Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2009, Article ID 186406, 17 pages
doi:10.1155/2009/186406

Research Article

Firefly Clock Synchronization in an 802.15.4 Wireless Network

Robert Leidenfrost[1] and Wilfried Elmenreich[2]

[1] *Institute of Computer Engineering, Vienna University of Technology, 1040 Vienna, Austria*
[2] *Mobile Systems Group, Institute of Networked and Embedded Systems, University of Klagenfurt, 9020 Klagenfurt, Austria*

Correspondence should be addressed to Robert Leidenfrost, robert.leidenfrost@gmx.at

Received 2 September 2008; Revised 19 February 2009; Accepted 30 March 2009

Recommended by Volker Turau

This paper describes the design and implementation of a distributed self-stabilizing clock synchronization algorithm based on the biological example of Asian Fireflies. Huge swarms of these fireflies use the principle of pulse coupled oscillators in order to synchronously emit light flashes to attract mating partners. When applying this algorithm to real sensor networks, typically, nodes cannot receive messages while transmitting, which prevents the networked nodes from reaching synchronization. In order to counteract this deafness problem, we adopt a variant of the Reachback Firefly Algorithm to distribute the timing of light flashes in a given time window without affecting the quality of the synchronization. A case study implemented on 802.15.4 Zigbee nodes presents the application of this approach for a time-triggered communication scheduling and coordinated duty cycling in order to enhance the battery lifetime of the nodes.

1. Introduction

In South-East Asia, huge swarms of fireflies synchronously emit light flashes to attract mating partners [1]. This paper describes the adaption of the underlying biological principle for a robust self-stabilizing distributed synchronization in wireless sensor networks.

An ensemble of nodes is synchronized in order to execute a collision-free communication schedule following a time-triggered paradigm [2]. The basic element of a time-triggered system is a global timebase that is distributed among the nodes through clock synchronization. In order to provide a common timebase we propose the application of Reachback Firefly Algorithm (RFA), which is a Firefly-inspired algorithm that works despite the limitations of current radio controllers, which are deaf to incoming transmissions while in sending mode. This deafness problem is mitigated by distributing the timing of light flashes in a given time window. Using the global timebase, 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. Furthermore, 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, for example, a sensor network that is utilizing only a fraction of its available bandwidth. 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 consumer power 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.

As a proof of concept, the algorithm has been evaluated by simulation and in a case study consisting of a network of battery-powered low-cost nodes based on an off-the-shelf IEEE 802.15.4 MAC layer. The evaluation results 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

Table 1: A demonstration of the PCO model. The columns correspond to the ongoing time sequence.

c_A	12:00	02:00 → 02:30	12:00	08:00 → 10:00	12:00
c_B	08:00 → 10:00	12:00	09:30 → 11:52	12:00	11:50 → 12:00

scheme. Sections 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 [3] and supports scalability, graceful degradation, and a simple calculation. The algorithm can be classified as a self-stabilizing distributed push-based clock synchronization algorithm. The advantage is that it naturally provides self-stabilization, that is, in any initial configuration, the clocks eventually become synchronized. The concept is based on the Pulse-Coupled Biological Oscillators (PCO) phase advance synchronization model [4], 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.

To understand the principle behind the main concept of the PCO model, consider the following simple example: Assume two persons A and B want to synchronize their wrist watches but can only inform the other one if the own watch indicates twelve o'clock. Let c_A and c_B denote the time of the persons' clocks. Every time a person is notified, it advances the own watch by a factor (in our example 1.25) to at most twelve o'clock. The higher, the multiplication factor, the faster the clocks converge, but the system becomes less robust to faulty notifications then. This algorithm describes the simplified phase advance synchronization model of the fireflies, which is described in more detail in the next section. Based on the initial configuration $c_A = 12{:}00$ and $c_B = 8{:}00$, Table 1 shows that after 5 periods the clocks are synchronized.

However, in the case all clocks are synchronized, they will indicate the clock event at the same time. Using a broadcast communication medium, this causes message collisions, and a "deafness" problem in many wireless systems, since standard wireless transmitters cannot receive messages while being in transmission mode.

The problem can be bypassed by sending the synchronization messages with a random offset, while transmitting the particular offset with the message. The receiver can then reconstruct the intended synchronization instant and perform a clock adjustment with respect to the received offset values. Obviously, this random offset results in an out-of-order reception of synchronization messages which causes a problem in the case of the simple synchronization approach. A solution to this problem is to gather all synchronization events until reaching the period end and then react to the received time information from the last period. This idea was introduced in [3] and is called *reachback response*. However, a reachback response variant of the mentioned simple synchronization approach then equals the below described RFA algorithm with $\alpha = \infty$ and is proven in Lemma 3.7 to be unfeasible for clock synchronization.

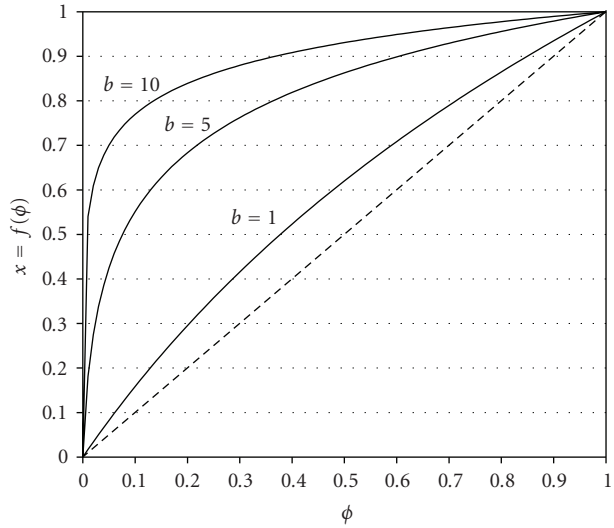


Figure 1: The state function dependent on different dissipation factors.

The formal description is based on the *phase variable* $\phi \in [0,1]$. This variable is characterized by (i) $d\phi/dt = 1/T$, where T denotes the cycle period and (ii) $\phi = 0$ at the beginning of a cycle. Let $x = f(\phi)$ denote the *state variable* corresponding to the charge of a firefly. The authors of the PCO model have proven that the *state function* $f : [0,1] \to [0,1]$ must be a smooth, monotonically increasing, and concave down function in order to achieve synchronicity. In [4], Mirollo and Strogatz have stated a general state function as shown in (1) whereas the form of the curve depends on a parameter named *dissipation factor*, denoted by b, and measures the extent to which $f(\phi)$ is concave down. Figure 1 visualizes the state function for different dissipation factors,

$$f(\phi) = \frac{1}{b} \cdot \ln\left(1 + \left[e^b - 1\right] \cdot \phi\right) \quad \text{with } b > 0. \tag{1}$$

The coupling between the oscillators is defined by the *firing function* (ϕ) and depends on the state function and the *pulse strength* ε, where f^{-1} denotes the inverse state function

$$\phi_{\text{new}} = g(\phi) = \min\left(1, f^{-1}\left(f(\phi) + \varepsilon\right)\right). \tag{2}$$

The firing function is calculated immediately after an oscillator receives a firing event (or flash in case of a firefly). We further use the term of *phase advance* to define the

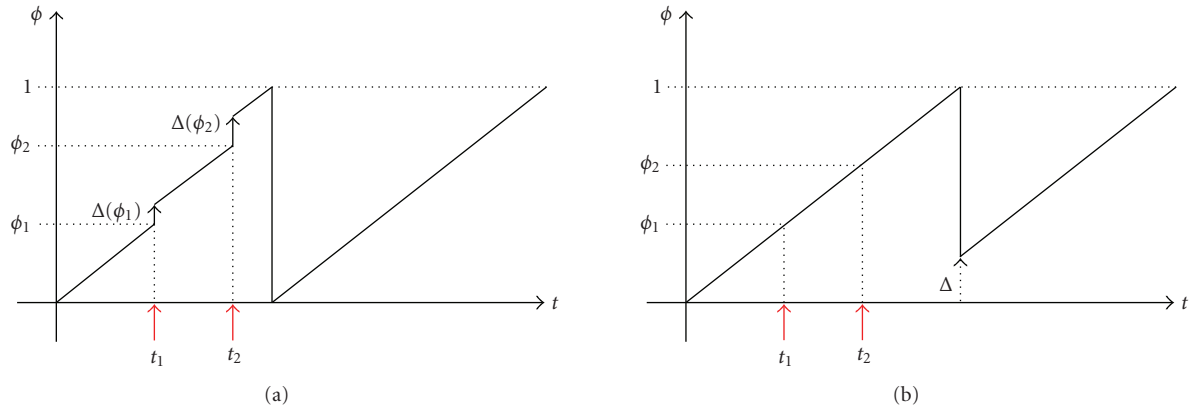


Figure 2: Comparison of (a) the original PCO model and (b) the RFA. In the PCO-model, an oscillator immediately reacts to a firing event. In contrast, The RFA applies the overall phase jump at the beginning of the next cycle: $\Delta = \Delta(\Delta(\phi_1) + \phi_2)$.

increase in the phase domain, denoted by $\Delta(\phi) = g(\phi) - \phi$. Due to the concave down state function, a constant addition in the state-domain results in a variable increase in the phase-domain where a phase advance in the beginning of a cycle is smaller than later in the cycle.

To combat the assumption problems of the PCO model in wireless networks, the RFA additionally uses the notion of a *reachback response* and *pre-emptive message staggering*. *Pre-emptive message staggering* means that a node broadcasts its synchronization message with some random time offset before it reaches the period end and thus is able to gather the time information of all other nodes during a period with a lower probability of message collisions.

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/CA scheme takes effect. The resulting delay jitter then can be avoided by using MAC timestamping. However, the backoff scheme of the IEEE 802.15.4 standard [5] allows to backoff a message at most 5 times which results in a maximum backoff time of at most 36.48 milliseconds (at 2.4 GHz). Since the serialization delay of a full message is at most 4.256 milliseconds (133 bytes at 250 kbps), there can be at most 8 active wireless nodes without losing messages in the best case. Therefore, a bigger network comprising more than 8 nodes in the same broadcast domain requires an additional message staggering delay at an upper layer. A second reason for the additional message staggering is that the original IEEE 802.15.4 standard does not provide an MAC timestamping mechanism and thus does not allow to reduce the delay jitter due to the backoff scheme. The only way to reduce the delay jitter then was to modify the default values of some MAC specific attributes in order to switch off the backoff mechanism. To avoid the resulting higher probability of transmission failures, the *pre-emptive message staggering* explicitly adds a timestamped random transmission delay to the firing messages at the application layer.

3. Applying RFA to Wireless Sensor Networks

The principal purpose of many protocols used in sensor networks is aimed at reducing the consumed power 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. Our concept allows to perform duty-cycling in a more effective way by utilizing a time-triggered approach where a node takes advantage of the a priori known transmission events. These events are globally coordinated by the use of rounds stored in a file called Round Description List (RODL) file. In the current implementation such a round corresponds to a complete 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 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. Figure 3 shows the time diagram of a time-triggered approach for a single node. Therein, a period is subdivided into several slots whereas each slot corresponds to either a *receiving slot*, a *sending slot*, an *execution slot*, or an *idle slot*. Concerning the energy awareness, the most important slots are the receiving slots since they determine how much energy is spent on listening and receiving. In the diagram, the first and the second slot are assigned to be receiving slots. Note that the active time for the receiver unit differs between these slots. This comes from the automatic deactivation after the receiver has recognized the end of a transmission. The parameter w denotes the

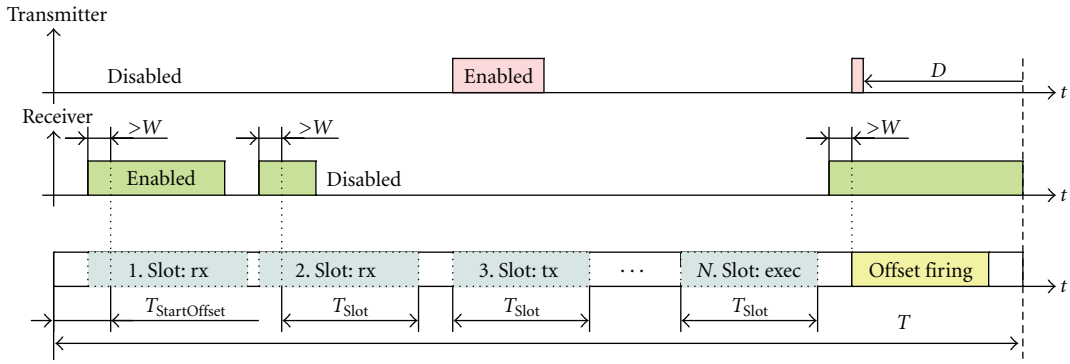


Figure 3: The principle of the time-triggered approach.

synchronization window and guarantees that the receiver module is enabled some time prior before any transmission takes place.

The time-triggered approach requires the notion of a *global time* which is provided by the RFA clock synchronization algorithm. Note that the algorithm can only approximate the global time. The best achievable precision Π in an ensemble of clocks is lower bounded to the *convergence function* Φ of the synchronization algorithm and the maximum *drift offset* $\Gamma = 2 \cdot \rho \cdot T$, where ρ denotes the maximum drift rate of all clocks in that ensemble. This is also known as the *synchronization condition* $\Pi \geq \Phi + \Gamma$. In our approach, the convergence function is defined by the RFA and heavily depends on the maximum delay jitter ϵ which is the maximum absolute deviation of the delay a message encounters during the communication.

In order to get promising results, the global time must be approximated with a very high precision. One way is 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. Secondly, a shorter period time results in the exchange of more synchronization messages in the same time and thus would affect the energy consumption. Alternatively, 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 representing a macrotick in order to adjust the granularity and 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, monotonically increasing, and concave down state function of (1) to calculate the overall phase advance Δ. Consider the dissipation factor $b > 1$ and the pulse strength within $0 < \varepsilon < 1$, then the phase advance equals

$$\Delta(\phi) = \min\left(1, f^{-1}\left(f(\phi) + \varepsilon\right)\right) - \phi. \tag{3}$$

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 $f^{-1}(x) = (e^{bx} - 1)/(e^b - 1)$ in (3). Let $\alpha = e^{\varepsilon b}$ and $\beta = (\alpha - 1)/(e^b - 1)$, then (3) can be transformed to

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

Assuming a strong dissipation factor $b \gg 1$ and a small pulse strength s.t. $0 < \varepsilon = b^{-1}$, then we can replace $e^{\varepsilon b}$ by the first-order approximation of the Taylor expansion $1 + \varepsilon b$ and thus β is negligible. The phase advance then can be reduced to

$$\Delta(\phi) = \min\left(1, \alpha \cdot \phi\right) - \phi. \tag{5}$$

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 [3].

In contrast to the original RFA algorithm, our approach achieves a better synchronization precision and a faster convergence time by indirectly performing a clustering of the received synchronization events. This is done by ignoring all events which are within the phase advance of the last event to which a node reacted. In fact, this corresponds to the introduction of a short refractory period. Additionally, we do not allow a node to react to firing events which would originally occur after the node reached the period end. This ensures that in the case of synchronized nodes, the fastest node then does not advance its phase anymore resulting in a better precision. The algorithm is formally analyzed in more detail and guarantees network synchronization as long as the bounds for several parameters are maintained. Algorithm 1 explains the behavior of this extended RFA (E-RFA) algorithm with the use of pseudocode. The refractory period is implemented by the condition in Line 9. The variable *eventset* contains the correct phase of all received firing messages and Φ_{rnd} denotes the random amount for the

```
   Initially eventset := ∅, Δ_i := 0, φ_i := 0, Φ_i := Φ_th − (Φ_msd)
(1)upon event φ_i(t) = Φ_th − Φ_rnd do            // preopened transmission of sync-message
(2)   trigger send_i(φ_i(t))                      // broadcast current phase to all neighbors
(3)upon event recv_i(φ_j) from p_j do             // received sync-message
(4)   if φ_i(t) − φ_j < 0 then                    // check if real firing-event is within the period
(5)      add ⟨φ_i(t) + Φ_th − φ_j⟩ to eventset
(6)upon event φ_i(t) = Φ_th do                    // threshold reached
(7)   φ_last := δ_last := Δ_i := 0                // clean up
(8)   for each event φ_j ∈ eventset in increasing order do
(9)      if Δ_i + φ_j < Φ_th and φ_last + δ_last < φ_j then   // check time consistency
(10)        δ_last := min(Φ_th, (φ_j + Δ_i) · α) − (φ_j + Δ_i)   // calculate phase advance
(11)        Δ_i := Δ_i + δ_last
(12)        φ_last := φ_j
(13)  φ_i(t) := Δ_i                               // Apply recheckback response
(14)  Φ_rnd := (Φ_msd^max − Φ_msd^min) + Φ_msd^min   // Calculate firing offset
(15)  eventset := ∅
```

Algorithm 1: E-RFA: code for $\mathbf{p_i}$, $\mathbf{0 \le i \le n-1}$.

preponed transmission with at most the maximum, respectively, minimum message staggering delay Φ_{msd}^{max}, respectively, Φ_{msd}^{min}.

Since the purpose of this work should demonstrate that such a synchronization approach works with an off-the-shelf communication stack without MAC-timestamping, we have to expect a delay jitter in the order of milliseconds due to the uncertainty in the application and MAC layer. It should be mentioned that Lundelius and Lynch have shown in [6] that in the presence of a *maximum delay jitter* ϵ, an assemble of N clocks cannot be synchronized to a precision better than $\epsilon \cdot (1 - 1/N)$.

Lower Bound for the Coupling Factor α. We assume that every processor p_i consists of a hardware clock $HC_i(t)$ which generates the phase $\varphi_i(t)$. This clock stays within a *linear envelope* of the real time. Note that whereas the hardware clock continuously increases, the phase is periodically reset to 0 with respect to $\varphi_i(t) \equiv HC_i(t) + \text{offs}_i(t) (\text{mod} \Phi_{th})$ where $\text{offs}_i(t)$ denotes a dynamic offset value which changes due to the state correction algorithm. Φ_{th} represents the granularity of the hardware clock which corresponds to the synchronization period T. We therefore assume that there exists a positive constant ρ (*maximum drift rate*) such that $(\Phi_{th}/T)/(1+\rho) \le d\varphi_i(t)/dt \le (\Phi_{th}/T)/(1-\rho)$. Note that this definition of the bounded drift simplifies the calculation of the precision and may differ from literature. We further assume a fully connected network in which the message delay d is always in the range $d \in [\varsigma, \varsigma + \epsilon]$, where ς denotes the constant part and ϵ the *maximum delay jitter* of the communication delay in real time. The lower bound for α in an ensemble of nodes in a fully connected network then depends on the maximum drift rate ρ, the message staggering delay Φ_{msd}, and the communication delay d. Note that all parameters with a preceding Φ are defined with respect to Φ_{th}. However, for simplifaction we now always assume that Φ_{th} is normalized to 1. Let r_{max}, respectively, r_{min} denote the maximum, respectively, minimum relative message staggering delay $r_{max} = \Phi_{msd}^{max}/\Phi_{th}$ and $r_{min} = \Phi_{msd}^{min}/\Phi_{th}$. We now show that in the case of two clocks, the modified RFA provides a bounded precision Π. Therefore, $\Pi(\varphi) = \max(|rt_{\varphi_j=\varphi} - rt_{\varphi_i=\varphi}|)$ for $\varphi \in [0,1)$ denotes the maximum time difference among all nodes $p_i \neq p_j$ in real time units between the time p_i reached φ and the time p_j reached φ.

Lemma 3.1. *Let* $R = (1+\rho)/(1-\rho)$ *and* $\Gamma = 2\rho T$ *be the drift offset. In the case of two clocks and no message loss, if the coupling factor* α *is lower bounded to* $\alpha > (1 - r_{max} \cdot (R-1) - (\Pi^U - \varsigma)/(T \cdot (1-\rho)))^{-1}$ *and* $\rho < 1/7$, *then for* $\varphi \in [0,1)$ *and* $(\Pi^U + \varsigma + \epsilon)/(T \cdot (1-\rho)) < r_{min} \le r_{max} < 1/2$, *Algorithm 1 keeps the network synchronized with a worst case precision bounded to*

$$|\Pi(\varphi)| \le \Pi^U = (1 + r_{max})\Gamma + \epsilon R + \max(\Gamma r_{max}, \varsigma R). \quad (6)$$

Proof. Assume the clocks are initially synchronized to $-\Pi^U \le \Pi \le \Pi^U$. W.l.o.g. let p_i be the faster node. We further use p_i as the reference for the precision $\Pi(\varphi) = rt_{\varphi_j=\varphi} - rt_{\varphi_i=\varphi}$, where $rt_{\varphi_j=\varphi}$ denotes the real time when p_j's phase φ_j reached φ. We further assume that the next time p_i reaches the threshold 1 is at time $t = 0$. Let $\Pi_0 = \Pi(0)$ be the corresponding precision at $t = 0$. For $t \le 0$ we then have $\varphi_i(t) = 1 + t/(T \cdot (1-\rho))$ and $\varphi_j(t) = 1 + (t - \Pi_0)/(T \cdot (1+\rho))$. Let r_i, respectively, r_j denote the relative message staggering delay the node p_i, respectively, p_j has calculated for the last transmission. If the last fire event of p_i was at $\varphi_i = 1 - r_i$, then with respect to the communication delay d, p_j received the phase at $\varphi_j^{recv} = 1 + (-r_i \cdot T \cdot (1-\rho) + d - \Pi_0)/(T \cdot (1+\rho))$ and consequently adds the offset r_i leading to $\varphi_j^{fire} = 1 + r_i \cdot (1 - 1/R) + (d - \Pi_0)/(T \cdot (1+\rho))$. Similarly, a fire event from p_j with offset r_j is received by p_i at phase $\varphi_i^{fire} = 1 - r_j \cdot (R-1) + (d + \Pi_0)/(T \cdot (1-\rho))$. Let $\varphi_{j,min}^{fire}$, $\varphi_{j,max}^{fire}$, $\varphi_{i,min}^{fire}$, $\varphi_{i,max}^{fire}$ be the minimum, respectively, maximum possible phases of the calculated firing events. If $\alpha > \max(1/\varphi_{i,min}^{fire}, 1/\varphi_{j,min}^{fire})$, then it is guaranteed that $\Delta(\varphi_i^{fire}) = 1 - \varphi_i^{fire}$, respectively, $\Delta(\varphi_j^{fire}) = 1 - \varphi_j^{fire}$. Since $\varphi_{i,min}^{fire} < \varphi_{j,min}^{fire}$, we have $\alpha > 1/\varphi_{i,min}^{fire}$ as stated.

Based on the current precision Π_0 and the phase advance of p_i and p_j at time $t = 0$ labeled by $\Delta_i = \Delta(\varphi_i^{\text{fire}})$ and $\Delta_j = \Delta(\varphi_j^{\text{fire}})$, we are able to calculate the precision Π_{next} the next time p_i reaches the threshold. That is, $\Pi_{\text{next}} = \Pi_0 + \Gamma + T \cdot (\Delta_i \cdot (1-\rho) - \Delta_j \cdot (1+\rho))$. However, we have to distinguish between three cases depending on Π_0. In detail, if $\Pi_0 \in [0, \Pi^U]$, then (1) $\Delta_i = 0$ and $\Delta_j > 0$, or if $\Pi_0 \in [0, \Gamma)$, then also (2) $\Delta_i > 0$ and $\Delta_j > 0$, or finally if $\Pi_0 \in [-\Pi^U, 0]$, then due to Line 4 of Algorithm 1 we have (3) $\Delta_i > 0$ and $\Delta_j = 0$. Note that the overlapping of (1) and (2) is volitional, because if $\Pi_0 \in [0, \Pi^U]$, then both cases can occur and hence must be considered. Further note that the bound of Γ ensures that the interception point of the phase of both nodes is within the last period. In order to keep the clocks within the precision, the inequality $-\Pi^U \leq \Pi_{\text{next}} \leq \Pi^U$ must be valid for all three cases. From the first case we get $\Pi^U \geq (1 + r_{\max})\Gamma + \varsigma + \epsilon$ and $r_{\min} \geq -1 - (\Pi^U + \varsigma)/\Gamma$. From the third case it follows $\Pi^U \geq (1 + r_{\max})\Gamma - \varsigma$ and $r_{\min} \geq -1 - (\Pi^U - \varsigma - \epsilon)/\Gamma$. Note that $r_{\min}$ is always valid due to the definition of Π^U. From the second case, it can be derived that $\Pi^U \geq (1 + 2r_{\max})\Gamma + \epsilon$ and $r_{\min} \geq (\epsilon - \Pi^U)/2\Gamma$. Again, Π^U ensures that $r_{\min}$ is valid. The worst case precision with respect to these three cases then equals $\Pi^U = (1 + r_{\max})\Gamma + \epsilon + \max(\Gamma r_{\max}, \varsigma)$.

Note that the correctness of the proof requires that a node advances its phase at most once per period. However, if $\Pi_0 > 0$, then p_j may initiate a firing event after p_i already passed the threshold. Simply setting $r_{\min} \geq (\Pi^U + \varsigma + \epsilon)/(T \cdot (1-\rho))$ avoids this effect.

In order to get the worst case precision, we further have to incorporate the precision (I) $\Pi(\Delta_i)$ and (II) $\Pi(\Delta_j)$ for all three mentioned cases. In detail, for $\Pi_0 \in [0, \Pi^U]$ we additionally have to analyze for case (1) if the equation $-\Pi^U \leq \Pi_0 - \Delta_j \cdot (1-\rho) \cdot T \leq \Pi^U$ holds and for case (2), if $-\Pi^U \leq (\Delta_i - \Delta_j) \cdot (1+\rho) \cdot T + \Pi \leq \Pi^U$ and $-\Pi^U \leq \Pi - (\Delta_j - \Delta_i) \cdot (1-\rho) \cdot T \leq \Pi^U$ are valid. Similarly for $\Pi_0 \in [-\Pi^U, 0]$ it must be ensured that $-\Pi^U \leq \Delta_i \cdot (1+\rho) \cdot T + \Pi \leq \Pi^U$. From these equations we can derive the following additional bounds: $(R-2)/(4-3R) \leq r_{\max} \leq 1/(R-1)$, and $\Pi^U \geq (\varsigma + \epsilon)R - \Gamma R r_{\min}$. Therefore, if we want $r_{\max}$ bounded between $[0, 1]$, then $\rho < 1/7$ must hold. Furthermore, in the case of $\rho = 0$, we have to adapt the worst case precision to $\Pi^U = (1 + r_{\max})\Gamma + \epsilon R + \max(\Gamma r_{\max}, \varsigma R)$ which now equals the worst case upper bound, since all possible cases were considered.

Finally, it should be mentioned that the maximum relative message staggering delay $r_{\max}$ must be smaller than 1/2. Otherwise, assume the case where both nodes are initially 1/2 apart. Then both nodes will never perform a phase advance due to Line 4 of the algorithm. □

Note that in the case of a fully connected network comprising more than two nodes, all nodes synchronize to the fastest one due to Line 4 and Line 9 of Algorithm 1. Especially the condition $\varphi_{\text{last}} + \delta_{\text{last}} < \varphi_j$ in Line 9 ensures if a node advances its phase due to some received firing event φ_j, then all events immediately following some short time after φ_j are ignored. This condition is necessary. Otherwise, assume n nodes are perfectly synchronized. Consequently, a node would perform n times a phase advance, which results in a mutual excitation in the case n is very large.

Theorem 3.2. *Let $R = (1+\rho)/(1-\rho)$ and $\Gamma = 2\rho T$ be the drift offset. In the case of $n \geq 2$ clocks and no message loss, if the coupling factor α is lower bounded to $\alpha > (1 - r_{\max} \cdot (R-1) - (\Pi^U - \varsigma)/(T \cdot (1-\rho)))^{-1}$ and $\rho < 1/7$, then for $\varphi \in [0, 1)$ and $(\Pi^U + \varsigma + \epsilon)/(T \cdot (1-\rho)) < r_{\min} \leq r_{\max} < 1/2$, Algorithm 1 keeps the network synchronized with a worst case precision bounded to*

$$|\Pi(\varphi)| \leq \Pi^U = (1 + r_{\max})\Gamma + \epsilon R + \max(\Gamma r_{\max}, \varsigma R). \quad (7)$$

Corollary 3.3. *If a fully connected network comprises only of perfect clocks ($\rho = 0$) and the communication network suffers from no delay jitter ($\varsigma = \epsilon = 0$), then the network keeps synchronized with a precision of $\Pi = 0$, if $\alpha > 1$.*

Note that Corollary 3.3 states that it is sufficient that the network is connected.

Corollary 3.4. *If a fully connected network comprises only of perfect clocks ($\rho = 0$) and the communication network suffers only from delay jitter ($\epsilon > 0$, $\varsigma = 0$), then the network keeps synchronized with a precision of $|\Pi| \leq \epsilon$, if $\alpha > T/(T - \epsilon)$.*

Corollary 3.5. *If a fully connected network comprises of clocks with a maximum drift rate of $\rho < 1/7$ and the network suffers from no communication delay ($\varsigma = \epsilon = 0$) and $r_{\max} = 1/2$, then the network keeps synchronized with a precision of $|\Pi| \leq 2\Gamma$, if $\alpha > (1-\rho)/(1-6\rho)$.*

Upper Bound for the Coupling Factor α. One may ask why not setting $\alpha = \infty$ such that a node immediately adjusts its phase to a neighboring clock every time receiving a firing message from this clock. However, the following lemmata shows that there exists a basic upper bound which holds for every network.

Definition 3.6. A firing configuration $C(N, k, m) = (\varphi_{0,m}; \varphi_{1,m} \cdots \Delta_{k,m} \cdots \varphi_{n-1,m})$ of a fully connected network N comprising n nodes is defined to be the concatenation of the phase $\varphi_{i,m} = \varphi_i(t)$ of node p_i at the time t when p_k just reached the threshold for the mth time and consequently applied the phase advance $\Delta_{k,m}$.

Lemma 3.7. *In a fully connected network N comprising of $n = 2$ perfect clocks, if the coupling factor $\alpha \geq 3/2$, then the nodes may never become synchronized.*

Proof. The proof is based on the fact that if α is too large, then the nodes will infinitely often enter the same firing configuration. Let p_A and p_B be the two participating processors where p_A is the first node reaching the threshold. The initial firing configuration then is $C(N, A, 1) = (\Delta_{A,1}, \varphi_{B,1})$ with $\Delta_{A,1} < \varphi_{B,1}$. Next, p_B reaches the threshold leading to $C(N, B, 1) = (\varphi_{A,1}; \Delta_{B,1})$ with $\varphi_{A,1} = \Delta_{A,1} + 1 - \varphi_{B,1}$ and $\Delta_{B,1} = \Delta(\varphi_{B,1})$. The next time p_A reaches the threshold is at $C(N, A, 2) = (\Delta_{A,2}; \varphi_{B,2})$ with $\Delta_{A,2} = \Delta(\Delta_{A,1} + 1 - \varphi_{B,1})$ and $\varphi_{B,2} = \Delta(\varphi_{B,1}) + \varphi_{B,1} - \Delta_{A,1}$. Finally p_B again reaches the

threshold at $C(N,B,2) = (\varphi_{A,2}; \Delta_{B,2})$ with $\varphi_{A,2} = \Delta(\Delta_{A,1} + 1 - \varphi_{B,1}) + 1 - \Delta(\varphi_{B,1}) - \varphi_{B,1} + \Delta_{A,1}$ and $\Delta_{B,2} = \Delta(\Delta(\varphi_{B,1}) + \varphi_{B,1} - \Delta_{A,1})$.

If we assume that (1) $\alpha \cdot (\varphi_{B,1}) \geq 1$, then the phase advance can be reduced to $\Delta(\varphi_{B,1}) = 1 - \varphi_{B,1}$. The same applies to (2) $\alpha \cdot (\Delta_{A,1} + 1 - \varphi_{B,1}) \geq 1$ and (3) $\alpha \cdot (1 - \Delta_{A,1}) \geq 1$. Thus, if all three conditions are true, $C(N,B,2)$ can be redefined to $C(N,B,2) = (\varphi_{B,1}, \Delta_{A,0})$. In other words, the nodes will infinitely often enter the initial firing configuration. We now have to find the lowest α where the inequation $\alpha \geq \max(1/\varphi_{B,1}, 1/(1 - \Delta_{A,1}), 1/(1 + \Delta_{A,1} - \varphi_{B,1}))$ is valid. Equalizing all three conditions yields $\Delta_{A,1} = 1/3$ and $\varphi_{B,1} = 2/3$. Thus we get $\alpha \geq \max(3/2, 3/2, 3/2) = 3/2$. □

Since the algorithm ignores all firing events immediately following some short time after a previous firing event due to Line 9, a node may realize a set of nodes as a single node and therefore Lemma 3.7 also applies to networks comprising more than two nodes. We now exploit the intuition behind Lemma 3.7 and extend this problem to a general network comprising $n \geq 2$ nodes.

Definition 3.8. $C(N,k,m)$ is called to be an infeasible firing configuration, if there exists a positive integer $i > 0$ such that $C(N,k,m) = C(N,k,m+i)$ and the network is not synchronized.

Lemma 3.9. *The maximum phase advance a node can perform in a fully connected network N comprising n nodes equals* $\Delta = ((2\alpha - 1)^{n-1} - 1)/((2\alpha - 1)^{n-1} + 1)$.

Proof. The maximum phase advance occurs if the firing events are at close quarters such that no event is ignored due to Line 9 of Algorithm 1. In detail, assume a node received the firing event at the phases $\varphi_0 < \varphi_1 < \cdots < \varphi_{n-1} = 1$. The first phase advance then equals $\lambda_0 = \varphi_0 \cdot \gamma$, where $\gamma = \alpha - 1$. Due to Line 9 of Algorithm 1, the earliest next time the node performs a phase advance can only be at $\varphi_1 = \varphi_0 + \lambda_0$ and equals $\lambda_1 = (\varphi_1 + \lambda_0)\gamma$. Generally, $\varphi_{k+1} = \varphi_k + \lambda_k$ and $\lambda_{k+1} = (\varphi_{k+1} + \sum_{i=1}^{k} \lambda_i)\gamma$ for $0 \leq k < n - 1$. Solving the recursion leads to $\varphi_k = \varphi_0 + \sum_{i=0}^{k-1} \lambda_i$ and thus $\lambda_{k+1} = (\varphi_0 + 2\sum_{i=0}^{k} \lambda_i)\gamma$. Solving the equation for $\lambda_{k+1} - \lambda_k$ then yields $\lambda_k = (1 + 2\gamma)^k \gamma \varphi_0$. The overall phase advance thus equals $\Delta = \sum_{i=0}^{n-2} \lambda_i = \gamma \sum_{i=0}^{n-2} (1 + 2\gamma)^i = ((1 + 2\gamma)^{n-1} - 1)\varphi_0/2$. Since the maximum Δ occurs when $\varphi_{n-1} = 1$, we finally get $\Delta = ((1 + 2\gamma)^{n-1} - 1)/((1 + 2\gamma)^{n-1} + 1)$. □

A weak upper bound results from the fact that we do not want a node to perform a phase advance which is greater than 1/2 and directly follows from Lemma 3.9.

Corollary 3.10. *In a fully connected network comprising of* $n \geq 2$ *perfect clocks, if the coupling factor* $\alpha < (\sqrt[n-1]{3} - 1)/2$, *then in every feasible execution a node will never perform a phase advance which is greater than* 1/2.

Note that even if the weak bound is maintained, it can be shown that there exist infeasible firing configurations. However, due to imprecisions in calculations, the varying short-term drift, the delay jitter, and due to several other indeterministic environmental effects, this bound is generally applicable. A stronger bound results from empirical studies which have shown that infeasible firing configurations do not exist, if the maximum phase advance $\Delta_{\max} < 1/(n+1)$. The resulting bound for α again can be deduced from Lemma 3.9.

Theorem 3.11. *In a fully connected network comprising of* $n \geq 2$ *perfect clocks, if the coupling factor* $\alpha < (1/2)(1 + \sqrt[n-1]{1 + 2/n})$, *then the nodes will never enter an infeasible firing configuration.*

Rate of Synchronization. Theorem 3.15 analyzes the time to sync for the case of two oscillators. The authors of [4] have also analyzed the case of n oscillators. However, considering a multihop topology requires a more sophisticated solution. For the following proofs, let $\gamma = \alpha - 1$ and $\Phi_0 = \varphi_A - \varphi_B$ denote the initial phase difference between the clocks A and B with $0 \leq \varphi_B \leq \varphi_A \leq 1$ in network N.

Lemma 3.12. *The infeasible firing configuration* $C(N,A) = (\Delta_A^*; \varphi_B^*)$ *with* $\Delta_A^* = (\alpha - 1)/(3 - \alpha)$ *and* $\varphi_B^* = 1/(3 - \alpha)$ *is a unique fixpoint and has a phase difference of* $\delta^* = (2 - \alpha)/(3 - \alpha)$.

Proof. If we set $C(N,A,k+1) = C(N,A,k)$, we get $\Delta_{A,k+1} = \Delta_{A,k} = (\Delta_{A,k} + 1 - \varphi_{B,k}) \cdot (\alpha - 1)$ and $\varphi_{B,k+1} = \varphi_{B,k} = \varphi_{B,k} \cdot \alpha - \Delta_{A,k}$ and thus $\Delta_A^* = (\alpha - 1)/(3 - \alpha)$ and $\varphi_B^* = 1/(3 - \alpha)$. □

Although this fixpoint is a repeller, the roundoff error in the calculation may cause a node to enter the fixpoint. This is especially a concern if the granularity of the hardware clock is very low. The rate of sync with respect to different initial phase differences is visualized in Figure 4. It is obvious that there exists a special initial configuration Φ_0^* which causes the network to enter this fixpoint. To analyze this initial configuration, we first transform the recursion of the dynamic system into a closed term.

Lemma 3.13. *The phase difference* δ_k *of* $C(N,A,k)$ *for* $k \geq 1$ *equals*

$$\delta_k = \delta^* + \frac{\left((A_2 \cdot \gamma(1-\gamma) - A_1) \cdot z_1^{-k} + (B_2 \cdot \gamma(1-\gamma) - B_1) \cdot z_2^{-k}\right)}{\gamma^2}, \tag{8}$$

where $z_1 = (1 + 2\gamma + \sqrt{1 + 4\gamma})/2\gamma^2$, $z_2 = (1 + 2\gamma - \sqrt{1 + 4\gamma})/2\gamma^2$, $A_1 = (1 - \Phi_0 - z_1\gamma)/(z_1 - z_2)$, $B_1 = (z_2\gamma - 1 + \Phi_0)/(z_1 - z_2)$, $A_2 = z_1^2/((1 - z_1) \cdot (z_1 - z_2))$, $B_2 = -z_2^2/((1 - z_2) \cdot (z_1 - z_2))$, *and* δ^* *from Lemma 3.12.*

Proof. Let $C(N,A,1) = (\Delta_{A,1}, \varphi_{B,1})$ be the initial firing configuration with $\Delta_{A,1} < \varphi_{B,1}$ where $\Delta_{A,1} = 0$ and $\varphi_{B,1} = 1 - \Phi_0$. The phase difference when p_A reached the threshold for the kth time is $\delta_k = \varphi_{B,k} - \Delta_{A,k}$. From Lemma 3.7 we know that $C(N,A,k+1) = (\Delta_{A,k+1}; \varphi_{B,k+1})$ with $\Delta_{A,k+1} = (\Delta_{A,k} + 1 - \varphi_{B,k}) \cdot (\alpha - 1)$ and $\varphi_{B,k+1} = \varphi_{B,k} \cdot \alpha - \Delta_{A,k}$. If we substitute γ for $\alpha - 1$ and consider the phase difference δ_k of

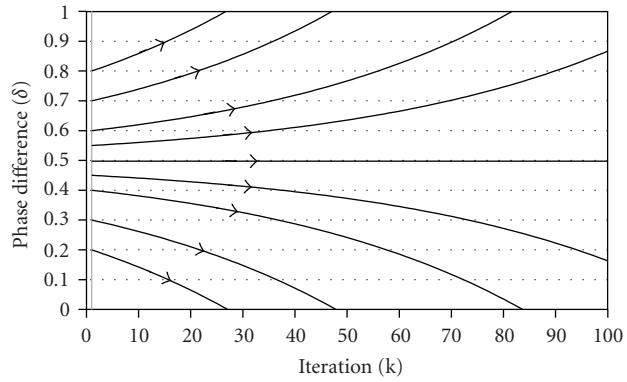


Figure 4: The rate of sync for different initial configurations with $\alpha = 1.01$.

$C(N, A, k)$, we get $\Delta_{A,k+1} = \gamma(1-\delta_k)$ and $\varphi_{B,k+1} = \delta_k + \gamma \cdot \varphi_{B,k}$ which yields $\delta_{k+2} = \delta_{k+1} \cdot (1+2\gamma) - \delta_k \cdot \gamma^2 - \gamma(1-\gamma)$ for $k \geq 1$. The dissolving of the recursion is left to the reader and leads to the solution as stated. □

Lemma 3.14. *There exists a unique initial phase difference* $\Phi_0^* \in (0,1)$ *where the network eventually enters the fixpoint of Lemma 3.12 and equals* $\Phi_0^* = 1 - z_2\gamma \cdot (1 - z_2\gamma)/(1 - z_2)$ *with* z_2 *from Lemma 3.13.*

Proof. If the network enters the fixpoint in $C(N, A, m)$ at some $m > 1$, then we have a phase difference of $\delta_k = \delta^*$ for $k \geq m$ with δ^* from Lemma 3.12. Using (8) then yields $(z_2/z_1)^{k+1} = -(B_2 \cdot \gamma(1-\gamma) - B_1)/(A_2 \cdot \gamma(1-\gamma) - A_1)$. Since $z_1 > z_2$ we get $\lim_{k\to\infty}(z_2/z_1)^{k+1} = 0$ and thus $B_2 \cdot \gamma(1-\gamma) = B_1$. Using B_1 and B_2 from Lemma 3.13 results in $\delta_1 = z_2\gamma \cdot ((1 - z_2\gamma)/(1 - z_2))$. The initial phase difference then has to be $\Phi_0^* = 1 - \delta_1$ as stated. □

Theorem 3.15. *The number of iterations* k *until synchrony is at most* $k \leq \log_{z_2}((B_2 \cdot \gamma(1-\gamma) - B_1)/((\delta^* - \delta) \cdot \gamma^2))$ *with* B_1, B_2, *and* z_2 *from Lemma 3.13 and*

$$\delta = \begin{cases} 1, & \text{if } \Phi_0 \leq \Phi_0^*, \\ 0, & \text{if } \Phi_0 > \Phi_0^*. \end{cases} \tag{9}$$

Proof. Note that $\lim_{k\to\infty} C(N, A, k) = (\Delta_{A,k}, \varphi_{B,k})$ either converges to $(0,0)$ or $(1,1)$ as visualized in Figure 4. Therefore, we simply equate (8) with 1 if $\Phi_0 \leq \Phi_0^*$ or with 0 if $\Phi_0 < \Phi_0^*$. Since $z_1 > 7$ for $\alpha < 3/2$ and the multiplicative factor is smaller than 1, the term with respect to z_1^{-k-1} does not influence the rate of sync for larger k and hence can be neglected. This leads to the equation as stated. □

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. Note that the rate correction can be performed completely independent from the clock state correction scheme.

The core concept of our rate calibration algorithm is that a processor p_j implements a virtual clock VC_j which abstracts the hardware clock HC_j. The algorithm implemented on p_j then only reads the time from the VC_j. We further denote the ticks from the HC_j by *microticks* and that from the VC_j by *ticks*. One tick of VC_j comprises several microticks which we denote by T_j^{th}. By adjusting T_j^{th}, the time duration of one tick can be increased or decreased. Let T_{nom} be the nominal threshold level and H_j the *absolute adjustment value* s.t. $T_j^{th} = T_{\text{nom}} + H_j$. Note that the corresponding *relative adjustment value* for p_j equals $h_j = H_j/T_{\text{nom}}$. In order to perform the rate calibration, every processor p_j periodically broadcasts a synchronization messages m_j. Let $VC_j(m_j)$, respectively, $HC_j(m_j)$ denote the timestamps of p_j when p_j transmitted m_j and $VC_r(m_j)$, respectively, $HC_r(m_j)$ the timestamps of p_r when p_r received m_j from p_j. Let $m_{j,k}^r$ be the kth message p_r received from p_j and $m_{j,k}$ the kth message p_j broadcasted. We further assume that $m_{j,k+1}$ is not received at some p_r before $m_{j,k}$ is received for $k \geq 1$. The dependency between the virtual and the hardware clock with respect to some message m_j is characterized as $VC_j(m_{j,k}) \equiv (HC_j(m_{j,k})/(1+h_j))(\text{mod } \Phi_{th})$. Note that we assume that the hardware clock is a linear function of real time within a sufficient long period of time. In contrast, the virtual clock is periodically reset with respect to the resynchronization interval. This assumption is required in order to realize a pulse synchronization scheme.

The rate correction algorithm works as follows: based on the timestamp $HC_j(m_{j,k}^r)$ stored in $m_{j,k}^r$, the receiving processor p_r can calculate p_j's relative adjustment value in its own granularity, denoted by $h_{j,k+1}^r$, with $h_{j,k+1}^r = (HC_r(m_{j,k+1}^r) - HC_r(m_{j,k}^r))/((HC_j(m_{j,k+1}) - HC_j(m_{j,k}))/(1+h_{j,N})) - 1$. Therein, the term $h_{j,N}$ denotes the latest received adjustment value from p_j, that is, the relative adjustment value contained in the latest received message $m_{j,N}$ from p_j. In order to reduce the impact of the delay jitter, we should choose the time interval between the two received messages as large as possible. Note that there still exists an upper limit due to the long-term stability of an oscillator, which is usually in the order of minutes. However, the optimal time interval also depends on the underlying oscillator type. In our case, we store the last N received messages from each node and calculate the relative deviation with respect to the buffer size N. To visualize the impact of the delay jitter, we replace $HC_r(m_{j,k}^r)$ by $HC_r(m_{j,k}^r) + d_k$, where d_k corresponds to the delay of message $m_{j,k}^r$ in p_r's clock granularity. From this it follows:

$$h_j^r = \frac{HC_r\left(m_{j,N}^r\right) - HC_r\left(m_{j,1}^r\right)}{\left(HC_j\left(m_{j,N}^r\right) - HC_j\left(m_{j,1}^r\right)\right)/\left(1+h_{j,N}\right)} - 1 + \frac{(d_N - d_1) \cdot \left(1+h_{j,N}\right)}{HC_j\left(m_{j,N}^r\right) - HC_j\left(m_{j,1}^r\right)}. \tag{10}$$

In our implementation we set $N = 8$ reducing the impact of the jitter with respect to the resynchronization period to at about $\sim \epsilon/8$. The next step of the algorithm is to calculate

the average relative phase adjustment value of all processors $p_j \in P$ in the broadcast domain where $p_r \notin P$, that is,

$$\overline{h}^r = \frac{h_r + \sum_{p_j \in P} \overline{h}^r_j}{|P| + 1}. \tag{11}$$

The main challenge, however, concerns the adjustments of $\overline{h}^r$ to the new relative adjustment value h^r_{new} of p_r. Due to natural imprecisions and the delay jitter, a direct adjustment of $h^r_{\text{new}} = \overline{h}^r$ is inappropriate resulting in a continuous increase or decrease of the real overall average relative adjustment value. This effect is also known as the *common-mode drift*. A better approach is to implement a parametrized adjustment by the use of a smoothing factor σ as shown in (12) which ensures that the virtual clocks smoothly converge to the overall average interval,

$$h^r_{\text{new}} = h^r_{\text{old}} + \left(\overline{h}^r - h^r_{\text{old}}\right) \cdot \sigma \tag{12}$$

In our implementation we have chosen $\sigma = 1/2$. However, it seems that the common-mode drift cannot be avoided but the effect can be minimized by carefully choosing N with respect to the delay jitter. For this reason we have introduced bounds for the relative adjustment value, that is, $h^r \in [-2 \cdot \rho, 2 \cdot \rho]$. Furthermore, if the bound is exceeded, respectively, under-run, we decrease, respectively, increase $\overline{h}^r$ by some small value before calculating h^r_{new}.

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 [7]. Therefore, it is necessary to reduce the major energy sources. Some Media Access Control (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.

A usual approach in reducing the consumed power is to periodically turn off the transceiver module if it is not required. A protocol using such a scheme is called a *duty-cycle protocol*. The *duty-cycle* is determined to be the ratio between the duration used for listening on synchronization messages to the medium and the duration of the complete period. As already mentioned, the bounded synchronization precision necessitates that the receiver module must be enabled some time prior, before any transmission takes place. This safety margin equals the synchronization window w and should be greater than the upper bound of the synchronization precision. A node considers itself to be synchronized, if the maximum absolute deviation to all neighboring nodes is smaller than the specified synchronization window. With respect to the relative message staggering delays $r_{\min}$, respectively, $r_{\max}$ and the period length T, the duty-cycle equals $r_{\max} - r_{\min} + 2 \cdot (w/T)$. Note that after a number of periods, each node has to listen to the medium for a full period in order to avoid clique building.

4. Evaluation by Simulation

We evaluated our approach with a probabilistic wireless sensor network simulator called JProwler (available at http://www.isis.vanderbilt.edu/Projects/nest/jprowler). 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 the Berkeley Mica Motes running TinyOS with the B-MAC protocol. It is a Java version of the Prowler [8] network simulator which is used for verifying and analyzing communication protocols of adhoc wireless sensor networks. Note that B-MAC is very similar to the IEEE 802.15.4 standard, because both implement the same CSMA/CA mechanism. Therefore, by modifying the MAC layer specific constants, the simulator can be used for the ZigBee nodes.

For instance, the transmission time is based on the amount of transmitted data. In our case, the synchronization frame contains the frame identifier (8 bit), the synchronization state (8 bit), the nominal phase offset (16 bit), the phase adjustment value (16 bit), the sender timestamp (32 bit), the tick-number (16 bit), and a checksum (8 bit). In sum, the application needs 13 byte for one synchronization message. The real amount of transmitted data is greater due to the payload of the MAC and the physical layer. According to the 802.15.4 MAC standard [5], the complete payload is about 15 byte (9 byte from the MAC layer and 6 byte from the physical layer). Assuming that the system works in the 2.4 GHz ISM band, the bit rate is 250 kbps. As a result the serialization delay t_{sd} equals $t_{\text{sd}} = [28 \cdot 8/250\,000]$ seconds $= 0.896$ milliseconds and therefore is assumed to be about one millisecond. Note that the propagation delay is negligible. Let $t_{\text{sw}} = 1$ milliseconds be the worst case constant software dependent time of the sender and receiver (including interrupt and buffer handling) used for the transmission, then the worst case constant transmission time is about 2 milliseconds. These parameters are reflected in JProwler's specific MAC constants by the *minimum waiting time* and the *transmission time* of a sync-message. Both parameters were modified to 1 milliseconds. To be more realistic, we have set the *random waiting time* to 2 milliseconds which corresponds to the delay jitter we have observed in several experiments. Note that we do not guarantee the correctness of these values. Similarly to the implementation in the real hardware, the backoff scheme was deactivated in order to reduce the additional resulting delay jitter. The graphical user interface was enhanced by several new dialogs which enables the user to modify various parameters during the simulation. We further extended the simulator by an oscillator model. Thus, every virtual node is based on an oscillator, for example, RC-oscillator or several crystal cuts. This allows the simulation of clock drift and its 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 consider the adjustment of the simulation speed and enabling/disabling nodes during the simulation.

4.1. 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 with respect to different parameter choices, that is, the coupling factor α and the number of nodes in the network.

In order to compare the simulation results with the outcomes from [3], 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. The *time to sync* defines the time until all nodes have entered the synchronization state and is determined by two parameters. These are the synchronization window w and the number of required periods a node has to keep within this window. In the following we call the amount of required periods *synchronization periods* and is 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. The definition of the *50th and 90th Percentile Group Spread* differs from that one defined in [3], because we only 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.

Incorrect results due to settling effects during the startup phase are avoided by postponing the start of the group spread measurement against the time to sync (t_s) and the time the experiment ends (t_e). On this account, the group spread measurement is performed only during the interval $[t_s + (t_e - t_s)/2, t_e]$.

4.1.1. 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. According to the datasheet, the real nodes have a nominal frequency of 8 MHz ± 10%. For this reason, every virtual node encounters a random initial clock drift between −100 milliseconds and +100 milliseconds per second. The general values of the other parameters are denoted in Table 2.

4.1.2. 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.

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 each other.

The simulation results based on this topology give a good overview on the impact of different coupling factors. According to the diagrams in Figure 5, 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. Table 3 summarizes the upper bound for different values of α with respect to Corollary 3.10. The stronger bound of Theorem 3.11 was neglected due to the fact that most network simulations using the weak upper bound with a random inital configuration have achieved synchronicity. The time to sync calculated in Table 3 are based on Theorem 3.15 for an initial maximum phase difference of $\Phi_0 = 0.4$. Note that a constant offset of 10 was added so that the values can be compared with the simulation result, because the simulator declares a set of nodes synchronized only if they are within some precision for at least 10 consecutive periods. The high time to sync bar in Figure 5(a) with $\alpha = 1.1$ and a number of 20 nodes comes from the fact that the coupling factor was too high.

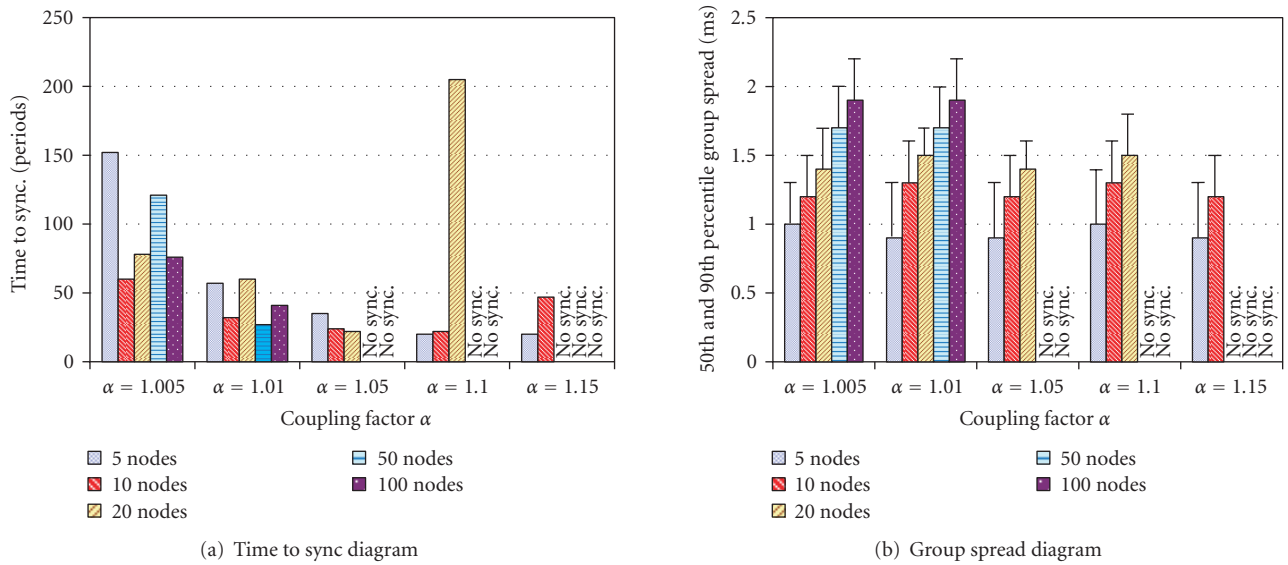
If we assume that the rate calibration scheme reduces the worst case drift to $\rho = 10$ ppm, then based on the parameters from Table 2 and Theorem 3.2, the worst case precision for $\alpha > 1.002$ equals $\Pi^U = 2.032$ milliseconds. Note that without the rate calibration scheme, we would have $\Pi^U = 322$ milliseconds. The group spread diagram complies with our theoretical result that the precision does not depend on α if α maintains the lower bound. Note that the simulations are based on a realistic radio model considering message collisions and transmission strength as well as the backoff scheme of the MAC layer. Therefore, the results show that if all network parameters (e.g., transmission delay, maximum drift rate, etc.) are correctly defined, then the worst case precision is maintained most of the time. However, there may exist outliers in the case of an omission failure of the fastest node, because this algorithm does not allow a node to adjust its clock backward.

Table 2: The general parameter choice used in all simulator experiments.

Parameter	Value
Oscillator technology	RC-oscillator
Initial clock drift ρ [ppm]	100000
Interval time T [milliseconds]	1000
Granularity (ticks/period) [Φ_{th}]	10000
Minimum message staggering delay $\Phi_{\text{msd}}^{\min}$ [milliseconds]	10
Maximum message staggering delay $\Phi_{\text{msd}}^{\max}$ [milliseconds]	300
Coupling factor α	1.01
Constant transmission delay ς [milliseconds]	1
Maximum delay jitter ϵ [milliseconds]	2
Synchronization window w [milliseconds]	10
Evaluation end [periods]	3600

Table 3: Calculated bounds for the coupling factor and the time to sync.

Number of nodes n	5	10	20	50	100
Upper bound for α	1.158	1.065	1.030	1.011	1.006
Coupling factor α	1.15	1.1	1.05	1.01	1.005
Estimated time to sync [s]	17	20	28	92	173



(a) Time to sync diagram

(b) Group spread diagram

Figure 5: 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.

The Multihop Topology. This communication topology is the most important one, because in reality many sensor networks are based on a source-to-sink communication topology with a communication path consisting several hops. The simplest multihop 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. Therefore, an estimation for the achievable precision with respect to Theorem 3.2 in a network with the network diameter d is about $\Pi \sim d \cdot \Pi^U$. Our solution is based on *grouped multihop* networks. Therein, the nodes are replaced with groups comprising several nodes in all-to-all topology which all have a bidirectional communication link to all nodes in the immediate neighboring groups. Figure 6 is a snapshot of a running simulation regarding the grouped multihop topology with JProwler. Therein, a dot represents a node and an arrow visualizes that a node is currently transmitting data. The gray scale visualizes the deviation. Note that all grouped multihop topologies treated in our experiments have the same network diameter of 10 hops but vary in the group size.

The diagrams in Figure 7 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. These diagrams lead 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. To sum up, it is difficult to find the best parameter settings for a given multihop 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.

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 [9]. 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 savings, the time-triggered approach also serves as an evaluation protocol. For this, we used a modified version of the TTP/A protocol [10].

The synchronization algorithm was implemented analogously to the implementation in JProwler. A simple RC-oscillator-based 16-bit timer was used to generate the

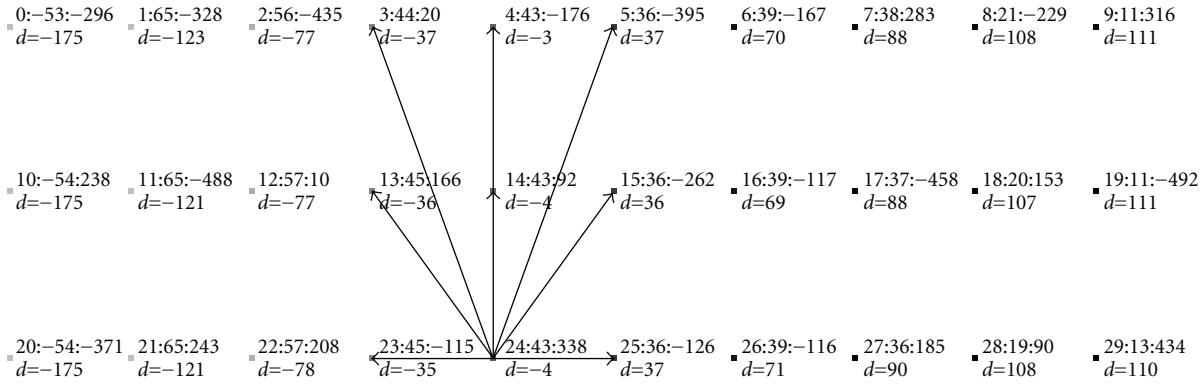
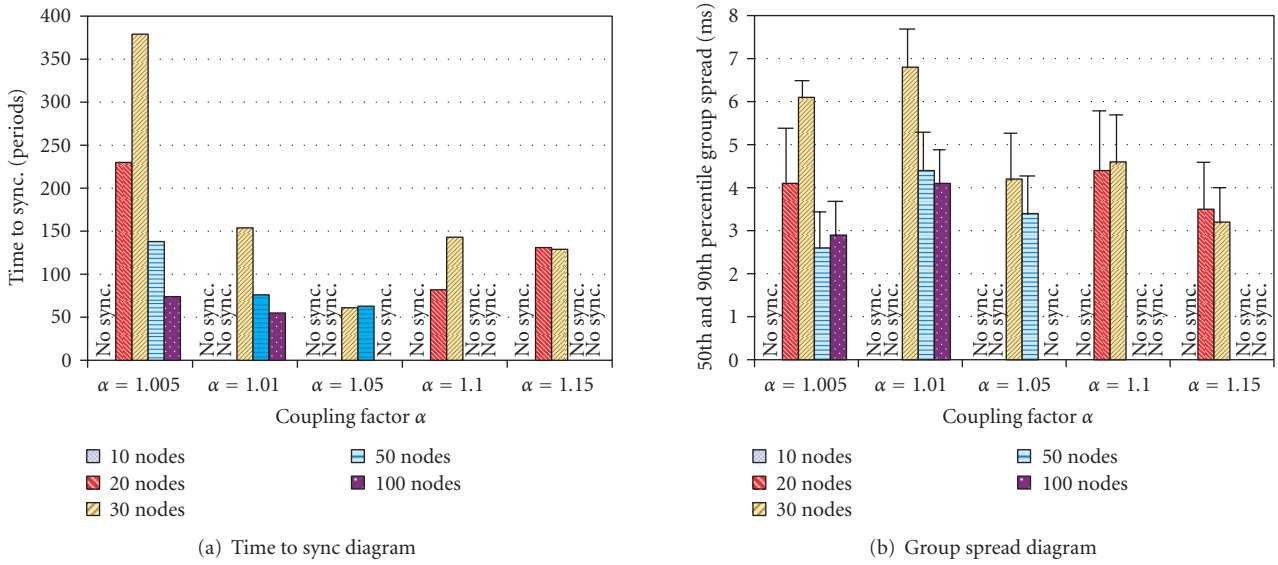


Figure 6: A simulation snapshot of a grouped multihop topology with a network diameter of 10 and a group size of 3 nodes.



(a) Time to sync diagram

(b) Group spread diagram

Figure 7: The time to sync and the group spread for a multihop 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.

synchronization interval with a duration of one second. The only differences with respect to the parameter choices in Table 2 are a higher granularity of the virtual clock (31250 ticks/period) and a higher transmission delay of 896 microseconds.

We modified the initial settings of the MAC sublayer, that is, the minimum backoff exponent, to reduce the transmission delay. We further assumed that it is better to omit a message than to transmit postponed synchronization data since the underlying MAC layer does not support MAC timestamping. However, several measurements have shown that regardless of this configuration, some messages are still transmitted with a high delay. Therefore the tradeoff between the probability of an omission failure and a high delay jitter with respect to the precision degradation has to be chosen. In detail, whereas high delay jitter only degrades the deterministic worst case precision, omission failures may mostly provide a better precision, though it is more indeterministic and therefore results in a worse worst case precision. Note that omission failures are especially a problem in the case they occur at the fastest node and may degrade the worst case precision by at about $k \cdot \Gamma$, where k denotes the maximum number of consecutive omission failures at the same node. 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 16-bit short addresses for each node.

5.2. Experiments and Results. The evaluation metrics for the testbed experiments are similar to the one used for the simulation experiments. In order 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.

Table 4: 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 configuration comprising 5 nodes.

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)

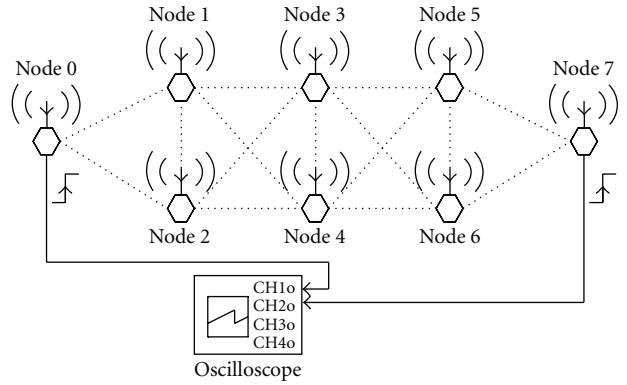


Figure 8: The measuring setup for visualizing the deviation between the edge nodes of a quasigrouped multihop network comprising 3 clusters with a cluster size of 2 and 2 edge nodes.

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. Since in reality it is not possible to speedup the time, we reduced the experiment time to 720 periods.

5.2.1. Testbed Results

The All-to-All Topology. For the all-to-all topology experiment, we again measured the group spread in dependence of several coupling factors.

Table 4 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 system are mostly better than the simulation results. Furthermore, the calculated worst case precision of $\Pi^U = 2.032$ milliseconds is also maintained by the 90th percentile group spread, even though it is worse compared to the simulation results. The outliers come from the fact that the delay jitter of the communication delay due to the MAC stack was sometimes higher than expected. 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. Furthermore, the worst case precision may also result from an omission failure from a synchronization message of the fastest node. The results from an all-to-all topology comprising 9 nodes are comparable with those denoted in Table 4. Note that only a maximum number of 9 nodes were available for our experiments.

The Multihop Topology. The results from the multihop 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 cannot be gathered automatically over several periods. Therefore, we manually made snapshots over several minutes and took those diagrams, which display the biggest time deviation. To simulate the multihop network, we have simply implemented a message filter.

The results show that the precision of a realistic multihop network with 4 hops is about 3 milliseconds and even maintains the calculated worst case precision of a fully connected network multiplied by 4. Interestingly, the simulation of the same network with a configured delay jitter of 1250 microseconds leads to the same result. To get an overview of the precision degradation with respect to the network diameter, another multihop experiment with 9 nodes was performed. The measurement setup is similar to the previous multihop network. The measurement results show a maximum deviation between the edge nodes of up to 14 milliseconds and is better than we have observed during the simulation where synchronicity was sometimes not achieved at all. Note that the measured worst case precision is again maintained by the calculated worst case precsion for a fully connected network multiplied by 9. However, such a high deviation was measured very seldom. In summary, the worst case precision of our synchronization algorithm degrades by at most the worst case precision for the corresponding fully connected network with each hop.

Table 5: Listing of the major energy consumers and their corresponding battery discharge. The energy calculation assumes a working voltage of 3 Volt.

Energy consumer	I [mA]	t [s]	Battery discharge per cycle	Energy consumption per cycle
Firing time, part 1	20.0	0.060	1.200 mAs	3.600 mJ
Firing time, part 2	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}	6.2 mA $\cdot$ t_{idle}	18.6 mW $\cdot$ t_{idle}
$\sum$		T	1.648 mAs + 6.2 mA $\cdot$ t_{idle}	4.944 mJ + 18.6 mW $\cdot$ t_{idle}

Note that with respect to [3], the results can be compared with the simulation results regarding a regular grid topology containing 16 nodes. Therein, they have measured a 90th percentile group spread of about 10 milliseconds. Compared to their results, our worst case deviation of 14 milliseconds is not so digressive.

We further measured the behavior of a grouped multihop network as shown in Figure 8 comprising 3 groups with a group 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 available nodes. The maximum deviation between the edge nodes measured over about 10 minutes was 7 milliseconds. In the simulator we had to configure a delay jitter of 7 milliseconds or in the case of no delay jitter an uncompensated additional transmission delay of 1.5 milliseconds 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 1.5 V 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 3 V $\cdot$ 1200 mAh = 3600 mWh. The lifetime in hours is the ratio $W_{\text{bat}}/P_{\text{avg}}$ and can be reduced to the equivalent formula $t_{\text{life}} = E_{\text{bat}}/I_{\text{avg}}$, 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 and were measured by the use of an oscilloscope and a current shunt resistor. 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 50 milliseconds. The consumed current during this time is about 20 mA. 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 safety margin is further named part 2 of the firing time and consumes a current of 24 mA.

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, that is, the transceiver is disabled. With our ZigBee nodes, we measured a current of about 6.2 mA.

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 11 mA for a duration of 1 milliseconds. 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 25 mA over a time of 4.8 milliseconds. Note that this duration does not equal the real transmission time of about 1 milliseconds. This comes from the fact, that the transceiver requires some time for the startup phase and that the Carrier Sense Multiple Access with Colission Avoidance (CSMA/CA) scheme at the MAC layer may also cause some delay although the backoff scheme was disabled.

Table 5 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 $E_{\text{device}} = 7.358$ mAs. Thus, the average current consumption I_{avg} also equals 7.358 mA. Assuming that our batteries deliver

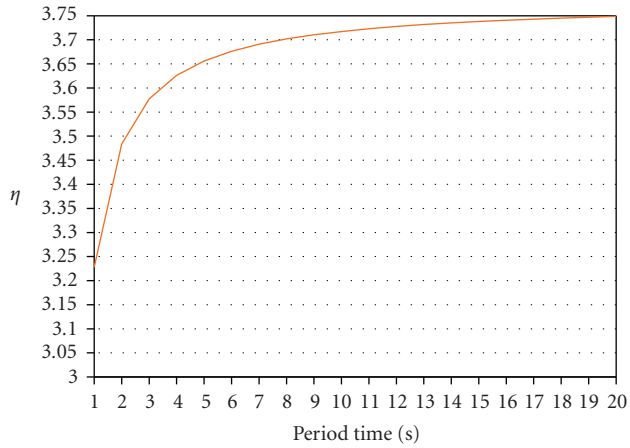


FIGURE 9: The lifetime improvement η as a function of the period time.

a charge of about $E_{\text{bat}} = 1200\,\text{mAh}$, then the resulting lifetime (t_{life}) in hours can be calculated as follows:

$$t_{\text{life}} = \frac{E_{\text{bat}}}{I_{\text{avg}}} = \frac{1200\,\text{mAh}}{7.358\,\text{mA}} = 163\,\text{h} \simeq 1\ \text{week}. \quad (13)$$

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 $\text{DC} = (t_{f,1} + t_{f,2} + t_e + t_t)/T$.

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, that is, the duty-cycle equals 100%. In that case, the average current consumption equals 23.752 mA. Consequently, a duty-cycle of 100% corresponds to a lifetime of about 50.5 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 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 of time. The improvement factor equals $\eta = I_{\text{avg}}(100\%)/I_{\text{avg}}(\text{DC}(T))$ and as visualized in Figure 9.

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 multihop 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 regarding highly multihop scenarios with high requirements on availability and dependability makes the employment with low-cost nodes in such an environment usually improper. Otherwise, the proposed grouped multihop topology with a small network diameter allows the forwarding more reliable than it would be in a standard adhoc network. For instance, the nodes in such a single group could be statically configured via the RODL file such that they all forward the same message to the neighboring groups in different slots, but within the same period. In other words, such a group comprises several nodes which have the same functionality (e.g., sensor measurement, relaying, sensor fusion, computation, actuator control in combination with Triple Modular Redundancy (TMR)). If so, then a single node failure has no impact on the network behavior, since there exist no dedicated nodes. Thus, graceful degradation is the main advantage of this protocol.

The time-triggered approach a priori requires a static communication plan (the RODL file) for each node in the network. Therefore, the network and possible multihop scenarios must be analyzed before the RODL file can be created. This requires additional work and expenses compared to other protocols. However, the inherent advantage is that the deployed communication schedule can be perfectly adapted to the current known static network topology. Therefore, multihop routing algorithms are no longer necessary, since the routing is done implicitly during the configuration process of the RODL files. Though, a change in the network topology may also involve a reconfiguration of all RODL files.

We have shown that our approach provides a duty-cycle of about 7% and thus reduces the energy consumption in a simple network by at least a factor of three. Note that the duty-cycle heavily depends on the synchronization precision. Since we consider the implementation on low-cost nodes by the use of an off-the-shelf MAC layer which does not support MAC timestamping, our synchronization precision is limited to the MAC specific delay jitter. Other established protocols such as S-MAC or the slotted variant of IEEE 802.15.4 use an alternative or more sophisticated clock synchronization approach and thus usually achieve a duty-cycle of lower than 1%. However, such protocols usually demand a dedicated master node (and maybe additional backup nodes) which represents a single point of failure. In contrast, our approach has the inherent advantage that it does not require any dedicated nodes. Furthermore, a higher synchronization precision and thus a lower duty-cycle can indeed be achieved by more accurate but also more expensive external crystal oscillators. As a result, the tradeoff lies between the costs of a node and the power consumption.

7. Related Work

Literature on biologically inspired Firefly synchronization can be categorized into papers about the biological/mathemathical models of the Firefly approach [1, 4, 11], work that treats the biologically inspired Firefly synchronization model for realizing the communication in sensor networks [3, 12] and architectures and evaluations that apply the Firefly synchronization model to establish a time-triggered service [13].

Werner-Allen et al. [3] present the Reachback Firefly Algorithm, which is well-suited for the implementation in sensor networks. The algorithm was simulated with TOSSIM over several parameter choices (e.g., different node topologies, coupling strength, and network diameters). values, and network diameter.

In [14], the authors introduce a time advance strategy based on the PCO model, which takes the delays in wireless systems into account. Similarly to [3], 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 based on the off-the-shelf MAC layer IEEE 802.15.4. 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. This has the inherent advantage, that no dedicated synchronization node is required and thus there exist no single point of failure. 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 1 milliseconds. Unfortunately, the testbed system suffered from an unexpected delay jitter and an additional communication delay. For this reason, the testbed results considering highly multihop topologies were worse compared to the simulation results with a low delay jitter.

Future work will rely on the improvement of the synchronization precision by the use of an alternative MAC-Stack supporting MAC timestamping. Furthermore, we want to extend the algorithm in order to be resilient to compromised nodes which may behave as an adversary trying to destroy the synchronization. Last but not least, we want to compare the results of our approach with other synchronization schmes designed for wireless networks.

Acknowledgments

This work was supported by the Austrian FWF project TTCAR under contract no. P18060-N04. The authors would like to thank Johannes Klinglmayr for constructive comments on an earlier version of this paper.

References

[1] J. Buck, "Synchronous rhythmic flashing of fireflies. II," *The Quarterly Review of Biology*, vol. 63, no. 3, pp. 265–289, 1988.

[2] 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, December 2003.

[3] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)*, pp. 142–153, San Diego, Calif, USA, November 2005.

[4] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, no. 6, pp. 1645–1662, 1990.

[5] IEEE Computer Society, "IEEE Standard for Information technology—telecommunication and information exchange between systems—local and metropolitan area networks—specific requirements—part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," Institute of Electrical and Electronics Engineers, September 2003.

[6] J. Lundelius and N. Lynch, "An upper and lower bound for clock synchronization," *Information and Control*, vol. 62, no. 1, pp. 190–204, 1984.

[7] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[8] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in *Proceedings of the IEEE Aerospace Conference*, vol. 3, pp. 1339–1346, Big Sky, Mont, USA, March 2003.

[9] Atmel Corporation, "ATAVRRZ200 Demonstration Kit AT86RF230 (2450 MHz band) Radio Transceiver User Guide," Document No. 5183AZIGB12/07/06, July 2006.

[10] S. Eberle, C. Ebner, W. Elmenreich, et al., "Specification of the TTP/A protocol," Research Report 61/2002, Institut für Technische Informatik, Technische Universität Wien, Vienna, Austria, September 2002, Version 2.00.

[11] J. Buck and E. Buck, "Synchronous fireflies," *Scientific American*, vol. 234, no. 5, pp. 74–85, 1976.

[12] A. Tyrell, G. Auer, and C. Bettstetter, "Biologically inspired synchronization for wireless networks," in *Studies in Computational Intelligence*, F. Dressler and I. Carreras, Eds., vol. 69, pp. 47–62, Springer, Berlin, Germany, 2007.

[13] R. Leidenfrost and W. Elmenreich, "Establishing wireless time-triggered communication using a firefly clock synchronization approach," in *Proceedings of the 6th International Workshop on*

Intelligent Solutions in Embedded Systems (WISES '08), pp. 1–18, Regensburg, Germany, July 2008.

[14] A. Tyrrell, G. Auer, and C. Bettstetter, "Firefly synchronization in ad hoc networks," in *Proceedings of the MiNEMA Workshop*, Leuven, Belgium, February 2006.