ENERGY-EFFICIENT TIME SYNCHRONIZATION FOR WIRELESS SENSOR NETWORKS (WSN)

By

Anam Ghaffar

Sana Jan

Hassan Azim Qureshi

Submitted to the Faculty of Computer Science, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of BE Degree in Computer Software Engineering

April 2007

ABSTRACT

ENERGY-EFFICIENT TIME SYNCHRONIZATION FOR

FOR WIRELESS SENSOR NETWORKS (WSN)

As wireless sensor networks continue to grow, so does the need for effective power management. Sensor nodes are small battery powered devices with scarce energy, which is usually impossible to replenish. Network configuration and management algorithms, which consume less energy, are still a challenge in WSN. Our goal is to propose improvements in existing time synchronization algorithms and extend them to meet the needs of WSN. In this work, we review the existing time synchronization algorithms such as Network Time Protocol (NTP), Reference Broadcast Synchronization (RBS) and Timing sync Protocol for Sensor Networks (TPSN).

Time synchronization schemes developed for traditional networks such as NTP are ill-suited for WSN due to its complexity and excessive energy consumption. We investigate the redesign of NTP for WSN environment by introducing the concept of energy-awareness. We also outline the energy consumption of RBS and TPSN and propose low power algorithms, which minimize the number of messages exchanged needed to synchronize the network.

To evaluate the appropriateness of our approach, we analyze the evaluated performance of our proposed techniques against existing ones.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of any other award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents and family, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible.

To our supervisors, without whose unstinting cooperation and guidance, achieving this much level of knowledge would have been impossible for us.

ACKNOWLEDGMENTS

All acclamation to Almighty Allah Who has empowered and enabled us to accomplish this task successfully.

We gratefully recognize the continuous supervision and motivation provided to us by our project supervisor Miss Hajra Batool. Acknowledgements are due to Maj Athar, Maj Rizwan, MCS faculty members and administration for their ever-extended moral and technical support.

We are extremely thankful to our external supervisor Dr. Arshad Ali, Director NIIT, and co-advisors Saad Liaquat Kiani, Nauman Qureshi, for their guidance and instructive supervision throughout the course of project even with their highly busy schedule within and outside the country.

We would also like to thank Jeremy Elson and Lewis Girod professors at University of California, Los Angeles (UCLA) for discussing and acknowledging theory of our proposed solutions.

We deeply treasure the unparallel support and forbearance that we received from our friends for their critical reviews and useful suggestions that helped us in completion of this project. We are also deeply obliged to our families for their never ending patience and support for our mental peace and to our parents for the strength that they gave us through their prayers.

TABLE OF CONTENTS

Titles *Page*

LIST OF FIGURES ..... **IX**
LIST OF TABLES ..... **X**
KEY TO SYMBOLS/ABBREVIATION ..... **XI**
1 Wireless Sensor Networks an Overview **1**
1.1 Sensor Nodes 1
1.2 Wireless Sensor Networks 2
1.3 WSN Features 3
1.3.1 Deployment 4
1.3.2 Unattended Operation 4
1.3.3 Restricted resources 4
1.3.4 Network dynamics 5
1.3.5 Time Synchronization 5
1.4 Design Factors of WSN 5
1.5 Applications 6
1.5.1 Military Applications 6
1.5.1.1 Monitoring Friendly Forces, Equipment and Ammunition 7
1.5.1.2 Battlefield surveillance and damage assessment 7
1.5.1.3 Nuclear, biological and chemical (NBC) attack detection and reconnaissance 7
1.5.2 Habitat Monitoring Applications 7
1.5.2.1 Great Duck Island System: 8
1.5.2.2 Zebra net: 8
1.5.3 Environmental Monitoring Applications 8
1.5.3.1 Glacier monitoring 8
1.5.3.2 Ocean Water Monitoring 9
1.5.3.3 Vineyard monitoring 9
1.5.4 Health and home applications 9
1.5.5 Other commercial applications 9
1.5.5.1 Intrusion monitoring 10
1.5.5.2 Smart roads 10
1.5.5.3 Power monitoring 10
1.5.5.4 Rescue of victims 10
2 Project Scope **11**
2.1 Problem Statement 11
2.2 Scope 12
2.3 Project Objectives 12
2.4 Design Strategy 13
2.5 Deliverables 13
3 Literature Review **14**
3.1 Packet Loss 14

3.2 Flooding 14
3.3 Time Synchronization 15
3.4 Reference Broadcast Synchronization 15
3.4.1 Significance 15
3.4.2 Basic Algorithm 15
3.4.3 Minimal Scenario 16
3.4.4 RBS Issues 17
3.5 Time Sync Protocol for Sensor Networks 18
3.5.1 Significance 18
3.5.2 System Model 19
3.5.3 Basic Algorithm 20
3.5.3.1 Level Discovery Phase 20
3.5.3.2 Minimal Scenario for Synchronization Phase 20
3.5.4 TPSN Issues 21
3.6 Network Time Protocol 22
3.6.1 Significance 22
3.6.2 Basic Algorithm 22
3.6.3 Minimal Scenario 23
3.6.4 NTP Issues 24
3.7 Other Synchronization Issues irrespective of techniques 25
4 EETS: Energy-Efficient Time Synchronization 26
4.1 Assumptions 26
4.2 Adaptive Model 26
4.3 Push and Pull Model 27
4.4 Features 27
4.4.1 Deterministic Energy Consumption 27
4.4.2 Accuracy 28
4.4.3 Single and Multi hop 28
4.5 EERBS 28
4.5.1 Algorithm 28
4.5.2 Design 29
4.6 EETPSN 31
4.6.1 Algorithm 32
4.6.2 Design 31
4.7 WSNTP 33
4.7.1 Assumptions 33
4.7.2 Algorithm 33
4.7.3 Design 33
4.8 Basic Design of EETS 36
5 Simulator 37
5.1 Requirement analysis for Simulator Construction 37
5.1.1 Existing Simulators Support 37
5.1.2 Functional Requirements of our System 37
5.1.2.1 Network Setup: 38
5.1.2.2 Synchronization Simulation 38
5.1.2.3 Simulation 38

5.1.2.4 Results 39
5.1.3 Non-Functional Requirements of our System 39
5.2 Implementation Tool 39
5.3 Basic Features 40
5.3.1 Input Parameters 40
5.3.2 Wireless Sensor Node Distribution on Grid 40
5.3.3 Runtime Node Analysis 41
5.3.4 Simulation 41
5.3.5 Performance Comparison 41
5.4 Testing 41
5.4.1 Unit Testing 41
5.4.2 Integration Testing 41
5.5 User Manual 41
5.6 Simulations and Results 42
5.6.1 RBS Validation 42
5.6.2 TPSN Validation 42
5.6.3 Verification of Improved Techniques 43
5.6.4 EERBS 44
5.6.4.1 Results and Analysis 44
5.6.5 EETPSN 48
5.6.5.1 Results and Analysis 48
5.6.6 Error Analysis 54
5.6.6.1 RBS Time Delays 55
5.6.6.2 TPSN Time delays 55
5.6.6.3 Error and Delay 56
6 Conclusion and Future Work 58

LIST OF TABLES

Table Numbers *Page*

Table 1: Number of Transmission in RBS and EERBS 45
Table 2: Number of Receptions in RBS and EERBS 46
Table 3: Total energy consumption of RBS and EERBS 47
Table 4: Number of Transmissions in TPSN and EETPSN 50
Table 5: Number of Receptions in TPSN and EETPSN..... 51
Table 6: Synchronization Time in TPSN and EETPSN 52
Table 7: Total energy consumption of TPSN and EETPSN..... 54

LIST OF FIGURES

Figure Numbers *Page*

Figure 1: Sensor Node Architecture 2
Figure 2: Wireless Sensor Network 3
Figure 3: Project Scope 12
Figure 4: Minimal Scenario of RBS 17
Figure 5: Minimal Scenario of TPSN 21
Figure 6: Hierarchical Clock Strata for NTP 24
Figure 7: Simulator Architecture 40
Figure 8: RBS Validation Graph 42
Figure 9: Comparison of number of Transmission of RBS and EERBS 45
Figure 10: Comparison of Number of Receptions of RBS and EERBS 46
Figure 11: Total energy consumption of RBS and EERBS 47
Figure 12: Comparison of Number of Transmissions of TPSN and EETPSN 50
Figure 13: Comparison of Number of Receptions of TPSN and EETPSN 51
Figure 14: Average Synchronization Time for TPSN and EETPSN 52
Figure 15: Synchronization Time Versus Flood Time for TPSN and EETPSN 53
Figure 17: Total energy consumption of TPSN and EETPSN 54

KEY TO SYMBOLS/ABBREVIATION

EETS	Energy-Efficient Time Synchronization
WSN	Wireless Sensor Network
NTP	Network Time Protocol
RBS	Reference Broadcast Synchronization
TPSN	Time sync Protocol for Sensor Network
GPS	Global Positioning System

1 Wireless Sensor Networks an Overview

Leaving the era of mainframes computing behind, we are now in the era of personal computing. The third wave of computing i.e. ubiquitous has just begun where the processing environment exists around us and ebbs into background of our lives. Recent developments of such environments are being proposed and worked upon; Wireless Sensor Network (WSN) [1] being the foremost.

WSN is large scale distributed network consisting of many arbitrarily installed small sensing nodes. These nodes are small independent devices capable of performing sensing, processing and communication. The main motivation, behind research into sensor networks, is its military applications. Recent advances in WSN have opened up new opportunities in ubiquitous applications in daily life. Thus, distributed sensor networks represent an evolutionary development step towards smart environments. The importance of sensor networks can be highlighted by the number of prominent research projects funded by DARPA such as Smart Dust, NEST etc.

1.1 Sensor Nodes

WSN is a mesh network which consists of many small sensor nodes randomly deployed over a vast area either inside a phenomenon or in its surroundings. These nodes are small devices autonomous in their communication and computation capabilities. A *sensor node* measures physical value in its environment and passes the information to the mote. Mote consists of a small processor, power unit and A/D converter for connecting to sensor. The sensor and mote together are known as a sensor node. A sensor node's basic architecture is shown in Figure 1.

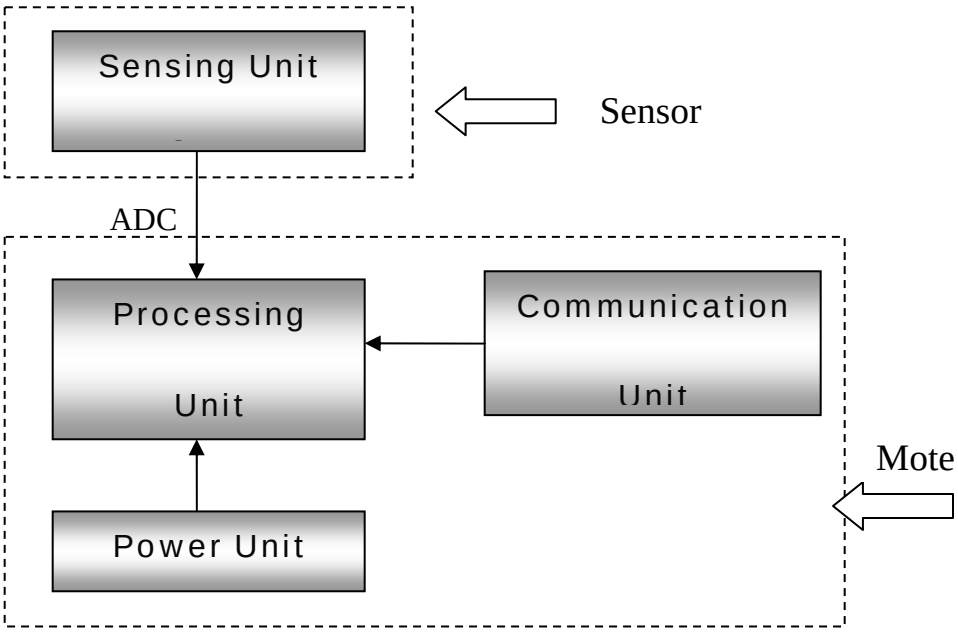


Figure 1: Sensor Node Architecture

1.2 Wireless Sensor Networks

Sensor networks previously consisted of passive devices wired to a central processing station where data was collected. Nowadays they consist of active devices that measure and control tasks within a sensor network. Control tasks include data aggregation and refinement, location awareness, routing, etc. Basic operation of such a network is explained in the following paragraph.

Sensor nodes are fixed or randomly deployed over a region or environment around a phenomenon to be observed. They may remain unattended after deployment. Redeployment or replacement occurs on failure or expiry of the nodes. As soon as the deployment phase is over distribution of tasks amongst the nodes is done either by some external enforcement or preferably by self organization protocols and topology transparent algorithms. These small devices doing different tasks can be clustered

together as smaller network. Sensors usually communicate with each other using a multi hop approach.

Sensor broadcasts its data to its neighboring nodes. Data is aggregated and filtered before further broadcast. Thus data travels from hop to hop and route back to sink nodes. Sink nodes provide networks an interface to the outside world. Thus, each sensor node scattered in the network has the capabilities to collect data and route it back to the sink without an infrastructure.

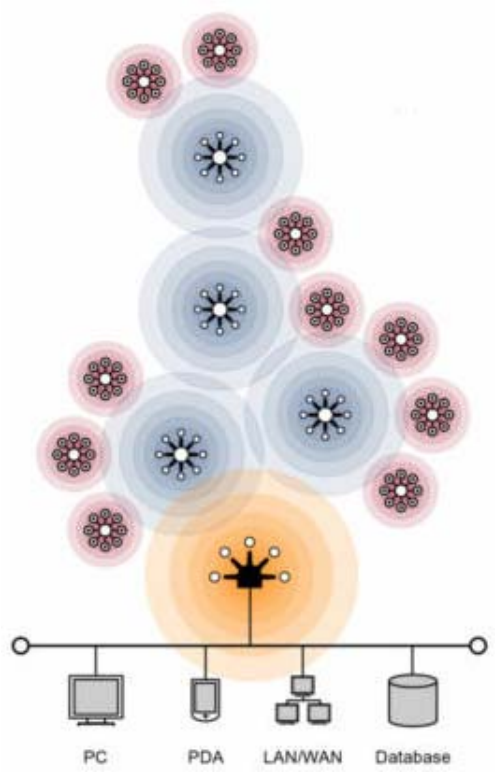


Figure 2: Wireless Sensor Network

1.3 WSN Features

WSN behaves uniquely as judged against classical distributed wired environments. Some of the leading unique features of WSN are discussed as follows:

1.3.1 Deployment

Sensor nodes in most cases are deployed randomly (e.g. dropping them from an aircraft) especially in areas with inaccessible terrain. These areas have no infrastructure available hence the nodes have to identify its location, connectivity and distribution in a co-ordinate system. Deployment affects important properties such as the expected node density, node locations, regular patterns in node locations, and the expected degree of network dynamics.

1.3.2 Unattended Operation

Sensor nodes are usually installed in regions where human intervention is not possible. Secondly in some cases human presence can cause disturbance in behavioral patterns and distribution. Hence, the nodes have to reconfigure themselves in case of changes (e.g. addition or failure of nodes).

1.3.3 Restricted resources

Sensor networks are constrained by resources such as memory and energy because of their small size. The elementary problem of sensor networks is energy consumption. Since sensor nodes are not connected to an energy source, there is only a finite sources of energy, which can either be stored (e.g. batteries) or taken from the environment (e.g. solar, wind).

It has been noted that communication of data requires more energy as compared to processing, therefore, redundant communication in a network should be minimized as much as possible.

1.3.4 Time Synchronization

Time synchronization is a crucial issue in sensor networks. Many applications require local clocks to be synchronized to a high degree of precision. When modeling real world scenarios, identifying time and location in sensor networks is imperative because even a millisecond error can lead to a wrong estimation. Some properties of sensor networks such as limited resources of energy, storage and computation make traditional synchronization protocols infeasible for these networks.

1.3.5 Network dynamics

Mobility has a great effect on network dynamics. It is necessary for the sensor network to be adaptable to changes in connectivity as movement of nodes may have an impact on the amount of time during which nodes stay in communication range of each other.

1.4 Design Factors of WSN

WSN vary from conventional networks in many ways; with scarce energy and other resources limited by size and cost, WSN face a lot of challenges during its design. These challenges are vastly surveyed and investigated by various researchers [1, 6, 7, 8]. Some of the technical challenges posed by WSN are highlighted below.

Energy conservation is the prime issue in design of WSN carefully analyzed. The lack of energy or physical damage to sensors is frequent event in sensor networks, requiring WSN to be *fault tolerant*. It also affects the optimization of *routing* protocols and algorithms based on energy-efficient routes. *Memory limitations* and *data redundancy* encourage optimization of processing at local nodes. *Cost, size and age factor* have to be catered for single nodes to justify overall expense, expanse and expiry of the network. *Deployment* and its maintenance in *network topology control* have to be seen, specifically

in the unattended and unreachable environments. *Coverage* of and by the networks is an application oriented challenge which is evaluated by signal-to-noise ratio (SNR). Few dynamic WSN applications also challenge to handle *mobility* of nodes. *Automatic configuration* and reconfiguration is necessary for remote randomly placed or mobile nodes. *Security* issue emerges as a consequence of wireless communication. It includes, apart from the typical security issues; privacy, availability, authentication, integrity and denial of service. *Scalability* issue arises due to large number of redundant nodes and also due to *heterogeneity*, thus, WSN must accommodate new schemes. *Heterogeneity* promotes adaptable, scalable, generic and reflective middlewares. *Quality of service* is the most demanding issue and it is grasped by identifying and overcoming the challenges which are most likely to affect the application or objective of the wireless sensor network.

1.5 Applications

Wireless technology has found way into variety of applications and systems. Here we give an overview of some applications of WSN, broadly classifying them into five categories such as Military, Environmental, Habitat monitoring, Health and Home, and other applications.

1.5.1 Military Applications

Research into wireless sensor technology was initially motivated by military applications [9]. Some examples of military applications of wireless sensor networks are Monitoring Friendly Forces, Equipment and Ammunition; battlefield surveillance; targeting; battle damage assessment; sniper localization and nuclear, biological and chemical (NBC) attack detection and reconnaissance.

1.5.1.1 Monitoring Friendly Forces, Equipment and Ammunition

WSN can be constantly used to monitor the existing state of friendly troops and the condition and availability of equipment and ammunition in the battlefield. Data is gathered at sink nodes and sent to troop leaders or commanders so that appropriate action could be taken keeping in consideration the situation at hand.

1.5.1.2 Battlefield surveillance and damage assessment

Sensors can be deployed in critical areas where human involvement is not possible, to keep a close watch on enemy forces. This can include tracking the path of military vehicles (e.g. tanks) with magnetometer sensors or tracking the path of enemy troops and feeding it into an intelligent ammunition system which can take an aggressive action when the need arises. Sensor networks can also be deployed in the battlefield to gather information about damage in a battle.

1.5.1.3 Nuclear, biological and chemical (NBC) attack detection and reconnaissance

Senor networks can be used to detect and raise an alarm in case of such attacks. This will lead to a significant decrease in the number of casualties. They can also be used to take counter measures in case of attacks as well.

1.5.2 Habitat Monitoring Applications

WSN present a more economical method for conducting long-term studies than traditional methods. Sensor nodes are deployed prior to the breeding season (in case of animals), to monitor and collect data about animal life. Data is gathered without human intervention, and useful information is inferred regarding the nesting pattern and breeding

season. Some examples of habitat monitoring include Great Duck Island System and Zebra net.

1.5.2.1 Great Duck Island System:

WSN observe the breeding behavior of Leach's Storm Petrel (a small bird). Small battery powered sensor nodes are deployed in patches in the area of interest. The nodes measure temperature, humidity and pressure and light levels. The sensor nodes transmit their data through sensor network to a network gateway. The gateway is responsible for transmitting data to the remote base station, which is connected to a database through a satellite link. The data collected is finally displayed to scientists for evaluation.

1.5.2.2 Zebra net:

Zebra Net WSN observe the behavior of wild animals such as zebras; lions, etc. Animals are equipped with sensor nodes and GPS is used to monitor their exact location. Data is collected from the animals at regular intervals by a mobile base station (e.g. airplane).

1.5.3 Environmental Monitoring Applications

Some environmental applications of sensor networks include, monitoring of environmental conditions for crops, observing effects of environmental changes on animal life, ocean monitoring, forest fire and flood detection, and glacier monitoring.

1.5.3.1 Glacier monitoring

To understand changes in sea level owing to global warming, WSN are used to monitor how glaciers contribute by releasing fresh water into sea because this can cause great disturbances to the thermohaline circulation of water. Currently they are being used to monitor movement of sub glacial bed at Briksdalsbreen, Norway, to better understand the Earth's climate.

1.5.3.2 Ocean Water Monitoring

Sensor nodes equipped with temperature and salinity sensors are dropped from an airplane. They are used to observe the temperature and salinity of the upper ocean. Data measurement in this case is available in real time.

1.5.3.3 Vineyard monitoring

WSN monitor moisture, rainfall, and air and soil temperature to observe the affect of these conditions on plant growth. This gives vine growers better ways to monitor their crop and anticipate problems such as frost in timely manner to avoid crop loss.

1.5.4 Health and home applications

Some health applications of WSN include monitoring of blood oxygen, breathing, heart rate (sleep apnea); diagnosis of rheumatic disease, monitoring abnormal changes in heartbeat, observing side effects of medicine and little sleep, drug administration in hospitals and tracking doctors and patients in a hospital.

Future will see integration of wireless sensor networks with existing specialized medical technology. Medical systems which can benefit from WSN are smart homecare. They extend traditional clinic to patients' home and make In-home tasks easier for example using remote device control, medicine reminders, object location, emergency communications.

1.5.5 Other commercial applications

Other applications of WSN include intrusion monitoring, monitoring product quality, power monitoring and environmental control in an office building, vehicle tracking on a road, and rescue victims in disaster hit areas.

1.5.5.1 Intrusion monitoring

WSNs can detect movements inside a building which is supposed to be empty. If a movement is detected, the network raises an alarm and the security is informed. A security team when enter the building can collect the logs of alarm from the network with the help of any mobile device (e.g. PDA, Laptop).

1.5.5.2 Smart roads

The goal of such a WSN is to improve driving safety by providing the drivers with an early warning of dangerous situations that may arise. The system can be further enhanced to provide facilities such as vehicle tracking over large distances, vehicle counting at intersections for traffic management, pedestrian on road notification, erratic driver warning etc.

1.5.5.3 Power monitoring

WSN can monitor power consumption in a building, by detecting devices which consume a lot of energy. The goal is to identify these devices and then find ways to reduce power consumption.

1.5.5.4 Rescue of victims

WSN are currently being used to rescue victims buried in avalanches. The goal is to locate the exact position of the victims and to give the rescue workers more information about the state of the victim.

Sensor networks can also be used in underground mines to determine hazardous conditions to prevent occurrence of accidents. They can also be used in mines to rescue victims in case if a mine collapses.

2 Introduction

Time Synchronization provides the backbone for any network that looks towards an efficient operation; however, the synchronization protocol itself consumes a lot of energy. Time synchronization schemes developed for traditional networks such as Network Time Protocol (NTP) [5] are ill suited for WSN due to its complexity and excessive energy consumption Reference Broadcast Synchronization (RBS) [2] and Time synchronization Protocol for Sensor Networks (TPSN) [3] both report high precision with a few microseconds of uncertainty. However, the performance of these protocols is dependent on the number of packet communication. These algorithms work effectively for smaller networks; however as the network size increases, their energy consumption becomes inefficient.

2.1 Problem Statement

To save energy in existing time synchronization algorithms i.e. Reference Broadcast synchronization (RBS), Time Synchronization Protocol for Sensor networks (TPSN) to maximize lifetime of WSN and to investigate implementation of Network Time Protocol (NTP) for WSN.

2.2 Scope

Scope of our project can be seen in wireless sensor network design factors paradigm.

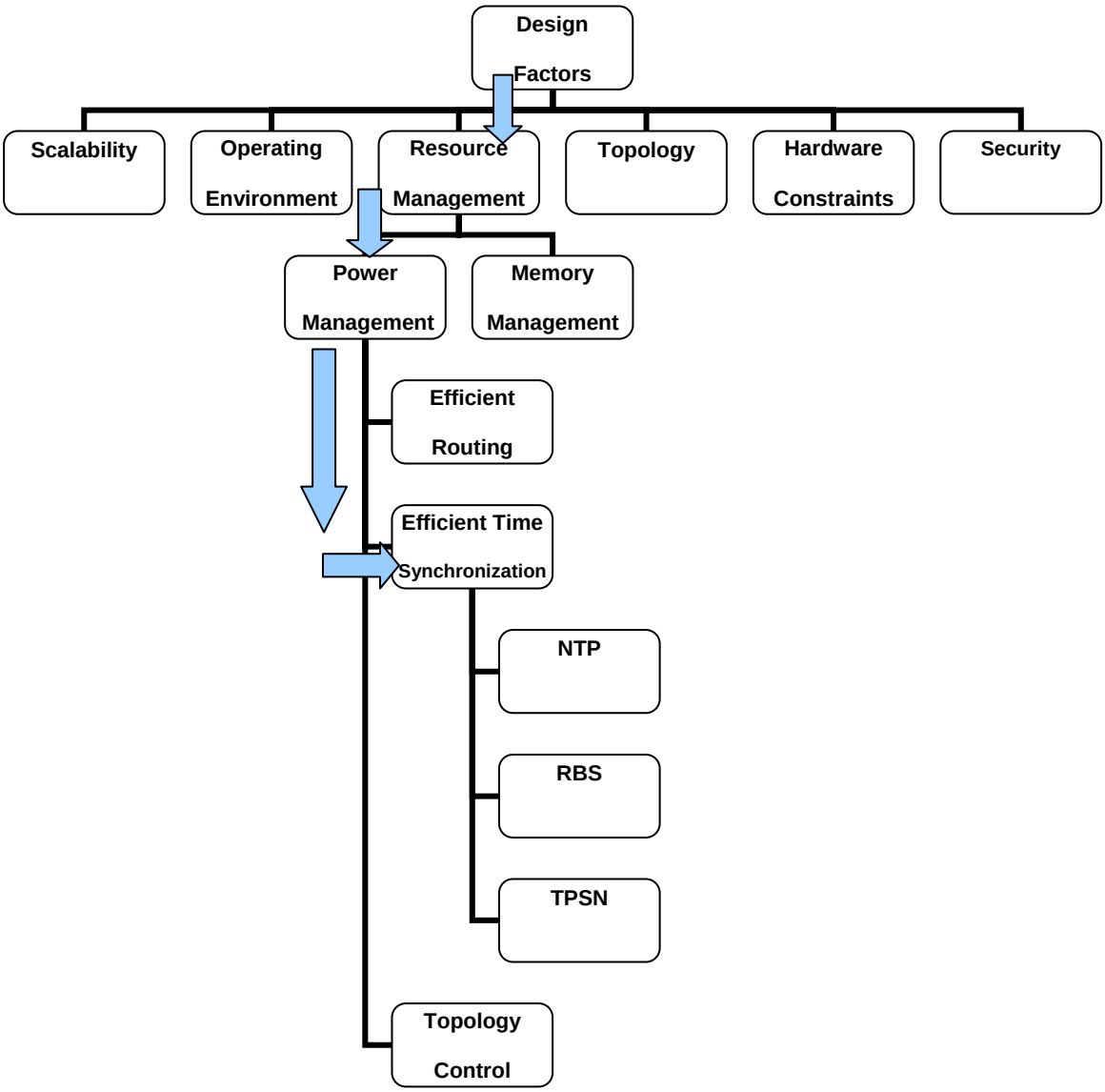


Figure 3: Project Scope

2.3 Project Objectives

- Outline the energy consumption of RBS and propose low power algorithm, which minimize the number of messages exchange needed;

- Design and implement improvement in TPSN that would achieve synchronization at low energy cost;
- Investigate redesign of NTP, originally designed for the wired networks, for WSN by introducing the concept of energy-awareness.

2.4 Design Strategy

The following phases were a part of our research study:

- Literature Review
- Problem Identification
- Problem Analysis
- Proposition of new approach
- Simulator Construction
- Testing and Validation of simulator
- Simulation of relevant algorithms
- Evaluation of relevant algorithms and
- Simulation of proposed algorithm
- Evaluation of proposed algorithm
- Documentation

2.5 Deliverables

- Algorithm
- Simulation of Algorithm
- Research Paper
- Documentation

3 Literature Review

3.1 Packet Loss

Reliability and routing was the main topic of research initially. WSN makes wired networks impractical for they are deployed in relatively remote environments. Consequently, packet exchange is unreliable and lost packets are common. Equation 1 shows sensitivity threshold P_R i.e. the faded signal power as it travels away from the transmitter:

$$P_R = \frac{P_T}{d^c}, \tag{1}$$

where P_T represents the power of the transmitted signal, d is the distance from the transmitter, and c is the path loss coefficient. WSN typical environments have large path loss coefficients because the signal fades due to reflections, diffractions, and scattering off the hindrances.

3.2 Flooding

Flooding algorithms such as the one in [15] have been used to study multi-hop routing in WSN. Unreliable networks have several issues that were revealed with this algorithm:

- *Backward links:* a link that transmits flood packets back towards the source.
- *Long links:* a link that is significantly longer than would be expected given the transmission power level.
- *Stragglers:* sensors that do not receive flood packets, despite having a high probability of reception from a neighboring transmitter.
- *Clustering:* a node that connects to a very large number of receivers.

3.3 Time Synchronization

A typical wireless sensor node such as Crossbow's Mica2DOT [16] has very constrained resources as compared to wired LANs packed with immense power and memory resources. Its processor performs at 4 MHz with 4Kb memory. Most of the battery's energy is consumed by radio transmissions. Continuous transmission of radio signals can deplete the node within few hours even with this low-power hardware. It makes conventional synchronization procedures like NTP inadequate for WSN which acts as parasites for energy. A few novel synchronization methods have surfaced specifically for sensor networks, such as TSPN [3], RBS [2] etc. Appraisal strategy of these algorithms, to identify which is the best technique, in reference to each other, is a complex task. Choosing appropriate factors for evaluation across a palette of design issues totally depends upon the criteria defined for certain applications and its requirements. We explain the significance of each protocol we choose before giving details to highlight the importance of the original techniques.

3.4 Reference Broadcast Synchronization

3.4.1 Significance

RBS exploits broadcast communication at MAC layer by sending messages simultaneously to synchronize the network. Thus it works faster than classical synchronization technique NTP which is meant for wired medium.

3.4.2 Basic Algorithm

Reference broadcast synchronization is a simple receiver-receiver protocol, which synchronizes a set of receivers with each other. In this scheme, a beacon node sends a reference broadcast to its neighbors. The reference packet does not contain any timing

information; instead, the receivers compare their clocks with one another based on the arrival time of the reference beacon at its MAC layer.

The reference broadcast algorithm works in the following way [10]

- A beacon node sends *m* number of reference packets.
- *n* number of receivers observe the time of arrival of the packets according to their local clock.
- The receivers exchange observations.
- Each receiver computes its phase offset to any other receiver in the network to determine its clock drift from the other.

3.4.3 Minimal Scenario

The simplest form of RBS involves transmission of a single reference packet to two receivers (A and B). The receivers record the time of arrival of the packet according to their local clocks. Let the nodes be A, B, and S (A and B to be synchronized by S).

- S sends a broadcast.
- A and B receive reference packets and record the arrival time of the packet.
- They then send pulse reports to S (A and B unicast).
- S computes the conversion and publishes the conversion parameters.

In all, exchange of four messages takes place to synchronize two receivers as shown in Figure 4. However, the above exchanges do not synchronize S to A or B, for that A or B would also need to broadcast.

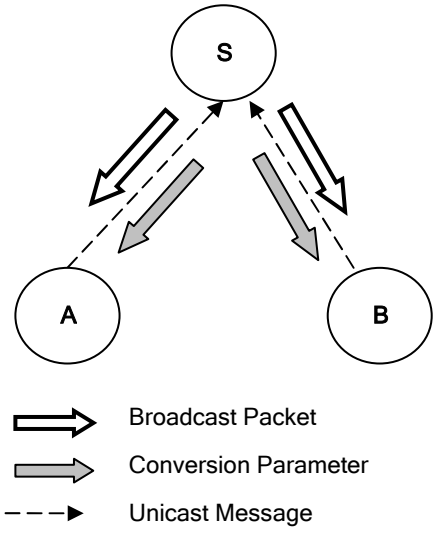


Figure 4: Minimal Scenario of RBS

The authors of RBS reported 11.2 μsec for the synchronization error on the MICA2 wireless sensors. It is also found, average synchronization error of RBS is $6{:}29 \pm 6.45\mu$sec, 8 times better than that of NTP in a lightly loaded network, when using IPAQ PDAs with 802.11 wireless Ethernet.

3.4.4 RBS Issues

There are a few issues associated with RBS synchronization protocol. First, it uses the broadcast channel to synchronize receivers to one another; therefore, this protocol is not applicable in a point-to-point network.

Furthermore, the fundamental property of RBS is that it allows synchronization of receivers in a single broadcast to one another. To accomplish receiver-receiver synchronization, each node sends a reference signal; resulting in reports that go back to the beacon nodes, consequently signaling of conversion parameters publishing.

RBS protocol is used in many different variations [11]. In the above mentioned implementation of RBS, all nodes beacon [10]. The performance of this protocol is

dependent on the number of receivers that are to be synchronized [12, 13]. RBS scales poorly with dense networks where there are many receivers for each transmitter.

Given *n* number of nodes, we have following number of transmissions and receptions for a single transmitter.

$$TX_{RBS} = 2 + (n - 1) \quad (2)$$

$$RX_{RBS} = 3\,(n - 1) \quad (3)$$

For a single-hop network of *n* nodes, when all nodes can beacon, this protocol requires $O(n^2)$ message exchanges. This can be computationally expensive and the time required to synchronize the network can be high.

Given *b* number of beacon nodes, we have the following number of transmissions and reception when all nodes are beacon nodes.

$$TX_{RBS} = \sum_{b=2}^{n} b\,[2 + (n-1)] = n\,[2 + (n-1)] = n^2 + n \quad (4)$$

$$RX_{RBS} = \sum_{b=2}^{n} b\,[3\,(n-1)] = 3n\,(n-1) \quad (5)$$

Thus, for a large number of receivers per transmitter makes this protocol infeasible due to energy constraints.

3.5 Time-Sync Protocol for Sensor Networks

3.5.1 Significance

We choose Time-sync Protocol for Sensor Networks (TPSN) for further enhancements based on the facts that it synchronizes the entire network achieving accurate clock synchronization within *20 μsec* of uncertainty and it is scalable for multihop ad hoc networks. Moreover, its claim to supersede NTP, a high class universal Network Time

Protocol for wired network such as internet, and its two times better performance than RBS, gives us enough confidence to mark it as one of highly suitable protocol proposed for sensor networks [3].

Several techniques like RBS exploiting MAC layer, TPSN intends for a large-scale network where it can synchronize the network through multiple hops. Starting from the root node, TPSN synchronizes the complete network in a hierarchical fashion that maintains the idea of an infrastructure amongst the nodes. However, infrastructure is very flexible to be changed accordingly when network breaks or has to reconfigure.

Due to sender and receiver participation in TPSN and acknowledging the time stamps makes it more precise than other WSN synchronization techniques. However, it is still not as precise as standard but expensive and computationally rich techniques like NTP, GPS etc. TPSN, as claimed by its authors, is a highly adaptive and flexible technique that can maintain accuracy, and many other design factors in variations of TPSN algorithm. Making use of this adaptability, we try to maintain the outcome in both respects of energy and accuracy to achieve solution that is more efficient.

3.5.2 System Model

With a few assumptions, TPSN starts simulating the system scenario with *N* number of nodes deployed in a certain area. In this technique network is configurable as a multihop. In a multihop scenario each node synchronizes to its parent node in a hierarchical tree, developed through flooding, where each parent establishes a separate hop. Each node maintains its own clock and it sets to its corresponding parent in the hierarchy. Network initiates synchronization from a root node, which spreads out through the network from node to node and hop to hop.

3.5.3 Basic Algorithm

Two basic phases summarize TPSN in its simplest form. Level Discovery Phase assigns each node a level in a multihop network. Root node initializes Synchronization Phase.

3.5.3.1 Level Discovery Phase

Level discovery simulates flooding algorithm

- Assigning level *0* to root node
- Neighboring nodes within the coverage and capacity of a node becomes the children of the parent node
- Each child node is assigned *i+1* level where *i* is the level of parent node
- Node at level *i* can communicate with at least one node at level *i-1* i.e. its parent

Later in synchronization phase, each i^{th} level node synchronizes to $i\text{-}1^{th}$ node to which it corresponds.

3.5.3.2 Minimal Scenario for Synchronization Phase

Basic communication can be understood by considering three nodes, of which one is sender S (parent) others are receiver A and B (children of source S). Mechanism is described in Figure 4. In this scenario, S sends a broadcast packet to A and B. In return, S receives unicast pulse report packets from both A and B. Computed result from S is published back to both A and B. S receives an acknowledgment packet from A and B respectively.

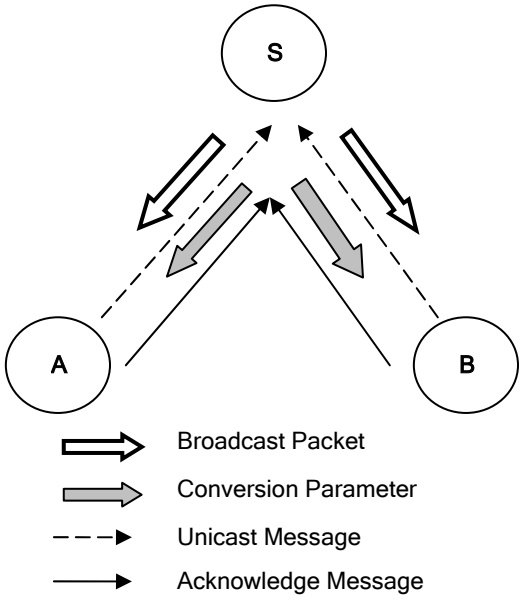


Figure 5: Minimal Scenario of TPSN

3.5.4 TPSN Issues

TPSN [3] simulates the concept of NTP [4] in its algorithm to some extent. Thus, it generates few superfluous inherited steps in its basic implementation that we can reduce to minimize the packet traffic. In NTP, an external source injects time stamps to root level servers also called as stratum 1. The hierarchical servers at stratum n synchronize the leaf terminals called clients. The WSN requirements radically vary from traditional wired computer networks. Highly dense sensor networks need scalable algorithm to cater synchronization of random number of nodes deployed and those that enter or leave the network at adhoc basis. Another important design factor is energy consumption due to battery scantiness of wireless sensor nodes. Hence, power greedy techniques like GPS, and processing complex, always-on models like NTP prove to be inappropriate for WSN. Nevertheless, TPSN with several variations follows the same concept of NTP. The simplicity of algorithm introduces many such steps, which may not be necessary in order

to carry out the synchronization in a network, hence minimizing the steps of an always-on model.

3.6 Network Time Protocol

3.6.1 Significance

NTP is the internet standard time synchronization protocol used for computer systems. The primary author and maintainer is David Mills at the University of Delaware. The NTP code has been ported to a large amount of UNIX and non-UNIX machines and is rigorously backwards compatible.

3.6.2 Basic Algorithm

After receiving a sync request the higher stratum node implements the following instructions:

1. For each m associations construct a correctness interval $x = t \pm rootdist$
2. Consider the low point, highpoint and midpoint of these intervals. Sort these values in a list from lowest to highest. Set the no. of false tickers f = 0.
3. Set the no. of midpoints d = 0. Set c = 0. Scan the lowest endpoint to the highest. Add one to c for every low point, subtract one for every highpoint, and add one to d for every midpoint. If c >= to m-f stop; set l = current low point.
4. Set c = 0, scan the highest endpoint to the lowest. Add one to c for every high point, subtract one for every low point, and add one to d for every midpoint.

   If c >= m-f, stop; set u = current highpoint.
5. If d >, some midpoints are outside the interval
   - if yes add one to f. is f < m/2
     - If yes return to step 3

- if no error, a majority clique could not be found

- else
  - If u > l?, success the intersection interval is [l,u]
  - If no then return to step 5.

3.6.3 Minimal Scenario

Like many other Internet-based distributed system, NTP is hierarchically-oriented. That is, there is a small core of primary time servers who set their clocks against external, highly accurate sources of time information (Cesium clocks, GPS receivers, etc.). Below this level are secondary servers which are responsible for distributing time from the primary servers to the rest of the Internet. In its most basic case there is one stratum-1 clock and two stratum-2 clock. The root node sends a broadcast signal and waits for the acknowledgment. After it receives the acknowledgment from the two secondary level clocks, it will store their information and start with the selection, update and combination algorithms. This procedure will re-occur if and when a sync request is sent to the higher stratum clock or after every 10 seconds the poll interrupt counter automatically sends an interrupt signal and sync process starts. According to the author Dave Mills the sync process even in the most extreme conditions takes place within seconds (on average 1-3) and in the best case scenario the average sync time is 0.25ms [5].

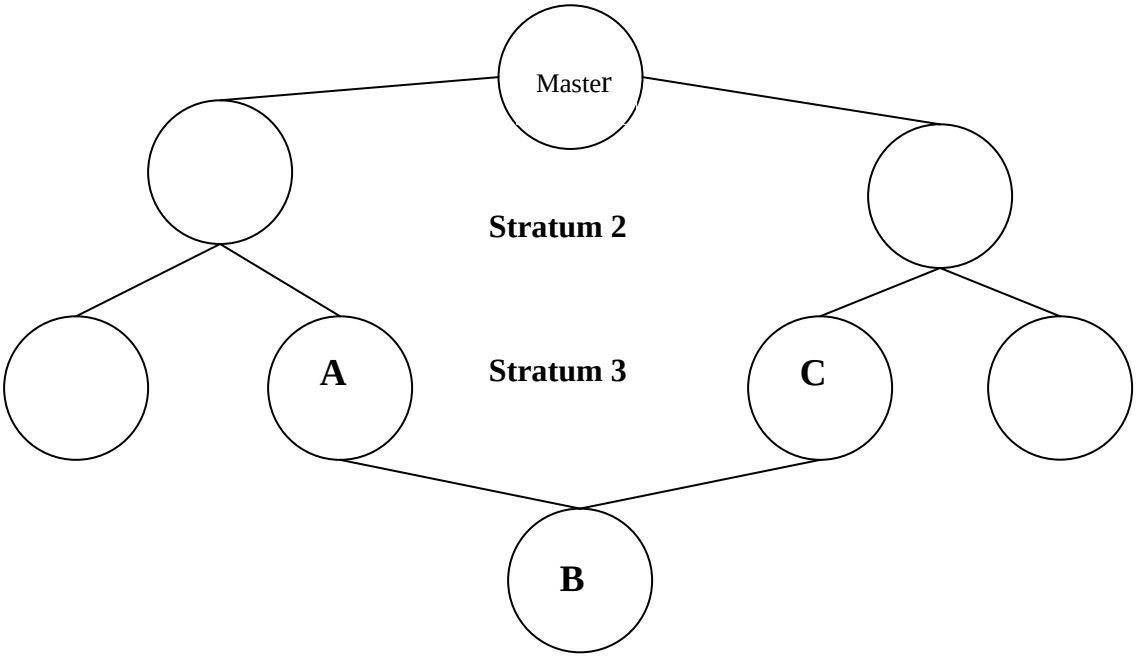


Figure 6: Hierarchical Clock Strata for NTP

3.6.4 NTP Issues

This protocol uses a fairly large amount of memory to store data for synchronization sources, authentication codes, monitoring options, and access options. As mentioned earlier, typical wireless sensor nodes have limited onboard memory. A large sensor network will entail large files for synchronization sources and codes. Even if they can be programmed into each node, it would leave very little memory to hold the data monitored by the sensor, limiting NTP's use for WSNs.

Our area of interest was to make NTP feasible for WSN and for that we chose to introduce a concept of energy awareness in NTP. Since NTP has been specifically designed for wired networks its basic assumptions violate WSN requirements:

- CPU listening is free
- Retransmissions have a negligible impact
- Network is based on an always on model

These three issues have been the reason why work on NTP for WSN has been discarded. [4] points out the three issues mentioned above as well as other problems involved with NTP being introduced in WSN, such as the transmission path between a sender node and a receiver node is not reciprocal which is assumed to be true for two way communication in wired networks and that NTP does not provide low error rates required for WSN and thus protocols specially tailored for WSN need to be implemented.

3.7 Other Synchronization Issues irrespective of techniques

Despite of the basic algorithm, may it be the original or improved versions; there is an observation that there is no need to repeat the synchronization for the complete network every time. Moreover, in a randomly deployed network, the dynamic connectivity and coverage of the nodes, it becomes important for them to be discovered and remain in the hierarchical infrastructure. Network must handle such problems locally instead of reconfiguring the entire network repeatedly.

Another inspiration is using the existing energy-efficient algorithms of other design factors and features of WSN within the scheme. Such as while constructing the network hierarchical levels for nodes' discovery, we make most optimal tree hierarchy, which reduces the overall transmission strain in a multihop network. Similarly, while adding a new node to network or updating a specific node can retrieve through source node via energy-efficient routing techniques.

4 EETS: Energy-Efficient Time Synchronization

EETS is a self-contained scheme of synchronization leveraging the capabilities of WSN by eliminating the bottleneck of energy consumption to great extent.

4.1 Assumptions

The Wireless Sensor Network consists of *n* number of sensor nodes. It has two types of nodes; beacon nodes that broadcast packets and nodes that receive these broadcast packets to synchronize their clocks. Our assumption is that these nodes receive synchronization signals as an indication to align their clocks. In our technique EERBS, it does not identify the source of these signals but is only concerned with the nodes, which hear them. In EETPSN Nodes synchronize to some real time through a single source node that maintains ideal time in hierarchical network configurations.

It is also assumed that packets are not lost or corrupted in transmissions on the wireless network. Nodes within the network may move at any time without notice, and may even move continuously, it is assumed in this document that nodes are immobile and the topology of the network remains constant throughout the synchronization process.

4.2 Adaptive Model

Original TPSN bases on post facto synchronization [3], which is the most suitable methodology for WSN. This technique proposes that the synchronization initiates only when it is required instead of always-on model that keeps refreshing the clocks continuously. Extending this idea by synchronizing clocks once at the time of network setup phase and then at an occurrence of sensing event, if less frequent. Otherwise, for frequently occurring events a suitable time can be set, after which the network may

resynchronize. Reducing the steps to minimal with respect to the scenario and the requirements eliminates some work surplus to them. Customization of the algorithm for a specific technique may make it a bit complex but it sieves away the extra tasking that is not required otherwise.

4.3 Push and Pull Model

Push model is the simplest scenario of the algorithm where source initiates the synchronization and pushes outwards through levels, synchronizing all of the nodes within the network. This method must be called once at configuration time of the network and again when the complete network is to be synchronized.

Pull model will be implemented generally more often when a new node is added to the network, or it is rediscovered due to break or disconnection in network. Depletion of a parent node or out of range coverage because of mobility can cause such disconnections. Thus, in pull model the node sends messages to its neighbor nodes at level *i* and joins the network by assigning *i+1* level to itself. Through the parent node, it route back from parent to parent towards source node and get the synchronized value.

4.4 Features

4.4.1 Deterministic Energy Consumption

Foremost consumption of energy is because of communication between nodes. Number of transmissions and receptions determines it by knowing the transmission draw current and reception draw current values of the sensors being used. Reception to transmission ratio can be used to determine consumption in normalized units.

4.4.2 Accuracy

Due to real-time performance of WSN we can not determine the exact error of the timestamps after nodes are synchronized. We targeted to achieve deterministic errors by classifying it in sent time domains, propagation, and access, receive times.

4.4.3 Single and Multi hop

Tiered architecture is common to virtually all sensor networks. Sensor nodes are randomly deployed over a region around a phenomenon to be observed. Initial deployment experience has shown that these systems will require a hierarchy of nodes starting with simple sensor nodes and continuing up to data aggregation, analysis and storage nodes. The higher-level nodes form a distributed infrastructure designed to support the operation.

4.5 EERBS

4.5.1 Algorithm

In Section 3.4 we showed that the current single hop implementation of RBS generates a lot of traffic. To extend the life of a sensor network we propose an improvement in the algorithm, which conserves energy besides synchronizing the network. Algorithm 1 shows that by varying the number of beacon nodes in the network, we can reduce the number of transmissions and receptions and save considerable amount of energy

Algorithm 1: EERBS

```
Select numBeacon number of beacon nodes
For each Beacon node
     Transmitter broadcasts sync_request
     Receive pulse reports from its neighbors
     Broadcast conversion parameters
For each receiver
     Record local time of reception for sync_request
     Unicast pulse reports to the beacon node
     Receive conversion parameters from the beacon node
```

4.5.2 Design

The algorithm minimizes the number of transmissions and reception by selecting a subset of the total nodes to beacon. When a node receives a synchronization signal, it runs the *SelectBeaconNodes* function to select *numBeacon* number of beacon nodes. Remember that for *n* numbers of nodes, those lie in a single broadcast domain; require a minimum of two beacon nodes to synchronize the entire network. A value less than two passed to the function generates an error as shown in Algorithm 2.

Algorithm 2: `SelectBeaconNodes`

```
If numBeacons < 2
     'Error - minimum 2 Beacon Nodes required'
End if
For 1 to numBeacons
     Randomly select beacon nodes
     If current_node_power >= Energy_Threshold
               Mark node as beacon node
     Else
          Mark node as depleted
          Change beacon node
     End if
End For
```

Once the required numbers of beacon nodes are selected, the *SyncReqRBS ( )* is invoked. Broadcast messages are sent to all the neighboring nodes that lie within the range of the beacon node.

As the nodes receive broadcast messages the record local time of reception for *sync_request*. The receivers then invoke `SyncReport` function to unicast pulse report packets to the beacon nodes.

Beacon nodes receive these pulse reports and perform conversion computation. They then broadcast the conversion parameters to all the other nodes in the network.

4.6 EETPSN

Our technique is based on post facto synchronization [3], which is the most suitable methodology for WSN. It proposes that the synchronization initiates only when it is required instead of *always-on* model that keeps refreshing the clocks continuously. Regardless of the fact that the network is *always-on* or *post-facto*, we improve minimal scenario and overall phased algorithm of TPSN for complete synchronization of WSN.

4.6.1 Design

Our proposition is to challenge the simplicity of TPSN algorithm which overall requires more computation because of greater number of steps. We can reduce the unnecessary steps as discussed below.

Conceptually TPSN defines two separate phases. Nodes send prompt messages to its neighboring nodes for hierarchical development, and then separately send synchronization information in another message. We propose combining *Level discovery* phase with the *Synchronization* phase. We suggest that instead of wasting *hello* message, send useful information of time stamps at *level discovery* phase. Thus combining the initial steps of synchronization with level discovery and represent it as single handshake.

Algorithm 3: `Flooding based hierarchy of network`

```
Send msg synchronization request with parent's time stamp
If msg received for the first time then
     Set parent on tree = source of msg,
Change source field to current id
Increment hop_count field
Rebroadcast packet
```

4.6.2 Algorithm

The minimal scenario as mentioned in section 3.5.3.2 represents the actual traffic generated at *synchronization* phase. We omit acknowledgment packet with the justification that when *A* sends back pulse report r_1 it already acknowledges the connection between *S* and *A*. If the packet r_n is lost somehow, *A* will prompt *S* again with the request of synchronization through pull model. Such an exception is very less probable, giving us enough confidence in excluding this packet. Even with worst-case scenario, when packet is lost fault tolerance is kept in stance. Simplicity of our refinement saves a lot of traffic when it comes to a larger network. The Algorithm 4 shows implementation of EETPSN.

Algorithm 4: `EETPSN`

```
For each Transmitter
     Transmitter broadcasts sync_request
     Receive pulse reports from its child nodes
     Broadcast conversion parameters
For each Receiver
     Record local time of reception for sync_request
     Unicast pulse reports to the transmitter
     Receive conversion parameters from the parent node
```

Adaptive approach of the algorithm allows synchronizing the entire network, from a source node to its children level by level in a hierarchical fashion it can synchronize the children of the given node; or two nodes with respect to each other. Furthermore, its implementation is scalable to single hop as well as multihop. Introducing pull model we provide scalability to synchronize newly added.

4.7 WSNTP

4.7.1 Assumptions

For our investigation into NTP we have made certain assumptions. We assume that there is little or no network jitter causing packet loss and/or retransmission due to noise and that there are two type of relationships between higher and lower stratum nodes i.e. master/slave and peer to peer. The NTP algorithm is running on static nodes deployed in a grid or random basis. Also we assume that the communication speed between wired and WSN is the same.

4.7.2 Algorithm

In section 3.6 we outlined certain issues concerning NTP in its original form. We propose to introduce the concept of energy awareness into NTP. Its aim will be to have a check introduced within NTP. The check will provide a basis for the algorithm to work for a WSN environment. The check is to be introduced in the clock *update ( )* function.

Algorithm 5: WSNTP

```
J=0
While (a (J) <1) do
     Saturate shortest path
     Update vectors b and c
     F (J) = F (J-1) + c
     J=J+1
```

4.7.3 Design

On termination of the algorithm the function F (J) will give the total amount of data collected. Whilst choosing the shortest path from source to sink the algorithm takes into

account the amount of energy consumed (energy awareness) and the amount of data sent or received (data awareness).

An iterative implementation of the above algorithm is as follows:

- The sink sends a message to all its neighbors specifying iteration number *k* and declaring shortest path to sink as 0.
- Each neighbor then initializes shortest path to sink.
- The active and threshold (neighbor) nodes deploy a distance vector algorithm after which each node has length of shortest path and next hop to sink.
- Each active and threshold node sends a response to sink to find the capacity of path along which it has been sent.
- The sinks selects candidate node with min pk and if the value is greater than or equal to 1 the sink sends a message to all nodes to scale down by a certain factor and program terminates.
- If less than 1 sink sends message to node z sending minimum capacity info along the path.
- Node *z* updates its routing table.
- Return to first step

Along with this NTP at every poll interrupt calls the following functions as well:

Cluster( ) – Allocation of highpoints and lowpoints.

Intersection( ) – Allocation of midpoints

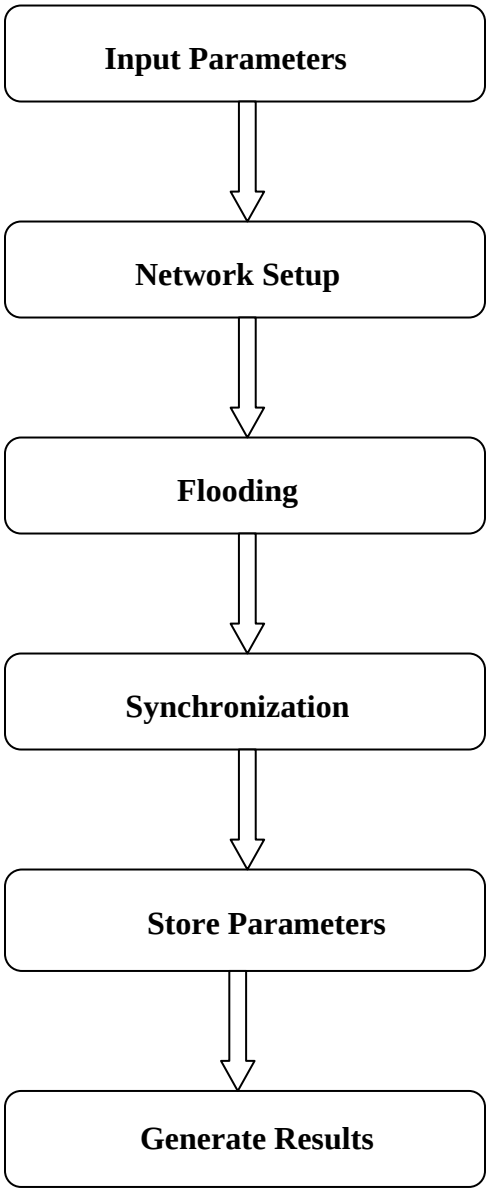
Combination( ) – Combining selected peers

Packet( ) – Packet information/sync request

Distance( ) – Calculate distance algorithm

Select( ) – Selection of peers

4.8 Basic Design of EETS



5 Simulator

5.1 Requirement analysis for Simulator Construction

5.1.1 Existing Simulators Support

WSN is although currently gaining popularity in distributed application, it lacks much standardization which is a major requirement in future of wireless distributed computing. There are a large number of simulator systems available that provide adhoc networks simulations of different quality and maturity. Majority sensors' test-beds simulate the protocols in TinyOS establishing a real world WSN. For that purpose the most accepted sensors mote kits are Micaz Motes [16]. These require a self configured establishment of a test-bed first which is impractical in such a short span of project scope.

Generic architectures are still not standardized and require a lot of initial configurations for WSN setup. Only then any protocol can be tested over those configured network. Configuration itself is a tedious task with indefinite number of design factors. Many other simulators are too specific to certain proposals of other design issues that they become irrelevant to synchronization protocols. Separation of concerns therefore becomes important in the services of simulators. In most of the cases, the features are strongly coupled and entangled to an extent that it is very difficult to utilize them independently.

5.1.2 Functional Requirements of our System

The motivation for our simulator was to design and implement a simulator independent of WSN complexities. In our case, we first identify the core requirements for simulations and assume all those design factors constant which are ineffective to our approaches.

The functional requirements have been identified to include:

5.1.2.1 Network Setup:

- Hardware parameters must be loaded to make available sensor node's capacity
- Flood parameters discovers the node's capacity to provide a certain coverage
- Grid parameters defines physical deployment of the network
- Grid must be plotted such that user can easily identify the source and ordinary nodes. In addition the node's status should be visible to know whether the node is depleted or left orphan.
- Node analysis on runtime
- Change source option on runtime

5.1.2.2 Synchronization Simulation

- Network establishment based on basic decay flooding protocol [15] and hardware parameters [16] of deployed nodes.
- Choice of single and multihop scenarios
- Synchronization from child to a single source
- Choice of synchronization technique
- Transmission and receptions uniform for all techniques
- Power consumption standard for certain sample based on reception to transmission ratio

5.1.2.3 Simulation

- Simulation should be visually visible with user defined pause interval
- Flooded network should remain intact with synchronization call.
- Any node can be traced to its root

- Resynchronize a specific node to its root
- Other functions must be disabled while synchronization is being performed

5.1.2.4 Results

- Runtime remaining batteries of nodes
- Average number of nodes communication
- Flooding and synchronization time
- Testing results should be generic to user to choose number of samples
- Save files on system for each simulation sample to tolerate faults
- Generate result by loading the saved file

5.1.3 Non-Functional Requirements of our System

The non-functional requirements have been identified to include:

- **Easy Execution**: The system is required to have easily executed by user.
- **Fault Tolerance**: The system is designed to support fault tolerance and to provide quality of service (QoS) to the applications in which it works.
- **Efficiency**: The requirement of the system is to provide a highly efficient system that resembles the efficiency of normal execution.

5.2 Implementation Tool

Simulator is built in Matlab (version 7.0.0.19920 (R14)). It consists of three sections. Section one allows the user to input various parameters such as number of nodes, area in which the nodes are to be deployed, maximum power of the nodes etc. Section two is the main interface which allows the user to view the deployment of the nodes on the grid. It

also allows the user to choose the type of algorithm, to view its simulation. Section three allows the user to view graphs generated for various output parameters

5.3 Basic Features

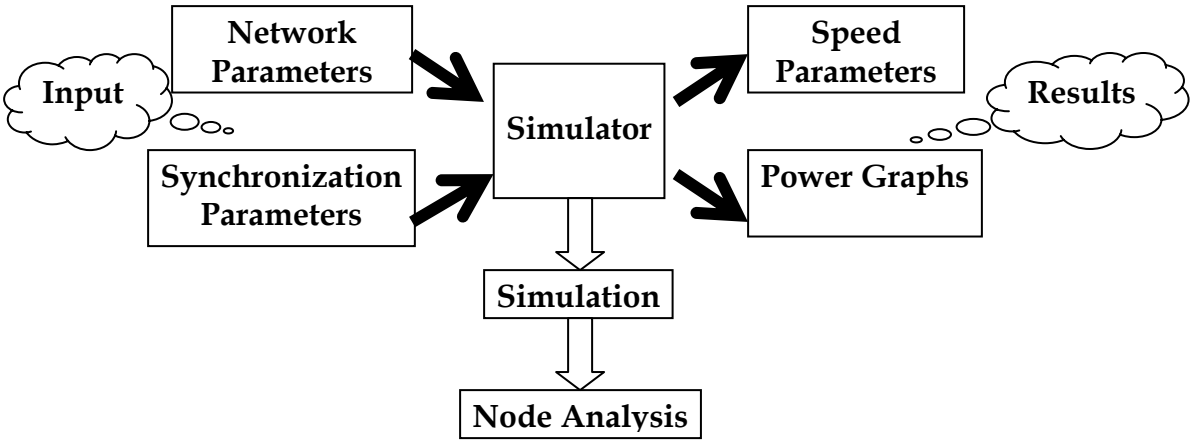


Figure 7: Simulator Architecture

5.3.1 Input Parameters

We take input parameters from simulator interface. It allows:

- Specifying grid where the network is to be deployed.
- Hardware Parameters define the nodes power specifications
- Flood parameter entail coverage capacity of nodes

5.3.2 Wireless Sensor Node Distribution on Grid

Deployment is visualized on grid where nodes are randomly deployed. One source node is elected randomly or it can be selected on grid. This source node starts discovering neighbors and builds relation between nodes. Those nodes which are not configured are left as orphan nodes.

5.3.3 Runtime Node Analysis

Any selected nodes data can be analyzed on runtime. It shows its ID, remaining battery, parent, and children.

5.3.4 Simulation

All connected nodes synchronize on the basis of selected synchronization algorithm. Simulations can be tested continuously for various deployments through flood tests. These simulations are saved on system to tolerate faults.

5.3.5 Performance Comparison

All data sets are compared on generated graphs which maintain comparison between the algorithms and shows the network and communication parameters.

5.4 Testing

5.4.1 Unit Testing

Unit testing concentrates on simulation of each technique as implemented in source code. In EETS each component as developed is individually tested so as to check for possible errors that could occur.

5.4.2 Integration Testing

In integration testing focus is on design and the construction of software architecture. We have integrated the techniques keeping generality of simulations.

5.5 User Manual

Refer Appendix

5.6 Simulations and Results

5.6.1 RBS Validation

The first sets of simulations were run to validate equation 4, which is the basis for the RBS algorithm's behavior. In this experiment, all other parameters are kept constant: 10 simulations are run over a 1m x 1m area which is randomly populated with 10 sensors, which are then varied up to 100 in increments of 10. The RBS algorithm is then executed for each value and a graph is plotted in Excel. The figure below shows how the change in number of nodes affects the number of transmissions and reception of RBS

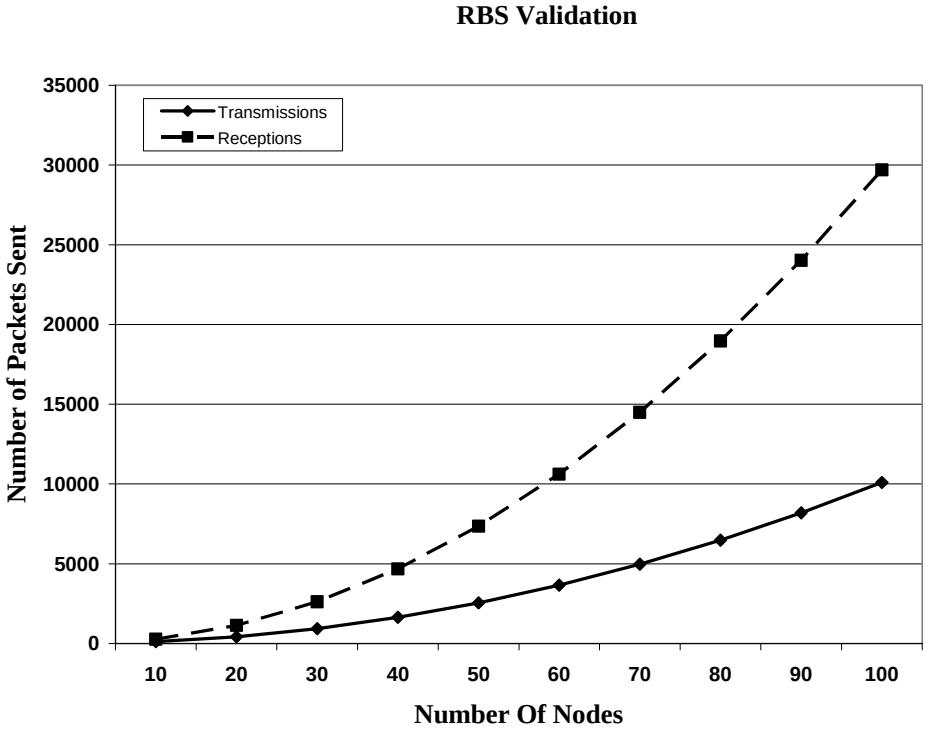
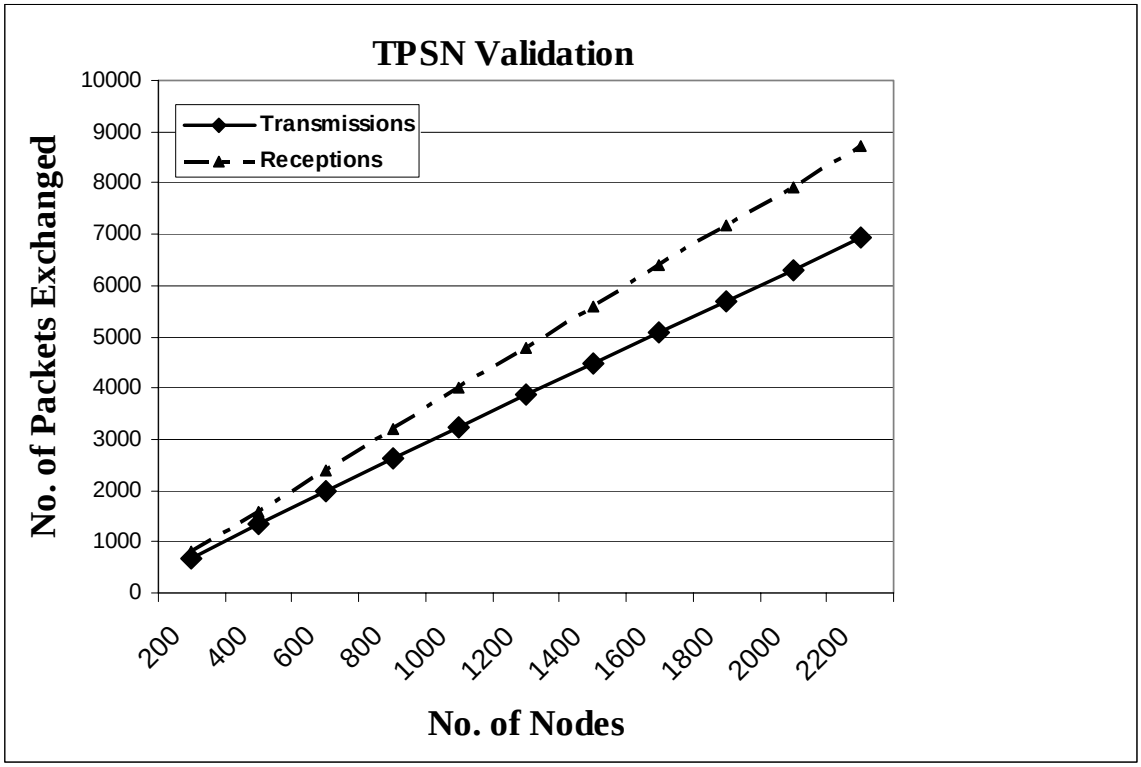


Figure 8: RBS Validation Graph

5.6.2 TPSN Validation

The first sets of simulations were run to validate basic equation of TPSN. In this experiment, all other parameters are kept constant: We simulated our algorithm in Matlab

and tested it over a network area of 1000m x 1000m which is randomly populated with 200 sensors, which are then varied up to 2200 in increments of 200. Our reported data averages over a set of 20 independent runs for each configuration. The TPSN algorithm is then executed for each value and a graph is plotted in Excel. The figure below shows how the change in number of nodes affects the number of transmissions and reception of RBS



5.6.3 Verification of Improved Techniques

It is important to compare the number of transmissions and receptions amongst algorithms to determine the total energy expended when synchronizing the network. Since the transmitted packets are the same size for both the algorithm, the energy consumption is calculated as follows [4]:

$$Energy = numTx + (numRx)*(RX/TX\ ratio) \quad (6)$$

5.6.4 EERBS

We tested our algorithm over a network area of 10m x 10m by taking samples of 10 synchronizations. In our simulation results, we compare the performance of our proposed technique against original RBS protocol.

We compared the following metrics of both the protocols

- Number of transmissions
- Number of receptions
- Overall energy consumption of the network

5.6.4.1 Results and Analysis

In each set of simulations, number of nodes was varied from 10 up to 100, in increments of 10. For each simulation of EERBS 10 percent of the total nodes could beacon as a defining criterion.

Since in RBS all the nodes beacon, the number of transmissions and receptions increase exponentially (see equations 3 & 4). The simulation results show that the proposed protocol provides considerable reduction in the number of transmissions and receptions (see figures below)

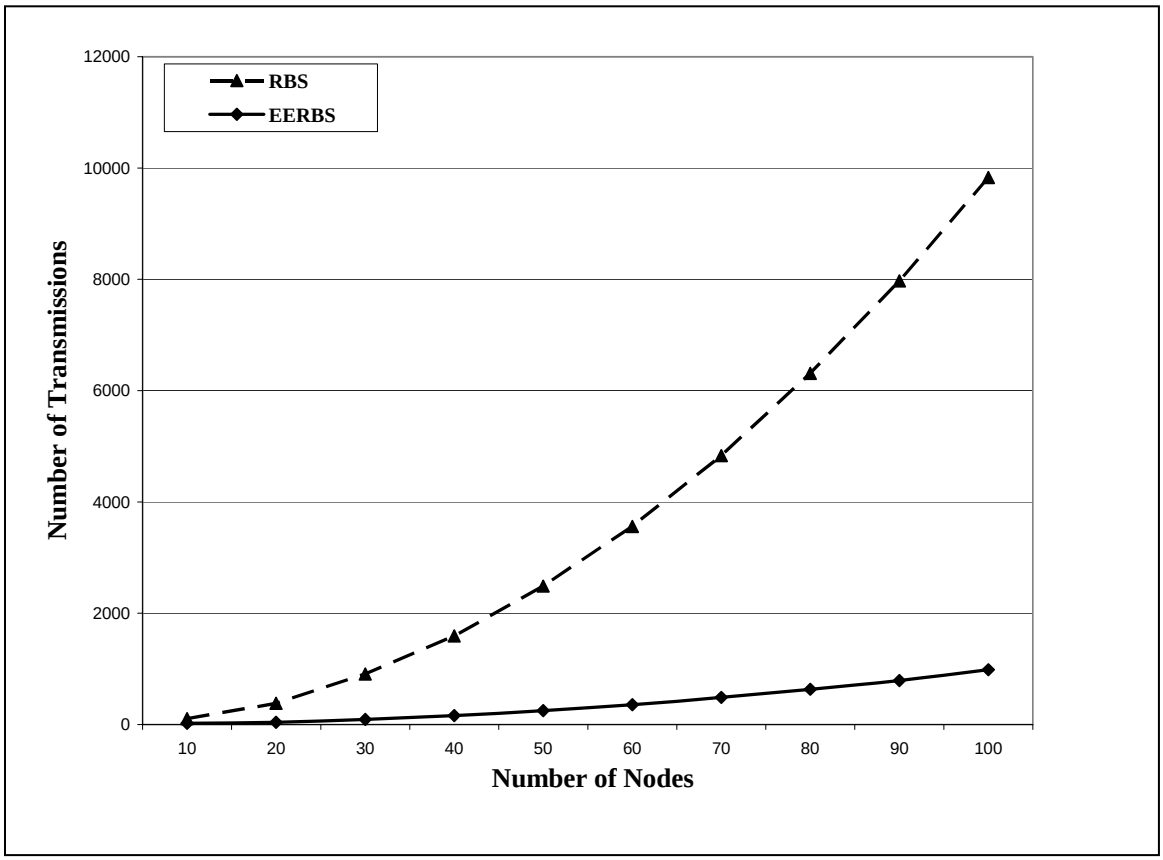


Figure 9: Comparison of number of Transmission of RBS and EERBS

Table 1: Number of Transmission in RBS and EERBS

Sensors	10	20	30	40	50
RBS	107.5	379	907.4	1592.9	2486.4
EERBS	20.8	41.1	91	160	248.1
EERBS Saving %	**80.65**	**89.16**	**89.97**	**89.95**	**90.02**

Sensors	60	70	80	90	100
RBS	3560.2	4832	6311.9	7972.4	9830.4
EERBS	355.5	485.1	633.3	788.7	982.7
EERBS Saving %	**90.01**	**89.96**	**89.97**	**90.11**	**90.00**

Table 1 shows the average number of transmissions of both RBS and EERBS. The results show a quadratic rise in the number of transmission of RBS with the increase in network size.

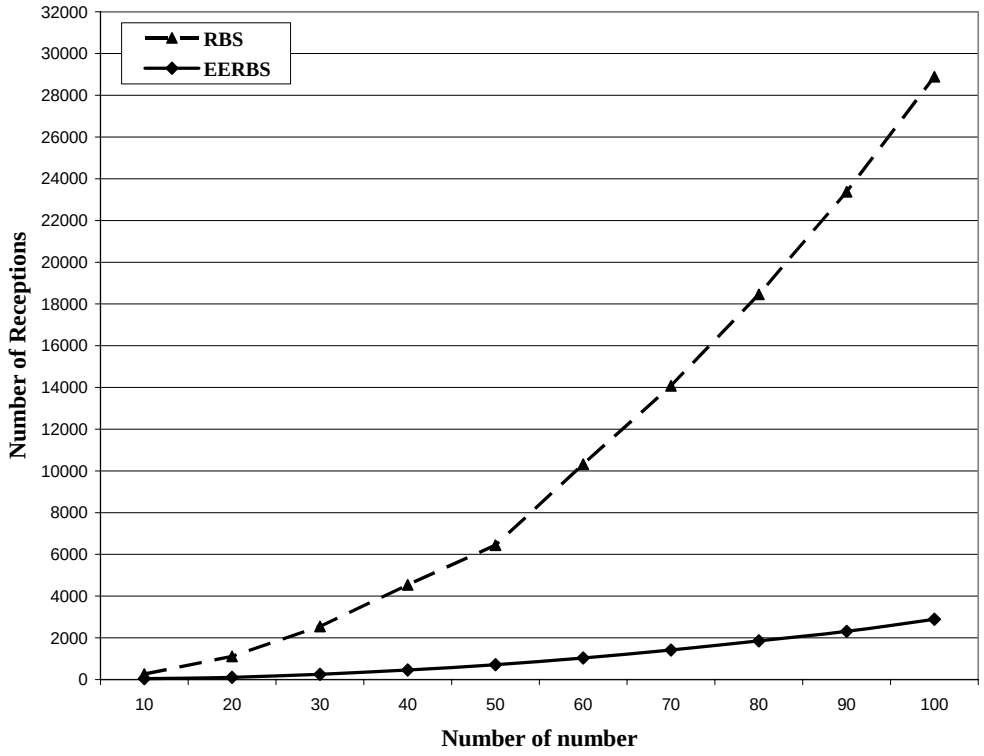


Figure 10: Comparison of Number of Receptions of RBS and EERBS

Table 2: Number of Receptions in RBS and EERBS

Sensors	10	20	30	40	50
RBS	262.5	1110.3	2542.2	4538.7	6443.49
EERBS	50.4	111.3	254.7	456	714.3
EERBS Saving %	**80.8**	**89.98**	**89.98**	**89.95**	**88.91**

Sensors	60	70	80	90	100
RBS	10320.6	14076	18455.7	23377.2	28891.2
EERBS	1030.5	1413.3	1851.9	2312.7	2888.1
EERBS Saving %	**90.02**	**89.96**	**89.97**	**90.10**	**90.00**

Table 2 shows the average number of receptions of both RBS and EERBS. The results show a quadratic rise in the number of transmission of RBS with the increase in network size.

Transmissions and receptions, consume a lot of energy, RBS shows greater energy consumption as compared to EEBRS (see figure 4). This is because number of receptions and transmission of RBS is greater than EERBS.

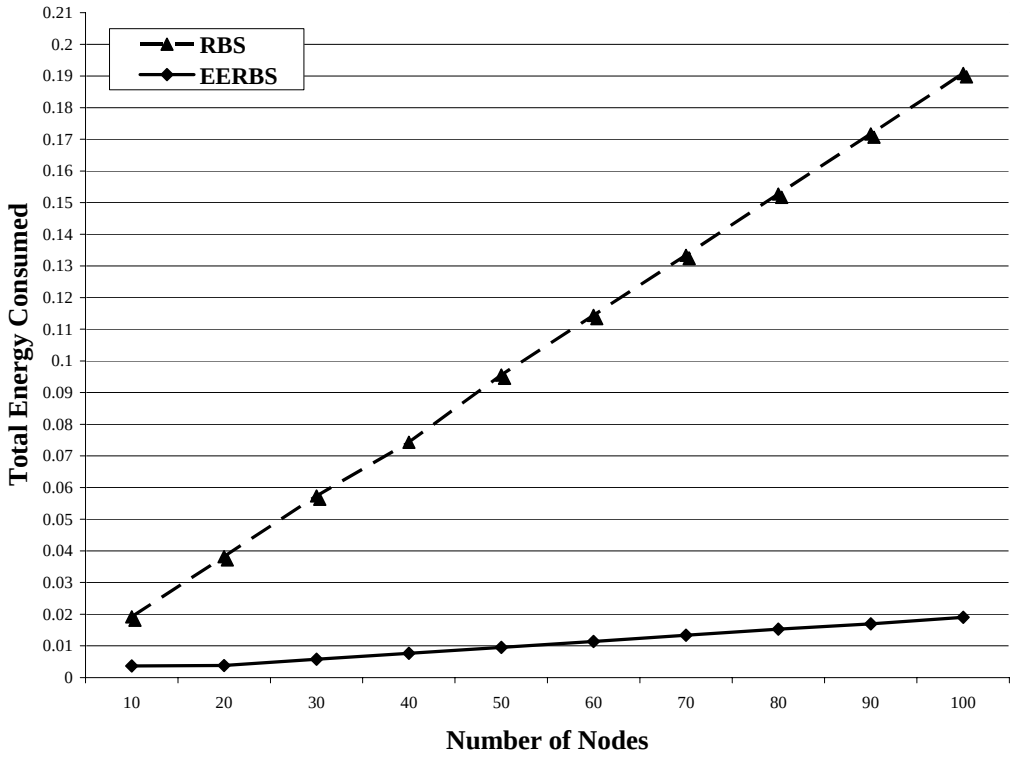


Figure 11: Total energy consumption of RBS and EERBS

Table 3: Total energy consumption of RBS and EERBS

Sensors	10	20	30	40	50
RBS	0.01914	0.0382	0.05735	0.07432	0.09554
EERBS	0.00366	0.00381	0.00576	0.00764	0.00953
EERBS Saving %	**80.88**	**90.03**	**89.96**	**89.72**	**90.02**

Sensors	60	70	80	90	100
RBS	0.11437	0.13338	0.15272	0.1717	0.19077
EERBS	0.01141	0.01339	0.0153	0.01699	0.01905
EERBS Saving%	**90.02**	**89.96**	**89.98**	**90.10**	**90.01**

5.6.5 EETPSN

In our simulation results, we compare the performance of our proposed technique against original TPSN protocol.

We compared the following metrics of both the protocols

- Number of transmissions
- Number of receptions
- Total energy consumption
- Synchronization Time

5.6.5.1 Results and Analysis

We compare the complexities of both algorithms by formulating them mathematically. Equation 7 - 10 depicts the formulae for number of transmissions (Tx) and receptions (Rx) occurring between a source node and its children in a single hop and Equation 11 - 12 shows them for multihop.

Number of Packet Exchange for Single hop

$$TX_{TPSN} = 1 + 3\ (n - 1) \quad (7)$$

$$RX_{TPSN} = 4\ (n - 1) \quad (8)$$

$$TX_{EETPSN} = 1 + 2(n - 1) \quad (9)$$

$$RX_{EETPSN} = 3\ (n - 1) \quad (10)$$

Where n is number of child nodes

Number of Packet Exchange for Multihop

for 0 to L-1

$$Tx_{EETPSN} = \sum_{L=0}^{l-1} B^L\ (Tx) \tag{11}$$

$$Rx_{EETPSN} = \sum_{L=0}^{l-1} B^L (Rx) \tag{12}$$

where *B* refers to number of children e.g. 2 for a binary tree.

levels *L* are the stratum in the hierarchical structure

Corresponding *Tx* and *Rx* are computed in *equation (7-10)*

We validate these complexities for number of transmissions and receptions against implementation simulating real time network where *N* nodes are synchronizing. Mathematically, they are justified in ideal conditions. Nevertheless, due to several constraints of WSN like network dynamics and disconnections, some nodes might be left as orphan nodes and flooding of the network for level discovery may form a different structure with different number of child nodes at average. For such scenarios, total number of communications occurring in the process may vary for same number of total nodes. Simulation results produce following graphs as shown in following figures where with the increase in number of nodes, energy decreases. Transmissions consume more energy than receptions, so we gauge energy in normalized form with equation 6, with a certain reception to transmission ratio for the particular technology used.

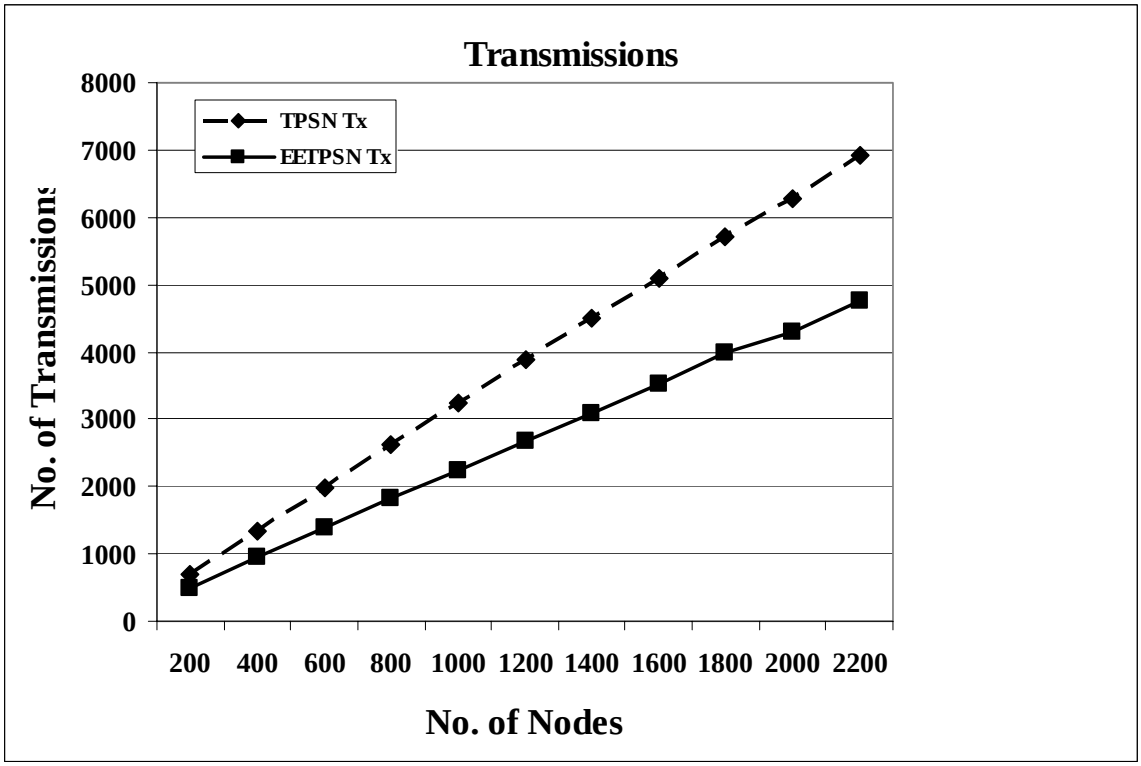


Figure 12: Comparison of Number of Transmissions of TPSN and EETPSN

Table 4: Number of Transmissions in TPSN and EETPSN

Sensors	200	400	600	800	1000
TPSN	683.15	1341.2	1979.55	2614.8	3248.2
EETPSN	484.9	942.25	1380.55	1815.8	2249.2
EETPSN Saving %	**29.02**	**29.74**	**30.26**	**30.56**	**30.76**

1200	1400	1600	1800	2000	2200
3878.8	4493.6	5100.85	5706.9	6283.7	6921.529
2679.222	3096.444	3531.182	3975.833	4300.6	4746.667
30.93	**31.09**	**30.77**	**30.33**	**31.56**	**31.42**

Table 4 shows the average number of transmissions of both TPSN and EETPSN. The results show 30 % savage of energy consumption.

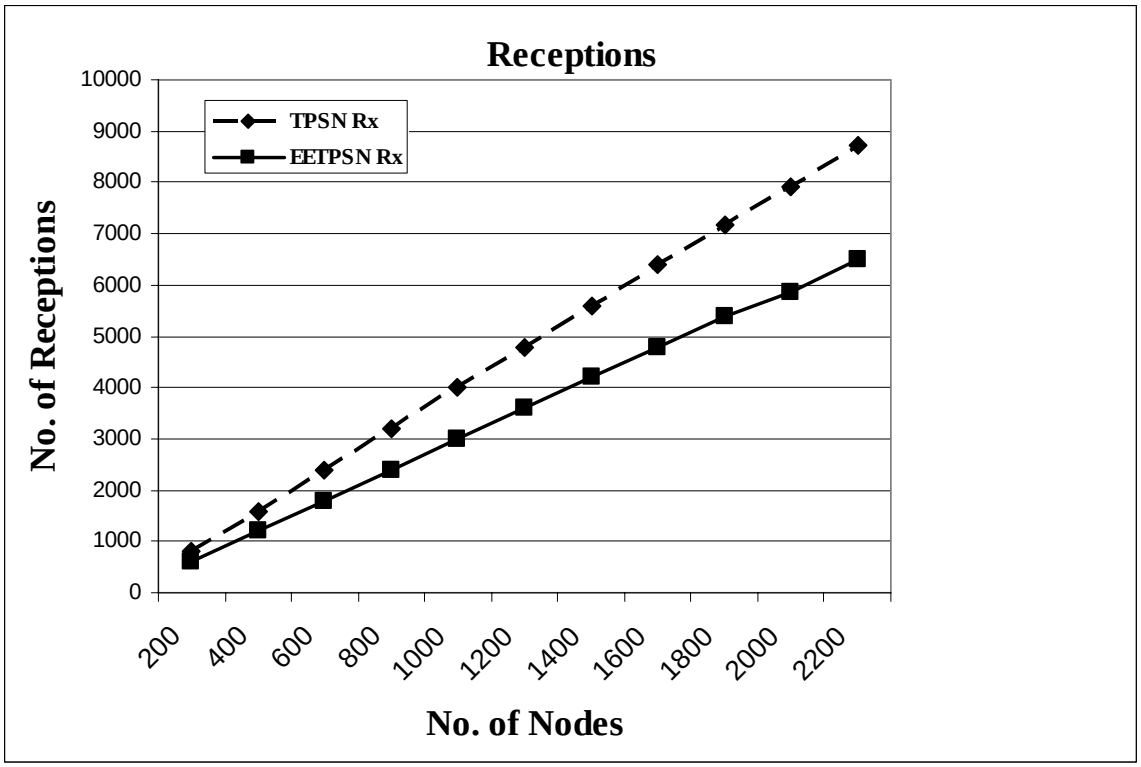


Figure 13: Comparison of Number of Receptions of TPSN and EETPSN

Table 5: Number of Receptions in TPSN and EETPSN

Sensors	**200**	**400**	**600**	**800**	**1000**
RBS	793	1595.8	2396	3196	3996
EETPSN	594.75	1196.85	1797	2397	2997
EETPSN Saving %	**25**	**25**	**25**	**25**	**25**

1200	**1400**	**1600**	**1800**	**2000**	**2200**
4796	5594.2	6385.7	7155.95	7915.3	8731.706
3597	4197	4797	5397	5875.2	6502
25	**24.98**	**24.88**	**24.58**	**25.77**	**25.54**

Similarly, there is a vivid improvement in synchronization speed for the new algorithm. It takes lesser time to synchronize the network as shown Figure

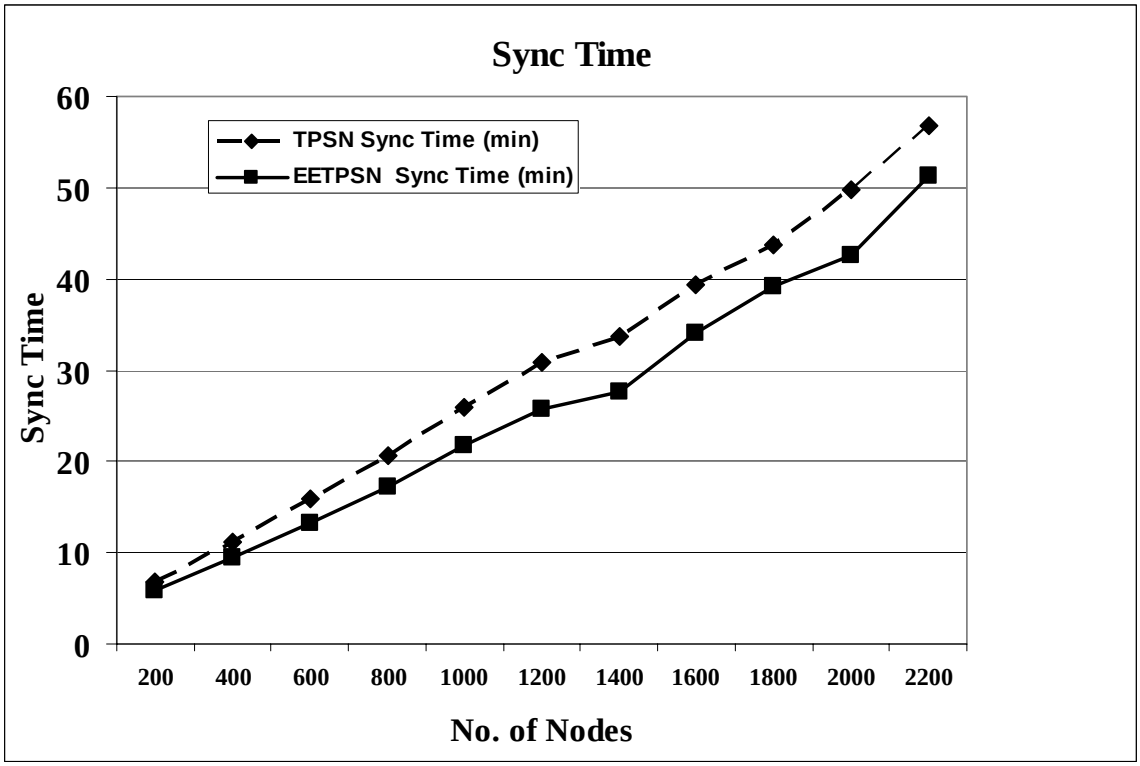


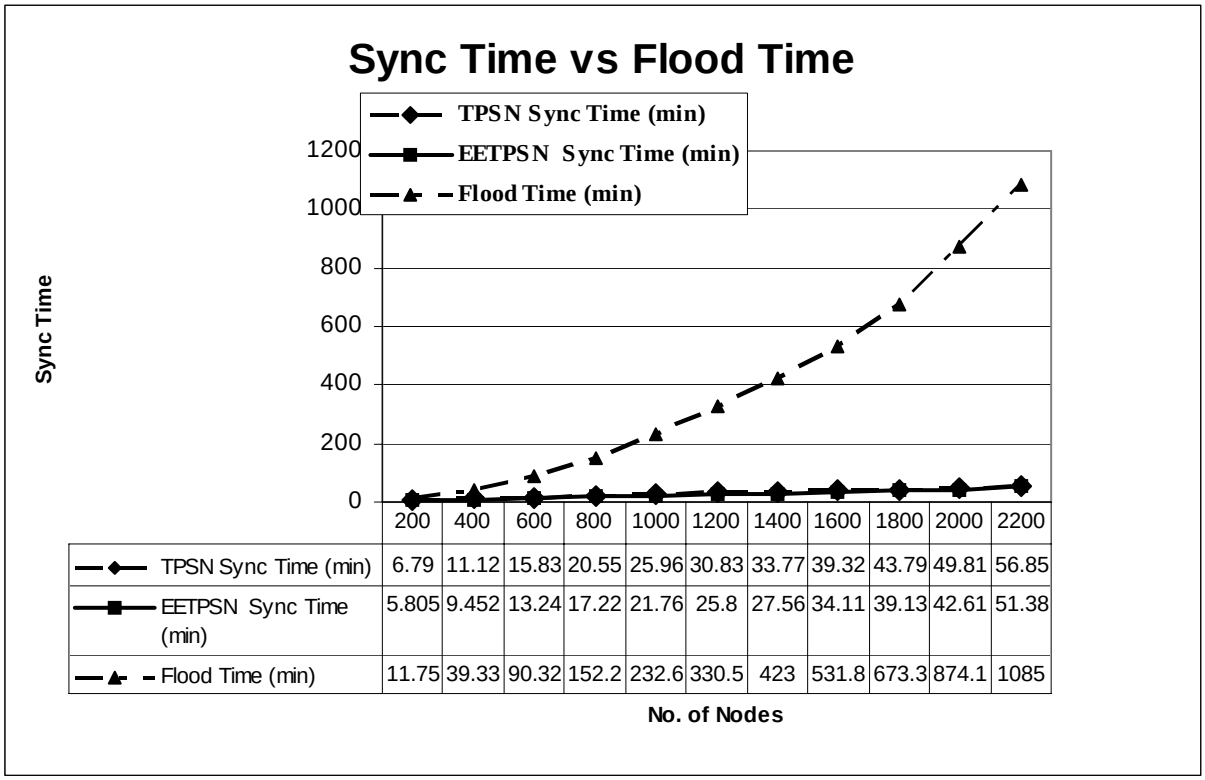
Figure 14: Average Synchronization Time for TPSN and EETPSN

Table 6: Synchronization Time in TPSN and EETPSN

Sensors	200	400	600	800	1000
TPSN	6.789855	11.11564	15.82971	20.55469	25.96486
EETPSN	5.80548	9.45236	13.23986	17.21587	21.75548
EETPSN Saving %	**14.50**	**14.96**	**16.36**	**16.24**	**16.21**

1200	1400	1600	1800	2000	2200
30.8344	33.76642	39.31564	43.79298	49.80925	56.84561
25.80036	27.56338	34.11081	39.12763	42.61252	51.375
16.33	**18.37**	**13.24**	**10.65**	**14.45**	**9.62**

It consumes significant magnitude of time to flood the network for level discovery. It is a non-linear speed function where synchronizing takes drastically lesser time then flooding as shown in Figure. Combining the synchronizing request with flood packets, we save some more transmissions and time in configuration.



	200	400	600	800	1000	1200	1400	1600	1800	2000	2200
TPSN Sync Time (min)	6.79	11.12	15.83	20.55	25.96	30.83	33.77	39.32	43.79	49.81	56.85
EETPSN Sync Time (min)	5.805	9.452	13.24	17.22	21.76	25.8	27.56	34.11	39.13	42.61	51.38
Flood Time (min)	11.75	39.33	90.32	152.2	232.6	330.5	423	531.8	673.3	874.1	1085

Figure 15: Synchronization Time Versus Flood Time for TPSN and EETPSN

Transmissions and receptions, consume a major portion of sensor battery, TPSN shows greater energy consumption as compared to EETPSN as in Figure 14.

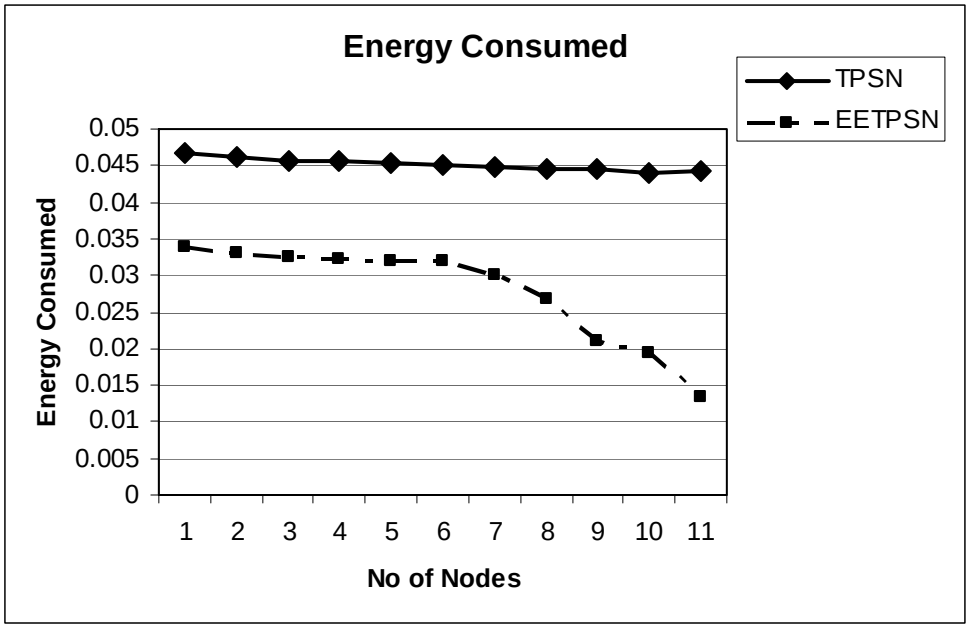


Figure 16: Total energy consumption of TPSN and EETPSN

Table 7: Total energy consumption of TPSN and EETPSN

Sensors	200	400	600	800	1000
TPSN	0.046835	0.04629	0.045765	0.04575	0.04527
EETPSN	0.03378	0.033145	0.03261	0.03229	0.03209
EERBS Saving %	**27.87445**	**28.39706**	**28.74467**	**29.42077**	**29.1142**

1200	1400	1600	1800	2000	2200
0.04511	0.044885	0.04463	0.044435	0.04408	0.044159
0.0319	0.030042	0.0268	0.021122	0.019313	0.0133
29.28397	**33.06872**	**39.95071**	**52.4649**	**56.18761**	**69.88144**

5.6.6 Error Analysis

Uncertainty and the error analysis remain unaffected because we maintain same sender and receiver approach. We have not played around with time stamping capabilities of the original technique.

5.6.6.1 RBS Time Delays

RBS algorithm defines the critical path, which is the portion of the network where a significant amount of clock uncertainty exists. A long critical path results in high uncertainty and low accuracy in the synchronization. RBS improves upon NTP by reducing the length of the critical path, which can improve the accuracy of the synchronization to 7 μs in light traffic. There are four main sources of delays that must be accounted to have accurate time synchronization:

- *Send time:* this is the time to create the message packet.
- *Access time:* this is a delay when the transmission medium is busy, forcing the message to wait.
- *Propagation time:* this is the delay required for the message to traverse the transmission medium from sender to receiver.
- *Receive time:* similar to the send time, this is the amount of time required for the message to be processed once it is received.

5.6.6.2 TPSN Time delays

The Timing-Sync Protocol for Sensor Networks (TSPN) was developed in 2003 in an attempt to further refine time synchronization beyond RBS's capabilities. TPSN uses the same sources of uncertainty as RBS does (send, access, propagation, and receive), with the addition of two more:

- *Transmission time:* the time for each bit to go, be processed and sent through the RF transceiver during transmission.
- *Access time:* the time for each bit to be processed from the RF transceiver during signal reception.

5.6.6.3 Error and Delay

In the synchronization phase of TPSN, bidirectional synchronization is performed between the transmitter and receiver nodes using a 2-way handshake. Given a parent node *A* and a child node *B*, node *A* sends a *synchronization_pulse* to *B*, time stamped at *T1*. Once node *B* receives the pulse, it timestamps at *T2*, then sends a *pulse* packet back to *A* at *T3*. The parent node receives timestamps one last time at *T4*. These 4 timestamps provide estimates for clock drift in equation 13 and propagation delay in equation 14:

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2}, \tag{13}$$

$$d = \frac{(T2 - T1) + (T4 - T3)}{2}. \tag{14}$$

The synchronization error can be calculated from the clock drift between the two nodes as well as the drift at *T4*:

$$Error = \Delta - D_{t4}^{A \to B}. \tag{15}$$

The following equations characterize *T2* and *T4*:

$$T2 = T1 + S_A + P_{A \to B} + R_B + D_{t1}^{A \to B}, \tag{16}$$

$$T4 = T3 + S_B + P_{B \to A} + R_A + D_{t3}^{B \to A},$$

$$T4 \approx T3 + S_B + P_{B \to A} + R_A - D_{t4}^{A \to B}, \tag{17}$$

where S_A, $P_{A \to B}$, and R_B refer to the time to send the packet at node *A* (send time + access time + transmission time), the propagation time between nodes *A* and *B*, and the time to receive the packet at node *B*, respectively.

Equations 16 and 17 can be combined and used in 15 to get the theoretical error for TPSN:

$$Error_{TPSN} = \Delta - D_{t4}^{A \to B} = \frac{S^{UC}}{2} + \frac{P^{UC}}{2} + \frac{R^{UC}}{2} + \frac{RD_{t1 \to t4}^{A \to B}}{2}. \quad (18)$$

By contrast, the error for RBS as claimed by the TPSN authors is:

$$Error_{RBS} = \Delta - D_{t4}^{A \to B} = P_D^{UC} + R^{UC} + RD_{t1 \to t4}^{A \to B}. \quad (19)$$

In equations 18 and 19, S^{UC}, P^{UC}, and R^{UC} refer to the uncertainty in the send time, propagation time, and receive time respectively. *RD* is the relative drift between nodes *A* and *B* from time *T1* through *T4*.

6 Conclusion and Future Work

Our improvement successfully overcomes energy consumption largely in time synchronization by reducing communication traffic of those messages that may be otherwise eating away the network power unnecessarily, especially when there is a chance to save energy. We achieve reduction of 30 - 70% of traffic for synchronization phase in EETPSN and 80- 90% in case of EERBS. In this research, we have highlighted few design factors that number of transmissions directly affects. Identifying similar issues, we may be able to progress in same line of conservation of energy.

In addition, we implement the basic algorithm with a scalable fault tolerant approach.

As a part of our continuing research, we plan to test our improved algorithms on real time sensors to substantiate our hypothesis where we can scrutinize actual network constraints and parameters, in order to suggest results that are more realistic. We also plan to give a proposition for the EERBS algorithm for a multihop sensor network environment. Moreover a hybrid solution by identifying a receiver threshold can be built when EERBS works for multihop like EETPSN. EERBS excels with lesser number of receivers per transmitters whereas EETPSN is suitable with many numbers of receivers per transmitter. The proposed hybrid algorithm would combine both EERBS and EETPSN and would choose the best algorithm with respect to a certain scenario and environmental circumstances. This would further minimize the total power requirement for synchronizing the network.

Apart from that mobility can be incorporated into our algorithms since the current implementations assume immobile nodes and static topology. Our algorithms also

assume that there is no security threat and no malicious code tampers with local clock time. Security can be also incorporated into our algorithms as a further extension of our work. Similarly many other design issues can be raised step by step to implement the techniques in a generalized fashion for an entire WSN test bed.

1. 1a. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. “Wireless Sensor Networks: A Survey” IEEE Computer Networks, vol. 38 no. 4 pp. 393–422, March 2002.
2. Jeremy Elson, Lewis Girod, and Deborah Estrin, “Fine-grained network time synchronization using reference broadcasts”, In Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), December 2002.
3. Saurabh Ganeriwal, Ram Kumar, Mani B. Srivastava “Timing-sync protocol for sensor networks” SenSys ‘03 pp 138-149 Los Angeles CA USA November, 2003.
4. Jeremy Elson, Kay Romer “Wireless Sensor Networks: A New Regime for Time Synchronization”
5. David L. Mills. Internet Time Synchronization: The Network Time Protocol. In Zhonghua Yang and T. Anthony Marsland, editors, Global States and Time in Distributed Systems. IEEE Computer Society Press, 1994.
6. 6i K. Romer, O. Kasten, F. Mattern “Middleware Challenges for Wireless Sensor Networks” ACM Mobile Computing and Communications Review, vol. 6, no. 4, October 2002.
7. 7i J. Bluementhal, M-Handy, F-Golatowski, M. Haase, D. Timmermann “Wireless Sensor Networks-New Challenges in Software Engineering”

merging Technologies and Factory Automation, Proc. IEEE Conference, (ETFA '03), vol.1 pp. 551- 556, 2003

8. 8i Yang Yu, Bhaskar Krishnamachari, and Viktor K. Prasanna "Issues in Designing Middleware for Wireless Sensor Networks" *IEEE Network* vol. 18 no. 1 pp. 15-21, January/February 2004 January/February 2004
9. 9i Kay Romer and Friedemann Mattern, "The Design Space of Wireless Sensor Networks", Wireless Communications, IEEE, Volume 11, Issue 6, December 2004.
10. Kay Romer, Philipp Blum, Lennart Meier. "Time Synchronization and Calibration in Wireless Sensor Networks in Handbook of Sensor Networks: Algorithms and Architectures", Ivan Stojmenovic (Ed.), John Wiley & Sons, September 2005.
11. Yanos Saravanos," Energy-Aware Synchronization in Wireless Sensor Networks", M.S. thesis, University of North Texas, December 2006.
12. Stephen F. Bush. "Low-Energy Sensor Network Time Synchronization as an Emergent Property", Proceedings of the Fourteenth International Conference on Computer Communications and Networks (IEEE ICCCN), San Diego, California USA, October 17-19, 2005.
13. Kay Romer and Friedemann Mattern, "The Design Space of Wireless Sensor Networks", Wireless Communications, IEEE, Volume 11, Issue 6, December 2004.
14. http://www.ntp.org
15. Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, Stephen Wicker, "Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks," Technical Report UCLA/CSD-TR 02-0013, 2002.
16. Crossbow MICAz Wireless Measurement System, Document Part Number 6020-0060-03 Rev A, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.