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*—Synchronization is a one of the main issues in the design of *Wireless Sensor Networks* (WSNs). Most applications of WSNs make extensive use of synchronization mechanisms. The unique requirements of WSNs in terms of precision, lifetime, energy and scope of the synchronization achieved, make the synchronization methods developed for centralized systems unsuitable for WSNs. This motivates the research of synchronization methods which are aligned to the specific properties of WSN. Interfering nodes, due to mobility and vibration, lead to diverging synchronization algorithms. Median[14] algorithm is implemented currently on the wireless sensor nodes in MyriaNed[1] project. In this research, the flaw in the Median algorithm is explored and three algorithms are proposed to achieve a decentralized, stable, and energy-efficient synchronization of a TDMA-based WSN, especially taking the effect of mobility. The algorithms achieve synchronization by using the phase errors of a node's clock with those of the neighboring nodes, without exchanging the information about the clock time of the sender. So, the methods avoid time stamping which reduces the message overload. Simulation results are presented and discussed. The research is concluded with the comparison of the proposed algorithms in terms of energy consumption and performance with the Median. 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 synchronization, decentralized synchronization, wireless sensor networks synchronization.**

I. INTRODUCTION

A. Synchronization in wireless sensor networks

WSNs are distributed networks of low cost sensors, dedicated to closely observing real-world phenomena. Various applications are realized using WSNs[5]. One of the design issues in WSN technology is 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. Due to physical factors, the frequency of the oscillator in the node has drift. When no provisions are taken, it causes the nodes to run out of synchronization.

[1]MyriaNed is a DevLab project to create a large functional wireless sensor network for research on protocols, power management, programming models, and security.

B. Existing work on WSN frame synchronization

Several algorithms have been proposed and researched for frame synchronization in WSNs. The *Reference Broadcast Synchronization* (RBS) stated in [10] 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. But a central node is needed for broadcasting reference signals. Another algorithm for a decentralized slot synchronization is presented in [12]. This method uses the topology of the nodes as a means to weigh the phase error between the sender and the receiver. In this case, higher message overhead is observed in sending the degree of connection.

Another approach stated in [16] establishes a table to correspond the clock of the sender with the receiver so that a good estimation of the neighbors' clocks is achieved using different estimation techniques. But message overhead is high as the messages are times tamped. This approach is most suitable for static networks as the neighbors are constantly changing. A different approach to a weight-based synchronization for interference elimination of a TDMA-based ad-hoc networks is presented in [15]. The algorithm achieves synchronization by eliminating the impact of interfering nodes. But this method directly eliminates nodes which are newly joining or interfering nodes. Correlation method is used in [1] in order to decode the message and learn about the status of the sender node which is used for synchronization. The downside of this approach is that it has a higher cost on the message overload.

C. Objective and overview of the research

The primary objective of this research is to develop an algorithm to achieve a long-term decentralized frame synchronization of a WSN in an energy-efficient method. Three algorithms are proposed and compared in terms of performance and energy consumption. As interference due to mobility and newly joining nodes is a frequent realty, the algorithms address the dynamic nature of the network to achieve synchronization during tough times. These methods use the phase errors between the receiver and its neighbors, without estimating the neighbor's clock as a way of achieving synchronization.

The remainder of the paper is organized as follows: Section II discusses the sources of synchronization error and presents

the set back of the Median algorithm. Section III presents the mathematical models of the proposed algorithms for the synchronization of a WSN. In Section IV, simulation results are presented. Analysis of the results in addition to the comparison of the methods with respect to performance and energy consumption is discussed. Finally, Section V draws the conclusions from the research and suggests future work.

II. Problem Formulation

A. Wireless sensor networks: Why different ?

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

- **Energy limitation**: Due to the nodes' 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 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 others can work with a relative timescale.
- **Cost of the nodes**: Sensor nodes must be cheap cost wise.

B. Sources of synchronization error

The different factors which give rise to errors between two nodes can be identified[16] as:

Oscillator characteristics: The following two oscillator characteristics are prone to error.

- ***Accuracy***: is a measure of the difference between oscillators expected (ideal) frequency and actual frequency.
- ***Stability***: is oscillator's tendency to stay at the same frequency over time.

Hardware and environmental factors: The non-determinism in the message delivery latency 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 receiver.

Some definitions which are used in the paper are presented below.

- **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 measures the difference in the clock rates.
- **Clock drift**: It is not just that the clocks are running at different rates, but the frequency of each clock does not stay constant over a period of time.
- **Clock cycle (*clk*)**: The time between two adjacent pulses of the oscillator.
- **Wakeup time**:Time that the node starts listening.

C. Clock time

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 but can be usually assumed that it doesn't exceed a maximum value ρ. 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[6] as

$$f_i(t) = f_o + \Delta f + a(t - t_o) + \Delta f_e(t) + \Delta f_n(t) \tag{4}$$

where t_o = the start time of the clock, a = aging factor, f_o = nominal frequency, Δf = calibration error, $\Delta f_n(t)$ = frequency instability (noise) term, $\Delta f_e(t)$ = frequency error which occurs due to outside factors like temperature and voltage instability.

From (3) and (4),

$$C_i(t) - C_i(t_o) = \frac{1}{f_o}\int_{t_o}^{t} f_i(\tau)d\tau, \tag{5}$$

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

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

A 32 kHz crystal clock [8], [7] is used in the MyriaNode[2]. Some notes to be considered in the model are:

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

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

- Clock parameters are normally distributed. The variances of the constants (initial time, initial frequency error, aging rate)[8], [7] are all inputs to the model.
- The spread in clock time grows almost linearly as a function of time, due to the dominance of the linear term in the clock drift equation.

D. *Median algorithm and the setback*

The Median algorithm is currently implemented in the MyriaNode. With the simplicity concerning the computational need, it is the best fit for WSN requirements discussed in the previous section. The algorithm is described as follows:

1) Nodes broadcast packets.
2) Each receiver records the time that the packet is received according to the local clock.
3) Each receiver i computes its phase error to any other node j in the neighborhood as
$$\Delta t_{ij}^{(n)} = t_i^{(n)} - t_j^{(n)}, \tag{6}$$
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 offset, ξ_i, which is the median of the phase errors,
$$\xi_i^{(n)} = median(\Delta t_{ij}^{(n)}), \forall j \tag{7}$$
5) Receivers adjust their wake-up time by the computed offset value,
$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - G\xi_i^{(n)}, \tag{8}$$
where G is the gain factor.

By multiplying the median with a gain factor, $G\xi_i^{(n)}$, 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.

A typical scenario is presented where the Median algorithm takes a longer time to achieve synchronization. In Figure 1, 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.

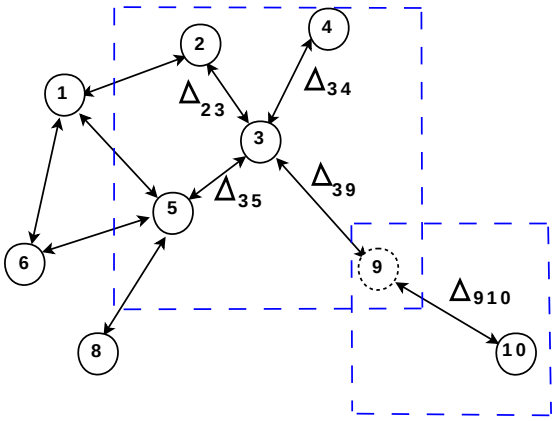


Fig. 1. A WSN scenario

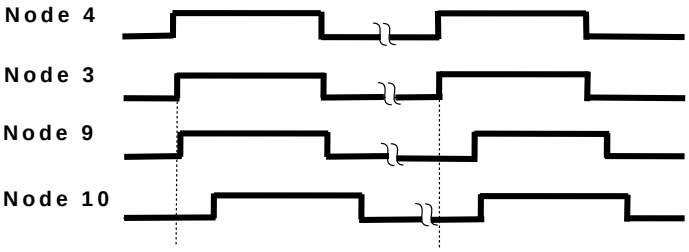


Fig. 2. Using median for phase error correction

Thus, upon the application of the median algorithm, the node tends to adjust its wake-up time with the median of the phase errors from its neighbors, Node 3 and Node 10. Adjusting the wake-up time, the offset is

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

Being in synchronization with the other nodes, Node 9 will drift away from the network after its adjustment with Node 10. Therefore, it will take Node 9 more time to synchronize with the network again. Figure 2 shows the state of the network after the synchronization using the Median.

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

III. SYNCHRONICITY PROTOCOL

A. *Synchronization frequency*

Decreasing the timing of the synchronization and executing the synchronization algorithms periodically reduces the energy consumption of the WSN greatly. Thus, a synchronization period, T_{sync}, is defined as the period in which the network can stay synchronized without the application of the synchronization algorithm.

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

$$t_{diff} = 2\frac{T_{sync}}{T}\rho, \tag{10}$$

where T is the period of the frame and the factor of 2 reflects the worst case scenario where each node's clock drifts in the opposite direction.

A time interval, guard time t_{guard}, is usually left vacant (i.e., during which no data is sent) on a transmission channel that can be used for synchronization and/or compensating for a signal distortion. 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 3. Since the relative time difference between two nodes can be in two direction, the guard time needs to be twice t_{diff},

$$t_{guard} = 2t_{diff} = 4\frac{T_{sync}}{T}\rho. \tag{11}$$

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

$$t_{slot} \geq t_{guard} + T_{tx}, \tag{12}$$

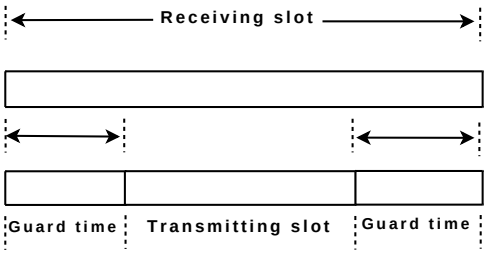


Fig. 3. Guard time of the nodes

where T_{tx} is the time required to send a packet from one node to other.

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

$$\frac{T_{sync}}{T} \geq 1. \tag{13}$$

To obtain the value of T_{sync} for a particular system with a given duty cycle, the following equations are to be seen:

$$t_{guard} \geq 4\rho \frac{T_{sync}}{T} \tag{14}$$

$$t_{slot} \geq t_{guard} + T_{tx} \tag{15}$$

$$\frac{T_{sync}}{T} \geq 1. \tag{16}$$

In general, there is a trade-off in determining T_{sync}: increasing T_{sync} reduces the energy costs of synchronization, whereas decreasing the network performance.

B. *Mathematical model*

As part of the message, the nodes send the slot number which they are transmitting to the neighbors. This information is used in determining the time that the message is sent. Each message that is sent within a slot is exactly received by a neighbor at a known clock tick number

$$tick_{rx} = T_{tx} + \frac{t_{guard}}{2} \tag{17}$$

Whenever (17) is not satisfied, a phase error is observed.

The wake-up time of a node after n periods of firings since it is turned on is

$$t_i^{(n)} = \sum_n T_i^{(n)} + t_{io}, \tag{18}$$

where $T_i^{(n)}$ is the period of the crystal clock at the n^{th} period which changes with time according to (4) and t_{io} is the initial start-up time of the node.

The difference in the wake-up time, (6), of the nodes is therefore given by

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

$$= (\sum_n T_i^{(n)} - \sum_n T_j^{(n)}) + (t_{io} - t_{jo}). \tag{20}$$

This phase error can be compensated by adjusting the next wake-up time depending on the difference between the current wake-up time of the node and its neighbors. 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,

$$t_i^{(n+1)} = t_i^{(n)} + T_i^{(n)} - \xi_i^{(n)}, \tag{21}$$

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

$$\xi_i = f(\Delta t_{ij}). \tag{22}$$

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.

Three algorithms are proposed in the next subsections.

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

The offset is then

$$\xi_i^{(n)} = \sum_j w_{ij}^{(n)} \Delta t_{ij}^{(n)}, \tag{23}$$

where $\sum_j w_{ij}^{(n)} = 1$ and w_{ij} is the weight factor for the phase error, Δt_{ij}.

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

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

The main task in this model is how to choose the weight factors so that the stability of the network (time wise) is achieved faster. The weight is selected in such a way 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 weight factor to which shows how far is the node concerning the time that it drifted away from its neighbors.

In the selection of the weight factor, two steps are taken.

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

$$\delta_{ij} = ae^{-b\Delta t_{ij}}. \tag{24}$$

The parameters a and b 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, the closer the phase error is to zero, the higher value of δ it is assigned. The assignment of high

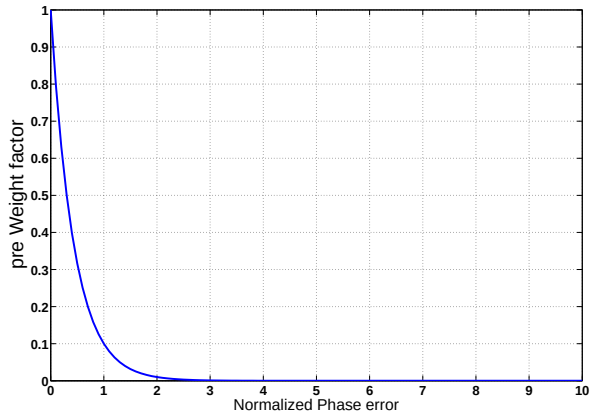


Fig. 4. pre-weight factors for the phase error distribution

values to small phase errors will decrease the time it takes to synchronize this node with the network.
Secondly, the weight factor is calculated based on the distribution of the assigned values for each phase error. If all or most of the phase errors are small, this corresponds to the fact that the node has drifted away from most of the neighbors. Hence, the synchronization is made to adjust the nodes wake-up time towards the neighbors.
Hence, the weight factor is

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

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

Therefore, in WM, the first step describes how the node has drifted away from its neighbors whereas the second step describes how the node is positioned in the network topology which surrounds it. Using the WM 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 neighboring nodes.

*2) **Least squares method**:* Least Squares (LS) is a method to fit a set of n observations with a model of m unknown parameters by minimizing the sum of the squares of the errors ("the residuals").

It uses a set of n data points, which are (x_1, y_1), (x_2, y_2),. . . ,(x_n, y_n), and a model function, $y = f(x, \beta)$, that in addition to the variable x also depends on m parameters($\beta_1, \beta_2, ..., \beta_m$). It is desired to find the vector β of parameters such that the curve best fits the given data in the least squares, that is, the sum of squares

$$S = \sum_{i=1}^{n} r_i^2, \tag{25}$$

is minimized, where the residuals r_i are given by

$$r_i = y_i - f(x_i, \beta), \tag{26}$$

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

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

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0 \quad for \quad j = 1, 2..., m. \tag{27}$$

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,

$$\beta_j^{p+1} = \beta_j^p + \Delta\beta_j. \tag{28}$$

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

$$f(x_i, \beta) \approx f(x_i, \beta^p) + \sum_j \frac{\partial f(x_i, \beta^p)}{\partial \beta_j} \left(\beta_j^p - \beta_j\right) \tag{29}$$

$$f(x_i, \beta) = f(x_i, \beta^p) + \sum_j J_{ij} \Delta\beta_j. \tag{30}$$

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,

$$\frac{\partial r_i}{\partial \beta_j} = -J_{ij} \tag{31}$$

and the residuals are given by

$$r_i = \Delta y_i - \sum_{j=1}^{m} J_{ij} \Delta\beta_j, \tag{32}$$

where

$$\Delta y_i = y_i - f(x_i, \beta^p). \tag{33}$$

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

$$\left(J^T J\right) \Delta\beta = J^T \Delta y. \tag{34}$$

Model design

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

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

$$f(x_i, \beta) = \beta_1 + \beta_2 log(x_i), \tag{35}$$

is chosen. A logarithmic function can be used to represent the distribution of the phase errors in the neighborhood, as

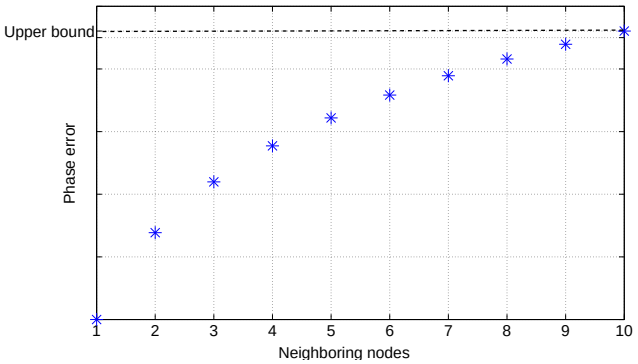


Fig. 5. Curve fitting using logarithmic function

shown in Figure 5. In the figure, the neighborhood is taken to have 50 nodes. Hence, the maximum phase error between the node and its neighbors i.e. upper bound is taken to be the guard time of the node, t_{guard}. Fitting the curve, the node will be synchronized with its neighbors. Normalized phase error values tend to be small and follow a curve in such away that the maximum value of the phase error should not be greater than the guard time.

The initial values of the parameters β_1 and β_2 are estimated using the state of a synchronized network taking into account. As each measurement arrives from the neighbors, the parameters are calculated in such a way that the measurement error from a pre-determined offset value is reduced. This ensures that the offset to be added in the next wake-up time doesn't diverge in a large amount from the predicted value.

*3) **Discrete time kalman filter for synchronization**:* The kalman filter[9] estimates a process by using a form of feedback control. It estimates the process state at some time and then obtains feedback in the form of (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 a priori estimates for the next time step.

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

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}, \tag{36}$$

with a measurement $z \in R$ that is

$$z_k = Hx_k + v_k. \tag{37}$$

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

$$p(w) \approx N(0, Q), \tag{38}$$

$$p(v) \approx N(0, R). \tag{39}$$

The $n\ x\ n$ matrix A in (36) relates the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. Note that in practice A might change with each time step, but here we assume it is constant. The n l matrix B relates the optional control input $u \in R$ to the state x. The $m\ x\ n$ matrix H in 37 the relates the state to the measurement z_k. In practice, H might change with each time step or measurement, but here we assume it is constant.

We define $\tilde{x}_k^- \in R$ to be our a priori state estimate at step k given knowledge of the process prior to step k, and $\tilde{x}_k \in R$ to be our a posteriori state estimate at step k given measurement z_k. Hence, a priori and a posteriori estimate errors are defined as

$$e_k^- = x_k \tilde{x}_k^-, \tag{40}$$

$$e_k = x_k \tilde{x}_k. \tag{41}$$

The a priori estimate error covariance is then

$$P_k^- = E[e_k^- e_k^{-T}], \tag{42}$$

and the a posteriori estimate error covariance is

$$P_k = E[e_k e_k^T]. \tag{43}$$

With the initial estimates of x_{k-1} and P_{k-1}, the predictor equations are

$$x_k = Hx_{k-1} + Bu_k, \tag{44}$$

$$P_k = HP_{k-1}H^T + Q. \tag{45}$$

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.

$$K_k = P_k H^T (HP_k H^T + R)^{-1} \tag{46}$$

$$x_k = x_k + K_k(z_k - Hx_x) \tag{47}$$

$$P_k = (I - K_k H)P_k. \tag{48}$$

The matrix K in (46) is chosen to be the gain or blending factor that minimizes the posteriori error covariance. Taking the derivative of the trace of the resulting derivative with respect to K, setting that result equal to zero, and then solving for K provides

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

$$K_k = \frac{P_k H^T}{HP_k H^T + R}. \tag{50}$$

Looking at (50), it is seen that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}. \tag{51}$$

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

$$\lim_{P_k \to 0} K_k = 0. \tag{52}$$

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

The update equation is

$$\tilde{x}_k = \tilde{x}_k + K(z_k - H\tilde{x}_k). \tag{53}$$

The difference $z_k - H\tilde{x}_k$ in (53) is 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.

Filter design

The transition matrix A plays an important role in achieving the proper synchronization. It indicates how fast/slow the next wake up time should be compared to it's previous value and/or neighbors. The initial estimates of x and P are selected from the previous firing time of the node, one period earlier. This ensures that the nodes are on the same track as the previous time, since the neighbors remain the same with some exception of mobility and interference. H is a factor used to compare the predicted value to the measured value.

In the filter design, the covariance matrices R and Q have a significant role in the overall implementation of the algorithm. R represents how the measurements are valued to affect the outcome and Q sends a signal as to how the estimated value is weighed.

IV. Results and Discussion

A. Simulation setup

The simulation parameters are presented in the following table.

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

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

B. Simulation results

1) ***Case I****:* In the first set of simulations, the synchronization error is simulated for the nodes which are static, hence no effect of mobility.

Figure $6a$ shows the synchronization error for 20 nodes operating in a static environment. The algorithms achieve a stable synchronization. As LS and WM has the same effect as the Median in case of stable network, the performance of both algorithms is similar with that of Median. Without the dynamic nature of the network, both tend to equal the average of the phase errors. KF has the best performacesince correcting the phase errors involves predict and update which results in better precision.

Figure $6b$ is the result of a simulation for 50 nodes. The synchronization error in general increases as the number of nodes increases. Comparing the individual algorithms, KF has the best precision whereas LS and WM have a better performance each than the Median. Median results in a lower performance on average.

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 20. The simulations are conducted for different speeds, at average speeds of $6km/hr$ and $20km/hr$. The chosen speeds emulate the speed of a walking man and an average speed of a slowly moving vehicle.

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

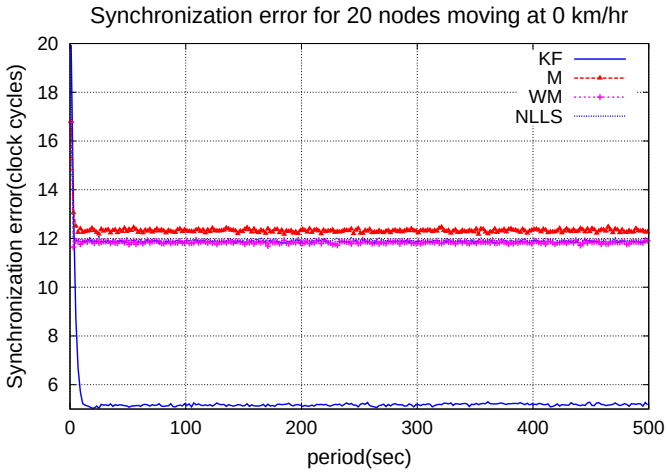
Figure $7a$ shows 20 nodes with a speed which is an gaussian random variable with mean $6km/hr$ and standard deviation of $1km/hr$. WM and LS show a better tolerance to dynamic networks due to the algorthms capapbilty to synchronize towards a more stable network. LS has also a better tolerance with less value given for large phase errors. Hence, WM and LS perform better concerning the synchronization of the frame than the Median. Median algorithm has a disruptions which made the network out-of-synchronization, resulting in high synchronization errors at times. This occurs due to the mobility of the nodes which disrupts the stability networks as it tries to compute the median of the phase errors, as indicated in Secton II. KF outperforms all due to its dynamic and recursive nature.

The speed of the nodes is changed to a random variable with mean $20km/hr$ and standard deviation $1km/hr$. The simulation results are presented in Figure $7b$. KF performs the best showing that the algorithm is stable in dynamic networks. WM and LS perform good when it comes to stability. As the dynamics of the networks increases, Median becomes more unstable and exhibits large synchronization error. LS and WM have a stable performance when the network becomes dynamic with constantly changing connection between the nodes.

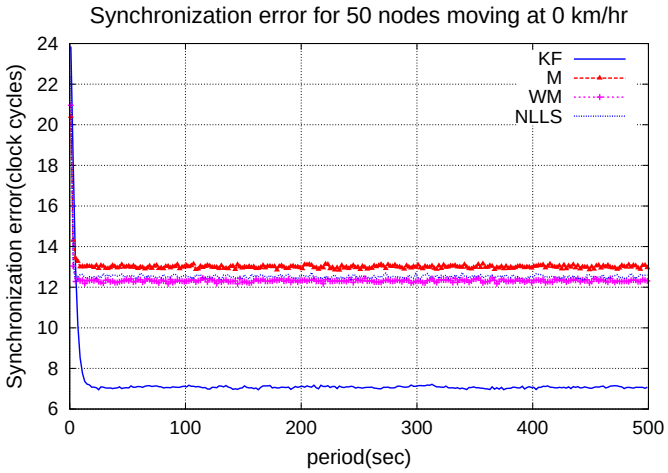
The relative comparison of the algorithms performance improvement with the Median is shown in Figure $8a$ for nodes moving at $20km/hr$. WM and LS perform, on average, $6-8\%$ better than Median, but with more tolerance in handling the disruptions. The best performer, KF, has on average 60% better performance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

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 $9a$. Large disruptions occur when using Median due to nodes moving slowly in the surrounding (leave the network and join again after some time), resulting in a larger drift with neighbors before getting back to the network. LS and WM perform better when it comes to stability, as out-of-sync nodes are forced to join the stable network. KF has the best preformance, both in precision and stability.

With a speed of $20km/hr$, the simulation results are presented in Figure $9b$. As the speed increases, the instability on the synchronization of the nodes also increases, which makes the Median more prone to high synchronization errors. WM has better tolerance towards mobility of the nodes, whereas the LS does well regarding stability. KF has the best

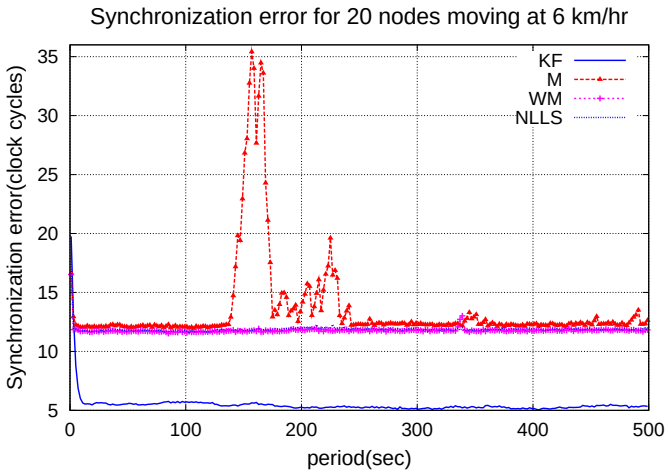


(a) Number of Nodes - 20

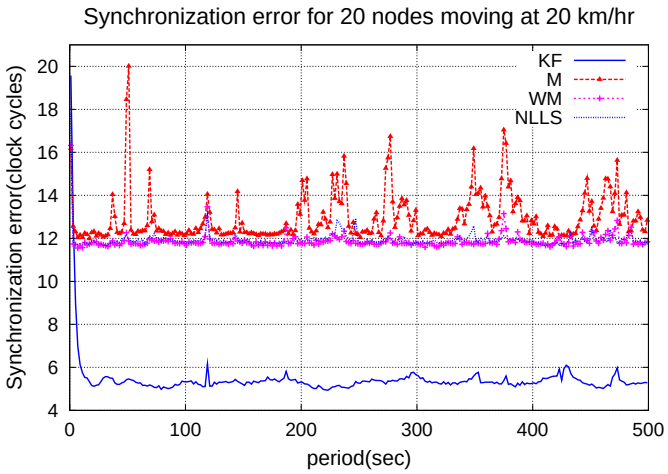


(b) Number of Nodes - 50

Fig. 6. Simulation results for Static Nodes

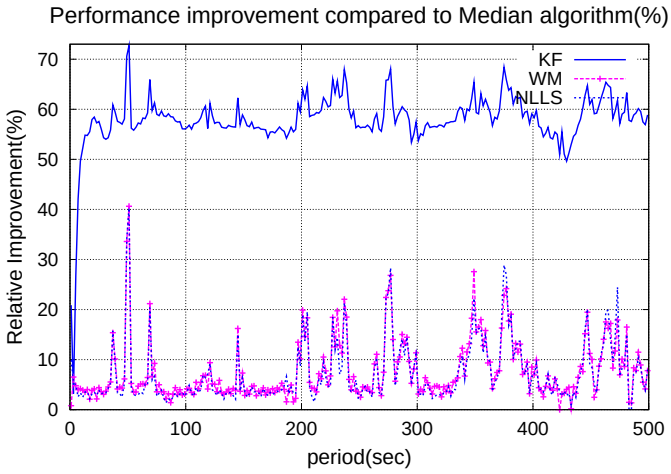


(a) Speed - 6km/hr

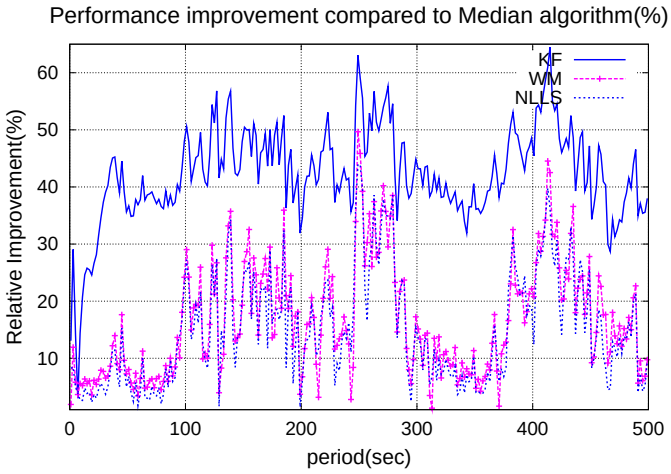


(b) Speed - 20km/hr

Fig. 7. Simulation results for 20 Nodes

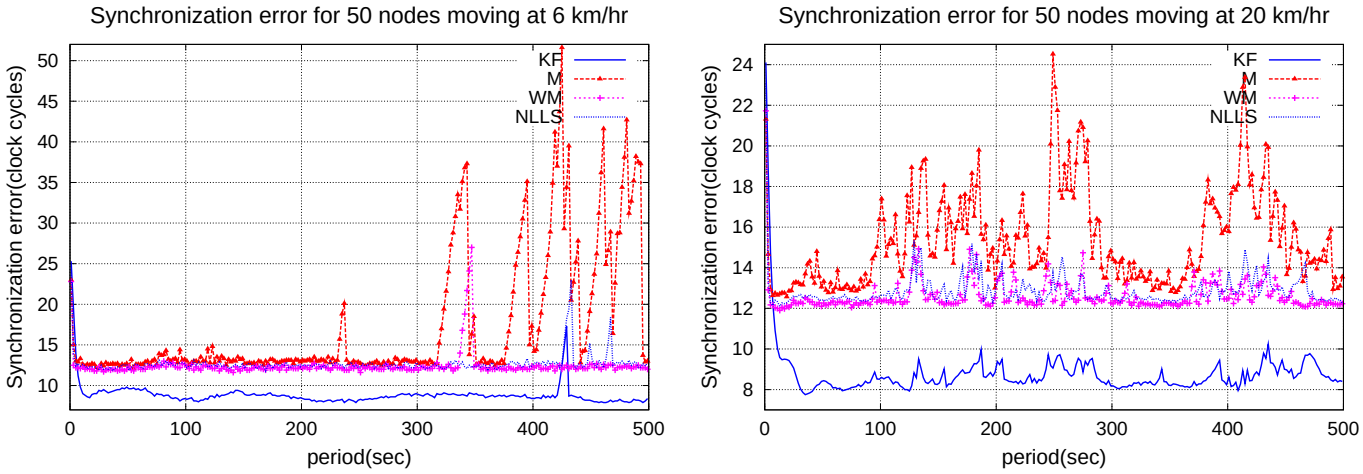


(a) for 20 nodes



(b) for 50 nodes

Fig. 8. Relative error of the algorithms with respect to Median



(a) Speed - 5km/hr (b) Speed - 20km/hr

Fig. 9. Simulation results for 50 Nodes

precision out of all. As the speed increases, the precision of the synchronization increases too since faster speed results in faster synchronization within the network.

For the set of nodes with a higher speed , a relative comparison is made to see the performance of the nodes with the Median algorithm. The Median algorithm is shown to perform the worst in this case. There are a lot of disruptions in the network, making it more unstable whereas the other algorithms adapt to the changes faster. Figure $8b$ shows that KF performs the best against Median algorithm, 45%. WM and LS perform well against Median too, 20% on average. It has been shown that the median is prone to error in case of high dynamics in the network.

The Kalman filter has a better precision and stability than the rest whereas WM and LS has a better tolerance to a dynamic network.

C. *Measurements concerning the energy consumption*

Additional energy expenditure

As the results in the previous subsection shows, KF algorithm performs well in all conditions, so do WM and LS when stability is a vital metric. The downside in implementing these algorithms, despite the performance gain achieved, is the energy consumption. As the algorithms are going to be implemented on the sensor nodes, the energy consumption is a priority in embedded system algorithm development. The algorithms are written in C and implemented on the test nodes to see the effect that they are going to have on the energy consumption of the nodes and presented below.

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

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

Energy gain

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

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

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

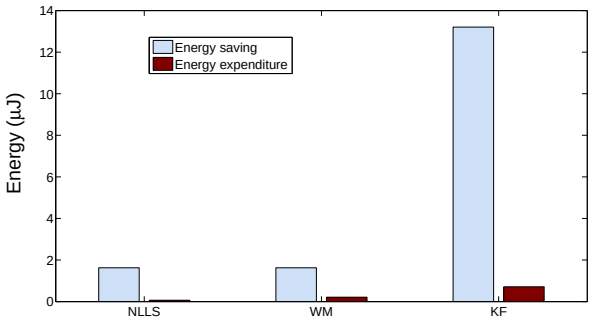


Fig. 10. The energy gain of reducing the guard time compared with the computational cost per RX slot

the length of the guard time is dependent on the number of slots, the energy to be saved increases as the number of slots increases.

V. Conclusion and Recommendation

Conclusion

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

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

WM and LS show a better tolerance in a dynamic network. Having a faster adaptation to the changes in the network made the algorithms preferred ones to a network characterized with dynamic behavior. KF estimation of the next wake up time is the best with its adaptive feedback nature. KF has also better stability, better resistant to topology changes occurring in the network.

Using these algorithms for a TDMA-based WSN synchronization (WM, LS and KF), the precision of the synchronization error as well as the stability increases. This in turn has a positive impact on the energy consumption of the nodes because the synchronization period, T_{sync} can be increased or the guard time of the node's frame, t_{guard}, can be reduced to achieve the same performance as the Median algorithm. Results are presented and discussed. But in the downside, the energy consumption of the algorithms is greater than the Median algorithm's energy consumption as the algorithms are more complex computationally. Analysis is made and presented about the energy saving made by decreasing the guard time of the slot. Communication is in general more expensive than computation on MyriaNodes, when put into perspective. A net gain of battery life can thus be achieved by using the proposed algorithms.

Recommendation

Different software power minimization techniques can be applied to further reduce the power consumption of the proposed algorithms, making them more energy efficient for implementation. In addition to that, evaluation and enhancements on the proposed algorithms can be done after implementing them on the MyriaNodes. Real time feedbacks can be used to improve the algorithms towards perfection.

VI. Acknowledgment

This research is part of the MyriaNed[17] Project. The author would like to thank Frits van der Wateren from Chess[18] 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] A.Ebner and et al. "Decentralized Slot Synchronization in Highly Dynamic Ad Hoc Networks," Wireless Personal Multimedia Communications, vol.2, pp.494-498, 2002.

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

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

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

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

[6] Symmetricom, Inc. white paper. "Stochastic Model Estimation of network time variance," USA.2004.

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

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

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

[10] J.Elson, L.Girod and D.Estrin. "Fine-grained network time synchronization using reference broadcasts," Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Vol.36 , Issue SI, pp.147-163, 2002.

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

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

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

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

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

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

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

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