Decentralized Frame Synchronization of a TDMA-based Wireless Sensor Network

Fasika Assegei
Radiocommunication Group
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: f.a.assegei@student.tue.nl

***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 WSNs, 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 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 Protocol (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 as a metric for selecting the algorithm.**

***Index Terms*—adhoc networks, frame synchronization, decentralized synchronization, wireless sensor networks, slot synchronization.**

I. INTRODUCTION

A. *Synchronization in Wireless Sensor Networks*

WSNs are distributed networks of sensors, dedicated to closely observing real-world phenomena. Various applications are realized using WSNs[8]. One of the design issues in WSN technology is clock synchronization, which is a critical piece of infrastructure in any distributed system. There are many reasons why a synchronized time is needed in a WSN, one of them being adjusting the slots in a TDMA-based communication.

In order to 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.

B. *Existing Work on WSN clock synchronization*

Several algorithms have been proposed and researched for time synchronization in WSNs. The *Refererence 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. [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 approach stated in [11] 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 using different estimation techniques. [7] 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. Correlation method is used in [12] inorder to decode the message and learn about the status of the sender node which is used for synchronization.

C. *Objective and overview of the research*

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 neighbor's 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. These algorithms have the following characteristics: higher precision, adaptive to environmental effects, energy-efficient and long-term synchronization.

The remainder of the paper is organized as follows: Section 2 presents a general overview of synchronization in WSN

and the need for a synchronized time. Section 3 discusses the synchronization error and formulate the problem. Section 4 presents the mathematical models of the algorithms for the synchronization of the network. In Section 5, simulation results are presented and analysis of the results in addition to the comparison of the methods with respect to energy consumption is discussed. Finally, Section 6 draws the conclusions from the research and suggests future work.

II. Problem Formulation

A. 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.
- **Dynamic Nature of the Network**: In a WSN, network dynamics results from various factors like mobility of nodes, node failures, environmental obstructions etc., which prevents simple static configurations.
- **Diverse Applications**: A variety of applications have totally different needs as far as synchronization is concerned. Some applications might need a global timescale while some others can work with a local timescale and some require a relative 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.

B. Sources of synchronization error

The different factors which give rise to errors in clocks of nodes or in synchronization algorithm can be identified as:
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.
- ***Stability***: This is oscillator tendency to stay at the same frequency over time.

Hardware and Environmental factors: The non-determinism in the message delivery latency is a major source of error. This can be categorized in four type of delays:

- ***Send Time***: The time spent at the Sender to build the message.
- ***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.
- **Frequency error**: Frequency error, in contrast, measures the difference in the clockrates.
- **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.

C. Clock Drift

From the definition of frequency:

$$f = d\phi/dt, \tag{1}$$

and integrating both sides over time,

$$\phi = \int f(t)dt, \tag{2}$$

where f is frequency, ϕ is phase, and t is time.
Thus, the clock time is described as

$$C(t) = \frac{1}{f_o}\int_{t_o}^{t} f(\tau)d\tau + C(t_o), \tag{3}$$

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

$$1 - \rho \leq \frac{dC(t)}{dt} \leq 1 + \rho, \tag{4}$$

where ρ represents the maximum clock drift.
Note that different clocks have different maximum clock drift values ρ.
The frequency of the clock is dependent on different factors and can be given as

$$f_i(t) = f_o + \Delta f + a(t - t_o) + \Delta f_e(t - t_o) + f_r(t) \tag{5}$$

where
t_o = the start time of the clock,
a = aging factor,
f_o = nominal frequency,
Δf = calibration error,
$f_r(t)$ = frequency instability (noise) term,
Δf_e = frequency error which occurs due to outside factors like temperature, voltage instability.
From (3) and (5), we get

$$C_i(t) - C_i(t_o) = \frac{1}{f_o}\int_{t_o}^{t} f_i(\tau)d\tau, \tag{6}$$

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

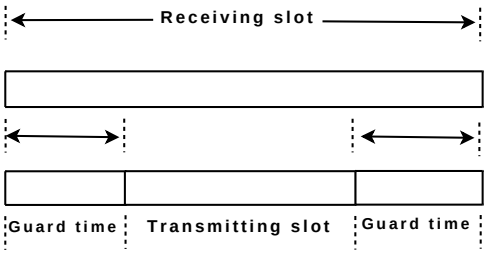


Fig. 1. Guard time of the nodes

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

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.

III. SYNCHRONICITY PROTOCOL

A. Synchronization Frequency

Precise synchronization: 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 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 1. As the guard time increases, the probability that the message is received in the time slot increases although the energy consumption increases. 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

$$t_{slot} = kT, \tag{7}$$

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

$$t_{diff} = 2\frac{T_{sync}}{T}\rho, \tag{8}$$

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},

$$t_{guard} = 2t_{diff} = 4\frac{T_{sync}}{T}\rho. \tag{9}$$

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

$$t_{slot} \geq t_{guard} + T_{tx}, \tag{10}$$

where T_{tx} is the time required 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

$$tick_{rx} = T_{tx} + \frac{t_{guard}}{2} \tag{11}$$

Hence, each time when a node receives a message, it has to be received exactly at the clock tick given by (11). Whenever this number is not equal with the desired one, a phase error is observed.
Two nodes have a good communication link when they synchronize their clocks at least once every T that gives the following relation,

$$\frac{T_{sync}}{T} \geq 1. \tag{12}$$

This has a negative impact on the battery life of the nodes. To obtain the value of T_{sync} for a particular system with a given duty cycle, we have to solve the next problems

$$t_{guard} \geq 4\rho\frac{T_{sync}}{T} \tag{13}$$

$$t_{slot} \geq t_{guard} + T_{tx} \tag{14}$$

$$\frac{T_{sync}}{T} \geq 1. \tag{15}$$

In general, 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.

B. 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 determination of the time that the message is sent. The difference between the wake up times of node i and node $j2$ is

$$\Delta t_{ij}^{(n)} = t_i^{(n)} - t_j^{(n)}, \tag{16}$$

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

$$t_i^{(n)} = \sum_n T_i^{(n)} + t_{io}, \tag{17}$$

where $T_i^{(n)}$ is the period of the crystal clock at the n^{th} period which changes with time according to (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 (5).
The difference in the wake-up time of the nodes is given by

$$\Delta t_{ij} = \sum_n T_i^{(n)} + t_{io} - (\sum_n T_j^{(n)} + t_{jo}) \tag{18}$$

$$= (\sum_n T_i^{(n)} - \sum_n T_j^{(n)}) + (t_{io} - t_{jo}). \tag{19}$$

This phase error can be adjusted 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, and
- The complexity of the implementation.

Hence, 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,

$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - \xi_i^{(n)}, \tag{20}$$

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

$$\xi_i = f(\Delta t_{ij}). \tag{21}$$

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 section of the report.

1) ***Median algorithm and the setback****:* The Median algorithm 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

$$\Delta t_{ij}^{(n)} = t_i^{(n)} - t_j^{(n)}, \tag{22}$$

where t_i is the wake-up time of node i and t_j is the wake-up time of node j at the n^{th} period.
4) Receivers compute the median of the offsets

$$\xi_i^{(n)} = m(\Delta t_{ij}^{(n)}), \forall j \tag{23}$$

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

$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - k\xi_i^{(n)}, \tag{24}$$

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.
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 2, a distribution of 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

$$\xi_9 = \frac{\Delta t_{910} + \Delta t_{39}}{2}. \tag{25}$$

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 after its adjustment with the Node 10. It will take more time to synchronize with the network again. Figure 3 shows the state of the network after the synchronization using the Median.
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.

2) ***Weighted Measurements - Interference-phobic approach****:* One form of approach to tackle the dynamic behavior of a WSN, due to channel conditions as well as collisions, is a Weighted Measurement (WM) approach. Using this approach, different weights are 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".
The offset is then

$$\xi_i^{(n)} = \sum w_{ij}^{(n)} \Delta t_{ij}^{(n)}, \tag{26}$$

where $\sum w_{ij}^{(n)} = 1$.
The weighted adjustment is used to modify the next wake-up time of the node,

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

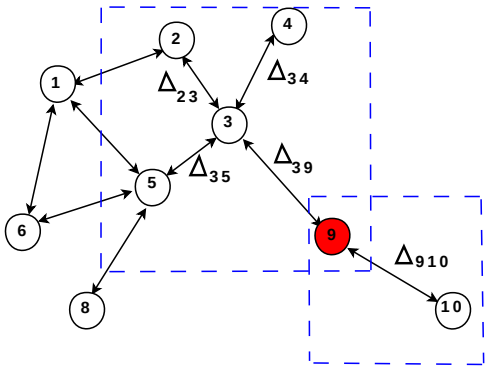


Fig. 2. A WSN scenario

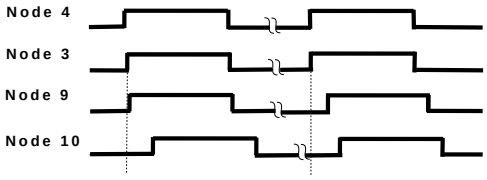


Fig. 3. Using median for phase error correction

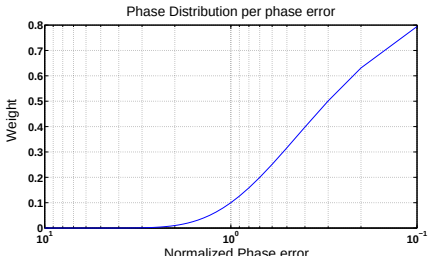


Fig. 4. Weight factors for the phase error distribution

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 the 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:

$$\delta_{ij} = ae^{-b\Delta t_{ij}}. \tag{27}$$

The parameters are selected using the initial conditions, $\delta_{ij} = 0.1$ for Δt_{ij} = t_{guard} and $\delta_{ij} = 1$ for Δt_{ij} = 0.
As shown in Figure 4, 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 ,

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

where m is the mean function. Hence, the weight moves towards the large phase errors if the node is a new-comer which wants to join the "already established" network.
As shown earlier, 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.

*3) **Non Linear Least Squares - Interference elimination**:*
Non-Linear Least Squares

Non Linear Least Squares (NLLS) curve fitting 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,

$$f(x_i, \beta) = \beta_1 + \beta_2 log(x_i), \tag{28}$$

where β_1 and β_2 are the parameters to be estimated.
As the phase errors are of stochastic nature, the curve fit we selected is 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, the synchronization error. See Figure 5.
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,

$$\beta = (\beta_1, \beta_2). \tag{29}$$

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

$$S = \sum_{i=1}^{m} r_i^2, \tag{30}$$

is minimized, where the residuals r_i are given by

$$r_i = y_i - f(x_i, \beta) \tag{31}$$

for i=1,2,. . ., n.
The minimum value of S occurs when the gradient is zero.

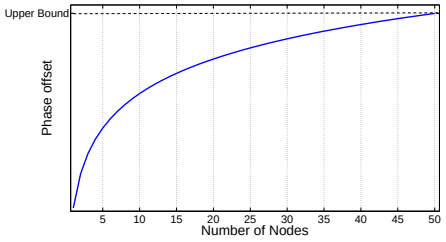


Fig. 5. Curve fitting using logarithmic function

Hence, there are gradient equations to be solved:

$$\frac{\partial S}{\partial \beta_j} = 2\sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0 \ (j = 1, 2). \tag{32}$$

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,

$$\beta_j^{k+1} = \beta_j^k + \Delta\beta_j. \tag{33}$$

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.

$$f(x_i, \beta) \approx f(x_i, \beta^k) + \sum_j \frac{\partial f(x_i, \beta^k)}{\partial \beta_j} \left(\beta_j^k - \beta_j\right) \tag{34}$$

$$f(x_i, \beta) = f(x_i, \beta^k) + \sum_j J_{ij} \Delta\beta_j. \tag{35}$$

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,

$$\frac{\partial r_i}{\partial \beta_j} = -J_{ij} \tag{36}$$

and the residuals are given by

$$r_i = \Delta y_i - \sum_{j=1}^{j=n} J_{ij} \Delta\beta_j, \tag{37}$$

where

$$\Delta y_i = y_i - f(x_i, \beta^k). \tag{38}$$

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

$$\left(J^T J\right) \Delta\beta = J^T \Delta y \tag{39}$$

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),

$$S = \sum_{i=1}^{i=m} W_{ii} r_i^2. \tag{40}$$

The normal equations are then

$$\left(J^T W J\right) \Delta\beta = J^T W \Delta y. \tag{41}$$

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.

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 for the function

$$f(x_i, \beta) = \beta_1 + \beta_2 log(x_i), \tag{42}$$

The initial values of the parameters β_1 and β_2 is estimated taking into account the state of the network to be.

As each measurement arrives from the neighbors, an iteration is made in such a way that the measurement error from a predetermined 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.

*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

$$x_k = Hx_{k-1} + Bu_k + w_{k-1}, \tag{43}$$

with a measurement z that is

$$z_k = Hx_k + v_k. \tag{44}$$

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

$$p(w) \approx N(0, Q), \tag{45}$$

$$p(v) \approx N(0, R). \tag{46}$$

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

$$x_k = Hx_{k-1} + Bu_k \tag{47}$$

$$P_k = HP_{k-1}H^T + Q \tag{48}$$

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.

$$K_k = P_k H^T (HP_k H^T + R)^{-1} \tag{49}$$

$$x_k = x_k + K_k(z_k - Hx_x) \tag{50}$$

$$P_k = (I - K_k H)P_k \tag{51}$$

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

$$e_k^- = x_k \tilde{x}_k^-, \tag{52}$$

$$e_k = x_k \tilde{x}_k. \tag{53}$$

The a priori estimate error covariance is then

$$P_k^- = E[e_k^- e_k^{-T}] \tag{54}$$

and the a posteriori estimate error covariance is

$$P_k = E[e_k e_k^T] \tag{55}$$

Following these, we can see the equation more deeply.

$$\tilde{x}_k = \tilde{x}_k + K(z_k - H\tilde{x}_k). \tag{56}$$

The difference $z_k - H\tilde{x}_k$ in (56) is called 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 (49) 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

$$K_k = P_k H^T (H P_k H^T + R)^{-1} \tag{57}$$

$$K_k = \frac{P_k H^T}{H P_k H^T + R} \tag{58}$$

Looking at (58), it is seen that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}. \tag{59}$$

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

$$\lim_{P_k \to 0} K_k = 0. \tag{60}$$

Filter 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. H 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.

$$\tilde{x}_k = \tilde{x}_k + K(z_k - H\tilde{x}_k), \tag{61}$$

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. 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 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.

C. *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 6). Thus, the slot duration will be

$$T_{slot} = 2x + T_x, \tag{62}$$

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

$$NT_{slot} = N(2x + T_x). \tag{63}$$

The duty cycle is then

$$D = \frac{NT_{slot}}{T}, \tag{64}$$

where T is the period of a time frame.
Substituting (63) in (64) equation, the duty cycle becomes

$$D = \frac{N(2x + T_{slot})}{T}. \tag{65}$$

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

$$D_n = \frac{(2(x - \epsilon) + T_x)N}{T}. \tag{66}$$

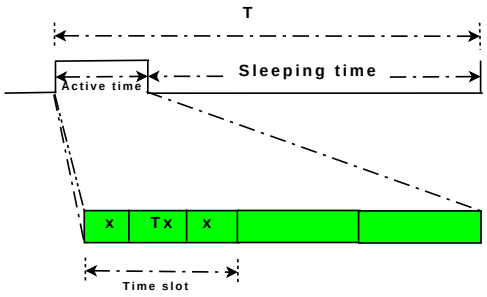


Fig. 6. Guard time saving

Arranging the equation results in

$$D_n = \frac{(2x + T_x)N}{T} - \frac{(2\epsilon)N}{T}. \quad (67)$$

Using the value of 64, it can be written as

$$D_n = D - \frac{(2\epsilon)N}{T}. \quad (68)$$

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

$$D - D_n = \frac{(2\epsilon)N}{T}. \quad (69)$$

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.

IV. Results and Discussion

A. Simulation setup

In the simulation, a Discrete Event Simulator (DES) is used. 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 MiXiM[1]. It provides a component architecture for models. Components are programmed in C++, then assembled into larger components and models using a high-level language. Octave[2] is used for interpretation of the data from the network simulator.
The simulation is conducted 1000 times to counter the effect of randomness introduced in the simulation.

B. 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.

*1) **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 $7a$ shows the synchronization error for 16 nodes operating in a static environment. KF has the lowest synchronization error, at an avergage of $4clk$. NLLS and WM perform similarly with Median at average of $13clk$.
Figure $7b$ 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 NLLS and WM have a better performance ($1clk$) than the Median.

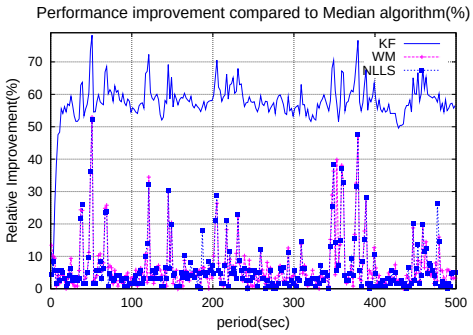


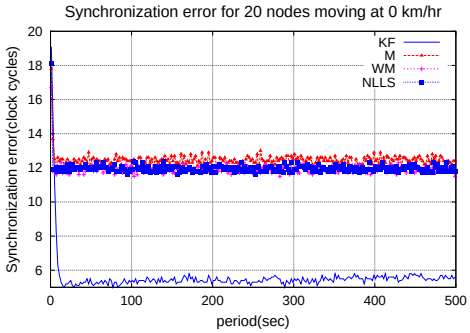
Fig. 9. Relative error of the algorithms w.r.t. Median

*2) **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 $6km/hr$ and $20km/hr$. Chosen speeds emulate the speed of a walking man and an average speed of a slowly moving vehicle.
Figure $8a$ shows 16 nodes with a constant speed of $6km/hr$ random mobility. As the results shows, WM and NLLS 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 $8b$). 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 NLLS 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 9. WM and NLLS perform, on average, $8 - 10\%$ better than Median. The best performer, KF, has on average 60% better performance than the Median one.

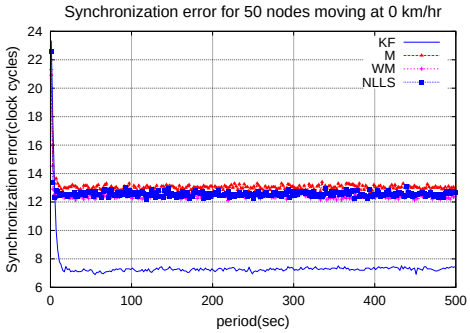
*3) **Case III**:* In this set of simulations, the number of the nodes is increased, to 50. With slowly moving nodes ($6km/hr$), the results are shown in Figure $10a$. Large disruptions occur due to nodes moving slowly in the surrounding (leave the network and join again after some time), resulting in a larger drift with neighbors before getting back to the network.
With a speed of $20km/hr$, the simulation result is presented in Figure $10b$. 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

[1]MiXiM (Mixed Simulator) is a simulation framework for wireless and mobile networks using the OMNeT++ [16] 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.

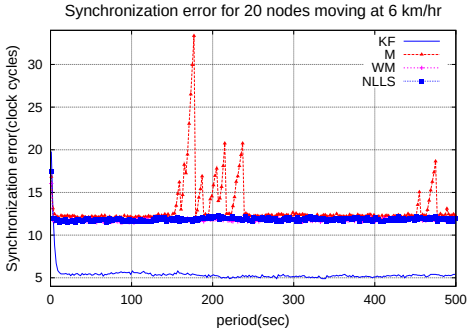


(a) Number of Nodes - 16

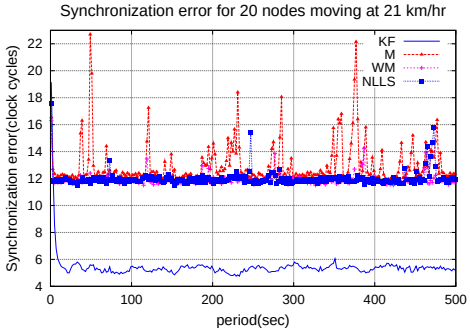


(b) Number of Nodes - 50

Fig. 7. Simulation results for Static Nodes

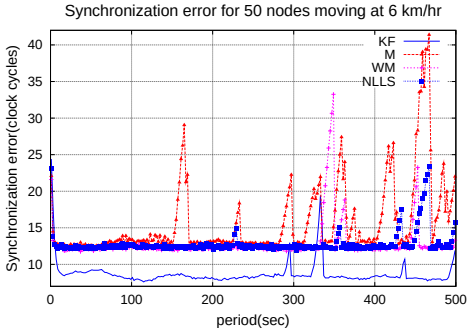


(a) Speed - 6km/hr

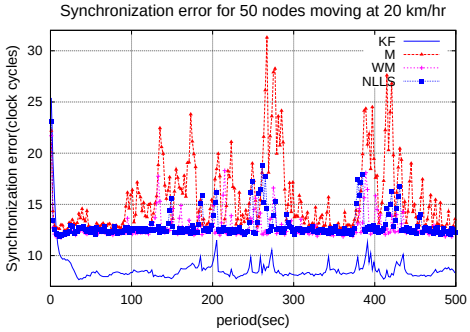


(b) Speed - 20km/hr

Fig. 8. Simulation results for 16 Nodes



(a) Speed - 6km/hr



(b) Speed - 20km/hr

Fig. 10. Simulation results for 50 Nodes

it more unstable whereas the other algorithms adapt to the changes faster.

In general, Figure 11 shows that KF performs the best against Median algorithm, 45%. WM and NLLS 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 NLLS has a better convergence speed.

C. *Measurements concerning the energy consumption*

As the results in the previous section shows, KF algorithm performs well in all conditions, so do WM and NLLS.

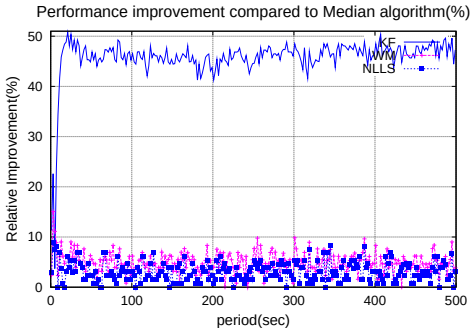


Fig. 11. Relative error of the algorithms w.r.t. Median

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

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 NLLS. The energy consumption of the guard time is shown in Figure 12.

With the performance gain obtained, the guard time of the slot can be decreased, hereby decreasing the energy consumption of the node in general. Hence, by decreasing the guard time in proportion with the performance enhancement, we can save energy for the energy demand of the algorithm.

V. Conclusion and Recommendation

A decentralized frame synchronization of a TDMA-based WSN is achieved using Weighted Measurements(WM), Non Linear Curve Fitting(NLLS) 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.

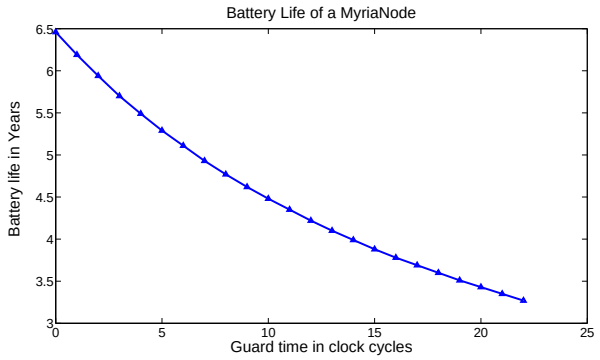


Fig. 12. The energy consumption of Guard time per RX slot

In a static environment, the KF performs the best whereas the WM and NLLS 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 NLLS methods for the synchronization of TDMA frames. Both WM and NLLS 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.

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.

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 suggestions for future work, a Software Defined Radio (SDR) for the inspection of the nodes' wake-up times, evaluation and enhancements on the algorithms, being implemented on the MyriaNodes, can be used. In addition, different software power minimization techniques can be applied to further reduce the power consumption of the algorithm implementation, making it more energy-efficient for implementation. As part of reducing the phase error between the clocks, compensating the frequency error using the available resource on the sensor nodes can be used. This can be achieved using the resources like the temperature sensor in the microcontroller of the MyriaNode.

VI. Acknowledgment

This research is part of the MyriaNed[3] Project and is conducted at Chess[4] B.V. The author would like to thank Frits van der Wateren from Chess Innovation Team for his continuous support, suggestions and advice during the project. Many thanks to dr.ir. Peter Smulders, prof.dr.ir. Erik Fledderus and prof.dr.ir. Peter Baltus from TU/e for their guidance.

References

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

[3]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.

[4]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.

[4] E.Anceaume and I.Puaut. A taxonomy of clock synchronization algorithms. IRISA Research Report No. PI1103, IRISA, 1997.
[5] Q.Yang and J.Shi. An interference elimination method for decentralized slot synchronization in TDMA-based wireless ad hoc network.
[6] P.Anemaet. Determining G-MAC potential with {S,L,SCP}-MAC. Masters thesis. Technische Universteit Delft. Delft. August 2008.
[7] R.John. Introduction to Quartz Frequency Standards. Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate. October 1992.
[8] H.Karl and A.Willig. Protocols and Architectures for Wireless Sensor Networks.p3-6. Wiley. July 2006.
[9] C.Cordeiro and D.Agrawal. Ad hoc and Sensor Networks Theory and applications. p.429-441. World Scientific Publishing. 2006.
[10] F.Zhao and L.Guibas. Wireless Sensor Networks: an Information Processing approach. p.107-108. Elsevier. 2004.
[11] S.Raje. Time synchronization of in Network-centric sensor networks. Master research. University of Texas, Arlington. Texas. August 2005.
[12] A.Ebner, H.Rohling, M.Lott, and R.Halfmann. Decentralized Slot Synchronization in Highly Dynamic Ad Hoc Networks. Proc. WPMC '02, Honolulu, Hawaii, Oct. 2002.
[13] S.PalChaudhuri and et al. Adaptive Clock Synchronization in Sensor Networks, Information Processing in Sensor Networks, April 2004.
[14] $http://www.golledge.co.uk/pdf/products/xtl\_sm/cc7v.pdf$
[15] $http://pdfserv.maxim-ic.com/en/an/AN58.pdf$.
[16] $http://www.omnetpp.org$.
[17] $http://www.cs.unc.edu/\ welch/kalman/$.