Improved Interval-Based Clock Synchronization in Sensor Networks

Philipp Blum*
blum@tik.ee.ethz.ch

Lennart Meier†
meier@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

Interval-based synchronization provides the nodes of a distributed system with guaranteed bounds on a common time. This is a crucial piece of infrastructure in many distributed sensing and actuating systems. In this paper, we propose a modification to a known interval-based synchronization algorithm; our new algorithm obtains substantially better results in sensor-network scenarios by taking advantage of the typical drift diversity of the nodes' clocks.

We propose a model for synchronization in ad-hoc, sporadic-communication scenarios. The model allows us to identify the worst and the best case in terms of achievable time uncertainty and to show the worst-case optimality of the discussed algorithms. Simulations show that in the average case, our modification significantly reduces the time uncertainty.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance, Theory.

Keywords

Sensor Networks, Time Synchronization, Clock Drift, Delay Uncertainty.

*The author was supported by the Swiss Commission for Technology and Innovation CTI/KTI, Grant 4957.2.

†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.

IPSN'04, April 26–27, 2004, Berkeley, California, USA.

1. INTRODUCTION

Clock synchronization is an important service in sensor networks. For instance, the fusion of distributed sensor data may require information about the chronology of the sensor observations [10]. In addition, the energy consumption of wireless devices can be reduced by synchronous power-on and shutdown of the communication circuits of a sender–receiver pair [2, 3, 12].

A clock-synchronization service for ad-hoc sensor networks has to meet challenges that are substantially different from those in infrastructure-based networks [5]. *Robustness:* There is no guarantee of stable connectivity between nodes. *Energy efficiency:* Communication, which is needed to achieve and maintain synchronization, is expensive in terms of energy. *Ad-hoc deployment:* The clock-synchronization service must not rely on any a-priori configuration or infrastructure. The conclusion is that well-known algorithms such as NTP [8] are not applicable in ad-hoc sensor networks.

In this paper, we examine local, on-line algorithms which compute an interval in which the current *real time* is contained. The interval is represented by a lower and an upper bound on real time. The goal of each network node is to minimize the size of this interval, i.e. the *time uncertainty*. The paradigm of interval-based synchronization in the context of infrastructure-based networks was proposed by Marzullo and Owicki [7] and refined by Schmid and Schossmaier [11]. An interval-based algorithm for internal synchronization in ad-hoc networks was proposed in [10]. We believe that in ad-hoc *sensor* networks, interval-based synchronization has advantages over using time estimates: (i) Time-stamping single sensor-data items with guaranteed bounds allows to obtain guaranteed bounds from sensor-data fusion. (ii) In hard-real-time applications, nodes can enter a fail-safe state when they realize their time uncertainty has grown excessively large. (iii) The concerted action (sensing, actuating, communicating) of several nodes at a predetermined real time always succeeds: each node can minimize its uptime while guaranteeing its activity at the predetermined real time. (iv) The combination of intervals by intersection is unambiguous and optimal, while the reasonable combination of time estimates requires additional information about the quality of the estimates.

1.1 New Results

We propose a formal system model and problem specification for clock synchronization in ad-hoc sensor networks. We use our model to identify the worst and the best case in terms of achievable time uncertainty and to show the worst-case optimality of the algorithm IM from [7]. We then propose an improved algorithm which also is worst-case-optimal but yields better results in the average case. We show that this improvement is achieved by exploiting the typical drift diversity of the nodes' clocks. Simulation results show that the improvement is substantial for various sensor-network scenarios.

1.2 Related Work

Clock synchronization faces two fundamental problems: The time information of a node degrades over time due to the clock drift, and it cannot be communicated without loss to another node due to unknown message delays. A third problem arises in larger networks, where it is not a priori clear which node's time shall be used as the reference.

Most of the work on synchronization in ad-hoc networks has concentrated on delay uncertainty; recent algorithms reduce it to a few microseconds [1, 4, 6, 9]. They then achieve good synchronization by continued and frequent communication, which keeps the impact of clock drift negligible.

In settings with *sporadic* communication, the impact of clock drift has to be taken into account explicitly. Sporadic communication is typical for ad-hoc sensor networks where nodes may fail, be temporarily out of reach or abstain from communication to save energy. Our interval-based approach is particularly suited for such settings. Note that we do not neglect delay uncertainty: Methods as in [1, 4, 6, 9] can be used to exchange time bounds with negligible delay uncertainty. Delay and drift uncertainty are orthogonal issues.

Interval-based synchronization naturally solves the problem of choosing reference nodes in large networks.

1.3 Overview

First, we define the system model and give the problem statement in Sect. 2. In Sect. 3, we introduce the concept of communication paths and use them to analyze the algorithm IM from [7], showing its worst-case optimality. We then propose and analyze an improved algorithm, the BP-ISA, in Sect. 4. Finally, in Sect. 5 we simulate various ad-hoc sensor networks and show that the BP-ISA substantially outperforms the algorithm IM in terms of time uncertainty.

2. SYSTEM MODEL

In this section, we give a model that captures the aspects of an ad-hoc sensor network which are essential to our analysis. We assume that the network consists of a large number of *sensor nodes* which have a drifting local clock, and a considerably smaller number of *anchor nodes* that have access to time information of constant uncertainty[1]. We refer to nodes that are within each other's transmission range as *neighbors*.

2.1 Scenarios and Executions

2.1.1 Scenario.

To model the interaction of the nodes in a sensor network, we define three types of events: source, communication, and destination events. We call the set of all events in a system, the real times at which they occur, and the time uncertainties associated with source events a *scenario*.

Source events. Any sensor node of the system may receive bounds on the current real time by communicating with a neighboring anchor node. We model this as a source event s which occurs at the sensor node at real time t_s and provides an uncertainty of ΔT_s about the current real time.

Communication events. Any two neighboring sensor nodes may exchange their time bounds by communicating with each other. We model this as a communication event c which occurs at both nodes at real time t_c. The nodes simultaneously acquire mutual knowledge about their time bounds. This is a high-level abstraction. In reality, every communication takes a certain time, and may consist of a whole sequence of messages. In Sect. 5.2, we give quantitative evidence that this abstraction is well justified in sporadic-communication scenarios.

Destination events. A destination event d is an artificial event introduced for analysis purposes. A destination event can be defined for an arbitrary sensor node in the system and with an arbitrary real time t_d. In the analysis, we are then interested in the time bounds of that node at time t_d.

2.1.2 Execution.

A scenario as defined above describes a particular synchronization problem. For a given scenario, the local times of events and the bounds obtained at source events can still be chosen within certain limits. If we fix these parameters, we obtain an *execution*: We call the combination of a scenario, corresponding local times for all events and the lower and upper time bounds[2] for the source events an *execution*.

2.2 Clock Model

Each node of the network has a hardware clock. The reading of this clock at time t is denoted as the node's local time $h(t)$. The *drift* of a clock at time t is defined as the deviation of its speed from the "correct" speed and is thus given by

$$\rho(t) = \mathrm{d}h(t)/\mathrm{d}t - 1 \ . \tag{1}$$

To simplify the notation, we denote the properties of the clock at event e as $h_e = h(t_e)$ and $\rho_e = \rho(t_e)$. Note that at the occurrence of some event e, a node does typically not know the real time t_e or the drift ρ_e of its clock, but only the local time h_e.

The quality of synchronization that can be achieved depends on the assumptions about the nodes' clocks. In this paper, we deal with *drift-constraint clocks*: a constant $\hat{\rho} > 0$ limits the drift of any clock according to

$$-\hat{\rho} \le \rho(t) \le \hat{\rho} \quad \forall t \ . \tag{2}$$

Furthermore, we require $\rho(t) > -1$ for all times t. This means that a local clock can never stop ($\rho(t) = -1$) or run backward ($\rho(t) < -1$).

2.2.1 Admissible execution.

We call an execution *admissible* if it satisfies the constraints on clock drifts as defined in (2) and the real times of source events are contained in the associated time bounds.

3. WORST-CASE ANALYSIS

In [7], an upper bound on the time uncertainty provided by algorithm IM is given for completely connected networks with periodic communication. In this section, we extend this result to scenarios where connectedness and periodic communication are not guaranteed, thus making it applicable to ad-hoc sensor networks. We additionally show that the algorithm IM is worst-case-optimal.

3.1 Interval Synchronization from [7]

The algorithm IM from [7] is given in Algorithm 1. During a communication event, nodes executing the algorithm exchange their current time bounds and intersect the intervals given by their own and by the received bounds.

[1] E.g. via GPS where the uncertainty is smaller than 1 microsecond.

[2] The scenario contains the time uncertainty $\Delta T = T^u - T^l$, but not the bounds T^u and T^l.

Algorithm 1 Algorithm IM from [7]

procedure ***initialize*** [in: - / out: -]
$(T^l_{\mathrm{M}}, T^u_{\mathrm{M}}) \leftarrow (-\infty, \infty)$ // most recent bounds
$h_{\mathrm{M}} \leftarrow 0$ // local time when bounds were obtained

procedure ***updateBounds*** [in: $(T^l_{\mathrm{old}}, T^u_{\mathrm{old}})$,$\Delta h$ / out: (T^l, T^u)]
$(T^l, T^u) \leftarrow \big(T^l_{\mathrm{old}} + \Delta h/(1+\hat{\rho}), T^u_{\mathrm{old}} + \Delta h/(1-\hat{\rho})\big)$

procedure ***currentBounds*** [in: h / out: (T^l, T^u)]
$(T^l, T^u) \leftarrow$ *updateBounds* $\big((T^l_{\mathrm{M}}, T^u_{\mathrm{M}}), h - h_{\mathrm{M}}\big)$

procedure ***intersect*** [in: $(T^l_A, T^u_A), (T^l_B, T^u_B)$ / out: (T^l, T^u)]
$(T^l, T^u) \leftarrow \big(\max(T^l_A, T^l_B), \min(T^u_A, T^u_B)\big)$

procedure ***generateMessage*** [in: h / out: (T^l_c, T^u_c)]
// at local time h
$(T^l_c, T^u_c) \leftarrow$ *currentBounds* (h)

procedure ***processMessage*** [in: $(T^l_c, T^u_c), h$ / out: -]
// at local time h
$(T^l_{\mathrm{M}}, T^u_{\mathrm{M}}) \leftarrow$ *intersect* $\big((T^l_c, T^u_c)$, *currentBounds* $(h)\big)$
$h_{\mathrm{M}} \leftarrow h$

3.2 Path-Based Analysis

For a given scenario and a destination event d in this scenario, our goal is to compute the worst-case uncertainty $\Delta T_d = T^u_d - T^l_d$ provided by the algorithm IM. By the worst case we mean the most unfavourable, admissible execution for a given scenario. In the following, we consider the simple example shown on the left in Fig. 1.

3.2.1 *View.*

For a given scenario and a real time t, the *view* $V_i(t)$ of node N_i is defined as all the information that node N_i could have obtained until time t. If knowledge about an event e is contained in $V_i(t)$, we write $e \in V_i(t)$.

3.2.2 *Timing graph.*

Let an event chart be given. For a destination event d in the event chart, the corresponding *timing graph* is obtained as follows:

1. For each non-communication event $e \in V_i(t_d)$, a vertex e is created. For each communication event $c \in V_i(t_d)$ which occurs at nodes N_i and N_j, two vertices c_i, c_j are created.
2. A directed edge (a, b) from vertex a to vertex b is created in the following cases: (i) The vertices a and b are derived from two different events a, b which both happened at the same network node N such that a precedes b and there is no other event between the two. The weight of such an edge (a, b) represents the increase of uncertainty between the events a and b. It has the value $((1-\hat{\rho})^{-1} - (1+\hat{\rho})^{-1})(h_b - h_a)$. (ii) The vertices a and b are derived from the same communication event. The weight of such an edge is 0.

3.2.3 *Remarks.*

Only the events which can have a causal effect on the given destination event d are contained in the timing graph; these are exactly the events $e \in V_i(t_d)$. The edges in the timing graph correspond to the transformation of the local-state information within a single network node between events, or to the information exchange at communication events. Note that for vertices a, b derived from the same communication event, the timing graph contains both edges (a, b) and (b, a). A path in the timing graph describes a causal dependency between the states of two network nodes. We call such a path a *communication path*. We will now see that the algorithm IM provides the uncertainty at a given destination event by computing a shortest path from some source event to the destination event.

THEOREM 3.1 (UPPER BOUND FOR THE ALGORITHM IM). *Let a network of nodes with drift-constraint clocks employing Algorithm 1 be given. For any destination event d occurring at time t_d, let $\mathcal{S}$ be the set of all source events in the corresponding timing graph. Then the time uncertainty ΔT_d can be bounded by*

$$\Delta T_d \leq \min_{s\in\mathcal{S}} \left\{ \Delta T_s + (t_d - t_s)\left(\frac{2\hat{\rho}}{1-\hat{\rho}}\right) \right\} .$$

PROOF. We use a variant of the algorithm IM defined in Algorithm 1: we let *intersect* return (T^l_A, T^u_A) if $T^u_A - T^l_A < T^u_B - T^l_B$ and (T^l_B, T^u_B) otherwise. This algorithm clearly yields equal or larger values for the uncertainty ΔT_d than the algorithm IM. Therefore, an upper bound on its uncertainty is also an upper bound on the uncertainty of the algorithm IM. This variant is identical to the algorithm MM from [7].

We now associate to each vertex of the timing graph the uncertainty $\Delta T = T^u_{\mathrm{M}} - T^l_{\mathrm{M}}$ after the local-state update. The result of the modified algorithm IM can be interpreted as a shortest-path calculation on the timing graph (the original algorithm IM may use different paths for the lower and the upper bound). The uncertainty ΔT_d is bounded from above by the initial uncertainty ΔT_s at a source event s plus the length of a shortest path from s to d.

We now compute this length. From the clock model in Sect. 2.2 we know that for any two events a and b, the inequality $h_b - h_a \leq (t_b - t_a)(1+\hat{\rho})$ holds. Calculating the path length in the timing graph gives $\Delta T_d \leq \Delta T_s + ((1-\hat{\rho})^{-1} - (1+\hat{\rho})^{-1})(t_d - t_s)(1+\hat{\rho})$. Some algebraic transformations yield the desired expression. □

3.3 Worst-Case Optimality

We now show that the algorithm IM is worst-case-optimal. We first give a universal lower bound on the uncertainty for a given communication scenario and destination event. This lower bound is equal to the upper bound from Theorem 3.1, hence the algorithm IM is worst-case-optimal.

THEOREM 3.2 (LOWER BOUND). *Let a communication scenario in a network of nodes with drift-constraint clocks and a destination event d be given. Let $\mathcal{S}$ be the set of all source events in the corresponding timing graph, and for all $s \in \mathcal{S}$, let ΔT_s be the uncertainty of source event s. Then there exists an admissible execution such that for any correct, deterministic and local clock-synchronization algorithm, the time uncertainty ΔT_d for the destination event d is bounded by*

$$\Delta T_d \geq \min_{s\in\mathcal{S}} \left\{ \Delta T_s + (t_d - t_s)\left(\frac{2\hat{\rho}}{1-\hat{\rho}}\right) \right\} . \tag{3}$$

PROOF. For simplicity, let us first assume that the time uncertainties at all source events are of size 0, i.e. $\Delta T_s = 0$ for all $s \in \mathcal{S}$. In this case, the minimum in (3) is attained by maximizing t_s, i.e. by choosing the latest source event $s \in \mathcal{S}$.

3.3.1 *Proof for $\Delta T_s = 0,\ s \in \mathcal{S}$.*

We derive the admissible execution by letting all clocks run with maximum speed. We now construct a second execution which is indistinguishable from the first, i.e. all views are identical in both executions. The real time at which the destination event d occurs is different in the two executions. The uncertainty of any correct,

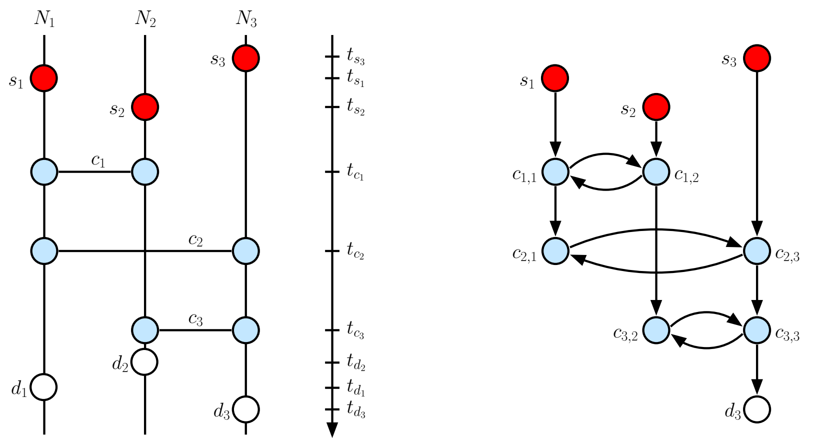

Figure 1: The event chart on the left shows the times of all events and the involved nodes. The timing graph for destination event d_3 is shown on the right: Its vertices correspond to the circles that represent the events of the event chart, and its edges represent the information flow through the local states of the nodes.

deterministic algorithm at event d has therefore to be at least as large as the time difference of event d in the two executions. This is illustrated in Fig. 2: On the left, we have the execution where $\rho = \hat{\rho}$ at all times. On the right, we have the constructed execution, in which from time $t_{\check{s}} = t_{s_1}$ on, all clocks run with minimal speed. Up to time $t_{\check{s}}$, both executions are identical.

For the two executions to be indistinguishable, all events must occur at the same *local* times, while the *real* times differ after time $t_{\check{s}}$. Note how the time intervals after time $t_{\check{s}}$ are stretched in the event chart on the right with respect to the chart on the left.

Any event e with $t_e > t_{\check{s}}$ in the left-hand execution has to occur at such a time $\bar{t}_e$ in the right-hand execution that the local-clock time differences are identical in both executions. In the following, we will use $h_{\check{s}}$ to denote a local clock's reading at time $t_{\check{s}}$. From $h_e - h_{\check{s}} = (t_e - t_{\check{s}})(1+\hat{\rho}) = (\bar{t}_e - t_{\check{s}})(1-\hat{\rho})$, we obtain

$$\bar{t}_e = t_{\check{s}} + (t_e - t_{\check{s}}) \cdot \frac{1+\hat{\rho}}{1-\hat{\rho}}$$

and hence

$$\bar{t}_e - t_e = (t_e - t_{\check{s}}) \cdot \frac{1+\hat{\rho}}{1-\hat{\rho}} - (t_e - t_{\check{s}}) = (t_e - t_{\check{s}}) \left(\frac{2\hat{\rho}}{1-\hat{\rho}} \right)$$

which for $e = d$ proves the claim for $\Delta T_s = 0$.

3.3.2 *Proof sketch for $\Delta T_s \geq 0,\ s \in \mathcal{S}$.*

The proof is essentially the same as the one for $\Delta T_s = 0,\ s \in \mathcal{S}$. For lack of space, we only identify the differences in the following.

To completely specify the executions, we have to give the bounds at source events. For a source event s with uncertainty ΔT_s, we set $T_s^l = t_s$ and $T_s^u = t_s + \Delta T_s$. Now, the event $\check{s}$ for which the minimum in (3) is attained is not necessarily the latest source event. We have to ensure that for all source events s with $t_s \geq t_{\check{s}}$, the time $\bar{t}_s$ at which event s occurs in the constructed execution lies within the time bounds of event s, i.e. that $T_s^l \leq \bar{t}_s \leq T_s^u$. To achieve this, we set $\bar{t}_s = \Delta T_{\check{s}} + t_{\check{s}} + (t_s - t_{\check{s}}) \frac{1+\hat{\rho}}{1-\hat{\rho}}$. Note that in the constructed execution we therefore also have $\bar{t}_{\check{s}} = T_{\check{s}}^u$, and for any event e that occurs at time $t_e \geq t_{\check{s}}$ in the original execution, we obtain

$$\bar{t}_e - t_e = \Delta T_{\check{s}} + (t_e - t_{\check{s}}) \left(\frac{2\hat{\rho}}{1-\hat{\rho}} \right) ,$$

which for $e = d$ proves the claim for $\Delta T_s \geq 0$. $\square$

COROLLARY 3.3. *The lower bound in Theorem 3.2 is tight.*

COROLLARY 3.4. *The algorithm IM is worst-case optimal.*

Corollary 3.3 is a direct consequence of Theorem 3.1: The algorithm IM is guaranteed to achieve a time uncertainty which is not greater than the lower bound given in Theorem 3.2. Corollary 3.4 follows directly from Theorems 3.1 and 3.2 and Corollary 3.3.

3.3.3 *Worst-Case and Best-Case Executions.*

The proofs of Theorem 3.2 and Corollary 3.3 illustrate that the worst-case execution for a given scenario is the one where all clocks run with maximum speed. Analogously, the best case occurs when the drift difference of any two nodes is maximal. Then the intersection of the two time intervals at communication events is minimal.

4. IMPROVED ALGORITHM

In this section, we present and analyze an improved version of the algorithm IM, the Back-Path Interval Synchronization Algorithm (BP-ISA). The BP-ISA is worst-case-optimal like the algorithm IM, but achieves better results in non-worst-case executions.

4.1 Computing Back-Paths

We propose an improvement of the algorithm IM. For each neighbor, nodes store the time bounds of the last communication event with this neighbor. Whenever a node can improve its current bounds through communication, it tries to use them to also improve bounds in its history. At communication events, the nodes exchange their current bounds and the bounds of the previous encounter (if it exists). Each node then uses both bounds to improve all bounds in its history. The use of previous bounds introduces additional paths in the timing graph, as we will see in the following.

We showed that the algorithm IM computes shortest paths in the timing graph corresponding to a scenario. We now extend the timing graph by adding an edge (b, a) for each two vertices a and b that are derived from two different events which both happened at the same network node N such that a precedes b and there is no other event between the two. In the timing graph in Fig. 1, each edge directed downwards is complemented by an inverse edge. The new edges may result in new and possibly shorter paths from a source event to a destination event. By considering these paths, the BP-ISA can achieve better time uncertainties than the algorithm IM.

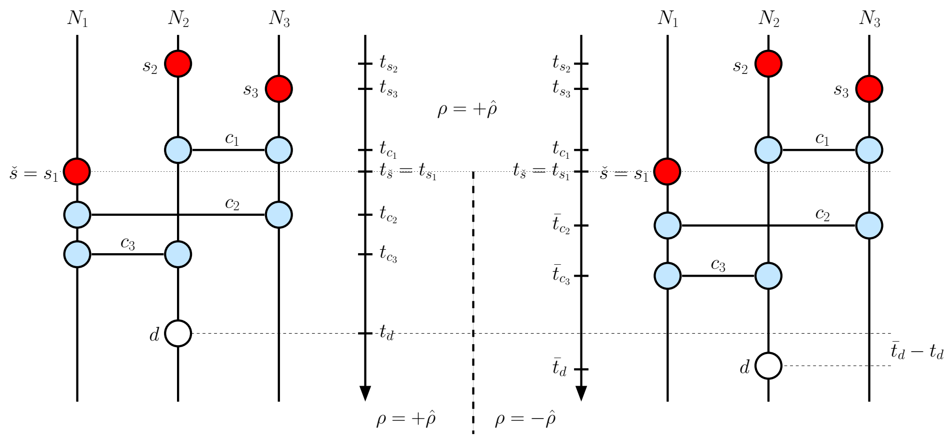

Figure 2: Two indistinguishable executions. On the left, $\rho = \hat{\rho}$ at all times. After time $t_{\check{s}}$, the clocks in the constructed execution on the right run with minimal speed. As can be seen, the event $\check{s}$ is the latest source event from which there is a path to d. Since the two executions are indistinguishable from each other, the time uncertainty at event d has to be at least $\bar{t}_d - t_d$.

Algorithm 2 BP-ISA

---

procedure ***initialize*** [in: - / out: -]
// $N =$ maximum number of neighbor nodes
  for $\forall i \in \{1, \ldots, N\}$ **do**
    $(T^l_{\mathrm{M}}[i], T^u_{\mathrm{M}}[i]) \leftarrow (-\infty, \infty)$ // most recent bounds from node with index i
    $h_{\mathrm{M}}[i] \leftarrow 0$ // local time when bounds were obtained
  end for
  $n_{\mathrm{M}} \leftarrow 1$ // $n_{\mathrm{M}} =$ index of last node encountered

---

procedure ***updateBounds*** [in: $(T^l_{\mathrm{old}}, T^u_{\mathrm{old}}), \Delta h$ / out: (T^l, T^u)]
  if $\Delta h \geq 0$ **then**
    $(T^l, T^u) \leftarrow \big(T^l_{\mathrm{old}} + \Delta h/(1+\hat{\rho}), T^u_{\mathrm{old}} + \Delta h/(1-\hat{\rho})\big)$
  else
    $(T^l, T^u) \leftarrow \big(T^l_{\mathrm{old}} + \Delta h/(1-\hat{\rho}), T^u_{\mathrm{old}} + \Delta h/(1+\hat{\rho})\big)$
  end if

procedure ***currentBounds*** [in: h / out: (T^l, T^u)]
  $(T^l, T^u) \leftarrow$ *updateBounds* $\big((T^l_{\mathrm{M}}[n_{\mathrm{M}}], T^u_{\mathrm{M}}[n_{\mathrm{M}}]), h - h_{\mathrm{M}}[n_{\mathrm{M}}]\big)$

procedure ***intersect*** [in: $(T^l_A, T^u_A), (T^l_B, T^u_B)$ / out: (T^l, T^u)]
  $(T^l, T^u) \leftarrow \big(\max(T^l_A, T^l_B), \min(T^u_A, T^u_B)\big)$

procedure ***updateMemory*** [in: $(T^l, T^u), h$ / out: -]
  for $\forall i \in \{1, \ldots, N\}$ **do**
    $(T^l_{\mathrm{new}}, T^u_{\mathrm{new}}) \leftarrow$ *updateBounds* $\big((T^l, T^u), h_{\mathrm{M}}[i] - h\big)$
    $(T^l_{\mathrm{M}}[i], T^u_{\mathrm{M}}[i]) \leftarrow$ *intersect* $\big((T^l_{\mathrm{M}}[i], T^u_{\mathrm{M}}[i]), (T^l_{\mathrm{new}}, T^u_{\mathrm{new}})\big)$
  end for

---

procedure ***generateMessage*** [in: h, n / out: $(T^l_c, T^u_c), (T^l_p, T^u_p)$]
// to node n at local time h
  $\big((T^l_c, T^u_c), (T^l_p, T^u_p)\big) \leftarrow \big($*currentBounds* (h) , $(T^l_{\mathrm{M}}[n], T^u_{\mathrm{M}}[n])\big)$

procedure ***processMessage*** [in: $(T^l_c, T^u_c), (T^l_p, T^u_p), h, n$ / out: -]
// from node n at local time h
  updateMemory $\big((T^l_p, T^u_p), h_{\mathrm{M}}[n]\big)$
  $(T^l_{\mathrm{M}}[n], T^u_{\mathrm{M}}[n]) \leftarrow$ *intersect* $\big((T^l_c, T^u_c),$ *currentBounds* $(h)\big)$
  $h_{\mathrm{M}}[n] \leftarrow h$
  $n_{\mathrm{M}} \leftarrow n$
  updateMemory $($*currentBounds*$(h), h)$

4.2 Comparison with the Algorithm IM

The BP-ISA has additional state information: The arrays $T^l_{\mathrm{M}}[N]$, $T^u_{\mathrm{M}}[N]$, and $h_{\mathrm{M}}[N]$ store the bounds on real time and the local time for the last communication event with every neighbor node.

The procedure *updateBounds* can also compute bounds for past events, i.e. for a negative Δh. Note the change of the sign in the terms $\Delta h/(1 \pm \hat{\rho})$: While the lower bound is increased at *minimal speed* $(1 + \hat{\rho})$ to guarantee validity in the future, it is decreased at *maximal speed* $(1 - \hat{\rho})$ in order to guarantee validity in the past.

The procedure *currentBounds* computes bounds based on the stored bounds for the most recent communication event, which occurred with the remote node with index n_{M}.

The procedure *generateMessage* prepares two sets of bounds: This node's current bounds (T^l_c, T^u_c) and the bounds (T^l_p, T^u_p) for the previous communication event with the remote node identified by the index n.

The procedure *processMessage* has additional parameters: T^l_p and T^u_p represent the time bounds the remote node has in memory for the last communication event with this node. The value n is used to identify the remote node and to access the stored values $(T^l_{\mathrm{M}}[n], T^u_{\mathrm{M}}[n])$ and $h_{\mathrm{M}}[n]$.

4.2.1 *Example.*

To explain how the BP-ISA works, we use the simple scenario shown in the upper left corner of Fig. 3. Two nodes N_1 and N_2 communicate at events a and c. Node N_1 has an additional communication b with a third node that is not shown.

At event a, nodes N_1 and N_2 exchange their current and previous bounds. Let us assume that they have never communicated before and therefore the previous bounds are uninitialized $(-\infty, \infty)$. Therefore the first call to *updateMemory* in *processMessage* has no effect on the stored bounds. The nodes combine the received current bounds with their own current bounds and store the result in $(T^l_{\mathrm{M}}[2], T^u_{\mathrm{M}}[2])$ on N_1 and in $(T^l_{\mathrm{M}}[1], T^u_{\mathrm{M}}[1])$ on N_2. The algorithm IM does exactly the same (using $(T^l_{\mathrm{M}}, T^u_{\mathrm{M}})$).

At event b, node N_1 communicates with some other node. We assume that N_1 can improve its current bounds by using the remote node's current bounds. The final call to *updateMemory* thus may improve the stored bounds $(T^l_{\mathrm{M}}[2], T^u_{\mathrm{M}}[2])$ for the communication event a with node N_2. Such an improvement constitutes the computation of the first part (from b to a) of the back-path displayed in Fig. 3 as a dashed arrow.

At event c, N_1 and N_2 exchange their current bounds and their previous bounds for event a. While the previous bounds of N_2 have not been modified since event a, those of N_1 have been improved. When node N_2 thus calls *updateMemory* and then *currentBounds*, the remaining part of the back-path is computed.

4.2.2 Computation and memory requirements.

The BP-ISA is more complex than the algorithm IM. The computational cost of the BP-ISA per communication event is approximately $(2N + 1)$ times the cost of the algorithm IM, where N is the number of neighbors a node communicates with. The cost of the BP-ISA in terms of memory is N times that of the algorithm IM.

4.3 Analysis

We show that the BP-ISA performs at least as well as the algorithm IM and is thus also worst-case optimal. By means of a simple scenario, we illustrate how and when the BP-ISA can provide better synchronization than the algorithm IM.

LEMMA 4.1. *The BP-ISA always provides equal or better time bounds than the algorithm IM.*

PROOF. We remove the calls to *updateMemory* from the procedure *processMessage* of the BP-ISA. Now, $T^l_{\mathrm{M}}[n_{\mathrm{M}}]$ takes the role of T^l_{M} in the algorithm IM, $T^u_{\mathrm{M}}[n_{\mathrm{M}}]$ that of T^u_{M} and $h_{\mathrm{M}}[n_{\mathrm{M}}]$ that of h_{M}. It is clear that this modified BP-ISA is equivalent to the algorithm IM.

We now only have to show that the call to *updateMemory* cannot degrade bounds. This procedure modifies the stored bounds only by intersection of the corresponding intervals. It is clear that this never degrades bounds. □

COROLLARY 4.2. *The BP-ISA is worst-case-optimal.*

4.3.1 Improvement over the algorithm IM.

The amount of improvement of the BP-ISA is illustrated in Fig. 3. It depends on many parameters: the nodes' drift rates ρ_1 and ρ_2, the position of event b and the uncertainty achieved at b in comparison to the uncertainty ΔT at a. These four parameters are explored in the five graphs in Fig. 3.

In the upper right corner, event b occurs immediately after a ($x \approx 0$), and it provides zero uncertainty ($y = 0$). In this case, the improvement of the BP-ISA is 100% if the nodes' rate-difference is maximal. The improvement is 0% if the drift rates are equal.

The graphs in the center row of Fig. 3 explore the effect of the position of event b. The average improvement decreases with increasing distance between a and b. The maximal improvement is still 100% for maximal drift-rate difference.

The graphs in the bottom row of Fig. 3 explore the effect of the time uncertainty achieved at b. With increasing uncertainty, the maximal improvement decreases, e.g. at $y = 0.5$, the BP-ISA can achieve 34% less uncertainty than the algorithm IM if the drift-rate difference is maximal.

We thus conclude that depending on the scenario and the drift rates, the improvement of the BP-ISA varies between 0% and 100%. In the next section, we examine the *average* improvement for realistic sensor-network scenarios through simulation.

5. SIMULATION

In this section, we describe the scenarios that we simulated and the performance measures used to evaluate the synchronization algorithms. We examine the absolute performance of the algorithm IM and the BP-ISA in these scenarios and quantify the relative improvement of the BP-ISA.

The simulations were performed by a dedicated program written in C++.

5.1 Scenarios and Measures

We simulated sensor-network scenarios for a simulation time of 500 hours, i.e. almost 21 days. The network contained 100 sensor nodes that were distributed uniformly at random in a square area. Their local clocks had a constant drift rate between -100ppm and $+100$ppm. The number N_A of anchor nodes was either $5, 10$ or 20. Two nodes could communicate if they were within each other's transmission range. All nodes had the same transmission range. It was varied between 0.1 and 0.5 times the width of the area in which the nodes were placed. Sensor nodes communicated with other sensor nodes at a variable average frequency of $f_C \in [1/\mathrm{h}, 20/\mathrm{h}]$.[3] Anchor nodes communicated with sensor nodes at a much smaller average frequency $f_A \in [0.002/\mathrm{h}, 2/\mathrm{h}]$.

The measure for the performance of an interval-based synchronization algorithm was the time uncertainty. We evaluated it after every communication event. Then we took the average over all communication events, but excluded those events after which the nodes had an infinite time uncertainty, i.e. the events from which there existed no path to a source event in the corresponding timing graph. We thus produced the average time uncertainties for the algorithm IM and the BP-ISA, as shown in Fig. 4. From these values, we computed the improvement of the BP-ISA. Figures 5 and 6 give these results in % and milliseconds. For every data point, at least 50 scenarios were evaluated.

5.2 Discussion of the Results

5.2.1 Absolute performance.

Figure 4 displays the time uncertainty achieved by the two algorithms. As expected, the uncertainty decreases when communication becomes more frequent. Increasing the communication among sensor nodes decreases the uncertainty to a constant level, while increasing the anchor communication frequency f_A leads to arbitrarily small uncertainties.

Figure 4 shows that for all parameters used in this study, the uncertainty remains in the order of milliseconds. As shown e.g. in [1, 4, 6, 9], the delay-induced uncertainty can be reduced to a few microseconds. Therefore our assumption that the delay-induced uncertainty is negligible in comparison to the drift-induced uncertainty is correct in ad-hoc, sporadic-communication networks.

5.2.2 Relative improvement of the BP-ISA.

Figure 5 displays the relative improvement of the BP-ISA for a variable transmission range. The improvement increases significantly up to a transmission-range/area-width ratio of ≈ 0.15 and remains approximately constant at larger values. Figure 7 explains why. With a small transmission range, sensor nodes are divided into two categories: (i) Nodes that have no communication path to an anchor node. Such nodes have an infinite uncertainty and are not included in the average measure. (ii) Nodes that have a communication path to an anchor node and thus have a finite uncertainty. Such nodes are typically very few hops distant from an anchor node. Their uncertainty therefore is small and the BP-ISA cannot improve much. Also for large transmission ranges, all sensor nodes are very few hops distant from an anchor node.

[3] For $f_C = 10/\mathrm{h}$, 5000 communication events occured at uniformly random times.

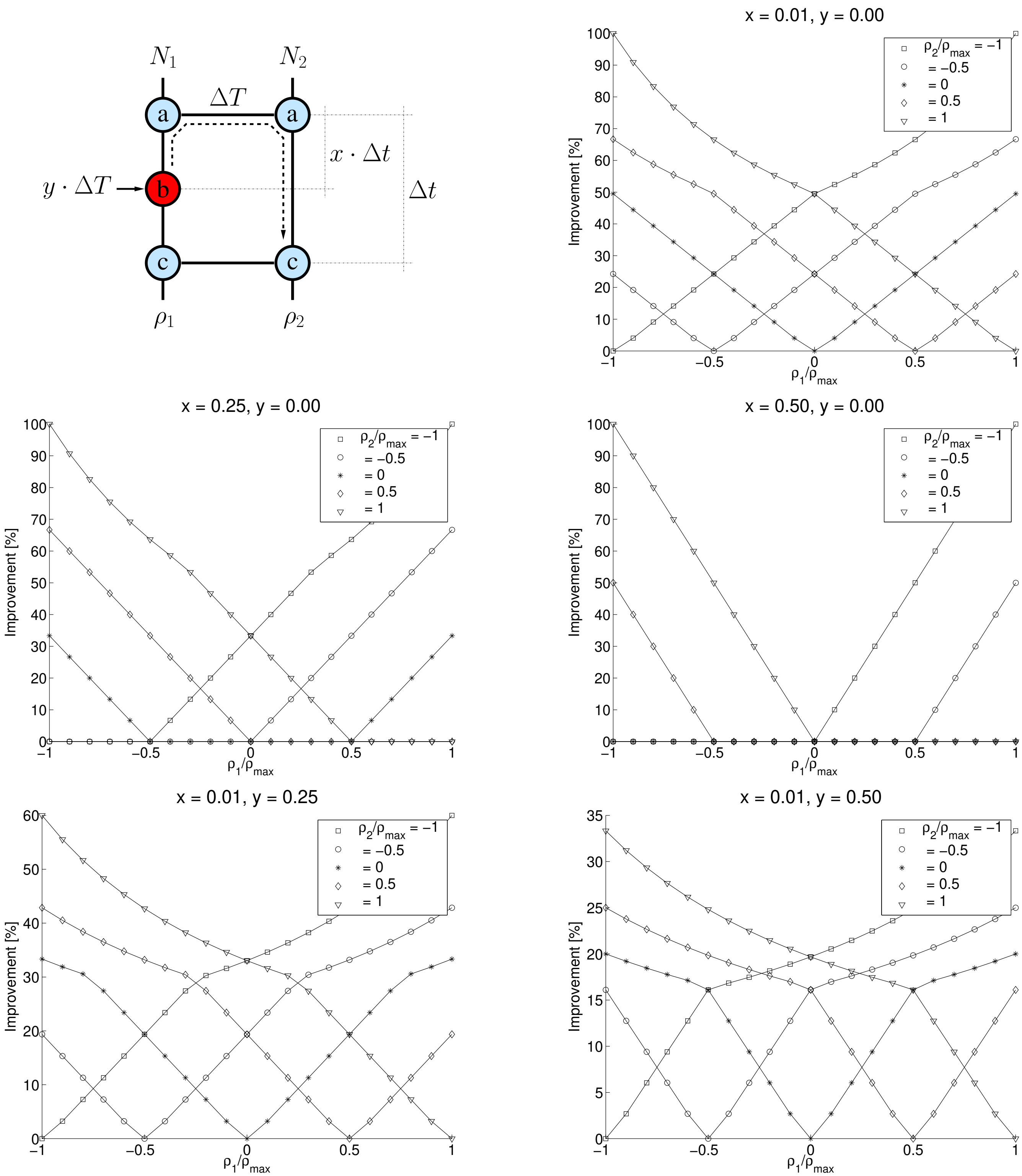

Figure 3: ***Improvement of BP-ISA in the simple scenario depicted in the upper left corner.*** **In all figures, the drift rates of both nodes are varied between $-\hat{\rho}$ and $\hat{\rho}$. Each graph shows one combination of the parameters x and y. The parameter x determines the position of event b, the parameter y determines the time uncertainty achieved in this event. The figure in the upper right corner shows that BP-ISA can achieve 100% improvement if event b provides uncertainty zero and occurs immediately after event a. In the center-row graphs, the position of event b is varied. The bottom-row graphs show varying uncertainty of this event.**

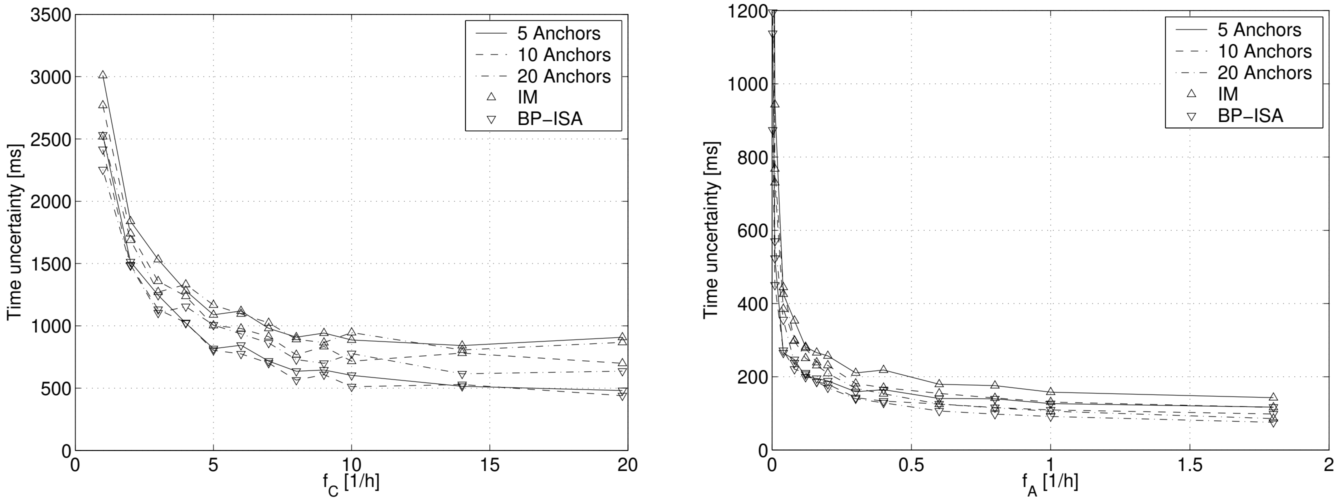

Figure 4: ***Time uncertainty achieved by the algorithm IM and the BP-ISA.*** **Left: For varying frequencies of communication among sensor nodes. Right: For varying frequencies of communication with anchor nodes. The time uncertainty decreases when the sensor nodes communicate more frequently. The effect of the communications with anchor nodes is much stronger.**

The largest improvement of the BP-ISA is obtained when many nodes have a path to an anchor, and the average path length in terms of hops is large. This is the case for a transmission-range/area-width ratio of ≈ 0.15, which is the range at which almost all nodes are connected.

Figure 6 displays the improvement as a function of sensor and anchor communication frequencies. The improvement of the BP-ISA increases when the sensor nodes communicate more frequently among each other. The improvement decreases if the sensor nodes communicate often with the anchor nodes. The improvement also decreases with increasing number of anchor nodes.

5.2.3 *Remarks.*

The simulation results show that the BP-ISA provides substantial improvements over the algorithm IM, and that these improvements are large in comparison to the delay-induced time uncertainty. The amount of the improvement is strongly dependent on the scenario: The improvements can achieve several tens of percents if the sensor nodes communicate relatively often among each other, but only rarely with anchor nodes. The BP-ISA performs best in networks where the average length of paths to anchor nodes is large. For a given node density, this is the case at those small transmission ranges that lead to (almost) connected networks.

Note that the time uncertainty of interval-based algorithms is not directly comparable to the time error of algorithms that compute time estimates. In the average case, the error of an estimate computed as the average of the lower and the upper bound will be much lower than the *guaranteed* uncertainty. Also, the microsecond-range errors reported in [1, 4, 6, 9] are achieved in networks with frequent communication.

6. CONCLUSION

We proposed a model for synchronization in ad-hoc, sporadic-communication scenarios and used it to identify the worst and the best case in terms of achievable time uncertainty. We showed that the algorithm IM from [7] is worst-case-optimal and proposed an improved algorithm which additionally takes advantage of the typical drift diversity of the nodes' clocks.

We illustrated by simulation that in the average case, our algorithm significantly outperforms the algorithm IM from [7] at a moderate increase in computation and memory cost. The conceivably optimal synchronization would be obtained by exchanging and accumulating all available information. Open questions are how much better this approach would be and what pieces of information may be discarded without loss.

An orthogonal improvement can be achieved by an extension of the clock model: Bounds on the variation $\mathrm{d}\rho(t)/\mathrm{d}t$ of the drift (as proposed in [11]) allow to estimate and compensate the current drift. Back-paths can improve synchronization also here.

We expect node mobility to further reduce time uncertainty due to better connectivity. The influence of mobility on the relative improvement of the BP-ISA remains to be examined. It is unclear whether the positive effects (e.g., increased connectivity and drift diversity) or the negative effects (e.g., increased time between repeated encounters of the same two nodes) dominate.

Computing the back-paths as introduced in this paper can also be applied to internal synchronization.

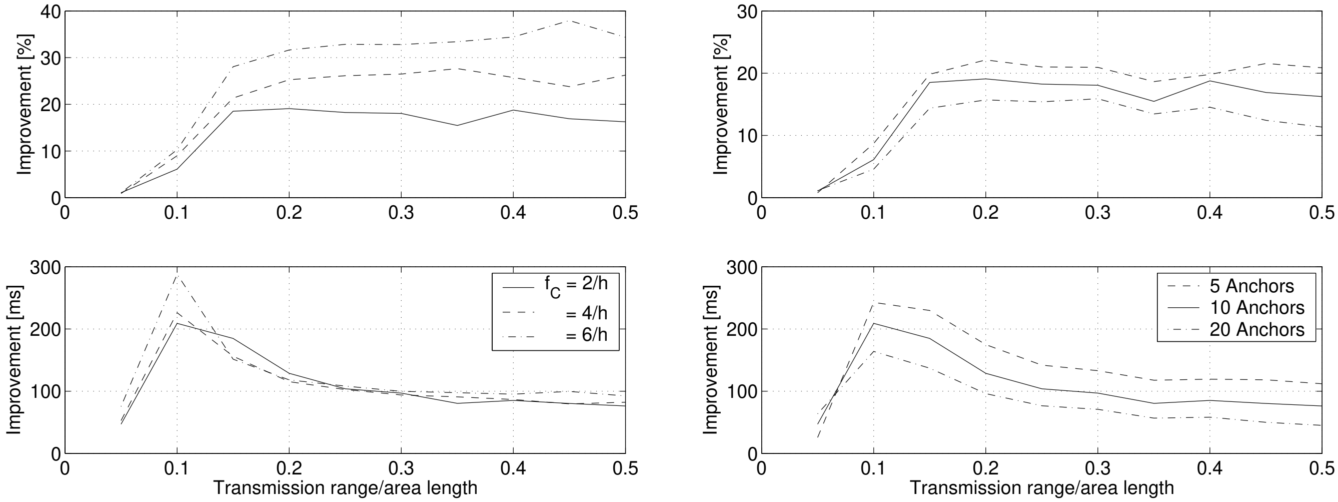

Figure 5: ***Improvement of the BP-ISA for varying transmission ranges.*** **Left: For varying communication frequencies f_C. Right: For varying numbers of anchor nodes. Note that the solid lines on the left and right represent the same data.**

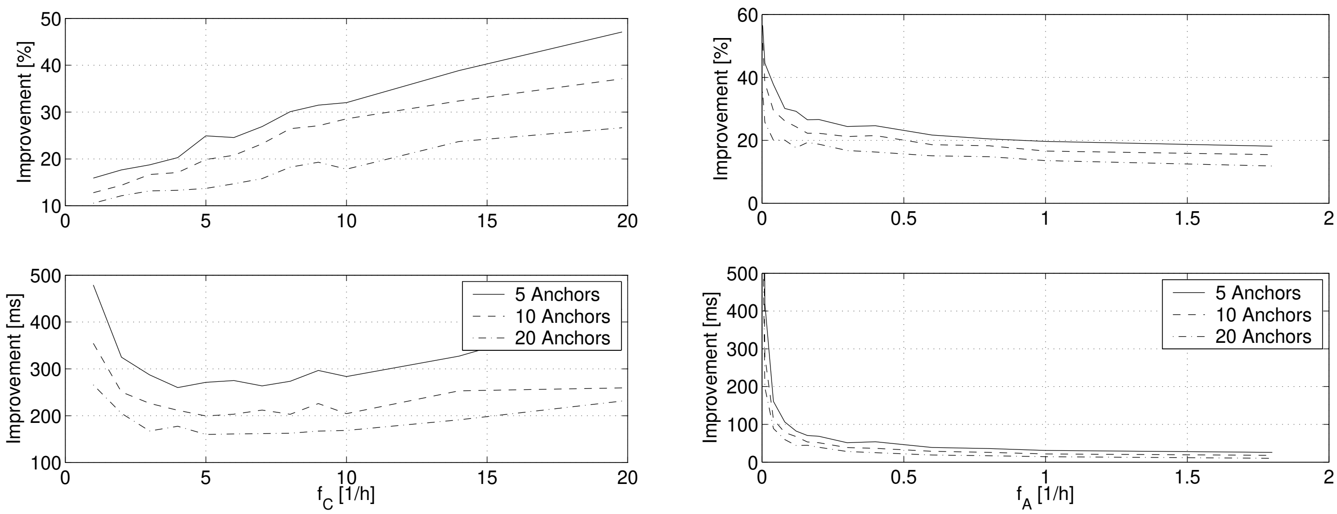

Figure 6: ***Improvement of the BP-ISA for varying communication frequencies.*** **Left: With increasing number of communications among the sensor nodes, the improvement of the BP-ISA grows. Right: With increasing number of communications with anchor nodes, the improvement of the BP-ISA diminishes.**

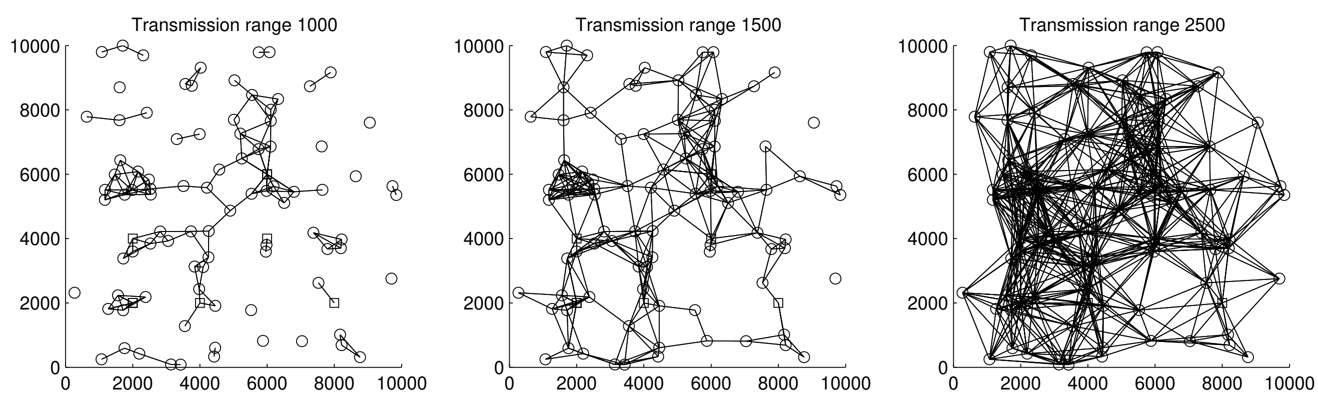

Figure 7: ***Illustration of the transmission range.*** **Anchor nodes are shown as squares, sensor nodes as circles. Left: A small transmission range leads to many partitions in the network. Middle: At a transmission range of 0.15 times the width of the area, almost all nodes are connected. As shown in Fig. 5, the improvement of the BP-ISA is particularly high for such transmission ranges (in milliseconds). Right: At higher transmission ranges, most nodes can communicate with an anchor node.**

7. 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] Jeremy Elson and Kay Römer, *Wireless sensor networks: A new regime for time synchronization*, Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, New Jersey, October 2002.

[6] Jason Hill and David Culler, *MICA: A wireless platform for deeply embedded networks*, IEEE Micro **22** (2002), no. 6, 12–24.

[7] 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.

[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] 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.