [19]; is presented to achieve a reasonable level of accuracy while using reasonable amount of computational resources like memory space and CPU time. The author divided LTS into two categories; centralized and decentralized. In the centralized, each round starts by only one node with a certain frequency whereas in the decentralized, each node can start the synchronization.

LTS algorithm uses the search to construct the tree-based structure for the whole network. Tree nodes share the synchronization data with each other. The drawback of this algorithm is that the accuracy of the synchronization decreases with increasing depth of tree and this will increase the error value for each node.

DMTS [17] collects different concepts at the same time such as master-slave synchronization, sender-receiver synchronization and clock correction approach. This protocol was created to avoid the round trip time estimation in the previous protocols. DMTS synchronizes the transmitter with multiple receivers at the same time with less number of packets when compared to RBS. In this protocol, the leader node is selected as time master and broadcasts its time. All receivers estimate the delay value and set their time the same as the master time. All nodes that receive this packet can synchronize with this leader. DMTS has some advantages like [17, 22]; computational complexity is low and energy efficiency is high. On the other hand, one of the drawbacks of DMTS protocol [17, 22] is that it uses only low frequency external clocks.

Jiming in [20] proposed a time synchronization algorithm called feedback-based synchronization that considers the synchronization problem is a closed-loop control problem and using proportional-integral (PI) controller to compensate the drift of clock that results from the internal and external factors. The accuracy of this algorithm depends on the response and overshoot time. This algorithm needs a reference node and its tree-based synchronization which suffers from link and node failures.

Tiny-sync and mini-sync presented in [21]; it depends on a set of data points, where each point is collected by two-way message exchange and consisted of two constraints which

are bounded by the offset and the skew parameters. Increasing the number of the data points will increase precision of the estimation bounds of the two parameters. The computational complexity of the tiny-sync algorithm is low because it is dependent on the specification of only four points with few operations. The mini-sync algorithm has improved accuracy greater than tiny-sync, which is achieved at a small computational cost. This algorithm has an accurate offset and drift information together with tight, deterministic bound, accuracy, low computation and storage complexity, insensitivity to communication errors and each clock can be approximated by an oscillator with fixed frequency.

Generally, it is easy to implement the tree-protocols. However, these protocols have different drawbacks such as the high overhead behind constructing the whole tree structure, not suitable in the dynamic topology, and need more time and overhead when there is a new node added to the network (that requires building new tree structure for the network). Additionally, when there are two nodes close to each other on different branches of the tree, this will cause a high difference in their clocks.

### **2.3.2 Cluster Structure Protocols**

In the second group (cluster structure), sensor nodes are divided into subgroups called clusters named in regard to their locations. Each cluster elects a leader node called cluster-head node. All nodes in the same cluster synchronize only with the cluster-head node and all cluster-heads synchronize with each other. There are many protocols that follow this group such as pairwise broadcast synchronization (PBS) [23], reference broadcast synchronization (RBS) [15], hierarchy referencing time synchronization

(HRTS) [24], and probabilistic clock synchronization (PCS) [14]. These protocols described as follow:

In Reference broadcast synchronization (RBS) protocol, the node broadcasts multiple reference beacons to all neighboring nodes. The main advantage of this method (receiver–receiver approach) is that it has better precision in synchronization and allows the nodes to construct the local timescales comparing with the previous protocols. RBS is reliable and flexible since there is no leader election procedure or multi-hop synchronization protocols that are needed. In RBS, the nodes periodically broadcast their own time and justify their own clock as the received time from other nodes. However, for RBS to work in all conditions, the speed and accuracy depend on the network topology [25]. On the other hand, RBS suffers from high overhead when dividing the network into clusters and electing the reference node for each cluster. This will cause failure in the nodes [26]. There are several advantages for using RBS protocol: minimizing errors by decoupling the sender from the receivers, clock offset and skew are estimated separately to minimize the interferences, minimize the energy waste by using Post-facto synchronization, support Multi-hop, applicable to both wired and wireless networks and timescales can be edited and corrected. On the other hand, the disadvantages include; using this protocol is not suitable for point-to-point (P2P) networks, convergence time is high due to the number of messages exchanges and the sender is kept without synchronization.

TSync in [24] is divided into two categories similar to LTS that discussed above, the centralized, called the hierarchical referencing time synchronization (HRTS) protocol and the decentralized one called the individual time request (ITR) protocol. The HRTS

protocol combines the concept of hierarchical synchronization in addition to the receiver-to-receiver synchronization. The researchers have improved the performance of the two protocols by devoting the MAC-layer for the synchronization process. Unlike in the HRTS protocol where the node cannot start the synchronization, in ITR any node can start the synchronization operation. HRTS can achieve accuracy close to the RBS extension with decreasing number of exchanged messages. ITR is a worst algorithm compared to HRTS and RBS especially in multi-hop synchronization.

Arvind in [14] presented a probabilistic clock synchronization (PCS) algorithm for wired networks. This is a new version of RBS protocol for supplying probabilistic clock synchronization. There are many deterministic algorithms which have an upper bound of error in clock offset estimation. When the network is badly constrained, the accuracy of large amount of messages during the synchronization phase will be affected. On the other hand, PCS provides good precision with low complexity and less overhead when compared to the deterministic algorithms. Elson et al. in [15] presented a distribution of the synchronization error for these nodes where different messages were sent to the receivers and the time taken to receive these packets at the receivers are different from node to node. So, the author used gaussian distribution for the error to eliminate the effect of these errors among the receivers with zero mean. This algorithm can be extended to serve the communication with multiple hops from the transmitter. But, this extension differs from the multi-hop RBS as it considers that all nodes are connected to only one hop from the transmitter [15]. The advantages of using PCS protocol in WSNs is that it decreases the number of exchange messages and computational load on these nodes. There is a tradeoff between accuracy and resource cost and it supports multi-hop networks.

However, this algorithm also suffers from some shortcomings like [22]; probabilistic guarantee on accuracy that may not be suitable for critical applications and this algorithm is sensitive to the packet loss.

In general, this group suffers from an overhead problem when dividing the network into clusters and selects the cluster-head of each cluster. These protocols usually suffer from link and node failures.

### **2.3.3 Distributed Protocols**

The above two groups are centralized protocols. However, this group consists of fully distributed protocols. There is no reference or leader node within the network and all nodes use the same protocol to be synchronized. These protocols are highly scalable and robust to node/link failure and it is easy to add new nodes in the network. In addition, distributed synchronization protocols depend on the consensus concept where it is an agreement between set of nodes on a certain value using only the local information of each node. Consensus techniques are used in distributed, dynamic topologies. Many of these distributed algorithms achieve the consensus concept in both frequency and phase values [27]. Consensus clock synchronization (CCS) protocols use an external time reference or UTC and an internal consensus within the network at specific time. Each synchronization round in CCS updates the estimation of these parameters for each node. CCS technique consists of two stages they are offset and skew estimation. In the offset stage, nodes use the local clock readings in order to be synchronized. While in the skew stage, nodes depend on the comparison between the current and previous synchronization round to achieve more accurate estimation [7].

Consensus protocol is an iterative process where nodes communicate with each other to achieve the agreement point depending on a certain value without depending on a leader or reference point. Each node shares data locally with different number of iterations until a common value is reached. Some of these distributed protocols such as time diffusion protocol (TDP) [28], reach-back firefly algorithm (RFA) [29], gradient time synchronization protocol(GTSP) [30], external gradient time synchronization protocol (EGSync) [31], average time-sync protocol(ATS) [26], maximum time synchronization protocol [18], weighted maximum time synchronization protocol(WMTS) [16], and time synchronization protocol using the maximum and average values (TSMA) [32]. These protocols described as follows:

Weilian et al. [28] proposed time diffusion protocol (TDP) that pushes all nodes to have time slot with a small difference. Since there is a drift between sensor nodes, this algorithm will be applied periodically. It is divided into two parts; active and inactive parts. In the active part, there are multiple of cycles with  $\tau$  for each cycle. During each cycle, a set of nodes are selected as master nodes by election. Each master node starts the diffusion of timing messages; it builds tree-based scenario in the network. Additionally, the network has non-leaf nodes which are considered as diffused leaders and elected by the election procedure. This will make some propagation on the timing messages. The main objectives of this election are: to remove nodes regarding to the clock variance and to achieve the load distribution for all these nodes. There are some benefits of using this algorithm like; tolerate to packet losses, the equilibrium can be achieved for all nodes during all synchronization times; also since it is not dependent on the static structure, this will provide the network with flexibility, mobility, and many of the master nodes are



distributed in the network with the hierarchal structure. The last advantage is that the synchronization can be done without using an external time. On the other hand, there are many drawbacks for this algorithm including; high complexity, convergence time is high when there is no external time, and clocks can run backward. This can occur when the value of clock is changed to a lower value.

Yi et al. in [29] proposed a clustering firefly synchronization algorithm called reach-back firefly algorithm (RFA) that depends on the initial phases of all nodes. Due to the difference between the initial phases, the number of clusters will be evaluated. Each cluster starts the synchronization process independently and each node receives firing packets from its cluster, until all clusters reach the synchronous state. These synchronous clusters are considered as new integrated nodes when the clusters enter the synchronization phase. This technique deals with nodes that are randomly distributed, all-to-all communication, has homogeneous oscillators and bi-directional links. The simple RFA technique mainly suffers from a worse precision in averaging the packet delays and is not robust. Leidenfrost et al. in [33] proposed another technique that overcome this drawback by using the two techniques together which called Fault-Tolerant Averaging (FTA) and robust RFA. This technique is suitable for network that suffers from delays and provides a high level of synchrony in multi-hop networks.

Sommer in [30] presented the gradient time synchronization protocol (GTSP) which is fully distributed time synchronization. Every node periodically sends a broadcast packet with the time information. This packet will be received by all neighbors and used to estimate their clocks. In this network neither tree nor any reference point is required that makes GTSP robust to link/node failures; GTSP depends only on the local information of

the nodes. Apicharttrisorn et al. in [34] proposed an energy-efficient gradient time synchronization protocol (EGTSP) that is distributed, gradient-based and energy-efficient. This protocol is completely localized, achieves time consensus and gradient using drift estimation and incremental average estimation. In GTSP, every node estimates its clock by using the received time from all neighbors. According to this estimation, the global clock is adjusted. This adjustment can be large, this may cause some errors. In GTSP the broadcasting period is constant and therefore it has small trends that significantly consume sensor networks' energy. Each node in EGTPS estimates the incremental average of time immediately after receiving the broadcasting packet from its neighbors. Whenever the incremental averaging is less, the global time is improved.

Yildirim et al. in [31] presented another time synchronization algorithm called external gradient time synchronization protocol (EGSync) to provide a tight synchronization between nodes when synchronizing one node to its neighbors at the same time. All these nodes agree on the speed and clock values of the reference node by broadcasting the time of the reference node to the neighbors based on the average of this time. However, EGSync has disadvantage; since there is only one reference point for some nodes, if this node fails, EGSync cannot maintain the synchronization process with the neighbors. This protocol works by using the received packets from the reference point before the failure. To solve this, the network needs a redundant node that has an access to the UTC time to complete the synchronization process.

Qun et al. in [35] discussed a distributed time synchronization protocol (DTSC), it is consensus-based algorithm that uses to maintain only the clock offsets and neglecting the clock drifts. On the other hand, Cremaschi et al. in [36] discussed distributed frequency

compensation i.e. clock drift compensation for phase locked loops (PLLs) using consensus techniques. Additionally, Carli in [37] proposed a proportional-integral (PI) consensus-based controller that compensates both clock offset and clock drift. But, still these algorithms consume more energy to reach the synchronous state since the internal components are complex. This will reduce the lifetime of all nodes when they are deployed.

Schenato et al. in [26] proposed another consensus algorithm called average time sync (ATS) algorithm. It is an asynchronous consensus protocol and it is used to average the local time of the nodes to agree on the global synchronization in the network. Correspondingly, it is used to cascade the two consensus methods to estimate the clock parameters where the clock converges to a specific value. This algorithm has three main properties. First it is fully distributed and it is robust to node failure and it is easy to add a new node. Secondly, it maintains the clock skew differences among all nodes. Thirdly, it involves only simple computations like sum/product operations [38]. ATS algorithm is adaptive to slowly time-varying clock drifts and need minimal memory and computational resources. Since ATS is a fully distributed communication topology, there are no specific nodes such as roots and all nodes run with the same algorithm; the nodes broadcast their local time to calculate the skew rates relative to each other. Thereafter, the nodes broadcast their current estimate of the skew rate. Finally, the receiving nodes measure the relative skew estimates depending on the skew rate of other nodes to justify their own virtual clock estimate.

Table 2-1 Capabilities for different protocols

	Distributed	Skew Compensation
TPSN	No	No
LTS	No	No
FTSP	No	Yes
RBS	No	Yes
RFA	Yes	Yes
DTSP	Yes	Yes
ATS	Yes	Yes

Jianping et al. in [18] presented the maximum time synchronization (MTS) protocol that depends on the maximum values and the objective is to maximize the local time to get global synchronization within the network. The benefits of this algorithm compared to other algorithms is that it has higher convergence speed with a finite value, compensate the skew/offset values at the same time, it is fully distributed, asynchronous, robustness to node failure and replacement or adding new nodes is easier. This algorithm pushes the nodes to get the maximum value of time for all nodes and each of these nodes broadcasts a packet with its local hardware clock and relative logical clock skew and offset, without any feedback data from the neighboring nodes.

The same author proposed another algorithm in [16] called weighted maximum time synchronization (WMTS) protocol by taking care of the delay problem in the reception and transmission packets. In this algorithm there are two decision variables; source reference node and the number of hops where the logical clock information will be sent according to these variables from a source node to the receiver node. MTS and WMTS have many advantages over ATS [26] and GTSP [30] such as; GTSP and ATS have asymptotic convergence while MTS converges to the global synchronization with finite time. The convergence time of ATS and GTSP depends on the error value but MTS does

not, and the compensation of skew and offset can be done simultaneously using MTS but in GTSP and ATS, offset will be started after skew has been completed. So, the MTS/WMTS has higher speed convergence compared to the other techniques. Moreover, these two algorithms are asynchronous, distributed, and robust to packet losses and node failure, replacement or relocation is possible or easier. On the other hand, WMTS needs a reference node in its operation.

Qun and Rus in [32] discussed a new time synchronization consensus protocol using maximum and average values called TSMA. The main idea is that this technique is based on the maximum and averaging time values to estimate the offset and skew values. This algorithm is fully distributed like ATS, does the skew compensation, contributes MAC-layer to increase the accuracy, does not need a root node, it is asynchronous, robust to node failure and replacement and high convergence speed compared to ATS. This algorithm uses average consensus to estimate the clock offset. It aims to obtain an internal agreement of the network on the time and how fast it travels. For each synchronization round, this algorithm updates the skew and offset for each node until the clocks converge to a specific value. Mainly, this process is divided into two parts; offset and skew estimation. In the offset estimation part, nodes exchange their local clocks to synchronize nodes to the same time. While in the skew estimation, nodes compare their current and previous values in each round to improve the accuracy of these parameters.

These protocols are robust and flexible to the variations in the network topology and have a steady state value. Additionally, similar to other protocols they are affecting the propagation delays and noise. These protocols are characterized by low complexity iterative process since the neighboring nodes can communicate with each other to achieve

the agreement point depending only on the initial evaluations without going to transmit data to a reference point [39]. Different applications achieve the consensus concept such as load balancing in parallel computing [40], coordination of autonomous agents [41], distributed control [42], data fusion problems [43], and flocking in dynamical systems [44].

## **2.4 Comparison between Time Synchronization Protocols**

Regarding these algorithms, nodes can be synchronized with other nodes in the same network by the following ways: (1) Synchronizing nodes with an external time source, (2) synchronizing nodes with a root node in the same network, and (3) synchronizing all nodes to a specific value. As mentioned previously, synchronization protocols are divided into two groups they are centralized and distributed. The centralized group is further divided into two structures they are tree and cluster. The tree structure protocols such as (TPSN [12], FTSP [13], LTS [19], DMTS [17], Tiny [21], FBS [20]) suffers from different challenges as follows:

1. Overhead in the network, in tree structure building stage.
2. Node/link failures in this structure (since there is a root node in the tree).
3. Power consumption is high, and hence the lifetime of all nodes is reduced.
4. Not suitable for the topology changes in the network, and not suitable for the multi-hop communication as well.
5. Some of the tree protocols (DMTS) uses low frequencies to be deployed and this is not suitable for critical applications.

6. Accuracy is low when for long tree structures.

Regarding the cluster structure such as (RBS [15], PCS [14], TSync [24]), similar to tree structure there are some drawbacks of the group as follows:

1. Overhead caused by clustering the network and nominating the cluster head (needs more time to build the structure).
2. Convergence time is high.
3. Node/link failures.
4. Power consumption is high.
5. Sensitive for packet loss.
6. Not suitable for topology changes.

For the last group, there are many distributed synchronization protocols such as (TDP [28], RFA [29], GTSP [30], ESync [31], ATS [26], MTS [18], WMTS [16], TSMA [32]).

For some of these protocols such as (TDP [28], RFA [29], GTSP [30], ATS [26], ESync [31]) there are several drawbacks including:

1. Compensate the drift and offset individually, and need multiple operations to do the drift and offset compensations.
2. Keep tracking the neighboring information which will cause an overhead on the memory and processing unit.
3. There are asynchronous protocols.
4. ESync protocol needs a reference node to start its operation.

On the other hand, the rest of the distributed protocols such as (MTS [18], WMTS [16], TSMA [32]), are more effective and easy to implement when compared to the previous

protocols. Additionally, they are compensating drift and offset at same time, however, still they are keeping track the neighboring nodes. Generally, the main task for the researchers is to synchronize the nodes with less value of error, time, overhead, and consumption energy to be more effective.

Using the CCS algorithms instead of the centralized algorithms can minimize the faults caused by clocks between the sensor nodes that are located geographically close to each other to achieve an accurate synchronization. Consensus-based synchronization algorithm is used to maintain the time offsets and clock frequency skews dynamically. The advantages of using this concept are computationally light, scalable, robust to node and link failure, and it does not need a leader node [45]. Furthermore, the consensus-based approach is not fixed and dynamically chooses the leader node when it is needed.

The convergence speed in the iterative process depends on the number of iterations that are required to achieve the steady state point. The protocol that needs small number of iterations to achieve the steady state point is considered as the fastest convergence protocol. Luckily enough, reducing the number of iterations to achieve the convergence point in the network will decrease the consumption energy for each node within the network. Some of the consensus protocols were implemented with static topologies where nodes and communication links are usually fixed all the time [46].

## **2.5 Stopping Criterion**

In general, it is used to detect the iterative process when there is no sense in proceeding with more iteration. This acts like a controller within the system that decide wither to stop or continue the iterative process (it uses to achieve good performance for the WSN).



There are many stopping criteria that are mentioned in the previous researches and used different stopping conditions such as maximum time, maximum number iterations, reach a specific bound, mean value, standard deviation, variance, relative function, absolute function, ... etc.

### **2.5.1 Literature on Different Stopping Criteria**

Different stopping criteria have been discussed in the previous researches and these criteria can be classified into two categories direct and derived stopping criteria. Both stopping criteria depend on the condition that uses to stop the iterative process.

#### **I. Direct Stopping Criterion:**

This type of SC depends directly on the iterative process, simple and does not need any calculations such as maximum time, maximum number of iterations and reach a required bound [47, 48] as follows:

##### **a. Maximum Time and Maximum Number of Iterations:**

The iterative process can be stopped either using the maximum time value or using the maximum number of iterations and this can be represented by the following equation:

$$SC \rightarrow t(k + 1) \geq t_{max} \quad 2.4$$

Where;  $k$  is the iteration value and it changes from  $(0, 1, \dots, N)$ ,  $t(k + 1)$  is the time of the current iteration and  $t_{max}$  is the stopping condition that uses to stop the iterative process, all these variables are linearly dependent to each other.

**b. Reach the required Bound:**

This criterion differs from the previous one in the stopping condition and does not require any complex operations; it depends only on the time threshold that is given to stop the iterative process called  $t_{limit}$  as in the following equation:

$$SC \rightarrow t(k + 1) \geq t_{limit} \quad 2.5$$

**II. Derived Stopping Criteria:**

This type of SC uses the proceed output of the iterative process to evaluate the measured variable that will be used to stop the iterative process and it needs more calculations such as mean, standard deviation, relative and absolute functions[49, 50]:

**a. Mean:**

This criterion represents the absolute difference between the objective time  $t(k + 1)$  for the current iteration  $(k + 1)$  and the average values of all time values up to the current iteration  $t(0), \dots, t(k + 1)$  and this measured value uses to stop the iterative process regarding to specific threshold called  $\varepsilon$  that depends on the accuracy of the application as in the following equation:

$$SC \rightarrow \left| t(k + 1) - \left( \frac{\sum_0^{k+1} t(k+1)}{N} \right) \right| \leq \varepsilon \quad 2.6$$

**b. Standard Deviation:**

This criterion uses the standard deviation concept to stop the iterative process and it needs more operations than the mean SC. Where the measured value  $\sigma^t$  is equal to the standard deviation of all times  $t$  until the current iteration  $(k + 1)$  and it uses to stop the

iterative process regarding to specific threshold called  $\varepsilon$  that depends on the accuracy of the application as in the following equation:

$$SC \rightarrow \sigma^t = \sqrt{\frac{1}{N} \left( \sum_{k=1}^N (t(k+1) - \frac{1}{N} (\sum_{k=1}^N Nt(k+1))) \right)^2} \leq \varepsilon \quad 2.7$$

Where:  $N$  represents number of iterations.

**c. Relative function criterion:**

The termination condition for the relative function depends on a small relative difference between the time value of the current iteration  $t(k+1)$  and the time value of the previous iteration  $t(k)$  dividing by the maximum of all time values  $t$  and this value uses to stop the iterative process regarding to specific threshold called  $\varepsilon$  that depends on the accuracy of the application as in the following equation:

$$SC \rightarrow \left( \frac{|t(k+1) - t(k)|}{\max(t)} \right) \leq \varepsilon \quad 2.8$$

**d. Absolute function criterion:**

The termination condition for the absolute function depends on the difference between the time value of the current iteration  $t(k+1)$  and the time value of the previous iteration  $t(k)$  and this value uses to stop the iterative process regarding to specific threshold called  $\varepsilon$  that depends on the accuracy of the application as in the following equation:

$$SC \rightarrow (|t(k+1) - t(k)|) \leq \varepsilon \quad 2.9$$

Next section describes different SCs that have been used to stop the iterative process in some of the previous time synchronization protocols, where the most SC that used is the absolute SC.

### 2.5.2 Stopping Threshold

Several techniques were proposed in this field to deploy different algorithms with the absolute stopping criterion using testbeds of sensor nodes as mentioned in [51], [52] and [17] (this part shows the accuracy of some protocols that discussed before in the literature). In [51], Djenouri et al. implemented a testbed of sensor nodes to deploy the fast distributed time synchronization algorithm. In this research, authors used small number of Micaz nodes to implement this algorithm experimentally using the external oscillator frequency 32KHz and the error value has been estimated using these nodes between 1 $\mu$ s and 7 $\mu$ s, where the most values located between 3 $\mu$ s and 5 $\mu$ s; and with average is 3.50 $\mu$ s.

In the second research [52], Huang proposed a new time synchronization algorithm called 2LTSP (Long Term and Large Scale Time Synchronization Protocol) which was implemented using Arduino WSN platform. The error value of this protocol when the synchronization period is less than 100s is around 0.6ms. In this research, authors compared this protocol with other previous protocols as shown in Table 2-2:

Table 2-2 Average absolute errors in millisecond

<b>T</b>	<b>100s</b>	<b>300s</b>	<b>500s</b>
<b>2LTSP</b>	[0.59, 0.62]	[1.13, 1.17]	[1.52, 1.56]
<b>FTSP</b>	[10.12, 11.27]	[18.19, 20.95]	[23.23, 26.19]
<b>PulseSync</b>	[8.93, 10.24]	[15.74, 18.80]	[21.22, 24.49]

In the last research [17], authors implemented two time synchronization protocols using Micaz nodes with different topologies the first one called average time synchronization (ATS) and the second one called maximum time synchronization. The estimated errors for the two algorithms with different topologies as in Table 2-3:

Table 2-3 Convergence and accuracy results (1 tick equal to  $=1/32K=30.5\mu s$ )

Protocol	Topologies	Grid	Ring	Linear
MTS	Cycles	5	5	8
	Accuracy/ticks	3.7	4.1	7.4
ATS	Cycles	16	42	122
	Accuracy/ticks	5.5	9.5	18.6

From the above error values and the typical sensor nodes (Micaz and IRIS) have drift rate of  $\pm (30-100)$  microseconds. The termination threshold depends on the timer that uses in the sensor node either an internal timer or external with high frequency. The convergence time of any synchronization algorithm in WSNs can be found using this threshold. In general, Micaz have multiple of timers and each of these timers has different specifications that may use in the implementation.

## 2.6 Thesis Contributions

As can be noticed from the literature survey and the clock model, most of the WSNs are deployed under harsh environments such as vast fluctuation in temperatures, rain, vibration, humidity, chemicals, electrical shock, pressure, physical damage, etc. This will change the normal operation of nodes to serve their tasks with an accurate time, minimum latency, and high performance without any packet loss. Consequently, accurate timing is an important factor and essential for many applications such as assigning a

global timestamp to sensed data/events, cooperation of multiple sensor nodes, precise event localization (e.g., shooter detection), and coordination of wake-up and sleeping times (energy efficiency). Under unexpected conditions the hardware clocks of these nodes may drift and increase the required time for the network to be synchronized with the global clock with some skew and offset errors.

All previous averaging researches depend on the averaging time values of the neighboring nodes with a reference node and update the time of each node regarding to this average value. However, in our proposed algorithm each node communicates with the neighboring nodes and averages the value of neighbors with respect to server time. After that, each node updates their values with respect to the updated value in this network. Consequently, all nodes will update their values at each iteration until reach the server time. At this point these nodes will stop updating their values to minimize the consumption in both memory and energy.

Motivated by what is mentioned before, we propose this consensus distributed time synchronization protocol for WSN at which the consensus clock synchronization (CCS) is used to minimize the clock differences between nodes that are located geographically close to each other. These nodes keep update their values until reach the global time for all nodes. In the meanwhile, clocks will reset and stop the communication process. This protocol is mostly deployed in the harsh environments with some properties such as computationally light, scalable, applicable for the topology changes, fully distributed, robust to node and link failure, it does not need a leader node, has global stability regardless to the network connectivity, controllable time accuracy, single hop communication among nodes, simplicity with little communication overhead, and

Hardware-friendly. This protocol can be deployed to work in different systems such as monitoring pollution, tracking objects, oil industry, precise event localization (e.g., shooter detection), and coordination of wake-up and sleeping times (energy efficiency).

In the next chapters introduce the following:

1. The protocol with simple mathematical analysis that has sum and product operations.
2. Extensive computer simulation using MATLAB with different topologies and sizes.
3. Experimental validation using Micaz nodes for the grid topology with different sizes.
4. A stopping criterion for the steady state region and find the convergence time in both simulation and experimental scenarios.
5. A stopping criterion for the transient region and find the convergence time in both simulation and experimental scenarios.

This model has some differences and modifications that improve the system performance by reducing the consumption energy of those sensor nodes, eliminate any overhead on the nodes, increase the reliability of the network under any variations, and reduce number of the communication cycles to reach the convergence state with the minimum error. All these specifications can be achieved with simple operations that are relative to the averaging concept (sum/product operations) as described in Chapter 3.

# CHAPTER THREE

## Averaging Protocol

This chapter discusses the proposed protocol for time synchronization in WSN. First, it introduces the Averaging Protocol (AP) for the synchronous and asynchronous cases. Then, it shows the mathematical representation of the synchronous averaging protocol. Subsequently, it introduces different simulation and practical topologies such as Grid, Hexa, and Random network with different sizes. The chapter also discusses the error values behavior for these topologies. Finally, it compares among the results for different sizes of networks.

### 3.1 Averaging Protocol Concept

The AP concept relies on exchanging time information among neighbor nodes until all nodes (consensus) reach the same time stamps with acceptable small errors. The AP treats the local time as a dynamical variable that is updated by the AP algorithm. The AP algorithm, through local interaction with one hop neighbors, drives the local time of each node to that of the master node until all nodes converge to almost the same time. Once convergence is detected, the local clocks are reset to the ones obtained from the consensus algorithm as shown Figure 3-1.



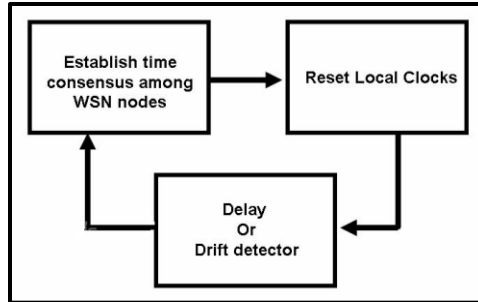


Figure 3-1 Continuous re-setting of node clock by consensus protocol

The nodes have information about their nearest neighbors or the nodes that are linked together and the protocol assumes that it doesn't have global information about the network, i.e. these nodes update their estimates with respect to their neighbors only. This algorithm is known as the average consensus algorithm since all nodes converge to the same average value of all local states for the nodes with respect to the server time.

Generally, there are two average consensus algorithms; synchronous algorithm and asynchronous algorithm. At each iteration, these nodes transmit their estimated time values to the neighboring nodes. The nodes then update their values to the new values by averaging the received estimates. In the synchronous version, all nodes, depending on their connectivity, exchange their time estimated values with the neighboring nodes at the same iteration. Subsequently, all nodes in the network update their time estimations using the AP. In this version, all nodes must update their information at the same time. On the other hand, in asynchronous version, not all nodes participate in updating its information at each iteration but all of them do so in a number of iterations. Generally, synchronous algorithms are easy to explain and analyze than asynchronous algorithms. Therefore and without loss of generality, the focus will be on the synchronous version. This protocol is divided into multiple stages as indicated in Figure 3-1. The first stage represents the averaging concept of the local time. Its flow chart is shown in Figure 3-2. Stage two

represents the detection scenario for the iterative process and indicates when the process should be stopped depending on a certain stopping criterion which will be discussed in Chapter 4. In the next stage, the time values of all components in the network are reset to keep them synchronized and updated at any time.

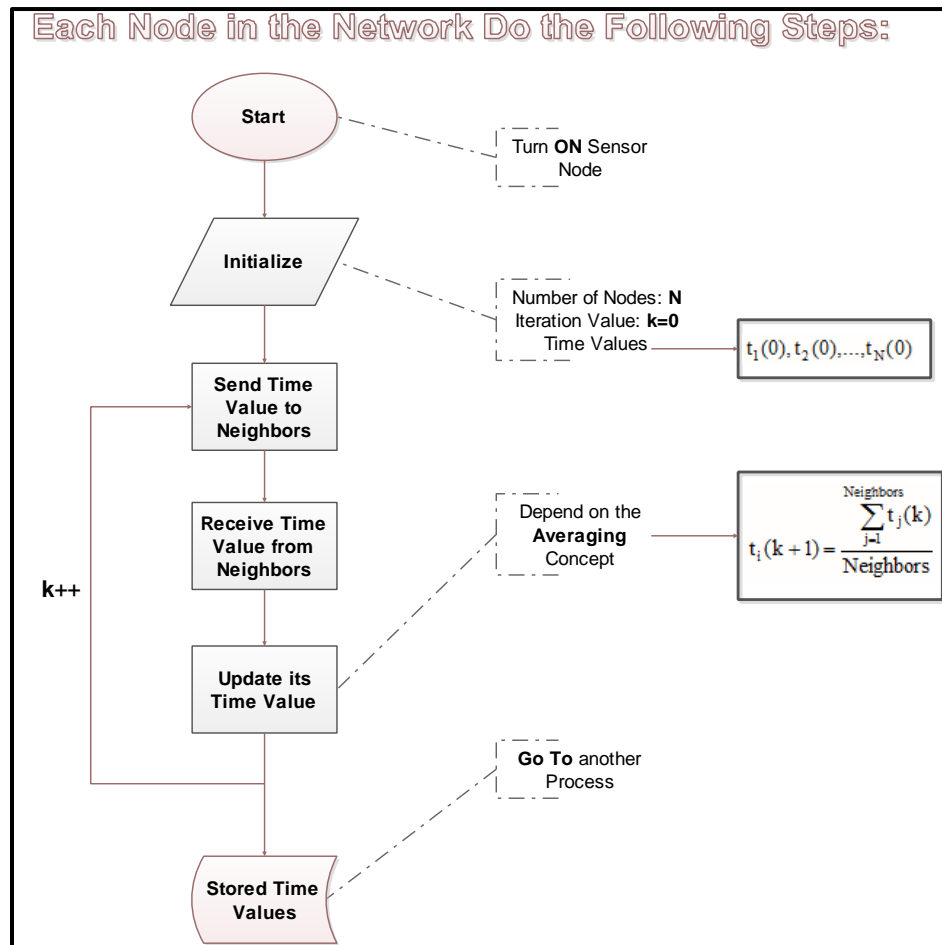


Figure 3-2 Flow chart of the Averaging Protocol

### 3.1.1 Description of Asynchronous Average Consensus Algorithm

In asynchronous average consensus algorithm, there is only one node that wakes up at each iteration. This node starts sending messages to other nodes. In this algorithm, at every iteration of time, there is a subset of nodes that updates its estimation regarding to

the average of this subset and so on until all nodes update their values and this takes more than one iteration.

### 3.1.2 Description of Synchronous Average Consensus Algorithm

Consider a network of  $N$  nodes as in Figure 3-3 where the nodes are numbered from 1 to  $N-1$  and the master node  $n_m$  is the last one. A node  $n_i$  has a local time value  $t_i$  where  $i = 1, 2, \dots, N-1$ , and the master node has the time  $t_m$ .

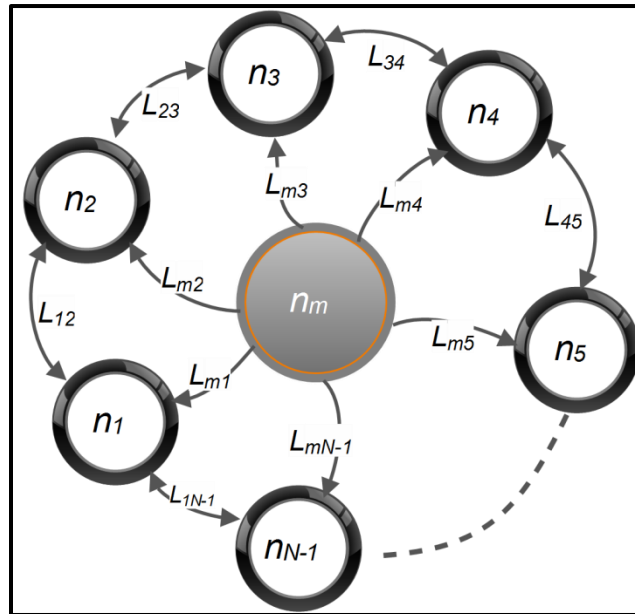


Figure 3-3 Network with  $N$  sensor nodes and  $L$  links

Depending on the connectivity, each node calculates its new time value by averaging the values it receives from the neighboring nodes. If the nodes in the network are fully connected, then the average time,  $t_{avg}$ , value of a certain node  $t_i$  is calculated as follows:

$$t_{avg} = \frac{\sum_{j=1}^{N-2} \text{such that } j \neq i} t_j + t_m}{N-1} \quad 3.1$$

Generally, if a node  $n_i$  is connected to  $n$  nodes where  $n \leq N-1$ , then at iteration  $k$ , the new average time at that node is denoted by  $t_i(k+1)$  and is given by:

$$t_i(k + 1) = \frac{\sum_{j=1}^{j=n} t_j(k)}{n} \quad 3.2$$

If the master node is connected to this node then it can be factored out from (3.4) and yields the following

$$t_i(k + 1) = \frac{\sum_{j=1}^{j=n-1} t_j(k)}{n} + \frac{t_m(k)}{n} \quad 3.3$$

The above equation can be used for all nodes to yields the new time vector  $\mathbf{t}(\mathbf{k} + \mathbf{1})$  from the current time vector and it can be put a matrix form as follows:

$$\mathbf{t}(\mathbf{k} + \mathbf{1}) = \mathbf{A}\mathbf{t}(\mathbf{k}) + \mathbf{B}t_m(\mathbf{k}) \quad 3.4$$

Where  $\mathbf{B}$  represents the connectivity between the nodes and the master node, and  $\mathbf{A}$  represents the connectivity matrix between the nodes and their neighbors excluding the master node; where the size of this matrix  $\mathbf{A}$  is equal to  $(N - 1) \times (N - 1)$  and matrix  $\mathbf{B}$  is  $(N - 1) \times 1$ . And the following equations described the two matrices:

$$\mathbf{A}_{xy} = \begin{cases} \frac{1}{n}, & \text{if } L_{xy} \text{ exists; } L \text{ is the link between node } x \text{ and nodes } y \\ 0, & \text{if } L_{xy} \text{ not exist or at } x = y \end{cases} \quad 3.5$$

$$\mathbf{B}_{xy} = \begin{cases} \frac{1}{n}, & \text{if } L_{my} \text{ exists; } L \text{ is the link between master node and nodes } y \\ 0, & \text{if } L_{my} \text{ not exist or at } x = y \end{cases} \quad 3.6$$

And the server time can be represented by this equation that depends on the incremented time  $\Delta t$  multiply by the iteration value  $k$ :

$$t_m(k) = k \times \Delta t \quad 3.7$$

## 3.2 Examples on this Protocol

To illustrate how the protocol updates the time values for the nodes, various examples are introduced for different network topology. This part includes multiple examples on WSN scenarios, where the sensor nodes are distributed in different forms such as Grid, Hexa, and Random with different sizes. The first section introduces the mathematical representation of this protocol for the small networks whereas the second section discusses the higher level of networks.

### 3.2.1 Examples on Small Networks

#### I. 4-Nodes Grid Topology:

The distribution of nodes as indicated in Figure 3-4 shows that this network consists of one master node and three normal nodes:

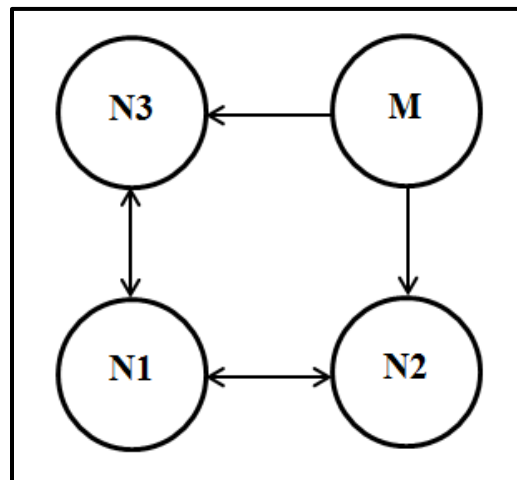


Figure 3-4 4-nodes with Grid Topology

We can write the updated time equation for each node depending on the connectivity links between the node and its neighbors which are directly connected to it. The time equation for Node 1,  $n_1$ , in the iteration  $k$  is as follows, using the averaging equation:

$$t_1(k + 1) = \frac{t_2(k) + t_3(k)}{2} \quad 3.8$$

where  $n_1$  as seen in Figure 3-4 is connected to nodes  $n_2$  and  $n_3$ . The time equations for Node 2 and Node 3 in the iteration  $k$  are;

$$t_2(k + 1) = \frac{t_1(k) + t_m(k)}{2} \quad 3.9$$

$$t_3(k + 1) = \frac{t_1(k) + t_m(k)}{2} \quad 3.10$$

where  $n_2$  is connected to nodes  $n_1$  and  $n_m$  while  $n_3$  is connected to nodes  $n_2$  and  $n_m$ . The time equation of these nodes has two parts; first one depends on the connectivity between the node and his neighbors, and the second one depends on the connectivity between the node and the master node if there is a link between them.

From these values for each node, we can write the equation for all as in linear equation.

$$[\mathbf{t}(\mathbf{k} + \mathbf{1})]_{3 \times 1} = [\mathbf{A}]_{3 \times 3} [\mathbf{t}(\mathbf{k})]_{3 \times 1} + [\mathbf{B}]_{3 \times 1} \mathbf{t}_m(\mathbf{k}) \quad 3.11$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} t_m(k) \quad 3.12$$

## II. 4-Nodes Hexa Topology:

The distribution of nodes as indicated in Figure 3-5 reveals that this network consists of one master node and three normal nodes:

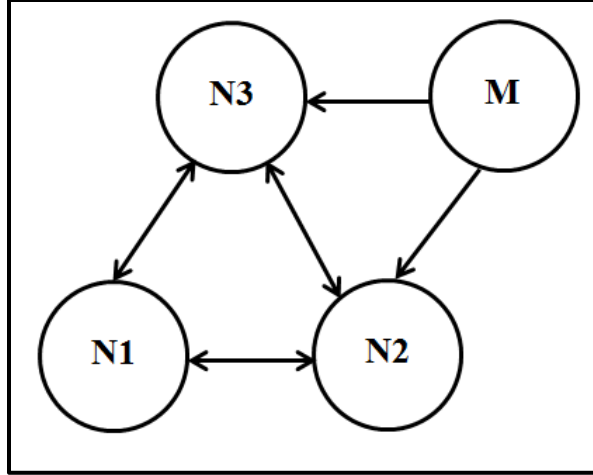


Figure 3-5 4-nodes with Hexa Topology

We can write the updated time equation for each node depending on the connectivity links between the node and its neighbors which are directly connected to it. The time equation for Node 1,  $n_1$ , in the iteration  $k$  is as follows, using the averaging equation:

$$t_1(k + 1) = \frac{t_2(k) + t_3(k)}{2} \quad 3.13$$

where  $n_1$  as seen in Figure 3-5 is connected to nodes  $n_2$  and  $n_3$ . The time equations for Node 2 and Node 3 in the iteration  $k$  are;

$$t_2(k + 1) = \frac{t_1(k) + t_3(k) + t_m(k)}{3} \quad 3.14$$

$$t_3(k + 1) = \frac{t_1(k) + t_2(k) + t_m(k)}{3} \quad 3.15$$

where  $n_2$  is connected to nodes  $n_1$ ,  $n_3$  and  $n_m$  while  $n_3$  is connected to nodes  $n_1$ ,  $n_2$  and  $n_m$ . The time equation of these nodes depends on the connectivity between the node and his neighbors, master node if there is a link between them.

From these values for each node, we can write the equation for all as in linear equation.

$$[\mathbf{t}(k+1)]_{3 \times 1} = [\mathbf{A}]_{3 \times 3}[\mathbf{t}(k)]_{3 \times 1} + [\mathbf{B}]_{3 \times 1}t_m(k) \quad 3.16$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} t_m(k) \quad 3.17$$

### III. 4-Nodes Random Topology:

The distribution of nodes as indicated in Figure 3-6 reveals that this network consists of one master node and three normal nodes, all these nodes are randomly distributed in the first iteration and have same distribution for all iterations:

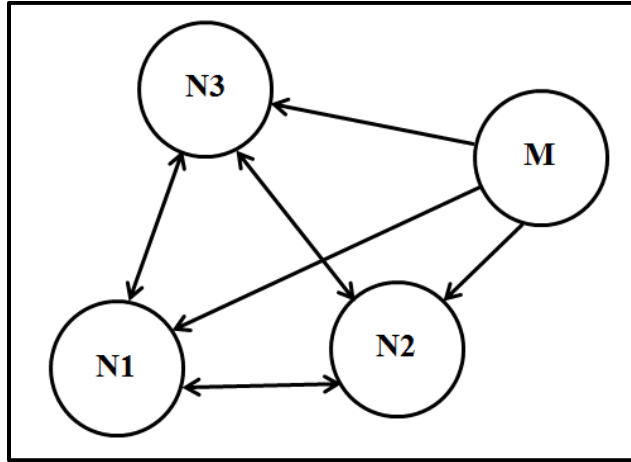


Figure 3-6 4-nodes with Random Topology

We can write the updated time equation for each node depending on the connectivity links between the node and its neighbors which are directly connected to it. The time equation for Node 1,  $n_1$ , in the iteration  $k$  is as follows, using the averaging equation:

$$t_1(k+1) = \frac{t_2(k) + t_3(k) + t_m(k)}{3} \quad 3.18$$



where  $n_1$  as seen in Figure 3-6 is connected to nodes  $n_2$ ,  $n_3$  and  $n_m$ . The time equations for Node 2 and Node 3 in the iteration  $k$  are;

$$t_2(k + 1) = \frac{t_1(k) + t_3(k) + t_m(k)}{3} \quad 3.19$$

$$t_3(k + 1) = \frac{t_1(k) + t_2(k) + t_m(k)}{3} \quad 3.20$$

where  $n_2$  is connected to nodes  $n_1$ ,  $n_3$  and  $n_m$  while  $n_3$  is connected to nodes  $n_1$ ,  $n_2$  and  $n_m$ . The time equation of these nodes depends on the connectivity between the node and his neighbors, master node if there is a link between them.

From these values for each node, we can write the equation for all as in linear equation.

$$[\mathbf{t}(k + 1)]_{3 \times 1} = [\mathbf{A}]_{3 \times 3} [\mathbf{t}(k)]_{3 \times 1} + [\mathbf{B}]_{3 \times 1} t_m(k) \quad 3.21$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \end{pmatrix} + \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} t_m(k) \quad 3.22$$

### 3.2.2 Examples on Large Networks

This part describes two sizes of networks, 9 nodes and 16 nodes with different topologies like Grid, Hexa, and Random. It shows the distribution of these networks and the mathematical representation for the time equation of each node.

#### I. 9-Nodes Grid Topology:

The distribution of these nodes is as illustrated in Figure 3-7. This network consists of one master node and eight normal nodes:

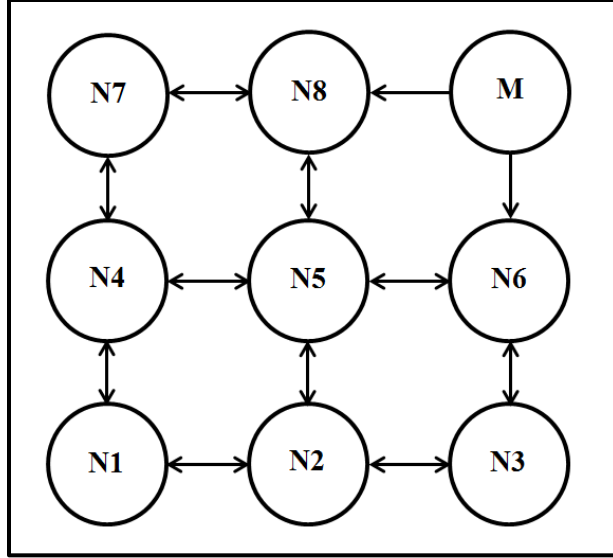


Figure 3-7 9-nodes with Grid Topology

The updated equation of the time is summarized as in the following linear equation that is depending on the connectivity matrix and the averaging equation;

$$[\mathbf{t}(k+1)]_{8 \times 1} = [\mathbf{A}]_{8 \times 8}[\mathbf{t}(k)]_{8 \times 1} + [\mathbf{B}]_{8 \times 1}t_m(k) \quad 3.23$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \\ t_4(k+1) \\ t_5(k+1) \\ t_6(k+1) \\ t_7(k+1) \\ t_8(k+1) \end{pmatrix} = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \\ t_4(k) \\ t_5(k) \\ t_6(k) \\ t_7(k) \\ t_8(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/3 \\ 0 \\ 1/3 \end{pmatrix} t_m(k) \quad 3.24$$

## II. 9-Nodes Hexa Topology:

The distribution of these nodes is as in Figure 3-8. This network consists of one master node and eight normal nodes:

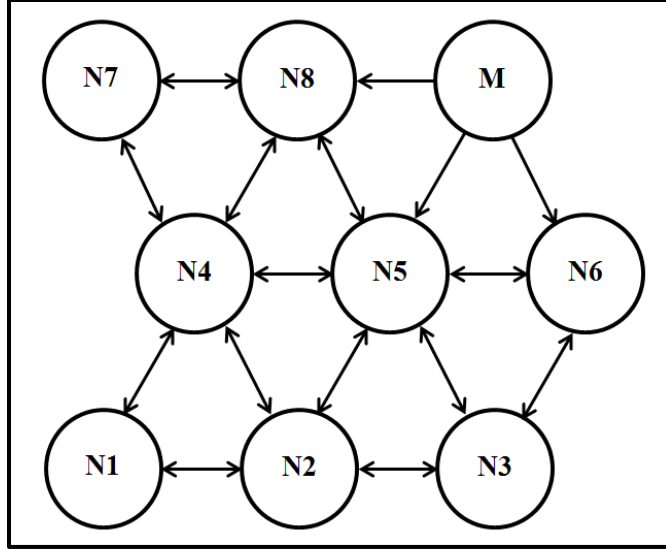


Figure 3-8 9-nodes with Hexa Topology

Likewise, updated equation of the time is summarized as in the following linear equation that is depending on the connectivity matrix and the averaging equation;

$$[\mathbf{t}(\mathbf{k} + \mathbf{1})]_{8 \times 1} = [\mathbf{A}]_{8 \times 8}[\mathbf{t}(\mathbf{k})]_{8 \times 1} + [\mathbf{B}]_{8 \times 1}t_m(k) \quad 3.25$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \\ t_4(k+1) \\ t_5(k+1) \\ t_6(k+1) \\ t_7(k+1) \\ t_8(k+1) \end{pmatrix} = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 & 0 & 0 \\ 1/5 & 1/5 & 0 & 0 & 1/5 & 0 & 1/5 & 1/5 \\ 0 & 1/6 & 1/6 & 1/6 & 0 & 1/6 & 0 & 1/6 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \\ t_4(k) \\ t_5(k) \\ t_6(k) \\ t_7(k) \\ t_8(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/6 \\ 1/3 \\ 0 \\ 1/4 \end{pmatrix} t_m(k) \quad 3.26$$

### III. 9-Nodes Random Topology:

Figure 3-9 shows the distribution of the nodes in this network. This network consists of one master node and eight normal nodes:

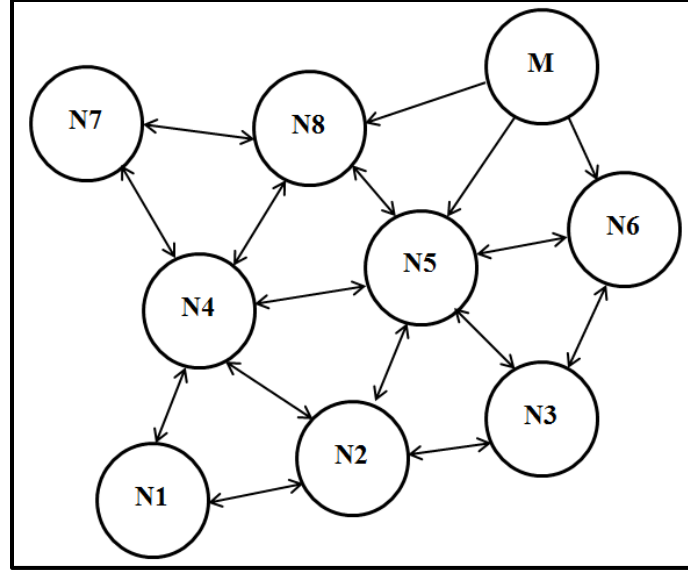


Figure 3-9 9-nodes with Random Topology

Likewise, updated equation of the time is summarized as in the following linear equation that is depending on the connectivity matrix and the averaging equation;

$$[\mathbf{t}(k + 1)]_{8 \times 1} = [\mathbf{A}]_{8 \times 8}[\mathbf{t}(k)]_{8 \times 1} + [\mathbf{B}]_{8 \times 1}t_m(k) \quad 3.27$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \\ t_4(k+1) \\ t_5(k+1) \\ t_6(k+1) \\ t_7(k+1) \\ t_8(k+1) \end{pmatrix} = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 & 0 & 0 \\ 1/5 & 1/5 & 0 & 0 & 1/5 & 0 & 1/5 & 1/5 \\ 0 & 1/6 & 1/6 & 1/6 & 0 & 1/6 & 0 & 1/6 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \\ t_4(k) \\ t_5(k) \\ t_6(k) \\ t_7(k) \\ t_8(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/6 \\ 1/3 \\ 0 \\ 1/4 \end{pmatrix} t_m(k) \quad 3.28$$

#### IV. 16-Nodes Grid Topology:

The distribution of these nodes is indicated in Figure 3-10. This network consists of one master node and fifteen normal nodes:

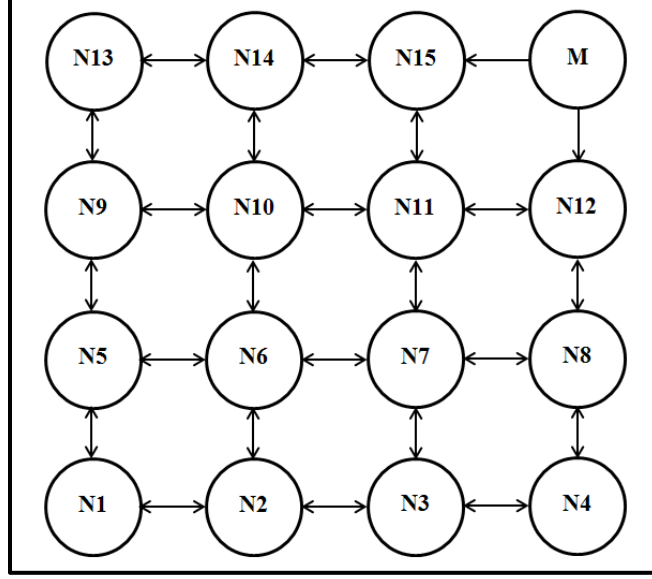


Figure 3-10 16-nodes with Grid Topology

Depending on the connectivity matrix and the averaging equation, the updated time equation for each node is summarized as follow;

$$[\mathbf{t}(k+1)]_{15 \times 1} = [\mathbf{A}]_{15 \times 15} [\mathbf{t}(k)]_{8 \times 1} + [\mathbf{B}]_{15 \times 1} t_m(k) \quad 3.29$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \\ t_4(k+1) \\ t_5(k+1) \\ t_6(k+1) \\ t_7(k+1) \\ t_8(k+1) \\ t_9(k+1) \\ t_{10}(k+1) \\ t_{11}(k+1) \\ t_{12}(k+1) \\ t_{13}(k+1) \\ t_{14}(k+1) \\ t_{15}(k+1) \end{pmatrix} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 & 1/4 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 1/4 & 0 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \\ t_4(k) \\ t_5(k) \\ t_6(k) \\ t_7(k) \\ t_8(k) \\ t_9(k) \\ t_{10}(k) \\ t_{11}(k) \\ t_{12}(k) \\ t_{13}(k) \\ t_{14}(k) \\ t_{15}(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/3 \\ 0 \\ 0 \\ 1/3 \end{pmatrix} t_m(k) \quad 3.30$$

## V. 16-Nodes Hexa Topology:

Figure 3-12 indicated the distribution of these nodes in which the network consists of one master node and fifteen normal nodes:

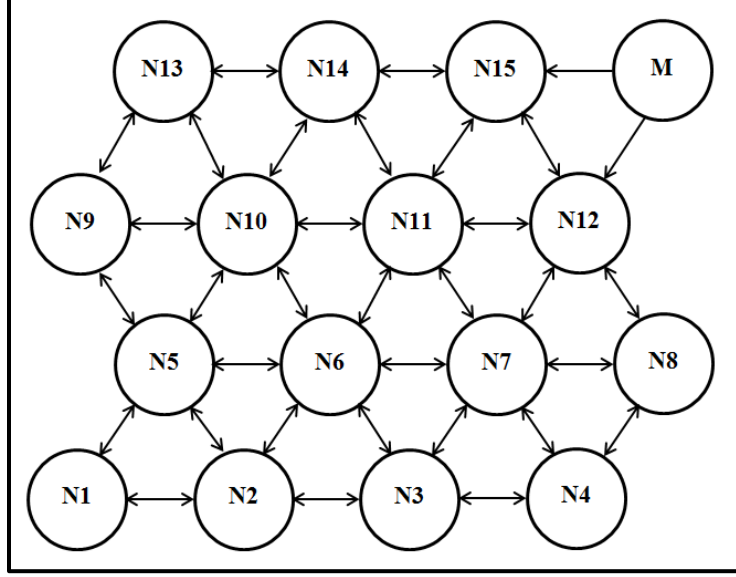


Figure 3-11 16-nodes with Hexa Topology

Depending on the connectivity matrix and the averaging equation, the updated time equation for each node is summarized as follow;

$$[\mathbf{t}(k+1)]_{15 \times 1} = [\mathbf{A}]_{15 \times 15} [\mathbf{t}(k)]_{15 \times 1} + [\mathbf{B}]_{15 \times 1} t_m(k) \quad 3.31$$

$$\begin{pmatrix} t_1(k+1) \\ t_2(k+1) \\ t_3(k+1) \\ t_4(k+1) \\ t_5(k+1) \\ t_6(k+1) \\ t_7(k+1) \\ t_8(k+1) \\ t_9(k+1) \\ t_{10}(k+1) \\ t_{11}(k+1) \\ t_{12}(k+1) \\ t_{13}(k+1) \\ t_{14}(k+1) \\ t_{15}(k+1) \end{pmatrix} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 0 & 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/5 & 1/5 & 0 & 0 & 0 & 1/5 & 0 & 0 & 1/5 & 1/5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 1/6 & 0 & 1/6 & 0 & 1/6 & 0 & 0 & 1/6 & 1/6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/6 & 1/6 & 0 & 1/6 & 0 & 1/6 & 0 & 0 & 1/6 & 1/6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/6 & 1/6 & 0 & 0 & 1/6 & 0 & 1/6 & 0 & 1/6 & 1/6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/6 & 1/6 & 0 & 0 & 1/6 & 0 & 1/6 & 0 & 1/6 & 1/6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/5 & 1/5 & 0 & 0 & 1/5 & 0 & 0 & 0 & 1/5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \end{pmatrix} \begin{pmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \\ t_4(k) \\ t_5(k) \\ t_6(k) \\ t_7(k) \\ t_8(k) \\ t_9(k) \\ t_{10}(k) \\ t_{11}(k) \\ t_{12}(k) \\ t_{13}(k) \\ t_{14}(k) \\ t_{15}(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/5 \\ 0 \\ 0 \\ 1/4 \end{pmatrix} t_m(k) \quad 3.32$$

## VI. 16-Nodes Random Topology:

This network consists of one master node and fifteen normal nodes and the distribution of these nodes is as shown in Figure 3-12.

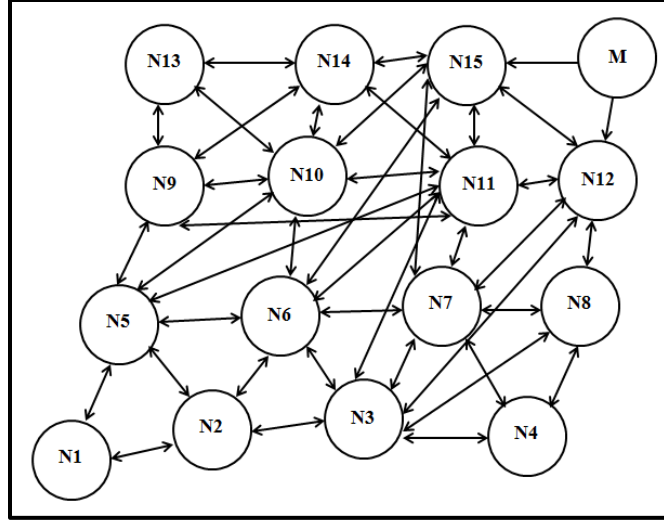


Figure 3-12 16-nodes with Random Topology

Depending on the connectivity matrix and the averaging equation, the updated time equation for each node is summarized as follow;

$$[\mathbf{t}(\mathbf{k} + \mathbf{1})]_{15 \times 1} = [\mathbf{A}]_{15 \times 15} [\mathbf{t}(\mathbf{k})]_{8 \times 1} + [\mathbf{B}]_{15 \times 1} t_m(k) \quad 3.33$$

In the next section, these networks will be simulated using MATLAB to study the behavior of the time and error curves for each node in these networks.

### 3.3 Simulation Results

The AP has been tested by simulating the previously introduced networks topologies of different sizes with the AP using MATLAB. The general protocol used in the simulation is described as in the pseudo code shown in Figure 3-13 and Flow Chart in Figure 3-2.

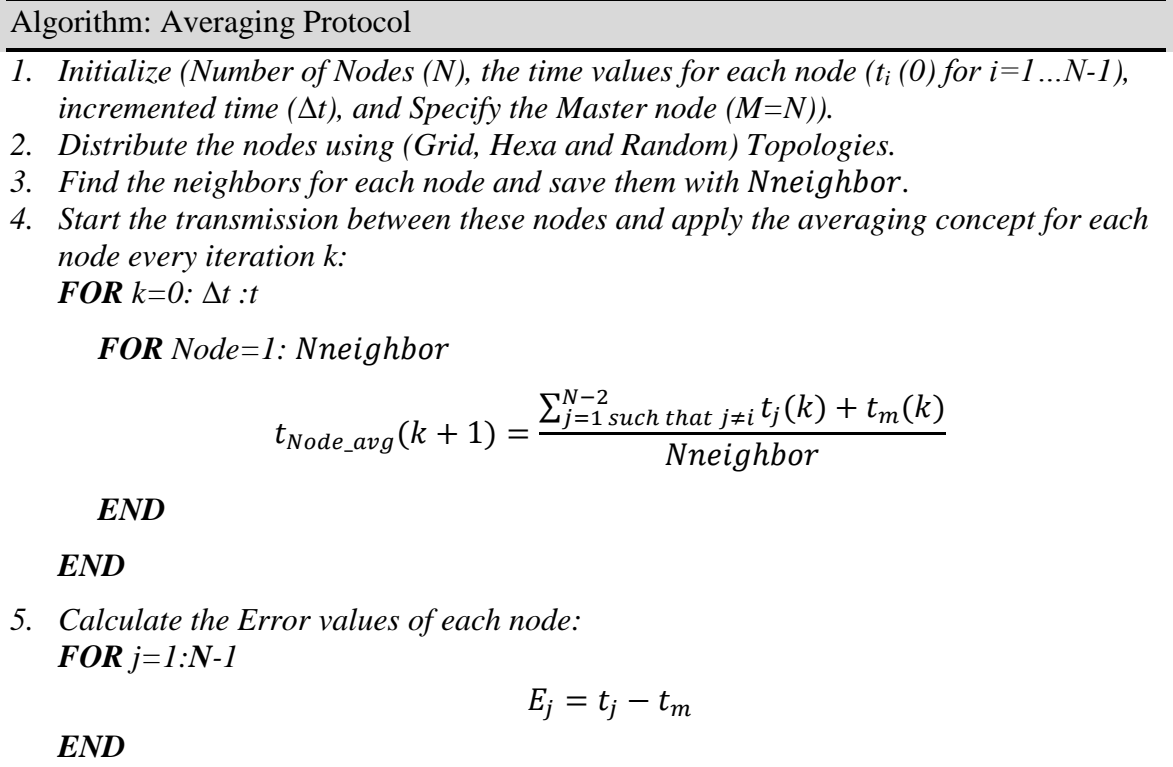


Figure 3-13 Pseudo code of the Averaging Protocol

There are some specifications and parameters that are used in MATLAB as given in Table 3-1.

Table 3-1 Specifications of the network

Factor	Specification
Number of nodes $N$	3, 8, and 15 normal nodes + 1 master node
Step size $\Delta t$	0.001
Topology	Grid, Hexa, and Random
Initial time values	Randomly selected from [0.20-0.30]

The output results for each network described in section 3.2 will be explained in the following subsections.



### 3.3.1 Simulation Results for Small Networks

The simulation results for the small size networks of 4 nodes with different topologies are shown. For each topology, two plots are shown which are the average time of each node versus master time and the error between the average value and the master node time value.

#### I. 4-Nodes Grid Topology:

Figure 3-14 and Figure 3-15 show the average time calculated using the averaging protocol and error value between the average time and master node time for each node at each iteration, respectively. As shown in Figure 3-13, time values at each node starts with initial random values and they are kept updated at each iteration until the values are close to each other. Additionally, it is observed from the error curves in Figure 3-14 that the this proposed averaging protocol exhibits an important phenomenon in which it reaches a minimum error value in a region in the middle of the curve called (**Dip Region**) way before it converges to the master node value at the (**Steady State Region**) as shown in Figure 3-15. This implies that this protocol can achieve two goals which are reaching convergence earlier than the steady state which means that we need less number of iteration and hence operations plus at this dip region actually the error is even smaller than the steady state error which means that the synchronization is more accurate.

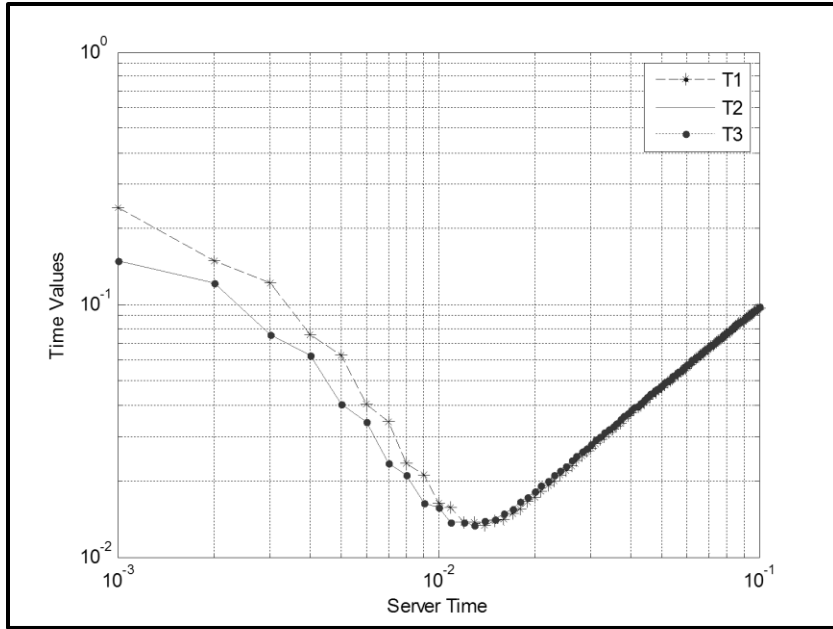


Figure 3-14 Time values for each node in 4-Nodes Grid Topology

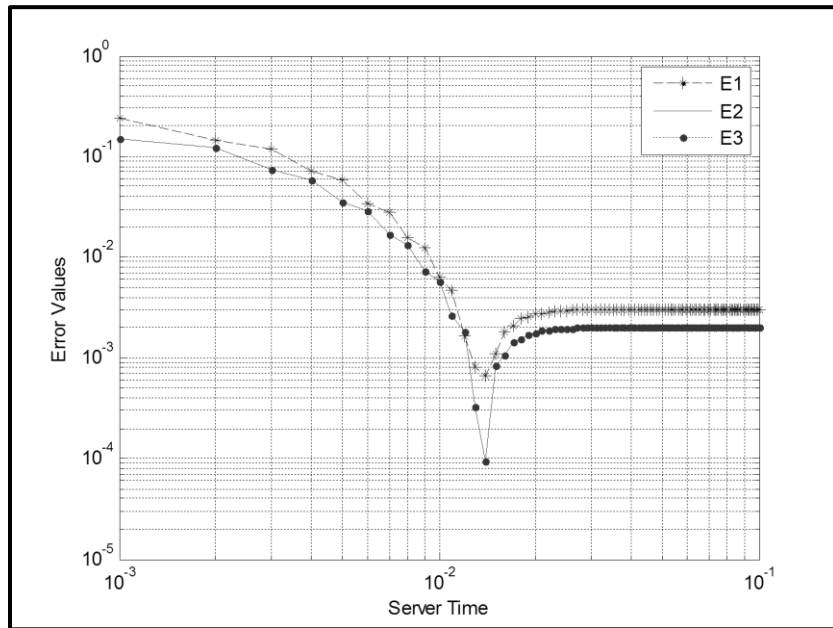


Figure 3-15 Error values for each node in 4-Nodes Grid Topology

From the time and error figures shown above, we can summarize the critical values for the time and error as shown in Table 3-2 and the average number of iterations to reach the average minimum error for all nodes was noted to be 15 iteration with 0.00028695 error value.

Table 3-2 Summarized data for 4-Nodes Grid Topology

3+1-nodes	Dip Region	
Nodes	Iterations	Error Value
N1	15	0.000671774
N2	15	9.45E-05
N3	15	9.45E-05
Maximum	15	0.000671774
Minimum	15	9.45386E-05
Deviation	0	0.000577235
Average	15	0.00028695

**II. 4-Nodes Hexa Topology:**

Likewise for Hexa topology, Figure 3-16 and Figure 3-17 show the average time calculated using the averaging protocol and error value between the average time and master node time for each node at each iteration, respectively. As shown in Figure 3-16, time values at each node starts with initial random values and they are kept updated at each iteration until the values are close to each other. Additionally, it is observed from the error curves in Figure 3-17 that the this proposed averaging protocol exhibits an important phenomenon in which it reaches a minimum error value in a region in the middle of the curve called (**Dip Region**) way before it converges to the master node value at the (**Steady State Region**) as shown in Figure 3-17. This implies that this protocol can achieve two goals which are reaching convergence earlier than the steady state which means that we need less number of iteration and hence operations plus at this dip region actually the error is even smaller than the steady state error which means that the synchronization is more accurate. And still the behavior of the error curve for both Grid and Hexa topologies has same trends.

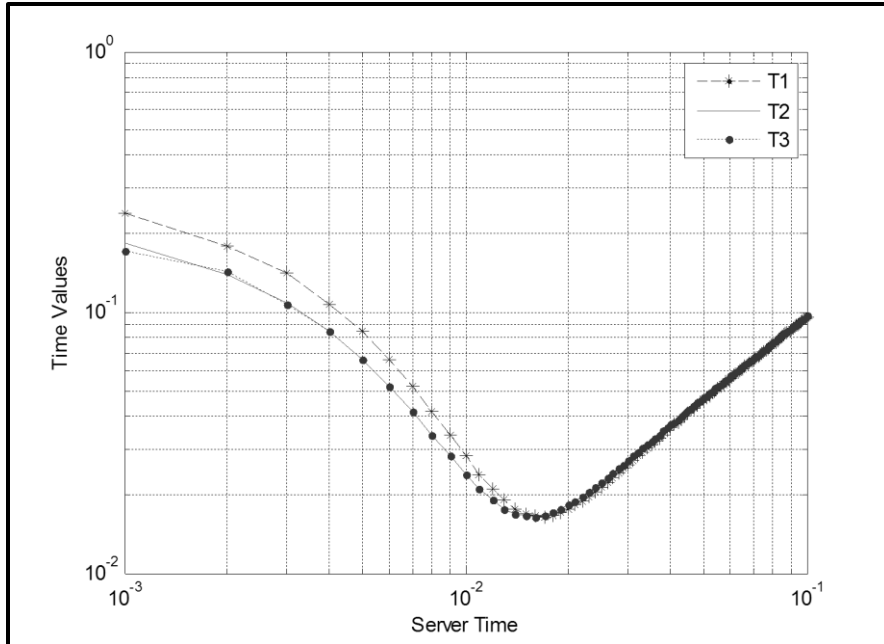


Figure 3-16 Time values for each node in 4-Nodes Hexa Topology

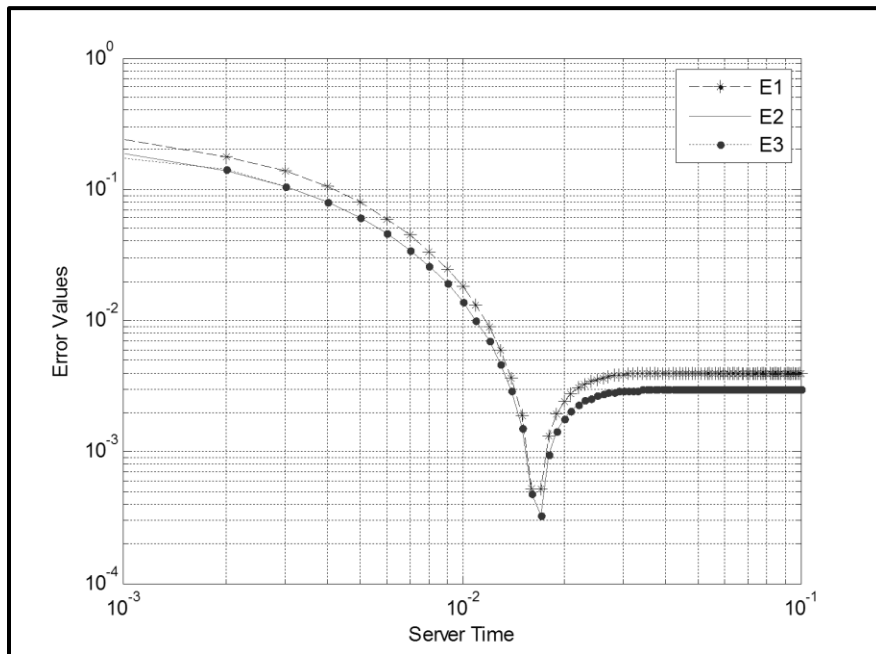


Figure 3-17 Error values for each node in 4-Nodes Hexa Topology

From the time and error figures shown above, we can summarize the critical values for the time and error as shown in Table 3-3 and the average number of iterations to reach

the average minimum error for all nodes was noted to be 18 iteration with 0.000395728 error value.

Table 3-3 Summarized data for 4-Nodes Hexa Topology

3+1-nodes	Dip Region	
Nodes	Iterations	Error Value
N1	18	0.000523788
N2	18	3.32E-04
N3	18	3.32E-04
Maximum	18	0.000523788
Minimum	18	0.000331698
Deviation	0	0.000192089
Average	18	0.000395728

### III. 4-Nodes Random Topology:

Same thing in Random topology, Figure 3-18 and Figure 3-19 shows the time and error values for each node. The time curves start with different values and keep updated until the values are close to each other. Similarly, the error curves have two regions, one in the middle with the minimum value of error (**Dip Region**) and the second one in the last section of the curve when the error values reach the steady state values (**Steady State Region**) as illustrated in Figure 3-19. Also, still the error curve has same behavior for Random comparing to Grid and Hexa topologies.

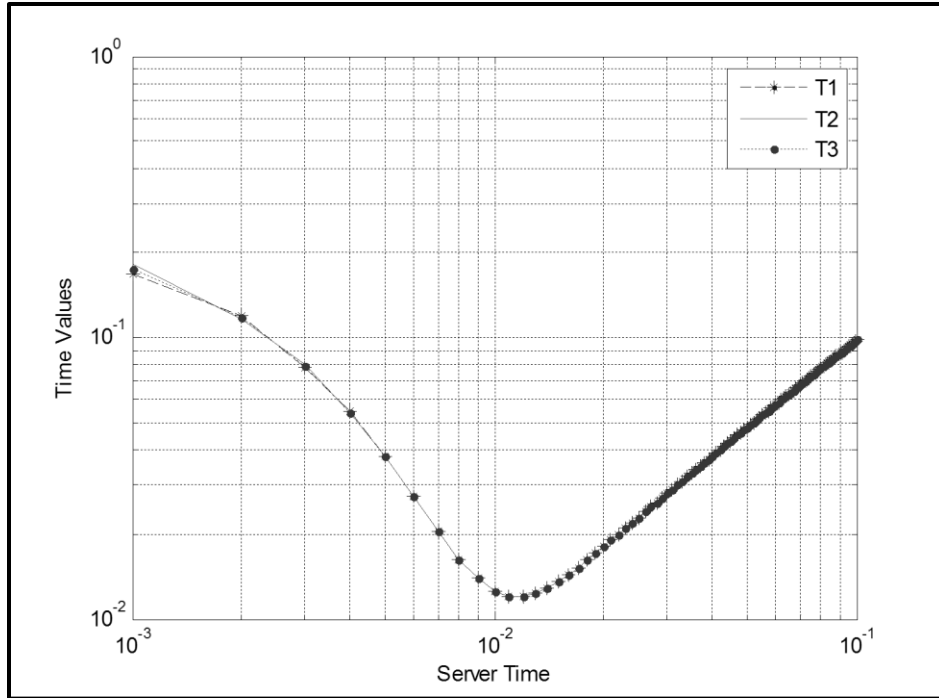


Figure 3-18 Time values for each node in 4-Nodes Random Topology

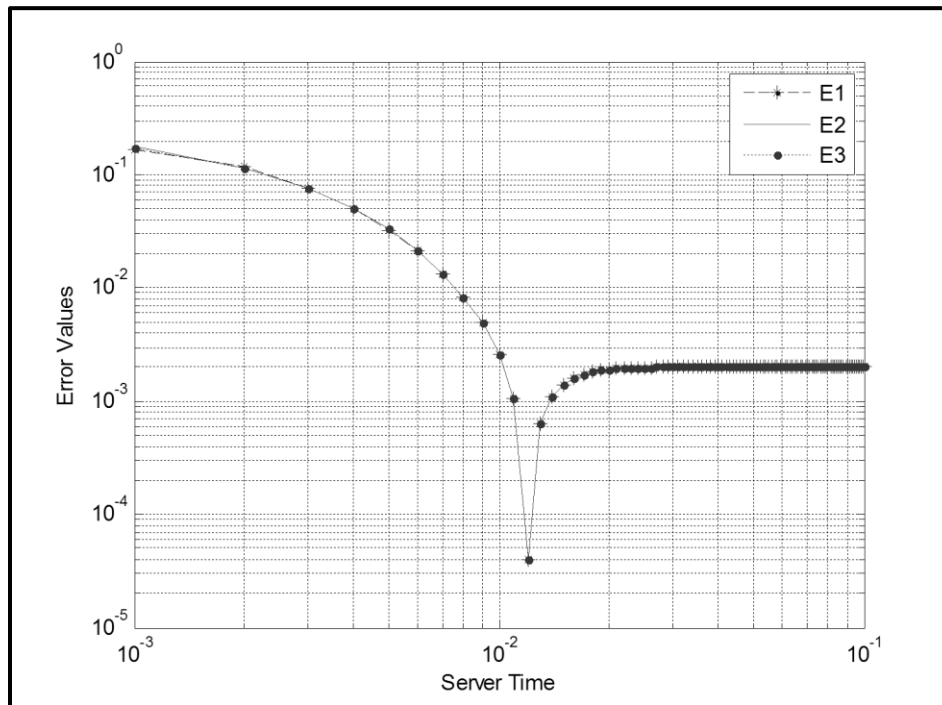


Figure 3-19 Error values for each node in 4-Nodes Random Topology

From the time and error figures above, we can summarize the critical values for the time and error as indicated in Table 3-4 and the average number of iterations to reach the average minimum error for all nodes is equal 13 iteration with 0.0000402741 error value.

Table 3-4 Summarized data for 4-Nodes Random Topology

3+1-nodes	Dip Region	
Nodes	Iterations	Error Value
N1	13	4.03E-05
N2	13	4.02E-05
N3	13	4.03E-05
Maximum	13	4.03107E-05
Minimum	13	4.02398E-05
Deviation	0	7.08835E-08
Average	13	4.02741E-05

### 3.3.2 Simulation Results for Large Networks

In this part, we simulated two sizes of networks (9 and 16 nodes) for different topologies and we show the time and error curves for each node in these networks.

#### I. 9-Nodes Grid Topology:

Figure 3-20 and Figure 3-21 show the time and error values for each node; where the time curves starting with different values and keep updated until the values are closed to each other. The error curves still have two regions, one in the middle with the minimum value of error (Dip Region) as in Figure 3-21 and the other one in the last part of the cure when the error values reach the steady state values as in Figure 3-21.

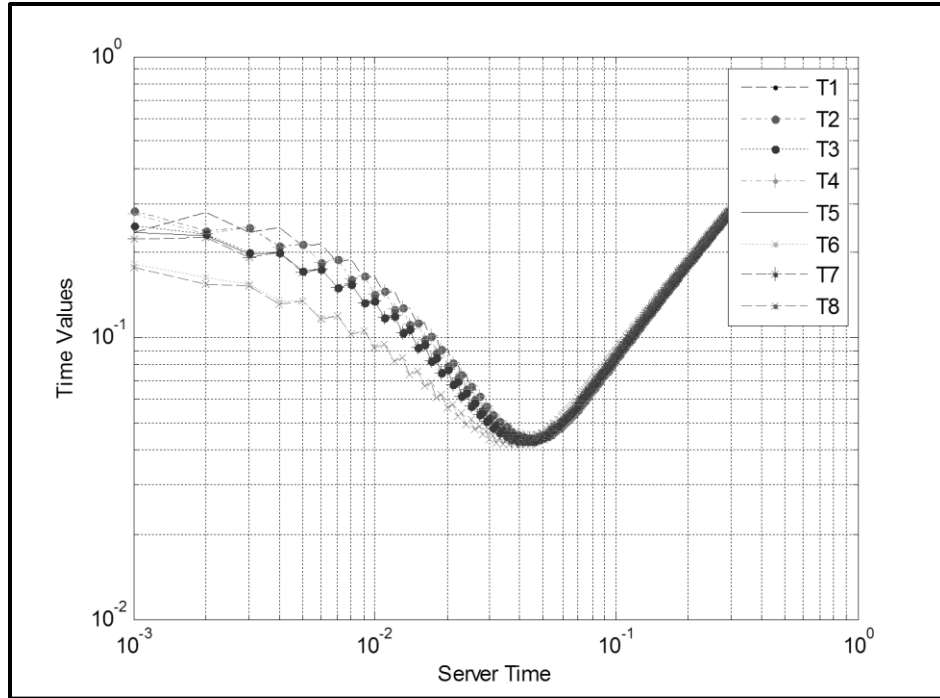


Figure 3-20 Time values for each node in 9-Nodes Grid Topology

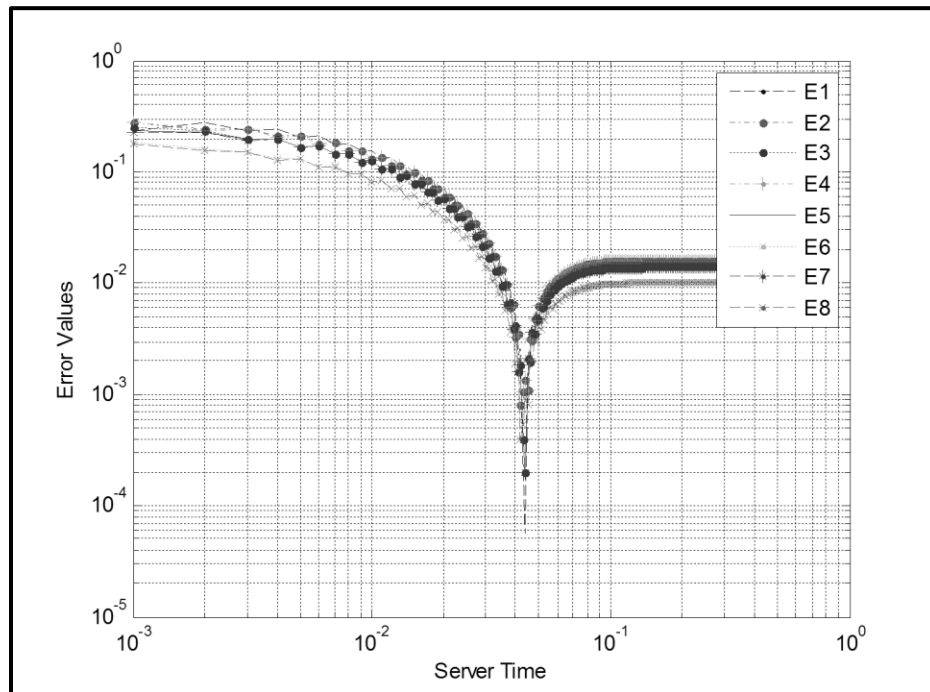


Figure 3-21 Error values for each node in 9-Nodes Grid Topology

From the time and error Figure 3-20 and Figure 3-21 respectively, we can summarize the critical values for the time and error as laid out in Table 3-5 and the average number of



iterations to reach the average minimum error for all nodes was recorded to be 44 iteration with 0.000384398 error value.

Table 3-5 Summarized data for 9-Nodes Grid Topology

8+1-nodes	Dip Region	
Nodes	Iterations	Error Value
N1	45	5.70E-05
N2	43	0.000815579
N3	45	0.000200615
N4	43	0.000815579
N5	45	0.000200615
N6	43	0.0003926
N7	45	0.000200615
N8	43	0.0003926
Maximum	45	0.000815579
Minimum	43	5.69782E-05
Deviation	2	0.000758601
Average	44	0.000384398

## II. 9-Nodes Hexa Topology:

Figure 3-22 and Figure 3-23 show the time and error values for each node whereby the time curves start with different values and are periodically updated until the values are closed to each other. The error curves have two regions, one in the middle with the minimum value of error (Dip Region) and the second one in the last part of the curve when the error values reach the steady state values (Steady State Region) as in Figure 3-23.

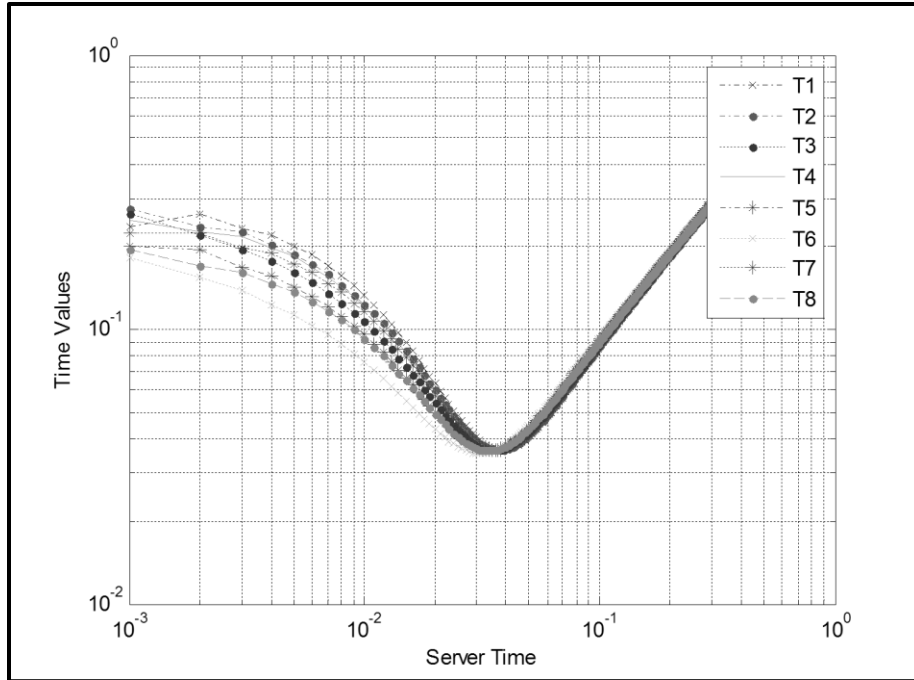


Figure 3-22 Time values for each node in 9-Nodes Hexa Topology

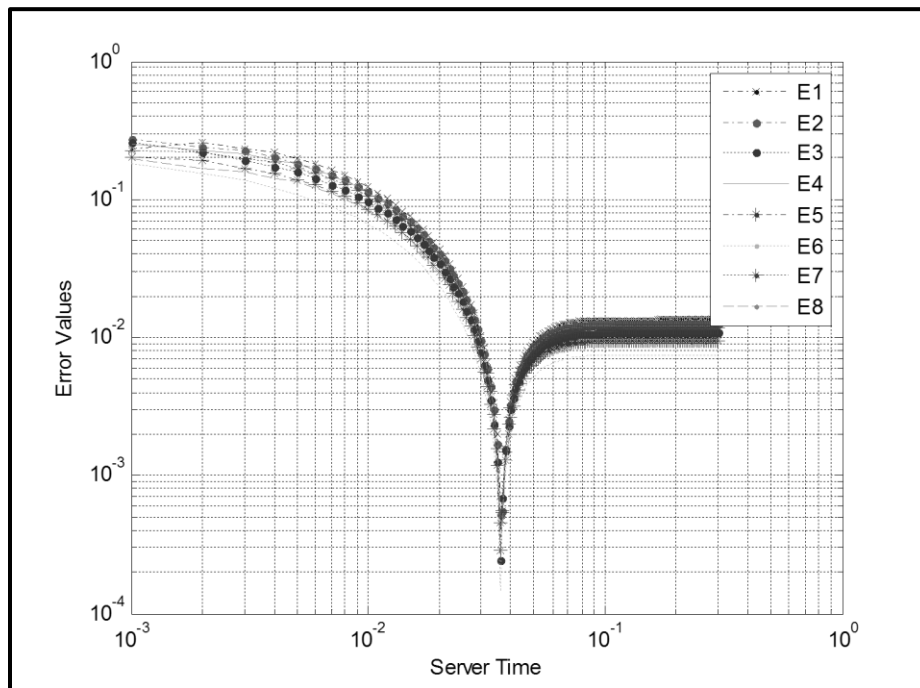


Figure 3-23 Error values for each node in 9-Nodes Hexa Topology

From the time and error figures above, we can summarize the critical values for the time and error as in Table 3-6. The average number of iterations to reach the average minimum error for all nodes is equal 37.125 iteration with 0.000377289 error value.

Table 3-6 Summarized data for 9-Nodes Hexa Topology

8+1-nodes	Dip region	
Nodes	Iterations	Error Value
N1	38	4.70E-04
N2	37	0.000527282
N3	37	0.000241893
N4	37	0.000532532
N5	37	0.000285639
N6	37	0.000142758
N7	37	0.000458648
N8	37	0.00035947
Maximum	38	0.000532532
Minimum	37	0.000142758
Deviation	1	0.000389774
Average	37.125	0.000377289

### III. 9-Nodes Random Topology:

Figure 3-24 and Figure 3-25 show the time and error values for each node in which the time curves start with different values and keep on being updated until the values are closed to each other. The error curves have two regions, one in the middle with the minimum value of error (Dip Region) and the second one in the last when the error values reach the steady state values (Steady State Region).

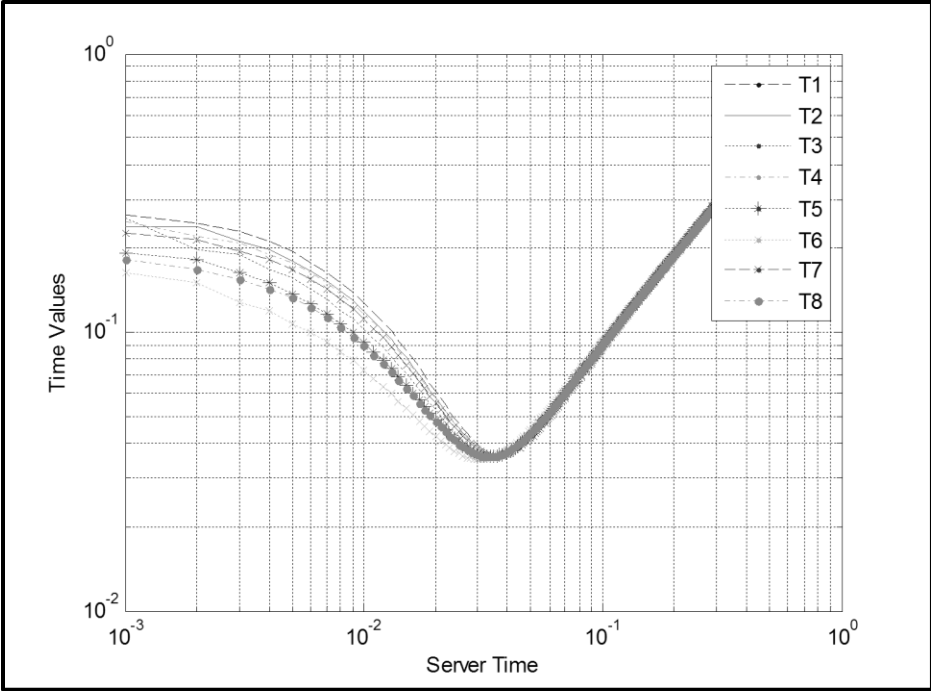


Figure 3-24 Time values for each node in 9-Nodes Random Topology

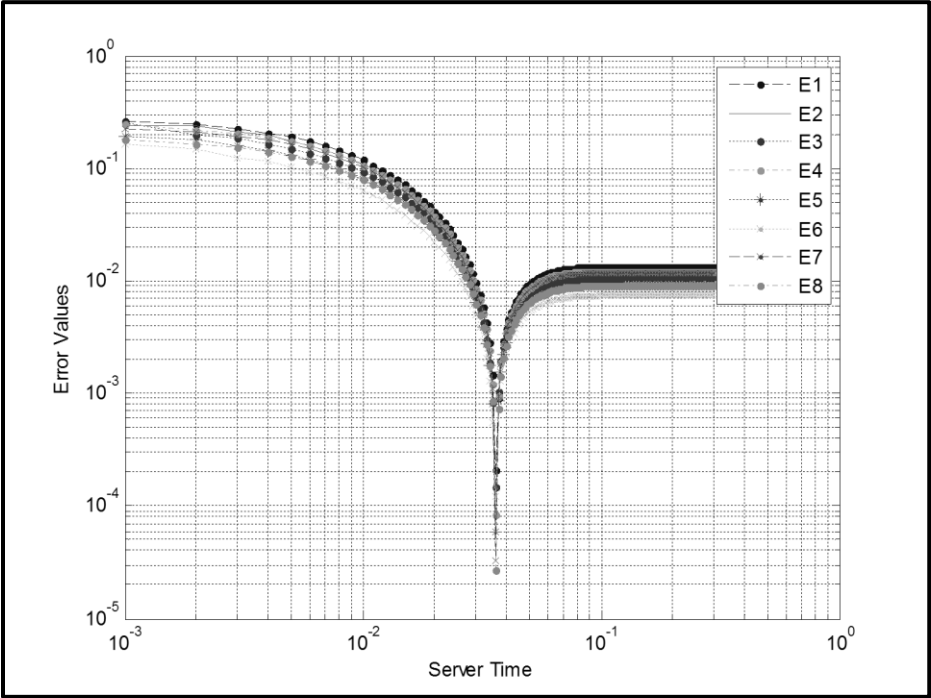


Figure 3-25 Error values for each node in 9-Nodes Random Topology

From the time and error figures above, we can summarize the critical values for the time and error as in Table 3-7 and the average number of iterations to reach the average minimum error for all nodes is equal 37 iteration with 0.0000942024 error value.

Table 3-7 Summarized data for 9-Nodes Random Topology

8+1-nodes	Dip Region	
Nodes	Iterations	Error Value
N1	37	2.10E-04
N2	37	7.04E-05
N3	37	0.000146435
N4	37	8.44E-05
N5	37	5.85E-05
N6	37	0.000123708
N7	37	3.28E-05
N8	37	2.71E-05
Maximum	37	0.000210247
Minimum	37	2.71062E-05
Deviation	0	0.000183141
Average	37	9.42024E-05

#### **IV. 16-Nodes Grid Topology:**

The minimum time and error values for each node in this network is as indicated in Table 3-8. From this table we can see that the average number of iterations needed to reach the average minimum error of all nodes is 79 iterations with 0.000342492 error value.

Table 3-8 Summarized data for 16-Nodes Grid Topology

15+1	Exact in Dip Region	
Nodes	Iterations	Error Value
N1	80	0.000711993
N2	80	0.000145546
N3	79	0.000633557
N4	79	0.000265769
N5	80	0.000145546
N6	79	0.00075918
N7	79	0.000277854
N8	79	7.58253E-05
N9	79	0.000633557
N10	79	0.000277854
N11	79	7.74176E-05
N12	79	0.000449196
N13	79	0.000265769
N14	79	7.58255E-05
N15	79	0.000449196
Maximum	80	0.00075918
Minimum	79	7.58253E-05
Average	79.2	0.000349605

**V. 16-Nodes Hexa Topology:**

The minimum time and error values for each node in this network are shown in Table 3-9. From this table we can see that the average number of iterations needed to reach the average minimum error of all nodes is 88.3 iterations with 0.000256281 error value.

Table 3-9 Summarized data for 16-Nodes Hexa Topology

15+1	Exact in Dip Region	
Nodes	Iterations	Error Value
N1	89	0.000359786
N2	89	0.000472502
N3	88	0.000313058
N4	88	4.88986E-05
N5	89	0.000456458
N6	88	0.000379195
N7	88	2.22991E-05
N8	88	0.00013476
N9	89	0.000529627
N10	88	0.000381809
N11	88	9.87801E-07
N12	88	0.000154575
N13	88	0.000380926
N14	88	8.68129E-05
N15	88	0.000122524
Maximum	89	0.000529627
Minimum	88	9.87801E-07
Average	88.26667	0.000256281

**VI. 16-Nodes Random Topology:**

The minimum time and error values for each node in this network are as shown in Table 3-10. From this table we can see that the average number of iterations needed to reach the average minimum error of all nodes is 90 iterations with 0.0000999658 error value.

Table 3-10 Summarized data for 16-Nodes Random Topology

15+1	Exact in Dip Region	
Nodes	Iterations	Error Value
N1	90	0.000206789
N2	90	0.000127355
N3	90	6.06407E-05
N4	90	0.000121399
N5	90	0.000148503
N6	90	3.0945E-05
N7	90	0.000101025
N8	90	0.000143706
N9	90	0.000135718
N10	90	9.23675E-05
N11	90	1.07263E-05
N12	90	8.36796E-05
N13	90	0.000144487
N14	90	7.95591E-05
N15	90	1.25872E-05
Maximum	90	0.000206789
Minimum	90	1.07263E-05
Average	90	9.99658E-05

### 3.4 Practical Results

This part indicates how the protocol was implemented using MEMSIC components. Micaz nodes are used and the operating system of those nodes is TinyOS that is using nesC as a programming language. Micaz nodes contain different kind of hardware components like: MPR2400 (Micaz mote), MIB520 gateway, and sensing boards.

#### 3.4.1 Practical Results for Small Networks

In this part, practical implementation of one size of networks (4 nodes) with different topologies described in section 3.2.1, and the time and error curves for each node in these networks will be discussed as follow:



### I. 4-Nodes Grid Topology:

Similar to the simulation part, the time curves start with different values and keep updated until values are close to each other. The error curves likewise have two regions, one in the middle with the minimum value of error (**Dip Region**) and the second one in the last when the error values reach the steady state values (**Steady State Region**) as indicated in Figure 3-26 and Figure 3-27.

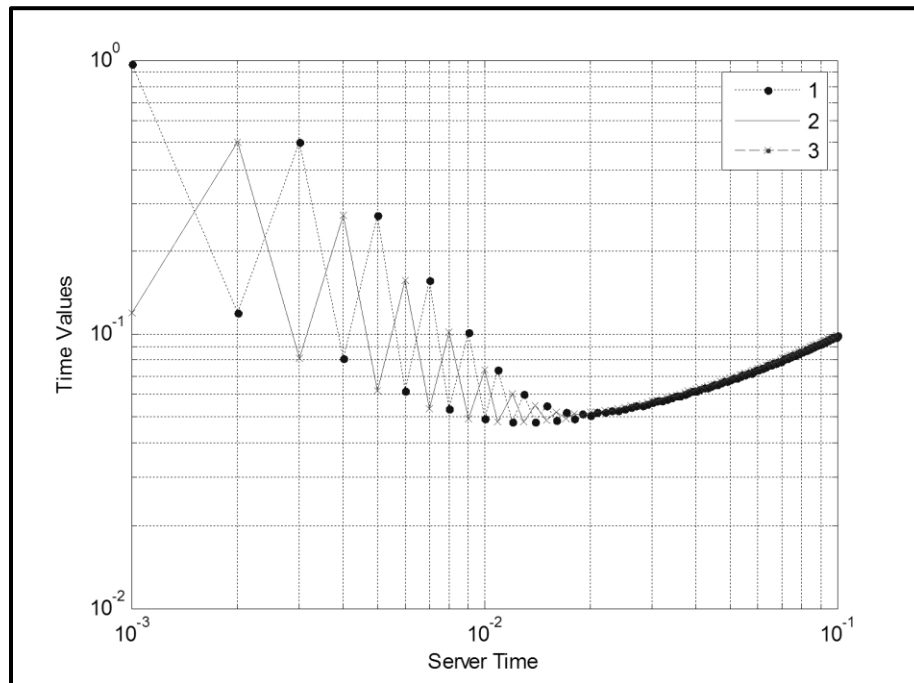


Figure 3-26 Time values for each node for a 4-Nodes Grid Topology

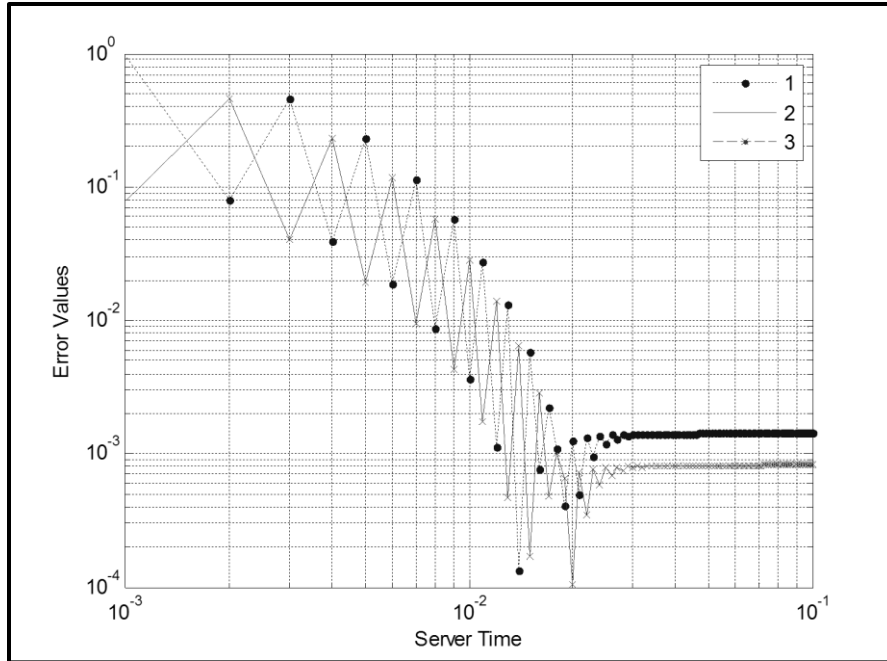


Figure 3-27 Error values for each node for a 4-Nodes Grid Topology

From the time and error figures above, the critical values for the time and error are summarized in Table 3-11. The average number of iterations to reach the average minimum error for all nodes was found to be 15.3 iterations with 0.000197656 error value.

Table 3-11 Summarized practical data for 4-Nodes Grid Topology

<b>3+1</b>		
Exact in Dip Region		
Nodes	Iterations	Error Value
N1	18	0.000463281
N2	14	6.48437E-05
N3	14	6.48437E-05
Maximum	18	0.000463281
Minimum	14	6.48437E-05
Average	15.33333	0.000197656

### 3.4.2 Practical Results for Large Networks

Two sizes of networks (9 and 16 nodes) with different topologies were also implemented. As well, the time and error curves for each node in these networks are discussed as follow.

#### I. 9-Nodes Grid Topology:

For the higher size of networks, still the behavior of the time and error curves for practical results have same characteristics for each node when compared with the simulation results. The time curves Figure 3-28 start with different values and they keep updated until the values are close to each other. Repeatedly, the error curves have two regions, one in the middle with the minimum value of error (Dip Region) as shown in Figure 3-29 and the second one in the extreme end of the curve occurring when the error values reach the steady state values (Steady State Region) as illustrated in Figure 3-29.

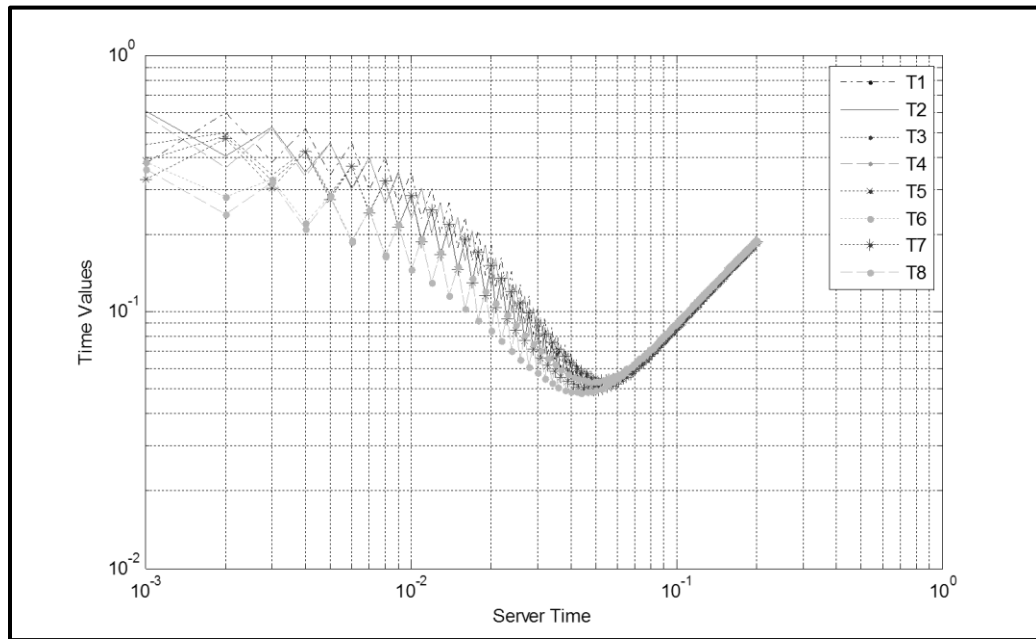


Figure 3-28 Time values for each node for a 9-Nodes Grid Topology

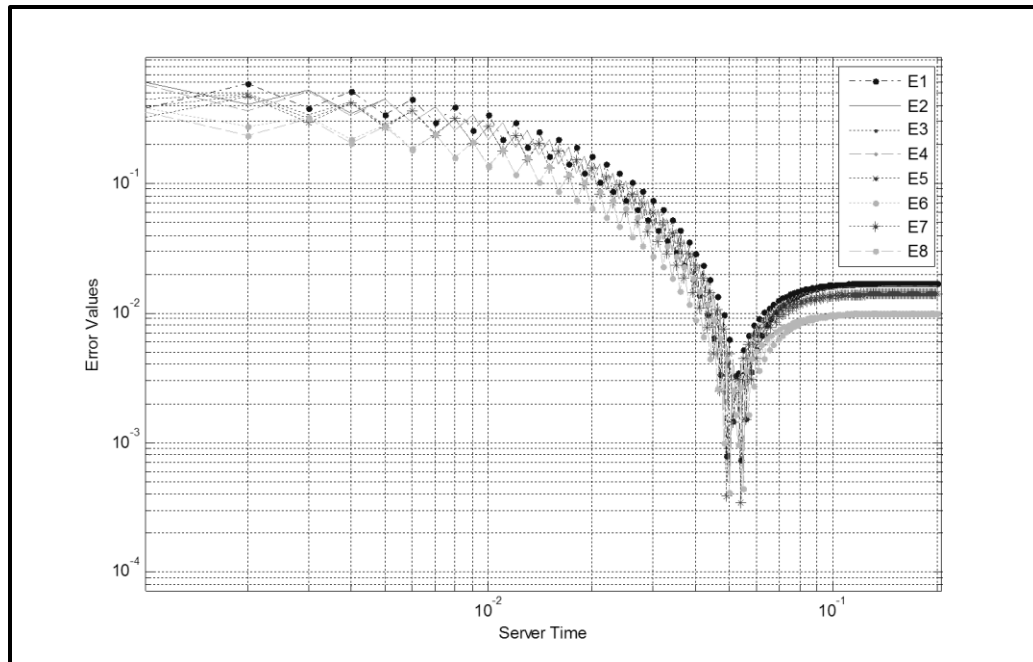


Figure 3-29 Error values for each node for a 9-Nodes Grid Topology

From the time and error figures above, we can summarize the critical values for the time and error as in Table 3-12 and the average number of iterations to reach the average minimum error for all nodes is equal 53 iteration with 0.000442363 error value.

Table 3-12 Summarized practical data for 9-Nodes Grid Topology

8+1	Exact in Dip Region	
Nodes	Iterations	Error Value
N1	55	0.000729527
N2	51	0.000480831
N3	55	0.000343489
N4	51	0.000480831
N5	55	0.000343489
N6	51	0.000408626
N7	55	0.000343489
N8	51	0.000408626
Maximum	55	0.000729527
Minimum	51	0.000343489
Average	53	0.000442363

## II. 16-Nodes Grid Topology:

Also, in 16-Nodes still the behavior of the time and error curves for practical results has same characteristics for each node when compared with the simulation results. The time curves start with different values and they keep updated until the values are close to each other as in Figure 3-30. Repeatedly, the error curves have two regions, one in the middle with the minimum value of error (Dip Region) as shown in Figure 3-31 and the second one in the extreme end of the curve occurring when the error values reach the steady state values (Steady State Region) as illustrated in Figure 3-31.

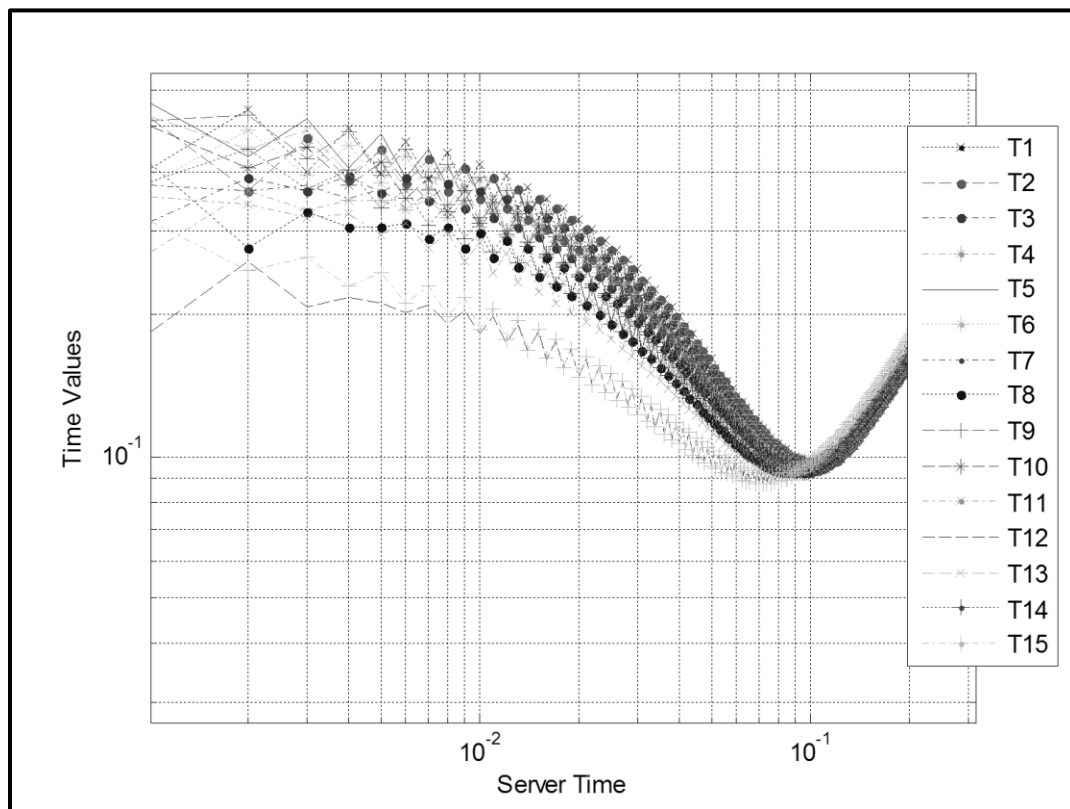


Figure 3-30 Time values for each node in 16-Grid Topology

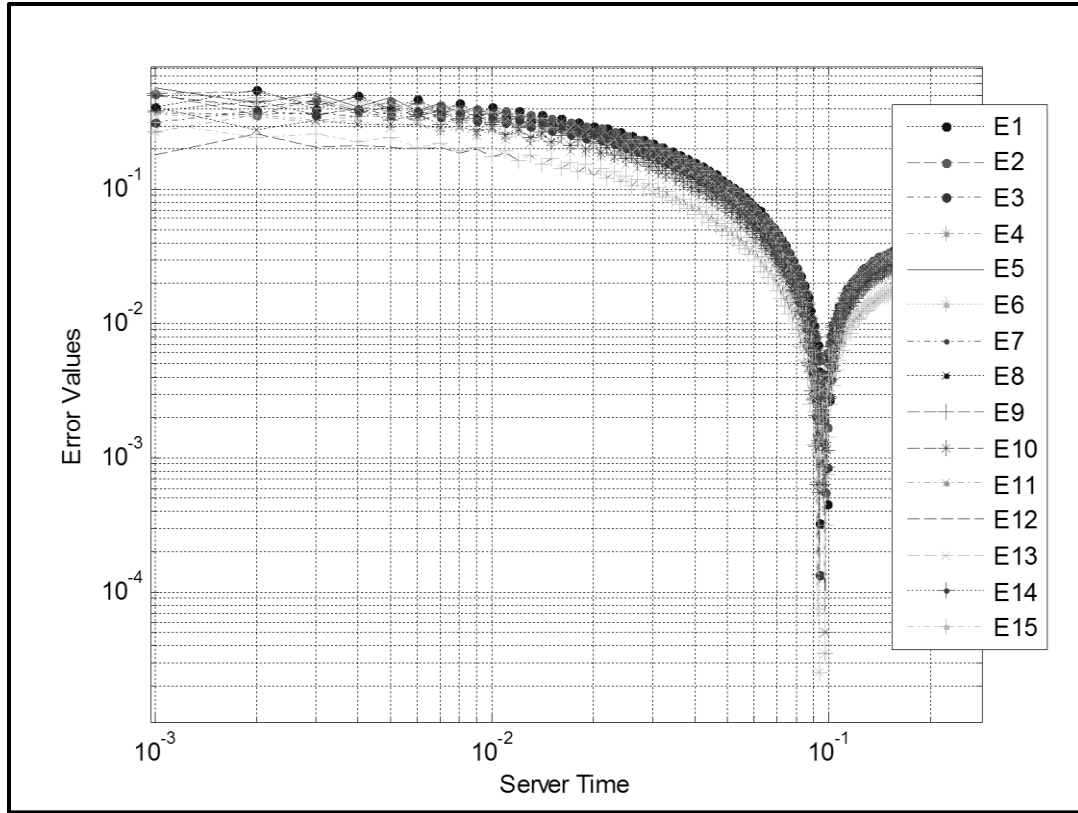


Figure 3-31 Error values for each node in 16-Grid Topology

From the time and error figures above, we can summarize the critical values for the time and error as in Table 3-13 and the average number of iterations to reach the average minimum error for all nodes is equal 95.67 iterations with 0.000238497 error value.

Table 3-13 Summarized practical data for 16-Nodes Grid Topology

15+1	Exact in Dip Region	
Nodes	Iterations	Error Value
N1	94	0.000328504
N2	98	0.000552229
N3	94	0.00013527
N4	98	3.44644E-05
N5	98	0.000552229
N6	94	2.5456E-05
N7	98	4.97064E-05
N8	94	0.000553441
N9	94	0.00013527
N10	98	4.97064E-05
N11	97	0.000425289
N12	93	7.39918E-05
N13	98	3.44644E-05
N14	94	0.000553441
N15	93	7.39918E-05
Maximum	98	0.000553441
Minimum	93	2.5456E-05
Average	95.66667	0.000238497

From the above simulation and practical results; increasing the number of nodes for different topologies requires more iteration value to reach the dip region and achieve the minimum value of error. Also, the error curves for all scenarios have same shape.

### 3.5 Summary

In this thesis, a new protocol has been introduced that consists of simpler operations like sum and division compared to other previous protocols. Each node in the network communicates with only the nearest neighbors after which it averages the time of those neighbors and updates its time regarding to this value. Usually, nodes keep updating their values until a convergence state is reached. This protocol depends only on the local information of each node without knowing anything about the global information of the

whole network. So, since it is simple and does not need reference node, this protocol is suitable for all networks either light or dense with different shapes and can be deployed in harsh environments.

From all examples that have been simulated and implemented, it has been noticed that the error curves have two regions; the first one is the dip region that has minimum error values with less number of iterations and the second region when the system reach the steady state though the number of iterations needed to reach this region is larger than that needed in the first region. If the first region can be stopped, then the energy and memory of the sensor nodes can be saved with less number of iterations as well as with minimum value of error. Most of the previous protocols did not exhibit this behavior for the error plots collected between the two regions and this will be discussed in Chapter 4.

This implies that this protocol can achieve two goals which are reaching convergence earlier than the steady state which means that we need less number of iteration and hence operations plus at this dip region actually the error is even smaller than the steady state error which means that the synchronization is more accurate.

Practically, the node only knows the time value of its clock and the neighboring nodes values. The question here and to benefit from the protocol, how a node knows that it reaches to the minimum error value by observing only its own time values. Also, add to this that the observation window must not be too large since the protocol must be kept simple and light and should not use large processing power or large memory size from the nodes. These questions will be answered and analyzed in Chapter 4.



Chapter 4 discusses theoretically and practically how the node knows when to stop in the dip region in or close to the minimum error value by developing a stopping criterion from only the time values of the nodes itself without knowing much about the network in general.

# CHAPTER FOUR

## Stopping Criterion

This chapter introduces a criterion that enables the node, from only its own time values calculated from the AP, to know that it reaches an acceptable synchronization accuracy in the dip region before reaching the steady state region and stops synchronization process. This is called stopping criterion (SC) protocol. The chapter discusses the stopping criterion concept for the iterative process, it uses one of the steady state stopping criteria that had been mentioned in the previous researches, and introduces a new stopping criterion to stop in the dip region. First, it introduces the stopping criterion concept and why it uses in the iterative process. Then, it discusses one of the stopping criteria for the steady state region depending on the previous researches and tests this criterion on both the simulation and experimental parts. Next, it introduces a new stopping criterion to stop in the dip region depending only on the local information for the sensor nodes. Finally, it shows the performance of this criterion in both the simulation and experimental parts for WSNs.

### 4.1 Stopping Criterion

The objective of the stopping criterion in this research is to terminate the iterative process (the iterations) way before the steady state by exploiting the dip behaviour exhibited in the AP algorithm. Without finding a suitable SC, the advantage and characteristics of the AP algorithm will not be exploited. At each iteration of the AP algorithm, the designed

SC is used as decision rule to stop the iterative process and indicated that synchronization has been reached and it works as shown in the below algorithm Figure 4-1.

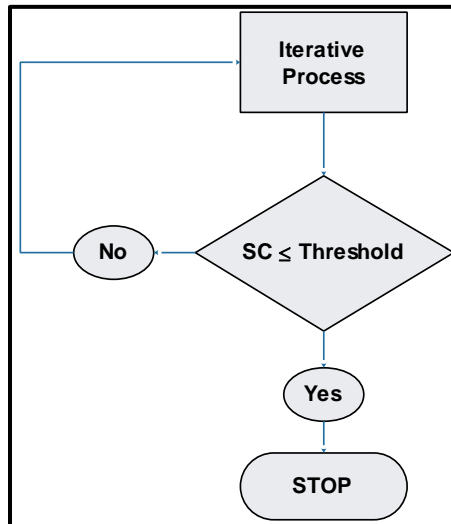


Figure 4-1 SC Concept

In practical applications, the SC terminates the iterative process once it is satisfied and this improves the efficiency of the system. . The SC optimally stops the iterative process when the following conditions occurred:

1. The amount of computation is sufficient.
2. A solution obtained so far is satisfactory.
3. The solution is not satisfactory, but a better one is unlikely to be produced.
4. The method is not able to converge to a solution.
5. Additional computation will provide little or no improvements in the current solution.

As mentioned in Chapter 2, there are many SC exist in the literature. We design a SC that having the following features: simple, robust, requires as little operations as possible, practical, and exploits the dip region of the AP algorithm. Before we introduce our proposed SC, one SC from literature will be tested for our proposed AP algorithm.

## 4.2 Steady State Stopping Criterion

The AP protocol is tested with a number of existing SC to see the usability of them with our AP algorithm. These SC normally works well in the steady state region. When they are applied to our protocol, they stopped the iteration in random different location of the chart; making them impractical to be used. This is because of the unique behaviour of our protocol as shown before. For example, from Figure 4-2 and we deployed the AP algorithm with the absolute SC; some time stops in the three regions and this not our objective, we need to stop either in dip region or in the steady state region.

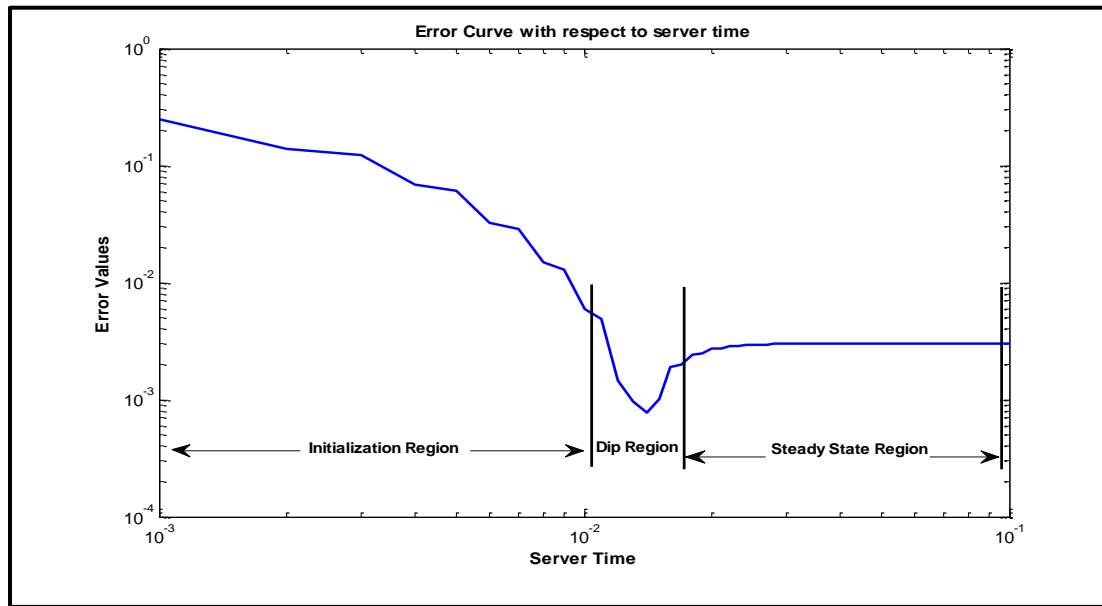


Figure 4-2 Error Curve

To be able to use this SC in the steady state, the absolute SC can be modified as shown in Figure 4-3. The modified absolute SC works by ignoring the dip region, then save the current time and previous time. After that, evaluate the difference between the two values and store it in memory. Next, we do the previous until reach the length of our window if this happens it will evaluate the average value of the differences for the window size.

Then, if the average value is less than the absolute threshold; the iterative process will be stopped but if this not happens it will go back to the iterative process and so on.

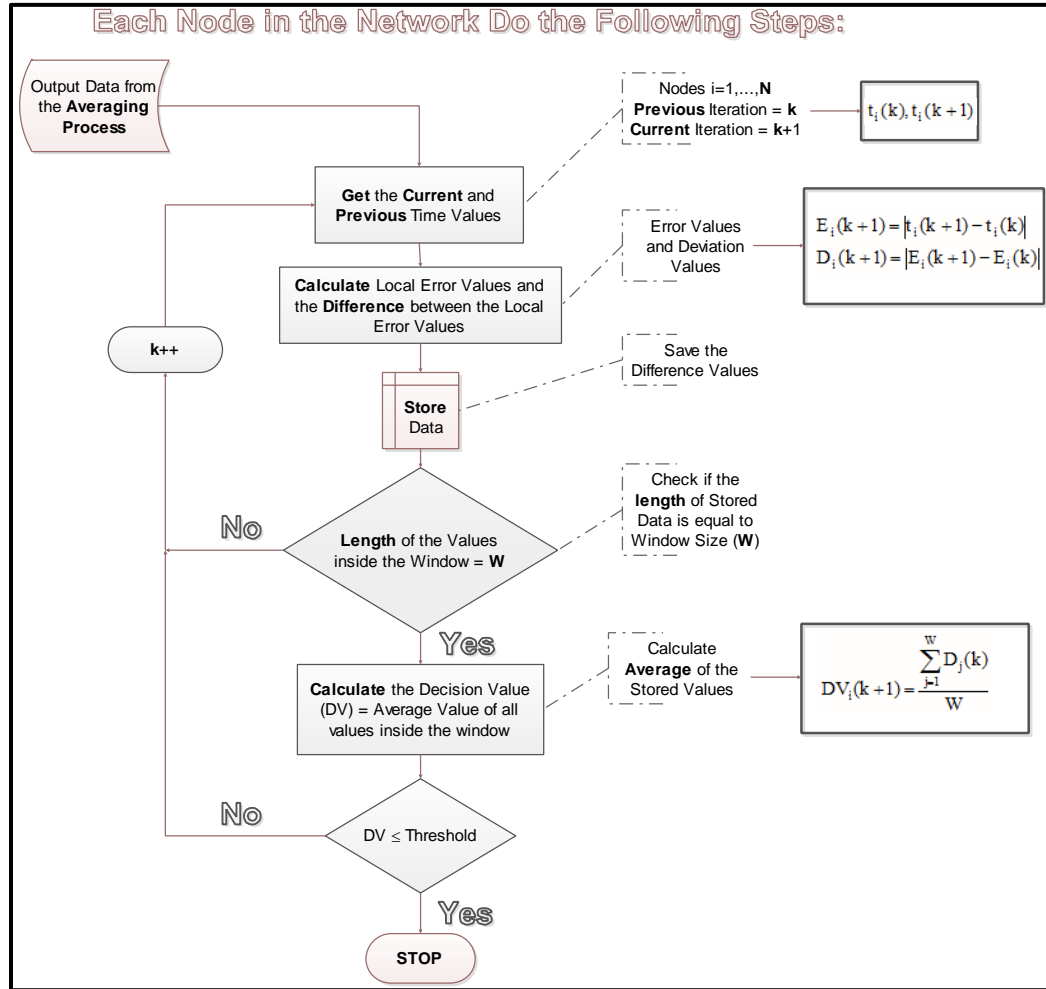


Figure 4-3 Flow chart of the modified SS-SC

We deployed this SC for the networks - (4, 9 and 16)-nodes using only the local values for each node with different size of window (number of samples), and the mathematical representation of this algorithm as in Figure 4-4 by going from the lower to the upper level.

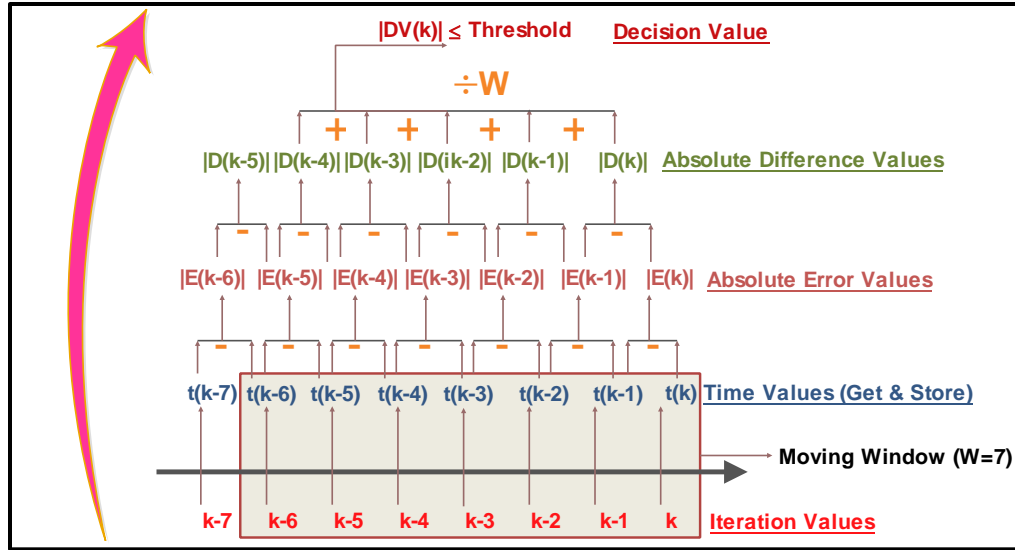


Figure 4-4 Mathematical representation of the modified absolute SC

We deployed this stopping criterion in both the simulation and the practical parts for the previous examples that were discussed in Chapter 3 and this technique succeed in detecting the iterative process for the steady state region with reasonable values as in section 4.4 and 4.5.

### 4.3 Proposed Stopping Criterion for the Dip Region

It was noted that all synchronization error curves in Chapter 3 have the same shape with a dip region where the minimum error values reached before and the error values of the steady state region for all previous networks. Therefore, it is important to propose a new stopping criterion that can terminate the iterative process of the AP algorithm for all nodes in the Dip region by utilizing only the local time information of these nodes without knowing any global information of the whole network. This will save time, memory and battery life for the sensor nodes in the network.

We need to find a way or method that can detect the minimum error value. One way to do that is to use the discrete differentiation operator. It is known that differentiation can

detect a flipping point in a curve. This is what needed to detect the minimum/maximum point in a curve. From the theory of filter design [53], an optimum discrete difference filter can be designed. In literature, two such filters of length 5 and 7 are given by the following impulse responses, respectively:

$$h_1(k) = \frac{-1}{12} \delta(k+2) + \frac{2}{3} \delta(k+1) + \frac{-2}{3} \delta(k-1) + \frac{1}{12} \delta(k-2) \quad 4.1$$

$$h_2(k) = \frac{1}{60} \delta(k+3) + \frac{-3}{20} \delta(k+2) + \frac{3}{4} \delta(k+1) + \frac{-3}{4} \delta(k-1) + \frac{3}{20} \delta(k-2) + \frac{-1}{60} \delta(k-3) \quad 4.2$$

The above two filters are non-causal and a causal version of them will be of length 4 and 6 and are given by:

$$h_1(k) = \frac{-1}{12} \delta(k) + \frac{2}{3} \delta(k-1) + \frac{-2}{3} \delta(k-2) + \frac{1}{12} \delta(k-3) \quad 4.3$$

$$h_2(k) = \frac{1}{60} \delta(k) + \frac{-3}{20} \delta(k-1) + \frac{3}{4} \delta(k-2) + \frac{-3}{4} \delta(k-3) + \frac{3}{20} \delta(k-4) + \frac{-1}{60} \delta(k-5) \quad 4.4$$

Along with this difference filter and to smooth the time data to get better result, averaging filters of the same size of the difference filter is used whether before or after the difference filter. These two filters constitute our method to stop the iteration and detect the minimum value. The method works by filtering the time data of each node with these two finite impulse response (FIR) filters, then checks a sign change of the resultant value at the current iteration with respect to the previous iteration and save it as bipolar data of 1's and -1's in which a 1 represents a positive value and -1 represents a negative value. If

there is a change in the sign, the iterative process stops and declares a minimum value; otherwise the next iteration continues as shown in Figure 4-5.

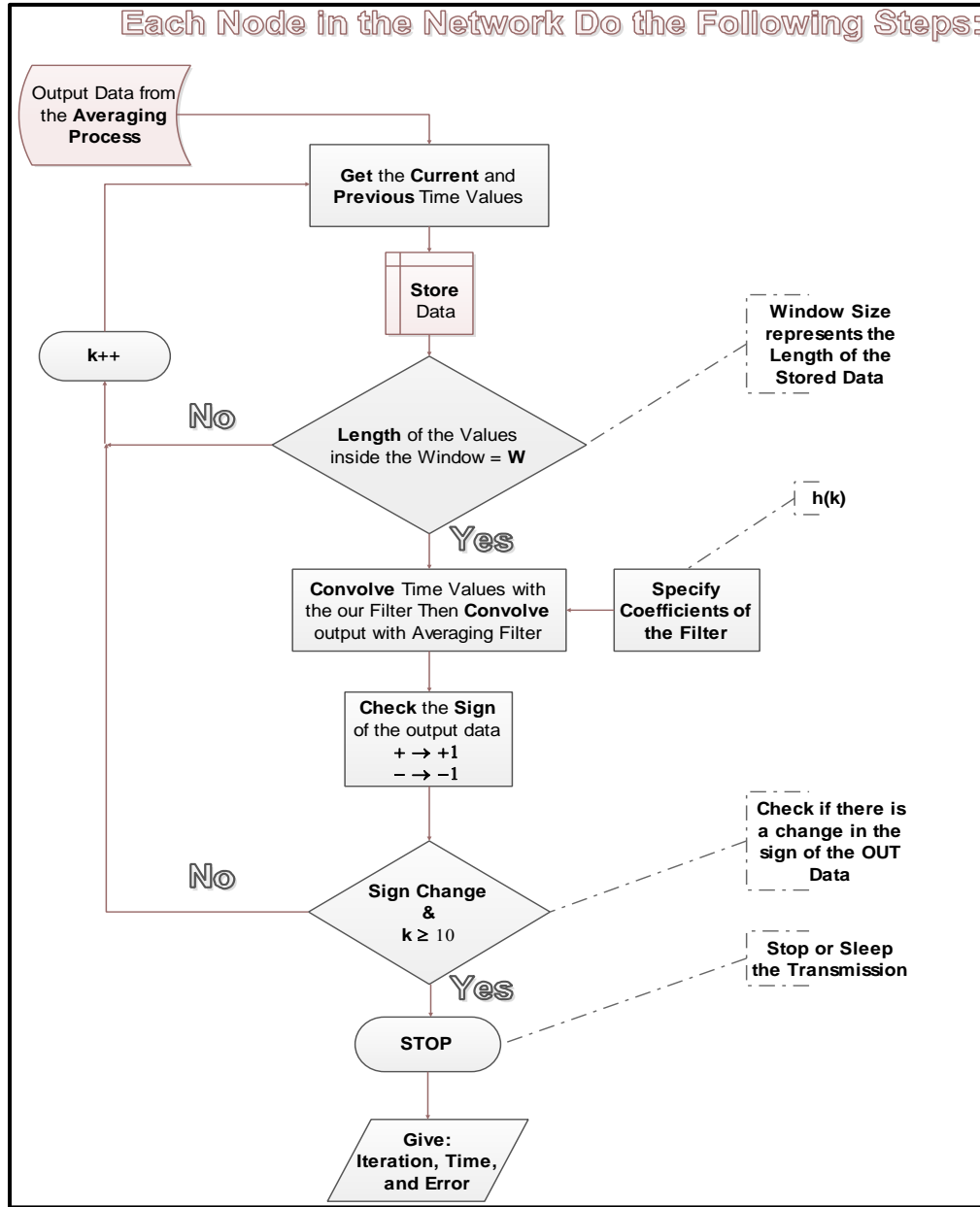


Figure 4-5 Flow Chart of the propose SC

Based on this premise, we wanted to design a better difference filter that works better for our AP algorithm especially in the practical situations. Based on extensive trials, we found a better filter and is given by the following:



$$h_d(k) = 0.2C \cdot \delta(k + 3) + 0.5C \cdot \delta(k + 2) + 0.2C \cdot \delta(k + 1) - 0.2 \cdot \delta(k - 1) - 0.5 \cdot \delta(k - 2) - 0.2 \cdot \delta(k - 3) \quad 4.5$$

Where;  $C$  is used to optimize the filter coefficients for the best results that detect the minimum value. To find the optimal  $C$ , the procedure starts by taking range of  $C$  values. Then, specify the exact iteration value that has the minimum value for each node and determine the detected iteration by the filter for the first  $C$ . After that, subtract the detected iteration from the exact iteration and find the square of these values for each node. Next, define the cost function as the sum of the squared values for all nodes. Finally, repeat the same steps for all  $C$  values and find the minimum cost of all values; where the optimal  $C$  is equal to the  $C$  value that achieves the minimum cost.

Usually, the filter size ( $W$ ) should be selected to achieve the best detection with acceptable cost. Therefore, to study the effect of changing the filter size on the network, we simulated grid network with different sizes 4, 9 and 16 nodes to specify the minimum cost of two filter sizes (5 and 7) and compare between the two sizes. Table 4-1 shows the cost and the optimal  $C$  of the three sizes with the two filter sizes. The cost value for  $W=5$  is less than the cost values for  $W=7$  and with less value of  $C$ ; we see that for 4-nodes the cost value is 1 in both filter sizes, while the cost value for 9 nodes is equal to 43 for  $W=5$  and 53 for  $W=7$ . Therefore, when the size of the filter increases the cost value will be slightly increased and the optimal  $C$  will be closed to one for the Large Network. In addition, increasing the size of the filter requires more processing and memory size in the sensor nodes but it achieves more accurate data and this is what is the required in the synchronization process.

Table 4-1 Summarized Data of different filter sizes (W) for Grid network

W	Parameters	4-nodes Grid	9-nodes Grid	16-nodes Grid
5	C Values	1.421	1.0160	1.00153
	Cost	1	43	915
7	C Values	1.821	1.035	1.002
	Cost	1	53	930

So that, we used the size (W=7) for more accurate detection in the filter stage. Then, we optimized  $C$  parameter for other topologies and we got the following  $C$ -values for different sizes as in Table 4-2, and we used these  $C$  values in the simulated and practical networks:

Table 4-2 Optimized  $C$  values for different topologies and sizes

Topology	4-Nodes	9-Nodes	16-Nodes
<b>Grid</b>	1.821	1.035	1.002
<b>Hexa</b>	1.38	1.045	1.004
<b>Random</b>	2.453	1.04	1.004

From Table 4-2;  $C$  values change from 0.95 to 1.05 for 9 and 16 nodes while for 4 nodes  $C$  changes from 1.3 to 2.5, and we noted that the optimum  $C$  for the dense and Large Network will be closed to 1.

Figure 4-6 shows an example on the cost function for Grid 9-nodes; where is the minimum cost occurred at  $C=1.035$  and it is equal to 53, assume that the filter size equal 7.

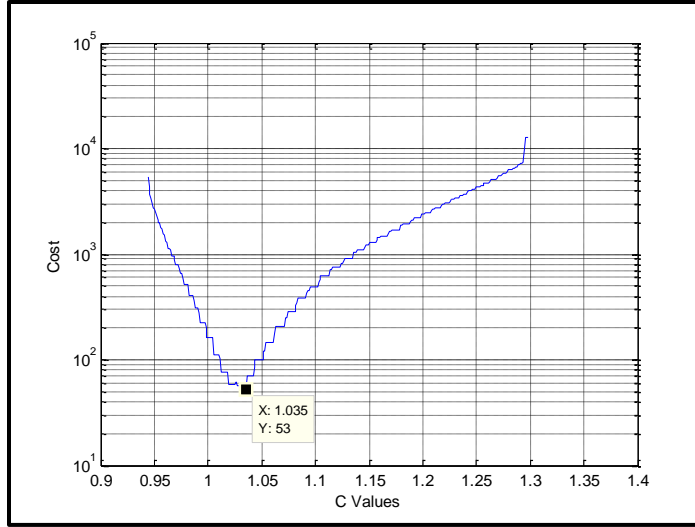


Figure 4-6 Cost function for 9-nodes with Grid Distribution

The responses of this filter in time and frequency domains when  $C = 1$  are shown in Figure 4-7 and Figure 4-8, and the form of this filter that tries to detect the minimum value  $h_d$ , as follow:

$$h_d(k) = 0.2 \cdot \delta(k + 3) + 0.5 \cdot \delta(k + 2) + 0.2 \cdot \delta(k + 1) - 0.2 \cdot \delta(k - 1) - 0.5 \cdot \delta(k - 2) - 0.2 \cdot \delta(k - 3) \quad 4.6$$

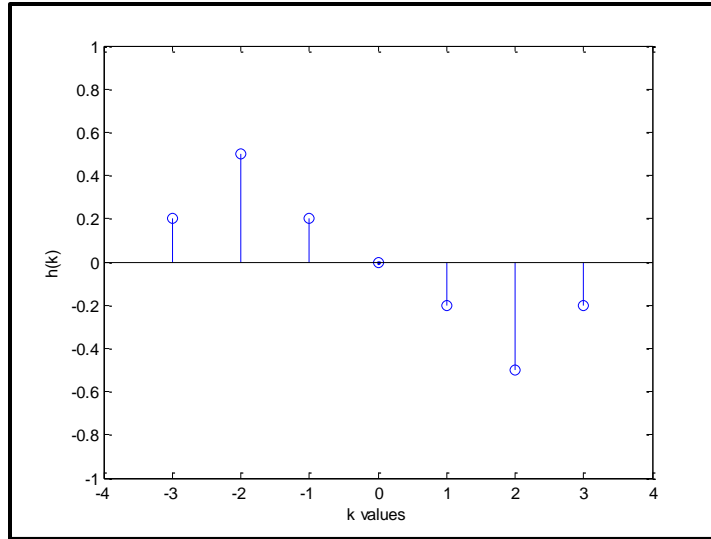


Figure 4-7 Filter in time domain with  $C = 1$

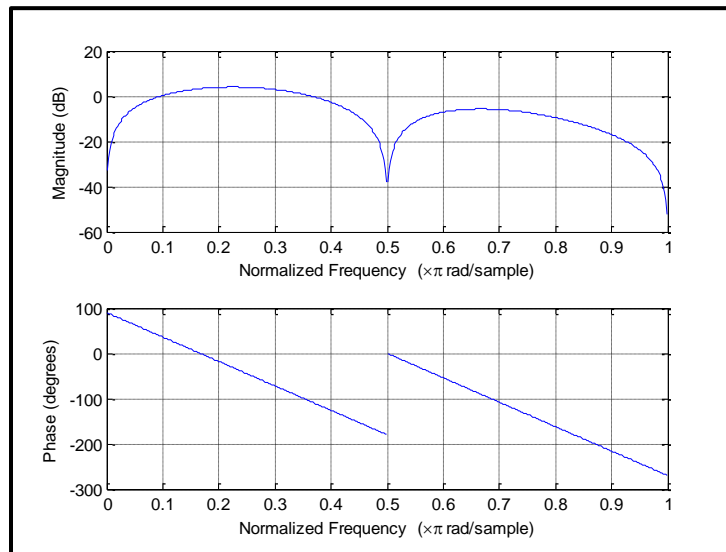


Figure 4-8 Filter in frequency domain

Then, this filter with the optimal  $C$  will be convolved with time values of each node and the output data will be convolved with the rectangular pulse to minimize the effect of the noise (fluctuation of the time values from the AP algorithm); this rectangular pulse can be represented as in Figure 4-9, and the form of this filter  $h_a$  as follow:

$$h_a(k) = \delta(k + 3) + \delta(k + 2) + \delta(k + 1) + \delta(k) + \delta(k - 1) + \delta(k - 2) + \delta(k -$$

3) 4.7

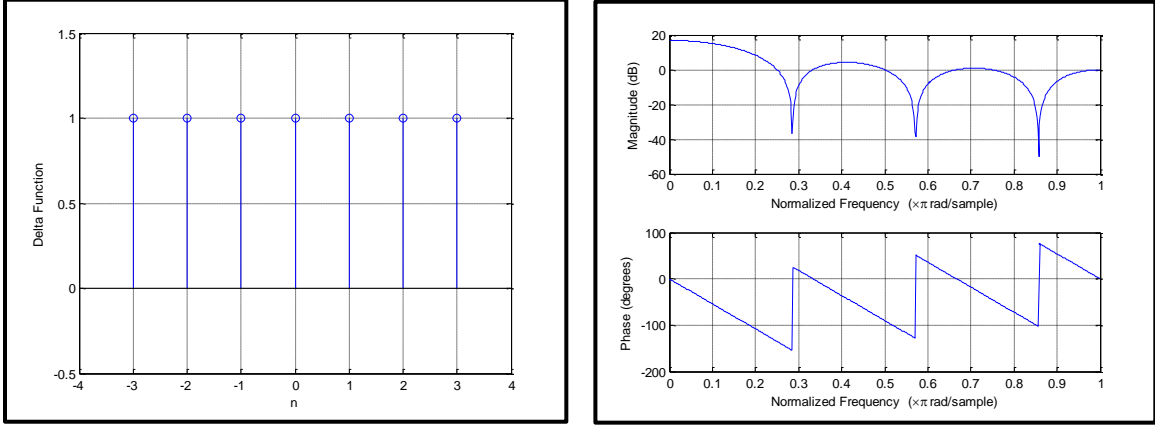


Figure 4-9 Response of the rectangular pulse

In short, we can think of the procedure as convolving the time values with two filters with an overall impulse response that detect the minimum and denoted by  $h_m$ . for node  $i$  and it is given as follows:

$$Out_i(k) = h_d(k) * t_i(k) \quad 4.8$$

$$h_{mi}(k) = Out_i(k) * h_a(k) \quad 4.9$$

Basically, in the AP with this filter; a node stores a finite number of data depending on the size of the filter used. These data come from the AP algorithm, then, they are convolved with the impulse response of the filter. Then, the output data will be convolved once more with the averaging filter that has same size of the filter. The resulted values along with the previous results from previous iteration are checked for a sign change in the values to obtain bipolar data. So, if there is a variation in the sign; this gives an indication about a new crossing iteration or minimum point. So that, we must track the time values of these nodes to stop the iterative process at the minimum point as much as

possible in the dip region. The first 10 iterations in the simulation and practical examples are neglected since those represent transient data. The diagram in Figure 4-10 summarizes these steps:



Figure 4-10 Block Diagram of the Dip SC

The above described method was tested in simulation for a number of networks for the two filters  $h_2(k)$  and  $h_d(k)$  where the outputs as shown in Figure 4-11 and Figure 4-12. Figure 4-11 shows the average and maximum detected iterations by the two filters; the maximum iterations that can be achieved using the difference filter  $h_2(k)$  is equal to 52 iteration while by using our filter  $h_d(k)$  the number of iteration is less and equal to 46 iterations and this value is closed to the exact iterations for Grid 9-nodes. In addition, increasing the number of nodes will increase the number of iteration that is detected by any filter and same relation achieves for the average iteration curve. Figure 4-12 shows the average and maximum detected errors for these two filters and the exact error; the maximum error value that can be reached using the difference filter  $h_2(k)$  is equal to 5.8 ms while by using our filter  $h_d(k)$  the error value is less and equal to 3.8 ms and this value is closed to the exact error comparing to the error value of  $h_2(k)$  for Grid 9-nodes. So that, the two filters work very well and either it detects the minimum or close to it, and the best filter is the optimized filter  $h_d(k)$ .

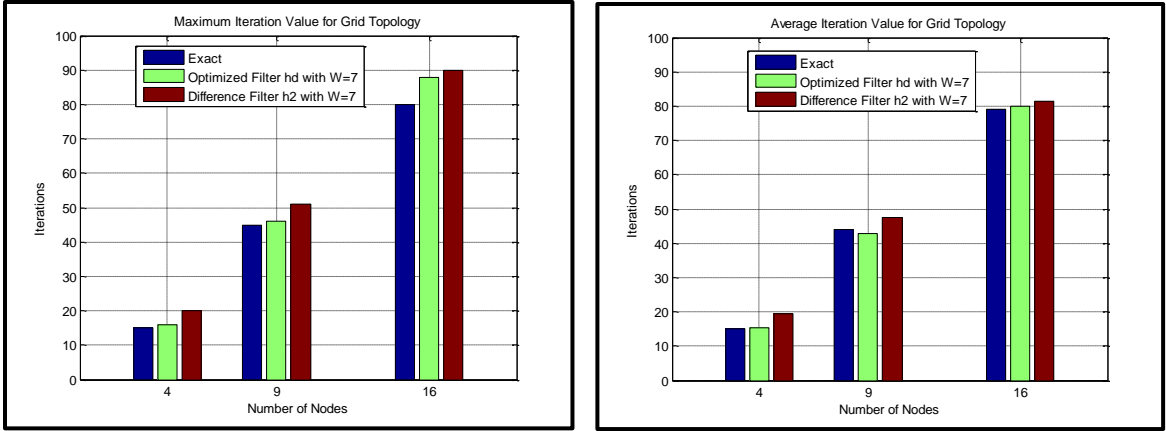


Figure 4-11 Maximum and Average detected iterations by the 2-Filters

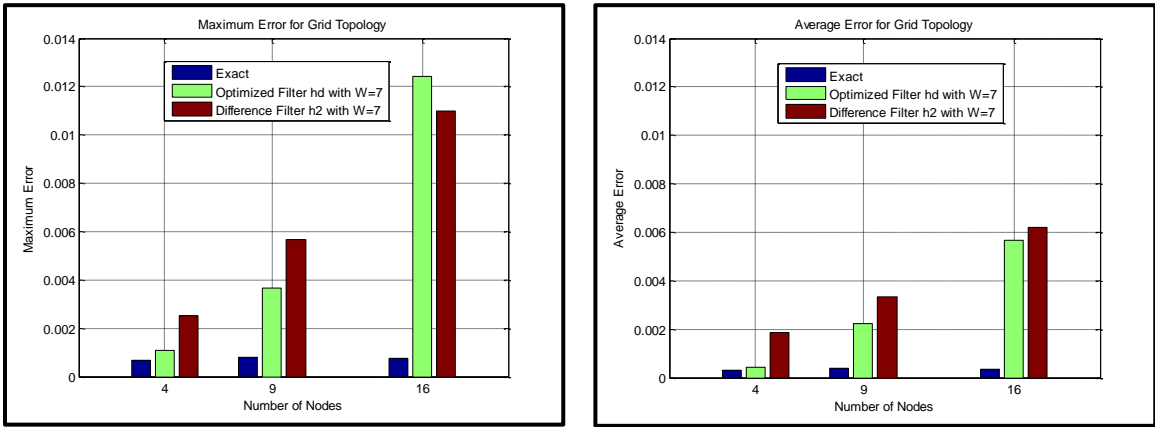


Figure 4-12 Maximum and Average error values for the 2-Filters

We deployed this stopping criterion in both the simulation and the practical networks of different sizes that were discussed in Chapter 3. It is found that this method succeeded in detecting the crossing iterations for all nodes in the Dip region around the minimum error values with very good accuracy. The next section presents the simulation and practical results for various networks.

## 4.4 Simulation Results

### 4.4.1 Simulation Results for the Small Networks

#### I. 4-nodes Grid Topology:

Table 4-3 summarizes the simulation results for the 4 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 15.3 iterations with respect to the exact 15 iterations. Also, the average minimum error of all nodes is 0.43 ms compared to 0.29 ms of the exact minimum error and the variance of all detected iterations is equal 0.333.

Table 4-3 Simulation outputs for 4-nodes Grid Topology

3+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	15	0.000671774	38	0.00299907	16	0.001094539	0.000422765
	N2	15	9.45386E-05	37	0.00199907	15	9.45386E-05	0
	N3	15	9.45386E-05	37	0.00199907	15	9.45386E-05	0
	Maximum	15	0.000671774	38	0.00299907	16	0.001094539	0.000422765
	Minimum	15	9.45386E-05	37	0.00199907	15	9.45386E-05	0
	Average	15	0.00028695	37.33333333	0.002332403	15.33333333	0.000427872	0.000140922
	Variance	0	1.11067E-07	0.333333333	3.33333E-07	0.333333333	3.33333E-07	5.95766E-08

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. Figure 4-13 shows the results for



node 1 in which the method declares a stop in three locations: one detection in the beginning (transient), near the DR, and at the SSR. If we ignore the transient region, the method is able to detect the DR and hlats the iteration and declares a minimum is reached. The same can be said for the other two nodes: node 2 in Figure 4-14 and node 3 for Figure 4-15.

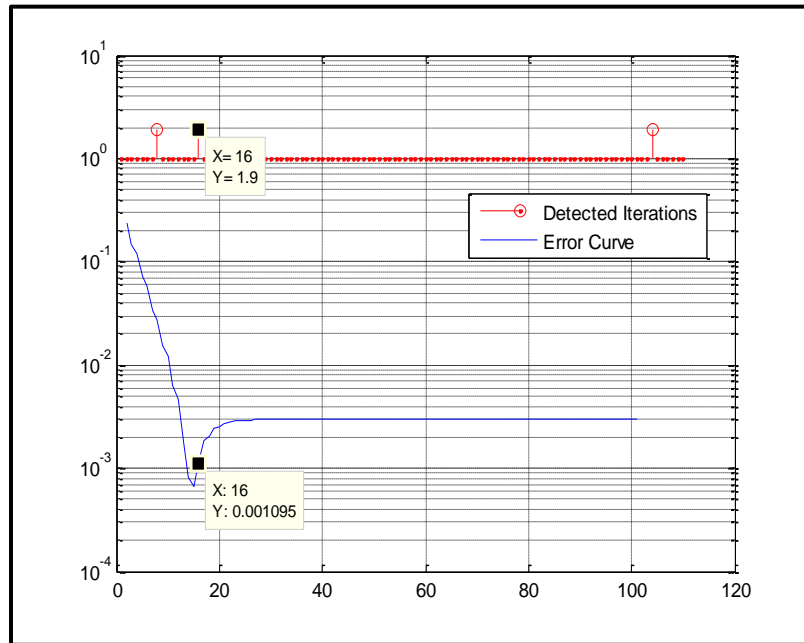


Figure 4-13 Simulation Error and the stopping locations for node 1 in 4-Grid

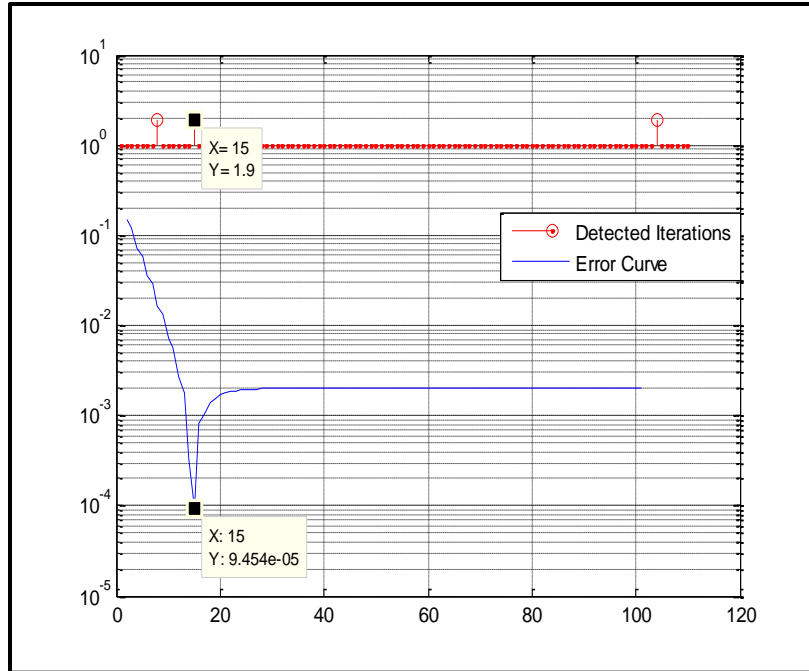


Figure 4-14 Simulation Error and the stopping locations for node 2 in 4-Grid

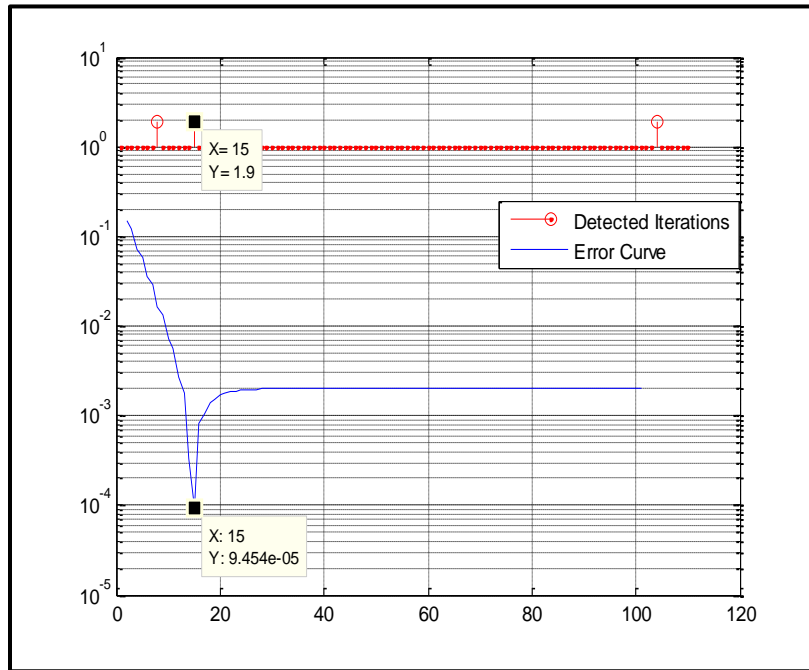


Figure 4-15 Simulation Error and the stopping locations for node 3 in 4-Grid

## II. 4-nodes Hexa Topology:

Table 4-4 summarizes the simulation results for the 4 nodes Hexa topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 18.3 iterations with respect to the exact 18 iterations. Also, the average minimum error of all nodes is 0.66 ms compared to 0.39 ms of the exact minimum error and the variance of all detected iterations is equal 0.333.

Table 4-4 Simulation outputs for 4-nodes Hexa Topology

3+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations		Error Value
	N1	18	0.000523788	43	0.003995329	19	0.001331698	0.000807911
	N2	18	0.000331698	42	0.002995329	18	0.000331698	0
	N3	18	0.000331698	42	0.002995329	18	0.000331698	0
	Maximum	18	0.000523788	43	0.003995329	19	0.001331698	0.000807911
	Minimum	18	0.000331698	42	0.002995329	18	0.000331698	0
	Average	18	0.000395728	42.33333333	0.003328663	18.33333333	0.000665032	0.000269304
	Variance	0	1.22994E-08	0.33333333	3.33333E-07	0.33333333	3.33333E-07	2.17573E-07

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. The same results were observed here when compared to the results in Grid topology.

### III. 4-nodes Random Topology:

Table 4-5 summarizes the simulation results for the 4 nodes Random topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 13 iterations with respect to the exact 13 iterations. Also, the average minimum error of all nodes is 0.04 ms which is equal to the exact minimum error and the variance of all detected iterations is equal zero.

Table 4-5 Simulation outputs for 4-nodes Random Topology

3+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	13	4.03107E-05	32	0.00199908	13	4.03107E-05	0
	N2	13	4.02398E-05	32	0.00199908	13	4.02398E-05	0
	N3	13	4.02719E-05	32	0.00199908	13	4.02719E-05	0
	Maximum	13	4.03107E-05	32	0.00199908	13	4.03107E-05	0
	Minimum	13	4.02398E-05	32	0.00199908	13	4.02398E-05	0
	Average	13	4.02741E-05	32	0.00199908	13	4.02741E-05	0
	Variance	0	1.25978E-15	0	8.95445E-34	0	1.25978E-15	0

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. The same results were observed here when compared to the results in Grid topology.

## 4.4.2 Simulation Results for the Large Networks

### I. 9-nodes Grid Topology:

Table 4-6 summarizes the simulation results for the 9 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 44 iterations with respect to the exact 42.875 iterations. Also, the average minimum error of all nodes is 2.2 ms compared to 0.38 ms of the exact minimum error and the variance of all detected iterations is equal 7.

Table 4-6 Simulation outputs for 9-nodes Grid Topology

8+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	45	5.69782E-05	165	0.016995179	46	0.002325414	0.002268436
	N2	43	0.000815579	164	0.015995179	45	0.001325414	0.000509834
	N3	45	0.000200615	162	0.013995594	43	0.001812688	0.001612074
	N4	43	0.000815579	164	0.015995179	45	0.001325414	0.000509834
	N5	45	0.000200615	162	0.013995594	43	0.001812688	0.001612074
	N6	43	0.0003926	157	0.00999558	39	0.003646339	0.003253739
	N7	45	0.000200615	162	0.013995594	43	0.001812688	0.001612074
	N8	43	0.0003926	157	0.00999558	39	0.003646339	0.003253739
	Maximum	45	0.000815579	165	0.016995179	46	0.003646339	0.003253739
	Minimum	43	5.69782E-05	157	0.00999558	39	0.001325414	0.000509834
	Average	44	0.000384398	161.625	0.013870435	42.875	0.002213373	0.001828975
	Variance	1.142857	8.29288E-08	9.410714286	6.9813E-06	6.982142857	8.8256E-07	1.12492E-06

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. Figure 4-16 shows the results for node 1 in which the method declares a stop in three locations: one detection in the beginning (transient), near the DR, and at the SSR. If we ignore the transient region, the method is able to detect the DR and halts the iteration and declares a minimum is reached. The same can be said for another two nodes: node 2 in Figure 4-17 and node 3 for Figure 4-18. The same behavior is observed for the remaining nodes.

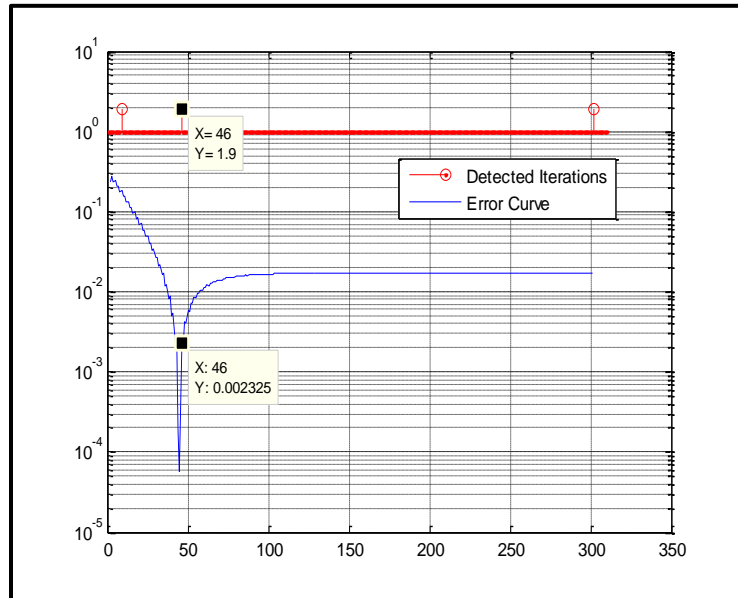


Figure 4-16 Simulation Error and the stopping locations for node 1 in 9-Grid

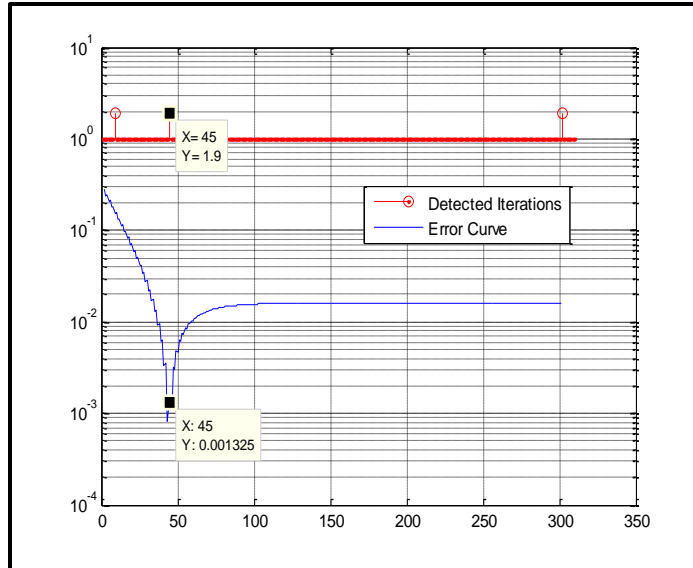


Figure 4-17 Simulation Error and the stopping locations for node 2 in 9-Grid

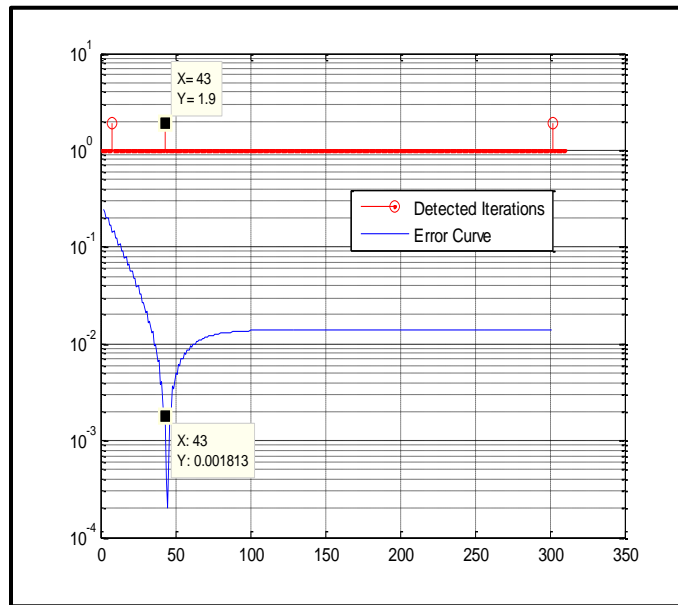


Figure 4-18 Simulation Error and the stopping locations for node 3 in 9-Grid

## II. 9-nodes Hexa Topology:

Table 4-7 summarizes the simulation results for the 9 nodes Hexa topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in

detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 37.25 iterations with respect to the exact 37.125 iterations. Also, the average minimum error of all nodes is 1.5 ms compared to 0.38 ms of the exact minimum error and the variance of all detected iterations is equal 5.

Table 4-7 Simulation outputs for 9-nodes Hexa Topology

8+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	38	0.000470093	95	0.01323586	40	0.002531802	0.002061709
	N2	37	0.000527282	94	0.012361519	39	0.001554297	0.001027015
	N3	37	0.000241893	92	0.010699382	37	0.000241893	0
	N4	37	0.000532532	94	0.0121102	39	0.001509306	0.000976774
	N5	37	0.000285639	90	0.009395969	36	0.001178707	0.000893067
	N6	37	0.000142758	88	0.007339151	33	0.003311162	0.003168404
	N7	37	0.000458648	93	0.011549251	38	0.000553999	9.5351E-05
	N8	37	0.00035947	90	0.008990248	36	0.001221922	0.000862452
	Maximum	38	0.000532532	95	0.01323586	40	0.003311162	0.003168404
	Minimum	37	0.000142758	88	0.007339151	33	0.000241893	0
	Average	37.125	0.000377289	92	0.010710197	37.25	0.001512886	0.001135597
	Variance	0.125	2.05549E-08	6	3.97419E-06	5.071428571	1.00071E-06	1.07602E-06

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. The same results were observed here when compared to the results in Grid topology.

### III. 9-nodes Random Topology:

Table 4-8 summarizes the simulation results for the 9 nodes Random topology when applying the AP algorithm along the stopping method. The table shows that some nodes



were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 36.75 iterations with respect to the exact 37 iterations. Also, the average minimum error of all nodes is 1.54 ms compared to 0.1 ms of the exact minimum error and the variance of all detected iterations is equal 4.7.

Table 4-8 Simulation outputs for 9-nodes Random Topology

8+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	37	0.000210247	94	0.013231195	39	0.001960585	0.001750338
	N2	37	7.04358E-05	93	0.012356809	39	0.001937868	0.001867432
	N3	37	0.000146435	91	0.010694614	37	0.000146435	0
	N4	37	8.44079E-05	93	0.01210558	38	0.000943595	0.000859187
	N5	37	5.85219E-05	90	0.009399316	35	0.001743444	0.001684922
	N6	37	0.000123708	87	0.007334509	33	0.002933161	0.002809453
	N7	37	3.27568E-05	93	0.011552438	38	0.000944243	0.000911486
	N8	37	2.71062E-05	90	0.008993481	35	0.001767299	0.001740193
	Maximum	37	0.000210247	94	0.013231195	39	0.002933161	0.002809453
	Minimum	37	2.71062E-05	87	0.007334509	33	0.000146435	0
	Average	37	9.42024E-05	91.375	0.010708493	36.75	0.001547079	0.001452876
	Variance	0	3.89666E-09	5.410714	3.96918E-06	4.785714	7.17341E-07	7.13351E-07

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. The same results were observed here when compared to the results in Grid topology.

#### IV. 16-nodes Grid Topology:

Table 4-9 summarizes the simulation results for the 16 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes

were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 79.867 iterations with respect to the exact 79.2 iterations. Also, the average minimum error of all nodes is 5.6 ms compared to 0.35 ms of the exact minimum error and the variance of all detected iterations is equal 69.

Table 4-9 Simulation outputs for 16-nodes Grid Topology

15+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value
N1	80	0.000711993	357	0.043543766	88	0.008921185	0.008209192
N2	80	0.000145546	356	0.042543766	87	0.007921185	0.007775639
N3	79	0.000633557	354	0.04018738	85	0.005387451	0.004753894
N4	79	0.000265769	351	0.037758506	82	0.002567965	0.002302196
N5	80	0.000145546	356	0.042543766	87	0.007921185	0.007775639
N6	79	0.00075918	354	0.040901112	86	0.006881178	0.006121999
N7	79	0.000277854	351	0.037258878	82	0.002548629	0.002270775
N8	79	7.58253E-05	346	0.033329904	77	0.001901044	0.001825219
N9	79	0.000633557	354	0.04018738	85	0.005387451	0.004753894
N10	79	0.000277854	351	0.037258878	82	0.002548629	0.002270775
N11	79	7.74176E-05	343	0.030615441	74	0.003698408	0.00362099
N12	79	0.000449196	330	0.02197224	62	0.012420188	0.011970992
N13	79	0.000265769	351	0.037758506	82	0.002567965	0.002302196
N14	79	7.58255E-05	346	0.033329904	77	0.001901044	0.001825219
N15	79	0.000449196	330	0.02197224	62	0.012420194	0.011970998
Maximum	80	0.00075918	357	0.043543766	88	0.012420194	0.011970998
Minimum	79	7.58253E-05	330	0.02197224	62	0.001901044	0.001825219
Average	79.2	0.000349605	348.6666667	0.036077444	79.86666667	0.005666247	0.005316641
Variance	0.171429	5.79858E-08	73.38095238	4.62651E-05	68.98095238	1.31779E-05	1.2449E-05

The stopping method was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. Figure 4-19 shows the results for node 1 in which the method declares a stop in three locations: one detection in the beginning (transient), near the DR, and at the SSR. If we ignore the transient region, the method is able to detect the DR and halts the iteration and declares a minimum is reached. The same can be said for another two nodes: node 2 in Figure 4-20 and node 3 for Figure 4-18. The same behavior is observed for the remaining nodes.

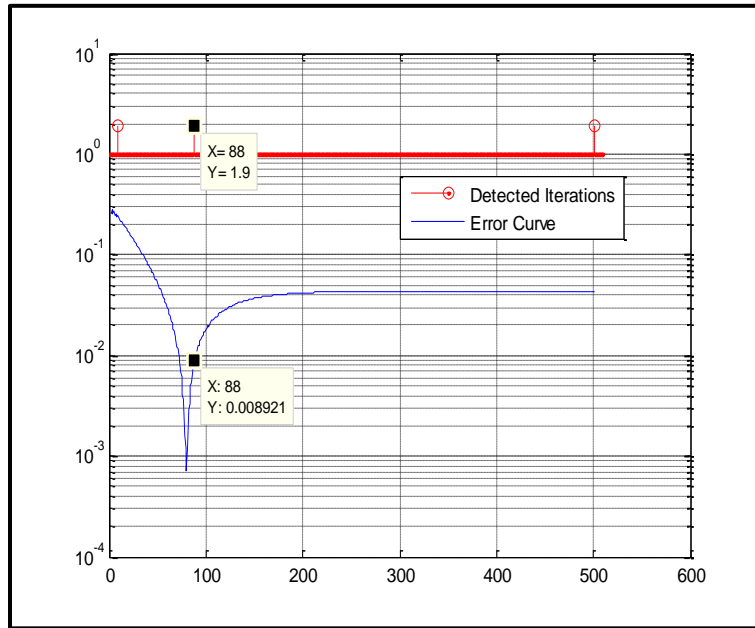


Figure 4-19 Simulation Error and the stopping locations for node 1 in 16-Grid

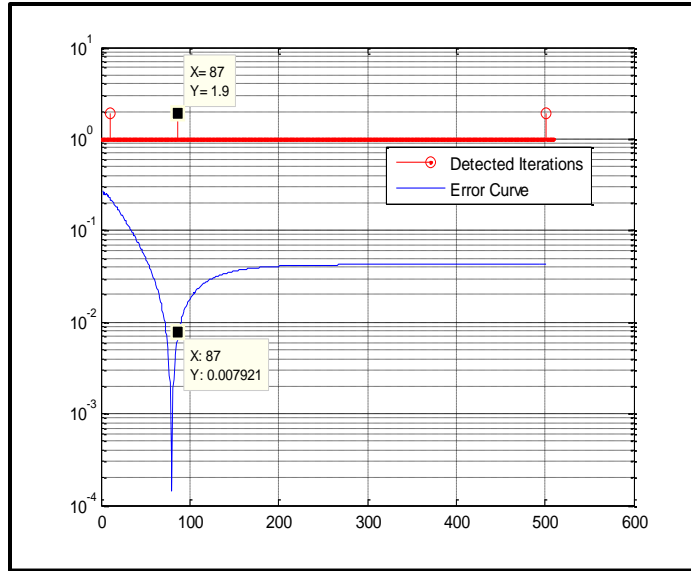


Figure 4-20 Simulation Error and the stopping locations for node 2 in 16-Grid

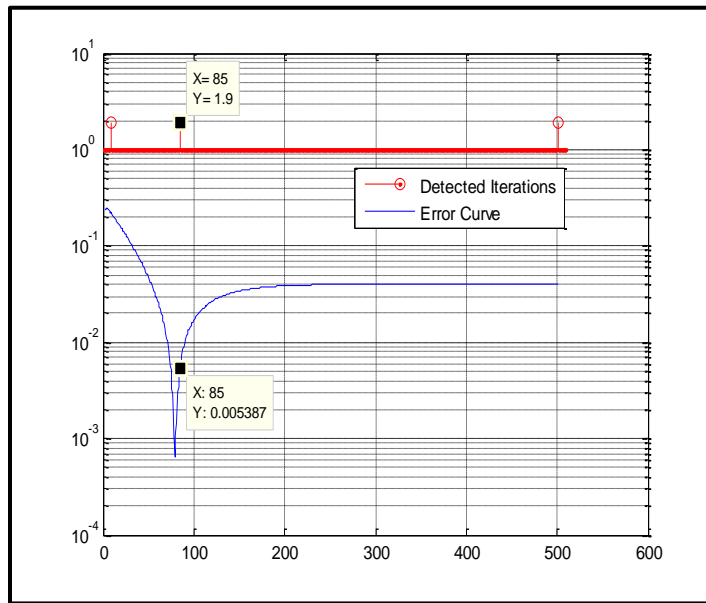


Figure 4-21 Simulation Error and the stopping locations for node 3 in 16-Grid

## V. 16-nodes Hexa Topology:

Table 4-10 summarizes the simulation results for the 16 nodes Hexa topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in

detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 88.267 iterations with respect to the exact 86.6 iterations. Also, the average minimum error of all nodes is 4.5 ms compared to 0.26 ms of the exact minimum error and the variance of all detected iterations is equal 54.2.

Table 4-10 Simulation outputs for 16-nodes Hexa Topology

15+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	89	0.000359786	239	0.049468008	94	0.005649195	0.005289409
	N2	89	0.000472502	238	0.04846966	93	0.004656662	0.00418416
	N3	88	0.000313058	235	0.045905541	90	0.001749268	0.00143621
	N4	88	4.88986E-05	233	0.043409819	88	4.88986E-05	0
	N5	89	0.000456458	238	0.048466356	93	0.004641728	0.004185271
	N6	88	0.000379195	236	0.046043134	90	0.001690352	0.001311156
	N7	88	2.22991E-05	231	0.041796743	86	0.001979288	0.001956989
	N8	88	0.00013476	228	0.03945525	83	0.004635745	0.004500984
	N9	89	0.000529627	237	0.047632921	92	0.00364551	0.003115883
	N10	88	0.000381809	235	0.045722651	90	0.001675424	0.001293615
	N11	88	9.87801E-07	229	0.039763147	84	0.003813714	0.003812726
	N12	88	0.000154575	217	0.030227179	72	0.013253929	0.013099355
	N13	88	0.000380926	235	0.045712115	90	0.001675798	0.001294872
	N14	88	8.68129E-05	230	0.040793424	85	0.002991204	0.002904391
	N15	88	0.000122524	213	0.027945782	69	0.015221346	0.015098822
	Maximum	89	0.000529627	239	0.049468008	94	0.015221346	0.015098822
	Minimum	88	9.87801E-07	213	0.027945782	69	4.88986E-05	0
	Average	88.26667	0.000256281	231.6	0.042720782	86.6	0.004488537	0.004232256
	Variance	0.209524	3.26093E-08	57.4	4.09786E-05	54.25714286	1.81442E-05	1.84136E-05

## VI. 16-nodes Random Topology:

Table 4-11 summarizes the simulation results for the 16 nodes Random topology when applying the AP algorithm along the stopping method. The table shows that some nodes

were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 90 iterations with respect to the exact 90 iterations. Also, the average minimum error of all nodes is 2.5 ms compared to 0.1 ms of the exact minimum error and the variance of all detected iterations is equal 15.1.

Table 4-11 Simulation outputs for 16-nodes Random Topology

15+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	90	0.000206789	244	0.04970001	94	0.003938713	0.003731924
	N2	90	0.000127355	243	0.048666345	93	0.0029465	0.002819145
	N3	90	6.06407E-05	240	0.045624842	90	6.06407E-05	0
	N4	90	0.000121399	240	0.04572843	90	0.000121399	0
	N5	90	0.000148503	243	0.048733675	93	0.002930926	0.002782423
	N6	90	3.0945E-05	241	0.046611968	91	0.000970757	0.000939812
	N7	90	0.000101025	238	0.044360777	89	0.000872278	0.000771253
	N8	90	0.000143706	238	0.044156018	88	0.001812313	0.001668608
	N9	90	0.000135718	243	0.048407629	93	0.002922366	0.002786648
	N10	90	9.23675E-05	241	0.047197819	92	0.001917977	0.00182561
	N11	90	1.07263E-05	240	0.045822958	90	1.07263E-05	0
	N12	90	8.36796E-05	230	0.036939189	80	0.008909279	0.0088256
	N13	90	0.000144487	243	0.048461796	93	0.002917562	0.002773075
	N14	90	7.95591E-05	241	0.046736317	91	0.000925858	0.000846299
	N15	90	1.25872E-05	232	0.038780077	83	0.006385163	0.006372576
	Maximum	90	0.000206789	244	0.04970001	94	0.008909279	0.0088256
	Minimum	90	1.07263E-05	230	0.036939189	80	1.07263E-05	0
	Average	90	9.99658E-05	239.8	0.045728523	90	0.002509497	0.002409531
	Variance	0	3.05713E-09	16.17143	1.29953E-05	15.14286	6.07586E-06	6.05691E-06

## **4.5 Practical Results**

In this part, we generate real data by applying AP algorithm on real sensor nodes without stopping criterion. Then, stopping criterion was tested on these real data using MATLAB to check if the SC method can detect the minimum error in the DR for each node. The grid topology of different sizes will be presented next. It is worth mentioning here that other topologies exhibit similar behaviour.

### **4.5.1 Practical Results for the Small Networks**

#### **I. 4-nodes Grid Topology:**

Table 4-12 summarizes the practical results for the 4 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes were able to find the exact minimum value and some other nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 17.3 iterations with respect to the exact 15.3 iterations. Also, the average minimum error of all nodes is 1.1 ms compared to 0.2 ms of the exact minimum error and the variance of all detected iterations is equal 0.333.

Table 4-12 Practical outputs for 4-nodes Grid Topology

3+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value
N1	18	0.000463281	47	0.002999968	18	0.000463281	0
N2	14	6.48437E-05	46	0.001999968	17	0.001463281	0.001398438
N3	14	6.48437E-05	46	0.001999968	17	0.001463281	0.001398438
Maximum	18	0.000463281	47	0.002999968	18	0.001463281	0.001398438
Minimum	14	6.48437E-05	46	0.001999968	17	0.000463281	0
Average	15.33333	0.000197656	46.33333333	0.002333302	17.33333333	0.001129948	0.000932292
Variance	5.333333	5.29175E-08	0.33333333	3.33333E-07	0.33333333	3.33333E-07	6.51876E-07

The stopping method with length of filter equal 7 was applied for each node separately for all regions in the curve to test where and how many times it can declare a halt. Figure 4-22 shows the results for node 1 in which the method declares a stop in three locations: one detection in the beginning (transient), near the DR, and at the SSR. If we ignore the transient region, the method is able to detect the DR and halts the iteration and declares a minimum is reached. The same can be said for the other two nodes: node 2 in Figure 4-23 and node 3 for Figure 4-24.



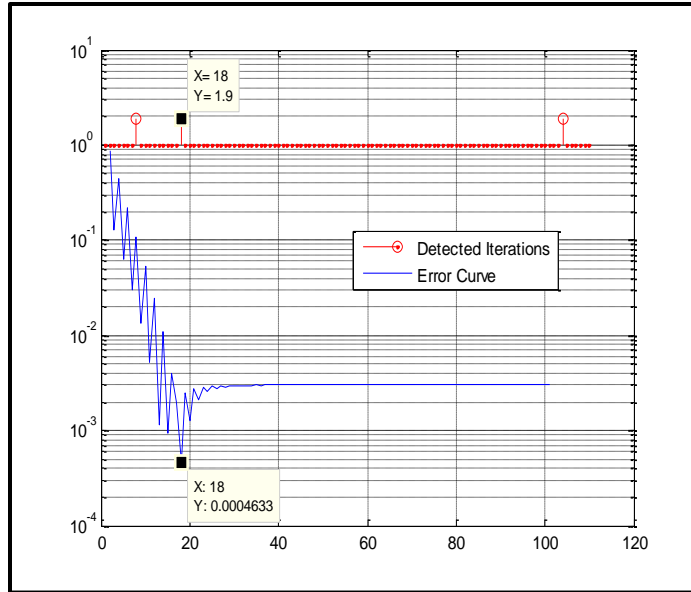


Figure 4-22 Practical error curve and the stopping locations for node 1 in 4-Grid

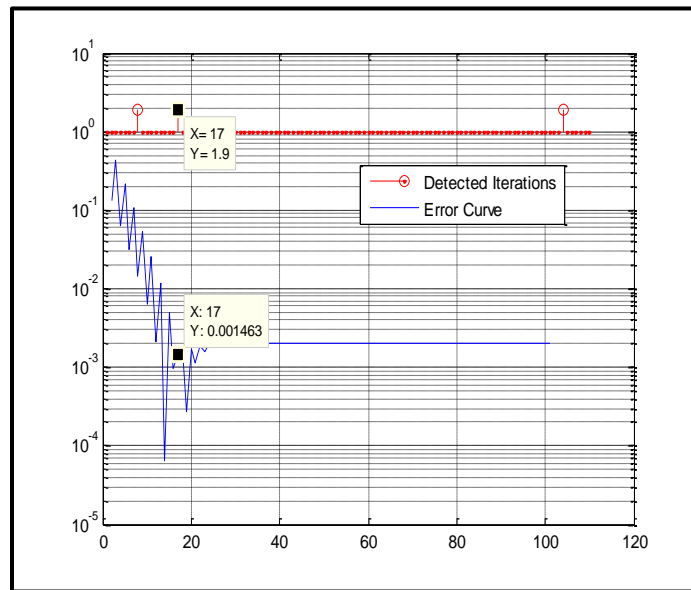


Figure 4-23 Practical error curve and the stopping locations for node 2 in 4-Grid

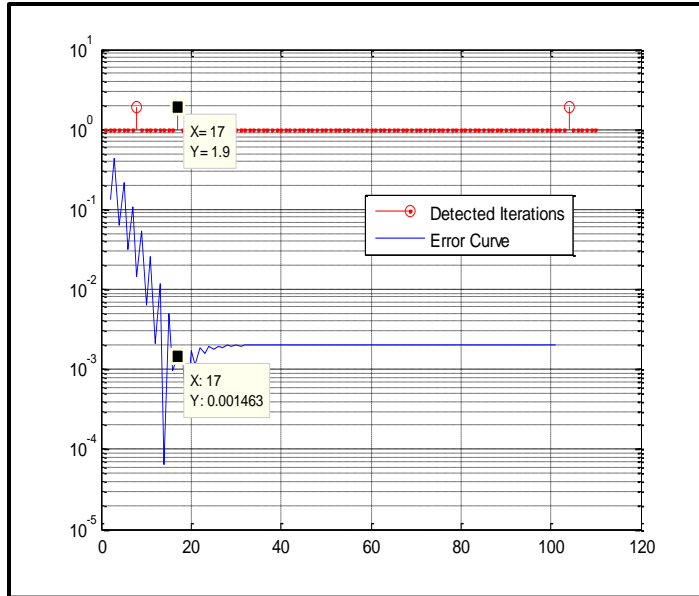


Figure 4-24 Practical error curve and the stopping locations for node 3 in 4-Grid

## 4.5.2 Practical Results for the Large Networks

### I. 9-nodes Grid Topology:

Table 4-13 summarizes the practical results for the 9 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 52.625 iterations with respect to the exact 53 iterations. Also, the average minimum error of all nodes is 4.2 ms compared to 0.44 ms of the exact minimum error and the variance of all detected iterations is equal 9.4.

Table 4-13 Practical outputs for 9-nodes Grid Topology

8+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	55	0.000729527	195	0.016998716	56	0.005181115	0.004451588
	N2	51	0.000480831	194	0.015998716	55	0.004181115	0.003700284
	N3	55	0.000343489	191	0.013998636	53	0.002436176	0.002092687
	N4	51	0.000480831	194	0.015998716	55	0.004181115	0.003700284
	N5	55	0.000343489	191	0.013998636	53	0.002436176	0.002092687
	N6	51	0.000408626	187	0.009999088	48	0.00648722	0.006078594
	N7	55	0.000343489	191	0.013998636	53	0.002436176	0.002092687
	N8	51	0.000408626	187	0.009999088	48	0.00648722	0.006078594
	Maximum	55	0.000729527	195	0.016998716	56	0.00648722	0.006078594
	Minimum	51	0.000343489	187	0.009999088	48	0.002436176	0.002092687
	Average	53	0.000442363	191.25	0.013873779	52.625	0.004228289	0.003785925
	Variance	4.571429	1.67182E-08	9.357142857	6.98131E-06	9.410714286	2.9647E-06	2.79594E-06

**II. 16-nodes Grid Topology:**

Table 4-14 summarizes the practical results for the 16 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 94.8 iterations with respect to the exact 95.67 iterations. Also, the average minimum error of all nodes is 5.1 ms compared to 0.23 ms of the exact minimum error and the variance of all detected iterations is equal 68.1.

Table 4-14 Practical outputs for 16-nodes Grid Topology

15+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value
N1	94	0.000328504	447	0.04356728	103	0.004796991	0.004468487
N2	98	0.000552229	446	0.04256728	102	0.003796991	0.003244763
N3	94	0.00013527	444	0.040210629	100	0.006042952	0.005907682
N4	98	3.44644E-05	441	0.037782017	97	0.003230824	0.003196359
N5	98	0.000552229	446	0.04256728	102	0.003796991	0.003244763
N6	94	2.5456E-05	444	0.04092484	100	0.006054611	0.006029155
N7	98	4.97064E-05	441	0.037282067	97	0.003202441	0.003152734
N8	94	0.000553441	436	0.033353441	92	0.001237424	0.000683983
N9	94	0.00013527	444	0.040210629	100	0.006042952	0.005907682
N10	98	4.97064E-05	441	0.037282067	97	0.003202441	0.003152734
N11	97	0.000425289	433	0.030638745	89	0.007785826	0.007360537
N12	93	7.39918E-05	420	0.021995837	77	0.011772343	0.011698351
N13	98	3.44644E-05	441	0.037782017	97	0.003230824	0.003196359
N14	94	0.000553441	436	0.033353441	92	0.001237424	0.000683983
N15	93	7.39918E-05	420	0.021995837	77	0.011772344	0.011698352
Maximum	98	0.000553441	447	0.04356728	103	0.011772344	0.011698352
Minimum	93	2.5456E-05	420	0.021995837	77	0.001237424	0.000683983
Average	95.66667	0.000238497	438.6666667	0.036100894	94.8	0.005146892	0.004908395
Variance	4.666667	5.09708E-08	73.38095238	4.62647E-05	68.17142857	1.0459E-05	1.10657E-05

## 4.6 Hardware Platform and Implementation Details for Real-world Experiments

The AP algorithm and our designed SC provided promising results in simulation and when they are applied to real data. Therefore, the ultimate test will be to test these two algorithms for synchronization in the real network. We performed our experiments with single hop wireless sensor networks using one type of commercially available sensor nodes called Micaz made by Memsic Company supported with multiple of oscillators 7.37 MHz and 32 KHz, 8-bit Atmel Atmega128L microcontroller, 4kB RAM, 128kB

program flash and Chipcon CC2420 radio chip has data rate equal to 250 kbps; CC2420 transceiver on Micaz nodes has the capability to timestamp synchronization packets at the MAC layer with the timer used for timing measurements. We used the packet level time synchronization interfaces provided by TinyOS to timestamp synchronization messages at the MAC layer. Micaz nodes run on the open source TinyOS operating system and operate over the 2.4 GHz IEEE 802.15.4 protocol, compliant to Direct Sequence Spread Spectrum (DSSS) radio with Orthogonal-QPSK modulation technique. We used a total of 4, 9, and 16 Micaz nodes and another node called sink or Base station node. This sink node was connected through a MIB520 gateway supported with USB port and directly connected to Laptop that is using an open source OS Ubuntu. TinyOS 2.1.2 package was installed on that Laptop. The sink node was used to capture the transmitted packets between the sensor nodes over the USB interface. The oscillators of the Micaz nodes represent the clock source for the timer that uses for the timing measurements. Micaz have three timers: TMilli, T32khz, and TMicro, so the accuracy of these sensor nodes will depend on the timer. In this implementation we used the TMilli timer and the accuracy will be within milliseconds. Besides to the implementation of our protocol, we implemented two other protocols for reference which are RFTSP and EGTSP and the results are presented in section 4.7.4. The whole flow charts of this protocol with the two stopping criteria can be found in the APPENDIX. In this part, we implemented the whole system using Micaz nodes that were discussed in Chapter 1 using NesC language. The specifications used in the implementation are summarized in Table 4-15. We concentrate also here in the grid topology of various sizes and similar results were observed for other topologies.

Table 4-15 Specification of the implementation part

Parameter	Specification
Topology	Grid
Nodes	4-nodes, 9-nodes, and 16-nodes.
Packet Size	32bit(Time)+32bit(Error)+16bit(ID)+16bit(Iteration)=64bits
C Value	1.8 (4-Nodes), 1.035 (9-Nodes), and 1.002 (16-Nodes)
Stopping Condition	Give the (Iteration, Time, and Error) Values + Stopping the Transmission
Timer	TMilli

#### 4.6.1 4-Nodes Grid Topology

Figure 4-25 shows how the 4-nodes grid topology is distributed in real time implementation before taking the output data.

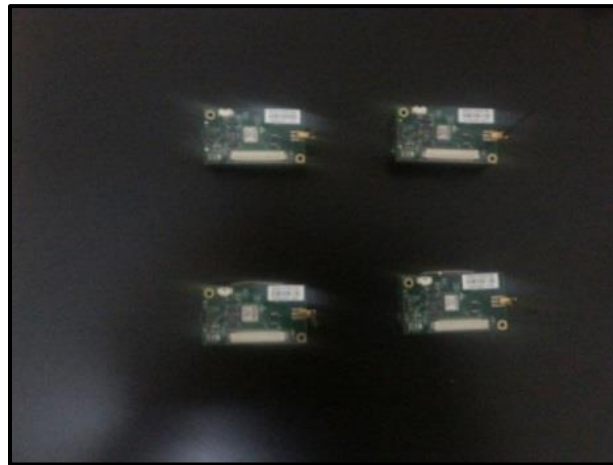


Figure 4-25 Real Time implementation for 4-nodes

The time values and the error values are plotted for each node as shown in Figure 4-26 and Figure 4-27, respectively.

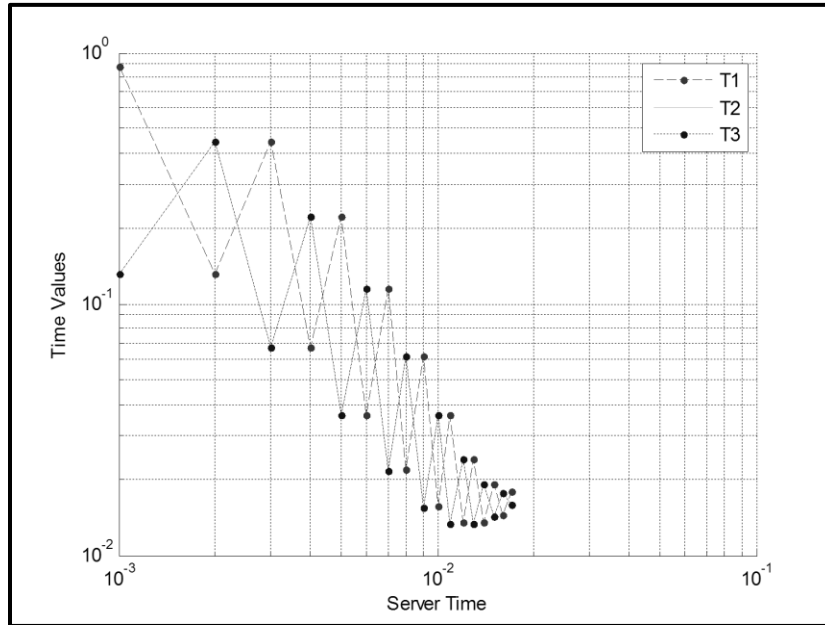


Figure 4-26 Practical Time Values for each node in 4-Grid

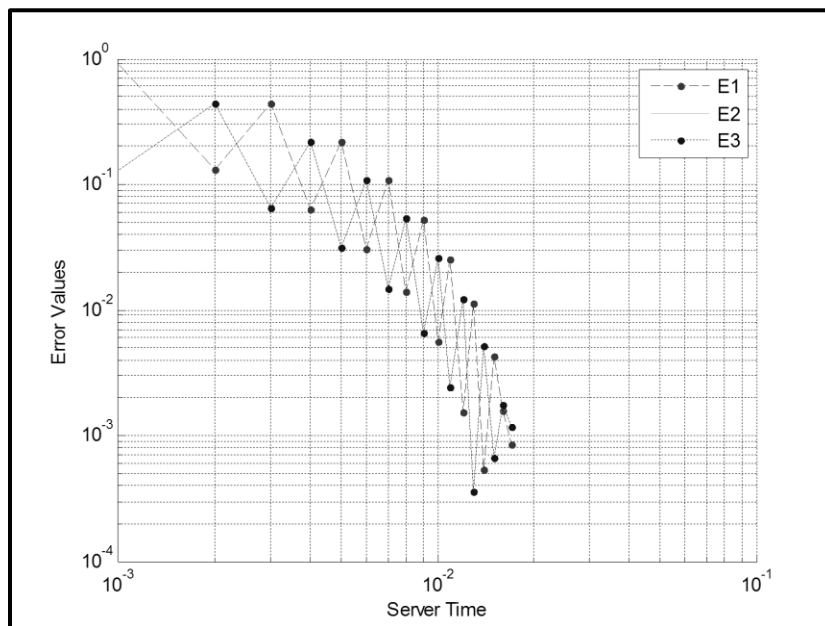


Figure 4-27 Practical Error Values for each node in 4-Grid

The two figures show that the protocol works. Each node is able to detect the minimum error using our protocol and stops the iterative process inside the dip region. Table 4-16 shows the values in the above two figures. Also, the Table shows the exact minimum values and the steady state values as presented in section 4.2 for comparison.

Table 4-16 Practical Iteration and Error Values for 4-nodes

3+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations		Error Value
	N1	15	0.000538281	47	0.002599969	18	0.000862109	0.000323828
	N2	19	3.10547E-05	46	0.001699969	18	0.00118457	0.001153516
	N3	19	3.10547E-05	46	0.001699969	18	0.00118457	0.001153516
	Maximum	19	0.000538281	47	0.002599969	18	0.00118457	0.001153516
	Minimum	15	3.10547E-05	46	0.001699969	18	0.000862109	0.000323828
	Average	17.66667	0.00020013	46.33333333	0.001999969	18	0.001077083	0.000876953
	Variance	5.333333	8.57596E-08	0.33333333	0.00000027	0	3.46604E-08	2.2946E-07

Table 4-16 summarizes the real results for the 4 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 18 iterations with respect to the exact 17.7 iterations. Also, the average minimum error of all nodes is 1ms compared to 0.2 ms of the exact minimum error and the variance of all detected iterations is equal 0.

#### 4.6.2 9-Nodes Grid Topology

Figure 4-28 shows how the 9-nodes grid topology is distributed in real time implementation before taking the output data.



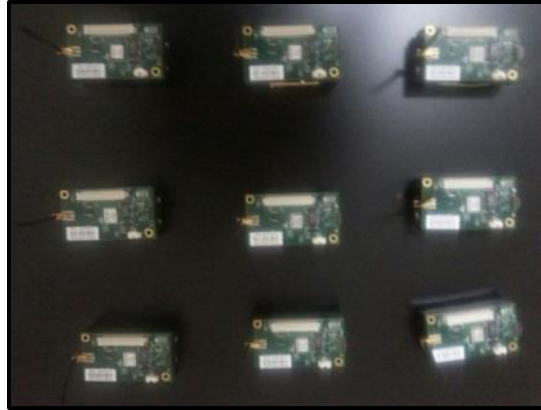


Figure 4-28 Real Time implementation of 9-nodes

The time values and the error values are plotted for each node as shown in Figure 4-29 and Figure 4-30, respectively.

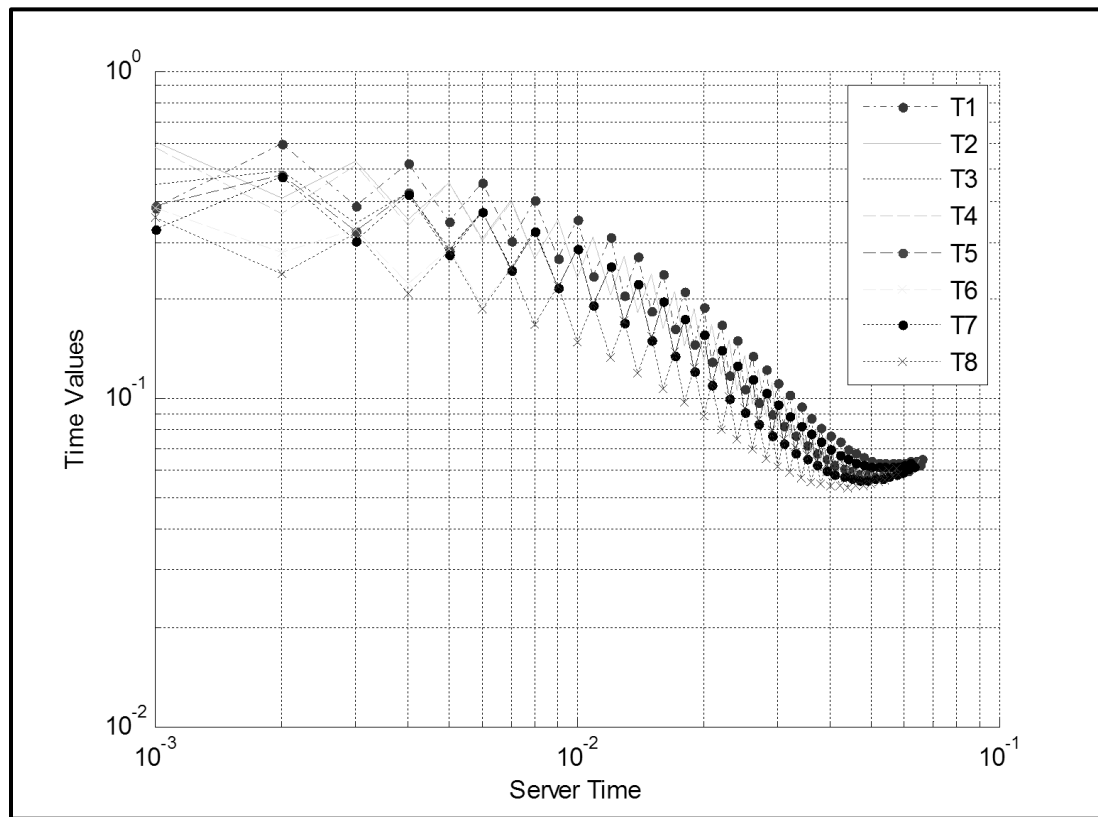


Figure 4-29 Practical Time Values for each node in 9-Grid

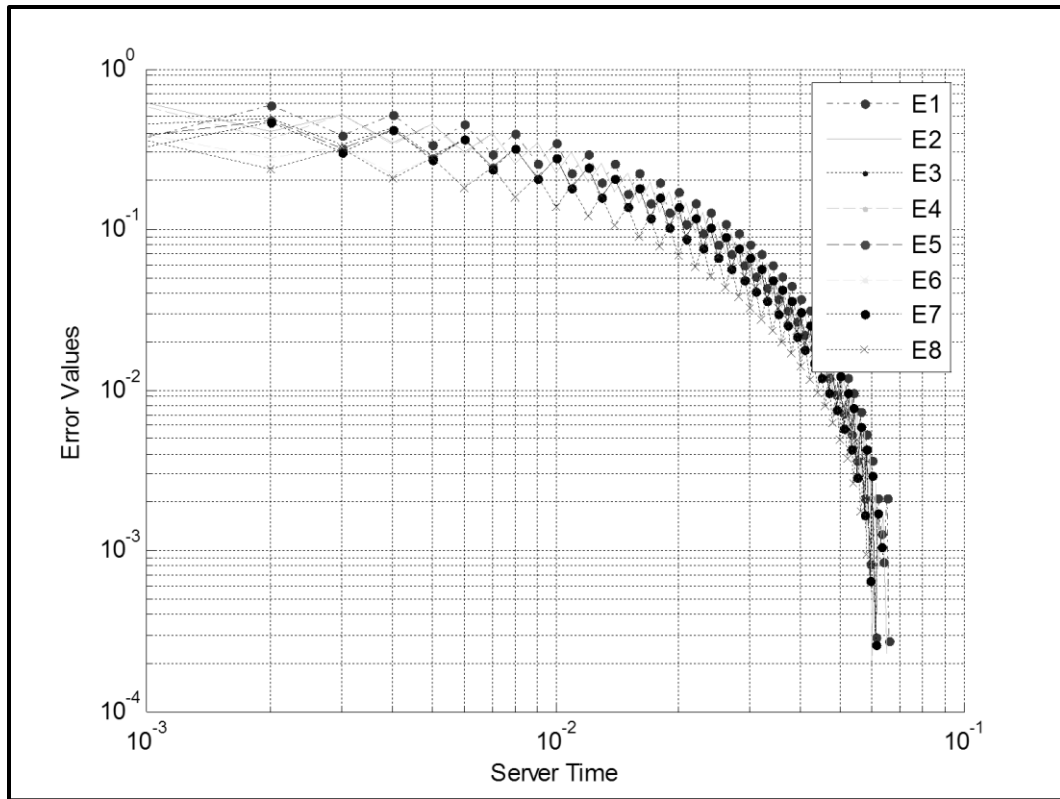


Figure 4-30 Practical Error Values for each node in 9-Grid

The two figures show that the protocol works. Each node is able to detect the minimum error using our protocol and stops the iterative process. Table 4-17 shows the values in the above two figures. Also, the Table shows the exact minimum values and the steady state values as presented in section 4.2 for comparison.

Table 4-17 Iteration and Error Values for 9-nodes

8+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error	
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value	Error Value
	N1	67	0.000271674	195	0.007998733	67	0.000271674	0
	N2	61	0.000210105	194	0.007498733	66	0.000228326	1.82214E-05
	N3	67	0.000247653	191	0.006498654	64	0.001056578	0.000808926
	N4	61	0.000210105	194	0.007498733	66	0.000228326	1.82214E-05
	N5	67	0.000247653	191	0.006498654	64	0.001056578	0.000808926
	N6	61	0.000265107	187	0.004499105	59	0.000960327	0.00069522
	N7	67	0.000247653	191	0.006498654	64	0.001056578	0.000808926
	N8	61	0.000265107	187	0.004499105	59	0.000960327	0.00069522
	Maximum	67	0.000271674	195	0.007998733	67	0.001056578	0.000808926
	Minimum	61	0.000210105	187	0.004499105	59	0.000228326	0
	Average	64	0.000245632	191.25	0.006436296	63.625	0.000727339	0.000481707
	Variance	10.28571	5.67611E-10	9.357142857	1.74512E-06	9.410714286	1.62774E-07	1.53439E-07

Table 4-17 summarizes the real results for the 9 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 63.625 iterations with respect to the exact 64 iterations. Also, the average minimum error of all nodes is 0.7 ms compared to 0.25 ms of the exact minimum error and the variance of all detected iterations is equal 9.4.

### 4.6.3 16-Nodes Grid Topology

Figure 4-31 shows how the 16-nodes grid topology is distributed in real time implementation before taking the output data.



Figure 4-31 Real Time implementation of 16-nodes

The time values and the error values are plotted for each node as shown in Figure 4-32 and Figure 4-33, respectively.

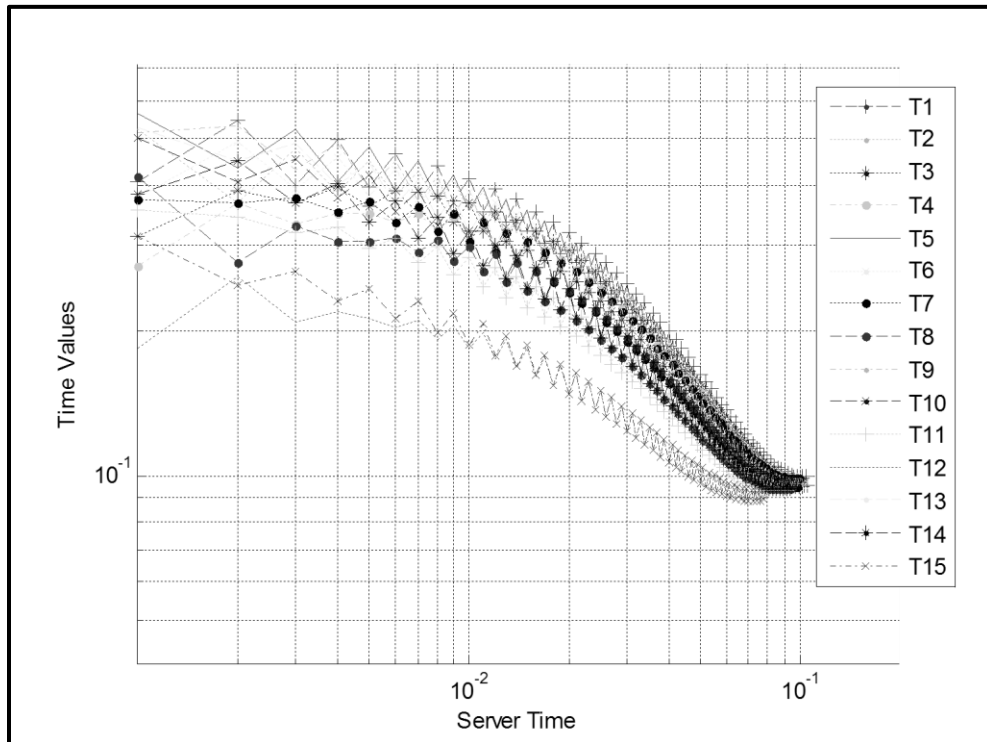


Figure 4-32 Practical Time Values for each node in 16-Grid

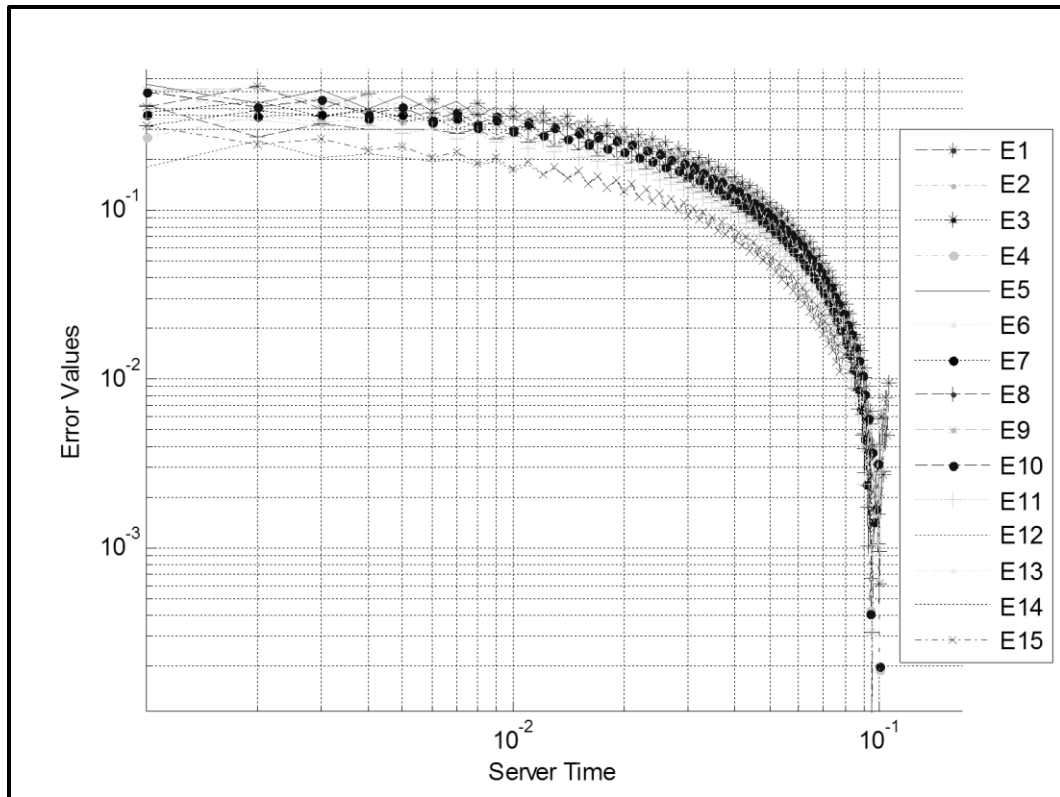


Figure 4-33 Practical Error Values for each node in 16-Grid

The two figures show that the protocol works. Each node is able to detect the minimum error using our protocol and stops the iterative process. Table 4-18 shows the values in the above two figures. Also, the table shows the exact minimum values and the steady state values as presented in section 4.2 for comparison.

Table 4-18 Iteration and Error Values for 16-nodes

15+1	Exact in Dip Region		Stopping in SSR		Stopping in Dip		Difference between the errors using the stopping method and the exact error
	Nodes	Iterations	Error Value	Iterations	Error Value	Iterations	Error Value
N1	96	0.00010285	447	0.041338725	106	0.00957014	0.009467291
N2	100	0.00033942	446	0.040388725	105	0.00862014	0.00828072
N3	96	0.000315173	444	0.038149931	103	0.002891579	0.002576406
N4	100	0.000188637	441	0.035842747	100	0.000188637	0
N5	100	0.00033942	446	0.040388725	105	0.00862014	0.00828072
N6	96	0.000215716	444	0.038828428	104	0.007612213	0.007396497
N7	100	0.000200307	441	0.035367798	100	0.000200307	0
N8	99	0.000459066	436	0.0316356	96	0.000669474	0.000210408
N9	96	0.000315173	444	0.038149931	103	0.002891579	0.002576406
N10	100	0.000200307	441	0.035367798	100	0.000200307	0
N11	99	0.000284579	433	0.029056619	92	0.002428683	0.002144104
N12	95	1.00546E-05	420	0.020845853	80	0.014000267	0.013990213
N13	100	0.000188637	441	0.035842747	100	0.000188637	0
N14	99	0.000459066	436	0.0316356	96	0.000669474	0.000210408
N15	95	1.00546E-05	420	0.020845853	80	0.014000268	0.013990213
Maximum	100	0.000459066	447	0.041338725	106	0.014000268	0.013990213
Minimum	95	1.00546E-05	420	0.020845853	80	0.000188637	0
Average	98.06667	0.000241897	438.6666667	0.034245672	98	0.004850123	0.004608226
Variance	4.352381	1.87544E-08	73.38095238	4.17539E-05	68.28571429	2.5784E-05	2.65968E-05

Table 4-18 summarizes the real results for the 16 nodes Grid topology when applying the AP algorithm along the stopping method. The table shows that some nodes were very close in detecting the minimum value. Also, compared to SSR, there is a large saving in the number of iteration to halt the algorithm which means as stated before more saving and faster reaching to the minimum value. On average, it is noted that the stopping method for all nodes detects the minimum in 98 iterations with respect to the exact 98.1 iterations. Also, the average minimum error of all nodes is 4.8 ms compared to 0.25 ms of the exact minimum error and the variance of all detected iterations is equal 68.2.

## 4.7 Comparisons and Tests of the protocol Under Various Scenarios

In this section, the synchronization protocol is tested extensively under various scenarios such as different sizes, different topologies, different C values and different initial parameters. Also, comparisons of the results are presented for simulated and practical networks.

### 4.7.1 Summarized Simulation Results for Different Sizes (4, 9, and 16)

This part shows the effects of increasing the number of nodes on the detected minimum values by calculating the error values between the exact minimum and detected minimum. This effect is presented using bar plots for different sizes with different topologies. Here, the error values are represented by the (maximum, average, and minimum) deviation error between the exact and detected time for all nodes. The following simulation results for the Grid, Hexa, and Random Topologies with different sizes (4, 9, and 16 nodes) are presented.

Figure 4-34 shows the deviation error for the Grid topology with various sizes. The figure shows that the deviation error increases as the network size increases. The error deviates between  $(0 - 0.3 \times 10^{-3})$  for the 4-nodes network, between  $(0.3 \times 10^{-3} - 0.3 \times 10^{-2})$  for the 9-nodes network, and between  $(0.1 \times 10^{-2} - 0.1 \times 10^{-1})$  for the 16-nodes network. This behaviour is expected as the network size increases.

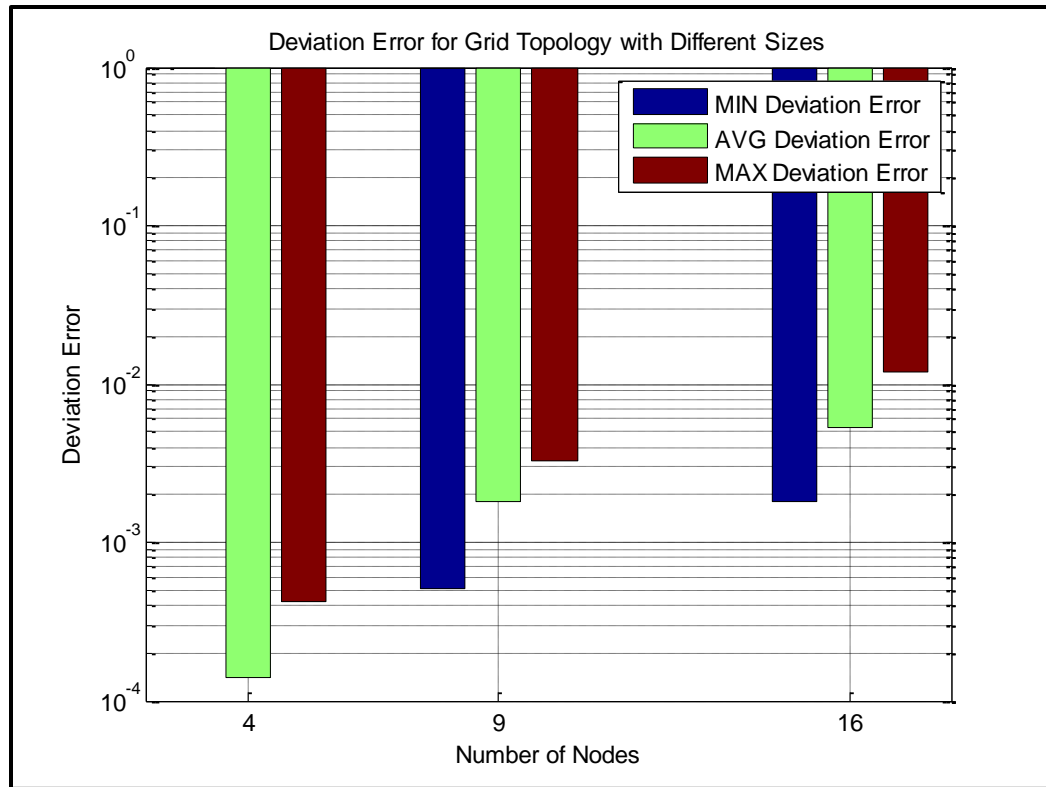


Figure 4-34 Deviation error for Grid Topology

Figure 4-35 shows the deviation error for the Hexa topology with various sizes. The figure shows that the deviation error increases as the network size increases. The error deviates between  $(0 - 0.8 \times 10^{-3})$  for the 4-nodes network, between  $(0 - 0.3 \times 10^{-2})$  for the 9-nodes network, and between  $(0 - 0.1 \times 10^{-1})$  for the 16-nodes network. This behaviour is expected as the network size increases.



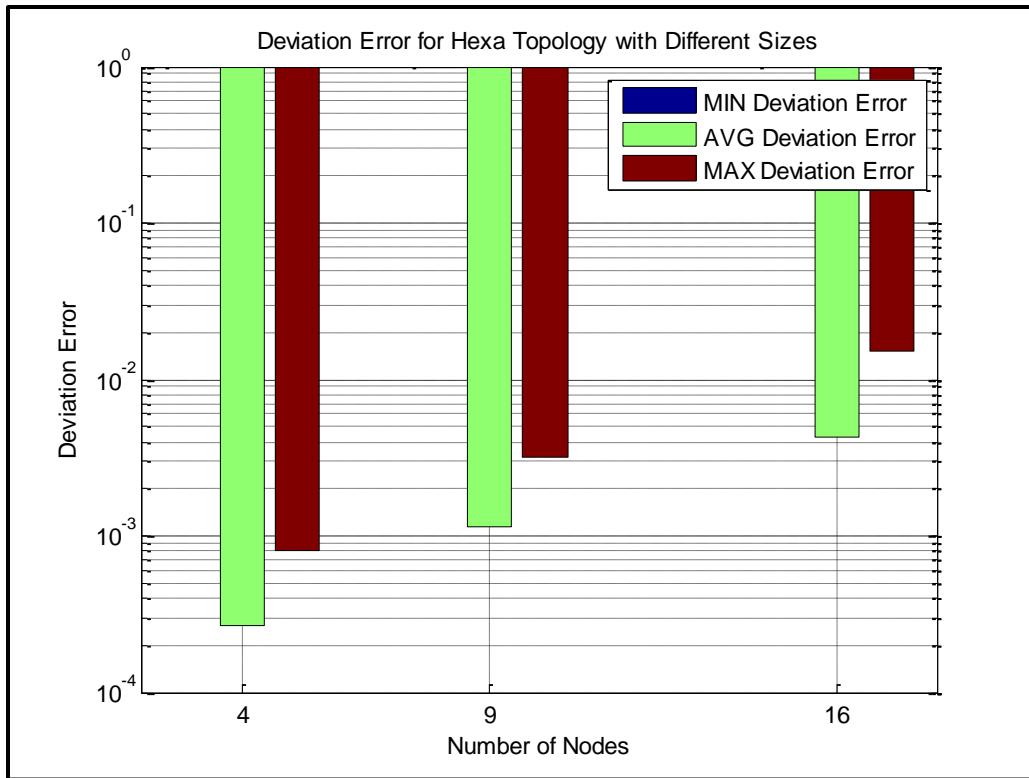


Figure 4-35 Deviation error for Hexa Topology

Figure 4-36 shows the deviation error for the Random topology with various sizes. The figure shows that the deviation error increases as the network size increases. The error equal to zero (no bars) for the 4-nodes network, between  $(0 - 0.2 \times 10^{-2})$  for the 9-nodes network, and between  $(0 - 0.9 \times 10^{-2})$  for the 16-nodes network. This behaviour is expected as the network size increases.

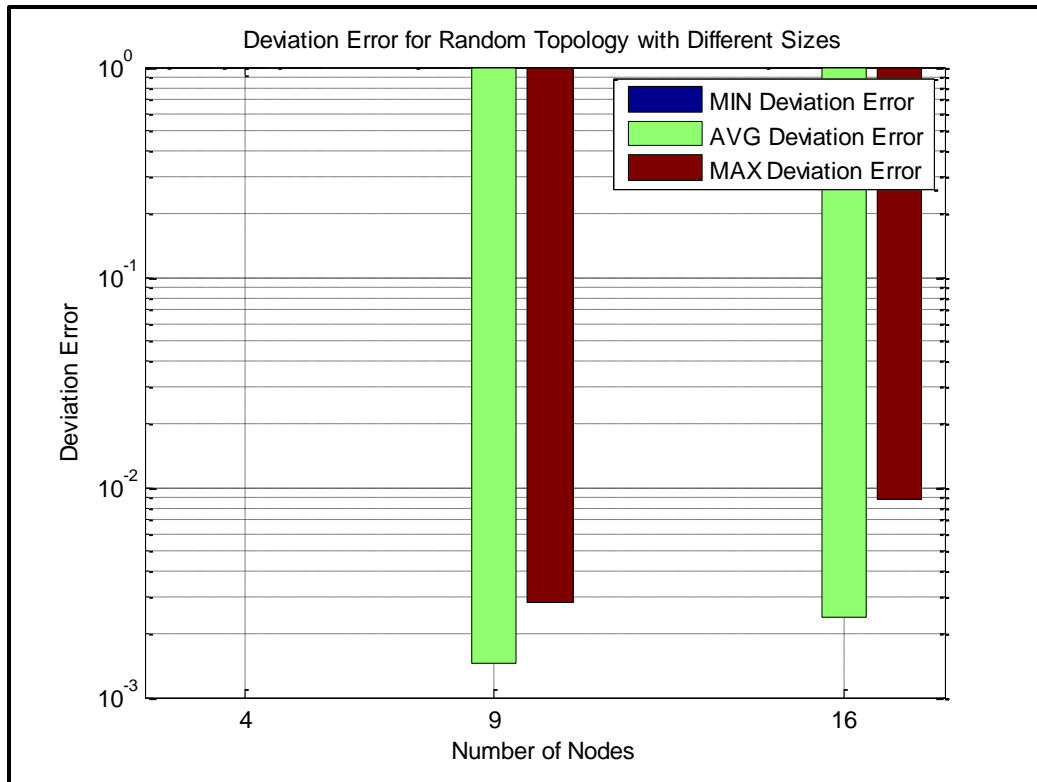


Figure 4-36 Deviation error for Random Topology

#### 4.7.2 Summarized Practical Results for Different Sizes (4, 9, and 16)

Figure 4-37 shows the practical deviation error for the Grid topology with various sizes. The figure shows that the deviation error increases as the network size increases. The error deviates between  $(0 - 0.2 \times 10^{-2})$  for the 4-nodes network, between  $(0.2 \times 10^{-2} - 0.6 \times 10^{-2})$  for the 9-nodes network, and between  $(0.7 \times 10^{-3} - 0.2 \times 10^{-1})$  for the 16-nodes network. This behaviour is expected as the network size increases.

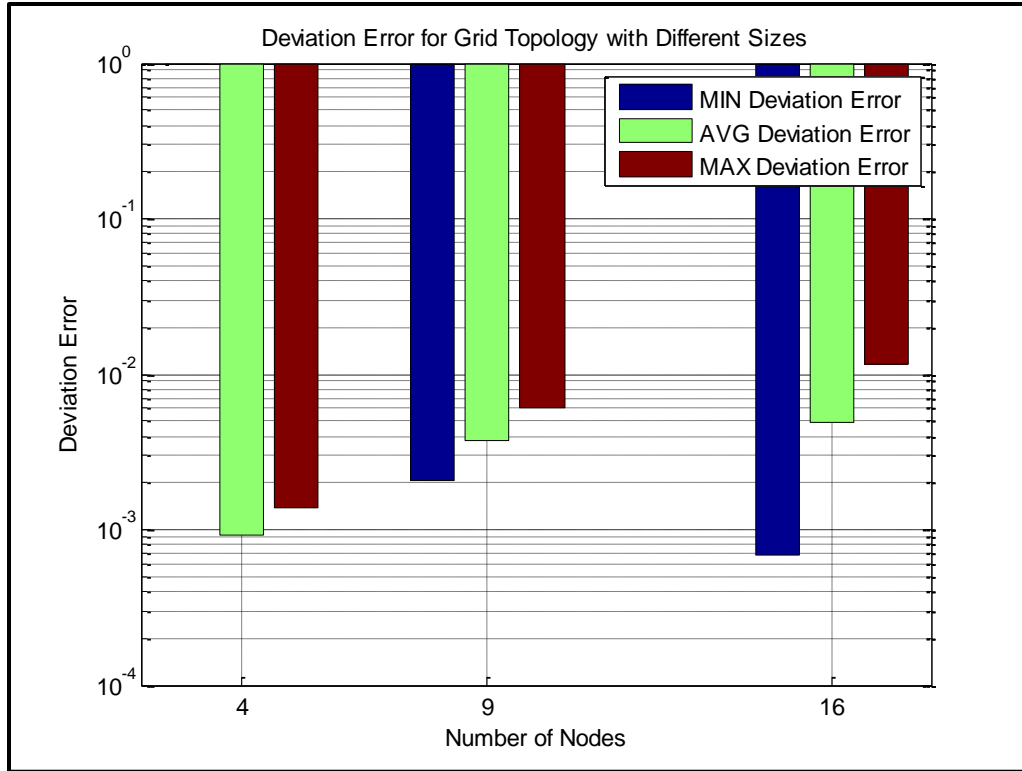


Figure 4-37 Deviation error for the practical results

### 4.7.3 Simulation vs. Practical Results for Different Sizes (4, 9, and 16)

The error results presented in sections 4.7.1 and 4.7.2 are compared in this section. Figure 4-38 shows the deviation error for the 4-node grid topology. It can be noted that the error values increase slightly when comparing the simulation and practical results. The maximum simulation error is  $=0.3 \times 10^{-3}$  while the maximum practical error is  $=0.1 \times 10^{-2}$ ; The average simulation error is  $=0.1 \times 10^{-3}$  while the average practical error  $0.9 \times 10^{-3}$ , and minimum simulation and practical error are both Zero which represents by blue colour in this curve.

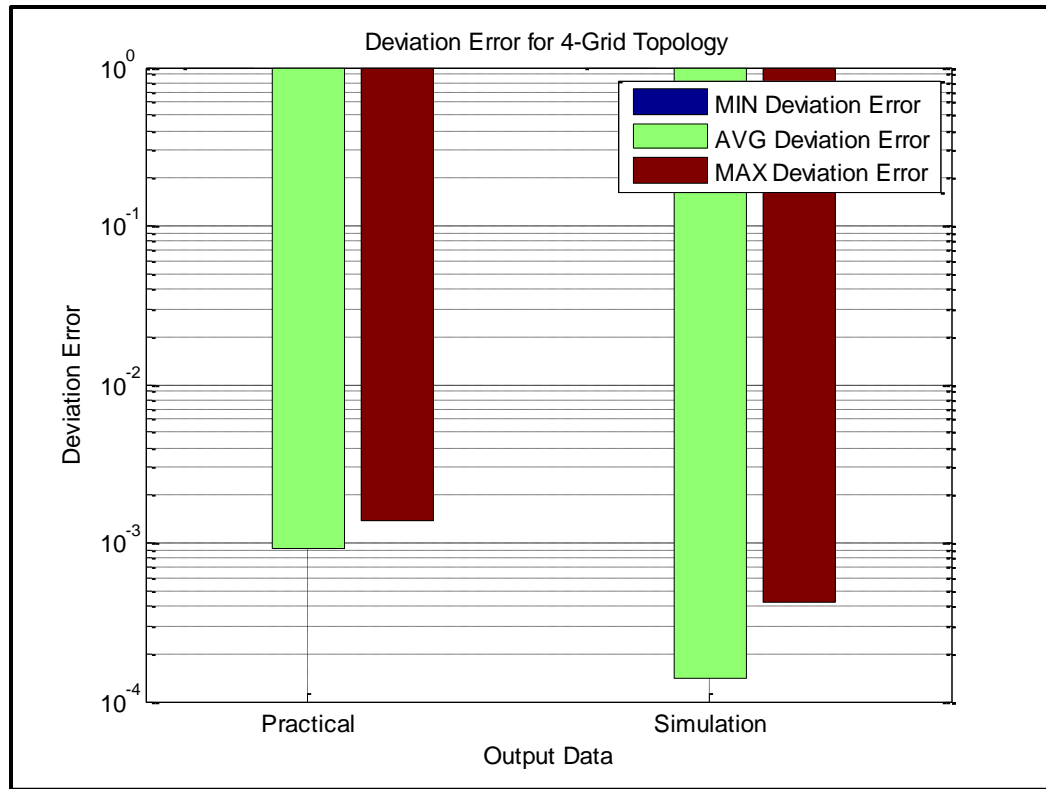


Figure 4-38 Simulation and practical deviation error for 4-nodes Grid

Figure 4-39 shows the deviation error for the 9-node grid topology. It can be noted that the error values increase slightly when comparing the simulation and practical results. The maximum simulation Error is  $=0.5 \times 10^{-3}$  while the maximum practical error is  $= 0.2 \times 10^{-2}$ ; The average simulation error is  $=0.2 \times 10^{-2}$  while the average practical error  $0.4 \times 10^{-2}$ , and the minimum simulation error ( $=0.5 \times 10^{-2}$ ) while the minimum practical error ( $= 0.2 \times 10^{-3}$ ).

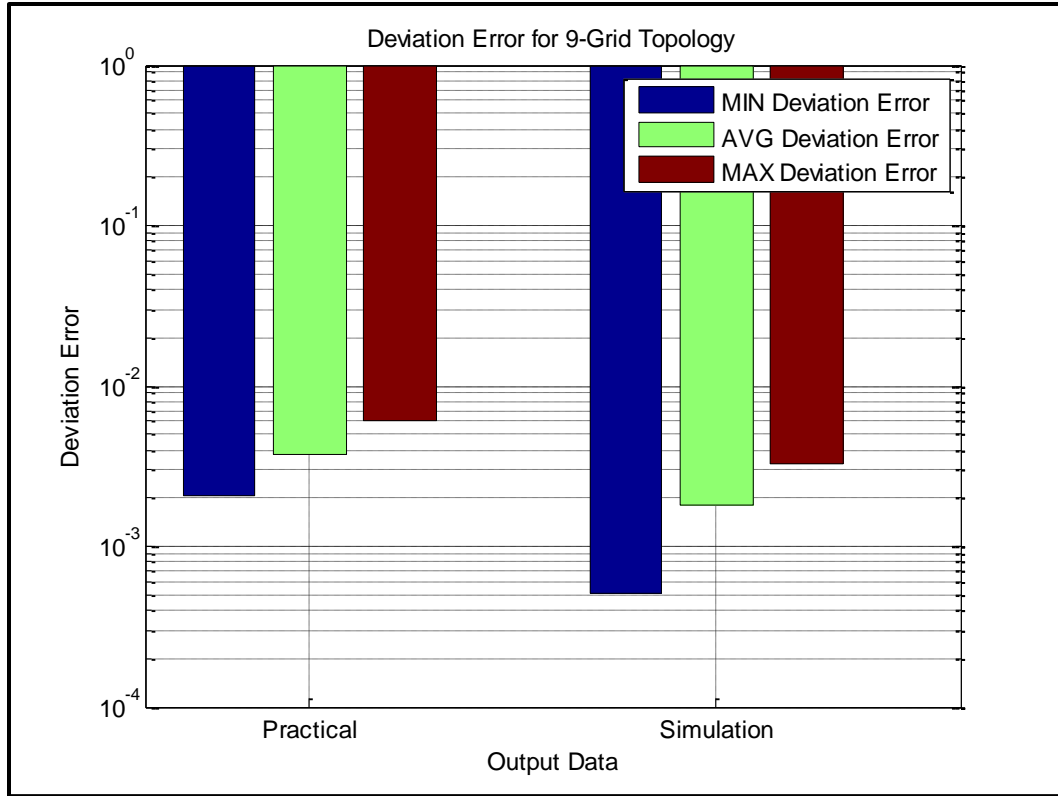


Figure 4-39 Simulation and practical deviation error for 9-nodes Grid

Figure 4-40 shows the deviation error for the 16-node grid topology. It can be noted that the error values increase slightly when comparing the simulation and practical results. The maximum simulation Error is  $=0.2 \times 10^{-1}$  while the maximum practical error is  $= 0.2 \times 10^{-1}$ ; The average simulation error is  $=0.5 \times 10^{-2}$  while the average practical error  $= 0.5 \times 10^{-2}$ , and the minimum simulation error ( $=0.2 \times 10^{-2}$ ) while the minimum practical error ( $= 0.7 \times 10^{-3}$ ).

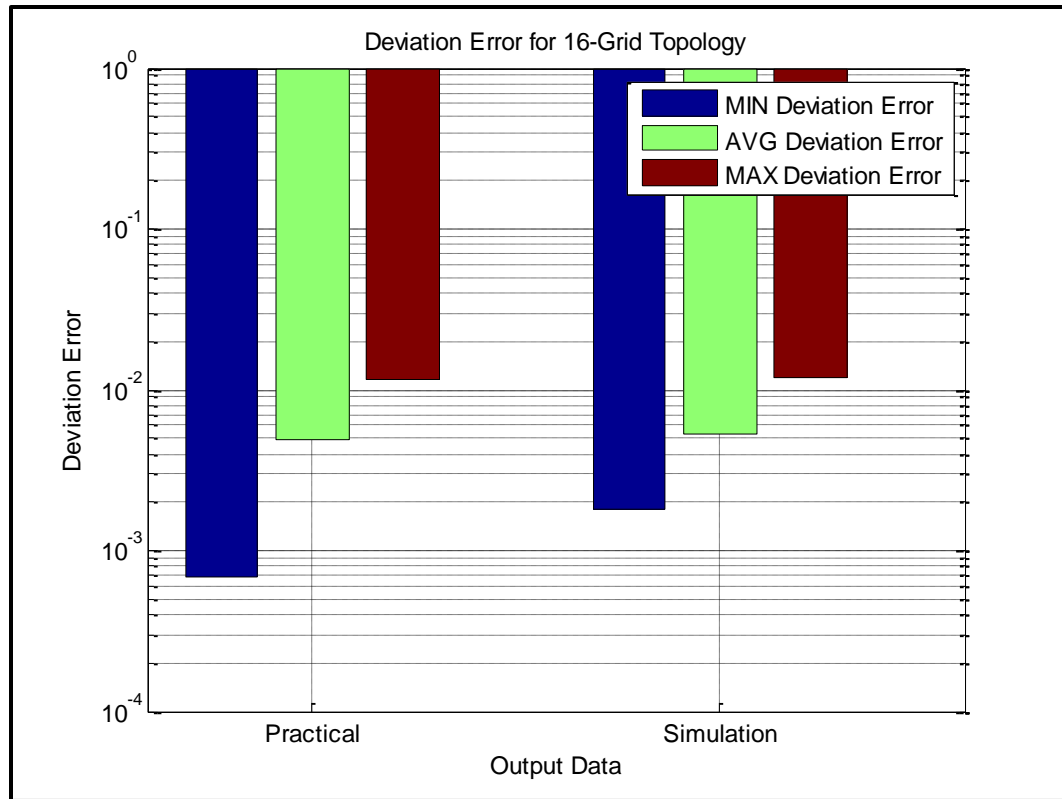


Figure 4-40 Simulation and practical deviation error for 16-nodes Grid

#### 4.7.4 Practical Results for Different Synchronization Protocols

This section shows the behaviour without stopping criterion of our protocol and other two protocols that discussed in the literature: Rftsp (Rated Flooding Time Synchronization Protocol) [13] and Egtsp (Energy-Efficient Gradient Time Synchronization Protocol) [34]. Table 4-19 describes the specifications of the RFTSP, EGTSP and proposed protocol.

Table 4-19 Specifications of three protocols

Specification	RFTSP	EGTSP	Our Protocol
Type	Centralized/Tree	Distributed	Distributed
Reference/Root Node	Reference/Root Node to start the Flooding Process	Broadcasting Packet contain the local information about the neighbours to start periodically the updates	Directly communicate with the neighbours and no reference node
Failures	Node/Link Failures	None	None
Overhead Problem	High overhead, power consumption is high and life of time is low	Less overhead, power efficient and high life of time comparing with FTSP	Suitable for dense network, power efficient and high life of time
Communication Type	Multi Hop Communication	Single Hop Communication	Single Hop Communication
Compensation	Compensate drift and offset at the same time	Compensate drift and offset individually	Compensate drift and offset at the same time
Communication Cycles	High	Medium	Low

We implemented these protocols 9-nodes grid topology using Micaz sensor nodes and the **TMilli** timer has been used in this implementation for three protocols. Figure 4-41 and Figure 4-42 show the error curves of the two protocols compared to our protocol for 9-nodes grid topology. We notice that our protocol is better and faster than the two protocols and has different shape from the previous protocols.

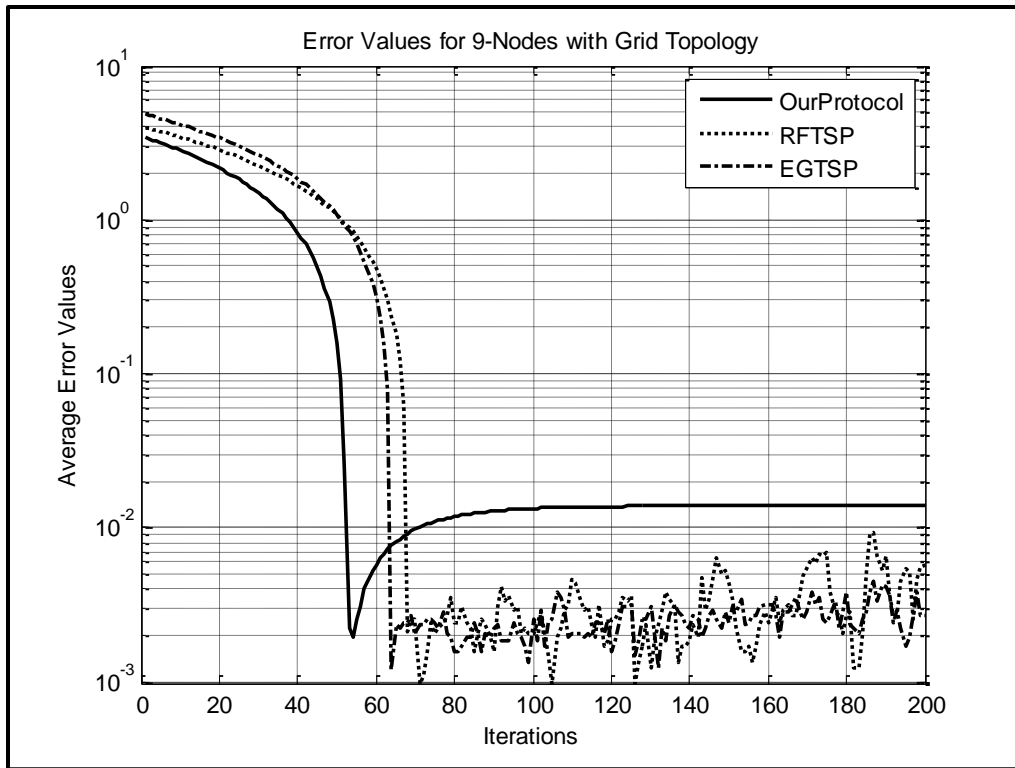


Figure 4-41 Average error curve of different protocols with 9-Grid nodes

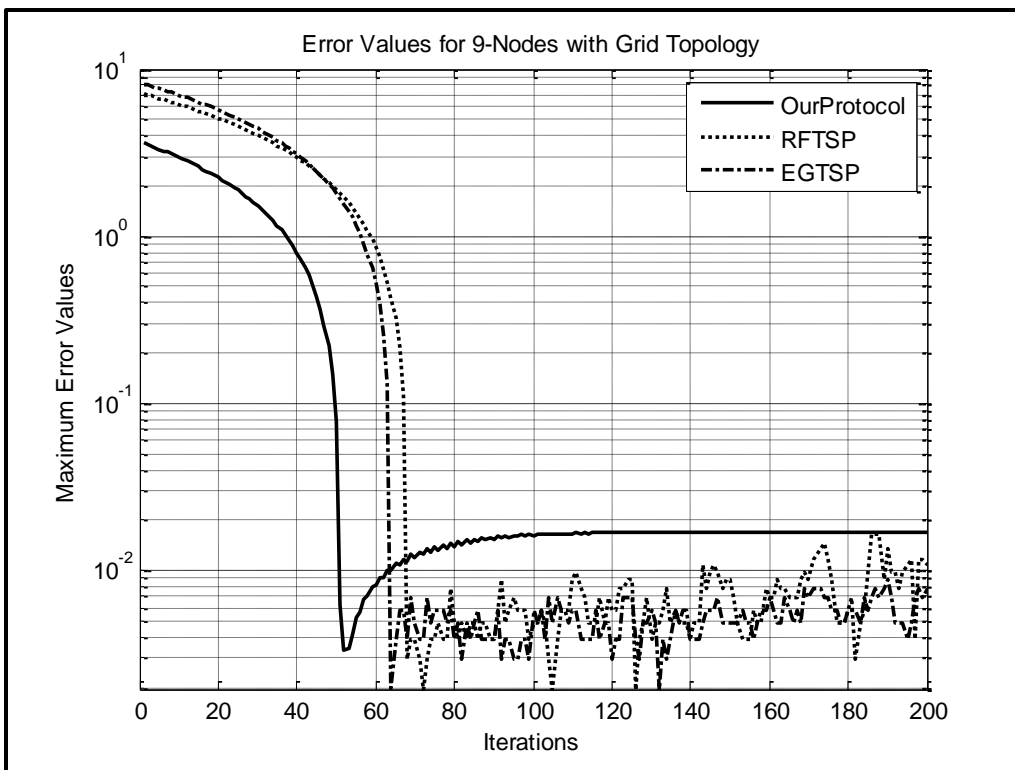


Figure 4-42 Maximum error curve of different protocols with 9-Grid nodes



The two figures show that our protocol differs from the other protocols in the shape of error curve; our protocol has two regions: dip and steady state regions with different error values while other protocols have only steady state region. Our protocol reaches the minimum values with less number of iteration comparing to other protocols. Therefore, our protocol is faster than the other protocols and we can stop at the dip region to save time, memory, and power of the sensor nodes. Our protocol needs around 53 iterations to reach the dip region with minimum error while EGTSP needs 62 iterations and RFTSP needs around 70 iterations to reach the steady state with minimum error.

From practical view, communication and memory overheads in the time synchronization protocols are important factors to take in account. For RFTSP, the amount of memory which is used to save the collected time values specifies the memory requirements. While in the distributed protocols the amount of memory which is used to save and track the neighbors' information defined the memory requirements. Since the ROM is used to store the program code of the protocol, the overhead of any protocol depends on the code size. The energy consumption of the sensor nodes depends on the communication between the nodes and the packet size: more communication cycles and larger packet size increase the consumption energy; while small packet size consumes less energy. Therefore, the overhead problem is another factor which influences the energy consumption; from receiving time of the time information to the processing state until the clock is updated; amount of energy will be consumed.

Regarding receiving a new synchronization packet from the server node, RFTSP is required to save 8 values from the received times to perform a least-squares regression. This regression consists of many multiplication and floating point division operations

besides the effect of the flooding process on the network; this will increase the overhead and energy consumption of the network. While in EGTSP is not only performed these steps, but also it evaluates the average of the clock offset and rate multipliers of the neighbors by considering saved time for these neighbors; this will increase the overhead on all nodes and increase the energy consumption for the network. On the other hand, our protocol uses sum and product operations on the time information for each node by considering the saved time for the neighbors, and the packet size of this protocol is less than the RFTSP and EGTSP. Therefore, the overhead and energy consumption in our protocol are less than comparing to RFTSP and EGTSP.

#### **4.7.5 Simulation Results for Changing the Initial Values/Different Runs**

This section shows the effect of changing the initial time values for the sensor nodes. Two ranges of time were used in the simulation part for 9 nodes; first range of the time locates between [0.2-0.3] and the period is 0.1, while the second range locates between [0.2-0.5] and the period is 0.3. Table 4-20 and Table 4-21 summarized the simulation results for three topologies: Grid, Hexa and Random with 9-nodes, the effect of increasing the period of initial values that increases the number of iterations that are detected by the filter in the dip region and error values increases for all topologies. On the other hand, the iteration and error values slightly change for the same period either 0.1 or 0.3. The average of the exact iterations mentioned by (**AIE**) in the grid topology is 44 iterations and the average of the detected iterations by the filter in the dip region mentioned by (**AID**) is 42.875 iterations when the period is equal to 0.1, while the average steady state iterations mentioned by (**AIS**) is higher than **AID** by three times. On the other hand, when the period is 0.3; the number of iterations that are detected in the

three regions is higher than the values when the period is 0.1; also this has occurred in hex and random topologies.

Table 4-20 Number of iterations for different periods of 9-nodes

Topology		[0.2-0.3]	[0.2-0.5]
Grid	C	1.035	1.025
	AIE	44	49
	AIS	161.625	182.625
	AID	42.875	48.625
Hexa	C	1.045	1.046
	AIE	37.125	41
	AIS	92	95.625
	AID	37.25	40.75
Random	C	1.04	1.04
	AIE	37	40.125
	AIS	91.375	94.875
	AID	36.75	40.25

Table 4-21 shows the simulation error values for the three topologies with two periods 0.1 and 0.3, we notice that the difference between the average exact error mentioned by (AEE) for the grid topology and the average error that is detected by the filter mentioned by (AED) for 0.1 is equal to 1.82 ms, while for 0.3 the difference is 3.419 ms and the difference between the (AEE) and the average steady state error mentioned by (AES) is for 0.3 higher than the difference between the two when the period is 0.1. It seems that the error values decreases when the period is decreased and same thing for other topologies.

Table 4-21 Error value for different periods of 9-nodes

Topology		[0.2-0.3]	[0.2-0.5]
Grid	<b>C</b>	1.035	1.025
	<b>AEE</b>	0.000384398	0.000251152
	<b>AES</b>	0.013870435	0.01387339
	<b>AED</b>	0.002213373	0.003677471
Hexa	<b>AEE</b>	1.045	1.046
	<b>AES</b>	0.000377289	0.000293071
	<b>AED</b>	0.010710197	0.010708134
	<b>AEE</b>	0.001512886	0.00150807
Random	<b>AEE</b>	1.04	1.04
	<b>AES</b>	9.42024E-05	0.00036984
	<b>AED</b>	0.010708493	0.010709287
	<b>AEE</b>	0.001547079	0.001512971

## 4.8 Summary

In this chapter, a new stopping criterion was proposed to detect that a synchronization time has been reached in the DR of the AP algorithm. Once the synchronization time is detected, the iteration process of the AP algorithm stops.

This criterion consists of a two filters: a form of difference filter to detect a flipping point (minimum point) in the AP algorithm curve and an averaging filter to smooth the fluctuation of the time values of the AP algorithm. The stopping criterion was tested using simulation and practical networks for various network topologies and/or for various network sizes. Also, this criterion was compared with steady state criterion. Extensive simulation was carried out to verify that the criterion works. The stopping criterion and the AP algorithm were deployed in real grid networks of various sizes using Micaz sensor nodes. Various parameters of the criterion were tested to test its sensitivity in detecting synchronization. It was concluded that the AP algorithm and the stopping criterion constitute a very good protocol that is able to synchronize all the nodes in a network with

less number of iterations compared to steady state synchronization that is usually used for many protocol and with very good accuracy compared with the exact synchronization with the master node.

The protocol was also compared with two other time synchronization protocols. It was noted that the performance of the protocol in synchronization outperform these protocols. In addition to that, the proposed protocol is very simple, needs only the local times of the node and the nodes that are connected to it, and can work for any size and type of network. The suggested protocol due to its simplicity can be used in harsh environment in unstructured networks. The protocol is also globally stable and it is linear in complexity. All these excellent features of the protocol make it a very excellent choice in many applications.

## CHAPTER FIVE

### Conclusion: Summary and Future Work

This thesis proposes a new time synchronization protocol for wireless sensor networks. The protocol consists of a new averaging protocol algorithm and a new stopping criterion. The AP algorithm iteratively synchronizes all nodes in a network with the master node time. It is based on averaging the time values received from only neighboring nodes. It uses simple operations of scalar multiplication and addition. The stopping criterion enables the node using only its time values to detect that synchronization is reached and halt the iterative synchronization process. The stopping criterion is nothing but two FIR filters: a difference filter and an averaging filter.

#### 5.1 Summary

The thesis detailed work, achievements, and contribution can be summarized in the following paragraphs.

First, a literature review has been accomplished for the wireless sensor networks; this review compares wireless and wired networks, shows the advantages of using wireless networks over the wired ones in the sensor networks. In addition, the review presents the structure of the sensor nodes, types, and specifications. It also highlights the main features of the hardware that used to implement the wireless network which is Micaz nodes. Moreover, the review presents the importance of time synchronization of the nodes in WSNs.

Second, it is noted that the main causes of time variations of the nodes are the drift and the offset of the clocks in the nodes in the sensor network. Therefore, different time synchronization protocols have been developed to minimize the drifts among nodes. The thesis presents a comprehensive literature review for the of time synchronization protocols of WSNs. These protocols were divided into three categories depending on the hierarchy structure of the network: tree protocols, cluster protocols, and distributed protocols. It is noted that tree and cluster protocols have many problems that preventing them from being used efficiently (fast convergence, low processing hardware) in dynamic and dense topologies and in harsh environment. The distributed protocols have the capability to overcome these problems if one such protocol can be designed. A new consensus distributed time synchronization protocol is proposed for WSNs; where the Consensus Clock Synchronization (CCS) is used to minimize the clock differences between nodes that are located geographically in close proximity to each other especially for dense networks. The proposed protocol synchronizes each node by only receiving the time values from the neighboring nodes connected to that node. The protocol does not need to know about the whole network. The protocol consists of an averaging protocol algorithm and a stopping method. In a node, the averaging algorithm iteratively averages the received time values from the neighboring nodes and updates its time values with this new average value. At each iteration each node keeps updating its time value until the convergence state is reached and detected by the stopping method in which the node stops the iteration and declares synchronization is reached. The proposed averaging protocol exhibits a unique behavior compared to other protocols. The error curve, that represents the difference between the current time value of a certain node with the time

value of the master node, dips quickly to a very low value then increases slowly and saturates in the steady state region. The stopping method detects that a minimum error is reached while the process is still in the dip region and declares synchronization. By doing this, synchronization is achieved for each node with less number of iteration and with better accuracy.

The protocol is tested using three types of topologies –Grid, Hexa, and Random with various sizes 4, 9, and 16 nodes. Conformity of the simulated and practical results are observed.

The deployment of the protocol in real networks shows that the proposed protocol reaches synchronization with error value of 1.1 ms in 17 iterations which is less than the steady state iterations by 2.7 times in the grid topology of size 4. In the grid topology of size 9, it reaches synchronization with error value of 4.2 ms in 53 iterations which is less than the steady state iterations by 3.6 times. For the same topology of size 16, it reaches synchronization with error value of 5.1 ms in 95 iterations which is less than the steady state iterations by 4.6 times. While in the simulation results for the same topology, the proposed protocol reaches synchronization with error value of 0.5 ms in 15 iterations which is less than the steady state iterations by 2.4 times in the topology of size 4. When the network size is 9 nodes, it reaches synchronization with error value of 2.2 ms in 43 iterations which is less than the steady state iterations by 3.77 times. For the same topology of size 16, it reaches synchronization with error value of 5.7 ms in 80 iterations which is less than the steady state iterations by 4.37 times. The same relationship resulted from the hexa and random topologies.



The protocol was also compared with two other time synchronization protocols in the real grid networks of size 9 nodes. It was noted that the performance of the proposed protocol in synchronization outperform these protocols. In addition, the error curve of our protocol has different shape comparing with the two protocols and can reach minimum error with less iteration. The proposed protocol reaches synchronization in 53 iterations with error value of 4.2 ms. On the other hand, RFTSP reaches synchronization in 70 iterations and EGTSP in 62 iterations with error value between [3-8] ms. Therefore, proposed protocol achieves the synchronization with less iteration and minimum error, less operations, it is suitable for dense networks under harsh environments, less overhead, and linear complexity.

The summary is concluded by stating the main features of this protocol. It is applicable to deploy in harsh environments and has some properties such as: computationally light, scalable, applicable for topology changes, distributed, robust to node and link failure, it does not need a leader node, it has global stability regardless to the network connectivity and the stopping criterion, controllable time accuracy, single hop communication among nodes, simplicity with little communication overhead, and Hardware-friendly using Micaz nodes. This protocol can serve different applications like monitoring pollution, tracking objects and monitoring the oil industry.

## **5.2 Future Work**

In order to develop a time synchronization protocol for synchronizing the WSNs, number of problems should be solved. These problems result variety of research directions that need to be pursued to make the protocol more effective by: modifying this protocol to serve multi-hop communication and take in the consideration delay factor in the

transmission and reception packets between nodes, studying the behavior of this protocol under random connectivity with various sizes, designing an adaptive filter to track the exact minimum and stop at this minimum exactly, and designing a fitting model to smooth the time values of each node and easily detect the iteration with the minimum error.

## REFERENCES

1. Chee-Yee, C. and S.P. Kumar, Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 2003. 91(8): p. 1247-1256.
2. Akyildiz, I.F., et al., A survey on sensor networks. *Communications Magazine*, IEEE, 2002. 40(8): p. 102-114.
3. Bernard, T. and H. Fouchal. Efficient Communications over Wireless Sensor Networks. in *Global Telecommunications Conference (GLOBECOM 2010)*, 2010 IEEE. 2010.
4. Sisinni, E., et al. High availability wireless temperature sensors for harsh environments. in *Sensors Applications Symposium (SAS)*, 2012 IEEE. 2012.
5. Yoshigoe, K. Data-driven data transmission mechanism for wireless sensor networks in harsh communication environment. in *GLOBECOM Workshops (GC Wkshps)*, 2010 IEEE. 2010.
6. Gandelli, A., S. Marchi, and R.E. Zich. Sensor Networks Performance in Harsh Environments. in *Sensors for Industry Conference*, 2005. 2005.
7. Kadri, A. Performance of IEEE 802.15.4-based wireless sensors in harsh environments. 2012.
8. Szewczyk, R., et al., Habitat monitoring with sensor networks. *Commun. ACM*, 2004. 47(6): p. 34-40.
9. Hohlt, B., L. Doherty, and E. Brewer, Flexible power scheduling for sensor networks, in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. 2004, ACM: Berkeley, California, USA. p. 205-214.
10. Zennaro, D., et al. Fast clock synchronization in wireless sensor networks via ADMM-based consensus. in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2011 International Symposium on. 2011.
11. Herman, T. and C. Zhang, Best paper: Stabilizing clock synchronization for wireless sensor networks. 2006. p. 335-349.
12. Ganeriwal, S., R. Kumar, and M.B. Srivastava, Timing-sync protocol for sensor networks, in *Proceedings of the 1st international conference on Embedded networked sensor systems*. 2003, ACM: Los Angeles, California, USA. p. 138-149.
13. Maroti, M., et al., The flooding time synchronization protocol, in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. 2004, ACM: Baltimore, MD, USA. p. 39-49.
14. Arvind, K., Probabilistic clock synchronization in distributed systems. *Parallel and Distributed Systems*, *IEEE Transactions on*, 1994. 5(5): p. 474-487.
15. Elson, J., L. Girod, and D. Estrin, Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 2002. 36(SI): p. 147-163.
16. He, J., et al., Time Synchronization in WSNs: A Maximum-Value-Based Consensus Approach\*. *Automatic Control*, *IEEE Transactions on*, 2013. PP(99): p. 1-1.

17. He, J., et al., Study of consensus-based time synchronization in wireless sensor networks. *ISA Transactions*, 2014. 53(2): p. 347-357.
18. Jianping, H., et al. Time synchronization in WSNs: A maximum value based consensus approach. in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. 2011.
19. Greunen, J.v. and J. Rabaey, Lightweight time synchronization for sensor networks, in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. 2003, ACM: San Diego, CA, USA. p. 11-19.
20. Jiming, C., et al., Feedback-Based Clock Synchronization in Wireless Sensor Networks: A Control Theoretic Approach. *Vehicular Technology, IEEE Transactions on*, 2010. 59(6): p. 2963-2973.
21. Yoon, S., C. Veerarittiphan, and M.L. Sichitiu, Tiny-sync: Tight time synchronization for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2007. 3(2): p. 8.
22. Sundararaman, B.e.a., Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Netw.*
23. Kyoung-lae, N., E. Serpedin, and K. Qaraqe, A New Approach for Time Synchronization in Wireless Sensor Networks: Pairwise Broadcast Synchronization. *Wireless Communications, IEEE Transactions on*, 2008. 7(9): p. 3318-3322.
24. Dai, H. and R. Han, TSync: a lightweight bidirectional time synchronization service for wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 2004. 8(1): p. 125-139.
25. Fontanelli, D. and D. Macii. Master-less time synchronization for wireless sensor networks with generic topology. in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*. 2012.
26. Schenato, L. and F. Fiorentin, Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, 2011. 47(9): p. 1878-1886.
27. Mallada, E. and T. Ao. Distributed clock synchronization: Joint frequency and phase consensus. in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. 2011.
28. Weilian, S. and I.F. Akyildiz, Time-diffusion synchronization protocol for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 2005. 13(2): p. 384-397.
29. Yi, S., J. Qing, and Z. Kai. A clustering scheme for Reachback Firefly Synchronicity in wireless sensor networks. in *Network Infrastructure and Digital Content (IC-NIDC), 2012 3rd IEEE International Conference on*. 2012.
30. Sommer, P. and R. Wattenhofer, Gradient clock synchronization in wireless sensor networks, in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. 2009, IEEE Computer Society. p. 37-48.
31. Yildirim, K.S. and A. Kantarci, External Gradient Time Synchronization in Wireless Sensor Networks. *Parallel and Distributed Systems, IEEE Transactions on*, 2014. 25(3): p. 633-641.

32. Zhao, D., Z. An, and Y. Xu, Time Synchronization in Wireless Sensor Networks Using Max and Average Consensus Protocol. *International Journal of Distributed Sensor Networks*, 2013. 2013.
33. Leidenfrost, R., W. Elmenreich, and C. Bettstetter. Fault-tolerant averaging for self-organizing synchronization in wireless ad hoc networks. in *Wireless Communication Systems (ISWCS), 2010 7th International Symposium on*. 2010.
34. Apicharttrisorn, K., S. Choochaisri, and C. Intanagonwiwat. Energy-Efficient Gradient Time Synchronization for Wireless Sensor Networks. in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on*. 2010.
35. Qun, L. and D. Rus, Global clock synchronization in sensor networks. *Computers, IEEE Transactions on*, 2006. 55(2): p. 214-226.
36. Cremaschi, M., O. Simeone, and U. Spagnolini. Distributed timing synchronization for sensor networks with coupled discrete-time oscillators. 2007.
37. Carli, R., E. D'Elia, and S. Zampieri. A PI controller based on asymmetric gossip communications for clocks synchronization in wireless sensors networks. in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. 2011.
38. Schenato, L. and G. Gamba. A distributed consensus protocol for clock synchronization in wireless sensor network. in *Decision and Control, 2007 46th IEEE Conference on*. 2007.
39. Olfati-Saber, R. and R.M. Murray, Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 2004. 49(9): p. 1520-1533.
40. Bertsekas, D.P. and J.N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. 1989: Prentice-Hall, Inc. 715.
41. Olshevsky, A. and J.N. Tsitsiklis. Convergence Rates in Distributed Consensus and Averaging. in *Decision and Control, 2006 45th IEEE Conference on*. 2006.
42. Wu, C.W. Agreement and consensus problems in groups of autonomous agents with linear dynamics. in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. 2005.
43. Schizas, I.D., A. Ribeiro, and G.B. Giannakis, Consensus in Ad Hoc WSNs With Noisy Links Part I: Distributed Estimation of Deterministic Signals. *Signal Processing, IEEE Transactions on*, 2008. 56(1): p. 350-364.
44. Olfati-Saber, R., J.A. Fax, and R.M. Murray, Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 2007. 95(1): p. 215-233.
45. Maggs, M.K., S.G. O'Keefe, and D.V. Thiel, Consensus Clock Synchronization for Wireless Sensor Networks. *Sensors Journal, IEEE*, 2012. 12(6): p. 2269-2277.
46. Scherber, D.S. and H.C. Papadopoulos. Locally constructed algorithms for distributed computations in ad-hoc networks. in *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*. 2004.
47. Hoffmeister and Baeck, *Genetic Algorithms and Evolution Strategies: Similarities and Differences* 1992.
48. Pohlheim, H., *Evolutionäre Algorithmen - Verfahren, Operatoren, Hinweise aus der Praxis*. Berlin, Heidelberg, New York.

49. Raymer, M.L., et al., Dimensionality Reduction Using Genetic Algorithms. 2000.
50. Siegfried, R.E. and S. Vössner, Genetic Algorithms With Cluster Analysis For Production Simulation. 1997.
51. Djenouri, D., et al., Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks. *Ad Hoc Networks*, 2013. 11(8): p. 2329-2344.
52. Huang, G., et al., Long term and large scale time synchronization in wireless sensor networks. *Computer Communications*, 2014. 37(0): p. 77-91.
53. *Handbook for Digital Signal Processing*, ed. K.M. Sanjit and F.K. James. 1993: John Wiley & Sons, Inc. 1312.

# APPENDIX

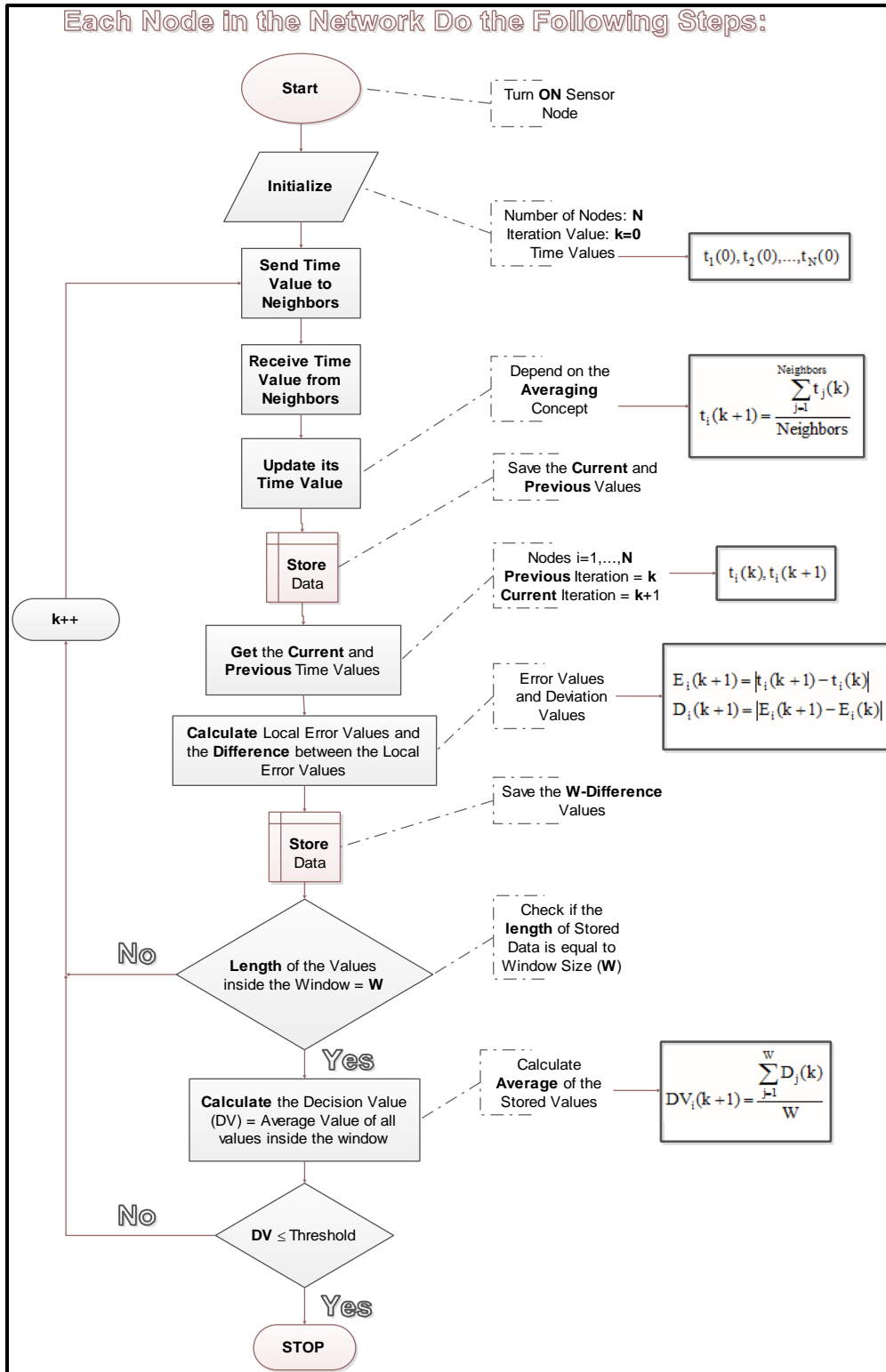


Figure A-1 Averaging Protocol with Steady State Stopping Criterion

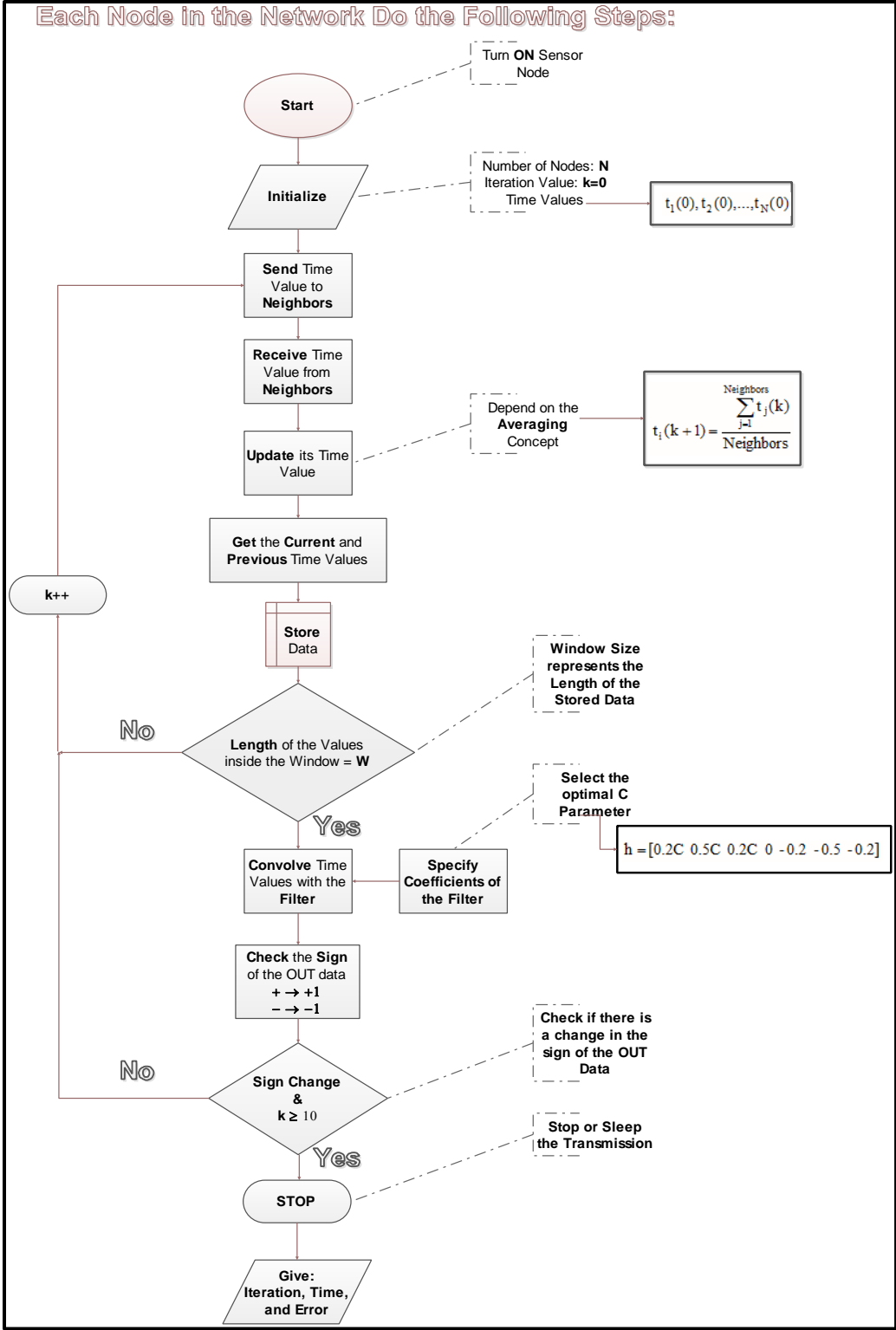


Figure A-2 Averaging Protocol with Dip Stopping Criterion



## VITAE

**Name** : Ibrahim Ahmed Abdallah Nemer.

**Nationality** : Jordanian.

**Date of Birth** : 3/24/1988

**Email** : ibrahim.nimer@hotmail.com, inemer@kfupm.edu.sa

**Address** : KFUPM, Dhahran, Al-Sharqiya, KSA.

**Academic Background** :

1. Bachelor Degree in Science of Electrical Engineering at Palestine, Birzeit University, Rammallah in June 2011.
2. Teaching Assistant in Electrical Engineering Department of Birzeit University, Rammallah, Palestine from September 2011 to January 2013.
3. Research Assistant in Electrical Engineering Department of KFUPM, Dhahran, KSA from January 2013 until now.
4. Master Degree in Science of Telecommunication Engineering at KFUPM, Dhahran, KSA from January 2013 until now.