Fast Distributed Multi-hop Relative Time Synchronization Protocol and Estimators for Wireless Sensor Networks

Djamel Djenouri[1,*], Nassima Merabtine[2]
Fatma Zohra Mekahlia[2] Messaoud Doudou[1],

1: *CERIST Center of Research, Ben-aknoun, BP 143, Algiers 16030, Algeria.*
2: *Saad Dahlab University, Blida, Algeria*
E-mails: ddjenouri@acm.org, nassimane@gmail.com, mekahlia.fzohra@yahoo.fr, doudou@mail.cerist.dz
Corresponding author(*), Tel: +213 (0)554 689372, Fax: +213 (0)21 912126

Abstract

The challenging problem of time synchronization in wireless sensor networks is considered in this paper, where a new distributed protocol is proposed for both local and multi-hop synchronization. The receiver-to-receiver paradigm is used, which has the advantage of reducing the time-critical-path and thus improving the accuracy compared to common sender-to-receiver protocols. The protocol is fully distributed and does not rely on any fixed reference. The role of the reference is divided amongst all nodes, while timestamp exchange is integrated with synchronization signals (beacons). This enables fast acquisition of timestamps that are used as samples to estimate relative synchronization parameters. An appropriate model is used to derive maximum likelihood estimators (MLE) and the Cramer-Rao lower bounds (CRLB) for both the offset-only, and the joint offset/skew estimation. The model permits to directly estimating relative parameters without using or referring to a reference' clock. The proposed protocol is extended to multi-hop environment, where local synchronization is performed proactively and the resulted estimates are transferred to the intermediate/end-point nodes on-demand, i.e. as soon as a multi-hop communication that needs synchronization is initiated. On-demand synchronization is targeted for multi-hop synchronization instead of the always-on global synchronization model, which avoids periodic and continuous propagation of synchronization signals beyond a single-hop. Extension of local MLE estimators is proposed to derive relative multi-hop estimators. The protocol is compared by simulation to some state-of-the-art protocols, and results show much faster convergence of the proposed protocol. The difference has been on the order of more than twice compared to CS-MNS, more than ten times compared to RBS, and more than twenty times compared to TPSN. Results also show scalability of the proposed protocol concerning the multi-hop synchronization. The error does not exceed few microseconds for as much as 10 hops in R4Syn, while in CS-MNS, and TPSN, it reaches few tens of microseconds. Implementation and tests of the protocol on real sensor motes confirm microsecond level precision even in multi-hop scenarios, and high stability (long lifetime) of the skew/offset model.

Key words: wireless sensor networks, time synchronization, distributed algorithms.

1 Introduction

Time synchronization has always been one of the fundamental problems in distributed systems. The lack of a common clock and a shared memory makes message-exchange-based algorithms the only way to ensure synchronization. However, delays of exchanged messages are not constant, which raises the need of appropriate and accurate estimators. The problem is more complex in wireless networks, where communication may be prone to high delay variability. In wireless sensor networks (WSN), node limitations (computation, memory, energy) add more constraints to the problem, while fine-grained synchronization remains essential for many applications and protocols in such networks; e.g., data fusion/aggregation, TDMA scheduling, duty- cycle coordination, realtime monitoring/actuation, etc. Several synchronization protocols for WSN have been proposed in the literature; they can be classified in different ways [1], e.g. single-hop vs. multi-hop, clock correction vs. relative synchronization, sender-to-receiver vs. receiver-to-receiver, etc. We consider in this paper relative on-demand synchronization, instead of the global always-on synchronization where nodes are continuously kept tuned to a reference clock (wall-clock model). The latter requires high overhead and periodic propagation of synchronization messages through the network [2]. This is in addition to the problem of the single-point of failure (reference).

The receiver-to-receiver approach applied to WSN by the Reference Broadcast Synchronization (RBS) protocol [3] exploits the broadcast property of the wireless communication medium; The broadcast medium allows receivers located within listening distance from the same sender to receive a broadcast message at approximately the same time, with little variability due to the reception timestamping at the receivers. RBS uses a sequence of synchronization messages (beacons) from a fixed sender (reference), which allows its neighboring nodes receiving the beacons to estimate relative synchronization parameters that are used for time conversion. Detailed description of RBS will be given in the next section. It is important here to mention that RBS exploits the concept of time-critical path, which is defined as the path of a message that contributes to non-deterministic errors in a protocol [2].

With any sender-to-receiver protocol, the time involved in sending a message from a sender to a receiver is the result of the following four factors that can vary non-deterministically: i) Send time, which is the time spent by the sender for message construction along with the time spent to transmit the message from the sender's host to the network interface. ii) Access time; the time spent for medium access at the MAC layer. iii) Propagation time, as the time for the message to reach the receiver once it has left the sender's radio. iv) Receive time, which is the time spent by the receiver to process the message. RBS removes the send time and the access time from the critical path [1, P. 294], which represent the two largest sources of non-determinism. This provides a high degree of synchronization accuracy. Still, timestamping may witness some variability due to the reception delays at the receivers. Appropriate estimators are then essential. RBS can also be extended to multi-hop networks by simply applying a series of conversion through the active route. The major drawback of RBS is the need of/ dependency on a fixed reference. This centralization of the reference might be inappropriate for some self-organized WSN applications. RBS also requires important steps (time) to acquire samples, as each timestamping must be separately exchanged between synchronizing nodes. To tackle these issues, a new synchronization protocol for WSN is proposed in this paper. The protocol takes advantage of the receiver-to-receiver approach, while being totally decentralized. It distributes the role of the reference amongst all nodes and integrates beaconing and timestamp exchange in a single step. Local synchronization between direct neighbors is performed proactively, but the multi-hop extension operates on-demand according to the application need. This is motivated by the gradient property considering that the local synchronization (between neighboring nodes) to be more important than global one for most applications [4].

Models that are appropriate to relative receiver-to-receiver synchronization are defined. These models permit to derive maximum likelihood estimators (MLE) and the corresponding Cramer-Rao lower bounds (CRLB). The estimators are numerically analyzed and compared with the appropriate CRLB, and the protocol is compared with existing solutions by simulation. It is also implemented on real sensor motes and empirically evaluated in single-hop, as well as multi-hop scenarios.

The rest of the paper is organized as follows. The next section introduces background concepts and the network model. The solution description is provided in Section 3, and the simulation study in Section 4. Implementation of the protocol on sensor motes and empirical results are described in Section 5. Section 6 summarizes the related work, and finally, Section 7 draws conclusions.

2 Background and Network Model

Like any computing device, a sensor mote is equipped with a hardware oscillator that is used to implement the internal clock. Let $C(t)$ denotes the value of the internal clock at time t. The oscillator frequency determines the rate at which the clock runs. The rate of an ideal clock, $\partial C/\partial t$, would equal 1. However, such a clock is not available in practice, and all clocks are subject to clock drift; i.e. oscillator frequency will vary unpredictably due to various physical effects. Despite clock drifting over time, its value can be approximated using appropriate estimators. Two types of models can be used for the estimation; the offset-only vs. joint offset/skew model. The former has the advantage of deriving simple and easy-to-implement estimators but does not ensure long-term estimation, and thus beacons need to be exchanged at a high frequencies to assure good precision. The second model provides long-term estimators and allow for low frequency beaconing once the estimators converge, which is power-efficient. Nonetheless, calculation complexity of the estimators is relatively high compared to the first model. Both models are considered in this paper.

2.1 Offset-only Model

For some node in the network, say n_i, the local clock can be approximated by [3]:

$$C_i(t) = t + \theta_i, \tag{1}$$

where θ_i is the absolute offset of node i's clock [1].

Relative equation relating two nodes' clocks, C_i and C_j, can be given by:

$$C_j(t) = C_i(t) + \theta_{n_i \to n_j}, \tag{2}$$

where $\theta_{n_i \to n_j}$ represents the *relative* offset relating time at node, n_i, to the corresponding one at node, n_j.

The purpose of any relative synchronization protocol is to efficiently and accurately estimate these parameters that varies over time. Relative synchronization enables nodes to run their local clocks independently and use the estimated values to convert time whenever needed. This is conceptually different from clock update protocols, which define distributed mechanisms for

[1] absolute offset denotes the offset to realtime

nodes to update their clocks and converge to common values. This update generally involve jump/freeze of the clock value, which may affect correctness of local events' timestamping. Relative synchronization does not suffer from such a problem. Further, conversion to a reference (global) time is still possible with relative synchronization, provided that one of the nodes executing the protocol uses/provides such a time. RBS uses relative synchronization and implements the receiver-to-receiver synchronization. A sequence of synchronization messages (beacons) are periodically transmitted from a given and fixed sender– termed reference– and intercepted by all synchronizing nodes. The nodes timestamp the arrival-time of each beacon using their local clocks, then every pair of nodes exchange the recorded timestamps to construct samples. That is, if the i^{th} beacon is timestamped, u_i, by node, n_1, and v_i, by node, n_2, then the pair (u_i, v_i) forms a sample. These samples are used to estimate the relative offset and skew between nodes n_1 and n_2.

As mentioned in Section 1, RBS reduces the time-critical path compared to the send-to-receiver protocols by only considering the instants at which a message reaches different receivers. This eliminates the send-time and access-time, and it provides a high degree of synchronization accuracy [3], [1]. For this reason, the proposed protocol uses the RBS receiver-to-receiver concept. The protocol also allows for multi-hop relative synchronization extension, where no synchronization signals are exchanged beyond a single-hop, contrary to tree-based single reference protocols (e.g. FTSP [5]). Intermediate routers can perform conversion in a hop-by-hop process using local estimate, as it will be illustrated later. However, RBS supposes the existence of a fixed reference, which may be inappropriate for self-organized WSN. High latency is needed for nodes to acquire samples due to the exchange of timestamps in separate steps. Moreover, the reference is left unsynchronized. All these issues are tackled in this paper that proposes a fully-distributed solution, where all nodes have the same role and get synchronized faster.

Let, d_{ui}, d_{vi}, denote the reception delays for the i^{th} samples, respectively u_i and v_i, $i \in \{1, ..., K\}$, where K is the number of samples. Each of d_{ui} (respectively d_{vi}) is composed of a fixed portion, say fd_{ui} (respectively fd_{vi}), and a variable portion, say X_{ui} (respectively X_{vi}). The fixed portions are assumed to be equal and the variable portions to be Gaussian distributed random variables (rv) with the same parameters, i.e., $X_{vi}, X_{ui} \sim \mathcal{N}(\mu, \sigma_0^2)$. Note that Gaussian distribution for the delays– when packet transmission involve no queueing delay– has been empirically justified in [3]. It follows that $d_{ui} - d_{vi} = X_{ui} - X_{vi}$. Let us denote $X_{ui} - X_{vi}$ by X_i; Application of E.q. 2 yields, $u_i = v_i + \theta + X_i$. Therefore,

$$X_i = u_i - v_i - \theta, \tag{3}$$

where θ denotes the relative offset. X_i is the difference between two Gaussian rv with the same parameters. It is thus a zero mean Gaussian rv; i.e. $X_i \sim \mathcal{N}(0, \sigma^2)$, where $\sigma^2 = 2\sigma_0^2$.

2.2 Joint Offset/Skew Model

The local clock of node, n_i, can be approximated by [2],

$$C_i(t) = \alpha_i t + \beta_i, \tag{4}$$

where $\alpha_i(t)$ is the absolute *skew*, and $\beta_i(t)$ is the absolute *offset* of node i's clock. Relative equation relating two nodes' clocks, C_i and C_j, is given by,

$$C_j(t) = \alpha_{n_i \to n_j} C_i(t) + \beta_{n_i \to n_j}, \tag{5}$$

where $\alpha_{n_i \to n_j}$ and $\beta_{n_i \to n_j}$ represents the *relative* offset and skew, respectively, relating time at node, n_i, to the corresponding one at node, n_j.

Application of E.q. 5, and by removing indices of α and β for simplification, we get, $u_i = \alpha v_i + \beta + X_i$. Therefore,

$$X_i = u_i - \alpha v_i - \beta, \tag{6}$$

3 Proposed Solution

3.1 Assumptions

The following assumptions are supposed:

- The network topology may be arbitrary, with the only requirement that the network density is high enough to make sure there is at least one common neighbor for every couple of neighboring nodes.
- Nodes are neighbor-aware, i.e., every node is supposed to have a list of current neighbors that it can communicate with. This can be achieved using some neighbor discovery/management protocol, such as [6].
- Relative synchronization is used, where every node has its own clock that runs independently from the others. The synchronization is achieved by estimating parameters reflecting relative deviation with respect to one other, and no local-clock update is needed.

- For multi-hop synchronization, on-demand synchronization is considered instead of the global always-on model. That is, the aim is not to keep all the nodes tuned to a common global time (wall-clock model), but to provide reactive mechanisms allowing nodes to mutually synchronize whenever needed (on-demand).
- A synchronous MAC protocol is used where nodes have common active periods; This is to facilitate broadcasting, which is problematic in duty-cycled MAC protocols. However, it is not important which paradigms the MAC protocol uses (contention, FDMA, TDMA, etc.), and the proposed protocol works with any of them.
- Timestamping may be performed at any layer. The proposed protocol and estimators are not limited to a particular timestamping layer. But low-layer timestamping implementation trivially provides higher accuracy compared to high-layer tiemstamping.

3.2 Single-hop Synchronization

3.2.1 Description

Contrary to RBS, no anchor (reference) or super node is needed; all nodes are sensor motes that cooperatively get synchronized. The solution is proposed at a high level of abstraction, independently of the underlying protocols. This enables its implementation with any protocol stack, but there is ample room for optimization in real implementation through the use of particular protocols and cross-layer design, such as by using/integrating MAC protocol's messages/cycles. Despite continuous beacon broadcasting (as it will be shown in the following), duty cycling is enabled. The nodes just need to send beacons during active periods. The RBS reference role is equally divided amongst all participating nodes, while beacons and timestamp exchange are merged in the same packets/steps.

The proposed protocol runs in cycles, where nodes sequentially broadcast enhanced beacons, i.e. beacons carrying timestamps. IDs can be used to determine the order of the sequence. By knowing the IDs of its neighbors through the underlying neighbor discovery protocol, every node sends its beacon upon getting the one from its predecessor. If a node does not report its beacon during a predefined timeout, then the next one can implicitly use its slot. This ensures fault-tolerance, where the protocol continues performing correctly despite node failure. A beacon carries timestamps that report local reception times of previous beacons. For a neighborhood of N nodes, every beacon would carry $N-1$ timestamps. Without loss of generality, the beacon/timestamp exchange process for $N = 4$, and one cycle, is illustrated with the example of Fig. 1. $B_{i,j}$ denotes the j^{th} beacon (beacon transmitted at the j^{th} cycle)

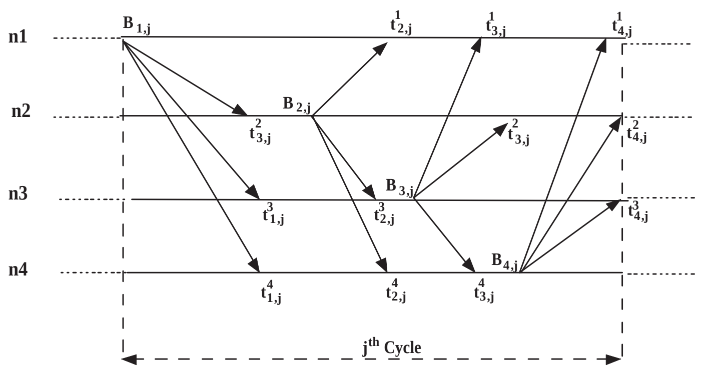


Fig. 1. Example of beacon broadcast during one cycle

of node, n_i, and $t^k_{i,j}$ refers to the reception timestamp at node, n_k, of node n_i's j^{th} beacon. Every beacon piggybacks the previous 3 timestamps. For instance, $B_{1,j}$ includes $t^1_{2,j-1}, t^1_{3,j-1}, t^1_{4,j-1}$, while $B_{4,j}$ includes $t^4_{1,j}, t^4_{2,j}, t^4_{3,j}$. These timestamps are then used by every node as samples to estimate relative synchronization parameters. Note that timestamp piggybacking into the beacons is not represented in the figure, but for each beacon, the reception timestamps at neighboring nodes are illustrated.

In what follows, synchronization between two arbitrary nodes, say, n_1 and n_2, is described, i.e., n_2's estimation of synchronization parameters with regard to n_1. The same process is to be applied for each pair of communicating nodes. Let u_i and v_i, $i \in \{1, ..., K\}$, denote the i^{th} beacon reception timestamp of nodes, n_1 and n_2, respectively, and, d_{ui}, d_{vi} the corresponding reception delays. Only beacons received by both nodes are used to construct samples, (u_i, v_i). The protocol tolerates beacon loss that is common due to the lossy wireless channels. A beacon loss at some node engenders a miss of one sample, which will be substituted by the next bacon received by the two sides. Refer to the previous example (Fig. 1) and assume– to simplify the description– all packets are received. The samples are, $t^1_{3,1} = u_1$, $t^1_{4,1} = u_2$,..., $t^1_{3,j} = u_{2j-1}$, $t^1_{4,j} = u_{2j}$, and $t^2_{3,1} = v_1$, $t^2_{4,1} = v_2$,..., $t^2_{3,j} = v_{2j-1}$, $t^2_{4,j} = v_{2j}$. Note that only timestamps of beacons received by the two synchronizing nodes from common neighbors are used to form tuples (u_i, v_i), where timestamps of beacons not received from both nodes are not used e.g., $B_{2,j}$ and $B_{1,j}$ [2].

[2] these beacons are actually used to form samples between other couples, i.e. (n_1, n_3) (n_1, n_4), (n_3, n_4), and (n_2, n_3), (n_2, n_4), (n_3, n_4) respectively

3.2.2 Estimators

In the offset-only model, the maximum likelihood estimator (MLE) is trivially the same as that of [3], which is given by:

$$\widehat{\theta}_{mle} = \frac{\sum_{i=1}^{K}(u_i - v_i)}{K} \tag{7}$$

Few has been done for the joint offset/skew estimation in the receiver-to-receiver synchronization. The model of [7] is the only one that considers MLE, but it completely deviates from the receiver-to-receiver concepts as it will be detailed in Section 6. In the following, the MLE in the joint offset/skew model is derived.
Using E.q. 6, the likelihood function gathering K samples, $\mathcal{L}(\alpha, \beta|X_1, ...X_K)$, is given by,

$$\mathcal{L}(\alpha, \beta|X_1, ...X_K) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(X_i)^2}$$

$$= (\frac{1}{\sqrt{2\pi\sigma^2}})^K e^{\frac{-1}{2\sigma^2}\sum_{i=1}^{K}(u_i - \alpha v_i - \beta)^2}. \tag{8}$$

Since

$$\widehat{\alpha}_{mle}, \widehat{\beta}_{mle} = \arg\max(\ln \mathcal{L}(\alpha, \beta|X_1, ...X_K)), \tag{9}$$

$\widehat{\alpha}_{mle}, \widehat{\beta}_{mle}$ may be obtained by vanishing the partial derivatives of the likelihood function's logarithm. That is, by resolving the system of equations,

$\frac{\partial \ln \mathcal{L}(\alpha,\beta|X_1,...X_K)}{\partial\alpha} = 0$, and $\frac{\partial \ln \mathcal{L}(\alpha,\beta|X_1,...X_K)}{\partial\beta} = 0$.

The resulted estimators are:

$$\widehat{\alpha}_{mle} = \frac{\sum_{i=1}^{K} u_i \sum_{i=1}^{K} v_i - K\sum_{i=1}^{K} v_i u_i}{(\sum_{i=1}^{K} v_i)^2 - K\sum_{i=1}^{K} v_i^2}, \tag{10}$$

$$\widehat{\beta}_{mle} = \tfrac{1}{K}(\sum_{i=1}^{K} u_i - \widehat{\alpha}_{mle} \sum_{i=1}^{K} v_i)$$

$$= \frac{1}{K}(\sum_{i=1}^{K} u_i - \frac{\sum_{i=1}^{K} u_i \sum_{i=1}^{K} v_i - K \sum_{i=1}^{K} v_i u_i}{(\sum_{i=1}^{K} v_i)^2 - K \sum_{i=1}^{K} v_i^2} \sum_{i=1}^{K} v_i). \tag{11}$$

Therefore, node n_2 uses samples (u_i, v_i), and applies Eq. (10) (resp. Eq. (11)) to calculate MLE for the relative skew, α, (respectively offset, β), or it applies Eq. (7) to calculate the relative offset (if the offset-only model is used). This is without the need to estimate the unknown delays. That is, all the delay parameters used in the model are unknown; $\mu, \sigma, X_{ui}, X_{vi}, d_{ui}, d_{vi}, f_{ui}, f_{vi}$, and there is no need to estimate them. Only samples (u_i, v_i) are empirically observed and used for the estimation (by node n_2), while the other parameters are vanished through the standard MLE method.

3.2.3 Lower-Bounds and Analysis

The CRLB (Cramer-Rao lower-bound) represents the theoretical lower-bound for any unbiased estimator. In other words, it is the lower-bound for the variance of any unbiased estimator. In this section, the CRLB is derived for both the offset-only and the joint offset/skew models.

Offset-only model: The CRLB is given by [8, p. 327].

$$Var(\widehat{\theta}) \geq \frac{1}{I(\Theta)}, \tag{12}$$

where $I(\theta)$ is the Fisher information defined as:

$$I(\Theta) = E[(\frac{\partial \ln \mathcal{L}(x, \theta)}{\partial \theta})^2] = -E[(\frac{\partial^2 \ln \mathcal{L}(x, \theta)}{\partial \theta^2})] \tag{13}$$

and $\mathcal{L}(x, \theta)$ denotes the likelihood function with K observations of x, i.e., $\mathcal{L}(\theta|X_1, ...X_K)$. It is given by,

$$\mathcal{L}(\theta|X_1,...X_K) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(X_i)^2}$$

$$= (\frac{1}{\sqrt{2\pi\sigma^2}})^K e^{\frac{-1}{2\sigma^2}\sum_{i=1}^{K}(u_i - v_i - \theta)^2}. \qquad (14)$$

After calculating second derivative of the logarithm of the previous expression, the following has be obtained.

$$Var(\widehat{\theta}) \geq \frac{\sigma^2}{K}, \qquad (15)$$

Joint offset/skew model: The CRLB can be derived from the inverse of the 2×2 Fisher information vector, say I^{-1}, using the property, $Var(\widehat{\alpha}) \geq (I^{-1})_{1,1}$, $Var(\widehat{\beta}) \geq (I^{-1})_{2,2}$. I is defined as [8, p. 343].

$$I = \begin{bmatrix} -\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\alpha^2} & -\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\alpha\partial\beta} \\ -\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\beta\partial\alpha} & -\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\beta^2} \end{bmatrix}. \qquad (16)$$

I^{-1} can thus be given by,

$$I^{-1} = \frac{1}{\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\alpha^2}\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\beta^2} - (\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\alpha\partial\beta})^2}$$

$$\begin{bmatrix} -\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\beta^2} & \frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\alpha\partial\beta} \\ \frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\beta\partial\alpha} & -\frac{\partial^2 \ln \mathcal{L}(\alpha,\beta|X_i)}{\partial\alpha^2} \end{bmatrix}. \qquad (17)$$

After calculating partial derivatives, the following bounds are obtained,

$$Var(\widehat{\alpha}) \geq (I^{-1})_{1,1} = \frac{K\sigma^2}{K\sum_{i=1}^{K} v_i^2 - (\sum_{i=1}^{K} v_i)^2}, \qquad (18)$$

$$Var(\widehat{\beta}) \geq (I^{-1})_{2,2} = \frac{\sigma^2 \sum_{i=1}^{K} v_i^2}{K\sum_{i=1}^{K} v_i^2 - (\sum_{i=1}^{K} v_i)^2} \qquad (19)$$

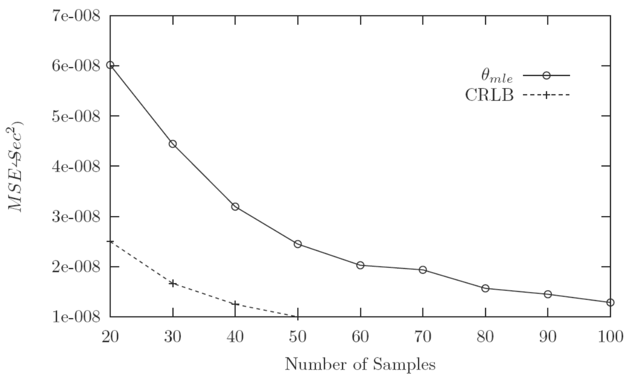
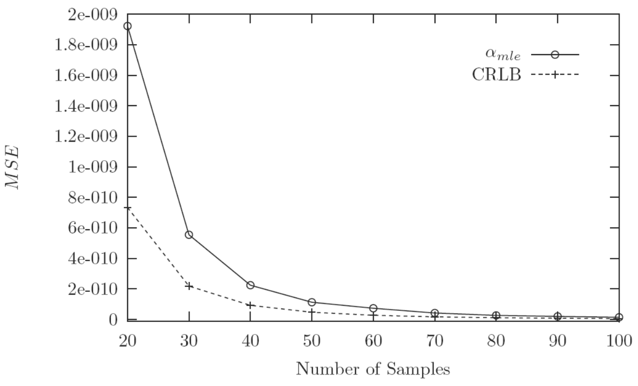
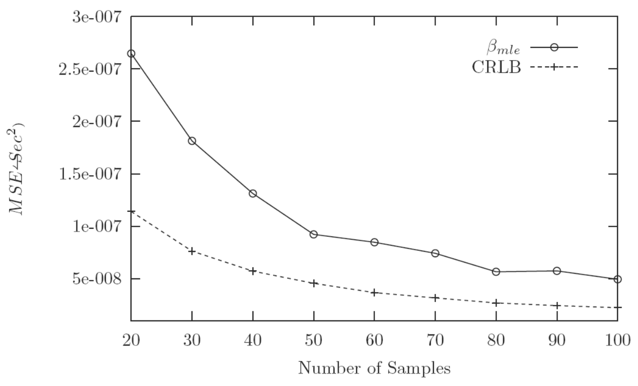


Fig. 2. MSE of θ estimation vs. number of beacons



(a)



(b)

Fig. 3. MSE of α and β estimation vs. number of beacons

A numerical analysis has been carried out to analyze the estimators and compare their mean square errors (MSE) with their respective theoretical lower-bound (CRLB). μ and $sigma$ has been fixed to $10{-}3sec$, while the relative skew and offset have been randomly sleeted in each run. The aim here is limited to analyzing the estimators in generic settings. Protocol simulation and comparison with some state-of-the-art protocols in a network simulation is given later.

Fig. 2 shows simulation results for the offset-only model, while plots of Fig. 3 show the results in the joint offset/skew model. Each point of these plots is the average of 10^4 measurements. Even with 99% confidence interval, the error bars are completely invisible. The figures illustrate how the proposed estimators' MSE decrease and quickly converge of the parameters to their respective CRLB as the number of beacons (K) increases.

3.2.4 Complexity and Discussion

By gathering beaconing and timestamp exchange, sample acquisition is performed faster compared to RBS-based protocols, where timing information exchange between receivers are performed in different steps posterior to beacon broadcast. No such steps are needed for the proposed protocol. In one cycle, the proposed protocol causes transmissions of n beacons (one per node) and provides $n-2$ samples for every pair of nodes, which makes $\frac{n(n-1)}{2}(n-2)$ in total. RBS provides one sample for each pair of nodes in a cycle, with $n+1$ transmissions (one beacon and n packets for timestamp exchange) if timestamps are exchanged using broadcast communications. This totally results in $\frac{n(n-1)}{2}$ per cycle. In other words, the proposed protocol needs $O(n)$ transmissions (or one cycle) to provide $O(n^3)$ samples, whereas RBS would need $O(n^2)$ transmissions to provide the same number of samples, i.e. $n-2$ cycles.

The previous analysis supposes broadcast mode exchange for both protocols (a trivial optimization of RBS). If unicast exchange is used like proposed in the original paper, $O(n^2)$ transmission is needed in every cycle to exchange timestamps instead of $O(n)$, [1], which results in $O(n^3)$ transmissions to provide the $O(n^3)$ samples. Fast acquisition of samples ensures convergence of the estimators as demonstrated in Fig 3. This will be investigated in deep later via a comparative simulation study. In addition to the low complexity for samples acquisition, the protocol allows to synchronize *all* the nodes, whereas RBS does not synchronize the reference.

3.3 Multi-hop Extension

First, an extension of the model and the estimators is given for a multi-hop route, then the protocol extension is described. Consider node, n_1, synchronization to another remote node, n_h, and assume the established route is $(n_1, n_2, n_3, ..., n_h)$. MLEs of local synchronization parameters on each hop (n_i, n_{i+1})– where $i \in \{1, ..., h-1\}$– are to be calculated as described in Section 3.2. They will be used in the following multi-hop extension.

3.3.1 Multi-hop Models

The multi-hop models for both the offset-only and skew/offset models are given in the following.

Offset-only Model: Let, t_{n_i}, denotes the time reading of node n_i's clock at instant, t, and, $\theta_{n_i \to n_j}$, denotes the relative offset relating time at node, n_i, to the corresponding one at node, n_j. That is, $t_{n_j} = t_{n_i} + \theta_{n_i \to n_j}$. Time readings of nodes on the route can hence be related by: $t_{n_{i+1}} = t_{n_i} + \theta_{n_i \to n_{i+1}}, i \in \{1, ..., h-1\}$.

By successive substitutions of t_{n_i} expressions in $t_{n_{i+1}}$ equations ($i \in \{2, ..., h\}$), the following has be obtained,

$t_{n_h} = t_{n_1} + \sum\limits_{i=1}^{h-1} \widehat{\theta}_{n_i \to n_{i+1}}$.
Consequently,

$$\widehat{\theta} = \sum_{i=2}^{h} \widehat{\theta}_{n_i}, \tag{20}$$

where for simplifying notations, each $\widehat{\theta}_{n_i \to n_{i+1}}$ in the considered route is denoted by $\widehat{\theta}_{n_{i+1}}$. It represents the local estimator that is calculated as described in Section 3.2.

Joint Offset/Skew Model: Let $\alpha_{n_i \to n_j}$ (respectively $\beta_{n_i \to n_j}$) denotes the relative skew (respectively offset) relating time at node, n_i, to the corresponding one at node, n_j. That is, $t_{n_j} = \alpha_{n_i \to n_j} t_{n_i} + \beta_{n_i \to n_j}$. Time readings of nodes on the route can thus be related by: $t_{n_{i+1}} = \alpha_{n_i \to n_{i+1}} t_{n_i} + \beta_{n_{i+1} \to n_i}, i \in \{1, ..., h-1\}$

By successive substitutions of t_{n_i} expressions in $t_{n_{i+1}}$ equations ($i \in \{2, ..., h\}$), the following has be obtained,

$$t_{n_h} = (\prod_{i=1}^{h-1} \alpha_{n_i \to n_{i+1}}) t_{n_h} + \sum_{i=2}^{h-1} [(\prod_{j=2}^{i} \alpha_{n_{j-1} \to n_j}) \beta_{n_i \to n_{i+1}}] + \beta_{n_1 \to n_2}.$$

Consequently,

$$\alpha_{n_1 \to n_h} = \prod_{i=1}^{h-1} \alpha_{n_i \to n_{i+1}}, \tag{21}$$

$$\beta_{n_1 \to n_h} = \sum_{i=2}^{h-1} [(\prod_{j=2}^{i} \alpha_{n_{j-1} \to n_j}) \beta_{n_i \to n_{i+1}}] + \beta_{n_1 \to n_2}. \tag{22}$$

For the sake of simplification on the considered route $\{n_1, n_2, n_3, ..., n_h\}$, $\alpha_{n_i \to n_{i+1}}$ ($i \in \{1, .., h-1\}$) is denoted in the following by α_{i+1} (the initial term α_1 is set by definition to 1), $\beta_{n_i \to n_{i+1}}$ is denoted by β_{i+1}, $\alpha_{n_1 \to n_h}$ by α, and $\beta_{n_1 \to n_h}$ by β. Applying these notations to Eq. (21) and Eq. (22), α and β estimators can be written as,

$$\widehat{\alpha} = \prod_{i=2}^{h} \widehat{\alpha}_i, \tag{23}$$

$$\widehat{\beta} = \sum_{i=1}^{h-1} [(\prod_{j=1}^{i} \widehat{\alpha}_j) \widehat{\beta}_{i+1}], \tag{24}$$

where, $\widehat{\alpha_i}$, $\widehat{\beta i}$, on each hop, represent local MLE estimators that are calculated as described in Section 3.2.

3.3.2 Protocol

Every node runs the local synchronization protocol with all its neighbors, as described in the previous section. This process will be executed continuously, which allows each node to permanently have local pairwise estimates for every neighboring node. The set of neighbors of a node may be defined either as all the nodes within the node's vicinity, or a subset of such nodes selected by the neighbor discovery protocol when a topology control mechanism is used. This set defines the nodes with which the appropriate node communicates, and all possible links that may include the node in potential routes. As soon as a node initiates a multi-hop communication that needs synchronization between the end-points, intermediate routers may forward local relative synchronization parameters (the single-hop estimators resulted from the local protocol execution) to the communicating nodes that are used to calculate multi-hop relative parameters. Therefore, the multi-hop synchronization is performed only on-demand.

Figure 4 represents a network topology for an illustrative example of the multi-hop extension. Giving that nodes run the local synchronization continuously, every node will have local estimates at any time. For instance, node, n_1, maintains relative synchronization parameters (offset when the offset-only model is used, and both offset and skew when the joint skew/offset model is used) to nodes, n_2, n_3, n_4, node, n_2, maintains parameters to nodes, n_1, n_4, n_7, where node, n_6, maintains relative parameters to nodes, n_4, n_5, n_7, n_8. To estimate parameters (local synchronization) between nodes, n_1, and n_2, for example, reception timestamps of beacons from node n_4 (the only common neighbor) are used, whereas beacons from nodes, n_5 and n_7, are used to synchronize nodes, n_4, n_6. If the route connecting nodes, n_1, n_8, resulted from the route discovery procedure is (n_1, n_4, n_6, n_8), then node, n_1, provides relative parameters relating its clock to the one of node, n_4, i.e. $\theta_{n_1 \to n_4}$ (respectively $\alpha_{n_1 \to n_4}$, $\beta_{n_1 \to n_4}$ for the joint model). Node, n_4, provides parameters relating its clock to the one of node, n_6, i.e. $\theta_{n_4 \to n_6}$ (resp. $\alpha_{n_4 \to n_6}, \beta_{n_4 \to n_6}$ for the joint model), and finally the latter provides parameters relating its clock to that of node, n_8, say $\theta_{n_6 \to n_8}$ (resp. $\alpha_{n_6 \to n_8}, \beta_{n_6 \to n_8}$ for the joint model). All parameters are forwarded to node, n_8, by piggybacking to the first packet of the session, allowing node, n_8, to calculate the end-to-end offset (resp. skew and offset) using Eq. (20) (resp. Eq. (23) and Eq. (24)), which enables it converting time at node, n_1, to its own. The packets following the one carrying the synchronization parameters may follow any arbitrary route without affecting the obtained estimators, and they are not constrained by the route followed by that packet. The resulted estimator(s), however, must have an expiration period, during which they can be used by node, n_8, for conversion (if needed). Upon expiration, node, n_1 launches an update by including synchronization parameter(s) in the first packet following the expatriation time (if any). For this purpose, node, n_1, just needs to trigger a local timer. The expiration period depends upon many factors such as the required precision, the estimation model, etc. The experiments confirm that the joint skew/offset model allows for high precision during a much longer period than the offset-only model, as it will be demonstrated in Section 5.

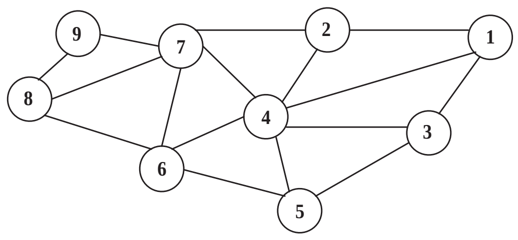


Fig. 4. Example of a multi-hop topology

4 Simulation and Comparison

In this section, the proposed protocol– termed in what follows R4Syn, for receiver-to-receiver referenceless synchronization – is compared by simulation with RBS [3], TPSN [9], CS-MNS [10]. On the one hand RBS and TPSN represent, respectively, a receiver-to-receiver and a sender-to-receiver protocol, and they both use relative synchronization similarly to R4Syn. On the other hand CS-MNS uses mutual synchronization through virtual clock update, where the aim is to allow virtual clocks at all nodes to mutually converge, instead of estimating *relative* skews/offsets. The goal of this part is to compare the concepts of the protocols and particularly investigate the multi-hop extension. To make the comparison fair, RBS has been implemented with estimators proposed in this paper, and TPSN with the estimators proposed in [11] that use the joint skew/offset model and are appropriate to sender-to-receiver protocols. CS-MNS as described in the original paper considers skew update in its model, through the called *correction factor* that is mutually updated.

First, the protocols are compared in single-hop environment, where all the nodes are within the vicinity of each other. Neighborhood has been statically fixed at every node without any topology control. The number of nodes has been varied to measure the performance metrics. For simplification, the propagation model has been simulated with no packet loss. Each point of the following plots is the average of 200 measurements, where in each run the delays (jitters) have been randomly set to values in the interval $]0ms, 10ms]$, which represents a typical range of values for delays in WSN when timestamping at a higher layer. The clock skews have been picked up from the interval $[1.001, 1.002]$ for each node. The high layer timestamping has been simulated to investigate the impact of fast sample acquisition. Note that no significant difference with respect to the precision would be noted between the protocols when timestamping at the MAC layer, notably for the offset-only model, where the sample size will have no significant impact. The results presented in the following have been obtained with 99% confidence interval [3] . Fig 5 shows the obtained number of samples, in one cycle, vs. the number of nodes. The cycle

[3] note that the error bars could not be made visible in all plots, as they are at a tiny scale compared to the y axis scale

has been fixed to $10sec$ for all protocols. It is clear that the proposed protocol acquires more samples as the number of nodes rises, and that its growth is much more important than the other protocols. This confirms the analysis of Section 3.2.4 reporting that R4syn enables in one cycle to acquire $(n-2)$ samples for every pair of nodes, which makes $O(n^3)$ in total. Like RBS, TPSN allows to get a single sample for every pair of nodes, which makes $O(n^2)$ in total. CS-MNS allows to get samples in a faster rate compared to RBS and TPSN, as every beacon transmitted by a node will carry one timestamp that can be used by all its neighborhood for updating the correction factor [10]. This results in $n*(n-1)$ samples in a cycle, which is twice compared to RBS and TPSN, but the difference with R4syn is more important where the number of samples is proportional to the number of pairs of nodes for every beacon.

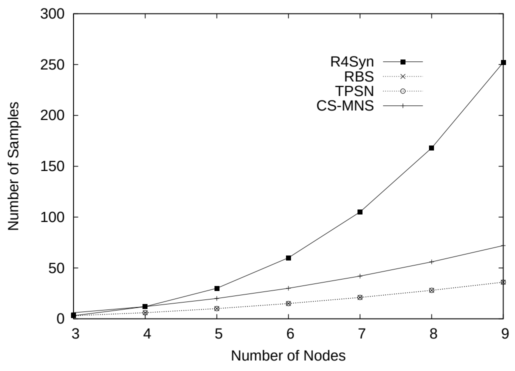


Fig. 5. Number of Samples vs. number of nodes

The impact of the fast sample acquisition on the synchronization precision is investigated in the following. Fig. 6 represents the absolute synchronization error vs. time. The real skew and offsets are randomly selected for all nodes, and synchronization error between two randomly selected nodes has been measured after nodes estimate relative skew/offset by running the appropriate protocol, or after updating virtual time in case of CS-MNS. Note that the same values are selected for all the protocols. Two different scales have been used for the x axis; The smaller scale on the left side (from 0 to $500sec$) clarifies the difference between RBS and R4syn, whereas the relaxed scale on the remaining period permits to investigate the convergence of the protocols' precision on a long period. All the protocols converge towards almost the same precision, about $5\mu sec$, with a slightly larger value for CS-MNS (about $7\mu sec$). However, there is a clear difference for the convergence time. R4syn converges in less than $200sec$, where CS-MNS needs about $500\mu sec$ (more than twice higher than R4syn), and RBS needs about $2000sec$ (10 times higher than R4syn). TPSN needs the largest period, more than $4000sec$ (more than 20 times higher than R4syn). It has been noted that after $4500sec$, all the protocols stabilize around $5\mu sec$ (or around $7\mu sec$ for CS-MNC), and no significant variance would be shown. The fast convergence of the proposed protocol is basically due to the fast acquisition of samples, as explained previously. The minor additional error for CS-MNS can be argued by the fact that it does not consider the delay jitter and its distribution in the model, contrary to the other protocols.

To measure the precision in multi-hop environment and investigate the proposed multi-hop extension, the network presented in Fig. 7 has been simulated. It consists of 33 nodes ranged in a greed-like-topology, with a 10-hop diameter. The simulation started by a warm-up step that consists in running local synchronization in every neighborhood for a long period. This is to assure local parameter estimators convergence for all the protocols. The synchronization error has been then measured between node n_0, and nodes n_1, n_2, ..., n_{10}, respectively, which represents a series from one to ten hops. The results shown in Fig. 8 clearly demonstrate that the proposed protocol outperforms the other ones. R4Syn synchronization error did not exceed $5.6\mu sec$, even for 10 hops synchronization. RBS also provides good precision, and its error is below $7\mu sec$ for 10 hops. However, the gap between the previous protocols and the sender-to-receiver ones (TPSN and CS-MNS) is important. The error of the TPSN exceeds $10\mu sec$ upon 3 hops, and it reaches $20\mu sec$ for 10 hops, while that of CS-MNS is even higher and it exceeds $30\mu sec$ for 10 hops. Note the use of two different scales for the y axis; the use of the higher scale would not show any difference/variance between/for RBS and R4Syn, where the other scale would not allow to plot TPSN and CS-MNS that have relatively much higher values. This difference is basically due to the use of completely different paradigms. The receiver-to-receive extends better to multi-hop environment, where the error precision considerably amplifies in multi-hop environment with the sender-to-receiver synchronization. Further, the mutual clock update of CS-MNS demonstrates the worst performance in the multi-hop scenario. The results also show clear improvement of R4syn over RBS in multi-hop environment, as the gap rises with the increase of hop number. Finally, note that the protocols do not use multi-hop signaling, so it is insignificant to measure the other metrics in multi-hop environment.

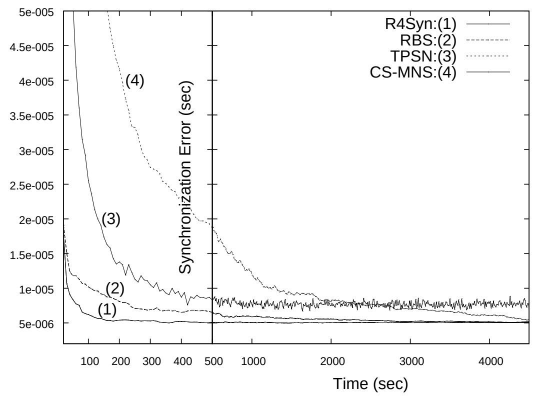


Fig. 6. Precision vs. time

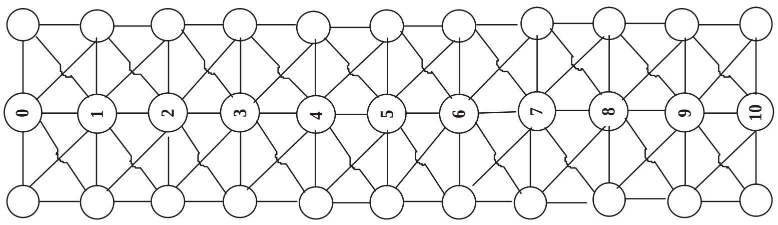


Fig. 7. Multi-hop simulation topology

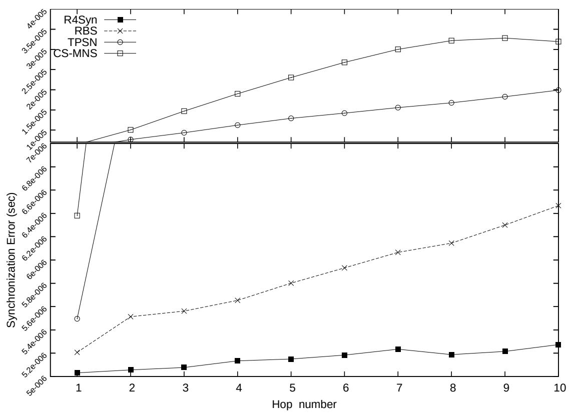


Fig. 8. Precision in multi-hop environment

5 Implementation on Sensor Motes and Tests

To investigate its accuracy and effectiveness and confirm the simulation results, the proposed solution has been implemented on real senor motes. MICAz mote[4] has been used, along with TinyOS as the underlying operating system [12]. The used mote has an IEEE 802.15.4 radio (CC2420), which is a ZigBee compliant RF transceiver that operates in the Industrial Scientific and Medical (ISM) band, at 2.4 to 2.4834 GHz. MICAz has low power consumption and uses two AA batteries. Its micro-controller is an 8-bit, 8 MHz ATmega128L, with a 4 KB RAM and 128 KB of external flash memory. The mote integrates two clocks; i) an internal clock with a high granularity, but also with a drift at the order of few tens PPM (up to 100PPM) [13], and ii) a more stable external crystal clock but with $32khz$ frequency, i.e. enabling a tick (granularity) of about $32\mu sec$. Although being more stable, the later does not allow for a microsecond level precision. Given that the aim is to confirm the high precision of the proposed solution, the former clock has been used in the experiment. It is straight forward to use the current implementation with the other clock, if a lower precision satisfies the application requirements (e.g. a precision at the order of milliseconds or tens of microseconds). An Arduino has been used for measuring the synchronization error. This microcontroller is an 8-bit, $16Mhz$

[4] http://www.memsic.com/userfiles/files/DataSheets/WSN/micaz_datasheet-t.pdf

ATmega328P, with 2 KB RAM, and 32 KB flash memory [5] .

The major problem we faced was the implementation of the estimators proposed in Eq. (10) and Eq. (11). Theses formulas include multiplications of very large numbers (clock values), which causes memory overflow after few seconds due to sensors' limitation on memory and arithmetic unit, together with TinyOS that does not support float numbers beyond 64 bits. This prevents getting large samples, and thus achieving microsecond level precision would not be possible. To tackle this, the formulas have been rewritten. It can be noticed that both the numerator and denominator of Eq. (10) are the difference of two large terms, but the final differences are small numbers that do not cause any computation problem for division. The problem is caused by the calculation of the large intermediate terms. For the numerator, the first term can be seen as the sum of products $(u_i v_j)$, consisting of k^2 elementary terms overall, and the second is the sum of products $u_j v_j$ that repeats K times, resulting in k^2 elementary terms as well. Therefore, instead of calculating the two large terms separately then subtracting them, it is more practical to subtracting from each elementary term of the first term, $u_i v_j$, an elementary term of the second one, $u_j v_j$. The same principle is applied to the denominator, which yields the following expression,

$$\widehat{\alpha}_{mle} = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K}(u_i v_j - u_j v_j)}{\sum_{i=1}^{K}\sum_{j=1}^{K}(v_i v_j - v_j^2)}, \tag{25}$$

The difference of large products are then transformed into the sum of differences of small products, which does not cause any problem for the implementation. β estimator is then calculated by replacing the resulted estimator of α in Eq. 11.

5.1 Preliminary Experiment

In the first preliminary tests, three nodes running the protocol have been used, two of them have been plugged to an Arduino through available *pins* for precision measurement (Fig. 9). The size of the binary code of this three nodes implementation was (for every node) 17600 bytes in ROM and 984 bytes in RAM for the offset-only model, 28470 bytes in ROM and 1434 bytes in RAM for the skew/offset model. This includes code for protocol signaling, estimator

[5] http://arduino.cc/en/Main/ArduinoBoardUno

calculations, the operating system kernel and libraries, and control operations for measurement, all uploaded as a single file.

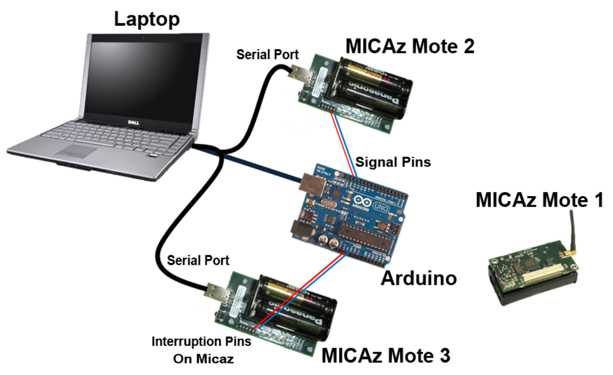


Fig. 9. Preliminary experiment setup

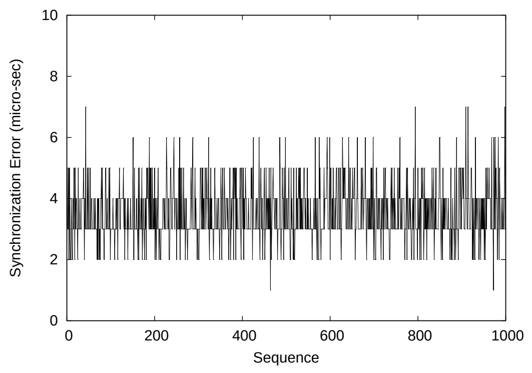


Fig. 10. Instant synchronization precision

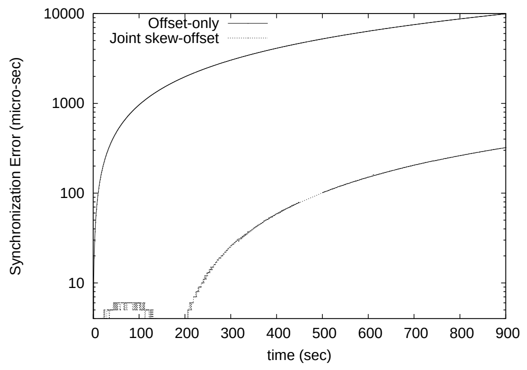


Fig. 11. Stability of synchronization precision

The Arduino has been programmed to transmit a signal to both motes at regular intervals. This is after few seconds of warm-up phase, allowing estimators to converge at both nodes. The interrupt handler for the appropriate *pin* on each node has been programmed as follows. The first mote picks its clock up, and the second one picks its clock up and converts the obtained value to the other's clock (estimate the other node's clock at the same moments the latter picks it up). This conversion is performed using the estimators resulted from the execution of the protocol. Every node then writes the obtained value into a laptop through the serial connection to construct a log file.

The experiment has been conducted using both offset-only and joint skew/offset models. Fig. 10 shows the precision for the offset-only model. As it can be seen, the error ranges from $1\mu sec$ to $7\mu sec$, with most values between $3\mu sec$ and $5\mu sec$; the average is 3.50. These results confirm fine-grained precision at the level of few microseconds as claimed in the simulation study. Similar result has been obtained for the joint model, with the same range and average of $3.24\mu sec$. The fast and immediate microsecond level obtained in the experiment results, compared to those of simulation, is due to the low-layer timestamping through the *CC2420ReceiveP* TinyOS module, whereas high layer timestamping was investigated in the simulation. Note that these experimental results are obtained when measurements are launched as soon as the estimators are calculated (with a variability of few microseconds). However, it is obvious that in practice, estimators should generally be used for some period before updating them. The longest the period is, the more stable and efficient the protocol is considered. This enables reducing the frequency of signaling and thus the overhead (in terms of number of packets), while keeping the high precision. To investigate this issue, the same experiment has been repeated, but protocol execution has been stopped after a warm-up period (for estimates acquisition), and measurers have been taken from that moment. Fig. 11 depicts the synchronization error vs. time for both models. As predicted, the offset-only model– which does not consider clock drifts– causes fast degradation of the precision. For instance, the error exceeds $500\mu sec$ after $1min$, $1msec$ after $2min$, and $6msec$ after $10min$. On the other hand, the skew/offset model assures more stability through predicting the drift. Its precision remains at the order of few microseconds (less than $10\mu sec$) during more than $200sec$, and it does not exceed $330\mu sec$ during the whole $15min$ experiment duration. We conclude that the joint model is more effective for general WSN applications. The offset model has the simplicity advantage, and it may be useful when implemented with more stable clocks (such as the external crystal clock), provided that only low precision is required (e.g; at the order of millisecond).

5.2 Multi-hop Experiment

The following describes an extensive experiment performed with seven nodes, $\{n_1, ..., n_7\}$. All nodes are in the vicinity (broadcast domain) of each other. However, a controlled (logical) topology is used for communication; a linear topology where node, n_1, is only neighbor to node, n_2, node, n_2, to node, n_3, etc. For node, n_1, to communicate with node, n_4, for instance, it must use the unique route (n_1, n_2, n_3, n_4), instead of direct communication. Therefore, full-power is used to exchange beacons between all nodes, but the predefined topology is used for data communication. This is to enforce multi-hop communications. Synchronization follows the same route followed by the first packet, as described previously. In other words, local synchronization is executed between the seven nodes, but nodes are programmed to use multi-hop synchronization for time conversion. For example, node, n_1, uses parameters of nodes, n_2, n_3, to estimate relative skew/offset to node, n_4, instead of using local (direct) estimation. This is, obviously, for the purpose of evaluating the multi-hop protocol/estimators.

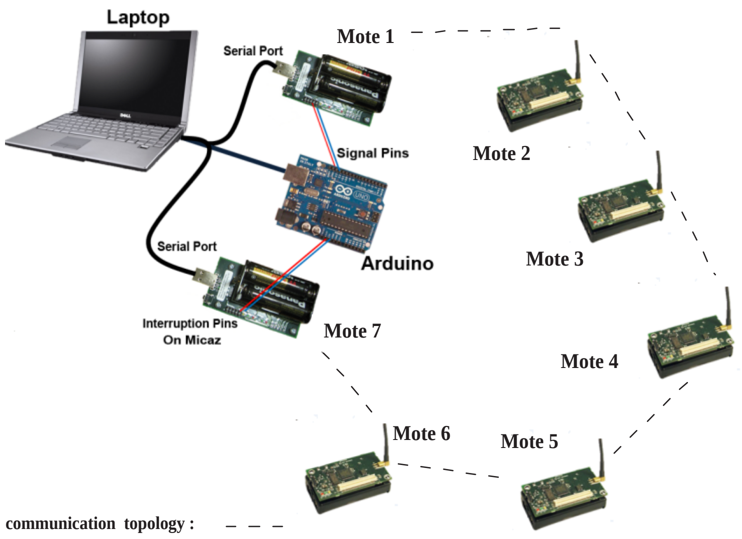


Fig. 12. Multi-hop experiment setup (e.g. for 6 hops)

Synchronization between node, n_1, and nodes, n_2, n_3,..., n_7, respectively for, 1-hop, 2-hop,..., 6-hop are investigated. Similar setting as presented in Fig. 9 is used, where the two synchronizing nodes are plugged to Arduino (node n_1, and node, n_i, $i \in \{2, ..., 7\}$, respectively). Fig. 12 illustrates the setting for 6-hop synchronization measurement, i.e. between nodes, n_1, and, n_7. The memory footprint of this seven-node implementation was 17600 bytes in ROM and 1046 bytes in RAM for the offset-only model, 28470 bytes in ROM and 1945 bytes in RAM for the skew/offset model. Fig. 13, through Fig. 18 represent results from 1-hop to 6-hop, respectively, and table 1 summarizes the results.

Fig. 13 confirms the results obtained with the three-node experiment, as the same precision order has been obtained, although the number of nodes has doubled. The synchronization error was between $1\mu sec$ and $6\mu sec$ with $2.45\mu sec$ on average. Similar results are obtained for the skew/offset model– which are not presented to avoid a redundant and overwhelming presentation– and the synchronization error was between $0\mu sec$ and $5\mu sec$, with $2.59\mu sec$ on average. Fig. 14 to Fig. 18 illustrate the behavior of the proposed multi-hop synchronization. They demonstrate that the increase in both synchronization error with the number of hops is smooth, and such is the flip of this error. The average value ranges from $3.65\mu sec$ for two hops to $19.93\mu sec$ for six hops.

The highest value has been below $40\mu sec$. Further, most of the values are below the average value, i.e., more than 50% of values are below the average value for all numbers hops. We should point out here the difference in values obtained in comparison with the simulation results. Although both simulation and testbed experiment demonstrated precision at the microsecond level, real experiment results are higher. This is because ideal models have been used in the simulation (propagation with no packet loss, similar delays in all links, etc.), where the experimentation with real motes reflects results in more realistic conditions.

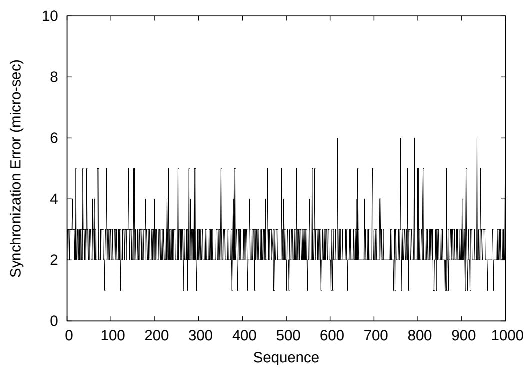


Fig. 13. Synchronization error in seven nodes scenario, one hop

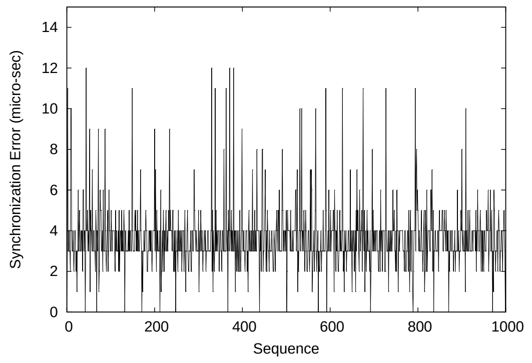


Fig. 14. Synchronization error in seven nodes scenario, two hops

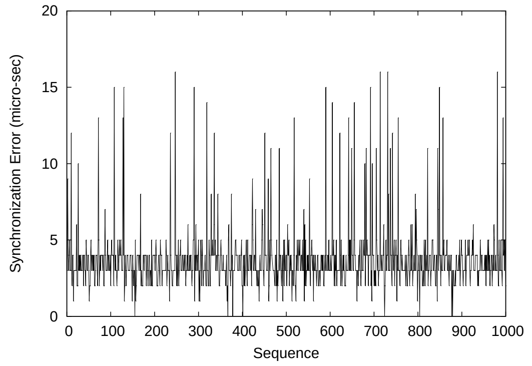


Fig. 15. Synchronization error in seven nodes scenario, three hops

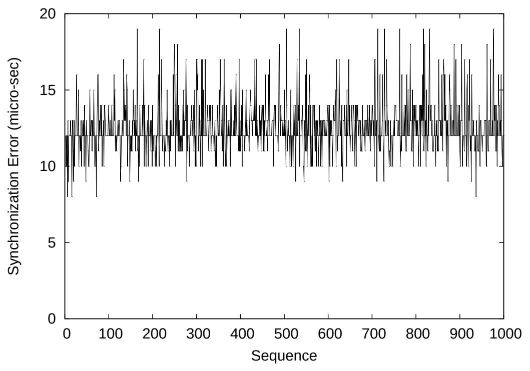


Fig. 16. Synchronization error in seven nodes scenario, four hops

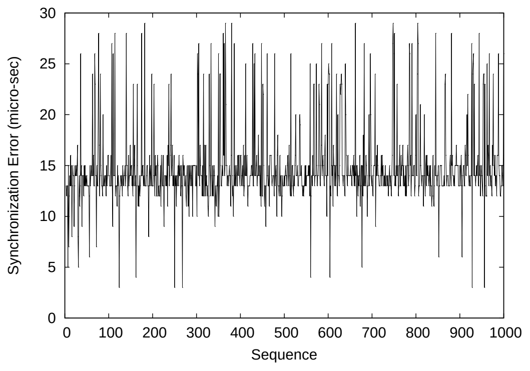


Fig. 17. Synchronization error in seven nodes scenario, five hops

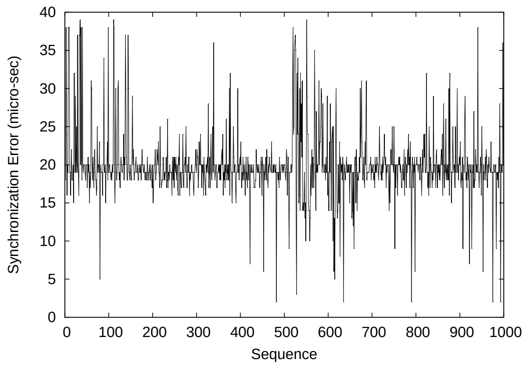


Fig. 18. Synchronization error in seven nodes scenario, six hops

6 Related Work

Sivrikaya and Yener [2] published a good introductive survey on time synchronization in wireless sensor networks (WSN), while Sundararaman et al. [1] provide more taxonomy and descriptions. Many solutions have been proposed after the publication of these two papers. Simeone et al. [14] and Lenzen et al. review some recent solutions, while Wu et al. [15] present an interesting analysis of the estimation methods used for time synchronization. Still, an up

Table 1
Error precision in μsec

	1-hop	**2-hop**	**3-hop**	**4-hop**	**5-hop**	**6-hop**
Average	2.45	3.65	3.74	12.58	14.84	19.93
Worst case	6	12	16	19	29	39
Best case	1	0	0	8	3	2
Below average	60.3%	50.8%	54.3%	57.9%	63.2%	51.3%

to date comprehensive review on time synchronization protocols in wireless sensor networks is missing, given the massive work in recent years.

Most of the protocols proposed thus far use sender-to-receiver synchronization, where the synchronizing nodes exchange messages and use sending and receiving events to record timestamps. Some solutions allow nodes to run their clocks independently and define mechanisms to calculate (estimate) relative skews and/or offsets, i.e. relative synchronization. Noh et al. [11] consider local (one-hop) relative synchronization and provide general offset/skew MLE estimators for sender/receiver protocols, where only the first and last observations of timing message exchanges are used. The approach was generalized in [16]. A similar mathematical approach is used in this paper, but the model, and consequently all the derived estimators and bounds, are completely different. The model of [11] does not apply to the proposed solution, as the latter is receiver-to-receiver-based. RAT [17] is another local sender/receiver synchronization protocol. It has been integrated with B-MAC to incrementally improve its performance. Since the authors' aim was to reduce the preamble period at the MAC layer, the protocol provides a weak synchronization, where linear regression was used to calculate relative skews/offsets. The Elapsed Time on Arrival (ETA) is another relative synchronization based algorithm [18]. This has been used in two implementations; The Routing Integrated Time Synchronization (RITS), which is a reactive time synchronization protocol that can serve in correlating multiple event detections at one or more locations, and the Rapid Time Synchronization (RATS) as a proactive protocol that utilizes RITS to achieve network-wide synchronization. Useful API is also provided for application use.

Another category of solutions consists in defining distributed mechanisms that allow nodes to update their clocks and converge to common values, i.e. continuously performing clock updates. This is conceptually different from relative synchronization (relative offset/skew estimation). In [19], [4], and [20] the authors propose solutions that deal both with single and multi-hop synchronization. These solutions have the gradient property and focus on local synchronization, where high precision is provided for neighboring nodes (their clock values converge closely), but the precision decreases with the distance. The solution proposed in this paper uses this property in the sense that more importance is given to local synchronization that is performed continually, but the performance evaluation results for multi-hop synchronization reflect high precision. Solutions based on distributed consensus techniques are very similar to gradient ones. AverageTimeSynch [21] and DCTS [22] protocols are examples of such solutions. The later has been analyzed in [23] for a Gaussian-delay environment. Sadler [24] considers the uses of an external accurate clock whose values may be occasionally observed through broadcast signal reception (such as GPS receivers). He provides estimators to synchronize the sensor mote's clock to such an accurate clock. A common feature of all these solutions is that nodes continuously update their clocks with respect to each other. This update generally involves jump/freeze of the clock value, which may affect correctness of local events' timestamping.

Contrary to gradient solutions, other ones focus on global synchronization and attempt to improve the multi-hop precision. [25] proposes a global synchronization protocol based on spanning tree. [26] provides probabilistic lower bounds for multi-hop synchronization. Secondis [13] defines strategies to disseminate synchronization messages from the root to the whole network. CS-MNS by Rentel et al. [10] define a mutual synchronization multi-hop solution that applies to both MANET (mobile ad hoc networks) and WSN. Time update of virtual clocks is used, where a correction factor of the skew is mutually updated upon beacon exchange using the sender-to-receiver paradigm. The protocol has been implemented on TelosB/MICAz motes [27], where the external low resolution clock have been used permitting only a weak resolution ($32\mu sec/tick$). The protocol proposed in [28] deals with multi-hop synchronization by clustering the networks. The protocol starts by synchronizing cluster-heads to the base station, then cluster-heads synchronize regular nodes. In [29], a centralized approach to estimate clock relative offset is proposed, where topology constraints are considered. The basic idea is that in a cycle, the sum of all relative offsets must be null. The authors formulated the problem with graph theory and the mean square method, as a constrained optimization problem. FTSP [5], TPSN [9], solutions of [30], [31], and [32] are other examples of other multi-hop-centric synchronization protocols.

As described previously, RBS [3] uses a completely different approach, namely the receiver-to-receiver synchronization, which has the advantage of reducing the time-critical path [1, P. 294]. However, the major drawback of RBS is the need of a fixed reference, which might be inappropriate for some self-organized wireless sensor network (WSN) applications. The work in [7] is the only one that considers joint skew/offset maximum likelihood estimation (MLE) for RBS. The model used herein is different from the one of Sari et al. In the latter, synchronization is proportional to a single reference, while there is no such a common reference with the proposed solution's model. Further, [7] considers exponentially distributed delays, while the proposed one considers Gaussian distributed delays. Exponential model is more appropriate when delays include queueing time, which is typical when the sending delay is affecting the time critical path. However, it has been realized that the reception delays tend to follow Gaussian distributions [3]. Therefore, Gaussian distribution is more suitable for receiver-to-receiver protocols. More importantly, the model of [7] does not synchronize receivers directly but through synchronization to the reference clock, which completely deviates from the RBS concept. Note that the concept of a reference in RBS is to use a common source for signal broadcast, but not a common source of time. Although relative parameters can be determined by synchronizing the receivers to the reference, this way of synchronization causes cumulative errors on the estimators. Moreover, eliminating the sender uncertainty from the critical path is at the core of RBS, while this uncertainty is not eliminated by relating the transmitter's (reference's) time to the receiver's time in the model[6]. The model in this paper is different, it allows to directly estimating relative parameters without using or referring to the reference clock. This faithfully reflects the RBS concept.

The authors in [33] apply artificial intelligence (AI) techniques to an RBS-like solution. The applicability of such AI method to resource constrained sensors is questionable. PBS [34] is a hybrid solution between sender-to-receiver and receiver-to-receiver approaches, which uses overhearing of the sender/receiver message exchange. The basic sender-to-receiver approach is used to synchronize two super nodes, while all the other nodes within the range of both nodes overhear messages and synchronize to the super nodes. The solution reduces the overheard of RBS, but it is also centralized. In [35], the authors tried to make the solution more distributed by proposing round-robin timing exchange protocol (RRTE), where one reference is fixed and the other changes in a round robin way. Still, one fixed reference is needed. Some solutions are applications-driven and attempt to couple synchronization with application services. For instance, Glossy [36] exploits IEEE 802.15.4 interferences, to couple synchronization with network flooding. In [37], a solution that couples localization to synchronization is proposed. Liu et al. [38] consider synchronization for mobile underwater networks, where the precision is at the order of hundreds of *msec* and even the *sec* level due to the high latency of the acoustic communications. Some other hardware solutions propose synchronization techniques to be implemented at the physical layer such as [39]. This may provide high precision but would require additional hardware modules at the sensor motes. The solution proposed in this paper does not need any such modules, and it is simply implementable at the software level.

[6] Refer to Eq. (1) and Eq. (2) in [7, p. 1], v_{x,λ_x}, v_{y,λ_y} would be the noise (variable delay) due to both transmission and reception. This adds transmission delays to the time critical path

7 Conclusion

Time synchronization has always been a challenging problem in distributed systems. This problem is more complex in wireless networks, where communication may be prone to high delay variability. In wireless sensor networks (WSN), node limitations (computation, memory, energy) add more constraints to the problem. The problem has been dealt with in this work, and a new distributed referenceless protocol has been proposed for single-hop synchronization, and then extended to multi-hop synchronization. The proposed protocol is receiver-to-receiver based, but it is fully distributed and does not rely on any fixed reference. The role of the reference is distributed amongst all nodes, while timestamps are piggybacked to synchronization signals (beacons). This enables to perform beaconing and timestamp exchange in a single step, and it provides fast acquisition of timestamps that are used as samples to estimate relative synchronization parameters. The proposed relative synchronization protocol has been extended to multi-hop environment, where local synchronization is performed proactively, and the resulted parameters are transferred to the intermediate/end-point nodes as soon as a multi-hop communication that needs synchronization is initiated. The multi-hop extension is on-demand, as no signals are periodically transferred on multi-hop links.

The maximum likelihood estimators (MLE) and the Cramer-Rao lower bounds (CRLB) has been proposed for both the offset-only and the joint offset/skew estimation. Contrary to existing MLE models for receiver-to-receiver synchronization, the model used in this paper directly relates the synchronization parameters of the receivers without using the sender's clock, which accurately reflects the receiver-to-receiver principle. The proposed estimators have been numerically analyzed and compared to the CRLB, as the theoretical optimum. Results show convergence of the proposed estimators' precision to their respective CRLB with the increase of the number of signals. The protocol has been compared by simulation to some state-of-the-art sender-to-receiver (TPSN, CS-MNS), and receiver-to-receiver (RBS) protocols. Results show much faster convergence of the proposed protocol, in the order of more than twice compared to CS-MNS, more than ten times compared to RBS, and more than twenty times compared to TPSN. Obtained results also show that the proposed protocol outperforms all the protocols in with respect to the synchronization precision, both in single-hop and multi-hop synchronization. In addition to the simulation evaluation, the proposed protocol has been implemented and tested on real sensor motes both in one-hop and multi-hop scenarios. The results were very satisfactory and confirm microsecond precision of both offset-only and skew/offset models, and the synchronization error has not exceeded $6\mu sec$ for 1-hop scenario and $39\mu sec$ for 6-hop scenario, in the worst cases. On average, it was between $2.45\mu sec$ for 1-hop and $19.93\mu sec$ for 6-hop, and most values were always below the average value. The skew/offset model– which captures clock drifts– demonstrated more stability, where the estimators have much longer lifetime and remain effective in keeping the microsecond-level precision for several minutes.

Acknowledgment

This work has been carried out at CERIST research center and supported by the Algerian Ministry of Higher Education and Scientific Research. The authors would like to thank the anonymous referees for devoting their time to reviewing the paper and their invaluable comments and suggestions, which significantly helped to improve the quality of the manuscript.

References

[1] B. Sundararaman, U. Buy, A. D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey, Ad hoc Networks 3 (3) (2005) 281–323.

[2] F. Sivrikaya, B. Yener, Time synchronization in sensor networks: A survey, IEEE Network Magazin 18 (4) (2004) 45–50.

[3] J. Elson, L. Girod, D. Estrin, Fine-grained network time synchronization using reference broadcasts, in: 5th USENIX Symposium on Operating System Design and Implementation (OSDI'02), 2002, pp. 147–163.

[4] P. Sommer, R. Wattenhofer, Gradient clock synchronization in wireless sensor networks, in: Proceedings of the 8th ACM International Conference on Information Processing in Sensor Networks (IPSN'08), 2009, pp. 37–48.

[5] M. Maróti, B. Kusy, G. Simon, Á. Lédeczi, The flooding time synchronization protocol, in: SenSys, 2004, pp. 39–49.

[6] M. Kohvakka, J. Suhonen, M. Kuorilehto, V. Kaseva, M. Hännikäinen, T. D. Hämäläinen, Energy-efficient neighbor discovery protocol for mobile wireless sensor networks, Ad Hoc Networks 7 (1) (2009) 24–41.

[7] I. Sari, E. Serpedin, K.-L. Noh, Q. M. Chaudhari, B. W. Suter, On the joint synchronization of clock offset and skew in rbs-protocol, IEEE Transactions on Communications 56 (5) (2008) 700–703.

[8] A. Papoulis, S. U. Pillai, Probability, Random Variables and Stochastic Processes, 4th Edition, McGraw Hill Higher Education, 2002.

[9] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks, in: Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03, 2003, pp. 138–149.

[10] C. H. Rentel, T. Kunz, A mutual network synchronization method for wireless ad hoc and sensor networks, IEEE Trans. Mobile Computing 7 (5) (2008) 633–646.

[11] K.-L. Noh, Q. M. Chaudhari, E. Serpedin, B. W. Suter, Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks, IEEE Transactions on Communications 55 (4) (2007) 766–777.

[12] P. Levis, D. Gay, TinyOs Programming, Cambridge University Press, 2009.

[13] F. Ferrari, A. Meier, L. Thiele, Secondis: An adaptive dissemination protocol for synchronizing wireless sensor networks, in: Proceedings of the 7th IEEE Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON'10), 2010, pp. 1–9.

[14] O. Simeone, U. Spagnolini, Y. Bar-Ness, S. H. Strogatz, Distributed synchronization in wireless networks, IEEE Signal Processing Magazine 25 (5) (2008) 81–97.

[15] Y.-C. Wu, Q. M. Chaudhari, E. Serpedin, Clock synchronization of wireless sensor networks, IEEE Signal Process. Mag. 28 (1) (2011) 124–138.

[16] M. Leng, Y.-C. Wu, On clock synchronization algorithms for wireless sensor networks under unknown delay, IEEE Transactions on Vehicular Technology 59 (1) (2010) 182–190.

[17] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. B. Srivastava, D. Ganesan, Estimating clock uncertainty for efficient duty-cycling in sensor networks, IEEE/ACM Transations on Networking. 17 (2009) 843–856.

[18] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, D. Culler, Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services, Int. Journal of Ad Hoc and Ubiquitous Computing 1 (4).

[19] R. M. Pussente, V. C. Barbosa, An algorithm for clock synchronization with the gradient property in sensor networks, Journal of Parallel Distributed Computing 69 (2009) 261–265.

[20] C. Lenzen, T. Locher, R. Wattenhofer, Tight bounds for clock synchronization, Journal of the ACM 57 (2).

[21] L. Schenato, F. Fiorentin, Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks, Automatica 47 (9) (2011) 1878–1886.

[22] A. Scaglione, R. Pagliari, Non-cooperative versus cooperative approaches for distributed network synchronization, in: IEEE PerCom Workshops Proceedings, 2007, pp. 537–541.

[23] G. Xiong, S. Kishore, Analysis of distributed consensus time synchronization with gaussian delay over wireless sensor networks, EURASIP Journal on Wireless Communications and Networking 2009 (May).

[24] B. M. Sadler, Local and broadcast clock synchronization in a sensor node, IEEE Sig. Proc. Letters 13 (1) (2006) 9–12.

[25] L.-M. He, Time synchronization based on spanning tree for wireless sensor networks, in: Proceedings of the 4th IEEE Conference on Wireless Communications, Networking and Mobile Computing(WiCOM'08), 2008, pp. 1–4.

[26] C. Lenzen, P. Sommer, R. Wattenhofer, Optimal clock synchronization in networks, in: Proceedings of the 7th ACM International Conference on Embedded Networked Sensor Systems (SenSys'09), 2009, pp. 225–238.

[27] T. Kunz, E. McKnight-MacNeil, Implementing clock synchronization in wsn: Cs-mns vs. ftsp, in: WiMob, 2011, pp. 157–164.

[28] L. Kong, Q. Wang, Y. Zhao, Time synchronization algorithm based on cluster for wsn, in: The 2nd IEEE International Conference on Information Management and Engineering (ICIME), 2010, pp. 126–130.

[29] I. Shames, A. N. Bishop, Relative clock synchronization in wireless networks, IEEE Communications Letters. 14 (2010) 348–350.

[30] J. Koo, R. K. Panta, S. Bagchi, L. A. Montestruque, A tale of two synchronizing clocks, in: Proceedings of the 7th ACM International Conference on Embedded Networked Sensor Systems (SenSys'09), 2009, pp. 239–252.

[31] B. Wehbi, A. Laouiti, A. R. Cavalli, Efficient time synchronization mechanism for wireless multi-hop networks, in: PIMRC, 2008, pp. 1–6.

[32] A.-s. Hu, S. D. Servetto, Algorithmic aspects of the time synchronization problem in large-scale sensor networks, Mob. Netw. Appl. 10 (4) (2005) 491–503.

[33] L. Paladina, A. Biundo, M. Scarpa, A. Puliafito, Artificial intelligence and synchronization in wireless sensor networks, Journal of Networks 4 (6) (2009) 382–391.

[34] K.-L. Noh, E. Serpedin, K. A. Qaraqe, A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization, IEEE Transactions on Wireless Communications 7 (9) (2008) 3318–3322.

[35] Y.-H. Huang, S.-H. Wu, Time synchronization protocol for small-scale wireless sensor networks, in: IEEE Wireless Communications and Networking Conference (WCNC'10), 2010, pp. 1–5.

[36] F. Ferrari, M. Zimmerling, L. Thiele, O. Saukh, Efficient network flooding and time synchronization with glossy, in: IPSN, 2011, pp. 73–84.

[37] J. Zheng, Y.-C. Wu, Joint time synchronization and localization of an unknown node in wireless sensor networks, Trans. Sig. Proc. 58 (3) (2010) 1309–1320.

[38] J. Liu, Z. Zhou, Z. Peng, J.-H. Cui, M. Zuba, L. Fiondella, Mobi-sync: Efficient time synchronization for mobile underwater sensor networks, IEEE Trans. Parallel Distrib. Syst. 24 (2) (2013) 406–416.

[39] T. Beluch, D. Dragomirescu, F. Perget, R. Plana, Cross-layered synchronization protocol for wireless sensor networks, in: Proceedings of the 2010 Ninth International Conference on Networks (ICN'10), 2010, pp. 167–172.