Ad Hoc Networks 11 (2013) 765–777

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Drift estimation using pairwise slope with minimum variance in wireless sensor networks

Kasım Sinan Yıldırım *, Aylin Kantarcı

Department of Computer Engineering, Ege University, İzmir, Turkey

ARTICLE INFO

Article history:
Received 24 February 2012
Received in revised form 4 September 2012
Accepted 5 September 2012
Available online 23 September 2012

Keywords:
Wireless sensor networks
Time synchronization
Least-squares regression

ABSTRACT

Time synchronization is mandatory for applications and services in wireless sensor networks which demand common notion of time. If synchronization to stable time sources such as Coordinated Universal Time (UTC) is required, employing the method of flooding in order to provide time synchronization becomes crucial. In flooding based time synchronization protocols, current time information of a reference node is periodically flooded into the network. Sensor nodes collect the time information of the reference node and perform least-squares regression in order to estimate the reference time. However, least-squares regression exhibits a poor performance since sensor nodes far away from the reference node collect the time information with large deviations. Due to this fact, the slopes of their least-squares line exhibit large errors and instabilities. As a consequence, the reference time estimates of these nodes also exhibit large errors.

This paper proposes a new slope estimation strategy for linear regression to be used by flooding based time synchronization protocols. The proposed method, namely Pairwise Slope With Minimum Variance (PSMV), calculates the slope of the estimated regression line by considering the pairwise slope between the earliest and the most recently collected data points. The PSMV slope is less affected by the large errors on the received data, i.e. it is more stable, and it is more computationally efficient when compared to the slope of the least-squares line. We incorporated PSMV into two flooding based time synchronization protocols, namely Flooding Time Synchronization Protocol (FTSP) and PulseSync. Experimental results collected from a testbed setup including 20 sensor nodes show that PSMV strategy improves the performance of FTSP by a factor of 4 and preserves the performance of PulseSync in terms of synchronization error with 40% less CPU overhead for linear regression. Our simulations show that these results also hold for networks with larger diameters and densities.

1. Introduction

Wireless sensor networks (WSNs) consist of tiny, cheap and low-power sensor nodes that have the ability of sensing the environment. Each sensor node is equipped with a hardware clock which counts oscillations of a quartz crystal. Due to their low-end crystals, the hardware clocks frequently drift apart which leads to loss of synchronization between the sensor nodes. This situation is problematic for applications and protocols which depend on a synchronized notion of time. For instance, synchronous power on and shutdown of the radios reduce energy consumption where energy is a scarce resource for battery-powered sensor nodes. Such time coordinated actions can be performed if the clocks of the sensors are well synchronized. Thus, time synchronization is mandatory for the correct and efficient operation of the WSNs [1–3].

* Corresponding author.
E-mail addresses: sinan.yildirim@ege.edu.tr (K.S. Yıldırım), aylin.kantarci@ege.edu.tr (A. Kantarcı).

1570-8705/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2012.09.003

In the course of time synchronization in WSNs, the hardware clock values of the nodes and the information carried by the synchronization messages are considered to calculate the global (common) time in the network. Most of the time synchronization protocols in the literature [4–8] use least-squares regression in order to establish a linear relationship between the hardware clocks of the nodes and the global clock in the network. Thus, sensor nodes can predict future global clock values without communicating frequently. However, least-squares regression exhibits poor performance if the data points have large errors [9–11]. Such a situation may decrease the performance of time synchronization protocols, especially the ones that use flooding. If synchronization to stable time sources such as Coordinated Universal Time (UTC) time is required, employing the method of flooding in order to provide time synchronization becomes crucial [12].

As an example, in Flooding Time Synchronization Protocol (FTSP) [4] which is the de facto standard time synchronization protocol in WSNs, current time information of a dynamically elected reference node is periodically flooded into the network. Due to the nature of flooding and waiting times at each sensor node, the sensor nodes which are far away from the reference node collect the estimated reference time with large errors, which leads to large errors and instabilities in the slope of their least-squares regression line. As a consequence, the reference time estimates of these nodes which are calculated using least-squares exhibit substantially large synchronization errors when compared to that of nearby nodes to the reference. This situation has a negative impact on the scalability of FTSP.

In this paper, we present a new slope estimation method for linear regression to be used by flooding based time synchronization protocols in WSNs. The proposed method, namely Pairwise Slope With Minimum Variance (PSMV), considers only the pairwise slope between the earliest and the most recently collected data points instead of considering all pairwise slopes as in least-squares. The PSMV slope is less affected by the large errors on the received data, i.e. it is more stable, and it is more computationally efficient when compared to the slope of the least-squares line. For the implementation, we considered two flooding based time synchronization protocols, FTSP and PulseSync [7], and incorporated PSMV into the source code of these protocols. Experimental results collected from a testbed setup including 20 sensor nodes show that PSMV strategy improves the performance of FTSP by a factor of 4 and preserves the performance of PulseSync in terms of synchronization error with 40% less CPU overhead for linear regression. Our simulations show that these results also hold for networks with larger diameters and densities.

The remainder of this paper is organized as follows: Section 2 describes related work on time synchronization. In Section 3, we describe our system model. We present and analyze the packet timestamping mechanism in WSNs and the least-squares method for establishing linear relationship between the clocks of sensor nodes in Section 4 and 5, respectively. The flooding based time synchronization protocols FTSP and PulseSync are analyzed in Section 6. We present and analyze the proposed drift estimation method in Section 7. Implementation details and experimental results are given in Section 8. We also present simulation results for networks with larger diameters and densities in Section 9. Finally, we present our conclusions and discuss future work in Section 10.

2. Related work

Time synchronization protocols for WSNs generally deal with making the clocks of the nodes as close as possible towards the precision requirement of the sensor network. In order minimize energy consumption, they try to establish relationships between the clocks of the nodes and estimate future clock values without communicating frequently. There are a lot of protocol based studies in the literature which use linear regression for the establishment of a linear relationship [13,4,5,7,8,12].

In Reference Broadcast Synchronization (RBS) [13], a sender node transmits a reference packet for its neighbors. The receiver nodes record the time at which they received that packet. Then, the receivers exchange their observations to learn about the clocks of the other nodes. Linear regression is used to compensate the effects of clock drifts. One major problem of RBS is its high message complexity. For a single-hop network of n nodes, $\mathcal{O}(n^2)$ message exchange is required. In addition, it was not designed for large multi-hop networks.

Flooding Time-Synchronization Protocol (FTSP) [4] synchronizes the whole network by electing a leader node based on the smallest node identifier which serves as the source of reference time. Whenever a slave node receives an up-to-date synchronization message, it stores the received clock value of the reference node and uses linear regression in order to establish a linear relationship between its hardware clock and the clock of the reference node. Using this relationship, each sensor node can predict future clock values of the reference node without communicating frequently. The nodes broadcast their predicted clock values of the reference node to their neighbors for the synchronization of the whole network. FTSP is the de facto standard time synchronization protocol in WSNs and it is used as a benchmark by most of the studies in the literature [6–8,12,14].

In Rapid Time Synchronization (RATS) [5], a designated reference node periodically floods its hardware clock value into the network. Each node converts the timestamp of the synchronization messages from the reference node's time base to its local time to calculate the clock offset and uses linear regression to compensate clock drifts. However, RATS cannot maintain network-wide synchronization if the network is partitioned due to a root failure.

PulseSync [7] propagates time information from a reference node as fast as possible. Apart from the *fast flooding* approach, each nodes use a linear regression table to estimate the clock of the reference node. Although this protocol outperforms FTSP drastically, rapid-flooding can also be very slow due to neighborhood contention since nodes cannot propagate their messages until their neighbors have finished their transmissions [15,12].

Temperature Compensated Time Synchronization (TCTS) [8] uses the temperature sensor to calibrate the frequency of the oscillator which is majorly affected by the

environmental temperature changes. This strategy allows TCTS to increase the resynchronization period without losing synchronization accuracy and thus saves energy. The nodes perform linear regression to estimate their offset and frequency errors with respect to a reference node.

In [12], in order to achieve more accurate global time synchronization with flooding, a node chooses the synchronization to the neighbor which offers the greatest frequency stability. Linear regression is used in the calculations for determining the most stable path from the reference node.

The protocols we mentioned above employ the method of least-squares for linear regression. Especially in flooding based protocols, there may be nodes which collect time information with large errors. Unfortunately, the method of least-squares exhibits poor performance when the collected data points have large errors [10,11].

3. System model

In this section, we introduce the system model that we use throughout our analysis in the rest of the paper. We model a WSN as a graph $G = (V,E)$ which consists of a set of n sensor nodes represented by vertex set $V = \{1, \ldots, n\}$. The nodes are located in the Euclidean plane $\mathbb{R}^2$ and communicate with each other through *wireless broadcasts* by exchanging messages. Bidirectional communication links between nodes are represented by the edge set $E \subseteq V \times V$. Any node u can communicate with any node v to which node u is directly connected (inside its broadcast region). These nodes are referred as the neighbors of node u which is denoted by $N_u = \{v \in V | \{u, v\} \in E\}$. The *distance* between any nodes u and v is defined as the number of edges on the shortest path between those two nodes in the graph G. The *diameter* of the graph G is the maximum distance between any two nodes.

Each sensor node u in the system is equipped with an unmodifiable *hardware clock* $H_u()$. We denote the reading of the hardware clock of the node u at real-time t as

$$H_u(t) = H_u(0) + \int_0^t h_u(\tau) d\tau \tag{3.1}$$

where $H_u(0)$ represents *hardware clock offset* and $h_u(\tau)$ represents *hardware clock rate* at time τ. External crystal oscillators in sensor nodes are used as the clock source for their hardware clocks and the frequencies of the oscillators exhibit a drift between 30 and 100 ppm.[1] Therefore, we assume that hardware clocks have *bounded drifts* such that

$$\forall t : 1 - \varepsilon \leqslant h_u(t) \leqslant 1 + \varepsilon, \qquad 0 < \varepsilon \ll 1. \tag{3.2}$$

A time synchronization protocol allows nodes to estimate the hardware clock of a reference node $r \in V$, which acts as a time reference for the whole network. The estimated clock value of the reference node from node v's perspective at time t is denoted by $H_v^r(t)$ and this value represents the common notion of time for this node. The goal of time synchronization is to minimize the differences of the estimated reference clock values of the nodes, i.e. the *clock skew*. The *global skew* at any time t is defined as the largest clock skew between any two nodes, i.e. $\max_{v,u \in V} \{|H_v^r(t) - H_u^r(t)|\}$. Similarly, the *local skew* at any time t is defined as the largest clock skew between neighboring nodes, i.e. $\max_{v \in V, u \in \mathcal{N}_v} \{|H_v^r(t) - H_u^r(t)|\}$.

[1] ppm: parts per million, i.e., 10^{-6}.

4. MAC layer timestamping in WSNs

Time synchronization protocols are based on exchanging time information between sensor nodes through synchronization messages. For any message sent, the time that passes until the recipient node receives it is referred as *message delay* and it is composed of deterministic and non-deterministic components [4,13,16,17]. The non-deterministic components of the message delay directly effect the synchronization error of the time synchronization protocols in sensor networks. It is shown that MAC layer timestamping reduces the effects of these components [4,16]. In order to analyze the synchronization errors occurred with MAC layer timestamping, we consider the operation of *packet-level time synchronization* mechanism [18] implemented in TinyOS, which is semantically equivalent to *elapsed time on arrival* (ETA) [5] primitives.

4.1. A simple synchronization scenario between two nodes

Consider the scenario such that any node u decides to send the value of its hardware clock to node v at time t_1. For simplicity, it is assumed that each synchronization packet has *clock* and *eventTime* fields. Hence, node u records the value of its hardware clock at time t_1, i.e. $H_u(t_1)$, both in the *clock* field and the *eventTime*field of a synchronization packet. When the communication layer accesses the wireless channel and the first bytes of this packet start being transmitted over the communication medium at t_2, the value of $H_u(t_2) - H_u(t_1)$, which is the *age* of the send event in the sender node's hardware clock ticks, is written into the *eventTime* field of that packet. When the first bytes of the packet are received by node v at t_3, the packet is timestamped with the hardware clock reading of that node, i.e. $H_v(t_3)$, and that timestamp is stored in the system. The node v can query the communication layer at t_4 in order to get an estimate about its hardware clock reading at time t_1. For this request, the communication layer returns the following value:

$$eventTime = H_v(t_3) - (H_u(t_2) - H_u(t_1)). \tag{4.1}$$

The node v can also acquire the *clock* field of the packet which holds the value $H_u(t_1)$.

4.2. Analysis

Let t' be the time such that $H_v(t') = eventTime$. If we assume that the hardware clock rates of nodes u and v remain constant in the time interval $[t_1, t_4]$,[2] $H_v(t')$ can be rewritten as follows:

[2] This assumption is realistic since the clock rates change slowly in real hardware through the time interval which starts at the sending time and ends at the reception time of a synchronization packet [19].

$$\begin{aligned} H_v(t') &= H_v(t_3) - H_u(t_2) + H_u(t_1) \\ &= H_v(t_3) - (t_2 - t_1)h_u \\ &= H_v(t_1) + (t_3 - t_1)h_v - (t_2 - t_1)h_u \\ &= H_v(t_1) + (t_3 - t_2)h_v + (t_2 - t_1)(h_v - h_u). \end{aligned} \tag{4.2}$$

From the equality (4.2), it follows that:

$$t_1 = t' - (t_3 - t_2) - (t_2 - t_1)(h_v - h_u)\frac{1}{h_v}. \tag{4.3}$$

In the equation above, $(t_3 - t_2)$ can be assumed to be deterministic, however, $(t_2 - t_1)$ consists of deterministic and variable components of the message delay [4,13,16,17]. In [20,21], the variable component of $(t_2 - t_1)$ is modeled as Gaussian random variable due to central limit theorem because it is thought to be the addition of numerous independent random processes. It is also shown in [13] that the variable component of the message delay can be modeled as Gaussian distributed random variables with 99.8% confidence. Thus, we assume that the variable component of $(t_2 - t_1)$ is a normally distributed random variable.

Let the deterministic part of $(t_2 - t_1)$ be represented by γ and let the variable portion of $(t_2 - t_1)$ be a normally distributed random variable with expected value μ and variance σ^2. Let the value of μ_{r_v} and $\sigma^2_{r_v}$ be as follows:

$$\mu_{r_v} = (t_3 - t_2) + (\gamma + \mu)(h_v - h_u)\frac{1}{h_v}, \tag{4.4}$$

$$\sigma^2_{r_v} = \sigma^2\left((h_v - h_u)\frac{1}{h_v}\right)^2. \tag{4.5}$$

Thus, the equality (4.3) can be rewritten as:

$$t_1 = t' - r_v \tag{4.6}$$

such that r_v is a random variable with $r_v \sim \mathcal{N}(\mu_{r_v}, \sigma^2_{r_v})$. Using this equality, it can be shown that $H_u(t_1)$ can be considered as an estimate for $H_u(t')$ as follows:

$$\widehat{H}_u(t') = H_u(t_1) = H_u(t') - h_u r_v = H_u(t') - e_v \tag{4.7}$$

such that $e_v \sim \mathcal{N}\left(\mu_{e_v} = \mu_{r_v} h_u, \sigma^2_{e_v} = \sigma^2_{r_v} h_u^2\right)$. The equality above shows that the estimate $\widehat{H}_u(t')$ has a systematic and a random error. The variance of the error e_v has great impact on the accuracy of this estimate.

5. Establishing a linear relationship between the clocks using least squares regression

Establishing a relationship between the clocks of nodes is an energy efficient method since the nodes can predict future clock values of the other nodes without exchanging messages frequently. In this section, we analyze the least-squares method which is a standard approach to establish a linear relationship between the clocks of sensor nodes in many time synchronization protocols.

We consider time synchronization between two nodes u, $v \in V$ which is based on periodical synchronization packet exchange. We assume that each node uses the packet time-stamping mechanism explained in the previous section. Hence, upon receiving a synchronization message from node u at time t, node v extracts its hardware clock reading at time t', i.e. $H_v(t')$, and a corresponding estimate of the hardware clock reading of node u at time t', i.e. $\widehat{H}_u(t')$. [3] Node v stores $H_v(t')$ and $\widehat{H}_u(t')$ as a pair (x, Y) in its *timestamp table* which has a capacity of N pairs to hold the most recent N time information. During time synchronization process, the node v uses least-squares regression which assumes an explicit linear relationship between x and Y as $Y = \alpha + \beta x - e_v$, where e_v is a random error which is normally distributed having mean μ_{e_v} and variance $\sigma^2_{e_v}$. Since the observed pairs are independent from each other, this model is suitable if we consider the equality (4.7). In least-squares regression, if we let $\bar{x} = \sum x_i/N$, $\overline{Y} = \sum Y_i/N$, $S_{xx} = \sum(x_i - \bar{x})^2$ and $S_{xY} = \sum(x_i - \bar{x})(Y_i - \overline{Y})$ by using the (x_i, Y_i) pairs for $i = 0, \ldots, N-1$ in the timestamp table, $\hat{\beta}_{reg}$ and $\hat{\alpha}_{reg}$ are computed using the pairs in the timestamp table as follows:

$$\hat{\beta}_{reg} = S_{xY}/S_{xx}, \tag{5.1}$$

$$\hat{\alpha}_{reg} = \overline{Y} - \hat{\beta}_{reg}\bar{x}. \tag{5.2}$$

The parameter $\hat{\beta}_{reg}$ is called the*slope* and the parameter $\hat{\alpha}_{reg}$ is called the *intercept* of the *estimated regression line*which is defined as follows:

$$\widehat{Y}_{reg} = \hat{\alpha}_{reg} + \hat{\beta}_{reg}x = \overline{Y} + \hat{\beta}_{reg}(x - \bar{x}). \tag{5.3}$$

It should be noted that $\hat{\beta}_{reg}$ is an estimate of the rate of the sender node's clock with respect to the receiver node's clock and it is the key factor in drift estimation. The probability distribution of $\hat{\beta}_{reg}$ is given as follows [22]:

$$\hat{\beta}_{reg} \sim \mathcal{N}\left(\beta, \frac{\sigma^2_{e_v}}{S_{xx}}\right). \tag{5.4}$$

Given a hardware clock reading $H_v(t)$ at any time t, node v uses the estimated regression line in order to estimate the clock of the node u at time t. This estimate, which is denoted by $H^u_v(t)$, is calculated as follows [22]:

$$\begin{aligned} H^u_v(t) &= \hat{\alpha}_{reg} + \hat{\beta}_{reg}H_v(t) \\ &= \alpha + \beta H_v(t) - e_{v^{est}}(t) = H_u(t) - e_{v^{est}}(t), \end{aligned} \tag{5.5}$$

$$e_{v^{est}}(t) \sim \mathcal{N}\left(\mu_{e_v}, \sigma^2_{e_v}\left(1 + \frac{1}{N} + \frac{(H_v(t) - \bar{x})^2}{S_{xx}}\right)\right). \tag{5.6}$$

$e_{v^{est}}$ represents the error distribution of the predicted clock value $H^u_v(t)$ by using least-squares regression.

Until a recent (x, Y) pair is collected and the new least-squares line is recalculated, the value of $H_v(t) - \bar{x}$ increases as time passes. It should be noted that whenever a recent (x, Y) pair is collected, the earliest pair is removed from the timestamp table, the received pair is stored in that table and the least-squares line is recalculated. Hence, the value of $H_v(t) - \bar{x}$ decreases upon the recalculation of the least-squares line. This situation leads us to the following fact if we consider the variance of the $e_{v^{est}}$.

Fact 5.1. *From the time at which node v calculated current least-squares line to the time at which the new least-squares line will be calculated upon receiving a new synchronization point, the variance of the estimated clock values increases.*

[3] Note that t' is the time such that $H_v(t')$ is equal to the value of the *eventTime* field of the packet received at time t.

Hence, the error of the predicted clock values grows as time passes at each sensor node.

Since node v may miss several synchronization points due to the corruption of signals by ambient noises or collisions, the regular message receipt would be interrupted. However, it is still quite reasonable to assume that there is an upper bound on the value of $(H_v(t)-\bar{x})/S_{xx}$, which will simplify our analysis in the next subsection. Hence, we assume that

$$1+\frac{1}{N}+\frac{(H_v(t)-\bar{x})^2}{S_{xx}} \leqslant \mathcal{F}_v. \tag{5.7}$$

6. Flooding based time synchronization by using least-squares

There are several studies in the literature [4,5,7] which use flooding mechanism in order to achieve network-wide time synchronization. In these flooding based approaches, a reference node, which is the source of time, floods its current time information into the network at every beacon interval B. Upon receiving synchronization messages, each receiver node gets a synchronization point, i.e. a (*local time-reference time*) pair, using the packet timestamping mechanism presented in Section 4. Nodes assume a linear relationship between their hardware clocks and the reference clock. Each node performs least-squares regression using N (local time, reference time) pairs to estimate its offset and rate difference to the reference clock. Each node broadcasts its estimate about the reference time for its neighbors in order to achieve network-wide synchronization.

In this section, we consider two flooding based time synchronization protocols, namely FTSP and PulseSync, in order to reveal the impact of least-squares regression on their estimation accuracy. Assume that FTSP is executed on a line of n nodes $v_0, v_1, \ldots, v_{n-1} \in V$ such that v_0 is the reference node. During the execution, v_1 receives synchronization messages from v_0 and it stores the collected synchronization points in its timestamp table. According to the equality (4.7), the received reference clock estimates suffer from an error $e_{v_1} \sim \mathcal{N}(\mu_{e_{v_1}}, \sigma^2_{e_{v_1}})$. Given a hardware clock reading at any time t, v_1 uses its least-squares line in order to estimate the clock of v_0. By using (5.6) and (5.7), this estimate suffers from an error err_{v_1}, which is as follows:

$$err_{v_1} \sim \mathcal{N}\left(\mu_{v_1}, \sigma^2_{v_1}\mathcal{F}_{v_1}\right). \tag{6.1}$$

Hence, the variance of the received estimates is *amplified* by a factor of $\mathcal{F}_{v_1}$. Similarly, during the execution of FTSP, v_2 receives synchronization messages from v_1. These messages carry the estimated clock value of the reference node which suffers from an error $e_{v_1} \sim \mathcal{N}\left(\mu_{v_1}+\mu_{v_2}, \sigma^2_{v_1}\mathcal{F}_{v_1}+\sigma^2_{v_2}\right)$. When v_2 performs least-squares regression in order to estimate the clock of v_0, this estimate suffers from an error err_{v_2} such that

$$err_{v_2} \sim \mathcal{N}\left(\mu_{v_1}+\mu_{v_2}, \left(\sigma^2_{v_1}\mathcal{F}_{v_1}+\sigma^2_{v_2}\right)\mathcal{F}_{v_2}\right). \tag{6.2}$$

With this execution, the clock estimate of v_{n-1} at any time t can be generalized as:

$$err_{v_{n-1}} \sim \mathcal{N}\left(\sum_{i=1}^{n-1}\mu_{v_i}, \sum_{j=1}^{n-1}\left(\sigma^2_{e_{v_j}}\prod_{k=j+1}^{n-1}\mathcal{F}_{v_k}\right)\right). \tag{6.3}$$

The probability distribution above shows that the variance of the clock estimate of the reference node is self-amplificated at each hop $v_i \in V$ by approximately a factor of $\mathcal{F}_{v_i}$. During the execution of FTSP, significant amount of time may pass until a node rebroadcasts its estimate about the clock of the reference node upon receiving a new message. Due to *slow flooding*, FTSP suffers from the large values of $\mathcal{F}_{v_i}$ at each hop (according to Fact 5.1) and hence large estimation errors for especially far-away nodes to the reference. If it is assumed $\forall v_i, v_j \in V : \mathcal{F}_{v_i} \approx \mathcal{F}_{v_j}$, it can be deduced that the synchronization error of FTSP grows *exponentially* with the network diameter, as shown in [7].

In order to prevent the shortcomings of FTSP, PulseSync offers a *rapid flooding* approach. Instead of propagating the estimated clock value of the reference node slowly as in FTSP, nodes executing PulseSync propagate each "pulse" of the reference node as fast as possible. This avoids self-amplification of the estimation errors at each hop and the synchronization error grows with the *square root* of the network diameter. Although rapid flooding reduces the variance of the collected time information collected by the far-away nodes considerably, it does not eliminate the accumulation of variances at each hop. This is a consequence of the nature of flooding. Hence, we can reach the following fact by considering the flooding based time synchronization protocols FTSP and PulseSync.

Fact 6.1. *In flooding based time synchronization, far-away nodes from the reference node collect time information with larger deviations when compared to the nearby nodes to the reference node.*

It is well known that the slope of the least-squares line can easily be affected by the data points with large errors and may exhibit instabilities [10,11]. Thus, Fact 6.1 reveals a fundamental problem for the far-away nodes when these nodes perform least-squares regression on the received time information having large deviations.

Fact 6.2. *In flooding based time synchronization, the slope of the least-squares line of a far-away node from the reference node may exhibit large instabilities and this situation may lead to large estimation errors.*

As we mentioned, the deviation of the collected time information is considerably large especially for far-away nodes in FTSP when compared to those nodes in PulseSync. Hence, the problem introduced by Fact 6.2 is highly critical especially for FTSP.

7. Drift estimation by considering pairwise slope with minimum variance

In this section, we propose a new slope estimation method for linear regression to be used by flooding based time synchronization protocols. This method reduces the instability of the slope of the estimated regression line for the far-away nodes from the reference node when these

nodes collect time information with large deviations. Moreover, the proposed method is more computationally efficient when compared to the slope estimation steps in least-squares.

There exists another way in order to obtain the slope of the least-squares line apart from the equality (5.1). Let $\beta_{i,j}$ denote the slope of the line joining the data points (x_i, Y_i) and (x_j, Y_j), which is given as follows:

$$\beta_{i,j} = \frac{Y_i - Y_j}{x_i - x_j}, \quad 0 \leqslant i < j \leqslant N-1. \tag{7.1}$$

Also let $w_{i,j}$ denote the weight of the pairwise slope $\beta_{i,j}$, which is defined as follows:

$$w_{i,j} = \frac{(x_i - x_j)^2}{\sum_{k=0}^{N-2}\sum_{l=k+1}^{N-1}(x_k - x_l)^2}. \tag{7.2}$$

Using the pairwise slopes and their weights, the slope of the estimated regression line can also be represented as a weighted mean of all paired slopes [23]:

$$\hat{\beta}_{reg} = \sum_{i,j} w_{i,j}\beta_{i,j}. \tag{7.3}$$

In order to reveal how these slopes are affected by the errors of the data points, we now consider their variances. The variance of each pairwise slope β_{ij} of any node $v \in V$ can be written as:

$$Var(\beta_{i,j}) = Var\left(\frac{Y_i - Y_j}{x_i - x_j}\right) = \frac{Var(Y_i - Y_j)}{(x_i - x_j)^2} = \frac{2\sigma_{e_v}^2}{(x_i - x_j)^2}. \tag{7.4}$$

It can be observed from the equality above that, as the difference $x_i - x_j$ gets larger, the variance of the slope gets smaller.

We now propose to consider only the pairwise slope between the points which have the maximum distance among N collected points as the slope of the estimated regression line. By using the oldest and the most recent pairs in the repository, we get the pairwise slope which has the minimum variance among the others. Hence, we define *Pairwise Slope With Minimum Variance* (PSMV) as:

$$\hat{\beta}_{psmv} = \left\{\frac{Y_i - Y_j}{x_i - x_j}\middle| x_i - x_j = \max_{k,l}\{x_k - x_l\}\right\}. \tag{7.5}$$

Fig. 7.1 presents 10 pairwise slopes for five collected data points. Slope 3 in this figure is the PSMV estimator of the slope of the estimated regression line.

Using the PSMV slope estimator, the intercept of the regression line is estimated as follows:

$$\hat{\alpha}_{psmv} = \overline{Y} - \hat{\beta}_{psmv}\bar{x} \tag{7.6}$$

It should be noted that the computation of the intercept $\hat{\alpha}_{psmv}$ is identical to the computation of $\hat{\alpha}_{reg}$ in Eq. (5.2) except for $\hat{\beta}_{psmv}$ in Eq. (7.5) is used instead of $\hat{\beta}_{reg}$ in Eq. (5.1).

7.1. Comparison of least-squares and PSMV slope estimators

PSMV slope is less effected by the large errors of the received data points due to its smaller variance, when compared to the other pairwise slopes. Consider the most recent N data points (x_i, Y_i) such that $x_i < x_{i+1}$ holds for $i = 0, \ldots, N-1$. If it is assumed that there is not any packet loss, it holds due to the periodical information exchange with a period of B time units that

$$E[x_i] = x_0 + iB, \quad i = 1, \ldots, N-1. \tag{7.7}$$

Using this equality, the expected weight of the PSMV slope $\beta_{N-1,0}$ in least-squares can be written using equality (7.3) as follows:

$$\begin{aligned} E[w_{N-1,0}] &= E\left[\frac{(x_{N-1} - x_0)^2}{\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1}(x_i - x_j)^2}\right] \\ &= E\left[\frac{((N-1)B)^2}{\sum_{i=0}^{N-1}(N-i)(iB)^2}\right] = E\left[\frac{12(N-1)}{N^2(N+1)}\right]. \end{aligned} \tag{7.8}$$

Since the typical capacity of the timestamp table of sensor nodes is $N = 8$, we get that

$$E[w_{N-1,0}] \simeq 0.146. \tag{7.9}$$

This result indicates that, in least-squares, the pairwise slope which is less effected by the errors and hence more stable than the other slopes has a weight of approximately 15% in slope calculation. Due to the small contribution of this slope, the slope of the estimated regression line of a far-away node from the reference node calculated by least-squares may exhibit large errors when compared to PSMV slope estimator. Hence, we claim the following:

Claim 7.1. *Since PSMV slope estimator is less effected by time information with large deviations and hence more stable when compared to least-squares slope estimator, it will decrease the estimation errors of the far-away nodes.*

In the following section, we present experimental and simulation results to justify that our claim is consistent with the practice.

8. Experimental work

In this section, we evaluate the performance of the proposed drift estimation method in practice. We focus on the instantaneous global skew and local skew between sensor nodes. We are also interested in *average global skew* which is defined as the instantaneous average of the global skew and *average local skew* which is defined as the instantaneous average of the local skew by considering all nodes.

In order to evaluate the performance of the proposed method, we considered the flooding based time synchronization protocols FTSP and PulseSync. We used the publicly available implementation of FTSP coming with TinyOS 2.1.1. Since PulseSync does not have any publicly available implementation, we implemented it in TinyOS ourselves. We modified standard FTSP and PulseSync in such a way that in addition to the least-squares drift estimation, we integrated PSMV drift estimation for linear regression into these protocols. Apart from the table which stores (*local time-reference time*) pairs for the least-squares, we integrated another table which stores the corresponding pairs for the PSMV method. Hence, a sensor node is able to compute the estimated linear regression line by using both least-squares and PSMV drift estimation methods. Additionally, we enlarged the synchronization messages so that

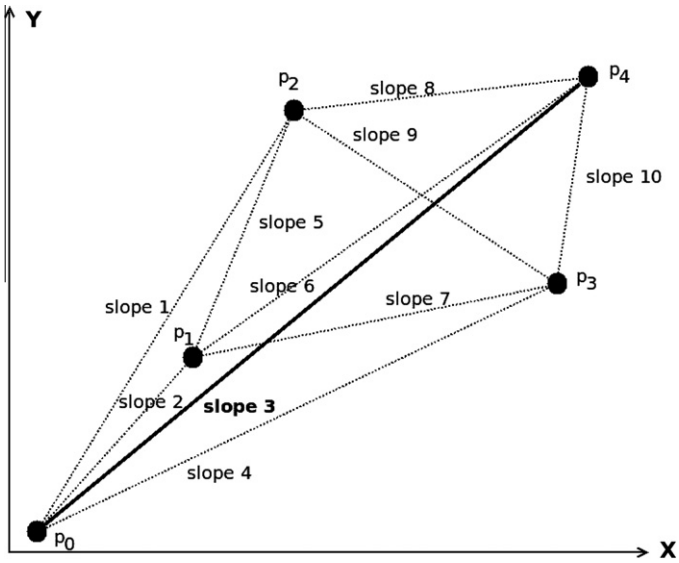

Fig. 7.1. Pairwise slopes for five data points. Slope 3 is the PSMV slope estimator for the slope of the estimated regression line.

they carry the estimates of reference time calculated using least-squares and PSMV drift estimation strategies. In this way, when a synchronization message is received by a sensor node, the least-squares and PSMV based estimates of the reference time carried with the message is stored at the tables which are used to perform least-squares and PSMV based regression, respectively. As a final modification, we added an interface in order for applications to query the estimated reference time calculated with PSMV drift estimation method. With such modifications to FTSP and PulseSync, we could evaluate both of the drift estimation strategies under identical message delays, packet loss rates and environmental conditions that effect the frequency of the crystal oscillators of the sensor nodes.

We executed the modified FTSP and PulseSync protocols on an identical testbed including 20 sensor nodes. We collected estimated clock values from sensor nodes, applied evaluation metrics to the collected data and then analyzed the results.

8.1. Hardware platform

The hardware platform used for the implementation and experiments is MICAz sensor nodes from Memsic.[4] MICAz platform includes low-power 8-bit Atmel Atmega128L micro-controller which has 4 kB RAM and 128 kB program flash memory. The Chipcon CC2420 radio chip provides a 250 kbps data rate at 2.4 GHz frequency.

8.2. Testbed setup

For our tests, we constructed a line topology by configuring each node such that it will accept incoming messages from the nodes with id one more or one less than the id of the that node. Packets from all other nodes are ignored. We have chosen the line topology since the error of flooding based time synchronization grows as the diameter of the network increases and the largest synchronization error is obtained with this topology. Moreover, for many applications a long linear topology need to be used [7].

We used a testbed of 20 sensor nodes which are placed in the communication range of a reference broadcaster node. At the end of each interval which is uniformly distributed between 20 and 23 s, the reference broadcaster transmits a reference packet. The corresponding packet is received approximately at the same time by all nodes. Then, the nodes broadcast their clock values (the estimated time of the leader node) at the reception time of the reference packet. The base station node attached to a PC transfers these messages to the serial port. An application listening the serial port logs these messages.

8.3. Experimental results

We collected the estimated clock values from sensor nodes during approximately an 8 h run for the modified versions of FTSP and PulseSync. For our experiments, the beacon period was 30 s. The experiments have been performed using the tables having a capacity of eight entries. We powered on sensor nodes randomly in the first 3 min. During our experiments, the environmental conditions were quite stable and there were not sudden temperature changes. It took approximately 2000 s until all nodes get synchronized. Thus, we did not take into account the measurement results during this period.

Fig. 8.1 shows the slope values[5] of the estimated regression line calculated by using least-squares and PSMV drift estimation after all nodes get synchronized. It should be noted that node 1 becomes the time reference for the other

[4] http://www.memsic.com.

[5] It should be noted that the values in the Fig. 8.1 represent the slope values from which 1.0 is subtracted.

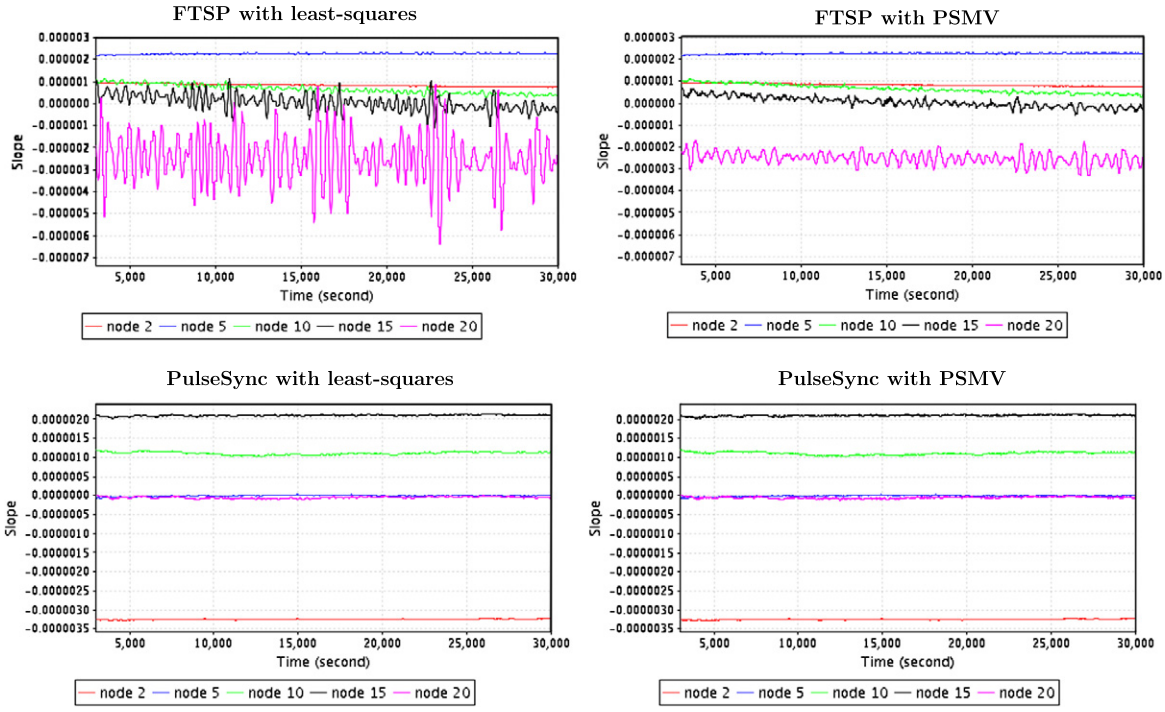

Fig. 8.1. The slope of the estimated regression line (from which 1.0 is subtracted) calculated by FTSP and PulseSync with least-squares (left) and with PSMV (right) on a line topology of 20 MICAz sensor nodes.

nodes in the network since FTSP and PulseSync elect the node with the smallest identifier as the leader. It can be observed from Fig. 8.1 that as the distance from the node 1 gets larger, the collected slope values show quite variability and they are unstable for FTSP with least-squares. On the other hand, the slope values for FTSP with PSMV exhibit less variability most of the time compared to least-squares. Especially, the slope values of the nodes which have large distance from the leader do not exhibit large variations. If node 20 which is executing FTSP with least-squares is considered, the amplitude of the variation of its slope values is much higher than that of the slope values of the other nodes. However, under the same message delay uncertainties, the amplitude of the variation of this node in PSMV slope is much lower. In the previous sections, we theoretically showed that the error of the slope of the estimated linear regression line is amplified at each hop and we claimed that the PSMV method is less affected by the estimates with large errors and hence it is more stable. Experimental results show that our claims are quite consistent with the practice.

It can be observed from the same figure that PulseSync with least-squares reduced the variation of the slope values considerably when compared to FTSP since it employs a rapid flooding approach and reduces the amplification of the estimation errors at each hop. As Fact 6.1 stated, far-away nodes from the reference node still suffer from relatively high slope variations when compared to the nearby nodes to the reference. But these variations are dramatically smaller than that in FTSP. Since the deviation of the time information which far-away nodes collect is quite small, the performances of the slope estimation with least-squares and PSMV are quite comparable in PulseSync.

Figs. 8.2 and 8.3 show the synchronization errors occurred with least-squares and PSMV drift estimation strategies for FTSP and PulseSync. Table 1 summarizes the maximum skew values observed during the experiments. From the figures, it can be observed that the maximum local and maximum global skew values for FTSP with least-squares occurred in the time interval [20,000, 25,000]. As we mentioned previously, the amplitude of the variation in the least-squares slope is much higher and the slope values are quite unstable in this time interval for the far-away nodes. Although the only modification is replacing the drift estimation strategy, PSMV decreased the skew values in this time interval considerably and improved the performance of FTSP by approximately a factor of 4 in terms of maximum global skew. Hence, we conclude that skew values decreased quite considerably due to the slope stability gained by using PSMV method. If we consider PulseSync with least-squares, the skew values decreased drastically when compared to FTSP. Since far-away nodes collect time information with substantially small deviations, the performances of least-squares and PSMV drift estimation methods are quite similar in terms of clock skew for PulseSync.

Fig. 8.4 shows the maximum synchronization error observed between the nodes 2 and 20 and the reference node 1[6] with FTSP and PulseSync protocols. It can be

[6] It should be noted that node 1 is the root of the ad hoc tree which is formed by flooding.

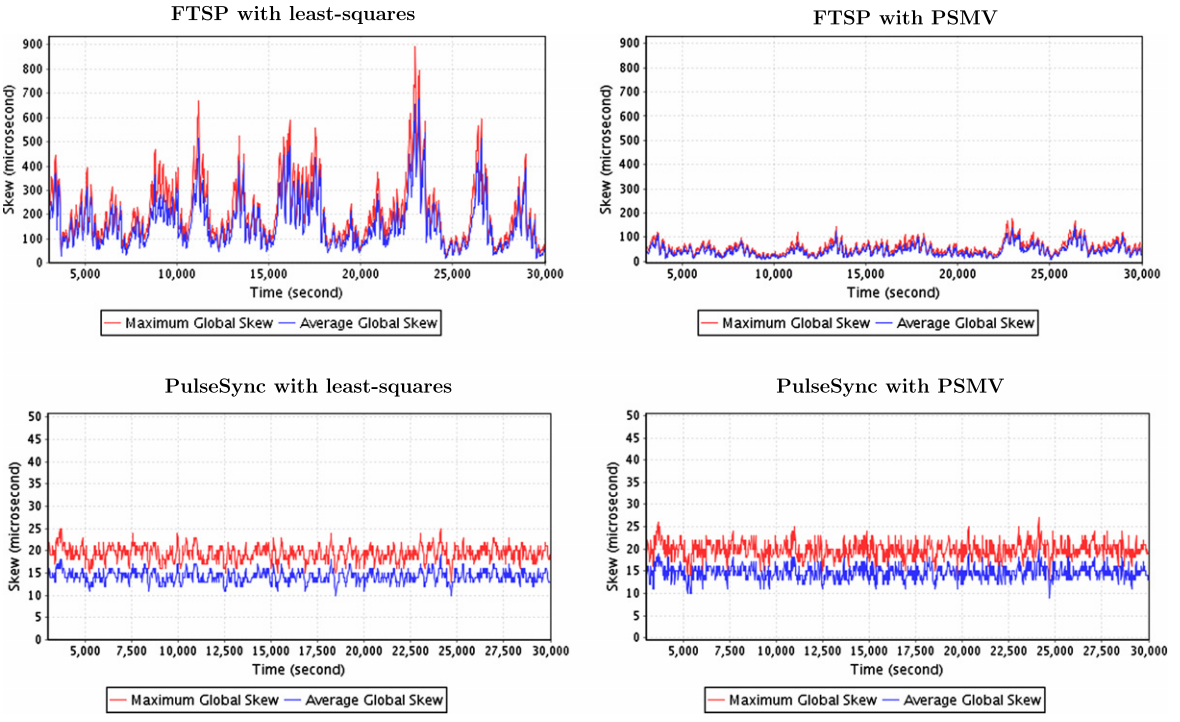

Fig. 8.2. Maximum and average global skews measured for FTSP and PulseSync with least-squares regression (left column) and PSMV (right column) on a line topology of 20 MICAz sensor nodes.

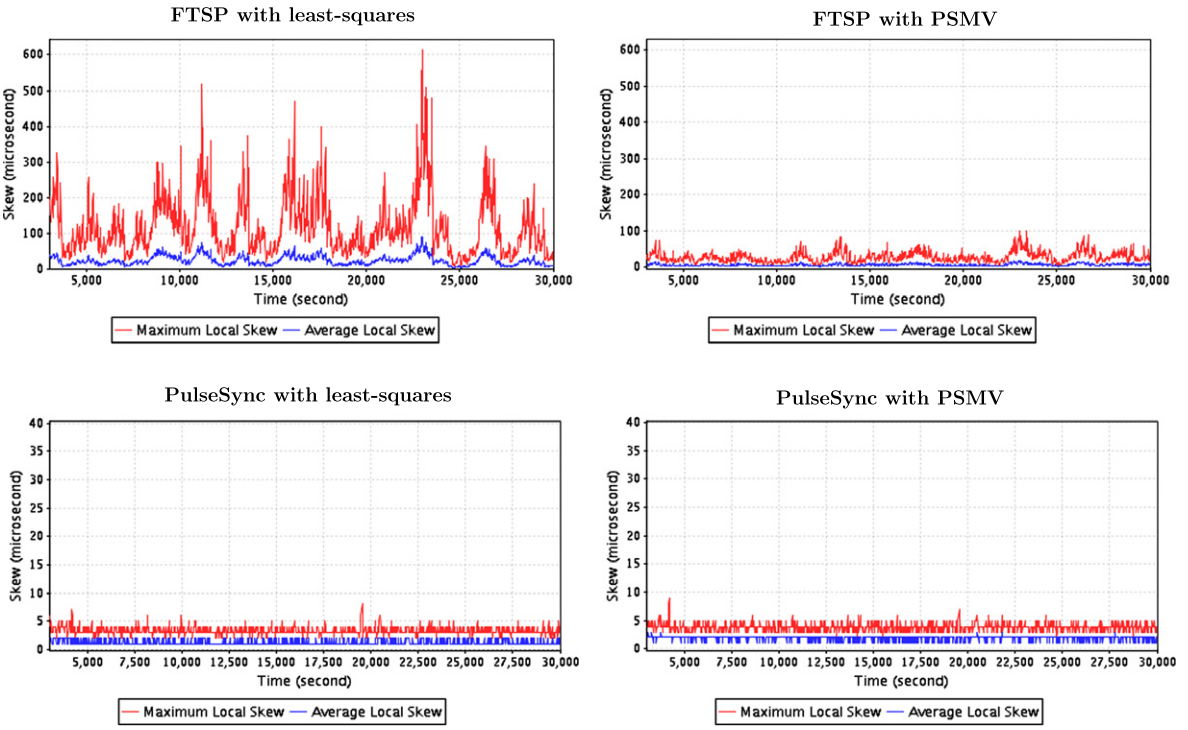

Fig. 8.3. Maximum and average local skews measured for FTSP and PulseSync with least-squares regression (left column) and PSMV (right column) on a line topology of 20 MICAz sensor nodes.

Table 1
Maximum values of the observed skews for FTSP and PulseSync with different drift estimation strategies.

	FTSP		PulseSync	
	L. squares (μs)	PSMV (μs)	L. squares (μs)	PSMV (μs)
Global	892	176	25	27
Avg. global	729	142	19	20
Local	614	101	8	9
Avg. local	91	17	2	3

observed that the error of the nodes 2–8 in FTSP with least-squares is quite comparable to the results with the PSMV approach. As the distance from the root increases, the drift estimation strategy PSMV exhibits its superiority. Especially, the synchronization error of the nodes 14–20 in FTSP with least-squares are quite large when compared to the corresponding values in the PSMV approach. Although the increase of the synchronization error with the increase of the distance to the reference was not prevented, PSMV reduced that effect considerably in FTSP. As can be observed from the same figure, the exponential growth of the synchronization error in FTSP is prevented with PulseSync. Since far-away nodes do not collect time information of the reference node with large deviations in PulseSync, least-squares and PSMV slope estimators performed quite similar in this case as well.

The experimental results show that the PSMV solution for estimating the slope of the linear regression line is less affected by the amplified errors. We are also interested in the computational complexity of $\hat{\beta}_{psmv}$ and compare it with $\hat{\beta}_{reg}$. The computation of $\hat{\beta}_{psmv}$ requires two subtraction and one division operation, as shown in equality (7.5). On the other hand, $\hat{\beta}_{reg}$ requires $6N + 1$[7] arithmetic operations. Each data point x_i and Y_i is represented by 32 bit unsigned integers in real sensor platforms. In the computation of $\hat{\beta}_{reg}$, more than 32 bits are required to represent the results of the multiplications $(x_i - \bar{x})(Y_i - \overline{Y})$ and $(x_i - \bar{x})^2$. Thus, for the numerator and the denominator of the fraction in the Eq. (5.1), it is required to perform 64 bit mathematical operations. If we consider the computation of $\hat{\beta}_{psmv}$, $(Y_{i+1} - Y_i)$ and $(x_{i+1} - x_i)$ are 32 bit integers. Thus, 64 bit operations are eliminated if we consider Eq. (7.5). The CPU overhead of performing 64 bit arithmetic operations on a real sensor platform is much higher than that of performing 32 bit arithmetic operations. Using these facts, we argue that the CPU overhead of the computation of $\hat{\beta}_{psmv}$ is much lower than that of $\hat{\beta}_{reg}$. Our experimental findings justify this argument, as it is expected. The computation of the least-squares spent a maximum value of 5394 μs CPU time during our experiments. However, PSMV drift estimation strategy reduced this overhead to a maximum value of 3145 μs CPU time.

In summary, our experiments showed that PSMV drift estimation strategy improved the performance of FTSP by approximately a factor of 4 and preserved the performance of PulseSync on a line topology of 20 sensor nodes. Moreover, PSMV improved both FTSP and PulseSync in terms of computation and reduced the CPU overhead of linear regression by approximately 40%.

9. Simulations

In addition to the experiments performed on the actual testbed, we implemented FTSP and PulseSync with least-squares and PSMV strategies in our WSN simulator, which we implemented using the Java programming language. Our simulations gained us impression on how least-squares and PSMV drift estimation perform on large and dense networks. During our simulations, we implemented the hardware clocks of nodes in software with a random drift of ± 50 ppm. We modeled the variances in the message delay with a normally distributed random variable. We constructed different networks, performed 10 simulation runs for each network and averaged the calculated synchronization errors for these runs.

Fig. 9.1 shows the simulation results on networks of line topology which have different diameters. It can be concluded from these results that the synchronization error of FTSP with PSMV grows substantially slowly than that with least-squares as the diameter of the network grows. On the other hand, PSMV slope estimator preserves the synchronization accuracy of PulseSync on longer networks as well.

On the line topology, the messages originated from the reference node follow the same path to reach the node at the end of the network since each node has at most two neighbors (one on the left and one on the right). In order to observe the behavior of FTSP and PulseSync with PSMV on networks in which messages may follow many alternative paths, we also performed simulations on networks of grid topology which have the same diameters as the line topologies. Fig. 9.2 shows the results of these simulations. If we consider 21×21 grid topology,[8] the neighborhood cardinality of the nodes is four times the cardinality of the line topology with 40 nodes and there may be more collisions. On the contrary to the line topology, the synchronization messages of the reference node may follow many paths in order to reach to the far-away nodes on the grid topology. Hence, far-away nodes on the grid topology may collect time information with smaller error when compared to that on the line topology. Our simulations showed that FTSP with PSMV slope estimation strategy preserved its superiority over least-squares on the grid topology as well as on the line topology. Similar to the line topology, PulseSync with PSMV performed nearly the same when compared to that with least-squares on the grid topology.

We also performed simulations in order to observe the performance of PSMV drift estimation on dense networks. We considered three different networks which consist of 100 sensor nodes and have average neighborhood cardinality of 6, 12 and 24, respectively. Due to their high density, these networks may introduce quite a few packet collisions when compared to line and grid topologies. On the other hand, nodes in these networks may collect time

[7] $2N$ subtraction, $2N$ addition, $2N$ multiplication and 1 division.

[8] The diameter of this topology is 40.

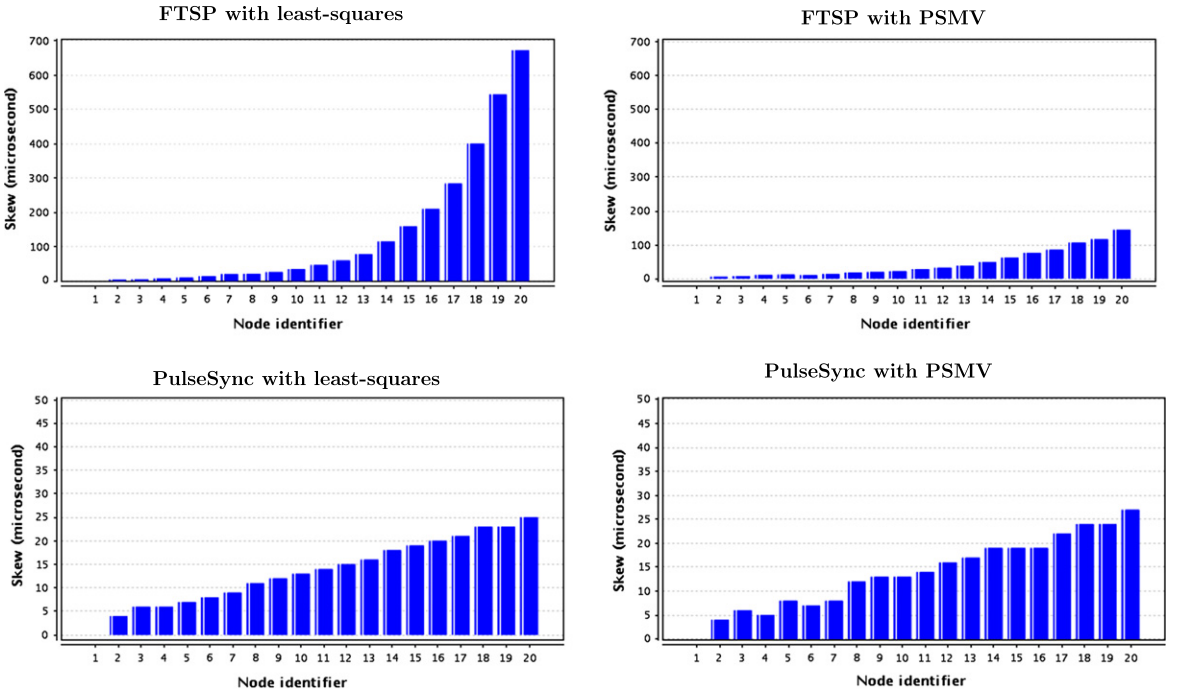

Fig. 8.4. The maximum synchronization error versus distance from the leader node 1 (root) for FTSP and PulseSync with least-squares regression (left) and PSMV (right).

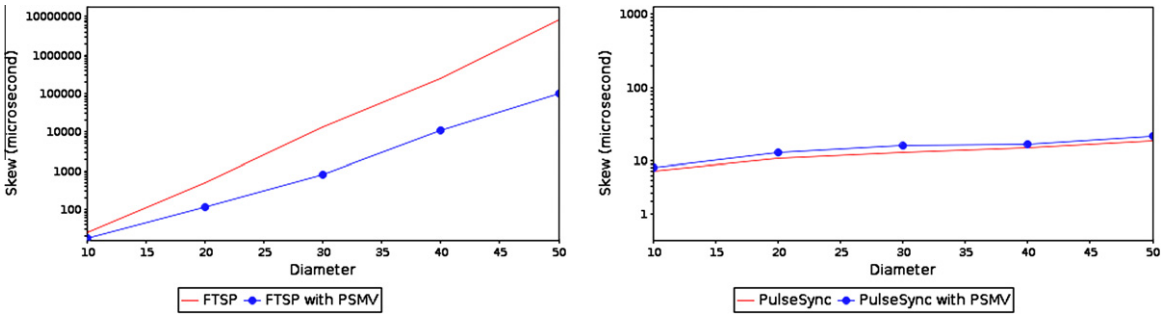

Fig. 9.1. The simulation results for FTSP and PulseSync with least-squares and PSMV drift estimation on the line topology.

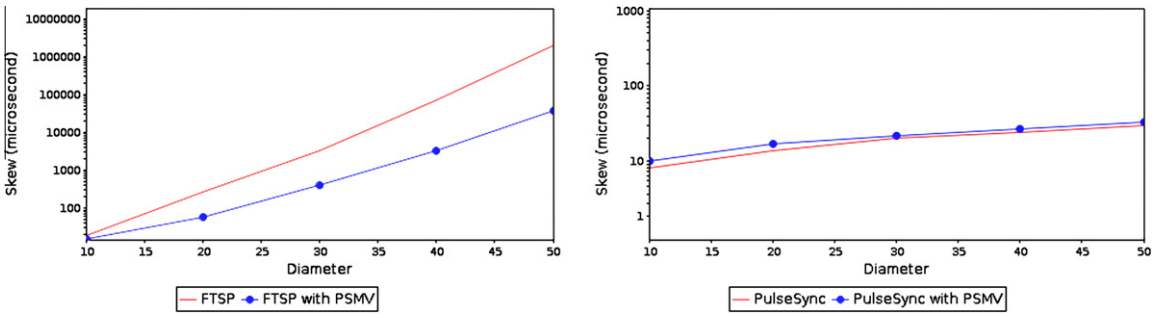

Fig. 9.2. The simulation results for FTSP and PulseSync with least-squares and PSMV drift estimation on the grid topology.

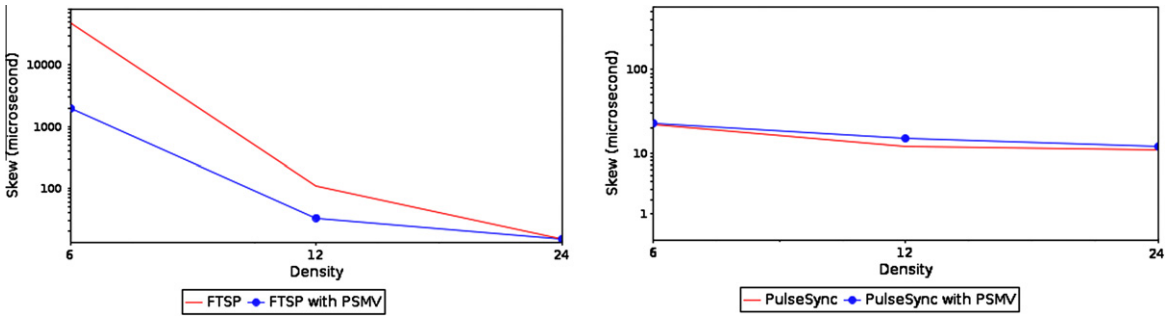

Fig. 9.3. The simulation results for FTSP and PulseSync with least-squares and PSMV drift estimation for different neighborhood densities.

information with smaller errors since there are much more alternative paths and the synchronization messages may follow shorter paths to reach far-away nodes as the density of the network increases. As can be observed from Fig. 9.3 which presents the results of our simulations, PSMV improved the synchronization accuracy of FTSP and preserved the synchronization accuracy of PulseSync on dense networks.

As a summary of our simulations, it can be concluded that PSMV improves the performance of FTSP considerably and preserves the synchronization accuracy of PulseSync with lower computational overhead on longer, larger and denser networks as well.

10. Conclusion and future work

In this study, we argued that the method of least-squares for linear regression may exhibit an undesired performance for flooding based time synchronization protocols in WSNs when the deviation of the received time information is large. We showed that throughout the execution of these protocols, far-away nodes may collect estimates from their neighbors with large errors, which leads to large instabilities in the slope of the least-squares regression line. Due to the poor performance of least-squares, the global time estimates of these nodes exhibit substantially large synchronization error. This situation introduces large skew between the reference node and the far-away nodes and decreases the scalability of these protocols.

Alternatively, we proposed a more stable and computationally efficient slope estimation method for linear regression, namely Pairwise Slope With Minimum Variance (PSMV), which considers only the pairwise slope between the earliest and the most recently collected data points. In order to show its superiority in practice, we incorporated PSMV method into FTSP and PulseSync protocols. Our experimental results showed that the effective implementation of PSMV improves the synchronization quality and hence the scalability of FTSP considerably. A desirable property of the proposed drift estimation method is that it decreases the computation overhead of linear regression, which is also a significant improvement for the sensor nodes which have limited processing capacity. We also showed through our experiments that PSMV preserves the synchronization accuracy of PulseSync with significantly lower computational overhead.

As we stated, PulseSync outperforms FTSP drastically in terms of synchronization error by employing rapid-flooding approach. However, rapid-flooding in WSNs can also be slow due to neighborhood contention since the nodes cannot propagate the flood until their neighbors have finished their transmissions [15,12]. In order to propagate the flood as quickly as possible, the transmissions of the sensor nodes must be scheduled [7]. Due to these disadvantages of PulseSync, FTSP may be more preferable in practice. Therefore, we think that the improved performance of FTSP with PSMV drift estimation is quite significant for time synchronization in WSNs.

As a conclusion, we believe that the proposed drift estimation method is useful to improve the performance, robustness and energy efficiency of time synchronization protocols especially the ones which use flooding in WSNs.

Acknowledgments

Kasım Sinan YILDIRIM acknowledges The Turkish Scientific and Technical Research Council (TÜBİTAK) for supporting this work through a domestic PhD scholarship program (BAYG-2211). Special thanks to Efendi Nasibov for his valuable comments to improve this study.

References

[1] F. Sivrikaya, B. Yener, Time synchronization in sensor networks: a survey, Network, IEEE 18 (4) (2004) 45–50, http://dx.doi.org/10.1109/MNET.2004.1316761.
[2] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330. http://dx.doi.org/10.1016/j.comnet.2008.04.002.
[3] I.F. Akyildiz, M.C. Vuran, Wireless Sensor Networks, John Wiley & Sons, 2010.
[4] M. Maróti, B. Kusy, G. Simon, A. Lédeczi, The flooding time synchronization protocol, in: SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, 2004, pp. 39–49. http://doi.acm.org/10.1145/1031495.1031501.
[5] 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. J. Ad Hoc Ubiquitous Comput. 1 (4) (2006) 239–251. http://dx.doi.org/10.1504/IJAHUC.2006.010505.
[6] P. Sommer, R. Wattenhofer, Gradient clock synchronization in wireless sensor networks, in: 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), San Francisco, USA, 2009.

[7] C. Lenzen, P. Sommer, R. Wattenhofer, Optimal Clock Synchronization in Networks, in: 7th ACM Conference on Embedded Networked Sensor Systems (SenSys), Berkeley, California, USA, 2009.
[8] T. Schmid, Z. Charbiwala, R. Shea, M. Srivastava, Temperature compensated time synchronization, Embedded Systems Letters, IEEE 1 (2) (2009) 37–41, http://dx.doi.org/10.1109/LES.2009.2028103.
[9] T. Roosta, S. Sastry, Securing Flooding Time Synchronization Protocol in Sensor Networks, Tech. rep., 2007.
[10] P.S.S.S. Hussain, Non-parametric regression, J. R. Statist. Soc., Series A (General) 146 (2) (1983) 182–191. http://www.jstor.org/stable/2982016.
[11] R.R. Wilcox, Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy, second ed., Springer, New York, 2010. http://www.springerlink.com/content/978-1-4419-5524-1.
[12] T. Schmid, Z. Charbiwala, Z. Anagnostopoulou, M.B. Srivastava, P. Dutta, A case against routing-integrated time synchronization, in: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10, ACM, New York, NY, USA, 2010, pp. 267–280. http://doi.acm.org/10.1145/1869983.1870010.
[13] J. Elson, L. Girod, D. Estrin, Fine-grained network time synchronization using reference broadcasts, SIGOPS Oper. Syst. Rev. 36 (SI) (2002) 147–163. http://doi.acm.org/10.1145/844128.844143.
[14] T. Schmid, R. Shea, Z. Charbiwala, J. Friedman, M.B. Srivastava, Y.H. Cho, On the interaction of clocks, power, and synchronization in duty-cycled embedded sensor nodes, ACM Trans. Sen. Netw. 7 (2010) 24:1–24:19. http://doi.acm.org/10.1145/1807048.1807053.
[15] J. Lu, K. Whitehouse, Flash flooding: exploiting the capture effect for rapid flooding in wireless sensor networks, in: IEEE Infocom 2009 – IEEE Conference on Computer Communications, vols. 1–5, IEEE Infocom, IEEE, IEEE, 2009, pp. 2491–2499, IEEE INFOCOM Conference 2009, Rio de Janeiro, Brazil, April 19–25, 2009.
[16] S. Ganeriwal, R. Kumar, M.B. Srivastava, Timing-sync protocol for sensor networks, in: SenSys '03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, 2003, pp. 138–149. http://doi.acm.org/10.1145/958491.958508.
[17] K. Römer, P. Blum, L. Meier, Time synchronization and calibration in wireless sensor networks, in: I. Stojmenovic (Ed.), Handbook of Sensor Networks: Algorithms and Architectures, John Wiley & Sons, 2005, pp. 199–237.
[18] M. Maroti, J. Sallai, Packet-level time synchronization, Technical report, TinyOS Core Working Group, May 2008 <http://www.tinyos.net/tinyos-2.x/doc/pdf/tep133.pdf>.
[19] J. Sallai, B. Kusy, A. Ledeczi, P. Dutta, On the scalability of routing integrated time synchronization, in: 3rd European Workshop on Wireless Sensor Networks (EWSN 2006).
[20] K. lae Noh, E. Serpedin, K. Qaraqe, A new approach for time synchronization in wireless sensor networks: pairwise broadcast synchronization, IEEE Transactions on Wireless Communications 7 (9) (2008) 3318–3322, http://dx.doi.org/10.1109/TWC.2008.070343.
[21] Y.-C. Wu, Q. Chaudhari, E. Serpedin, Clock synchronization of wireless sensor networks, Signal Processing Magazine, IEEE 28 (1) (2011) 124–138, http://dx.doi.org/10.1109/MSP.2010.938757.
[22] S.M. Ross, Introduction to Probability and Statistics for Engineers and Scientists, third ed., Elsevier Academic Press, 2004.
[23] A. Gelman, D. Park, Splitting a predictor at the upper quarter or third and the lower quarter or third, The American Statistician 63 (1) (2009) 1–8. doi:http://dx.doi.org/10.1198/tast.2009.0001.

Kasım Sinan Yıldırım received the B.Eng., M.Sc. and Ph.D. degrees from Ege University, İzmir, Turkey in 2003, 2006 and 2012 respectively. He works as a research assistant at the Department of Computer Engineering at Ege University since 2007. His research interests are embedded systems, distributed systems, distributed algorithms and wireless sensor networks.

Aylin Kantarcı received the B.Sc., M.Sc. and Ph.D. degrees from Ege University, İzmir, Turkey, in 1992, 1994 and 2000, respectively. She is an associate professor at the Department of Computer Engineering at Ege University. Her current research issues include distributed systems, wireless sensor networks, and multimedia.