Time Synchronization Algorithms in Low-Cost Wireless Sensor Network Systems

Jue Yang and Xinrong Li

***Abstract*—Time synchronization is an essential building block of sensor network systems. In this paper, we present a comprehensive study of least squares algorithms for time synchronization in sensor networks, with a focus on continuous monitoring and data collection applications. We propose a set of algorithms to address a number of issues in practical implementation on typical low-cost sensor network platforms, including a scaled signal model to achieve numerical stability in an ill-conditioned problem, sequential estimators for the scaled signal model to reduce computational complexity, a fast initialization scheme to improve energy efficiency, and outlier detection algorithms to improve robustness in long-term autonomous operations. The proposed algorithms are implemented and a measurement-based simulation method is employed to study the performance.**

***Index Terms*—Energy efficient, least squares, continuous monitoring, time synchronization, sensor networks.**

I. Introduction

Wireless sensor networks (WSN), or simply sensor networks, have attracted considerable research and development efforts in the past few years and a wide range of innovative applications have been envisioned for sensor networks in consumer, industrial, military, public safety and security application sectors [1]–[9]. However, despite significant recent advances, there are still many challenging issues to be addressed to fulfill the full potential of the emerging sensor networking technology in practice.

Time synchronization is an essential building block of sensor network systems [1], [7]. In general, sensors are event-driven and sensor networks are mostly used for monitoring purposes. In such applications, sensor readings are collected periodically and the data are time-stamped for time-sensitive event detection, data aggregation, and/or coordinated control and actuation, necessitating time synchronization in the network [10]–[14]. In addition, duty-cycling and coordination-based medium access control (MAC) protocols are often used in energy-constrained sensor networks, especially in long-term monitoring applications [6], [10], [15]. Performance of such methods directly depends on the accuracy of time synchronization. Design requirements of time synchronization methods are largely application-specific. In this research, we focus on continuous monitoring and data collection systems that are commonly required in many applications such as intelligent healthcare, telemedicine, home and industrial automation, structural and environmental monitoring, etc., although many of the developments herein are readily applicable to many other types of applications.

Time synchronization in sensor networks has been studied extensively in recent years. However, the existing literature are mostly focused on the protocol aspect of time synchronization whereas little emphasis has been put on the algorithmic aspect, especially from practical implementation perspectives. Time synchronization in sensor networks generally begins with a message exchange protocol with which the nodes exchange their local time information. Then, the nodes calibrate their clocks to a common reference using some time synchronization algorithms. Reference broadcast synchronization (RBS) method [11], timing synchronization protocol for sensor networks (TPSN) [12], and flooding time synchronization protocol (FTSP) [13] are among the most widely cited time synchronization methods, or time synchronization protocols to be more accurate. In their original design, simple time synchronization algorithms are employed without in-depth analysis. In this paper, we intend to fill the gap in the literature with a comprehensive study of time synchronization algorithms. More specifically, we present a unified formulation of least squares (LS) time synchronization algorithms to estimate clock offset, skew, and drift (see Section II for definition) using both batch and sequential estimators. The simple algorithms used in RBS, TPSN, and FTSP are special cases of the ones presented in this paper. For example, TPSN only accounts for clock offset and its algorithm can be viewed as a 0th-order batch estimator. Both RBS and FTSP utilize a 1st-order batch estimator to estimate clock offset and skew. However, it is important to note that, as discussed in the following sections, most of the algorithms presented in this paper can be used together with the protocols of RBS, TPSN, and FTSP directly or with some minor adjustments.

Through extensive measurement and implementation experience, we have identified a number of key issues in the implementation of time synchronization methods, especially for continuous monitoring applications. In this paper, we propose a suite of algorithms to address such issues, including a scaled signal model to achieve numerical stability in an ill-conditioned problem, sequential estimators for the scaled signal model to reduce computational complexity, a fast initialization scheme to improve energy efficiency, and outlier detection algorithms to improve robustness in long-term autonomous operations. The proposed algorithms are implemented in an actual WSN platform to demonstrate their practicality and to study their computational complexity empirically. Performance of the algorithms is studied through extensive measurement-based simulations with the measurement

The authors are with the Department of Electrical Engineering, University of North Texas, 1155 Union Circle #310470, Denton, Texas 76203 USA (phone: 940-891-6875, fax: 940-891-6881, e-mail: xinrong@unt.edu).

This research is supported in part by NSF under Grants OCI-0636421, CNS-0709285, and EEC-0431818.

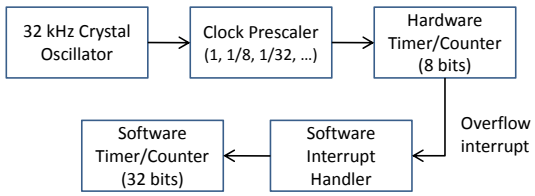

Fig. 1. Typical clock counter architecture of IRIS with ATmega1281 processor, where the overflow interrupt is actually implemented with an output compare register that can be flexibly configured [17].

data collected in typical application scenarios.

The rest of the paper is organized as follows. In Section II, we define the time synchronization problem and survey the related literature. In Section III, we present both batch and sequential LS time synchronization algorithms and several related algorithms for initialization and outlier detection. Then, in Section IV, we discuss issues in practical implementation and compare computational complexity of these algorithms. In Section V, the measurement and experimental setup used in this research is described. Measurement-based simulation results are presented in Section VI, followed by a summary of conclusions in Section VII.

II. Time Synchronization in WSN

It is usually impractical (or too expensive) to equip every sensor node with a real-time clock. Instead, each sensor node maintains a local clock that is essentially an integer counter triggered by an external crystal oscillator. For example, Fig. 1 shows a typical configuration of the clock counter of the IRIS mote with ATmega1281 micro controller unit (MCU) [17], [21], a widely used low-cost sensor network system [6]–[8]. The 32 kHz oscillator signal is first prescaled by dividing the frequency by a factor of 1, 8, 32, 64, 128, 256, or 1024 to configure hardware timer counter (HTC) resolution. Upon overflow of the 8-bit HTC, a software interrupt handler increases the value of a 32-bit software timer counter (STC) by 2^8, instead of by 1, to preserve the time resolution of the HTC. Thus, the clock time t of a sensor node is defined as

$$t = t_{\mathrm{HTC}} + t_{\mathrm{STC}}, \tag{1}$$

where t_{HTC} and t_{STC} are the current readings of the HTC and STC, respectively. The clock time t can be converted to a standard time unit based on the HTC's time resolution.

In sensor networks, each node has a clock counter that starts independently. The difference between the initial starting times is known as clock offset. Two oscillators may not run at an exactly same frequency, causing clock skew between two nodes. Oscillators suffer from aging effects and are affected by environmental variables, such as mechanical vibration, magnetic fields, and especially temperature [23]. Thus, the frequency of two oscillators may vary differently, causing clock drift between two nodes. If the clock offset θ_o, skew θ_s and drift θ_d coefficients between two nodes are known, time synchronization between a local time t and a reference time τ, which may or may not be the global or standard time, can be achieved through a 2nd-order polynomial time conversion formula,

$$t = \theta_o + \theta_s \tau + \theta_d \tau^2. \tag{2}$$

The choice of converting τ to t or t to τ is an application-specific design requirement; in some applications, both conversions may be required.

To synchronize two sensor nodes, a synchronization message is sent from one node, reference node, to the other node, client node. The reference node time-stamps the message before transmission and the client node time-stamps the message upon reception. Then, the client node can synchronize to the reference node based on the time-stamps of one or a sequence of synchronization messages. The synchronization message inevitably experiences variable delays at both transmitter and receiver due to uncertain send, access, transmission, propagation and reception times [13]. Recently, a number of efficient synchronization protocols have been proposed including RBS [11], TPSN [12], and FTSP [13].

With the RBS approach, a beacon message is broadcasted by a beacon node and two sensor nodes synchronize between themselves by exchanging their local receiving times of the beacon. Thus, RBS eliminates transmitter-side uncertainty, although time-stamping at the lower layers of networking protocol stack may achieve the same effect. Such a method incurs a large communication overhead in large-scale networks due to pair-wise message exchange. The TPSN approach removes delay uncertainties at both sender and receiver through MAC layer time-stamping and it gains additional accuracy over RBS by averaging time-stamps of two messages. However, the two-way communication required in this method also results in a high communication overhead. The FTSP approach performs MAC layer time-stamping and it is able to synchronize multiple receivers with a single broadcast message. Such a flooding-based method is also insensitive to topological changes. On the other hand, TPSN only adjusts the clock offset between two nodes, while RBS and FTSP estimate both clock offset and skew through LS estimation.

In continuous monitoring applications, duty cycling is often used to achieve energy-efficiency, which makes it desirable to reduce synchronization frequency. In order to maintain synchronization over a long synchronization period, it may be necessary to estimate clock drift in addition to clock offset and skew, especially in harsh environmental conditions where oscillator frequency drifts significantly. In this paper, we derive various LS algorithms to estimate clock offset, skew, and drift for the one-way broadcast-based synchronization model as employed in the RBS and FTSP approaches and study their performance based on measurement data.

III. Least Squares Estimation for Time Synchronization

In this paper, we adopt a simple broadcast-based model for synchronization; that is, a reference node broadcasts a time-stamped synchronization message periodically and a client node time-stamps the received messages. The client node synchronizes its time t to the reference node's time τ based on a sequence of the time-stamp measurement data sets

$\{\tau[i], t[i] : 1 \le i \le n\}$. In this section, we derive several LS estimation algorithms for such a synchronization model. The derivation is generalized for the p-th order polynomial estimator to suit both of the estimation of only clock offset and skew and the estimation of all three parameters.

A. Linear Least Squares Estimators

The signal model for the p-th order polynomial estimator at time n with w time-stamp measurement data sets, i.e., with a window size of w, can be defined as

$$\mathbf{x}[n] = \mathbf{H}[n]\theta + \mathbf{v}[n], \tag{3}$$

where θ is the vector of $p+1$ unknown polynomial coefficients to be estimated, and the observation data $\mathbf{x}[n]$, the observation matrix $\mathbf{H}[n]$, and the observation noise $\mathbf{v}[n]$ at time n are defined as

$$\begin{aligned}
\mathbf{x}[n] &= [t[n-w+1] \ \cdots \ t[n]]^T, \\
\mathbf{H}[n] &= [\mathbf{h}[n-w+1] \ \cdots \mathbf{h}[n]]^T \\
&= \begin{bmatrix} 1 & \tau[n-w+1] & \cdots & \tau^p[n-w+1] \\ 1 & \tau[n-w+2] & \cdots & \tau^p[n-w+2] \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \tau[n] & \cdots & \tau^p[n] \end{bmatrix}, \\
\mathbf{v}[n] &= [v[n-w+1] \ \cdots \ v[n]]^T.
\end{aligned}$$

In practice, statistical characteristics of the observation noises are typically unknown a priori. Thus, the LS estimation method is desirable for such time synchronization problems, especially comparing to other model-based methods such as maximum likelihood (ML) and minimum mean squared error (MMSE) estimators. Derivation of LS estimators (LSE) does not rely on any probabilistic assumption of the observation noises and it is optimal in the sense that it is the best linear unbiased estimator (BLUE) for linear signal models where the observation noises are uncorrelated and have a zero mean and equal variances [19].

As discussed in Section II, the time-stamp data $\tau[i]$ and $t[i]$ are 32-bit integer values. Thus, when $p > 1$, using the data directly in determining $\mathbf{H}[n]$ results in numerical overflow errors in computers. Converting the data to a standard time unit such as seconds cannot completely eliminate such overflow errors in practical implementation. Furthermore, given the form of $\mathbf{H}[n]$ in (3), computation of $(\mathbf{H}^T[n]\mathbf{H}[n])^{-1}$ directly or through Gaussian elimination to determine the LSE is an ill-conditioned problem, although well-posed. It is well known that if the elements of a matrix vary greatly in size, it is likely that large loss-of-significance errors will be introduced and the propagation of rounding errors will be worse [16]. To avoid such a problem, the matrix is generally scaled so that its elements vary less.

Following the discussion in [16], here we propose a scaling method to systematically address the numerical instability issue in time synchronization. Specifically, we scale the data $\tau[i]$ and $t[i]$, $n-w+1 \le i \le n$, with $\tau[n]$ and $t[n]$, respectively, resulting in an equivalent scaled signal model,

$$\tilde{\mathbf{x}}[n] = \tilde{\mathbf{H}}[n]\beta + \tilde{\mathbf{v}}[n], \tag{4}$$

where

$$\begin{aligned}
\tilde{\mathbf{x}}[n] &= \frac{1}{t[n]}\mathbf{x}[n] = [\frac{t[n-w+1]}{t[n]} \ \cdots \ 1]^T, \\
\tilde{\mathbf{H}}[n] &= [\tilde{\mathbf{h}}[n-w+1] \ \cdots \ \tilde{\mathbf{h}}[n]]^T = \mathbf{H}[n]\mathbf{S}[n], \\
\beta &= \frac{1}{t[n]}\mathbf{S}^{-1}[n]\theta, \\
\tilde{\mathbf{v}}[n] &= \frac{1}{t[n]}\mathbf{v}[n],
\end{aligned}$$

and the scaling matrix $\mathbf{S}[n]$ is a diagonal matrix and $\tilde{\mathbf{h}}[i]$, $n-w+1 \le i \le n$, are the vectors of $\mathbf{h}[i]$ scaled by $\mathbf{S}[n]$,

$$\mathbf{S}[n] = \mathrm{diag}\{1 \ \frac{1}{\tau[n]} \ \cdots \ \frac{1}{\tau^p[n]}\}, \tag{5}$$

$$\tilde{\mathbf{h}}^T[i] = \mathbf{h}^T[i]\mathbf{S}[n] = [1 \ \frac{\tau[i]}{\tau[n]} \ \cdots \ (\frac{\tau[i]}{\tau[n]})^p]. \tag{6}$$

Then, by minimizing the LS error criterion

$$\tilde{J}[n] = (\tilde{\mathbf{x}}[n] - \tilde{\mathbf{H}}[n]\beta)^T(\tilde{\mathbf{x}}[n] - \tilde{\mathbf{H}}[n]\beta), \tag{7}$$

the standard linear LSE at time n can be derived for the scaled signal model as

$$\hat{\beta}[n] = (\tilde{\mathbf{H}}^T[n]\tilde{\mathbf{H}}[n])^{-1}\tilde{\mathbf{H}}^T[n]\tilde{\mathbf{x}}[n]. \tag{8}$$

With such an estimator, a set of w observation data samples $\{\tau[i], t[i] : n-w+1 \le i \le n\}$ are collected first and then processed all at once at time n. Thus, the estimator (8) is widely known as batch estimator, especially in contrast to the sequential estimator presented in the next section. With sequential estimators, at time n, only the new observation data sample $\{\tau[n], t[n]\}$ is processed to update the estimator derived at time $n-1$, without processing any of the previous data $\{\tau[i], t[i] : i < n\}$; that is, the observation data sample is processed sequentially in time.

B. Sequential Least Squares Estimator

The sequential LSE for the unscaled signal model (3) can be found in [19]. Some simulation results of such an estimator can be found in [14], where the numerical stability issue in practical implementation is not considered. In this paper we derive the sequential LSE for the scaled signal model (4), which can be directly employed in implementation of time synchronization algorithms. Benefits of the sequential estimator, as compared to the batch estimator (8), are widely known, including reduced computational complexity, lower memory requirement, and faster computation in real-time applications [19].

The sequential LSE for the scaled signal model (4) can be derived directly from the unscaled sequential LSE by incorporating scaling updates at every time steps. For the scaled signal model (4), the sequential estimator recursively determines the LSE of the unknown parameter $\hat{\beta}[n]$ and a covariance matrix (following the derivation in [19])

$$\tilde{\mathbf{\Sigma}}[n] = (\tilde{\mathbf{H}}^T[n]\mathbf{W}[n]\tilde{\mathbf{H}}[n])^{-1}, \tag{9}$$

where $\mathbf{W}[n]$ is a diagonal weighting matrix,

$$\mathbf{W}[n] = \mathrm{diag}\{\frac{1}{\lambda} \ \frac{1}{\lambda^2} \ \cdots \frac{1}{\lambda^n}\}. \tag{10}$$

The parameter λ is a forgetting factor, $0 < \lambda < 1$, which is commonly used in sequential LS algorithms. By using the forgetting factor, previous data samples are exponentially down-weighted, effectively limiting the influence of the earlier data and allowing the estimator to react more quickly to model changes [19].

It is important to note that by using the sequential estimator, the estimation of $\hat{\beta}[n]$, $\tilde{\mathbf{\Sigma}}[n]$, and the minimum LS error $\tilde{J}_{\min}[n]$ is based on $\tilde{\mathbf{x}}[n]$ and $\tilde{\mathbf{H}}[n]$ with a window size $w = n$, i.e., the data $\tau[i]$ and $t[i]$, $1 \le i \le n$, scaled by $\tau[n]$ and $t[n]$ as in (4). To determine $\hat{\beta}[n]$, $\tilde{\mathbf{\Sigma}}[n]$, and $\tilde{J}_{\min}[n]$ recursively with the sequential estimator, we need to determine the estimators of the unknown parameter $\hat{\beta}_n[n-1]$, the covariance $\tilde{\mathbf{\Sigma}}_n[n-1]$, and the minimum LS error $\tilde{J}_{\min,n}[n-1]$ at time $n-1$ that are derived based on the data $\tau[i]$ and $t[i]$, $1 \le i \le n-1$, but scaled by $\tau[n]$ and $t[n]$. It can be easily verified that

$$\begin{aligned}
\hat{\beta}_n[n-1] &= \frac{1}{t[n]}\mathbf{S}^{-1}[n]\hat{\theta}[n-1] \qquad (11)\\
&= \frac{t[n-1]}{t[n]}\mathbf{S}_n[n-1]\hat{\beta}[n-1],\\
\tilde{\mathbf{\Sigma}}_n[n-1] &= \mathbf{S}^{-1}[n]\mathbf{\Sigma}[n-1]\mathbf{S}^{-1}[n] \qquad (12)\\
&= \mathbf{S}_n[n-1]\tilde{\mathbf{\Sigma}}[n-1]\mathbf{S}_n[n-1],\\
\tilde{J}_{\min,n}[n-1] &= (\frac{t[n-1]}{t[n]})^2\tilde{J}_{\min}[n-1],
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{\Sigma}[n] &= (\mathbf{H}^T[n]\mathbf{W}[n]\mathbf{H}[n])^{-1},\\
\mathbf{S}_n[n-1] &= \mathbf{S}^{-1}[n]\mathbf{S}[n-1]\\
&= \mathrm{diag}\{1 \ \frac{\tau[n]}{\tau[n-1]} \ \cdots \ (\frac{\tau[n]}{\tau[n-1]})^p\}.
\end{aligned}$$

Thus, following the derivation for the unscaled signal model in [19], the sequential estimator for the scaled signal model (4) can be summarized as follows:

Scaling Update:

$$\begin{aligned}
\hat{\beta}_n[n-1] &= \frac{t[n-1]}{t[n]}\mathbf{S}_n[n-1]\hat{\beta}[n-1], \qquad (13)\\
\tilde{\mathbf{\Sigma}}_n[n-1] &= \mathbf{S}_n[n-1]\tilde{\mathbf{\Sigma}}[n-1]\mathbf{S}_n[n-1],\\
\tilde{J}_{\min,n}[n-1] &= (\frac{t[n-1]}{t[n]})^2\tilde{J}_{\min}[n-1].
\end{aligned}$$

Estimator Update:

$$\hat{\beta}[n] = \hat{\beta}_n[n-1] + \tilde{\mathbf{K}}[n](1 - \tilde{\mathbf{h}}^T[n]\hat{\beta}_n[n-1]), \qquad (14)$$

where

$$\begin{aligned}
\tilde{\mathbf{K}}[n] &= \frac{\tilde{\mathbf{\Sigma}}_n[n-1]\tilde{\mathbf{h}}[n]}{\lambda^n + \tilde{\mathbf{h}}^T[n]\tilde{\mathbf{\Sigma}}_n[n-1]\tilde{\mathbf{h}}[n]},\\
\tilde{\mathbf{h}}[n] &= [1 \ 1 \ \cdots \ 1]^T.
\end{aligned}$$

Covariance Update:

$$\tilde{\mathbf{\Sigma}}[n] = (\mathbf{I} - \tilde{\mathbf{K}}[n]\tilde{\mathbf{h}}^T[n])\tilde{\mathbf{\Sigma}}_n[n-1]. \qquad (15)$$

Minimum LS Error Update:

$$\tilde{J}_{\min}[n] = \tilde{J}_{\min,n}[n-1] + \frac{(1 - \tilde{\mathbf{h}}^T[n]\hat{\beta}_n[n-1])^2}{\lambda^n + \tilde{\mathbf{h}}^T[n]\tilde{\mathbf{\Sigma}}_n[n-1]\tilde{\mathbf{h}}[n]}. \qquad (16)$$

It is important to note that no matrix inversions are required in the sequential algorithm, which significantly reduces computational complexity as compared to the batch estimator (8). To initialize the sequential algorithm, $\hat{\beta}[0]$ may be determined with a batch estimator using initial w data samples while $\tilde{J}_{\min}[0]$ and $\tilde{\mathbf{\Sigma}}[0]$ may be initialized with 0 and a large diagonal matrix, for example, $10^5\mathbf{I}$, respectively. We may also initialize with $\hat{\beta}[0] = \mathbf{0}$ [19]. After a burn-in period n_0, the estimator will converge with little biasing effect of the initial values. In long-term monitoring applications, the time index n increases indefinitely and thus λ^n in $\tilde{\mathbf{K}}[n]$ in (14) decreases indefinitely, causing numerical instability. Therefore, it is necessary to reinitialize the sequential algorithm periodically after n becomes too large, e.g., 10^3, by resetting n to 1, $\hat{\beta}[0]$ to the current estimate, $\tilde{J}_{\min}[0]$ to 0, and $\tilde{\mathbf{\Sigma}}[0]$ to $10^5\mathbf{I}$.

C. Energy-Efficient Fast Initialization Scheme

An initialization stage is required in both of the batch and sequential estimators; that is, with a window size of w, the batch estimator requires a collection of w data samples to start the estimation process while the sequential estimator requires an additional n_0 data samples for the burn-in period. In continuous monitoring applications, large data sampling interval will make the initialization process very long. For example, assuming a duty-cycling period T of 15 minutes with 14 minutes of sleep period T_s and 1 minute of active period T_a, if the initialization stage is 10 data samples' long, sensor nodes need to stay awake for 150 minutes initially before starting duty-cycling with a synchronized time. Such a requirement puts a major burden on the energy resource of sensor nodes that are often powered by battery or solar cells in harsh environmental conditions. In this section, we propose a novel initialization scheme to speed up the initialization process and significantly improve energy efficiency.

In designing the new scheme, it is important to note that data sampling for time synchronization does not need to be strictly periodic. The key idea of the new scheme is to use a small initial sampling period, then exponentially expand the sampling period until it reaches the regular duty-cycling period. Such a scheme makes it possible to start duty-cycling during the initialization stage to significantly reduce energy consumption. More specifically, a small initial sampling period T_0, e.g. 1 s, is employed at the beginning to collect $w+n_0$ data samples for initial estimation and burning-in. Then, the estimation algorithm continues with the sampling period increased by a times, e.g., 2 or 3 times. After the new sampling period is used to collect n_1 samples, it is increased again by a times to collect n_1 additional samples while the estimation algorithm continues. Such an exponential expansion of the sampling interval continues until the sampling period reaches the regular duty-cycling period and sensor node starts to follow the regular duty-cycling schedule. By gradually increasing the sampling period, sensor nodes are able to maintain synchronized time with adequate accuracy during the initialization stage. Thus, sensor nodes can start duty-cycling as soon as the current sampling period T_i is greater than the required active period T_a for data sampling; for example, stay active for T_a time

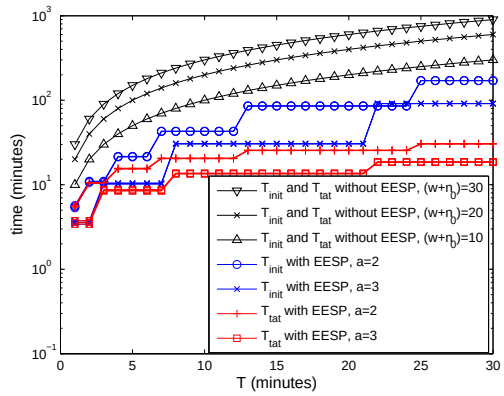

Fig. 2. Comparison of $T_{\rm init}$ and $T_{\rm tat}$ with and without EESP.

units, then sleep for $T_i - T_a$ time units. We refer to the proposed method as the exponential expansion of sampling period (EESP) initialization scheme and the parameter a as the exponential expansion parameter.

Using the EESP initialization scheme, the time required to reach the regular duty-cycling schedule is

$$T_{\rm init} = (w + n_0)T_0 + n_1 T_0 \frac{a^{m+1} - a}{a - 1}, \tag{17}$$

where

$$m = \text{round}(\log_a(T/a))$$

with the function $\text{round}(\cdot)$ denotes the rounding operation. If $T_a > T_0$, the total active time can be determined as

$$T_{\rm tat} = (w+n_0)T_0 + n_1 T_0 \frac{a^{m_1+1} - a}{a - 1} + n_1(m - m_1)T_a, \tag{18}$$

where

$$m_1 = \lfloor \log_a(T_a/T_0) \rfloor$$

with the symbol $\lfloor \cdot \rfloor$ denotes the flooring operation; if $T_a \le T_0$,

$$T_{\rm tat} = (w + n_0)T_0 + n_1 T_a \frac{a^{m+1} - a}{a - 1}, \tag{19}$$

To better appreciate the benefits of the new initialization scheme, here we consider a simple example. Assume in a continuous monitoring application, $T = 15$ minutes, $T_a = 1$ minute, $(w + n_0) = 10$, $T_0 = 1$ s, $a = 3$, $n_1 = 5$. Then, $T_{\rm init} = 30.4$ minutes and $T_{\rm tat} = 13.4$ minutes with the EESP scheme and 150 minutes without it. More comparison results are shown in Fig. 2. The advantage of the EESP scheme is more significant for the larger values of T and $(w + n_0)$. Therefore, the EESP scheme is strongly desirable for sequential estimators, especially when the burn-in period n_0 is large; with the EESP scheme, the extra cost of sequential estimators due to a long burn-in period, as compared to batch estimators, tend to be negligible in practice.

D. Outlier Detection

The synchronization message inevitably experiences variable delays at both transmitter and receiver due to uncertain send, access, transmission, propagation and reception times [13]. Due to the random nature of such delays, outliers may occur inevitably in time-stamp measurements. Outliers may also occur due to undetected hardware and software bugs that are impossible to avoid completely in implementation. Reading of the clock time needs to be an atomic operation, which must complete without anything else being able to change the value during the operation [22]. However, the clock time is a summation of two counters' readings as in (1), which is extremely difficult to implement as an atomic operation. Simply disabling interrupt does not solve such a problem; extra safeguard measures are needed to address all possible scenarios, which is an extremely difficult task in practice. In addition, reading HTC shortly after wake-up may also give an incorrect result in some hardware platforms [17].

The outlier issue in time-stamp data must be effectively addressed, especially in duty cycling protocols. An incorrect time-stamp may result in misalignment of duty cycling schedule, causing failure of entire network. Many outlier detection methods are available in the statistics literature, including methods based on global distribution, local distribution, distance, and deviation [18]. In this section we present a few simple distance-based algorithms that can be used together with the LSE in the preceding sections.

An iterative minimum residual (IMR) algorithm is proposed in [20] to detect and reject outliers by iteratively searching for the minimum residual estimator among the LSE derived from different combinations of the data. With the IMR algorithm, outliers are eliminated one-by-one in an iterative way. For example, given a set of w data, we first derive the batch LSE based on all of the data. Second, we derive w batch LSE based on all possible combinations of the data, taking $w-1$ data at a time. Then, determine the best estimator in terms of minimum root-mean-square (RMS) LS residual error. The data not employed in the derivation of the best estimator is eliminated from the data set. This process continues iteratively with the reduced set of data until a certain criterion is met; for example, iterations have been conducted for a predefined number or the change in the minimum RMS residual error is less than a predefined tolerance δ, comparing to the previous iteration. Such an IMR algorithm can be used in the initialization stage of both batch and sequential estimators to reject outliers in the very first w data samples.

Starting from the $(w+1)$-th data, only the latest data may be examined sequentially for outlier detection as explained in the following. With the LSE at time n, $\hat{\beta}[n]$, an estimate of $t[n+1]$ may be determined for a given $\tau[n+1]$ from

$$\hat{t}[n+1] = t[n]\tilde{\mathbf{h}}_n^T[n+1]\hat{\beta}[n], \tag{20}$$

where $\tilde{\mathbf{h}}_n[n+1]$ is the vector $\mathbf{h}[n+1]$ scaled by $\mathbf{S}[n]$,

$$\begin{aligned}\tilde{\mathbf{h}}_n^T[n+1] &= \mathbf{h}^T[n+1]\mathbf{S}[n] \\ &= [1 \ \frac{\tau[n+1]}{\tau[n]} \ \cdots \ (\frac{\tau[n+1]}{\tau[n]})^p].\end{aligned}$$

The prediction error at time $n+1$ for the scaled signal model is then defined as

$$\epsilon[n+1] = t[n+1] - \hat{t}[n+1]. \tag{21}$$

Algorithm 1 The LS time synchronization algorithm with EESP and outlier detection at client node at time n

```
Sample the current time-stamp data {τ[n], t[n]}.
if n < w then
    Do nothing here.
end if
if n = w then
    Use IMR algorithm to detect and eliminate outlier.
    Determine a batch estimator β̂[n].
end if
if n > w then
    Use (22) to detect outlier sequentially.
    if {τ[n], t[n]} is outlier then
        Eliminate {τ[n], t[n]} from data sequence.
    else
        Determine a sequential estimator β̂[n].
    end if
end if
Use EESP to determine a new sampling interval.
```

The data $\{\tau[n+1],\, t[n+1]\}$ is considered an outlier if

$$|\epsilon[n+1]| \geq \min\{\epsilon_u,\, \max\{\epsilon_l,\, \epsilon_{\rm th}[n]\}\}, \tag{22}$$

where the threshold $\epsilon_{\rm th}[n]$ may be defined as three times or more of the RMS residual error at time n, $\epsilon_{\rm rms}[n]$, while ϵ_l and ϵ_u are the empirical lower and upper bounds of the residual errors, respectively, employed to improve the robustness of the algorithm in practical implementation. When batch estimators are used, the RMS residual error can be determined at every time step as

$$\epsilon_{\rm rms}[n] = t[n]\sqrt{\frac{1}{w}\tilde{J}_{\rm min}[n]}. \tag{23}$$

With the forgetting factor λ, the equivalent LS error criterion of sequential estimators is weighted by the diagonal weighting matrix (10) [19]. Thus, the RMS residual error of the sequential estimator can be determined as

$$\epsilon_{\rm rms}[n] = t[n]\sqrt{\frac{\lambda^n(1-\lambda)}{1-\lambda^n}\tilde{J}_{\rm min}[n]}. \tag{24}$$

In summary, a pseudo code of the LS time synchronization algorithm is provided in Algorithm 1, which integrates the batch and sequential LSE, EESP, and outlier detection algorithms presented in this paper.

IV. Implementation and Computational Complexity

In order to study the practicality of the algorithms presented in this paper, we implemented the LS time synchronization algorithms together with a modified version of FTSP [13] in the IRIS motes. In particular, we employ the MAC layer time-stamping method and the timing message exchange protocol of FTSP, but replace the native clock offset and skew estimation algorithm with the batch and sequential estimators described in Section III. The experimental results presented in this section is obtained in a simple experimental setup consisting of a pair of motes, a parent node and a child node. The two nodes, placed 5 m apart in an outdoor experimental site, run the modified FTSP protocol to synchronize the child's time to the parent's. A third data collection node, deployed near the pair, sends a clock inquiry message every 30 seconds; both parent and child time-stamp the arrival time of the message and report the time-stamps to the data collection node. The differences between the time stamps of each query, namely the time synchronization errors between the parent and child nodes, are calculated and stored in the data collection node's flash memory for postprocessing.

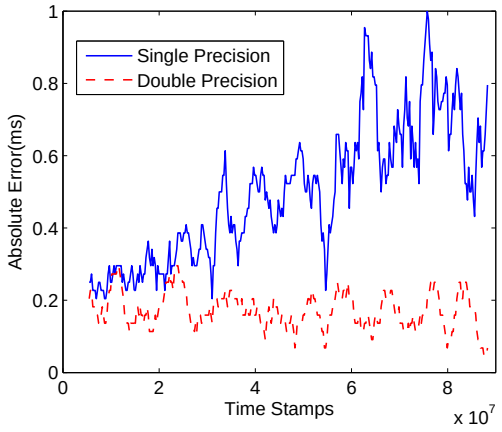

Fig. 3. Synchronization errors of the 1st-order batch estimator, implemented in the IRIS mote.

The compiler (i.e., avr-gcc) for the ATmega1281 MCU on IRIS only supports 32-bit single-precision floating-point computation. In the IEEE 754 standard, a single-precision binary floating-point number has 23-bit fraction component; however, the time stamps are 32-bit integers and the conversion from integer to single-precision floating-point number results in severe precision loss, especially when time stamps are large numbers. To address such a problem, we implemented 64-bit double-precision floating-point operations in the IRIS platform. Fig. 3 shows the synchronization errors of the 1st-order batch estimator with both single- and double-precision floating-point computations, implemented in IRIS. With a synchronization sampling period of 1 min, the time stamps grow from 0 to 9×10^7 in 6 hours. We can clearly observe the increasing trend of synchronization errors with the single-precision implementation. The 2nd-order estimators even fail to yield any valid result due to the divide-by-zero errors when implemented in single-precision floating-point.

To assess the computational complexity of the LS time synchronization algorithms, we have transplanted the code segments of the algorithms from TinyOS to C and evaluated the execution time of the C program in AVR Studio, which is an integrated development environment (IDE) for writing and debugging programs for Atmel MCUs including ATmega1281. The C program is first compiled with the avr-gcc compiler and the machine code is then executed on the AVR Simulator, which is a part of AVR Studio. Unlike the high-level network simulators such as NS2 and TOSSIM that are insufficient to measure the execution time of a task or a code block,

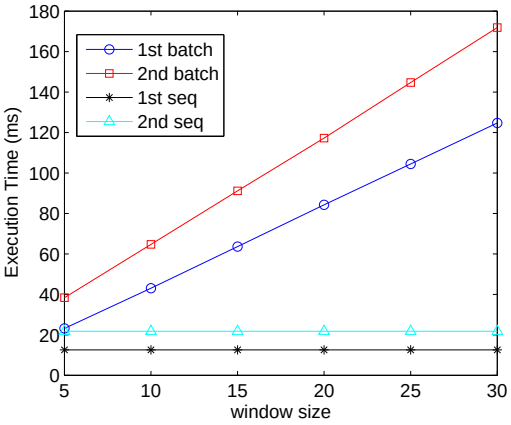

Fig. 4. Execution time of the LS time synchronization algorithms measured in AVR Simulator.

AVR Simulator is able to simulate the program instruction-by-instruction and the clock cycle of each instruction can be precisely recorded. When simulating the program, we set up a watch timer at a breakpoint before entering the function of an algorithm. After the simulator exits the function block and stops at another breakpoint, the watch timer records the execution time between the breakpoints.

Fig. 4 compares the computation time of the four estimators. From the results, we can observe that the computation time of batch estimators is a linear function of window size whereas the execution time of sequential estimators is constant, independent of the forgetting factor (and window size), since only one time stamp is processed at a time with such estimators. On the other hand, the computation time of the 2nd-order estimators is only slightly higher than the 1st-order estimators. Computational complexity of an algorithm is directly related to its energy efficiency, which is one of the major design considerations of resource-constrained WSN. The energy cost of each estimator can be obtained by multiplying the execution time by the average power consumption of a specific MCU, which is typically provided by manufacturers in terms of average current draw and operating voltage of the MCU.

In the next two sections, we present a comprehensive performance evaluation of the LS time synchronization algorithms using extensive measurement-based simulation results.

V. Measurement and Data Collection System

To study the behavior of the clock of typical sensor nodes, we have conducted extensive measurements using a simple experimental setup. The sensor network platform used in this research is the IRIS mote, a widely used low-cost sensor network platform [6]–[8], [21]. The measurement data are used in various simulations to evaluate the performance of time synchronization algorithms in realistic scenarios.

Our measurement and data collection system consists of three sensor nodes, among which one node periodically broadcasts beacon messages while the other two nodes receive the messages and record their arrival time. In the implementation, we employ the MAC-layer time-stamping technique to eliminate the uncertainty in the reception time. All sensor nodes are stationary during measurement. Time-stamp data are saved in the local flash memory; data are retrieved through serial port after measurement. The time-stamp data are indexed by the sequence number of the beacon messages. In the post-processing stage, the time synchronization process between two sensor nodes is simulated using the time-stamp data collected using our measurement system; that is, the time-stamp data from one receiving node is used as the time τ of the reference node and the data from the other receiving node as the time t of the client node. Then, a time synchronization algorithm can be applied to the data to synchronize the time of the client node to the reference node's as presented in Section III. Such a measurement-based simulation method can be used to conveniently evaluate the performance of time synchronization algorithms employed in a wide range of broadcast-based time synchronization methods such as RBS and FTSP. Furthermore, such a measurement-based simulation method is especially useful when comparing the performance of different algorithms since the same set of measurement data can be used repeatedly for all algorithms, which ensures the fairness of the comparison.

As discussed in Section II, the highest HTC time resolution may be achieved in the IRIS mote by setting the prescaling factor to 1, with which a time synchronization accuracy in the order of a few microseconds can be achieved. However, energy consumption of sensor nodes in sleep mode is directly related to the HTC time resolution since the 8-bit HTC interrupts faster with a higher time resolution. Thus, there is a trade-off between time synchronization accuracy and energy-efficiency. Targeting at low-power continuous monitoring applications such as environmental monitoring, here we employ a prescaling factor of 8 that results in a HTC time resolution of 0.25 ms. In our measurement system, the broadcasting period of the beacon messages, i.e., the sampling period of the measurement system, is set to 4 s. Thus, with a 128 kB flash memory on each node, we are able to collect about 10^5 data samples continuously for up to 11 hours. With such a sequence of measurement data, a new data sequence of lower sampling rate can be constructed in the post-processing stage through proper decimation.

Frequency drifts and aging effects of oscillators are affected by environmental conditions [23]. Thus, here we consider three typical scenarios, including indoor same condition (ISC), outdoor same condition (OSC), and outdoor different condition (ODC). In the ISC, all sensor nodes are placed inside an air-conditioned building. In the OSC, all sensor nodes are placed directly under the sun without any shade. In the ODC, sensor nodes are placed in an environment where surrounding trees form different dynamic shading patterns for the nodes with the sun movement. We start outdoor measurements in the afternoon to capture data during both daytime and nighttime. To ensure the quality of the measurement data, all experiments are repeated several times with different combinations of motes, selecting 3 motes from a pool of about 20, and the data from the repeated measurements are cross-checked carefully for consistency. Through such a process, we indeed found and eliminated some abnormal data sets caused by hardware anomalies in some of the motes. Sample measurement data

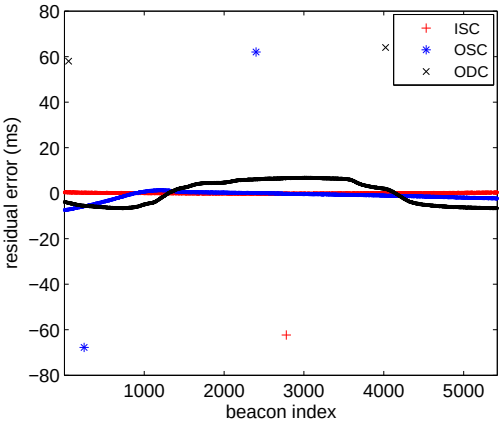

Fig. 5. Residual errors of three measurement data sequences after linear regression with a straight line.

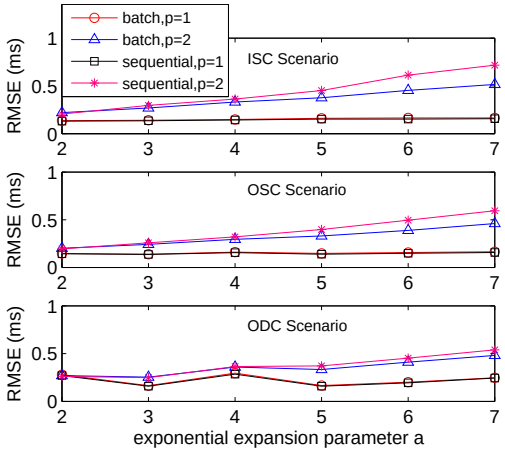

Fig. 6. Performance of the EESP algorithm versus parameter a.

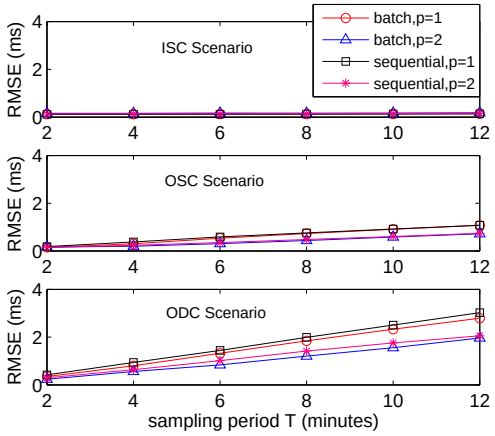

Fig. 7. Performance of the LSE versus sampling period.

are shown in Fig. 5. To better observe the nonlinear clock drifting effect, here we only plot the residual errors of each data sequence after linear regression with a straight line. The ODC scenario shows highest drifting effect as expected. Also, outliers are evident in the data, which is quite common in motes as discussed in Section III-D.

VI. Measurement-based Simulation Results

In this section, we study the performance of the proposed time synchronization algorithms through a number of measurement-based simulation results. The parameters of EESP and outlier detection algorithms are tuned through extensive simulations; for example, burn-in period $n_0 = 30$, exponential expansion parameter $a = 3$, the number of samples collected with each intermediate sampling period $n_1 = 5$, initial sampling period $T_0 = 4\,\text{s}$ (which is determined by the measurement system configuration), outlier detection lower bound $\epsilon_l = 8\,\text{ms}$ and upper bound $\epsilon_u = 48\,\text{ms}$. In addition, in the simulations, we use window size $w = 10$, forgetting factor $\lambda = 0.8$, and regular sampling period $T = 5$ minutes, if not otherwise specified. Various time synchronization algorithms are compared in terms of their root-mean-squared prediction error (RMSE) performance, which is determined as the root-mean-squared (RMS) value of the prediction errors, defined in (21), over the measurement data sequence.

A. Performance of EESP Algorithm

To study the effectiveness of the EESP algorithm, we conduct a series of simulations in all three measurement scenarios. The RMSE performances of the LS estimators in the initialization stage are presented in Fig. 6. From the results we can clearly observe that all estimators are quite insensitive to the parameter a, although a faster expansion tends to introduce larger estimation errors as expected intuitively. For example, even with 7 times expansion of the sampling period at each step, all estimators achieve less than $1\,\text{ms}$ RMSE, which may well be an adequate performance to enable duty-cycling in many practical applications.

It is interesting to note that the 1st-order estimators have better performance than the 2nd-order ones. Such an observation is due to the fact that clock drifting effects are negligible within a short period of time; only over a long time period, clock drifting effects are significant. On average, sampling intervals are very small during the initialization stage. Thus, samples within a short window (e.g., $w = 10$ in the simulations) are more accurately modeled with the 1st-order polynomial model. In addition, the performances of batch and sequential estimators are comparable; the slight difference between them is because of the use of a fixed window size in batch estimators and a forgetting factor in sequential estimators to down-weight the previous data samples.

B. Effects of Sampling Period

Sampling period is an important system configuration parameter in practical implementations, especially with duty-cycling. In duty-cycling protocols, the sampling period determines how often a sensor node needs to wake up for synchronization purposes, which directly relates to the energy efficiency of the system. Performances of the estimators versus sampling period are shown in Fig. 7. In the ISC scenario, the performance of all four estimators are very close to each other

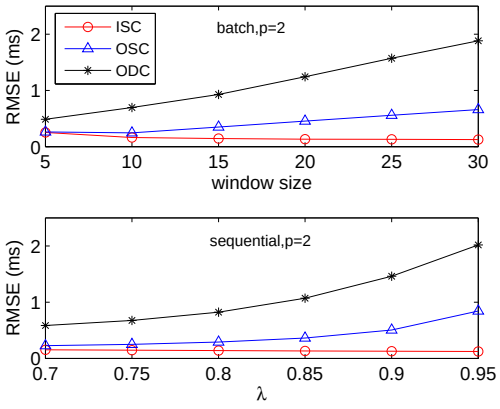

Fig. 8. Performance of the 2nd-order LSE with $T = 5$ minutes.

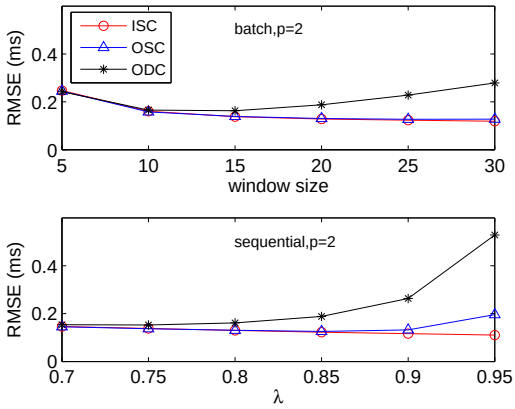

Fig. 9. Performance of the 2nd-order LSE with $T = 1$ minute.

and the sampling period does not have significant effect. However, in outdoor scenarios, especially in ODC, performances of all four estimators deteriorate steadily as sampling period increases. Such an observation is easily justified intuitively, considering that there are very minimum clock drifting effects in controlled indoor environments while in random outdoor deployments, clock frequency drifts much more significantly, especially over a long period of time. Thus, there is a trade-off between time synchronization accuracy and energy efficiency; in practice, the sampling period may be maximized subject to a design requirement of time synchronization accuracy.

From the results, we can also observe that the 2nd-order estimators have better performance than the 1st-order estimators when the sampling period is larger, especially in the ODC scenario, justifying the use of the higher order estimators.

C. Effects of Window Size and Forgetting Factor

To study the effects of window size and forgetting factor, we present the RMSE performances of the 2nd-order estimators in Fig. 8 and Fig. 9. Similar to the preceding discussion, we can observe that in the ISC scenario, the performance of the estimators tends to improve, although slightly, as window size or forgetting factor increases because of the inherent linearity of the clock behavior. However, a smaller window size or forgetting factor is preferred in outdoor scenarios.

By comparing Fig. 8 and Fig. 9, we can also conclude that when the sampling period is smaller, consecutive data samples are more closely correlated and thus a little larger window size is preferred; for example, the best overall performance is achieved with $w = 5$ and $\lambda = 0.7$ when $T = 5$ minutes, but with $w = 10$ and $\lambda = 0.8$ when $T = 1$ minute.

D. Effects of Missing Data

To study the robustness of the proposed algorithms in realistic lossy communication conditions, we conduct a series of simulations of the 2nd-order estimators in the ODC scenario by varying the missing data rate. When the data is missing at time n, the latest estimator at time $n-1$ will be used to predict the time at the next time step $n+1$. From the

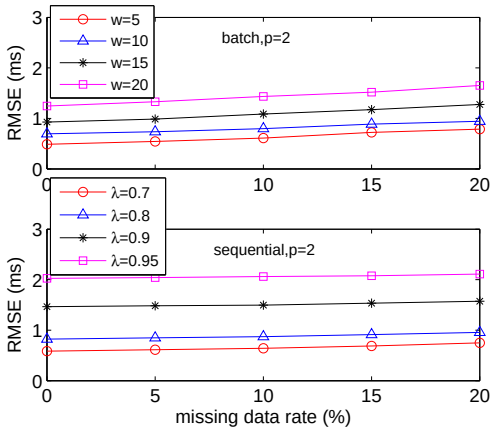

Fig. 10. Performance of the 2nd-order LSE with different missing data rate when $T = 5$ minutes.

results shown in Fig. 10 and Fig. 11, we can clearly observe that the estimators are remarkably robust in missing data conditions. For example, even with a missing data rate of 20%, the performance degradation is quite insignificant in all simulation configurations. In addition, even with missing data, a smaller window size or forgetting factor is preferred when the sampling period is relatively large, e.g., $T = 5$ minutes, as shown in Fig. 11. Better performance can be achieved with a larger window size or forgetting factor when the sampling period is smaller, e.g., $T = 1$ minute, as shown in Fig. 11.

VII. Conclusions

In this paper, we derived a set of LSE and related algorithms for time synchronization in continuous monitoring sensor network systems. The measurement-based simulation method employed in this research makes it convenient to study the performance of time synchronization algorithms in realistic scenarios. Through simulation results, we demonstrated the effectiveness of the proposed algorithms. In particular, we conclude that 1) the 2nd-order estimators are preferred in outdoor random deployment scenarios due to significant clock drifting effects that are inherently nonlinear over a long period of time, 2) the sequential estimators are able to achieve

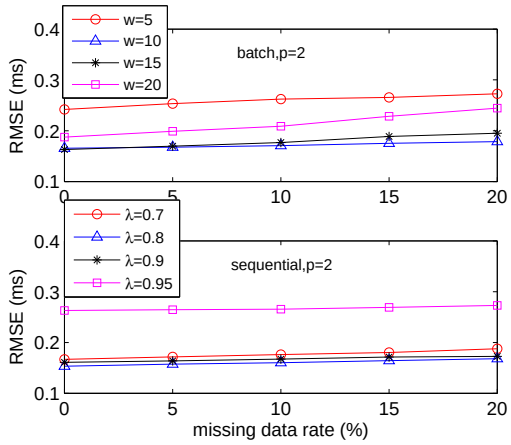

Fig. 11. Performance of the 2nd-order LSE with different missing data rate when $T = 1$ minutes.

comparable performances to the batch estimators with reduced computational cost, and 3) small sampling period, and small window size and forgetting factor are preferred in outdoor conditions, especially when the sampling period is large.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, Aug. 2002.

[2] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer*, pp. 41-49, Aug. 2004.

[3] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *IEEE Computer*, pp. 50-56, Aug. 2004.

[4] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A ZigBee-based home automation system," *IEEE Trans. on Consumer Electronics*, vol. 55, no. 2, pp. 422-430, May 2009.

[5] S.-H. Hong, B. Kim, and D.-S. Eom, "A base-station centric data gathering routing protocol in sensor networks useful in home automation applications," *IEEE Trans. on Consumer Electronics*, vol. 53, no. 3, pp. 945-951, Aug. 2007.

[6] J. Yang, C. Zhang, X. Li, Y. Huang, S. Fu, and M. Acevedo, "Integration of wireless sensor networks in environmental monitoring cyber infrastructure," *Wireless Networks*, Springer/ACM, DOI: 10.1007/s11276-009-0190-1, June 2009.

[7] V.C. Gungor and G.P. Hancke, "Industrial wireless sensor networks: challenges, design principles, and technical approaches," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4258-4265, Oct. 2009.

[8] K.A. Agha, M.-H. Bertin, T. Dang, A. Guitton, P. Minet, T. Val, and J.-B. Viollet, "Which wireless technology for industrial wireless sensor networks? The development of OCARI technology," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4266-4278, Oct. 2009.

[9] B. Lu and V.C. Gungor, "Online and Remote Motor Energy Monitoring and Fault Diagnostics Using Wireless Sensor Networks," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4651-4659, Oct. 2009.

[10] D.-S. Kim, S.-Y. Lee, K.-H. Won, D.-J. Chung, and J.-H. Kim, "Time-synchronized forwarding protocol for remote control of home appliances based on wireless sensor network," *IEEE Trans. on Consumer Electronics*, vol. 53, no. 4, pp. 1427-1433, Nov. 2007.

[11] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *Proc. of the 5th Symposium on Operating Systems Design and Implementation* (OSDI), Dec. 2002.

[12] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," *Proc. of the 1st Int'l Conference on Embedded Networked Sensor Systems* (SenSys), Nov. 2003.

[13] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," *Proc. of the 2nd Int'l Conference on Embedded Networked Sensor Systems* (SenSys), Nov. 2004.

[14] S. Raje and Q. Liang, "Time synchronization in network-centric sensor networks, " *Proc. of IEEE Radio and Wireless Symposium*, 2007.

[15] W. Ye, J. Heidemann, D. Estrin, "An energy-ecient MAC protocol for wireless sensor networks," *Proc. of the Int'l Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM), vol. 3, pp. 1567-1576, June 2002.

[16] K. E. Atkinson, *An introduction to Numerical Analysis*, 2nd edition, John Wiley & Sons, Inc., 1988.

[17] Atmel Co., *ATmega640/1280/1281/2560/2561 Preliminary*, Rev. L, available at http://www.atmel.com/, 2013.

[18] J. Han and M Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Elsevier Inc., 2006.

[19] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall PTR, 1993.

[20] X. Li, "An iterative NLOS mitigation algorithm for location estimation in sensor networks," *Proc. of The 15th IST Mobile & Wireless Communications Summit*, June 2006.

[21] Wireless Sensor Networks, MEMSIC Inc., available at http://www.memsic.com/wireless-sensor-networks/, 2013.

[22] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 7th edition, John Wiley & Sons, Inc., 2005.

[23] Symmetricom Inc., "Stochastic model estimation of network time variance," White Paper, 2003, available at http://www.ntp-systems.com/.