Establishing Wireless Time-Triggered Communication using a Firefly Clock Synchronization Approach

Abstract

In South-East Asia, huge swarms of fireflies synchronously emit light flashes to attract mating partners. The underlying principle can be used to implement a robust and scalable distributed synchronization approach in wireless sensor networks. This paper describes the adaption of this principle for wireless sensor networks using an off the shelf IEEE 802.15.4 MAC layer. The method described in this paper applies the Reachback Firefly Algorithm on battery-powered low-cost wireless nodes to establish a wireless time-triggered network with a global notion of time. This global notion of time is used by the protocol but also provides a service that can be used by real-time applications. The synchronized nodes exercise a time-triggered communication, where the sending instant of each message is known a priori to all nodes. This enables the implementation of an energy-efficient low duty-cycle protocol, where sender and receiver units can be turned off during silent phases. The approach is evaluated by simulation and a real world case study using AVR®Z-Link™802.15.4/ZigBee nodes. Results exploring various network topologies, parameter choices, and realistic clock source deviations show that this approach works in multi-hop topologies and there exists the potentiality to save more than two thirds of a node's energy consumption.

1. Introduction

The time-triggered paradigm [3] is well established for high dependable systems such as X-by-wire applications in the automotive and avionics domain. The basic element is a global timebase that is distributed among the nodes through clock synchronization. All communication activities are scheduled according to a predefined, periodic scheme. This simple but robust scheme enables the design of dependable distributed systems and simplifies system verification and diagnosis.

However, the time-triggered approach is usually thought of not being very resource-efficient regarding its average performance and, therefore, was considered of minor interest to sensor networks with a high demand for energy efficiency so far.

In this paper we will show a different application of a time-triggered architecture, where the global synchronicity is used to enable synchronized sleep schedules in a wireless network cluster which can save a considerable amount of energy at each node. This is especially useful in situations with low duty-cycles, e. g., a sensor network that is utilizing only a fraction of its available bandwidth. In order to provide a common timebase we propose the application of a Reachback Firefly Algorithm (RFA), which is an algorithm inspired by the synchronous blinking behavior of natural fireflies in South-East Asia [2]. Thus we gain a robust self-organizing approach for synchronization without the need for dedicated time server nodes.

Due to the *a priori* known message schedule, the synchronized nodes are then able to predict the timing of incoming messages and can turn off their receivers when no transmissions of interest are scheduled. Since listening on the channel is a significant energy consumer of a typical wireless sensor node, the overall power consumer can thus be reduced in favor of battery lifetime. The global time can also support the application in tasks like timestamping, synchronous measurements, and timely coordinated distributed actions.

The objective of this paper is to describe a solution for a time-triggered wireless communication approach supporting scalability, graceful degradation, and efficient power management. The evaluation in this paper give realistic figures for the precision of the clock synchronization and the achievable savings in power consumption.

The rest of the paper is structured as follows: Section 2 describes the basic features and operation of the RFA. Section 3 presents the design of our approach consisting of clock synchronization, a modified RFA and an energy saving scheme. Section 4 and 5 describe the evaluation of a case study implementation by simulation and on real hardware. Results are discussed in Section 6. Related work is treated in Section 7. The paper is concluded in Section 8.

2. Reachback Firefly Algorithm

The RFA was introduced in [10] and supports complete scalability, graceful degradation, and a simple calculation. It is based on the Pulse-coupled Biological Oscillators (PCO) model [7], but with the difference that it is more appropriate for the practical implementation in wireless networks. For instance, the following assumptions from the original PCO model make a practical application very difficult: 1. The oscillators have identical dynamics, 2. Nodes can instantaneously fire, 3. Every firing event must be observed immediately, 4. All computations are performed perfectly and instantaneously.

The synchronization concept is based on a *state variable* $x = f(\phi)$, which characterizes an oscillator and corresponds to the charge of a firefly. The authors of the PCO model state that the *state function* $f : [0,1] \rightarrow [0,1]$ must be a smooth, monotonic increasing, and concave up function in order to achieve synchronicity. The *phase variable* ϕ therefore linearly increases with time. Note that we assume $x, \phi \in [0,1]$ for normalization. Thus, ϕ is characterized by (i) $d\phi/dt = 1/T$, where T denotes the cycle period, (ii) $\phi = 0$ at the beginning of a cycle, and (iii) $\phi = 1$ when the oscillator reaches the threshold $x = 1$.

The coupling between the oscillators is defined by the *firing function* $\phi'(\phi) = min(f^{-1}(f(\phi) + \varepsilon), 1)$ and depends on the *pulse strength* $\varepsilon > 0$. This function is always calculated immediately after an oscillator receives a firing event (or flash in case of a firefly). We further use the term of *phase advance* or *phase jump* to define the increase in the phase-domain, denoted by $\Delta(\phi) = \phi'(\phi) - \phi$. Thus, due to the concave down state function, a constant addition in the state-domain results in a variable increase in the phase-domain and the phase advance in the beginning of a cycle is smaller than later in the cycle.

To combat the assumption problems of the PCO model resulting from the limits in wireless networks, the RFA additionally uses the notion of reachback response and pre-emptive message staggering. In the original PCO model, an oscillator immediately reacts to each firing event. In contrast, the *reachback response* records the timestamps of all received firing events and calculates an overall phase jump once at the end of each period which is then applied at the beginning of the next cycle. Thus, if a node reaches the period end, it "reaches back in time" and reacts to the firing events of the past period. This principle is visualized in Figure 2. A further problem in the PCO model occurs in the case of an already synchronized network comprising several nodes. If so, all nodes will trigger the transmission event for the synchronization message at the same time. As a result, the messages will collide and the collision avoidance mechanism of the CSMA scheme takes effect. To avoid the random backoff in such a case, the *pre-emptive message staggering* explicitly adds a random transmission delay to the firing messages at the application layer. This reduces the probability of messages collisions.

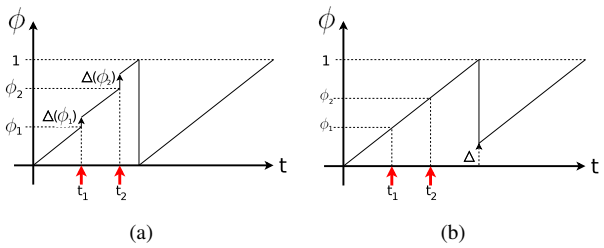


Figure 1. Comparison of the original PCO model (a) and the RFA (b). In the PCO-model, an oscillator immediately reacts to a firing event. In contrast, The RFA applies the overall phase jump at the beginning of the next cycle: $\Delta = \Delta(\Delta(\phi_1) + \phi_2)$.

3. Time-Triggered Approach

The principal purpose of many protocols used in sensor networks is aimed at reducing power consumption through synchronized sleep schedules. Such an approach is also referred to as a low duty-cycle concept where the transceiver module of all nodes is periodically activated only for a short time with a period length from seconds up to hours. In contrast to such protocols, our concept is based on a time-triggered approach where a node takes advantage of the *a priori* known transmission events. These events must be globally coordinated by the use of rounds and are stored in a file called Round Description List (RODL) file. In the current implementation such a round corresponds to a cycle of our synchronization algorithm. A round is further divided into a number of slots. Every node in a network must have its own RODL file and statically assigns a communication activity (e.g, idle slot, transmission, reception, task execution) to each slot in each round. This allows the setup of a collision-free communication and further improves the energy consumption by switching off the transceiver if it is not required.

The time-triggered approach also depends on the notion of a *global time*. Note that this is an abstract notion, because a distributed clock synchronization algorithm can only approximate the global time. Thus, the best achievable precision Π in an ensemble of clocks is lower bounded to the convergence function of the synchronization algorithm and the maximum drift rate ρ of all clocks in that ensemble. This is also known as the synchronization condition $\Pi \geq \Phi + \Gamma$. Therein $\Gamma = 2 \cdot \rho \cdot T$ denotes the drift offset and Γ the convergence function. In our approach the convergence function is defined by the RFA which periodically performs a state correction due to the reachback re-

sponse. In order to get promising results, the global time must be approximated with a very high precision. Thus we have to minimize the drift offset. This can either be done by using high quality crystal oscillators or a more frequent resynchronization. Both approaches have their drawbacks, because in mass production, crystal oscillators would be expensive compared to the cheap internal RC-oscillators in low-cost nodes. Furthermore, a shorter period time results in the exchange of more synchronization messages in the same time and thus would affect the energy performance. However, the reduction of the maximum drift rate ρ can also be achieved by a *rate correction algorithm*. In our approach, this algorithm is performed in the digital domain and makes use of the concept of virtual clocks. A *virtual clock* abstracts the physical clock by the use of macroticks. A macrotick comprises several microticks which are generated by a physical clock. The principle of this concept is to change the number of microticks a macrotick contains in order to adjust the granularity respectively frequency of the virtual clock. In the current implementation a macrotick corresponds to a complete cycle length. Thus, the duration of the periods can easily be changed by adjusting the threshold value of the physical timer/counter.

3.1. Clock State Correction

The clock state synchronization is established by the RFA model and uses the definition of the smooth, monotonic increasing, and concave down state function from [7] to calculate the overall phase jump Δ. The original state function in the PCO model is defined to be

$$f(\phi) = \frac{1}{b} \cdot ln(1 + [e^b - 1] \cdot \phi) \tag{1}$$

where the parameter b is called *dissipation factor* and measures the extent to which $f(\phi)$ is concave down. Consider the dissipation factor is bigger than 1 and the pulse strength within $0 < \varepsilon < 1$, then the phase jump equals

$$\Delta(\phi) = \min(1, f^{-1}(f(\phi) + \varepsilon)) - \phi. \tag{2}$$

The direct implementation of all these functions would result in a time-consuming calculation process. Therefore, we simplified the equation by inserting the inverse function f^{-1} in Equation 2. Let $f^{-1}(x) = \frac{e^{bx}-1}{e^b-1}$, then the simplified phase jump equals

$$\Delta(\phi) = \min(1, \alpha \cdot \phi + \beta) - \phi \tag{3}$$

where $\alpha = e^{\varepsilon b}$ and $\beta = \frac{\alpha-1}{e^b-1}$. Assuming a strong dissipation factor $b \gg 1$, then β is negligible and the approximated phase jump can be reduced to

$$\Delta(\phi) = \min(1, \alpha \cdot \phi) - \phi. \tag{4}$$

As a result, we have a linear Phase Response Curve (PRC), where the *coupling factor* α specifies the strength of coupling between the oscillators and depends on the product of the dissipation factor b and the pulse strength ε. This result is similar to the simplified firing function described in [10].

In order to improve the achievable synchronization precision, we further figured out the most significant cause of the communication delay which results from the media access strategy at the Media Access Control (MAC) layer. The best way to eliminate this delay is to measure it and include it into every transmitted message. We implemented this measurement by the use of MAC-timestamping. In simple terms, we measure the time interval between a transmission event is triggered at the application layer and the start of the transmission at the physical layer. The receiver does the same reversely. As a result, the application layer has knowledge about the MAC-specific delay a message encounters at the transmitter and the receiver. Considering our RFA approach, the receiver then compensates this delay by including it into the message staggering delay of a synchronization message.

However, the MAC-timestamping in the current implementation does not consider the backoff timer of the CSMA/CA scheme in the case of message collisions. Therefore, the network suffers from delay jitter resulting from the collision avoidance mechanism. The *delay jitter* ε is defined to be the maximum absolute deviation of the delay a message encounters during the communication. The reason why we have not incorporated the backoff delay is that we used an off the shelf MAC-stack which was challenging to modify. Furthermore, the purpose of this work should also demonstrate that such a synchronization approach works with an off the shelf communication stack without MAC-timestamping. Unfortunately, due to the impossibility result [6], this delay jitter lower bounds the best achievable synchronization precision. In [6], Lundelius and Lynch state that the synchronization precision in a network comprising of N perfect clocks is lower bounded to $\varepsilon \cdot \left(1 - \frac{1}{N}\right)$.

Lower bound for the coupling factor α. To obtain good synchronization results with respect to different parameter choices, it is important to find bounds for these parameters. Further we want to calculate the optimal coupling factor with respect to the clock drift, so that the Firefly algorithm can synchronize the nodes. For this reason, we first denote a virtual clock of node k with VT^k. Next we want to get the duration of the synchronization interval of a specific clock k. For this, we introduce the term I^k, which determines the interval of clock k with respect to a very precise reference clock z.

$$I^k = z(VC^k_{i+T}) - z(VC^k_i)$$

Now we need to define the maximum deviation between the intervals of any clocks in an ensemble and is herein after

referred to as the *virtual clock skew*, denoted by T_{skew}.

$$T_{skew} = \max_{\forall j,k}(|I^j - I^k|)$$

Without loss of generality we assume $I^j > I^k$. If the Firefly algorithm is based on Equation 4, then the coupling factor α must be greater than $\frac{I^j}{I^k}$. Otherwise, the system will never get synchronized. Regarding an ensemble of clocks with a maximum drift rate ρ, then we can also define the lower bound by

$$\alpha > \frac{1+\rho}{1-\rho}.$$

The proof is trivial. Consider the clocks are already perfectly synchronized, but because of the virtual clock skew the node with the shorter interval I^k will reach the phase threshold ϕ_{th} earlier than the other one with the interval I^j. Consequently, node k fires and node j receives the fire event at ϕ^j_{fire}, where $\phi^j_{fire} = \phi_{th} \cdot \frac{I^k}{I^j}$. Further the phase jump performed by clock j at the phase ϕ^j_{fire} is denoted with $\Delta(\phi^j_{fire})$. In order to keep the system synchronized, the following equation must be valid: $\Delta(\phi^j_{fire}) + \phi^j_{fire} \geq \phi_{th}$ Otherwise, the system will definitely get unsynchronized. Note that ϕ_{th} is defined to be 1. The result after substituting $\Delta(\phi^j_{fire})$ with Equation 4 is $\alpha \geq \frac{I^j}{I^k}$. Hence, if the equation above is valid, the precision of an ensemble of clocks based on the Firefly algorithm is lower bounded to T_{skew} and the overall synchronization interval is determined by the clock with the shortest period.

Note that in the absence of a rate correction mechanism, a good choice for the coupling factor is $\alpha = \frac{1+3\rho}{1-\rho}$.

Upper bound for the coupling factor α. It is obvious that the coupling factor α for the phase jump function must have an upper bound. Otherwise, the clocks may not get synchronized due to a mutual excitation. For instance, if α is so large that the phase jump $\Delta(\phi)$ at the phase ϕ always returns a phase jump of about $1-\phi$, then this will definitely keep a network of more than two nodes unsynchronized. Note that if a network is already perfectly synchronized and we neglect the communication jitter and any inaccuracies, then the system will keep synchronized independent of the choice of the coupling factor α.

The coupling factor α must be chosen, so that the condition

$$\alpha < \frac{2 \cdot N}{2 \cdot N - 1}$$

is valid, where N denotes the maximum number of firing events a node may receive within a period, i.e., the maximum number of phase jumps a node may perform during a single period. The equation is based on the assumption that the maximum allowed overall phase jump must not be greater than half the threshold phase ϕ_{th}. This comes from our implementation, since we decided that if the phase jump is greater, then a node should not transmit a synchronization message. This is a necessary, because otherwise the new start phase could be withing the message staggering delay. Therefore, we denote the maximum possible phase response with Δ_{max} which equals $\phi_{th} - \frac{\phi_{th}}{\alpha}$. Next we assume the worst case that a node performs at most N phase jumps. This corresponds to the reception of N firing events. Consequently, the equation $N \cdot \Delta_{max} < \frac{\phi_{th}}{2}$ must hold and after solving the equation for α, we come to the result stated above.

It is important that there exist another upper bound which comes from the original RFA model, because there may exist initial configurations of an all-to-all topology such that the nodes achieve a fixpoint. For instance, given two nodes A and B with the corresponding phases ϕ_A and ϕ_B immediately after A has reached the period end, where $\phi_A < \phi_B$, then the following condition must hold: $\alpha < \phi_{th}/min(\phi_B, \phi_{th} - \phi_A, \phi_{th} + \phi_A - \phi_B)$. Note that ϕ_{th} defines the specified threshold of the virtual clock and corresponds to the period length. This variable is usually normalized to 1. If the condition does not hold, then the phase jump function can be reduced to $\Delta(\phi) = 1 - \phi$. This results in a fixpoint and the nodes will periodically change their phase state without achieving synchronicity. It can be shown that the minimum coupling factor for two nodes occurs when $\phi_A = \frac{\phi_{th}}{3}$ and $\phi_B = \frac{2\phi_{th}}{3}$. Therefore, if $\alpha < \frac{3}{2}$, then the network will never result in a fixpoint. Note that such fixpoints also exist in larger topologies, but are more complex and thus occur more infrequent. However, in reality natural inaccuracies and jitter will let such a network eventually become synchronized with a high time-to-sync.

Rate of synchronization. The authors from [7] and [10] have analyzed the synchronization rate for a network with two oscillators. In this case, they have proven that the time to synchrony is inversely proportional to the coupling factor α and further depends on the initial phase difference of the nodes at $t = 0$, denoted by $\delta = \phi^{(0)}_A - \phi^{(0)}_B$. Therefore, the number of iterations k until synchronicity equals

$$k \approx \frac{1}{\alpha} \cdot \ln \frac{1}{2 \cdot \delta - 1}.$$

The authors have also analyzed the case of n oscillators. However, considering a multi-hop topology requires a more sophisticated solution and is stated in [5].

3.2. Clock Rate Calibration

The concept of clock rate calibration combats the problem of frequency deviations due to the high clock drift of the RC-oscillators usually used in low-cost devices. This approach should allow a longer resynchronization interval with the same synchronization precision.

The core concept of our rate calibration algorithm is that a node measures and stores the durations of one or more

synchronization intervals of all neighboring nodes. The node then smoothly adapts the own synchronization interval to the average of all the measured intervals. Note that a positive side effect of this approach is that the partial averaging of several intervals from the same node also reduces the influence of the delay jitter. The rate adaption is performed by changing the threshold value of the physical timer/counter which also represents the duration of a macrotick of the virtual clock. This is done by adding an adjustment value H to the threshold. If the adjustment value is positive, then the synchronization interval is getting longer. Otherwise, a negative adjustment value results in a shorter synchronization interval.

Computation and employment of the phase adjustment value H. In order to compute the phase adjustment value, we have to calculate the overall average interval. For this reason, we denote a receiver node with the letter r and the sender nodes with the variable j, where j can acquire a value from 1 to n. In order to measure the interval, every firing event has to be timestamped both at the transmitter and the receiver. For instance, if the firing event e^j_{fire} from the sender j is received at the node r, then the timestamp of this event at the receiver is denoted by $C^r(e^j_{fire})$. To distinguish different firing events from the same node, the events are chronologically ordered and numbered serially. Thus, two consecutive firing events from a node j are labeled with $e^j_{fire,k}$ and $e^j_{fire,k+1}$, where k defines the chronologically ordered position in the *rate-calibration buffer* and can take a value from 1 up to the maximum capacity of the buffer, denoted with m. The duration between two synchronization messages from a sender j in the receiver's clock granularity corresponds to the time difference between the two timestamps and is declared as $I^r_{j,k} = C^r(e^j_{fire,k+1}) - C^r(e^j_{fire,k})$.

Let $C^j(e^j_{fire,k})$ be the timestamp from the sender before the synchronization message is transmitted, then we can denote the equivalence $I^j_{j,k} \equiv I^r_{j,k}$. The number of ticks in the receiver's granularity for the corresponding nominal number of ticks at the sender can now be calculated with the term $\tilde{I}^r_{j,k} = \phi_{th} \cdot I^r_{j,k}/I^j_{j,k}$. Note that the nominal number of ticks is the same for all nodes and equals the nominal threshold value of the physical timer/counter, denoted by ϕ_{th}.

Now we have measured the duration of the neighbor's nominal synchronization intervals in the receiver's clock granularity. However, this interval does not reflect the real adjusted interval length of a node due to the rate calibration scheme. For this reason, the receiver has to scale the measured nominal intervals according to the corresponding sender's latest phase adjustment value, denoted by H^j. The resulting sender's actual synchronization interval in the receiver's granularity corresponds to

$$\hat{I}^r_{j,k} = \tilde{I}^r_{j,k} \cdot \frac{\phi_{th} + H^j}{\phi_{th}}.$$

As a next step, all scaled intervals of the same node are averaged. This ensures that the error due to the jitter and imprecision in computation is reduced. If the capacity of the rate-calibration buffer is big enough, then the error should be negligible. Our experiments have shown that a buffering over eight periods is good enough for also compensating short-term drift and requires not so much memory. Further, we denote the average interval at the receiver r for each sender node j with $\bar{I}^r_j$ and equals

$$\bar{I}^r_j = \frac{1}{m-1} \sum_{k=1}^{m-1} \hat{I}^r_{j,k}$$

where m denotes the capacity of the rate-calibration buffer.

In order to involve the occurrence of erroneous nodes, we have introduced a simple concept to avoid the worst case. For instance, a network may contain a node with a clock drift, which heavily deviates from the other ones. Consequently, the network would take a long time to get synchronized with a bad precision, or in the worst case would never get synchronized. A simple way to exclude such an erroneous node is to remove the nodes with the biggest absolute interval deviation with respect to the average interval.

It is obvious that H can be calculated through the difference between the overall average interval $\bar{\mathcal{I}}^r$ and the nominal interval. However, the results from several experiments have shown that such a direct adjustment results in a common-mode drift over all clocks. In other words, the average interval over all clocks is getting longer or shorter. For this reason, we decided to introduce a regularization parameter called smoothing factor, denoted by σ. This should ensure that the virtual clocks smoothly converge to the overall average interval. The equation for this calculation is

$$H^r = H^r_{old} \cdot (1 - \sigma) + (\bar{\mathcal{I}}^r - \phi_{th}) \cdot \sigma.$$

The smoothing factor σ defines the level of smoothness and must be in the range of $[0, 1]$. In detail, if σ is very small, then the adaption will be performed slowly. Otherwise, a big σ results in a fast adaption. The experiments show that a small σ let the overall average interval getting shorter and a bigger σ lets it getting longer. It seems that if the measurement interval is greater than one period, then $\frac{1}{2}$ is mostly the swell for the change of the drift behavior of the common-mode drift. This property makes it possible to use it as a control parameter.

Common-mode drift stabilization. The smoothing factor σ has an impact on the overall interval drift. It seems to be impossible to hold the duration of the common period

constant. However, this factor can be used to compensate this problem. The midpoint value for the drift change was observed to be about $\sigma = \frac{1}{2}$. For this reason, we only have to find a parameter which can be taken to control the value of the smoothing factor. Such a parameter could be the average over all received adjustment values H_j during a period.

3.3. Energy Awareness

The energy consumption is an important quality characteristic of each communication protocol used in sensor networks. Often more than 50 percent of energy is used for idle listening [11]. Therefore, it is necessary to reduce the major energy sources. Some MAC protocols have already incorporated such a concept (e.g. S-MAC, T-MAC, etc.). However, we assume that the underlying MAC layer is only responsible for the medium access control and not for energy improvements. For this reason, we assign the tasks for energy reduction to the upper layers.

In order to reduce power consumption caused by idle listening, it is necessary to turn off the transceiver module as much as possible. In literature a protocol based on such a scheme is called to be a duty-cycle protocol. In such protocols a node becomes dormant most of the time and only wakes up for a short time if it is necessary. The *duty-cycle* is determined to be the ratio between the duration used for listening to the medium and the duration of the complete period. Note that the bounded synchronization precision necessitates that the receiver module must be enabled some time prior, before any transmission is already started. To guarantee this behavior, the difference of the point in time the receiver is enabled and the first transmission may start should be greater than the synchronization window w, where w defines the upper bound of the synchronization precision. A good way is to choose $2 \cdot w$. Therefore, we say that a node is synchronized, if the maximum absolute deviation to all other nodes is smaller than the specified synchronization window. Last but not least, it is important that after a number of periods, each node enables the transceiver for a complete period. This idle listening avoids clique building in the case a synchronized cluster is not able to receive messages from another cluster which may has a completely different synchronization schedule.

4. Evaluation by Simulation

We evaluated our approach with a probabilistic wireless sensor network simulator called JProwler [1]. JProwler has been developed by the Institute of Software Integrated Systems at the University of Vanderbilt and is basically configured to simulate the behavior of Berkeley Motes running TinyOS. For this reason, JProwler also provides the simulation of the standard MAC protocol used in TinyOs. It is a Java version of the Prowler[8] network simulator which is used for verifying and analyzing communication protocols of ad-hoc wireless sensor networks.

[1] ISIS, Institute For Software Integrated Systems: http://www.isis.vanderbilt.edu/Projects/nest/jprowler

4.1. Simulation Environment

In order to visualize the influence of several parameter choices, we enhanced the graphical user interface by several new dialogs which enables the user to modify various parameters during the simulation. JProwler was modified in order to simulate the behavior of our testbed system. We further extended the simulator by an oscillator model. Thus, every virtual node must be based on an oscillator, e.g., RC-oscillator or several crystal cuts. This allows the simulation of clock drift and the influence on the clock synchronization. Due to the fact that the frequency of an oscillator heavily depends on the supply voltage and the ambient temperature, the enhanced JProwler also contains the simulation of the ambient temperature. Other new features the adjustment of the simulation speed, enabling/disabling nodes during the simulation, and so on.

4.2. Experiments and Results

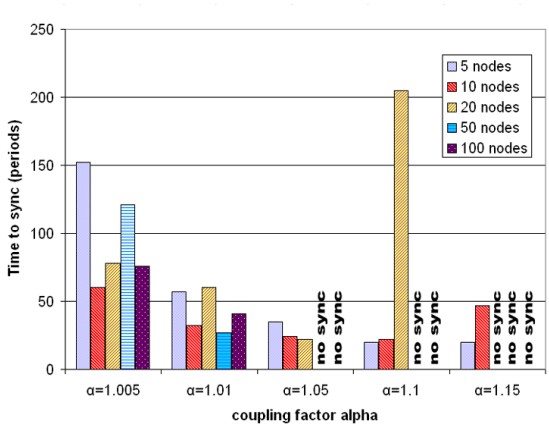
The simulation results discussed in this chapter should give an overview of the achievable quality of our synchronization approach. For this reason, several network topologies have been developed and simulated. The results are compared due to different parameter choices, i.e., the coupling factor α and the number of nodes in the network.

4.2.1 Evaluation Metrics

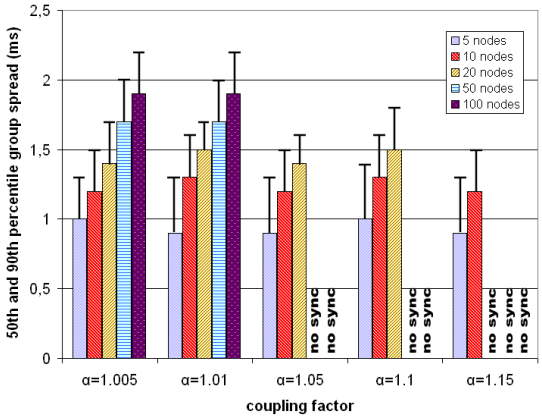
In order to compare the simulation results with the outcomes from [10], the evaluation metrics are similar. Therefore, the two important parameters are the amount of time until the system achieves synchronicity and the quality of precision.

Time To Sync: This metric defines the time until all nodes have entered the synchronization state whereas the time to sync is determined by two parameters. These are the synchronization window w and the number of required periods where a node keeps within this window. In the following we call the amount of required periods *synchronization periods* and is usually set to 10. A node only enters the sync-state, if the maximum absolute deviation with respect to the other nodes is within the synchronization window for 10 out of the last 11 firing iterations.

50th and 90th Percentile Group Spread: This metric differs from that one defined in [10], because we only



(a) Time to sync diagram



(b) Group spread diagram

Figure 2. The time to sync and the group spread for an all-to-all topology experiment in dependence of the network diameter and different coupling factors. The solid bars in (b) represent the 50th percentile group spread, while the error bars correspond to the 90th percentile.

refer to one firing group. Therefore, the group spread in the simulation is defined to be the maximum absolute time difference between any two nodes in the network and thus cannot be greater than half the synchronization interval. We characterize the group spread distribution with the 50th and 90th percentile.

In order to avoid incorrect results due to settling effects during the startup phase, the start of the group spread measurement is postponed against the time to sync t_s and the time the experiment end t_e. On this account, the group spread measurement is performed during the interval $[t_s + \frac{t_e - t_s}{2}, t_e]$.

4.2.2 Parameter Settings

Several parameter settings are the same for all experiments and are adapted to simulate the behavior of our testbed environment. For instance, every virtual node is based on a virtual RC-oscillator. Regarding the datasheet, the real nodes have a nominal frequency of $8MHz \pm 10\%$. For this reason, every virtual node encounters a random initial clock drift between $-100ms$ and $+100ms$ per second. The general values of the other parameters are denoted in the table from Figure 3.

4.2.3 Simulation Results

The next paragraphs discuss the simulation results in dependence of several network topologies and parameter choices. Every configuration was simulated over 3600 periods.

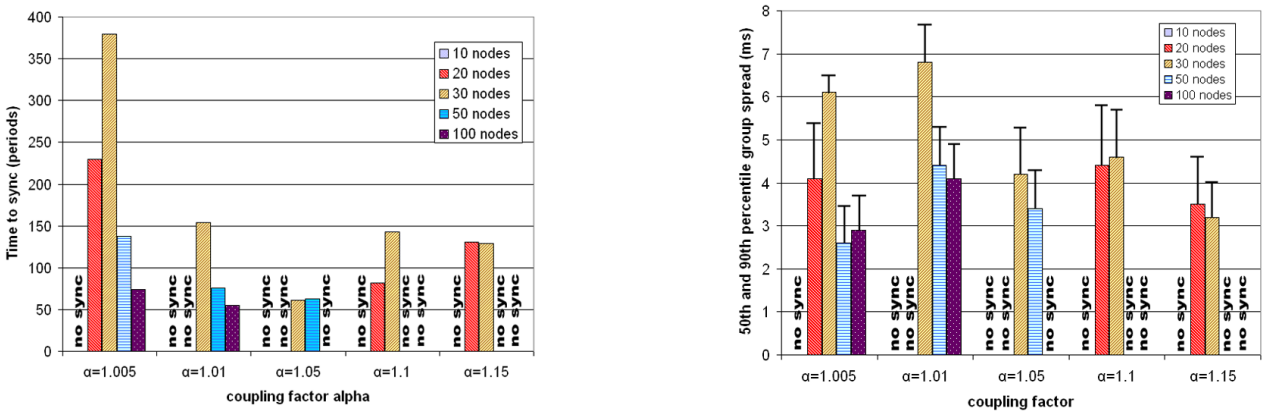
Parameter	Value
Oscillator technology	RC-oscillator
Initial clock drift $[ppm]$	$\pm$100000
Interval time $[ms]$	1000
Granularity	10000 ticks/period
Coupling factor	1.01
Transmission delay $[\mu s]$	375
Delay jitter $[\mu s]$	1250
Synchronization window $[ms]$	10
Evaluation end [periods]	3600

Figure 3. The general parameter choice used in all simulator experiments.

The all-to-all topology. The all-to-all communication topology is mainly used to measure the quality of the synchronization in dependence of the number of nodes and the coupling factor α. Therein, every node is in the transmission range of every other node.

The simulation results based on this topology should give a good overview on the impact of different coupling factors. According to the diagrams in Figure 2, the time to sync decreases with an increasing coupling factor α. If the factor is too big, then synchronicity will not be achieved. This effect is due to the upper bound of the coupling factor. It seems that the group spread is the same and thus independent of the coupling factor. The high time to sync bar in Figure 2(a) with $\alpha = 1.1$ and a number of 20 nodes comes from the fact that the coupling factor was too high.

The multi-hop topology. This communication topology is the most important one, because in reality many sen-



(a) Time to sync diagram

(b) Group spread diagram

Figure 4. The time to sync and the group spread for a multi-hop topology with a network diameter of 10 in dependence of the cluster size and different coupling factors. Note that the number of nodes must be divided by 10 to get the group size. The solid bars in (b) represent the 50th percentile group spread, while the error bars correspond to the 90th percentile.

sor network are based on a source-to-sink communication topology with a communication path consisting several hops. The simplest multi-hop scenario is a network comprising n nodes, which are ordered in a chain and can only communicate with the immediate neighbors. We further call the chain size *network diameter*. Such a network with a big network diameter is often very problematic to synchronize, because every hop involves a communication delay, which degrades the overall synchronization precision. Our solution is based on grouped multi-hop networks. Therein, the nodes are replaced with groups comprising several nodes in all-to-all topology which all have a bidirectional communication link to the immediate neighboring groups. Note that all grouped multi-hop topologies treated in our experiments have the same network diameter of 10 hops but vary in the group size.

The diagrams in Figure 4 shows the time to sync and the group spread in dependence of a different group size and coupling factor. The network diameter is always 10 hops. These diagrams leads to the result that the precision increases with a bigger group size. This effect is caused by the better information about the interval drift due to the increased number of neighboring nodes. If a node has more neighboring nodes, then the node receives more information about the clock drift and can more precisely calibrate the interval duration, which also improves the synchronization precision. However, it is also important to have a preferably small coupling factor. On the one hand this increases the time to sync, but on the other hand this also increases the possibility that the network achieves synchronicity. As a result, it is difficult to find the best parameter settings for a given multi-hop network, but it is definitely a good choice to have a group size of more than one node. This also increases the dependability and availability of the network.

A ring topology experiment. The ring topology was also simulated with terms of asynchronous communication patterns, i.e., unidirectional communication links. Other experiments concerning asynchronous communication have shown that this heavily affects the synchronization, especially if the network is based on a multi-hop topology. In some cases, the network never achieved synchronicity. Experiments with a ring topology have partially disproved the problem of asynchronous communication. The unidirectional ring topology experiments lead to the result that network synchronicity can be achieved, even in the presence of asymmetric connections. The important prerequisite for achieving synchronicity in such a network is that according to the graph theory, for every two nodes v and w, there must exist a closed directed path, with repeated nodes allowed. Note that the cycle length is also an indicator for the convergence time and achievable synchronization precision.

5. Evaluation on Real Hardware

The simulation results provide a good basis for several parameter estimations in order to optimize the synchronization precision for different network topologies. However, the simulator does not support information about power consumption and further never fully reflects the real world scenario. For this reason, we implemented and evaluated our distributed algorithm in combination with the time-triggered approach on real hardware.

5.1. Testbed Description

The testbed is based on Atmel's demonstration kit ATAVRRZ200 [1]. The kit features two component boards: The Display Board and the Remote Controller Board (RCB)s. The Display Board is based on an Atmega128 controller and features an LCD-module. This board also works as a docking station for programming the RCBs. The RCBs therefore are based on an Atmega1281 controller and contain an AT86RF230 (2450 MHz band) radio transceiver. The implementation of our approach is done with the AVR®Z-Link™802.15.4/ZigBee nodes. The information about the synchronization precision is gathered via the established TDMA scheme. Therefore, beside energy saving, the time-triggered approach also serves as an evaluation protocol. For this, we used a modified version of the TTP/A protocol [4].

The synchronization algorithm was implemented analogously to the implementation in JProwler. A simple RC-oscillator based 16-bit timer was used to generate the synchronization interval with a duration of one second. The only differences with respect to the parameter choices in Figure 3 are a higher granularity of the virtual clock (31250 ticks/period) and a higher transmission delay of $896\mu s$.

We modified the initial settings of the MAC sublayer, i.e. the minimum backoff exponent, to reduce the transmission delay. We further assumed that it is better to lose a message than to transmit postponed synchronization data, where the time information is out of date. In our implementation, every node is configured as a Full-function Device (FFD) and no association process is required. This necessitates the use of individual predefined addresses for each node. This address corresponds to a 16-bit short address which is originally assigned by a coordinator during the association process.

5.2. Experiments and Results

The evaluation metrics for the testbed experiments are similar to the one used for the simulation experiments. Because it is not easy to observe the relative deviations over all nodes in a network, we decided that every node transmits its own maximum absolute deviation of the last period to a central evaluation node, which then calculates the maximum over all received deviations. These values over several minutes are then taken to compute the 50th and the 90th percentile group spread.

To be able to compare the testbed results with the simulator results, the parameter configuration has to be the same as used in the simulator experiments. Unfortunately, in reality it is not possible to speedup the time. For this reason, we reduced the simulation end to 720 periods.

5.2.1 Testbed Results

The all-to-all topology. For the all-to-all topology experiment, we have measured the group spread in dependence of several coupling factors. The results of such a network comprising 5 ZigBee nodes were visualized as histograms. All histograms in dependence of different coupling factors look similar and have a right-skewed distribution.

Parameter choice	Time to sync (periods)	50th-percentile (μs)	90th-percentile (μs)	Maximum deviation (μs)	Standard deviation (μs)
$\alpha = 1.005$	105 (152)	672 (1000)	2005 (1300)	3456 (2200)	538 (257)
$\alpha = 1.010$	79 (57)	704 (900)	1632 (1300)	2592 (2000)	410 (250)
$\alpha = 1.050$	24 (35)	704 (900)	1973 (1300)	3040 (1900)	501 (262)
$\alpha = 1.100$	33 (20)	672 (1000)	1723 (1400)	3104 (2000)	451 (267)
$\alpha = 1.150$	14 (20)	732 (900)	1965 (1300)	3776 (1800)	565 (250)

Figure 5. Comparison of several parameters in dependence of different coupling factors. The values between the brackets correspond to the simulation results with the same all-to-all network comprising 5 nodes.

The table in Figure 5 contains the simulation results and the testbed results with the same network configuration comprising 5 nodes in all-to-all topology. This demonstrates that the results are similar. For instance, the time to sync and the 50th percentile group spread of the testbed systeme are mostly better than the simulation results. Only the 90th percentile group spread and the maximum group spread are worse compared to the simulation results. The results from an all-to-all topology comprising 9 nodes are comparable with those denoted in Figure 5. Note that only a maximum number of 9 nodes were available for our experiments. We investigated the behavior of precision degradation and found out that the testbed system suffered from an unexpected delay jitter. Furthermore, the higher transmission delay also degrades the precision. Note that we have already compensated the constant delays in the implementation. However, there may exist other delays which we have not considered. Simulation experiments have shown that a higher transmission delay or a higher delay jitter are the major reasons for the precision degradation due to the state correction algorithm and hardly affect the rate calibration scheme.

The multi-hop topology. The results from the multi-hop experiments are important in order to get an overview about the limits of our synchronization approach. The first scenario was made up of 5 nodes ordered in a chain, where a node can only communicate with the immediate neighbors. The only difference between the simulation and the testbed environment is that the testbed environment does not have an omniscient observer, which is able to continuously measure the synchronization deviation among all

nodes. For this reason, we decided to measure the time difference between the edge nodes with the aid of an oscilloscope, whereas each node periodically sets an output pin at the same phase state for a short time. Unfortunately, these measurements can not be gathered automatically over several periods. Therefore, we manually made snapshots over several minutes and took those diagrams, which display the biggest time deviation. The results show that the precision of a realistic multi-hop network with 4 hops is about $3ms$. It is interesting that the simulation of the same network with a configured delay jitter of $1250\mu s$ lead to the same result.

To get an overview of the precision degradation with respect to the network diameter, another multi-hop experiment with 9 nodes was performed. The measurement setup is similar to the previous multi-hop network. The measurement results show a maximum deviation between the edge nodes of up to $14ms$. It is obvious that such a precision is unacceptable. In summary, our synchronization algorithm dramatically degrades with each hop and is therefore not applicable for the use in a real sensor network application. These bad results highly likely come from the delay jitter. The simulator presented similar results, if we adjust a delay jitter of $2500\mu s$. The higher delay jitter in the testbed system may result from the fact that each node in the testbed environment is in the communication range of each other. To simulate the multi-hop network, we have simply implemented a message filter. Thus the probability of message collisions in the testbed environment is higher than in the simulation environment and therefore may cause the unexpected higher delay jitter.

We further measured the behavior of a grouped multi-hop network comprising 3 clusters with a cluster size of 2 and additional two edge nodes which have a communication link to the corresponding two edge groups. This was the only acceptable configuration with a number of 9 nodes. The maximum deviation between the edge nodes which was measured over about 10 minutes is $7ms$. In the simulator we had to configure a delay jitter of $7ms$ or in the case of no delay jitter an uncompensated additional transmission delay of $1.5ms$ to get the same result. Therefore, it is highly likely that beside the delay jitter, the testbed environment additionally suffers from a longer communication delay, which was not regarded in the current implementation. Unfortunately, due to the limited number of nodes, we were not able to make further experiments with a bigger group size. Thus we must rely on the simulation experiments which show us that a bigger group size usually results in a better synchronization precision.

5.2.2 Energy Measurements

The energy consumption plays an important role for the device lifetime in battery-powered wireless networks, especially if no infrastructure is available. All RCBs are battery-powered with two $1.5V$ AAA-batteries and thus have a voltage supply of 3 Volt. In order to get the device lifetime, the average power consumption P_{avg} is compared with the electrical energy W_{bat} of the batteries, which we assume to be about $3V \cdot 1200mAh = 3600mWh$. The lifetime in hours is the ratio W_{bat}/P_{avg} and can be reduced to the equivalent formula $t_{life} = E_{bat}/I_{avg}$, where E_{bat} corresponds to the battery charge, denoted in mAh. If so, then the formula determines the device lifetime in hours and I_{avg} defines the average current consumption.

For further energy calculations, the current consumption during a complete period can be classified into four parts. These are the firing time, idle time, execution time, and transmission time.

The *firing time* results from the message staggering delay and corresponds to the interval where the transceiver is enabled and the nodes are allowed to transmit their synchronization messages. In our test application, this interval is also called part 1 of the firing time and equals the duration of $50ms$. The consumed current during this time is about $20mA$. Note that there exists an interval between the end of the firing time and the period end which acts as a safety margin in the case a node starts a transmission exactly at the end of the firing time. If so, the transceiver must be enabled as long as the transmission continuous. This safte margin is further named part 2 of the firing time and consumes a current of $24mA$.

The *idle time* is the part, where the current drops to a minimum. The reason for the small current lies in the fact that the device is dormant, i.e., the transceiver is disabled. With our ZigBee nodes, we measured a current of about $6.2mA$.

The *execution time* is the time where the RCB device executes some code. This is always the case at the end of each period, where the device has to execute the RFA. Other execution tasks must be configured in the RODL file. In the test application used for the energy measurement, the RODL file only contains one execution slot in each period. This task is responsible for data preparation. We measured a current of about $11mA$ for a duration of $1ms$. This energy part mainly depends on the amount of code of the executed tasks.

The *transmission time* corresponds to the time, where the device is transmitting data. Normally, this is always the case when the RCB wants to broadcast its synchronization message during the firing time. Other transmissions during the period must be registered in the RODL file. We further measure the energy consumption of a registered transmission slot, which is used to broadcast test data. We measured a current consumption of about $25mA$ over a time of $4.8ms$. Note that this duration does not equal the real transmission time of about $1ms$. This comes from the fact, that the transceiver requires some time for the startup phase and that the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme at the MAC layer also causes some delay.

Energy consumer	I [mA]	t [s]	Battery discharge per cycle	Energy consumption per cycle
Firing Time, Part1	20.0	0.060	$1.200\ mAs$	$3.600\ mJ$
Firing Time, Part2	24.0	0.013	$0.312\ mAs$	$0.936\ mJ$
Execution Time	11.0	0.001	$0.011\ mAs$	$0.033\ mJ$
Transmission Time	25.0	0.005	$0.125\ mAs$	$0.375\ mJ$
Idle Time	6.2	t_{idle}	$6.2mA \cdot t_{idle}$	$18.6mW \cdot t_{idle}$
$\sum$		T	$1.648\ mAs + 6.2mA \cdot t_{idle}$	$4.944\ mJ + 18.6mW \cdot t_{idle}$

Figure 6. Listing of the major energy consumers and their corresponding battery discharge. The energy calculation assumes a working voltage of 3 Volt.

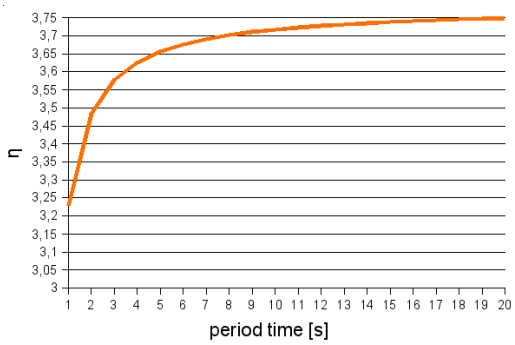


Figure 7. The lifetime improvement η as a function of the period time.

Figure 6 sums up all different energy consumers with the corresponding battery discharge in mAs. In the case the period duration T is exactly one second, the values defined in this table results in a battery discharge per cycle of about $E_{device} = 7.358mAs$. Thus, the average current consumption I_{avg} also equals $7.358mA$. Assuming that our batteries deliver a charge of about $E_{bat} = 1200mAh$, then the resulting lifetime (t_{life}) in hours can be calculated as follows:

$$t_{life} = \frac{E_{bat}}{I_{avg}} = \frac{1200mAh}{7.358mA} = 163h \simeq 1week$$

The configured duty-cycle for this result is about 7 percent, but could be reduced by increasing the period time. If we assume that the time slices of the other consumers are for the most part constant, then a larger period time induces also a larger idle time. Note that a larger period time usually also entails a degradation in precision. The duty-cycle is defined to be the ratio between the sum of the two firing times ($t_{f,1}$,$t_{f,2}$), the execution time (t_e), and the transmission time (t_t) and the complete period time (T). Thus, the duty-cycle is hereinafter denoted by DC and corresponds to the equation $DC = \frac{t_{f,1}+t_{f,2}+t_e+t_t}{T}$.

To follow up on our special energy example, we further want to calculate the improvement of the lifetime with respect to the lifetime as if no synchronization approach would be established, i.e., the duty-cycle equals 100%. In that case, the average current consumption equals $23.752mA$. Consequently, a duty-cycle of 100% corresponds to a lifetime of about $50\frac{1}{2}$ hours. A comparison among the lifetime with a duty-cycle of 100% and the achieved lifetime with our configured duty-cycle of about 7% shows that the synchronization approach improves the lifetime by a at least a factor of three.

To illustrate the dependence between the lifetime improvement and the period time, we introduce the improvement factor, denoted by η. This factor is the ratio between the improved lifetime and the reference lifetime corresponding to a duty-cycle of 100% at the same period time. The improvement factor equals $\eta = \frac{I_{avg}(100\%)}{I_{avg}(DC(T))}$ and is visualized in Figure 7.

6. Discussion

The simulator and testbed results have shown that the simulator provides promising results which are mostly similar to the testbed results. Several experiments have shown that the rate calibration works well in the presence of high delay jitter and transmission delay. However, the results from the multi-hop topology experiments in the testbed system are worse compared to the simulation results. This comes from the fact that our testbed environment suffers from an unexpected delay jitter and further an additional communication delay which was not regarded in the state correction algorithm.

These conditions and the presence of asynchronous communication patterns in realistic sensor networks with high requirements on availability and dependability makes the employment with low-cost nodes in such an environment usually unacceptable. Otherwise, if such a network is only used to gather data where it is not dramatic if some messages are lost, then the use of such a simple synchronization scheme could be a good choice. This distributed synchronization approach and the use of a time-triggered communication provides a communication protocol which supports graceful degradation and further reduces the energy consumption by at least a factor of three. Furthermore, simulation experiments have shown that if the synchronization algorithm is based on high quality crystals with low drift rates, then the precision in multi-hop networks can be lower than $1ms$.

7. Related Work

There exists many work which treats the biologically inspired Firefly synchronization model for realizing the com-

munication in sensor networks. However, we have not found any reports covering such a synchronization approach for establishing a TDMA communication scheme. This may be due to the bounded precision caused by the clock drift of cheap oscillators used in low-cost nodes and that the time to sync is relatively big in contrast to other synchronization schemes.

The most important related work on Firefly synchronicity refers to [10]. Therein, the authors present the Reachback Firefly Algorithm , which is well-suited for the implementation in sensor networks. The algorithm was simulated with TOSSIM in contrast to several parameter choices (e.g., different node topologies, Firing Function Constant (FFC)[2] values, and network diameter).

In [9], the authors introduce a time advance strategy based on the PCO model, which takes the delays in wireless systems into account. Similarly to [10], they incorporate the fact that a node cannot transmit and receive at the same time. The time advance strategy presented in this paper compensates the delay, which is responsible for the lower bound of the accuracy. This delay depends on the dominant transmission and decoding delay. The compensation is done by delaying the transmission of the synchronization messages.

8. Conclusion and Outlook

An alternative synchronization algorithm based on the synchronous flashing of fireflies was introduced in order to establish a global timebase that supports the implementation of a time-triggered approach. This allows a collision-free communication and a reduction of power consumption by at least a factor of 3. The synchronization is based on a self-organized principle with a simple calculation and provides complete scalability and graceful degradation. This is beneficial for the use in sensor networks. Furthermore, the additional rate calibration scheme allows a longer resynchronization interval and the use of cheap oscillators with high drift rates, which are usually featured in low-cost nodes.

The approach has been evaluated by simulation and an implementation in a real testbed environment. Several experiments based on an all-to-all topology have shown that it is possible to achieve a synchronization precision which is lower than $1ms$. Unfortunately, the testbed system suffered from an unexpected delay jitter and an additional communication delay. For this reason, the testbed results considering multi-hop topologies were worse compared to the simulation results with a low delay jitter.

Future work will rely on the reduction of the delay jitter by the use of a different MAC-Stack. Furthermore, we want to compare the results of our approach with the use of different testbed environment and also a different synchronization approach.

[2]The FFC is defined to be the inverse of the coupling strength ϵ.

Acknowledgments

This work was supported by the Austrian FWF project TTCAR under contract No. P18060-N04.

References

[1] Atmel Corporation. *ATAVRRZ200 Demonstration Kit AT86RF230 (2450 MHz band) Radio Transceiver*, July 2006.

[2] J. Buck. Synchronous rythmic flashing of fireflies. *The Quarterly Review of Biology*, 63(3):265–289, Sept. 1988.

[3] W. Elmenreich, G. Bauer, and H. Kopetz. The time-triggered paradigm. In *Proceedings of the Workshop on Time-Triggered and Real-Time Communication*, Manno, Switzerland, Dec. 2003.

[4] H. Kopetz et al. Specification of the TTP/A protocol. Research Report 61/2002, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, Sept. 2002. Version 2.00.

[5] D. Lucarelli and I.-J. Wang. Decentralized synchronization protocols with nearest neighbor communication. *SenSys'04*, pages 62–68, Nov. 2004.

[6] J. Lundelius and N. Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(1):190–204, 1984.

[7] R. E. Mirollo and S. H. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662, Dec. 1990.

[8] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. *Aerospace Conference*, 3:1339 – 1346, Mar. 2003.

[9] A. Tyrrell, G. Auer, and C. Bettstetter. Firefly synchronization in ad hoc networks. In *MiNEMA Workshop*, Feb. 2006.

[10] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *3rd international conference on Embedded networked sensor systems*, pages 142–153, Nov. 2005.

[11] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. In *In IEEE/ACM Transactions on Networking*, volume 12, page 493, June 2004.