MD ABDULLAH AL IMRAN

ATILIM UNIVERSITY 2020

A PRACTICAL TIME SYNCHRONIZATION ALGORITHM FOR LINEAR WIRELESS SENSOR NETWORKS INCLUDING LINK-LAYER PROTOCOL IMPLEMENTATION

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

MD ABDULLAH AL IMRAN

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2020

A PRACTICAL TIME SYNCHRONIZATION ALGORITHM FOR LINEAR WIRELESS SENSOR NETWORKS INCLUDING LINK-LAYER PROTOCOL IMPLEMENTATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

MD ABDULLAH AL IMRAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2020

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

Prof. Dr. Ali KARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Electrical and Electronics Engineering Department, Atilim University**.

Assoc. Prof. Dr. Kemal Efe ESELLER
Head of Department

This is to certify that we have read the thesis A PRACTICAL TIME SYNCHRONIZATION ALGORITHM FOR LINEAR WIRELESS SENSOR NETWORKS INCLUDING LINK-LAYER PROTOCOL IMPLEMENTATION submitted by MD ABDULLAH AL IMRAN and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ali KARA
Supervisor

Examining Committee Members:

Prof. Dr. Bulent TAVLI
Electrical and Electronics Eng., TOBB ETU

Prof. Dr. Ali KARA
Electrical and Electronics Eng., Atilim University

Asst. Prof. Mehmet ÜNLÜ
Electrical and Electronics Eng., TOBB ETU

Asst. Prof. Dr. Özgür ERGÜL
Electrical and Electronics Eng., Atilim University

Assoc. Prof. Dr. Murat KOYUNCU
Information Systems Engineering, Atilim University

Date: January 21, 2020

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : MD ABDULLAH AL IMRAN

Signature :

ABSTRACT

A PRACTICAL TIME SYNCHRONIZATION ALGORITHM FOR LINEAR WIRELESS SENSOR NETWORKS INCLUDING LINK-LAYER PROTOCOL IMPLEMENTATION

Al Imran, Md Abdullah

M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Ali KARA

January 2020, 46 pages

Time synchronization is of tremendous importance in networked systems, especially in distributed network systems namely Wireless Sensor Networks (WSNs). WSNs extensively utilizes the time synchronization aspect for coordination and correlation of distributed entities and events (e.g., data fusion, synchronized sleeps and wake-ups, channel sharing etc.). They are typically designed with very stringent constraints, due to their inherent properties like limited resources of energy, storage, computation and bandwidth combined with the ambient conditions and application specific requirements, making traditional synchronization methods unfeasible. Hence, numerous design parameters including the form factor, precision, processing time and power, memory, overhead traffic, efficiency and scalability has led to extensive research of time synchronization relevant to WSNs over the past decades. In this paper, we investigate the linear WSNs or LWSNs and propose a network wide time synchronization method for a multi-hop linear network along with a robust protocol. Furthermore, the protocol is equipped with data aggregation and simple error handling features. Instead of proposing a new algorithm, we will reuse an existing synchronization algorithm. The system was tested on realistic test-beds. Through experimentation we conclude that increasing the synchronization window increases the synchronization accuracy

with a shorter network lifetime. Moreover, the joint impact of relative offset only, and relative offset and drift on the synchronization accuracy was studied. Tests exhibit a higher accuracy for smaller synchronization windows i.e. use of offset parameter only is sufficient to achieve tolerable levels of accuracy. Needless to say, the energy consumption also can be lowered using offset only during time synchronizations for shorter windows.

Keywords: LWSN, Time Synchronization, Data Aggregation, Link-Layer Protocol, Energy Signature

ÖZ

DOĞRUSAL KABLOSUZ ALGILAYICI AĞLAR İÇİN BİR EŞZAMANLAMA ALGORİTMASI VE LİNK KATMANI PROTOKOLÜN GERÇEKLENMESİ

Al Imran, Md Abdullah

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği

Tez Yöneticisi : Prof. Dr. Ali KARA

Ocak 2019, 46 sayfa

Eşzamanlama, Kablosuz Algılayıcı Ağlar (KAA) gibi dağıtılmış ağ sistemlerinde büyük önem taşımaktadır. KAA'lar dağıtılmış varlıkların ve olayların koordinasyonu ve korelasyonu için eşzamanlamayı yaygın olarak kullanır. Tipik olarak, enerji, depolama, hesaplama ve bant genişliği gibi sınırlı kaynaklar, ortam koşulları ve uygulamaya özel gereksinimlerle birleştirilmiş sınırlı özellikleri nedeniyle, geleneksel eşzamanlama yöntemlerini olanaksız kılabilir. Bu nedenle, form faktörü, hassasiyet, işlem süresi ve gücü, bellek, trafik, verimlilik ve ölçeklenebilirlik gibi çok sayıda tasarım parametresi, KAA'ların eşzamanlanmasında çalışılmaktadır. Bu tez çalışmasında, doğrusal KAA'larda eşzamanlanma çalışılmış ve buna yönelik bütünsel bir protokol geliştirilerek deneysel sonuçlar irdelenmektedir. Geliştirilen protokol, düğümler üzerinden veri toplama ve basit hata işleme özellikleri ile donatılmıştır. Deneysel çalışmalar sonucunda, geliştirilen protokol ve eşzamanlama yönteminin etkinliği irdelenmiş ve ağ ömrünün optimizasyonlar ile arttırılabildiği gösterilmiştir. Bu kapsamda, eşzamanlamadaki kritik parametreler olan ofset ve zamanda kaymaların eşzamanlama doğruluğuna etkisi incelenmiştir. Yapılan testlerde, daha küçük eşzamanlama pencereleri ile daha yüksek bir eşzamanlama doğruluğu sağlanmaktadır. Buna bağlı olarak enerji

tüketimi de optimize edilebilmektedir.

Anahtar Kelimeler: Doğrusal Kablosuz Algılayıcı Ağlar, Eşzamanlama, Veri Toplama, Bağlantı Katmanı Protokolü, Enerji Optimizasyonu

To

My Mother,
The unsung and anonymous hero in my life.
For being my first and foremost teacher. And the unconditional shower of love and support.

My Father,
A strong and gentle soul.
For the unheralded sacrifices, relentless encouragements and being a dad.

My Siblings,
For believing in me even when I didn't have the courage to and tolerating my endless mischievousness.

ACKNOWLEDGMENTS

I would like to express immense gratitude to my supervisor Prof. Dr. Ali KARA, who facilitated this rough journey with his solution-oriented approach.

I shall also thank my close friends and family members for making this adventure colorful and their emotional support.

Furthermore, I thank all the members of the Department of Electrical and Electronics Engineering of Atilim University for helping me throughout the dissertation phase.

This work was partially funded by *The Scientific and Technical Research Council of Turkey (TÜBİTAK)* and *Kortek Corrosion Technologies Co. Ltd* under the Research and Development grant #5160097. The project was a joint venture with Atilim University, located in Ankara, Turkey.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

WSN	:	Wireless Sensor Network
LWSN	:	Linear Wireless Sensor Network
GPS	:	Global Positioning System
GPRS	:	General Packet Radio Service
GSM	:	Global System for Mobile Communications
RTC	:	Real-Time Clock
NTP	:	Network Time Protocol
NODE	:	Sensory end devices in the network
PARENT	:	Node with a greater precedence in the hierarchy
CHILD	:	Node with a lesser precedence in the hierarchy
DoS	:	Denial of Service/Sleep
PROTOCOL	:	A set of rules for efficient communication
Time Synchronization	:	Synchronizing clocks of two or more number of nodes
BROADCAST	:	Flooding of data targeted to all nodes
CONVERGECAST	:	Accumulation of data at a single node
Ad-Hoc	:	A decentralized type of wireless network
CRC	:	Cyclic Redundancy Check
ACK	:	Acknowledgment packet
PIPELINING	:	Parallel transaction of data

CHAPTER 1

INTRODUCTION

Wireless Sensor Networks (WSNs) are composed of distributed sensor nodes communicating in a networked fashion to carry out an intended task. WSNs have become remarkably effective in diverse areas (e.g. remote sensing, military applications, medical systems,smart systems) due the readily available inexpensive but capable hardware components as seen in [1–7]. These accelerated advancements allowed the WSNs to be adopted for resource-constrained application environments ranging from mesh, disc, star and grid topology to linear topology of the distribution of the plurality of the sensors. It is obvious that the implementations are application-specific; each realization of WSN is well-suited to the intended area of use i.e. lead to development of numerous protocols and frameworks. The Linear topology of WSN is one such aspect and is the focus of this paper.

Linear Wireless Sensor Networks (LWSNs) is under the WSNs' umbrella that is specifically used to monitor and control linear structures like roads, bridges, pipelines, tunnels, traffic lights and similar. However, deployment of WSNs requires addressing of design challenges like energy efficiency, scalability, resource constraints, network latency, robustness, harsh environments and security [8–10]. WSNs, like most networked systems, are prone to security threats and need to be protected against. These threats include sophisticated Denial of Service (DoS) attacks like sleep deprivation, barrage, synchronization, replay, packet injection, collision and broadcast attacks [11–14] along with environmental and natural phenomena such as physical destruction, landslides, falling trees and earthquakes. The impact of these aforementioned threats are significantly amplified in the linear topology compared to other topologies as the potential to create a network hole is higher i.e. easier to shorten

the natural network lifetime. Furthermore, the linear-topology applications are susceptible to single point of failure and relatively longer end-to-end delays as presented in [15] and [16]. Therefore, efficient operation of LWSNs is even more important than the traditional WSNs due to these challenges. This means there are multitude of critical nodes that affect the lifespan of the whole network. Although remedies for such design challenges are available with some additive cost [17], which might not be achievable depending on the application. Needless to say, the key unifying characteristic of any WSN is the task of data aggregation through sensing, measuring and communicating. Often the collection of data alone is insufficient and the individual time-stamps are desired. To do so, a network-wide global clock can be satisfied by synchronizing the clocks at each node. Accurate time sources like NTP, GPS, GPRS and RTC are available. These techniques require prior existence infrastructure, consistent network coverage and might be exuberant depending on the application. Throughout the literature, numerous cheaper WSN time synchronization techniques were proposed [18–21], as well as for LWSN [22–27]. The time synchronization aspect of LWSN itself requires independent research and is beyond the scope of this letter.

Having said so, this paper will focus on the abstract level protocol design and rules. Rather than developing a time synchronization algorithm from basis and reinventing the wheel, we will concentrate on tweaking an existing algorithm if possible and implement a protocol on top of the algorithm. Our objectives are as follows:

1. Light-weight protocol design to enable implementing on a low-cost resource-constrained hardware.
2. Capable of achieving network-wide time synchronization with acceptable marginal error.
3. Data collection at the sink/base node in a multi-hop network where the intermediate nodes relay the data.
4. Error reporting and error handling for troubleshooting purposes.
5. A portable design in the context of both software and hardware to promote upgradability and modularity.

This work presents findings of a LWSN design project. In this context, the paper reports design considerations, challenges and implementation details of a protocol well-suited to linear wireless sensor networks (LSWNs). This may help designer to overcome similar problems. The rest of the paper is organized as follows: the literature is reviewed in chapter 2. Chapter 3 presents the proposed design of LWSN protocol featuring time synchronization, data aggregation and error handling. The implemented system, experimental setup and various findings are presented in chapter 4. Finally, the conclusion and future research directions are given in the chapters 5 and 6, respectively.

CHAPTER 2

LITERATURE REVIEW

The entire work can be largely divided into two sections based on the task to be carried out: Time synchronization and data aggregation. The extensive literature review is presented below.

2.1 Time Synchronization in LWSN

The time or clock synchronization field of WSN has seen extensive research due to its paramount importance for distributed networked sensing systems. WSNs extensively utilize the time synchronization service for coordination among sensor nodes (e.g., data fusion, synchronized sleeps and wakeups, channel sharing). WSN protocols and algorithms are, typically, designed with very stringent constraints due to the node's inherent properties like limited resources of energy, storage, computation, and bandwidth. Here, in this paper, we will not visit the available synchronization algorithms as they were mildly described before. Instead, a short yet concise abstract level description of the algorithm utilized will be presented. This is because, the time synchronization aspect requires a standalone research of its own and would deviate the sole purpose of this letter.

The time synchronization methods for LWSN in the literature are rather cumbersome and resource-hungry as proposed in [22–26] with the exception of [27], where a rather straightforward, yet application-specific approach was presented. Of course, accurate time sources like NTP, GPS, GPRS and RTC can be used to keep the nodes in sync, but the operating conditions of WSN nodes are often harsh and cheaper alternatives

are desired. Also, there exists radio silent regions e.g. mountain valleys and cloudy weather conditions which make these technologies void. Therefore, we adopted the algorithm of [28] for use in LWSN as the methodology was fairly implementable and computationally light. The algorithm acquires time stamp from a node via a two-way handshaking mechanism for a predefined number of times to compute both the relative offset and drift. Along with the type and quality of manufacturing, the crystal oscillators of the clock are usually prone to ambient environmental factors like temperature, humidity, pressure, acceleration and vibration, magnetic field, electric field, load, and radiation as discussed in [29]. The sensitivity is quite complex to model and we assume that for a short period of time the oscillator is unaffected to the external influences. Nonetheless, there exists synchronization algorithms [30] that considers and compensates for the impact of temperature. Then the relative offset and relative drift parameters are used to update the local time of the node.

2.2 Data Aggregation in LWSN

Regardless of the intended area and task of any WSN, there always is some flow of data in the form of commands or sensor measurements. These data are collected for monitoring, control and analysis of certain events. Literature review shows only a handful of researches dedicated to LWSN, most notably [31–33]. Nonetheless, data acquisition itself demands independent research as can be seen in [34–36]. The dimension of the problem becomes much more complex when multimedia data are to be collected [37–40].

In this paper, we will attempt to develop an easy and intuitive data aggregation technique for LWSN, assuming that the generated data are small in size e.g. voltage, current, temperature, humidity and similar measurements. Even though the process of data collection might seem straightforward in the first glance, it requires careful design decisions. Data acquisition can be categorized into: ad-hoc or converge-cast and mobile data collection. The converge-cast, inverse of broadcast/flooding, is when the data is relayed through intermediate nodes to the base station. On the other hand, mobile data accumulation requires a mobile node traveling around the data points to collect the data. We assume that the LWSN is to run through harsh and rather remote

areas making human intervention quite difficult. Therefore, we focus on the ad-hoc method. This method of data collection can be carried out in one of the two fashions: base node requesting for data from the children and secondly, the child nodes relaying data to the base node periodically or whenever ready. The latter option is comparatively more automated than the first but has a higher energy consumption. Moreover, if an intermediate critical node were to retire immaturely, the nodes down in the hierarchy would still be generating data targeted to the base station causing the partitioned network to have a shortened operational lifetime. Additionally, some of the sensor nodes relaying the data forward will be depleted of energy prematurely and create hole(s) in the linear network. Therefore, this automated form of data collection is not desired and we opted out for the first, on-demand data collection technique.

The on-demand data collection method requires the base node to generate periodic or user-triggered requests to the child nodes. In other words, the periodicity of some events are assumed to be known beforehand or the occurrence of such events is sparse. Since the nodes have a hierarchical relationship between them, the data can be collected using a fixed payload size or a variable payload size. The fixed length data would mean each of the nodes will send its own data first and then relay its child's once its data has reached the sink node. The variable payload allows the nodes to append incoming data to its own data and forward. As a result, the packet size would be increasing linearly for larger data sizes. Here, the trade-off between the fixed and variable packet length is the speed versus energy depending on the payload size. We assume that the effective data is fairly small with a fixed small header and choose the variable packet size. In this case, the packet length must be attached into the header but this is a small price to pay for the greater savings.

CHAPTER 3

PROPOSED METHOD

The proposed LWSN methodology is discussed in details in this chapter. Therefore, the system structure, protocol design and rules, and algorithmic flow are explained here.

3.1 System Model

The system components of Linear Wireless Sensor Network (LWSN) will be described here. Later, this model will be used to propose the necessary algorithms. Considering our model of LWSN topology where the nodes are allowed to communicate while preserving the hierarchical order (i.e., only with the immediate parent and child), the traditional distributed WSN architectures are irrelevant. This is because, a broadcast beacon can reach a certain node in exactly one hop in a mesh, disc or star topology but $(N-1)$ hops in a N node linear network. It is assumed that, each node is at least in the vicinity of two other nodes, if any. Also, each node is only allowed to directly communicate with its immediate neighbors. The hierarchical relationship between the nodes ensures the flawless communication of the nodes. Furthermore, the explicit structure allows identification of the complex system pieces and the modularization eases maintenance and updating of the system. The system is depicted on the Fig. 3.1.

The figure 3.1 illustrates the linear WSN model employed in this work. Referring to the figure, the first node is considered as the root node acting as the base/sink node. There exists a hierarchical relationship between the nodes; Node 1 is the parent

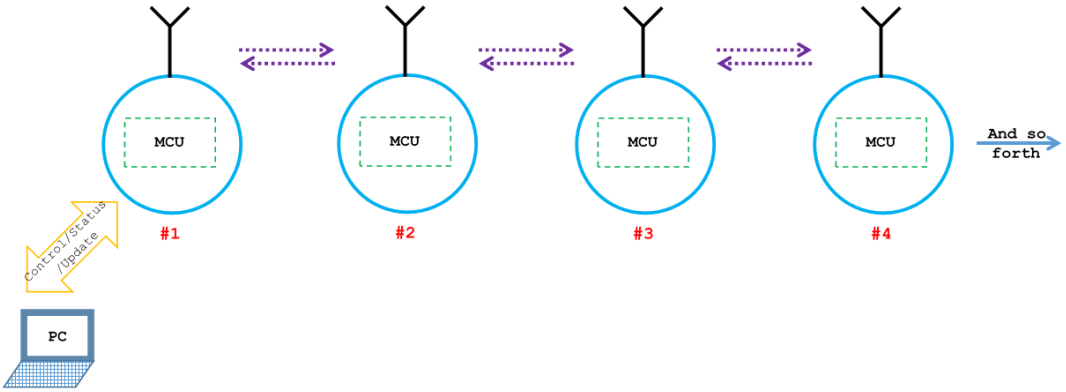

Figure 3.1: Linear Wireless Sensor Network (LWSN) System Model

of Node 2 and grandparent of Node 3 and onward, and so on. This parent-child duo enforces a layering abstract that helps construction of the protocol. Each node is composed of a Microcontroller Unit (MCU) responsible for running the protocol well-suited to the task, a radio device to enable wireless communication, at least a sensor or sensor arrays to take measurements, a power supply possibly in the form of battery with the added exception of the root node which has access to advanced hardware capabilities.

As stated before, the nodes are only allowed to communicate with its neighboring nodes (if any) i.e. satisfied by Eqn. 3.1, where n is the current node ID, N is the set of all nodes.

$$\forall n \in N, \quad \exists n \leq \pm 1 \tag{3.1}$$

The protocol can be subdivided into Time Synchronization, Data Aggregation and Error Handling. These subsections will be described on an abstract-level without loss of functionality in order to refrain from delving into core technical details. Finally, an effective link-layer protocol will be developed and implemented.

3.2 Time Synchronization

Firstly, the time synchronization theory will be studied to present a solid understanding of the techniques used. Afterwards, the algorithmic flow and the flow chart will be presented.

3.2.1 Mathematical Formulation

The timekeeping module is susceptible to clock offsets and drifts due to factors like component quality and aging, ambient conditions like temperature and humidity [29]. Nonetheless, the clock can be modelled [28, 41, 42] as

$$t_{node} = \alpha_{node} + \beta_{node} \times t_{ref} \tag{3.2}$$

where α and β are the offset and drift, respectively, and t_{ref} is the global or universal reference time assuming the clock offset and drift are constant over a period of short observation time. The α term is also referred to as the bias and is subjective to the clock start time. The clock drift, on the contrary, is affected by the surroundings as discussed previously. In fact, Eqn. 3.2 can be modified to account for pairwise synchronization as

$$t_2 = \alpha_{12} + \beta_{12} \times t_1 \tag{3.3}$$

where α_{12} and β_{12} are the relative offset and drift, respectively, and t_1 and t_2 are local time of the two nodes in the context. A two-way handshaking mechanism to pass time-stamps can be used to estimate relative offset and drift at a node as noted in [19].

Refer to the Fig. 3.2 for a visual representation of the said handshaking process where the node initiating the synchronization process is the parent and the other node is the child, both denoted by the superscript. Moreover, the time T_A is the time of transmission, and the time T_C is the time of reception at the origin node while T_B is the time of reception at the target node. This model assumes there are no incurred delays, retransmissions, and packet losses which are far from the reality. The time period where

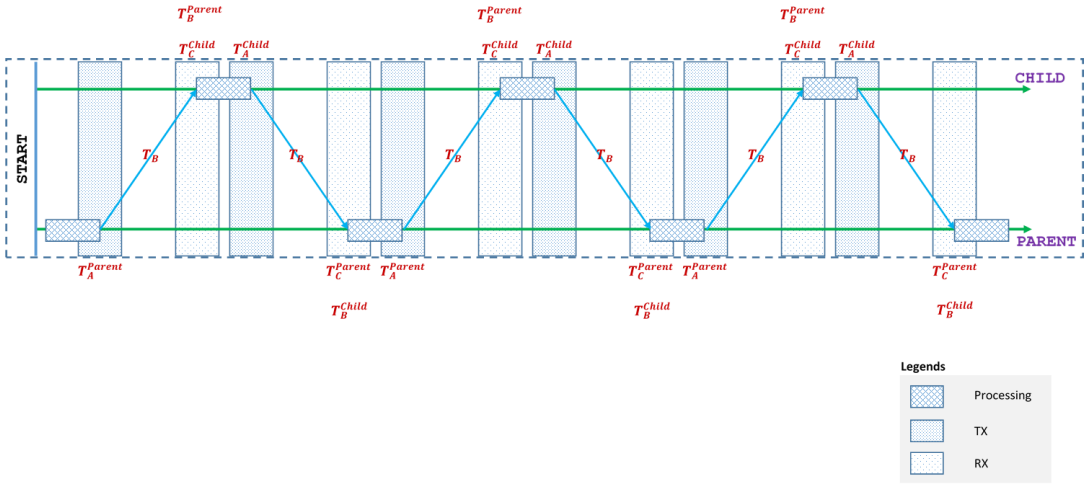

Figure 3.2: Parent-Child Pairwise Handshaking Mechanism

a specific number of two-way handshaking messages are exchanged is referred to as the synchronization window, W. One could compute relative offset and drift terms once ample amount of time-stamps are collected. These time-stamps are then time averaged as of [28] to estimate the relative offset and drift. The upper and lower limits (denoted by the subscripts U and L) of relative drift can be calculated using Eqn. 3.4 and 3.5, respectively. The i on these two equations refers to the index of the sample.

$$\beta_{12U}(i) = \frac{T_A(i) - T_A(i-1)}{T_B(i) - T_B(i-1)} \tag{3.4}$$

$$\beta_{12L}(i) = \frac{T_C(i) - T_C(i-1)}{T_B(i) - T_B(i-1)} \tag{3.5}$$

Rearranging (3.3), the upper and lower bounds of the offset can be calculated using the equations 3.6 and 3.7.

$$\alpha_{12U}(i) = T_A(i) - \beta_{12U}(i) \times T_B(i) \tag{3.6}$$

$$\alpha_{12L}(i) = T_C(i) - \beta_{12L}(i) \times T_B(i) \tag{3.7}$$

The average offset and drift can be estimated by employing the equations 3.8 and 3.9

as

$$\alpha_{12}(i) = \frac{\alpha_{12U}(i) + \alpha_{12L}(i)}{2} \tag{3.8}$$

$$\beta_{12}(i) = \frac{\beta_{12U}(i) + \beta_{12L}(i)}{2} \tag{3.9}$$

Finally, the offsets are time-averaged in order to eliminate the random time dependent delays using the following equations

$$\alpha_{12-AVG}(i) = \frac{1}{W} \sum_{k=i-W+1}^{i} \alpha_{12}(k) \tag{3.10}$$

$$\beta_{12-AVG}(i) = \frac{1}{W} \sum_{k=i-W+1}^{i} \beta_{12}(k) \tag{3.11}$$

Hence, it is obvious from the equations 3.10 and 3.11 that a smaller windows will contribute to faster synchronization while deteriorating the accuracy as lower number of samples is used in the computation of offset and drift. Depending on the required accuracy and application, an optimal synchronization window can be chosen.

3.2.2 Time Synchronization Flow Chart

As opposed to classical WSN topology, the nodes perform pairwise synchronization with its parent. It is assumed that the base node is already synced or has an accurate source of time so that rest of the nodes in the tree can update their clocks with respect to this node. The frequency of network-wide clock synchronization depends on the periodicity of the data collection or event reporting. There exists a clear trade-off between the frequency of synchronization and the network lifespan and requires extensive research. Nevertheless, the pairwise synchronization continues until there are no unsynchronized children. Referring to the Fig. 3.1, a synchronization command generated at the base node (#1) is sent to its immediate child (#2). Upon reception of the sync command, the child performs pairwise synchronization with its parent. The

amount of time-stamp packets requested depends on the application i.e. is run-time programmable.

During the synchronization process, the child node determines the frame counter which is used to determine packet losses and serves as acknowledgement packets. After the second node is synchronized, the parent/base node is notified to be in a lesser power consumption mode or standby mode with a timer running. This timer has a timeout period proportional to the weighted number of nodes in the network. In the meantime, the second node reverses its role and becomes a parent for its child (if any). The whole process repeats until there are no children i.e. when the leaf/last node is reached.

When the very last node has synced with its parent, it notifies the parent that all its children has successfully synced and puts itself to sleep, the most power saving mode. The parent then notifies its parent up in the hierarchy. This continues until the base node is aware of the network-wide synchronization. Intuitively, the child nodes syncs their clock with respect to its parent nodes. Therefore, the synchronization procedure is a top-down approach. Figure 3.3 represents a loosely accurate flowchart of the above mentioned synchronization method. See A.1 in Appendix A for a basic algorithm for pairwise time synchronization.

3.3 Data Aggregation

Data collection requires no mathematical operation whatsoever and is relatively easier to implement compared to time synchronization discussed in section 3.2. The on-demand data collection method requires the base node to ask for data from its children nodes. As considered before, the data can be collection method was selected to be the one with a variable payload size. The variable payload allows the nodes to append incoming data to its own data and forward i.e. a single run is enough for the cumulative data to reach the sink node. It should be noted that the packet length must be attached into the header as the packets arriving and leaving an arbitrary node are dynamic in size. In this paper, we aim to develop a rather primitive data acquisition algorithm since we only aim to demonstrate the functionality of the protocol.

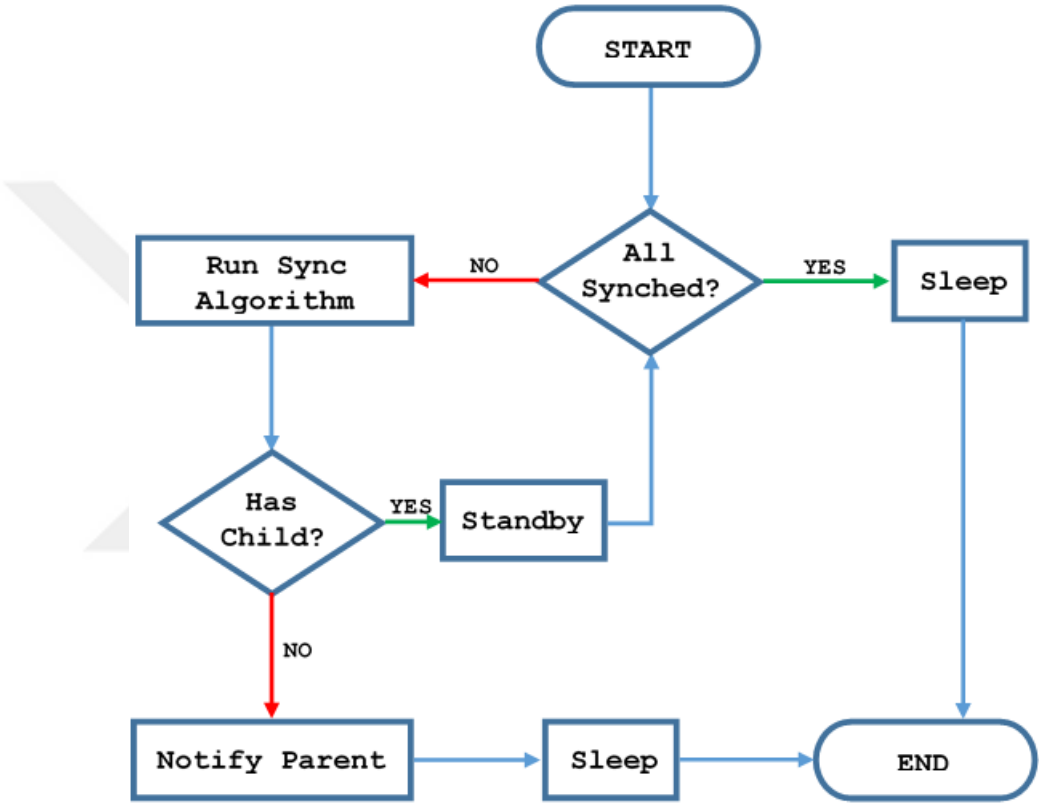

Figure 3.3: Time Synchronization Algorithmic Flow Chart

Unlike the synchronization process, the data aggregation follows a bottom-up hierarchy; once a request is triggered, the data is directed from the last/leaf node and relayed to the next one until the sink node is reached. Of course, the data collection command must first be relayed to the leaf node via the intermediate nodes. The procedure is depicted on the Fig. 3.4, an approximate representation. Upon reception of the command, each node enters into the standby mode and waits for its child to generate the data packet. Given the condition that an event has occurred and proper data are available at each of the nodes, the leaf node will send a packet containing its data as the payload. The leaf nodes goes to sleep upon reception of an acknowledgment from its parent. The parent then appends its data, increasing the payload size and sends to its parent up in the hierarchy. This appending of data in the payload and relaying to the parent is repeated until all the data have reached the sink node. See A.2 in Appendix A for a basic algorithm for data acquisition.

3.4 Error Handling

Like most networked communication systems, WSNs are susceptible to sophisticated threats and attacks [11–14]. Moreover, the radio communication is prone to packet losses and transmission-reception duo problem. Therefore, the WSNs should have some form of counteracting measures to deal with these. On the contrary, it is somewhat difficult to implement a proper security framework due to WSN's inherent properties like limited resources and battery-powered. In the proposed protocol, we integrated a basic error handling feature capable of performing re-transmissions in case of packet loss and reporting the error origin and the type of error.

3.4.1 Error Detection

Consider the following figure on Fig. 3.5 showing transaction between two arbitrary nodes. The numbers alongside the vertical lines represent the frame counter or frame id. This id is used to differentiate sequential requests/responses i.e. packet losses can be identified easily by looking at the frame counter field of the header alone. For example, the parent is requesting some data from the child with a frame ID of 1

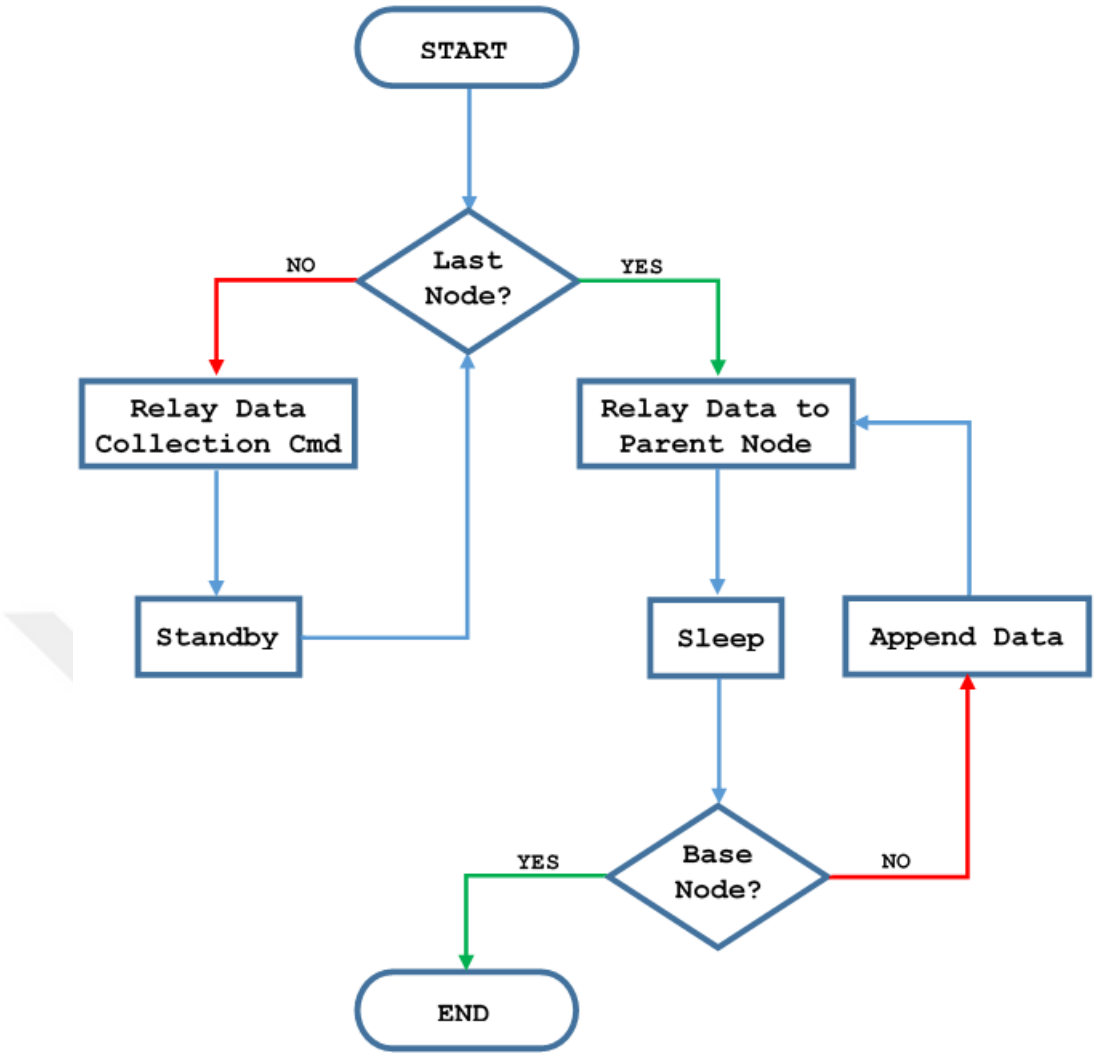

Figure 3.4: Data Aggregation Algorithmic Flow Chart

and expects a response having the same ID. On the child side, upon reception of the request from the parent, a response having the same ID as the request i.e. 1. This process continues until there are no more transactions needed. Meaning, the request generator side is satisfied.

In the scenario presented in Fig. 3.6, we examine a case where the parent node is assumed to be in distress. This implies that the child was not acknowledged of its previous response and the current parent packet failed to reach the child as well. Here, it is assumed that the acknowledgement packet was appended to the next packet to lower the overall overhead. After the timeout occurs, the parent will have to re-transmit the failed packet. Depending on the packet content, the payload might need to be up-

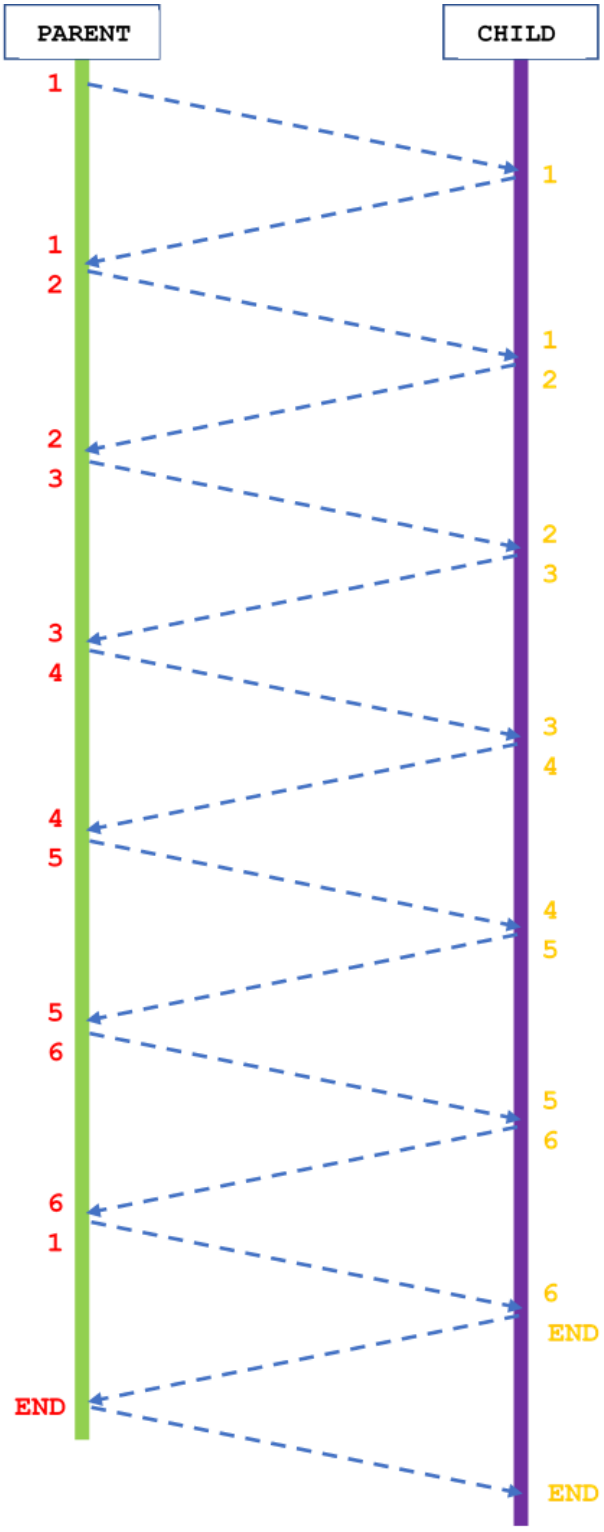

Figure 3.5: Frame Counter for Parent-Child duo (Ideal Scenario)

dated. For example, if the packet was a time-stamp for time synchronization, the time data must be updated but for simple commands and data packets, its not crucial. As shown on this figure, the child re-sends its packet after the timeout period because an acknowledgement was missed. Which side starts the re-transmission is determined by the timeout of the nodes. Furthermore, there has to be some form of retrial limits to avoid exhaustion of reception/transmission and the system being locked at a task.

Similarly, the case of child being in distress is investigated here. Refer to Fig. 3.7 for the visualization of this scenario. In this context, the child is unresponsive to its parent's requests. This also infers that the child didn't receive the acknowledgment of its previous packet. Depending on how the parent side is implemented, the parent might wait for timeout to re-transmit or may go into energy saving state. On the other side, after a timeout, the child re-sends its packet hoping to get a response from the parent. If the parent is never up again, after a pre-specified amount of time, the child goes into sleep.

As a summary, both the parent and the child in a transaction keeps track of each others' frame counter. In a packet loss situation, wait for a predefined amount of time, T_1, before requesting/responding again. Reset the timer T_2 upon successful packet transmission: session resumed. Otherwise restart the session. Note that T_2 is longer than T_1. After the shorter timeout, either of the devices can resume the session.

3.4.2 Error Handling Algorithmic Flow Chart

Having discussed the basic error detection mechanism on section 3.4.1, now the error handling aspect can be constructed. This is pretty straightforward and is illustrated on Fig. 3.8 without loss of generality. Again, the timeout value must be experimentally found and is application dependent. A short timeout would cause the system to halt prematurely while a longer timeout will cause the battery to deplete faster, decreasing the overall operational network lifetime. See A.3 in Appendix A for a basic, yet functional algorithm for error handling.

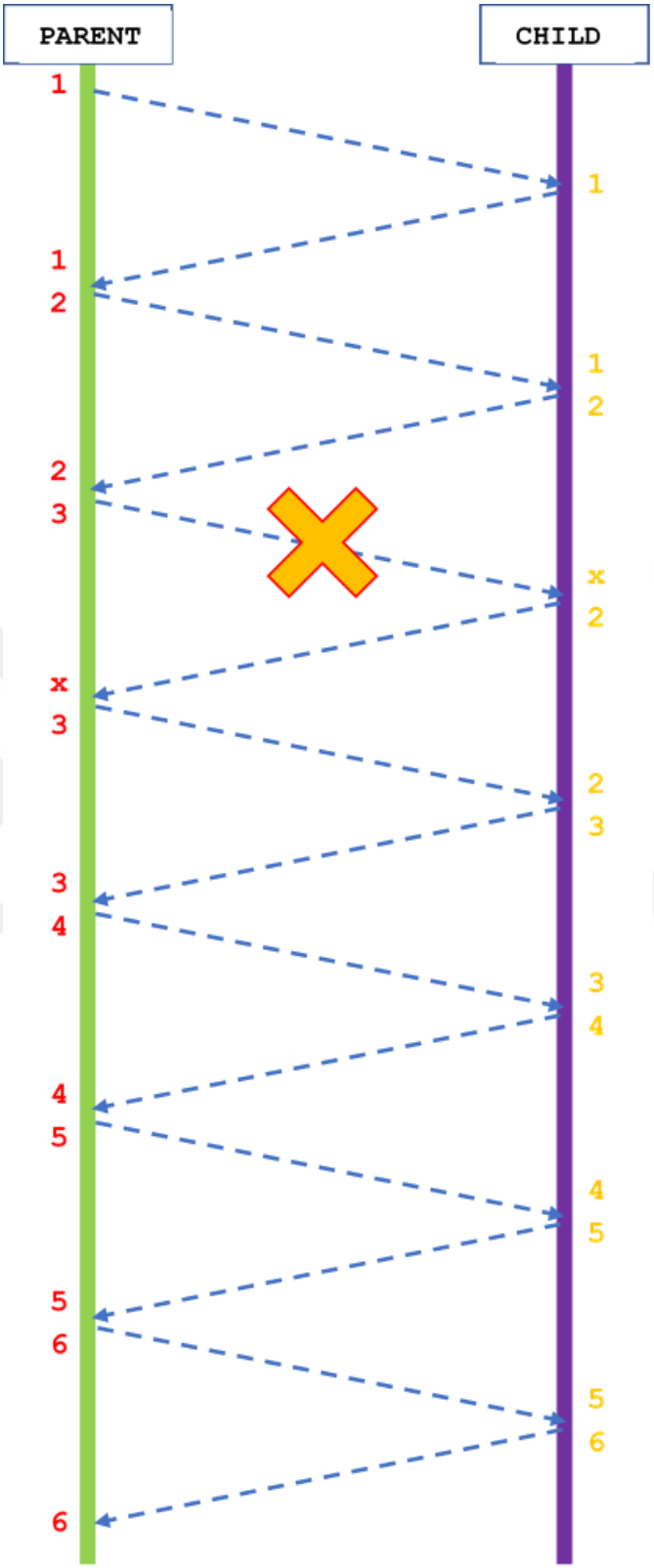

Figure 3.6: Frame Counter for Parent-Child duo (Parent in Distress Scenario)

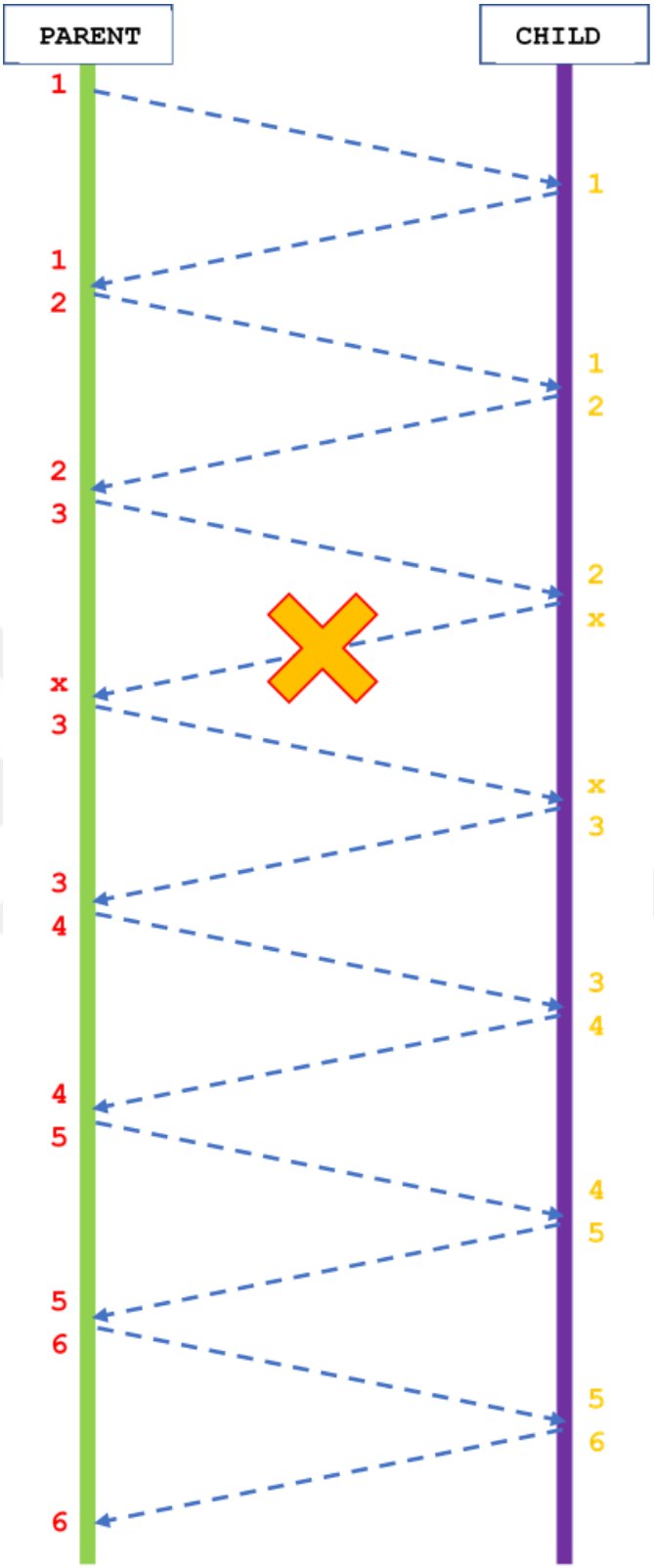

Figure 3.7: Frame Counter for Parent-Child duo (Child in Distress Scenario)

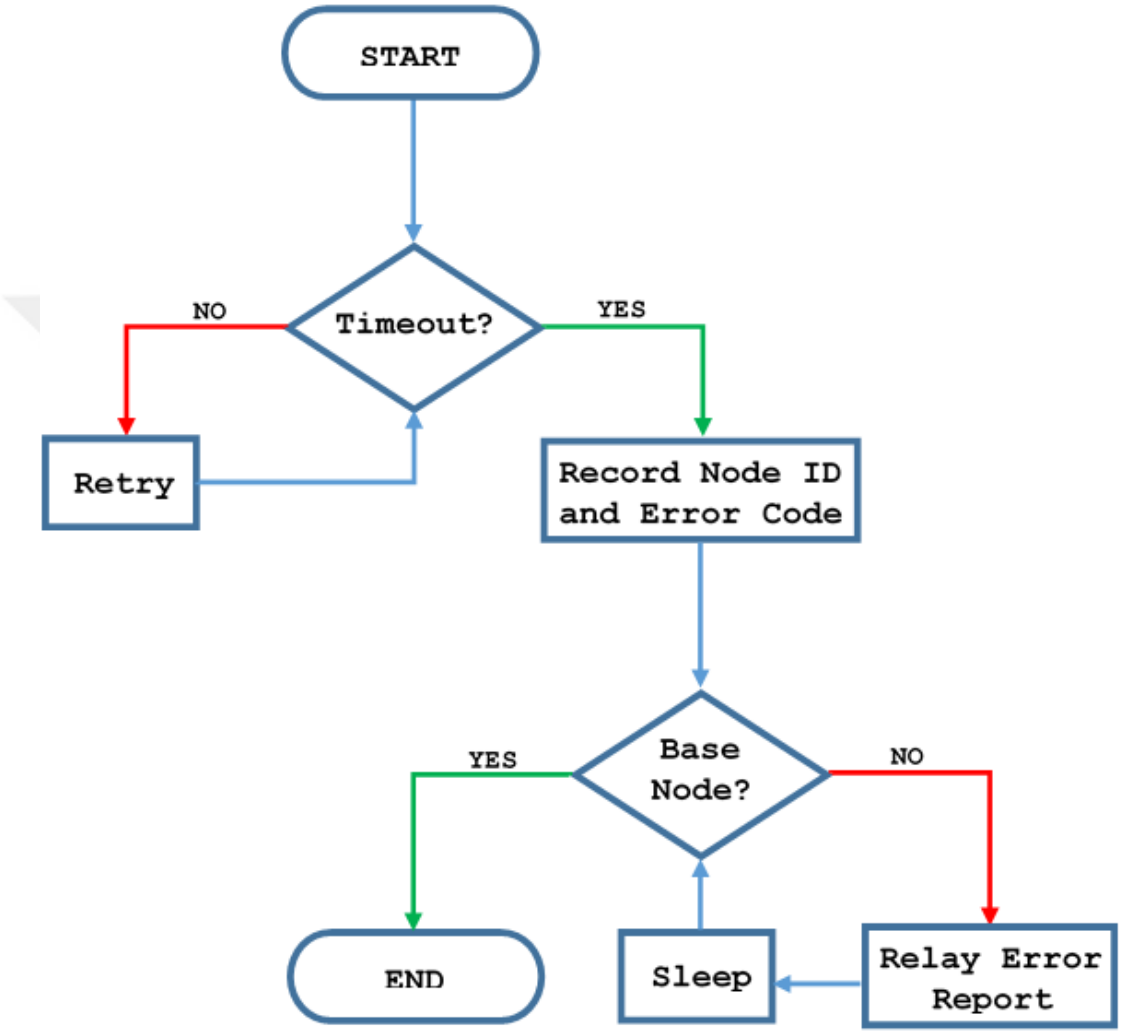

Figure 3.8: Error Handling Algorithmic Flow Chart

CHAPTER 4

EXPERIMENTAL

This chapter presents the findings regarding energy consumption, sleep duration and synchronization techniques. These studies should enable us to benchmark the system performance and come to a conclusion. Also, the system hardware is illustrated.

4.1 System Components

In this section, the hardware composing the system are examined in details and presented. We sought out for cheap hardware solutions the system not only to demonstrate and establish the proof-of-concept, but also to provide a sense of lightness of the software components. Therefore, the microcotroller (MCU) was chosen as the 8-bit AVR RISC-based *ATmega2560* chip from *Microchip*. This due to the chip's readily availability, huge community with software support and a well-established development platform. The communication module was selected to be the cost-effective, high performance *CC1200* RF transceiver from *Texas Instruments* which is well suited for the ISM (Industrial, Scientific, and Medical) frequency bands. These components are enough to get the system up and running. A typical node hardware is depicted on Fig. 4.1, a side view and a cross-sectional view on Fig. 4.2.

Furthermore, an external Real-Time Clock (RTC) can be used for better time keeping accuracy. The low-cost and highly accurate *DS3231* RTC from *Maxim Integrated* is a good option (see Fig. 4.5). To measure the energy consumption of the system, the *INA219B* power monitor chip manufactured by *Texas Instruments* can be utilized. Refer to the Fig. 4.3. Energy measurements often require extended periods of data

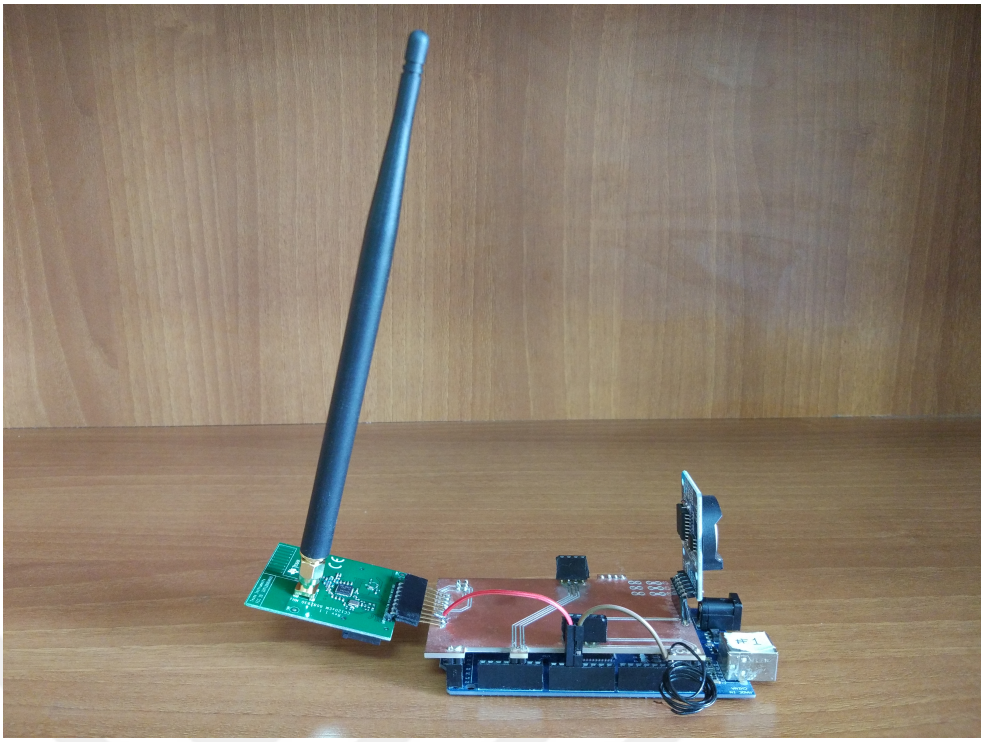

Figure 4.1: Hardware setup of a Node (Side View)

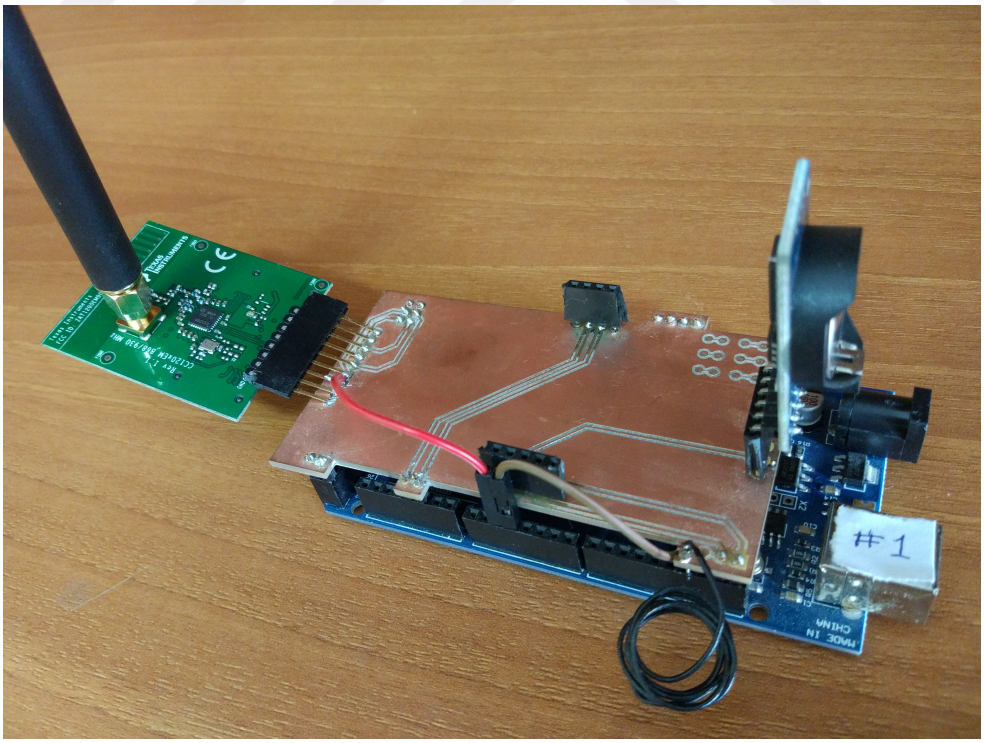

Figure 4.2: Hardware setup of a Node (Cross View)

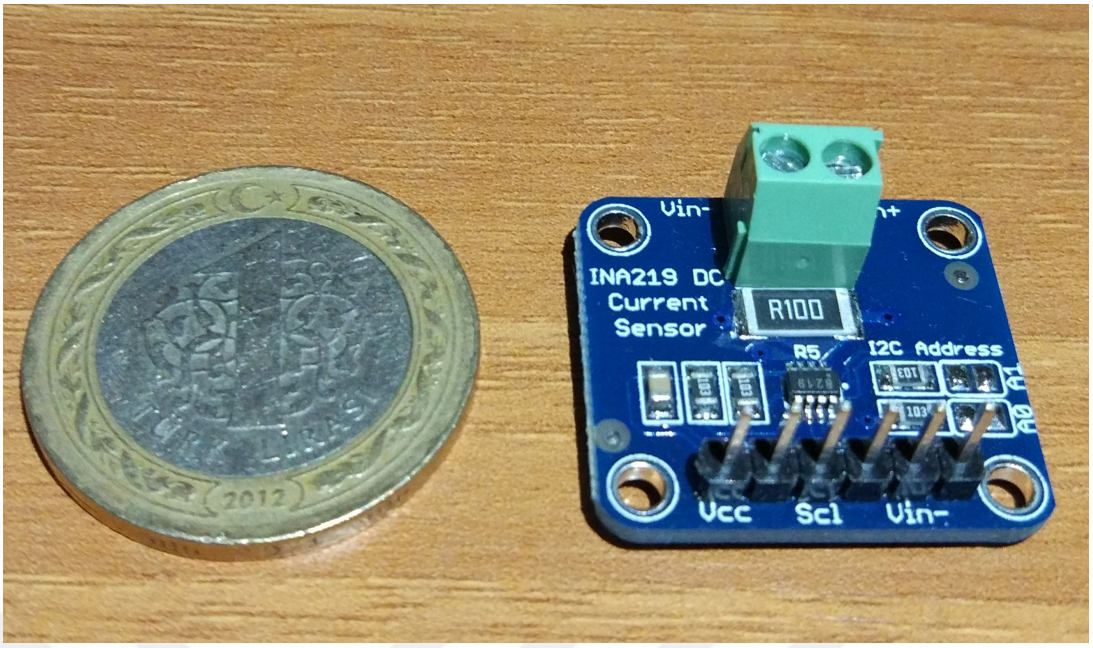

Figure 4.3: Power Monitor Module

collection and logging. The on-chip memory is often not enough and alternatives exist e.g. inexpensive SPI-based generic SD card reader/writer from *RobotDyn* as shown on Fig. 4.6.

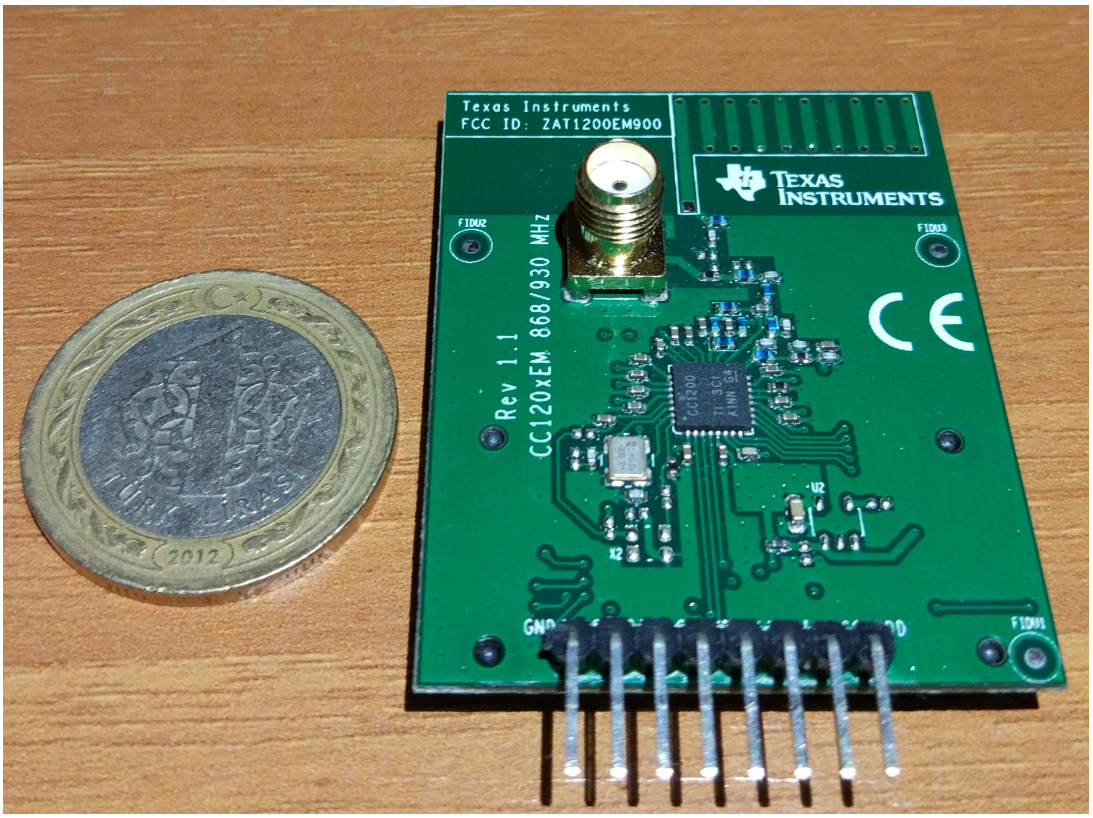

Figure 4.4: RF CC1200 (Without the Antenna)

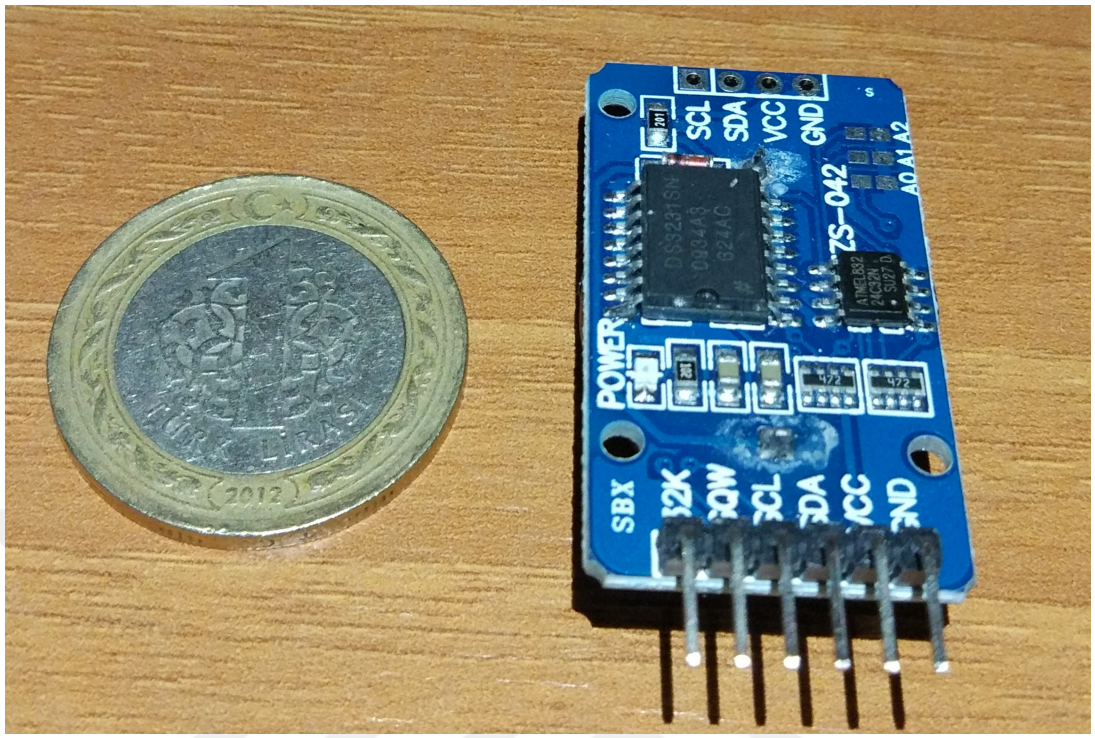

Figure 4.5: Precision RTC Module

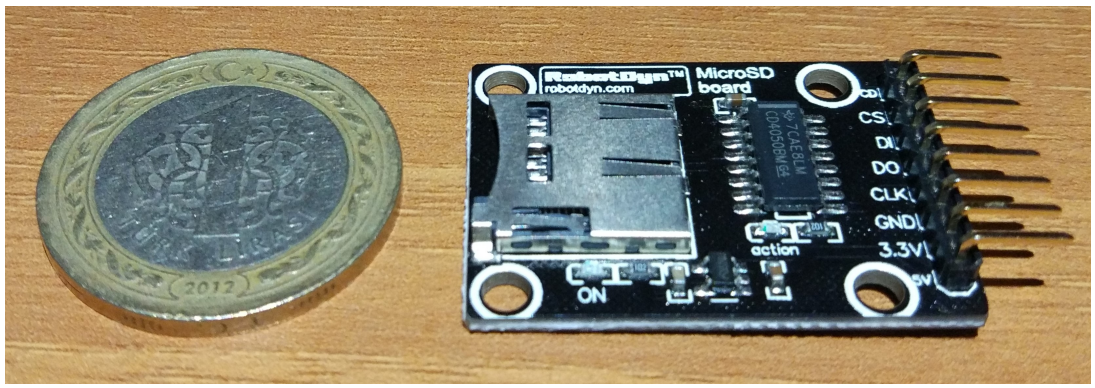

Figure 4.6: Micro SD Card Module

4.2 Energy Budget

Wireless Sensor Networks (WSNs) are often designed with very strict constraints due to numerous reasons like the intended area of use, customer's requirement and etc. One such restriction is the low power requirements thus the system being able to carry out its task for longer periods of time on battery. Therefore, energy estimation is of tremendous importance. Although, energy estimation requires rigorous experimentation, we will not dive deep into the topic and give a general idea about the designed system.

Time synchronization is a widely studied in the area of WSNs to coordinate distributed events. Scheduled wake-ups can be employed to use energy as efficiently as possible. This study explores the energy requirements of time synchronization in WSNs, LWSNs to be specific. In the experiment, it is assumed that no packets are lost during communications i.e. no external interference and all the nodes consume same amount of energy, which is far from the reality.

4.2.1 Theoretical Energy Calculation

Synchronization of two nodes namely some arbitrary Child and its Parent, denoted by t_{CP}, was experimentally found to be 3.729 seconds. This duration consists of all the delays like processing delays, propagation delay and etc. The synchronization window, W, is taken to be 30 which is the amount of time-stamp packets exchanged prior to the calculation of clock offset and drift. All the packets are assumed to have a constant size i.e. $L = 10$ bytes. The radio module was configured to transmit at the rate, R, of 4.8 kbps. The linear tree structure is composed of N nodes. For simplicity, N was taken to be 6. For precision timekeeping a temperature-compensated Real-Time Clock (RTC) was used. Both the radio and the RTC requires 3.3 Volts for stable operation. Moreover, the total synchronization time, T, can be approximated as $(N \times t_{CP})$. Having said so, the total energy of a node during synchronization can be approximated (Eqn. 4.3) using the following equations (4.1) and (4.2).

$$E_{RADIO} = (P_{TX} + P_{RX}) \times t_{CP} + P_{RXs} \times (T - t_{CP}) \tag{4.1}$$

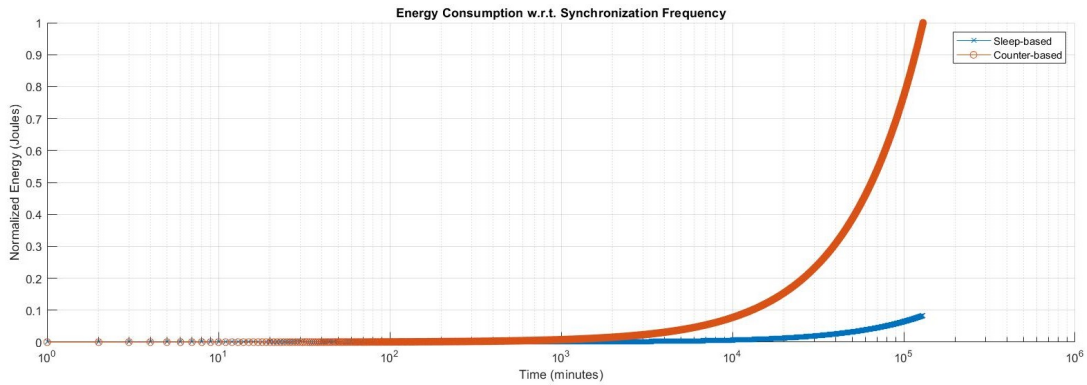

Figure 4.7: Accumulative Energy Consumption in Time Synchronization

$$E_{RTC} = 2(P_{RTCa} \times t_{CP}) + P_{RTCs} \times (T - 2t_{CP}) \tag{4.2}$$

$$E_{SYNC} = E_{RADIO} + E_{RTC} \tag{4.3}$$

On the above equations, P_{TX} and P_{RX} denote the transmission and reception power. The sleep state reception power is symbolized using P_{RXs}. The notations P_{RTCa} and P_{RTCs} are used to define active and sleep state power requirement of the RTC. If we assume that the aforementioned devices are dormant when not trying to synchronize. Depending on the wake-up frequency or intermediate sleep periods T_F, the total energy can be formulated as

$$E_{TOTAL} = E_{SYNC} + (P_{RXs} + P_{RTCs}) \times +(T_F - T) \tag{4.4}$$

This theoretical energy estimate was simulated and illustrated with the Fig. 4.7. As the frequency of synchronization increases, more energy is consumed as expected. The counter based method represents the case where only a single sync was carried out at the very start and then the internal clock (counter) were run to simulate synced clock. Nonetheless, this simulation doesn't take the clock accuracy into consideration.

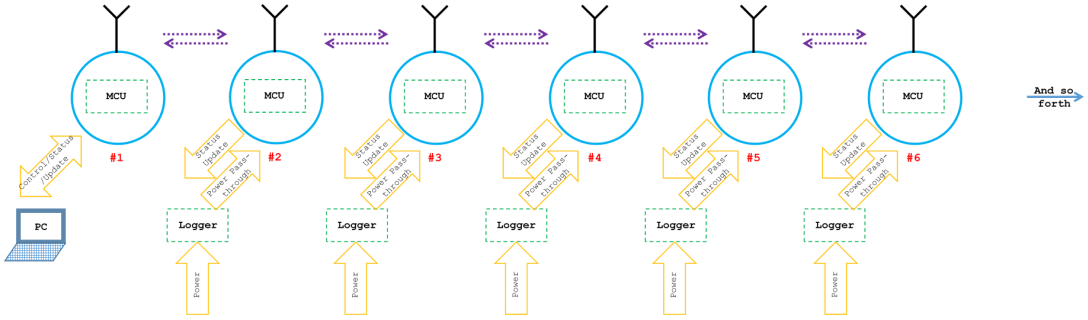

Figure 4.8: Energy Consumption Hardware Setup

4.2.2 Practical Energy Consumption

Although the theoretical calculations give a general idea about the energy demand of this system, a practical approach is vital to reflect the actual scenario. The system model given on Fig. 3.1 can be modified as Fig. 4.8.

In a LWSN composed of 6 nodes, the most energy consumption is expected to be at the second node after the base node. This is because, the synchronization is performed in a top-down manner and hence the second node will wait for all of its children to synchronize with respect to itself. Here, the base or the sink node is omitted as it is assumed to have unlimited power supply and mainly serves the purpose of a gateway for the network to the larger network entity. The network lifetime should be computed according to this node as suggested in [43]: the operational lifespan of WSNs is defined as the interval between the deployment and when the first node is exhausted. Consequently, the second node will be analyzed in the following sections to reflect the notion of system's energy requirement.

Having said so, seven distinct synchronization windows were chosen to exhibit their impact on the node's energy consumption. From this point forward, node refers to the second node unless stated otherwise. The synchronization windows were selected as 5, 7, 12, 16, 20, 26 & 30. Then, network-wide synchronization was carried out multiple times and the average energy consumption was computed at each window. The result is plotted on the Fig. 4.9.

Note that, the above mentioned hardware (section 4.1) were used to measure the energy consumption of the node. It can be seen from the Fig. 4.9, that the average energy

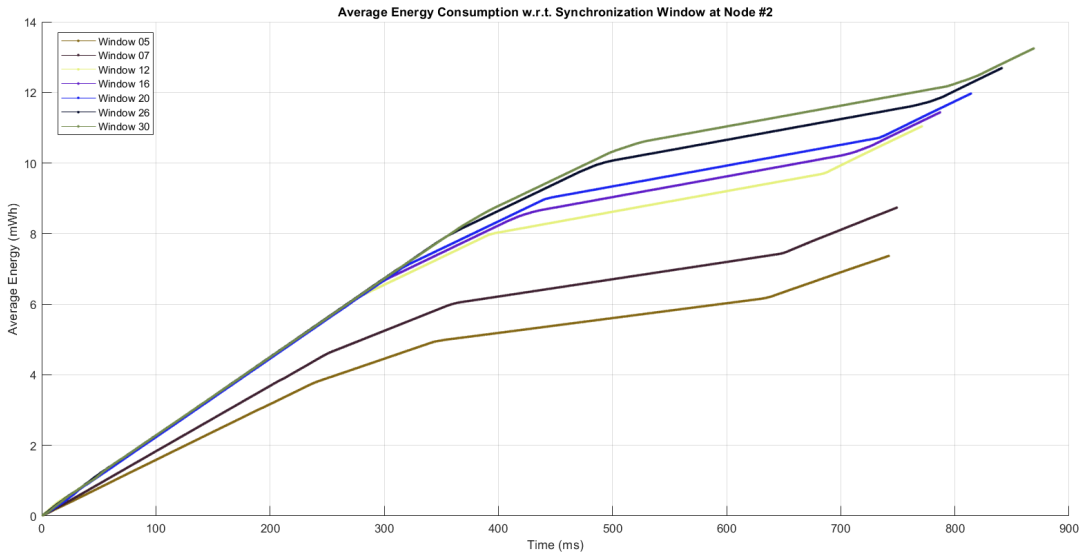

Figure 4.9: Average Energy Consumption w.r.t. Synchronization Window

consumption increases with synchronization window size. The bottom is the energy signature of window 5 and the top is for window 30 with the intermediary windows in-between them in a ordered fashion.

The varying slopes of the energy curve is due to the system being in different states, namely: Start of synchronization, End of local synchronization or Wait for global synchronization, End of global synchronization, Sleep, Wake-up and Stop. These transitions are marked with dotted lines on the Fig. 4.10 respectively. This figure is just for a single synchronization event. The states are self-explanatory. The term local refers to pairwise synchronization of the second node relative to the first node whereas global refers to all the children of second node being synchronized. Reading from the plot, the local sync duration is $(49 - 1) = 4.9s$ seconds and global sync time is $(345 - 49) = 29.6s$. Then the system pauses for $(477 - 345) = 13.2s$ before going to sleep for 30 seconds. The experiment shows that, the second node synced at window size of 28 sleeps for $(771 - 477) = 29.4s$ i.e. there is an error of 0.6 milliseconds. Finally, the system comes to an halt after a few seconds of waking up.

This process is repeated multiple times for each of the sync windows and then the average sleep duration are calculated. The mean sleep time vs synchronization window is given on the Fig. 4.11. The general trend is that the sleep duration approaches the

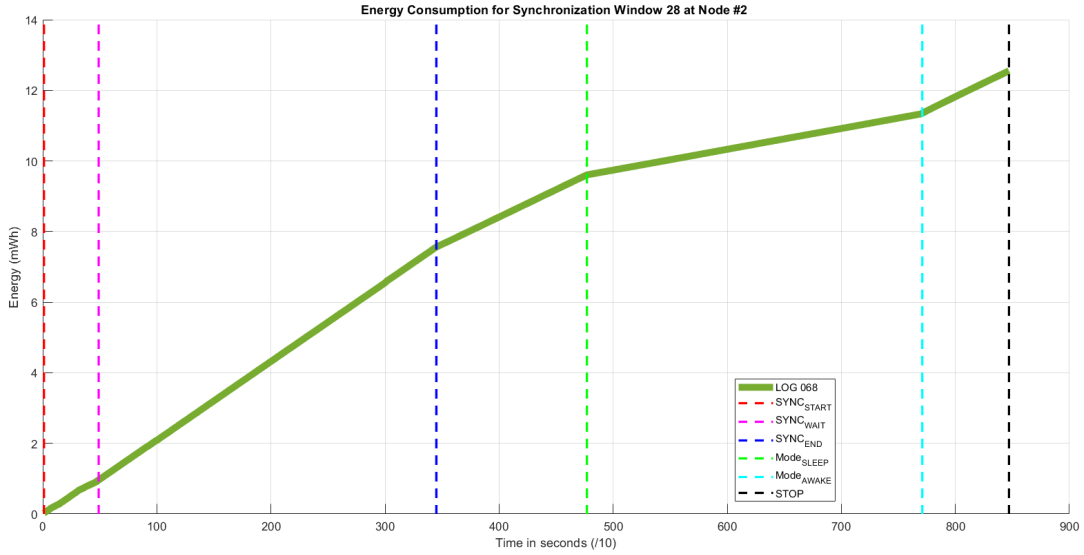

Figure 4.10: Energy Consumption Illustrating Sync States

true value of 30 seconds. Synchronization window 7 sees a drop in average sleep time and can be considered as an anomaly. The increase in average sleep time saturates after/around 15. This can be considered as the optimum synchronization window for node 2.

4.3 Time Synchronization Modes

As discussed before, LWSN nodes are susceptible to longer end-to-end delays. In this section, we intend to find remedies to this issue. Recall from section 3.2 that time synchronization calculation consists of relative offset (α) and relative drift (β) parameters. The individual affect of these parameters need to be examined to optimize the time synchronization of LWSN nodes. We assume that drift is considerably slow in our synchronization context. Henceforth, we will try to measure the impact of Offset-Only (OO) and Offset-and-Drift (OD) in the accuracy of achievable synchronization.

4.3.1 System Definition

Three different synchronization windows, 5, 15, and 25 were chosen for each of the two different synchronization scenarios. Note that, the relative drift term on 3.3 is

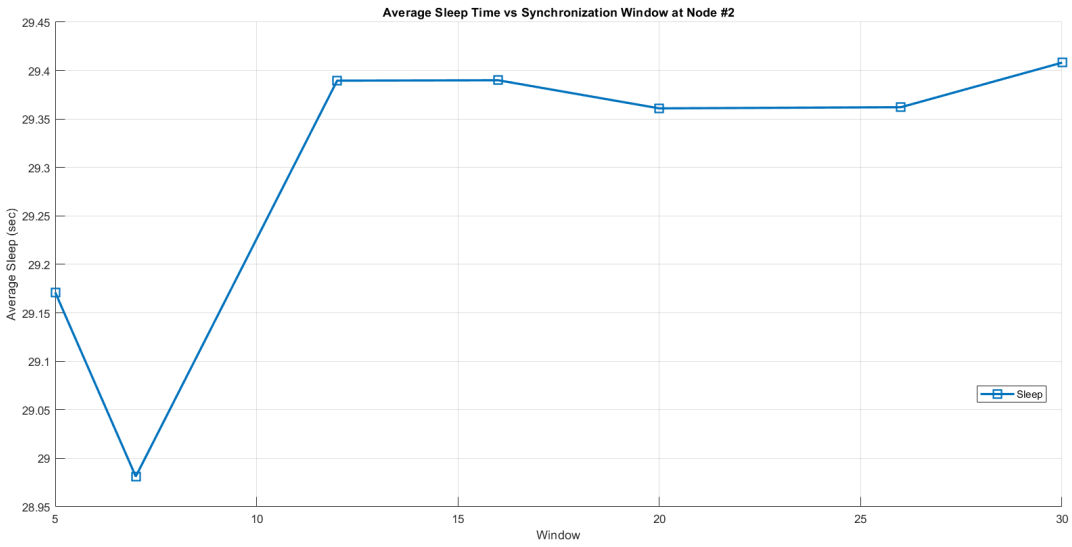

Figure 4.11: Average Sleep Duration w.r.t. Synchronization Window

unity for the OO synchronization unlike OD where it is computed from the time-stamp packets. Each data point consists of a single synchronization operation followed by sampling periods. The time synchronization is characterized by the synchronization window and one of the test scenarios. During the sampling period, the timing data of a node is recorded at the frequency of one Hz for three observation periods of 10, 30, and 60 minutes. The times elapsed are compared against the observation duration to draw conclusions. Again, the LWSN consisted of 6 nodes including a base station.

To be able to analyze and comment on the findings, we establish some terms. The absolute difference or $\Delta\varepsilon$ is the difference between the expected and the measured average times which is given in Eqn. 4.5. The percentage relative difference, ε, is given by Eqn. 4.6. Equation 4.7 represents the differential normalized time and is denoted by $\bar{\varepsilon}$.

$$\Delta\varepsilon = T_{TRUE} + T_{EXP} \tag{4.5}$$

Lower the absolute difference, better the synchronization has been. This is only true for a single data point and provides no information on the general trend. The percentage relative difference term enables to visualize the bigger picture. Therefore, lower the relative difference, the better it is.

$$\varepsilon = \frac{\Delta\varepsilon}{T_{TRUE}} \times 100\% \tag{4.6}$$

$$\bar{\varepsilon} = 1 - \frac{T_{EXP}}{T_{TRUE}} \tag{4.7}$$

The absolute and relative difference terms depend on the mean or average values. Although they are useful in determining the system trend, the minimums/maximums must be analyzed to look for any anomalies. Theses fluctuations might negatively impact the overall system performance. Apart from the above terms, it is a good idea to see the overall trend of the time differences across the observation periods. However, in order to draw conclusions, the normalized data should be analyzed. If the differences are very subtle, the normalized data can be scaled to visualize the differences better.

4.3.2 System Performance

The data for absolute differences, relative differences and normalized average data are tabularized in Tables 4.1 and 4.2, 4.3 respectively.

Table 4.1: Absolute Difference (in seconds)

Window	**Duration** / **Scenario**	**10 Min**	**30 Min**	**60 Min**
5	OD	0.13987	0.18527	0.25320
	OO	0.13787	0.18307	0.25130
15	OD	0.14247	0.18777	0.25567
	OO	0.14837	0.19340	0.26090
25	OD	0.15953	0.20497	0.27323
	OO	0.16363	0.20897	0.27691

Referring to the Table 4.1 of absolute differences, it is obvious that the absolute difference increases with the duration for any of the particular synchronization windows regardless of the synchronization method (i.e., offset-and-drift or offset-only) which is due to the timekeeping module's drift over time. The difference between the two synchronization methods' percentage relative difference for a given window decreases

Table 4.2: Relative Difference (as percentages)

Window	Scenario \ Duration	10 Min	30 Min	60 Min
5	OD	0.023311	0.010293	0.0070333
	OO	0.022978	0.010170	0.0069806
15	OD	0.023744	0.010431	0.0071019
	OO	0.024728	0.010744	0.0072472
25	OD	0.026589	0.011387	0.0075898
	OO	0.027271	0.011610	0.0076921

over time as can be observed from Table 4.2. This decreasing trend can also be observed going diagonally – the difference in ε decreases with increasing both the sampling time and the window. Additionally, it can be concluded from Table 4.3 that using only the offset for smaller synchronization windows is sufficient. In other words, higher the normalized mean is, the better the synchronization is.

Table 4.3: Normalized mean time elapsed (scaled by 100)

Window	OD	OO
5	99.9865	99.9866
15	99.9862	99.9858
25	99.9848	99.9845

For finding the optimum synchronization window, differential normalized mean can be plotted as of Fig. 4.12. Lower the differential normalized value, better the accuracy is. It can be seen on the figure that OO or Offset-Only method for time synchronization performs better at lower synchronization windows compared Offset-and-Drift or OD. This observation is justified because the impact of drift for short period of time is negligible. The significance of this finding is that relatively less computation power is needed at lower synchronization windows and the energy consumption is lowered as well.

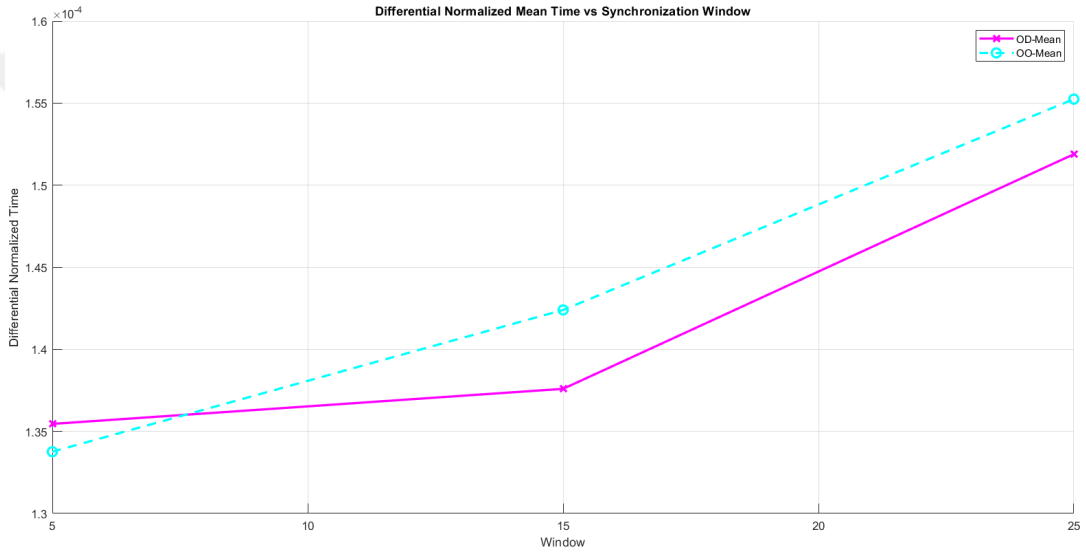

Figure 4.12: Normalized Mean Time Elapsed w.r.t. Synchronization Window

CHAPTER 5

CONCLUSION

The proposed protocol was implemented on a system composed of 8-bit AVR RISC-based *ATmega2560* chip from *Microchip* as the MCU and high performance *CC1200* RF transceiver from *Texas Instruments* operating in the ISM band as the radio module. These inexpensive hardware combination was capable enough to demonstrate all the objectives mentioned in chapter 1. This protocol is also portable as was later realized using *SX1276* LoRa module manufactured by *Semtech* for long range linear network topology.

The energy budget of the node was calculated both theoretically and by experimentation. It is seen that energy consumption of nodes increase with increasing time synchronization window (refer to Fig. 4.9). This is anticipated as any particular node has to spend more time in transmission and reception for higher synchronization windows. Moreover, a generic energy expenditure was illustrated to show the rate of battery depletion at each step of time synchronization. See Fig. 4.10. This can be used to put an upper limit to the energy requirement of the whole system because the second node was examined and it has the highest energy requirement in the network except the base node.

To the best of our knowledge, the contribution of drift along with the synchronization window on the accuracy of synchronization have never been investigated. More importantly, numerous synchronization techniques have been proposed throughout the literature but no studies have been carried out on the determination of optimal synchronization parameters. In this study, we explore the joint effect of synchronization window and offset and/or drift on the time synchronization of Linear Wireless Sensor

Networks (LWSNs). The synchronization accuracy increases with synchronization window size as more samples are used to compute relative offset and drift parameters (Fig. 4.11). Furthermore, we show that computation of offset alone for lower synchronization windows should be sufficient in attaining reasonable accuracy while lowering both the overall energy consumption and system-wide synchronization duration. This is concluded from Fig. 4.12.

If we are to summarize this extensive work, there exists an acute close relationship of time synchronization window, time synchronization accuracy and energy signature of a node. The optimum parameters can be used to characterize the system and the resources can be utilized efficiently. There exists no best selection of parameters, instead the optimum parameters can be chosen based on the system needs and application. This is vital for the practical usage of time synchronization in the fields like the emerging 5G mobile technology, Internet of Things (IoT), Machine to Machine (M2M) communication and similar.

CHAPTER 6

FUTURE WORKS

Like most radio-based systems, the transaction packets are prone to contamination by noise or intentional attacks from adversaries. There are a few ways to address this. The basic and most used method is appending Cyclic Redundancy Check (CRC) bytes to each packet. This was adopted in our application. For rigorous error-detection, both the payload and its CRC can be encrypted with the added cost of packet length. Also, the nodes should make sure that the transmitted packet has indeed reached the target node. A very basic approach of frame counter method was used here to detect packet losses. Additionally, acknowledgment or (ACK) packets can also be used to satisfy this. In this implementation, one-to-one acknowledgment method was used; each reception is acknowledged with a transmission of an ACK packet. While this works flawlessly, it is not the best solution for repetitive packet transmissions/receptions e.g. time synchronization with some medium-sized synchronization window. As a remedy, cumulative ACK packets can be transmitted upon reception of bulk amount of packets belonging to the same task. This pipelining approach can significantly elevate the system performance for relatively larger packets and should be utilized.

Furthermore, this protocol assumes the node IDs are allocated and assigned in the run-time by the developer. To be specific, we have used a sequential incremental numbers as IDs to differentiate between the nodes and establish a hierarchical relationship in LWSN. It can pose a significant threat to the nodes e.g. man-in-the-middle attacks, denial-of-sleep/service (DoS) and etc. A dynamic ID assignment technique might be used although this itself is a research topic and might not be resource friendly i.e. wasn't sought in this work.

One of the most energy-demanding task in any WSN node is radio transmission. We assumed the distance between the nodes is fixed and used constant transmission power throughout the network. If the linear topology homes variable distance between the nodes, variable transmission power can be used. This should considerably improve the energy efficiency.

And lastly, for any implementations or proposed frameworks, the developed system must perform better than some if not all existing solutions. This demands for benchmarking the protocol. Due to lack of ample amount of realistic protocols and systems, this was not achievable at the time of this thesis writing.

REFERENCES

[1] M. Erol-Kantarci and H. T. Mouftah, "Wireless sensor networks for cost-efficient residential energy management in the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 314–325, 2011.

[2] J. M. Barcelo-Ordinas, J.-P. Chanet, K.-M. Hou, and J. García-Vidal, "A survey of wireless sensor technologies applied to precision agriculture," in *Precision agriculture'13*, pp. 801–808, Springer, 2013.

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," 2004.

[4] A. Milenković, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Computer communications*, vol. 29, no. 13-14, pp. 2521–2533, 2006.

[5] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "Pipeneta wireless sensor network for pipeline monitoring," in *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 264–273, ACM, 2007.

[6] M. A. Kumbhar, "Wireless sensor networks: A solution for smart transportation," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 4, 2012.

[7] M. Y. Aalsalem, W. Z. Khan, W. Gharibi, M. K. Khan, and Q. Arshad, "Wireless sensor networks in oil and gas industry: Recent advances, taxonomy, requirements, and open challenges," *Journal of Network and Computer Applications*, vol. 113, pp. 87–97, 2018.

[8] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE transactions on industrial electronics*, vol. 57, no. 10, pp. 3557–3564, 2010.

[9] A.-S. K. Pathan, H.-W. Lee, and C. S. Hong, "Security in wireless sensor networks: issues and challenges," in *2006 8th International Conference Advanced Communication Technology*, vol. 2, pp. 6–pp, IEEE, 2006.

[10] K. Römer, O. Kasten, and F. Mattern, "Middleware challenges for wireless sensor networks.," *Mobile computing and communications review*, vol. 6, no. 4, pp. 59–61, 2002.

[11] D. E. Boubiche and A. Bilami, "A defense strategy against energy exhausting attacks in wireless sensor networks," *Journal Of Emerging Technologies In Web Intelligence*, vol. 5, no. 1, 2013.

[12] T. Bhattasali, R. Chaki, and S. Sanyal, "Sleep deprivation attack detection in wireless sensor network," *arXiv preprint arXiv:1203.0231*, 2012.

[13] R. Dubey, V. Jain, R. S. Thakur, and S. D. Choubey, "Attacks in wireless sensor networks," *International Journal of Scientific & Engineering Research*, vol. 3, no. 3, pp. 1–4, 2012.

[14] D. R. Raymond, R. C. Marchany, M. I. Brownfield, and S. F. Midkiff, "Effects of denial-of-sleep attacks on wireless sensor network mac protocols," *IEEE transactions on vehicular technology*, vol. 58, no. 1, pp. 367–380, 2008.

[15] R. Sokullu and E. Demir, "A comparative study of mac protocols for linear wsns," *Procedia Computer Science*, vol. 52, pp. 492–499, 2015.

[16] F. Alfayez and I. Neill, "Topology-based optimization of linear wireless sensor networks," in *2015 European Intelligence and Security Informatics Conference*, pp. 183–183, IEEE, 2015.

[17] Y.-S. Chen, S.-Y. Chang, T.-W. Chang, and M.-J. Tsai, "Multiple sink placement with latency and reliability guarantee in lossy wireless sensor networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[18] K. S. Yıldırım, R. Carli, and L. Schenato, "Adaptive proportional–integral clock synchronization in wireless sensor networks," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 610–623, 2017.

[19] A. Al-Shaikhi and A. Masoud, "Efficient, single hop time synchronization protocol for randomly connected wsns," *IEEE Wireless Communications Letters*, vol. 6, no. 2, pp. 170–173, 2017.

[20] A. Al-Shaikhi, "Improved-precision time synchronization protocol for wsns based on averaging consensus control," *IEEE Access*, vol. 6, pp. 62261–62271, 2018.

[21] P. Briff, A. Lutenberg, L. R. Vega, F. Vargas, and M. Patwary, "Generalised trade-off model for energy-efficient wsn synchronisation," *Electronics Letters*, vol. 51, no. 3, pp. 291–292, 2015.

[22] F. Wang, C. Yu, X. Wu, and Y. Hu, "Dual time synchronisation method for wireless sensor networks," *Electronics Letters*, vol. 51, no. 2, pp. 179–181, 2015.

[23] H. Wang, H. Zeng, and P. Wang, "Linear estimation of clock frequency offset for time synchronization based on overhearing in wireless sensor networks," *IEEE Communications Letters*, vol. 20, no. 2, pp. 288–291, 2015.

[24] B. Barbagli, L. Bencini, I. Magrini, G. Manes, and A. Manes, "A real-time traffic monitoring based on wireless sensor network technologies," in *2011 7th International Wireless Communications and Mobile Computing Conference*, pp. 820–825, IEEE, 2011.

[25] S. Ali, A. Ashraf, S. B. Qaisar, M. K. Afridi, H. Saeed, S. Rashid, E. A. Felemban, and A. A. Sheikh, "Simplimote: A wireless sensor network monitoring platform for oil and gas pipelines," *IEEE Systems Journal*, vol. 12, no. 1, pp. 778–789, 2016.

[26] K. Aziz, S. Tarapiah, M. Alsaedi, S. H. Ismail, and S. Atalla, "Wireless sensor networks for road traffic monitoring," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 11, pp. 265–270, 2015.

[27] B. Barbagli, L. Bencini, I. Magrini, G. Manes, and A. Manes, "A traffic monitoring and queue detection system based on an acoustic sensor network," *International Journal on Advances in Networks and Services Volume 4, Number 1 & 2, 2011*, 2011.

[28] G. Bam, E. Dilcan, B. Dogan, B. Dinc, and B. Tavli, "Dlwts: Distributed light weight time synchronization for wireless sensor networks," in *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 447–450, IEEE, 2015.

[29] F. L. Walls and J.-J. Gagnepain, "Environmental sensitivities of quartz oscillators," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 39, no. 2, pp. 241–249, 1992.

[30] T. Schmid, Z. Charbiwala, R. Shea, and M. B. Srivastava, "Temperature compensated time synchronization," *IEEE Embedded Systems Letters*, vol. 1, no. 2, pp. 37–41, 2009.

[31] Y. K. Yousif, R. Badlishah, N. Yaakob, and A. Amir, "A review of data collection approaches in linear wireless sensor networks (lwsns)," in *Journal of Physics: Conference Series*, vol. 1019, p. 012006, IOP Publishing, 2018.

[32] M. Abdelhafidh, M. Fourati, L. C. Fourati, A. B. Mnaouer, and M. Zid, "Lifetime maximization for pipeline monitoring based on data aggregation and bio-inspired clustering algorithm," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 666–671, IEEE, 2018.

[33] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile computing*, vol. 11, no. 1, pp. 86–99, 2011.

[34] X.-Y. Liu, Y. Zhu, L. Kong, C. Liu, Y. Gu, A. V. Vasilakos, and M.-Y. Wu, "Cdc: Compressive data collection for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, pp. 2188–2197, 2014.

[35] S. Li, L. Da Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and internet of things," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2177–2186, 2012.

[36] M. Sartipi and R. Fletcher, "Energy-efficient data acquisition in wireless sensor networks using compressed sensing," in *2011 Data Compression Conference*, pp. 223–232, IEEE, 2011.

[37] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 18–39, 2008.

[38] V. Potdar, A. Sharif, and E. Chang, "Wireless multimedia sensor networks: a survey," in *Proceedings of the international workshop on security in RFID and its*

industrial applications with IEEE 23rd international conference on advanced information networking and applications (AINA 2009), pp. 636–641, IEEE Computer Society, 2009.

[39] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer networks*, vol. 51, no. 4, pp. 921–960, 2007.

[40] E. Gürses and Ö. B. Akan, "Multimedia communication in wireless sensor networks," in *Annales des Télécommunications*, vol. 60, pp. 872–900, Springer, 2005.

[41] L. Ferrigno, V. Paciello, and A. Pietrosanto, "Experimental characterization of synchronization protocols for instrument wireless interface," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 3, pp. 1037–1046, 2010.

[42] J. Chen, Q. Yu, Y. Zhang, H.-H. Chen, and Y. Sun, "Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, pp. 2963–2973, 2010.

[43] Z. Cheng, M. Perillo, and W. B. Heinzelman, "General network lifetime and cost models for evaluating sensor network deployment strategies," *IEEE Transactions on mobile computing*, vol. 7, no. 4, pp. 484–497, 2008.

APPENDIX A

ALGORITHMS

A.1 Time Synchronization Algorithm

Algorithm 1: Pairwise Time Synchronization Algorithm at `CHILD`

Result: The local clock gets synced at the end of the process if no error occurs.

```
1  begin
2      initialization;
3      for w ← 1 to W do
4          Request time-stamp from PARENT ;
5          while true do
6              if Response Received then
7                  Calculate α and β for this run;
8                  Reset Timeout;
9                  break; // Break out of While loop.
10             else if Timeout then
11                 if Number of Retries Exceeded then
12                     Throw Error; // Break out of For loop.
13                 else
14                     Retransmit for Current w;// Used for retrial.
15                 end
16             end
17         end
18     end
19     if Errors then
20         Report Error to its PARENT;
21     else
22         Update Local Clock; // Time synced.
23         Go to Energy Saving State;
24     end
25 end
```

A.2 Data Aggregation Algorithm

Algorithm 2: Data Aggregation Algorithm at `CHILD`

Result: Incoming data appended to own (current node's) data is transmitted to `PARENT`

```
1  begin
2      initialization;
3      Append incoming data (if any) to own data;
4      Transmit the Payload to PARENT ;
5      while true do
6          if Ack Received then
7              break; // Break out of While loop.
8          else if Timeout then
9              if Number of Retries Exceeded then
10                 Throw Error; // Break out of For loop.
11             else
12                 Retransmit Current Data Packet;// Used for retrial.
13             end
14         end
15     end
16     if Errors then
17         Report Error to its PARENT;
18     else
19         Go to Energy Saving State; // Data successfully sent.
20     end
21 end
```

A.3 Error Handling Algorithm

Algorithm 3: Error Handling Algorithm at `CHILD`

Result: Error Information is Relayed to the `PARENT`.

```
1  begin
2      initialization;
3      Collect Error Type and Source Node; // Error data.
4      Send Error Data to Immediate PARENT;
5      while true do
6          if Ack Received then
7              break; // Break out of While loop.
8          end
9          else if Timeout then
10             if Number of Retries Exceeded then
11                 break; // Break out of While loop.
12             else
13                 Retransmit Error Data Packet;// Used for retrial.
14             end
15         end
16     end
17     Energy Saving State; // Regardless of successful or
           erroneous operation.
18 end
```