Eindhoven University of Technology
Department of Electrical Engineering
Telecommunications and Electromagnetism

TU/e technische universiteit eindhoven

Decentralized frame synchronization of a TDMA-based wireless sensor network

Fasika A. Assegei

August 14, 2008

Master of Science Thesis
Group: Radiocommunications (ECR)
Supervisors:
Frits van der Wateren (Chess B.V.)
dr.ir. P.F.M. Smulders (TU/e)
Graduation professor:
Prof.dr.ir. E.R. Fledderus

Abstract

Synchronization is a one of the main issues in the design of *Wireless Sensor Networks*(WSNs). Most applications of WSNs make extensive use of synchronization mechanisms like *Time Division Multiple Access* scheduling, accurate time stamping of events, coordinate activities of the network or data fusion. The unique requirements of WSNs in terms of precision, lifetime, energy and scope of the synchronization achieved, make the synchronization methods developed for centralized systems unsuitable for WSNs. This motivates the research of synchronization methods which are aligned to the specific properties of WSN. Interfering nodes, due to mobility and newly incoming nodes, lead to instability in synchronization. Median, an extension of [24] algorithm is implemented currently on the wireless sensor nodes in MyriaNed[10] project. In this research, the flaw in the Median algorithm is explored and three algorithms have been proposed to achieve a decentralized, stable, and energy-efficient synchronization of a WSN. The algorithms achieve synchronization by using the phase error of a node's wakeup time with those of the neighboring nodes, without exchanging the information about the clock time of the sender. So, the methods avoid time keeping on the messages (time stamping) which reduces the overall overload. Stability of the algorithms is simulated in a highly dynamic network. Results from simulation of the models are presented and discussed. The algorithms can be integrated with the slot allocation algorithm to form the MAC layer protocol for a better throughput. The research is concluded with the comparison of three algorithms in terms of energy consumption and performance. A low energy-consumption or a better convergence as well as the length of the guard time can be used as a metric for selecting the algorithm.

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Frits van der Wateren from Chess Innovation Team, for all the support he has given me during the project. I am greatly indebted to him for all the insight and support that he showed in every phase of the project. I am grateful to dr.ir.Peter Smulders and prof.dr.ir. Peter Baltus for providing guidance during the project.

I am greatly indebted to prof.dr.ir. Erik Fledderus for providing me the opportunity to work on such an interesting topic for my Master's thesis. He has been extremely helpful, and was always availble whenever I needed help.

I like to express my appreciation to prof.dr.ir. Peter Baltus and prof.dr.ir. Jean-Paul Linnartz for taking time to serve on my thesis panel.

I want to say thanks for everyone in Chess Innovation team for the wonderful time that I had working at Chess. They are exremely helpful and make the environment lively. Special thanks to my friends, Simon and Aman, and all my friends at TU/e, for their help and encouragement and for making my stay in the Netherlands unforgettable.

Finally, I want to extend my gratitude to my family. No words can describe how grateful I am. Thanks a lot.

Contents

List of Figures

List of abbreviations and symbols

ρ	Maximum clock drift
ξ_i	Offset to be added to Node i's wakeup time
t_{guard}	Length of the guard time
t_{slot}	Time duration of a slot
T_{sync}	Synchronization frequency
clk	A clock cycle
wakeup time	Time that the node starts listening.
CSMA	Carrier Sense Multiple Access
GPS	Global Positioning System
GSM	Global System for Mobile communications
KF	Kalman Filter
LS	Least Squares
MAC	Medium Access Protocol
NTP	Network Time Protocol
RBS	Reference Broadcast Synchronization
TDMA	Time Division Multiple Access
UTC	Coordinated Universal Time
WAN	Wide Area Network
WM	Weighted Measurments
WSN	Wireless Sensor Network

Chapter 1

Introduction

1.1 Wireless sensor networks

Technological advances have led to the development of low-cost sensors, which are capable of wireless communication and data processing. WSNs are distributed networks of such sensors, dedicated to closely observing real-world phenomena. Such sensors can be embedded in the environment or enabled with mobility. They can be deployed in inaccessible, dangerous or hostile environments. The sensors need to configure themselves in a communication network in order to collect information that has to be pieced together to have a broader picture of the environment than what each sensor individually senses. Various applications are realized using sensor networks[6]. As the WSNs become an integral part of the modern era, addressing issues in designing such networks becomes necessary.

One of the design issues in WSN technology is synchronization, which is a critical piece of infrastructure in any distributed system. In sensor networks, a number of factors makes flexible and robust time synchronization particularly important and more difficult to achieve than in centralized networks. Collaboration among nodes is often required for different purposes[2]. A common view of physical time is a basic requirement for nodes to reason about events that occur in the physical world. In addition to these domain-specific requirements, sensor network applications often rely on synchronization as typical distributed systems do. These applications include proper *Time Division Multiple Access* (TDMA) scheduling, coordination of future action and interaction with users[5].

1.2 Existing work on WSN synchronization

Several algorithms have been proposed and researched for synchronization in WSNs. The *Reference Broadcast Synchronization* (RBS) stated in [15] is an important scheme in the area of WSN synchronization. It achieves a Receiver-Receiver pairwise synchronization to remove sender nondeterminism and results in a precision of a few microseconds. It also provides clock frequency estimates between two receivers using a linear regression technique. But a central node is needed for broadcasting reference signals in order to achieve synchronization. Another method for achieving a network-wide synchronization is suggested by [28]. This approach establishes a time conversion table between clocks of two nodes using different estimation techniques. This method has a higher overload as it involves the transfer of sender's clock time and a central node is needed for the synchronization.

Another approach to a decentralized slot synchronization for TDMA-based networks is presented in [21]. It uses the topology of the nodes as a means to weigh the phase error of the sender with the receiver.In this case, higher message overhead is observed in sending the degree of connection (topology information) as a method of synchronization. A different approach to weight-based synchronization for interference elimination for a TDMA based ad-hoc networks is presented in [25]. The algorithm achieves synchronization in a decentralized manner using the nodes offset with its neighbors with a goal of eliminating the interference. The algorithm achieves synchronization by eliminating the impact of interference nodes. But this method directly eliminates nodes which are newly joining or interfering nodes. Correlation method is used in [1] in order to decode the message and learn about the status of the sender node which is used for synchronization. High message overhead is the setback of the algorithm. A decentralized method is introduced in [24] which uses the average of the phase errors to adjust the wakeup time of the node. Even though this method is simple, the method is highly vulnerable in dynamic networks.

1.3 Objective and overview of the research

Upon the literature survey performed, it is observed that some of the synchronization schemes use a central (master) node in order to achieve a synchronization with the nodes. Upon the methods which uses the decentralized methods, interference and mobility are the least addressed issues. Since nodes are subjected to severe changes in environmental conditions, the accuracy of these synchronization schemes might suffer a lot due to the dynamics of the network. As part of the MyriaNed project, Median is used in the synchronization of the wireless sensor nodes implemented in the project.

The primary objective of this research is to address the problems in the Median and develop an algorithm to achieve a decentralized and stable synchronization of a WSN in an energy-efficient method. The algorithm is aimed to replace the currently implemented Median algorithm. Hence, three algorithms are proposed and compared with Median algorithm in terms of performance and energy consumption. The methods use the phase errors between the receiver and its neighbors, without estimating the neighbor's clock as a way of achieving long-term synchronization. Through integration with the MAC layer protocol, a better throughput can be obtained. The algorithms developed have the following characteristics: provides higher precision, adaptive to environmental effects, energy-efficient and achieve long-term synchronization.

The remainder of the report is organized as follows: Chapter 2 presents a general overview of synchronization in WSN and the need for a synchronized time. Chapter 3 discusses the synchronization error and discusses the problem associated with the Median algorithm. Chapter 4 presents the mathematical models of the proposed algorithms for the synchronization of the network. In Chapter 5, simulation results are presented. Analysis of the results in addition to the comparison of the methods with respect to energy consumption are discussed. Finally, Chapter 6 draws the conclusions from the research and suggests future work.

Chapter 2

Synchronization in wireless sensor networks

2.1 Introduction

Chess[20] created a MAC protocol for gossip communication, gMAC, for the wireless sensor network project, MyriaNed, with the gossiping technique in mind. Being a wireless sensor network, the duty cycle of the network is small, around 1%. Looking back at the 1% active time versus the 99% nonactive time ratio, it is a non-trivial matter to keep the network synchronized.

A number of approaches have been introduced for different multiple access techniques, of which CSMA and TDMA are the most common. One problem with CSMA is the time that passes when the radio is busy with idle listening time. Nodes need to listen to the radio for periods of time before they can actually send data.

In TDMA, time is divided into discrete slots. Nodes transmit in rapid succession in one slot and listen in the others. In theory, there will not be any idle listening. There are some pitfalls to this approach. Firstly, the clocks of the nodes are less than perfect; they will never run at exactly the same clock-speed. To keep the nodes synchronized and thus to let them share the same schedule, the small timing error needs to be compensated. Second, there has to be a certain algorithm to allocate the slots among the nodes. They cannot simply all start broadcasting in the first slot, since that would cause nothing but collisions. The research on slot allocation algorithms is being conducted alongside this research[23].

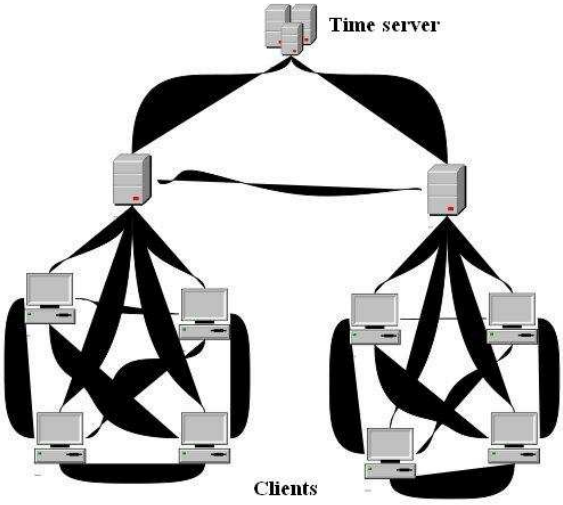

Figure 2.1: A simple NTP diagram

2.2 Synchronization in centralized networks

In a centralized system, the solution to clock drift is trivial. The centralized server will dictate the system time because time is unambiguous. With the centralized concept in mind, different protocols have been proposed and implemented to achieve synchronization in those networks, *Network Time Protocol* (NTP) being the most popular[19].

Due to the infrastructure advantage, NTP is one of the most accurate and flexible means of sending time over the Internet. The protocol is designed to compensate for some, but not all, network time delays between the server and the client. NTP is most successful across local area networks and can give accuracy as good as a few milliseconds. On the world wide web, however, time transfer delays are at the mercy of server traffic and network bottlenecks, and accuracy figures cannot be quoted as easily. NTP uses a round time delay to a universal time reference so that it can adjust the clock of the networked component. It measures delays within the network and within the algorithms on the machine on which it is running. Using these tools and techniques, it is able to synchronize clocks to within milliseconds of each other when connected on a Local Area Network and within hundreds of milliseconds of each other when connected to a *Wide Area Network* (WAN). Figure 2.1 shows the simplified structure of how the NTP protocol works to synchronize client computers. The tiered nature of the NTP time distribution tree enables a user to choose the accuracy needed by selecting a level (stratum) within the tree for machine placement. A time server placed higher in the tree (lower stratum number), provides a higher likelihood of agreement with the *UTC* standard.

Synchronization in distributed systems takes a different kind of approach since there is no central authority to be referred in case of time requests. Central nodes can be assigned as to play the role of a server as in the case of the centralized system. But, most of the time, distributed systems, due to their unique properties, have no central command. Development of a synchronization scheme that satisfies such requirements is challenging. The task becomes particularly daunting in WSNs, in light of their additional domain requirements. Before exploring the specific requirements of WSN, a discussion about the need of a synchronized time is presented.

2.3 The need for a synchronized time

There are many reasons why a synchronized time is needed in a WSN. Some of them include data integration, TDMA scheduling, target tracking and localization. Two of the most common areas where synchronization is a necessity are described below.

Data Integration

Data collection is the main task of the sensor nodes, hence data gathering and integration is a crucial component. The signal processing literature sometimes refers to this as array processing; with heterogeneous sensors, it is often called *data fusion*. There are many applications which need data fusion such as signal enhancement (noise reduction) and source localization. It would seem to be a natural match to implement such algorithms in distributed sensor networks, and there has been great interest in doing so. However, much of the extensive prior art in the field assumes centralized sensor fusion. That is, even if the sensors gathering data are distributed, they are often assumed to be wired into a single processor. Centralized processing makes use of implicit time synchronization. Sensor channels sampled by the same processor also share a common time base. Figure 2.2 shows one example of data fusion application. In order to locate the moving object, in this case the lion, nodes have to have a common notion of time. Upon integration, not only the

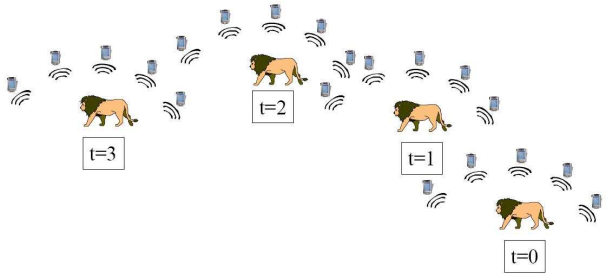

Figure 2.2: Tracking the movement of an object

location of the lion but also the time when the lion was spotted is necessary. So, having the same notion of time is important.

TDMA slot alignment

As it is implemented in WSN, the duty cycle is of prime importance due to the energy requirement of the nodes. A duty cycle is the fraction of time that a system is in an "active" state. Major benefits can be achieved by using this scheduled communication/sleeping protocol:

- Low duty cycles - a node can operate in low duty cycles, hence reducing the energy consumption of the nodes.
- Efficiency in transmission - a sender can efficiently transmit a message to its neighbors by just waking up and sending exactly when a receiver is listening.
- Efficiency in receiving - a receiver can schedule its own time intervals to receive a particular neighboring transmitter.

The scheduled communication/sleeping protocol, the multihop scheme and the rippling of the message through the network are best implemented using a TDMA scheme. Usually, this medium access scheme is best suited for the infrastructure mode where the base stations schedule the TDMA slots for each node. However, such structure is not used in WSN. The scheduling of the TDMA slots is made locally by reaching an agreement between direct neighbors. As previously stated, the WSN being developed at Chess uses a TDMA protocol for channel access, hence the reason for synchronization. TDMA is a channel access method for shared medium networks. It allows several users to share the same frequency channel by dividing the signal into different time slots. The users transmit in rapid succession, one after the other, each using its own time slot. This allows multiple stations to share the same transmission medium while using only the part of the bandwidth they require. Figure 2.3 shows how a time frame is divided into receiving and

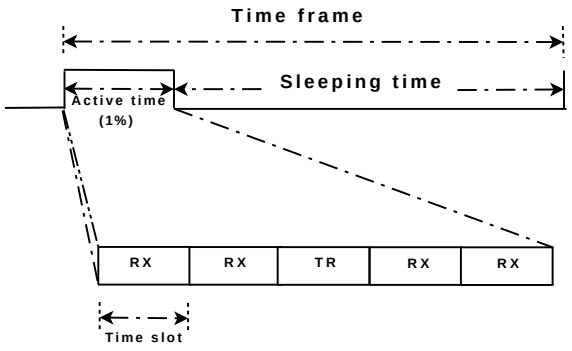

Figure 2.3: TDMA frame

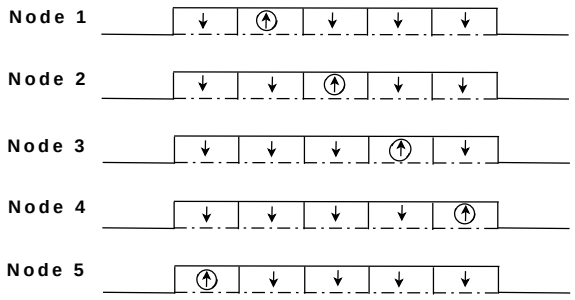

Figure 2.4: TDMA channel access

transmitting slots. In the time frame, the active slot is divided into one transmitting slot and many receiving slots.

In this approach, nodes use time slots in order to communicate with each other. It is considered that two nodes have their TDMA schedule synchronized when the numbers of the current time slot of any of the TDMA schedules involved are the same for any given moment in time. The types of the slot may be different, TX or RX. To successfully initiate a network, the nodes has to be synchronized and capable to send and receive messages. To establish a link between neighboring nodes, a particular node needs to know the schedule of all its neighbors, which is the purpose of the slot allocation algorithm. The MAC protocol used in MyriaNed is the gMAC[23]. The g in gMAC comes from Gossiping, which represents the gossip protocol implemented in the network layer[3].

Each neighbor has a different transmitting slot so that there will be no collision. As shown in Figure 2.4, the transmission slots of the nodes are at different times. Thus, a message transmitted should be received in the receiving slot of the neighboring nodes. In order to have a seamless communication between the nodes, the synchronization of the frames is a necessary part. The processing element and other functions on a WSN operate on a local clock. Due to physical factors, the frequency of the oscillator has drift. When no provisions are taken, it causes the nodes to run out of sync. Given that there is a certain error, the node has to adjust it's wakeup time at the end of the last receiver slot in order to synchronize with the neighbors.

The drift in the node's clock causes the TDMA slot boundaries to drift when we compare two nodes with the same schedule. Hence, the need for synchronization is vital for the seamless communication of the nodes. But synchronization in WSNs is more challenging, which is explained in the next section.

2.4 Wireless sensor networks: Why different ?

Are synchronization methods developed for centralized systems applicable in the case of WSNs? Many assumptions in the centralized schemes do not hold in the case of WSNs. Some of these factors can be described as follows:

- **Energy limitation**: Due to their small size and nature of applications which they are designed for, energy consumption is a major concern in a WSN. Nodes are mostly battery-powered and are expected to run for a long time before they run out of power. An energy efficient method is a requirement for WSNs synchronization.
- **Dynamic nature of the network**: In a centralized system, the topology remains more or less static even if there are physical dynamics involved in some centralized systems like *GSM* . In a WSN, network dynamics results from various factors like mobility of nodes, node failures and environmental obstructions, which prevents simple static configurations. Hierarchical structures might get nodes poorly synchronized, and nodes might not be connected when they need synchronization the most.
- **Diverse applications**: WSNs are used for a variety of applications, which can have totally different needs as far as synchronization is concerned. For example, localization applications need a short-lived but highly precise synchronization, while target tracking applications can tolerate a little lower precision, but want the synchronization to last longer. Some applications might need a global timescale while some others can work with a local timescale.
- **Cost of the nodes**: Sensor nodes are very small in size and must be cheap cost wise since they are implemented in large numbers. Now, a very good synchronization with *GPS* receivers can be obtained, but it will be unreasonable to put an expensive and energy-hungry GPS receivers on a sensor node.

All of the factors mentioned above make the problem of time synchronization more challenging in case of WSNs. Keeping this in mind, we can formulate some design requirements for WSN frame synchronization.

2.5 Requirements of the algorithm for MyriaNed

Precise synchronization

Synchronization cab be of two types: Course and Precise synchronization. Course synchronization is implemented when a node is joining in the network. After a newly joining node listened the synchronization information sent by the other nodes in the network, it adjusts its time slot reference

to the time when it detects the synchronization information, which includes the propagation delay. Precise synchronization is implemented when a node has already joined the network, which repeatedly adjusts the diversion of its time slot reference caused by propagation delay and clock drift. In this research, we focus on precise synchronization.

Lifetime

Lifetime is the interval that clocks remain synchronized, or the interval over which a particular timescale is defined. Sensor networks need synchronization over a wide range of lifetimes, ranging from less than a second to a longer period. In our implementation, the length of lifetime required for synchronization is throughout the life time of the network since the primary reason for the synchronization of the MyriaNed network is for TDMA frame synchronization.

Scope

The scope is the size of the region in which a timescale is defined. In some cases, the scope may be purely geographic (e.g., distance in meters). In other contexts, it is more useful to think about the logical distance, such as the number of hops through a network. Naturally, these two forms are correlated in a sensor network, scaled by its radio's nominal range. A node moving across the field, should be synchronized to the remaining nodes, which it might join again after some periods. A disruption of communication can also result in the disappearance of a node where after some periods of out-of-sight, a communication link can be established again. During the 'come-back', the node should be able to adapt its clock to the remaining nodes in the neighborhood.

Internal vs. External Synchronization

In many distributed systems, synchronization simply means adjusting the clock to a correct time from an outside source. This definition implies that a notion of the correct time, such as UTC, exists. However, in sensor networks, UTC is not always needed. There are situations where distributed nodes need to be synchronized, but not necessarily with the absolute time. In many cases, the exact time at which the action is executed is far less important than ensuring that all nodes act simultaneously. This includes TDMA frame alignment. This function requires internal or relative synchronization: the network must be internally consistent, but its relationship to outside time standards is not needed or known.

Energy budget

The need for energy efficiency permeates virtually in all aspects of sensor networks design. The exact energy constraints are difficult to quantify because there is a wide range of hardware found

across the spectrum of network implementations, and often within a single network. Some nodes have high power batteries and run all the time; others are so constrained that they only wake up occasionally, take a single sensor reading, transmit it and then immediately returning to sleep.

Simplicity

Extra overhead in the adds up to more computational cost. This in turn has a disadvantage in the energy-constraint WSN. A simple message is preferable although it passes most of the burden to the synchronization algorithm to determine the next wakeup time of the node.

Chapter 3

Problem formulation

3.1 Sources of synchronization error

The first step in designing any time-synchronization algorithm would be to understand why and where it is required. The different factors which give rise to errors between the clocks of two nodes can be divided into two main categories[28]:

1. **Oscillator Characteristics**: The sensor nodes' clocks run on very cheap oscillators. The following two properties are prone to error.

   Accuracy: This is a measure of difference between oscillator's expected (ideal) frequency and actual frequency.

   Stability: This is the oscillator's tendency to stay at the same frequency over time. There can be short-term instability due to environmental effects, supply voltage, long-term instability due to temperature effects and crystal aging.

2. **Hardware and environmental factors**: The non-determinism in the message delivery latency is a major source of error in any synchronization algorithm, when applied into real sensor networks. This can be categorized in four type of delays:

   Send time: The time spent at the sender to build the message, i.e. the time duration between generating the message and injecting it into the network.

   Access time: Delay occurred while waiting for access to the transmit channel.

   Propagation time: Time required for the message to travel from sender to receiver.

   Receive time: Time needed for processing at the receiver's network interface.

Even if two clocks are assumed to have the same frequency and no drift, the above mentioned delays in the network result in phase error between two clocks. This is because when the message

from the sender reaches a receiver, the receiver adjusts its clock according to the received message. But the sender's clock changes in the mean time due to network delays. For a network of nodes, delays give rise to errors in accuracy from an ideal clocks.

Below are presented definitions which are used throughout the report.

Phase error: The oscillators of any two nodes can be out of phase at any given time, resulting in different time on both clocks. There can be some initial difference between nodes' clocks at the start of a synchronization procedure. It describes how far apart are the clocks at a given time.

Frequency error: Frequency error measures the difference in the clock rates. This metric is important because it predicts the growth of phase error over time. That is, if clocks are perfectly synchronized now, at what rate are they drifting apart? How well will they be synchronized in 60 seconds?

Clock drift: It is not just that the clocks are running at different rates, but even the frequency of each clock does not stay constant over a period of time. Clock drift arises from the instability of oscillators. For instance, if each clock drifts at a rate of θ msec/sec, then maximum relative drift between two clocks can be 2θ msec/sec.

Wakeup time: Wakeup time is the time that the node starts to listen to the channel.

Clock cycle (*clk*): A clock cycle is the time between two adjacent pulses of the oscillator. The number of these pulses per second is the clock speed, measured in kHz, MHz.

3.2 Clock and frequency standards

The quality of a clock usually amounts to its frequency stability which is the ability of its frequency standard to emit events at a constant frequency over time. The absolute value of the frequency compared to the desired value or, its frequency accuracy is also important. Calibration can easily compensate for an inaccurate but stable clock.

As per our motive to a synchronized clock, clocks are very important in this research. The error bound achieved by a clock synchronization method is linked to both the error inherent in the method itself, and the stability of the clock's frequency. In fact, to some extent, the two are interchangeable. Stable clocks can compensate for a synchronization channel between them that is prone to large but unbiased errors. Many synchronization events can be averaged over a long time. Similarly, a precise synchronization channel can compensate for a poor-stability crystal oscillator

whereas frequent synchronization minimizes the time in-between when the clock is left to fend for itself.

Many types of frequency standards exist. In general, as the stability and accuracy of the clocks increase, so do their power requirements, size, and cost, all of which are important in WSNs. Most commonly found in computer clocks are quartz crystal oscillators, characterized by [25]. Quartz crystals are attractive because they are inexpensive, small, use little power, and perform surprisingly well despite their low resource requirements. The frequency generated by a quartz oscillator is affected by a number of environmental factors: the voltage applied, the ambient temperature, acceleration in space (e.g., shock or altitude changes), magnetic fields, and so forth. More subtle effects as the oscillator ages also cause longer-term frequency changes. The inexpensive oscillators commonly found in computers have a nominal frequency accuracy on the order of between 10^4 to 10^6 that is, two similar but uncalibrated oscillators will drift apart between 1 and 100 microseconds every second, or, between about 0.1 and 10 seconds per day [25], [16]. However, their frequency stability is significantly better with a change in frequency of one part in 10^9 to 10^{11} when averaged over several seconds or more. In our implementation, a 32 kHz crystal clock [8], [7] is used in the MyriaNode[1].

3.2.1 Clock time

In this section, we will see the clock time and its expression. From the definition of frequency:

$$f = d\phi/dt, \tag{3.1}$$

and integrating both sides over time,

$$\phi = \int f(t)dt, \tag{3.2}$$

where f is frequency, ϕ is phase, and t is time.

Thus, the clock time is described as

$$C(t) = \frac{1}{f_o}\int_{t_o}^{t} f(\tau)d\tau + C(t_o), \tag{3.3}$$

where $f(\tau)$ is the frequency of the clock, f_o is the nominal frequency of the crystal oscillator and t_o is the start time of the node. The exact clock drift is hard to predict because it depends on environmental influences (such as temperature, pressure and power). One can usually assume that the clock drift of a crystal clock doesn't exceed a maximum value ρ where ρ represents the maximum clock drift. Figure 3.1 shows the relationship between clock time and absolute time for different drift rate ρ. Note that different clocks have different maximum clock drift values ρ. The

[1]MyriaNode is the sensor node which is built for the project MyriaNed.

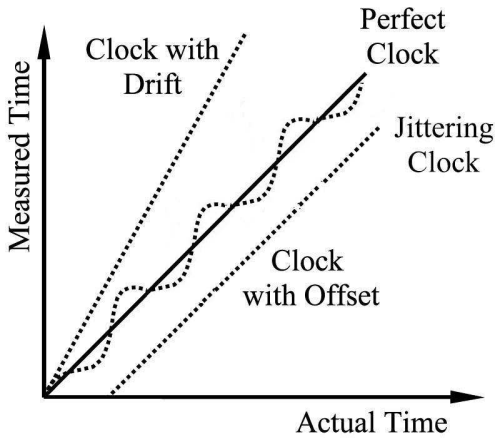

Figure 3.1: Measured time versus absolute time

frequency of the clock is dependent on many factors and is given[27] as

$$f_i(t) = f_o + \Delta f + a(t - t_o) + \Delta f_e(t) + \Delta f_n(t) \tag{3.4}$$

where

t_o = the start time of the clock,

a = aging factor,

f_o = nominal frequency,

Δf = calibration error,

$\Delta f_n(t)$ = frequency instability (noise) term,

Δf_e = frequency error which occurs due to outside factors such as temperature and voltage instability.

The nominal frequency is the ideal frequency at which the oscillator is supposed to run. Hence, all the above factors affect the value of ρ. From (3.3) and (3.4), we get

$$C_i(t) - C_i(t_o) = \frac{1}{f_o}\int_{t_o}^{t} f_i(\tau)d\tau, \tag{3.5}$$

$$C_i(t) - C_i(t_o) = \frac{1}{f_o}\int_{t_o}^{t} [f_o + \Delta f + a(\tau - t_o) + \Delta f_e(\tau) + \Delta f_r(\tau)]d\tau, \tag{3.6}$$

$$C_i(t) = C_i(t_o) + (t - t_o) + \frac{\Delta f}{f_o}(t - t_o) + \frac{a}{2f_o}(t - t_o)^2 + \frac{1}{f_o}\int_{t_o}^{t} f_e(\tau)d\tau + \frac{1}{f_o}\int_{t_o}^{t} \Delta f_n(\tau)d\tau. \tag{3.7}$$

Here are some notes to be considered in the model.

- The amplitude of the short term variations due to noise $\Delta f_n(t)$ is small enough that they do not cause the clock to accelerate or decelerate in a large amount in the long run.
- Individual clock properties are uncorrelated.

- Clock parameters are normally distributed. The variances of the constants in the clock drift equation (initial time error, initial frequency error, aging rate) are all inputs to the model.
- The spread in clock time grows almost linearly as a function of time, due to the dominance of the linear term in the clock drift equation.

The resulting expression in (3.7) is used to model the clock drift of an oscillator.

3.3 Building blocks of a synchronization protocol

The synchronization protocols can be decomposed into four conceptual building blocks[4]. It is shown in Figure 3.2.

- The resynchronization event detection block identifies the points in time where resynchronization is triggered. A single synchronization process is called a round. If rounds can overlap in time, sequence numbers are needed to distinguish them and to let a node ignore all but the newest resynchronization rounds.
- The remote clock estimation block acquires clock values from remote nodes/remote clocks.
- The clock correction block computes adjustments of the local clock based on the results of the remote clock estimating block.
- The synchronization mesh setup block determines which nodes synchronize with each other in a multihop network. In fully connected networks, this block is trivial.

3.4 Performance metrics of a synchronization protocol

There are different metrics in which a synchronization algorithm can be measured. Some of the parameters are discussed below.

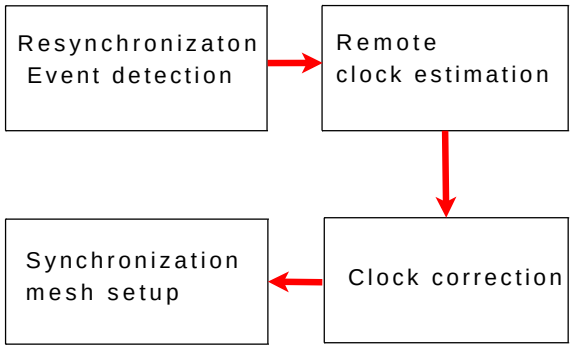

Figure 3.2: Building blocks of a synchronization algorithm

Precision

Deterministic algorithms guarantee absolute upper bounds on the synchronization error between the nodes or with respect to external time source. In this case, the maximum synchronization error between a node and real time or between two nodes is interesting. For stochastic algorithms (which can only give stochastic bounds in the sense that synchronization error is with some probability smaller than a prescribed bound), the mean error, the error variance or some other quantity is relevant.

Energy costs

The energy costs of a time synchronization protocol depend on several factors: the number of packets exchanged in one round of the algorithm, the amount of computation needed to process the packets, and the required synchronization frequency.

Memory requirement

To estimate drift rates, a history of previous time synchronization packets is needed. In general, a longer history allows for more accurate estimates at the cost of increased memory consumption.

Fault tolerance

How well can the algorithm cope with failing nodes, with error-prone and time-variable communication links, or even with network partitions? Can the algorithm handle mobility?

3.5 Synchronization frequency

Decreasing the timing of the synchronization and do periodic execution of the synchronization algorithm greatly reduces the energy consumption of the wireless sensor network. Thus, a synchronization period, T_{sync}, is defined here as the period in which the network can stay synchronized without the application of the synchronization algorithm.

With a synchronization period T_{sync} and the maximum clock drift of a clock ρ , the maximum time difference between a sender and a receiver is

$$t_{diff} = 2\frac{T_{sync}}{T}\rho, \tag{3.8}$$

where the factor of 2 reflects the worst case scenario where each node drifts in the opposite direction.

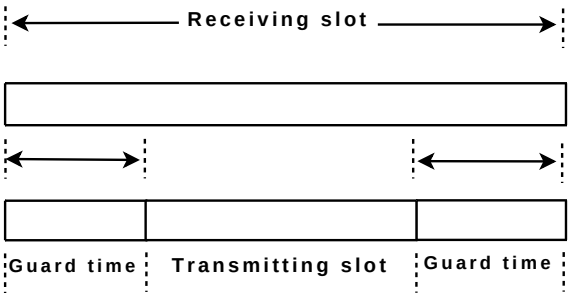

Figure 3.3: Guard time for drift compensation

In the implementation of TDMA for communication, a time interval, called guard time t_{guard}, is usually left vacant in the beginning and end of a slot during which no data is sent. This time can be used for synchronization and compensating for a signal distortion, as shown in Figure 3.3. The guard time provides a safety margin against symbol interference in the time between sequential operations such as transmission, encoding, decoding or switching. As the guard time increases, the probability that the message is received in the time slot increases. But in its downturn, it consumes energy as the node is listening throughout this time. Since the relative time difference between two nodes can be in two direction, the guard time, t_{guard}, needs to be twice t_{diff},

$$t_{guard} = 2t_{diff} = 4\frac{T_{sync}}{T}\rho. \tag{3.9}$$

At this end, the minimum duration for a time slot t_{slot} is

$$t_{slot} \geq t_{guard} + T_{tx}, \tag{3.10}$$

where T_{tx} is a system constant and it represents the required time to send a packet from one node to other. Two nodes have a good communication link when they synchronize their clocks at least once every T_{sync} that gives the following relation,

$$\frac{T_{sync}}{T} \geq 1. \tag{3.11}$$

It is a good practice to design systems having a T_{sync} greater than the time frame. It increases the battery life of a node. However, this has a negative impact on the performance of the network.

To obtain the value of T_{sync} for a particular system with a given duty cycle, the following are considered

$$t_{guard} \geq 4\rho\frac{T_{sync}}{T} \tag{3.12}$$

$$t_{slot} \geq t_{guard} + T_{tx} \tag{3.13}$$

$$\frac{T_{sync}}{T} \geq 1 \tag{3.14}$$

Hence, there is a trade-off in determining T_{sync}: increasing T_{sync} reduces the energy costs of synchronization, but decreases the network perfromance. This is because the synchronization done less frequently so than nodes might drift out of synchronization in the mean time.

3.6 Median algorithm and the flaw

As stated earlier, the Median algorithm is currently implemented in the MyriaNode. With the simplicity concerning the computational need, the Median algorithm is the best fit for the energy requirment. The algorithm is described as follows:

1. Nodes broadcast packets.
2. Each receiver records the time that the packet is received according to the local clock.
3. Each receiver i computes its phase error to any other node j in the neighborhood,

$$\Delta t_{ij}^{(n)} = t_i^{(n)} - t_j^{(n)}, \tag{3.15}$$

   where t_i is the wakeup time of node i and t_j is the wakeup time of node j at the n^{th} period.
4. Receivers compute the offset, ξ_i, to be the median of the phase errors,

$$\xi_i^{(n)} = median(\Delta t_{ij}^{(n)}), \forall j \tag{3.16}$$

5. Receivers adjust their wakeup time by the computed offset value,

$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - G\xi_i^{(n)}, \tag{3.17}$$

where G is the *gain factor*. By multiplying the median with a gain factor, i.e. $G\xi_i^{(n)}$, the output can be adjusted for better performance. Hence a gain factor is added to ensure better precision of the synchronization error.
As per the application of Median, there are flaws on its implementation. In some test-cases, the algorithm fails to converge or stays unsynchronized for a certain period of time. This results in a communication disruption in places where synchronization is essential.

A typical scenario is presented where the Median algorithm takes a longer time to achieve synchronization. In Figure 3.4, a distribution of wireless sensor nodes is shown. Node 10 joins the stable network communication through Node 9. Assume Node 9 belongs to two broadcast domains, Node 3's and Node 10's. Thus, upon the application of the median algorithm, the node tends to adjust its wakeup time with the median of the offsets from its neighbors, Node 3 and Node 10. Adjusting the wakeup time, the offset to be is

$$\xi_9 = median(\Delta t_{9,10}, \Delta t_{9,3}). \tag{3.18}$$

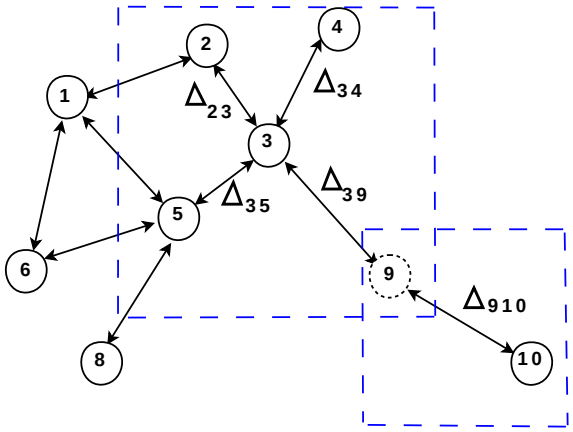

Figure 3.4: A WSN scenario

With Node 10 being isolated from the well established network, it has a major drift from the other nodes. Being in synchronization with the other nodes, Node 9 will drift away from the network after its adjustment with Node 10. Due to the adjustment, tt will take Node 9 more time to synchronize with the network again. Therefore, the performance of the median algorithm decreases with the increase in the dynamic nature of the network. Figure 3.5 shows the state of the network after the synchronization using the Median. Node 9 has drifted away from the networks in order to incorporate the effect of the new neighbor Node 10.

So, in order to address this problem, three algorithms are proposed to realize an energy-efficient, more precise and simple frame synchronization. The algorithms are explained in the next chapter.

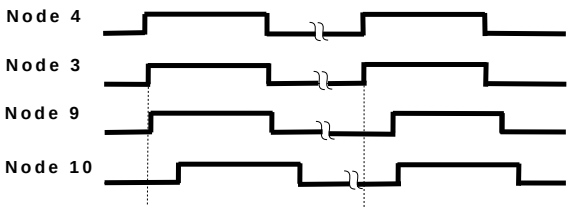

Figure 3.5: Using median for phase error correction

Chapter 4

Synchronicity protocol

4.1 Mathematical model

As part of the message, the nodes send the slot number which they are transmitting to the neighbours. This information is used in determining the time that the message is sent. This information is used in determining the time that the message is sent. Each message that is sent withtin a slot is received by a neighbor at a known clock tick number, $tick_{rx}$, as

$$tick_{rx} = T_{tx} + \frac{t_{guard}}{2}, \tag{4.1}$$

where t_{guard} is the guard time and T_{tx} is the transmission time of a packet.
Whenever (4.1) is not satisfied, a phase error is observed. The wakeup time of a node at a random time after n periods of firings since it is turned on is

$$t_i^{(n)} = \sum_n T_i^{(n)} + t_{io}, \tag{4.2}$$

where $T_i^{(n)}$ is the period of the crystal clock at the n^{th} period since it changes with time according to (3.4) and t_{io} is the initial start-up time of the node. The frequency of the node varies due to the different factors mentioned in (3.4).

Substituting (4.2) in (3.15), the difference in the wakeup time of the nodes will be

$$\Delta t_{ij} = \sum_n T_i^{(n)} + t_{io} - (\sum_n T_j^{(n)} + t_{jo}) \tag{4.3}$$

$$= (\sum_n T_i^{(n)} - \sum_n T_j^{(n)}) + (t_{io} - t_{jo}). \tag{4.4}$$

The offset being applied should be able to compensate for the phase error introduced by the drift as well as the frequency changes. This can be done by adjusting the next wakeup time depending on the difference between the current wakeup time of the node and its neighbors. Application of

offset compensating using prediction of the next wake up time is the promising approach to be implemented on the MyriaNode. The next wakeup time of the node is dependent on the current wakeup time of its neighbors in relation to its previous wakeup time,

$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - \xi_i^{(n)}, \tag{4.5}$$

where T_i is the period of the node's clock and ξ_i is given by

$$\xi_i = f(\Delta t_{ij}). \tag{4.6}$$

The function f is based on an algorithm which takes the wakeup time differences between the node and its neighbors and determines the optimal offset to be added to the next wakeup time of the node.
Different algorithms are presented here and discussed with the simulation results presented in the next chapter of the report.

4.1.1 Weighted measurements

One form of approach to tackle the dynamic behavior of a WSN is a Weighted Measurements (WM) approach. Using this approach, different weights are given to the different phase errors. A weight is added to increase the influence of the close by neighbors and ensure faster synchronization. In addition to that, a new joining neighbor can get synchronized with out disturbing the existing neighbors, adjusting its time to the big swarm of nodes.

The offset, ξ_i, is calculted as

$$\xi_i^{(n)} = \sum_j w_{ij}^{(n)} \Delta t_{ij}^{(n)}, \tag{4.7}$$

where $\sum_j w_{ij}^{(n)} = 1$ and w_{ij} is the weight factor for the phase error Δt_{ij} and n is the number of periods that passed since the node is turned on.

The weighted adjustment is used to modify the next wakeup time of the node,

$$\begin{aligned} t_i^{(n+1)} &= t_i^{(n)} + T_i^{(n)} - \xi_i^{(n)} \\ &= t_i^{(n)} + T_i^{(n)} - \sum_{j=0}^{N} w_{ij}^{(n)} \Delta t_{ij}^{(n)} \\ &= t_i^{(n)} + T_i^{(n)} - \sum_{j=0}^{N} w_{ij}^{(n)} (t_i^{(n)} - t_j^{(n)}) \\ &= T_i^{(n)} + \sum_{j=0}^{N} w_{ij}^{(n)} t_j^{(n)}. \end{aligned}$$

The main task in this algorithm is how to choose the weight factors so that the stability of the network (timewise) is achieved faster. In the selection of the weight factor, two steps are taken.

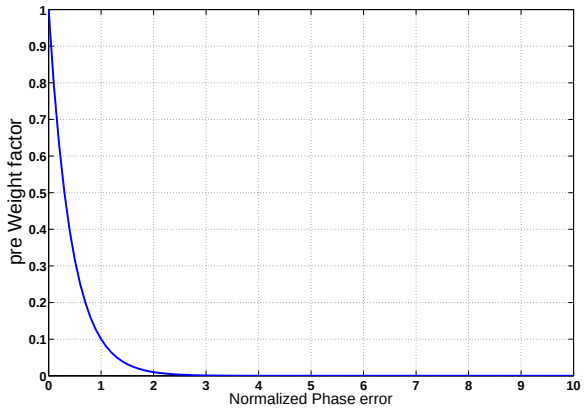

Figure 4.1: pre-Weight factors for the phase error distribution

Firstly, each phase error Δt_{ij} is associated with a pre-weight factor, δ_{ij}. δ_{ij} is used to weigh how far the neighbor nodes has drifted. The δ values are calculated as:

$$\delta_{ij} = ae^{-b\Delta t_{ij}}. \tag{4.8}$$

The parameters a and b are selected using the initial conditions, $\delta_{ij} = 0.1$ for $\Delta t_{ij} = \frac{t_{guard}}{2}$ and $\delta_{ij} = 1$ for $\Delta t_{ij} = 0$. As shown in Figure 4.1, the closer the phase error is to zero, the higher value of pre-weight δ is given to the phase error. The assignment of high values to small phase errors will decrease the time it takes to synchronize the node with the closby neighbours than the rest.

Secondly, the weight factor is calculated based on the distribution of the assigned re-weight factors δ for each phase error. If all or most of the phase errors are small, this corresponds to the fact that the node has drifted away from most of the neighbours. Hence, the synchroniztion is made to adjust the nodes wakeup time towards the neghbours.

Hence, the weight factor is

$$w_{ij} = \begin{cases} 1 - \delta_{ij}, & \text{if } mean(\delta_{ij}) < 0.5 \\ \delta_{ij}, & \text{if } mean(\delta_{ij}) > 0.5 \end{cases}$$

where $mean$ is the average function. The weight then moves towards the large phase errors if the node is a new-comer that wants to join the "already established" network.

Therefore, in WM approach, the first step describes how the node has drifted away from its neighbors whereas the second one describes how the node is positioned in the time topology which surrounds it. Using the WM approach, a series of measurements will be used to estimate the next wakeup time of the node, giving less value/emphasis to the nodes which are out of reach from

the neighboring nodes. The simulation result is shown in the next section to see the effect of the algorithm in comparison to the other proposed algorithms.

Algorithm for WM

1. Nodes broadcast packets.
2. Each receiver records the time that the packet is received with respect to the local clock.
3. Each receiver i computes its phase error to any other node j in the neighborhood.
4. Each receiver i computes the pre-weight factors for the corresponding phase errors.
5. Each receiver i computes the weight factor for each pre-weight factors.
6. Each receiver i computes the offset using the weight factor method.
7. Each receiver adjusts its wakeup time using the offset calculated.

4.1.2 Least squares method

Least squares

Least Squares (LS) is a method to fit a set of n observations with a model of m unknown parameters by minimizing the sum of the squares of the errors ("the residuals"). It uses a set of n data points, which correspond to n neighbors, which are (1, Δt_{i1}), (2, Δt_{i2}),. . . ,(n, Δt_{in}), and a curve (model function) $f(j, \beta)$, also depends on m parameters($\beta_1, \beta_2, ..., \beta_m$). It is desired to find the vector β of parameters such that the curve best fits the given data in the least squares, that is, the sum of squares

$$S = \sum_{j=1}^{n} r_j^2, \tag{4.9}$$

is minimized, where the residuals r_j are given by

$$r_j = \Delta t_{ij} - f(j, \beta), \tag{4.10}$$

for j=1,2,. . . , n.

The minimum value of S occurs when the gradient is zero. Hence, there are m gradient equations to be solved:

$$\frac{\partial S}{\partial \beta_q} = 2 \sum_j r_j \frac{\partial r_j}{\partial \beta_q} = 0 \quad for \quad q = 1, 2..., m. \tag{4.11}$$

In a non-linear system, the derivatives $\frac{\partial r_j}{\partial \beta_q}$ are functions of both the independent variable and the parameters. These gradient equations do not have a closed solution. Instead, initial values must be chosen for the parameters. Then, the parameters are refined iteratively, that is, the values are obtained by successive approximation,

$$\beta_q^{p+1} = \beta_q + \Delta \beta_q, \tag{4.12}$$

for $q = 1, 2..., m..$ Here, p is an iteration number and the vector of increments, $\Delta\beta_q$, is known as the shift vector. At each iteration, the model is linearized by approximation to a first-order Taylor series expansion about β^q.

$$f(j, \beta) \approx f(j, \beta^q) + \sum_q \frac{\partial f(j, \beta^p)}{\partial \beta_q} \left(\beta_q^p - \beta_q\right) \tag{4.13}$$

$$f(j, \beta) = f(j, \beta^q) + \sum_q J_{jq} \Delta\beta_q. \tag{4.14}$$

The Jacobian, J, is a function of constants, the independent variable and the parameters. So it changes from one iteration to the next. Thus, in terms of the linearized model,

$$\frac{\partial r_j}{\partial \beta_q} = -J_{jq} \tag{4.15}$$

and the residuals are given by

$$r_j = y - \sum_{q=1}^{m} J_{jq} \Delta\beta_q, \tag{4.16}$$

where

$$y = \Delta t_{ij} - f(j, \beta^p). \tag{4.17}$$

Substituting these expressions into the gradient equations and equating to 0, the resulting system of equations can be represented as a matrix notation as

$$\left(J^T J\right) \Delta\beta = J^T y. \tag{4.18}$$

Model design

As the algorithm is decentralized, the next wakeup time of the node depends on the current phase errors that it has with the other nodes. The distribution of the phase error is the crucial factor in deciding the type of curve to fit in. The phase errors shows how many time units that the neighbor node is out of touch with the node in focus. The larger the phase error is, the more out-of-sync the node is.

The set of data are the measured phase errors of the node with its neighbors. In order to meet the demand of adjusting the time offset and stabilize the network, the logarithmic curve with the model

$$f(j, \beta) = \beta_1 + \beta_2 log(j), \tag{4.19}$$

is chosen.

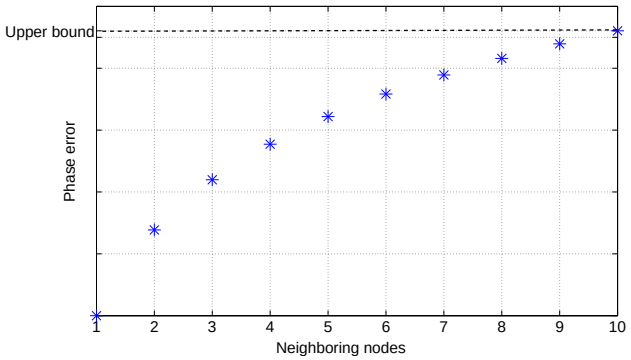

Figure 4.2: Curve fitting using logarithmic function

A logarithmic function can be used to represent the distribution of the offsets in the neighborhood, as shown in Figure 4.2. As shown in the figure, the phase errors are distributed (in an increasing order) so that the maximum phase error will be $\frac{t_{guard}}{2}$. Normalized phase error values tend to be small and follow a curve in such away that the maximum value of the phase error should not be greater than guard time.

The initial values of the parameters β_1 and β_2 is estimated taking into account the state of a synchronized network, which means $f(0) = 0$ and $f(N-1) = \frac{t_{guard}}{2}$ where N is the number of slots. As each measurement arrives from the neighbors, the parameters are calculated in such a way that the measurement error from a pre-determined offset value is reduced. This ensures that the offset to be added in the next wakeup time doesn't diverge in a large amount from the predicted value.

Algorithm for LS

1. Nodes broadcast packets.
2. Each receiver records the time that the packet is received with respect to the local clock.
3. Each receiver i computes its phase error to any other node j in the neighborhood.
4. Each receiver i computes the optimal curve fit for the corresponding phase errors.
5. Each receiver i computes the offset using the function parameters obtained in Step 4.
6. Each receiver adjusts its wakeup time using the offset.

4.1.3 Discrete time kalman filter for synchronization

Discrete time kalman filter

The kalman filter[9] estimates a process by using a form of feedback control. It estimates the process state at some time and then obtains feedback in the form of measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement

update equations. The time update equations are responsible for projecting forward the current state and error covariance estimates to obtain apriori estimates for the next time step.

The kalman filter addresses the general problem of trying to estimate the offset ξ of a discrete-time estimation process that is governed by the linear stochastic difference equation

$$\xi_{ij} = A\xi_{ij-1} + Bu_j + w_{j-1}, \tag{4.20}$$

with a measurement phase error Δt_{ij} that is

$$\Delta t_{ij} = H\xi_{ij} + v_j. \tag{4.21}$$

The random variables w_j and v_j represent the process and measurement noise respectively. They are assumed to be independent of each other, white, and with normal probability distributions

$$p(w) \approx N(0, Q), \tag{4.22}$$

$$p(v) \approx N(0, R). \tag{4.23}$$

The matrix A in (4.20) relates the state at the previous neighbor $j-1$ to the state at the neighbor j, in the absence of either a driving function or process noise. The matrix B relates the optional control input u to the state x. The matrix H in 4.21 the relates the offset ξ_{ij} to the measurement Δt_{ij}. In practice, H might change with each time step or measurement, but here we assume it is constant.

We define $\tilde{\xi}_{ij}^{-}$ to be our a priori state estimate for neighbor j given knowledge of the process prior to neighbor j, and $\tilde{\xi}_{ij}$ to be our a posteriori state estimate at neighbor j given measurement Δt_{ij}. Hence, a priori and a posteriori estimate errors are defined as

$$e_j^{-} = \xi_{ij} \tilde{\xi}_{ij}^{-}, \tag{4.24}$$

$$e_j = \xi_{ij} \tilde{\xi}_{ij}. \tag{4.25}$$

The apriori estimate error covariance is then

$$P_{ij}^{-} = E[e_j^{-} e_j^{-T}], \tag{4.26}$$

and the aposteriori estimate error covariance is

$$P_{ij} = E[e_j e_j^T]. \tag{4.27}$$

With the initial estimates of ξ_{ij-1} and P_{ij-1}, the predictor equations are

$$\xi_{ij} = A\xi_{ij-1} + Bu_j, \tag{4.28}$$

$$P_{ij} = AP_{ij-1}A^T + Q. \tag{4.29}$$

The measurement update equations are responsible for the feedbacks i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

$$K_j = P_{ij}H^T(HP_kH^T + R)^{-1} \tag{4.30}$$

$$\xi_{ij} = \xi_{ij} + K_j(\Delta t_{ij} - H\xi_{ij}) \tag{4.31}$$

$$P_{ij} = (I - K_jH)P_{ij}. \tag{4.32}$$

The matrix K in (4.30) is chosen to be the gain or blending factor that minimizes the posteriori error covariance. Taking the derivative of the trace of the resulting derivative with respect to K, setting that result equal to zero, and then solving for K provides

$$K_j = P_{ij}H^T(HP_{ij}H^T + R)^{-1} \tag{4.33}$$

$$K_j = \frac{P_{ij}H^T}{HP_{ij}H^T + R}. \tag{4.34}$$

Looking at (4.34), it is seen that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_j \to 0} K_j = H^{-1}. \tag{4.35}$$

On the other hand, as the a priori estimate error covariance P_{ij} approaches zero, the gain K weights the residual less heavily. Specifically,

$$\lim_{P_{ij} \to 0} K_j = 0. \tag{4.36}$$

The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations.

The update equation is

$$\xi_{ij} = \xi_{ij} + K(\Delta t_{ij} - H\xi_{ij}). \tag{4.37}$$

The difference $\Delta t_{ij} - H\xi_{ij}$ in (4.37) is the residual. The residual reflects the discrepancy between the predicted measurement $H\xi_{ij}$ and the actual measurement Δt_{ij}. A residual of zero means that the two are in complete agreement.

Filter design

The parameters in the update and predict equations of the kalman filter are chosen to achive the best performance. The transition matrix A indicates how fast/slow the next wakeup time should be compared to it's previous value and/or neighbors, which is taken to be 1. The initial estimates of ξ and P are selected from the previous firing time of the node, one period earlier. This ensures that the nodes are on the same track as the previous time, since the neighbors remain the same with some exception of mobility and interference. H is used to compare the predicted value with the observed value, which is taken as 1.

In the filter design, the covariance matrices R and Q have a significant role in the overall implementation of the algorithm. R represents how the measurments are valued to affect the outcome and Q sends a signal as to how the estimated value is weighed. By chosing the value of R to be $10e-6$ and Q to be 1, the direction of focus can be adjusted so that a maximum performace is achieved.

Algorithm for Kalman Filter

1. Nodes broadcast packets.
2. Each receiver records the time that the packet is received.
3. Each receiver i computes its phase errors to any other node j in the neighborhood.
4. Each receiver i computes the optimum kalman update value for each phase error recursively.
5. Each receiver adjusts its wakeup time using the offset calculated.

4.2 Reducing the guard time

Three algorithms are proposed in the previous section. As the performance of the algorithms increases, the guard time can also be reduced to conserve energy. This inturn reduces the duty cycle.

Let x denote the guard time when the median algorithm is implemented (Figure 4.3). Thus, the slot duration will be

$$T_{slot} = 2x + T_x, \tag{4.38}$$

where T_x is the transmit time of the node.

With N slots, the duty cycle is then

$$D = \frac{NT_{slot}}{T}, \tag{4.39}$$

where T is the period of a time frame.

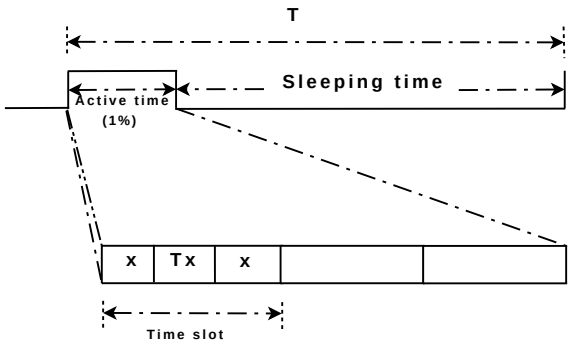

Figure 4.3: Guard time and reducing duty cycle

Substituting (4.38) in (4.39) equation, the duty cycle becomes

$$D = \frac{N(2x + T_{slot})}{T}. \tag{4.40}$$

With the better performance achievement with the other algorithms, the guard time can be reduced depending on the precision of the algorithm. Let ϵ be the guard time reduction in clock cycles. Hence, the new guard time will be $2(x - \epsilon)$.

The new duty cycle becomes

$$D_n = \frac{(2(x - \epsilon) + T_x)N}{T}. \tag{4.41}$$

Arranging the equation results in

$$D_n = \frac{(2x + T_x)N}{T} - \frac{(2\epsilon)N}{T}. \tag{4.42}$$

Hence, with a performance increase in ϵ clk results in the duty cycle reduction of

$$D - D_n = \frac{(2\epsilon)N}{T}. \tag{4.43}$$

The decrease in the guard time of the slot is thus dependent on the algorithm's performance (ϵ) and the number of slots in the frame. As the number of slots increases with a constant performance increase ϵ, the energy conservation also increases linearly.

Chapter 5

Results and discussion

5.1 Simulation setup

The simulation parameters are presented in the following table.

Duration time frame	1s
Number of slots	10
Data rate	2Mbps

Nodes communicate by broadcasting. The start up time of the nodes is gaussian distributed random variable, t_{io}. The nodes are positioned uniformly across the simulation area at the start. The proposed algorithms are KF, M, WM, LS which represent Kalman Filter, Median, Weighted Measurments and Least Squares respectively. A clock cycle is represented here as clk which shows the resolution of the quartz crystal clock. The synchronization error is defined as the maximum difference between the wakeup time of the nodes in the neighborhood. The simulation is conducted in different scenarios and some test case results are presented here for discussion. Results with different scenarios and parameters are also included in Appendix A.

5.2 Simulation results

Case I

In the first set of simulations, the synchronization error is simulated for the nodes which are static, hence no effect of mobility. The number of nodes is taken to be 20 and 50 in this set of simulations.

Figure 5.1 shows the synchronization error for 20 nodes operating in a static environment. The algorithms achieve a stable synchronization. As LS and WM has the same effect as the Median in case of stable network, the performance of both algorithms is similar with that of Median.

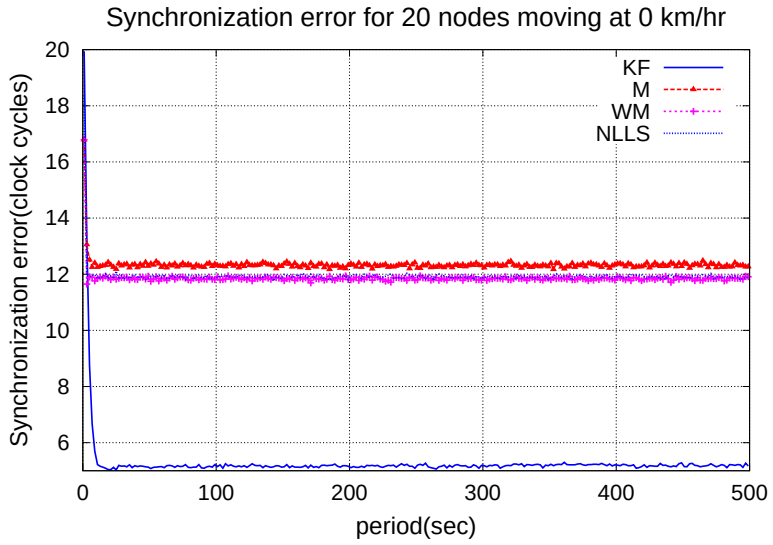

Figure 5.1: Synchronization error for 20 nodes - static

Without the dynamic nature of the network, both tend to equal the average of the phase errors. KF has the best performace since correcting the phase errors involves predict and update which results in better precision.

Figure 5.2 is the result of a simulation for 50 nodes. The synchronization error in general increases as the number of nodes increases. Comparing the individual algorithms, KF has the best precision whereas LS and WM have a better performance each than the Median. Median results in a lower performance on average.

Case II

In second set of simulations, the mobility of the nodes is taken into account. Here, the number of the nodes is taken to be 20. The simulations are conducted for different speeds, at average speeds of $6km/hr$ and $20km/hr$. The chosen speeds emulate the speed of a walking man and an average speed of a slowly moving vehicle.

Figure 5.3 shows 20 nodes with a speed which is an gaussian random variable with mean $6km/hr$ and standard deviation of $1km/hr$. WM and LS show a better tolerance to dynamic networks due to the algorthms capabilty to synchronize towards a more stable network. LS has also a better tolerance with less value given for large phase errors. Hence, WM and LS perform better concerning the synchronization of the frame than the Median. Median algorithm has a disruptions which made the network out-of-synchronization, resulting in high synchronization errors at times. This occurs due to the mobility of the nodes which disrupts the stability networks as it tries to

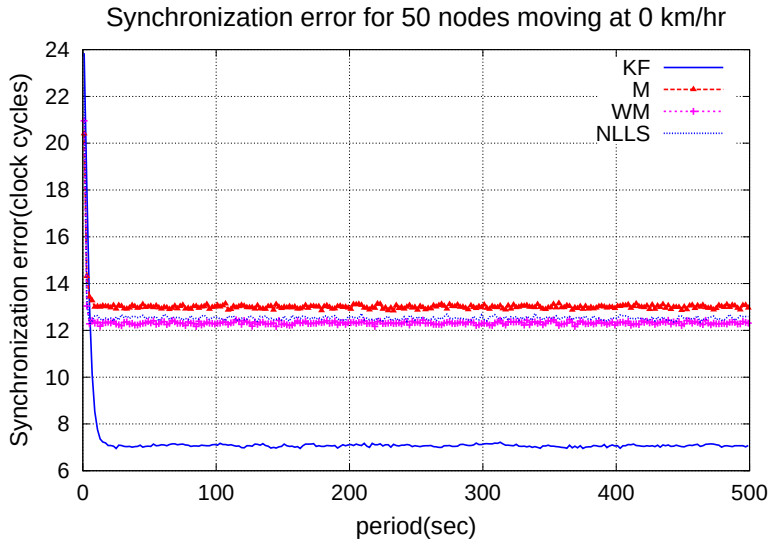

Figure 5.2: Synchronization error for 50 nodes - static

compute the median of the phase errors, as indicated in Chapter 3. KF outperforms all due to its dynamic and recursive nature.

The speed of the nodes is changed to a random variable with mean $20km/hr$ and standard deviation $1km/hr$. The simulation results are presented in Figure 5.4. KF performs the best showing that the algorithm is stable in dynamic networks. WM and LS perform good when it comes to stability. As the dynamics of the networks increases, Median becomes more unstable and exhibits large synchronization error. LS and WM have a stable performance when the network becomes dynamic with constantly changing connection between the nodes.

The relative comparison of the algorithms performance improvement with the Median is shown in Figure 5.5 for nodes moving at $20km/hr$. WM and LS perform, on average, $6-8\%$ better than Median, but with more tolerance in handling the disruptions. The best performer, KF, has on average 60% better performance than the Median one.

Case III

In this set of simulations, the number of the nodes is increased, to 50. With slowly moving nodes ($6km/hr$), the results are shown in Figure 5.6. Large disruptions occur when using Median due to nodes moving slowly in the surrounding (leave the network and join again after some time), resulting in a larger drift with neighbors before getting back to the network. LS and WM perform better when it comes to stability, as out-of-sync nodes are forced to join the stable network. KF has the best preformance, both in precision and stability.

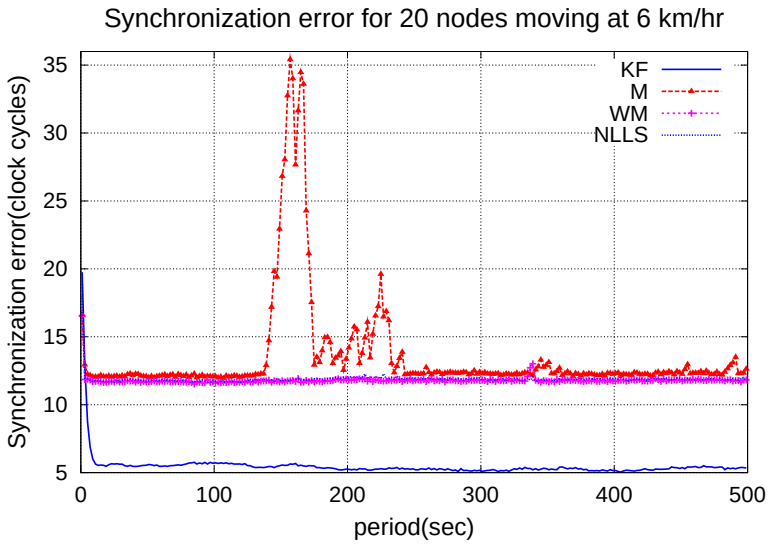

Figure 5.3: Synchronization error for 20 nodes - speed 6km/hr

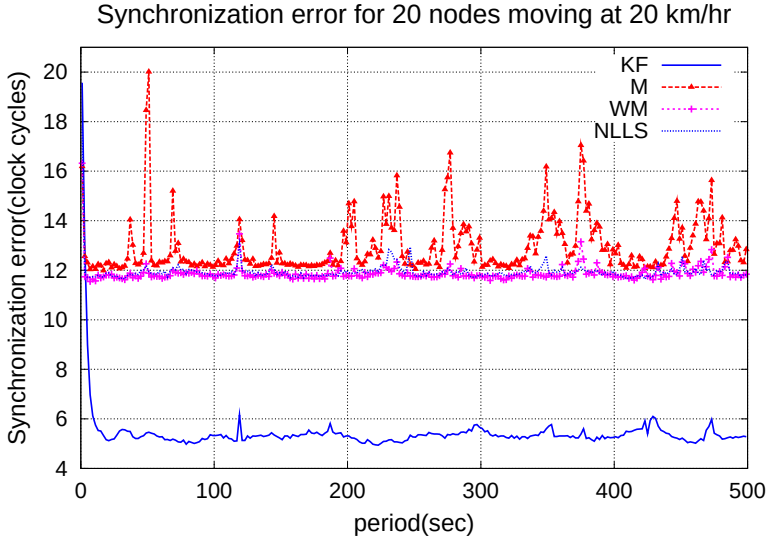

Figure 5.4: Synchronization error for 50 nodes - speed 20km/hr

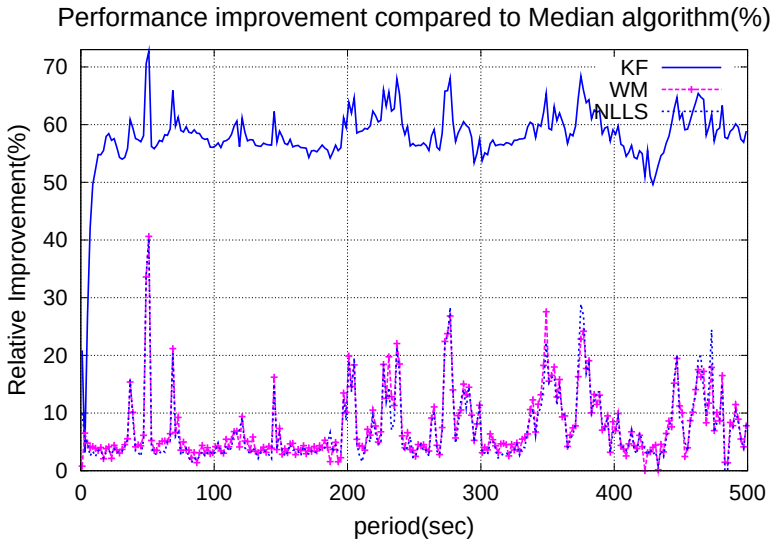

Figure 5.5: Relative performance improvement of algorithms from Median - 20 nodes

With a speed of $20km/hr$, the simulation results are presented in Figure 5.7. As the speed increases, the instability on the synchronization of the nodes also increases, which makes the Median more prone to high synchronization errors. WM has better tolerance towards mobility of the nodes, whereas the LS does well regarding stability. KF has the best precision out of all. As the speed increases, the precision of the synchronization increases too since faster speed results in faster synchronization within the network.

For the set of nodes with a higher speed , a relative comparison is made to see the performance of the nodes with the Median algorithm. The Median algorithm is shown to perform the worst in this case. There are a lot of disruptions in the network, making it more unstable whereas the other algorithms adapt to the changes faster. Figure 5.8 shows that KF performs the best against Median algorithm, 45%. WM and LS perform well against Median too, 20% on average. It has been shown that the median is prone to error in case of high dynamics in the network.
The Kalman filter has a better precision and stability than the rest whereas WM and LS has a better tolerance to a dynamic network.

5.3 Energy consumption

Additional energy expenditure

As the results in the previous subsection shows, KF algorithm performs well in all conditions, so do WM and LS when stability is a vital metric. The downside in implementing these algorithms, despite the performance gain achieved, is the energy consumption. As the algorithms are going

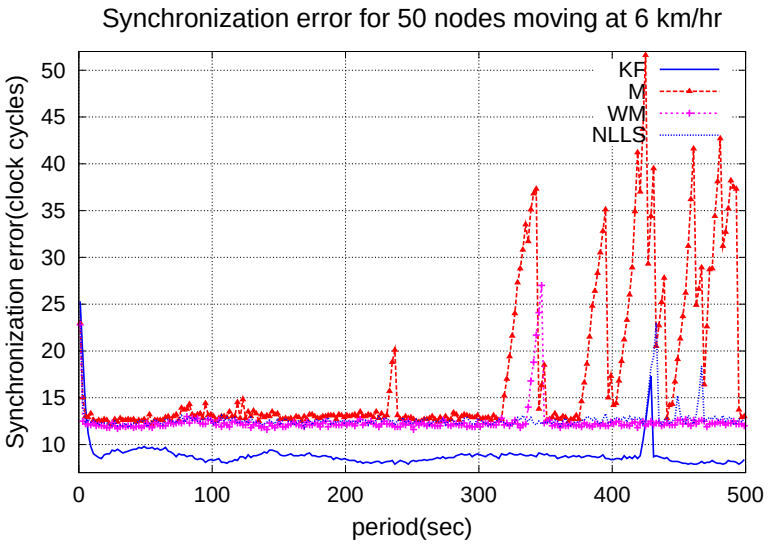

Figure 5.6: Synchronization error for 50 nodes - speed 6km/hr

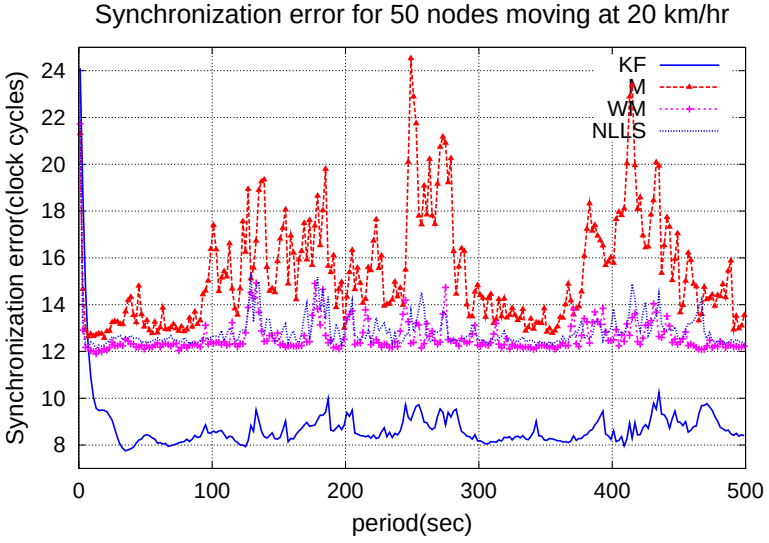

Figure 5.7: Synchronization error for 50 nodes - speed 20km/hr

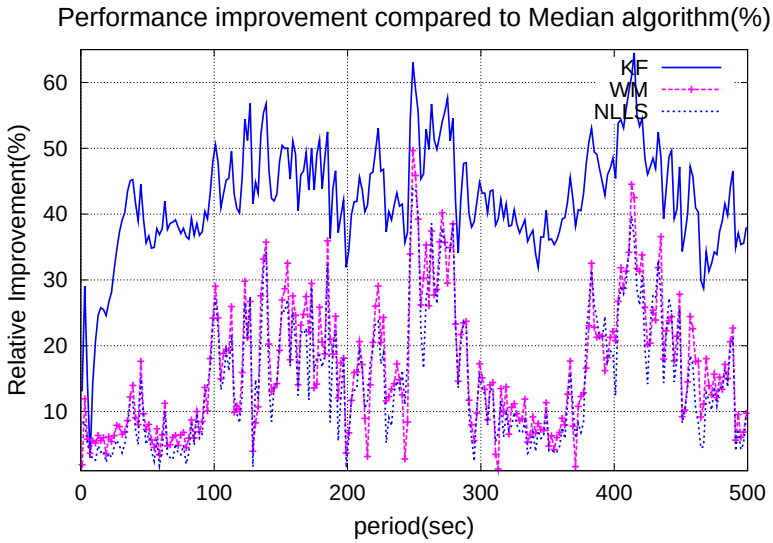

Figure 5.8: Relative performance improvement of algorithms from Median - 50 nodes

to be implemented on the sensor nodes, the energy consumption is a priority in embedded system algorithm development. The algorithms are written in C and implemented on the test nodes to see the effect that they are going to have on the energy consumption of the nodes and presented below.

The results shown in Table 5.1do not reflect the exact energy consumption of the algorithms as the energy consumption is dependent on other factors besides the execution time. But it can be used to put the comparison into perspective.

Table 5.1: Table to represent the energy expenditure of the algorithms

Algorithm	Average Execution Time per frame	Increase in execution time per frame	Energy Consumed per frame
M	65 μs	0	0
LS	82.5 μs	7.5μs	0.063 μJ
WM	90 μs	25μs	0.21 μJ
KF	150 μs	85μs	0.714 μJ

Energy gain

In order to reduce the energy that is going to be spent on the algorithm with a better performance, the guard time of the slot can be reduced, due to a better performance showing by KF, WM and LS. Hence, t_{guard} for KF is taken to be $10clk$ whereas the guard time for WM and LS is taken to be $24clk$ each from the simulation results. t_{guard} for the Median is taken to be $26clk$. The number of slots is taken to be 10, so the total guard time per frame will be 10 times t_{guard}. In the following

Table 5.2: Table to show the energy saving due to the guard time reduction

Algorithm	Guard time per frame	Reduction in t_{guard} per frame	Energy saving per frame
M	792 μs x 10	0	0
LS	732 μs x 10	600 μs	16.27μJ
WM	732 μs x 10	600 μs	16.27μJ
KF	305 μs x 10	4870 μs	132.07μJ

table, the energy saving is calculated in comparison with the Median. The length of the guard time has a close-to-linear relation with the power being consumed in listening to the channel. Thus, with the performance gain obtained, the guard time of the slot can be decreased, hereby decreasing the energy consumption of the node in general. Comparing the energy consumption and the energy expenditures, a large net gain can be obtained from the proposed algorithms. It can be noted here that communication costs more than computation (as listening radio consumes more energy than CPU active time). Since the length of the guard time is dependent on the number of slots, the energy to be saved increases as the number of slots increases.

The comparison on energy gain and expenditure is shown in Figure 5.9. As the number of slots increases, the net gain in energy increases.

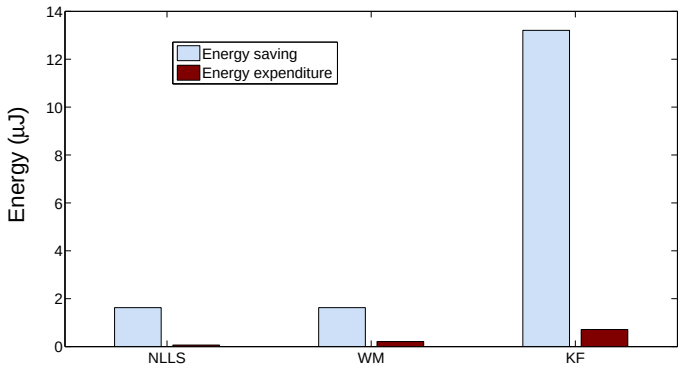

Figure 5.9: The energy gain of reducing the guard time compared with the compuational cost per RX slot

Chapter 6

Conclusion and recommendation

6.1 Conclusion

Decentralized synchronization algorithms are proposed for a TDMA-based WSN using Weighted Measurements (WM), Least Squares(LS) and Kalman Filter (KF) methods. Simulation is conducted with different scenarios, especially taking the effect of mobility. Comparison of the algorithms with the currently implemented Median algorithm is conducted and the results are presented.

In a static environment where the nodes are stationary, the KF performs the best whereas the WM and LS have shown a similar performance since they incline to calculate the average of the phase errors. In terms of stability, all the three algorithms have shown a similar performance.

WM and LS show a very good tolerance in a dynamic network. Having a faster adaptation to the changes in the network made the algorithms preferred ones to a network characterized with dynamic behavior. Besides, simple designs of the algorithms is an advantege in energy-constrained WSN. KF estimation of the next wakeup time is yields a very good performance with its recursive and dynamic nature. KF has also good stability making it resistant to topology changes occurring in the network.

Using these algorithms for a TDMA-based WSN synchronization (WM, LS and KF), the precision of the synchronization error as well as the stability increases. This in turn has a positive impact on the energy consumption of the nodes because the synchronization period, T_{sync} can be increased or the guard time of the node's frame, t_{guard}, can be reduced to achieve the same performance as the Median algorithm. Results are presented and discussed. But in the downside, the energy consumption of the algorithms is greater than the Median algorithm's energy consumption

as the algorithms are more complex computationally. Analysis is made and presented about the energy saving made by decreasing the guard time of the slot. A net gain of battery life can thus be achieved by using the proposed algorithms.

6.2 Recommendation

As is the case with new design techniques for any system of protocol, there is always room for improvement. Although the performance of all our approaches are more precise and stable, some factors which could be considered to make it a good synchronization scheme on all aspects have been identified. Below are some suggestions on the future work.

- Different software power minimization techniques can be applied to reduce the power consumption of the algorithm implementation, making them more energy efficient for implementation.
- As the main sources of error for the clock inaccuracies are identified, resolving these errors is one approach which can be cost effective as well as simple in implementation. This can be achieved using the available resources like the temperature sensor in the microcontroller of the MyriaNode. Constantly updating the temprature of the surrounding and correcting the drift on the node can achieve a better precision on the synchronization.
- Practical implementation of the algorithms and inspect the behavior is another step in pefecting the algorithms. Upon the completion of the ongoing project on developing a Software Defined Radio (SDR) for the inspection of the nodes' wakeup times, a proper evaluation and enhancements on the algorithms, being implemented on the MyriaNodes, is achievable. Real time feedbacks can be used to improve the algorithms towards perfection.

Appendix A

Additional simulation results

A.1 Synchronization frequency

Increasing the time interval in which a synchronization algorithm is executed has the advantage in energy conservation. On the other hand, the performance of the network decrease as the nodes are getting sycnrhoniazed less frequently. Simulation results of different scenrarios are conducted to see the effect of the synchronizatio frequency and are presented here.

Simulation is conducted for different synchronization frequencies T_{sync} on the KF algorithm. Figure $A.1$ shows the performance of KF with varying the synchronization frequency T_{sync} for static nodes. As the synchronization frequency increases, the performance of the algorithm decreases, herby increasing the synchronization error. Even if this approach decreases the computational cost on the nodes, it has a negative impact by decreasing the network performance. In this set of simulations, the speed of the nodes increased to see the effect of mobility on the synchronization frequency. Figure $A.2$ and Figure $A.3$ shows the performance of KF at different speeds, $6km/hr$ and $20km/hr$ respectively. As the speed increases, the synchronization error increases. Besides the synchronization error, the ability of KF to adapt to changes is also affected due to the increased in the synchronization frequency. As shown in the results, there are large disruptions in the synchronization error when the nodes speed is set to an average speed of $6km/hr$. With the increased speed, the synchronization error decreases as the nodes are fast in joining the network that they left. As the speed of the nodes increases to $20km/hr$, the effect of increasing T_{sync} also increases with diversions of synchronization error. As the faster moving nodes join the network in less time, the upper bound of the synchronization error is less as the speeds increases.

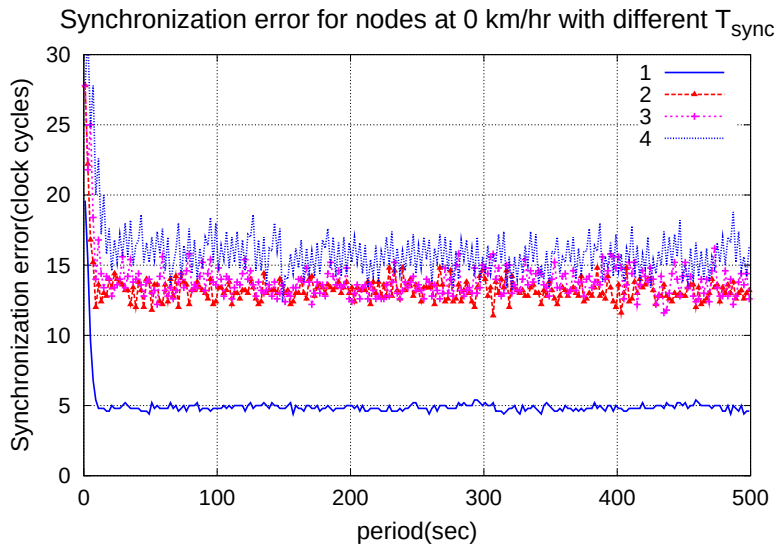

Figure A.1: Synchronization error of a KF for static nodes with different T_{sync}

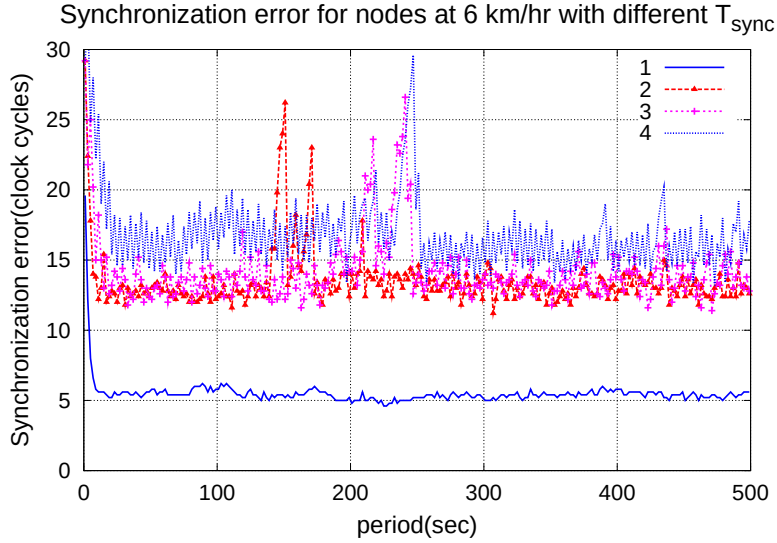

Figure A.2: Synchronization error of a KF for nodes at $6km/hr$ with different T_{sync}

A.2 Synchronization error for nodes with different speeds

As in a real world scenario, a simulation is conducted with nodes having different speeds. The speed of the nodes is taken to be a gaussion distributed random variable with a mean speed of $10km/hr$ and standard deviation of $5km/hr$.

The synchronization error for nodes travelling with different speeds is given in Figure $A.4$ for 20 nodes. Similar results are obtained for the networks where unstability is observed in the Median

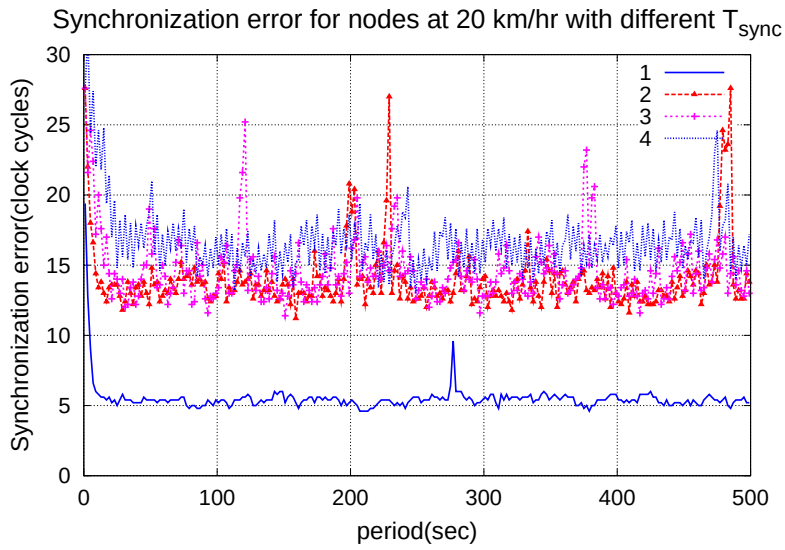

Figure A.3: Synchronization error of a KF for nodes at $20km/hr$ with different T_{sync}

algorithm. WM and LS has a good tolerance towards the dynamic nature of the network. KF has the best precision, in addition to the stabililty.

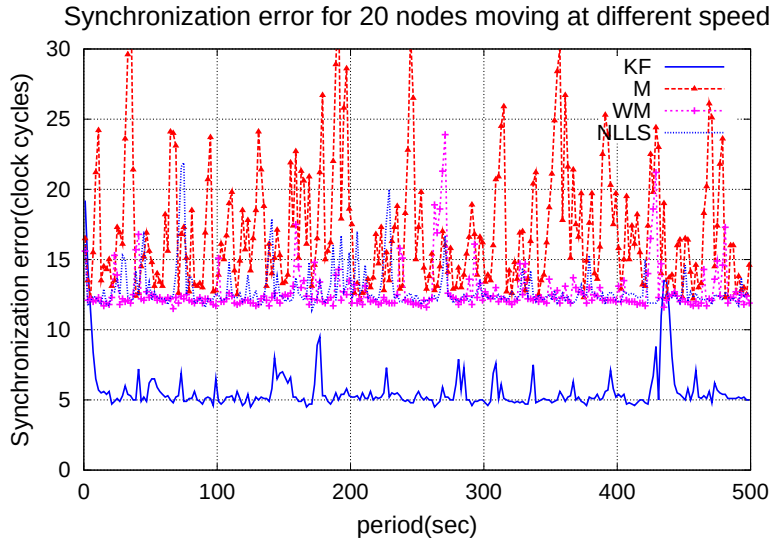

Figure A.4: Synchronization error for 20 nodes moving at different speeds

For 50 nodes, the simulation result for is shown in Figure A.5.In both cases, KF performs the best out of all the algorithms whereas the tolerance of WM is better than Median. LS also performs better than Median interms of stablity.

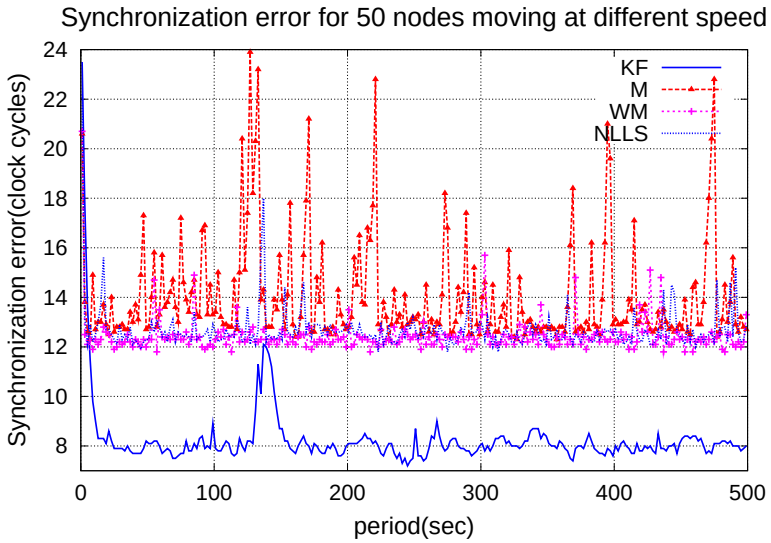

Figure A.5: Synchronization error for 50 nodes moving at different speeds

A.3 The effect of number of nodes on KF

The synchronization error increases as the number of nodes increases, as dictated by the gradient equations in [26] and [18]. Figure *A*.6 shows the synchronization error for KF using different number of nodes, distributed across the simulation area uniformly. The synchronization error increases besides the fact that the logical distance (timewise) between two nodes remain the same.

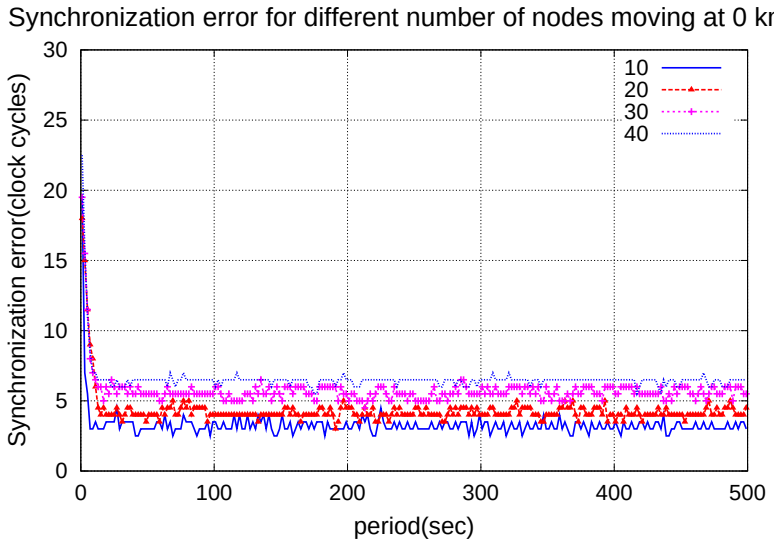

Figure A.6: Synchronization error of KF for different number of static nodes

A.4 Battery life of a Myrianode versus the guard time

Using a 2400 mAh battery, the battery life of a MyriaNode is calculated by varying the length of the guard time. Figure $A.7$ shows how the battery life is affected by the increase in the guard time of the nodes frame. The model uses a TDMA frame of 10 slots.

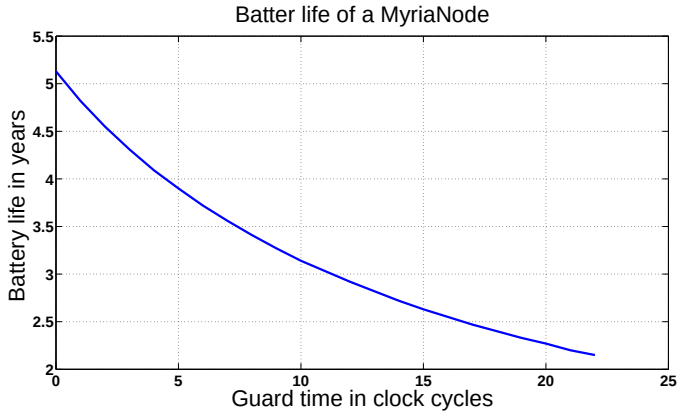

Figure A.7: Battery life of a Myrianode vs the guard time

Appendix B

MyriaNode

Features

- 2.4 GHz ISM band
- Nordic nRF24L01 Radio
- Integrated 1/4λ PCB antenna
- ATMega645 processor
- 64kB FLASH
- 4kB SRAM
- 2kB EEPROM
- 32kHz Crystal clock
- Size: 20 X 40 mm
- Single supply voltage: 1.9V - 3.6V

Figure B.1: MyriaNode v2.0

Description and architecture

This Wireless Sensor Node is the second generation product for the MyriaNed project. It integrates a Nordic[11] radio module, antenna and embedded processor all on a PCB. The module is equipped with the software modules as they are being developed by one or more of the working groups of MyriaNed.

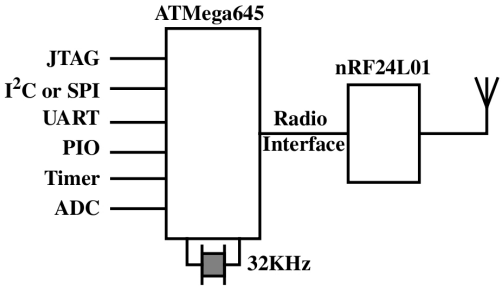

Figure B.2: MyriaNode architecture

PCB Pin #	ATMega pin	Description
Z701	54	PF7, JTAG TDI
Z702	55	PF6, JTAG TDO
Z703	56	PF5, JTAG TMS
Z704	57	PF4, JTAG TCK
Z705	58	PF3, ADC3
Z706	61	PF0, ADC0
Z707	63	Analog Gnd
Z708	-	Supply (Z727)
Z709	2	PE0, UART RxD
Z710	3	PE1, UART TxD
Z711	4	PE2, AIN0
Z712	5	PE3, AIN1
Z713	6	PE4, SPI_SCK, I2C_SCL
Z714	7	PE5, SPI_DI, I2C_SDA
Z715	8	PE6, SPI_DO
Z716	9	PE7, CLKO
Z717	-	Gnd (Z726)
Z718	-	Gnd (Z726)
Z719	14	PB4, OC0A
Z720	15	PB5, OC1A
Z721	16	PB6, OC1B
Z722	17	PB7, OC2A
Z723	18	PG3, T1
Z724	19	PG4, T0
Z725	20	RESET_N
Z726	-	Power supply Gnd
Z727	-	Power supply input

Figure B.4: External interfaces

Radio interface

SPI is used as interface between the processor and the radio, with the following interconnections:

ATMega		nRF24L01		Description
Signal	pin	Signal	pin	
SS	10	CS_N	2	SPI Slave Select
SCK	11	SCK	3	SPI SCK
MOSI	12	MOSI	4	SPI MOSI
MISO	13	MISO	5	SPI MISO
ICP1	25	CE	1	nRF24L01 Chip Sel.
INT0	26	IRQ	6	Interrupt from Radio

Figure B.3: Radio interfaces

Mechanical and Mounting

Figure *B*.5 shows the top (component) view of the module. It can be mounted as a Surface Mount Device (SMD) directly onto a base PCB.

Energy Supply

Both the embedded processor and the radio are connected to the same supply rail. The supply voltage must therefore remain in a range that meet the supply voltage specifications of both devices, which is: 1.9V - 3.6V.

In order to reduce conversion noise of the ADC to a minimum, the power decoupling circuit is implemented following the guidelines in the datasheet of the ATMega645[13].

The energy consumption very much depends on the network parameters and application software. Under average conditions of a network cycle time of 1s, and a frame size of 32 bytes, the node can last for at least 5 years on a Lithium Thionyl Chloride battery of 900mAh.

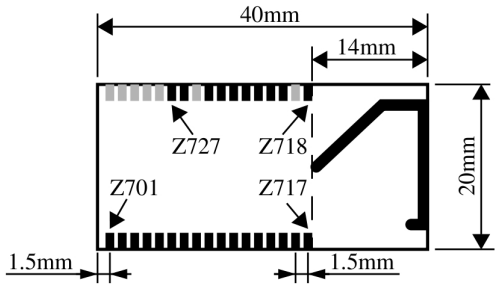

Figure B.5: MyriaNode dimentions

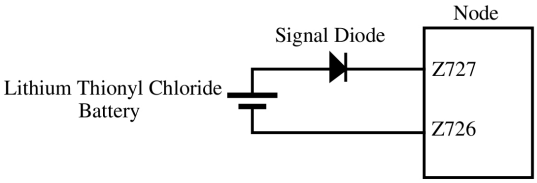

Figure B.6: Battery structure of a MyriaNode

Battery

The ER14250[14] from EMB is a Lithium Thionyl Chloride battery. It has a form factor of 1/2 AA size, and has a capacity of 1000mAh.

Programming

The node can be programmed and debugged via the JTAG interface. Although there are many second source suppliers of JTAG development tools, it is advised to use the AVR JTAGICE mkII from Atmel, ordering code: ATJTAGICE2[12].

Software

The software is documented by the other working groups of MyriaNed.

Bibliography

[1] A.Ebner and et al. "Decentralized Slot Synchronization in Highly Dynamic Ad Hoc Networks," Wireless Personal Multimedia Communications, vol.2, pp. 494- 498, 2002.

[2] C.Cordeiro and D.Agrawal. "Wireless sensor networks", Ad hoc and Sensor Networks Theory and applications. p.429-441. World Scientific Publishing. 2006.

[3] D.Gavidia, S.Voulgaris, and M. van Steen. "A gossip-based distributed news service for wireless mesh networks,". In Proceedings 3rd IEEE Conference on Wireless On demand Network Systems and Services (WONS), Les Menuires, France, January 2006.

[4] E.Anceaume and I.Puaut."A taxonomy of clock synchronization algorithms," IRISA Research Report No. PI1103, IRISA, 1997.

[5] F.Zhao and L.Guibas. "Time synchronization", Wireless Sensor Networks: an Information Processing approach. pp.107-108. Elsevier. 2004.

[6] H.Karl and A.Willig. "Introduction" in Protocols and Architectures for Wireless Sensor Networks. pp. 3-6. Wiley. July 2006.

[7] Dallas semiconductor."Crystal considerations with Dallas Real time clocks," Available online http://pdfserv.maxim-ic.com/en/an/AN58.pdf [Accessed on July 15,2008].

[8] Golledge frequency products."SW Watch crystal temp mil," Available online http://www.golledge.co.uk/pdf/products/xtl_sm/cc7v.pdf [Accessed on July 15,2008].

[9] G.Welch and G.Bishop. "The kalman filter," University of North Carolina. Avalable online http://www.cs.unc.edu/ welch/kalman/[Accessed on June 10, 2008].

[10] MyrianNed project. "Project description," Available online. http://www.chess.nl/en/home/Innovatie/Onderzoek/MyriaNed [Accessed on August 3, 2008].

[11] Nordic semiconductor. Datasheet.Available online http://www.nordicsemi.no/files/Product/data_sheet/Product_Specification_nRF24L01_1.0.pdf [Accessed August 3, 2008].

[12] Atmel. "AVR JTAGICE mkII," Available online. http://www.atmel.com/dyn/resources/prod_documents/doc2489.pdf [Accessed August 3, 2008].

[13] Atmel "8-bit Microcontroller with In-System Programmable Flash," Available online. http://www.atmel.com/dyn/resources/prod_documents/doc2570.pdf. [Accessed August 3, 2008].

[14] Nrgeurope. Battery datasheet. Available online. http://www.nrgeurope.com [Accessed August 3, 2008].

[15] J.Elson, L.Girod and D.Estrin. "Fine-grained network time synchronization using reference broadcasts," Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Boston, USA, 2002.

[16] J.Elson and D.Estrin. "Time Synchronization for Wireless Sensor Networks," In Proceedings of the 15th International Parallel and Distributed Processing Symposium. IEEE Computer Society, April 23-27. 2001.

[17] K.Romer. "Time Synchronization in Ad Hoc Networks," In Proceedings of the Second ACM International Symposium on Mobile Ad Hoc Networking and Computing, Long Beach, California. 2001.

[18] L.Meier,and L.Thiele. "Gradient clock synchronization in sensor networks," Technical report, Computer Engineering and Networks Laboratory. Swiss Federal Institute of Technology Zurich, 2005.

[19] "NTP Public Services Project," Available online http://support.ntp.org/bin/view/Main/WebHome [Accessed on June 12,2008].

[20] Company profile. Available online http://www.chess.nl [Accessed on August 6, 2008].

[21] Q.Yang, and et al. "A Decentralized Slot Synchronization Algorithm for TDMA-Based Ad Hoc Networks," Wireless Communications, Networking and Mobile Computing, Vol., Issue,21-25, pp. 1717 - 1721, 2007.

[22] Q.Yang and J.Shi. "An interference elimination method for decentralized slot synchronization in TDMA-based wireless ad hoc network," Intelligent Signal Processing and Communication Systems, pp.236-239, 2007.

[23] P.Anemaet. "Determining G-MAC potential with {S,L,SCP}-MAC," Masters thesis. Technische Universteit Delft. Delft. August 2008.

[24] R.Tjoa and et al. "Clock drift reduction for relative time slot tdma-based sensor networks," Personal, Indoor and Mobile Radio Communications, Vol.2, 5-8 , pp.1042 - 1047. Sept. 2004.

[25] R.John. "Introduction to Quartz Frequency Standards," Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate. October 1992.

[26] R. Fan and N. Lynch. "Gradient clock synchronization," In Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, pp.320-327. ACM Press, 2004.

[27] Symmetricom, Inc. "Stochastic Model Estimation of network time variance," Available online http://www.symmttm.com/pdf/Network_Timing/wp_Stochastic_Model.pdf [Accessed August 3,2008].

[28] S.Raje and Q.Liang. "Time synchronization in Network-centric sensor networks," Radio and Wireless Symposium IEEE, pp.333-336. 2007.