Eindhoven University of Technology

Department of Electrical Engineering

Telecommunications and Electromagnetism

Decentralized frame synchronization of a TDMA-based Wireless Sensor Network

Fasika A. Assegei

July 24, 2008

Master of Science Thesis

Project period: December 2007 – August 2008

Department: Radiocommunications (ECR)

Supervisors:

Frits van der Wateren (Chess B.V.)

dr.ir. P.F.M. Smulders (TU/e)

Acknowledgment

Abstract

Time synchronization is a crucial component of infrastructure for *Wireless Sensor Networks*(WSNs). Most applications of WSNs make extensive use of time synchronization mechanisms like *Time Division Multiple Access* (TDMA) scheduling, accurate timestamping of events, coordinate activities of the network or data fusion. The unique requirements of wireless sensor networks, compared to traditional networks, in terms of precision, lifetime, energy and scope of the synchronization achieved, make the traditional synchronization methods unsuitable for WSNs. This motivates the research of synchronization methods for WSNs which are aligned to the specific properties of WSN. In this research, different algorithms have been developed to achieve a stable, convergent and energy-efficient synchronization of a decentralized WSN. The algorithms achieve synchronization by using the phase error of a node's wake-up time with that of the neighboring node's, without actually exchanging the information about the clock time of the sender. So, the method avoids time keeping on the messages(time stamping) which reduces the message overload. The algorithm can be integrated with the slot allocation algorithm to form the *Medium Access Control*(MAC) layer protocol for a better throughput. The research is concluded with the comparison of 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 for selecting the algorithm.

Contents

List of Figures

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. *Wireless Sensor Networks*(WSNs) are distributed networks of such sensors, dedicated to closely observing real-world phenomena. Such sensors may 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 assemble a broader picture of the environment than what each sensor individually senses. Various applications are realized using sensor networks[13]. 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 clock 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 traditional networks.

Collaboration among nodes is often required for different purposes[14]. 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: for a proper *Time Division Multiple*

Access (TDMA) scheduling, for database queries, cryptography and authentication schemes, coordination of future action, interaction with users and ordering logged events during system debugging[15].

1.2 Existing Work on WSN clock synchronization

Several algorithms have been proposed and researched for time synchronization in WSNs. The *Reference Broadcast Synchronization*(RBS) stated in [1] 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 good precision of a few microseconds. It also provides clock frequency estimates between two receivers using a linear regression technique. But the message overhead is very large. [2] presents a decentralized slot synchronization algorithm based for TDMA networks which uses the topology of the nodes as a means to weigh the phase error of the sender with the receiver.

Another method for achieving a network-wide synchronization is suggested by [16]. This approach establishes a table to correspond the clock of the sender with that of the receiver clock so that a good estimation of the neighbors clock is achieved. Different estimation techniques are presented. Another approach is used in [3] to use the metaphor of fireflies to the existing problem of synchronization. The method bases itself for the synchronization of the network where the nodes are on the same domain, firing at the same time. [8] presents a different approach to weight based synchronization for interference elimination for a TDMA based ad-hoc networks. The algorithm achieves synchronization in a decentralized manner using the nodes offset with its neighbors with a goal of eliminating the interference.

1.3 Objective and overview of the research

There are many schemes presented to provide time synchronization for the wireless sensor networks. It was observed that some of the schemes use a central node in order to achieve a synchronization with the nodes. Some methods are used to build a table corresponding to the clock of the sender and the receiver using timestamping the messages. Also, if the nodes are subjected to severe changes in environmental conditions, then the accuracy of these short-term synchronization schemes might suffer. In addition to that, a small scale synchronization is dealt in a decentralized manner for slot synchronization.

The primary objective of this research is to develop an algorithm to achieve a long time synchronization of a WSN in a low cost and energy-efficient method which is decentralized and employs no timestamping on the messages. The methods use the phase errors between the receiver and its neighbors, without actually estimating the neighbors clock as a way of achieving long-term time synchronization in sensor networks. In this research, different algorithms are presented to achieve a long term decentralized synchronization of a WSN and compared in terms of performance and energy consumption. Through integration with the MAC layer protocol, a better throughput can be obtained. These algorithms have the following characteristics: provides high precision, adaptive to environmental effects, energy-efficient and achieve long-time synchronization.

The remainder of the paper 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 formulate the problem. Chapter 4 presents the mathematical models of the algorithms for the synchronization of the network. In Chapter 5, the simulation results are presented and the analysis of the results in addition to the comparison of the methods with respect to energy consumption is 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[1] created a MAC protocol for gossip communication, gMAC, for the wireless sensor network, MyriaNed[2], 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 nodes time synchronized. A number of approaches have been introduced for different multiple access techniques, of which *Carrier Sense Multiple Access*(CSMA) and *Time Division Multiple Access*(TDMA) are the most common. One problem with CSMA is the times 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

[1]Chess is specialized in innovative technical systems and business critical solutions. Chess creates and develops sophisticated solutions and products for electronic systems, transactions and electronic payments, Machine to Machine (M2M) systems, sensor applications and digital multimedia. Chess is located in Haarlem, the Netherlands.

[2]MyriaNed is a project to create a large functional network of 10,000 nodes with wireless communication for research on protocols, power management, programming models, and security.

time-slot and listen in the others. To reduce the idle listening time even more, the listening nodes can turn off the radio for the time between the end of the transmission and the start of a new slot. 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 algorithm is being conducted alongside this research[7].

2.2 Clock Synchronization in traditional 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 traditional networks, *Network Time Protocol*(NTP) being the most popular[4].

Infrastructure advantage being the most important factor, the *Network Time Protocol* (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 NTP is a tiered time distribution system with redundancy capability. 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. Figure 2.1 shows the structure of the NTP protocol when it is used to synchronize client computers.

The tiered nature of the NTP time distribution tree enables a user to choose the accuracy

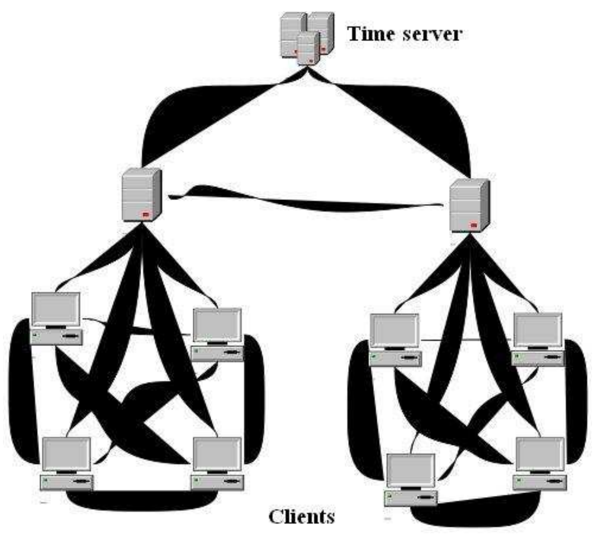


Figure 2.1: A simple NTP diagram

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 *Coordinated Universal Time* (UTC) time standard.

Synchronization in traditional distributed systems takes a different kind of approach since there is no central authority to be referred in case of time requests. But 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. Creation of a synchronization scheme that satisfies such requirements is challenging. The task becomes particularly daunting in WSNs, in light of their additional domain requirements, which are mentioned below. These complications are startling to look for a new arena for the synchronization of WSNs.

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 reasons are described below.

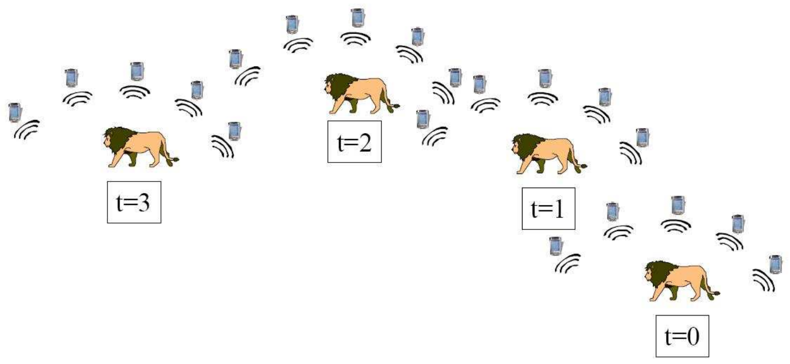


Figure 2.2: Tracking the movement of a moving object

Data Integration

The signal processing literature sometimes refers to this as array processing; with heterogeneous sensors, it is often called *data fusion*.
There are many applications, such as signal enhancement (noise reduction), source localization, process control, and source coding. 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.

TDMA schedule

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 (usually radio) 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 his own time slot. This allows multiple stations to share the same

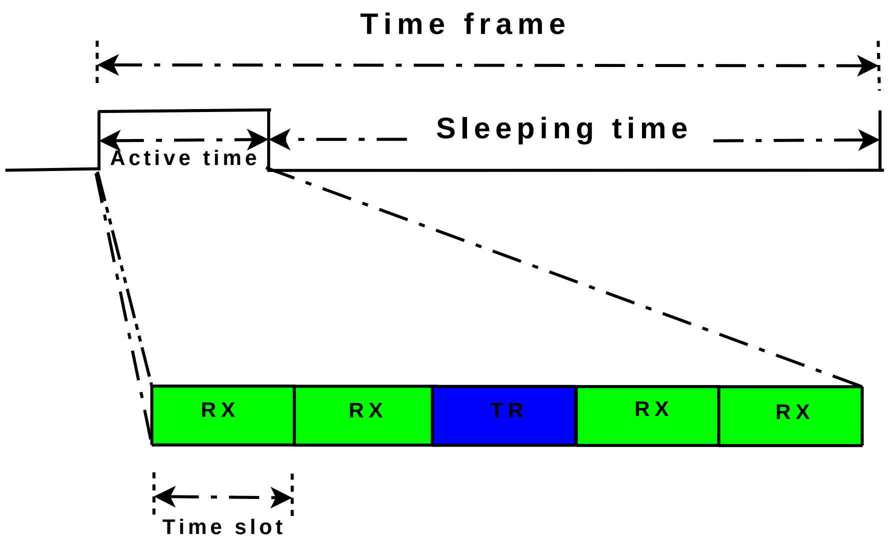


Figure 2.3: TDMA frame

transmission medium while using only the part of its bandwidth they require. Figure 2.3 shows how a time frame is divided into receiver and transmitter slots. In the time frame, the active slot is divided into one transmitting slot and many receiving slots.The number of visible slots, five, have been picked arbitrarily in the figure and do not map to the selection in the simulation.

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.

The MAC protocol used in MyriaNed is the gMAC[7]. The g in gMAC comes from Gossiping, which represents the gossip protocol implemented in the network layer.

In the protocol, time is divided into a number of different slots as shown in Figure 2.4. Each neighbor has a different transmitting slot so that there will be no collision. Thus, a message transmitted should be in received in the receiving slot of the neighboring nodes. In order to

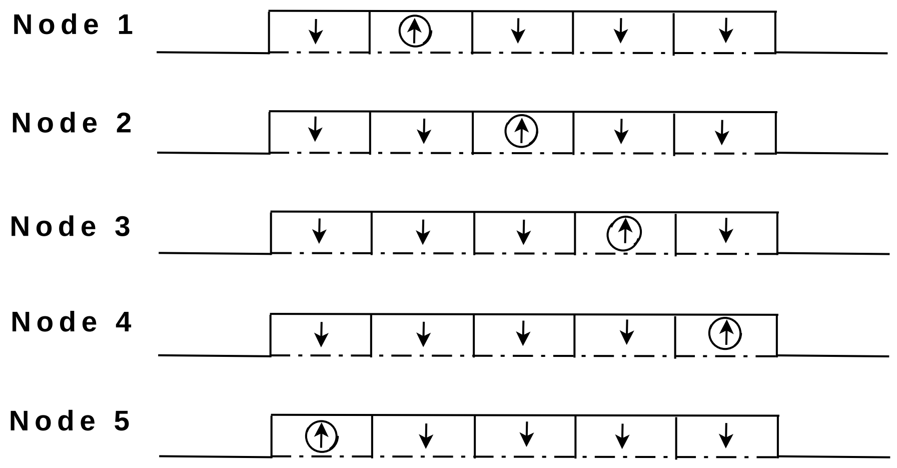


Figure 2.4: TDMA Channel access

have a seamless communication between the nodes, the synchronization of the frames is a necessary part of the MAC protocol. The processing element and other functions on a WSN operate on a local oscillator. Due to physical factors, the frequency of the oscillator has a drift. When no provisions are taken, it causes the nodes to run out of sync. Given that there is a certain error, the node will adjust its wake-up time at the end of the last receiver slot with the offset calculated with an algorithm. As mentioned before, networks can drift out of sync with each other, forming multiple subnets. Hence, the number of slots is chosen accordingly. 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 fixed length of time which is divided into a communication time and sleeping time. 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 neighbor 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 communication quality of service is guaranteed by the gossiping layer probabilistic features. 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 (e.g. GSM). However, such structure is not used in the network. The scheduling of the TDMA slots is made locally by reaching an agreement between direct neighbors. Whenever a collision is detected by a node, it changes the local variables trying to solve this issue for the next transmission time. Each node has a crystal oscillator that has a certain accuracy, which results in time drifts. This causes the TDMA time slot boundaries to drift when we compare two nodes with the same schedule.

2.4 Wireless Sensor Networks: Why different ?

Are traditional synchronization methods applicable in the case of WSNs? Many assumptions in the traditional 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 number of years before they run out of power. An energy efficient method is a requirement for WSNs.

- **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 *Global System for Mobile* communications (GSM). In a WSN, network dynamics results from various factors like mobility of nodes, node failures, environmental obstructions etc., 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. And while a few require absolute or real time as reference, for others, a relative notion of time is enough.

- **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, we can obtain very good synchronization with *Global Positioning System* (GPS) receivers, but it will be unreasonable to put an expensive and energy-hungry GPS receivers on a sensor node. This is too costly.

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

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 in clocks of nodes or in synchronization algorithm can be divided into two main categories:

1. **Oscillator Characteristics**: The sensor node's clocks run on very cheap oscillators. The following two characteristics are the main sources of errors between the clocks of two different nodes.

   Accuracy: This is a measure of difference between oscillators expected (ideal) frequency and actual frequency. It is also called as the calibration tolerance of the clock; its maximum is specified by the manufacturer.

   Stability: This is oscillator 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 receivers network interface.

All the above factors result in following errors between the clocks of two nodes:

Phase error: The oscillators of any two nodes can be out of phase at any given time, resulting into different time on both clocks. There can be some initial time-offset between nodes at the start of a synchronization procedure. Phase error is an instantaneous metric. It describes how far apart are the clocks at a given time.

Frequency error: Frequency error, in contrast, 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, because clocks drift from their initial frequency. 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.

Even if two clocks are assumed to have the same frequency and no drift, the above mentioned delays in the network result in phase offset 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, and dispersion amongst their clocks.

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 clocks' 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; 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 their stability and accuracy increase, so do their power requirements, size, and cost, all of which are important in sensor networks. Most commonly found in computer clocks are quartz crystal oscillators, characterized by [8]. 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 to it, the ambient temperature, acceleration in space (e.g., shock or attitude 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 [8] [11]. 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 [20] [21] is used in the MyriaNode[1].

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

3.3 Clock Drift and Delay

In this section, we will see the clock drift 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 voltage). One can usually assume that the clock drift of a computer clock doesn't exceed a maximum value ρ . This means that it can be assumed

$$1 - \rho \leq \frac{dC(t)}{dt} \leq 1 + \rho, \tag{3.4}$$

where ρ represents the maximum clock drift. Figure 3.1 shows the relationship between clock time and absolute time for different drift rate ρ.

A typical value for ρ achievable with our crystal clock is $30ppm$ which means that the clock drifts away from real time by no more than 30 seconds in ten days, which is still a significant value. Note that different clocks have different maximum clock drift values ρ.

The frequency of the clock is dependent on many factors [16] and is given as

$$f_i(t) = f_o + \Delta f + a(t - t_o) + \Delta f_e(t - t_o) + f_r(t) \tag{3.5}$$

where

t_o = the start time of the clock,

a = aging factor,

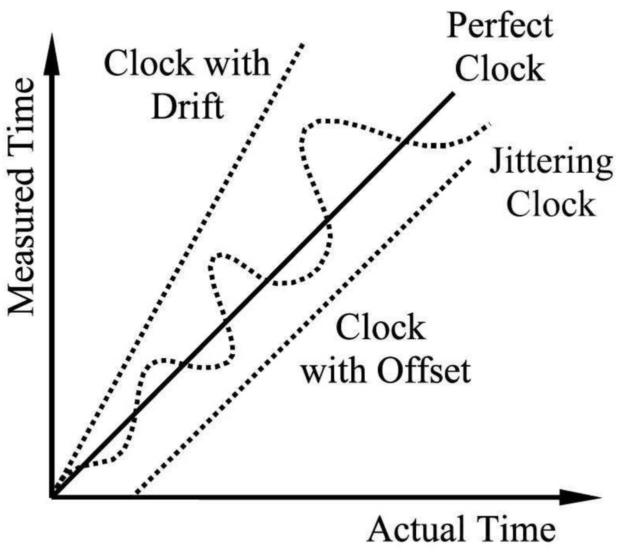


Figure 3.1: Measured time versus absolute time

f_o = nominal frequency,

Δf = calibration error,

$f_r(t)$ = frequency instability (noise) term,

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

Hence, all the above factors affect the value of ρ. The nominal frequency can also be identified as the ideal frequency at which the oscillator is supposed to run. From (3.3) and (3.5), we get

$$C_i(t) - C_i(t_o) = \frac{1}{f_o}\int_{t_o}^{t} f_i(\tau)d\tau, \tag{3.6}$$

$$\begin{aligned} C_i(t) - C_i(t_o) &= \frac{1}{f_o}\int_{t_o}^{t} [f_o + \Delta f + a(\tau - t_o) \\ &\quad + \Delta f_e(\tau - t_o) + f_r(\tau)]d\tau, \end{aligned}$$

$$\begin{aligned} C_i(t) &= C_i(t_o) + (t - t_o) + \frac{\Delta f}{f_o}(t - t_o) \\ &\quad + \frac{a}{2f_o}(t - t_o)^2 + \frac{\Delta f_e}{2f_o}(t - t_o)^2 + \frac{1}{f_o}\int_{t_o}^{t} f_r(\tau)d\tau. \end{aligned}$$

The amplitude of the short term variations due to noise (clock jitter) is small enough that they do not cause the clock to accelerate or decelerate in a very large amount in the long run.

But the environmental term, primarily due to temperature, and the error due to calibration can be significant. Thus, the phase or time offset between two clocks at any given time results from: a combination of initial phase offset, frequency bias, calibration error, frequency drift and the environmental terms. As more time passes from the synchronization point, the drift and environmental terms become more significant. The resulting expression is used to model the clock drift of an oscillator.

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

Message Delay: The other source of error which arises between the wireless sensor nodes is due to the delay in the transfer of messages between the sender and the receiver node. This is included in the system and networks category of the errors that the clock doesn't have any effect on it. This delay is comprised of the following:

- **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. This delay is affected by the hardware capacity of the nodes. It is the time it takes to compose a message and send it to the channel. A delay of $1clk$ is taken for the aggregate send and receive time of the node.

- **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. Upon the implementation of the WSN, the propagation delay between nodes which have a radio range of $\approx 10m - 30m$ is in the range of $100ns$. For practical reasons, it can be neglected in the simulation since the clock cycle of the clock in the Myrianode is $\approx 30\mu s$.

- **Receive Time**: Time needed for processing at the receivers network interface. This delay is also affected by the hardware capacity of the node.

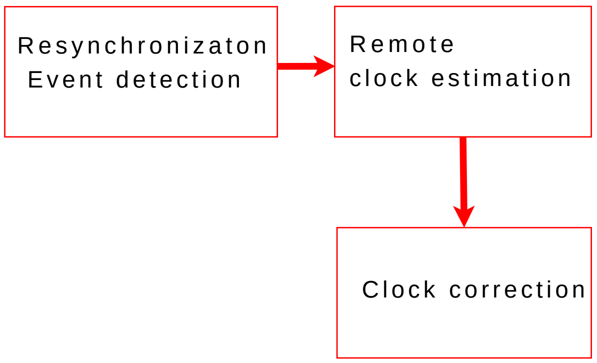


Figure 3.2: Building block of a Time synchronization algorithm

3.4 Performance metrics of a time synchronization protocol

The time synchronization protocols can be decomposed into four conceptual building blocks$cite$11. It is shown in Figure [5].

- 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.

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

3.4.1 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 that synchronization error is with some probability smaller than a prescribed bound), the mean error, the error variance or some other quantity is relevant.

3.4.2 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.

3.4.3 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. In addition, the read/write cycle of the EPPROM is also affected with the continuous access for the synchronization, which reduces its life time.

3.4.4 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?

Chapter 4

Synchronicity protocol

4.1 Requirements of the Algorithm for MyriaNed

Precise synchronization

Time synchronization contains two parts: 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 in 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 several years. In our implementation here, 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.
In this research, exploration of a global synchronization algorithm is conducted. Hence, 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, network time synchronization simply means adjusting the clock to a correct time from an outsider. 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 running at a particular frequency known previously. 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 scheduling. 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 large batteries and run all the time; others are so constrained that they only wake up occasionally, take a single sensor reading, and transmit it then immediately returning to sleep.

Convergence Time

The importance of convergence time as a metric is directly tied to the need for energy efficiency. If energy would not be a concern, the chosen form of synchronization could simply be left running continuously. Though convergence time might be important due to its effect on network start-up, it would make no difference in the steady-state. However, in this energy-constrained world, systems are often forced to power down as many services as possible for as long as possible. This makes convergence time important to consider.

Simplicity

Extra overhead in the message should be avoided. A timestamp is not needed in the frame in order to decrease the message overload. Thus, this in turn has an advantage in the energy-constraint WSN, although it passes most of the burden to the synchronization algorithm to determine the next wake-up time of the node.

4.2 Synchronization Frequency

In achieving frame synchronization in a TDMA scheduling, a guard time is given for a fault tolerance which is used to accommodate the phase errors, as shown in Figure 4.1. The message has the error tolerance of a guard time. 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.

Decreasing the timing of the synchronization and do periodic execution of the synchronization of the 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.

A time slot can be defined as being expressed in a number of clock cycles as

$$t_{slot} = kT, \tag{4.1}$$

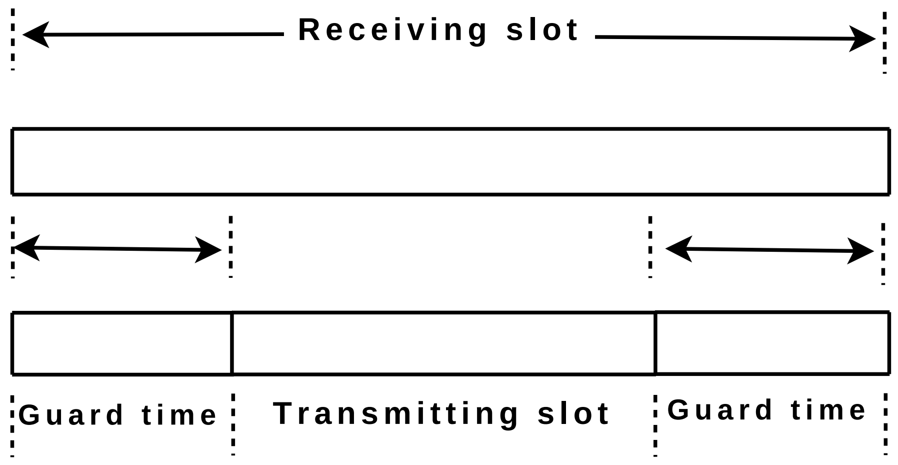


Figure 4.1: Guard time of the nodes

where T is the period of the time frame in clock cycles and t_{slot} is the time duration of a TDMA slot and k is the duty cycle.

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

$$t_{diff} = 2\frac{T_{sync}}{T}\rho, \tag{4.2}$$

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

Since the relative time difference between two nodes can be in two direction, the guard time needs to be twice t_{diff},

$$t_{guard} = 2t_{diff} = 4\frac{T_{sync}}{T}\rho. \tag{4.3}$$

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

$$t_{slot} \geq t_{guard} + T_{tx}, \tag{4.4}$$

where T_{tx} is a system constant and it represents the required time to send a packet from one node to other.

Each message that is send within a time slot is exactly received by a neighbor at a known clock tick number

$$tick_{rx} = T_{tx} + \frac{t_{guard}}{2} \tag{4.5}$$

Hence, each time when a node receives a message, it has to be received exactly at the clock tick given by (4.5). Whenever this number is not equal with the desired one, we record the slot tick counter value. This easy mechanism adjusts the clock drifts at the receiver side. The resolution is given by the clock frequency and complies to the following proposition: the faster the clock , the higher the resolution.

Two nodes have a good communication link when they synchronize their clocks at least once every T_{sync} that gives the following relation,

$$\frac{T_{sync}}{T} \geq 1. \tag{4.6}$$

It is a good practice to design systems having a T_{sync} greater than the time frame. It increases the lifetime of a node. However, this degrades the tolerance of the system. To obtain the value of T_{sync} for a particular system with a given duty cycle, we have to solve the next problem

$$t_{guard} \geq 4\rho \frac{T_{sync}}{T} \tag{4.7}$$

$$t_{slot} \geq t_{guard} + T_{tx} \tag{4.8}$$

$$\frac{T_{sync}}{T} \geq 1 \tag{4.9}$$

There is a trade-off in determining T_{sync}: increasing T_{sync} reduces the energy costs of synchronization, on the other hand we increase the cost of guard time and decrease the network performance.

4.3 Lower bound of Synchronization

As in traditional distributed clock synchronization, a network of nodes equipped with hardware clocks with bounded drift is considered here. Nodes compute logical clock values based on their hardware clocks and information that they have, and the goal is to synchronize the nodes' logical clocks as closely as possible, while satisfying certain validity conditions.

There are different approaches towards the lower bound that a time synchronization algorithm achieve. In [9], the clock drift is taken to be zero and the delay impact on the lower bound of the synchronization algorithm is presented. But a more realistic approach is presented

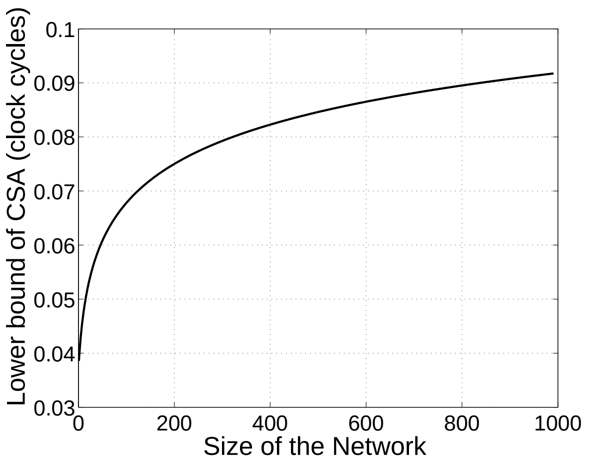


Figure 4.2: Network size and the lower bound of Synchronization

in [10]. Gradient clock synchronization is shown to require that the skew between any two nodes' logical clocks be bounded by a nondecreasing function of the uncertainty in message delay (call this the distance) between the two nodes. Nearby nodes are required to be closely synchronized, and allow faraway nodes to be more loosely synchronized. Hence, the result is that the worst case clock skew between two nodes at distance d from each other is

$$\Delta L \geq \frac{\tilde{\rho} d}{8(1+\tilde{\rho})} \frac{log(n-1)}{log(\frac{8(1+\tilde{\rho})}{\tilde{\rho}} log(n-1))}. \tag{4.10}$$

where d is the distance between two neighboring nodes and n is the number of nodes in the network. This means that clock synchronization is not only a local property, in the sense that the clock skew between two nodes depends not only on the distance between the nodes, but also on the size of the network.

Our lower bound implies as in the case of MyriaNed, that the TDMA protocol with a fixed slot granularity will fail as the network grows, even if the maximum degree of each node stays constant. As shown in Figure 4.2, as the network size grows, the lower bound of the synchronization also increases.

4.4 Mathematical Model

As it is described earlier, the wake up time of the nodes should be synchronized in such away that the nodes are synchronized in the long term, as the TDMA scheduling requires. As part of the message, the nodes are transmitting the slot number which it is transmitting. This information is used in the calculation of the time that the message is sent.

The difference between the transmitting times of node i and node j is

$$\Delta t_{ij}^{(n)} = t_i^{(n)} - t_j^{(n)}, \tag{4.11}$$

where t_i and t_j are the wake-up times of node i and node j at the n^{th} period. The wake-up time of a node at a random time after n periods of firings after it is turned on is

$$t_i^{(n)} = \sum_n T_i^{(n)} + t_{io}, \tag{4.12}$$

where $T_i^{(n)}$ is the period of the crystal clock at the n^{th} period since it changes with time according to (3.5) 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 (3.5).

The difference in the wake-up time of the nodes is given by

$$\begin{aligned} \Delta t_{ij} &= \sum_n T_i^{(n)} + t_{io} - (\sum_n T_j^{(n)} + t_{jo}) \quad (4.13) \\ &= (\sum_n T_i^{(n)} - \sum_n T_j^{(n)}) + (t_{io} - t_{jo}). \quad (4.14) \end{aligned}$$

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 using two approaches, which are not exclusive. The first one is adjusting the clock frequency to come up with a better wake-up time. The second one is to adjust the next wake-up time depending on the difference between the current wake-up time of the node and its neighbors. In this research, the second option is explored due to:

- The cost of adjusting the frequency of the clock: This is more expensive (hardware based) than adjusting the next wake-up time of the node.
- The complexity of the implementation: This is larger which in turn results in more energy consumption.

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

$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - \xi_i^{(n)}, \tag{4.15}$$

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

$$\xi_i = f(\Delta t_{ij}). \tag{4.16}$$

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

4.4.1 Median algorithm and the setback

In this section, we will see the Median algorithm which is currently implemented in the MyriaNode. 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 offset to any other node j in the neighborhood as the difference of the phase offsets implied by nodes i and j. That is, given n is the number of periods after the node starts functioning

   $$\Delta t_{ij}^{(n)} = t_i^{(n)} - t_j^{(n)}, \tag{4.17}$$

   where t_i is the wake-up time of node i and t_j is the wake-up time of node j.

4. Receivers compute the median of the offsets

   $$\xi_i^{(n)} = m(\Delta t_{ij}^{(n)}), \forall j \tag{4.18}$$

5. Receivers adjust their wake-up time by the computed offset value,

   $$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - k\xi_i^{(n)}, \tag{4.19}$$

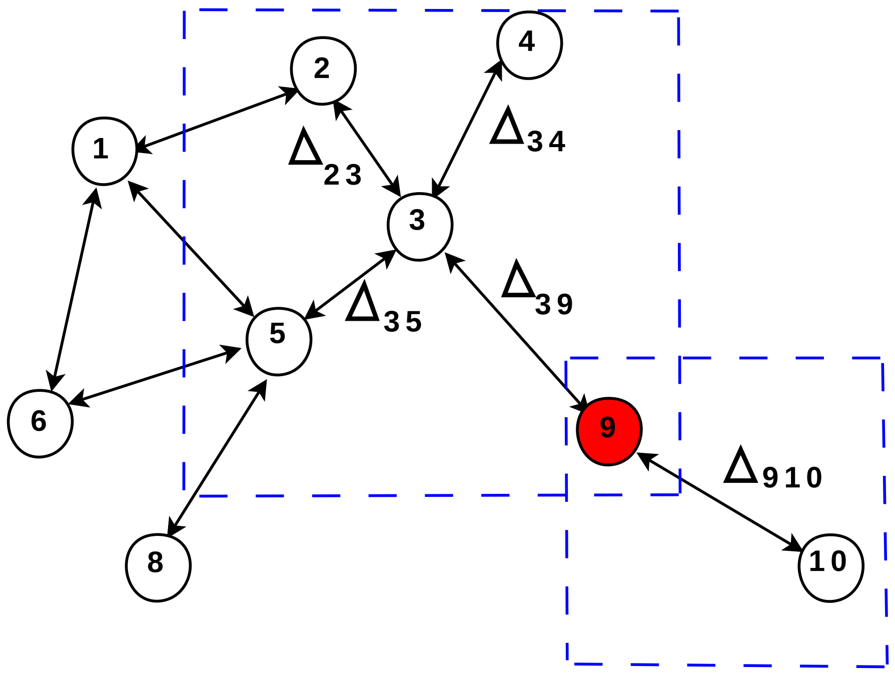


Figure 4.3: A WSN scenario

where k is the *gain factor*. By multiplying the median with a gain factor, i.e. $k\xi_i^{(n)}$, the output can be adjusted for better performance. Hence a gain factor could be added to ensure better precision of the synchronization error.

As per the application of Median, there are setbacks 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 4.3, 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 wake-up time with the median of the offsets from its neighbors, Node 3 and Node 10. Adjusting the wake-up time, we get the offset to be

$$\xi_9 = \frac{\Delta t_{910} + \Delta t_{39}}{2}. \tag{4.20}$$

With Node 10 being isolated from the well established network, it has a major drift from the other nodes. Being in sync with the other nodes, Node 9 will drift away from the network

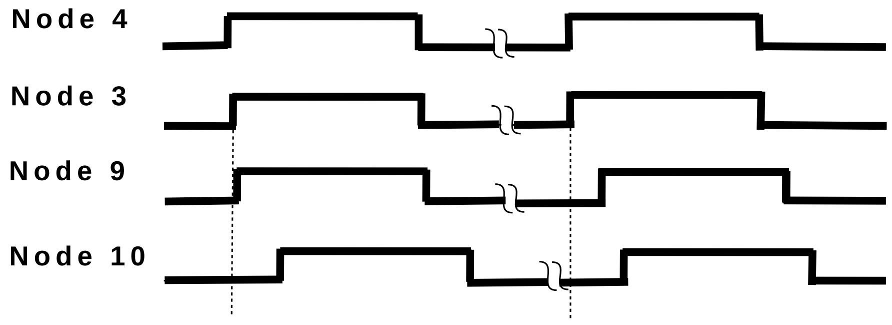


Figure 4.4: Using median for phase error correction

after its adjustment with the Node 10. It will take more time to synchronize with the network again. Hence, the performance of the median algorithm decreases with the dynamic nature of the network. Figure 4.4 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 Node 10.

So, in order to address this problem, a range of algorithms are explored in the next subsections to realize an energy-efficient, more precise and simple frame synchronization.

4.4.2 Weighted Measurements - Interference-phobic approach

One form of approach to tackle the dynamic behavior of a wireless sensor network, due to channel conditions as well as collisions is a Weighted Measurement (WM) approach. Using this approach, different weight is given to the different measurements. A weight is added to the 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. Its a metaphor of "The Majority Wins". Hence,

$$\xi_i^{(n)} = \sum w_{ij}^{(n)} \Delta t_{ij}^{(n)}, \tag{4.21}$$

where $\sum w_{ij}^{(n)} = 1$.

The weighted adjustment is used to modify the local wake-up time of the node,

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

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

The main task in this algorithm is how to choose the weight factors so that the stability of the network (timewise) is achieved in a faster time. In order to see how the weight factors should affect the next wake-up time of the node, we will discuss scenarios.

The weight is selected by the fact that the a node joining a network should adjust its time with the network that it is joining. After each measurement, a phase error is associated with the tabled values to recognize how far is the sending node concerning the time that it drifted away from the receiving node. The weights are calculated as:

$$\delta_{ij} = ae^{-b\Delta t_{ij}}. \tag{4.22}$$

The parameters are selected using the initial conditions, $\delta_{ij} = 0.1$ for $\Delta t_{ij} = t_{guard}$ and $\delta_{ij} = 1$ for $\Delta t_{ij} = 0$.

As shown in Figure 4.5, the closer the phase error is to zero, the higher weight it is given. The assignment of high values to the small phase errors will decrease the time it takes to synchronize this node with the network.

In the assignment of the weight factor, there is another issue to be considered. If all the phase errors are in the lower region, this corresponds to the fact that the node is a newly joining node to a well-established and synchronized network. In order to incorporate the effect of the distribution of the phase errors, another quantity, δ, is introduced in the calculation of the weight factors so that the node will adjust its wake-up time towards the more stabilized network. This parameter expresses the distribution of the weight factors ,

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

Hence, the weight moves towards the large phase errors if the node is a new-comer which wants to join the "already established" network.

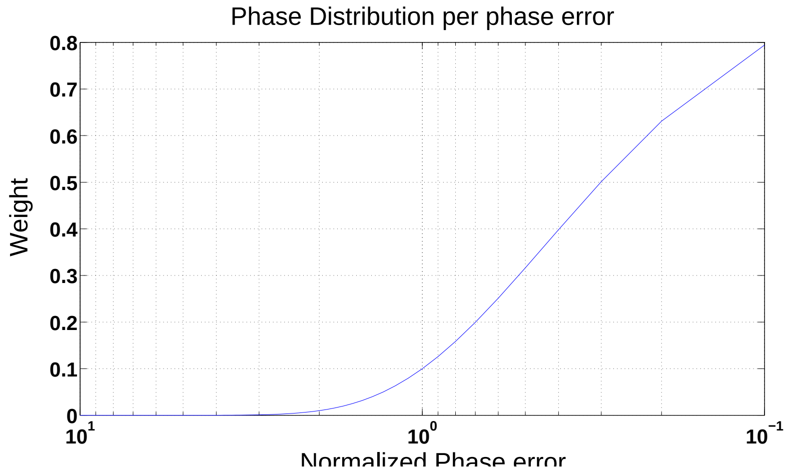


Figure 4.5: Weight factors for the phase error distribution

Hence, WM is a two step process in which the first one describes how the nodes is offsetted from its neighbor whereas the second one describes how the node is positioned in the network topology which surrounds it.

Using the weighted approach, a series of measurements will be used to estimate the next wake-up time of the node, giving less value/emphasis to the nodes which are out of reach from the other nodes. The simulation result is shown in the next section to see the effect of the algorithm in comparison to the other algorithms.

Algorithm for WM

```
1. Nodes broadcast packets.
2. Each receiver records the time that the packet is received.
3. Each receiver i computes its phase error to any other node j in the neighborhood.
4. Each receiver i computes the weight factor for the corresponding phase error.
5. Each receiver i computes the phase offset using the weigh factor method.
6. Each receiver adjusts its wake up time using the phase offset.
```

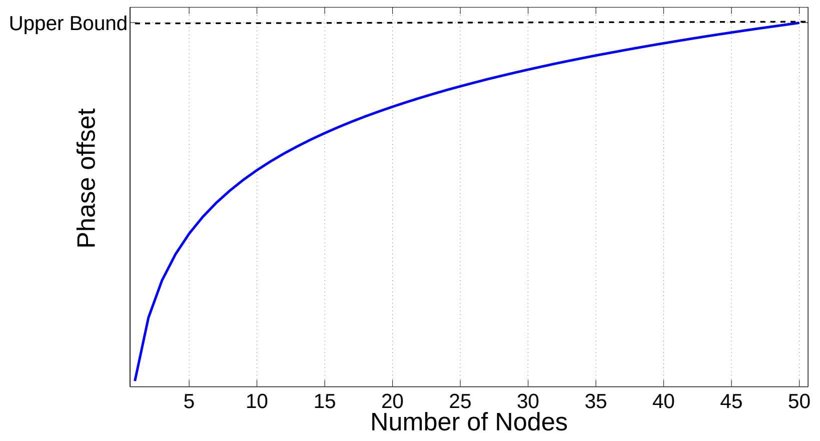


Figure 4.6: Curve fitting using logarithmic function

4.4.3 Non-Linear Curve fitting - Interference elimination

Non-Linear Curve fitting

In this section, another approach is used for synchronization of the nodes.

As the phase errors are non-linear by their stochastic nature, the curve fit should also be a non-linear curve. Normalized offset values tend to be small and follow a curve in such away that the maximum value of the offset should not be greater than the threshold value that was intended to be. See Figure 4.6.

Non Linear lease squares Curve Fitting (NLCF) is a mathematical procedure for finding the best fitting curve to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve. In order to fit the curve, we have chosen the logarithmic curve in order to meet the demand of adjusting the time offset and stabilize the network. Logarithmic curve fitting calculates the best fitting logarithmic curve for a given set of data. A logarithmic function can be used to represent the distribution of the offsets in the neighborhood,

$$f(x_i, \beta) = \beta_1 + \beta_2 log(x_i), \tag{4.23}$$

where β_1 and $beta_2$ are the parameters to be estimated.

We have a set of n data points(corresponding to n neighbors) which are $(x_1,\ y_1)$, $(x_2,\ y_2)$,. . .,$(x_n,\ y_n)$, and a curve (model function) $y = f(x, \beta)$, that in addition to the variable x

also depends on n parameters,

$$\beta = (\beta_1, \beta_2). \tag{4.24}$$

It is desired to find the vector β of parameters such that the curve fits best the given data in the least squares sense, that is, the sum of squares

$$S = \sum_{i=1}^{m} r_i^2, \tag{4.25}$$

is minimized, where the residuals r_i are given by

$$r_i = y_i - f(x_i, \beta) \tag{4.26}$$

for i=1,2,. . ., n.

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

$$\frac{\partial S}{\partial \beta_j} = 2\sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0 \ (j = 1, 2). \tag{4.27}$$

In a non-linear system, the derivatives $\frac{\partial r_i}{\partial \beta_j}$ 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,

$$\beta_j^{k+1} = \beta_j^k + \Delta\beta_j. \tag{4.28}$$

Here, k is an iteration number and the vector of increments, $\Delta\beta_j$, is known as the shift vector. At each iteration, the model is linearized by approximation to a first-order Taylor series expansion about β^k.

$$f(x_i, \beta) \approx f(x_i, \beta^k) + \sum_j \frac{\partial f(x_i, \beta^k)}{\partial \beta_j} \left(\beta_j^k - \beta_j\right) \tag{4.29}$$

$$f(x_i, \beta) = f(x_i, \beta^k) + \sum_j J_{ij} \Delta\beta_j. \tag{4.30}$$

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,

$$\frac{\partial r_i}{\partial \beta_j} = -J_{ij} \tag{4.31}$$

and the residuals are given by

$$r_i = \Delta y_i - \sum_{j=1}^{j=n} J_{ij} \Delta \beta_j, \tag{4.32}$$

where

$$\Delta y_i = y_i - f(x_i, \beta^k). \tag{4.33}$$

Substituting these expressions into the gradient equations and equating to 0, we get a matrix notation of

$$\left(J^T J\right) \Delta \beta = J^T \Delta y \tag{4.34}$$

When the observations are not equally reliable, as in case of WSNs which are being studied, a weighted sum of squares may be minimized using the weights(Figure 4.5),

$$S = \sum_{i=1}^{i=m} W_{ii} r_i^2. \tag{4.35}$$

The normal equations are then

$$\left(J^T W J\right) \Delta \beta = J^T W \Delta y. \tag{4.36}$$

Model Design

As we are developing a decentralized algorithm, the next wake-up time of the node depends on the current offset that it has with the other nodes. The distribution of the offset is the crucial factor in deciding the type of curve which we want to fit in. The offsets shows how many time units that the neighbor node is out of touch with the node in focus. The larger the offset is, the more out-of-sync the node is. This happens because of

- nodes joining the network
- interference from the environment
- clock drift
- mobility of Nodes

Since these nodes might have been long enough without synchronizing, the time offset that they are going to have is large compared to the nodes which were in the neighborhood during

the last round of communication. Thus, to join the mass of the network rather than the mass joining the single node, the algorithm is required to push the synchronization towards the more established network, rather than away.
The set of data are the measured time offsets of the node and the parameters β_1 and β_2 are estimated using a least squares approximation.

$$f(x_i, \beta) = \beta_1 + \beta_2 log(x_i), \tag{4.37}$$

The initial values of the parameters β_1 and β_2 is estimated taking into account the state of the network to be. Hence, for the perfect function of log estimation. i.e.

As each measurement arrives from the neighbors, an iteration is made in such a way that the measurement error from a pre-determined offset value is reduced. This ensures that the phase offset to be added in the next wake-up time doesn't diverge in a large amount from the predicted value. Hence, the difference in measured values and thus the offset is considered big. In the desire to stay in the originally stabilized network, the nodes tendency to adapt a new environment is very low, in this case resisting any change that is going to happen to its wake-up time.

```
Algorithm for NLCF
1. Nodes broadcast packets.
2. Each receiver records the time that the packet is received.
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 phase offset using the function.
6. Each receiver adjusts its wake up time using the phase offset.
```

4.4.4 Discrete time Kalman Filter for synchronization

Discrete time Kalman Filter

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) 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 (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step.

The Kalman filter addresses the general problem of trying to estimate the state x of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Hx_{k-1} + Bu_k + w_{k-1}, \tag{4.38}$$

with a measurement z that is

$$z_k = Hx_k + v_k. \tag{4.39}$$

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

$$p(w) \approx N(0, Q), \tag{4.40}$$

$$p(v) \approx N(0, R). \tag{4.41}$$

With the initial estimates of x_{k-1} and P_{k-1}

$$x_k = Hx_{k-1} + Bu_k \tag{4.42}$$

$$P_k = HP_{k-1}H^T + Q \tag{4.43}$$

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.

$$K_k = P_kH^T(HP_kH^T + R)^{-1} \tag{4.44}$$

$$x_k = x_k + K_k(z_k - Hx_x) \tag{4.45}$$

$$P_k = (I - K_kH)P_k \tag{4.46}$$

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

Hence, a priori and a posteriori estimate errors are defined as

$$e_k^- = x_k\tilde{x}_k^-, \tag{4.47}$$

$$e_k = x_k \tilde{x}_k. \tag{4.48}$$

The a priori estimate error covariance is then

$$P_k^- = E[e_k^- e_k^{-T}] \tag{4.49}$$

and the a posteriori estimate error covariance is

$$P_k = E[e_k e_k^T] \tag{4.50}$$

Following these, we can see the equation more deeply.

$$\tilde{x}_k = \tilde{x}_k + K(z_k - H\tilde{x}_k). \tag{4.51}$$

The difference $z_k - H\tilde{x}_k$ in 4.51 is called the the residual. The residual reflects the discrepancy between the predicted measurement $H\tilde{x}_k$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.
The matrix K in (4.44) is chosen to be the gain or blending factor that minimizes the posteriori error covariance. Taking the derivative of the trace of the result with respect to K, setting that result equal to zero, and then solving for K, we get

$$K_k = P_k H^T (H P_k H^T + R)^{-1} \tag{4.52}$$

$$K_k = \frac{P_k H^T}{H P_k H^T + R} \tag{4.53}$$

Looking at (4.53), it is seen that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}. \tag{4.54}$$

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

$$\lim_{P_k \to 0} K_k = 0. \tag{4.55}$$

Model design

In the selection of the matrices for the synchronization algorithm, we will consider different situations of the WSN.

The transition matrix H plays an important role in achieving the proper synchronization as it determines the weight that should be put in the previous value, as to how it influences the next wake-up time.

$$\tilde{x}_k = \tilde{x}_k + K(z_k - H\tilde{x}_k), \tag{4.56}$$

where x_k is the previous value of the node's offset, z_k is the measured phase offset of the nodes with one of its neighbors, and K is the Kalman gain.

The initial estimates are selected from the previous firing time of the node. Hence,the initial X and P are estimated from the previous values, 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. The initial value of the estimated value of the wake-up time is taken from the previous value, with a factor of p which represents the wake-up time of the neighboring nodes. Hence, p indicates how fast/slow is the next wake up time should be compared to it's previous value and/or neighbors. With the stable network where the nodes remain intact, the wake-up time of the node is expected to be the same despite the clock drift.

Hence, to tackle the dynamic nature of the WSN, the new node joining the network should be synchronized with the already established "status quo" of the network. In the filter design, the covariance matrices R and Q also have a significant role in the overall implementation of the algorithm. R represents how we value the measured values to affect the result of the outcome and Q sends a signal as to how we have to evaluate the estimated value.

The estimated values, which base on a stable background are more prone to be right after a series of firings. On the contrary, the measured values have large deviations in such away that more of the weight is going to the estimated values, tending to stabilize the network.

The update equations are applied in the series of measurement to end up in an optimal(next time) wake-up time of the node.

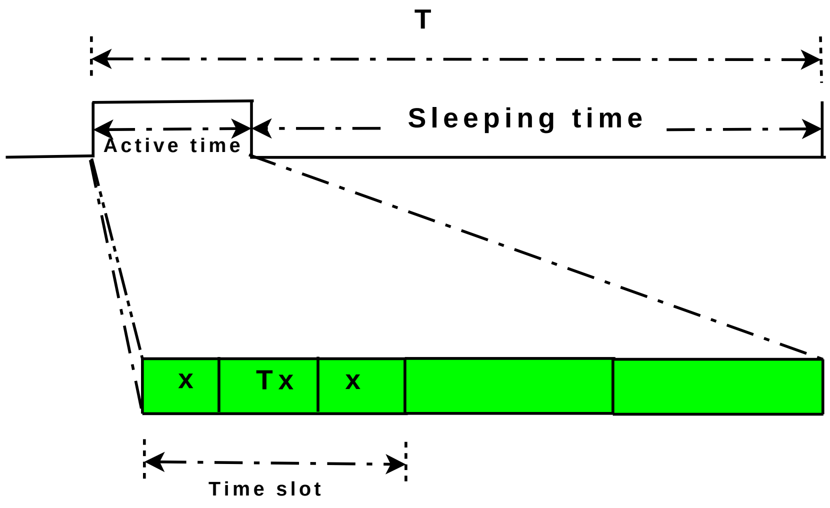


Figure 4.7: Guard time saving

Algorithm for KF

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

4.5 Reducing the guard time

As the precision of the algorithms decrease (performance increase), 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 4.7). Thus, the slot duration will be

$$T_{slot} = 2x + T_x, \tag{4.57}$$

where T_x is the transmit time of the node.

With N slots, the time duration of the active period of the node will be

$$NT_{slot} = N(2x + T_x). \tag{4.58}$$

The duty cycle is then

$$D = \frac{NT_{slot}}{T}, \tag{4.59}$$

where T is the period of a time frame.

Substituting (4.58) in (4.59) equation, the duty cycle becomes

$$D = \frac{N(2x + T_{slot}}{T}. \tag{4.60}$$

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 $2x - \epsilon$.

The new duty cycle becomes

$$D_n = \frac{(2(x - \epsilon) + T_x)N}{T}. \tag{4.61}$$

Arranging the equation results in

$$D_n = \frac{(2x + T_x)N}{T} - \frac{(2\epsilon)N}{T}. \tag{4.62}$$

Using the value of 4.59, it can be written as

$$D_n = D - \frac{(2\epsilon)N}{T}. \tag{4.63}$$

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

$$D - D_n = \frac{(2\epsilon)N}{T}. \tag{4.64}$$

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. The number of slots is determined by the MAC protocol.

Chapter 5

Results and Discussion

5.1 Simulation setup

In this section, the simulation setup is given which is used in the research to test the performance of the presented frame synchronization algorithms. The simulated wireless sensor network operates in the 2.4GHz ISM band at a data rate of 2Mbps. We use a Discrete Event Simulator (DES).

A DES will break down a simulation into discrete chunks. Every event will occur at some countable time moment and will be given in chronological order. The advantage of this distinction is two-fold. First, simulations will not be dependant on some real-time clock. Second, events can be isolated to perform certain measurements.

One such DES is called OMNeT++[22]. It provides a component architecture for models. Components are programmed in C++, then assembled into larger components and models using a high-level language. These components were merged into a joined simulation effort by academic institutes in Europe: MiXiM[1]. It is specialized on wireless sensor networks. Nodes are modeled as a set of separate components stacked upon each other. Octave[2] is used for

[1]MiXiM (MIXed sIMulator) is a simulation framework for wireless and mobile networks using the OMNeT++ simulation engine.It is a collaborative project between TU Berlin, TU Delft and Universitaet Paderborn.

[2]Octave is a free program for performing numerical computations which is mostly compatible with MATLAB. It is part of the GNU project.

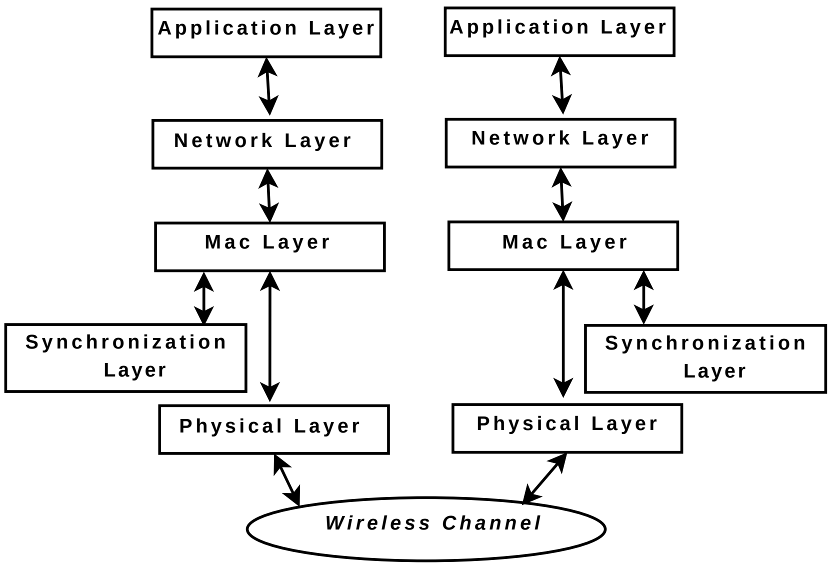


Figure 5.1: Layered Model

interpretation of the data from the network simulator.

As shown in Figure 5.1, the synchronization is computed in a separate layer. Each time the message is received, the time of arrival is recorded and used to calculate the next wake up time of the node. The communication over the channels is done in a unidisc way. All nodes in range of the radio actually receive the data, and those that are outside are excluded. The number of nodes is varied for different scenarios. The movement of the nodes is modeled from the static to an average speed of a slowly moving object. The simulation is conducted 1000 times to counter the effect of randomness introduced in the simulation.

5.1.1 Simulation Results

The nodes are deployed uniformly across the field. The neighborhood is limited to 10 nodes. The start up time of the nodes is random, Gaussian distributed variable, t_{io}. The synchronization error is the difference between the wake-up time of the nodes in the neighborhood.

Duration time frame	1s
Radio range	10 - 30m

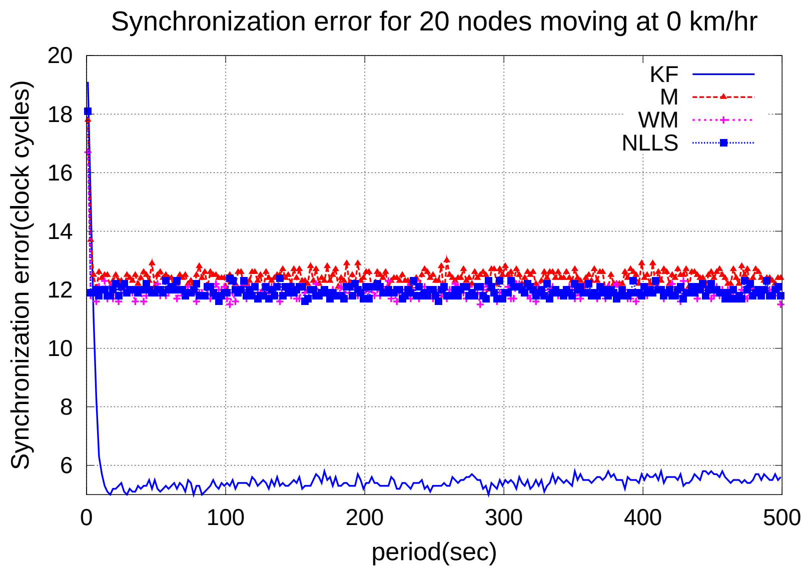


Figure 5.2: 16 nodes - Static

Case I

In the first set of simulation, the synchronization error is simulated for the nodes which are static, hence no effect of mobility. The number of nodes is taken to be 16 and 50 in the set of simulations.

Figure 5.2 shows the synchronization error for 16 nodes operating in a static environment. KF has the lowest synchronization error, at an avergage of $4clk$. NLCF and WM perform similarly with Median at average of $13clk$.

Figure 5.3 is the result of a simulation for 50 nodes. The synchronization error in general reduces as the number of nodes increases the number of neighbors, resulting in more data for the synchronization layer. Comparing the individual algorithms, KF has over $10clk$ performance improvement than the Median whereas NLCF and WM have a better performance ($1clk$) than the Median.

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 16. The simulations are conducted for different speeds, at $6km/hr$

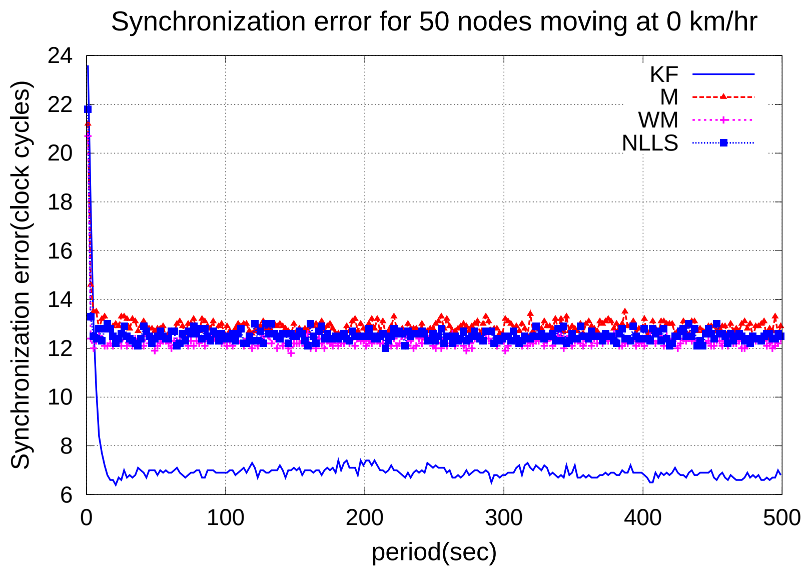


Figure 5.3: 50 nodes - Static

and $20km/hr$. Chosen speeds emulate the speed of a walking man and an average speed of a slowly moving vehicle.

Figure 5.4 shows 16 nodes with a constant speed of $6km/hr$ random mobility. As the results shows, WM and NLCF perform better the synchronization of the frame, $1clk$ each. KF outperforms all the best, with $10clk$ from the median algorithm.

The simulation is also conducted with a speed of $20km/hr$, having similar results(Figure 5.5). With the increase in the speed of the nodes, the precision of the algorithms improves. KF, with an average of of $10clk$ performs the best, whereas WM and NLCF perform well too compared with the median, $1clk$ and $1clk$ respectively. The relative comparison of the algorithms performance improvement with the Median is shown in Figure $A.5$. WM and NLCF perform, on average, $8 - 10\%$ better than Median. 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 due to nodes moving slowly in the surrounding (leave the network and join again after some time), resulting

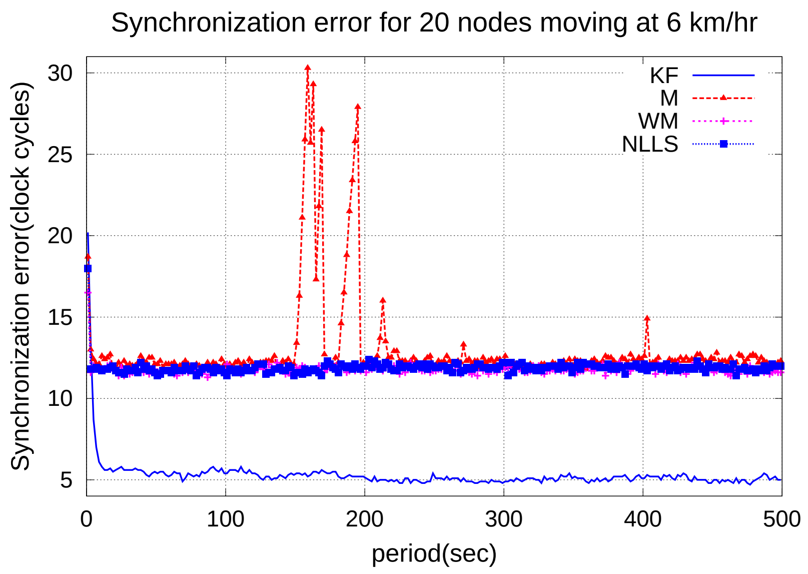


Figure 5.4: 16 nodes - Speed 6km/hr

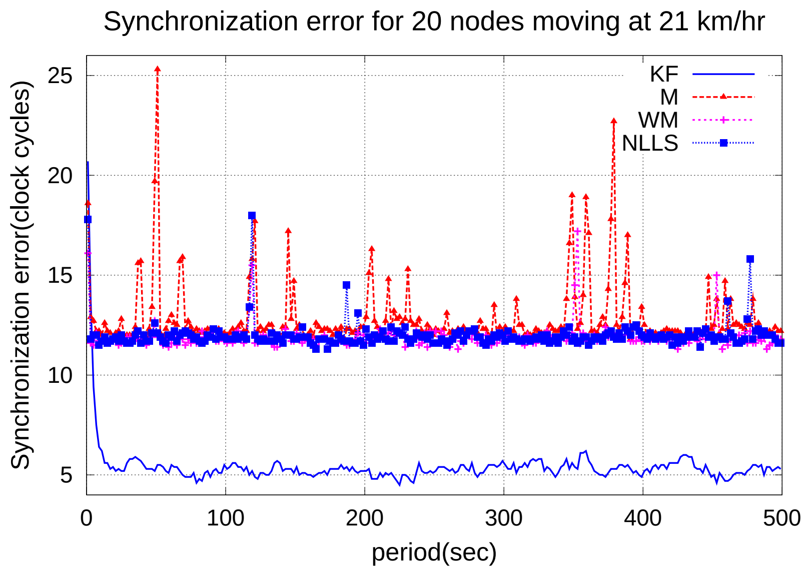


Figure 5.5: 50 nodes - Speed 20km/hr

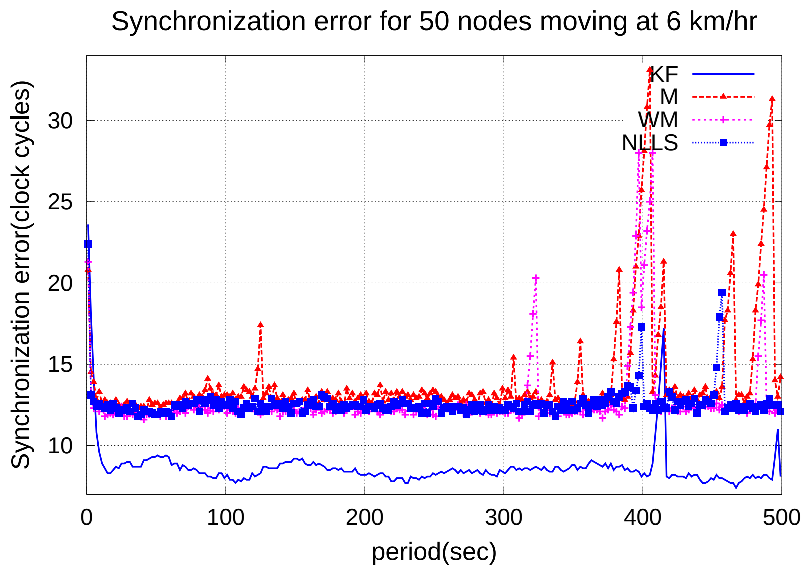


Figure 5.6: 50 nodes - Speed 6km/hr

in a larger drift with neighbors before getting back to the network.

With a speed of $20km/hr$, the simulation result is presented in Figure 5.7. With this speed, the performance is better due to the faster moving nodes, joining the networks at a faster rate. This helps in getting synchronized with the nodes in a faster time, without drifting away for a longer period of time(which is the case in the first set of nodes with speed of $6km/hr$). Again, 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 algorithms 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.

In general, Figure $A.6$ shows that KF performs the best against Median algorithm, 45%. WM and NLCF perform well against Median too, 10% each. But it has been shown that the median is prone to error in case of high dynamics in the network. In general, the Kalman filter has a better precision and convergence speed than the rest whereas WM is also doing in precision whereas NLCF has a better convergence speed.

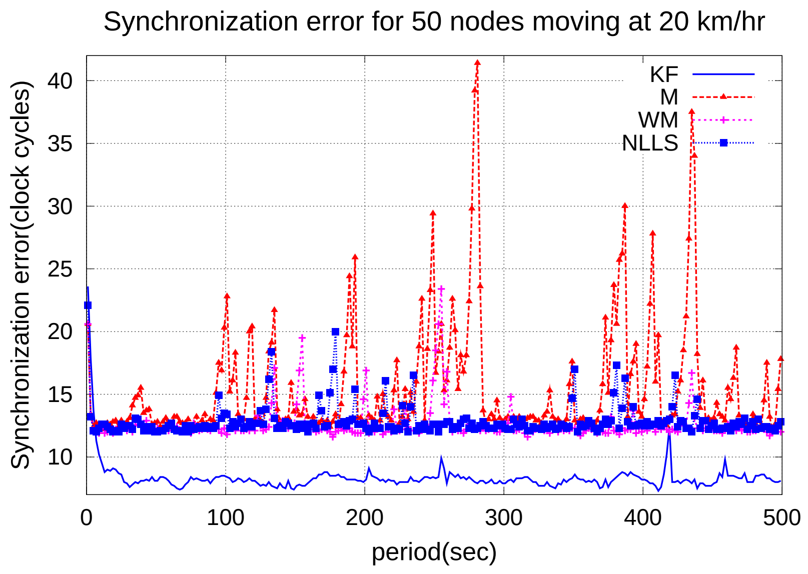


Figure 5.7: 50 nodes - Speed 20km/hr

5.2 Energy consumption

As the results in the previous section shows, KF algorithm performs well in all conditions, so do WM and NLCF. The downside in implementing these algorithms, despite the performance gain that all have against the Median is the energy consumption. As the algorithms are going to be implemented on the sensor nodes, the energy consumption is a priority in the study of 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.

Algorithm	Average Execution Time
M	65 μs
NLCF	82.5 μs
WM	90 μs
KF	150 μs

The table shows the average execution time of the algorithms in comparison to implemented on the MyriaNode. This doesn't show the exact energy consumption of the algorithms but it can give us an approximation on the relative comparison of the algorithms about the energy

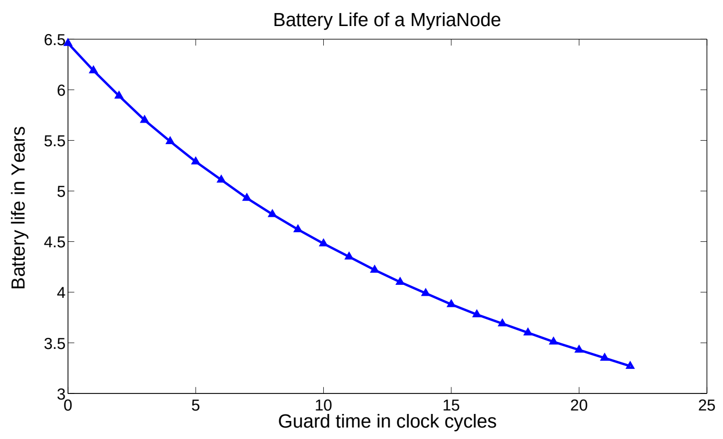


Figure 5.8: The energy consumption of Guard time per RX slot

consumption on the MyriaNode. As the table shows, the consumption of WM is 5% higher than the median algorithms. This is also true for NLCF and KF which consume 9% and 4 % higher than the median algorithm respectively.

In order to reduce the energy that is going to be spend 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 NLCF. The energy consumption of the guard time is shown in Figure 5.8. The length of the guard time has a 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. The relative energy reduction is shown in the chart below. Hence, by decreasing the guard time in proportion with the performance enhancement, we can save energy for the energy demand of the algorithm. This energy save is in a per slot basis. As the number of slots increases, the energy to be saved increases.

Chapter 6

Conclusion and Recommendation

6.1 Conclusion

A decentralized frame synchronization of a TDMA-based WSN is achieved using Weighted Measurements(WM), Non Linear Curve Fitting(NLCF) and Kalman Filter(KF) methods. A simulation is conducted with different scenarios, especially taking the effect of mobility. A comparison of the algorithms with the currently implemented Median algorithm is conducted and the results are presented.

In a static environment, the KF performs the best whereas the WM and NLCF have shown a similar performance as the Median. In terms of convergence time, all have shown a similar performance.

A 60% improvement in the performance of the synchronization can achieved on average using the Kalman Filter for dynamic WSN. A lower improvement, 10% and 8% is obtained using WM and NLCF methods for the synchronization of TDMA frames. Both WM and NLCF show a better tolerance against Median in a dynamic network. KF estimation of the next wake up time is the best with its iterative capability and adaptive nature.

Using these algorithms for a TDMA-based WSN synchronization(WM, NLCF and KF), the precision of the network increases. This in turn has a positive impact on the network 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 for an increased synchronization frequency.

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. As shown in the measurements and analysis results, communication is around 5 times costlier than computation, on average. A net gain of battery life can thus be achieved by reducing the guard time.

Median is shown to be still the best choice in a static environment with the current implementation,with less energy consumption and simplicity into consideration. But, for dynamic networks, the other algorithms perform better, in terms of performance. KF has the edge due to its adaptive nature.

6.2 Recommendation

There are some suggestions on the future work of the algorithms.

- Upon the completion of the ongoing project on developing a Software Defined Radio (SDR) for the inspection of the nodes' wake-up 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.

- As KF is shown to have the best performance, different software power minimization techniques can be applied to further reduce the power consumption of the algorithm implementation, making it more attractive for implementation.

- As it is shown in Section 3, the main sources of error for the clock inaccuracies are identified. Hence, solving the problem at its source 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.

- In depth study of the different biased-estimation techniques can be applied to have a better performance with reduced power.

Appendix A

Simulation Results

A.1 Median as a synchronization method

The performance of Median algorithm is dependent on the gain factor used in the offset calculation. Figure $A.1$ shows the performance of the Median algorithm for different gain factors.

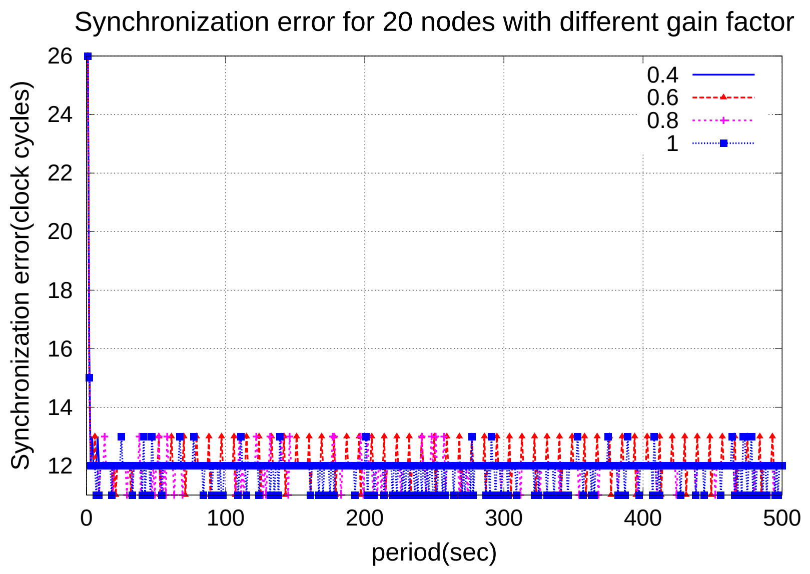


Figure A.1: Median algorithm with different gain factors

A.2 Synchronization frequency

Increasing the synchronization frequency increases performance but the precision decrease. Figure $A.2$ shows the performance of KF with varying the synchronization frequency T_{sync}.

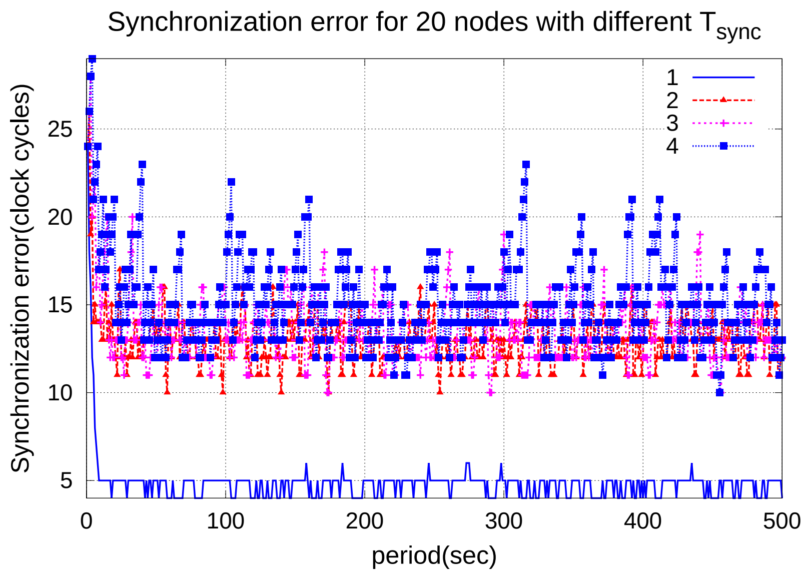


Figure A.2: Synchronization frequency on Kalman Filter

A.3 Synchronizztion error for nodes with different speeds

The synchronization error for nodes travelling with different speeds is given in Figure $A.3$ for 20 nodes. For 50 nodes, the simulation result is shown in Figure $A.4$. In both cases, KF performs the best out of all the algorithms.

A.4 Relative performance improvement of the algorithms

The performance of the algorithms is compared with that of the Median. For 16 nodes, see Figure $A.5$. The performance of the algorithms is compared with that of the Median. For 50 nodes, see Figure $A.6$.

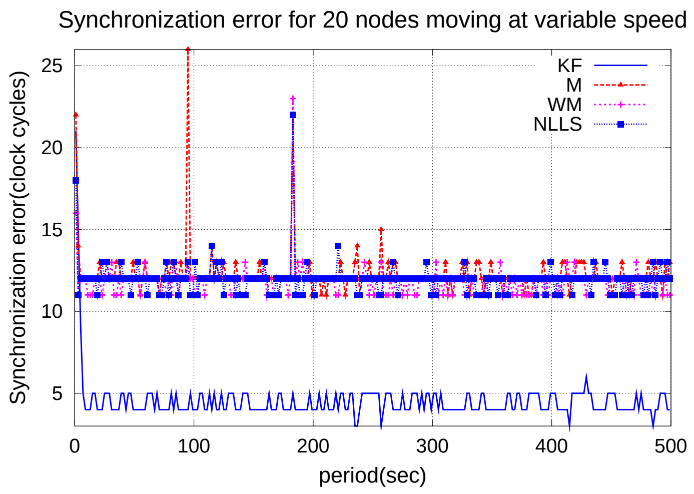


Figure A.3: Synchronization error for 20 nodes moving at different speed

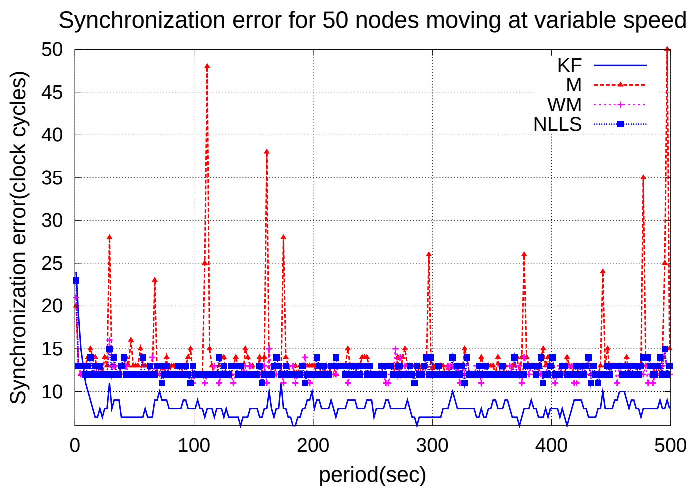


Figure A.4: Synchronization error for 50 nodes moving at different speed

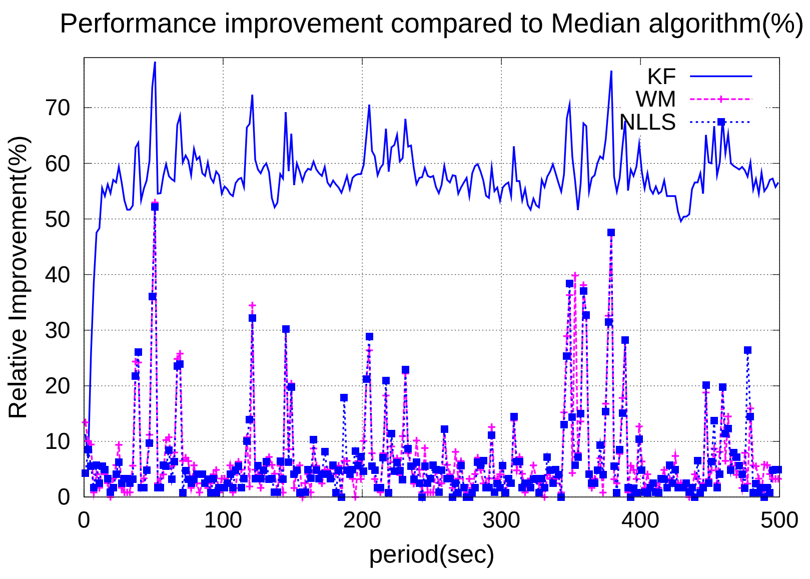


Figure A.5: Relative performance improvement of algorithms from Median - 16 nodes

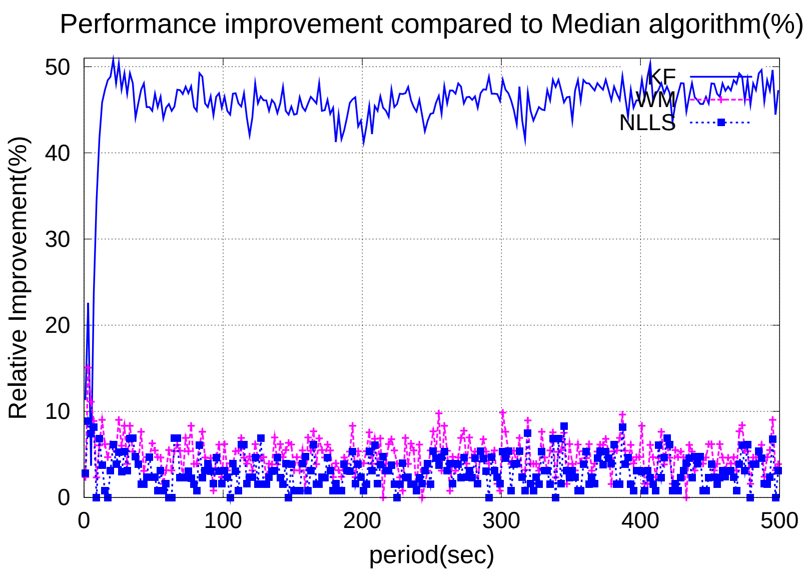


Figure A.6: Relative performance improvement of algorithms from Median - 50 nodes

Appendix B

MyriaNode

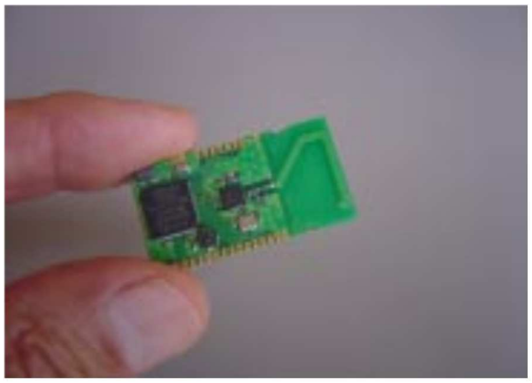


Figure B.1: MyriaNode v2.0

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

Description

This Wireless Sensor Node is the second generation product for the MyriaNed project. It integrates a Nordic 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.

Architecture

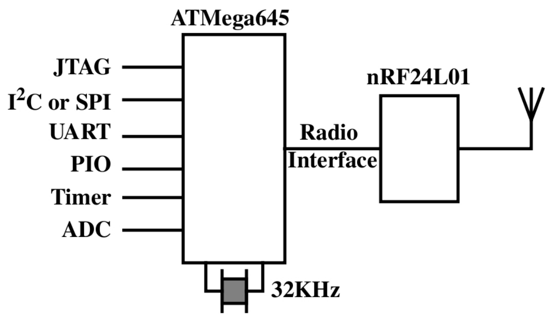


Figure B.2: MyriaNode Architecture

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 Interface.

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. The antenna area must be positioned in free air.

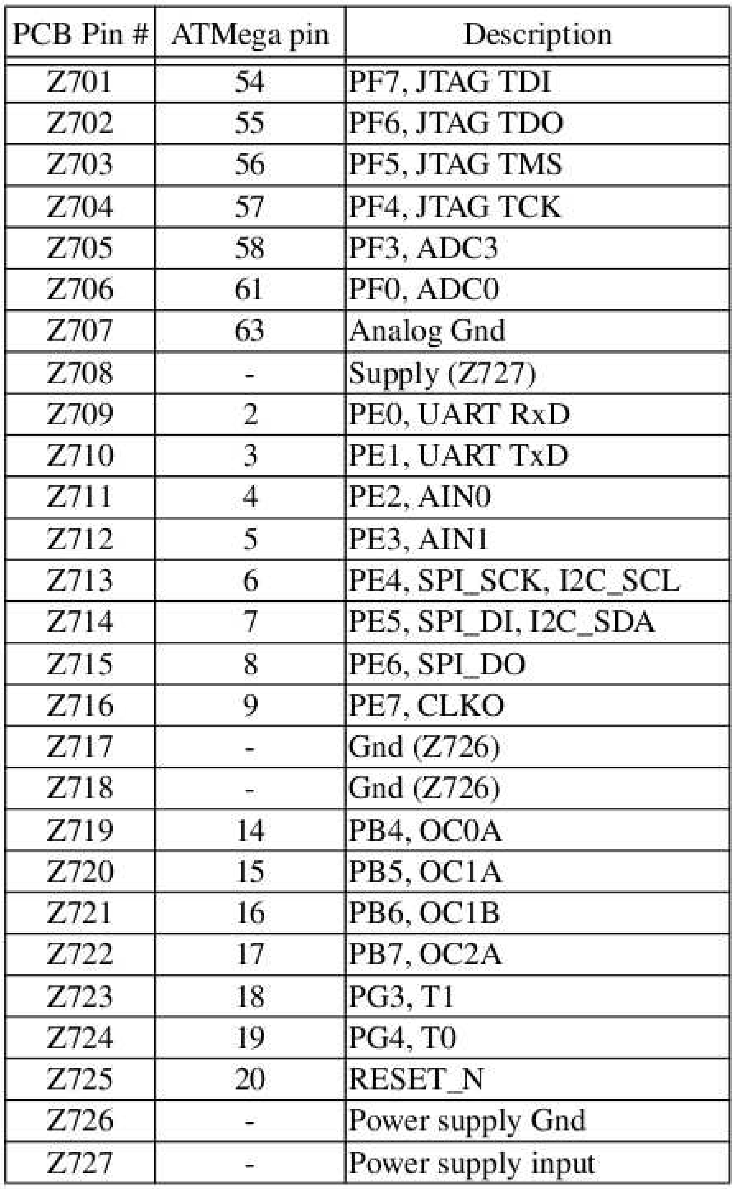
Energy Supply

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.

The energy consumption very much depend es 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.

The voltage level of a new Lithium Thionyl Chloride battery is 3.67V, while the absolute maximum operating voltage of the nRF24L01 is 3.60V. This requires a voltage reduction circuit. The most simple one is to connect the node via a diode to the battery, as is shown in Figure $B.6$.



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.

Battery

The ER14250STU from EMB is a Lithium Thionyl Chloride battery. It has a form factor of 1/2 AA size, and has a capacity of 1000mAh.

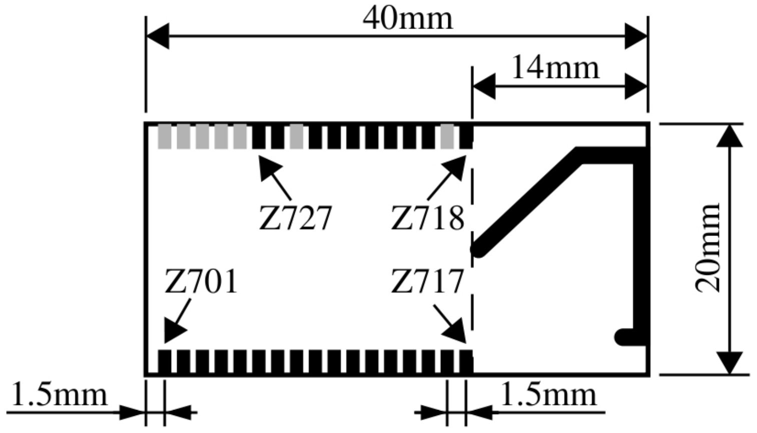


Figure B.5: MyriaNode dimensions

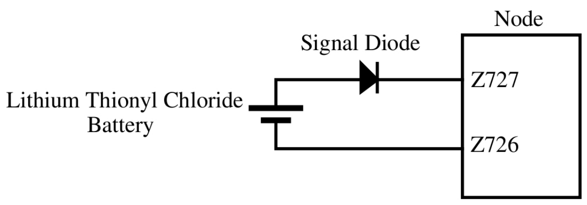


Figure B.6: Battery structure of a MyriaNode

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.

Software

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

Bibliography

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

[2] Q.Yang, and et al. A Decentralized Slot Synchronization Algorithm for TDMA-Based Ad Hoc Networks.

[3] A.Tyrrell, and et al. Firefly synchronization in ad hoc networks, in Proc. MiNEMA Workshop 2006, February, 2006.

[4] NTP Public Services Project http://support.ntp.org/bin/view/Main/WebHome.

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

[6] Q.Yang and J.Shi. An interference elimination method for decentralized slot synchronization in TDMA-based wireless ad hoc network.

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

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

[9] R. Fan and N. Lynch. Gradient clock synchronization. In Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing (PODC 04), pages 320327. ACM Press, 2004.

[10] 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.

[11] 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.

[12] 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.

[13] H.Karl and A.Willig. Protocols and Architectures for Wireless Sensor Networks.p3-6. Wiley. July 2006.

[14] C.Cordeiro and D.Agrawal. Ad hoc and Sensor Networks Theory and applications. p.429-441. World Scientific Publishing. 2006.

[15] F.Zhao and L.Guibas. Wireless Sensor Networks: an Information Processing approach. p.107=108. Elsevier. 2004.

[16] S.Raje. Time synchronization of in Network-centric sensor networks. Master research. University of Texas, Arlington. Texas. August 2005.

[17] K.Romer. Time synchronization in ad hoc networks. In: Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing , Long Beach, California, USA. October 2001.

[18] S.PalChaudhuri and et al. Adaptive Clock Synchronization in Sensor Networks, Information Processing in Sensor Networks, April 2004.

[19] G.Werner-Allen and et al. Firefly-inspired sensor network synchronicity with realistic radio effects. Proceedings of the 3rd international conference on Embedded networked sensor systems, San Diego, California. November 2005.

[20] *http : //www.golledge.co.uk/pdf/products/xtl_sm/cc7v.pdf*

[21] *http : //pdfserv.maxim − ic.com/en/an/AN*58*.pdf*.

[22] *http : //www.omnetpp.org*.

[23] *http : //www.cs.unc.edu/ welch/kalman/*.