# Consensus-based Time Synchronization Algorithms for Wireless Sensor Networks with Topological Optimization Strategies for Performance Improvement

**Niranjan Panigrahi**

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

# Consensus-based Time Synchronization Algorithms for Wireless Sensor Networks with Topological Optimization Strategies for Performance Improvement

*Dissertation submitted in partial fulfillment*

*of the requirements of the degree of*

***Doctor of Philosophy***

*in*

***Computer Science and Engineering***

*by*

***Niranjan Panigrahi***

(Roll Number: 511CS110)

*based on research carried out*

*under the supervision of*

***Prof. Pabitra Mohan Khilar***

,

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

,

# Certificate of Examination

Roll Number: *511CS110*
Name: *Niranjan Panigrahi*
Title of Dissertation: *Consensus-based Time Synchronization Algorithms for Wireless Sensor Networks with Topological Optimization Strategies for Performance Improvement*

We the below signed, after checking the dissertation mentioned above and the official record book (s) of the student, hereby state our approval of the dissertation submitted in partial fulfillment of the requirements of the degree of *Doctor of Philosophy* in *Computer Science and Engineering* at *National Institute of Technology Rourkela*. We are satisfied with the volume, quality, correctness, and originality of the work.

Pabitra Mohan Khilar
Principal Supervisor

Durga Prasad Mohapatra
Member, DSC

Suchismita Chinara
Member, DSC

Susovan Samanta
Member, DSC

External Examiner

Chairperson, DSC

Head of the Department

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

**Prof. Pabitra Mohan Khilar**
Assistant Professor

,

# Supervisor's Certificate

This is to certify that the work presented in the dissertation entitled *Consensus-based Time Synchronization Algorithms for Wireless Sensor Networks with Topological Optimization Strategies for Performance Improvement* submitted by *Niranjan Panigrahi*, Roll Number 511CS110, is a record of original research carried out by him under my supervision and guidance in partial fulfillment of the requirements of the degree of *Doctor of Philosophy* in *Computer Science and Engineering*. Neither this dissertation nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Pabitra Mohan Khilar

# Dedication

*Dedicated to My Family...*

# Declaration of Originality

I, *Niranjan Panigrahi*, Roll Number *511CS110* hereby declare that this dissertation entitled *Consensus-based Time Synchronization Algorithms for Wireless Sensor Networks with Topological Optimization Strategies for Performance Improvement* presents my original work carried out as a doctoral student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections ‘‘Bibliography’’. I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

NIT Rourkela

*Niranjan Panigrahi*

# Acknowledgment

Firstly, I would like to express my sincere gratitude to my supervisor, *Prof. Pabitra Mohan Khilar* for the continuous support of my Ph. D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph. D. study.

Besides my supervisor, I would like to thank the rest of my DSC committee: *Prof. S. K. Rath*, *Prof. D. P. Mohapatra*, *Prof. S. Chinara* of our Department and *Prof. S. Samanta* of Electrical Engineering Department for their insightful comments and encouragement, but also for the hard question which incited me to widen my research from various perspectives.

I thank my fellow lab-mates for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years. Special thanks to *Srichandan*, *Soubhagya*, and *Ravi Sankar* of the department whose helping-hand solved many problems during the course of this work.

Last but not the least, my heartfelt thanks to my wife *Manasee* and my son *Dibyanshu* for their unconditional love and support. I have left with no words to express my gratitude to my beloved parents, brother, sister and parents-in-law who took untold pain to make my life happy and smooth throughout this research work.

*Niranjan Panigrahi*
NIT Rourkela
Roll Number: 511CS110

# Abstract

Wireless Sensor Networks (WSNs) have received considerable attention in recent years because of its broad area of applications. In the same breadth, it also faces many challenges. Time synchronization is one of those fundamental challenges faced by WSN being a distributed system. It is a service by which all nodes in the network will share a common notion of time. It is a prerequisite for correctness of other protocols and services like security, localization and tracking protocols. Several approaches have been proposed in the last decade for time synchronization in WSNs. The well-known methods are based on synchronizing to a reference (root) node's time by considering a hierarchical backbone for the network. However, this approach seems to be not purely distributed, higher accumulated synchronization error for the farthest node from the root and subjected to the root node failure problem. Recently, consensus based approaches are gaining popularity due its computational lightness, robustness, and distributed nature.

In this thesis, average consensus-based time synchronization algorithms are proposed, aiming to improve the performance metrics like number of iterations for convergence, total synchronization error, local synchronization error, message complexity, and scalability. Further, to cope up with energy constraint environment, Genetic algorithm based topological optimization strategies are proposed to minimize energy consumption and to accelerate the consensus convergence of the existing consensus-based time synchronization algorithms. All algorithms are analyzed mathematically and validated through simulation in MATLAB based PROWLER simulator.

Firstly, a distributed Selective Average Time Synchronization (SATS) algorithm is proposed based on average consensus theory. The algorithm is purely distributed (runs at each node), and each node exploits a selective averaging with the neighboring node having maximum clock difference. To identify the neighboring node with maximum clock difference, every node broadcasts a synchronization initiation message to the neighboring nodes at its local oscillation period and waits for a random interval to get the synchronization acknowledgment messages. After receiving acknowledgment messages, a node estimates relative clock value and sends an averaging message to the selected node. The iteration continues until all nodes reach an acceptable synchronization error bound. The optimal convergence of the proposed SATS algorithm is analyzed and validated through simulation and compared with some state-of-the-art, average consensus based time synchronization algorithms.

Furthermore, it is observed that most of the consensus-based time synchronization algorithms are one-hop in nature, i.e., the algorithms iterate by averaging with one-hop neighbors’ clock value. In a sparse network with a lower average degree of connectivity, these algorithms show poor performance. In order to have better convergence on the sparse network, a multi-hop SATS algorithm is proposed. The basic principle of multi-hop SATS algorithm remains same as that of SATS algorithm, i.e., performing selective averaging with the neighboring node, having maximum clock difference. But, in this case, the search for neighboring node goes beyond one hop. The major challenge lies in multi-hop search is the end-to-end delay which increases with the increase in hop count. So, to search a multi-hop neighboring node with maximum clock difference and with minimum and bounded end-to-end delay, a distributed, constraint-based dynamic programming approach is proposed for multi-hop SATS algorithm. The performance of the proposed multi-hop SATS algorithm is compared with some one-hop consensus time synchronization algorithms. Simulation results show notable improvement in terms of convergence speed, total synchronization error within a restricted hop count. The trade-off with the increase in number of hops is also studied.

The well-known consensus-based time synchronization algorithms are ‘‘all node based’’, i.e., every node iterates the algorithm to reach the synchronized state. This increases the overall message complexity and consumption of energy. Further, congestion in the network increases due to extensive synchronization message exchanges and induces the delay in the network. The delay induced in the message exchange is the main source of synchronization error and slows down the convergence speed to the synchronized (consensus) state. Hence, it is desirable that a subset of sensors along with a reasonable number of neighboring sensors should be selected in such a way that the resultant logical topology will accelerate the consensus algorithm with optimal message complexity and minimizes energy consumption. This problem is formulated as topological optimization problem which is claimed to be NP-complete in nature. Therefore, Genetic Algorithm (GA) based approaches are used to tackle this problem. Considering dense network topology, a single objective GA-based approach is proposed and considering sparse topology, a multi-objective Random Weighted GA based approach is proposed. Using the proposed topological optimization strategy, significant improvements are observed for consensus-based time synchronization algorithms in terms of average number of messages exchanged, energy consumption, and average mean square synchronization error.

***Keywords*: *Wireless Sensor Network*, *Consensus Time Synchronization*, *Distributed Constraint Dynamic Programming*, *Topological Optimization*, *Genetic Algorithm*, *Random Weighted Genetic Algorithm***

# Contents

**Certificate of Examination** **ii**

**Supervisor’s Certificate** **iii**

**Dedication** **iv**

**Declaration of Originality** **v**

**Acknowledgment** **vi**

**Abstract** **vii**

**List of Figures** **xiii**

**List of Tables** **xvi**

**List of Abbreviations** **xviii**

**List of Symbols** **xx**

**1 Introduction** **1**

- 1.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  - 1.1.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . 2
  - 1.1.2 Performance Metrics . . . . . . . . . . . . . . . . . . . 5
- 1.2 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
- 1.3 Research Objective . . . . . . . . . . . . . . . . . . . . . . . . 8
- 1.4 Major Contribution . . . . . . . . . . . . . . . . . . . . . . . . 8
- 1.5 Thesis Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
- 1.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

**2 Background & Literature Survey** **11**

- 2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
- 2.2 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
- 2.3 Traditional Time Synchronization Methods . . . . . . . . . . . . 13
  - 2.3.1 Remote Clock Reading Method . . . . . . . . . . . . . . 13
  - 2.3.2 Time Transmission Method . . . . . . . . . . . . . . . . 13
  - 2.3.3 Offset Delay Estimation Method . . . . . . . . . . . . . 14

2.3.4 Model based Method . . . . . . . . . . . . . . . . . . . . . . . . 14

2.4 Taxonomy of Time Synchronization Methods in WSNs . . . . . . . . . . . 15

2.4.1 Taxonomy based on synchronization issues . . . . . . . . . . . . . 15

2.4.2 Taxonomy based on application requirements . . . . . . . . . . . . 17

2.4.3 Taxonomy based on approaches . . . . . . . . . . . . . . . . . . . 18

2.5 Case study of state-of-the-art synchronization algorithms . . . . . . . . . . 22

2.6 Key Observations . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

2.7 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

**3 Consensus Time Synchronization Algorithm by Selective Averaging 31**

3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

3.2 System Model & Problem Formulation . . . . . . . . . . . . . . . . . . 32

3.2.1 Clock Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

3.2.2 Network Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

3.2.3 Energy Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

3.2.4 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . 33

3.2.5 Mathematical Preliminaries . . . . . . . . . . . . . . . . . . . . . 35

3.3 The SATS Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

3.4 Performance Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . 39

3.4.1 Optimality of Synchronization Error . . . . . . . . . . . . . . . . 40

3.4.2 Consensus Convergence . . . . . . . . . . . . . . . . . . . . . . . 41

3.4.3 Message Complexity . . . . . . . . . . . . . . . . . . . . . . . . 43

3.4.4 Energy Consumption Analysis . . . . . . . . . . . . . . . . . . . 44

3.5 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

3.5.1 Simulation Configuration . . . . . . . . . . . . . . . . . . . . . . 46

3.5.2 Simulation Results and Analysis . . . . . . . . . . . . . . . . . . 46

3.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

**4 Multi-hop Consensus Time Synchronization Algorithm: A Distributed Dynamic Programming Approach 53**

4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

4.2 System Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

4.2.1 Network Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

4.3 Multi-hop SATS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

4.3.1 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . 55

4.3.2 Motivational Example . . . . . . . . . . . . . . . . . . . . . . . . 55

4.4 Proposed Distributed Dynamic Programming Approach . . . . . . . . . . . 57

4.4.1 Overlapping Sub-structure . . . . . . . . . . . . . . . . . . . . . . 57

4.4.2 Recurrence Relation Formulation . . . . . . . . . . . . . . . . . . 58

4.5 The multi-hop SATS Algorithm . . . . . . . . . . . . . . . . . . . . . . 60

4.5.1 Algorithm Illustration . . . . . . . . . . . . . . . . . . . . . . . 61
4.6 Performance Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . 64
4.6.1 Message Complexity . . . . . . . . . . . . . . . . . . . . . . . 64
4.6.2 Energy Consumption Analysis . . . . . . . . . . . . . . . . . . . 65
4.7 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
4.7.1 Simulation Configuration . . . . . . . . . . . . . . . . . . . . . 67
4.7.2 Simulation Results and Analysis . . . . . . . . . . . . . . . . . . 68
4.8 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

**5 Topological Optimization Strategy for Consensus Time Synchronization Algorithms on dense topology: A Genetic Algorithm based Approach 76**
5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
5.2 System Models & Definitions . . . . . . . . . . . . . . . . . . . . . . 77
5.2.1 Generic CTS Framework . . . . . . . . . . . . . . . . . . . . . . 78
5.2.2 Consensus Energy Model . . . . . . . . . . . . . . . . . . . . . . 79
5.3 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . 79
5.3.1 Problem Analysis . . . . . . . . . . . . . . . . . . . . . . . . . 79
5.3.2 Problem Objective . . . . . . . . . . . . . . . . . . . . . . . . . 80
5.3.3 Intractability of CDSDBT problem . . . . . . . . . . . . . . . . . 81
5.4 Proposed GA based Strategy for CDSDBT problem . . . . . . . . . . . . 82
5.4.1 Chromosome encoding . . . . . . . . . . . . . . . . . . . . . . . 82
5.4.2 Initial Population Generation . . . . . . . . . . . . . . . . . . . . 83
5.4.3 Fitness Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . 83
5.4.4 Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.4.5 Crossover . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.4.6 Mutation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.5 Simulation Results & Discussion . . . . . . . . . . . . . . . . . . . . . 85
5.5.1 Evaluation of the proposed GACDBT strategy . . . . . . . . . . . 86
5.5.2 Independent evaluation of CTS algorithms on proposed GACDBT Strategy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
5.5.3 Comparative evaluation of CTS algorithms on proposed GACDBT strategy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
5.5.4 Scalability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
5.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99

**6 Topological Optimization Strategy for Consensus Time Synchronization Algorithms on sparse topology: A Random Weighted Genetic Algorithm based Approach 100**
6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
6.2 System Model & Definitions . . . . . . . . . . . . . . . . . . . . . . 101
6.2.1 Network Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 101

6.3 Problem Formulation . . . 102
6.3.1 A Priori Analysis . . . 102
6.3.2 Problem Objective . . . 103
6.3.3 Motivating Example . . . 104
6.4 Multi Objective Approach . . . 104
6.4.1 Preliminaries & Background . . . 105
6.4.2 Overview of RWGA . . . 106
6.5 Proposed RWGA based Strategy for CDSDBT Problem . . . 106
6.5.1 Chromosome encoding . . . 107
6.5.2 Initial Population Generation . . . 108
6.5.3 Fitness Evaluation . . . 108
6.5.4 Selection . . . 109
6.5.5 Crossover . . . 110
6.5.6 Mutation . . . 110
6.5.7 Termination Criteria & User Selection . . . 111
6.6 Simulation Results & Discussion . . . 111
6.6.1 Performance Evaluation of the Proposed RWGA based Strategy . . 112
6.6.2 Independent evaluation of CTS algorithms on the proposed RGACDBT Strategy . . . 113
6.6.3 Comparative evaluation of CTS algorithms on RWGA based Strategy 121
6.6.4 Scalability . . . 122
6.7 Summary . . . 126
**7 Conclusion & Future Scope** **128**
7.1 Conclusion . . . 128
7.2 Future Scope . . . 130
**Bibliography** **131**
**Dissemination** **139**

# List of Figures

1.1 Behavior of fast, perfect & slow clock w. r. to real time [1] . . . . . . . . . 3

2.1 Offset delay estimation method [2] . . . . . . . . . . . . . . . . . . . . . . 14
2.2 Data triples plotted with the local time of $P_j$ on the X-axis and the local time of $P_i$ on the Y-axis [3] . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.3 Survey tree on time synchronization methods in WSNs . . . . . . . . . . . 19
2.4 Pairwise Broadcast Synchronization [4] . . . . . . . . . . . . . . . . . . . 24

3.1 Message Passing in SATS . . . . . . . . . . . . . . . . . . . . . . . . . . 36
3.2 (a)Timestamps exchange between node i and node j, (b) Relative clock skew estimation of node‘i’ w. r. to node ‘j’ . . . . . . . . . . . . . . . . . . . . . 39
3.3 Skew convergence of (a) ATSP, (b) CCS, and (c) SATS . . . . . . . . . . . 47
3.4 Offset convergence of (a) ATSP, (b) CCS, and (c) SATS . . . . . . . . . . . 48
3.5 Average global synchronization error of ATSP, CCS, and SATS . . . . . . . 49
3.6 Average local error of individual node for (a) ATSP, (b) CCS and (c) SATS 50
3.7 Average number of exchanged messages for ATSP, CCS and SATS . . . . . 51
3.8 Average energy consumption Vs No.of nodes . . . . . . . . . . . . . . . . 51
3.9 Average number of iterations Vs No.of nodes . . . . . . . . . . . . . . . . 51
3.10 Average Mean Square Error Vs No.of nodes . . . . . . . . . . . . . . . . . 51
3.11 Average MSE Vs No.of nodes for dense topology . . . . . . . . . . . . . . 52
3.12 Average MSE Vs No.of nodes for sparse topology . . . . . . . . . . . . . . 52

4.1 Effect of 1-hop averaging Vs. 2-hop averaging . . . . . . . . . . . . . . . 56
4.2 Message passing model for multi-hop SATS . . . . . . . . . . . . . . . . . 61
4.3 An example network of 16 nodes . . . . . . . . . . . . . . . . . . . . . . . 63
4.4 The 2-hop sync tree rooted at ‘A’ of Fig. 4.3 . . . . . . . . . . . . . . . . . 63
4.5 Skew convergence of (a) ATSP, (b) CCS, (c) 1-hop SATS, and (d) 2-hop SATS 68
4.6 Offset convergence of (a) ATSP, (b) CCS, (c) 1-hop SATS, and (d) 2-hop SATS 69
4.7 Average global synchronization error Vs. Iteration number . . . . . . . . . 70
4.8 Average local synchronization error of (a) ATSP, (b) CCS, (c) 1-hop SATS, and (d) 2-hop SATS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
4.9 Avg. no. of messages Vs. No. of nodes . . . . . . . . . . . . . . . . . . . 71
4.10 Avg. energy consumption Vs. No. of nodes . . . . . . . . . . . . . . . . . 71

4.11 Avg. MSE Vs No. of nodes . . . . . . . . . . . . . . . . . . . . . . . . . 72
4.12 Avg. iterations Vs No. of nodes . . . . . . . . . . . . . . . . . . . . . . 72
4.13 Mean of MSE Vs. No. of nodes . . . . . . . . . . . . . . . . . . . . . . 73
4.14 Standard deviation of MSE Vs. No. of nodes . . . . . . . . . . . . . . . . 73
4.15 End-to-end delay Vs No. of hop . . . . . . . . . . . . . . . . . . . . . . 74
4.16 Avg. global sync. error Vs Iteration number . . . . . . . . . . . . . . . . 74
4.17 Impact of hop count on convergence of multi-hop SATS . . . . . . . . . . 74

5.1 2D hypercube and its CDS based delay balanced topology . . . . . . . . . 81
5.2 (a) WSN topology of 8 nodes, (b) Valid Chromosome, and (c) Invalid Chromosome . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
5.3 (a) Chromosome before mutation, (b) Chromosome after mutation, (c) topology corresponds to chromosome (a), and (d) topology corresponds to chromosome (b) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
5.4 Convergence of Proposed GA with topology of 100, 200 and 300 nodes . . 86
5.5 Delay Distribution at Individual Node using Different Topological Strategies 87
5.6 Offset convergence of ATSP on different topological strategies . . . . . . . 88
5.7 Average global synchronization error of ATSP on different topological strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
5.8 Average local synchronization error of ATSP on different topological strategies 90
5.9 Offset convergence of CCS on different topological strategies . . . . . . . . 90
5.10 Average global synchronization error of CCS on different topological strategies 91
5.11 Average local synchronization error of CCS on different topological strategies 92
5.12 Offset convergence of SATS on different topological strategies . . . . . . . 93
5.13 Average global synchronization error of SATS on different topological strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
5.14 Average local synchronization error of SATS on different topological strategies 94
5.15 Offset convergence of (a) ATSP (b) CCS, and (c) SATS on GACDBT . . . 96
5.16 Average local synchronization error of (a) ATSP, (b) CCS, and (c) SATS on GACDBT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
5.17 Average global synchronization error of ATSP, CCS, and SATS on GACDBT 98

6.1 Trade-off between Avg. degree of connectivity and Avg. Euclidean Distance in Random Topology . . . . . . . . . . . . . . . . . . . . . . . . . 103
6.2 WSN topology of 8 nodes . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.3 MCDS based sensor topology for Fig. 6.2 . . . . . . . . . . . . . . . . . 104
6.4 (a) Scenario-1: CDS based optimized topology, (b) Scenario-2: CDS based optimized topology, (c) Scenario-3: CDS based optimized topology . . . . 105
6.5 Flow chart of RWGA method . . . . . . . . . . . . . . . . . . . . . . . . 107
6.6 Chromosome Encoding . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

6.7 Illustrative Examples:(a) Valid chromosome (b) Invalid Chromosome . . . 108
6.8 (a) Chromosome before mutation, (b) Chromosome after mutation . . . . . 110
6.9 Optimization of Objective Functions using the Proposed RWGA Strategy . 112
6.10 Feasible Solutions and Pareto Optimal Solutions generated using the proposed RWGA strategy . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
6.11 Delay Distribution at Individual Node using Different Topological Strategies 113
6.12 Convergence Behavior of ATSP algorithm on different Topological Strategies 115
6.13 Average global synchronization error of ATSP algorithm on different Topological Strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.14 Average local synchronization error of ATSP algorithm on different Topological Strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
6.15 Convergence Behavior of CCS algorithm on different Topological Strategies 118
6.16 Average global synchronization error of CCS algorithm on different Topological Strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
6.17 Average local synchronization error of CCS algorithm on different Topological Strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.18 Convergence Behavior of SATS algorithm on different Topological Strategies 121
6.19 Average global synchronization error of SATS algorithm on different Topological Strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . 122
6.20 Average local synchronization error of SATS algorithm on different Topological Strategies . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
6.21 Convergence Behavior of CTS algorithms on RGACDBT based Strategy . 124
6.22 Average local synchronization error of CTS algorithms on RGACDBT based Strategy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
6.23 Average global synchronization error of CTS algorithms on RGACDBT based Strategy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126

# List of Tables

2.1 Quantitative & Qualitative Analysis of State-of-the-art Synchronization Algorithms . . . . . 29

3.1 Comparison of Message Complexity . . . . . 44
3.2 Simulation Parameters . . . . . 45

4.1 Simulation Parameters . . . . . 68

5.1 SP nodes with possible neighbor SI nodes . . . . . 84
5.2 Simulation Parameters . . . . . 86
5.3 Scalability of Delay Analysis on Different Topological Strategies . . . . . 87
5.4 Average number of messages & energy consumption of ATSP on different topological strategies for 50 nodes . . . . . 89
5.5 Average number of messages & energy consumption of CCS on different topological strategies for 50 nodes . . . . . 92
5.6 Average number of messages & energy consumption of SATS on different topological strategies for 50 nodes . . . . . 94
5.7 Performance of CTS algorithms on different topological strategies for network of 100 nodes . . . . . 96
5.8 Performance of CTS algorithms on different topological strategies for network of 200 nodes . . . . . 97
5.9 Performance of CTS algorithms on different topological strategies for network of 300 nodes . . . . . 98
5.10 Performance of CTS algorithms on different topological strategies for network of 400 nodes . . . . . 98
5.11 Performance of CTS algorithms on different topological strategies for network of 500 nodes . . . . . 99

6.1 Objective functions for different topology selections . . . . . 105
6.2 SP nodes with possible neighbor SI nodes . . . . . 109
6.3 Scalability of Delay Analysis on Different Topological Strategies . . . . . 114
6.4 Simulation Parameters . . . . . 114

6.5 Average number of messages & energy consumption of ATSP on different topological strategies for 50 nodes . . . . . . . . . . . . . . . . . . . . . . . 116
6.6 Average number of messages & energy consumption of CCS on different topological strategies for 50 nodes . . . . . . . . . . . . . . . . . . . . . . . 118
6.7 Average number of messages & energy consumption of SATS on different topological strategies for 50 nodes . . . . . . . . . . . . . . . . . . . . . . . 120
6.8 Performance comparison of CTS algorithms on sparse network of 100 nodes 123
6.9 Performance comparison of CTS algorithms on sparse network of 150 nodes 124
6.10 Performance comparison of CTS algorithms on sparse network of 200 nodes 125
6.11 Performance comparison of CTS algorithms on sparse network of 250 nodes 126

# List of Abbreviations

| | |
|---|---|
| **WSNs** | Wireless Sensor Networks |
| **GPS** | Global Positioning System |
| **NTP** | Network Time Protocol |
| **PROWLER** | PRObabilistic WireLess nEtwork simulatoR |
| **SATS** | Selective Average Time Synchronization |
| **ATSP** | Average Time Synchronization by Pairwise-averaging |
| **CCS** | Consensus Clock Synchronization |
| **GA** | Genetic Algorithm |
| **RWGA** | Random Weighted Genetic Algorithm |
| **TTP** | Time Transmission Protocol |
| **UTC** | Universal Coordinated Time |
| **RBS** | Reference Broadcast Synchronization |
| **SRS** | Sender-to-Receiver Synchronization |
| **RRS** | Receiver-to-Receiver Synchronization |
| **ROS** | Receiver Only Synchronization |
| **PBS** | Pairwise Broadcast Synchronization |
| **TPSN** | Time synchronization Protocol for Sensor Network |
| **TDP** | Time Diffusion Protocol |
| **FTSP** | Flooding Time Synchronization Protocol |
| $R^4$**Sync** | Relative Reference-less Receiver-Receiver Synchronization |
| **CWA** | Confidence Weighted Average |
| **CMA** | Cumulative Moving Average |
| **MTS** | Maximum consensus Time Synchronization |
| **CCTS** | Clustered Consensus Time Synchronization |
| **GGE** | Greedy Gossip with Eavesdropping |
| **CDS** | Connected Dominating Set |
| **MCDS** | Minimum Connected Dominating Set |
| **CDSDBT** | CDS based Delay Balanced Topology |
| **GACDBT** | GA-Connected dominating set based Delay Balanced Topology |
| **RGACDBT** | Random weighted GA-Connected dominating set based Delay Balanced Topology |

| | |
|---|---|
| **SI node** | Synchronization Initiating node |
| **SP node** | Synchronization Participating node |
| **LBCDS** | Load Balanced CDS |
| **PPM** | Parts Per Million |
| **MLE** | Maximum Likelihood Estimator |
| **FWA** | Forwards Weighted Average |
| **WMTS** | Weighted Maximum Time Synchronization |
| **SMTS** | Secured Maximum Time Synchronization |
| **RMTS** | Revised Maximum Time Synchronization |

# List of Symbols

| | |
|---|---|
| $n$ | Number of nodes in the network |
| $C_i(t)$ | Clock value of node $i$ at time-stamp $t$ |
| $\omega$ | Angular frequency of clock |
| $\alpha$ | Clock skew |
| $\beta$ | Clock offset |
| $G$ | Random connected graph |
| $V$ | Set of vertices of graph $G$ |
| $E$ | Set of edges of graph $G$ |
| $K$ | Average degree of connectivity |
| $r$ | radius of connectivity |
| $N_i$ | Neighbor set of node $i$ |
| $P_{tx}$ | Energy consumption for message transmission |
| $P_{rx}$ | Energy consumption for message reception |
| $l(i,j)$ | Euclidean distance between node $i$ and $j$ |
| $M$ | Message Size |
| $\zeta$ | Path loss exponent |
| $\epsilon$ | Acceptable synchronization error |
| $\boldsymbol{C}(t_k)$ | Clock vector |
| $\boldsymbol{M}_k$ | Compensation Matrix |
| $Er(t)$ | Synchronization error function |
| $Deg_i$ | Degree of node i |
| $\tau$ | Sparsity factor |
| $\beta_1$ | Energy dissipated by transmitter module |
| $\beta_2$ | Energy dissipated by transmitter amplifier |
| $\gamma$ | Energy dissipated by receiver module |
| $\lambda_M$ | Largest eigen value |
| $L$ | Side of square deployed area |
| $D_{Th}$ | Threshold delay |
| $\alpha_m^i$ | Skew of node i w. r. to its m-hop neighbors |
| $\beta_m^i$ | Offset of node i w. r. to its m-hop neighbors |
| $\delta$ | one-hop delay |

|  |  |
|---|---|
| $PHY_{delay}$ | Physical layer delay |
| $MAC_{delay}$ | MAC layer delay |
| $\mu$ | Average degree of connectivity |

# Chapter 1

# Introduction

Time measurement has aroused the curiosity among research community from a long time. In the 17th century, Galileo and Christian Huygens were the first scientists to pioneer the work in this field [5]. They have developed an accurate scientific clock based on their theories of the motion of a pendulum. Further advancement in this area includes the development of the atomic clocks that are used in various applications, e.g., the Global Positioning System (GPS), digital telephone communication network, to provide more accurate time information [6]. Furthermore, it is also essential that different nodes in the network should agree on a common time. Therefore, time agreement in the network is a prerequisite, which helps successful working of other protocols and applications. The mechanism to provide the common notion of time across all the nodes in the network is called time synchronization. To achieve time synchronization, it is required that the nodes in the network should communicate through the communication links. Those links can be wired or wireless links. Our research is aimed at the latter one because the inherent challenges are more in wireless networks.

## 1.1 Introduction

Among various wireless networks, Wireless Sensor Networks (WSNs) have received recent scientific attention due to their ease of deployment and broad area of applications, starting from terrestrial to underwater scenarios [1]. WSNs consist of small and cheap sensor nodes. It is a resource constrained distributed network, consisting of large-scale of sensors with limited battery power, short communication range, low bandwidth, and limited processing and storage capability. In recent past, WSNs have witnessed many applications such as environmental monitoring, target tracking, event detection, security and target localization. For all these applications, time synchronization is an indispensable component. It is also required for correctness of other protocols like TDMA, duty cycle scheduling, fault detection and diagnosis and routing.

Time synchronization has remained one of the basic challenges in traditional distributed system due to lack of global physical clock. It is also a well-studied problem in wired network. The Network Time Protocol (NTP) has remained the *de-facto* synchronization protocol in the Internet. But the protocols designed for traditional distributed and wired system do not suit for WSN because of the following reasons.

Firstly, sensors are battery operated and hence, limited energy is available. But accurate time synchronization requires a series of message exchanges which consume much energy. Secondly, the sensor networks are subjected to high degree of failures (nodes/links) because of lack of infrastructure and in some scenario like underwater, due to the mobility of nodes and dynamic topology. So, synchronization cannot be guaranteed all the time. Thirdly, sensor nodes' clocks are made up of a cheap crystal oscillator. So the clock's frequency drift is very high which needs some hardware level calibration for better precision which is out of scope of this thesis.

Therefore, a trade-off always exists between different aspects of synchronization schemes for wireless sensor network. There has been extensive research on time synchronization in wireless networks during the last few years. Several surveys [1, 7–10] have been written about this issue. Still, designing efficient time synchronization algorithms for wireless sensor network is a burning issue in the research community.

The rest of this Chapter is organized as follows. A preliminary about definitions of a clock and its model for sensor nodes is presented in Section 1.1.1. Section 1.1.2 discusses the performance metrics used to evaluate synchronization algorithms. Section 1.3 presents the motivation for the proposed works. Then the research objective is given in Section 1.4 and finally, the major contribution and organization of the thesis are discussed in Section 1.5 and 1.6 respectively.

### 1.1.1 Preliminaries

#### (i) Hardware Clock

The clock component of a sensor node consists of an oscillator of specified frequency, and a counter register. Oscillators embedded in micro-controllers are of four types, viz. crystal oscillators, ceramic resonators, RC oscillators, and silicon oscillators [11]. Since crystal oscillators provide more accuracy with lower cost, all commercially available sensor motes use crystal oscillators in their timer circuitry. The cost of a crystal oscillator is proportional to the accuracy of the clock it provides, and therefore, low-cost sensor motes generally use less accurate crystal oscillator [12].

The hardware clock available in sensor motes are of two types, viz. internal hardware clock and external on-board clock. The internal hardware clock is embedded inside the micro-controller. The crystal oscillators used in the internal hardware clocks have lower frequency stability. Typically, for Tmote Sky sensor nodes, the micro controller MSP430 is embedded with a digitally controlled oscillator running at 8MHz [13]. So, the clock resolution is 1/8MHz=0.125 $\mu$s which is quite low. This value is also known as one tick. Also, the internal clock is switched off when the CPU is in sleep mode. Therefore, internal clock are generally not used for many applications.

Hence, to provide stable timing service for long interval, the external on-board hardware clock must be used. Moreover, the external clock remains active when the CPU is in a sleep state. To read the real time of hardware clock, each hardware clock is associated with a counter register. The counter register is incremented after a certain number of oscillator pulses. The software clock module only extracts the value of the counter register and uses this value as the clock time of the sensor node.

### (ii) Software Clock

The external hardware clock which counts an approximation of real time *'t'*, can be mathematically expressed as [14]:

$$C(t) = k \int_0^t \omega\ (t)\ dt + C\ (t_0) \tag{1.1}$$

where $\omega(t)$ is the angular frequency of the oscillator, *'k'* is a proportionality coefficient and $C\ (t_0)$ is the initial value of the clock. For an ideal clock, $\frac{dC}{dt} = 1$. But for various reasons like temperature, vibration, magnetic field, aging effect of the quartz oscillator, the angular frequency of the oscillator varies, and the clock drifts. As shown in Fig.1.1, if $\frac{dC}{dt} < 1$, the clock is treated as slower clock and if $\frac{dC}{dt} > 1$, it is treated as a faster clock. If the angular frequency can be approximated to a fixed value, then for a node *'i'*, the clock can be expressed as :

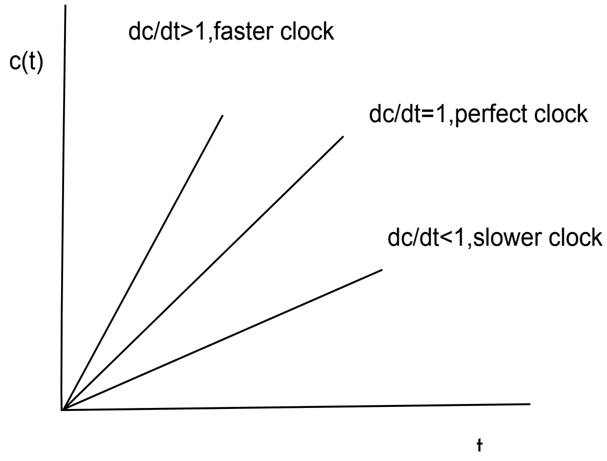


Figure 1.1: Behavior of fast, perfect & slow clock w. r. to real time [1]

$$C_i(t) = a_i(t) + b_i \tag{1.2}$$

where $a_i$=clock skew and $b_i$ =clock offset.

Skew is defined as the rate or frequency of the clock and offset is defined as the deviation from the real time. To compare the local clock of one node *'i'* with relative to another node *'j'* , the above expression can be rewritten as:

$$C_i(t) = a_{ij}(t) + b_{ij} \tag{1.3}$$

where $a_{ij}$=relative skew and $b_{ij}$ =relative offset. If the two nodes are synchronized, then $a_{ij}$=1 and $b_{ij}$=0.

### (iii) Synchronization Error

Let $C_i(t)$ be the software or logical clock time obtained from the physical clock time *'t'* of the node *'i'* using Equation 1.2. Then, the synchronization error between clocks of node *'i'* and *'j'* at real time *'t'* is defined as $|C_i(t) - C_j(t)|$. Thus, the average synchronization error is the average of clock difference between every pair of nodes in the network. At real time *'t'*, the average synchronization error in a WSN having *'n'* number of nodes is defined as $\frac{2}{n(n-1)} \sum |C_i(t) - C_j(t)|, \forall i, i \neq j$.

The synchronization problem in a network of *'n'* nodes is defined as equalizing $C_i(t)$, $\forall$ *i*= 1, 2, 3,..., *n* or some nodes which are neighbors to each other or take part in a communication process. But there lie some limits to achieve ideal synchronization in WSN [15], mostly because of uncertainty in communication delay, mobility, link failures, etc. So, most of the algorithms try to achieve synchronization at least asymptotically. Some algorithms deal with equalizing drifts, some with offset and some with both drift and offset. Equalizing both will help in achieving long-term synchronization [14]. Otherwise, the synchronization process has to be repeated at a regular interval of time to keep the whole network synchronized. Precisely, the basic objective of any time synchronization algorithm is to ensure that the average synchronization error at any real time *'t'* is less than the maximum acceptable synchronization error. The maximum acceptable synchronization error is application dependent. For time-critical applications, it is in the scale of $\mu$s. The time synchronization algorithm also needs to ensure that the logical clock should be monotonic in nature.

### (iv) Sources of Synchronization Error

Time synchronization in WSN is generally achieved by a series of message exchanges. The message transmission suffers broadly two types of delays; fixed delay and variable delay. The fixed delay is due to message preparation, MAC access whereas the variable delay is due to message transmission. The variable delay is the main source of synchronization error in large network. But it can be neglected in a small network where communication takes place with neighboring nodes. The followings are the detailed classification of delay factors which are the major sources of synchronization error [1].

**(a) Send Time:** This is the time required to prepare a message at the operating system level and to send it to the network layer. It is non-deterministic in nature.

**(b) Access Time:** his is the delay incurred at MAC layer to get the transmission channel.

It is specific to the MAC scheme employed by the protocol. For example, in TDMA based protocol, it should wait for a specific time slot to transmit message. In CSMA/CA based protocol, it must wait until the channel is clear.

**(c) Transmission/Reception Time:** This is the time taken by the sender or receiver to send or receive the message bit-by-bit at the physical layer. It is deterministic in nature because it depends on packet size and data rate of the channel.

**(d) Propagation Time:** This is the time required for the propagation of the message between network interfaces of the sender and receiver. It is quite negligible for short range communication as in wireless sensor network.

**(e) Receive Time:** This is the time required for the network interface at receiver to receive and transfer it to the host. It is also non-deterministic in nature as send time.

Many time synchronization algorithms in WSNs follow the usage of MAC layer time-stamping to reduce delay uncertainty [16–19]. Time-stamping of a synchronization packet is done at MAC layer just before the transmission begins and immediately after the packet is received at the MAC layer. As a consequence, the transmission delay would only consist of transmission time, propagation time and reception time, which are quite deterministic in nature. Our work also assumes MAC layer time-stamping to minimize delay uncertainty.

### 1.1.2 Performance Metrics

Different applications put different demands on clock synchronization scheme. For instance, some applications need high accuracy, e.g., TDMA, while some applications need energy efficiency, e.g., power management in WSNs. Further, though there exist a rich set of performance metrics in the literature, they face trade-offs, e.g., synchronization accuracy versus energy efficiency, scalability versus robustness, etc. So an algorithm may not satisfy all the requirements but always tries to optimize it. The followings are the broad set of metrics for WSN, which can be used to evaluate any synchronization algorithm [1].

**(a) Energy Efficiency:** WSNs are generally battery operated and hence, a limited energy source is available. The major cause of energy depletion is the exchange of messages. So, while designing any synchronization algorithm, it should aim for minimizing number transmissions (minimizing the number of message exchanges) over the network.

**(b) Scalability:** Scalability demands the accuracy of the synchronization algorithm must be preserved as the network grows in size or density.

**(c) Precision:** Synchronization precision or accuracy requirement varies according to the type of applications. For some applications, a simple ordering of events and messages is sufficient, e.g., some monitoring applications. Whereas, some time-critical applications require precision in the order of microseconds, e.g., body area sensor network for surgical purposes.

**(d) Robustness:** Robustness indicates the fault tolerance aspect of the protocol. In hostile environment, some sensor nodes may not participate in the synchronization process or some link failure may occur due to short range of radio wave. In such scenario, the synchronization algorithm should continue to operate with desirable accuracy.

**(e) Lifetime:** Lifetime is the period up to which the nodes are remained synchronized. If the protocol synchronizes both skew and offset, then the lifetime will be increased.

**(f) Scope:** The synchronization algorithm may provide network-wide synchronization or synchronization within a subset of neighboring nodes. Because of the scalability issue, network-wide synchronization is difficult to achieve in a large sensor network.

**(g) Cost and Size:** WSNs are generally made up of cheap sensor nodes with limited energy resources. So, deploying costlier and large hardware like GPS to achieve external synchronization is not desirable in WSN.

## 1.2 Motivation

Time synchronization is a fundamental challenge to any distributed system because of the absence of a centralized clock. Being a distributed system, WSNs also face the same challenge. A rich set of time synchronization algorithms has been proposed in the literature for traditional wired networks as well as wireless sensor networks in the near past. Being distributed systems, both wired and wireless networks carry some common characteristics. But, time synchronization issues and mechanism in WSNs are quite different from that in wired networks due to certain fundamental differences between these two types of distributed systems. Hence, time synchronization algorithms designed for wired networks cannot be directly used in WSNs. These differences arise because of the following reasons [1].

**Limited Resources:** Sensor nodes in WSNs have limited energy, bandwidth and processing capability. For wired network, there is no such type of limitations. As a result, time synchronization algorithms for WSNs cannot exploit end-to-end communication between any pair of sensor nodes which are multi-hop away.

In case of direct end-to-end communication, the delay is also quite higher which creates difficulty in designing time synchronization algorithms with high synchronization accuracy. This approach also suffers from consumption of more resources as every sensor node tries to communicate with every other sensor node. Hence, time synchronization algorithms in WSNs essentially rely on one-hop communication. As a result, the protocol like NTP which requires end-to-end communication becomes infeasible on WSNs.

**Nature of Communication:** Another disadvantage of wireless communication is external interference due to problems such as hidden nodes which has to be considered while designing time synchronization algorithms for WSNs. But in the case of wired networks, such interference is almost negligible.

In wireless networks, the electromagnetic signal gets attenuated with the distance and therefore, the quality of reception at each node depends on its distance from the sender. In addition, applications like underwater, communications are established through transmission of acoustic waves. In such applications, issues like limited bandwidth, long propagation delay, and signal fading make traditional time synchronization algorithms infeasible on WSNs. Besides, time synchronization mechanism in WSNs does not permit a synchronization packet to be retransmitted to ensure its reliable delivery because, in case of retransmission, the time-stamp will change which makes the synchronization algorithm inconsistent.

**Type of Deployment:** Some application like environmental monitoring requires sensor nodes to be densely deployed. Also, limited sensing range is another reason for the dense deployment of sensor nodes. In such case, collisions are more likely. Hence, there is usually more possibility of message loss in WSNs as compared to that in wired networks.

On the other hand, in sparse deployment, to achieve network-wide synchronization, sensor nodes need to follow multi-hop communication whose disadvantages are already discussed above. So, type of deployment also indirectly affects synchronization process.

**Dynamic topology:** Wired networks are static in nature. But, in the case of WSNs, the connectivity among sensor nodes changes because of various reasons. For example, in duty-cycled WSNs, the radio module of sensor nodes are switched on and off at regular interval to save energy. So, the link connectivity changes which makes the topology dynamic. As a result, the synchronization process can not be initiated as and when required, thus, making it more challenging.

The above discussion has pointed out the existing challenges in the design of time synchronization algorithm for wireless sensor networks which catalyzes the need to develop efficient time synchronization algorithms for WSNs. For the last decade, a number of synchronization algorithms have been proposed for WSNs. Some of the works [20, 21] are based on synchronizing to a reference node's time by considering a hierarchical backbone for the network. But a common problem in this approach is the root node failure problem. Also, the synchronization error is accumulated along the path from the reference node.

Recently, to develop fully distributed and internal time synchronization mechanism, consensustime synchronization method has gained much attention [17, 18, 22–25]. Consensus Time Synchronization (CTS) is mainly based on distributed average consensus principle [26, 27] which states that all the nodes in a network can converge to a consensus or synchronized state after a finite number of iterations, by communicating and performing averaging only with the neighbors. Because of its simplicity, computational lightness, robustness to node/link failure and purely distributed nature, CTS is more suitable for WSN.

Furthermore, the convergence of consensus-based algorithms depends on the type of averaging schemes they employ and the topological connectivity of the network [28, 29]. This motivates to design average consensus based time synchronization algorithm for WSNs with topological optimization strategy for performance improvement. Based on this criteria of research, the following section highlights the research objectives.

## 1.3 Research Objective

In this thesis, we have aimed at developing new average time synchronization algorithms based on consensus theory to minimize convergence speed, global synchronization error, local synchronization error with low message complexity and scalability. We have also targeted to propose topological optimization strategies to improve the performance of consensus-based time synchronization algorithms. In particular, the objectives of this research are as follows.

1. To propose a distributed, average consensus time synchronization algorithm for dense and single-hop WSNs with better convergence speed, global synchronization error, local synchronization error with low message complexity and scalability as compared to some state-of-the-art CTS algorithms.

2. To propose a distributed, average consensus time synchronization algorithm for sparse, multi-hop WSNS with bounded end-to-end delay without compromising the convergence speed, global synchronization error, local synchronization error with low message complexity and scalability as compared to some state-of-the-art CTS algorithms.

3. To propose a topological optimization strategy for dense and single-hop WSNs to improve the performance of consensus-based time synchronization algorithms in terms of convergence speed, synchronization error, number of messages exchanged and energy consumption.

4. To propose a topological optimization strategy for sparse and multi-hop WSNs to improve the performance of consensus based time synchronization algorithms in terms of convergence speed, synchronization error, number of messages exchanged and energy consumption.

5. To validate the proposed algorithms using PROWLER, a MATLAB based discrete event simulator designed for WSNs [30]. This simulator is chosen because of rapid prototyping feature and better support for optimization problems.

6. To evaluate the algorithms in terms of some standard performance metrics, described in Section 1.1.2.

## 1.4 Major Contribution

In this dissertation, four algorithms are proposed related to average consensus time synchronization problem and topological optimization for performance improvement. The descriptions of the contributory chapters are given below.

- In Chapter 3, an average consensus-based, distributed time synchronization algorithm, named as Selective Average Time Synchronization (SATS), is proposed. The optimality and convergence property of the algorithm is analyzed mathematically and validated through simulation. Simulation results show that the performance of the algorithm is improved as compared to ATSP and CCS algorithms.

- In Chapter 4, an average-consensus based, distributed time synchronization algorithm, named as multi-hop Selective Average Time Synchronization (multi-hop SATS), is proposed for the sparse network using distributed constraint-based dynamic programming approach. Simulation results show that the performance has been improved significantly as compared to ATS, CCS and SATS algorithms by restricting the hop count. The trade-off is also studied through simulation with the increase in the number of hops.

- In Chapter 5, a Genetic Algorithm (GA) based topological optimization strategy for dense topology is proposed to improve the performance of consensus time synchronization algorithms in terms of mean delay, average Mean Square Error (MSE), average number of iterations for consensus convergence and energy consumption.

- In Chapter 6, a multi-objective genetic algorithm scheme, Random Weighted Genetic Algorithm (RWGA), based topological optimization strategy for sparse topology is proposed to improve the performance of consensus time synchronization algorithms in terms of mean delay, average Mean Square Error (MSE), average number of iterations for consensus convergence and energy consumption. The performance of proposed topological optimization strategy is also studied on dynamic topology.

## 1.5 Thesis Outline

- **In Chapter 1**, introduction to WSNs, an overview of hardware clock, clock model, sources of synchronization error and performance metrics for evaluation of time synchronization algorithms are presented. The motivation behind designing distributed, consensus-based time synchronization algorithm is outlined along with the research objectives. The major contributions are highlighted followed by the thesis organization.

- **In Chapter 2**, a comprehensive overview of the related work done by different authors in the area of time synchronization in WSNs is presented. The main focus is given on distributed consensus-based time synchronization algorithms in WSNs.

- **In Chapter 3**, an average consensus-based, distributed time synchronization algorithm, named as Selective Average Time Synchronization (SATS), is proposed.

The optimality and convergence property of the algorithm is analyzed mathematically and validated through simulation. Simulation results show that the performance of the algorithm is improved as compared to ATSP and CCS algorithms.

- **In Chapter 4**, an average consensus-based, distributed time synchronization algorithm, named as multi-hop Selective Average Time Synchronization (multi-hop SATS), is proposed for the sparse network using distributed, constraint-based dynamic programming approach. Simulation results show that the performance has been improved significantly as compared to ATSP, CCS and SATS algorithms by restricting the hop-count. The trade-off is also studied through simulation with the increase in the number of hops.

- **In Chapter 5**, a Genetic Algorithm (GA) based topological optimization strategy for dense topology is proposed to improve the performance of consensus time synchronization algorithms in terms of mean delay, average Mean Square Error (MSE), average number of iterations for consensus convergence and energy consumption.

- **In Chapter 6**, a multi-objective genetic algorithm scheme, Random Weighted Genetic Algorithm (RWGA), based topological optimization strategy for sparse topology is proposed to improve the performance of consensus based synchronization algorithms in terms of mean delay, average Mean Square Error (MSE), average number of iterations for consensus convergence and energy consumption. The performance of proposed topological optimization strategy is also studied on dynamic topology.

- **In Chapter 7**, a brief description of the whole work is presented. It also discusses the improvements and limitations of the results obtained and suggests the future scope of the work done.

## 1.6 Summary

This Chapter provides a brief introduction to WSNs and needs for time synchronization in WSNs. It also meticulously outlines the scope, the motivation, and the objectives of the thesis. A precise presentation of the research work carried out in the whole thesis, and the contribution made in the thesis have also been highlighted. In brief, this chapter provides a complete overview of the whole thesis in a concise manner.

# Chapter 2

# Background & Literature Survey

This Chapter briefly presents the journey of time service from a centralized system to time synchronization methods in traditional distributed system to WSNs. Then, based on the existing literature, a survey tree is presented which helps us to classify the different time synchronization methods available for WSNs. Furthermore, the survey tree enables us to select our research objectives which are pointed out in Chapter 1 and to carry out the research in a particular direction.

## 2.1 Introduction

In centralized systems, synchronization issue does not arise because there is no time ambiguity. A process obtains the time by simply using a system call to the kernel, and the kernel is responsible for providing the time to all processes centrally [1]. When another process requests for time, a higher time value is provided by the kernel. Hence, the events are ordered chronologically without any ambiguity.

In distributed systems, there is no global physical clock. Each node in the system has its internal clock and uses its local time. In practice, these clocks drift from each other in seconds scale, and the errors get accumulated to a reasonable value over time. Also, different clocks have different oscillation frequencies. As a result, they may not always continue in a synchronized state though they might have initially synchronized to each other. This creates problems with the applications that are solely dependent on a synchronized notion of time. So, synchronization has remained an important issue for traditional distributed systems [31].

The rest of the Chapter is organized as follows. Section 2.2 gives a brief background of handling synchronization problem in traditional distributed system followed by a detail descriptions of some well known traditional synchronization methods in Section 2.3. Section 2.4 presents a taxonomy of time synchronization methods available for WSNs. Section 2.5 presents a case study of some recent and state-of-the-art synchronization protocols in WSNs. Section 2.6 highlights the key observations from the literature survey followed by conclusion in Section 2.7.

## 2.2 Background

In traditional distributed system, to handle time synchronization problem, there are two basic methods adopted in the literature [8]. The first method deals with synchronizing the physical clock and the second method deals with synchronizing the logical clock. In physical clock synchronization, the objective is to make the physical clock of each node to agree on a common value; whereas in logical clock synchronization, the requirement is the chronological ordering of relevant events.

Network Time Protocol (NTP) is the most commonly used protocol on the Internet for physical clock synchronization [8]. NTP follows a layered client-server architecture, based on UDP message passing paradigm. It follows a hierarchical architecture, where each level is called as a stratum. The lowest level is called as stratum-1 which contains the primary servers and they are directly synchronized to stratum-0 devices. The stratum-0 contains high-precision timekeeping devices such as atomic (cesium, rubidium) clocks, GPS clocks or other radio clocks. The next level contains the secondary servers which are known as stratum-2, and they get synchronized with the stratum-1 servers. The hierarchy continues up to stratum-15, the highest level in the hierarchy.

In some applications of distributed systems, the logical ordering of events is more important than knowing the actual occurrence time for each event. In such cases, the absolute physical clock synchronization is not necessary. In this context, Lamport [32] and Fidge [33] have proposed two schemes for logical clock synchronization in distributed systems.

Lamport has defined an ordering of events using the principle of causality. If an event affects the outcome of another event, then it is termed as "happened before" event. The partial ordering of events is obtained by the "happened before" relation. For the partial ordering, two rules are defined. The first rule states that the local clock is to be incremented between any two consecutive local events. The second rule states that upon receiving a message from a sender process with a local time-stamp, sets the local clock of the receiver process as greater than or equal to the maximum value between the local clock value and the sender time-stamp. Finally, these logical clocks are used to obtain a total ordering of all events.

Fidge has also proposed a partial ordering of events using the principle of causality. But, instead of using a single time-stamp value, a vector of time-stamp values is used. The vector is initially set to (0, 0, . . . , 0), where each index corresponds to a process. If a local event occurs at a process, the value at that index is incremented. When a process receives a message from another process with time-stamp vector, it sets the time in each index to the maximum value of either the corresponding value of or the local vector value. The advantage of keeping a vector of timestamps and maximizing it among processes is that it allows ordering not only the events within a local process but also the events in other processes in the system.

## 2.3 Traditional Time Synchronization Methods

The above-discussed protocol and methods give the basis of synchronization handling mechanisms in traditional distributed systems. Subsequently, a number of protocols and methods have been proposed in the past few decades to handle the synchronization problem in traditional distributed systems. Some of the representative methods are briefly highlighted below.

### 2.3.1 Remote Clock Reading Method

The remote clock reading method is proposed by Cristian, which considers non-deterministic message delays between processes [34]. This method basically assumes a client-server architecture for the distributed system. When a client process wants a time estimation, it sends a request to the remote server and waits for the server to respond. When the client receives the reply, it calculates the round-trip delay '$rtt$' as the difference between the time at which it has sent the request and the time at which it received the reply. The reply message contains the estimate of the time 't' on the remote server. Upon receiving the reply, it corrects its local clock as $t + rtt/2$. Multiple rounds of message passing are carried to compute and choose the least round-trip delay, or the average of multiple round-trip delays is chosen. This method synchronizes the clients with the remote server which is connected to an accurate time service (Universal Coordinated Time).

The drawbacks of remote clock reading method are: (i) sending time uncertainty due to network traffic and routing (ii) high message complexity and (iii) no definitive way to decide the number of multiple rounds to be performed to find out the exact round-trip delay.

### 2.3.2 Time Transmission Method

This method is named as Time Transmission Protocol (TTP) and is proposed by Arvind [35]. The method states that a node communicates its own clock time to a target node. The target node, upon receiving the message from the source node, computes the time in the source node and the delay statistics by using the timestamps appended in the message. The algorithm is briefly described below.

Assume that 'S' is the source node and 'T' is the target node. 'S' sends a series of synchronization messages to 'T'. The $i^{th}$ message is sent at time $T_i$ of S's clock and received at time $R_i$ of T's clock. 'T' estimates S's time as, $T_{est} = R_n - (R'(n) - T'(n)) + d'$ where $R'(n) = \frac{1}{n}\sum_{i=1}^{n} R_i$ and $T'(n) = \frac{1}{n}\sum_{i=1}^{n} T_i$. $d'$ is the expected value of message delay.$R_n$ is the time at which $n^{th}$ message is received by 'T'. $T_{est}$ is the target's estimate of the time at the source. Once the time on the source is estimated, the target corrects its local clock to achieve synchronization. The drawback of the TTP is its high message complexity.

### 2.3.3 Offset Delay Estimation Method

This is the basic principle used by NTP for its operation [2]. In fact, a source node cannot exactly compute the local time on the target node due to non-deterministic delays between the nodes. This method employs a series of message exchange rounds and chooses the round with the minimum delay. The remote clock reading method, propose by Cristian [34], also follows the same method to compute the message delay.

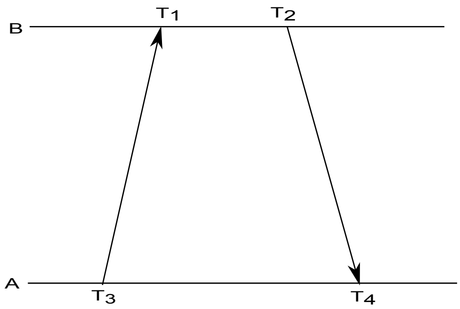


Figure 2.1: Offset delay estimation method [2]

Fig. 2.1 shows that how timestamped messages are exchanged between two nodes A and B. Let $T_1$, $T_2$, $T_3$, and $T_4$ be the most recent timestamps at node A and B. Assuming that clocks of A and B have same oscillation frequency, then, $a = T_1 - T_3$ and $b = T_2 - T_4$. Assuming the transmission delay between A and B is small, the clock offset $\theta$ and round trip delay $\delta$ of B relative to A at time $T_4$ are approximately given as: $\theta = (a + b)/2$ , $\delta = a - b$. Each NTP message contains the recent three timestamps $T_1$, $T_2$ and $T_3$, while $T_4$ is computed upon arrival. Thus, the delay and offset can be calculated independently by both the nodes A and B using a single bi-directional message stream.

Since, this method also follows a series of message exchange rounds like Cristian's method [34], both of them have the same drawback, i.e., high communication overhead in terms of message complexity. However, the accuracy of this method is better than Cristian's protocol because delays are partly compensated.

### 2.3.4 Model based Method

It is based on set-valued estimation method [3] which works as follows. Let a distributed system have 'N' number of nodes. Let $t_i$ denotes the local time on the clock of node $P_i$. Then, the local times $t_i$ and $t_j$ on two nodes $P_i$ and $P_j$ are related as: $t_i = a_{ij}t_j + b_{ij}$ where $a_{ij}$ and $b_{ij}$ denotes, respectively, the relative skew and offset between the two hardware clocks.

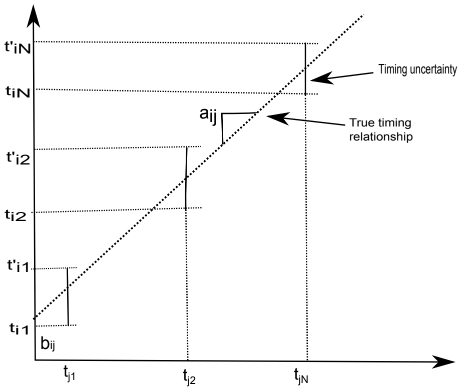


Figure 2.2: Data triples plotted with the local time of $P_j$ on the X-axis and the local time of $P_i$ on the Y-axis [3]

Node $P_i$ sends 'N' number of messages to node $P_j$ at local times $t_{ik}$ for *k* = 1, . . . , N. Node $P_i$ receives replies to these messages from node $P_j$ at times $t'_{ik}$ and each received message is stamped with $t_{ik}$ and the local time, $t_{jk}$, when node $P_j$received the $k_{th}$ message. When node $P_i$ receives the last reply message from node $P_j$ , node $P_i$ has a triplet of timestamps, $(t_{ik}, t_{jk}, t'_{ik})$. Using this triplet, a graph is drawn as shown in Fig. 2.2 with the local time on node $P_j$ on the X-axis and local time on node $P_i$ on the Y-axis. Each data triplet can be plotted as an error bar. The relative drift $a_{ij}$ and the relative offset $b_{ij}$ are computed from the slope and Y-intercept of any line that passes through all of the error bars. The drawback again remains the same, i.e., high message complexity.

The aforesaid traditional synchronization methods are mostly used in a wired network and are not feasible on WSNs because of various reasons which are already discussed in Section 1.2 of Chapter 1. Therefore, to cope up with the various issues, in the recent past, a number of time synchronization algorithms are proposed by different authors for WSNs. The following section highlights the classification of different synchronization algorithms for WSNs.

## 2.4 Taxonomy of Time Synchronization Methods in WSNs

In the literature, the classification of synchronization methods is done based on different perspectives [1, 9]. In [1], the synchronization methods are broadly divided into two types, one is related to synchronization issues, and another is based on application-dependent features. The following subsections highlight these two types.

### 2.4.1 Taxonomy based on synchronization issues

**(a) Master-slave versus peer-to-peer synchronization**

Master-slave: This method considers one node as the master and the others are considered as slaves. The slave nodes take the master node's clock reading as the reference time and attempt to synchronize with the master. Some of the example protocols in this class are the protocol proposed by Mock *et al.* [36] and Ping's protocol [37]. The demerit of this method is: high computational resource requirement for the master node.

Peer-to-peer: In this method, any node can communicate directly with any other node in the network. Such approach is tolerant to single-point (master node) failure problem. Therefore, the protocols which are based on this method, are more flexible but also more uncontrollable. RBS [20], protocol by Romer *et al.* [38], protocol by PalChaudhuri *et al.* [39], TDP [40], and the asynchronous diffusion protocol of Li and Rus [23] are based this method.

#### (b) Internal synchronization versus external synchronization

Internal synchronization: A global reference time is not available for the system and therefore, this mechanism attempts to reduce the maximum difference between the values of local clocks of the nodes. The protocol proposed by Mock *et al.* [36] belongs to this category. Internal synchronization can follow both master-slave and peer-to-peer method.

External synchronization: In this type of synchronization, a standard reference time source such as UTC (Universal Coordinated Time) is available. The local clocks of the nodes try to synchronize to this external source. NTP [2] follows external synchronization method. This method of synchronization is feasible where energy is not a constraint like Internet. This method of synchronization can only adapt master-slave mode.

#### (c) Probabilistic versus deterministic synchronization

Probabilistic synchronization: This method provides a probabilistic upper bound on the maximum clock offset with a failure probability that can be bounded or determined. This method is quite expensive in an energy constraint environment. The protocol proposed by PalChaudhuri *et al.* [39] is a probabilistic approach of RBS [20].

Deterministic synchronization: This method ensures a deterministic upper bound on the clock offset. Examples of such protocols are RBS [20] and TDP [40].

#### (d) Sender-to-receiver versus receiver-to-receiver versus receiver-only synchronization

Sender-to-receiver synchronization (SRS): The sender node at regular interval sends a timestamped message to the receiver nodes and then the receivers synchronize with the sender using the time-stamp received from the sender. TPSN [21], Tiny-Sync, and Mini-Sync [41] are based on this approach.

Receiver-to-receiver synchronization (RRS): This method assumes that if any two receivers receive the same message within a single-hop, they receive it approximately at the same time. Then, the receivers exchange the time at which they received the same message and compute their offset based on the difference in reception times and using linear regression. RBS [20] follows this principle of synchronization.

Receiver-only synchronization (ROS): A group of nodes can be simultaneously synchronized by only overhearing the message exchanges of a pair of nodes using the broadcast nature of wireless medium. PBS [4] and multi-hop PBS [42, 43] follow this method of synchronization.

#### (e) Clock correction versus untethered clocks

Clock correction: In practice, most synchronization methods follow the principle of correcting the local clock in each node to run at par with a global reference time scale or an atomic clock. The protocol proposed by Mock *et al.* [36] and Ping's protocol [37] are based on this method. The local clocks of nodes that are present in the network are corrected either instantaneously or at regular interval to keep the entire network synchronized.

Untethered clocks: To get a common notion of time without initiating the synchronization process is becoming an attractive method, because a reasonable amount of energy can be saved by this approach. RBS [20] follows this principle by building a table of parameters that relate the local clock of each node to the local clock of every other node in the network. Local timestamps are then compared using this table. So, a global time scale is preserved while allowing the clocks run untethered. The protocol proposed by Romer *et al.* [38] also uses this principle.

#### (f) Pairwise Synchronization versus network-wide synchronization

Pairwise synchronization: The method is primarily focused on to synchronize a pair of nodes and can be extended to deal with the synchronization of a group of nodes.

Network-wide synchronization: The protocols are mainly designed to synchronize all the nodes present in the network.

### 2.4.2 Taxonomy based on application requirements

#### (a) Single-hop versus multi-hop networks

Single-hop communication: In a single-hop network, a sensor node can directly communicate with any other node in the network. The protocol proposed by Mock *et al.* [36] is an example, following single-hop communication. However, it can be extended to multi-hop communication.

Multi-hop communication: In the case of a large and sparse network, every node is not within the vicinity of every other node. In this case, multi-hop communication can occur as a sequence of hop-wise communication through connected, pairwise sensors. Extension of PBS [42, 43] are examples of such protocols.

#### (b) Stationary networks versus mobile networks

Stationary networks: Sensor nodes are assumed to have fixed geographical locations. Most of the protocols in the literature are designed assuming stationary network.

Mobile networks: In applications like underwater sensor networks, sensors are mobile in nature, and they connect with other sensors only when they are within the communication range of each other. The dynamic topology is often a challenge to design synchronization protocols as it needs resynchronization of nodes and re-computation of the neighborhoods which is an extra computational overhead.

#### (c) MAC-layer-based approach versus standard approach

The protocol like RBS does not depend on MAC protocols so as to avoid a tight integration of the application with the MAC layer. On the other hand, the protocols proposed by Ganeriwal *et al.* [21] and Mock *et al.* [36] rely on the CSMA/CA protocol for the MAC layer. In fact, MAC layer based approaches have the advantage of reduced delay, to have better synchronization accuracy.

### 2.4.3 Taxonomy based on approaches

Based on the context of our research work and the exhaustive survey carried out, one broader way of classifying the synchronization methods is shown in Fig. 2.3. They are mainly categorized into two types, viz., (a) Non-consensus approaches and (b) Consensus approaches. The details about each of the approach are explained below.

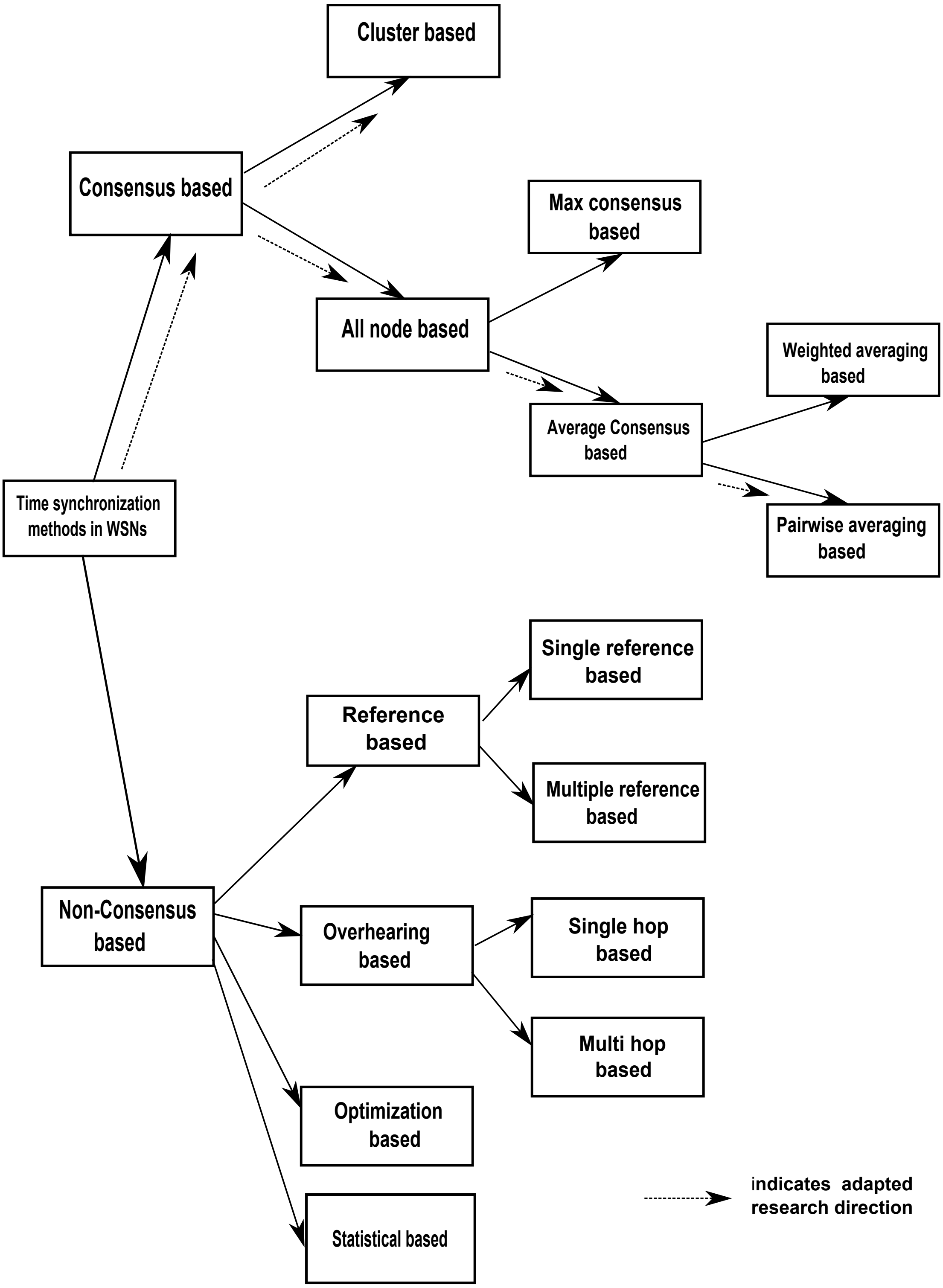


Figure 2.3: Survey tree on time synchronization methods in WSNs

### (a) Non-consensus based approach

The algorithms under this category are not purely distributed or semi-distributed. Either they refer to an external or internal node for synchronization. The reference node may be one node or more than one. Accordingly, they are further divided into following types.

### (i) Reference based approach

In this approach, one or more reference nodes are used to achieve synchronization. They follow sender-to-receiver synchronization (SRS) or receiver-to-receiver synchronization (RRS) principle. TPSN [21] and FTSP [44] are single reference, and SRS based synchronization protocols. RBS [20] is single reference based and $R^4$Sync [16] is multiple reference based, and both of them follow RRS mechanism.

### (ii) Overhearing based approach

The protocols under this approach follow receiver-only synchronization (ROS) principle. Nodes in the network get synchronized by overhearing the synchronization packets, exchanged between two other nodes within their vicinity. PBS [4] and its variants [42, 43] follow this principle.

### (iii) Optimization based approach

Some optimization methods have been applied to existing protocols to improve the performance. In [45], Ant Colony Optimization (ACO) is applied on Time synchronization Protocol for Sensor Network (TPSN), to minimize the synchronization error and to increase the precision of time synchronization. The authors have claimed that the precision is improved by nearly 10%. It removes the accidental error and enhances the convergence speed. It is also claimed that the synchronization precision is around 2-8$\mu$s in a 5-hop range. In [46], optimal foraging theory is applied on Reference Broadcast Synchronization (RBS) protocol to minimize redundant message exchanges and to maximize the lifetime of the network. The authors have claimed that the time spent in synchronization is reduced from $O\ (n^m)$ to $O\ (nm)$ where *'n'* is the number of nodes in the network and *'m'* is the number of synchronization packets. The protocol also minimizes the storage space for each sensor node, and hence, saves energy.

### (iv) Statistical approach

Classical tools from mathematics and statistics have been applied in some time synchronization protocols for WSN. These protocols are mainly based on parameter estimation methods to perform time synchronization. The most common methods are least square method [47], Bayesian estimation method [48, 49], Kalman filter method [50–52], linear regression method [53], maximum likelihood estimator (MLE) [4], and Crammers-Rao upper bound rule [16].

### (b) Consensus based approach

In recent years, consensus approach from control theory has been widely implemented in many problems of computer science, e.g., peer-to-peer network [54], load balancing in distributed system and sensor network [55]. Consensus problem is the problem of making the scalar states of a set of agents converge to the same value using local communication [26, 27].

Among different types of consensus (max value consensus, min value consensus, and average consensus), the average consensus has gained more popularity because of its feasibility in many applications [56]. The average consensus principle is the mostly adapted principle in recent time synchronization algorithms for wireless sensor network. Some of the recent and state-of-the-art consensus based time synchronization algorithms are presented below. They are mainly categorized into two types: (i) all node based consensus algorithms and (ii) cluster based consensus algorithms.

#### (i) All node based approach

In this approach, every node in the network participates in the consensus seeking process by communicating with the neighboring nodes. Different authors have proposed different averaging schemes for faster consensus convergence. Based on the averaging schemes, the approaches can be further divided into two types, viz., (i) weighted averaging scheme and (ii) pairwise averaging scheme.

In [17], the authors have proposed three weighted averaging consensus methods for clock synchronization, namely, Cumulative Moving Average (CMA), Forwards Weighted Average (FWA), and Confidence Weighted Average (CWA). Using a network of 100 nodes and random linear drift, the authors claimed that CWA proved most reliable. Also, FWA performance is same as CWA and has the advantage of reduced computational complexity.

The synchronization algorithm in [18] is also based on weighted averaging method. It uses cascading of two consensus algorithm, one for skew compensation and another for offset compensation. The communication protocol used is pseudo-periodic broadcast. The algorithm is claimed to be fully distributed, asynchronous, and computationally light.

In [23], four protocols are proposed, namely, all-node-based method, cluster-based method, diffusion-based method, and fault tolerant diffusion-based method. The first two methods require a node to initiate the synchronization process. So, these are not fault-tolerant and localized. The last two are based on local communication and can achieve the average consensus. These protocols are also analyzed in the presence of a byzantine fault and claimed to be fault tolerant.

The authors in [24] presented a pure average consensus-based synchronization algorithm. It is claimed that the proposed algorithm is fully distributed, asynchronous, includes skew compensation and computationally light. It is also robust to dynamic topology. Similarly, in [25], an average synchronization algorithm is proposed with non-linear dynamical network and with random time delays.

### (ii) Cluster based approach

In [57], the authors have proposed a cluster consensus-based time synchronization method. The basic objective of embedding clustering into time synchronization is to minimize energy consumption and to achieve faster convergence.

**Advantages of Consensus-based time synchronization**

From the above-discussed literature, the advantages of consensus-based time synchronization methods can be summarized as:

1. It can work in a distributed way without depending on any hierarchical structure or a node as a reference. So, they are tolerant to single-point failure problem.
2. Neighboring nodes are more accurately synchronized which is a major requirement in most of the WSN applications.
3. As it uses only local communications, no routing is needed and hence, the network congestion is avoided.

The above highlighted key points about consensus-based time synchronization algorithms motivate us to choose our research direction towards this approach. The next Section describes briefly some of the recent and state-of-the-art synchronization algorithms, both from non-consensus and consensus based approaches.

## 2.5 Case study of state-of-the-art synchronization algorithms

In this Section, a comprehensive study is carried out about some representative time synchronization algorithms which are proposed in the recent past. The selected algorithms are: TPSN [21], RBS [20], PBS [4] and its variants [42, 43], $R^4$Sync [16], multi-hop $R^4$Sync [58], Average TimeSync [18], ATSP [22], MTS [19], CCS [17], and CCTS [57].

### (i) Time synchronization Protocol for Sensor Network (TPSN)

It is based on conventional sender-receiver handshake method. The nodes are structured in a hierarchical manner similar to NTP. Nodes are self-organized to act as a server to some node while client to another server. Level-0 nodes are called root node. It can be either external time source or one of the nodes of the network. Periodically root node is selected using some leader election algorithm. All nodes have ID, and each node assumed to know his neighbors. It is also assumed that all nodes have bi-directional links with their neighbors. It follows two phases to achieve synchronization, viz., (i) Level Discovery Phase: In this phase, root level node broadcasts “level discovery” packet to its neighbors. These neighbors assign themselves level-1 and broadcast another packet to its neighbor with their level. This process continues till all neighbors have assign levels. In complex networks, some node may not receive any packet, or it may join the network after level discovery phase is over. Then it will wait for a time-out and then send a level request message to neighbors. If a root node dies, the level 1 node will not get any acknowledgment from it, and they will wait until time-out occurs. After the time-out, they will initiate a leader election algorithm to elect a Root Node. (ii) Synchronization Phase: In this phase, two-way message exchange is done between sender and receiver. Root node initiates by sending a time synchronization packet and after a random time period, level-1 nodes behave as the sender and initiate two-way message exchange. Every level node sends an acknowledgment after it is synchronized with higher level node. The lower level node will overhear this message exchange and after some random back-off time, it initiates two-way message exchange with neighboring higher level nodes.

## (ii) Reference Broadcast Synchronization (RBS)

It is based on receiver-receiver approach. A beacon node is required to synchronize all its neighboring nodes with one another. It exploits the broadcast nature of wireless medium. After receiving the beacon message from the reference node, clients exchange their respective reception times of the beacon message and calculate the relative offset and rate differences with other clients. Then, they transform local time reading into any other client's local timescale. This method uses a series of synchronization messages from a sender to estimate both relative offset and skew of the local clocks of receivers. The algorithm exploits the principle of a time-critical path, i.e., the temporal path of a message that leads to non-deterministic error in the protocol. To eradicate non-deterministic factors, the RBS algorithm uses a sequence of reference messages from the same reference node, rather than a single message. Then, receiver *'$j$'* will estimate its offset relative to any other receiver *'$i$'* as the average of clock differences for each message received by nodes *'$i$'* and *'$j$'* using the formula: *Offset* $[i, j] = \frac{1}{m}\sum_{k=1}^{m}(T_{i,k} - T_{j,k})$.

In the above formula, *'$i$'* and *'$j$'* denote two receivers, *'$m$'* is the number of beacon messages, and $T_{i,k}$ is node *$i$*'s clock when it receives broadcast *'$k$'*.

## (iii) Pair-wise Broadcast Synchronization (PBS)

It is based on both SRS (sender-receiver synchronization) and ROS (receiver only synchronization) approaches to achieve network-wide synchronization. In this approach, a subset of sensor nodes is synchronized by overhearing the timing message which is exchanged between a pair of nodes. The PBS method works as follows:

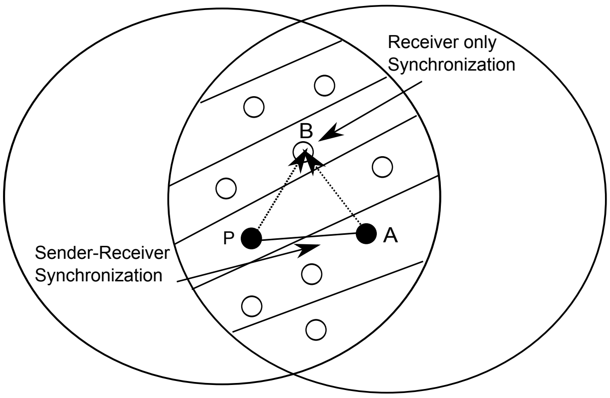


Figure 2.4: Pairwise Broadcast Synchronization [4]

Consider a node, say node *'B'*, in the marked region as shown in Fig. 2.4. While node *'P'* and node *'A'* exchange timestamped messages, Node *'B'* can receive messages from both the nodes. Hence, node *'B'* can observe a set of time readings $\{T_{2,i}^{B}\}_{i=1}^{N}$ at its local clock when it receives packets from Node *'A'*. Similarly, a set of time stamps $\{T_{2,i}^{P}\}_{i=1}^{N}$ can also be obtained by receiving packets from Node *'P'*. Then, linear regression and joint skew estimator technique are applied to synchronize node *'B'* and compensate the effect of the relative clock skew between node *'P'* and node *'A'*. Similarly, the other nodes in the marked region get synchronized with *'P'*.

The applicability of PBS is only limited to the single-hop network where every node is assumed to be within the communication range of the two selected super nodes (*'P'* and *'A'* in the given example). In this context, the extension of PBS for multi-hop synchronization is proposed by Noh *et al.* in [42]. Two algorithms are proposed to achieve network-wide synchronization by reducing the message complexity. The first extension is a centralized algorithm, named as Network-wide Pair Selection (NPS) algorithm which follows the principle of hierarchy creation. Then, it performs PBS by selecting a pair of nodes along the hierarchy with the maximum number of unsynchronized neighbors in their common coverage. Since, the method is dependent on hierarchy creation and selection of synchronization pair with maximum common nodes, it incurs large communication overhead. To alleviate this problem, another extension is proposed which is called Group-wise Pair Selection(GPS) algorithm. In this case, the selection is done locally which reduces the complexity to some extent. But, some unwanted message exchanges are carried out which worsen the performance of GPS algorithm.

Another extension to PBS is proposed by Cheng *et al.* in [43], for multi-hop sensor networks. The authors claimed that to achieve network-wide synchronization, selection of the minimum number of pairs to perform PBS is an NP-complete problem. They proposed a greedy-based, distributed algorithm which performs better than NPS and GPS algorithms. It is also energy efficient and scalable.

### (iv) Relative Reference less Receiver-Receiver Synchronization ($R^4$Sync)

It follows the receiver-to-receiver principle, introduced by Reference Broadcast Synchronization (RBS). It allocates the reference's function to all sensors instead of to only one sensor as done in RBS, to eliminate the single point failure problem. It also piggybacks the time-stamp with the regular signals (beacons) which removes the need of sending separate synchronization signal, thus, saving the energy. The synchronization is ensured by estimating parameters, reflecting relative deviation with respect to every other node which does not require any local clock update. It runs in cycles and the nodes as per their IDs sequentially broadcast beacons. A beacon carries timestamps, reporting local reception times of previous beacons. For a neighborhood of nodes, every beacon would carry (N-1) timestamps. These timestamps are then used by every node as samples to estimate relative synchronization parameters. The analysis is done using joint skew/offset MLE estimators.

A multi-hop extension to $R^4$Sync is proposed by Djamel *et al.* in [58]. For multi-hop synchronization, on-demand synchronization is considered instead of the global always-on model. The objective is not to keep all the nodes synchronized to a common global time, but to provide reactive mechanism permitting nodes to mutually synchronize whenever needed, i.e., on-demand.

### (v) Average TimeSync (ATS)

This protocol uses a combination of two consensus algorithms, one for clock drift and another for clock offset. The whole algorithm proceeds through 3 steps: (i) relative drift estimation (ii) drift compensation, and (iii) offset compensation. The underlying communication protocol is assumed to be fully asynchronous, based on the principle of pseudo-periodic broadcast. In the first step, nodes broadcast local timestamps as per their own clock oscillation period to estimate the clock skew rates relative to each other. To avoid quantization errors, the estimate of parameters are performed through a low-pass filter. Then, nodes broadcast their current estimate of the virtual clock skew rate. At the receiving nodes, this value is combined with their relative skew estimates through weighted sum method to adjust their own virtual clock estimate. The same method is then applied to remove the offset errors. The authors claimed that the offset compensation and skew compensation can be carried out simultaneously for faster convergence.

### (vi) Average Time Synchronization (ATSP)

This method suggests that if the nodes in a network like WSN, have lack of standard crystal oscillators, then compensating the skew and offset of all the nodes to their average is a best method. For the protocol, the local clock of node *'i'* is modeled as $x_i(t) = w_i t + x_i(0)$, *i*=1, 2, ..., *n*, where $w_i$=frequency of crystal oscillator, $x_i(0)$=initial local clock at *t*=0. Assuming all the crystal oscillators of the nodes in the network have identical frequency or can be compensated to an identical frequency, i.e., $w_1 = w_2 = ... = w_n = w$, then after some update operations on the network which update partial or all the nodes' local clocks, the average timestamp of the network at any instance *'t'* can be expressed as $x_{avg-syn}(t) = wt + avg(x(0))$ where $avg(x(0)) = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$ is the average clock of all nodes at time *t*=0. ATS uses pair-wise message exchanges between neighboring nodes to achieve network-wide synchronization.

## (vii) Maximum consensus Time Synchronization (MTS)

In [19], the authors have proposed a maximum consensus based time synchronization protocol. The basic objective of the protocol is to maximize local information to reach global synchronization. The communication protocol is assumed to be pseudo-periodic broadcast which is same as in [18]. The protocol compensates both skew and offset simultaneously. The delay in packet transmission and reception is assumed to be negligible in this case.

## (viii) Consensus Clock Synchronization (CCS)

In [17], the authors have proposed Consensus Clock Synchronization (CCS) protocol. In each synchronization round, the CCS algorithm compensates the clock parameters for each node and after a finite synchronization round, the clocks of all nodes converge to a consensus. To compensate the parameters, nodes iterate the CCS algorithm which consists of two main phases: (i) offset compensation and (ii) skew compensation. In the offset compensation phase, nodes exchange local clock readings which are then updated using one of the weighted averaging methods. In skew compensation, the nodes iteratively compare the results from the current and previous synchronization round to improve their skew compensation parameter. The synchronization rounds are repeated at intervals of $t_{sync}$ which can be increased or decreased, depending on the precision requirement of the network. Three weighted averaging methods are proposed for CCS, namely, Cumulative Moving Average (CMA), Forwards Weighted Average (FWA), and Confidence Weighted Average (CWA). Using a network of 100 nodes and random linear drift, the authors claimed that CWA proved most reliable. Also, FWA performance is same as CWA and has the advantage of reduced computational complexity.

## (ix) Clustered Consensus Time Synchronization (CCTS)

In order to improve convergence speed and energy efficiency, Jie Wu *et al.* in [57] have adapted clustering technique in consensus based clock synchronization algorithm. They have incorporated LEACH clustering algorithm as a pre-step to their synchronization algorithm. The first stage of the proposed CCTS algorithm is the intra-cluster time synchronization. In this stage, the cluster heads estimate the average values of the skew compensation parameters of clocks of nodes within their clusters and the average values of intra-cluster virtual clocks of nodes, and then they update the clock compensation parameters of intra-cluster virtual clocks and simultaneously broadcast them to the neighboring nodes. Cluster member nodes receive the messages and update the local intra-cluster virtual clock compensation parameters to achieve the synchronization of intra-cluster virtual clocks. The second stage of the algorithm is the inter-cluster time synchronization. The cluster heads exchange their intra-cluster virtual clocks and their clock compensation parameters through gateway nodes. The received messages are given corresponding weights according to the size of each cluster. Then cluster heads update skew and offset compensation parameters of network virtual clocks in order to achieve the synchronization of network virtual clocks.

Some of the aforementioned time synchronization algorithms have been recently extended by considering different performance metrics. For example, the WMTS (Weighted Maximum Time Synchronization) [59] is an extension of MTS [19] which considers communication delay between sensor nodes. The SMTS (Secured Maximum Time Synchronization) [60] considers the security metric (message manipulation attack) in MTS and RMTS [61] is an extension of MTS on a mobile random network. Similarly, the message manipulation attack is also considered in SATS (Secured Average Time Synchronization) [62] as an extension of ATS protocol proposed by Schenato *et al.* in [18].

Table 2.1 gives a brief quantitative & qualitative analysis of the above-discussed state-of-the-art synchronization algorithms.

## 2.6 Key Observations

The literature survey carried out in this Chapter leads to the following key observations regarding time synchronization problem which act as the major source of motivation for designing new synchronization algorithms for WSNs.

1. Time synchronization problem does not arise in a centralized system.
2. It is a fundamental challenge in traditional distributed systems due to lack of the centralized physical clock.
3. WSNs, being a distributed system, also face this challenge.
4. The conventional protocols designed for traditional distributed system can not be applied on WSNs due to limited resource, random deployment, dynamic topology, etc.
5. The well-known protocols designed so far for WSNs are hierarchy based or reference based which suffer single-point failure problem and higher synchronization error.

6. Consensus-based Time Synchronization (CTS) is a recent method and is more suitable for WSNs because of computational lightness, purely distributed nature and robustness.

7. The state-of-the-art consensus-based synchronization algorithms have slower convergence speed. So, there is a need for designing faster consensus-based synchronization algorithms with better synchronization accuracy.

8. In CTS algorithms, every node iterates the algorithm till the desired synchronization accuracy is achieved. Due to the participation of all node, the message overhead increases, so also, the energy consumption. So, there is a need for topological optimization which can minimize the number of message exchanges and hence, energy consumption without compromising synchronization accuracy.

Table 2.1: Quantitative & Qualitative Analysis of State-of-the-art Synchronization Algorithms

| **Protocols** | **Quantitative Metrics** | | | | **Qualitative Metrics** | | |
|---|---|---|---|---|---|---|---|
| | **Precision** | **Network Size** | **Convergence Time/ No. of iterations** | **Message Complexity** | **Scalability** | **Fault Tolerant** | **Energy Efficiency** |
| TPSN[21] | 16.9 $\mu$s per hop | 150-300 | - | O($n^2$) | Yes | Yes | Low |
| RBS[20] | 29.1 $\mu$s per hop | 2-20 | - | O($n^2$) | Yes | No | High |
| PBS[4] | 30 $\mu$s | - | - | Independent of n | - | Yes | High |
| Multi-hop PBS[43] | - | 200-400 | - | O($n^2$) | Yes | - | Medium |
| ATS[18] | 600 $\mu$s | 9-35 | 10 mins. ( 5 sec/polling cycle × 120 polling‘ cycle) | - | Yes | Yes | Low |
| ATSP[22] | 0.5 $\mu$s | 300 | $\approx$ 10 iterations | O($n$) | - | - | - |
| $R^4$Sync[16] | 5 $\mu$s | 10 | 200 sec. | O($n^3$) | - | Yes | - |
| Multi-hop $R^4$Sync[58] | 5.6 $\mu$s upto 10 hop | 33 | - | - | - | - | - |
| MTS[19] | 100 $\mu$s | 30-50 | $\approx$ 212 iterations | - | Yes | - | - |
| CCS[17] | 270 $\mu$s | 100 | $\approx$ 15 iterations | - | - | - | - |
| CCTS[57] | 30.2 $\mu$s | 50-400 | $\approx$ 30 iterations | - | Yes | - | High |

## 2.7 Summary

A comprehensive study of the time synchronization algorithms for traditional distributed system and WSNs is presented in this Chapter. For the last decade, quite a good number of time synchronization algorithms have been proposed in the literature. Still, there exist a trade-off between the metrics and the requirements. Classifications of time synchronization algorithms, based on issues and application requirements, are presented. Based on our exhaustive survey, a broader way of classifying time synchronization algorithms is also suggested. A detail study of some recent and state-of-the-art algorithms, in the field of time synchronization for WSNs, is also carried out. Finally, a conclusive quantitative and qualitative analysis in terms of some standard performance metrics is presented to know the merits and demerits of the existing synchronization algorithms.

# Chapter 3

# Consensus Time Synchronization Algorithm by Selective Averaging

In this Chapter, a distributed, internal time synchronization algorithm is proposed based on average consensus theory. The algorithm is purely distributed (runs at each node) and every node exploits selective averaging with the neighboring node having maximum clock difference. To identify the neighboring node with maximum clock difference, every node broadcasts a synchronization initiation message to the neighboring nodes at its local oscillation period and waits for a random interval to get the synchronization acknowledgment messages. After receiving acknowledgment messages, a node estimates relative clock value and sends an averaging message to the selected node. The iteration continues until all nodes reach an acceptable synchronization error bound. The optimal convergence of the proposed SATS algorithm is analyzed and validated through simulation and compared with some state-of-the-art, average consensus based time synchronization algorithms.

## 3.1 Introduction

Clocks of sensor motes drift from real time because of various reasons like temperature, vibration, magnetic field, and aging effect of the quartz oscillator. In order to maintain a common timescale for consistency and correctness of other protocols and applications in WSNs, the clocks of the nodes need to be synchronized. Recently, to develop fully distributed and internal time synchronization mechanism, consensus time synchronization (CTS) method has gained much attention [17–19, 22, 23, 25, 63]. CTS is essentially based on the distributed average consensus method in which the synchronization can be achieved by communicating only with neighbors [26, 27]. Because of its simplicity, computational lightness, robustness to node and link failure, and purely distributed nature, CTS is more suitable for WSNs than hierarchy based synchronization methods. This motivates us to design a consensus based time synchronization algorithm for WSNs.

In this Chapter, a novel average consensus based time synchronization algorithm is proposed for WSN. It exploits the broadcast nature of wireless communication medium for offset and skew estimation and then performs a selective pair-wise averaging to achieve faster convergence and better synchronization accuracy than some state-of-the-art CTS algorithms.

The major contributions of this Chapter are the followings.

1. A hybrid approach of broadcasting and pair-wise message passing paradigm is proposed to achieve average consensus based time synchronization.
2. A maximum difference based generic approach for both offset and skew compensation is proposed which enables the algorithm to achieve faster convergence and better synchronization accuracy.
3. An in-depth mathematical analysis is carried out to prove the optimal behavior of the algorithm and its performance is verified through simulation and compared with some recent, state-of-the-art consensus-based synchronization algorithms.

The rest of this Chapter is organized as follows. Section 3.2 presents some relevant definitions, models, and the problem formulation. Section 3.3 outlines the proposed Selective Average Time Synchronization (SATS) algorithm. Section 3.4 presents the mathematical analysis of the optimality, consensus convergence, message complexity and energy consumption of the proposed algorithm. Section 3.5 shows the simulation results followed by conclusion in Section 3.6.

## 3.2 System Model & Problem Formulation

In this Section, we have introduced the clock model and the network model, based on which the synchronization algorithm is proposed. Subsequently, the consensus-based time Synchronization (CTS) problem is formulated using average consensus theory.

### 3.2.1 Clock Model

Sensor nodes are generally associated with a hardware-based oscillator clock [14]. It counts an approximation of real time *‘t’* which can be mathematically expressed as Equation 3.1:

$$C(t) = k\int_0^t \omega\ (t)dt + C\ (t_0) \tag{3.1}$$

where $\omega(t)$ is the angular frequency of the oscillator, *‘k’* is a proportionality coefficient and C($t_0$) is the initial value of the clock. If the angular frequency can be approximated to a fixed value, then for a node *‘i’*, the affine clock model can be expressed as Equation 3.2:

$$C_i(t) = \alpha_i t + \beta_i \tag{3.2}$$

where $\alpha_i$=clock skew, $\beta_i$=clock offset.

Clock skew is defined as the rate or frequency of the clock and clock offset is defined as the deviation from the real time. To compare the local clock of any node *‘i’* w. r. to another node *‘j’*, the above expression can be rewritten as Equation 3.3:

$$C_i(t) = \alpha_{ij}t + \beta_{ij} \tag{3.3}$$

where $\alpha_{ij}$=relative skew, $\beta_i$=relative offset. If two nodes are synchronized, then $\alpha_{ij} = 1$ and $\beta_{ij} = 0$.

### 3.2.2 Network Model

For our proposed algorithm, the WSN is assumed to be a random connected graph *G= (V, E, r)*, where V denotes set of *'n'* nodes, set *'E'* represents the connectivity matrix between the nodes and *'r'* is the connectivity radius. Two nodes are said to be neighboring nodes if the Euclidean distance between them is less than the connecting radius. *E* is a *n* × *n* adjacency matrix where $e_{ij}$ represents the entries in the matrix. All nodes have unique IDs. The communication channel between the pair of nodes is assumed to be static, symmetric and undirected, i.e., upstream delay and downstream delay is same ($d_{ip}$=$d_{pi}$=$d_1$, *p*=1, 2..., k,..., j) in Fig. 3.1. If node $v_i$ and $v_j$ are neighboring nodes, then $e_{ij} = e_{ji} = 1$. Otherwise, $e_{ij} = e_{ji} = 0$. Let, $N_i = j : (i, j) \in E$ denotes the set of one-hop neighbors of node $v_i$. The communication topology is assumed to be fully distributed where there is no special node such as root or reference node and all nodes execute the same algorithm.

### 3.2.3 Energy Model

To estimate energy consumption in wireless communication, two mostly used radio models are: free-space (fs) model and multi-path (mp) model [64]. Since the proposed algorithm is based on consensus and consensus algorithms follow one hop communication, the free- space model is more suitable. Further, the major energy consumption to achieve synchronization is due to synchronization message transmission and reception. Therefore, we have considered these two factors for energy consumption estimation. Using free-space model, the energy consumption $P_{tx}$ for a message transmission and the energy consumption $P_{rx}$ for a message reception is given as follows.

$$P_{tx} = M(\beta_1 + \beta_2 l(i, j)^{\zeta}) \; and \; P_{rx} = M\gamma \tag{3.4}$$

where '$\zeta$' is the path loss exponent, typically set to 2 for free-space model. The constants $\beta_1$, $\beta_2$ and $\gamma$ are the energy dissipated by the transmitter module, transmit amplifier, and the receiver module respectively. The estimated distance between nodes *'i'* and *'j'* is denoted as $l(i, j)$ and the length of message as '$M$'.

### 3.2.4 Problem Formulation

Based on above models and average consensus principle [26, 27], the consensus-based time synchronization (CTS) problem in a network of '$n$' nodes can be defined as converging each node's clock value to a consensus value, i.e., the average of the initial clock values of all nodes, after a series of iterations. In each iteration, the nodes will communicate only with the neighboring nodes to estimate the relative clock values and compensate the local clock as per the proposed algorithm to reach the consensus value. Mathematically, CTS can be defined as:

$$\lim_{k\to\infty} C_i(t_k) = \frac{1}{n}\sum_{i=1}^{n} C_i(0), i = 1, 2, 3, ..., n \tag{3.5}$$

where $C_i(t_k)$=clock value of node *'i'* at $k^{th}$ iteration, $C_i(0)$=initial clock value of node *'i'*.

From Equation 3.2,

$$\frac{1}{n}\sum_{i=1}^{n} C_i(0) = \frac{1}{n}\sum_{i=1}^{n} \beta_i(0), i = 1, 2, 3, ..., n \tag{3.6}$$

where $\beta_i(0)$= offsets of all clocks at time *t*=0. This is known as the initial offset. So, using Equation 3.6, Equation 3.5 can be rewritten as:

$$\lim_{k\to\infty} C_i(t_k) = \frac{1}{n}\sum_{i=1}^{n} \beta_i(0) \tag{3.7}$$

So, CTS is said to be achieved when Equation 3.7 is satisfied after a series of iterations *'k'* by each node in the network. Equation 3.7 signifies the ideal definition of consensus-based time synchronization. Practically, it is quite impossible to achieve consensus on the exact global average time for the whole WSN because of clock skew, random delays and network dynamism [15]. So, we have assumed that converging to an approximate value for Equation 3.7 will suffice the definition for CTS. On this basis, the term ''acceptable synchronization error" is defined as given below and used throughout the thesis. This also acts as the termination criteria for the synchronization algorithm.

**Definition:** Acceptable Synchronization Error ($\epsilon$)

It is a pre-specified value provided to the CTS algorithm such that each node in the network will iterate the algorithm until the following condition is satisfied.

$$\lim_{k\to\infty} \left| C_i(t_k) - \frac{1}{n}\sum_{i=1}^{n} \beta_i(0) \right| \leq \epsilon \tag{3.8}$$

Since, the consensus value, i.e., the initial global average is unknown locally at each node in the network, and by principle of average consensus theory, every node will reach the consensus state after a series of iterations, thus, each node can verify the bound for '$\epsilon$' by simply checking the clock difference with the neighboring nodes. So, basically, '$\epsilon$' denotes the upper bound for the local synchronization error at every node.

### 3.2.5 Mathematical Preliminaries

This Section briefly introduces the mathematical properties from matrix and consensus theory which are used to validate the proposed algorithm in Section 3.3. In particular, the Greedy Gossip with Eavesdropping (GGE) consensus theory [65] is employed (Lemma 3.4.2) to prove the optimality of synchronization error in Theorem 3.4.1. The GGE consensus theory suggests that for convex optimization problem, the equivalent randomized incremental expression is given by maximum-difference based averaging (Lemma 3.4.2). Since the clock model is an affine function and the affine function is a well-known convex function, the optimality of error function can be suitably realized using GGE principle.

Further, the convergence analysis of the proposed algorithm is carried out using the special properties of compensation matrices which are proved to be doubly stochastic matrices with positive diagonal entries in Lemma 3.4.4. The semi-contractive properties of product of compensation matrix-chain is analyzed in its 2-norm (Lemma 3.4.5) which defines an approximation to the averaging factor. In fact, an approximation to the averaging factor is more realistic in a delay-sensitive and dynamic environment where exact averaging is practically infeasible. Then using rank one matrix property, the consensus convergence is proved in Theorem 3.4.7.

Based on the above-discussed models, definitions, assumptions and mathematical background, the following Section describes the proposed Selective Average Time Synchronization (SATS) algorithm in detail.

## 3.3 The SATS Algorithm

The proposed algorithm consists of a series of synchronization round which is iterated at each node according to its local clock's oscillation period. The first synchronization round is triggered by a random subset of initiating nodes upon receipt of a signal from a base station. The subsequent rounds are initiated when a synchronization initiation message (SYN_INIT) message is received from a node as shown in Fig. 3.1. All the one-hop neighboring nodes which receive SYN_INIT message, reply with synchronization acknowledgment (SYN_ACK) messages. Then using the proposed algorithm, the initiating node will send a synchronization averaging (SYN_AVG) message to a selected node.

Each synchronization round is divided into two phases: phase-1 is for parameter (offset/skew) estimation and phase-2 is for parameter compensation. Both offset and skew estimation is performed in the same synchronization round. Phase-1 of a node 'i' is assigned with a random duration of time which is given by $t^{i}_{rand}$=uniform (0, $P_i$-$\kappa$) where $\kappa$=constant, 0< $\kappa$<$P_i$, $P_i$= oscillation period of node *'i'*. Thus, duration of phase-2 is given by ($P_i$-$t^{i}_{rand}$). One iteration is said to be over when each node has completed at least one synchronization round. This is ensured by assigning the duration of one iteration=max$\{P_i\}$, *i=1, 2,..., n* so that even if the node with the slowest clock can complete one synchronization round per

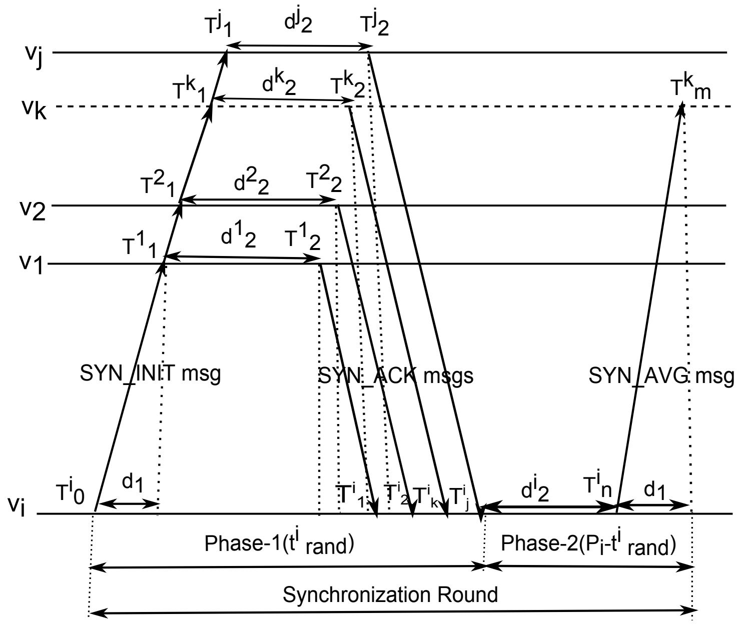


Figure 3.1: Message Passing in SATS

iteration.

The algorithm considers physical layer and MAC layer delays for synchronization. In Fig. 3.1, $d_1$ is the physical (propagation) delay and $d_2^p$, $p$=1, 2,..., k,..., j is the MAC layer delay at each neighbor ‘*p*’. Since the propagation delay in wireless medium is proportional to distance [66] and the proposed synchronization algorithm requires communication with the neighboring nodes within a fixed connectivity radius *‘r’*, $d_1^p$ is assumed to be equal for all the neighbors which are denoted as $d_1$. The delay $d_1$ is quite negligible for 1-hop communication [18]. The algorithm assumes the usage of MAC layer time-stamps [67], appended in the control frames (e.g. IEEE 802.11 RTS/ CTS control packets for CSMA/ CA) and the averaged parameter’s(offset/skew) value in the actual data frame, to minimize the MAC delay.

In the long run, the clocks may get unsynchronized when the skew has again occurred. So, the base station periodically checks if $\Delta$t $> \epsilon$ where $\Delta$t=max$\{C_i(t) - C_j(t)\}$, $i, j \in V$. This condition is true when the skew is reoccurred. Then the algorithm is re-initiated by the base station to maintain the synchronization.

The pseudo codes of the SATS algorithm are given in Algorithm 1. It consists of both offset and skew averaging methods which can be performed simultaneously for faster convergence to the consensus value. Algorithm 2 discusses the offset averaging process, and Algorithm 3 discusses the skew averaging process. Fig. 3.2 (a) and (b) depict the skew estimation method.

**Remark:** It is important to convey here that both skew and offset compensation can be performed simultaneously and with the same neighboring node because the neighboring node with the maximum relative skew will also have the maximum offset difference. This

**Algorithm 1** SATS ( $v_i$)

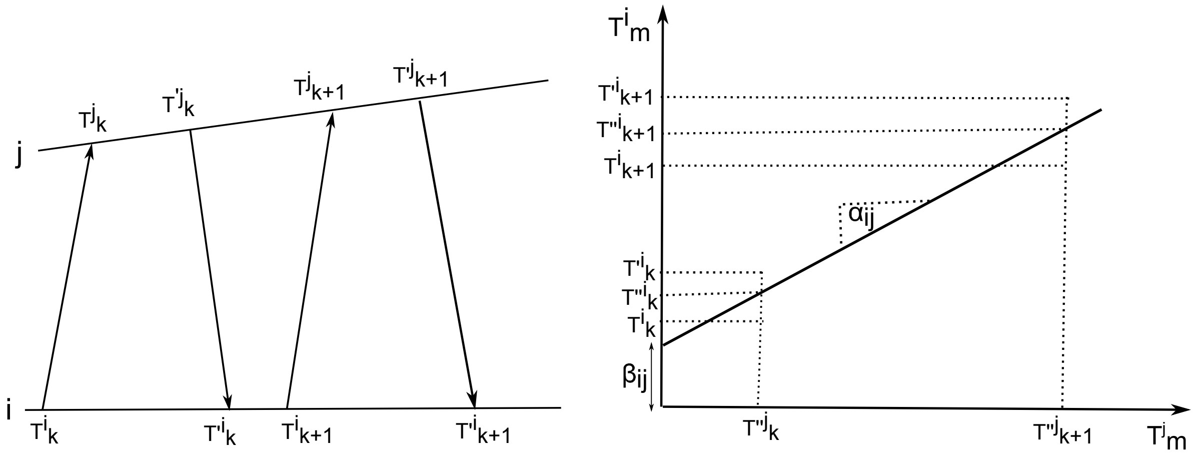
```
Input: Acceptable synchronization error ($\epsilon$)
Output: Consensus state ($C_i(0)$)
for all node $v_i \in$ V do
    while local synchronization error $\leq \epsilon$ do
        for each iteration do
            OFFSET_AVERAGING ( $v_i$)
            SKEW_AVERAGING ( $v_i$)
        end for
    end while
end for
```

**Algorithm 2** OFFSET_AVERAGING ( $v_i$)

```
for each synchronization round of duration $P_i$ do
    /******** Phase-1: Broadcast based offset estimation ********/
    $v_i$ broadcasts the message SYN_INIT($T_i^0$) to its 1-hop neighbours.
    if $v_p \in N_{v_i}$ and receives SYN_INIT message then
        $v_p$ sends the message SYN_ACK($T_1^p$, $T_2^p$) to $v_i$, where
        $T_1^p$=received timestamp of SYN_INIT message at node $v_p$,
        $T_2^p$= sent timestamp of SYN_ACK message by node $v_p$,
        $N_{v_i}$=set of 1-hop neighbouring nodes of $v_i$
    end if
    for each SYN_ACK message received from $v_p$ within $t_{rand}^i$ do
        Node $v_i$ estimates the relative offset and the delay as:
        $\beta_{ip}$=($T_2^p$-$T_p^i$+$T_1^p$-$T_0^i$)/2 , p=1,2,..,k,..,j
        $d_2^p$=$T_2^p$-$T_1^p$
        $d_1^p$=$d_1$=($T_p^i$-$T_2^p$+$T_1^p$-$T_0^i$)/2 , p=1,2,..,k,..,j
    end for
    /********* Phase-2: Pairwise offset compensation *********/
    After estimation, node $v_i$ selects the neighbour with maximum relative offset.
    Let, node $v_k$ is the neighbour node having maximum relative offset. Then,
    Node $v_i$ updates its local clock as:
    $(T_1^k + T_k^i + d_1 + d_2^k)/2$=$(T_2^k + d_1) - \beta_{ik}/2$
    Finally, node $v_i$ sends the message SYN_AVG($T_k^i$,$T_n^i$) within interval ($P_i$-$t_{rand}^i$)
    which also contains the values of $d_1$, $d_2^i$ and $\beta_{ik}$/2.
    When node $v_k$ receives this message at $T_m^k$, it updates its local clock as:
    $T_k^i + d_1 + T_m^k + d_2^i + \beta_{ik}/2$
end for
```

**Algorithm 3** SKEW_AVERAGING ( $v_i$)

```
 1: /********** Phase-1: Broadcast based Skew Estimation ***********/
 2: for first synchronization round do
 3:     Node ‘i’ broadcasts the message SYN_INIT($T_1^i$) to its 1-hop neighbours.
 4:     if j $\in N_i$ and receives SYN_INIT message then
 5:         Node ‘j’ sends the message SYN_ACK($T_1^j$,$T_1^{\prime j}$)
 6:     end if
 7:     for a SYN_ACK message received at $T_1^{\prime i}$ from ‘j’ within $t^i_{rand}$ do
 8:         Node ‘i’ computes $T_1^{\prime\prime j} = (T_1^j + T_1^{\prime j})/2$ and $T_1^{\prime\prime i} = (T_1^i + T_1^{\prime i})/2$
 9:         and stores the timestamp as ( $T_1^{\prime\prime j}$, $T_1^{\prime\prime i}$)
10:     end for
11: end for
12: for subsequent synchronization round do
13:     /**** Assume current synchronization round as ‘k+1’ and previous round as ‘k’****/
14:     Node ‘i’ computes the timestamp ($T_{k+1}^{\prime\prime j}$, $T_{k+1}^{\prime\prime i}$) in the current round using step-4 to
        step-9 where $T_{k+1}^{\prime\prime j} = (T_{k+1}^j + T_{k+1}^{\prime j})/2$ and $T_{k+1}^{\prime\prime i} = (T_{k+1}^i + T_{k+1}^{\prime i})/2$
15:     Retrieves the stored previous round timestamp ($T_k^{\prime\prime j}$, $T_k^{\prime\prime i}$)
16:     Using this timestamp pair, node ‘i’ computes the relative skew $\alpha_{ij} = (T_{k+1}^{\prime\prime i} -
        T_k^{\prime\prime i})/(T_{k+1}^{\prime\prime j} - T_k^{\prime\prime j})$ for every j$\in N_i$
17:     Node ‘i’ stores the current round timestamp ($T_{k+1}^{\prime\prime j}$, $T_{k+1}^{\prime\prime i}$)
18:     /********* Phase-2: Pairwise Skew Compenssation **********/
19:     In the current round, after computing the relative skew for each j$\in N_i$, node‘i’ sends
        SYN_AVG message to the neighbour node with maximum relative skew and also sets
        its own clock’s skew as $\alpha_i$=max$\{\alpha_{ij}/2\}$
20:     Node’j’after receiving the SYN_AVG message sets its clock’s skew as $\alpha_j = \alpha_{ij}/2$
21: end for
```



(a) (b)

Figure 3.2: (a)Timestamps exchange between node i and node j, (b) Relative clock skew estimation of node'i' w. r. to node 'j'

can be verified as follows. The relative time instances between a node *'i'* and its neighbor node *'j'* can be written as $t_i = \frac{\alpha_i}{\alpha_j} t_j + \left(\beta_i - \frac{\alpha_i}{\alpha_j}\beta_j\right)$[18]. The factor $\left(\beta_i - \frac{\alpha_i}{\alpha_j}\beta_j\right)$ denotes the relative offset between the nodes. Now, considering the synchronized state between the nodes i.e. equating the factor to zero, we have $\left(\beta_i - \frac{\alpha_i}{\alpha_j}\beta_j\right)$=0. So, $\frac{\alpha_i}{\alpha_j} = \frac{\beta_i}{\beta_j}$. Hence, as the time elapses, a proportionate increase in the relative skew ($\frac{\alpha_i}{\alpha_j}$) also increases the offset in the same ratio. So, the neighboring node with maximum relative skew will also have the maximum offset difference.

So, maximum difference based average compensation enables the algorithm for faster convergence with optimal message complexity and synchronization error. This is mathematically validated by the following section.

## 3.4 Performance Analysis

In this Section, the proposed SATS algorithm is validated mathematically. It is shown that the algorithm optimizes the synchronization error and also converges to the global average consensus. Since the clock model described in Section 3.2 is an affine function and affine function is convex in nature [68], the optimality analysis in Theorem 3.4.3 is based on the convex property of the affine clock function and GGE consensus theory [65]. Further, the convergence analysis is carried out using the special properties of compensation matrices which are proved to be doubly stochastic matrices in Lemma 3.4.4. The compensation matrices are analyzed in its 2-norm which defines an approximation to the averaging factor $\frac{\mathbf{1}\mathbf{1}'}{n}$ [69].

### 3.4.1 Optimality of Synchronization Error

In this Section, the optimality proof of the SATS algorithm is derived in Theorem 3.4.3 using the principle of convex optimization and greedy gossip theory which is given below.

**Lemma 3.4.1.** *[70] For the generic convex optimization problem which can be defined as:*
*Minimize* $\sum_{i=1}^{n} f_i(x), x \in X$
*where each* $f_i(x)$ *is a convex function, not necessarily differentiable, the optimal solution using incremental sub-gradient method is given by:*

$$x_{k+1} = P_x \left[ x_k - \alpha_k g\left(\omega_k, x_k\right) \right] \tag{3.9}$$

*where* $P_x$*=Projection onto the set* $X$*,* $\alpha_k$*=step-size, g(*$\omega_k$*,*$x_k$*)=sub-gradient of* $f_{\omega_k}$ *at* $x_k$*,* $\omega_k$ *=random variable taking equi-probable values from the set* $\{1, 2, ..., n\}$

**Lemma 3.4.2.** *[65] According to GGE consensus theory, the equivalent randomized incremental sub-gradient expression can be written as:*

$$min \sum_{i=1}^{n} max_{j \in N_i} \frac{1}{2} \{x_i - x_j\} \tag{3.10}$$

*The sub-gradient for the above expression is given by:* $g_i\left(k+1\right) = |x_i(k) - x_j(k)|$

**Theorem 3.4.3.** *The maximum difference based clock averaging is an optimal solution for the total synchronization error function* $Er(t)$ *which is given by:*

$$Er(t) = \sum_{i=1}^{n} \left[ C_i(t) - avg\ C_i(0) \right] \tag{3.11}$$

*where* $C_i(t) = \alpha_i t + \beta_i$*,* $avg\ C_i(0) = \frac{1}{n} \sum_{i=1}^{n} C_i(0)$*,* $\alpha_i$ *=frequency of oscillation of node i,* $\beta_i$*=offset of node i, i=1, 2, 3,..., n.*

*Proof.* The clock function $C_i(t)$ is an affine function which is of convex type [70]. So, the minimization problem can be considered as a convex optimization problem. Hence, using Lemma 3.4.1, the incremental sub-gradient solution for Equation 3.11 will be given by the iterative update,

$$C_i(t_k) = P_t \left[ C_i(t_{k-1}) - \alpha_k g_i\left(k\right) \right] \tag{3.12}$$

where $t_k$, $t_{k-1}$ denotes timestamps at iteration k and k-1 respectively.

Since, the averaging is done pair-wise, i.e., $\alpha_k = 1/2$ (a constant), the projection $P_t$ is not necessary. Now mapping with Lemma 3.4.2, the sub-gradient $g_i(k)$ in Equation 3.12 mathematically signifies the neighbor selection with maximum difference clock value. Thus, the proposed maximum difference based clock averaging scheme can be expressed using Equation 3.12 as:

$$C_i(t_k) = \left[ C_i(t_{k-1}) - \frac{1}{2} g_i\left(k\right) \right] \tag{3.13}$$

Since, the algorithm needs to be iterated a number of time until the acceptable synchronization error is reached, equivalently, solving the recurrence Equation 3.13 by repeated substitution method, we have

$$C_i(t_k) = \left[C_i(t_{k-1}) - \frac{1}{2}g_i\left(k\right)\right] \tag{3.14}$$

$$= \left[C_i(t_{k-2}) - \frac{1}{2}\sum_{s=k-1}^{k} g_i\left(s\right)\right]$$

.

.

$$= \left[C_i(0) - \frac{1}{2}\sum_{s=1}^{k} g_i\left(s\right)\right]$$

After infinite series of iterations, i.e., $k \to \infty$, sum of sub-gradients $\sum_{s=1}^{k} g_i(s)$ converges to zero almost surely [65]. So, as $k \to \infty$,

$$C_i(t_k) \approx C_i(0) \tag{3.15}$$

Substituting the value of $C_i(t_k)$ from Equation 3.15 in Equation 3.11, the total synchronization error function $Er\ (t)$ after infinite iterations can be approximated as:

$$Er(t) \approx \sum_{i=1}^{n} \left[C_i(0) - avg\ C_i(0)\right] \tag{3.16}$$

$$\begin{aligned}
&\approx \sum_{i=1}^{n}\left[C_i(0) - \frac{1}{n}\sum_{i=1}^{n} C_i(0)\right] \\
&\approx \sum_{i=1}^{n} C_i(0) - \sum_{i=1}^{n}\frac{1}{n}\sum_{i=1}^{n} C_i(0) \\
&\approx \sum_{i=1}^{n} C_i(0) - \sum_{i=1}^{n} C_i(0) \\
&\approx 0
\end{aligned}$$

Hence, $Er\ (t) \approx 0$ is nothing but the near optimal value for the total synchronization error function. □

### 3.4.2 Consensus Convergence

This Section shows the consensus convergence of the algorithm by exploring the special property of clock compensation process as given below.

**Lemma 3.4.4.** *For the proposed algorithm, the pair-wise average compensation process produces doubly stochastic matrices with positive diagonal entries.*

*Proof.* Consider, the clock values of a network of *'n'* nodes is represented as a vector $\boldsymbol{C}(t_k)$ of order $1\times$ n where $t_k$ represents the time-stamp at a particular iteration *'k'*. Lets, the synchronization algorithm is initiated by a node *'s'* and after estimating the clock values using phase-1 of the algorithm, lets the neighboring node ‘$r$’ is having the maximum clock difference with node *‘s’*. Then, the compensation phase can be expressed as:

$$\boldsymbol{C}(t_k) = \boldsymbol{M}_k \boldsymbol{C}(t_{k-1}) \tag{3.17}$$

where entries in the matrix $\boldsymbol{M}_k$ are defined as:

$$\boldsymbol{M}_k(i,j) = \frac{1}{2}\, if\ i \in \{s,r\}\ and\ j \in \{s,r\} \tag{3.18}$$

$$\boldsymbol{M}_k(i,j) = 1\ if\ i = j, i \notin \{s,r\} \tag{3.19}$$

$$\boldsymbol{M}_k(i,j) = 0,\ elsewhere \tag{3.20}$$

Clearly, the diagonal entries satisfy Equation 3.18 or 3.19, hence, positive. Furthermore, when node *‘r’* and *‘s’* are synchronizing , the corresponding $r^{th}$ row and $r^{th}$ column in $\boldsymbol{M}_k$ will contain $\frac{1}{2}$ at indexes *(r, r)* and *(r, s)* and rest indexes are zero. Hence, $r^{th}$ row sum and column sum is 1. Similar case happens for $s^{th}$ row and column. For remaining nodes which are not synchronizing, the corresponding rows and columns in $\boldsymbol{M}_k$ are zero except 1 at diagonal entries. Hence, for rest of the rows and columns, the sum is also 1. So, $\boldsymbol{M}_k$ is a doubly stochastic matrix with positive diagonal entries in each iteration. □

**Lemma 3.4.5.** *[69] Doubly stochastic matrix with positive diagonal entries are semi-contractive in 2-norm and product of semi-contractive matrix chain* $\boldsymbol{M}_i\boldsymbol{M}_{i-1}...\boldsymbol{M}_1$, $i \in \{1,2,..\}$ *in 2-norm converges to a rank one matrix of the form* $\mathbf{1}\boldsymbol{c}'$ *as* $i \rightarrow \infty$.

**Lemma 3.4.6.** *[71] For a non-zero rank one matrix* $\boldsymbol{A}$. *If* $\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}'$ *and if* $\boldsymbol{A} = \boldsymbol{p}\boldsymbol{q}'$, *then* $\boldsymbol{p} = k\boldsymbol{x}$ *and* $\boldsymbol{q} = \boldsymbol{y}/k$ *for some scalar ‘k’.*

Based on above Lemmas, the following Theorem proves that the synchronization algorithm converges to the average of the initial clock values.

**Theorem 3.4.7.** *After ’k’ iterations as* $k \rightarrow \infty$, *the product of average compensation matrices* $\boldsymbol{M}_k$ *converges to* $\frac{\mathbf{1}\mathbf{1}'}{n}$. *In other words, the proposed algorithm converges to the average consensus.*

*Proof.* Applying repeated substitution in Equation 3.17, the clock compensation phase after iterations ‘k’ can be rewritten as:

$$\boldsymbol{C}(t_k) = \boldsymbol{M}_k\boldsymbol{C}(t_{k-1}) = \boldsymbol{M}_k\boldsymbol{M}_{k-1}...\boldsymbol{M}_1\boldsymbol{C}\,(0) \tag{3.21}$$

Lemma 3.4.4 shows that the compensation matrices $\boldsymbol{M}_j$’s are doubly stochastic with positive diagonal entries. Hence, using lemma 3.4.5, $\boldsymbol{M}_j$’s are semi-contractive in its 2-norm and the product of compensation matrices in its 2-norm after infinite iterations will be:

$$\boldsymbol{M}_k\boldsymbol{M}_{k-1}...\boldsymbol{M}_1 = \mathbf{1}\boldsymbol{c}' \tag{3.22}$$

Since, $\boldsymbol{M}_k$’s are symmetric, the transpose of the matrices are also doubly stochastic with positive diagonal entries. Hence, applying Lemma 3.4.5 on transpose of the matrices, we have

$$(\boldsymbol{M}_k\boldsymbol{M}_{k-1}...\boldsymbol{M}_1)' = \mathbf{1}\boldsymbol{d}' \tag{3.23}$$

Substituting the value of the matrix chain from Equation 3.22 in Equation 3.23, $(\mathbf{1}\boldsymbol{c}')$'=$\mathbf{1}\boldsymbol{d}'$. So, $\boldsymbol{c}\mathbf{1}' = \mathbf{1}\boldsymbol{d}'$. As per Lemma 3.4.6, $\mathbf{1}\boldsymbol{c}'$ and $\mathbf{1}\boldsymbol{d}'$ are rank one matrices. Lets denote $\mathbf{A} = \mathbf{c}\mathbf{1}' = \mathbf{1}\mathbf{d}'$. Using Lemma 3.4.6, $\boldsymbol{c}$=$\boldsymbol{d}$=$\frac{1}{k}\mathbf{1}$. Since, $\boldsymbol{c}$ and $\boldsymbol{d}$ are stochastic and $\mathbf{1}$ is a unit vector of order 1×n, $\frac{1}{k} + \frac{1}{k} + \frac{1}{k} + ... + n\ times = 1$. So, k=n. $\boldsymbol{c}$=$\boldsymbol{d}$=$\frac{1}{n}\mathbf{1}$. Hence, $\boldsymbol{M}_k\boldsymbol{M}_{k-1}...\boldsymbol{M}_1 = \frac{\mathbf{1}\mathbf{1}'}{n}$. In other words, after ‘k’ iterations as $k \to \infty$, $\boldsymbol{C}(t_k) = \frac{\mathbf{1}\mathbf{1}'}{n}\boldsymbol{C}(0)$ which shows the synchronization algorithm converges to the average of the initial clock values.

□

### 3.4.3 Message Complexity

The message complexity analysis of the proposed algorithm is based on the basic handshaking Lemma for a connected, undirected graph which is reproduced as Lemma 3.4.8 as follows.

**Lemma 3.4.8.** *[72] In a finite undirected graph, the sum of all the vertex's degree is equal to twice the number of edges i.e.* $\sum_{i=1}^{n} Deg_i$=$2|E|$ *where ‘n’ is the number of nodes and ‘E’ is the set of edges in the graph.*

Based on the above Lemma, the following Lemma proves that the message complexity per iteration of the proposed algorithm is linearly upper bounded by the number of nodes in the network.

**Lemma 3.4.9.** *The asymptotic message complexity per iteration for SATS algorithm is* $O\ (n)$ *where n=number of sensor nodes in the network.*

*Proof.* Considering WSN as a random connected graph of *‘n’* nodes, the number of messages required per iteration can be summarized as follows. Since, each node broadcasts at least one SYN_INIT to all its neighbors, the number of SYN_INIT messages for *‘n’* nodes= $O\ (n)$.

The number of SYN_ACK messages received by a node will be equal to the number of neighboring nodes, i.e., number of SYN_ACK messages=$O\ (\sum_{i=1}^{n} Deg_i)$, where $Deg_i$ denotes degree of a node *‘i’* or number of neighboring nodes. Now, after clock parameter estimation phase, each node will perform pairwise averaging with only one neighboring node which satisfies the maximum difference criteria. So, each node will send one SYN_AVG message to one of its neighbors. For, *‘n’* nodes, number of SYN_AVG messages= $O\ (n)$. So, the total number of messages *‘M’* per iteration is given by:

$M\ (n)$=$O\ (n)$+ $O\ (\sum_{i=1}^{n} Deg_i)$+ $O\ (n)$. According to Lemma 3.4.8, for a connected graph, $\sum_{i=1}^{n} Deg_i$=$2|E|$. But, in a connected graph of *‘n’* nodes, number of edges, $E = (n-1)$. So, $M\ (n)$= $O\ (n)$+ $O\ (2(n-1))$+ $O\ (n)$= $O\ (n)$. This shows that the message complexity of the proposed algorithm is linear with respect to the number of nodes.

□

Table 3.1 shows the total number of messages required and the asymptotic message complexity comparison with some existing synchronization protocols. Since, the proposed algorithm is averaging based, comparing with ATSP [22] and CCS [17], it is observed that though both of them have same asymptotic message complexity, in our algorithm, the number of iterations (I) required is minimized due to maximum difference based averaging

Table 3.1: Comparison of Message Complexity

| *Algorithm* | *No. of Messages Required* | *Asymptotic Complexity* |
|---|---|---|
| RBS [20] | $I+n(n-1)/2$ | $O\left(n^2\right)$ |
| TPSN [21] | $2I(n-1)$ | $O\left(n\right)$ |
| FTSP [4] | $In$ | $O\left(n\right)$ |
| ATSP [22] | $I(3n)$ | $O\left(n\right)$ |
| CCS [17] | $I(3n-2)$ | $O\left(n\right)$ |
| Proposed SATS | $I(4n-2)$ | $O\left(n\right)$ |

as shown in simulation results in Fig. 3.3. Hence, the total number of messages (total number of iterations × number of messages per iteration) required to achieve synchronization is less in our algorithm.

### 3.4.4 Energy Consumption Analysis

To achieve synchronization up to an acceptable synchronization error bound, the SATS algorithm executes at each node in the network by a certain number of iterations. In a particular iteration, the energy consumption at node *'i'* is the sum of energy consumption for SYN_INIT message transmission, energy consumption for receiving SYN_ACK messages from neighbors and energy consumption for sending a SYN_AVG message to a selected node. Following the energy model given in Section 3.2, the following lemma gives an estimation of average energy consumption to achieve desired level of synchronization.

**Lemma 3.4.10.** *The average energy consumption to achieve synchronization in a network of 'n' nodes using SATS algorithm is* $P^{avg}(i) = \frac{1}{n}\sum_{i=1}^{n} It(i) * P(i)$ *where* $P(i) = M[2\beta_1 + \beta_2\{max\{l(i,j)\}^{\zeta} + l(i,k)^{\zeta}, j,k \in N_i\} + \gamma|N_i|]$, *is the total energy consumption per iteration at node 'i' and It(i) is the number of iterations required at node 'i' to reach the acceptable synchronization error bound.*

*Proof.* Using the energy model given in Section 3.2, the following Equations can be derived.

For broadcasting a SYN_INIT message at node *'i'*, the energy consumption is given by:

$$P_{tx}^{SYN\_INIT}(i) = M(\beta_1 + \beta_2\{maxl(i,j), j \in N_i\}^{\zeta}) \tag{3.24}$$

For receiving SYN_ACK messages from neighbors at node *'i'*, the energy consumption is given by:

$$P_{rx}^{SYN\_ACK}(i) = M\gamma|N_i| \tag{3.25}$$

Similarly, for sending a SYN_AVG message by node *'i'* to a selected node *'k'*, the energy consumption is given by:

$$P_{tx}^{SYN\_AVG}(i) = M(\beta_1 + \beta_2\{l(i,k), k \in N_i\}^{\zeta}) \tag{3.26}$$

Summing up the above Equations, the total energy consumption per iteration at node *'i'* is given by:

$$P(i) = P_{tx}^{SYN\_INIT}(i) + P_{rx}^{SYN\_ACK}(i) + P_{tx}^{SYN\_AVG}(i) \tag{3.27}$$

$$= M[2\beta_1 + \beta_2\{max\{l(i,j)\}^{\zeta} + l(i,k)^{\zeta}, j,k \in N_i\} + \gamma|N_i|]$$

Table 3.2: Simulation Parameters

| $Parameter$ | $Values$ |
|---|---|
| Deployment area | 10 × 10 square unit |
| Topology | Random |
| No. of nodes ($n$) | 50-500 |
| Initial skew ($\alpha$) | uniform(-5,5) |
| Initial offset ($\beta$) | uniform(0,1) |
| Iteration interval | 10 sec |
| Acceptable Syn. error ($\epsilon$) | 0.0001sec. |
| Path Loss Exponent ($\zeta$) | 2 |
| $\beta_1$ | 45 nJ/bit |
| $\beta_2$ | 10 pJ/bit |
| $\gamma$ | 35 nJ/bit |
| Message size ($M$) | 320 bits |
| MAC Protocol | CSMA/CA |

Thus, the average energy consumption to achieve network wide synchronization for a network with *'n'* nodes is given by:

$$P^{avg}(i) = \frac{1}{n}\sum_{i=1}^{n} It(i) * P(i) \tag{3.28}$$

where *It(i)* is the number of iterations required at node *'i'* to reach the acceptable synchronization error bound. This proves Lemma 3.4.10.

□

To the best of our knowledge, energy consumption analysis has not been considered in most of the consensus-based synchronization algorithms. For a comparative analysis and evaluation purpose, we have also derived the following corollaries for ATSP [22] and CCS [17] algorithms using a similar reasoning as given in Lemma 3.4.10.

**Corollary 3.4.11.** *The average energy consumption to achieve synchronization in a network of 'n' nodes using ATSP algorithm is* $P^{avg}(i) = \frac{1}{n}\sum_{i=1}^{n} It(i) * P(i)$ *where* $P(i) = M[2\beta_1 + 2\beta_2\{l(i,k)^{\zeta}, k \in N_i\} + \gamma]$*, is the total energy consumption per iteration at node 'i' and It(i) is the number of iterations required at node 'i' to reach the acceptable synchronization error bound.*

**Corollary 3.4.12.** *The average energy consumption to achieve synchronization in a network of 'n' nodes using CCS algorithm is* $P^{avg}(i) = \frac{1}{n}\sum_{i=1}^{n} It(i) * P(i)$ *where* $P(i) = M[\beta_1 + \beta_2\{max\{l(i,j), j \in N_i\}^{\zeta} + \gamma|N_i|]$*, is the total energy consumption per iteration at node 'i' and It(i) is the number of iterations required at node 'i' to reach the acceptable synchronization error bound.*

## 3.5 Simulation

To study the correctness and behavior of the algorithm, it has been simulated in PROWLER simulator [30] under MATLAB environment and the performances have been compared with two recent state-of-the-art average consensus synchronization algorithms, ATSP [22], and CCS [17]. Table 3.2 shows the simulation specifications incorporated in the simulator.

### 3.5.1 Simulation Configuration

The simulations are performed on a random topology of 50 to 500 nodes in a square deployed area of side $(L)$ equals to 10 unit. To ensure a connected topology for consensus propagation, the connectivity radius $(r)$ is calculated using the formula $r = L\sqrt{2\log n/n}$ to make the topology connected with high probability [65].

To simulate the clock, an explicit clock function is defined in the application file of the simulator following the affine clock model given in section 3.2. The real time `$t'$ is assumed to be the 'cputime' for all the nodes. As per TelosB data sheet specification mentioned in [18], the typical skew range is between -5 PPM to 5 PPM. So, to have a close resemblance with the realistic environment, the skew is generated in the specified range using random uniform distribution. To have a fair comparison with ATSP [22] and CCS [17], the clock offsets are generated using random uniform distribution between 0 and 1 which is same as specified in [22][17]. The interval for one iteration, which denotes an upper bound for maximum oscillation period, is set to 10 seconds. The clock parameters, i.e., relative skew and offsets are observed at a pause time of 10 sec. which is the interval of one iteration. This is a valid value between 5sec.-30 sec., a typical range specified in most consensus-based algorithms [17–19]. The default MAC protocol provided in PROWLER simulator is CSMA/CA.

### 3.5.2 Simulation Results and Analysis

In this Section, the simulation results are analyzed according to various performance metrics, viz., convergence speed, average global synchronization error, average local synchronization error, the number of messages exchanged, energy consumption, and scalability in terms of network size and network density.

#### (i) Convergence Speed

The convergence speed to the consensus value is tested for both skew and offset parameter by considering a network of 50 nodes and the observations are analyzed below.

#### (a) Skew Convergence

To test skew convergence, the observations are recorded by considering maximum 50 iterations as shown in Fig. 3.3 (a)-(c). It is observed that ATSP [22] algorithm, which uses random pairwise averaging, achieves skew convergence between 15 to 20 iterations, CCS [17] algorithm which uses cumulative weighted averaging (CWA) method convergences between 10 to 12 iterations, both with an acceptable synchronization error $(\epsilon)$ of 0.0001 sec. Whereas our SATS algorithm, as shown in Fig. 3.3 (c), achieves skew convergence within 5 to 10 iterations for the same value of acceptable synchronization error. So, convergence is faster in our algorithm.

### (b) Offset Convergence

The offset convergence is tested and observed simultaneously with the skew convergence with the same value of acceptable synchronization error as shown in Fig. 3.4 (a)-(c). The initial average of random offset distribution is recorded as 0.49. Similar behavior is observed as in skew convergence. Our algorithm has converged faster than ATSP and CCS algorithm. Overall, our algorithm converges 16% faster than CCS and 50% faster than ATSP.

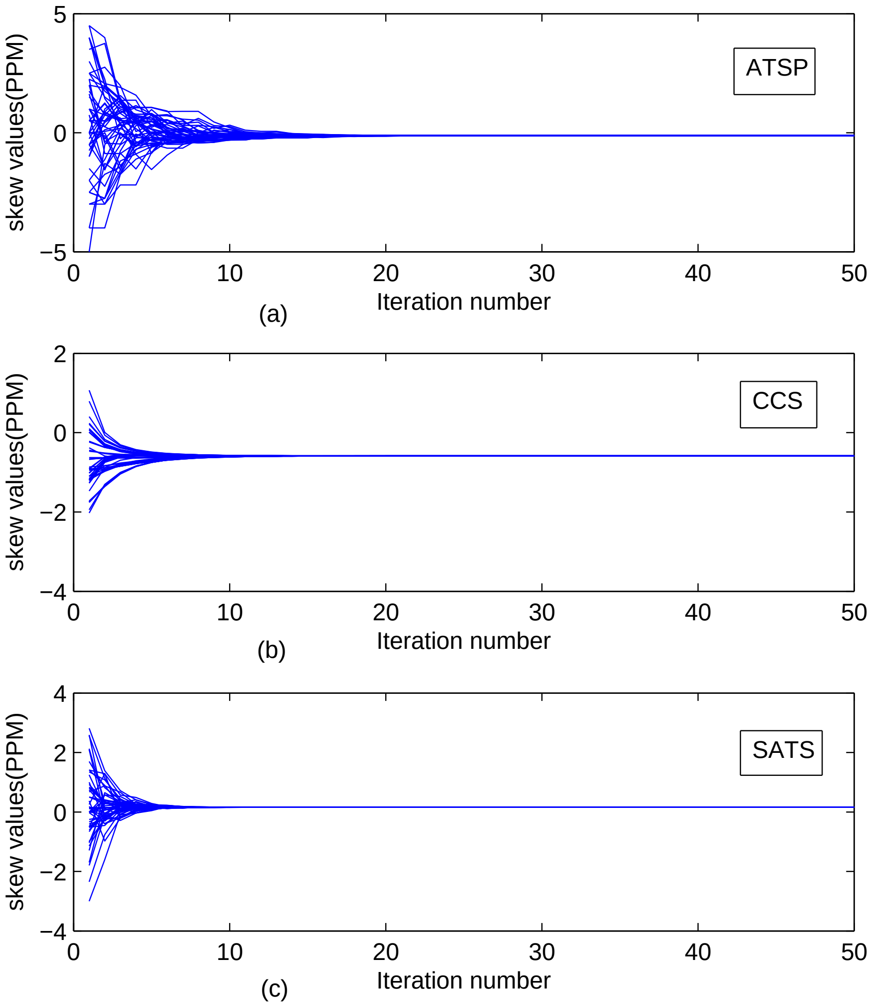


Figure 3.3: Skew convergence of (a) ATSP, (b) CCS, and (c) SATS

### (ii) Average Global Synchronization Error

Fig. 3.5 shows average global (network-wide) synchronization error of 50 nodes after each iteration by considering maximum 50 iterations. It is observed that our algorithm has less synchronization error, nearly 90%, as compared to ATSP and 70% as compared to CCS.

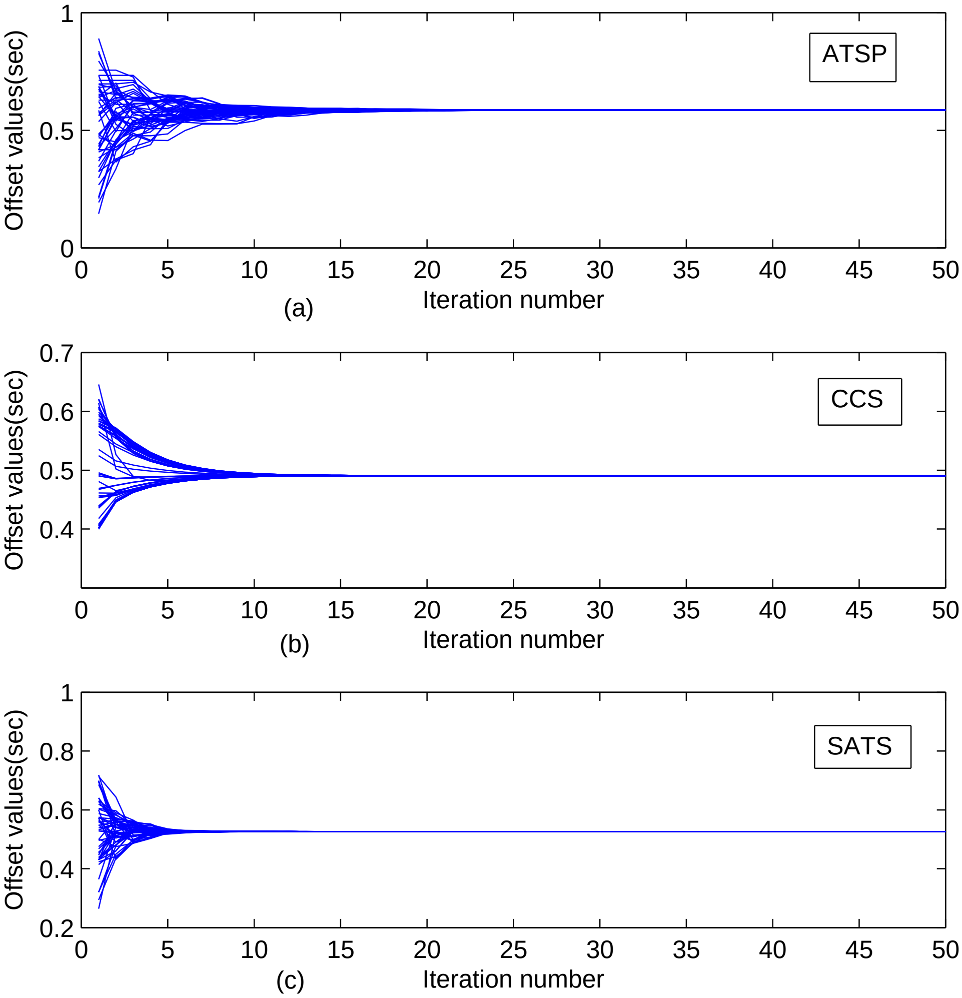


Figure 3.4: Offset convergence of (a) ATSP, (b) CCS, and (c) SATS

### (iii) Average local synchronization error

Fig. 3.6 (a)-(c) depicts average local synchronization error of each node for a topology of 50 nodes for 50 iterations. This shows the upper bound of local synchronization error of every node. It is observed that our algorithm has less local synchronization error, nearly 80%, as compared to ATSP and 82% as compared to CCS. So, local synchronization error is also optimized by our algorithm.

### (iv) Number of messages

To compare the average number of messages exchanged to achieve synchronization with the given error bound, the number of messages exchanged at each node is recorded and the average is computed for the whole network with different network size varying from 100-500 nodes. It is observed from Fig. 3.7 that our SATS algorithm has almost exchanged 50 % less

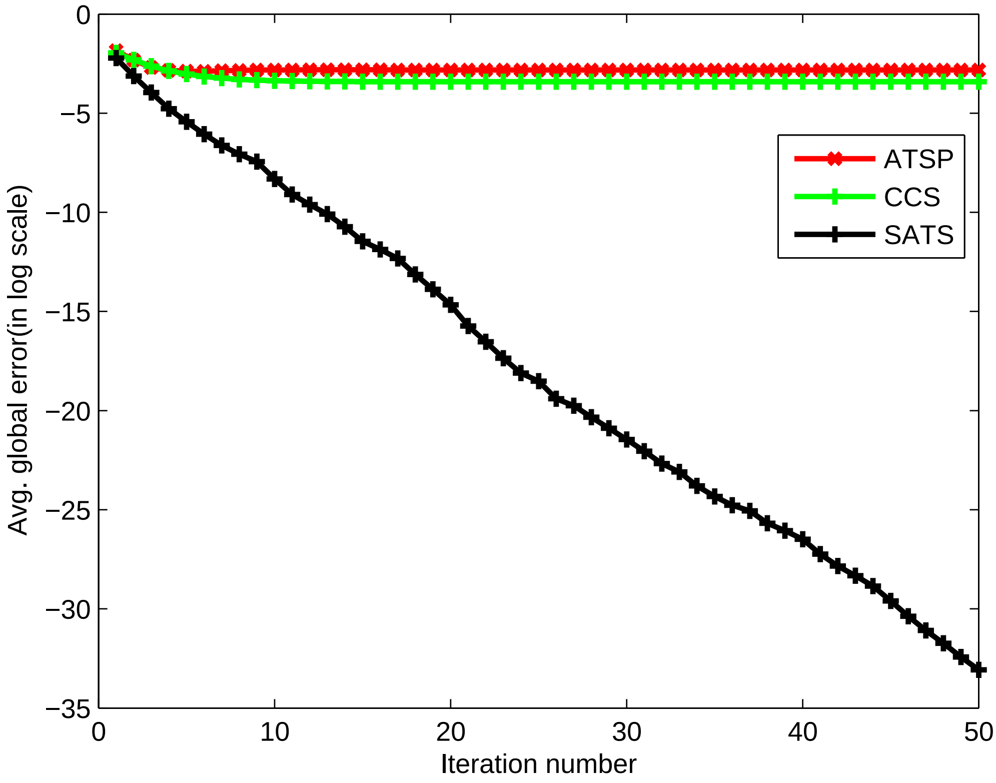


Figure 3.5: Average global synchronization error of ATSP, CCS, and SATS

messages than ATSP and 10% less messages than CCS algorithm.

Mathematical analysis shows that for a network of $`n'$ nodes, our SATS algorithm exchanges $(4n-2)$ number of messages per iteration which is comparatively higher than ATSP algorithm which exchanges $3n$ number of messages per iteration and CCS algorithm which exchanges $(3n-2)$ messages. But, due to faster convergence of our algorithm, the total number of messages exchanged is minimized in our case.

### (v) Energy consumption

The average energy consumption is estimated using the mathematical derivations obtained in section 3.4.4. The number of nodes is varied from 100-500. It is observed from Fig. 3.8 that in an average, our SATS algorithm has consumed 60 % less energy than ATSP and 20 % less energy than CCS algorithm. This is also due to faster convergence and hence, minimization of total iterations in our algorithm which has major impact on energy consumption of consensus-based synchronization algorithms.

### (vi) Impact of scalability

The scalable performance of the algorithms is tested according to two aspects. The first scalability test is based on increasing the network size and the second test is based on varying

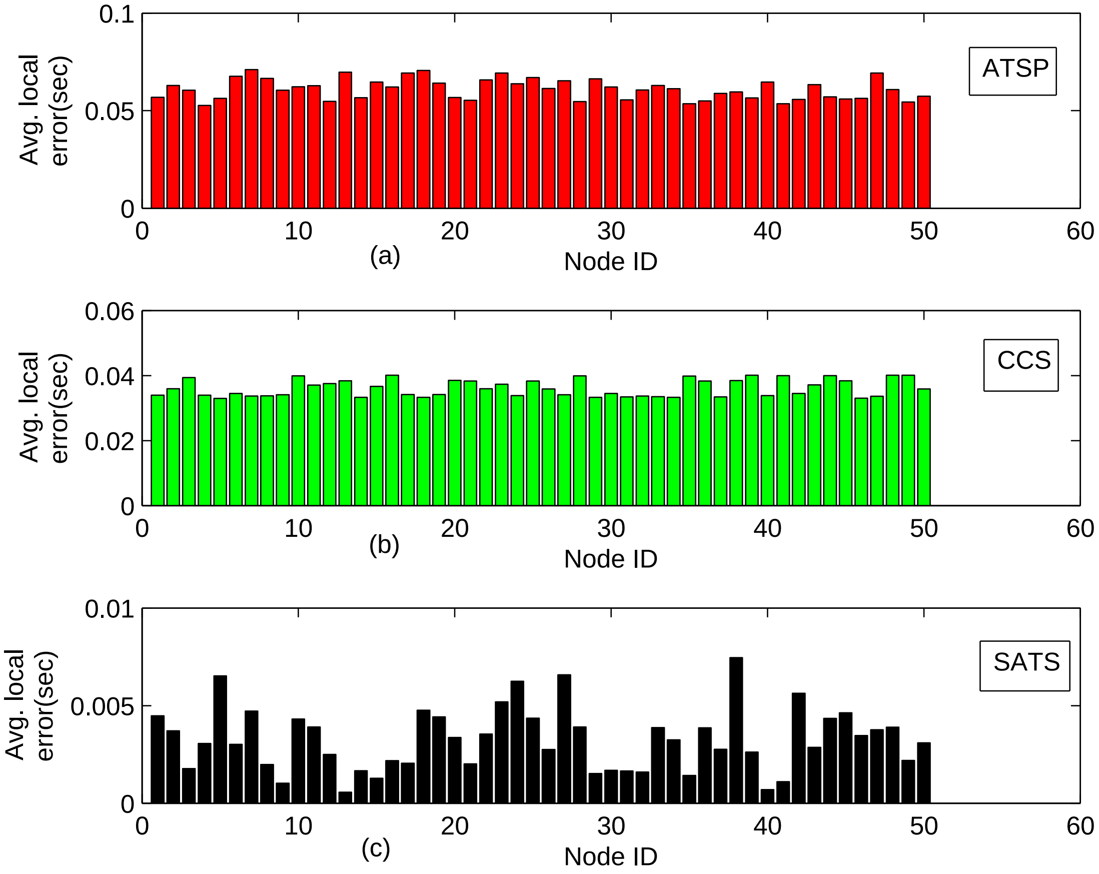


Figure 3.6: Average local error of individual node for (a) ATSP, (b) CCS and (c) SATS

the network density.

**(a) Scalability in terms of network size**

Fig. 3.9 and 3.10 show the behavior of the algorithms with the increase in network size. We have considered average number of iterations and average Mean Square Error (MSE) as performance metrics to evaluate the scalability of the algorithms.

The observations are made on random topologies, varying the number of nodes from 100-500 and defining the connecting radius accordingly. The average is calculated for 100 realizations of such random topologies for each number of nodes. From Fig. 3.9, it is observed that the mean of the average number of iterations is 18.182 for ATSP, 8.79 for CCS, and 4.31 for SATS. The standard deviation for ATSP is 1.16, 1.03 and for SATS, it is 0.71.

Similarly, from Fig. 3.10, it is observed that the mean of average MSE is 0.0048 for ATSP, 0.0042 for CCS, and 0.0017 for SATS. The standard deviation for ATSP is 0.0012, 0.0009 for CCS and, for SATS, it is 0.0005. So, it is inferred that the lower mean value of the proposed SATS shows its optimal performance, and lower standard deviation shows its

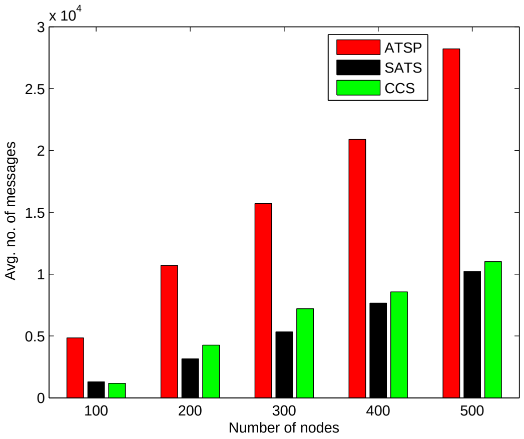


Figure 3.7: Average number of exchanged messages for ATSP, CCS and SATS

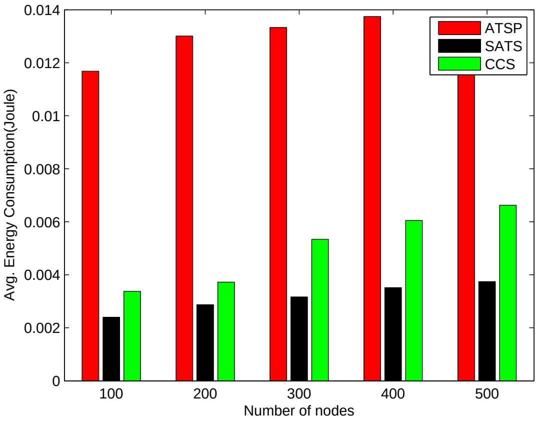


Figure 3.8: Average energy consumption Vs No.of nodes

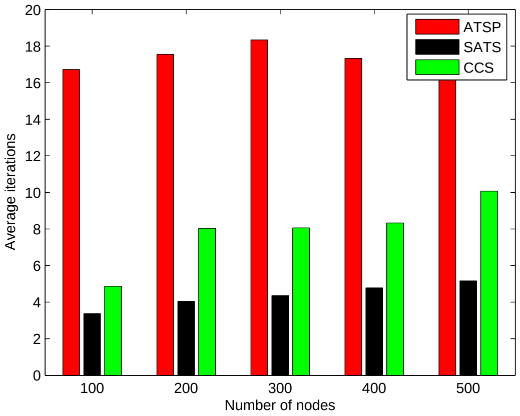


Figure 3.9: Average number of iterations Vs No.of nodes

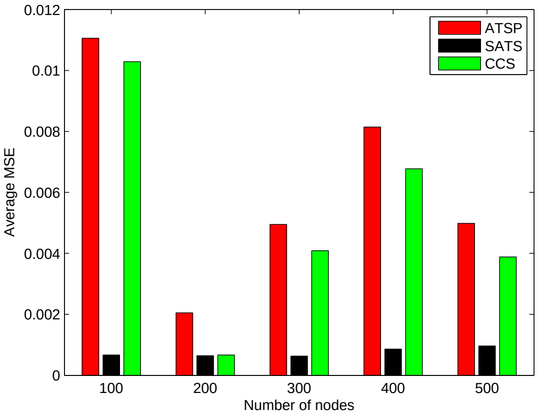


Figure 3.10: Average Mean Square Error Vs No.of nodes

consistency. Hence, our algorithm is more scalable than ATSP and CCS.

### (b) Scalability in terms of network density

The algorithms are also tested in a scenario where the node density is varied in a constant deployment area of $10 \times 10$ square unit. The radius of connectivity is set for 500 nodes, which is found to be 1.04 unit. The dense networks are created by increasing the number of nodes from 500-900, and the sparse networks are created by decreasing the number of nodes from 500-100. On sparse networks, it is found out from Fig. 3.11 that the mean of average MSE is 0.0107 for ATSP, 0.0129 for CCS, and 0.0008425 for SATS. The standard deviation is respectively 0.0117, 0.0174, and 0.00005172.

On dense networks, it is observed from Fig. 3.12 that the mean of average MSE is 0.0026 for ATSP, 0.0022 for CCS, and 0.0007934 for SATS. The standard deviation is respectively 0.0005415, 0.0015, and 0.00008337. In both types of networks, our SATS algorithm has a lower mean of MSE and lower standard deviation. This indicates that the SATS algorithm is more scalable.

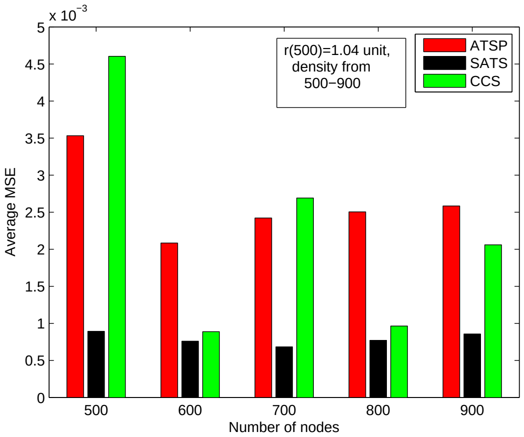


Figure 3.11: Average MSE Vs No.of nodes for dense topology

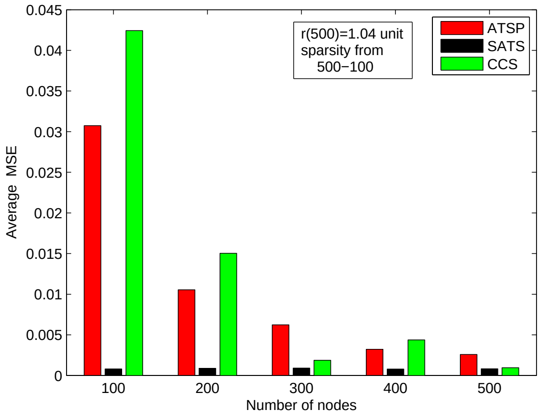


Figure 3.12: Average MSE Vs No.of nodes for sparse topology

## 3.6 Summary

A distributed, average consensus-based time synchronization algorithm (SATS) is proposed in this Chapter. It exploits maximum difference based, selective pair-wise averaging method for faster convergence and better synchronization accuracy. The optimality proof of the algorithm is carried out using the principle of convex optimization and greedy gossip theory. The consensus convergence analysis is done using doubly stochastic matrix properties. The asymptotic message complexity of the proposed algorithm is proved to be $O(n)$. A thorough energy consumption analysis is carried out for the proposed algorithm and also for the referred algorithms.

Simulation results show that the convergence speed of proposed SATS algorithm is 16 % faster than CCS and 50 % faster than ATSP. The network-wide synchronization error is minimized by 90 % than ATSP and 70 % than CCS. The local synchronization error is also improved by 80 % as compared to ATSP and 82 % as compared to CCS. Due to faster convergence, the average number of messages exchanged has shown significant improvement, nearly 50 % less than ATSP and 10 % less than CCS. The average energy consumption to achieve acceptable synchronization error is also minimized due to lesser number of messages exchanged. The SATS algorithm has consumed 60 % less energy than ATSP and 20 % less than CCS. The proposed SATS algorithm has shown consistent behavior with the increase in network size and variable network density. So, it is more scalable than ATSP and CCS. In the next Chapter, a multi-hop SATS algorithm is proposed using distributed dynamic programming approach for sparse and multi-hop networks.

# Chapter 4

# Multi-hop Consensus Time Synchronization Algorithm: A Distributed Dynamic Programming Approach

The recent consensus-based time synchronization algorithms are mostly one-hop in nature, i.e., every node communicates with its one-hop neighbors and performs offset or skew averaging to reach the consensus state or synchronized state. As per consensus theory, apart from the averaging scheme employed by the consensus algorithm, another factor that affects the consensus algorithms' performance is the topological connectivity of the networks. In topologies of lower degree of connectivity like sparse network, these one-hop consensus algorithms have exhibited poor performance in terms of convergence speed and accuracy. This requires the design of multi-hop consensus- based algorithm for WSN. In this context, we have proposed a multi-hop consensus based time synchronization algorithm for sparse, multi-hop WSN in this Chapter. A distributed, dynamic programming based approach is employed to propose the multi-hop average synchronization algorithms. Simulation results show that the proposed algorithm outperforms some one-hop consensus based time synchronization algorithms within a restricted hop count.

## 4.1 Introduction

In recent past, though some multi-hop extensions [42, 43, 58] to the state-of-the-art, single-hop synchronization mechanisms have been proposed, the multi-hop time synchronization is still challenging for various reasons [73]. Firstly, the number of hops become a major parameter for accurate protocols design. This is because the multi-hop synchronization error accumulates along the hops. For example, in case of RBS, the multi-hop error increases with the square root of the hops. In order to reduce the cumulative multi-hop error, the inherent delay must be bounded. Secondly, synchronization overhead (number of synchronization messages) is another challenge that multi-hop synchronization faces due to the limited power sources. In fact, the overhead is inversely proportional to synchronization accuracy.

Recent literature [42, 43, 58] reveals that multi-hop synchronization algorithms are available as an extension of reference based approach (sender-receiver synchronization, receiver-receiver synchronization) and overhearing based approach (pairwise broadcast synchronization). But, to the best of our knowledge, till now, no multi-hop algorithm is designed for consensus-based synchronization. Further, the multi-hop extension to reference based approach and overhearing based approach suffers from higher cumulative multi-hop error. So, utilizing multi-hop consensus based averaging approach can significantly reduce cumulative synchronization error.

The major challenges faced in the design of multi-hop consensus time synchronization algorithms are: (i) mechanisms for estimation and averaging of offset and skew of nodes which are multi-hop away for faster convergence and (ii) bounding the hop delay to improve the synchronization precision and to ensure consensus stability.

In Chapter 3, the proposed SATS algorithm selects a one-hop neighbor node which is having the maximum clock difference and performs pairwise averaging. This method shows optimal behavior as compared to some other consensus-based synchronization algorithms. In a densely deployed topology, the resultant communication topology is a completely connected graph. So, the probability of getting a maximum clock differed node at one hop in a completely connected graph is quite high. On the other hand, when the nodes are sparsely deployed, the maximum clock differed node may be located at multi-hop away. This motivates us to propose the multi-hop SATS algorithm.

The major contribution of this Chapter are the followings.

1. The feasibility of distributed dynamic programming technique on multi-hop SATS problem is studied thoroughly.
2. A novel multi-hop clock parameters estimation technique is proposed based on distributed, constraint-based dynamic programming approach.
3. The performance of the algorithm is analyzed mathematically, and extensive simulations are carried out to show the efficacy of the proposed multi-hop SATS algorithm.

## 4.2 System Models

In this Chapter, the same clock model and energy model is used as in Chapter 3. But, the network model is assumed to be a sparsely deployed random graph which is described below.

### 4.2.1 Network Model

For our proposed algorithm, the WSN is assumed to be a random, sparse graph *G= (V, E)*, where V denotes set of *'n'* nodes and set *'E'* represents the connectivity matrix between the nodes. $e_{ij}$ represents the entry in the $i^{th}$ row and $j^{th}$ column in *'E'*. *'E'* is a $n \times n$ adjacency matrix which is a sparse, Boolean matrix with $\tau \times n \times n$ number of uniformly distributed non-zero entries and $0 \leq \tau \leq 1$. All nodes have unique IDs. Communication channel between pair of node is assumed to be static, symmetric and undirected, i.e., upstream delay and downstream delay are same. If node $i$ and $j$ are neighbouring nodes, then $e_{ij} = e_{ji} = 1$. Otherwise, $e_{ij} = e_{ji} = 0$. Let, $N_i = j : (i, j) \in E$ denotes the set of one hop neighbours of node $v_i$. The communication topology is multi-hop and fully distributed where there is no special node such as root or reference node and all nodes execute the same algorithm.

## 4.3 Multi-hop SATS

This Section first formulates the multi-hop SATS problem and then illustrates its inherent advantage through an example.

### 4.3.1 Problem Formulation

The one-hop SATS algorithm which is presented in Chapter 3, mainly consists of two phases: (i) selection of the one-hop neighbor with maximum difference clock value and (ii) pairwise averaging. For any node '$i$' and its one-hop neighbor set '$N_i$', at $k^{th}$ iteration, these phases can be mathematically expressed as:

Phase-1: Maximum difference based selection:

$$max_{j \in N_i} \left\{ C_i^k(t) - C_j^k(t) \right\} \tag{4.1}$$

Phase-2: Pairwise averaging:

$$C_i^k(t) = C_j^k(t) = \frac{1}{2}\{C_i^k(t) + C_j^k(t)\} \tag{4.2}$$

Following the above principle of one hop SATS algorithm, the selection phase for multi hop SATS algorithm can be formulated as:

$$max \left[ \left\{ C_i^k(t) - C_j^k(t) \right\}_{j \in N_i^1}, \left\{ C_i^k(t) - C_l^k(t) \right\}_{l \in N_i^2}, ..., \left\{ C_i^k(t) - C_q^k(t) \right\}_{q \in N_i^m} \right] \tag{4.3}$$

where $N_i^1, N_i^2, ..., andN_i^m$ denote respectively the one-hop, two-hop, and m-hop neighbor sets of node '$i$'. After selecting the node using Equation 4.3, the node '$i$' performs phase-2 with the selected node, situated at multi-hop away.

### 4.3.2 Motivational Example

In Chapter 3, we have shown that selectively choosing a one-hop neighbor with maximum relative clock difference and performing pairwise averaging improves the consensus convergence and synchronization error significantly over other consensus-based synchronization algorithms. In fact, in a dense topology, there is a high probability to get the maximum clock differed neighbor at one hop in each iteration. Now, considering a case where the nodes are sparsely deployed, there is a high chance that the maximum differed node may be located at multi-hop away from the synchronization initiating node.

For example, consider the following topology given in Fig. 4.1, consisting of 9 nodes and assume that the clock values at a particular instance are denoted by $C_{node\_id}(t)$. Suppose, node 'A' initiates the synchronization algorithm. If it searches for maximum differed clock only within one-hop, then it will select node 'C'. Assume that the initial clock values at all the 9 nodes, respectively, are {A, B, C, D, E, F, G, H, I}={1, 2, 3, 4, 5, 6, 7, 8, 9} and thus, average of initial clock value is 5. Then after performing pairwise averaging with 'C' by node 'A', the updated clock values will be { 2, 2, 2, 4, 5, 6, 7, 8, 9} with a variance of 6.44. If the search by node 'A' will be expanded to two-hop, then it will select node 'E' for pairwise averaging. So, for the two-hop case, the updated clock values will be { 3, 2, 3, 4, 3, 6, 7, 8, 9} with a variance of 5.7 which shows that the two-hop averaging will enhance the convergence to the initial value.

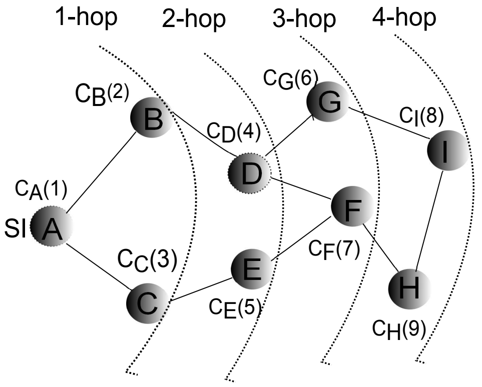


Figure 4.1: Effect of 1-hop averaging Vs. 2-hop averaging

This shows that expanding the search for maximum differed node to a higher neighborhood (multi-hop) can improve the convergence. At the same time, increasing the number of hops will incur hop delay which affects the synchronization precision and hence, hamper the speed of convergence. So, the end-to-end delay must be taken into account to design an efficient multi-hop SATS algorithm.

In the following Section, we have proposed a distributed dynamic programming based approach to solve the multi-hop SATS problem.

## 4.4 Proposed Distributed Dynamic Programming Approach

In this Section, we first present the feasibility of dynamic programming approach for the multi-hop SATS problem and then provide a detailed solution using this approach.

### 4.4.1 Overlapping Sub-structure

The dynamic programming paradigm is applicable to a problem which can be modeled as a multi-stage decision problem and can be decomposed into a number of overlapping sub-problems or sub-structures. The overlapping sub-problems are then solved using a recursive relation, satisfying the principle of optimality [74]. The principle of optimality applies if the optimal solution to a problem always contains optimal solutions to all sub-problems. The following lemma proves that the multi-hop clock parameter estimation follows overlapping sub-structures.

**Lemma 4.4.1.** *The multi-hop clock parameter estimation for multi-hop SATS problem follows overlapping sub-structures.*

*Proof.* The multi-hop SATS problem involves in the process of selecting a node which is multi-hop away from the initiating node and should have maximum relative clock skew or offset. In other words, the multi-hop path, thus, established between the initiating node and the selected node will have the maximum sum of offset among all the possible paths from the initiating node to all multi-hop neighbors. Thus, the problem falls into a decision-making problem of selecting the maximum sum-of-offset path.

Further, the initiating node can not estimate the relative offset with the multi-hop neighbors directly in a sparsely deployed network. It has to go through a cascading of message exchanges with the one-hop neighbors as shown in Fig. 4.2. Hence, the multi-hop relative offset or skew estimation problem consists of a series of hop-wise estimation (sub-problems) which can be recursively derived as follows [58].

Let, $\alpha_{n_i \to n_j}$ and $\beta_{n_i \to n_j}$ denote relative skew and offset between two nodes $n_i$ and $n_j$ respectively. Then, the time-stamps at node $n_j$ w. r. to $n_i$ is given by $t_{n_j} = \alpha_{n_i \to n_j} t_{n_i} + \beta_{n_i \to n_j}$. Thus, time-stamps on a multi-hop path can be derived as:

$$t_{n_{i+1}} = \alpha_{n_i \to n_{i+1}} t_{n_i} + \beta_{n_{i+1} \to n_i}, i \in \{1, 2, .., m-1\} \tag{4.4}$$

By repeatedly substituting the values for time-stamps in Equation 4.4, the multi hop skew and offset are given as:

$$\alpha_{n_1 \to n_m} = \prod_{i=1}^{m-1} \alpha_{n_i \to n_{i+1}} \tag{4.5}$$

$$\beta_{n_1 \to n_m} = \sum_{i=2}^{m-1} \left[ \left( \prod_{j=2}^{i} \alpha_{n_{j-1} \to n_j} \right) \beta_{n_i \to n_{i+1}} \right] + \beta_{n_1 \to n_2} \tag{4.6}$$

The Equations 4.5 and 4.6 clearly indicates that the estimation of $m$-hop parameters (both skew and offset) is dependent on $(m-1)$-hop parameter values, and hence, the estimation method follows overlapping of sub-problems. This proves Lemma 4.4.1. □

Dynamic programming paradigm can be applied in a distributed way [75]. Since the multi-hop parameter estimation has to be performed at each node in a distributed way, the problem can be effectively modeled using distributed dynamic programming approach. Now, in order to minimize synchronization error and ensure consensus convergence, the multi-hop maximum parameter estimation and pairwise averaging must be carried out within a bounded delay. To bound the end-to-end delay for consensus stability, the author in [76] has derived a relationship between hop number, per hop delay and topological connectivity property of the network which is given as the following Lemma.

**Lemma 4.4.2.** *[76] For a connected network, the m-hop Request-Reply based consensus protocol is globally asymptotically stable for each '$m$', if it satisfies the following condition.*

$$D_{max} \leq \frac{\pi}{2m\overline{\lambda}_M} \tag{4.7}$$

*with '$m$' number of hop, $D_{max} = max_j \left\{D_j\right\}, \overline{\lambda}_M = max_j \left\{\lambda_M \left(L_j\right)\right\}$ for j=1, 2,..., m where $D_j$ denotes maximum hop delay at $j^{th}$ hop, $\lambda_M$ denotes largest eigenvalue of Laplacian matrix $L_j$ of j-hop graph.*

Using the above Lemma, the threshold delay $D_{Th}$ can be set to $D_{max}$ to ensure consensus stability. The following section describes the recursive formulations of the parameters (skew and offset) estimation and end-to-end delay.

### 4.4.2 Recurrence Relation Formulation

Let $\alpha^i_m(k)$ be the skew value at node '$i$' with respect to its m-hop neighbors at $k^{th}$ iteration. Since, multi-hop skew is a product of hop-wise skew values as shown in Equation 4.5, it can be recursively defined as:

$$\alpha^i_m(k) = \begin{cases} j \in N_i\{\alpha_{ij} \times \alpha^j_{m-1}(k)\}, \text{ if } m > 1 \\ \alpha_{ij}, \text{ if } m = 1 \end{cases} \tag{4.8}$$

where $\alpha_{ij}$ denotes the skew difference at node '$i$' with respect to its one-hop neighbors '$j$' which can be estimated directly, using the two-way message exchange schemes shown in Fig. 3.2 in Chapter 3 and hence, acts as the base condition for the recurrence relation.

Similarly, the additive nature of multi-hop offset value as given in Equation 4.6 can be recursive defined as:

$$\beta^i_m(k) = \begin{cases} j \in N_i\{\beta_{ij} + \beta^j_{m-1}(k)\}, \text{ if } m > 1 \\ \beta_{ij}, \text{ if } m = 1 \end{cases} \tag{4.9}$$

where $\beta_{ij}$ denotes the offset difference at node '$i$' with respect to its one-hop neighbors '$j$' which can be estimated directly, using the two-way message exchange schemes discussed in

Chapter 3 and hence, acts as the base condition for the recurrence relation.

Further, the multi-hop end-to-end delay at iteration *'k'* also can be recursively defined as follows.

$$Delay^i_m(k) = \begin{cases} j \in N_i\{\delta_{ij} + Delay^j_{m-1}(k)\}, \text{if } m > 1 \\ \delta_{ij}, \text{if } m = 1 \end{cases} \tag{4.10}$$

where $\delta_{ij}$ denotes the one-hop delay which can be computed directly, using two-way message exchange scheme, as $((T^i_4 - T^j_3) + (T^j_2 - T^i_1))/2$. This acts as the base condition for the recursive delay estimation.

The multi-hop SATS algorithm involves in the process of selecting a multi-hop neighbor node with the maximum parameter (skew and offset) along with the constraint that the maximum end-to-end delay lies within the threshold delay given in Lemma 4.4.2. So, the objective function for distributed, multi-hop SATS algorithm and the constraint for end-to-end delay can be stated, using the principle of optimality, as follows.

**Objective function:** $\forall i \in V$, Maximize $\alpha^i_m(k)$ and $\beta^i_m(k)$ which can be expressed recursively, using Equation 4.8 and Equation 4.9, as:

$$Max\ \alpha^i_m(k) = \begin{cases} j \in N_i\{\alpha_{ij} \times Max\ \alpha^j_{m-1}(k)\},\ \text{if } m > 1 \\ \alpha^{max}_{ij},\ \text{if } m = 1 \end{cases} \tag{4.11}$$

$$Max\ \beta^i_m(k) = \begin{cases} j \in N_i\{\beta_{ij} + Max\ \beta^j_{m-1}(k)\},\ \text{if } m > 1 \\ \beta^{max}_{ij},\ \text{if } m = 1 \end{cases} \tag{4.12}$$

**Constraint:** subject to $Max\ Delay^i_m(k) \leq D_{Th}$ where maximum end-to-end delay at iteration *'k'* is estimated as follows.

$$Max\ Delay^i_m(k) = \begin{cases} j \in N_i\{\delta_{ij} + Max\ Delay^j_{m-1}(k)\}, \text{if } m > 1 \\ \delta^{max}_{ij}, \text{if } m = 1 \end{cases} \tag{4.13}$$

Though Equation 4.11 and 4.12 represent distinct methods of parameter estimation, they refer to the same m-hop neighbor which is already discussed in Chapter 3 that the node with maximum relative skew has also maximum relative offset. Hence, the objective function can be precisely defined using the atomic clock value $C^i_m$ as: $\forall i \in V, Max\ C^i_m(k)$ *s. t.* $Max\ Delay^i_m(k) \leq D_{Th}$.

## 4.5 The multi-hop SATS Algorithm

To implement the proposed distributed, dynamic programming approach, the corresponding message passing model for m-hop SATS can be designed as shown in Fig. 4.2 which can be realized at every node in the network. The synchronization round at every node is divided into two phases: phase-1 is for parameter (offset and skew) estimation, and phase-2 is for parameter compensation. Both offset and skew estimation are performed in the same synchronization round.

To satisfy the constraint given in Equation 4.13, the duration of synchronization round for every node is set to the threshold delay value ($D_{Th}$) which is estimated centrally by the sink node and disseminated to every node in the network. Phase-1 of a node *'$i$'* is assigned with a random duration of time which is given by $t^i_{rand}$=uniform (0, $D_{Th}$-$\kappa$) where $\kappa$=constant, 0< $\kappa$<$D_{Th}$. Thus, the duration of phase-2 is given by $D_{Th}$-$t^i_{rand}$.

Each node in the network, at its local oscillation period, broadcasts a synchronization initiating message (SYN_INIT) to its one-hop neighbors and waits for a random duration $t^i_{rand}$ to receive acknowledgment messages. The (SYN_INIT) message, along with normal information like node ID, current time-stamp, contains additional information, i.e., the current hop count ($h$) and maximum hop count ($h_{max}$). The initiating node initially sets $h = 0$ and $h_{max} = m$. The one-hop neighbors after receiving SYN_message, increment $h$ by 1 and compare with $h_{max}$. If $h < h_{max}$, the one-hop neighbors further send the SYN_INIT requests to their upstream one-hop neighbors. This process continues up to '$m$' hop, i.e., until $h = h_{max}$.

Then, the nodes at $m^{th}$ hop reply back by sending SYN_ACK messages to their requester at $(m-1)^{th}$ hop. The requester upon receiving the SYN_ACK message compute the clock parameters (skew and offset) and the delay. Then it sends the estimated maximum values along with the corresponding neighbor's node ID to the downstream requester. This process continues until the initiating node receives the acknowledgment message. Thus, a synchronization tree is established at each node as shown in Fig. 4.4 whose root node is the synchronization initiating node and its depth is bounded by '$m$'.

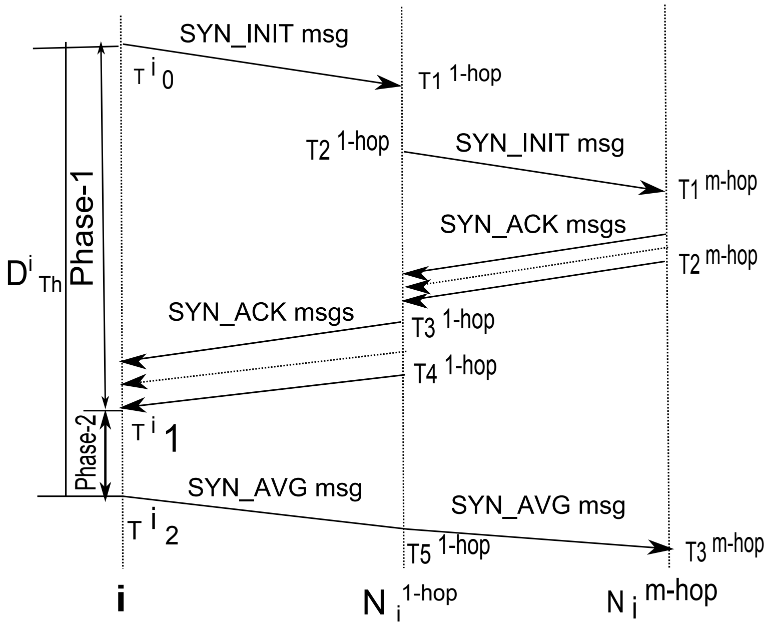


Figure 4.2: Message passing model for multi-hop SATS

Finally, the initiating node estimates the maximum multi-hop skew, offset, and end-to-end delay using Equation 4.11, 4.12 and 4.13 respectively. If the maximum estimated end-to-end delay is within the assigned interval for phase-1, then it selects the multi-hop neighbor having maximum clock value and performs a pairwise averaging with the selected node by sending a synchronization averaging (SYN_AVG) message within the assigned duration of phase-2, along the shortest route. The pseudo code of multi-hop SATS algorithm is given below.

### 4.5.1 Algorithm Illustration

To explain the working of proposed multi-hop SATS algorithm, an example network of 16 nodes is considered as shown in Fig. 4.3. Let the multi-hop SATS algorithm is executed at synchronization initiating node ‘A’ up to 2 hops. Fig. 4.4 shows the synchronization tree rooted at node ‘A’ of Fig. 4.3. For the sake of simplicity, we have considered the atomic clock ‘C’ value instead of individual skew ($\alpha$) and offset ($\beta$) to illustrate the algorithm. In fact, the clock with maximum skew and offset also has maximum atomic value. The ($t$) value at each node denotes the atomic clock value, and the edge weights denote the estimated round trip delay between a pair of nodes. The threshold delay ($D_{Th}$) is estimated as follows. For ease of estimation of graph-related parameters, a MATLAB tool known as ‘matgraph’ is used.

The Laplacian of the 1-hop graph (example network), $L_1$, is estimated using the adjacency matrix and diagonal matrix of the network. Its largest eigen value $\lambda_M(L_1)$ is computed to be 5.7537. Similarly, the Laplacian of the 2-hop graph $L_2$ is estimated using the adjacency matrix and diagonal matrix of the 2-hop graph of the given network. The

**Algorithm 4** Distributed DP based Multi-hop SATS ($v_i$)

```
Input: Acceptable synchronization error ($\epsilon$), hop count ($m$), Threshold delay ($D_{Th}$)
Output: Consensus state ($C_i(0)$)
Sink node computes $D_{Th}$=$\frac{\pi}{2m\bar{\lambda}_M}$ and disseminate to each node 'i' /***m-hop threshold delay estimation using Lemma 4.4.2***/
while local synchronization error $\leq \epsilon$ do
    for all node $i \in V$ do
        Compute $t^i_{rand}$=rand(0, $D_{Th}$-$\kappa$), $\kappa$=constant, 0< $\kappa$<$D_{Th}$
        Assign duration of Phase-1 at node '$i$'=$t^i_{rand}$
        for Duration=$t^i_{rand}$ do
            /************* Phase-1**************/
            for $hop$=1 to $m$ do
                Node '$i$' broadcasts SYN _INIT message to $N_i^{hop}$
                For $j \in N_i^{hop}$ and SYN_INIT.received($j$)=TRUE
                $Node\ i = Node\ j$
                Continue
            end for
            for $hop$=$m$ downto 1 do
                For $j \in N_i^{hop}$ and SYN_INIT.received($j$)=TRUE
                $Node\ j$ sends SYN_ACK to the requesters (r) at $(m-1)^{th}$ hop
                Requesters (r) at $(m-1)^{th}$ compute $\alpha^{max}_{rj}$, $\beta^{max}_{rj}$, and $\delta^{max}_{rj}$
                Requesters (r) send $< \alpha^{max}_{rj}, \beta^{max}_{rj}, \delta^{max}_{rj}, node\_ID\_max >$ in SYN _ACK messages to downstream requesters
                Continue
            end for
            $Node\ i$ estimates $max\ C^i_m$, and $max\ D^i_m$ using Equations 4.11, 4.12 and 4.13
        end for
        /************* Phase-2**************/
        if $max\ D^i_m \leq t^i_{rand}$ then
            Selects the m-hop node with max $C^i_m$
            Within Duration=$D_{Th} - t^i_{rand}$
            $Node\ i$ sends SYN_AVG message to m-hop selected node
        end if
    end for
end while
```

2-hop graph's adjacency matrix is computed by first finding out a path matrix of length 2 for the given network and then replacing its non-zeroes value by 1 and setting all diagonal entries as 0. Its largest eigen value $\lambda_M(L_2)$ is found out to be 9.5504. Hence, $\overline{\lambda}_M$ is 9.5504. Thus, the $D_{max}$ using Lemma 4.4.1 is estimated as 0.17. Let, the duration of phase-1 using the threshold delay is set to 0.11 and phase-2 is set to 0.06. Then, the execution of proposed multi-hop SATS algorithm will proceed as follows.

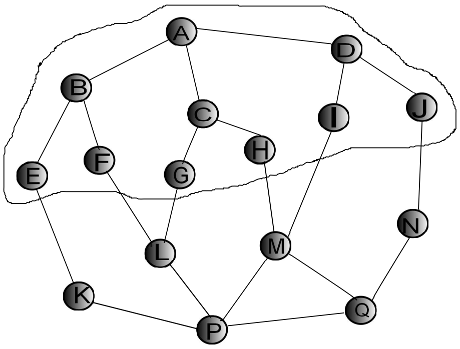


Figure 4.3: An example network of 16 nodes

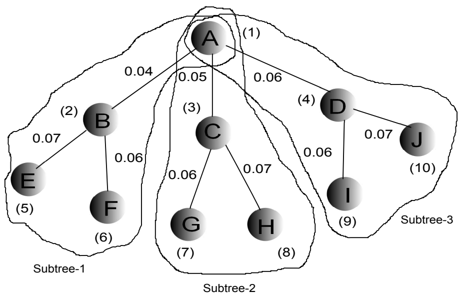


Figure 4.4: The 2-hop sync tree rooted at 'A' of Fig. 4.3

Node 'A' broadcasts the SYN_INIT message to B, C, and D. Then, SYN _INIT message is forwarded by B, C, and D to their upstream neighbors E, F, G, H, I, and J and the messages are reached at maximum hop ($h_{max} = m = 2$). Let, the SYN_ACK messages are sent from leaf nodes (nodes at m=2) to their corresponding requesters. For E and F, the corresponding requester is B. Similarly, for G and H, it is C and for I and J, it is D. After, receiving the ACK messages, let, the estimated delays are as shown in Fig. 4.4 as edge weight. Then, using the dynamic programming approach, the maximum parameters estimation is done as follows.

For the constraint to satisfy, the delay estimation up to 2 hops using Equation 4.13 at different sub-trees rooted at 'A' is illustrated as follows.

$$\begin{aligned}
Delay_2^A &= j \in N_A\{\delta_{Aj} + max\ Delay_1^j\} \\
&= \{\delta_{AB} + max\ Delay_1^B, \delta_{AC} + max\ Delay_1^C, \\
&\quad\ \delta_{AD} + max\ Delay_1^D\} \\
&= \{\delta_{AB} + max\ \{Delay_{BE}, Delay_{BF}\}, \\
&\quad\ \delta_{AC} + max\ \{Delay_{CG}, Delay_{CH}\}, \\
&\quad\ \delta_{AD} + max\ \{Delay_{DI}, Delay_{DJ}\} \\
&= \{0.04 + 0.07, 0.05 + 0.07, 0.06 + 0.07\} \\
&= \{0.11, 0.12, 0.13\}
\end{aligned}$$

From the above delay estimation, the maximum end-to-end round-trip delay along sub-tree 1, sub-tree 2, and sub-tree 3 are respectively, 0.11, 0.12, and 0.13. Similarly, the maximum clock value estimation at node 'A' up to 2 hop is given below.

$$\begin{aligned}
max\ (t)_2^A &= max_{j \in N_A}\{(t)_{Aj} + max\ (t)_1^j\} \\
&= max\ \{(t)_{AB} + max\ (t)_1^B, (t)_{AC} + max\ (t)_1^C, \\
&\quad (t)_{AD} + max\ (t)_1^D\} \\
&= max\ \{(t)_{AB} + max\ \{(t)_{BE}, (t)_{BF}\}, \\
&\quad (t)_{AC} + max\ \{(t)_{CG}, (t)_{CH}\}, (t)_{AD} + max\ \{(t)_{DI}, (t)_{DJ}\}, \\
&= max\ \{(t)^A - (t)^B + max\ \{(t)^B - (t)^E, (t)^B - (t)^F\}, \\
&\quad (t)^A - (t)^C + max\ \{(t)^C - (t)^G, (t)^C - (t)^H\}, \\
&\quad (t)^A - (t)^D + max\ \{(t)^D - (t)^I, (t)^D - (t)^J\}\} \\
&= max\ \{1 + 4, 2 + 5, 3 + 6\} = 9
\end{aligned}$$

From the above clock estimation, the maximum 2-hop clock value is 9 which is along path A-D-J which is in sub-tree 3. But, delay estimation shows that the end-to-end round trip delay in sub-tree 3 exceeds the duration of phase 1 which is 0.11. So, A-D-J path is not selected though it is an optimal path. The similar case happens with sub-tree 2. Instead, the path A-B-F is chosen in sub-tree 1 because it satisfies the constraint and hence, node A will perform pairwise averaging with node F.

## 4.6 Performance Analysis

The proof of optimality of synchronization error and consensus convergence of multi-hop SATS algorithm is same as discussed in Chapter 3 because the basic principle of averaging remains the same, i.e., maximum difference based pair-wise averaging and the estimation and compensation processes are carried out within the delay bound as per Lemma 4.4.2. The following subsections analyze the message complexity and energy consumption of multi-hop SATS algorithm.

### 4.6.1 Message Complexity

This subsection analyzes the message complexity of the multi-hop SATS algorithm.

**Theorem 4.6.1.** *The distributed, multi-hop SATS algorithm has asymptotic message complexity* $O(n(\log n)^m)$ *where 'n' is number of sensor nodes in the network and 'm' is the hop number.*

*Proof.* The multi-hop SATS algorithm will be initiated at every node of the network. Thus, it will generate *'n'* number of synchronization trees with depth *'m'*. Let the whole network has an average degree of connectivity *'K'*. Hence, the number of SYN_INIT messages broadcasted in a synchronization tree up to '$m$' hop (depth of the tree) will be derived as follows.

$$\begin{aligned} &= 1 + \theta(K) + \theta(K^2) + ... + \theta(K^{m-1}) \\ &= \theta\Big(\frac{K^{m+1}-1}{K-1}\Big) \end{aligned} \tag{4.14}$$

Similarly, the number of SYN_ACK messages replied from the child nodes to the root (SI node) of the synchronization tree will be:

$$\begin{aligned} &= \theta(K^m) + \theta(K^{m-1}) + ... + \theta(K) \\ &= \theta(K(K^{m-1} + K^{m-2} + ... + 1)) \\ &= \theta\Big\{K\Big(\frac{K^m-1}{K-1}\Big)\Big\} \end{aligned} \tag{4.15}$$

Then, the SI node will send a single SYN_AVG message to the m-hop selected node to perform pairwise averaging. So, the number of SYN_AVG message is $\theta(1)$.

For the whole network total number of messages exchanged will be:

$$\begin{aligned} &= O\Big(n\Big(\frac{K^{m+1}-1}{K-1}\Big)\Big) + O\Big(n\Big(K\Big(\frac{K^m-1}{K-1}\Big)\Big) + O(n) \\ &= O\Big(n\Big(1 + \frac{K^{m+1}-1}{K-1} + K\Big(\frac{K^m-1}{K-1}\Big)\Big) \\ &= O\Big(n\Big(\frac{K^{m+1}-1}{K-1}\Big)\Big) \end{aligned} \tag{4.16}$$

In a random, sparse graph, the relation between the average degree of connectivity and number of nodes is given by $K = \theta(\log n)$. Hence, substituting the value of *'K'* in Equation 4.16, we have

$$\begin{aligned} &= O\Big(n\Big(\frac{(\log n)^{m+1}-1}{(\log n - 1)}\Big)\Big) \\ &= O\Big(n\Big(\frac{(\log n - 1)((\log n)^m + (\log n)^{m-1} + ... + 1)}{(\log n - 1)}\Big)\Big) \\ &= O(n(\log n)^m) \end{aligned} \tag{4.17}$$

This proves that the message complexity of multi-hop SATS algorithm is $O\ (n(\log n)^m)$.
□

### 4.6.2 Energy Consumption Analysis

In Chapter 3, the energy consumption of one-hop SATS algorithm is investigated at each node for sending and receiving synchronization messages. But, in multi-hop SATS algorithm, the synchronization messages traverse across m-hop paths. So, the energy consumption in a particular iteration is the sum of energy consumption at intermediate nodes which constitutes the path. So, for each type of synchronization message, the energy consumption across an m-hop path can be derived recursively as follows.

Using the energy model given in Section 3.2, the following Equations can be derived.

For broadcasting a SYN_INIT message up to m-hop neighbors, the energy consumption across the m-hop paths originating at node *'$i$'*, $P_{tx}^{SYN\_INIT}(i, m)$, is given by:

$$P_{tx}^{SYN\_INIT}(i,m) = \begin{cases} j \in N_i\{P_{tx}^{SYN\_INIT}(i) + P_{tx}^{SYN\_INIT}(j,m-1)\}, \text{ if } m > 1 \\ P_{tx}^{SYN\_INIT}(i), \text{ if } m = 1 \end{cases} \tag{4.18}$$

where $P_{tx}^{SYN\_INIT}(i)$ is estimated using Equation 3.24 given in Chapter 3. Similarly, for receiving SYN_ACK messages across the m-hop path terminating at node *'i'* is given by:

$$P_{rx}^{SYN\_ACK}(i,m) = \begin{cases} j \in N_i\{P_{rx}^{SYN\_ACK}(i) + P_{tx}^{SYN\_ACK}(j,m-1)\}, \text{ if } m > 1 \\ P_{rx}^{SYN\_ACK}(i), \text{ if } m = 1 \end{cases} \tag{4.19}$$

where $P_{rx}^{SYN\_ACK}(i)$ is estimated using Equation 3.25 given in Chapter 3. Finally, to send the SYN_AVG message by node *'i'* to a selected node at m-hop away through a shortest path is given by:

$$P_{tx}^{SYN\_AVG}(i,m) = \begin{cases} j \in N_i\{P_{tx}^{SYN\_AVG}(i) + P_{tx}^{SYN\_AVG}(j,m-1)\}, \text{ if } m > 1 \\ P_{tx}^{SYN\_AVG}(i), \text{ if } m = 1 \end{cases} \tag{4.20}$$

where $P_{tx}^{SYN\_AVG}(i)$ is estimated as follows.

$$P_{tx}^{SYN\_AVG}(i) = M(\beta_1 + \beta_2\{min\ l(i,j), j \in N_i\}^{\zeta}) \tag{4.21}$$

So, the total energy consumption per iteration for a network of *'n'* nodes is given by:

$$P_{total} = \sum_{i=1}^{n} \left[P_{tx}^{SYN\_INIT}(i,m) + P_{rx}^{SYN\_ACK}(i,m) + P_{tx}^{SYN\_AVG}(i,m)\right] \tag{4.22}$$

To compute the average energy consumption in a network of *'n'* nodes for m-hop SATS algorithm, the number of m-hop paths need to be found out. Given, the adjacency matrix of the network as *'A'*, the number of m-hop paths, $N_{m-hop}$, can be computed as:

$$N_{m-hop} = \sum_{k=2}^{m} \sum_{ij=1}^{n} (A^k)_{ij} \tag{4.23}$$

Hence, the average energy consumption per iteration can be estimated as:

$$P_{iteration} = P_{total}/N_{m-hop} \tag{4.24}$$

Thus, average energy consumption to achieve network-wide synchronization is given by:

$$P_{avg} = P_{iteration}\frac{1}{n}\sum_{i=1}^{n} It(i) \tag{4.25}$$

where $It\ (i)$ is the number of iterations required at node *'$i$'* to reach the acceptable synchronization error bound.

## 4.7 Simulation

In this Section, the proposed dynamic programming based multi-hop SATS algorithm is evaluated and compared with some standard one hop consensus based time synchronization algorithms like ATSP [22], CCS [17], and proposed one hop SATS algorithm in Chapter 3. The performance is evaluated using some relevant performance metrics such as convergence speed, average global synchronization error, average local synchronization error, average number of messages exchanged, average energy consumption, Impact of number of hops, and scalability. The algorithms are implemented in PROWLER simulator, a MATLAB based simulator which is a simple yet strong simulator designed for wireless sensor network. The simulation parameters used are shown in Table 4.1.

### 4.7.1 Simulation Configuration

The simulations are performed on random, sparse topology by varying the number of nodes from 50-250. Since the multi-hop SATS algorithm is targeted for the sparse network which requires multi-hop communication; to realize the sparse network, the $\tau$ parameter mentioned in the network model is set between 0.1-0.5. As per TelosB data sheet specification mentioned in [18], the typical skew range is between -5 PPM to 5 PPM. So, to have a close resemblance with the realistic environment, the skew is generated in the specified range using random uniform distribution.

To have a fair comparison with ATS [22] and CCS [17], the clock offsets are generated using random uniform distribution between 0 and 1 which is same as specified in [22]. The duration for one iteration is set to 10 seconds. The default MAC protocol provided in PROWLER simulator is CSMA/ CA. The hop count *'m'* varies from 2-4. The theoretical threshold delay $D_{Th}$ is calculated by the sink node centrally and disseminated to the each node of the network where every node executes the same algorithm in a distributed fashion.The algorithm has incorporated the usage of MAC layer time stamps [67], appended in the control frames (e.g. IEEE 802.11 RTS/ CTS control packets for CSMA/CA) and the averaged parameter's (offset/ skew) value in the actual data frame, to minimize the MAC delay.

Table 4.1: Simulation Parameters

| *Parameter* | *Values* |
|---|---|
| Deployment area | 10 × 10 square unit |
| Topology | Random, Sparse |
| No. of nodes ($n$) | 50-250 |
| Hop Count ($m$) | 2-4 |
| Initial skew ($\alpha$) | uniform(-5,5) |
| Initial offset ($\beta$) | uniform(0,1) |
| Acceptable synchronization error ($\epsilon$) | 0.0001 sec. |
| Iteration interval | 10 sec |
| Path Loss Exponent ($\zeta$) | 2 |
| $\beta_1$ | 45 nJ/bit |
| $\beta_2$ | 10 pJ/bit |
| $\gamma$ | 35 nJ/bit |
| Message size ($M$) | 320 bits |
| MAC Protocol | CSMA/CA |

## 4.7.2 Simulation Results and Analysis

The following Sections show the performance of the proposed algorithm with respect to various performance metrics.

### (i) Skew convergence

To test the skew convergence, the observations are recorded by considering maximum 50 iterations as shown in Fig. 4.5 (a)-(d). It is observed that ATSP [22] algorithm achieves skew convergence at around 20 iterations, CCS [17] algorithm which uses cumulative weighted averaging (CWA) method convergences at around 15 iterations, and 1-hop SATS algorithm takes around 12 iterations, all are with an acceptable synchronization error ($\epsilon$) of 0.0001 sec. Whereas the 2-hop SATS algorithm, as shown in Fig. 4.5 (d), achieves skew convergence within 10 iterations for the same value of acceptable synchronization error. So, convergence is faster in 2-hop SATS algorithm.

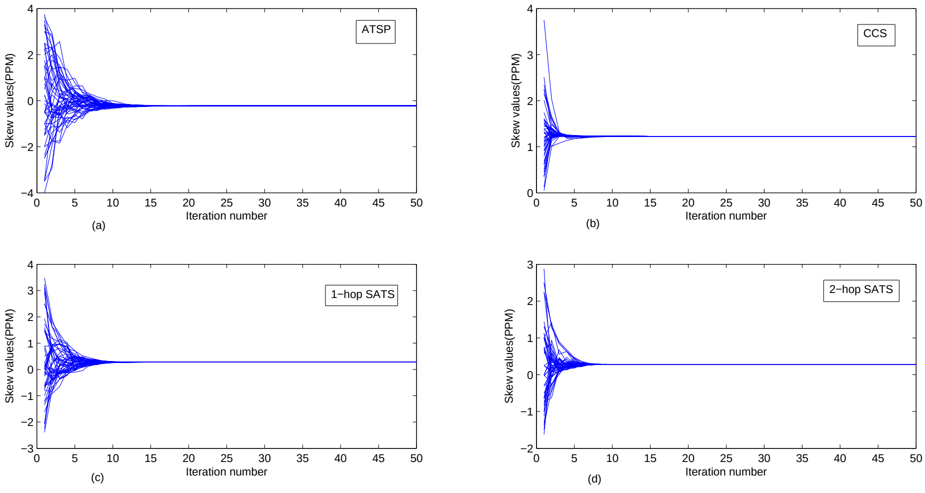


Figure 4.5: Skew convergence of (a) ATSP, (b) CCS, (c) 1-hop SATS, and (d) 2-hop SATS

### (ii) Offset convergence

The offset convergence is also tested and observed simultaneously with the skew convergence with the same value of acceptable synchronization error as shown in Fig. 4.6 (a)-(d). The initial average of random offset distribution is recorded as 0.513. Similar behavior is observed as in skew convergence. The 2-hop SATS algorithm has faster convergence than 1-hop SATS, ATSP and CCS algorithms. Overall, the 2-hop SATS algorithm converges 16% faster than 1-hop SATS, 33 % faster than CCS and 50% faster than ATSP.

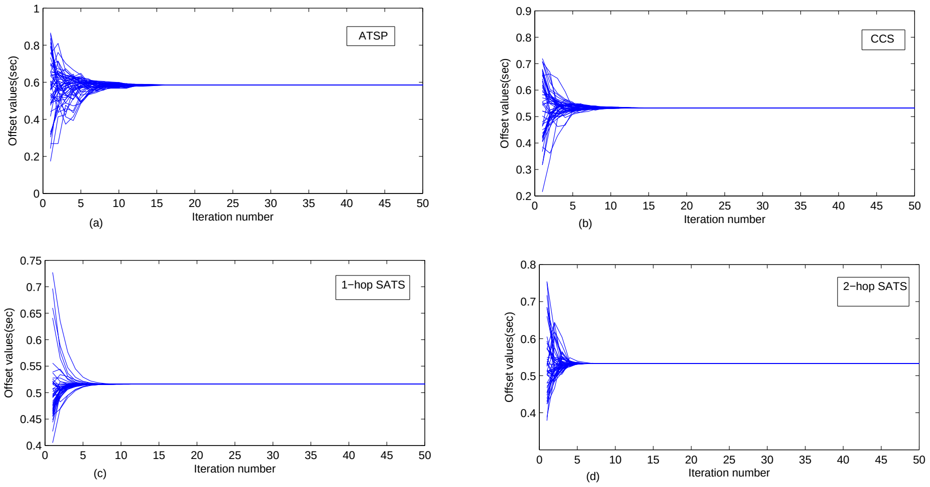


Figure 4.6: Offset convergence of (a) ATSP, (b) CCS, (c) 1-hop SATS, and (d) 2-hop SATS

### (iii) Average global synchronization error

Fig. 4.7 shows average global (network-wide) synchronization error of 50 nodes after each iteration by considering maximum 50 iterations. It is observed that the average global synchronization error for 50 iterations for ATSP is 0.0550 sec., 0.0180 sec. for CCS, 0.0040 sec. for 1-hop SATS, and 0.0023 sec. for 2-hop SATS algorithms. Hence, the 2-hop SATS algorithm has 95% improvement of average global synchronization error over ATSP, 86% over CCS, and 46% improvement over 1-hop SATS algorithm.

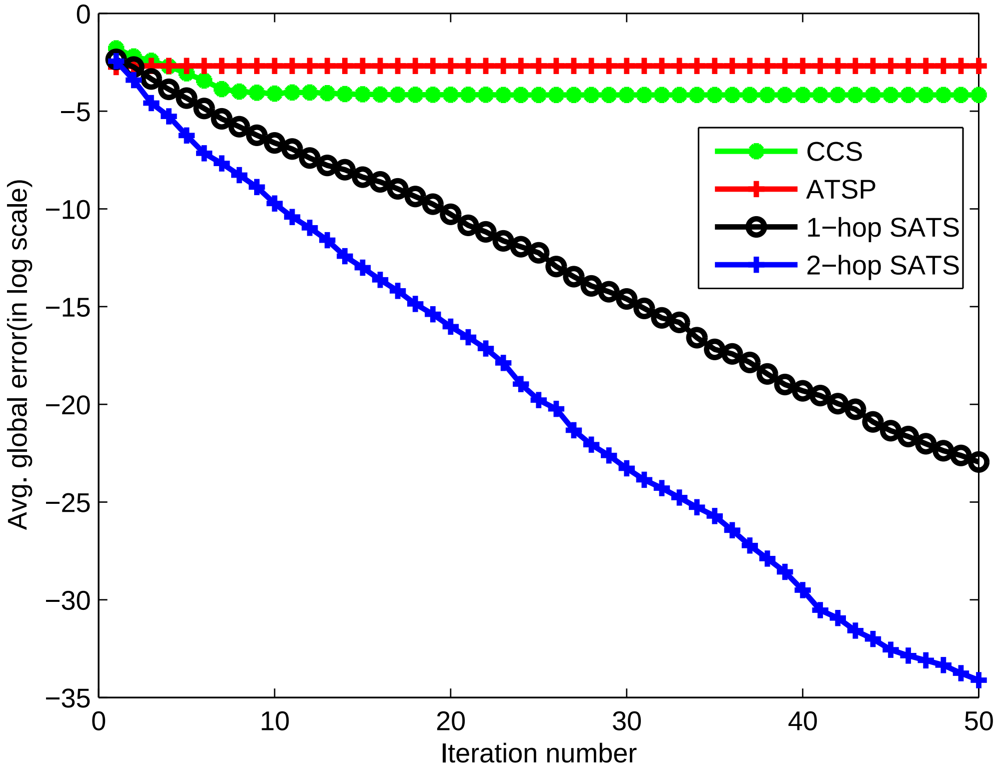


Figure 4.7: Average global synchronization error Vs. Iteration number

**(iv) Average local synchronization error**

Fig. 4.8 (a)-(d) depicts average local synchronization error of each node for a topology of 50 nodes for 50 iterations. This shows the upper bound of local synchronization error of every node. It is observed ATSP has maximum local synchronization error of 0.0610sec., CCS has 0.0523 sec., 1-hop SATS has 0.0124 sec., and 2-hop SATS has 0.0073 seconds. So, the 2-hop SATS algorithm has less local synchronization error, nearly 88%, as compared to ATSP, 86% to CCS, and 41% to 1-hop SATS algorithm. So, local synchronization error is also optimized by the 2-hop SATS algorithm.

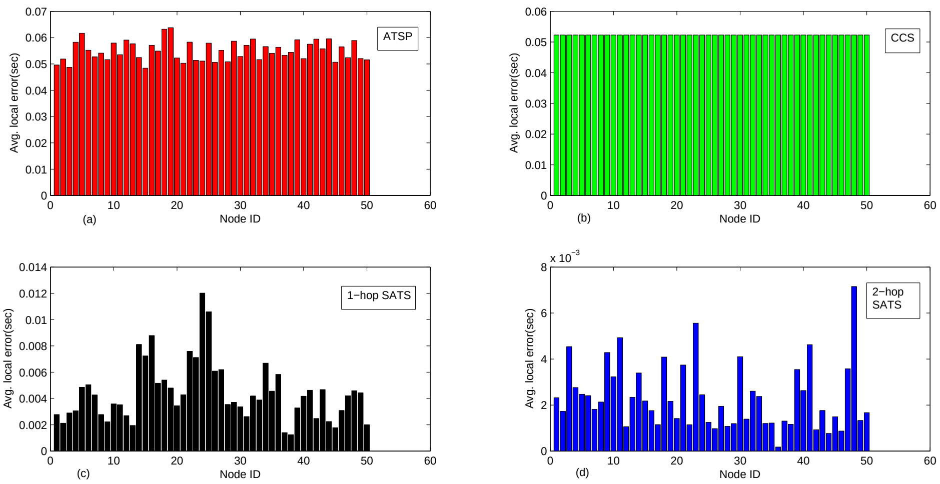


Figure 4.8: Average local synchronization error of (a) ATSP, (b) CCS, (c) 1-hop SATS, and (d) 2-hop SATS

### (v) Number of messages

To achieve synchronization with the given error bound, the number of messages exchanged at each node is recorded, and the average is computed for the whole network with different network size varying from 50-250 nodes. It is observed from Fig. 4.9 that the 2-hop SATS algorithm has almost exchanged 29 % less messages than ATSP but 83% more messages than CCS and 5% more messages than 1-hop SATS. This is because of multi-hop communication in 2-hop SATS. Further, the message overhead in the proposed 2-hop SATS algorithm w. r. to CCS is more because CCS follows one-way message passing paradigm whereas 2-hop SATS follows two-way messaging scheme. So, there exist a trade-off between synchronization accuracy and number of messages exchanged between 2-hop SATS and CCS.

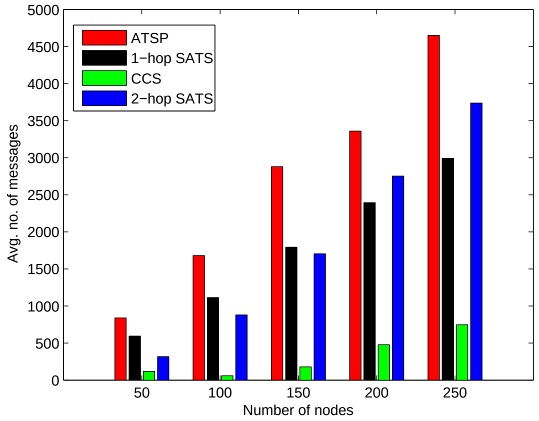


Figure 4.9: Avg. no. of messages Vs. No. of nodes

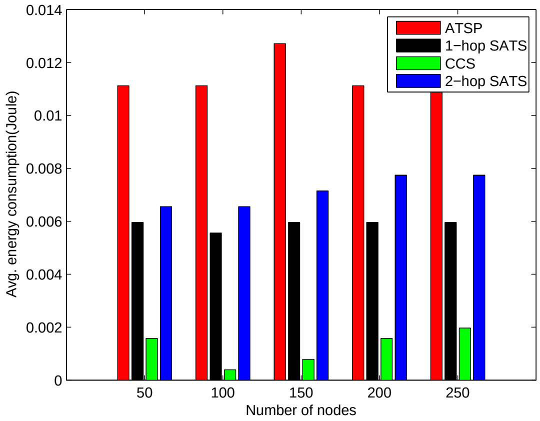


Figure 4.10: Avg. energy consumption Vs. No. of nodes

### (vi) Energy consumption

The average energy consumption is estimated using the mathematical derivations obtained in section 3.4.4. The number of nodes is varied from 50-250. It is observed from Fig. 4.10 that in an average, the 2-hop SATS algorithm has 38 % less energy consumption than ATSP but 82 % and 18 % more energy consumption than CCS and 1-hop SATS algorithm respectively. The extra energy consumption in 2-hop SATS algorithm is also due to multi-hop communication and message overhead.

### (vii) Scalability

The scalable performance of the algorithms is tested according to two aspects. The first scalability test is based on increasing the network size and the second test is based on varying the sparsity factor '$\tau$'.

### (a) Impact of network size

Fig. 4.11 and 4.12 show the behavior of the algorithms with different network size with a constant sparsity factor $\tau$=0.2. We have considered the average number of iterations and average Mean Square Error (MSE) as performance metrics to evaluate the scalability of the algorithms. The observations are made on random, sparse topologies varying nodes from 50-250 nodes. The average is calculated for 100 realizations of such random topologies for each number of nodes. From Fig. 4.12, it is observed that the mean of the average number of iterations is 5.84 for ATSP, 2.84 for CCS, and 2.24 for 1-hop SATS, and 0.8 for 2-hop SATS. The standard deviation for ATSP is 0.49, 0.316 or CCS, 0.26 for 1-hop SATS, and 0.08 for 2-hop SATS.

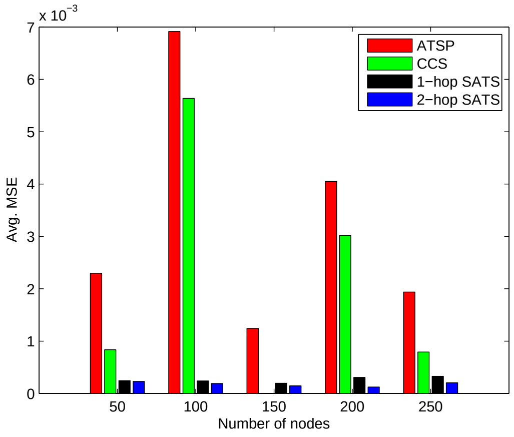


Figure 4.11: Avg. MSE Vs No. of nodes

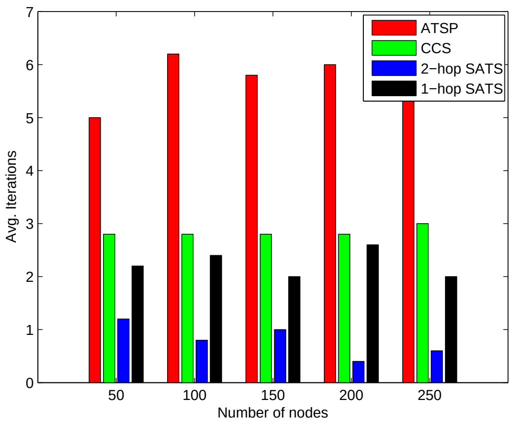


Figure 4.12: Avg. iterations Vs No. of nodes

Similarly, from Fig. 4.11, it is observed that the mean of average MSE is 0.0033 for ATSP, 0.0021 for CCS, 0.000264 for 1-hop SATS, and 0.00018 for 2-hop SATS. The standard deviation for ATSP is 0.0023, 0.0024 for CCS, 0.000054 for 1-hop SATS, and

0.0000433 for 2-hop SATS, it is 0.0005. So, it is inferred that the lower mean value of the proposed 2-hop SATS shows its optimal performance, and lower standard deviation shows its consistency. Hence, the 2-hop SATS algorithm is more scalable than 1-hop SATS, ATSP, and CCS algorithm on sparse network.

### (b) Impact of sparsity factor'$\tau$'

The impact of sparsity factor '$\tau$' has been studied by considering different network size and varying the sparsity factor between 0.2 and 0.5. The considered network size is from 50-250 and for each network size, the sparsity factor is varied in the given range to check its impact on the performance of the algorithms. For each network size, the means and standard deviations of MSE are observed for different values of sparsity factor as shown in Fig. 4.13 and Fig. 4.14.

It is observed that the average mean of MSE for different network size is 0.0027 for ATSP, 0.0015 for CCS, 0.00022 for 1-hop SATS, and 0.000153 for 2-hop SATS. Similarly, the average standard deviation of MSE is 0.0012 for ATSP, 0.0011 for CCS, 0.000086 for 1-hop SATS and 0.0000587 for 2-hop SATS. So, it is inferred that the lower mean value of 2-hop SATS shows its optimal performance, and lower standard deviation shows its consistency. Hence, 2-hop SATS algorithm is more scalable than ATSP, CCS, and 1-hop SATS w. r. to different sparsity factor.

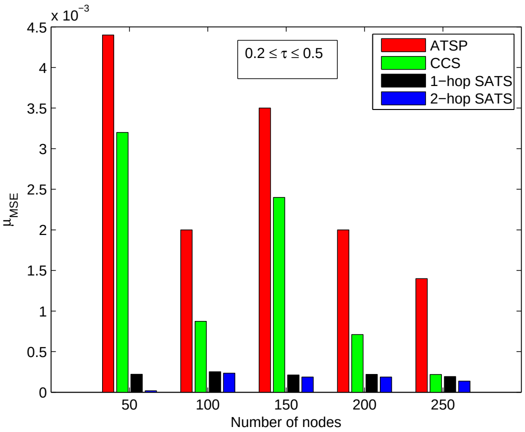


Figure 4.13: Mean of MSE Vs. No. of nodes

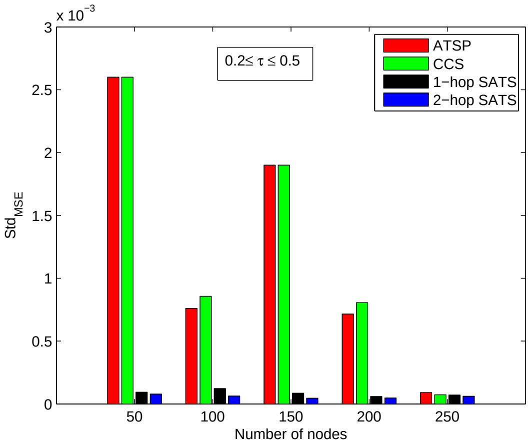


Figure 4.14: Standard deviation of MSE Vs. No. of nodes

### (viii) Impact of number of hop

The impact of increasing the number of hop on the proposed multi-hop SATS algorithm is also studied. By increasing the number of hop from 2 to 3 and 4, the average end-to-end delay is estimated from the time instance of sending the SYN_INIT message to the instance of receiving SYN _ACK message and is compared with the theoretical threshold delay as shown in Fig. 4.15. It is observed that with the increase in the number of hops, the estimated end-to-end delay increases and the theoretical threshold delay decreases. After, hop count 4, the estimated average end-to-end delay supersedes the threshold delay. As a result, the consensus stability is disturbed as shown in Fig. 4.17 (c) for 4-hop SATS. Also, it is observed from Fig. 4.16 that due to consensus instability, the synchronization error also increases for 4-hop SATS algorithm. So, a restricted hop selection between 2 and 3 can improve the performance of the multi-hop SATS algorithm over the one-hop consensus-based synchronization algorithms.

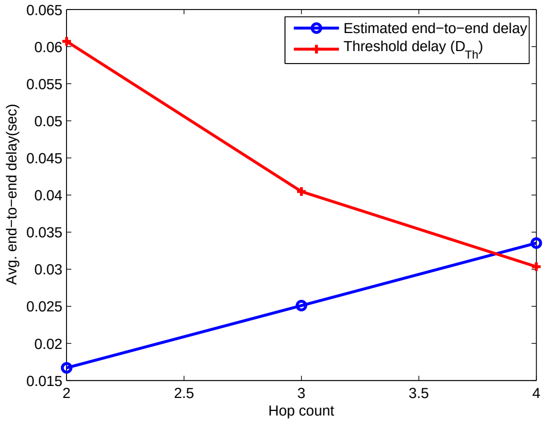


Figure 4.15: End-to-end delay Vs No. of hop

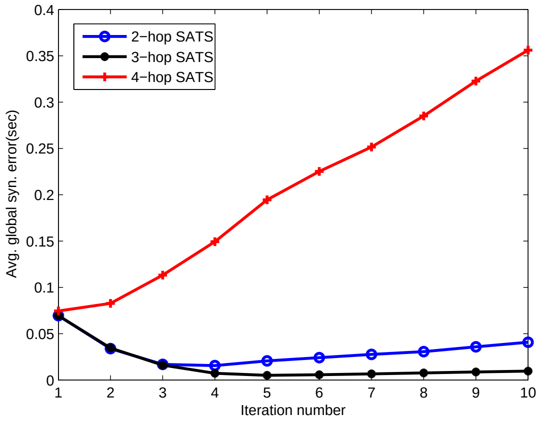


Figure 4.16: Avg. global sync. error Vs Iteration number

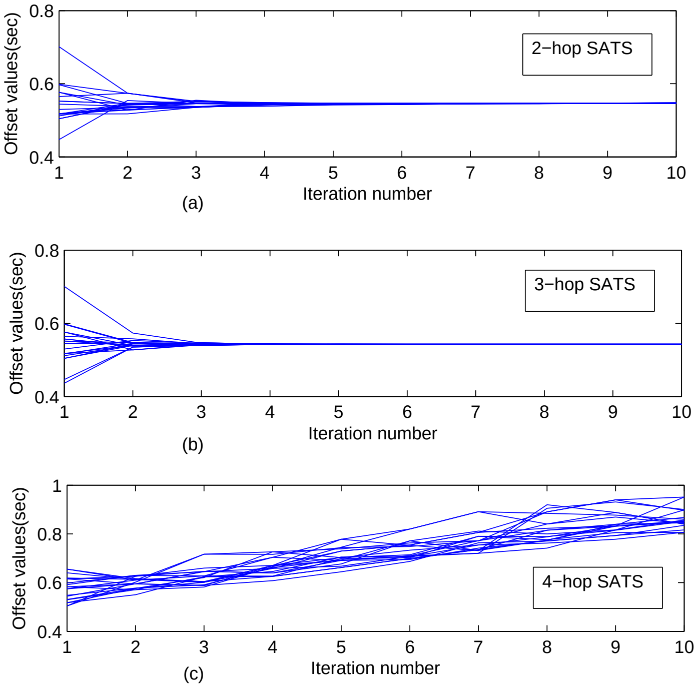


Figure 4.17: Impact of hop count on convergence of multi-hop SATS

## 4.8 Summary

In this Chapter, a consensus-based multi-hop SATS algorithm is proposed which is targeted for the random and sparse network. Since consensus-based algorithms are greatly affected by topological connectivity, we aimed at increasing the topological connectivity by using multi-hop communication for the underlying sparse network. At the same time, increasing hop count incurs higher end-to-end delay which affects the consensus stability. In order to restrict the delay and select a multi-hop node with maximum relative clock values, a distributed, constraint-based dynamic programming approach is suggested. Using multi-hop communication, a node is selected using the proposed multi-hop SATS algorithm and pairwise averaging is performed between the initiating node and the selected node. The asymptotic message complexity of multi-hop SATS algorithm in a network of $n$ nodes and up to $m$ hop is proved to be $O\left(n(\log n)^m\right)$. A thorough energy consumption analysis is also carried out for the proposed multi-hop SATS algorithm.

Simulation results show that on sparse topology, the proposed multi-hop SATS algorithm with hop count 2 has 16 % faster convergence speed than the proposed SATS (1-hop) algorithm in chapter 3. Also, the 2-hop SATS algorithm has 33 % faster convergence speed than CCS and 50 % faster than ATSP on sparse topology. The 2-hop SATS algorithm has 95% improvement of average global synchronization error over ATSP, 86% over CCS, and 46% improvement over 1-hop SATS algorithm. The local synchronization error using 2-hop SATS is also optimized, nearly 88% as compared to ATSP, 86% to CCS, and 41% to 1-hop SATS algorithm. As compared to ATSP, the average number of messages exchanged for 2-hop SATS algorithm is 29 % less but, when compared with CCS and 1-hop SATS, it is respectively, 83 % and 5 % more. The message overhead of 2-hop SATS as compared to CCS is high because CCS follows one-way message passing paradigm for weighted averaging whereas 2-hop SATS follows two-way message passing paradigm to perform pair-wise averaging. So, there exist a trade-off between message exchanges and synchronization accuracy in case of 2-hop SATS and CCS.

The 2-hop SATS algorithm has shown better scalability, both in varying network size and varying sparsity factor scenario. Increasing the hop count from 2 to 3 also improves the convergence speed and synchronization error. But, simulation results show that with the further increase in hop count, the end-to-end delay supersedes the threshold delay. So, the optimal behavior of the algorithm lies in the restricted selection of hop count which also ensures consensus stability of multi-hop SATS algorithm. In fact, restricted hop count also makes the algorithm optimal in terms of message complexity. In the next two Chapters, topological optimization strategies are proposed to create logical communication topologies for consensus-based synchronization algorithms. The basic objective is to minimize message overhead, energy consumption without compromising synchronization precision.

# Chapter 5

# Topological Optimization Strategy for Consensus Time Synchronization Algorithms on dense topology: A Genetic Algorithm based Approach

Recent approaches to Consensus Time Synchronization (CTS) algorithms are ‘‘all node based”, i.e., every node iterates the consensus algorithm to reach to the synchronized state by exchanging synchronization messages with neighbors. This increases the congestion in the network due to extensive synchronization message exchange and induces delay in the network. The delay induced in the packet exchange is the main source of synchronization error and slows down the convergence speed to the synchronized (consensus) state. Also, extensive use of synchronization messages causes more energy consumption. Hence, it is desirable that a ‘‘subset” of sensors along with a balanced number of neighboring sensors should be selected during topology construction such that an optimal logical topology can be established. Embedding this logical communication topology, the performance of CTS algorithms can be improved significantly. In this Chapter, a Connected Dominating Set (CDS) based topological optimization strategy is proposed using Genetic Algorithm (GA) for CTS algorithms. Using this optimized generic communication topology, it is observed that the performance of some state-of-the-art CTS algorithms has been improved significantly.

## 5.1 Introduction

Topological optimization for a specific objective in WSNs is a well-researched area. This problem is also referred as sensor selection problem in some literature [4, 69, 77]. The major objectives include network coverage and connectivity, energy savings, delay minimization, optimal routing, and broadcasting [78–80]. Specific to clock synchronization problem, in [77], the authors have used MCDS and k-CDS (for fault tolerant environment) to solve the sensor selection problem for TPSN [21] and RBS [20] synchronization protocols. In [43], CDS based and set cover based approaches are used for multi-hop PBS protocol [4] to minimize message complexity. A generic strategy is proposed for hierarchy based synchronization protocols in [69] to select multiple clock reference nodes, which is formulated as a k-median problem. The strategies so far proposed for clock synchronization problem are all for hierarchy based protocols.

Topological optimization for CTS algorithms is a least explored area. To the best of our knowledge, our work is the first work to address the topological optimization problem for CTS algorithms. A recent work has been reported by Jie Wu *et al.* in [57] which incorporates LEACH clustering technique to improve convergence speed and energy efficiency of distributed CTS algorithms. But, the basic difference between our work and the work proposed in [57] is: our approach is based on creating an optimal, logical communication topology for CTS algorithms which can be incorporated during topology construction phase whereas the work in [57] is based on clustering as a pre-step for CTS algorithms which is an extra overhead for the synchronization process.

The major contributions of this Chapter are the followings.

1. Proposes a novel GA-based approach for topological optimization problem for CTS algorithms, based on the delay balanced topology concept introduced by [66].
2. Validates the optimal behavior of the proposed strategy through extensive simulation based performance analysis of some recent state-of-the-art CTS algorithms.
3. Compares the performance with all node based CTS algorithms, Minimum Connected Dominating Set (MCDS) strategy, and Load Balanced Connected Dominating Set (LBCDS) strategy which are widely used as generic logical communication strategies.

The rest of the Chapter is organized as follows. Section 5.2 introduces the system models and definitions; Section 5.3 formulates the problem; Section 5.4 presents the detailed GA-based proposal and Section 5.5 gives the simulation results followed by conclusion in Section 5.6.

## 5.2 System Models & Definitions

In this Chapter, the clock model and the network model are the same as described in Chapter 3. The additional models used in this chapter are described below along with the following definitions.

**Definition 1**: Synchronization Initiating (SI) Node

This is the subset of sensor nodes which are selected for initiating the synchronization algorithm. For the WSN *'G'*, let *'I'* denotes the set of SI nodes where $I \subset V$. It is required that the nodes in *'I'* must be connected for achieving network wide synchronization.

**Definition 2**: Synchronization Participating (SP) Node

This is the subset of one hop neighbor sensor nodes which are allocated to the SI nodes. These nodes involve in the synchronization process only upon receipt of messages from the SI nodes. It is given by *P=V-I*. Each SP node is allocated to only one SI node to avoid synchronization update inconsistency, congestion, and packet losses. It is important to note here that the SI nodes also behave as SP nodes when they receive messages from the neighboring SI nodes. This helps in propagating the consensus throughout the network.

### 5.2.1 Generic CTS Framework

In each iteration of the CTS algorithm, every node initiates the synchronization process by sending an initiation message. After receiving the time-stamped reply messages from its neighbors, it estimates the arrival time of its neighbors' messages. Each node then updates its local clock time using pairwise averaging method [22] or weighted averaging methods [17, 18] until all nodes converge to the average of the initial clock differences between the nodes with some tolerable synchronization error. In the presence of both random and deterministic delays during message exchanges, the clock update rule at each node *'i'* is given as [66]:

$$C_i(t_{k+1}) = C_i(t_k) + \epsilon \sum_{j \in N_i} \left| C'_j(t_k) - C_i(t_k) \right| \tag{5.1}$$

where $C_i(t_k)$ is the local time at node '$i$' during iteration '$k$' and '$\epsilon$' is the constant step size for each iteration. $C'_j(t_k)$=$C_j(t_k) + T_{delay}$. The total delay $T_{delay}$, ignoring system level delay factors, is given as:

$$T_{delay} = T^{PHY}_{delay} + T^{MAC}_{delay} \tag{5.2}$$

where $T^{PHY}_{delay}$ is the physical layer delay and $T^{MAC}_{delay}$ is the MAC layer delay.

According to [28], if the topology is balanced, then consensus can be achieved even if in the presence of delay. Similar analysis exist in [66] on the basis of CTS protocols and the authors have introduced the concept of ''delay balanced network" which is defined as:

**Definition 3**: A network is said to be delay balanced if $\sum_{j \in N_i}(T_c + l_{ij}/c) = \sum_{m \in N_k}(T_c + l_{km}/c) = ... = \sum_{q \in N_p}(T_c + l_{pq}/c)$ for *(i, j), (k, m),..., (p, q)* $\in$ *E*. *'l'* represents the distance between the neighbouring nodes, *'c'* is the speed of light and $T_c$ is a constant.

In [66], Definition 3 is used for all node based approach. But, our objective is to select a subset of nodes to minimize message complexity and energy consumption along with delay balancing to accelerate consensus. So, the following definition, based on CDS, is introduced.

**Definition 4**: CDS based delay balanced network

A network is said to be CDS based delay balanced if $\sum_{j \in N_i}(T_c + l_{ij}/c) = \sum_{m \in N_k}(T_c + l_{km}/c) = ... = \sum_{q \in N_p}(T_c + l_{pq}/c)$ for *(i, j), (k, m),..., (p, q)*$\in$ *E'*. *i, k, .., p* are the nodes in CDS and *E'* represents connectivity among dominators.

### 5.2.2 Consensus Energy Model

To compute the energy consumption for the CTS protocols, the power model-1, proposed in [81], is closely followed. For the sake of simplicity, the model only considers energy consumption in message transmission $P_{tx}$ and reception $P_{rx}$, defined as follows:

$$P_{tx} = M(\beta_1 + \beta_2 l(i,j)^{\zeta}) \; and \; P_{rx} = M\gamma \tag{5.3}$$

where '$\zeta$' is the path loss exponent, typically within the range between 2 and 6. The constants $\beta_1$, $\beta_2$ and $\gamma$ are the energy dissipated by the transmitter module, transmit amplifier, and the receiver module respectively. The estimated distance between nodes '$i$' and '$j$' is denoted as $l(i,j)$ and the length of message as '$M$'. Using CTS framework, a node transmits and receives to and from each of its neighbors at every iteration. Assuming local broadcasting, the energy consumed by a node '$i$' after '$t$' iteration is given by Equation 5.4.

$$P(i) = tM(\beta_1 + \beta_2 max\{l(i,j), j \in N_i\}^{\zeta} + \gamma|N_i|) \tag{5.4}$$

Thus, the average nodal energy consumption for a network of '$n$' nodes is given by Equation 5.5:

$$P_{avg} = \frac{1}{n}\sum_{i=1}^{n} P(i) \tag{5.5}$$

## 5.3 Problem Formulation

Based on the above-discussed models and definitions, the following section first analyzes the relationship between topological parameters and delay and then formulates the objective.

### 5.3.1 Problem Analysis

The major factor that affects the synchronization accuracy and convergence speed is the delay incurred at different layers of communication to send the synchronization packets. Since the system level and communication delay estimation is not feasible during topology construction phase [82], some topological parameters need to be identified to model the delay cost. From literature [66, 80, 82], it is observed that the physical layer delay is proportional to the distance between the nodes in a wireless medium, i.e., $T^{PHY}_{delay} \propto l_{ij}$ where $l_{ij}$ is the Euclidean distance between sender and receiver. On the other hand, as multiple SP nodes are associated with a single SI node, there exists a channel competition among SP nodes at the MAC level to send the synchronization reply messages. Assuming each SP node has an equal probability of accessing the channel, the MAC delay, as a result of SP nodes competition, is directly influenced by the number of SP nodes associated with a SI node, i.e., $T^{MAC}_{delay} \propto d_i$ [82] where $d_i$ is the degree of connectivity of the node i. For CTS protocols which rely on one hop communication, the $T_{delay}^{PHY}$ can be neglected [18]. Hence, the $T_{delay}^{MAC}$ mostly affects the synchronization accuracy.

As a matter of fact, there are various other parameters, both deterministic and random, e.g., network traffic, protocol processing time, transmission collision, that count for delay. In this Chapter, since, our focus is on selection of SI nodes and assignment of SP nodes prior to the synchronization process, i.e., during the topology construction phase of the network, it is quite infeasible to measure all the delay components at this phase [82]. Hence, we have considered ‘‘degree of connectivity” of SI nodes as discussed above as an equivalent metric for delay to design our objective function which is given below.

### 5.3.2 Problem Objective

Based on the definition 4, the problem is named as CDS based Delay Balanced Topology problem (CDSDBT). Our objective is to select a subset of sensors known as SI nodes where the SI nodes form a CDS and to assign the remaining nodes known as SP nodes such that each SP node is assigned to exactly one SI node and the delay is balanced at each SI node with the overall delay being minimized. The problem is formally described below. Let,

(i) the set of total sensor nodes is denoted by $V$ where $|V|=n$ .

(ii) the set of SI nodes is denoted by $I=\{i_1, i_2,..., i_m\}$, $m < n$.

(iii) the set of SP nodes is denoted by $P=\{p_1, p_2,..., p_{n-m}\}$ and $I \cup P=V$.

(iv) the set of SI nodes to which a SP node $p_k$ can be assigned is denoted by $G_k$

(v) the Boolean variable $b_{kj}$=1, if the SP node $p_k$ is assigned to the SI node $i_j$ and $b_{kj}$=0, otherwise.

(vi) the degree of connectivity at a SI node *‘i’* is given by $d_i$.

(vii) the average degree of connectivity of SI nodes is given by $\mu=\frac{1}{m}\sum_{i=1}^{m} d_i$.

The objective is to:

$$Minimize \sum_{i=1}^{m} |d_i - \mu| \tag{5.6}$$

$$subject\ to \sum_{i_j \in G_k} b_{kj} = 1, \forall p_k \in P \tag{5.7}$$

The constraint in Equation 5.7 signifies the assignment of a SP node to exactly one SI node.

For structured networks, e.g., ring, hypercube, etc., CDSDBT problem is analytically tractable. For example, a 2D hypercube with its CDS based balanced topology is given in Fig. 5.1, assuming the distance between neighboring nodes is same. The SI nodes are {1, 2, 3, 4} and the SP nodes are {5, 6, 7, 8}. Each SI node has a balanced degree of 3. But, for WSN where nodes are deployed in large scale and random, employing brute force method to reassign SP nodes to possible SI nodes for topological optimization has high computational complexity. In fact, CDSDBT problem can be reduced to an NP-complete problem which is proved in the following subsection.

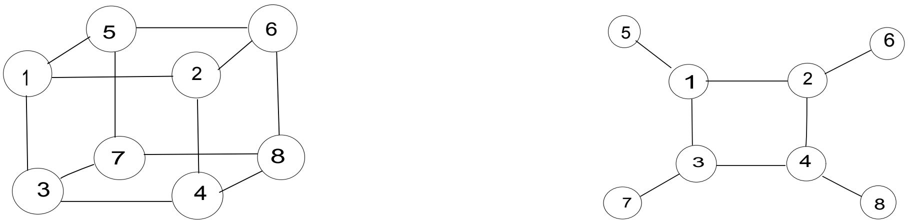


Figure 5.1: 2D hypercube and its CDS based delay balanced topology

### 5.3.3 Intractability of CDSDBT problem

The CDSDBT problem is a generalization of Load Balanced Connected Dominating Set (LBCDS) problem which is described below.

**The LBCDS Problem** [83]: For a WSN represented by a graph *G=(V, E)*, the LBCDS problem is to find out a node set $S \subset V$ , $S$=$\{s_1, s_2, ..., s_m\}$, such that:

(i) *G [S]=(S, E')*, where *E'*=$\{e|e$*=(u, v), u*$\in$ *S, v*$\in$ *S, (u, v)*$\in$ *E*$\}$, is connected.

(ii) $\forall$ *u* $\in$ *V and u* $\notin$ *S*, $\exists$ *v* $\in$ *S, such that (u, v)*$\in$ *E*

(iii) *min* $|S|_2 = (\sum_{i=1}^{m} |d_i - \mu|^2)^{\frac{1}{2}}$

In the following Lemma, it is shown that LBCDS is reducible to CDSDBT problem under the limiting condition.

**Lemma 5.3.1.** *CDS based Delay Balanced Topology (CDSDBT) problem is NP-complete.*

*Proof.* The proof is based on a limiting condition assumption using Definition 4. For a random graph with radius of connectivity *'r'* , the Equation 5.8 must hold for the graph to be connected [72].

$$l_{ij} \leq r, \forall (i,j) \in E \tag{5.8}$$

Lets, under limiting condition, $l_{ij} = r, \forall (i,j) \in E$. Now, the sufficient condition in definition 4 can be splitted into *'m'* terms as given in Equation 5.9.

$$t_1 = \sum_{j \in N_i} (T_c + r/c), t_2 = \sum_{s \in N_k} (T_c + r/c), ...t_m = \sum_{q \in N_p} (T_c + r/c) \tag{5.9}$$

where $N_i$, $N_k$, ..., $N_p$ represents the set of SP nodes assigned to SI nodes. The terms $t_1$, $t_2$,..., $t_m$ must be same , for the definition 4 to be true, if Equation 5.10 is satisfied.

$$|N_i| = |N_k| = ... = |N_p| \tag{5.10}$$

Equation 5.10 states that all SI nodes, assumed to be a CDS, have equal degree, i.e., $d_1 = d_2 = ... = d_m$. Hence, $\mu = \frac{1}{m} \sum_1^m d_i = d_i$. $|S|_2 = 0$. So, condition (iii) in definition of LBCDS problem is satisfied. This is verifiable in polynomial time of O($d_{avg} m^2$) where $d_{avg}$ is the average degree of the network and 'm' is the cardinality of the CDS (set of SI nodes). So, under the assumption of above limiting condition, CDSDBT problem is a generalization of LBCDS problem. Since, LBCDS is an NP-complete problem [83], by the principle of reducibility [74], CDSDBT problem is also NP-complete. □

So, to deal with the intractability nature of the problem, a GA based approach is proposed to get a near optimal solution for the problem. To solve the problem, a set of nodes needs to be initially selected as SI nodes, which forms a CDS and serves a virtual backbone. In fact, a majority of works have been proposed in the literature to construct CDS whose basic objective is to minimize the cardinality of CDS [65]. Minimizing the cardinality of CDS may not ensure balanced allocation of dominatees to dominators. As a result, the delay balancing criteria may not be satisfied which is our primary requirement. Also, if the CDS are not balanced, some heavily loaded dominators will deplete their energy quickly, resulting in a disconnected network. Recently, a CDS heuristic is proposed by Jing *et al.* [83] to balance CDS. But, the heuristic does not explicitly allocate dominatees to dominators in a balanced way.

Hence, our proposal proceeds by cascading the following two steps:

(i) Selecting a set of SI nodes (dominators) using the CDS heuristic [83] described in algorithm 5.

(ii) Allocating the SP nodes (dominatees) to SI nodes (dominators) using the proposed GA based strategy to get the CDS based delay balanced logical topology.

**Algorithm 5** CDS Heuristic [83]

```
1:  /***‘n’ is total number of nodes in the network and ‘$deg_i$’ is the degree of node $i$***/
2:  Set DS=$\phi$
3:  Compute $\overline{deg} = \frac{1}{n}\sum_{i=1}^{n} deg_i$
4:  for i=1 to n do
5:      Compute $var_i = \left|deg_i - \overline{deg}\right|$
6:  end for
7:  Select Node $i$ such that ‘$var_i$’ is minimum
8:  DS=DS $\cup$ $i$
9:  if DS dominates all other nodes then
10:     if Connected(DS)=TRUE then
11:         The required CDS is: DS
12:         Exit
13:     else
14:         Goto Step 7
15:     end if
16: end if
```

## 5.4 Proposed GA based Strategy for CDSDBT problem

After selecting the set of SI nodes using Algorithm 5, the proposed GA proceeds as follows.

### 5.4.1 Chromosome encoding

Each chromosome is encoded as a structured string of length $(n - m)$ where $(n - m)$ is the number of SP nodes. Each gene represents two values, one is the SP node ID and the other is the assigned SI node ID. The nodes are assigned with ids as 1, 2,..., n. The following example illustrates the chromosome representation.

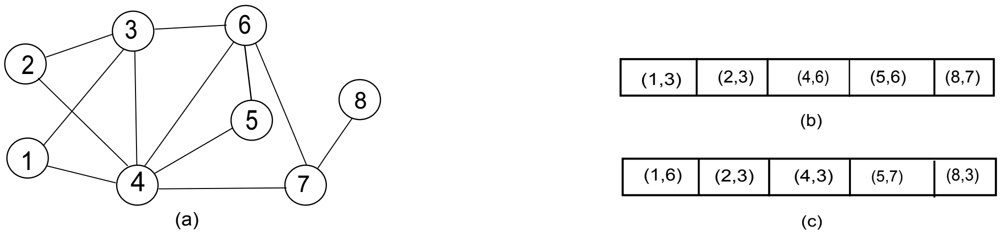


Figure 5.2: (a) WSN topology of 8 nodes, (b) Valid Chromosome, and (c) Invalid Chromosome

**Example**: Consider a WSN of 8 nodes as shown in Fig. 5.2 (a) where the set of SI nodes, obtained by the CDS heuristic given in algorithm 5, is $I$={3, 6, 7}. So, the set of SP nodes is given by $P$={1, 2, 4, 5, 8}. Thus, the length of the chromosome is 5. Fig. 5.1(b) shows a valid chromosome representation. The value at gene position 1 is (1, 3) which means SP node 1 is assigned to SI node 3. Similar interpretations can be drawn for other positions.

### 5.4.2 Initial Population Generation

The initial population in GA is generally generated randomly. But, in our problem, total randomness may generate invalid chromosomes which will make the selection process slower. So, each SP node is assigned to one of the randomly selected neighboring SI nodes instead of assigning it to any randomly selected SI node. The idea is illustrated below to differentiate between valid and invalid chromosome. In Fig. 5.1(c), the gene value at position 4 is (5, 7) which means SP node 5 is assigned to SI node 7. Though node 7 is a SI node, it is not a neighboring SI node of SP node 5 as given in Table 5.1. So, this creates invalid chromosome for our problem.

All the chromosomes generated in this process represent valid assignments of SP node to SI nodes but may not be optimal. To find an optimal chromosome, its fitness is evaluated as follows.

### 5.4.3 Fitness Evaluation

The fitness value of each chromosome is evaluated using Equation 5.6. The '$d_i$' value in Equation 5.6 is calculated using the following Equation.

$$d_i = f_i + k_i \tag{5.11}$$

where $d_i$=degree of SI node *'i'*, $f_i$=frequency of SI node *'i'* in the chromosome, $k_i$=number of neighbor SI nodes of SI node *'i', i=1, 2,..., m*. The '$f_i$' value is obtained by counting the number of appearance of a SI node id in the second gene value and the '$k_i$'

Table 5.1: SP nodes with possible neighbor SI nodes

| $SP\ node\ id$ | $Neighbour\ SI\ nodes\ id$ |
|---|---|
| 1 | $\{3\}$ |
| 2 | $\{3\}$ |
| 4 | $\{3,6,7\}$ |
| 5 | $\{6\}$ |
| 8 | $\{7\}$ |

value is obtained from the dominators (SI nodes) adjacency matrix. This is illustrated in the following example.

**Example**: Consider, a valid chromosome as shown in Fig. 5.2 (b). For each SI node, the $d_i$ value can be calculated as follows:

$d_3$=3+ 1=4, $d_6$=1+ 2=3, $d_7$=1+ 1=2. The number of neighboring SI nodes of a particular SI node $k_i$ is obtained from the SI node adjacency matrix E' as shown below. First row corresponds to SI node 3, second row to SI node 6 and third row to SI node 7 and similar for columns.

$$E' = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The lower value of Equation 5.6 gives better fitness value and the chromosomes with better fitness values are selected using the following step.

### 5.4.4 Selection

The selection step determines which chromosomes from the current generation will mate to create new chromosomes. For this step, we have used tournament selection with tournament size 2. It selects better of two randomly selected chromosomes with the probability given by the tournament selection parameter $t_{sp}$ (in our case, it is 0.75). With probability (1-$t_{sp}$), the worse of the two chromosomes is selected. The selected chromosomes will mate by the crossover method described below to produce new off-springs.

### 5.4.5 Crossover

The crossover takes place between the chromosomes selected in the above step with certain crossover probability $cr_p$ (in our case, it is 0.8). We have used 1-point crossover where a point is chosen at random and the two selected chromosomes swap their gene values after that point.

### 5.4.6 Mutation

The mutation operator is applied at a selected gene position instead of random gene position. This is the heart of the delay balanced sensor selection strategy. The SI node ID with maximum $d_i$ value is selected from the chromosome for mutation as shown in Fig. 5.3 (a). It is replaced with the SI node ID with minimum $d_i$ value which belongs to the neighbor SI ID of the corresponding SP node ID given in Table 5.1. This is illustrated in the following example.

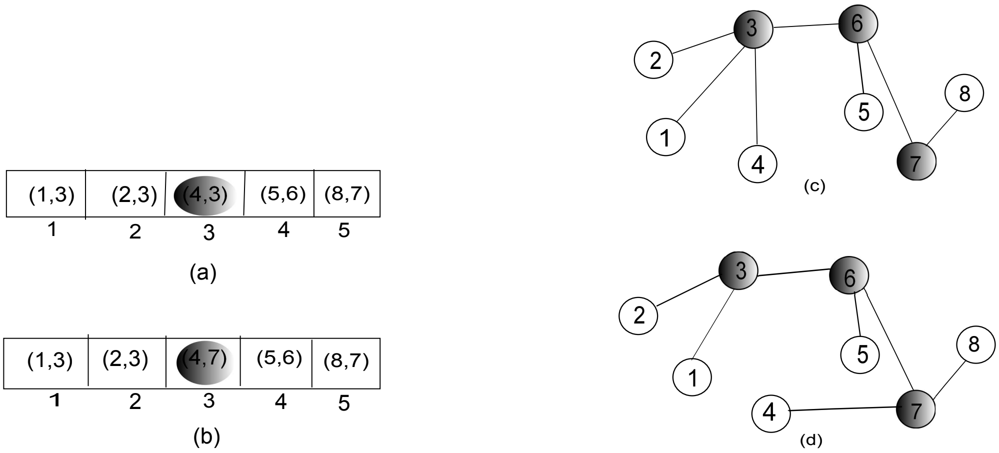


Figure 5.3: (a) Chromosome before mutation, (b) Chromosome after mutation, (c) topology corresponds to chromosome (a), and (d) topology corresponds to chromosome (b)

Since, $d_3$ has the maximum value, a gene position, with its second value as 3, has to be chosen for mutation. The selection is made on position 3 (satisfying criteria for a valid chromosome) and mutated with value 7 because $d_7$ is minimum.The fitness values using equation 5.6 is 3.6 for unmutated chromosome and 1.8 for the mutated chromosome as shown in Fig. 5.3 (b). The corresponding topologies are shown in Fig. 5.3 (c) and (d).

Thus, the proposed strategy produces an optimal chromosome after a number of generations which represents an optimal balanced, logical topology that will satisfy definition 4. Embedding this logical topology as the virtual backbone for the CTS algorithms can balance and minimize the overall delay with optimal consensus convergence, message complexity and energy efficiency as given below in simulation results.

## 5.5 Simulation Results & Discussion

The synchronizing nodes are selected offline using the proposed GA-based strategy. The obtained optimal communication topology is tuned with the PROWLER simulator [30] to study the performance of the recent state-of-the-art CTS algorithms. The simulation parameters considered for the evaluation of synchronization algorithms are mentioned in Table 5.2. For comparative analysis, the traditional MCDS and the recent LBCDS topological strategies are also considered. Since, both MCDS and LBCDS are proved to be NP-complete, some recent GA based approaches [84, 85] to these problems are considered to have a fair comparison with our GA based proposal.

Table 5.2: Simulation Parameters

| $Parameter$ | $Values$ |
|---|---|
| $Deployment\ area$ | $10 \times 10\ square\ unit$ |
| $Topology$ | $Random$ |
| $No.\ of\ nodes$ (n) | $100 - 500$ |
| $Connectivity\ radius$ (r) | 2 $unit$ |
| $Initial\ skew\ (\alpha)$ | $uniform(-5, 5)$ |
| $Initial\ offset\ (\beta)$ | $uniform(0, 1)$ |
| $Iteration\ interval$ | 10 $sec$ |
| $Acceptable\ Syn.\ error$ | $0.0001 sec.$ |
| $MAC\ Protocol$ | $CSMA/CA$ |
| $Communication\ Standard$ | $IEEE$ 802.11 |
| $Path\ Loss\ Exponent\ (\zeta)$ | 2 |
| $\beta_1$ | 45 $nJ/bit$ |
| $\beta_2$ | 10 $pJ/bit$ |
| $\gamma$ | 35 $nJ/bit$ |
| $Message\ size$ (M) | 320 $bits$ |

### 5.5.1 Evaluation of the proposed GACDBT strategy

The offline evaluation of the proposed strategy is done considering topologies of 100, 200 and 300 nodes as shown in Fig. 5.4. The normalized average fitness is plotted for each generation which is minimized as the generation progresses. Maximum generation considered is 50. The number of chromosomes generated is 50 with tournament selection probability as 0.75 and crossover probability as 0.8. After 50 generations (stopping criteria for our proposed strategy), the chromosome with best fitness value is selected which gives the optimal, balanced topology.

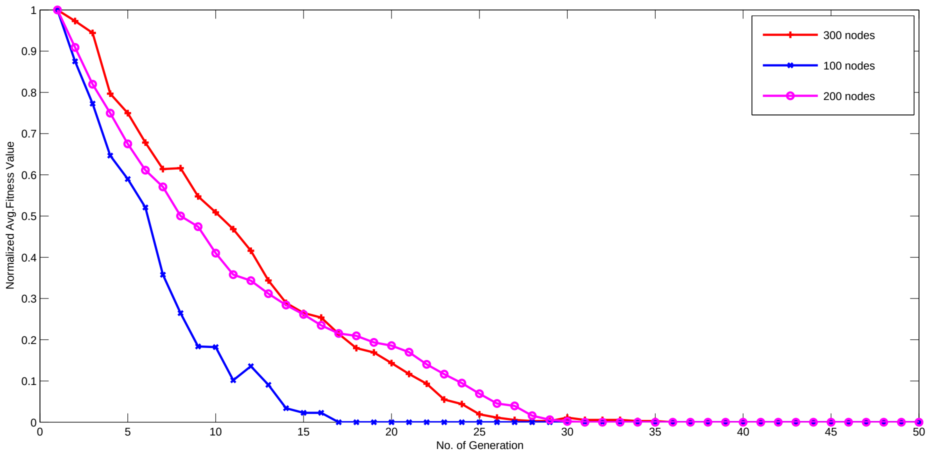


Figure 5.4: Convergence of Proposed GA with topology of 100, 200 and 300 nodes

#### Online Delay Estimation

Fig.5.5(a)-(d) shows the delay distribution on different topological strategies by estimating the delay in sending and receiving the control packets (RTS/CTS packets) at each node to its one-hop neighbors, using the MAC layer time-stamping mechanism for CSMA/CA and following IEEE 802.11 standard [86]. The simulation time taken is 100 sec. and iterated for 50 times. The average delay thus computed at each node, for a random topology of 100 nodes, is plotted as shown in Fig. 5.5.

It is observed that the proposed strategy has a mean delay of 1.046 sec whereas the mean delay for all node, MCDS and LBCDS strategies are 5.828 sec., 2.713 sec. and 1.713 sec. respectively. So, the proposed strategy minimizes the average delay almost up to 80 %, 50% and 30% as compared to the all-node, MCDS and LBCDS strategies respectively. Further, the observed delay variance for the proposed strategy is 2.077 sec. whereas it is 3.583 sec. for LBCDS, 7.514 sec.for MCDS, and 9.660 sec. for all node which is minimum among other strategies' delay variance. So, the delay is comparatively more balanced than other strategies. Table 5.3 shows the scalable performance of the proposed strategy w. r. to delay and comparative optimal behavior among other strategies.

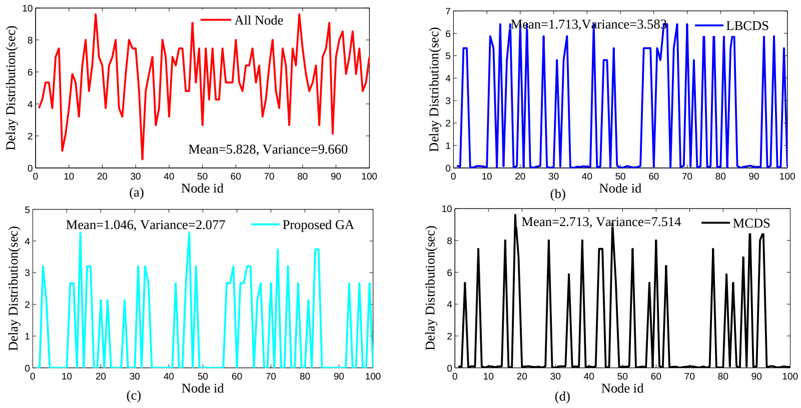


Figure 5.5: Delay Distribution at Individual Node using Different Topological Strategies

Table 5.3: Scalability of Delay Analysis on Different Topological Strategies

| No. of nodes / Topological Strategy | 100 | | 200 | | 300 | | 400 | | 500 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ |
| All node | 0.358 | 0.034 | 9.673 | 20.627 | 12.616 | 21.345 | 7.314 | 15.907 | 10.164 | 27.925 |
| LBCDS | 0.150 | 0.018 | 5.321 | 7.896 | 7.495 | 9.552 | 5.573 | 7.396 | 7.667 | 10.163 |
| MCDS | 0.213 | 0.028 | 8.828 | 14.013 | 8.067 | 12.616 | 6.212 | 9.010 | 8.829 | 16.286 |
| Proposed GACDBT | **0.121** | **0.015** | **2.647** | **4.287** | **2.435** | **7.445** | **2.264** | **5.242** | **2.880** | **7.844** |

### 5.5.2 Independent evaluation of CTS algorithms on proposed GACDBT Strategy

After tuning the optimized topology with the simulator, the proposed SATS algorithm in Chapter 3 along with two recent consensus based synchronization algorithms, namely CCS [17] and ATSP [22], are tested using the performance metrics: (i) convergence speed, (ii)average global and local synchronization error, (iii) Average number of messages and energy consumption.

### (a) Test algorithm-1: ATSP

The ATSP algorithm which uses random pairwise averaging is tested on the proposed GACDBT on a network of 50 nodes and the results obtained are discussed below.

#### (i) Convergence speed

The convergence speed of ATSP in terms of offset convergence is tested for an acceptable synchronization error ($\epsilon$) of 0.0001 sec. as mentioned in Table 5.2. From Fig. 5.6 (a)-(d), it is observed that all node ATSP takes around 28 iterations, GACDBT based ATSP takes around 18 iterations, MCDS based ATSP takes around 22 iterations, and LBCDS based ATSP takes around 20 iterations to converge to the consensus value 0.4132. So, the convergence speed of ATSP on GACDBT has improved by 35 % over all node ATSP, 18 % over MCDS based ATSP and 10 % over LBCDS based ATSP.

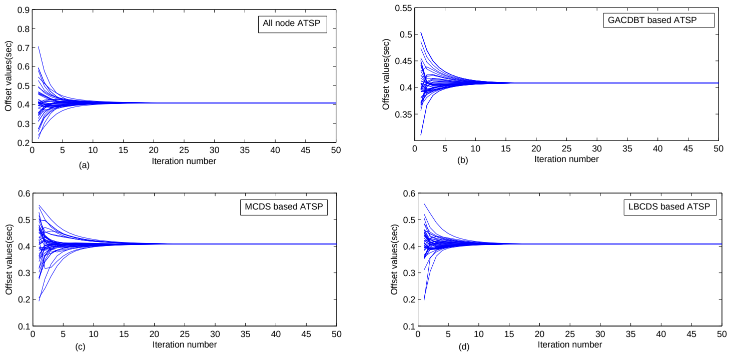


Figure 5.6: Offset convergence of ATSP on different topological strategies

#### (ii) Average global synchronization error

The average global synchronization error per iteration is shown in Fig. 5.7 for ATSP on different topological strategies. It is observed that ATSP on GACDBT has, in an average, 60 % less global synchronization error than all node ATSP, 28 % less than MCDS based ATSP and 22 % less synchronization error than LBCDS based ATSP.

#### (iii) Average local synchronization error

The local synchronization error is averaged for 50 iterations at individual node as shown in Fig. 5.8. The percentage improvement is calculated by considering the maximum local synchronization error among 50 nodes. It is observed that ATSP on GACDBT has 85 % less local synchronization error than all node ATSP, 98 % less than MCDS based ATSP, and 40 % less local synchronization error than LBCDS based ATSP.

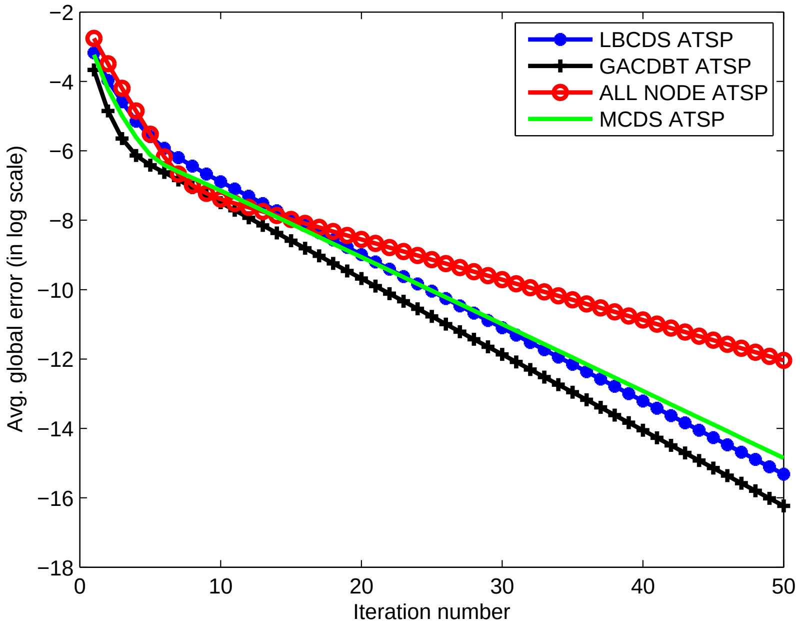


Figure 5.7: Average global synchronization error of ATSP on different topological strategies

### (iv) Average number of messages & energy consumption

The average number of messages exchanged and average energy consumption to achieve the acceptable synchronization error is estimated as given in Table 5.4. For computing the average energy consumption, the model discussed in Section 5.2 is followed. It is observed that ATSP on GACDBT has 70 % less messages exchanged than all node ATSP, 30 % less than LBCDS based ATSP and 15 % less messages exchanged than MCDS based ATSP.

From Table 5.4, it is also observed that ATSP on GACDBT has 73 % less energy consumption than all node ATSP, 19 % less than LBCDS based ATSP and 12 % less energy consumption than MCDS based ATSP.

Table 5.4: Average number of messages & energy consumption of ATSP on different topological strategies for 50 nodes

| **Parameters** | **All node ATSP** | **LBCDS based ATSP** | **MCDS based ATSP** | **GACDBT based ATSP** |
|---|---|---|---|---|
| Avg. number of messages | 1223 | 512 | 423 | 357 |
| Avg. energy consumption (in Joule) | 0.0019 | 0.00062 | 0.00057 | 0.0005 |

### (b) Test algorithm-2: CCS

The CCS algorithm which uses cumulative weighted averaging (CWA) is tested on the proposed GACDBT on a network of 50 nodes and the results obtained are discussed below.

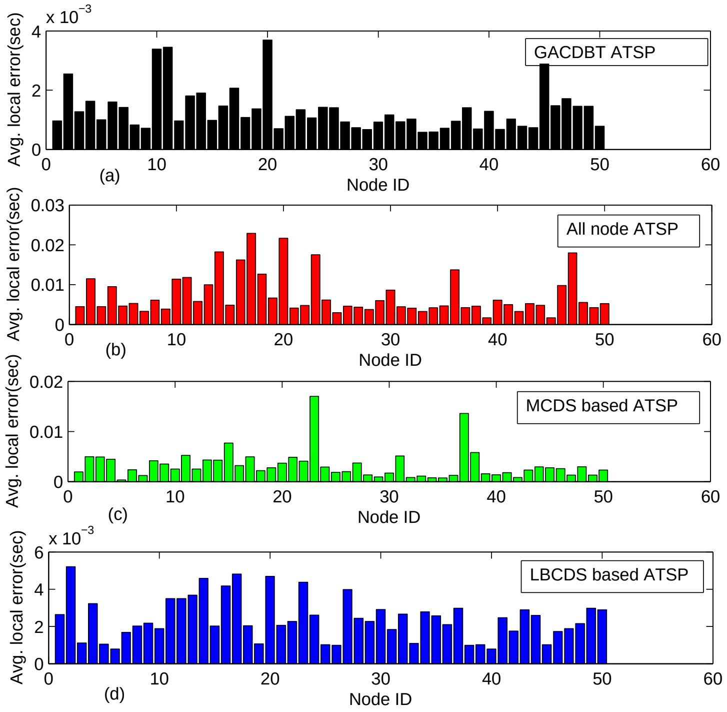


Figure 5.8: Average local synchronization error of ATSP on different topological strategies

### (i) Convergence speed

The convergence speed of CCS in terms of offset convergence is tested for an acceptable synchronization error ($\epsilon$) of 0.0001 sec. as mentioned in Table 5.2. From Fig. 5.9 (a)-(d), it is observed that all node CCS takes around 17 iterations, GACDBT based CCS takes around 8 iterations, MCDS based CCS takes around 20 iterations and LBCDS based CCS takes around 12 iterations to converge to the weighted consensus value 0.6013. So, the convergence speed of CCS on GACDBT has improved by 52 % over all node CCS, 60 % over MCDS based CCS and 33 % over LBCDS based CCS.

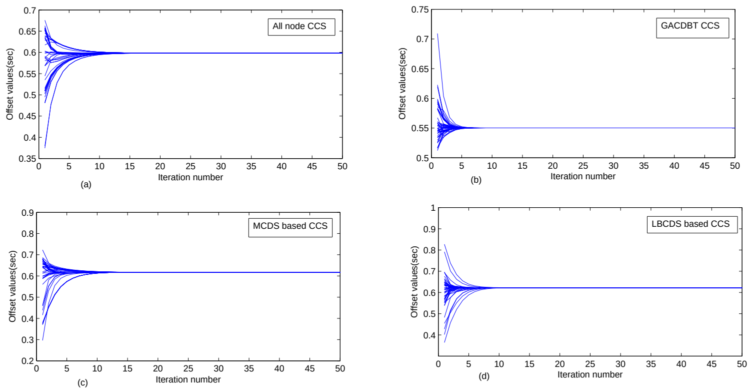


Figure 5.9: Offset convergence of CCS on different topological strategies

### (ii) Average global synchronization error

The average global synchronization error per iteration is shown in Fig. 5.10 for CCS on different topological strategies. It is observed that CCS on GACDBT has, in an average, 84 % less global synchronization error than all node CCS, 78 % less than MCDS based CCS and 53 % less synchronization error than LBCDS based CCS.

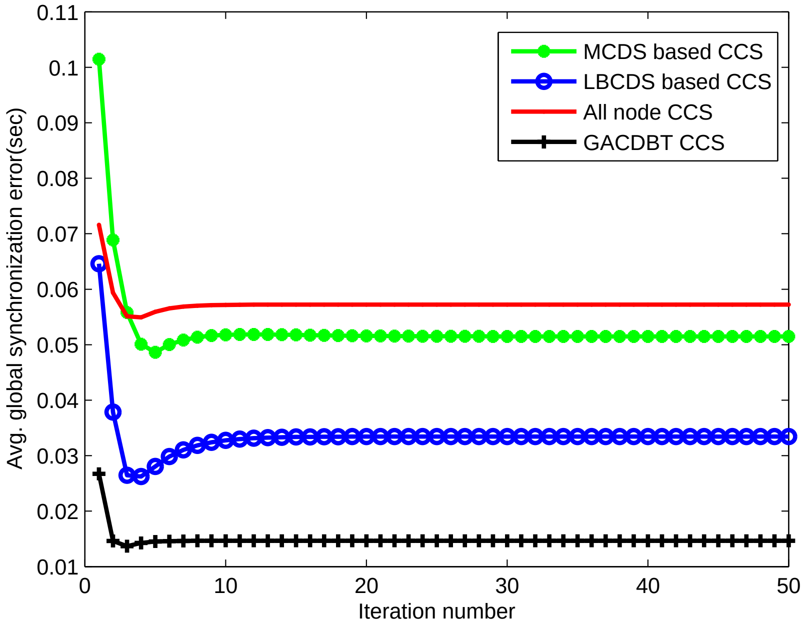


Figure 5.10: Average global synchronization error of CCS on different topological strategies

### (iii) Average local synchronization error

The local synchronization error is averaged for 50 iterations at individual node as shown in Fig. 5.11. The percentage improvement is calculated by considering the maximum local synchronization error among 50 nodes. It is observed that CCS on GACDBT has 63 % less local synchronization error than all node CCS, 64 % less than MCDS based CCS, and 53 % less local synchronization error than LBCDS based CCS.

### (iv) Average number of messages & energy consumption

The average number of messages exchanged and average energy consumption to achieve the acceptable synchronization error is estimated as given in Table 5.5. It is observed that CCS on GACDBT has 71 % less messages exchanged than all node CCS, 40 % less than LBCDS based CCS and 13 % less messages exchanged than MCDS based CCS.

From Table 5.5, it is also observed that CCS on GACDBT has 64 % less energy consumption than all node CCS, 27 % less than LBCDS based CCS and 6 % less energy consumption than MCDS based CCS.

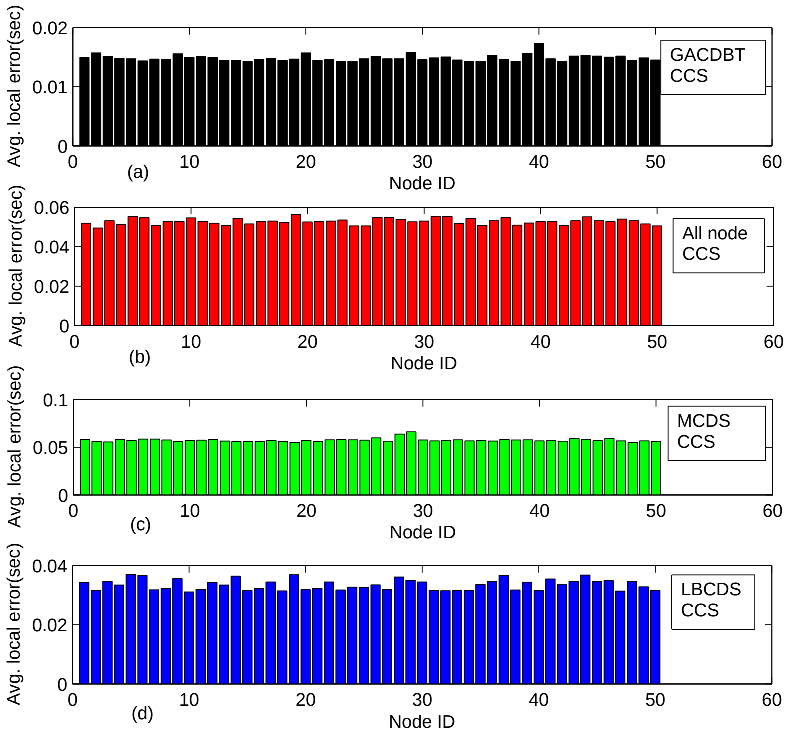


Figure 5.11: Average local synchronization error of CCS on different topological strategies

Table 5.5: Average number of messages & energy consumption of CCS on different topological strategies for 50 nodes

| **Parameters** | **All node CCS** | **LBCDS based CCS** | **MCDS based CCS** | **GACDBT based CCS** |
|---|---|---|---|---|
| Avg. number of messages | 1024 | 484 | 332 | 287 |
| Avg. energy consumption (in Joule) | 0.0012 | 0.00059 | 0.00046 | 0.00043 |

### (c) Test algorithm-3: SATS

The SATS algorithm which is proposed in Chapter 3 is also tested on the proposed GACDBT on a network of 50 nodes and the results obtained are discussed below.

#### (i) Convergence speed

The convergence speed of SATS in terms of offset convergence is tested for an acceptable synchronization error ($\epsilon$) of 0.0001 sec. as mentioned in Table 5.2. From Fig. 5.12 (a)-(d), it is observed that all node SATS takes around 17 iterations, GACDBT based SATS takes around 11 iterations, MCDS based SATS takes around 25 iterations and LBCDS based SATS takes around 19 iterations to converge to the consensus value 0.5002. So, the convergence speed of SATS on GACDBT has improved by 35 % over all node SATS, 56 % over MCDS based SATS and 42 % over LBCDS based SATS.

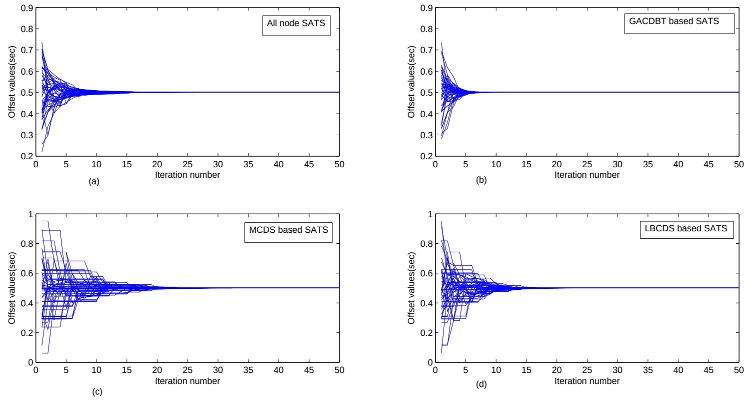


Figure 5.12: Offset convergence of SATS on different topological strategies

### (ii) Average global synchronization error

The average global synchronization error per iteration is shown in Fig. 5.13 for SATS on different topological strategies. It is observed that SATS on GACDBT has, in an average, 81 % less global synchronization error than all node SATS, 66 % less than MCDS based SATS and 53 % less synchronization error than LBCDS based SATS.

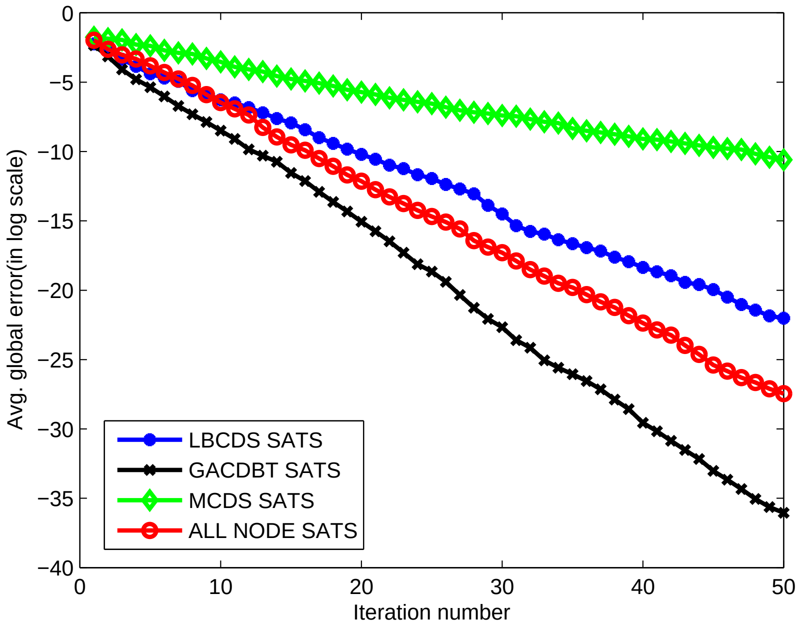


Figure 5.13: Average global synchronization error of SATS on different topological strategies

### (iii) Average local synchronization error

The local synchronization error is averaged for 50 iterations at individual node as shown in Fig. 5.14. The percentage improvement is calculated by considering the maximum local synchronization error among 50 nodes. It is observed that SATS on GACDBT has 46 % less local synchronization error than all node SATS, 72 % less than MCDS based SATS, and 60 % less local synchronization error than LBCDS based SATS.

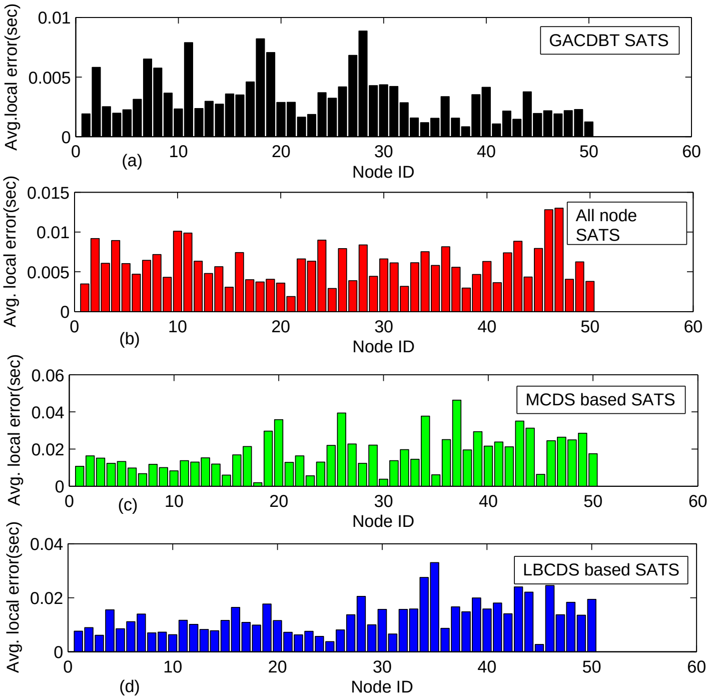


Figure 5.14: Average local synchronization error of SATS on different topological strategies

### (iv) Average number of messages & energy consumption

The average number of messages exchanged and average energy consumption to achieve the acceptable synchronization error is estimated as given in Table 5.6. For computing the average energy consumption, the equation derived in chapter 3 is followed. It is observed that SATS on GACDBT has 83 % less messages exchanged than all node SATS, 76 % less than LBCDS based SATS and 51 % less messages exchanged than MCDS based SATS.

From Table 5.6, it is also observed that SATS on GACDBT has 62 % less energy consumption than all node SATS, 38 % less than LBCDS based SATS and 18 % less energy consumption than MCDS based SATS.

Table 5.6: Average number of messages & energy consumption of SATS on different topological strategies for 50 nodes

| **Parameters** | **All node SATS** | **LBCDS based SATS** | **MCDS based SATS** | **GACDBT based SATS** |
|---|---|---|---|---|
| Avg. number of messages | 932 | 662 | 329 | 158 |
| Avg. energy consumption (in Joule) | 0.0011 | 0.00067 | 0.0005 | 0.00041 |

### 5.5.3 Comparative evaluation of CTS algorithms on proposed GACDBT strategy

In this Section, the comparative evaluation of ATSP, CCS and SATS algorithms are carried out on the proposed GACDBT strategy using the same performance metrics. The results obtained are discussed below.

**(i) Convergence speed**

The convergence speed is tested through offset convergence. The offset values for 50 nodes are recorded by considering maximum 50 iterations as shown in Fig. 5.15 (a)-(c). It is observed that ATSP [22] on GACDBT takes around 15 iterations to achieve convergence, CCS [17] algorithm which uses cumulative weighted averaging (CWA) takes approximately 10 iterations to convergence, both with an acceptable synchronization error ($\epsilon$) of 0.0001 sec. Whereas our SATS algorithm on GACDBT achieves convergence within 8 iterations for the same value of acceptable synchronization error. So, convergence speed of SATS on GACDBT has improved by 20 % over CCS on GACDBT and almost 50 % over ATSP on GACDBT.

**(ii) Average local synchronization error**

Fig. 5.16 (a)-(c) depicts average local synchronization error of each node for a topology of 50 nodes for 50 iterations. This shows the upper bound of local synchronization error of every node. By considering the maximum local synchronization error, it is observed that SATS on GACDBT has less local synchronization error, nearly 79%, as compared to ATSP on GACDBT and 65% as compared to CCS on GACDBT.

**(iii) Average global synchronization error**

Fig. 5.17 shows average global(network-wide) synchronization error of 50 nodes after each iteration by considering maximum 50 iterations. It is observed that SATS on GACDBT has less synchronization error, nearly 85%, as compared to ATSP on GACDBT and 71% as compared to CCS on GACDBT.

### 5.5.4 Scalability

To study the scalable performance of the proposed GACDBT strategy, the above test algorithms are also evaluated and compared with other topological strategies by varying the number of nodes from 100-500 as shown in Tables 5.7-5.11. Due to randomness of the topology, the average is calculated for 100 realizations of such random topologies for each number of nodes. The scalability is analyzed by estimating (i) Average Mean Square synchronization Error, (ii) Average Number of Iterations, (iii) Average Number of Messages Exchanged and (iv) Average Energy Consumption as shown in the tables. From the tables,

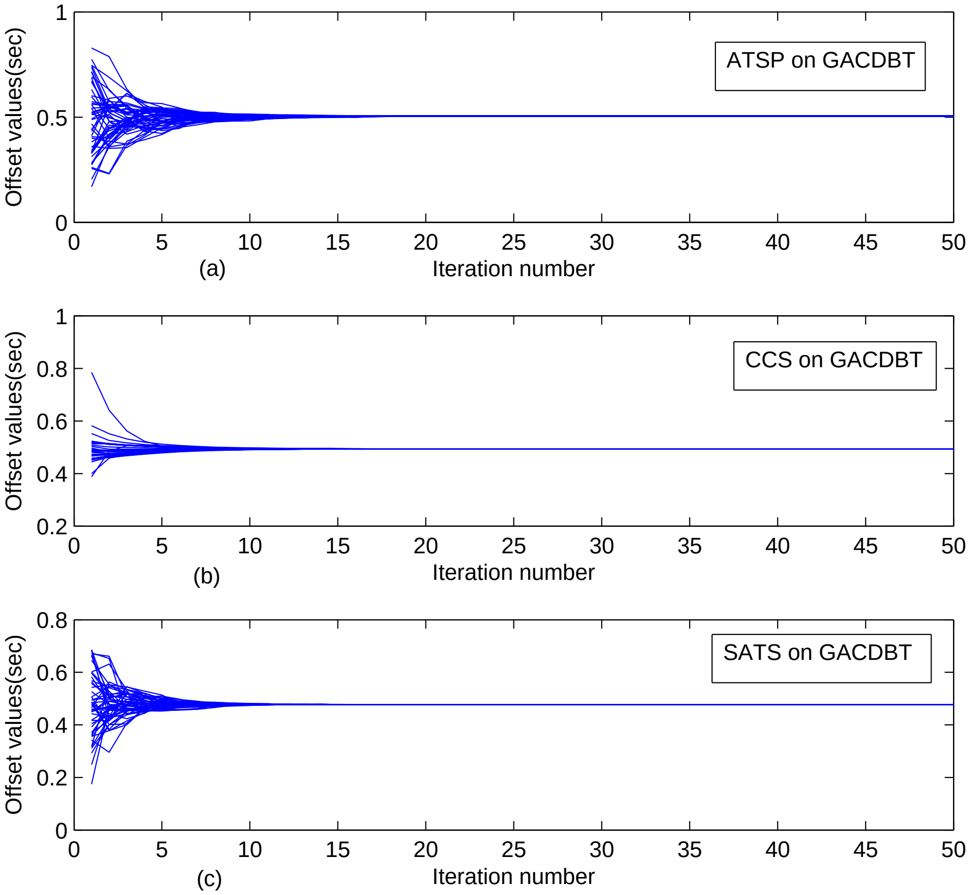


Figure 5.15: Offset convergence of (a) ATSP (b) CCS, and (c) SATS on GACDBT

it is observed that the proposed strategy's performance remains consistent and optimized as compared to other strategies with the increase in number of nodes.

Table 5.7: Performance of CTS algorithms on different topological strategies for network of 100 nodes

| **Network size=100 nodes, Random dense topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0114 | 20 | 4064 | 0.011 |
| | MCDS | 0.0093 | 19 | 3027 | 0.0040 |
| | LBCDS | 0.0069 | 17 | 3678 | 0.0043 |
| | GACDBT | 0.0059 | 15 | 1025 | 0.0019 |
| CCS | All node | 0.0098 | 19 | 2124 | 0.0032 |
| | MCDS | 0.0073 | 18 | 1063 | 0.0015 |
| | LBCDS | 0.0061 | 16 | 1208 | 0.0017 |
| | GACDBT | 0.0040 | 14 | 956 | 0.0012 |
| SATS | All node | 0.0008 | 15 | 2112 | 0.0030 |
| | MCDS | 0.0017 | 18 | 1218 | 0.0019 |
| | LBCDS | 0.0012 | 17 | 1015 | 0.0014 |
| | GACDBT | 0.0007 | 12 | 936 | 0.0011 |

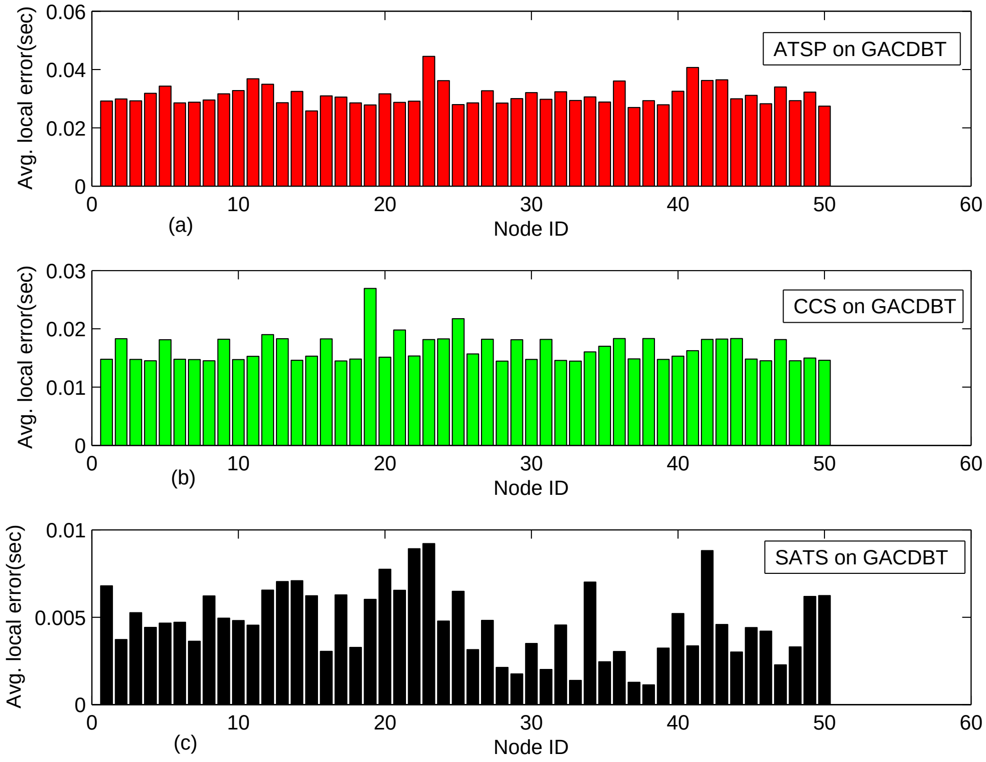


Figure 5.16: Average local synchronization error of (a) ATSP, (b) CCS, and (c) SATS on GACDBT

Table 5.8: Performance of CTS algorithms on different topological strategies for network of 200 nodes

| **Network size=200 nodes, Random dense topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0092 | 21 | 8878 | 0.0137 |
| | MCDS | 0.0075 | 20 | 6437 | 0.0070 |
| | LBCDS | 0.0061 | 19 | 6989 | 0.0072 |
| | GACDBT | 0.0058 | 18 | 2029 | 0.0020 |
| CCS | All node | 0.0013 | 15 | 4232 | 0.0041 |
| | MCDS | 0.0012 | 13 | 2115 | 0.0023 |
| | LBCDS | 0.0011 | 11 | 2813 | 0.0031 |
| | GACDBT | 0.0009 | 09 | 1827 | 0.0019 |
| SATS | All node | 0.00079 | 13 | 3688 | 0.0039 |
| | MCDS | 0.00192 | 15 | 1845 | 0.0025 |
| | LBCDS | 0.00157 | 12 | 1676 | 0.0021 |
| | GACDBT | 0.00073 | 10 | 1238 | 0.0018 |

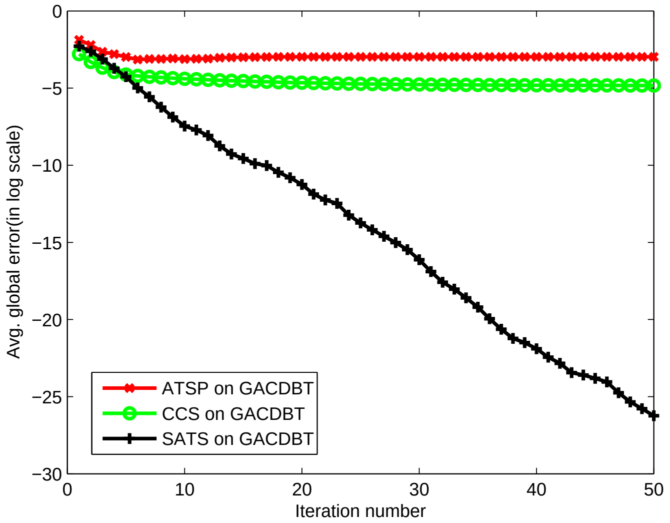


Figure 5.17: Average global synchronization error of ATSP, CCS, and SATS on GACDBT

Table 5.9: Performance of CTS algorithms on different topological strategies for network of 300 nodes

| **Network size=300 nodes, Random dense topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0071 | 19 | 13583 | 0.0193 |
| | MCDS | 0.0054 | 18 | 10045 | 0.0079 |
| | LBCDS | 0.0034 | 17 | 11056 | 0.0081 |
| | GACDBT | 0.0033 | 16 | 4031 | 0.0023 |
| CCS | All node | 0.0041 | 17 | 5125 | 0.0049 |
| | MCDS | 0.0039 | 16 | 2719 | 0.0033 |
| | LBCDS | 0.0036 | 14 | 3011 | 0.0034 |
| | GACDBT | 0.0031 | 12 | 2535 | 0.0029 |
| SATS | All node | 0.00080 | 12 | 5012 | 0.0041 |
| | MCDS | 0.00195 | 14 | 2452 | 0.0031 |
| | LBCDS | 0.00162 | 13 | 2181 | 0.0029 |
| | GACDBT | 0.00071 | 11 | 1667 | 0.0021 |

Table 5.10: Performance of CTS algorithms on different topological strategies for network of 400 nodes

| **Network size=400 nodes, Random dense topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0087 | 23 | 19692 | 0.0198 |
| | MCDS | 0.0072 | 22 | 15650 | 0.0083 |
| | LBCDS | 0.0067 | 20 | 16054 | 0.0090 |
| | GACDBT | 0.0062 | 18 | 10032 | 0.0082 |
| CCS | All node | 0.0064 | 16 | 5912 | 0.0052 |
| | MCDS | 0.0062 | 15 | 3432 | 0.0043 |
| | LBCDS | 0.0059 | 14 | 3971 | 0.0039 |
| | GACDBT | 0.0056 | 11 | 3128 | 0.0035 |
| SATS | All node | 0.0078 | 12 | 6832 | 0.0043 |
| | MCDS | 0.0021 | 11 | 3274 | 0.0036 |
| | LBCDS | 0.0020 | 10 | 2912 | 0.0033 |
| | GACDBT | 0.00089 | 10 | 2119 | 0.0028 |

Table 5.11: Performance of CTS algorithms on different topological strategies for network of 500 nodes

| Network size=500 nodes, Random dense topology | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0059 | 23 | 21796 | 0.0204 |
| | MCDS | 0.0036 | 22 | 19945 | 0.0088 |
| | LBCDS | 0.0035 | 21 | 20115 | 0.0089 |
| | GACDBT | 0.0032 | 19 | 18035 | 0.0086 |
| CCS | All node | 0.0039 | 14 | 7915 | 0.0061 |
| | MCDS | 0.0038 | 13 | 4112 | 0.0032 |
| | LBCDS | 0.0035 | 12 | 4305 | 0.0033 |
| | GACDBT | 0.0032 | 11 | 3917 | 0.0030 |
| SATS | All node | 0.0078 | 12 | 7523 | 0.0045 |
| | MCDS | 0.0021 | 11 | 3751 | 0.0034 |
| | LBCDS | 0.0020 | 10 | 3197 | 0.0032 |
| | GACDBT | 0.00094 | 09 | 2703 | 0.0030 |

## 5.6 Summary

In this Chapter, a GA based topological optimization strategy is proposed, based on delay balanced topology concept, to accelerate CTS algorithms for wireless sensor network. Extensive simulations have been carried out to show the effectiveness and scalability of the proposed strategy on recent and state-of-the-art CTS algorithms. Simulation results show that using the proposed strategy, the number of iterations for consensus convergence, mean square synchronization error, the number of messages exchanged to achieve consensus and energy consumption have been optimized significantly. In the next Chapter, the topological balancing strategy is proposed for sparse topology using Random Weighted Genetic Algorithm (RWGA) based approach.

# Chapter 6

# Topological Optimization Strategy for Consensus Time Synchronization Algorithms on sparse topology: A Random Weighted Genetic Algorithm based Approach

In the previous Chapter, a topological optimization strategy is proposed for improving the performance of consensus-based synchronization algorithms for WSNs. In fact, the network is considered to have a densely deployed topology where the degree of connectivity has a significant impact on the medium access delay incurred in the network. But, in a sparse network, the physical layer delay also has an equivalent impact. It is solely dependent on the propagation delay along the multi-hop path which is directly proportional to the Euclidean distance between the nodes in a wireless medium. In this Chapter, a topological optimization strategy is proposed, considering both the degree of connectivity and Euclidean distance as parameters in the objective functions. In order to handle the trade-off between the objective functions, a multi-objective Random Weighted Genetic Algorithm (RWGA) approach is presented. Using this optimized communication topology, it is shown that some recent state-of-the-art CTS algorithms have shown improved performance.

## 6.1 Introduction

The objective of topological optimization in various problem domain of WSNs is highlighted in Chapter 5. It is also incorporated with some existing time synchronization algorithms, as discussed in Chapter 5, to improve their performances. In this chapter, we have targeted towards developing topological optimization strategy for CTS algorithms on the sparse sensor network. For such type of network, the physical layer delay has equal impact on CTS algorithms' performance as medium access delay. In a wireless medium, the physical layer delay is directly related to the hop count between two nodes which in turn, can be equivalently replaced with the Euclidean distance between them. Considering these two metrics, a multi-objective Random Weighted Genetic Algorithm (RWGA) approach is proposed in this Chapter to obtain an optimal logical topology.

The major contributions of this Chapter are the following:

1. Proposes a generic topological optimization strategy for CTS algorithms that significantly minimizes physical and MAC layer delay.
2. Proposes a novel RWGA based multi-objective approach to select the logical, optimized topology to handle the trade-off between the delay cost functions.
3. Demonstrates the efficacy of the proposed strategy by conducting extensive simulations for some state-of-the-art CTS algorithms and using standard performance metrics like number iterations for convergence, total synchronization error, the number of exchanged messages and energy consumption.

The rest of the Chapter is organized as follows. Section 6.2 highlights the system models and definitions used in this Chapter. A priori analysis and problem formulation is presented in section 6.3. Section 6.4 gives a brief overview of the multi-objective approach based on RWGA. Section 6.5 presents the RWGA based proposed strategy in detail. The simulation results and discussion is given in Section 6.6, followed by conclusion in Section 6.7.

## 6.2 System Model & Definitions

In this Chapter, the same CTS framework, clock model, and energy model are followed as presented in Chapter 5 except the network model which is described below. The definitions used in this Chapter are also same as given in Chapter 5.

### 6.2.1 Network Model

For our proposed strategy, the WSN is assumed to be a random, sparse and weighted graph *G= (V, W)*, where *V* denotes set of *'n'* nodes, set *'W'* represents a weight matrix. Two nodes are said to be neighboring nodes if the Euclidean distance $l_{ij}$ between them is less than the connectivity radius. *W* is a $n \times n$ weight matrix with $\tau \times n \times n$ number of non-zero entries and the non-zero entries are represented as $w_{ij}$ in the matrix. If node $v_i$ and $v_j$ are neighboring nodes, then $w_{ij} = w_{ji} = l_{ij}$. Otherwise, $w_{ij} = w_{ji} = 0$. All nodes have unique IDs. The communication channel between a pair of nodes is assumed to be static, symmetric and undirected, i.e., upstream delay, and downstream delay is same. $N_i = j : (i, j) \in E$, denotes the set of one-hop neighbors of node *'i'*. The communication topology is assumed to be fully distributed where there is no special node such as root or reference node.

## 6.3 Problem Formulation

From the framework of CTS algorithm presented in Chapter 5, the following subsection first analyzes the relationship between delay and topological parameters and then formulate the objective.

### 6.3.1 A Priori Analysis

The major factor that affects the synchronization accuracy and convergence speed is the delay incurred at the physical and MAC layer to send the synchronization packets [66]. During topology control (optimization) phase, since, the communication delay estimation is not feasible [82], some topological parameters need to be identified to model the delay cost. From literature [66][67][80], it is observed that the physical layer delay is proportional to the distance between the nodes in a wireless medium, i.e., $T_{PHY_{delay}} \propto l_{ij}$, $l_{ij}$ is the Euclidean distance between sender and receiver.

On the other hand, as multiple SP nodes are associated with a single SI node, there exists a channel competition among SP nodes at the MAC level to send the synchronization reply messages. Assuming each SP node has an equal probability of accessing the channel, the MAC delay, as a result of SP nodes competition, is directly influenced by the number of SP nodes associated with an SI node, i.e., the degree of connectivity of SI nodes, i.e., $T_{MAC_{delay}} \propto d_i$ [82], $d_i$ is the degree of connectivity of the node i. As a matter of fact, there are various other parameters, both deterministic and random, e.g., network traffic, protocol processing time, transmission collision, that count for the delay. Since our focus is to design a generic optimization strategy at topology control phase, these two major topological parameters (Euclidean distance and degree of connectivity) are considered to model the delay cost function.

Further, an empirical analysis is conducted to study the relationship between average Euclidean distance and the average degree of connectivity for generic random topology. Random topologies are generated by deploying 50-1000 nodes randomly and uniformly in an area of $10\times10$ square unit. To make the random topology connected, the radius of connectivity is calculated as: $r(L,n) = L\sqrt{(\frac{2\log n}{n})}$, where $L$=side of square deployed area, and $n$=number of nodes [72]. For each number of nodes, 100 instances of topology are generated, and the average value is calculated and normalized for the average degree of connectivity and average Euclidean distance.

From the study, as shown in Fig. 6.1, it is observed that a trade-off exists between the average degree of connectivity and average Euclidean distance. So, minimization of both these conflicting parameters at topological optimization phase, to minimize both physical and MAC delay during communication phase, can be better modeled as a multi-objective optimization problem which is discussed below.

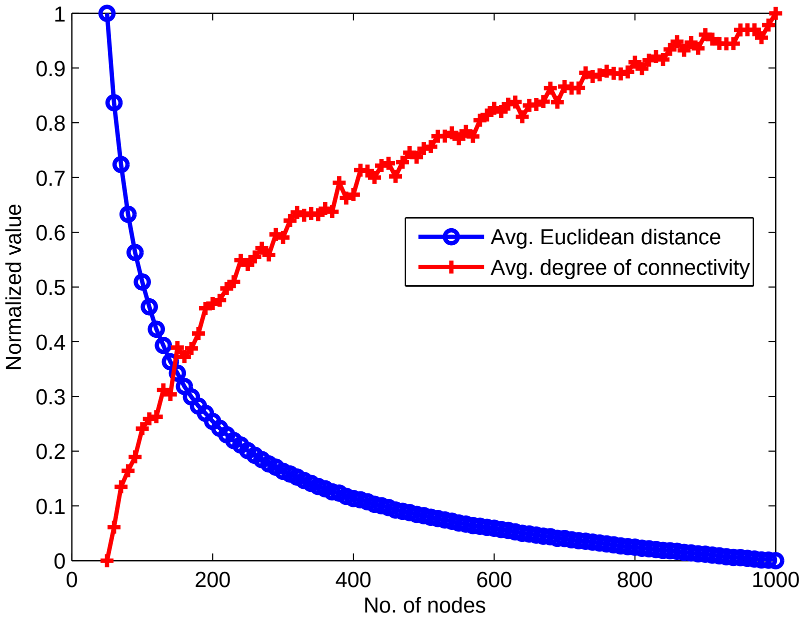


Figure 6.1: Trade-off between Avg. degree of connectivity and Avg. Euclidean Distance in Random Topology

### 6.3.2 Problem Objective

In this Section, the overall problem is also defined and named as CDS based Delay Balance Topology (CDSDBT) problem as in Chapter 5 with one more objective function. We adopt the following notations in the problem formulation. Let,

(i) the set of total sensor nodes is denoted by $V$ where $|V|=n$ .

(ii) the set of SI nodes is denoted by $I=\{i_1,i_2,...,i_m\}$, $m < n$.

(iii) the set of SP nodes is denoted by $P=\{p_1,p_2,...,p_{n-m}\}$ and $I \cup P=V$.

(iv) the set of SI nodes to which a SP node $p_k$ can be assigned is denoted by $G_k$

(v) the Boolean variable $b_{kj}$=1 ,if the SP node $p_k$ is assigned to the SI node $i_j$ and $b_{kj}$=0, otherwise.

(vi) the degree of connectivity at a SI node *'i'* is given by $d_i$.

(vii) the Euclidean distance between node *'i'* and *'j'* is given by $l_{ij}$

Based on the definition 4 given in Chapter 5, our objectives are defined as follows.

**Objective (a):** To select a subset of connected dominating sensors, known as SI nodes, to minimize message complexity for energy saving.

**Objective (b):** To assign the remaining nodes, known as SP nodes, to the SI nodes with minimum Euclidean distance for physical layer delay minimization.

$$Minimize \sum_{i=1}^{m} \sum_{j \in N_i} l_{ij} \tag{6.1}$$

**Objective (c):** To balance the MAC layer delay by minimizing the variance of degree of connectivity at each SI node with the overall delay being minimized.

$$Minimize \sum_{i=1}^{m} |d_i - \mu|, \mu = \frac{1}{m}\sum_{i=1}^{m} d_i \tag{6.2}$$

$$subject\ to \sum_{i_j \in G_k} b_{kj} = 1, \forall p_k \in P \tag{6.3}$$

The constraint in Equation 6.3 signifies the assignment of a SP node to exactly one SI node.

### 6.3.3 Motivating Example

A straightforward solution to the objective (a) is finding out an MCDS with the required constraint for the given network and assigning the role of SI nodes to the dominators and SP nodes to dominatees. Similar approaches exist in [43][77]. But, for consensus-based synchronization algorithms, MCDS approaches can minimize message complexity because of the lesser number of dominators as shown in Fig.6.3 , but may not ensure objective (b) and (c) which is necessary for optimal consensus convergence.

For example, Fig. 6.3 shows the MCDS based sensor selection for the WSN shown in Fig. 6.2. The SI nodes are {4,7}, which are the MCDS of the given topology, and the SP nodes are {1, 2, 3, 5, 6, 8}. The SP nodes are assigned to the SI nodes satisfying constraint mentioned in (6.3). Whereas, in Fig. 6.4, the CDS is first constructed based on objective (c). Then the topology is optimized by reassigning SP node 4 to possible dominators to satisfy both objective (b) and (c). The resultant topologies thus obtained by the brute force method are shown in Fig. 6.4(a)-(c). The objective functions are evaluated using (6.1) and (6.2) and are given in Table 6.1. It can be observed that scenario-3 is the optimal topology as compared to MCDS based topology. Further, from the table 6.1, a comparative analysis among different optimized topologies (scenario-1, 2 and 3) reveals that a trade-off exists while optimizing both the objective functions.

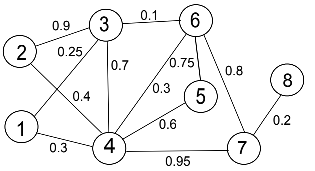


Figure 6.2: WSN topology of 8 nodes

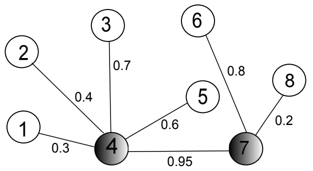


Figure 6.3: MCDS based sensor topology for Fig. 6.2

## 6.4 Multi Objective Approach

Due to the intractability nature of CDSDBT problem which is already proved in Chapter 5, Genetic Algorithm (GA) is one of the most suitable heuristics that can be applied for

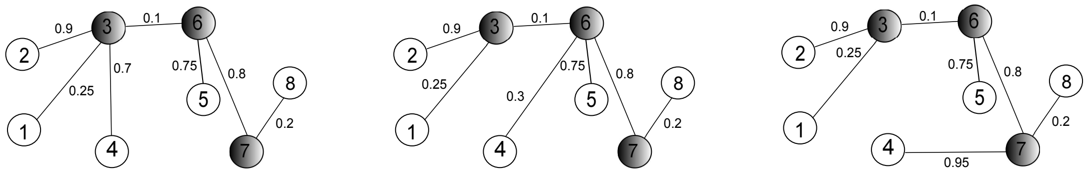


Figure 6.4: (a) Scenario-1: CDS based optimized topology, (b) Scenario-2: CDS based optimized topology, (c) Scenario-3: CDS based optimized topology

Table 6.1: Objective functions for different topology selections

| Objective Functions<br>Topological Strategy | $Objective\ (b) : (\sum_{i=1}^{m} \sum_{j \in N_i} l_{ij})$ | $Objective\ (c) : (\sum_{i=1}^{m} \lvert d_i - \mu \rvert)$ |
|---|---|---|
| MCDS based | 4.9 | 2 |
| CDS based (Scenario-1) | 4.6 | 2.6 |
| CDS based (Scenario-2) | 4.2 | 2.6 |
| CDS based (Scenario-3) | 4.8 | 1.8 |

efficient topological optimization from such a large solution space. In fact to handle the trade-off between both the delay metrics as discussed above, a RWGA based multi-objective approach is proposed to get Pareto optimal solutions for the given problem. A brief overview of multi-objective approach and the detailed proposed RWGA method is described below.

### 6.4.1 Preliminaries & Background

The solution to the CDSDBT problem involves in finding out a CDS based topology which optimizes the two objective functions, namely objective (b) and (c), as discussed in Section 6.3. So, it can be formulated as a multi-objective optimization problem. A general multi-objective optimization (minimization) problem is expressed as [87]:

Min $F(x) = ((f_1(x), f_2(x), ..., f_k(x))^T$ s.t. x∈ $S$, $x = ((x_1, x_2, ..., x_n))^T$
where $f_1(x), f_2(x), ..., f_k(x)$ are the $k$ objective functions, $(x_1, x_2, ..., x_n)$ are the n optimization parameters and S $\in R^n$ is the solution or parameter space.
The aim of multi-objective optimization problems is to handle all the possible tradeoffs among multiple objective functions that are usually conflicting. Since it is difficult to select a single solution for a multi-objective optimization problem without iterative interaction with the decision maker, one general approach is to show the set of Pareto optimal solutions to the decision maker [88].Then one of the Pareto optimal solutions can be chosen depending on the preference. A Pareto optimal solution is defined as follows:

**Pareto optimal solution**:$x^*$ is said to be a Pareto optimal solution if there exists no other feasible x $\in$ S such that,

$f_j(x) \leq f_j(x^*)$, $\forall$ j $\in \{1, 2...m\}$ and

$f_j(x) < f_j(x^*)$, for atleast one objective function.

To solve multi objective optimization problem by GA, a number of variant of GA have been proposed [87]. In [88], the authors have proposed a MOGA based on a weighted sum of multiple objective functions, known as RWGA, where a normalized weight vector $w_i$ is randomly generated for each solution $x_i$ during the selection phase at each generation. This approach aims to stipulate multiple search directions in a single run without using additional parameters. Due to its simplicity and suitability for discrete objective functions as in our case, this approach is adopted for our problem. The general procedure of the RWGA is given in Fig. 6.5 and a brief overview is given below.

### 6.4.2 Overview of RWGA

**Selection Procedure**

A classical approach to combine multiple objective functions into a scalar fitness function is using Equation 6.4.

$$f(x) = \sum_{i=1}^{n} w_i f_i(x) \tag{6.4}$$

where x is a string (individual), $f(x)$ is a combined fitness function, $f_i(x)$ is the $i^{th}$ objective function, $w_i$ is a weight value for $f_i(x)$ , and *'n'* is the number of objective functions. If constant weights are assigned to $w_i$ in Equation 6.4 then the search will be restricted to only one direction. RWGA proposes to assign random weights to search for Pareto optimal solutions by exploring various search directions. Each time when a pair of strings are selected for crossover, random weights are assigned as given in Equation 6.5.

$$w_i = \frac{rand_i}{\sum_{j=1}^{n} rand_j}, i = 1, 2, ..., n \tag{6.5}$$

where $rand_j$ is a positive random number and $w_i$ is a real number in the closed interval [0,1].

**Elite Preserve Strategy**

During the execution of the RWGA, a tentative set of Pareto optimal solutions is stored and updated at every generation. A certain number (say, $N_{elite}$) of individuals are randomly selected from the set at each generation. Those solutions are used as elite individuals in RWGA. This elite preserve strategy has an effect in keeping the variety of each population in RWGA.

## 6.5 Proposed RWGA based Strategy for CDSDBT Problem

To solve the CDSDBT problem, a set of nodes needs to be initially selected as SI nodes which must form a CDS to satisfy objective (a). Most of the works on CDS have a

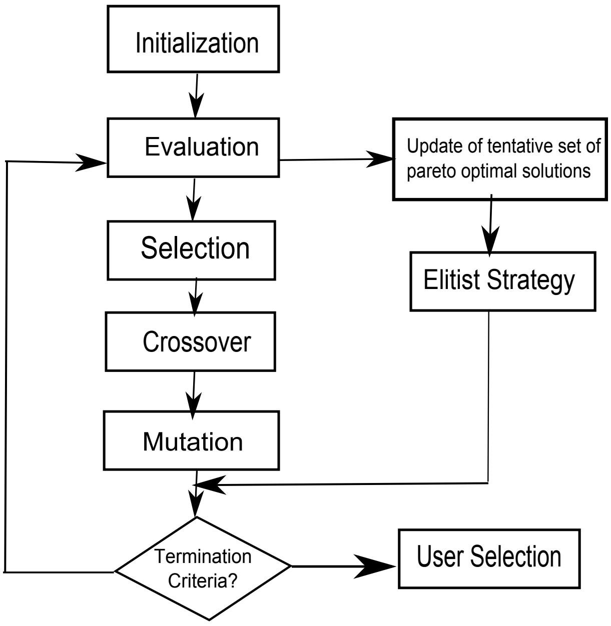


Figure 6.5: Flow chart of RWGA method

common objective to minimize the cardinality of CDS [89]. But, MCDS does not ensure balanced allocation of dominatees which is a prerequisite for CDSDBT problem. Recently, meta-heuristics are proposed by A. Potluri *et al.*[90] for computing capacitated dominating set with uniform and variable capacities which resembles CDSDBT problem. But, they do not consider the connectivity among dominators and the Euclidean distance between the nodes which is primarily required for the CDSDBT problem. A CDS heuristic is proposed by Jing *et al.* [83] to find out a load balanced Connected Dominating Set. We have used this heuristic for initial SI nodes selection. Then, the proposed RWGA is used to assign the SP nodes to get the optimal CDS based delay balanced topology. The detail GA implementation is given below.

### 6.5.1 Chromosome encoding

Each chromosome is encoded as a structured string of length $(n - m)$ where $(n - m)$ is the number of SP nodes. A gene at any position '$i$', denoted as $G_i$, represents a triplet as shown in Fig. 6.6. The first value of the triplet represents an SP node id; second value represents a SI node id and the third value is the Euclidean distance between them. The nodes are assigned with ids as 1, 2,..., n. The following example illustrates the chromosome encoding.

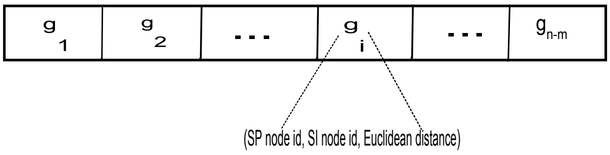
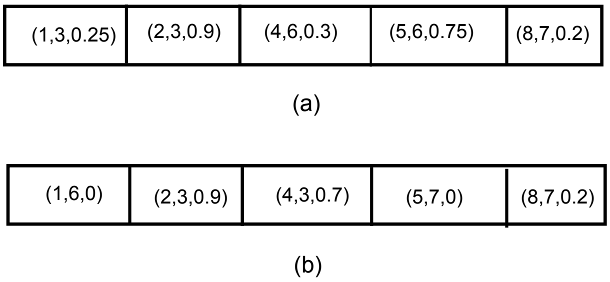


Figure 6.6: Chromosome Encoding

Example: Consider a WSN of 8 nodes as shown in Fig. 6.2 where the set of SI nodes are obtained by using the existing CDS heuristic is $I$={3, 6, 7}. So, the set of SP nodes is given by $P$={1, 2, 4, 5, 8}. Thus, the length of the chromosome is 5. Fig. 6.7 (a) shows a valid chromosome representation. The value at gene position 1 is (1, 3, 0.25) which means SP node 1 is assigned to SI node 3 and the Euclidean distance between them is 0.25. A similar interpretation can be drawn for other positions.



| | | | | |
|---|---|---|---|---|
| (1,3,0.25) | (2,3,0.9) | (4,6,0.3) | (5,6,0.75) | (8,7,0.2) |

(a)

| | | | | |
|---|---|---|---|---|
| (1,6,0) | (2,3,0.9) | (4,3,0.7) | (5,7,0) | (8,7,0.2) |

(b)

Figure 6.7: Illustrative Examples:(a) Valid chromosome (b) Invalid Chromosome

### 6.5.2 Initial Population Generation

The initial population for GA is generally generated randomly. But, in our problem, total randomness may generate invalid chromosomes which will make the selection process slower, and the result will not be a valid topology. The idea is illustrated below to differentiate between valid and invalid chromosome. In Fig. 6.7 (b), the gene value at position 4 is (5, 7, 0) which means SP node 5 is assigned to SI node 7. Though node 7 is a SI node, it is not a neighboring SI node of SP node 5 as given in Table 6.2. So, this creates invalid chromosome for our problem. To create valid chromosomes, a data structure is first created as shown in Table 6.2. The table contains the list of SP nodes in the first column and their respective neighboring SI nodes with the Euclidean distance in the second column. The valid chromosomes are then generated using Table 6.2 by assigning each SP node to one of the randomly selected neighboring SI nodes instead of assigning it to any randomly selected SI node. All the chromosomes generated in this process represent valid assignments of SP node to SI nodes but may not be optimal. The fitness value of each chromosome is evaluated as given below.

### 6.5.3 Fitness Evaluation

The fitness function of each chromosome is defined as given in Equation 6.6 by taking a weighted sum of Equation 6.1 and Equation 6.2 where the weight values of $w_1$ and $w_2$ are assigned using Equation 6.5 and a tentative set of Pareto optimal solutions are preserved

Table 6.2: SP nodes with possible neighbor SI nodes

| $SP\ node\ id$ | $(Neighbour\ SI\ node\ id,\ Euclidean\ distance)$ |
|---|---|
| 1 | (3,0.25) |
| 2 | (3,0.9) |
| 4 | (3,0.7),(6,0.3),(7,0.95) |
| 5 | (6,0.75) |
| 8 | (7,0.2) |

which is known as the elite group.

$$Minimize\,D = w_1 \sum_{i=1}^{m} \sum_{j \in N_i} l_{ij} + w_2 \sum_{i=1}^{m} |d_i - \mu| \tag{6.6}$$

The '$l_{ij}$' value is obtained from the third position of the gene value and '$d_i$' is calculated using Equation 6.7.

$$d_i = f_i + k_i \tag{6.7}$$

where $d_i$=degree of SI node *'i'*, $f_i$=frequency of SI node 'i' in the chromosome, $k_i$=number of neighbor SI nodes of SI node *'i', i=1, 2,..., m*. The '$f_i$' value is obtained by counting the number of appearance of SI node *'i'* in the second gene value and the '$k_i$' value is obtained from the dominators (SI nodes) adjacency matrix. This is illustrated in the following example.

Example: Consider, a valid chromosome according to Fig. 6.7 (a). For the SI nodes $\{3, 6, 7\}$, the degree $d_i$ can be calculated as follows:

$d_3$=$f_3$+ $k_3$=3+ 1=4, $d_6$=$f_6$+ $k_6$=1+ 2=3, $d_7$=$f_7$+ $k_7$=1+ 1=2. The frequency of SI node $f_i$ is obtained from the second position of the gene values in the chromosome. The number of neighboring SI nodes of a particular SI node $k_i$ is obtained from the SI node adjacency matrix E' as shown below. First row corresponds to SI node 3, second row to SI node 6 and third row to SI node 7 and similar for columns.

$$E' = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

### 6.5.4 Selection

The selection step determines which chromosomes from the current generation will mate to create new chromosomes. The selection directs GA search towards promising regions in the search space. A roulette wheel selection mechanism is employed where the individuals on each generation are selected for survival into the next generation according to a probability value '*P*(x)' given by Equation 6.8.

$$P(x) = \frac{D(x) - D_{max}(\eta)}{\sum_{x \in \eta}\{D(x) - D_{max}(\eta)\}}, D_{max}(\eta) = max\{D(x)|x \in \eta\} \tag{6.8}$$

where $D$(x) is the fitness value evaluated using equation for an individual *'x'* in the current population '$\eta$'. This step is repeated for selecting *N/2* pairs of strings from the current populations where *N* is the total number of individuals.

### 6.5.5 Crossover

The crossover takes place between the chromosomes selected in the above step with certain crossover probability '$cr_p$'. We have used 1-point crossover where a point is chosen at random and the two selected chromosomes swap their gene values after that point.

### 6.5.6 Mutation

The mutation operator is applied at a selected gene position instead of random gene position with certain mutation probability '$m_p$'. The selected gene position is decided by Equation 6.9 and the mutated value is given by Equation 6.10.

$$g_k(j, i, dist) = \{i|max\{d_i\}, i \in I \; and \; |G_j| > 1, j \in P\} \tag{6.9}$$

$$g_k(j, i', dist') = \{i' = k|min\{l_{jk}\}, k \in G_j, d_k < d_i, dist' = l_{ji'}\} \tag{6.10}$$

Equation 6.9 signifies that the SI node id with maximum degree is selected from the chromosome for mutation along with the condition that the associated SP node should have more than one neighboring SI nodes. The imposed condition avoids generation of invalid chromosomes. It is then replaced with the neighboring SI node id to which the associated SP node is more closer (less Euclidean distance) and lesser degree than the selected SI node id, according to Equation 6.10. This is illustrated in the following example.

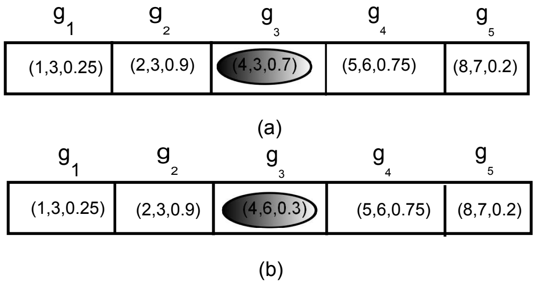


Figure 6.8: (a) Chromosome before mutation, (b) Chromosome after mutation

In chromosome given in Fig. 6.8 (a), the degree of SI node $\{3, 6, 7\}$, calculated using Equation 6.7, are $d_3$=3+ 1=4, $d_6$=1+ 2=3, and $d_7$=1+ 1=2 respectively. The SI node id '3' is having the maximum degree. The associated SP nodes with '3' are 1, 2 and 4 at gene position 1, 2 and 3 respectively. From the Table 6.2, only SP node '4' is having more than

1 neighboring SI nodes (satisfies condition $|G_j| > 1$). So, gene position '$g_3$' is selected for mutation as shown encircled in Fig. 6.8(a). From the Table 6.2, SP node '4' is more closer to neighboring SI node '6' having Euclidean distance 0.3 and $d_6 < d_3$. So, according to (6.9), the mutated gene value at $g_3$ is (4, 6, 0.3). The topology corresponding to chromosome in Fig. 6.8 (a) is shown in Fig. 6.4 (a) and the topology corresponding to chromosome in Fig. 6.8 (b) is shown in Fig.6.4 (b). From the table which shows the objective functions' values, the topology corresponding to mutated chromosome is a Pareto optimal solution.

### 6.5.7 Termination Criteria & User Selection

The proposed strategy terminates when the maximum number of generations is reached. A set of Pareto optimal solutions is obtained as shown in Fig. 6.10. Since, the topology is random, the decision of the user, to select the best optimum trade-off, is quite imprecise in nature. To cope with the impreciseness, the fuzzy based approach [91] is used to select a compromised solution from the set of Pareto front. The fuzziness for each objective function is defined by membership function having values in the range [0,1]. The membership value for $i^{th}$ objective of $j^{th}$ solution in the final Pareto front is calculated using Equation 6.11.

$$\mu_i^j = \begin{cases} 1, & \text{if } F_i \leq F_i^{min} \\ \frac{F_i^{max} - F_i}{F_i^{max} - F_i^{min}}, & \text{if } F_i^{min} < F_i \leq F_i^{max} \\ 0, & \text{if } F_i > F_i^{max} \end{cases} \tag{6.11}$$

where $F_i^{min}$ and $F_i^{max}$ are the minimum and maximum values from non-dominated solutions of each objective function, respectively. For each non-dominated solution, the normalized membership function can be calculated using Equation 6.12.

$$\mu^j = \frac{\sum_{i=1}^{2} \mu_i^j}{\sum_{j=1}^{N} \sum_{i=1}^{2} \mu_i^j} \tag{6.12}$$

The solution with maximum value of $\mu^j$ is a compromised solution that can be selected by the user.

## 6.6 Simulation Results & Discussion

The proposed strategy is simulated under MATLAB environment with an initial population of 100 chromosomes. The maximum number of generations taken is 200 with crossover probability '$cr_p$' equals to 0.8 and mutation probability '$m_p$' equals to 0.05. The optimal chromosome (topology) is selected using the preference criteria as discussed above and then tuned with the PROWLER simulator [30] to study the behavior the synchronization algorithms. The simulation setup for the synchronization algorithms are given in Table 6.4. For comparative analysis, the traditional MCDS and the recent LBCDS topological strategies are also considered. Since, both MCDS and LBCDS are proved to be NP-complete, some recent GA based approaches [84, 85] to these problems are considered to have a fair comparison with our GA based proposal.

### 6.6.1 Performance Evaluation of the Proposed RWGA based Strategy

Fig. 6.9 shows the optimization of the objective functions with the progress of number of generation using the proposed RWGA based strategy. The trade-off between the objective functions is also identified as shown in the Fig. 6.9. At the end of maximum number of generation which is equal to 200 , there is no change in the fitness value. At this point, the feasible Pareto points are extracted and plotted in the two dimensional objective space as shown in Fig. 6.10. From the feasible Pareto points, the Pareto optimal solution is chosen using the fuzzy based approach as discussed above.

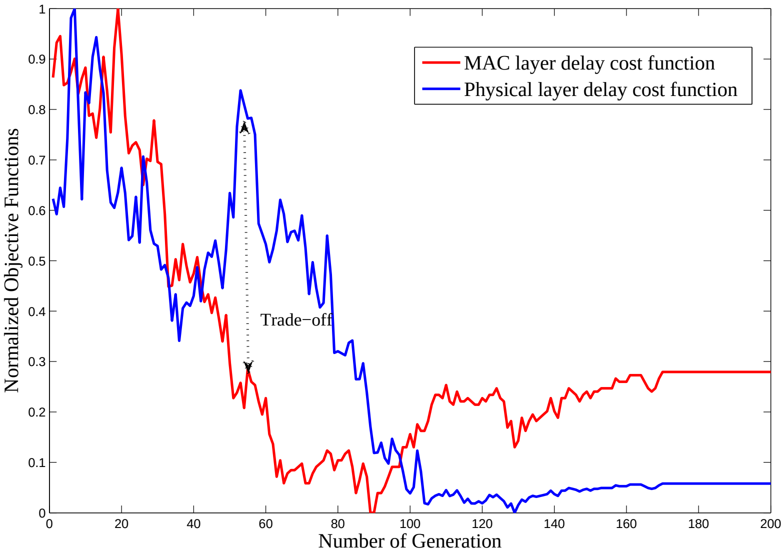


Figure 6.9: Optimization of Objective Functions using the Proposed RWGA Strategy

#### Online Delay Estimation

Fig. 6.11(a)-(d) shows the delay distribution on different topological strategies by estimating the delay in sending and receiving the control packets (RTS/CTS packets) at each node to its one hop neighbors, using the MAC layer time-stamping mechanism for CSMA/CA and following IEEE 802.11 standard [86]. The simulation time taken is 100 sec. and iterated for 50 times. The average delay thus computed at each node, for a random sparse topology of 50 nodes, is plotted as shown in Fig. 6.11.

It is observed that the proposed RGACDBT strategy has a mean delay of 0.1078 sec. whereas the mean delay for all-node, MCDS, and LBCDS strategies are 0.1665 sec., 0.1423 sec. and 0.1167 sec. respectively. So, the proposed strategy minimizes the average delay almost up to 35 % , 24 % and 7 % as compared to the all-node, MCDS and LBCDS strategies

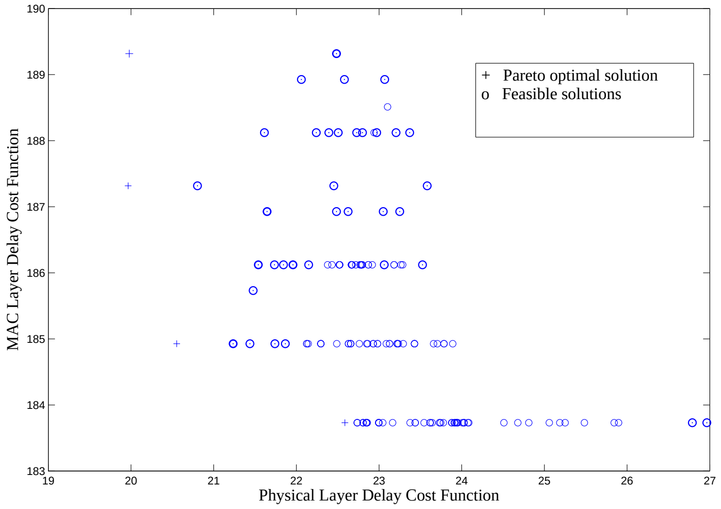


Figure 6.10: Feasible Solutions and Pareto Optimal Solutions generated using the proposed RWGA strategy

respectively. Further, the observed delay variance for the proposed strategy is 0.0027 sec. whereas it is 0.0029 sec. for LBCDS, 0.0036 sec. for MCDS, and 0.0066 sec. for all-node. So, the delay is comparatively more balanced than other strategies. Table 6.3 shows the scalable performance of the proposed strategy w. r. to delay and optimal behavior among other strategies.

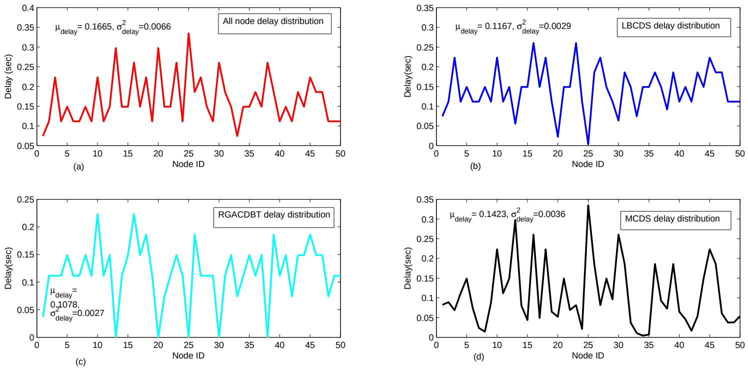


Figure 6.11: Delay Distribution at Individual Node using Different Topological Strategies

### 6.6.2 Independent evaluation of CTS algorithms on the proposed RGACDBT Strategy

To evaluate the performance of CTS algorithms on the proposed strategy, the same state-of-the-art algorithms are considered which are tested in Chapter 5. The typical algorithms which are tested are: (i) Confidence Weighted running Average (CWA) based CCS [17], (ii) ATSP [22], and (iii) SATS proposed in Chapter 3. The ‘‘all node’’ terminology is used for the existing algorithms because they do not use any topological strategy

Table 6.3: Scalability of Delay Analysis on Different Topological Strategies

| Topological Strategy \ No. of nodes | 50 | | 100 | | 150 | | 200 | | 250 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ | $\mu_{delay}$ | $\sigma^2_{delay}$ |
| All node | 0.1665 | 0.0066 | 2.463 | 2.293 | 2.064 | 1.532 | 3.044 | 3.122 | 3.583 | 4.544 |
| LBCDS | 0.1167 | 0.0029 | 1.612 | 0.933 | 1.482 | 0.715 | 2.232 | 1.448 | 2.651 | 1.896 |
| MCDS | 0.1423 | 0.0036 | 1.943 | 1.327 | 1.650 | 0.848 | 2.508 | 1.638 | 3.025 | 2.363 |
| RGACDBT | **0.1078** | **0.0027** | **1.047** | **0.926** | **0.863** | **0.651** | **1.321** | **0.960** | **1.552** | **1.466** |

Table 6.4: Simulation Parameters

| *Parameter* | *Values* |
|---|---|
| *Deployment area* | $10 \times 10$ *square unit* |
| *Topology* | *Random* |
| *No. of nodes* (n) | $50 - 250$ |
| *Initial skew* ($\alpha$) | $uniform(-5, 5)$ |
| *Initial offset* ($\beta$) | $uniform(0, 1)$ |
| *Iteration interval* | 10 *sec* |
| *Acceptable Syn. error* ($\epsilon$) | 0.0001*sec.* |
| *MAC Protocol* | *CSMA/CA* |
| *Communication Standard* | *IEEE* 802.11 |
| *Path Loss Exponent* ($\zeta$) | 2 |
| $\beta_1$ | 45 $nJ/bit$ |
| $\beta_2$ | 10 $pJ/bit$ |
| $\gamma$ | 35 $nJ/bit$ |
| *Message size* (M) | 320 *bits* |

and assumes all node execute the synchronization algorithm. A comparative evaluation of the proposed strategy is also carried out with the traditional MCDS and the recent LBCDS strategy to analyze its performance. The following metrics are considered for the synchronization algorithms, same as in Chapter 5, to study their performance on different topological strategies.

(i) Convergence speed, (ii) Average global synchronization error, (iii) Average local synchronization error, (iii) Average number of messages exchanged, and (iv) Average energy consumption. Finally, the scalability is also studied.

All the algorithms are simulated on a random sparse topology of 50 nodes with random and uniform offset and skew distribution for the clocks as given in Table 6.3. The acceptable synchronization error ($\epsilon$) at every node is set to 0.0001 sec. The timestamps are appended in the RTS/CTS control packets at the MAC layer by closely following the modified MAC layer time-stamping format for IEEE 802.11 standard for CSMA/CA proposed in [86]. The following subsections show the performance of the tested algorithms.

### (a) Test Algorithm-1: ATSP

The ATSP algorithm which uses random pairwise averaging is tested on the proposed RGACDBT on a network of 50 nodes and the results obtained are discussed below.

### (i) Convergence speed

The convergence speed of ATSP in terms of offset convergence is tested for an acceptable synchronization error ($\epsilon$) of 0.0001 sec. as mentioned in Table 6.4. From Fig. 6.12 (a)-(d), it is observed that all node ATSP takes around 26 iterations, RGACDBT based ATSP takes around 19 iterations, MCDS based ATSP takes around 24 iterations, and LBCDS based ATSP takes around 21 iterations to converge to the consensus value 0.5041. So, the convergence speed of ATSP on RGACDBT has improved by 26 % over all node ATSP, 20 % over MCDS based ATSP and 9 % over LBCDS based ATSP.

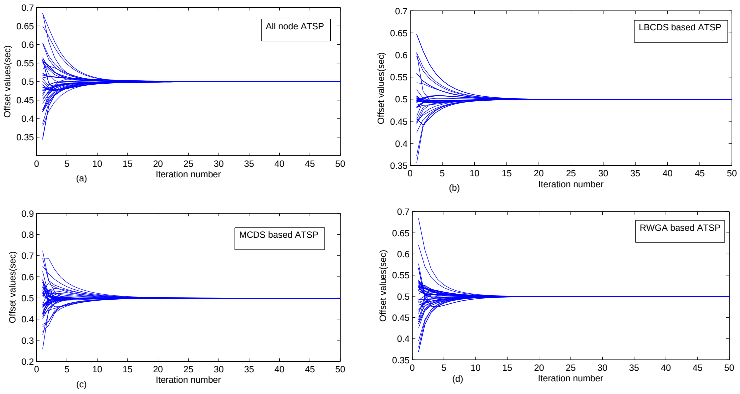


Figure 6.12: Convergence Behavior of ATSP algorithm on different Topological Strategies

### (ii) Average global synchronization error

The average global synchronization error per iteration is shown in Fig. 6.13 for ATSP on different topological strategies. It is observed that ATSP on RGACDBT has, in an average, 42 % less global synchronization error than all node ATSP, 25 % less than MCDS based ATSP and 2 % less synchronization error than LBCDS based ATSP.

### (iii) Average local synchronization error

The local synchronization error is averaged for 50 iterations at individual node as shown in Fig. 6.14. The percentage improvement is calculated by considering the maximum local synchronization error among 50 nodes. It is observed that ATSP on RGACDBT has 65 % less local synchronization error than all node ATSP, 63 % less than MCDS based ATSP, and 25 % less local synchronization error than LBCDS based ATSP.

### (iv) Average number of messages & energy consumption

The average number of messages exchanged and average energy consumption to achieve the acceptable synchronization error is estimated as given in Table 6.5. For computing the average energy consumption, the model discussed in Section 5.2 of Chapter 5 is followed. It is observed that ATSP on RGACDBT has 43 % less messages exchanged than all node ATSP, 30 % less than LBCDS based ATSP and 8 % less messages exchanged than MCDS based ATSP.

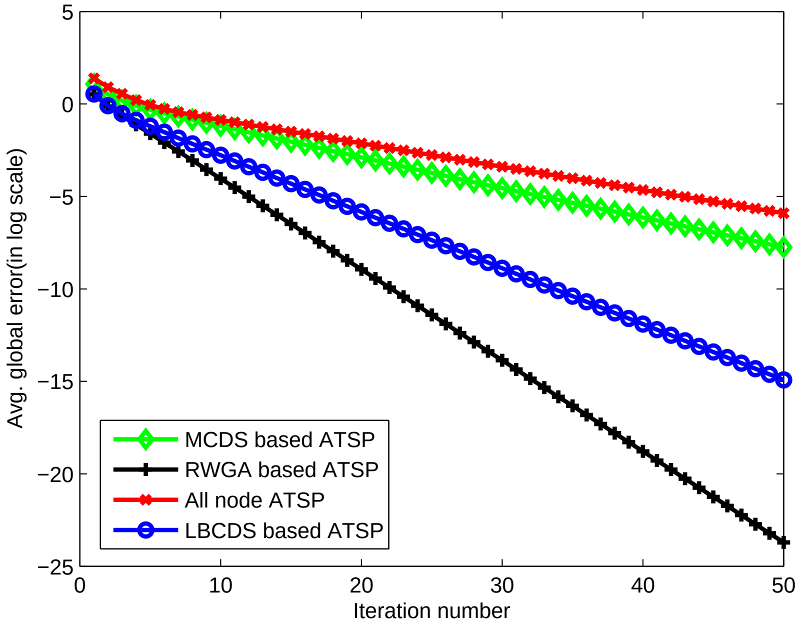


Figure 6.13: Average global synchronization error of ATSP algorithm on different Topological Strategies

From Table 6.5, it is also observed that ATSP on RGACDBT has 25 % less energy consumption than all node ATSP, 11 % less than LBCDS based ATSP and 5 % less energy consumption than MCDS based ATSP.

Table 6.5: Average number of messages & energy consumption of ATSP on different topological strategies for 50 nodes

| **Parameters** | **All node ATSP** | **LBCDS based ATSP** | **MCDS based ATSP** | **RGACDBT based ATSP** |
|---|---|---|---|---|
| Avg. number of messages | 827 | 673 | 514 | 469 |
| Avg. energy consumption (in Joule) | 0.011 | 0.0093 | 0.0087 | 0.0082 |

### (b) Test Algorithm-2: CCS

The CCS algorithm which uses cumulative weighted averaging (CWA) is tested on the proposed RGACDBT on a network of 50 nodes and the results obtained are discussed below.

#### (i) Convergence speed

The convergence speed of CCS in terms of offset convergence is tested for an acceptable synchronization error ($\epsilon$) of 0.0001 sec. as mentioned in Table 6.4. From Fig. 6.15 (a)-(d), it is observed that all node CCS and MCDS based CCS takes beyond 50 iterations to converge to the consensus. RGACDBT based CCS takes around 10 iterations and LBCDS based CCS takes around 15 iterations to converge to the weighted consensus value 0.5011. So, the

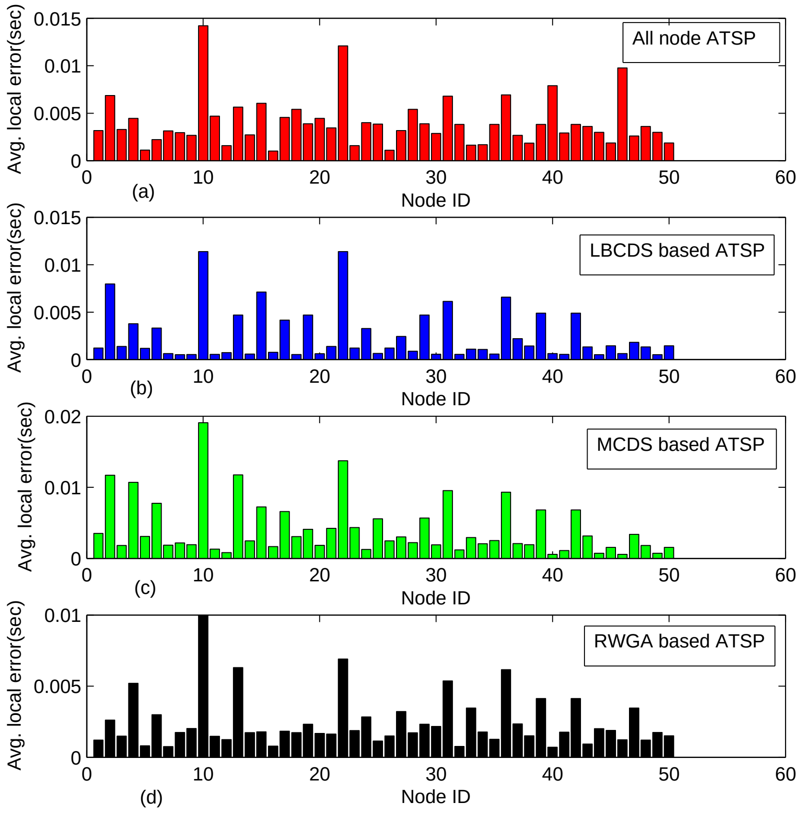


Figure 6.14: Average local synchronization error of ATSP algorithm on different Topological Strategies

convergence speed of CCS on RGACDBT has improved significantly over all node CCS and MCDS based CCS and 33 % over LBCDS based CCS.

### (ii) Average global synchronization error

The average global synchronization error per iteration is shown in Fig. 6.16 for CCS on different topological strategies. It is observed that CCS on RGACDBT has, in an average, 82 % less global synchronization error than all node CCS, 66 % less than MCDS based CCS and 55 % less synchronization error than LBCDS based CCS.

### (iii) Average local synchronization error

The local synchronization error is averaged for 50 iterations at individual node as shown in Fig. 6.17. The percentage improvement is calculated by considering the maximum local synchronization error among 50 nodes. It is observed that CCS on RGACDBT has 51 % less local synchronization error than all node CCS, 34 % less than MCDS based CCS, and 19 % less local synchronization error than LBCDS based CCS.

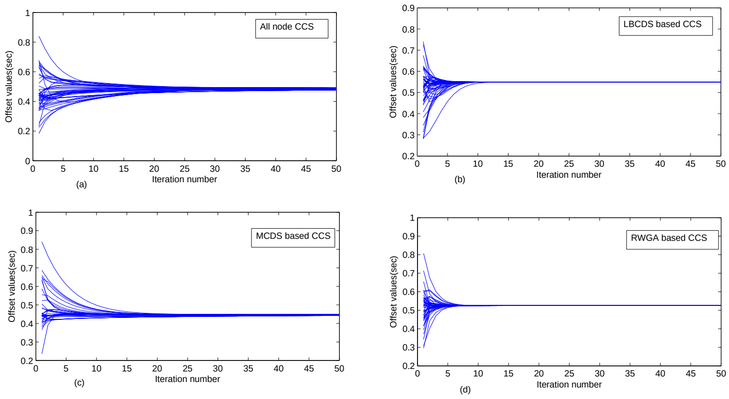


Figure 6.15: Convergence Behavior of CCS algorithm on different Topological Strategies

### (iv) Average number of messages & energy consumption

The average number of messages exchanged and average energy consumption to achieve the acceptable synchronization error is estimated as given in Table 6.6. It is observed that CCS on RGACDBT has 39 % less messages exchanged than all node CCS, 24 % less than LBCDS based CCS and 4 % less messages exchanged than MCDS based CCS.

From Table 6.6, it is also observed that CCS on RGACDBT has 39 % less energy consumption than all node CCS, 26 % less than LBCDS based CCS and 12 % less energy consumption than MCDS based CCS.

Table 6.6: Average number of messages & energy consumption of CCS on different topological strategies for 50 nodes

| **Parameters** | **All node CCS** | **LBCDS based CCS** | **MCDS based CCS** | **RGACDBT based CCS** |
|---|---|---|---|---|
| Avg. number of messages | 467 | 373 | 296 | 283 |
| Avg. energy consumption (in Joule) | 0.0023 | 0.0019 | 0.0016 | 0.0014 |

### Test Algorithm-3: SATS

The SATS algorithm which is proposed in chapter 3 is also tested on the proposed RGACDBT on a network of 50 nodes and the results obtained are discussed below.

### (i) Convergence speed

The convergence speed of SATS in terms of offset convergence is tested for an acceptable synchronization error ($\epsilon$) of 0.0001 sec. as mentioned in Table 6.4. From Fig. 6.18 (a)-(d), it is observed that all node SATS takes around 25 iterations, RGACDBT based SATS takes around 15 iterations, MCDS based SATS takes around 20 iterations, and LBCDS based SATS takes around 19 iterations to converge to the consensus value 0.493. So, the convergence

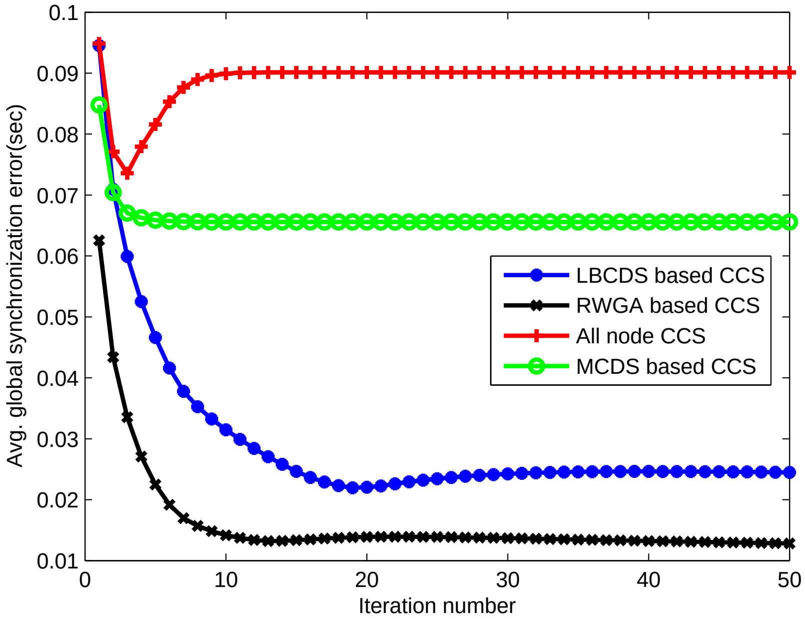


Figure 6.16: Average global synchronization error of CCS algorithm on different Topological Strategies

speed of SATS on RGACDBT has improved by 40 % over all node SATS, 25 % over MCDS based SATS and 21 % over LBCDS based SATS.

### (ii) Average global synchronization error

The average global synchronization error per iteration is shown in Fig. 6.19 for SATS on different topological strategies. It is observed that SATS on RGACDBT has, in an average, 64 % less global synchronization error than all node SATS, 7 % less than MCDS based SATS and 2 % less synchronization error than LBCDS based SATS.

### (iii) Average local synchronization error

The local synchronization error is averaged for 50 iterations at individual node as shown in Fig. 6.20. The percentage improvement is calculated by considering the maximum local synchronization error among 50 nodes. It is observed that SATS on RGACDBT has 69 % less local synchronization error than all node SATS, 21 % less than MCDS based SATS, and 5 % less local synchronization error than LBCDS based SATS.

### (iv) Average number of messages & energy consumption

The average number of messages exchanged and average energy consumption to achieve the acceptable synchronization error is estimated as given in Table 6.7. For computing the average energy consumption, the equation derived in Chapter 3 is followed. It is observed

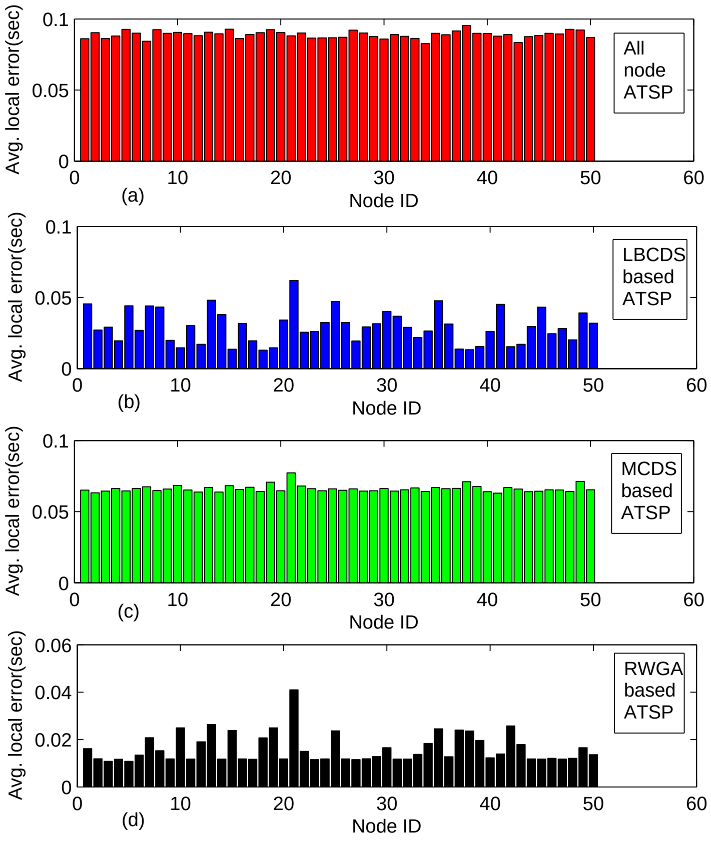


Figure 6.17: Average local synchronization error of CCS algorithm on different Topological Strategies

that SATS on RGACDBT has 53 % less messages exchanged than all node SATS, 40 % less than LBCDS based SATS and 27 % less messages exchanged than MCDS based SATS.

From Table 6.7, it is also observed that SATS on RGACDBT has 43 % less energy consumption than all node SATS, 30 % less than LBCDS based SATS and 18 % less energy consumption than MCDS based SATS.

Table 6.7: Average number of messages & energy consumption of SATS on different topological strategies for 50 nodes

| **Parameters** | **All node SATS** | **LBCDS based SATS** | **MCDS based SATS** | **RGACDBT based SATS** |
|---|---|---|---|---|
| Avg. number of messages | 582 | 458 | 376 | 273 |
| Avg. energy consumption (in Joule) | 0.0048 | 0.0039 | 0.0033 | 0.0027 |

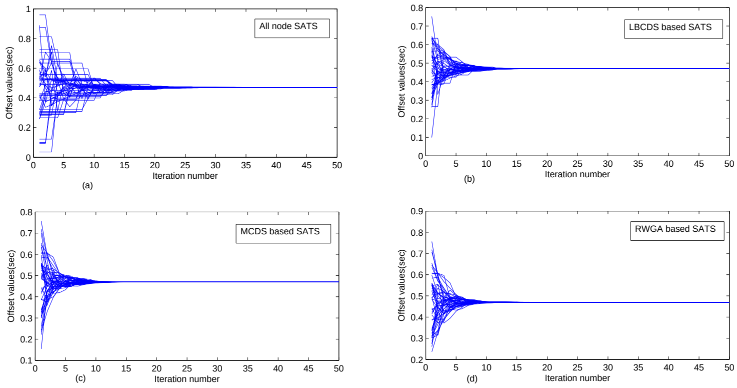


Figure 6.18: Convergence Behavior of SATS algorithm on different Topological Strategies

### 6.6.3 Comparative evaluation of CTS algorithms on RWGA based Strategy

In this Section, the comparative evaluation of ATSP, CCS and SATS algorithms are carried out on the proposed RGACDBT strategy using the same performance metrics. The results obtained are discussed below.

**(i) Convergence speed**

The convergence speed is tested through offset convergence. The offset values for 50 nodes are recorded by considering maximum 50 iterations as shown in Fig. 6.21 (a)-(c). It is observed that ATSP [22] on RGACDBT takes around 20 iterations to achieve convergence, CCS [17] algorithm which uses cumulative weighted averaging (CWA) takes approximately 30 iterations to converge, both with an acceptable synchronization error ($\epsilon$) of 0.0001 sec. Whereas our SATS algorithm on RGACDBT achieves convergence within 15 iterations for the same value of acceptable synchronization error. So, the convergence speed of SATS on RGACDBT has improved by 50 % over CCS on RGACDBT and almost 25 % over ATSP on RGACDBT.

**(ii) Average local synchronization error**

Fig. 6.22 (a)-(c) depicts average local synchronization error of each node for a topology of 50 nodes for 50 iterations. This shows the upper bound of local synchronization error of every node. By considering the maximum local synchronization error, it is observed that SATS on RGACDBT has less local synchronization error, nearly 48 %, as compared to ATSP on RGACDBT and 54 % as compared to CCS on RGACDBT.

**Average global synchronization error**

Fig. 6.23 shows average global (network-wide) synchronization error of 50 nodes after each iteration by considering maximum 50 iterations. It is observed that SATS on RGACDBT

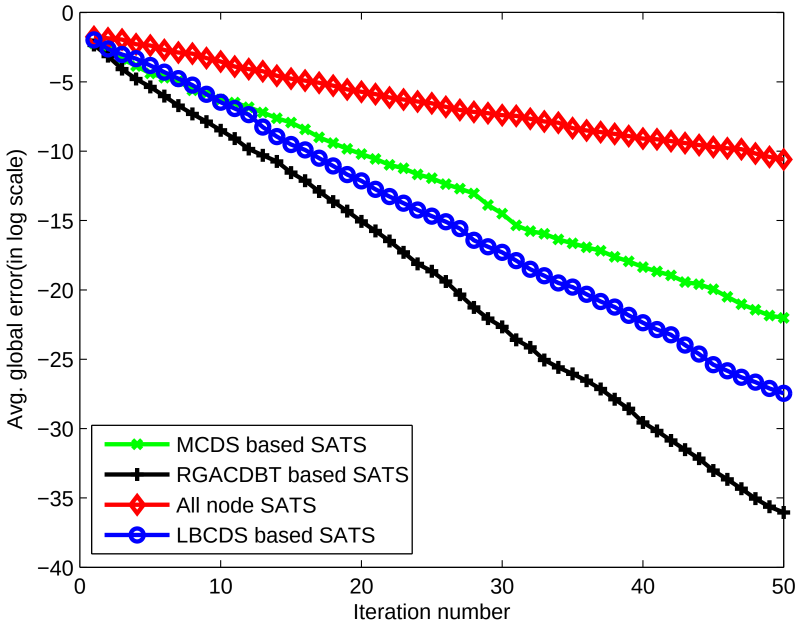


Figure 6.19: Average global synchronization error of SATS algorithm on different Topological Strategies

has less synchronization error, nearly 47%, as compared to ATSP on RGACDBT and 55% as compared to CCS on RGACDBT.

### 6.6.4 Scalability

To study the scalable performance of the proposed strategy, the above test algorithms are also evaluated and compared with other topological strategy by varying the number of nodes from 50-250 as shown in Tables 6.8-6.11. Due to randomness of the topology, the average is calculated for 100 realizations of such random topologies for each number of nodes. The scalability is analyzed by estimating (i) Average mean square synchronization Error, (ii) Average number of iterations, (iii) Average number of messages exchanged, and (iv) Average energy consumption as shown in the Tables 6.8-6.11. From the Tables, it is observed that the proposed strategy's performance remains consistent and optimized as compared to other strategies.

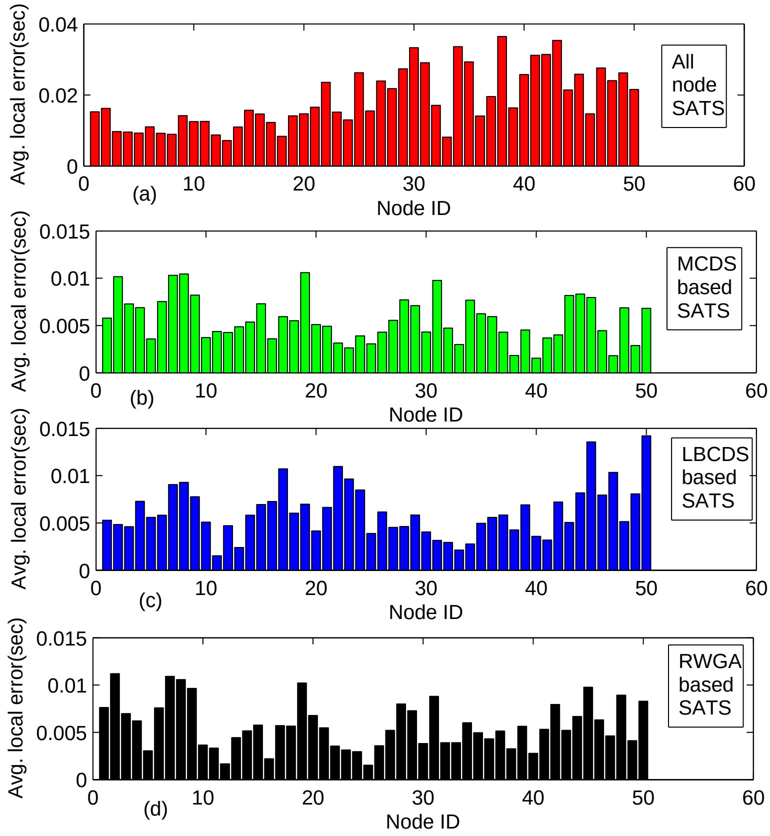


Figure 6.20: Average local synchronization error of SATS algorithm on different Topological Strategies

Table 6.8: Performance comparison of CTS algorithms on sparse network of 100 nodes

| **Network size=100 nodes, Random sparse topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0069 | 25 | 1612 | 0.0236 |
| | MCDS | 0.0067 | 23 | 1135 | 0.0174 |
| | LBCDS | 0.0066 | 21 | 1306 | 0.0213 |
| | RGACDBT | 0.0063 | 19 | 1019 | 0.0158 |
| CCS | All node | 0.0013 | 30 | 518 | 0.0029 |
| | MCDS | 0.0012 | 28 | 356 | 0.0019 |
| | LBCDS | 0.0011 | 25 | 423 | 0.0021 |
| | RGACDBT | 0.0009 | 20 | 321 | 0.0018 |
| SATS | All node | 0.0007 | 20 | 814 | 0.0059 |
| | MCDS | 0.0005 | 19 | 594 | 0.0047 |
| | LBCDS | 0.0006 | 18 | 703 | 0.0052 |
| | RGACDBT | 0.0004 | 17 | 483 | 0.0041 |

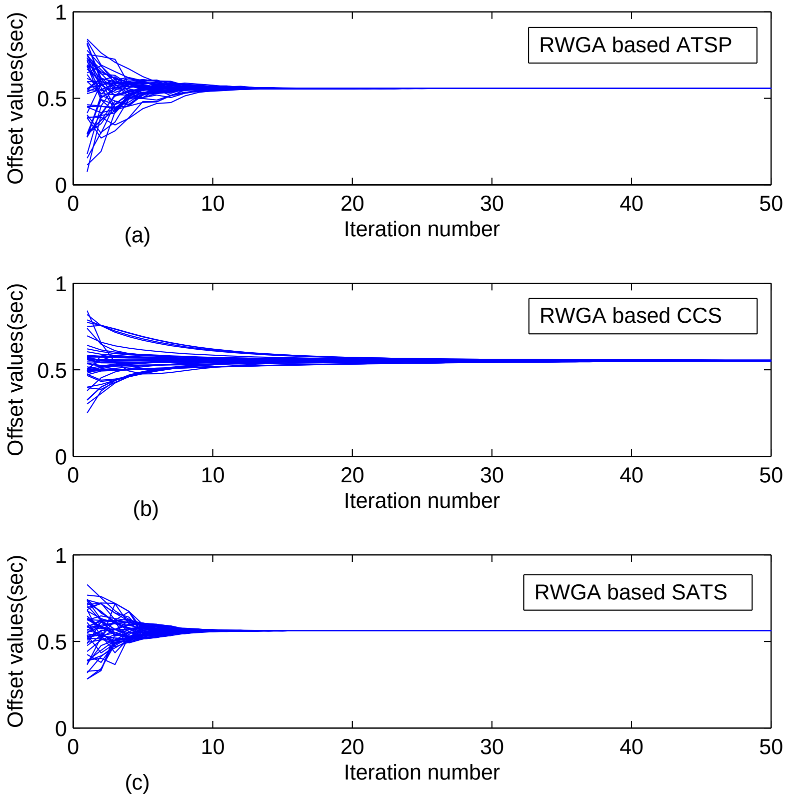


Figure 6.21: Convergence Behavior of CTS algorithms on RGACDBT based Strategy

Table 6.9: Performance comparison of CTS algorithms on sparse network of 150 nodes

| **Network size=150 nodes, Random sparse topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0032 | 26 | 2752 | 0.0391 |
| | MCDS | 0.0029 | 24 | 2117 | 0.0314 |
| | LBCDS | 0.0027 | 23 | 2431 | 0.0353 |
| | RGACDBT | 0.0024 | 22 | 1967 | 0.0289 |
| CCS | All node | 0.0011 | 28 | 673 | 0.0031 |
| | MCDS | 0.0010 | 27 | 426 | 0.0025 |
| | LBCDS | 0.0009 | 26 | 507 | 0.0029 |
| | RGACDBT | 0.0007 | 24 | 367 | 0.0021 |
| SATS | All node | 0.0009 | 19 | 1203 | 0.0064 |
| | MCDS | 0.0006 | 18 | 879 | 0.0057 |
| | LBCDS | 0.0007 | 17 | 1019 | 0.0061 |
| | RGACDBT | 0.0004 | 15 | 692 | 0.0052 |

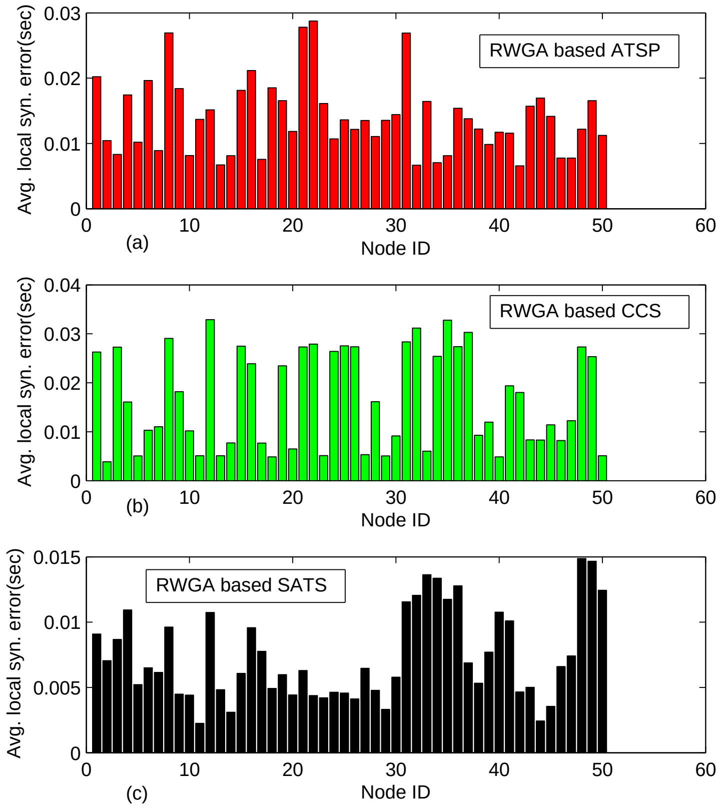


Figure 6.22: Average local synchronization error of CTS algorithms on RGACDBT based Strategy

Table 6.10: Performance comparison of CTS algorithms on sparse network of 200 nodes

| **Network size=200 nodes, Random sparse topology** | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0043 | 28 | 3237 | 0.0432 |
| | MCDS | 0.0042 | 22 | 2705 | 0.0375 |
| | LBCDS | 0.0041 | 25 | 2984 | 0.0417 |
| | RGACDBT | 0.0039 | 21 | 2436 | 0.0334 |
| CCS | All node | 0.0033 | 32 | 952 | 0.0037 |
| | MCDS | 0.0031 | 31 | 608 | 0.0030 |
| | LBCDS | 0.0030 | 30 | 773 | 0.0034 |
| | RGACDBT | 0.0026 | 29 | 523 | 0.0028 |
| SATS | All node | 0.00075 | 20 | 1624 | 0.0069 |
| | MCDS | 0.00071 | 19 | 1206 | 0.0063 |
| | LBCDS | 0.00073 | 18 | 1443 | 0.0066 |
| | RGACDBT | 0.00069 | 17 | 1025 | 0.0061 |

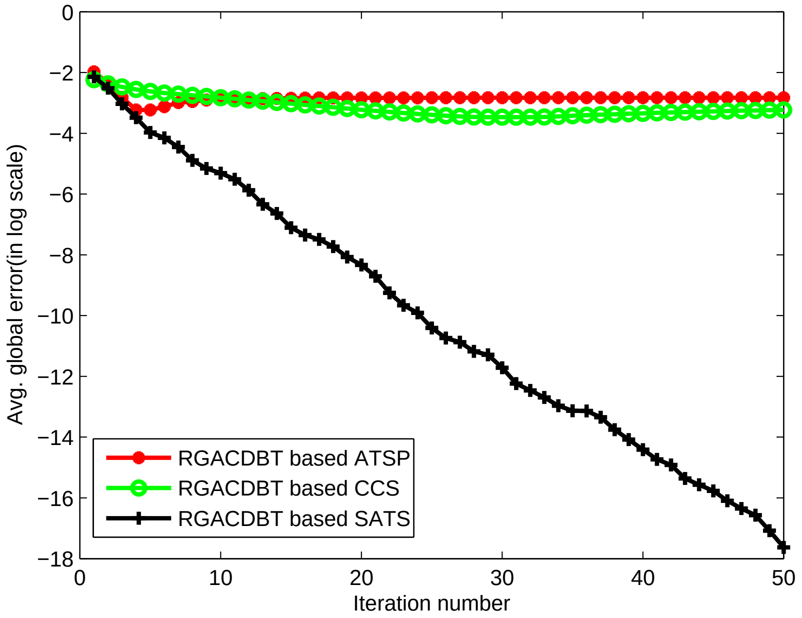


Figure 6.23: Average global synchronization error of CTS algorithms on RGACDBT based Strategy

Table 6.11: Performance comparison of CTS algorithms on sparse network of 250 nodes

| Network size=250 nodes, Random sparse topology | | | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Topological Strategy** | **Avg. MSE** | **Avg. iterations** | **Avg. Messages** | **Avg. Energy (in Joule)** |
| ATSP | All node | 0.0029 | 26 | 3923 | 0.0472 |
| | MCDS | 0.0024 | 21 | 3207 | 0.0397 |
| | LBCDS | 0.0023 | 23 | 3682 | 0.0431 |
| | RGACDBT | 0.0022 | 20 | 3019 | 0.0374 |
| CCS | All node | 0.0017 | 29 | 1173 | 0.0043 |
| | MCDS | 0.0016 | 28 | 754 | 0.0036 |
| | LBCDS | 0.0015 | 27 | 986 | 0.0039 |
| | RGACDBT | 0.0013 | 26 | 593 | 0.0032 |
| SATS | All node | 0.00063 | 22 | 2129 | 0.0073 |
| | MCDS | 0.00058 | 18 | 1721 | 0.0067 |
| | LBCDS | 0.00061 | 17 | 1916 | 0.0070 |
| | RGACDBT | 0.00053 | 16 | 1534 | 0.0064 |

## 6.7 Summary

In this Chapter, a topological optimization strategy is proposed, based on delay balanced topology concept, to accelerate consensus based time synchronization algorithms for wireless sensor network on sparse topology. To handle the trade-off between the MAC and physical delay cost functions, the problem is modeled as a bi-objective topological optimization problem. One of the most effective and simplest optimization method, known as Random Weighted Genetic Algorithm (RWGA) based approach has been applied to get Pareto optimal solutions for the given problem. Extensive simulations have been carried out to show the effectiveness and scalability of the proposed strategy on three recent and state of the art consensus-based synchronization algorithms. Simulation results show that using the proposed strategy, the convergence speed, synchronization error, number of messages exchanged to achieve consensus, and energy consumption have shown notable improvement.

# Chapter 7

# Conclusion & Future Scope

The work presented in this thesis broadly addresses two issues regarding consensus-based time synchronization problem in wireless sensor networks. The first issue is: designing average consensus-based time synchronization algorithms with better convergence speed and synchronization accuracy for dense and sparse networks. The second issue is: improving the performance of consensus-based time synchronization algorithms by topological optimization strategies on dense and sparse topologies. The overall contribution of this thesis is highlighted in this Chapter along with the possible future research directions.

## 7.1 Conclusion

In this thesis, two average consensus based time synchronization algorithms and two topological optimization strategies are proposed as per the aforesaid issues. Firstly, a distributed, average consensus-based time synchronization algorithm (SATS) is proposed for dense, one-hop sensor network. It exploits a novel maximum difference based, selective pair-wise averaging method for faster convergence and better synchronization accuracy. The asymptotic message complexity of the proposed algorithm is proved to be $O\left(n\right)$. Simulation results show that the convergence speed of proposed SATS algorithm is 16 % faster than CCS and 50 % faster than ATSP. The network-wide synchronization error is minimized by 90 % than ATSP and 70 % than CCS. The local synchronization error is also improved by 80 % as compared to ATSP and 82 % as compared to CCS. Due to faster convergence, the average number of message-exchanged has shown significant improvement, nearly 50 % less than ATSP and 10 % less than CCS. The average energy consumption to achieve acceptable synchronization error is also minimized due to the lesser number of message-exchanged. The SATS algorithm has consumed 60 % less energy than ATSP and 20 % less than CCS. The proposed SATS algorithm has shown consistent behavior with the increase in network size and variable network density. So, it is more scalable than ATSP and CCS.

In order to improve the performance of SATS algorithm on sparse, multi-hop network, a multi-hop SATS algorithm is proposed. Since, consensus-based algorithms are greatly affected by topological connectivity, we aimed at increasing the topological connectivity by using multi-hop communication for the underlying sparse network. At the same time, increasing hop count incurs higher end-to-end delay which affects the consensus stability. In order to restrict the delay and select a multi-hop node with maximum relative clock values, a distributed, constraint-based dynamic programming approach is suggested. Using multi-hop communication, a node is selected using the proposed multi-hop SATS algorithm and pairwise averaging is performed between the initiating node and the selected node. The asymptotic message complexity of multi-hop SATS algorithm in a network of $n$ nodes and up to $m$ hop is proved to be $O\left(n(\log n)^m\right)$. A thorough energy consumption analysis is also carried out for the proposed multi-hop SATS algorithm.

Simulation results show that on sparse topology, the proposed multi-hop SATS algorithm with hop count 2 has 16 % faster convergence speed than the proposed SATS (1-hop) algorithm in Chapter 3. Also, the 2-hop SATS algorithm has 33 % faster convergence speed than CCS and 50 % faster than ATSP on sparse topology. The 2-hop SATS algorithm has 95% improvement of average global synchronization error over ATSP, 86% over CCS, and 46% improvement over 1-hop SATS algorithm. The local synchronization error using 2-hop SATS is also optimized, nearly 88% as compared to ATSP, 86% to CCS, and 41% to 1-hop SATS algorithm. As compared to ATSP, the average number of messages exchanged for 2-hop SATS algorithm is 29 % less but, when compared with CCS and 1-hop SATS, it is respectively, 83 % and 5 % more.

The message overhead of 2-hop SATS as compared to CCS is high because CCS follows one-way message passing paradigm for weighted averaging whereas 2-hop SATS follows two-way message passing paradigm to perform pair-wise averaging. So, there exist a trade-off between message exchanges and synchronization accuracy in case of 2-hop SATS and CCS. The 2-hop SATS algorithm has shown better scalability, both in varying network size and varying sparsity factor scenario. Increasing the hop count from 2 to 3 also improves the convergence speed and synchronization error. But, simulation results show that with the further increase in hop count, the end-to-end delay supersedes the threshold delay. So, the optimal behavior of the algorithm lies in the restricted selection of hop count which also ensures consensus stability of multi-hop SATS algorithm. In fact, restricted hop count also makes the algorithm optimal in terms of message complexity.

Further, in order to address the second issue, the topological optimization problem for consensus-based time synchronization algorithm is first proved to be an NP-complete problem, using delay balanced topology concept. Therefore, a GA based topological balancing strategy is proposed to accelerate consensus-based time synchronization algorithms for the dense sensor network. Extensive simulations have been carried out to show the effectiveness and scalability of the proposed strategy on three recent and state-of-the-art consensus-based algorithms. Simulation results show that using the proposed strategy, the number of iterations for consensus convergence, mean square synchronization error, the number of messages exchanged to achieve consensus and energy consumption have been optimized significantly.

Finally, a topological optimization strategy is proposed to accelerate consensus based time synchronization algorithms for the sparse sensor network. In a sparse network, to handle the trade-off between the MAC and physical delay cost functions, the problem is modeled as a bi-objective topological optimization problem. One of the most effective and simplest optimization method, known as Random Weighted Genetic Algorithm (RWGA) based approach has been applied to get Pareto optimal solutions for the given problem. Extensive simulations have been carried out to show the effectiveness and scalability of the proposed strategy on three recent and state of the art consensus-based synchronization algorithms. Simulation results show that using the proposed strategy, the convergence speed to the consensus value, total synchronization error, the number of messages exchanged to achieve consensus and energy consumption have been optimized significantly.

## 7.2 Future Scope

The proposed SATS and multi-hop SATS algorithms presented in this thesis are based on the assumption that the network topology is static in nature. But, in some applications of sensor networks like underwater sensor network, vehicular sensor network, the topology is dynamic due to the mobility of sensor nodes. As a result, the neighborhood of each sensor node changes at different instances of communication. Since, SATS and multi-hop SATS are consensus-based and consensus algorithms are neighbor-dependent, a further study is required to know the effect of dynamic topology on these algorithms.

In the proposed synchronization algorithms, the nodes are assumed to be fault free. But, sensor nodes are generally deployed in an unattended and hostile environment where there is a high chance of getting faulty nodes. So, synchronization in the presence of faulty node is highly challenging, especially in the presence of byzantine faults. So, the proposed algorithms can be extended so that they can operate in the presence of different types of faults without compromising the desired synchronization accuracy. Some recent works highlighted that average consensus-based synchronization algorithms are vulnerable to message manipulation attack. So, the performance of proposed algorithms can be examined under such type of security attacks and methods must be designed how to defend it.

Further, the topological optimization problem for consensus time synchronization algorithms has proven to be NP-complete. So, different heuristic and meta-heuristic approaches can be applied to study the optimal behavior of the algorithms and their impact on the consensus algorithms’ performances.

# Bibliography

[1] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, ''Clock synchronization for wireless sensor networks: a survey,'' *Ad Hoc Networks*, vol. 3, no. 3, pp. 281 – 323, 2005.

[2] D. L. Mills, ''Internet time synchronization: the network time protocol,'' *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct 1991.

[3] M. D. Lemmon, J. Ganguly, and L. Xia, ''Model-based clock synchronization in networks with drifting clocks,'' in *Proc. of Pacific Rim International Symposium on Dependable Computing*, 2000, pp. 177–184.

[4] K. lae Noh, E. Serpedin, and K. Qaraqe, ''A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization,'' *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3318–3322, Sept 2008.

[5] W. Sun, ''On clock synchronization in wireless networks using parameter estimation and consensus techniques,'' Ph.D. dissertation, Chalmers University of Technology, 2013.

[6] P. Kartaschoff, ''Synchronization in digital communications networks,'' *Proceedings of the IEEE*, vol. 79, no. 7, pp. 1019–1028, Jul 1991.

[7] I. K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y. C. Wu, ''Clock synchronization in wireless sensor networks: An overview,'' *Sensors*, vol. 9, no. 1, p. 56, 2009. [Online]. Available: http://www.mdpi.com/1424-8220/9/1/56

[8] S. Youn, ''A comparison of clock synchronization in wireless sensor networks,'' *International Journal of Distributed Sensor Networks*, vol. 2013, no. 1, p. 10, 2013.

[9] A. R. Swain and R. Hansdah, ''A model for the classification and survey of clock synchronization protocols in wsns,'' *Ad Hoc Networks*, vol. 27, pp. 219 – 241, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870514002960

[10] F. Sivrikaya and B. Yener, ''Time synchronization in sensor networks: a survey,'' *IEEE Network*, vol. 18, no. 4, pp. 45–50, July 2004.

[11] ''Microcontroller clock–rc oscillator, crystal or resonator,'' http://www.hosonic.com/quartz crystal clock oscillator.htm.

[12] J. Lichtenauer, J. Shen, M. Valstar, and M. Pantic, ''Cost-effective solution to synchronised audio-visual data capture using multiple sensors,'' *Image and Vision Computing*, vol. 29, no. 10, pp. 666 – 680, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885611000618

[13] ''Tmotesky data sheet,'' www.eecs.harvard.edu/ konrad/projects/tmote-sky-datasheet.pdf.

[14] Zheng and A. Jamalipour, *Wireless sensor network: A Networking Perspective*. Willey Publication, 2009.

[15] N. M. Freris, S. R. Graham, and P. R. Kumar, ''Fundamental limits on synchronizing clocks over networks,'' *IEEE Transactions On Automatic Control*, vol. 56, no. 6, pp. 1352–1364, June 2011.

[16] D. Djenouri, ''R4sync: Relative referenceless receiver/receiver time synchronization in wireless sensor networks,'' *IEEE Signal Processing Letters*, vol. 19, no. 4, pp. 175–178, April 2012.

[17] M. K. Maggs, S. G. Keefe, and D. V. Thiel, ‘‘Consensus clock synchronization for wireless sensor networks,’’ *IEEE Sensors Journal*, vol. 12, no. 6, pp. 2269–2277, June 2012.

[18] L. Schenato and F. Fiorentin, ‘‘Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks,’’ *Automatica*, vol. 47, pp. 1878–1886, July 2011.

[19] J. He, P. Cheng, L. Shi, and J. Chen, ‘‘Time synchronization in wsns: A maximum value based consensus approach,’’ in *Proc. of 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 12-15 Dec 2011, pp. 7882–7887.

[20] J. Elson, L. Girod, and D. Estrin, ‘‘Fine-grained network time synchronization using reference broadcasts,’’ in *Proc. of 5th USENIX Symposium on Operating System Design and Implementation (OSDI’02)*, Dec 2002, pp. 147–163.

[21] S. Ganeriwal, R. Kumar, and M. Srivastava, ‘‘Timing-sync protocol for sensor networks,’’ in *Proc. of 1st ACM conference on embedded networked sensor systems*, 2003, pp. 138–149.

[22] J.Wu, L.Jiao, and R.Ding, ‘‘Average time synchronization in wireless sensor networks using pairwise messages,’’ *Computer Communications*, vol. 35, pp. 221–233, Jan 2012.

[23] Q. Li and D. Rus, ‘‘Global clock synchronization in sensor networks,’’ *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214– 226, Feb. 2006.

[24] L. Schenato and G. Gamba, ‘‘A distributed consensus protocol for clock synchronization in wireless sensor network,’’ in *Proc. of 46th IEEE Conference on Decision and Control*, 12-14 Dec. 2007, pp. 2289–2294.

[25] J. Chen, M. Yu, L. H. Dou, and M. G. Gan, ‘‘A fast averaging synchronization algorithm for clock oscillators in nonlinear dynamical network with arbitrary time-delays,’’ *Acta Automatica Sinica*, vol. 36, no. 6, pp. 873–880, June 2010.

[26] M. Franceschelli, A. Giua, and C. Seatzu, ‘‘Distributed averaging in sensor networks based on broadcast gossip algorithms,’’ *IEEE Sensors Journal*, vol. 11, no. 3, pp. 808–817, March 2011.

[27] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, ‘‘Broadcast gossip algorithms for consensus,’’ *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, July 2009.

[28] R. O. Saber and R. M. Murray, ‘‘Consensus problems in networks of agents with switching topology and time-delays,’’ *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sept 2004.

[29] C. Asensio-Marco and B. Beferull-Lozano, ‘‘Network topology optimization for accelerating consensus algorithms under power constraints,’’ in *Proc. of IEEE 8th International Conference on Distributed Computing in Sensor Systems*, May 2012, pp. 224–229.

[30] *Prowler Simulator*, www.isis.vanderbilt.edu/projects/nest/prowler.

[31] M. Singhal and N. Shivaratri, *Advanced concepts in Opeating System*. TMH Publication, 2007.

[32] L. Lamport, ‘‘Time, clocks, and the ordering of events in a distributed system,’’ *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978. [Online]. Available: http://doi.acm.org/10.1145/359545.359563

[33] C. Fidge, ‘‘Logical time in distributed computing systems,’’ *Computer*, vol. 24, no. 8, pp. 28–33, Aug 1991.

[34] F. Cristian, ‘‘Probabilistic clock synchronization,’’ *Distributed Computing*, vol. 3, no. 3, pp. 146–158, 1989. [Online]. Available: http://dx.doi.org/10.1007/BF01784024

[35] K. Arvind, ‘‘Probabilistic clock synchronization in distributed systems,’’ *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 5, pp. 474–487, 1994.

[36] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, ‘‘Continuous clock synchronization in wireless real-time applications,’’ in *Proc. of 19th IEEE Symposium on Reliable Distributed Systems*, 2000, pp. 125–132.

[37] S.Ping, ‘‘Delay measurement time synchronization for wireless sensor networks,’’ 2003.

[38] K. Römer, ‘‘Time synchronization in ad hoc networks,’’ in *Proc. of 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, ser. MobiHoc ’01. ACM, 2001, pp. 173–182. [Online]. Available: http://doi.acm.org/10.1145/501436.501440

[39] S. PalChaudhuri, A. Saha, and D. B. Johnson, ‘‘Probabilistic clock synchronization service in sensor networks,’’ *IEEE Transactions on Networking*, vol. 2, no. 2, pp. 177–189, 2003.

[40] W. Su and I. F. Akyildiz, ‘‘Time-diffusion synchronization protocol for wireless sensor networks,’’ *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 384–397, Apr. 2005. [Online]. Available: http://dx.doi.org/10.1109/TNET.2004.842228

[41] M. L. Sichitiu and C. Veerarittiphan, ‘‘Simple, accurate time synchronization for wireless sensor networks,’’ in *Proc. of Wireless Communications and Networking, WCNC*, vol. 2, March 2003, pp. 1266–1273.

[42] K. L. Noh, Y. C. Wu, K. Qaraqe, and B. Suter, ‘‘Extension of pairwise broadcasting clock synchronization for multi-cluster sensor networks,’’ *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 71:1–71:10, Jan. 2008.

[43] K. Cheng, K. S. Lui, Y. C. Wu, and V. Tam, ‘‘A distributed multihop time synchronization protocol for wireless sensor networks using pairwise broadcast synchronization,’’ *IEEE Transaction on wireless communications*, vol. 8, no. 4, pp. 1764–1772, April 2009.

[44] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, ‘‘The flooding time synchronization protocol,’’ in *Proc. of 2nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys ’04, 2004, pp. 39–49. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031501

[45] G. Qi, P. Song, K. Li, and C. Chen, ‘‘One improved time synchronization based on ant colony optimization and tpsn mechanism,’’ in *Proc. of International Conference on Networking, Sensing and Control (ICNSC), 2010*, April 2010, pp. 223–227.

[46] L. Liu, Y. Xiao, and J. Zhang, ‘‘A bio-inspired time synchronization algorithm for wireless sensor networks,’’ in *Proc. of 2nd International Conference on Computer Engineering and Technology (ICCET),*, vol. 4, April 2010, pp. V4–306–V4–311.

[47] Y. P. Tian, ‘‘Lsts: A new time synchronization protocol for networks with random communication delays,’’ in *Proc. of 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 7404–7409.

[48] J. S. Kim, J. Lee, E. Serpedin, and K. Qaraqe, ‘‘A robust clock synchronization algorithm for wireless sensor networks,’’ in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3512–3515.

[49] J. Li and A. Nehorai, ‘‘Joint sequential target estimation and clock synchronization in wireless sensor networks,’’ *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 74–88, June 2015.

[50] Q. Liu, X. Liu, J. L. Zhou, G. Zhou, G. Jin, Q. Sun, and M. Xi, ‘‘Adasynch: A general adaptive clock synchronization scheme based on kalman filter for wsns,’’ *Wireless Personal Communications*, vol. 63, no. 1, pp. 217–239, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11277-010-0116-3

[51] X. Li, M. Xi, Y. Cao, and J. Yuan, ‘‘A general clock synchronization method based on kalman filter model in wireless sensor networks,’’ in *Proc. of 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, April 2012, pp. 2277–2280.

[52] B. R. Hamilton, X. Ma, Q. Zhao, and J. Xu, ‘‘Aces: Adaptive clock estimation and synchronization using kalman filtering,’’ in *Proc. of 14th ACM International Conference on Mobile Computing and Networking*, ser. MobiCom ’08, 2008, pp. 152–162. [Online]. Available: http://doi.acm.org/10.1145/1409944.1409963

[53] L. Ma, H. Zhu, G. Nallamothu, B. Ryu, and Z. Zhang, ‘‘Impact of linear regression on time synchronization accuracy and energy consumption for wireless sensor networks,’’ in *Proc. of IEEE Military Communications Conference*, Nov 2008, pp. 1–7.

[54] K. Huang and D. Lee, ‘‘Consensus-based peer-to-peer control architecture for multiuser haptic interaction over the internet,’’ *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 417–431, April 2013.

[55] Z. Wang, M. M. Hayat, M. Rahnamay-Naeini, Y. Mostofi, and J. E. Pezoa, ‘‘Consensus-based estimation protocol for decentralized dynamic load balancing over partially connected networks,’’ in *Proc. of 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 4572–4579.

[56] J. Zhou, *First International Conference on Complex Science, Shanghai, China, February 23-25, 2009*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2009. [Online]. Available: https://books.google.co.in/books?id=9ScGkMmyp8QC

[57] J. Wu, L. Zhang, Y. Bai, and Y. Sun, ‘‘Cluster-based consensus time synchronization for wireless sensor networks,’’ *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1404–1413, March 2015.

[58] D. Djenouri, N. Merabtine, F. Z. Mekahlia, and M. Doudou, ‘‘Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks,’’ *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2329–2344, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2013.06.001

[59] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, ‘‘Time synchronization in wsns: A maximum-value-based consensus approach,’’ *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 660–675, March 2014.

[60] J. He, J. Chen, P. Cheng, and X. Cao, ‘‘Secure time synchronization in wireless sensor networks: A maximum consensus-based approach,’’ *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1055–1065, April 2014.

[61] J. He, P. Cheng, J. Chen, L. Shi, and R. Lu, ‘‘Time synchronization for random mobile sensor networks,’’ *IEEE Transactions on Vehicular Technology*, vol. 63, no. 8, pp. 3935–3946, Oct 2014.

[62] J. He, P. Cheng, L. Shi, and J. Chen, ‘‘Sats: Secure average-consensus-based time synchronization in wireless sensor networks,’’ *IEEE Transactions on Signal Processing*, vol. 61, no. 24, pp. 6387–6400, Dec 2013.

[63] B. J. Choi, H. Liang, X. Shen, and W. Zhuang, ‘‘Dcs: Distributed asynchronous clock synchronization in delay tolerant networkd,’’ *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 491–504, March 2012.

[64] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, ‘‘Models and solutions for radio irregularity in wireless sensor networks,’’ *ACM Trans. Sen. Netw.*, vol. 2, no. 2, pp. 221–262, May 2006. [Online]. Available: http://doi.acm.org/10.1145/1149283.1149287

[65] D. Ustebay, B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, ‘‘Greedy gossip with eavesdropping,’’ *IEEE Transactions on Signal Processing*, vol. 58, no. 7, pp. 3765–3776, July 2010.

[66] G. Xiong and S. Kishore, ‘‘Analysis of distributed consensus time synchronization with gaussian delay over wireless sensor networks,’’ *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, 2009.

[67] L. Yong and G. Lixin, ‘‘On the placement of clock reference nodes for time synchronization in sensor networks,’’ rio.ecs.umass.edu, Tech. Rep.

[68] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[69] J. Liu, A. S. Morse, B. D. O. Anderson, and C. Yu, ‘‘Contractions for consensus processes,’’ in *Proc. of 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Orlando, FL, USA, Dec. 12-15 2011, pp. 1974–1979.

[70] A. Nedic and D. Bertsekas, ‘‘Incremental sub-gradient methods for non differentiable optimization,’’ *SIAM Journal on Optimization.*, vol. 12, no. 1, pp. 109–138, 2001.

[71] *Matrix Reference Manual*, http://www.ee.ic.ac.uk/hp/staff/dmb/matrix.

[72] G. Chartrand and O. R. Oellermann, *Applied and algorithmic graph theory*. McGraw-Hill, 1993.

[73] J. Wang, S. Zhang, D. Gao, and Y. Wang, ‘‘Two-hop time synchronization protocol for sensor networks,’’ *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–10, 2014. [Online]. Available: http://dx.doi.org/10.1186/1687-1499-2014-39

[74] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press and McGraw-Hill, 2009.

[75] J. K. Antonio, G. M. Huang, and W. K. Tsai, ‘‘A fast distributed shortest path algorithm for a class of hierarchically clustered data networks,’’ *IEEE Transactions on Computers*, vol. 41, no. 6, pp. 710–724, Jun 1992.

[76] S. Manfredi, ‘‘A theoretical analysis of multi-hop consensus algorithms for wireless networks: Trade off among reliability, responsiveness and delay tolerance,’’ *Ad Hoc Networks*, vol. 13, Part A, pp. 234 – 244, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870511001004

[77] T. Ratnaparkhe, S. Natekar, S. Chandan, and V. P. Sadaphal, ‘‘Selection of time synchronizing nodes in wireless sensor network,’’ in *Proc. of Communication Systems and Networks (COMSNETS)*, Jan 2010, pp. 1–8.

[78] Y. Zou and K. Chakrabarty, ‘‘A distributed coverage and connectivity-centric technique for selecting active nodes in wireless sensor networks,’’ *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 978–991, Aug 2005.

[79] H. Zhou, T. Liang, C. Xu, and J. Xie, ‘‘Multiobjective coverage control strategy for energy-efficient wireless sensor networks,’’ *International Journal of Distributed Sensor Networks*, vol. 2012, pp. 1–10, 2012.

[80] Y. K. Park, M. G. Lee, K. K. Jung, J. J. Yoo, S. H. Lee, and H. S. Kim, ‘‘Optimum sensor nodes deployment using fuzzy c-means algorithm,’’ in *Proc. of International Symposium on Computer Science and Society (ISCCS)*, July 2011, pp. 389–392.

[81] J. Ryu, J. Yu, E. Noel, and K. W. Tang, ‘‘Borel cayley graph-based topology control for consensus protocol in wireless sensor networks,’’ *ISRN Sensor Networks*, vol. 2013, pp. 1–15, 2013.

[82] S. Chu, P. Wei, X. Zhong, X. Wang, and Y. Zhou, ‘‘Deployment of a connected reinforced backbone network with a limited number of backbone nodes,’’ *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1188–1200, June 2013.

[83] J. He, S. Ji, Y. Pan, and Z. Cai, ‘‘Approximation algorithms for load-balanced virtual backbone construction in wireless sensor networks,’’ *Theoretical Computer Science*, vol. 507, no. 0, pp. 2–16, 2013.

[84] J. He, S. Ji, M. Yan, Y. Pan, and Y. Li, ‘‘Genetic-algorithm-based construction of load-balanced cdss in wireless sensor networks,’’ in *Proc. of MILITARY COMMUNICATIONS CONFERENCE*, Nov 2011, pp. 667–672.

[85] D. Manohari and G. S. A. Mala, ‘‘An evolutionary algorithmic approach to construct connected dominating set in manets,’’ in *Proc. of International Conference on Software Engineering and Mobile Application Modelling and Development (ICSEMA)*, Dec 2012, pp. 1–6.

[86] F. Nawab, K. Jamshaid, B. Shihada, and P. H. Ho, ''Fair packet scheduling in wireless mesh networks,'' *Ad Hoc Networks*, vol. 13, Part B, no. 0, pp. 414 – 427, 2014.

[87] A. Konak, D. W. Coit, and A. E. Smith, ''Multi-objective optimization using genetic algorithms: A tutorial,'' *Reliability Engineering and System Safety*, vol. 91, no. 9, pp. 992 – 1007, 2006, special Issue - Genetic Algorithms and Reliability.

[88] T. Murata and H. Ishibuchi, ''Moga: multi-objective genetic algorithms,'' in *Proc. of IEEE International Conference on Evolutionary Computation*, vol. 1, Nov 1995, pp. 289–294.

[89] J. Yu, N. Wang, G. Wang, and D. Yu, ''Connected dominating sets in wireless ad hoc and sensor networks – a comprehensive survey,'' *Computer Communications*, vol. 36, no. 2, pp. 121 – 134, 2013.

[90] A. Potluri and A. Singh, ''Metaheuristic algorithms for computing capacitated dominating set with uniform and variable capacities,'' *Swarm and Evolutionary Computation*, vol. 13, pp. 22 – 33, 2013.

[91] A. Mahapatro and P. M. Khilar, ''Detection and diagnosis of node failure in wireless sensor networks: A multiobjective optimization approach,'' *Swarm and Evolutionary Computation*, vol. 13, pp. 74 – 84, 2013.

[92] S. H. Lee and L. Choi, ''Chaining clock synchronization: an energy-efficient clock synchronization scheme for wireless sensor networks,'' in *Proc. of 10th International Symposium on Pervasive Systems, Algorithms, and Networks(ISPAN'09)*, 2009, pp. 171–177.

[93] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, ''Continuous clock synchronization in wireless real-time application,'' in *Proc. of 19th IEEE Symposium on Reliable Distributed Systems (SRDS-00)*, Oct. 2000, pp. 125–133.

[94] S. Ahmed, F. Xiao, and T. Chen, ''Asynchronous consensus-based time synchronisation in wireless sensor networks using unreliable communication links,'' *IET Control Theory Applications*, vol. 8, no. 12, pp. 1083–1090, August 2014.

[95] C. Li, Y. Wang, and M. Hurfin, ''Clock synchronization in mobile ad hoc networks based on an iterative approximate byzantine consensus protocol,'' in *Proc. of IEEE 28th International Conference on Advanced Information Networking and Applications*, May 2014, pp. 210–217.

[96] H. Cheng, S. Yang, and J. Cao, ''Dynamic genetic algorithms for the dynamic load balanced clustering problem in mobile ad hoc networks,'' *Expert Systems with Applications*, vol. 40, pp. 1381 – 1392, 2013.

[97] S. Manfredi, ''Design of a multi-hop dynamic consensus algorithm over wireless sensor networks,'' *Control Engineering Practice*, vol. 21, no. 4, pp. 381 – 394, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066112002559

[98] N. Panigrahi and P. M. Khilar, ''Optimal consensus-based clock synchronisation algorithm in wireless sensor network by selective averaging,'' *IET Wireless Sensor Systems*, vol. 5, no. 3, pp. 166–174, 2015.

[99] ——, ''Optimal topological balancing strategy for performance optimisation of consensus-based clock synchronisation protocols in wireless sensor networks: a genetic algorithm-based approach,'' *IET Wireless Sensor Systems*, vol. 4, no. 4, pp. 213–222, 2014.

[100] C. Kahraman, *Computational Intelligence Systems in Industrial Engineering: With Recent Theory and Applications*, ser. Atlantis Computational Intelligence Systems. Atlantis Press, 2012. [Online]. Available: https://books.google.co.in/books?id=RZZGIp7VGCIC

[101] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, ''Firefly-inspired sensor network synchronicity with realistic radio effects,'' in *Proc. of 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '05, 2005, pp. 142–153. [Online]. Available: http://doi.acm.org/10.1145/1098918.1098934

[102] P. Sommer and R. Wattenhofer, ''Gradient clock synchronization in wireless sensor networks,'' in *Proc. of International Conference on Information Processing in Sensor Networks*, April 2009, pp. 37–48.

[103] F. Fagnani and S. Zampieri, ‘‘Randomized consensus algorithms over large scale networks,’’ *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 634–649, May 2008.

[104] A. Marco, R. Casas, J. L. S. Ramos, V. Coarasa, A. Asensio, and M. S. Obaidat, ‘‘Synchronization of multihop wireless sensor networks at the application layer,’’ *IEEE Wireless Communications*, vol. 18, no. 1, pp. 82–88, February 2011.

[105] J. Liu, Z. Zhou, Z. Peng, J. H. Cui, M. Zuba, and L. Fiondella, ‘‘Mobi-sync: Efficient time synchronization for mobile underwater sensor networks,’’ *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 2, pp. 406–416, Feb 2013.

[106] L. Paladina, A. Biundo, M. Scarpa, and A. Puliafito, ‘‘Self organizing maps for synchronization in wireless sensor networks,’’ in *Proc. of Conference on New Technologies, Mobility and Security*, Nov 2008, pp. 1–6.

[107] I. Shames and A. N. Bishop, ‘‘Relative clock synchronization in wireless networks,’’ *IEEE Communications Letters*, vol. 14, no. 4, pp. 348–350, April 2010.

[108] L. M. He, ‘‘Time synchronization based on spanning tree for wireless sensor networks,’’ in *Proc. of 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Oct 2008, pp. 1–4.

[109] B. M. Sadler, ‘‘Local and broadcast clock synchronization in a sensor node,’’ *IEEE Signal Processing Letters*, vol. 13, no. 1, pp. 9–12, Jan 2006.

[110] Y. Wang, J. Huang, L. Yang, and Y. Xue, ‘‘Toa-based joint synchronization and source localization with random errors in sensor positions and sensor clock biases,’’ *Ad Hoc Netw.*, vol. 27, no. C, pp. 99–111, Apr. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2014.12.001

[111] K. L. Noh, Q. M. Chaudhari, E. Serpedin, and B. W. Suter, ‘‘Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks,’’ *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 766–777, April 2007.

[112] M. Leng and Y. C. Wu, ‘‘On clock synchronization algorithms for wireless sensor networks under unknown delay,’’ *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 182–190, Jan 2010.

[113] P.-H. Huang, M. Desai, X. Qiu, and B. Krishnamachari, ‘‘On the multihop performance of synchronization mechanisms in high propagation delay networks,’’ *IEEE Transactions on Computers*, vol. 58, no. 5, pp. 577–590, 2009.

[114] A. R. Swain and R. C. Hansdah, ‘‘An energy efficient and fault-tolerant clock synchronization protocol for wireless sensor networks,’’ in *Proc. of 2nd International Conference on COMmunication Systems and NETworks (COMSNETS 2010)*, Jan 2010, pp. 1–10.

[115] N. Panigrahi and P. M. Khilar, ‘‘An evolutionary based topological optimization strategy for consensus based clock synchronization protocols in wireless sensor network,’’ *Swarm and Evolutionary Computation*, vol. 22, pp. 66 – 85, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650215000164

[116] M. Akhlaq and T. R. Sheltami, ‘‘Rtsp: An accurate and energy-efficient protocol for clock synchronization in wsns,’’ *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 3, pp. 578–589, March 2013.

[117] S. Ganguly, N. Sahoo, and D. Das, ‘‘Multi-objective planning of electrical distribution systems using dynamic programming,’’ *International Journal of Electrical Power and Energy Systems*, vol. 46, pp. 65 – 78, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061512005935

[118] J. Wu, S. Ren, Y. Jiang, and L. Song, ‘‘Qos-aware multihop routing in wireless sensor networks with power control using demodulation-and-forward protocol,’’ *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–9, 2012.

[119] Y. Wang, M. Z. Bocus, and J. P. Coon, ‘‘Dynamic programming for route selection in multihop fixed gain amplify-and-forward relay networks,’’ *IEEE Communications Letters*, vol. 17, no. 5, pp. 932–935, May 2013.

[120] J. Zhang, X. Jia, and G. Xing, ‘‘Real-time data aggregation in contention-based wireless sensor networks,’’ *ACM Trans. Sen. Netw.*, vol. 7, no. 1, pp. 2:1–2:25, Aug. 2010.

[121] J. He, H. Li, J. Chen, and P. Cheng, ‘‘Study of consensus-based time synchronization in wireless sensor networks,’’ *ISA Transactions*, vol. 53, no. 2, pp. 347 – 357, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019057813001900

[122] F. Wang, X. Wu, Y. Pang, C. Yu, Y. Hu, and X. Liu, ‘‘A time synchronization method of wireless sensor networks based on the simulated annealing algorithm,’’ in *Proc. of 26th Chinese Control and Decision Conference (CCDC)*, May 2014, pp. 870–875.

# Dissemination

**Journal Articles (Published)**

1. N. Panigrahi and P. M. Khilar, Optimal consensus-based clock synchronization algorithm in wireless sensor network by selective averaging, IET Wireless Sensor Systems, vol. 5, no. 3, pp. 166-174, 2015, doi: 10.1049/iet-wss.2013.0102 **(Chapter 3)**
2. N. Panigrahi and P. M. Khilar, Optimal topological balancing strategy for performance optimization of consensus-based clock synchronization protocols in wireless sensor networks: a genetic algorithm-based approach, IET Wireless Sensor Systems, vol. 4, no. 4, pp. 213-222, 2014, doi: 10.1049/iet-wss.2014.0063 **(Chapter 5)**
3. N. Panigrahi and P. M. Khilar, An evolutionary based topological optimization strategy for consensus based clock synchronization protocols in wireless sensor network, Swarm and Evolutionary Computation, Elsevier, Vol. 22, June 2015, pp. 66-85, ISSN 2210-6502, http://dx.doi.org/10.1016/j.swevo.2015.02.001 **(Chapter 6)**

**Journal Articles (Communicated)**

1. N. Panigrahi and P. M. Khilar, Multi-hop Consensus Time Synchronization Algorithm for sparse Wireless Sensor Network: A Distributed Constraint-based Dynamic Programming Approach, Journal of Computer & System science, Elsevier **(Chapter 4)**
2. N. Panigrahi and P. M. Khilar, Energy Consumption Analysis of Consensus Time Synchronization algorithms for Wireless Sensor Networks, Journal of Wireless Communication, SISpress.

# Resume

## Niranjan Panigrahi

Department of CSE
NIT, Rourkela, 769008, Odisha, India
Ph No: +91-9861093074
E-mail: er.niranjan@gmail.com
niranjan.cse@pmec.ac.in
Date of Birth: August 29, 1980
Nationality: Indian

## Current Position

*Assistant Professor*, Parala Maharaja Engineering College, Berhampur, Odisha, India

## Qualification

- Doctor of Philosophy (Continuing), NIT, Rourkela, Odisha, India
- Master of Technology, NIT, Rourkela, Odisha, India
- Bachelor of Engineering, ITER, Bhubaneswar, Odisha, India

## Publications

- 03 journal articles

## Permanent Address

H.No-JA-57, Jagda,
Rourkela, Sundergarh,
Odisha, 769014, India