

# Internal Synchronization of Drift-Constraint Clocks in Ad-Hoc Sensor Networks

Lennart Meier<sup>\*</sup>  
meier@tik.ee.ethz.ch

Philipp Blum<sup>†</sup>  
blum@tik.ee.ethz.ch

Lothar Thiele  
thiele@tik.ee.ethz.ch

Computer Engineering and Networks Laboratory  
Swiss Federal Institute of Technology (ETH) Zürich  
CH-8092 Zürich, Switzerland

## ABSTRACT

Clock synchronization is a crucial basic service in typical sensor networks, since the observations of distributed sensors more often than not need to be ordered (“ $a$  happened before  $b$ ”) or otherwise related (“ $a$  and  $b$  happened within a time window of size  $x$ ”) in time. Ad-hoc networks may exhibit characteristics which make the use of traditional clock-synchronization algorithms infeasible. Recently, algorithms suitable for ad-hoc networks have been presented.

We first propose an improvement to an existing algorithm. While needing less computation and no more communication or memory than the original algorithm, our new algorithm always yields equal or better results and thus outperforms the original algorithm. We then examine how even better synchronization can be obtained, possibly at the cost of additional computation, communication, and memory. To this end, we introduce a model for internal synchronization. This model allows us to find an algorithm which makes use of all the data a node can obtain from the network for a given communication pattern and thus provides optimal synchronization in our model.

## Categories and Subject Descriptors

C.2.4 [Computer Systems Organization]:  
Computer-Communication Networks—*Distributed Systems*.

## General Terms

Algorithms, Performance, Reliability, Theory.

## Keywords

Sensor Networks, Time Synchronization, Clock Drift, Delay.

<sup>\*</sup>The author was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

<sup>†</sup>The author was supported by the Swiss Commission for Technology and Innovation CTI/KTI, Grant 4957.2.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*MobiHoc'04*, May 24–26, 2004, Roppongi, Japan.

Copyright 2004 ACM 1-58113-849-0/04/0005 ...\$5.00.

## 1. INTRODUCTION

Clock synchronization is an important service in typical ad-hoc sensor networks. For example, the correct evaluation of distributed sensor data may require knowledge about the chronology of the sensor observations [10]. In addition, energy consumption can be reduced by synchronous power-on and shutdown of the wireless-communication circuits of a sender–receiver pair [2, 3, 13].

Clock synchronization in ad-hoc (and thus wireless) sensor networks poses challenges that are substantially different from those in infrastructure-based networks. *Robustness*: There is no stable connectivity between nodes. *Energy efficiency*: Synchronization can only be achieved and maintained by communication. Communication is expensive in terms of energy, which typically is a scarce resource for sensor nodes. *Ad-hoc deployment*: The clock-synchronization service must not rely on any a-priori configuration or on infrastructure.

The conclusion is that traditional algorithms, such as the well-known Network Time Protocol [8], cannot be directly applied in ad-hoc sensor networks. One suitable algorithm was proposed in [10]. However, it does not provide optimal synchronization.

### 1.1 New results and related work

This paper extends the work presented in [10] both on an algorithmic and on a theoretical level. We modify the algorithm presented in [10] and obtain better synchronization with less computation and no additional communication or memory. We then examine how even better synchronization can be obtained, possibly at the cost of additional computation, communication, and memory. To this end, we introduce a model for internal synchronization. This model allows us to find an algorithm which makes use of all the data a node can obtain from the network for a given communication pattern.

The algorithms in this paper provide guaranteed bounds on the local times of nodes. Computing such bounds instead of time estimates was first proposed in the context of external synchronization by Marzullo and Owicki [6] and was further studied by Schmid and Schossmaier [11].

There has been much work on clock synchronization [7, 12]. However, most of the proposed synchronization algorithms rely on a network that is not partitioned and where good estimates for message delays are available. Some algorithms are also based on the assumption that the actual clock drift is a linear function of time. These assumptions may not hold in ad-hoc sensor networks where connectivity is limited, delays may vary due to the shared wireless medium, and clock drifts may fluctuate considerably because of changing environmental conditions.

## 1.2 Overview

In Sect. 2, we formally define the system model and state the problem we want to solve. In Sect. 3, we revisit the algorithm from [10], propose an improvement and show that our improved algorithm always outperforms the original algorithm. In the rest of the paper, we make use of a simplified system model which we introduce in Sect. 4. In Sect. 5, we then proceed to show how optimal synchronization in simple scenarios is achieved, laying the foundation for the analysis of more complex scenarios in Sect. 6, where we propose an algorithm which makes use of all the data available for a given communication pattern. It thus provides optimal bounds, as we show in Sect. 7.

## 2. SYSTEM MODEL AND PROBLEM STATEMENT

In this section, we give a model that captures those aspects of an ad-hoc sensor network that are essential to our analysis. We then state the problem that we want to solve in the rest of the paper.

### 2.1 Scenarios, traces, and views

To model the interaction of the nodes in a sensor network, we define two types of events: sensor and communication events.

*Sensor events.* A sensor node may receive sensor data from outside the network. We model this as a *sensor event*  $s$  which occurs at the sensor node at real time  $t_s$ .

*Communication events.* We assume that communication occurs point-to-point between two sensor nodes. We model this as a *communication event*  $c$  which occurs at real time  $t_c$ . A communication event can consist of either one message being sent from a sender to a receiver, or of two messages being exchanged between the two communicating nodes. We will use the first notion of communication event until the end of Sect. 3, and then switch to the second one.

#### 2.1.1 Scenarios, traces, and views.

We call the set of all events in a system and the real times at which they occur a *scenario*. A scenario describes a particular communication pattern.

For a given scenario, the local times of events can still be chosen within certain limits imposed by the clock model (Sect. 2.2). If we fix these times, we obtain an extension of a scenario: We call the combination of a scenario and corresponding local times for all events a *trace*.

A synchronization algorithm obviously has access only to the information known to the nodes executing it. We define the *view*  $V_i(t)$  of node  $N_i$  at real time  $t$  as all the information that node  $N_i$  could have obtained until time  $t$ .

### 2.2 Clock model

Each node of the network is equipped with a local clock; the reading of node  $N_i$ 's clock at real time  $t$  is denoted as  $h_i(t)$ . We will refer to the clock of node  $N_i$  simply as  $h_i$ . The drift of a clock  $h_i$  at time  $t$  is defined as the deviation of its speed from the "correct" speed and is thus given by

$$\rho_i(t) = \frac{dh_i(t)}{dt} - 1 . \quad (1)$$

For times  $t_a, t_b$  with  $t_a \neq t_b$ , we define the average drift in  $[t_a, t_b]$  as

$$\hat{\rho}_i(t_a, t_b) = (h_i(t_b) - h_i(t_a)) / (t_b - t_a) .$$

The quality of synchronization that can be achieved depends on the assumptions about the properties of the nodes' clocks. In this paper,

we suppose that the absolute value of clock  $h_i$ 's drift is limited by a known constant  $\hat{\rho}_i$ :

$$-\hat{\rho}_i \leq \rho_i(t) \leq \hat{\rho}_i \quad \forall t . \quad (2)$$

We call a clock for which (2) holds a *drift-constraint clock* (DCC) and a node equipped with such a clock a DCC node.

We require  $\rho_i(t) > -1$  for all times  $t$  (and also  $\rho_i(t) < 1$ , for reasons that will become clear later). This means that a local clock can never stop ( $\rho_i(t) = -1$ ) or run backward ( $\rho_i(t) < -1$ ). Thus, for two events  $a, b$  with  $t_a < t_b$  occurring at node  $N_i$  (whose clock's drift  $\rho_i$  is bounded according to (2)), node  $N_i$  can compute bounds  $\Delta_i^l[a, b], \Delta_i^u[a, b]$  on the real-time difference  $\Delta[a, b] := t_b - t_a$  as

$$\Delta_i^l[a, b] := \frac{h_i(t_b) - h_i(t_a)}{1 + \hat{\rho}_i} \quad \Delta_i^u[a, b] := \frac{h_i(t_b) - h_i(t_a)}{1 - \hat{\rho}_i} .$$

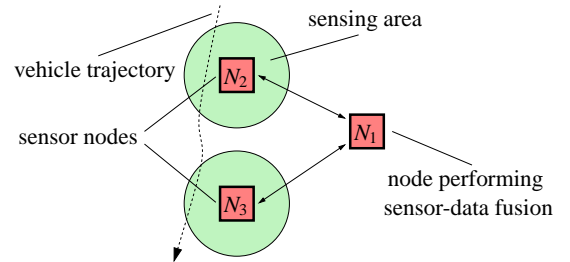
#### 2.2.1 Admissible trace.

We call a trace *admissible* if it satisfies the constraints on clock drifts as defined in (2).

### 2.3 Problem statement

We define the problem of internal synchronization as follows: Given a trace with an event occurring at node  $N_j$  at local time  $h_j(t)$ , we are interested in tight bounds  $H_i^l(t), H_i^u(t)$  on the local time  $h_i(t)$  of another node  $N_i$  at time  $t$ , such that  $H_i^l(t) \leq h_i(t) \leq H_i^u(t)$ .

In the context of a sensor network, this situation arises, e.g., when a sensor event  $s$  is observed at time  $t_s$  by node  $N_j$ , and another node  $N_i$  wants to obtain bounds on  $h_i(t_s)$ . In the example in Fig. 1, the sensor nodes  $N_2$  and  $N_3$  sense a vehicle at local times  $h_2(t_1)$  and  $h_3(t_2)$ .<sup>1</sup> If node  $N_1$  can compute bounds on  $h_1(t_1)$  and  $h_1(t_2)$  which show that  $t_1 < t_2$  (i.e.  $H_1^u(t_1) < H_1^l(t_2)$ ), it can conclude that the vehicle entered the sensing area of  $N_2$  first and then that of  $N_3$ .



**Figure 1: To perform correct sensor-data fusion, node  $N_1$  has to be able to relate the observations of nodes  $N_2$  and  $N_3$  in time.**

#### 2.3.1 Guaranteed bounds on local time.

All algorithms discussed in this paper provide guaranteed bounds on *local time*. Given the local time of one node at a certain real time, bounds on the local time of another node at the same real time are computed. We call the difference between the upper and the lower bound the *uncertainty*.

Interestingly, the use of guaranteed bounds has not received much attention, although it has a number of advantages over using time estimates: (a) Guaranteed bounds on the local times at which sensor events occurred allow to obtain guaranteed bounds from sensor-data fusion. (b) The concerted action (sensing, actuating, communicating) of several nodes at a predetermined local time of a specific

<sup>1</sup>For simplicity, we assume that a sensor event is only generated at the first sensing of the vehicle, i.e. at the moment where the vehicle enters the sensing area.

clock always succeeds: each node can minimize its uptime while guaranteeing its activity at the predetermined time. (c) The combination of several bounds for a single local time is unambiguous and optimal, while the reasonable combination of time estimates requires additional information about the quality of the estimates.

### 3. IMPROVING AN EXISTING ALGORITHM

In this section, we first revisit the algorithm from [10]. We then identify a straightforward improvement of that algorithm, and finally compare the two algorithms by quantifying this improvement. The result is that our improved algorithm always outperforms the original algorithm from [10].

#### 3.1 The algorithm from [10]

In [10], synchronization is achieved as follows: At some time after the occurrence of a sensor event  $s$  at node  $N_j$ , this node sends bounds on the local time  $h_j(t_s)$  at which the event occurred to node  $N_i$ . Node  $N_i$  receives these bounds and transforms them into bounds on the local time  $h_i(t_s)$  its own clock was showing at time  $t_s$ . We will now see in detail how this transformation is done.

The message exchange between a sender and a receiver node is depicted in Fig. 2. At time  $t_3$ , the sender node  $N_j$  observes a sensor event and stores its local time  $h_j(t_3)$ . At time  $t_4$ , node  $N_j$  sends  $h_j(t_3)$ , the local time  $h_j(t_2)$  at which the last acknowledgment ACK<sub>1</sub> was received from node  $N_i$ , and the current local time  $h_j(t_4)$  to the receiver node  $N_i$  in a single message M<sub>2</sub>. This message is received by  $N_i$  at local time  $h_i(t_5)$ . Node  $N_i$  can now compute bounds on  $h_i(t_3)$  by subtracting from  $h_i(t_5)$  the maximal and minimal local time that can have elapsed in the real-time interval  $[t_3, t_5]$ . For the lower bound, this yields

$$H_i^l(t_3) = h_i(t_5) - (h_j(t_4) - h_j(t_3)) \frac{1 + \hat{\rho}_i}{1 - \hat{\rho}_j} - \left( (h_i(t_5) - h_i(t_1)) - (h_j(t_4) - h_j(t_2)) \frac{1 - \hat{\rho}_i}{1 + \hat{\rho}_j} \right). \quad (3)$$

The expression on the second line of (3) is an upper bound on the local-time difference  $h_i(t_5) - h_i(t_4)$ , i.e. on the delay of M<sub>2</sub> expressed in local time of  $N_i$ . For the upper bound, we have

$$H_i^u(t_3) = h_i(t_5) - (h_j(t_4) - h_j(t_3)) \frac{1 - \hat{\rho}_i}{1 + \hat{\rho}_j}. \quad (4)$$

#### 3.2 Improvement to the algorithm

Equation (3) can be simplified algebraically to

$$H_i^l(t_3) = h_i(t_1) + (h_j(t_4) - h_j(t_2)) \frac{1 - \hat{\rho}_i}{1 + \hat{\rho}_j} - (h_j(t_4) - h_j(t_3)) \frac{1 + \hat{\rho}_i}{1 - \hat{\rho}_j}.$$

This equation shows that the lower bound is actually computed by first advancing the local time  $h_i(t_1)$  by the minimum local time (of node  $N_i$ ) that passes in the real-time interval  $[t_1, t_4]$  (note that  $t_2 - t_1 = 0$  is assumed, i.e. the delay of ACK<sub>1</sub> is assumed to be zero), and then subtracting the maximum local time that passes in the interval  $[t_3, t_4]$ . Informally speaking, we start at  $t_1$ , walk past  $t_2$  and  $t_3$  to  $t_4$  and then back to  $t_3$ . Our improvement is straightforward: We compute the lower bound  $H_i^l(t_3)$  by advancing  $h_i(t_1)$  by the minimum local time that passes in the real-time interval  $[t_1, t_3]$ , i.e.

$$H_i^l(t_3) = h_i(t_1) + (h_j(t_3) - h_j(t_2)) \frac{1 - \hat{\rho}_i}{1 + \hat{\rho}_j}. \quad (5)$$

Note that we do not change the way the upper bound  $H_i^u(t_3)$  is computed.

### 3.3 Comparison

The difference in uncertainty of the two algorithms is

$$\begin{aligned} \Delta U &= (h_j(t_4) - h_j(t_3)) \left( \frac{1 + \hat{\rho}_i}{1 - \hat{\rho}_j} - \frac{1 - \hat{\rho}_i}{1 + \hat{\rho}_j} \right) \\ &= (h_j(t_4) - h_j(t_3)) \frac{2(\hat{\rho}_i + \hat{\rho}_j)}{1 - (\hat{\rho}_j)^2}. \end{aligned}$$

The improvement over the original algorithm is largest for large values of  $h_j(t_4) - h_j(t_3)$  and large maximal drifts. Since our algorithm provides an equal or better uncertainty with less computation and otherwise equal resources, it is strictly superior to the algorithm from [10].

We now give a numerical example which we will come back to later in the paper. We let  $N_i$  and  $N_j$  communicate periodically every hour. The event  $s$  occurs 100 seconds after a communication, and  $N_i$  and  $N_j$  both constantly have maximal drift  $+\hat{\rho}_i = +\hat{\rho}_j = 100\text{ppm}$ . We need not make any assumption about the message delays, since they do not influence the difference in uncertainty, which for the above values is greater than 1400ms.

## 4. A SIMPLIFIED MODEL

Our improved algorithm from Sect. 3 does not provide optimal bounds in all cases. In the rest of the paper, we examine how optimal bounds can always be obtained, possibly at the cost of additional computation, communication and memory. In this section, we introduce a simplified system model which allows us to focus on the relevant factors.

### 4.1 System model

We now introduce a simplified model which we will use in the rest of the paper. The simplification consists in ignoring message-delay uncertainties and combining a message exchange between two nodes into a single, atomic communication event. This is shown in Fig. 3, where during each of the communication events  $a$  and  $b$ , each node sends and receives one message (we do not count acknowledgments as messages here).

Although our model contains some simplifications, our analysis is still useful because it gives bounds on what is achievable in case of known message delays. By exchanging multiple messages between two nodes, the delay uncertainty can be bounded to a few microseconds [1, 4, 5, 9]. Such bounds on the delay can be integrated into our equations. We will now argue that for typical sensor networks, the impact of message-delay uncertainty can be neglected.

#### 4.1.1 Drift vs. delay uncertainty.

Clock-synchronization algorithms face two problems: The information a node has about the local time of another node degrades over time due to clock drift (this is illustrated in Fig. 4), and its improvement through communication is hindered by message-delay uncertainty. The influence of drift and delay uncertainty can to a large extent be studied separately.

The influence of the clock drift on the quality of synchronization may dominate over the influence of the message delays. This is the case in those ad-hoc sensor networks where communication is sporadic not only in the sense of unpredictable, but also in the sense of *infrequent*. With decreasing frequency of communication, the

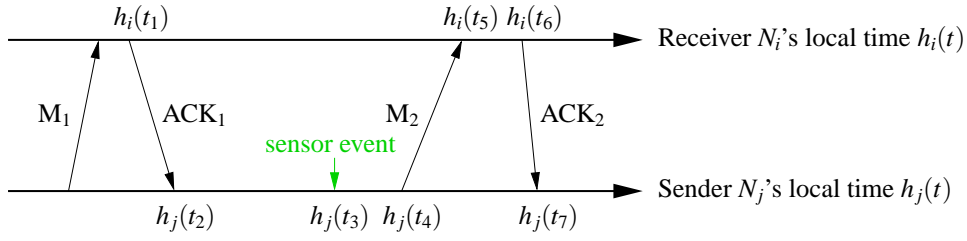


Figure 2: Message exchange between a sender and a receiver.

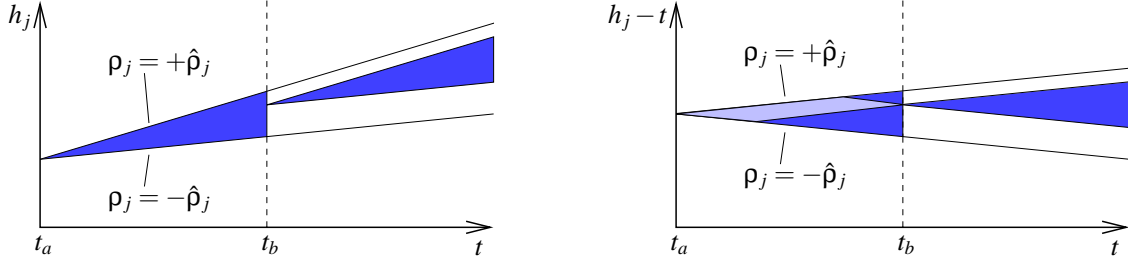


Figure 4: Graphical representation of the knowledge of node  $N_i$  about the local time  $h_j$  of node  $N_j$  as a function of real time  $t$ . The shaded area is the region in which  $h_j$  can lie. The two nodes communicate at events  $a$  and  $b$ . On the right,  $h_j - t$  is plotted against  $t$ ; additionally, the information about past values of  $h_j$  that  $N_i$  gathers at event  $b$  is shown as a lighter-shaded area.

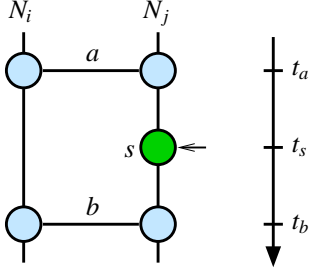


Figure 3: Two nodes  $N_i, N_j$  communicate at events  $a$  and  $b$ . A sensor event  $s$  occurs at node  $N_j$ .

uncertainty due to clock drift increases, while the uncertainty due to message delays remains constant. A numeric example: Suppose the message delay contributes 1 millisecond to a node's uncertainty, and the clock drift is bounded by  $\hat{\rho} = 100\text{ppm}$ . After 5 seconds, the drift's contribution to the uncertainty equals that of the delay. After one hour, it is 720 times larger. In such settings, neglecting the delay uncertainty is acceptable.

## 5. OPTIMAL SYNCHRONIZATION IN A SIMPLE SCENARIO

In this section, we examine the simple scenario depicted in Fig. 3. We propose straightforward bounds and show that they are tight. This lays the foundation for the analysis of more complex scenarios in Section 6.

### 5.1 Basic setting

In the scenario depicted in Fig. 3, we assume that both nodes gather and exchange data as follows:

1. At event  $a$ , the nodes exchange their local-clock readings  $h_i(t_a), h_j(t_a)$ .

2. At event  $s$ , node  $N_j$  timestamps the event with the local-clock reading  $h_j(t_s)$ .
3. At event  $b$ , the nodes exchange their local-clock readings  $h_i(t_b), h_j(t_b)$ . Furthermore, node  $N_j$  communicates the time stamp  $h_j(t_s)$  to node  $N_i$ .

In the end, both nodes share all the available information. The computation of the bounds  $H_i^l(t_s), H_i^u(t_s)$  we show in the following can thus be performed by any of the two nodes.

### 5.2 Computation of the bounds

The following two lemmas show how tight bounds  $H_i^l(t_s), H_i^u(t_s)$  on  $h_i(t_s)$  are obtained from local times of events preceding or succeeding event  $s$ .

LEMMA 5.1. *Let two DCC nodes  $N_i, N_j$  with drift constraints  $\hat{\rho}_i, \hat{\rho}_j$  be given. If only the local-clock readings  $h_i(t_a), h_j(t_a)$  of both clocks at some time  $t_a$  and the value  $h_j(t_s)$  for some  $t_s > t_a$  are given, then tight bounds  $H_i^l(t_s), H_i^u(t_s)$  with  $H_i^l(t_s) \leq h_i(t_s) \leq H_i^u(t_s)$  are given by*

$$H_i^l(t_s) = h_i(t_a) + \frac{h_j(t_s) - h_j(t_a)}{1 + \hat{\rho}_j} (1 - \hat{\rho}_i)$$

$$H_i^u(t_s) = h_i(t_a) + \frac{h_j(t_s) - h_j(t_a)}{1 - \hat{\rho}_j} (1 + \hat{\rho}_i) .$$

PROOF. According to (2), the real-time difference  $\Delta[a, s] = t_s - t_a$  is limited by

$$\frac{h_j(t_s) - h_j(t_a)}{1 + \hat{\rho}_j} \leq \Delta[a, s] \leq \frac{h_j(t_s) - h_j(t_a)}{1 - \hat{\rho}_j} .$$

The local clock of node  $h_i$  has advanced for a period of  $\Delta[a, s]$  with a speed between  $1 - \hat{\rho}_i$  and  $1 + \hat{\rho}_i$ . The lower bound  $H_i^l(t_s)$  follows by combination of minimum  $\Delta[a, s]$  with minimum speed, the upper bound  $H_i^u(t_s)$  analogously for maximum values. The bounds are tight since they are indeed attained for these combinations.  $\square$

Looking again at the scenario depicted in Fig. 3, bounds on  $h_i(t_s)$  can of course be computed also by relating  $h_j(t_s)$  to  $h_j(t_b)$  instead of to  $h_j(t_a)$ .

LEMMA 5.2. *Let two DCC nodes  $N_i, N_j$  with drift constraints  $\hat{\rho}_i, \hat{\rho}_j$  be given. If only the local-clock readings  $h_i(t_b), h_j(t_b)$  of both clocks at some time  $t_b$  and the value  $h_j(t_s)$  for some  $t_s < t_b$  are given, then tight bounds  $H_i^l(t_s), H_i^u(t_s)$  with  $H_i^l(t_s) \leq h_i(t_s) \leq H_i^u(t_s)$  are given by*

$$H_i^l(t_s) = h_i(t_b) - \frac{h_j(t_b) - h_j(t_s)}{1 - \hat{\rho}_j} (1 + \hat{\rho}_i) \quad (6)$$

$$H_i^u(t_s) = h_i(t_b) - \frac{h_j(t_b) - h_j(t_s)}{1 + \hat{\rho}_j} (1 - \hat{\rho}_i) \quad (7)$$

PROOF. According to (2), the real-time difference  $\Delta[s, b] = t_b - t_s$  is limited by

$$\frac{h_j(t_b) - h_j(t_s)}{1 + \hat{\rho}_j} \leq \Delta[s, b] \leq \frac{h_j(t_b) - h_j(t_s)}{1 - \hat{\rho}_j}.$$

The local clock of node  $h_i$  has advanced for a period of  $\Delta[s, b]$  with a speed between  $1 - \hat{\rho}_i$  and  $1 + \hat{\rho}_i$ . Since  $t_s < t_b$ , the lower bound  $H_i^l(t_s)$  follows by combination of maximum  $\Delta[s, b]$  with maximum speed, the upper bound  $H_i^u(t_s)$  for minimum values. The bounds are tight since they are indeed attained for these combinations.  $\square$

### 5.3 Comparing and combining Lemmas 5.1 and 5.2

If we compare the uncertainty about  $h_i(t_s)$  for the bounds from Lemmas 5.1 and 5.2, we can observe that the two methods yield the same uncertainty (but not the same bounds) if

$$h_j(t_s) = \frac{1}{2}(h_j(t_b) + h_j(t_a)).$$

(Note that this does generally not mean  $t_s = \frac{1}{2}(t_b + t_a)$ .) Otherwise, using the local-clock reading which is closer to  $h_j(t_s)$  yields a smaller uncertainty and thus a better result.

Obviously, we can use both  $h_j(t_a)$  and  $h_j(t_b)$ , combining the different bounds from Lemmas 5.1 and 5.2. We state this in the following corollary.

COROLLARY 5.3. *Let two DCC nodes  $N_i, N_j$  with drift constraints  $\hat{\rho}_i, \hat{\rho}_j$  be given. If for times  $t_a < t_s < t_b$  only the local-clock readings  $h_i(t_a), h_j(t_a), h_i(t_b), h_j(t_b)$  of both clocks and the value  $h_j(t_s)$  are given, then tight bounds  $H_i^l(t_s), H_i^u(t_s)$  with  $H_i^l(t_s) \leq h_i(t_s) \leq H_i^u(t_s)$  are given as the maximum and the minimum of the lower and upper bounds computed according to Lemmas 5.1 and 5.2.*

The uncertainty  $H_i^u(t_s) - H_i^l(t_s)$  according to Corollary 5.3 becomes minimal (i.e., zero) for maximal drift diversity, i.e. for  $\bar{\rho}_i = \pm \hat{\rho}_i$  and  $\bar{\rho}_j = \mp \hat{\rho}_j$ . We will see in the following that drift diversity is always beneficial.

## 6. ARBITRARY SCENARIOS

In this section, we examine synchronization in arbitrary scenarios. We first identify the general principle by which bounds on local times are computed from bounds on real-time intervals. We then propose an algorithm which computes these intervals using all the data available for a given communication pattern.

### 6.1 General approach

Let us revisit the scenario in Fig. 3: To obtain bounds on  $h_i(t_s)$ , we computed bounds on the difference  $t_s - t_a$  (either directly from  $h_j(t_s) - h_j(t_a)$  or using  $t_s - t_a = t_b - t_a - (t_b - t_s)$ ) and then multiplied them with  $1 - \hat{\rho}_i$  and  $1 + \hat{\rho}_i$ , respectively. This resulted in bounds on  $h_i(t_s) - h_i(t_a)$  which we added to  $h_i(t_a)$  to obtain bounds on  $h_i(t_s)$ . We then analogously computed another set of bounds on  $h_i(t_s)$  from  $t_b - t_s$  and finally chose the best bounds.

In a more complex scenario, there may be various bounds  $\Delta^l, \Delta^u$  on the time differences that we use to compute bounds  $H_i^l(t_s), H_i^u(t_s)$  on the local time  $h_i(t_s)$ . The bounds  $\Delta^l, \Delta^u$  can be illustrated by paths in the event chart as shown in Fig. 5.

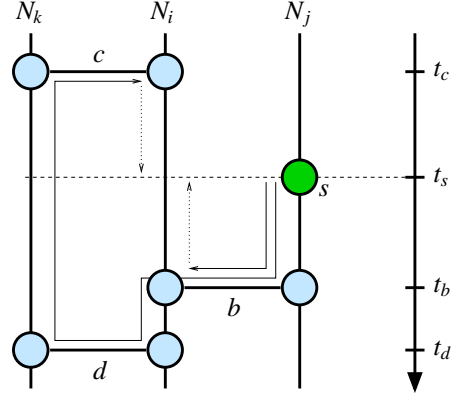


Figure 5: Three nodes  $N_i, N_j, N_k$  with communication events  $b, \dots, d$ . A sensor event  $s$  occurs at node  $N_j$ . The communication events  $c$  and  $d$  may help to find better bounds on  $h_i(t_s)$ . The dotted lines indicate the real-time intervals on which bounds are computed using the paths shown by the corresponding solid lines.

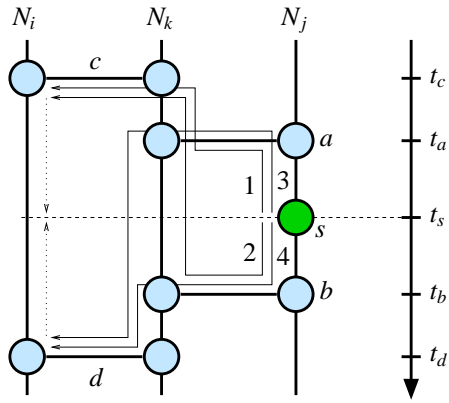
### 6.2 Paths in the event chart

In the scenario depicted in Fig. 5, we can compute bounds on  $h_i(t_s)$  simply by using (6) and (7) from Lemma 5.2. However, we then do not use any information from the events  $c$  and  $d$ . In the ideal case of maximum drift diversity between nodes  $N_i$  and  $N_k$  (i.e.  $\rho_i = \pm \hat{\rho}_i$  and  $\rho_k = \mp \hat{\rho}_k$ ), the uncertainty about  $h_i(t_s)$  can be reduced with this additional information, because  $\bar{\rho}_i(t_c, t_d)$  is known, and hence also  $\bar{\rho}_i(t_s, t_b)$ . We need not make this knowledge explicit by replacing  $(1 \pm \hat{\rho}_i)$  with  $(1 \mp \hat{\rho}_i)$  in (6) and (7), but can rather compute for instance the lower bound  $H_i^l(t_s)$  as  $h_i(t_c) + \Delta^l[c, s](1 - \hat{\rho}_i)$ , where we compute  $\Delta^l[c, s]$  along the path  $s-b-d-c$  in Fig. 5.

The general approach consists in computing the bounds on  $h_i(t_s)$  using all possible paths in the event chart which start at event  $s$  on node  $N_j$  and arrive at some event on node  $N_i$ . This approach is elegant in that all improvements due to drift diversity are “automatically” taken into account by traversing all possible paths and choosing the best, i.e. the one that provides the best bound. The quality of a path or of a path segment depends on how close the assumed drift on it is to the actual drift. If the two are equal, we call the path or path segment *tight*. If a bound on  $h_i(t_s)$  results from a tight path, then the bound is equal to  $h_i(t_s)$ .

### 6.3 General algorithm

The algorithm we now propose chooses the best of all available bounds on time differences to compute bounds on  $h_i(t_s)$  by evaluating all possible paths.



**Figure 6: Illustration of paths in the event chart. Four different paths for computing a lower bound  $H_i^l(t_s)$  on  $h_i(t_s)$  are shown.**

**ALGORITHM 6.1.** Let a communication scenario between  $n$  network nodes  $N_1, \dots, N_n$  be given. Assume that during communication events, the two nodes involved exchange their complete views. Then for any event  $s$  that occurs at time  $t_s$  at node  $N_j$  and not at node  $N_i$ , node  $N_i$  can compute tight bounds  $H_i^l(t_s)$ ,  $H_i^u(t_s)$  on the local time  $h_i(t_s)$  from its maximal (i.e. latest) view as follows:

1. For any event  $e$  at node  $N_i$ ,  $N_i$  computes bounds  $\Delta_i^l[e, s]$ ,  $\Delta_i^u[e, s]$  on  $t_s - t_e$  by traversing all paths between  $s$  and  $e$  and choosing the best among all the resulting bounds. We illustrate this in Fig. 6 for  $\Delta^l[c, s]$  and  $\Delta^u[d, s]$ .
2. Bounds on  $h_i(t_s)$  are computed from the bounds obtained in Step 1: For every event  $e$ , we obtain bounds according to (we use  $[x]^+$  as an abbreviation for  $\max\{x, 0\}$ )

$$H_i^l(t_s) \geq h_i(t_e) + \left[ \Delta_i^l[e, s] \cdot (1 - \hat{\rho}_i) \right]^+ - \left[ -\Delta_i^l[e, s] \cdot (1 + \hat{\rho}_i) \right]^+ \\ H_i^u(t_s) \leq h_i(t_e) + \left[ \Delta_i^u[e, s] \cdot (1 + \hat{\rho}_i) \right]^+ - \left[ -\Delta_i^u[e, s] \cdot (1 - \hat{\rho}_i) \right]^+ .$$

In each of the two expressions above, only one of the maximization operations can yield a value strictly greater than zero (remember that  $\hat{\rho}_i < 1$ ), depending on whether  $e$  precedes or succeeds  $s$ .

3. The final bounds  $H_i^l(t_s)$ ,  $H_i^u(t_s)$  on  $h_i(t_s)$  are given as the maximum of all lower and the minimum of all upper bounds obtained in Step 2.

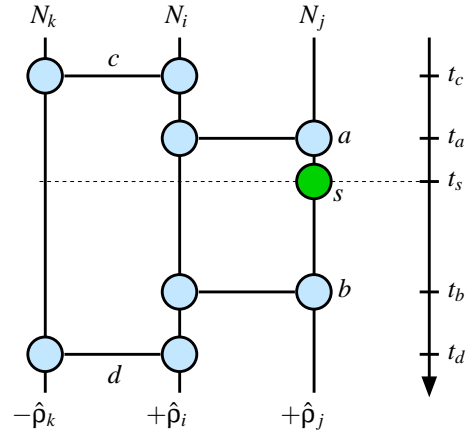
## 6.4 Comparison with Corollary 5.3

We now revisit the numerical example introduced in Sect. 3. To quantify the improvement of using additional paths, we extend the example with an additional node  $N_k$  and events  $c, d$  as shown in Fig. 7. Note that the time axis is not to scale: as before, we have  $t_b - t_a = 3600$ s and  $t_s - t_a = 100$ s. If the drifts are as indicated, then using all paths yields  $H_i^l(t_s) = H_i^u(t_s) = h_i(t_s)$ . If we use only the simple paths from Corollary 5.3, the uncertainty is above 40ms.

Algorithm 6.1 obviously requires a lot of communication, memory, and computation. We present it as the theoretical reference for the maximum synchronization quality that can be achieved. In the next section, we show that the algorithm provides the best possible synchronization.

Different paths for lower bounds on  $h_i(t_s)$ :

$$1: \Delta^l[c, s] \geq \frac{h_j(t_s) - h_j(t_a)}{1 + \hat{\rho}_j} + \frac{h_k(t_a) - h_k(t_c)}{1 + \hat{\rho}_k} \\ 2: \Delta^l[c, s] \geq -\frac{h_j(t_b) - h_j(t_s)}{1 - \hat{\rho}_j} + \frac{h_k(t_b) - h_k(t_c)}{1 + \hat{\rho}_k} \\ 3: \Delta^u[d, s] \leq \frac{h_j(t_s) - h_j(t_a)}{1 + \hat{\rho}_j} - \frac{h_k(t_a) - h_k(t_b)}{1 - \hat{\rho}_k} \\ 4: \Delta^u[d, s] \leq -\frac{h_j(t_b) - h_j(t_s)}{1 + \hat{\rho}_j} - \frac{h_k(t_b) - h_k(t_c)}{1 + \hat{\rho}_k}$$



**Figure 7: Illustration of an extension of the numerical example we introduced in Sect. 3.**

## 7. OPTIMALITY OF ALGORITHM 6.1

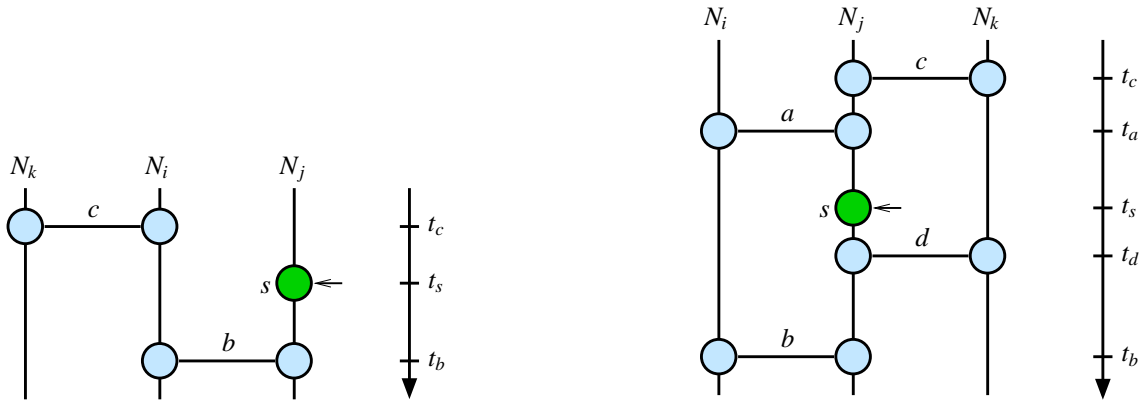
We now show that in our model, no correct, deterministic algorithm can obtain better bounds than those given by Algorithm 6.1.

**THEOREM 7.1 (OPTIMALITY OF ALGORITHM 6.1).** Let a trace  $T$ , a sensor event  $s$  occurring at node  $N_j$ , and another node  $N_i$  be given. Then no correct deterministic algorithm  $A$  exists that provides better bounds on  $h_i(t_s)$  than Algorithm 6.1.

**PROOF.** We prove Theorem 7.1 by contradiction: Assume that given the maximal (i.e. latest) views of all nodes resulting from trace  $T$ , subsumed in a total view  $V$ , an algorithm  $A$  provides the bounds  $\hat{H}_i^l(t_s)$  and  $\hat{H}_i^u(t_s)$  which are correct in  $T$ , i.e.  $\hat{H}_i^l(t_s) \leq h_i(t_s) \leq \hat{H}_i^u(t_s)$ . Further assume that algorithm  $A$  provides better bounds than Algorithm 6.1, i.e. either  $\hat{H}_i^l(t_s) > H_i^l(t_s)$  or  $\hat{H}_i^u(t_s) < H_i^u(t_s)$ , where  $H_i^l(t_s), H_i^u(t_s)$  are the bounds computed by Algorithm 6.1.

As we will show in Sect. 7.1, it is always possible to construct an admissible trace  $T'$  with view  $V$  (i.e.  $T'$  is indistinguishable from  $T$ ) in which  $h_i(t_s) = H_i^l(t_s)$ . Thus, Algorithm 6.1 provides a tight lower bound. Equally, it is always possible to construct an admissible trace  $T''$  with view  $V$  in which  $h_i(t_s) = H_i^u(t_s)$ . Thus, Algorithm 6.1 provides a tight upper bound.

Therefore, algorithm  $A$  is not correct for either the trace  $T'$  (since  $h_i(t_s) = H_i^l(t_s) < \hat{H}_i^l(t_s)$ ) or the trace  $T''$  (since  $h_i(t_s) = H_i^u(t_s) > \hat{H}_i^u(t_s)$ ). This contradicts our initial assumption.  $\square$



**Figure 8:** On the left, there are no cycles in the event chart. It is therefore straightforward that all path segments can be made tight. On the right, several paths form cycles; therefore it is not a priori clear whether clock drifts and event times can always be modified such that  $H_i^l(t_s) = h_i(t_s)$ .

## 7.1 Construction of indistinguishable traces

In the proof of Theorem 7.1, we required that for a given trace  $T$ , it is possible to construct an admissible and indistinguishable trace  $T'$  in which  $h_i(t_s) = H_i^l(t_s)$ . The trace  $T'$  is constructed by making the path leading to the lower bound  $H_i^l(t_s)$  tight; this is achieved by adjusting the drift rates along this path and shifting the real times of the events limiting the path segments. Shifting the real time of a communication event always affects two nodes and might conceivably lead to a violation of (2). We will now show that this cannot happen.

**LEMMA 7.2.** *Let an admissible trace  $T$  with view  $V$  and the sensor event  $s$  occurring at time  $t_s$  at node  $N_j$  be given. For any event  $e$  occurring at node  $N_i$ , let  $\Delta^l[e, s]$  be the lower bound on  $t_s - t_e$  computed by Algorithm 6.1. Then an admissible trace  $T'$  with view  $V'$  can be constructed such that  $V' = V$  and the lower bound  $\Delta^l[e, s]$  is tight in  $T'$ .*

**PROOF.** If  $\Delta^l[e, s] = t_s - t_e$  in trace  $T$ , we set  $T' = T$  and are done. If  $\Delta^l[e, s] < t_s - t_e$ , let  $p_{\text{best}}$  be the best path which is used by Algorithm 6.1 to compute  $\Delta^l[e, s]$ . If there is no other path from  $s$  to  $e$ , then all path segments along  $p_{\text{best}}$  can be made tight without influencing other paths, and we are done.

If there is at least one other path, we have a cycle and another path might be influenced by the tightening of  $p_{\text{best}}$ . Let event  $j$  (for join) be an event on the best path  $p_{\text{best}}$  at which another path  $p_{\text{other}}$  joins. Let event  $f$  (for fork) be the latest event on  $p_{\text{best}}$  before  $j$  which is also on  $p_{\text{other}}$ . In the right-hand chart in Fig. 8, we might for instance have  $e = j = a$ ,  $f = s$ ,  $p_{\text{best}} = s-a$ , and  $p_{\text{other}} = s-d-c-a$ .

Since  $p_{\text{best}}$  is the best path, we have

$$\Delta_{\text{best}}^l[j, f] \geq \Delta_{\text{other}}^l[j, f]. \quad (8)$$

In the trace  $T'$ , the best path shall be tight, i.e.  $\Delta_{\text{best}}^l[j, f] = \Delta[j, f] = t'_f - t'_j$ . We set the times at which events  $f$  and  $j$  occur in  $T'$  to  $t'_f = t_f$  and

$$t'_j = t_f - \Delta_{\text{best}}^l[j, f] \geq t_f - (t_f - t_j) = t_j. \quad (9)$$

To show that  $T'$  is admissible, we have to show that  $\Delta_{\text{other}}^l[j, f]$  is a valid lower bound on  $t_f - t'_j$  in  $T'$ , even though  $t'_j \geq t_j$ . This follows from (8) and (9):  $t_f - t'_j = \Delta_{\text{best}}^l[j, f] \geq \Delta_{\text{other}}^l[j, f]$ .  $\square$

**COROLLARY 7.3.** *The bound  $H_i^l(t_s)$  can be made tight. According to Lemma 7.2, the best path from  $s$  to some event  $e$  on node  $N_i$  can be made tight. Analogously, we can also set  $\rho_i = \pm \hat{\rho}_i$  between events  $s$  and  $e$ , thus making  $H_i^l(t_s)$  tight.*

We will now illustrate Theorem 7.1 with two examples. In the left-hand scenario in Fig. 8, there is only one path; no matter how large and complex the portion of the event chart to the left of event  $c$  at node  $N_i$  is, we can shift  $t_c$  arbitrarily as long as there is only one path from  $s$  to  $N_i$ . This also applies to  $b$  and  $t_b$  here. Thus,  $H_i^l(t_s) = h_i(t_s)$  can always be achieved.

In the right-hand scenario in Fig. 8, there is more than one path from  $s$  to  $N_i$ . The best path used by Algorithm 6.1 forms at least one cycle with some other path. Thus, the real times of the events along the best path cannot be shifted arbitrarily. In the following, we show that they can always be shifted enough to make the best path tight. Let us assume that the lower bound on  $h_i(t_s)$  provided by Algorithm 6.1 is  $H_i^l(t_s) = h_i(t_a) + \Delta^l[a, s](1 - \hat{\rho}_i)$ , and that the path  $s-a$  from which  $\Delta^l[a, s]$  is derived is not tight, i.e.  $\Delta^l[a, s] < t_s - t_a$ . To obtain the trace  $T'$ , we therefore increase  $t_a$  until  $\Delta^l[a, s] = t_s - t_a$ . To keep  $h_j(t_a) - h_j(t_c)$  constant, we need to decrease  $\bar{\rho}_j(t_c, t_a)$ , increase  $t_c$ , or both. Suppose that  $\bar{\rho}_j(t_c, t_a) = -\hat{\rho}_j$  and hence cannot be decreased. We hence have to increase  $t_c$ . To keep  $h_k(t_d) - h_k(t_c)$  constant, we need to increase  $\bar{\rho}_k(t_c, t_d)$ , increase  $t_d$ , or both. Suppose that  $\bar{\rho}_k(t_c, t_d) = +\hat{\rho}_k$  and hence cannot be increased. We hence have to increase  $t_d$ . To keep  $h_j(t_d) - h_j(t_s)$  constant, we need to decrease  $\bar{\rho}_j(t_s, t_d)$  (remember that we cannot modify  $t_s$ ). Suppose that  $\bar{\rho}_j(t_s, t_d) = -\hat{\rho}_j$  and hence cannot be decreased. Now, it would seem that we cannot construct  $T'$  with  $H_i^l(t_s) = h_i(t_s)$ .

Actually, we already have  $T'$ : We assumed

$$\bar{\rho}_j(t_c, t_a) = -\hat{\rho}_j \quad \bar{\rho}_k(t_c, t_d) = +\hat{\rho}_k \quad \bar{\rho}_j(t_s, t_d) = -\hat{\rho}_j.$$

These assumptions provide us with a tight path  $s-d-c-a$  and thus a tight bound

$$\begin{aligned} \Delta^l[a, s] &= -\frac{h_j(t_d) - h_j(t_s)}{1 - \hat{\rho}_j} + \frac{h_j(t_d) - h_j(t_c)}{1 + \hat{\rho}_j} - \frac{h_j(t_a) - h_j(t_c)}{1 - \hat{\rho}_j} \\ &= t_s - t_a. \end{aligned}$$

Obviously, the path  $s-a$  cannot be made tight if there already is another tight path  $s-d-c-a$ . Our initial assumption that  $s-a$  is the best path used by Algorithm 6.1 was therefore wrong.

## 8. CONCLUSION

We modified the algorithm for internal clock synchronization from [10], obtaining better synchronization while reducing computation and without requiring additional communication or memory. We then introduced a new model for internal synchronization and argued why the abstractions the model imposes are reasonable. Using this model, we devised an algorithm which uses all the data that can be obtained for a given communication pattern, at the expense of additional computation, communication and memory. Future work consists in the simulation and quantitative analysis of the gains in synchronization quality and of the additional requirements for a large network; this allows to find a trade-off between synchronization quality and required resources.

## 9. REFERENCES

- [1] Philipp Blum and Lothar Thiele, *Clock synchronization using packet streams*, DISC 2002, Brief Announcements (Dahlia Malkhi, ed.), 2002, pp. 1–8.
- [2] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris, *Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks*, Mobile Computing and Networking, 2001, pp. 85–96.
- [3] IEEE Computer Society LAN MAN Standards Committee, *IEEE 802.11: Wireless LAN medium access control and physical layer specifications*, August 1999.
- [4] Jeremy Elson, Lewis Girod, and Deborah Estrin, *Fine-grained network time synchronization using reference broadcasts*, SIGOPS Oper. Syst. Rev. **36** (2002), no. SI, 147–163.
- [5] Jason Hill and David Culler, *MICA: A wireless platform for deeply embedded networks*, IEEE Micro **22** (2002), no. 6, 12–24.
- [6] Keith Marzullo and Susan Owicki, *Maintaining the time in a distributed system*, Proceedings of the second annual ACM symposium on Principles of distributed computing, ACM Press, 1983, pp. 295–305.
- [7] David Mills, *Bibliography on computer network time synchronization*, available on-line at <http://www.eecis.udel.edu/~mills/bib.html>.
- [8] David L. Mills, *Internet time synchronization: The network time protocol*, IEEE Transactions on Communications **39** (1991), no. 10, 1482–1493.
- [9] Michael Mock, Reiner Frings, Edgar Nett, and Spiro Trikaliotis, *Clock synchronization in wireless local area networks*, Proceedings of the 12th Euromicro Conference on Real Time Systems, June 2000, pp. 183–189.
- [10] Kay Römer, *Time synchronization in ad hoc networks*, Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, ACM Press, 2001, pp. 173–182.
- [11] Ulrich Schmid and Klaus Schossmaier, *Interval-based clock synchronization*, Real-Time Systems **12** (1997), no. 2, 173–228.
- [12] Barbara Simons, Jennifer L. Welch, and Nancy A. Lynch, *An overview of clock synchronization*, Fault-Tolerant Distributed Computing (Barbara B. Simons and Alfred Z. Spector, eds.), Lecture Notes in Computer Science, vol. 448, Springer, 1990, pp. 84–96.
- [13] Hagen Woesner, Jean-Pierre Ebert, Morten Schlager, and Adam Wolisz, *Power saving mechanisms in emerging standards for wireless LANs: The MAC level perspective*, IEEE Personal Communications **5** (1998), no. 3, 40–48.